November days in Gjøvik

Hanne Fagerjord Karlsen

01.12.2009

Contents

1 Abstract

The project "November Days in Gjøvik" is a project with focus on the creation of a timelapse movie as Gjøvik's 150th birthday approaches. The creation of the movies creation was automated with a shell script consisting of simple bash code and CLI commands from various open source software. It was important to keep the entire process as open source as possible, from software used to music featured in the final movie. The process was segmented into the gathering of data and programming/testing and there were few major issues besides adding metadata to a video file. The final script produces a film with quality depending on the images/audio submitted by the user. The user gets a couple of different prompts where (s)he can customize the movie to some extent. It is absolutely possible to automate this process and in the future develop the script into a more elegant tool for quick and easy creation of timelapse movies.

2 Introduction

The project has its roots in the course IMT4951, Applied Digital Workflow, which builds upon the course IMT4891:Digital Workflow Fundamentals. It was expected of the students to put the knowledge from Digital Workflow Fundamentals into real world use, this being knowledge regarding automation, metadata, recontextualisation and use of open source/free systems.

When Gjøvik celebrates its 150th birthday, there will be a need to show off the city at its best, which is a job done well by timelapse movies. Timelapse movies are brilliant at capturing the pulse of the many stages one can experience during a day in a city. This could be anything from a sunrise to a steam of people walking in and out of a popular building. Creating timelapse movies is a timestealing process which can be both tedious and troublesome. This project was needed in order to make the creation process easier and make the wish for a timelapse presentation of Gjøvik possible

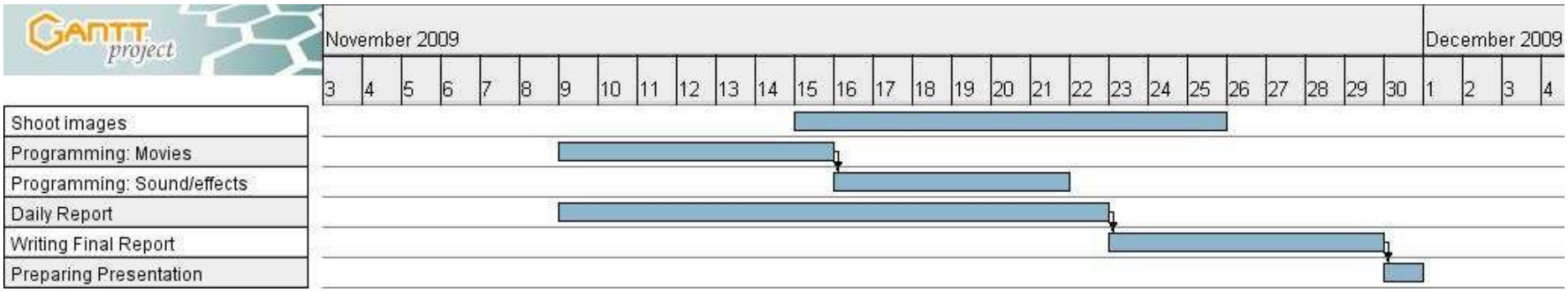
The goal was to automate the process of creating, describing and publishing a timelapse movie set to music by using various open source command line tools[1]. The entire process was kept as open as possible by use of CC[2] and open source material.

2.1 Planning

At the beginning of the project planning in week 43, a Gantt diagram was made in order to split the process up. See the Pre-Project report for more detail.

Due to uncontrollable variables, such as availability of cameras and weather, the real diagram ended up differently.

[1] Shortened to CLI
[2] Creative Commons


GANTT project
November 2009
December 2009
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4
Shoot images
Programming: Movies
Programming: Sound/effects
Daily Report
Writing Final Report
Preparing Presentation

3 Methods

3.1 Segmentation

The project was split into two segments, the gathering of data and the initial coding and processing of it. These two segments were not depending on each other during the process, which made it possible to work on each of them independently. The gathering of data was depending on the supply of special cameras from the school and when these were late this segment was pushed. Fortunately, this had no effect on the part regarding the coding, which could continue by the use of sample data. Another uncontrollable variable, the weather, also slowed down the process of gathering data. However, the final project did not suffer because of this.

3.2 Gathering of data

The images were shot using a Nikon D5000 and camera stand, provided by the University College.

The music was picked out by using a website[3] for distribution of music under Creative Commons Licences. Creative Commons is an alternative to the typically restritive compyright. As mentioned on Wikipedia:

> Creative Commons licenses are several copyright licenses released on December 16, 2002 by Creative Commons, a U.S. non-profit corporation founded in 2001.
>
> Many of the licenses, notably all the original licenses, grant certain "baseline rights", such as the right to distribute the copyrighted work without changes, at no charge. Some of the newer licenses do not grant these rights.
>
> Creative Commons licenses are currently available in 43 different jurisdictions worldwide, with more than nineteen others under development. Licenses for jurisdictions outside of the United States are under the purview of Creative Commons International.

3.3 Coding and testing the script

After segmenting the entire project, further segmentation was needed on the processes regarding the coding of the script. The processes were split into four main segments:

- Movie Creation
- Adding cut/fade to music
- Reading Input
- Adding metadata

[3] http://www.beatpick.com/

The working process consisted of working through the list point by point, finding the suitable software to solve the problem, closely followed by software implementation and testing with sample data. As expected from the task given, a unix based system and shell scripting/bash programming were used.

3.3.1 Movie creation

For creating a movie with CLI commands there were two main software choices, FFmpeg and MEncoder. MEncoder was chosen for this task because of its use of non-proprietary formats and codecs. MEncoder is also under the GNU General Public License, which was an important factor in order to keep the project as open source as possible. MEncoder is based on the same source as Mplayer and therefore has a huge number of available options regarding video. MEncoder also has the option to add music to its videos, which made the automation of this point fairly easy.

```
Example:
mencoder -audiofile song_fade.wav "mf://*.JPG"
-mf fps=$fps -lavcopts aspect=$aspect -ovc lavc
-lavcopts vcodec=msmpeg4v2:vbitrate=80 -o $title.avi -oac mp3lame
```

3.3.2 Adding cut/fade to music

After creating the movie, it was quickly found out that the movie would always last as long as the song was, as opposed to the movie length the stills provided. A side effect of this was that the entire movie lasted as long as the song went on, freezing at the last image. Since the movie is built up by still images, this was easy to overcome. The selected software for handling music was SoX, widely known as "The Swiss Army knife of sound processing programs". SoX contains a cut function which cuts off unneeded parts of audio. The amount of still images were counted and divided on the input fps, which produced the amount of seconds the movie would last. This input could be used by SoX in order to cut the audio and add a fade out to it. A couple of extra sound files were created during this process, however they automatically get deleted at the end of the script. The length of the fade was read in as a variable. It was easily possible to add a fade in as well, however this was chosen away for the time being.

```
Example:
count=$(ls -1 *.JPG | wc -l)
lengde=‘expr $count / $fps‘
sox *.wav song_cut.wav trim 0 $length
sox song_cut.wav song_fade.wav fade t 0 0 $fade
```

3.3.3 Reading input

In order to make the automation process better, there would be need for some input from the user. This is important in order to give the user, as much as possible, a chance to affect the final result of the movie. A selection of variables were read and saved through bash code in the shell script. Movie title was read in order for the user to name the finalized movie. The FPS was also read as the end film is a timelapse movie and the fps has a lot to say in regards to the final effect. Another option for user customization is aspect ratio. The user might be making a movie meant for either TV's with different aspect ratios, PC screens and other formats. That's why it is very important for the user to be able to chose this option. Another option was the amount of seconds used to fade the audio at the end of the film. Sometimes it is desirable with a long fade, sometimes none at all.

After reading the input from the user, a confirmation came to let the user know exactly what (s)he had written.

```
echo "Enter movie title:"
read title
echo "The title has been set to $title"
export title
```

3.3.4 Adding Metadata

Adding metadata to the film was concidered a very important point, as it often is the only additional info that follows a movie widely distributed. There were several points who were desireable to add to the metadata of the film:

- Name of the creator
- Creative Commons Licence
- Information about the music used
- Additional comments

For this job, the open program ExifTool was wanted used. However, it was later found out that ExifTool can only read and not write metadata to various movie formats. At the time, another way of dealing with metadata was not found and the entire point was put on hold. The adding of metadata is concidered to be an important point in regards to future developement as it is highly wished for.

4 Results

4.1 The final script

When the final script is run, it should happen in the same directory as the audio and still images. The images should, for the time being, stay in JPG, while the audio should be WAV. As the script is running, the user get several prompts where it is possible to customize the film. After each input, a confirming message is given to show the user the written input is correct:

```
javi@javi-laptop ~/Pictures/bla $ ./Timelapse.sh
Enter movie title:
Timelapse_test
The title has been set to Timelapse_test
Enter FPS, single number:
24
The FPS has been set to 24
Enter Aspect Ratio (F.ex: 4/3 16/9):
16/10
The aspect ratio has been set to 16/10
Enter length of audio fade out(in seconds):
6
The audio fade out has been set to 6
```

The movie will then render. This varies, depending on the amount of images and their quality. As the script finishes off, it cleans up unneccessary files and automatically starts the movie for the user to see.

4.2 The Movie

The finalized movie became, depending on the input form the user, a high quality video set to music. It was quickly found out that if the images used for the timelapse were too big in resolution or file size, it would only slow down the process of rendering without much gain. Therefore it is reccomended to keep the image file size and resolution as low as possible, without it breaking the end quality of the film.

Sample frames from the final movie:

5 Conclusion

5.1 Is it possible?

Through this project it is learned that the automation of timelapse movie creation is very much possible. Through the use of software like MEncoder and SoX in a Shell environment it is possible to create simple movies. However, it has not been researched to how much of an extent it is possible to go to when automating the process in this manner.

The value of this project lies within its simplicity. As mentioned in the introduction, creating a timelapse movies from scratch can be tedious work, specially if you are creating many.

5.2 Is it necessary?

While this automation can not comete with professional editing software, it proves an effective way of creating simple timelapse movies which results would have been identical when using complex software. There are several professional video editing programs available with options regarding creating of timelapse movies. However, these may cost money, steal many computer resources or they may simply not be wanted used by a certain amount of people. The running of the script could possibly save a lot of precious time while creating timelapse movies and could be even more useful if built out more.

5.3 The future

There are many ways in which this project can be built upon. Further automation of typical movie actions is particularily wished for. Examples of this could be the adding of a title and credits sequence, additional frame fading at end/beginning and in between clips. Additional options for metadata and movie preferences should be implemented as well. A scenario could be that the user is able to chose several profiles specifically tailored for different tasks, such as video for web, TV, high definition etc.

In addition, it should also be possible to create a cleaner process where the user each time will be prompted to chose different directories for images and sound.