COS/MUS 314

Twinkle++

Lab assignment due 2/21, 11:59pm

In this assignment, you'll be working with your lab partner. Take a look at http://en.wikipedia.org/wiki/Pair_programming for a description of pair programming. The main idea here is that any given time, one person will be typing while the other helps and watches for mistakes, and you'll switch between these roles at least for every coding question. You'll hand in one submission and receive the same grade.

0. **Reading**
   a. Types, values, and variables
   http://chuck.cs.princeton.edu/doc/language/type.html
   b. Unit Generators in ChucK
   http://chuck.cs.princeton.edu/doc/language/ugen.html
   c. Manipulating Time in ChucK
   http://chuck.cs.princeton.edu/doc/language/time.html
   d. She's a witch!
   http://www.youtube.com/watch?v=zrzMhU_4m-g

1. **Twinkle STK (3 points)**
Spice up one of your twinkle codes from Assignment 1 by replacing the SinOsc with an STK instrument. See the ChucK UGen specification at http://chuck.cs.princeton.edu/doc/program/ugen.html to see what parameters are available for the different instruments, and don't forget to use the noteOn and noteOff functions. For extra help, look at the sample code stk_examples.ck.

Note on inheritance
Many of the UGen (unit generator) classes **extend** other UGens. For example, all STK instrument classes (such as `Bowed`) extend the `StkInstrument` class (you'll see this explicitly stated in the documentation for each class, for example http://chuck.cs.princeton.edu/doc/program/ugen_full.html#Bowed says "extends StkInstrument"). This means that all `Bowed` objects are *also* `StkInstrument` objects, and they *inherit* all of the parameters and functions of `StkInstrument`**,** like `noteOn()` and `freq()`. `StkInstrument` is therefore called a "parent" or a "superclass" of `Bowed`, and `Bowed` is a "child" or "subclass" of `StkInstrument`. So, if you want to know all the parameters and functions available to a ChucK object, be sure to look at the documentation for that object *and* its parent (and its parent's parent, and so on…). Unlike some other languages, there is no multiple inheritance allowed in ChucK, so each class may only have a single parent.

2. Logic exercise (2 points)
Download the skeleton code witch.ck.

a. Once you complete this code, it will print out whether someone is a witch, given a gender and weight. Replace the if-block at line 13 with something a bit more complicated, to encode the following logic:

> If something weighs the same as the duck, it floats.
> If something floats, it's made of wood.
> If it's made of wood, it burns.
> If it burns and it is female, it is a witch.

Don't add any more variables, but use all the variables defined in the skeleton. Try changing the initial values of isFemale, weight, and weightOfDuck to verify that the code does what you expect.

b. Add an appropriate sound effect according to whether the person is found to be a witch. You can use a UGen we've already introduced, like SinOsc or an StkInstrument, or you can live dangerously and use a SndBuf object to play back a .wav or .aiff file. Your choice.

See http://chuck.cs.princeton.edu/doc/examples/basic/sndbuf.ck for an example of using SndBuf; in order for this to work in miniAudicle, you'll have to initialize the filename variable to a filename that's relative to your miniAudicle "current directory", which is set in the miniAudicle Preferences menu, under Miscellaneous. For example, if my miniAudicle working directory is "/Users/rebecca/assignment2/", the code

```
"mySound.wav" => string filename;
```

indicates that I'm going to be using the file /Users/rebecca/assignment2/mySound.wav. You can play a file once just by connecting a SndBuf to the dac, loading the file using the .read function, and advancing time. If you use a SndBuf, be sure to include your sound file(s) in your submission.

3. Interactive expressivity (5 points)
(Don't forget to switch programmers!)

a. Surround your twinkle code from question 1 with a while-loop so it never ends. (Or, if you're sick of Twinkle, make a different melody or piece of musical code.)

b. Next, add expressive, real-time control to this "composition" by using one or more of following classes:

TrackpadFeatureExtractor (314_TrackpadFeatureExtractor.ck) (**You may have to change mouseDevice to 1 on line 25, especially for Lion)
MotionFeatureExtractor (314_MotionFeatureExtractor.ck)
JoystickFeatureExtractor (314_JoystickFeatureExtractor.ck)
BreathFeatureExtractor (314_BreathEnvelopeExtractor.ck)

Objects of these types have functions that you can call to get the current trackpad X and Y positions, the current X/Y/Z accelerometer positions, the current joystick configuration (from a Logitech Extreme 3D pro joystick -- ask us if you want to check one out from our stash), or an estimate of the current breath amplitude of someone blowing into the microphone. The comments at the beginning of each file describe the functions available and give you example code to try out.

Notice that, unlike SinOsc or Bowed or other built-in UGens, these classes aren't built into ChucK. Instead, they're defined *in* ChucK .ck files. In order to use an object of a *user-defined* type, you'll need to add the .ck file containing the class definition to the virtual machine (VM) *before* adding any code that references the class. For example, start miniAudicle, add 314_MotionFeatureExtractork.ck to the VM, then add motion_mandolin.ck. This example file declares a variable of type MotionFeatureExtractor, calls its setup and extract functions, then uses the motion sensor values in a simple way to change the pitch of a Mandolin.

For Question 3, pick one or more aspects of your twinkle (or other) piece to control expressively, using one or more of the input extractor classes above. For example, you could change the tonality according to the trackpad position and the tempo according to the tilt. It doesn't have to be complicated, but it should be creative and expressively controllable by a human performer.

What to hand in:

- ChucK code for your Twinkle, Twinkle masterpiece for Question 1.
- ChucK code for your witch hunt in Question 2, along with any sound files you used, if you used a SndBuf.
- ChucK code for your interactive masterpiece in Question 3.
- A brief written description of how to expressively control your composition in Question 3.
- Be prepared to demo Question 3 in the next precept!

How to hand it in:
Submit just one file per group. Submit via Blackboard, like you did in assignment 1. Also, **in the comments box, list your partner(s),** so they can get credit, too.