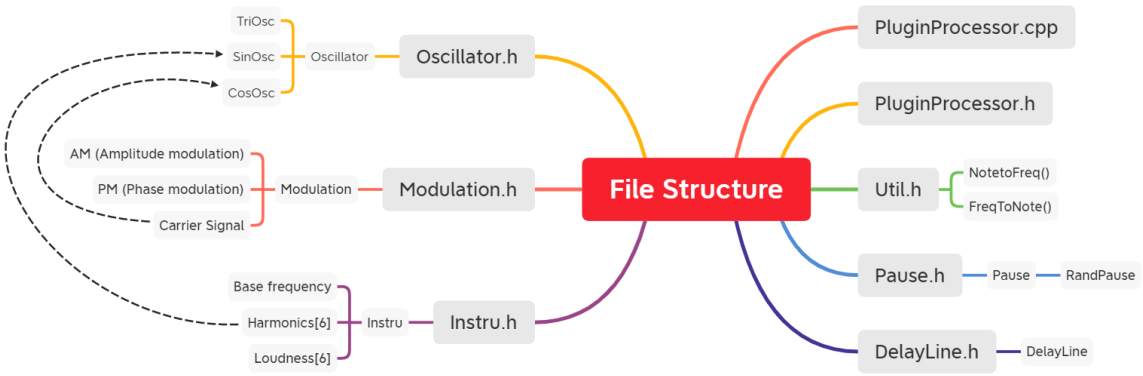
Assignment 2

File Structure

Other than .cpp/.h file that JUCE generate automatically, I implement the following header file for different usages:

- **Oscillator**: Generate waves for periodic changes in other modules and processing. Three subclasses generate triangle, sine and cosine waves.
- **Modulation**: Take signal as input and modulate it into carrier signal. It has two modes of Amplitude Modulation (AM) and Phase Modulation (PM).
- **Instru**: Generate sound with harmonics (imitating **Instru**ment).
- **Util**: For now, there're only two functions that convert note and frequency and vice versa.
- **Pause**: Make pausing effects. Can set specific pause time or random pause time in a range.
- **DelayLine**: Input and store signal, output delayed samples.



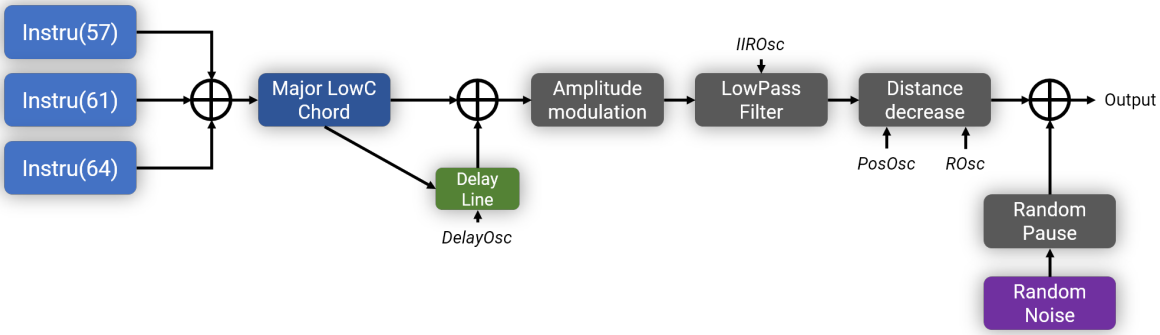
Flowchart

As the assignment introduction says, we want to make *drone music*, which doesn't rely on notes and rhythms. At first, I heard some provided pieces and tried to make some clips on my own. Continuous sounds with high frequency is very uncomfortable to me, and ones with low frequency are much better. Imitating natural sound like water waves is what I originally want to do, but it proved to be hard to generate satisfying samples with only oscillators. In the process, I found some pieces were similar to background music for horror atmosphere.

For a thicker sound rather than sharp one by single frequency signal, I searched online and implement chord and harmonics. Delay line we learned in course was just flanging effect needed, which make a distortion effect. A lowpass filter is added to remove high frequency and spikes in processing.

Stereo effect was ideal to simulate a moving sound source, like a ghost wandering around. Random noise and pause are added at last, making it sound like "interfered". Those two are very common audio elements in horror games.

The whole process is shown as follows:



1. Generate three samples with harmonics and add them together. Their base frequencies is set to A(57), C#(61) and E(64) for a chorus.
2. Write samples to delay line and read delayed samples from it. The delayed time is controlled by a LFO from 7.5ms to 12.5ms for flanging effects.
3. Amplitude modulate samples. Just for interesting effects here.
4. Pass a low pass filter to make it sound lower. Its cut is controlled by another LFO.
5. Calculate sound source's virtual "position". Here, I make the sound spin in a stable speed, with radius go up and down. All of these periods are also controlled by LFOs. From the position, we can get its distance to two virtual ears, and decays the loudness by the distance. It's very simple stereo effects compared to professional ones like Dolby, but sounds not bad to me.
6. Last, we generate noise and pass it through random pause. It will make short time of noise with random intervals. Add the noise to previous samples make the final output.