# Assignment 5: Magic Harp

## Overview

In Lecture, we saw how to set up Leap Motion and get data flowing into our python apps. We saw how to track hands and display their position on screen. We also saw how to create a simple gesture detector (Clap).

For this assignment, use the Leap Motion sensor to make a virtual harp. You will use Leap as the input device and output notes using the FluidSynth synthesizer. Note that pset5.py has some template classes. Feel free to add, change, remove methods and arguments as you see fit.

The module `leaputil.py` has some helper functions. You may use the Leap API directly as well. Documentation is here: https://developer.leapmotion.com/documentation/v2/python/index.html

## Part 1: Cursor and String Display [10 pts]

- Your hand controls a cursor, much in the same way that we created a cursor in class. The cursor represents a finger, moves in 2D, and will be able to "pluck" the strings.
- Define your Harp size and position on screen. Make a Cursor3D instance that displays the position of the hand in “Harp space.” The Harp object is the owner of the Cursor3D.
- Use z-depth to alter states between *active* and *inactive*: When the user extends their hand forward, plucking is enabled. Otherwise, plucking is disabled. Create a graphical indication of the state (for example, you can change the color or shape of the cursor).
- Create a single `String` (for now) in your `Harp` class. `String` should contain a kivy `Line` object with 3 points – bottom, middle, and top. The top and bottom points stay fixed, but the middle point can move around to simulate the finger “grabbing” and bending the string.

## Part 2: Plucking Gesture [15 pts]

- Write the code to detect a plucking gesture. A pluck happens when a “grabbed” string is pulled far enough left or right from its neutral position by the finger.
- You will need two thresholds, a *grab* threshold to determine when the finger should grab the string, and a (larger) *pluck* threshold that determines when the string should be released back to neutral and trigger the pluck.
- Grabbing and plucking should only happen when *active*. If the hand becomes *inactive* during a string grab, the string should return to neutral, but not pluck.
- When a pluck happens, call a callback on the Harp object. For now, you can just print that the pluck happened. The string should go back to its resting state.
- You may optionally create a string animation that goes along with the pluck.
- Test `String` and `PluckGesture` thoroughly to make sure everything behaves well and all edge cases are OK.

## Part 3: Complete the Harp [10 pts]

- Now that a single string and pluck gesture are working, create N of these to have N functional strings on your Harp. Each pluck gesture gets a unique id (corresponding to the string index) so that it can inform the callback which string was plucked.
- Create a system that can play notes based on the plucked strings. Use Fluidsynth and call `noteon` and `noteoff` functions as needed.
- Choose a "tuning" or (ie, series of appropriate notes) that should be played as the harp strings are plucked.

## Part 4: Additional Creative Element [15 pts]

Since this is a virtual harp, you can code it to do whatever you want! The analogy to a real harp is useful - people will understand how to approach it - but you can also bend some rules for more interesting interactions. Do one or more of these or come up with your own:

- Strings don't always have to have the same tunings (pitches). Find a method for dynamically changing the pitches of the strings. Remember, NO KEYBOARD here. All control of your harp must happen via the Leap.
- Add something to control with your other hand or explore what you might do with custom gestures or specific fingers.
- How would you control velocity (ie, loudness) of the plucked strings?
- Add more graphical feedback to the virtual harp. Think about colors, shapes, and animations.

Provide a brief video of you playing your harp and a README that explains the extra creative element(s).

## Finally...

Please do not upload the 147MB soundfont bank. You do not need to have individually testable parts. A single running Harp with the above specifications is fine.

Please have good comments in your code. When submitting your solution, submit a zip file that has all the necessary files (except for the soundfont bank).

# 21M.385 Interactive Music Systems

Fall 2016