20. Rendering files

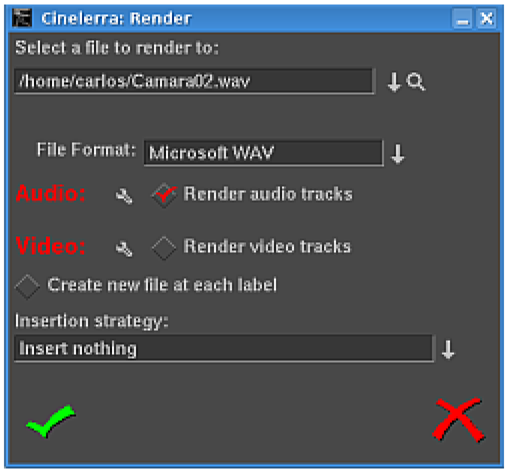
Rendering takes a section of the timeline, performs all the editing, effects and compositing, and stores it in a pure movie file. You can then delete all the source assets, play the rendered file in a movie player, or bring it back into Cinelerra for more editing. It is very difficult to retouch any editing decisions in the pure movie file, however, so keep the original assets and XML file around several days after you render it.

All rendering operations are based on a region of the timeline to be rendered. You need to define this region on the timeline. The navigation section describes methods of defining regions. See section Timebar. The rendering functions define the region based on a set of rules. When a region is highlighted or in/out points are set, the affected region is rendered. When no region is highlighted, everything after the insertion point is rendered. Merely by positioning the insertion point at the beginning of a track and unsetting all in/out points, the entire track is rendered.

20.1 Single file rendering

The fastest way to get media to disk is to use the single file rendering function.

Go to **File->render** or press SHIFT-R to bring up the render dialog. Select the magnifying glass 🔍 to bring up a file selection dialog. This determines the filename to write the rendered file to and the encoding parameters.


Cinelerra: Render
Select a file to render to:
/home/carlos/Camara02.wav
File Format: Microsoft WAV
Audio:
Render audio tracks
Video:
Render video tracks
Create new file at each label
Insertion strategy:
Insert nothing

The render window

In the render dialog select a format from the **File Format** menu. The format of the file determines whether you can render audio or video or both. Select the **Render audio tracks** toggle to generate audio tracks and **Render video tracks** to generate video tracks.

Select the wrench next to each toggle to set compression parameters. If the file format can not store audio or video the compression parameters will be blank. If **Render audio tracks** or **Render video tracks** is selected and the file format does not support it, trying to render will pop up an error.

20.2 Separate files rendering

The **Create new file at each label** option causes a new file to be created when every label in the timeline is encountered. This is useful for dividing long audio recordings into individual tracks. When using the renderfarm, **Create new file at each label** causes one renderfarm job to be created at every label instead of using the internal load balancing algorithm to space jobs.

When **Create new file at each label** is selected, a new filename is created for every output file. If the filename given in the render dialog has a 2 digit number in it, the 2 digit number is overwritten with a different incremental number for every output file. If no 2 digit number is given, Cinelerra automatically concatenates a number to the end of the given filename for every output file.

In the filename `` `/hmov/track01.wav' `` the `` `01' `` would be overwritten for every output file. The filename `` `/hmov/track.wav' ``; however, would become `` `/hmov/track.wav001' `` and so on and so forth. Filename regeneration is only used when either renderfarm mode is active or creating new files for every label is active.

20.3 Insertion strategy of rendered files

Finally the render dialog lets you select an insertion mode. The insertion modes are the same as with loading files. In this case if you select **insert nothing** the file will be written out to disk without changing the current project. For other insertion strategies be sure to prepare the timeline to have the output inserted at the right position before the rendering operation is finished. See section Editing. Editing describes how to cause output to be inserted at the right position.

It should be noted that even if you only have audio or only have video rendered, a **paste** insertion strategy will behave like a normal paste operation, erasing any selected region of the timeline and pasting just the data that was rendered. If you render only audio and have some video tracks armed, the video tracks will get truncated while the audio output is pasted into the audio tracks.

20.4 Batch rendering

Batch Rendering is one of Cinelerra's great but lesser-known strengths. It allows you to eliminate manual repetitive keystrokes and mouse clicks, and automate the rendering of audio-video files. It even allows for Cinelerra to be completely driven by external programs, with no need for the user to manually interact with the Cinelerra user interface.

If you want to render many projects to media files without having to repeatedly attend to the **Render** dialog, **batch rendering** is the function to use. In this function, you specify one or more Cinelerra project XML files (EDL) to render and the unique output files for each. Then Cinelerra loads each project file and renders it automatically, without any user intervention. Each Cinelerra project XML file, combined with the settings for rendering an output file, are called a **batch**. This allows a huge amount of media to be processed and greatly increases the value of an expensive computer.

The first thing to do when preparing to do batch rendering is to create one or more Cinelerra projects (EDL) to be rendered and save them as normal Cinelerra project (`` `myproject.cin.xml' ``) files. The batch renderer requires a separate Cinelerra project file for every batch to be rendered. You can use the same Cinelerra project file if you are rendering to different output files, for example, creating the same output video in different file formats.

To create a Cinelerra project file which can be used in batch render, set up a Cinelerra project and define the region to be rendered either by highlighting it, setting in/out points around it, or positioning the insertion point before it. Then save the project in the normal way to a `` `myproject.cin.xml' `` file (EDL). Define as many projects as needed this way. The batch renderer takes the active region from the EDL file for rendering.

With all the Cinelerra project files (EDL) prepared with active regions, go to **File->batch render**. This brings up the batch rendering dialog. The interface for batch rendering is a bit more complex than for single file rendering.

A list of batches must be defined before starting a batch rendering operation. The table of batches appears on the bottom of the batch render dialog and is called **batches to render**. Above this are the configuration parameters for a single batch.
A batch is simply a pairing of a Cinelerra project file with a choice of output file and render settings.

Set the **output path**, **file format**, **Audio**, **Video**, and **Create new file at each label** parameters as if you were rendering a single file. These parameters apply to only one batch. In addition to the standard rendering parameters, you must select the Cinelerra project file (`` `myproject.cin.xml' ``) to be used in the batch. Do this by setting the **EDL path**. Use the magnifier to bring a drop down menu with your files or write manually the path to your regular Cinelerra project file (`` `myproject.cin.xml' ``). In this case, **EDL path** has nothing to do with EDL files as created by **File/Export EDL**.
Cinelerra in batch render mode will not overwrite an existing output file. The batch render will simply fail. Make sure that no files with the same name as the output files exist before starting the render.

If the **batches to render** list is empty or nothing is highlighted, click **New** to create a new batch. The new batch will contain all the parameters you just set. Repeatedly press the **New** button to create more batches with the same parameters. Highlight any batch and edit the configuration on the top of the batch render window. The highlighted batch is always synchronized to the information displayed.
Click and drag batches to change the order in which they are rendered. Hit **delete** to permanently remove the highlighted batch.
In the list box is a column which enables or disables the batch. This way batches can be skipped without being deleted. Click on the **Enabled** column in the list box to enable or disable a batch. If it is checked, the batch is rendered. If it is blank, the batch is skipped.

The other columns in the batch list are informative.

- **Output** The output path of the batch.
- **EDL** The source EDL of the batch.
- **Elapsed** The amount of time taken to render the batch if it is finished.

To start rendering from the first enabled batch, hit **Start**.
Once rendering, the main window shows the progress of the batch. Once the batch finishes, the elapsed column in the batch list is updated and the next batch is rendered until all the enabled batches are finished. The currently rendering batch is always highlighted red.
To stop rendering before the batches are finished without closing the batch render dialog, hit **Stop**.
To stop rendering before the batches are finished and close the batch render dialog, hit **Cancel**.
To exit the batch render dialog whether or not anything is being rendered, hit **Cancel**.

You can automate Cinelerra batch renders from other programs. In the Cinelerra batch render dialog, once you have created your list of batch render jobs, you can click the button **Save List** and choose a file to save your **batch render list** to. We suggest you use a filename like `` `myrenderlist.batchrender.cin.xml' ``. Once you have created this file, you can start up a batch render without needing to interact with the Cinelerra user interface.
From a shell prompt (or from a script, or other program), execute:

```
cinelerra -r myrenderlist.batchrender.cin.xml
```

(changing `` `myrenderlist.batchrender.cin.xml' `` to whatever filename you chose for saving your batch render list).
When invoked with these parameters, Cinelerra will start up and perform the rendering jobs in that list, without creating its usual windows.

Programmers, please note: this is a powerful feature indeed. It means that if you can create valid Cinelerra project xml files and Cinelerra render list files from other programs (which requires just a small amount of skill with your favourite XML library), then you can gain full automated access to all of Cinelerra's functionality without needing to interact with the Cinelerra user interface. The possibilities for this are endless. You can leverage the power of Cinelerra and incorporate it into your own programs. It's a good idea if you can create simple Cinelerra project files and batch render files and study the XML format. By trial and error, you'll be able to generate valid Cinelerra xml files for projects and batch render lists, and thus create your own Cinelerra automation library in your favourite programming language.

20.5 The render farm

When bicubic interpolation and HDTV was first done on Cinelerra, the time needed to produce the simplest output became unbearable even on the fastest dual 1.7 GHz Xeon of the time. Renderfarm support even in the simplest form brings HDTV times back in line with SD while making SD faster than real-time.

While the renderfarm interface is not spectacular, it is simple enough to use inside an editing suite with less than a dozen nodes without going through the same amount of hassle you would with a several hundred node farm. Renderfarm is invoked transparently for all file->render operations when it is enabled in the preferences.

Cinelerra divides the selected region of the timeline into a certain number of jobs which are then dispatched to the different nodes depending on the load balance. The nodes process the jobs and write their output to individual files on the filesystem. The output files are not concatenated. It is important for all the nodes to have access to the same filesystem on the same mount point for assets.

If a node can not access an input asset it will display error messages to its console but probably not die. If it can not access an output asset it will cause the rendering to abort.

It should be noted that in the render dialog, the **Create new file at each label** option causes a new renderfarm job to be created at each label instead of by the load balancer. If this option is selected when no labels exist, only one job will be created.

A Cinelerra renderfarm is organized into a master node and any number of slave nodes. The master node is the computer which is running the GUI. The slave nodes are anywhere else on the network and are run from the command line. Run a slave node from the command line with `cinelerra -d`

That is the simplest configuration. Type `cinelerra -h` to see more options. The default port number may be overridden by passing a port number after the `` `-d' ``.

Most of the time you will want to bring in the rendered output and fine tune the timing on the timeline. Also some file formats like MPEG can not be direct copied. Because of this, the jobs are left in individual files.

You can load these by creating a new track and specifying **concatenate to existing tracks** in the load dialog. Files which support direct copy can be concatenated into a single file by rendering to the same file format with renderfarm disabled. Also to get direct copy, the track dimensions, output dimensions, and asset dimensions must be equal.

MPEG files or files which do not support direct copy have to be concatenated with a command line utility. MPEG files can be concatenated with **cat**.

Configuration of the renderfarm is described in the configuration chapter See section Renderfarm. The slave nodes traditionally read and write data to a common filesystem over a network, thus they do not need hard drives.

Ideally all the nodes on the renderfarm have similar CPU performance. Cinelerra load balances on a first come first serve basis. If the last segment is dispatched to the slowest node, all the fastest nodes may end up waiting for the slowest node to finish while they themselves could have rendered it faster.

20.6 Command line rendering

The command line rendering facility consists of a way to load the current set of batch rendering jobs and process them without a GUI. This is useful if you are planning on crashing X repeatedly or want to do rendering on the other side of a low bandwidth network. You might have access to a supercomputer in India but still be stuck in America, exiled you might say. A command line interface is ideal for this.

To perform rendering from the command line, first run Cinelerra in graphical mode. Go to **file->batch render**. Create the batches you intend to render in the batch window and close the window. This saves the batches in a file. Set up the desired renderfarm attributes in **settings->preferences** and exit Cinelerra. These settings are used the next time command line rendering is used.

On the command line run: `cinelerra -r`

to processes the current batch jobs without a GUI. Setting up all the parameters for this operation is hard. That is why the command line aborts if any output files already exist.

Other parameters exist for specifying alternative files for the preferences and the batches. Attempting to use anything but the defaults is very involved so it has not been tested.

20.7 Rendering videos for the internet

If you want to encode a video in order to put it on the internet, we recommend to render it as a Quicktime4linux file, and then encode that file in MPEG4 of FLV formats. The Quicktime4linux file rendered from Cinelerra must have the following properties:

- Audio option Two Complements 16bits (PCM)
- Video option DV

20.7.1 Encoding a video in MPEG4 format for the internet

To get the best quality, you should encode your Quicktime4linux file with mencoder in two passes.

First pass:

```
mencoder input.mov -ovc xvid -xvidencopts bitrate=600:pass=1 \
-vf scale=320:240 -oac mp3lame -lameopts abr:br=64 -o output.avi
```

Second pass:

```
mencoder input.mov -ovc xvid -xvidencopts bitrate=600:pass=2 \
-vf scale=320:240 -oac mp3lame -lameopts abr:br=64 -o output.avi
```

Do not forget to change the output size of the video, set with the `` `-vf scale=' `` option.

Here are some other command lines. They give output video files whose weight is around 13 Mb for 3 minutes.
First pass:

```
mencoder -oac pcm -sws 2 -vf scale=${width}:${height},hqdn3d=2:1:2 \
-ovc lavc -lavcopts vcodec=mpeg4:vbitrate=${video_bitrate}:vlelim=-4:\
vcelim=7:lumi_mask=0.05:dark_mask=0.01:scplx_mask=0.3:naq:v4mv:mbd=2:\
trell:cmp=3:subcmp=3:mbcmp=3:aspect=4/3:sc_threshold=1000000000:\
vmax_b_frames=2:vb_strategy=1:dia=3:predia=3:cbp:mv0:preme=2:\
last_pred=3:vpass=1:cgop -ofps 25 -of avi movie.mov -o /dev/null\
-ffourcc DIVX
```

Second pass:

```
mencoder -srate 32000 -oac mp3lame -lameopts cbr:br=${audio_bitrate}:\
aq=0 -sws 2 -vf scale=${width}:${height},hqdn3d=2:1:2 -ovc lavc\
-lavcopts vcodec=mpeg4:vbitrate=${video_bitrate}:vlelim=-4:vcelim=7:\
lumi_mask=0.05:dark_mask=0.01:scplx_mask=0.3:naq:v4mv:mbd=2:trell:\
cmp=3:subcmp=3:mbcmp=3:aspect=4/3:sc_threshold=1000000000:\
vmax_b_frames=2:dia=3:predia=3:cbp:mv0:preme=2:last_pred=3:vpass=3:\
cgop -ofps 25 -of avi movie.mov -o movie.avi -ffourcc DIVX
```

You probably have to adapt those command lines if your material is noisy; have a look at mencoder's pre-processing filters. The *_mask parameters are really important when encoding at low bitrate.

If you want your video file to be displayed properly on a well know media player which runs on Windows you should be aware that:

- the aspect ratio information contained in the AVI header will not be taken into account by that player. That is why you must scale the image to the right aspect ratio. Width and height must be multiples of 16. Those are the recommended resolutions for 4/3 PAL material: 384:288, 448:336, 512:384 or 704:528.
- the media player running on Windows will loose A/V sync if a VBR audio bitrate is used instead of CBR.
- the `` `-ffourcc' `` parameter is needed for the video codec to be recognized as Divx.

20.7.2 Encoding a video in FLV format for the internet

FLV files (FLash Video) weight is very small and the only thing needed to play those files is an internet browser with flash plugin version 7 or higher installed. That format is really useful when one wants to share a video with a wide audience over the internet.

First pass:

```
ffmpeg -i movie.mov -b 430k -s 320x240 -aspect 4:3 -pass 1 -ar 22050 movie.flv
```

Second pass:

```
ffmpeg -i movie.mov -b 430k -s 320x240 -aspect 4:3 -pass 2 -ar 22050 movie.flv
```

Pay attention to the output file extension. Ffmpeg uses it to determine the output format. The audio sampling frequency to use is 22050 and the `` `-ar' `` parameter must be used for the video to be properly encoded. Ffmpeg does not write metadata information in the flv file. The duration has to be written in the metadata information in order for some flash players to display a progress bar. FLVTool2 (http://www.inlet-media.de/flvtool2) can be used to insert that information:

```
cat input_file.flv | flvtool2 -U stdin output_file.flv
```

There are a number of options for embedding the flv file in a web page. You can use ming or flv2swf to create an swf file. http://klaus.geekserver.net/flash/streaming.html has detailed instructions for ming and http://search.cpan.org/~clotho/FLV-Info-0.17/bin/flv2swf can be installed with cpan> install FLV::ToSWF. Or you can use the Creative Commons Non-Commercial licenced JW FLV Player http://www.jeroenwijering.com/?item=JW_FLV_Player, or the Apache Licenced FlowPlayer http://flowplayer.org. Both of these allow you to use the flv as created above, and have controls for stopping and playing the movie etc.

20.8 Quicktime for GNU/Linux compatibility chart

Scott Frase wrote a Quicktime for GNU/Linux compatibility chart. It contains an exhaustive list of all the Quicktime compression schemes available and their compatibility in Cinelerra, Mplayer and some other media players. That document has two main sections, one based on an HDV resolution-formatted project and another based on a DV resolution-format project.

It is available here:
http://content.serveftp.net/video/qtcompatibility.ods

Some interesting notes:

- Mplayer does behave better with smaller, DV resolution video
- Cinelerra compatibility with files rendered from a DV project is not much different than its compatibility with files rendered from an HDV project.
- Comparison chart of DV/HDV mplayer/cinelerra compatibility included

20.9 Making a DVD

20.9.1 Rendering to mpeg2

Here is a method to export mpeg2 video for DVD. This method allows you to precisely set the encoding option you want and produces an mpeg2 file which is 100% compatible with all DVD standalone players. For how to make a DVD from the output See section Authoring a DVD.

Audio and video are rendered seperately and combined later in a procedure external to Cinelerra.
Audio is rendered into **.ac3**, and **video** is rendered into a **yuv4mpeg stream** which is piped through either **mpeg2enc** or **ffmpeg** into a **.m2v** file. Both

variants are described in detail below.

(Apparently depending on footage and player engine, one or the other variant may produce better results. Check out which one works best for you by rendering a short test edit of a few seconds length, authoring to DVD according to the sections below, and playing it in your cheapest standalone player to really see wether it is foolproof or displays errors.)

In both cases, make sure you properly defined your Cinelerra project format before rendering your video (menu **Settings->Format...**), preferably even before loading any raw footage.
TV standards:
PAL is 720x576 at 25 frames per second,
NTSC is 720x480 at 29.97 frames per second.

20.9.1.1 yuv4mpeg pipe through mpeg2enc

The mplex program from **mjpegtools** must be installed. The mjpegtools package is built in the hvirtual distribution and the mplex utility may be extracted from there.

1. Create a script `` `~/cine_render.sh' ``
2. Copy in `` `~/cine_render.sh file' `` the following lines:
   ```
   #/bin/bash
   mpeg2enc -v 0 -K tmpgenc -r 16 -4 1 -2 1 -D 10 -E 10 -g 15 -G 15 -q 6 -b 8600 -f 8 -o $1
   ```
3. Put the execute permissions on that file: `chmod 777 ~/cine_render.sh`
4. Within Cinelerra, and select the part of the project you want to render with the [and] points
5. Press `SHIFT-R`
6. Select the **YUV4MPEG Stream** file format
7. Deselect **Render audio tracks** and select **Render video tracks**
8. Click on the wrench
9. In the newly opened window, indicate the name of the `` `m2v' `` file you want to create. That file will contain video only.
10. Click on **Use pipe** and write this command: `/home/<your_user>/cine_render.sh %`
11. Click OK to close the second window, and OK again to render your `` `m2v' `` file
12. When the m2v file is rendered, open the rendering window again, and render an AC3 file at 224kbits
13. Finally, combine video and audio with this command:
    ```
    mplex -f 8 your_video_file.m2v your_audio_file.ac3 -o video_audio_file.mpeg
    ```
    If you obtain errors while using mplex, increase the quantizer (`` `-q' `` option, see below).

You can modify the mpeg2enc parameters if you want to. Look at the mpeg2enc manpage. Some details about the settings:

> `` `-b 8600' `` : This is the maximum bitrate of your `` `m2v' `` file (it does not include the audio bitrate). We recommend you to do not increase that value or you could get errors when mplexing the video and the audio.
> `` `-q 6' `` : This is the quantizer setting. If you reduce it (do not go below 3), the quality increases. But the bitrate will increase. It's recommended to keep the medium bitrate achieved (that's displayed when mplexing the audio and video files) around 10% lower than the bitrate defined with the `` `-b' `` setting.
> `` `-K tmpgenc' `` : invokes the TMPGEnc matrices. It reduces the average bitrate by about 10% compared to the default tables. For very-high quality material, you can try removing this option.

If your material is noisy (Hi8 analog material for example), you can add some mjpegtools in the command line written in `` `~/cine_render.sh' ``:

- `y4mshift` and `y4mscaler` can be used to remove the noisy borders around the video. For example, those commands added at the beginning of the command line in `` `cine_render.sh' `` remove the black borders around a Hi8 video:
  ```
  yuvscaler -v 0 -I ACTIVE_700x560+8+8 | y4mshift -n 2 |
  ```
- `yuvdenoise` and `yuvmedianfilter` can help removing noise. Example:
  ```
  yuvdenoise -F | yuvmedianfilter -T 3 |
  ```
  Denoising is a complex task, and the options given above are just an example. Please read the mjpegtools'manual and subscribe to its mailing-list for more information.

20.9.1.2 yuv4mpeg pipe through ffmpeg

1. Select **File->Render...** or press `SHIFT-R`. The render dialog pops up.
2. In the render dialog, you have the choice to render 1. the entire project, or 2. the highlighted selection, or 3. from In-point "[" to Out-point "]".
3. Make sure the **Insertion strategy** is "Create new resources only".
4. Select the **AC3** audio output file format.
5. Specify the audio output file name and path (example: `your-movie.ac3`).
6. Select **Render audio tracks** and deselect **Render video tracks**.
7. Click on the wrench next to "Audio:". A new dialog "Cinelerra: Audio Compression" pops up.
8. Set the bitrate to **128 kbps** (or leave it there).
9. Click OK, the compression dialog disappears.
10. In the render dialog, click OK, the dialogu disappears. Audio is rendered.

Rendering audio is much faster than rendering video but might still take some seconds. Watch the progress bar in the main window's lower right corner.

11. Again, press `SHIFT-R`. The render dialog pops up again.
12. Select the **YUV4MPEG Stream** file format.
13. Specify the video output file name and path (example: `your-movie.m2v`).
14. Deselect **Render audio tracks** and select **Render video tracks**.
15. Click on the wrench next to "Video:". A new dialog window "Cinelerra: YUV4MPEG stream" pops up. The first textbox should already contain the output filename and path you had specified in the render dialog.
16. Select "Use Pipe:".
17. Fill the following command line into the second textbox:
    ```
    ffmpeg -f yuv4mpegpipe -i - -y -target dvd -flags +ilme+ildct %
    ```
18. Click OK in the yuv4mpeg dialog and in the render dialog to render video output.
19. The resulting .m2v can be further processed together with the .ac3 audio with the following shell command, producing a dvd-compatible mpeg stream:
    ```
    ffmpeg -i your-movie.ac3 -i your-movie.m2v -target dvd -flags +ilme+ildct your-movie.mpg
    ```
    (Yes, the stream is sent through ffmpeg a second time.)

Note on ffmpeg command line options:
`-i` tells ffmpeg to read from standard input (in our pipe, this means from Cinelerra's render stream).
The `-y` option allows to overwrite existing target files (of course, it is safer to omit this, but then you must make sure to rename or delete previous results each time you want to render a new version).
The `+ilme+ildct` flags are for proper interlacing, bottom fields first, tested with PAL footage. Some Cinelerra versions suggest a similar command line in the ffmpeg pipe presets for DVD, however with erroneous syntax of the interlacing flags or without the flags.

Before proceeding to put your rendered mpeg2 data on DVD, you might want to watch and check your-movie.mpg in mplayer or xine/kaffeine.

20.9.2 Making a DVD menu

A DVD menu is composed of:

- a background (still image or video)
- buttons
- sound/music

You can build a menu with a GUI such as qdvdauthor, dvdstyler, dvdwizard or tovid. However, using those GUI is not perfect for the moment, since they are bugged or limited for the moment.

If you prefer to use a GUI, we recommend you to try tovid:
http://tovid.wikia.com/wiki/Main_Page
QDVDAuthor contained a lot of bugs sometime ago, but its author fixed some of them recently, which makes QDVDAuthor more usable.

The method we explain below is more complicated than using a GUI, however it:

- produces DVD playable on all standalone players
- is not subject to bugs
- will save you a lot of time since all you have to do to author a new DVD is to modify text files

Here are the steps needed to create your DVD menu:

- create the menu background with cinelerra
- add the buttons by creating PNG images
- combine the menu and the buttons with spumux

We suppose that you want to create a menu with an animated background. Launch Cinelerra and create a project containing what you want to be the background of the menu. You can add a music if you wish to. Pay attention to the fact that this menu will play in loop.

To draw the buttons, you have two possibilities:

- display them in Cinelerra. That way, you will be able to make animated buttons, such as a video thumbnail for each part of your video.
- do not draw the buttons in Cinelerra. You will add them later on, from PNG images "added" to the mpeg2 menu file. This is the simpliest method, but you won't be able to display animated buttons.

Render that video into m2v and ac3 using the `cine_render.sh` method explain above. Combine the audio and video with mplex as you would do with any "normal" video.

You obtain a mpeg2 file containing the menu background, and some buttons displayed above it if you added them in Cinelerra.

We have to use spumux to define each button position in that mpeg2 file. If you did not draw the buttons in Cinelerra, you will be able to put them in with spumux.

Spumux is a command line utility which takes 2 arguments:

- an XML file explaining where the buttons are
- the mpeg2 file name (the one you rendered for the menu)

Here is a spumux example XML file:

```
<subpictures>
 <stream>
  <spu start="00:00:00.0" image="buttons_normal.png" highlight=
  "buttons_highlight.png" select="buttons_select.png">
   <button name="1" x0="94 " y0="234 " x1="253 " y1="278"
   down="2" right="4" />
   <button name="2" x0="63 " y0="287 " x1="379 " y1="331" up="1"
   down="3" right="5" />
  </spu>
 </stream>
</subpictures>
```

- **image="buttons_normal.png"** This png image contains the buttons as they should appear when they are not selected nor highlighted.
- **highlight="buttons_highlight.png"** This png image contains the buttons in their highlighted state.
- **select="buttons_select.png** This png image contains the buttons in their selected state.

If you already made the buttons in Cinelerra, you have to specify empty (100% transparent) PNG images here.

The PNG images used in spumux must:

- contain an **alpha channel** (ie support transparency)
- be in **4 indexed colors**. You can easily convert an image to 4 indexed colors using Gimp.

There is one line per button. Each line contains the button coordinates, a button having a rectangular shape:

- **x0, y0**: upper left corner
- **x1, y1**: bottom right corner

You also have to set which button to move to when using the up, down, left and right buttons of the DVD remote. Here is an example:

```
<button name="3" ...coordinates... up="1" down="5" left="2" right="4" />
```

When button 3 is selected, if the "Up" key is pressed on the remote then the button 1 will be highlighted. If the "Right" key is pressed on the remote, then button 4 will be highlighted.

When you have finished editing your spumux XML file, you have to type this command:

```
spumux menu.xml < menu.mpeg > menu_with_buttons.mpeg
```

That will make a `` `menu_with_buttons.mpeg' ``. It is an mpeg2 files with buttons.

20.9.3 Authoring a DVD

After having rendered to mpeg2 your video files, and having prepared a menu with spumux, you need to "author" the DVD with dvdauthor, that is another command line application.

dvdauthor uses XML files to describe the DVD structure. You need to create an XML file in a text editor and save it as `` `simple_example.xml' `` in the same folder of your `` `/the/mpeg/file.mpeg' `` mpeg2 video file. You should really pay a lot of attention to the .xml file syntax since it is very rigorous. The risk is the DVD to be readable on some standalone players, but not on all ofthem.

To help you start using dvdauthor, here are some example XML files you can copy and paste into your `` `simple_example.xml' `` file. Replace example filenames and paths with the ones right for your project.

```
<dvdauthor dest="/path/to/the/folder/which/will/contain/the/dvd">
    <vmgm />
    <titleset>
        <titles>
            <pgc>
                <vob file="/the/mpeg/file.mpeg" />
                        <post>
                    jump chapter 1;
                </post>
            </pgc>
        </titles>
    </titleset>
</dvdauthor>
```

This is a very simple dvdauthor XML file. There are no menus, so the video file `` `/the/mpeg/file.mpeg' `` will be played as soon as you insert the DVD in the player.

The command in <post> tag means the video should be played in a loop. When the DVD player reaches the end of the video, it will jump to the first chapter of the video (which dvdautor assumes to be the beginning of the video since chapters haven't been defined). To make the video play only once without jumping from the end to the beginning, remove the following lines from your XML file.

```
                <post>
                    jump chapter 1;
                </post>
```

To author the DVD, go to the folder that contains the video and the XML file and type the following command:

```
dvdauthor -x simple_example.xml
```

Now, let's have a look at a more complex example. When the DVD is inserted, a menu is displayed and you can choose to play any of 4 videos.

```
<dvdauthor dest="/path/to/the/folder/which/will/contain/the/dvd" jumppad="yes" >
<vmgm>
 <fpc> jump menu 1; </fpc>
  <menus>
   <video format="pal" aspect="4:3" resolution="720x576" />
   <pgc entry="title" >
    <vob file="menu.mpeg" pause="0" />
    <button name="1" > { g3=1; jump titleset 1 menu entry root; } </button>
    <button name="2" > { g3=2; jump titleset 1 menu entry root; } </button>
    <button name="3" > { g3=3; jump titleset 1 menu entry root; } </button>
    <button name="4" > { g3=4; jump titleset 1 menu entry root; } </button>
     <post> { jump cell 1; } </post>
   </pgc>
  </menus>
 </vmgm>
 <titleset>
  <menus>
   <pgc entry="root" >
    <pre> { if ( g3 gt 0 )  {
                if ( g3 eq 1 ) { g3=0; jump title 1  chapter 1; }
                if ( g3 eq 2 ) { g3=0; jump title 1  chapter 3; }
                if ( g3 eq 3 ) { g3=0; jump title 1  chapter 5; }
                if ( g3 eq 4 ) { g3=0; jump title 1  chapter 7; }
                jump vmgm menu entry title;
                }
        } </pre>
    <post> { jump vmgm menu entry title; } </post>
   </pgc>
  </menus>
  <titles>
   <video format="pal" aspect="4:3" resolution="720x576" />
   <pgc pause="0" >
    <vob file="video_1.mpeg" pause="0" />
    <vob file="blackvideo.mpg" pause="0" />
    <vob file="video_2.mpeg" pause="0" />
    <vob file="blackvideo.mpg" pause="0" />
    <vob file="video_3.mpeg" pause="0" />
    <vob file="blackvideo.mpg" pause="0" />
    <vob file="video_4.mpeg" pause="0" />
    <post> { call vmgm menu entry title; } </post>
   </pgc>
  </titles>
 </titleset>
```

```
</dvdauthor>
```

The file `blackvideo.mpg' is used to add a 2 second black screen between each video. Here is how to create it:

```
convert -size 720x576 xc:black -depth 8 blackframe.ppm
dd if=/dev/zero bs=4 count=960000 | toolame -b 128 -s 48 /dev/stdin emptyaudio.mpa
ppmtoy4m -S 420mpeg2 -n 50 -F 25:1 -r blackframe.ppm | mpeg2enc -a 2 -n p -f 8 -o blackvideo.mpv
mplex -f 8 -o blackvideo.mpg blackvideo.mpv emptyaudio.mpa
```

20.9.4 Burning a DVD

When you have finished authoring the DVD, you will find in the destination folder the following directories: `AUDIO_TS' and `VIDEO_TS'. To test your DVD before burning it, cd into this folder, and type:

```
xine dvd:`pwd`
```

If your DVD plays fine on your computer, it is time to burn it. When you are in the folder containing `AUDIO_TS' and `VIDEO_TS', type this command (adjusting for your dvd burner device, eg /dev/dvdrw):

```
nice -n -20 growisofs -dvd-compat -speed=2 -Z /dev/dvd -dvd-video -V VIDEO ./ && eject /dev/dvd
```

If you have a lot of copies to do, you can first make an .iso master in the parent folder using this command:

```
nice -n -20 mkisofs -dvd-video -V VIDEO -o ../dvd.iso .
```

This `../dvd.iso' file can be burnt using this command:

```
nice -n -20 growisofs -dvd-compat -speed=2 -Z /dev/dvd=../dvd.iso && eject /dev/cdrom
```

We recommend you do not burn at a speed higher than 4x. Use good quality DVD-R only.

To test your DVD on a standalone player without wasting several DVD-R, you can burn on DVD-RW. First, format your DVD-RW using this command:

```
dvd+rw-format -lead-out /dev/dvd
```

Then, burn the DVD-RW using the commands above.

20.10 Using background rendering

Background rendering allows impossibly slow effects to play back in real-time shortly after the effect is pasted in the timeline. It continuously renders temporary output. When renderfarm is enabled, background rendering uses the renderfarm continuously. This way, any size video can be seen in real-time merely by creating a fast enough network with enough nodes.

Background rendering is enabled in settings->preferences->performance. It has one interactive function: **settings->set background render**. This sets the point where background rendering begins to where the in point is. If any video exists, a red bar appears in the timeline showing what has been background rendered.

It is often useful to insert an effect or a transition and then select settings->set background render right before the effect to preview it at full framerates.

[<<] [>>] [Top] [Contents] [Index] [?]

This document was generated by *root* on *March, 8 2008* using *texi2html 1.76*.