Computer Graphics Homework 3

Goal :

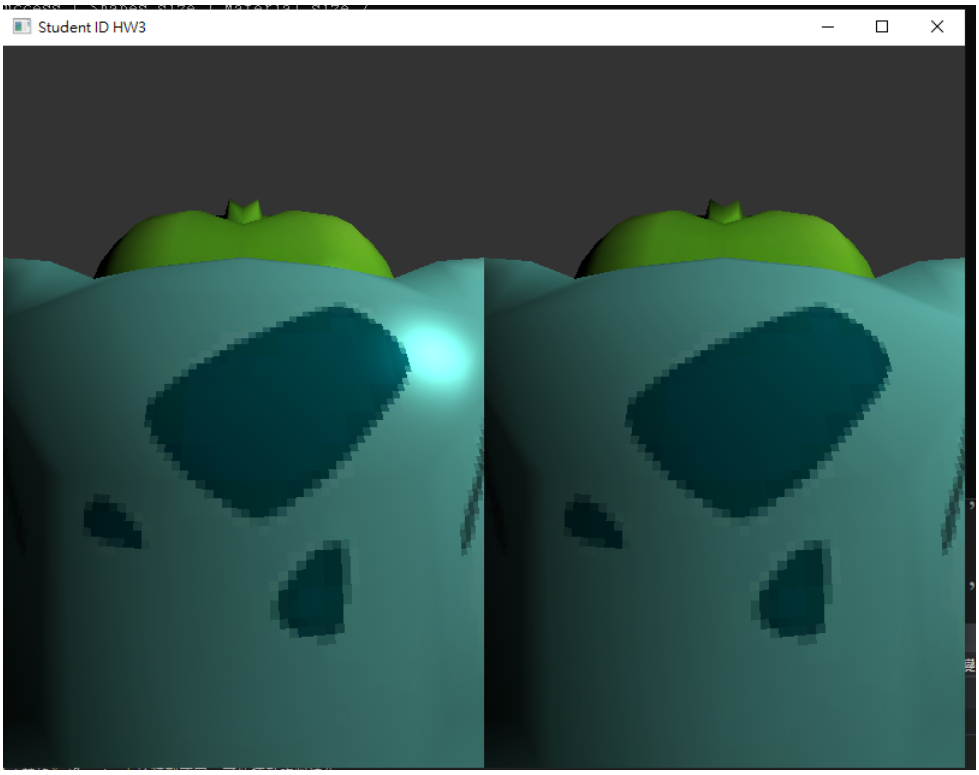
1. Modifying the texture filtering & wrapping mode.
2. Binding and passing the texture to shader.

Function Demo

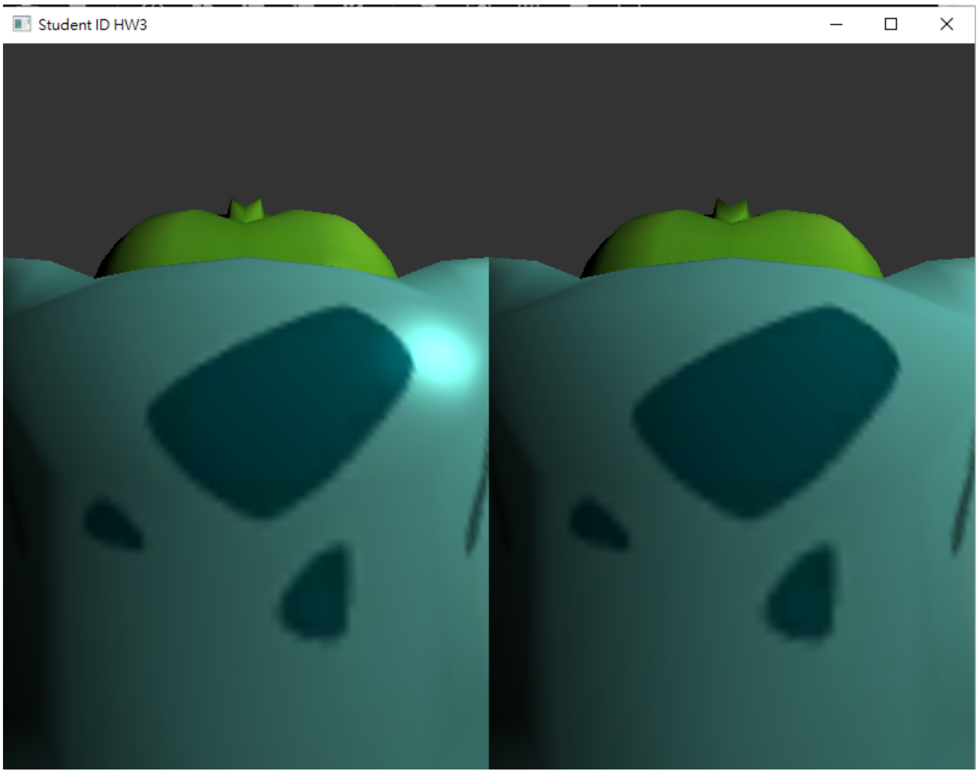
G: switch the `magnification` texture filtering mode

between **nearest / linear sampling**.

Mode: `GL_NEAREST`



Mode: `GL_LINEAR`



B: switch the minification texture filtering mode

between **nearest / linear_mipmap_linear** sampling.

Mode: GL_NEAREST

Mode: GL_LINEAR

細節介紹

`main.cpp`

```
// Global variable
// --- texture attribute
bool mag_mode_is_nearest = true;
bool min_mode_is_nearest = true;

// uniforms texture
GLuint iLocTexture;
GLuint iLocOffset, iLocIsEye;

// binding
void setUniformVariables()
{
    // model parametet and light parameter
    // ...
    // [TODO] Get uniform location of texture
    iLocTexture = glGetUniformLocation(program, "ourTexture");
    iLocIsEye = glGetUniformLocation(program, "isEye");
    iLocOffset = glGetUniformLocation(program, "eyeOffset");
    // ...
}
```

Load Texture mode：

這邊有一點要注意的就是 `glTexImage2D` ，助教給的必須第三個 parameter 告訴 OpenGL 希望把 Texture 儲存為何種格式。原本在寫的時候，以為是GL_RGB，結果出來的 texture 就會是網格網格狀的，後來才發現原來是這邊在讀的時候，格式寫錯～

```
GLuint LoadTextureImage(string image_path)
{
    // ...
    stbi_uc *data = stbi_load(image_path.c_str(), &width, &height, &channel, requi
    if (data != NULL)
    {
        GLuint tex = 0;

        // [TODO] Bind the image to texture
        // Hint: glGenTextures, glBindTexture, glTexImage2D, glGenerateMipmap
        glGenTextures(1, &tex);
        glBindTexture(GL_TEXTURE_2D, tex);
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA32F, width, height, 0, GL_RGBA, GL_U
        glGenerateMipmap(GL_TEXTURE_2D);

        // free the image from memory after binding to texture
        stbi_image_free(data);
        glBindTexture(GL_TEXTURE_2D, 0);
        return tex;
    }
    // ...
}

void LoadTexturedModels(string model_path)
{
    // ...
    for (int i = 0; i < materials.size(); i++)
    {
        // ...
        // set Eye parameter
        if (materials[i].diffuse_texname.find("EyeDh") != std::string::npos)
            material.isEye = 1;
        else
            material.isEye = 0;
        // ...
    }
    // ...
}
```

KeyCallback & RenderScene:

```
// Call back function for keyboard
void KeyCallback(...)
{
    if (action == GLFW_PRESS) {
        switch (key)
        {
        // switch magnification texture mode
        case GLFW_KEY_G:
            mag_mode_is_nearest = !mag_mode_is_nearest;
            cout << "mag_mode_is_nearest: " << mag_mode_is_nearest << endl;
            break;
        // switch minification texture mode
        case GLFW_KEY_B:
            min_mode_is_nearest = !min_mode_is_nearest;
            cout << "min_mode_is_nearest: " << min_mode_is_nearest << endl;
            break;
        // switch to next image
        case GLFW_KEY_RIGHT:
            models[cur_idx].cur_eye_offset_idx += 1;
            models[cur_idx].cur_eye_offset_idx %= models[cur_idx].max_eye_offset;
            break;
        // switch to previous image
        case GLFW_KEY_LEFT:
            models[cur_idx].cur_eye_offset_idx -= 1;
            if (models[cur_idx].cur_eye_offset_idx < 0)
                models[cur_idx].cur_eye_offset_idx = models[cur_idx].max_eye_offse
            break;
        }
}

void RenderScene(int per_vertex_or_per_pixel) {

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

        // [TODO] Bind texture and modify texture filtering & wrapping mode
        // Hint: glActiveTexture, glBindTexture, glTexParameteri

        glActiveTexture(GL_TEXTURE0); // before binding texture, Activate texture
        glBindTexture(GL_TEXTURE_2D, models[cur_idx].shapes[i].material.diffuseTex

        // set repeat mode for texture coordinate
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

        // magnification mode
        if (mag_mode_is_nearest)
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        else
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

        // minification mode
        if (min_mode_is_nearest)
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST_MIPMA
        else
            glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP

        glUniform1i(iLocTexture, 0);

        // Eye Loading
        float offset_x = (models[cur_idx].cur_eye_offset_idx / 4) * 0.5;
        float offset_y = 1 - (models[cur_idx].cur_eye_offset_idx % 4) * 0.25;
        glUniform2f(iLocOffset, offset_x, offset_y);
        glUniform1i(iLocIsEye, models[cur_idx].shapes[i].material.isEye);


        // Light
        // vertex or per-pixel lighting
        // ...

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

這邊在 init Parameter 中，有更改燈的 position，為了讓看放大效果更明顯，不然 shadow 會太重：

```
// Directional light
light[0].position = Vector3(1.0f, 1.0f, 2.0f);
light[0].direction = Vector3(0, 0, 0);
```

vertext shader

```
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aColor;
layout (location = 2) in vec3 aNormal;
layout (location = 3) in vec2 aTexCoord;

out vec2 texCoord;
out vec4 vertex_color;
out vec3 vertex_normal;
out vec3 vertex_view;

uniform mat4 um4p;
uniform mat4 um4v;
uniform mat4 um4m;
uniform int light_type;
uniform Material material;
uniform Light light[3];

// [TODO] passing uniform variable for texture coordinate offset
uniform vec2 eyeOffset;
uniform int isEye;

// .. 省略光 method

void main()
{
        vec4 vertex = um4v * um4m * vec4(aPos.x, aPos.y, aPos.z, 1.0);
        vec4 normal = transpose(inverse(um4v * um4m)) * vec4(aNormal, 0.0);

        vertex_view = vertex.xyz;
        vertex_normal = normal.xyz;

        vec3 N = normalize(vertex_normal);
        vec3 V = -vertex_view;

        vertex_color = vec4(0, 0, 0, 0);

        if(light_type == 0)
                vertex_color += directionalLight(N, V);
        else if(light_type == 1)
                vertex_color += pointLight(N, V);
        else if(light_type == 2)
                vertex_color += spotLight(N, V);

        // [TODO]
        texCoord = aTexCoord;
        if (isEye == 1)
                texCoord = aTexCoord + eyeOffset;
        else
                texCoord = aTexCoord;
        gl_Position = um4p * um4v * um4m * vec4(aPos, 1.0);
}
```

fragment shader

```
in vec2 texCoord;
in vec4 vertex_color;
in vec3 vertex_normal;
in vec3 vertex_view;

out vec4 fragColor;


uniform mat4 um4v;
uniform Material material;
uniform int light_type;
uniform Light light[3];
uniform int is_perpixel;

// [TODO] passing texture from main.cpp
// Hint: sampler2D
uniform sampler2D ourTexture;

void main() {
        // [TODO] sampleing from texture
        // Hint: texture
        vec3 N = normalize(vertex_normal);
        vec3 V = - vertex_view;
        vec4 color = vec4(0, 0, 0, 0);

        if(light_type == 0)
                color += directionalLight(N, V);
        else if(light_type == 1)
                color += pointLight(N, V);
        else if(light_type == 2)
                color += spotLight(N, V);

        if(is_perpixel == 0)
                fragColor = texture(ourTexture, texCoord) * vertex_color;
        else
                fragColor = texture(ourTexture, texCoord) * color;
}

```

Reference

紋理 - LearnOpenGL-CN (https://learnopengl-cn.readthedocs.io/zh/latest/01%20Getting%20started/06%20Textures/)