CS1566 Assignment 1
Basic 2D Shapes, Interaction and Physics

Out: Tue 08/29[1]. Due: Thu 09/06 11:59pm

1 Overview

In this assignment you will learn the basics of OpenGL and GLUT drawing and interaction. You will create a window; draw various 2D shapes using various drawing styles and colors. You will modify callback functions to interpret correctly mouse and keyboard input; you will create a simple user-driven, physically-based animation; and you will build and learn to use your first, very simple scenegraph.

2 Using OpenGL and GLUT

Before you embark on this assignment, make sure you can compile and run the sample warmup program 'glhello'. Instructions for compiling and running the sample program on the departmental machines (or at home) are on the course webpage, under the Assignments link.

The sample glhello files are also available from the course webpage, under the Assignments link. Skim through the two source files, glhello.h and glhello.c and modify them as you wish. For example, modify the my_display() function in the sample program so that it displays something more interesting than the default (e.g., adding more polygons in different colors or rendering styles).

3 The Assignment

Download the assignment example code (hellomouse.h, hellomouse.c), available from the course web-page, under the Assignments link. Please note that the example code is a courtesy, and you are under no obligation to use these files; feel free to write your own code from scratch, should you choose so.

Make sure you can compile and run the support code (hellomouse.h, hellomouse.c; use the Make file if appropriate). It should display a blue rectangle on a black background; the rectangle should show up on the bottom left corner of the window. When you click the left mouse button within the window, the rectangle should get relocated. Try random mouse clicks, keyboard hits etc, and check out the messages that the program prints on each event.

[1] Posted and due early **as a courtesy** to those considering the add/drop deadline.

Then, read the support code and try to understand what each line does. For the first part of this assignment, you'd only need to modify the my display() and my mouse() functions. Play some more with the program.

3.1 The Actual Assignment

Task 1: Modify the code so that the rectangle gets drawn at the location where you click the left mouse button. When you click, the rectangle should show up where you clicked. (Don't sweat it too much trying to get the rectangle nicely aligned with the click, any corner of the rectangle over-imposed with the mouse click location will do).[Hint: see the my_mouse() function; there's only one line of code you'd need to modify].

Task 2: Modify the code so that mouse clicks alternatively create a blue rectangle or a red circle [Hint: there is no glCircle() function. Think fan-strip, cosine and sine; trigonometric functions are accessible via math.h]. I.e., first mouse click results in a rectangle, second mouse click results in a circle, third in a rectangle, fourth in a circle, etc. (you get it).

[Hint: you probably want to create a list or array of objects at this point; for each object you want to store at least the current location of the object, its set of vertices, as well as the object type – circle or rectangle; a 0 or 1 as the type would do. Then you want to iterate through this list/array when drawing, as well as when processing timeout events. Voila. This is your first, very simple, completely flat scenegraph. Don't worry about trees yet.]

Task 3: Use the timer to make the rectangles and circles slide down automatically once initially placed via the mouse click (i.e., update automatically the position of the rectangle/circle). E.g.: click the mouse, the rectangle appears, then starts sliding down, Tetris style.
Note: You are not allowed to call any of the renderer-based OpenGL transformations (i.e., no glTranslate() etc). You need to, literally, update the vertex coordinates of each object as it moves.

Task 4: Add physics to your program. Your shapes should accelerate as they slide towards the bottom of the window. When they reach the bottom of the window, they should bounce back up, then decelerate (slow down) as they travel upwards, then start falling back down etc., every time losing some of velocity and height.

Hint: There are multiple silly ways to fudge this result; but we strongly recommend you treat your objects as particles (see the course notes), with acceleration, velocity and position associated with them. The only forces acting on the object initially are G and possibly air drag; velocity flips when colliding with the bottom etc.

Task 5: Modify the code so that it handles at least 25 objects on the screen at the same time. All existing objects should continue to move. After the 26^{th} click you can replace the first object with the last one (i.e. a static 25 element circular array would do, if you are not comfortable using lists or other more sophisticated structures).

Note: The last 4 tasks require some actual thought put into the implementation details. You will need to build some sort of data structure to hold your objects etc.

For Extra Credit idea #1, implement collision detection between the objects.
For Extra Credit idea #2, go creative and do something fun using the keyboard, mouse, and/or timer.
For Extra Credit idea #3, go ballistic and add some more complex physics to your brand new toy. Tim Luciani is a big fan of the Google logo for September 7th, 2011: http://www.youtube.com/watch?v=bEukHREWhwQ

4 Support/Example Code

- hellomouse.h - header file for the program
- hellomouse.c - main file for the program
- Make file – Make file; if using Windows VS or Mac OSX , you won't need it.
- Readme.txt - text file containing your name, ID, description of Extra Credit work, and any additional comments.

5 Handing In

To hand in this assignment, follow the Submit link on the course website and upload the following files:

- your modified source files (i.e., hellomouse.c and hellomouse.h; or your own files)
- your Make file (if any)
- filled out Readme.txt file

In the unlikely event you run into any trouble while submitting the homework online through the Submit script, do not panic. Email the TA immediately, and use anonymous ftp to transfer the files into the directory that has been created for you: /public/incoming/CS1566/yourdir/. Note: use ftp only as an emergency backup plan.
NB: We will grade this homework only after you have signed a copy of the Collaboration Policy.

6 Grade

Task	**Points**
Building and running source As-Is	5
Drawing where the mouse click happened	15
Drawing a circle on clicks	15
Animating the rectangle/circle	10
Simple physics (collision w/ bottom 5 pts,	25

acceleration 10 pts, deceleration 10 pts)	
Handling correctly at least 25 objects	20
Submitting the completed Readme.txt file	10
Total	**100**

Extra Credit Items have variable value depending on difficulty (EC1 is 10, while EC4 may go up to 30).
On the other hand, you may lose up to 15 points if the grader finds your code particularly badly designed or difficult to understand (e.g., drawing code really belongs into your my_display() or similar function, and not in some other callback function like the mouse one; duh!).

Good luck!