Assignment 6

Overview

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

For this assignment, use the Kinect sensor to make a virtual harp. You will use Kinect as the input device, and output notes using the FluidSynth synthesizer.

NOTE: I fixed a small bug in common/core.py. Make sure to use the latest common code provided in the zip file.

Synapse Installation instructions are here:
https://drive.google.com/folderview?id=0B2elrq8YD1jxbWxnMlo2bnV4eUU&usp=sharing

Part 1: Cursor and String Display

- 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 puts their hand towards the Kinect, they become *active* and can pluck the stings. When their hand is closer to their bodies, plucking is disabled. Make sure to have a graphical indication of the state (for example, you can change the color or shape of the cursor).
- Note the kUseKinect flag. You will be able to make faster progress by simulating a Kinect's hand tracking using the mouse position.
- Create a single String (for now) in your Harp class. The String should be a 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

- 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 value) *pluck* threshold that helps determine when the string should be released back to neutral and cause a 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. Test with Kinect turned on as well (not just the mouse).

Part 3: Complete the Harp

- Now that a single string and pluck gesture are working, create N of these to have N functional strings on your Harp. Each gesture gets a unique id so that it can report in the callback which string was plucked.
- Complete the NotePlayer class that plays a note and also takes care of the note-off automatically.
- When a pluck happens, play a note based on which string was plucked using NotePlayer. Choose a "tuning" (ie, series of notes) that should be played as the harp strings are plucked.

Part 4: Additional Creative Element

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. Try 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 Kinect.
- Add something to control with your other hand or explore what you might do with other Kinect joints: Knees? Feet? Head?
- 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. Add a sentence on collaboration if you collaborated beyond lab time. 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). Upload your zip file to Stellar / Homework.