There are two main purposes of the creation of this plugin.

Firstly, I was inspired by our second assignment where we were required to build drones. This type of music can be considered as ambient music, however the interesting part of the drones that we built was that it was generative. Although in the assignment the values were already determine, I thought that there was a lot more potential for the music by taking in input from elsewhere. In particular, I was inspired to apply this to games, where the input to determine the music could be from the environment in the game, event triggers and player movement. Therefore, my goal was to create a plugin which could potentially be converted into a plugin used by game where interactive ambient music is needed.

The second purpose of this project is to create a plugin which is accessible to users who are less familiar with music theory but would still like to create music. My goal was to provide some control for the user to determine some aspects of the music while still ensuring that the music sounded good. The more interesting aspects that users could control from this plugin include the mode of the music and the combination of modes between the different synthesisers. Besides that, users are free to determine the overall structure of the music (tempo or time signatures or the lack of) regardless of the plugin. To summarise, the use of this plugin is meant to be easy, without needing much knowledge in music theory or skills in keyboard playing while still generating nice sounding music.

Since I aimed to make this plugin easy to use, it relies heavily on predetermined and varied elements to make interesting-sounding music. Therefore, a more "complete" sounding music would require harmony, melody and some rhythmic element or ornaments. These are covered by the plugin as the plugin can be separated mainly into three sections, each section covering mainly one of the elements although some aspects of the elements cannot be purely isolated from one another.

The harmony is mainly controlled by the middle range of the midi notes (between C2 (inclusive) and C3 (exclusive)). The middle range is controlled by "synth2" which uses the voice from "FMsynthVoice". In "FMsynthVoice", I have determined 6 variations of chord forms in "setFrequencies()". The chords (seventh chords) are built upon the note played by the user, which is used as the base note (tonic) of the chord. The user can select the modes that they prefer for the chords in the interface. The modes will then be randomly chosen from the user-selected modes. The purpose of this is to allow some flexibility to control the mood of the music depending on the user's needs. For example, the Aeolian mode could be interpreted as sad while the Lydian mode is often thought of as bright. Another aspect that I was interested in was the combination of modes. Nice harmony could be generated even by just hitting the same note in this particular range as music sounds good with different mode even when based on the same mode (parallel modes). However, for some of the modes, the traits are not obvious in the seventh chords. Hence, melody can be generated ("MelodyVoice") in the lower range of the midi notes (below C3).

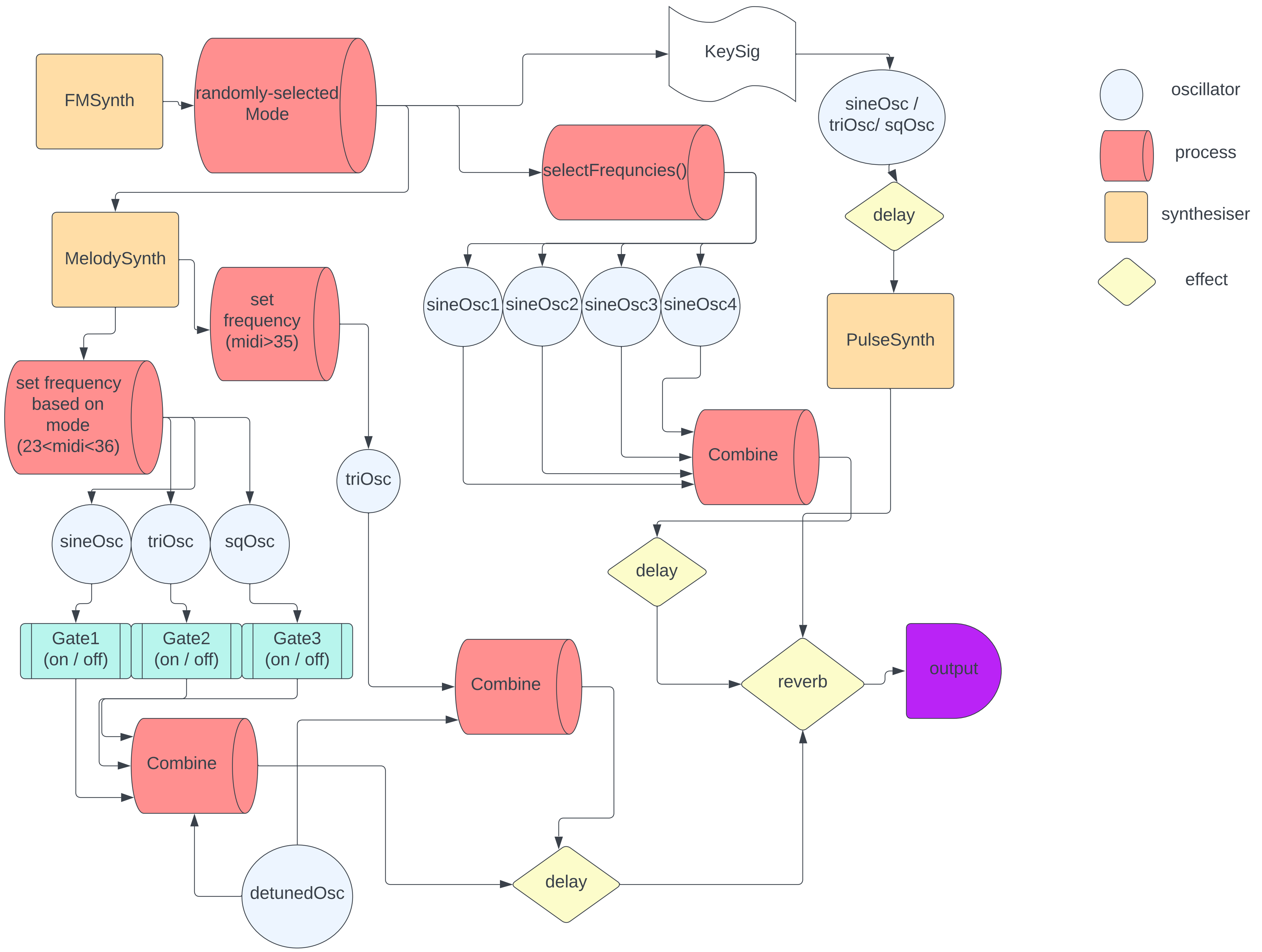
In "MelodyVoice", between C1 (inclusive) and C2 (exclusive), the notes are locked to fit a randomly-chosen mode based on the user-selected modes from the interface. The mode is changed every time a note starts in "FMsynthVoice". The reason the mode chosen for "MelodyVoice" is not set to be the same as the mode for "FMsynthVoice" is to provide more interesting melodies to the music. The mode for the melody could or could not be the same as the mode for the harmony, this create interesting moments in the music while increasing the variation in music. In addition, the combination of modes is often seen in blues music, so it was interesting to implement this to hear the outcome. The user is free to create melody without the fear of playing notes that might want to be avoided. Thus, the player only need to worry about the rhythm and touch (velocity) of the notes.

For below C1, the notes carry the same midi value as the midi input. This is to provide some freedom in the base of the music.

In the top range, “pulseSynthVoice” controls the music where pulse-like sounds are generated. The purpose is to create some variation (ornaments) or rhythm to the music. These “pulses” are varied in their frequencies and waveforms (sine, square and triangular). Moreover, the speed for the pulses can be changed with the mod wheel. The pitch of the notes are again determined by a mode, chosen in “FMsynthVoice”.

Besides the randomness of modes and waveforms, another interesting aspect is the use of velocity. The velocity is used in the synth to determine various elements such as detuned amount, ADSR parameters and pitch. The velocity is scaled in various ways and combined with some randomness to fit the various elements. Again, this helps with adding variety to the music.

On the next page, there is a flow chart illustrating the relationship between oscillators, synthesisers and effects.


FMSynth
randomly-selected Mode
KeySig
sineOsc / triOsc/ sqOsc
selectFrequncies()
delay
MelodySynth
set frequency (midi>35)
sineOsc1
sineOsc2
sineOsc3
sineOsc4
PulseSynth
set frequency based on mode (23<midi<36)
Combine
triOsc
sineOsc
triOsc
sqOsc
delay
Gate1 (on / off)
Gate2 (on / off)
Gate3 (on / off)
Combine
reverb
output
Combine
detunedOsc
delay
oscillator
process
synthesiser
effect