ChucK Assignment 1

29 March 2012

Welcome to ChucK! ChucK is a powerful audio language that allows you to control sound synthesis in a very precise manner. In this first lesson you will learn to use oscillators and control their properties directly.

1. If you have not already done so, go to `http://chuck.cs.princeton.edu/release/` and download the **midiAudicle** for your operating system. (Mac 10.7 "Lion" users should download this: `https://ccrma.stanford.edu/~spencer/chuck/miniAudicle-0.2.1-beta-1.tgz`)

   This is the built-in editor for ChucK called the miniAudicle. Open the program and choose **Start Virtual Machine** to launch ChucK. Then type your code into the blank document and click the + button to run it.

2. Let's begin with a simple Sine Oscillator.

   ```
   SinOsc myOsc => dac;
   ```

   - *ChucK is case-sensitive, so be sure to observe the correct capitalization.*
   - *If you forget to end a line with a semicolon you will get an error message. This is a very common mistake.*

3. This creates a Sine Oscillator called **myOsc** and connects it to the **dac**, the *digital-to-analog convertor*. The dac is the only way to get sound output. Let's start by giving it a frequency:

   ```
   440 => myOsc.freq;
   ```

4. The finally necessary element is **time.** Time is ChucK's real specialty; it lets you deal with precise time down to 1 audio sample, and up to 1 week!

   In ChucK you must explicitly specify how long to generate sound. Let's make it last 2 seconds:

   ```
   2::second => now; // <-- notice the 2 colons
   ```

5. Run this (press the + button), and you should get a fabulous sine oscillator for 2 seconds. (You might want to turn down your computer's volume a bit.)

   SinOsc is a built-in audio object; in computer music these are called Unit Generators or UGens (pronounced "You-Jens").[1]

   Think about a hardware oscillator box. It must have some controls on it, right? Probably a volume control, and certainly a frequency control. In ChucK, these controls are called "methods." The volume control is called **gain** and the frequency control is called **freq**. You can find them listed in the ChucK manual on page 133.

6. If you want to change the value of one of the methods, you need to ChucK a value to it.

```
0.5 => myOsc.gain; // half volume
700 => myOsc.freq; // 700 Hz
500::ms => now; // 500 milliseconds = 0.5 seconds
1400 => myOsc.freq; // one octave higher
800::ms => now;
```

7. It's not very practical to always think in frequency. If you want notes from the equal tempered scale, you must convert them using another method, **Std.mtof**. This stands for MIDI-to-frequency. MIDI key 60 is middle C, 61 is C#, etc.[2]

```
65 => Std.mtof => myOsc.freq; // F above middle C
900::ms => now;
```

8. SinOsc is not the only kind of oscillator available in ChucK. Pages 132-135 of the ChucK Manual list several different oscillators, each with its own unique sound, and sometimes with different control methods. Try some different oscillators by replacing the SinOsc at the top of this code with SawOsc, SqrOsc, or any of the others. Don't worry if things don't work or if you get an error. Just experiment with changing this sample code.

Assignment 1: Experiment with Oscillators

Using the code above as a model, experiment with different oscillators and their control methods. Create a melody with at least 10 pitches. Save your ChucK code as a .ck or .txt file and upload it to the assignment page on Blackboard by this Tuesday, April 3.

[1] Although Max and ChucK look very different, they have a common ancestry, Max Mathew's Music III, and "under the hood" they function quite similarly. Both ChucK and Max create sound by connecting UGens together, ending with the dac. Max's SinOsc object is called cycle~ and it functions in a very similar way as ChucK's.

[2] mtof is also a Max object that does the same thing, converting MIDI key numbers to frequency.