CAP 5705 Computer Graphics
Dr. Corey Toler-Franklin

GLSL Programing
DUE: November 3rd, 11:59 pm

Overview | Details | Resources | Getting Help | Submitting

Overview

You may complete this project alone or in groups of two.

Shader programs are essential components of the modern OpenGL pipeline. You will use the OpenGL shader language, GLSL, to implement shader programs that process high level algorithms efficiently using the graphics processing unit (GPU). You will be evaluated on (1) source code completion and correctness 30% (2) reflectance direction calculation 10% (3) cube map 20% (4) texture lookup 20% (5) result image 10% and (6) written report 10%.

Details

Getting Started

Extract the file proj3_glslshaders.zip to your local folder. Feel free to make any changes to the framework to complete the task. The source code framework is in the *src* folder. Datasets (models and images) are found in the *data* folder. Other helpful materials are in the *docs* folder. To build the application on linux, type *make clean* then *make all* in the *src/mainsrc* folder. A visual studio 2017 solution file is also provided in the outer *src* folder. Use any platform but make sure your code compiles and runs on Ubuntu Virtual Box before you submit (see *docs* for instructions). Included are the latest versions of the Simple-OpenGL-Image-Library (SOIL), OpenGL Wrangler Library (Glew), GLFW library, and the OpenGL mathematics library (GLM) . The program is executed from the main function in src/mainsrc/main.cpp.

Environment lighting uses texture-mapping (instead of point lights) to implement complex scene lighting. This is different from other examples that compute reflectance by plugging parameters into a specific lighting equation. To implement a specular (mirror-like) reflection under environment lighting, you will compute a reflection ray at a point, and perform a texture look-up to determine the object's color at the specified point. Implement the vertex and fragment shader programs in *src/mainsrc/Shaders*, and complete functions for building your cube map in main.cpp. Follow the **TO DO:** notes and refer to the lecture notes and section 11.6 of the textbook for more details.

Figure 1: Reflections: Environment Lighting

OpenGL environment mapping uses a cubemap that stores six images on a cube (the environment). The environment light intensity in a certain direction is determined by sampling a point along a 3-D vector that intersects the cube map. To implement the St. Peter's cubemap, initialize the variable *cmapFiles* with the texture files in *data/images* and complete the *InitCubeCoords*, *GenerateCubeMap* and *LoadTextures* functions. The *Shader* class provides infrastructure for compiling, loading and running your shaders. Implement the cubemap shaders in *environment.vert* and *environment.frag*. The vertex shader sets the position of the cube and texture coordinates. The fragment shader uses a sampler object to set the color. Use the *Geometry* class to load the provided teapot model. Implement reflection mapping in *reflection.vert* and *reflection.frag*. The vertex shader sets the model position in relation to the view. Given a viewing direction and a normal direction, the fragment shader computes the specular reflection direction using the GLSL built-in reflectance equation, and sets the color by sampling the cubemap where the reflected ray intersects it. The interior of the cube map is the background. Remember (for realism) to look-up the background environment intensity for rays that do not hit the model. Very basic camera functions are provided. Extend mousecallbacks to rotate the model. Save two views of your reflectance result. Fig. 1 shows an example. Also adjust the teapot position so that it is centered and reasonably scaled in the view.

Resources

The openGL Programming Guide, https://www.opengl.org/sdk/ is a good reference. Appendix A also provides information about GLUT and GLEW. Chapter 8 of the textbook, Fundamentals in Computer Graphics, covers the graphics pipeline.

If you are not in the CISE department, and would like computer lab access, you can register for a CISE account at https://www.cise.ufl.edu/help/account. The source packet proj3_GLSLShaders.zip will run on machines in the computer labs on the first floor of the CSE building. Labs are open 24 hours (see https://www.cise.ufl.edu/help/access). **Remember to back-up your work regularly!!!** Use version control to store your work. DO NOT PUBLISIZE SOLUTIONS.

Getting Help

Source Code Please do not re-invent the wheel. Use the source code framework (and comments) and review the notes in the docs folder.

Discussion Group Post questions to Canvas (everyone benefits in this case), or send me an email at ctoler@cise.ufl.edu. I will check both daily.

Office Hours CSE 332 (or lab CSE 319) MWF 1:30 to 2:30pm.

Collaborating Credit outside sources and follow University policies on academic integrity.

Submitting

Upload your proj3_glslshaders.zip file to Canvas. It should include:

- Source code with make file
- One written report that includes:
  - reflection environment map result (two views)
  - a few lines that explain your implementation
  - anything you would like us to know (how to run your code, bugs, difficulties)
- original reflection environment map result from shader