Assignment 1: Hello World and Then Some

CS148 Fall 2015-2016

Introduction

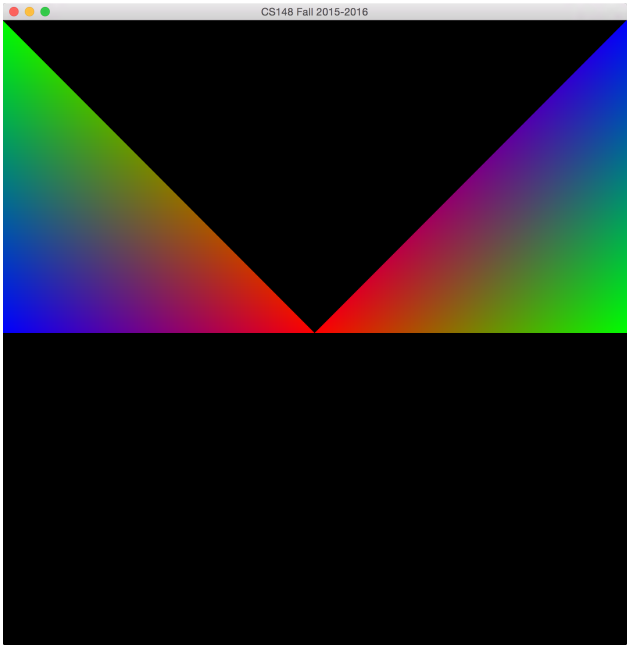

Figure 1: Just the usual OpenGL "Hello World" triangles

If you have not already, make sure you have downloaded the assignemt framework and have gotten it to compile! See the setup guide if you have not done so already. This assignment is meant to give you a gentle introduction to OpenGL 4 and the assignment framework (as well as give you time to work out all the kinks with configuring and compiling for your system!). After completing this assignment, you should:

1. Understand what SDL does with regards to using OpenGL.
2. Be familiar with how SDL processes mouse/keyboard events and how to send them to your program.
3. Know how OpenGL 4 takes a bunch of vertices and displays colors on your screen as well as the role of vertex and fragment shaders in this process.
4. Know your way around GLSL syntax and be comfortable with passing vertex attributes and uniforms to the shader.

The assignment framework is fairly large so we encourage you to start EARLY and ask questions in office hours and/or Piazza if you get stuck and need help.

Useful Resources

1. OpenGL Programming Guide Chapters 1-3 (the book is not required, but recommended!)
2. OpenGL 4 Documentation: here. Note that these pages are for OpenGL 4.5 but most should be equivalent (unless otherwise noted on the function's page).
3. SDL 2 Documentation: here.
4. GLSL Tutorial: here.
5. GLM: Manual and Documentation.
6. C++ Reference: here
7. Assignment Framework Documentation: This can be found in "source/doxygen"

Assignment

Before starting to write your program, you will need to decide what you want to draw using OpenGL. Here are some ideas (and links to examples and definitions in red) to get you started.

- Spirographs or similar functions
- Fractals, drawn to a certain level, see e.g. the Koch Snowflake
- Simple space filling curves
- Simple L-systems, e.g. use them to create simple trees (lines for branches, triangles for leaves)
- Conway's game of life
- A simple arcade game - Pong, Pacman, Snake...
- XScreenSaver has many good examples
- If you are feeling adventurous, try something with animation and/or particle effects, add physics, and/or try adding some user input. These linked examples were done in JavaScript, but you could easily make something similar with OpenGL and some creativity!

Assignment 1 Code Explanation

Look in the Assignment1 class documentation in the assignment framework documentation found at "source/-doxygen".

Shader Loading

Shaders are stored in the "shaders" folder. When the program is compiled, a script is run to copy this folder into the right location. As a result, to load a shader, you only have to specify the shader path relative to the shaders folder. For example, for the vertex shader located at "shaders/basicColor/basicColor.vert", Assignment1.cpp only tells the framework to load the shader from "basicColor/basicColor.vert". If you make new shaders, make sure they are in the shaders folder!

Pointers to Get You Started

- Without transformations, the coordinates of your scene will be equivalent to the "normalized device coordinates" which has the top left corner coordinate as $(-1, 1)$ and the bottom right corner coordinate as $(1, -1)$.
- Look in "source/common/MediaLayer.h/cpp" to see how OpenGL and SDL are setup, in particular the InitializeOpenGL and InitializeSDL functions.

- Look in "source/common/MediaLayer.h/cpp" to see how SDL handles events (i.e. keyboard) in the 'Tick' function. You will need to modify this function if you want to use the mouse in your program! The SDL_Event documentation may be helpful.
- To see how the framework takes in your vertices and whatever else and sets it up to display on the screen look in "source/common/Rendering/RenderingObject.h/cpp". The "UpdateVertexXYZ" functions takes in the raw data and passes it to OpenGL. The BeginRender(), Render(), and EndRender() functions performs setup/drawing/teardown during the rendering loop. Questions about a specific OpenGL function? Look it up in the documentation first (linked in "Useful Resources" above). Still confused? Ask a CA!
- "std::unique_ptr ⟨T⟩" and "std::shared_ptr ⟨T⟩" are special types of pointers introduced in C++11. Using "std::shared_ptr⟨T⟩" gets you reference counting and thus lets you (for the most part) not worry about deallocating memory. Using "std::unique_ptr⟨T⟩" is similar except that you can not make a copy of a unique pointer! The pointer has to be unique! Thus, if you want to "transfer ownership" somewhere else, you will have to use std::move (you will see alot of this in the code) which will invoke the move constructor (should you pass the unique pointer to a function or otherwise assign it). After doing so, the original unique pointer will no longer be valid!
- If you want to modify the shaders, you will want to do so in the "source/shaders/basicColor" folder.

Tips and Tricks

- While developing it will be useful to compile your program in "Debug" mode. This will enable the assertions in the code to help you debug. Additionally, it will enable the OGL_CALL macro to do an OpenGL error check which can provide useful information if nothing is showing up on your screen!
- Want circular points instead of square ones? Look into gl_PointCoord! The "Point Sprites" section of Chapter 6 of the OpenGL Programming Guide (Red Book) is useful too!

Questions

1. Explain how a "Vertex Array Object" and a "Vertex Buffer Object" are related.
2. Which OpenGL calls relate to the generation and usage of a "Vertex Array Object" and which relate to the generation and usage of a "Vertex Buffer Object".
3. How does OpenGL know whether you want to draw a triangle, line, point or other some other primitive?

Feedback (Optional)

1. On a scale of 1-10 [10 being very clear]: How clear were the instructions to configure and compile the assignment framework?
2. On a scale of 1-10 [10 being the most work]: How much work was this assignment (including configuring and compiling the project)?
3. On a scale of 1-10 [10 being very easy to use]: How understandable/easy to use is the assignment framework?
4. How much experience with C++ and OpenGL did you have before taking the course?

Grading

This assignment will be graded on the following requirements

- The program draws using at least two of the OpenGL primitives in at least two different colors.

- The program incorporates some form of user interaction.
- The student is able to adequetely answer questions asked by the CAs.
- The student's vertex or fragment shader uses one (or more) uniforms. (optional)

according to the following rubric.

- + – Exceeds the requirements via one or more artistic/technical contributions
- ✓ – Meets all of the requirements
- − – Does not meet the requirements but still produces a drawing.
- 0 – The submitted solution does not produce a drawing.