COMPLEX GAME SYSTEMS ASSESSMENT

AUDIO EDITOR TOOL PROPOSAL DOCUMENT

BY MATTHEW LE NEPVEU

Table of Contents

Overview of Game System

The main purpose that the Audio Editor tool strives to deliver will be to allow users to load and save audio tracks into the program and adjust various features of the audio track (e.g. Pitch, Frequency, etc.). The editor will also be featured as a .lib file and a .dll file so it can be transferred and used in other places.

In terms of usefulness, this program would be a handy tool to create sound for any sort of project. Because the Audio Editor has the ability to load, save and adjust audio to meet a user's standards, a group would be able to utilise the tool to change the features of any sound effect or music track and as a result, they would be able to save it out and use it for whatever they want without needing to search and settle for stock music all over the internet.

Functionality

The application will aim to include the following functionality:

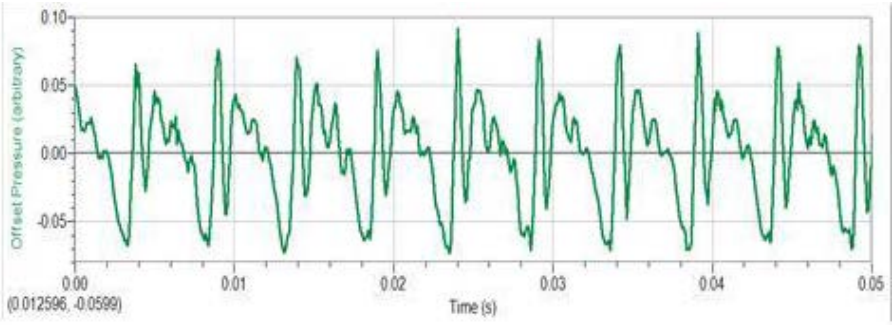
- Loading audio files into the application:
    - This will be the core functionality of the entire program. It will be designed to load files with the file extension of either ".wav", ".mp3" or "mp4" from the user's computer. Loading can be performed after starting up the application by pressing the "Load" button towards the top of the window.

- Saving out edited files from the application:
    - Along with loading files, saving files will be another key functionality of the Editor. Assuming the user has audio already loaded into the program, they can click the save button towards the top of the window to save out their edited audio and can save it as either a ".wav", ".mp3" or a ".mp4" file.

- Having the ability to adjust the features of any audio file:
    - Once the user has loaded in an audio file, they will then be able to use the control panel and tamper with the characteristics of the loaded in audio. This will all be displayed in the window of the application. The user can alter features such as the frequency, pitch and speed of the audio.

- Play back any loaded in or adjusted audio:
    - When adjusting with the audio, the user will have the option to, at any point, play back the audio either from the beginning or from a specific point in the track. This will become possible with two play buttons being present in the bottom of the window, one being for playing back from the beginning of the track, whilst the other will be for playing back at a specific time code of the audio.

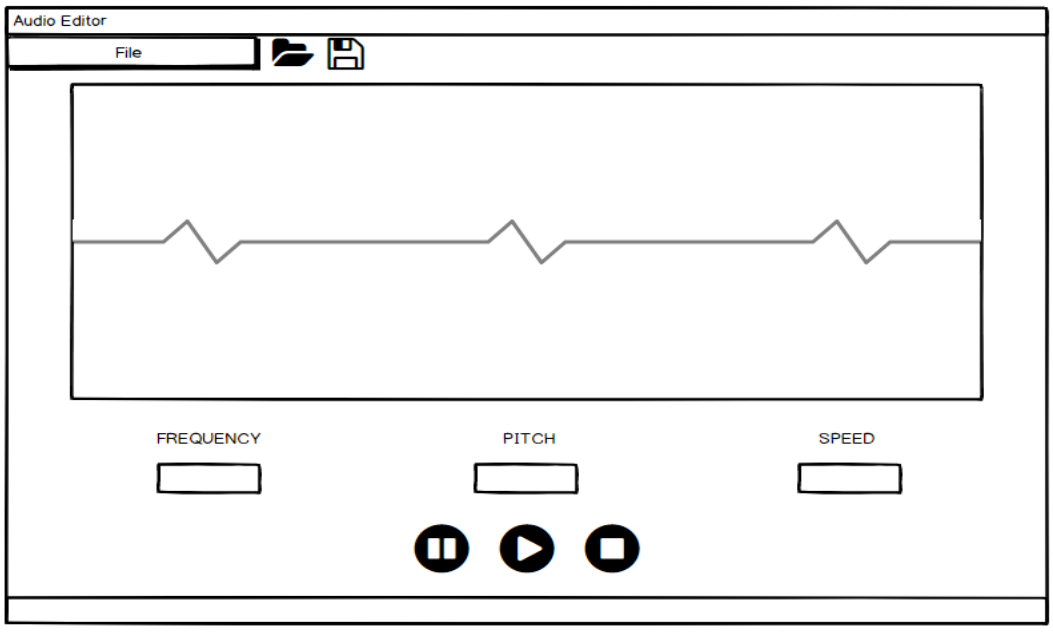
- A graph to visually display the graph:
    - This would be an additional visual element that would help the user see what the sound would look like as a wave. This would be displayed once a sound is loaded into the program. If the user chooses to adjust the frequency, pitch or speed of the audio, then the graph would change once the change has been inputted.



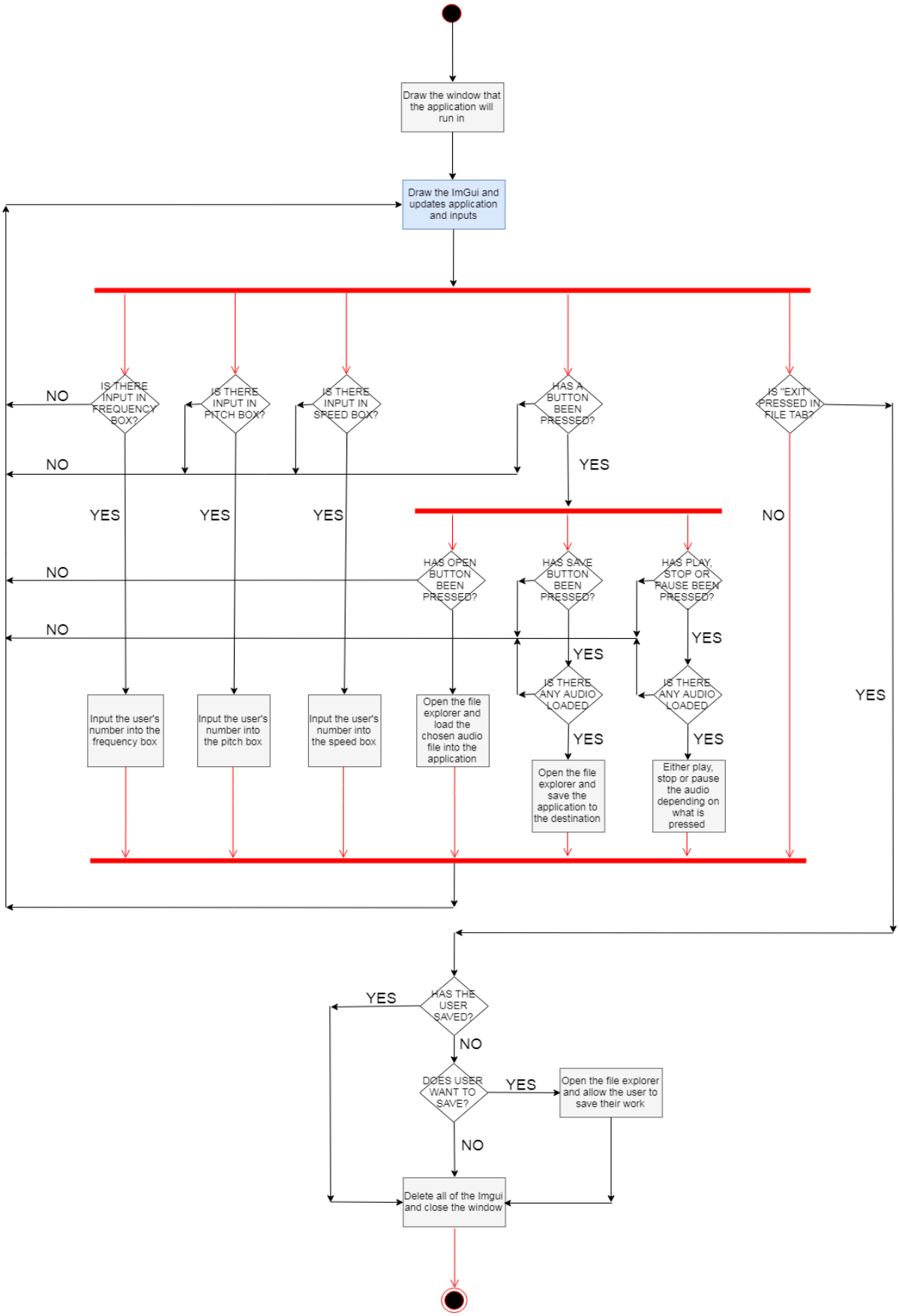
Minimum Viable Product

For this assessment task, the project must include all the functionality that were outlined under that heading. The control panel for the program will include either slider bars or digital knobs to adjust the frequency, pitch and speed of the audio track as well as having basic buttons for loading, saving and playing back audio files.

The Minimum Viable Product for the Audio Editor would hopefully look like the mock up below:



Down below is a UML Activity Diagram which shows how the Minimum Viable Product would run:



Extra Features

If there is enough time left, extra features can be implemented into the project once the Minimum Viable Product has been achieved. These features are not requirements as they are complicated attributes to include and the application would still hit a pass mark, even if the extra features aren't additions to the program. As a result, these features have a lower priority then the functionality listed above.

The functionality that could be added may include:

- A button that converts the loaded sound into 8-Bit:
    - This would be a complex task to do, therefore being why it won't be a requirement for the project. In saying that however, it would be an interesting feature to include so if there is time, it may be included as part of the final application.

- Drag and Drop elements:
    - Using this as a feature would not be necessary for the application to perform hence why it is an additional feature. However, it would still be cool to have the user load all the audio files in and place them on the left side of the window, then have them drag it onto where the graph would be and have the graph change to suit the audio's properties.

- Play the audio backwards:
    - This seems like it would be easy to implement however it is a quirky element to have so there would be no need for it to be in the minimum viable product. Despite this however, it could be an element added if there is enough time to do so.

- Visual line moving along the graph when sound is playing:
    - This would be an addition that would help the user visualize how the graph works in tangent with the audio. It isn't mandatory to feature as part of the application however it could be included as an extra feature.

Risks and Concerns

The biggest threat with the application being completed would be the amount of time allocated for the project to be complete. This means that there are limitations in what can be included into the project as there may not be enough time to include certain features or polish the program to perfection. Therefore, it is important to acknowledge the Minimum Viable Product first before adding anything else to the application.

The fact that I am familiar with C++ now would be a strength for getting the project completed on time. This means that the coding part of the project shouldn't present me with too many difficulties. This also goes for the fact that I am familiar with Visual Studio as an external tool so there shouldn't be too much to worry about with Visual Studio.

Being that the application is based on Audio Programming and Engineering, completing this application to a reasonable standard would be a handy portfolio inclusion for if I wanted to search for a job in those areas. It also could present me opportunities for working with ImGui, as the project would be helpful for showing that I have worked quite thoroughly with the system.

Weaknesses and other threats to the completion of the project could also include the fact that I have never worked with FMOD before, which could mean that the project completion date may have to be pushed back. I am also limited in the amount of resources I am able to use as some may cost too much money for me to utilize properly to create the application.

External Tools

This application will be completed using three main External Tools. These tools are:

- FMOD.io:
    - This is a complete library full of high quality sound effects. The library is designed for use for games however it can be utilized for sound applications too.

- ImGui:
    - The ImGui system (which is short for Immediate Mode GUI) is a system that is commonly used for creating new editor windows and tools. This will be used so the actual window for the application can be drawn.

- Visual Studio C++:
    - In this case, C++ is the programming language being used to write the application in and Visual Studio is the program that will be used to write the code in.

References

- Author Unidentified. (2013). *NGSS – Using Mathematics and Computational Thinking*. Available: https://physics242.wordpress.com/tag/circular-motion/. Last Accessed 3 April 2018.
- Author Unidentified. (2018). *Fmod.io.* Available: https://fmod.io/. Last Accessed 3 April 2018.
- Unity. (2018). *Immediate Mode GUI (IMGUI).* Available: https://docs.unity3d.com/Manual/GUIScriptingGuide.html. Last Accessed 3 April 2018.