COS/MUS 314

Classes and Filters

Assignment due 2 April 2008

0. Reading:

Classes and Objects:
http://chuck.cs.princeton.edu/doc/language/class.html

Example: dinky.ck and try.ck from
http://chuck.cs.princeton.edu/doc/examples/index.html#class

Extending a class:
http://chuck.cs.princeton.edu/doc/examples/event/event_extend.ck

Let us know if you would like to check out a PLOrktop to assist you in testing your code for Question 1.

1. Download the skeleton files. Read the README. Fill in the required gaps in the code so that it runs on at least two machines (and makes sound!). Modify it by adding functionality to the Player, and possibly adding parameters to the play() function (this will require some editing of switchboard.ck). Also modify the [machine_name]_go.ck files so that you like how it sounds!

2. Describe how you might use inheritance and polymorphism to make this code more usable or interesting. (2-point bonus: Implement it!)

3. Pick two ChucK filters (that extend FilterBasic). Write a ChucK file that uses these filters to filter some sound input (from the adc, a sample, a UGen, your choice). Describe what is happening when you change the filter parameters for each filter, and how and why this affects the sound that you hear.

4. PRC **2-point bonus** question on filtering:
Pick one (or both) of these two Fun Filter Frolics:

A) Math/ChucK NERD:
Implement a resonant (two pole) filter directly in ChucK (use a 1.0::samp=>now update). The "difference equation" for a two-pole filter is:

y(n) = b0*x(n)-a1*y(n-1)-a2*y(n-2)

Compare performance to the built-in ChucK TwoPole UG. Make sure they sound the same (We suggest using Noise as an input). Comment.

Note: Any DSP operation (including those of existing unit generators) can be implemented in native ChucK. The power is in being able to implement something that ChucK doesn't provide a unit generator for. For example, to hear an impulse each time an input mic/line signal is greater than 0.99, we could:

```
adc => blackhole;
Impulse i => dac;

while (1)   {
    1 :: samp => now;
    if (adc.last() > 0.99) 1.0 => i.next;
}
```

This example is of questionable value, but it lets you know how one might implement any function at the sample rate in ChucK.

B) Composer NERD:
Listen to Richard Karpen's "Exchange" http://www.cs.princeton.edu/courses/archive/spr07/cos325/music/ExchangeRichardKarpen1987.mp3 This piece uses resonant filters to create a tape accompaniment for a flute, controlling the filter resonance right up to and beyond stability (radius = 1+epsilon). Do something interesting along these lines, but explain what you're doing.

What to hand in:

- Your code for question 1 (see README on what to submit).
- Your written response (and possibly code) for question 2.
- Your code and written response for question 3.
- Optionally your response/code for question 4.

Be ready to demo your code for Questions 1 and 3 in class.