CS 450/550 -- Fall Quarter 2020
Project #5

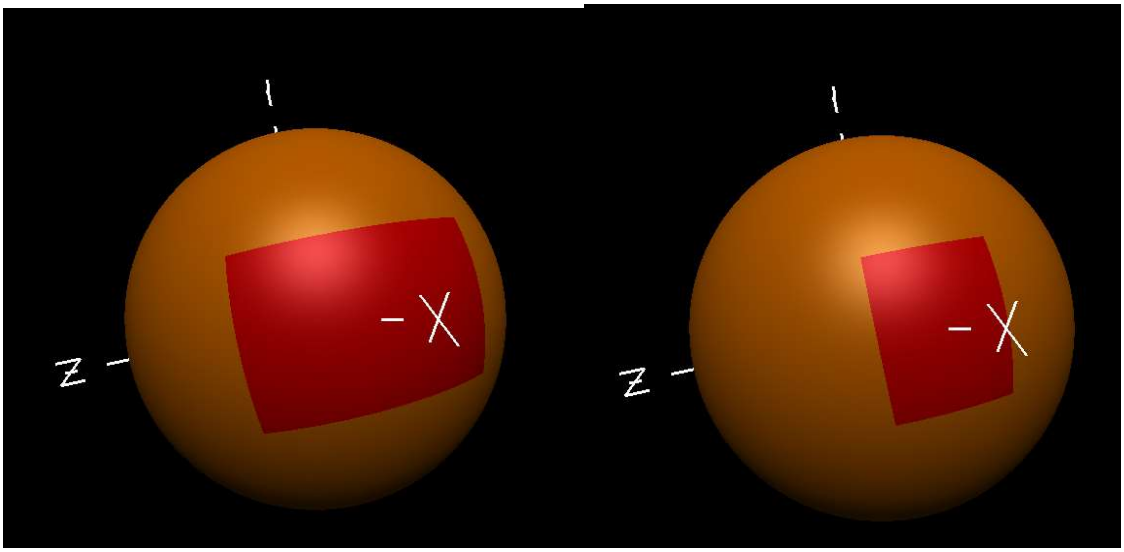
100 Points

Due: November 12 (Note: Nov 11 is a holiday)

Shaders

This page was last updated: July 27, 2020

Introduction:



The goal of this project is to create an animated pattern by using OpenGL vertex and fragment shaders. The choice of pattern is up to you, **but it cannot be a rectangle, because that's what the un-imaginative Joe Graphics showed you how to do.**

Requirements:

1. Draw a 3D object (your choice).
2. Use a vertex shader to place a vertex-changing pattern on it. The pattern must be different for different parts of the object. Key off of the (s,t) coordinates or the (x,y,z) coordinates. Doing the exact same operation to all vertices doesn't count!
3. Use a fragment shader to place a color-changing pattern on it. The pattern make not be a rectangle. The pattern must be different in different parts of the object. Key off of the (s,t) coordinates or the (x,y,z) coordinates. Doing the exact same operation to all fragments doesn't count!

4. The choice of patterns is (almost) up to you (see above).
5. The choice of colors is up to you.
6. Perform some sort of animation (your choice) on the patterns. That is, make the vertex and fragment patterns change with time.
7. You can key off of anything you would like. Typically that is XYZ in model coordinates, XYZ in world coordinates, or ST. Be aware that, in the GLUT library, only the teapot has ST coordinates. The OsuSphere( ) also does.
8. You must have the ability to show both patterns animating at the same time, and the ability to have each pattern animating separately. One way to control this is with keyboard keys:

'b'	Animate both the vertex and fragment shader patterns together
'f'	Freeze both animations
'F'	Animate just the fragment shader pattern, freezing the vertex shader pattern
'V'	Animate just the vertex shader pattern, freezing the fragment shader pattern

The GLSLProgram C++ class

Here are the 2 files you will need:

1. glslprogram.h
2. glslprogram.cpp

See the class Shader notes for how to use them.

```
#include "glslprogram.h"
```

after you

```
#include <GL/gl.h>
```

A Head Start

To help you get started, here are some program skeletons to work from:

- pattern.cpp
- pattern.vert
- pattern.frag

If you want to do some cool-looking per-fragment lighting, here is what you need.

- lighting.vert
- lighting.frag

You can only have one vertex/fragment combination active at any one time. Thus, you need to *integrate the lighting shader code in with the pattern shader code*.

Turning Vertex and Fragment Effects On and Off

You can't do vertex+no-fragment or fragment+no-vertex. You either use the full fixed-function pipeline, or you use a full vertex+fragment shader program.

I recommend that you pass in your pattern parameters as uniform variables and just control when you do it. For example, suppose that uA and uB are your vertex pattern parameters and uC and uD are your fragment shader pattern parameters. Then, in Display( ), you might say:

```
Pattern->Use( );
A = << some constant >>
B = << some constant >>
if( ImUsingTheVertexShader )
{
	A = << some function of Time >>
	B = << some function of Time >>
}
Pattern->SetUniformVariable( "uA", A );
Pattern->SetUniformVariable( "uB", B );

C = << some constant >>
D = << some constant >>
if( ImUsingTheFragmentShader )
{
	C = << some function of Time >>
	D = << some function of Time >>
}
Pattern->SetUniformVariable( "uC", C );
Pattern->SetUniformVariable( "uD", D );

<< do the drawing >>
```

Turn-in:

Use the Teach system to turn in your:

1. Your PDF report, describing what you did and where we can find your video.
2. .cpp file
3. .vert file
4. .frag file

Be sure that your video is flagged as *unlisted*.

Bonus Days:

Each of you has been granted five total Bonus Days, which are no-questions-asked one-day project extensions, but no more than **2** Bonus Days may be applied to any one project.

Grading:

Item	Points
Stationary vertex shader pattern	20
Animated vertex shader pattern	30
Stationary fragment shader pattern	20
Animated fragment shader pattern	30
Potential Total	100