CS/SE 3GC3 Lab 5

November 12, 2020

1 Resources

1. Red Book Chapter 5 `http://www.glprogramming.com/red/chapter05.html`
2. GLUT documentation (e.g., `glutInitWindowSize`) `https://www.opengl.org/resources/libraries/glut/spec3/spec3.html`
3. `glFrontFace` `https://www.khronos.org/opengl/wiki/GLAPI/glFrontFace`
4. `glCullFace` `https://www.khronos.org/opengl/wiki/GLAPI/glCullFace`

2 Lab 5

1. Run `make` to compile and run `tut5.cc`.
2. You have been given the mesh for a cube. This mesh assumes that front faces are drawn with a counter-clockwise winding order. The order we specify vertices in a triangle is what determines if the triangle is "front facing" or "back facing". Refer to Figure 1.

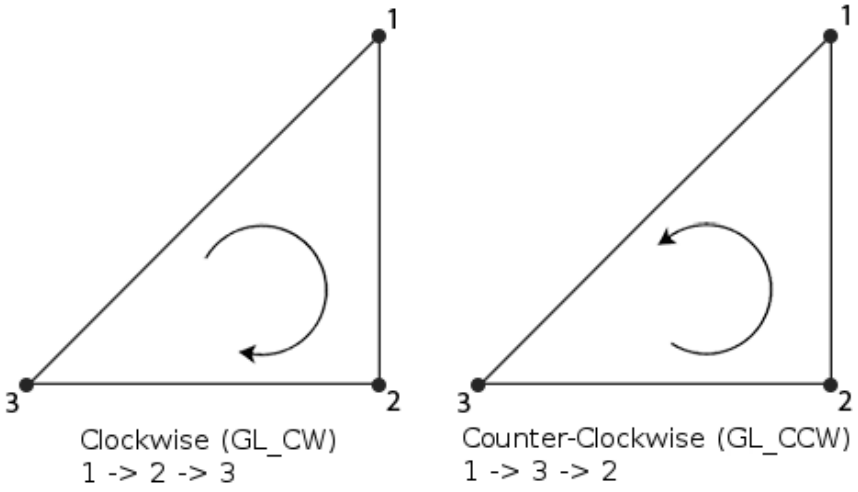

Figure 1: From `https://www.khronos.org/opengl/wiki/Face_Culling`

3. You have not been given the normals for the cube. Your normals should be perpendicular to each face of the cube. They should all point outwards. Fill in the `cubeNormals` values. You should not guess what these normals should be! For each triangle you obtain a normal by taking the cross product of two edges of the triangle (and then normalizing the result). We are only using face normals here, not vertex normals.

4. Consider the first triangle of the front face of the cube.

   (a) It uses vertices 4, 5, and 6 (in that exact counter-clockwise order).

   (b) The coordinates there are $(-1, 1, 1)$, $(-1, -1, 1)$, and $(1, -1, 1)$.

   (c) Compute two vectors: edge 1 and 2 from vertices 4, 5 and 4, 6. This produces edge 1: $(0, -2, 0)$ and edge 2: $(2, -2, 0)$.

   (d) Now take the cross product of edge 1 and 2. You'll get $(0, 0, 4)$.

   (e) Normalize this and you get $(0, 0, 1)$.

   (f) This is the unit vector perpendicular to the front face of the cube, pointing away from the surface instead of pointing to the inside of the object. This is exactly what we want.

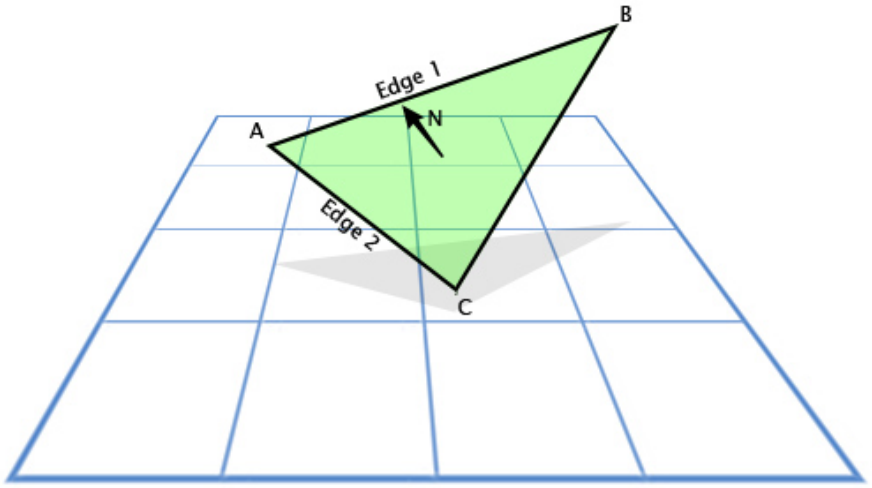

Figure 2: From `http://blog.wolfire.com/2009/07/linear-algebra-for-game-developers-part-2/`

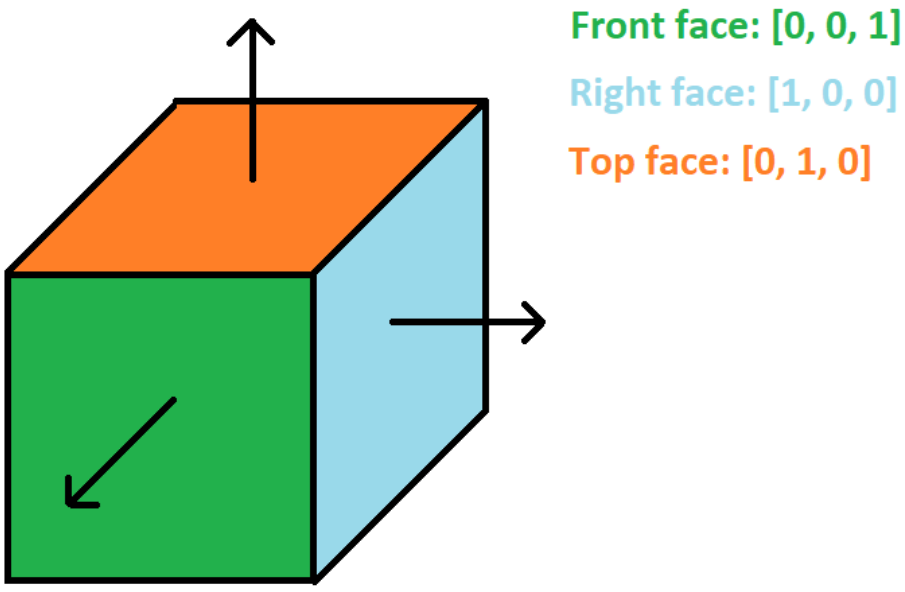

Figure 3: From `http://www.mbsoftworks.sk/tutorials/opengl4/014-normals-diffuse-lighting/`

5. Complete the `drawCube` function.

6. We can make non-photo realistic graphics with techniques like "cel" or "toon" shading.

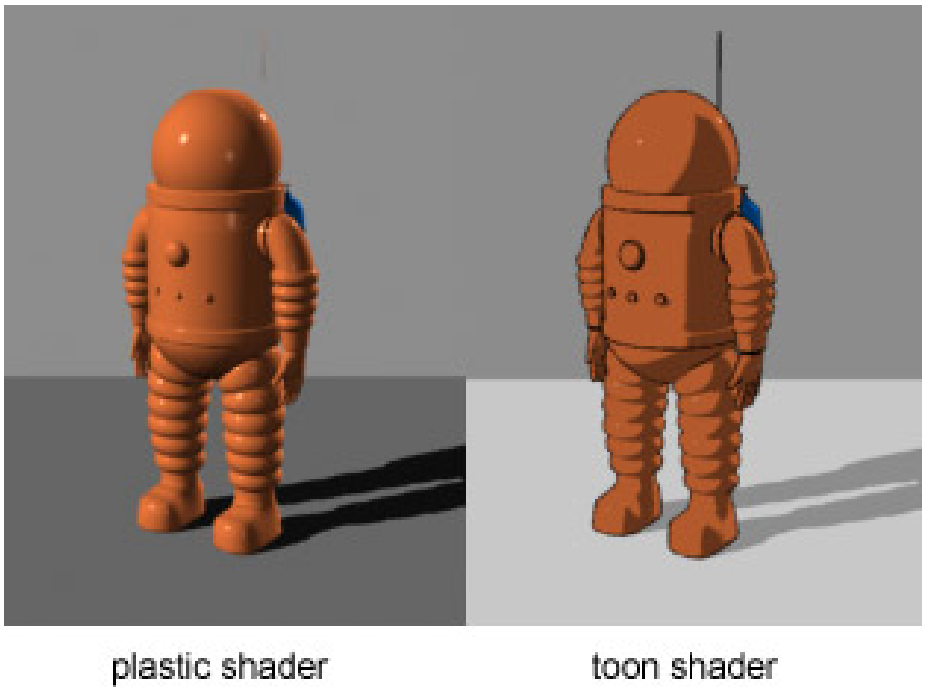

Figure 4: From `https://en.wikipedia.org/wiki/Cel_shading`

7. Without more advanced graphics pipeline techniques like *shaders* (outside the scope of this course) we can only approximate cel shading.

8. This approximation relies on drawing a black (or a different outline color) version of the shape and being clever with face culling.

   (a) Draw a copy of the object that is to be cel shaded.

   (b) This copy should use the outline material (call `setOutlineMaterial`).

   (c) This copy should be slightly bigger.

   (d) This copy should use `glCullFace` to cull the front faces instead of the back faces.

   (e) Your original object should use `glCullFace` to cull the back faces.

9. Apply this technique to the teapot and cube.

10. Your final result should look something like this.

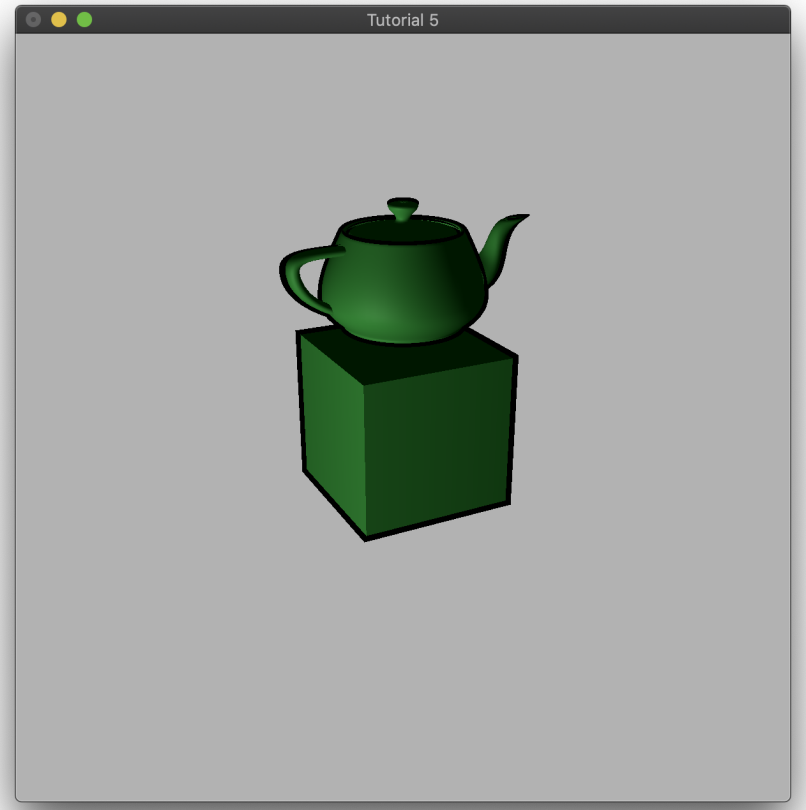

Figure 5: Result