Computer Graphics Assignment 3

107062313 黃寶萱

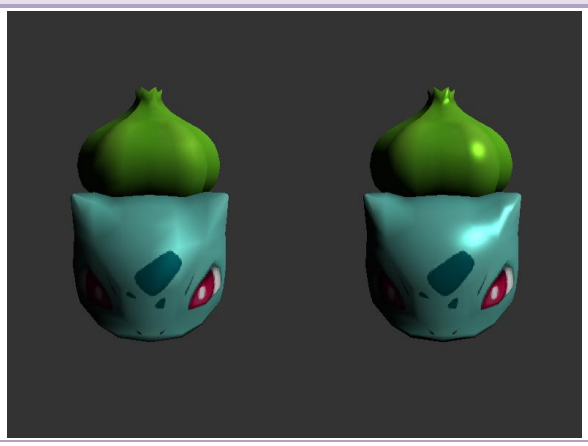
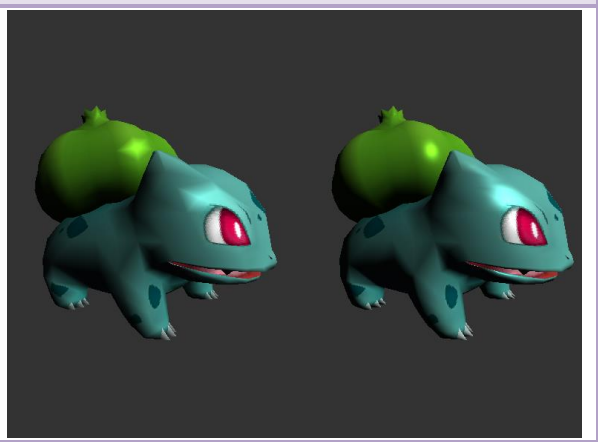
I. Grading Policy

Item	Done
Textured model rendered	✓
Magnification texture filtering mode switch	✓
Minification texture filtering mode switch	✓
Texture transform	✓
Report	✓

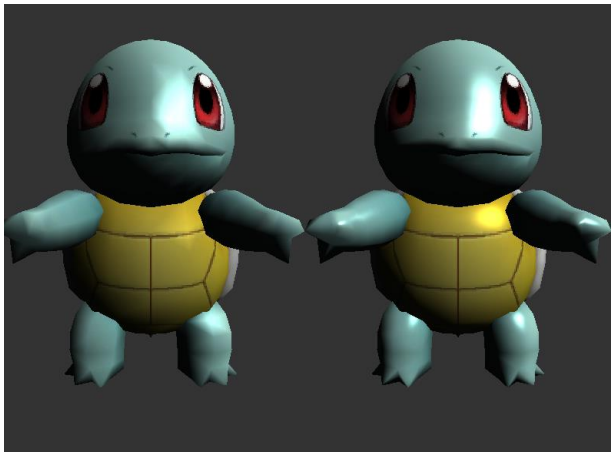
II. Keyboard Mapping

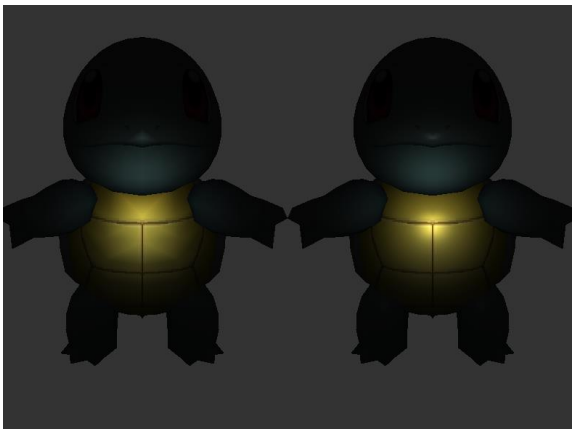
Keyboard_Z & Keyboard_X：控制切換不同的 model	
	

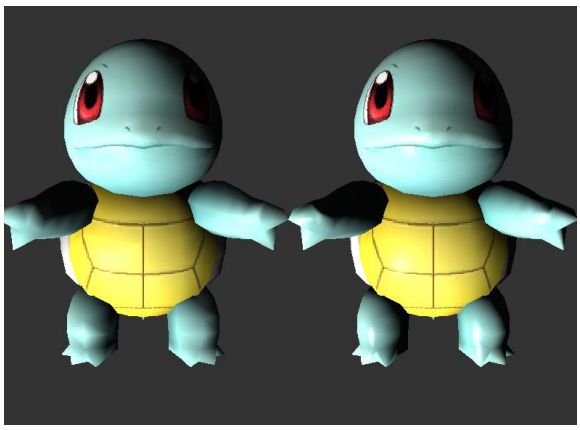
Keyboard_T：切換到 translation mode	
	

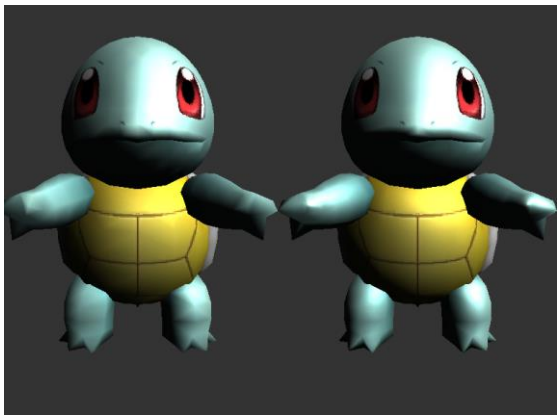
Keyboard_R：切換到 rotation mode	
	

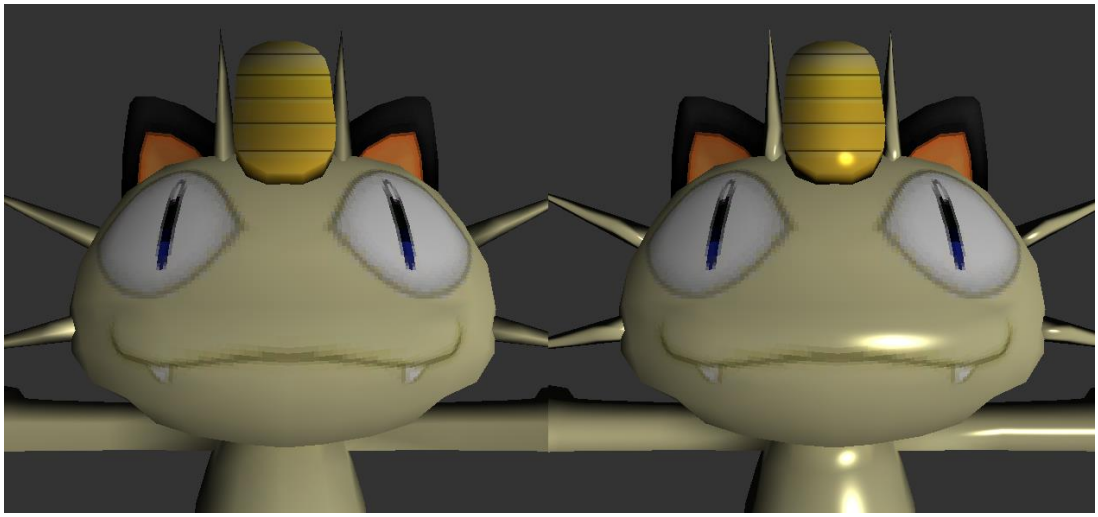
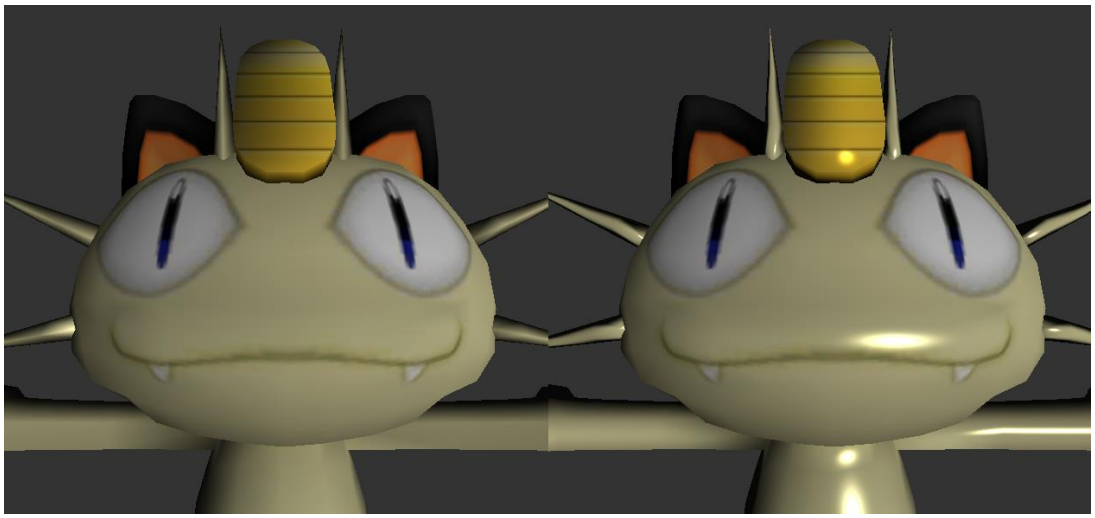
Keyboard_S：切換到 scale mode	
	

Keyboard_P	Keyboard_O
	

Keyboard_L：切換 light mode	
Point light	**Spot light**
	

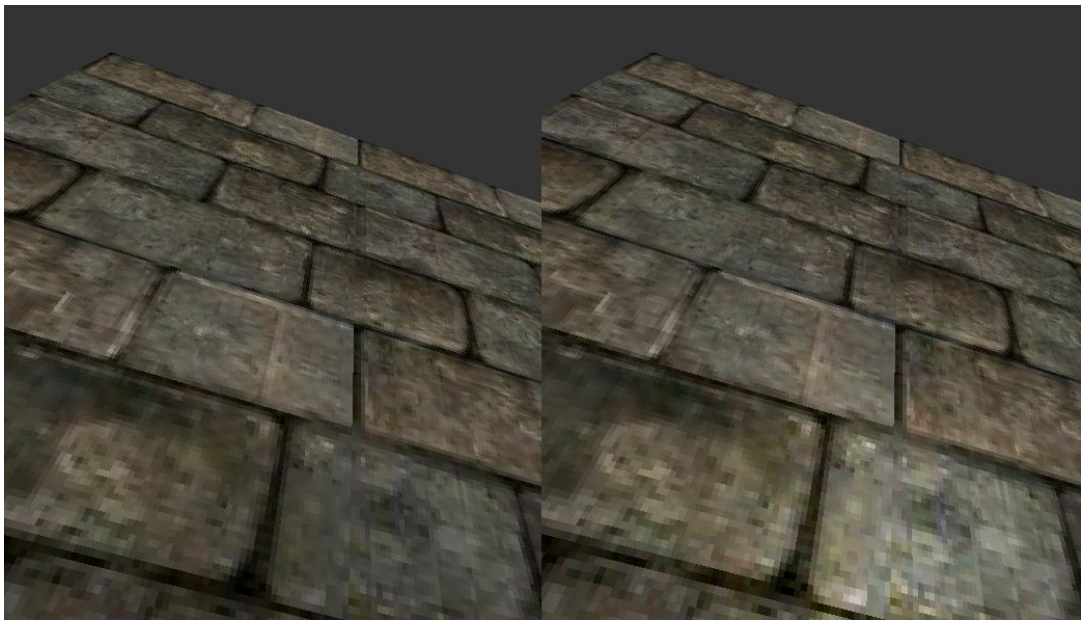
Keyboard_K：light editing mode	
	

Keyboard_J：shininess editing mode	
	

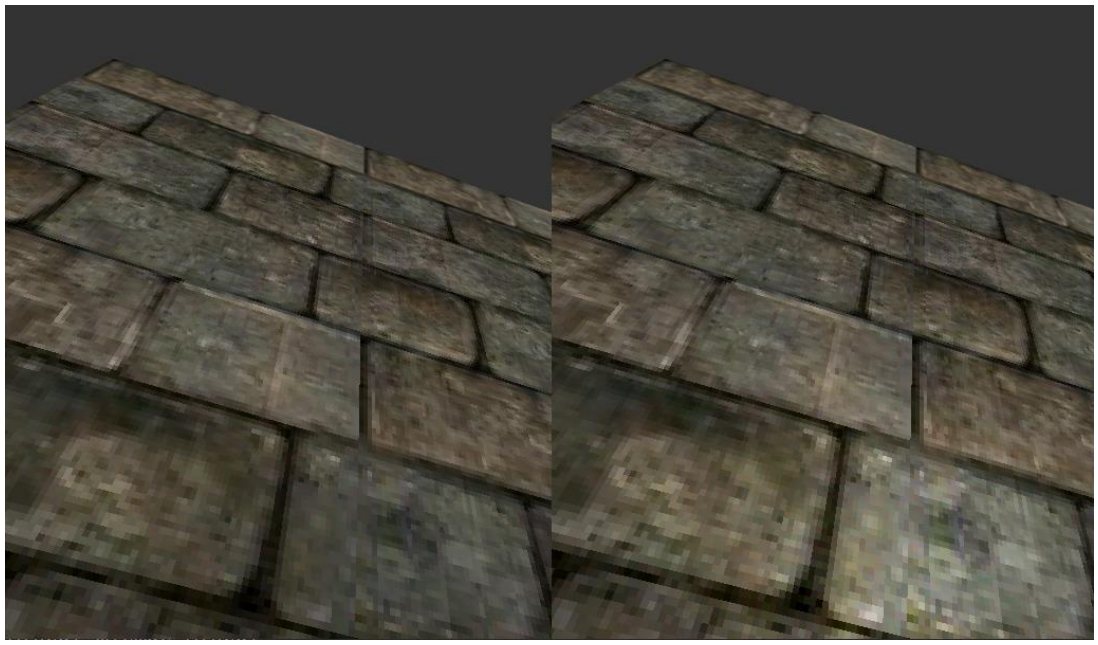
Keyboard_G：switch the magnification texture filtering mode
Nearest 
Linear 

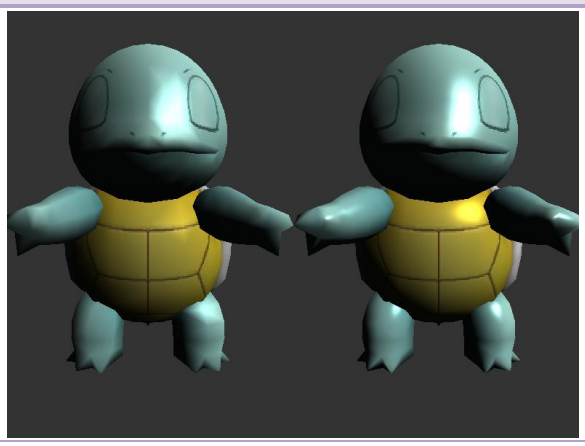
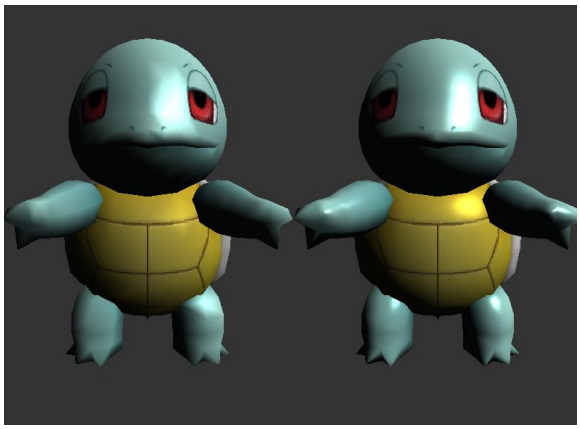
Keyboard_B：switch the minification texture filtering mode

Nearest_mipmap_linear



Linear_mipmap_linear



Keyboard_right/left_arrow	
	
	

III. Program control instructions

A. `LoadTextureImage()`

```
// [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_RGBA, width, height, 0, GL_RGBA, GL_UNSIGNED_BYTE, data);
glGenerateMipmap(GL_TEXTURE_2D);
// ============================================
```

B. `setUniformVariables()`

取得在 shader 計算需要的 data uniform location，以下為部分截圖。

```
uniform.iLocEyeIdx = glGetUniformLocation(program, "cur_eye_idx");
uniform.iLocIsEye = glGetUniformLocation(program, "isEye");
uniform.iLocRow = glGetUniformLocation(program, "row");
uniform.iLocCol = glGetUniformLocation(program, "col");
```

C. RenderScene(int per_vertex_or_per_pixel)

根據傳進來的參數 per_vertex_or_per_pixel 更改變數 draw，並將結果傳到 shader。

```
if(per_vertex_or_per_pixel==1){
    draw = 0;
    glUniform1i(uniform.iLocDraw, draw);
}
else if(per_vertex_or_per_pixel==0){
    draw = 1;
    glUniform1i(uniform.iLocDraw, draw);
}
```

根據該 model 中的變數 hasEye 與 isEye 傳遞參數到 fragment shader 判斷是否要使用眼睛的 texture。

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

if(models[cur_idx].hasEye == true){
    if (models[cur_idx].shapes[i].material.isEye == 0) glUniform1i(uniform.iLocIsEye, 0);
    else if(models[cur_idx].shapes[i].material.isEye == 1) glUniform1i(uniform.iLocIsEye, 1);
}
else {
    glUniform1i(uniform.iLocIsEye, 0);
}
```

根據變數 cur_eye_offset_idx 紀錄現在要畫第幾個眼睛，並計算他在 texture image 中的座標位置(用變數 row、col 紀錄)。

```
// repeat mode
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
// set R-coordinate of texture which represents third dimension (like z-coordinate)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_R, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

row = cur_eye_offset_idx / 4 * 0.5;
col = (cur_eye_offset_idx % 4) * 0.25;
//row = models[cur_idx].cur_eye_offset_idx / 4 * 0.5;
//col = (models[cur_idx].cur_eye_offset_idx % 4) * 0.25;
glUniform1f(uniform.iLocRow, row);
glUniform1f(uniform.iLocCol, col);
```

按鍵 G 與 B 會跟改變數 Nearest、Linear、Nearest_mipmap_Linear、Linear_mipmap_Linear 的值並使用對應的 texture filtering。

```
if(Nearest==true) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
if(Linear==true) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
if(Nearest_mipmap_Linear==true) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST_MIPMAP_LINEAR);
if(Linear_mipmap_Linear==true) glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
```

D. setHasEye()

因為只有 Pokémon 中的角色有眼睛，因此直接判斷哪幾個 model 為 Pokémon 並將該 model 的 hasEye 變數設為 true，反之設為 false。

```
void setHasEye() {
    models[0].hasEye = true;  // 2 materials
    models[1].hasEye = true;  // 2 materials
    models[2].hasEye = true;  // 3 materials
    models[3].hasEye = true;  // 2 materials
    models[4].hasEye = false; // people, 1 materials
    models[5].hasEye = false; // 2 materials
    models[6].hasEye = false; // wall, 1 materials
}
```

E. Vertex Shader

做法大致與作業二相同，但需要多傳遞 texture coordinate 參數，

```
void main()
{
    // [TODO]
    // ==========================
    texCoord = aTexCoord;
    gl_Position = um4p * um4v * um4m * vec4(aPos, 1.0);
    if(draw == 0) Lighting();
    else{
        vertex_color = aColor;
        vec4 view_aNormal = transpose(inverse(trs)) * vec4(aNormal, 0.0f);
        vertex_normal = view_aNormal.xyz;
    }
    // ==========================
}
```

F. Fragment Shader

與作業二作法相似，根據 uniform 變數 `draw` 的值判斷是否做 per_pixel lighting，同時也要判斷 uniform 變數 `isEye` 是否要做 texture transform，當 `isEye==true`，便要 update texture。

```
void main() {
    if(draw==1){
        Lighting();
    }else{
        vec2 updateTexture;
        if(isEye == 0){
            fragColor = vec4(vertex_color, 1.0f) * texture(uTexture, texCoord);
        }
        else{
            updateTexture = texCoord - vec2(row, col);
            fragColor = vec4(vertex_color, 1.0f) * texture(uTexture, updateTexture);
        }
    }
    // fragColor = vec4(texCoord.xy, 0, 1);
    // [TODO] sampleing from texture
    // Hint: texture
}
```