ME/HCI/CS/CprE 557 - Computer Graphics
Assignment 1
Geometric Modeling with OpenGL

Rafael Radkowski
Iowa State University
rafael@iastate.edu
Fall 2018

The goal of this homework is to practice geometric modeling using OpenGL primitives. When completing this assignment, you should be able to a) prepare a Visual Studio project for OpenGL code, b) be able to assemble a 3D model made of primitives, and c) be able to render an object in 3D.

Problem 1

Create a primitive model of the object you see in Figure 1. A drawing is attached as supplement *assignment-a1_sketch.pdf.*

Keep in mind, for 3D computer graphics, all geometric object must be represented as primitive models using triangles, triangle fans, or other primitive. Primitive models are optimized for fast rendering. They are basically "easy to use" (compared to NURBS or other surface representations) but also come along with some restrictions (see classnotes).

Your task is to represent the object shown in Figure 1 as two different OpenGL 3D primitive models. The first model should only contain **TRIANGLE_STRIP**s, the second model only **TRIANGLE**s. Note, you also need to add vertex color information to your model. Otherwise, it will not be rendered correctly (The color does not need to be blue).

- Create a sketch to plan the vertices and the number of edges on paper (you can use the attached sketch).
- Create a list of vertices with the coordinates for each vertex.
- Assemble the vertices to primitives. Represented the model as one triangle strip or multiple triangle strips but try to minimize the number of manifold surfaces.
- Use the function *unsigned int createTriangleStripModel(void)*, which uses the predefined vertex buffer array *vbaID[0]*. Fill your vertices into this object.

Problem 2

Prepare the object a second time. Use OpenGL **TRIANGLE**s only for the second model.

- Create also a sketch that indicates your vertices and the edges on paper (you can use the attached sketch).
- Create a list of vertices with the coordinates for each vertex.
- Assemble the vertices to triangles and / or polygons. The number of triangles and / or polygons is on your

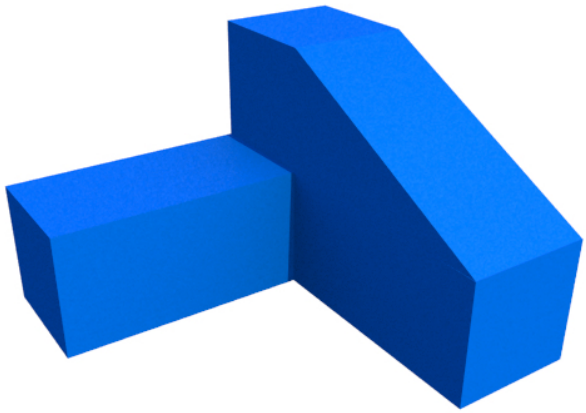


Figure 1: Goal of this assignment: create two versions of this model, each with a different primitive set

- Use the function *unsigned int createTriangleModel(void)*, which uses the predefined vertex buffer array *vbaID[1]*. Fill your vertices into this object.

Note

A shader program has already been prepared. The variable name of the shader program is *program*. You can use the program to render both object. You need to create vertex buffer objects to store the vertices and color information on the graphics card. To allow the shader program to work correctly, store the vertex information at index [0] of your vertex buffer object, e.g. vbo[0], and the color information at index position 1, e.g. vbo[1].

Deliverable

The following deliverables must be uploaded to Blackboard

- The sketch that you used to prepare the surface model.
- The code for your solution.
- Screenshots that show your solutions.

Due data: Friday, Sep. 7th, 2018, 8:00 pm
Late submissions will not be accepted!

Grading

The following rubric will be used for grading. Max. 10 points can be earned.

- Code to create one object using only triangle strips correctly completed (2pt).
- Code to create one object using only triangle correctly completed (2pt).

- Code to copy all your data to the GPU correctly completed. The GPU is correctly configured (2pt).
- Code to draw both objects correctly completed (2pt).
- Render both objects correctly without any visual errors (missing primitives or color, vertices at the wrong location) (2pt).

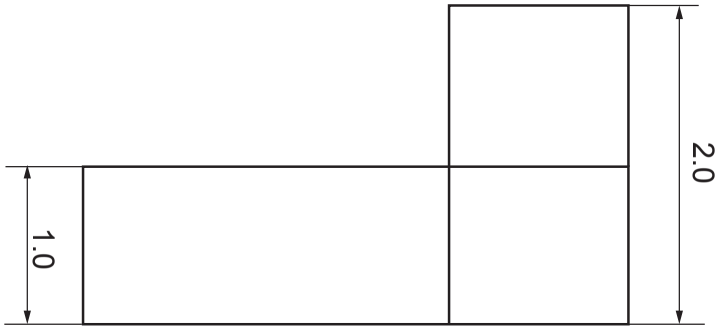
Code template

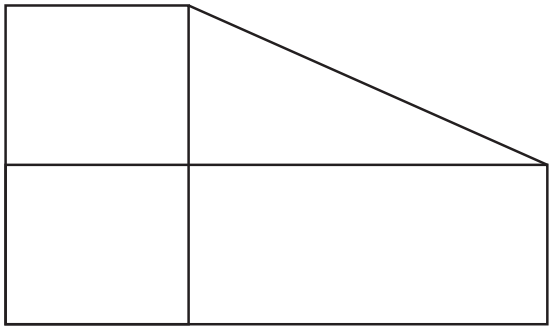
You will find a Visual Studio code template in the Git repository or on Blackboard, named *02_3D_Modeling*. You can ignore most of the code. The important part for you is part of the file *hw2_main.cpp*, especially the four functions:

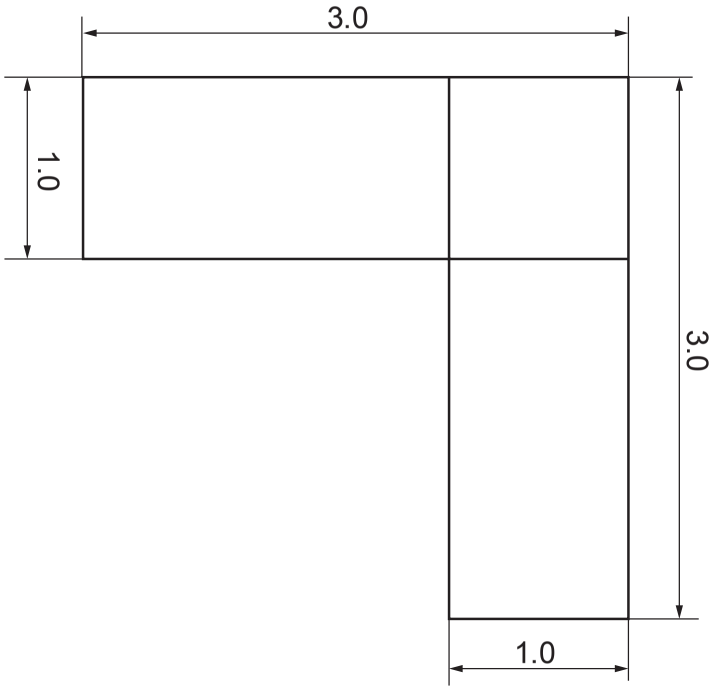
```
1
2 // USE THESE vertex array objects to define your objects
3 unsigned int vaoID[2];
4
5 unsigned int vboID[4];
6
7 /*!
8  ADD YOUR CODE TO CREATE THE TRIANGLE STRIP MODEL TO THIS FUNCTION
9  */
10 unsigned int createTriangleStripModel(void)
11 {
12     // use the vertex array object vaoID[0] for this model
       representation
13     //TODO:
14     vaoID[0];
15
16     return 1;
17 }
18
19 /*!
20  ADD YOUR CODE TO CREATE A MODEL USING TRIANGLE PRIMITIVES
21  */
22 unsigned int createTriangleModel(void)
23 {
24     // use the vertex array object vaoID[1] for this model
       representation
25
26     //TODO:
27     vaoID[1];
28
29     return 1;
30 }
31
32
```

```
33
34 /*!
35  ADD YOUR CODE TO RENDER THE TRIANGLE STRIP MODEL TO THIS FUNCTION
36  */
37 void renderTriangleStripModel(void)
38 {
39
40     // Bind the buffer and switch it to an active buffer
41     glBindVertexArray(vaoID[0]);
42
43   // HERE: THIS CAUSES AN ERROR BECAUSE I DO NOT KNOW HOW MANY
       TRIANGLES / VERTICES YOU HAVE.
44   // COMPLETE THE LINE
45     // Draw the triangles
46     glDrawArrays(GL_TRIANGLE_STRIP, ? , ? );
47
48
49     // Unbind our Vertex Array Object
50     glBindVertexArray(0);
51 }
52
53
54
55 /*!
56  ADD YOUR CODE TO RENDER THE TRIANGLE MODEL TO THIS FUNCTION
57  */
58 void renderTriangleModel(void)
59 {
60
61     // Bind the buffer and switch it to an active buffer
62     glBindVertexArray(vaoID[1]);
63
64
65   // HERE: THIS CAUSES AN ERROR BECAUSE I DO NOT KNOW HOW MANY
       POLYGONS YOU HAVE.
66   // COMPLETE THE LINE
67     // Draw the triangles
68     glDrawArrays(GL_POLYGON, ? , ?);
69
70     // Unbind our Vertex Array Object
71     glBindVertexArray(0);
72 }
```

Use the Vertex Array Objects vaoID[0] and vaoID[1] to prepare your models. The code will render the two objects automatically, when you use the vaoIDs as indicated. They will appear at two different locations.







Do not take a ruler to measure any length,
use the specified values