VAMIX Report

Logan Horton

Table of Contents

Illustration Index

Brief Summary

The VAMIX product is a Video a
nd Audio Mixer, which allows basic functions to do with audio and video, allowing downloading and opening of files and generic playback functions. This was implemented using Java functions, the vlcj library and linux based commands.

The functions added were:
-Playback (including seekable timerbar)
-Play/pause/rewind/fast forward/mute(and unmute)/volume bar
-Opening of files
-Downloading of files (check for open source also)
-Strip audio (getting a muted video and an audio file)
-Replace audio on a video
-Overlay audio on a video
-Text at beginning of video (title text) (80 character maximum)
-Text at end of video (credit text) (80 character maximum)
-Mutlitude of options in size/colour/style of text
-Load/save of options and text from the text panel
-Trim video
-Flip video
-Rotate video
-Add subtitle file to the video in use (with generator for the subtitles)
-Edit subtitle file in VAMIX
-General checking of file inputs to enable/disable functions depending on type (audio/video)

More general features:
-Swingworker usage meaning non-freezing UI
-Can create multiple files at once by using different panels at once
-Calm and dark colour scheme (grey in general to be easy on the eye)
-Loading bars to display that functions are completing their tasks
-Helpful messages if issues occur
-Helpful messages to appear if you have not correctly set up a task
-Help functionality for the subtitle panel

Target User

The VAMIX system is suited to being used by people who are reasonably proficient with computers and software packages, meaning it is suited for the middle aged group or teenagers to adults. This is further emphasised by the way VAMIX was constructed.

VAMIX contains a lot of information inside of it, but in general, the information is small (reasonably) and is compact for quick use. This would be a hinderance for the elderly and young that would use this, which is why it was crafted for the midground between these two groups. The smaller interface and compact button setup, with a flow for how things are crafted (buttons etc... are in a readable order) allows for quick crafting of a video file as all movements of the mouse are quick and efficient, perfect for people who have had experience with computers.

Illustration 1: Basic setup of button functionality

This image (Illustration 1) helps to convey this setup. The buttons and text areas are all alligned and setup in an easy flow of work. You begin with the opening of audio, then entering an outname, then one of the options. This setup allows for quick working through the system to create the output file required.

The

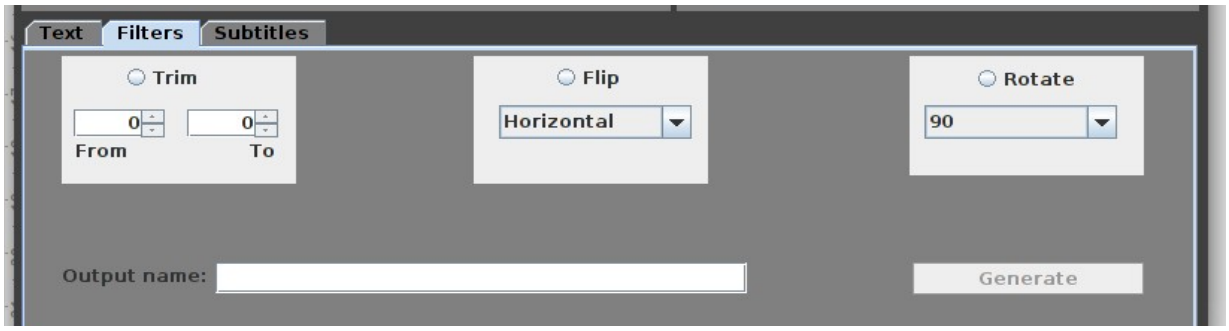

Illustration 2: Colour cooperation display

illustration 2 above displays another feature of the system used to help the target group create files quickly and efficiently (both time wise and with minimal effort). By utilising grouping of sections of information using colour, the user will be able to very easily see what sections refer to which. This allows easy use of the system as the colours will guide you to enter the correct information for the operation wanting to be completed.

The other efficiency measure thought about was making sure the panels were seperated well. By seperating the panels well, the user can use one panel to create a file, and while that operation is completing, you can work on another panel, allowing efficient use of time.

Design Decisions

Code and coding UI

VAMIX was written and designed for Java language utilising the vlcj library for its video capabilities. Java was chosen due to its cross-platform usability and its ease of creation. There is a wealth of knowledge on Java language which allows for easy searching for information on any issues and any unknowns to do with the system. The vlcj library, mainly the “avconv” commands also have a wide use meaning there is a lot of knowledge on them available.

The Eclipse UI was selected to make the system in due to its ease of use. Eclipse has a large internal information library that can be viewed to get help on a lot of simple issues, while also having error checking before compilation which heavily reduces creation time for the code. The eclipse UI also allows the use of plug-ins which were used during the creation of VAMIX. The main one utilised for the system creation was WindowBuilder. This plug-in allowed for easy creation of the systems UI as it displays the look of the UI before you run the program, allowing you to move sections around and get it looking correct before implementing any functionality. This helps get a feel for what the system should do and helps the one coding the system not forget any parts during creation.

Display considerations and information presentation

The colour scheme and display (working layout) of the VAMIX system were very important during creation. The colour scheme selected was a dark scheme, using dark and medium greys for the majority of the scheme, with light greys used for buttons and highlighted areas. This was mainly chosen due to the colour grey being soft on the eyes, and having a dark colour helping with eyesight not getting sore after hours of extended use. This system was designed to be used by people in their middle years, where in a workplace, the user may need to use the system for an extended period of time. Because of that fact, it was decided to use a dark grey scheme to try and make the experience as painless as possible for the user. This scheme can be seen in illustration ½ above.

The layout was also important, as efficiency was seen as very important. This is enforced by the system being very compact. To ensure this, the majority of the options are visible from the startup screen, with only 2 panels being hidden inside the tab-panel. This helps to give an instant view of what you can do, allowing for quick pick up and use capability. The all shown idea also helps as there is a minimal number of clicks between using any command meaning you can use the system very efficiently. This layout can be seen in illustration 3 below.

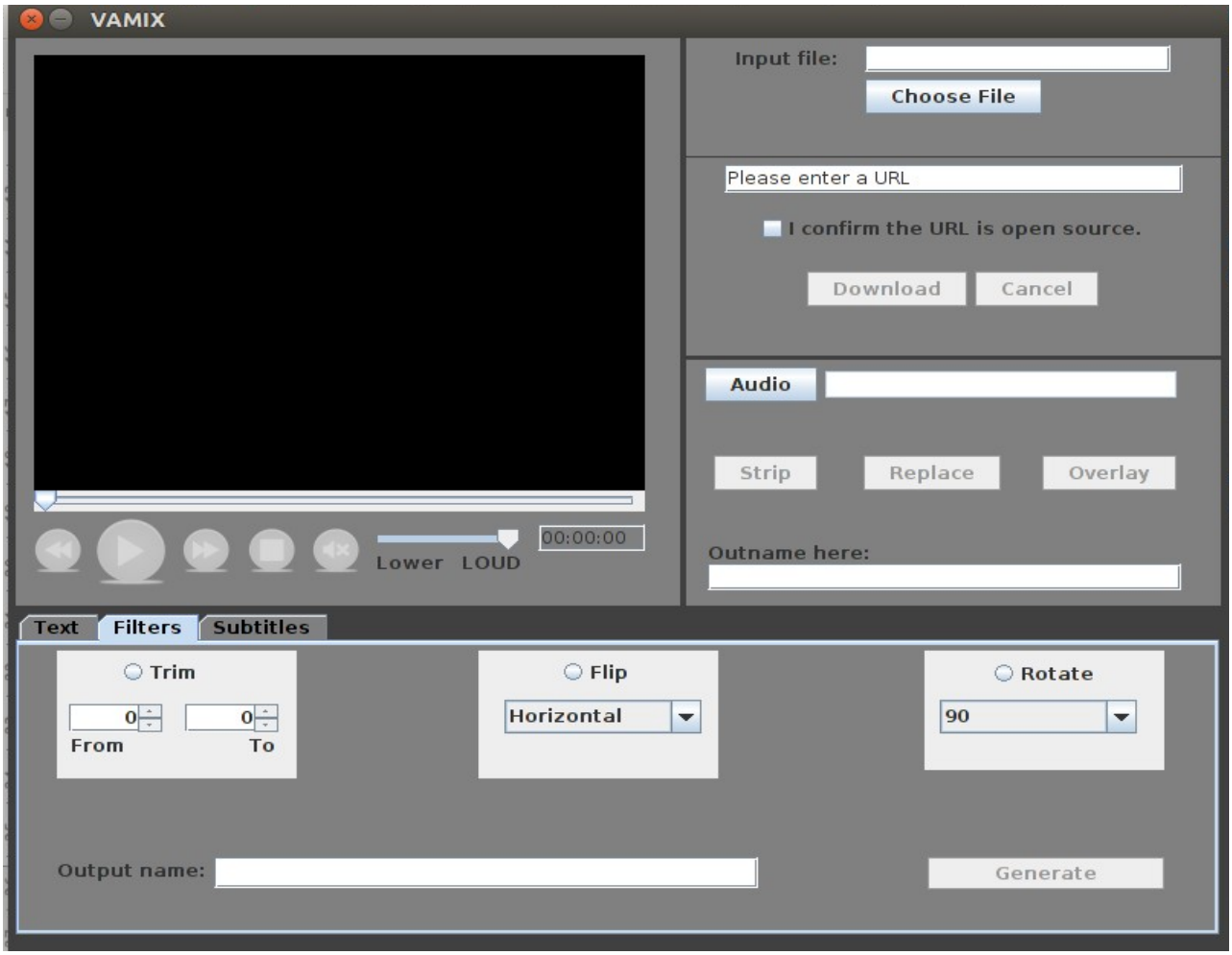

Illustration 3: The VAMIX system

The biggest issues found while trying to attain this layout was the functionality vs layout. The number of options possible and the size of information was restricted by trying to keep to a display-all style. This is visible in the need for 3 tabs at the bottom, where it would have been preferable to have all visible, but it just was not an option due to the problems with size and the amount of information needing to be displayed being too much.

There was also the consideration of resizability. The system is made to fit any screen as it is made to fit even 720p screens. Having this resizable would mean that the size of every button and text field, and the gaps between everything would be required to scale alongside the resizing of the system. This was deemed un-neccesary as it is not something that would cause a large issue with and user as even a large screen would display this well. The only downside of this decision is that the played would not resize for this decision not to clash with others. Though this could be seen as annoying, it was deemed fair as the system is not a video player, it is a video modifier and thus the decision to not have any resizing/fullscreen capability on the screen was deemed fair. If the user needs to view the video, there are players on any system readily accessable that would paly the video to a higher quiality or the same as VAMIX and be able to manouvere through them easier than that possible when using VAMIX.

Evaluation and Testing

Self testing

The system, through production and after production, was tested thoroughly. This was both through the use of general value tests and through self testing by entering a variety of values, from outside scope, on the boundaries, and inside the scope of the option. By testing this, it was possible to get rid of a lot of the errors and “bugs” in the system. The post-creation testing was done using outside sourced persons, which allowed for a number of general layout issues and combination of action issues to be fixed.

These issues from self tests were mainly to do with layout issues, such as a output file name being after a generate button. These issues were removed prompty.

Peer review evaluation

Code

In general, the code submitted resulted in a overall view of inconsistency. This was mainly to do with the lack of layout in the code, as there was no packaging at the time it was submitted, and there was a lack in proper commenting.

First, the classes were put into packages associated with the actions they performed. There was also a comment about moving the SwingWorker code into its own package and set of classes. This was thought about and deemed not possible for this project. The SwingWorker instances were created inside the classes they would be called from which increased the size of those classes unnecessarily. It would have been beneficial to move the instances into their own classes but the Workers were heavily intertwined with the fields in the classes that contained them. A large majority of them required 3+ fields from the class they were in meaning the swingworkers would require a lot of reworking to get them into their own classes. As the general class length was not too large, it was deemed not neccesary to change it, but for future ventures, it has been noted at a good idea and will be considered and applied to future uses of SwingWorker to prevent this issue occuring again.

Functionality

After the submission, it was clear there were a number of issues to do with the preview button not working correctly, and there being a few minor issues with error checking not working as intended and locking out functions once used once. These minor errors were fixed quickly.

The next major issues brought up was the subtitle panel. There were complaints about the subtitle panel being directly editable, meaing you can write into it and possibly break it. This was considered and deemed neccesary. Though this is prone to creating errors, the .srt loading from vlcj does handle most errors, and worst case is the subtitles dont load into the video. The main reason for keeping this editable is that, if you make an error when entering, such as a spelling mistake or a

timing error, the only way to fix it would be to open the .srt in another program nad edit it, then reload the file, then run it with the subtitles. This was seen as slightly too much to ask of a user, so it was left open to editing but with a help button to the side of it to help with the formatting if the user decides to directly edit the subtitles.

Interface Design

The main greivance with the interface was the button inconsistencies and some buttons having odd naming causing confusion.

The button inconsistencies were fixed, as there were slightly different button images for a few different buttons. These were changed to be consistent to help with ease of use. The panel buttons were also resized to be a consistent shape and size relative to the other panels, to help make the interface look cleaner.

The naming of buttons was the biggest issue from the peer review. There were a number of buttons, such as load/save state that were confusing both due to location and lack of documentation internally (such as better naming/a [?] button). This was heavily considered as it is important to have an easy to use interface, but was seen to be difficult to change. The load/save state was generally fine to most users, and the documentation in the README helps explain the features of all the buttons on each panel. The addition of the [?] button for subtitles was extremely needed as it would help people understand how to use the complicated .srt formatting.

General

There were a number of comments on the lack of output file location selection. This was done intentionally. There was originally a location selector but this caused issues when trying to save the .srt file for the subtitles and when trying to look for the audio files for the audio panel. This was easiest to solve by stating in the README that the files to work on should all be in the same directory as the .jar file running the operations and all output files will also be saved to that location. By using this setup, the location of everything becomes very simple. The only exception to this is the file selector for opening a file, where the opener can open from anywhere on the system, and will save the .srt to the same location as that file, but all output files will be in the .jar location.

Comments were also made about the layout naming. There were requests made to label each of the panels to help with readability of the layout, but upon inspection, it was decided that it was not needed. The open/download/audio panels all have the name in the first button, giving away instantly what the panels are for. The tabbed-pane has labels suited for each of the operations they hold, making those easy to understand. It is fair to say the audio panel is not instantly understandable, and thus more information was added to the README about that panel to help it be more understandable.

Honourable Mentions/References

-The system was developed mainly solo, with one section being a pair section with Sam Butchart where the creation of the start/end text and the video playing was handled by him.

-www.stackoverflow.com helped in creation of a number of the “avconv” commands that handle the video editing and audio editing.

-www.libav.org/avconv/html allowed basic “avconv” commands to be created and for outsourced commands to be edited for the systems exact needs.

-www.capricasoftware.co.uk/projects/vlcj/index.html allowed for video playback to be completed, by utilising the vlcj library.

-Dr Catherine Watson and Dr Nasser Giacaman – The two helped with creation with Nasser helping in the general coding basics for the system, and Catherine helping with the visual side of the project, and how the UI should be for it to succeed.