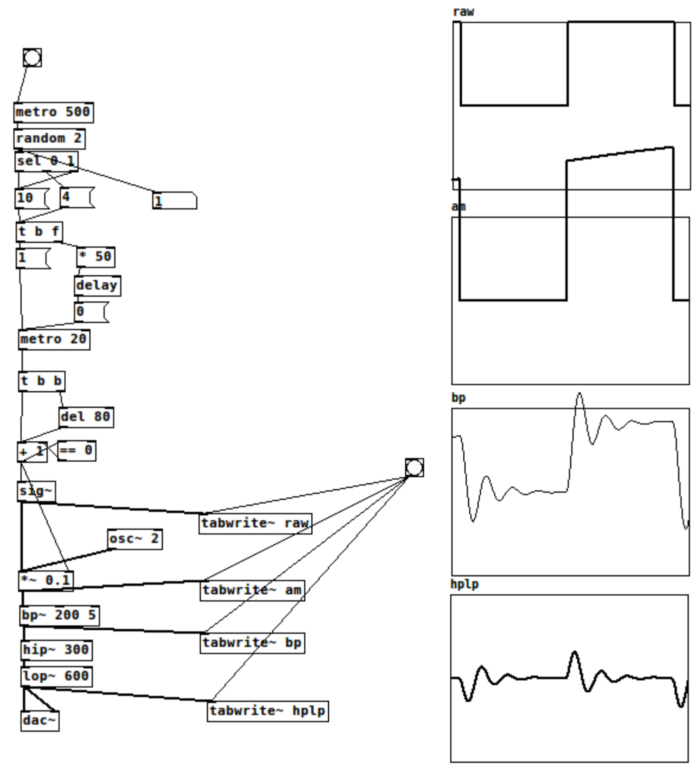
Nick Creel
Jim Mahoney
Sound Design
27 March 2017

Midterm Project: Randomly Generated Melody

For my midterm project, I created a patch that randomly generates a melody using three different oscillators. One produces a deep bass percussion, while the other two oscillators produce the melody and harmony tones in different octaves of the same musical scale. The project does not require any interaction from a user to run; it simply needs to be opened using Puredata and will run itself.

For the percussion part of the project, I modified one of the practicals from Andy Farnell's *Sound Design*. The original patch was meant to emulate a rotary dialer phone and the pulses that it sent depending on which number a person dialed. For this project, I included two options for the amount of pulses, one long and one short. The basic components of the percussion include a counter, various delays, metro objects, a random number generator, and various filters. The original design of the rotary dialer produced a square shaped audio signal that regularly alternated between an amplitude of 0 and 1, creating sharp pulses. The counter alternates between 1 and 0, creating the square peaks and flat troughs of the sound wave. The first delay, attached to the t b f object, controls the amplitude of the square peaks, while the



Percussion signal at different stages of modification second delay controls the frequency of the square peaks. Rather than having the pulses change sharply between 0 and 1, smooth transitions were created by adding the original signal to a low frequency oscillator. Various filters were created to shape the wave further. A bandpass filter was used to reduce clipping and round out the sound wave more, so that the resultant sound would be in a reasonable range of amplitude and sound more realistically like a bass drum synthesizer. High pass and low pass filters were used to limit the frequency of the sound wave; a high pass filter allows for frequencies above the set cutoff frequency, while a low pass filter allows for frequencies below the set cutoff frequency.

The synthesizer in the middle of the patch is a typical amplitude modulation synthesizer. A basic amplitude modulation synthesizer is formed of two oscillators that are multiplied with each other. For this particular synthesizer, two sawtooth synthesizers were multiplied. Because a sawtooth synthesizer changes amplitude from 0 to 1 in a linear fashion rather than forming a sine wave, the amplitude of the synthesizer changes sharply from 0 to 1 at the end of each oscillation. The synthesizer changes frequency every second, and the modulator has a frequency of 0.5 Hertz; there are two tones per oscillation.

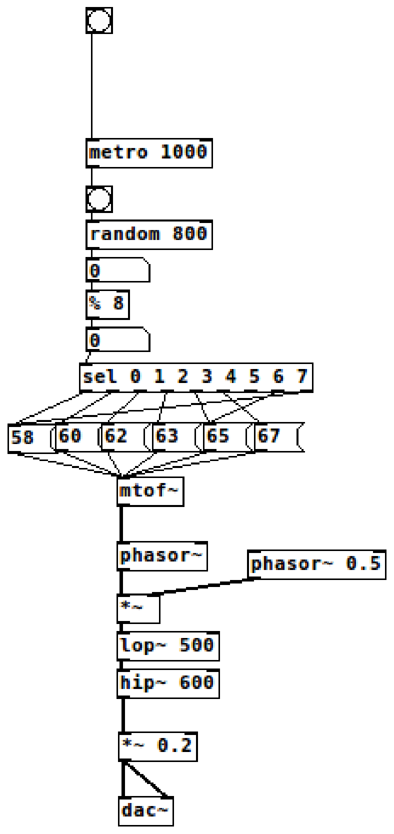


Illustration 1: The AM synth used in this patch

The synthesizer on the right of the patch is a frequency modulation synthesizer. Rather than simply multiplying both oscillators, one oscillator’s output is used as the frequency input for a second oscillator. In order to have the second oscillator sound differently from the first, the FM index and carrier frequency must not equal 0. The FM index is a number that is multiplied by the output of the initial oscillator, and is the rate of frequency deviation. Frequency deviation and the modulator frequency are summed, and an offset, known as the carrier frequency, is added to the sum of frequency deviation and modulator frequency. The resulting value is the frequency of the second oscillator. Both oscillators are then multiplied, and the resulting oscillator wobbles in a range between two frequencies.

The main frequency of both oscillators is chosen using a random number generator and the select object. Although the select object is usually used to compare values, here it is used to randomly select multiple values using the results of mod division. Because only one value is provided, there is nothing to compare, so the select object sends a bang message to whichever outlet matches the input number. I got the idea for this particular structure from one of Lysha's homework assignments that was shared on February 8th.

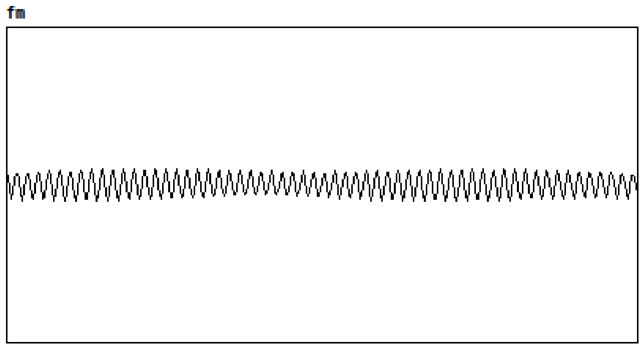


Illustration 2: Instead of maintaining a constant frequency, the resulting sound wave produced by frequency modulation wobbles between two frequency values

Bibliography

Farnell, Andy. (2010). *Designing Sound.* Cambridge, MA: The MIT Press.

Kreidler, Johannes. (2013). *Loadbang: Programming Electronic Music in Pure Data.* Hofheim en Tanus, Germany: Wolke Verlag