HW2 Reprt - GLSL

0616098 黃秉茂

Process

Step 1 create the shader and program

```
GLuint vert = createShader("Shaders/HW2_0616098.vert", "vertex");
GLuint frag = createShader("Shaders/HW2_0616098.frag", "fragment");
program = createProgram(vert, frag);
```

大致照著 exampleHW2 做，去 dll 新增資料夾 Shader，之後在

Shader 創建 vertex shader 跟 fragment shader

利用在 shader.h 寫好的 createShader 套上去，即可完成 shader

setting

接著一樣利用在 shader.h 寫好的 createProgram 套上去，即可完成

program setting

Step 2 finish LoadTexture()

```
glActiveTexture(GL_TEXTURE0 + i);
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

int width, height, nrChannels;
stbi_set_flip_vertically_on_load(true);
unsigned char* data = stbi_load(tFileName, &width, &height, &nrChannels, 0);

if (data)
{
// TODO : use image data to generate the texture here
// Hint :
//      # glTexImage2D()
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
}
```

照著講義給的資訊完成

建好 Texture buffer 後 bind 住，再把資料傳進去

```
glActiveTexture(GL_TEXTURE0);
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,
GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
                                LoadTexture() function
/* load texture image as data*/
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB,
GL_UNSIGNED_BYTE, data);
```

注意 GL_TEXTURE0 要隨著指定的 TEXTURE index 改變

Step 3 create VAO and VBO

先將 Pikachu 的 positions 和 textcoords 的資料用

vector<VertexAttribute>來記錄

```
// Pikachu
vector<VertexAttribute> Pikachu_VA;
VertexAttribute temp;
int id_pos = 0;
int id_tex = 0;
int delta_pos = 3;
int delta_tex = 2;
while (id_pos < model->positions.size())
{
    temp.setPosition(model->positions[id_pos], model->positions[id_pos + 1], model->positions[id_pos + 2]);
    id_pos += delta_pos;
    temp.setTexcoord(model->texcoords[id_tex], model->texcoords[id_tex + 1]);
    id_tex += delta_tex;
    Pikachu_VA.push_back(temp);
}
int vertices_length_pikachu = model->positions.size() / 3;
```

接著建立存放 Pikachu 的 VAO 和 VBO，然後把 Data 從

vector<VertexAttribute>放進去後，再指定應該如何存取

之後就把 bind 解開

```
// Generate a new buffer object
glGenBuffers(1, &Pikachu_VBO);
glBindBuffer(GL_ARRAY_BUFFER, Pikachu_VBO);

// Copy vertex data to the buffer object
glBufferData(GL_ARRAY_BUFFER, sizeof(VertexAttribute) * vertices_length_pikachu, &Pikachu_VA[0], GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
// It can let Vertex Attribute pointer know where to start load data in buffer
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, sizeof(VertexAttribute), // stride
    (void*)(offsetof(VertexAttribute, position)));

glEnableVertexAttribArray(1);
// It can let Vertex Attribute pointer know where to start load data in buffer
glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, sizeof(VertexAttribute), (void*)(offsetof(VertexAttribute, texcoord)
glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);
```

ball 也做相同的事，只是資料存在和 Pikachu 不同的 VAO 和 VBO

```
// ball
glGenVertexArrays(1, &ball_VAO);
glBindVertexArray(ball_VAO);
int vertices_length_ball = ball.size();
// Generate a new buffer object
glGenBuffers(1, &ball_VBO);
glBindBuffer(GL_ARRAY_BUFFER, ball_VBO);
// Copy vertex data to the buffer object
glBufferData(GL_ARRAY_BUFFER, sizeof(VertexAttribute) * vertices_length_ball, &ball[0], GL_STATIC_DRAW);
glEnableVertexAttribArray(0);
// It can let Vertex Attribute pointer know where to start load data in buffer
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, sizeof(VertexAttribute), // stride
    (void*)(offsetof(VertexAttribute, position)));
glEnableVertexAttribArray(1);
// It can let Vertex Attribute pointer know where to start load data in buffer
glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, sizeof(VertexAttribute), (void*)(offsetof(VertexAttribute, texcoord)
glBindBuffer(GL_ARRAY_BUFFER, 0);
glBindVertexArray(0);
```

Vertex shader 的 layout 記得依照 positions 和 textcoords 的位置對應

in 表示是從 openGL 取得

```
layout(location = 0) in vec3 position;
layout(location = 1) in vec2 texture;
```

Step 4 setting ModelView and Projection

進行要求的 transform 後，取得 openGL 目前的 ModelView and Projection，透過 Uniform 的方式傳給 vertex shader

```
// Pikachu
GLfloat pmtx[16];
GLfloat Pikachu_mmtx[16];

glPushMatrix();
glRotatef(rotate_angle, 0.0f, 1.0f, 0.0f);
glRotatef(25.0f, 0.0f, 1.0f, 0.0f);
glScalef(5.0f, 5.0f, 5.0f);
// get current modelview & projection matrix used in OpenGL rendering
glGetFloatv(GL_PROJECTION_MATRIX, pmtx);
glGetFloatv(GL_MODELVIEW_MATRIX, Pikachu_mmtx);

glPopMatrix();
// get the "location" (It's similar to pointer) of uniform variable in shader
GLint pmatLoc = glGetUniformLocation(program, "Projection");
GLint Pikachu_mmatLoc = glGetUniformLocation(program, "ModelView");

// using shader
glUseProgram(program);

// pass the projection matrix into vertex shader
glUniformMatrix4fv(pmatLoc, 1, GL_FALSE, pmtx);
// pass the modelview matrix into vertex shader
glUniformMatrix4fv(Pikachu_mmatLoc, 1, GL_FALSE, Pikachu_mmtx);
```

在 vertex shader 中將透過 uniform 取得的 matrix 對 position 進行座標轉換

並把 textcoord 傳入 fragment shader

```
#version 430

layout(location = 0) in vec3 position;
layout(location = 1) in vec2 texture;

out vec2 texcoord;

uniform mat4 Projection;
uniform mat4 ModelView;

void main() {
  // the final rendered position of vertex
  gl_Position = Projection * ModelView * vec4(position, 1.0);
  texcoord = texture;
}
```

Step 5 draw model - Pikachu

Bind VAO 後，啟動 Texture 和取得 Texture 後，讓 Texture 透過 uniform 傳給 fragment shader

Texture 要選對的 index

透過 glDrawArrays 呼叫 shader 畫圖，畫完圖之後 unbind

```
glBindVertexArray(Pikachu_VAO);
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, Pikachu_texture);
glUseProgram(program);

GLint texLoc = glGetUniformLocation(program, "Texture");
glUniform1i(texLoc, 0);
/* draw objects */
glDrawArrays(GL_TRIANGLES, 0, sizeof(VertexAttribute) * model->positions.size() / 3);

glBindTexture(GL_TEXTURE_2D, 0);
glBindVertexArray(0);
glActiveTexture(0);
glUseProgram(0);
```

fragment shader 透過依靠 uniform 傳來的 Texture 和 vertex shader 傳過來的 textcoord，利用函式進行上色

```
#version 430

uniform sampler2D Texture;
in vec2 texcoord;
out vec4 outColor;
void main() {
    outColor = texture2D(Texture, texcoord);
}
```

Step 6 draw ball

用 uv mapping 算好 ball 的 textcoord 後，把資料存回 vector<VertexAttribute>之後傳入 vertex shader

$$u = 0.5 + \frac{\arctan2(d_x, d_z)}{2\pi},$$
$$v = 0.5 - \frac{\arcsin(d_y)}{\pi}.$$

```
vert.normalize();
u = 0.5 + atan2(vert.x, vert.z) / (2 * M_PI);
v = 0.5 - asin(vert.y) / M_PI;
temp.setTexcoord(u, v);
```

進行要求的 transform 後，取得 openGL 目前的 ModelView and Projection，透過 Uniform 的方式傳給 vertex shader

```
// ball
GLfloat ball_mmtx[16];
glPushMatrix();
glRotatef(rotate_angle, 0.0f, 1.0f, 0.0f);
glRotatef(45.0f, 0.0f, 1.0f, 0.0f);
glTranslatef(3.0f, 0.0f, -3.0f);
glRotatef(180.0f, 0.0f, 0.0f, 1.0f);
//glBindVertexArray(vao);
// get current modelview & projection matrix used in OpenGL rendering
glGetFloatv(GL_MODELVIEW_MATRIX, ball_mmtx);

glPopMatrix();
// get the "location" (It's similar to pointer) of uniform variable in shader
GLint ball_mmatLoc = glGetUniformLocation(program, "ModelView");

// using shader
glUseProgram(program);

// pass the projection matrix into vertex shader
glUniformMatrix4fv(pmatLoc, 1, GL_FALSE, pmtx);
// pass the modelview matrix into vertex shader
glUniformMatrix4fv(ball_mmatLoc, 1, GL_FALSE, ball_mmtx);
```

bind 到 VAO 後，啟動 Texture 和取得 Texture 後，讓 Texture 透過 uniform 傳給 fragment shader

透過 glDrawArrays 呼叫 shader 畫圖，畫完圖之後 unbind

```
// using shader
glUseProgram(program);

// pass the projection matrix into vertex shader
glUniformMatrix4fv(pmatLoc, 1, GL_FALSE, pmtx);
// pass the modelview matrix into vertex shader
glUniformMatrix4fv(ball_mmatLoc, 1, GL_FALSE, ball_mmtx);

glBindVertexArray(ball_VAO);
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, ball_texture);
glUseProgram(program);

texLoc = glGetUniformLocation(program, "Texture");
glUniform1i(texLoc, 1);
/* draw objects */
glDrawArrays(GL_TRIANGLE_STRIP, 0, sizeof(VertexAttribute) * ball.size());

glBindTexture(GL_TEXTURE_2D, 0);
glBindVertexArray(0);
glActiveTexture(0);
glUseProgram(0);
```

fragment shader 透過依靠 uniform 傳來的 Texture 和 vertex shader 傳過來的 textcoord，利用函式進行上色

```
#version 430

uniform sampler2D Texture;
in vec2 texcoord;
out vec4 outColor;
void main() {
    outColor = texture2D(Texture, texcoord);
}
```

Step 7 Key function

用一個 bool 紀錄需不需要旋轉

```
if (key == 's' || key == 'S')
    is_rotate = !is_rotate;
```

需要旋轉時一直增加度數

```
if (is_rotate)
    rotate_angle++;
```

旋轉各個物件

```
glRotatef(rotate_angle, 0.0f, 1.0f, 0.0f);
```

the problems you met and how you solved them

Q: 一開始想用一個 VAO 解決 Pikachu 和 ball，但 layout location 會有問題

A: 用 2 個 VAO，分別記 Pikachu 和 ball

Q: 如何 bind Pikachu 到 VAO 跟 VBO?

A: 先把 model->positions 跟 model->texcoords 存成 vector<VertexAttribute>

Q: 如何得知 vertex 數量

A: 反算 model->positions

Q: 看起來 texture coordinate、shader 和 Texture 都沒錯，顏色卻怪怪的

A: .vert 少個分號不會報錯，修正分號