MU620A Music Systems Programming II

Jan Reinberg

Final Project

I built a Max object called tosc~ (table oscillator) is based on the table look-up oscillator and a Fourier series wavetable. Takes in harmonic values and iterates through them to create some wave-shaping effects. Largely inspired by concepts explored in class as well as well as exploring other concepts from outside sources (i.e. Audio Programmer's DSP Corner series by Rachel Locke). The code is written in C as in relies on the max-sdk MaxMSP API to build. Initially, I had written the core parts of the code in C++, but I then realized the max-sdk was more accessible and reliable as far as supporting documentation than the min-devkit. I ended up having to rewrite the code in C and re-orient it for implementation using the simplemsp~.c template provided by the Max Software Development kit.

The main code begins with oscillator structs and functions for the synth in the first portion of the code, were inspired by the class lectures and chapters in the text, especially the table-look up oscillator. Next you have the method prototypes prior to the main function (ext_main()) where the methods and objects are initialized. Then we initialize default parameters for the synth and the wave table processing in simplemsp_new(). The wave table in the simplemsp_new() method was inspired by a mix of the chapter on Spectral Processing and Rachel Locke's lecture on the Fourier Series:

The calculation is largely: wave += (sin(2*PI*Freq*Harmonic*time)/Harmonic. (Locke, 2022)

Time was calculated with the loop index over sample rate. Frequency dereferenced from the oscillator and then harmonics as well from the second inlet. This is scaled with the amplitude

(Lazzarini, n.d.)
(Lazzarini, n.d.) (Locke, 2022)

parameter. (In my C++ model, I also had a version where one could skip odd or even indexes in going through harmonics loop to provide sinusoidal or saw tooth variations to the wave form, but implementing it here was not working, so I left it out).

Next we have methods for handling freeing memory, and inputs to the inlets. The simplemsp_assist() handles messaging when a mouse hovers over the inlets (i.e. if the mouse is over inlet 1, it's say "Frequency"). The simplemsp_float() and int() methods handle data types going into the inlets and adjusting the type accordingly depending on whether it's an int or float applied to the frequency. The simplemsp_dsp64(), we not altered too much from the template as it registers the object for use in Max, alongside getting the sample rate and setting it, when this happens. Simplemsp_perform64() is where the signal routing occurs between the inlets and outlets. It receives a steady stream frequency and harmonic data from the inlets into the arrays where they are set to the proper designated dereferenced from oscillator parameters. Then the output is scaled (I had to scale it a few times cause of how harsh the frequencies can be). I also put in a script to make sure that no clipping occurs beyond 1.0 or -1.0.

The object becomes a .mxo file which was builts in Xcode for Max in MacOS. The .mxo file is placed in the externals msp folder of the contents folder for Max. Then when in the Max patcher, Max will automatically register it via the dsp64() method. The video provided gives a short demonstration of the basic character of

(Lazzarini, n.d.)
(Lazzarini, n.d.) (Locke, 2022)

Bibliography

Cyclin' 74 Max API Documentation
https://sdk.cdn.cycling74.com/max-sdk-7.1.0/chapter_msp_anatomy.html

Lazzarini, V., n.d. Chapter 1: Oscillators. In: s.l.:s.n., pp. 1 - 11.
Lazzarini, V., n.d. Chapter 7 Notes: Spectral Processing. In: s.l.:s.n., pp. 223-247.
Locke, R., 2022. *The Audio Programmer: DSP Corner: Introduction to Fourier Series | Rachel Locke (Dynamic Cast)*. [Online]
Available at: https://youtu.be/VSakODUEw_Y

(Lazzarini, n.d.)
(Lazzarini, n.d.) (Locke, 2022)