UP818988 CT6SOMUP

SPECTRAL RANDOMISER

ABSTRACT, AIMS AND OBJECTIVES

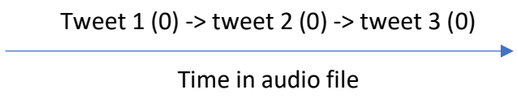
The Spectral Randomiser is a tool to randomise points of interest in an audio file to be used within a video-game.

The aims of the project are:

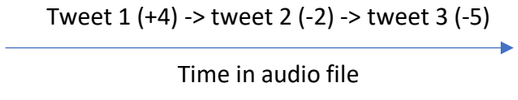
- Use spectral processing to randomise points of interest in an audio file
- Run the program alongside a game to randomise audio in real-time
- Display the frequency spectrum of the current audio file so the user can choose which points of interest they would like to randomise

The user would be able to create many events using Spectral Randomiser as an integrated audio middleware system. The main focus being ambient audio. A great example would be bird song. The objective would be to capture individual tweets and randomise the way they sound separate from each other despite being in the same audio file.

First time playing the audio file:



After randomisation on second time playing through:



The number in brackets represents an arbitrary pitch shift from the original pitches. Every time the audio file is played in game, it will have a different pitch on each of the points of interest.

CONTEXT

The inspiration of the project was from the randomise function that Wwise has and looking at an innovative way to increase the quality of randomisation in video game audio. Using randomisation is important in video games because using the same recording over and over would quickly sound unnatural and become an audio nuisance for the person playing the game (Audiokinetic, 2019). In the real world, when the same sound is heard or emulated again, it is different because it is impossible to recreate the exact same conditions for a sound to be the same. For example, a blacksmith hitting an anvil in the exact same way still creates a different sound to the last. This means that to create a believable audio setting in a video game, it is natural to use randomisation of audio to emulate how the real world would sound.

The current way of randomisation is either through; pitch shifting, low-pass filtering or high pass filtering. Despite these currents ways of randomisation are still very effective, the Spectral Randomiser is to gain an increase of control in the randomisation process. The problem with the current way is when the user wants to randomise files with long durations such as ambient noise effects. The difference between the two randomised files become noticeable because each time it's played and randomised, it applies the randomisation across the whole file. Randomising points of interest will keep the pitch values to a similar average each time that the audio file is played making it sound more natural.

IMPLEMENTATION

The Fourier Transform is the way to capture these points of interest in an audio file. The Fourier Transform represents the frequency domain of a discrete time signal in a certain period. The period in this context would be whole audio file that the user would like to analyse and process. By analysing the file with the Fourier Transform function, the magnitude of the frequencies can be captured and therefore information about the current points of interest in the audio file can be displayed to the user and used for processing.

Despite that the current version of the Spectral Randomiser does not work, its program flow is as follows:

1. Load in an audio file using the file chooser.
2. The file chooser puts the audio file's buffer into an AudioSampleBuffer.
3. The BinScrambler class takes the AudioSampleBuffer to use for processing.
4. Perform the forward fast Fourier transform on the buffer to create the spectral data.
5. Calculate the magnitudes of the frequencies using Pythagoras' equation: $c^2 = a^2 + b^2$. Where; c is the magnitude, a is the current real number and b is the current imaginary number.
6. Swap/randomise the magnitude spectral bins.
7. Perform the inverse fast Fourier Transform on the spectral data.
8. Load the newly made buffer from the inverse transform into the transport source that is used to play the original sound file of which, the audio would be randomised and different to the original.

EVALUATION

The project is nowhere near a success but, was a huge insight for personal development into advanced DSP programming. The project in its current state is just an audio player but it includes a class that looks into how to start processing the audio files.

The issue was that the project was too ambitious for my current level of programming, paired this with the lack of time spent on the product itself. Before the programming even started, I should have spent a large amount of time understanding digital signal processing and how samples and buffers work. Then looked into how samples and buffers are applied specifically within C++ and how to manipulate them. I did have prior knowledge of how samples and buffers work but only when it came to real time synthesise. The processing of audio files using FFT was too difficult especially that the literature surrounding it was too advanced to understand for my current knowledge.

BIBLIOGRAPHY

Randomizing Properties, Audiokinetic 2019 sourced from: https://www.audiokinetic.com/courses/wwise101/?source=wwise101&id=randomizing_properties#read

Digital Signal Processing – DFT Introduction, tutorialspoint, 2019 sourced from: https://www.tutorialspoint.com/digital_signal_processing/dsp_discrete_fourier_transform_introduction.htm