FM Synthesis Sample Composition with Processing and Beads

William Kostiuk

COSC 4P98

Application Description

For my project I chose to create an FM synthesis sample composition creator. This application uses Processing with Beads to create small compositions much like a midi development environment, however it also allows the user to alter the parameters of the underlying FM synthesis engine in real time in order to find a desired timbre. Once a pleasing timbre has been found and a small melody has been mapped out the user may export the sample as a .wav file for use in a larger project.

The user begins by creating a melody by placing notes along a timeline by clicking note boxes where they wish a note to be placed. The notes may also be removed by clicking them. Once a melody has been scratched up the user may click the 'play' button to preview their melody. The created composition will loop continuously until the 'stop' button is clicked. At this point they may wish to alter some of the of the FM synthesis engine's parameters such as standard carrier signal frequency, modulation ratio, carrier waveform, and modulator waveform.

Technical Foundations

Frequency modulation is an audio processing technique that allows a waveform to be altered by compressing or expanding the wave according to the pattern of a separate modulation wave. This effectively changes a standard tone that is easily produced with the Beads environment into one that changes in pitch over time as the frequency of the wave is altered as the tone is played. This allows for the creation of wobble effects and, at higher frequencies, completely new timbres.

The FM synthesis process used is detailed below in the architecture section, however the fundamentals of frequency modulation are as follows:

- Create a carrier wave at a chosen frequency
- Create a modulator wave with a frequency that is a ratio of the carrier wave frequency
- Map the frequency of the carrier wave to the value of the modulator wave

Architecture

This application is centered on the Clock object **mainClock**. A beats per minute (bpm) value is set in the code as it is unable to be changed dynamically after the program beings execution. This value determines how frequently **mainClock** produces a beat that may be reported to other objects.

The function of **mainClock** is to iterate through a two dimensional array **beatArray** that contains information about when tones should be generated and at what pitch they should be generated. The x-axis of beatArray is iterated through by **mainClock** with each x value being a time slot. The program contains 60 time slots, however there is an option to only loop the first 20, 40, or all 60 when the program is in playback mode.

For each time slot in **beatArray** there are 20 note slots. Each note slot is a Boolean, false representing no note to be played and true representing a note generation event. The y-index of a note slot coincides with that note slot's pitch. This representation of note events to be generated over time allows the user to easily paint notes in the user interface, while allowing the program to loop through the **beatArray** array one time slot at a time. Any number of tone generation events may take place for any given time slot, and all tone events to be generated for a similar time slot will be generated simultaneously.

A tone generation event begins by determining at what frequency the modulating waveform should be produced. This is done by multiplying the carrier frequency (*carrierFreq*) by the modulation ratio (*modRatio*) using the **ModFreq** function. The result is a frequency that ranges from $1/10^{th}$ the carrier frequency to double the carrier frequency. The ratio taken as the y input of the 2 dimensional mouse input plot found in the bottom left corner of the application.

Once the modulator frequency has been determined the **FreqMod** WavePlayer is created, using the specified modulator wave type (sine, square, or sawtooth). The WavePlayer accepts the **ModFreq** for the frequency parameter. This WavePlayer is a buffer filled with the wave type selected at a rate of **ModFreq** Hz.

Next, the **carrierMod** function maps the **FreqMod** buffer to the tone's original carrier frequency value. This function will allow the output WavePlayer to accept the altered **FreqMod** buffer as its frequency input, effectively allowing the output WavePlayer to have a variable frequency that changes over the duration of the tone.

This entire FM Synthesis process allows the original carrier waveform to be compressed or expanded dynamically as it is being generated, in accordance to the modulator waveform selected at a rate respective to the **ModRatio**.

The output WavePlayer is connected to a Gain object which amplifies the waveform before it is given to the AudioContext object which is responsible for outputting the waveform as audio to the operating system.

The 2 dimensional mouse input plot mentioned earlier also allows the carrier frequency (*carrierFreq*) to be finely tuned by altering the x-axis value of this plot. This allows for a pitch to be created that is outside the standard set defined by each note slot. The user may interact with the mouse plot in real time, listening to their painted melody looping while changing *carrierFreq* and *modRatio* parameters by clicking at different locations inside the mouse plot.

The carrier wave type and modulator wave type may also be altered in real time by clicking on a particular wave image displayed under “Carrier” or “Modulator”.

The design of this application was focused on allowing the user to play around with several parameters quickly and easily to maximize FM synthesis exploration productivity. When a pleasing result has been achieved the user may click the “OUTPUT” button to allow the program to play through one cycle of **beatArray**. In this case the Gain object gives the amplified audio buffer to both the AudioContext Object and a RecordToSample object (**rts**). **Rts** is responsible for saving all audio output into the buffer **sample**. Once all time slots have been iterated through by **mainClock** and all tones have been generated and added to the **sample** buffer, it is output as a .wav file for use in more sophisticated audio editing software.

User Manual

GETTING STARTED

- Begin by selecting a Carrier wave form and a Modulator wave form by clicking on the wave images in the center middle of the application window. Next, try clicking on boxes in note bar (the large grid) located at the top of the application window to paint notes. These notes represent tones to be played with tones occurring first on the left and last on the right.
- Test out your newly created melody by clicking the "PLAY" button found at the top right of the application window. The "STOP" button will pause playback and return the note cursor to the beginning of the note bar.
- When you have created something you like, click the "OUTPUT" button found in the bottom middle of the application window to save your creation to a .wav file. The file will be saved in the same location you are running the application from.

PLAYBACK CONTROL

- To loop through the note bar click the "PLAY" button found in the top-right corner of the application window.
- To stop loop playback of the note window click the "PAUSE" button found in the top-right corner of the application window, found under the "PLAY" button.

NOTE PLACEMENT

- The higher the note box is, the lower the pitch of the note generated. The horizontal index of the note box represents what time slot the note will be played at. The first time slot is the left-most column of the note bar. The column to the right of this is the second time slot to be played. This pattern continues until the final time slot (the right most column) is played, after which the program loops back to the first time slot.
- In order to place a note on the note bar simply click a note box in the note bar. The note box will be greyed out, showing the note will be created when playing through the note bar.
- To remove a note, click on the note box the note is in. The note box will no longer be greyed out.
- To clear all note boxes in the note bar click the "CLR" button found in the middle-right of the application window, under the "STOP" button.

NOTE BAR OPTIONS

- The size of the note bar may be adjusted in order to create short, medium, or long samples. Short samples have 20 time slots, medium samples have 40 time slots, and large samples have 60 times slots.
- To adjust the sample size click the “20”, “40”, or “60” buttons located on the left size of the application window, directly under the note bar.
- Smaller sample sizes allow for shorter loops to be played. This is recommended when first adjusting FM Synthesis parameters.

WAVEFORM SELECTION

- A Carrier waveform and a Modulator waveform may be selected at any time. To do so click on a wave image located in the bottom-middle of the application window. The three images on the left represent carrier wave forms while the three on the right represent modulator wave forms.

FM SYNTHESIS PARAMETER ADJUSTMENT

- The box in the bottom-left of the application window is the two dimensional mouse plot that allows the user to interact with the FM synthesis parameters *Carrier Frequency* and *Mod Ratio*.
- To change the FM synthesis parameters click on a position within the box.
- The *Mod Ratio* parameter is determined by the vertical value of the dot generated where the mouse is clicked inside the box.
- The *Carrier Frequency* parameter is determined by the horizontal value of the dot generated where the mouse is clicked inside the box.
- These parameters change in real time with no need to stop the note bar playback.

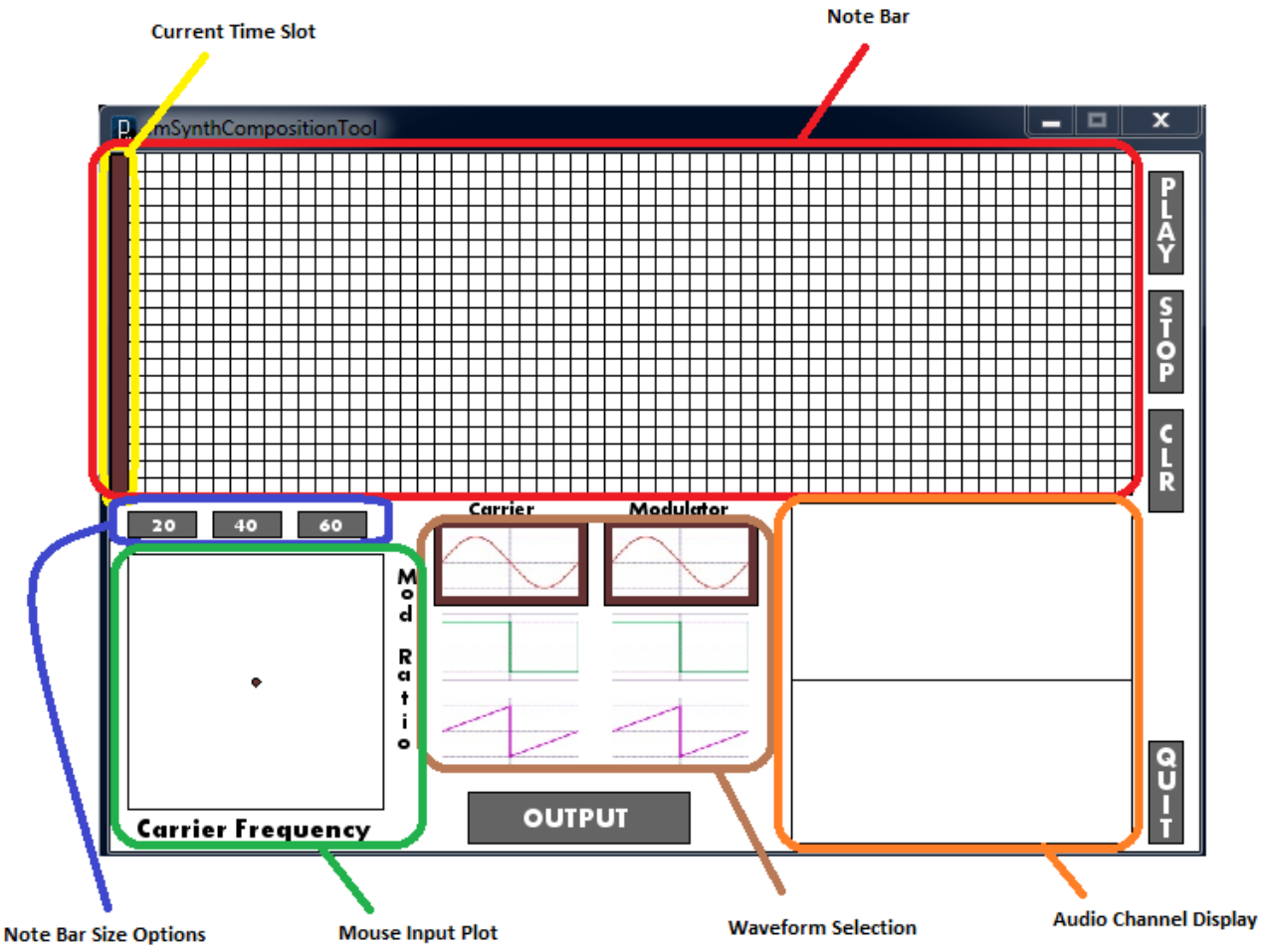
SAVING SAMPLES

- To save the sample created click the “OUTPUT” button located in the bottom-middle of the application window. Once clicked, no other input from the user will be allowed until

the note bar has been run through once entirely. After this the sample is saved and the user may once again interact with the application.

QUIT THE APPLICATION

- The “QUIT” button located in the bottom-right of the application window will allow the user to exit the application.
- The notes painted in the note bar will not be saved, so be sure to output the current sample before quitting if you want to save your work.



Bibliography

- http://processing.org/
  This site provided tutorials on how to use the Processing programing language.

- http://www.beadsproject.net/
  This site provided introductory tutorials to Processing with Beads as well as Javadocs for the Beads library.

- Lessons 3 and 10 from the Beads tutorials outlined the basics of FM synthesis, mouse interaction, and displaying the AudioContext buffer

- http://www.scribd.com/doc/66501315/34/Saving-Your-Sounds
  This website provided an explanation and example of how to save processed audio to a .wav file by exporting this information from a Gain class to a RecordToSample class and Sample class.

- http://en.wikipedia.org/wiki/Frequency_modulation
  This Wikipedia article provided a mathematic explanation of frequency modulation

- Inspiration for this project came from working with Steinberg's Cubase. Here a similar approach is taken for working with midi. The application I created could be implemented as a VST plugin for Cubase (it probably is already included in vanilla Cubase) however working with Processing allows for a lightweight, independent application.