

Computer Graphics Assignment #3

I. Implementation of key mapping

- Z/X: switch the model
- T: switch to translation mode
- S: switch to scale mode
- R: switch to rotation mode
- G: switch the magnification texture filtering mode between **nearest** / **linear sampling**
- B: switch the minification texture filtering mode between **nearest** / **linear_mipmap_linear sampling**
- Texture transform on some Pokemon models' eyes
 - right arrow key: Apply change on normal order (1-7)
 - left arrow key: Apply change on reverse order (7-1)

II. Implementation and explanation

- In Render Scene

glActiveTexture → glBindTexture → glTexParameteri (ex. GL_TEXTURE_MAG_FILTER, GL_TEXTURE_MIN_FILTER) → glTexParameteri (ex. GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T)

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, models[cur_idx].shapes[i].material.diffuseTexture);

if (mag_mode) {
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
} else if (!mag_mode) {
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
}

if (min_mode) {
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
} else if (!min_mode) {
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
}

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

glBindTexture(GL_TEXTURE_2D, models[cur_idx].shapes[i].material.diffuseTexture);

glDrawArrays(GL_TRIANGLES, 0, models[cur_idx].shapes[i].vertex_count);
```

```
GLuint LoadTextureImage(string image_path)
{
    int channel, width, height;
    int require_channel = 4;
    stbi_set_flip_vertically_on_load(true);
    stbi_uc *data = stbi_load(image_path.c_str(), &width, &height, &channel, require_channel);
    if (data != NULL)
    {
        GLuint tex = 0;

        // [TODO] Bind the image to texture
        // Hint: glGenTextures, glBindTexture, glTexImage2D, glGenerateMipmap
        // hw3
        glGenTextures(1, &tex);
        glBindTexture(GL_TEXTURE_2D, tex);
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, width, height, 0, GL_RGBA, GL_UNSIGNED_BYTE, data);
        glGenerateMipmap(GL_TEXTURE_2D);

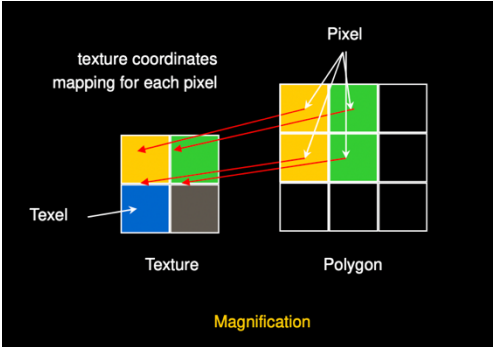
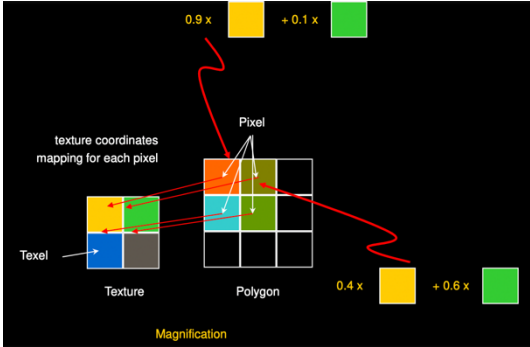


        // free the image from memory after binding to texture
        stbi_image_free(data);
        return tex;
    }
}
```

- Magnification texture filtering mode
 - Implement by glTexParameteri (GL_TEXTURE, GL_TEXTURE_MAG_FILTER, ...)

```

case GLFW_KEY_G:
    cout << "nearest / linear sampling" << endl;
    mag_mode = !mag_mode;
    break;

```

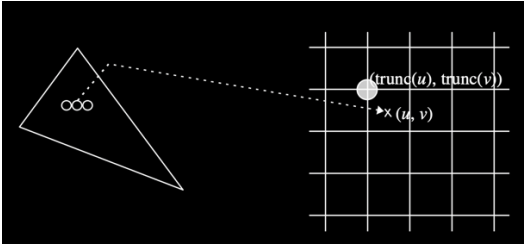
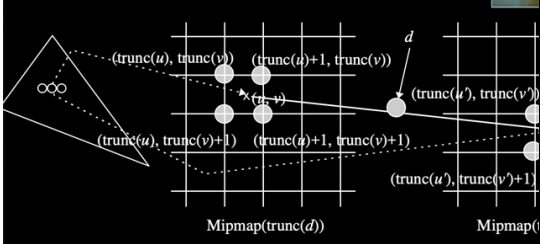


Nearest sampling	Linear sampling
Select the texel nearest the texture coordinate, and this is commonly called “point sampling”.	Perform a weighted linear blend between the nearest adjacent samples.
	
	

- Minification texture filtering mode
 - Implement by glTexParameteri (GL_TEXTURE, GL_TEXTURE_MIN_FILTER, ...)

```

case GLFW_KEY_B:
    cout << "nearest / linear_mipmap_linear sampling" << endl;
    min_mode = !min_mode;
    break;

```

Nearest sampling	Linear mipmap linear sampling
Select the nearest point.	Select between multiple mipmaps based on the angle and size of the texture relative to the screen, and blend between two different mipmap levels
	
	

- Texture transformation
 - Check whether this model has eye and set its offset as our eyes texture
 - GLSL's built-in texture function that takes as its first argument a **texture sampler** and as its second argument the corresponding **texture coordinate**
 - Change the texture coordinate in vertex shader by two global shader variables.

```
// left/right
case GLFW_KEY_RIGHT:
    eye_idx = (eye_idx + 1) % 8;
    break;
case GLFW_KEY_LEFT:
    eye_idx = (eye_idx - 1 + 8) % 8;
    break;
```

```
for (int i = 0; i < models[cur_idx].shapes.size(); i++)
{
    glBindVertexArray(models[cur_idx].shapes[i].vao);

    // hw3
    glUniform3fv(iLocKa, 1, &(models[cur_idx].shapes[i].material.Ka[0]));
    glUniform3fv(iLocKd, 1, &(models[cur_idx].shapes[i].material.Kd[0]));
    glUniform3fv(iLocKs, 1, &(models[cur_idx].shapes[i].material.Ks[0]));
    glUniform1i(iLocisEye, models[cur_idx].shapes[i].material.isEye);
```

```
uniform int iLocisEye; // whether is eye texture
uniform int eye_idx; // current eye idx

void main()
{
    if (iLocisEye == 1){
        if (eye_idx == 0)
            texCoord = vec2(0, 0.75);
        else if (eye_idx == 1)
            texCoord = vec2(0, 0.5);
        else if (eye_idx == 2)
            texCoord = vec2(0, 0.25);
        else if (eye_idx == 3)
            texCoord = vec2(0, 0);
        else if (eye_idx == 4)
            texCoord = vec2(0.5, 0.75);
        else if (eye_idx == 5)
            texCoord = vec2(0.5, 0.5);
        else if (eye_idx == 6)
            texCoord = vec2(0.5, 0.25);
        else if (eye_idx == 7)
            texCoord = vec2(0.5, 0);
        } else {
            texCoord = aTexCoord;
        }
```

- Example



III. Some screen shot

