The making of "Escena.org DVD #1"

We finally managed to release **the first ever escena.org DVD**. It had been the subject of speculations on repeated occasions during the past years and yet it was never done, until now :-)

I'm going to detail how we produced the DVD for two reasons:

1. Most of the tasks were done with Linux. And although building a DVD with Linux is *relatively* easy, the information is also *slightly* hidden and confusing
2. We would like to see more people involved in producing DVDs and doing stuff in general, specially now that the technology is there and very importantly, is free

The research

First thing I did was to turn to the immediate tools. Since my main computers are macs, running Mac Os, I thought that maybe the famous iDVD, included in iLife, would be all that we needed for creating a couple of clickable still pictures and a series of movies. I couldn't be more wrong.

In the aim for being easy to use, it's too limiting. It intently pushes you to all things Apple. Music? From your iTunes database. Pictures? From your iPhoto. And so on and on. It even tries to **brand your dvd** with an Apple logo on a corner! Yes, you can deactivate it once you find the odd configuration setting, but do you think that's *normal*? More annoyances include the inability to edit the provided themes, unless you're willing to find and edit some property lists hidden somewhere in your ~/Library, or maybe /Library. I am unable to tell exactly where, since no matter how many times I tried to create a new theme, it was never recognized.

A direct consequence of not being able to edit the themes was that you couldn't change the layout and add, say, three buttons where there was only one. Or push them two pixel up, or down. Apart from the fact that one could easily say “that has been done with an Apple product” from quite far away, even if you managed to remove the Apple logo.

I investigated a bit more and found that the themes were created with Apple's *Pro* application for DVD authoring, *DVD Studio Pro*. So I thought: “OK, probably this iDVD thing is done for people who simply want to torture their friends and family with pictures from their last holidays, I'll see

what can we do with Apple's Pro stuff".

Enter DVD Studio Pro. Being a professional product, I would expect it to be the swiss army knife of video production… **but it couldn't even load videos which weren't in MOV format**. I don't want to mess with installing codecs from third parties and etc. Anyway, I installed the missing codecs and tried again. It crashed continuously. When it didn't crash, it produced odd error messages and when not, it said it was doing stuff and kept saying that for hours. That is, until I forced it to quit. I just couldn't believe it was so disastrous.

Once I finished with all the immediate possibilities, I decided to go for the exotic ones. I knew there had been some advances in the multimedia field for linux — anyone heard about ubuntu studio, or any of the other multimedia-focused distros? — but I hadn't had a real chance to check them by myself.

I tried several open source DVD authoring programs. The first one, Q DVD-Author was announced as the definitive full solution for DVD authoring in Linux, and I wouldn't need to mess with scripts or any command line thing. I thought that it would be cool… if only it worked!

I then tried with DVD Styler, DVD flick, devede… all were promising but none did completely fulfil what we wanted: they weren't flexible enough or were buggy (like not detecting properly the aspect ratio of the videos and stretching them in a bad way). So I finally investigated a bit more about the technology underneath all these programs and found out that all of them basically are a front end for these external programs:

- ffmpeg
- dvdauthor
- spumux
- mplex
- pngtopnm
- ppmtoy4m
- mpeg2enc
- mkisofs

I found a good tutorial on DVD authoring on Linux, which pretty much answered most of the questions I had about the process of converting a series of MPEG files into a DVD, but I still had to find out how to capture the productions in a suitable way and how to convert them to MPEG files. Keep reading…

The workflow

kkapture

Once we agreed on which demos and intros should be featured in the DVD, it was time to capture them. As expected, I used ryg's kkapture. For getting maximum quality, I used the uncompressed option, which created big (2Gb) fragments of raw AVI with the video, plus a WAV file with the audio. **The ideal resolution was 1024×768** for demos without antialias (so they got *instant antialias* when resized to the DVD dimensions later on). Less detailed demos were fine at 800×600.

Some intros and demos didn't let themselves to be captured, or when they did, the audio was out of

sync in a very uncontrollable manner. Those had to be left out. There were also little quirks with some demos because they do not synch to the music timer (which is what kkapture intercepts, if I understand properly) but to another timer, although they were still passable. (Finding out which ones are slightly wrong is left as an exercise to the viewer!)

Converting to a handier format

Once I the video and audio were captured, it was time to join all the video segments together plus the audio track, using AVIDemux, in order to convert them into a compressed, handier format that I could transfer from the windows box where the captures were done to the linux machine where the DVD authoring process was going to happen. I would have loved to use the raw sources, but that meant **a lot of gigabytes** per capture.

I first tried to do that using VirtualDub but for some reason it simply choked and refused to load the videos, or refused to export them in the format I wanted, or something else. (I found about AVIDemux in wurstcaptures, an interesting resource about capturing all types of demos).

The captures were compressed to MP4, using h264 for the video (700k, two passes), and the audio was encoded to MP3 using Lame, at the highest quality possible.

Converting did take a bit of time, during which I prepared a script for doing the whole resizing and encoding process automatically.

The script

I didn't want to resize each video manually, specially since each one needed slightly different parameters for the padding. Also some of them were slightly different internally (having the audio stream first, then the video stream, or the opposite). All that called for a script!

So I would place the input files in a folder (*assets/videos*), specify the files I wanted to convert in a list in the script, and let it do its magic, which in pseudocode looks like this:

1. for each video in the playlist:
    1. if it's not converted to mpeg:
        1. find out the video properties (dimensions and position of the audio and video streams)
        2. calculate the aspect ratio
        3. calculate the final height (using the aspect ratio); make sure it is divisible by two
        4. calculate the amount of vertical padding (above and below the video), if needed
        5. extract the video stream onto a .m2v file (mpeg2 video), while resizing it to the final dimensions and constraining it to dvd frame rate and aspect ratio (25 fps and 4:3 because we were going for a PAL DVD)
        6. extract the audio stream onto a .m2a file, constraining it to dvd requirements (stereo, MP2)
        7. multiplex both files, m2v and m2a

Why would you need to do this? It seems that DVD players aren't very powerful per se. They just want to be fed a stream which contains packets of video, audio, video, audio, and repeat until the end. This is what is called interleaved data, and that is why we separate the original video in two

streams, convert them to the suitable dvd format, and then multiplex them (i.e. mix them together into the *easily digestible*, interleaved format that DVD players like).

In code, it was something such as this for each video:

```
# extract the video stream
ffmpeg -i "$src_file" -y -vcodec mpeg2video -s $size_param $padding_param -r 25 -bufsize
1835008 -packetsize 2048 -muxrate 10080000 -aspect 4:3 -minrate 8000k -maxrate 8000k -b
8000k -threads 4 -strict very -f mpeg2video -map $video_stream_number:0 "$tmp_video"

# extract the audio stream
ffmpeg -i "$src_file" -threads 4 -acodec mp2 -ab 256k -ac 2 "$tmp_audio";

# multiplex both together
mplex -f 8 -o "$multiplexed_filename" "$tmp_video" "$tmp_audio"
```

The menus

DVD jargon is slightly confusing for my taste. They speak about menus, titles, titlesets, entry menus, root menus and much more. Confusing, very confusing, specially since it seems that no one really uses entry menus and root, angle or chapter menus at all as they are meant to be used. Or maybe that's because I have hardly used a real DVD player and only watch them in a computer.

Anyway, I finally found out two things:

1. the structure we wanted consisted in a main menu (what they call the “vmgm menu”) which pointed to four sections (i.e four “title sets”): “Productions”, “Demoscene”, “Extras” and “Credits”. Furthermore, “Productions” had three submenus (what they call several menus) for selecting which demo or intro to play, and the demos or intros (the titles), while “Demoscene”, “Extras” and “Credits” simply had a submenu and no titles.
2. A DVD menu is simply another MPEG 2 file; how you generate it is up to you. Clickable areas are defined with a simple process that is described below

If you have ever inspected a DVD with a file explorer, you probably remember about the

VIDEO_TS and *AUDIO_TS* folders, and the .vob files inside them. That is exactly what **dvdauthor** generates, once you feed it the already converted to MPEG 2 videos that you want your DVD to include and the structure, defined in **an XML file**.

The actions for the buttons are specified in this XML. The are some special tags (*post tags*) which are used for specifying what you want to happen once the playback ends. We wanted the menus to loop so that the background is always moving, and the demos to loop, and when the last demo finishes, it goes back to the first one, non stop, so that you can use the DVD to leave it in loop for example in a festival or something (hint! hint!). There's more information about this in the tutorial about DVD authoring I mentioned above, with more variations and examples, and of course, in the manual for dvdauthor.

But we still had to generate the menus and specify which areas were clickable somehow. Not having more time for researching, the videos for the menus were assembled with After Effects and then exported as an image sequence (a series of PNG files) which we converted into the desired mpg format by joining them into a m2v file and them multiplexing it with a silence.mp2 (because everything in a DVD needs to have audio, even if it's a silence), with **mplex** again.

Silence.mp2 was generated using ffmpeg:

```
ffmpeg -ar 48000 -t 10 -f s16le -acodec pcm_s16le -i /dev/zero -ab 64K -f mp2 -acodec mp2 -y "$silence_file"
```

The oddest part of all this is when you find that the clickable areas are defined and mixed into the menu MPEG 2 file using the same tool which is used for adding subtitles to the interleaved .mpg file. This tool is called **spumux** and again, takes an XML file as input. This XML defines the coordinates of the clickable areas (i.e. the buttons), and also the images which are used for each state of the button.

You need a separate XML file for each menu you want to have, but it doesn't use a separate image for each button in the menu. Instead, what you do is to create a big image with DVD dimensions (720×576) which contains all the buttons of the menu, and the DVD player will draw only the areas which are described as a button, ignoring the rest. It's like a mask.

That way you can use different images for creating highlight and rollover effects. But if you inspect in detail the menus of a commercial DVD, you'll realise that the edges of the highlights are very jagged too. This is a limitation of DVD's, because the subtitles can't have more than 4 colours. So wave bye bye to antialias!

In our case since we simply use horizontal underlines, it can't be noticed, but I got lots of strange errors until I realised that I wasn't saving the different state images as **an 8 bit PNG**. Even if it has only 4 or less colours, **you can't use a 24 bit PNG**.

Note how we are using an empty image for "image" and "select", and only use an image for highlighting, so only when you put your mouse over a clickable area, it shows some kind of feedback.

Also, you can't have non-rectangular buttons. If you see any dvd with non rectangular buttons, it's just a mask trick.

This step was also added to the script, so that if it didn't find the mpg menus, it would try to

rebuild them from the *assets/menu* source files. And I had to delete the mpg files lots of times, to force them to be regenerated, until I found the right settings for them.

I had to find a work-around to another limitation: you can't have a menu without a title. Since the “Demoscene”, “Extras” and “Credits” sections consisted simply in a menu with informative text and a button to go back to the main menu, I had to build an empty title for those menus. I just created an empty PNG file, but **24 bit** this time, and multiplexed it with the same silence.mp2 that was created before, and used that empty.mpg as the source for the empty titles.

Create the DVD contents

Once all the movies are converted and the menus generated, it was time to call **dvdauthor** to obtain the famous AUDIO_TS and VIDEO_TS folders. This was quite slow and the worst of all was that it didn't check for errors first and then proceed, but instead tried to proceed and check for errors during its process. Quite often there were mistakes at the end of the dvdauthor xml but it didn't detect them until it had processed all the previous titles, so away went several minutes.

If all went well, we had a nice **./output** folder which contained AUDIO_TS and VIDEO_TS.

Extras

Distributing captured demos is cool, but it is cooler if you also include the executable of the demo in the DVD-Rom area of the DVD. So we had an “extras” folder into which we placed the data (demos, tools, etc) that we wanted to include in the DVD structure.

Build the ISO

Before building the ISO file, we need to copy the **extras** folder, with all its nice and juicy stuff, to the **./output** folder, so that it will appear in the root of the DVD file system when browsing it with a computer.

The ISO itself is generated with a program called **mkisofs** (*make iso file system*, I guess). This is part of a suite of tools called **cdrtools**, and very funnily, the version which comes with Ubuntu and several other distributions is broken because someone decided to make a fork of it and introduced some bugs, apart from making the author get quite angry, I would say. But do not worry!

Nothing that can't be fixed by downloading the source code and compiling it ourselves. Even more, I was really pleased to see how easy it was to download the whole set of tools, compile them and simply use the tool I wanted (mkisofs), from within the folder I wanted, without having to install them into /usr/local or anything like that. **Very neat**.

Once I got a new and functioning mkisofs, I was able to generate an ISO image which would work in Mac, Windows and Linux, whereas with the version which came with ubuntu I couldn't get it to work in Mac and Windows - the DVD appeared to be empty because the filesystem was *invisible* to them.

Try it yourself!

Here is a tar.gz which contains all the files required for building this DVD. Excepting the video captures, and the sources of the menus, which were pretty huge (being 500 frames each, what did you expect?). Only one frame is included in these cases.

The file structure is pretty immediate if you look at the script for building the DVD - which is done in php so that it could be modified by lots more of people than if it had been done in bash and because well, I don't like bash scripting too much ;)

So the script is invoked as:

php create.php

You will need to have the required tools installed, which I mostly did with a combination of apt-gets and enabling several repositories. In case of doubt you can always paste the error you got into the magic search genius that will find everything you ask him (google), since most probably someone else had the same problem than you before.

Download the DVD

And if you just want to download the DVD, it's fine too! Get this torrent and enjoy! (You can also print the cover, designed by the mighty trace of xplsv)