CIS 441/541: Project #2A
Due May 11th, 2021 (which means 6am May 12th, 2021)
Worth 8% of your grade

Overview:

You will implement an OpenGL program that makes the same image as your project 1F code.

OpenGL programming is difficult. It is very easy to end up with a shader that will not compile or to see nothing on the screen. These situations are very hard to debug. Therefore, we have organized the project in a series of phases, and each phase completes with a checkpoints. You should only attempt phase N+1 after phase N is fully completed.

If you have trouble with Checkpoints #1 or #2 (which are getting software setup), then please make a discussion post to the class Canvas. There is no "cheating" when it comes to these steps, and we should all trade tips.

After phases 3 and 4, you should make a copy of your code, i.e., project2A_phase3.cxx and project2A_phase4.cxx.

Checkpoint #1: Install software.

We hope that most students will be able to run OpenGL programs natively on their machines. If not, we have set up a virtual box for you to use.

Explicitly, try to install third party software on your machine. If that works, then super. If not, then install the VirtualBox image.

== installing for Mac ==

```
brew install glfw
brew install glew
brew install glm
```

== installing for Linux ==

```
sudo apt install libglfw3-dev libglew-dev libglm-dev
```

== virtual box install ==

The virtual box image is available at http://ix.cs.uoregon.edu/~hank/CIS441.ova

The username and password are both cis441

The project is located at ~/project2A

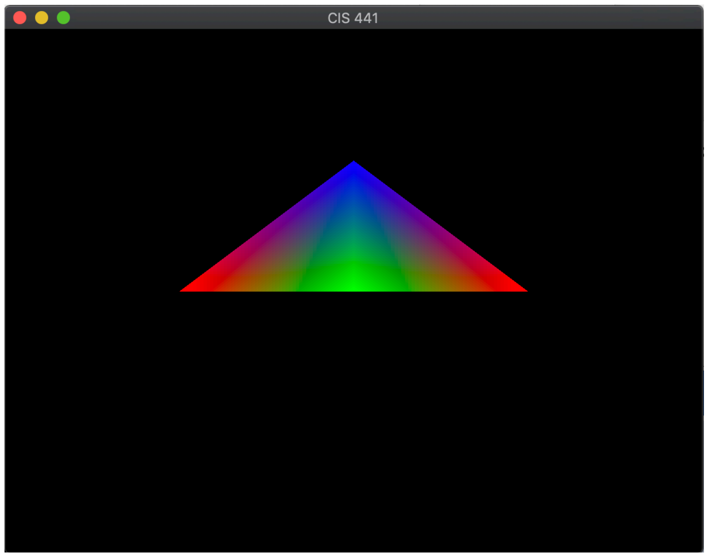
Checkpoint #2: Run sample program.

Download the starter code (named project2A.cxx), the data file (proj2_data.h) and the CMakeLists.txt file.

Run the following three commands:

```
cmake .
make
./project2A
```

You should see this:



If you do not see this, then either:

(1) Post a discussion to the class Canvas with details and we will see if we can debug
(2) Return to checkpoint #1 and install the VirtualBox

You should not move on from this checkpoint unless you see the image above.

Checkpoint #3: Modify the Vertex Array Object.

The starter code sets up two triangles. Two of the vertex buffer objects are for vertices, and they each have data for four vertices. These vertex buffer objects are for position and color. The third vertex buffer object is for indices.
The data in "proj2_data.h" has different data:

- tri_points: Position for each vertex (this is the same as the starter code)
- tri_normals: Normals for each vertex
- tri_data: Data value for each vertex (this is in place of color)
- tri_indices: Indices for each triangle (this is the same as the starter code)

The shader programs assume your data is organized as follows:
Positions are at location 0
Data is at location 1
Normals are at location 2

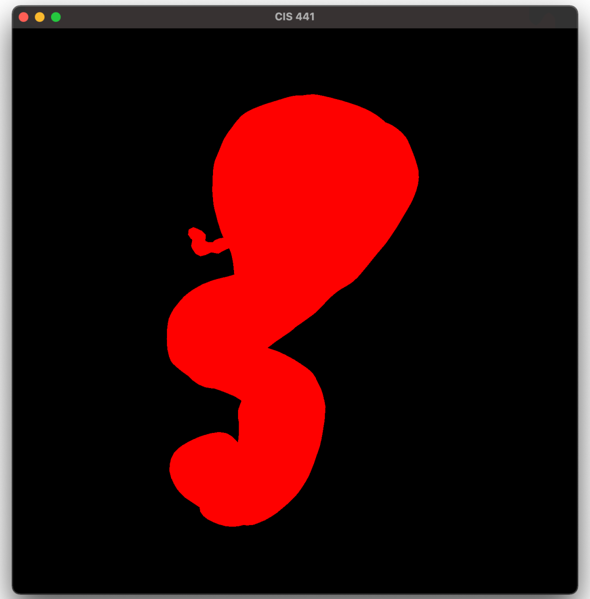
Please make sure you add these arrays in the correct location. This is done with the first argument of glVertexAttribPointer.

Indices are of type GL_ELEMENT_ARRAY_BUFFER, so they do not have a location in the shader program. (They are used before the shader executes.)

Also, make sure to update the glDrawElements function in main for the correct number of triangles.

Finally, you should add “#define PHASE3” to the top of your source code. This will make changes the shader programs and view matrices to work on your new data.

Compile and execute. You should see:



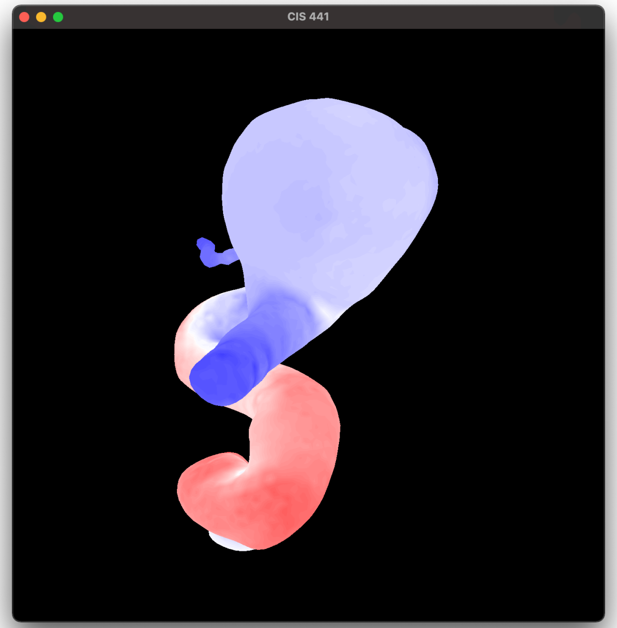
You should not move on from this checkpoint unless you see the image above.

Checkpoint #4: Add Color.

First, you should add “#define PHASE4” to your code. Further, do not remove “#define PHASE3” – you need both.

For this phase, you will modify the fragment shader. By adding “#define PHASE4”, the data you loaded in VBOs (for array “tri_data”) will now be passed out of the vertex shader and be LERPed by the GPU before it gets to the fragment shader. You should use this data to make a color. These data values range from 1 to 6. They should be colored as follows:
Values between 1 and 4.5: LERPed from (0.25, 0.25, 1.0) at value 1 to (1.0, 1.0, 1.0) at value 4.5.
Values between 4.5 and 6: LERPed from (1.0, 1.0, 1.0) at value 4.5 to (1.0, 0.25, 0.25) at value 6.

At the end of this phase, you should see this picture:



You should not move on from this checkpoint unless you see the image above.

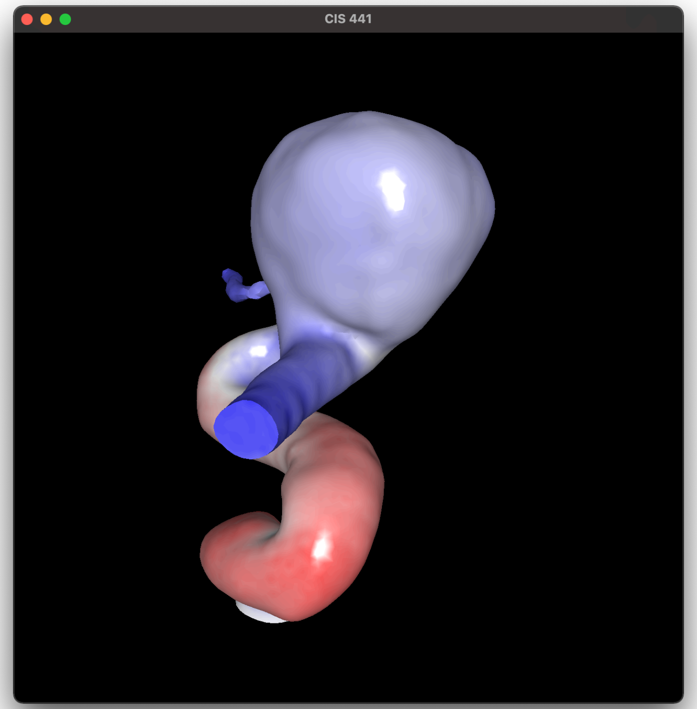
Checkpoint #5: Add Shading.

First, you should add "#define PHASE5" to your code. Further, do not remove "#define PHASE3" or "#define PHASE4" – you need all three.

For this phase, you will add shading by modifying both the vertex shader and the fragment shader. By adding "#define PHASE5", the vertex shader will now produce a shading value. That shading value will be LERPed by the GPU before it gets to the fragment shader. The fragment shader will then take the LERPed shading value and modify its colors to be darker.

Note that the majority of work for this phase is modifying the vertex shader to do Phong lighting. If you look at the "#define PHASE 5" code in the vertex shader, then you will see code for getting lighting information.

At the end of this phase, you should see this picture:



Once you have gotten this picture, you have finished Project 2A.

What to Turn In.

You should turn in a single file, project2A.cxx. Do not upload proj2_data.h or your CMakeLists.txt.

This code will be evaluated by the instruction staff by running your program and visually inspecting the output of Phase 5. There is no differencer program, so look closely for shading and coloring issues.

Projects that have non-obvious shading or coloring issues will receive 6.5/8.0. Projects that have obvious issues will receive at most 3.5/8.0.