COS/MUS 314

**Assignment 4: Drum Machine 2**
Assignment due 2 March 2009

0. **Reading**
a. Functions in ChucK
http://chuck.cs.princeton.edu/doc/language/func.html
b. Envelopes in ChucK
Handout:
http://www.cs.princeton.edu/~fiebrink/314/2009/Envelopes_handout.pdf
Run the ADSR and Envelope examples at
http://chuck.cs.princeton.edu/doc/examples/
Also read ADSR and Envelope specifications in the Programming Guide:
http://chuck.cs.princeton.edu/doc/program/ugen.html
Be sure you understand how these work, and how ADSR is different from Envelope.
c. Additive Synthesis
Tutorial:
http://soundlab.cs.princeton.edu/learning/tutorials/SoundVoice/pcm1.htm
(read through at least to end of Additive Synthesis section)
d. ChucK UGens:
Read GenX documentation at
http://chuck.cs.princeton.edu/doc/program/ugen_full.html#GenX
And http://chuck.cs.princeton.edu/doc/examples/special/readme-GenX.ck
See also examples for specific GenX objects, like Gen10

**1. Writing functions**
Encapsulate any drum machine ChucK file from your last assignment in a function called drumMachine. Expose control over musical aspects of the loop (tempo, instrumentation, … ?) via parameters of the function.

For example, your function might start like

```
fun void drumMachine(dur quarterNoteDur, int
instrumentChoice, float gain) {
    // ... do your function here
}
```

Play with calling your function with different parameters from the main section of the code.

**2. Understanding ADSR envelopes:**
Look at the code below. First, describe the characteristics of the envelope that is

being created using the a.set function in the second line. Second, discuss what is wrong with the code after line 3.

```
SinOsc s => ADSR a => dac;
(.001, 0.01, 0.9, 1.0) => a.set;
a.keyOn();
1::second => now;
a.keyOff();
```

**3. Additive synthesis, GenXs, envelopes, and channels**
Write another function called makeNote that synthesizes **a single note** using one of the GenX objects (e.g., Gen10) and an ADSR or Envelope object. The parameters of this function should minimally include pitch, duration, and channel, but you should add at least one other parameter for higher-level properties of the sound – perhaps "attack sharpness" or "brightness" (related to the parameters of the GenX and/or envelope used). Choose the GenX object, envelope, and their parameters to make the sound interesting!

About channels:
Your computer has 2 channels for built-in audio (left and right). The hemi has 6. You can find out how many channels are currently available by calling dac.channels(). You can chuck a UGen to a single channel of the dac using, e.g.,

```
SinOsc s => dac.chan(2); // to chuck to channel 2. Channels start at #0!
```

You can also refer to the channels of the dac by reference (using @=>). Check out clix.ck to see how that code "spins" the sound around the available channels.

**4. Putting it all together into a new drum machine**
Call your makeNote function from existing or new drum machine code so that you like how it sounds. (Feel free to keep using SndBuf, UGens, etc. in conjunction with your new function.) Find a good way to control the parameters of makeNote over time.

**5. Object scope**
Consider the two following code examples. Why might you choose to write A instead of B? B instead of A?

| A | B |
|---|---|
| `SinOsc s => dac;`<br>`while (true) {`<br>`  play();`<br>`}` | `while (true) {`<br>`  play();`<br>`}`<br>`fun void play() {` |

```
fun void play() {
     Std.rand2(100,1000)=> s.freq;
     .5::second => now;
}
```

```
     SinOsc s => dac;
     Std.rand2(100,1000)=> s.freq;
     .5::second => now;
     s =< dac;
}
```

**What to hand in:**

• Your ChucK code for Question 4 (will also contain code you wrote for Questions 1 and 3). Make sure you include all sound files you use in the drum machine.

• Your written answers to Questions 2 and 5.

**Submit using blackboard, with all your code & written work in a .zip file. Please put your written work in a .txt, .rtf, .pdf, or .doc file within the .zip, and don't forget to comment your code!**