CMPT 485/829: Assignment 1 Specification
Due – Sunday January 29, 2012: 11:59pm

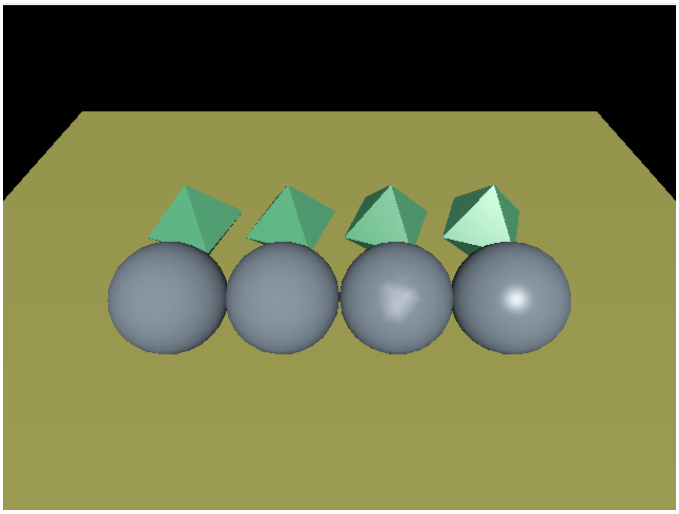

Figure 1: Screen capture from reference solution.

1 Introduction

This assignment is the first part of a multi-assignment mini-project that will span the term. At the very end you will have a single OpenGL program that:

1. Renders an environment in real-time;
2. Has simple camera controls to control where you are looking;
3. Renders shadows;
4. Allows you to ray trace a view with the press of a button; and
5. Incorporates some very basic animation.

In this assignment you are going to focus on the first two items on that list.

2 Framework

You are being provided with a C++ framework for a basic OpenGL renderer. You are expected to use this framework as the basis for your assignment. A diagram of the class heirarchy and some of the use-relationships is provided in figure 2.

Each of these modules/classes is explained in the corresponding header file; if you have questions, please ask. I expect that using this sort of framework will be a very new experience for most students; so, I expect there to be a lot of questions about the framework.

The framework has some internal documentation that tries to explain the OpenGL-related API calls, and how the framework uses OpenGL. I repeat, if you have questions then please ask.

3 Assignment 1

For this assignment, you have four goals:

1. Implement a camera that is implemented using a classical camera frame, can be toggled between orthographic and perspective projection, can be moved around the environment, and can be rotated around each of its axes.
   - Note: If you do any web searching for camera implementations/tutorials, then you will most likely encounter either an "Euler angle" or "quaternion" camera. This camera is to be implemented using neither of those techniques.
   - Your implementation will represent the camera with a camera position, and three vectors indicating view direction, up direction, and direction of right.
   - This part will require that you implement methods in Camera.
2. Implement GLSL shaders to provide coloring to the objects in your scene. There are four shaders in total to be implemented:
   (a) Shader::Constant::Lambertian::Gouraud
   (b) Shader::Constant::Lambertian::Phong
   (c) Shader::Constant::Specular::Gouraud
   (d) Shader::Constant::Specular::Phong

   Descriptions of what the shaders should do can be found in their header files.
3. Implement Object::Models::Plane. A description of the requirements can be found in the class's header file.
4. Create a simple scene that includes a ground plane, four spheres, and four octahedron. Each of your four shaders should be used on one sphere and one octahedron.

You may modify any code present in the framework, and add any new source files that you require.

Figure shows an example screenshot from my solution implementation. The camera settings are the default settings found in the framework, and objects in each column use the same shader. The shaders used, from left to right, are:

- Shader::Constant::Lambertian::Gouraud
- Shader::Constant::Lambertian::Phong
- Shader::Constant::Specular::Gouraud
- Shader::Constant::Specular::Phong

In this screen capture, all geometric objects are rotated by 25 degrees around their y-axis prior to translation.

3.1 CMPT 829 Students

Students enrolled in CMPT 829 have one additional goal:

- You must implement the four shaders:
  1. Shader::Texture::Lambertian::Gouraud
  2. Shader::Texture::Lambertian::Phong
  3. Shader::Texture::Specular::Gouraud
  4. Shader::Texture::Specular::Phong

  And add four more spheres to your scene that each use one of the new shaders.

You will have to add functionality to read in an image file, and have your GLSL programs use the texture (you may use glfwReadImage() for loading an image file [see: www.glfw.org – this is the windowing library used by the framework]).

Your implementation of these shaders should be placed in the "Shaders/Texture" directory of your source tree, and mimic the file structure found in "Shaders/Constant".

These shaders are similar to their "Constant" counterparts except that they extract the vertex/fragment surface reflectance from a 2D texture instead of being provided a constant surface reflectance.

4 A Suggested Approach

The following is a suggested approach to taking on this assignment.

1. Copy the Simple shader into the Constant::Lambertian::Gouraud shader, and get the Gouraud shader to take geometry in camera coordinates and apply a projection matrix to convert to normalized device coordinates. Simply pass an identity matrix to the shader for a projection matrix for now.

2. Implement the Camera class. Begin with the unimplemented setter functions, then the lookAt() function, then implement perspective projection, and finally implement all the movement and rotation functions.

3. Complete the implementation of the Constant::Lambertian::Gouraud shader by adding support for an omni-directional point light source, then add in the modelview matrix.

4. At this point, you can add geometry to your scene, and complete the remaining shaders one-by-one.

5 Deliverables

1. Upload your completed assignment as a zip or tgz file to "Assignment 1" on Moodle. Your source code must compile and run on the Linux machines in S360, and include a Makefile that builds the project.

2. Your zip/tgz file must include a "Readme.txt" file that briefly lists your student information (name, student #, nsid), and the files, functions, and data structures that you modified. For example, if you modify Assignment1::init() and Assignment1::repaint() in assign1.cpp, then you could write the following in your Readme.txt:
   File: assign1.cpp
   - Assignment1::init()
   - Assignment1::repaint()

6 Evaluation

The assignment will be graded subjectively by the marker with the following breakdown:

1. 25% – Code readability/clarity
2. 75% – Implemented to spec

Each component of the assignment is worth the following:

1. Camera – 40 marks
2. Shaders – 8 marks each
3. Plane geometry implementation – 16 marks
4. Scene configuration – 16 marks

The maximum grade for CMPT 485 students is 120 of 104 marks; 829-only components of the assignment may be completed for bonus marks.

The maximum grade for CMPT 829 students is 136 of 136 marks.

Note: Moodle has no way to differentiate students by class in a split class. So, please look at the absolute mark value instead of the percentage grade when looking at your grade.

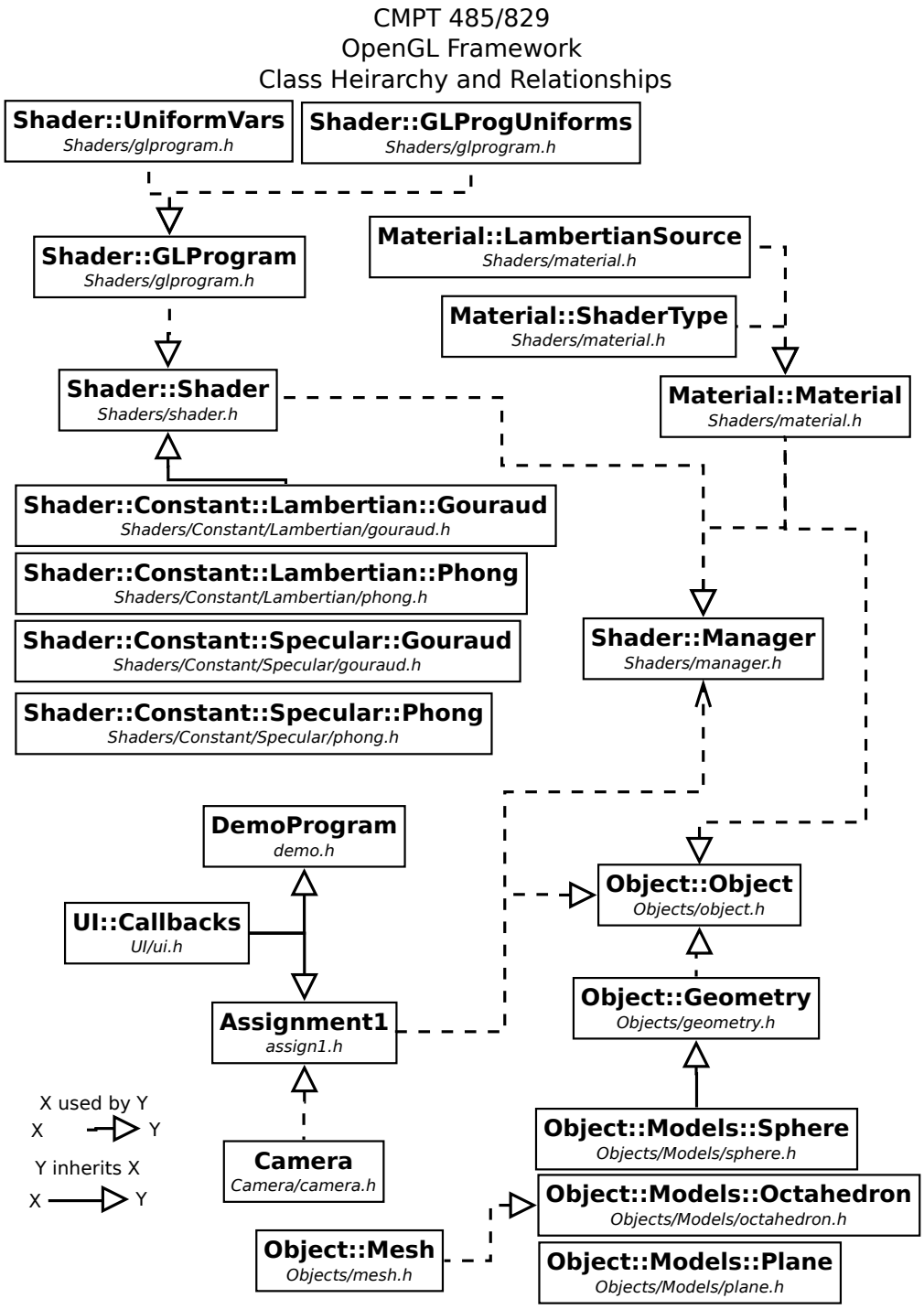

Figure 2: Framework Diagram