CPSC 314
Assignment 4: Texture Mapping

Due March 25, 11:59pm

1 Introduction

The main goals of this assignment are to explore texture mapping. There are three main texture mapping tasks to complete: apply a texture to the armadillo using its texture coordinates; model the armadillo as a perfectly reflective surface surrounded by an environment cube map; apply a projected texture to the armadillo.

Template: The template code is found in the main assignment directory. It is very similar to the previous template. The main differences are:

- The armadillo `.obj` files are updated to include texture coordinates.
- The template has been updated to support loading textures (currently only supports `.tga`, i.e., targa, images).

You will need to load textures in `main.cpp` and pass them into your shaders, which will apply them using the shader's texture coordinates. Chapter 15 of the textbook has a description of how to write your shaders, including code examples.

2 Work to be done (100 pts)

- **25 pts** Basic Texturing

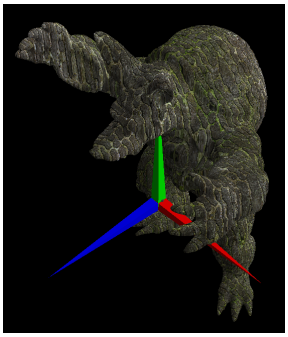

Figure 1: Texture is mapped to the armadillo's body

Load a texture and map it on to the armadillo. To load images into textures, you may use the provided function `LoadTGAFile()`, something like:

```
TGAFILE image[8];
Texture2D tex;
LoadTGAFile("path/to/.tga", &image[0];
tex.loadTexture(GL_TEXTURE0,
                texture_info(image[0].imageWidth,
                             image[0].imageHeight),
                image[0].imageData);
w_state->textures.push_back(tex);
```

The array of TGAFILEs will hold all your `.tga` texture files, which you will use to create Texture2D objects via the `loadTexture()` function. Then you will need to push them onto the `textures` vector. In `main.cpp`, add `w_state->loadTextures();` before `g_mesh->drawMesh();`. This loads all the textures into each shader. They will be stored as a uniform `sampler2D` object (declared as `uniform sampler2D TexSampler0`, `TexSampler1`, etc., based on the index of the desired texture). So in any of your fragment shaders you can access any of your textures by telling them to use the appropriate uniform `sampler2D` object.

The beginning of Chapter 15 of the textbook discusses how to write your shaders with sample code. You are free to use any texture you'd like, but we have also provided some in the the directory `CMaps` for your convenience. If you use your own texture, have the texture image handy when being graded se we can be sure it is being mapped correctly.

- **25 pts** Environment Cube Map.

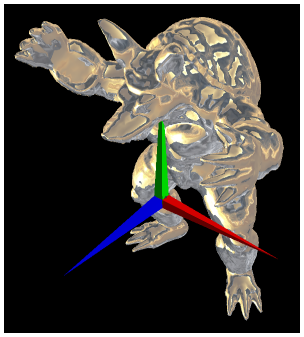

Figure 2: Armadillo reflects surrounding cube map

For this part of the assignment, make the armadillo a perfectly reflective surface, like an armadillo-shaped mirror. Conceptually, we do this by surrounding the armadillo with a cube that has a texture mapped on to each side so that the cube is reflected in the armadillo. Lecture 23 and Chapter 15 of the textbook have a description and some code to show how to write your shaders. In `main.cpp`, the calls are very similar but slightly different:

```
TextureCubeMap texCube;

LoadTGAFile("path/to/one/cube/side.tga", &image[1]);
texCube.loadTexture0(GL_TEXTURE1,...);

LoadTGAFile("path/to/next/cube/side.tga",&image[2]);
texCube.loadTexture1(GL_TEXTURE1,...);

...

w_state->textures.push_back(texCube);
```

Again, we will provide a cube map but you are free to use your own, and again be sure to have the image handy when being graded so we can be sure you are using it correctly.

- **20 pts** Projector Texture Map

Figure 3: UBC logo is projected onto the armadillo along the z-axis

Projector texture mapping is exactly what its name suggests - you project a texture image onto a surface like batman shines the bat-symbol onto the clouds, only now the bat-symbol is any image you like, and the clouds are the armadillo. The nice part of this texture technique is that you don't need texture coordinates. The more complicated part is that it will require projection matrices. Lecture 23 and Chapter 15 of the textbook have a description and some code to show how to do this. In `main.cpp`, the calls will be just like the first texture:

```
Texture2D texProj;

LoadTGAFile("path/to/.tga", &image[7]);
texProj.loadTexture0(GL_TEXTURE2,...);

w_state->textures.push_back(texProj);
```

Again, we will provide a texture but you are free to build the shaders, and again, be sure to have the image handy when being graded so we can be sure you are using it correctly.

- **30 pts** Creative License

  For this part we want to see what you can do. You know the drill. Note that if you add your own meshes/geometries to the scene and you want to apply textures to them, your `.obj` files will need to have texture coordinates. Some possible suggestions to include might be:

  - Combine texture mapping with the Phong shading you did in Assignment 3, by interpreting texture values as reflectances.
  - Use a 'dynamic' texture (maybe have 2 or more textures that you linearly interpolate using the clock time).
  - Render the environment cube map around the reflective armadillo like the image in the textbook. This technique is know as the "sky box".

- (Challenging) Create a shadow map from a light source and add it to the scene. This is described in the book but no sample code is given.

Bonus marks may be given at the discretion of the marker for particularly noteworthy explorations.

Hand-in Instructions: You do not have to hand in any printed code. Create a README.txt file that includes your name, student number, and login ID, and any information you would like to pass on to the marker. Create a folder called "assn4" under your "cs314" directory. Within this directory put all the source files, your makefile, and your README.txt file. Do not use further sub-directories. The assignment should be handed in with the exact command:

```
handin cs314 assn4
```