COS/MUS 314

Assignment 3: Envelopes, Functions, and Sporking Fun!

Assignment due 29 February 2012, 11:59 pm

This assignment is done **individually**. However, feel free to discuss it with your lab partners or other classmates.

Reading:

Handout on envelopes:
http://www.cs.princeton.edu/courses/archive/spring12/cos314/handouts/Envelopes101.pdf
Handouts on functions & concurrency:
http://www.cs.princeton.edu/~fiebrink/314/functions101.pdf
Handout on loops:
http://www.cs.princeton.edu/~fiebrink/314/2009/chuck_loops.pdf
Also see class examples from 2/14, 2/21 (on wiki)

Question 1: Envelope Practice (2 points)
Create two patches of your own that use envelopes. One should use an Envelope for amplitude control, and the other should use an envelope for something else. These can be very simple patches.

Optional: Read about ADSR envelopes and use an ADSR instead of an Envelope for amplitude control. See the ADSR example here: http://chuck.cs.princeton.edu/doc/examples/basic/adsr.ck.
Notice that A = attack duration, D = decay duration, S = sustain ***amplitude****, R = release time, which together describe an envelope that looks like this http://upload.wikimedia.org/wikipedia/commons/thumb/e/ea/ADSR_parameter.svg/800px-ADSR_parameter.svg.png.*

Example ADSR code:

```
SinOsc s => ADSR a => dac;
(.01::second, .01::second, .8, 1::second) => a.set;

a.keyOn();
1::second => now;
a.keyOff();
5::second => now;
```

Question 2: Loop and function practice (3 points)
a) Write a while-loop that plays one octave of a chromatic scale using an StkInstrument of your choice. That is, your function plays MIDI note n, n+1, n+2, … n+12, for some starting note n of your choice. Your code should play the scale just once.

b) Re-write that while-loop in question 2a) as an equivalent for-loop.

c) Put your for-loop from question 2b) inside a function called `playChromatic` with parameters `startNote` (of type int), `numNotes` (of type int), and `instrument` (of type StkInstrument). **Modify the code** within your loop so that it plays a chromatic scale beginning on `startNote` and ending after `numNotes` (e.g., if `numNotes` is 2 and `startNote` is 60, then your function should play C and then C#). **Also modify your code** so that the scale is played using whatever instrument object you pass in as the `instrument` argument.

> Hint: Because the `instrument` parameter is of type StkInstrument, which is the parent class of all STK instruments such as Mandolin, BlowBotl, Clarinet, etc., you can pass the function a Mandolin, a BlowBotl, a Clarinet, etc. as an argument when you actually call `playChromatic`. For example, the following valid code uses a much simpler function with an StkInstrument type parameter:
>
> ```
> Mandolin m => dac;
> Clarinet c => dac;
> playNote(m);
> playNote(c);
>
> fun void playNote(StkInstrument i) {
>     1 => i.noteOn;
>     1::second => now;
>     1 => i.noteOff;
>   }
> ```

To test your function, make sure that you can insert the following code snippet into your program, and it will play 5 notes of a chromatic scale starting on middle C:

```
Mandolin m => dac;
playChromatic(60, 5, m);
```

d) Add some code that calls `playChromatic` (perhaps using sporking and multiple instruments) to make some interesting, layered textures.

e) Optional (but strongly suggested if this assignment has been easy for you so far): Also change your function behavior to make the sound more interesting in some way, perhaps adding parameters, randomness, etc. Write code calling/sporking the function to take advantage of these new capabilities.

Question 3: Sound creation & exploration (5 points)
a) Create 3 chuck files that combine basic unit generators (not StkInstruments) in **interesting** ways to make sound. These can be fairly simple files that make basically one type of sound each.

The only requirements are:
1. At least one patch must use a UGen as an LFO (using .last() and sporking)
2. The patches must sound interesting/cool, and they should sound different from each other.

If you want some inspiration to get started, consider:
- Oscillators you might use: SinOsc, SawOsc, SqrOsc, TriOsc, Noise
- Envelope, ADSR might be fun, for amplitude, frequency, or other things
- LPF: A low-pass filter object. Play with Q and .freq, but be careful with your ears when experimenting!
- JCRev: a fun little reverb UGen
- Consider how randomness might add interesting effects
- Consider layering multiple sounds – for example, adding together sines of different frequencies in an additive-synthesis-like manner, or changing the blend between different sound sources over time

b) Embed one or more of your new sound patches from part a) in a ChucK file that controls your sounds (non-interactively) in a short composition (1–2 minutes long). It is up to you how to structure your code (e.g., encapsulating code from part a) in one or more functions, copying & pasting, etc.)

You will be graded on correctness and creativity!

What to turn in:
- One .ck file for Question 1
- Four .ck files for Question 2: one each for parts a) through d), plus another optional file for e) if you choose.
- Four .ck files for Question 4: Three for part a) and one for part b).
- Be sure to include comments in your files to help us run & grade them.