CS228: Human Computer Interaction

Deliverable 2

Description

1. In this deliverable we are going to draw three-dimensional rather than two-dimensional graphics. To begin with, create a new empty Python program, Del2.py, which only contains an infinite loop that does nothing (for the time being):

   (a) `while True:`

   (b) `    pass`

   You will need to stop the program yourself, or it will run forever.

2. We will need to import some new libraries. At the top of your program, add the following lines:

   (a) `import matplotlib.pyplot as plt`

   (b) `from mpl_toolkits.mplot3d import Axes3D`

   (c) `import matplotlib`

   Run your program; there should be no change in its behavior.

3. Now, initiatialize a 3D drawing window by placing these lines

   (a) `matplotlib.interactive(True)`

   (b) `fig = plt.figure( figsize=(8,6) )`

   (c) `ax = fig.add_subplot( 111, projection='3d' )`

   (d) `plt.draw()`

   just before you enter the infinite loop. When you run your program now you should see three orthogonal axes, but nothing drawn within them.

4. Now, copying and pasting bit by bit from last week's code...

   (a) import the Leap package,

   (b) create a leap controller object just before you enter the infinite loop,

   (c) grab the position of the tip of the index finger when a hand is over the device,

   (d) and print the x, y, and z coordinates of `tip` (but don't draw the black dot).

   Now when you run your program you should see triplets of numbers printed whenever a hand is above the device.

5. Let us also extract the position of the *base* of this bone. Add these lines

   (a) `base = distalPhalange...`

   (b) `print base`

   after printing the tip position and replace the ellipsis with the appropriate code. (Hint: consult the `bone` class in the Leap documentation.)

   Now when you run your code you should see two triplets, corresponding to the current position of the base and tip of this bone.

6. Now let us store the x, y and z coordinates of the bone's tip and base. The following lines

   (a) `xBase = base[0]`

   (b) `yBase = ...`

   (c) `zBase = ...`

   (d) `xTip = ...`

   (e) `yTip = ...`

   (f) `zTip = ...`

   should be added after line 5(b) and the appropriate code should replace these ellipses.

7. Now let us draw a line in 3D that represents this bone. Just before entering the infinite loop, place this line

   (a) `lines = []`

   which will create a list that will store all of the lines drawn to the screen.

8. Now, add these lines

   (a) `lines.append(ax.plot([xBase,xTip],[yBase,yTip],[zBase,zTip],'r'))`

   (b) `plt.draw()`

   just after line 6(f). This ensures that during each pass through the loop, a new red line is drawn to the screen. When you run your code now you should see red lines accumulating in the drawing window.

9. You can remove the lines that print the 3D coordinates to the screen now, if you haven't already, as we have a visual (rather than text) representation of the data we have captured from the device.

10. Now we will erase a line just after we draw it. Add the following lines

    (a) `while ( len(lines) > 0 ):`

    (b) `    ln = lines.pop()`

(c) `ln.pop(0).remove()`

(d) `del ln`

(e) `ln = []`

just after line 8(b). When you run your program now you should see a line that moves whenever your hand is over the device. Note also that the axes change as well.

11. We are now going to draw all of the bones detected by the device. To do this, you must create two nested loops: one that iterates over each of the five fingers, and another that iterates over each of the four major bones in each finger. To do so, replace the lines that grab just the base and tip of the distal phalange of the index finger, and replace it with

(a) `hand = ...`

(b) `for i in range(0,5):`

(c) `    finger = ...i...`

(d) `    for j in range(0,3):`

(e) `        bone = ...j...`

(f) `        base = bone...`

(g) `        tip = bone...`

(h) *[extract coordinates from base and tip as in step 6]*

(i) `        lines.append(...)`

(j) `plt.draw()`

(k) *[erase all of the lines drawn above as in step 10]*

The ...i... and ...j... segments indicate that these variables should be used to extract information about the jth bone from the ith finger. Note that the lines are drawn (line (j)) only after all of the lines have been added to the plot (line (i)). This is indicated by the fact that line (j) is aligned with line (b). When you run your code now, you should see something like this.

12. Capture a short video of your code in action at this stage. Upload it to YouTube and add it to a new playlist called `2015_UVM_CS228_YourLastName_YourFirstName_Deliverable_2`. Make sure the playlist and the video is set to Public so the grader can view and grade your submission.

13. Let's now stabilize the axes. Add the following lines

(a) `ax.set_xlim(-1000,1000)`

(b) `ax.set_ylim(-1000,1000)`

(c) `ax.set_zlim(0,1000)`

between lines 10(c) and 10(d). Run your code again. You should now see a small hand moving around in a much larger volume.

14. You will notice that the drawn hand is probably moving differently than your actual hand. To correct this, we will first need to swap the y and z axes. Alter line 11(i) to

    (a) `lines.append(ax.plot([xBase,xTip],[zBase,zTip],[yBase,yTip],'r'))`

    You will now see that the hand moves as your hand does, but from the wrong viewpoint.

15. Let us correct the view by rotating the virtual camera to point at the virtual hand from another position. We can do this by changing the azimuth of the camera by 90 degrees. Add this line

    (a) `ax.view_init(azim=90)`

    just after line 13(c). When you run your code now you should see that the virtual camera is looking at the virtual hand from the same direction that your eyes are looking at your real hand. In other words, your real wrist should be closer to your eyes than your real hand, and it should seem that the wrist of the virtual hand is coming 'out' of the screen and the virtual hand is pointing 'into' the screen.

16. Finally, you should notice that the hand appears 'flipped': when you move your real hand left, the virtual hand moves right, and vice versa. You can correct this by negating the values of the x coordinates: change line 11(i) to

    (a) `lines.append(ax.plot([-xBase,-xTip],[zBase,zTip],[yBase,yTip],'r'))`

    to accomplish this.

17. Now, using the strategy from the previous deliverable, determine the minimum and maximum values obtained for the x, y, and z coordinates attained by any of the finger bones. Copy these six numbers into lines 20(a)-(c). The hand should now be larger, and move within a smaller volume like this.

18. Capture a short video of your system in action, upload it to YouTube, append it to your `Deliverable_2` playlist, and submit the playlist link to BlackBoard by the deadline.