Homework 11-12

Music 208 Winter 2014

MUSC 208 Winter 2014
John Ellinger Carleton College

MUSIC 208 Homework 11-12

MUSIC 208 Homework 11-12

This homework will create a GUI (Graphics User Interface) to experiment with the five common waveforms you studied in Lab 11. Last year's project created the interface in MAUI (Mini Audicle User Interface) which is only provided on the Mac version of miniAudicle. This year's Homework11-12 will build the GUI using a Java based open source programming language called Processing which workson Mac, Windows, and Linux.

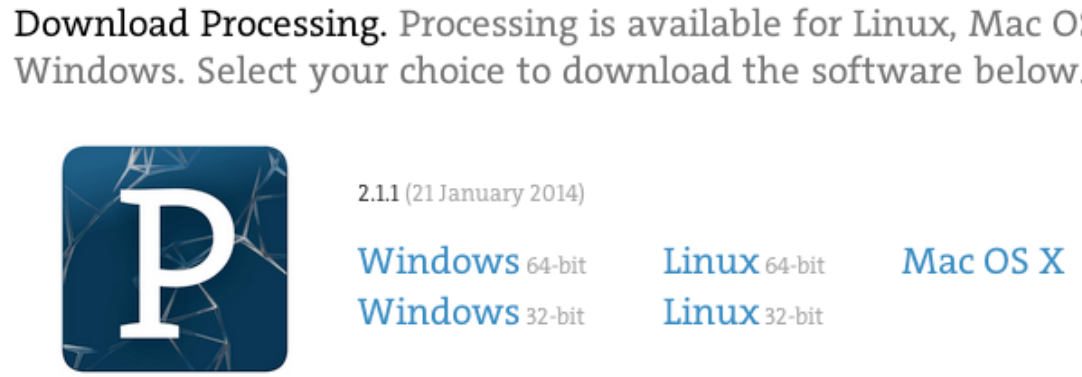
Homework 11-12 will require you to install the Processing software and two Processing code libraries on your laptop. You'll have to do independent reading and research to familiarize yourself with the Processing language, the controlP5 library that Processing uses to create GUI controls like buttons, checkboxes, and sliders, and the oscP5 library that sends information between Processing and miniAudicle. OSC stands for Open Sound Control. It's another open source program that is able to route information between two applications on the same computer or between applications on networked computers. Both ChucK and Processing understand OSC communication. I've done a lot of the work for you, but you'll have to complete it.

Setup

Download and install Processing 2.1.1 on your computer. I've tested on Mac OS X and both Windows 32. I do not have Windows 64.

http://processing.org/

http://processing.org/download/?processing

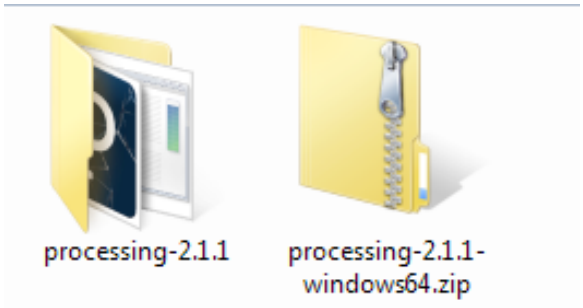


Mac

Whe you unzip the processing-2.1.1-windowsXX.zip file you downloaded you'll see an application called Processing. Copy the Processing application to the /Applications folder.

Win

Whe you unzip the processing-2.1.1-windowsXX.zip file you downloaded you'll see a folder called processing-2.1.1.



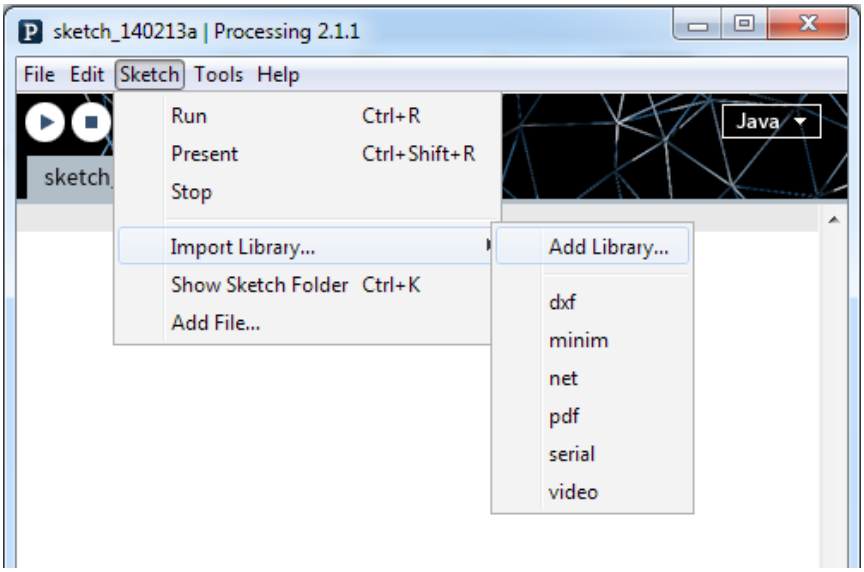
Move the processing-2.1.1 folder to C:\Program Files.

First Run

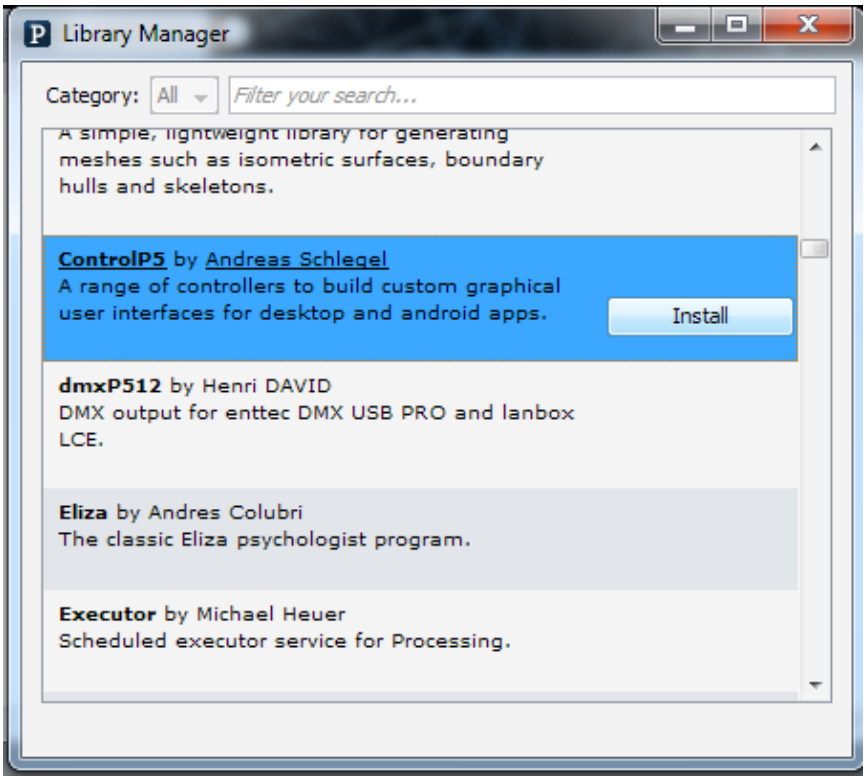
Double click the Processing application.

Add the controlP5 library to Processing

Choose Add Library from the Sketch menu.

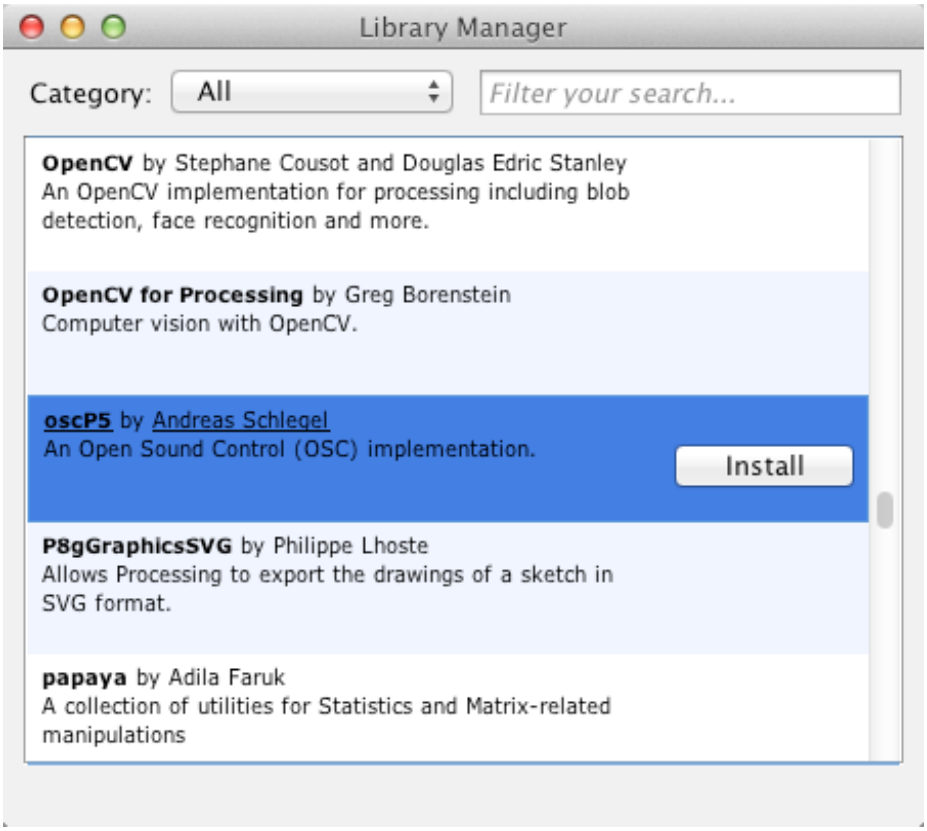


Select controlP5 and click the Install button



Add the oscP5 library to Processing

Follow the same steps to install oscP5.



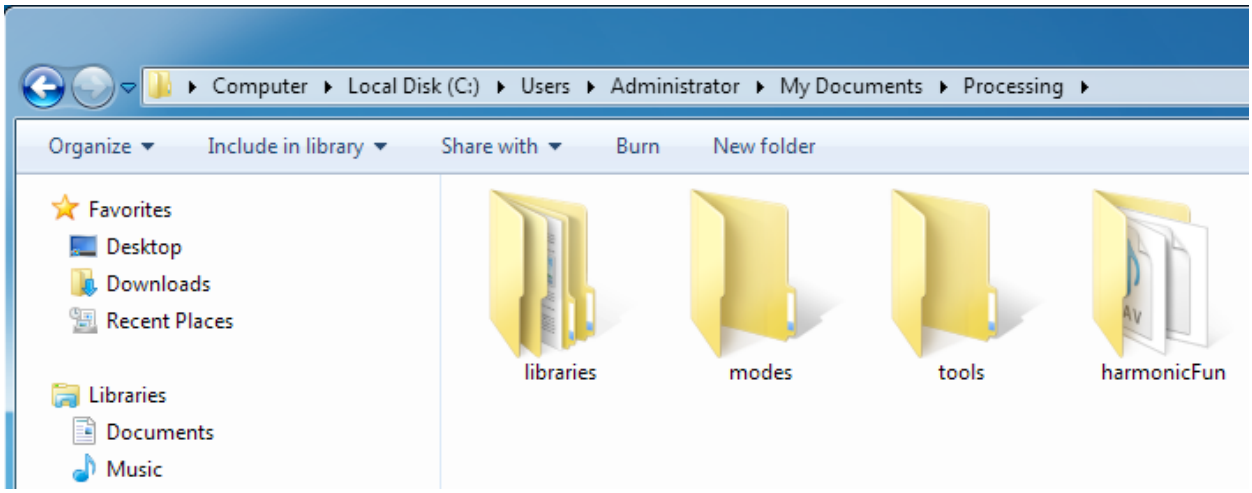
Quit Processing

The first time you run Processing it will create a new folder called Processing in the Documents folder in your home directory on both Mac and Windows. The Processing folder is where all code libraries used by the Processing application are located. It's where the controlP5 and oscP5 code libraries you just installed were placed and is also the folder where you should save any Processing projects and code you create.

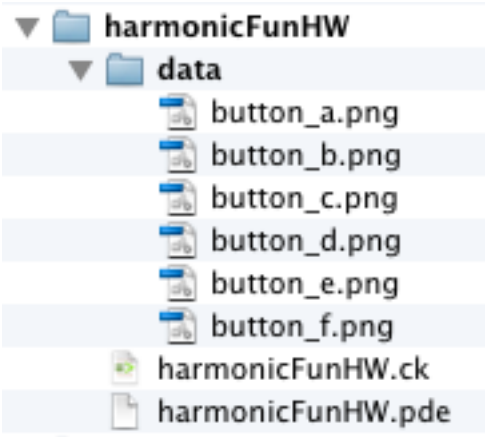
Download harmonicFunHW.zip

http://acad.carleton.edu/courses/musc208-00-w14/binInstallers/harmonicFunHW.zip

Unzip harmonicFunHW.zip and copy the harmonicFunHW folder into the Processing folder in your Documents folder.

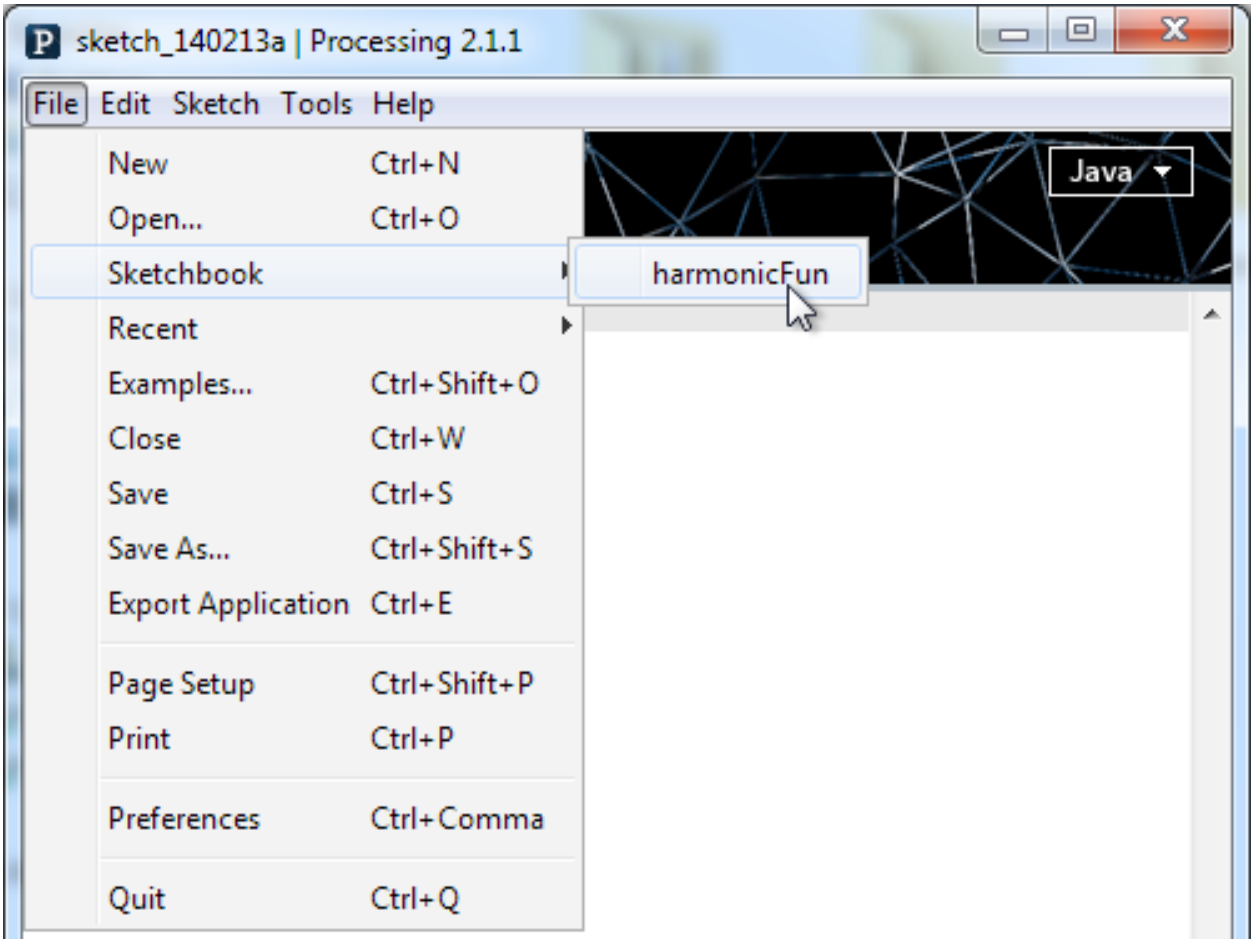


The harmonicFun folder should contain a data folder and two code files: harmonicFun.ck (the ChucK file) and harmonicFun.pde (the Processing file)

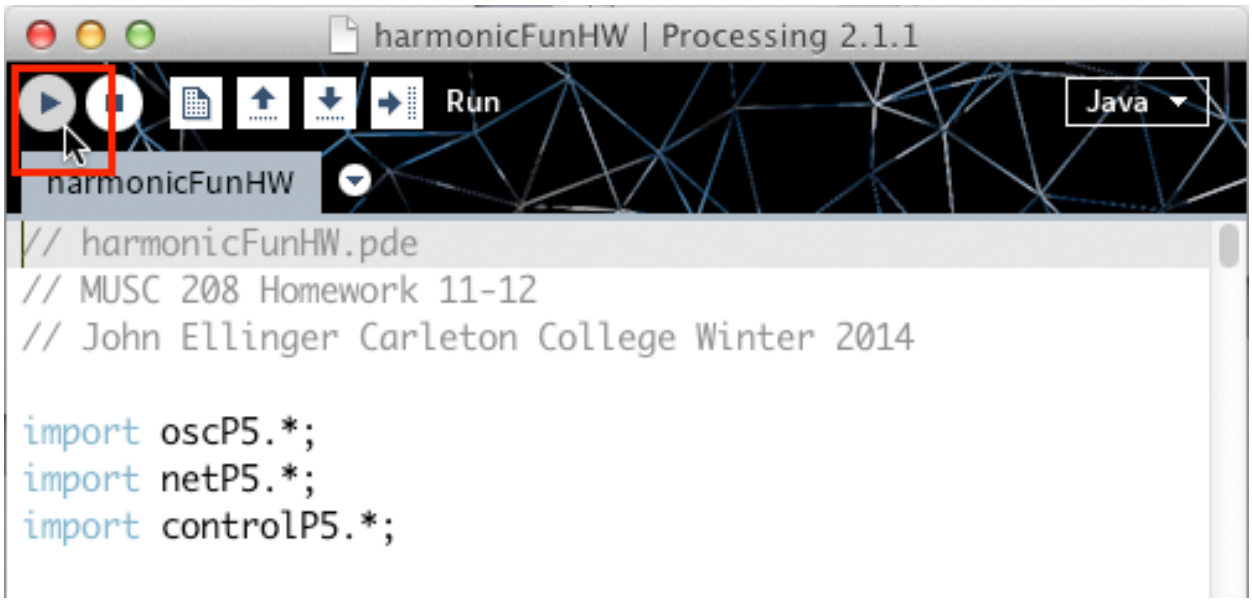


Run harmonicFun.pde in Processing

Open the Processing application and choose harmonicFun from the Sketchbook menu in the File menu.



When the harmonicFun window opens click the Run button to run the sketch.

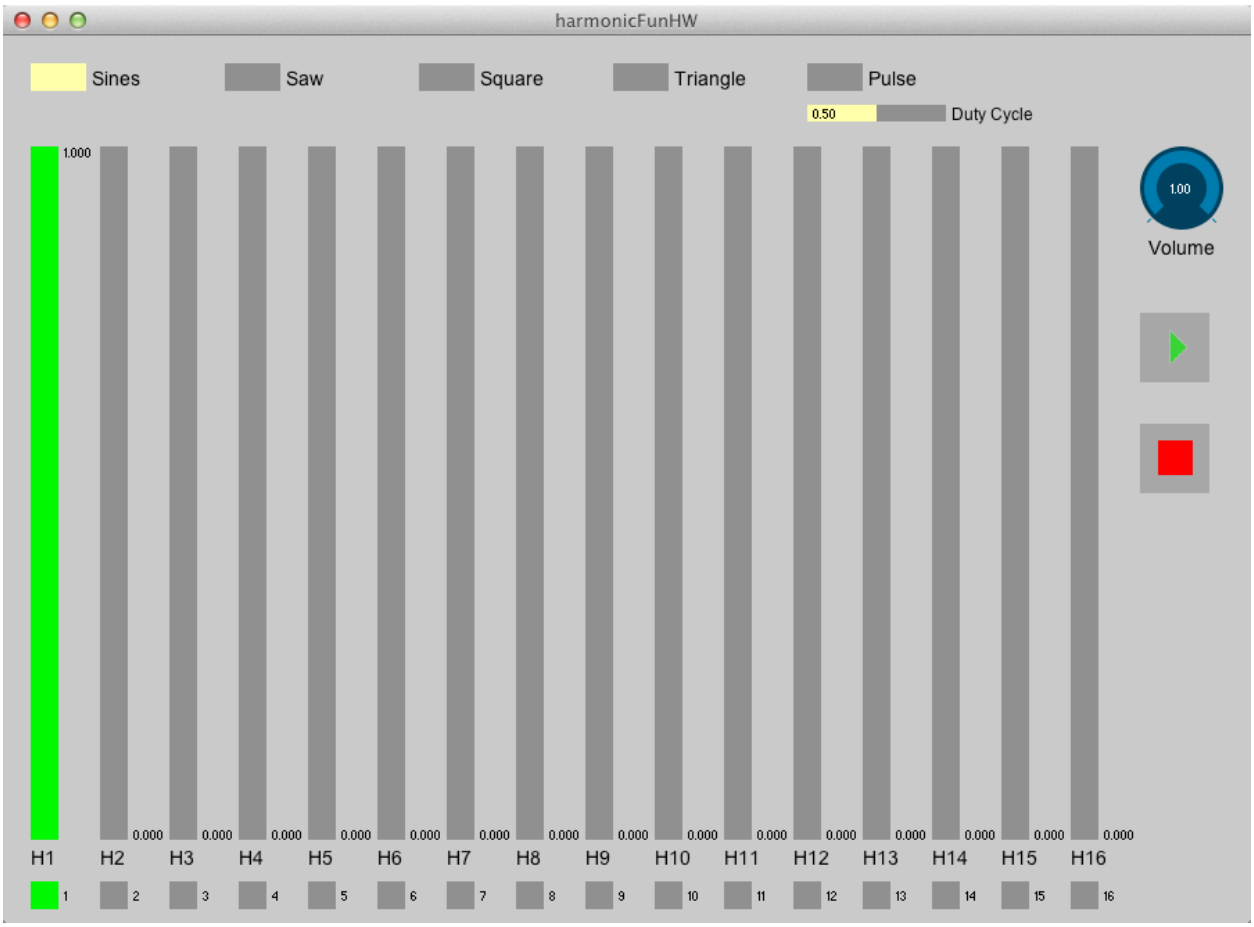


The harmonicFun GUI

The harmonicFunHW window will appear. There are several types of controls in the window that will be used to control sound generated by the harmonicFun.ck program you'll complete. At the top of the window are five buttons to select the wave type. Only one of these can be on at a time. Under the Pulse button is a horizontal slider that will set the Duty Cyle of the Pulse wave. A pulse wave only has two values plus 1.0 and minus 1.0. The duty cycle is the amount of time the waveform is positive.

At the right side of the window are a Volume knob control and two picture push buttons for play and stop.

The 16 harmonic amplitude control sliders control the gain of the 16 harmonics, H1 - H16. Under each slider is a checkbox on/off control.



Run harmonicFun.ck

You can run either from miniAudicle or the Terminal. "Listening for GUI controls" should appear in the console.

Experiment with the GUI controls. Every control you click or drag will send a message that will appear in ChucK's output window. Everything you need to create the sounds in are in the messages, you just have to write the code.

Your Assignment

Step 1

Get ChucK to produce sound when the Sines button is selected. The 16 sliders represent the 16 harmonics of a fundamental frequency (H1) hard coded to 220 Hz.

Movements of harmonic sliders must change the sound in real time. The harmonic checkboxes below each of the 16 sliders must turn that harmonic on and off in real time.

You'll need to account for amplitude clipping. Each of the 16 harmonics can have a maximum gain or amplitude of 1.0. The sum of all 16 gains must be less than or equal to 1.0. You'll need to introduce a scale factor variable to keep the total amplitude within the 0 to 1.0 range. The scale factor will change depending on the wave type and the sum of the harmonic gains in use.

The Volume knob is a secondary volume control that scales the sum of the harmonic gains from 0 to 1.0. It should function just like the volume knob on a radio.

The Stop button stops all sound. The Play button restarts sound using the current settings.

Step 2

Write ChucK code to implement the Saw, Square, and Triangle waves based on the GUI slider positions. If you've implemented the Step 1 well this should be

relatively easy. Remember to alternate plus and minus gains for the Triangle wave.

Step 3

Write the Pulse wave function in ChucK. The pulse wave must respond to the Duty Cycle slider.

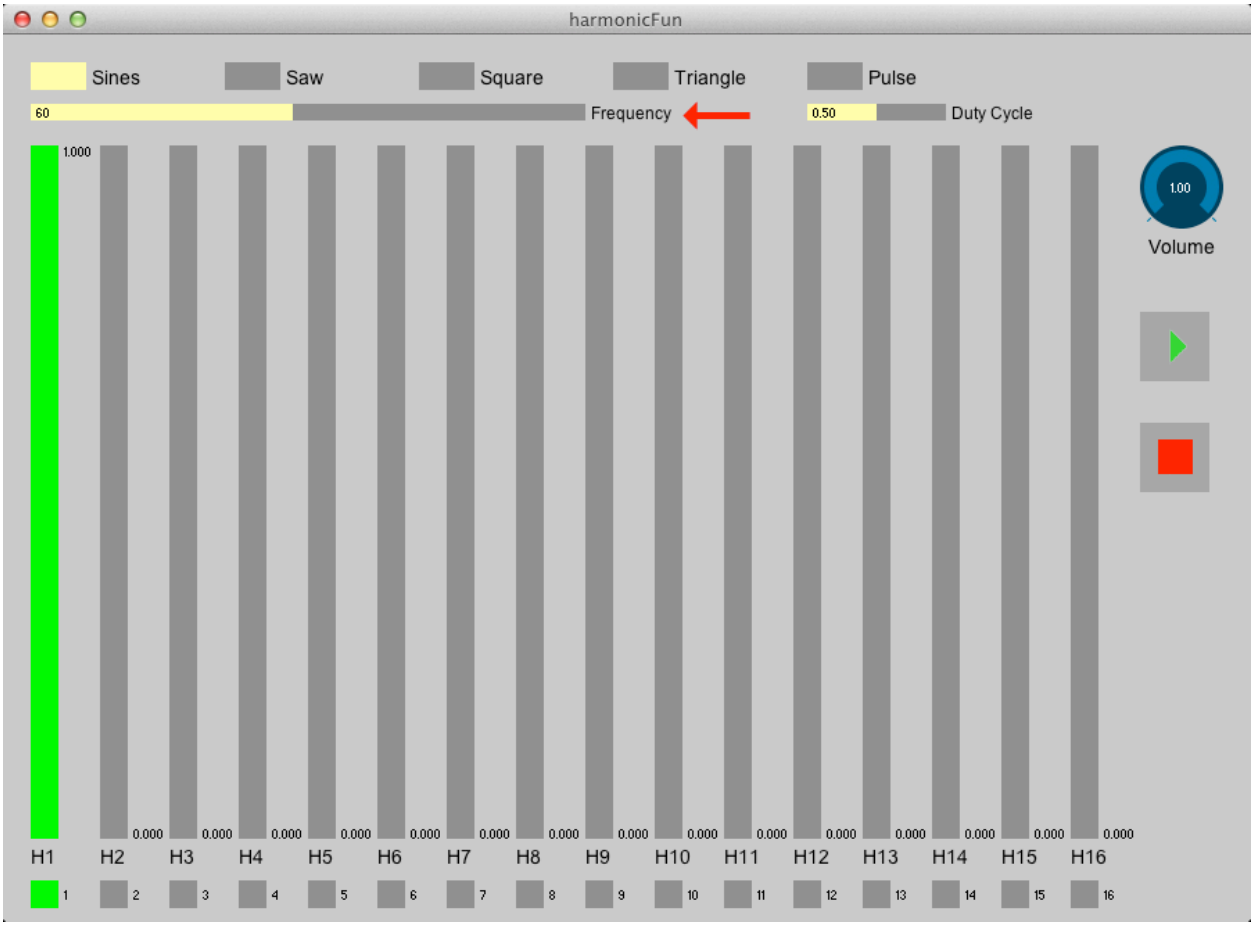
```
function void genPulse()
{
    <<< "ENTERING genPulse()", wavType >>>;
    second/samp => float SR;
    // Impulse imp; is declared at top but is not connected to dac;
    // Use the Impulse function to create a wave that is either +1.0 or -1.0
    // based on the frequency f0 and the dutyCycle (percent of 1.0's over total
    // number of samples in one period of f0 at sample rate SR
    // Pulse wave gain is  harmSinOsc[0] setting;
    // connect imp to dac
    // calculate how many SR samples would occur in one period of global frequency f0
    // get the dutyCycle or percent that the impulse = 1
    // sampOnePeriod * dutyCycle is the number of samples of value 1.0
    // 1.0 => imp.next for that many samples
    // chuck each 1.0 sample to now
    // number remaining in sampOnePeriod will have value -1.0
    // chuck each -1.0 sample to now
    // repeat when number of samples chucked is equal to one period of f0 samples
    // continue to repeat until user chooses another wave type and then
    // disconnect imp from dac
    <<< "EXITING genPulse()", wavType >>>;
}

function void doPulse( float duty )
{
    <<< "\tdoPulse()", duty >>>;
    genPulse();
}
```

Step 4

Add a Frequency slider to Processing GUI that will be used to set the fundamental frequency f0 to any MIDI note number 0 –127. Study the code in harmonicFunHW.pde for the other GUI elements to figure it out. You'll need to implement Processing code to draw the slider, get the value, and send the value to ChucK using OSC. Then you'll need to write code in ChucK to receive the frequency slider data. Study the ChucK code to see how ChucK respones to other GUI control messages.

Processing will send a float as the slider value that represents a MIDI note number. ChucK should convert the float value to an int (no decimal places) and pass the int to the ChucK Std.mtof() function which converts a MIDI note number to frequency. That frequency will become the new fundamental frequency, f0. You'll need to write code to rescale the remaining 15 harmonic frequencies to the new f0. The pitch must change in real time when the frequency slider is moved.



Helpful Documentation

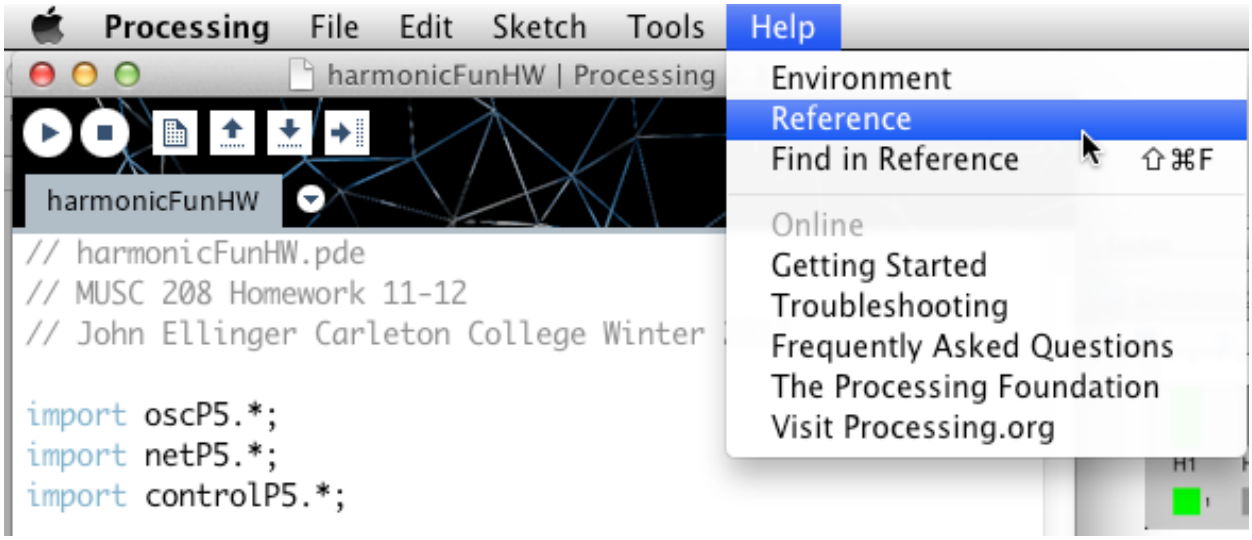
The harmonicFun.pde and harmonicFun.ck source code contain helpful comments.

Read these Processing web pages.

http://processing.org/reference/environment/

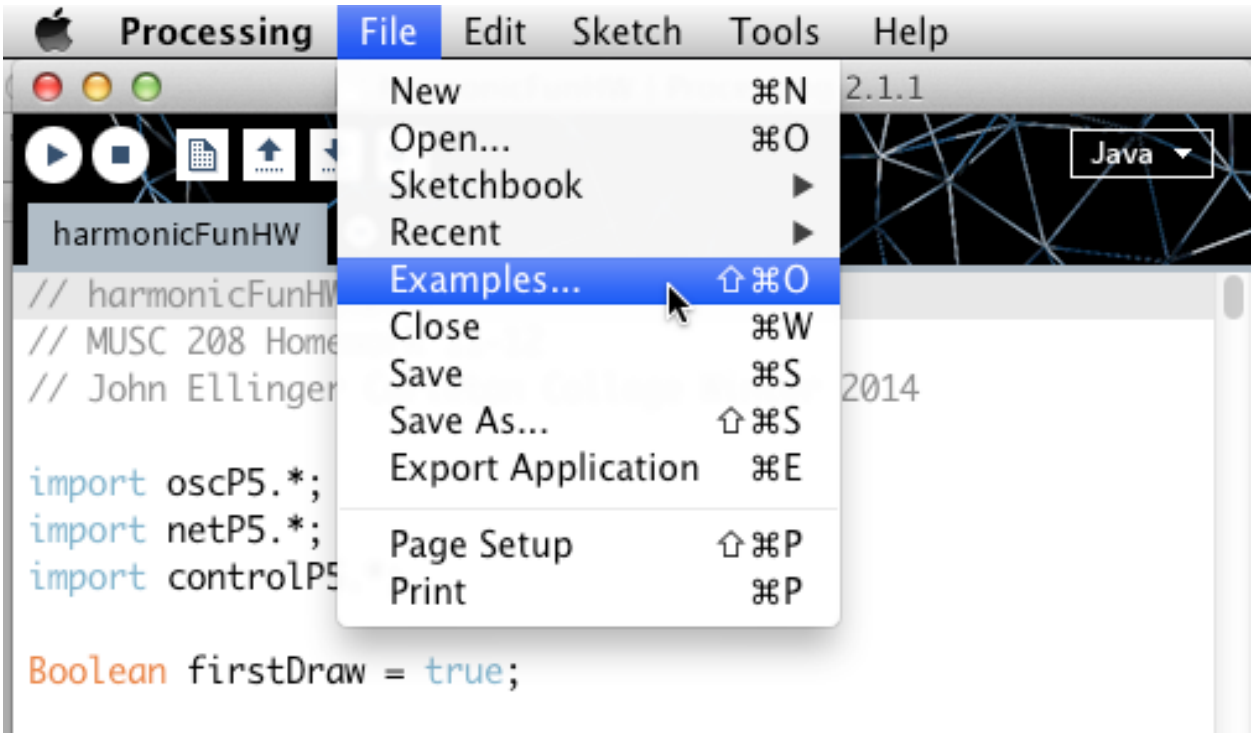
http://processing.org/tutorials/

The Processing Help Menu



The Processing Examples Folder

I used the Examples folder to figure out how the controlP5 controls worked.



controlP5 and oscP5 documentation

http://www.sojamo.de/code/

OSC documentation

http://opensoundcontrol.org/introduction-osc

Turn In

Turn in your working harmonicFunHW.pde (Processing code) and harmonicFunHW.ck (ChucK) code to the course Hand-in folder.