Introduction to Computer Graphics CMPS 4213

Purpose

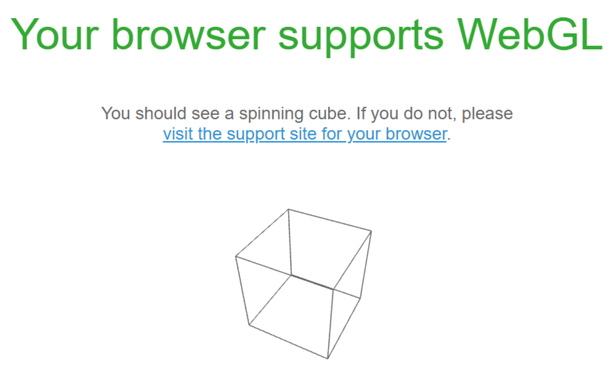
The goal of this assignment is to establish your WebGL development environment and to deepen your understanding of key concepts such as geometry, colorization, and coordinate systems. You'll begin by creating a basic "Hello World" application in graphics, which involves drawing a simple triangle. From there, you'll expand your project to incorporate more advanced features, such as custom colorization and the ability to rotate the triangle. Furthermore, you will create a more detailed scene, introducing additional geometry and color elements.

Note: DO NOT Use immediate mode graphics, that is your code should not call GLBegin()/GLEnd(). It should be using shaders and VBOs and VAOs!

Part I Set up your environment – 0 points

If you're using WebGL, there's typically no need to install extra libraries or perform any setup. To check if your web browser supports WebGL, visit get.webgl.org. If you see a rotating cube, your browser supports WebGL and it's functioning properly.

However, for performing math or vector operations, you may need additional libraries like gl-matrix.js, MV.js, or math.js. These are not part of WebGL itself but are commonly used by JavaScript libraries to assist with these tasks.



Your browser supports WebGL

You should see a spinning cube. If you do not, please visit the support site for your browser.

Part II – 50 points

Draw a triangle on the screen. We previously followed a WebGL tutorial for rendering a basic triangle—feel free to start from that example, but be sure to cite the original resource and clearly explain any changes you made. For instance, you might improve the code structure by organizing it into separate modules instead of a single flat script, or you could modify the triangle's coordinates or color.

Extending Part II – 15 points

- add more triangles
- add a color attribute and change the color from the application program (JS).
- add some interaction.
- the color, and even the vertex position, can be changed each time through the loop so that it changes. The changes could be random, driven by a math formula, or based on user interaction.

* These are just a few suggested extensions you can try. You are not required to follow them exactly.

Part III – 20 points

Using points and triangles is fundamental in computer graphics, and with these basic elements, you can create highly complex scenes. In this part, you are tasked to draw a 2D scene on campus (or maybe another place in Wichita Falls). For example, the scene can depict a front of a building, complete with generated bricks and their respective colors, as well as windows and doors. The level of detail is up to you, but it should go beyond a simple red rectangle representing Bolin Hall. This exercise will help you practice managing more complex geometry by breaking it down into a set of basic shapes, points, and triangles.

Understanding the geometry of your scene is key—it will also give you insights into how to procedurally generate the scene's elements. It's always a good idea to start with a simple model and gradually increase the complexity as you go.

Additionally, using pencil and paper to sketch the scene is a valuable tool for visualizing and planning your design. Examples from the book can also provide useful insights into the process of point generation and scene creation.

Debugging

Debugging is one of the most essential skills you'll develop in this course. It can be particularly challenging, especially when working with shader programs, as your image may not display on the screen. Here are some tips to help:

- Use the Console Log: When debugging shader programs, check the console for error messages. These will often provide hints about what went wrong.
- Open the Console: Press Ctrl + Shift + J (or Cmd + Option + J on Mac) to open the console window in most browsers. This is where you'll find error messages.

- Work Incrementally: Add small features to your code, and make sure each part is working before moving on to the next. This approach makes it easier to isolate and fix problems.
- Code Organization: As your code grows, ensure you manage it properly. Plan your development process and keep your code clean so it scales well.

Report – 5 points

The report should be done in a word processor. Either a .doc, .docx, or .pdf file will be accepted.

1. Don't forget your name!
2. Provide an estimate of the total number of hours for the entire assignment
3. Results from Parst II and III
4. Write up your experience, including any problems you ran into. Describe what you did for extra credit.
5. List any sources you used
6. Provide snapshots for parts II and III. ***Name the images FirstLastLab2a.jpg, FirstLastLab2b.jpg,*** where *First* and *Last* are **Your** names. The snapshots should also be included in the report.

Submission

All files (all source, your report, and your images) should be put into a single zip file and uploaded. I can handle, .zip,

A zip file **with your final source code for parts II and III, the report, and the images**. ***The images need to be embedded in the report and also as separate files in the zip.***