CS/SE 3GC3 Lab 1

September 8, 2019

1 Resources

1. Red Book Chapter 1 `http://www.glprogramming.com/red/chapter01.html` (particularly "A Smidgen of OpenGL Code")
2. Red Book Chapter 2 `http://www.glprogramming.com/red/chapter02.html` (particularly "OpenGL Geometric Drawing Primitives")
3. GLUT documentation (e.g., `glutInitWindowSize`) `https://www.opengl.org/resources/libraries/glut/spec3/spec3.html`

2 Exercises

These exercises will not be graded. This lab is an opportunity to get 1-on-1 help and practice for future lab tests with access to Google/Ecosia/preferred search engine. Future labs will contain a mix of practice and graded exercises.

1. Write a C++ program which takes two arguments and prints them to standard output. Print an error message if the executable is ran without exactly two arguments.

   ```
   $ ./a.out 640 480
   640 480
   $ ./a.out 640
   Error! Must specify two outputs.
   ```

2. Compile and run `square1.cc` on the department `gpu1` server. You can do this by simply executing `make`.

   (a) Modify the program to call `glutInitWindowSize`, allowing the window dimensions to be configured with command line arguments (use exercise 1).

(b) Use `glutKeyboardFunc` to close the window when the user hits Escape or 'q'. Consult the GLUT documentation or Chapter 1 of the Red Book. You can use `std::cout` or `printf` to find the ASCII code for the escape button and play with the arguments the callback function receives (or use Google!). You will use this code in future assignments!

(c) Consult Chapter 2 of the Red Book on "OpenGL Geometric Drawing Primitives" and change the `GL_POLYGON` primitive to other types! Experiment! Try drawing a shape other than a square.

(d) Call `glColor3f` once before all the calls to `glVertex2f`. You should achieve a solid color shape. This is OpenGL acting like a state machine: you set the color state once for all future vertices.

(e) Call `glColor3f` before multiple `glVertex2f` calls to change the color at each vertex of your shape. Note how the colors are interpolated.

(f) Use the code below (or similar) and `glColor3fv` instead of `glColor3f` (note the `v` suffix) to specify vertex colors with arrays instead of "hard-coded" positional arguments. This will be very useful for future assignments!

```
static GLfloat red[] = { 1.0, 0.0, 0.0 };
```

(g) Add a second square or other shape by using a second `glBegin` (and end) call or changing primitives (i.e., draw independent triangles with a single `glBegin/end` call by using the `GL_TRIANGLES` primitive).

(h) Consult the documentation (`https://www.opengl.org/resources/libraries/glut/spec3/node89.html`) for `glutSolidTeapot` (or other shapes) and draw a 2D teapot in a single call!

(i) Consult the documentation (`https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluOrtho2D.xml`) for `gluOrtho2D` and chapter 3 of the Red Book. Call `gluOrtho2D` before the call to `glutMainLoop` call with some values like `gluOrtho2D(-5, 5, -5, 5)`. Compile and look at the results, play with the values. You can try adding keyboard controls to "zoom" in and out.

(j) Call `glutPostRedisplay` (`https://www.opengl.org/resources/libraries/glut/spec3/node20.html`) at the end of your `display` function. Your main loop is now going to keep re-drawing (rather pointlessly, since there's no animation happening).

(k) Print how many frames per second are being rendered by using `glutGet(GLUT_ELAPSED_TIME)` to get the milliseconds since the program started.

3 Advanced Exercises

The previous exercises are essential for your understanding of the course material and future problems you will be expected to solve. However, if you happen to

be bored or finish early, try the following exercises (in no particular order).

1. Enable double buffering and animate color or orthographic perspective (with `gluOrtho2D`).

2. Read vertex positions from a file and render the corresponding triangles. This "model file" scheme could be used for bonus features in future assignments!