GPU and GPGPU Programming

Shuai Lu 170742

Assignment 2

Task
1. Setup a GLSL program: create files for the shaders (at least vertex and fragment shaders) and load them at runtime of the program.
Your shaders must include uniform variables for the camera transformation (view and projection matrix), object transformation (model matrix), and the lighting models (see below).

2. Set up camera and object transformations (at least rotation, and zoom for camera and translation for objects) that can be manipulated using the mouse and keyboard [3].

3. Implement Phong lighting+Gouraud shading [1] (the Phong lighting model is evaluated in the vertex shader).

4. Implement Phong lighting+Phong shading [2] (the Phong lighting model is evaluated in the fragment shader, not in the vertex shader as for Gouraud shading).

5. Implement a class that generates and stores the mesh geometry for a) a disc, and b) a cylinder.
Have a look at how the geometry of the cube (VBOCube class) is specified and used.
The classes for the disc and cylinder should be similar. However, instead of hardcoding the attributes (vertex-position, indices for edges, ...), compute them when constructing the class.
The constructor of the disc and the cylinder should have appropriate parameters (like radius, number of triangles, height, ...)

6. Render multiple instances of an object within one scene. Render the same object multiple times, applying different transformations to each instance.
To achieve this you can set a different transformation matrix for each instance as a uniform variable in the vertex shader.

7. Perform different kinds of procedural shading (in the fragment shader):
Implement the following procedural shaders - Stripes described in chapter 11.1 of the 'OpenGL Shading Language' book (this is not the 'OpenGL 4.0 Shading Language Cookbook')
- Lattice described in chapter 11.3 of the 'OpenGL Shading Language' book (this is not the 'OpenGL 4.0 Shading Language Cookbook')
- Toon shading described in chapter 3 section 'Creating a cartoon shading effect' of the 'OpenGL 4.0 Shading Language Cookbook'
- Fog described in chapter 3 section 'Simulating fog' of the 'OpenGL 4.0 Shading Language Cookbook'

8. Provide key mappings to allow the user to switch between different kinds of geometry, as well as shading methods, and to set parameters for the lighting models.

9. Submit your program and a report including the comparison of Phong and Gouraud shading and results of the different procedural shading methods.

BONUS: - implement sphere geometry
- implement (procedural) bump mapping (normal mapping) as in chapter 11.4 of the 'OpenGL Shading Language' book.1. Setup

Outcome:
The GLSL program is setup and related uniform variables are setted successfully.

All shaders are store in the 'shader' folder, including basic shader, Gouraud shader, Phong shader, Stripes shader, Lattice shader, Toon shader and fog shader.

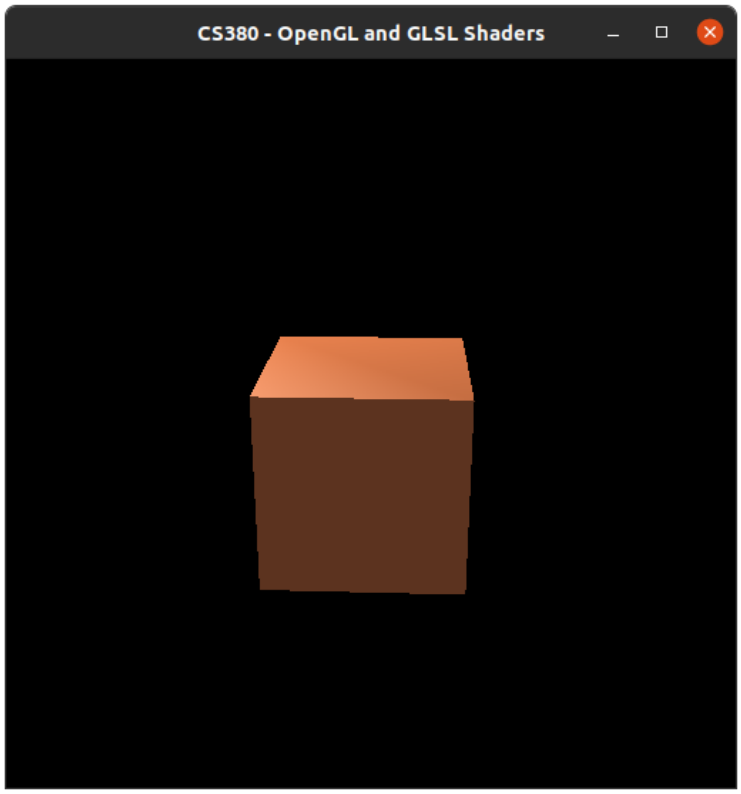

Figure 1: Gouraud shading

The classes for disc, cylinder and sphere are developed.

Multiple instances are rendered within one scene(Fig.6).

Different instances are rendered with different shaders within one scene(Fig.7).

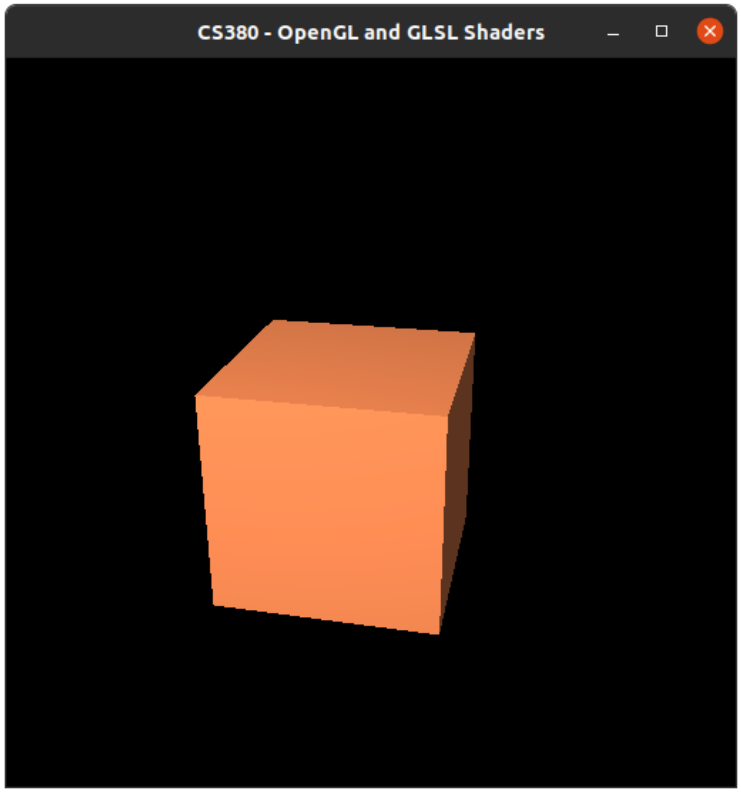

Figure 2: Phong shading

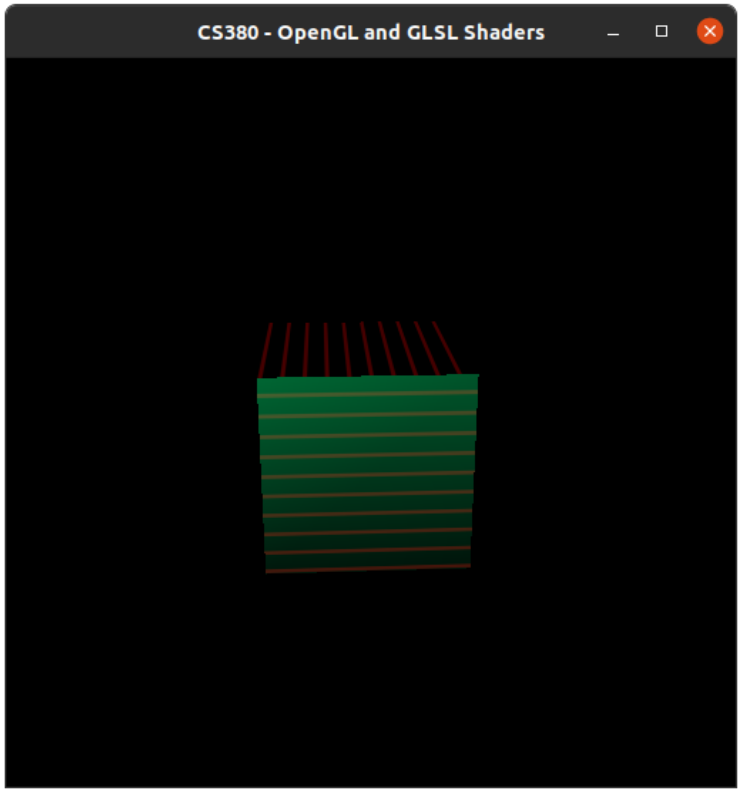

Figure 3: Stripes shading

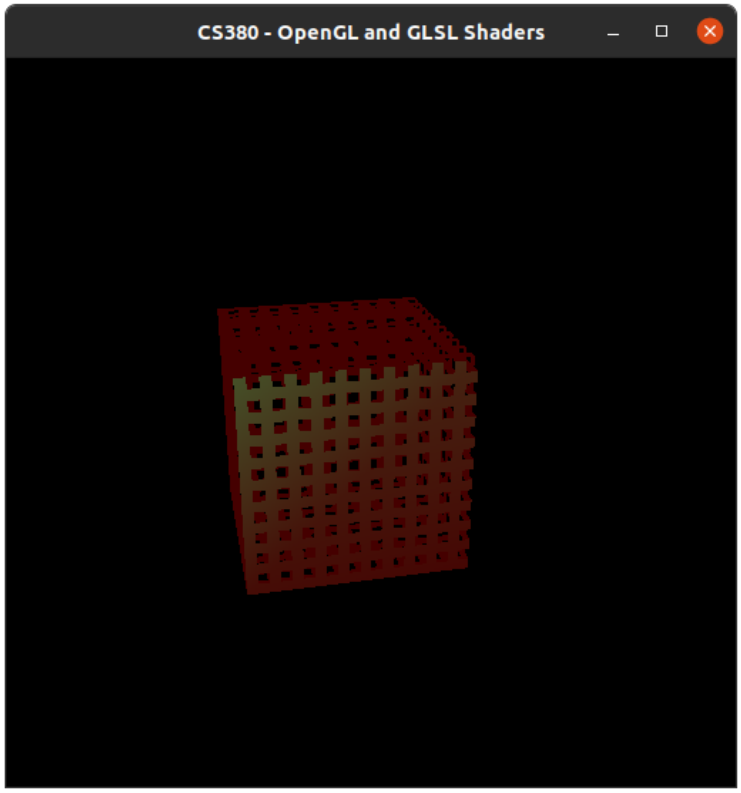

Figure 4: Lattice shading

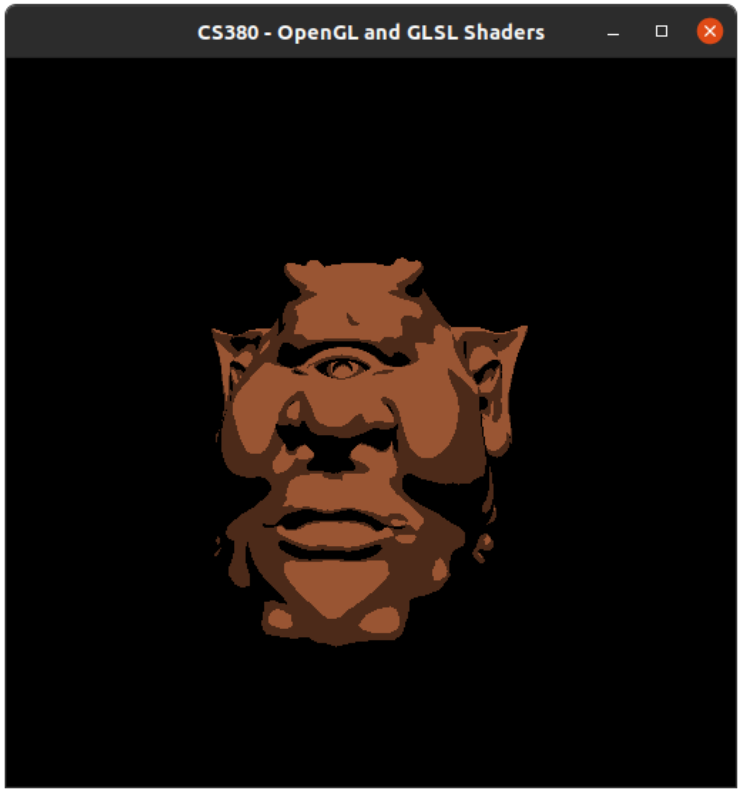

Figure 5: Toon shading

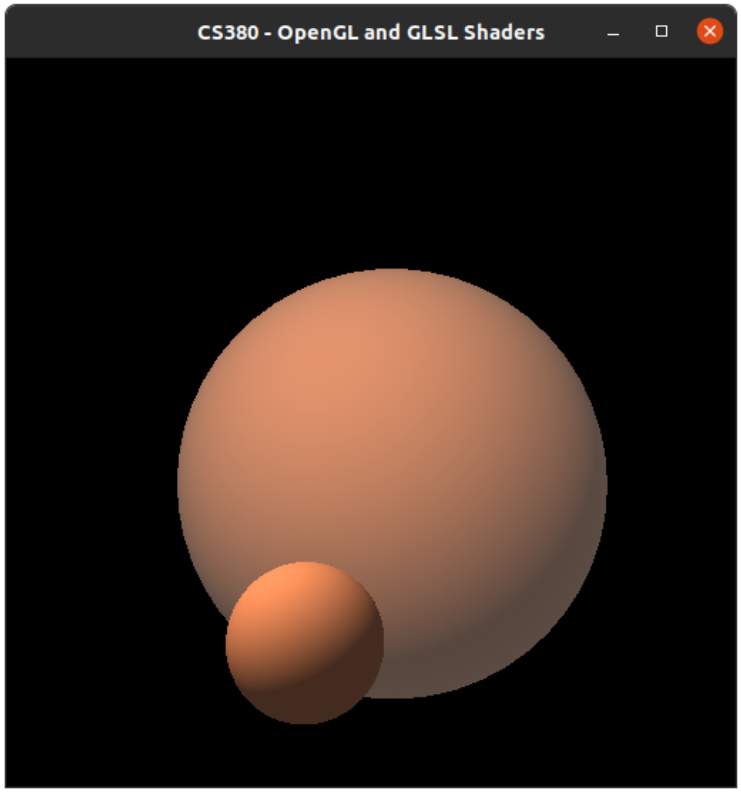

Figure 6: Fog shading

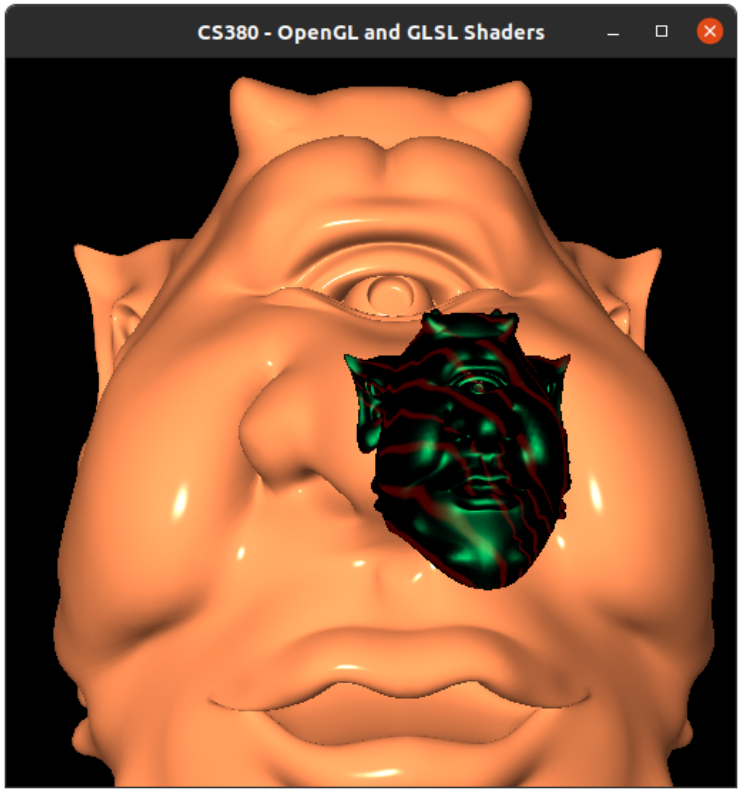

Figure 7: One use Stripes shading, another use Phong shading