20201130 CG hw2 Report

0716207 呂思函

- vertexShader.vert

```
// 在vertex shader先畫好圖形的點，每個點會收到不同的資料
layout(location = 0) in vec3 position;// the position variable has attribute position 0
layout(location = 1) in vec3 normal;// the  normal variable has attribute position 1
layout(location = 2) in vec2 texcoord;// the  normal variable has attribute position 2

// uniform : 不會去動到的東西、不會被兩個shader共用的參數(ex. position or color)；類似global用法
uniform mat4 M;
uniform mat4 Projection;
uniform mat4 ModelView;

// set output，要傳給fragment shader的
out vec3 outNormal;
out vec2 outTexcoord;

void main(){
gl_Position = Projection * ModelView * M * vec4(position, 1.0);// 轉成四維輸出(因為要做矩陣乘法)
outNormal = normal;
outTexcoord = texcoord;
}
// 最後輸出的vertices一定要用gl_Position接(內建固定用法)
// 乘上model matrix(M)，伊布就會長好長在自己的位置上
// normal和texcoord要接到(in->out)
// 我哩個傻逼這個問題找了一整天、ㄑ
```

- fragmentShader.frag

```
// 在fragment shader著色
// 因為用layout(binding=0)會怪怪的，所以直接用uniform宣告
uniform sampler2D Texture;

// input要接vertexShader的output，名稱要相同!(out->in)
in vec3 outNormal;
in vec2 outTexcoord;

// 最終輸出vertices呈現的顏色， frag_color是固定內建格式
out vec4  frag_color;

void main(){
 frag_color = texture2D(Texture, outTexcoord);
}
// texture2D(Texture, outTexcoord)直接出來就是四維(2d + 2d)
```

- main.cpp

```
    GLuint program;
    GLuint VAO, VBO[3];// VBO[3]分開存positions, normals, texcoords(三條buffer)
    unsigned int basistexture, modeltexture;
    int windowSize[2] = { 600, 600 };
    float angle = 0.0f;

    Object* model = new Object("UmbreonHighPoly.obj");
```

```
void shaderInit() {
    //// TODO: ////
    //
    // Hint:
    // 1. createShader
    // 2. createProgram
    GLuint vert = createShader("Shaders/vertexShader.vert", "vertex");
    GLuint frag = createShader("Shaders/fragmentShader.frag", "fragment");
    program = createProgram(vert, frag);
}
```

初始化 shader，分別建立 vertexShader 和 fragmentShader，並利用這兩個 shader 去建立一個 program(主要畫畫的區域)

```
void bindbufferInit() {
    //// TODO: ////
    // Hint:
    // 1. Setup VAO
    // 2. Setup VBO of vertex positions, normals, and texcoords

    // generate a new array object
    glGenVertexArrays(1, &VAO);
    // bind a vertex array object
    glBindVertexArray(VAO);
```

建立 vertex array object(VAO)，用來存放三個 VBO(VBO[3])

```
80  // vertex
81  glGenBuffers(1, &VBO[0]);//generate a new buffer object
82  glBindBuffer(GL_ARRAY_BUFFER, VBO[0]);// how to use buffer
83  glBufferData(GL_ARRAY_BUFFER, sizeof(GLfloat) * size(model->positions), model->positions.data(), GL_STATIC_DRAW);// copy vertex data to the buffer object
84  glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);// stride : xyz；起始點從0開始
85  glEnableVertexAttribArray(0);
86  glBindBuffer(GL_ARRAY_BUFFER, 0);// unbind VBO
87  glBindVertexArray(0);// unbind VAO
```

VBO[0]存放 vertices

*glVertexAttribPointer 的第一項、glEnableVertexAttribArray、glBindVertexArray 的數字要對到(pointer 指到的位置)

```
89  // normal
90  glGenVertexArrays(1, &VBO[1]);//generate a new buffer object
91  glBindBuffer(GL_ARRAY_BUFFER, VBO[1]);// how to use buffer
92  glBufferData(GL_ARRAY_BUFFER, sizeof(GLfloat) * size(model->normals), model->normals.data(), GL_STATIC_DRAW);// copy vertex data to the buffer object
93  glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);// stride : nx, ny, nz；起始點從0開始
94  glEnableVertexAttribArray(1);
95  glBindBuffer(GL_ARRAY_BUFFER, 0);// unbind VBO
96  glBindVertexArray(1);// unbind VAO
```

VBO[1]存放 normals

```
98  // texcoord
99  glGenVertexArrays(1, &VBO[2]);//generate a new buffer object
100 glBindBuffer(GL_ARRAY_BUFFER, VBO[2]);// how to use buffer
101 glBufferData(GL_ARRAY_BUFFER, sizeof(GLfloat) * size(model->texcoords), model->texcoords.data(), GL_STATIC_DRAW);// copy vertex data to the buffer object
102 glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE, 2 * sizeof(float), (void*)0);// stride : s0, t0；起始點從0開始
103 glEnableVertexAttribArray(2);
104 glBindBuffer(GL_ARRAY_BUFFER, 0);// unbind VBO
105 glBindVertexArray(2);// unbind VAO
```

VBO[2]存放 texcoords(texture coordinates： 材質座標)

```
206 glActiveTexture(GL_TEXTURE0);
207 glBindTexture(GL_TEXTURE_2D, basistexture);
208
209 /// top
210 glBegin(GL_POLYGON);
211 for (int i = 0; i < sides; i++){ ... }
218 glEnd();
219
220 /// bottom
221 glBegin(GL_POLYGON);
222 for (int i = 0; i < sides; i++){ ... }
229 glEnd();
```

DrawBasis()跟 hw1 差不多，因為想讓上下兩面咖咖的，所以讓 glActiveTexture 和 glBindTexture 從 206 行這邊包

```
231 /// draw square
232 glBegin(GL_QUADS);
233 for (int i = 0; i < sides; i++)
234 {
235   double t = i;
236   double ang = PI * i * 2 / sides;
237   double tmp = PI * (t + 1) * 2 / sides;
238
239   /// 左下
240   glNormal3f(sin(ang), 0.0, cos(ang));
241   glTexCoord2f(1.0f, 0.0f);
242   glVertex3d(r * sin(ang), -3.0, r * cos(ang));
243   /// 右下
244   glNormal3f(sin(ang), 0.0, cos(ang));
245   glTexCoord2f(1.0f, 1.0f);
246   glVertex3d(r * sin(tmp), -3.0, r * cos(tmp));
247   /// 右上
248   glNormal3f(sin(ang), 0.0, cos(ang));
249   glTexCoord2f(0.0f, 1.0f);
250   glVertex3d(r * sin(tmp), 0.0, r * cos(tmp));
251   /// 左上
252   glNormal3f(sin(ang), 0.0, cos(ang));
253   glTexCoord2f(0.0f, 0.0f);
254   glVertex3d(r * sin(ang), 0.0, r * cos(ang));
255 }
256 glEnd();
257
258 glBindTexture(GL_TEXTURE_2D, 0);// unbind texture
259 glActiveTexture(0);
```

記得畫每個側邊的點之前都要加 glTexCoord2f，讓 shader 知道貼圖要怎麼貼

```
264  void DrawUmbreon()
265  {
266      glUseProgram(program);// 選擇要用哪個shader
267
268      glm::mat4 M(1.0f);
269      M = glm::rotate(M, glm::radians(angle), glm::vec3(0, 1, 0));
270      M = glm::translate(M, glm::vec3(0, 1.3, 0));
271
272      GLuint ModelMatrixID = glGetUniformLocation(program, "M");
273      glUniformMatrix4fv(ModelMatrixID, 1, GL_FALSE, &M[0][0]);// 從M[0][0]開始放，送到shader
```

- ✧ glUseProgram : 跟程式說要開始畫了(這行要放在最開始ヽ ノ)
- ✧ 初始化一個四維矩陣放模型
- ✧ 並設置 model 的旋轉和位置
- ✧ 把 model 的四維矩陣丟進 program
- ✧ 再把上一步處理好的 program 送到 shader 去畫畫

```
279      // GLfloat pmtx[16];
280      // GLfloat mmtx[16];
281      glm::mat4 pmtx = getP();
282      glm::mat4 mmtx = getV();
283
284      // glGetFloatv(GL_PROJECTION_MATRIX, pmtx);
285      // glGetFloatv(GL_MODELVIEW_MATRIX, mmtx);
286      GLint pmatLoc = glGetUniformLocation(program, "Projection");
287      GLint mmatLoc = glGetUniformLocation(program, "ModelView");
288
289      //input the modelview matrix into vertex shader
290      glUniformMatrix4fv(pmatLoc, 1, GL_FALSE, &pmtx[0][0]);// 從pmtx[0][0]開始放，送到shader，第二個數字是location ID
291      //input the rotation matrix into vertex shader
292      glUniformMatrix4fv(mmatLoc, 1, GL_FALSE, &mmtx[0][0]);//從mmtx[0][0]開始放，送到shader，第二個數字是location ID
```

- ✧ getP()和 getV()會回傳定義在其他 functions 的透視投影法和視角(相機位置)
- ✧ (注意型態要相同!!!(glm::mat4)註解的部分是 spec 上的，正確的寫法由註解的東西來的)
- ✧ 將 Projection 和 ModelView 丟進 program，讓 shader 帶著走

```
294      glActiveTexture(GL_TEXTURE0);
295      glBindTexture(GL_TEXTURE_2D, modeltexture);
296      GLint texLoc = glGetUniformLocation(program, "Texture");
297      glUniform1i(texLoc, 0);
298
299      glBindVertexArray(VAO);
300      glDrawArrays(GL_QUADS, 0, 4 * model->fNum);// draw object : 0 : starting index in the enabled arrays ; 4 * model->fNum個頂點，頂點重複
301      glBindVertexArray(0);// unbind VAO
302      glBindTexture(GL_TEXTURE_2D, 0);// unbind texture
303      glActiveTexture(0);
304      glUseProgram(0);// switch program
```

- ✧ 開始貼皮
- ✧ 把要用的 texture 丟進來(在 LoadTexture 處理過的皮)
- ✧ 把 texture 也丟進 program，要傳給 shader 給他素材可以畫畫
- ✧ glUniform1i(texLoc, 0)，本來後面的 0 是要開通道讓 texture 可以進 fragmentShader，但 binding 的用法卡住了，所以就不開這個通道
- ✧ 拿一個 VAO
- ✧ 開始畫很多小塊方形在月亮伊布上
- ✧ 畫完要記得把 VAO 和 VBO 都 unbind，釋出空間，讓之後要用 buffer 和 array 的使用者能用
- ✧ 最後要記得把 program 關掉 ： 跟程式說畫完了，或有用到其他 program 的話，要先關掉原本的 program 才能用下一個 program

- 遇到的問題….已經有化成自我小提醒和怨念寫在上面的截圖裡或在每張截圖下的註解裡惹
  - Gl_FragColor 不能重複輸出，但是今天我想要輸出兩個 texture：使用 gl_FragData[]來把輸出包成矩陣=>但是我用了還是失敗