



A User's Guide to Authoring a Bluray Disc

2nd edition

based on BDS Standard and BDS MX V4.6 (.0.2089)

Theo de Klerk

.

Acknowledgements:

Thanks to Anna Schobers for allowing the use of her holiday movie of Sydney as exercise material.

Thanks to native speakers Adam Langham and Maritza Brikisak for checking on my (Dutch-)English spelling and grammar in a first edition. Maritza also supplied a solution to add a thumbnail to the disc when played on PlayStation 4.

Thanks to Alexey Kolesnikov from Blu-Disc Studio for answering all my questions and solving my problems with the product along the way.

The 2nd edition reflects some changes in the windows shown by BDS and some added or modified behaviour of V4.4 or later. With any new edition, some menus may change without notice – although I try to keep up with all changes, don't be alarmed if a certain menu looks slightly different than shown in this guide.

The author of this user's guide is not associated with the makers of Blu-Disc Studio – DVD Logic Software or Disc Art Authoring, in any way other than being a mere user of the BDS MX product who thinks more people should use it and appreciate its capabilities.

If by reading this guide, you find inconsistencies or errors, please report them to the author so they can be corrected for the next (online) edition of the guide: theo.de.klerk@gmx.com. Of course praise is also welcomed.

Initial release: 2019-10-01

Date last modification: 2023-12-30

Table of Contents

Part 1: The Setup	5
Introduction to Blu Disc Studio Standard and MX (Pro)	7
Getting started: Blu Disc Studio (BDS) requirements	16
Initial BDS configuration	27
Using Blu Disc Studio and customizing it.....	38
Part 2: Basic operations	47
Project 1: Create the simplest bluray	49
Project 2: Creating a simple menu-driven bluray disc.....	76
Project 3: Creating a bluray disc with setup menu for audio and subtitles	125
Project 4: Creating a bluray disc with popup menu	146
Project 5: A Setting menu as popup and a chapter menu as carousel popup	156
Summary of basic steps to author a bluray disc.....	194
Part 3: Special Operations (<i>not of the Putin kind</i>)	197
Interlude: Chapter marks – standard or direct editing.....	199
Interlude: Multi-action and Switch actions	203
Interlude: JAR title files with menu or movie	207
Project 6: Start movie when no menu choice is made	219
Project 7: Restartable disc (Restore/Resume)	225
Project 8: Bookmarks	234
D.I.Y. project: changing submenus using Switch and Multi-Action	244
Part 4: Animations	247
Project 8: Animations.....	249
Part 5: Advanced Operations and programming.....	285
Using Registers (simple programming)	287
Project 9: Easter Eggs.....	293
Project 10: watching episodes without titles	296
The Reverse: cutting up a long movie into parts.....	307
Programming in Java.....	310
Project 11: Remove all bookmarks.....	315
User defined variables and functions.....	317
Project 12: Add a popup Timeline menu.....	327
Project 13: Popup Timeline with bookmarks	332
Project 14: Playing an A-B loop	347
Jump between movies (from feature to making of and vice versa)	350

Part 6: MX only mostly (Slideshows and PIP)	373
Slideshows	375
Project 15: Create a BDS MX slideshow	381
Picture in Picture.....	383
Project 16: Create a BDS MX picture-in-picture (simple)	393
Real PIP example: Big Buck Bunny	398
Part 7: 3-D	403
3D – in 2D and true 3D flavour.....	405
Mixing 2D and 3D.....	408
True 3D movies (mvc files).....	409
Flat 3D: Side-By-Side (SBS) and OU (Over-Under/Top-Bottom/Above-Below)	419
3D images	430
Part 8: The Rest.....	443
What’s more...	445
A final word.....	459
Appendices	461
Appendix A: The user’s guide source files.....	463
Appendix B: Commercial settings	465
Appendix C: Bluray specifications	469
Appendix D: Errors occurred: possible causes	483
Appendix E: External Tools.....	496
Appendix F: Java programming.....	504
Appendix G: Dockable windows.....	520
Appendix H: Required disc space	525
Appendix I: Checklist.....	526
Appendix J The Project file (bdmd file)	529
Index	531

Part 1: The Setup

Introduction to Blu Disc Studio Standard and MX (Pro)

There are not as many bluray authoring programs as there are DVD authoring programs. This may be due to the fact that following the VHS boom the DVD disc was a huge step forward in quality and finally amateurs were able to author their own movies in non-linear fashion on computer and create their own DVD discs.

Bluray discs came later and although giving a noticeable improvement in picture quality from SD (standard definition) 720x576 pixels to HD (high definition) 1920x1080 pixels (more than twice better in horizontal and vertical resolution), it so far did not completely replace the DVD and its authoring. The upcoming online streaming capability may also in part be a reason. Amateur cameras recording in HD often come with an application allowing HD recordings on DVD media – usually in interlaced (AVC)HD format and with only 30 minutes playing time on a single DVD disc. You could even edit those on computer and create a new DVD disc that was bluray player compatible.

For anyone who wants to store films of more than 30 minutes on a single disc, the bluray disc is the only solution. Rather than 4 or 8 GB storage or a (single/double layered) DVD, you now have 22 and 50 GB (single/double layered) bluray discs – five to ten times as much as a DVD. Even 100 GB and 200 GB discs have been made (but are expensive). Bluray discs allows for more (and/or longer) movies at the higher bitrates needed for HD recordings.

The standards for creating a bluray disc that is acceptable to stand-alone set up players have been laid down by the Bluray Disc Association (BDA) and any authoring program must obey these if it is to produce discs that can be given away or sold commercially with a guarantee they will play on a home set up bluray player¹.

Most of the few home-use bluray authoring programs fall short on many features a user may want. Many will transcode all movies, required or not, to their own format. In doing so the movies occasionally become larger than the original. In the transcoding, some movies are scaled up from lower resolutions (720x1280 pixels, even the PAL DVD standard SD resolution of 576x720 pixels (480 x 720 pixels for NTSC)) to blown-up full HD resolution. The picture doesn't become better but the movie does grow in size. Most of these programs are of American origin and allow only a single audio track (everyone speaks English, no?) and no subtitles or only a single one burned-in on the picture image. Often you need to use a fixed menu layout (flexible to the extent of allowing to use your own background image).

Enter the one product that doesn't do any of it and allows total freedom to the user: the Ukrainian based Blu-Disc Studio. It comes at a price: total freedom means no decisions are taken for you and you need to do everything yourself. This may sound a burden (and initially it is) but once the workflow has become familiar to you, it works like a breeze. With results that are exactly as you want them to be. And you

¹ See Appendix C: Bluray specifications

need to select a disc burning application as it produces a disc image but cannot burn a disc itself.

What version do I need?

Blue-Disc Studio comes in a number of flavours and you need to select the one best suited to your needs:

- Blu-Disc Studio Lite – a free version that everyone should try out first. Its release is frozen on version 1.0.10 and will not be further developed. It is in fact BDS V1 and became “free” when BDS V2 came out.
The BDS Lite edition is no longer maintained (“obsolete”) or corrected. It runs on both Windows 32 bits and 64 bits systems.
Meanwhile BDS evolved into V4 at the time of writing. The interface of BDS V4 compared to “BDS Lite” differs more and more. As does the added functionality.
“Lite” suggests it may be useless for real use, but it is not. You can author as many as 32 movies with 4 audio tracks and 4 subtitle tracks and make menus the way you want. It uses an external muxer (such as tsMuxer or Scenarist) to compile the discs. Some people may never need more. A separate user’s guide for this version as well as a YouTube quick starter’s movie are available. They can be downloaded (<https://blu-disc.net/downloads> – near the bottom of the window) or seen (<https://blu-disc.net/guides>) from the Blu-Disc Studio website.
To overcome its restrictions you need to upgrade to one of the paid BDS editions.

The commercial, paid-for versions only run on Windows 64 bit as of BDS V4.1:

- Blu-Disc Studio (Standard) – removes the limits of the Lite version: unlimited number of movies (except the trial editions), 32 audio tracks, 32 subtitle tracks (the maximum number specified by the BDA). In addition, you have more interactivity in menus such as play lists (“play all”, “play episode”). It also includes menu and button animation where the menu changes itself as a small movie or pops up certain parts. And those using Scenarist as authoring tool: it can import from and export projects to this product. It can also use an external muxer (tsMuxer). It makes a set top player remember where you stopped a movie so you can continue from there (“resume”). And you can author 3-D movies although 3-D support recently seems dropped again from many set top players and television or video projector equipment.
- Blu-Disc Studio MX – first step up from “Standard”. Basically, it does what Standard does, but it has its own media muxer to allow PIP (picture in picture) and slideshows – ways never envisioned or supported by tsMuxer used in “BDS Lite” or

“BDS Standard”. This internal muxer is quite picky on file conformance to BDA (bluray BDAV) standards. Several home user video editing tools (such as Cyberlink’s Power Director 16) provide MPEG-TS streams as .m2ts and not BDA MPEG4-AVC streams and will therefore give problems when you use the internal muxer. The external (freeware) muxer tsMuxer happily accepts both. Except if you have PIP or slideshow elements on your disc. Using a non-official format like MPEG-TS will work fine on most players, but some may object. Transcode to proper AVC to avoid this problem.

- Blu-Disc Studio MX-Pro – identical to the MX version, but it can also do CMF (Cutting Master Format) output. CMF allows for region coding and CSS encryption to be added to the master image to be sent to a disc duplicator. They “stamp” blurays rather than burn them and this requires a CMF master rather than the \Output burn-masters BDS delivers in BDS Standard and MX. Something interesting for companies making commercial blockbuster movies on bluray, not for others that burn their discs. But if you don’t know what CMF is, you won’t need it.

In addition, the Blu-Disc Studio team now also have 4K authoring programs available (Blu-Disc UHD) for the next generation of video recording. This guide limits itself to the bluray HD versions.

The company also sells “re-engineering” tools like BR-Reauthor (https://dvd-logic.com/bd_reauthor) . If you need to rebuild an existing bluray disc but haven’t got the original authoring files anymore, these re-engineering tools can recreate most or all of them, giving you a head start to modify the contents without having to recreate all elements of the original disc yourself

There are two websites that are “home” to the BDS products:

- https://dvd-logic.com/blu_disc_studio (the www.bluediscstudio.com redirects to this location) – DVD Logic is the company specializing in bluray and 4K software. As part of their site you also find details on Blue Disc Studio products as you would on <http://www.blu-disc.net>
- <https://www.blu-disc.net/> – the original home of BDS and a subset of the DVD Logic site

Both sites refer to various downloadable (trial) products or sample files as well as various video tutorials that you may find useful to look at once you get the hang of the product. For example, for inspiration on menus and their animations see <https://blu-disc.net/videos> .

Because as of BDS V4.1 (subversion 1654) the application only runs on 64-bits Windows which means it installs in the \Program Files folder.

Version comparison

The Blu-Disc Studio website gives a quick overview of the many features supported by the various functions. Although a summary is

provided in the previous section, here is a more extensive list for completeness sake.

As a rule of the thumb you might say that:

- BDS Lite suffices for all home users that do not require elaborate menu operations and can live with a maximum of 32 movies and 4 audio and 4 subtitle tracks. It requires an external muxer, tsMuxer.
- BDS Standard suffices for home and professional users that want to lift the limitations of Lite (allow animation and playlists) but do not need PIP or slideshows. It also requires an external muxer, tsMuxer.
- BDS MX suffices for home and professional users that do require PIP or slideshows. Their video files must conform strictly to BDA bluray standards as the internal muxer is quite insistent on this. The use of tsMuxer is also supported although this muxer does not support PIP or slideshows.
- BDS MX Pro is the next step up from MX. It is needed only for professional authors using disc duplicating manufacturers using CMF to stamp their discs and disc encryption.

Feature	Lite	Standard	MX	MX Pro
Runs on Windows 32 bit systems	yes	no	no	no
Ability to create main menus, popup menus and movies	Max 32	Unlimited	Unlimited	Unlimited
Maximum audio tracks per movie	4	32	32	32
Maximum subtitle tracks per movie	4	32	32	32
Menu designer with the ability to use the full-color images (PNG 32-bit)	X	X	X	X
Import PSD-files	X	X	X	X
Button drawing modes	1	8	8	8
Resizing menu objects (in Designer Window): scale text, rectangles		X	X	X
Ability to create scenes (play marks)	X	X	X	X
Automatically create a scene selection menu with presets	X	X	X	X
Ability to assign actions to up/down/left/right and Enter (OK)	X	X	X	X
Ability to assign actions to a chapter (reaching a play mark)	X	X	X	X
Auto-assignment of transitions between the buttons	X	X	X	X
Assign sounds to buttons	X	X	X	X
WAV mixer (assign mono tracks to left/right/center) and WAV file fixer		X	X	X
Highlight the current audio and subtitle tracks	X	X	X	X
Highlight the current chapter	X	X	X	X
Multiple actions at once (Multi-action)	X	X	X	X
Menu simulation (preview of the menu)	X	X	X	X
Mux project into the bluray compliant folder structure using tsMuxer	x	x	x	X
Playlists support		X	X	X
Ability to use animation in actions		X	X	X
Ability to use animation at the start of the menu		X	X	X
Permanent animation with several images		X	X	X
Intro movie for the menu		X	X	X
Automatically close the popup menu		X	X	X
Ability to assign action to Top Menu button		X	X	X
Conditional actions (SWITCH)		X	X	X
Ability to use GPR-registers		X	X	X
JAR signing		X	X	X
Ability to store images outside the JAR		X	X	X
Resume movie feature		X	X	X
Resume disc playback		X	X	X
Ability to write Java-code (script) in actions and when reaching a chapter in a movie		X	X	X

Feature	Lite	Standard	MX	MX Pro
Compilation of 3D bluray (MVC + 2D menu)		X	X	X
Export project to the Scenarist® BD		X	X	X
Ability to use fonts and text to create menu objects (static, buttons)		X	X	X
Ability to scale video		X	X	X
Bluray Audio and Pure Audio® wizards		X	X	X
Support for templates and project merging		X	X	X
User Defined Functions and Advanced Parameters		X	X	X
Muxing project using internal muxer			X	X
Carousel menu wizard for any menu and popup menus		X	X	X
Chapter menu generation wizard	X	X	X	X
Carousel chapter menu generation wizard		X	X	X
Chapter image generation or “do not scan” (keep assigned images)		X	X	X
CMF output support				X
Technical support		X	X	X

Updates, bug fixes and changes occur regularly on the BDS productline. For a summary of changes in any new release, visit the BDS website at <https://blu-disc.net/news>.

Flexibility at a (labour) cost

The flexibility of Blu Disc Studio comes at a price. It is based on a “construction” model where you need to think through the bluray disc elements and creation process for the disc you want to make. Blu Disc Studio only puts together the movies and menus in a bluray specification (as defined by the Bluray Disc Association – BDA) compliant manner. It assists you in making menus and navigation, it does nothing for you in editing the actual movies.

If the movies are not compliant, the disc will be unplayable.

The following two error situations may occur using movie files:

- proper files have the wrong file extension: those files are not listed in navigation windows and cannot be selected in BDS even if they are compliant.
- improper (non-compliant) files but with the recognizable file extension: those files will be processed by the used muxer for BDS. Their incorrect nature may not show until much later – possibly only when you try to play the finalized disc.

Many external tools, both free or commercial, exist to ensure your video, audio, menu and subtitle files comply. See section “Appendix E: External Tools” at the end of this user’s guide for an incomplete list of possible contenders.

Each bluray project consists of two elements that must be fully planned and created:

- menu and menu navigation
- movies

Both elements can be defined independently. Setting up menu navigation is done whilst the movies are only present through movie placeholders.

- menu navigation
 - items such as background videos or stills, menus, popup menus, buttons, sounds, animated backgrounds
 - actions on items: what happens if a button is clicked, an item selected, navigation between buttons, a new menu called for, animation, playing order

This part of the project creates Java code that is stored in a Java archive (JAR) by BDS. It can run independently of the creation of the movies, which is a much more time-consuming process.

You can simulate the navigation (to find errors) by itself and correct it before starting the time-consuming phase of muxing the movie assets.

If the muxing has already been done and the navigation is modified, it only requires to rebuild and replace the JAR file.

- movie assets
 - video and audio tracks must be provided separately (demuxed) in bluray specification compliant formats, not as a combined movie-with-sound. Non-compliant formats are not accepted and need to be fixed by external tools.
 - Subtitles are provided in text based .srt or image based .sup format. Other formats need to be transcoded in one of these two formats using external tools.

The coverage by this user's guide

This user's guide covers the working of the Blu-Disc Studio Standard and MX versions. The MX-unique features (PIP and slideshows) are covered in Part 6 by this user's guide.

The author uses the BDS MX version so screen shots may show this name – but BDS Standard should look no different. Between the start of the guide and its completion, BDS went through some minor updates that sometimes changed the exact wording in menus. What you see in this guide is the way it looked when a project was executed.

The user's guide will show how to make a simple disc and elaborate from there by adding additional features. Not every nook and cranny of the product is shown: it has too many and they can be used in many different ways. But showing you how the basic product works hopefully encourages you to explore the other features or familiar features to let the disc do exactly what you want it to do.

A piece of advice. Re-reading what I wrote in this guide, I can visualize what happens. And know when and how such a feature is used. Because I've done all those steps. Learnt from the failures of doing it wrong.

Don't use this guide to just read about BDS – the knowledge will only sink in when you actively work with the product. And learn by your mistakes. Executing the projects in this guide may be a good starting

point. Or your own projects. But work with BDS and notice how your knowledge about the product grows to the point where you become independent of manuals and user guides. You don't learn how to swim from a book – even with video tutorials attached. You need to hit the water and feel for yourself what works and what doesn't to stay afloat and swim ahead. I've got loads of "Learn XXX in a Day"-type books. But none have helped me by just reading them. Use product XXX and work with it. Usually it takes (far) more than a (marketing) day to feel confident about and comfortable with product XXX. But once you passed the top of the hill of what is known as "the learning curve", you know the product has grown on you – becomes "intuitive" – even if it is not.

If you want to perform the steps of the various projects yourself, the projects are available for download as specified in Appendix A: The user's guide source files.

This user's guide is divided in several parts.

- Part 1 explains how to setup the BDS software and to "personalize" it in part for your usage.
- Part 2 introduces you to the basic functions of BDS to create a bluray disc with menus. This part has 4 "projects" of increasing complexity. The steps you need to take are explained in detail.
- Part 3, 4, 5 describes the more complex functions of BDS – not all of which you will need or use. The steps describing the projects in these parts are more cursory and assume you have familiarized yourself with the BDS user interface and its functionality as described in Part 2.
- Part 6 describes some features exclusive to the BDS MX version (slideshows and picture-in-picture).
- Part 7 shows how to create 3D discs. Either in a SBS (side-by-side) manner or as proper MVC (multi view coding) 3D disc.
- Part 8 shortly discusses all the other things BDS is capable of but is not discussed at length.
- The Appendices provide, amongst others, information on the sample files you can use for the projects in this user's guide, BDA specifications for bluray discs, a list of external tools that may prove useful, a terse Java programming guide and how to handle dockable windows.

All projects in parts 2 to 5 have "ready to run" BDS projects as well as "build your own" components to build your own project. The "ready to run" projects can also be used to inspect the various settings and assignments to project objects if you get "stuck" in your own building. In the remaining parts the projects are more of a hinting nature, assuming that by then you should not need as much hand-holding as in the beginning.

In real life you need to do most of the video editing outside BDS in order to provide what it needs, we once more point to the use of external tools, some of which are mentioned in Appendix E: External Tools.

There is a help file (Tools > Help) that opens a Windows help file with chapters pointing to specific information. If you're familiar with the "Lite" help, the HTML help files for Standard and MX are infinitely better usable – even if the 20 years old format, a .chm (compiled HTML) file is used. This format has been discontinued by Microsoft itself for its Windows product and got replaced by a direct website help or GIT-based help (GIT is an open software version control system increasingly used to store programs and help documents). None of these modern methods are useful if you are not connected to the internet. In these cases, the local .chm file is far better: it's local and therefore always available.

The Help file has a good overview and also a "FAQ" (Frequently Asked Questions) section that may answer some of your problems.

BDS – quick and how it is used in real life

Rather than reading through this guide entirely, you may want a taste of BDS by quickly creating a bluray disc using the application. You can. Switch to read and follow the instructions in the companion guide "Your First Bluray Disc Quickly" that can be downloaded from the Blu-disc Studio website at <https://www.blu-disc.net/downloads>. This will help you create a bluray disc with some short movies and a simple menu. If you master that, the reasons may fall into place when you resume reading this guide and you have a visual image in your mind of what you did.

Also, in parallel to this user's guide a companion document has been written with the title "Bludisc Studio Live" that can be downloaded from the same Bludisc Studio website. Consider it a "follow up" to this guide. It contains a number of example projects available through the Blu-Disc Studio website as either or both video instruction and BDS skeleton project files. When you mastered the basics of BDS from this guide, you can see how BDS has been used to create (commercial) blue-ray discs with their menus and animation. It shows ways on how BDS features can be applied. It does not contain project steps for hand-holding, but points to what features are used where and how. They may inspire you to do similar or grander things yourself. It will be more fruitful if you understand the products capabilities as described in this guide.

Get some RW bluray discs

In the following chapters we will develop BDS bluray projects. The final results, the files you will burn on a bluray disc, can be checked by playing them through a software bluray player on your computer.

Those players are not always without their own quirks. Some are known not to show subtitles of DVD-interlaced movies although you know they are there.

The real proof of the pudding is in the eating. Make a true physical disc from the files created by BDS and play it in a set top player. If that works fine: you did nothing wrong even if the software player

suggested otherwise. Then again, some set up players don't seem to implement the BDA standards either: especially in the field of interleaved video some foul up. To be very compatible to all sorts of players, you may want to use progressive video, preferably at 24 fps.

Because you don't want to end up with a number of disc toasters of not-quite-right discs created on write-once-only discs, invest in a small number of rewriteable bluray discs. They are slightly more expensive than the write-once discs and have slower burning speeds. That investment is easily recouped when you use them repeatedly for test runs until the disc behaviour is entirely to your satisfaction.

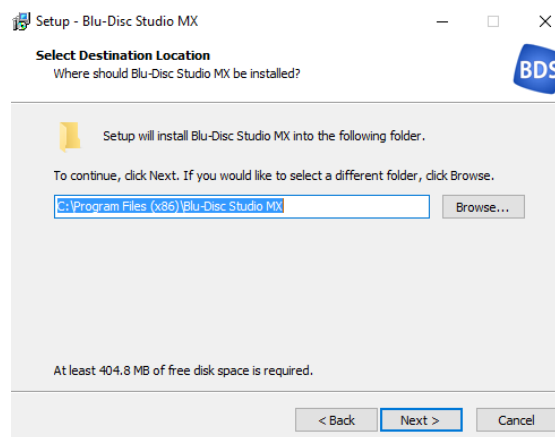
Getting started: Blu Disc Studio (BDS) requirements

This chapter discusses installation of BDS and lists the requirements that disc components must meet in order to be acceptable as element of BDS and be output to the final authored disc.

Blu Disc Studio installation

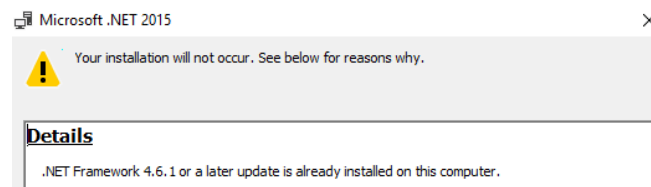
Installation

If you haven't installed BDS already, you can do so by downloading and installing the product from its website. The product only runs on Windows based systems from Windows XP upwards (at least up to the current Windows 10). It also requires Windows.NET V4 or higher.

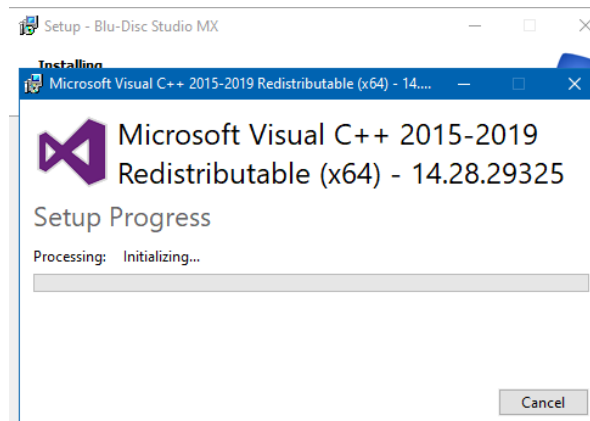


The BDS download location is found at <https://blu-disc.net/downloads> for all flavours of BDS.

- Follow instructions once you started the BDS_Setup_<version>.exe installation program. Accept the license agreements,
- Select the installation location (for at least 400 MB).
- When not already present, you may need to install the Windows .NET V4.61 libraries too (this can be done in the final step of the installation – if the system already has this library or a more recent edition installed it will tell you and not continue with the installation of the .NET libraries).

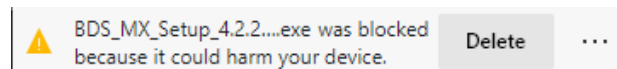


- During the installation it may discover that your system hasn't got the proper Visual C++ runtime library and in that case, that software will be installed also.

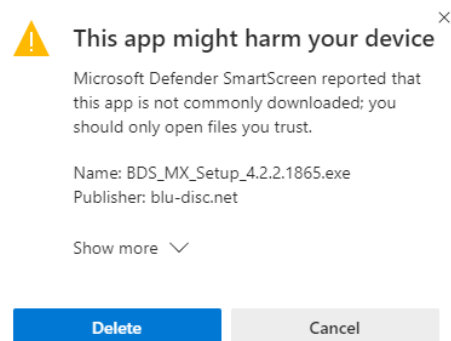


Virus scanners

Because the setup file has no signature and is not downloaded from a site with frequent downloads, some internet browsers might throw a warning about an unsafe application.



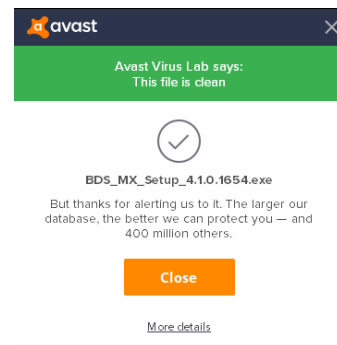
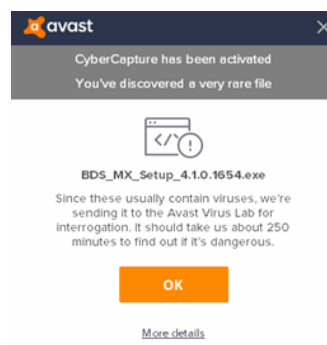
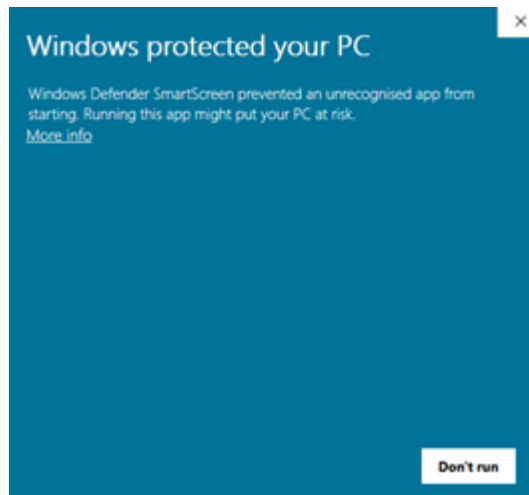
You need to press “...” and indicate to “keep it” and on the next warning



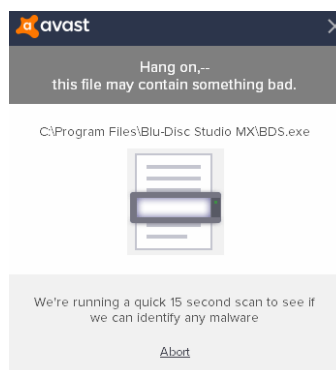
you need to expand the “Show More” details, indicate the file is safe and also specify the browser to save/keep it rather than deleting it.

Apparently, the setup file also triggers some virus scanners. Depending on your virus history, you may install without problem or encounter some warning windows that should be ignored.

- First Windows might claim the file is suspicious (click “more info” to enable the “Run Anyway” button) and next a virus checker such as Avast goes into overdrive and even sends the program to its lab for inspection – to come back clean. (You may wish to disable the virus scanner during installation).

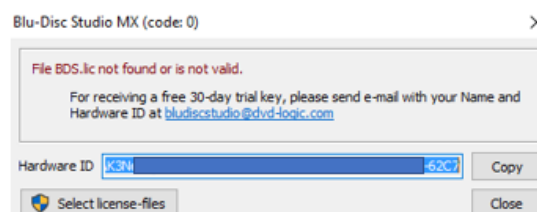


- Once installed, the virus checking software may trigger again when you run the BDS program first time only to give it again a clean bill of health after some inspection.

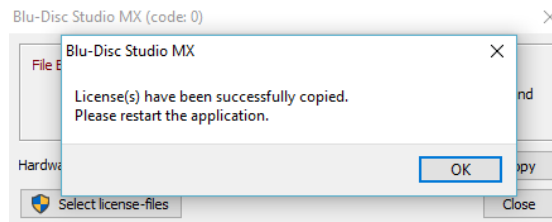


- When you update to a later version, the same warnings may occur again and should be equally ignored.

Once installed, on first startup a window will appear stating you don't have a license yet: it suggests to run in "trial mode". It provides you with a hardware-ID.



When you email this to Blu-Disc Studio support, in return they will send you the BDS.lic and <name>.lic license files – for a trial period or (once paid) indefinite with a one-year period update guarantee. Once you get the license file, press the “Select license-files” button and navigate to the stored BDS.lic file. When the license file is valid, this is confirmed and you need to restart the application.



When the BDS license file is sent to you through email, you need to copy it into the \Program Files\Blue Disc Studio or \Program Files\Blue Disc Studio MX folder. Windows may restrict write access to Administrators. In that case you may need to save the file to another folder first and copy/paste it to the application folder. That will then trigger Windows' question whether you accept this insertion. Yes, you do.

MX Muxer license

Users of the BDS MX application also need to install a separate license for the internal muxer. For the Standard version this is not needed, as only the external muxer tsMuxer is used.

The hardware ID can always be found as the second half of the “Hardware ID” textbox when the BDS application runs and you open the menu option Help > About.

It is easier to run the HID.EXE program from the BDS installation folder. This appears to do nothing, but does deliver a small text file hardwareid.txt with a single line: the hardware id (HID) of your pc, needed for the muxer license. The internal muxer is developed separately from the BDS application – therefore it has its own hardware ID code and does not use the BDS one. It also avoids BDS Standard users from activating the internal muxer for free.

Entering the license file for the muxer is done differently from the product license. The <number> BDM Full.license file must be copied to the installation folder of the BDS product. Usually this is the “C:\Program Files\Blu-Disc Studio MX folder. Afterwards, the BDS application must be restarted. You can then safely select the internal muxer for play mark indexing (movie chapters) and creating the final disc image.

Hardware requirements

Obviously, your pc must have sufficient horse power and memory to run the program and its large movie assets reasonably fast. Because you author and create blurays, you need at least 50 GB of disc space for a single single-layer bluray (25 GB for authoring, 25 GB to create the disc image).

It may come as no surprise, but you need a bluray disc burner attached to your pc to be able to burn the final disc images. A software bluray player may “run” the files from hard disc and doesn’t need the physical item. Some are mentioned in section Software video and bluray players on page 501.

Authors that send their disc images to a commercial duplicator don’t need a burner. They will need the BDS MX Pro version to create CMF disc versions to use by the duplicator.

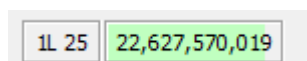
Bluray disc capacity

Whatever bluray disc project you try to create, the result must fit on a BR-25 single layer or BR-50 double layer disc. (See section Disc capacity in Appendix C: Bluray specifications at the end of this document²).

- BR 25 can hold 23 866 Mb of data
- BR 50 can hold 47 732 Mb of data

It’s my experience that several bluray set top players have problems with BR 50 discs. Their laser doesn’t read the second layer as well as the first and that may result in hanging discs or distorted images. It should always work, but the reflection of light by the second layer may be insufficient. If possible, stick to BR 25 (far cheaper too).

BDS shows you the total size of your project on its toolbar at the far right end. There you can select between values of “1L” (single layer) and “2L” (double layer) capacity.



Next to this label the toolbar shows the current size of the project. If it is shown green, you’re in the clear, if it is yellow the project may be too large, if it shows red it is definitely too large. If you changed something in the project, clicking on the icon refreshes the information.

If the final size exceeds the disc capacity (and only after muxing the project you know the exact size), you need to reduce some elements that make up the disc. Most likely candidates are the movies streams.

- Remove some movies
- Transcode some movies to a lower bitrate
- Shrink the disc content to BR25 or BR 50 using a tool like BD Rebuilder or DVDfab Bluray copier.
- If capacity exceeds BR-25, use the double sized BR-50 disc.

Bluray disc set top player specification

Resolution and frame rate

The bluray disc association (BDA) specified how movies are to be stored on bluray disc. Any other format may be supported by set top players, but there is no guarantee. See Appendix C: Bluray

² Or look at <https://en.wikipedia.org/wiki/Bluray#Video> for a Wikipedia article

specifications for full details. You may use a tool like MediaInfo (see Appendix E: External Tools) to uncover the properties of your movies. Your bluray discs created with BDS must adhere to these standards. Therefore, you **must** ensure that any movie file that is part of a BDS project conforms to these standards. BDS doesn't "do" format conversions. For the most part it simply handles the movie streams to the muxer used. If that muxer does not complain when a non-standard stream is offered, BDS may create a disc that will turn out to be unplayable.

Resolution	Indicated as	Frame rate
1920x1080	1080p	59.94, 29.97, 24, 23.97
1920x1080	1080i	59.94, 29.97, 24, 23.97
1280x720	720p	59.94, 50, 24, 23.97
720x576	576i and 576p	25i (PAL DVD)
720x480	480i and 480p	29.97i (NTSC DVD)

Note that American frame rates 30 fps and 60 fps are not legitimate frame rates for bluray discs although they can be found often on YouTube downloaded movies. BDS will fail if you try (with a mysterious "cannot find playlist" error message). European frame rates 25 fps and 50 fps are often acceptable for all resolutions.

All bluray disc players must support the frame rates specified in the table above for primary video (the indicator specifies the height in pixels, followed by "p" for progressive or "i" for interlaced):

A bluray disc can also contain SD (standard definition) movies ripped from a DVD (MPEG-2 encoding) with corresponding low resolution. The ripped files are always interlaced in either PAL or NTSC format. When such (.vob or .mpg) movies are demuxed into video and audio streams, they become .m2v (or .mpv) and .ac3 files. SmartRipper is such a ripper/demuxer.

When movie files are copied from the internet, chances are they are progressive and in all sorts of formats and resolutions. You need to convert these to

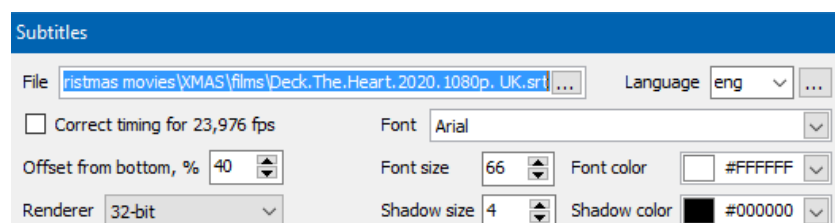
- one of the bluray disc allowed resolutions. This means you need to convert to resolutions 720x480, 1280x720 or 1920x1080
- the proper aspect ratio. These need not be 4:3 or 16:9 . For SD formats with 16:9 widescreen the player will stretch the SD image like ordinary old fashioned widescreen tv sets. It appears any aspect ratio (like 1920x900 for extreme widescreen) is accepted by players and television sets that will show the movie properly, adding letterbox or pillarbox black bars.

Subtitles

Subtitles are usually specified at the bottom region of the image and have a certain height and offset from the bottom.

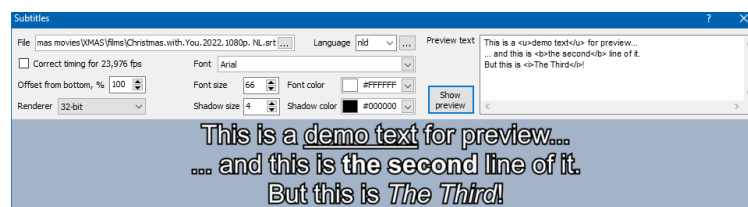
Different sized fonts and offsets are required for different resolutions. Where a 66 point font looks good on a HD screen with 1080 pixels height, they look humongous if the screen is only 720 or 480 pixels high. With a smaller height resolution, smaller fonts are required.

Not only the font size should fit the resolution, also the offset from the bottom of the screen must be modified. BDS has a separate window where you can specify the subtitle setting.



The “offset” and “font size” are the important boxes. There is a problem with the values you specify in the “offset” box. It specifies the position of the subtitle as a percentage from the bottom line of the screen. You expect a value of 50% to set the subtitles in the middle of the screen height. Not so.

- Using the internal muxer of BDS MX the percentage offset is indeed a percentage of screen height. 50% of height 1080 means they are positioned at a height of 540 pixels. However, the offset value applies to the top of the first line of the subtitle.



1% does not mean the subtitles drop below the screen: BDS then increases the value of that subtitle so that the bottom line touches the bottom of the screen. Any other low value is similarly increased until the value where the all subtitles still fit. Only then the titles move upwards with larger offset values.



Example with 66pt font on HD: offsets 1% upto 21% give the same position for three subtitle lines. (21% is 227 pixels from the bottom) They occupy the bottom 1/5th of the screen.

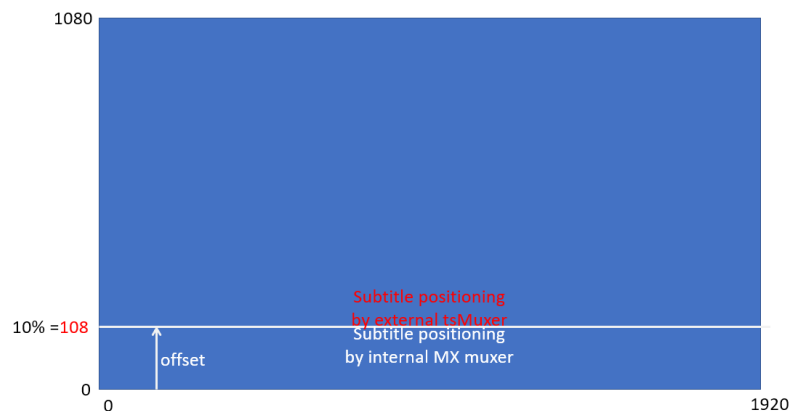
- Using the external muxer tsMuxer, the percentage is not a percentage but number of pixels from the bottom.

The offset value applies to the bottom of the lowest subtitle line. 1% means subtitles from the bottom upwards.

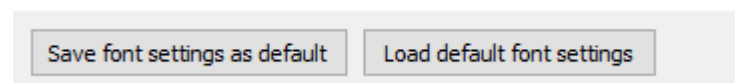
An example: what does 30% mean on an HD resolution with a 1080 pixels height?

- Internal muxer: $30\% \times 1080 = 324$ pixels from the bottom. The top line is at 324 pixels. Second and third lines are at lower positions but never below the bottom.
- tsMuxer: 30 pixels from the bottom. This is the position of the bottom most line.

The following illustration shows once more how to interpret the percentage offset (10% in the example) for subtitles. .



Any set specification can be kept as default if you press the "Save font default setting". This default can always be restored by "Load default font setting". There is only one default setting, stored within BDS. You cannot save the setting to a file and select which setting you want to reuse. Both buttons are found at the left bottom part of the subtitle screen.



Subtitles for MX muxer

The table below shows some recommended sizes for subtitles if you use the internal muxer..

Resolution in pixels	Subtitle font size	Subtitle offset percentage	maximum number of chars on a single line
720x480i 720x576i SD	25 pt	7%	50
1280x720p Half HD	45 pt	10%	50
1920x1080p Full HD	66 pt	15%	50

The “preview” button shows how the subtitles would be positioned on screen.

Subtitles for tsMuxer

Because tsMuxer interprets the percentage as real number of pixels, the table gives an indication of what values are suitable for subtitles.

Resolution in pixels	Subtitle font size	Subtitle offset percentage	maximum number of chars on a single line
720x480i 720x576i SD	25 pt	17%	50
1280x720p Half HD	45 pt	30%	50
1920x1080p Full HD	66 pt	40%	50

For your own preferred placement and size of subtitles, experiment with these values until they satisfy your taste. You can see how they turn out by making a small one-movie project and play the movie in its muxed output folder \BDMV\STREAM\00001.m2ts and enable viewing the subtitle. You cannot use the “preview” button on the subtitle window as that only shows how the internal muxer would place them.

Subtitle appearance

Some example positioning is shown below through screen captures made with VLC. There are two subtitle lines of 50 characters.



The top line shows the font size and offset settings as suggested for tsMuxer for HD 1080p and half-HD.

The SD resolution is shown at the bottom part of the figure. At the left the default 45pt font /30% offset setting for tsMuxer is shown at the left. It is clearly too large and only the middle 30 characters are visible. Modified to a 25 pt font and 17% offset all 50 characters are visible again at a more acceptable size.

Stream type of audio and video

The mandatory playback capabilities for players are:

- Video decoder circuitry for MPEG-2 (for DVD SD and H.262 HD), MPEG-4 AVC/H.264, and SMPTE VC-1 (for bluray HD) with frame rates and resolutions as specified earlier. Note that MPEG-TS output produced by some video editors like PowerDirector, is acceptable to tsMuxer and produces discs that work well on most players, it is *not* a BDA-compliant format and may fail to play on certain players or be handled by the internal MX muxer
- Audio reproduction capability for:
 - Dolby Digital (.ac3 AC-3), 640 kbps, at least two channels (up to 5.1), 48 kHz sampling
 - DTS Digital Surround, at least two channels (.dts)
 - Linear PCM (.wav, LPCM) at 48, 96 and 192 kHz sampling, at least two (max. 7.1) channels.
- There are four optional audio formats for bluray disc –
 - Dolby Digital Plus (.eac3), up to 7.1 channels, 48 kHz sampling
 - Dolby TrueHD, 4736 kbps, up to 7.1 channels, 48 kHz sampling
 - DTS Surround (.dts) 1509 kbps, 5.1 channels, 48 kHz sampling
 - DTS-HD Master Audio™, (.dts lossless) 3000 kbps, up to 7.1 channels, 48, 96 or 192 kHz sampling

You may wonder if BDS (rather its muxers) supports the more recent Dolby Digital Atmos sound. It does. Physically Dolby Atmos is a modification of Dolby True-HD. The muxers support it – audio files must be provided the same way as True-HD (two separate files for the internal muxer ,mlp and .ac3³ or one multiplexed for tsMuxer).

Note that audio tracks do not support .flac, .mp2 or other MP3-similar formats nor does it support the CD common sample rate of 44.1 kHz. .

In case the movie also has associated PIP (picture-in-picture) movies, additional restrictions apply to those PIP files. See "Number of BDAV/BDMV elements

There are maximum numbers of allowed audio streams and subtitles: 32 video and audio streams each, 255 subtitle streams.

Number of angles	9
Secondary video streams	32
Audio streams	32
Secondary audio streams	32

³ You should specify .mlp file only. The .ac3 file should have the same name and be located next to the .mlp. For example: C:\Project\audio.mlp and C:\Project\audio.ac3

Text subtitle stream	255
Popup menus	32
Chapters (playlist marks)	255
Playlists	999
Play items in a playlist	255
JAR files	Limited by disc space
Fonts	255
Menu sounds	128

Supported primary and secondary video streams” on page 473. PIP movies are only supposed by the BDS MX version.

Initial BDS configuration

This will be a short chapter, but some settings must be set right for BDS to work properly. Additional settings we will cover when they become important to do the job. Those options are best left to their initial settings until they get more meaning once you get familiar with the program.

Migrating from the “Lite” version

If you are like me, you tried the “Lite” version first and then after a while discovered that however good that already is, some things you want are missing. Time to upgrade. The big advantage is that you probably already know the user interface and work flow expected to create a bluray disc. However, a number of steps in the BDS versions are hidden in other menus or buttons than you know them from “Lite”. They may also have a different icon or colour. And of course, some steps now offer many more options than the few that “Lite” was limited to. For example, in the paid versions of BDS there is a third way to create menus. This is done from within BDS Designers View window rather than importing multi-layer Photoshop or single layer .png files.

You will find that the steps for a simple “Lite”-like disc are largely the same in the BDS versions, but with some twists and some new ways of doing things. You may quickly scan through some of the text in this user’s guide if you get stuck in your “Lite-to-BDS” approach. Parts 3 and later will be specific to features of BDS versions that BDS Lite does not have.

Lite users who turn to BDS and who are home users (non-commercial) may want to uncheck two items that are potentially of no interest to them and can cause confusion:

1. Project>Project Settings>General tab: uncheck “Sign the JAR”
2. Tools>Options>Compile tab: uncheck “Validate SIG”

If you want to employ the “resume” functionality, you do need to sign the JAR.

You can open any Lite project by BDS and further edit it. However, once you have saved the new project under BDS the project file becomes unusable to Lite. If you want to keep both, you need to either

1. recreate the Lite project, set all pointers for files, menus and buttons to the BDS project, define its own output folder and regenerate the disc into that folder.
2. Copy the “Lite” project folder to a new folder and edit that copy with BDS.

The project file is actually an XML file that contains all the BDS specifics such as menus, movies, actions, scripts. All but the movie files and menu images themselves. Adding some BDS functionality inserts this into the project file. The Lite edition doesn’t know about this additional BDS functionality and therefore may consider the project file corrupt.

Clean start: no previous BDS experience

If you're new to BDS without the "Lite" experience: the learning curve of "Lite" still is present, but you'll quickly be able to accomplish what you intended. If you want the Moon it may take a little longer. Read on.

A project always creates a project.bdmd file with the project item configuration. If you expect to create several similar looking discs, you may rename the project file to e.g. "disk 1.bdmd" and then copy the final version as "disk 2.bdmd" to start with a very similar looking project (of course the movie files and menu text will be changed for disk 2).

BDS Project > Project settings

When you work on a project, specific characteristics for that one project can be set using this menu option. The relevant items will be discussed when we work on a project.

BDS Tools > Options

Characteristics for all future projects you will work on, are configured through the menu option Tools > Options. They are valid for all projects and are related to ways of muxing, BDS developer interface and such.

Tools > Options has four tabs: General, Interface, Compile and Muxers. Change their contents to meet your preferences. To ensure these preferences are used in all new projects, click the "Save as default" button after all settings are done.

General

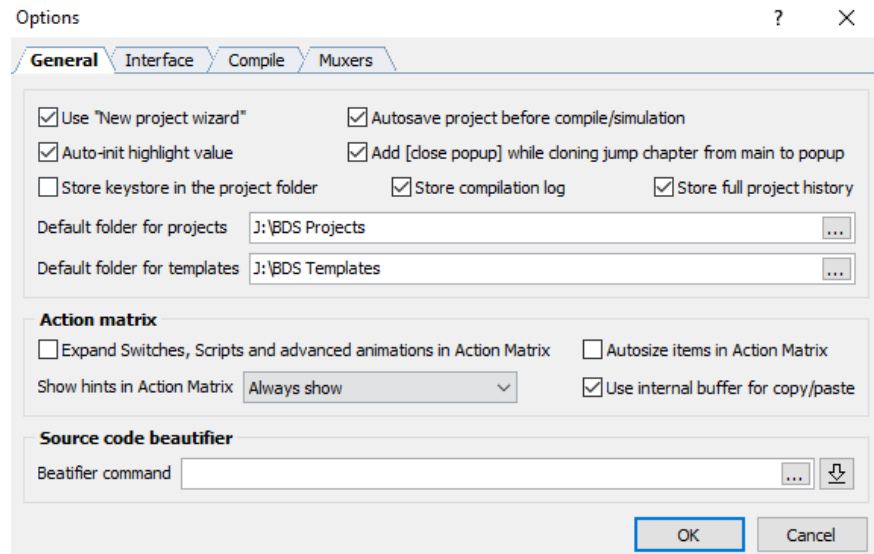
"General" provides some overall settings, like whether to use the project wizard for creating new projects, to save the project before compiling it and others, where to locate projects by default (so the "create project" wizard knows and suggests).

It also allows you to specify where "template projects" are stored: incomplete projects that contain shared properties and settings on which real projects are based. Useful if you want to make several discs with a similar appearance (such as 3 discs covering a television series of 4 episodes per disc).

A template project is created by saving a normal project (which may be incomplete) as a template. This results in a template project with local copies of the menus and buttons. Any movie placeholder is also copied, but their pointers to movie files remain pointing to the originals. You may wish to open a template as a regular project and remove either the file pointers in the movies and playlists or delete them altogether.

Note that you need to create a top-level folder for the template project under the BDS templates folder. Otherwise all copies reside directly under the BDS templates folder. The question asked for the template name only sets the name of the .bdmd file of the template— it does not create a folder of that name.

Using the template to create a new project creates this project in the default BDS project folder by making local project-local copies of menus and buttons. Any movie placeholder that points to a file remains pointing to that file. Here too the creation of the new project is done under the BDS project map. You need to create a project folder under which to copy the template files. The question asked for the project name only sets the name of the .bdmd file of the project – it does not create a folder of that name.



The bottom “source code beautifier” (as of V4.1.0.1676) allows created Java functions to be formatted in a predefined way (proper indents, colours etc), the “beautifier” does this. This is a JAR (Java Archive) file. By pressing on the down-button at the right, you open a GIT repository on the web where you can download the beautifier: `google-java-format-1.7-all-deps.jar`.

Add the command in the “Beautifier command” text box. E.g.

`D:\<folder path to the jar file> "java.exe -jar ↵`

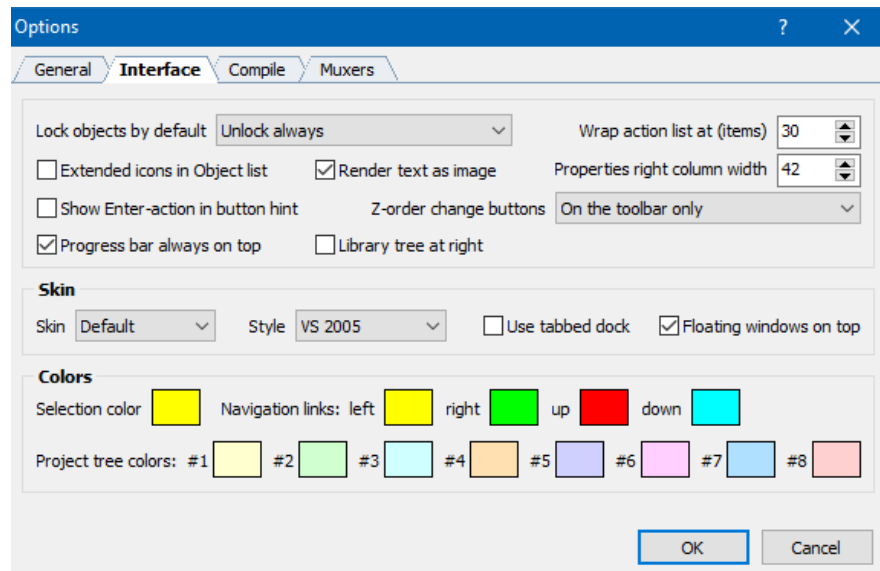
`D:\Programs\google-java-format-1.7-all-deps.jar -r". (one line!)`

For it to work, you have to have or install the Java runtime environment.

Interface

The “Interface” tab contains settings that allow specific colours to be used in BDS application.

To move menu objects around the screen during design, these objects must not be locked to their position. They can be unlocked at any time, but in this setup menu you can specify how the initial behaviour should be.



The “Render text as image” helps in the Designer Window to scale text if you don’t watch at 100%.

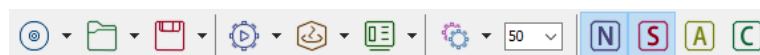
You may want to change the “Skin” setting which rules the appearance of the BDS interface. Screenshots in this user’s guide use the “Default” flat skin, but the optional “New Style” gives a more 3D type impression, “Neon” more of an outline impression. All three are shown below.



Default



New Style



Neon

Compile

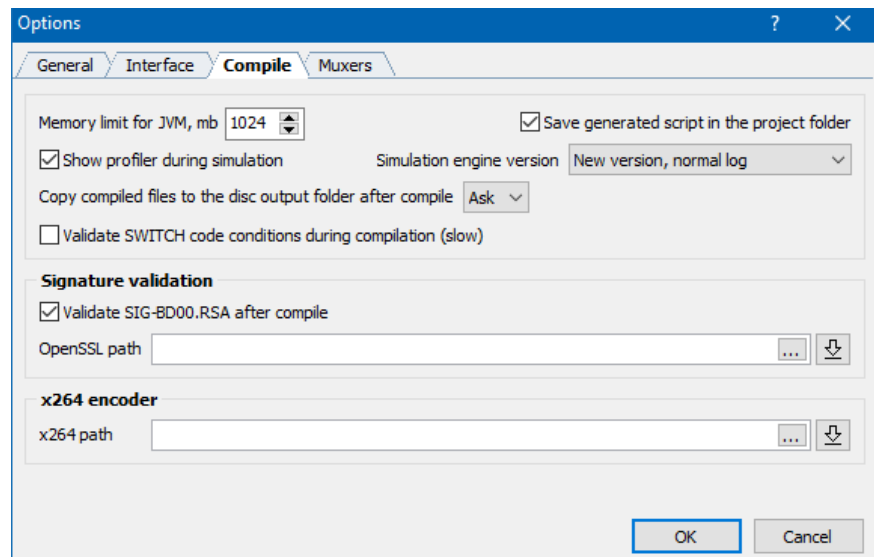
This window allows you to specify the amount of memory to use for the JVM (Java Virtual Machine). This memory is required during the simulation process. This limit is also used during the compilation process. The JVM requires additional memory for the graphic images that it draws on the screen during simulation. If you use a lot of graphics you may need to increase JVM memory limit value.

Java stores its Javas and program files in a specific project location (subfolder __SCRIPTS) that cannot be changed. Those files also become part of the final disc image. They are then stored in the __JAVA folder in a Java archive (JAR) file. Using a separate (disc)folder for the Java code allows to replace it if during authoring you only change menus and navigation. A recompilation of the Java code and

replacing the old JAR file updates the muxer output folder without having to remux the movie streams.

Most Compile settings are best left to their initial settings for now.

If projects need not to be signed you may clear the “Validate SIG” check in the “signature validation” section. This may apply to home users but not commercial users (see Appendix B: Commercial settings). Signing is required if you create a multi-jar project (Interlude: JAR title files with menu or movie on page 207).

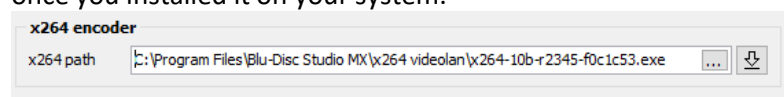


If you need to sign your projects:

- The “download” button next to the “OpenSSL path” redirects you to the wiki.openssl.org website to download the OpenSSL binary distribution for Windows. The “path” must contain the complete file specification of the “Win32 (or 64) OpenSSL <version> Light.exe” file, known once you installed it on your system. BDS recommends the use of versions from overbyte.eu (Download OpenSSL Binaries section) or bintray.com/vszakats or indy.fulgan.com.

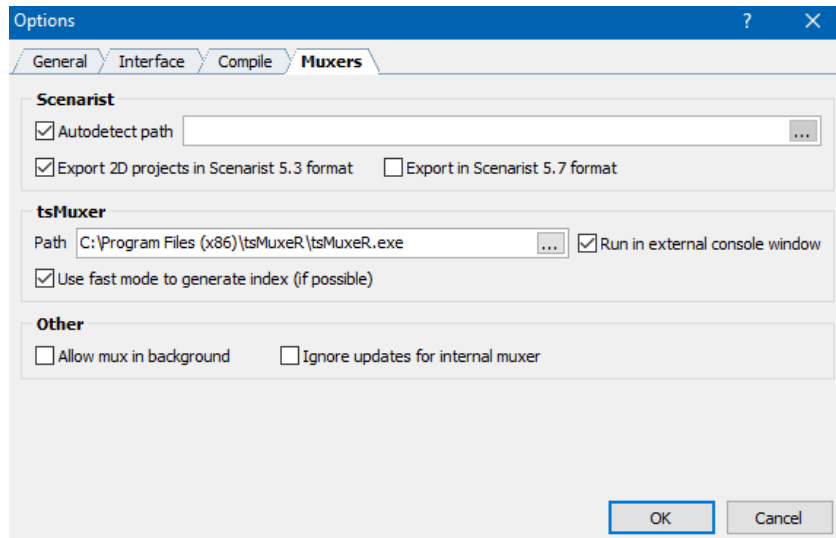
If you create slideshows (BDS MX version only):

- The “download” button next to the “x264 path” redirects you to the download.videolan.org/pub/x264/binaries/win32 website to download and install the x264 library that is required for BDS MX if you want to create slideshows. X264 is a free software library and application created by Videolan for encoding video streams into the H.264/MPEG-4 AVC compression format. The “path” must contain the complete file specification of the “x264 <hashvalue>.exe” file, known once you installed it on your system.



Muxers

The “Muxers” window is specific to the process step that converts the project into a bluray disc. The most important part is the specification of the location of the tsMuxer. The muxer combines video, audio and subtitle streams and inserts chapter marks.



The BDS MX version has its own internal muxer (V5.0.1 at time of writing) once its license file makes it available. It need not be specified explicitly. Updates on this muxer are automatically provided except when the “Ignore Updates” checkbox is checked.

The internal muxer requires BDA compliant MPEG4-AVC video streams. tsMuxer is more relaxed and also accepts some other formats such as MPEG-TS.

The creator of tsMuxer, Roman, worked with satellite formats that needed to be converted into IPTV formats and which may explain why tsMuxer is more lenient towards source files, ignoring BDA errors but the process does not produce a fully BDA compliant disc (although it plays fine on almost all players – which is what matters most to home users).

Cyberlink's PowerDirector creates MPEG-TS streams (even if said to use H264/AVC) which is acceptable to tsMuxer but not the internal MX muxer. I'm told Adobe Premiere does deliver BDA compliant files but am unable to confirm or deny this.

Unchecking the check mark “Run tsMuxer in external console window”, the tsMuxer output is redirected to the “output” window of BDS rather than having separate muxer DOS command windows to pop up each time the muxer starts working on a movie file. This removes the risk that you interrupt the muxer's working because suddenly you find yourself typing in its window (it gets focus!).

I found that if tsMuxer is used in BDS MX 4.1, disabling the external console window may cause the muxing of the project to abort when the “Updating” phase of the disc building starts. An error message of “The system cannot locate the object specified. Line: 0” error pops up. In that case: check the checkbox for the external console window and rebuild the disc again – hopefully this time without an error.

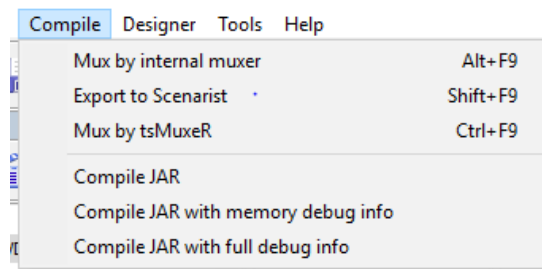
The muxer also creates the index file required for chapter marks. To create a new index file quickly, check “Use fast mode”. This skips

several steps not immediately needed when a new file is created. The tsMuxer and internal MX muxer use different index files – the right ones are created depending on your setting of the muxer to use.

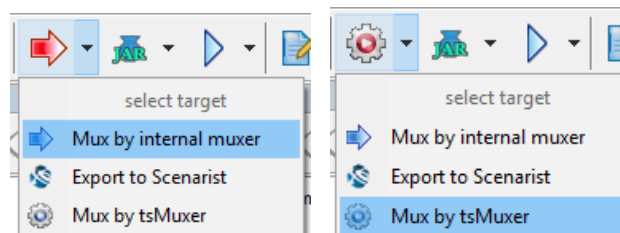
There is also a section for settings for the Rovi Scenarist BD application. This is an external authoring application with its own muxer (capable of 3D movie handling). BDS Standard and MX can export its final data to Scenarist for muxing to a disc image. Its settings may be left empty if you don't use it.

In BDS MX you specify which muxer to use in the “Compile” dropdown on the menu. Select “Mux by tsMuxer” or “internal muxer” For the BDS MX specific features slideshow or PIP, you must use the internal muxer.

If you want the Rovi Scenarist BD program to do the final muxing, skip the muxing step in BDS and instead export to Scenarist.



The “muxer” button indicates through its icon what muxer will be used.



BDS Movies – filename requirements

Officially there are no restrictions to file names. However, experience shows that sometimes during the building of the bluray, the process is aborted (“cannot copy 0000.jar file”) which is a strange way of saying “some file names have characters in them that they better not have.”

Some satellite receiver stations (notably BBC) insert non-printable characters in their file names. They don't show in the Windows Explorer file lists. These are acceptable to BDS but not to tsMuxer and it will issue error messages for those files. BDS tries to make them acceptable by replacing non-printable characters by “#” characters.

It seems safe to avoid “strange” characters in their names. This also includes punctuation marks like apostrophe ('), ampersand (%) as well as diacritics such as ë, ø and any non-A-Z or 0-9 characters. Several other pitfalls are also mentioned in the online help file in section “How to avoid problems”.

BDS Movies – movie picture and audio requirements

:BDS allows an unlimited number of movies to be added to a project – as long as their total size fits within the target size of the bluray disc. They need to conform to the BDA defined standards (see Appendix C: Bluray specifications). If they do not, you need to transpose them to one of the BDA standards.

Especially the resolution must be one of bluray's allowed versions, most likely 1920x1080, 1280x720, 720x480 (NTSC) or 720x576 (PAL). The aspect ratio of the movie itself (16:9, 2.35:1, 2.55:1 etc) are not important: the player will add letterbox or pillar box black bars to make it fit the movie resolution of the screen without distorting the picture.

You can use the MediaInfo tool to inspect the movie properties. The video stream must be of MPEG4-AVC or other BDA compliant format type. Many .ts files are just that – transport streams (and have MPEG-TS stream type) and are not BDA compliant. They are acceptable to tsMuxer but they need to be converted into BDAV AVC to become acceptable to the internal MX muxer. Also, MPEG-TS, being not compliant, may cause a disc not to run in all standalone players although in my experience all of the ones I used do.

During the editing phase of a movie you may want to remove certain parts (like commercial interruptions). In that case, ensure that each section starts with a full frame (I-frame). When not, the transition at a join will show pixels and a stuttering picture at the transition. Also, the audio may start to run out of sync. A true video editor will create I-frames automatically at editing cuts but simple cutting tools do not.

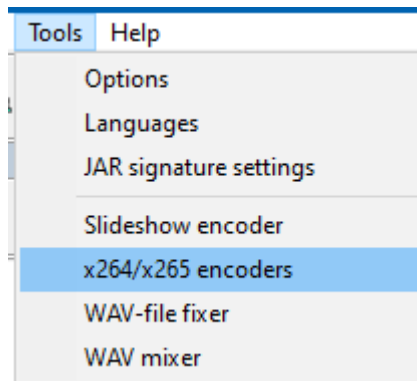
When a movie has a combined video/audio track, as is usual when ripped from DVD, BR, television, YouTube or video camera, it must be demuxed into its component tracks after it has been transcoded to a BDA compliant format. One video track can have several audio tracks (original, dubbed translation, editor comments, music-only and such) and subtitles (for different languages).

Transcode to AVC

There is a range of free or paid encoders that may assist you in converting film formats into BDA acceptable AVC files. Some of them only have a command line interface, some have a Windows GUI around them. and some are full fledged Windows applications (like Wonderfox HD Video Converter Factory).

But BDS has such an encoder inside: x264. You can use it standalone (use the command file that can be downloaded from the BDS site https://www.blu-disc.net/download/x264_cmd.zip) or from within BDS.

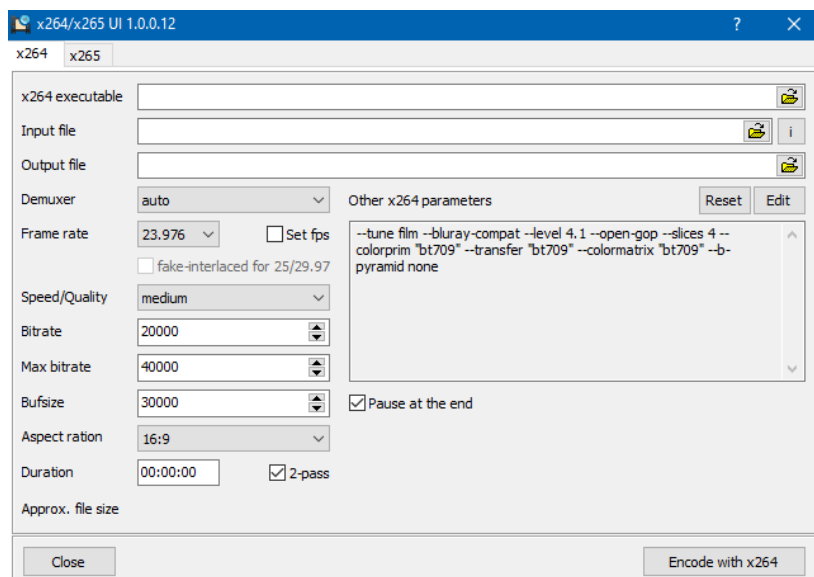
On the menu bar, select Tools> x264/x265 encoders



This opens a window in which you can specify your source movie file and the transcoded AVC version of it. You may drag the file from an Explorer window onto the encoder window.

The encoder itself is automatically installed in <system disk>:\Program Files\Blu-Disc Studio MX\EncoderUI when you install 64 bit BDS.

You need to explicitly mention it in the "x.264 executable text box) as the interface also allows other encoders (that use the same command line format) to work with this. You need to specify the .exe version and not the GUI file that is located in the same folder.



You may wish to change frame rate and bit rates when these are different for your own source file. The higher bitrates provide a better image (not true for "upsampling") and the correct frame rate avoids single frames to be duplicated or omitted to meet the new frame rate specified.

To transcode, click the "Encode with x.264" button. This opens a command window for the encoder to do its work. The encoder is rather slow and only produces a video stream. The audio must be demuxed using another tool, such as tsMuxer. You may need to apply the command mode eac3to transcoder to make the audio BDA compliant by a command like

```
eac3to input.wav output.ac3 -448
```

The odd thing is this window remains open and for each encoding a new command window and GUI are opened. You need to manually close the older ones. Or end up with several GUI and command line windows.

BDS Movies – subtitles requirement

Subtitles can be added to the movies on disc. To be acceptable, they have to be created in either

- .srt – the well-known subtitle format which is a text file with time codes
- .sup – a transparent image with the subtitle as opaque image part. Basically, it is a picture whose text cannot be edited (usually created when a ripped bluray .m2ts file demuxes .sup subtitle files).
- .xml – the BDN Xml format (sometimes delivered in .bdn file type). A bluray specific subtitle format defined by Sony which has 3D extensions to specify how deep the titles are immersed in the image.
- .pes+.mui – the PES format – a variant similar to .sup subtitles

The tool “subtitle edit” can convert many of these formats into the more common .srt. For image based subtitles it attempts to convert it to text using OCR conversion (See Appendix E: External Tools).

The text-based .srt file must not contain non-printable characters. This will not be acceptable for the BDS used muxer but is only discovered at disc muxing time (logged error: “Meta data generation failed”).

Sometimes they contain instructions meant for .ass files where you can also specify subtitles to move to the top of the screen (with the {\an8} instruction). The muxers don’t detect this (as it is plain text) unlike tags like <i> for *italic*. The .ass instructions need to be removed.

For suitable size of the subtitle depending on the resolution, see section Bluray disc set top player specification on page 20.

BDS Menu requirements

Menus consist of background movies, images (mostly textual descriptions) and buttons.

There is a maximum number of pixels for all menus together: 7 900 000 for V2 of the Java JVM (selecting V1 allows even less: 59 000 pixels). This corresponds with about three full screen images. They are stored in a single Java JAR file archive. If you need more pixels on your menus, you must divide the menus over multiple JAR files (see Interlude: JAR title files with menu or movie on page 207).

To reduce the demand of menu pixels, you can provide a menu background through a menu background video. BDS insists that the background is a (short) movie and its resolution defines the size of the menus in which it plays. For best results, the background movie *must be* of 1920 x 1080 pixels resolution. Any other size may distort a menu

on playback as each menu is cut to fit the movie resolution. This may imply that a menu movie of 1280 x 720 (not full HD) combined with a menu at 1920x1080 (full HD) will only show 1280x720 pixels of that menu: the left top part only.

The background movie does not count against the total pixel number (7 900 000) allowed for all menus together. It is therefore advisable to always use a (even solid black or other coloured) movie as menu background rather than a black still image.

We will cover menus later, but keep in mind that menu texts, buttons and other visible objects are created from picture .png files. These .png files allow for transparent parts instead of filling it with some default colour. But transparent pixels also contribute to the pixel count.

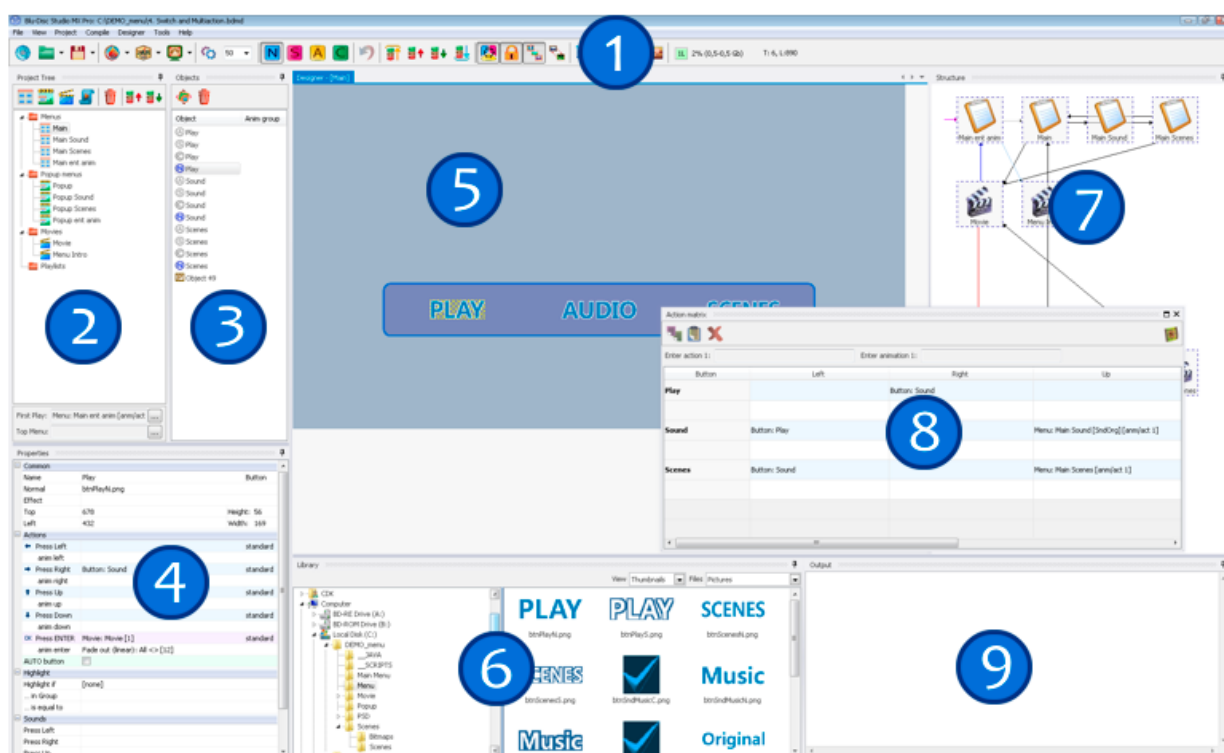
Using Blu Disc Studio and customizing it

The Blu Disc Studio interface

The BDS interface has many objects that all are useful at some point in the project. To create a basic project, you may use the wizard that is invoked through File > New. This will ask for a movie (with all its streams), the main menu (composed by a .psd file with layers or set of .png files), bonus movies and popup-menus. Any element you do not have you simply skip by clicking on the “next” button of the wizard. Those missing elements are added later manually. The wizard sets up a basic project to which you can add any element at any later time.

You can just specify File > New and only specify project name and project folder and leave it at that for the moment. The required elements will be added later.

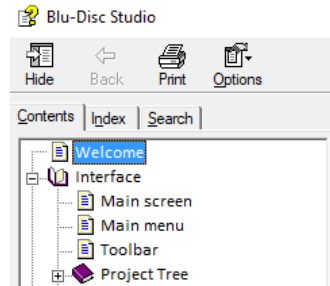
The user interface of Blu Disc Studio looks a bit like the figure below – but this can be customized to fit your needs as most windows are dockable. That means you can pick up any part at their title bar and move it to any place you like. Independently as a floating window or docked at another location within the interface (see Appendix G: Dockable windows for more details).



We will build the project manually to make you familiar with the interface. There are various independent windows that show information of the project and allow you to modify it. These windows are made visible using the “View” menu option that lists the various windows. They can be positioned independently. And by pressing their

“X” button in the top right corner you may make them disappear from view.

In the discussions on making a project, we will indicate what window to use. But for completeness sake the figure above shows all of them (as does the Help > Help file in its “Interface > main screen” topic).



1. Main menu and Toolbar – The area containing pull-down menu options and the bar that contains shortcuts to menu and other commands. Many buttons have a menu text equivalent.
2. Project tree – shows all menus, movies and playlists. Basically, it shows what objects your current project contains.
3. Objects – shows objects of the selected menu such as pictures and buttons.
4. Properties window – shows properties for the selected menu, movie and graphic object such as buttons.
5. Designer window – shows the selected menu, where you can select objects and move them. Used to compose a menu screen with all its objects. From 4.2.1.1794 onwards you can also resize objects.
6. Library window – shows graphic, video and audio assets to import in the project. It is shown here in its undocked state.
7. Project structure window – displays an overview of the current project and its navigation: how buttons connect to menus and movies.
8. Action matrix – indicates what happens when a button on a menu is pressed. This window is used for quick copy actions, animations and highlights between buttons that have similar properties.
9. Output window (basically used during compile/export). This output is also stored in a log file in the project \Log subfolder.

General and project specific settings

There are two major places where you customize your authoring process:

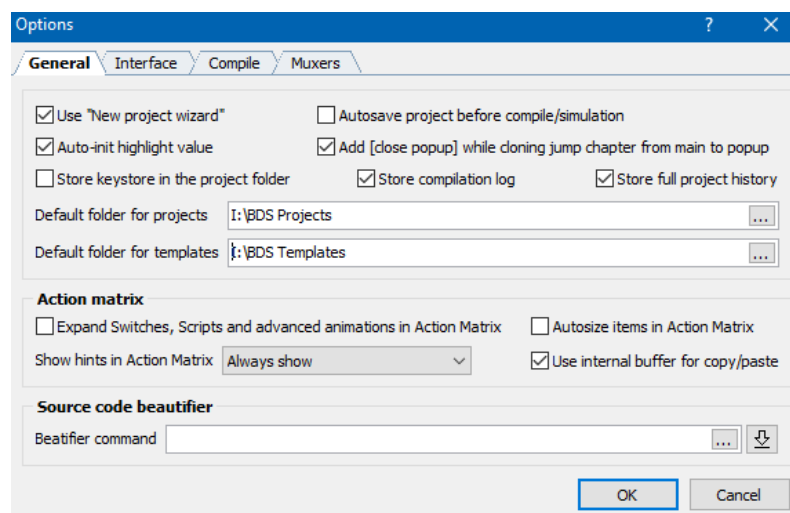
1. Always: settings in Tools > Options
Use this to specify the muxer to use, Java coding options, default project locations, etc.
2. Project specific: settings in Project > Project Settings.
Use this to specify what default audio tracks to use, what Java functions are coded for this disc, where to create the disc image, etc.

Step 1: Get organized

When you're an organized person you can skip this preliminary step as you obviously know where you put everything.

But for all others: make sure you know where you get your stuff, where you leave your stuff and what you create in the middle phases. Making a mess of a project with elements all over the place and possibly with duplicate items, is guaranteed to create a nightmare for you.

BDS helps you by setting a default top level root folder for all your projects specified in Tools>Options "General" folder. Keep in mind that the folders you select must already exist. BDS does not create them for you.



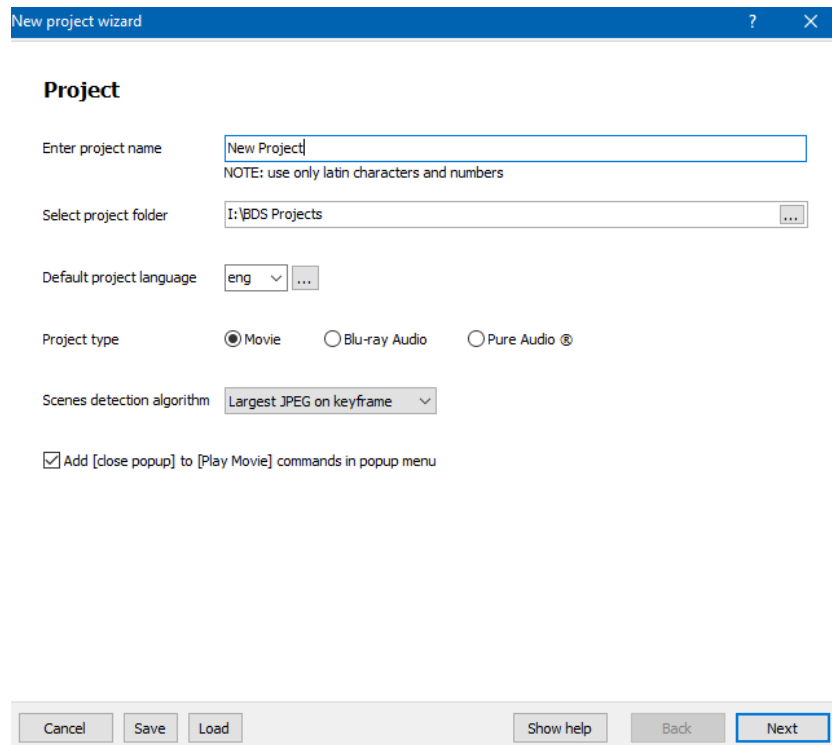
If you have a template project, that contains standard objects such as button shapes or fixed menu pictorial objects that you always use in your projects, you can store each template as subfolder to "default folder for templates". You use BDS to create such a template project and save it as such using the menu "File > Export template...". You can create different templates for different occasions.

When creating a new project that you want to build upon a template, use the File>Import from template. This allows you to select a template project. Once loaded, it will ask what the name will be for the new project .bdmd file. You need to (first) create a project folder yourself under the BDS default folder.

If you want to base your project on a template, do not create a new project by the New Project wizard. This does not allow you to use a template project.

Its "Load" button means "Load wizard project" and allows you to load a project that you can save from the wizard window (also saved automatically at the end of the wizard). This is not a real project – it is just the wizard window settings saved in an external file.

Any time you create a new project (File>New or New Project wizard), it will be added as a folder below the "default folder for projects" folder. For illustration purposes, let's make a "New Project" project.

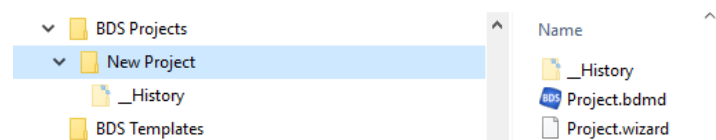


With BDS you can select what sort of project you are making: the usual movie disc or an audio-only disc (two flavours – neither of which have so far reached the general public at large). For our purposes, the default “movie” is the right choice.

Use the “Load” button to reopen an existing project rather than creating a new one.

Pressing “Next” repeatedly results in a set of new project folders that will be populated during your work on the project. It already contains a hidden “__History” folder that BDS uses to keep the “old” project file in case you decide to exit without saving changes. Hidden folders are normally not seen in Windows unless you go to the File Explorer > View window and check “Show hidden folders” in the “Options” menu item.

The project file itself (that “knows” how you have setup the movie and menu objects) is stored as project.bdmd file in the top project folder.



You can open this file (double click suffices if the file type is linked to start the BDS program or via the BDS interface File>Open) and edit the project.

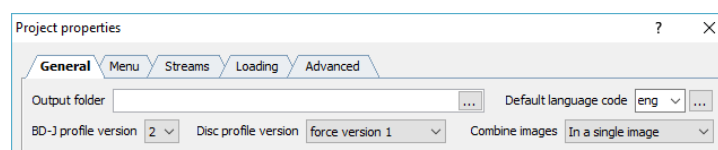
You can copy or rename the file into anything you like. This way multiple .bdmd files can reside in the same project. This is handy if you want to have multiple projects that use the same ingredients but are handled slightly different (e.g. with or without menu animation or

switches or several disc projects for a set of bluray discs that contain a television series).

Within the “New Project” folder you may want to add some subfolders to get organised. I suggest you use a minimum of

```
\New Project
  \films
  \original sources
  \buttons
  \fonts
  \output
```

- In \films you can add all the movies demuxed in their video, audio and subtitle components. Those components are created by external applications. BDS will connect to these when you fill in the movie properties.
- In \original sources you can store all menu files you create as Photoshop .psd or image .png files. It is not needed if you create the menus through BDS itself. BDS will read .psd files and store them in a new subfolder \<menuname> (which is the .psd file name) that it creates for the New Project. It decomposes the .psd file layers into separate .png files and simply copies any .png file into its \<menuname> folder: this includes any picture and all button states file.
If you used .png files, they remain where they are and no \menuname folder is created. In that case you refer to your .png files where you defined them (in \original sources and/or \buttons)
- In \buttons you can store .png images that represent the various states of a button (more about those later). These images are used if you create menus from within BDS and specify button objects.
If buttons are part of a Photoshop image (a separate layer) buttons are created in a \<menuname> folder after a Photoshop file import.
- In \fonts you collect all .ttf font files you use in creating BDS made menus with their background image objects.
- \output will contain the final muxed bluray image. You specify this output folder in the Project > Project Properties menu option window under the “Output Folder” text box. It is the only time BDS will create this folder if it doesn’t exist.

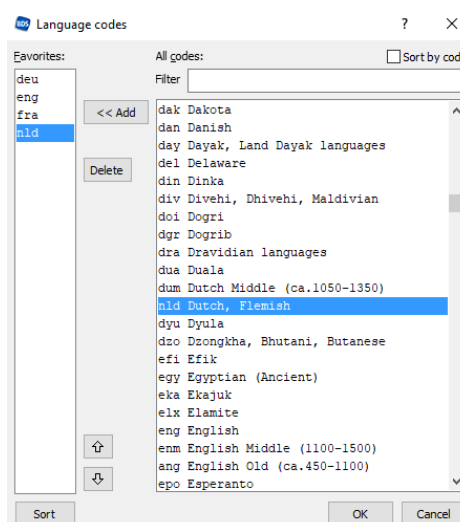


Step 2: Specify your languages

For any audio or subtitle stream using a particular language, you need to ensure that language is part of the list of languages from which you

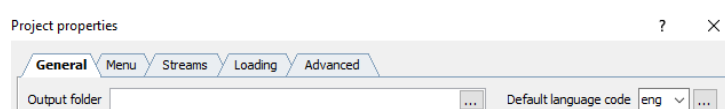
can choose. By default, BDS has a short list of languages which reveals its East European origin.

Set the required language codes by going to the main menu option Tools > Languages. The popup menu allows you to remove unwanted languages (“Delete” button) and add (“<<Add”) desired ones into the “Favourites” column. Those codes are shown on playback if the user selects an audio or subtitle stream using the “audio” or “subtitle” button on the remote-control.



The languages are listed by their name, not their code or country. So “Dutch” will be found under “D” and not under “N” for “Netherlands”. A checkbox allows you to sort the list in code-order. Then “nld” will be found under “N” for the Dutch language code.

The most applicable language can be moved to the top of the selected list using the up/down arrow keys on the popup window.



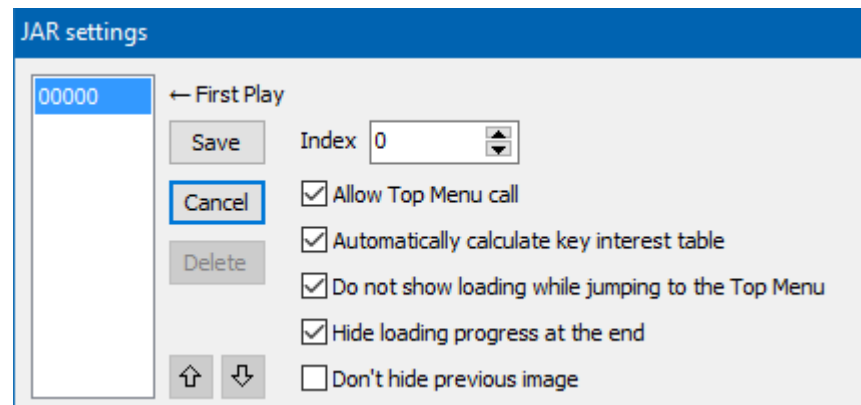
The default language to use in a project is set in the Project Properties window (Project > Project Properties) in its “General” tab indicating the “Default language code”. This will set any audio and subtitle track by default as being that language. You can still change this language assignment manually for each track.

When you add audio or subtitle tracks to your project, you can indicate the language associated with the track. However, if the file name of the track includes the official language code, BDS will automatically apply that as the language indicator. For example “commentary DAN” as an audio track file will be given “danish” as language since “DAN” is the official code for Danish. Of course, you can always manually change the language setting for each track, overruling the code that is part of the file name.

Step 3: Enable Top menu

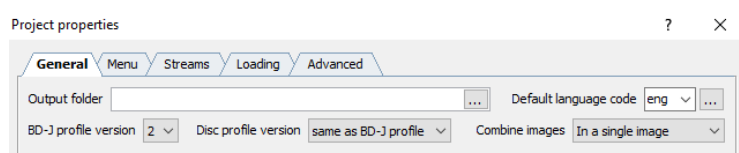
A disc in a player can be forced to go to the top menu via the remote control. This only works (as of V4.3) if the Java code, contained in a Java Archive (JAR) file, has been set to allow a Top Menu call.

You enable Top Menu instructions by going to the Project menu option, select "JAR Properties" from its dropdown list⁴. Select the 0000 Jar and click "Edit". This allows you to check the item "Allow Top Menu Call". Once set, save the setting by clicking the "Save" button. Then click "Close" to close the properties window.



Step 4: Specify the Java version to use for menus

While you're at it, you may also wish to change the BD-Java profile version to V2 in the same "General" tab. This allows for a larger total pixel size of all menus together (up to 7 900 000 pixels – about 3 ½ full HD screens worth of pixels in V1). By changing item "Combine images" from the default "in a single image" into "Split into separate files" you have twice as many pixels for the images.



The output disc profile version may be forced to Version 1 (for older BR players) or to be the same as the BD-J profile you use ("same BD-J profile").

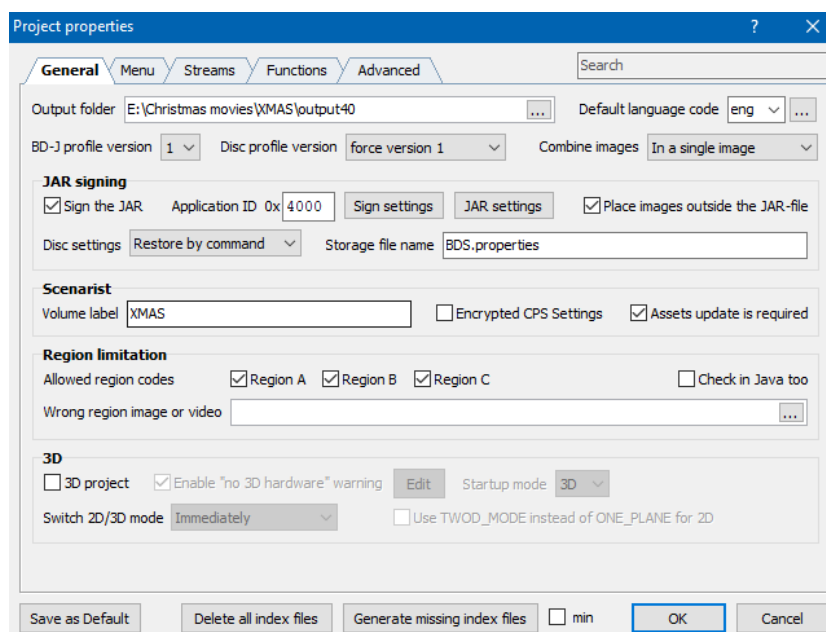
It is advised to keep the disc profile version to "force version 1". The other option, "same BD-J profile" results in discs that are not universally accepted by (older) set top players. The disc may play only on some set top players and software players).

Step 5: Signing and restoring (resuming)

All characteristics specific for the disc you author are found under the Project > Project Settings menu windows. Two items are important:

⁴ It also opens through Project > Project settings > button "JAR Settings" in the JAR Signing section

- do you sign your Java generated code: check “sign the JAR” and specify values for signing by clicking on “Sign Settings”
- do you want to resume playing your disc from the point of interruption (set “Disc Setting” to either “Restore by command” or “Restore on disc load”).⁵



The “General” tab contains the most likely information you may wish to edit. The other tabs will be discussed when they become relevant during authoring.

The Output folder indicates where you want BDS to create the bluray disc image when everything is muxed into a final disc. It is common to use a subfolder \Output to the the project folder for this.

Projects with multiple JAR files need to be signed (see Using multiple JAR titles on page 207).

The other tabs are relevant for several other project-specific settings that will be discussed when relevant (like "menu" for some exotic features, "streams" to indicate what audio and subtitle tracks should be set for specific movies, "Functions" for any privately developed Java code to perform certain actions not readily available by BDS options itself and "Advanced" which is best left alone but allows audio/video synchronisation, default size setting for menus (10870x1920 vs 1280x720 or 1440x1080).

3-D movies

For a short time, it looked like bluray had all the needed qualities to allow 3D movies to be recorded: sufficient space to store both left and right eye images in full colour. The problem remained with the tv sets and video projectors. Both eye images needed to be shown in quick

⁵ Resume is recognized by the project ID of your project. This is a number generated by BDS from the Organisation ID (set when you sign the JAR), Application ID and the Storage File Name – both values set on the General tab in the JAR Signing section.

succession to the designated eye, requiring still shutter glasses to be worn. Some people get headaches watching 3D that way. It did not become a big success so far and as of 2018 most major home entertainment studios have discontinued the bluray 3D format in North America and other regions. No television manufacturer still produces 3D-capable television sets. Some bluray disc players and video projectors are still capable to show 3D formatted movies – the use of a set of passive or active 3D glasses is required.

BDS support two types of 3D:

- "flat" Side-By-Side where each image is split in two halves – a left eye and right eye image. The video projector or television set must be capable of showing these 3D movies as “side by side” or “over-under”. The effective resolution is half of a normal image since the image area is divided between two images.
- "true" 3D where left and right image are both in full resolution and are interleaved where projectors and television sets shows left and right images in rapid succession.

Both types are discussed at some length in part Part 7: 3-D on page 403.

Burning discs

BDS has no built in capability to burn the final bluray disc image. It produces the \BDMV and \CERTIFICATE folders that make up a bluray disc that can be played in a set top box but it cannot burn it. You need to use a separate application such as the freeware ImgBurn or commercial Nero suite to burn those folders to a bluray disc.

If you got the BDS MX Pro version, you can make a “stamp” file (CMF) that commercial disc manufacturers use to mass-produce a disc title.

Part 2: Basic operations

Project 1: Create the simplest bluray

The Project Goal

The goal is to produce a bluray that simply plays a movie directly after inserting it in a bluray player. And if there is a second movie, start playing that one after the first. When all is done, it starts again (loop) or stops – depending on how you author it.

This is using BDS capability to create a disc image (set of folders to burn on a disc) but leave much of the work to external tools – mostly the muxer. Either the external tsMuxer or the internal BDS muxer.

There are no menus, no buttons, no frills. But the movie has chapter points that can be moved to using the remote-control buttons. In the steps below you will

- Create a bluray disc
- Fill the disc with movies, played in sequence
- Add audio tracks, subtitle tracks and chapters to movies

If you want to create this project yourself, use the source files as specified in Appendix A: The user's guide source files. Just give the project a unique name in order not to collide with the one we use below.

BDS general behaviour can be configured through the Tools > Options window. Settings specific for a project (such as which audio or subtitle stream to show by default) are set through Project > Project Settings.

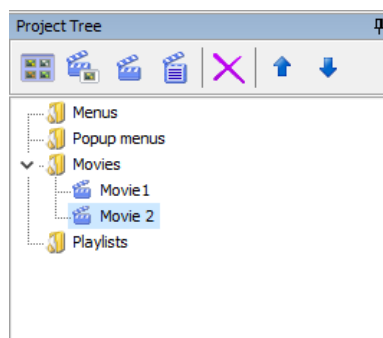
Project Elements

Our first project requires only one part of the BDS functionality: everything needed to mux a disc with video movies.

The other part of BDS, menus, buttons and navigation, are not needed and are left to the next project.

Movies

The most important ingredient of a simple bluray disc. BDS uses movie placeholders in its project tree to link to the physical files of the video movies.



You add movie placeholders in the “Movies” branch of the project tree by selecting the “Movies” branch and via right-click open a context

menu from which you select “New menu” (or press SHIFT/CTRL/V).

There is also a button that adds a movie placeholder:  .

The placeholder is all that is needed to author a BDS bluray disc if you use menus, buttons and navigation. We’re not doing that in this project, but the placeholder is still needed.

Movie streams

To create a disc it does not suffice to have movie placeholders: they also need to be filled in. They must be linked with the movie streams of the real movie.

BDS cannot work with “full movies”. It needs the individual streams. A movie has the following streams:

- one video image stream (a “silent” movie)
- one or more audio streams (different languages, music only etc)
- zero or more subtitle streams (for various languages)

These streams must adhere to strict standards imposed by the BDA – Bluray Disc Association. BDS relies on you providing the right streams – it won’t correct them. The only thing it does is combining the streams into a single complete movie in .m2ts format. The BDA requirements can be read in Appendix C: Bluray specifications on page 469. We already discussed the most important aspects in the earlier section BDS Movies – movie picture and audio requirements on page 34.

The most often acceptable formats are:

- movies in 1920x1080 or 1280x720 pixel resolution
- movies with frame rates 23.9, 24, 25, 29.9 or 50 fps (not 30 or 60 fps)
- movies in BDAV format (not MP4, MPEG etc)
- audio in uncompressed LPCM or compressed Dolby .ac3 format
- subtitles in .srt or .sup format

The BDA has set limits to each stream: in a movie can be

1 video stream

32 audio streams

255 subtitles

(see section Number of BDAV/BDMV elements on page 473)

Properties	
[-] Common	
Name	Movie1
Group	Movie1
Disabled actions	
Start Action	
End Action	
Action every second	
Scenes	1
Auto show popup	<input type="checkbox"/>
Numeric select scenes	<input type="checkbox"/>
Virtual	<input type="checkbox"/>
Allow save state	<input checked="" type="checkbox"/>
Popup menu	
[+] Remote control buttons	
[-] Streams	
Video	
Audio 1	
Sub 1	
PTP	

These streams are specified as properties in the movie placeholder. If you have “complete” movies with these streams combined in a single movie file, you need to demux them into their streams. BDS will then mux them back into a .m2ts movie file for the bluray disc.

From the properties list shown above, the important elements are the placeholder name, the “Video”, “Audio 1” and “Sub 1” specifications. As soon as an audio or subtitle link is provided, a new empty audio or subtitle property is added (BDA allows a maximum of 32 video or audio streams, 255 subtitles).

Note that some editing tools like Power Director create MPEG-TS streams rather than BDAV)). Their output is acceptable for tsMuxer (and produces playable discs for most standalone players) but not for the internal muxer if you use BDS MX and its muxer. You need to transcode the files to BDAV AVC streams.

When movies are ripped from DVD or taken from the internet, you need to transcode them in case they have a non-standard resolution required by the Bluray standards. The 720x480 (or 575) resolutions of old tv screen work for both the Academy format 4:3 as well as the DVD widescreen 16:9 format (which is also 4:3 but with stretched pixels to make the image wider).

Use one of the tools mentioned in section "Video editing and format converting" on page 497 to transcode if needed).

Some consumer video cameras deliver HD files in AVCHD format (Advanced Video Codec High Definition) . This is an interlaced format allowing a high definition movie to be stored in a bluray disc structure on a DVD disc. Bluray players recognize the format and will play the DVD as if it was a bluray. Because a DVD only has 4.3 GB of space (as opposed to bluray with 25 GB) the play time of an AVCHD movie is limited. The format needs to be transcoded into a BDA compliant format before it can be used in BDS to become part of a real bluray disc project.

DTS Atmos audio

Some video files contain Dolby True HD 7.1 audio (Atmos). When demuxed with tsMuxer a single .ac3 file results and can be used in an audio track of the movie object of BDS. When using the internal muxer you need to demux this audio (using the freeware eac3to tool) into two separate streams: an .ac3 file and a .mlp file that have the same

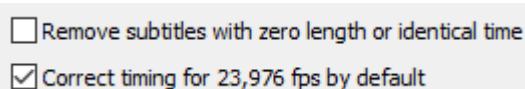
name. Then the .mlp file is specified in the audio track. The BDS internal muxer then automatically looks for the .ac3 file with the same name in the same folder.

Language audio

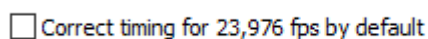
Each audio stream has an associated language. You need to specify this in the audio track property of the movie. The dropdown list of languages consists of your favourite languages as specified in Tools > Languages. If your audio files have the language code postfixed to the file name, the language setting is automatically modified. E.g. a file “main movie NLD.srt” uses the NLD (=Dutch) language as language indicator in the audio stream property.

Subtitles

Subtitles are added as property of a movie placeholder. When the subtitle specification window opens, notice that the subtitle frame rate may have the checkbox for “Correct 23.97 fps” set. Uncheck it if your movie(s) has or have a different frame rate.



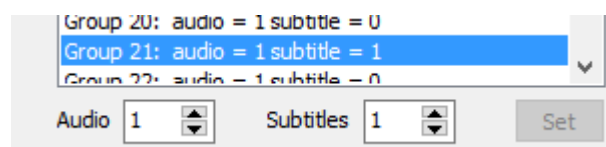
You can set the default value for this checkbox by modifying its setting in Project > Project Settings > Streams.



The size of the subtitle lines should depend on the movie resolution. The higher the resolution (e.g. movie frame in 1080 pixels high or just 720 pixels high) the more pixels can be used for the height of a title. The default is set for HD movies: the subtitle font is Arial, its size is 45p and the offset from the bottom is 30% (i.e. 30 pixels from the bottom for tsMuxer). See section Bluray disc set top player specification on page 20 for movies with different resolutions.

Again, under the Project > Project Settings > “Streams” tab you can specify for each movie what audio and subtitle track to use when running it. By default it is always the first audio track and no subtitle. But using the Streams tab setting you can specify a specific language and/or subtitle to use when the movie starts running.

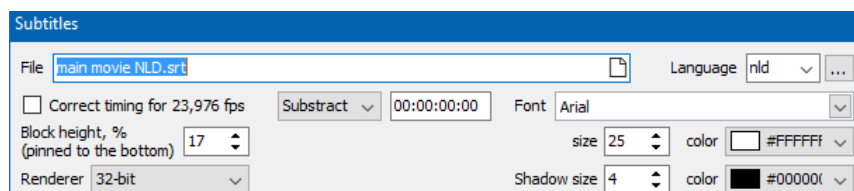
If you grouped all movies in a single group⁶ (in the movie properties) they all will use the same audio and subtitle tracks. By default BDS puts each movie in its own group so you can determine the movie behaviour on each individual movie.



⁶ movie groups are discussed later. By default each movie is in its own group.

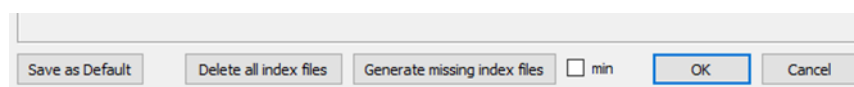
Language subtitle

Finally you need to specify the language of the subtitles in the subtitle window. The dropdown list of languages consists of your favourite languages as specified in Tools > Languages. If your subtitle files have the language code postfixed to the file name, the language setting is automatically modified. E.g. a file “main movie NLD.srt” uses the NLD (=Dutch) language as language indicator in the subtitle window.



Scenes – index files

Each movie has chapters (known as “play marks”). At minimum it has one chapter at the start of the movie. But you can specify as many chapters in a movie as you want – usually at important movie scene positions.

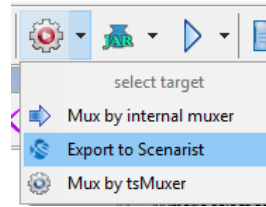


To set play marks, first each movie must be “indexed”. This is done by double clicking on the movie property “scenes”. If you want to generate index files for all movies in one go, open the Project > Project Settings > General tab and look at the bottom part. It has two buttons “Delete all index files” and “Generate missing index files”. The latter will visit all movie placeholders and generate the index file if it does not yet exist.

The “Delete all index files” and “Generate missed index files” come in handy if you want to (re)generate the scenes index files in one go (and while doing it, sometimes grab a cup of coffee – indexing is usually “fast” but for some reason some movies take several minutes to get indexed).

The index files use the same name as the movie file itself. However, please note that at BDS does not accept playmark file names that have punctuation marks in them. If present, change the movie file name by removing those characters (a movie name “This is Ken's Movie” is unacceptable: it has an apostrophe character in it. The index file that BDS tries to open simply ignores the apostrophe in the name (“This is Kens Movie”) and then tries to open a file that does not exist!

From the dropdown box for “muxer” you select either the tsMuxer or (BDS MX only) the internal muxer. Each muxer creates its own set of index files and unfortunately their play mark positioning differs slightly from one another. Switching muxers in a project will give warnings during disc creation that some chapter marks are “adjusted” to the currently selected muxer.



Scenes – playmarks or chapters


To add playmarks:

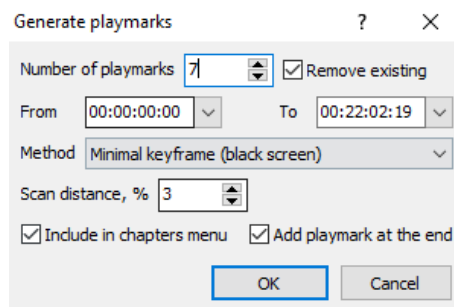
1. Double click on the “Scenes” property of the movie. This opens the scenes window
2. Use the slider to position to a point in the movie. Use the >> button to move to the next complete image (“I keyframe) or << to the previous key frame.



3. Add a playmark (or shift it or remove) by using the buttons in that scenes window. The buttons mean from left to right: add, move, delete, delete all marks.



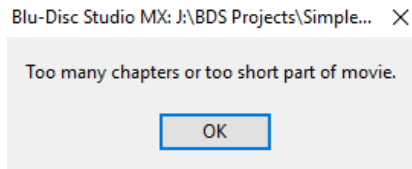
Play marks can also be generated automatically. Generation is done by clicking the “Generate Play marks” button  (Ctrl/M). This opens up the “Generate play marks” window where you can specify how many play marks you want. By default, the checkmark on “Remove existing” is set. This removes any previously set play marks for a clean start. If you want to keep those and generate additional ones, uncheck this option.



Note that the “From:” and “To:” textboxes indicate the start and end of the movie and not some small part of it.

If too many play marks are specified for the movie length an error message is shown telling you too many chapters are specified or that the movie is too short for the specified number. This error never occurs on manual positioning of chapter marks.⁷

⁷ This error may also occur if for some reason the end time value in the :To:” time box is modified and its value very close to the starting value in the “From:” box



The “Scan Distance” rules the accuracy of the positioning. The larger the value (ranges between 3% and 49%), the more freedom BDS has to position the marker according to the specified method. It is best to play with the settings to see what you like best. At 49% you will see that the chapter marks may be very unevenly spread out. When using 3% the play marks are set almost at fixed intervals.

The result is shown as set of play marks under the video image and a list is shown to the right. Check boxes indicate whether the play mark will show up in a chapter menu (which we won’t make here, so it doesn’t matter). The end-of-movie marker should be unchecked (if ever you want a chapter menu the final frame as chapter makes no sense to show).

When you create play marks, the tsMuxer creates two index files (.idx, and .inf) in the project \films folder. The internal muxer creates other files (.mes and .mom) also in the \films folder but also in the application folder found at %APPDATA%\DVDLogic\Muxer. The information in these files is used when showing the chapter marks later or when the disc is created. The ones in \films are not used.

The chapter information is stored inside the project.bdmf file and is kept, even if you replace the original movie file with another one.

- If the replacement has the same name, the index file exists but with incorrect chapter positioning. Re-specify these chapters.
- If the replacement has a different name, BDS will create a new index file with that movie file name (the “scenes” counter reads “1” when done)..

Important for tsMuxer users

Recent V4.6 (as of subversion.2074) releases have modified the switches used for tsMuxer to avoid some problems for some users. In doing so, they create problems for other users, especially those using Cyberlink PowerDirector for their .m2ts file creation.

This tsMuxer switch change may result in chapters that, although set, always let the movie restart from chapter 1. In order to avoid this (if you notice this behaviour), goto Project > Project Settings > Streams tab and look for the tsMuxer box. In the empty textbox “video” add the words “insertSEI, contSPS” (without the quotes). “OK” the window and proceed.

This setting is kept only during the use of this project and is not kept if you reopen the project. If you want to keep it, you can make it a default by doing the following:

- Open BDS without opening a project
- Goto Project > Project Setting > Streams tab

- Add “insertSEI, contSPS” (without the quotes) in the tsMuxer box in the “video” labeled text box
- Click on the “Save as Default” button
- Close BDS
- If BDS wants to save the modified project, click “No”

Project Settings

Apart from the movies, some project specifics need to be set before you can create the disc.

Sign JAR

Whether or not to sign the JAR file depends on whether you defined a signature in the general Tools > Options settings. Signing is required when you want a disc to resume playing when it was interrupted earlier. Signing is not needed for this project.

Output of disc creation

You can set the output location the moment you mux your project into a bluray disc, but you can also specify the output folder location now in the “Output Folder” text box.

Functions

If you are going to write any Java code to have BDS perform something not readily available out of the box, you define these "User Defined Functions" (UDF) and "User Defined Parameters" in the "Functions" tab. How to write such UDF programs is the topic for Part 5: Advanced Operations and programming on page 285. They are not needed for this project.

Advanced

Usually best left alone unless you know what you're doing.

Graphic mode - sets the screen resolution for graphics layer where the menu will be drawn to. It's recommended to set the graphical mode equals to the resolution of the movie. WARNING: for 3D projects always use 1920x1080.

Animation engine FPS - sets the frame rate for the animation engine. We recommend using the frame rate of the main movie/menu. Animation is discussed in Part 4: Animations in page 247.

D.I.Y. adventure

If you think you know enough to go out on your own, try to create the project on your own. Otherwise, take step 0 (Ready to run) to get a more or less completed project. Or follow the step-by-step instructions to build the project.

But if you want to venture out on your own, just keep the following in mind:

- Add two movies to the project. All video, audio and subtitle tracks
- Connect the first one with the second one.

- Start the disc by playing the first movie.


Step 0: Ready to run

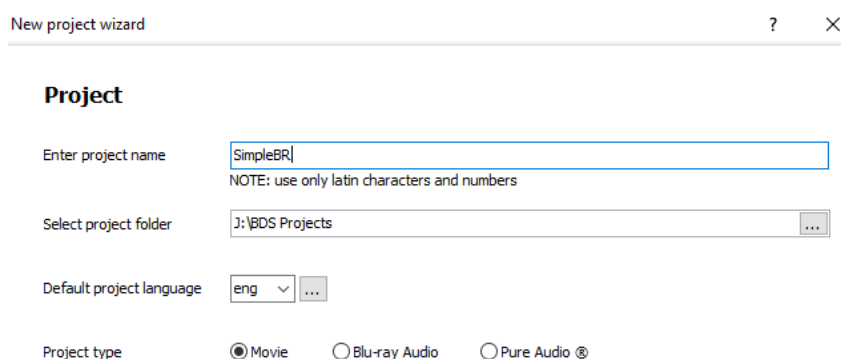
If you do not want to go through the following steps creating the “SimpleBR” project by yourself, you can run it from an already made project.

1. Create a top folder that will contain all projects that are described in this book. E.g. J:\BDS Projects
2. Copy the folder \Sources\Projects\SimpleBR as subfolder of the folder that is to contain all BDS projects (e.g. J:\BDS Projects\SimpleBR)
3. Copy the two movies “Coasts”, and “Australia” and their audio streams from \Sources\Movies\demuxed to the project’s \films folder (e.g. J:\BDS Projects\SimpleBR\films)
4. Copy the two subtitle tracks for “Coasts” and “Australia” from \Sources\Movies\Subtitles to the project’s \films folder.
5. Open the project by double clicking on the project’s “SimpleBRt.bdmd” file (e.g. J:\BDS Projects\SimpleBR\SimpleBR.bdmd)
6. Click the “mux” button and rebuild the disc image (in the project’s \Output folder)

Step 1: Create new project

Before going to the next step, make sure you set up the BDS program with your desired preferred settings as described in the previous chapter.

A new project must be created first. Go to the menu File > New or the “New Project” wizard (icon ) and specify a project name (e.g. “SimpleBR”). The project folder follows from the set default project folder (you can change it here for this project if you wish).



New project wizard

Project

Enter project name: SimpleBR
NOTE: use only latin characters and numbers

Select project folder: J:\BDS Projects

Default project language: eng

Project type: ☒ Movie ☐ Blu-ray Audio ☐ Pure Audio

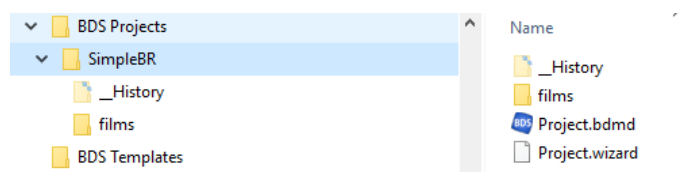
Click repeatedly on the “Next” button (skipping windows asking for (bonus) movies and (popup) menus) until you can click “Finish” to complete creating the project.

The created project folder has the Project.bdmd file as BDS project file. Its backup is stored in the hidden _History folder. If you wish you may rename the project file name to something more significant (like SimpleBR – use File > Save As... to do this).

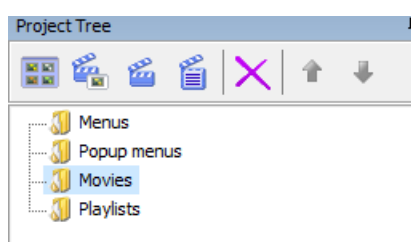
To be organized, the created project folder is manually enhanced by adding the folder

<default project folder>\SimpleBR\films

where we will store all demuxed movie files. These consist of video, audio and subtitles. The resulting set of files is shown in the File Explorer of Windows as shown below.



The resulting project has a minimum of entries, as the Project Tree window shows.



Step 2: Include the movies

The \films folder is populated with the movie files. For this user's guide we use two files, "Australia" and "Coasts". Both have one video (<name>_movie.264) and two audio tracks (one <name>_live.ac3 file with "live" sound, the other <name>_music.ac3 as a "music" track). They can be found in the folders of the downloadable BDS kit.zip file:

- Video and audio in \Sources\movies\demuxed
- Subtitles in \Sources\movies\subtitles

Demux movies

If you use the \demuxed folder objects, you're set and can go to the next section.

If you work from the "combined" movie files (usually with file extensions .mp4, .ts or .m2ts), you need to demux these first into audio and video tracks, resulting in the separate object files. Demuxing can be done using the tsMuxer GUI or tools like TS Doctor (Tools > External tools > Demux).

The demuxing produces one video file (.264) and two audio files for each episode (.ac3), one with live sound (mark it as "live" in the file spec) and one with music (mark it "music" in the file spec).

Subtitles

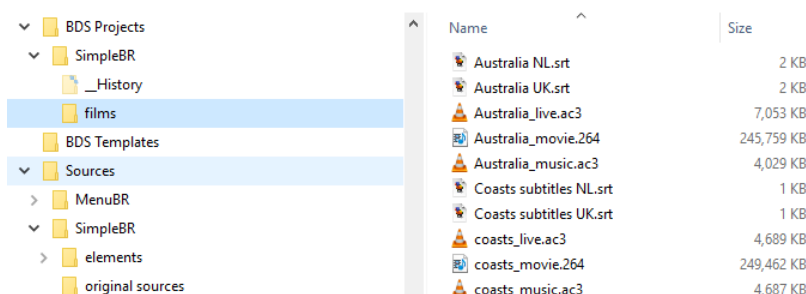
There are also two subtitle files (.srt) for each movie. One in Dutch (marked "NL") and one in English (marked "UK"). They only contain texts like "ondertitels Nederlands" or "subtitles English" repeatedly so you can distinguish between them. There is no talk in the movie, so

the subtitles are a bit of an overkill but here to get acquainted with subtitles.

With the two subtitle streams, the total number of streams is 2 video + 2x2 audio + 2x2 subtitle files: a total of 10 streams.

Contents of \films folder

Move these demuxed files into the project's \films subfolder.



The screenshot shows a file explorer window. On the left, the project tree is expanded to show the 'films' folder under 'BDS Projects'. On the right, a list of files is displayed with their names and sizes.


Name	Size
Australia.NL.srt	2 KB
Australia.UK.srt	2 KB
Australia_live.ac3	7,053 KB
Australia_movie.264	245,759 KB
Australia_music.ac3	4,029 KB
Coasts subtitles.NL.srt	1 KB
Coasts subtitles.UK.srt	1 KB
coasts_live.ac3	4,689 KB
coasts_movie.264	249,462 KB
coasts_music.ac3	4,687 KB

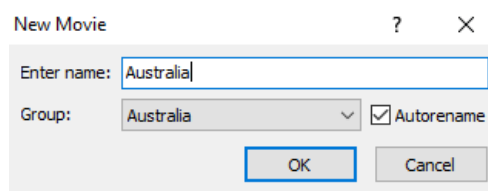
During the disc building process, the files of each episode will be muxed together into a single .m2ts file where video, audio and subtitles are recombined.

Add movies to their project movie placeholders

We need to add the movie files to the BDS project. For two episodes we require two movie placeholders in the project.

To add a movie placeholder to the project, go through these steps:

1. Open the Project Tree window (View > Project Tree (F3))
Right-click on "Movies" and select "New Movie" (SHIFT Ctrl/V) or click on icon .
2. Give a useful name to the movie you're going to add and click OK (this will add the movie entry to the project window but also show it on the Structure window (F7)).
In our example: "Australia" and "Coasts" are added.



Repeat these steps for the other movie, "Coasts".

These steps are sufficient to add a movie placeholder. These placeholders are needed to define the entire menu navigation. What movie is actually represented by the movie placeholder is specified in some of the placeholder properties.

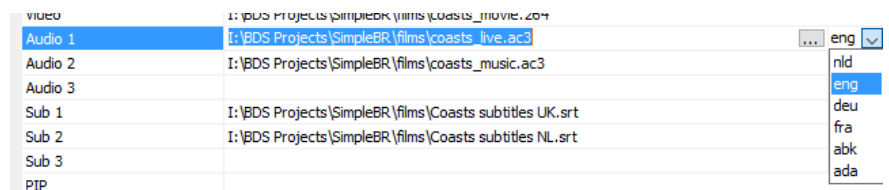
Because we don't have menus in this project, we must continue to provide values to these movie placeholder properties. These values are the file specifications to the video, audio and subtitles files of each movie.

3. Select a movie placeholder
4. Open the “Properties” window (View > Properties (F11)). This will show the properties of the movie title.
5. Under the section “Streams” select the video stream. This will show the “...” button at the far end at the right of the text line. Click on this to open a navigation window and find the compliant video stream (“Australia.264” and “Coasts.264”) and select it.

The file must have the right file extension (like .264) or otherwise it will not show and cannot be selected even if it is a proper file.

Add audio tracks to the movie placeholder

6. Repeat the previous step for any of the audio streams to go with the video and indicate the audio language. After “Audio 1” is used of the movie properties (for the “live” track), automatically a blank “Audio 2” stream is shown for an additional language (in our case for the “music” track).
- 7.



Select the language code for each stream. The language codes to choose from are those you selected in section “Step 2: Specify your languages” on page 42 earlier. For your own sanity and that of the future viewer of your disc, keep the order of language tracks the same for all movies on the disc! For example, English audio in the first track, Dutch audio second, French audio third. For all movies.

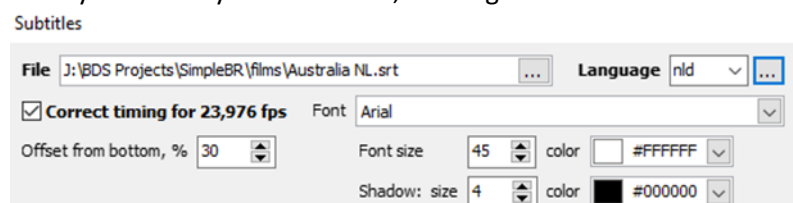
(In our project the audio is language-free audio. For their “live” and “music” tracks, just pick two languages or twice the same).

Under the menu “Tools” you find a WAV mixer that allows you to specify a separate WAV audio file for each channel – from mono 1.0 to Surround 7.1

Ensure the audio is sampled at 48 kHz (not 44.1 kHz used for audio CDs. This applies to Dolby Digital/ac3 as well as LPCM/wav. Some players may accept 44.1 kHz, others do not produce sound. Only 48 kHz is BDA standard.

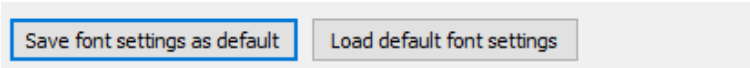
Add subtitles to the movie placeholder

8. Repeat the steps for any subtitle stream in the movie properties “Subtitle” entry. This will open a “Subtitles” window with a “File” selection box where the .srt or .sup files can be selected. For .sup titles you can only select the file, nothing more.



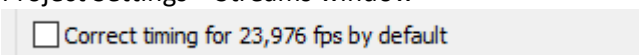
For .srt subtitles additional changes can be made:

- a. You can indicate subtitle font, size and color and preview how that would look (button “show preview”). To set the font and size for all subtitle files in the project, click on “Save font settings as default” or “Load default font settings” button at the bottom of the window.



Because there is only one set of defaults, “Save” saves them, and “Load” loads them. You have no choice in sets of defaults. Any manual change can be restored to the set default values by clicking “load default font settings”.

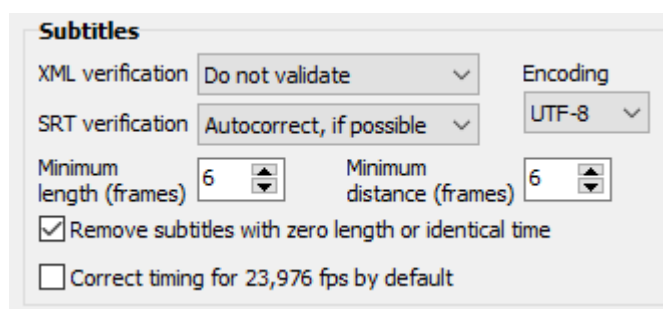
- b. It can have a shadow (size 0 means no shadow) and you can specify the position (closer to the bottom or higher up in the picture).
- c. Check or uncheck the “Correct timings for 23.976” if your movie is or is not recorded at 23.976 fps. The value can be set for all subtitles of the project by (un)checking its value in Project Settings > Streams window



In our project with 25 fps uncheck the correction for 23.97 fps (the used comma is really a decimal point – the Europeans and Americans swap the use of comma and full stop).

- d. Finally, you specify the language of the subtitles. Available languages were set previously in Tools > Languages. For consistency keep the order of language subtitle tracks the same for all movies on the disc! Click “OK” to complete the settings.

On a project basis many subtitle properties can be set with default values by setting them in Project > Project Properties under the “streams” tab and the “subtitles” section.



Note: if you have a standard definition (720x480 or 720x572 pixels – DVD resolution) movie with subtitles, you must reduce the font size to 25 points (or less) to fit the titles properly on the screen which is only 480 or 576 pixels high. For suitable size of the subtitle depending on the resolution, see section Blu-ray disc set top player specification on page 20.

Note on subtitle text: If subtitles were ripped (e.g. with SubRip) ensure they do not contain invalid characters. Usually these are diacritical characters not recognized properly ending up as “non-printable” characters. That will cause “Meta Data generation failed” by BDS at the point where the subtitle and movie are muxed into final output. For example:

```

168
00:16:07,657 --> 00:16:10,148
<i>¡Atención!</i>

```

Here the original Spanish language subtitle should read “¡Atención!” but was ripped incorrectly.

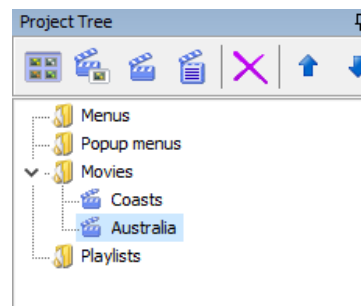
You now have added one movie to its placeholder with all its trimmings (properties). Repeat the previous steps for additional movies to the project, as long as the total size of the movies does not exceed the 25 GB disc capacity of a bluray BD-25 disc and leaves some room for buttons and (generated) Javas and indices. For our project we need to add properties to one more movie: “Coasts”.

The streams properties of the movie “Australia” are shown below.

Video	J:\BDS Projects\SimpleBR\Films\Australia_movie.264	
Audio 1	J:\BDS Projects\SimpleBR\films\Australia_live....	eng
Audio 2	Projects\SimpleBR\films\Australia_music.ac3...	nld
Audio 3		eng
Sub 1	J:\BDS Projects\SimpleBR\films\Australia subti...	eng
Sub 2	J:\BDS Projects\SimpleBR\films\Australia subti...	nld
Sub 3		eng

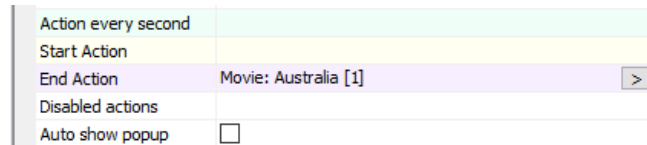
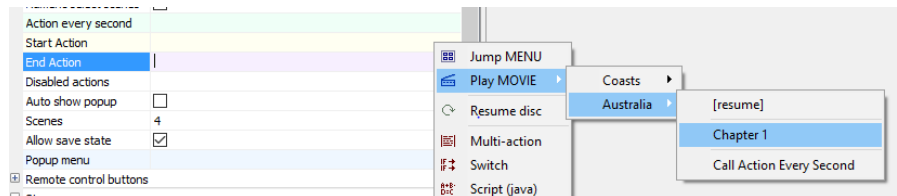
If by error you added an audio or subtitle track too many, you can delete it by selecting its row and pressing the keyboard <delete> key. If you want to rename the movie, simply select the “Name” field in the property window of the movie and type a new name.

The Project Tree window will show all movies added. The branches for “Menus”, “Popup menus” and “Playlists” remain empty.



What comes after the first movie?

In the movie properties of the first movie “Coasts” there is also an “End Action” property. Specify here what is to happen after the movie ends. In our simple case, we start the next movie, “Australia”. Click on the “>” button of the “End Action” line of the “Coasts” movie properties window. A list of possible actions pops up. Select the right action: in our case play the “Australia” movie starting at the beginning (always chapter 1).



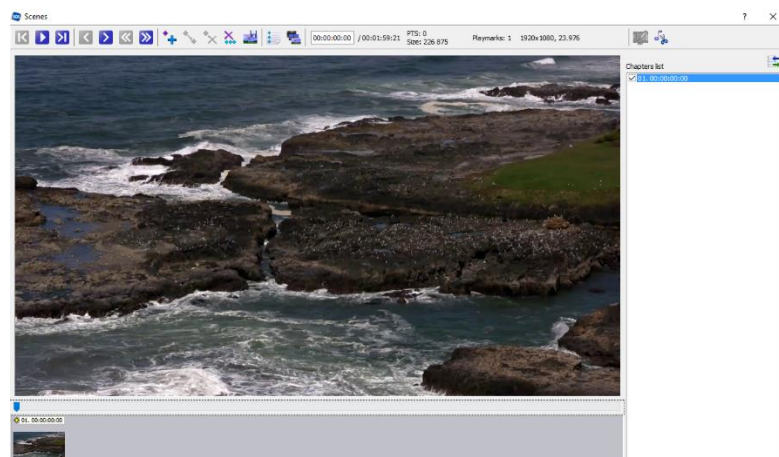
Repeat these steps for all movies so that the end there is a chain of one movie starting the playback of the next movie. The last movie to be played either leaves the “End Action” empty (disc stops playing) or specifies the first movie again in which case you created an endless loop.

There is also a “Popup Menu” property entry. This is not used in our simple disc – there is nothing (no menu) to pop up.

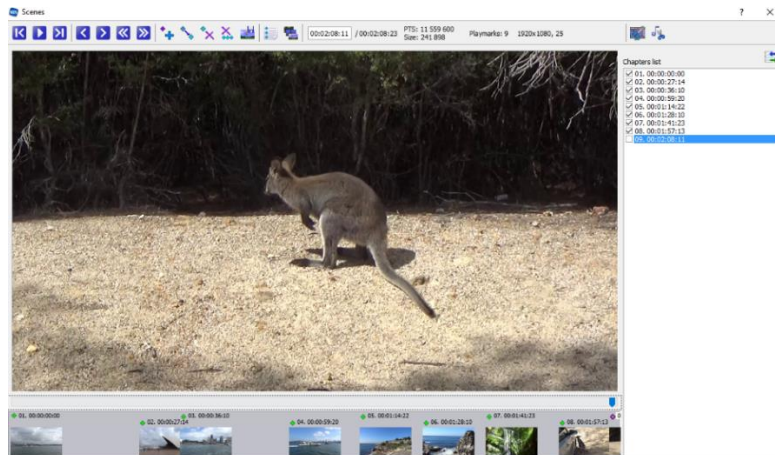
Step 3: Creating chapters

Chapter or Play marks are created by the muxer you use. The tsMuxer or internal muxer do it slightly different. This is the time you make your choice and stick to it for the rest of the project.

To generate play marks, open the movie’s Properties window. If closed, open it using View > Properties.



1. Add some chapter marks manually or generate them automatically



When satisfied, close the chapter window by clicking the “X” “close window” button at the top right-hand corner. There is no other close mechanism.

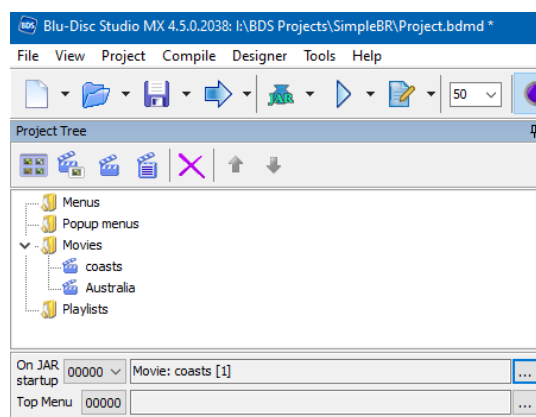
2. Note that upon generation you get always 2 more play marks than specified: the one at the start and at the very end are also counted. This number is reflected in the final “Scenes” property of the movie.

Auto show popup	<input type="checkbox"/>
Scenes	9
Allow save state	<input checked="" type="checkbox"/>

Step 4: Build the disc

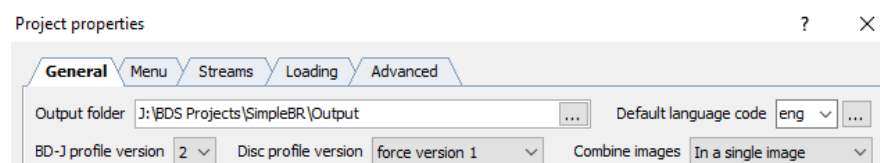
Final settings

Before a disc image can be created, you must specify where to start playback if the disc is inserted in a set top player. This is specified in the Project Tree window in the “On JAR startup” box at the bottom of the window. Specify as the first movie, “Coasts”. The [1] indicates that playback starts at the first chapter (i.e. at the beginning of the movie).



During the build of the disc, BDS requires all movies to have their chapter index. If you never created them during authoring, the files are created during disc build which will then take that much longer to complete.



Specify the output location for the disc image file by opening the project properties window (Project > Project Properties window (Ctrl Shift F11) using the General tab). Initially it is a blank text box. Specify a folder (such as “output”) under the project top folder. This is the only time BDS will create the output folder if it does not already exist.



The project is created by muxing all objects of a movie together into a single .m2ts file and store the resulting files in the output folder \Output\BDMV\STREAM. The output folder contains additional folders and files created by BDS to make a BDA compliant disc image.

Remember whether you want the discs to be signed or not. (Un)check “Sign the Jar” at Project>Project Properties>General tab and Tools>Options>Compile tab “Validate SIG” box.

Create the disc files

The muxing starts by clicking the mux button. Depending on your muxer choice, it is shown as a cogwheel  (tsMuxer selected) or an arrow  (the MX internal muxer selected) button on the menu bar.

When you click on it, the Project Properties window opens to allow a final change in output folder (preset with the value you specified in Project > Project Properties>General tab). If blank, BDS will create a \Output folder in the current project tree.

When satisfied with choices made, simply click the “OK” button.

The muxing process is logged in the Log Window. It pops up if it is not docked somewhere in the BDS interface. The log results can also be viewed later in the project \Log subfolder log files.

Muxing a project into a disc image consists of three steps.

1. Compile all Java-related code to ensure all files are present and a Java JAR file can be made as part of the disc. This includes a check on all (menu) video, audio files needed for this disc. In addition (when preferences are so set) the disc indicates an organization ID signature. This step is identical to what happens if you press the Simulation button or the “JAR” button on the BDS menu bar.
2. Mux the movies. This may take a considerable time. The video, audio tracks and subtitles of each movie are merged into a single .m2ts movie file. BDS writes them into the \Output\BDMV\STREAM folder of the project.
The muxed movies can be played individually by a video player program such as VLC. This player is useful to check output movie files individually for having the assumed audio tracks and subtitles. A bluray player application can check the entire produced disc.

If lengthy movie files mux very quickly, you can rest assured that some files for the muxing are missing (possibly renamed or removed?). BDS tries to find these situations (from V4.1 onwards) and issue error messages, but certain situations it cannot prevent. In that case the dreaded uninformative “Line:0” error will show during muxing. All you can do is:

- check each movie and menu item referring to files that do exist and are accessible.
- check that the filenames contain only visible and printable characters that may be acceptable to BDS but not to tsMuxer if this muxer is used

A quick way to locate file names with not-acceptable characters is using the Windows Command Console (right click on the “Start” icon and select “Command prompt”. Set the folder location to the \films folder of your project. (you must first set the disc drive (e.g. J:) and then the folder (BDS Projects\SimpleBR\films)


Issue the command `dir/s >files.txt`. This produces a folder listing in a text file files.txt. Open it with Notepad and find all filenames listed s.txt file. Any occurrence of “?” means the name contains a non-printable character.

```
2019-07-06 09:30      7,128  Wild New Zealand.srt
2019-06-08 09:49    23,438  ?Holst The Planets with? Brian Cox fixed.log
2019-06-08 09:49 2,183,467,200 ?Holst The Planets with? Brian Cox fixed.m2ts
```


3. The final build step is updating all links between menu buttons and movies on disc. It is in this phase that “Line: 0” errors may occur again. In this case most likely one of the files that make up the movie (audio or subtitles) may seem corrupt to BDS. The last movie listed in the “Updating” list is the likely suspect. See section “Possible problems ” on page 69 for more details.

In the worst case, you may need to create a test project and load one movie file each time and see if it processes. Repeat this until you find the bad one and correct it.

Using the MX internal muxer

The “Mux by internal muxer” uses the arrow menu icon () and does all its output in the log file window and a mux window of its own.

Using tsMuxer

The “Mux by tsMuxer” uses the cogwheel menu icon (). BDS opens a progress log window (if it is not a docked window) showing each step along the way, supplemented by additional come-and-go tsMuxer DOS windows for each movie it muxes until done.

For each movie it muxes, it creates a small textfile *movienamename.inf* that you can open with a plain text editor like Windows Notepad and see what information tsMuxer used for the muxed movie. This info is also shown in its console window but disappears when that window closes.

If you get an error message about incorrect frame rate, then open the appropriate INF-file in Notepad (Movie.264.inf for Movie.264) and make sure it contains correct frame rate:

```
Network Optix tsMuxer. Version 2.6.11. www.networkoptix.com
Stream type: H.264
Stream ID: V_MPEG4/ISO/AVC
Stream info: Profile: High@4.1 Resolution: 1920:1080p Frame rate: 23.976
Stream lang:
```

If it is incorrect (i.e. 30 fps or 60 fps) , transcode those movie files to an acceptable frame rate.

Log window

The project log window indicates progress in the processing by BDS. The log window can be docked in the BDS interface or be undocked. BDS will open it when its disc building phase starts.

BDS also creates a subfolder \Log in the project folder where you can revisit these log files (named by disc build time).

```
Log

16:43:01 - Finished (warnings: 2, notices: 7)

mode: tsMuxer

16:55:28 - Saving...
16:55:29 - Preparing...
16:55:29 - (notice) Popup Menu not specified in movie "Space Race"
16:55:29 - Adding video asset for "Space Race"
16:55:29 - Adding audio asset #1 for "Space Race"
16:55:29 - Adding audio asset #2 for "Space Race"
16:55:29 - FL BDID = 00001 (movie "Space Race")
16:55:29 - [warning] End Action not specified in movie "Crosscut"
16:55:29 - (notice) Popup Menu not specified in movie "Crosscut"
16:55:29 - Adding video asset for "Crosscut"
16:55:29 - Adding audio asset #1 for "Crosscut"
16:55:29 - Adding audio asset #2 for "Crosscut"
16:55:29 - FL BDID = 00002 (movie "Crosscut")

16:55:29 - Preparing menu...
16:55:29 - Optimizing...
16:55:29 - Step 1/4 ...
16:55:29 - Step 1/4: done
16:55:29 - Step 2/4 ...
16:55:29 - Step 2/4: done
16:55:29 - Step 3/4 ...
16:55:29 - Step 3/4: done
16:55:30 - Step 4/4 ...
16:55:30 - Step 4/4: done

16:55:30 - Starting sign process...
16:55:30 - [warning] You use the default values for Organization IDs. See help: Settings
16:55:30 - OrgID: 7fff2222, AppID: 4000
16:55:30 - Signing...
16:55:31 - (notice) OpenSSL not specified/installed - sign verification skipped
16:55:31 - Sign process finished

16:55:31 - Step 4/4: done

Source graphic size: 0 px
Result graphic size: 4 px

16:55:32 - (notice) changed events: 187, removed events: 0
16:55:32 - (notice) changed events: 107, removed events: 0
16:55:32 - Muxing movie "Space Race" (BDID: 1)
Network Optix tsMuxer. Version 2.6.12. www.networkoptix.com
Encoding H264 stream (track 1): Profile: High@4.1 Resolution: 1920:1080p Frame rate: 23.976
```

Errors are indicated in the on screen Log window in blue (warning) and red (real error). Some of these indications may be ignored at this stage (or always, depending on what you do).

In the case shown above, a blue warning states that no popup menu is specified for either movie. Correct. Not used in this project. Can be ignored.

Another blue warning you may ignore is the “changed events”. This refers to a slight modification of timings of a subtitle. If a movie has 187 of those events, 187 subtitles were shifted. You may suppress

these messages by changing the subtitle XML setting in Project Properties > Streams > Subtitles.

A red error indicates that there is no “End Action” specified for the last movie. Correct. The disc stops playing. If you want it to restart from movie “Coasts”, provide this as “End Action” to the second movie “Australia” and create the disc again. Since the movies are not changed, it suffices to recompile the Java programs in the JAR file and have the new copy be placed in the disc output folder.

But you may also restart the entire muxing process – it will overwrite everything it made earlier in the output folder. It just takes much longer than recompiling and replacing the JAR file.

Some red errors may occur when you switched muxers: it warns you that the muxer had to adapt the chapter timings to fit its own requirements. The differences may be 1 or 2 seconds to the original setting with the other (now not in use) muxer. In general: do not change muxers in mid-flight!

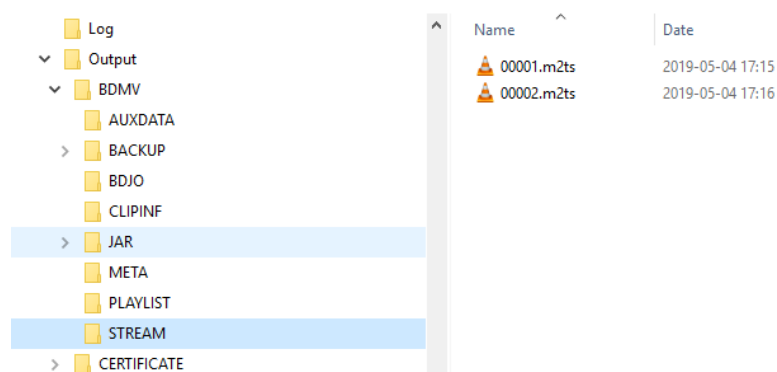
Another red error states you did not specify who made this disc, an organisation ID. That is important for users that want the “Resume disc playing” function to work. If you don’t want to resume, this type of error message can be switched prevented through Project > Project Settings > General tab and the unchecking of “Sign the JAR file”.

A blue warning indicates we did not sign the project. OpenSSL used to verify the JAR signature. This check is enabled by default to force the user to use it just in case. It can be disabled in Tools > Options > Compile tab and unchecking the “validate” checkbox. Physically, it unpacks the JAR-file, then unpacks META-INF\SIG-BD00.RSA and compares the data from it with the data in the JAR signature settings. If the signature is changed it indicates that someone has tampered with the files: any change (even a single bit) modifies the signature value.

Note: sometimes a red line indicates no error. The application uses some words like “error”, “unexpected” etc in a message to indicate it needs to be in red. So a perfectly fine “Unexpected Christmas” movie title might light up in red – though there is nothing wrong.

Other than those “errors”, we’re fine!

The bluray compliant file structure is written in the folder indicated in the specified Output folder:



with the actual video movies stored in the STREAM folder – just as large as their original source input (before they were demuxed). All other folders are created and populated by BDS.

Possible problems during muxing

Check Appendix D: Errors occurred: possible causes for some that may be less trivial to correct.

Step 5: Test running the disc

You can now play the disc. Either by taking your chances, burn it on a BR-writeable disc and play in it a set top player. As long as you're testing your project, use re-writable discs. This may save you a lot of "toaster" discs if the authored disc turns out to be not exactly right.

Software disc player

You can also check the results first with a software player on your pc. This allows you to correct any missing items or wrong behavior without losing time on burning a disc. PowerDVD is such a software disc player. It allows you to set the \output folder (and enables its "OK" button as proof that folder contains bluray compliant folders) and play the produced disc from the folders BDS created.

From experience I know that some software players have their own quirks and don't always work as they should. The physical disc in a set top player is the definitive proof!

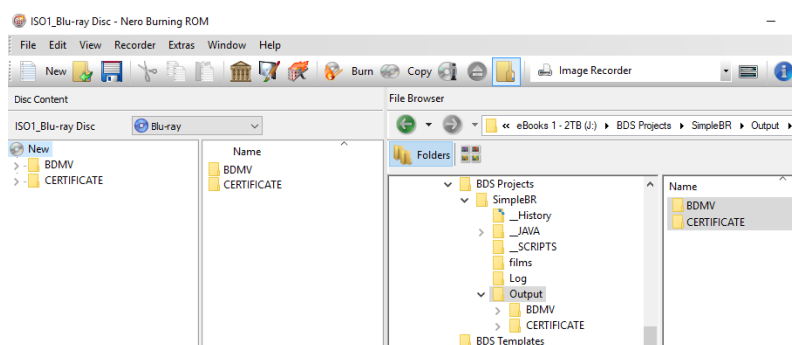
Create ISO file as disc image

Some software players only accept a real disc to be played. Or an .iso image of such a disc.

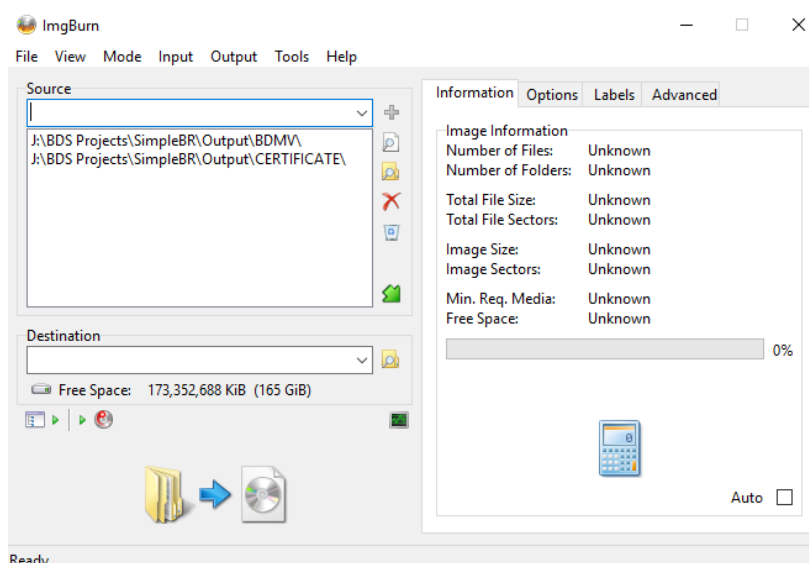
Virtual CloneDrive can open such an .iso file and emulates it as if it were a physical disc. Therefore Virtual Clone is known as a disc emulator.

To create an .iso image for the folders produced by BDS, you can use

- Nero BurningROM: set device to "image recorder", use the BD Video templates and select "bluray (iso)" option. Drag the \output folders to the Disc Content region.

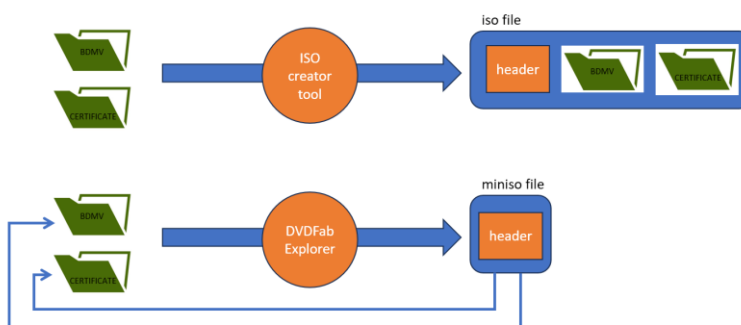


- ImgBurn freeware software (select “Create image file from files/folders” from the mode EZ-Picker). The BDS produced \output subfolders \BDMV and \CERTIFICATE are specified as starting folder, using data type Mode1/2048, UDF 2.50 file system, recurse subdirectories). Indicate a hard disc location to write the SimpleBR.iso file to.



Create a Miniso file as disc image

Miniso looks like an .iso image but only contains the headers of what a full .iso image would have. The folders that are part of the .iso image are not part of the miniso. Instead the .miniso file points to the real folders. It is therefore very space effective.



Miniso is a format created and supported by DVDfab Explorer (see section "Virtual disc drives" on page 502 for download addresses). This freeware tool creates this .miniso file and can mount it as a virtual drive which then looks the same as if a real .iso file was mounted (like VirtualClone would do).

You do need DVDfab Explorer to create the .miniso file. Although (through file rename to .iso) it can be mounted by other virtual disc drive software such as Virtual CloneDrive, this does not seem to work properly. All files shown look the same, but a “copy disc” to a bluray results in a disc that doesn’t work in set top players.

Step 6: Burn the bluray disc

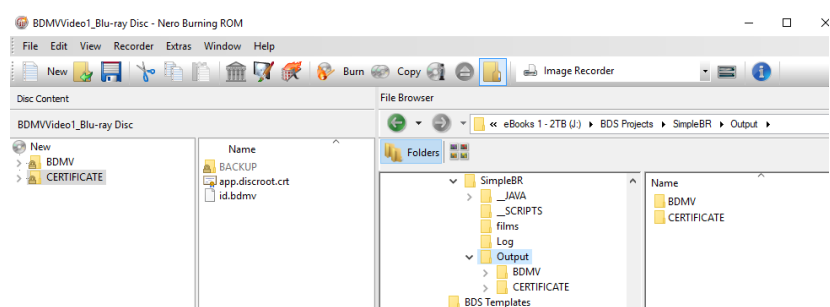
Test run using a set top player

The ultimate test is to burn a bluray disc .iso file or the \output folder of BDS to a BR-25 or BR-50 (re)writable disc and upon completion play it in a “real” bluray player. Rewritable discs are preferred if you test the disc: they can be reused after changes have been made until the disc is to your satisfaction.

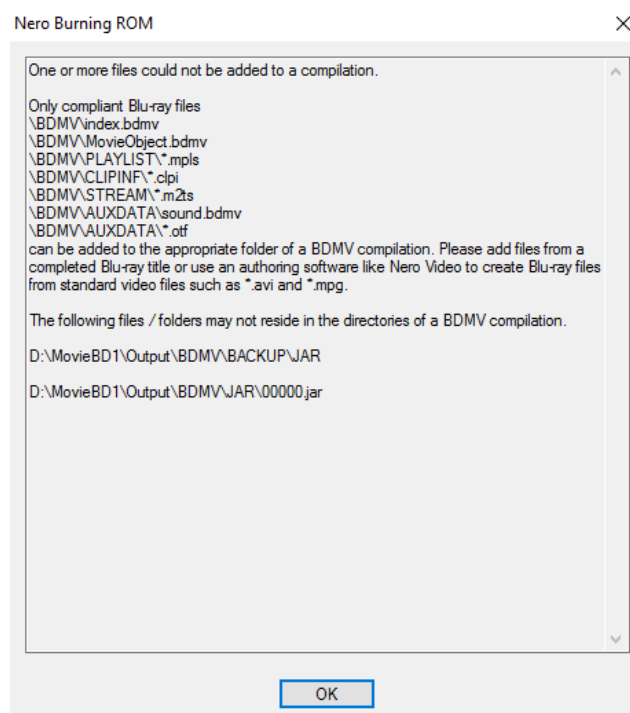
Burn folders

To burn the disc, you can use Nero BurningRom or freeware ImgBurn.

- Nero 12 (or later): use the “bluray” templates and select “BDMV video” template. Upon opening, drag the folders in BDS \output folder in the file browser to the disc content left-hand part of the interface.

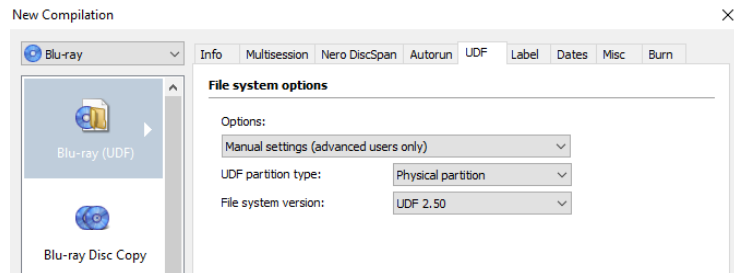


This may produce some warnings that can be ignored as the burned disc will play:

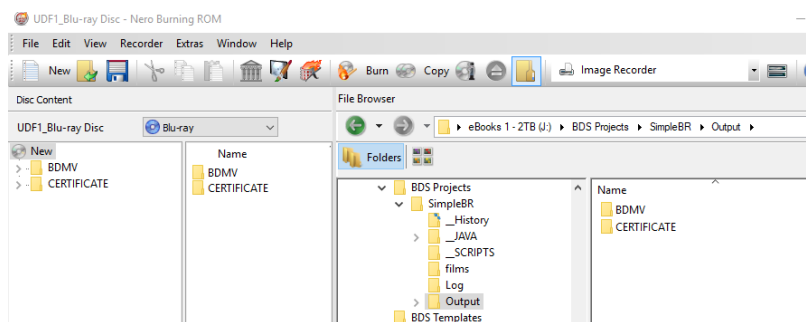


To avoid these warnings, you can also burn the disc folder using the “bluray” option, select the “bluray (UDF)” template, select the

UDF tab at the right and select “manual settings” and indicate “physical partition” and “UDF 2.50”.

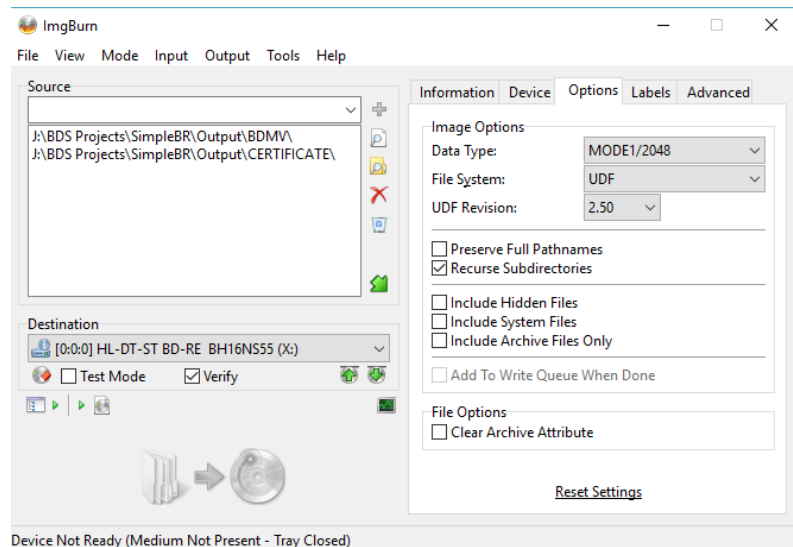


Once the “Disc Content” window opens, you drag the two main folders BDMV and CERTIFICATE from the File Browser area to the disc content area and start the burn process.



Note: It is my impression that using this UDF V2.50 method produces a disc that switches much quicker between menu items and movies than the disc made using the BDMV template.

- ImgBurn: select “Write files/folders to disc” (if this does not show, select the menu option Mode > EZ-mode picker). From the next screen:
 - select the “Options” tab and fill in the Image Options box with:
 - Data Type specify “Mode 1/2048”
 - File system UDF
 - UDF Revision: 2.50
 - select the bluray drive in the “Destination” field
 - check the “verify” box if you want to check the disc burned.
 - Select the “labels” tab and specify a disc label.
 - Burn the disc



Burn iso file

When you collected all BDS folders into a single .iso file, you can use both Nero or ImgBurn to burn a disc:

- Nero: Use menu option Recorder > burn image to transfer the contents of the .iso file onto a physical disc (*do not* use the option New > Bluray (ISO) as this is a way to create a regular .iso file from folders: not to burn them onto disc)
- ImgBurn: select “Write image file to disc” (you may have to select the creation mode first through Mode > Ez Mode picker).

Burn a mounted virtual drive

If you created an .iso (or a .miniso renamed to .iso) file you can mount that image as a virtual drive (using tools like Virtual CloneDrive or DVDFab Explorer). The .iso file will then become a drive of its own (e.g. M:\)

You can use Nero or ImgBurn to copy the contents of that virtual disc to a physical disc:

- Nero: select New > Disc copy
- ImgBurn: don’t as it requires a two step (time consuming) method half of which you already did:
 - copy disc to iso
 - burn iso to disc

The first step you already did, the second step is identical to the earlier described “Burn iso file”.

The alternative is to burn the folders you find on the virtual drive. These are \BDMV and \CERTIFICATE. How to do this is identical to what’s described in section Burn folders on page 71.

3D Warning

As will be described in Part 7: 3-D on page 403, 3D bluray discs have special SSIF files that indicate left and right eye frames. You cannot simply copy folders from a mounted .iso disc or those produced by

BDS as this does not show the \BDMV\SSIF folder files. You need to create a .miniso file and copy the resulting.iso file as image to a disc as described in section Burn iso file on page 73.

Step 7: Problems with the final disc: some possible solutions

Running the created disc may uncover navigational errors in the menu transitions if they were not caught during simulation. These can be corrected in BDS and the entire disc can be recreated and re-checked.

If some movies do not play properly in a player or the entire disc refuses to play, check:

- Does the muxer fail to produce a disc? Check if all movie files restrict themselves to characters “A”-“Z” or “0” to “9”. Other characters may not be acceptable to tsMuxer but its problem shows only when it fails.
- Has the project property “disc profile version” been set to “same as BD-J”? Retry using “force version 1”. Some BR players won’t accept versions other than 1.
- Has the disc a “signed” JAR? And did you incorrectly change the JAR settings?
Go to Project>Project Settings>General tab “Sign the JAR” uncheck “Sign the JAR” and remux again. (Press “Save as Default” on the Project>Project Properties to keep this setting).
Alternatively, sign the JAR but provide valid signing information.
- Non-compliance may lead to discs with movies not showing. Are all video files BDA standards compliant? (see the section “Bluray disc set top player specification” earlier)? If not, correct this and rebuild the disc (e.g. Power Director V16 creates MPEG-TS videos that are not BDAV)) MPEG4-AVC streams. Acceptable for tsMuxer but not the internal muxer – and won't play on some standalone players).
- If the movie seems jerky (jitter and stutter), there may be different causes:
 - Does the movie play smoothlessly in movie player on your pc? If not, it may be your pc or its video card is too slow for proper playback. In this case, burn the bluray disc on a rewritable disc and see if the stutter also occurs on a set top player.
Also check if you do not run multiple applications at the same time – this could put a burden on your processor causing the stutter by not having enough time dedicated to the player.
 - Check with a tool like MediaInfo whether the frame rate is the expected fixed rate (such as 25 fps) or variable. It should be fixed and supported (30 fps or 60 fps are not supported)
 - You may try a different video editor or converter to see if that makes a difference.
 - Does the video have a stable time base? Often these problems can be solved by using TS Doctor (Tools > Remux) or tsMuxerGUI to recreate the .m2ts source file without making any changes to the source. It will provide a new time

base. As last resort (at cost of picture quality) you may be successful in transcoding the original file into a different format (.m2ts into .mp4 for example) and back. The intermediate format should be overdimensioned (e.g. specified with higher bitrates) to limit the loss in picture quality. Transcoding might be done using the (partly freeware) HD Video Conversion Factory tool.

- Transcode the resulting file to bluray standards and finally demux the file into video/audio components for BDS and rebuild the disc.
- Use a different bitrate (higher seems to work better) in a video editor to create the movie.
- It sometimes helps to move the starting point of the new movie to a slightly later point (I-frame) in time using a video editor.

Project 2: Creating a simple menu-driven bluray disc

The Project Goal

This chapter will show how to make a simple bluray disc with a single static menu showing the titles of the movies contained on the disc.

A proper bluray disc consists of two major components:

- The movies to play (as we saw in the previous project)
- The menu structure with text, images, buttons and the navigation amongst them. That's what we're going to add in this project.


The addition of menus, buttons, navigation: this is where BDS does its work. Everything but the movies themselves and therefore all of the visual experience for a disc viewer, are programmed through BDS in a set of Java programming language routines, collected in a "Java Archive" or JAR file.

Java or mux

BDS consists of two relatively independent parts:

1. the visual experience (menus etc): a set of Java routines
2. the playback of the movies: put together by the muxer

Because they are independent, any change in the visual experience such as a new menu or different navigation or new buttons requires only to re-create the Java code to do this. BDS has a "JAR" button to do this.

Initially the entire disc must be created: the Java code and the muxed movies. But once made, you often only need to recompile the Java code and replace the JAR file in the final image (use the  menu button). That saves a lot of time when you're finalizing your disc with the final corrections. As long as you do not change anything to the movies themselves (such as new tracks, different chapters, other actions), just recompiling the JAR file suffices. Changing the name of a movie placeholder also means a JAR replacement is enough.

The visual experience: the Java part

The viewer uses his remote-control arrow keys to navigate between the various menu elements on screen such as selecting movie titles and selecting one to play.

This project extends the previous one by:

- adding a full screen menu shown at startup
- use buttons with a visible "Selected" state and different looking "Normal" state
- add navigation between buttons and action when a button is activated
- use a single button as prototype of all buttons

If you want to create this project yourself, the source files can be found in the .zip file mentioned in Appendix A: The user's guide source files.

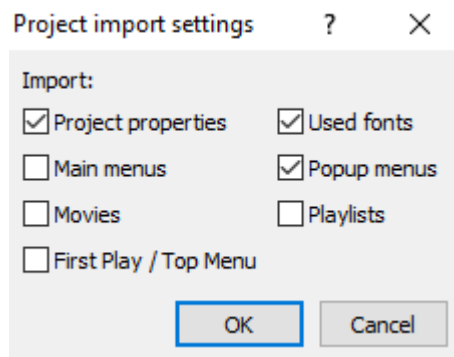
For those that want a demonstration video on creating menus, the BDS website provides one as YouTube movie:

<https://youtu.be/CPjBdnSfCXc>

Reuse: merge, template or copy project

This is as good a place as any: BDS has a File > Merge projects option. If in future you have created another project and want to reuse parts of it in a new project, you can use "Merge project". Of course, "Import from template" is another re-use option. This is preferred if you intend to re-use some elements frequently. And copying the project file and giving it a unique name allows you to use the earlier project as starting point.

If you click on "Merge Project", you are asked what project to merge with. In a popup window you then specify what elements you want to merge into your project.




This is not on a by-object-item basis but rather on a "by-object-type" such as all menus or movies. This may be too much – in which case you need to do some pruning after the merge. Alternatively, you may open the to-be-merged project first, prune it, save it under a different project name. Reopen your current project and merge with the pruned project. Do keep in mind that the project may not mux very well into a disc format. The merged elements may refer to unmerged objects and obviously some changes must be made to those references to work for your project. Merging is not ideal, but can be a handy copy/paste mechanism.

The Menu

A menu consists of various graphical parts that are layered on top of each other. An empty layer is totally transparent. Put something on a layer and it will hide anything on the same position below that layer. The layers work the same way as a stack of transparent sheets that cover whatever is pictured on lower layers.

Menus and all its elements (buttons, images) are BDS generated Java code. They are stored in a Java archive (JAR) file. Any change will

require BDS to recompile the JAR file. Use the JAR button () to do this.

Menu object layers

All menu objects reside on their own layer. They are stacked from top to bottom. The order is shown in the “Objects Window” if you selected a menu.

You can stack the objects in any order you want. It is common sense to adhere to the following order, from lowest to highest layer:

- menu graphics and text (“pictures”): the “static” elements at the bottom of the stack
- menu buttons: the “interactive” elements occupying the higher stack levels

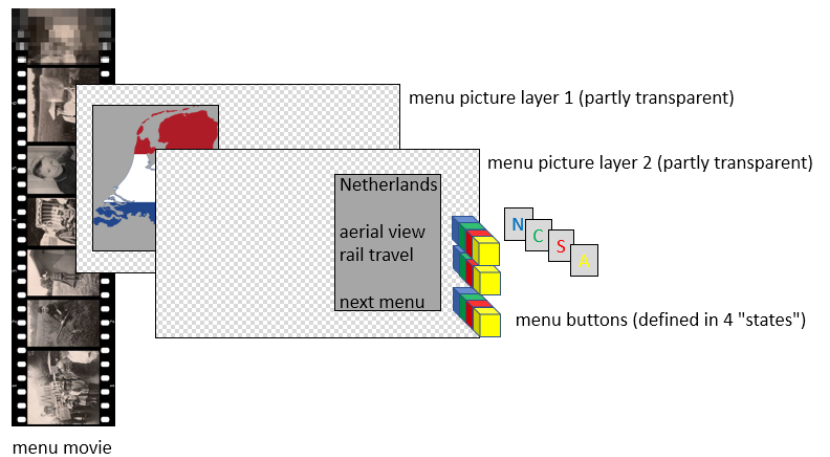
Menu components

A menu, once opened, remains visible until the viewer decides otherwise. He does so by activating a button on the menu that either replaces the menu by another or starts playing a movie.

The illustration below shows some layers in a menu – all stacked on top of each other. Of course other menus have more or fewer pictures and buttons.

From lowest to highest layer you see:

- **A picture layer** with a map of the Netherlands. The checkered parts of the layer are transparent. The solid grey parts around the map too, but any picture in BDS menus is a rectangular area and all pixels of that area contribute to the total picture pixel count – transparent or not. This count is therefore higher than the visible pixels that make up the country map.
The map partly hides the background movie.
- **A text layer** which is simply another picture layer that is fully transparent except for the text. Some of this text may be a menu list: buttons along side it allow such a list item to be selected and activated. The checkered and grey parts are transparent, but the all pixels in the grey part contribute to the total pixel count.



- Menu buttons** – Each button is on its own layer (that is completely transparent apart from the button image). Each button has four states (and it always in one state) that are sub-layers ordered from lowest to highest: N(ormal), C(urrent), S(elected) and A(ctivated). Only the “Selected” state is currently important. If you have separate images defined for each state, they stack on top of each other in the N,C,S,A order. Commonly they are positioned at the same location so one state “covers” all lower states. But you can position each state at any location.
 Here 3 buttons are shown, each with their four possible states where images are on top of each other. For our purposes at the moment, all states are transparent. The button only shows when it is selected (e.g. by showing its checkmark image). This way you have a visual indication of what button is currently selected.

The way a viewer would see this menu is illustrated below. Here we specified a blue square for each button’s “N”ormal state and a red square for each button’s “S”elected state. Only a single button can be the selected one – here the middle one (next to menu item “rail travel”). Therefore it shows red, covering its N state which is also blue. If the OK button is pressed on the remote-control a movie showing a rail road trip of the Netherlands is started. The other buttons are shown in their normal state, coloured blue. With the remote control arrow keys you can move up or down to select another button. The red square seems to move along.

In most cases, the button N state will have no image and therefore its state is transparent.

- The menu movie** is a black/white movie that plays in the background of the menu.



Component: Menu movie

Each menu has a menu movie. that plays continuously behind all other menu components. It must be a BDA compliant movie but its resolution determines the part of the menu shown on screen. A High Definition menu movie (1920x1080 pixels) will show the entire menu of 1920x1080 pixels. A half-HD movie of 1280x720 pixels only shows the left top most part of a menu: the area of 1280x720 pixels. Keep this in mind when you provide a menu movie and are surprised some buttons don't show. Then the menu movie has a smaller area than for which you designed the menu. Either replace the menu movie or shrink the menu so it does fit.⁸

The movie itself is just like any other movie: it is not part of the menu and thus not stored in the menu JAR file. You don't play it on command: as soon as a menu opens, its menu movie plays.

If you use the same menu movie(same video/audio streams) for all menus, switching between these menus does not interrupt the playing of the menu movie. Its video and sound continue seamlessly between menus. If only one menu is assigned a menu movie, BDS asks you if you want to use the same movie for all other menus. If at disc creation time some menus have no menu movie, they are assigned the menu movie of their parent.

Because there is a Java imposed maximum pixel count for all picture elements on a menu, you can reduce this count by adding as many picture elements to the menu movie. Then they have zero contribution to the pixel count. This is particularly useful if the same objects occur on many menus. Rather than being a menu part, insert it into the menu movie at the right position. Reuse with benefits.

Note: When a menu button starts a movie that is encoded in a different frame rate than the menu movie, there is a noticeable delay between activating the button and the start of the movie because the set top player must adjust the play setting. It may also cause video and

⁸ This is also true for popup-menus (discussed later). The menu must fit within the playing movie's resolution.

audio to become out-of-sync. If possible, try to keep menu movie and other movies to use the same frame rate.

Note: A menu movie on a commercial bluray disc cannot be paused. There was a request from/need for some users of BDS to be able to pause the menu movie. You can set your preference in the Project > Project Settings > Menu tab. By default you cannot pause the menu movie (this is the “usual” behaviour of commercial discs).

☐ Allow pause/play for menu

Component: Menu image

Images such as photos and illustrations can be objects on a menu. They are stored as part of the Java generated code by BDS and stored in a Java Archive JAR file.

Maximum size

There is a Java imposed maximum size for all images you include in the menus. For BD-J Version 1 it is 590 000 pixels.

Using V1 may give an error message when you include too many images.

ERROR: image buffer overflow.

All images takes 11756160 px, but buffer limit is 5900000 px.

You can switch the profile to version 2 in the "Project properties" or enable "Split into separate files" or "Load dynamically" for "Combine images" (read help FAQ → "How to reduce the amount of graphics" for details).

OK

Changing your setting to BD-J V2 and also setting the "Disc profile version" to "same as BD-J profile" alleviates the maximum size problem to a larger value (7 900 000) which may just let your disc be muxed⁹.

BD-J profile version 2 Disc profile version same as BD-J profile Combine images Split into separate files

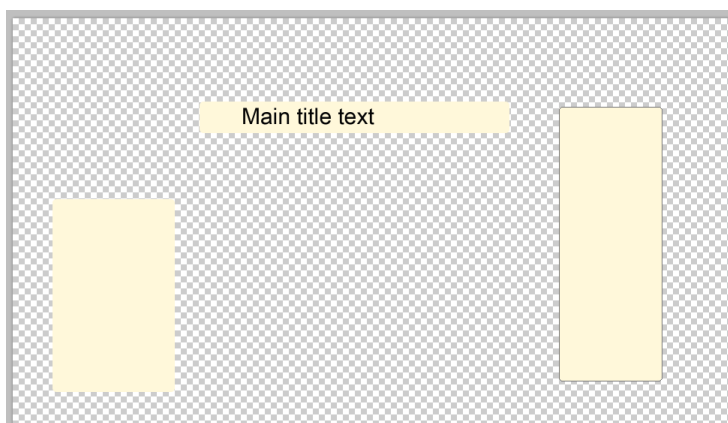
In addition you may set "Combine images" to "split into separate files". This will produce separate image files used in the disc image in the project folder __JAVA/00000 .

If you still require more pixels for all images than allowed, you need to divide the images over several JAR files (see Using multiple JAR titles on page 210). This won't be necessary for the current project and for many of your own projects.

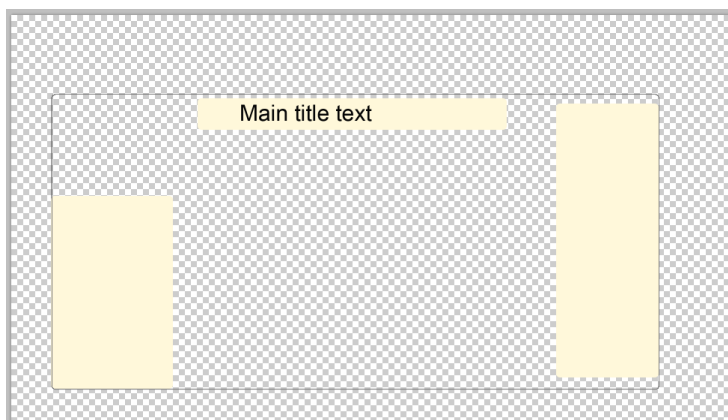
⁹ Some older set top players have problems with BD-J V2, it is then advised to use BD-J V2 but keep "Disc profile version" set to "force version 1"

A menu picture is a rectangular part of the menu. This rectangle may be partly transparent, but the entire rectangle is contributing to the menu pixels.

The illustration below shows 3 small visible picture parts as individual objects (each one in its own layer). Each part contributes to the pixel count.



When the three objects are in one image with a lot of transparent pixels in between, the surrounding rectangle contributes to the pixel count and many pixels are “transparent” but still add to the count.



Stack order of objects (layers)

Picture objects can be made as .png files (as .png supports transparency). Each picture file becomes an object on the menu and each object has its own layer (seen in the Objects window when you work in the associated Designer Window). You can change the stack order of the layers of the objects by shifting their order in the Objects window using the up/down arrows in the BDS menu bar.



You can show these buttons in both the Objects window and in the main menu of BDS or in either one. This is set through Tools > Options > Interface tab and select the desired behaviour in the “Show object move buttons” dropdown list. The illustration below sets the buttons on both locations.

Component: Menu buttons and button states

The second part of a menu consists of button objects. Using the arrow keys on the remote control, the user moves from one button to the next. What is “next” is determined by the button action for the arrow key. It is customary to set the navigation between buttons from the top button to bottom button but if you don’t you may surprise the viewer who presses the down arrow key and sees that a button at the top of the menu is selected simply because your navigational actions insist it to work that way.

When you then press the “OK” button on the remote-control, the state of the button changes from “Selected” to “Active” and the button actions defined in “Press ENTER” property are executed. This can be the start of a movie, a jump to another menu, or any other action you defined.

When navigating between buttons you often highlight the selected button by giving it a special image for its selected state. You may also make all other states of the buttons invisible (i.e. no image associated with the state) or with normal state dimmed.¹⁰

Moving from button to button, the button you leave changes its state from “Selected” to “Normal” and the next one changes from “Normal” to “Selected”.

As already mentioned, each button has four states:

- normal
- selected
- activated
- current

Each state can have an image. These images are stacked bottom-to-top as indicated. The “Normal” image is hidden behind the “Selected” image if a button has the “Selected” state.

Only the “Selected” state must have a visible button image. The other three states are “nice to have” but not visually essential. That means you don’t have to assign an image to all states: the ones not relevant to you fully transparent, i.e. have no image.

For this project we will only provide a button image to its “selected” state for the moment. And a white circle to its “Normal” state – just to show it’s there (in real life you probably would not provide an image).


To see any of the button states besides “Normal” you need to make those states visible ” in the Designer window by clicking on the related “Show <state>” button on the BDS menu bar:

¹⁰ A button can also be text. Then the selected text button shows the text in a different colour or font to make it stand out from the “normal” state buttons. For example, all texts are blue but the selected one changes the text into a bordered yellow colour.



The order of the buttons is to show the normal (violet), selected (red), activated (yellow), current (green) states of the buttons.


State: Normal

The “Normal” state  shows the button when it is not selected. When a menu offers 10 choices, 9 of them will have a button showing its “Normal” state and only one in the “Selected” state. The “Normal” state is *always* visible (unless transparent or absent). Other button states are shown on top of the normal state image and may cover it completely.

Note: Any non-normal state image of a button therefore usually covers the entire normal state image. If it has transparent parts, the normal image shines through those parts.

The “Normal” state can be a dimmed version of the “Selected” state, but often it is totally invisible. This means that the “Normal” state has no image file associated with it.

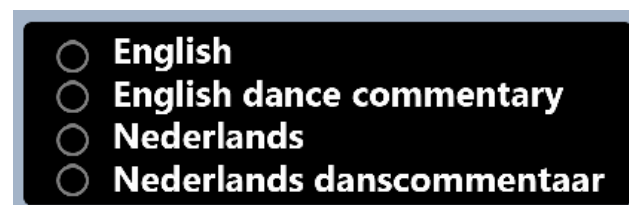
State: Selected


The “Selected” state  shows the image of a highlighted button. As such, it will always be a visible state of the button. It can be a bright arrow, a check mark, a pointing finger or some other indication of what menu item is currently selected through arrow navigation on the remote-control.

The “Selected” image differs from that of the “Normal” state. When the button is in its “Selected” state, it often totally covers the “Normal” image.

Normal and Selected states of buttons during menu design

In the figure below a small menu is shown as static black image with white text. At the left of the text the “Normal” states of four buttons are shown. Here we use a grey outlined circle as “Normal” state image although often it would be absent entirely. But this way we can see the buttons in their “Normal” state.



The same buttons are shown again, but now their “Selected” state (when the BDS menu button  is clicked).





Notice that all four buttons have a “Selected” state looking like a yellow circle with an arrow. Each button has such an image for its “Selected” state and therefore in the Designer Window they are shown, covering the “Normal” state images that are on a lower layer.

When the disc is played, only one button is in its “Selected” state, all others are “Normal” – as the image below shows.



States: Activated or Current

The other two states, “Activated”  and “Current”  are less important at the moment, but have their uses in elaborate menus. “Current” we will discuss in the next project.

A button goes from “Selected” into its “Activated” state when the “OK” button is pressed on the remote-control.

If you implement an image (.png file) for the “Activated” state it will briefly show this image before executing the action associated with its “Press ENTER” property (e.g. play a movie or jump to another menu)¹¹.

Menu items copied

The same way you copy items in other applications, you can copy menu objects. Or cut them from one to paste them elsewhere.



- select the item in Designer Window or Objects window
- Click “Copy” (or Cut) or press CTRL/C (or CTRL/X)
- Click “Paste” (or CTRL/V)
- Indicate on the opening window what menu you want to paste to.

If the paste is done to the same menu you may wonder where the copy is. BDS positioned it on top of the original item. Hence click that item and drag it away to its final position.

Positioning objects

Use the Designer window to reposition each object relative to the others. Use the object window to define their relative order (where items on higher layers can cover items on lower layers)

¹¹ Also an animation may precede the action of the button (animations are covered in Project 8: Animations on page 231).

Ensure the menu objects are unlocked for this purpose (button  set to ). Once you are satisfied with the arrangement you can lock the objects in place to avoid accidentally moving them.

Instead of creating a .png file for each menu object, you can create a single multi-layered Photoshop file where each layer will become a menu object when BDS imports it. This is the only way to create objects and layers in BDS Lite. The preferred way now is to use separate .png files. We'll show how both methods work later in this chapter.


You can check on how many pixels are used as all menu images are stored in the final bluray disc output folder as a Java .jar container in folder \BDMV\JAR\00000.jar. If you rename this file to 00000.zip and open it, the zip container will show a m1.png file which contains all pixels of all menus (if you have chosen all menu picture objects to reside in a single image). You can extract it or open it with an image editor to see whether its total size exceeds the current limit of 7 900 000 pixels.

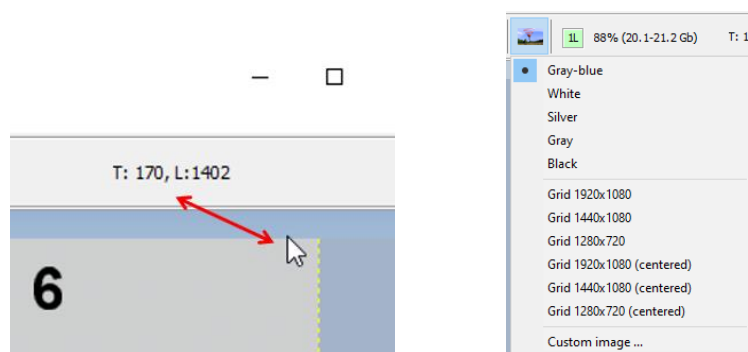
Online help is provided under the help heading “How to reduce the amount of graphics”.

Note that when you use a carousel option for menu buttons (see section Carousel on page 174) this adds to the total pixel count too.

The Designer Window allows you to rescale images. Keep in mind that scaling only changes the visual representation of the image. It does not change the size of the image in pixels. That is still the 100% size. Rather than scaling an object, you may create a smaller sized object (.png file) using any image editor.

Note: *rescaling significantly (reducing to 50% or less) may impact playback on certain players. Best to keep images for buttons and menu approximately equal to the size required.*

The Designer Window allows menus of 1920x1080 pixels in size. The cursor position tells you where you are on the menu. You can also select one of the menu sizes by pressing on the “Designer background” menu button ().




You can setup a background for the Designer window. This background will not copy into the final movie, but it will help to contrast menu objects with colours that may otherwise be hard to see.

If a background movie is used with an area in which you want to position the menu texts and buttons, it may help to make a screenshot

of that movie and use it as background in the Designer's Window. This allows you to position the menu items correctly.

As an example: a menu movie is a rattling projector. Apart from the reels spinning, the image is more or less static. Using a screenshot as Designer Window background allows us to position the text and buttons within the projection screen area.



By default a grey-bluish background is provided. Change it to anything you like by clicking the “Designer Background” button () on the menu bar.

Menu creation: 3 ways

To create a menu, you can use one of three methods:

- use a layered Photoshop file and create a separate layer for each menu object. Each layer will become a menu object layer.
- use the Designer Window and import .png files for text images, illustration and button state images
- use the Design Window and associated Objects window and create new objects for a menu using wizard-type windows.

The first two methods are the only one available in the BDS Lite edition. The commercial editions of BDS also allow the third option which is by far the most productive one. The first two ways allow for a more richer menu showing all the image manipulation tricks image editors provide. For a simple menu, the third way is the quickest.

Once a menu is made, any modification can be made using either of the three methods.

Menus through Photoshop

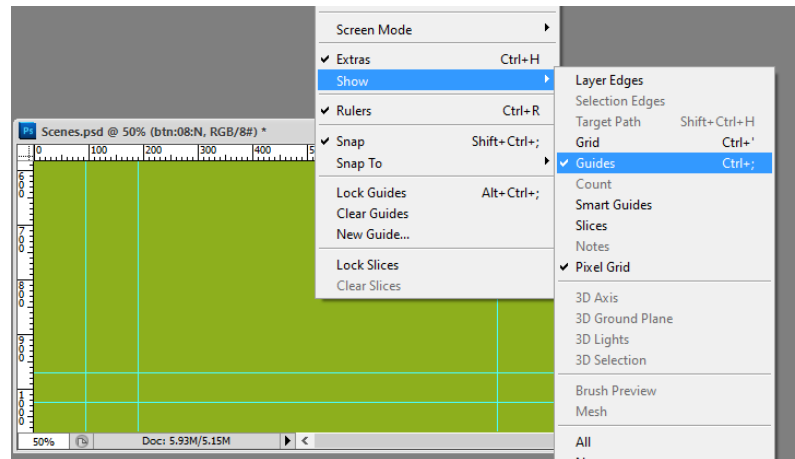
The method of creating menus in Photoshop has been around since V1 of BDS and is available in the free Lite edition that is based on V1.

The Photoshop (or similar layer-supporting) image editor creates graphics files that consist of multiple layers. Each layer is filled and moved independently of the other layers. The final Photoshop .psd file must use 8-bits/channel for colour.

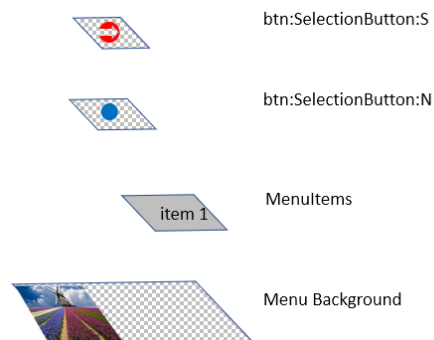
Each imported .psd file becomes a menu. Each layer in the .psd file will become a menu object. Either a static menu picture or a picture representing one state of a button.

The layers need not be completely filled (for HD you can limit the Photoshop canvas size to 1920x1080 pixels). Buttons usually are very small in size and use little space on a layer.

Tip: to have precise alignment or positioning of picture and button elements, Photoshop has design-aid only “guides” (Check View > Extras and use View > Show Guides. Create these lines using View > New Guide).



Layers that represent a single state of a button must have a layer name that adheres to a naming convention. Only that way BDS can distinguish between static picture layers and button state layers. Any layer name that does not adhere to the standard naming convention automatically becomes a plain, static picture.



The layer naming convention for button states is “*btn:name:state*” where the *name* is some unique name for the button and *state* is one of the four states a button can be in. It is one character in size and is either “N” (Normal), “S” (Selected), “C” (Current) or “A” (Activated).

The illustration above shows a menu of four layers.

- Two are images for two states of a button “SelectionButton” (states “normal” (a blue circle) and “selected” (a red arrow, on top of the normal state blue circle))
- One static image has text to be shown on the menu
- One static image shows windmill in a tulip field.

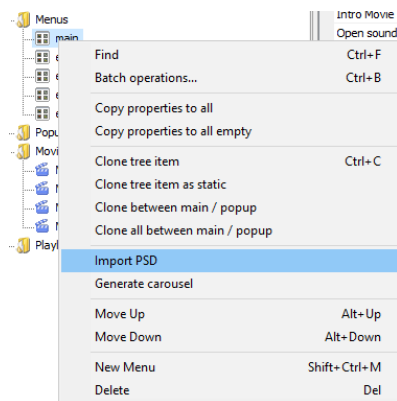
For on screen visibility, the button size should be 40 pixels high or more. Much smaller makes them difficult to see on screen.

Tip: Create one .psd file with all different types of buttons you are likely to need and import these into a template project. That way you define the buttons once and can reuse them in many projects if a new project is based on such a template.

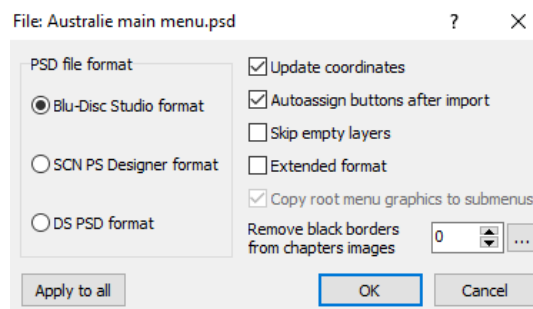
Some simple button shapes can be taken from the Wingdings font if you use a text layer for a button state.

Tip: Create only one button of each type for a menu. During the design of the menu in BDS you can duplicate any button as often as needed.

A Photoshop file is imported into your BDS project by going to the Project View, selecting the “Menu” item and open the dropdown menu by a right click of the mouse. Select the “Import PSD” option.





That opens a navigation window to specify the .psd file to import. Then BSD opens its import window to allow you to specify what sort of file you import.



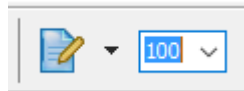
Use it with the preset options. The check with “Autoassign buttons after import” creates button states for all Photoshop layers that follow the naming convention for button layers. By clicking “OK” the file is imported and BDS creates a subfolder \menuname in the project folder, that contains all the .png files BDS created from the .psd layers.

If you discover during work that you have forgotten something: simply edit the .psd file and re-import it. BDS realizes it already has such a menu and asks if you want to update it. Say “yes”. New additions will automatically appear. Removed items don’t show, but their .png files still exist in the \menuname subfolder.

Do check that some picture layers do not unwantedly overlap each other, one covering the other. You can move them apart by unlocking



the objects in Designer window (set lock object button from  to ). You can also change the relative order of these layers from A-B into B-A if needed by changing the object order in the Objects window.

You may want to increase the Designer window to 100% (use the scaling box at the top) to ensure proper positioning:




The same works when the Designer Window has focus and you roll your mouse wheel.

By pressing the CTRL key while rolling the mouse wheel you can move up and down in the Designer window if the menu does not fit in the window. Pressing SHIFT while rolling the mouse wheel you can shift left and right in the Designer window.

For buttons it is important that their state images do overlap (only one will be visible at any moment, except “normal” that is always visible – if not absent or transparent). If state images do not overlap each other, reposition them so they do. This may require temporarily undoing the “move as one object” setting (from  (blue background) to ).

Menus through plain .png files as image or button


If you provide all menu elements as separate .png files, you simply import those as menu objects. Create a blank menu in the Project Tree and open its Designer Window. Then in its associated Objects window, click the “New Object button () and create a new object. Assign the .png file to the picture object.


Ordering of the menu objects or positioning is done the same way as when a Photoshop .psd file had been imported to create a menu. In fact, such import does nothing else than producing a set of .png files – one for each Photoshop file layer.

Menus made within BDS

Since BDS V3 you can also create menus entirely from within BDS and compose it in the Designer Window and its associated Objects window.

This method works by far the quickest and is the most flexible. But it only works in BDS Standard or MX. Not in the BDS Lite edition.

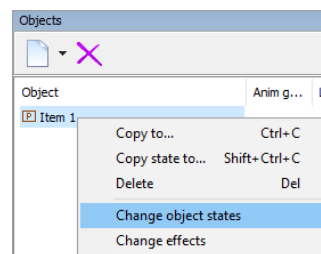
You must first create a blank menu by clicking on the menu button in the Project Tree window (). The same result is obtained by selecting the “menus” branch in the Project Tree and right mouse click to select “New menu” from the dropdown list. Or press SHIFT+CTRL+M at the same time. Once the menu is made it is shown (blank) in the Designer window. Its associated Objects window is also shown for the properties of all menu objects. Since there are no objects yet in the blank menu, this Object window is empty.

To add an object, you need to either click on the “New Object” button in the Objects View window () or click on the little arrow next to it to show the dropdown menu. When the object is created after an “OK” button is clicked, it comes out as “Item 1”. Change this name (in the properties window of the Object window) into something more descriptive.

Once made, you can edit the object (change text, colour, transparency) by one of two methods:

- Double click on the object in the Designer window
- Right-click on the object in the Object Window and from the dropdown menu select “Change object state” or “Change effects”

The “Change effects” is only available through the second method. It allows to scale the object by percentages. The same percentage for width and height scales the object proportionally. You can also manually scale it by dragging the small square at the right bottom corner. There is no proportional scaling when you rescale an object in the Designer window (other programs use the SHIFT key for this, BDS does not) so you may need to check the scaling percentages afterwards if it is important.



Add text object fonts

When you create text objects, it uses text fonts that need to be local to BDS. Before creating text objects, you must first create a folder containing the TTF (TrueType Fonts) files of the fonts you are going to use. To do this:

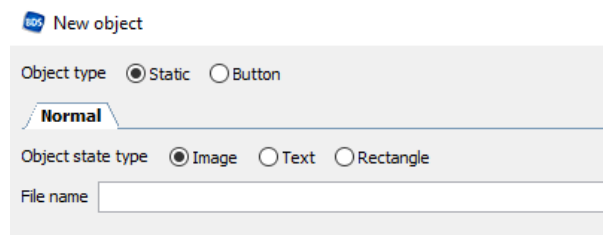
1. Create a new folder (this can be outside the BDS projects folder but I prefer it to become a subfolder \fonts – this makes the project self-supporting or all-inclusive of the used resources).
2. Copy TTF font files from either a CD or DVD disc or existing hard drive folder into this folder. Windows itself stores all its font files in a hidden folder %windir%\Fonts folder on the system drive. Because of its hidden nature, you cannot select this folder in BDS. Any font there you fancy must be copied to another visible folder like \fonts.
3. Make BDS aware of the fonts you want to use through menu option Project > Used fonts. Initially none will show. Click the “Add” button to navigate to the \fonts created folder from step 1. All font files you copied into it, will be shown even if it looks for .OTF and .TTF files.

4. Select the fonts your project uses and again click “Open”. The “Used Fonts” window now shows those fonts only. You can add more from different locations or from the same location if you did not select all font files initially. In the same manner you can select some font files and remove them.
5. Finally click “Close” to make the fonts known to the project.

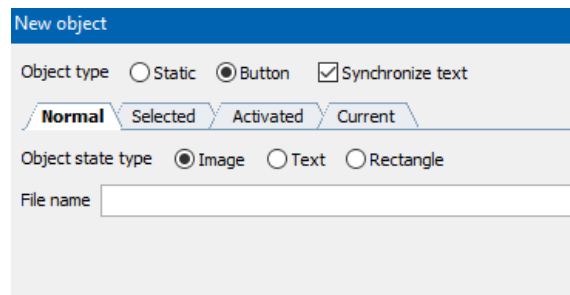
Note: when BDS opens a project with used fonts, it tries to include them in the project. If you have deleted some of the fonts or the entire map or renamed it, BDS fails to find it. It then inquires you for the new location. You can then select "ignore" or "no" to continue loading the project without the fonts. Any text in a menu using the unfound fonts cannot be changed unless you then specify where to find the font.

Open the Designer window to start building the menu. This also opens the associated Objects window.

If you click on the New Object button in the Objects window, a window opens to ask what object you want to add.



Initially it assumes a new static object with a single image or text is going to be added (all only in a “normal” state). But if you select the “Button” choice, suddenly all four button states are shown and you can select a different image for each state.



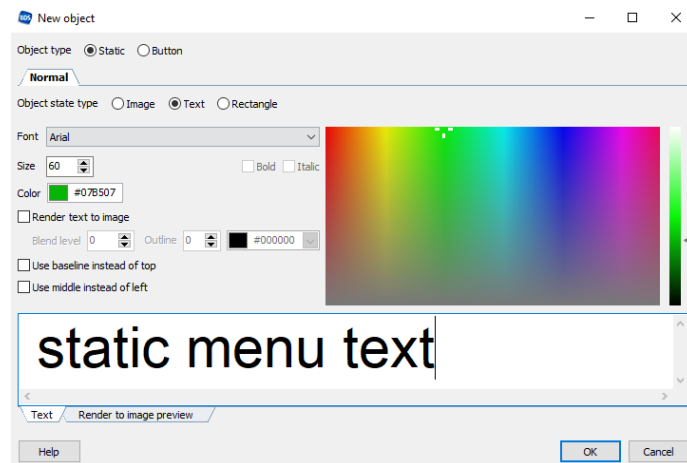
A checkbox for “synchronized text” is added meaning that if you add text to the button (rather than an image) the same text is copied to all states. In which case of course you need to use different colours for each state.

Static menu objects (text, image or rectangle)

A static object is always a pictorial object. It can be:

- an image (a .png file which you specify in the “File Name” box). Such an image can consist of text, but text is easier created using the “Text” option described in the following bullet.
- a text object (creating a block of menu text that may contain menu selection text). Select the font you use for the text.

The “Font” dropdown box contains all fonts you pre-selected earlier via Projects>Used fonts. If you did not add any fonts, you’re told to do so first. The project must also sign the JAR file (Project Properties>General>check “Sign the Jar”) You can move the pointer in the colour block to determine what colour the text will have.

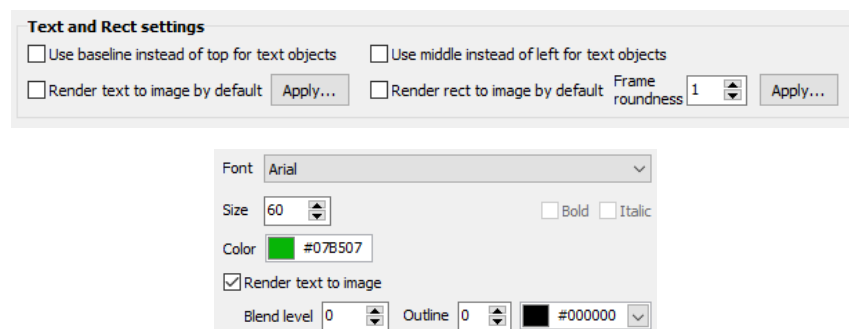


- a rectangle (to easily add a rectangle of a specific colour to use as a background for text) Its dimensions must be specified but the Designer window can be used to check its size or you can scale it yourself.

Tip: if you create multiple text objects which all must have the same colour, then write down (or copy to the clipboard or write it in a Notepad file) the value of the selected colour (in the example: 07B507). Then copy/paste it in each new object window.

Please note that for a final product you may need to convert the static text objects to an image. If you do not, the buttons placed next to the menu text may not align in the final disc image (they will in simulation mode) as rendering of both is done differently. Use the “Render text to image” checkbox for this. This still allows you to modify the text – the rendering only happens during disc building although BDS attempts to show it correctly in Designer window.

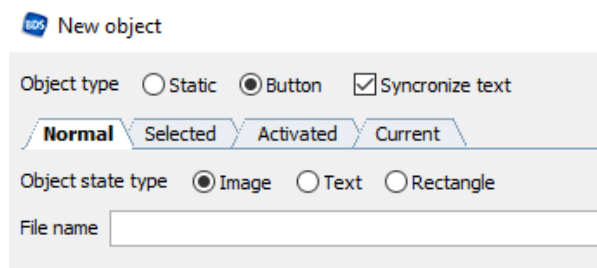
If you want the rendering always to happen, check in the Project > Project Settings > Menu tab the item “Render text to image by default”.



Once done, the “Change object states” dropdown list option for the text object is disabled as there is nothing to change to a rendered image.

Button objects

A button always requires a different representation of each of its four states. Only “Selected” is the one state you always need to specify. “Normal” can have an image but can also be transparent or absent and the latter may not require a .png file. The state image .png file is specified in the “File name” box. For visibility, the button size should be 40 pixels high or more.



When a button image file cannot be found, BDS will ask you to specify its location. If you select to ignore the error, the project will load but the button will not show. If you try to use it, BDS will again ask for the location of the image – at this moment you may decide to specify another image.


Creating textual buttons

Rather than using arrow-type button images to indicate what a viewer selected, you can use the text itself to indicate selection. The selected text is then highlighted in a different colour.

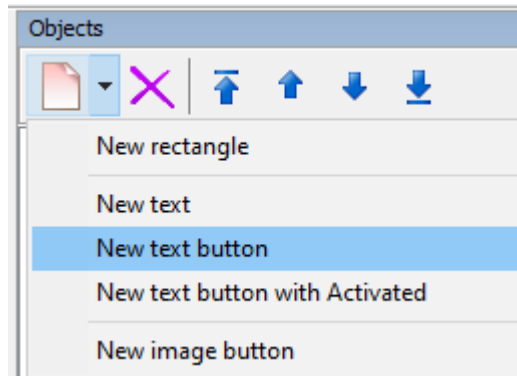
The menu texts are all in “normal” state (in terms of font, colour, size, embellishments) and the one selected has a different setting for its “selected” state.

The easy way

To use a textual button, its font must be available to BDS (see “Add text object fonts” on page 91). Use the Arial font in the kit from \Sources\BDS Fonts\arial.ttf

In the Objects window you press on the arrow next to the “new object” button ()¹². That opens a context window from which you select “New text button”.

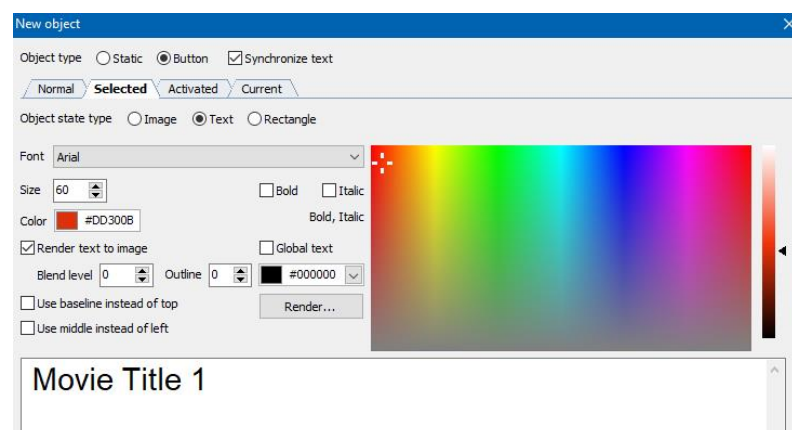
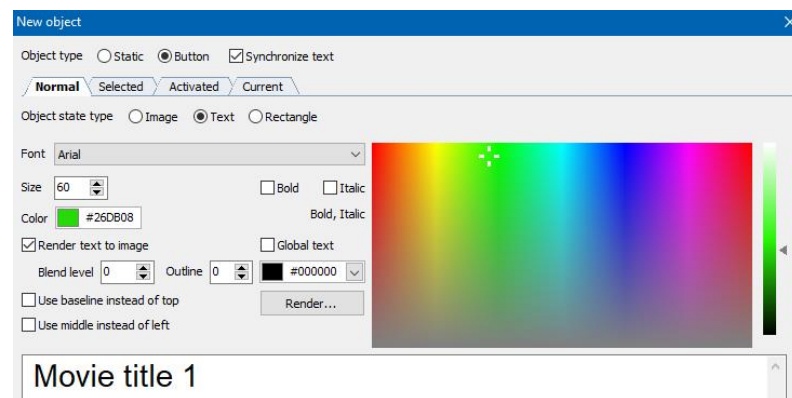
¹² You should also be able to press the button itself and select the radio buttons “button” and “text”. This however only sets the button Normal state. You would need to redo this for all other states (select “button” and “text” again).



This opens a window in which you can write the text of the button.

You can ensure the same text is used for all states if the “Synchronize text” checkbox is checked. This way the button and its text are the same in all states. All you have to do is change its colour to distinguish between states.

The figures below show a text button in its green “Normal” and red coloured “Selected” state.



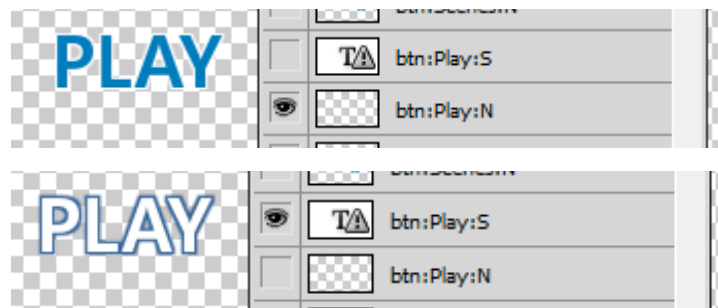


By default, BDS only shows the normal state of a button on the Object window and Designer window. To show the other states too, you need to enable their visibility through the visibility menu buttons illustrated below. The “selected” state is shown by clicking on the red button.



The special way


Text buttons with different colours in different states are also easy to make in an image editor that allows layers – such as Adobe’s Photoshop, Corel’s Paintshop or the free Gimp applications. The example files provided by Blu Disc Studio use a Photoshop file with two layers showing the “normal” state (btn:Play:N) and “selected” state (btn:Play:S). The selected state uses a white font with blue outline while the normal state has it reversed with blue font and white outline.




You can also export those layers individually as .png files (transparency kept) and import them as different images for states of a new button object.

Steps to take for best results

Importing button layers from a Photoshop file is not always the way to go. At import, any embellishment added to the layer gets lost. If you want to keep these, you need to export each layer and its embellishments as a separate .png image file. When you create a button object you specify each .png file for one of its state properties (hide all other Photoshop layers) and position all images so that they overlap:

1. Create a new button object. Select for its state’s image one of the .png files that contains the embellished text.
2. Allow the button state images to be moved independently (BDS menu button  - not blue background) in Designer window and position the various states on top of each other. Use the property’s Left and Top indications for perfect alignment. Once properly layered on top of each other, lock


the button states to move as one. (See section Menus through Photoshop on page 87 for details).

3. Position the menu items and buttons at their proper location on the menu. You may need to unlock objects through BDS menu button . (See section Menus through Photoshop on page 87 for details). Once in the right position, you may want to lock them again.

You now have a menu with textual buttons where the state of the button is reflected by the different rendition of the text. A number of possible renditions are given in the sections below.

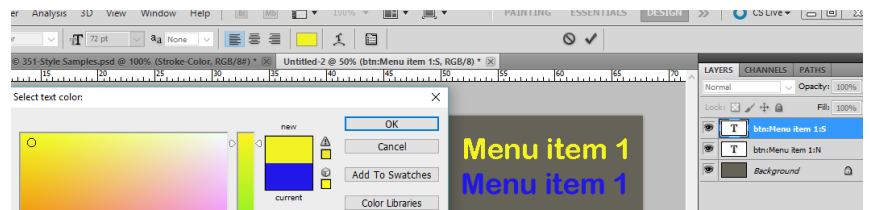
Different colour

The simplest case is using the same text and only change the colour. For example, using Photoshop, for the “normal” state we use the colour blue, for the “selected” state the colour yellow.

1. In Photoshop, use the Text tool  to create a text layer and write the text.



2. Duplicate the layer, select the menu text and select a different colour in the colour swatch box in the menu bar.





3. Rename both text layers to the BDS naming convention for buttons (btn:name:N or S)
4. Either here or once imported into BDS superimpose both states on top of each other so the viewer gets the impression it is a single text that can change its colour.

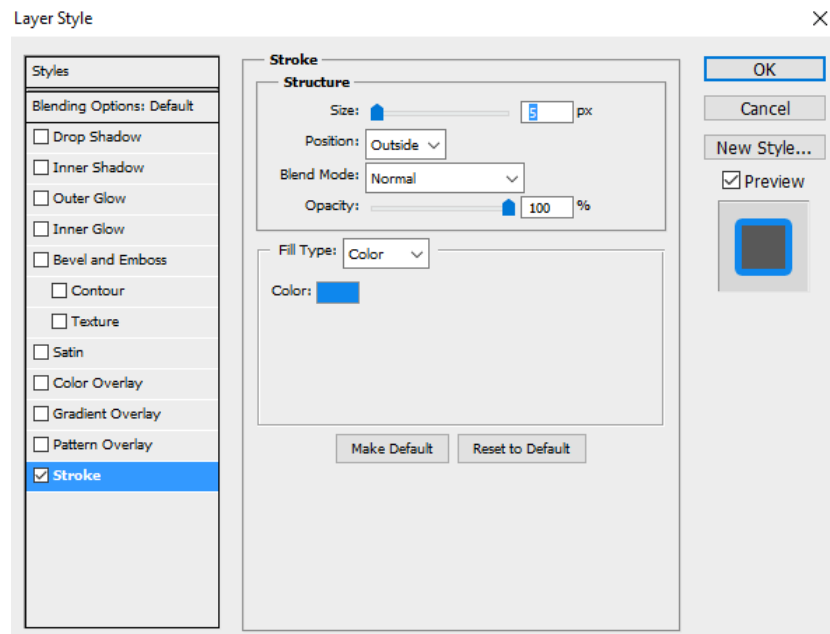
Swapped colour and outline

A more sophisticated different rendering (than only changing font colour) can also be achieved. As an example the same text is used but the “selected” state uses an inverted version of the same text. This resembles the example files provided by BDS most.



For both text layers you perform the same actions in Photoshop (but interchanging the font colour and outline colour).

1. Type the text and duplicate the layer
2. For each layer:
 - a. Select the font colour in the “text layer” menu colour swatch box () (this menu may show when you click on the text rectangle () in the Photoshop Layers window)
 - b. Double click on the layer bar in the Photoshop Layers window (the blue part, not the name field of the layer). This opens the Layer Style window. In this you check the “Stroke” style by clicking on the text “Stroke”. This opens a new right-hand part of the window specific to “stroke” options.

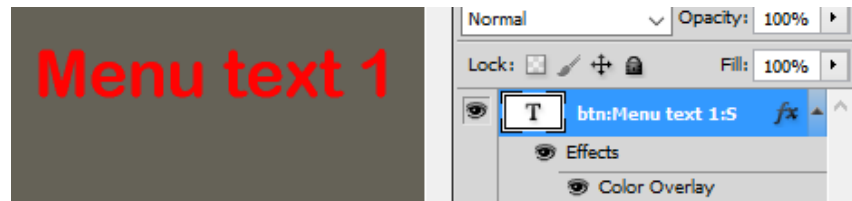


- c. To ensure you change the outline of the text, select “Outside” in the “Position” box
 - d. To change the colour of the outline click on the “Color” swatch box and specify the right colour (note the RGB numbers to reproduce the exact same colour in the other text layer)
 - e. To set the thickness of the outline, set the “Size” slider to the right thickness
 - f. If the outline is not fully opaque, set its slider to more transparent values
 - g. The result is shown in the preview at the right
 - h. The layer information in the Layers window for the text layer now expands (or click the f_x button) to show you the “stroke” effect. Double clicking on “Stroke” effect, will re-open the Layer Style window in case you want to change the settings.
3. Rename both text layers to the BDS naming convention for buttons (btn:name:N or S)

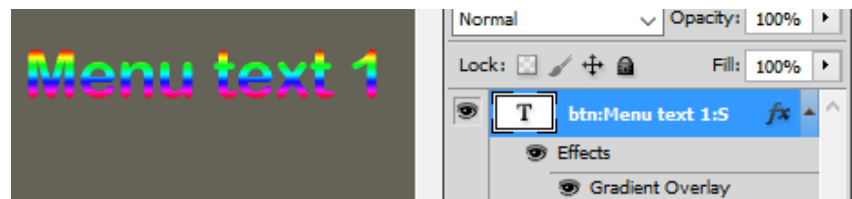
4. Once the .psd file is imported into BDS, superimpose both states on top of each other so the viewer gets the impression it is a single text that can change its colour.

Other layer style options

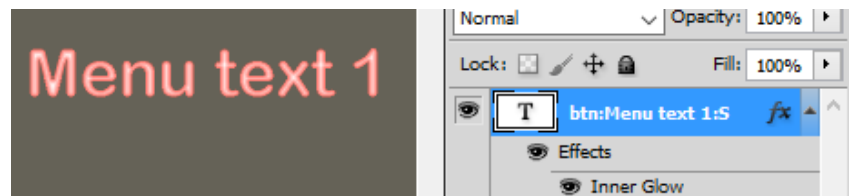
Browsing through layer style options in Photoshop, you will find many other ways to highlight text if selected. Several style options can be combined (such as colour and gradient). Most overlays require you export the results as .png files and specify those images as states of a button.



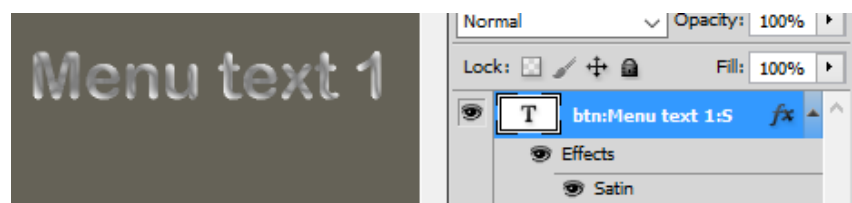
Color overlay



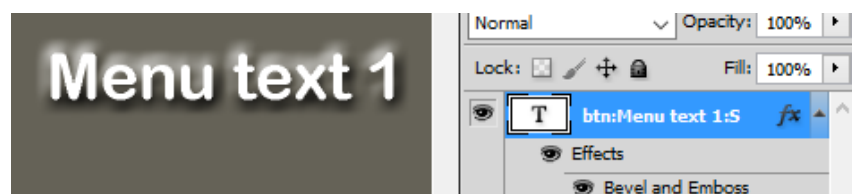
Gradient overlay (solid)



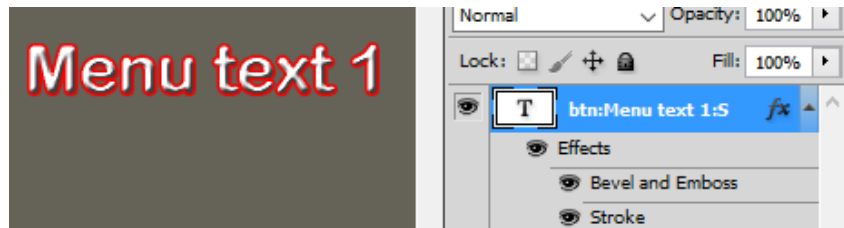
Inner Glow – edge



Satin



Bevel and Emboss: outer level



Bevel and Emboss + Stroke

Aligning menu objects and adding navigation

A menu looks neater if buttons are properly arranged and if they connect to each other in a consistent manner.


BDS provides one option to align objects and one option to automatically provide navigation.

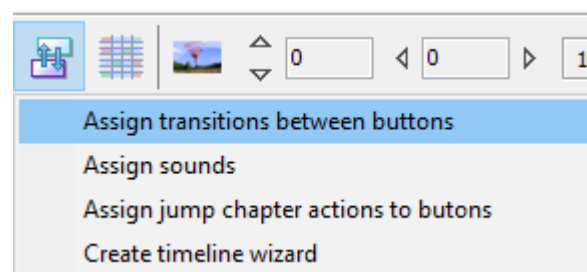
Align

1. To align objects, select all objects that need aligning.
2. Right click on the selection opens a context menu with many alignment options (to align all left sides (useful for vertical lists), to align all tops (useful for horizontal lists)).

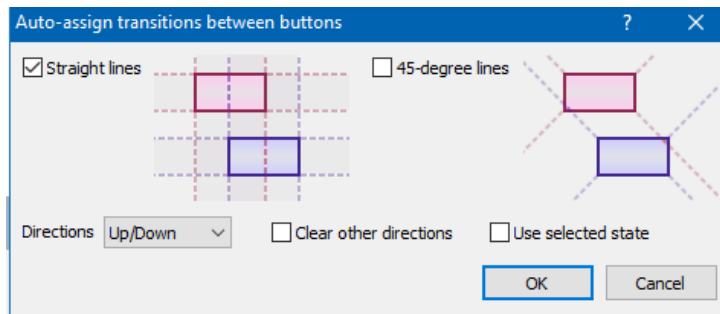
Auto-assign

Button selection can be changed by pressing the up/down/left/right arrow buttons on the remote control. You need to specify in the button properties what happens if such a button is pressed. There are the “Press Up” / “Press Down” / “Press Left” and “Press Right” properties that link the current button to the next one. If there are a lot of buttons and they are neatly aligned in a row or column, the auto assignment feature of BDS can provide most of the navigation.

This feature is activated by the Auto Assignment and Wizards button on the BDS menu bar (). Clicking on the button opens the context menu from which you select “Assign transition between buttons”.

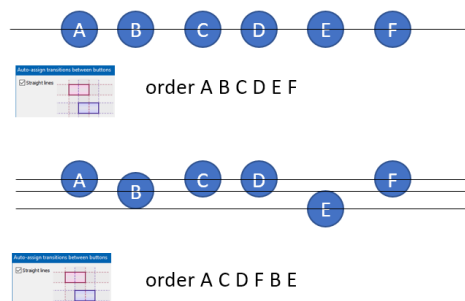


That choice will allow you to provide transitions from top to bottom of a list of buttons or from left to right, depending on your choice for Directions: Up/Down or Left/Right.



By checking the “Clear other directions” any previous navigation assignments are removed. But unchecking it allows you to first generate up/down assignments in the “Press Up”/”Press Down” properties of the button. Then next you change directions to Left/Right and assign again. This time the “Press Left”/”Press Right” properties of the button are filled.

Do interpret “straight lines” correctly: it means that assignment is done strictly in horizontal (or vertical) manner from left to right, top to bottom. Hence if one button is slightly lower than the rest of a row of buttons, that button is considered to be “next row”.



Other auto-assign option are less frequently used: audio assignment for a button when you navigate to another button or buttons with chapters to skip 1,2 or more chapters in a movie.

Note: If your buttons do not align, check if the button images are heavily scaled down. BDS cannot handle that. Ensure button images have almost no scaling when used on a menu.

D.I.Y. Adventure

If you want to go by yourself, keep in mind:

- Add two movies to the project, incl. video, audio and subtitle tracks
- Define a menu with text elements for the viewer to show all movies they can choose from
- Place buttons next to each menu text to actually select a movie to start. Return to the menu upon completion.

Step 0: Ready to run

If you do not want to create the project yourself, you can inspect and use the completed project by going through the following steps:

1. Copy the \Projects\MenuBR folder tree to your BDS projects folder (e.g. J:\BDS Projects\MenuBR)
2. Copy the \films folder of the previous project “SimpleBR” to your project (e.g. J:\BDS Projects\MenuBR\films)
(Alternatively: copy these movie files “Coasts” and “Australia” from \Sources\movies\demuxed and \Sources\movies\subtitles)
3. Copy the \Sources\movies\demuxed\menu.264 and menu.ac3 to the project’s \films folder.
4. Open the project by double clicking on its project.bdmd file (e.g. J:\BDS Projects\MenuBR\MenuBR.bdmd)
5. Click the “mux” button and rebuild the disc image (in the project’s \Output folder).

Step 1: Get organized

This step is mostly identical to the steps specified in section “Step 1: Create new project” of the previous chapter.

Now that we are introducing menus into the project, the project folder needs to be enhanced by adding a new folder “\original sources” for the menu sources such as Photoshop layered files or set of .png files. The MenuBR project looks like the figure below. The sometimes hidden folders _Java and _Scripts are added by BDS upon project creation. Additional ones (_History and \Log) are created whenever you perform a mux and they do not yet exist. Hidden folders are made visible through File Explorer > View window and checking “Show hidden folders” in the Options menu item.



Step 2: Include the movies

This step is mostly identical to the step described in section “Step 2: Include the movies” of the previous chapter. It may be postponed until after the menu has been developed.

The two movie placeholders for our project are “Coasts” and “Australia”.

Fill in the movie placeholder properties with the actual file specifications for video and audio streams and subtitles as you did before in SimpleBR. They can be found in the downloaded BDS Kit.zip file in \Sources\movies\demuxed and \Sources\movies\subtitles.

Since they are the same as the ones used in the previous project SimpleBR, you may also copy the contents of the SimpleBR\films folder to this new MenuBR project.

New for this project is the “menu movie”. Each menu has a menu movie that runs in the background as a sort of moving wallpaper.

As menu movie we will use a prepared file you can copy from the BDS Kit.zip file as \Sources\movies\demuxed\menu.264 and menu.ac3. Copy these two files into the project's \films folder. (e.g. J:\BDS Projects\MenuBR\films). Check that the menu movie has the same frame rate as the movies you start by selecting its title from the menu.

Step 3: Create the menus

This project starts a bluray disc by showing a main menu rather than starting a movie directly. You select which movie to play by selecting its menu item. Once the menu is added to the project it will look like the picture shown below.



The menu background movie shows rolling waves. It has two picture objects: the Australian flag and some text in red. It has two buttons, each with a normal state (a white circle – just to make the state visible – usually it is transparent or absent and hence not visible in the menu) and a selected state (a yellow arrow). A very simple, single menu.

Create the static background

The static pictures themselves must be created outside BDS. Either as a set of .png files or as a single layered Photoshop file.

There are two ways to compose the menu:

- Make a full screen 1080x1920 pixel menu with an image with text (and buttons will come on top)
- Build the menu from several small parts. Each part has only the text or flag image to show. The rest of the menu is filled by the background menu movie

If it is a simple disc with a single (or maximum 3) menus, the full screen menu image can be used. If you have more (than 3) menus, the second option of small parts must be used. The total pixel count of all menus must remain below 7 900 000 pixels.¹³

¹³ Unless you opt to use multiple JAR files – this is not needed in this project.

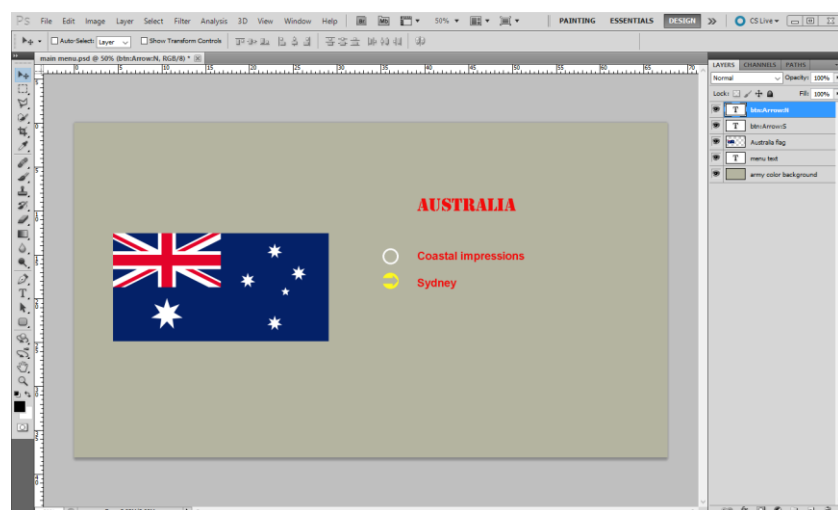
We will use the small parts as it is the way to go if many menus and buttons are going to be used.

Menu through Photoshop

Let's start with the "all in one" approach of creating a Photoshop file "main menu.psd" with each menu object on a separate layer and each button using as many layers as it has visible states.

The Australian flag, the text and the button layer for the selected state are all present as the figure below shows.

To provide some contrast to all objects in the layers while working in Photoshop, an extra layer in army green/brown is added as lowest layer. This layer will be removed before we save the .psd file or otherwise it gets imported into BDS and we delete that object.



The button layers follow the compulsory naming convention of btn:Arrow:N for the normal state (layer with the the grey circle) and btn:Arrow:S for the selected state (layer with the yellow circle with arrow). For visibility, the button size should be 40 pixels high or more.

This yellow circle is in fact a text character from the "Wingdings" font (glyph number 0xDC, in normal text fonts it is usually "Ü"): ☉ where the black circle is coloured yellow by taking a yellow font colour. The font size is increased to fit the text of the menu background.

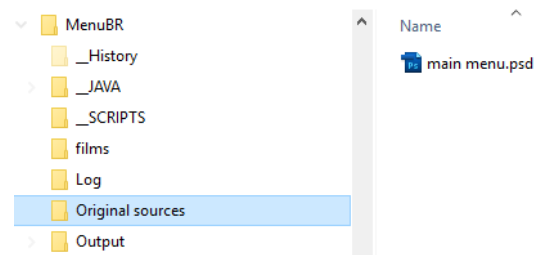
The white outline circle is another character from the Wingdings font (glyph 0xA1, usually "i" in normal text fonts): ○ where the font colour is chosen to be white. This layer is stored as btn:Button:N .

Of course for any project you can make the image as elaborate as you want: small part of a photo, a home-made icon, anything. But here we keep it simple: a simple Wingdings character.

Within BDS we can copy the button multiple times. Each copy will get its own name and in its properties it will be connected to the specific movie or menu it can activate.

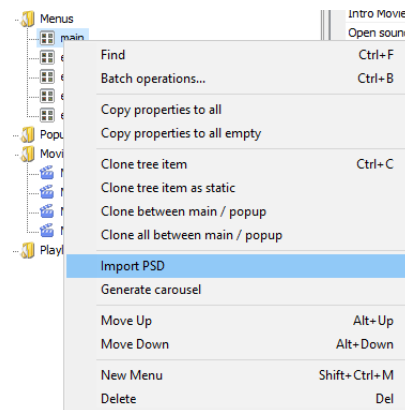
The final Photoshop file with menu image and button layers is stored in the project folder under the new subfolder "original sources" under

the name Main Menu.psd. (It can be copied from \Sources\Project objects\original sources).



To add this menu to the project, it needs to be imported.

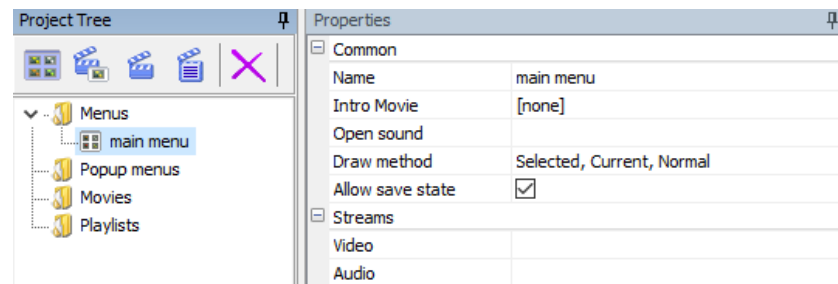
Go to the Project Tree window of BDS and select the “menus” item. Right click and from the dropdown menu select “Import PSD”.



Navigate to the \original sources\main menu.psd file and import it as a “Blu-Disc Studio Format”.

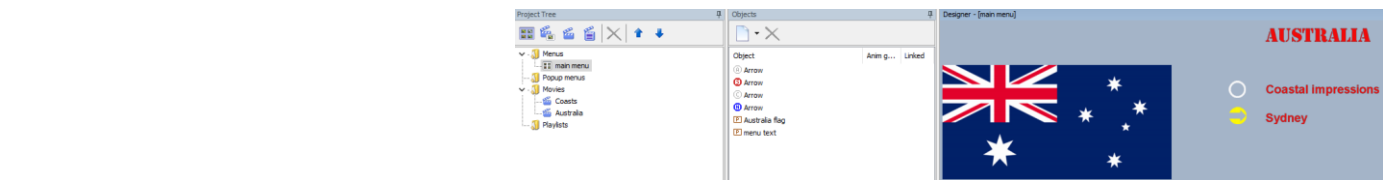
This creates a “main menu” menu – the same name as the Photoshop file. At the same time, BDS creates a project folder also called “main menu” that contains the Photoshop layers as single .png files.

You can change the menu name in the menu properties if you like.



In the Objects window that belongs to this main menu you can see the various objects that are imported: the pictures and the buttons.

The full-sized background layer in army colour can be deleted (in BDS Objects window or by deleting it when shown in the Designer window) as it serves no purpose. The current MenuBR project windows for Project Tree, the menu Objects window and the Designer (with chosen blue background) window will then look like the figure below.



How to arrange the objects in Designer window we'll discuss in a following section after we go over the alternative ways to make this menu.

Just to keep source files at an easy to remember location, we move (or copy) the button state images (arrowN.png and arrowS.png) into a special \buttons folder of the project (create it if it doesn't exist) and modify the button "normal" and "selected" state image locations accordingly. This way any future menus we may add to the project don't need any buttons defined in their Photoshop files – the buttons are all re-using the files in the \buttons folder (e.g. in J:\BDS Projects\MenuBR\buttons).

Menu through Objects window


The same menu can be made from individual .png files that were created with any image editor that can create .png files for menu pictures and button states. These button files can have any name, but it is useful to adhere to the naming standard BDS applies when it imports Photoshop files, i.e. the files are named <button name><state>.png where <state> is a single character "N", "S", "C" or "A" depending on the state it represents. The figure below shows the .png files of the "selected" and "normal" states. The checkered backgrounds are transparent. The files are very small and the images are no larger than the pictures that represent the states. (the Normal state colour changed here to grey as its white makes it difficult to see). For visibility, the button size should be 40 pixels high or more.



Textual objects can be either stored as an image in a .png file or they can be added as a new "Text" object. (The example files are provided in \Sources\Project objects\buttons but also in \Projects\MenuBR\original sources).

Store the .png files in a separate \buttons folder as a common repository for all button images used in the project.

Next you need to define new objects in the menu manually. For this, perform the following steps:

1. Create the "main menu" as placeholder in the Project Tree window.
2. In the Objects view create a new object (button )

3. Select a static object being an image file. Navigate to the “Australian flag.png” (you may have stored it in \original sources of the project once copied from the examples’ \Projects\Project objects\menus). Add it by clicking on “OK”
4. Add a new object for all other picture objects (here only: “\Projects\MenuBR\original sources\main menu text.png to the project local \original sources folder)
5. Add a new object being a button. Specify for its “selected” state the image that you made for this. (copy \Projects\MenuBR\original sources\ArrowS.png to the project \buttons folder)
6. Repeat the previous step for the “normal” state of the button (ArrowN.png)
7. Repeat this step for all other buttons and states (but there are none for this project)

In fact by doing this, you manually perform the same actions as the import of the Photoshop layered file achieves.

Step 4: Layout of the menus and duplicating buttons

Once the menu is created through Photoshop or the Objects window, you need to arrange the objects so they are at their proper place.

This means performing the following tasks:

- Positioning all picture objects
- Bring objects in proper order (modify their order in the menu objects window)
- Copy button as many times as needed onto the same menu to allow the viewer to select each menu item
- Rename copied objects
- Position buttons

Remember: the menu movie resolution determines the size of the menu. In our example we use a (black) movie of 1920x1080 pixels which is the full sized menu.

Position picture objects

The Designer Window is vital in the layout of a menu. It is opened via View > Designer (or press F2). When a menu is chosen from the Project Tree, all objects of this menu are shown in the Designer window. The properties of a selected object are shown in the Objects window.

You select an object by clicking on it in the Designer window. This updates the Properties window to show its properties.

It also works the other way around: select an item in the Objects view and this object is highlighted in the Designer window.

All menu objects occupy their own layer. As such they can overlay each other or completely hide objects on lower layers.

The state layers of a button can be moved independently or are also locked together. The latter is the more usual situation. A button is a

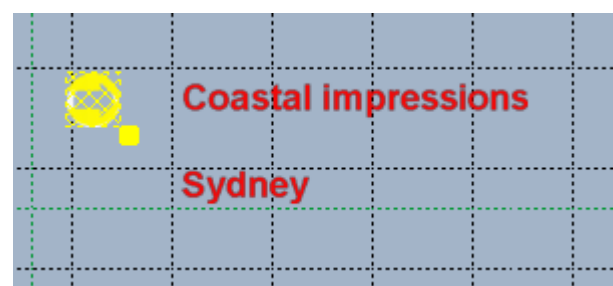
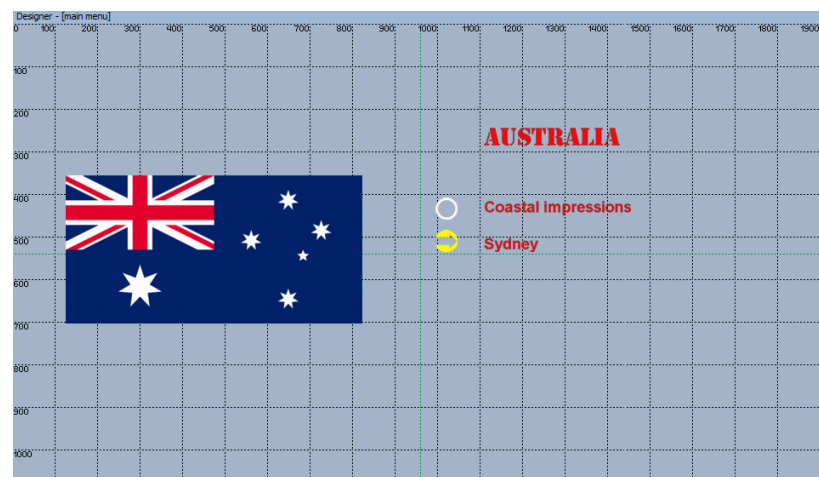
single object on the screen and usually has different images for its states that are stacked at the same position – the higher state layers overlapping the lower states. The “Normal” state image layer is at the bottom. If the various state images are not stacked and locked, you can move them independently. Locking and unlocking of button state images is achieved by clicking on the “Move whole button” menu

button (is the independent state move (clear background), is the whole button move (blue background)).

For our project, move the Australian flag object and the menu text to a suitable position.

Because in designing the button in Photoshop we wanted to see both states at the same time, their images are not on top of each other. For a button to look and feel like a button, we need to bring both normal and selected state images together. Unlock the “move as one button” state and move the “selected” state arrow on top of the “normal” state circle. This gives the visual impression of a single button. Lock the button states so they will move together as a single button.

Move the one button “Arrow” (with both states superimposed) to align with the “Coastal impressions” menu text.

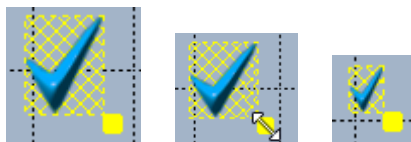


Resizing objects

From within the Designer Window you can not only reposition objects, you can also make them larger or smaller. Keep in mind however that scaling does not change the size of the object in terms of pixels stored in the JAR menu file. If you run out of pixels that can be stored, resize the menu objects using an image editor that physically reduces the size

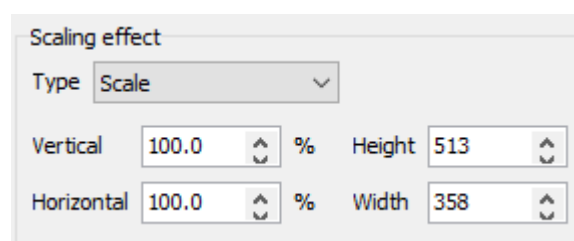
of the object images. Only this reduces their size inside the JAR file when used.

At the right hand side bottom a small yellow square is shown on the selected object. This allows you to pull on it with you mouse and make the object bigger or smaller. Although the mouse changes to a diagonal shape, you can also stretch or squeeze the object to any width or height desired.



BDS has no “keep aspect ratio” option. To ensure a change in size keeps the aspect ratio in tact, you need to explicitly set the scaling factor to the same percentage for horizontal and vertical scaling.

This can be done by selecting the object¹⁴ and through a right mouse click select the "Change Effects" which opens a window that contains a dropdown box for Scaling Effect with choice "no" or "scale". If you want to maintain the aspect ratio, specify the same scaling percentage for horizontal and vertical direction.



For proper functioning of the disc and its authoring, it is best to not scale or only slightly.

Order object layers

Each menu object occupies a layer. The layer stack order is determined by the order in which the objects are listed in the Objects window. The first object in that list is in the top layer. Lower positioned objects can be obscured by a higher positioned objects.

To avoid any button to be ever be hidden behind a picture layer, all picture objects should occupy the lowest menu layers whilst the buttons must occupy the top layers.

Moving a layer one level up or down or to the top or bottom is achieved by selecting the object (in Object or Designer window) and then clicking on the repositioning buttons shown below.



You can see these position buttons in the Objects window and in the main menu of BDS. If you want it only in one of the two places, use

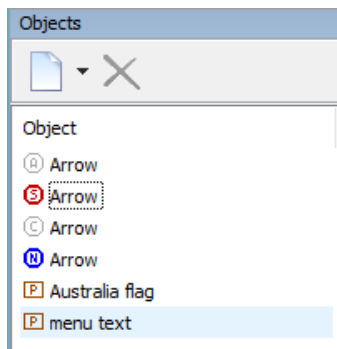
¹⁴ You can also select the object in the Objects window and via a right mouse click select the “Change Effects”

Tools > Options > Interface tab and select the desired behaviour in the “Show object move buttons” dropdown list

Show object move buttons On the toolbar and in the object list ▼

The button layers (4 sub layers, one for each button state) can only be repositioned together in the Objects window. Select one layer of a button in the Object window and all layers of the button are selected.

It is customary to position the top button on a menu as the top button in the Objects window. The second button as the second button in the Objects window and the last button at the lowest position, just above the picture objects.



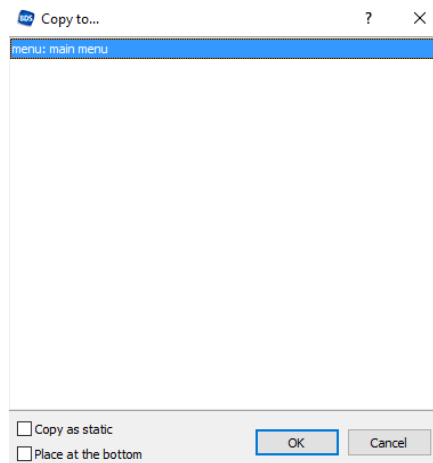
Although customary, it is the navigation between buttons that decides in which order the viewer will experience the move between buttons.¹⁵

Copy buttons

Any menu item (listed in the Objects window) can be duplicated as often as required and to any existing menu.

We need two buttons for our menu as it lists two movie titles to choose from. Duplicate the one button by selecting it in the Designer window and then right-click to open the dropdown menu from which you select “Copy to...” (or press “CTRL/C”). A window opens to ask for the destination menu. That can be the same (or only) menu or any other menu. This way it is easy to define a single button and copy it multiple times on the same or other menus.

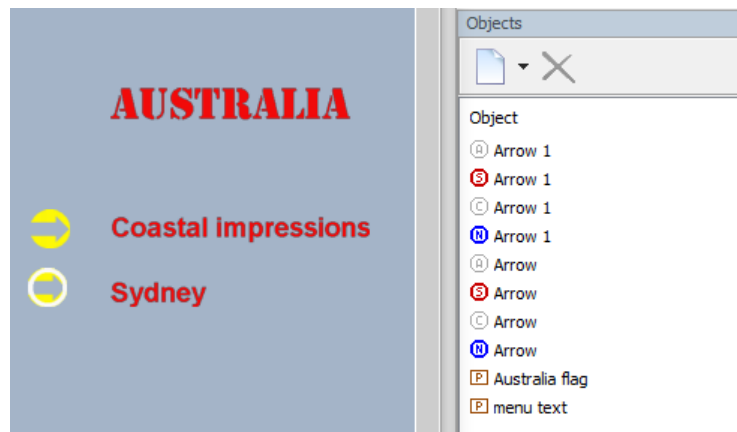
¹⁵ If you want a carousel type menu, the menu buttons are sorted from top layer to bottom layer so then the order is important. We will discuss carousel menus later (in section Carousel menus on page 154).



When copied to the same menu, you may wonder where the copy is copied to. You may be surprised to learn that the copy is exactly on top of the original. Use the mouse to select the button and move it sideways. The original button remains put, the copy will move with the mouse.

From experience, it is preferable to order buttons the same way as they are positioned on the menu. When copying buttons, this means that the original is positioned as lowest in the Objects window. The copies are automatically placed higher in that window and should manually be moved to higher positions in the Designer window.

For our project, we need to copy the Arrow button only once and move the copy to align with the “Sydney” menu item. The button copy is named the same as the original but a sequence number is added (Arrow 1)

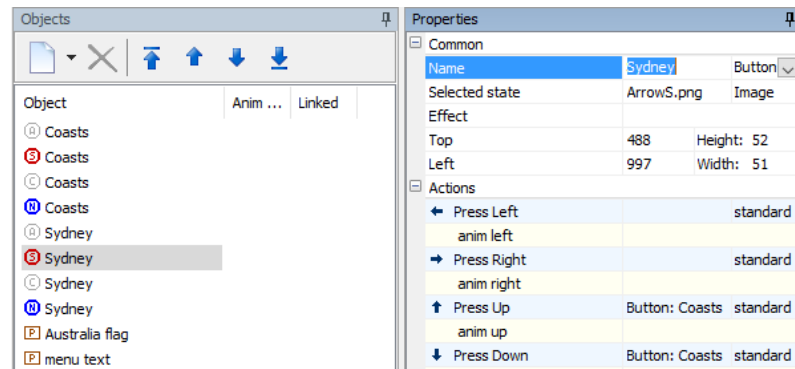


Rename objects

Although not required, it is extremely useful to rename objects such as buttons to a meaningful name. The pictorial objects often already have proper names, taken from their .png file names. But buttons may benefit from more descriptive names than “Arrow”.

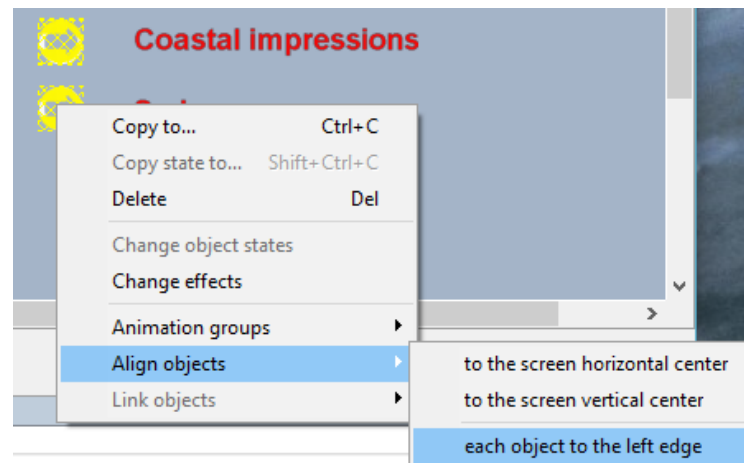
To rename a button, you need to modify its “Name” property in the object’s Properties window. Select the button (in Designer window or any state in the Objects window) and rename it.

For our project, “Arrow” is better named “Coastal” and “Arrow 1” becomes “Sydney”. Use the Objects window to move “Coasts” to be the button at the top.



Position buttons

During duplication of buttons they may not be very well aligned if they are positioned in vertical or horizontal order. To properly align them, select all buttons in the Objects view. This also selects all of them in the Designer window. Move your mouse to the selected buttons in the Designer window and right click for the dropdown menu. Select “Align objects” and select the proper alignment.



If there is no background image that is positioned below the buttons, you can also select the buttons directly in the Designer window. Selection can be done by drawing a rectangle around all buttons using the pressed left mouse. Right click will open the dropdown menu from which you can select “Align”.

If there is a background below the buttons, selection of the buttons must be done individually using the CTRL key. Or you select them (using the CTRL or SHIFT key) in the Objects window.

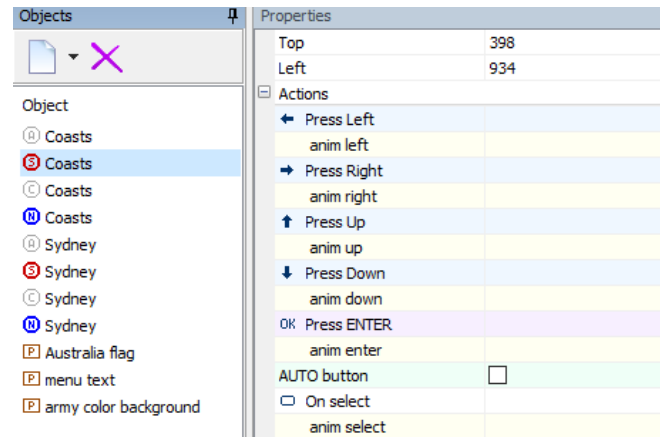
Step 5: Add button operations

Now that the menu looks exactly as you wish, there are two things left over to do:

- specify how to move from one button to the next (and reverse): navigation
- what happens if “OK” is on the remote-control is pressed and the button goes into “active” state?

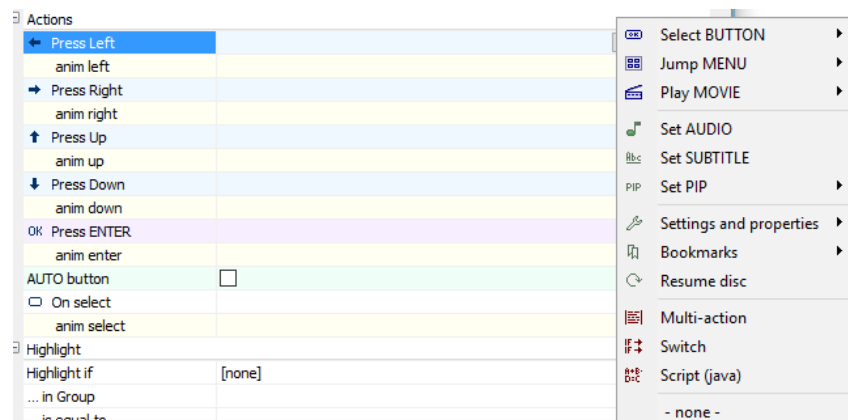
This behaviour is specified in the “button actions” properties. Select a button and the properties in the Properties window will show.

Look for a section called “Actions”.



The figure above shows the properties for the “Coasts” button. Selecting any state in the Objects window selects the whole button. The Properties window indicates what to do if the ← left arrow is pressed on the remote-control (Press Left). And ditto for all other buttons on the remote-control, including the “OK” button (Press Enter).

By clicking on any of the possible actions, the “>” symbol becomes visible at the far right. Click on it and a menu opens listing all options available as action to perform. These include the jump to another menu, to another button, the selection of an audio track, subtitle, but also “none” (to undo any previous action assignment).




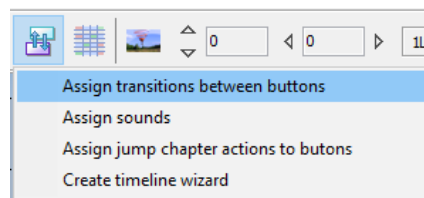
For our project, we need to specify that:

- the Coasts button has a “Press Down” action to select button Sydney
- the Sydney button has a “Press Up” action to select button Coasts

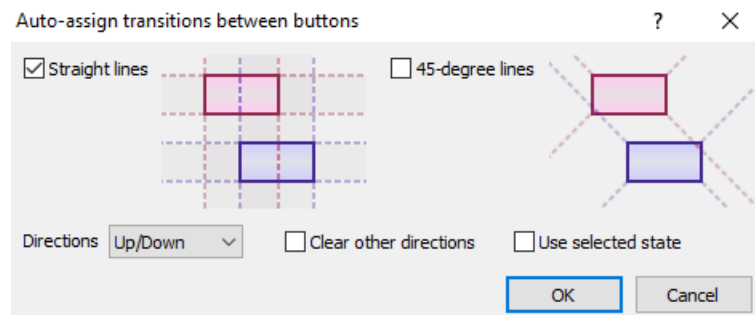
- the Coasts button has a “Press Enter” action to Play Movie Coasts
- the Sydney button has a “Press Enter” action to Play Movie Australia

If you favour circular movement, you may also enter “Press Up” for the Coasts button to select the “Sydney” button. And a “Press Down” for the Sydney button to select the “Coasts” button.

If there are many buttons in (aligned) vertical or horizontal order, you can select all of them (Objects or Designer window) and then select the “Auto-Assign” menu button ().



That opens a window in which you can specify if the navigation between the selected buttons must go from top to bottom or left to right or under a 45 degrees angle.



This allows you to automatically fill in all navigation to the next button or previous button. This way you do not have to do this manually for each button. Make sure no button image overlaps with others. If they do, this auto-assign doesn't work properly. If you want to keep earlier assignments, uncheck the “clear other directions”.

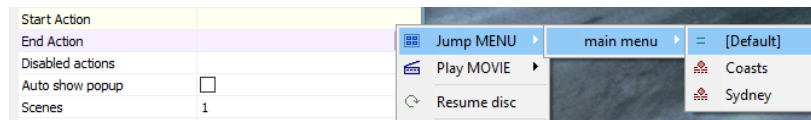
The “Press ENTER” property often will start the play of a movie. It may also open another menu. In this case you specify what button on that menu is pre-selected. If you chose “default” the button last selected on that menu is re-selected.¹⁶

Step 6: Action at end of a movie

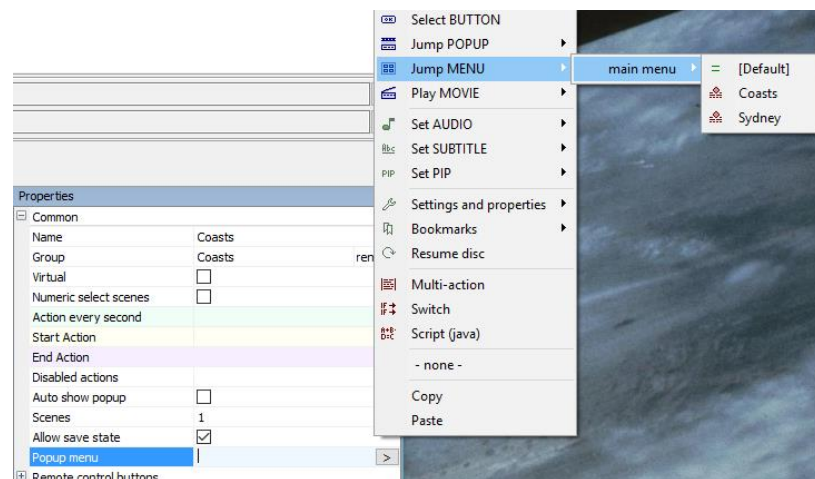
Once a button is selected and activated, its “Press Enter” action is executed. Often this is the playing of a movie. But what to do at the end of a movie? Or when it is interrupted in the middle?

¹⁶ The menu has an Action and an Animation property. When specified, you can open the menu itself or first do the animation and action first before opening the menu. This feature is used in carousel-like menus where before the menu is shown, animation is performed.

For this reason, you need to specify for each movie the “End Action” and “Popup Menu” to be to return to the (main) menu.



If you select an action that jumps to the “Default” button of the menu, the menu is shown and the button that was selected when the movie started is selected again. You can also specify the selection of a particular button. For example, at the end of a movie, the new selected button is the one associated with playing the next movie.



The “Popup Menu” is slightly misused here. In the next project we will properly use it.

For this second project we don’t have a proper popup menu and we (mis)use the pressing the popup button on the remote as a quick way to get back to the menu.

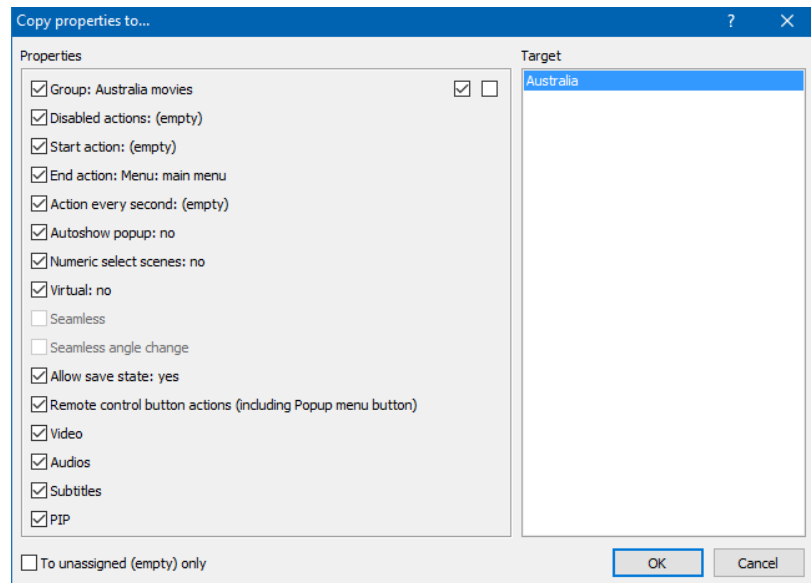
Copy properties of movies

If you have many movie objects that should all have the same End Action (or some other properties), you can quickly achieve that by right-mouse selecting the movie with properties properly specified. From the opening context menu, select “Copy properties to all...”. Alternatively you can copy/paste those values in the Action Matrix window, discussed next.


The “copy properties to all” allows you to check what properties to copy (and whether always or only to those properties not already filled). Click on the right top “blank” checkbox to uncheck all or the “checked” box to check all properties.



Select or unselect the movies in the “Target” list at the right whose properties should be changed. Use the CTRL key to select some movies, use the SHIFT key to select a block of movie titles by clicking on the first title and (while pressing the SHIFT key) clicking on the last title of the movie block.



Action Matrix: copy properties of buttons

The actions specified for movies and buttons are shown in the Action Matrix window (View > Action Matrix (F8) or by menu icon ).

The contents of the Action Matrix window changes with what is selected in the Project Tree window. Certain actions can easily be copied and pasted between movies or buttons in this matrix. Once you know what to do using the Properties window for each button or menu, the Action Matrix can be a real time saver. Especially if many actions are identical for different buttons or movies.

Use one button or movie to provide the right actions (right click on the matrix cell) and then copy/paste the values from that cell to the same cell of other movies or buttons.

The figure below shows all actions available for buttons: left/right/down/up arrow navigation but also Press Enter and On Select.

Action matrix								
Buttons Enter animation/action Remote control								
Button	Left	Right	Up	Down	Enter	On select	Highlight	
all	Button: 11	Button: 11	Button: 11	Button: 11	PL: all [1]			
animations								
sounds								
11	Button: all	Button: 12		Button: 21	Movie: Movie 1 [1]			
animations								
sounds								
12	Button: 11	Button: 13		Button: 22	Movie: Movie 2 [1]			
animations								
sounds								

Step 7: Add menu background movie

The final steps to take before a bluray disc can be created is specifying the background movie for the menu and where to start if the disc is played back. And to return to what menu if the Title Menu button on the remote-control is pressed.

Select the properties of the menu and look for the “Streams” section of the menu property. Here you specify the video and audio stream for the background movie.

For our project we use the “menu” movie that has nothing but rolling waves and some background music.

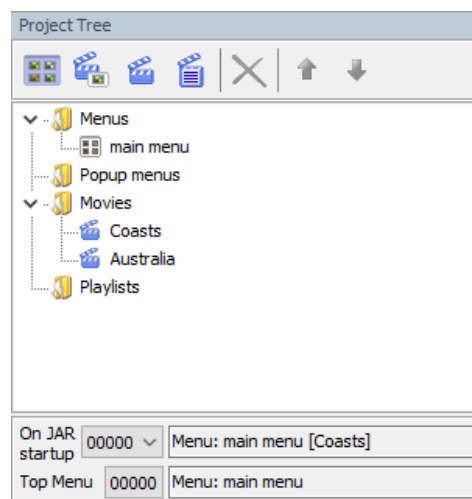
Streams	
Video	J:\BDS Projects\MenuBR\films\Menu.264
Audio	J:\BDS Projects\MenuBR\films\Menu.ac3

If you forget this, the muxing operation will abort with an error stating this omission.

Step 8: Where to start playing

If the disc is to play in a set top player, it must start somewhere. The “On JAR startup” item in the Project Tree window specifies where to start. It's what the player will first play when a disc is inserted (it used to be called "First Play" in earlier versions)¹⁷

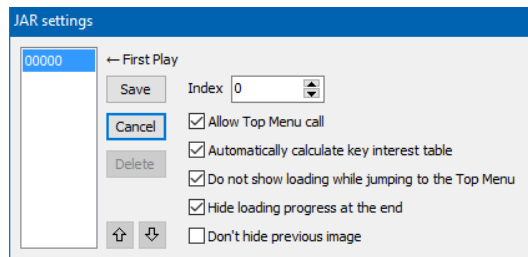
For our project, we start in the main menu and select the button “Coasts” next to the “Coastal impressions” menu text to be initially selected.



In addition, you may enter “Top Menu”¹⁸ (not present in “BDS Lite”) to point to a menu that should be opened if the viewer presses the “Top Menu” (also known as “Disc Menu”) button on the remote. This may occur while a movie plays. The movie is then interrupted and the disc is reloaded, showing the “Top Menu” menu.

¹⁷ As of version 4.3.19 The "First Play" has been renamed into "On JAR startup". When a disc starts, a Java Archive (JAR) is run which activates the item specified (like main menu selecting the Coasts button).

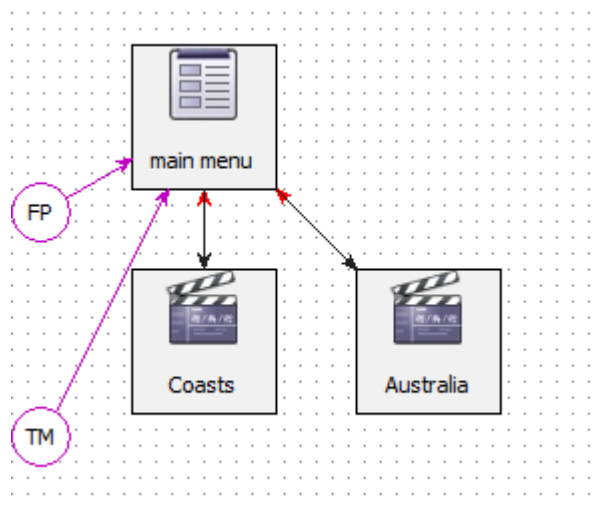
¹⁸ For the Top Menu item to be eligible, you need to enable the Top Menu feature. For this, select the JAR 0000 file (via menu Project > JAR Settings , select the "Edit" button and check the "Allow Top Menu call" option.



Step 9: Check the project: simulation

Menu structure

The whole set of navigational and action instructions specified are shown graphically in the “Structure” window (View > Structure (F7)),



It shows with “FP” the “First Play” that starts the disc, “TM” the Top Menu (if enabled in its JAR file) and how one can perform an action from the First Play menu (black arrow) to play a movie and what happens if you press the popup menu (red arrow – in our simple project it simply returns to the menu). When the movie runs to its end a blue arrow indicates what happens next (but it may overlap with identical actions indicated by other arrows in different colours).

The bottom of the Structure window indicates the meaning of the various coloured arrows.

End action, Auto-close action Auto-actions, Start action, Action every second Popup Includes (movies in playlist, intro movie)


This menu structure visually represents the navigation implemented. It can quickly indicate if there are orphan objects that are present but can never be reached or whether there are dead-end alleys in which you can get to but never can get out of.


The colours of the arrows can be set in Tools > Options > Interface. The setting applies to all projects.

With many movies or menus this structure overview quickly becomes less useful as action lines seem to merge together. The ultimate test is to perform a navigation simulation.

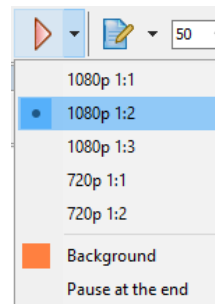
Simulation

Simulation only checks the navigational objects of the project. In fact it runs as a Java program executing the JAR file with Java code that realizes the menu.¹⁹

You can exercise these thoroughly to ensure all menus are linked together, each button can be reached. The simulation objects are all stored in a compiled Java archive. As such, if you change any of these items, all you need to do is recreate the archive by using the  button on the BDS menu bar. This avoids a lengthy remuxing of the actual movie files.

The menu has a special “Play” button  to simulate the working of the menu navigation you specified. (Simulation (Test Menu) (F9)). This will compile all navigational information and menus into a Java archive. Hence simulation can be done quickly without the need to mux the movie files.

If the native 1920x1080 or 1280x720 pixels for HD movies is too large for your pc monitor, before clicking on “simulation”, first set the reduced size by which the simulation window must be displayed. At the same time, you may wish to change the background colour that represents transparent pixels of menus.



During the simulation, the “On JAR startup” specified menu is shown and you can try out the navigation between buttons. You use the keyboard arrow keys to move between buttons and use the <enter> key to simulate pressing the OK on the remote-control. If this OK jumps to another menu, that menu will show. Upon return to the earlier menu you can check whether the specified button or the “default” (=last OK-ed) button is selected.

Because the simulator only simulates the menus, it cannot run any movie. Instead, if a selected item is activated to show a movie, a black window is opened. This contains help text on how to activate the “End Action” and “Popup Menu” action. If no action is defined for either, you’re “stuck” and clearly your design work is not completed yet. Stop simulation by closing its window (through the X icon at the top right of the window).

¹⁹ The simulator runs a single JAR file. Usually BDS creates only a single JAR file for all menus. However as of BDS V4.3 you can create multiple JAR files. In those cases, the simulator runs only the JAR code in a single JAR file. In the “On JAR Startup” in the Project Tree window you specify what JAR title to simulate.

Now playing [Coasts]
Press END for EndAction or F5 for Popup.

Note that if you transfer from BDS Lite, the “END” key is specific for BDS Standard and MX. For BDS Lite it is F1.

The simulation uses certain keyboard keys as stand-ins for remote control buttons:

- F1 – Red button
- F2 – Green button
- F3 – Yellow button
- F4 – Blue button
- F5 – Popup menu button
- End – movie/playlist End Action
- 1..0 – Numeric buttons 1..0
- P – Play button
- U – Pause button
- F – Fast forward button
- R – Rewind button
- < – Previous track button
- > – Next track button

The simulation stops when the simulation window is closed (use the top right X window button). Repeat simulations as often as needed until you’re satisfied the entire navigational structure is correct.

Before simulation starts, BDS first compiles the Java code of the project to ensure all objects are present and connected. Where a movie placeholder property has no “End Action” or “Popup Menu” action defined, this is logged and you can take action to correct this first before starting a new simulation.

```

mode: Simulation

13:51:34 - Validating...
13:51:34 - Preparing...
13:51:34 - Adding video asset for "main menu"
13:51:34 - Adding audio asset for "main menu"
    PL BDID = 00001 (menu "main menu")
13:51:34 - Adding video asset for "Coasts"
13:51:34 - Adding audio asset #1 for "Coasts"
13:51:34 - Adding audio asset #2 for "Coasts"
13:51:34 - Adding subtitles #1 for "Coasts"
13:51:34 - Adding subtitles #2 for "Coasts"
    PL BDID = 00002 (movie "Coasts")
13:51:34 - Adding video asset for "Australia"
13:51:34 - Adding audio asset #1 for "Australia"
13:51:34 - Adding audio asset #2 for "Australia"
13:51:34 - Adding subtitles #1 for "Australia"
13:51:34 - Adding subtitles #2 for "Australia"
    PL BDID = 00003 (movie "Australia")

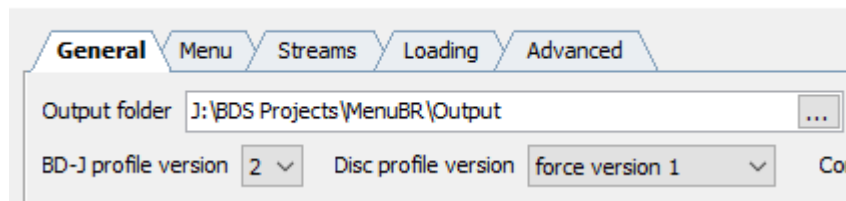
13:51:34 - Preparing menu...
13:51:34 - Optimizing...
13:51:35 - Step 1/4 ...
13:51:36 - Step 1/4: done
13:51:36 - Step 2/4 ...
13:51:36 - Step 2/4: done
13:51:36 - Step 3/4 ...
13:51:37 - Step 3/4: done
13:51:37 - Step 4/4 ...
13:51:37 - Running ...
13:51:41 - Step 4/4: done

Source graphic size: 350 336 px
Result graphic size: 365 054 px

```

If an error occurs stating “Image buffer overflow in profile 1”, change the project properties to use profile 2 (menu Project > Project Properties > General tab, “BD-J profile version” set to 2).

Project properties



If the overflow message returns, it means there are too many image pixels in the JAR file and you may need to divide the code over several JAR files (see Using multiple JAR titles on page 210) or reduce the images used until the overflow message disappears.

Step 10: Build the disc


This step is identical again to the step described in the previous chapter (see “Step 4: Build the disc” on page 64). Make sure the Project > Project Settings > General tab points to the correct \Output folder (especially if you copied a project into a new folder and need a new output folder to avoid overwriting the one of the other project). An additional problem may occur:

- Access violation at start of the disc creation – odd error message that usually means
 - “you did not specify a menu movie”

- you changed a movie file (.264/.ac3) but did not remux the playlists in which it features
- you made many modifications (in chapter marks) and BDS has not kept up with them. Closing BDS (+save) and reopening it may solve this problem.

The progress of the muxing and final result of the building is shown in a separate window shown below. The same text is also copied in a log file in the \Log project folder.

The first part repeats the simulation process in as much as the menus are now build, stored in a Java Archive that is signed (if you want to) and copied to the __JAVA as well as the \Output project folder.

If you need to change any of these navigational items (menu, buttons, animations), all you need to do is recreate the archive by using the  button on the BDS menu bar and confirm you want it copied into the output folder. This avoids a lengthy remuxing of the actual movie files.

```

mode: Mux by internal muxer

13:40:43 - Saving...
13:40:43 - Preparing...
13:40:44 - Adding video asset for "main menu"
13:40:45 - Adding audio asset for "main menu"
    PL BDID = 00001 (menu "main menu")
13:40:45 - Adding video asset for "Coasts"
13:40:46 - Adding audio asset #1 for "Coasts"
13:40:46 - Adding audio asset #2 for "Coasts"
13:40:47 - Adding subtitles #1 for "Coasts"
13:40:47 - (notice) correct timing for 23.976
13:40:47 - Adding subtitles #2 for "Coasts"
13:40:47 - (notice) correct timing for 23.976
    PL BDID = 00002 (movie "Coasts")
13:40:48 - Adding video asset for "Australia"
13:40:49 - Adding audio asset #1 for "Australia"
13:40:49 - Adding audio asset #2 for "Australia"
13:40:50 - Adding subtitles #1 for "Australia"
13:40:50 - (notice) correct timing for 23.976
13:40:50 - Adding subtitles #2 for "Australia"
13:40:50 - (notice) correct timing for 23.976
    PL BDID = 00003 (movie "Australia")

13:40:51 - Preparing menu...
13:40:51 - Optimizing...
13:40:51 - Generating fonts...

13:40:51 - Step 1/4 ...
13:40:52 - Step 1/4: done
13:40:52 - Step 2/4 ...
13:40:52 - Step 2/4: done
13:40:52 - Step 3/4 ...
13:40:53 - Step 3/4: done
13:40:53 - Step 4/4 ...

13:40:54 - Starting sign process...
13:40:54 - [warning] You use the default values for Organization IDs.
13:40:54 - OrgID: 7fff2222, AppID: 4000
13:40:54 - Signing...
13:40:55 - Sign process finished

13:40:55 - Step 4/4: done

Source graphic size: 350 336 px
Result graphic size: 365 054 px

13:40:55 - starting mux process...
13:41:02 mux bd type: v200. recording rate: 6 000 000.
13:41:02 compiling destination project...
13:41:02 destination bdmv path is: j:\bds projects\menubr\output\
13:41:02 starting mux 00005.m2ts ...
13:41:03 thread: starting mux 00003.m2ts ...
13:41:36 finished 00005.m2ts.
13:41:36 starting mux 00001.m2ts ...
13:41:36 thread: finished 00003.m2ts.
13:41:55 finished 00001.m2ts.
13:41:56 elapsed time: 00:00:53.4625878
13:41:56 compilation is finished.
13:41:56 - finished mux process

Fixed images:    8
Src overs:      7
Groups:         3
Fades:          0
Translators:    0
Scales:         0
Clipped:        0
Assemblies:     2
RC handlers:    27
Segments:       17
Segment-dones:  1
Other:          3
TOTAL:          68

13:41:56 - Finished (warnings: 1, notices: 4)

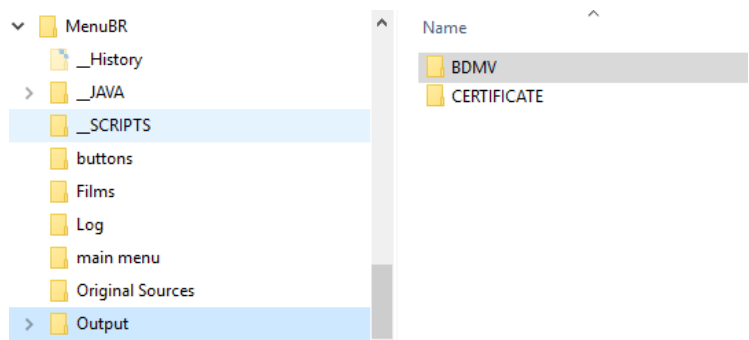
```

The second part performs the muxing of the movies where video, audio and subtitles are combined. This muxing may open and close several windows if tsMuxer is used, that show the process of the

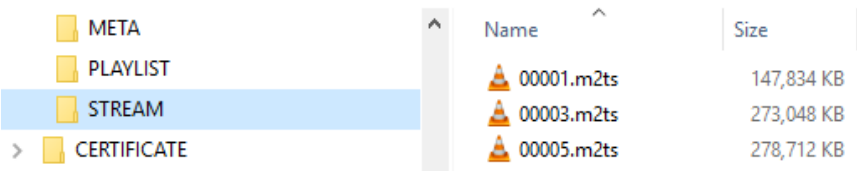
muxer. It should end with “Finished” rather than “Terminated”. The latter means one of the video components did not adhere to the BDA standards. The MX internal muxer is much stricter in this than the tsMuxer. The disadvantage is that the muxing process may end as “Finished” whereas tsMuxer simply aborted the muxing of the offending movie streams. Usually in the final phase of the disc building process (where the playlists are updated) BDS cannot find certain information and will complain about files missing.

This window can be closed once the build process completed before you can do anything else. The log file is available in the \Log subfolder of the project.

The bluray compliant file structure is written in the folder indicated in the Project Properties:



with the actual video movies stored in the STREAM folder – just as large as their original source input (before they were demuxed).



Step 11: Test run the disc

This step again is identical to the one described in the previous chapter (section Step 5: Test running the disc on page 69).

Make sure menu and menu movie have the same resolution as otherwise the menu is only shown with the movie resolution.

Step 12: Burn the bluray disc

This step is identical to the one described in the previous chapter (section Step 6: Burn the bluray disc on page 71)

Project 3: Creating a bluray disc with setup menu for audio and subtitles

The project goal

We expand the created second project by

- add a “play all” play list
- adding an intro part of the menu movie played before the menu is shown
- add a “set up” menu to specify audio and subtitle track (introducing the “current” button state)

Playlists is also BDS Java generated code and if changed, recompiling the JAR file is sufficient.

Only the 3rd item is also available in the Lite edition.

If you want to create this project yourself, the source files can be found in the .zip file mentioned in Appendix A: The user’s guide source files.

Play lists

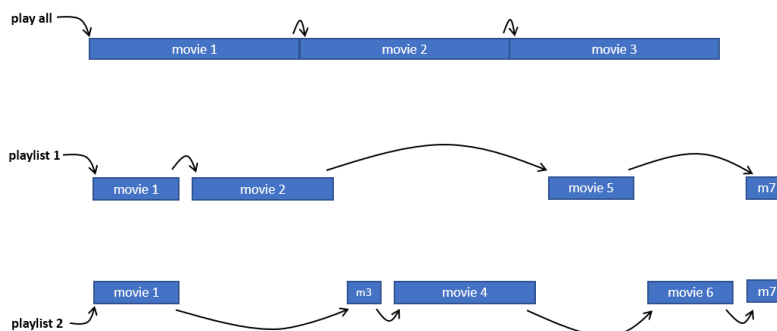
Use of playlists

A playlist is an instruction to BDS to play a set of movies. Upto now every single movie has its own playlist with only the movie itself on it.

But you can create additional playlists. Such a list combines several movies that are played in sequence as if it was a single long movie. A corresponding menu text could be “Play All”. It can also be made a bit more elaborate as a “binge watch” of single episode movies, skipping their intro, “previously on”, “next episode” scenes and end credit titles.

All movies in a playlist must have the same BDA characteristics. Same resolution, same frame rate, preferably same number of audio and subtitle tracks and equally preferable in the same order.

Some possible constructions for playlist elements is shown below.

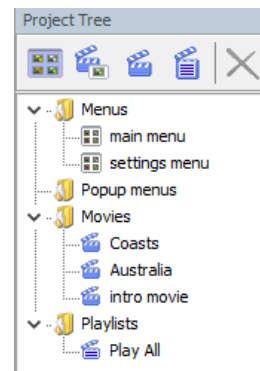


- The first illustration above gives the simplest use of a playlist. The playlist is called “play all” and that’s what it does: movies 1, 2 and 3 are made part of the playlist. Selecting this “play all”

list starts running movie 1 and continues with movies 2 and 3. The transition or jump between movies appears seamless.

- The second and third illustration shows how seven movies are joined into two playlists.
 - Playlist 1 starts with movie 1, plays movie 2, then movie 5 and finally movie 7.
 - Playlist 2 plays movie 1 and then movie 3, movie 4, movie 6 and finally movie 7.

Playlists have their own branch in the Project Tree and all playlist placeholders act just as individual movie placeholders – with many of the same properties as movies²⁰. Yet they are slightly different from regular movies. Hence the “Clone properties to all...” context menu works for cloning playlists, but not to movies. And vice versa.



A playlist is referred to the same way as a movie. The “Press ENTER” property of a button can have an action to play the playlist.

The playlist placeholder properties contain a “Movies” item where all movies to be played can be specified. It also contains, like any movie, an “End Action” to specify what to do at the end of the playlist and a “Scenes” property. There is also a “Popup menu” property for a popup menu specific to the playlist.

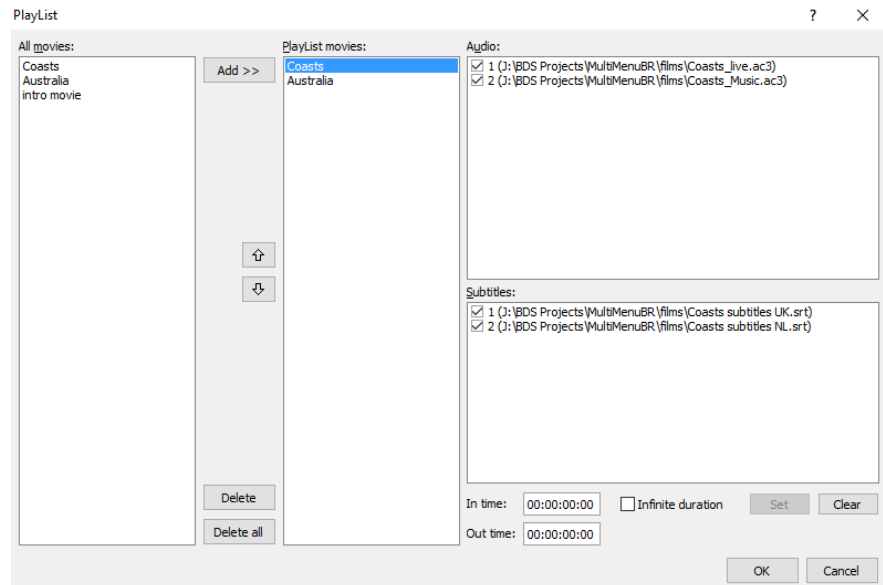
A playlist can have different scenes than the individual movies it is made up of.

End Action	Menu: main menu
Disabled actions	
Auto show popup	<input type="checkbox"/>
Seamless	<input checked="" type="checkbox"/>
Movies	Coasts, Australia
Scenes	1

- When you select the “Movies” property you open the dropdown menu when you click on “...”. A list of all movies is shown. The ones to add to the playlist must be selected and by clicking the “Add >>” button they are added to the list. Clicking “Delete” and the selected movie is removed from the list. You always select complete movies. If you want to show only parts

²⁰ Like movies, playlists are not stored in a menu JAR file. A change in a playlist means the entire disc must be remixed.

of them (starting with a chapter) you need to direct-edit the chapterlist (as explained in Project 10: watching episodes without titles” on page 296).



Note that a new playlist acts like a new movie. It has no chapter (play)marks defined even if the individual movies do have them.

Chapters for a playlist are added the same way as for a single movie: either manually or generated with certain time intervals between chapters.

There are two distinct ways of opening the Scenes windows:


- Standard (either select the playlist placeholder and double click or press ENTER or select its “Scenes” property and open it)
- Direct Editing (select playlist and press SHIFT-ENTER)

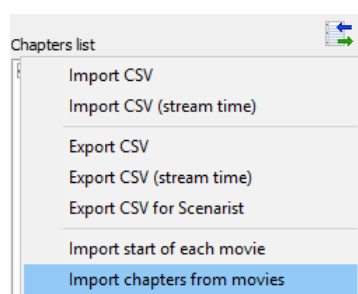
The standard Scenes window will be the same but “Direct Editing” Scenes allows actions to be assigned to chapters (like jump to another chapter). A full discussion is deferred to section “Interlude: Chapter marks – standard or direct editing” on page 199.

In addition, the Playlist properties include an option “Browsable audio”. This is only useful using the MX internal muxer. When you click on its “>” button at the far right end of the property, a navigation window opens to select a music file (WAV, AC3, MP3 and many other formats). If specified, the playlist will become a browsable slideshow (see Slideshows on page 375) and videos in all movies will be marked as browsable. All movies in browsable slideshow should be without audio/subtitles and you can’t use angles here.

Reuse chapters from individual movies or other discs

To reuse the chapter marks made in the individual movies of which the playlist is composed, open the “Scenes” window of the playlist in either mode. You then see at the top right-hand corner of the chapter

list a button () that on click reveals a dropdown menu. The last menu item allows you to import all chapter marks of the individual movies. Or, if you just want the start of each movie to be marked: take the one but last option.

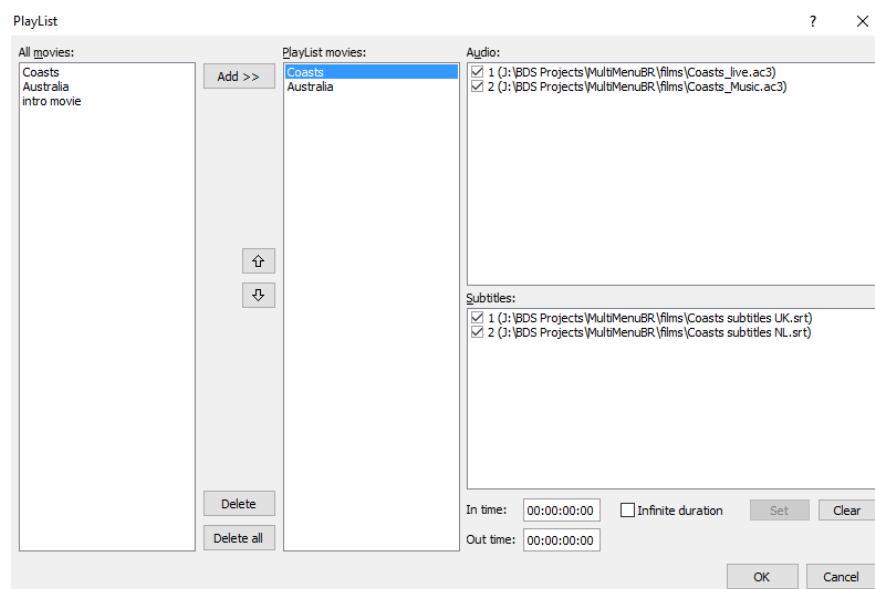


The CSV options (comma separated values) are only of importance for import or export to Scenarist (see online Help topic “Import and export Scenes”).

Note: Importing chapters may cause the “end of movie” chapter to be almost at the same spot as the “start of movie” chapter of the next movie on the list. This may cause the MX muxer to warn you about duplicate chapters. They are no duplicates, but too near each other for MX muxer’s liking and one is ignored. Better even: the duplicates are removed when the “Scenes” window is reopened (manually or by the muxer) and then closed. The warning will not occur again.

For proper functioning of your bluray disc, ensure you specify what to do while a playlist plays or finishes: add actions to it properties “End Action” and “Popup Menu”. Without them, the disc may “hang” at the end of the playlist and not return to any menu or start of other movie.

The playlist can also limit the number of audio or subtitle tracks available. By selecting a movie from the playlist its audio and subtitle tracks are shown in the right hand panel. Unchecking any of them makes them unavailable when the playlist is played. This may be of use if you make two playlists, one for playing movies in English language and one in a dubbed language.



Finally, you may wish not to list certain movies in the normal “Play Movie” list that is shown when a button is pressed (“Press Enter” property of the button). By omitting them from the list they can not be associated with a button. You “hide” those movies by checking the movie property “virtual”.

Virtual	<input type="checkbox"/>
Numeric select scenes	<input type="checkbox"/>

Changes to a playlist movie

A playlist is a collection of movies. Associated with it is a set of chapter marks. Suppose you discover a glitch in the 3rd movie – audio goes out of sync for example. Your disc is all authored and the playlist has been completely set up. Now suddenly the entries of the third movie are replaced by the same movie, but corrected for its glitches. Do you have to redo the entire playlist? Fortunately not if the replacement movie is basically the same one but with errors corrected.

- Add the repaired movie in the demuxed movie stream folder
- Delete the index files that were made for the old movie
- Replace the individual movie in its movie placeholder in the BDS Project tree window
- Regenerate the index files
- You may want to delete all chapters in its Scenes window and re-insert them. But they may also be still valid and need no change
- The playlist will use the renewed index file of the movie. All settings may still be valid (like for the individual movie)
- Rebuild the project

If the replacement movie is an entirely different movie or a substantially edited version of the old one, you do need to re—author the chapters of the individual movie as well as remove the old chapters of that movie from the playlist Scenes window and re-author the new ones. Make a correct decision on authoring in “Standard” or “Direct Editing” mode. The latter is probably more appropriate. (See section “Interlude: Chapter marks – standard or direct editing” on page 199 for a discussion on both methods).

Intro movies

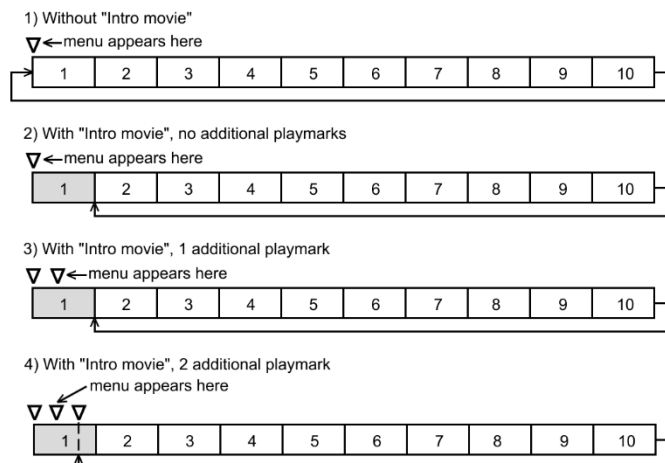
Menu movie and menu objects are two different things (the latter a Java program that executes). As such the Java menu may show before the menu movie starts. That looks odd.

The intro menu is designed to start first and delay the show of the menu objects until the intro menu reaches its 2nd chapter mark. When the intro movie finishes, the regular menu movie takes over. Usually it is the same movie so the viewer doesn’t notice the switch of movies.

Playlists for Intro movie and Menu movie

For any menu you can define an “intro” movie. The intro movie is shown before the “menu movie” of the menu.

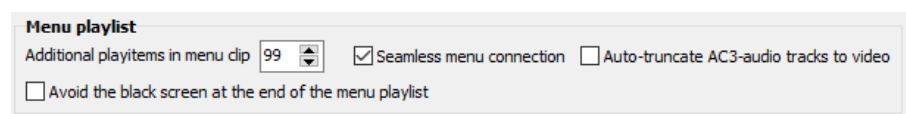
- The intro movie is a regular movie in the movie list.
- It has 2 chapters only (if there are more, they are ignored). You set the chapter 2 yourself.



Chapter 2 of the intro movie signals to show the menu objects from that time forward. This ensures the intro movie plays as background before the menu texts and images are shown.

At the end of the intro movie, the regular menu movie starts playing a number of times. If intro movie and menu movie are the same, the viewer won't notice the switch between background movies.

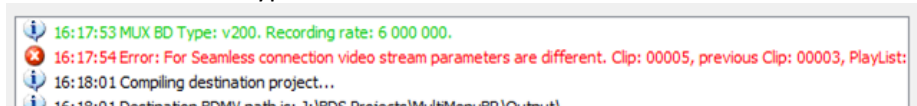
The number of times the menu movie is repeated is set in Project > Project Properties > Menus tab for "Additional play items". When this count is exceeded, the intro movie is played again (but the menu items are already visible) after which another loop of the menu movie starts.



The default setting for the loop count is 99. This implies that if the viewer takes no action, the background movie shows 1 x intro, 99 x menu movie, 1 x intro, 99 x menu movie, etc.

The intro movie should not be started through a menu button but it does have its own movie placeholder. The intro movie must not have an End Action or a Popup menu action – because it is never found playing except within the context of a menu.

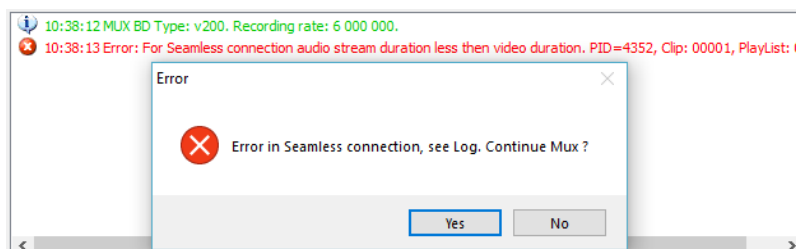
If intro movie and menu movie are different, they must have the same movie specification in aspect ratio, resolution and format to make the transition seamless. The MX muxer will issue an error if the streams are not of the same type:



The tsMuxer does not issue an error in those cases or when streams like MPEG-TS are used, it may even continue to produce the disc but

this disc won't play very well on set top players at the transition between intro and menu movie.

Another error may be seen by the MX Muxer and not tsMuxer when the video and audio stream seem not equally long.



This error may be ignored and the mux allowed to continue – even if the log ends with “Terminated”. The tsMuxer won't see this error at all (and ends with a successful “Finished” in the log). In both cases a playable disc is made.²¹

No menu objects delay

If after the end of a movie a menu with an intro movie is opened, the menu objects won't show until after chapter 2 of the intro movie. If you want the menu text to be shown directly, you need to open an identical menu but without the Intro Movie set. A identical copy of a menu is made via the right click menu option “Clone Tree Item” (CTRL/C) after which you give the menu a unique name and remove the Intro Movie property setting.

First play movies (versus intro movies)

Do not confuse this “intro movie” with first movies. First movies are those that show a studio trailer, a censorship notice, the use of Dolby Digital or other prequels that precede the main menu.

First movies are movies to be played in a sequential chain when the disc is started. They normally can not be selected through a menu button.



See Appendix B: Commercial settings on how to prevent viewers from skipping these opening movies.

Movie groups

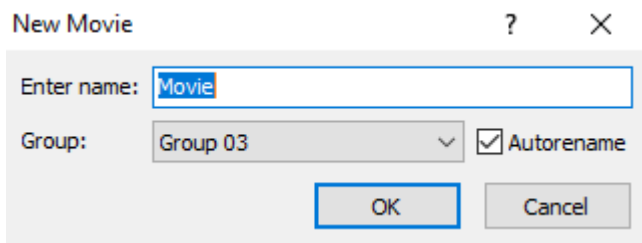
Movie groups allow several movies to be grouped together. As a group they share the viewer's choice in audio and subtitle tracks. A group is no playlist. But if for one movie in a group you select Ukrainian

²¹ If you use the carousel menu generator the intro movie must be specified in the menu you specify as opening menu (“First Play” in the project tree window pre-V4.3, “On JAR startup” for post V4.3). JAR title defined as “First Play” in Project > JAR Settings. It should not be specified in the menu from which you generated the menus.

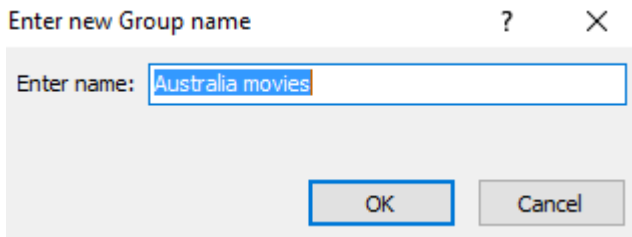
subtitles (subtitle track 3), that choice is automatically applied to all movies in the same group. Movie groups become important with the fourth button state: “Current”.

When groups are used it is important of course that all movies have the same number of audio and subtitle streams and in the same order.

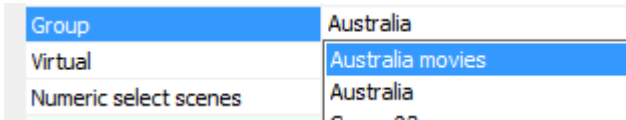
Initially each movie is in its own group. But its “group” property can be changed. You can also use the “Clone project properties” to use the same group name for multiple movie placeholders.



For the two movies we added in the previous project, we define a new group (double click “rename” on the “Coasts” movie group property) “Australia movies”.



Coasts now belongs to the new group. Move to the other movies (only Australia in our case) and the added name occurs in the dropdown list of available group names

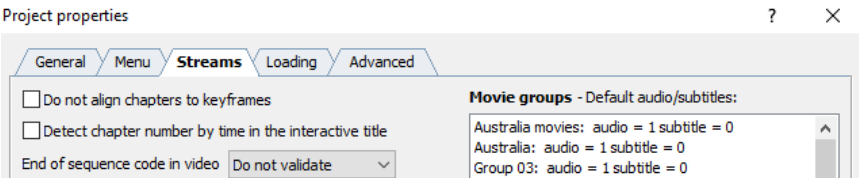


Select the name group name “Australia movies” and both movies are now sharing the same group.

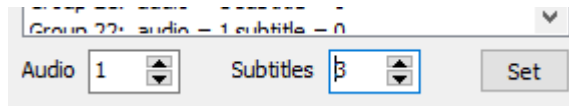
Name	Coasts
Group	Australia movies

Name	Australia
Group	Australia movies

The default group properties can be seen and specified in the window Project>Project Properties>Streams tab.



At the bottom of the overview you can change the default setting for each of the groups individually (select a group, set defaults, click “Set”).



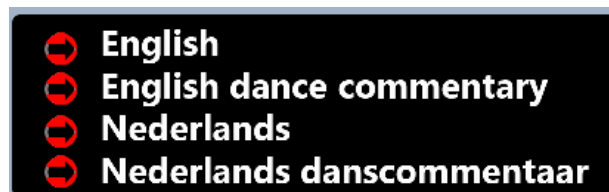
Note: If through a written Java program you change movies within the same group, it ignores the group membership. You need to specify explicitly what audio and subtitle track to use for the movie you switch over to.

The “Current” button state

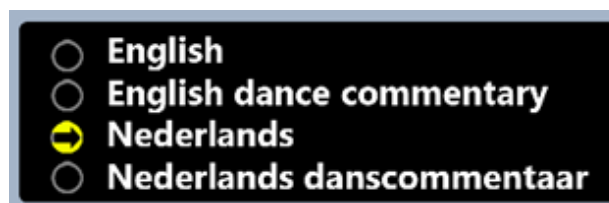
The “Current” state of a button is shown in menus where a previous setting must be shown by highlighting the menu item. Of all buttons on a menu, only one can be selected and that one can be activated. If it is, that button also sets its “current” state. The “current” state acts as a reminder of what button was last activated on a menu.

The image below gives you a menu list with various audio tracks to choose from. These tracks are available to all movies that belong to the same group, in this case we named it WestSideStoryMulti.

Each button has a state “Current” image: a red circle with arrow.

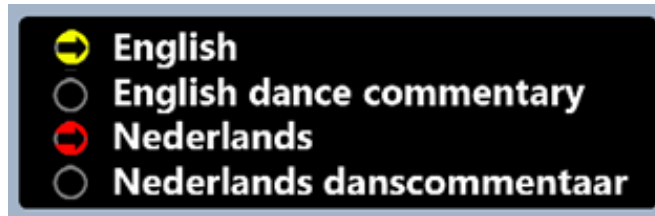


The viewer with a set top player can use the arrow keys to navigate to select “Nederlands” (3rd menu option – corresponds with 3rd audio stream in the movie group). The button state will become “Selected” and shows a yellow circle.

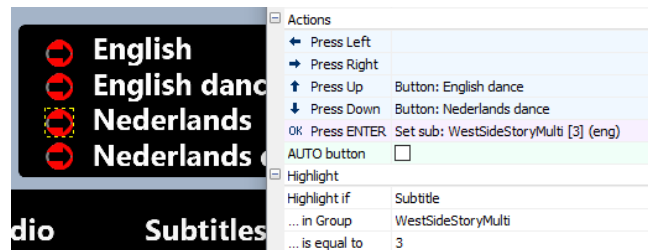


When this choice is activated (by pressing “OK”) to indicate the preferred choice, the Dutch language track (3rd audio stream) will be set as the current one. At the same time, that button’s “current” state is also set. The stream is set for all movies that belong to the same group.

Revisiting this menu later, the previous choice is remembered: the button near “Nederlands” shows as “Current”. If the viewer wants to change, he can still navigate to any of the choices. A yellow circle shows his “selected” choice.



If he presses “OK” on the remote-control while “English” is selected, the English audio will become current (1st audio stream of the group). If you exit without changing your choice, the Dutch track will remain current.



How does BDS know what button to highlight with its “Current” state image? At first we choose the third audio track as active for movies in the movie group (named WestSideStoryMulti in the example). The “Highlight” section shown above for the third button (next to “Nederlands”) specifies a condition: show the “Current” image of this button only when in group WestSideStoryMulti the active audio stream is the third stream (i.e. the Dutch audio).

Any movie belonging to a different group is not affected.

The “Highlight” section is automatically filled by BDS when a Current state is defined for a button, but you can change the settings if you like.

D.I.Y. Adventure

To create the project on your own, keep in mind to:

- Start from the previous project
- Add a menu to select on audio and subtitle track
- Keep a “Current” pointer to that choice made
- Add a menu item to “play all” through a playlist
- Add an “intro movie” to the main menu to delay display of the menu items

Step 0: Ready to run

If you don’t want to create this project yourself, you can setup the project from provided sources following these steps:

1. Copy the \Projects\MultiMenuBR project in the BDS kit.zip to your BDS projects folder (e.g. J:\BDS Projects\MultiMenuBR) You may skip or delete the contents of \output
2. Copy all movie files from the previous project \MenuBR\films into this project’s \films folder (e.g. J:\BDS Projects\MultiMenuBR\films)

Alternatively you can copy them from the BDS kit.zip file
\\Sources\\movies\\demuxed and Sources\\movies\\subtitles.
They include “Coasts”, “Australia” and “menu”.

3. Double click on the project’s MultiMenuBR.bdmd file to open BDS for this project MultiMenuBR.
4. Click the “mux” button and rebuild the disc image (in the project’s \\Output folder).

Step 1: Get organized

This step is mostly identical to the steps specified in section “Step 1: Create new project” of the first project. The new project is called MultimenusBR. Either you create this project entirely from scratch or you copy the previous project to continue from there.

There are two ways you can copy a project:

1. Copy the entire project folder tree of the previous project from the \\BDS Projects\\MenuBR project. Rename the top project folder to \\MultiMenuBR.
2. Use a template. The previous project MenuBR is saved as a template (File>Export Template). Then the template is the basis for the new project (File>Import from Template). Once the template is loaded, it is stored as a new project in a new project folder you create manually.

There are no new project subfolders to add, but a new menu movie, intro movie and menu objects are added to the project in this project.

Step 2: Include the movies

Copy the \\films folder of the previous project MenuBR to the \\films folder of this MultiMenuBR project.(this step has implicitly been executed when you copied the project tree of the “MenuBR” project and renamed it to MultitMenuBR).

The actual films are not needed for the menu design steps – their placeholders suffice for the moment. They need to be there when the final disc creation phase is entered.


1. We need to collect all movies into the same movie group. Therefore, for each movie placeholder’s property “groupname” must be set as “Australia movies” (through its Properties window).
2. The name “Australia Movies” does not yet exist. Therefore, open one movie (e.g. Coasts) and double click on “Rename” on its Group property and change the current group name into “Australia Movies”.
3. The remaining movies must also be added to this group but the new name is now listed in their dropdown list of the group and easily selected. Do this for the “Australia” movie.

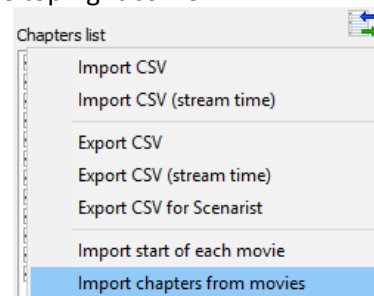
Both movies “Coasts” and “Australia” are now part of the “Australia Movies” group. Setting audio and subtitles for this group sets them for all both movies.

Step 3: Add a “play all” play list

We will add a “Play all” menu item in the main menu using a play list.

The following steps are taken:

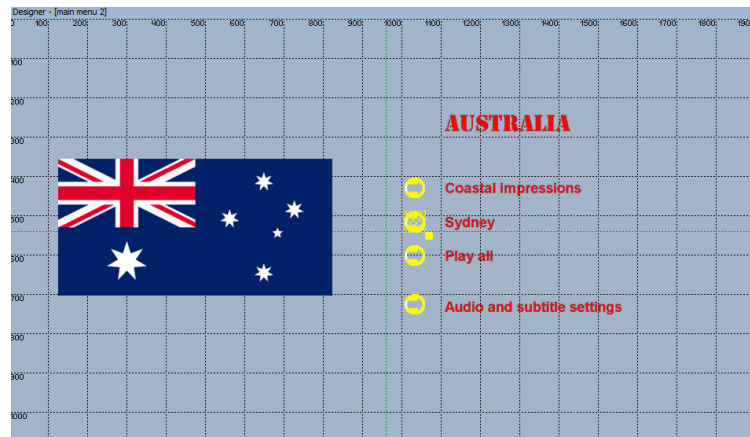
1. Define a new playlist by right-clicking on Playlists in the Project Tree window. This opens a dropdown menu with the “New Playlist” option. Pressing SHIFT/CTRL/L achieves the same, as does pressing the menu button  on the Project tree window menu bar.
2. Give the new play list a name, like “Play All” in its Name property.
3. It can (and should) also be made part of the group of movies its list contains using its Group property. That way audio or subtitle settings for a single movie also affects the setting of the play list.
4. Add all movies that should be played in sequence using the playlist through adding them to the “Movies” property. Here the playlist consists of 2 movies: Coasts and Australia.
5. Open the playlist’s “Scenes”. Initially it will only have a single chapter at the very start of the playlis. Import the chapter marks of the individual movies by clicking on the “chapter list” icon at the top right corner.



6. Just like any movie, the playlist entry must have its “End Action” and “Popup menu” set to Jump to main menu.

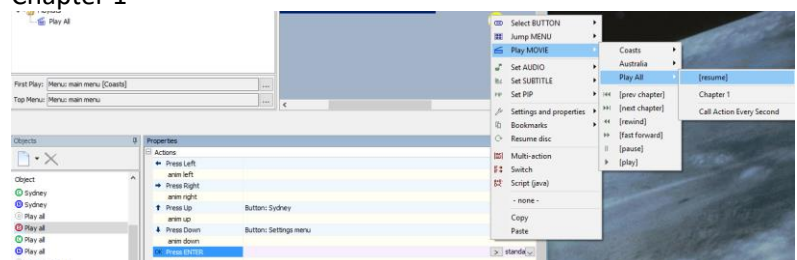
Step 4: Add items to the main menu

The playlist must be associated with a new button whose “Press Enter” action starts the playlist. We therefore need to modify the main menu by adding an item “Play All” with a button that activates the playlist in its “Press ENTER” action.



Photoshop menu

1. A modified version of “main menu.psd” is available in \Sources\Project objects\original sources\main menu 2.psd. Copy this file into the \MultiMenuBR\original sources folder.
2. If the main menu still exists in the menu branch (when you copied the previous project), delete it.
3. From the Project tree, import the menu (right mouse, “Import PSD”).
4. Merge the “normal” and “selected” state images into a single button and duplicate the button for each of the movies and playlist (see previous project on how to do this)
5. Set the navigation actions of the button next to “Play all” so that the viewer can navigate to and from it. The “Press ENTER” action must perform a “play movie> playlist> Chapter 1”



6. Do the same for the “Audio and subtitles setting” button. The navigation actions can be set. As long as we haven’t made the settings menu (next step) you cannot set its “Press Enter” action correctly. You can however set its action to “none” rather than the incorrect value the button copy still holds.

Menu in .png file

If you create the menu through .png files instead of importing a Photoshop file, the individual menu file can be copied from \Sources\Project objects\menus\main menu 2.png into the \MultiMenuBR\original sources project folder. Or you rename it to “main menu.png” you will see that the BDS Designer menu immediately shows the added menu items since it refers to “main menu.png” and that file got replaced (but kept its name). Buttons and actions must still be added (steps 3 – 5 above).

Menu made in BDS

If the menu list is a text object in the menu, select it and open it. Then add a new line that will contain the “Play All” text.

Also add two new buttons (through copy/paste of an existing one).

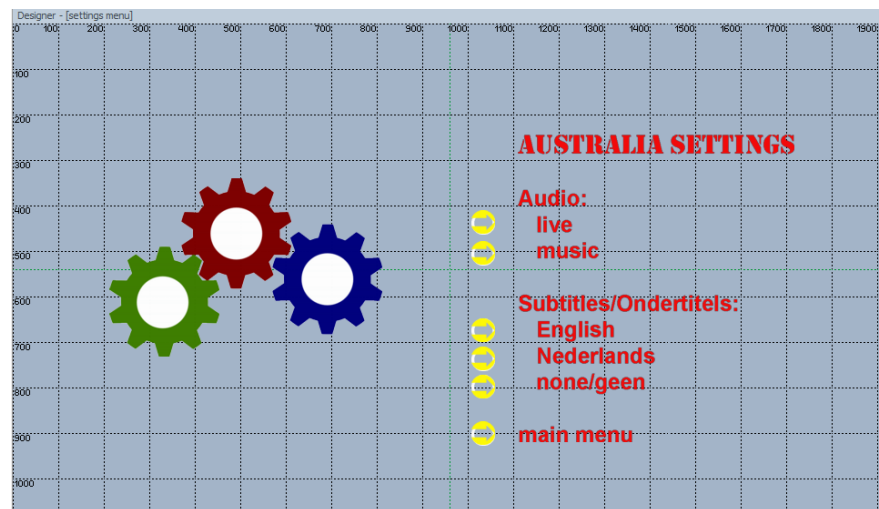
Make sure the navigation to/from the various buttons visits all buttons. The “play all” button can also be set to start the playlist (steps 3 – 5 above).

Step 5: Add or insert settings menu

Create Setup menu as set of background objects

Rather than using the remote-control buttons to select audio and subtitle track, we implement it as separate menu. The remote-control can still be used for this purpose if so desired.

The setup menu will consist of a static image and the text and will look like the figure below.



In the completed disc the viewer will see it with the menu movie.



It has a text block containing:

- Audio
 - live

- music
- Subtitles / Ondertitels
 - English
 - Nederlands
 - None / Geen
- Main Menu

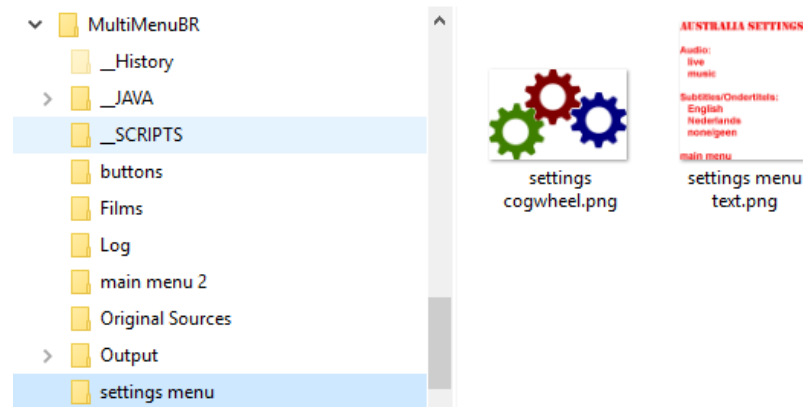
The menu text lines will get buttons to set the preference and the Main Menu item is needed to return to the main menu once our preferred setting has been set up.

The static image also shows a few cogwheels.

Setup menu as Photoshop file

You can create the entire menu in Photoshop and call it “settings menu.psd”.

You can also copy such a file from \Sources\Project objects\original sources\settings menu.psd and copy it to the project’s \MultiMenuBR\original sources folder.



Once the menu is made in Photoshop, you import it into BDS as a new menu.


Setup menu as set of .png files

If you create the menu using a separate .png file for each menu object, you can find these already made in \Sources\Project objects\menus and can copy these to the project’s \MultiMenuBR\original sources folder. The files needed are

- Settings cogwheel.png – picture of cogwheels
- Settings menu text.png – the menu items for settings

Take the following steps:

1. Right click on the “Menus” branch of the Project View Window
2. From the dropdown menu select “New Menu” and provide it with the name “Settings Menu”. This creates a blank menu without objects.
3. Open the Objects View. Select “Settings Menu” on the Project View. This will synchronize the Objects View for the menu. There are no objects yet

4. On the Objects View, click on “Add” (icon ). Specify the item you want to add as explained in “Menu through Objects window” on page 90.

In our case we add 2 static objects with an image:

- a. Object “settings cogwheel” based on image “settings cogwheel.png”.
- b. Object “setting menu text” based on image “setting menu text.png”

Setup menu through BDS



Menu and images are defined entirely as objects within BDS Designer window. It is partly identical to the previous method of adding .png files.

Take the following steps:

1. Use the Project Tree to create a new menu, called “Setting menu”.
2. Add a new static image object and assign it to cogwheel.png
3. Add a new text object “menu items” and enter all text for audio/subtitle/main menu. (the project must have some fonts defined through Project > Used Fonts).

Customizing the setup menu

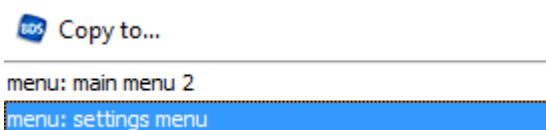
Once imported or created, the menu, like any menu, must be given a background movie. Specify the same one as for the main menu (\films\menu.264 and menu.ac3). Using the same menu movie allows transition between menus to let the background movie continue without interruption or restart.

Use the Designer window to position all picture layers where you want them. Make sure the objects are not locked through the menu icon  (unlock by clicking, changing the icon into ).



Add menu buttons


The new Settings menu has no buttons. We use the same buttons as on the main menu.

1. Select the main menu 2 in the Project Tree. The menu will show in the Designer window. Select a button and press CTRL/C (or right click and select “copy to...” from the dropdown menu). From the opened “Copy to...” window, select the new “Settings menu”

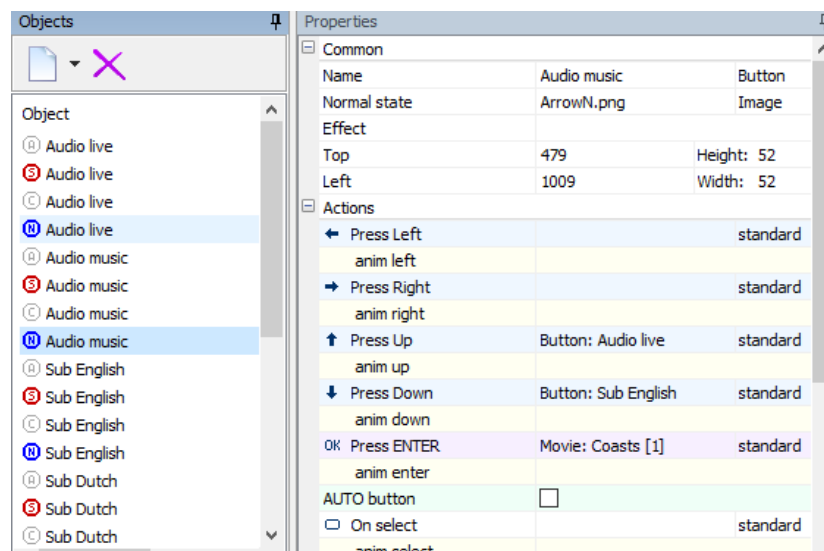


And presto: the button is copied at the same spot as it was on the main menu. Move the button to the proper position next to a menu text – preferably next to the last menu item. To ensure the entire button is moved, check that the “Move

entire button” is set as such ( - blue surrounded icon). To see other states, you may need to show them using the “Show” buttons ().

2. Make as many duplicates (by pressing CTRL/C) of the button as needed, this time on the same Settings Menu (preferably next to increasingly higher menu items).
3. Position them correctly aligned and rename them to proper names. If necessary, re-read “Step 4: Layout of the menus and duplicating buttons” on page 107.
4. Set the order of the buttons (in Objects) so that the first menu item and top button align and each lower button aligns with the next menu item. If you positioned the copies of the button starting from the bottom, this order is already correct.
5. Specify navigation between the buttons. The “Press Up” and “Press Down” actions in the button properties must point to the buttons above or below the button. Use the Auto-Assign button () to facilitate this.

The end result will be like the picture below. It shows the properties for the button next to menu item “music” : “Audio music”.

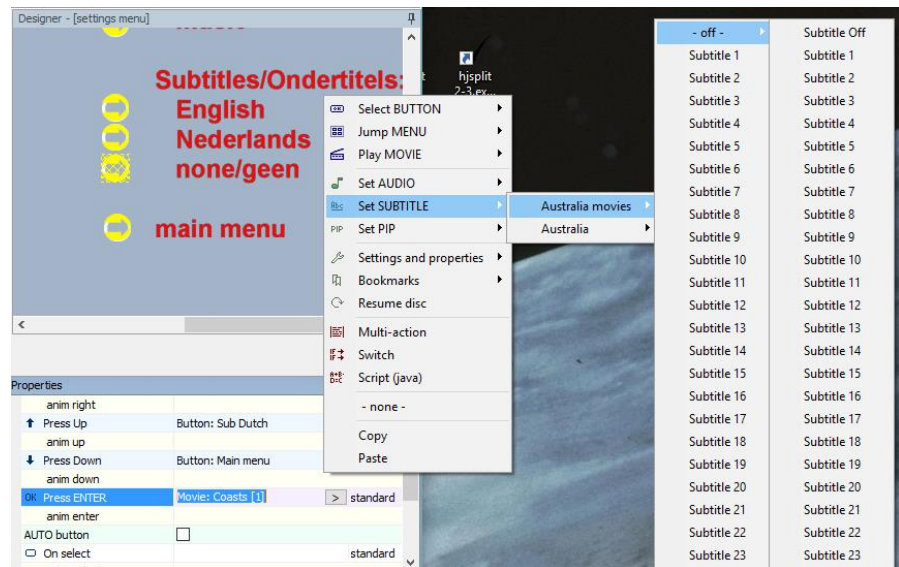


Add button actions to Setup Menu

So far the Setup Menu has been composed of picture objects and buttons, but no actions apart from navigation we have added to it. But BDS has given them Press Enter actions! The wrong ones. By copying a button, its properties are copied too and they point to objects that we don’t want to point at. We’ll correct that now.

1. The easiest modification is to modify a “Press Enter” action to the button next to the “main menu” menu item. Set its action to “Jump Menu” > “Main Menu” > “default” (or a specific button on the main menu).

The other modification requires the buttons to change the audio or subtitle settings when one is selected.



2. Select the “Press Enter” property of the button, then as action set “Set Subtitle” or “Set Audio” depending on what the button is supposed to set.
3. Next, select the group the movies belong to (here: “Australia movies”) and select the subtitle stream number to use or “Subtitle off”. The figure above shows the BDS menus for the button next to the menu item “subtitles – none”.
4. Repeat these steps for each button that modifies audio or subtitles.

Adding button state “Current” to prototype-button

The Settings Menu is now part of the project and sets audio and subtitles to the desired setting for all movies in the “Australia movies” group. But the viewer is not given a clue about what the previous selected settings are. This is where the “Current” state of the button comes in. “Current” here means “this is the setting as specified earlier by the viewer”.

For the current state, another button image is needed. We will use the same button image as for “selected” but coloured red instead of yellow.



This button state image can be found in \Sources\Project objects\buttons\ArrowC.png. Copy it to the project’s \buttons folder.

The use of “current” state only makes sense to most of the buttons on the Settings menu. The “main menu” button of course needs no “current” state – when selected, it makes you jump back to that menu.

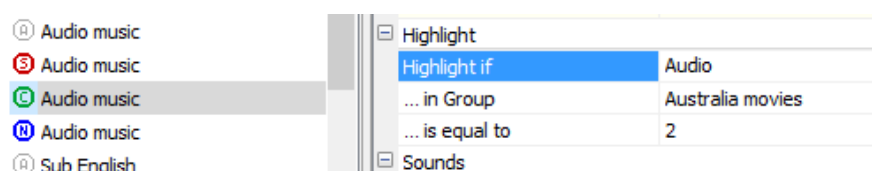
To make this button state visible, enable its green coloured “Show” button on the BDS menu bar ().

Show Current state during playback

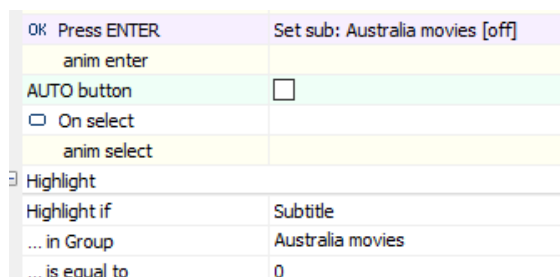
Once all eligible buttons have an image associated to their “current” state, this state must show when the bluray disc is played back.

As an example, if the audio track is set to “music” then a red button must show next to this menu option.

1. Open the button properties and find the “Highlight” segment. This segment specifies under what conditions the current state of the button must show.
2. For the audio track “music only” (the button named “Audio > music”) it highlights when the following conditions are met:
 - Movie belongs to the “Australia movies” group
 - It applies to the audio tracks
 - The audio track (stream) number must be equal to 2



3. All buttons that set an audio or subtitle track must be provided with the proper conditions in the Highlight section.
4. If the “Set sub(titles) > off” action is required for the button’s “Press Enter” action, its corresponding Highlight is a value of 0 for the group of “Australia movies”.



Step 6: Update the Main Menu to activate the “Audio and subtitles Settings” menu item

The main menu has already been updated with two menu items, “play all” and “Audio and subtitles setting”. The navigation between all buttons has already been setup and the “play all” button activates the “play all” playlist item.

The “Audio and subtitle setting” button has no “Press ENTER” action yet. When clicked it must open the Setting Menu. This menu is now available, so open the properties of the button and set its “Press ENTER” action to open the Settings menu.²²

²² It is not used here. But if a menu has an Action/Animation property you can open the menu not just by stating its name, but specify to first execute the action and/or animation before the menu opens.

Step 7: Add an intro movie

As last addition to the project, we add an intro movie to the main menu. In fact, any menu can have an intro – it will then start running the intro movie and delay the display of its menu objects. We will use an intro movie in this project that is identical to the normal menu movie so the transition from intro to regular menu movie goes unnoticeable.

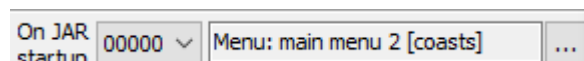
You can copy intro menu streams from \Sources\Project objects\movies\demuxed\menu.264 and menu.ac3 to the \films folder of the current project (e.g. J:\BDS Projects\MultiMenuBR\films).

The steps to take are:

1. Add another movie placeholders in Project Tree Movies list. Rename it to “Intro movie”
2. Specify the video and audio stream files for this intro movie: its location is the same as for the regular menu movie: \MultiMenuBR\films\menu.264 and menu.ac3
3. Double click the “into movie” to open its Scenes window. Add a second chapter about 4 seconds from the beginning of the movie. Close the Scenes window.
4. Adjust the “main menu 2” property “intro movie” to now use the “intro movie” movie placeholder

Step 8: Open the main menu at disc start

Specify the “main menu 2” (or “main menu”) as “On JAR startup” in the Project Tree with the Coasts button as pre-selected button



Step 9: Check the project

This step checks if all links are present and is no different from the description in section “Step 9: Check the project” on page 118.

Step 10: Run simulation

Once the entire menu structure has been set up, the simulation²³ will allow you to run through all scenarios and see if the links are set correctly. See description in Step 9: Check the project on page 118.

See the “menu simulation” section in the online help for a list of the mapping of keyboard keys onto remote-control buttons or refer to section Simulation on page 119.

If during simulation errors are detected (wrong navigation, no menu movie, no End Action or Popup action specified or other omissions) you can correct these first and start simulation again.

²³ the simulator only runs code in a single JAR. By default BDS only has a single JAR title, 00000, but as of V4.3 you may define multiple JAR files. The one simulated is the one currently set in the Project Tree window under “On JAR startup”.

A common error is forgetting that not only the new setup menu must be reachable from the main menu but you must be able to return there from the setup menu.

Step 11: Build the disc

This step is identical to section “Step 4: Build the disc” on page 64.). Make sure the Project > Project Settings > General tab points to the correct \Output folder (especially if you copied a project into a new folder).

Step 12: Burn the bluray disc

This step is identical to the one described in the previous chapter (section Step 6: Burn the bluray disc on page 71)

Project 4: Creating a bluray disc with popup menu

Project Goal

After creating a bluray disc with a regular main menu from which to select a movie, this project adds another type of menu: the popup menu. It literally “pops up” but only while a movie plays.

This project will

- Add a popup menu to a movie
- Use the timeline generator to generate a popup (menu) with a timeline

Popup menu

A popup menu is linked to one or more movies. It only shows when a movie play. The movie has a property that attaches a popup menu to it.

The popup menu only appears

- when the remote control button “popup” is pressed while a movie plays

The popup menu disappears

- when the “popup” button on the remote-control is pressed again.
- automatically if
 - the “Inactivity Timer” property time setting of the popup menu has expired
 - **and** the Inactivity Action is set to close the popup menu

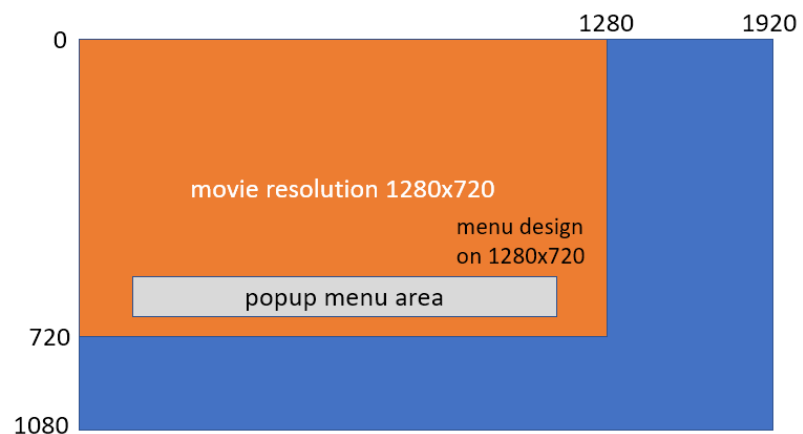
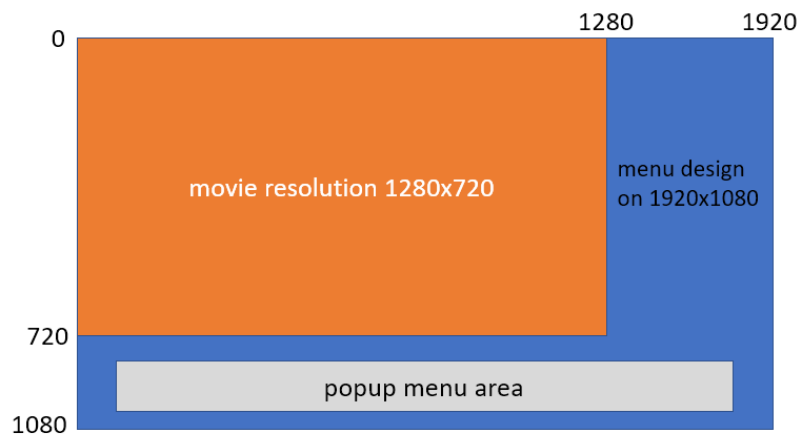
(Often this automatic closure is used with temporary menus that contain a message and must close after a certain period).

- explicitly if the popup menu has a button whose “Press ENTER” action is set to close the popup menu.

In all aspects, the popup menu is a menu like any regular menu. It has images, buttons and animation (discussed later) just like a regular menu. Only difference: a regular menu is shown without a movie running, a popup menu is shown only during a movie running. Usually also a popup menu has a limited size – like an area at the bottom of the movie screen.

Because it is linked to a movie, the popup menu must have the same size (resolution) as its movie. You easily forget this and it may take you hours to figure out why the popup menu does not show if its parts are positioned below pixelheight 720 in a movie that only has 720 pixels height.

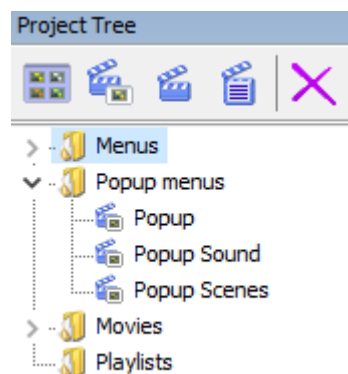
The two pictures below show how not to design a popup menu (remains invisible) and how to do it in case of a half-HD movie of resolution 1280 x 720 pixels.



Creating a popup menu

Each popup menu, like regular menu, must have at least one button that has a Press ENTER action for the viewer to activate.

A popup menu is created the same way as a normal menu. Because they are linked to movies, popup menus have their own branch in the project tree. You can copy menus between regular and popup menu branches in case you want to reuse some elements of a menu.



Each movie can have its own popup menu or several movies can share the same one. When they share a menu, they also share the actions enabled by the popup menu.

The name of the popup menu is specified as the “popup menu” property in the movie properties.

Start Action	
End Action	Menu: Main
Disabled actions	
Auto show popup	<input type="checkbox"/>
Scenes	5
Allow save state	<input checked="" type="checkbox"/>
Popup menu	Popup: Popup

Just like any menu can invoke another menu, each popup in turn can call additional popup menus, other menus or start other movies.

Popup menu timeout

Like a regular menu the display of a popup menu may be limited in time: there is an Inactivity Timer (set in seconds) and what to do next: the Inactivity Action. This may help to show a popup but also make it disappear when no viewer action is taken.

Popup menu wizards

BDS also provides several wizards that create popup menus for movies:

- Carousel wizard for chapters (allowing you to jump to any chapter in a movie (explained in Carousel menus on page157)
- Timeline wizard to show the progress in running the movie sofar. Several set top players have such a timeline themselves, but BDS can also provide one now easily (or you may use Java programming to make your own – see Project 13: Popup Timeline with bookmarks on page 332)

D.I.Y. Adventure

The first project to implement a popup menu is a simple one: each movie can open the same popup menu with a single button “Close” that will close the popup. Any other button with any other function can be used too – this project continues on the previous project and adds a popup menu. The popup menu will be made within BDS so we need to load some fonts.

To venture out on your own, keep in mind:

- Start from the previous project MultiMenuBR and copy its project folder to a new project PopupMenuBR
- Create a popup menu for both movies. The menu only has a “Close” button that closes the popup
- Assign popup menu to both movies (and if you like, add a popup to a playlist item)

Step 0: Ready to run

If you don’t want to create this project yourself, you can setup the project from provided sources following these steps:

1. Copy the folder tree \Projects\PopupMenuBR to your BDS projects folder (e.g. J:\BDS Projects\PopupMenuBR)

2. Copy the movie files Australia, Coasts and Menu from \Sources\movies\demuxed to the project's \films folder (e.g. J:\BDS Projects\PopupMenuBR\films)
3. Copy all subtitles files from \Sources\movies\subtitles to the project's \films folder
4. Create a \fonts folder and copy the file BDS Kit\Sources\BDS fonts\arial.ttf into it
5. Double click on the PopupMenuBR.bdmd file in \PopupMenuBR to start BDS and open the project
6. Click on the mux button and create the disc image. This will be created in the project folder \output popup
7. Experiment

To experiment with the timeline generated by the wizard, open the timeline.bdmd project file in the project folder.

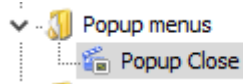
Step 1: Get organized

The new project is called PopmenuBR. Either you create this project entirely from scratch or you copy the previous project to continue from there.

1. Create a new project folder PopupMenuBR in your BDS projects tree (e.g. J:\BDS Projects\PopupMenuBR)
2. Copy the entire project folder tree of the previous project from the \BDS Projects\MultiMenuBR project. Rename the top project folder to \PopupMenuBR.
3. Delete the \output folder
4. Create an additional folder \fonts and copy the font file Arial.ttf from the BDS Kit\Sources\BDS fonts folder.


Step 2: Add popup menu

1. Open the PopupMenuBR.bdmd file. Although it was renamed, it is currently identical to MultiMenuBR of the previous project.
2. Use Projects > Used fonts menu option to add the arial.ttf font file in \fonts folder to be used by this project.
3. Add a popup menu in the Project Tree branch "popup menus" and call the new menu "popup Close"

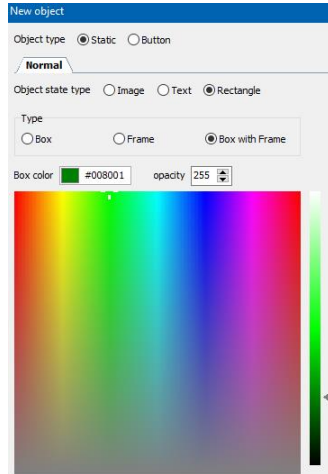


4. Select this popup menu so that the Designer Window shows a blank menu canvas. Because the movies Australia and Coasts are in full HD, we can use the entire area of 1920x1080 pixels to design our popup menu.
5. In the Objects menu, create a rectangle object. By clicking on

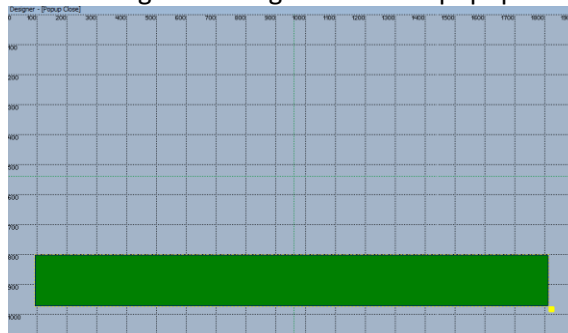


the "new object" button () a window opens where you select the "Rectangle" radio button. Select a green colour for the "box color" box. Use the slider at the right hand side to determine the exact colour to choose. Select either the radio button for "Box"

or “Box with frame”. Click OK when done.

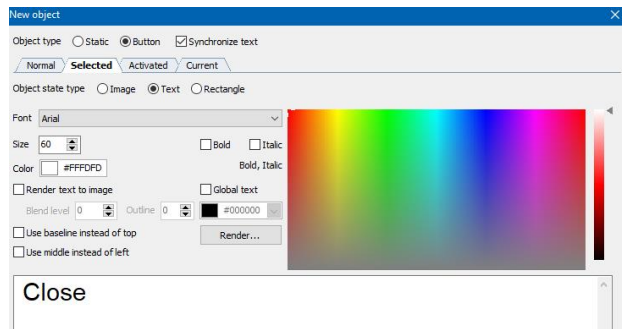





6. This results in a green rectangle at the top left hand corner. Drag it to a position around 800 vertical and stretch it (use the yellow square at the bottom righthand corner) as a bar about 150 pixels thick to cover the horizontal area around 100-1820 . This will be the green background of the popup menu.



7. Call the rectangle item “popup background”
8. Add a button with the text “Close” and position it in the center of the green rectangle:
 1. Click the arrow next to the “new object” button²⁴ to reveal the dropdown menu and select “New text button”.
 2. This opens the button text window. Ensure the “synchronize text” checkbox is checked. Enter the text “Close” . Visit the different states (Normal and Selected) to ensure the same text is rendered in different colours (e.g. “Normal” in red and “Selected” in white – as long as it is clear what the selected state looks like).

²⁴ Note that if you later want to change the appearance or text of the button, you need to select it and via right click select “Change object states”. Only this way any change made in one button state is also applied on the other states..

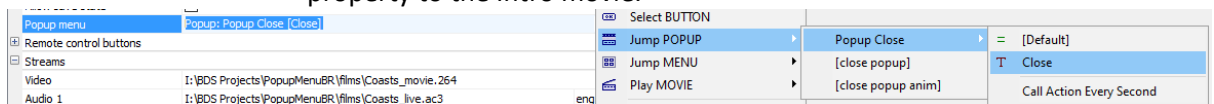


3. The button is added in the lefthand top corner. Drag it to the middle of the green popup menu background. For this, you may need to:
 - a. Enable showing button selected states (click on BDS menu button )
 - b. Disable locking menu objects (click on  to change it into )

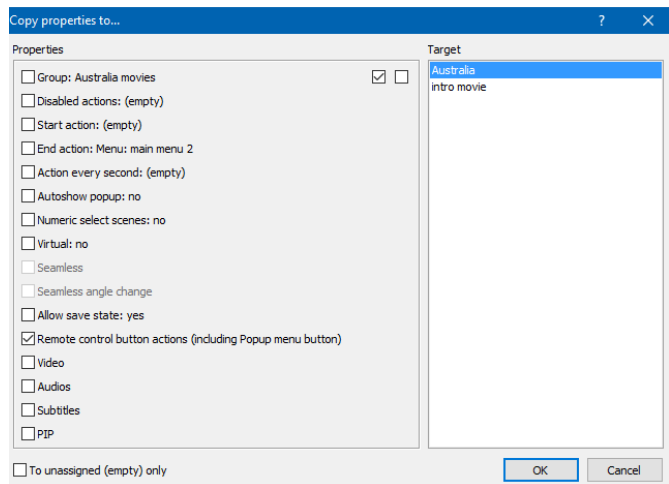
The end result should look like this:



4. Rename the button to "Close"
5. In the button properties add "Press Enter" action "Jump Popup" > [close popup]
9. Associate popup withh movie Coasts. Set the movie/playlist property "Popup menu" to the "Popup Close" menu and its initial (but here: only) button "Close". Do not assign this property to the Intro movie.



10. Repeat the previous step also for the movie "Australia" and the playlist "play all". You can do this manually but if some properties are identical between movies, the BDS option "Clone properties to all..." allows you to select what properties to copy to what movies. In a similar manner you can clone properties between playlists (but unfortunately not both). So to clone the popup menu you can do:
 1. Right click on Coasts movie and select "Clone properties to all..."
 2. From the menu, select all movies (here: only Australia) and check all properties to clone (here: only popup menu that is part of the Remote Control Buttons)
 3. Click OK



11. Ensure both movies “Coasts”, “Australia” and playlist “Play All” have “End Action” property set to “Menu 2 > Default”

Step 3: Run simulation

Check if the modification has the desired result.

1. Start the simulator
2. Select either Coast / Sydney movie to run
3. Press on F5 to produce the popup
4. It should show the green popup menu with selected “Close” button.
5. Press <Enter> as if Close was selected. The popup should close
6. Press “End Movie” to return to the main menu
7. When done, save the project (File > Save)

Step 4: Build the bluray disc

This step is identical to section “Step 4: Build the disc” on page 64.). Make sure the Project > Project Settings > General tab points to the correct \Output folder (especially if you copied a project into a new folder).

Step 5: Burn the bluray disc

This step is identical to the one described in the previous chapter (section Step 6: Burn the bluray disc on page 71)

Using the timeline wizard

Now that a simple popup bluray has been created, we extend the project a bit further by adding a timeline popup. This will show the progress of the running of the movie.

BDS provides a wizard for this. It requires a popup with 3 fields:

- A timeline (a horizontal bar that extends from 0 % to 100% during the movie’s playback)

- A text field that contains the exact running time so far (like 1:14:12 for 1 hour 14 minutes 12 seconds)
- A button (as any menu must have at least 1 button) – usually to close the menu.

The use of the wizard is explained in the steps to take.

Step 6: Add new project

We want to keep the current project but add something to it using a different project file. Therefore, in the \BDS Projects\PopupMenuBR folder you see project file PopupMernuBR.bdmd. Copy this file and paste it in the same folder but name it “project timeline.bdmd”.²⁵

Step 7: Add a timeline popup menu

Rather than starting from scratch we clone the existing “Popup Close” as new popup with a new name “Popup Timeline”.

(For this: right click on “Popup Close”, Select “Clone Tree Item” and give it a new name).

Step 8: Enhance popup Timeline

1. Select the popup Timeline. This will display the familiar “popup close” clone.
2. Move the “Close” button to the right side. If necessary, make the font size a bit smaller so it fit nicely at the end of the green menu rectangle.
3. Add a new object by clicking on the arrow next to the “new object” button in the Objects window and select “New rectangle”. Repeat the actions you took to create the green background. Now make the rectangle yellow and thinner and a bit shorter than the green background. Position it on top of the green background. In the rectangle properties, rename it from “Item 3” into “Timeline”.
4. Add a “new text” object using the Arial font available. Text colour white, smaller font size and with text “hh:mm:ss” (the text isn’t relevant – the wizard will overwrite it). Position the textbox somewhere within the green popup menu area. Rename the text box to “Timer”
5. The final Popup Timeline should look similar to this:



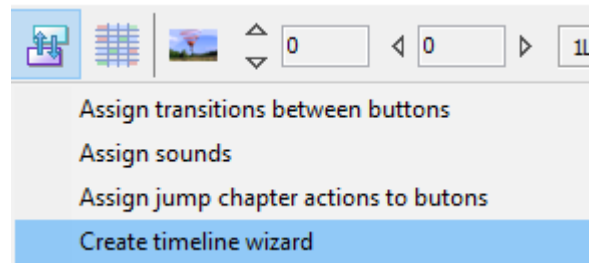
Step 9: Use the timeline wizard

Now that the popup has been made, execution code must be added to the “timeline” bar so it will grow while the movie runs to its end and the “Timer” box to display the current running time. You can program

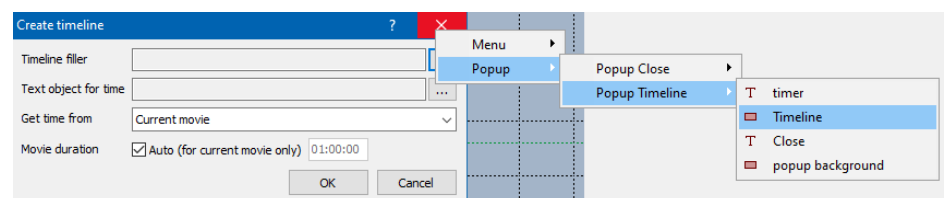
²⁵ When the project is still open after step 5, you can first save it (File > Save) and then use “Save As” to save it again but under a different name like “project timeline”

this yourself (and needed to before BDS V4.4) but the wizard can do it for you. The popup menu with timeline can be made as fancy as you like – for this project we keep it simple with a green popup menu and yellow bar.

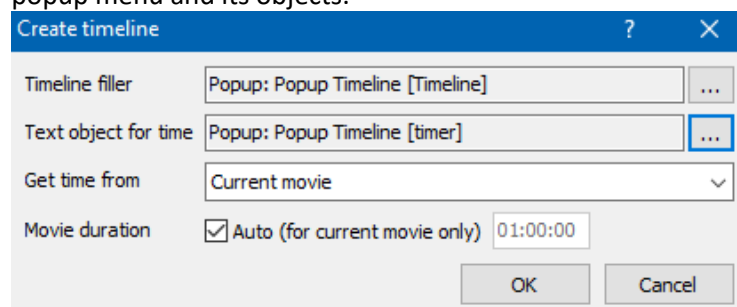
1. To invoke the timeline wizard, click on the BDS button “Auto assignments and wizards” and select the “Create timeline wizard” option.



2. Fill in the two items it needs to know:
 - a. The “Filler” is the extending progress bar – the yellow “Timeline” object in our case
 - b. The “Text object for time” is the “timer” object
 - c. Both items are part of the “Popup Timeline” menu and you should select those:



3. In the end the wizard is passed the correct information on popup menu and its objects.



4. Because most timelines are associated with movies, the “Get time from” is left to its default choice of “Current movie”. That way the same timeline popup can be used for all movies in the project. (other options are selection of a specific movie or playlist).
5. Click OK to let the wizard adjust the project properties: the Popup Timeline gets a “Action Every Second” set to a multi-action (discussed later in Multi-action on page 203) that instructs the player to:
 - a. Check the running time each second
 - b. Adjust the timeline bar accordingly as percentage of total length

- c. Display the time in the time object

Step 10: Connect timeline popup with movie

Because the timeline is a popup menu it must be attached to a movie. In our case we attach it to each of the movies Coasts and Australia but also to the Play All playlist.

1. You may wish to add the “Press Down” property of the Close button to also close the popup menu (Jump Popup > Popup Timeline > [close popup]) That way either pressing on the “Close” selection or simply pressing the down arrow on the remote control both make the popup menu disappear.
2. For each of the movies, change its “Popup Menu” property to “Jump Popup > Popup Timeline > Close” (which selects the only button on the menu: Close)

Alternatively, you may leave the “popup menu” to open whatever popup menu you made (like the “popup close” we made earlier) and use one of the special buttons on the remote control to open the timeline. All buttons that can be selected (one or several) are listed as movie property in the section “remote control buttons” .

Let’s configure the project in such a way that the Red button opens the timeline popup. The Close text button on the menu (and optionally the down arrow on the remote control) will close the timeline popup.

3. Select from the movie “Remote Control Buttons” the “Red” button
4. Set its action to Jump Popup > Popup Timeline > Close

These steps must be repeated for all movies and playlists for which you want to show a timeline. For this you can use the “Clone properties to all...” option of BDS from the context menu if you right click on the Coasts movie.

Step 11: Simulate project

Use the BDS simulator to see if the timeline menu works as intended. Note however that because no movie actually plays, the popup menu for the timeline shows the entire timeline (as if 100% completed). Only the building of the disc and playing the result with a player will show the popup timeline grow.

If you programmed the red button, in simulation this is the F1 button on your keyboard. (see Simulation on page 119 for all key assignments during simulation).

Project 5: A Setting menu as popup and a chapter menu as carousel popup

The project goal

In previous projects you created a bluray disc with a menu from which to select what movie (or playlist) to play; another menu to set your choice of audio and subtitle track, and you designed a popup menu with a timeline through its wizard.

Many discs have such a “Setup” menu that can be chosen, but the same choices can also be made or modified whilst a movie is running. A menu displayed while a movie runs means it is a popup menu. But it can offer the same set of choices a regular menu provides.

There is another wizard that generates a popup menu: on that shows chapter images to allow the viewer to jump to a particular menu.

An example (from the BDS examples on their website) is a popup that allows to jump to another popup menu to setup sounds and subtitles. It also allows to jump to a popup menu with scene selections.



This project will let you make:

- The initial popup menu shows a choice of additional popup menus: one for audio/subtitle settings and one to select a chapter of the movie from
- A popup menu called “Settings” menu with buttons to select what audio or subtitle track to use – basically the same as what the ordinary Settings menu we made earlier also allows.
- A popup menu showing chapter images. Each image is a button to jump to the corresponding chapter in the movie. If there are more chapters than fit on the screen, the chapter images rotate in a “carousel” fashion.

We are going to change the already made simple popup menu with just a “Close” button to cater for all settings of audio and subtitle streams.

The carousel (popup) menu for the chapter marks will be generated by wizard (there are two such wizards – each producing different chapter popup menus).

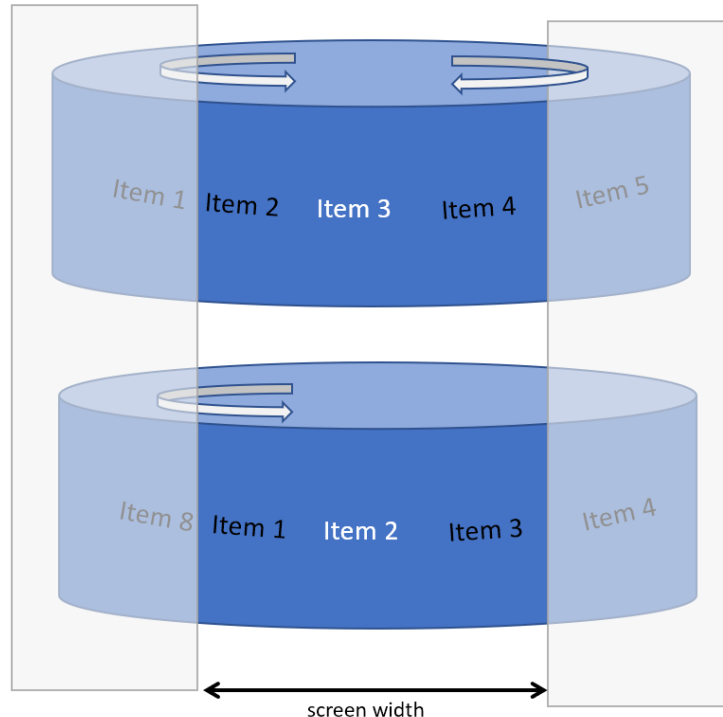
If you discover that chapters always restart a title from chapter 1, see the note made in Important for tsMuxer users on page 55,

If you want to create this project yourself, the source files can be found in the BDS kits.zip file mentioned in Appendix A: The user's guide source files.

Carousel menus

A carousel menu introduces animation in a menu. Any menu can be made into a carousel: regular and popup menus. We will employ a wizard that does all the animation for you. For any other animation that you need to define yourself, we'll discuss it at length in Part 4: Animations on page 247.

There can be many items (buttons) on the menu to select from. But because of their number, not all are always visible at the same time. In a carousel menu it looks like all buttons are on the surface of a rotating cylinder and only a few are shown on screen. The rest is "hidden". A single menu button is (pre)selected. The cylinder can rotate either way – in each case displaying other buttons on screen. Some become visible, others disappear. The movement of the buttons look like horses on a children's fun fare carousel.



The figure above first has the button for "Item 3" selected in the middle (out of the 8 available menu buttons). Using the left arrow key on the remote control the menu rotates and puts the button for "Item

2" as selected button in the middle. It looks like the cylinder has rotated anti-clockwise to the right (but you pressed the "left arrow" key on the remote control).

BDS provides a wizard to create this type of carousel menus. The rotation can be horizontal (as in the figure) or vertical (like the common volume counters at fuel stations or clocks with digits).

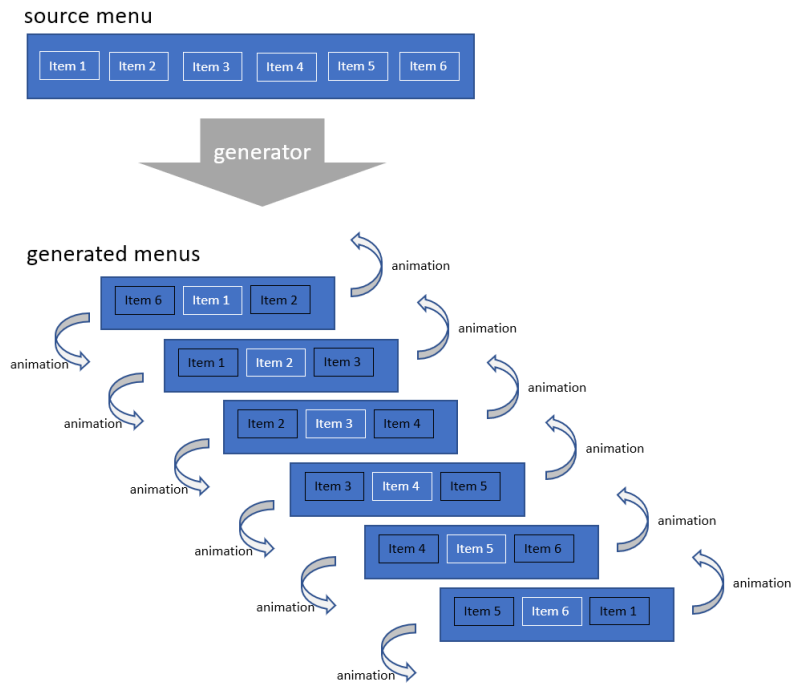
The basis for a carousel is a "source" menu that has all menu buttons defined. They all have the "Press ENTER" property set to some action like showing a menu or playing a movie. No navigation needs to be set – the wizard will do that.

The wizard then generates many menus based on that single source menu. Each generated menu looks like a "snapshot" of the source menu that has been rotated over one button making the next one the selected one. That selected button is also the only button on the generated menu. All the other buttons have become plain images using their "normal" state image. This is shown in the illustration below. All white buttons are real buttons. All black buttons are simply images (as if a button in "normal" state).

Rotating to another button effectively implies replacing one menu with another. That other menu has the other button as selected button. The former button changes to a static image.

For example a menu with button "Item 3" is replaced by another similar looking menu that has "Item 2" as button. Like a cartoon movie where each frame is slightly different, a carousel menu simulates movement by showing different menus for different buttons but add some animation to make it look like a smooth transition within the same menu appearance.

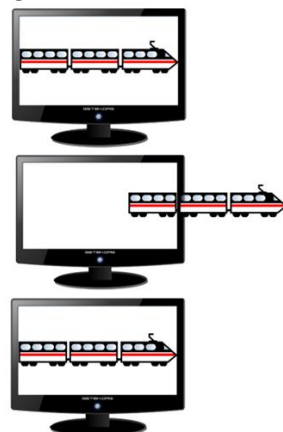
The generator takes care of the navigation on a menu and its replacement with another menu and the animation that takes place in between.



The source menu is never used on the final bluray disc but it is kept just in case you need to change the carousel appearance. You then modify the source menu, delete the older generated menus and re-generate the menus again. Of course if you made many changes to the generated menus these are all lost – so better make sure the source menu is OK and unlikely to need changes in future.

BDS allows several options to make a carousel menu:

- movement: the menu moves horizontally or vertically
- selected button: may appear in the middle of row or column or at the left/top. Centered in the middle is only useful if all buttons fit on the screen
- rotation speed: from one menu with a button to the other. You can specify how many “frames” a menu transition lasts. There are usually 24 frames in a second (but you can set this value)
- type of carousel: circular fashion (like a roundabout) or like a train departing. Once completely gone, its head part is shown again.



This carousel concept was first introduced to chapter menus but has been expanded to include normal menus and popup menus too from BDS V4.2.2 onwards. Online help is found under “Carousel menu generator” and a video tutorial is found under <https://www.youtube.com/watch?v=2Te7w2l3Zel>.

The generation of carousel menus is subjected to a few conditions:

- All buttons on the source menu take part in the carousel
- The order in which the buttons are lined up is the order (first to last) of the button objects in the Objects window of the source menu (which may be different from the ordering in the Designer’s Window)
- All buttons are lined up in a row or column. The length of this row or column must be less than the “virtual size” of the screen. In HD resolution, the physical screen length is 1920 pixels. With 15 buttons each 200 pixels in size, a horizontal carousel requires a virtual screen of minimal $15 \times 200 = 3000$ pixels. Plus some separation between the button That means only a few ($1920/200 = 9$ or less) buttons can be shown on screen at any time. The virtual screen size must be larger than (number of buttons) x (size of 1 button + 1 separation distance)
- All generated menus share the same menu movie (best specified in the source menu)
- No menu (i.e. the source menu) should have an intro movie. (A practical consideration when the starting menu is part of a carousel menu set: the first menu to be shown may have an intro movie that delays the buttons to be shown. None of the other generated menus should have one).

To create a carousel menu, you need to start with designing an ordinary, source, menu with all its buttons and surrounding static fields. Never throw away this source menu. If you decide to make some changes to the menu or button images or texts, the carousel must be re-generated from this source menu.

You do not specify any navigation properties for the buttons: these are generated by the carousel wizard.

All buttons must fit on the virtual screen size²⁶. As given before, 15 buttons with a width of 200 pixels set in a row requires a screen of at least $15 \times 200 = 3000$ pixels. Add separation between buttons to this. The real Designer Window allows for an HD screen of 1920x1080 pixels. So the calculated minimum 3000 pixels is definitely too much.

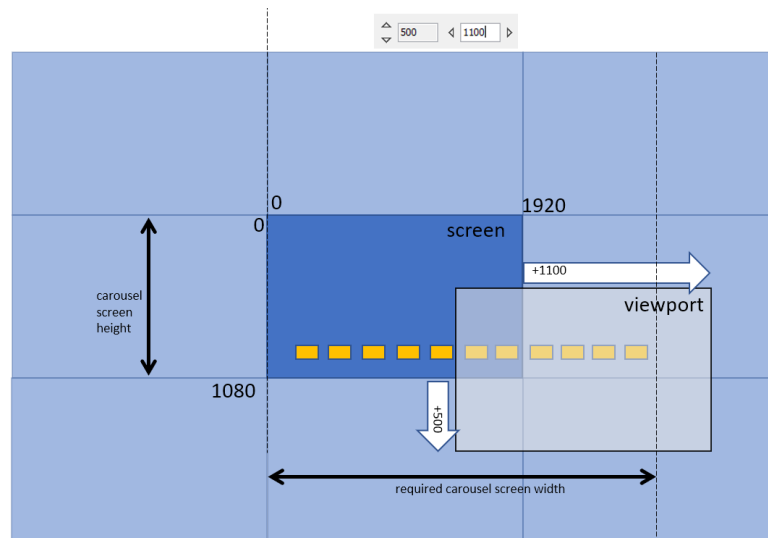
Enter the “viewports” buttons. These are two menu boxes on the BDS screen when you have the Designer Window open.

²⁶ You may squeeze all buttons on the physical screen – even if they overlap each other. The wizard will put them in a row and equally divides them over the specified screen width or height.



For normal design, the two values are set to 0 and what you see on screen are the 1920x1080 pixels you can use to design a menu.

However, clicking on the arrows next to the viewport box, you can shift the view to positions outside the visible screen. That's useful to position those "out of view" buttons. .



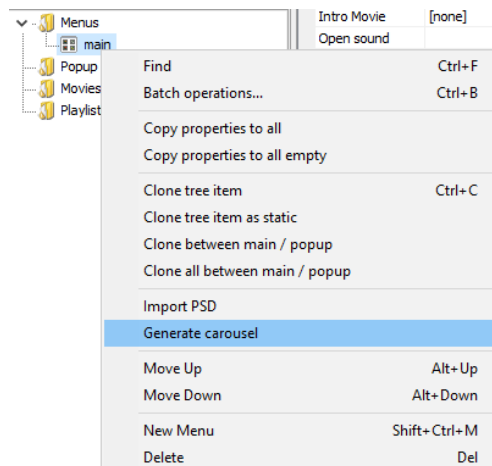
This is illustrated above. A set of yellow buttons meant for a carousel doesn't fit on the (dark blue) HD screen. By shifting your view vertically downwards (+500) and horizontally rightwards (+1100) you see the (grey) Viewport .part of the virtual screen. There you can position those buttons. When you start the carousel generator it will ask you what size the virtual screen needs to be to fit all buttons (and separation distance between them). In this example, the height remains the 1080 pixels of an HD screen, the width is extended to accommodate all buttons.

To ensure all buttons align properly in their horizontal or vertical list, use the "Align" menu option on all buttons. Align them at their left or top sides. If you don't, the button list will crooked with the buttons apparently a little dis-aligned.

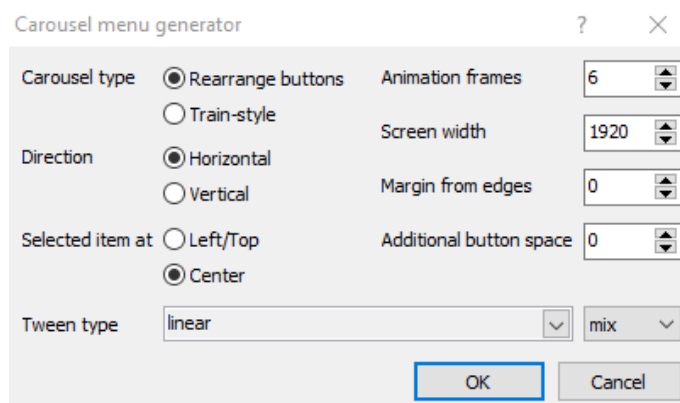
As an example: in a horizontal carousel all buttons should be aligned with the bottoms or tops (like in the picture above). Not arranged like the image below. If you would do that, you get a horizontal carousel but with buttons that jump up and down between generated menus.



A carousel generator is invoked through the context menu that shows if you right-mouse click on the source menu in the Project tree and select "Generate carousel".



It opens the carousel specification window. Here some already discussed possibilities must be selected. Once selected, click OK.



For each of the buttons, a menu is generated in which it is the only button– all other buttons have been converted into images. These images also count against the total pixel size of 7 900 000 pixels that is allowed for any one project.²⁷

The meaning of each of the choices made in the generation wizard window is explained below.

Carousel type – circular or linear

“Rearrange buttons” means that all buttons form a circular ring where the last button is followed by the first button as if all buttons are positioned on a cylinder.

Trainstyle means buttons are shifted until the last button disappears. Only then the first few buttons are shown again.

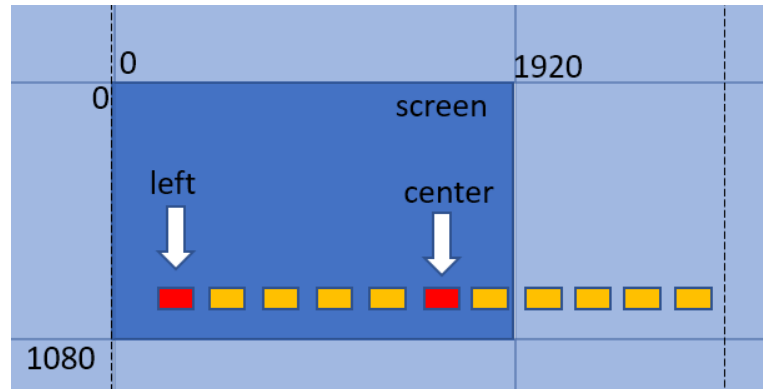
Carousel direction – horizontal or vertical

This simply specifies if all buttons are ordered in a row horizontally or in a column vertically.

²⁷ If you exceed this number, some menus may have to be stored in a different JAR file (see Using multiple JAR titles on page 197)

Carousel – selected item

The only button on the generated screen is in the middle of the virtual screen or the left/top of the visible part of the carousel. Better not use the middle position if you have an extended virtual screen. The middle of that screen may well be far off the physical screen the viewer sees and any button on it remains invisible.



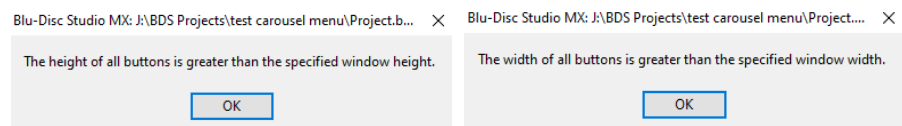
Carousel – animation speed

The speed of the sliding of the menu is specified by the number of frames the sliding takes. In Project Properties > Advanced you can set the number of frames per second. Usually this is the speed of the movies themselves – typically 23.97 fps or 24 fps. Hence 6 frames means the animation lasts for 6/23.97 or 6/24 seconds.

Carousel – screen width or height

The width or height of the menu is also specified. When you create the source menu, all buttons (plus any separation between them) must fit within the virtual screen width or height. They are all present and visible on source menu. The wizard will divide them over the specified width or height with equal separation between them. If the separation is 0 or less, the width or height is insufficient and buttons overlap. In that case you must increase the setting of the virtual screen width (or height).

If the size is specified as too small an error message occurs: either height (vertical carousel) or width (horizontal carousel) is too small:



If this error occurs, you need to either

- reduce the number of buttons,
- reduce their size
- reduce the button space (separation, 0 = BDS decides)
- reduce the margins of the border where no buttons are shown.

Or simply enlarge the window size. Any measure will do as long as all buttons fit in a row (or column) smaller than the screen width (or height).

Margin from edges

The value specified for margins from edges applies to the distance the leftmost and rightmost button keep from the virtual screen edge. For the left or top button this works. But if the virtual screen width/height is larger than the physical size, the margin is applied off screen to the virtual edge. As such, buttons might be cropped by the right side or bottom side of the screen.

Additional button space

Given the screen width or height, all buttons are divided over this length with equal separation. By specifying additional button space, the separation between each is enlarged by the specified amount. Of course the width of the buttons plus additional space must be larger than the virtual length. When it is not, you will get an error stating the screen width is too small.

A carousel mixed with animation

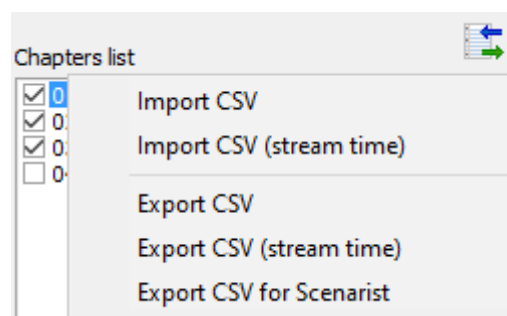
An example of a carousel generated menu where several generated menus are the basis for showing submenus through animation can be found in the section Animation with carousel menus (revisited) on page 267.

Chapter marks import/export

Movies are often divided into chapters and the viewer can decide to start a movie from a particular chapter point rather than from the start of the movie. When you re-use a movie you imported from another bluray disc, you can reuse the chapters defined for such movie.

Export chapters to another BDS project

When you have created your own chapter marks in a movie, their setting can be exported using the Import/Export chapters button in the Scenes window.



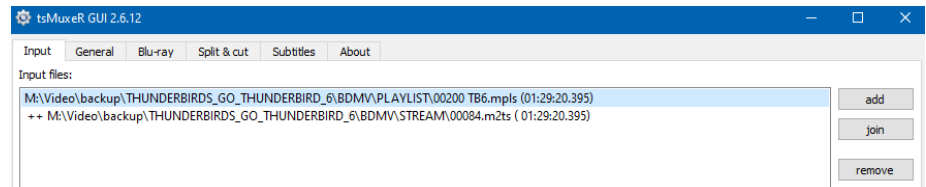
The import/export files are in a comma-separated-values format (CSV).

In the same way you can import them into another BDS project using the same movie.

Import chapters from bluray movies

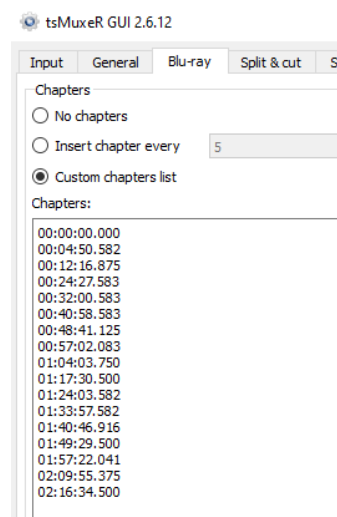
When you reuse a movie from a bluray disc (in its \BDMV\STREAM folder), the chapter marks are stored in the \BDMV\PLAYLIST folder in a playlist .mpls file.

To know what .m2ts stream file(s) belong to the .mpls playlist, the tsMuxer GUI shows these links.



The picture above shows an opened 00000.mpls file that describes the 00084.m2ts stream file. . The same is true if you open an .mkv file with chapters in tsMuxer²⁸. Because most playlist files have the same size (1 KB) it is trial-and-error to find the playlist that belongs to the movie you copied.

By opening the .mpls file using tsMuxer through its GUI, the chapter marks of the movie are made visible in the “Bluray” tab. You can select and copy/paste these chapter timings (in hh:mm:ss.hhh format) into a text file that can subsequently be imported into the BDS project’s Scenes window of the movie (Its menu allows for CSV files, but TXT files are also allowed).



Import chapters in play lists from movies

Play lists are discussed in section “Play lists” on page 125. A play list consists of one or more movies. Each movie can have its own set of chapter marks. Opening a playlist allows you to specify the chapters of this playlist. The section on play lists describes how to import those chapter marks.

²⁸ these files must be decrypted, i.e. probably won’t work when copied direct from the bluray disc

If the playlist consists of movies imported from other discs, you first define the chapters for those movies before you import those chapters into the playlist scenes window.


Chapter popup menu generation through wizards

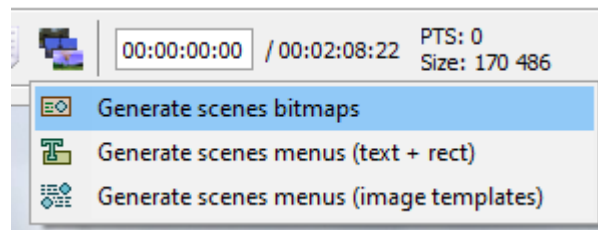
Chapters (or play marks) can be defined for each movie. You can navigate from chapter to chapter using the remove control buttons for “next chapter” (>|) or “previous chapter” (|<).

You can also define your own chapter menu for a movie that lists all chapters and a button has an “Press ENTER” property that jumps to that chapter in the movie.

Apart from all this manual work, BDS provides two chapter menu wizards that automatically create a set of menus that have thumbnail images that work as a button and will jump to the specific chapter. These chapter wizards were the first (horizontal) carousel wizards in BDS.

The two menu generation wizards are started when the Scenes window of a movie is opened and chapters are defined for this movie.

When the window is open, click the button “Generate scenes” () or press Ctrl/S. This will provide three options, the final two being the wizards:



- Generate scene bitmaps just generates the bitmap images (type .bmp) that can be used to create the thumbnail images (in .png format). It does not invoke a chapter generator.
- Generate scenes menus (text + rect) – this wizard creates a strip of chapter thumbnails in horizontal or vertical fashion. The strip can move as a carousel. The selected chapter image is always in the middle and the chapter images move one position to the left or right, giving a carousel feeling. Animating this movement enhances the illusion.
- Generate scenes menus (image templates) – this wizard creates a strip of chapter thumbnails with “next” or “previous” buttons to open the next or previous set of chapters. The selected chapter moves with the cursor from left to right. An initial version of this wizard is also part of the now obsolete (=not maintained) “BDS Lite” edition.

The general principle of both wizards is the same:

1. Generate the chapter images
2. position the images on one or more “chapter menus”

3. set navigation properties between images and “Press ENTER” when clicked to move to the movie chapter
4. link all chapter menus to allow jumping from one to other
5. Add a link from a popup menu button to one of the chapter images


Of course you can do all this yourself but you may have to create a lot of chapter menus and define a lot of links and actions. The wizard is slightly more rigid but an enormous time saver (think in hours and days).

Chapter menu wizards: Create your own bitmaps

In the sections below we will use the bitmap generator and chapter menu wizards to speed things along. One problem here is that the bitmaps taken for each chapter are taken based on some algorithm that picks an image “somewhere around” the chapter mark.

In some situations you may want to have full control over the images to use. For example, a chapter running from 0:00:04 upto 0:00:12 minutes may have the perfect image to represent it at 0:00:09 minutes. The BDS bitmap generator will always pick one closer to the 4 minutes mark.

To be in full control, create the bitmap images and the reduced chapter images (thumbnails that fit the menus) yourself. Then when you use the chapter wizards later on, instruct it to use your images and skip the bitmap creation that would overwrite them.

Create full sized images of a movie through a screen-capture application while running the movie. Within the Scenes window, BDS offers this option through its “Save current bitmap” button () or press CTRL/S.

Store those captured images in a movie-specific folder of the project, such as `\<projectname>\<moviename>ChapterImages\Bitmaps`

Create thumbnail images (that must have the .png format to use in BDS) in the same folder tree, one level higher, such as `\<projectname>\<moviename>ChapterImages .`

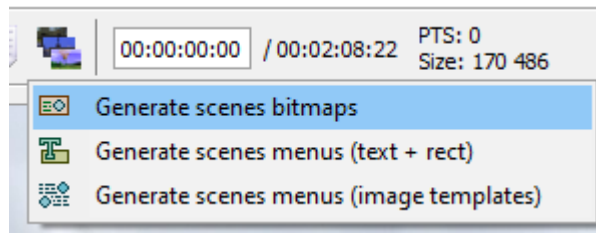
The thumbnails must be reduced to 273x151 pixels (or similar size, keep the aspect ratio 16:9). Use any image editor to do this.

They also have to have specific names consisting of double digits if you intend to use the chapter menu wizard later: `<sequence_number>.png`. Such as 01.png, 02.png, upto 99.png. You must start with 01 for the first chapter image, 02 for the second and so on.

Now simply start the wizard of your choice, using your images rather than generate them.


Generate scenes bitmaps (for both wizards)

Thumbnail images that represent chapters must be created as first step. Both chapter wizards generate these bitmaps as first step. Skip this step if you already generated your own images.

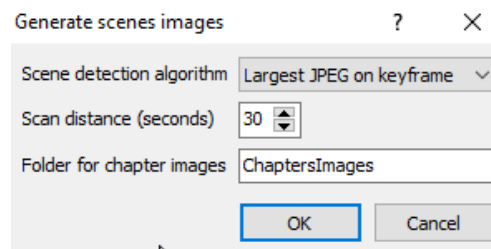


The bitmaps and thumbnails are saved in a special folder that must not be used for anything else. If you have several movies, each movie must use its own unique thumbnail folder.

To create chapter thumbnail images, the movie must have chapter marks (or play marks) defined. See section “Step 3: Creating chapters” on page 63 on how to do this.

To generate chapter images, open the “Scenes” window (if it isn’t already open after you defined the chapter points)²⁹ and click the button “Generate scenes” ().

Selecting the first option “Generate Scenes Bitmaps”, the bitmaps are created from the start of each chapter mark.



The image selected for the chapter start is taken at the set chapter marks or within the “scan distance” to that position. By default it is set to 30 seconds. You could narrow it down to say 5 seconds. (the image must be a complete image, an I-frame in the movie hence the distance from the chapter setting as that may not be an I-frame).

All thumbnails are stored in the project folder under subfolder with the name specified in “Folder for chapter images”. Because it is a subfolder of the project, you can enter any valid folder name here. There is no navigation because the location is fixed: inside the project folder. By default the folder name is set to “ChapterImages”. This results in a folder `\<projectname>\ChapterImages`.

Important: If you generate chapter images for several movies in the project, you need to specify a unique folder name for each set of chapter thumbnails of each movie. If you don’t each subsequent movie overwrites the thumbnails of the previous movie.

There are two folders generated for each chapter image generation:

²⁹ You can also double click on the movie in the project tree. That too will open the scenes window.

- \ChaptersImages - contains the reduced small .png images. The folder name can be changed in “Folder for chapter images”
- \ChaptersImages\Bitmaps contains the .bmp images (uncompressed, not reduced, large size). The foldername \Bitmaps is fixed.

This is where it ends: you need to do the rest of the chapter menu creation yourself. The two chapter wizards use this bitmap generation step as first stage of their chapter menu generation and then continue with additional steps. But you can also use the bitmaps to create your own type of chapter popup menu using BDS animation features. Most will stick to the chapter menu wizards as it does most of what you would want.

Chapter menu wizards: Which menu wizard to use?

There are two menu wizards, each with its own speciality.

- Use “text + rect” if you want a menu with chapters neatly in a row or column and the menu appears to shift to keep the selected chapter in the middle. The background of the row or column as well as the frame around each chapter image can be modified. The selected chapter is highlighted (white frame by default) and remains the middle image of the shown image strip.



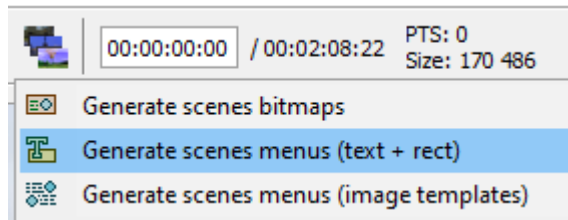
- Use “image template” for a fixed set of chapter images and buttons “previous” and “next” to replace them with a fresh set of chapter images. The buttons, chapter numbers and frames can be modified. There is no background. The selected chapter is highlighted (white frame by default) and this frame moves from left to right.



Both wizards have a step to generate chapter images. If you already made your own (or revisit the wizard to renew the menus), you can indicate you want to keep them rather than generating them again.

Chapter menu wizards: Create scenes menus via text and rectangles

When you select the second option, a chapter menu wizard is started that allows you to generate images for each chapter of a movie.



The menus that will be generated need to be opened from a popup menu of a movie. The “Generate scene menus” window contains very many settings you may need to specify.

Online help is provided via “Create a scene selection menu using text and rectangles”. Some sample projects (without the actual movies) are provided at <https://blu-disc.net/download/examples/ScenesMenuGenerator.zip> for your perusal.

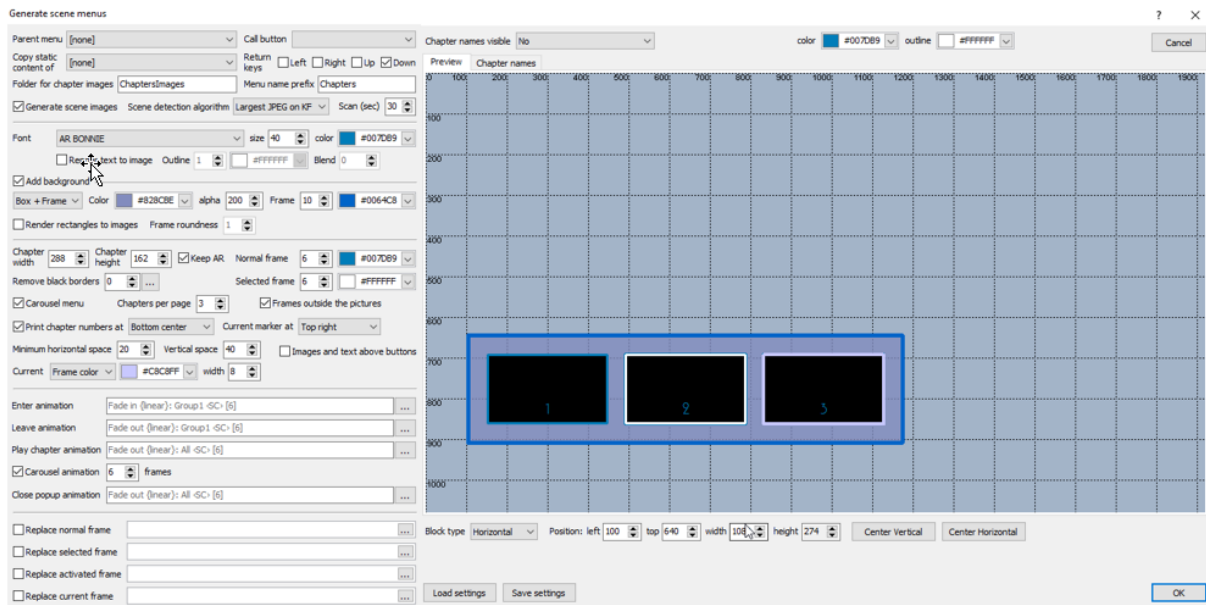
Each of the three projects can be unzipped in a project folder. The actual movies do not exist (some dummy black movies are added), but to see how a chapter image works, it suffices to use the “simulation”. The project has two project files:

- *template – the project before any chapter menus are generated (You cannot generate yourself because you do not have the movies)*
- *menu – after chapter menus have been generated*

See the “template” project first, then the resulting “Menus” project. Look at some of the generated chapter menus and its objects. See what images are used for background, frame etc. The settings used can be seen by opening the (black) movie’s scene window, open the wizard and load the .scnCreator file.

The wizard opens a new (fixed sized) window with a lot of options in the left-hand half of the window – many of which you can best discover by simply using them. The right-hand half shows the result in a preview window.

If you want to cancel the operation, the “Cancel” button is found at the top right hand corner. If you want to use the settings you are going to make, use the “OK” button at the bottom right hand corner. They’re not really adjacent as is normally the case.



Many of the colour selection options are best tried using a very different colour to see in the preview window what part of the image is affected. We will cover some of the important settings further down in this section.

The “Save Settings” button stores the settings you made so you can later reuse them through the “Load settings” button. This is also useful if you want to create identical chapter menus for different movies. Only some fields (especially the menu names and folder in which to store images) need to be changed on a per-movie basis. Some movie specific fields are not saved. So go over all settings to ensure they have the settings you want.

The saved values are stored in an XML structured file describing how menus change from one to the other. You specify its location when you press “Save Settings”.

When you click “OK” the current settings are stored in a “work file” in the project folder named <project name>. scnCreator. This file is overwritten each time you press “OK”, so save your settings using “Save” if you want to reuse them later.

Specify fonts to use

Because the wizard uses text, it requires a font for this. Before starting the wizard, ensure at least one font is specified in Project > Used Fonts (see “Add text object fonts” on page 91).

Connect chapter menus to a popup menu

The chapters belong to a movie. To show the chapter menu you normally open the popup menu while that movie plays. Or that popup menu will be an item “chapters” and when selected that button should open the chapter popup.

Whatever you do, it is important to link the movie “popup menu” property with a popup menu that shows the chapters. To link both menus, specify the movie “popup menu” with its chapter menu

through the “parent menu” text box. And link the specific “chapters” button on that movie popup with the showing of the chapter menu (“call button”).

Generate scene menus

Parent menu: Popup: popup menu Australia

Call button: chapter menu

Copy static content of: [none]

Return keys: ☐ Left ☐ Right ☐ Up ☒ Down

The “Return keys” indicate what remote control button to press to let the chapter menu disappear and return to the parent popup menu. By default the “Down” key is used as often the chapter menu is shown above the parent popup menu. To keep this parent menu visible (as a picture element), select its name in the “Copy static content of” box.

Where to store the chapter images

The next settings determine what thumbnail to generate. This will look familiar to the “Generate scenes bitmaps” window when you would only generate the bitmaps.

Folder for chapter images: ChaptersImages

Chapter menu name: Chapters

☒ Generate scene images

Scene detection algorithm: Largest JPEG on keyframe

Scan (sec): 30

Whether you generated the chapter images or set them yourself, they are stored in a specific folder. By default the generator uses “ChapterImages” as name (in folder \<projectname>\ChapterImages). If you have more movies with chapters, the “Chapterimages” name is probably set differently by yourself. In that case you also want to provide a movie-specific name for “Chapter menu name”.

Inside the folder all thumbnails for the chapters have the same format: the same name but are postfixed with “1”, “2” and further – as many sequential numbers as needed.

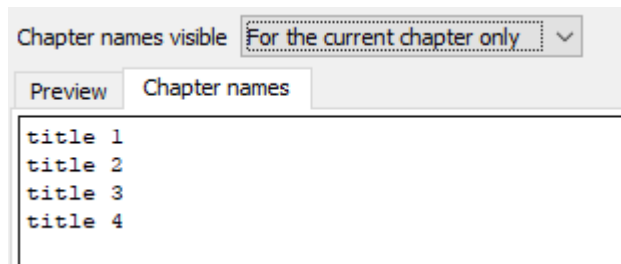
If you created the chapter images yourself, uncheck the “Generate scene images” and specify the “Folder for chapter images” to reflect to the folder in which you stored your thumbnails. (The wizard will ask you whether you want to overwrite the images if “Generate scenes images” is checked – say “no” or better: uncheck the option before you continue).

Specify font to use for any chapter titles and numbers

All chapters are numbered and optionally given a (short) title. For this some fonts are needed. When you want to use this wizard, you need to specify these fonts (Project > Used Fonts – see Add text object fonts on page 91) beforehand.

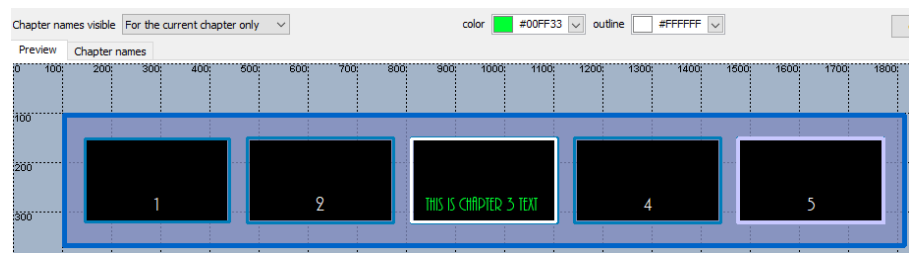
Names and numbers for chapters

Each chapter is numbered sequentially from 1 to final chapter. In addition you can specify a textual description for each chapter. To show text, when, where or not, can be specified in the “Chapter names visible” box. When no titles are entered, the numbers are shown.



The text for each chapter is entered in sequential order in the “Chapter names” tab. Selecting “For the current chapter only” means that the text is shown when the chapter image is the currently selected on. All other chapter images just show their number.

The colour of the text or chapter number of the selected chapter and that of its outline are specified by the colour boxes “color” and “outline” and is seen in the preview (under the tab with that name).³⁰



Horizontal or vertical chapter menus

You can decide whether the chapter images are shown as a horizontal or vertical strip of image buttons. The setting is done below the preview image of the chapter buttons.

In “Block type” you specify the ordering: horizontal or vertical. The position of the images strip is specified by “Position”. Note that changing values here does not show in the preview until the moment you move focus to another text box. You cannot use these settings to “nudge” the position of the chapter menu. Set a value, move away from the setting box and see the result. Alternatively you can center the strip vertically or horizontally.



Chapter image size

The number of chapter images is set in “Chapters per page”. The width and height of each image is specified in the corresponding boxes (which may be squeezed or stretched). If you want to maintain the 16:9 aspect ratio, check the “Keep AR” box.

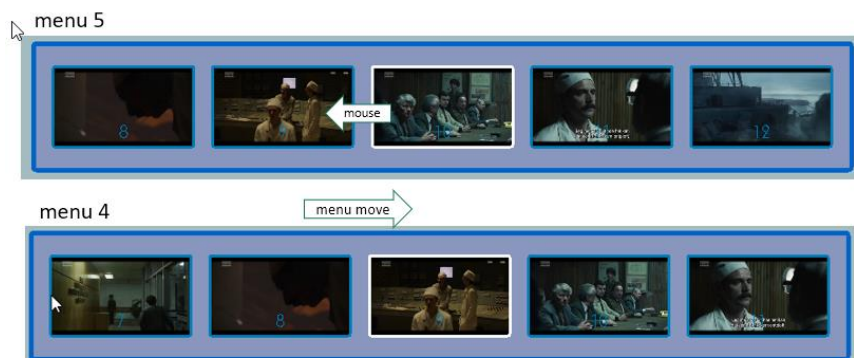
³⁰ The “This is Chapter N text” is a generic one given by BDS and does not reflect any title given under the “Chapter names” tab.

Chapter width 288 Chapter height 162 ☒ Keep AR Normal frame 6 #007DB9
 Remove black borders 0 ... Selected frame 6 #FFFFFF
☒ Carousel menu Chapters per page 3 ☒ Frames outside the pictures
☒ Print chapter numbers at Bottom center Current marker at Top right
 Minimum horizontal space 20 Vertical space 40 ☐ Images and text above buttons
 Current Frame color #C8C8FF width 8

Carousel

A carousel is a strip of moving images where the illusion of a loop is given in which chapter images can move to left or right (or top and bottom) in a continuous manner and where the last chapter connects with the first chapter. For this you check the “Carousel” box in the wizard menu (it’s checked by default).

If you want to move to an earlier or later chapter using the navigation buttons on the remote control, in fact it is the menu that seems to move (using menu animation – details in Part 4: Animations on page 247) in the opposite direction to get that earlier or later chapter image in the middle of the chapter images strip. In reality, one menu (with one button linking with a movie chapter N) is replaced by another menu (with another button linking to chapter N+1 or N-1).



Non-carousel

The non-carousel option (uncheck the “Carousel” textbox) produces a similar output chapter menu but the images do not move. A frame highlights what chapter is selected and the selection frame moves from left to right (or reverse) through the menu. Beyond the leftmost or rightmost chapter the chapter menu is replaced by another chapter image menu. After the last chapter you move to the first chapter and vice versa.

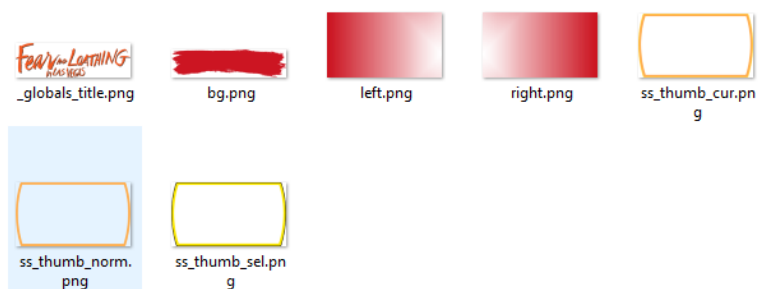
Different looks

The borders around the images as well as the background itself can be differently coloured or made of a different shape (specified as .png file) from the default rectangular form by replacing their frame in the bottom set of options.

In the example below (used in example project “Fear and Loathing in Las Vegas”) three out of four frames have been replaced by another one.

<input checked="" type="checkbox"/> Replace normal frame	J:\BDS Projects\PopupChapterMenuBR\Menu\ss_thumb_norm.png
<input checked="" type="checkbox"/> Replace selected frame	J:\BDS Projects\PopupChapterMenuBR\Menu\ss_thumb_sel.png
<input type="checkbox"/> Replace activated frame	
<input checked="" type="checkbox"/> Replace current frame	J:\BDS Projects\PopupChapterMenuBR\Menu\ss_thumb_cur.png

The corresponding images are shown below. Note that the “bg.png” image is set as the background image for the parent popup menu (the one with “play scenes options” on it that is copied as static content of the chapter generated menu).



and the result of a simulation project show the effects as the illustration below indicates (images taken from opening that example project and run its simulator).



The setting can be reused by loading an earlier configuration.

Create the menus

Clicking the “OK” button executes the settings (bitmaps and menus are generated). Images are created in the specified images folder. With 14 chapters you need 14 chapter menus in carousel mode (because you need another menu for each image chapter that is the only button on the menu (being the middle chapter image). In both cases the transition between menus is achieved through animations that show a sliding movement where apparently all chapter images shift by one position.

The chapter menus show up in the Project Tree “Menus” or “Popup Menus” branch. Each menu contains the images of the various chapters.

If you look at the objects inside a chapter menu, the images are picture elements of the generated chapters (“chapter NN img”).

Chapter menu wizards: Create scenes menus via image templates

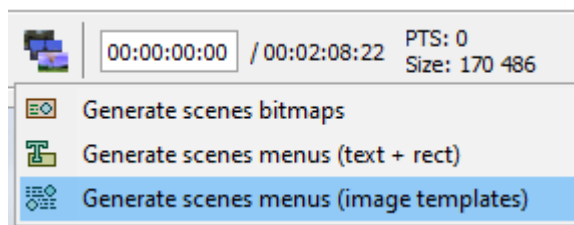
The second chapter wizard is invoked when the third menu option is selected. It creates chapter images that are buttons to jump to the movie chapter. The chapter images show up by themselves. There is no background – just a frame that tells you which of the chapter images is the currently selected one. The images do not move – the selection moves. When you arrive at the first or last displayed chapter, the entire chapter menu is replaced by the previous or next set of chapters.



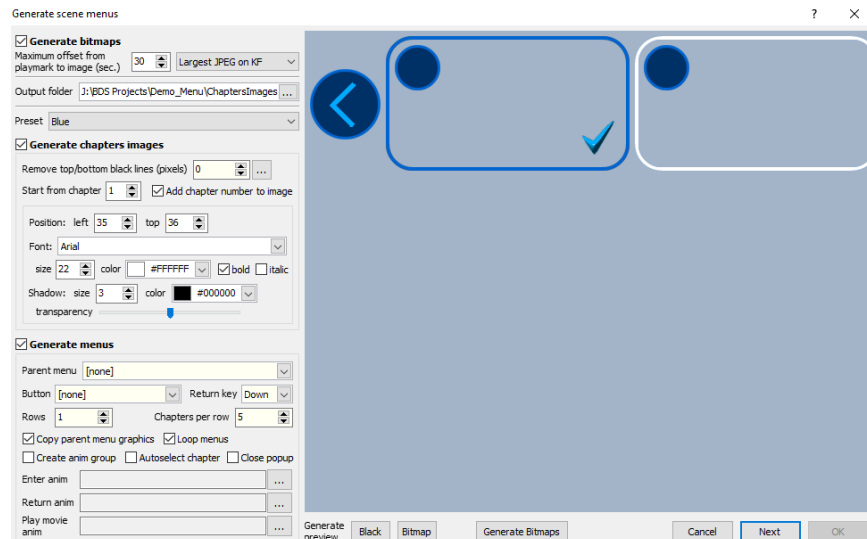
The chapters are connected and the last one goes on to the first one and vice versa. Moving left at chapter 1 jumps the menu from the first one shown above to the second one and the selected chapter is the last one, number 7. Moving right from 7 brings back the first menu with chapter 1 selected. Moving left from chapter 6 also the first menu is shown, but now with chapter 5 selected.

The arrows at left and right of the chapters are not really buttons but give the impression because they seem to cause the change in menus. In reality, the left or right most chapter button has the property “Press Left” or “Press Right” defined to change the menu.

The wizard is selected by clicking on the third menu option of the “Generate menus” button in the Scenes window that contains the chapters for a particular movie.



The wizard window that opens shows the three steps to create a menu from chapter images and allows you to perform all three steps or just some (in case you only need to redo a single step).



Online help is provided via “Create a scene selection menu using image templates”. Unlike the other chapter menu wizard, there is no “Save” or “Load” for the settings you specify. This gives the impression that both wizards were developed independently or in different time frames when other insights were applied.

The wizard can be used for multiple movies, just like the other one. As long as you specify separate folders to store chapter images for each movie so one movie won’t overwrite the images of the other.

It is advisable to change the generated menu names. These are named “Scenes N” or “popup Scenes N” where the number N increases from 1 to whatever number is needed. A second movie with chapters gets the next higher number N+1 and onwards. This makes it hard to distinguish whether “Scenes 5” is belonging to movie 1 or movie 2. It’s easier to rename a set of generated menus into names indicative of the movie they belong to. For example “movie 1 Scenes 1” and “movie 2 Scenes 5”. Or start renumbering the menus for each movie (“Movie 2 Scenes 1”) – as long as the menu name is unique.

The left-hand part of the window specifies the steps, enabled by checking the box of the step.

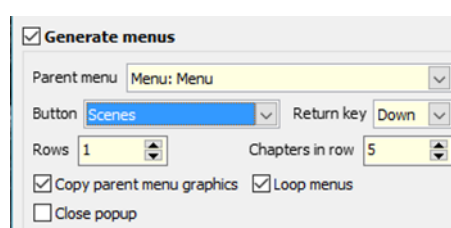
- check “Generate bitmaps” (creates the thumbnails). Do not check this if you created your own thumbnails – but you do need to specify the proper folder location of your images in “output folder”.
- check “Generate chapter images” (uses the thumbnails to create chapter buttons). Do not check this if you created your own thumbnails – but you do need to specify the proper folder location of your images in “output folder”.
- check “Generate menus” (uses the chapter buttons to create the chapter or scenes menu). The menu is automatically inserted into the menu list of the Project Tree window.

The right-hand part shows how each chapter image is framed and numbered. There are three pre-defined “presets” you can choose from (all identical but in different colours). The currently selected one is

highlighted with a white border. If the number of chapters is more than fits on a screen, a follow-on menu is created and linked using the > and < buttons shown.

With these three generating checks in place, the generation of images as well as menus for the chapters becomes fully automatic.

1. specify the output folder where the chapter images are stored. A suggestion \<project>\ChapterImages folder is already made as empty subfolder within the project folder. Change it to a unique name if there is more than one movie. When you created your own thumbnails, ensure you specify that folder.
2. Indicate how many thumbnail chapters are shown on a screen per popup menu (in the “Generate menus” block, “Rows” and “Chapters per row”). For a full screen several rows can be specified. For a popup window usually a single row is used.

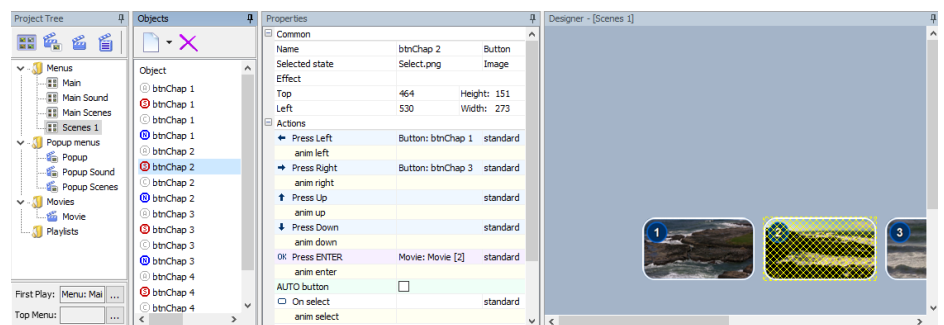


3. Indicate the parent menu of the chapter popup. This can be a movie popup window or a normal window. The dropdown list provides all eligible names of the menus. The menu is then shown in the chapter designer window so you can visually check where you want to display the thumbnails.
4. Set the “Return key” value. By default, it says that if the down arrow is pressed on the remote-control, the chapter popup menu disappears and the parent (popup) menu is displayed again.
5. Click on the “next” button to specify the area of the screen where the popup menu with chapters is going to overlap the movie (the previous step by default has the “Copy parent menu graphics” box checked so you see what space this parent already occupies on the screen)



6. Click on “OK” to start the generation. A scenes menu is now generated and inserted in the Project Tree list of menus. Each chapter thumbnail becomes a button that is linked with that movie chapter.

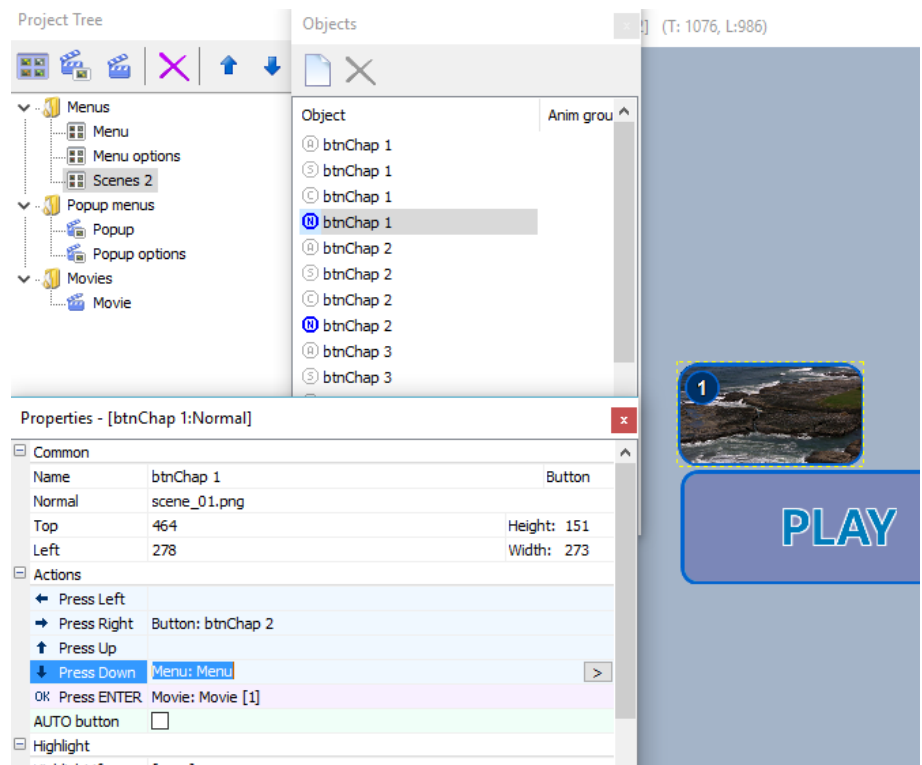
7. If the parent is a popup, the chapter menu(s) is found under Popup menus. If the parent is a normal menu, it is found under the Menus. You can (and should) rename the chapter menus in something more descriptive than the default “Scenes 1”, “Scenes 2” etc. With two movies it becomes difficult to distinguish whether “Scenes 15” belongs to the first or second movie. Give more descriptive names like “scenes movie 1-1”, “scenes movie 1-2” etc.
8. If there are more chapter images than fit on the number of rows then as many chapter menus are created as necessary. They are linked by buttons for “previous chapters” or “next chapters”. These buttons are also generated and entered in the Objects view of the Objects window. The figure below shows the “Scenes 1” generated menu. The second chapter thumbnail works as a button, “btnChap 2”, whose “Press Enter” property specifies to play the movie starting at Chapter 2.



The Designer Window of a chapter menu shows how it will look eventually when the “Scenes” button is activated. You can change the appearance by modifying the “Generate menus” data. For example, you may decide on having four buttons instead of the original five. This requires a re-generation of the menu (not the thumbnails) and may end up looking like the result below.

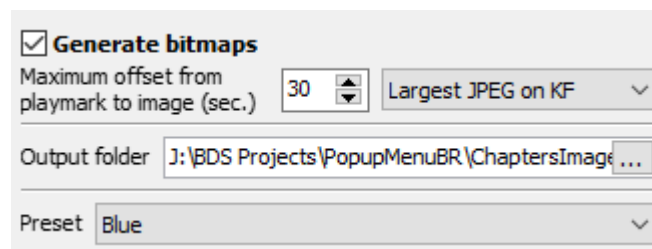
Before regenerating, you should remove the earlier generated Scenes menus (otherwise the new ones will sit next to the old ones as the wizard continues its numbering upwards. The older chapter menus will not be used anymore and are only confusing. Remove them from the project.





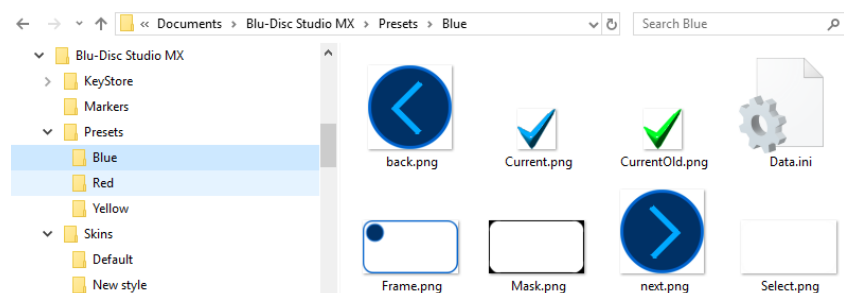
Customizing or making your own chapter menu preset

The BDS software comes with three presets for the chapters generated by this wizard (called “Blue”, “Yellow” and “Red”) and you select the preset in the “Generate bitmaps” part of the scenes generation menu.



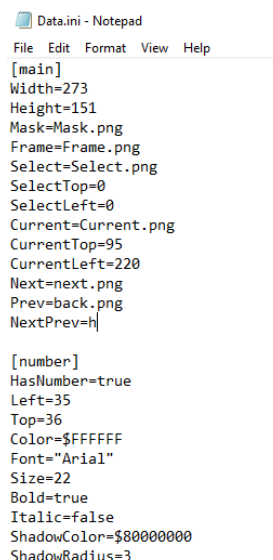
These presets are stored in your own local \Documents folder of the Windows system. You will find a subfolder there named <disk>:\Users<name>\Documents\Blu-Disc Studio MX where you will find the three presets.

The contents of the \Blue folder (one of the three standard presets) is shown below.



By adding new folders with unique folder names, you add new presets. The easiest way to do this, is by copying an existing standard preset folder and then modify all files in it to your liking. The names of the files must remain the same: BDS depends on their name to know when and where to use them. These names are specified in the data.ini file.

The data.ini file must remain in the folder and retain its name. Its contents can be modified to reflect if you want numbers assigned to your chapters and if so, where in the thumbnail you want the number to be displayed.



```
Data.ini - Notepad
File Edit Format View Help
[main]
Width=273
Height=151
Mask=Mask.png
Frame=Frame.png
Select=Select.png
SelectTop=0
SelectLeft=0
Current=Current.png
CurrentTop=95
CurrentLeft=220
Next=next.png
Prev=back.png
NextPrev=h

[number]
HasNumber=true
Left=35
Top=36
Color=$FFFFFF
Font="Arial"
Size=22
Bold=true
Italic=false
ShadowColor=$80000000
ShadowRadius=3
```

This is done in its [number] section.

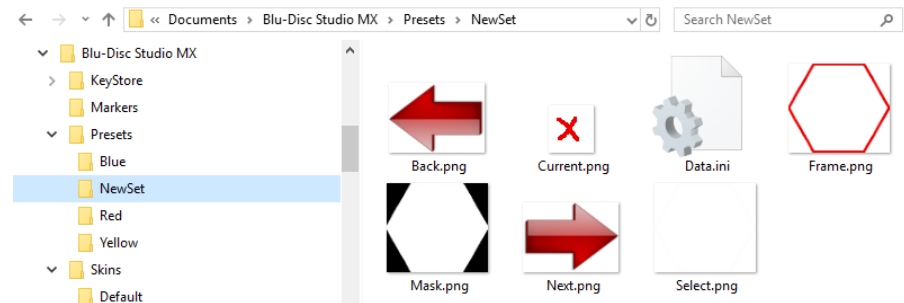
The other section, [main], you normally don't touch unless you feel an irresistible urge to modify:

- the filenames in the preset folder (like "next.png")
- the size of the chapter button images (Width and Height) – this may be needed for Academy sized 4:3 images or extreme widescreen not fitting the 16:9 aspect ratio.
- the position on the "current" button image (CurrentTop, CurrentLeft) or the "selected" image (SelectTop, SelectLeft)
- how to handle cursor movements (NextPrev = v (vertical) or h (horizontal)). Chapters shown horizontally have the left/right arrow keys defined to move between chapter menus. Chapters shown vertically, have the up/down arrow keys defined to facilitate moving to another chapters menu.

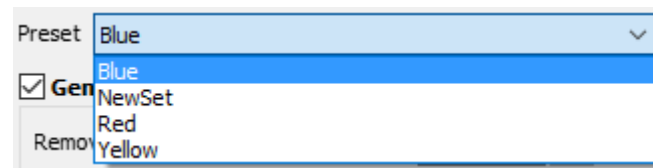
The correct setting may require a few trial-runs before you're satisfied with the final outcome.

Note: very confusingly, the preset folders are also part of the standard installation folder of BDS (on <systemdisk>:\Program Files\Blu-Disc Studio MX\Presets) but the software does not use this folder – it uses your personalized copy in your \Documents folder.

A simple set of modified chapter thumbnails masks and “back”/”next” arrows is given in the \Sources\Additional stuff\chapter frames folder in the example files for this user’s guide.



When copied to \Documents\Blu-disk Studio MX\Presets\NewSet they will show up in the dropdown menu as “NewSet” from which you select what type of chapter menu to use.



When this new preset is used, the result in Simulation looks like the figure below (on “Coasts” movie). The mask.png should mask more on the right side to hide the rectangular left-overs of the chapter images. Both mask and frame should be modified to fit the height of the thumbnails (or they should be generated to be larger and taller).



Chapters as menu option

We have shown how to generate a set of menus using one of two wizards. Usually these menus are connected to the popup menu of a movie to allow you to select a particular chapter of the movie currently running.

There may be a case where you may want to have an opening menu where the viewer is allowed to select:

- a feature film (with the usual popup menu to select a chapter of the feature film)
- a bonus movie (often behind the scenes of the feature film) where the viewer selects to either
 - run the entire bonus movie
 - run one chapter of the bonus movie by selecting that chapter from the main menu

Of course both movies could use the regular popup menu to select a chapter but in case of the bonus movie you would like to make that

choice beforehand – not by first starting the movie and then selecting the chapter.

With some “direct editing” and the use of a flag indicating the choice (a value set in an internal memory register) this can be done. This requires some more knowledge on using registers, so this is delayed to section Behind the scenes movie or just a chapter on page 362.**Error! Bookmark not defined.**

Recap: Quick Start Guide in 10 minutes

Now that you completed all of Part 2 with the basic steps needed to create a bluray disc, you may want to sit back, and watch all the steps you took in the previous chapters and see it done again in a silent demonstration movie “Quick Start Guide in 10 minutes”, available on the Blu-Disc Studio website or YouTube: https://youtu.be/0l_H6cuox0s

There is a written guide for this Quick Start Guide – in fact it was the first guide in the series on BDS I ever wrote as I could not make heads or tails from the BDS Lite product without any guide explaining the product. The movie helped a lot – when played frame by frame as you had to figure out what was going on there. Once you understood (and I hope the Part 2 projects gave you this understanding) it all became clear. As a famous Dutch soccer player (Johan Cruijff) once said in his inimitable logic: “you understand it, once you got it”.

The written guide can be copied from https://blu-disc.net/download/quick_start_guide.pdf if you still need it.

D.I.Y. Adventure

To venture out on your own, keep in mind:

- Start from the previous project
- Add popup menus to both movies and populate it with buttons that do the same as the Setting menu created earlier. It also has a button to open the chapter popup menu
- Generate chapter popup menus for both movies
- Assign popup menus to both movies. The menus should allow to invoke the chapter menus (generated by BDS) as well as set audio and subtitle tracks.

Step 0: Ready to run

If you don’t want to create this project yourself, you can setup the project from provided sources following these steps:

For a popup menu of the Settings (inactive chapter menu item):

1. Copy the folder tree \Sources\Projects\PopupSettingMenuBR to your BDS projects folder (e.g. J:\BDS Projects\PopupSettingMenuBR)

2. Copy the movie files Australia, Coasts and Menu from \Sources\Project objects\movies\demuxed to the project's \films folder (e.g. J:\BDS Projects\PopupMenuBR\films)
3. Copy all subtitles files from \Sources\Project objects\movies\subtitles to the project's \films folder
4. Double click on the PopupSettingMenu.bdmd.bdmd file in \PopupSettingMenuBR to start BDS and open the project
5. Click on the mux button and create the disc image
6. Experiment

For a popup menu of the Settings and chapters made by wizard 1: perform the same steps but in step 4 open project PopupSettingMenuChapterWizard1.bdmd

For a popup menu of the Settings and chapters made by wizard 2: perform the same steps but in step 4 open project PopupSettingMenuChapterWizard2.bdmd

Step 1: Get organized

We create a new project PopupSettingMenuBR that is based on an earlier project with a simple menu, a Setting menu and a simple popup menu(PopupMenuBR). It has the main ingredients upon which we can build, removing other extras to stay focused on the changes we want to make (popup menu and chapter popup).

1. Copy the entire \BDS Projects\PopupMenuBR folder tree to a new tree (and project) \BDSProjects\PopupSettingMenuBR. Skip or delete anything in the \Output.
2. Delete the project.bdmd file in this tree, rename project timeline.bdmd into project.bdmd
3. Delete the "\output timeline" folder and contents
4. Add two additional (empty) folders in the project folder tree that will receive the chapter thumbnails for the Coasts and Australia movies.
 PopupSettingMenuBR\ChaptersCoasts
 PopupSettingMenuBR\ChaptersAustralia
5. Open the project.bdmd project file to start BDS.

Step 2: Add a popup menu to set audio, subtitles and chapters

Use the regular procedures in BDS to create the popup menu. We will delete the "Popup Close" that in itself is pretty useless and create a new one.

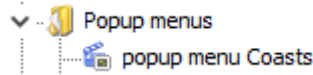
The popup menus should look like the figure below.



It has a solid white background. You may wish to make it more transparent to let the movie shine through or remove it entirely. In the

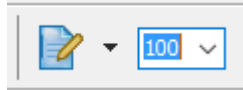
latter case, make sure the text and buttons stand all out against the movie image.

1. Delete the popup menu “Popup close”
2. Create a new popup menu placeholder. This will be a popup menu for a single movie, Coasts. Give it a descriptive name so you can tell which popup menu is shown during what movie.



We will complete one popup menu as much as possible and save time on the second or third similar looking popup menu by simply cloning it for the other movies.

3. Create the popup menu in one of two ways:
 1. Import the popup menu from a Photoshop file. A ready example can be copied from \Sources\Project objects\original sources\popup menu.psd
 2. Manually create the menu in BDS:
 - Add a white rectangle object (approx. top at 760, left at 100, height 250, width 1750 pixels). Rename it to “Rectangle”
 - Add a text box with the texts spread out like the figure above (with room to place the arrow buttons later). Rename text object to “Menu text”. (use approx. font Arial, font size 36 pt, bold, colour red)
 - Position the text box on top of the white rectangle
4. Add new buttons to the popup menu. Two ways of doing this:
 1. Copy a button from the main menu to the popup menu. A total of 7 buttons are needed for the items mentioned in the popup menu text. Their action properties will need to be adjusted later to work in the popup menu
 2. Create a button through the “New Object” button of the Objects window of the popup menu.
 - Specify an image for the Button’s “normal”, “selected” state. Images can be found in the project’s \buttons folder (or can be copied into this folder from \Sources\Project objects\buttons).
 - Duplicate this button 6 times (7 buttons in total) to have each menu item possess a button. All buttons show their “selected” state image on the Designer’s Window
 3. Give all buttons proper names (same as menu item they belong to)
5. Use the Designer window to adjust the position of menu background, texts and buttons to your liking. You may want to increase its window to 100% (use the scaling box at the top:



6. For the Audio (live/music) and subtitles (English/Dutch/none) buttons you need to specify an image for the “current” button state. A red arrow image is found in the \buttons project folder.
Possibly you need to scale it to fit the images of the other two states (right click on button state > change effects > scaling effect Type = Scale or use the “Effect” property of the button state to open the effects/scale menu)
(if you have many buttons with a state image that need to be rescaled by the same factor, you might select them all (in Object Window) and via a right mouse click select the “change effects” and for all specify scaling to x %).

***Note:** rescaling significantly (reducing to 50% or less) may impact playback on certain players. Best to keep images for buttons and menu approximately equal to the size required.*

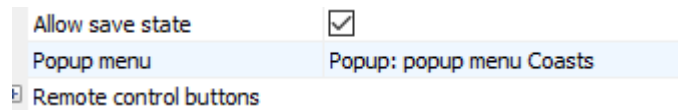
Alternatively, when you create the first button, assign images to normal, selected and current states. Then copy the button as often as needed. Remove the current state from the “chapter menu” and “close” buttons

7. Provide navigation between the buttons (On Left, On Right properties, etc) (The “Auto Assignment” button of BDS may help here if buttons are neatly aligned horizontally and vertically)
8. For Audio – live/music the “Press ENTER” action is Set Audio > Australia movies > Audio 1 (or Audio 2)
9. For Subtitles – English/Dutch/none the “Press ENTER” action is Set Subtitles > Australia movies > Subtitle 1 (or Subtitle 2 or -off- > subtitles off)
10. Skip the “Chapters” button for a moment (we haven’t created chapter popup menus yet)
11. For the “Close menu” button the “Press ENTER” action is Jump popup > [close popup]. Do the same for “Press Down”. Both will close the popup menu Coasts
12. Add the conditions for the “highlight” for the audio and subtitle buttons depending on what choice is made:
Highlight if Audio (or Subtitle)
in group Australia movies
is equal to 1, 2 (or 0 for subtitle “none”) (BDS should have filled this in for you)
13. Clone the menu “popup menu Coasts” and rename it “popup menu Australia”. Do it once more for “popup menu Playall”. Normally, this popup must be modified to remove any links to “Coasts” and change them into “Australia”. So far, there is nothing to change as there is no reference to either Coasts, Australia or playlist Playall. That will come: once the chapter scene menus are generated.

Each popup menu must be attached to the movie they belong to:

14. Set each movie's property "Popup Menu" to jump to the right popup menu

For the "Coasts" movie it is set as shown in the figure below. Either "Default" or "Live Audio" is taken as preselected button.



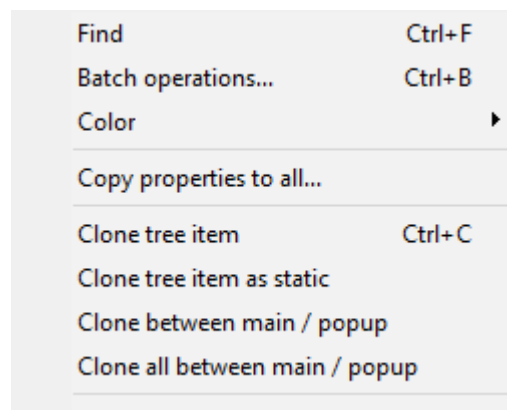
15. Save the current project file as File > Save As > "PopupSettingMenu.bdmd". We can later start with this project file if we use the other wizard without having any elements of the wizard we are going to use now.
16. Save the project once more as "PopupSettingMenuChapterWizard1"

Next, we'll add a chapter menu using both wizards for both movies. So take your pick which one to use. Or use both, but ensure they use unique menu names and picture folders to avoid problems with overwriting files.

Step 30: Clone items and Clone All items

In the next two steps, we'll generate menus that end up in the "Menus" branch of the project tree. You have to move these to the "Popup Menus" branch. BDS has a nifty option for this: you can clone items between "Menu" and "Poupup Menu" branches in either direction. There is however a fundamental difference between two clone options offered and you should avoid using the wrong one. The two options are:

- clone all between main/popup (or reverse)
- clone between main/popup (reverse)



You get this menu if you right click on a single menu item or if you click on the "Menus" or "Popup menus" to apply it to all items in that branch.

If the menus have navigation you want to preserve, use "Clone all". This copies all items and checks navigation afterwards. But nothing should happen: if navigation was correct in once branch, it should be OK in the clone in the other branch.


If you select one or more menus explicitly and only clone those, navigational checks are made after every cloned item. That means that some navigation may be pointing to menus not yet cloned. That navigation is removed. This may be totally unexpected behaviour because in the end all menus are cloned – but too late for the intervening checks.

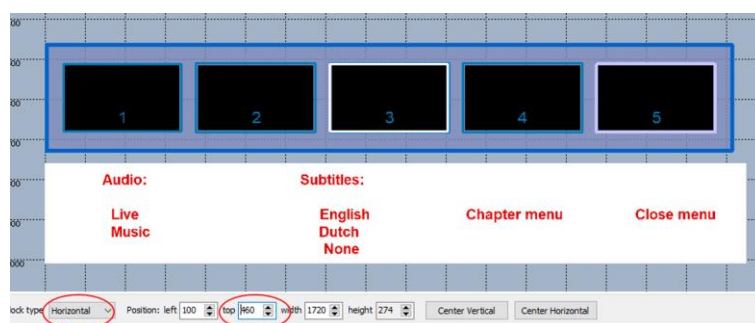
So the safest way not to lose any navigation is to use “clone all” – even if it means cloning items you do not want to clone. They need then to be deleted after the cloning.

Step 3A: Create chapter thumbnails with the 1st chapter wizard

Either the project file is still open and now named `PopupSettingMenuChapterWizard1` or it is closed and you open this project file from the root folder of the project `PopupSettingMenuBR`.

We start with creating a chapter menu for the Australia movie. Afterwards we repeat all steps for the Coasts movie – only the names use “coast” instead of “aus”).

1. Open the movie Australia and open the Scenes window
2. Click on the “Generate scenes” () button and select the second option “Generate scenes menus (text + rect)”.
3. Set the “Parent menu” to “popup: popup menu Australia”
This way you open the chapter menu from the popup menu assigned to the Australia movie
4. Set the “Call button” to “Chapter menu” (the name of the button on the Australia popup menu that will open the chapter menu)
5. Clone all elements of the “popup menu Australia” by specifying that menu name in “Copy static content of”.
6. This automatically inserts the menu elements of the popup as a picture. Adjust the height of the chapter images to place them above the popup menu picture element. For this, modify the chapter block position through the settings under the preview window. The “top” position must be set to 460 instead of the default 600. Use the “horizontal” block type.



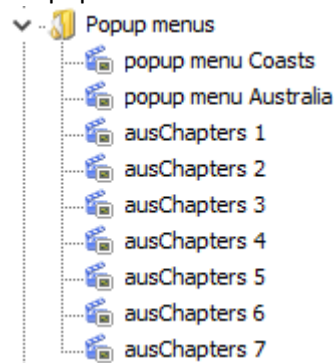
7. Modify the “Folder for chapter images” to “ausChapterImages” and the “Chapter menu name” to “ausChapters” (BDS will append a sequence number to each menu required)

8. Check “Generate scene images” and reduce the scan distance to 5 seconds

Generate scene menus

Parent menu	Popup: popup menu Australia	Call button	chapter menu
Copy static content of	Popup: popup menu Australia	Return keys	<input type="checkbox"/> Left <input type="checkbox"/> Right <input type="checkbox"/> Up <input checked="" type="checkbox"/> Down
Folder for chapter images	ausChaptersImages	Chapter menu name	ausChapters
<input checked="" type="checkbox"/> Generate scene images	Scene detection algorithm	Largest JPEG on keyframe	Scan (sec) 5

9. Check the “Carousel menu” checkbox (is checked by default)
10. Leave all other settings to their default (or experiment with them later)
11. Save your settings by clicking “Save Settings”. By default the settings are stored in the project folder and get file type .scnCreator. Give it the name “AusCarousel”.
12. Generate all chapter menus by clicking “OK”. When the wizard finishes, close the movie’s Scenes window. Notice the “Popup menu” now contains 7 chapter menus



Several menu object properties have been modified by the wizard to show the chapter popup and the change in selection between chapters.

13. Run the simulation, select the “Sydney” Australia movie, press F5 (simulates a popup button), select the “chapters” menu item and see if the chapter menu works. Notice it opens with the starting chapter 1 in the middle. Selecting another chapter makes the chapter images move to keep the selected chapter in the middle.
14. By clicking the down arrow the chapter menu disappears and the “Chapters menu” button becomes selected again.



Now repeat all steps for the Coasts movie (use “coast” prefix instead of “aus”) and link those chapters to the coasts popup menu and its “chapter menu” button. If the Coasts movie has fewer than 5 chapters all chapters can be shown at once. A carousel is then not needed. If BDS detects this, it warns you and asks to ignore carousel.

Repeat the steps once more for the playlisat “Playall” using the prefix “play”.

Save the project. This way you reopen the project using this project file. All elements of the wizard are then available again.

Alternative non-Carousel

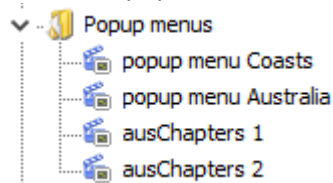
You may recreate the chapter menus using the non-Carousel manner. For this:

1. Reload the project “PopupSettingMenus.bdmd” (effectively starting from scratch again)
2. Repeat all steps from above, except for step 10: uncheck the “Carousel menu” (and further down: “Carousel animation”)

Generate scene menus

Parent menu	Popup: popup menu Australia	Call button	chapter menu
Copy static content of	Popup: popup menu Australia	Return keys	<input type="checkbox"/> Left <input type="checkbox"/> Right <input type="checkbox"/> Up <input checked="" type="checkbox"/> Down
Folder for chapter images	ausChaptersImages	Chapter menu name	ausChapters
<input checked="" type="checkbox"/> Generate scene images	Scene detection algorithm	Largest JPEG on keyframe	Scan (sec) 5
Font	AR BONNIE	size	40
	color	#007DB9	
<input type="checkbox"/> Render text to image	Outline	1	
	Blend	0	
<input checked="" type="checkbox"/> Add background			
Box + Frame	Color	#828CBE	alpha 200
	Frame	10	
<input type="checkbox"/> Render rectangles to images	Frame roundness	1	
Chapter width	288	Chapter height	162
<input checked="" type="checkbox"/> Keep AR	Normal frame	6	
Remove black borders	0	Selected frame	6
<input type="checkbox"/> Carousel menu	Chapters per page	5	
	<input checked="" type="checkbox"/> Frames outside the pictures		

3. Save the settings as “AusNonCarousel” file for future (re)use
4. Run the wizard by clicking on “OK”
5. Notice now only 2 chapter menus “ausChapters” are created under “Popup Menus”



6. Run the simulator and see the effects (select “Sydney” movie and press F5 to enter the popup menu of the movie)




The resulting menu is static and the selected (white framed) chapter stays in place while the selection frame changes. Moving before the leftmost or after the rightmost chapter, the menu is changed to show the other chapter menu. First and last chapter are connected.

7. Repeat the steps for the Coasts movie (using “coast” as prefix rather than “aus”) and the play all playlist.
8. Save the project (as PopupSettingMenuChapterWizard1)

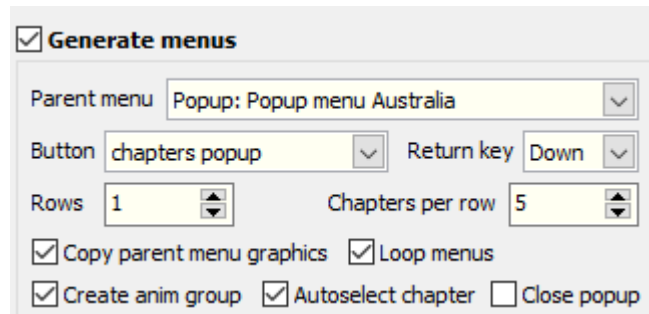
Step 3B: Create chapter thumbnails with the 2nd chapter wizard

For each movie, chapter menus (and button links to the chapters) must be generated. We will use the second chapter wizard here.

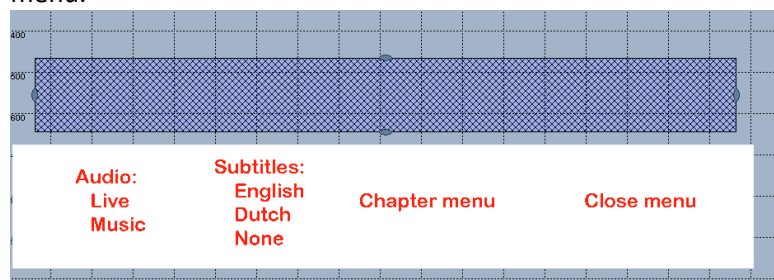
1. Start fresh by opening the project “PopupSettingMenu.bdmd” which has the project state without any chapter menus.
2. Save the project as “PopupSettingMenuChapterWizard2”
3. Open the Australia movie and open the Scenes window
4. Click on the “Generate scenes” () button and select the third option “Generate scenes menus (image templates)”.
5. Set the offset from playmark to 5 seconds
6. Modify the “Output” folder below the “Generate bitmaps” to link to the movie-specific chapter images. Because the project folder is also used by the other project using Wizard 1, use a different prefix, like \aus2ChapterImages

Output folder I:\BDS Projects\PopupSettingMenuBR\aus2Ch. ...

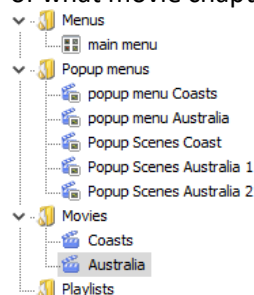
7. Select a preset for the appearance of the chapter menu. (E.g. take the “blue” preset that comes with BDS)
8. In section “Generate menus” specify the parent menu. This is the popup menu that belongs to the movie. Keep the other options at their default for now. In the case of the “Australia” movie the chapter popup has “popup menu Australia” as parent.



9. Ensure all three checkboxes to Generate bitmaps, Generate chapters images and Generate menus are checked (if you need to redo some actions, some may be unchecked if that part is not modified)
10. Click “Next” to position the chapter images (hatched rectangle) to a suitable position not overlapping the parent menu.



11. Click “OK” to get the process started.
12. Rename the created “Scenes” popup menu to be descriptive of what movie chapters they contain.

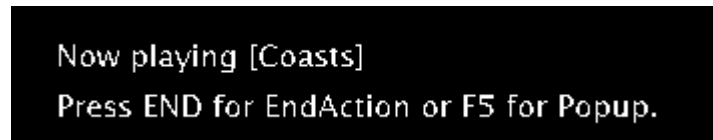


13. Repeat these steps for the movie Coasts. Use prefix coast2 for the images. Also rename the resulting popup menus as “Popup Scenes Coasts *n*”
14. Repeat the steps once more for playlist Playall. Use prefix play2. Rename the created popup menus to “Popup Scenes Playall *n*”
15. The popup menus for movie Australia and Coasts as well as the popup menu for playlist PlayAll have now been updated: the “chapter popup” button has a Switch action (will be discussed in Switch on page 203) that implies that depending on the situation one or the other menu and button is shown as selected.

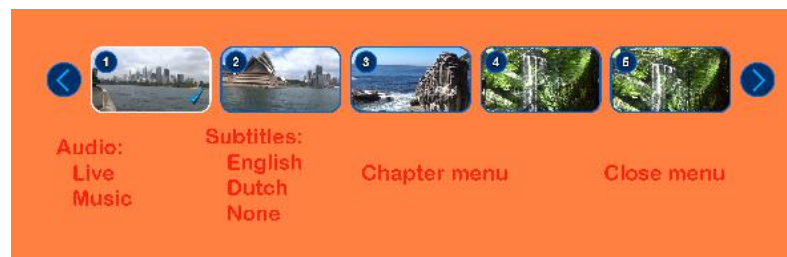
Save the project. This way you can reopen the project using this project file `PopupSettingMenuChapterWizard2.bdmd`. All elements of the wizard are then available again for further experimentation.

Step 4: Run simulation

Click the simulation button on the BDS menu to inspect if all navigation between buttons, their states as well as actions perform well. The actual movies will not be shown when you click on a button, instead a window is displayed what you can do.



Press the “End” key on your keyboard to execute the “End Action” of the movie (usually this goes back to the menu) or “F5” to enable the popup menu of the movie to appear. The “chapters” button should open the chapters menu and some next/previous buttons if there are more chapters than fit on the screen. (I choose to remove the white background of the popup menu if the chapters popup is shown – you can of course keep it).



The simulator does not enter a movie chapter – a checkmark indicates what chapter you choose. Nothing further happens. Pressing the down button or “popup menu” on the remote control makes the chapter popup disappear and returns to the popup that contains the chapter menu button.

Step 5: Create the bluray disc

This step is identical to the first project Step 4: Build the disc on page 64.). Make sure the Project > Project Settings > General tab points to the correct \Output folder (especially if you copied a project into a new folder).

Step 6: Burn the bluray disc

This step is identical to the first project Step 6: Burn the bluray disc on page 71.

Summary of basic steps to author a bluray disc

If you followed the project steps outlined in this part of the document, you may already know the basic steps to take. By default BDS stores all menu items in a single Java archive (JAR)³¹.

As a memory aid, we'll summarize those steps here.

Two major parts

Authoring a bluray disc consists of two major steps, that can be executed in either order:

- Create movie placeholders assign streams to their video, audio and subtitle streams
- Create menus, buttons and navigation. Check navigation and Press ENTER properties with "simulation"

Create movies

1. Demux all participating movies
2. Add as many movie placeholders in a BDS project in the project tree under the "Movie" branch
3. Assign video, audio and subtitle stream files to each movie placeholder (in the property window of that movie placeholder)
4. Create chapter index file (through "Scenes" property or double clicking on the movie placeholder). The start is automatically set as chapter 1. Any other chapters you may enter manually or automatically (fixed time intervals)
5. Assign End Action and Popup Menu action to each movie (can be "Jump Menu: main")
(This step only works if menus have been defined – may have to do this later)

Create menus

With Photoshop

1. Create new .psd file for menu sized 1920x1080 pixels
2. Position all image and text elements. Put them in several layers (only the non transparent rectangular parts count as pixel in BDS – a single layer full sized Photoshop image counts for 1920x1080 pixels). One text element may describe all movies to be shown.
Optionally, add one button (max 4 layers for states N,S,C,A) occupies 4 layers under strict naming convention. Do not create more buttons: copy/paste will work better in BDS. Better is not to add any buttons but do it entirely from BDS (see further down)

³¹ From V4.3 onwards also in multiple JARs if needed – but only if you specify this yourself (see Using multiple JAR titles on page 195)

3. Import menu into BDS as Photoshop file. Results in a number of “Picture” objects in the menu (all result in .png files in the \projectname\menuname folder)

With any image editor

1. Create each menu element (picture, text) as a separate .png file. Make sure unused parts remain transparent.
2. Create a blank BDS menu
3. Add a new object into the menu (in its Objects window)
4. In the properties of the new object, specify “picture object” and assign a .png image to it (in the properties window of the object).
5. Position the added item in BDS Designer window
6. Repeat steps 3-5 as often as there are menu elements

Within BDS

1. Load any font that may be needed for menu texts (Project > Usd Fonts)
2. Create a blank menu
3. Add menu objects to create texts, rectangles using the Objects Window button “New Object”³²

For all: add buttons

1. Activate both “Normal” and “Selected” (blue & red buttons on BDS menu strip at the top) states to be shown in BDS Designer window
2. Select a menu (this synchronizes with the Objects menu showing all menu elements and for the one selected object the properties window synchronizes with menu or menu element)

The next steps can be skipped if you made a button in Photoshop:

3. Add a new object in the Objects window³³
4. In the properties of the new object, specify “button” as type.
5. Specify the “Selected” state image of the button (e.g. a checkmark or arrow)
6. The new object is created at the top left in the menu.
7. Position the button to its proper place next to the text describing the movie to show (you may need to unlock objects before you can move them)

The following steps apply to all:

8. Copy/Paste the button (copy is on top of the original) and drag the copy to another text referring to another movie
9. Repeat previous step for as many buttons as required
10. Select all buttons (in menu Object view or in Designer window) and align them properly (right mouse click)

³² Preferably select from the dropdown list made visible by clicking on the arrow next to the button – these choices seem to work more consistent.

³³ See footnote 25

11. In the Objects window rearrange buttons so they are listed from top to bottom as they appear on the menu. All picture elements of the menu should follow below the buttons.
12. Add navigation between buttons by specifying "Press Left"/"Right"/"Up"/"Down" what to do (jump to other button). The "Auto assignment" feature of BDS may be useful here.
13. Add action to button by specifying the "Press Enter" action (play movie, jump to other menu). With a movie to play, you specify the movie placeholder. The actual movie need not yet exist.
14. Run simulation to check buttons work properly. Use F1 (end movie) and F5 (open popup menu) when you "OK"ed a button to run a movie and need to get back to the menu
15. Assign "End Action" and "Popup Menu" action to jump to the created menu (if this has not already been done)
16. Fill menu with its "menu movie": assign video and audio streams. This movie is not listed in the movie placeholders.

Prepare disc

1. Fill "On JAR startup" text box below the project tree with the first menu or movie to run when the disc starts playing.
2. If you want "Top Menu" also to work from the remote control, enable "Top Menu" (Project > JAR Setting > Edit 0000 > check "Allow Top Menu Call", Save, close) and specify first menu or movie to run when Top Menu button is pressed.

Build disc

1. Run the muxer to mux the entire disc. Specify the folder in which the \BDMV and \CERTIFICATE folders ought to be stored. Rerunning the muxer renews these folders
2. Check result in a s/w player or burn to the RW disc for playing in a set top player
3. Make any corrections:
 - a. if only on navigation or menu elements: renew only the JAR part (press JAR button, and replace jar file in the output folder)
 - b. if movies are added, removed, changed: remux entire project

Burn disc

1. Burn the folders \BDMV and \CERTIFICATE onto a blank disc using a disc burning program like Nero or the freeware ImgBurn
 - a. Ensure you use UDF V2.50 format for the physical partitions
 - b. Use Datatype Mode 1/2048

Part 3: Special Operations

(not of the Putin kind)

Interlude: Chapter marks – standard or direct editing

Chapters are added to movies through the Scenes window. You can either manually specify the points where a chapter must start or let them be generated with specific intervals.

There are two ways of adding (modifying or deleting) chapter marks, both of which open the same Scenes window but partly allows different things to do:

- Standard
 - double click on the movie placeholder and enter the Scenes window
 - right click on the movie placeholder and select “Open Scenes”
 - select movie placeholder in project tree and press “Enter”
 - select movie placeholder properties. Select “Scenes” and click on “>”
- Direct editing:
 - right click on the movie placeholder and select “Open Scenes (Direct Editing)”
 - Select movie placeholder and press SHIFT+Enter

The general steps on what you can do in the opened Scenes window is explained in “Step 3: Creating chapters” on page 63 and some further remarks in “Chapter marks import/export” on page 164.

When chapter marks are created they are normally used for two purposes:

- To start playing a movie from that point onwards. Either by pressing the “next chapter” on your remote control or by pressing a button on a displayed chapter menu whose Press ENTER action is defined as “Play Movie *name*[chapter]”. Chapter menus often have (image)buttons like these. Chapter menus are discussed in “Chapter popup menu generation through wizards” on page 166.
- To program an action to jump to a chapter and play that movie from that point onwards. These actions are often used to skip certain parts (like end and opening titles of subsequent episodes). This is discussed in “Method 1: Use a playlist with direct editing scenes (Binge Watching)” on page 296.

So why are there two ways to define chapter marks and when do you use which method?

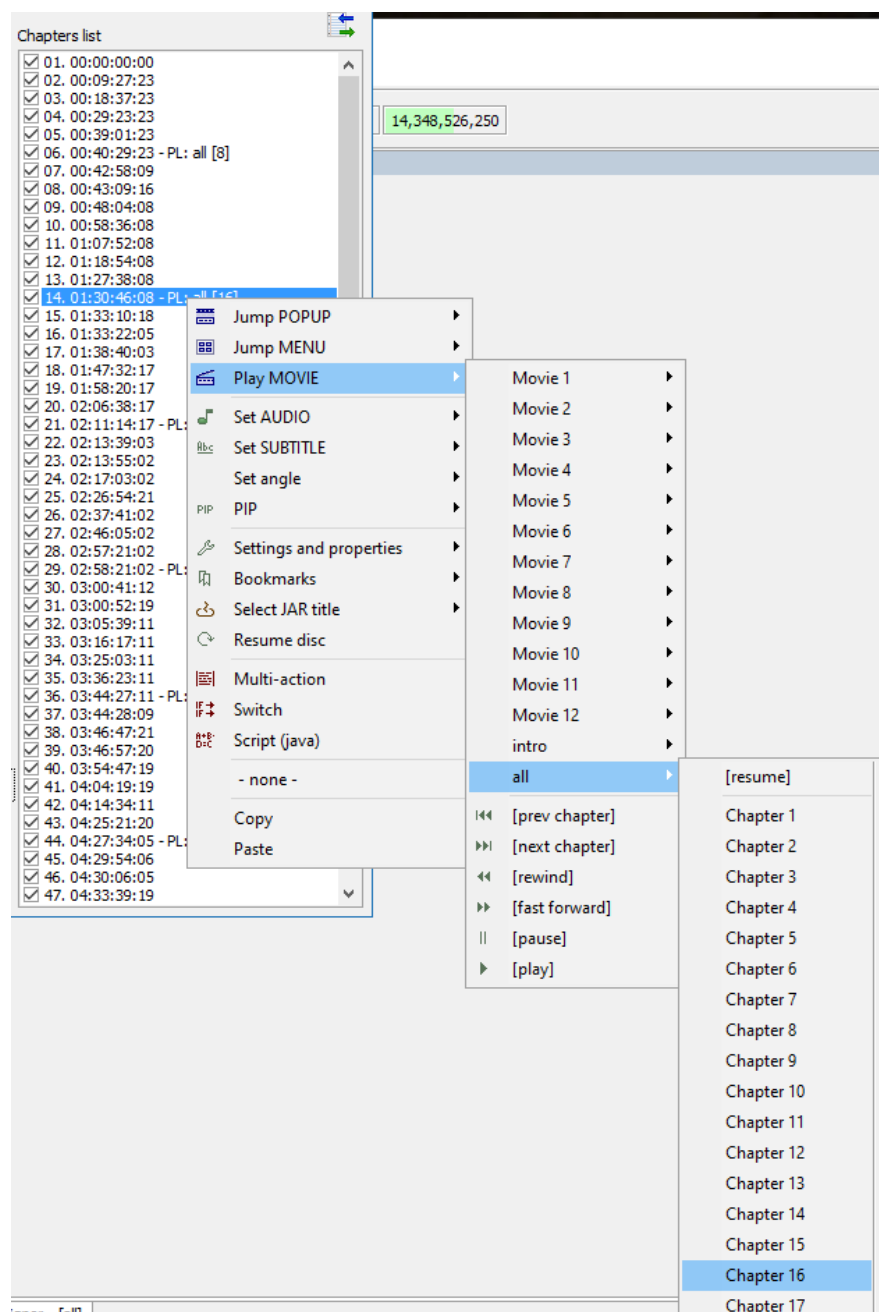
Standard

In “Standard” mode, a chapter mark happens to have a name and happens to have a point in the movie. Both are volatile. When 5 more chapters are added (in Standard mode) before chapter 10 then the old chapter 10 becomes chapter 15. Anything referring to “chapter 10” now refers to the new mark with the name “chapter 10”. That is an earlier position in the movie.

In Standard you cannot right-click on a chapter to get a context menu of actions.

Direct Editing

In “Direct Editing” mode, a chapter mark is bound to the time code. It may change its name, but the time code remains unaltered. Imagine chapter 10 refers to position 0:10:00 in the movie. When 5 more chapters are added (in Direct Editing mode) before chapter 10 then the old chapter 10 becomes chapter 15. But this chapter 15 now has the same time code of 0:10:00



Direct Editing mode allows you to define one or more (using Multi-Action – see next section) actions to each chapter mark. Select the chapter and right click to open the context menu for actions to specify.

For example (as illustrated), you can specify that on reaching chapter 14 of the “play all” playlist, you need to jump to chapter 16 and skip any movie material between chapters 14 and 16 (probably end and opening titles of two movies joined in a playlist).

If for some reason you add 5 more chapters before chapter 14, then chapter 14 will become $14+5=19$ and the old chapter 16 becomes $16+5=21$. The “jump” instruction on arrival of the now chapter 19 will read “jump to chapter 21”. The same movie material is skipped as before the addition of chapters. The “jump” instruction keeps the same time code regardless what chapter number points to it.³⁴

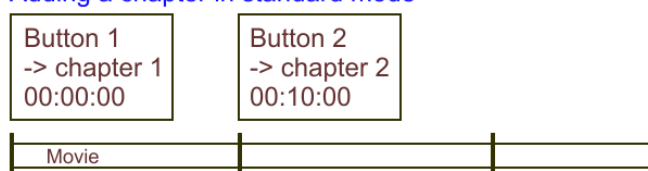
Original



Adding a chapter in direct mode



Adding a chapter in standard mode



When to use Direct Editing?

- When you add actions at a particular time code. That point is indicated by a chapter mark.
- The action remains at that time code regardless of whether chapter marks are added before or after that time code. Also if the action involves a time code in the future, it keeps that time code. The associated chapter numbers may change but the time code does not.

When to use Standard?

- Only if you want chapter marks made without any menu or programmatic link (e.g. simply use remote control to skip over chapters)
- If there are buttons linked to a particular chapter number, it keeps that number even if the time code changes. (like “always go to chapter 5” (regardless its time code position)).

³⁴ For programmers: the chapter is an object. The jump refers to the object. The object remains the same, just its name (number) changes and hence the jump is still to the same timecode stored in the object

- ***Do not*** use it if you got actions defined in Direct Editing mode. Changing chapters in “Standard” mode will then change the destination time codes.

Interlude: Multi-action and Switch actions

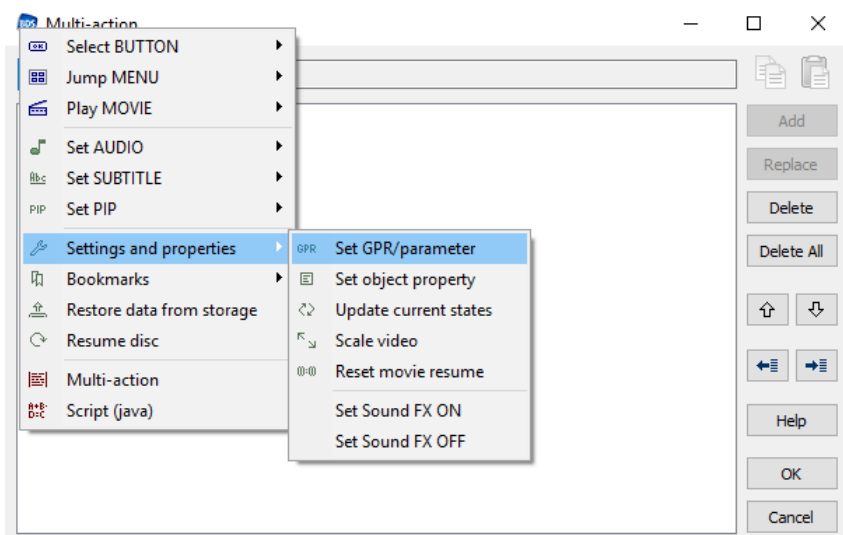
Multi-action

The Multi-action action can be specified for any property like “Press Enter”, “End Action” of menus, movies and buttons where usually a single action is specified. But sometimes this is not enough. You need two actions to execute or more. This is where multiple actions are needed.

Click on the “>” button at the far right of the property line and from the dropdown menu, select the “Multi-action” option. This opens a window in which you can specify what action to use. Once added, new actions can be added one by one.



Simply click on the “...” to show all possible actions available to add. These include moving to a button, menu or play a movie with specific audio or subtitle track. More specific (if you need to enter the programming realm) you can set GPR register values to be used in other Switch actions that can conditionally check these values. And more options.



Each action is added to the list when you click the “Add” button.

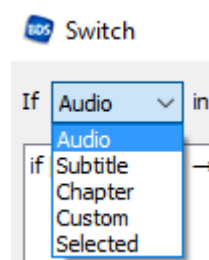
The order in which they are to be executed can be modified by using the up and down arrow buttons.

Once satisfied with the actions, click “OK” to copy them to the action property of the menu, button or movie.

Switch

The Switch action is a conditional variation on the Multi-action option as action for a menu, button or movie’s action property like “Press Enter” or “End Action”.

Like the Multi-action option, the Switch can specify several actions to perform, but only if conditions are met. To add a condition, click on the “...” button.



This provides for a number of pre-defined conditions that can check the current setting of an audio track, subtitle track, playing chapter or what menu or button currently has the “selected” state.

A condition is added using the “Add” button. By clicking on “...” button again a new one is composed. The added conditions can be ordered using the up and down arrow buttons of the window. You need to click on “OK” for the Switch conditions to be set in the property of the menu, button or movie whose action is to execute the Switch.

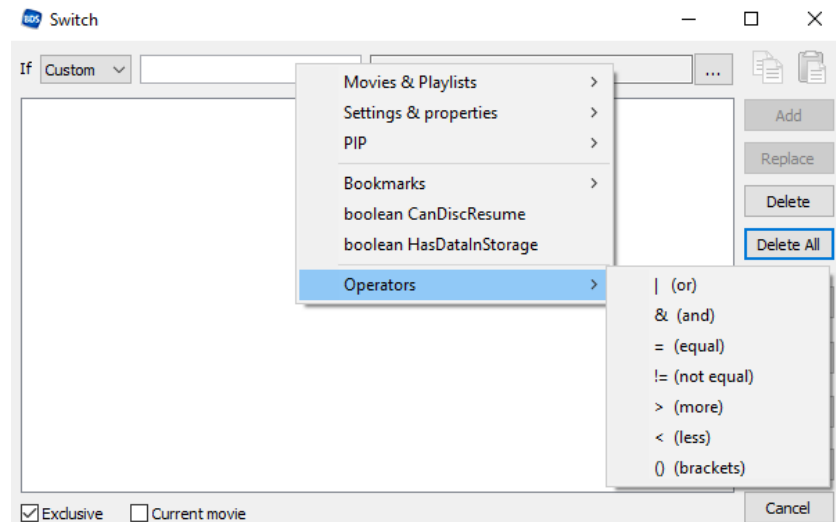
If you enter the programming realm, the “Custom” condition is what you will need. When you select “Custom” the next box is cleared.



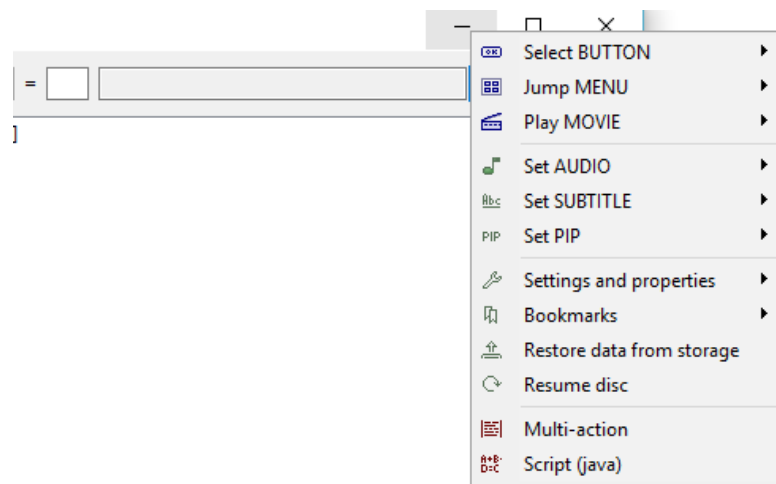
To execute the action (to be specified in the next text box) always, leave the custom condition empty.

If a condition is to be added, right click on the condition text box next to “Custom” and select the right one from the dropdown menu. See the online help section “Switch” for a description of all possible conditions.

The item “operations” in the condition dropdown menu highlights what logical operations can be used to compose the condition. In addition, the “>=” (greater or equal) and “<=” (less or equal) are also possible. You can repeatedly open this menu and use it to build a condition consisting of, for example, a particular menu setting and value in a GPR register, to be combined through a logical operator.



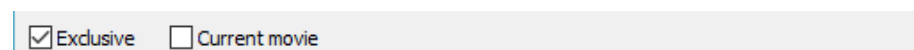
Once the condition is composed, its associated action (if condition evaluates to “true”) must be entered in the following text box. Again, click the right mouse button to display a menu of choices.



One of the options is “Multi-Action”. Hence if a condition is met, though the selection of Multi-Action you can specify a whole set of actions to take on that condition (the reverse is not true: inside a Multi-Action you cannot specify a Switch).

The default setting of a Switch is “exclusive”. This means that the list of conditions is traversed until the first condition is true. The associated action is performed. Conditions further down the Switch list are not examined.

To execute all of the (evaluating to true) conditions, you need to uncheck the “Exclusive” check box at the bottom of the Switch window.



Whatever is triggered by the conditional action, may need to reset the situation such that the condition becomes false – this may be the case in Easter egg situations where you do not want the egg to be found always once the right combination of buttons is pressed and the GPR

register has the correct value. This value should be reset so that each time again the “secret combination” of button presses is executed.

Interlude: JAR title files with menu or movie

The topic of this section may not be of any relevance to you (and irrelevant if you use BDS pre-V4.3). Graphics are stored in a Java JAR file by BDS. Pre-V4.3 only a single JAR file existed. This limited your graphics content to a maximum of 7 900 000 pixels³⁵.

If you needed more, you ran out of luck. Java doesn't allow any more. As of V4.3 the BDS software is capable of storing menus, buttons and graphics in multiple JAR files. Hence the original "First Play" in the Project Tree was replaced by "On JAR Startup" with the number of the JAR file. The first (and often only) JAR file has number 00000. But as of V4.3 you can have more (they get sequentially numbered) and you can select any of those JAR files to start your disc with.

The solution of the graphics limitation comes with added bookkeeping requirement for the disc author. What menu, button or movieplaceholder goes into what JAR file?

To use multiple JAR files, a requirement is that they are signed (see "Step 5: Signing and restoring (resuming)" on page 44 or "Project Settings

Apart from the movies, some project specifics need to be set before you can create the disc.

Sign JAR" on page 56).

Menus and buttons are stored in JAR files. Movies are not part of a JAR. Some project specifics need to be set before you can create the disc.

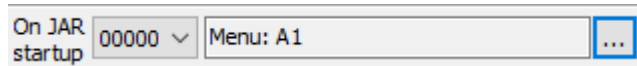
The BDS team provided an example project using this multi-jar approach in their collection of "Simple" projects that can be downloaded from their website (look on page <https://blu-disc.net/examples>). Any menu switch that involves changing JAR titles takes a little longer than displaying another menu from the same JAR title.

Warning If there is no real need for multiple jar files (titles), you should not consider using this feature. It brings you nothing extra but a lot of bookkeeping, debugging and possible headaches. "Proceed at your own risk" is the motto.

Warning If one of the menu movies is somehow corrupt, you may get a successfully muxed disc but on playback the disc may not play or "hang". Be aware of this – it cost me over a day to figure that out.

When not using multiple JAR titles, simply use the Project Tree "On JAR startup" box to set the movie or menu to show as "First Play" for the only JAR title in existence: 00000.

³⁵ another reason to use multiple JAR files is when 3D movies are used. A JAR title is either 2D or 3D. Regular menus pointing to 3D movies are in a 2D JAR but a button that starts a 3D movie points to another JAR that has "On JAR Startup" this 3D movie – even if movies are not part of the JAR itself.



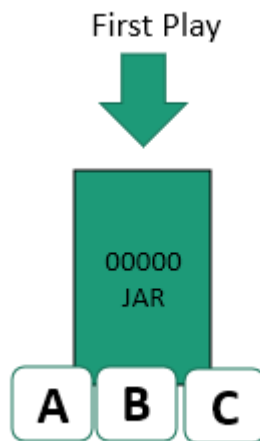
The JAR file

BDS is an application that runs Java code. The player will invoke the Java code that is stored in so called "Java ARchives", or JAR files for short.

"Title" in Blu-ray does not mean movie - it is a container that can be of Java (BD-J) or HDMV (High Definition Movie) type. For Java it just starts the JAR file, but for HDMV it contains HDMV commands. So, usually, HDMV disc uses several titles (each command set, like starting a movie, is a separate title), but for Java it usually runs in one title: the 00000.JAR file. Hence (software) players will always show information about "Title 00000" and never use the Title count value for a movie that is played like HDMV does.

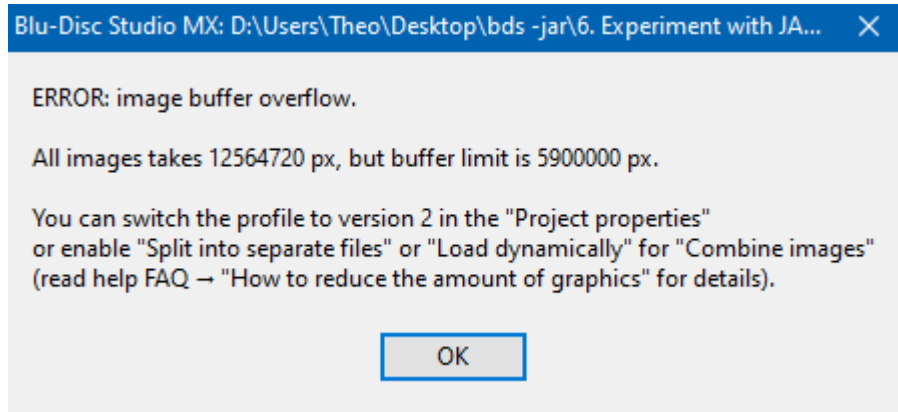
Pre-BDS V4.3

The JAR file contains all disc items that have graphical elements. And Java sets a limit to the amount of graphics that a JAR file can accommodate. BDS pre-V4.3 had everything in a single JAR file titled 00000).



This situation looks like the figure above: a disc with 3 (popup)menus A, B and C. All three inside the title 00000. This JAR includes also all graphical elements (all images and button images) of these menus. And in the project tree window you also specified as "First Play" which of the three menus would be displayed at disc startup.

If you exceeded the maximum allowed number of graphical pixels in BDS-J V1 (590 000) or V2 (7 900 000) you got an error message during muxing:



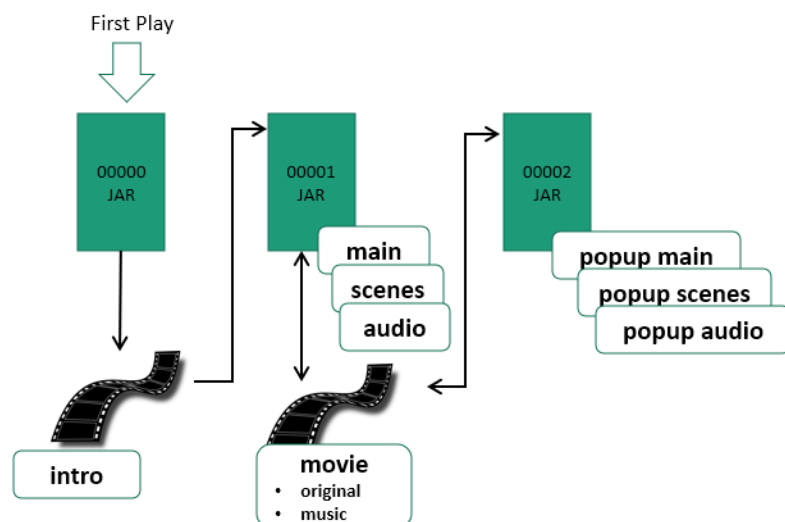
Following the advice in the message, you may postpone the maximum from being reached by using profile 2 instead of the default profile 1 or follow the advice to read the online help on what to do before going the path of multiple JAR titles provided as of BDS V4.3.

BDS V4.3 onwards

To elevate the graphical size restriction, BDS developed the use of multiple JARs where the user can decide what (popup)menus and their graphics go into which JAR file – as long as any single JAR does not exceed the total graphical size limit. Then the running of the disc may jump from one JAR file to the next, showing whatever menu is selected that is stored in that JAR file.

Because a disc needs to start somewhere, only one JAR has the indication “First Play” by which any set top player knows what JAR file (title) to start which and then move to whatever object is specified in “On JAR startup” for that particular JAR file.

The example in the BDS "Simple" set of projects puts all menus in JAR title 00001, all popup menus for the movie in 00002 and starts the disc with title 00000 which ends by running a short intro movie that, via its End Action, transfers control to the main menu in title 00001.



Warning: Keep in mind that if you use generated scenes menus or carousel menus, these are by default located in title 00000. You need to

reposition them into other titles if needed (as well as rename them if there are multiple movies each with their own scenes menus). The Simple example doubled them (scenes from the main menu and an identical set in the popup menu) repositioned them into JAR title 00001 for the main menu and 00002 for the popup menu. This is easily done via the context menu in the Project Tree for the Scenes menu: Clone All between Main and Popup and use the menu properties to assign the proper JAR title.

You can do the same for your own project and divide the menus over several JAR titles. There are a few things you need to keep in mind if you do:

- menus in a JAR file (title) can only invoke or call other menus in the same JAR file.
- If you try to invoke a menu in another JAR file, the button to achieve this must specify the JAR file to open
- Each JAR file has a menu to open with, specified in "On Jar Startup". If you want another menu, you must specify this explicitly, usually via a GPR register where a value is connected to a menu in that JAR file.
- any BDS element with graphics is stored entirely in a single JAR. This applies to menus, popup menus and their buttons and images
- any BDS element without graphics is not stored in a JAR: movies, playlists

The importance of JAR files as separate entities to movies may explain why when only changes are made in menus it suffices to use the "Compile JAR" button to replace the JAR files in the final disc project without having to re-mux all movies.

The "Simple" example neatly follows these rules as all menus are in one JAR title and the popups that belong to a movie in another. The two types of menus never invoke one another – popups belong to movies (or other popups), not other menus.

Important: the simulation of menus works on a per-JAR basis. If you have multiple JAR titles and want to test each one, you need to change the value of "On JAR startup" in the project tree. You need to cycle through all JAR titles one by one in that box and start the simulator for each JAR title again. In multi-jar projects the simulator does not give a project-wide simulation like standalone players will do running the disc. It only shows what happens inside a single JAR title. The "On JAR Startup" in the end must contain the JAR file (title) that will start when the authored disc is started. But during development you may need to change the value to test that JAR file.

Using multiple JAR titles

In the remainder of this chapter we will look at the steps to take to use multiple JAR titles. In short this consists of:

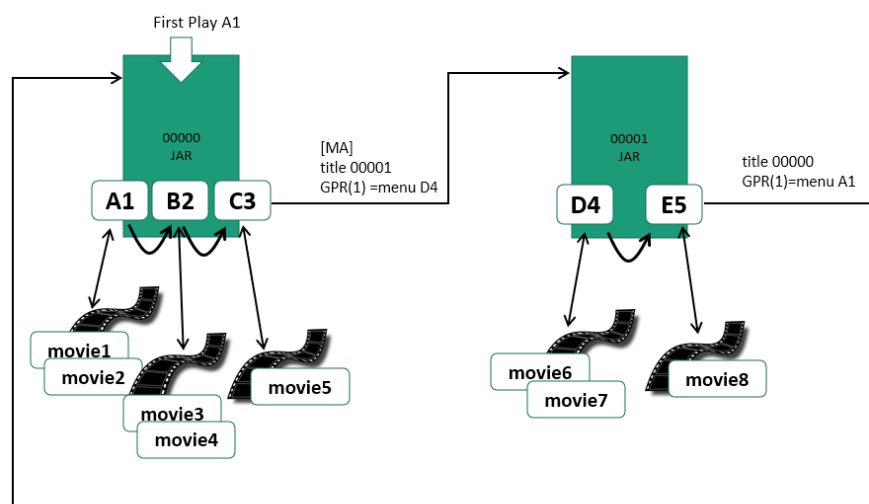
- create(and configure) all JAR titles needed
- assign each menu to a JAR title
- Any menu button that refers to a menu in another JAR title must specify in proper sequence:

- what menu to show (via a register - for GPR register usage, see Using Registers (simple programming) on page 287).
 - what JAR title to open (that contains the menu)
- This will require the use of a "Multi Action" action.
- The End Action of any movie can refer to any menu in the same JAR title that started it. If you want to open a menu in another JAR, you need to use a Multi Action again and specify in proper order:
 - what menu to show (via GPR register)
 - which JAR title to open that contains that menu (If no menu is specified, the "default" or main menu of the JAR title is opened)
 - The JAR opens and displays its object specified in "On JAR Startup". This is overruled if an action specifies to open another object in the JAR file. This is accomplished by reading a GPR register value associated with the menu to display (using a Switch action)
 - Only one JAR file will start when the disc is inserted. This is indicated in the JAR settings as "First Play"

The illustrations will use a project with only menus (no popups) called A1, B2, C3, D4 and E5. Each can run one or more movies but they all contain a button "Next menu" that cycles through the menus from A1 to E5 and back to A1. (Just forward navigation is shown here – you can make it as complicated as you wish of course).

Menus A1,B2 and C3 are in JAR title 00000. This title is set to start when the disc is inserted in a player, D4 and E5 are in JAR title 00001.

Hence moving from menu C3 to D4 a jump must be made from JAR file (title) 00000 to 00001. If nothing is specified, the "On JAR 00001 Startup" object is shown. Otherwise (or always, just to be sure) some GPR register value must indicate that menu D4 must be displayed. Similarly, moving from E5 in JAR 00001 back to A1 in JAR 00000 requires some attention.



Adding, deleting, moving JAR titles

Before being able to assign a (popup)menu to a JAR title, the title must exist. By default only title 00000 exists and will contain everything. This corresponds with the pre-V4.3 situation and is the preferred way to go if graphics size limitations do not hinder you.

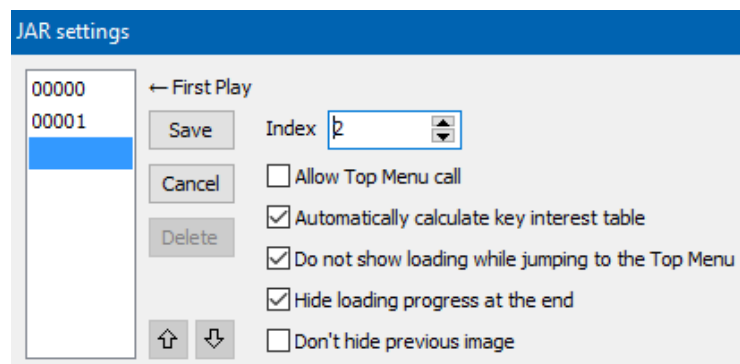
If you go the multi-JAR route, you need to be able to create, modify or delete JAR titles. All JAR titles work exactly the same, just some items can be configured to be unique for the specific JAR title:

- enable or disable certain features
- add animation (as text or images)
- add a specific Java written set of instructions to add to execute before the JAR code is executed

JAR titles can be shown or added by using the menu option Project > JAR Settings.

Adding JAR titles

To add a new JAR title, click the "Add" button on the JAR settings window. It must be given an unused index number (e.g. number *nnn* will translate to the title name 00*nnn*).



Save the new title by clicking on the "Save" button (the Add and Edit buttons changed into "Save" and "Cancel"). Configuration of the JAR can be done before saving it but can also be done later by editing the title. The "Cancel" just ignores everything done and the title is not added.

Deleting a JAR title

If you want to delete a particular JAR title, select it from the list of JAR titles shown in the JAR Settings window and click "Delete".

Moving JAR order

All JAR titles are shown in a list. The title on top of the list is the one that is always executed when a disc is inserted. It has the " ← First Play" arrow pointing to it.

By default it is the 00000 title, but if you use the up arrow/down arrow buttons in the JAR settings window you can modify the order of JAR titles. Using the up arrow button you can make title 00002 the top one and hence the one used on disc startup. The order of the JAR titles is unimportant – except for the top one that will be started on disc insertion because it contains the "First Play" indication.



Each JAR file when opened displays one object it contains, unless it is overruled to display something else. The item to show is specified in the “On JAR startup”. This value can be entered in the Project Tree but also in the Project > JAR Settings menu in the “On JAR Setup” box.

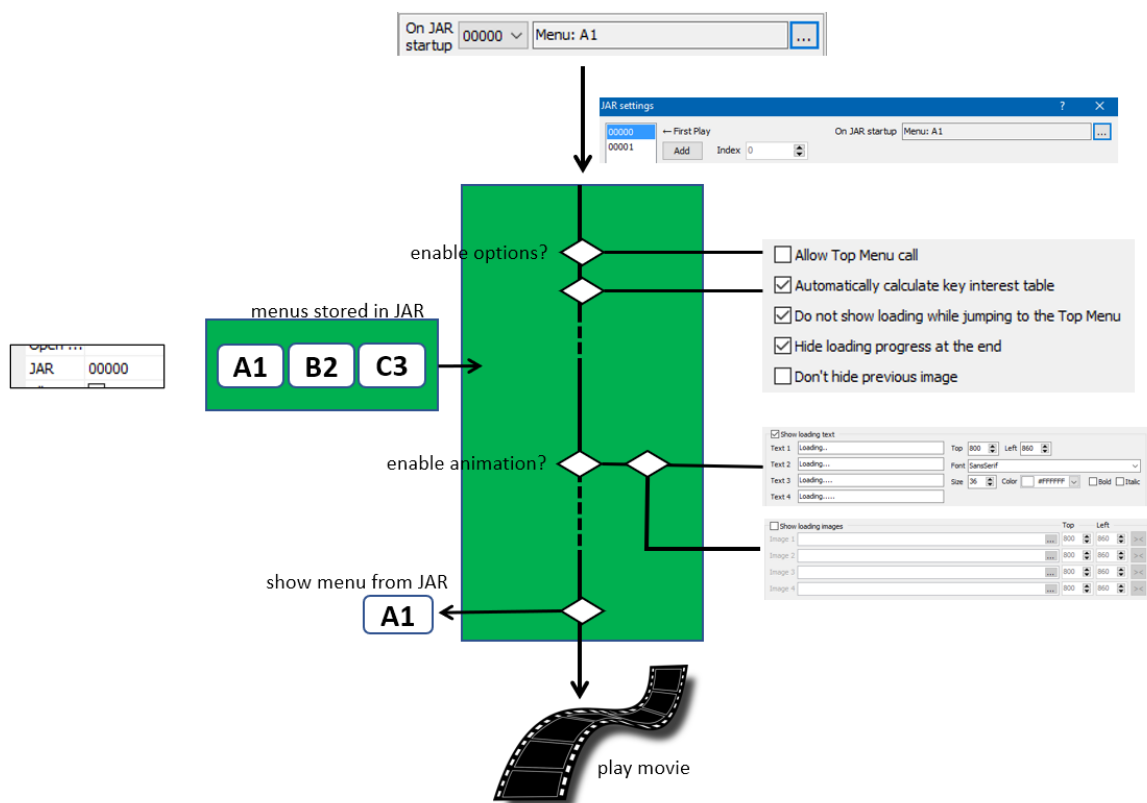
Configuring (editing) JAR titles

To configure the features of a JAR title, select the title name from the JAR Settings window and click on the "Edit" button. The "Add" and "Edit" buttons then change into a "Save" and "Cancel" button.

Each JAR file contains the same code to execute, but parts can be omitted or executed extra by setting values in the JAR settings window.

This is illustrated by the figure below. JAR 0000 will be the one started on disc insert (it has the “First Play” indication). And when it plays, it starts with the “On JAR 00000 Startup” menu A1.

We’ll have a closer look at JAR 00000. Its code is represented by the green block. Inside the code there are a few "hooks" where you enable/disable a few options (such as allowing the remote control to call for the Top Menu), add some animation in text or images. Menus are called A1,B2 and C3 to make it easier to remember by what GPR register number the menus A(1),B(2),C(3) are referred to.



Enable/Disable

The most important feature is whether the JAR code enable the remote control when pressing the "Top Menu" button. When not enabled, you can press the button but it is ignored. And remains to be ignored until another JAR title is executed that has this feature enabled.

Other items:

- *Automatically calculate key interest table* – influences the working of "prev/next/play/pause/forward/rewind" buttons on your remote. See the online help "interactive mode" for details
- *Do not show loading when jumping to Top Menu*. Exactly what it says: if this JAR title is executed to open the Top Menu (that must be in this JAR title) any animation should not occur.³⁶
- *Hide loading progress at end* – leaves the animation text or images visible once the First Play menu is loaded and shown.
- *Do not hide previous images* – animation images 1 to 4 are left on screen instead of only showing one at the time

Set animation

Important differences between JAR titles are the contents of either the "Show loading text" or "Show loading images". Here you can specify different texts and/or images to show on screen while the JAR title code executes. Each JAR can have different values for these. Each time you open the JAR title, the animation is shown first.

The screenshot shows a configuration window with two main sections. The first section, 'Show loading text', is checked and contains four text input fields labeled Text 1 through Text 4. Text 1 contains '1..', Text 2 contains '1..2..', Text 3 contains '1..2..3..', and Text 4 contains '1..2..3..4..'. To the right of these fields are controls for 'Top' (800), 'Left' (860), 'Font' (SansSerif), 'Size' (36), 'Color' (#FFFFFF), and checkboxes for 'Bold' and 'Italic'. The second section, 'Show loading images', is unchecked and contains four rows labeled Image 1 through Image 4. Each row has a file selection button (three dots) and a set of position controls for 'Top' and 'Left' (800 and 860 respectively) with arrows for adjustment.

- The text option allows 4 lines of text to be shown one after the other at the specified top/left position using the font and font size indicated. (the example shown above does a "1..", "1...2..", "1..2..3.." and "1..2..3..4.." after which the first text line is shown until the menu or movie to run with this title is available.
- If you use the images option, those images are shown in repeated succession as some sort of mini-movie animation (see Movie animation by frame animation (animated movies) on page 278).

³⁶ Top Menu may point to a menu in a different JAR file than the "First Play" containing JAR file

Normally only one of the 4 animation images is shown. But if you enabled "Don't hide previous image" the images remain on screen so they could build up an image (like progress bar)

Add script

You can precede the JAR code execution by some home-written Java code. It is often used to set some values in GPR registers. Java programming is not for the feeble of heart. See "Part 5: Advanced Operations and programming" on page 285 for more programming information.

The code is executed before the animation text or images are shown.

Assign menu to a JAR title

Each (popup)menu is located in a specific JAR file. You specify in which JAR file it is stored by setting the menu property "JAR". The JAR files must already exist (see previous section on how to create them).

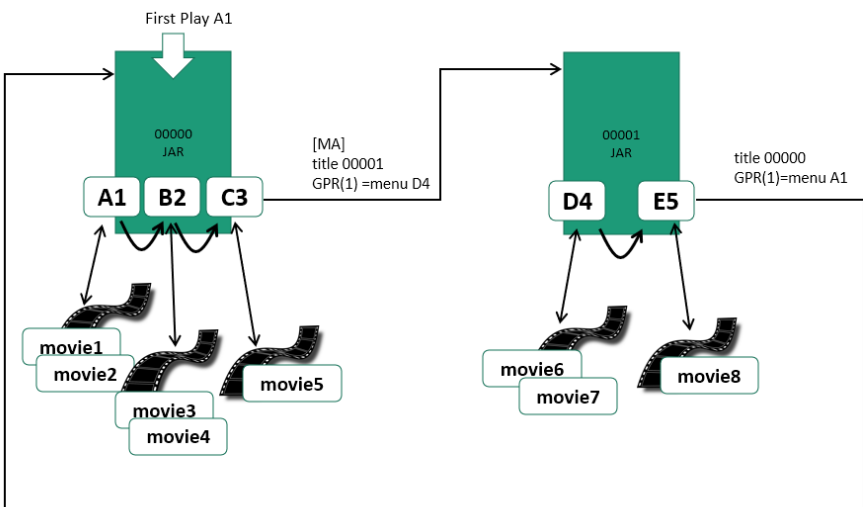
Open sound	
JAR	00000
Allow save state	00000
Streams	00001

Displaying menus in the same JAR file (title)

From one menu you can switch to any other menu without problems as long as it resides in the same JAR file. A button "Next Menu" in menu A1 might display menu C3 and select its "Play 1" button by setting its Press ENTER action accordingly:

OK Press ENTER	Menu: C3 [play 1]	standard
----------------	-------------------	----------

Menus A1 and C3 are in the same JAR file (00000).



Displaying menus in a different JAR file: Preparing the JAR jump

If you want to open a menu in a different JAR file, you need to specify that JAR file number. Each JAR file has a "On JAR Startup" property.

The menu specified there will display, if a movie is specified, that will start playing.

If you want another menu or movie in that JAR file to open, you need to explicitly specify this. Rather than depending on what is set as the “On JAR startup” value, you may decide to always be explicit.

Let’s see how to be explicit always: we want to jump to JAR file 00001 and open its D4 menu (which was also set as On Jar Startup).

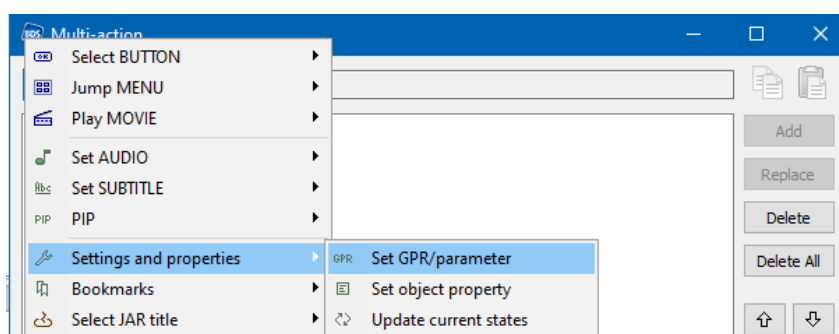
Suppose you want to open menu D4 in JAR file 00001 through a button in menu A1 which is in another JAR file 00000. In that case the button action for “Press ENTER” requires two actions in the right order³⁷:

- specify what menu to show
- specify what JAR title to open that contains that menu

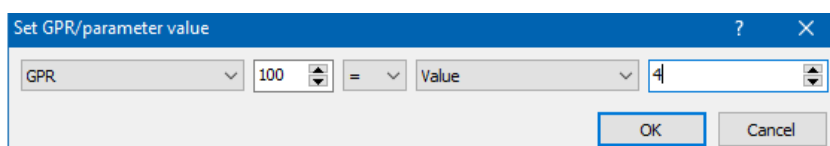
Because multiple actions are needed, the “Press ENTER” action uses a Multi-Action ([MA]) action. How to use Multi-actions is explained in another section of this part, Multi-action on page 203.

For the first action, you need a GPR register to specify the menu to open in the other JAR title. Any GPR register will do, but obviously its content should not be used by other parts of your project³⁸. In the example we use GPR 100 to specify the menu represented by a value in the register. Value 1 means menu A1, 2 means B2, 3 means C3 (all in JAR file 00000), 4 means D4 and 5 means E5 (both in JAR file 00001).

In the "Multi Action" window, click the "..." button and select the Settings and properties > Set GPR/parameter.



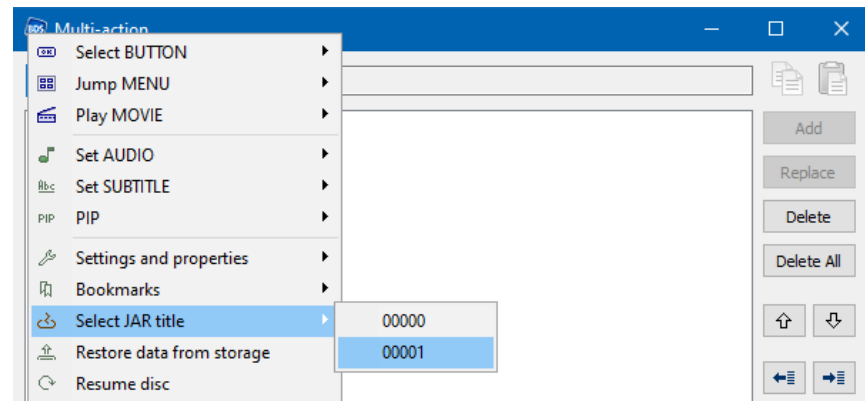
Next specify the menu C4 to open. So a GPR value of 4 must be set.



³⁷ If you open the JAR file first, it will automatically run the default “On JAR startup” menu or movie. This prevents you from specifying a different menu or movie to run.

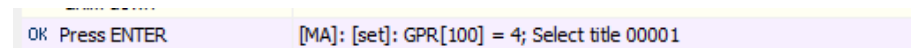
³⁸ usually in home-written User Defined Functions in Java. Registers and their values should have a unique meaning. See Using Registers (simple programming) on page 265.

The second action must set the JAR file (title) via "Select JAR title" action. This action is added by clicking on "..." again in the Multi-action window.



For our example, we need to select JAR file (title) 00001 as that one contains menu D4.

This multi-action is the "next menu" button action of the current menu:



Opening the menu

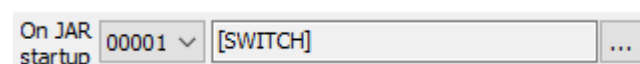
When the jump is made to the JAR file without further specification, the JAR file is opened and it runs what is specified in the "On JAR startup" property.

We explicitly set the item to run in GPR 100. Then we ordered JAR file 00001 to be opened.

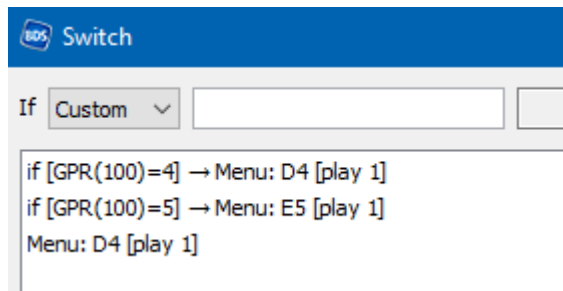
On opening, the JAR file runs its "On JAR startup" Switch statement. This retrieves the GPR 100 value and via a set of conditions finds that if the value of GPR 100 is equal to 4 then menu D4 must be shown.

If no valid value is stored in GPR 100, it should always fall back to one of the menus. Because we specified a Switch for "On JAR Startup" this Switch must end with a "catch all" for the JAR file to open if no condition is met. Usually this will be the "main menu" of the JAR file. In our case we take menu D4 and preselect its "play 1" button.

The opening Switch for JAR title 00001 looks like:



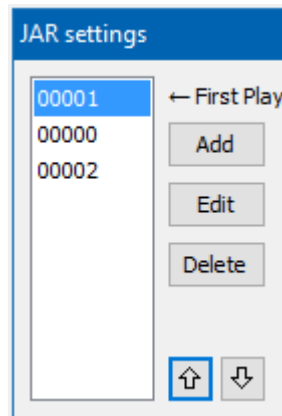
while the corresponding Switch statement is filled with a set of IF statements to select the right menu to display. If no correct value is found, the catch-all "main menu" for the title, set to be D4, is opened.



Special menus

First Play

The first JAR file (title) in the list of JAR files is the one that executes when the disc is inserted. Often it is 00000 JAR (title 00000).



But any of the available JAR titles can become the one to use as "First Play". Use the up arrow/down arrow button to move a specific title to the top of the list.

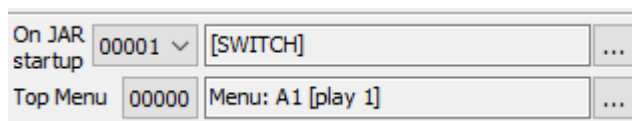
The order of the second and subsequent JAR titles is unimportant.

Every JAR file, when opened, starts with what is specified for it in "On JAR startup". When this is a Switch, all conditions in the Switch are tested. If none is matched, there must be a catch-all object set as startup.

Top Menu

The Top Menu is activated by the Top Menu remote control button. This button can be disabled by the last run JAR title if it was configured to disable that button. (Project > JAR Settings > JAR file checkbox "Allow Top Menu Call").

In the illustration below the disc starts up with JAR title 00001 (and its associated opening menu or movie defined in the Switch action) while when the Top Menu is pressed on the remote control, JAR title 00000 is started, displaying menu A1.



Project 6: Start movie when no menu choice is made

The Project Goal

Several bluray discs will automatically start the first movie if the viewer doesn't make a choice within a certain amount of time.

As of BDS V4.3 there are properties "Inactivity Timer" and "Inactivity action" not only for popup menus (that existed already) but also for main menus.

That makes creating a opening menu that automatically starts running a movie or displaying a different menu, quite easy and makes this project almost a no-brainer. Although this project will concern itself with a regular menu, the same works also for popup menus as they have the same properties "Inactivity Timer" and "Inactivity action" and behave the same way as regular menus.

This project will be based on Project 2's MenuBR but:

- The main menu can have a timeout and perform the action specified.

Timeout of (popup) menus

The concept of menu timeout

First play when a disc is started

When the disc starts, the JAR file that is set as FirstPlay, will start (Project > JAR Settings > first listed JAR file).

Usually it displays a menu or runs a movie – specified in the "On JAR startup" for that JAR file. Here, we assume a menu is opened.

Set timeout

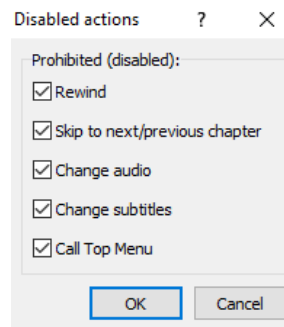
If this menu has its "Inactivity timeout" property set (in seconds) and the viewer takes no action, the menu times out after that time interval. Then it will invoke the Inactivity action.

Steps to take

1. The menu is setup as any menu: with a background menu movie (optionally an intro movie) and set of buttons and other menu objects such as menu texts.
2. Set the maximum time interval of inactivity in the menu property "Inactivity timeout" (number in seconds). The default value of 0 means that there is no timeout
3. Specify the action to take if a timeout occurs. This action can be a single action or Multi Action. The action may include the display of another menu or the start of running a movie.

User inhibited actions

For popup menus you can also disable buttons on the remote control. This only works for popup menus. Before opening the popup menu, the "Disabled actions" of the movie can be applied:



Project steps to take

We will create a project based on the Project 2 “MenuBR” that has one menu from which two movies are started. We will change this menu into a popup menu with (background) movie.

If you want to create this project yourself, the source files can be found in the .zip file mentioned in Appendix A: The user’s guide source files.

Ready to run

For a complete project, perform these steps:

1. Copy the example \Sources\Projects\AutostartBR folder tree to your BDS project tree (e.g. J:\BDS Projects\AutostartBR)
2. Copy the Australia* and Coasts* files from \Sources\Project objects\movies\demuxed into the project’s \films folder
3. Copy the “black silent 5s 1920x1080 25 fps” .264 and .ac3 files from \Sources\Project objects\movies\demuxed into the project’s \films folder
4. Copy all subtitles from \Sources\Project objects\movies\subtitles to the project’s \films folder
5. Double click on the \AutostartBR\project.bdmd file to start BDS and open the project
6. Click on the “mux” button and create the disc image.

Build your own

To create a simple project that performs this feature (starting “Coasts” as default movie to play), take the following steps:

1. Copy the project tree \BDS Projects\MenuBR to \BDS Projects\AutostartBR to create a new project. You may skip the \Output folder.
2. Copy the black silent movie from \Sources\Project objects\movies\demuxed\black silent 5s 1920x1080 25 fps.264 and .ac3 to the \AutostartBR\films folder of the new project.
3. Adjust the menu “main menu”:
 - Set Inactivity Timeout to 10 seconds
 - Set Inactivity Action to “play movie Coasts”
4. Mux the project with results in \output

Variation: changing menu (Carousel)

Just like you can auto-start a movie, you can also rotate between menus (show menu A > B > C > D > A > B... etc.: a carousel). Note that a carousel of menus (changing entire menus) is different from a menu carousel (sliding buttons in one visible menu).

For this you need to consider:

- Design as many menus as you want to rotate between
- Each menu has its “Inactivity timer” set
- Each menu has its “Inactivity action” set to open another menu
 - In case the menu resides in a different JAR (called “title” number 0000n), an action may be needed to start the JAR (Inactivity action = “Select title 0000n”. It then opens the default “On JAR startup” object. If you don’t want that you also need to indicate the menu to use (through GPR number) and use a Multi Action (see Assign menu to a JAR title on page 215)

An example project (without actual movies) is provided as \Projects\CarouselMenuBR project. It only works in simulation mode until you add some real movies for background and actual movies. But the timeout is simulated.

Variation: popup with rescaled smaller movie

In an earlier project you encountered the popup menu. It can be called when a movie plays and it has its “Popup menu” property set. The movie plays full screen, the popup menu is displayed on top of it.

An interesting alternative way to use a popup menu is one where the popup menu takes most of the screen, leaving a hole through which the movie plays. A reversal of roles so to speak.

This is used in a BDS example for movie “Valerian” (“Ravian” for the Dutch) where the movie continues in the center part of the screen at 1/2 of its original width and height (really at 1/4th its original area).



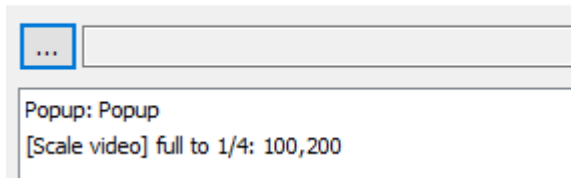
To create something similar you use the multi-action action.

In the “popup menu” property of the movie you specify a multi-action to:

Popup menu [MA]: Popup: Popup; [Scale video] full to 1/4: 100,200

- Open the popup menu (with a hole in its canvas) (and its animation)
- Rescale and position the main movie to fit the hole (in the example: at 1/4th size of its width and height)

BDS Multi-action

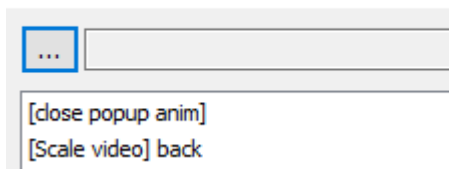


And the end, when the popup is removed, you do the reverse. This time you need to set the multi-action in the “Inactivity Action” of the popup menu. (the Inactivity Action has been made in BDS to rescale to full size automatically).

Inactivity action [MA]: [close popup anim]; [Scale video] back

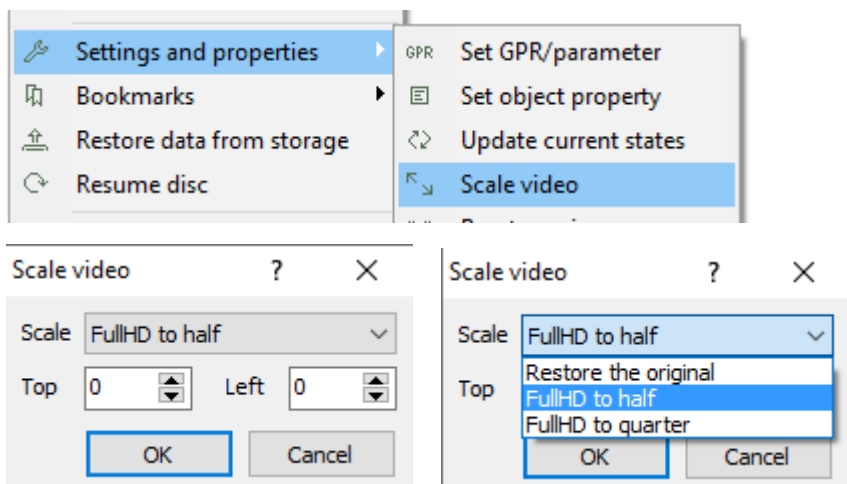
- Close the popup menu (and its animation)
- Resize the movie to full size

BDS Multi-action



Reducing (or restoring) the movie size is done via the “Settings and properties” > Scale video action – usually as part of a multi-action.

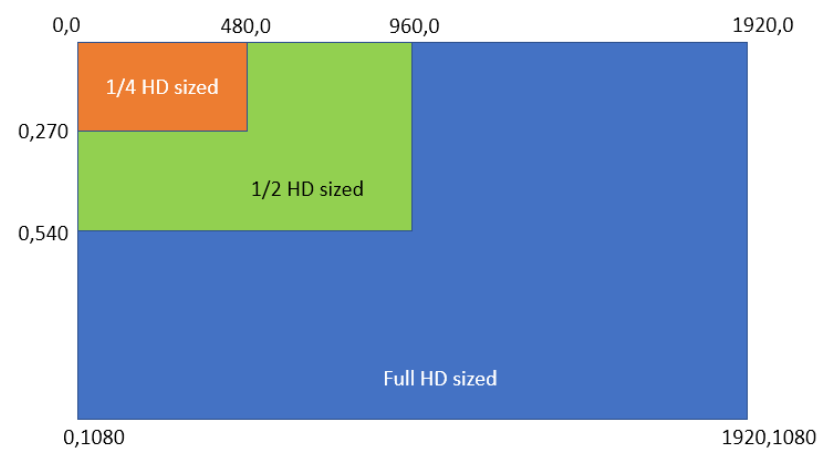
This opens a small menu where you can set the video size to Full HD, half or quarter HD.



Using the “Top” and “Left” you can specify where you want the reduced movie to show (see future section “Screen positions and displacement” on page 251 for details on positioning).

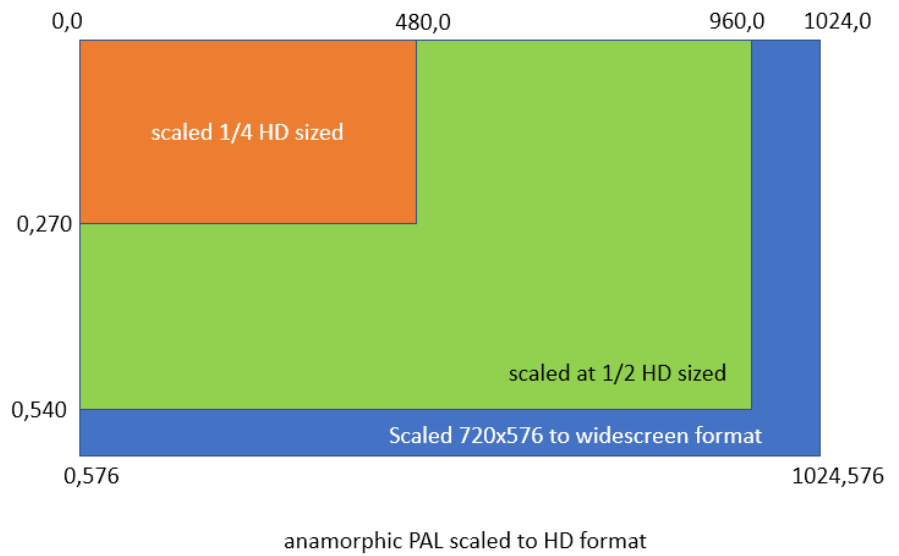
Size	Height (pixels)	Width (pixels)
Full HD	1080	1920
Half HD	540	960
Quarter HD	270	480

Note that the relation is to full HD, 1920x1080 pixels. Unlike you might think, the “half” and “quarter” refer to the lengths of the sides – not the surface area! In practice this means that a ½ HD sized video takes up half the height of the screen, which is ¼ of the screen surface area. The ¼ HD sized video only 1/16th part of it.



It all works fine for material in HD resolution. Using lower resolutions or movies grabbed from DVD (SD resolution), the results are somewhat unexpected. This has to do with the fact that an SD movie of 720x576 pixels (in PAL) is much smaller than an HD movie of 1920x1080 pixels. The scaling algorithm seems to use the HD numbers and applies them to SD also. This means that "Half sized" has a height of ½ x 1080 = 540 pixels. And that is what you see when SD is used. Half sized is not ½ x 576 pixels but ½ x 1080 =540 pixels. It is therefore only marginally smaller than full sized SD.

Size widescreen aspect ratio 16:9 SD	Height (pixels)	Width (pixels)
Full SD	576 (PAL) 480 (NTSC)	1024 (PAL) 854 (NTSC)
Half SD	540	960
Quarter SD	270	480



It will be clear that for SD resolution movies the $\frac{1}{2}$ HD will not fit 4 movie images in a single screen and $\frac{1}{4}$ HD do not fit 16 movie images.

Project 7: Restartable disc (Restore/Resume)

The project goal

This chapter discusses how to author a bluray disc that can resume its playback from where it left off last time. This feature helps to continue watching a disc when you interrupted watching earlier.

Note what is understood about “a movie is interrupted”. It is considered interrupted and hence resumable if:

- it plays and the disc is ejected during playback
- its play is interrupted by pressing the “Top Menu”, “Title Menu” or “Disc Menu” buttons resulting in the menu to be shown
- its play is interrupted by a player power off

It is important to realize that especially the second reason may lead to confusion: a menu is shown but on resume the movie that was interrupted for the menu, will resume – you might have expected the menu to show. It doesn’t.

The state of progress of the viewing is stored inside the flash memory of the bluray disc player. BDS writes and updates a so called “Storage File” (name can be set in Project > Project Properties > General tab) in that flash memory. Each BDS project creates a unique project number by which the contents of the disc is recognized. This project-unique number plus the current run time of the movie is stored in that storage file. The run time is reset when the movie completes normally. When a disc is inserted, the player reads the storage file, checks if the project ID is the same as for the current disc and if so, if there is a non-zero running time. That indicates the previous run was interrupted and can potentially be resumed.

Because the flash memory of the player has limited storage capacity, only the most recently watched and interrupted disc states can be stored and used to resume playing when the disc is re-inserted. Older plays of discs may get “forgotten” and even if interrupted at the time, such a disc may start from the beginning.

This project

- will show you how to create a disc that can be resumed.

To focus on this one aspect, we reuse the results of Project 2, the “MenuBR” project.

If you want to create this project yourself, the source files can be found in the .zip file mentioned in Appendix A: The user’s guide source files.

How is a disc unique?

Each bluray disc project is provided with a unique number through the authoring software. This is stored in the flash memory of the player. The software run by the playing disc can retrieve values from this

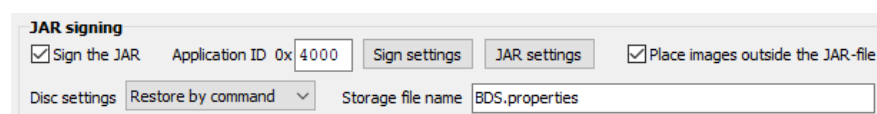
storage and determine whether the current disc contains the project ID in the storage file and was interrupted.³⁹

Because the unique number is attached to the authored disc, all copies of that disc have the same number. So you may interrupt disc copy 1 and insert disc copy 5 (identical to copy 1) and the player will recognize that that title was interrupted. That's also how commercial discs work. You may interrupt, say, a Star Wars IV disc. When you insert another identical copy of Star Wars IV disc the player will think that disc was interrupted.

How does the authoring program compose this unique number? The number builds from:

- Organization ID
- Application ID
- Storage file name

All three elements are found in Project > Project Properties under the "Jar Signing" section.



To create a new unique number you need to change at least one of these three values. It won't help burning the same project on another bluray disc (either R or RW variant) or run it from the same muxed output folder.

Why is it not unique?

Given the three parts from which a disc is identified, it may not always work, as without any modification, all BDS created discs have the same identification. Which leads to the odd facts that:

- a remux of the same disc is not recognized as new. If the previous mux was interrupted on playback, the new mux will think it was interrupted and resumes at the same spot. Even if it has never been played before. This is true for software players (such as PowerDVD) as well as for hardware set top players.
- Totally different discs on topic A or topic B will not be distinguished. If A is interrupted and B inserted, B will resume playing from the playlist movie and its last playing time. They have the same identification so as far as the player is concerned, it is the same disc. Even if it is not.

³⁹ The storage file is named "BDS.properties" (name can be set in Project > Project Settings > General tab, and it is located in a special folder named after the used JAR signed Organization ID (e.g. 7fff4321) + Application ID (e.g. 4000). So as an example, a storage file might be at 7fff4321\4000\BDS.properties .

So the moral of the story: if you really want unique discs, change one of the three parts from which the ident is built. The most obvious choice is to increment or decrement the Application ID.

You might think BDS should assign a unique number to each mux, based on the time of muxing. Or a Global Unique ID (GUID). But BDA stipulates the disc ID to be created using the aforementioned elements. So you need to be inventive yourself to differentiate between your various project results.

Resume disc

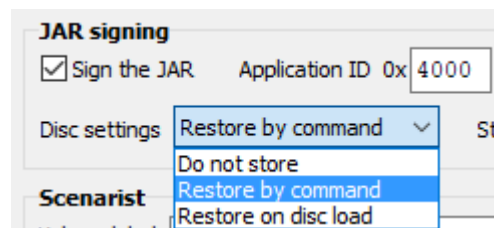
In order to resume the playback of a disc, data is needed of where the playback stopped last time.

This data about the interruption (which disc and where in the movie did it get interrupted) is stored in the bluray player memory. That means that the bluray disc programming must actively store and update its current play position. That can only be done when the disc runs a program that performs the update.

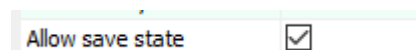
This program runs when it is told to run by BDS by setting the right checks in checkboxes. And then that program uses the unique project identification to make the player aware what disc (project) was interrupted.

To resume a disc you need to set three things in your project:

- In Project>Project Properties>General tab under the JAR Signing you need to provide a unique set of Organisation ID (though JAR Signing), Application ID and Storage File name to create a unique project ID for the player to remember
- For the resume program to run and use the project ID, you need to select the “Disc settings” to either “Restore by command” (restore conditionally) or “Restore on disc load” (Resume always). Both manners are discussed further down.



- All movies and menus must have their "Allow save state" checked.



Resume Always

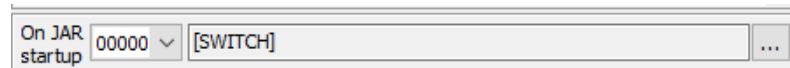
For a disc to resume without questioning the user:

- Set Project > Project Properties – set Disc settings = “Restore on disc load”

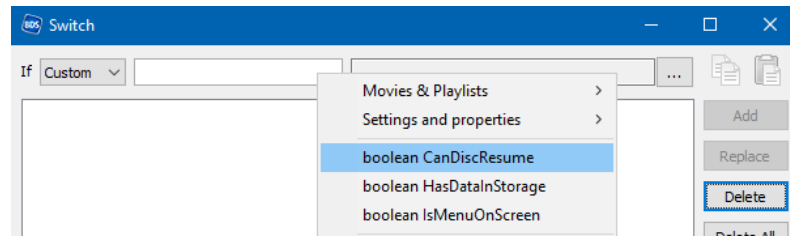
Note: if you forget this and set it to “Restore on command” upon restore it starts playing the interrupted movie in real time upto the

time of interruption without showing it. Once that point is reached, it restarts so you never see anything.

- set the "On JAR Startup" box to an exclusive Switch statement (which stops executing as soon as the first Switch statement that is true has been found).⁴⁰

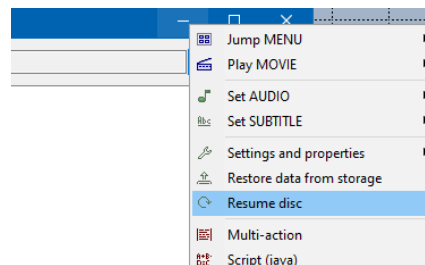


The Switch statement is set up as follows.

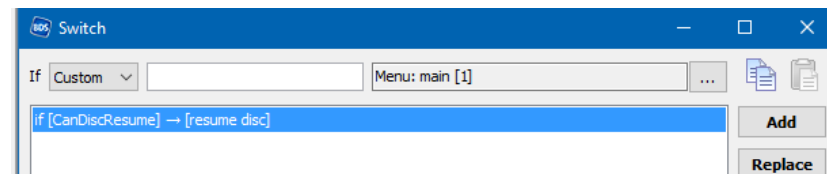


Select the "Custom" IF statement. Pressing right mouse button while hovering on the blank text box shows a dropdown menu from which you select the "CanDiscResume" option.

As action (clicking the "..." button), select the "Resume Disc".



And finally click the "Add" button to add the switch action to the list. In case the disc cannot resume, the disc needs to open the menu it would otherwise open. That is the second Switch statement (keep the "If custom" but leave the condition blank and select the menu and button to start with and click "Add").



Resume conditionally

Resume conditionally means you allow the viewer to decide whether or not to resume.

This requires that a menu is shown when the disc resumes, giving the user a choice to start from the beginning or to resume. One menu button will open the disc as if no resume exists, the other button will issue the "Resume disc" action.

⁴⁰ As escape, you may also want to enable Top Menu for the JAR file (Project > JAR settings) to start with the opening menu in case you decide to cancel the resume

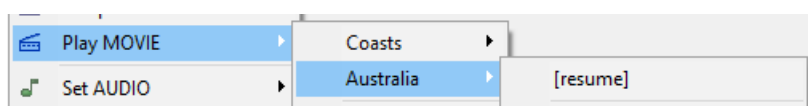
You need to

- Set Project > Project Properties – set Disc setting = “Restore by command”
- Add Switch statement to On JAR Setup with one exclusive action for each movie to open the resume menu with the resume button as default
- Add a Switch statement to the resume button on the resume menu with an exclusive action for each movie to resume itself

See section "Build your own – Resume disc conditionally" at the end of this chapter for details.

Resume a movie

There is another type of resume not related to a power interruption or disc eject: the viewer of a movie may get side-tracked through a popup menu to inspect some background information or other movie. After finishing this “side viewing” movie its “End Action” can return to the point where the main movie was interrupted though “Play Movie > *movie name* > [resume]”



Resume and BD-J

There is some controversy about the resume feature and bluray discs that have their navigation programmed in Java – like BDS does.

Discs that do not use Java use HDMV⁴¹ and have no problem using resume of an interrupted title. The Java code seems to disable the player’s resume feature in favour of “doing it in Java”. But apparently it is not always successful. And what was a no-brainer for most DVD players seems a huge step backwards on bluray where Java was introduced to allow for “bonus” features, BD-Live “live-link” to connect to the internet and download updates on films on disc.

An alternative to resume is to “bookmark” a disc: actively state the playback position before ejecting the disc.

Project steps to take

To build this project yourself, start with creating a new project “ResumeBR” and copy the subfolders \files and \original sources from the project 2 “MenuBR” folder. After import of the main menu, create all buttons and set up the proper navigation and Press Enter actions as was done in Project 2.

Ready to run

To have a completed project to build directly, you need to take the following steps:

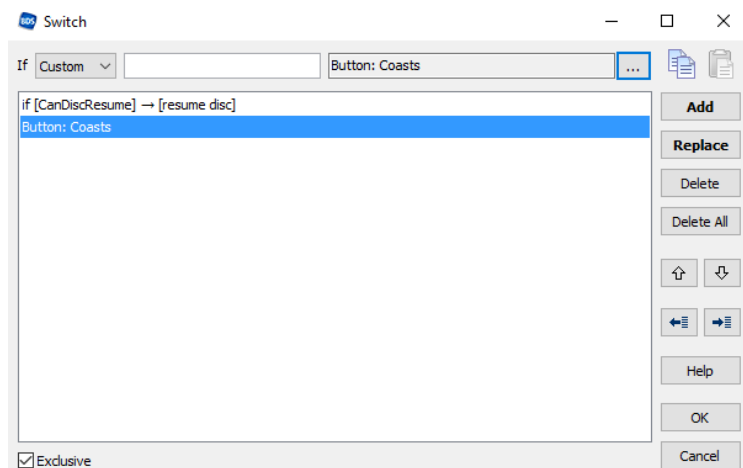
⁴¹ See HDMV versus BD-J playback on page 431 for some more details on differences of BD-J and HDMV

1. Copy the example \Sources\Projects\ResumeBR folder tree to your BDS project tree (e.g. J:\BDS Projects\ResumeBR). This includes a special folder \BDS fonts with fonts used in the Resume menu.
2. Copy the Australia* and Coasts* files from \Sources\Project objects\movies\demuxed into the project's \films folder
3. Copy black silent 5s 1920x1080 25 fps.264 and .ac3 files from \Sources\Project objects\movies\demuxed into the project's \films folder
4. Double click on the \ResumeBR\project-unconditional.bdmd or project-conditional.bdmd file to start BDS and open the project
5. Click on the "mux" button and create the disc image (in \Output-unconditional or \output-conditional respectively)

Build your own – Resume disc always

The following steps are executed:

1. Open the project (double click its project.bdmd file that is made as part of the project creation. You may want to rename (save as) it as "project-unconditional.bdmd")
2. Ensure the JAR is signed and disc is set to "Restore by command" (In Project>Project Settings>General tab).⁴²
3. Movies and menus must have "Allow save state" checked
4. Set the On JAR Startup action to Switch. The Switch is exclusive (stops looking for actions after the first condition that evaluates to "true" – ensure the "Exclusive" check is set). The Switch entries are:
 - a. If disc can resume, resume it
 - b. In all other cases, open the opening menu or movie



5. Set output folder (Project > Project Properties) to \ResumeBR\project-unconditional and remux.

⁴² You may specify a different Application ID and/or Storage File Name but this is only needed during testing. A new unique project ID is build from these properties and it will differ (not suffer) from interruptions of earlier project output.

That's it.

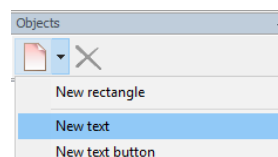
Build your own – Resume disc conditionally

A conditional resume depends on the response of the viewer to the question whether to resume or not. So, an extra menu is required.

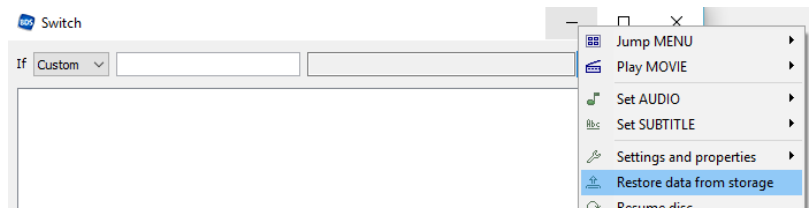
Then the following steps are executed:

1. You may re-use the project made for unconditional resume, but save the project as “project-conditional.bdmd” and work in this newly saved project (that way you do not delete the work done earlier)
2. Ensure the JAR is signed and disc is set to “restore by command” (In Project>Project Settings>General tab)
3. All menus and movies must have “Allow save state” checked
4. Create a Resume menu with two buttons for “Resume playback” and “Play from beginning”. This menu should *not* have the “Allow save state” checkbox set (to avoid a resume of a resume).

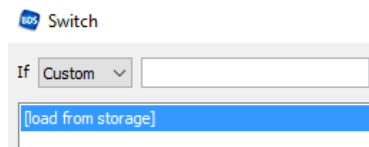
Rather than using Photoshop, you can try the in-line BDS method of making a simple menu by clicking on the dropdown menu of the “Add” button in the Objects window (See section “Menus made within BDS” on page 90) and create text and buttons this way. The fonts to use for the text need to be added first (Project > Used Fonts) from \Sources\BDS Fonts\Arial.ttf (which may be copied to a project local folder such as J:\BDS Projects\ResumeBR\BDS Fonts)



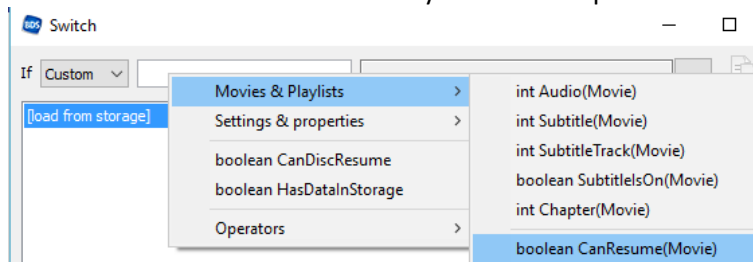
5. In the Project Tree, assign a Switch action to the “On JAR Startup” action. This Switch (see Switch section on page 203) performs two conditions:
 - a. Load the storage data held by the player
 - b. If disc can be resumed, show the Resume menu
 - c. If not, show the opening menu (which may have an intro movie)
6. For (a) to work, the Switch window (important: with “Exclusive” checked) must use the “Custom” option. Leave the condition box empty (i.e. it will always execute. There is no earlier conditional statement that can become true, so the conditions that follow it will be executed until one evaluates to “true”).
As action for this condition, we load the data by “Restore data from storage” offered by a right click on the action text box. This results in the [load from Storage] action.



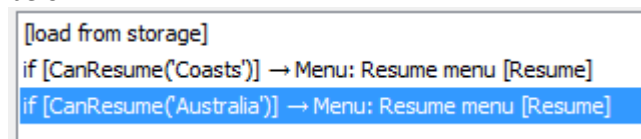
Click “Add” to add it to the list of Switch actions.



7. For (b) to work, add two additional conditions, each checking on whether the movie “Coasts” or “Australia” was interrupted. If so, open the Resume menu with default button “Resume”. The condition on the movie is obtained by right click on the condition and follow the “Movie & Playlists” menu option.

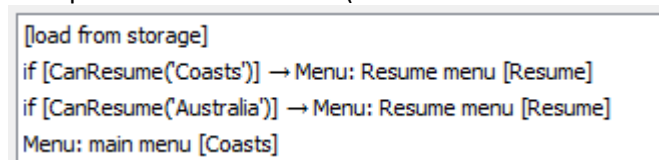


Select the “CanResume(Movie)” option and manually add “Coasts” or “Australia” as title (the names of the movie placeholders in the Movies list of the Project Tree). As action select the Jump to Menu>Resume menu (button “Resume” as default). The three Switch actions are listed in the figure below.



Click “Add” to add it to the list of Switch actions.

8. For (c) to work a final step must be made: an unconditional “Jump Menu” > Main menu (with “Coasts” as default button)

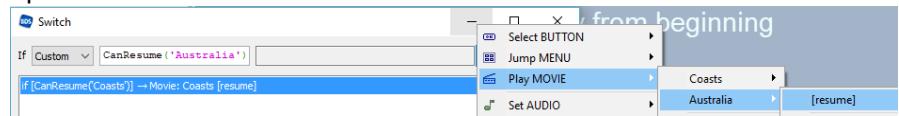


Click “Add” to add it to the list of Switch actions.

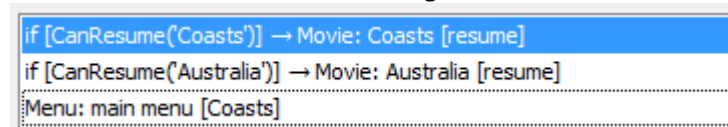
Click “OK” to save the Switch as “On JAR Startup” action.

9. In the Resume menu, connect the “Press Enter” button of the “Play from beginning” with jumping to the main menu.
10. In the Resume menu, connect the “Press Enter” button of the Resume button with a Switch action (again, with “Exclusive” checked) to find out what movie to resume playing. The Switch action does multiple checks:
 - a. If a movie is interrupted, resume it
 - b. In all other cases show the main menu

11. If there are multiple movies, each movie state must be interrogated to resume the correct movie. Again the “Movies & Playlists” > CanResume(Movie) condition is used.
12. For the action to take, use [resume] of the Play movie option.



The Switch conditions for two movies “Coasts” and “Australia” will then look like the figure below.



Don’t forget to click “Add” to add the condition to the Switch and finally to click “OK” to add the switch to the Resume button’s “Press Enter” action.

If neither movie can be resumed, the main menu will show just like the “Play from beginning” button. This third action should normally not happen since one of the two movies already indicated to be resumable in the FirstPlay Switch. But if the disc is modified and a third movie added (without updating the Switch conditions) it will happen.

13. Specify output folder \ResumeBR\output-conditional and remux the project.

Project 8: Bookmarks

What are bookmarks

Bookmarks are what they are in books too: markers to remember a location in a book. Replace “book” by “movie” and you get the idea. Just like books, bookmarks act the same way as chapter marks. Big difference: the viewer determines where bookmarks go, the disc author has decided on where chapters went.

BDS bookmarks can be used to remember a position in a movie. Each movie has its own list of bookmarks. There are actions to

- Create a bookmark
- Select first bookmark
- Select last bookmark (“last” in position, not “latest” in time)
- Select next bookmark
- Select previous bookmark
- Delete currently selected bookmark
- Play movie from currently selected bookmark

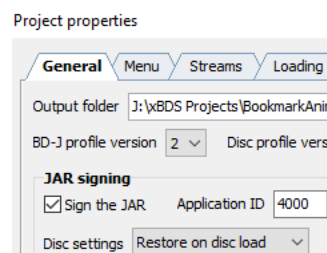
There is no “delete all bookmarks” action – this must be programmed yourself in Java. Though programming can be difficult, this one is easily done – see section “Project 11: Remove all bookmarks” on page 315.

Be aware that bookmark usage may lead to different experiences with the viewer depending on what player he uses and also on how a bluray disc is programmed for using bookmarks. It is not a feature with a universal, common behaviour.

Bookmarks are stored in the player’s storage by the Java code of the disc. That code can also read back that information, provided it is still there. This depends very much on the player’s storage capacity. In the worst-case bookmarks are only remembered as long as the disc is not ejected or until the player is powered off (this is different from “stand by” and seems to apply to Samsung players, whereas Sony stores it in non-volatile memory between power-offs).

In a slightly better case, one or more other discs can be played before you want to reinsert the first disc and its bookmarks are still available.

Bookmarks, even when generated entirely through menu selections, result in Java code and must be saved as part of the project. Hence in Project>Project Properties>General tab you need to check “Sign the JAR”. Also, the “Disc Settings” must be set to “Restore on Disc load”.



The viewer can set bookmarks in a movie (not in a menu, for I hope obvious reasons) if you have a button assigned to this action through its Press ENTER property. (There is no standard, but the green button seems mostly used for this).

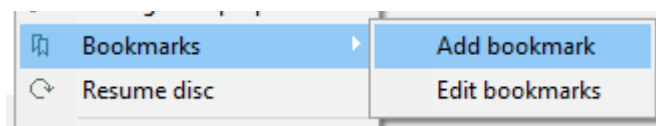
Viewers often use it as replacement for the “resume” feature: set a bookmark before leaving the movie and then at some future time, resume playing from that bookmark once it is selected from a menu of set bookmarks.

Some discs have elaborate code for showing bookmarks. One such scenario is starting the movie, press “up arrow” to display a popup menu with a timeline with all bookmarks shown as small triangles along the timeline. Pressing the “next chapter” button moves you to the next bookmark (much like a next chapter would do without the timeline popup) and at the bookmark of your choice you press “OK” and play from there. The bookmark menu disappears when the “down arrow” is pressed.

None of this comes “out of the box” with BDS. The timeline requires easy but still substantial programming as can be seen in Project 12: Add a popup Timeline on page 327 . Adding bookmarks on top of this is shown in Project 13: Popup Timeline with bookmarks on page 332.

Add a bookmark

A bookmark can be added when a movie’s button is set to execute the action “>” Bookmarks > “Add Bookmark”. Bookmarks can only be added to the currently playing movie.

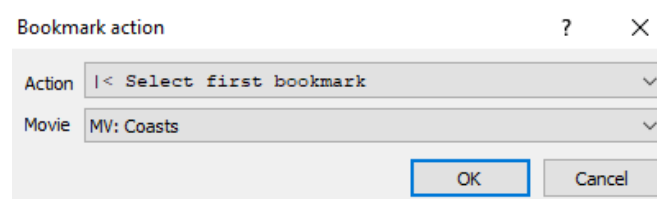


Some remote controls have a “Marker” or “Bookmark” button that is supposed to do this (directly in the hardware of the set top player) but BDS has no remote control button by this name as this button is not available on all players.

If there is any common use of a button to allow “Add bookmark” it appears to be the green remote control button.

Select bookmark

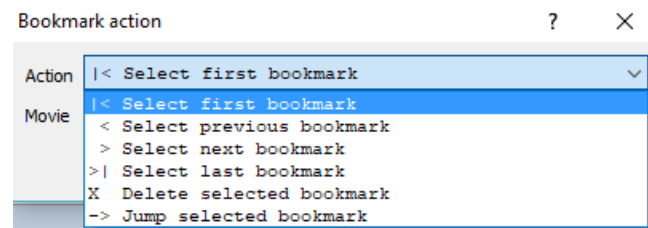
Selecting a bookmarked position is done through the action “>” Bookmarks > Edit Bookmark. This opens the bookmark window.



Here you indicate in which movie the bookmark must be found and which bookmark in that movie (notice that “Movie” is not filled with

the movie whose properties you are modifying: it takes the first movie on the list of movies! Which is slightly annoying).

The “action” choice may be confusing.



You must know the difference between “Last” (in position) and “Latest” (in time). You can revisit the last bookmark in a movie, not the one added last (i.e. the latest in time). Therefore, a bookmark at 10 minutes into a movie can be the last one, even if moments ago you added one at 5 minutes into the movie.

Delete current bookmark

A bookmark can be deleted, once it has been selected first. The BDS procedure is the same as selecting a bookmark (perhaps repeatedly: first / next / next) before using the “Delete selected bookmark” action.

This may require a button with action “select next bookmark” and another to perform “Delete selected bookmark”.

Delete all bookmarks

There is no BDS action that deletes all bookmarks. However, this action can be accomplished with a small bit of Java coding with a minimal change to what the script code for deleting a single bookmark requires. See section “Project 11: Remove all bookmarks” on page 315.

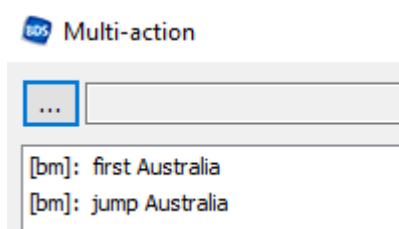
Play from bookmark

Play from first bookmark

Playing a movie starting at a bookmark requires the use of a multi-action.

- The first action is to select the wanted (first) bookmark
- The second action is to jump to the selected bookmark (and play from there on).

The figure below shows a button programmed to start from the first bookmark in “Australia”. The first action is a selection of the Bookmarks > Edit Bookmark > “Select first bookmark”.



Play from any bookmark

To start from a particular bookmark, you need to be able to display all bookmarks and use the Bookmarks > Edit > Select next bookmark repeatedly to get to the bookmark of choice. To play it, you “Jump to selected bookmark”.

This may require a small (popup) menu to show the time of the bookmark with three buttons: show next bookmark, show previous bookmark, play current bookmark.

Project steps to take

This project will reuse the simple result of project 2 (MenuBR) with only a single audio track and no subtitles to keep it simple. It has added functionality on some remote-control buttons:

- Red button adds a bookmark to the running movie
- Yellow button returns to the main menu
- Blue button erases all bookmarks of the running movie
- Both movies “Coasts” and “Australia” have their own popup menu (hence the yellow button to return to the main menu):
 - Bookmark popup Coasts
 - Bookmark popup Australia
- Each of these popup menus shows on opening the time of the first bookmark. You can move through the bookmarks using the “previous” and “next” buttons and start playing from the currently selected bookmark pressing the “play” button.

Ready to run

To have a completed project to build directly, you need to take the following steps:

1. Copy the example \Sources\Projects\BookmarkBR folder tree to your BDS project tree (e.g. J:\BDS Projects\BookmarkBR). This includes a special folder \BDS fonts with fonts used in the popup menu.
2. Copy the Australia_movie.264, Australia_live.ac3, Coasts_movie.264 and Coasts_live.ac3, menu.264 and menu.ac3 files from \Sources\Project objects\movies\demuxed into the project's \films folder.
3. Double click on the \BookmarkBR\project.bdmd file to start BDS and open the project
4. Click on the “mux” button and create the disc image (in \Output)
5. Experiment

Build your own

We base this project on the MenuBR project made earlier.

1. Copy the \Projects\MenuBR folder to the current BDS projects tree and rename it to BookmarkBR (e.g. J:\BDS Projects\BookmarkBR)

2. Copy the movies
\Sources\movies\demuxed\Australia_movie.264 ,
Australia_live.ac3, Coasts_movie.264, Coasts_live.ac3,
menu.264 and menu.ac3 to the project's \films folder (e.g.
J:\BDS Projects\BookmarkBR\films)
3. Copy the \Sources\project objects\original sources\bookmark
popup.psd to the project's \original sources folder (e.g.
J:\BDS Projects\BookmarkBR\original sources)
4. Copy the \Sources\BDS fonts folder to the project folder (e.g.
J:\BDS Projects\BookmarkBR\BDS Fonts)
5. Copy some buttons from the \Sources\project objects\buttons
folder to the local project's folder \buttons. We need ArrowN,
ArrowS, "left/right arrow button green small" and "play button
green small".
6. Open the project BookmarkBR with BDS.
7. Add the fonts to the project for use later when a text object is
created. Project > Used Fonts and click "Add" and navigate to
the project's \BDS Fonts folder and select the Arial font.

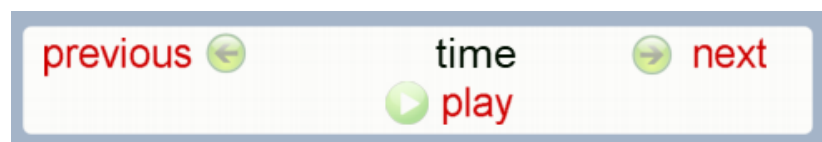
Add remote-control buttons to the movie

8. Add some remote button functionality to the "Australia"
movie:
 - a. The "Red" button gets a simple action to add a
bookmark (">" Bookmarks > Add bookmark)
 - b. The "Yellow" button returns us to the main menu (">"
Jump MENU > main menu)
 - c. The "Blue" button erases all bookmarks from the
movie. This is not provided by BDS and needs to be
programmed (see section "Project 11: Remove all
bookmarks" on page 315 for details).
Add a script as action (">" Script (Java)) and add the
following Java code verbatim:


```
while
(manager.getBookmarksCount('Australia') !=0) {
    manager.firstBookmark('Australia');
    manager.deleteBookmark('Australia');
}
```
9. Do the same for the "Coasts" movie. The script of course must
use 'Coasts' rather than 'Australia'
This process can be sped up by selecting the 'Australia' movie
and by right clicking specify "Copy properties to all" and select
the properties to copy to other movies. This way "Coasts" has
all buttons defined the same way as "Australia". You still must
change the script for the blue button to use "Coasts" rather
than "Australia" as the cloned copy uses.

Create a popup menu displaying bookmarks

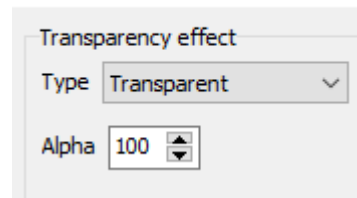
10. A popup menu is needed for each movie to display all
bookmarks entered.



Do one of the following:

- a. The basic format for the popup is given in the Photoshop file “bookmark popup.psd”. Import this file from the project’s \original sources folder.
 - b. If you work on separate .png files, first create a new popup menu named “bookmark popup”. Populate it by adding 2 static objects whose images can be found in \Sources\Project objects\menus\bookmark*.png . You might first copy those files to the project’s \original sources first and use those.
 - c. If you create it from within BDS, create a new menu with a white rectangle object as background. Then a text object with the texts (or multiple text objects with one text each).
11. Reposition the popup menu elements to a suitable place on the final screen – for example somewhere at the bottom.
 12. Rename the popup menu to “bookmark popup Coasts”. We will work on this menu and then later clone it for the “Australia” movie popup – with some adjustments where the name “Coasts” must be replaced by “Australia”. Although the code for both popups is almost identical, BDS has no way to reuse the menu between both movies.
 13. The “time” placeholder is not part of the Photoshop or .png images. It is not an image but a text object that needs to be created in the Objects window for the popup menu. (See section “Menus made within BDS” on page 90).
 - a. Create a new static text object through the  button.
 - b. Specify a dummy text like “text” at 50 pt black Arial. The font is known since you specified the project’s “Used Fonts” location of the font.
 - c. Position the “texts” object to fit between the “previous” and “next” text already there.
 14. Three buttons are needed to allow to move to the previous or next bookmark as well as play the bookmark. They are present in the project’s \buttons folder. We use the same image for the normal and selected state – the normal state is partly transparent; the selected state is opaque.
 - a. Create a new button object “previous” with “Normal” and “Selected” state image “left arrow button green small.png”
 - b. Make the “normal” states partly transparent. For this, right click on the “normal” state in the Objects window and select “Change Effects”. In the window that opens, set “Transparency effect” Type to “Transparent” with an alpha value of 100 (0 = fully transparent, 255 = opaque).

Effects



The same window is opened when you select the “Effect” property of the button’s normal state.

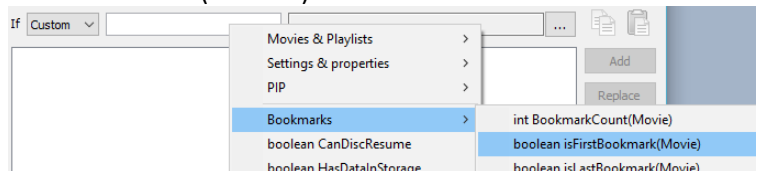
Name	previous
Normal state	left arrow button green small.png
Effect	alpha 100

15. Do the same for the “next” and “play” buttons
16. Add navigation between the buttons.

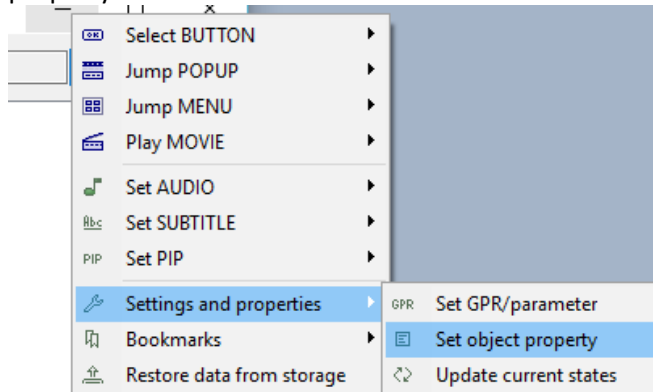
Add actions to the button

17. The buttons need to “do” something when the “OK” button is pressed. The “previous” must show the previous bookmark except when it is already on the first bookmark. Ditto for the “next” when it is already at the last bookmark.

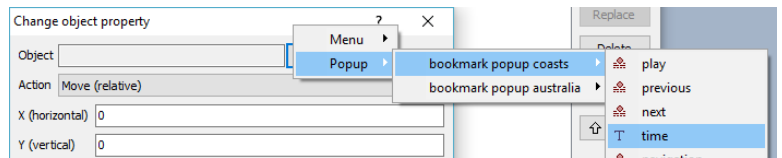
- a. Open the “previous” button property “Press ENTER” and select a Switch. It must be exclusive.
- b. Add a custom condition that checks if it is on the first bookmark in this movie (Coasts). Right click in the “custom” conditional field and select Bookmarks > isFirstBookmark(‘Coasts’)



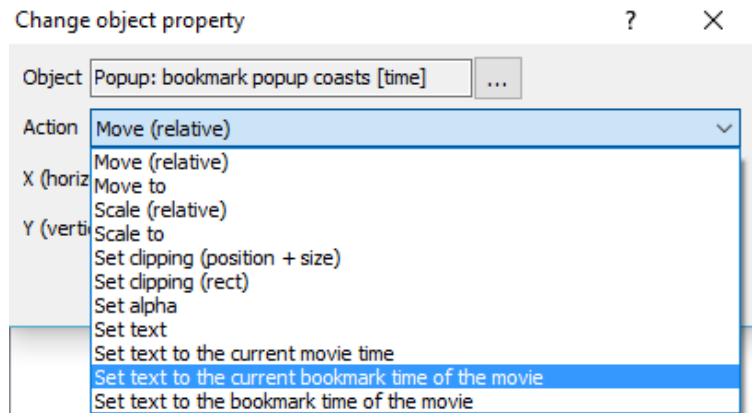
- c. If so, set the “time” object to a text value that shows the first bookmark time.
For this, click the “...” button for the action field and select “Settings and properties” > “Set object property”.



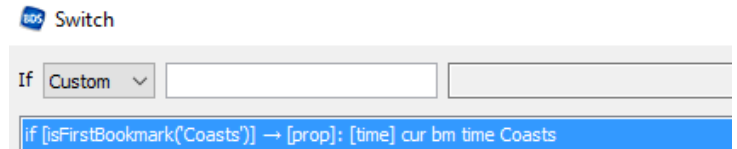
This opens a window in which the object can be specified whose property is changed. In this case it is the “time” object in the popup menu.



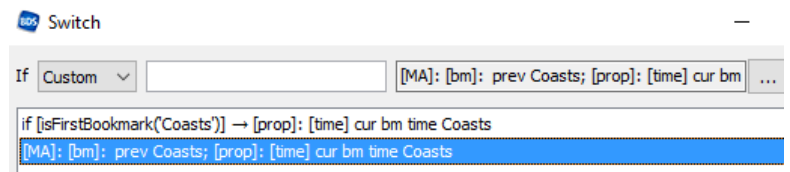
The action to perform is selected next: it must set the text to current bookmark time value.



Specify also that it concerns the “Coasts” movie. Click on “Add” to add the condition to the Switch. This results in the first Switch condition:

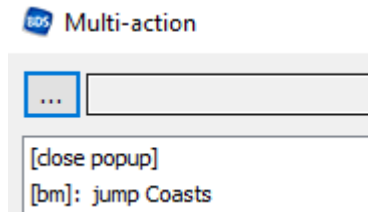


- d. Now that we covered the situation when the first bookmark was already shown, we still need to add an action in case we were not on the first bookmark and “previous” must go the that previous bookmark. This requires a second Switch action. However, this must be a multi-action to perform two steps:
 - i. Move to previous bookmark (Bookmarks > Edit Bookmarks > Select previous bookmark on movie “Coasts”)
 - ii. Update the “time” object property with the time of that bookmark (Settings and properties > Set object property for “time” with action “Set text to the current bookmark time of the movie”) for “Coasts”
- e. Save the multi-action and “Add” it as second action of the Switch. The Switch should now look like:

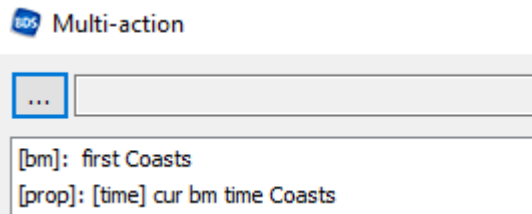


- f. Repeat these steps for the “next” button. Except use the bookmarks condition Bookmarks > IsLastBookmark for “Coasts”

- g. Finally, the “Play” button action must be specified. This is straight forward. It also has a multi-action performing two things:
 - i. Close the popup menu (Jump POPUP > [close popup])
 - ii. Start playing the movie from the current bookmark time (Bookmarks > Edit bookmarks > Jump selected bookmark for movie “Coasts”)
- h. Save the multi-action. The “play” button action should now look like this:



- 18. The menu objects have now all been edited. Remains the popup menu itself. When it opens, it needs to set the “time” value to the first bookmark time. This requires a multi-action performing two things:
 - a. Select the first bookmark
 - b. Set the “time” object text to the bookmark time
 These steps have been covered before. In the end the “Enter” property contains the following multi-action:

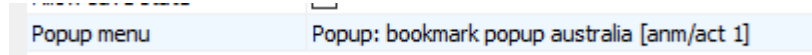


- 19. The “bookmark popup coasts” menu is now ready. Rather than doing the same for “bookmark popup Australia” we simply clone the “bookmark popup coasts” and rename the clone to “bookmark popup Australia”.
- 20. Almost done. But all references in the clone to “Coasts” must be replaced by “Australia”. This remains a bit tedious, but if you open a Switch or Multi-action that contains the movie name “hardcoded”, open it, double click on the action line and it is duplicated in the conditional and action boxes. Right clicking (condition) or clicking on “...” buttons (action and multi-action boxes) allows to reopen the statements. Replace the movie name and close the action. Rather than using the “Add” button you can click the “Replace” button to replace the original statement by the edited statement.

Connect popup menu with movie

- 21. Finally, the popup menus must be connected to the respective movies. For this, select the movie properties and set their “Popup Menu” property to open the popup menu we just made. But not just a plain open – use its [anm/act 1] option since the popup menu opens with executing some code in its

“Enter” property. This would not happen if you simply open the popup menu as any other “plain” menu.
For the “Australia” movie the popup menu property should read as the figure below. The “Coasts” movie ought to have something similar.



A small diversion

The conditions needed to check if the popup menu bookmark is already first or last, has been checked as described above.

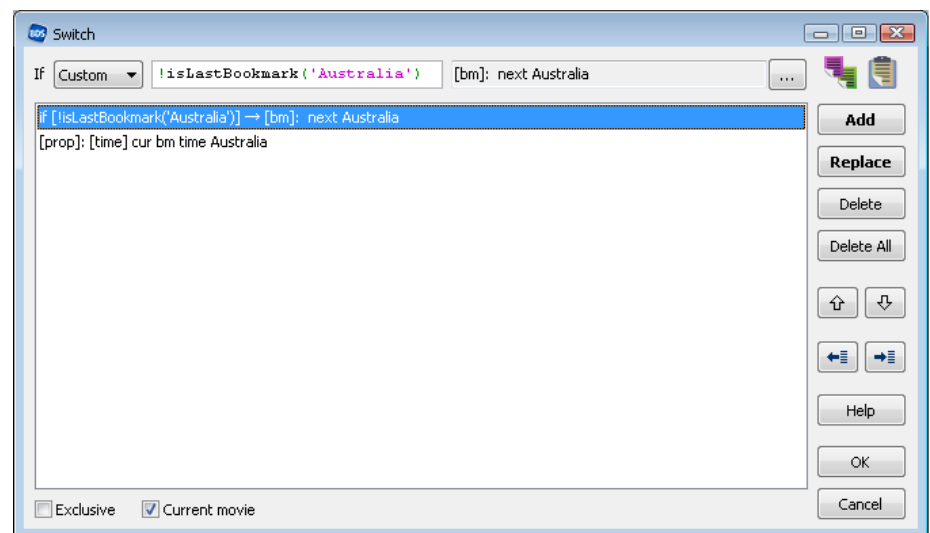
It can be made more compact. If you realize that the “!” exclamation mark means “not” in Java, the condition for not being on the last bookmark can be written as

```
!IsLastBookmark('moviename')
```

This will return true for anything but the last bookmark. In that case, the action is to move to the next bookmark.

In all cases the selected bookmark must then be copied into the “text” object. This requires a non-exclusive Switch.

For the “Australia” popup menu “next” button, the Switch action would look as the figure below. Note its “Exclusive” is not checked.

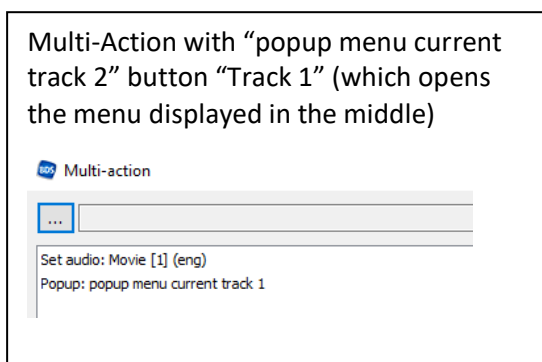
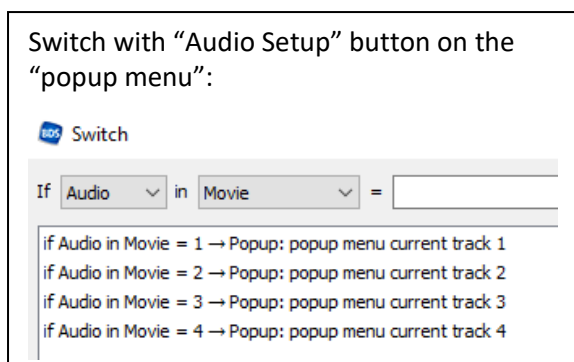
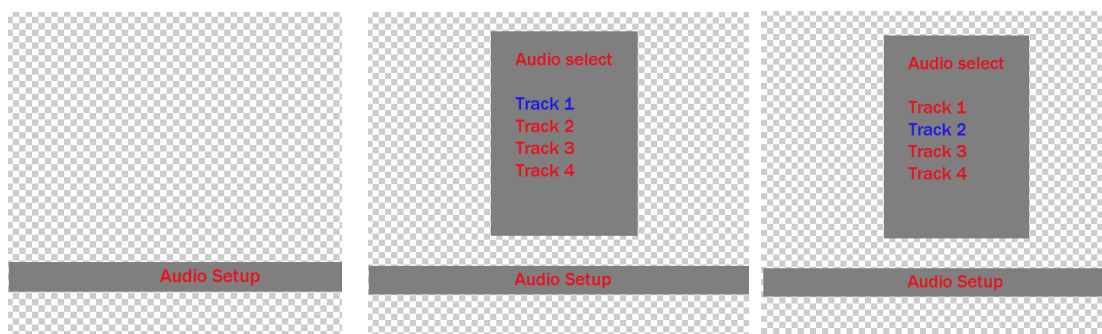


D.I.Y. project: changing submenus using Switch and Multi-Action

A small home project without handholding but that may come in useful.

Many bluray discs have a specific “Audio Setup” popup menu while running a movie that leads you to a popup menu with all the audio tracks you can choose from. Using the “Current” state of a button you can indicate what the current setting is.⁴³

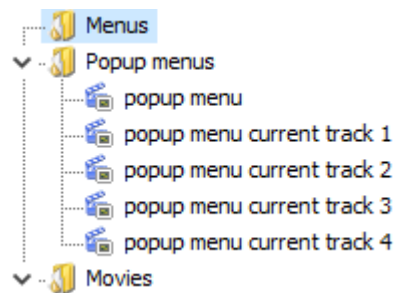
Some discs have a different approach: they show the Audio Setup list with track choices but the current track is dimmed – you cannot select it (why would you? It’s already the current setting). As such there is no “Current” state for buttons: the choice that is current has no button on the menu. As soon as you select a different track, that one becomes dimmed and the earlier current one now becomes selectable.



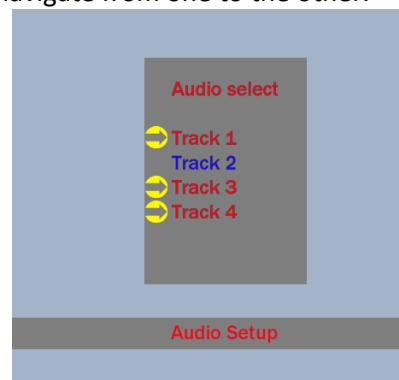
How do you do this?

Some starting suggestions, based on the screenshots above. Rather than “dimmed” (that can be obtained using the button “Change Effects” property) the currently selected audio track is displayed in blue. The other, selectable, tracks are in red.

⁴³ This approach can also be used with regular movies that show the audio submenu (making the main menu a static clone and the buttons on the submenu the real buttons)



- Create a popup menu that displays when a movie runs. It can be a horizontal bar with several options such as “Audio Setup”, “Main Menu”, “Scenes” etc. Selecting “Audio Setup” and activating it (or press Up arrow on the remote-control), opens a popup menu with audio tracks. (Which menu is determined by the Switch action that is triggered – see below).
- Create as many additional Audio popup menus as you have choices. They all look identical but for the selected track: each menu has (another) audio track dimmed and not selectable. Call them “Audio track X current” when track X is the dimmed track.
- Add buttons to all tracks except the dimmed one and make sure you can navigate from one to the other.



- When you select the “Audio Setup” button on the main popup menu, it invokes a Switch action that does one thing: If Audio in movie = track X, then open menu “Audio track X current”
- All “Audio Track X current” menus have the main popup menu as a picture (it remains visible, but inactive)
- When you select a button on the Audio menu, it activates a multi-action that performs two actions:
 - a. Set the audio track to the selected track X
 - b. Open the menu “Audio track X current”

To avoid lots of work, consider to:

- Create one button on menu “popup menu current track 1”
- Make two extra copies (copy/paste in same menu) and position them next to the red tracks
- Set up the Press ENTER multi-actions for each button
- Select all three buttons and copy/paste them to the other audio popup menus (this also copies their Press Enter actions)

- Personalise each menu and move the one button in front of a blue track to the red track without a button. Modify its multi-action. The other buttons are already correct.

Part 4: Animations

Project 8: Animations

The Project Goal

To spruce up a menu, animation allows to have menu objects move about the screen. The menu movie plays in the background and has no interaction with the viewer.

Animation happens because the viewer presses on buttons that activate it. Some animations happen automatically because it is programmed that some objects move at the opening of a menu.

Animation is the programmed movement of objects on a menu. But animation may also occur when one menu closes (with some animation) and another opens (with its own opening animation). This way animation may simulate a fluent movement from one menu to the next. A wizard-generated animation you may already know: the sliding chapter menus where chapter images are replaced by the next set of chapter images.

Animation can be seen as movement (sliding) of objects, fading in or out or disappearing partly (clipping).

If animation moves objects, this happens from a “From” position to a “To” position. At the end of the animation the objects are displayed in their “To” specified position. If buttons take part in animation, they are only buttons once the animation is completed.

This project

- will show you how to animate your menu using predefined animation features

To focus on this one aspect, we reuse the results of Project 2, the “MenuBR” project.

If you want to create this project yourself, the source files can be found in the .zip file mentioned in Appendix A: The user’s guide source files.

Animation: objects or movie-type

BDS allows to add animations to the menu objects: all objects, be them pictures or buttons. There are two types of animation:

- **Menu:** objects on a menu (texts or images as well as buttons) move or fade or get clipped. The objects that take part in the animation do not change themselves. A menu can have multiple animations, called anim1, anim2,... Whatever action opens the menu makes a choice what animation to apply. Or none if just the menu title is mentioned.
- **Movie-type:** An animation movie must be made frame-by-frame and is then played back as movie. This looks like flipping a notebook of pages where each page has a slightly different image.

A movie-type animation is started only when a JAR file (title) is. It is therefore not part of the menu background movie. When a disc is inserted, the JAR file set to be “First Play” will run its animation file if present. On commercial discs such an animation may show “loading...” text or a spinning disc.

Because animations translate to Java code, if all you change in a project is the animation, it suffices to recompile the JAR files and copy them to the output folders of the disc. No need to remux all movies.

Some use of animation is:

- More dramatic entry of a(nother) menu
- After a button is activated on a menu, a submenu slides in for further options
- Depending on the situation, a new menu may be opened, using one of several animation options
- If there are multiple chapter menus, moving from one menu to the next (or previous) can be animated as new chapter buttons slide into place (if this is an endless loop where the last chapter reconnects to the first chapter it is called “carousel animation” – some “live” examples are provided by the BDS website for “the King’s Speech” and “Force of Execution”).

Animate an object

There are several types of animation for an object such as moving, fading or clipping (partly being cut off). We will look into these types of animation later (see Types of Animation on page 255). To animate an object, you need to specify its “anim” properties.

- Menu: when a (popup) menu opens, it can execute one of whatever number of animations are defined for its opening. Whatever opens the menu must specify what animation to execute. When none is specified, no animation is executed. A new opening Action/animation pair is created when one set is specified. The figure below shows two animations (anm1 and anm2) defined for a menu. No action (act1 and act2) is performed when opened.

Enter	
Animation 1	Slide (0, 591) - (0, 0) [0, 1920] - [0, 1080] {cubic-in}; Group1 <S> [12]
Action 1	
Animation 2	Slide (0, 680) - (0, 0) [0, 1920] - [0, 1080] {cubic-in}; All <S> [12]
Action 2	
Animation 3	
Action 3	

An action from a button to display the menu would indicate its name (e.g. “Extras”) and which animation to apply:

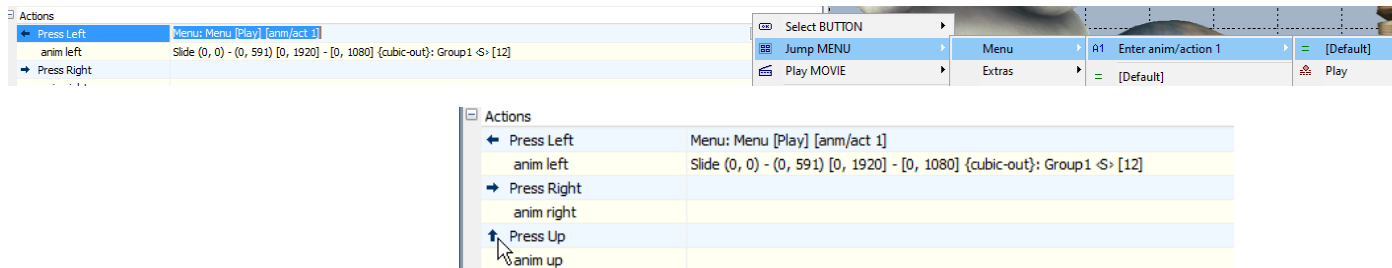
OK Press ENTER	Menu: Extras [anm/act 1]
OK Press ENTER	Menu: Extras [anm/act 2]
OK Press ENTER	Menu: Extras

A popup menu can close itself – a menu cannot (other than replacing itself by another). . For this reason there is also a “Close popup anim” property to specify any animation to associate with closing the popup menu

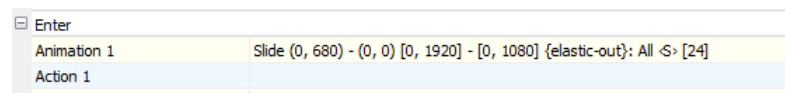
- Button: for a button there is a maximum of two animations for each action:
 - A single animation related to the button action
 - A second animation if the button opens a menu that has “Enter” animations defined.

It is set by pressing the “>” button in the “anim left” property field and select the menu

The example below shows an animation when the left arrow button is pressed. It first executes its own animation (anim left) and then the one specified in “Press Left” property which is open the menu called “Menu”. For this menu it selects its first animation.



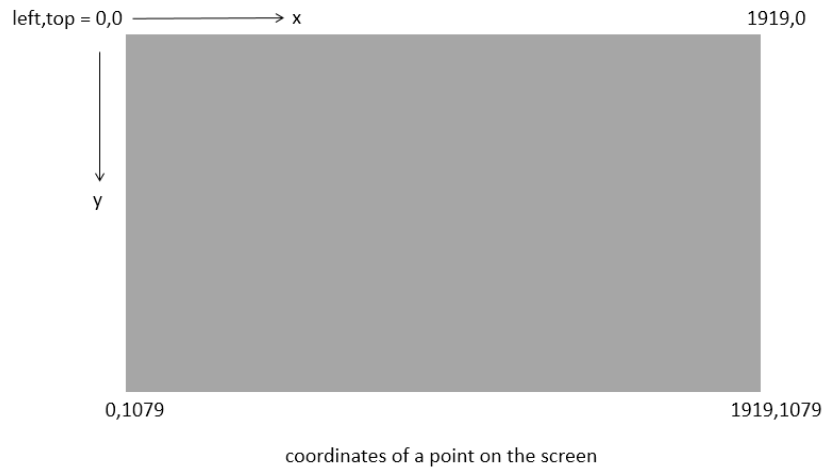
The “Menu” object’s property “Animation 1” defines what to do with “anm1” which is to slide in:



Screen positions and displacement

Screen position

For animation to work, you need to know how the pixels are counted on a screen. Unlike what you may have learnt at school, the origin is positioned at the top left corner and works its way rightwards and downwards. So the position of a point P is given by (pixels-from-the-left, pixels-from-the-top), (x,y).



Any menu object, large or small, has its position defined as the position of the top left hand corner. When that corner is moved, the rest of the object moves with it.

The x-coordinate is called “left”. That may sound odd. “Left 1919” means the position 1919 – which is the right end margin. But anything horizontal is offset as “left” value because the left most point of an object is at its “Left” position.

In the same manner, the y-coordinate is called “Top”. A top at 1079 is in fact at the bottom of the screen. Unlike high school mathematics that have y-values increase upwards, here it increases downwards. The highest point of an object is at its “Top” position.

In the Designer Window you place all menu objects at whatever position (Left,Top) you want.

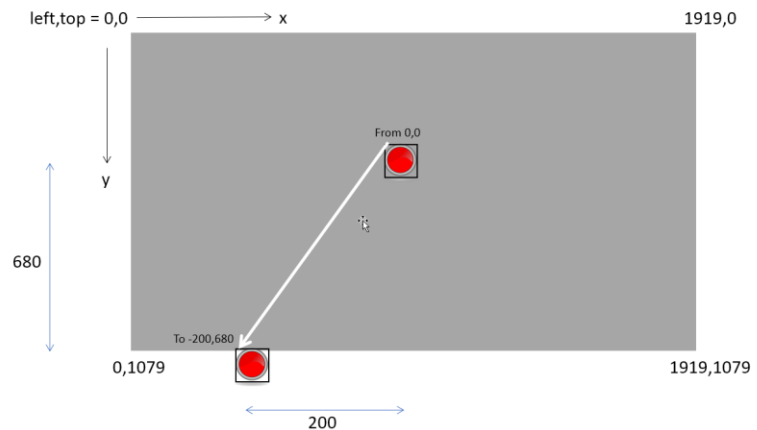
Displacement

Animation of an object is always given as displacement. It moves “From” an initial position “To” its final position. This movement is specified as the number of pixels difference between the “From” position (x_1, y_1) and the “To” position (x_2, y_2) and therefore as ($x_1 - x_2, y_1 - y_2$). Here x is the horizontal coordinate (left to right) and y is the vertical coordinate (top to bottom). Both coordinates x and y can only have positive values and $0 \leq x \leq 1919$ and $0 \leq y \leq 1079$ for a full HD screen of 1920x1080 pixels

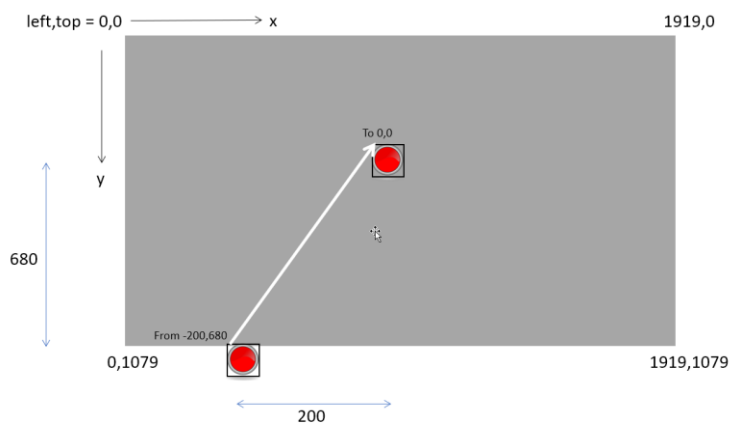
Because animation can move to the left or to the right (or up or down) the relative displacement in either direction can be negative or positive.

Animation of an object is always given relative to the position it is placed in the Designer Window.

- When the “From” position is set to (Left 0, Top 0) and “To” as a non-zero value, the animation is from the position on the Designer’s window to the final position. When the animation completes the object is back at its (Left 0, Top 0) position unless some action prevents this (like displaying a different menu where the moved object is at its new “To” position)



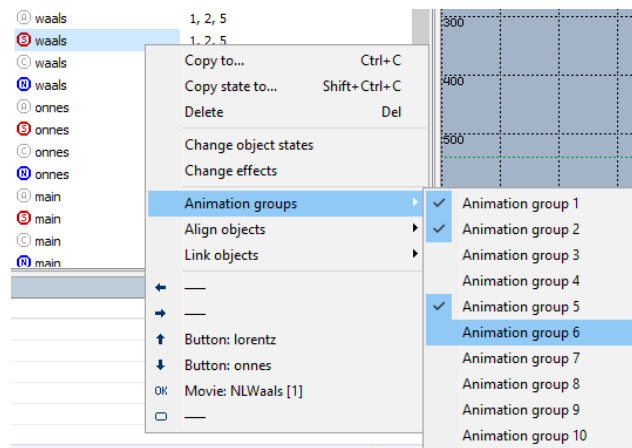
- When the “From” position is some non-zero value and “To” is set to (Left 0, Top 0), the animation is from the “From” position to the position given on the Designer’s window. It will remain there as that’s where it was designed to be (in (Left 0, Top 0)).
So moving objects without changing menus afterwards is usually employing this method: from anywhere to the position given during the menu design.



Animation groups

All objects that are to be animated need to be part of an animation group. Animating a single object means it has to be its own animation group. Only (popup)menus have their own “animation” properties and animate without being part of a group.

Objects can be joined together into a single animation group. BDS allows a maximum of 10 groups. That also means a maximum of 10 animatable objects if each has its own unique animation group.



An object is added to a group through a right mouse click on the object in the Object window. The context menu that opens has a “Animation Groups” option which in turn shows a list of 10 animation groups. By clicking on one of these groups, the object is joined to that group. A checkmark indicates its membership.

Clicking again removes it from the group. If an object belongs to no group it cannot be animated.

The Object Window shows in the “Anim” column to which groups an object belongs.

Ⓐ waals	1, 2, 5
Ⓑ waals	1, 2, 5
Ⓒ waals	1, 2, 5
Ⓓ waals	1, 2, 5

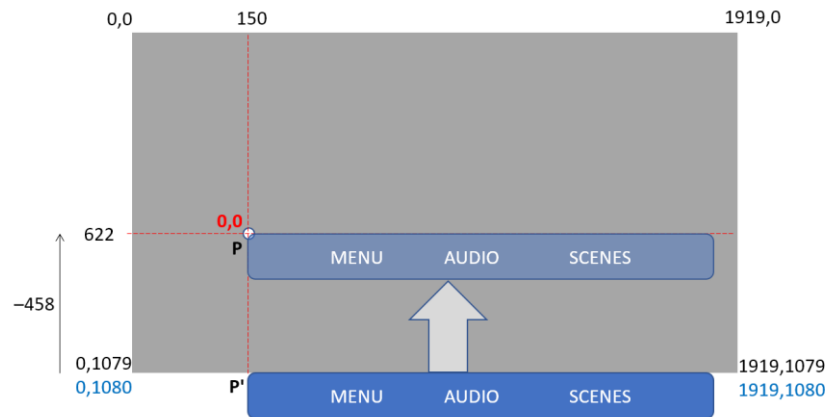
An object can be member of multiple groups. If several animations are of the moving (slide) type, it can only execute a single sliding animation at any one time. Other types of animation (such as fading or clipping) can happen parallel to the sliding.

An object can also perform an animation on itself, regardless of what groups it is a member of.

Example: animate a submenu

Take an example of a more complex object: a set of menu objects consisting of texts, a background rectangle and some text buttons that move together as one submenu. They are joined by being made a member of the same animation group. Animation is specified for that group.

Once the animation ends, text buttons with names “Menu”, “Audio” and “Scenes” become real buttons again.



The objects of the submenu are designed to slide into view from the bottom of the screen. The animation group moves from position P' to P.

To animate this move the animation needs to move (slide) from absolute position Left 150 to Left 150. That means a relative shift "From:" Left 0 to the new "To:" Left 0. A vertical move upwards – no change in Left position.

The vertical position does change its absolute position "From:" Top 1080 to the new "To:" Top 622. That means a relative move "From:" Top (622-1080 =) - 458 to the new "To:" Top 0.

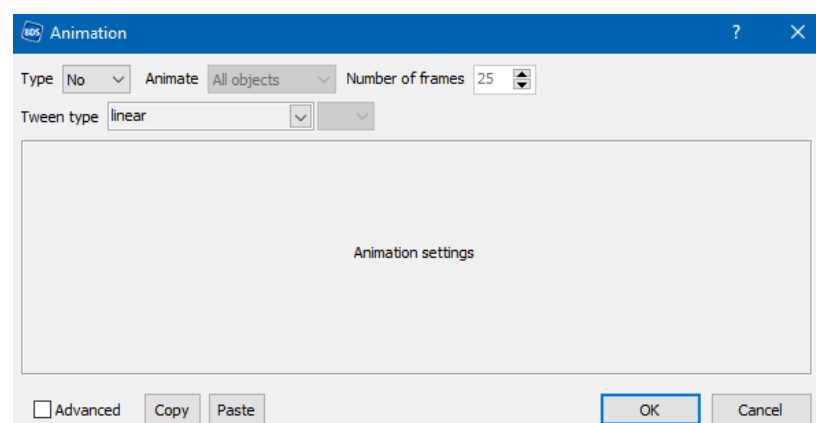
In short, this means the submenu moves relatively From (Left 0, Top - 458) To (Left 0, Top 0).

When the animation ends, the objects remain at the "To" position as designed in the Designer Window and the three text buttons "Menu", "Audio" and "Scenes" become true buttons again.

Types of Animation

Sofar we looked at how to associate an animation with a menu or button but did not look to closely at what that animation entailed.

The precise syntax required in the animation properties is set when in an animation property of menu or button you click on the ">" at the far right. This opens the animation specification window.



Initially it is set to Type = No (animation). If for some reason you want to remove a previously set animation, simply double click on the animation property and when the Animation window opens, set its value to Type = No.⁴⁴

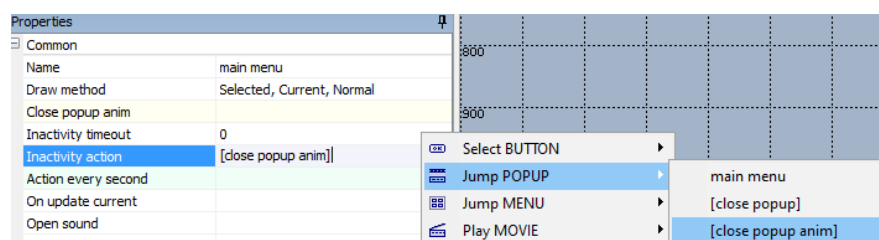
Selecting any other value for “Type” produces a menu animation choice:

- Slide (movement) in a restricted number of ways (from A to B)
- Fade in or fade out (change transparency from 100-0% and 0-100%)
- Clip objects (cut off outside boundaries)
- Scale objects (become larger or smaller)

Three of the most common types of animation (slide, fade and clip) we will cover in a bit more detail in a next section.

Closing popup menus

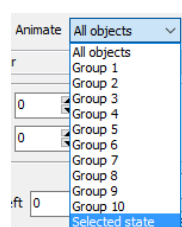
Important: when animation is added for when a popup menu closes, you must explicitly set the “Inactivity Action” to “[close popup anim]” via the Jump Popup action. Otherwise it will close without any animation. The same is true for closing a regular menu.



Selecting object(s) to animate

Selecting one object in a menu means selecting the animation group it belongs to. In the “animation” property you click on its “>” button to open the Animation window.

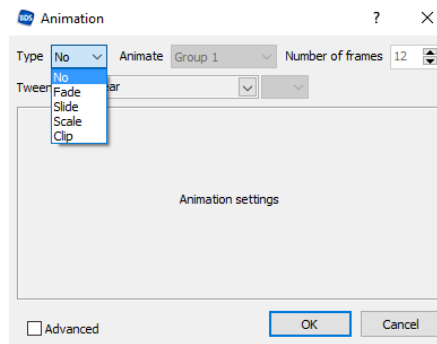
If the “Type” is set to anything but “No”, you can select its “Animate” box in which you specify whether the animation applies to an animation group, all objects on the menu or, if you selected a button state, to the selected state (and not the other three possible states).



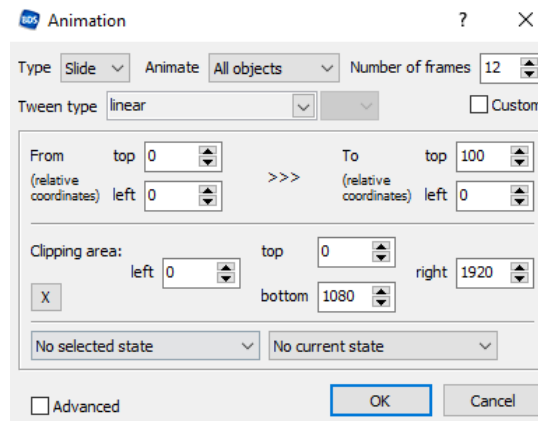
Animation: Slide

To move an object from A to B you need the Slide animation. In the Animation window select “Slide” in the “Type” dropdown box.

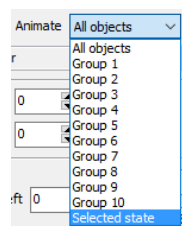
⁴⁴ This is different than for some other properties where either pressing the “delete” key or selecting “-none-” from the context menu suffices



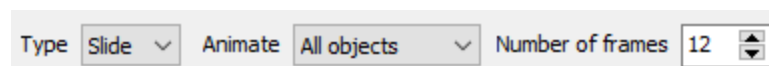
The Slide Type provides all the items you must specify.



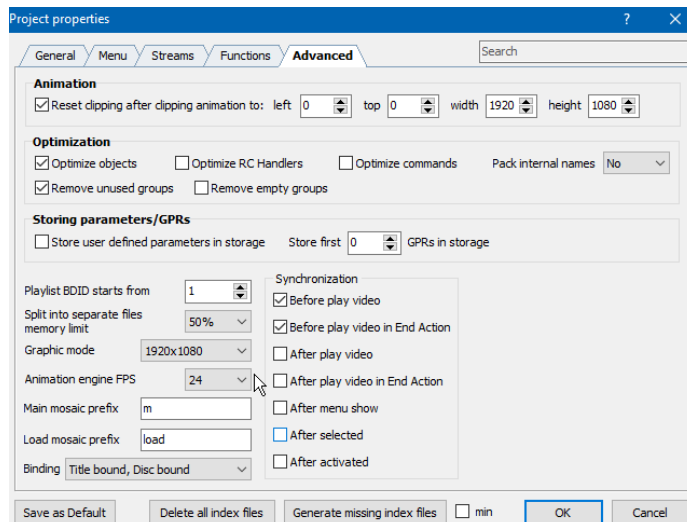
The default is to apply this animation to “All objects” (if a menu animation is going to be set). For a single button you specify the animation for its state for which you opened the animation window. The most visible animation is of course for its “selected” state. When it opens a menu, the animation specified for that menu of course is executed.



Next the speed of the animation is given in the “Number of frames”. The more frames – just like a movie – the longer it takes. The number must be between 1 and 250 frames.



How many frames are used for a second of playing time is specified as the “animation speed”. You can set it in Project > Project Settings > Advanced tab in the “Animation engine FPS” box speed (23.976, 24, 25, 29.97, 50, 50.94 or 60).

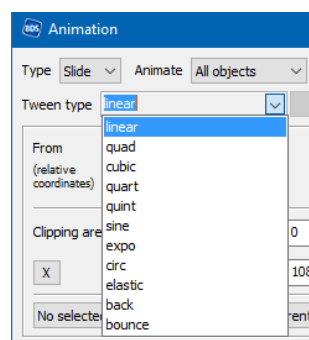


In the animation property box you can see the speed because the number of frames used for the animation is given in square brackets, like [12] (for 12 frames).

The actual movement from A to B is given by the “From” and “To” boxes.



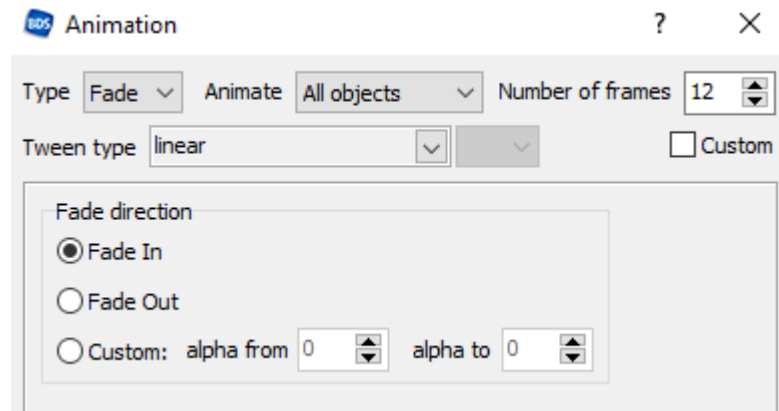
When you want the objects to remain at their final destination (set in the Designer Window), set the “To” boxes to Left 0, Top 0, and the “From” values to some other non-zero value position. If that position is off screen it looks like the objects slide in.



The movement is a straight line from A to B but the speed along the way can vary: its “Tween type”. The default is a constant speed, but more options are possible that vary this speed. For example, “sine” gives a sinusoidal change in speed: slow first, faster and slow again at the end. “Sine” does not mean “wiggle”! The A-B path remains a straight line. See online help topic “Tween type”. The best is to experiment with these settings to see the animation in action (e.g. in simulation mode for the menu).

Animation: Fade

To let an object appear or disappear while keeping its position is called “fade in” and “fade out”. The Type of animation is “Fade”.



Like with Slide, all or some objects can be grouped in an animation group to be faded simultaneously. The speed of fading is the number of frames. You can specify whether the fade is appearing or disappearing. If you don’t want a complete fade in or fade out, you define the transparency limits through “custom” setting with alpha values between 0 (fully transparent) and 255 (opaque, not transparent).

The speed of fading depends again on the speed specified by the number of frames (see previous section).

Animation: Clip

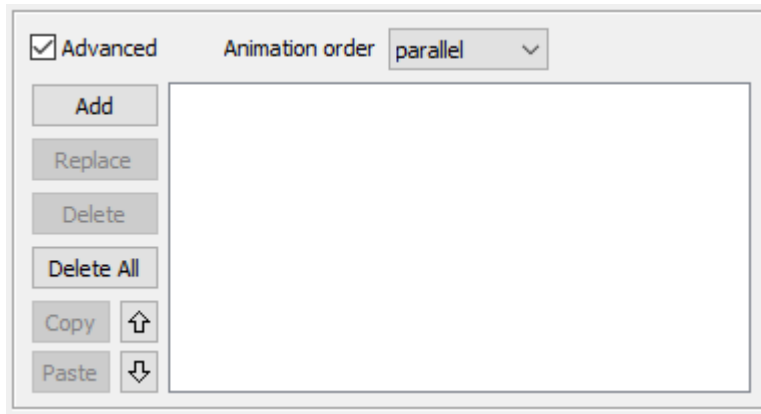
Animation occurs on screen. By clipping an area on screen the animation is only visible within that area. If the animation goes on beyond the area it becomes unseen – as if it moved behind curtains.

This type of animation is useful if you want to show a static object that (by changing the clipping area’s width or height) becomes more and more visible as if the curtains are drawn back.

One use of clipping is to produce an extending progress bar on a timeline. The progress bar has its “Enable Clipping” property set (via its object window Change Effects>Enable Clipping checked) and its visible length increases as the clipping area increases from 0 to final length of the bar. An example of this can be seen looking at the online Help under FAQ > Timeline. For a Java programmed example see Project 12: Add a popup Timeline menu on page 327.

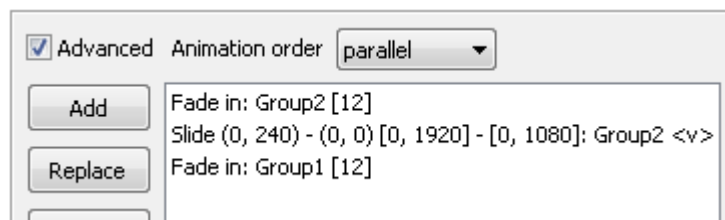
Multiple animations

You can perform multiple animations on the same object or different groups using the “Advanced” option checkbox. This allows you add all animations to a list that can be executed one after the other (sequentially) or all simultaneously (parallel). All animation types have the “Advanced” checkbox.



The way to go about it to define one type on animation (Slide or Fade) and Add it to the list. Within the same window, select another Type (and/or animation group), set its parameters and add it to the list too.

The multi-animation resembles the Multi-action of the Action properties of menus or buttons. A difference is that animations can be executed in parallel whereas Multi-action can only execute one action at the time (sequential).



Animation menus as “in betweenner” or “transition”

An animation menu is a menu that has no buttons – only picture (or static) image elements. Such a menu can only exist in between two normal menus with buttons. Its animation can be simple or [Advanced], meaning that multiple animations are executed in parallel or in sequence.

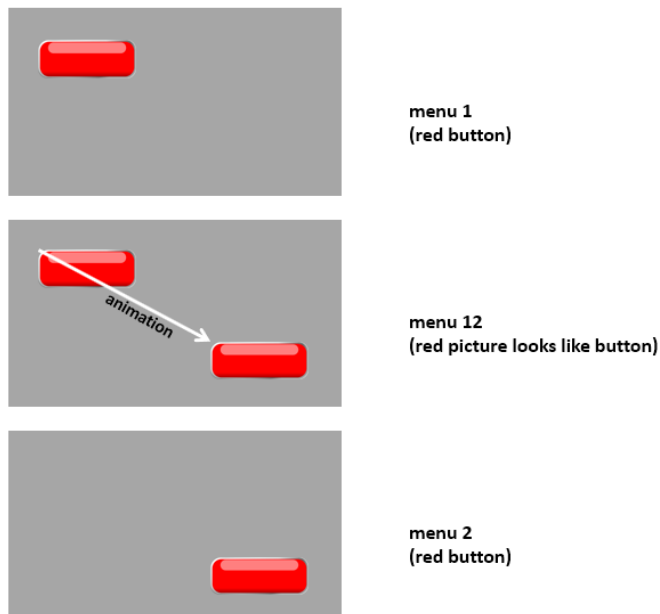
Because the animation menu has no button, you could never press a button to leave it as it has no buttons. For this reason, BDS allows “picture only” menus, but only if its “Enter” properties contain an action that relegates its control to another menu that has buttons. (You can transfer to another animation menu, but at the end of the chain of menus we must end on a menu with buttons).

Enter	
Animation 1	[Advanced]
Action 1	Menu: Main

An example of an animation menu is where it animates (part of) a “Menu 1” into a “Menu 2”. The “From” starting position of the objects on Menu 1 coincides with the position in “Menu 1” given in the Designer Window. The final “To” position after the animation coincides with the Designer Window position in “Menu 2”. The transfer between menus is seamless and visibly unnoticeable to the viewer.

The sequence is

- On “Menu 1” an action (when a button is pressed) opens the animation menu “Menu 12”. Menu 1 instantly disappears.
- “Menu 12” is a static (pictures only) clone of “Menu 1”. The transfer to “Menu 12” is visually unnoticeable. But there are no buttons that can be selected or pressed. They have become mere images.
- “Menu 12” starts its animation. Image objects of buttons move from their position on “Menu 1” to the new position on “Menu 2”
- As soon as the animation ends, “Menu 12” transfers control to “Menu 2”. “Menu 12” instantly disappears. The transfer to “Menu 2” is visually unnoticeable as the end position of the animated objects is exactly where “Menu 2” wants to have them.



Animation and/or action at menu open

You may have realized it from the previous section, but it’s worth making it explicit nonetheless. The “Action” and “Animation” as menu properties can be used when you want the menu to open. The “in betweener” menus did this, but you can use it for any menu opening: when you specify “Jump to Menu” in some (button) action and specify the menu you have two options:

- provide menu name and pre-selected button.
The menu will open and the button is selected. The action line will look like “MenuName [button1]”
- provide menu name, pre-selected button and action/animation. When there are multiple pairs of animation/actions defined, you need to select the one pair you want to execute.
The menu opens, the animation is executed first, followed by

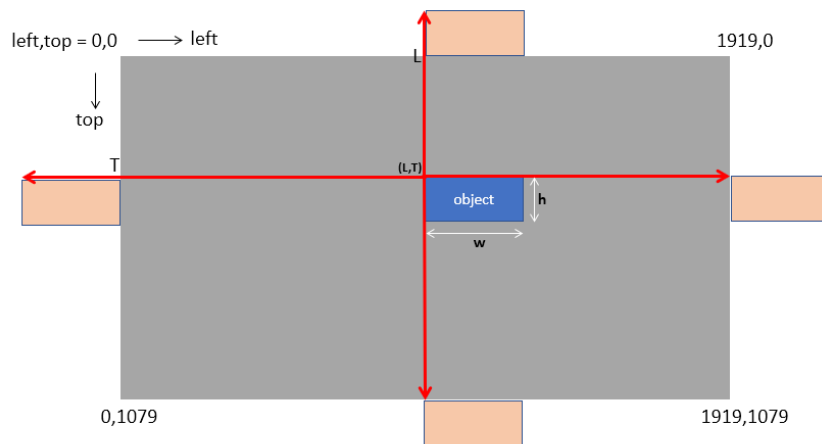
the action, the indicated button is selected. The action line will look like “MenuName [button1],[anm/act1]”

Sliding animation arithmetic (or D.I.Y.)

It sounds as if sliding animation is difficult because of all the calculations you must make. But it is not really. The “offset” values are easy to use once you understand how they work. In the following sections we assume an HD menu of 1920x1080 pixels is used.

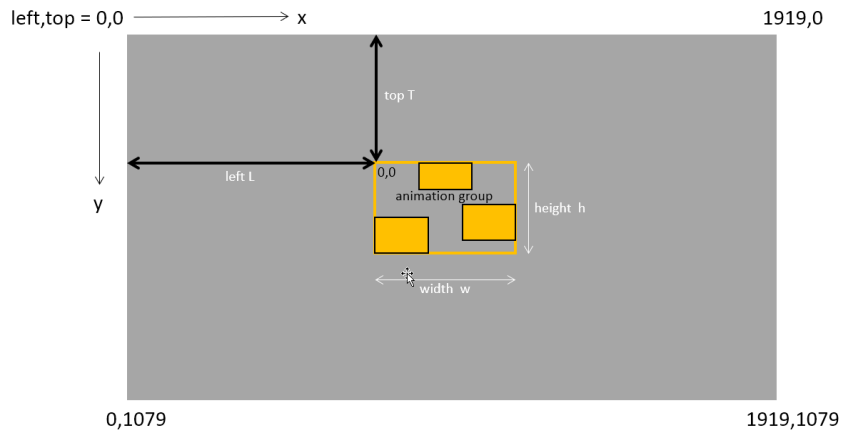
Moving an object out of sight

Suppose you have an object that has its top left corner positioned on screen at (T,L) where $0 \leq L \leq 1919$ (that is, the value of L is between 0 and 1919) and $0 \leq T \leq 1079$. The height of the object is h and its width is w .

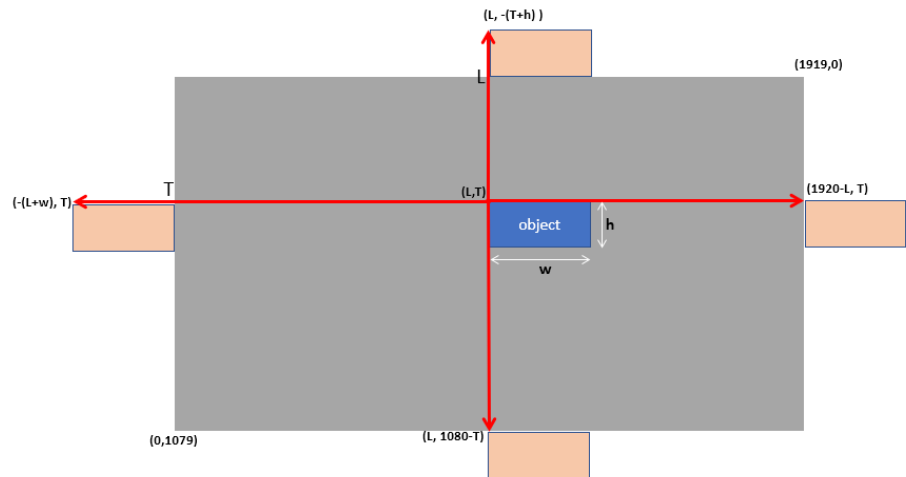


To move such an object out of sight (that is, off screen) you need to shift it from its current (T,L) position as set in the Designer Window, to a position off screen. The current position is always indicated by the position of the top left corner. It has relative position (Left 0, Top 0).

If the object is an animation group, all elements perform the same sliding animation and act as a single object whose top left corner is that of the object that is the left most and top most object of the group. The width and height of the group are determined by the objects that form the top left and bottom right part of the rectangle that represents the group as if it is a single large object.



- A slide to the left must take into account the width of the object. The slide is then not just the position L of the corner of the object but an additional w to move the object off screen entirely. The sliding animation is therefore horizontally (T remains the same) over a distance of $-(L+w)$.
The slide is negative since the relative origin is at the corner of the object.
Seen from the position of the object, the animation is relative from $(T,L) = (0,0)$ to $(0, -(L+w))$
- Slide to the top must take into account the height of the object. The sliding animation is vertical (L does not change) over a distance of T and then over the height h of the object. Because the origin is at the corner of the object and higher than the origin is considered negative, the slide is vertical over a distance $-(T+h)$.
Seen from the position of the object, the animation is relative from $(T,L) = (0,0)$ to $(-(T+h), 0)$
- Slide to the right is relatively simple. The right edge of the screen is at $\text{left}=1919$, the first pixel column off-screen is $\text{left}=1920$. The slide to get the object there is over a distance of $1920-L$. The position of the corner's top remains the same. We move to the right, in increasing left-position: a positive slide.
Seen from the position of the object, the animation is relative from $(T,L) = (0,0)$ to $(0, 1920 - L)$
- Slide to the bottom is equally simple. The bottom of the screen is at $\text{top}=1079$, the first off-screen. The distance to slide the object down (in positive direction) is then $1080 - T$. The position of the corner's left remains the same.
The relative slide seen from the object is then from $(T,L) = (0,0)$ to $(1080 - T, 0)$.



Moving an object in sight

Sometimes you want an object to slide from an off-screen position to the (L, T) location it has on the Designer window (or as you specified as properties of the object).

This animation is in fact the reverse of the previous situation where we moved an object from its position off screen. Therefore, without too much discussion and using the same illustrations as for “Moving an object out of sight” the required animations are given below.

- Slide in from the left: relative slide from $(T, L) = (0, -(L+w))$ to $(0, 0)$
- Slide in from the top: relative slide from $(T, L) = (-(T+h), 0)$ to $(0, 0)$
- Slide in from the right: relative slide from $(T, L) = (0, 1920-L)$ to $(0, 0)$
- Slide in from the bottom: relative slide from $(T, L) = (1080-T, 0)$ to $(0, 0)$

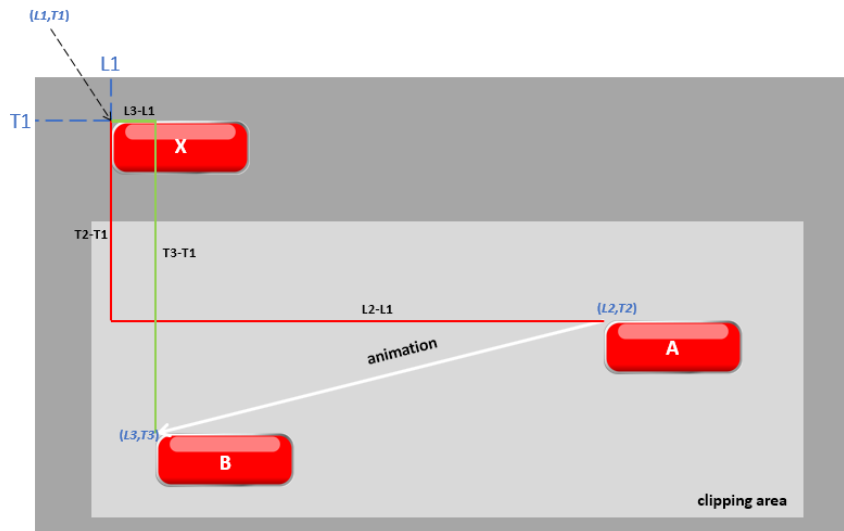
Moving an object positioned at X from A to B

Although you will often animate to move an object from somewhere to its position given in the Designer window (at relative coordinates $(0, 0)$) this is not a requirement.

The object may be positioned at point X on the screen (or off the screen) but the animation moves it from A to B. The position of X is irrelevant for the animation, but it is relevant as the origin to which all animation is calculated.

This type of animation is often used in animation menus that disappear as soon as the animation completes.

One problem though: when the animation ends, the object is suddenly found back at position X. This problem is resolved if you define a clipping area where button X remains invisible outside the clipping area.



The figure above shows the animation of a button A with its left top corner at (T2,L2) in to a button B at position (L3,T3) while the button object is X, positioned at (L1,T1).

Button X has absolute coordinates (L1,T1). The coordinates of the button A relative to button X is the (L2-L1,T2-T1). For button B the relative coordinates are (L3-L1, T3-T1).

The sliding animation from A to B therefore is

- Start at left=L2-L1 and top=T2-T1
- End at left=L3-L1 and top=T3-T1

When the animation stops, the button is back at position X again and both A and B are gone.

Animating several menu items one at the time

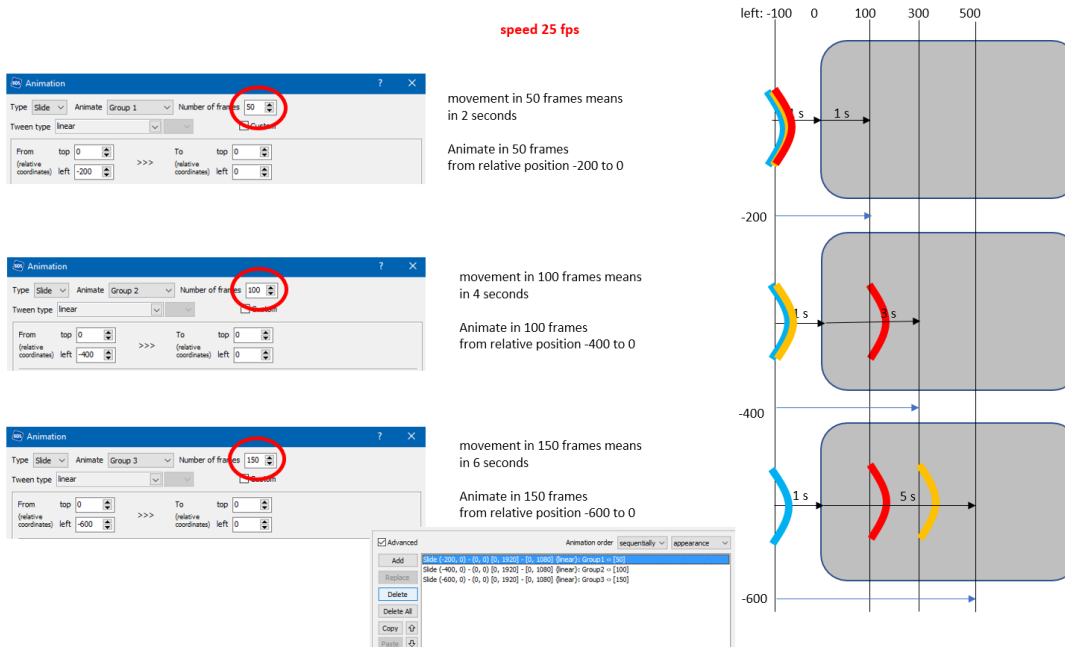
At times you may wish to build up a menu by sliding in different items. One request once was "I have 5 items that need to slide in. First item 1, then a pause, then the 2nd item, a pause, then the third etc".

Initially you might think you may need something like multi-actions containing pauses, but the solution is simpler: open the menu with an animation consisting of sequentially starting objects (in an animation group) to move. As long as the total number of frames for the animation remains less than 250 frames. This equals 10 seconds interval when the animation engine is set to 25 frames/seconds (The playing speed is specified in Project > Project Settings > tab "Advanced" item "Animation engine FPS").

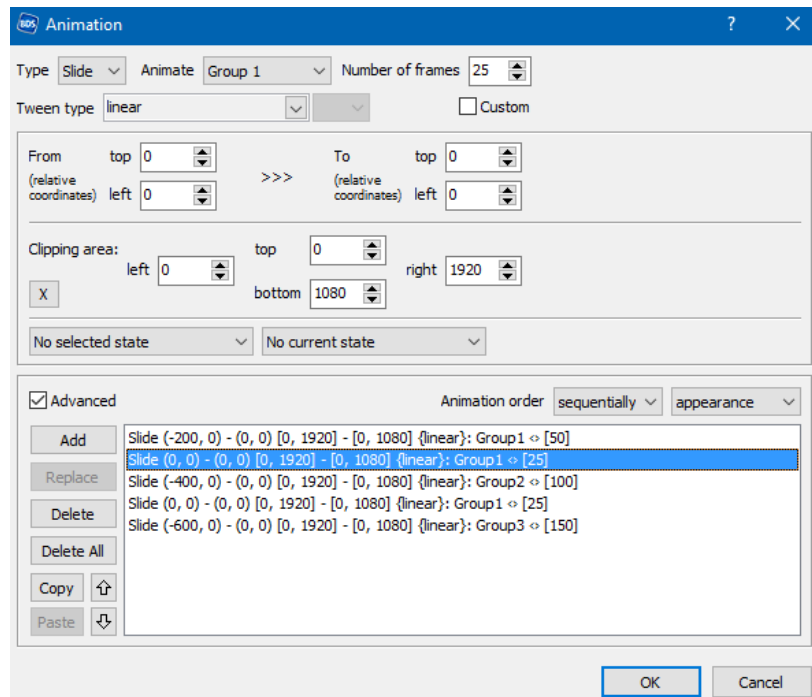


Say for example, you want "1" to appear on a left position of +100, a "2" to follow next to +300, a "3" moving to +500 once the "2" reached its position.

- For this, create the "1", "2" and "3" items at their final left positions at 100, 300 and 500 respectively. They have the same Top value.
- Add each to their own animation group ("1" in Group 1, "2" in group 2 and "3" in group 3)
- Item "1" moves from (relative position) left -200 to 0 (which is from -100 and +100 in absolute measurements). It uses 50 frames to cover this distance – equal to 2 seconds with engine speed set to 25 fps. Set "Number of Frames" to 50.
- Add this animation to the "Advanced" box that is set to "sequential"
- Item "2" moves from left -400 to 0 (from -100 to + 300 absolute). It starts after "1" has reached its position. Since the total distance animated is twice that of the first object ($400/200=2$) it needs twice the number of frames if it is to move with the same speed, hence the "Number of Frames" is set to $2 \times 50 = 100$.
- Item "3" moves from left - 600 to 0 (from -100 to +500 absolute). It travels three times the distance as the "1" object ($600/200$) and therefore "Number of Frames" is set to $3 \times 50 = 150$.



If you want to insert a pause between animating items, you need to insert an additional animation step. That step is very easy: simply take any object (or the one you just moved) and move it from (relative position) 0 to the same position 0. That way nothing moves but the animation takes its prescribed time.



For example, to add a 1 second (=25 frames at animation engine speed of 25 fps) delay you define an additional animation of "group 1" for the duration of "Number of frames = 25" moving from relative position 0 to 0. Add this animation in the "Advanced" window. Copy it as often as you need and use the up and down arrows to position them in between the real animations

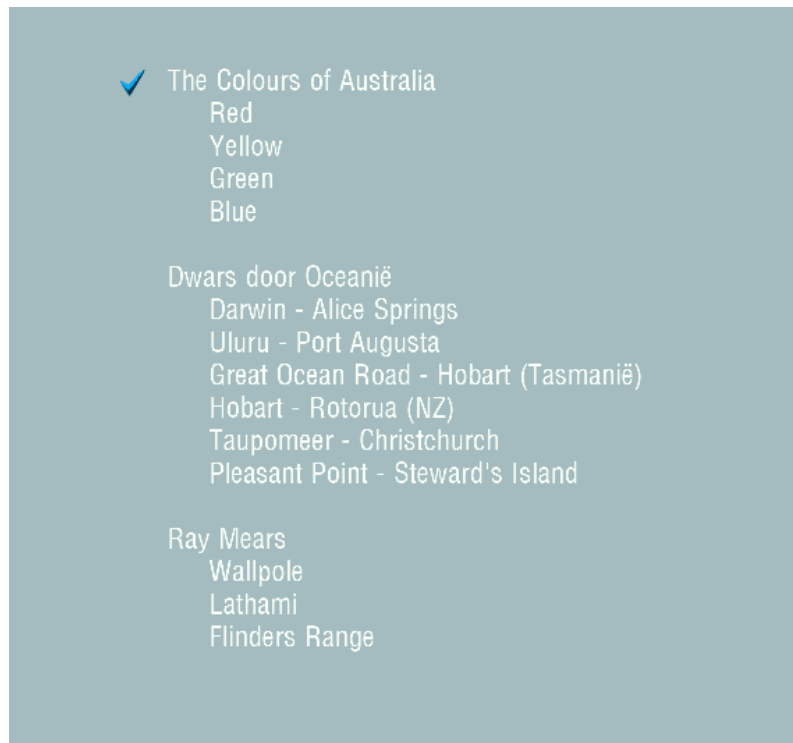
Animation with carousel menus (revisited)

From BDS V4.2.2 onwards, not only chapter image menus can be animated but any menu, main or popup. We already introduced this in section Carousel menus on page 157. It may also be useful to watch the video tutorial on carousel menus on YouTube <https://www.youtube.com/watch?v=2Te7w2l3Zel>.

A carousel can be useful to have all “main items” displayed on the carousel and when one item is selected, show the items that go with the main item.

We illustrate this with a disc about Australia that has six sets of documentaries.

The simple way would be to make one or two main menus that contains all items. Like the first menu shown below.



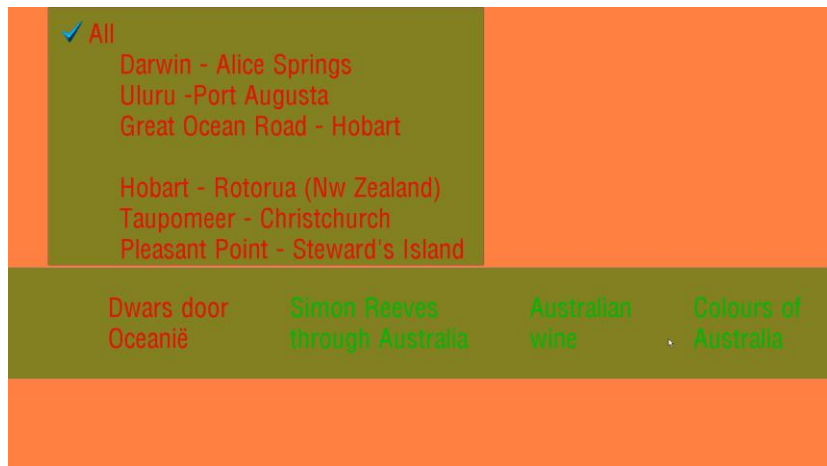
The alternative is to use a carousel with 6 buttons – one for each of the main series of documentaries. For a carousel, these six buttons are the only ones on the menu. Not all buttons will fit on the limited width of 1920 pixels, but we can rotate between them. That’s what the carousel does.

The following images are taken from a Simulation of such a carousel menu.



You only see 4 titles out of the six – the remaining two don’t fit on the green menu bar that has the maximum HD width of 1920 pixels.

One of the carousel windows has “Dwars door Oceanië” as the only button highlighted in red. Because it is the only button, it is automatically selected and shown in a different colour. Pressing “OK” or the “up arrow” on that button triggers the “Press ENTER” action which causes the second screen with titles to appear by sliding upwards from the green menu bar.



The “All” (Play all) is a playlist item to run all episodes as a single movie (using the manner to skip identical title films and credit titles as explained in Project 10: watching episodes without titles on page 296).

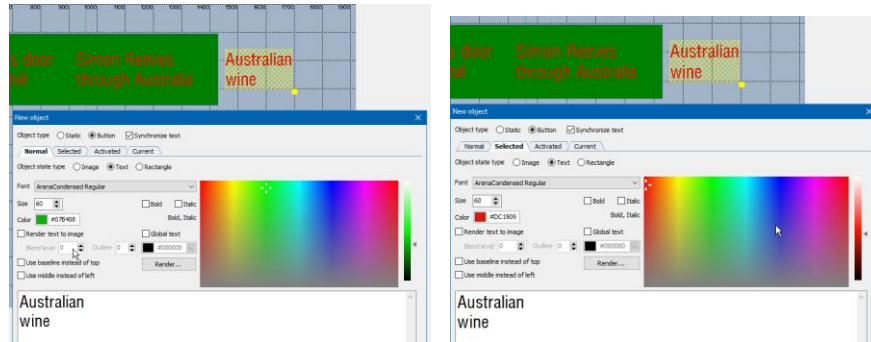
Pressing the down arrow key visits all the individual movie titles and once at the bottom the submenu with “Dwars door Oceanië” titles disappears. (Using up arrow or right/left arrow also makes the submenu disappear).

Moving the carousel to another main documentary series title and activating that button shows different menus as you can see below for the other two buttons (on two different screens).

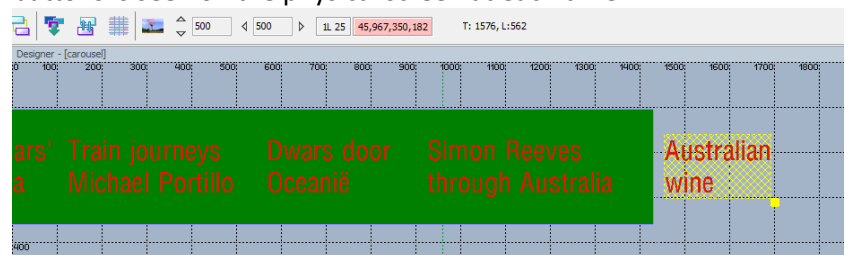


The following steps are taken to create such a disc menu:

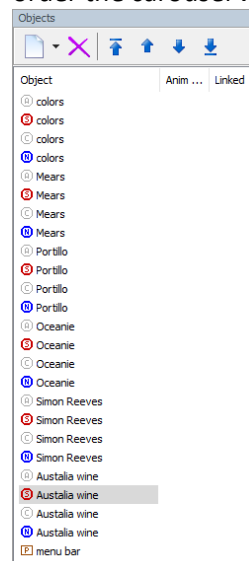
1. First create the carousel menu (called “carousel”) by creating a “source menu” with a green rectangle in which all six buttons will reside. They won’t fit, but we use the Viewport to extend the Designer Window.
2. Ensure the “Normal” and “Selected” states of buttons are visible in the Designer Window.
3. The buttons are created using the BDS feature to add text buttons to a menu. For this we use the menu’s Object Window and add a text button object. The button has light green colour text in “normal” state and red coloured text in its selected state.



4. Arrange all buttons in a horizontal row. They all have the same Top value so they align neatly in a horizontal row.
5. The button row doesn't fit in the 1920 wide screen, so we use the Viewport of the Designer Window to extend the physical screen at the right hand side to allow all buttons to fit in a row. The "Australian Wine" button is placed to the right of the green bar (that ends at physical screen at 1920 pixels). The carousel wizard will later create menus where a subset of buttons is seen on the physical screen at each time.

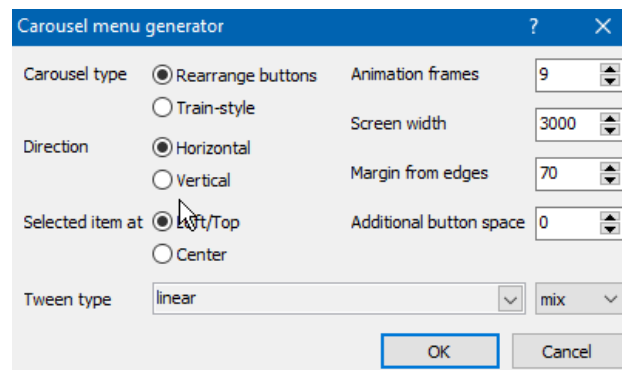


6. All buttons must be set in the correct order (left to right on screen, top to bottom in the Object Window) as that's the order the carousel wizard will sort them into.

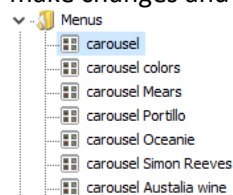


7. Using the "source" menu (called "carousel"), right click to select the "Generate carousel" option. Set the values to
 - a. a never ending carousel (rearrange),
 - b. rotating horizontally with the button at the left.

- c. The rotation animation between buttons (really between displaying two generated menus) is done in 9 frames (of 9/25th of a second).
- d. The buttons at either side of the menu are 70 pixels removed from the left screen edge (you don't see it work properly on the right edge as it assumes that edge is at 3000 pixels instead of 1920).
- e. The virtual screen is set to 3000 to accommodate all buttons. Keeping in mind the margin of 70 pixels either side, the six buttons are divided over the width with identical spacing.
- f. No extra spacing is set between the buttons



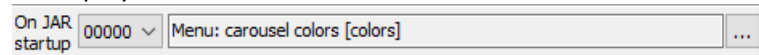
8. Click OK to generate all carousel menus. Those will be used, the “source” menu will not. But keep it in case ou decide to make changes and require another generation.



The generated menu “carousel colors” looks like all the other generated menus except for the one button. In this menu it is the “Colours of Australia” button as left most button. The remainder light green button texts look like buttons but they are plain static images.



9. Set the “On JAR Startup” to display this menu when this JAR file (title) is started. Because it is the only JAR file, it is also “First play” so it will show the “carousel colors” menu.

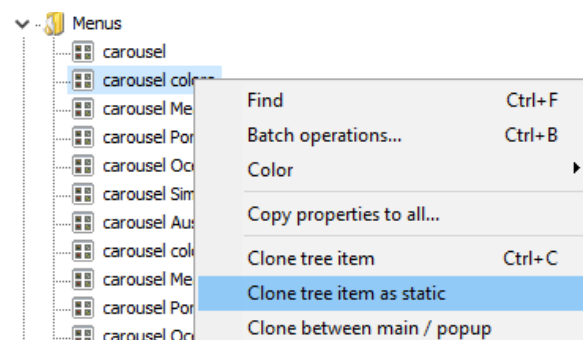


10. Test the current set of carousel menus in the simulator. You should be able to rotate amongst the six text buttons. Repair anything that’s needed before progressing.

We want additional submenus with the titles of each documentary episode when buttons from the carousel are selected. Now is time to add those submenus.

We’ll just show how to do it for the “Colours of Australia”. The steps need to be repeated for the other buttons that have similar submenus.

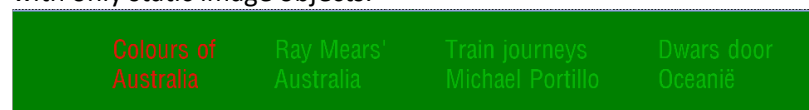
11. Clone the “carousel colors” menu into “carousel colors menu” with only static items.



If done correctly you will have an identical menu layout where all objects are picture elements. The button “Colors of Australia” has also become a static image object in the “normal” state green colour.

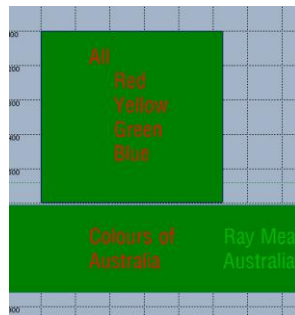


12. But we still want to remind the viewer that “Colours of Australia” was selected and activated. Therefore, select the “Colours of Australia” picture element and recolour it to the same red colour as it had as “Selected” button state on the “carousel color” menu. The end would look like this, but still with only static image objects.



13. Remove all (wizard) animation groups copied by the cloning process (shown in the “anim group” column of the Objects window)
14. Remove the (wizard) menu animation copied by the cloning process (in the “Enter” “Animation 1” and “Animation 2” properties of the cloned menu)

15. Add a new text item with “Play All, Red, Yellow, Green, Blue” as title text of the episodes. Each item on a separate line. Position the text block above the “Colours of Australia” item.



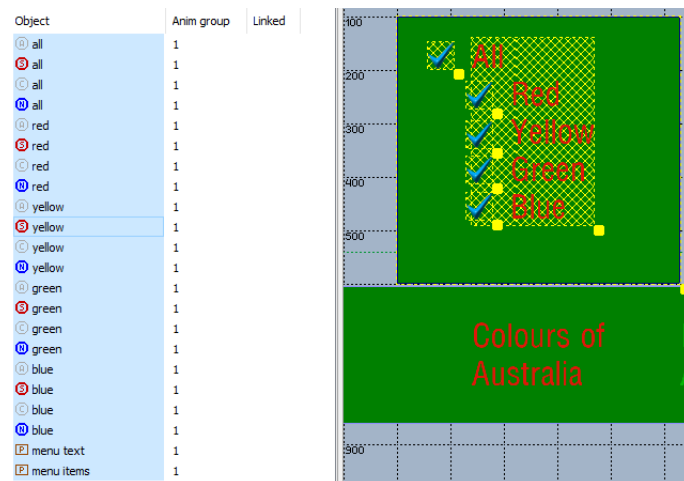
16. Add and copy a button with a checkmark image for its “selected state” for each of the title options. This is how you can select a title or a playlist to play.



17. Specify navigation between these check buttons and the “Press Enter” action to play the documentary movie title that stands next to the button.

Now the submenu is created, all these elements must be joined in a single animation group and that group will slide upwards from the green menu bar.

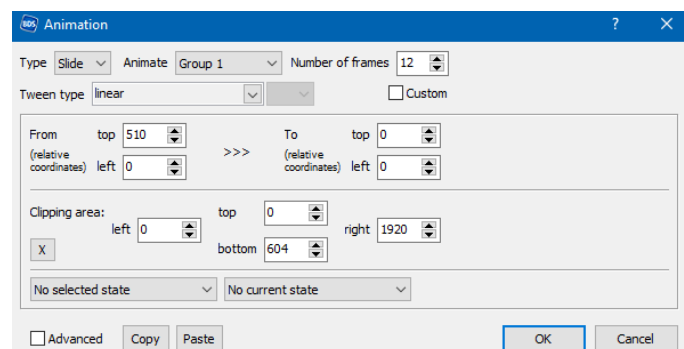
18. Join all objects in the submenu (green rectangle, titles, buttons) to “animation group 1”



19. Add opening menu navigation. The submenu is hidden under the green bar and moves upward until it is shown completely if

the “carousel colors menu” is opened because the “Colours of Australia” button on the main menu bar is pressed.

- a. What is the Top position of the green menu bar? (in our example: 604)
 - b. What is the height of the submenu? (example: 500)
 - c. What is its left position? (example: 100)
20. Animation values are relative to the final “To” position. The menu must rise vertically to its full height from under the main menu bar.
- a. Horizontally nothing happens: “From” and “To” both have Left=100
 - b. Vertically the final “To” position is Top=0 but the start is “From” Top = 510 (500 would do, but better to start well below the main menu bar).
21. The “carousel colors menu” must have an opening animation that reflects the sliding movement.

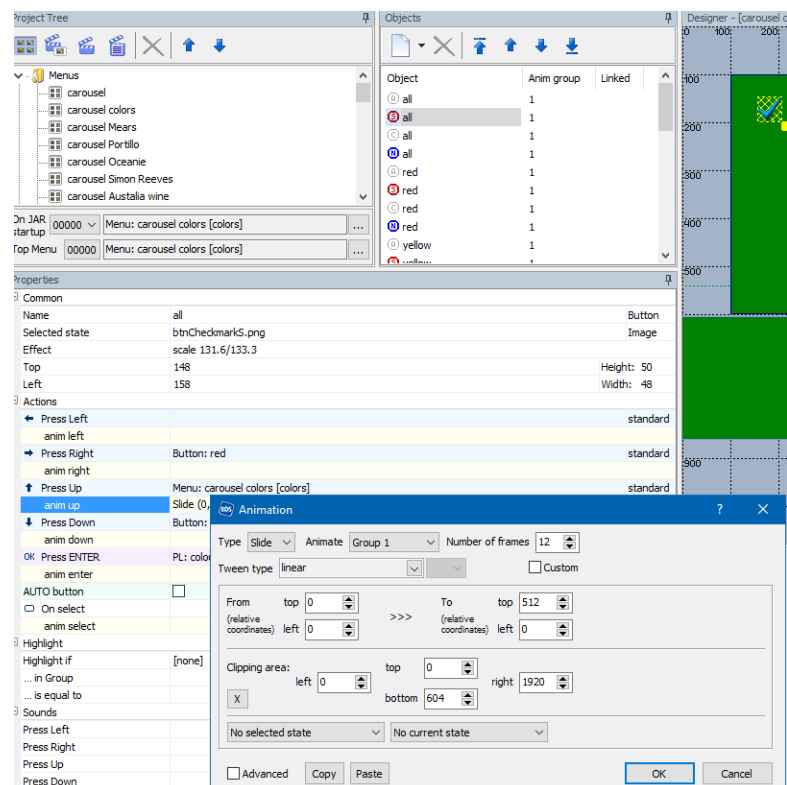


You need to limit the screen area where the animation can be seen. This is the screen part above the menu bar. Hence the clipping area is set over the full width (Left 0-1920) but only the top part of the screen (Top 0 – 604).

22. In the “carousel colors” menu you need to add actions to the “Press Enter” and the “Press Up” properties of the only text button “Colours of Australia” to open the “carousel colors menu” menu using the (only) [act/anm 1] animation.
23. The submenu is can now be shown. Any button selected will start playing the associated documentary title. When it does, the submenu disappears.
24. The submenu must also disappear if no movie is played but some button is pressed. We take the “Press Up” action of the “All” button to do this. Also the “Press Down” on the bottom most submenu list will close the submenu⁴⁵.
To slide the submenu down and out of sight and return to the main menu bar with the “Colours of Australia” button selected again we must do 2 things:
 - The action must reopen the “carousel colors” menu
 - The animation with the action must reverse the sliding movement. This means
 - a setting for “From” Top = 0” to “To” Top=510” (Left remains unchanged again).

⁴⁵ You can add additional or other buttons to achieve the same.

- The same clipping area is needed to avoid the animation to show the submenu to sink under the main menu bar.



25. Check proper functioning by running the simulator. Repair where you went wrong.
26. Repeat these steps for the other main menu bar text buttons on the generated “carousel *” menus.

Project Steps to Take

We will use the results of Project 2 (“MenuBR”, a simple menu) to show some of the simple animations. The menu texts will appear first, by moving in from the bottom, next the buttons appear by fading in. Both animations are executed following each other.

If you want to create this project yourself, the source files can be found in the .zip file mentioned in Appendix A: The user’s guide source files.

Ready to run

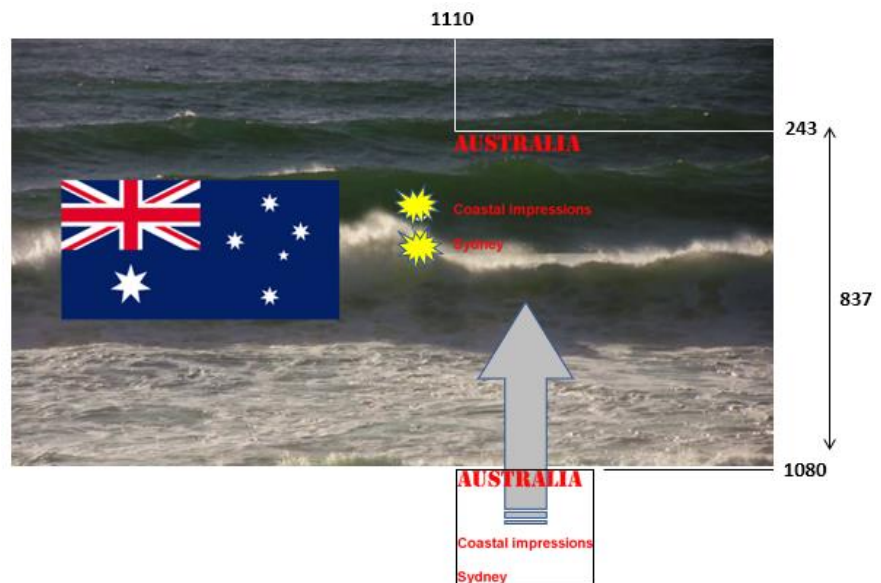
Steps to take are:

1. Copy the project tree \Projects\SimpleAnimationBR to your local BDS project tree (e.g. J:\BDS Projects\SimpleAnimationBR)
2. Copy movie files from the examples folder \Sources\Movies\Demuxed\Australia*.*, “Coasts*.*, menu*.* to your local project folder \SimpleAnimationBR\films
3. Open BDS and the SimpleAnimationBR project by double clicking on the \SimpleAnimationBR\project.bdmd file

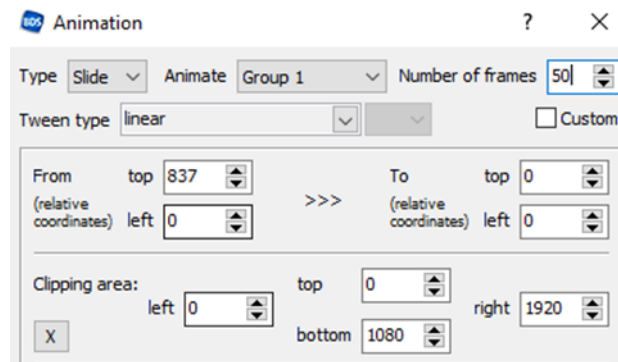
4. Mux the project
5. Experiment!

Build your own

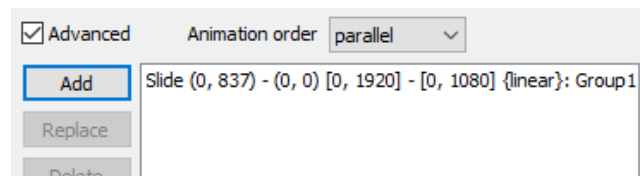
Steps to take to build your own project, based on the “MenuBR” project are:



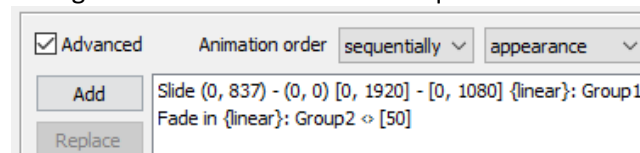
1. Copy the local project tree \MenuBR into a new folder \SimpleAnimationBR
2. Open the copied project.bdmd file (and thereby the SimpleAnimationBR project)
3. Calculate the movement.
The final text object “menu text” is positioned on screen at (left,top)=(1110,243). To move it straight up this means it has to come from the bottom which has value top=1080 . The vertical animation of the object therefore is $1080 - 243 = 837$. The horizontal animation remains $(1110 - 1110 =) 0$.
4. Set the “menu text” as animation group 1 (“menu text” object of the main menu: right mouse click > Animation groups > animation group 1)
5. Define both buttons as animation group 2
6. In properties for “main menu” open the “Enter” segment and fill in “Animation 1” by clicking on the “>” button
 - a. Open the Animation window
 - b. Type = Slide
 - c. Animate = Group 1
 - d. Frames = 50 (about 2 seconds)
 - e. Tween = linear
 - f. From Top=837, Left=0 To Top=0, Left=0



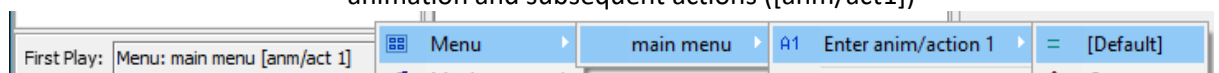
7. Check "Advanced"
8. Click "Add" to make the Slide of the menu text the first item on the list



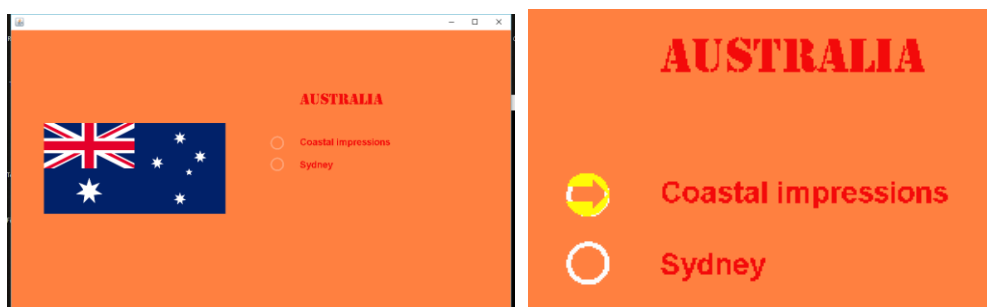
9. Change the "Type" to Fade
10. Animate = Group 2
11. Frames = 50 (time depends on the "Animation engine FPS" setting)
12. Select "Fade in"
13. Click "Add" in the open Advanced segment
14. Change "Animation order" into "sequential"



15. Click "OK" to confirm the animation of the main menu
16. Modify the "FirstPlay" to open the main menu starting with its animation and subsequent actions ([anm/act1])



Now run the animation by pressing the "Simulation" button. You should see the text with "Coasts" and "Sydney" items move upwards and once they are in position, the two menu buttons "Coasts" and "Australia" appear, first in their normal state and then one of them in the selected state.



Movie animation by frame animation (animated movies)

The menu objects animation described in previous sections is by far the most important.

But in addition you can add some “all moving, all dancing” but silent movie animation at the opening of a JAR file (Like when starting a disc and opening the “First Play” (and possibly only) JAR file).

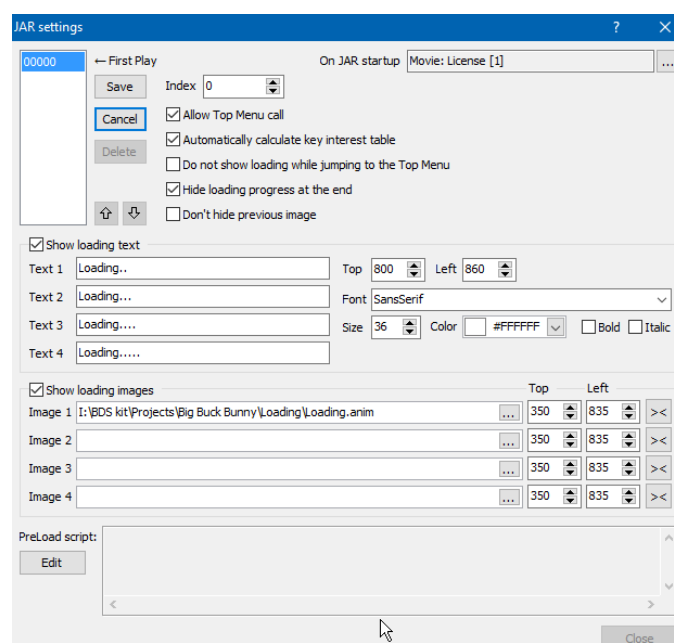
Movie animation differs from menu animation by

- Consisting of individual image frames shown like movie frames
- Using a .anim file to specify the order and duration of images to show
- An animation can repeat indefinitely (rather than just once)
- It is specific for the execution of a JAR file (title). It is not triggered by a press on a button on the remote-control – it is a background animation on top of the background menu movie.

Because it shows images displayed as frames in a movie, it is called “frame animation”. Frame animation uses .anim files that contain the instructions for animation (like “show picture 1, then picture 2 each for 4 frames”). Anywhere where a .png file is allowed in properties of objects in BDS, you can also specify an .anim file. After all – animation is nothing more than flipping through .png pictures.

This opens up a lot of opportunities to show animation. Useable properties are:

- On disc loading (the Project> JAR Settings) the JAR file indicated “First Play” is started. It has either 4 texts that could be shown in succession (like “Loading.” “Loading..” “Loading...” “Loading....” Or some countdown) or 4 “image” fields can have 4 separate images or 1 animation (potentially with a lot of images) for BTS specified.



- Each button's "state" image (cannot see much use here myself – except perhaps "Active" just before the action happens – although it has its own animation fields for that)
- Menu static backgrounds (animation while the menu is visible)

The short animation is unlike the menu animation that only moves a set of objects that themselves don't change. This type of animation with its "all moving" bit is achieved by slightly different images shown in sequence. (See online help section "Animated images and button states"). It looks like you are flipping through a stack of slightly different .png images.

Some image editors have options to create animated .gif files (such as Photoshop CS Extended: use its Windows > Animation panel) but allow such animations also to be exported as sequence of image files – including .png format.

When you use the Designer window to author a menu or button, it shows only the first frame of an animation (as Designer window itself is a static representation of the menu to be).



The illustration above shows four frames (images) from the BDS sample project "Big Buck Bunny". When shown in sequence it appears the bird is flapping its wings without moving its position. (It is seen set in the JAR Setting window that opens for the First Play JAR in the earlier figure).

The .png images that make up the animation are displayed on the screen to imitate a movie. Do not use huge images – recommended image size is no more than 640*480 pixels (a SD size) or less.

For BDS to understand how to show these images as an animated movie, it reads the associated .anim file. For the Big Buck Bunny it contains 8 images (we only showed four of these above):

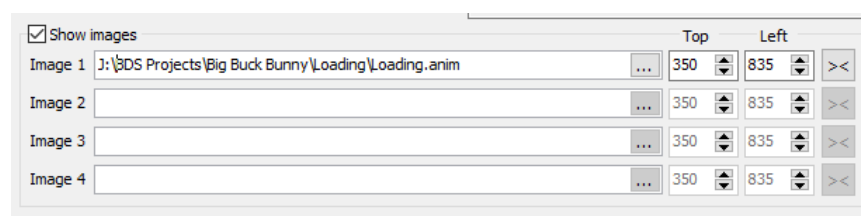
1
1.png
2.png
3.png
4.png
5.png
6.png
7.png
8.png

The animation consists of 8 pictures in .png format, shown in sequence from first to last (1-8). The opening "1" means each image is shown for the duration of 1 frame. Change it to 2 and the animation takes twice

as long and the bird flaps at half speed. In Project Properties > Advanced tab you can set the number of frames per second.

The animation is repeated endlessly, except when after the last image line in the .anim file, a line with only the asterisk "*" character is added: that will stop the animation and freeze on the last image (8.png in the example) that will remain visible on screen.

Because you can specify this type of animation for each JAR file of the project, a change from one JAR file to another will trigger the associated animation. For the Big Buck Bunny project (discussed below) the animation is defined in its "Loading.anim" file as shown earlier.



Because the animation is limited in size, you indicate the top and left pixel position for the images that make up the animation. In the example, the left-top corner of the bird's pictures are set at Left 350 and Top 835 (which for an HD screen means 350 pixels to the left on a range of 0-1919 and 835 pixels downwards from the top on a range of 0-1079).

The button "><" provides centering of the image at 1920*1080 screen depending on real selected image size. It will change the values in the Top and Left fields.

Note

You may define a loading animation to be followed by a movie (like copyright or warning) before the main menu shows (through End Action of the movie). To ensure they are played in that order you must modify the movie "Disabled actions" property to uncheck "Call Top Menu". To disable Top Menu call such movies should be moved outside the JAR (if you are using one JAR). Check the online help "Creating a new project" > "How to create a movie" > "Movie properties" > "Disabled actions".

You can avoid such behavior by using a separate JAR for such movies - check an example at

<https://blu-disc.net/download/examples/SimpleJARs.zip> (you need <https://blu-disc.net/download/examples/Simple.zip> as well)

Examples for D.I.Y. projects

There are a number of examples that can be downloaded from the BDS website that are worth looking into as (most with Russian menu text) examples for your own inspiration or duplication:

- Big Buck Bunny
- Bravo (music concert)

- Kings Speech
- Senna
- The Brest Fortress
- Valerian (known as “Ravian” to the Dutch and as “Valentin” in Scandinavia, “Varerian” in Japan, “Valeriano” in Lithuania and “Walerian” in Poland)

The Big Buck Bunny project is also added in the sample projects of this guide – just to show some of the animation possibilities as well as PIP movies (see “Real PIP example: Big Buck Bunny” on page 398). The project doesn’t have the real movies included – but it has the movie placeholders. The animation can be seen if the project is “simulated”.

For example, in “Big Buck Bunny”:

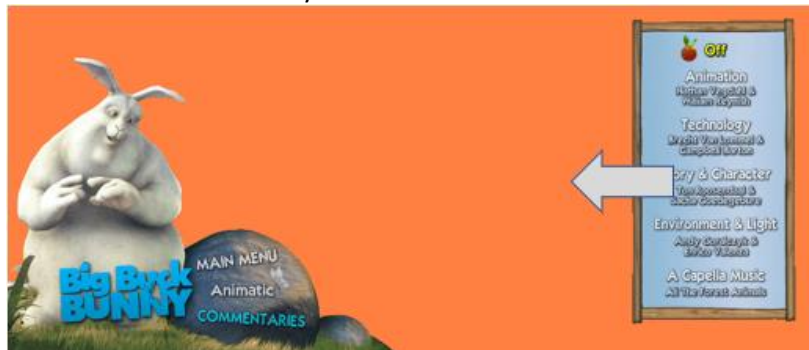
- it opens (Project Settings>JAR Setting for the First Play JAR file) with playing the flapping bird animation
- plays the “license” movie (as FirstPlay item)
- plays “Dynamic HD” movie (the follow-on by End Action of the “license” movie)
- The menu rises up from the bottom of the screen (menu animation, bounce, because the End Action of “Dynamic HD” specifies this menu to show)



- Selecting button “Play movie” makes the menu bounce out of screen and starts the movie
- During the movie a popup menu can be activated by pressing the popup button on the remote-control, which pops up from the bottom and looks like the main menu except that the playing movie button is replaced by a “return to main menu” button.



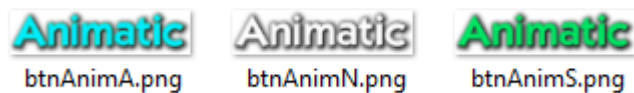
- If the “Main menu” is selected from the popup menu, the popup slides down, the main menu returns and drops/bounces from the top (as it does when it is started from the boot sequence)
- The “Commentaries” popup menu item slides in from the right a submenu with commentaries to choose from. Selecting any one of them makes the apple image 🍏 (shown near “off”) move to the selected item. This is the “Current” button state. The selected commentary is set.



- Unselecting the commentaries (arrow ← button is associated with “jump to popup menu” for all buttons on the commentaries menu) returns to the menu options of the popup menu. This allows you to return to the main menu.
- From the main menu, when you select “extras” up pops a popup menu “Extras” through simple sliding animation. It makes use of the “picture in picture” feature available for BDS MX users (for a lot more about this in Big Buck Bunny, see section “Real PIP example: Big Buck Bunny” on page 398). This feature can be switched on or off – it is indicated by the colour of the little butterfly on top of “Animatic”



- Select any of the options and it highlights your choice like the “Animatic” text does in the figure. This is another way of connecting different images to the “normal” and “selected” state of a button.



(figure above shows the “activated”, “normal” and “selected” state image files of the Animatic button).

By now, you get the drift of what it is you can do with BDS in terms of menu creation. Simple button states with markers or modified text appearance. Popup menus that can be simple animated to move in or out of screen. Main menu items that can do the same. Add some additional animated movie during the disc booting (with may remain if the FirstPlay JAR file (title) ends with opening a menu) and it has become all dancing, all moving, all talking.

The Big Buck Bunny project is discussed more extensively in a separate guide, called “BDS Live” where a number of real life projects are dissected in how they are authored to achieve the effects their menus have.

Part 5: Advanced Operations and programming

Using Registers (simple programming)

This chapter is not about creating another project but to highlight some of the programming capabilities of BDS. This allows the creation of a disc that will act differently when certain conditions are met.

Examples of such discs are those that contain Easter Eggs: hidden programs not shown on menus and triggered by a secret combination of remote-control buttons. The registers can also be used as flags: set by one condition, cleared by others. In between its value is read and action taken depending on its value. An example is creating a “play all” type continuous watching of episode movies, skipping all identical title and end title sequences.

Registers are stored in the volatile memory of the set top player. The BDS Java programs write and retrieve their values. A range of registers can be stored more permanently in the flash memory of the player.⁴⁶ That way their values survive a change of discs.

There are three objects in BDS that allow Programming (Java):

- Multi-actions
- Switch
- Java programming

This chapter will deal foremost with the first two options. Java is cursory dealt with in a future chapter (Programming in Java on page 310).

The Blu-disc Association (BDA) allows a limited number of ways to program a bluray disc: HDMV and BD-J. The BD-J (Blu-Disc Java programming) is the one supported by BDS.

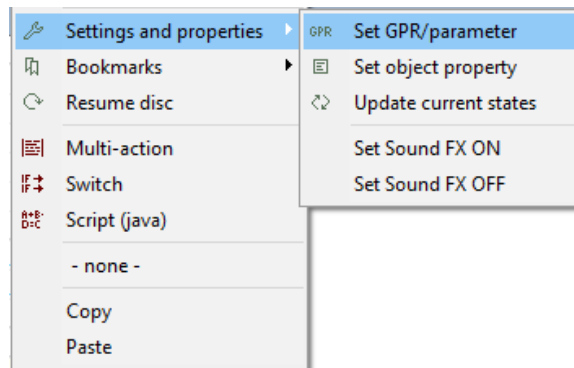
Part of the programming ability is the use of these registers. They are storage locations (read-only in case of PSR) that can be used to store data (as 32-bit integers). That data can become part of a condition that, when met, triggers an action to take place.

There are two types of registers:

- GPR (General Purpose Register) – written and read by BDS programs
- PSR (Player Status Register) – written by the player and indicating its status. Can only be read by BDS programs

To set or read values in these registers, you activate the “>” button at the far right of the action property of a BDS object.

⁴⁶ The BDS storage file is used for it – stored in that flash memory. Its name can be set through Project > Project Properties > General tab.



Writing GPR registers

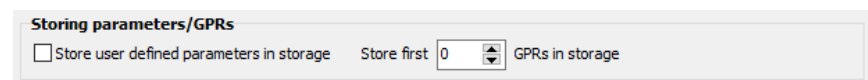
The General Purpose Registers (GPRs) are (an array of) storage locations that can be used by the disc author/programmer to store some values. They can serve as value storing location or as flag to indicate a certain condition. Much of the code behind BDS disc authoring results in Java code that is loaded from disc into the player's memory to execute. The author can write additional Java code that makes use of these registers.

There are 4096 General Purpose Registers (0-4095). They can each store a 32-bit integer value. Registers 1000-4005 and 4091-4095 are reserved for use by BDS Java code itself. Your own programs should not use these registers.

- 1000-1999 – stores the current audio, subtitle and chapter number for each playlist
- 2000-3999 – stores the current play time for resume feature for each playlist
- 4001 – sound FX on/off
- 4003 – 3D mode
- 4005 – “Top menu” pressed flag

That leaves 0-999 and 4006-4090 for your own use: still 1085 registers you can use.

The register values are cleared when a you change BDS discs. The new disc starts with initial value 0 in each register. Through the Project > Project Properties > Advanced tab you can instruct to keep the values of the first N (with $N \leq 999$) registers and restore them when a new BDS disc starts⁴⁷. This way some information can be transferred from the old to the new disc. By default no registers are saved ($N=0$).



A value can be set to any GPR by a single action, a multi-action, a switch or through Java defined for

- a menu (Press Enter) property
- a menu button property (Press Enter)

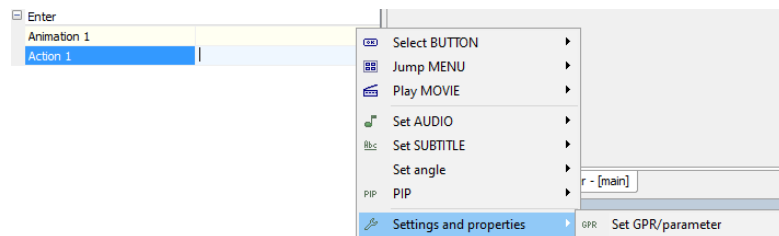
⁴⁷ They are kept in a storage file in flash memory of the player (name can be set in Project > Project Properties > General tab, textbox Storage File Name)

- a remote-control button (“Remote-control Button” section of menu or movie properties)

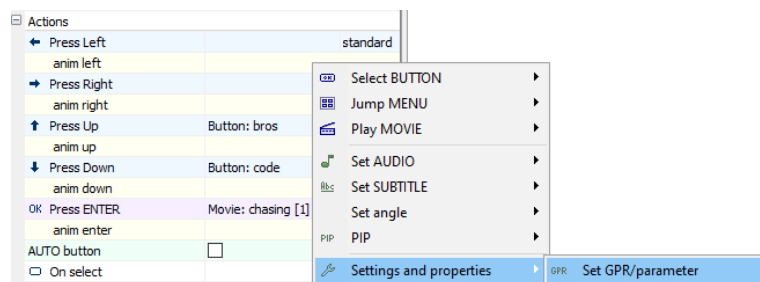
Writing GPR by Multi-action

Clicking on the “>” button at the right end of a menu or button “Press Enter” property allows you to select “Settings & Properties > Set GPR/Parameter” directly or as item in a Multi-action.

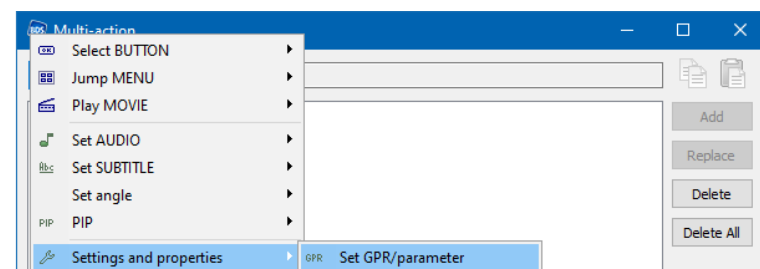
Set a GPR value as a menu Enter action:



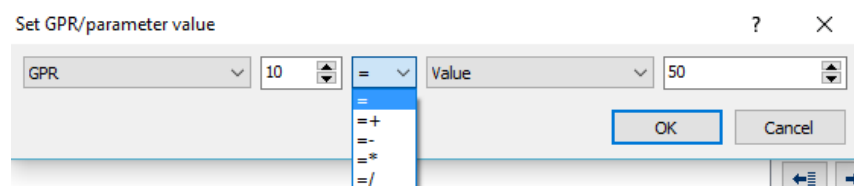
Set a GPR value as a button navigation Press Right action:



Set a GPR value through a Multi-Action:



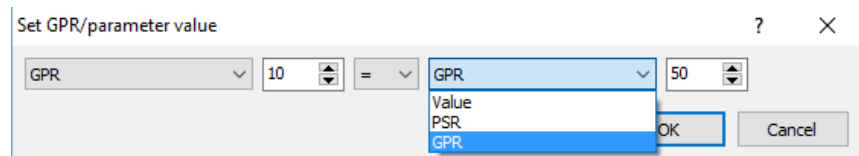
In all cases, once you selected “Set GPR/Parameter” a small window appears where you can set a value for the register. In the figure below, GPR 10 is set to value 50.



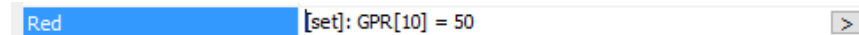
Only use the unreserved registers (0-999, 4006-4090).

Rather than “Value” you can also use the current value stored in another GPR or PSR register (as illustrated below: the value in GPR 50 is used). This is a one-time assignment: if the indicated GPR register

(GPR 50 in the example) changes its value in future, your register (GPR10 in the example) keeps its initial value.



When all is done, click on “OK” . The action property reflects the setting. For example, we use the Red button to set GPR10 to a value of 50:



Writing by Java

This is the more cumbersome way of changing the register value. But coding allows you to do much more than is possible by BDS menu options that provide a limited choice of actions. Before attempting this, make sure you are well versed in programming in Java.

The Java program is stored in a Java archive file. The online help “Script (java)” contains the details. You modify the files that get created in the \<projectname>_SCRIPTS folder. For more info on programming in Java, see the section Programming in Java on page 310. In section Player, GPR or disc states on page 519 the BDS exposed Java functions to set GPR registers can be found.

Reading GPR registers

To retrieve a value from a register, you use the Switch action on the button. Switch uses conditions and their actions to execute only if the conditions are met. Such a condition can be checking the value stored in a GPR register.

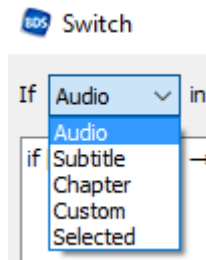
If you need to perform a certain action always, regardless of the condition, use an empty Custom condition.

You can specify the Switch action on the “Press Enter” property of the button that uses the Switch. In our example in this section, we’ll use the “Green” button.

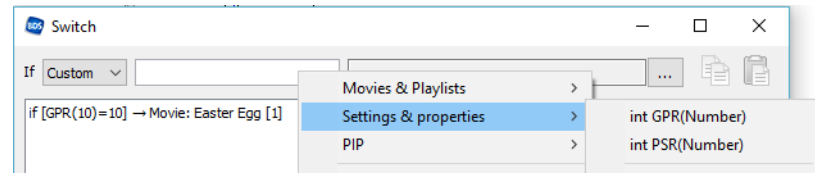


Click on the “>” button at the far right end and a menu of possible options open. Select the “Switch” option.

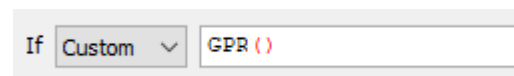
It opens a conditional screen where the If condition is shown. Switch can check on values of currently selected audio track, subtitle track, movie chapter number or if something (menu, button) is currently selected. It also offers the “Custom” option. That’s the one we need to use.



The text box that is to contain the condition is cleared. A right-click opens a dropdown menu from which you select Settings & Properties > int GPR(number)

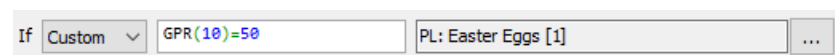


This results in a GPR function without a specific number for the register



This needs to be entered manually (e.g. register 10) as well as the condition (say value 50) and the final box is filled with the action to take if the condition is met. Conditional operators are equals (=), greater than (>), less than (<), greater/equal or less/equal (>=, <=) not equal (!=), AND (&) and OR (|). Note that “not equal” is specified with the mathematical “not” (!) and hence as “!=” rather than “<>”.

The colour-coded green formula indicates the syntax is correct.



Click on the “Add” button to add this condition to the switch.

You can add additional conditions. All of them are executed sequentially until a condition will be met (true) and it will then perform the action specified. It does not look at further switch statements. If no condition is met, a “catch all” action is needed as last line.

You may need to undo the condition in the (multi)action that is executed to avoid it to remain true on all further invocations of this Switch condition (e.g. if Easter eggs are found, reset the condition for future egg hunting).

Finally, you click “OK” to set all switch conditions to the Press Enter properties for the remote-control button.

Red	[set]: GPR[10] = 50
Green	if GPR(10)=50 → PL: Easter Eggs [1][exclusive]

When a movie plays pressing the Red button sets a value of 50 in GPR10. When you press the Green button afterwards (not before!) it checks GPR10 to have value 50 and if so, the Easter Eggs movie is

played. And will play each time the green button is pressed unless somewhere the value in the GPR10 is reset.

Reading by Java

Retrieving values from the register in Java and activating the action that should follow is done the same way as setting a value. In section Player, GPR or disc states on page 519 the BDS exposed Java functions to set GPR registers can be found.

Reading PSR registers

There are 128 Player Status Registers (0-127). They contain information about the player and the disc played. They are written into by the set top player but can be read by the BDS Java files. Some Java functions exposed by BDS implicitly read these values.

Some registers are used and others are “reserved for future” (45-127) use. A PSR register value can be read using a Switch or Java.

The content of each of the PSR registers is described in the online help section “PSR registers”. For example, PSR 16 contains the current audio track, PSR 5 the current chapter of the playing movie.

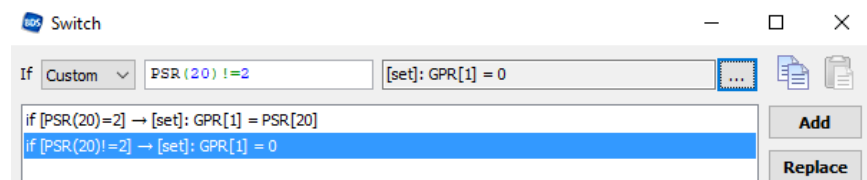
Reading them can be of help to the disc programmer. For example, by reading the value in PSR 5 you know what chapter currently plays.

Reading by Switch

The value of a PSR register can be read using the Switch condition, specifying what to do if the condition is met. The value can be compared using logical comparisons like “equal” (=), “not equal” (!=) and others.

What to do when the condition is met is selected from the menu choices that are shown when the “...” button is clicked.

The switch setting below inspects PSR 20 (containing the region code). If it is Region 2, the GPR 1 register is set to 2, in all other cases it is set to 0.



Note: if the “Exclusive” checkbox is checked in the Switch window, only conditions are tested until the first condition that evaluates to true.

Reading by Java

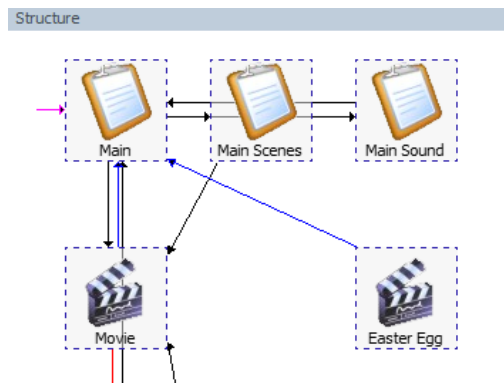
As part of a Java, you have the most flexible option of using the value of the PSR. It does require you are well versed in writing Java code, which is beyond the objectives of this user’s guide.

See “Writing by Java” for accessing PSR registers on page 290.

Project 9: Easter Eggs

Project goal

Easter eggs are parts of a DVD or bluray disc that cannot be accessed through buttons next to items on the menus. Rather, a “secret” combination of buttons on the remote-control must be pressed. When you show the “Structure” window it is clear that “Easter Egg” is not connected to anything and cannot be selected from any menu item. It only has an End Action that returns to the main menu.



But by pressing a certain sequence of buttons on the remote-control it is activated. For example, pressing the red and green button in succession activates the Easter egg.

This project

- shows the use of General Purpose Registers and their use as flag or temporary storage space.

There are other ways to accomplish this than registers – like using a UDV (user defined variable – we’ll look at that later) instead of a register.

To create an Easter egg, you need to:

1. specify the “secret” buttons on the remote
2. each must set or update a GPR register of the player
3. the final button must check whether the GPR register contains the expected value and if so, perform the Easter egg action.
4. The final button must also reset the expected value for a fresh start

To focus on this one aspect, we reuse the results of Project 2, the “MenuBR” project.

If you want to create this project yourself, the source files can be found in the .zip file mentioned in Appendix A: The user’s guide source files.

Project steps to take

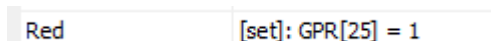
Ready to run

1. Copy the project folder \Projects\EasterEggsBR of the examples .zip to your local BDS projects (e.g. J:\BDS Projects) as a new folder in the same BDS projects folder.
2. Copy the easter egg file from \Sources\movies\demuxed\Tutorial*. *you're your local project's folder \EasterEggsBR\films
3. Mux the project
4. Experiment!

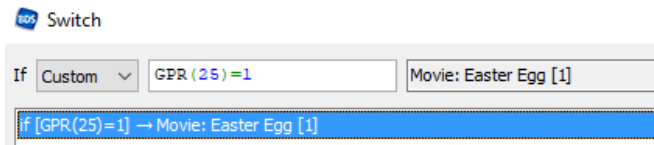
Build your own

Steps to take:

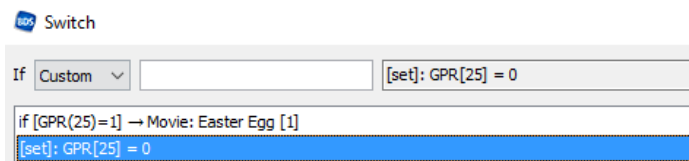
1. Copy the project folder \MenuBR of your local BDS projects (e.g. J:\BDS Projects\MenuBR) as a new folder in the same BDS projects folder. Rename it to \EasterEggsBR).
2. Copy the movie files from \Sources\movies\demuxed\Tutorial*. * to \EasterEggsBR\films
3. Open the project by double clicking on the \EasterEggsBR\project.bdmd file
4. Add a movie placeholder "Easter Egg". Its video and audio stream are the \films\Tutorial.264 and .ac3 files.
5. Set the End Action and Popup Menu for the "Easter Egg" to return to the main menu.
6. In the menu properties set an action to the "Red" button. As it must change a GPR select as action "Settings and properties > Set GPR" and in the window that opens, set GPR 25 to value 1. Click on OK to close the settings. This should now look like the figure below.



7. The "Green" button gets a Switch action. Only Switch actions can perform conditional checks. Here it must check whether the red button has set GPR 25 to 1. If so, it plays the Easter egg movie. Therefore 2 steps are taken:
 - a. the "Exclusive" checkbox of the Switch window must be unchecked (all conditions must be checked) – important to reset the GPR value once the Easter Egg is found.
 - b. To check for the GPR value, select a "custom" condition. From the right-mouse click menu select "Settings and properties" > "int GPR(number)". Manually modify the condition to GPR(25) and value 1.
 - c. The action to perform if the condition is satisfied is specified in the action box. Click on "..." at the right of the action box and select "Play movie" > "Easter Egg" >Chapter 1
Click "Add" to add the condition to the Switch collection.



- d. Regardless of what the value of GPR 25 is, it must be reset to 0. This way the Easter egg is not shown anymore unless the red button is pressed first again. If the If condition box remains empty there is no condition and its action is always executed (Remember the “Exclusive” checkbox is unchecked for this purpose). Click on “Add” to add the action to the Switch collection.



- e. Close the Switch. The red and green button actions should now look like this:

Red	[set]: GPR[25] = 1
Green	[SWITCH]

8. Simulate the project. The red button is replaced by the F1 button on your keyboard, the green one with F2. Check to see if F1 – F2 gives the “play of Easter eggs” and any other combination (such as F2 on its own or F2-F1) will not do anything.
9. You may find a loophole: F1 – F3 – F2 also runs the Easter egg since any key but F2 does not reset the register. The F3 in the middle does nothing and the value 1 set by F1 still stands. Think about how you can avoid this and make the Easter egg code more robust.

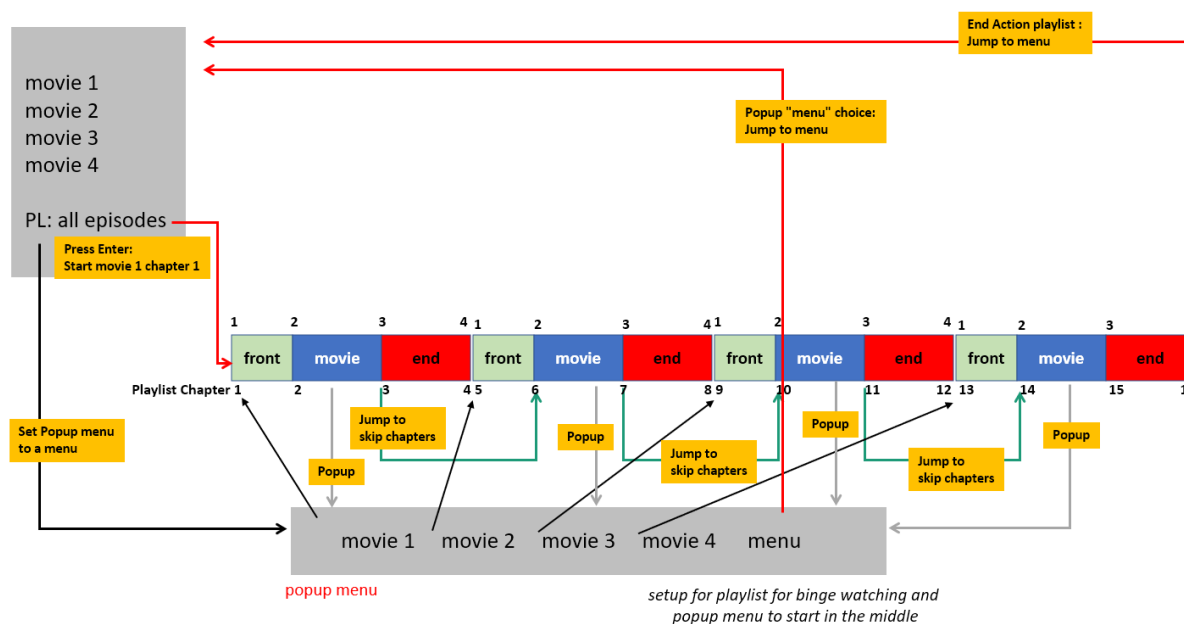
Project Goal

How can you skip these repeated sequences? Two options. The easy way out is to manually do it yourself each time you watch:

1. Create a chapter mark (chapter 2) following the opening credits (and perhaps “previously” recaps) and before the start of the end credits (chapter N-1) and end of the episode (Chapter N).
2. Use your remote-control to manually skip to chapter 2 for each episode that starts playing on the playlist
3. Watch the episode
4. Use your remote-control to skip over the end titles (move from chapter N-1 to chapter N, causing the playlist to start with the next episode)
5. Repeat all steps 2 – 4

Method 1: Use a playlist with direct editing scenes (Binge Watching)

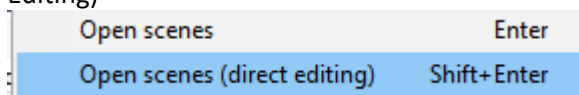
The fastest and least complicated way is to create a playlist of all episode movies and use chapter jumping to skip the unwanted parts



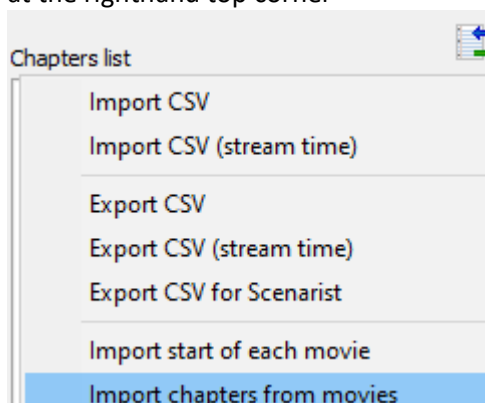
This method needs no special programming or register use and may be enough for what you want. But “better” options follow, using registers.

These are the steps:

1. In each movie you open the Scenes window and only add chapters at the points where
 - a. the opening “previously” and titles end
 - b. the end credits start
2. Create a playlist with all episode movies
3. Right-click on the playlist and select “Open Scenes (Direct Editing)”



4. In the opened scenes window, import all chapters of all movies of the playlist by selecting this option from the button at the righthand top corner

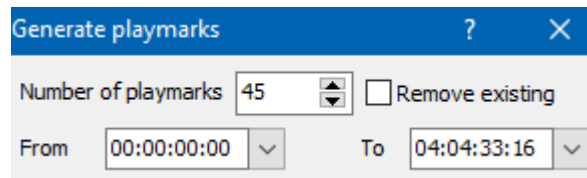


5. If you executed the first step properly, only a few chapters are shown of each movie. Click on any chapter to find which one starts the end titles. Remember the chapter number. Then click on a next chapter to see if that starts the movie after the opening titles have ended.
6. Go to the “start end titles” chapter, right click and specify “Play movie” and select the playlist name and the chapter that starts after the opening titles. (Example: jump from chapter 10 to 13 on playlist “fortuyn”)

☒ 09. 00:42:38:23
☒ 10. 00:47:48:08 - PL: fortuyn [13]
☒ 11. 00:48:15:09
☒ 12. 00:48:22:05
☒ 13. 00:49:37:14
☒ 14. 00:53:23:08

7. Repeat this for every jump from end-of-episode to start-of-next-episode, circumventing the end and opening titles.

Once done, you can revisit each episode individually and add as many chapters as you want (these do not show in the playlist). And in the playlist you can also add chapters manually or generated – as long as you keep the ones already there untouched (so uncheck the “Remove existing” box if you use playmark generation)



This method works fine in forward direction. However, if the viewer jumps backwards in chapters by remote control, he will visit the chapters that start the episode or that will show the endtitles.

Extra

You may want to offer additional service to the viewer by adding a popup menu for the playlist. This popup shows the proper start of each episode. So if someone restarts the playlist but wants to continue from episode 4, he opens the popup window, selects chapter 4 (that triggers a “Play movie” action from the playlist at the right chapter).

You need to remove the popup window after selecting the episode of course, so the “Press ENTER” button action is a multi-action one that starts the movie but also hides the popup menu.

For example a button for Episode 4 may start the playlist from chapter 11 onwards and closes the popup (Jump Popop > [Close popup anim]).

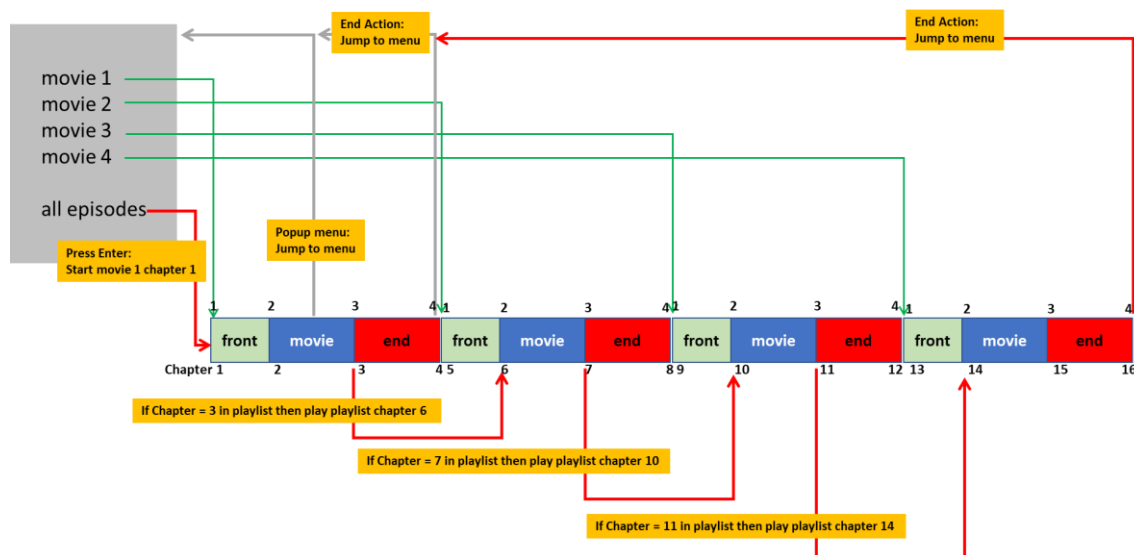
OK Press ENTER	[MA]: PL: fortuyn [11]; [close popup anim]
----------------	--

Method 2: Use a playlist and “Action Every Second”

Episode watching can be done through a play list that combines all episodes. To skip the end titles of one episode and subsequent opening titles of the following episodes, requires the use of the exclusive Switch action. See section “Switch” on page 203.

For some setup condition multiple actions must be executed – for this the Multi-action (see section “Multi-action” on page 203) comes into play.

The possible programming is indicated in the figure below.



The green arrows indicate the normal flow when a menu option (the menu is shown as grey block) selects a single episode with 4 chapters (1=start, 2=start of episode, 3=start end titles, 4=end). Only for the first episode we show the “Popup menu” and “End Action” but they must be present in all movies.

The red arrows indicate the sequential episode viewing skipping end and opening titles. It looks like a single long movie with one chapter at the start of each episode following the opening titles.

In the orange coloured blocks the Switch or Multi-action actions are specified.

For the example we use the figure above as illustration for 4 episodes in a row). One of the steps, the playlist “Action Every Second” condition ensures only the blue coloured movie parts (each starting with a chapter mark) are played. Therefore some of the chapters present in the playlist must be skipped during playback. In the schema below only the blue coloured chapter numbers are part of the playback:

Movie	playlist	chapter
Movie 1	1 2 3 4	
Movie 2	5 6 7 8	
Movie 3	9 10 11 12	
Movie 4	13 14 15 16	

At the start movie 1 will run and chapters 1 and 2 are played. Reaching chapter 3 it contains a jump to chapter 6, skipping the intermediate chapters 4 and 5. The same way reaching chapter 7 forces a jump to 10, 11 to 14 and then the final episode ends including the end titles that start at chapter 15.

Project Steps to take

You can build such an episode system based on a playlist yourself of you can use the “ready to run” system.

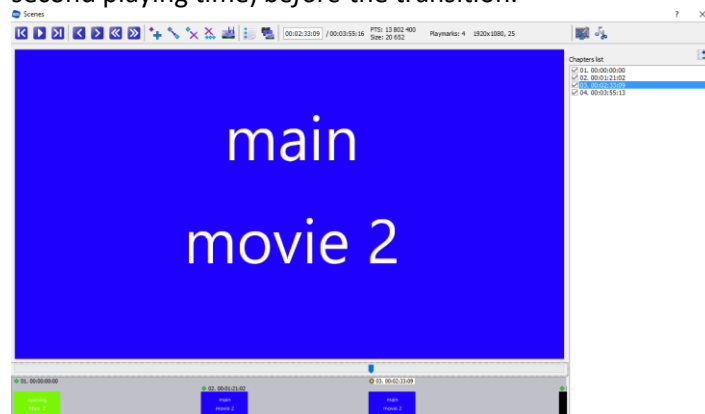
Ready to run

To build the project directly, perform the following steps:

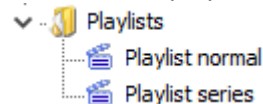
1. Copy the folder tree from \Sources\Projects\EpisodesBR to your project's default folder (e.g. J:\BDS Projects\EpisodesBR)
2. Copy a special set of 4 movies (with a green coloured opening, blue coloured episode and red coloured end titles) from \Sources\Additional stuff\movies\demuxed\movie*.* (.264 and .ac3 types)
3. As background menu movie use black silent 5s 1920x1080 25 fps.264 and .ac3 from \Sources\movies\demuxed. Copy it also to the project's \films folder.
4. Start the project and BDS by double clicking on second project file "2 episodes by playlist.bdmd" project file.

Build your own

1. Create a new project called "EpisodeBR". Add folders \buttons, \original sources and \films to the project folder (you may want to save the project under another name than the default "project.bdmd", e.g. "2 episodes by playlist.bdmd")
2. Copy the film movie files movie*.264 and .ac3 from \Sources\Additional stuff\movies\demuxed to the project's \films folder
3. Copy the main menu movie "black silent 5s 1920x1080 25 fps.264" and .ac3 from \Sources\movies\demuxed folder to the project's \films folder.
4. Copy the simple menu from \Sources\project objects\original sources\episodes playlist menu.psd to your project's folder \original sources.
5. Copy the one button image we need from \Sources\Project objects\buttons\ArrowS.png to the project's \buttons folder.
6. Create 4 movies (movie 1, movie 2, movie 3, movie 4) in the Movies branch of the Project Tree
7. Set the video and audio streams to the respective movie files in the \films folder
8. For each movie add 4 chapters through the "Scenes" property. The chapter 2 is positioned at the transition from opening to main episode. Chapter 3 at the transition from main episode to end titles. Set this chapter at least about 30 frames (1 second playing time) *before* the transition.



- a. Both the “Popup menu” and “End Action” of all four movies have the same action: Jump to menu “episodes playlist menu”
9. Import the \original sources\episodes playlist menu.psd to create a new menu “episodes playlist menu”
10. Set the menu movie stream of the menu to \films\black silent 5s 1920x1080 25 fps.264 and .ac3
11. Define a new play list Playlist normal
12. Define a new play list Playlist series

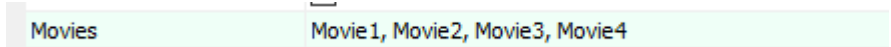


13. Import the chapter marks of the individual movies into the playlists. For this, open the playlist “Scenes” and click on the

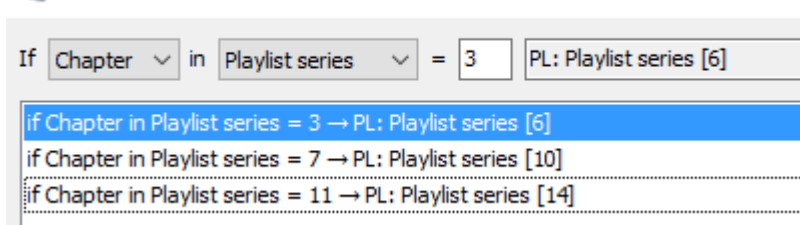
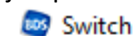


button (top right corner) and select “Import chapters from movies”

14. Set “movies” property of “Playlist normal” as well as “Playlist series” to all episode movies

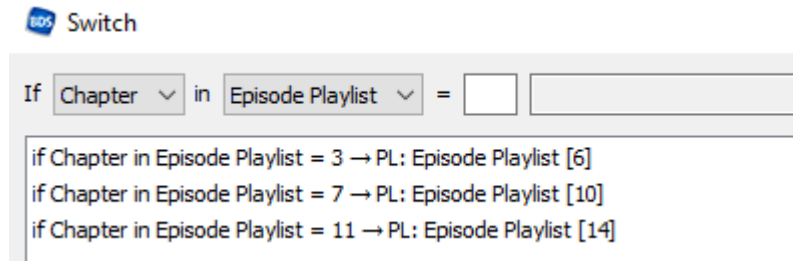


- a. Set “popup menu” of both playlists to the same action: Jump MENU > episodes playlist menu
15. The “End Action” of both playlists has the same action as the previous step
16. The Playlist series must check whether the movie plays only up to the end of an episode. It then skips the end titles and opening titles. It therefore also has a “Action every second” as an exclusive Switch that jumps to the next chapter of a main episode, thereby skipping end titles and opening titles. Because the check is made every second, the chapter mark that ends an episode main part must be set about 30 frames earlier than the actual transition. This way the end titles never show regardless when the check is made. The Switch contains a number of conditions to perform the jump over closing and opening titles:



17. Add buttons to all menu items through the Objects window and the “New object” button. They only have a visible image for their “Selected” state. The image is found in \buttons\ArrowS.png
18. Add navigation between the buttons
19. Add “Press Enter” action on each movie to play that one movie starting at chapter 1
20. Add “Press Enter” action on the Playlist normal to play the playlist normal starting at chapter 1

21. Add “Press Enter” action on the Playlist episodes to play the playlist episodes starting at chapter 1
22. Set “Action Every Second” of the EpisodePlaylist to an exclusive Switch with conditions:
 - a. If chapter = 3 in EpisodePlaylist then Play movie EpisodePlaylist chapter 6 (skip closing and opening titles)
 - b. If chapter = 7 in Episode Playlist then Play movie Episode Playlist chapter 10
 - c. ... etc for remaining movies
 - d. Allow the last episode to run to its end (incl. end titles)



Because the action is executed every second, when a viewer moves backwards into a “forbidden” chapter, he is quickly moved out of it (in forward direction). The problem may be that his backwards movement is not quick enough and the code will not allow him to move to a previous episode.

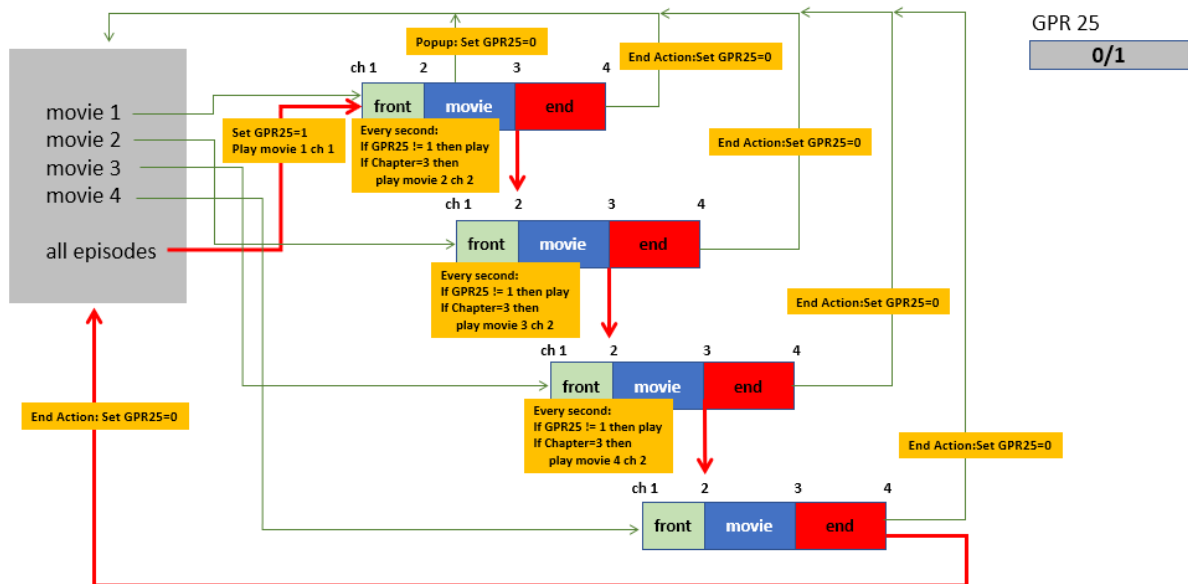
Method 3: chain individual movies

Rather than using a playlist, you may connect one episode with the next whereby the closing and subsequent opening titles are skipped.

To indicate that these conditions must be checked, a GPR register is needed to be used as a flag: if it is set to 1 you need to skip titles, if it is set to 0 you play a regular single episode with its opening and closing titles.

This requires a different setup in setting the GPR(25) register and each movie must have a “Action Each Second” to find out when the end titles are starting and it is time to jump to the next episode.

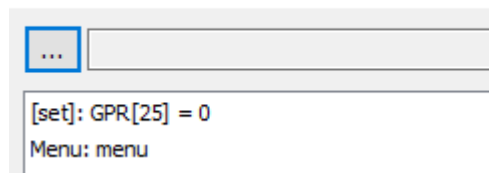
The general logic is illustrated in the figure below.



These are the steps to take:

1. All movies have a multi-action when they are interrupted (popup menu) or normally finish (End Action). This is the same as in Method 1.
 - a. Set GPR 25 to 0
 - b. Jump to menu

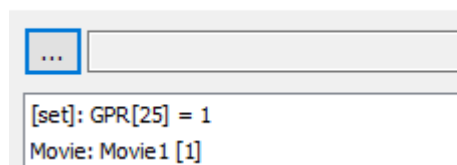
Multi-action



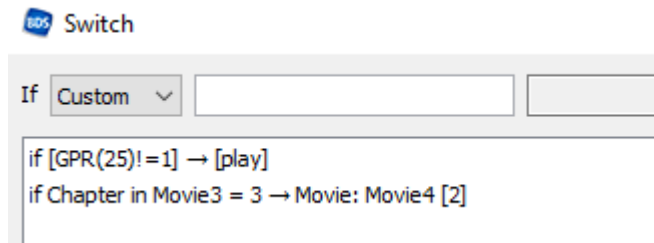
Only for the first movie the popup is shown to avoid cluttering the figure.

2. Menu button "All Episodes" has a Multi-action:
 - a. Set GPR 25 to value 1
 - b. Start playing movie 1 from the beginning (chapter 1)
 This step is also similar to Method 1 (for the playlist button).

Multi-action



3. Every movie has an Action Every Second that is an exclusive Switch:
 - a. checks the value of GPR 25 (or any other available register). If not equal to 1, simply play
 - b. if chapter number is 3 (start of the end titles) start playing the next movie from chapter 2 (after the opening titles).

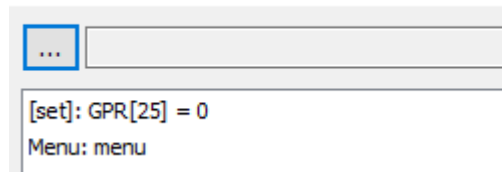


Note that being an exclusive Switch, if condition (a) is not met, automatically it means GPR 25 is equal to 1 and therefore the action (b) is executed.

This step is by far the most time consuming to program – especially if the chapter number just before the end titles start, is different for each episode.

4. The final movie does not have an Action Every Second and plays to the end (including end titles). Its End Action resets GPR 25 to value 0 and jumps to the menu.

Multi-action



We leave the precise steps to create such a project to the reader. It should give no problems if you could copy our steps up to here. A project file “1 episodes by movie.bdmd” is provided when you copy the \Projects\EpisodesBR project from the examples .zip file and provides also the 4 movies and the menu movie (as explained in the steps of the previous method).

Method 4: movie chapter action

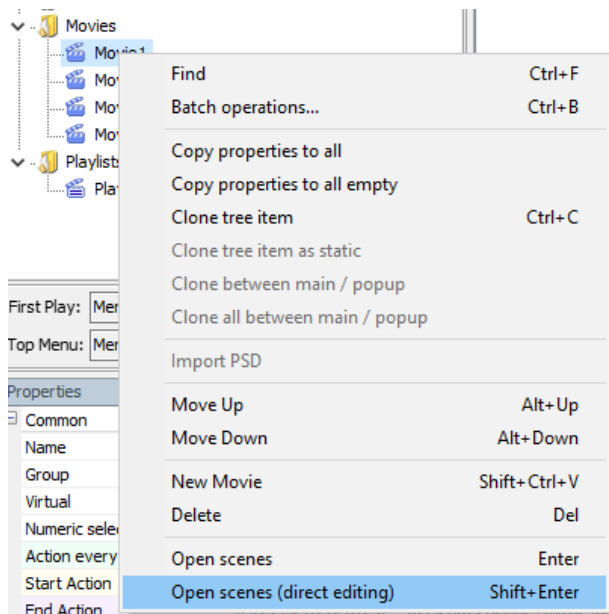
A fourth way is a variation on the third method. It is movie based and therefore a GPR is used to flag the situation of whether we are watching a movie proper or as part of an episode series.

- The first episode to watch sets the “episodes watching” flag: e.g. it sets the value in GPR 25 to 1
- At the chapter where the end titles start, a Switch action is performed that jumps to the next movie’s chapter 2 (skipping the ending titles and next movie’s opening titles)
- The final episode has no chapter action and shows the end titles.

This method seems to have a slow transition between episodes (4-5 seconds) whereas the earlier two methods have a much smaller transition period.

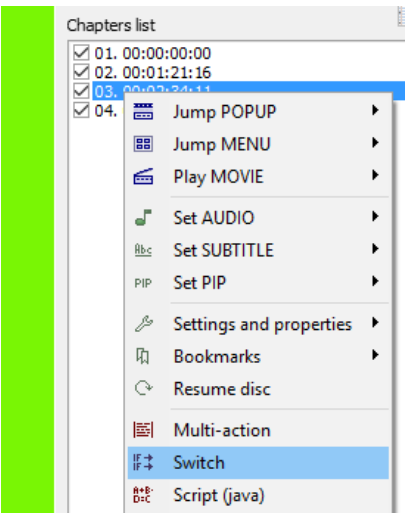
A chapter action is specified at the specific chapter. The only way to get to the chapters is using the “Scenes” playmarks of a movie. You do not open it the usual way (clicking on the “Scenes” property of a movie) but by selecting the movie from the Project Tree. Right clicking

on it opens a dropdown menu from which you select “Open Scenes (direct editing)”.

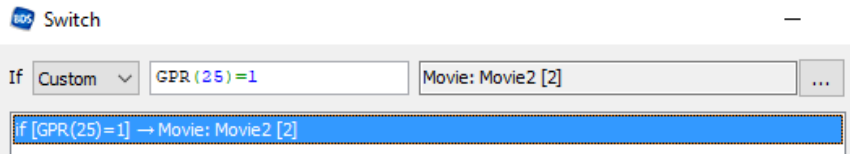


This opens the familiar Scenes windows with all playmarks of the movie. However, you can now edit the chapter properties.

Select the chapter that starts the end titles from the chapter list at the right-hand side next to the movie preview. Again, a dropdown list shows when you right-click on it.

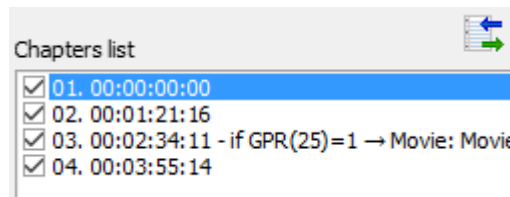


You add a Switch statement that checks for the GPR 25 value. If it is set to 1, we’re episode watching, and therefore must jump to the next movie, starting at chapter 2.



The action to take when the chapter is reached is indicated in the chapter list. It also shows when you open the Scenes window through

the Scenes property of the movie, but this does not allow you to change it.



You perform each of these steps for all movies that are part of the episode series.

Note that while using the “Direct Editing” mode, assignments at a particular chapter to continue “Play Movie” from another chapter, may go wrong if you also add new chapter marks. The chapter you want to start may not show, rather N+1 is shown. It should not, but it does. Workaround: close the scenes window and reopen it using a new “Direct Editing”.

A sample project is given as \Projects\EpisodesBR\3 episodes by chapter switch.bdmd. The same steps as for the Method 1 or 2 project need to be taken to ensure the movie files are copied into the project’s \films folder and buttons are available in the \buttons folder.

The Reverse: cutting up a long movie into parts

The more usual operation is to connect separate movies (each an episode) into a long continuous movie. This has been discussed in the previous sections.

But what if you have one long movie that you like to split in several parts?

One way is to use a video editor and create separate parts that can then be watched individually or as a long movie using the methods of the previous sections.

Using BDS Multi-action and Switch

The other way is using BDS to do it for you using the following steps:

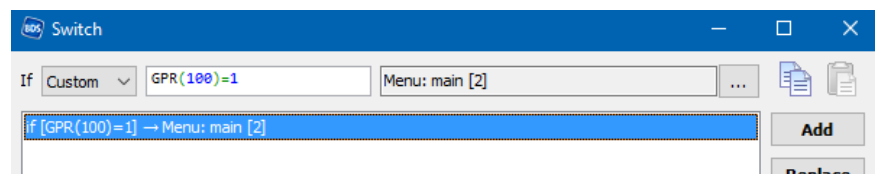
- Add a chapter at all spots in the movie that is the start of a section: name them Ch1, Ch2, Ch3, etc.
- Add buttons on the menu that have an “On ENTER” action to play the movie from those chapter points.
- Add additional chapters just before the section start chapters. These chapters will be the “exit points” if a single section is played: name them Ch1E, ch2E, ch3E, etc.
- Add a button with an “On ENTER” action to play the movie from chapter 1 (Ch1) until the end

How can you differentiate between “run movie” and “run a single section”? This is where the GPR registers come in to flag the choice. Let’s take GPR 100 with two values: 0 for “run movie” and “1” for “run a single section”.

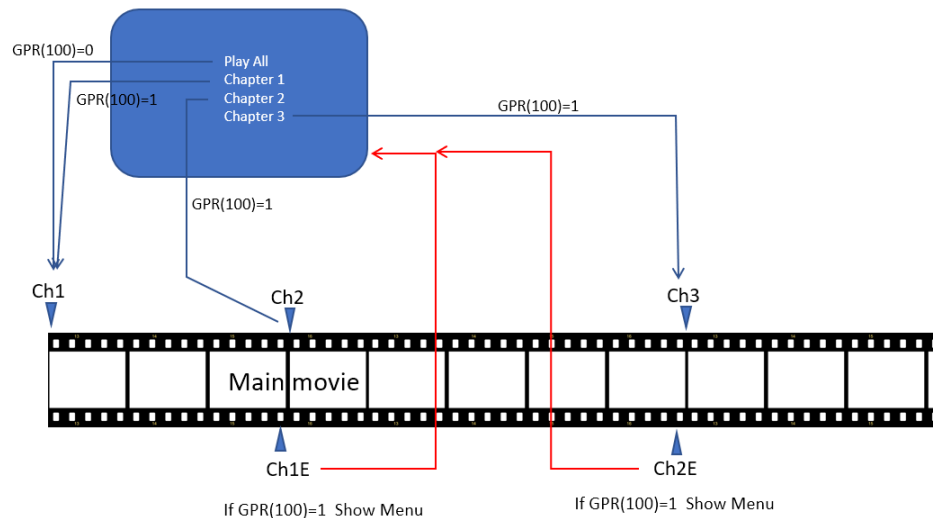
- Each section button gets a MultiAction with two actions:
 - Set GPR 100 = 1
 - Play movie from Ch n ($n=1,2,3...$)
- The “play movie” button gets a MultiAction with also two actions:
 - Set GPR 100 = 0
 - Play movie from Ch1

Next we open the movie’s Scenes for Direct Editing (movie right click > Open Scenes (Direct Editing))

- For each of the section ends chapters (Ch n E) add a Switch action (right click on a chapter in the chapters list):
 - check condition If GPR(100)=1 Then show menu
 - no action (i.e. movie continues playing) in all other cases



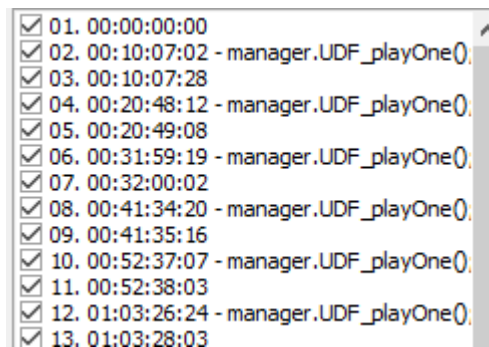
The reason you need ChnE chapters just before the start of the next section is that the switch condition is tested when you start the chapter position. Adding the switch to the section start at Chn means that switch is executed at that time and because the GPR100 has a value of 1 and therefore you return to the menu without starting the section. By having a section start without a Switch condition allows the section to run. Upto the moment the end of the section is reached where a ChnE is encountered where the Switch condition ends the movie and returns to the menu.



Using Java

Rather than using MultiAction and Switch statements, the use of a user defined function (UDF – see the next section Programming in Java on page 310) can be used to replace the Switch statement. The Multi-action for the “On ENTER” action of the buttons remains.

But each Switch action in the “Direct Editing” mode for Scenes we simply refer to the UDF in each of the ChnE chapters at the end of a section. The starting points of each section are the chapters without any action.



The code of the UDF called “UDF_playOne” is entered using Project > Project Properties > Functions tab where the function is “Added” (see User Defined Functions (UDF) on page 320 for details). The script is very simple but the same for all:


```

public void UDF_playOne() {
    int j;
    long lngDuration;

    if (getGPR(100)==1) {
        setGPR(100,0);
        lngDuration = getDurationInSeconds();
        j = (int) lngDuration;
        jumpTime(j);
    }
}

```

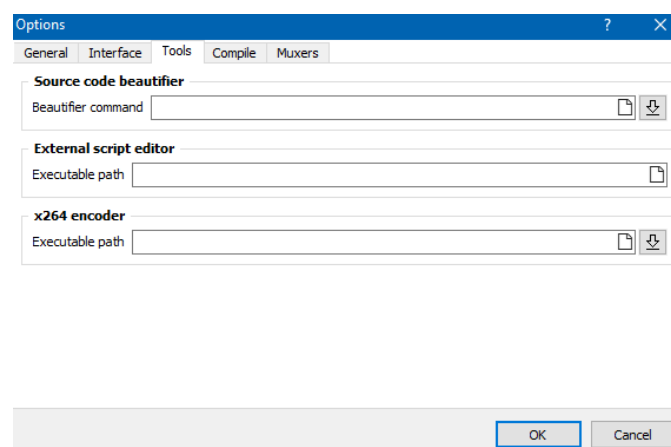
It checks if GPR100 is set to 1 and if it is, it retrieves the runtime of the entire movie and jumps to its end so the “End Action” of the movie can take over.

The BDS function `getDurationInSeconds` requires a “long” variable to store the number. But the `jumpTime` function only accepts short values, called integers. The `(int)` is called “type casting” to squeeze a long value into an integer. In this case that is no problem since the length of a movie never exceeds 2^{32} seconds (over a million hours).

Programming in Java

The actions you specify through selection dropdown menus at the action properties of objects, end up in the action property line. If they do most of what you want but not quite – you may try to write some (simple) Java code to get done what you want. A list of available BDS functions in Java is found in Appendix F: Java programming on page 504.

As of BDS V4.6.2089 you may select to use your favourite text editor to write your Java code. You may use a language-sensitive editor for this that knows about valid Java constructs. If you want to use your own editor, you need to specify its location in Tools > Options > Tools under “External Script Editor”. If you got a program to “beautify” your code (giving it proper indents) that can be specified here too.



BDA specifies to use Java as programming language for BD-J type bluray disc. This is a full fledged programming language and although it has similarities with the web page scripting language Javascript, it is a totally different beast.

For simple programs you don't need to know very much about Java (although it helps) – experience in any programming language is enough for the “frame of mind” to setup the logics of a program. The internet has many introductory courses on Java if you google on terms like “basic Java”, “course Java” or more specific “Java variables” etc.

The written Java code is stored in a Java Archive and stored in the \<projectname>_SCRIPTS folder of the disc file structure.

Pur sang programmers may be disappointed by BDS (or any other authoring program) as it assumes that each menu or button is defined separately. There are limited opportunities to use subroutine or method like constructs to address one menu that is modified programmatically. BDS expects most menus to exist by themselves. And many popup menus are almost clones of their main menu counterparts. The programmer must resist the urge to think “I define one menu with one picture and I will programmatically change that picture”. If there are 5 pictures, 5 menus are needed. Unfortunately.

One primitive manner of logging some data or steps is using the good old Java method of writing to the console through the method

System.out.println("string"). The console here is the Simulation log window. Not perfect, but at least something.

The project.bdmd file is in fact an XML file. If you open it in Notepad you can quickly edit the script parts that are all written between the tags <script> and </script> . Before you make any changes this way, make a copy of the original .bdmd file first as backup so you can use it if your modification fails to keep the .bdmd file a proper BDS project file. Below an excerpt of such project file where a menu has an Press Enter Action1 action written as Java.

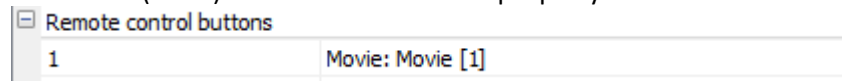
```
<AutoAction1 Type="TBDScript">
  <Script>      long time;
  long RunningTime=0; // needs to be i
  int playlistID;
  int nrBookmarks;
  float fractionComplete;
  boolean OK;
```

From action to script

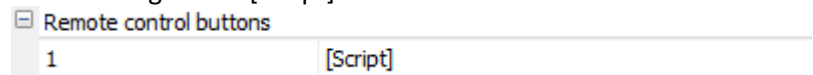
BDS has a feature that may look familiar if you ever tried to “record” steps taken in Microsoft Word or Excel. While you perform these steps, the recorder writes the corresponding (visual basic) code in a program called a “macro”. Opening this later gives you an idea of how to program the steps you took and what functions to use.

This “action Java code by example” is also present in BDS. It can show you the Java code that is generated when you specify actions to be performed. The code is revealed if you take the following steps:

1. First add a (multi)action or Switch to the property’s action line.

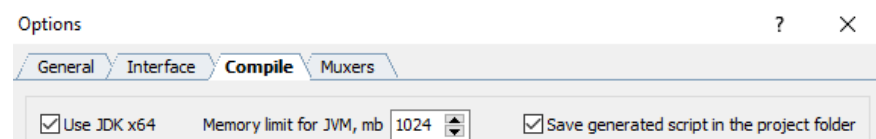


2. Reopen the action property but specify “Java”. The action you specified earlier is converted in (identical acting) Java code. This code is now shown in a Script window
3. Edit the code in the Script window and save it. The property action changes into [Script]



Unfortunately (but understandably), it does not work the other way around.

To use Java as “translation” of your action steps, you must check in Options > Compile tab the box “Save generated script in project folder”.



This will create a __SCRIPTS folder in your project containing some Java files that collect all these home-made scripts.

The following project shows how to remove all bookmarks from a movie (this is not available as action in BDS). Another project, specific to BDS MX is determining when to show picture-in-picture movies. This is shown in section “Project 4: the Java solution of to project 2” on page 396.

Create a timeline

It is possible to create some sort of progress bar as part of a movie popup menu to indicate how far we’ve come in playing the movie. A number of set top players offer this information themselves through an “Info” button. In this section we build it into the disc logic itself for a particular movie that displays the information if you press the “Up” button on the remote control.

The idea is that you know the total length (100% running time) of the progress bar for the complete movie. Using an animation clipping area you can cut off the 100% progress bar to less than 100% and only show the part visible in the clipping area.

To create such a running time progress bar you:

1. Determine current movie play time position
2. Compute the percentage played
3. Set clipping area accordingly, taking into account where the progress bar starts and how long 100% is

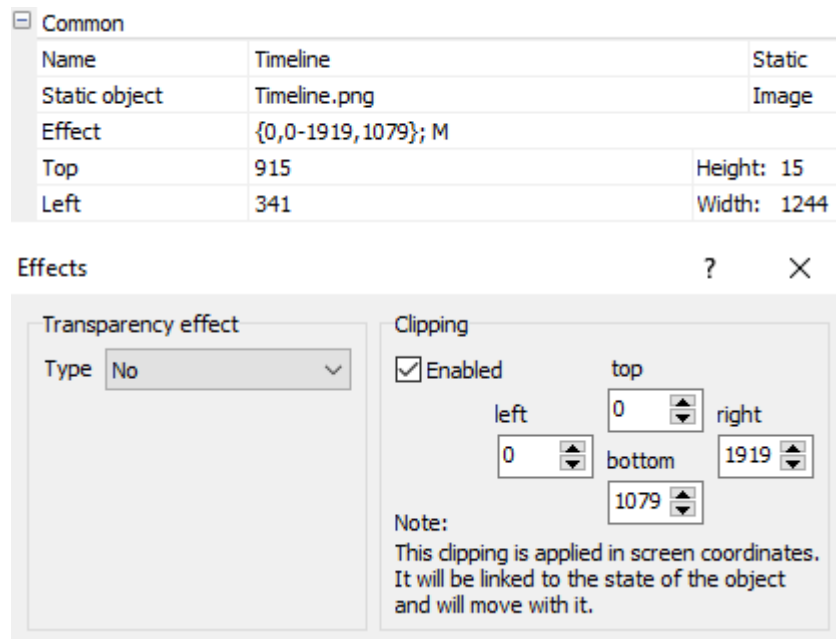
The online help has an example on how to create such a timeline for each movie individually by selecting a set of BDS menu options.

If you want a more generic approach with a single script to handle all movies (on the same or different discs), you need to do some programming. This programming needs to be executed at the start of the popup menu that represents the timeline but also every second of the movie running to update the playing time of the movie so far.

Suppose the movie is 137 minutes in length (can be obtained also at runtime by calling BDS function `manager.getDuration()`), the progress bar starts at position (left,top)=(341, 915) and has a width of 1244 pixels and a height of 15 pixels.

The popup menu with a progress bar is called “Timeline”. It has an action when it opens and an “Action Every Second” that both perform the same calculations: the progress of the movie.

The progress bar object on this menu is also named “Timeline” and an effect defined that enables clipping (required, as otherwise it is static and always visible in its entity). Initially the clipping area is the entire screen from (left,top) = (0,0) to (1919,1079). In addition, “Move Container” (“M”) is checked for effects, implying that whatever state the object has, it retains that state even after moving or clipping it.



For a menu to remain visible, there must be at least one button that the user can select. We don't want to show this button so no state has any visible image (in fact it is positioned at (left,top)=(0,0) with width and height set to 0), but it has a "Down" action defined that closes the popup menu. The button is named appropriately "0".

↓ Press Down	[close popup]
anim down	

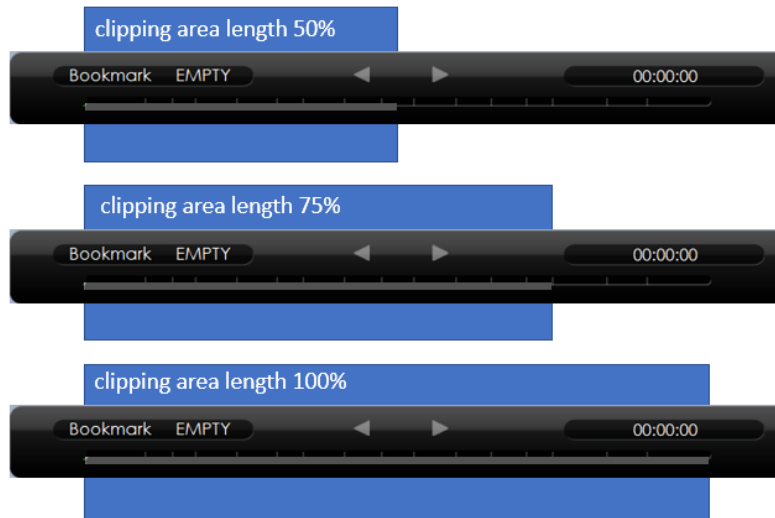
The example below shows such a progress bar on which bookmark positions can be placed later. This example is taken from the "Valerian" BDS project – a short movie can be seen on the BDS website.



The computations to be done are:

- 137 minutes running time = 8220 seconds
- 100 % (=8220 seconds) has a full progress bar of 1244 pixels
- Fraction p (=time in seconds t/8220 seconds) has length $L = t/8220 * 1244$
- Left end of the clipping area must be at 341 pixels
- Right end of clipping area must be at L + 341 pixels

The results of the calculations are shown below for running times of 50%, 75% of the movie run or the end of the movie (100%). The grey progress bar extends the entire 100% but is cut off by the clipping area so the viewer only sees part of the progress bar corresponding to the fraction of the movie already shown.



A project to create such a timeline is provided in Project 12: Add a popup Timeline on page 327.

Project 11: Remove all bookmarks

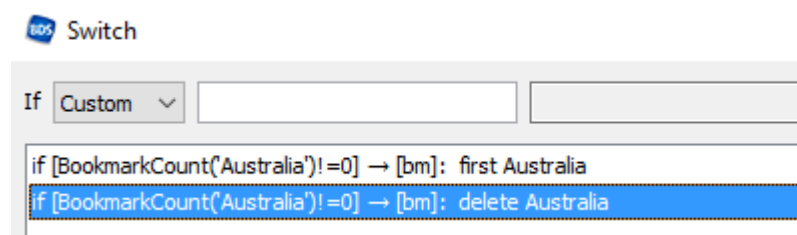
Project Goal

We want to delete all bookmarks of a movie called “Australia”, using the Java code generated by BDS. They have been set by the viewer while watching a movie (usually by pressing the red button whose action is set to be Bookmarks > Set bookmark).

This just inspects a single user-defined action and does not qualify as a full-fledged project. As such, there is no example project available. Simply follow the steps below in any project.

Project steps to take

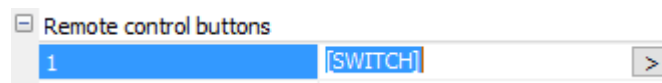
There is no such operation possible by menu selections of actions. What comes closest is the Switch that says “if there are bookmarks, go to the first one and delete it”. The Switch statement could look like this:



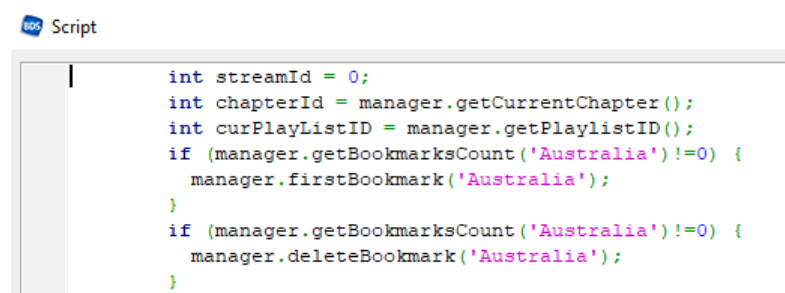
The Switch is unchecked for “Exclusive” as both (identical) conditions must be executed.

- The first one checks if there are bookmarks and if so, select the first one.
- The second one checks if there are bookmarks and if so, delete the selected one.

Save the Switch (as action (here: to remote-control button “1” when movie “Australia” plays but any available button would do). That takes care of the first bookmark – if it exists.



The Java code behind it is found if you re-open the [Switch] action by selecting “Script (java)” rather than “Switch”. That gives you the same switch action translated into Java code:



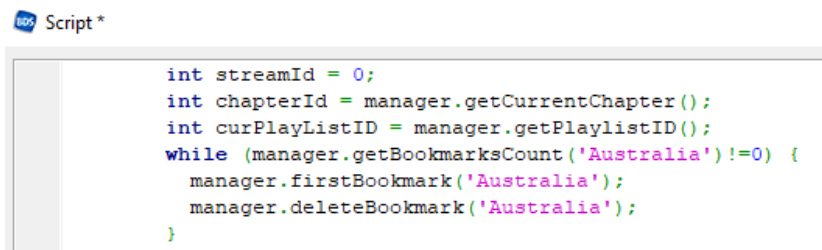
How do we modify this to take care of all bookmarks – if there is more than one? The Switch action doesn't know a conditional loop. But if you know a little bit about programming, almost all languages have something like WHILE <condition> DO ...actions... END WHILE or REPEAT ...actions... UNTIL <condition> . Or both.

Look it up in a Java manual (or in this case see section “Loop” on page 507) and find the construction we're looking for has the syntax

```
While (condition) { statements }
```

Look at the code and see that the first “if” can be changed into a “while” condition. If you type “while” you see that the window has some Java syntax knowledge as the word colours blue as “reserved word” for Java.

Both separate “if” statements of the Switch can be combined as statements (actions) of the while loop. It always sets the first bookmark and deletes it, until there is no first bookmark anymore.

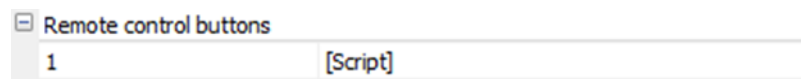


```
int streamId = 0;
int chapterId = manager.getCurrentChapter();
int curPlaylistID = manager.getPlaylistID();
while (manager.getBookmarksCount('Australia') != 0) {
    manager.firstBookmark('Australia');
    manager.deleteBookmark('Australia');
}
```

Closing the window replaces the [SWITCH] action by a [Script] action.

The generated integer variables streamID, chapterID and curPlaylistOD are not needed and could be deleted for our purposes. But they do no harm either if left there.

Note, that there is no way back once you converted actions into script (even if unmodified).



If you set some other button to create bookmarks you can see how pressing “1” deletes them all. It is customary to use the “red” button for this.

Creating bookmarks is simple. If you assign the “Red” button to this: its action is Bookmarks > Add bookmark.

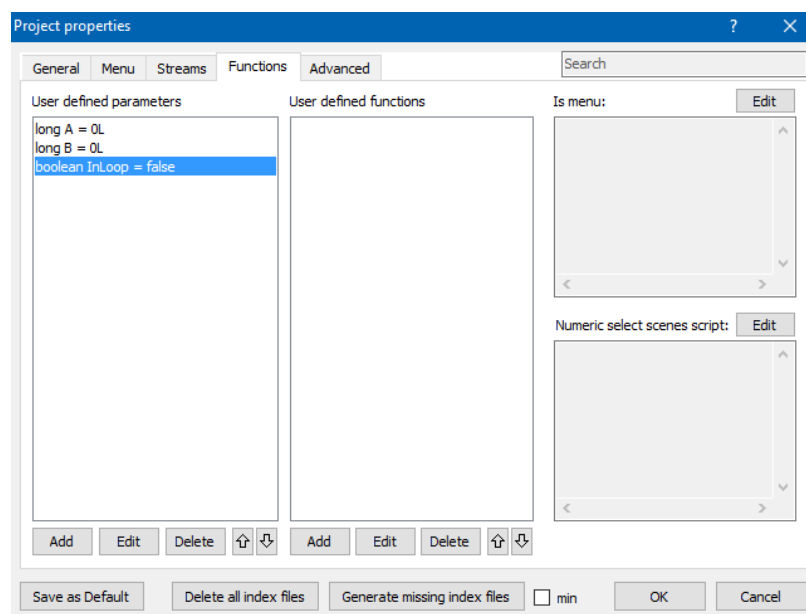
By using a call to manager.getPlaylistID() you may obtain the playlist ID of the current movie. Rather than explicitly stating ‘Australia’ (with single quotes) you might have specified the ID to make it more generic:

```
int playlistID = manager.getPlaylistID();
while (manager.getBookmarksCount(playlistID) != 0 ) {
    manager.firstBookmark(playlistID);
    manager.deleteBookmark(playlistID);
}
```


User defined variables and functions

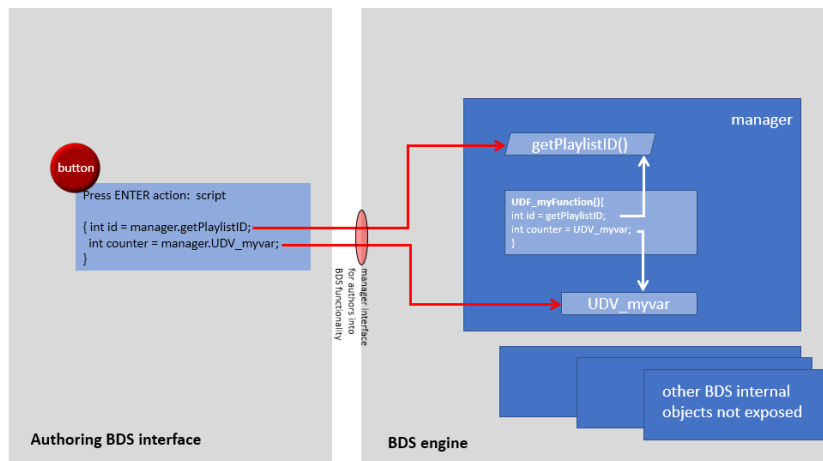
There are a number of GPR registers that you use to store integer values in, as we've seen in "Using Registers (simple programming)" on page 287. But at times you need other types of variables – those that can store boolean true/false statements, time values, strings.

For this purpose, the Project > Project Properties > Functions tab is meant. It allows you to create "user defined parameters" or variables, known as UDV's. The tab also allows for UDF's – User Defined Functions. As part of java, both variables and function names cannot contain spaces. Use underscore characters instead or use CamelCasing names that remove the need of spaces.



You should know that everything in Java is case sensitive. Therefore "myname" is a different variable from "myName" and both are different from "MyName".

Both UDFs and UDV's are publicly known everywhere in BDS. They are defined within one of the BDS owned objects of its engine. This engine makes BDS do what it does. Authors (users) can use some of this functionality in Java by specifying "manager." In front of the BDS exposed functions; such as "manager.getPlaylistID()" as well as access the UDV and UDF defined variables and functions. In programming terms, the names are part of the "manager" namespace.



When you use a UDV or UDF from an action script in the BDS menu or button properties, you need to address them via the prefix “manager.”. But if such a UDF function uses some other BDS functions, it needs not specify “manager.” As prefix since it resides within this manager object and its namespace. Simply addressing a function as “getPlaylistID()” suffices in that case.

As of BDS V4.1.0.1676 there is no need to remove the “manager” namespace prefix as BDS removes all these names internally in UDF functions.

Variables (UDVs)

You can add, delete or modify the definition of variables. Clicking on the “Add” button opens the variable definition window.

You can specify any name for the variable. To avoid clashing with names already used by Java or BDS itself, all names are internally prefixed by “UDV_” . A variable called “myName” therefore is internally known as “UDV_myName”.

The type of the variable is one of the Java known simple data types.

A variable should be given an initial value (if omitted, the value 0 or empty string ("" – two double quotes delimiting the string because

they are a String Java object that surrounds all values with double quotes) is assigned.

BDS UDV's are added to the namespace of the “manager” object. Therefore, a variable of name XX is prefixed with UDF_ and referred to from menu or button actions as manager.UDV_XX . For functions that reside within the manager object (such as User Defined Functions) you do not specify the namespace name, hence address the variable as UDV_XX.

Certain action properties can now be used to set or modify the values of the UDV's. Those values can be manually set, but can also be retrieved from data known to the system – such as movie name or current time position of a playing movie. Many of the functions that can be used are “exposed” to the programmer are described in the online help section “Available Java functions (methods)” or in section “Java functions in BDS” on page 509.

Some examples of using Java code as action can be found in

- Project 12: Add a popup Timeline menu on page 327
- Project 13: Popup Timeline with bookmarks on page 332
- Project 14: Playing an A-B loop on page 347.

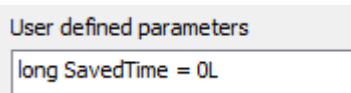
Example UDV: Restart a movie from a certain point

Without making it a project, the following illustrates how you can use a UDV to specify a time while playing a movie.

1. You define one remote control button to store the current time of the movie playing in a long UDV “SavedTime”.
2. You define a second button to jump back to this saved time and start playing from there.

To do this, take the following steps:

1. First define the UDV (project > project properties > Functions) as a long variable named SavedTime.



User defined parameters

long SavedTime = 0L

2. Next you need to use this variable and fill it with the current movie playing time using a single line of Java code. The variable is known as manager.UDV_SavedTime, the function that delivers the current playing time is a BDS function (see appendix section “Primary (main) Movie functions” on page 513).

```
// save current time  
manager.UDV_SavedTime = manager.getMediaTimeInSeconds();
```

This script is assigned to a remote-control button (say “red”) of a movie (or all movies)

3. Finally assign the following code to a second remote control button (say “blue”) of a movie (or all movies). This will restart

the movie at the point saved in step 2. Because the jumpTime function requires an integer value while the time is saved in a long variable, the use of a simple (int) type-cast modifies the long value into an integer (of course the value of the long variable must fit in the integer, i.e. must be smaller than 64,000 seconds or 1000 minutes – so no problem here!). The “S:Movie” is the generic way of stating “the current movie”.

```
// start movie from time saved in UDV variable
int intSavedTime;

{ intSavedTime = (int) manager.UDV_SavedTime;
  manager.jumpTime(intSavedTime);
  manager.activateSegment("S:Movie");
}
```


4. Compile the code and run the disc. Play a movie and see how “red” sets a replay point and “blue” activates this.

User Defined Functions (UDF)

UDF in general

A UDF function can be used in any property that allows an action – mostly in menus, buttons and movie placeholders. Be aware however that if you specify a UDF in the Start Action property of a movie, it is always executed, even if the movie is played from some chapter in the middle onwards.

A skeleton Java function is provided in the Script window that opens if you click on the “Add” button.

 Script

```
public void UDF_FunctionName_243340003 () {
}
```

The number following the FunctionName is generated at the time you create the function and ensures the function name is unique. You may however replace the “FunctionName_243340003” for something more descriptive to your liking. Leave the prefix UDF_ to ensure your function name does not collide with either Java or BDS internal code.

Each Java function has a declaration

```
public type UDF_name (arguments) { function statements }
```

The “public” statement is required as your UDF must be available throughout (Public) the BDS application. The type can be any of the common Java types (see online help “Script (java)”)

- void (nothing – makes it a subroutine)
- long or int – long or short integer is returned
- float or double – real number is returned
- string – a text string is returned

- boolean – a true/false value is returned

If the function is to return a value, the `return` statement is used. The use of `return` without a value is simply a quick exit from the method as any code following it is not executed.


When the UDF is used as an action of a button or menu, it cannot return any value and must be of type void. It can have arguments that need to be given a value when the UDF is called from elsewhere.

The UDF's are maintained within the "manager" BDS object and its namespace. Therefore, the UDF function code must not use the namespace prefix "manager." itself when using a BDS function, as it already resides in this manager object.

Therefore, an external Java action reference to `manager.getPlaylistID()` becomes `getPlaylistID()` when the script is turned into a UDF function.

When you refer to a menu or movie, use the names you gave them but replace any space by underscore and prefix with `MM_` (menu), `PM_` (popup menu) or `MV_` (movie).

Some more name conversions are listed in the online help topic "Script (java)".

To check for errors, you can compile the Java code after closing the script and click the  button. The log window and log file will contain and highlight any Java coding error when it reaches the "Compiling: Step 3".

```
Log
01:52:23 - [state] Compiling: step 3...
MediaManagement.java:163: illegal start of expression
    public int UDF_Increment(int oldValue) { oldValue += ; UDF_Increment := oldValue }
                                   ^
```

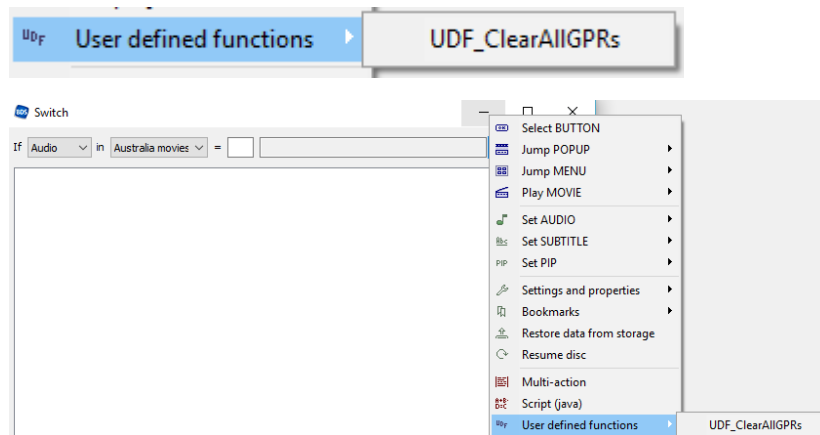
A simple Java script function is one that increments a value by one:

```
Script *
public int UDF_Increment(int oldValue)
{
    oldValue = oldValue + 1 ;
    return oldValue;
}
```

UDF as action

If a user defined function is selected as an action, it must be of type void. It can never function in conditional checks of Switch actions as it cannot be of type boolean that are used there. It can be an action to execute if a condition evaluates to true.

Note: Before BDS V4.1.0.1672 action scripts could not properly be stored in a UDF without generating errors. As of that release, you can. Provided that Java code written for actions of menus or buttons uses "manager.UDF_xxx" when calling a UDF function. The "manager" prefix specifies the namespace in which the UDF function is found. This prefix must be omitted when a UDF function calls another UDF function (as they both reside within the "manager" namespace already).



If the UDF is specified as a StartAction of a movie, it is always executed when the movie starts running – even if from some chapter in the middle.

PlaylistIDs

All movies have a unique sequence number assigned by BDS upon muxing the disc. The number is known as its playlist ID. The number may change in different muxings if you change the position of the movies relative to each other or insert or delete movies. The standard way of issuing numbers is first intro movies, then menu movies and finally actual movies.

When you're coding you can get the ID of the current movie by calling the `manager.getPlaylistID()` function.

But what if you need the number of a not-playing movie? BDS allows for this by specifying the movie object name in single quotes. Such as `thisID = 'Star Wars';`

Then at mux time the text 'Star Wars' is replaced by the playlist ID of the movie contained in the "Star Wars" object. This way you can initialize values of UDV variables if in an action shortly after the FirstPlay (like in opening a menu or End Action of an intro movie) does the assignment:

```
manager.UDV_StarWarsID = 'Star Wars';
```

If you look at several of the descriptions of the BDS supplied functions, the use of the playlist IDs becomes noticeable:

```
int manager.getAudioID(int playlistId)
```

```
int manager.getAudioID(#string movieName)
```

The first one can be called as either

- `manager.getAudioID(3);`
- `manager.getAudioID('Star Wars');`

Because during compile/mux time the single quoted name is replaced by the (type int) playlist number (here assumed to be 3).

The second one expects a Java string (#string) , hence double quotes:

- `manager.getAudioID ("Star Wars")`

Movie with different angles or in colour/black & white

One use of Java programming in BDS is to support the viewing of the same movie from different angles, if the movie has different angles.. Or if the same movie is available in b/w and colour.

Each angle of the movie is a complete movie in its own, but they show different viewpoints of the same action at the same time. Or the same film is shown in original black and white as well as in a colourized version.

Basically this means you need a button on the remote control (e.g. the RED button) that when pressed, switches to the same movie but different angle and at the same playing time. This is easily achieved with a User Defined Function invoked by the remote control button. The function determines the current movie's playlist ID and current timing inside the movie. Then it starts from the same time, but different playlistID.

Red	<code>manager.UDF_NextAngle();</code>
-----	---------------------------------------

The example below shows a UDF called "UDF_NextAngle". It is invoked by the remote control RED button on four movies, called "main", "angle1", "angle2" and "angle3". Each of these movies transfers playback to the next playlist ID movie (and "angle3" returns to "main" to create a loop). Remember that a single quoted movie name is translated by BDS into its playlist number.

```
public void UDF_NextAngle() {  
  
    // switches movies to show different angles.  
    // The next angle continues at the time the previous movie was interrupted.  
  
    // This UDF must be assigned to a remote control or pop up button action  
    // on all the movie angle variations to be able to switch between them  
  
    int PlaylistID;  
    long Playtime;    // time is returned in nanoseconds  
  
    // Step 1. find current movie ID  
    PlaylistID = getPlaylistID();  
  
    // Step 2. Increment the playlist ID or return to main movie  
    if (PlaylistID == 'main') {PlaylistID = 'angle1';}  
    else if (PlaylistID == 'angle1') {PlaylistID = 'angle2';}  
    else if (PlaylistID == 'angle2') {PlaylistID = 'angle3';}  
    else if (PlaylistID == 'angle3') {PlaylistID = 'main';}  
    else {PlaylistID = 'main';}  
  
    // Step 3: find current playing time  
    Playtime = getMediaTime();  
  
    // Step 4: switch movie to new angle version  
    playVideoFrom(PlaylistID, Playtime);  
  
}
```

Step 2 can also be replaced by the switch statement where you should not forget to add the break statement as last part of each case choice.

```
// Step 2. Increment the PlaylistID or return to main movie
switch (PlaylistID){
case 'main' : {PlaylistID = 'angle1'; break;}
case 'angle1' : {PlaylistID = 'angle2'; break;}
case 'angle2' : {PlaylistID = 'angle3'; break;}
case 'angle3' : {PlaylistID = 'main'; break; }
default : {PlaylistID = 'main'; break; }
}
```

The BDS supplied functions are printed in **bold**. They are listed in Appendix F: Java programming on page 504.

Switching between movie angles may also switch audio and subtitle tracks (which default to 1 or what is specified in “Streams” in the Project Properties). The use of movie groups does not seem to help (works if movies are selected through a button with “Play movie” event defined). If you programmatically switch movies, you need to save the current setting for audio and subtitle track and apply these again for the newly started movie using one of the selectAudioSubtitleStream functions. This is indicated in the code snippet below where one movie can be switched for its colourized and original black and white version.

```
public void UDF_SwitchBWorColour() {

// switches same movie between black/white or colourized

int PlaylistID;
int SubtitleID;
int AudioID;

long Playtime;    // time is returned in nanoseconds

// Step 1. find current movie ID and set language and
subtitle

PlaylistID = getPlaylistID();
SubtitleID = getCurrentSubID();
AudioID = getCurrentAudioID();

// Step 2. Increment the playlist ID or return to main
movie

if (PlaylistID == 'Wonderful Life bw')
    {PlaylistID = 'Wonderful Life colour';}
else {PlaylistID = 'Wonderful Life bw';}

// Step 3: find current playing time
Playtime = getMediaTime();

// Step 4: switch movie to new version
playVideoFrom(PlaylistID, Playtime);

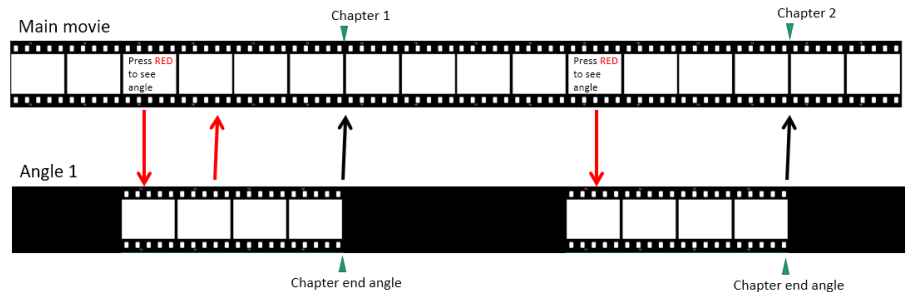
// Step 5: audio and subtitle streams the same as before
selectAudioSubtitleStream(PlaylistID, AudioID, SubtitleID,
true);

}
```


Short angle scenes

In case you only have certain moments with different angles instead of the whole movie, the described method only works for different angle versions running for the full movie length.

In this case you need to video-edit an angle movie as a full length black movie. The angled parts of the movie must be inserted at the right timings to correspond with the same scene in the main movie.



You use the red coloured remote control button to move through the various angles and main movie the same way as before: this button is programmed to invoke the Java routine described earlier.

It is common to add some indication in the main movie (hard coded into the movie or as subtitle stream) to indicate that there is an other angle shoot available if the red button is pressed. Doing so, brings you into the other angle version. At any time you press the red button you alternate through the various angles.

When you want to see the complete angle shoot and return to the movie at the end of the angle shoot, you need to insert a chapter at the end of the angle movie section and execute a chapter action. Then there are two options on what to do when you invoke that action.

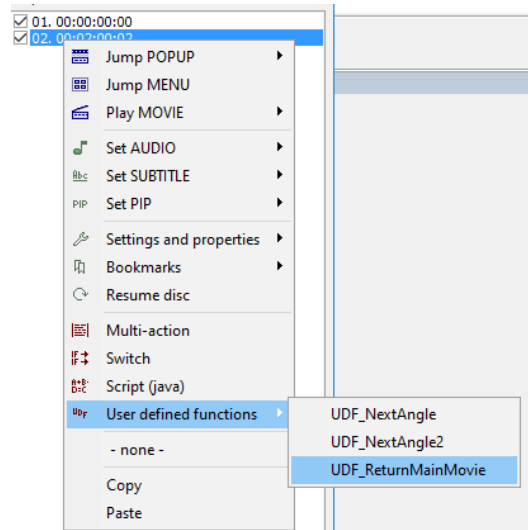
1. Invoke a routine to return to the main movie at the same moment in time

At the end of the angle scene chapter an instruction to invoke a UDF routine is entered using the “Direct Editing” feature of a chapter mark. This is described in Method 4: movie chapter action on page 304.

The UDF routine is even simpler than the earlier one described.

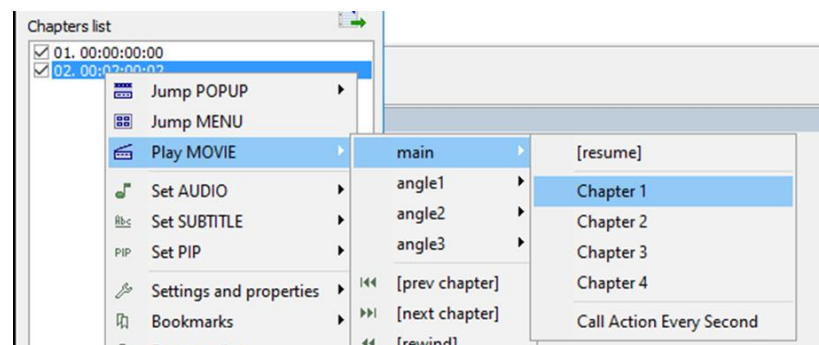
```
public void UDF_ReturnMainMovie() {  
    // returns from an angle movie bit to the main movie at the same time  
  
    long Playtime;    // in nanoseconds  
  
    //find current playing time  
    Playtime = getMediaTime();  
  
    // switch movie to new angle  
    playVideoFrom('main', Playtime); // continue main from same time  
}
```

The Direct Editing instruction is added using “User Defined Functions”



2. Jump to a chapter in the main movie

This method requires you define a chapter in the main movie at the spot where the angle movie view returns to the main movie. Then the instruction at the chapter mark at the end of the angle movie is simple: jump to the main movie starting at the main chapter where the movie continues.



This instruction is added using the “Direct Editing” feature where you select the final chapter mark of the other angle scene and add the instruction to move to another movie at a specific chapter mark. This is described in Method 4: movie chapter action on page 304.)” .

Project 12: Add a popup Timeline menu

Project Goal

This project is creating a simple popup menu that shows a timeline of the movie currently playing. It does nothing more.

You can show this timeline popup by remote-control button buttons Up Arrow (show it) and Down Arrow (remove it) buttons.



All this timeline menu does is show the progress of the movie – both in actual timing (hh:mm:ss format) shown at the right side of the menu and as fraction of a (grey) progress bar that runs from 0% to 100%.

You can provide all info for the timeline explicitly per movie or create a single Java module that asks for the current playing movie (identified by its playlistID) and then calculate all values for that movie. It is this approach we will follow in this project.

Remember again, that the running movie is leading in what menus can show. The timeline popup is defined in HD resolution. If the movie is also in HD, it works fine. If it is less than HD, you won't see part of the popup or none of the popup. In that case you need to adjust the popup menu to fit the movie resolution or (preferred?) transcode the movie to HD resolution.

Project steps to take

Things to consider

Because some programming is involved, you need to realise that

- the playlistID of a movie is determined by its position in the movies list and the number of menu movies and intro movies. Adding or moving any of these items may change the playlistID of each movie.
- The progress bar (called Timeline) is a static long strip that indicates a movie at 100% of its playing time. During playback it should be cut to anything between 0% and 100% complete. This is achieved by making the Timeline object enabled for (animation) clipping. And then clip it according to the time the movie has played so far.

Java functions

The code is not very complicated, but a few functions are used that are exposed by BDS to provide information to the script or perform some action on a BDS object. You may need to read up on these (see Java functions in BDS on page 509), but the important ones are:

- `manager.getDuration()` – returns the total playing time of the playing (current) movie in nanoseconds (mostly used in conjunction with `manager.time2text()` function that converts the time into a hh:mm:ss text format.)
- `manager.getDurationInSeconds()` – returns the total playing time of the playing (current) movie in seconds – often used to determine the fraction of the timeline to show as fraction of seconds compared to the total seconds a movie runs.
- `manager.getPlaylistID` – returns the sequence number by which BDS plays a movie. You need to know this number in order to “do something” on this movie
- `manager.activateSegment` – starts a movie or opens a menu
- `manager.activateButtonEx` – selects and activates (or not – if “false” is set) the specified button on a menu. There is always one button that is the “selected” one on a menu.
- `manager.getMediaTime` – returns the played time of the movie so far
- `manager.setText` – replaces the text in a textbox menu object (which is therefore updatable instead of a static fixed text)
- `manager.setClipping` – defines the animation clipping area. It is used to have 100% long bar as static (but clipping enabled) object in a menu to clip it to a smaller percentage and therefore shorter length

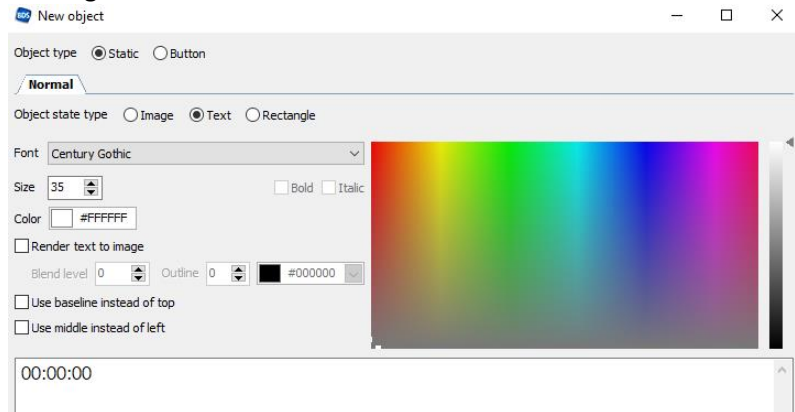
Ready to Run

1. Copy the project folder `\Projects\TimelineBR` of the BDS kit.zip to your local BDS projects (e.g. `J:\BDS Projects`) as a new folder in the same BDS projects folder.
2. Copy the movie files `Australia.*` and `Coasts.*` from the `\Sources\Movies\Demuxed` folder and copy them into the `\films` folder of your `\TimelineBR` project. A single video .264 and audio *_live.ac3 suffices to have movies
3. Copy the movie files “menu.*” from the BDS kit.zip `\Sources\Movies\Demuxed` folder into `\files` of your own `\TimelineBR` project
4. Mux the project
5. Experiment!

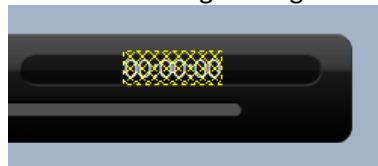
Build your own

1. Create a new project `\TimelineBR` folder in your local BDS projects folder (e.g. `J:\BDS Projects`)
2. Copy the `\MenuBR` project into the new `\Timeline` project. Of the `\films\Australia.*` and `Coasts.*` files essentially you only need their *_live.ac3 audio track and no subtitles but you can leave it the way `\MenuBR` was set up
3. Open the project.bdmd file of this project

4. Copy the \Sources\Projects\TimelineBR project of the BDS Kit.zip file, the Photoshop file “\original sources\main menu.psd” and “Timeline popup.psd” files into the \original sources folder of the new project.
5. Import the “main menu.psd” file into the project in the “menus” section of the Project Tree.
6. Add two buttons to link with the menu text. Their “Press ENTER” actions should start playing the movie, their “Press Up” and “Press Down” actions should allow to navigate between the buttons. Button images can be found in the \buttons folder of the project.
7. Import the Photoshop file “\original sources\Timeline popup.psd” into the Popup menus section of the Project Tree.
8. Create a folder \fonts in the project map. Copy the file “Gothic.ttf” from the BDS kit \Sources\BDS Fonts folder into this \fonts folder. This add the “Century Gothic” font to your project, as needed in the next step.
9. From within BDS, add an extra static text object to the “timeline popup” popup menu. Call it “Timer”. Use the \fonts\Century Gothic font for the text in this text object. This font file is part of the copied project folder set. Use white as the font colour by sliding the colour slider at the right edge to the (white) top. Set the font size to 35 pt. Initialize its text value as “00:00:00”. Because this value must be updated through script code, ensure that the “Render text to image” checkbox is clear.



10. Usually new objects are created in the top left corner in the Designer window. Reposition the text object to around (left,top)=(1426,841) with (height,width)=(42,132). It should fit at the right side end of the popup menu in the rounded cornered rectangle designed for it.



11. The movie “Australia” needs a few actions defined:
 - a. Popup button: open the “timeline popup” popup menu, perform its Action 1 on opening.
 - b. Up arrow: same as the popup button

- c. Down arrow: jump to the main menu (you can have a button for this on the popup menu, but here we don't).
- 12. The movie "Coasts" has the same actions defined as the "Australia" movie does (you may want to copy/paste these actions using the Action Matrix view for both movies).
- 13. Add a new button object to the "Timeline Popup" popup menu (just click on the "New object" button, do not expand the dropdown menu for choices):
 - a. Call it "ClosePopupButton"
 - b. Do not provide any image to any of its states
 - c. Define only a single action for its "Press Down" action: select [close popup] for this.

This makes it an "invisible" button. It is always selected (being the only one) and is required for the popup menu to stay alive (no buttons means it must pass control to another menu with buttons). It does respond to the "Down Arrow" button on the remote-control: it closes the popup menu (as does the "popup" button too by default).
- 14. The popup menu must have information on what movie is playing and what its total running time is. There are specific BDS functions for these needs (as of V4.1.0.1676)
- 15. All that remains is to ensure the popup menu is updated constantly. This is achieved with some Java code that performs the following steps:
 - a. determine what movie is playing (get its playlistID)
 - b. retrieve the current playing time for the movie and display it in the "Timer" text object
 - c. retrieve the running time of the currently playing movie
 - d. Compute the fraction the playing time is compared to the running time (100%).
 - e. Determine what fraction of the length of the progress bar (fraction x 1244 pixels) should be made visible.
 - f. Set the clipping area for the Timeline object to this value
- 16. All this is achieved by the Java code below. Rather than retyping it, you can copy it by opening BDS Kit folder \Projects\TimelineBR\original sources\Timeline-everysecond.txt and copy/paste it into the empty script window for the "Timeline Popup" menu for two of its actions:
 - a. Enter/Action 1
 - b. Action Every Second

```
int playlistID;
long RunningTime; // must always have a value
long time;

// User defined values
int BarX = 335; // offset of progress bar from left
int BarLength = 1244; // full length of progress bar

// determine running time of current movie
playlistID = manager.getPlaylistID();
RunningTime = manager.getDurationInSeconds();
```

```
// perform the rest only if movie is recognized
// get current playing time movie in nano-seconds and
// display that time as hh:mm:ss
time = manager.getMediaTime();
manager.setText("F:PM_Timeline_Popup.Timer",
               manager.time2str(time));

// Set time progress bar through clipping area.
time = manager.getMediaTimeInSeconds();
time = time * BarLength / RunningTime + BarX;

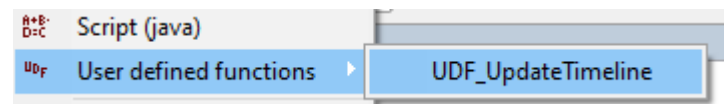
manager.setClipping("F:PM_Timeline_Popup.Timeline",
                   0, 0, (int)time, 1079);
}
```

17. Compile/mux the project and run the resulting disc.

Further enhancements

The popup menu can be made nicer if you let it slide in from the bottom and slide out to the bottom through some opening and closing animation.

The code can be moved from specific menus and buttons to one central location, as a User Defined Function (UDF) through Project > Project Properties > Functions tab. Copy/paste the code into the UDF function (and call that UDF_UpdateTimeline). Then all menu properties that used to have their on (identical) scripts, now simply refer to it: manager.UDF_UpdateTimeLine as one of the options of the action dropdown menu.



A project using this is stored as project – using UDF.bdmd in the \TimelineBR project folder.

Also when a movie plays, the “Pause” action of the remote control could automatically show the Timeline popup menu. When the “Play” button is pressed, it may be closed. The closing should only occur (Switch actions) if the timeline popup was not already visible before the movie was paused.

Project 13: Popup Timeline with bookmarks

Project Goal

This project continues on the result of the previous project by adding bookmarks to the timeline. Although BDS sets no limit to the number of bookmarks, this project has to, as each bookmark indicator is a small picture object that is positioned on the time line. BDS cannot create indicators on the run – they must all be in existence already, although those not in use can be hidden through transparency or positioning them off screen or behind some small opaque square menu object.

The project, using two short movies, allows for 4 bookmarks. They can be added while the movie runs by pressing the Green button on the remote-control. Addition is confirmed by a short display of a message. If you try to add a fifth bookmark another message will tell you that a maximum of four can be added.



The bookmarks are shown on the Timeline popup menu as little green bars. The Timeline menu is a popup menu, made visible by pressing the Up arrow button while the movie runs. The current bookmark is marked with a little green triangle on top. The timeline also shows the time value of the selected bookmark (at the left side of the menu) and the progress of the movie (at the right).

Using the “previous” and “next” arrow buttons on the remote control you can navigate between bookmarks, making another one current. Pressing “play” starts the movie from that bookmark onwards.

The blue button on the remote control removes all bookmarks.

While the Timeline popup menu is shown, a bookmark can be added by using the same green button on the remote control. But because you can navigate to a bookmark, the red button allows to delete that bookmark. The same informative message of a bookmark added or deleted is shown for a short period.

Project Steps (UDF)

Things to consider

To create bookmarks and show them on a timeline has a number of “problems” that must be faced:

- Only one menu can be shown at any one time. Showing menus that look like a small informative message such as “Bookmark Added” is simple to show while the movie runs and no

timeline menu is visible. But if the timeline menu is shown, showing the “Bookmark Added” message means another menu and this will remove the timeline menu. You might consider cloning the timeline as set of pictures to become part of the “Bookmark Added” menu. This is not possible because the timeline menu has flexible positions of the bookmark bars and the movie progress.

The way this solution implements it is using GPR 5 as a “comeback flag”. Just before the “Bookmark Added” message is shown, the register is set to 1. When the “Bookmark Added” menu shows, the timeline disappears. But when the “Bookmark Added” menu closes, it checks if GPR 5 has value 1 and if so, it resets it to 0 and reopens the timeline menu. If GPR 5 was 0, the timeline menu did not trigger the add operation and therefore need not to be (re)opened.

- All the actions applied to the “Australia” movie also apply to the “Coasts” movie. This means duplicating all actions and adjusting them to use the proper name of the movie. Rather than doing this (imagine you have 5 or 8 movies instead of 2) we decided to write all code only once and either copy/paste it between the two movies or store it in one place as User Defined Function and have all movies call that function.

- The calculation and positioning of the bookmarks and progress of the movie itself are difficult to do using the BDS menu options.

We decided to write most of the actions as Java code. This is a lot more flexible and easier to maintain

- The “Bookmark Added” menus are menus without buttons as they work as a short message before disappearing. In BDS however each menu that shows itself for some period, must have a button. Therefore, these menus have an invisible button with no actions. An Autoclose property ensures the menu closes after a set period (of 2 seconds)
- The timeline is a popup menu. But not a real popup menu in the sense that the viewer can select something. Therefore, the timeline is shown by pressing the “Up Arrow” and disappears by pressing the “Down Arrow”. This leaves the “Popup” button available for a real popup menu (not implemented in this project). Such a popup menu would have an option to go to the main menu. By lack of a true popup menu in this project, we used the “Down Arrow” in both movies to have the action “Jump to main menu”.

Java functions

This project is mostly written using Java instead of BDS button menu choices because the latter cannot do what we want it to do.

The code is not very complicated, but a few functions are used that are exposed by BDS to provide information to the script or perform some action on a BDS object. You may need to read up on these (see Java functions in BDS on page 509), but the important ones are:

- `manager.getPlaylistID` – returns the sequence number by which BDS plays a movie. You need to know this number in order to “do something” on this movie
- `manager.getDuration()` – returns the length of the movie in nanoseconds (often used in conjunction with `manager.time2text()` that converts the time into a text value formatted hh:mm:ss)
- `manager.getDurationInSeconds` – returns the length of the movie in seconds (used to determine the fraction of the timeline to expose in relation to total duration).
- `manager.activateSegment` – starts a movie or opens a menu
- `manager.activateButtonEx` – selects and activates (or not – if “false” is set) the specified button on a menu. There is always one button that is the “selected” one on a menu.
- `manager.moveToXY` – moves an object to a specific (absolute) location on screen
- `manager.getMediaTime` – returns the played time of the movie so far
- `manager.setText` – replaces the text in a textbox menu object (which is therefore updatable instead of a static fixed text)
- `manager.setClipping` – defines the animation clipping area. It is used to have 100% long bar as static (but clipping enabled) object in a menu to clip it to a smaller percentage and therefore shorter length
- `manager.getBookmarksCount` – returns the number of bookmarks set in the movie
- `manager.getBookmarkTime` – returns the timing that belongs to the current bookmark
- `manager.addBookmark` – inserts a new bookmark at the current timing of the playing movie
- `manager.deleteBookmark` – deletes the current bookmark
- `manager.firstBookmark` – makes the first bookmark current
- `manager.nextBookmark` – makes the next bookmark the current bookmark
- `manager.prevBookmark` – makes the previous bookmark current
- `manager.jumpBookmark` – starts playing the movie from the current bookmark forward
- `manager.getGRP` – returns the value stored in a GPR register
- `manager.setGPR` – sets a value in a GPR register

With these insights given, let’s build the timeline with bookmarks!

Ready to Run

1. Copy the folder tree from BDS Kits zip file \Projects\Timeline BookmarksBR to the local BDS projects tree (e.g. J:\BDS Projects) so a new project “Timeline BookmarksBR” is created locally.
2. Copy the files Australia.* and Coasts.* and menu.* from \Sources\movies\Demuxed into the local project’s \films folder.

3. Open the project.pdmd file
4. Mux the project
5. Experiment!

Build your own

This project builds on the previous one, TimelineBR. Because for some actions Java is used which is case-sensitive, ensure that in adding and naming new objects or menus:

1. The name is spelled exactly as given:
2. The name uses the same upper/lower case lettering

Only this way the programming code addresses the right and existing objects. If you divert from the suggested names, you need to change the Java named actions accordingly.

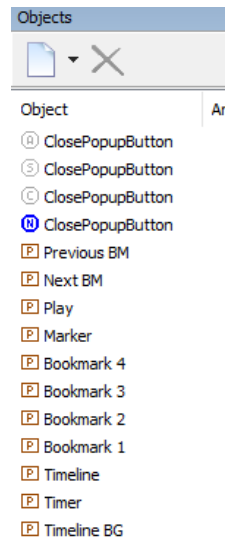
Get the “raw materials”

1. Copy the \TimelineBR project in your BDS projects folder and name it \Timeline BookmarksBR
2. Copy the files from the BDS kit “\Projects\Timeline BookmarksBR\original sources” folder to the local project “\original sources” folder. It contains the images of the timeline popup menu and some text files with Java code. The images are an extension to the ones used for just the timeline. By copying the files, some will overwrite the original Photoshop menu file used in the TimelineBR project.
3. Create a popup menu by importing the “timeline popup.psd” file. BDS will recognize the menu already exists (from the TimelineBR project). Confirm you want to keep the old one and update it with new objects. This will result in a new popup menu with additional buttons, indicators and a bookmark selection marker.

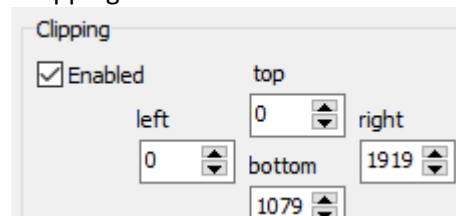


- The object “Timeline” is the progress bar that is clipped to show it grow from 0% to 100% playtime of the movie’s running time.
- The object “Timer” is a text object that is updated to show the movie progress time
- The object “Marker” is a small inverted triangle that is positioned on top of currently selected bookmark.
- The four bookmarks are named “Bookmark 1” to “Bookmark 4” and are small green bars that will be positioned on the timeline at the set timings.

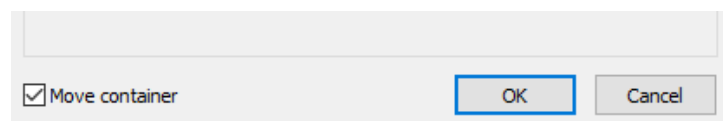
The import may have arranged the objects in the menu in an unwanted order. Reorganize them so they do not block each other from view. The resulting order should look like the order below.



4. Ensure that the object “Timeline” has been enabled for clipping (to “cut” the long Timeline to the portion of the film that has run). Select the object, right click and select the “Change effects” > check the “Enable” checkbox in the “Clipping” section.



5. Also ensure that the objects “Marker”, “Bookmark 1” to “Bookmark 4” have their effects set to “Move container” (right click and select “Change Effects” from the dropdown menu and check the “Move container” box.



This “container” property ensures that all effects applied to the object remain if its position is affected by a “moveToXY” operation in Java – and that’s what will happen with the indicators to position them properly on the timeline.

Add informational message menus

The idea is to have the red/green/blue buttons perform a function to delete/add/delete all bookmarks. This functionality must exist while any of the movies run. That means that for each movie, an action must be defined to enable the action.

6. Start with creating the three informational messages (that are BDS popup menus). Import from the \original sources folder the following popup menus (use “Add popup menu” > Import from Photoshop):
 - Delete Bookmark popup.psd

- Add Bookmark popup.psd
- Too many Bookmark popup.psd

This results in 3 popup menus of the same name.

7. To each of the three menus, add a button (Object window, click “Add Object” button). Call it “Button”.
8. Set the Autoclose time for each of these three menus to 2 seconds
9. Define two actions for the “Button” of each of these three menus. The execute a small script that can be copied from \original sources\MessageMenu-Close.txt.
 - The “Press Down” action
 - The “Press ENTER” action
10. Define two actions for each menu itself that do the same as the “Button” actions. Therefore, the code can also be copied from \original sources\MessageMenu-Close.txt:
 - The “Autoclose action”
 - The “Popup menu” action

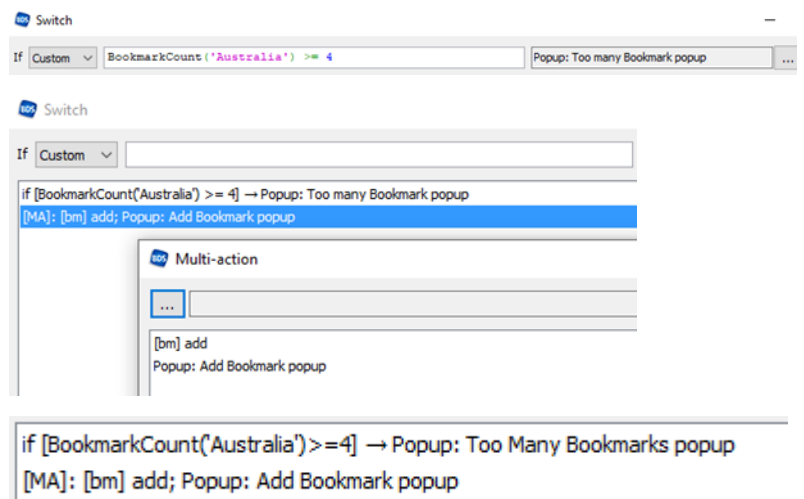
Both buttons allow the viewer to remove the messages in less than the 2 seconds the autoclose time has been set to.

Modify the movie properties – step 1: add the green/blue buttons

11. Select the properties of the “Australia” movie.
12. Add an action to the “Green” button: delete current bookmark and display a confirmation message.

The action associated with the Green button is a (exclusive) Switch action with two steps:

- If there are already 4 bookmarks, display the “Too many bookmarks” message menu
- If fewer than 4 bookmarks, perform a multi-action that adds a bookmark and displays the “Bookmark Added” message menu



13. For the “Blue” button we need to use a small script that deletes all bookmarks. The script can be copied/pasted from

\Original sources\timeline-blue.txt. It contains little code – just a repeated `manager.deleteBookmark()` call until there is no bookmark left.

```
Int playlistID;

// check the movie that's playing
playlistID = manager.getPlaylistID();

// remove all its bookmarks
while (manager.getBookmarksCount(playlistID) != 0) {
    manager.firstBookmark(playlistID);
    manager.deleteBookmark(playlistID);
}
```

Modify the movie properties – step 2: open Timeline popup or main menu

14. Assign an action to the “Up Arrow” to display the Timeline popup menu.
15. Because there is no real popup menu in the sense that it has menu items the viewer can select from, we use the “Down Arrow” to open up the main menu again.

Up	Popup: Timeline popup [TimelineButton] [anm/act 1]
Down	Menu: main menu [australia]

16. If you want to, you can assign the same action of opening the Timeline popup to the “Popup menu” action.

Repeat for the “Coasts” movie

17. Repeat all these steps taken for the “Australia” movie also for the movie “Coasts” for its green and blue button. Note that in the Switch statement you need to specify “Coasts” instead of “Australia”. Using the “Action Matrix” is a quick way to copy identical actions between movies.

Modify the Timeline popup – step 1: updating the timeline

Now that the movies have the required functionality, we want to add the same functionality when the Timeline popup is on display. A little difficulty: the informational messages of added/deleted bookmarks are menus and would remove the timeline. To avoid this, we must use some Java code instead of a simple Switch action. This code sets a “comeback” flag in the form of a value in GPR 5 and the message menus modify their closing actions by checking on this GPR 5 and either just close or reopen the Timeline popup menu again.

18. When the Timeline opens, it needs to set the progress bar correctly (same way as it did in the TimelineBR project). This requires a script to run on its Action 1 opening. Copy/paste the code from \original sources\timeline-open.txt
19. The timeline must be updated regularly, so another script (completely identical to the opening script) is needed for the “Action Every Second” action of the menu. Copy/paste the code from \original sources\timeline-everysecond.txt

20. If you want, you can assign the “Popup menu” action to be a close of the current Timeline popup menu.

[Modify the Timeline popup – step 2: add the current bookmark timing](#)

21. Add a text object named “Bookmark Time” the same way as the text object “Timer” has been made. This is done quickest by right clicking on “Timer” and select “Copy to” > Timeline popup. The resulting new object is called “Timer 1”. Rename it to “Bookmark Time” and position it at the left end of the Timeline popup and order it to be above or below the “Timer” object. Position it near the left edge of the rounded rectangle reserved for it (the code will write “BOOKMARK EMPTY” or “BOOKMARK hh:mm:ss” in this bookmark time indicator).



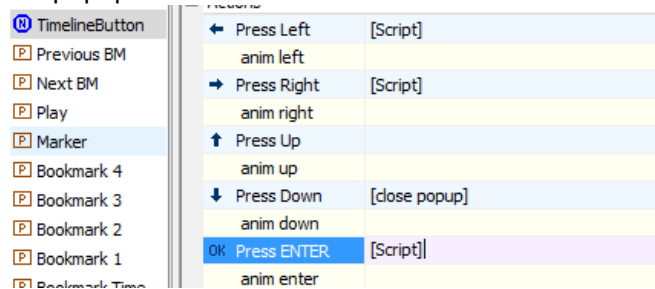
22. Make sure its object state has not set “Render to image” because Java code will continuously modify its text content and therefore this is not a still image.

[Modify the Timeline popup – step 3: add red/green/blue and TimelineButton buttons](#)

23. The action for its Green button can also no longer be a simple Switch statement, but needs to be some code because when the “Bookmark Added” menu shows, it needs to reopen the Timeline afterwards. Hence the GPR 5 must be set. Copy/paste the code from \original sources\timeline-green.txt
24. The action for the Blue button remains unaltered. It also runs a script. Copy/paste the code from \original sources\timeline-blue.txt
25. The Timeline popup menu has a single button. Call it “TimelineButton” (or rename it from its previous “ClosePopupButton” name given in TimelineBR project if it already exists). This button now must perform several functions for buttons on the remote control. The button itself is invisible.
- The action for the “Press Left” button is moving from the current bookmark to the previous bookmark that then becomes current. This is achieved by a small script. Copy/paste the code from \original sources\timelinebutton-previous.txt
 - The action for “Press Right” button is moving to the next bookmark. This is achieved by a small script. Copy/paste the code from \original sources\timelinebutton-next.txt
 - The action for “OK” (“Press ENTER”) is starting the movie to run from the current bookmark forwards.

Copy/paste the code from \original sources\timelinebutton-OK.txt

- The action with “Press Down” is simply closing the popup menu.



26. Unique for the Timeline popup is the ability to also delete bookmarks. Because they are shown and can be selected, the selected one can also be deleted. This is what the Red button does. Its action is also a small script. Copy/paste the code from \original sources\timeline-red.txt

Build!

When all these modifications have been made, mux the project and build the disc.

Javacode used explained

Various actions used Java code to get done what needed to be done. Below you find all the scripts explained. Their sources are in the “\original sources” folder. For the programmers under us much will hopefully be self-explanatory. Note that if you transfer these Java codes to a User Defined Function (UDF) you must remove all the “manager.” Prefixes as the UDF already operates from within the BDS manager object.

Timeline-open.txt and Timeline-everysecond.txt

To keep the script simple to maintain and copy for other uses, the position and length of the progress bar Timeline has been specified upfront as a variable value. This variable is then used everywhere in the code. That should relieve you from changing a value everywhere instead of just once.

The code first must figure out which movie is currently playing by checking its playlistID and comparing that with either “Australia” or “Coasts”.

```
// On open menu = identical to ActionEverySecond

// THE FOLLOWING SECTION SHOULD BE MODIFIED WHEN THE TIMELINE POPUP MENU
// CHANGES IN LAYOUT

// ++++ START USER MODIFIABLE SECTION ++++

// specify start of Timeline progress bar BarX and its length BarLength in
// pixels from left margin

int BarX = 335;
int BarLength = 1244;

// position settings vertically from top of screen for bookmark
// indicators and marker
```



```

int MarkerY = 889;
int IndicatorY = 908;
int MarkerWidth = 12; // marker point is inverted triangle point in middle

// +++ END USER MODIFIABLE SECTION +++

int playlistID;
int nrBookmarks;
long RunningTime;
long time;

// determine running time of current movie
playlistID = manager.getPlaylistID();

// determine total running time of the movie
RunningTime = manager.getDurationInSeconds();

// get current playing time movie in nano-seconds
// and display that time as hh:mm:ss
time = manager.getMediaTime();
    manager.setText("F:PM_Timeline_popup.Timer",
                    manager.time2str(time));

// Set time progress bar through clipping area.
// Bar starts at BarX and has width BarLength pixels
time = manager.getMediaTimeInSeconds();
time = time * BarLength / RunningTime + BarX;
manager.setClipping("F:PM_Timeline_popup.Timeline",
                    0, 0, (int)time, 1079);

// Determine number of bookmarks present for that movie
nrBookmarks = manager.getBookmarksCount(playlistID);

// Initially set all indicators off the timeline as
// if there are no bookmarks

// Note that this only works if the
// (dropdown menu)> Change Effects for the marker and bookmarks have
// "Move container" checked!
manager.moveToXY("F:PM_Timeline_popup.Bookmark_1", 332, IndicatorY);
manager.moveToXY("F:PM_Timeline_popup.Bookmark_2", 332, IndicatorY);
manager.moveToXY("F:PM_Timeline_popup.Bookmark_3", 332, IndicatorY);
manager.moveToXY("F:PM_Timeline_popup.Bookmark_4", 332, IndicatorY);

// no bookmarks
// Marker is at height MarkerY the bookmark indicators at IndicatorY
if (nrBookmarks < 1) {
    manager.setText("F:PM_Timeline_popup.Bookmark_Time",
                    "BOOKMARK " + "EMPTY");
    manager.moveToXY("F:PM_Timeline_popup.Marker", 332, MarkerY);
}

// set the marker to the current bookmark
if (nrBookmarks > 0){
    manager.setText("F:PM_Timeline_popup.Bookmark_Time",
                    "BOOKMARK " +
                    manager.currentBookmarkText(playlistID));
    time = manager.currentBookmarkTime(playlistID);
    time = time / 1000000000L * BarLength / RunningTime + BarX;
    manager.moveToXY("F:PM_Timeline_popup.Marker",
                    (int)(time - MarkerWidth/2), MarkerY);
}

// position first bookmark
if (nrBookmarks >= 1) {
    time = manager.getBookmarkTime(playlistID, 1);
    time = time / 1000000000L * BarLength / RunningTime + BarX;
    manager.moveToXY("F:PM_Timeline_popup.Bookmark_1",
                    (int)time, IndicatorY);
}

```

```

// position 2nd bookmark
if (nrBookmarks >= 2) {
    time = manager.getBookmarkTime(playlistID, 1);
    time = time / 1000000000L * BarLength / RunningTime + BarX;
    manager.moveToXY("F:PM_Timeline_popup.Bookmark_2",
        (int)time, IndicatorY);
}

// position 3rd bookmark
if (nrBookmarks >= 3) {
    time = manager.getBookmarkTime(playlistID, 1);
    time = time / 1000000000L * BarLength / RunningTime + BarX;
    manager.moveToXY("F:PM_Timeline_popup.Bookmark_3",
        (int)time, IndicatorY);
}

// position 4th bookmark
if (nrBookmarks >= 4) {
    time = manager.getBookmarkTime(playlistID, 1);
    time = time / 1000000000L * BarLength / RunningTime + BarX;
    manager.moveToXY("F:PM_Timeline_popup.Bookmark_4",
        (int)time, IndicatorY);
}

manager.activateButtonEx("H:PM_Timeline_popup.Handler",
    "TimelineButton", false);

```

Message-Close.txt

The code to this action is performed on all of the buttons and autoclose events of the “Bookmark Added”, “Bookmark Deleted” and “Too many bookmarks” messages that are displayed as a menu. If triggered from an action on the Timeline popup menu, it reopens this popup (the indication is when GPR(5) has value 1). In all cases, the register value is reset to 0.

```

// close the message menu (Add/Delete/Too many bookmarks)

// check if the Timeline popup menu must be reopened (GPR(5)==1)
if (manager.getGPR(5)==1) {
    manager.setGPR(5, 0);
    manager.activateButtonEx("H:PM_Timeline_popup.Handler",
        "TimelineButton", false);
    manager.activateSegment("S:PM_Timeline_popup.animate1");
}
else {manager.Close_Popup();
}

```

Timeline-green.txt

Code to add a bookmark as long as there are fewer than 4. Again, we first need to find out which movie is currently playing. To make sure the menu reopens after the “Bookmark Added” menu is displayed, the GPR 5 is used as “comeback” flag if its value is set to 1. (This code checks once more there are no more than 4 bookmarks set – this check is not strictly needed as the invocation of this routine only happens when this condition is met).

```

// add bookmark

int playlistID;

// determine running time of current movie

playlistID = manager.getPlaylistID();

if (manager.getBookmarksCount(playlistID)>=4) {
    // show warning menu
    // - this closes the Timeline popup and reopens it if GRR(5)=1
    manager.setGPR(5, 1);
}

```

```

manager.activateButtonEx("H:PM_Too_Many_Bookmarks_popup.Handler",
                        "Too_Many_Bookmarks", false);
manager.activateSegment("S:PM_Too_Many_Bookmarks_popup.show_menu");
}
else {
manager.addBookmark();
// show informative menu - this closes the Timeline popup,
// but will be reopened by the menu's AutoClose action
// and popup action provided GPR 5 is set to 1 ("comeback" flag)
manager.setGPR(5, 1);
manager.activateButtonEx("H:PM_Add_Bookmark_popup.Handler",
                        "Add_Bookmark_popup", false);
manager.activateSegment("S:PM_Add_Bookmark_popup.show_menu");
}

```

Timeline-red.txt

Code to delete a bookmark as long as there is at least one. Again, we first need to find out which movie is currently playing. To make sure the menu reopens after the "Bookmark Deleted" menu is displayed, the GPR 5 is used as "comeback" flag if its value is set to 1. The red button only functions within the Timeline popup menu.

```

// delete current bookmark

int playlistID;
playlistID = manager.getPlaylistID();

if (manager.getBookmarksCount(playlistID) > 0){
    manager.deleteBookmark(playlistID);

    // show informative menu - this closes the Timeline popup,
    // but will be reopened by the menu's AutoClose action and
    // popup action provided GPR 5 is set to 1 ("comeback" flag)

    manager.setGPR(5, 1);
    manager.activateButtonEx("H:PM_Delete_Bookmark_popup.Handler",
                            "Delete_Bookmark_popup", false);
    manager.activateSegment("S:PM_Delete_Bookmark_popup.show_menu");
}

```

Timeline-blue.txt

Code to delete all bookmarks. Again, we first need to find out which movie is currently playing. It works without informative messages from both movie and Timeline popup menu.

```

// remove all bookmarks

int playlistID;

// check the movie that's playing
playlistID = manager.getPlaylistID();

// remove all its bookmarks
while (manager.getBookmarksCount(playlistID) != 0){
    manager.firstBookmark(playlistID);
    manager.deleteBookmark(playlistID);
}

```

Timelinebutton-next.txt

This is code for the invisible button to move to the next bookmark. When you're on the last bookmark, nothing happens. Some code to identify the movie playing as well as its running time is identical to the code used for opening the Timeline popup. (I removed some comments to make it smaller)

```

// next bookmark

```

```

// position data of Timeline bar
int BarX = 335;
int BarLength = 1244;

// position settings vertically from top of screen for
// bookmark indicators and marker
int MarkerY = 889;
int IndicatorY = 908;
int MarkerWidth = 12; // point inverted triangle is in the middle

int playlistID;
long RunningTime;
long time;

// determine running time of current movie
RunningTime = manager.getDurationInSeconds();
playlistID = manager.getPlaylistID();

if (manager.getBookmarksCount(playlistID) > 0){
    // go to next bookmark. If already on last one, stay there
    manager.nextBookmark(playlistID);

    // move marker there
    manager.setText("F:PM_Timeline_popup.Bookmark_Time",
        "BOOKMARK " + manager.currentBookmarkText(playlistID));
    time = manager.currentBookmarkTime(playlistID);
    time = time / 1000000000L * BarLength / RunningTime + BarX;
    manager.moveToXY("F:PM_Timeline_popup.Marker",
        (int)(time - MarkerWidth/2), MarkerY);
    manager.activateButtonEx("H:PM_Timeline_popup.Handler",
        "TimelineButton", false);
}

```

Timelinebutton-previous.txt

This is code for the invisible button to move to the previous bookmark. When you're on the first bookmark, nothing happens.

```

// previous bookmark

//... we won't repeat identical code from "next bookmark"

...

if (manager.getBookmarksCount(playlistID) > 0){
    manager.prevBookmark(playlistID);

    // move marker there
    manager.setText("F:PM_Timeline_popup.Bookmark_Time",
        "BOOKMARK " + manager.currentBookmarkText(playlistID));
    time = manager.currentBookmarkTime(playlistID);
    time = time / 1000000000L * BarLength / RunningTime + BarX;
    manager.moveToXY("F:PM_Timeline_popup.Marker",
        (int)(time - MarkerWidth/2), MarkerY);
    manager.activateButtonEx("H:PM_Timeline_popup.Handler",
        "TimelineButton", false);
}

```

Timelinebutton-OK.txt

This is code for the invisible button to start playing the current movie from the current bookmark onwards.

```

// play from current bookmark

int playlistID;

// find out what movie plays
playlistID = manager.getPlaylistID();

```

```
// jump to the selected bookmark and play from there
manager.jumpBookmark(playlistID);
```

Further enhancements

Function instead of action

All unused bookmarks are now collected and shown at the far left of the timeline. You may define a small black rectangle to cover them up and make them invisible.

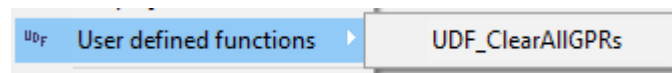
As of BDS MX V4.1.0.1072 it is also possible to store identical duplicate Java code in an User Defined Function (of type void). These are defined in Project > Project Settings > Functions tab.

You simply copy/paste the code from an action script into the UDF function. You should to remove all “manager.” prefixes that BDS functions use from the Java code. Since the UDF’s are maintained within the “manager” BDS object, the UDF code must not use the object name “manager.” itself as it already functions within this object.

So, an external Java action reference to `manager.getPlaylistID()` becomes `getPlaylistID()` as a UDF function.

As of V4.1.0.1676 the removal of the prefix is no longer needed (BDS strips the “manager” namespace on all lines in UDFs).

Then the actions that used to contain (a copy of) the code can now simply call it by using the “User defined functions” option from the action dropdown list.



That generates a very small Java simply calling `manager.UDF_functionname`.

The project has 2 project files: the “project.bdmnd” file that contains the project as you would build it yourself following the instructions in this guide. The “project – using UDF.bdmnd” file builds the same project but with most scripts transferred to UDF functions.

Note: Before transferring Java code to an UDF, make sure the code is generic enough. If specific references are made to a menu object that is unique to a menu, you cannot transfer the code to an UDF. When it would be called from a menu that does not have that object, a reference to such object would cause a runtime error.

Loop instead of repeated code

Some of the code for setting visible bookmarks can be made shorter using a loop:

```
// set visible bookmarks
for (int iBM = 1; iBM <= nrBookmarks; iBM++) {
    time = getBookmarkTime(playlistID, iBM);
    time = time / 1000000000L * BarLength / RunningTime
+ BarX;
    moveToXY("F:PM_Timeline_popup.Bookmark_" + iBM,
(int) time,
```

```
IndicatorY);  
}
```

More flexibility

The number of bookmarks you can set is hardcoded. You may decide to use a User Defined Variable (UDV) for this value.

Everywhere in the code where the hardcoded value (4 in the example) is used, you can refer to the variable instead.

For example, given an integer UDV called MaxBookmarks you refer to it in setting the initial position as

```
for (int nr = 1; nr<=UDV_MaxBookmarks;nr++) {  
    manager.moveToXY("F:PM_Timeline_popup.Bookmark_"  
        + nr, 332, IndicatorY);  
}
```

Note that inside Java code the UDV is recognized by its prefix UDV_ (which you cannot set as part of its name when you define the UDV).

You will still need to create that number of bookmark objects in the popup menu – BDS does not allow for dynamically created menu items.

Project 14: Playing an A-B loop

Project Goal

It is fairly straight forward to create a play loop where the same piece of movie is repeated from position A to a later position B. May discs allow this.

The “A” time is stored by pressing the red button on the remote-control, the “B” time by the green button. The loop is entered and played repeatedly (until stopped) by the blue button.

Some remote-controls also have a special A-B button but that one is not part of the “Remote-control buttons” defined in BDS.

We need to define two UDV's that can store the current position time in the movie at points A and B. Then invoke the play of a loop where we check that the time position remains between A and B.

A Java conditional statement is always written as

```
if ( condition ) { statements if condition is true }
```

A Java statement always ends with a semicolon (“;”)

To focus on this one aspect, we reuse the results of Project 2, the “MenuBR” project.

If you want to create this project yourself, the source files can be found in the .zip file mentioned in Appendix A: The user's guide source files.

Project steps to take (UDV)

Ready to run

Take the following steps:

1. Copy from the examples the folder \Projects\ABLoopBR and paste it in your local project's folder (e.g. J:\BDS Projects\ABLoopBR)
2. Copy the movie files from the examples \Sources\movies\demuxed\Australia*.*, Coasts*.* and menu*.* to the local project folder \films
3. Copy the movie subtitles files from the examples \Sources\movies\subtitles*.* to the local project's \films folder.
4. Double click on the project.bdmd file in the local \ABLoopBR folder to start BDS and the project.
5. Mux the project
6. Experiment!

Build your own

Take the following steps:

- Copy the \MenuBR project from the local project tree into a new project in the same local project tree and rename it \ABLoopBR (e.g. \BDS Projects\ABLoopBR)
- Open Project > Project Properties > Functions tab.
- Define two “long” variables A and B to hold the time values (number in nanoseconds) as provided by the Java function getMediatime.

A screenshot of a 'Variable' dialog box. It has three input fields: 'Name' with the value 'A', 'Type' with a dropdown menu showing 'long', and 'Value' with the value '0L'. There are 'OK' and 'Cancel' buttons at the bottom right.

In the Advanced tab, click on “Add” under the “user defined parameters” and add both variables.

- The terse description in the online help on “Available Java functions” says

long manager.getMediaTime() - returns the time for the current playlist from player in nanoseconds;

Define one “boolean” variable that functions as flag to indicate whether the time pointer is inside or outside the loop

User defined parameters

```
long A = 0L
long B = 0L
boolean inLoop = false
```

- Program the red button for the movie to set the variable A equal to the starting time of the loop. This requires to use the “Script” action option. A single line of Java code suffices. Remember all user defined variables are defined in the namespace “manager” and get UDV_ prefixed by BDS to avoid name collision. And also remember Java is case sensitive!

```
Manager.UDV_A = manager.getMediaTime();
```

Script

```
manager.UDV_A = manager.getMediaTime();
```

- Do the same for the end time of the loop, when the green button is pressed. The code is identical.
- The movie that allows A-B looping now has two assignments for the red and green button.

Red	manager.UDV_A = manager.getMediaTime();
Green	manager.UDV_B = manager.getMediaTime();

- The movie must check every second to ensure the time pointer is still between A and B. Therefore its “Action Every Second” must contain a script with the following content:
 - a. Check we’re inside the loop by checking the value of inLoop. It should be as (see next step) the playback has been started by the blue button from A onwards.

- b. For sanity reasons, time A should be larger than B to play the loop A-B. It uses the predefined function `getMediaTime()`.
- c. If both conditions (a) *and* (`&&` operator) (b) are fulfilled, don't do anything until the time pointer exceeds B in which case we restart the movie from A using function `playVideoFrom()` that is described in online Help as

void `manager.playVideoFrom(int playlistId, long time)` - starts the specified playlist from the specified time in nanoseconds;

void `manager.playVideoFrom(#string movieName, long time)` - same as above;

It requires two parameters: the moviename (or `playlistId` which we do not know) "Movie" and the time pointer of A.

The combination of steps (a), (b) and (c) result on the script code shown below for the movie's "Action every second".

```
If (manager.UDV_inLoop && manager.UDV_B >
manager.UDV_A) {
    if (manager.getMediaTime() >= manager.UDV_B) {
        manager.playVideoFrom('Movie',manager.UDV_A);
    }
}
```

 Script

```
if (manager.UDV_inLoop && manager.UDV_B > manager.UDV_A) {
    if (manager.getMediaTime() >= manager.UDV_B) {
        manager.playVideoFrom('Movie', manager.UDV_A);
    }
}
```

- The blue button must start the playing of the loop. This must also be a small Java at the action for the "Blue" button
 - d. The loop flag `inLoop` is set to true
 - e. Play the movie (titled 'Movie' – replace this by 'Australia' and 'Coasts') starting from A. Leave it to the "Action Every Second" part to ensure the play stops at B and returns to A.

```
manager.UDV_inLoop = true;
manager.playVideoFrom('movie', manager.UDV_A);
```

 Script

```
manager.UDV_inLoop = true;
manager.playVideoFrom('Movie', manager.UDV_A);
```

- We're done. If all is correct, we have properties for "Movie" set through [Script] as:
 - f. A red button action assigning time A
 - g. A green button action assigning time B
 - h. An action to check every second checking time pointer is and remains between A and B
 - i. A blue button action to start it all off. Set the loop flag and start playing from A

Jump between movies (from feature to making of and vice versa)

You may encounter one of the following two possible situations:

- you got a feature film with many separate “making of” movies that show how a scene was made. You would like to jump from the feature film into one of those “making of” scenes.
- you got a feature film and a single “behind the scenes” movie about the same feature film. You’d like to jump from the feature film to the “behind the scenes” movie describing the scene.

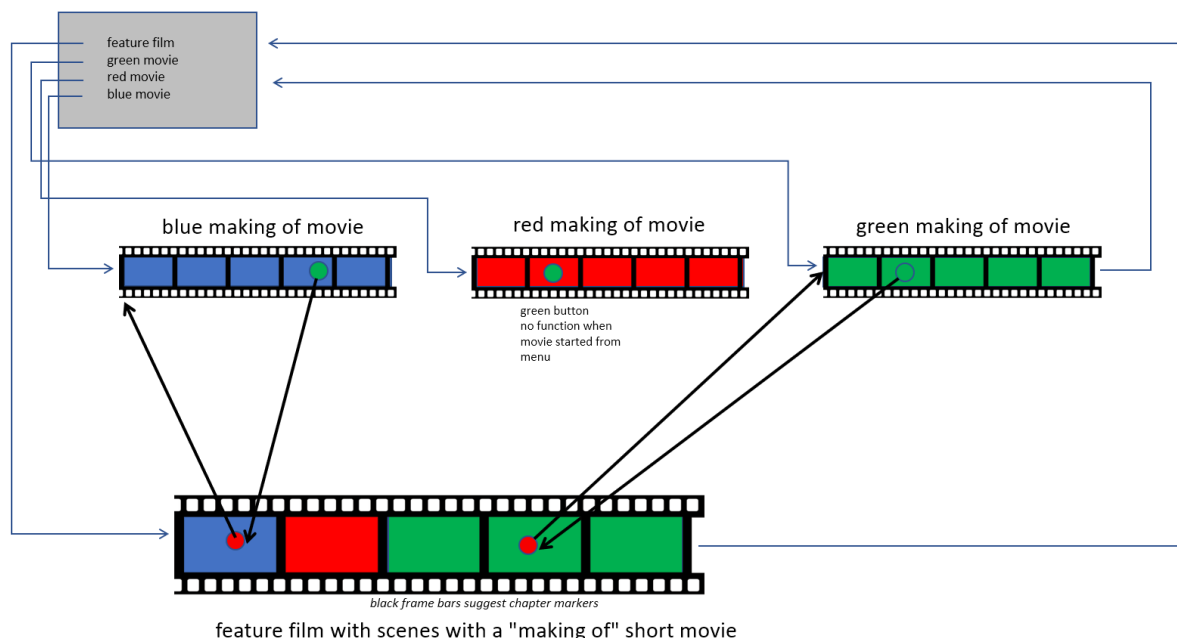
In both cases you’d like to jump back to the feature film once the extras are shown or interrupted. Those “making of” or “behind the scene” movies can also be played by themselves.

With extensive (and repeated) settings of BDS properties you can achieve both ways. You can also do it, with less effort, by programming it in a few Java user defined functions and a SWITCH action to determine where to go to. It can even be quicker by programming the logic of the SWITCH action.⁴⁸

The next two sections describe both situations.

From feature film to one of many “making of” short movies – option 1

The complete project is available in the BDS kit\Projects\JumpExtraMovieBR\MakingOf.bdmd if you want to run the project immediately and inspect it yourself.



⁴⁸ If you discover that chapters always restart a title from chapter 1, see the note made in section Important for tsMuxer users on page 57

The construction in the figure above shows how to have a single long movie and jump to one of several short ones that are related to a chapter (or set of chapters) in the long movie.

The menu allows to select any of the three short movies (blue, red or green) and the feature film has chapters to the scenes that have a “making of” movie.

When the feature film plays, the “red button” on the remote control activates a function that produces a jump to the proper making of short movie. Once this movie plays the “green button” on the remote control returns to the “making of” movie at the point where we left off. (you may want to define the same red button for this instead)

Once the menus and movie placeholders are defined in BDS, additional attention to some items is needed.

Things to think of:

- Chapter marks must be placed in the feature film at the beginning of each of the scenes that belong to a different “making of” movie (in the example: at start of the blue chapter, the red chapter and green chapter). Additional chapters can be defined within each section if required as well as in scenes not linked to any making of movie.
- When the viewer fast forwards or reverses the feature film, the correct movie link must be set between feature and making of movie. (this requires a “run every second” action – changing the setting only at chapter marks will not work as fast forwarding will skip such marks).
- actions must be attached to the red and green button of the remote control but these actions must only function when you jump from the feature film to the making of movies. When those movies are started individually the buttons should not do anything.
- We will ignore this in the text below, but if any movie has a specific “popup menu” you may need to extend its functionality to also allow movie jumping.

We define the following UDF Java functions that will take care of these requirements. You edit Java functions through Project > Project Settings > Functions tab and add the function with the specified name.

[enableJump](#)

This function is called to set register GPR 200 to a value of 1 to indicate that the “feature film plays and any jump from it to a making of movie should be allowed to return to the feature film. When any of the other movies is played by themselves, the register is not set. The ident by which BDS knows the feature film is its BDID that is filled in by the BDS pre-processor for all movie placeholder names surrounded by single quotes. This ident is stored in register 201.

```
public void enableJump() {  
    // Activated when red button on remote is pressed  
    // set up info to return to feature film
```

```
// 1. set GPR 200 as jump flag = 1
// 2. set GPR 201 as return BDID of feature film

// Ensure that all movies are enabled to resume (have
// "Allow save state" checkbox checked in their
// properties)
    setGPR(200,1);
    setGPR(201, 'feature film');
}
```

disableJump

The flag to indicate a jump to a movie can be made, but be cleared if any other movie than the feature film plays. So register 200 must be set to zero.

```
public void disableJump() {
    // invoke before an individual movie is started to
    // disable the green button to function.

    // set jump flag off
    setGPR(200,0);
}
```

jumpMovie

This function will be executed when the red button is pressed during the playback of the feature film. It stores the current play position (though the `saveCurrentResumeTime` function of BDS) before jumping to the movie whose ID is specified in register 202 (that value is set by the `SetJumpBDID` function at chapter changes of the feature film). The BDS function `playVideo` does this.

```
public void jumpMovie() {
    // jump to indicated movie when red button is pressed

    // check if jump is enabled
    if (getGPR(200)== 1) {
        // jump to movie indicated in GPR 202
        saveCurrentResumeTime();

        playVideo(getGPR(202));
    }
}
```

setJumpBDID

To jump to the proper movie from the “making of” you must know to what movie to jump to. The ident of that movie is stored in register 202 but this register must be filled. This is done by checking every second to where the “making of” movie has progressed and in what chapter it currently is. Using the “action every second” property of the movie ensures the check fills the register 202 with the proper movie ident always – regardless of whether the viewer fast forwards or reverses the playback of the “making of” movie.

```
public void setJumpBDID(int movieID) {
    // this function is called from a direct edit chapter
    // action to set the movie to jump to if the red button
    // is pressed
    // The movie has a BDID number only known at time of
    // muxing. Hence this conversion from name to number
    // that cannot be done directly in the chapter action
    // It is called as setJumpBDID('blue') if "blue" is
```

```

// the name of the movie placeholder containing the
// blue making of movie.

// Register GPR 202 contains the movie ID to jump to
setGPR (202, movieID);
}

```

The function has one argument, “movieID”, that needs to be provided when it is called. This is achieved by using a SWITCH action for the “action every second” property of the feature film. Remember that BDS pre-processes all functions and changes all movie placeholder names in single quotes by their ident that has become known during the muxing of the movie. Hence ‘blue movie’ will be converted into an integer value of, for example, 5 if 5 is the ident of the movie associated with placeholder “blue movie”.

returnMainMovie

Finally a function is needed for each of the movies to return to the “making of” movie from the spot where the jump was made. The resumeVideoAt function does this. The feature film ident was stored in register 201 by the “enableJump” function.

The return should only happen when we jumped from the feature film and ignored otherwise. Checking the value in register 200 allows this decision to be made.

```

public void returnMainMovie() {
// return to feature film if green button is pressed after
// excursion into making of movie.

// check if jump mode is enabled first
if (getGPR(200)==1) {
    // jump back to movie ID stored in GPR 201 and resume
    resumeVideoAt(getGPR(201));
}
}

```

doNothing

It won’t be needed in our example, but there will be sections in the feature film that do not refer to any particular movie. In those cases, the SWITCH statement that updates the movie ID to jump to must activate a function that does nothing, i.e. continues running the feature film.

It is the simplest user defined function imaginable:

```

public void doNothing() {
    // for those sections of the feature film that
    // are not linked to any making of movie, simply
    // continue running
}

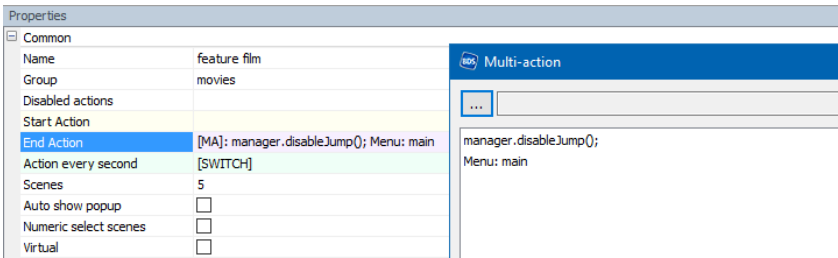
```

BDS movie property changes

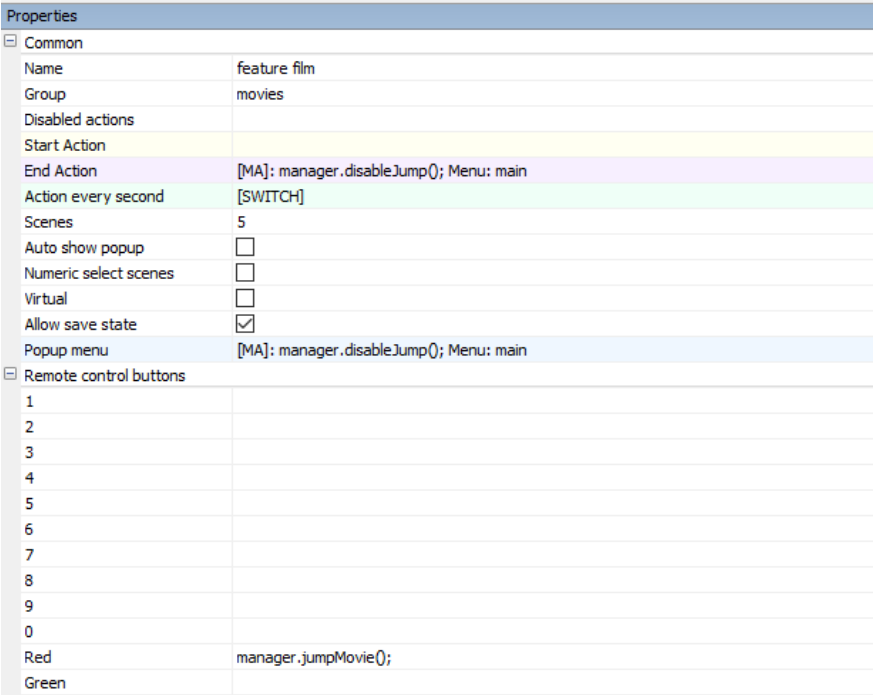
Finally these functions will only work if they are attached to properties of the movies. Select the property, click right mouse, select the “user defined function” menu option and select the earlier defined (UDF) function.

Feature Film properties

- To allow jumping from this movie, the jump flag must be set through enableJump function activated by the menu button that starts the movie.⁴⁹

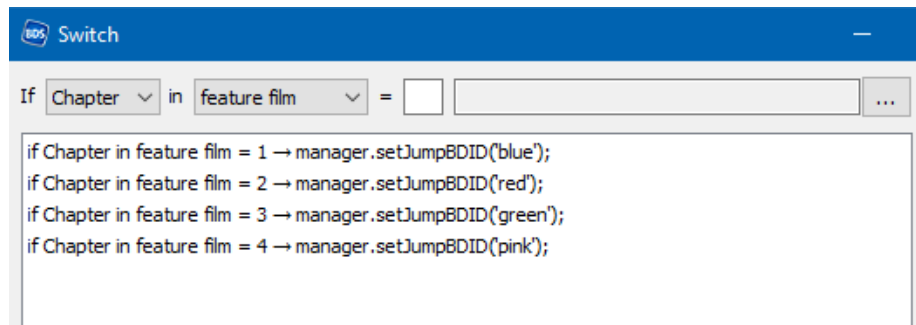


- The EndAction must clear the jump flag and return to the menu with movie titles. This requires a MultiAction (MA). This may also apply to the “popup menu” action if that allows to abort the movie play.
- To jump to the proper movie each second it should be checked what chapter currently plays and what movie is associated with it: a SWITCH statement in “Action Every Second”. .
- The remote control “red” button must be set to activate jumpMovie



The SWITCH statement for Action Every Second contains:

⁴⁹ You might think Start Action is the more appropriate action. It might, but this action is *always* performed whenever the movie starts – even if not from the start.



Note the names of the movie placeholders surrounded by single quotes. Also all chapter numbers must be tested as the condition allows only for “=” (equal) tests.

Each of the making of movies

- Their EndAction property must be a SWITCH action that is set to returnMainMovie if the jumpflag in GPR 200 is set and otherwise returns to the menu
- Their remote control “green” button must be set to returnMainMovie.

You should not set these movies’ “Start Action”⁵⁰ to disableJump since that disables jumps from the “making of” movie as at the start of the jump starts the movie where the jump is disabled.

Red	
Green	manager.returnMainMovie();
Yellow	

From feature film to one of many “making of” short movies – option 2

There is a more efficient way to achieve the same thing as option 1. Rather than executing an “Action every second” each second, nothing is done until the red button is pressed and everything is checked then. This reduces the processing time enormously and also allows just to specify actions for specific chapters with a “making of” movie. All other chapters are handled in one go: do nothing.

We’ll visit all items of the previous section and highlight their changes.

setJumpBDID

This function is no longer needed as the Action Every Second switch using this function no longer exists.

jumpMovie

The only thing required is a re-write to the jumpMovie function activated when the red button is pressed. It tests the current chapter (through the BDS getChapter function) and uses a switch Java statement to determine if and what jump to perform.

Again note that the movies are referred to by their movie placeholder name in single quotes. The BDS pre-processor replaces these names by

⁵⁰ Note that the Start Action is always executed – even if you jump to a chapter in the middle of the movie

their BDID playlist numbers which are only known once the muxing process starts.

```
public void jumpMovie() {
    // jump to indicated movie when red button is pressed
    // This routine checks the current chapter number of the
    // making of" movie and decides what real movie to show.

    // check if jump is enabled
    if (getGPR(200)== 1) {
        // remember where to return to in the feature film
        saveCurrentResumeTime();

        // Determine what making of movie to play.
        // Chapters not associated with making of movies
        // arrive at the "default" catch-all does nothing.

        switch (getChapter(getGPR(201))) {
            case 1 : {playVideo('blue') ; break; }
            case 2 : {playVideo('red') ; break; }
            case 3 : {playVideo('green') ; break; }
            case 4 : {playVideo('pink') ; break; }
            default : {break;} // do nothing
        }
    }
}
```

If several chapters should jump to the same section of the movie, the case statement is repeated:

```
case 1: case 3: case 5: {playVideo('blue'); break;}
case 2: case 4:         {playVidet('red'); break;}
```

If several chapters do not refer to a particular movie, don't mention them: they will be handled by the "default" switch value:

```
case 1 : {playVideo('blue') ; break; }
case 8 : {playVideo('red') ; break; }
case 10 : {playVideo('green') ; break; }
case 33 : {playVideo('pink') ; break; }
default : {break;}
```

Especially if many chapters are involved, the use of a simple UDF written function is a big time saver above specifying a long SWITCH statement as the below real-life sample shows on a movie with over 150 chapters out of which only a number are connected to a making of movie.

```
public void jumpMovie() {
+
    int chapterNr;

    // only jump if you are allowed from this movie
    if (getGPR(200) == 1){
        // find out what chapter we're on
        chapterNr = getChapter(getGPR(201));

        // note down resume time to return to after
        // jump movie completed
        saveCurrentResumeTime();

        // select movie to jump to
        switch (chapterNr) {
            case 3: case 5: case 9 : case 11 :
                {playVideo ('1 01'); break;}
            case 20 : case 22 : case 24 :
                {playVideo ('1 02'); break;}
        }
```



```

        case 26 : case 27: case 29: case 31 :
            {playVideo ('1 03'); break;}
        case 57 : case 58 :
            {playVideo ('1 04'); break;}
        case 93 : case 99 : case 100: case 101:
            case 103: case 105:
            {playVideo ('1 05'); break;}
        ...
        case 172 :
            {playVideo ('1 12'); break;}
        case 81: case 83: case 87 :
            {playVideo ('1 14'); break;}
        case 169 : case 176: case 178 :
            {playVideo ('1 15'); break;}
        default: {break;} // do nothing
    }
}
}

```

BDS movie property changes

Feature Film movie properties

We do not need the “Action Every Second” action specified for the feature film.

And the invocation of the jumpMovie function is still needed as action for the red button of the remote control.

Properties	
Common	
Name	feature film
Group	movies
Disabled actions	
Start Action	
End Action	[MA]: manager.disableJump(); Menu: main
Action every second	
Scenes	5
Auto show popup	<input type="checkbox"/>
Numeric select scenes	<input type="checkbox"/>
Virtual	<input type="checkbox"/>
Allow save state	<input checked="" type="checkbox"/>
Popup menu	[MA]: manager.disableJump(); Menu: main
Remote control buttons	
1	
2	
3	
4	
5	
6	
7	
8	
9	
0	
Red	manager.jumpMovie();
Green	

Each of the making of movies

No change - same as for option 1.

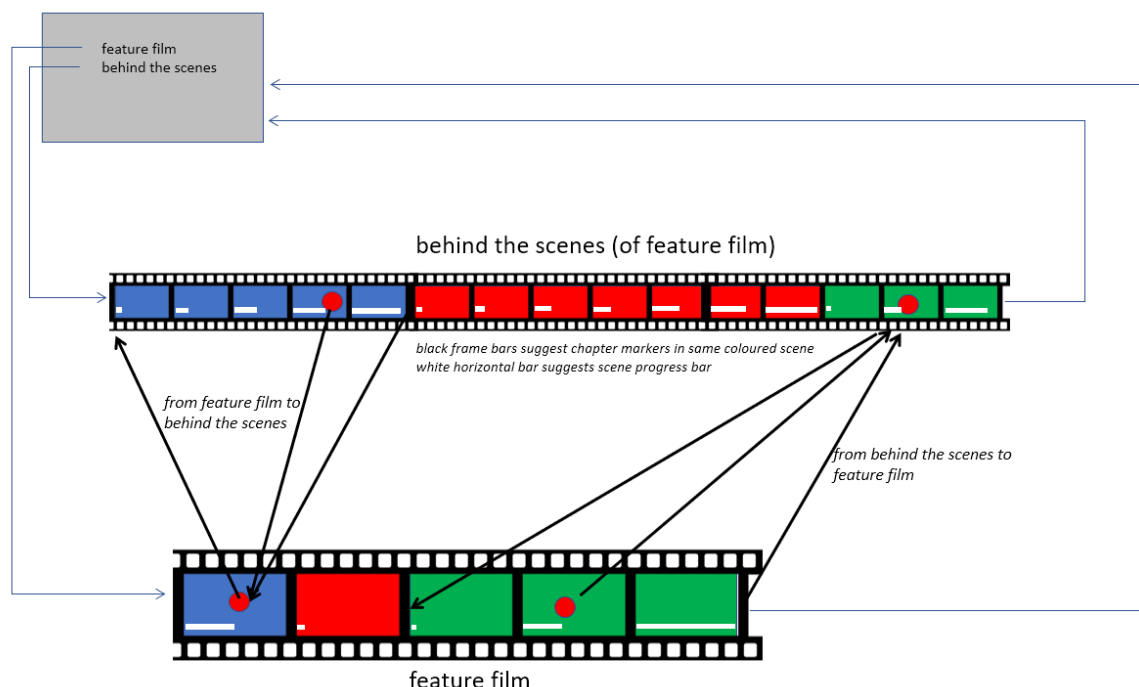
From feature film to one of many “making of” short movies – option 3

Rather than addressing individual “making of” movies you can consider making all those movies part of one long playlist movie. In that case you will want to structure the project as described in the section below

since the playlist movie now becomes the “behind the scenes” movie and each making of movie is a chapter in that movie.

From feature film to behind the scenes movie chapter and vice versa

The complete project is available in the BDS kit\Projects\JumpExtraMovieBR\ Jump between two movies.bdmd if you want to run the project immediately and inspect it yourself.



Here we have two movies that are linked to each other by covering the same scenes. One as part of the movie (“feature film”) and one showing how it was done (“behind the scenes”).

It is a more elaborate example of the “making of” section project in that it:

- uses a single “red button” to jump from one movie to the other and back
- you always jump to the start of the corresponding scene in the other movie
- you resume playback of the movie from where you started
- when you jumped you will automatically return at the end of the scene in the other movie.

It is illustrated above: if you start with the ‘feature film’ and you press the red button on the remote control while the blue scene is playing, you are moved to the “behind the scenes” movie and start playing from the start of its blue scene section. Any time you press the red button again, you return to resume the feature film. The same happens if you do nothing but the “behind the scenes” blue section ends.

If you start playing the “behind the scenes” movie and press the red button while the green section plays, you move to the start of the

green section of the feature film. Any time you press the red button you return to and resume the “behind the scenes” movie. Or arrive there when the green section of the feature film ends.

Only a few UDF Java functions are needed for this. To remember where we started off, a flag is set in the GPR 200 register if we run the feature film and a flag in GPR 300 is set if we run the behind the scenes movie.

In this project two movies are made: a feature film and a behind the scenes movie.

- The scenes are represented by differently coloured screens.
- The progress within a scene is shown by a white progress bar at the bottom that increases from nothing at the start of the scene to full width at the end of the scene
- A chapter point is set at the start of each scene
- A chapter point is set at the last I-frame of each scene
- Some more chapter points are set somewhere in the middle of a scene

The following user defined functions (UDF) are required:

[jumpToBTChapter](#)

This function checks if we depart from the feature film (flag GPR 200). If not, nothing happens. Otherwise the current chapter in the feature film is determined and that is linked to the start of the corresponding section in the “behind the scenes” movie.

Note that several chapters in the feature film are part of same scene but each chapter is mentioned as part of the case values. Also, the startVideoAt function of BDS requires a playmark ident rather than a chapter ident. The playmark ident value is the same as the chapter value minus one. In the example we always go to the first chapter of a scene and note this explicitly for reference (such as 7-1, 13-1) though of course you can do the maths immediately (using 6 and 12 instead).

The names of the movie placeholders are given explicitly though there are other ways to store their playlist ident. Surrounding them in single quotes ensures that at muxing time BDS will replace those names by the playlist idents of the movies. Those numbers are not known in advance.

```
public void jumpToBTChapter() {  
  
    // jump from feature film to behind the scenes at  
    // specific chapter when red button is pressed  
  
    // only jump if jump flag feature film is set  
    if (getGPR(200) == 1) {  
        // register the playing time to return to  
        saveCurrentResumeTime();  
  
        // determine where to jump to  
        switch (getChapter('behind the scenes')){  
            case 1: case 2: case 3:  
                {startVideoAt('behind the scenes', 1-1); break;}  
            case 4: case 5: case 6:  
                {startVideoAt('behind the scenes', 4-1); break;}  
        }
```

```

        case 7: case 8: case 9:
            {startVideoAt('behind the scenes', 7-1); break;}
        case 10: case 11: case 12:
            {startVideoAt('behind the scenes', 10-1); break;}
        case 13: case 14: case 15:
            {startVideoAt('behind the scenes', 13-1); break;}
        default : {break;}
    }
}
}

```

jumpToFeatureChapter

This function is identical to jumpToBTSCChapter but in reverse: you must start from “behind the scenes” so GPR 300 must be set to use the switch statement. Inside the switch statement you startVideoAt of the ‘feature film’ rather than the ‘behind the scenes’. Because in this example both films have chapters at the same spots and the same scenes, the case values and chapter values are the same.

resumeMovie

This function returns to the originally started movie and resumes playback from the time point saved before the jump was made. Depending on the flag GPR 200 or GPR 300 the corresponding movie is resumed (name of movie placeholder again between single quotes)

```

public void resumeMovie() {

    // return to the movie from where a jump was made and
    // resume this from point of jump away

    // check what movie to resume
    if (getGPR(200)==1){
        // resume feature film
        resumeVideoAt('feature film');
    }
    else if (getGPR(300)==1) {
        resumeVideoAt ('behind the scenes');
    }
}

```

BDS menu button property changes

The setting of the flags that determine whether the feature film or the behind the scenes movie was started is done by the menu button that activates the movie. Do not use the “Start Action” of the movie placeholder. That action is always executed when that movie starts playing – whether from the start or somewhere in the middle.

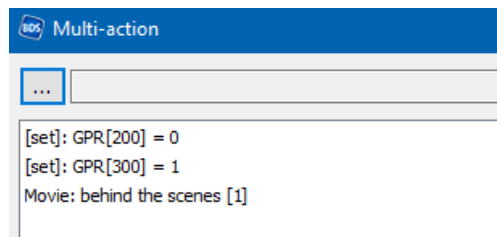
The button must execute several actions so a Multi-Action is required. For the feature film button the actions are:

- set GPR 200 = 1
- set GPR 300 = 0
- start playing the movie ‘feature film’ from the beginning’

Properties	
Common	
Name	feature film
Selected state	right arrow button green small.png
Effect	
Top	398
Left	586
Actions	
← Press Left	
anim left	
→ Press Right	
anim right	
↑ Press Up	Button: behind the scenes
anim up	
↓ Press Down	Button: behind the scenes
anim down	
OK: Press ENTER	[MA]: [set]: GPR[200] = 1; [set]: GPR[300] = 0; Movie: feature film [1]
anim enter	

For the behind the scenes movie button the actions are similar:

- set GPR 200 = 0
- set GPR 300 = 1
- start playing the movie 'behind the scenes' from the beginning.

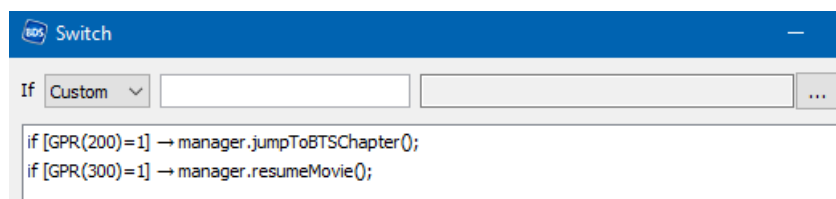


BDS movie property changes

Both movies must enable the red button functionality in their properties. But different functions should start, depending on what movie was started. That requires a SWITCH statement in the button action⁵¹

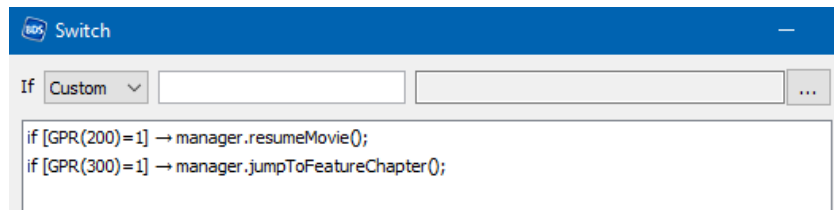
- if the feature film started, and then the red button is pressed, a jump to behind the scene is required. If we already jumped to the behind the scenes, then the red button must resume the feature film
- if the behind the scenes movie was started the same logic applies but in reverse.

The red button for the feature film contains:



The behind the scenes movie action for the red button is similar:

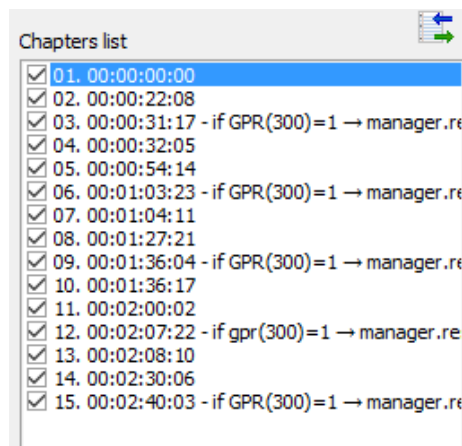
⁵¹ Rather than a SWITCH for each movie RED BUTTON property, a new UDF function can be written that performs the switch through if-then-else if depending on the setting of the GPR 200 and GPR 300 register flags. Then only this UDF is invoked instead of the SWITCH statement.



BDS movie chapter changes

If you want to ensure the viewer automatically resumes his movie once the scene in the jumped-to movie ends, we need to insert a chapter mark just before the scene ends and assign an action to that chapter though “direct editing” of that chapter. All it does is invoking the same UDF function otherwise invoked by the red button: resumeMovie. Provided of course, we jumped into this movie and did not start it. The value in GPR 200 and GPR 300 will tell. A SWITCH statement is needed to check this condition.⁵²

For the feature film, these chapters are the last ones of the case values for a particular scene as shown in the figure below. If we started with the feature film, GPR 200 = 1 and GPR 300 = 0. In that case the resume is ignored. Had we come from the behind the scenes movie, we resume it.



Caveat

This return at the last chapter of a section will suffice if a viewer watches a scene to its end. If he starts to fast forward or reverse the current movie, he will skip the chapter marks and their actions. This way other scenes than intended may be shown. A return by pressing the red button will still return to the stored resume time of the movie he started with.

Behind the scenes movie or just a chapter

A variation on the themes above is having a feature film and a “behind the scenes” movie on the same disc, both being offered to run independently.

⁵² Same reasoning as previous footnote applies.

The feature film has chapters and those can be selected through the film's popup menu connected to the menus generated by a chapter wizard.

The bonus movie can work exactly the same.

But you need a third option where the bonus movie chapters can be selected from the main menu without starting the bonus movie first. And when a chapter is selected, only that chapter is played.

The next discussion shows how to do this. In summary it involves:

- creating the chapter menus through one of the chapter menu wizards (discussed earlier in Chapter popup menu generation through wizards on page 166).
- Copy the menu as static picture on the main menu
- An arrow-down action on a main menu button opens the first of the generated menus (that looks like the one you made a static copy of).
- using a flag to know if you play a movie or a chapter only.

For the following discussion we use the second chapter wizard – it produces fewer menus. But you can use the other wizard also – it involves more work as all generated menus must be modified. The whole project can be found in the BDS kit under \Projects\Bts chaptersBR. You need to mux the project for a working disc.

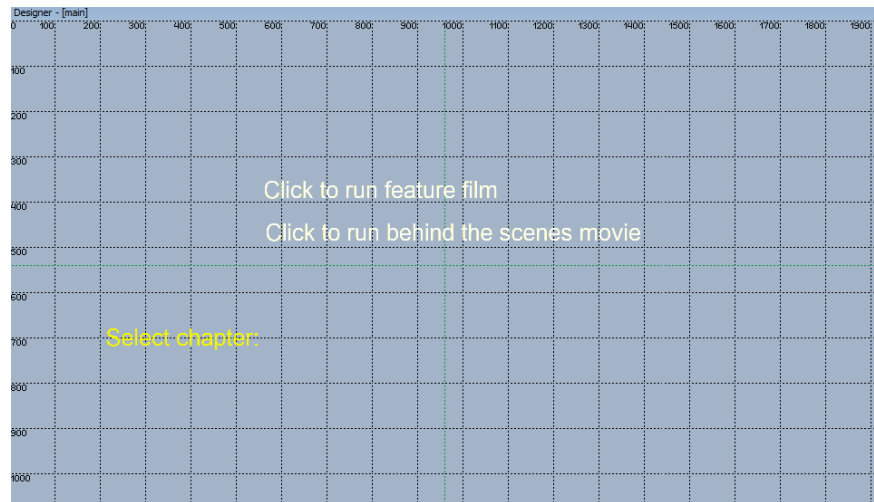
Because the generated chapter menus are modified only do the following steps if you are sure the chapters in the bonus movie are final. Otherwise you need to restart all work again at each re-generation.

Main menu

The main menu shows two buttons:

- start the feature film,
- start the behind the scenes movie

A very simplistic menu with two text buttons is shown below. Note it also has a plain text element that says "Select chapter". Below that text sufficient space should be allocated to paste a copy of the (to be generated) chapter menu into.



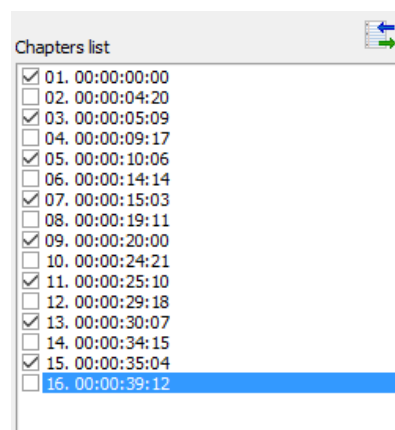
The Press Enter action of both feature film and behind the scenes movie text buttons are set to “play movie”.

A fake “button” is later to be added by assigning the down arrow of the behind the scenes button to open the first generated chapter menu. But first we must generate them.

Generating chapter menu

The generation of the chapter menu of a movie is done exactly the same way as for any movie. For this, you open the scenes window of the behind the scenes movie.

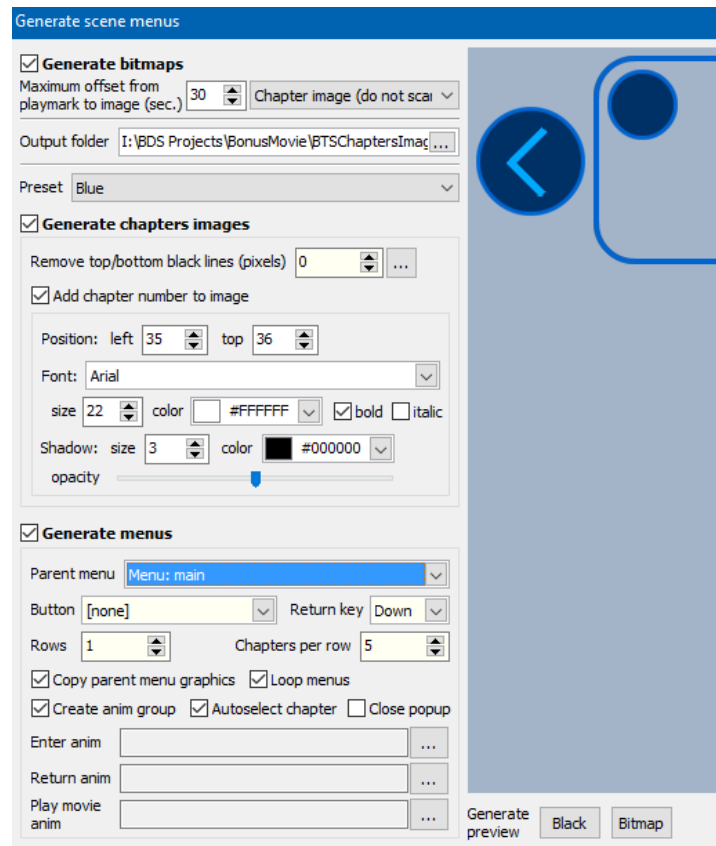
Specify the chapters at the points where a new topic in the movie starts. From this the chapter wizard will create the menus. You may add additional chapters but for those you do not want to include in the chapter wizard menu generation uncheck the box at the left of the chapter mark.



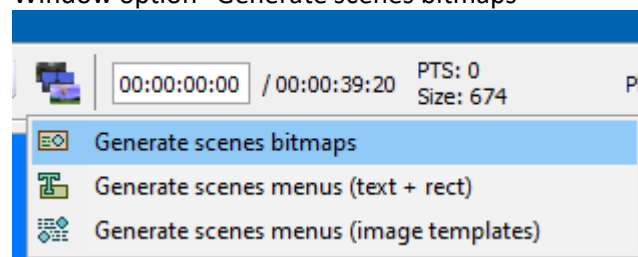
We need extra chapters at the end of each scene in the behind the scenes movie. These chapters are needed to stop playing the movie if we just want to select a single one. But for those chapters clearly no menu should be generated. These “end of scene” chapters are easily added. Simply select the chapter that starts the next scene. Then click on the “<<” button in the scenes window to find the I-frame before the start. That is the end of the previous scene. Add a chapter there.

We'll use the second wizard ("generate scenes menus (image templates)") here, but the other one works exactly the same way but generates more menus and therefore more modification.

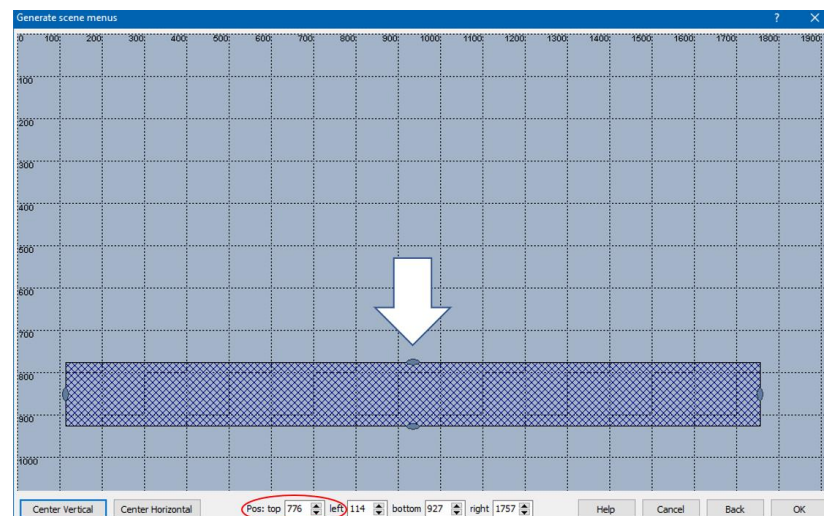
Don't just start the wizard but first modify a few settings:



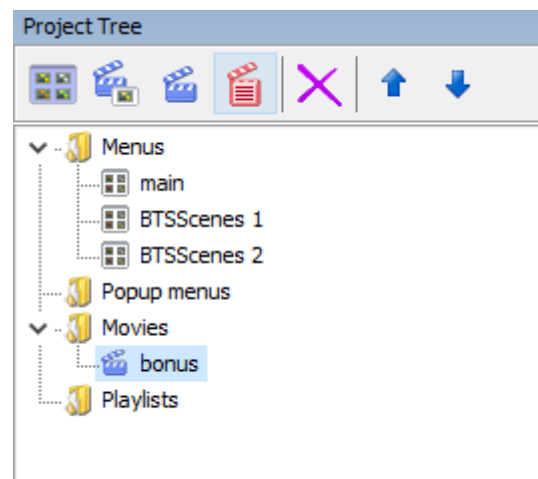
- Set the main menu as the parent menu and ensure a checkmark is placed in "copy parent menu graphics". This copies the static content of the main menu into every generated chapter menu. If you don't only the chapters will show on the screen.
- Give unique names to the folders for chapter images and the menus (we choose BTSChapterImages)
- Most important: in the top section "Generate bitmaps" set the scene detection algorithm to "chapter image (do not scan)". If you don't BDS will use its own detection mechanism and may provide an image that does not belong to the chapter you intend to show. You can generate images first yourself (then uncheck the "Generate bitmaps" and first run the Scenes Window option "Generate scenes bitmaps")



In the next wizard screen you position your band of chapter images. They should be positioned below the text “Select chapter” on the main menu (its bottom is around height 720 pixels – so go to 730 or higher)⁵³ Either use the mouse to drag the strip into position or specify its value in the “Page” boxes below the screen – notably the “Top” position.



When all is set and done, click on the “OK” button and let the menus be generated. They are entered in the “menu” branch of the project tree and named “scenes 1” and “scenes 2”. (these names are hardcoded, so if you intend to generate more menus, it’s advisable to rename them NOW, e.g. into BTSScenes 1 and BTSScenes 2).

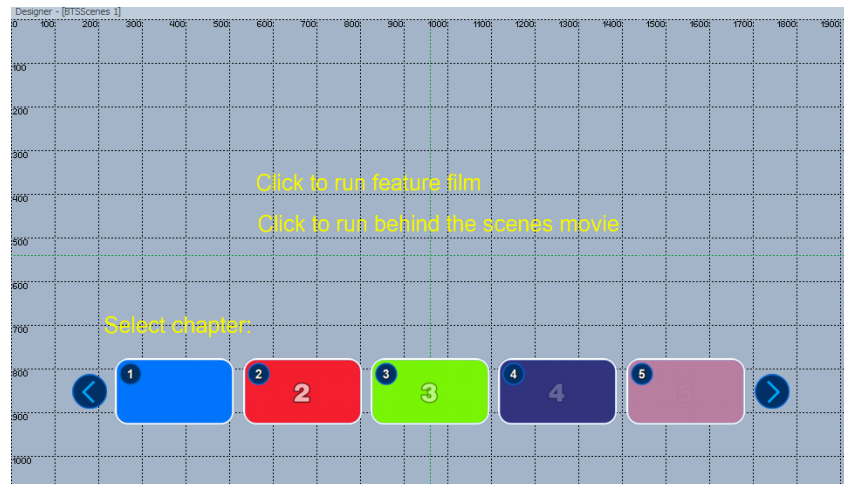


Opening one of the BTSScenes X menu shows whether the objects of the main menu are properly copied as static images.

The chapter images have now become the buttons on the menu. The buttons of the main menu have become plain text (using the colour of the “normal” state). For chapter 5 an additional action is generated for

⁵³ Unfortunately, menus entirely built from BDS text buttons do not show in the wizard preview. Menus with png image file texts do show.

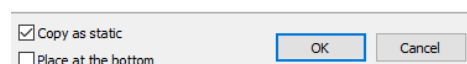
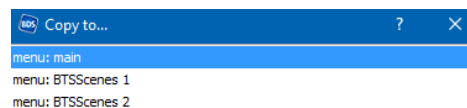
“Press Right” that opens the next scenes menu. The viewer will think he activated the “>” image (which is not a button).



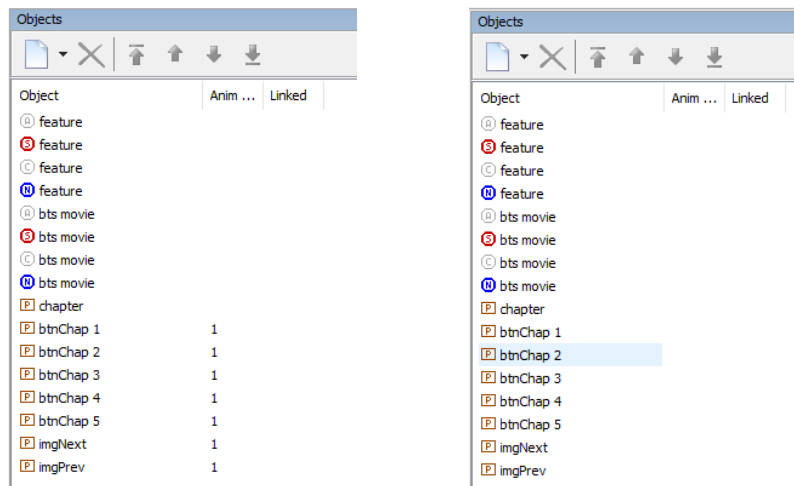
Copy the chapter images to the main menu

The main menu itself still hasn't got an image of the chapters yet: we need to copy this strip onto the main menu below the “Select Chapter” text at the exact same position (the copy operation does that).. This way when we switch from main menu to generated chapter menu, the viewer doesn't notice the difference.

On the “BTSScenes 1” menu, select the band with all chapter images (simply drag the mouse from left-top to right-bottom around the chapter images). Select and (right click) “Copy” (or press CTRL/C) and on the window that asks for the target, click on “main”. Ensure the checkbox “Copy as static” is checked.



The images are now part of the main menu but they have retained their animation group assignment. We don't use it, so you can ignore it or remove the group assignment (select them, right click > animation groups> remove checkmark before “Animation group 1”).



The main menu now looks like the generated menus except different buttons exist (white coloured as “selected” state) whilst others have become static images (yellow text or plain images).



Connect “chapter” menu to the main menu

Although the chapter images are visible on the main menu, they are not buttons and cannot be selected. To do that, we need to be on one of the generated chapter menus.

We introduce a “fake” button on the main menu. The down arrow navigation of the “bts movie” button becomes an action to open menu “BTSScenes 1” with the first chapter as selected button.

anim up	
↓ Press Down	Menu: BTSScenes 1 [btnChap 1]
anim down	
OK Press ENTER	Movie: bonus [0]
anim enter	

When the viewer thinks to navigate the main menu by moving from the “bts movie” button to a chapter button, he actually replaces the main menu by the “BTSScenes 1” menu where indeed the btnChap 1 button for chapter 1 is pre-selected (while all main menu buttons have now become pictures).

Connect “chapter” menus with main menu

Of course the viewer may decide not to select a chapter after all and return to the “Click to run behind the scenes movie”. That can only be done by using the “up arrow” from whatever chapter button is selected. That up arrow action silently reopens the main menu and pre-selects the “bts movie” button. For the btnChap 1 button the “Press UP” must be added to the other generated actions.

Name	btnChap 1
Selected state	Select.png
Effect	
Top	776
Left	238
Actions	
← Press Left	Menu: BTSScenes 2 [btnChap 8]
anim left	
→ Press Right	Button: btnChap 2
anim right	
↑ Press Up	Menu: main [bts movie]
anim up	
↓ Press Down	Menu: main
anim down	
OK Press ENTER	Movie: bonus [1]
anim enter	

The same must be done for all other chapter buttons on the BTSScenes 1 and BTSScenes 2 menu. Fortunately this is easily done via the Action Matrix display or simply by copying the action for Press UP from btnChap 1 and paste it to all other btnChap X buttons.

Flag to recognize “chapter play” from “movie play”

We’re almost set. We still need to distinguish between running a complete movie or simply playing a single chapter from the behind the scenes movie. Like we did in earlier projects, we can use a general register for this purpose. If GPR 200 contains a value of 1 we only want to play a single chapter. Any other value (usually 0) means we play the entire movie.

Because all “play movie” buttons are on the main menu and all “play chapter” buttons are on the generated BTSScenes X menus, it seems logical to add an action to the main menu that sets GPR 200=0. And on both chapter menus BTSScenes X an action that sets GPR200=1.

Properties	
Common	
Name	main
Draw method	Selected, Current, Normal
Intro Movie	[none]
Inactivity timeout	0
Inactivity action	
Action every second	
On update current	
Open sound	
JAR	00000
Allow save state	<input checked="" type="checkbox"/>
Streams	
Video	1:\BDS Projects\BonusMovie\film\black silent 5s 1920x1080 23.97 fps.264
Audio	1:\BDS Projects\BonusMovie\film\black silent 5s 1920x1080 23.97 fps.ac3
Remote control buttons	
Enter	
Action 1	[set]: GPR[200] = 0
Animation 2	

Properties	
Common	
Name	BTSScenes 2
Draw method	Selected, Current, Normal
Intro Movie	[none]
Inactivity timeout	0
Inactivity action	
Action every second	
On update current	
Open sound	
JAR	00000
Allow save state	<input checked="" type="checkbox"/>
Streams	
Video	1:\BDS Projects\BonusMovie\film\black silent 5s 1920x1080 23.97 fps.264
Audio	1:\BDS Projects\BonusMovie\film\black silent 5s 1920x1080 23.97 fps.ac3
Remote control buttons	
Enter	
Action 1	[set]: GPR[200] = 1
Animation 2	

The register is set by clicking on the action “>” button and then select “Settings and Properties” > “Set GPR/parameter”.

Now that the menus have an action, the opening of the menus must be changed to open the menu AND to run the action. This means you must revisit all buttons that (via Press UP or Press DOWN properties) open a menu: they should do so now with the action specified. In case of the main menu, the button “bts movie” needs to update its Press

DOWN action. After clicking on the action ">" button, select "Jump Menu" > "BTS Scenes 1" > "Enter anim/action 1" > "btnChap 1".

anim up	
↓ Press Down	Menu: BTSScenes 1 [btnChap 1] [anm/act 1]

For the BTSScenes X menu chapter buttons similar actions are specified. Specify action for one button, copy/paste the action to all other buttons).

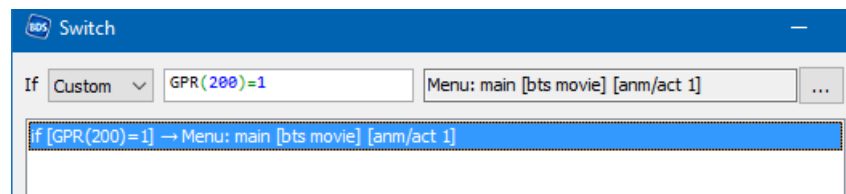
anim right	
↑ Press Up	Menu: main [bts movie] [anm/act 1]

Use flag to play a single chapter

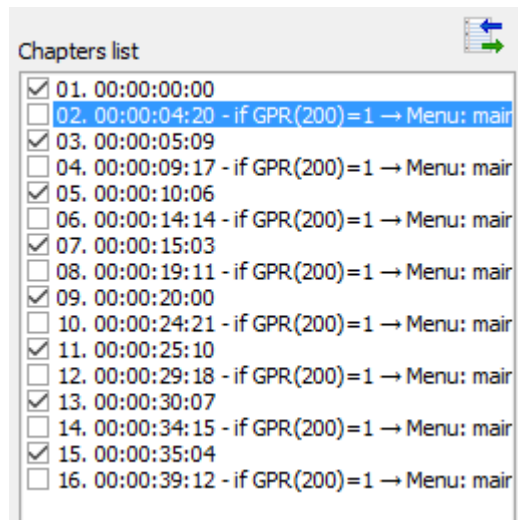
Now we know that GPR200=1 means play one chapter, we need to make BDS aware too. If a chapter button is activated it will start playing the behind the scenes movie from that chapter onwards. It will continue with all remaining chapters. Clearly this is not what is intended. So we need a condition at the end of each chapter of the behind the scenes movie that says "if GPR200 = 1 then return to the main menu and reset GPR200=0".

This is achieved by specifying chapters just one I-frame before the start of a new chapter. For those "end of chapter" marks a SWITCH statement is added.

Open the scenes window of the behind the scenes movie (called bonus in our example) by right click on its movie placeholder and select "Open scenes (direct editing)". This allows to add an action to a chapter mark. In our case, we add actions only on the "end of chapter" chapter marks (the ones without a checkmark in their box at the left).



If we entered the movie at chapter N then before starting chapter N+1 the switch condition is checked. Coming from a chapter menu button that menu has set GPR200=1 so the condition is met. At the end of the chapter the main menu is reopened, pre-selecting the "bts movie" button. But because the main menu has an action that sets GPR200=0 it must be executed: the [anm/act 1] action must perform when the menu opens.



You specify the switch for one chapter, use the (select, right click or CTRL/C and CTRL/V) copy/paste to copy the same switch action to all end-of-chapter chapter marks.

Part 6: MX only mostly (Slideshows and PIP)

Slideshows

Ordinary slideshows

There is nothing magic about a slideshow. All versions of BDS can add slideshows as variation of a movie. Several images are combined into a single movie where the video image is frozen for the duration of a slide. The duration can be set in the “time line” of the editor.

You use a commercial video editor such as Cyberlink’s Power Director, to create a movie from still images.

Video editors allow the addition of a music track to play while the slideshow is displayed. With some editors, such as Power Director, you can choose to use “Magic Music” – a collection of music tracks that are automatically extended or shortened to fit the duration of the slideshow.

The finally produced movie (which must be of BDAV type)) can be handled the same way as real movies. Each slide image can be given a chapter mark to allow the viewer to skip quickly through the slides.

Application of this type of slideshow is often a “Gallery” of images showing behind the scene impressions of how the main movie came about. It can also include pictures taken on holiday next to a holiday movie. But the duration of each image is set in the editing of the slideshow movie.

A special use of a slideshow is when it is used as a PIP (picture-in-picture) movie to show slides showing behind the scenes aspects of the playing movie.

BDS slideshows

Apart from ordinary slideshows that are simply movies with still images, BDS supports two additional types of slideshows in BDS MX. These slideshows are a (changing image) menu or a movie of its own:

- Browsable slideshow (1 ES – elementary stream – only if Rovi Scenarist)is used) – used for galleries of pictures, each image is shown for as long as the viewer wants it shown. In fact, each image is displayed for a specified duration (like time-based) but then it is displayed again until the viewer presses “next” or “previous”
- Time-based slideshow in menus – background menu movie based on a set of slides

Both types are created through a special encoder that takes still images and transforms them into a .264 movie (without sound). The images in the .264 movie are stored as individual images, regardless of the display time specified. In this respect the .264 produced file is different from the “ordinary slideshows” produced by video editor tools as it is much more compact (1 frame image instead of 720 frames for 30 seconds at 24 fps).

BDS allows you to create a playlist of several slide shows and have a continuous audio (music) track.

As an example: a menu movie made with the x264 encoder consisting of 13 still images each displayed for 5 s at 10,000 kbps (10 Mbps) bitrate, requires 10 MB on disc. Less even (8 MB) if all images are manually scaled and cropped to the required resolution (e.g. 1920x1080) beforehand.

Creating the same menu movie of 13 scaled and cropped images each shown for 5 s and 10 Mbps bitrate as an ordinary movie made with Power Director results in a 62 MB file – about 7 times larger (since there are 5 s x 24 = 120 frames (of I and B type) with identical image information as opposed to a single one).

The x264 encoder must be installed separately as it is not part of the BDS installation.

X264 encoder

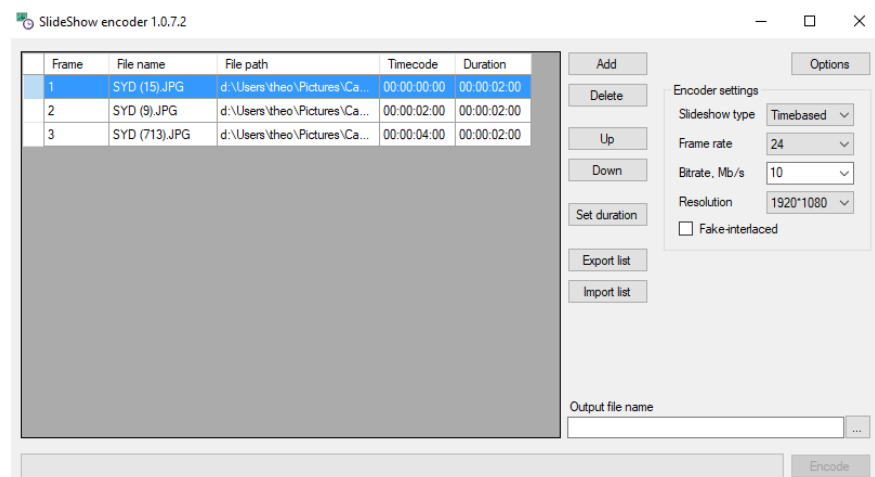
The x264 encoder is a freeware project created by VideoLan. It consists of a single .exe file that needs to be copied into a known folder. The complete file specification of the encoder must be provided to BDS through the Tools > Options > Compile tab in the section “x264 encoder”.

Any location for the encoder will do (such as a folder within \Program Files or in a folder under the BDS installation folder (such as C:\Program Files\Blu-Disc Studio MX). As long as BDS knows where to find it.

Create a slideshow with the encoder

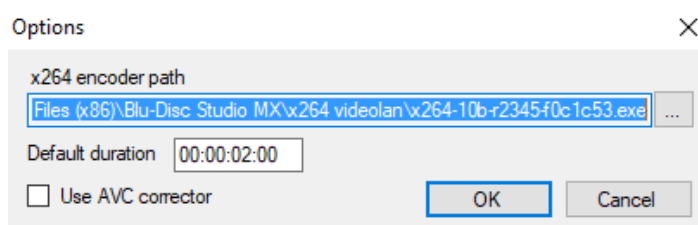
Because the slide encoder is an independent program, it is started as external program from within BDS MX through the option Tools > Slideshow Encoder.

If the location is properly specified in Tools > Options > Compile tab, the encoder opens a special window in which all slides are specified.



By pressing “Add” you navigate to the images you want to include as slides. Unlike BDS’s limited choice of .png only, slide images can be of types .png, .bmp and .jpg which may save you a lot of effort in resaving files in the .png format.

The “Options” button allows to specify the encoder location (already done if invoked from BDS MX) but also a default duration for each slide. The “AVC Corrector” is not recommended for use in BDS.

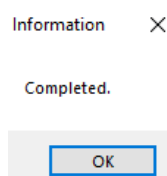


The bottom contains an important field: the output file name. You specify what the name of the slideshow file is going to be. You can specify the foldername of \films of your project as well as the filename itself. During the encoding process the .264 movie file is created, alongside a .spi file that contains info in how long each frame should be displayed.

The right-hand side of the window allows you to specify whether:

- the slideshow is (Type=) “time-based” or “browsable”
- what (BDA allowed) frame rate to use (if most or all of the BDS disc uses a certain framerate, it is logical to use that frame rate here too)
- the bitrate for the movie. More bits, more detailed slides
- resolution (BDA approved) of 1920x1080, 1440x1080 or 720x1280
- change the order of the slides (up/down)
- the duration of slide display (different from the set default)
- export to or import from an XML-based list of slides. (Useful if you may need to revisit the slideshow after spending a long time creating it).

When all is set and done, create the .264 output slideshow movie by clicking on “Encode”. Its completion is indicated with a small “completed” confirmation.



Important: Once the slideshow is created, you **must** close the encoder by clicking on its “X” button in the top corner at the right. Not doing so may cause another instance to open when you select Tools > Slideshow encoder. This second copy may produce fake error messages if you are to encode the same images once again (with different duration or other settings).

The .spi file is a text file you can open with Notepad. It contains display duration information as well as how to display an image..

Its structure is very simple. As an example, below a slideshow of 3 slides, shown at 24 fps for 30 seconds each reads:

```
TS
720
720
719
1
```

Here TS = time-based (as opposed to BS = browsable), followed by the display time in number of frames except for the last one that is one frame shorter since it is required for each movie to have 1 stopframe – which is also the last image, shown for another 1 frame (in total still 719+1=720 frames). The BS browsable types also have the display time in number of frames. But rather than moving to the next image it restarts displaying the same image again in an endless loop.

Important considerations

Using slideshows in bluray discs does not seem to be without risks.

When using the encoder with a set of slide images, keep in mind that:

- images are resized to the 16:9 aspect ratio at the specified resolution for the encoder (e.g. 1920x1080 or 1280x720).
- Resized images are scaled in their entirety. The whole image is kept, but the largest side is scaled to fit the resolution, the other side is scaled proportionally. This may result in black bars at two sides of the image (either horizontal or vertical).
- Portrait images are turned sideways (90 degrees clockwise) to fit the landscape widescreen aspect ratio. The largest side is scaled to fit the resolution, the other side scaled proportionally. This may result in black bars at two sides of the image (either horizontal or vertical).
- If you want full control, resize all images before use into the required resolution. This may result in cropping the image to fit the 16:9 aspect ratio at the specified resolution.
- Most software BR players don't handle slideshows well. They either show nothing (black screen instead of slides) or in rapid succession.
- Some hardware set top boxes appear also not to handle slideshows well (producing colour streaks on slides or hanging).
- The slideshow files in the \output\bdmv\stream\000*.m2ts files do not show properly in several software movie players or show slides in rapid succession (ignoring time bases)

XML import/export format

The XML structure is quickly understood when an export file is opened. The basic structure is (timings are given in hh:mm:ss:dd *hours:minutes:seconds:hundreds_of_seconds* format – 2 ½ seconds is 00:00:02:50):

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<ArrayOfFileItem
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <FileItem>
    <num>1</num>
    <fileName>filename.type</FileName>
    <filePath>full_filespec_of_image</filePath>
    <time>start_time_in_hh:mm:ss:dd</time>
    <duration>display_duration_in_hh:mm:ss:dd</duration>
  </FileItem>

  ...

</ArrayOfFileItem>

```

The <time> tag specifies the start time of an image from the start of playing the slideshow. With all durations equal, this means that for the N-th image its <time> is equal to $N \times \text{duration}$ seconds. In the XML file, this time is an increasing row of values from 0:0:0:0 to the end time of the slideshow. The format is the same regardless whether the images are time-based or browsable.

You can create your own slideshow file as export/import xml file (or have it generated). This may cut down significantly on the creation time. At the same time, storing the information saves you a lot of time if later you decide to change the slideshow by adding, removing or reshuffling slides as the resulting .264 movie file has no knowledge of how it got composed. You may want to save these export/import files in the \original sources folder of your project.

Time-based slideshow in menus

Purpose of a time-based slideshow in menus is to have a background movie that consists of still images that rotate on a time-based schedule. It is no different from an ordinary menu movie except that the movie is much smaller in size. But there is no visible difference to the viewer.

The menu itself is also no different to any other menu: it must have buttons as objects because there must be a way to exit the menu through a “Press Enter” action of the selected button – either to go to another menu or start a movie to play.

Time-based slideshow in movies

The use of time-based slideshows in movies is effectively the same as creating an ordinary slideshow through a video editor. The slide images, their order and the display duration are all set in advance before the encoder creates the .264 video file and the .spi file with timings.

The size of the movie made by the encoder is much smaller.

Steps to take to create a browsable slideshow:

1. Take all relevant slide images and create a time-based .264 video file (a .spi file is also created)
2. Create a movie object and specify the slideshow .264 movie as movie.

3. Create a button on a menu that links to the playback of the movie

Browsable slideshow in movies

Purpose of a browsable slideshow in movies is to provide story board images or other extras, shown via a popup menu. The movie of the popup is the slideshow movie. The viewer determines how long an image is displayed. Each image is its own chapter in the movie and the viewer must move from chapter to chapter.

If you rely entirely on remote control buttons without a visual (popup) menu aid, you can also simply start the browsable slideshow movie as any movie and use the “next chapter” and “previous chapter” remote control buttons to jump from image to image.

This viewer-driven slideshow is different from the “ready-made” slideshow of a video editor where the amount of time each image is displayed is fixed during the creation of the slideshow. The “ready-made” version can have an audio track as you know how long it will run.

Steps to take to create a browsable slideshow:

1. Take all relevant slide images and create a browsable .264 video file (a .spi file is also created)
2. Create a movie object and specify the slideshow .264 movie as movie.
3. Create a button on a menu that links to the playback of the movie
4. Create a popup menu for this movie with buttons that will provide actions
 - a. Previous slide
 - b. Back to menu
 - c. Next slide
5. In the movie placeholder properties, check the “Auto-show popup” checkbox.

Steps 4 and 5 can be omitted if the slideshow movie is activated as any movie. Its images are shown one by one and the user needs to press “next chapter” or “previous chapter” on his remote control.

The popup can keep the selected button (next, next, next may step through the slides) as selected or can be a Multi-action that first performs the “next chapter” (or “previous chapter”) action (which changes the slide shown) and then jumps to the “menu” button. That means that pressing the “OK” button while the “next” button is selected will give the next slide. Press “OK” again and you move back to the movie since “menu” is now the selected button. To show the next slide, you need to navigate to the “next” button first.

Project 15: Create a BDS MX slideshow

Project Goal

We will reuse the simple MenuBR project (2) to replace the “menu” movie by a time-based slideshow of Sydney-based images. We also add a third movie that is a browsable slideshow of coastal images. That can be invoked from an added menu item.

This project will only work with the BDS MX version as muxing the slideshow files into the disc image files can only be done by the internal muxer that understands the .spi files with timing information. The tsMuxer was never envisioned to do this.

Project steps to take

Ready to run

Steps to take:

1. Copy \Projects\SlideshowBR folder to your project tree
2. Copy \Sources\movies\demuxed\Australia*.* and Coasts*.* to your project's \SlideshowBR\films folder
3. Open the \SlideshowBR\project.bdmd file to start BDS and open the project
4. Mux the project using the internal muxer

Build your own

Take the following steps:

1. Create the \SlideshowBR project by copying the \MenuBR project
2. Copy \Sources\project objects\slides\slides coasts resized folder as a subfolder to \SlideshowBR\original sources
3. Do the same for the folder \Sources\project objects\slides\slides 381oolea resized
4. Open the x264 encoder (Tools > Slideshow encoder). Click on “Options” and set 5 seconds as default display time
5. Click on “Add”, navigate to \SlideshowBR\original sources\slides Sydney. Select all files and wit “OK” add them to the encoder list
6. Specify 1920x1080 resolution, 23.97 fps, 10 Mbps, type “time-based”
7. Specify output file as \SlideshowBR\films\slides Sydney ts.264
8. Encode the file
9. Optional: Save the setting by exporting the slideshow data
10. Select all slide images and delete them from the encoder list
11. Click on “Add”, navigate to \SlideshowBR\original sources\slides coasts. Select all files and wit “OK” add them to the encoder list
12. Specify 1920x1080 resolution, 23.97 fps, 10 Mbps, type “browsable”
13. Specify output file as \SlideshowBR\films\slides coasts bs.264

14. Encode the file
15. Optional: Save the setting by exporting the slideshow data
16. Close the encoder (click its "X" button)

17. Copy \Sources\project objects\original sources\slideshow*.psd menu files to \SlideshowBR\original sources
18. Import the "slideshow popup.psd" file as popup menu
19. Replace the menu movie of menu "main menu" by SlideshowBR\films\slides Sydney ts.264
20. Create a new movie placeholder:
 - a. Name "coastal impressions"
 - b. Video stream \SlideshowBR\films\slides coasts bs.264
 - c. End Action = jump to main menu
 - d. Popup menu = slideshow popup, default "next" button
 - e. Check its "Auto show popup" box

21. Import the "slideshow main menu.psd" as menu in the project Tree view. We're not going to use this menu: we just need a text part from it
22. Copy (replace) the item "main menu text" from the folder \slideshow main menu to \main menu (we need the extra menu item in the main menu). The menu must now show an extra option "Coastal slideshow"
23. Delete the "slideshow main menu" menu (you may also delete the "\SlideshowBR\slideshow main menu" folder)

24. Add a button in "main menu" for the "Coastal slideshow" text.
25. Update the navigation between buttons on the menu.
26. The new button's "Press Enter" starts the play of movie "Coastal impressions"

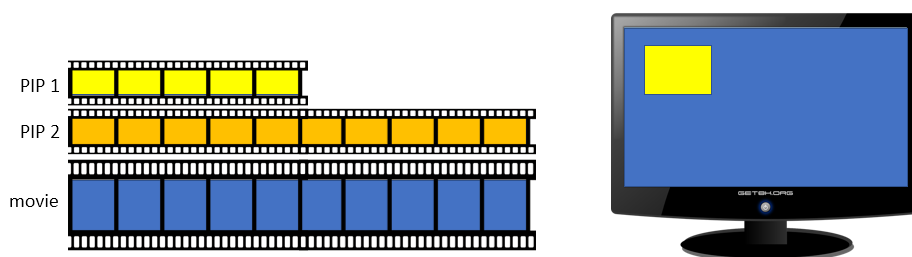
27. Set the First Play to "main menu", default "coasts" (this should already be so)
28. Set the "Top Menu" option to "main menu" (if Top Menu is activated via Project > JAR settings and edit 0000 JAR with checking the "Allow Top Menu call")
29. Mux the project
30. You may need to burn it on a BR-RW disc to check on the results as no BR software player seems to completely correctly handle the resulting disc files.

Picture in Picture

Bluray players with profile 1.1 or higher (which is almost all players these days) must support secondary video, called Picture-in-Picture (PIP). Both pictures can be shown at the same time.

Usually, the main movie is shown and in a smaller area an additional commentary or behind the scenes type of movie plays in sync. The PIP movies do not have to be the same duration as the main movie.

To show two movies at the same time requires special effort by the muxer – something tsMuxer was never designed to do. Therefore, this feature is only available to users of the BDS MX version that uses its own muxer.



The basic principle of a PIP is simple: a (main) movie placeholder has a property PIP that can hold up to four different PIP movies. Only one PIP movie can be shown at any one time. In the figure above, a movie is shown that has two PIP movies. On the tv screen shown at the right, the movie plays and the viewer has selected the shorter PIP 1 movie to play in sync.

PIP movies are not shown by default (unless programmed like that). They must be activated, usually via the popup menu of the movie or by pressing a remote-control button while the movie plays.

PIP movies and main movie are locked to the same time clock. They both play in sync. When after five minutes of movie play time, the PIP movie is made visible, it also shows its contents at five minutes into the PIP movie.

Note: The CMF builder of BDS MX Pro does not support PIP. You can use PIP, but the build process is then identical to BDS MX.

PIP movies are shorter or equally long as the main movie itself. If it is longer, the PIP movie is truncated to the length of the main movie. An odd thing happens with truncated movies:

- If the PIP reaches its end while visible: the last frame freezes for the rest of the duration of the main movie. It disappears when PIP=off, but shows again when PIP=on
- If the PIP reaches its end while not visible: it will not show in the rest of the duration of the main movie. It stays invisible regardless of PIP=on or PIP=off. It will show when PIP=on and you move to a time during which it can play.

In case you only have something to show at 30 s and 5 minutes into the movie and nowhere else, you do need a single PIP movie that has interesting parts at 30 s and at 5 minutes and nothing of interest (blank) in between. You can program BDS actions to switch on and off the showing of the PIP movie and show only the relevant parts.

Slideshows (with various display time of each slide) may well suit as PIP movie if the slides show aspects of the production of the movie (behind the scenes).

PIP and main movie

Each movie can have one to four PIP movies. Each of these movies can have 8 audio tracks. If there is more than one movie, either the program selects a specific one or the viewer selects one from a menu of choices.

The settings and condition checks described below are made In Java. However, keep in mind that PIP movies are muxed into/with the main movies. Just replacing the JAR file after code changes may not always be sufficient.

PIP video

There are some limitations set to PIP movies, imposed by BDA standards. See section “Number of BDAV/BDMV elements

There are maximum numbers of allowed audio streams and subtitles: 32 video and audio streams each, 255 subtitle streams.

Number of angles	9
Secondary video streams	32
Audio streams	32
Secondary audio streams	32
Text subtitle stream	255
Popup menus	32
Chapters (playlist marks)	255
Playlists	999
Play items in a playlist	255
JAR files	Limited by disc space
Fonts	255
Menu sounds	128

Supported primary and secondary video streams” on page 473.

Note that especially the frame rate of primary and secondary movies may differ.

Players may also (but need not) support full HD (1920x1080) for secondary video. BDS does not.

PIP Audio

BDA has a very odd restriction on secondary audio. Only Dolby Digital Plus and DTS-HD are specified, which is a very small subset of what the

primary audio stream can be. Especially the omission of plain Dolby Digital (.ac3) so common in many bluray projects, is inexplicable. Set top players however will also only support those audio streams for secondary audio. See section “Supported primary and secondary audio streams” on page 474.

BDS supports both the BDA approved streams of Dolby Digital Plus and DTS-HD. There do not seem to be many tools to convert from Dolby Digital (DD) to higher sound quality DTS-HD. “Upgrading” .ac3 to DTS-HD makes no sense quality-wise. But if BDA enforces the use of DTS-HD, you’re screwed. Same is true for DD to DD+. Nothing to gain, everything to lose, but without DD+ you’re screwed again.

PIP allowed combinations

The tables in Appendix C: Bluray specifications on page 469 lead to the following observations and additional BDS limitations:

- Video resolution can be less or equal to primary video resolution, except for HD 1920x1080.
Supported audio streams for the PIP movie are only Dolby Digital Plus and DTS-HD. When heard, both main audio (subdued) and secondary audio are heard.
- PIP movies can be without audio. They then display silently in a window inside the regular movie. This is often the case.
- PIP movies have none of the properties regular movies have. No chapters, no buttons, nothing. Anything you want to control needs to be done via the regular movie to which the PIP movie belongs.

The PIP movie is not shown in the list of movies in the Project Tree. The main movie is a regular movie in the “movies” section of the Project Tree. Like ordinary movies, the PIP movie must have been demuxed into its video and audio streams, all of which are compliant with the BDA approved formats.

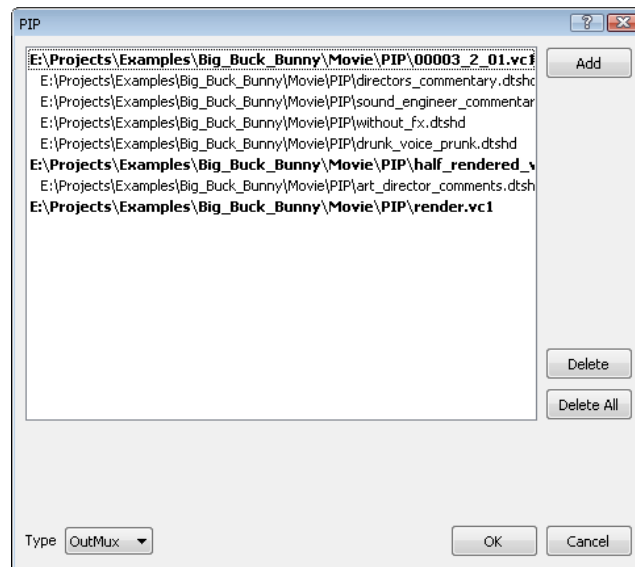
If an SD movie (MPEG-2 stream) is demuxed by tsMuxer, it produces a .mpv video stream. You need to rename it to .m2v (the stream is the same, just the file type different). Only then BDS MX will show it in the list of PIP movies to include.

To associate a PIP movie with a regular movie, select its movie placeholder and fill in its PIP property. This is done by clicking on the “...” button at the far right of the PIP property.

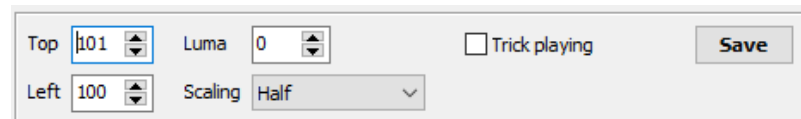


A window then opens that allows you to “Add” PIP movies. Video and audio must be selected one after the other. The audio tracks that belong to a movie are shown indented.

You specify the movie PIP files. By selecting one of them, you can add the audio tracks of that movie. These are shown indented to the movie name. There is a maximum of 4 PIP video streams. Each video stream can have upto 8 audio streams (of DD+ or DTS-HD format).

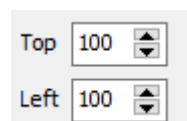


When you select a PIP video stream, additional settings can be specified for the PIP movie to show or merge with its main movie.



PIP position

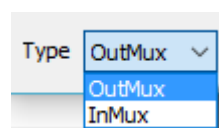
When you select any one of the PIP movies, the bottom part of the window provides ways to specify the PIP movie position within the main movie screen by specifying the left-hand top corner. The size is set by one of the five possible scale factors. The “One and half” is actually an increase in size, useful only for PIP movies with limited resolution like SD movies (DVD sized 720 x 480 pixels) or smaller.



Each PIP movie can be positioned independently provided you click the “Save” button before setting the position of another PIP movie. Without saving the position, it remains at the old values, most likely (0,0): the top left-hand corner.

PIP muxing

PIP movies can have one of three scenarios of getting muxed (multiplexed). You make your choice in the PIP window in the “Type” dropdown list.



- In Mux synchronous
PIP and main movie use the same time clock. They are muxed

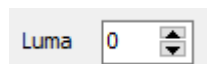
into a single stream. Typical use: director commentary (talking head – otherwise use it as audio stream to main movie)

- Out Mux:
Both streams are not muxed into a single stream. The PIP stream doesn't have to reside on the disc and could be loaded from the internet via BD-Live. BDS does not support this: the stream is available on disc. There are two types of out muxes:
 - Synchronous – both streams use the same time clock. This is what BDS creates. Both streams are on disc.
 - Asynchronous – PIP stream is not tied to the primary video stream. Both streams use different clocks and the viewer can start a PIP at will and at any position. Not supported by BDS.

As far as BDS goes, there is not much difference between Inmux and Outmux. The former produces a single .m2ts file with both video streams, the latter two .m2ts files with one stream each. There does seem to be a difference when multiple PIP files are used: then OutMux seems preferred for flawless working of BDS produced discs.

PIP Luma key

In rare occasions you may change the luminance (luma) of the PIP movie. You define an upper value for the luminance channel (Y). This number is entered in the PIP window for any of the PIP movies. By default it is set to 0.

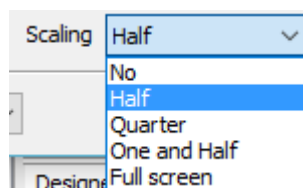


All pixels of the PIP movie with a luminance less than the set value are rendered transparent. This can blend the PIP movie into the main movie rather seamlessly. Obviously, this only works well if the PIP movie is not scaled to full screen. When you use luma keying, you should disable the use of “PIP Full Screen”.

PIP scaling

The size of the PIP movie is its own “native” resolution. But you can specify it to show at a quarter of its height and width, half its height and width, one and a half its size or even full screen. Please note that the scaling refers to the length of the movie's width and height, not to its surface area! At scaling $\frac{1}{2}$ the rescaled movie only takes up $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$ of the screen. At scaling $\frac{1}{4}$ the area is $\frac{1}{16}^{\text{th}}$ of the screen area.

Obviously only scaling settings that make the PIP movie fit on the screen are useful.



BDS indicators and activators for PIP

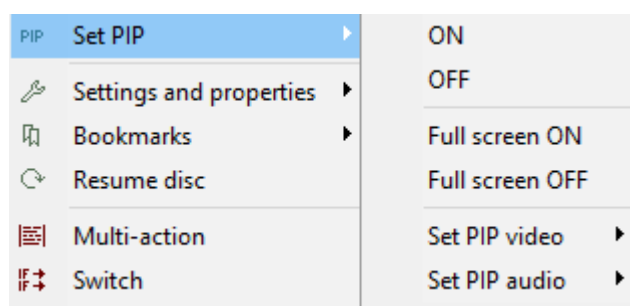
There are several actions that BDS can take and conditions it checks to start or stop the PIP movies. Because in most cases more than one action is required, the PIP actions and conditions usually occur in Multi-action or Switch actions.

(De)Activate PIP

First of all, there is the indicator that the viewer wants PIP to show or not. This indicator can be used in a Switch action to perform different actions depending on whether PIP is active or not.

The PIP enabling setting is achieved via the action

- Set PIP > ON
- Set PIP > OFF



Such action can be specified in a (switch) action, most likely for a button on the remote control. For both the “movie” as well as its popup menu, the yellow button can be made into a toggle switch (enable if disabled disable if enabled).

```
if [isPIPOn] → Set PIP: off  
if [isPIPoff] → Set PIP: on
```

Show PIP movie full screen or part-screen

There is a possibility to show the PIP movie full screen (covering the main movie itself). Or at the size specified in the PIP file window as set as main movie PIP property.

This is achieved through the action

- Set PIP > Full screen ON
- Set PIP > Full screen OFF

The setting only works of course if the PIP movie is on display, i.e. PIP=on.

Select what PIP movie to show

If a single PIP movie is specified for a regular movie, that PIP movie is automatically selected for playback. This is the normal case.

When there is more than one PIP movie associated with a regular movie, either the viewer (by specific remote-control buttons or popup menu items) or the application decides which PIP movie to show with what audio track. The application may also decide at what point in the

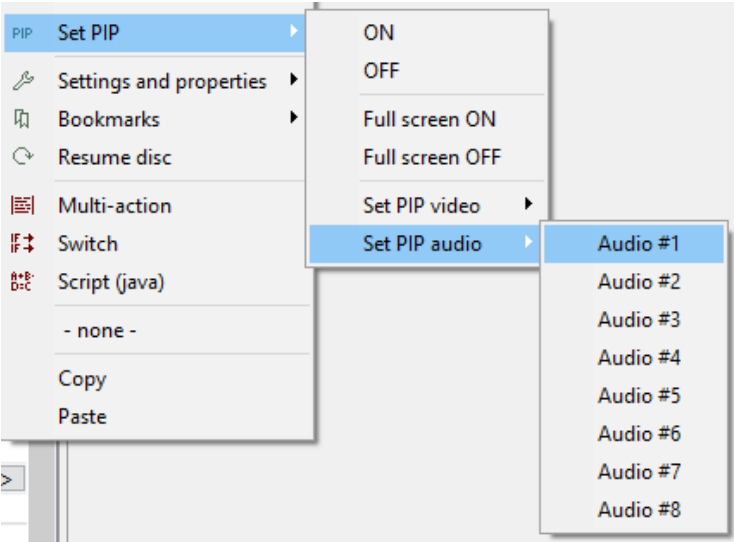
movie the PIP movie is shown. If the viewer may not decide between movies, obviously he must not be presented a way (such as a popup menu item) to do so.

The selection is set via

- Set PIP > Set PIP video
- Set PIP > Set PIP audio

Experience shows that selecting a PIP movie only works well if the muxing of the PIP movies is done as Type = “OutMux”.

One of four movies can be selected. Just like movie audio or subtitle tracks, you should only specify a track that actually exists.



The numbers correspond with the position of the movie track (or the audio tracks for a movie) in the PIP window.

BDS PIP condition status checks

With the various on/off settings the BDS application may decide to perform different actions. It is logical to use a Switch action to check on the PIP setting. There are six conditions that can be used in the “Custom” selected Switch:

PIP	boolean isPIPOn
Bookmarks	boolean isPIPOff
boolean CanDiscResume	boolean isPIPFSon - is PIP full screen on
boolean HasDataInStorage	boolean isPIPFSoff - is PIP full screen off
Operators	int PIPVideo
	int PIPAudio

PIP (de)active

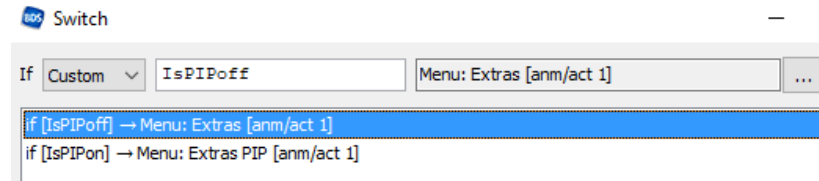
In some situations, it is required to know whether PIP has been activated or not.

There are two conditions for clarity although one would have sufficed as negating (by using exclamation mark “!”) it automatically gives the other (!IsPIPOn = IsPIPOff). Both provide either a true or false result:

- IsPIPoff
- IsPIPon

Possible use: [change of menu](#)

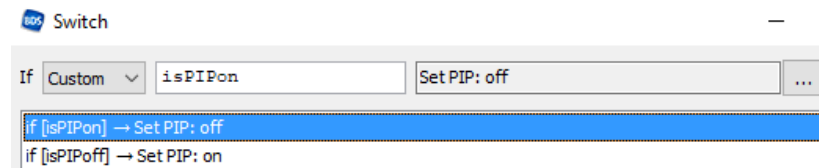
An example Switch action that displays different menus depending on whether the PIP is activated or not, is shown below.



The menu may have different options when PIP is activated or not.

Possible use: [toggle PIP on/off](#)

You can set an action for a menu button or a movie button as a switch that functions as a toggle action to either enable or disable the display of a PIP movie.



Possible use: [show PIP only when relevant](#)

The condition can be used within the regular movie to temporarily disable a PIP movie because it only contains useful information at certain times. Those times must then be marked by a chapter mark at the start and the end of the interesting part.

There are then two conditions that must be met:

- The PIP movie is only shown when PIP=on
- The movie is only shown during relevant time intervals that are surrounded by two chapter marks

This can be achieved by using two flags – e.g. by using GPR registers:

- If PIP can be shown, GPR(10)=1, otherwise not
- If the movie is in a relevant interval, GPR(11)=1
- The PIP movie is only shown when both GPR(10)=1 AND GPR(11)=1. Then the action is to Set PIP on, otherwise it is off.

To show or not show a PIP movie is a toggle by some remote-control button like we discussed before. Rather than directly setting SET PIP ON it only Set GPR(10)=1 or 0.

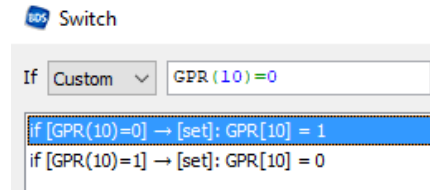
The start of a relevant interval needs to set the other flag. This requires an action to be performed on a chapter mark.

Chapter actions are explained in section “Method 4: movie chapter action” on page 304.

If a relevant interval starts at chapter 4 and ends at chapter 6, the chapter 4 action is simply “Set GPR(11)=1” and chapter 6 undoes this

with “Set GPR(11)=0”. The same undoing is set for the “popup menu” and “End Action” actions of the movie to ensure the movie leaves no unwanted display flags set. Leaving GPR(10) to its value allows to keep that setting over several movies if wanted.

The yellow button (toggle) switch action would look like this:



Then the chapter actions are also switches that turn the PIP on or off but only if a PIP can be displayed (that is, GPR(10)=1).

At chapter 4 the PIP movie is shown and as soon as chapter 6 begins, it is reset

```

03. 00:01:38:20
04. 00:03:06:13 - [set]: GPR[11] = 1
05. 00:03:06:22
06. 00:05:32:20 - [set]: GPR[11] = 0

```

The check whether both registers have a “1” value and the PIP movie can be shown is done in the “Action Every Second” of the movie. The PIP movie is shown when the PIP status is set to “on”. It requires an exclusive switch that checks for the display condition and if it is not met, executes the last action which resets the PIP status:

```

if [GPR(10)=1 & GPR(11)=1 & isPIPOff] → Set PIP: on
if [isPIPOn & (GPR(10)=0 | GPR(11)=0)] → Set PIP: off

```

The addition of check on isPIPOff makes sure we do not switch PIP on when it is already on. At the same time, if either or both registers GRP(10) and GPR(11) are zero, showing should stop: set PIP=off but only if it is not already off.

PIP Full screen

You can check if the current setting for the PIP movie is display at full screen or specified resolution. Again, there are two conditions that can be checked:

- IsPIPFson – full screen showing is set
- IsPIPFsoff – resolution specified in PIP window is set

PIP movie playing

Two more variables, returning an integer value, can be used in conditions where it is required to know which PIP movie and which audio track are currently playing:

- PIPVideo – returns a value of 1,2,3 or 4
- PIPaudio – returns a value of 1,2,3,4,5,6,7 or 8

An example Switch check for video #1 is made and if so, audio track #3 is selected.

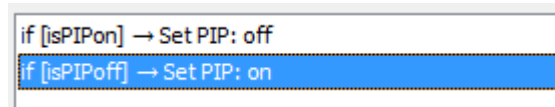


Display PIP movie

A menu item may list the regular movie. When selected, the “Press ENTER” action starts the playback of the movie. This movie has a PIP associated with it. It does not show.

A remote-control button (e.g. Yellow) defined on the movie object may start the PIP movie. Or, when pressed again, stop it. This requires a Switch action that checks whether:

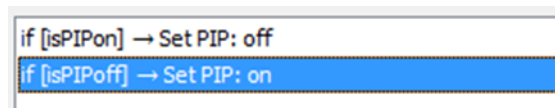
- IsPIPOff is set – then set PIP On and thereby play PIP movie
- IsPIPOn is set – then set PIP Off and thereby stop PIP movie



That’s all there is to it in the simplest of cases when only one PIP movie is specified.

Display PIP movie via popup menu

Rather than (or in addition to) displaying a PIP movie through a movie programmed button, you can also define a button on the movie’s popup menu that sets the PIP status to on or off. The action is similar to the switch statement of the movie button.



Project 16: Create a BDS MX picture-in-picture (simple)

Project Goal

The project reuses the simple MenuBR project that displays a single menu. When the “Sydney” choice is made, the “Australia” movie plays. For this movie the yellow button has been defined as a toggle to switch the associated PIP movie on or off. The PIP movie is a simple movie that shows a counting clock for the duration of the main movie. This illustrates that the PIP movie and main movie are muxed synchronously and use the same time clock.

Project steps to take

Rather than giving step by step instructions, we limit ourselves to the construction of a ready to run project. Its settings should give sufficient clues on how to create such a project by yourself if you have come this far into the user’s guide

Ready to Run

Take the following steps:

1. Copy the \Projects\PIPBR folder into your local BDS project tree
2. Copy the files \Sources\movies\demuxed\Australia*.*, Coasts*.*, menu.* and the following three PIP files from the same folder into the \films folder of your new project:
 - classical clock 30 seconds pal.m2v
 - digital clock 1 minute pal.m2v
 - Count up timer 2 minutes pal.m2v
3. Double click on one of the project.bdmd files to open one of the PIP projects
 - a. Project 1 only defines the yellow button as toggle switch to display the PIP movie (just as long as the movie itself) or not
 - b. Project 2 is like Project 1 but shows the PIP movie only during periods there is something to show.
 - a. Project 3 shows how to select between three PIP movies associated with the main movie. They are of different lengths (30 seconds, 1 minute, 2:08 minutes)
 - b. Project 4 solves the caveat of Project 2 by using java code that is run each second to see if the movie is inside an interesting time interval. The Java solution required is explained in more detail below. Output goes into \Output4.
4. Mux the project using the BDS MX internal muxer. Project 1 writes into folder \Output1, project 2 into folder \Output2, Project 3 into \Output3 and Project 4 into \Output4.

Project 1: simple on/off

The “1 PIP via movie yellow button.bdmd” project file shows the simple use of showing or not showing the PIP movie connected to the Australia movie.

The yellow button on the remote-control has been defined as a toggle switch to set PIP either to true (show) or false (hide). This is accomplished by using an exclusive Switch action on the yellow button.

Project 2: programmatically switch PIP on/off

The “2 PIP display at 10-30-50s for 10s yellow Pip on-off.bdmd” project uses a single PIP file with the same duration as de movie Australia. To show the viewer the PIP file only during those time intervals that it contains something interesting, the disc actively controls the PIP on/off status.

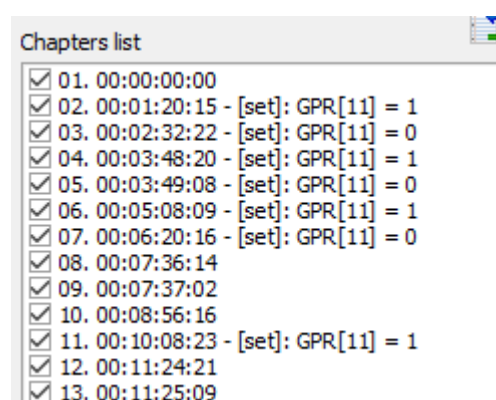
It does this by setting PIP off in uninteresting intervals and set it on in interesting time intervals. It only shows the PIP if the viewer does want to see it. Therefore two registers are used:

- GPR(10) indicates with value “1” that the viewer wants to see the PIP movie
- GPR(11) indicates with value “1” that the movie plays in an interesting time interval where the PIP can show something of interest.

The yellow button toggles the PIP status on or off. But not directly: it raises the value “1” in GPR(10). Depending on the value in GPR(11) the PIP movie may actually show. An exclusive Switch action can do this.

```
if [GPR(10)=1] → [set]: GPR[10] = 0  
[set]: GPR[10] = 1
```

The GPR(11) is set by chapter mark events. A chapter mark action sets the GPR(11) to “1” at the start of an interesting interval. Another chapter mark resets the value to “0” at the end of the interval. In this example, the interesting periods are between 10-20 seconds, 30-40 seconds, 50-60 seconds and 1:50-2:08 minutes.



How to set actions on chapter marks is explained in “Method 4: movie chapter action” on page 304.

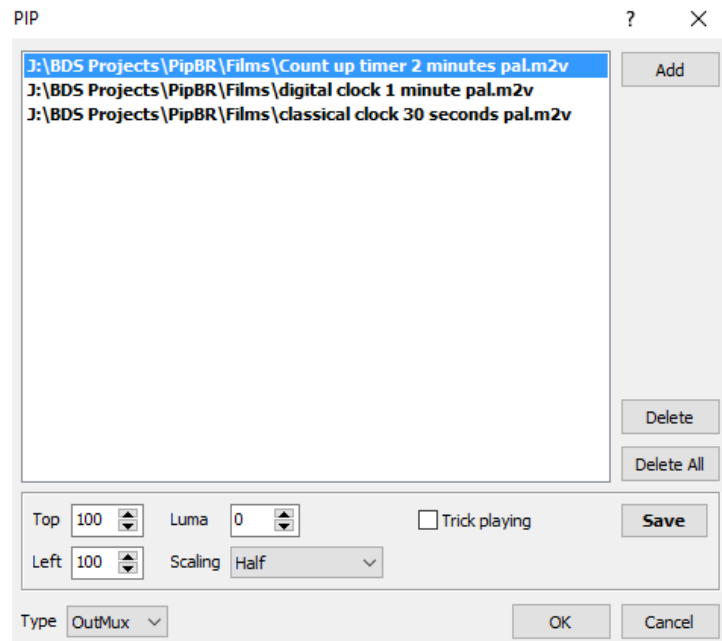
A “Action every second” on the movie checks every second if both registers GPR(10) and GPR(11) are set to “1”. Then, and only then, it switches the PIP status to on, otherwise to off.

One caveat not solved: if you manually move the progress bar of the movie and thereby skip a chapter with its action to set GRP(11)=1, the conditional values are not set by that chapter – which may result in not showing the PIP movie although it should. And vice versa: a PIP movie may show where it should not.

This problem can be fixed – but only in Java code as shown in Project 4

Project 3: select between PIP movies

The “3 PIP 3 clocks different lengths.bdmd” project has 3 different PIP movies associated to the Australia movie. They are of different lengths (30 seconds, 1 minute, 2:08 minutes).



The rest is straight forward:

- Define exclusive switch action to use the remote-control yellow button as toggle to show PIP or to hide it
- Define the “1” button action to select PIP video #1
- Define the “2” button action to select PIP video #2
- Define the “3” button action to select PIP video #3

Remote control buttons	
1	Set PIP video #1
2	Set PIP video #2
3	Set PIP video #3
4	
5	
6	
7	
8	
9	
0	
Red	
Green	
Yellow	[SWITCH]
Blue	
Right	

Keep in mind that not all PIP movies have the running time of the main movie Australia. Selecting them therefore, may not always show them.

Switching between PIP movies may temporarily freeze the main movie until the external PIP file has been opened.

Project 4: the Java solution of to project 2

The problem with Project 2 is that to work properly, it is necessary to pass each chapter mark. If you skip the chapter (by fast forwarding through the movie) the chapter mark action is not executed and the the GPR(11) register does not get modified. Something else than expected might happen.

This can be solved by using another approach, that requires some JAVA coding. Open BDS using the project file “4 PIP as display PIP selectively through Java.bdmd” in the \PIPBR project folder.

These are the steps taken:

- The yellow button sets GPR(10) to 0 or 1, indicating that PIP can be switched on or off. This is identical to the step in project 2.
You don't want the button to set PIP on or off itself as that would always show the PIP movie as soon as the button sets PIP=on. Its setting is then quickly modified by the “Action every second” action that is attached to the main movie.
- The movie “Every second action” checks the value of GPR(10). If it is 1 (PIP can be shown), it checks if the current play time of the movie is in an interesting region. If so, PIP is set on, otherwise it is set off.

This requires an exclusive Switch statement for the action of the remote-control yellow button:

```
if [GPR(10)=1] → [set]: GPR[10] = 0  
[set]: GPR[10] = 1
```

It functions as a toggle switch. Press the button and if the value of GPR(10) is 1, it is set to 0. If it was anything but 1, it is set to 1. Value 1 indicates the PIP movie can be shown. It now all depends on whether the movie is in a PIP-interesting time interval.

For this, a bit of Java code is written for the “Every Second Action” of the movie: a “Script (java)” action.

It runs every second, checks if GPR(10)=1. If not, it sets PIP off and exits execution quickly.

If it contains value 1, the movie time is retrieved and checked if it falls within an interesting time interval. If so, the PIP is set on, otherwise set off. The java code below can also be found as “Australia Action Every Second java.txt” in project folder “\PIPBR\original sources”.

```
// quick exit if no PIP should be shown, i.e. GPR(10) does  
// not contain value 1  
  
// do not use PIP=on in a Switch action as this will  
// shortly display the PIP even if outside special ranges  
// as these ranges are checked in this procedure  
if (manager.getGPR(10) != 1 )  
{ manager.setPIP(false);  
  return;  
}
```



```

// define 2-dimensional array with start and end times in
// seconds for relevant sections

// Java normally accepts
//      long RegionTimes [][] = new long[4][2]
// but BDS java does not as it must separate definitions
// of variables from their use to avoid problems with the
// same variables from different places
//

long RegionTimes[][];
RegionTimes = new long [4][2]; // 4 intervals start/stop
long MovieTime;

// specify start and end times of interesting PIP time
// intervals. Index starts at 0
RegionTimes[0][0] = 10;    // start region 1
RegionTimes[0][1] = 20;    // end region 1
RegionTimes[1][0] = 30;    // start region 2
RegionTimes[1][1] = 40;    // end region 2
RegionTimes[2][0] = 50;    // start region 3
RegionTimes[2][1] = 60;    // end region 3
RegionTimes[3][0] = 110;   // start region 4
RegionTimes[3][1] = 128;   // end region 4

// current play time
MovieTime = manager.getMediaTimeInSeconds();

// check if time is within a region
// if outside region, set PIP off
for (int i=0; i < RegionTimes.length; i++)
{ if (MovieTime > RegionTimes[i][0])
    { if (MovieTime < RegionTimes[i][1])
        { manager.setPIP(true); return; }
    }
}

manager.setPIP(false);

```

This script is part of a property of a movie. Therefore, it needs to be duplicated and made a “Script(java)” for the “Action every second” property of all movies that have PIP movies associated with them.

Of course, the array RegionTimes needs to be filled with start and end times in units of seconds that fit with the specific movie and its PIP movie. If there are more or less interesting time intervals (nr_of_intervals), the size of the RegionTimes array must also be adapted accordingly:

```
RegionTimes = new long [nr_of_intervals][2];
```

The second dimension remains 2 as it caters for the start [0] and end time [1] of each interesting time interval.

Real PIP example: Big Buck Bunny

Using PIP movies can be made more elegant than the basics discussed earlier.

A way to do this is shown in the BDS example project “Big Buck Bunny” (BBB for short). This project may seem daunting at first. The logic behind the buttons and menus is given in this section and we’ll take it step by step.

The movie that has a PIP movie associated with it, is called “movie”.

BBB is of Dutch origin. You may check it out on <http://bigbuckbunny.org> or <https://peach.blender.org/>.



A film review said:

“Radically different is Big Buck Bunny from the Netherlands computer graphics teacher Sacha Goedegebure, a comedy about a fat rabbit taking revenge on three irritating rodents. The film – in the fastest imaginable 3D computer animation – almost seems like being produced by Blue Sky studios, the producer of Ice Age 1-2-3, so natural and precise as the rabbit hairs wave in the wind.”

Regular menus

There is an opening menu with a menu item “Extras” and “Commentaries”. The latter opens a new menu (sliding in from the right-side edge) to select what audio track to hear.

Of interest for PIP movies is the menu item “Extras”. This opens either the “Extras” menu (if PIP is off) or the “Extras PIP” menu (if PIP is on).

Both Extras (PIP) menus are a clone of the main menu, but with an added billboard providing the Extra menu items. This billboard is animated and appears from below.

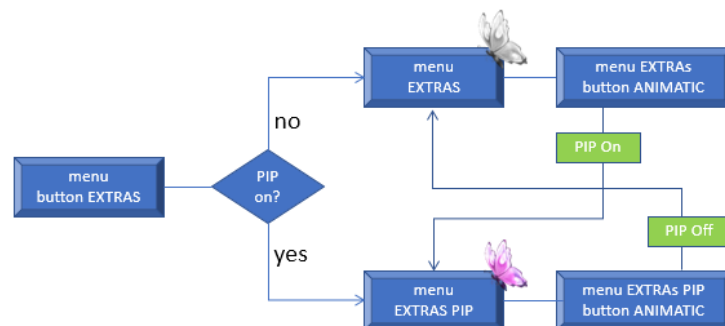
The items (and buttons) on the billboard are identical in both Extras (PIP) menus, including the one that we’re interested in: the first item, “Animatic”.



The button “Animatic” only toggles the status of PIP (Picture in Picture): on/off and changes the “Extras” menu for the “Extras PIP” menu (or vice versa). Both menus are identical except for the colour of a butterfly. The butterfly functions as a visual indicator that PIP is on (pink: menu Extras PIP) or off (grey: menu Extras).

You have to clone menus as BDS cannot change a picture element in shape or colour except by replacing the entire menu that is only different for this one picture element. The viewer won’t notice the difference except for the butterfly appearance – the only difference between both clones.

The BDS logic via a Switch statement on the “Extras” button of the main menu is shown below.



Nothing much happens until the viewer decides to “Play Movie”. For this he has to close the “Extras (PIP)” menu and return to the main menu.

Return to the main menu occurs when either:

- The left arrow button is pressed (all buttons have their “Press LEFT” action set to Jump menu – so any button can be the “selected” one)
- The down arrow button is pressed on the “Extras (PIP)” menu button at the bottom (that button has its “Press DOWN” action set to Jump menu)

Movie properties

Back on the main menu, the “Play Movie” button starts the movie. The movie has associated a PIP movie to it in its PIP property.

The movie also has two special actions:

- A popup menu that appears when the remote-control popup button is pressed
- Pressing the yellow button will display or hide the PIP movie. But only if the PIP status is on.

Movie popup menus

The movie has its “Popup menu” property set to a Switch action, that opens a different (but very similar looking) popup menu depending on the current state of PIP on/off. The state is again indicated visually by the same butterfly attached to the “Animatic” button.

The popup menu is a fair clone of the main menu.

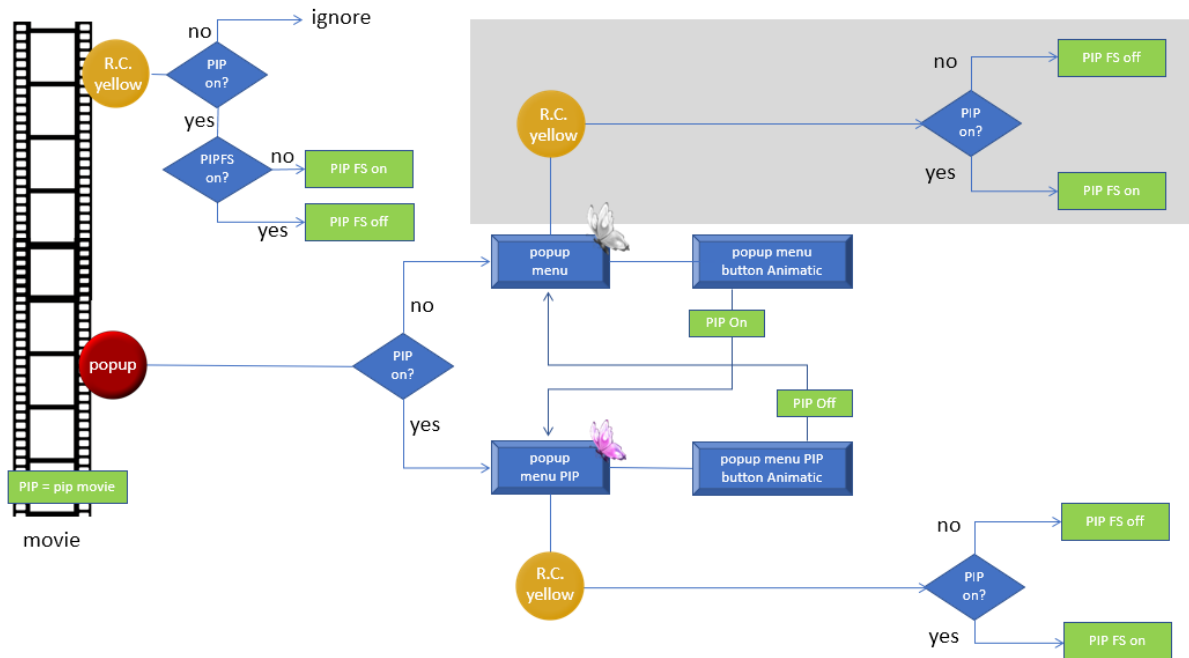


When PIP is on, the popup menu “Menu PIP” is shown, if it is off, the popup menu “Menu” is shown.

Just like the normal menus, one of the popup menus allows to enable PIP, the other to disable PIP by its “Animatic” button. The PIP status is reflected in the colour of the butterfly.

The “Commentaries” opens another menu to select the audio track to use for the main movie – as it does on the opening main menu.

When all is set and done, the popup menu can be hidden (animates out of screen below) by pressing the “popup” button on the remote-control again.



The logic programmed into the project is shown above. The yellow button is defined for both popup menus as well as the movie itself. This way it always works when the movie or its popup menus are displayed.

I inserted part of the logic in the project in a gray rectangle. This part seems pointless and not needed. When the shown popup “menu” implies that PIP is off, there is no point in checking if PIP is on. And setting PIP full screen or not is equally irrelevant if it is not shown at all because PIP=off. This superfluous button action is probably the result of a popup menu clone of “menu PIP” to “menu”.

When PIP=on, pressing the yellow button will toggle the PIP movie to be displayed in full screen or not. This yellow button action is defined for both the “menu PIP” popup as well as the movie itself.

In case of the yellow button for the movie, the toggle is achieved through a Switch action that is defined as

```
if [IsPIPFSoFF] → Set PIP FS: on
if [IsPIPFSon] → Set PIP FS: off
```

This full screen status only has effect if PIP is on. If not, neither condition is true (nor relevant) and the Switch results in “do nothing”.

Part 7: 3-D

3D – in 2D and true 3D flavour

“3D” (as abbreviation for “three dimensions”) is in fact the wrong name for that we’re going to talk about. True 3D means projecting an object you can walk around and look at it from different angles – just like in reality. 3D can be obtained using holograms but that technique is still in its infancy and certainly not part of what a bluray disc can deliver.

What the movie industry indicates as “3D” is in reality “stereoscopic”: two 2D images taken from slightly different positions, giving your brain the illusion of depth by the parallax between the images. But it is depth from a fixed position – the position the camera was in. You cannot walk around to look at it from a different angle – the image is fixed as if you look through a window.

There are two methods in use that create 3D movies

- frame packed (.mvc files) which are used for most commercial 3D bluray discs. The images are full frame and HD resolution. The disc has two .m2ts files (one left eye, one right eye) for each 3D movie and a special .ssif file that connects the two and is the one played by the 3D bluray player.
- Side-by-side or over-under (SBS/OU). Both left/right eye images are stored in the same frame – each using half of it. There is a single .m2ts file and any bluray player can play it since it is a standard 2D bluray disc.

Most commercial bluray discs advertised as 3D are post-processed 2D movies. They are still a lot better than the bluray player or video projector that offer “real time conversion 2D-to-3D” where identical pictures are shifted horizontally to create fake depth and where fast moving objects may be considered foreground as anything in the background cannot move that fast on screen.

The table below gives some major differences between the two 3D supported methods.

property	MVC	SBS
files per movie	3 (left/right eye .m2ts renamed to .mvc plus coordinating .ssif file)	1 (ordinary .m2ts movie fie)
Requires 3D player	yes	no
Requires 3D tv set	yes	yes
Requires 3D video projector	yes	yes
BDS handling	as 3D project #D movies need to be in their own JAR titles	as any other 2D project
disc muxing	result needs modifying before burning	result can be burned directly

True 3D (mvc)

The format BDS uses for true 3D movies is multiview video coding, file type .mvc. It is also known as “frame packing”: two full resolution left/right images projected quickly one after the other for the targeted left/right eye. Such discs can be shown in 2D or 3D. In 2D only the left image frames are shown.



Side-by-Side flat 3D

The Half Side-by-Side SBS (or its less commonly used sibling Over-Under OU) is often used for 3D movies shown on YouTube as well as those created by various consumer type 3D movie cameras and some stereoscopic-headsets for mobile phones.

Rather than having frame packed complete left/right images, Side-By-Side shows left/right images within the same frame and therefore at the same time. The HD image is either cut in two equal sized images (2x(1920x1080) pixels) known as “side-by-side” or cut in two equal size half-width frames (2x960)x1080 pixels, known as Half SBS format.

The former, full side-by-side, pairs are seen on mobile phones, headsets or stereoscopic viewers. It looks like looking in a ViewMaster 3D viewer – but with better resolution.

The latter, half-SBS, is used in stereo TVs and projectors because they can only show 1920x1080 pixel images and hence a stereo pair must fit inside this area – and can if squeezed horizontally.

An advantage of (Half)-SBS over frame-packed “true” 3D is that SBS discs do not need special bluray players. Any player can play the disc as it is a 2D picture. On normal tv sets or video projectors you would see two images, squeezed horizontally, next to each other on screen.

To see them as one stereoscopic picture, you need a special stereo 3D capable TV set or projector to interpret these two images as a left/right eye pair. This is done by setting the projector or tv to “3D – Side By Side”.

A disadvantage of half-sized SBS is that the horizontal resolution is effectively only half of 1920 pixels: 960 pixels. These pixels are stretched by a factor of two to use the full width of the HD screen. SBS can be used by any bluray player at the expense of losing half the horizontal resolution. If the reduction of height resolution is less important, sometimes the Half Over-Under format is used.

Because SBS is a 2D format, BDS can handle this type of movie as it does any other 2D movie.

You may not have to bother to author a bluray disc with BDS for a SBS 3D movie. Many bluray players and stereo tv sets are capable of:

- running a slide show directly from images (.jpg and other image formats) from a USB stick – you do need to make sure they have the proper widescreen format (see next section)
- playing .avi or .mkv movies directly in the proper 4:3 or 16:9 aspect ratio. when instructed to interpret the movie as SBS.

When slides or movies are SBS stereoscopic, you do need to set your television set or projector accordingly.

Mixing 2D and 3D

BDS knows not of stereo menus. All menus are 2D – even if the project is a 3D project with “true 3D” .mvc frame packed movies.

For SBS stereo it is clear: SBS is effectively a 2D image, that consists of two almost identical pairs for left/right. A menu for SBS stereo therefore needs to be twice the same menu where one menu is only 910 pixels wide and the left and right eye version are next to each other.

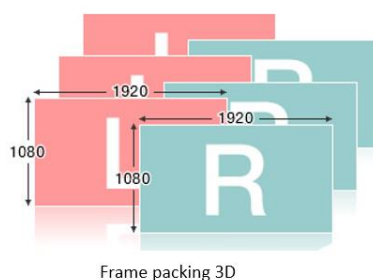
It is generally a bad idea to mix 2D and 3D movie contributions on the same disc. The menus on SBS discs may look double when shown without 3D or a full screen menu looks distorted when the right half is projected over the left half if the tv set is set to SBS.

To be consistent within a disc, any 2D movie can be converted into a 3D movie with two identical left/right images: it remains looking “flat”. Most video editors have options to convert a 2D movie into 3D SBS (even if there is no depth information present).

True 3D movies (mvc files)

True 3D movies on bluray are usually only made professionally and not by amateur movie cameras. But what to do if you have such a true 3D movie file?

Such 3D files are named "multiview video coding" or .mvc for short. They have full resolution images for the left and right eye. These are interleaved and the bluray player, in cooperation with 3D video projector or television and passive or active shutterglasses, show them in rapid succession, giving the illusion of stereoscopic images. This technique is known as "frame packing".



Some help to produce such an image with BDS can be found in its online help section "Creating project with 3D video and 2D menus".

If you encounter Side by Side stereoscopic files (discussed later in this chapter) you can convert it to a regular .mvc file if your video editor is capable of doing so. The resolution does not get any better but most likely the bluray player will automatically switch to 3D mode and you do not have to set it to SBS mode manually.

In Cyberlink's PowerDirector, open the SBS file for editing. Indicate clip attribute as "stereo - side by side" and go to the "Produce" tab. There, select the 3D tab, click the H264/AVC button and specify as output Multi Video Code – perhaps modify the profile to use the proper resolution, bitrate and frame speed.

SSIF files

True 3D movies on bluray are stored in a way that confuses most products that can rip a 3D bluray disc to a hard drive. A 23 GB disc can easily result in a 40 GB hard drive image. Where does all the new information come from? The reason is the presence of a set of .ssif files.

The main difference is that a special MVC (Multiview Video Codec) based on the AVC stereo profile is used for video compression, which was originally designed for compressing stereo pairs. The main*. M2TS file contains the left angle and a special "3D" flag for 3D players. The SIFF (StereoScopic Information File) subdirectory contains information files that allows you to calculate the right angle based on the left angle. With this encoding, you can save about 25% of disk space. The

accuracy of the synchronization of the camera angles is provided not by the player, but by the compression format itself.

The SSIF files augment the information of the regular "left eye" video file. Simply copying the disc would mean not only copying the .m2ts files but also whatever the SSIF files is using from these files: another video file that replicates image information already present in the first two .m2ts files. You don't get the SSIF: you duplicate part of the .m2ts files. And that ruins the entire 3D effect because the SSIF files with their left/right information of the movie are gone in the process.

To my knowledge only a single product, DVDFab Bluray Copier, is capable of duplicating a 3D bluray onto another 3D bluray (shrinking it to BD-25 if wanted, removing regional codes and encryption).

In order to make a 3D movie disc using some .mvc files using BDS requires some specific steps that we will highlight next.

Making a 3D MVC disc with BDS

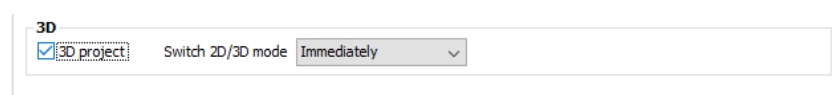
Stereoscopic camera output might be given file extension .mvc but it may also be a .m2ts. Check with a tool like MediaInfo whether the file does contain two video streams. Only then it is a 3D movie file.

When a camera has produced a .mvc 3D movie, how do we transfer it to a proper 3D capable bluray disc? Follow the following steps.

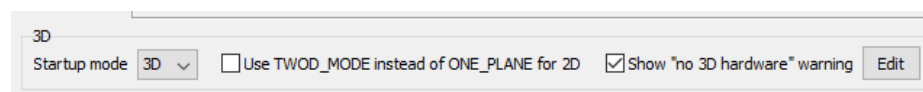
Define a BDS 3D project

BDS must provide special information in the muxed output result that this project contains 3D movie contents. To avoid disappointments or unexpected behaviour, it is advised not to mix 2D and 3D contents on a disc.

For this, you need to open Project > "Project Properties" and check the 3D box.



The presence of a (link to a) single 2D movie in a JAR file makes the entire JAR to show itself and linked movies in 2D – regardless whether they are encoded as 3D mvc. Because of this, there are also some settings required for 3D in the "Project > JAR Settings" (at the bottom of the window).



This box allows to show a message for 2D players that cannot handle 3D discs (check "Enable 'no 3D hardware' "). There is a default message text. You can edit this by clicking on the Edit button. If all movies are 3D they can share the same JAR title, with Startup mode

3D. But if a single one is 2D, the entire JAR title is shown as 2D regardless.

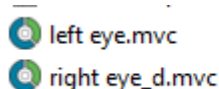
Be aware that the BDS Java functions `manager.set3Dmode()` and `manager.set2Dmode()` will throw errors when used while compiling with `tsMuxer`. If used, you need to comment those program lines to make them inactive.

Demux mvc files

Like any BDS project, the movies need to be demuxed into their video and audio streams. Where ordinary movies have one of each, 3D movies have two video streams in a single `.m2ts` movie file.

The demuxer might produce `.264` movie files by default from the muxed `.m2ts` file. In this case you need to rename their file extension to

- `*.mvc` for the `.264` file that contains the left eye image
- `*_d.mvc` for the `.264` file that contains the right eye image



If both files are demuxed as `.264` the slightly smaller one in size in the right eye one to be renamed `*_d.mvc`

(Note: the file names need to be identical – it's only for clarification purposes we named them "left eye" and "right eye_d" respectively).

When adding a 3D movie to the Project Tree you need to select the `.mvs` video file (rather than `.264`). Because the project property "3D movie" is checked, BDS knows it needs to read both the `.mvc` and `_d.mvc` video streams.

Streams	
Video	D:\Users\Theo\Documents\Downloads\complete\bluray\left eye.mvc
Audio 1	D:\Users\Theo\Documents\Downloads\complete\bluray\left and right eye sound.ac3
Audio 2	

Chapters

Once the movie is specified in all its properties (including End Action and Popup Menu), you can create chapter marks (play marks) the usual way: create the index files and set the chapters.

You can use the chapter menu wizards to create 2D chapter menus that you attach to the movie as Popup Menu like for any other movie. The menus must be part of the same JAR title as that starts the 3D movie.

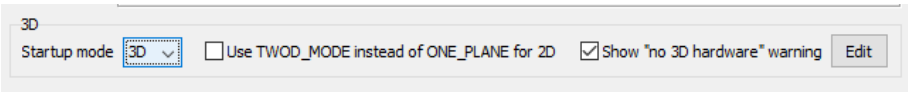
Menus: multiple JAR files

BDS can only create 2D menus.

This can force the use of multiple JAR files if you have a mixed 2D/3D movie content. A JAR file (called "title") can only link to either 2D or 3D movie parts. If both are present, the JAR title becomes 2D on playback.

If a 2D opening main menu movie is linked to a JAR (say 00000, which has also set its First Play to show the main menu when the disc is started) the disc opens as 2D.

If the menu movie is in 3D the opening menu will play it in 3D provided the project is set to startup in 3D (menu Project > Project Settings).

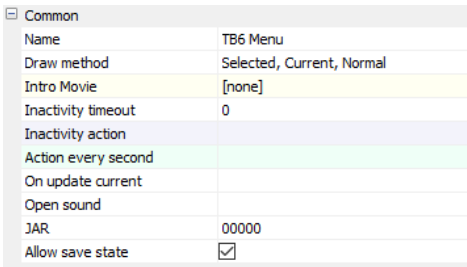


The button that activates the 3D movie points to a different JAR (say 00001). In the JAR Settings for that JAR you specify as "On JAR startup" the 3D movie to run.

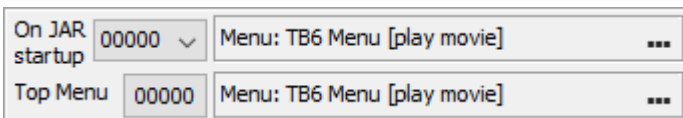
If all movies, including menu movies, are 3D you may use a single JAR title for all (this used to be the case in older versions of BDS when you could only use a single JAR title).

These are the steps to take when using menus:

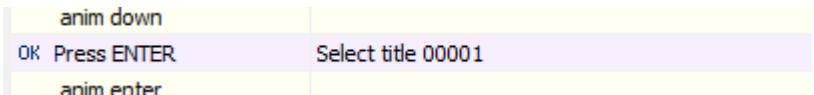
- in menu properties indicate in what JAR it resides (JAR property) (example: 00000)



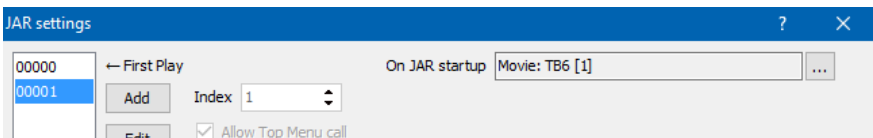
- For proper startup of the disc, set the menu as opener when that JAR title is set as "First Play"



- The button on the menu that activates the 3D movie simply selects another JAR – the one that has the movie as startup



- The properties of the JAR title mentioned (Project > JAR Settings) must indicate that it starts by playing the movie



- The 3D movie must indicate what to do when the movie completes (End Action) or a popup menu is requested

Start Action	
End Action	Select title 00000
Action every second	

- Generated chapter menus are also 2D and reside in the JAR title that has the 3D movie as On JAR startup. The Popup Menu property of the movie should point to those menus.

Although a movie is not part of a JAR title, it is linked to a specific JAR title because that JAR title refers to it (“First Play”). As such you might say “the movie belongs to a specific JAR title”.

Note: You can have different objects to run/show on JAR startup. The example above shows a movie to run. But when you click on the “...” button you can also select a SWITCH statement where depending on some condition (such as a set GPR register) another object is run/shown.

3D Subtitles

With any regular subtitle tool you can create a .srt file to sync titles with the image. A .mvc frame-packed movie is no different than a regular movie for BDS. The movie placeholder has two video streams (*.mvc and *_d.mvc). The subtitles are simply added in the subtitle property. Muxing the project will show the subtitle as if it was a regular subtitle superimposed upon whatever left/right eye image is shown. BDS does not allow to specify a particular depth of the subtitle relative to the 3D image.

When titles are ripped from existing 3D movie blurays, the titles will be in a .sup file – each title is basically a transparency laid over the movie image.

Mux the project

Then the final project can be created by muxing it. Preferably by the internal muxer of BDS MX but the latest version of the external tsMuxer also works.

The result is the usual output folder with two subfolders: BDMV and CERTIFICATE.

The BDMV\STREAM will contain both .m2ts files for the left and right eye. It is currently not usable as 3D bluray disc. Some additional steps must be taken for BDS and BDS MX. If you would use the currently created folders, a flat 2D bluray would be the result. So **DO NOT** burn the resulting folders on a bluray disc.

Only BDS MX Pro can create a production-ready CMF file that makes a proper 3D disc image.

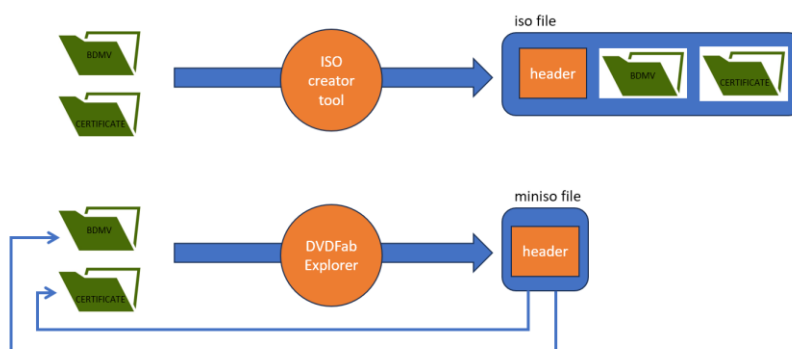
The following steps are needed to create a real 3D bluray disc for all other versions of BDS.

Create a miniso disc

A what? Never heard of miniso? Nor did I. But it is a specific format known only by the freeware DVDFab Explorer.

You need to install the DVDFab Explorer (see appendix section "Virtual disc drives" on page 502 or search on internet for "DVDFab Explorer"). Once installed it will normally start when Windows is started (but you can switch this off). It can be seen then as small icon in the system tray (🔍). Clicking on it opens the interface. A right click opens a context menu that may also suffice for our purposes and includes a "setting" option (useful for not autostarting at Windows startup).

Despite its name it does not take over Windows Explorer but rather is a tool that enhances this Explorer with some functionality to create iso images or to open iso images as a virtual disc.



Some media player applications can only play an actual disc or an .iso image file of that disc.

Because the BDS folder is already there, making an ISO file would duplicate all files by inserting them into this .iso file. The .miniso only provides the set of headers that full .iso files have, but it does not copy the actual files themselves. Yet it does assign a drive letter to the BDS created folder. Hence, if we mount a .miniso file through DVDFab Explorer the playback application thinks it's playing a (virtual) disc whereas it just plays back the BDS output folder.

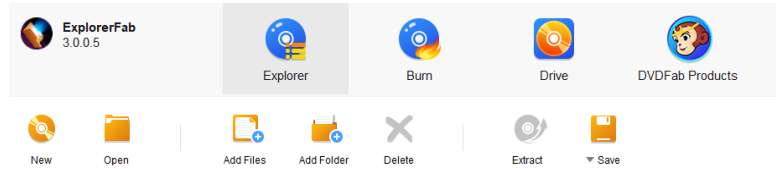
When the BDS folder is mounted as .miniso disc, look at the files on the mounted virtual disc under the \BDMV\STREAM. You'll discover that it has a \ssif folder with .ssif files that do not exist in the BDS output folders: precisely the files needed to make a 3D authored disc play in 3D. The FABdvd Explorer correctly interprets the BDS information on the two .m2ts files for the left/right eye and inserts the coordinating .ssif file for the coordination.

Execute the following steps to create a proper 3D bluray disc from one or more .mvc movie files.

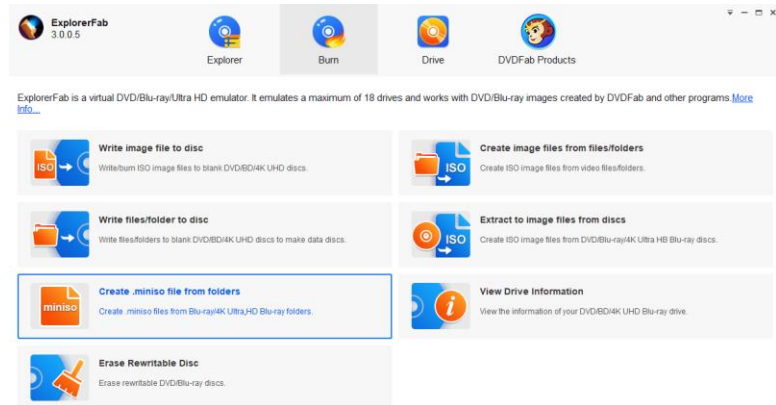
[DVDFab Explorer used from its user interface](#)

1. Open the user interface by clicking on its icon in the system tray. If there is no such icon, start it the usual way from the Windows Start menu. Look for "Explorer FAB".

This will show the general interface of the tool. By default the interface shows what the "Explorer" button choice shows.

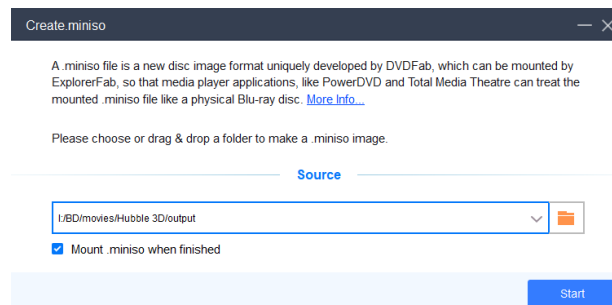


2. Click on the "Burn" button. This changes the interface and shows a set of options to create a proper .iso image or to create a .miniso from a top folder. Select this latter option.



This will open a navigation window to specify the top folder of the BDS created muxed project.

3. Navigate to the topfolder in which BDS created the \BDMV and \CERTIFICATE folders (e.g. the \OUTPUT folder as we used in many earlier projects). Check the "Mount .miniso when finished" box to automatically start the virtual drive that contains the folders.




You should be able to drag/drop the folder but this does not seem to work in version 3.0.1.7

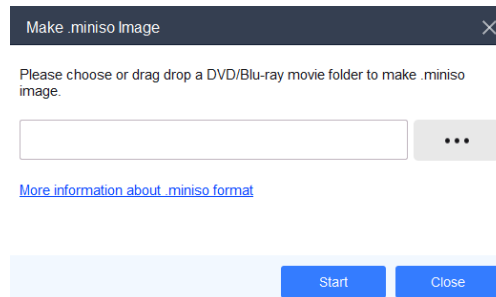
4. This opens a window asking whether you want to play the virtual disc with DVDFab Player (which you can purchase) or to open it in Explorer. Don't do either (close the window through its X icon)/

A new drive letter is created (e.g. G:\). When you inspect the drive in Windows' Explorer It should show the \BDMV and \CERTIFICATE folders of your 3D project output.

5. as a double-check, you can click on the \BDMV folder and navigate to the \SSIF folder with its .ssif files: a sign it now looks like a proper 3D bluray disc.

DVDFab Explorer used from context menu

When you have the DVDFab Explorer icon in the system tray (), right click on it to reveal the context menu. Select the "Make .miniso image" option. This opens a window in which you can navigate to the top folder of the BDS produced output. You can also drag/drop that folder onto the window.

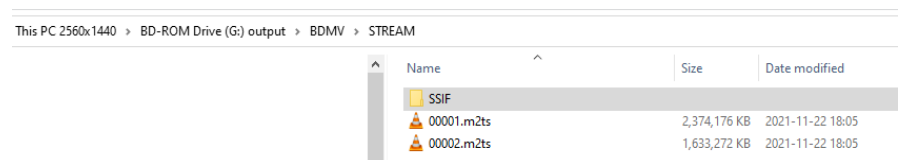


After this, steps 4 and 5 from the previous section are executed. Start the virtual disc by exploring it.

Burn the disc

6. You can use ExplorerFab (under the "Burn" button) to create a proper .iso file of the virtual drive opened via .miniso. The resulting .iso file can be burned to disc at any later moment in time using tools like ImageBurn or Nero.

Alternatively, you can use the mounted .miniso drive immediately to burn all of its contents to an empty bluray disc Nero's "Copy Bluray Disc"⁵⁴. Simply use that burning tool and burn the virtual disc containing the BDMV and CERTIFICATE folders.



The process of burning the 3D bluray disc is no different from burning the usual 2D bluray discs.

⁵⁴ There does not seem to be a "copy disc" like option in ImgBurn. You need to create a full ISO file first through DVDFab Explorer. Or – if you want to use ImgBurn all the way, it is a 2 step process: first use "Create image file from files/folders" and then "write image file to disc". The first step is identical to creating an ISO file from DVDFab Explorer.

Close left open windows of DVDFab Explorer

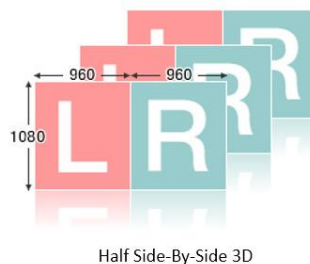
The DVDFab opened windows (navigate to folder and start, play or explore and mount disc) for some reason remain open even after use . Close them by clicking on its "X" button at the righthand top.

DVDFab Bluray Copier

If you have the DVDFab Bluray Copier package, this can also create and read .miniso files. Steps 3 upto and including 6 above can then be achieved within this product: it creates a .miniso file and from that it burns a disc directly or creates a proper .iso file.

Flat 3D: Side-By-Side (SBS) and OU (Over-Under/Top-Bottom/Above-Below)

A flat 3D movie is a movie like any other, but with the image frame subdivided into two equal halves, containing the left and right eye image side by side.



The 3D format “side by side” (SBS) is popular with

- most movies copied from the internet
- created with consumer 3D film cameras
- still images taken with 3D stills cameras

The width of the film frame is split in two equal halves, using one half to show the left-eye image and the other half to show the right-eye image.

It is a flat frame (and hence needs no special bluray player) where each half is shown in rapid succession for only the left or right eye, giving the illusion of depth at half the HD resolution (widths of 960 pixels for left/right image instead of 1920 pixels or a reduced height of 540 pixels instead of 1080). To show images or movies like this, the TV or projector needs to be capable of handling 3D movies and moreover needs to be capable of handling SBS or OU.

Several variations exist:

- Full SBS – two frames (one left, one right) each with width x height = 1920 x 1080 pixels are positioned next to each other. The composed frame has dimension of $(2 \times 1920) \times 1080 = 3840 \times 1080$ pixels. This format cannot be used on current 3D televisions or video projectors. If you want to keep the full HD resolution, you need a video editor capable of producing a .mvc file out of each left/right pair. Or reduce it to a Half SBS file.
- Half SBS – two frames of 960 x 1080 pixels are positioned next to each other. The horizontal width has been squeezed by a factor of two. The composed frame has dimension 1920 x 1080 pixels – suitable for all HD equipment. The 3D tv or video projector rescales the image back to 1920 x 1080 pixels by duplicating each horizontal pixel. The effective resolution in horizontal direction is only half the Full HD resolution.

- Full OU – over-under, also known as “top bottom” or “above below”. It is similar to SBS but rather than next to each other, both frames are stacked on top of each other. The composed frame has dimension 1920 x 2160 pixels. Some claim the eye is less receptive to this loss in vertical resolution than in horizontal resolution with the SBS image. Nonetheless almost all 3D images and movies are taken in (Half) SBS format.
- Half OU. Similar to Half SBS but with images stacked with half their width. The effective frame has dimension 1920 x 1080 pixels. The 3D TV or video projector will stretch the height by a factor of two by duplicating every vertical pixel. The effective resolution in vertical direction is only half the full HD resolution.

3D TVs and projectors need to rescale SBS to the final size of 1920x1080 pixels. With SBS you always have two images: one for the left eye and one for the right eye. They are positioned next to each other or on top of each other.

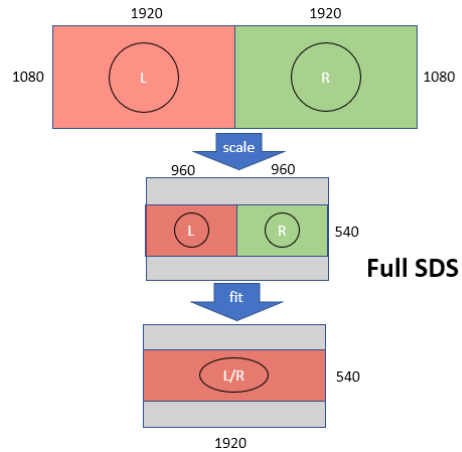
There are several video converter tools on the market that allow to convert a 2D movie into a 3D movie – usually in a (Half) SBS format. Many are not worth the money (produce anything but thure 3D) but some tools produce a serious output using depth maps created through fuzzy logic. It will never compete with true stereoscopic movies but can be quite acceptable. 3D Combine is such a tool – see Video editing and format converting on page 497.

Picture size

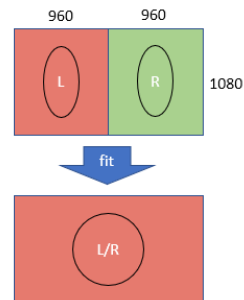
The basic question to ask is: what scaling factor is needed to reduce the image, keeping its aspect ratio, to fit the resolution of 1920x1080 pixels? The SBS image is then cut in half vertically and that image is stretched to full width (SBS) or height (OU). This stretching may cause deformation of the picture as the original aspect ratio is now lost.

This has some consequences for the various 3D formats. In the listing below it is implied that the image is reduced to have an aspect ratio of 16:9 (and if not, the hardware will make it so):

- Full SBS results in images with both left/right eye image in full width (1920 pixels). The HD size of 1920 pixels is half this width. Therefore, the height it also scaled by a factor of 0.5. The resulting picture is split in half for left/right eye image, resulting in a projected 3D image of proper width but half the height: 1920 x 540 pixels.

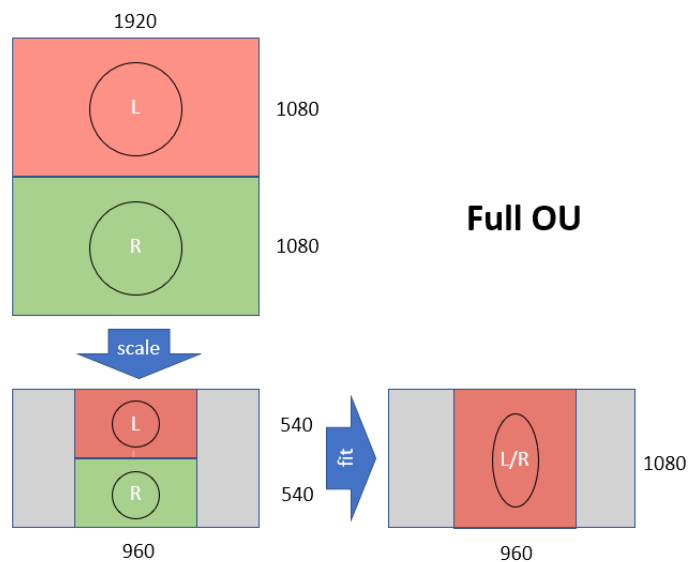


- Half SBS in images works fine. The aspect ratio fits the HD screen. If any scaling must be done, it keeps filling the HD screen. Because of SBS it is split in two halves for left/right eye image and stretched by a factor of two to fit the width again. The 3D image uses 1920x1080 pixels, The true horizontal resolution is only 960 pixels.



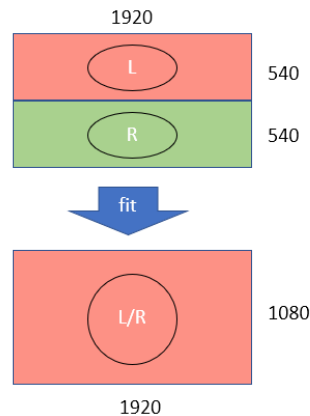
Half SDS

- Full OU reduces the height by a factor of 2 and therefore the width also. The resulting image has proper height but only half the width (960 x 1080 pixels)



Full OU

- Half OU will stretch its width by a factor of two for each of the left/right eye images, retaining their width. This will produce a proper image with an effective vertical resolution of 540 pixels.



Half OU

From these possibilities, only Half SBS or Half OU work for stereo tv and projector. These have a width of 1920 pixels and height of 1080 pixels (HD size) and both left and right eye image must be squeezed into this frame.

The video editor with which you process the images or movies may also only support creation of half-SBS movies. This effectively leaves Half-SBS to be the format of choice.⁵⁵ In the following, we assume Half-SBS is used when SBS is mentioned.

Files must be m2ts formats

Because BDS can only handle BDA compliant movie files, any flat 3D movie file must be converted to the BDS acceptable BDA (.m2ts) format. Several movie editors or file format converters will do this for you. This is no different from handling ordinary 2D movies. But many 3D SBS files may originate from internet or cameras using the MP4 format. These must be transcoded into .m2ts.

3D SBS menus

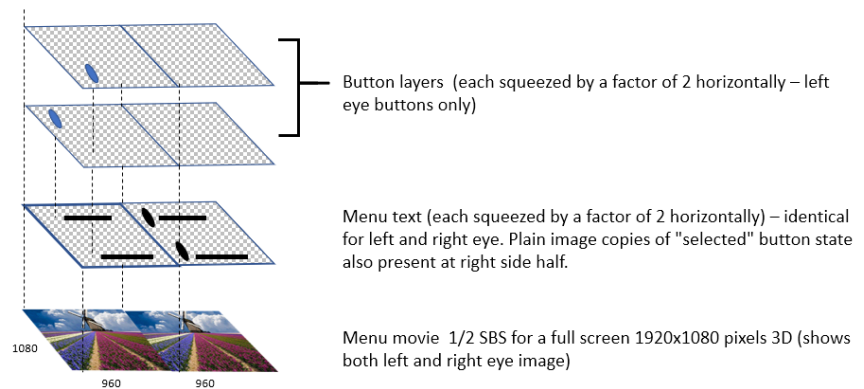
Because SBS is effectively a 2D format, you can create menus the usual way. With the target being played as 2D there are some things to consider though as the effective menu is only half-width.

In short: you need to build the menu twice: once for each eye. But only one of the eyes should have buttons that can be selected. The other eye has the exact same menu but all buttons are mere images.

Effectively this means that the menu must fit within half a full screen: its contents needs to fit within 960 x 1080 pixels (for a horizontally stretched projected 1920x1080 HD full screen image) or whatever size

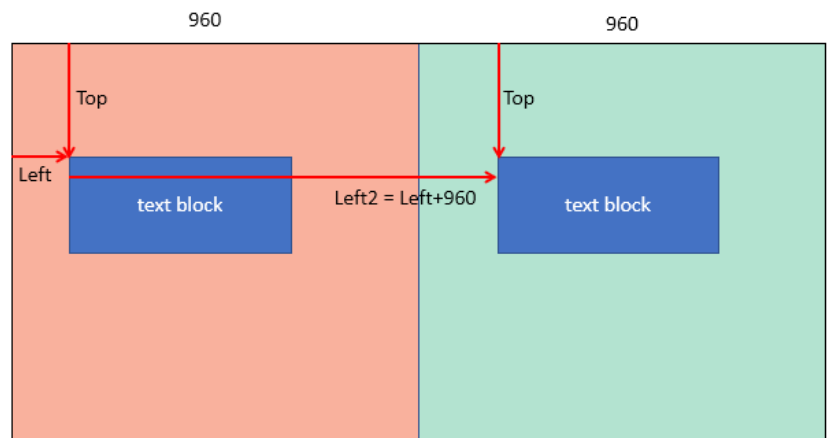
⁵⁵ Cyberlink PowerDirector accepts full SBS movies. The clip-attribute must be set to 3D SBS. On output, you can “produce” proper .m2ts frame-packed 3D but also full SBS or Half-SBS. Any subtitle added is “burned in” but properly set in both left/right image.

the background movie is (640x720 in case of 1280x720 pixels widescreen).



To ensure the menu looks “normal” when it is stretched to full screen on playback, the easiest way to achieve this is to:

- create it with an image editor such as Photoshop in full size (1920x1080 pixels or 1280x720 pixels). Or create it entirely from within BDS.
- Only fill the left half (960 x 1080 pixels) with any menu text and pictures
- After completing all the menu text delete the background layer.
- Import the Photoshop file as menu



- Notice the Top and Left position of the menu parts. Duplicate them in the BDS Designer Window to the right-hand part of the menu (CTRL/C, Copy to same menu). Keep the Top value the same, increase the Left position by half the menu width (+960 on a 1920x1080 pixels menu, +640 on a 1280x720 menu)
- Add buttons to the left part of the menu. Specify the image files that represent the button states. Add all navigation and “Press Enter” values. It’s easiest to only have a visible button if it is in the “selected” state.
- Duplicate the buttons to the right-hand part. To position them exactly right, keep “Top” values the same, increase the “Left” values by half the menu width.

- Change all button in the right half menu to a static image.
Renew the image file name if necessary. The image files must be identical to the image file used for the button state on the left half menu. This says each button state at the left has a corresponding image-only at the right.

From here on, there are several ways to make the left-hand side with the buttons of the menu look like the right-hand side with no buttons but images-of-buttons:

- Use linked objects (by far the easiest)
- Make the button-images visible in sync with the buttons through a multi-action

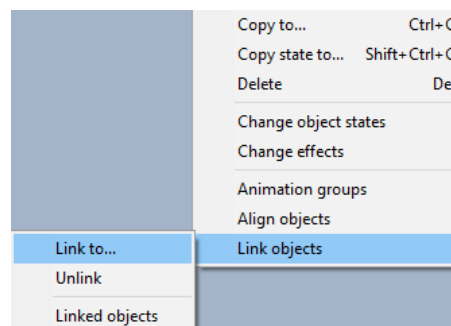
Use linked objects

Linked objects are picture images that you can link to another object in the menu. Whenever that other object becomes selected (for button objects), moves (any object part in an animation) or changes its transparency the linked object does the same. This way each state of a button on the left half of the menu has a corresponding image on the right half of the menu.

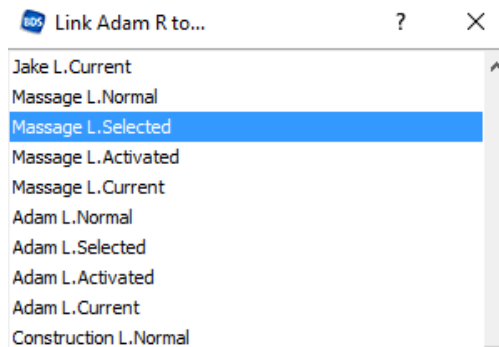
In our case (with only a “selected” image for the buttons) we need to copy all buttons on the left-hand half of the menu to the proper position on the right-hand half of the menu. When these copies have been converted to images, these images are linked to the “selected” state of the button. When this state becomes visible, so does the corresponding linked image. If other states also have images, you need to repeat this process (making the button states (in)visible may help) for each button state: link its left-menu button state with the identical looking right-menu button image.

To create linked images do the following:

- Copy the buttons from the left half to the right half of the menu
- Change the right half buttons into static images identical to the “selected” state images of the buttons in the left half
- Select each right half static image and right-click to open a menu you may have used for copying an image. However, it also has a “Linked objects > Link to...” option. Select this one.



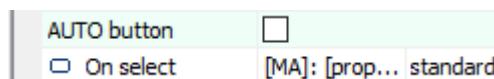
- Selecting the “Link to...” opens a window where all objects (and button states) are mentioned. Select the button and its selected state to link to the button-image in the right half of the menu.



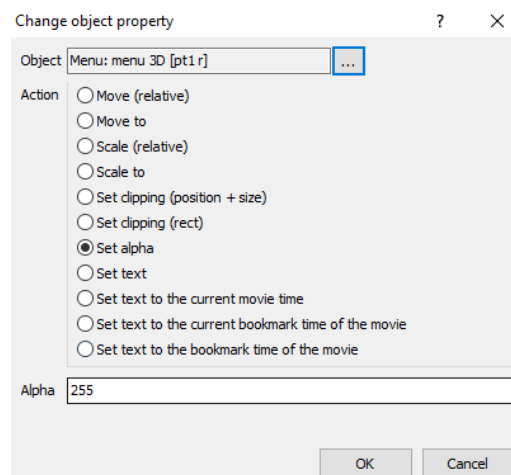
- Do this for all menu buttons and all of their defined button states that need a duplication as image on the right half of the menu
- Check the correct working by simulating the menu and try all buttons and their states. Left and right half of the menu should show the same button images

Use transparent/visible button images

- Add a Multi-Action to each button’s “On Select” state.

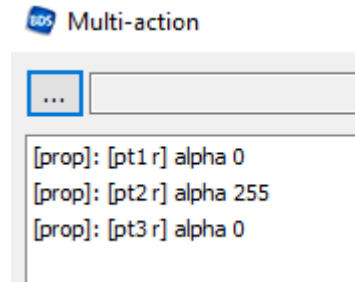


This multi-action sets the transparency of each right-hand side image to transparent (alpha=0) except for the image corresponding to the currently selected button. (object > Settings and Properties > Set Object Property and specify the image object whose transparency must change).



This way the same button image appears in each eye rather than only in the left eye. In case you have 3 buttons (pt1 l, pt2 l

and pt3 l – three parts, left eye button) and three identical looking images (pt1 r, pt2 r and pt3 r – right eye images identical to the “selected” state image of the left-hand side button), the multi-action looks like the one below when the “pt2 l” button is selected.



- Do this for all “On Select” actions for the buttons (and for any other button state you may have defined). Unfortunately, currently the Action Matrix fails to show the On Selected state so a quick copy is not possible.
- Check in simulation that any selected button in the left half also displays an identical picture in the right half of the menu.

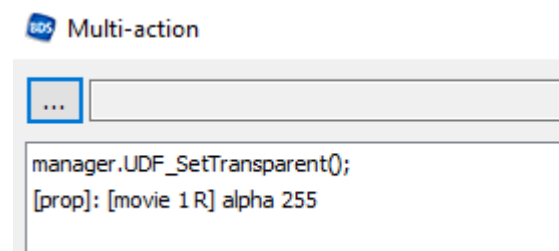
An alternative approach is to use a User Defined Function (Project Properties > Functions tab) called “SetTransparent” which uses the BDS Java function setAlpha() function that sets all transparencies to alpha = 0 for all images.

Such a function for “R”ight side button images in menu “main menu sbs” for buttons “movie N” (spaces need to be replaced by underscores in Java) would look like:

```
public void UDF_SetTransparent() {
    // set all button lookalikes in right side image to
    // transparent

    setAlpha("F:MM_main_menu_sbs.movie_1_R", 0);
    setAlpha("F:MM_main_menu_sbs.movie_2_R", 0);
    setAlpha("F:MM_main_menu_sbs.movie_3_R", 0);
    ...etc...
}
```

For each On Select action you then define a multi-action with two actions. First a all to the UDF to make all button images at the right transparent and then one additional action to set the proper button image back to visible (alpha =255). For the first button it would look like below.



(You may enhance the function with a parameter with the image name and have a single action calling the function to make all images transparent except the one specified as parameter).

```
public void UDF_SetTransparent(String imagename) {  
    ...set all images transparent...  
  
    // make the specified image visible  
    setAlpha("F:MM_main_menu_sbs." + imagename, 255);  
}
```

It may help do create one opening menu in plain 2D explaining the disc can only be viewed in SBS mode and TV set or video projector need to be set like that. The one button on that informative menu may open the first SBS menu created as described above. Any movie ending should return to the 3D menu and never to the opening 2D menu.

3D subtitles

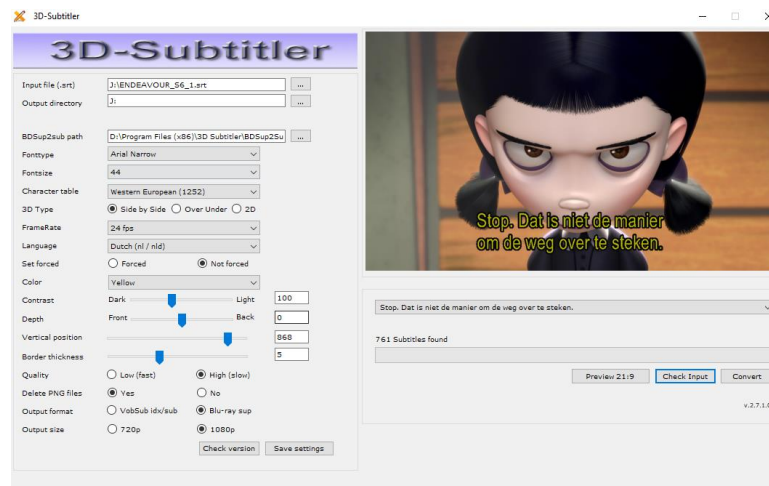
If your movie has subtitles, 3D causes another problem. On both left/right eye images on the SBS frame the same subtitles must be added and they must align properly to look like a single subtitle when both images are shown as a 3D frame by TV or projector.

Fortunately, there is an easy way to handle this. What you do need is a single .srt file that has the subtitles and timings as if it was meant for a regular 2D movie.

For 3D the positioning is very critical. A text based subtitle cannot do it – it is nearly impossible to write the subtitle twice on each half of the SBS image at the precise same relative position.

A tool called 3Dsubtitler takes the subtitle text and reproduces it for both image halves at the right offset. It then makes a transparent picture of the doubled subtitles. This transparency (.sup subtitle format) is then superimposed on the image frame by the muxer and shown if requested. The .sup format is acceptable to BDS.

You need to download this tool from <https://www.videohelp.com/software/3DSubtitler>. The program needs no installation – simply store it in some folder and start the .exe file.



It does require the Java library BDSup2Sub.jar which is part of the zip file in which the application is downloaded. Specify the library location (preferably in the same folder as the 3Dsubtitler tool) in the window shown by 3DSubtitler once you started it. It is then recommended to save the settings. This creates a .config file in the application folder. The library will then always be found automatically.

To create subtitles, follow these steps:

- Open the regular .srt subtitle file and specify the output folder
- Specify the subtitle font type and size (25 or 30 works fine and use a narrow font since it will be widened by a factor of 2 when displayed on stereo tv or projector)
- Specify 3D format for which titles are to be created (SBS or OU),
- Specify height of the picture (HD uses 1080 for SBS, 540 for OU)
- Specify frame rate and font colour, subtitle position (counted from the top: 0 – 1080) and border thickness around the letters to make them readable on light backgrounds
- Ensure the radio button specifies "Not forced" if you want a disc with optional subtitles
- Specify the “depth” of the titles into the picture: default is 0, but a positive value makes the subtitle appear to be in front of the image, a negative value somewhat behind or inside the image.
- Specify output type: Bluray .sup and check “delete .png files” (each title is first made in a .png file, in the end all are collected in a single .sup file and the separate .png files are deleted)
- After some initial checking on proper timings, click “Convert” and create the .sup file
- Import the .sup file as a BDS subtitle file for the 3D movie

2D images within a Flat 3D

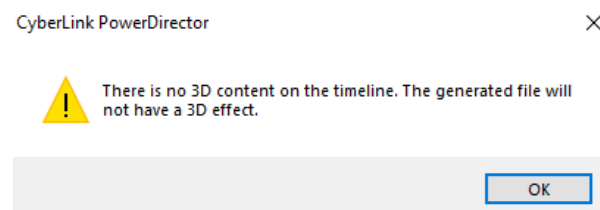
Mixing 2D and SBS 3D movies on a single disc is normally not recommended. The projector or tv set sees a flat image and you need

to manually tell it when it is 2D or SBS 3D. Mixing both types requires manual intervention each time because a 2D movie is scrambled in fake SBS while in 2D mode the 3D SBS is shown flat as two images next to each other.

At times you may need to create a SBS menu that really has identical left and right eye images. It would look flat when shown.

To create such a SBS twice-identical half frames, you can use a capable video editor such as Power Director. You simply add the movie to the timeline and in the “produce” stage select the 3D output, requesting 3D Half SBS output.

This triggers the warning that there is no 3D (source file) content and therefore the result, albeit Half SBS, will look flat.



But that’s exactly what we want: two identical pictures.



The horizontal resolution is half the normal full width 2D movie, but that’s the price to pay if you want to mix 2D and 3D movies with a minimum of TV or projector adjustments after the initial setup.

3D images

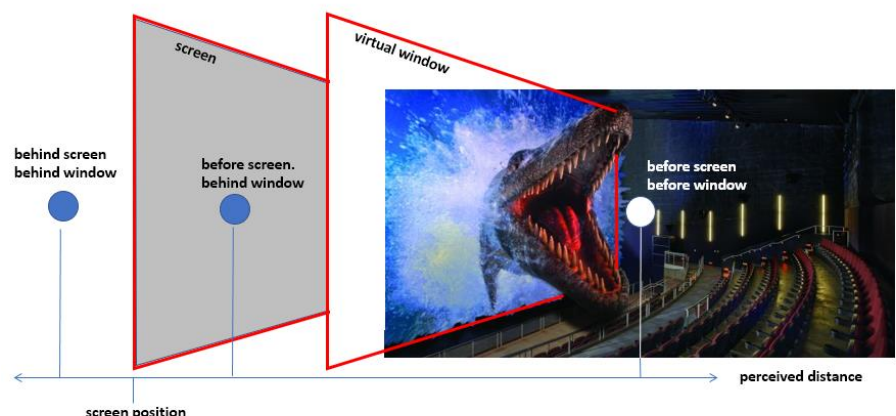
When you create 3D movies with a 3D camera, there is little you can do about the depth effect of the movie. The camera lenses have decided that for you.

It is different for still images taken by a stereo photo camera or with a single lense camera using the "cha-cha-cha" method where you take both left/right images one by one where for the second image you change your position a bit to get the depth parallax effect (of course this method only works on objects that do not move or change colour in between the two pictures taken).

In this section we will talk a bit on how you might change the perceived depth between the left and right eye image.

If you have 3D slides (left/right eye pairs) there are two things you always need to be aware of:

- The virtual window. The entire 3D scene takes place behind the window, as if you're looking through it. This window may be felt/seen positioned before the actual screen position. Anything sticking out of the window into the room is usually considered bad 3D and used only in 3D-effect movies (much like the "ping pong stereo" audio to let you hear a left/right channel – how artificial it may sound). One exception: if the entire object can stick out and is not cut off by the virtual window (i.e. the edges of the image).
- The position of the projection or television screen. The objects for which the left and right eye image overlap precisely, are considered to be positioned at the screen position. Other objects that do not overlap are either positioned behind the screen or before the screen but still behind the virtual window.



An additional consideration is applicable when you have 3D images that are not by themselves of a 16:9 ratio. You need to make sure that they show in their proper ratio (older ones scanned from film role slides are often square or 2:3 (small frame 24x36mm)) despite the fact

the TV screen or projector shows a widescreen 16:9 image. This means that you need to do two things to each stereo pair:

- Align the left and right eye image properly to a good 3D image so when shown viewers do not develop a headache due to misalignment (especially if this applies to the vertical positioning)
- Add pillar box black bars at the left and right side of the image to fill it up to 16:9 format. And block out any information on one image and not the other. This is called creating the “virtual window”. Software like Stereo Photo Maker (SPM) can help you with this.

Some 3D bluray players can perform the scaling themselves. In that case, you only need to ensure that the width is scaled properly. With some mathematics it will be clear that the scaling factor is given by

$$factor = \left(\frac{height\ image\ in\ pixels}{1080} \right)$$

The width is multiplied by the same factor but its size may not equal 1920. The difference is split in two parts – the width of the pillar bars on both sides of the image.

That sounds like a lot of work, but fortunately most of it can be handled by a very handy freeware program running on Windows XP to Windows 10: Stereo Photo Maker (SPM).

A few words on stereographic images

The depth perceived in stereographic images is related to the perspective and difference in viewing by the left and the right eye. This means that if you overlay a stereo picture for the left eye over one for the right eye, they do not precisely overlap.

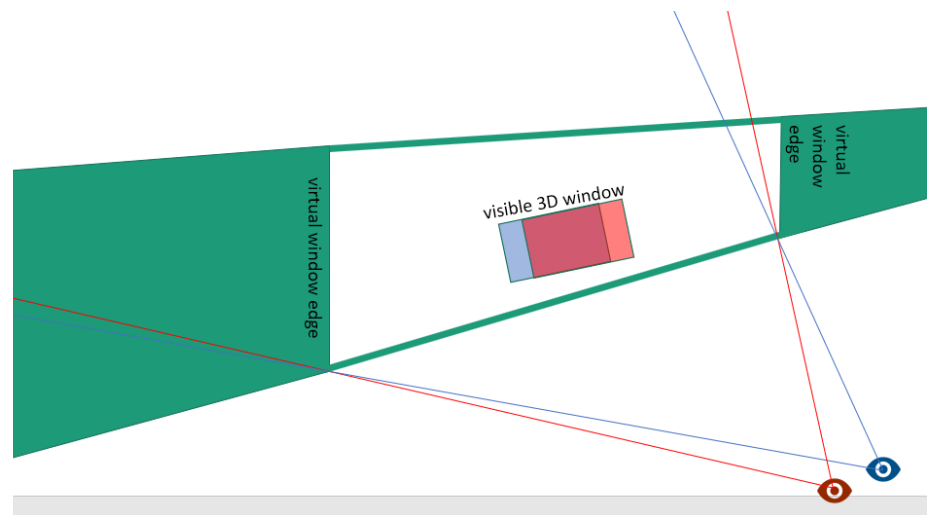
In the next discussion we use an anaglyphic stereoscopic image because it has a clear red coloured left image and a blue coloured right image. The two colours make it easier to discuss the issue regarding “left” (red) eye and “right” (blue) eye images but it is valid for full colour 3D images created with stereoscopic cameras and 3D tv sets or projectors.

Virtual window

The first difference is the fact that we view a scene through a window. Usually this is the edges of the frame of an ordinary photo or slide. The scene happens within the frame. It also does so for 3D images. This window is called the “virtual window”. You observe a scene as if you look through a home window. The entire scene happens behind this window.

For an ordinary 2D picture, the window shows the same picture for both eyes.

In 3D this is not the case. The left eye has a different field of view than the right eye. The left eye can see more of the right side of the scene than the right eye and vice versa.

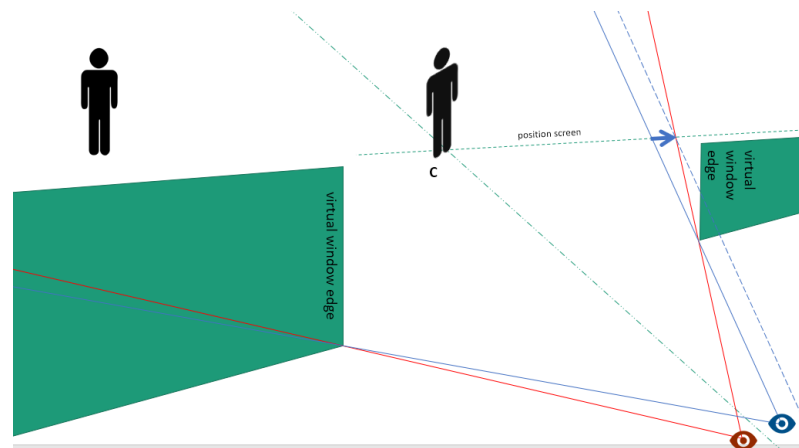


This difference is important for 3D pictures that have the left and right eye images superimposed.

Screen position

Because the eyes see a slightly shifted scene, the pictures do not overlap entirely. Except for one particular object of the scene where both left and right images coincide. Where this happens, the projected object occurs to the viewer to be positioned at the distance where the (tv or projection) screen is at.

The red and blue coloured left/right images of the object overlap. In practice they only do so if you shift either the red or blue image a little bit so the red or blue coloured object images do overlap. This shift is needed since both eyes have a slightly shifted view of the scene as explained earlier with the virtual window.



Which object you want to position “at screen distance” is partially your decision. All other objects are perceived further away or nearer by. The picture above shows a person object marked C at some distance from your eyes, but on line going through the middle between your eyes. Because the red image goes a bit further into the scene than the blue image, one of the needs to shift horizontally to match up.

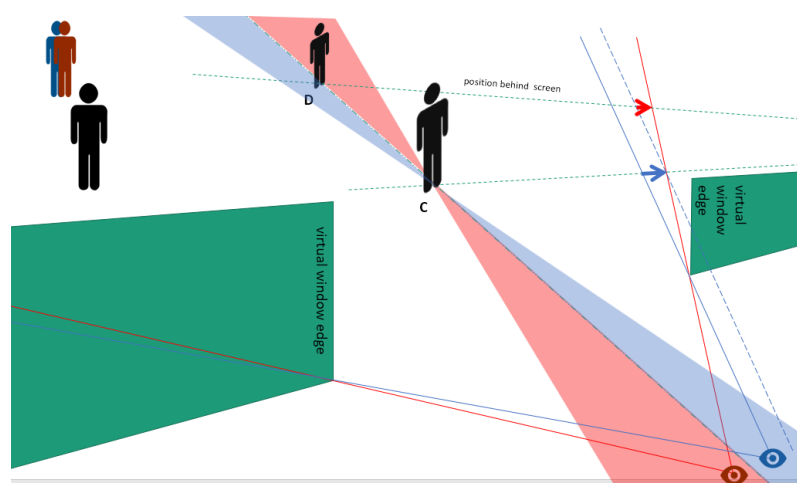
In the picture, we choose to move the blue coloured right eye image to the right to match the red coloured left eye image. You could have

chosen to move the red coloured left eye image just as much to the left side: it's all a matter of taste and relativity.

The icon in the left top corner shows how this object looks in the anaglyphic image if you just look with your naked eye (no coloured glasses).

Objects behind the screen (further away)

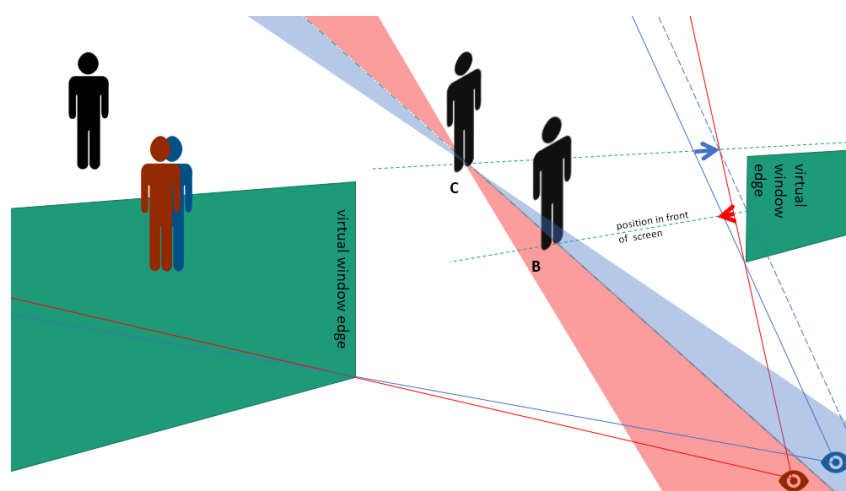
Another object D on the same viewing line but further away is shown in the picture below. As can be seen, it will look smaller in the image,



but also its red image will shift more to the right than its blue image when you shift your images to have object C to be at the screen position. The additional displacement is shown by the red arrow. When you look at the image with the naked eye you see that the distant object is shown twice with the red and blue colours shifted differently. The object D has a blue colour at the left edge of the object.

Objects in front of the screen (but behind the window)

For objects that are closer to the viewer than the object C at the screen position, the situation is reversed. Now the red image is displaced less than the neutral object C as is indicated by the red arrow (pointing in the opposite direction of the blue arrowed "screen shift").



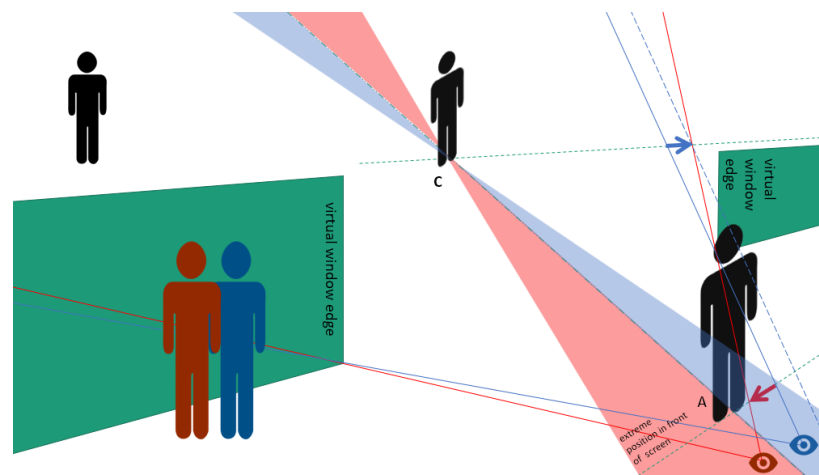
Therefore, the resulting image again shows the nearby object shifted differently in red and blue. When you look at the image with the naked eye you see that the distant object is shown twice with the red and blue colours shifted differently, giving the object B a red colour at its left edge. This is illustrated by the two overlapping objects at the left of the object B.

Often one tries to align the two images in such a way that the screen position C and the before the screen position B coincide. The screen then becomes the virtual window. In 3D movies the subtitles often are at the virtual window position.

Objects in front of the screen (and in front of the window)

The gimmicky 3D slides and movies like to “stick out of the window” and reach or even touch the viewer. This is sometimes permissible if an entire object (an arm, flower, knife) remains within the frame of the window and is not cut off. If it is cut off, it is considered bad 3D as part of the object is hidden by the window but another part protrudes into the room. An unlikely situation.

For those objects that quality, if they stick out of the window, they are extremely far in front of the screen position and as such are an extreme case of the previous situation where object B was in front of object C but still behind the virtual window through which the observer is watching the scene.



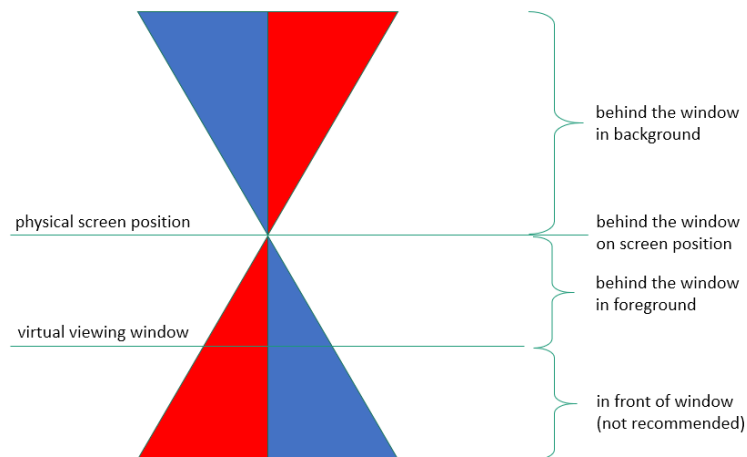
Such an object is A. Like the situation with object B it has a smaller displaced red image but the displacement between red and blue is much larger as is shown by the “naked eye” image at the left of the object A.

Summary of image positioning

The above illustration makes for the following “rule of thumb”:

- Objects at screen position overlap precisely.
- Objects further away do not overlap and have a blue (right eye) left edge

- Objects nearer by do not overlap and have a red (left eye) left edge



This is useful to know when you align and sort out 3D images using Stereo Photo Maker as discussed in the next section. Additional rule of the thumb to avoid viewer headaches is that the difference in displacement (known as disparity) should possibly not exceed $1/30^{\text{th}}$ of the width of the picture. This is called the deviation (of the disparity). That means that most of the images overlap with a small margin in which left and right image are different.

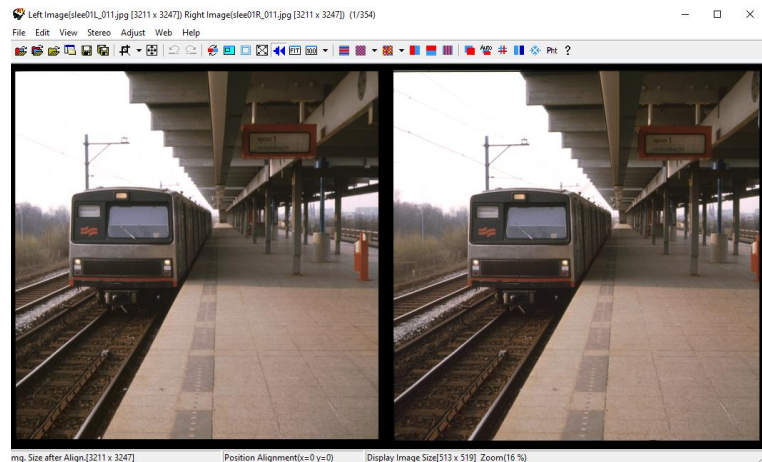
Stereo Photo Maker (SPM)

You can download this freeware program from a Japanese site (but using English – French, German and Japanese are an option) <http://stereo.jpn.org/eng/stphmkr/>. It only runs on Windows (or a Mac with Windows emulation). Its installation is quite simple: you only need to invoke the .exe file. There is no installation setup, no registry entry, no Windows library usage. Quite independent therefore and likely to survive many more Windows versions. At the time of writing (July 2019), it is in its version 5.30 and is actively developed and heavily promoted by the Dutch Association for Stereo Photography (NVvS – Nederlandse Vereniging voor Stereofotografie). Several user guides for SPM can be found by searching the internet for SPM or instruction videos on YouTube.

Adjusting image pairs

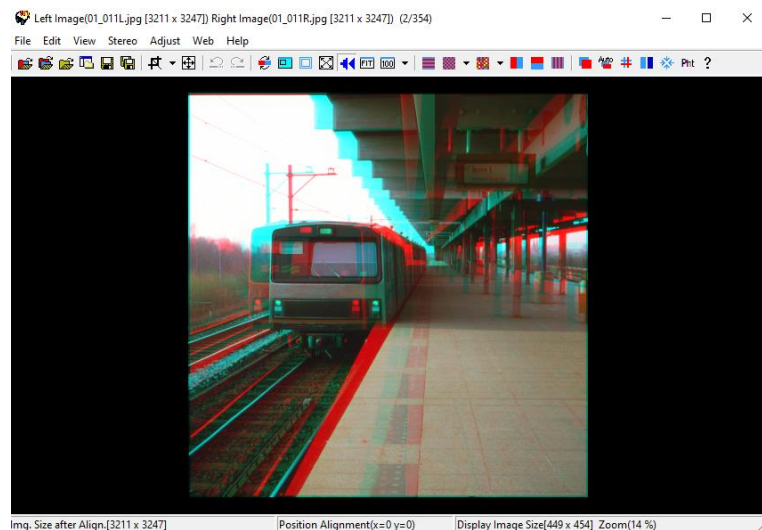
If you haven't done so, it is vital to adjust and align the image pairs. SPM makes it easy:

- Open a set of left/right pair images (if you got a pair in a single file, open that file, otherwise open the left and the right image separately). Each image in this example (the Amsterdam Underground in 1983 at a station above ground) is 3187 x 3223 pixels.

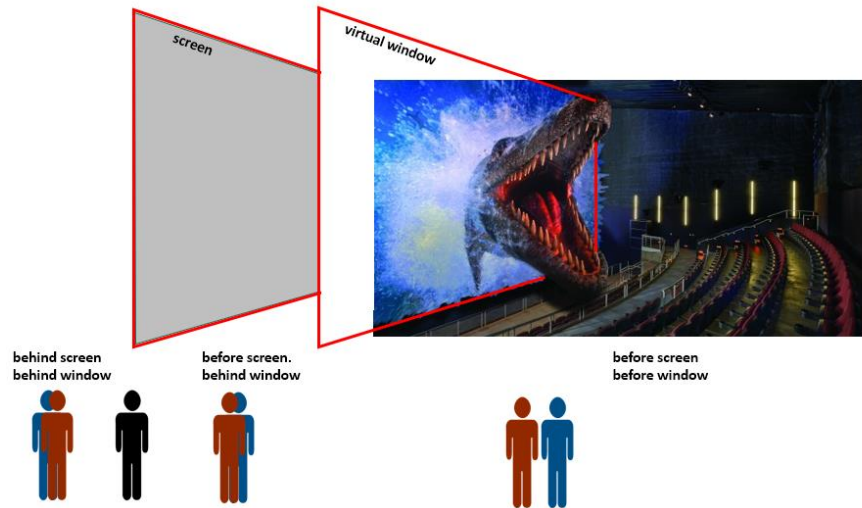


- Perform (vertical) alignment of both images (Adjust > Auto alignment or Adjust > Alignment mode)
- Select a stereo view mode (usually anaglyphic red/blue works best on non-3D pc monitors and most 3D enthusiasts will have multiple red/cyan carton glasses spread around the house)

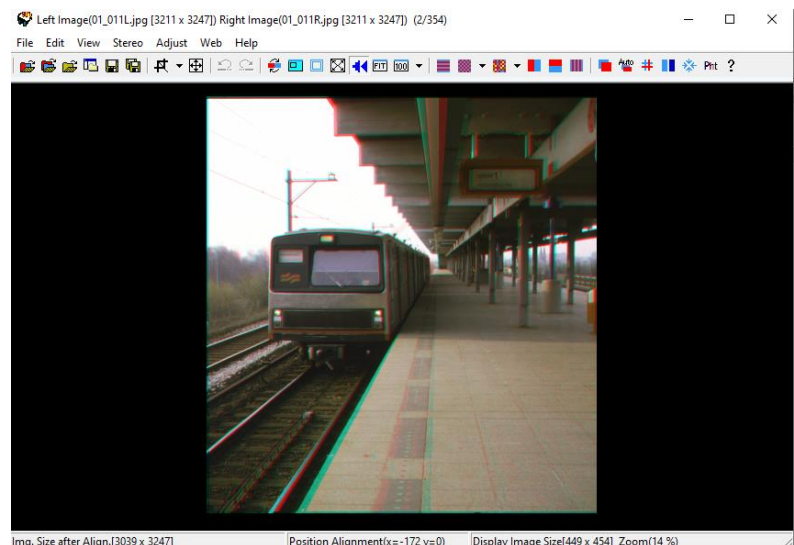
The image below shows a strongly exaggerated superposition of the left (red) and right (blue) image that needs (horizontal) adjusting.



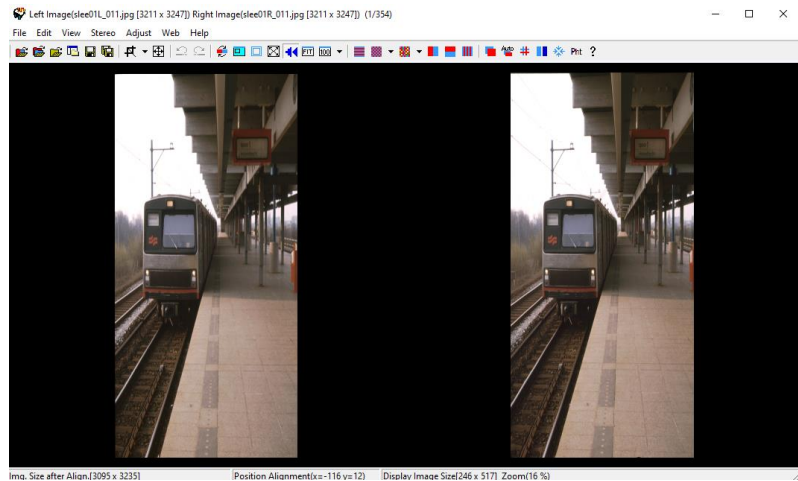
- Adjust the image by using up/down arrows and left/right arrows. You can see how the blue and red images are shifted in relation to each other. Determine which “homological” points (referring to the same physical spot) you want to overlap and determine their position at screen distance.



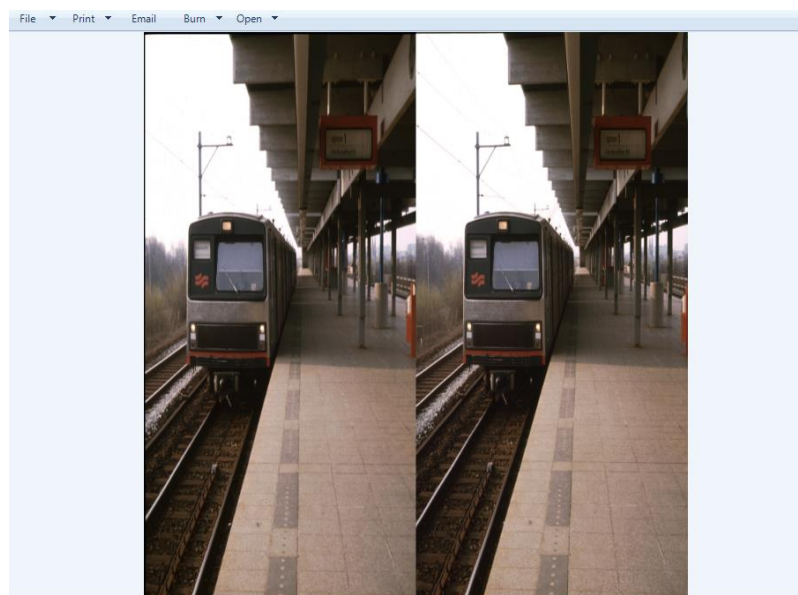
The resulting adjustment places the front on the train at screen distance. Some of the platform is a little closer, the rest further away.



- When adjusted, you may want to switch to stereo view of a half SBS pair if your image does not have the full HD width of 1920 pixels. The way you see it, is the way you want to store it as an SBS paired image file. If the image is square (or anything but 16:9 ratio widescreen) you do need the black pillar bars at left and right side of each image.



- Save the combined image as half SBS pair (the “save” is done in the format you use for viewing) however removes all the black side bars.



Note that the images are positioned next to each other without any separation and not like 2 x 960 pixels wide (total image is 3095 x 3235 pixels) as the initial preview suggested. Projecting this image will not give the proper 3D image. To retain the SBS pair viewing you need to take an extra step.

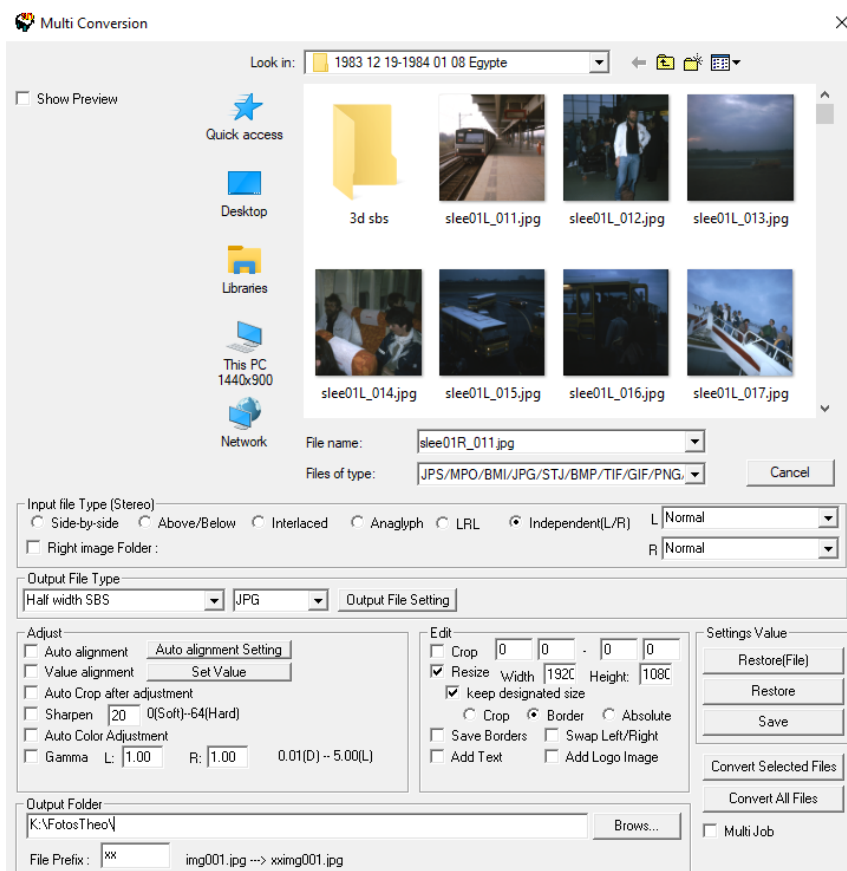
Make non-widescreen images widescreen: pillar box

It is likely that the steps described above are not enough. Before an image is saved as (Half) SBS pair, the images must themselves get aspect ratio. 16:9 and perhaps even resolution 1920x1080 pixels. Just reading in images into SPM doesn't do that as can be seen in the above section.

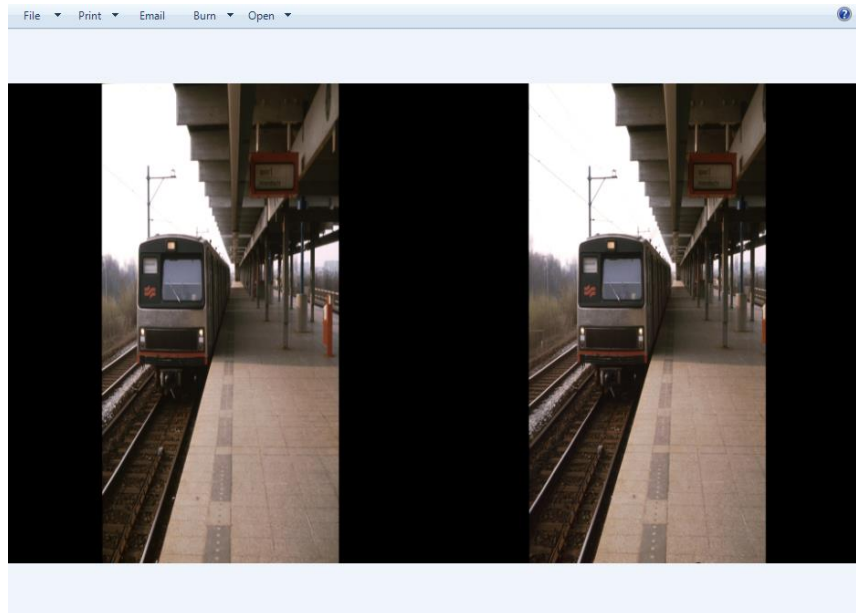
SPM can “convert” the images to the right aspect ratio and/or resolution. For this, before saving the adjusted image pair:

- Select File > Multi Conversion (even if we only do a single image)

- Select both the left and right eye image (keep CTRL pressed doing so). They will show up in the “File name” box.
- On the window form, set the input file type as “Independent (L/R)”
- Set output file type as “Half width SBS” or “SBS”
- Select the wanted file type. Most players accept .jpg. Not all accept .png. You need to experiment here (.png is lossless, .jpg is not)
- In the “Edit” box check the “resize” box and specify width (1920 pixels) and height (1080 pixels). You do not need to use 1920x1080 – any other size is valid too, as long as their ratio remains 16:9 (the player or projector will rescale to full HD)
- Check “Designated size” (that means, either width or height is made full-screen, the other one scaled accordingly)
- Set “Border” (any unused area is made black: the pillar bars)
- Specify the output folder (or if the input folder is used, you may want to specify a prefix to avoid overwriting any original file)
- Press “Convert Selected Files”



- Inspect the resulting output file and ensure that the pillar bars are present if the images are more square than widescreen.



The produced image above has the pillar bars and has size 1920x1080 pixels.

When projected it provides the right 16:9 aspect ratio (shown below in 2D). When projected the 2 images show in quick succession left after right, in sync with the shutterglass opening the corresponding glass for the proper eye.



Use batch mode

The fact that SPM has a File > Convert images (plural) and a button Convert All Files suggests you can make life easier. You can, provided all images are properly aligned. As output use (Half) SBS into a folder, setting the right "Edit" values to make a 16:9 pillar boxed output.

Do one of the following:

- Store all left images in one folder, all right images (same name) in another. Then select all left images and check the "Right image folder". The batch procedure then picks the same named left and right images

- If you already made SBS images in previous steps, (but not properly spaced), store these in a separate folder. Check “Side by Side” as input file type and select that SBS folder.

Finally, press “Convert all files” and sit back until it is finished.

You can store all the resulting images (.jpg) on a disc and play it as slideshow on any bluray player as long as your tv set or video projector is set to 3D Half SBS.

Use BDS as slideshow

If you want to create a bluray disc using BDS, you can do one of two things, in both cases you use BDS to produce a normal 2D movie disc with slides that are SBS formatted (have two almost identical images):

- With BDS MX you can add all images (.bmp, .jpg and .png) into a slideshow. This slideshow may be the FirstPlay option if you don’t want any menus (that are 2D with slides in 3D)
- With other versions of BDS you use a video editor and put all images (usually .bmp, .jpg, .png, .tif) in a movie. You must specify the length each image is displayed. You can add some background music if wanted. When imported as demuxed movie in BDS you may add chapters for each image.

Finally, you mux the resulting disc. When played in a bluray player it may recognize the SBS setting itself or you may need to set 3D explicitly on the 3D TV or projector.

Part 8: The Rest

What's more...

This chapter is just a short list of things you may also do to change the behaviour of the disc. Little things not worth spending a separate project on but still something you may wonder about. Or by reading the text below inspires you. The topics are not in any particular order. Here goes.

Recover projects

Ok, so you fouled up somehow and the project has become corrupt and won't restart. Not all hope is lost: look in the hidden project's _History folder and see that lots of project.bdmd files are saved there. BDS makes a backup copy there each time an implicit save action (when a mux or Jar rebuild is performed) is performed. If you're in luck one of those older project files can restore your health – even if it sets you back in time a few days. (if you don't see this folder, make the hidden folders visible through File Explorer > View menu and check "show hidden folders" in the Options menu item).

These project files are in fact XML files that contain the entire project structure, its menus and actions and scripts. (See Appendix J The Project file (bdmd file) on page 529 for details on the project file). Where a reference is made to an external file (a menu .png file, a movie .264 file and such) that file has to exist or on opening BDS will ask you where that file is located.

Use template projects

A template project is created from a new or existing project that is saved as a template (File>Export template). It is used as basis for a new project (File>Import from template – do not use the "Load" button on the New Project wizard).

Export copies all menu and button objects of a project in the template project folders. In this respect it is no different from an ordinary project.

Import from a project, to quick start a new project, copies all template project information to the new project.

Templates are useful if you create several bluray discs (projects) that together form a group. This can be a complete season of a series, a set of documentaries or other common features.

Alternatively, in case of series, it may be even quicker and more consistent if you rename the project file (by default called project.bdmd) to something like "Season 1.bdmd" and then duplicate this as "Season2.bdmd". Of course you need to replace all movie file links and may have to change some of the menu texts, but all of the navigation, menu movies, animations and other stuff you defined for the "Season 1" will now also be identical under "Season 2".

Some things to consider using template projects:

- Store only common information in a project (button shapes, common menus, identical intro movies), not movies (they are large and likely only used in one project. Templates point back to the original movie files). This may require some pruning once a template project is made. Simply open that project and remove unnecessary menu, movie and other placeholders and the physical files they point to. A template is a “foundation” of your new project with lots of items already defined.
- For a single, generic, item (such as a popup Timeline menu) it may be more useful to have a small project with just that item and merge it into your own project as a single “add on”.

Use Action Matrix for quick copy/paste of actions

Filling in the properties of a menu or button can be time consuming. Especially if many properties are the same: same multi-action, same switch, same...

For movies there is the “Copy properties to all”. Check the properties to copy to other movies.

Enter “Action Matrix” (activated by pressing on “F8” on the keyboard or menu View > Action Matrix). It was mentioned shortly in section “Action Matrix” on page 116. However, if you open it, it allows you to select a menu and copy/paste actions (also the “complex” actions like multi-action and switch) between items on the menu. In fact, when you select a menu, the Action Matrix shows 3 tabs revealing different aspects of the menu:

Button	Right	Up	Down	Enter	Highlight
Audio live		Button: Main menu	Button: Audio music	Set audio: Australia movies [1] (eng)	if Audio in Australia movies = 1
animations					
sounds					
Audio music		Button: Audio live	Button: Sub English	Set audio: Australia movies [2] (eng)	if Audio in Australia movies = 2
animations					

- Buttons: actions for their “Left”, “Press ENTER”, “Highlight” and other properties
- Enter Animation/action
- Remote control: all button in the “remote control” section of properties such as popup, 1-9 marked buttons, coloured ones.

Define an action for one object (just as you would for a single property through menu choices) and copy/paste it to all others with the same action. Or switch menu and paste actions there. Effectively you transfer actions between menus this way.

Selecting a movie, provides only the “remote control” tab but reveals movie-specific actions such as End Action, Popup Menu, 1-9 marked and coloured buttons.


Use Action Matrix for quick check

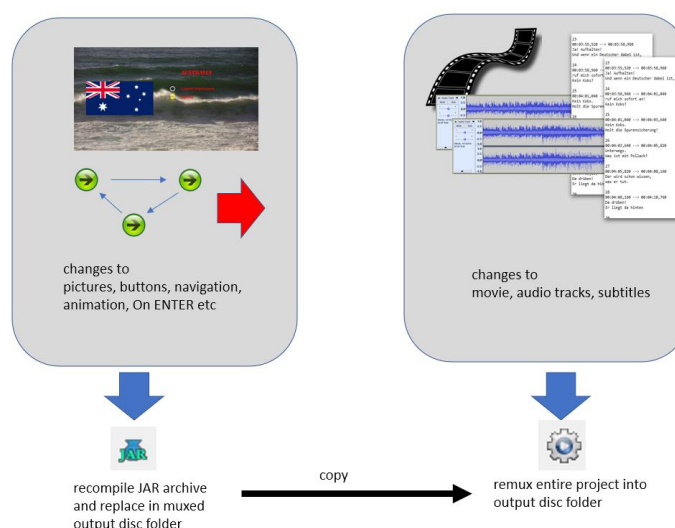
As an additional way of ensuring all properties have been defined correctly, use the Action Matrix view (press “F8”). It shows all button actions for all buttons on a menu in one quick overview. Do the navigation properties (left/right/up/down) have the right actions? Are all Press Enter properties assigned with a menu or movie? Are the movies defined with their Popup Menu and End Action defined?

No need to inspect the button properties one-by-one. Huge time saver. And if something was forgotten: add the action the usual way through dropdown menu list choices or copy/paste an action from another button.

Rebuild a project

When changes need to be made, it is not always necessary to rebuild the entire project.

- Menu navigation is stored in a Java archive (JAR). Simply press the  button to recompile this. It is stored in the BDMV\JAR folder of the disc file output and will be replaced. The information also includes the button links to references of any assets such as video or audio files (that should not be renamed or replaced). The JAR will be signed if so indicated (Project>Project Properties>General tab “Sign the JAR”). The log output refers to these as “Step 1” to “Step 4”
- Any file change, index change, replacement or rename, different Start/Popup Action requires that the video and audio assets are remuxed. The result is stored in the \BMDV\STREAM folder. Because of the change of the assets, also the Java Archive with navigational information is rebuilt. The log file refers to this phase as “Muxing <file>”



Multiple discs of a series

Do you need a number of discs to span a collection of movies that belong together? It usually pays off to give them all an identical

looking menu interface. This is achieved by duplicating the first project for all other discs of the series.

Create the project folder with a \films subfolder. Populate this folder with all demuxed movies and subtitles and menu movie – even if it far exceeds a single disc capacity.

Create the first BDS project for the series. This creates a project.bdmd file in the project folder. Close BDS and rename it to “disc 1” and reopen this project (click on the “disc 1.bdmd” file)⁵⁶. Complete the project with all required fonts, buttons, movies etc. It may be useful to have the disc title as a BDS text box (See section Static menu objects (text, image or rectangle) on page 92 for details). After checking for correctness, build a bluray disc in a newly created project output folder \disc1. Close BDS.

Copy “disc1.bdmd” and name the copy “disc2.bdmd”. Open the project disc2.bdmd. Of course this is identical to your first disc. We need to modify it to become a proper second disc. Remove the video/audio/subtitle assignments of the movie placeholders and renew them with the next series of episode titles. The menu title text box content may be altered to reflect “disc 2” instead of the original “disc 1”. Because menu movie, button navigation and assignments remain unaltered, the “disc 2” behaves exactly as “disc 1”. After completion, build a disc in a newly created output folder \disc2.

Repeat these steps for all other discs of the series.

(No) Bonus movies

Something specific to bluray “BD-J Live” (a marketing name for BD Profile 2) is the addition of bonus movies. Some details are given on Wikipedia (<https://en.wikipedia.org/wiki/BD-J>). As its name “BD-J” suggests, it involves some heavy Java coding. The BDA intended “bonus” to include internet access and playing games over the internet – internet connection is not supported in BDS.

Although “Bonus” is mentioned in the project wizard, they are simply movies in the “Movies” section of the Project Tree.

“Bonus” to BDS is simply some more movies in the “movies” branch of the Project Tree. A menu (usually an “Extra” or “Specials” menu) can then give access to these movies just like any (main) menu can give access to the (main) movies. BDS MX does support the PIP (picture in picture) feature where a bonus movie can tag along with the main movie.

Button actions order

See the online help for “Setting Actions for buttons”.

⁵⁶ You can also use File > Save As to give it a new name. The old “project.bdmd” file then still remains but could be deleted afterwards

By default, when you press the direction or OK (Enter) button on the remote-control sequence is as follows:

1. show the Activated state (if exist)
2. show animation (if exist)
3. execute the Press ENTER action
4. update the Current state

This is a *standard* sequence. But for some action may be necessary to update the current state before the animation. In this case, you need to use the other two modes, “between” and “before”.

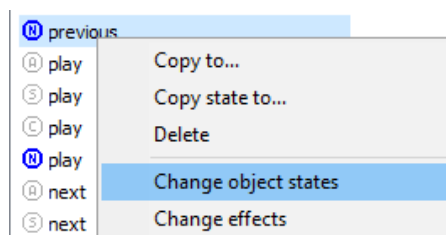
Remote-control buttons for interaction

The Menu/popup/movie/playlist properties contain a section called “Remote-control buttons”. It gives the ability to specify actions for remote-control buttons. Assigning actions for Play/Pause/Next track/Prev track/Fast forward/Rewind buttons changes the disc type into “Interactive” (intended to use with multimedia discs). In this mode some bluray features on some players may work “strange”. For example, PSR 5 register used for the current movie chapter always has -1 value.

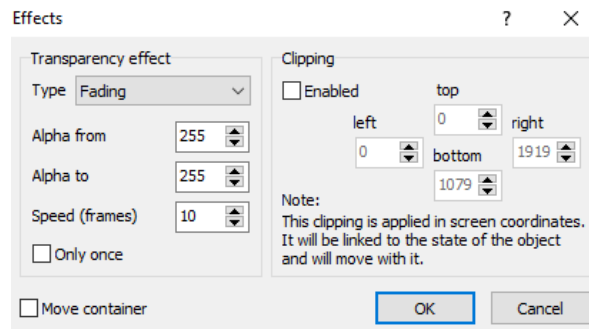
Change contents of menu item at design time

Menus and button images are changed by editing their Photoshop or .png files.

But for menu objects added within BDS (see section “Menus made within BDS” on page 90) you can change these by selecting the object (in Objects window or Designer window) and via right-mouse click select either Change object states or Change effects.



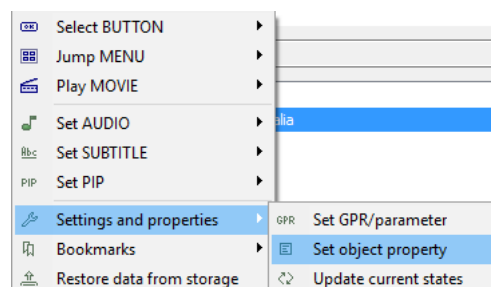
- change object states (edit text, font change and size, colour). For buttons, this can change for any one of its states.
- change effects (specify transparency (alpha=0 transparent, alpha=255 opaque), fade-in, fade-out or partial fade and clip object to a rectangle with given top left corner and width and height)



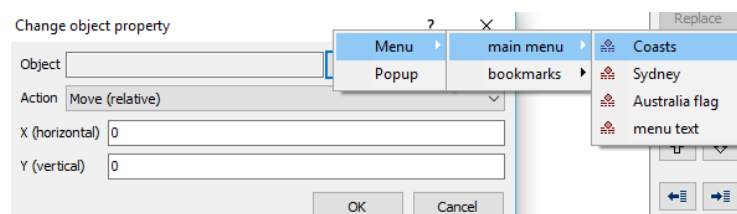
Change contents of menu items on the run

When you want to change the displayed text in a menu, you can change it dynamically during the execution of a (multi-)action. To get it to work:

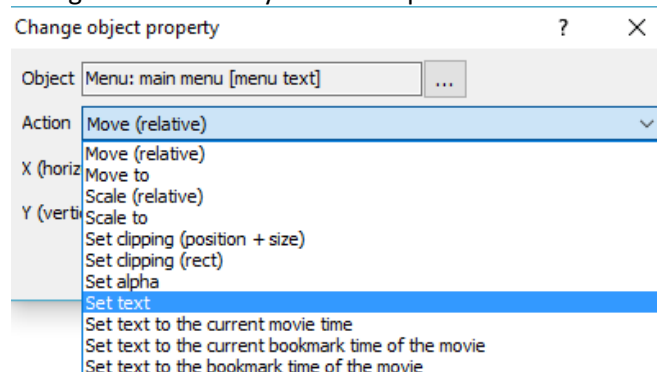
1. Make the menu object that will be modified a separate object (changes apply to an entire object; you cannot replace the 3rd line alone) with its own name.
2. As action use Settings and Properties > Set object property



3. This opens the Change object property window. Specify the name of the object. This can be a menu picture or button



4. Specify the Action field of what to change in the object. Depending on the choice made, the remainder of the window changes to ask for any further required data for such choice.



You can

- move the object (specify X and Y below the selection box),
- change its size (specify X and Y)
- clip it (specify top left corner of the box and its width and height)
- make it more or less transparent (alpha=0 transparent, alpha=255 opaque)
- set its text. This text is
 - free text
 - the current movie playing time (0:00:00 format)
 - the time of a specified bookmark (bookmark must be selected first – that requires this action to become a multi-action that first selects the bookmark (“Bookmark” action) and then uses “Settings and properties” action to copy the bookmark timing as text value)

[bm]: prev Australia

[prop]: [bookmark list] cur bm time Australia

Movies perform actions each second

The Popup/Movie properties has a property “Action every second” that allows an action that is performed every second. It is used for the timeline (the ability to show the progress of the movie, check online Help > FAQ). Selecting a Multi-action feature, several actions can be performed in sequence before starting the movie.

With multi-actions you can also specify Play Movie > <Movie name> - Call Action every second. This executes an action assigned to the “Action every second” for the current movie.

Movies perform action before starting

The Movie has a property “Start action”. When specified, the action is executed when the movie started. Use the Multi-action feature if you want multiple actions to be performed.

Resume a movie

Apart from the regular “resume” feature of a movie on a re-inserted disc, you can also define an action to do this while the disc is playing. If an action specifies Play Movie > <Movie name> -> [resume] this resumes the specified movie playback.

You can reset the resume time using the SWITCH feature with an action set through (right click on action text box) Settings and properties > Reset movie resume.

Disable ability to select a subtitle

When a movie is added that has a subtitle burned in as part of the image, you want to disable the viewer to select another subtitle that will be superimposed on the burned in subtitle (if there are any or even if not – it disables the “subtitle” button on the remote-control).

For this, use the “Start Action” of the movie to specify Set Subtitles > *movie group name* > off.

You may be best off by inserting all movies with burned in subtitles in a group of their own (like “burned in”) so the disabling does not interfere with movies that have no burned in subtitles and should allow the viewer to select any subtitle track available.

Movies in a playlist but not on the menu

A movie can remain unknown if you don’t list it as item on a menu list. However, it remains available to the disc author in the list of movies provided by e.g. the “Press Enter” action of a button will show “Play Movie” as an option and list all movies.

If you do not want a movie that is part of a playlist to also occur in this movie list, then check the “Virtual movie” in the movie’s properties. When checked, the movie can only be part of a play list. It cannot be selected from the “Press Enter” action “Play Movie” list.

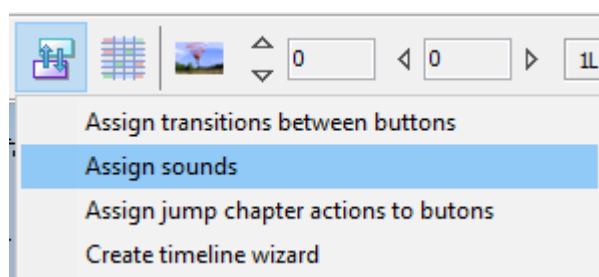
Chapter number via remote-control

Normally you go to a chapter by repeatedly pressing the “next chapter” button or by selecting a chapter from a special chapter (popup) menu. However, if you check the box “Numeric select scenes” from the movie properties, pressing 0-9 keys on the remote while the movie plays, jumps to the chapter number entered. E.g. pressing “1” and “4” jumps to chapter 14.

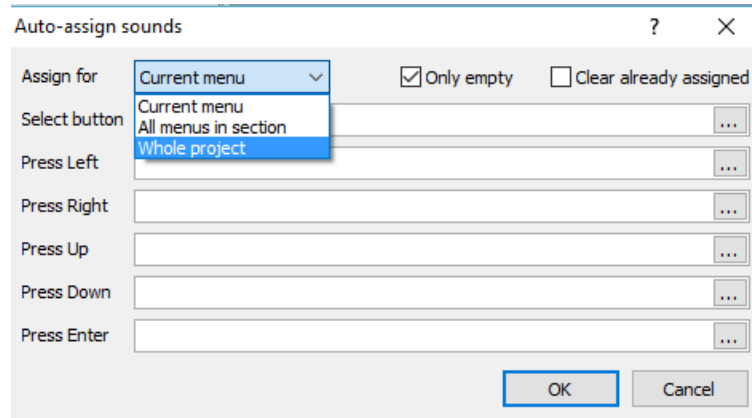
Navigation with sound

Similar to setting standard navigation properties to buttons, you can also specify a sound that is played when you move from one button to the next.

For this to work, you can use the “Auto Assign Sounds” choice under the “Auto Assignment” menu button.

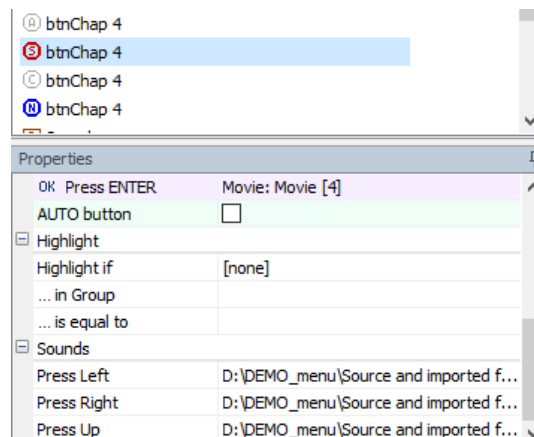


The next window that opens allows you to specify sounds to be played when a button on the remote-control is pressed. The setting can apply to all buttons or just the ones on the currently selected menu (and if it applies to all or only those buttons that have no sound assignment yet).



The audio file must be acceptable (regular CD format WAV/LPCM 44.1 kHz is not acceptable.) and once a project is built, you must check if your audio file was accepted.

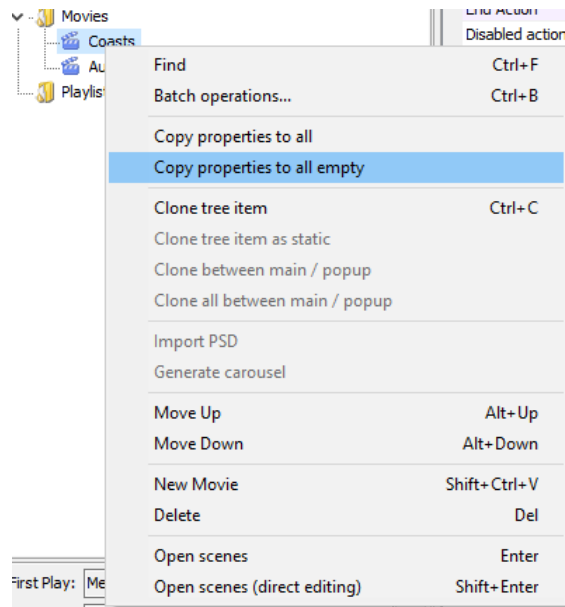
You can overrule a general setting on a per-button basis. This can also be used to specify sounds to only some buttons. Open the button properties in the Object View and scroll to the “Sounds” section of the button properties and fill in the location of the specific sound files.



Copying properties between movies

Any film clip to show, must be a “movie” entry in the Project Tree view. When you insert multiple movies, you need to set all their properties. Again, if some properties are the same between movies, BDS helps you by copying information.

Select a movie with the appropriate settings, right click on it in the Project Tree window to show its popup menu.



“Copy properties to all” is useful to copy the same properties to all other movie entries regardless overwriting whatever is specified (including any specified movie, audio and subtitle tracks).

“Copy properties to all empty” is more subtle: it fills only the empty properties of the movies, leaving any set property (such as what movie to play) alone.

Both copy methods may be useful. Be careful though: it may also copy properties you do not want copied (or left blank) for certain movies. Like Action (switch or multi-action) properties that are normally different for each movie. Leave the “copy properties” operation to last when all movie-specific properties have been filled in manually.

Duplicating a menu as static

You have a menu with picture and button objects. Now you need an identical menu but with other buttons. The objects of the old menu should just be pictures.

Select the menu in the Project Tree, right click on it, and select the “Clone tree item as static”. A new name for the cloned window is asked for. In this new menu only pictorial objects are shown. All buttons have been changed into pictures using their “normal” state representations. They inherited their names from the button names.

In the cloned menu you may want to represent the selected button of the previous menu as if it is still selected. This requires you replace the “normal” state image of the cloned button by its “selected” image. Ensure that the normal and selected images align (their “left,top” properties) with their previous menu counterparts. Otherwise some movement of button images between menus can be seen (a “cha cha cha” movement).

An example of a menu with buttons (left) and its clone (middle) is shown below. For the cloned menu that opens when the “19” button is selected, the static image of the “19” button must be replaced by its “selected” image rather than the “normal” one.



Active menu, "19" button is currently selected



Clone of menu, all buttons are static images using the "normal" image



Modify clone to use the "selected" image for the "19" button

Duplicating a (popup) menu live

There are times you need to spend a lot of time on creating the right menu with all buttons, actions, pictures. And then you need an almost identical menu. This happens frequently with popup menus that belong to a specific movie, but other movies have identical popup menus except that references to movie names are different.

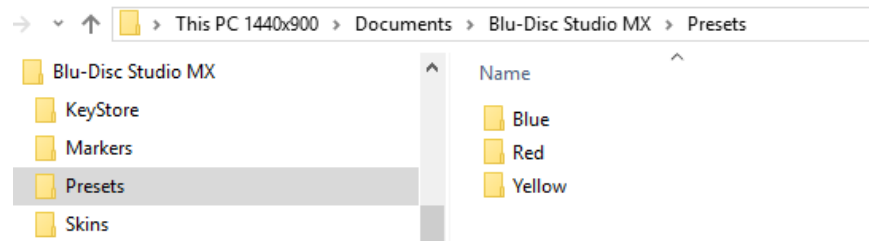
You can select a menu and via right mouse click copy an entire menu by "Clone tree item". Rename the clone to a descriptive name and edit the actions that need to be modified for the movie name.

Changing Chapter frames

If you use menus showing images of chapters in your movies, you can modify the frames in which the images are displayed. It is called "a new preset".

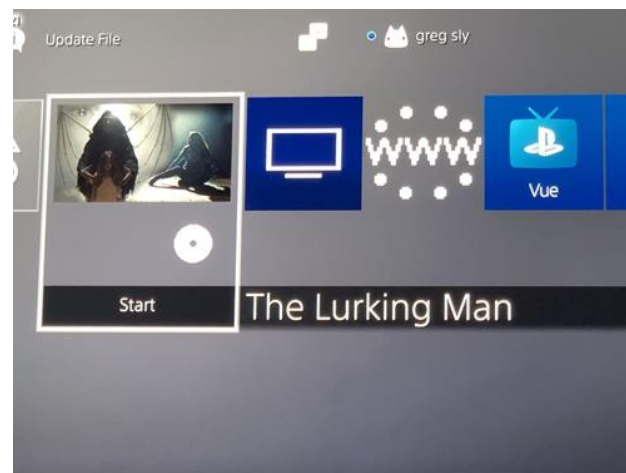
For this you need to place those images in your local Documents\Blue Disc Studio MX\Presets folder. If you modify existing BDS frames, first copy those from their installation folder (\Program Files(x86)\Blue Disc Studio MX\presets) to your local Documents\Blue Disc Studio MX\Presets folder.

Modify frames and frame folder names there. They will then show up in BDS dropdown menus as frames to choose from. The figure below shows a new preset called "Yellow" (the name of its preset subfolder). Also see section "Customizing or making your own chapter menu preset" on page 180.



Thumbnail of disc in Playstation 4 and some BR players

If you insert a disc in a Play Station or some bluray players, it shows the disc with a small thumbnail. That is – if the disc has something specific to Play Station to show it. Like the image below for a disc with a movie “The Lurking Man”.



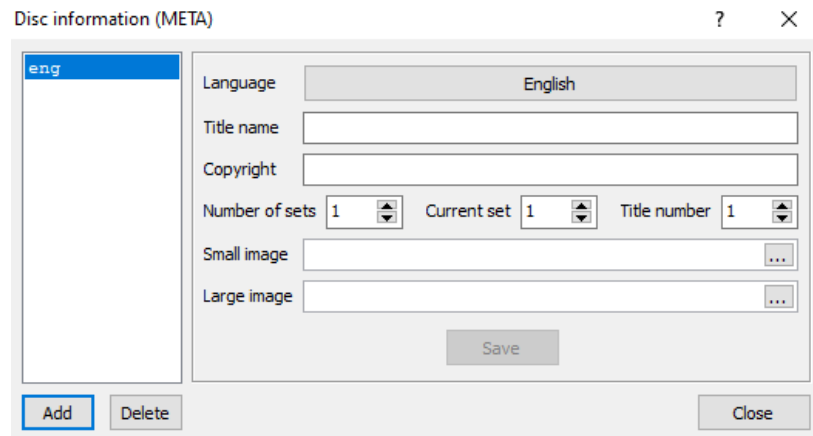
Some extravagantly priced software has this feature built in (Scenarist)) but for mere mortals no product I know of has it. For those who want to add this Play Station thumbnail feature, read on.

There are two ways of doing it – either manually or via BDS. The manual work is erased (as \Output is rewritten) on subsequent muxing efforts. The BDS generated way therefore is preferred.

BDS generated

Open the Projects > Disc Information (META) data menu item.

Initially it will open a blank window. You need to add at least a single language (by which the disc is made – such as “en” for English). This selection is made by clicking on “Add”. Another window opens with all languages selectable. Take one and click “OK”. That returns to the “Disc Information” window – now filled in.

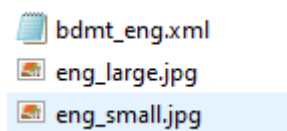


You can add a “small image” and a “large image” in .jpg format; one which is used as the thumbnail on the PlayStation display. The size doesn’t matter – BDS will rescale them into 630x360 pixels and 416x240 pixels. However if the original does not have the same aspect ratio (1:1.75) the generated images may look squeezed or stretched.

Note: The JPEG file must have a JFIF header. It appears Photoshop does not add this to its files, Corel Photopaint and freeware IrfanView do.

In the end click “Save”. When you now remux the project, a new folder in the \Output\BDMV\META folder appears:

\Output\BDMV\META\DL\ with several files: an eng_large.jpg and eng_small.jpg file (the “en” bit comes from the selected language) and an xml file bdmt_en.xml



The xml file has the generated content read by the PlayStation:

```
<?xml version="1.0" encoding="utf-8"?>
<disclib xmlns="urn:BDA:bdmv;disclib">
  <di:discinfo xmlns:di="urn:BDA:bdmv;discinfo">
    <di:title>
      <di:name>film title A</di:name>
      <di:numSets>1</di:numSets>
      <di:setNumber>1</di:setNumber>
    </di:title>
    <di:description>
      <di:tableOfContents>
        <di:titleName titleNumber="1">
          film title</di:titleName>
        </di:tableOfContents>
      <thumbnail xmlns="urn:BDA:bdmv;discinfo"
        href="eng_small.jpg" size="416x240"/>
      <thumbnail xmlns="urn:BDA:bdmv;discinfo"
        href="eng_large.jpg" size="640x360"/>
    </di:description>
    <di:language>eng</di:language>
    <di:rights>copyright statement</di:rights>
  </di:discinfo>
</disclib>
```

Manually

1. Goto the BDS project folder \Output\BDMV\META that BDS creates.
2. create a Directory DL
3. in it you put the thumbnail images in .png or .jpeg format... just make sure you use the same name in XM to create next. The requirement is that the images have a fixed size: one of 640x360 pixels, one of 416x240 pixels.
4. Create an XML file *exactly* as below (but change name of movie and pictures to your own movie and pictures):

```
<?xml version="1.0" encoding="utf-8"?>
<disclib xmlns="urn:BDA:bdmv;disclib"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:BDA:bdmv;disclib disclib.xsd">
<di:discinfo xmlns:di="urn:BDA:bdmv;discinfo">
<di:title>
<di:name>movie title</di:name>
</di:title>
<di:description>
<di:thumbnail href="./picture-filename.png" size="640x360"
/>
<di:thumbnail href="./other-picture-filename.png"
size="416x240" />
</di:description>
</di:discinfo>
</disclib>
```

5. Call the xml file bdmt_eng.xml and add it to the BDMV/META directory

Although the xml files generated or manually made seem to differ, they do not. The generated manner has a set way of naming files, but as long as the right file names are used in the xml file, it will work.

A final word

In the past chapters and projects we showed you how

- To make a basic disc with movies
- To add a menu to it
- To add submenus and popup menus for further details
- To give buttons a “current” state to remember your last setting of the item
- To use the program GPR registers and read the players PSR registers
- To have multiple actions occur when a menu item is selected
- To select a particular action, depending on current conditions. This includes episode watching and resume playing a disc
- To provide animation to menu objects
- To add background animations
- Create true .mvc based 3D discs and flat SBS 3D discs

It’s your imagination in combining these options and put them to use in any way you fancy, that is the limit in what you can achieve using BDS for authoring bluray discs.

It may have a bit of a learning curve to become familiar with it, but I hope this curve has been overcome after reading this guide.

Enjoy using the program. And for any remaining questions on “how do I do this” or “I cannot quite figure out how this feature works”: an email to the support staff of BDS will quickly help you on your way.

What we did not cover

Some items are not often used and hence became “out of scope” topics. Not all features possible for a certain action or item have been discussed – these are for you to explore when you become more familiar with BDS.

Some topics are not covered at all because they seem rather obscure to most users. The online help of BDS provides some basic information on these topics.

To these topics belong:

- Bonus items – see online help “Bonus Videos”. They are nothing special – just movies in the Project Tree view. They can be made accessible by providing links (buttons) on a menu. Some BR players have “Bonus View” buttons on their remote controls. They don’t seem supported by BDS.
- Much of Java programming and scripting (could fill a whole book)
- Merging projects (it’s mentioned, but simple enough to explore yourself – useful to have a few incomplete projects with reusable items that you can merge into your project. Templates can do similar things but are more a project foundation rather than a single “add on” through merge)

- Some intricate project or options settings or dropdown list options in several BDS menu options. You need to consult the online help for this or simply experiment (by lack of a technical manual on all of the product's nooks and crannies)

Appendices

Appendix A: The user's guide source files

No better way to learn how to use BDS to make bluray discs than by doing all steps yourself. The chapters describing different projects make use of two small (2 minutes running time) movie files and their menus and subtitles.

These files can be downloaded from <https://blu-disc.net/downloads> in a single .zip file to provide you with all the objects you need to create your own project.

Source files

All source files needed to recreate a project are found in the top folder \Sources:

- \Sources\Additional Stuff – not needed in projects but supplied as extras to experiment with (some fonts, buttons, chapter frames and movies)
- \Sources\movies – the movies used in one or more projects.
There are 3 subfolders:
 - \Sources\movies\combined – all movies in .m2ts or .ts format with audio, subtitles and video combined in one file
 - \Sources\movies\demuxed – separate video .264 and audio .ac3 streams
 - \Sources\movies\subtitles – subtitle files used for the movies

Whenever you recreate a project, files needed can be copied from the \Sources subfolders. If you decide to take the combined movies, you do need to demux them yourself into proper streams. The demuxer must deliver BDA compliant streams as otherwise BDS cannot create a disc image that is guaranteed to play on set top players.

To run a project, you:

1. create the project either:
 - a. using the Project Wizard, skipping all steps until "Finish"
 - b. copy an existing project folder tree as a new project folder tree (as indicated in the text)
2. In the created project folder, create additional folders such as \films, \original sources and \buttons as indicated in the text describing the project.

Ready-to-run

Complete projects are stored underneath \Projects. Each project is a subfolder of \Projects. In order to run the project, you need to:

1. Copy the \Projects\<project> folder tree to the default folder you defined to hold the BDS projects.
2. Copy the necessary files from \Sources\Movies\demuxed into the project folder \<project>\films

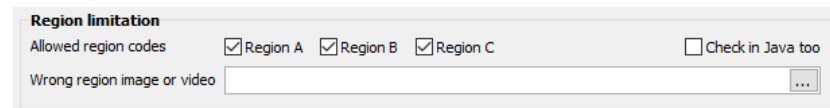
3. Open the project (or double click on the project.bdmd file in the project folder)
4. Mux the sources to a final disc in \<project>\Output folder
5. Experiment

Appendix B: Commercial settings

The following indicates what features are available for those that want to produce commercial blurays and may feel the need to restrict the use of the disc in certain areas or to add some UOP's (User Operations Prohibition).

Regional settings

In the Project > Project Settings > General tab you may indicate that your disc is only playable in certain regions as defined by the BDA (see Appendix C: Bluray specifications).

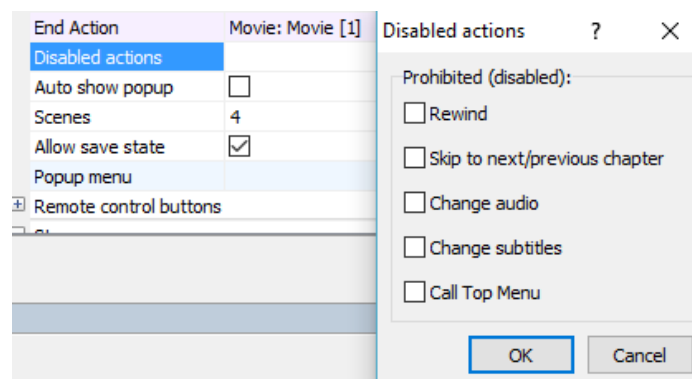


An image can be shown or (silent) video played when the set top player has the wrong region defined.

Check the “Check in Java too” box if you want a runtime check of the regional setting instead of just the hardware player check upon insert of the disc.

Disable user operations

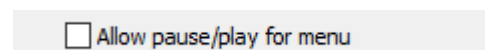
For each movie entry you can specify what actions a viewer may not perform: the so-called User Operations Prohibited or UOP. It is the “Disable” property of each movie. When clicked on its “>” button, a window with possible disable features pops up.



Simply check those actions that should be disabled for that movie. To avoid viewers to jump to the main menu, the “Call Top Menu” should be disabled.

These UOP limitations can be applied to each movie and each playlist in the project.

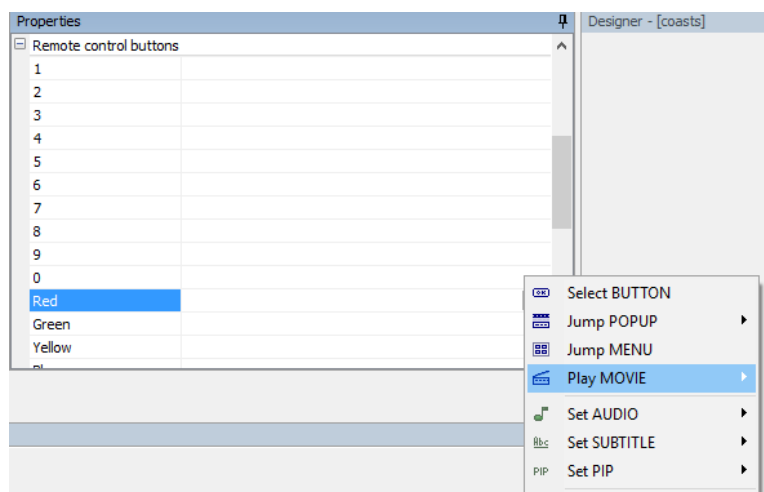
A special ability of BDS discs is that the user can be allowed to pause the menu movie. By default he cannot pause the movie (which is compliant with commercial discs) but you can set the checkbox “Allow pause/play for menu” in Project > Project Settings > Menu tab.



Remote-control buttons

The properties window of each movie contains a section “Remote-control buttons” where you can specify specific actions to take when the viewer presses the common coloured buttons, numeric buttons, arrow buttons and some more.

A menu is presented, identical to the one used to specify Popup Action or Start Action (show menu, play movie, set subtitle etc) and you can make a choice here of what pressing the specific button will do.



Resume

Discs when stopped before playing ends, can be resumed. The set top player can remember the position of a limited number of recently played discs. See Project 7: Restartable disc (Restore/Resume) on page 225.

Signing Java JAR

The menus and buttons are saved as a Java program (BDS uses BD-J for its menus, a program stored in a Java archive JAR file). This program can be signed in order to ensure it is not tampered with afterwards as well as that it is (irreputably) known who created it. Signing is also required when you want to allow “Resume” operations since the position where you stopped playing must be written in the player’s storage. This is only allowed for signed JARs.

The information needed to sign must be filled in as part of the project. Use the Tools > JAR Signing option for this and fill in the window presented. This window is also shown by clicking on the button shown at Project > Project Properties > General tab button “sign settings”.

JAR signing settings

Keystore password: [password] On compile: Check only Re-create keystore

confirmation: [password]

Use only latin characters and digits

Root certificate

Password: [password]

confirmation: [password]

Common name: Movie Studio

Email: mail@moviestudio.com

Organization: Movie Company

Organization ID: 7fff1111 ☒ Use

Organization unit: Department

Location (city): City

State/province: State

Country: US

Application certificate

Password: [password]

confirmation: [password]

Common name: BD Studio

Email: mail@bdstudio.com

Organization: BD Company

Organization ID: 7fff2222

Organization unit: Department

Location (city): City

State/province: State

Country: US

☐ Show detailed log of the signing process

OK Cancel

Specify passwords and organisation ID that apply to you. The bullets shown use default passwords as indicated in the help file if you look up “JAR signing settings”. For home users for whom organization IDs are unimportant yet needed, modify the default Organization ID values by changing the last digit into something else to avoid the muxer warning message that you use the default IDs used by BDS.

The disc generation process will then show some of the signing information used in creating the disc by generating root certificates and application certificates.

```
11:42:54 - Starting sign process...
11:42:54 - OrgID: , AppID: 4000
11:42:54 - Generating root certificate...
11:42:54 - Root DN: CN=Theo de Klerk, O=private, OU=home, L=Bussum, C=the Netherlands
11:42:54 - Generating app certificate...
11:42:54 - App DN: CN=myself, O=home, OU=Department, L=City, ST=State, C=US
11:42:55 - Signing...
11:42:56 - Sign process finished

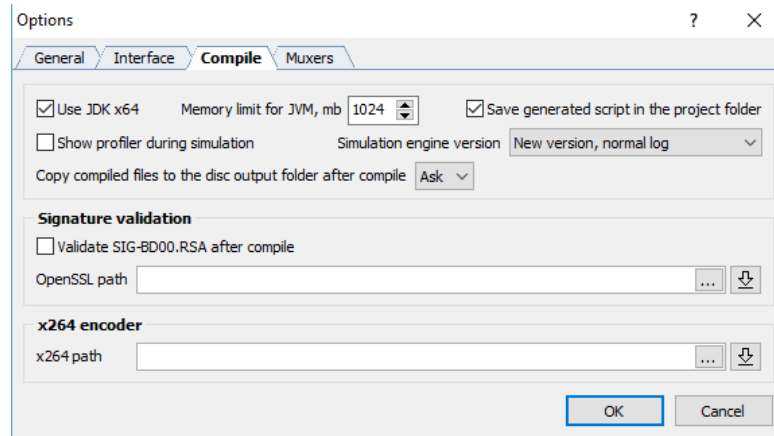
11:42:56 - Step 4/4: done
```

Signing a project

Signing is a well-known software method of creating a hash key out of the data that makes up the project. Changing a single bit in the project invalidates the signature and you know someone tampered with it.

The signing process is defined through “trusted” organisations that will vouch for your signature being generated by you and not someone else. These organisations are specified in the JAR (see previous section).

This SSL information (Secure Socket Layer) is specified in Tools> here you should specify the openssl.exe path. You can get OpenSSL via official website. We recommend to use versions from overbyte.eu (Download OpenSSL Binaries section) or bintray.com/vszakats or indy.fulgan.com.



Automatically start a movie

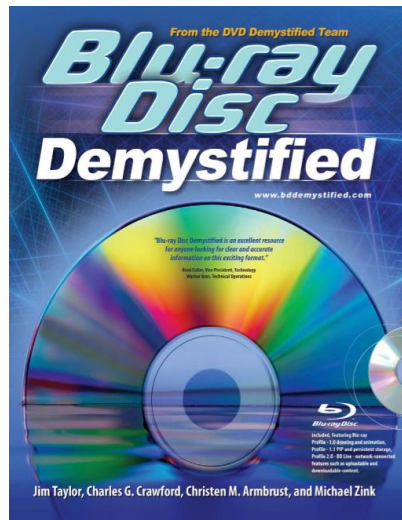
To avoid the main menu to be shown indefinitely, you can set a timing after which there is an automatic selection from this menu and that selection is executed. Usually this is the playing of the main movie.

Appendix C: Bluray specifications

The following data indicates the Bluray Disc Association (BDA) decisions on capabilities bluray discs and set top players must have. This will also reveal in how far Blu Disc Studio adheres to these standards. A Wikipedia article describes the structure of a bluray disc (what goes in what folder): see

<https://en.wikipedia.org/wiki/Bluray#Video> .

An interesting description of all these details is found in a book by Jim Taylor a.o. “Bluray Disc Demystified” (McGraw-Hill Publishers, 2009, ISBN 978-0-07-159092-1)



Bluray regions

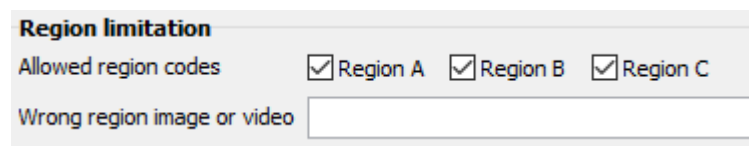
Following the easily circumvented DVD region codes 1-5 for parts of the world, the BDA still saw it fit to come up with a different division of the world in bluray regions A-C, mostly to protect content providers from discs being bought in region A at some lower price to play in region B where the disc was either unavailable or at a different price and through a different vendor. Regions are profit-inspired, not user friendly. There are tampered-with “region free” set up players but less than region free DVD players. It is estimated 70% of the commercial bluray discs have no regional restriction.



- Region A:

- North America
- Central America
- South America
- Korea
- Japan
- South East Asia
- Region B:
 - Europe
 - Middle East
 - Africa
 - Australia
 - New Zealand
- Region C:
 - Russia
 - India
 - China
- Rest of World

In BDS you can specify regions in Project > Project Setting > General tab.

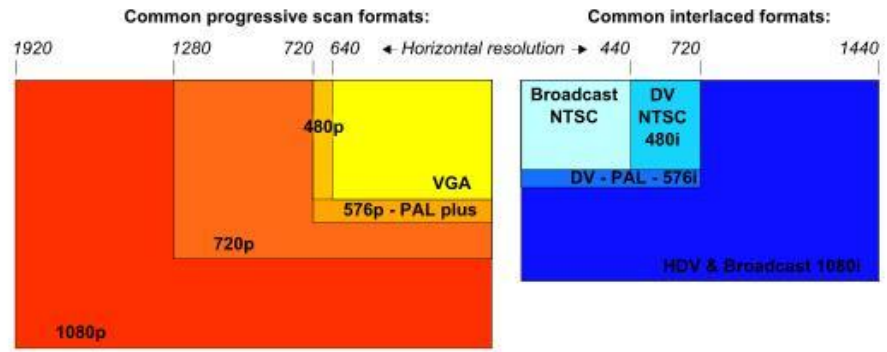


Aspect ratios and resolution

There is a subtle difference between aspect ratio and resolution.

- The aspect ratio defines the relation between vertical and horizontal size and as such can be anything in terms of resolution. A 4:3 ratio is satisfied by a 100:75 pixel resolution but also by 720:540. CinemaScope, TechniScope and other long image formats may not entirely fill the screen resolution.
- Resolution presents the real physical number of pixels in horizontal and vertical direction. The more pixels, the more detailed the picture. Hence a “widescreen” image at DVD’s 720 pixels (SD) standard resolution is much less detailed than the same “widescreen” image at bluray 1920 pixels full-HD resolution.

Bluray discs were defined for the widescreen world, showing images in 16:9 aspect ratio format. This is done by showing pixels that fill this ratio. Pixels may be shown (by television or video projector) as square or rectangular (anamorphic). Therefore, a number of resolutions are acceptable, all showing a 16:9 aspect ratio image. The image can be shown complete (progressive – p) or like the “old style” television showing even and uneven lines in quick succession (interlaced – i).



The figure above shows the various common formats in both aspect ratio and actual pixels (horizontal: shown on top, vertical: mentioned next to or at the bottom of the screen format).

Interlaced (television and DVD oriented)

- 480i = 720 x 480 – DVD NTSC interlaced (4:3 format or anamorphic 16:9)
- 576i = 720 x 576 – DVD PAL interlaced (4:3 format or anamorphic 16:9)

Progressive (full images, bluray specific, cinema digital projection)

- 480p = 720 x 480 (NTSC “full screen”)
- 576p = 720 x 576 (PAL-plus 4:3 format or anamorphic 16:9)
- 720p = 1280 x 720 – HD or HD Ready, always 16:9 aspect ratio with square pixels
- 1080p = 1920 x 1080 – Full HD, always 16:9 aspect ratio with square pixels

Movies that themselves have different aspect ratios are shown with black bars on top and bottom (extremes wider than 16:9) to make them fit the 16:9 aspect ratio. It is called “letterbox” for obvious reasons. These black bars are added by the player software so you need not transcode a CinemaScope movie with an editor that adds the letterbox bars. It suffices the movie has a 1920x1080 pixel resolution. Any aspect ratio is made to fit by adding black bars to avoid the image becoming distorted.

The same is done when left and right bars are added (less widescreen, like the old Academy standard 4:3 of older tv sets and movies). This is called “pillar box”.

For movies ripped from DVD the standard definition needs to be set to 720x480. The aspect ratio can be anything and the bluray player will give the proper ratio on screen by adding black bars where needed. However, standard definition resolutions can be Academy format (4:3) or widescreen (16:9). In that case ensure (e.g. by using the MediaInfo tool) that widescreen is indicated if the image is widescreen. Any other aspect ratio of movies ripped from internet (that may be 3:2 or 15:9) must transcode to the proper resolution, but their aspect ratio can remain untouched.

The newer game console and 4K and 8K standard are meant for extremer widescreen: 21:9 that may be a future standard for the UHD (Ultra-HD) successor of bluray discs:

- 1440p = 2560 x 1440 – Quad HD (2x2=4 x more pixels than 720p)
- 2160p = 3840 x 2160 – 4K or Ultra HD (2x2 = 4 x more pixels than 1080p)
- 4320p = 7680 x 4320 – 8K (4x4 = 16 x more pixels than 1080p)

It appears that tsMuxer, some software players like PowerDVD and set top bluray players are accomodating to various SD-like resolutions like 900x640 often found as .mp4 files on internet. Resolution should be 720x480 or 720x576 pixels (SD resolution) and aspect ratio is either 4:3 (Academy format with "square" pixels) or widescreen 16:9 (same number of pixels but stretched to make a 16:9 aspect ratio).

Supported frame rates

All bluray discs (and therefore the players) must support the following frame rates for primary video (i= interlaced, p=progressive):

Resolution	Frame rate
1920x1080	23.97p, 24p, 25i, 29.97i
1280x720	(this format only has progressive frames) 23.97p, 24p, 50p, 59.94p
720x576	25i (PAL DVD)
720x480	29.97i (NTSC DVD)

Note that some software players will playback 50i and 59.94i but set top bluray players do not have to support this. The trend for home entertainment is to use 24p (24 full frames per second – the speed of cinema movies).

Also note that 30 fps is not supported (though a favourite ratio for various internet movies from e.g. YouTube).

It seems various software players and set top players are accomodating to this frame rate however.

Element	Maximum
Angles	9
secondary video streams	32
audio streams	32
secondary audio streams	32
text subtitle streams	255
pop-up menus	32
chapters (playlist marks)	999
playlists (file names 00000.mpls to 01999.mpls)	2000
play items in playlist	999

JAR files	limited by disc capacity
movie objects (.bdjo objects)	4000
fonts	255
menu sounds (in sound.bmdv)	255

Television sets must be of LCD, plasma or LED variety to be able to show progressive frames. Old fashioned tv screens for PAL and NTSC only handle interlaced video at standard resolution. So, for those into High Definition and blurays these old tv sets are useless anyway.

Number of BDAV/BDMV elements

There are maximum numbers of allowed audio streams and subtitles: 32 video and audio streams each, 255 subtitle streams.

Number of angles	9
Secondary video streams	32
Audio streams	32
Secondary audio streams	32
Text subtitle stream	255
Popup menus	32
Chapters (playlist marks)	255
Playlists	999
Play items in a playlist	255
JAR files	Limited by disc space
Fonts	255
Menu sounds	128

Supported primary and secondary video streams

A number of video streams are supported for the primary video (“main movies”). When picture-in-picture (PIP) is used as secondary video, some restrictions apply.

The following two tables (from “Blu-disc Demystified”) provide the possible combinations.

Allowed video codecs

Primary video	Secondary video		
	MPEG-2	MPEG-4 AVC	SMPTE VC-1
MPEG-2	√	√	√
MPEG-4 AVC		√	
SMPTE VC-1			√

Note that this list does not include MPEG-TS as produced by PowerDirector (even if specified H264/AVC m2ts output). The tsMuxer will not complain and the resulting disc will play in many if not all standalone players. But this type of output ought to be transcoded to BDA MPEG4-AVC to be officially supported. (Some transcoders seem not to transcode to proper AVC – the BDS MX muxer might fail on them. Even MediaInfo may wrongly show a movie to be properly

AVC formatted. When the movie is muxed properly by the internal muxer you may be assured it meets BDA standards).

Allowed combinations of primary and secondary video formats

Primary video		Secondary video	
Resolution	Frame rate	Resolution	Frame rate
1920x1080	29.97i	1440x1080	29.97i
1440x1080		720x480	
	25i	1440x1080	29.97i
		720x480	
	23.976p	1440x1080	23.976p
		720x480	
	24p	1440x1080	24p
		720x480	
1280x720	59.95p	1280x720	59.94p
		720x480	29.97p
	50p	1280x720	50p
		720x480	25p
	23.976p	1280x720	23.976p
		720x480	
	24p	1280x720	24p
		720x480	
720x480	29.97i	720x480	29.97i
720x576	25i	720x576	25i

H264 video codec compatibility is found by using:

- x264 (https://www.blu-disc.net/download/x264_cmd.zip)
- Magix Vegas Pro (<https://www.vegascreativesoftware.com/>)
- Adobe Premiere – (<https://www.adobe.com/products/premiere.html>) only for progressive streams (for interlaced versions you need older versions from 2017 or lower)
- Grass Valley EDIUS (<https://www.ediusworld.com/>) – for progressive streams only

Do not use the freeware DaVinci Resolve – it doesn't produce BD specification compliant streams. They need to be transcoded to become compliant. (This is true for many other video editors too).

Supported primary and secondary audio streams

A number of audio streams are supported for primary video (not the "picture-in-picture" movies)

Audio	Loss less	Man-datory	Sampling frequency	Bits per sample	Maximum data rate (Mbps)	Maximum channels
LPCM	Yes	Yes	48/96/192 kHz	16/20/24 bits	27.6	8
Dolby Digital	No	Yes	48 kHz	compressed	0.640	5.1
Dolby Digital Plus	No	No	48 kHz	Compressed	1.7	7.1
Dolby True HD	Yes	Yes	48/96/192 kHz	Compressed	18.64	8
DTS	No	Yes	48 kHz	Compressed	1.509	5.1
DTS-HD	No	Yes	48/96 kHz	Compressed	6	7.1
DTS-HD MA	Yes	No	48/96/192 kHz	Compressed	24.5	7.1

For PIP movies, BDA (and therefore BDS) only specifies Dolby Digital Plus (1.0, 2.0 and 5.1 channels) and DTS-HD (2.0 and 5.1 channels).

Note that 44.1 kHz is the standard sampling frequency for CD music. This frequency is not acceptable. In BDS it causes an odd error message during muxing ("BDMV\PLAYLIST\00001.mpls not found. Restarting...")... And if it let things pass through, some players may produce sound, others remain silent (as 44.1 kHz is not a supported standard) – this applies equally to Dolby Digital .ac3 as to LPCM .wav files.

Mandatory playback for players

The BDA standards defined the minimum set of video and audio playback capabilities for bluray disc set top players. These are:

- Video – at frame rates 23.976, 24, 25i, 29.97 and 59.94
 - MPEG-2
 - MPEG-4 AVC/H.264
 - SMPTE VC-1
- Audio
 - Dolby Digital with at least 2 channels (.ac3) at 48 kHz sampling rate
 - DTS Digital surround (at least 2 channels) at 48 kHz sampling rate
 - Linear PCM (LPCM) at 48 and 96 kHz, at least 2 channels. 44.1 kHz is not supported (typical CD grab) ..

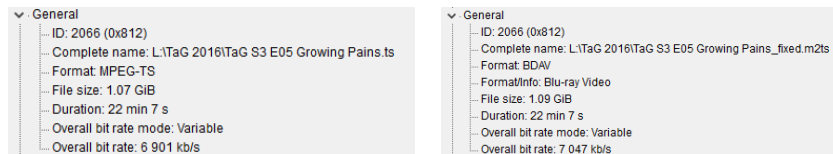
Note: although all players ought to adhere, some do not. From personal experience, the Medion players seem not to accept interlaced video (they should) and play only audio, Samsung does but not always plays interlaced video without stutter. Samsung does not accept 44.1 kHz audio, Sony does as well as the software player PowerDVD..

Best check your players first. But you can always blame the player if you stuck to the BDA requirements. A safe bet seems to be to use progressive video streams only. So far that is accepted by all players I've used.

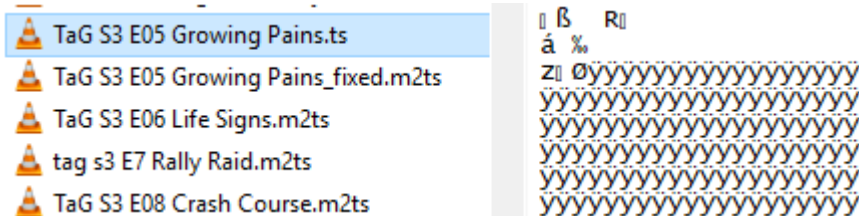
Optionally they can support

- Video
 - Frame rates 25p, 50p (usually provided for the European market players)
- Audio
 - Dolby Digital Plus (7.1)
 - Dolby TrueHD (7.1)
 - DTS-HD Audio (7.1)

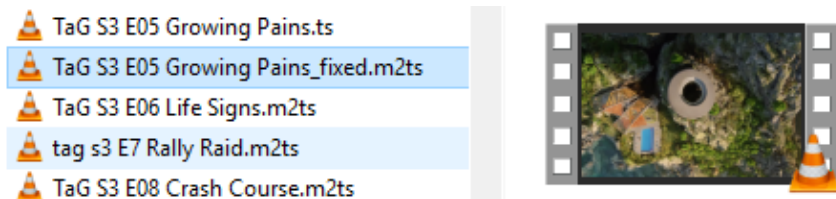
If you use the MediaInfo tool to check on your video file properties, ensure the video format is BDAV (and not MPEG2-TS as many .ts files are)). Compare both figures below(left: .ts file which is not BDA compliant and at right: .m2ts transcoded file that is BDA compliant).



This can also be seen if a demuxer is used to split video and audio. The “preview” window shows the .ts version has some data in its header before the movie starts.



A .ts version of an off-air recorded television episode.



The AVC .m2ts version of the .ts converted to BDAV stream. The MPEG4-AVC is officially supported, some video tools produce MPEG-TS files that seem to work fine but are not AVC compliant and might fail to play on some standalone players.

Player Profiles

Each player conforms to a certain “profile” level that it supports.

These “profiles” determine the quality of the video based on types of frames used, chroma (color) format etc.

- BP basic profile, low-end, video conferencing, mobile apps. Uses I,P frames
- MP main profile, HD tv broadcasts, DVB standard, uses I,P,B frames
- HP high profile, HDTV DVB broadcasts, bluray storage, uses I,P,B frames

The level specifies the bitrate with which the picture is shown. Levels exist from 1 upwards but only after 3 (which is audio only) they become useful for HD recordings.

Level	Resolution @max framerate	MP Bitrate kbps	HP Bitrate kbps
4	1280x720@68.3 1920x1080@30.1	Medium@4.0 20,000	High@4.0 25,000
4.1	1280x720@68.3 1920x1080@30.1	Medium@4.1 50,000	High@4.1 62,500
5	1920x1080@72.3	Medium@5.0	High@5.0

Standalone players can handle everything up to a certain “profile level”.

- Profile 1, version 1.0, players introduced before November 1, 2007. Only profile that does not support picture-in-picture (PIP)
- Profile 1, version 1.1. players introduced after October 31, 2007
- Profile 2 (required for BD-Live and PIP⁵⁷), which has mandatory internet connection capability
- Profile 3 (audio-only player),
- Profile 4 (real time recording and editing and PIP), which has mandatory internet connection capability
- Profile 5 (3D contents, PIP)

Blu Disc Studio is ignorant on profiles – the video and audio streams must adhere to them, but all BDS does is muxing the streams together regardless of profile, as long as the muxer muxes. Bluray players can upgrade their firmware – unlike the older DVD players.

MPEG2 H.262, MPEG4 H.264/VC-1 and AVC profiles

MPEG-2 is the standard introduced for DVDs. Later, it got enhanced to also embrace HD video MPEG2/H.262. It was used in the now defunct HD DVD commercial releases in brown keep cases that were in competition with bluray before bluray won the battle. The MPEG-2 encoding was used by many studios for the first commercial blurays. When demuxing a H.262 movie, it results in .m2v (or .mpv) and .ac3 video and audio files.

MPEG-2 for commercial discs got quickly replaced by the SMPTE (Society of Motion Picture Television Engineers) defined video codec VC-1 or by MPEG-4 (advanced video codec) AVC which compresses better and therefore takes less space on disc.

MPEG-4 is a standard compression method for video and by itself independent of bluray discs. One if the compression modes is H.264 and noted as MPEG4/H.264. It is also the broadcasting standard for most television stations. Due to the limited broadcasting bandwidth of 19 Mbps these stations can only broadcast up to 720p or 1080i. When demuxed it results in .264 and .ac3 video and audio files.

Home cameras and commercial bluray discs use it – even use up to 40 Mbps allowing for superior image quality compared to television broadcasts limited to 19 Mbps.

HDMV versus BD-J playback

Blu-Ray disc knows two author softwares: Blu-Ray Disc Java(BD-J) and High Definition Movie (HDMV). Both authoring systems use

⁵⁷ PIP = picture in picture

the container structure of MPEG-2 into which they insert a Packetized Elementary Stream(PES). HDMV is compatible with Transport Stream(TS). Blu Disc Studio uses BD-J.

The HDMV movie technique is used for creating interactive features. It defines the audiovisual and video capabilities of Ultra HD Blu-Ray. In this context, it supports simple graphical representations such as picture-in-picture(PiP) and graphical effects involving position, color, or framing. There are also some audio features such as multi-language voice recording. More advanced graphical requirements are realized with Blu-Ray Disc Java.

Generally, there tend to be fewer incompatibility issues with HDMV playback than with BD-J, mainly because the BD-J environment has many more options and is based on a wide array of specifications. Within the BD-J specification, there are several libraries or packages that do virtually the same function. Some of the early players did not implement all libraries properly, but the manufacturers are working diligently with the BDA to improve testing routines to remedy these situations.

With BD-J, the chance for misinterpretation and differences in interpretation, are great. At the core of BD-J is the Java Virtual Machine (JVM). Some player manufacturers have licensed the JVM code from Sun Microsystems. Others have decided to implement their own version of JVM.

Java is used to implement interactive menus on bluray discs made by BDS. Java creator James Gosling suggested that the inclusion of a Java virtual machine, as well as network connectivity in some BD devices, will allow updates to bluray Discs via the Internet, adding content such as additional subtitle languages and promotional features not included on the disc at pressing time. This Java Version is called BD-J and is built on a profile of the Globally Executable MHP (GEM) standard; GEM is the worldwide version of the Multimedia Home Platform standard.

In the BDS Project> Project Properties you can specify the BD-J profile level (default set to 2) and the disc profile version. Keep this on “force version 1”. The other option, “same as BD-J profile” results in discs that are not universally accepted by set top players (although the disc may play in some and on software players).

One limitation within the BDS products that use BD-J for navigation between menus is the maximum limit for total number of pixels in all menus together: these must be less than 7 900 000. However, as of BDS V4.3 you can use multiple JAR files to circumvent this problem (See Using multiple JAR titles on page 210).

Disc capacity

Bluray discs come as single and dual layered discs and their capacity is expressed in gigabytes or GB. This is where confusion may start.

There is a difference in expressing numbers as multiples of ten (the way we count in daily life) and in multiples of 2 (ordinary in digital life).

The word “kilo” is used for $1000 = 10^3$. But in the binary world, the nearest 2-fold of 1000 is 1024 ($=2^{10}$). It has become customary to express digital storage in multiples of 1024 and call that also a “kilo”.

Hence we get:

Unit	Real (scientific) value	Digital value
1 k	$1000 = 10^3$	$1024 = 2^{10}$
1 M	$1000^2 = 1000 \text{ k} = 10^6$	$1024^2 = 1,048,576 = 2^{20}$
1 G	$1000^3 = 1000 \text{ M} = 10^9$	$1024^3 = 1,073,741,824 = 2^{30}$

Because the digital values for kilo, mega and giga are larger than the scientific values of the same, it was decided to indicate 1000 bytes as 1 kb and 1024 bytes as 1 kB (capital “B”) where 1 B = 1.024 b(yte). To confuse the world, an IEC organisation introduced a standard using “B” for bytes always but replacing the digital value indication by “iB” (bibyte – from binary byte): 1 iB = 1.024 B (hence 1000 iB = 1024 B).

This different interpretation seems mostly ignored and in many publications it becomes unclear whether 1 MB means 1,000,000 bytes or 1,024,000 bytes.

The Windows operating system shows sizes in 1024-based units. A disc manufacturer may sell a 500 GB disc (using real value: $500 \cdot 10^9$ bytes), Windows will display its size expressed in units of 2^{10} as a 465 GB disc ($465 \times 1024^3 = 499 \text{ Gb}$). And so a 25 GB bluray disc becomes $25/(1.024)^3 = 23.3 \text{ GiB}$

disc	actual siz in GB	Windows shown iB units (GiB given as GB)	capacity in number of DVDs	Video typical play time	Video min- max play time
BD-25	25.025	23.306	5.3	2.3 h	1.2h-37h
BD-50	50.050	46.613	10.6	4.6 h	2.3h-74.1h

The size of the discs is 120 mm with inner spindle hole 15 mm and reflectivity 35%-70%. Data is read with 4.9 m/s on playback.

When you collect your video files in a Windows folder, make sure you do not exceed 22.5 GB in Windows file size as that is about the maximum of a bluray disc.

Stream data (bit rates)

To show pictures, the video is encoded in bit stream data rates. The higher the stream rate, the more bits are available each second to define the image: a better resolution is possible. The bluray standard defines no specific stream data rates.

Television stations usually are limited in streaming to 19 Mbps. For this reason stations broadcast in 1080i or 720p ($= (720 \times 1280) \times 25 = 19.4 \text{ Mbps}$).

The progressive 1080p ($= 1080 \times 1920 \times 25 = 51.8 \text{ Mbps}$) is way beyond the allowed 19 Mbps. They usually broadcast MPEG-4 streams (to which

the high definition format H.264 belongs). Of the available 19 Mbps bandwidth, often additional channels are transmitted, limiting the main picture to 12 or 14 Mbps maximum.

Alternative, more advanced encoding standards VC-1 and AVC can do a passable 1080p within the 19 Mbps and better at higher streaming rates. Home and professional cameras can do this. A commercial bluray disc movie uses up to 40 Mbps for a far superior picture.

The table below indicates typical streaming rates in Mbps.

Video	Minimum (Mbps) (poor)	Typical (Mbps)	Maximum (Mbps) (best)
MPEG-4 AVC	4	16	40
SMPTE VC-1	4	18	40
MPEG-4 H.264	4	24	40
audio			
Dolby Digital 2.0	0.064	0.192	0.640
Dolby Digital 5.1	0.384	0.448	0.640
Dolby Digital 7.1	0.640	1.024	4.736
DTS 5.1	0.192	1.509	1.509
DTS-HD 7.1	0.768	1.509	3.000

Data streams from bluray disc to player are also subjected to maximum limits:

ts transport stream	48 Mbps
video stream 1080x1920	40 Mbps
text subtitle stream	48 Mbps
DTS-HD	24.5 Mbps

Background on MPEG/H.264 compression

Consider a single high definition frame. It has resolution 1920x1080 pixels. Each pixel has a colour made from red, green and blue (RGB). Each colour can have 256 shades – an 8 bits or 1 byte number between 0 and $2^8-1 = 255$. That means a single frame requires $1920 \times 1080 \times 3 \times 8 = 49,766,400$ bits. And there are 24 frames per second in movie, totalling to 1,194,393,600 bits/second, 1,2 Gbps or 150 MBps (b for bit, B for byte=8 bits). There are few if any households that can stream movies over internet at that rate. A current bluray disc of 25 GB capacity would be completely filled in $25/0,150 = 167$ seconds – less than 3 minutes.

It took some time, but one started to compress images. From “complete” lossless images (like BMP and TIF) they invented lossy formats like JPEG under the assumption that irrelevant details could be left out. This reduced the image size a minimum factor of 10 but with more compression and more loss upto 20 or 30 times smaller.

Filling a bluray disc with JPG images as movie frames would allow 10 to 30 times 3 minutes. Better, but still a short running time.

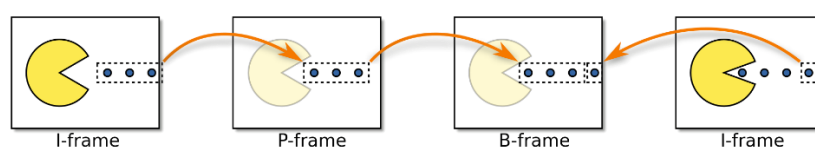
Based on the knowledge of compressing images, moving images could be compressed even further. The “complete” but lossy image would become a what is known as an I-frame. These full images would only occur every so often. In between were new frames, P- en B-frames. Both do not have a complete (lossy) image like the I-frame but only contain the changes from objects between two I-frames.

Some shop talk:

- An **I-frame** (Intra-coded picture) is a complete image, like a JPG or BMP image file.
- A **P-frame** (Predicted picture) holds only the changes in the image from the previous frame. The encoder does not need to store the unchanging background pixels in the P-frame, thus saving space. P-frames are also known as *delta-frames*.
- A **B-frame** (Bidirectional predicted picture) saves even more space by using differences between the current frame and both the preceding and following frames to specify its content.

P and B frames are also called Inter frames. The order in which the I, P and B frames are arranged is called the Group of pictures.

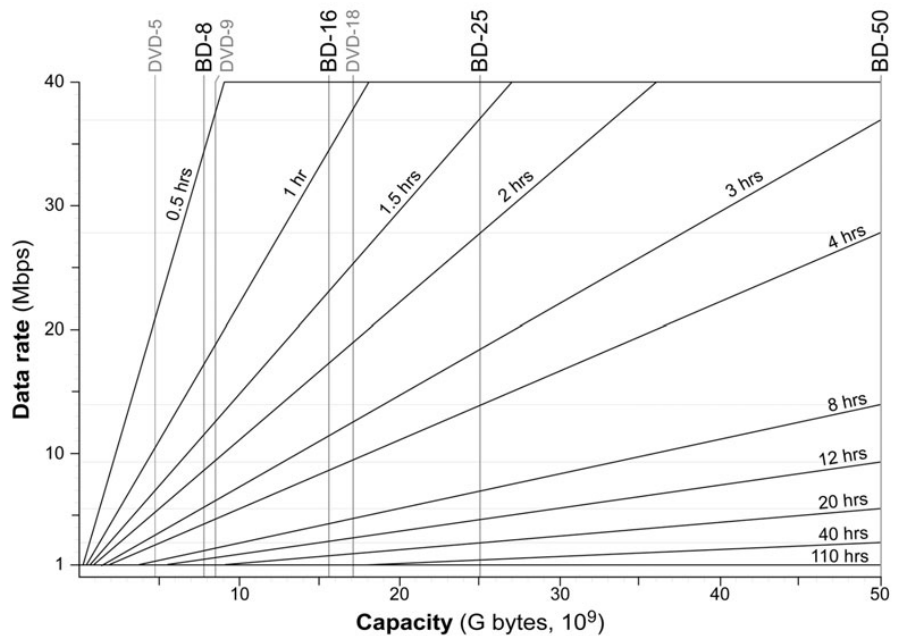
It is shown in the picture below. A cookie monster eats cookies coming in from the right.



At both ends there is a complete I-frame. The P frame shows the movement of only the cookies part of the image. The subsequent B-frame shows how a fourth cookie comes in (seen in the next frame) and the three existing cookies move further to the left.

DVDs used MPEG-2 as a standard for creating I,P and B-frames. Later developments came with MPEG-4 (also known as H.264) that deliver a similar quality at ¼ of the bitrate. Clearly this compression technique is used on blurays. The 4K discs use an even more efficient compression, H.265.

It is clear that using I,P and B-frames the amount of data needed to show one second of film decreases and thereby the bitrate. They can come down to a manageable level and increases a bluray disc capacity to contain a movie of 2 hours (high bitrate) to fill up the disc or it can contain the movie, a lot of extra's, additional audio and subtitle tracks and elaborate menus. It all fits the same 25 GB disc capacity but requires a lower bitrate. Of course the playback software on computer or as part of a standalone player, must be able to process that amount of data and recreate full frames for display.



(from: Blu-ray Disc Demystified)

Number of BDMV elements

A bluray disc consists of a number of “BDMV (=Blu-ray Disc Movie) elements” that can be selected using the remote-control. This includes number of angles used for the same scene, number of subtitles, chapters etc. The table below indicates the maximum defined numbers of each element. This also explains some of the limits set by BDS.

Angles	9
audio streams	32
text subtitle stream	255
pop-up menus	32
chapters (play mark lists)	999
playlists	2000
play items per playlist	999
movie objects (.bdjo files)	1001
JAR files	to disc capacity
fonts	255
menu sounds (in sound.bdmv)	128

Appendix D: Errors occurred: possible causes

BDS assumes certain things that may cause you to raise your eyebrows. A good place to start is reading the “How to avoid problems” section in the online help. Most of these errors I got at some point during authoring discs. Sometimes with some logical thinking the cause can quickly be revealed. Sometimes it took a lot more trial and error when the error message was less than helpful.

Here is a list of things you may encounter and a possible solution to it. See also the section “Possible problems” on page 69.

We separated the errors to the activities where they are most likely to first occur. That means that if you don't correct them during authoring, they are likely to re-occur during simulation or muxing – further down the processing line.

Errors during authoring

- **"Impossible to load frame-index file"**
This error occurs when you try to build an index file for a movie. Carefully look what file BDS tries to open. It may not exist because the original file name on which the play mark index file is based has an apostrophe or other punctuation mark in it. BDS removes these from the name and searches for a filename without those punctuation marks and of course will not find it. Remove these punctuation marks from the physical movie files, generate the index file and proceed.
- **Auto-align button navigation only works for some or none.**
This may have to do with the fact that images for one or more states of a buttons overlap with other buttons. Each button must be completely separate from the others. Another reason can be if the button images are scaled down versions of image files. Make sure you create a smaller button image that needs no scaling.

Errors during simulation

- **Java IO exception in step 1 / 4 "unable to arrange images in mosaic"**
The disc simulation or building fails during Step 1 / 4. This usually means the the total size of all images used in the menus exceeds the maximum allowed number of 7 900 000 pixels. You need to reduce the images or resize them. Note that scaling makes them look smaller in menus, but they still occupy their full size within the project. Also, when carousel menus are generated, each carousel menu adds pictures of the non-active buttons.
If you haven't done so already, you may set the BD-J version level to 2 and the disc profile to follow the BD-J version setting in the Project Properties. This alleviates the number of pixels to a larger number.

- **image buffer overflow**
This message may occur when you try to simulate the menus. It means the total size of the images is too big: beyond 590 000 pixels if you use BD-J V1. You can increase the size by going to Project Properties > General > and set BJ-Profile to V2, disc profile version "same as BD-J profile". Optionally you can set "Combine images" to "split into separate files".
Each image is then stored as separate file in your project folder __JAVA/00000 .
- **maximum menu pixel size** is 7 900 000 pixels. Regardless the number of menus or buttons used, their total size may not exceed 7 900 000 pixels.
 - You can try to double the number of pixels by setting Project>Project Properties>General item "Combine images" set to "Split into separate files".
 - try to reuse identical backgrounds (such as a black 1920x1080 png image) for all menus instead of duplicating these backgrounds
 - try to reuse identical button states (such as checkmarks for "select" or pointers for "current") for all menus
 - limit the amount of graphics or text on a menu
 - do not let a text or graphics image extend to the full screen width but limit it to its own size
 - Two small blocks making up an L-shape take fewer pixels than one rectangular block (the transparent pixels in the L-shape are counted as pixels in a rectangular shape)
 - make a "static image" into a background movie. Menu movies do not count in the total pixel count. It may liven up the menus as well.
- **Menu buttons move a little ("cha cha cha") during change of menu.** Make sure the positioning (left,top properties) of the static images of the buttons in the cloned menu align with the normal or selected state of the button in the button-menu. If a selected button opened a cloned menu, make sure you keep the "Selected" image rather than the normal image of that button on the cloned menu.
- **Wrong menu shown.** If you create the project using multiple jars, the simulator only shows the JAR title that is specified in "On JAR Startup" as it is designed only to work within a single JAR.
- **Menus seem incomplete.** Check that the resolution of the menu movie corresponds with the resolution of the menu itself. Small resolution movie "hides" the right/bottom most parts of a larger sized menu.

Errors during muxing (disc building)

- **Access violation at start of the disc creation** – odd error message that usually means "you did not specify a menu

movie". Or if you did, the files of a movie placeholder were changed (as well as a regenerated scenes index) but you did not update the playlists in which it features.

The log file will tell you so.

In other cases where the disc doesn't start to build: if many editorial changes were made, refresh BDS by closing it (+saving project) and reopen the project. Usually it then builds without problems.

- **Lots of errors in the tsMuxer window**

The movie showing the errors is probably not of the right aspect ratio or resolution. Several video converters do not add vertical bars if the original movie (like those downloaded as DivX or MP4 from internet) if the aspect ratio is not 16:9. Re-create the .m2ts file with proper ratio and resolution.

- **Build process fails in "update playlist" state**

Very likely the file where the update stopped has nonprintable characters in its filename (like ASCII 0x00 or 0x86) that are valid to Windows and BDS but not to tsMuxer. As of BDS V4.1.0.1692 BDS traps these names and suggests a replacement after which the problem should no longer occur.

If you do not let BDS replace these characters are invalid to tsMuxer and you need to remove or replace them yourself manually.(see Step 4: Build the disc on page 64). The log file will also show that tsMuxer almost needed no time to (not) mux the offending movie. But it continues with the next movie.

Although a valid character, also refrain from using punctuation marks like apostrophes or ampersands, diacritics in filenames such as ø, ë or anything outside the regular A-Z, a-z and 0-9 character set. It is usually the American tsMuxer tool that cannot handle these but doesn't report them as a problem until it fails.

- **After Muxing starts a failure to find .mpls file**

Once muxing starts you may get a warning like:

```
12:10:14 - Muxing menu "main" (BDID: 1)
```

```
12:10:14 - waiting...
```

```
12:10:17 - [warning] G:\BDS projects\My
Movies\output\BDMV\PLAYLIST\00001.mpls not found.
Restarting...
```

```
12:10:17 - waiting...
```

```
12:10:20 - [warning] G:\BDS projects\My Movies
\output\BDMV\PLAYLIST\00001.mpls not found.
Restarting...
```

A possible reason for failing to mux is an audio file associated with menu movie or regular movie. If this is a LPCM/WAV audio file with 44100 Hz sampling frequency (CD rip) is not supported. Replace it by or transcode it to one of 48000 Hz or a different type like .ac3. The muxing process seems to continue (after as many warnings as there are movie placeholders) but in fact the result is unplayable.

Another cause may be that the movie is set for 30 or 60 fps. This is not a valid frame rate for bluray discs. Transcode to 29.97 fps or 59.94 fps and the muxing process continues without a flaw.

A third, less likely, cause is if you rebuild a disc but still have a software player open that locks some files in the output folder that BDS tries to re-create. This may result in not being able to replace a file and read its .mpls file. Close the player application and remux.

- **End Action not specified**

Any individual movie must have an End Action specified – to execute when the movie ends. You do not specify this for an intro movie (used by a menu).

- **“Updating movie playlist”** reports error. Check the time it took to create that movie in the output BDMS\Stream folder. If it is much shorter than expected, the tsMuxer may have not completed its task correctly. Ditto if the file length of the movie is much shorter than expected. BDS only finds out during the playlist phase. You may need to re-create and demux those movie tracks, check if they are complete, and remux again.

- **Meta Data generation failed** This error is given before a movie with subtitles is muxed. Reason may be that some of the subtitles contain invalid characters. These must be removed. Often opening the .srt file in Notepad and simply saving it under a different name may rectify the problem although the incorrect characters remain. As an example:

```
168
00:16:07,657 --> 00:16:10,148
<i>¡Atención.¡</i>
```

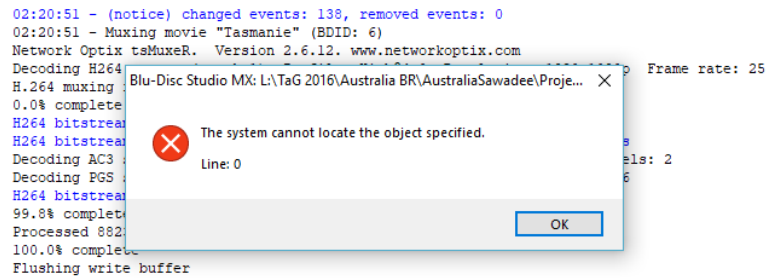
Here the original subtitle should read “¡Atención!” but was ripped incorrectly. Ensure the .srt file has attributes UTF-8 and BOM set. (Tools like Subtitle Edit will show this).

- **The log window has disappeared** (or any specific BDS interface window) and cannot be shown anymore. In that case:
 - close the BDS application
 - Delete registry key
HKEY_CURRENT_USER\Software\Disc-Art\Blu-Disc Studio\Windows (and all its subkeys)
 - Delete Blu-Disc Studio MX\layout.xml file in your Documents folder (if it exists)

Then restart the BDS application. The interface should look as “factory installed” and you may need to modify it back to what you find most pleasant. The log window should be visible as a docked window.

- **ES file older than original file** – can happen when you use the internal MX muxer. Muxing movies requires using the generated index files (.mes and .mom for internal muxer, .idx and .inf for tsMuxer). Some of these may have become corrupt. Delete and recreate the files (Project > Project Properties > General tab, click “Delete all index files” and then “Generate all index files”). It does mean you need to re-create all chapter settings too. Alternatively, you may look for the offending .es file and delete it (in project folder as well as in %appdata%\dvdlogic\muxer) and retry muxing the project.
- **“current” state not found**
 - a movie menu specifies an action for a button, but the action (set subtitle / audio) is specified for another movie (especially if movie is moved to another group)
 - in the properties window the state is associated with a file that no longer exists or was moved or renamed
- **cannot copy 0000.jar file**
 This error may be caused by filenames in the project that cannot be handled properly, even if acceptable by Windows itself.
 Especially punctuation marks like apostrophes, ampersands, brackets “(“ and “)” as well as non-printable characters (such as ASCII null, linefeed, etc) as part of the name seem notorious in being unacceptable. Use Latin characters, numbers and spaces only to avoid these problems.
- **“The system cannot locate the object specified. Line: 0”** – audio problem
 This error may occur when a movie has the LPCM audio (uncompressed) mode with sampling frequency of 44100 Hz rather 48000 Hz or a Dolby .ac3 or DTS. Replace the 44100 Hz WAV file by something else.
- **“The system cannot locate the object specified. Line: 0”** the menu building succeeds (JAR file created), but the muxing of the project files fails. This usually means BDS cannot find the file. You may think it is there, but if it contains non-printable characters they may not show up in the Windows Explorer. You need to remove those characters from the filenames or provide a completely new one.
- **“The system cannot locate the object specified. Line: 0”**
 The muxing of the movies will succeed (and the \output\BDMV\STREAM folder will contain the .m2ts files with all audio and subtitles muxed), but the final step of the building will fail. This is when the muxing progress window starts its “Updating....” Phase. The movie mentioned last before the process is stopped and the error message is shown, is the one where BDS found a problem.
 BDS tries to compose its menus and provide links to movies or their audio and subtitle tracks. For some reason or other it

cannot link to (one of) these tracks and then fails. Without telling you what link it cannot create.



02:22:47 - Updating movie in playlist "AustralieDVD"

This problem can only be figured out by trial-and-error and some deductive thinking. The problem lies in either:

- Let the muxer use an external window instead of reporting into the BDS log file window (Tools > Options, Muxing tab, check "Run in external console window")
- movie file is not in bluray specification resolution
- audio file(s) are corrupt
- subtitle file(s) are corrupt
- one or more files listed in the properties of the movie (video, audio or subtitle) do not exist (were renamed, deleted or made inaccessible)

To figure out the problem:

- save the current project under a different name
- start removing all movie files but the offending one
- remove the item you suspect (audio, subtitle) and remux
- if the error still occurs, remove another item
- if success, the removed item must be re-made: sometimes a simple opening and explicit closing (save-as instead of "save") does the trick. For subtitles you may wish to save in Windows 1252 format instead of UTF-8 or something else currently used Go back to the original project and remux. If the offending file is corrected, it should now complete successfully unless further down the line another movie has offending parts
- **UMaskTable not found** – this error may be seen in the log file during production of a disc. If you look at earlier warnings it may show that some movies (especially play lists) do not have an End Action or Popup Menu action specified. Once you run such a movie, there is no way to get out of it or the disc stops at the end of the movie. Specify the actions, recreate the JAR file and replace it in the muxed output. This gives a corrected version of the disc.
- **A message line is highlighted in red during muxing** Usually this means there is something wrong with the file mentioned, but BDS also highlights messages that contain certain words like "error" or "unexpected". Especially if these words are part of the movie title, they will be

highlighted unnecessarily. So a movie "unexpected Christmas" might mux perfectly fine but is still highlighted in red.

Errors during burning of a disc

- **size exceeds the BR-25 25 GB limit**
 - use a BR-50 disc to burn on
 - re-encode some of the movie items to a lower bitrate, creating smaller files (but with less picture detail)
 - reduce the size to BR-25 size using a compression tool such as BD Rebuilder (<http://www.videohelp.com/software/BD-Rebuilder>) or DVDFab.

Errors during playback of disc

- **Disc does not play, hangs, has subtitles in odd positions and other unexpected behaviour** This may have to do by having images or image buttons that have been rescaled down significantly (50% or less). Although within limits of a BDA specification, certain players start functioning erratically. Try to use as many images at their 100% scale (resize them in an image editor) and remux again.
- **Opening menu (or some other menus) remains black.** The reason may be that the menu movie associated with the menu is corrupt (if it cannot play BDS will show nothing, i.e. "black screen".) This may go unnoticed by the tsMuxer muxer.
- **Build completed with 0 errors, yet disc does not play** This may be due to incorrect muxing, especially when the external tsMuxer is used. This may have indicated several muxing errors, yet these are not transferred into the error count of BDS or its log file. Any completed tsMuxer process is considered successful. So rebuild the disc and keep an eye on the tsMuxer warnings and error messages. If all mux well, there is another reason why the disc does not play.
- **Menu audio cut short** The audio is trimmed to fit the length of the menu movie. If this movie is hidden behind a full screen menu and possibly "black" and runs for only 5 seconds, then the audio track is also cut to 5 seconds. Adjust the video length to fit the audio length.
- **Error on subtitles** (Distance between subtitles is incorrect). This may be a valid error: then you need to rectify. You may choose to ignore this error by setting Project>Project Properties>Streams block "Subtitles" SRT verification to "Autocorrect, if possible" and check "Remove subtitles with zero length".
- **Some graphical symbols (musical notes) and diacritical characters like ø or ä do not show correctly on screen.** This may have to do with improper processing of the subtitle

.srt file by the muxer. But also ensure the .srt file has the correct file attributes: UTF-8 and BOM encoding (=way the bits are ordered, big-endian or small-endian). (Tools like Subtitle Edit will show this)

I have seen a case where display of symbols differ: incorrect on a s/a player but then properly displayed after a new power-on of that device. Hardware/software error of the device?

- **Subtitles don't show or extend beyond screen edges** – if this happens to movies in SD resolution, subtitles require a reduced font size of 25 points or less to be in proportion to the image height. For suitable size of the subtitle depending on the resolution, see section Blu-ray disc set top player specification on page 20.
- **Movie remains black** – sound audible, no subtitles seen
This may be a problem with the software Blu-ray player you use. Try opening some of the .m2ts files in the output folder BDMV\STREAM\ and play it in a video player like VLC. If everything works there, it is a problem with the software player, not the disc image. However, check if the shown movie has aspect ratio 16:9. If not, set top players may stretch it to become 16:9, distorting the picture. In that case you need to transcode the movie to aspect ratio 16:9 (including black bars where needed) using one of the BDA approved pixel count resolutions and rebuild the disc again
- **Some (popup)menus don't show** – this is usually the case if the resolution of the movie doesn't match the resolution of the menu. Particularly if an SD movie (720x576) is used and has an associated popup menu in HD (1920x1080) only the first 720x576 pixels of the menu are shown. Remember that the resolution of the playing movie (real movie or menu movie) is leading in the decision of what is shown on screen.
- **Disc does not play in set top player.** Has the project property "disc profile version" been set to "same as BD-J"? Retry using "force version 1". Some BR players won't accept versions other than 1. Setting version 1 may cause other problems if the total number of pixels for images in menus exceeds the maximum for BD-J V1.
- **menu in final project doesn't show correctly** (only a part is seen)
The resolution of the menu background movie is leading. If it is set lower than the static menu text, only part of it is shown (its top left most part) – equal to the resolution of the movie.
- **Popup menu doesn't seem to show**
Are you sure the menu is not off screen? This happens if the popup is designed under the 720 pixels which is the bottom of half-HD (or 480 for SD). Simulation will show the popup menu as it always runs in full HD. If the movie of the popup is half-HD, the popup cannot show: the screen is limited in resolution to the movie playing. Adjust the popup menu elements by moving them into the visible area.

- **Popup chapter menu misses navigation**

This may happen if you generated the chapter menus. These end up in the “Menus” branch and you cloned the items into the “Popup menus” branch. Rather than use “clone”, use the “clone all between main/popup” . The latter checks navigation after the cloning. The former does it after each single clone and removes links to items not (yet) in the popup menu branch.

- **(popup) menu shows, but some objects are not set up correctly.**

If the menu is setup during opening, it has an “Enter” action specified. If it is opened from other menus or actions and in “Jump MENU” only the menu name is used, rather than its [anm/act 1] (opening with its animation and opening action) the setup actions are skipped. Change the actions that open the menu to allow [anm/act] to happen.

- **“current” settings do not show correctly on playback**

Check if the movies for which you set a current setting belong to the same movie group. All subtitle or audio settings in a group must adhere to the same order for all movies. You cannot have an English audio track as track 1 in one movie and as track 2 in another. If that needs to happen the movies cannot share the same group. Also check the position of the button state. It may be different from the other states and “way off” screen.

- **subtitle does not show**

Check if it is added to the movie properties. A .srt files is simply a text file with timings – based on whatever frame rate you used. Not showing while being sure it is there may be a deficiency of your software bluray player. Check the muxed movie file first. It is one of the files created in \OUTPUT\BDMV\STREAMS*.m2ts files. Play the right file (the BDID id during muxing tells you which sequence number it is usually the same one as its position in the movie list of the project tree view) using a movie player such as VLC. If you can enable the subtitles and they play in sync with the dialogue, nothing is wrong. Otherwise you need to figure out why those titles are not included.

- **Jittering (or stuttering) picture.** A picture smooth running in its original format turns out to jitter on bluray. Often this can already be seen in the converted output from a video editor if the frame rate had to be adjusted or interlaced picture was converted to progressive. The reason may lie in frame errors in the original that are not resolved by the time base that has become unreliable. This may also be the cause of lip-sync problems. Most problems can be resolved by correcting the time base:

- Re-encoding the source once more, using a bluray acceptable framerate as close to the original as possible

- Re-do the time code of the original. This often requires you to:
 - convert the source into a .mkv container (using MKVToolnix)
 - extract the source again from the .mkv container by muxing it to a .ts file
- or
 - transcode .m2ts into .wmv and back. Use a higher bitrate in .wmv to limit the loss in quality of the picture
- Re-do the time code of the original may also work by re-muxing the original:
 - open tsMuxer GUI and open the original source (it shows the video/audio tracks inside the file)
 - Select Output “M2TS” muxing and specify output file name

Note that this usually means subtitle streams may be lost on the output .ts file. You may want to extract them before entering the source in the .mkv container or to extract the subtitle from the .mkv file separately.

If re-doing the time code now introduces lipsync problems (see next item) it might be solved to save the audio of the stuttering video and use that audio track instead of the re-done time code video.

- **no audio on playback movie** Try to use the “previous chapter” button on the remote control to see if audio suddenly appears. If so, there may be a problem of different frame rates of both menu movie and selected movie (e.g. 23,9 fps menu, 25 fps movie). Some players have problems switching frame rates and keeping audio. Another reason can be a different type of audio between menu movie and selected movie (e.g. ac3 on menu and .dts on movie).
- **audio out of sync.** Audio that is in sync in the source (and possibly even during editing), is out of sync in the produced output of a video editor.

Before trying to remedy this situation, consider if you can output the individual video and audio streams (you need them separate for BDS anyway) and use those in BDS and see if the muxed project is in perfect sync.

If this is not the case, you may try to eliminate the error in the video editor. There may be several causes:

- A main reason may be due to the video stream if it has a variable bit rate. Try to transcode it with constant bit rate (but this too may cause sync problems). If a video converter has a “smart fit” bitrate, do not select it but specify a specific bitrate. You may also try Handbrake (Preset High Profile, codec H264, video tab select framerate, select

“constant frame rate”, quality 20) but it may be a slow process and the resulting .mp4 or .mkv needs further transcoding to BDA formats.

- If you tried to change several aspects at the same time (like frame rate, aspect ratio, inserts etc) try to do these in separate stages and re-import the output for next set of changes. Often video editors get “confused” doing several changes at the same time
- Switch off “ghost files” (or “shadow files”) or other low-res copies the video editor uses to speed up editing. Or at least wait until those ghost files have been completely generated (you may insert edits at the wrong spots if the ghost files are not yet finished)
- some sources have variable bitrates (VBR) and others constant bitrates (CBR). Most editors work best using CBR. Try to convert all to CBR.
- the source doesn’t start on an I-frame. Try to cut a small part at the beginning to start on an I-frame and output the rest again
- the source is made using a tool (like TS-Doctor) to remove commercials from television broadcasts. The starting points of each section may not be on an I-frame. Edit the front-to-back splices where the commercials are removed to a proper ending and starting of each part and output again.
- Users of TS Doctor may check the (bottom of the) log file when the source file is converted using this tool. It may indicate times when video or audio has problems. Note these times and remove those parts in your video editor before creating a bluray compliant H264 formatted file. You may miss a small part (of a second) of the final video, but lip-sync is maintained.
- Time base problems. This may happen when either or both video and audio tracks have problems that show once demuxed/remuxed. Try any of the methods suggested for jittering or stuttering pictures – especially transcoding to .wmv format and back to .m2ts
- Errors in frames. Especially satellite broadcasts may have a an occassional frame error with blocky images. This will cause havoc on the time base. Try to remove this frame error by either:
 - Cut the error frame bit from the movie between two I-frames
 - Splice the movie directly after the frame error and insert a one frame black image. The editor must then create an I frame there and for the rest of the movie one frame

later. This probably alleviates the sync problem.

- The (part of the) source video speed is modified (e.g. to compensate parts running at different speeds like 23.9 and 25 fps) – the output codecs improperly handle the speed up/down audio and/or video. Try find a source with no different video speed. Or try different outputs (mp4, mkv) – that may solve the problem. But this problem is rather persistent.
- software bluray player shows no picture (or sound or seems otherwise not to work)
Check the output folder for the BDMV\STREAMS folder and open some of the .m2ts files using some video player like VLC. If the movie plays well there, the problem seems more with the software bluray player. Try creating the bluray by burning it on a rewritable bluray disc and see if a standalone player works well with it. If so, burn it on a write-once bluray disc.
- **audio out of sync** – movies are in sync (even after editing) but get out of sync during muxing.
 - The culprit can be a different frame rate between menu movie and actual movie. Use the same frame rate.
 - Next check if the produced .m2ts movie files in the \BDMV\STREAM folder are in sync. If so, there must be something odd with the disc player as that file is the one played by the set top player. Try another disc player to make sure.
 - Inspect with Windows Explorer or Mediainfo the length of the muxed (original) .m2ts movie. The demuxed video and audio streams must have the same length. Use the Mediainfo tool to check on the audio lengths. They ought to be the same duration. If not, the demuxer goofed. Try another demuxer or switch between tsMuxer and the MX internal muxer to see if that rectifies the situation.
Mediainfo doesn't tell you the length of the video stream – just the audio ones. Video players like VLC or PowerDVD give the wrong duration indications.
- **chapter jumping doesn't work using tsMuxer** - this problem occurred from version V4.6 subrelease 2074 onwards. The switches to activate tsMuxer with were modified to remove some problems for some users but created new ones for those using PowerDirector to create their .m2ts files. This tsMuxer switch change may result in chapters that, although set, always let the movie restart from chapter 1.

One way to solve this problem is to convert the .m2ts file into another .m2ts file using a format converter such as HD Video Converter Factory.

The other way is to re-insert the missing switches permanently. For this, execute the following steps:

1. Open BDS without opening a project
2. Goto Project > Project Setting > Streams tab
3. Add "insertSEI, contSPS" (without the quotes) in the tsMuxer box in the "video" labeled text box
4. Click on the "Save as Default" button
5. Close BDS
6. If BDS wants to save the modified project, click "No"

If you want to modify the tsMuxer behaviour only for the current project session, perform only steps 2 and 3 on your current project. The setting is not kept between sessions of the project.

Appendix E: External Tools

BDS basically authors your bluray discs and is mostly active in defining menus, their links with movies and the movie chapters.

It does a basic check on the video and audio streams, but relies on them being conformant to BDA standards. It is left to the muxer in use to trap any errors in those streams.

Some of the following external tools may be found useful to convert any movie, subtitle or menu image to meet these BDA requirements. Use a search engine on the internet to find out where they can be downloaded and for more specific information or help about them. They can be freeware, shareware, trial-ware and fully commercial.

For most freeware tools: be careful on what links you click as most of their webpages are loaded with irrelevant adverts that try to redirect you to other commercial tools. The installation files may also lure you to “also install our recommendation” and before you know it a lot of bloatware gets installed also.

Bluray and DVD ripping

- MakeMKV (<http://www.makemkv.com/>) tool to convert blurays into .mkv containers, but it also allows to make a “backup” of the entire disc on a hard disc drive folder, removing bluray encryption. All movie files are stored in the BDMV/streams folder. The backup facility is free to use for 30 days. After that you may look at forum page <http://www.makemkv.com/forum2/viewtopic.php?f=5&t=1053&p=3548&hilit=key+code#p3548> for a renewal temporary key that lasts for 2 months.
- AnyDVD HD (<https://www.redfox.bz/en/anydvdhd.html>) continuation of the Slysoft AnyDVD software that removes all encryption and allows a disc (DVD or bluray) to be played regardless of region or encryption. And files can be copied unencrypted. Trialware.
- DVDFab (<http://www.dvdfab.cn/>) software to rip DVDs and blurays and shrink them to size if needed. Trialware.
- MediaInfo (<https://mediaarea.net/en/MediaInfo>) – freeware tool to inspect whether video files are interlaced or progressive, resolution, framerate and other useful information. Integrated in Windows Explorer. Use the “Tree View” for exhaustive information.
- BDEdit (<https://www.videohelp.com/software/Bdedit>) – freeware tool to inspect a bluray disc structure and modify it (a bit like the DVD’s VOBedit). Not very userfriendly – some guides on YouTube.

File joining/splitting

- HJSplit (www.freebyte.com/hjsplit or <http://www.hjsplit.org/>) – a program that either joins files together (all with added filetype <name>.<type>.001 to 00n)

or splits one in a set of files of specified length (output <filename>.<type>.001 to 00n). This tool is useful for HD recordings with consumer cameras that deliver interlaced HD movies in several *.m2ts files (usually for DVD burning) that can be joined to a single .m2ts file. The original .m2ts files must be renamed into .m2ts.001 and further.

Video editing and format converting

- The standard site for video help is www.videohelp.com and www.doom9.org – their pages contain many pointers to software, guides, forums
- TS Doctor (http://www.cypheros.de/index_e.html) – a basic editor for .ts files produced by satellite receivers. It can remove audio, subtitle and Ceefax/teletext streams that are not wanted and produces a .ts or .m2ts file as output as well as use OCR to create .srt files for DVB subtitles. Make sure that at cuts you start a new section with a complete (I) frame. Shareware.
- Cuttermaran (<http://www.cuttermaran.de/en/default.aspx>) is a .mpeg-2 file editor with LPCM (wav), ac3 or DTS audio for frame-accurate cutting (requires TEMPGEnc 2.5) and Microsoft .NET Framework 3.5 SP1
- SmartCutter (<http://www.fame-ring.com/>) a paid (\$40) editor for H264, AVCHD and other video formats. Frame-accurate.
- Handbrake (<https://handbrake.fr/>) – freeware. Converts videos to different formats and bluray specific resolutions, audio-pass through, .srt subtitles but the output is .mp4 or .mkv so further transcoding is needed. May be useful to convert variable frame rate to constant frame rate in case of audio sync problems at demuxing/remuxing
- iWisoft Free Video Converter (<http://www.iwisoft.com/videoconverter/>) freeware – editor and transcoder that can convert between encodings, and has some basic editing (trim, crop).
- Adobe Premiere Pro and Elements (www.adobe.com and <https://www.adobe.com/products/premiere-elements.html>) – commercial movie editor that can output HD progressive files that are BDA compliant. The “Elements” version is a trimmed down (cheaper) version of the full-fledged Premiere Pro.
- Cyberlink Power Director (https://www.cyberlink.com/index_en_US.html) – a movie editor (both picture and sound) that can output HD progressive files in MPEG-TS stream format, that are acceptable to tsMuxer but they are not BDA compliant MPEG4-AVC streams according to BDA standards (and hence will be refused by the BDS MX internal muxer). Output consists of one audio track and burned-in subtitles. All the “produce” settings for H.264/AVC for an .m2ts output (AVC, MP4 or m2ts) deliver MP4 or MPEG-TS streams. This

output must always be transcoded to BDAV streaming format of the internal MX muxer is used.

- Wonderfox HD Video Converter Factory (<https://www.videoconverterfactory.com/dvd-video-converter/>) – free and paid versions of a HD video converter from one format into many other formats. It includes HD .m2ts format and can modify interlaced to progressive files (that conform to BDA standards for the BDS MX internal muxer). You may need to disable the “hardware acceleration” if your video card doesn’t work with this converter (producing 0 bytes sized files). The paid Pro version includes a YouTube downloader, allows basic merging of video files and cropping of those. For BDS purposes to convert to MPEG-TS .m2ts files, specify parameters for encoder (H264)
For SD movies, specify aspect ratio 16:9 if it widescreen. For others the aspect ratio doesn’t seem to work properly and ends up with a distorted image stretched or squeezed to fit 16:9 as it adds no letterbox or pillar box bars. The makers would consider a change to rectify this.
Leave bit rate and resolution as original. This contradicts that the size needs to be a BDA approved size, but BDS and my players and television sets seem to accept all sorts of weird sizes like 900x640 and apply letterbox and pillar box automatically to make them look right.
- Xmedia Recode (<https://www.xmedia-recode.de/en/>) free conversion tool to convert between many video formats.
- 3DCombine (<https://www.3dcombine.com/>) – conversion tool to create stereoscopic 3D images from 2D sources. Works for images and movies (which are sequences of images). Conversion is done via depth maps that are created using neural networks and fuzzy logic. It accepts only .mp4 or .avi files and outputs the same in a variety of formats, amongst which Half-SBS (SBS/Sensio). Conversion of a movie may take upto 25 times its run time and the movie file size may be 10 times as big. The audio in the output is distorted – you need to edit the original audio track back into the output. At the same time you can then reduce the output back to the original size (as well as convert .mp4 into .m2ts as required for demuxing for BDS).

Audio editing/format conversion

- Audacity audio editor (<http://audacity.sourceforge.net/>)
Freeware audio editor accepting most audio formats as input and capable of producing AC3 (set bitrate to 48 kHz in “Options” before save) output required by BDS. This may be handy if a tv program recording demuxes into .mp2 audio that needs to be transcoded into .ac3 or .wav output.

- WAV to AC3 Encoder (<https://github.com/wieslawsoltes/wavtoac3encoder>) – freeware tool to convert uncompressed .wav (LPCM) audio files into compressed Dolby Digital .ac3 files. Although both formats are acceptable for bluray, the .wav LPCM file can take up seven times more disc space than the .ac3 version of the same audio. For most ears the differences are neglectable.
- WAV to AC3 Encoder (<https://www.videohelp.com/software/EncWAVtoAC3>)
(<https://www.nch.com.au/switch/index.html>)
Another pair of free tools to convert WAV to .ac3 files. Both allow batch mode for converting multiple files in one go. The Australian Switch allows conversion between many formats and has a free home use option.
- AC3 to DTS converter (<https://www.videohelp.com/software/eac3to>) – freeware tool to convert various Dolby Digital .ac3 files into other formats such as DTS or FLAC.
- Fre:ac (www.freac.org) - Open source freeware tool to convert many audio formats and allows ripping of CDs and matching them with CDDb compatible database entries. .Available for Windows, Apple MacOS, Linux. Note blurays require 48 kHz sampling, not 44.1 kHz used by CD audio.

Image (menu and button) editing

- Gimp (<https://www.gimp.org/>) Freeware image editing program that resembles Photoshop capability.
- Photoshop (<http://www.adobe.com>), PaintShop Pro (<http://www.corel.com>) or any other image editing tool that allows its files with several layers to be saved in .psd format. BDS regards a button as a set of picture layers. You can make a button as a single .psd Photoshop file with multiple layers. Alternatively, you can provide each layer as separate .png file using strict naming conventions that show which file represents which layer.

Subtitle editing

- Jubler (www.jubler.org) – freeware subtitle editor allowing you to create subtitles, splitting or joining titles, shifting or stretch timings, translate titles (side-by-side with original lines) and many other features
- Subtitle Edit (<http://www.nikse.dk/subtitleedit/>) – freeware subtitle editor. Also capable of reading .sup/.idx and .ass files and change these into .srt files through a Tesseract ocr process. Sup files may occur when an .mkv or .ts file is decomposed into objects and subtitles come out as .sup which are image overlayed pictures rather than text. It can also synchronise timings of one subtitle file with another

that is known to be in sync (sync by timing). It also allows automatic translation of subtitles through an interface with Google Translate.

- 3DSubtitler (<https://www.videohelp.com/software/3DSubtitler>) – freeware distributed in a zip file. It is required to transform .srt subtitles to 3D formatted SBS or OU movie subtitles to the bluray .sup format. It requires the Java runtime environment. The Java library it also needs is included in the zip file.

MKV container production and (de)muxing tools

- MakeMKV – see section “Bluray and DVD ripping”
- MKVToolnix (<https://mkvtoolnix.download/>) – this is a freeware tool using mkvmerge to create Matroska mkv videocontainers from separate video sources. It is a sort of zip-file containing a videostream and multiple audio and subtitle streams together. Many bluray disc players but also Windows media players can play these files. An mkv file is a good candidate to check if video, audio and subtitle streams are in sync with one another. If your bluray player accepts these files and you’re not interested in menus, you may even decide to burn the .mkv files to a DVD or bluray data disc and forget about (bluray) authoring altogether. You can also (remux) your (.m2ts) video file into a new one by remuxing – often solving possible jitter problems.
- MKVCleaver (<https://www.videohelp.com/software/MKVCleaver>) – the opposite of MKVToolnix. It demuxes an .mkv file into the streams you select. Subtitle files are in various formats depending on what was inserted into the mkv file.
- tsMuxer (<http://www.videohelp.com/software/tsMuxeR>) – a freeware tool to create a .m2ts file from its separate components such as .264 (H264 video) and .ac3 (Dolby Digital audio). This “muxer” can both mux (put together) or demux (split into streams – click the “Demux” radio button, for SD resolution movies you need to rename the video files .mpv into .m2v because BDS does not recognize .mpv although the tsMuxer output is a .m2v MPEG-2 video). Also available from the Blu Disc Studio website <https://blu-disc.net/download/tsMuxeR.rar>
- tsDemuxer (<https://www.videohelp.com/software/tsDemux>) – freeware tool to split a .ts or .m2ts file into its components.
- tsMuxerGUI (<https://www.videohelp.com/software/tsMuxeR>) – a Windows friendly interface to the commandline tsMuxer tool. Can also demux.

- MKVExtractGUI2 – (<https://sourceforge.net/projects/mkvextractgui-2/>) – freeware tool to extract objects from an MKV file.

Software video and bluray players

- VLC player (<http://www.videolan.org/vlc/>) – this is a freeware tool that seems to play almost any format video and/or audio file, including mkv, ts, mpeg or mp4 video and ac3 or wav audio. It allows you (via right-mouse click) to select the audio or subtitle stream of your choice. It plays both discs (DVD or bluray) as well as single files from the \streams folder. Menu Media>Open disc (and browsing to the bluray disc drive or navigate to the \output folder created by BDS) will play the bluray structure. It is important to have the Java runtime installed. For VLC 64 bits you need the JVM 64 bits, for the older VLC 32 bits you need JVM 32 bits.
One note of warning: the developers apparently do not find it important, but if you have an .m2ts file in a folder and an external subtitle file for that .m2ts file, VLC won't play at all. This has been the case since V1.0. Remove the subtitles (move to another folder) or rename .srt to .srtXX and the .m2ts file will play – also if it has internal DVB subtitles selected for display.
- Cyberlink PowerDVD (https://www.cyberlink.com/products/powerdvd-ultra/features_en_US.html) – paid software player (does not do bluray screen captures as Cyberlink considers this infringements of license agreements with the BDA). Often available in a “suite” of Cyberlink video tools, but available separately also. Plays most BDS produced discs flawlessly. Has a problem to sometimes position optional subtitles correctly.
- Leawo Bluray Player (<http://www.leawo.com/bluray-player/>) – a freeware bluray player capable of playing bluray discs on a pc bluray device but also to play individual bluray files in the \streams folder. It requires Windows XP or later.
- Nero 12 bluray player – discontinued part of commercial Nero burning package (available up to V12). Has a few quirks refusing to play legitimate BDS MX output.
- Bluray Master (<http://www.bluraycopys.com/free-bluray-player/>) free bluray player that plays discs as well as bluray structures on disc – as iso image or separate folders (use “Open disc” button).

Screen Capture

The BDA frowns on any way a bluray film might be snapshot from a bluray disc image stored on hard disc or on a true bluray disc. Hence tools like Cyberlink's PowerDVD (from V7 onwards) cannot snapshot anything bluray, whether it is commercial or your own movie.

“Paranoid” seems to be the word for the money-making Hollywood moguls. And Cyberlink complies.

Most software bluray players will use Direct Write for movies: a rectangle reserved on screen that is directly controlled by the video card and not by the software player. Therefore, if you use screen grabbing software it usually shows up as a black rectangle.

Player VLC can do screen grabs if you play any of the produced \STREAM\nnnnn.m2ts files. Each .m2ts file is a complete muxed movie with all audio and subtitle streams. VLC will do image grabs through SHIFT/S to a set location (Use Preferences to set)

The freeware Leawo software player (<https://www.leawo.org/blu-ray-player/>) can play your disc folders or discs and allows for screen captures (but in my experience skips bluray menus altogether).

Bluray burning software

- ImgBurn (<http://www.imgburn.com>) – a freeware program to burn a folder or .iso image to a physical disc (CD, DVD or bluray) provided the files or .iso image to burn are compliant for the medium to burn it to. ImgBurn also allows to create an .iso file from a compliant folder. Hence the Blu Disc Studio output (a set of files in bluray specific folders) can be converted to an .iso file or be burned directly to a bluray disc.
- Nero BurningROM – commercial package including bluray and DVD burning templates as well as complete suite for audio and video editing and bluray creation (but limited freedom in how to do this).
- Cyberlink Power2Go will burn DVD, Bluray and AVCHD disc images. Paid version.
- BD Rebuilder (<http://www.videohelp.com/software/BD-Rebuilder>) – freeware tool to resize a bluray folder to fit a BD-25, DVD-9 or DVD-5 sized disc before burning.

Virtual disc drives

- Virtual Clonedrive (<https://www.redfox.bz/virtual-clonedrive.html>) – virtual disc drive station freeware – program that allows you to “play” an .iso image of a bluray disc as if it ran from a real bluray drive inside your computer. You can mount the .iso file and suddenly a new drive letter appears (such as Y:\ that is “running” your disc. Useful program if your software player only accepts “real” bluray discs and cannot play a bluray folder. But creating a .miniso file with DVDFab Explorer may be quicker and more useful.
- DVDFab Explorer (<https://www.dvdfab.cn/explorerfab.htm>) – virtual drive station freeware from DVDFab (competitor of Redfox). It allows to create iso files and/or open them as virtual disc. It also allows creation of an .miniso file to

simulate a file folder (plus subfolders) as it is were a virtual disc. This allows playback of these folders by playback applications that can only handle discs. The ability to create a .miniso file is essential for creating proper 3D bluray discs with BDS.

- Cyberlink Power2Go has a virtual disc drive feature under its “Utilities”. Paid version.

Disc Cover Designers

- Nero CoverDesigner (www.nero.com/eng/downloads) – freeware tool to create CD, DVD and bluray covers, booklets as well as disc images to print on printable discs. Obsolete since 2019 but that version still works
- Canon ImageGarden (https://printinginnovations.cusa.canon.com/pixma/my_image_garden) – comes on CD with a Canon printer capable of printing discs but can also be downloaded from the Canon website.
- Epson Photo+ Free download but also comes with Epson printers. It has a CD/DVD label printing option. It requires an Epson printer driver is installed.
<https://support.epson.net/appinfo/photoplus/win/en/top.htm>

Appendix F: Java programming

If you are serious about extensive programming in Java to add functionality to your BDS program, you should become fluent in the Java language. There are many online courses. You will find them if you google on terms like “Java course”, “Simple Java” etc. Note that Java as proper programming language is not the same as the web page scripting language Javascript. To make it confusing, BDS talks in some menus and the online help about “Java (script)” but mean “script BDS functionality in Java”.


Interaction BDS – Java

BDS is written in Java to produce a BD-J (bluray Java) disc. The user interface allows you to define and perform most functions by selecting menu options or setting checkboxes.

User added Java code is integrated in the main body of BDS Java code in these locations:

- Project > Project Settings > Functions tab
Here you define user defined variables and functions. They are known throughout the project. A variable defined here can be used in an action or a menu (referred to as *manager.UDV_name* in Java). A function is defined as *manager.UDF_name*.
However, UDF functions should refer to UDV variables and BDS functions **without** the *manager*. Prefix (the UDF executes within the BDS manager object so it should not use the name of that object).
- Actions (and multi-actions and Switch) defined as “Script (Java)”. Others actions can convert to their Java code by selecting the action and select “Script (Java)”. Once this is done, you cannot undo it.

Compilation of the code to check on its correctness is a bit more cumbersome than using plain Java. The compiler is an integrated part of BDS and uses a lot of internally defined data types and functions (libraries). You compile your code by closing all script windows and

press the  button. This compiles all code and, if correct, bundles it into Java Archive files (JAR). If errors are found, they are reported in the log window and log file. Often, they indicate syntax errors and you need to correct them. If it compiles but functions unexpectedly, debugging is nearly impossible within BDS as it fails to provide for a “line-by-line” debugging tool common to ordinary virtual Java machines.

It’s advised to add small bits of Java code each time (such as for one action of an object only) and see if it still compiles by recreating the JAR file. You then limit the search area when errors occur. You can use `System.out.println(“string”)` to output information to the system console. That is the log window when you enter a simulation

Important: The project.bdmd file is in fact an XML file. If you open it in Notepad you can quickly edit the script parts that are all marked as <script>. Before you make any changes this way, make a copy of the original .bdmd file as backup so you can (re)use it if your modifications corrupt the .bdmd file. You can make as many copies of the .bdmd file as you like as long as they have unique names. This way you could have a “standard” project on which you make some modifications that you save as a project under a different name (File > Save As...). You may also look in the hidden _History folder of your project: everytime an implicit save operation is executed (when you start muxing) a copy of the project.bdmd file is saved in that folder.

If you’re unfamiliar with Java or Java, but used to other programming languages, some of the most likely pitfalls in your coding are:

- Different casing for variables and/or menu objects
- Incorrect casing for function names
- Not using = for assignment and == for comparison
- Forget to end an instruction with a semi-colon (;)
- Not defining all variables upfront before you use them

A simple error (such as forgetting a “}” delimiter) may cause a lot of other fake errors further downstream because a statement block did not end properly and what follows should not happen and is illegal. It can be a long and tedious search using BDS for compilation. In the end you may want to open each script and copy it into a Notepad file and save it. Doing this for all scripts and removing them one by one (or adding them back one by one) may show which module causes the compilation errors. That allows for a somewhat more targeted search of your syntax error.

Loading a copy of the project.bdmd file in some text editor that allows for line numbers (such as Microsoft Visual Studio) may also help to find the error that BDS finds at line 789...

Basic Java

Just as a “cheat sheet”, below you may find the most common Java declarations, expressions and conditional constructs. The brackets [] indicate those parts of the declaration are optional. Note that Java code is case sensitive. Apart from C, C++ or C# most languages are not. But “Hello”, “hello”, “HeLlo” are three different items to Java.

Primitive (simple) data types

- int – integer (32 bits) – like 5, -10, up to $\frac{1}{2}(2^{32}-1) = \pm 2\,147\,483\,648$
- long – long integer (64 bits) – up to $\frac{1}{2}(2^{64}-1) = \pm 9\,223\,372\,036\,854\,775\,807$
- float – floating point real number (32 bits) up to $\pm 3.40\,E+38$ and down to $\pm 1.40\,E-45$
- double – floating point real number (64 bits) – up to $1.79\,E+308$ and down to $4.94\,E-324$

- char – single character (16 bits) like 't' (single quotes). Special characters (that otherwise have special meaning in Java): \" (double quote), \' (quote), \\ (backslash) \xxx (octal value of the character), \uxxxx (hexadecimal value of character)
- boolean – has true or false value
- String – string of characters like "text" (double quotes!)
Strictly speaking a String is a complex data type of a number of char variables.

Arrays

Arrays belong to the complex data types of Java. But are often used to store a list of similar data. In object-oriented terms they form a class of their own, but in non-object-oriented fashion arrays have been a part of most programming languages.

In Java arrays are indexed from 0 upwards.

Variable declaration

Type name [, name];

Example: int number1, number2;

Variables can be given an initial value adding =:

Type name = value [, name [=value] ;

Arrays are defined by specifying their dimensions. BDS does not permit variable definition and use at the same time. Declaration and use must be split.

Hence:

1-dimensional:

*Type name[]
name = new type[dimension-size];*

2-dimensional:

*Type name[][];
name = new type [size1][size2];*

For example:

```
byte myList[]
myList = new byte[1024];
int my3D[][][]
my3D = new int[10][20][30]; // a 10x20x30 array
```

A multidimensional array is an array of an array of ... Thereby the only property of an array (its number of elements), myArray.length, for multidimensional arrays is given for the first dimension as myArray.length. Each element is an array of its own dimension, whose size is found as the length of that element. Hence for a 2-dimensional array myArray[10][5] each of the 10 elements in the first dimension has an array of 5 elements as second dimension: myArray[i].length (where $0 \leq i \leq 9$).

Variable assignment

To use the value stored in a variable, you assign its value to some other variable. Both must have compatible data types. The equal sign (“=”) is used for assignment (unlike some languages that use “:=”).

Some examples:

```
name = other_variable;

for (int j=0; j < myArray.length; j++)
    { myArray[j] = 4*j; }

for (int i=0; i<myArray.length; i++)
    for (int j=0; j<myArray[i].length, j++)
        { myArray[i][j] = i+j ; }
```

Comments

Code is easier to maintain when in future comments may remind the programmer why things were done the way they were done. Or explain the logic of the code.

- `// inline comment` – anything following on the same line is considered comment
- `/* comment text */` - comments that spread lines. Anything between the two markers is considered comment

Statements and Expressions

- A statement appears inside a method (routine) and describe the activity of a Java program
- Expressions evaluate to a value. They are part of other expressions or statements

Statements and expressions are written as “code block” and surrounded by curly braces { }. Variables defined within such a block are only known within that block.

```
{
    int NrWindows = 6;
    CleanWindow('main menu');
}
```

Loop

- `for (initial ; condition ; step) { statement; [statement;] }`

The “initial” defines the loop counter (e.g. `int i = 1;`), the condition indicates the condition to exit the loop (e.g. `i > 200;`) and the step indicates the increase of the loop counter for each loop (e.g. `i++` or `2`).

Example: `for (int j=5; j<-10; -1) { statement }` loops for `j=5` in steps of `-1` down to `-10`.

Conditional statements

- `if (boolean condition) { statement; [statement;] }`
`[else { statement; [statement;] }]`
- `while (boolean condition) { statement; [statement;] }`

The (*boolean condition*) is an expression that must result in a true or false value.

- ```
switch (int expression) {
 case int expression : {statement ; break;}
 [case int expression : {statement; break;}]
 [default : statement;}
}
```

The switch *expression* must match with one of the case expressions to execute the statement. If none match, the default statements are executed. The case expression must have a fixed value at compile time (cannot have variables whose values may be known only during the execution of the program). Each case statement needs to end with a **break** statement to avoid continuing with the next line (odd behaviour of Java). Rather than an integer expression, also other datatypes can be used in the switch condition, such as a string.

Example:

```
String month;
Int monthNumber; // this value may vary during execution
switch (monthNumber) {
 case 1 : {month = "January"; break;} // the "1" value is fixed
 case 2 : {month = "February"; break;}
 default : {month = "not Jan or Feb";}
}
```

A number of case values can be combined to do the same thing, e.g. to count days in a year (allowing for leap years):

```
int month = 2;
int year = 2000;
int numDays = 0;

switch (month) {
 case 1: case 3: case 5:
 case 7: case 8: case 10:
 case 12:
 numDays = 31;
 break;

 case 4: case 6:
 case 9: case 11:
 numDays = 30;
 break;

 case 2:
 if (((year % 4 == 0) && !(year % 100 == 0))
 || (year % 400 == 0))
 numDays = 29;
```



```

 else
 numDays = 28;
 break;

 default:
 System.out.println("Invalid month.");
 break;
}

```

## Operators

++ increment by one (i++)  
 -- decrement by one (i- -)  
 + add (also string concatenation)  
 - subtract  
 \* multiply  
 / divide  
 ! inverse logical (not)  
 == equal  
 != not equal  
 && conditional and  
 || conditional or  
 <, <=, >, >= comparison smaller, smaller-equal, larger, larger-equal

## Methods

Methods are Java's functions and subroutines.

```

[type] methodname ([argument] [; argument])
 { statement; [statement]; }

```

If the type is "void" (returning nothing) the method is a subroutine. Otherwise it has the datatype that is its value when the method completes. This value is returned through the "return" statement. When this statement executes, the remainder of the method is skipped. A "void" method can use the "return;" to skip executing the remainder of the method also (returning no value).

Examples:

```

void Hello () { System.out.println ("Hello world"); }

int Increment (int Nr) { Nr = Nr +1; return Nr; }

```

The system output console is diverted to the log window when you start a simulation. Using the println statements can help you to see the progress made through the script code you wrote or the values some variables are supposed to have.

## Java functions in BDS

A number of functions have been developed for the BDS application and some of them are also available for the disc author/developer. You can find them in the online help section "Available Java functions (methods)". This list is reproduced here, sorted on type of expected usage. The type indicates whether it is a function that returns

something or whether it is just called (void). And although it is called “Java functions” they really are Java functions!

All “movie name” arguments are the names you have specified for the movie placeholders: that’s the only “stable” item of a movie fort BDS. Whatever real movie file name is currently associated with it is of no importance.

Menus and movies defined in your project must follow naming rules:

- “**S:MM\_menuname.objectname**” for a menu object
- “**S:PM\_menuname.objectname**” for a popup menu object
- “**H:MM\_menuname.Handler.objectname**” ” is used for a menu handle (often a button)
- “**H:PM\_menuname.Handler.objectname**” ” is used for a popup menu handle (often a button)
- “**MV\_moviename**” for a movie
- More naming rules are found in the online Help of BDS under “Script (java)”

All BDS exposed functions are referred to as *manager.functionname*. BDS itself runs as a big object “manager” and to use its function you must address this manager object in Java written for actions of menus and buttons.

The menu names can be appended by .show\_menu or .animateN (N=1,2,3...) to open the menu sec or with execution of one of its [anm/act] settings through the use of the manager.activateSegment function. Examples:

- “S:MM\_TitleChoice.show\_menu” (for regular menu)
- “S:MP\_TitleChoice.animate4” (for a popup menu with at least 4 ways to open it)

```
manager.activateSegment("S:MM_Bookmarks.animate2");
```

or (without any animation)

```
manager.activateSegment("S:MM_Bookmarks.show_menu");
```

A menu is usually opened with a specific button as selected one. The “Press ENTER” code can be executed (exec = true) or not (then it is simply selected) through the manager.activateButtonEx function. A button is a handler of the menu.

```
manager.activateButtonEx ("H:MM_Bookmarks.Handler",
 "Button", false);
```

However, when you use a User Defined Function (UDF) they already run within the BDS manager object. Therefore, the code should omit the “*manager.*” prefix. Also when code is moved from a menu or button action script to a UDF, the code should be modified by removing all references to *manager* (but BDS removes these internally when they are found in UDF scripts).

## Generic Java functions

Generic Java functions can also be used in your scripting code. Sometimes they are different from the regular Java language functions, so do check you're using Java functions (the BDS compiler will issue errors if you don't). You may want to visit <https://javadevnotes.com> for some guidelines or examples.

## Conversion functions (Java)

| Function                                           | purpose                                                                                                                            |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| <b>double</b> Double.valueOf( <i>stringvalue</i> ) | returns a double value of a valid string double number<br>String myNumber = "123.54";<br>double number = Double.valueOf(myNumber); |
| <b>String</b> Integer.toString( <i>number</i> )    | returns a String value of an integer number<br>int number = 451;<br>myString = Integer.toString(number);                           |
| <b>int</b> Integer.valueOf( <i>stringvalue</i> )   | returns an integer value of a valid string number<br>String myNumber = "1234";<br>int number = Integer.valueOf(myNumber);          |
| <b>long</b> Long.valueOf( <i>stringvalue</i> )     | returns a long value of a valid string long number                                                                                 |
| <b>String</b> String.valueOf( <i>number</i> )      | returns a String value of a number (int, float, long)                                                                              |

## Arithmetic functions (Java)

| Function                                                                      | purpose                                                       |
|-------------------------------------------------------------------------------|---------------------------------------------------------------|
| <b>int/long/float/double</b> manager.abs( <b>int/long/float/double</b> value) | returns an absolute (positive) value                          |
| <b>int</b> manager.random( <b>int</b> maxValue)                               | returns a random value                                        |
| <b>int/long</b> manager.round( <b>float/double</b> value)                     | returns a rounded value                                       |
| String manager.time2str( <b>long</b> time)                                    | converts a time in nanoseconds to the string in form 01:23:45 |

## Menu functions

| Function                                                                     | purpose                                                                                                                               |
|------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <b>boolean</b> manager.activateSegment(String strSegment)                    | activates the menu or movie segment, e.g. strSegment = "S:PM_Timeline.animate1" or strSegment = "S:MM_main.show_menu"                 |
| <b>void</b> manager.Close_Popup()                                            | executes the [close popup] action                                                                                                     |
| <b>int</b> manager.getX(String featureName)                                  | returns X coordinate of the specified feature                                                                                         |
| <b>int</b> manager.getY(String featureName)                                  | returns Y coordinate of the specified feature                                                                                         |
| <b>boolean</b> manager.IsMenuOnScreen()                                      | returns true if there is a menu on screen (if there is no Movie segment)                                                              |
| <b>void</b> manager.moveToX(String featureName, <b>int</b> x)                | moves the specified feature to the specified coordinate along the X axis (you should specify a move container for the object effects) |
| <b>void</b> manager.moveToXY(String featureName, <b>int</b> x, <b>int</b> y) | moves the specified feature to the specified coordinates (you should specify a move container for the object effects)                 |
| <b>void</b> manager.moveToY(String featureName, <b>int</b> y)                | moves the specified feature to the specified coordinate along the Y axis (you should specify a move container for the object effects) |
| <b>void</b> manager.moveX(String featureName, <b>int</b> dx)                 | moves the specified feature for the specified delta along the X axis (you should specify a move container for the object effects)     |
| <b>void</b> manager.moveXY(String featureName, <b>int</b> dx, <b>int</b> dy) | moves the specified feature for the specified deltas (you should specify a move container for the object effects)                     |

| Function                                                                                          | purpose                                                                                                                                                                     |
|---------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>void</b> manager.moveY(String featureName, int dy)                                             | <i>moves the specified feature for the specified delta along the Y axis (you should specify a move container for the object effects)</i>                                    |
| <b>void</b> manager.playVideo(int playlistId)                                                     | <i>same as a startVideo, but resumes the playlist if possible</i>                                                                                                           |
| <b>void</b> manager.playVideo(#string movieName)                                                  | <i>same as above</i>                                                                                                                                                        |
| <b>void</b> manager.playVideoAt(int playlistId)                                                   | <i>same as a startVideoAt, but resumes the playlist if possible</i>                                                                                                         |
| <b>void</b> manager.playVideoAt(#string movieName)                                                | <i>same as above</i>                                                                                                                                                        |
| <b>void</b> manager.playVideoFrom(int playlistId, long time)                                      | <i>starts the specified playlist from the specified time in nanoseconds</i>                                                                                                 |
| <b>void</b> manager.playVideoFrom(#string movieName, long time)                                   | <i>same as above</i>                                                                                                                                                        |
| <b>void</b> manager.scaleToX(String featureName, int x)                                           | <i>scales the specified feature to the specified scale in mills along the X axis (you should specify a scale effect for the object state)</i>                               |
| <b>void</b> manager.scaleToXY(String featureName, int x, int y)                                   | <i>scales the specified feature to the specified scale in mills (you should specify a scale effect for the object state)</i>                                                |
| <b>void</b> manager.scaleToY(String featureName, int y)                                           | <i>scales the specified feature to the specified scale in mills along the Y axis (you should specify a scale effect for the object state)</i>                               |
| <b>void</b> manager.scaleUHDToHalf(int Left, int Top)                                             | <i>scales the video from UltraHD to the half</i>                                                                                                                            |
| <b>void</b> manager.scaleUHDToQuarter(int Left, int Top)                                          | <i>scales the video from UltraHD to the quarter</i>                                                                                                                         |
| <b>void</b> manager.scaleVideoSize(int L, int T, int W, int H, int _L, int _T, int _W, int _H)    | <i>custom scale from L, T, W, H to _L, _T, _W, _H</i>                                                                                                                       |
| <b>void</b> manager.scaleX(String featureName, int dx)                                            | <i>changes the scale of the specified feature by the specified delta in mills along the X axis (you should specify a scale effect for the object state)</i>                 |
| <b>void</b> manager.scaleXY(String featureName, int dx, int dy)                                   | <i>changes the scale of the specified feature by the specified delta in mills (you should specify a scale effect for the object state)</i>                                  |
| <b>void</b> manager.scaleY(String featureName, int dy)                                            | <i>changes the scale of the specified feature by the specified delta in mills along the Y axis (you should specify a scale effect for the object state)</i>                 |
| <b>void</b> manager.setAlpha(String featureName, double alpha)                                    | <i>changes the transparency for the specified feature (alpha should be from 0 to 255; you should specify a transparency effect for the object state)</i>                    |
| <b>void</b> manager.setClipping(String featureName, int left, int top, int width, int height)     | <i>changes the clipping (cropping) for the specified feature (you should specify a clipping effect for the object state)<br/>e.g. featureName = "F:PM_Timeline.Timebar"</i> |
| <b>void</b> manager.setClippingRect(String featureName, int left, int top, int right, int bottom) | <i>changes the clipping (cropping) for the specified feature (you should specify a clipping effect for the object state)</i>                                                |
| <b>void</b> manager.setText(String featureName, String text)                                      | <i>changes the first line of the text for the text feature</i>                                                                                                              |
| <b>void</b> manager.setText(String featureName, String text, int index)                           | <i>changes the specified line of the text for the text feature (index starts from 0)</i>                                                                                    |
| <b>void</b> manager.syncDisplay()                                                                 | <i>synchronizes graphics before long operations</i>                                                                                                                         |
| <b>void</b> manager.UpdateCurrent_...()                                                           | <i>updates the current state in the specified menu<br/>example:<br/>manager.UpdateCurrent_PM_Popup_Sound().</i>                                                             |

## Button functions

| Function                                                                | purpose                                                                                   |
|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| <b>void</b> manager.activateButton(String strHandler, String strButton) | <i>activates the button (strButton) in the menu specified by the handler (strHandler)</i> |

| Function                                                                                 | purpose                                                                                                                                   |
|------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                          | e.g. strHandler = "H:PM_Timeline.Handler" and strButton = "myButton"                                                                      |
| <b>void</b> manager.activateButtonEx(String strHandler, String strButton, boolean bExec) | <i>the same as the previous one, but allows to execute the Enter action of the button (it does not if bExec=false)</i>                    |
| <b>boolean</b> manager.isButtonSelected(String strAssembly, String strButton)            | <i>returns true if the specified button is selected in the menu, specified by the assembly name</i>                                       |
| <b>boolean</b> manager.isSFXoff()                                                        | <i>returns true if the button sounds are off</i>                                                                                          |
| <b>boolean</b> manager.isSFXon()                                                         | <i>returns true if the button sounds are on</i>                                                                                           |
| <b>boolean</b> manager.isTopMenuPressed()                                                | <i>returns true if the TopMenu button was pressed (used during initialization process)</i>                                                |
| <b>void</b> manager.playSFX(String strLocator)                                           | <i>plays sound of the button using the specified BDLocator (the sounds are numbered from 0) example: manager.playSFX("bd://SOUND:01")</i> |
| <b>void</b> manager.setAutoButton(String strValue)                                       | <i>sets the auto button which will be executed after the transition to the new menu</i>                                                   |
| <b>void</b> manager.setSoundState( <b>boolean</b> stateON)                               | <i>sets the sound state for button sounds</i>                                                                                             |

### Primary (main) Movie functions

argument "movieName" refers to the name of the movie placeholder

Using the name in single quotes allows the BDS preprocessor to replace the movie name by its ID. For example, argument 'movieA' will be replaced by number 14 (if its BDID=14)

| Function                                                                                 | purpose                                                                                                                                                                                          |
|------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>boolean</b> manager.activateSegment(String strSegment)                                | <i>activates the menu or movie segment</i>                                                                                                                                                       |
| <b>long</b> manager.duration                                                             | <i>Returns the duration of current playlist in nanoseconds (<b>not</b> a function but a value saved on playlist start)</i>                                                                       |
| <b>void</b> manager.fastForward()                                                        | <i>sets the current playback rate to the one of the fast forward values, depending on the current playback rate, the values can be specified in the project properties</i>                       |
| <b>int</b> manager.getAudioID( <b>int</b> playlistId)                                    | <i>returns the current audio track number (index) for the specified playlist (movie)</i>                                                                                                         |
| <b>int</b> manager.getAudioID(#string movieName)                                         | <i>same as above</i>                                                                                                                                                                             |
| <b>int</b> manager.getChapter( <b>int</b> playlistId)                                    | <i>returns the current chapter number (index) for the specified playlist (movie), starting from 1 (1 – first chapter, 0 – the playlist was not started or ended (the resume time was reset))</i> |
| <b>int</b> manager.getChapter(#string movieName)                                         | <i>same as above</i>                                                                                                                                                                             |
| <b>int</b> manager.getChapterNumberByTime( <b>int</b> playlistId, <b>long</b> time)      | <i>returns the current chapter number (index) for the specified playlist (movie), detecting the chapter number by the specified time (note: the value is approximate);</i>                       |
| <b>int</b> manager.getChapterNumberByTime(#string movieName, <b>long</b> time)           | <i>Same as above</i>                                                                                                                                                                             |
| <b>long</b> manager.getChapterTimeByNumber( <b>int</b> playlistId, <b>int</b> chapterId) | <i>returns time in nanoseconds for the specified chapter (index) of specified playlist (movie)</i>                                                                                               |
| <b>long</b> manager.getChapterTimeByNumber(#string movieName, <b>int</b> chapterId)      | <i>Same as above</i>                                                                                                                                                                             |
| <b>int</b> manager.getCurrentAngle()                                                     | <i>returns the current angle for the current playlist</i>                                                                                                                                        |
| <b>int</b> manager.getCurrentAudioID()                                                   | <i>returns the current audio track number (index) for current playlist (movie)</i>                                                                                                               |
| <b>int</b> manager.getCurrentChapter()                                                   | <i>returns the current chapter for current playlist (movie), starting from 1 (1 – first chapter)</i>                                                                                             |
| <b>String</b> manager.getCurrentSegmentName()                                            | <i>returns the current segment name</i>                                                                                                                                                          |

| Function                                                     | purpose                                                                                                           |
|--------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| <b>int</b> manager.getCurrentSubID()                         | returns the current subtitle track number (index) for the current movie, 0 – means subtitles is off               |
| <b>int</b> manager.getCurrentSubNumber()                     | returns active subtitle track number (index) for the current movie, ignoring on/off state                         |
| <b>int</b> manager.getCurrentSubState()                      | returns true if subtitles are on in the current movie                                                             |
| <b>long</b> manager.getDuration()                            | returns the duration for the current playlist in nanoseconds; (see also: manager.duration)                        |
| <b>long</b> manager.getDurationInSeconds()                   | returns the duration for the current playlist in seconds (see also: manager.duration)                             |
| <b>long</b> manager.getMediaTime()                           | returns the time for the current playlist from player in nanoseconds                                              |
| <b>long</b> manager.getMediaTimeInSeconds()                  | returns the time for the current playlist from player in seconds                                                  |
| <b>float</b> manager.getPlaybackRate()                       | returns the current playback rate (1 = play, 0 = pause, negative = rewind)                                        |
| <b>int</b> manager.getPlaylistID()                           | returns the current playlist number (index)                                                                       |
| <b>long</b> manager.getResumeTimeReg( <b>int</b> playlistId) | returns the resume time for specified playlist (movie) in nanoseconds                                             |
| <b>long</b> manager.getResumeTimeReg(#string movieName)      | same as above                                                                                                     |
| <b>int</b> manager.getSubID( <b>int</b> playlistId)          | returns the current subtitle track number (index) for the specified playlist (movie), 0 - means subtitles is off  |
| <b>int</b> manager.getSubID(#string movieName)               | same as above                                                                                                     |
| <b>int</b> manager.getSubNumber( <b>int</b> playlistId)      | returns active subtitle track number (index) for the specified playlist (movie), ignoring on/off state            |
| <b>int</b> manager.getSubNumber(#string movieName)           | same as above                                                                                                     |
| <b>boolean</b> manager.getSubState( <b>int</b> playlistId)   | returns true if subtitles are on in the specified playlist                                                        |
| <b>boolean</b> manager.getSubState(#string movieName)        | same as above                                                                                                     |
| <b>boolean</b> manager.is2D()                                | returns true, if the current mode is 2D                                                                           |
| <b>boolean</b> manager.is3D()                                | returns true, if the current mode is 3D                                                                           |
| <b>boolean</b> manager.isFastForward()                       | returns true if the current playback rate is fast forward                                                         |
| <b>boolean</b> manager.isPaused()                            | returns true if the current playback rate is "paused" (playback rate = 0)                                         |
| <b>boolean</b> manager.isPlaying()                           | returns true if the current playback rate is "played" (playback rate = 1)                                         |
| <b>boolean</b> manager.isPlaylist( <b>int</b> playlistId)    | returns true if the specified playlist (movie) is playing                                                         |
| <b>boolean</b> manager.isPlaylist(#string movieName)         | same as above                                                                                                     |
| <b>boolean</b> manager.isRewind()                            | returns true if the current playback rate is rewind                                                               |
| <b>boolean</b> manager.isSupportsStereo()                    | returns true if the 3D mode is supported                                                                          |
| <b>void</b> manager.jumpNextMark( <b>int</b> markCount)      | jumps to the next play mark (chapter) in the current playlist (movie), markCount the maximum number of chapters   |
| <b>void</b> manager.jumpNextMark()                           | jumps to the next play mark (chapter) in the current playlist (movie)                                             |
| <b>void</b> manager.jumpPrevMark( <b>int</b> minMark)        | jumps to the previous play mark (chapter) in the current playlist (movie), minMark the minimum number of chapters |
| <b>void</b> manager.jumpPrevMark()                           | jumps to the previous play mark (chapter) in the current playlist (movie)                                         |
| <b>void</b> manager.jumpPlay mark( <b>int</b> markId)        | jumps to the specified play mark (chapter) in the current playlist (movie)                                        |
| <b>void</b> manager.jumpPlay mark(#int markNumber)           | same as above                                                                                                     |

| Function                                                                                                  | purpose                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>void</b> manager.jumpTime(int timeInSeconds)                                                           | jumps to the specified time in the current playlist (movie)                                                                                                   |
| <b>void</b> manager.pauseMovie()                                                                          | pauses the current movie (sets playback rate to 0)                                                                                                            |
| <b>void</b> manager.playMovie()                                                                           | resumes the current (paused) movie (sets playback rate to 1)                                                                                                  |
| <b>void</b> manager.playPause()                                                                           | switches between pauseMovie and playMovie                                                                                                                     |
| <b>void</b> manager.playSFX(String strLocator)                                                            | plays sound of the button using the specified BDLocator (the sounds are numbered from 0); example: manager.playSFX("bd://SOUND:01");                          |
| <b>void</b> manager.playVideo(int playlistId)                                                             | same as a startVideo, but resumes the playlist if possible                                                                                                    |
| <b>void</b> manager.playVideo(#string movieName)                                                          | same as above                                                                                                                                                 |
| <b>void</b> manager.playVideoAt(int playlistId)                                                           | same as a startVideoAt, but resumes the playlist if possible                                                                                                  |
| <b>void</b> manager.playVideoAt(#string movieName)                                                        | same as above                                                                                                                                                 |
| <b>void</b> manager.playVideoFrom(int playlistId, long time)                                              | starts the specified playlist from the specified time in nanoseconds                                                                                          |
| <b>void</b> manager.playVideoFrom(#string movieName, long time)                                           | same as above                                                                                                                                                 |
| <b>void</b> manager.resetResumeTime(int playlistId)                                                       | clears the resume time for the specified playlist (movie)                                                                                                     |
| <b>void</b> manager.resetResumeTime(#string movieName)                                                    | same as above                                                                                                                                                 |
| <b>void</b> manager.rewind()                                                                              | sets the current playback rate to the one of the rewind values, depending on the current playback rate, the values can be specified in the project properties |
| <b>void</b> manager.saveAudio(int playlistId, int streamNumber)                                           | saves the specified audio track number (index) for the specified playlist (movie)                                                                             |
| <b>void</b> manager.saveAudio(#string movieName, int streamNumber)                                        | same as above                                                                                                                                                 |
| <b>void</b> manager.saveAudioSubtitle(int playlistId, int streamAudio, int streamSubtitle, boolean on)    | saves the specified audio and subtitle track number (index) for the specified playlist (movie)                                                                |
| <b>void</b> manager.saveAudioSubtitle(#string movieName, int streamAudio, int streamSubtitle, boolean on) | same as above                                                                                                                                                 |
| <b>void</b> manager.saveChapter(int playlistId, int chapterNumber)                                        | saves the specified chapter number (index) for the specified movie in GPR registers                                                                           |
| <b>void</b> manager.saveChapter(#string movieName, #int chapterNumber)                                    | same as above                                                                                                                                                 |
| <b>void</b> manager.saveCurrentResumeTime()                                                               | saves the current time of the current movie for resume                                                                                                        |
| <b>void</b> manager.saveMediaState()                                                                      | saves the current menu/movie state, use it with try/catch                                                                                                     |
| <b>void</b> manager.saveResumeTime(int playlistId, long mediaTime)                                        | saves the specified resume time for a given playlist (movie) for resume                                                                                       |
| <b>void</b> manager.saveResumeTime(#string movieName, long mediaTime)                                     | same as above                                                                                                                                                 |
| <b>void</b> manager.saveSubtitle(int playlistId, int streamNumber, boolean on)                            | saves the specified subtitle track number (index) for the specified playlist (movie)                                                                          |
| <b>void</b> manager.saveSubtitle(#string movieName, int streamNumber, boolean on)                         | same as above                                                                                                                                                 |
| <b>void</b> manager.scaleBack()                                                                           | scales the video to the original size                                                                                                                         |
| <b>void</b> manager.scaleFullToHalf(int Left, int Top)                                                    | scales the video from FullHD to half the length and width (in fact ¼ of the screen area)                                                                      |
| <b>void</b> manager.scaleFullToQuarter(int Left, int Top)                                                 | scales the video from FullHD to width and height at a quarter of full screen (hence 1/16 <sup>th</sup> of the screen area)                                    |
| <b>void</b> manager.selectAngle(int streamNumber)                                                         | selects (sets) the specified angle for the current playlist                                                                                                   |
| <b>void</b> manager.selectAudio(int streamNumber)                                                         | selects the specified audio track number (index) in the current movie                                                                                         |

| Function                                                                                                                                                              | purpose                                                                                                                                                              |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>void</b> manager.selectAudioStream( <b>int</b> playlistId, <b>int</b> streamNumber, <b>boolean</b> immediate)                                                      | selects the specified audio track number (index) in a given playlist (movie) and in the current playlist it if immediate is true                                     |
| <b>void</b> manager.selectAudioStream(#string movieName, <b>int</b> streamNumber, <b>boolean</b> immediate)                                                           | same as above                                                                                                                                                        |
| <b>void</b> manager.selectAudioSubtitleStream( <b>int</b> playlistId, <b>int</b> streamAudio, <b>int</b> streamSubtitle, <b>boolean</b> immediate)                    | selects the specified audio and subtitle track number (index) in a given playlist (movie) and in the current playlist it if immediate is true (subtitle 0 means off) |
| <b>void</b> manager.selectAudioSubtitleStream(#string movieName, <b>int</b> streamAudio, <b>int</b> streamSubtitle, <b>boolean</b> immediate)                         | same as above                                                                                                                                                        |
| <b>void</b> manager.selectAudioSubtitleStream( <b>int</b> playlistId, <b>int</b> streamAudio, <b>int</b> streamSubtitle, <b>boolean</b> on, <b>boolean</b> immediate) | selects the specified audio and subtitle track number (index) in a given playlist (movie) and in the current playlist it if immediate is true                        |
| <b>void</b> manager.selectAudioSubtitleStream(#string movieName, <b>int</b> streamAudio, <b>int</b> streamSubtitle, <b>boolean</b> on, <b>boolean</b> immediate)      | same as above                                                                                                                                                        |
| <b>void</b> manager.selectSubtitle( <b>int</b> streamNumber)                                                                                                          | selects the specified subtitle track number (index) in the current movie (0 means off)                                                                               |
| <b>void</b> manager.selectSubtitle( <b>int</b> streamNumber, <b>boolean</b> on)                                                                                       | selects the specified subtitle track number (index) in the current movie                                                                                             |
| <b>void</b> manager.selectSubtitleState( <b>boolean</b> on)                                                                                                           | allows to switch subtitles on or off                                                                                                                                 |
| <b>void</b> manager.selectSubtitleStream( <b>int</b> playlistId, <b>int</b> streamNumber, <b>boolean</b> immediate)                                                   | selects the specified subtitle track number (index) in a given playlist (movie) and in the current playlist it if immediate is true (0 means off)                    |
| <b>void</b> manager.selectSubtitleStream(#string movieName, <b>int</b> streamNumber, <b>boolean</b> immediate)                                                        | same as above                                                                                                                                                        |
| <b>void</b> manager.selectSubtitleStream( <b>int</b> playlistId, <b>int</b> streamNumber, <b>boolean</b> on, <b>boolean</b> immediate)                                | selects the specified subtitle track number (index) in a given playlist (movie) and in the current playlist it if immediate is true                                  |
| <b>void</b> manager.selectSubtitleStream(#string movieName, <b>int</b> streamNumber, <b>boolean</b> on, <b>boolean</b> immediate)                                     | same as above                                                                                                                                                        |
| <b>void</b> manager.selectTitle( <b>int</b> title)                                                                                                                    | selects specified title (unloads JAR, requires the JAR signing)                                                                                                      |
| <b>void</b> manager.selectTitle(#int jarIndex)                                                                                                                        | activates title corresponding to the specified JAR index (unloads JAR, requires the JAR signing)                                                                     |
| <b>void</b> manager.set2Dmode()                                                                                                                                       | sets 2D mode                                                                                                                                                         |
| <b>void</b> manager.set3Dmode()                                                                                                                                       | sets 3D mode                                                                                                                                                         |
| <b>void</b> manager.setDB( <b>float</b> mute)                                                                                                                         | sets the gain of the primary audio in decibels. Positive values amplify the audio signal and negative values attenuate the signal                                    |
| <b>void</b> manager.setLevel( <b>float</b> mute)                                                                                                                      | sets the gain of the primary audio using a floating point scale with values between 0.0 (silence) and 1.0 (the loudest level)                                        |
| <b>void</b> manager.setMute( <b>boolean</b> mute)                                                                                                                     | mutes or unmutes the primary audio                                                                                                                                   |
| <b>void</b> manager.setNeedMode3D( <b>int</b> mode3D, <b>boolean</b> InPrefetch)                                                                                      | sets the flag that causes the program to change the display mode on the next video start                                                                             |
| <b>float</b> manager.setPlaybackRate( <b>float</b> rate)                                                                                                              | sets new playback rate and returns the result (1 = play, 0 = pause, negative = rewind)                                                                               |
| <b>void</b> manager.setResumePlaylist( <b>int</b> playlistId)                                                                                                         | sets the flag that causes the program to resume the playlist on the next video start                                                                                 |
| <b>void</b> manager.setResumePlaylist(#string movieName)                                                                                                              | same as above                                                                                                                                                        |
| <b>void</b> manager.setStartPlay mark( <b>int</b> markId)                                                                                                             | sets the flag that causes the program to start the playlist from the specified play mark on the next video start                                                     |
| <b>void</b> manager.setStartPlay mark(#int markNumber)                                                                                                                | same as above                                                                                                                                                        |
| <b>void</b> manager.setStartPlay markResume()                                                                                                                         | sets the flag that causes the program to start the playlist from the resume time on the next video start                                                             |



| Function                                                             | purpose                                                                                                                                                                                                                   |
|----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>void</b> manager.setStartSegment(String strSegment)               | sets the segment, which will be activated (activateSegment) after the startVideo command                                                                                                                                  |
| <b>void</b> manager.startVideo(int playlistId, int markId)           | starts the specified playlist (movie) with making sure that jumpPlay mark is possible or not possible instead of real start (if possible then uses the jumpPlay mark instead of the real start)<br><br>See also playVideo |
| <b>void</b> manager.startVideo(#string movieName, #int markNumber)   | same as above                                                                                                                                                                                                             |
| <b>void</b> manager.startVideoAt(int playlistId, int markId)         | starts the specified playlist (movie) without any checks                                                                                                                                                                  |
| <b>void</b> manager.startVideoAt(#string movieName, #int markNumber) | same as above                                                                                                                                                                                                             |
| <b>void</b> manager.stopMovie()                                      | stops disc playback (so not just the movie!), this action requires JAR signing.                                                                                                                                           |

### Secondary (PIP) movies

|                                                            |                                                                                                                                     |
|------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <b>int</b> manager.getSecondaryAudio()                     | returns the actual PIP video track number (index)                                                                                   |
| <b>int</b> manager.getSecondaryScale()                     | returns the actual scale                                                                                                            |
| <b>int</b> manager.getSecondaryVideo()                     | returns the actual PIP audio track number (index)                                                                                   |
| <b>boolean</b> manager.isPIPAudioOff()                     | returns true if the PIP audio is off                                                                                                |
| <b>boolean</b> manager.isPIPAudioOn()                      | returns true if the PIP audio is on                                                                                                 |
| <b>boolean</b> manager.isPIPFSoFF()                        | returns true if the PIP is not in the full screen mode                                                                              |
| <b>boolean</b> manager.isPIPFSoN()                         | returns true if the PIP is in the full screen mode                                                                                  |
| <b>boolean</b> manager.isPIPOff()                          | returns true if the PIP is off                                                                                                      |
| <b>boolean</b> manager.isPIPOn()                           | returns true if the PIP is on                                                                                                       |
| <b>void</b> manager.selectSecondaryAudio(int streamNumber) | selects the PIP audio track                                                                                                         |
| <b>void</b> manager.selectSecondaryVideo(int streamNumber) | selects the PIP video track                                                                                                         |
| <b>void</b> manager.setPIP(boolean stateON)                | switch the PIP on or off                                                                                                            |
| <b>void</b> manager.setPIPDDB(float mute)                  | sets the gain of the secondary audio in decibels. Positive values amplify the audio signal and negative values attenuate the signal |
| <b>void</b> manager.setPIPFullScreen(boolean stateON)      | switch the PIP full screen on or off                                                                                                |
| <b>void</b> manager.setPIPLLevel(float mute)               | sets the gain of the secondary audio using a floating point scale with values between 0.0 (silence) and 1.0 (the loudest level)     |
| <b>void</b> manager.setPIPMute(boolean mute)               | mutes or unmutes the secondary audio                                                                                                |

### Bookmark functions

| Function                                                      | purpose                                                                             |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <b>void</b> manager.addBookmark()                             | adds a bookmark with the current time for the active movie                          |
| <b>void</b> manager.addBookmark(int playListID, long time)    | adds a bookmark with the specified time (in nanoseconds) for the specified playlist |
| <b>void</b> manager.addBookmark(#string movieName, long time) | same as above                                                                       |
| <b>Iterator</b> manager.bookmarkIterator(int playListID)      | returns Iterator to iterate through the bookmarks of the specified playlist         |
| <b>Iterator</b> manager.bookmarkIterator(#string movieName)   | same as above                                                                       |
| <b>int</b> manager.currentBookmarkNumber(int playListID)      | returns the current (active) bookmark number (index) for the specified playlist     |
| <b>int</b> manager.currentBookmarkNumber(#string movieName)   | same as above                                                                       |
| <b>int</b> manager.currentBookmarkNumber(#string movieName)   | same as above                                                                       |

| Function                                                                            | purpose                                                                                                       |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| <b>String</b> manager.currentBookmarkText( <b>int</b> playListID)                   | returns the current (active) bookmark time for the specified playlist in a text form (01:23:45)               |
| <b>String</b> manager.currentBookmarkText( <b>#string</b> movieName)                | same as above                                                                                                 |
| <b>long</b> manager.currentBookmarkTime( <b>int</b> playListID)                     | returns the current (active) bookmark time for the specified playlist in nanoseconds                          |
| <b>long</b> manager.currentBookmarkTime( <b>#string</b> movieName)                  | same as above                                                                                                 |
| <b>long</b> manager.currentBookmarkTime()                                           | same as above for current playlist                                                                            |
| <b>void</b> manager.deleteBookmark()                                                | deletes current bookmark for the current playlist                                                             |
| <b>void</b> manager.deleteBookmark( <b>int</b> playListID)                          | deletes current bookmark for the specified playlist                                                           |
| <b>void</b> manager.deleteBookmark( <b>#string</b> movieName)                       | same as above                                                                                                 |
| <b>void</b> manager.deleteBookmark( <b>int</b> playListID, <b>long</b> time)        | deletes a bookmark for the specified playlist with the specified time (in nanoseconds)                        |
| <b>void</b> manager.deleteBookmark( <b>#string</b> movieName, <b>long</b> time)     | same as above                                                                                                 |
| <b>void</b> manager.firstBookmark( <b>int</b> playListID)                           | selects the first bookmark for the specified playlist                                                         |
| <b>void</b> manager.firstBookmark( <b>#string</b> movieName)                        | same as above                                                                                                 |
| <b>void</b> manager.firstBookmark()                                                 | same as above but for current playlist                                                                        |
| <b>int</b> manager.getBookmarksCount( <b>int</b> playListID)                        | returns the number of bookmarks in the specified playlist                                                     |
| <b>int</b> manager.getBookmarksCount( <b>#string</b> movieName)                     | same as above                                                                                                 |
| <b>int</b> manager.getBookmarksCount()                                              | Same as above for current playlist                                                                            |
| <b>String</b> manager.getBookmarkText( <b>int</b> playListID, <b>int</b> number)    | returns the specified by the index bookmark time for the specified playlist in a text form (01:23:45)         |
| <b>String</b> manager.getBookmarkText( <b>#string</b> movieName, <b>int</b> number) | same as above                                                                                                 |
| <b>String</b> manager.getBookmarkText( <b>int</b> number)                           | same as above for current playlist                                                                            |
| <b>long</b> manager.getBookmarkTime( <b>int</b> playListID, <b>int</b> number)      | returns the specified by the index bookmark time for the specified playlist in nanoseconds                    |
| <b>long</b> manager.getBookmarkTime( <b>#string</b> movieName, <b>int</b> number)   | same as above                                                                                                 |
| <b>long</b> manager.getBookmarkTime( <b>int</b> number)                             | same as above for current playlist                                                                            |
| <b>boolean</b> manager.isFirstBookmark( <b>int</b> playlistId)                      | returns true if current bookmark of the specified movie is the first (returns true if there are no bookmarks) |
| <b>boolean</b> manager.isFirstBookmark( <b>#string</b> movieName)                   | same as above                                                                                                 |
| <b>boolean</b> manager.isFirstBookmark()                                            | same as above for current playlist                                                                            |
| <b>boolean</b> manager.isLastBookmark( <b>int</b> playlistId)                       | returns true if current bookmark of the specified movie is the last (returns true if there are no bookmarks)  |
| <b>boolean</b> manager.isLastBookmark( <b>#string</b> movieName)                    | same as above                                                                                                 |
| <b>boolean</b> manager.isLastBookmark()                                             | same as above for current playlist                                                                            |
| <b>void</b> manager.jumpBookmark( <b>int</b> playListID)                            | jumps to the current bookmark time in the specified playlist (movie plays from that point onwards)            |
| <b>void</b> manager.jumpBookmark( <b>#string</b> movieName)                         | same as above                                                                                                 |
| <b>void</b> manager.jumpBookmark()                                                  | same as above for current playlist                                                                            |
| <b>void</b> manager.lastBookmark( <b>int</b> playListID)                            | selects the last bookmark for the specified playlist                                                          |
| <b>void</b> manager.lastBookmark( <b>#string</b> movieName)                         | same as above                                                                                                 |
| <b>void</b> manager.lastBookmark()                                                  | same as above for current playlist                                                                            |
| <b>void</b> manager.nextBookmark( <b>int</b> playListID)                            | selects the next bookmark for the specified playlist                                                          |
| <b>void</b> manager.nextBookmark( <b>#string</b> movieName)                         | same as above                                                                                                 |
| <b>void</b> manager.nextBookmark()                                                  | same as above for current playlist                                                                            |
| <b>void</b> manager.prevBookmark( <b>int</b> playListID)                            | selects the previous bookmark for the specified playlist                                                      |
| <b>void</b> manager.prevBookmark( <b>#string</b> movieName)                         | same as above                                                                                                 |
| <b>void</b> manager.prevBookmark()                                                  | same as above for current playlist                                                                            |

| Function                                                                             | purpose                                                  |
|--------------------------------------------------------------------------------------|----------------------------------------------------------|
| <b>void</b> manager.removeAllBookmarks( <b>int</b> playListID)                       | deletes all bookmarks for the specified playlist         |
| <b>void</b> manager.removeAllBookmarks( <b>#string</b> movieName)                    | same as above                                            |
| <b>void</b> manager.removeAllBookmarks()                                             | same as above for current playlist                       |
| <b>void</b> manager.setCurrentBookmark( <b>int</b> playListID, <b>int</b> number)    | selects a bookmark with specified index, starting from 1 |
| <b>void</b> manager.setCurrentBookmark( <b>#string</b> movieName, <b>int</b> number) | same as above                                            |
| <b>void</b> manager.setCurrentBookmark( <b>int</b> number)                           | same as above, but using the current playlist            |
|                                                                                      |                                                          |

## Player, GPR or disc states

| Function                                                                            | purpose                                                                         |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| <b>void</b> manager.allowSaveState( <b>boolean</b> Allow)                           | tells the engine if it possible to store the state in the current menu or movie |
| <b>boolean</b> manager.canDiscResume()                                              | returns true, if the Disc resume is possible                                    |
| <b>boolean</b> manager.canResume( <b>int</b> playlistId)                            | returns true, if the Resume can be done for the specified playlist (movie)      |
| <b>boolean</b> manager.canResume( <b>#string</b> movieName)                         | same as above                                                                   |
| <b>void</b> manager.clearGPR()                                                      | clear GPRs from 1 to 4000 (set to 0)                                            |
| <b>boolean</b> manager.fileExists( <b>String</b> fileName)                          | checks if the specified file exists in the storage                              |
| <b>int</b> manager.getGPR( <b>int</b> gprNumber)                                    | returns the value for the specified GPR                                         |
| <b>int</b> manager.getPSR( <b>int</b> psrNumber)                                    | returns the value for the specified PSR                                         |
| <b>boolean</b> manager.hasDataInStorage()                                           | returns true if there are data for this disc in the storage                     |
| <b>boolean</b> manager.loadDataFromStorage()                                        | loads data from storage                                                         |
| <b>void</b> manager.logPSR()                                                        | logs all PSR values to the debug output                                         |
| <b>String</b> manager.readData( <b>String</b> key)                                  | reads data from properties (storage)                                            |
| <b>String</b> manager.readData( <b>String</b> key, <b>String</b> defValue)          | reads data from properties (storage) and returns defValue if no data found;     |
| <b>int</b> manager.readDataInt( <b>String</b> key)                                  | reads data from properties (storage)                                            |
| <b>int</b> manager.readDataInt( <b>String</b> key, <b>int</b> defValue)             | reads data from properties (storage) and returns defValue if no data found      |
| <b>long</b> manager.readDataLong( <b>String</b> key)                                | reads data from properties (storage)                                            |
| <b>long</b> manager.readDataLong( <b>String</b> key, <b>long</b> defValue)          | reads data from properties (storage) and returns defValue if no data found      |
| <b>float</b> manager.readDataFloat( <b>String</b> key)                              | reads data from properties (storage);                                           |
| <b>float</b> manager.readDataFloat( <b>String</b> key, <b>float</b> defValue)       | - reads data from properties (storage) and returns defValue if no data found    |
| <b>boolean</b> manager.readDataBoolean( <b>String</b> key)                          | reads data from properties (storage)                                            |
| <b>boolean</b> manager.readDataBoolean( <b>String</b> key, <b>boolean</b> defValue) | - reads data from properties (storage) and returns defValue if no data found    |
| <b>boolean</b> manager.restoreDataFromStorage()                                     | loads data from the storage and restores stream settings                        |
| <b>boolean</b> manager.resumeDisc()                                                 | resumes the state of the disc from the storage                                  |
| <b>void</b> manager.resumeVideoAt( <b>int</b> playlistId)                           | resume the specified playlist (movie)                                           |
| <b>void</b> manager.resumeVideoAt( <b>#string</b> movieName)                        | same as above                                                                   |
| <b>void</b> manager.saveStorage()                                                   | saves the disc state to the storage                                             |
| <b>void</b> manager.setGPR( <b>int</b> gprNumber, <b>int</b> gprValue)              | sets GPR value                                                                  |
| <b>void</b> manager.storeData( <b>String</b> key, <b>int</b> value)                 | stores the value in the properties (storage)                                    |
| <b>void</b> manager.storeData( <b>String</b> key, <b>String</b> value)              | stores the value in the properties (storage)                                    |
| <b>void</b> manager.storeData( <b>String</b> key, <b>long</b> value)                | stores the value in the properties (storage)                                    |
| <b>void</b> manager.storeData( <b>String</b> key, <b>float</b> value)               | stores the value in the properties (storage)                                    |
| <b>void</b> manager.storeData( <b>String</b> key, <b>boolean</b> value)             | stores the value in the properties (storage)                                    |
| <b>void</b> manager.Store_Streams()                                                 | calls selectAudioSubtitleStream                                                 |

## Appendix G: Dockable windows

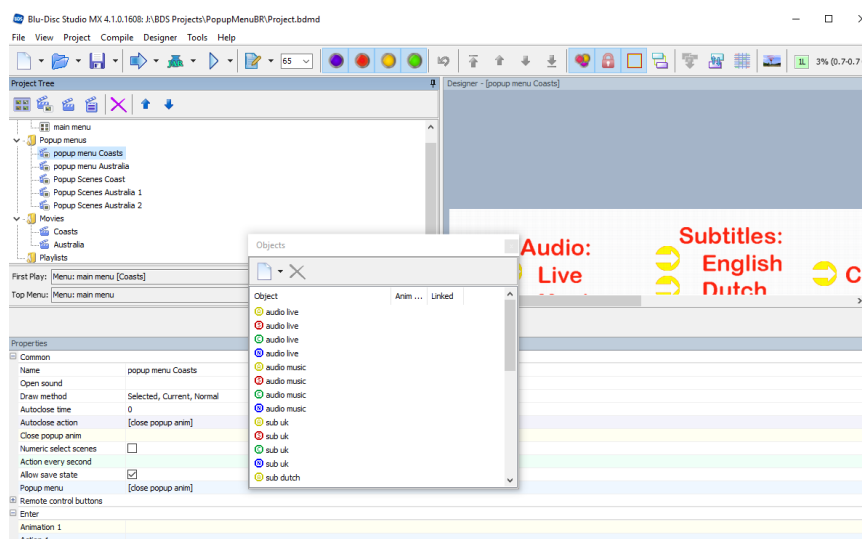
The current version of BDS has its user interface made as dockable windows. These windows are Project Tree, Objects, Properties, Designer, Action Matrix, Log. Each is a separate window that you can move around and place or “dock” at a specific location where you want it. You can make your own interface layout and determine the size of the windows yourself.

For those that are unfamiliar with the dockable windows, this appendix is a small guide to using them.

### Undocking windows

This is the simplest of operations. Simply point your mouse at the top bar of the window and drag it to some place on the screen. The window will detach from the BDS window and you can move and size it at will.

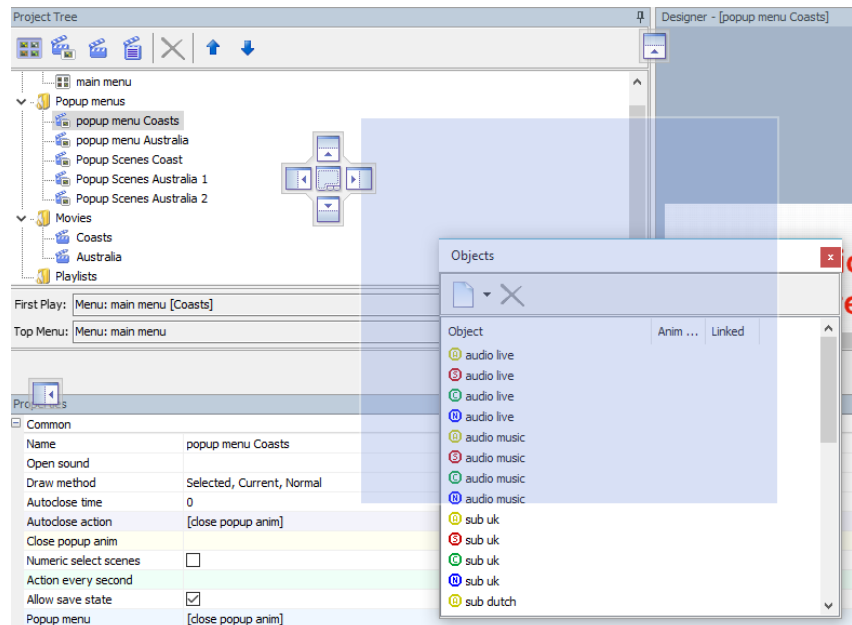
The figure below detaches the “objects” window. The windows still docked will stretch to fill the place now left over by the undocked window.



### Redocking windows

An undocked window can be re-docked at specific positions within other windows (or the basic BDS interface itself).

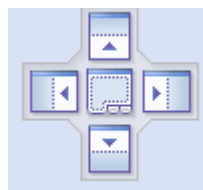
Moving the undocked Objects window upwards over the docked Project Tree window, a set of positioning areas in the form of a cross is made visible on the window over which you hover.



The blue rectangle shows where you are moving this Objects window (over the Project Tree). You can now dock the Objects window in 9 different ways. Five of them tell you how you can position it in relation to Project Tree, four of them indicate how you can position it in relation to the BDS window.

### In relation to a docked window

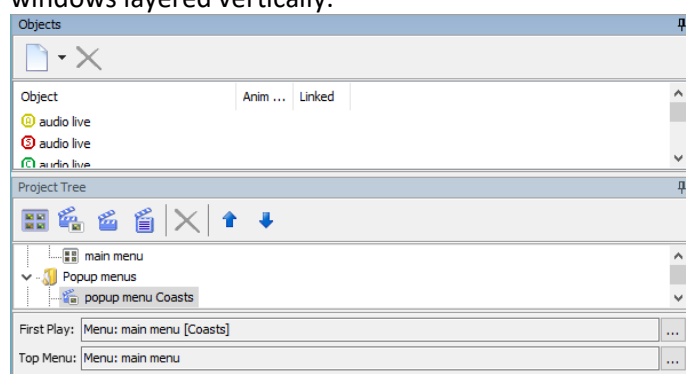
In the docked window area you see a cross of five positions that can be used for the to-be-docked window.



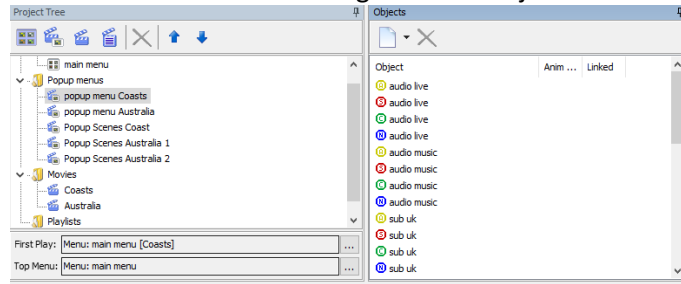
By dragging the undocked window on top of one of these buttons, docks the window at the position indicated.

Going around the clock starting at the top the positional markers mean:

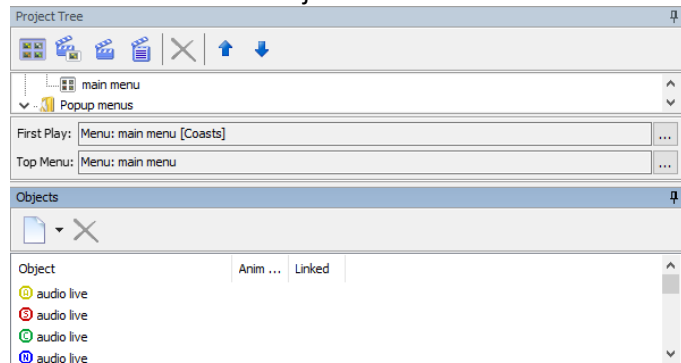
- Position it above the Project Tree window. The result is 2 windows layered vertically.



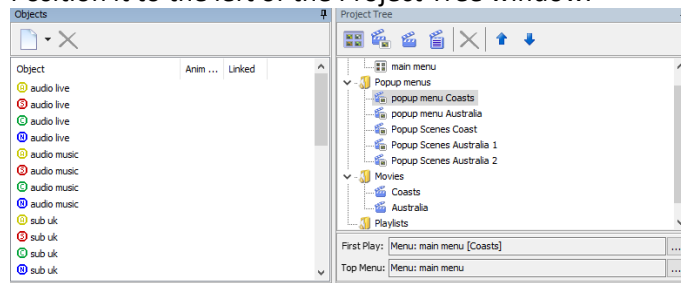
- Position the window to the right of the Project Tree window.



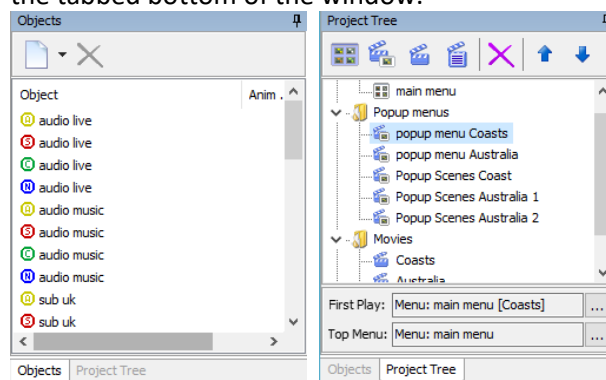
- Position it below the Project Tree window



- Position it to the left of the Project Tree window.

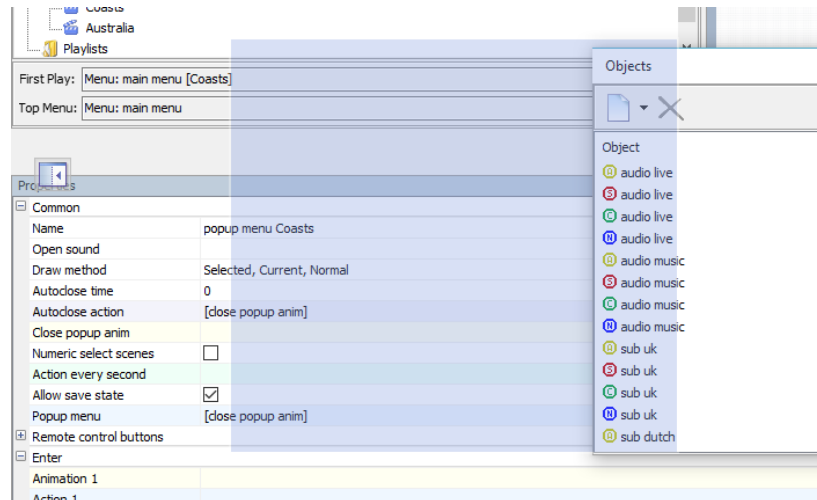


- The one in the middle is special: it is a tabbed icon and means that the undocked window will share the window space with the other window. Only one of the two is visible at any point in time, but they can be swapped by clicking on the tabbed bottom of the window.

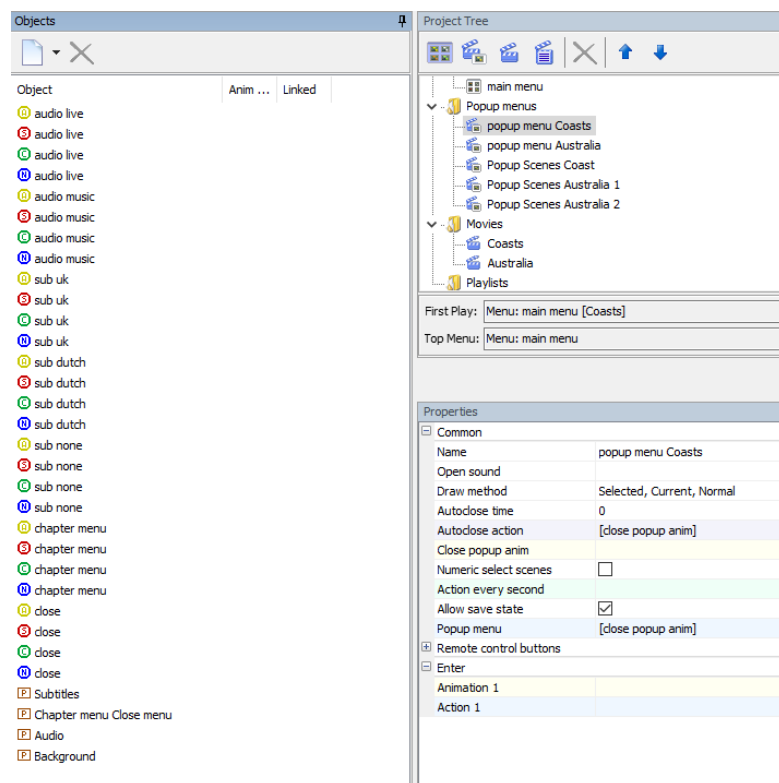


### In relation to the BDS window

There are four more positions that can be taken. They are indicated by position markers at the top, bottom, left and right edges of the BDS screen. The left marker is shown in the figure below.




Dragging the undocked Objects window to this left marker will dock the window alongside the entire left side.

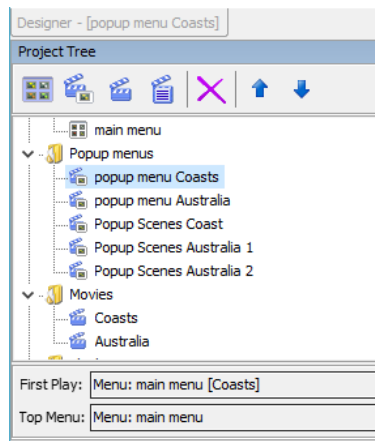



The position markers for top, right, bottom will produce similar results: the window is docked along the entire top/right/bottom edge.

## Unpinning a window

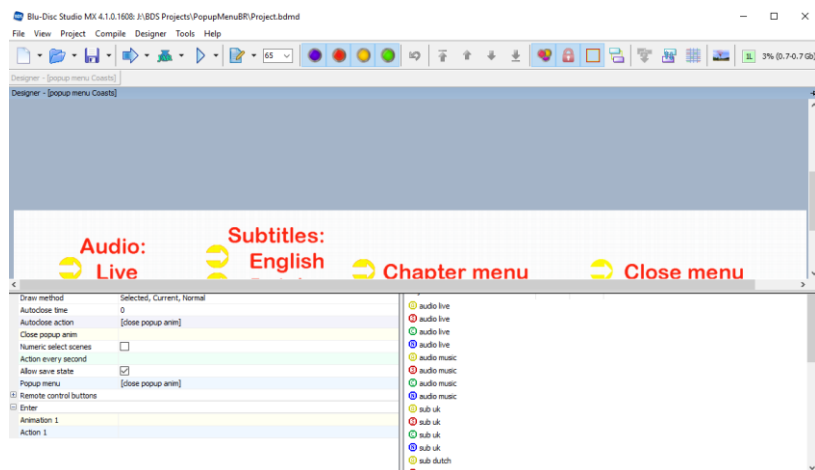
Each window, when docked, has a pin icon at the right top (  ).

When you click on this, the window becomes “unpinned” and reduces itself to a tab near the menu bar. The other windows below or next to it will resize to fill the space. When you do this to the Designer window that is next to the Project Tree window, the latter will expand, the former will become part of the menu bar.



Clicking on the “Designer” tab, opens the Designer window again, but now it is unpinned – as shown by the alternate pin icon (  ).

The window overlays any other window, hiding some of their content.



If you press the pin icon, the window becomes docked again and the partly covered windows resize and retake their original position.

## Reset to “default” division

If you got totally lost, everything can be restored to the initial “out of the box” position by clicking on the BDS menu bar: View > Reset Windows.



## Appendix H: Required disc space

### Movie space calculation calculation

There is nothing magic about the size needed for a movie to be stored on disc. This only depends on three properties:

- The resolution of the image. The larger the image, the more detailed the image. But more pixels mean more data per second.
- Framerate. The higher the frame rate, the smoother (less compressed) the movements of the movie. But each frame uses the resolution of a single image. A 50 fps progressive movie has twice as many frames as a 25p movie.
- Audio bitrate. The more bits used the more natural the sound will be. Uncompressed (LPCM - .wav) is more natural but much larger in size than compressed audio (Dolby - .ac3).  
Note that sampling frequency 44100 Hz is not supported.

These properties translate into a single physical property: movie file size. The larger the resolution and framerate, the larger the file size. And each bluray disc can only hold 25 GB or 50 GB.

At play time it translates into data rate which is expressed in units of kbps – kilobytes (1000 bytes) per second. The more bits can be used for a second of a movie's play time, the more detailed and smooth the movie image can be and the more natural its sound.

Whenever a project goes beyond the physical storage size, the data rate will have to be lowered making for smaller movie files. Audio and image will be of a lesser quality.

If there is room left and you want to squeeze in another movie, you need to make it fit in the existing space.

The calculation is simple (here the 8000 = 8 bits x 1000 kB/MB):

$$\text{size MB} = \frac{\text{running time (in minutes)} \cdot 60 \cdot \text{datarate (kbps)}}{8000}$$

Or

$$\text{datarate (kbps)} = \frac{\text{size MB} \cdot 8000}{\text{running time (in minutes)} \cdot 60}$$

where 1 byte = 8 bits and 1 Mega = 1000 k = 1000 000.

As an example: you've got 1068 MB of space left over and need to fit in a 30 minutes movie. Calculate its size:  $\frac{1068 \cdot 8000}{30 \cdot 60} = 4746$

## Appendix I: Checklist

### Things to remember

So, you've created your disc. Simulation or running it for real on a bluray player reveals some forgotten stuff. Back to the drawing board and start again.

This list may help you to check items beforehand:

#### **Movies**

- All movies are in BDAV (BDA compliant) format  
No frame rates 30 or 60 fps as they are no BDA approved standard and cause mpls not found errors during muxing. Frame rates 29.97 and 59.94 fps are valid. See Supported frame rates on page 472
- Audio on BR is 48 kHz sampled – for lossless LPCM as well as for lossy Dolby Digital. CDs and DVDs use 44.1 kHz. These audio tracks may need to be upsampled to 48 kHz.. See Supported primary and secondary audio streams on page 474
- Filenames have no non-printable characters
- All movie placeholders have filled in video, audio and subtitles streams (and .srt files have no non-printable characters)
- Audio and subtitles streams have the correct language settings and are in the same order in all movies and are fitting for the movie resolution. For suitable size of the subtitle depending on the resolution, see section Bluray disc set top player specification on page 20.
- If there are subtitles, all movie placeholders belong to the same movie group
- Subtitles have correct setting of checkbox for 23.97 fps
- Movie placeholders have End Action and Popup menu filled in (popup menu should turn back to the menu if no popup menu exists).
- Do not use serif fonts for subtitles. Use character outlines or drop shadows to make white letters readable on a white background.
- Subtitles are centered. No more than two lines each time.
- Sometimes a 20 seconds subtitle takes long to appear if you suddenly fast-forward to a scene with such a subtitle. Rather use consecutive 4 x 5 seconds.
- If you use PIP movies, the frame rate of the PIP movie must match that of the primary movie.

#### **Menus**

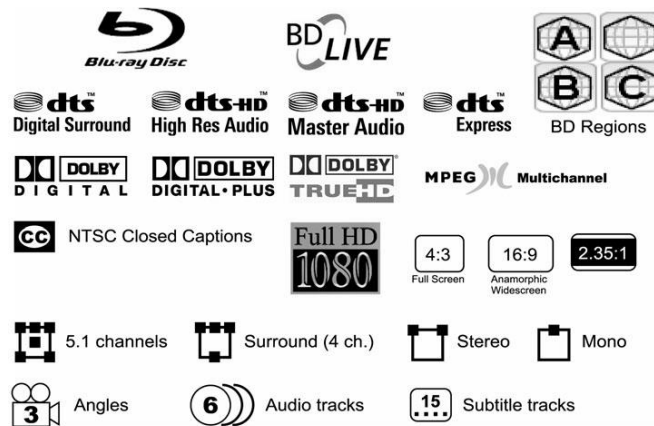
- All menus have a menu movie defined (BDS V4 and later will insert a black silent movie if absent). Optionally also an intro movie (that must be defined as movie placeholder)
- To avoid audio sync problems use the same frame rate for the menu movies as you do for the other movies.

- Menu movie and intro movie (if used) are the same movie if you want to seamlessly transfer from intro to menu movie
- Use the same menu movie for all menus if you want a seamless transfer between menus.
- FirstPlay and Top Menu have values and the Project > JAR settings allow the Top Menu to be used
- An intro movie should not have End Action or Popup menu defined nor should it be used as regular movie. It should have a second chapter mark that indicates when the menu objects are shown.
- Playlists also have End Action and Popup Menu defined the same way as for movies
- Playlists have either their own chapter marks or imported them from the individual movies (all chapters or just the movie starting points)
- For episode binge watching, the playlist chapters have code to skip closing and opening titles (and “previously in” opening parts) through “Open Scenes (Direct Editing)” on movie placeholder context menu
- Popup menu buttons need a MultiAction setting on the Press ENTER action of a button on the menu. First action is what to do (play movie chapter, show menu, whatever), second action is the fadeout of the popup menu (Jump Popup > [close popup anim])
- All buttons on a menu can be reached through navigation (and also departed from)
- All menus can be reached through navigation (and returned from)
- Disc runs on pc (software players) and set top players display correctly on HD television screens and/or video projectors.

## Additional items for commercial releases

Some less important topics (for home use, but important if authoring a commercial disc):

- Entry points in a movie (chapter marks) must start at a GOP header. An I-frame is not sufficient. GOP is a group of pictures. It is an elementary unit of MPEG video access. A GOP consists of I,P and B picture frames.
- Subtitle intervals should not include an entry point
- GPR values are reset to 0 if a disc stops or is ejected
- If the disc gets a package or its label is printed on the disc, ensure it specifies regions, audio types and other logos (search on internet for “bluray (region|audio|whatever) logos” to find transparent, vector or other types of logos). Or visit (at time of writing) [www.demolandia.net](http://www.demolandia.net).



- Appropriate restrictions are set on introductory movies (showing applied audio, FBI warnings and other stuff rippers usually remove a.s.a.p.)
- A thumbnail image is added in Metadata for use in PlayStation and some set top players
- Try to use the same framerate for all movie elements (especially menu movie). Switching between movies causes long (black) delays on playback as well as possible audio sync loss when frame rates are different. Preferred is progressive, 24 fps as universally acceptable with the least number of problems.
- Avoid burned-in subtitles. Use optional subtitles (even if switched on by default through project settings). These are sharper and give the viewer an option to not-use them.
- Have no subtitles running through on-screen titles (move them upwards to a safe area with no titles)

## Appendix J The Project file (bdmd file)

### Introduction

The entire project is stored in a number of files – most importantly the video, audio and menu image files. The configuration of the project is kept in the project file, stored in the project directory and of file type .bdmd.

Each time you open a project, the old version is saved in the hidden \\_History folder of the project. It is postfixed with the creation date of that project edition. This way you can always return to any earlier version of the project if so needed. (once you made the hidden folder visible in Explorer's "options").

The file itself is an XML structured file that contains all of the configuration you made to the project. It is this part that can be re-compiled as JAR file without having to remux the entire project.

Being XML structured, the file contains very recognizable tags that correspond with settings made for a movie (like EndAction) or button (like state image and action).

You should refrain from modifying the project file except in emergency cases and when you know extremely well what it is you're doing. It might get corrupted otherwise.

### Project file structure

The overall structure of the file is as follows.

```
<BDMD_Project Version="99">
 <Properties>
 <FirstPlay Type="type" />
 </Properties>
 <Fonts>
 dir1\dir2\fontname.ttf
 ...
 </Fonts>
 <Menus>
 <Menu Name="main" ... >
 <Text>
 menu texts
 </Text>
 ...
 </Menu>
 ...
 </Menus>
 <Popupmenus>
 ...
 </Popupmenus>
 <Movies>
 <Movie Name="moviename" ... ScnCount="nn" ... >
 <PIPVideo1 ... >
 <PIPVideo2 ... >
 <PIPVideo3 ... >
 <PIPVideo4 ... >
```

```

 <Chapters>
 <PTS0 realPTS="0" chk="true" />
 <PTS1 ... />
 ... as many as ScnCount upto PTSn-1 ...
 </Chapters>
 </Movie>
 ...
</Movies>

<Playlists>
 <Playlist Name="listname" ... >
 <Movies>
 <Movie0 Name="name" ... >
 </Movie0>
 ...
 </Movies>
 <Chapters>
 <PTSn ... />
 ...
 </Chapters>
 </Playlist>
 ...
</Playlists>

<Groups>
 <Group Name="name" DefAudio="n" ... />
</Groups>

</BDMD Project>

```

The subsections contain all possible settings to their maximum allowed number, even if you only provided one or two (e.g. there are 32 audio tags <Audio1> to <Audio32> as well as subtitle tags for each movie).

Replacing values in the project file should only be done with extreme caution and with an untampered copy available as backup. It can save you a lot of clicking in the BDS interface to change a value at multiple spots such as a different image for a button state when 50 buttons use the same image.

# Index





- .264. *See* movie
- .ac3. *See* audio
- .dts. *See* audio
- .eac3. *See* audio, Dolby Digital Plus
- .m2v. *See* movie
- .mvc. *See* movie
- .png. *See* menu, image
- .spi. *See* slide show
- .srt. *See* movie, subtitles
- .ssif. *See* 3D
- .sup. *See* movie, subtitles
- 3D
  - .ssif folder and files, 409
  - Creating MVC true 3D disc, 409
  - Creating SBS 3D disc, 419
  - Mode, 288
  - Movie. *See* movie
  - mvc files, 409
  - SPM (Stereo Photo Maker), 435
  - Subtitler, 427
- A-B loop, 347
- Action
  - Action Every Second, 354
  - Action Every Second ((popup) menu), 451
  - Autoclose (menu), 337
  - Chapter. *See* Action, Open Scenes
  - Conversion to Java, 311
  - End Action, 62, 68, 115, 119, 120, 130, 229, 299, 382, 391, 488, 526
  - Inactivity (popup menu), 256
  - Multi-action, 203, 244, 289
  - Navigation, 12, 39, 76, 100, 113, 114, 118, 449
  - Open scenes (direct editing), 305, 326
  - Popup, 488
  - Press Enter, 113, 114, 129, 141, 179
  - Remote control Left/Right/Up/Down. *See* Action Navigation
  - Remote control Play/Pause /Next /Prev /FastForward /Rewind. *See* Navigation
  - Start of chapter. *See* Action, Open Scenes
  - Switch, 203, 205, 230, 231, 244, 290, 292
- Action Matrix, 39, 116, 446, 447
- Aligning objects, 100
- Animation
  - Absolute position, 251
  - Animation group, 253
  - At disc boot, 214
  - At menu opening, 214
  - Close (popup menu), 251, 256
  - Displacement, 252
  - Pauses between animations, 266
  - Relative position, 252
  - Removal, 256
  - Set, 256
  - Slide, 256
  - submenu example, 254
  - Types, 255
- Animation ((popup)menu)
  - Multiple animations, 250
- Animation (button), 251
- Animation (menus), 9, 12, 249, 267
  - anim file, 278
  - Animation speed, 257
  - Animations sequentially, 265
  - chapters, 189
  - Clip, 259
  - Fade, 259
  - main menu and popup, 267
  - Movement calculations, 262
  - Multiple animations, 259
  - Transitions, 260
- Aspect ratio, 21, 58, 378, 407, 421, 438, 470, 490
- Audio
  - Bluray disc supported streams, 475
  - Dolby Digital, 51, 475
  - Dolby Digital Plus, 51, 384, 475
  - Dolby TrueHD, 475
  - DTS, 51, 475
  - DTS-HD, 384, 475
  - from CD, 25, 453, 475, 526
  - Language track, 43, 52, 60
  - Lip-sync problems, 34, 491, 492, 493
  - LPCM, 475
  - Reserved GPRs, 288
  - supported formats, 25
  - switch track, 244, 322
  - WAV mixer, 60
- Authoring bluray disc
  - Summary of steps, 194
- Auto-assignment button, 100, 154, 452
- AVC Encoder within BDS, 34
- AVC(HD). *See* video format
- BDAV. *See* video format
- BDID. *See* Programming (Java)
- BDID (playlist ident). *See* Programming (Java)
- BDMV elements, 25, 384, 473
- BDMV folder. *See* bluray disc

- BDS exposed functions. *See* programming (Java)
- Behind the scenes movie, 350
- B-frame, 481
- Binge watching, 296
- Bit rate, 479
  - Capacity versus bit rate, 482
- Blu Disc Studio
  - AVC Encoder, 34
  - Configuring workdesk, 28
  - Disc language specification, 42
  - Hardware requirements, 19
  - Installation, 16
  - Java BD-J profile, 44
  - Lite, 8
  - Menu requirements, 36
  - Migrating from Lite edition, 27
  - Muxers. *See* muxers
  - MX, 8
  - MX Pro, 9
  - Project specific setup, 39
  - Standard, 8
  - Subtitle requirements, 36
  - Version comparison, 9
  - Video requirements, 34
  - Workdesk interface, 38
  - Workdesk windows, 39
- Bluray disc
  - BD-J, 477
  - BDMV Elements, 482
  - BDMV folder, 65, 70, 72
  - Burn a disc, 71
  - Capacity, 478
  - Chapter mark, 165
  - Disc menu, 117, 225
  - HDMV, 477
  - Maximum number of BDMV elements, 25, 384, 473
  - PLAYLIST folder, 165
  - Regions, 469
  - Specifications (BDA), 469
  - Stream data. *See* Bit rate
  - STREAM folder, 65, 165
  - Supported aspect ratios, 470
  - Supported audio streams, 474
  - Supported frame rates, 472
  - Supported resolutions, 470
  - Supported video streams, 473
  - Top menu, 44, 117, 225, 465
  - Top Menu, 218
- Bluray Disc Association (BDA), 469
- Bluray player
  - Profile, 476
  - Video and audio standards, 475
- Bookmark, 234
  - Add, 235
  - Delete, 236
  - Delete all, 236
  - Play current bookmark, 237
  - Play from first bookmark, 236
  - Select, 235
- Button
  - Action. *See* action
  - Active state, 83, 134, 279
  - Aligning buttons, 100
  - Current state, 134, 142, 143
  - Navigation. *See* Action Navigation
  - Normal state, 79, 84, 96, 104
  - Remote control. *See* Action Navigation
  - Selected state, 84, 96, 104
  - Size, 94
- Carousel. *See* menu, *See* chapters
- CD
  - audio ripped from, 25, 453, 475, 485, 526
- Chapters. *See* Wizard, Chapter popup
  - Action when chapter reached. *See* Action
  - Automatic generation, 54
  - Carousel chapter menu, 250
  - Create popup chapter menus, 166
  - Creating playmarks, 63
  - Customizing popup menus, 180
  - data.ini file, 181
  - Direct Editing, 127, 199
  - Import/Export, 164
  - importing chapters from disc, 165
  - Invalid movie file names, 53
  - Manually set chapters, 54
  - Play mark, 19
  - Popup menu carousel, 174
  - Popup menu customization, 174
  - Popup menu horizontal or vertical, 173
  - Popup menu load settings, 171
  - Popup menu non-carousel, 174
  - Popup menu save settings, 171
  - Popup menu with images only, 176
  - Scan distance, 55
  - Standard, 127, 199
- Clone (all) between main/popup, 187
- Clone properties to all, 151, 155
- CMF (Cutting Master Format), 9
- Comma separated values (CSV), 164
- Commercial settings

- Disable features, 465
- Regional setting, 465
- Remote control buttons, 466
- Resume disc, 466
- Signing JAR file, 466
- UOP - User Operations Prohibited, 465
- Condition. *See* action, switch
- CSV. *See* comma separated values
- Demuxer, 21, 463, 500
- Designer Window, 107
  - viewport, 161, 270
- Direct editing. *See* Action, Open Scenes
- Disable features. *See* commercial settings
- Disc creation. *See* Bluray disc
- Dolby Atmos. *See* Audio
- Dolby Digital. *See* Audio
- Dolby Digital (ac3). *See* audio, Dolby Digital
- Dolby True HD. *See* Audio
- DTS. *See* audio, *See* Audio
- Encoder
  - AVC encoder in BDS, 34
- Errors and possible solutions, 483
- Event
  - Change (for subtitles), 68
- External tools, 496
- First Play. *See* On JAR Startup
- First Play menu. *See* On JAR startup
- First Play movie. *See* On JAR Startup
- Frame packed (Full 3D). *See* movie .mvc
- GPR (General Purpose Register). *See* programming
- H.264/AVC, 25, 31, 475, 477, 497
- H.264/MPEG4, 477
- HD (High Definition). *See* resolution
- HDMV, 477, *See also* Bluray disc HDMV
- HDMV Title, 208
- Help
  - Online, 14, 39, 459, 483
- I-frame, 481
- I-Frame, 34, 54, 493, 527
- Image. *See* Menu Image
- Index. *See* chapters
- JAR (Java ARchive). *See* programming
- JAR file
  - JAR title properties, 213
  - Multi-jar project, 207
- JAR property
  - Allow Top Menu call, 44
  - Animation, 214
- JAR Settings
  - First Play, 209, 211
- JAR title
  - Add, 212
  - Delete, 212
  - Modify, 213
- Java. *See* programming (Java)
- Java Developer Kit (JDK). *See* programming
- Java Virtual Machine (JVM). *See* programming
- Jittering picture. *See* movie
- Language track. *See* audio
- Lip-synch problems. *See* Audio
- Log file, 39, 65, 66, 67, 321, 488
- Log folder, 39
- LPCM. *See* Audio
- LPCM (WAV) audio. *See* Audio
- Making Of movie, 350
- Menu, 78
  - .png file, 90
  - Add button object, 94
  - Add button object – synchronize text, 95
  - Add Text button objects, 94
  - Add Text object, 92
  - Aligning objects, 100
  - Arranging objects in Designer Window, 107
  - Button. *See* button
  - Button navigation. *See* Action navigation
  - Carousel of buttons, 161, 267, 270
  - Carousel of menus, 221
  - Clone all between main/popup, 187
  - Clone between main/popup, 187
  - Creation from .png image files, 90
  - Creation through Objects Window, 90
  - Creation through Photoshop, 87
  - First Play menu, 207, 218
  - Image, 81
  - Intro menu, 78, 131
  - Linked objects, 424
  - Object copying, 110
  - Object effect, 449
  - Object ordering, 109
  - Object positioning, 112
  - Object renaming, 111
  - Object state, 449
  - Object, linked, 424
  - Open menu with action or animation, 261
  - Photoshop, 87
  - Popup, 115, 146, 156, 178, 184, 193, 221, 235, 256
  - Project specific setup, 49
  - Replace JAR file. *See* Programming, JAR
  - Seamless transfer between menus, 80, 260
  - Text fonts, 90, 91

- Timeline popup, 327
- Top Menu, 218
- Meta data generation failed, 36, 61, 486
- Miniso. *See* Virtual disc
- Movie
  - .264 file, 463, 477
  - .m2ts file, 59, 65, 165, 491, 497
  - .m2v file, 21, 477
  - .mpv file, 21, 385, 477
  - .mvc 3D format, 405, 409
  - 3D movie, 45, 406, 419
  - BDAV. *See* video format
  - Clone properties to all, 151, 155
  - colour and black and white in parallel, 323
  - different angles, 323
  - End Action, UMaskTable, 488
  - File name (acceptable), 33, 483
  - First Play movie, 131, 207
  - Frame packing (full 3D), 405
  - Intro movie, 129, 131, 322
  - Jittering picture, 74, 491
  - m2v files, 385
  - Menu movie, 78, 80, 129, 131, 322
  - Movie group, 131
  - Movie subtitles (SBS) 3D, 427
  - MPEG-TS. *See* video format
  - OU (Over Under) 3D, 419, 420
  - PIP video scaling, 387
  - Playlist. *See* Playlist
  - Popup menu, UMaskTable, 488
  - Properties, 42, 242, 399, 452
  - Properties - copy selected properties
    - between movies, 115
  - SBS (side by side) 3D, 406, 419, 422
  - Seamless jump between movies, 126
  - Stuttering picture, 74, 491
  - Subtitles (.srt and .sup), 36, 58
  - Subtitles ({\an8} ), 36
  - Subtitles (Change Event), 68
  - Subtitles (offset for MX or tsMuxer), 23
  - Subtitles (set language), 43, 53
  - Subtitles (settings for HD or SD), 23, 52
  - Time base, 74, 491, 493
  - title. *See* HDMV title, On JAR Startup
  - Video (re)scaling, 221, 376, 386, 512
  - Video track, 34, 58
  - Virtual, 129
- mpls file
  - chapter marks on ripped bluray disc, 165
  - not found error, 475
- Multi-jar project, 207
- Muxers
  - MX, 376, 381, 383
  - MX Muxer, 32
  - Output folder, 56, 65
- Normal state. *See* button
- Object. *See* Menu
  - linked, 424
- On JAR startup, 117, 119, 208, 209
- Opening menu (First Play). *See* menu
- OU (Over Under). *See* movie
- P-frame, 481
- Photoshop. *See* menu, creation
- Picture. *See* Menu Image
- Picture in Picture. *See* PIP
- PIP (Picture in Picture)
  - (De)Activate, 388
  - Audio requirements, 384
  - In mux, 386
  - Luma key, 387
  - Muxing (with MX muxer), 386
  - Out mux, 387
  - Positioning, 386
  - Video requirements, 384
  - Video scaling, 387
- Playlist, 125
  - Add chapter marks, 301
  - Episode watching, 298
  - PlaylistID, 322, 327
  - Select default audio/subtitle track, 128
- Positioning. *See* Menu, Object positioning
- Profile
  - BD-J Version 1, 81
  - BD-J Version 2, 81
- Programming (Java), 310, 504
  - basic constructs, 505
  - BDID (playlist ident), 351, 352, 355, 485, 491, 513
  - BDID preprocessor, 513
  - BDS exposed function specification, 511
  - BDS exposed functions, 509
  - GPR (General Purpose Register), 203, 206, 287, 288, 289, 290, 291, 293, 302, 304, 317, 351, 519, 527
  - JAR (Java Archive), 12, 30, 68, 93, 250
  - JDK (Java Development Kit), 478
  - JVM (Java Virtual Machine), 30
  - PSR (Player Status Register), 287, 292, 519
  - UDF (User Defined Function), 320
  - UDV (User Defined Variable), 318
- Project
  - Output folder, 56, 65

- Properties, 49
- Project (BDS), 38
  - bdmd file, 311, 504, 529
  - Merge projects, 77
  - Properties, 39
  - Signed, 31
  - Template, 28, 40
  - Tree view, 39
  - Wizard, 28
- PSR (Player Status Register). *See* programming
- Register. *See* Programming
- Resolution, 21, 58, 470, 490, 525
  - HD (High Definition), 490
  - SD (Standard Definition), 490
- Resume
  - disc playback, 45, 225, 227, 229
  - movie playback, 229
- SBS (side by side). *See* movie
- Scenarist, 8, 33, 128, 375, 456
- Scenes. *See* chapters
  - Direct Editing, 199, 201
  - Standard, 199, 201
- SD (Standard Definition). *See* resolution
- Seamless transfer. *See* menu
- Selected state. *See* button
- Simulation
  - Menu, 30, 118, 119
  - Remote control buttons, 120
- Slide show
  - .spi timing file, 376, 378
  - 3D images, 430, 441
  - MX, 31
  - Using video editor, 375
  - x264 encoder, 376, 381
- SPM (Stereo Photo Maker), 435
- STREAM folder. *See* bluray disc
- Stuttering picture. *See* movie
- Subtitles. *See* movie
- Switch. *See* action
- Synchronize text. *See* Menu-Add button
  - object
- Time base. *See* movie
- Timeline, 152, 312, 327
  - With bookmarks, 332
- Timer
  - Inactivity timer (menu and popup menu), 219
- Top menu. *See* bluray disc
- Top position. *See* menu, positioning
- UDF (User Defined Function). *See* programming
- UDV (User Defined Variable). *See* programming
- UMaskTable, 488
- Video format
  - AVCHD, 51
  - BDAV, 9, 34, 51, 74, 375, 475
  - MPEG4-AVC, 9
  - MPEG-TS, 51, 74, 475
- Video scaling. *See* movie
- Video track. *See* movie
- Viewport, 161, 270
- Virtual disc
  - iso, 70
  - miniso, 70, 413, 417, 503
- WAV. *See* Audio
- Wizard
  - chapter bitmaps, 167
  - Chapter popup menu via template, 176
  - Chapter popup menu via text and rectangles, 169
  - movie chapter popup menu, 166
  - New project, 57
  - Timeline, 152, 153