**MUS/COS 314 - Fall 2018 – Programming Assignment 3**

Due 10/9 (11:59PM)

Work with your partner!

Some useful documentation you should take a look at:

<http://chuck.cs.princeton.edu/doc/language/array.html>

**1) Create a ChucK patch that, when launched, creates an algorithmically generated piece that is different every time. The piece should be musically inspired by the music of Ryoji Ikeda. For examples of Ikeda’s music, listen to Formula, from the listening assignment, or any other piece by him you find.**

**(These will be played in class)**

The Ikeda inspiration means try to limit your materials to the following:

* sine waves
* filtered noise (LPF, BPF, HPF, multiple BPFs on the same sound)
* various ADSR envelopes
* interesting rhythmic patterns
* reverb (play a bit with the .mix parameter!)
* maybe a quiet “string section” like chordal sound
* you can use samples if you want to, in a way similar to how he uses them (like the strings at 7:15 in this piece: <https://www.youtube.com/watch?v=F5hhFMSAuf4>)

The challenge here is to make something musically interesting with such limited materials.

The mini-piece should be roughly around 60-120 seconds long, and then shut itself off.

**2) Even though this isn’t necessarily how Ryoji Ikeda does his pitches, I’d like to ask that you calculate the relationships between the frequencies of your sine waves directly (rather than thinking in MIDI notes like 60=middle C).**

You can get different pitch relationships by multiplying the frequency of one pitch by a ratio and using that frequency for another pitch. A simple example is that multiplying a frequency by 2/1 will bring the pitch an octave higher. Multiplying frequency by 5/4 (1.25) will produce a pitch a major third higher. This page explains the concept (called just intonation) well: <http://www.kylegann.com/tuning.html>

An example:

SinOsc s => dac;

261.16 => float myPrimaryFreq; // this is the frequency of middle C

.35::second => dur myBeat;

// written out the long way just to be clear

s.freq(myPrimaryFreq \* 1.0);

myBeat => now;

s.freq(myPrimaryFreq \* (5.0/4.0)); // 5/4 is a major third

myBeat => now;

s.freq(myPrimaryFreq \* (3.0/2.0)); // 3/2 is a perfect fifth

myBeat => now;

s.freq(myPrimaryFreq \* (7.0/4.0)); // 7/4 is a harmonic series minor seventh

myBeat => now;

s.freq(myPrimaryFreq \* (9.0/4.0)); // 9/4 is a major ninth

myBeat => now;

// or, written with an array to be more adjustable

// and with 11/4, 3/1 and 13/4 added to the pitch set.

[1.0, 1.25, 1.5, 1.75, 2.25, 2.75, 3.0, 3.25] @=> float myRatios[];

while(1)

{

for (0 => int i; i < 7; i++)

{

// read from the myRatios array

s.freq(myPrimaryFreq \* myRatios[Math.random2(0, 7)]);

// randomly switch between .25 and .5 of the beat length

myBeat \* (Math.random2(1, 2) \* .25) => now;

}

}

**3) After you’ve done the above, modify your patch to accept an argument from the**

**command line (in the terminal, or miniaudicle).**

Formatting:

While the flexibility of chuck means you can put functions and variables anywhere, let’s agree to code structure of:

-comment section at the top, with your names, the date, and a very brief description, plus what your argument does

-define the synth chain(s) (unless they happen within a function)

-global variables

-your functions

-the main part of the patch

Make sure you comment your code generously, explaining what is going on!

WHAT TO TURN IN:

- your one chuck file

- include any additional files if there are any (audio samples, etc)

- Just turn in one copy of the assignment for every pair of students. Make sure you put both your names at the top of the file in comments

HOW TO TURN IT IN

If it is just a chuck file, submit that. If you have any additional files, then zip everything together into one archive and submit that.