Student Name: **ID:**

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 a texture array

`Inputs:` Number of textures and array of texture names that are stored

`glBindTexture:` Binds a texture to a target

`Inputs:` The target to which the texture is binding and the texture name

`glTexParameter:` Returns the texture parameter

`Inputs:`
target: the target to which the texture is bound to
texture: the name of the texture
pname: symbolic name of a texture
param: the return variable that stores texture prams

Student Name: **ID:**

`glTexImage2D:` Allow shaders to read an array of image as textures

`Inputs:`

target: target texture

level: LOD number

internalFormat: number of color components in the texture

width and height of the image

border: must be 0

format: format of pixel data

type: data type of pixel data

data: pointer to image data

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

a. What do minifying and magnifying mean?

A single pixel on screen can correspond to either a single texel or a collection of texels.
When object moves closer to screen, multiple screen pixels are mapped to one texture texel, this is called magnify
And similarly, when object moves further away, called minify

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

GL_TEXTURE_MIN_FILTER

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

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: **ID:**

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, imageWidth, imageHeight,
                      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";


_____________________________________

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

    gl_TexCoord[0] = ____________________________________________________;
    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


_____________________________________

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

    vec4 tex_color = ____________________________________________________
    // set gl_FragColor to tex_color
    gl_FragColor =tex_color;
}
```

Student Name: **ID:**

Part III: Texture mapping coding practice

In this lab you will be practicing texture mapping with GLSL with a skeleton code.

The skeleton code has texture files (.tga images) monkey.tga and monkey_occlusion.tga, as well as the base code for creating an opengl window, loading a monkey model and drawing it.

Follow the steps below and implement texture mapping in the skeleton.

Step 0. Download the skeleton code from the lab webpage and unzip/untar it in a local directory in the lab machines. Open the image files and observe their content.

Step 1. Uploading monkey.tga to opengl:

- Locate the TODO section towards the end of the loadTarga function in **application.cpp**
- Use the code in the answer to Part I Question 3, to upload “data” to opengl.
  Note 1: the code at the beginning of the function reads the image file in “filename” to the “data” variable.
  Note 2: monkey.tga is in RGBA format, and the data is a pointer to UNSIGNED_BYTE array.

Step 2. Computing texture coordinate

- Locate vertex.glsl and compare the code with the vertex shader code in Part II Question 2.
- Note that the code that computes the texture coordinate of the vertex is already computed, and so you have nothing to do and may go to step 3.

Step 3. Computing the color of the fragment using texture color.

- Locate fragment.glsl and compare the code with the phong shader you implemented, in the previous part. This is a phong shader without a specular part.
- Now compute the texture color just like in the fragment shader in Part II Question 3 and store it in a tex_color variable. But, do not assign it to gl_FragColor.
- Here we would like to use the texture color instead of the material color (glFrontMaterial.diffuse.rgb), while keeping the rest of the computation.
- So, rewrite the line that computes the gl_FragColor accordingly.