Student Name: Stephanie Cabrera **ID:** 861288909

CS 130 - Lab 6: Texture Mapping

Introduction:

Texture mapping in GLSL consists of 3 parts

1. **Uploading a texture:** Handled in OpenGL program (C/C++) part. In a typical OpenGL program, textures are read from an image file (.png,. tga etc.) and loaded in to OpenGL. The parameters of the texture such as interpolation methods are also set in the program.
2. **Computing the texture coordinate of a vertex:** In GLSL, the texture coordinate **glTexCoord[i]** for a texture *i* and a vertex is determined in the vertex shader. This is the coordinate of the vertices corresponding texture positions in the image data of texture *i*, where *i* is the index of a texture (in case of multiple textures – *i* = 0 for a single texture).
3. **Getting the texture color for a fragment**: The texture coordinate, **glTexCoord[i]**, of texture i is readily interpolated to the fragment location by opengl. A lookup function such as **texture2D** is used to get the color from the texture.

Part I: Uploading a Texture

Read the tutorials about uploading a texture file in these links and answer the questions.

1. https://www.gamedev.net/articles/programming/graphics/opengl-texture-mapping-an-introduction-r947
2. http://www.opengl-tutorial.org/beginners-tutorials/tutorial-5-a-textured-cube/#how-to-load-texture-with-glfw (Until section "How to load texture with GLFW")

Question 1: Describe briefly with your own words each one of the following functions. Look at the OpenGL documentation for reference.
Llink: https://www.khronos.org/registry/OpenGL-Refpages/gl4/
Google: "opengl 4 references"

`glGenTextures`: Generate texture names

Inputs: (GLsizei n, GLuint * textures)

`glBindTexture`: bind a named texture to a texturing target

Inputs: (GLenum target, GLuint texture)

`glTexParameter`: set texture parameters

Inputs: (GLenum target, GLenum pname, GLfloat param)

Student Name: Stephanie Cabrera **ID:** 861288909

`glTexImage2D:` Specify a two-dimensional texture image

Inputs: GLenum target, GLint level, GLint internal format, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const void *data

Question 2: Answer the question below, briefly. *Hint: see glTexParameter's reference page.*

a. What do minifying and magnifying mean?

Minifying is used to lower resolution as determined by the level-of-detail (funct
Maxifying sets the index of the highest defined mipmap level, This is an integer value (1000).

b. What parameter name should be used in glTexParameter function in order to specify minifying function?

pname = GL_TEXTURE_MIN_FILTER

c. What parameter name should be used in glTexParameter function in order to specify magnifying function?

Pname = GL_TEXTURE_MAG_FILTER

d. What are the possible minifying and magnifying functions defined by opengl?

Answer: GL_LINEAR, GL_NEAREST, GL_NEAREST_MIPMAP_NEAREST, GL_LINEAR_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_LINEAR

Student Name: Stephanie Cabrera **ID:** 861288909

Question 3: Read the comments and fill out the code accordingly.

```
// ========================================================================
// Inputs:
//    data: a variable that stores the image data in "unsigned char*" (GL_UNSIGNED_BYTE) type
//    height: an integer storing the height of the image data
//    width: an integer storing the height of the image data
// Description:
//    A piece of code that uploads the image "data" to opengl
// ========================================================================

GLuint texture_id = 0;
// generate an opengl texture and store in texture_id variable
glGenTextures(1, texture_id);
// set/"bind" the active texture to texture_id
glBindTexture(GL_TEXTURE_2D, texture_id);
//Set the magnifying filter parameter of the active texture to linear
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
//Set the minifying filter parameter of the active texture to linear
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
//Set the wrap parameter of "S" coordinate to GL_REPEAT
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
//Set the wrap parameter of "T" coordinate to GL_REPEAT
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
//Upload the texture data, stored in variable "data" in RGBA format
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA,
GL_UNSIGNED_BYTE, data);
```

Student Name: **ID:**

Part II: Shading with Textures in GLSL

Read the tutorials below and answer the following questions

https://www.opengl.org/sdk/docs/tutorials/ClockworkCoders/texturing.php
Introduction section only
http://www.lighthouse3d.com/tutorials/glsl-12-tutorial/simple-texture/

1. Fill out the blanks in the vertex and fragment shaders below to compute the `gl_TexCoord[0]` using `gl_TextureMatrix[0]` and `glMultiTexCoord`.

vertex.glsl:

```
varying vec3 N;
varying vec4 position;
//create a uniform 2D texture sampler variable, with name "tex";

uniform sampler2D tex

void main() {
    // compute the gl_TexCoord[0] using gl_TextureMatrix[0] and
    // glMultiTexCoord

    gl_TexCoord[0] = glMultiTexCoord;
    N = gl_NormalMatrix * gl_Normal;
    gl_Position = gl_ProjectionMatrix * gl_ModelViewMatrix * gl_Vertex;
    position = gl_ModelViewMatrix * gl_Vertex;
}
```

fragment.glsl:

```
// create a uniform 2D texture sampler variable, with name "tex", so that
// it will be forwarded from the vertex shader

uniform sampler2D tex

void main() {
    //get the texture color from "tex", using a texture lookup function,
    //and s,t coordinates of gl_TexCoord[0].

    vec4 tex_color = texture2D(tex, gl_TexCoord[0].st)
    // set gl_FragColor to tex_color
    gl_FragColor =tex_color;
}
```