Lab/Homework #1
Due Sept. 3, 2019

Introduction to glut and OpenGL

Part 1. Glut Scavenger hunt: The goal of this lab is to learn to search for and read documentation, and to experiment with glut window management and event handling by writing and registering callbacks and setting glut state. For glut information see: the glut header files, the course textbook, glut documentation (https://www.opengl.org/resources/libraries/glut/spec3/spec3.html) and freeglut documentation (http://freeglut.sourceforge.net/docs/api.php).

Warning: A few glut features in the documentation are not fully supported on all systems. If you find one of these features make a note of it and move on.

- Build and run. Get the Lab0.zip file from Blackboard. Build and run the code in Visual Studio.

- OpenGL version. Look at the console window and note the OpenGL vendor and version. If you are on your own laptop make sure the vendor is not Microsoft and that the OpenGL version is 3.3 or greater. Otherwise you will have trouble later in the class.

- Basic debugging.
  - Open Lab0.cpp. Insert a breakpoint in the void mouse(…) function by clicking in the left margin of one of the source code lines. Press F5 to start debugging.
  - Add 'button' and 'state' variables to the Visual Studio watch window.
  - Press F5 to continue debugging after breaking.
  - Press and release different mouse buttons and see that the values in the watch window change.

- Getting glut state. Modify Lab0 or copy to a new project and add the following features:
  - **[5 pts]** Print "Shift active" / "Shift inactive" in the keyboard callback in response to shift key being held/not held when a key is pressed. Hint: use glutGetModifiers(…)
  - **[5 pts]** Print the following to the console at startup, after the OpenGL version strings are printed: (Hint use glutGet(…))
    - Screen width and height
    - Window width and height
    - Number of bits per pixel in the red, green, blue, and alpha channels of the color buffer.

- Window management and callbacks.
  - **[5 pts]** Add your name at the beginning of the titlebar text. You will be doing this for all future homework submissions.
  - **[5 pts]** When 'f' is pressed change to fullscreen mode. When 'f' is pressed again change back to windowed mode.
  - **[5 pts]** Allow the window to be moved using the arrow keys.
  - **[5 pts]** Make '+' and '-' increase and decrease the window size in 10 pixel increments

- **[5 pts]** When F1 is pressed hide the window.
- **[5 pts]** Make the function keys F5-F8 select different mouse cursors (your choice)

- Glut callbacks
  - **[5 pts]** Change the screen color to black every 5 seconds. Hint:(use glutTimerFunc(…))
  - **[5 pts]** If a function key or normal keyboard key is pressed print 'PRESSED", if it is released print "RELEASED"
  - **[5 pts]** Make mouse wheel up/down events increase/decrease the window size in 10 pixel increments
  - **[5 pts]** If the window is resized (reshaped) print the new size in pixels to the console.

Part 2. The goal of the following part of the lab is to get experience manipulating OpenGL state and editing shader code. For the following sections answer the questions or modify Lab1, or copy into a new project and make the following modifications:

- Are any glew functions called? What do they do?

- Polygon Mode. **[5 pts]** Look at the initOpenGl() function in Lab1.cpp. Try the different glPolygonMode options and tell me what you see.

- Fragment shader. Experiment with changing the fragment shader source code.

  - Open the lab1_fs.glsl file. Add some garbage text to the file and recompile (by pressing 'r'). What happens? Look at both the glut window and the console window. Try rebuilding and rerunning the project. What happens? What does this tell you about when shader compilation happens?

  - **[5 pts]** Change the fragment shader to draw the mesh in red instead of white by changing the assignment to the fragcolor variable.

- Time-varying fragment shader. Look at the idle(…) function in Lab1-2017.cpp. Pass the time_sec variable to the fragment shader and use it.

  - **[10 pts]** Modify idle(…) to send the value of time_sec to the fragment shader. Look at display(…) to see how it sends the PVM matrix to the shader. For the time variable use glUniform1f(GLint location, GLfloat f).

  - **[10 pts]** Change the lab1_fs.glsl file so that the color changes with time. Add a declaration for a uniform variable that holds the time which gets passed from the client program, like "uniform float time;" Then use the value of time when assigning to the fragcolor variable.

- Change the mesh. **[10 pts]** Find a new mesh (your own design or from some repository). Find the hardcoded name (cow.ply) in the source code and change it to the name of your mesh.