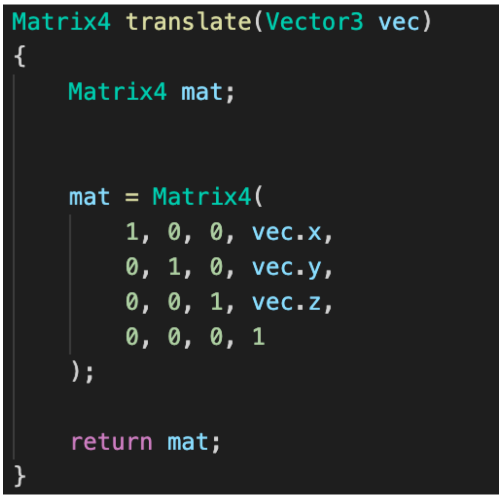
圖學HW1 Report

main.cpp:

- translate()、scaling()、rotateX()、rotateY()、rotateZ()：依照老師第五章講義將矩陣完成。下圖以translate為例。

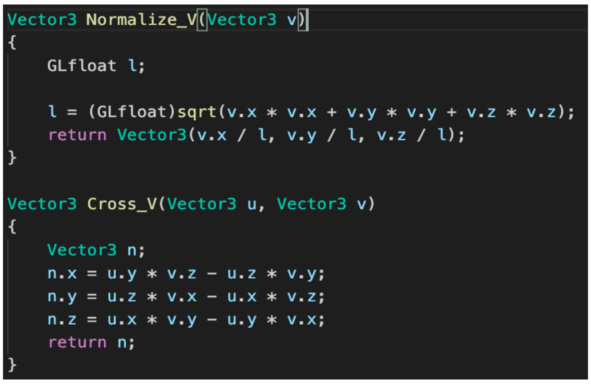


```
Matrix4 translate(Vector3 vec)
{
    Matrix4 mat;


    mat = Matrix4(
        1, 0, 0, vec.x,
        0, 1, 0, vec.y,
        0, 0, 1, vec.z,
        0, 0, 0, 1
    );

    return mat;
}
```

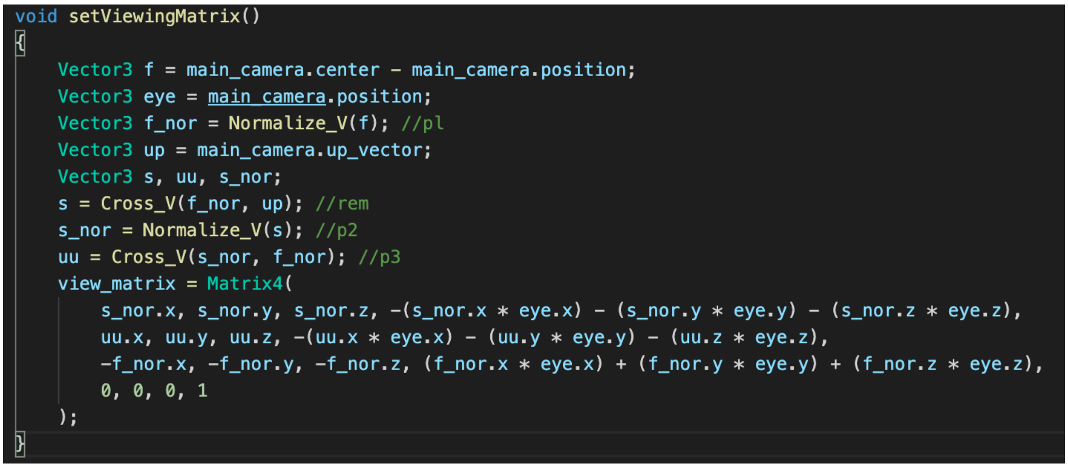
$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- setViewingMatrix()：參考老師第五章講義第72、74頁並實作，首先Rz為center-position後做normalize，Rx為Rz cross up_vector後做normalize，Ry則為Rx cross Rz，實作老師的公式後便可做出view_matrix。其中的Normalize_V及Cross_V與原本給的function算法相同，只是回傳的型態改為Vector3。

$$M_{view} = R \bullet T = \begin{bmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

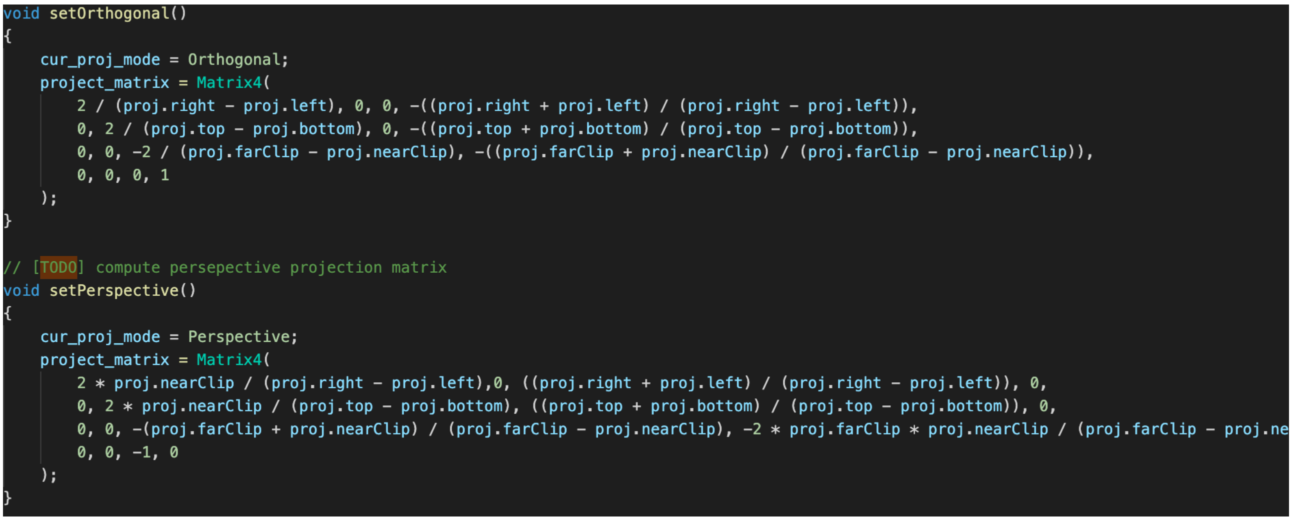


```
Vector3 Normalize_V(Vector3 v)
{
    GLfloat l;

    l = (GLfloat)sqrt(v.x * v.x + v.y * v.y + v.z * v.z);
    return Vector3(v.x / l, v.y / l, v.z / l);
}

Vector3 Cross_V(Vector3 u, Vector3 v)
{
    Vector3 n;
    n.x = u.y * v.z - u.z * v.y;
    n.y = u.z * v.x - u.x * v.z;
    n.z = u.x * v.y - u.y * v.x;
    return n;
}
```



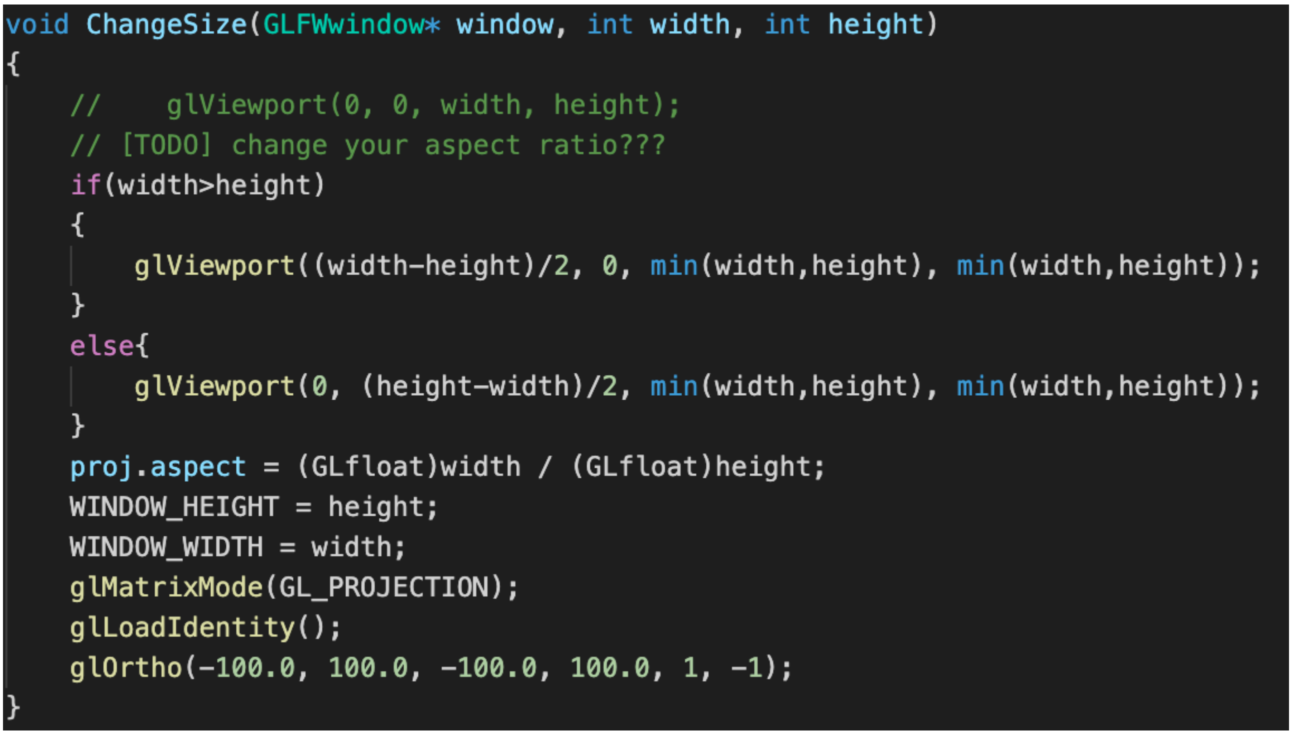
```
void setViewingMatrix()
{
    Vector3 f = main_camera.center - main_camera.position;
    Vector3 eye = main_camera.position;
    Vector3 f_nor = Normalize_V(f); //p1
    Vector3 up = main_camera.up_vector;
    Vector3 s, uu, s_nor;
    s = Cross_V(f_nor, up); //rem
    s_nor = Normalize_V(s); //p2
    uu = Cross_V(s_nor, f_nor); //p3
    view_matrix = Matrix4(
        s_nor.x, s_nor.y, s_nor.z, -(s_nor.x * eye.x) - (s_nor.y * eye.y) - (s_nor.z * eye.z),
        uu.x, uu.y, uu.z, -(uu.x * eye.x) - (uu.y * eye.y) - (uu.z * eye.z),
        -f_nor.x, -f_nor.y, -f_nor.z, (f_nor.x * eye.x) + (f_nor.y * eye.y) + (f_nor.z * eye.z),
        0, 0, 0, 1
    );
}
```

- setOrthogonal()、setPerspective()：首先先改變cur_proj_mode為相對應的模式，再利用老師講義的公式做出project_matrix，值得注意的是這兩個矩陣的第三列第三行需加上負號。



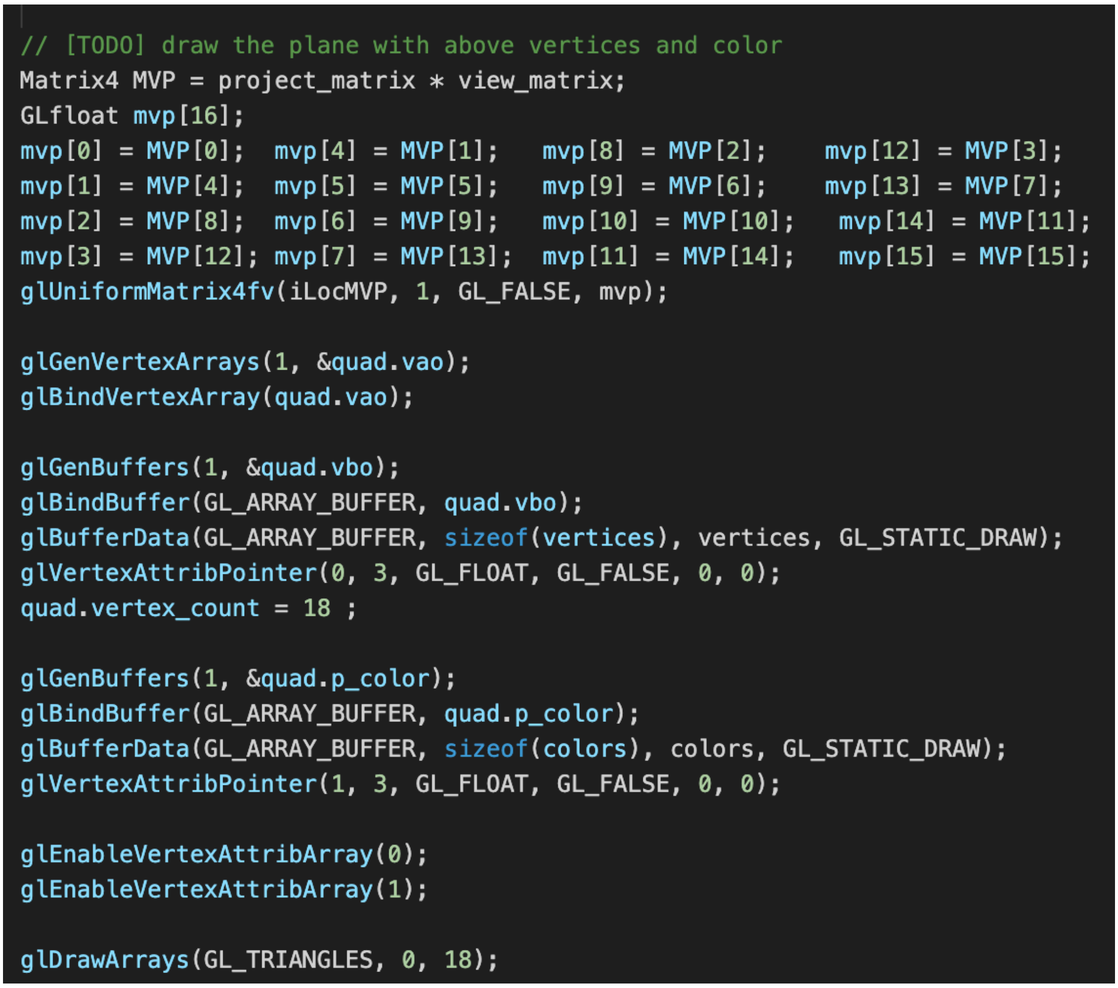
```
void setOrthogonal()
{
    cur_proj_mode = Orthogonal;
    project_matrix = Matrix4(
        2 / (proj.right - proj.left), 0, 0, -((proj.right + proj.left) / (proj.right - proj.left)),
        0, 2 / (proj.top - proj.bottom), 0, -((proj.top + proj.bottom) / (proj.top - proj.bottom)),
        0, 0, -2 / (proj.farClip - proj.nearClip), -((proj.farClip + proj.nearClip) / (proj.farClip - proj.nearClip)),
        0, 0, 0, 1
    );
}

// [TODO] compute persepective projection matrix
void setPerspective()
{
    cur_proj_mode = Perspective;
    project_matrix = Matrix4(
        2 * proj.nearClip / (proj.right - proj.left),0, ((proj.right + proj.left) / (proj.right - proj.left)), 0,
        0, 2 * proj.nearClip / (proj.top - proj.bottom), ((proj.top + proj.bottom) / (proj.top - proj.bottom)), 0,
        0, 0, -(proj.farClip + proj.nearClip) / (proj.farClip - proj.nearClip), -2 * proj.farClip * proj.nearClip / (proj.farClip - proj.ne
        0, 0, -1, 0
    );
}
```

- ChangeSize()：這個function是參考網路上的資源，將aspect改為width/height，改變Window的寬和高，最後呼叫glMatrixMode、glLoadIdentity、glOrtho。



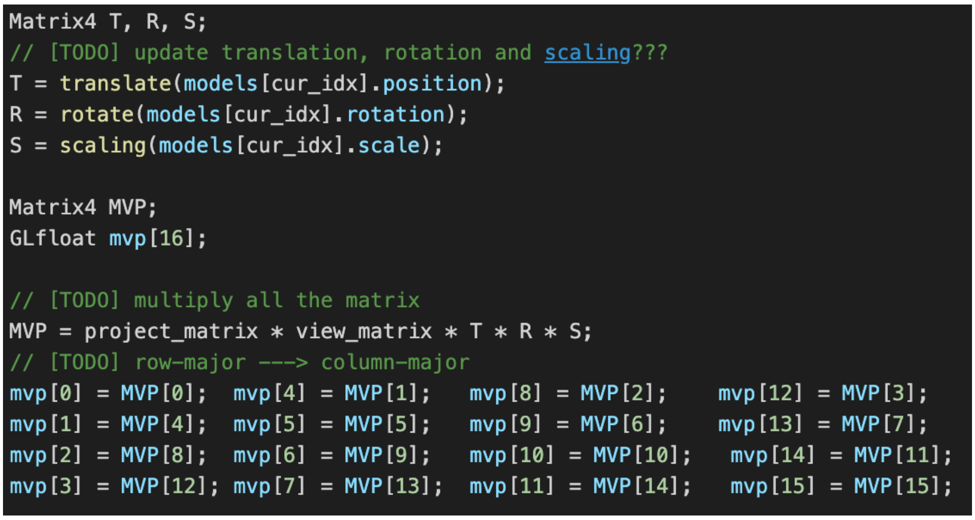
```
void ChangeSize(GLFWwindow* window, int width, int height)
{
    //    glViewport(0, 0, width, height);
    // [TODO] change your aspect ratio???
    if(width>height)
    {
        glViewport((width-height)/2, 0, min(width,height), min(width,height));
    }
    else{
        glViewport(0, (height-width)/2, min(width,height), min(width,height));
    }
    proj.aspect = (GLfloat)width / (GLfloat)height;
    WINDOW_HEIGHT = height;
    WINDOW_WIDTH = width;
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-100.0, 100.0, -100.0, 100.0, 1, -1);
}
```

- drawPlane()：先創造出MVP矩陣後，用上面定義好的Shape quad來做，步驟則是參考LoadModels function就可以畫出plane。



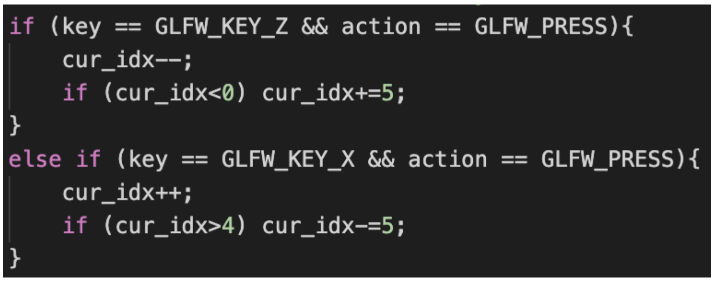
```
// [TODO] draw the plane with above vertices and color
Matrix4 MVP = project_matrix * view_matrix;
GLfloat mvp[16];
mvp[0] = MVP[0];  mvp[4] = MVP[1];   mvp[8] = MVP[2];    mvp[12] = MVP[3];
mvp[1] = MVP[4];  mvp[5] = MVP[5];   mvp[9] = MVP[6];    mvp[13] = MVP[7];
mvp[2] = MVP[8];  mvp[6] = MVP[9];   mvp[10] = MVP[10];   mvp[14] = MVP[11];
mvp[3] = MVP[12]; mvp[7] = MVP[13];  mvp[11] = MVP[14];   mvp[15] = MVP[15];
glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);

glGenVertexArrays(1, &quad.vao);
glBindVertexArray(quad.vao);

glGenBuffers(1, &quad.vbo);
glBindBuffer(GL_ARRAY_BUFFER, quad.vbo);
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, 0);
quad.vertex_count = 18 ;

glGenBuffers(1, &quad.p_color);
glBindBuffer(GL_ARRAY_BUFFER, quad.p_color);
glBufferData(GL_ARRAY_BUFFER, sizeof(colors), colors, GL_STATIC_DRAW);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 0, 0);

glEnableVertexAttribArray(0);
glEnableVertexAttribArray(1);

glDrawArrays(GL_TRIANGLES, 0, 18);
```

- RenderScene()：呼叫translate、rotate、scaling function將T、R、S矩陣算出，MVP矩陣則是由model matrix、view matrix、project matrix所組成，(project*view*model 其中model matrix為T*R*S)，row major 轉column major則是簡單的做行列互換。



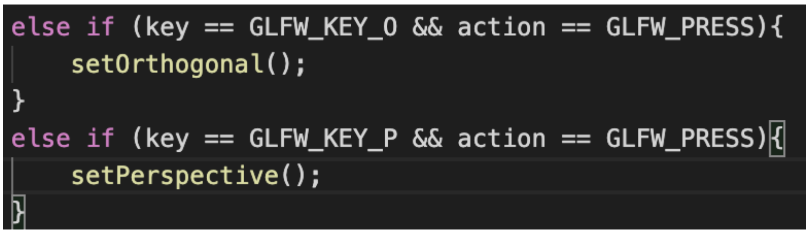
```
Matrix4 T, R, S;
// [TODO] update translation, rotation and scaling???
T = translate(models[cur_idx].position);
R = rotate(models[cur_idx].rotation);
S = scaling(models[cur_idx].scale);

Matrix4 MVP;
GLfloat mvp[16];

// [TODO] multiply all the matrix
MVP = project_matrix * view_matrix * T * R * S;
// [TODO] row-major ---> column-major
mvp[0] = MVP[0];  mvp[4] = MVP[1];   mvp[8] = MVP[2];    mvp[12] = MVP[3];
mvp[1] = MVP[4];  mvp[5] = MVP[5];   mvp[9] = MVP[6];    mvp[13] = MVP[7];
mvp[2] = MVP[8];  mvp[6] = MVP[9];   mvp[10] = MVP[10];   mvp[14] = MVP[11];
mvp[3] = MVP[12]; mvp[7] = MVP[13];  mvp[11] = MVP[14];   mvp[15] = MVP[15];
```

- KeyCallback()：
  - X、Z：這兩個指令是做圖片轉換，將cur_idx做加減即可。



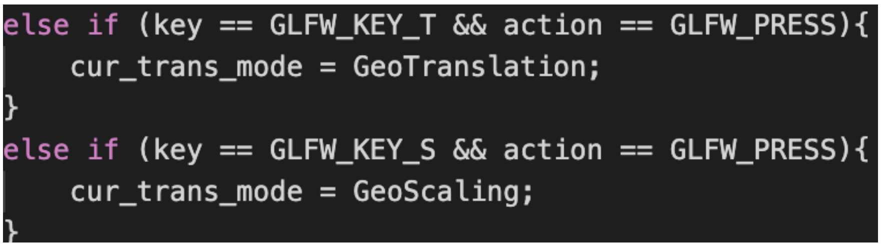
```
if (key == GLFW_KEY_Z && action == GLFW_PRESS){
    cur_idx--;
    if (cur_idx<0) cur_idx+=5;
}
else if (key == GLFW_KEY_X && action == GLFW_PRESS){
    cur_idx++;
    if (cur_idx>4) cur_idx-=5;
}
```

  - O、P：這兩個指令是做projection的轉換，分別呼叫setOrthogonal()、setPerspective()。



```
else if (key == GLFW_KEY_O && action == GLFW_PRESS){
    setOrthogonal();
}
else if (key == GLFW_KEY_P && action == GLFW_PRESS){
    setPerspective();
}
```

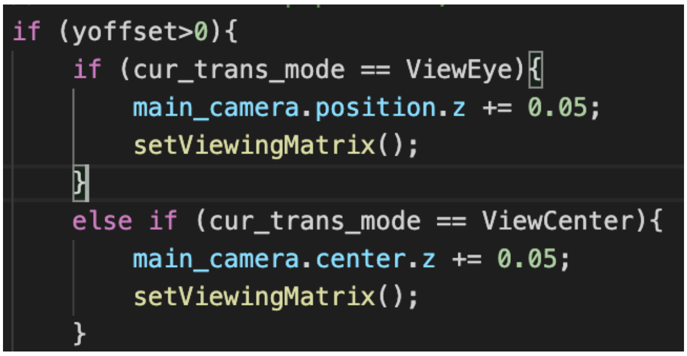
  - T、S、R、E、C、U：這六個指令均是改變cur_trans_mode，這裡用T、S的code表示。



```
else if (key == GLFW_KEY_T && action == GLFW_PRESS){
    cur_trans_mode = GeoTranslation;
}
else if (key == GLFW_KEY_S && action == GLFW_PRESS){
    cur_trans_mode = GeoScaling;
}
```

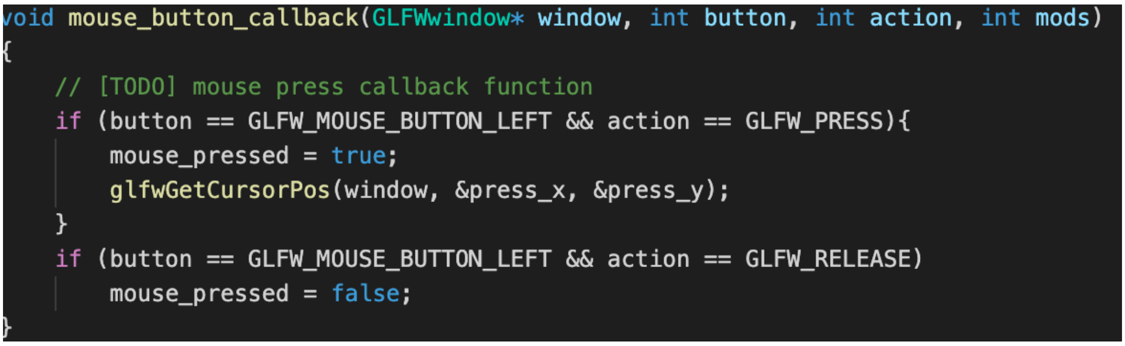
  - I：這個指令是要print出資訊，依照助教給的detail將translation、rotation、scaling、view、projection matrix print出來。

- scroll_callback()：這個callback處理滑鼠滾輪，也是z軸的變換，首先先判斷滑鼠滾輪的上下(yoffset)後，根據相對應的cur_trans_mode做z軸的加減。下圖以滾輪向上的一小部分為例。



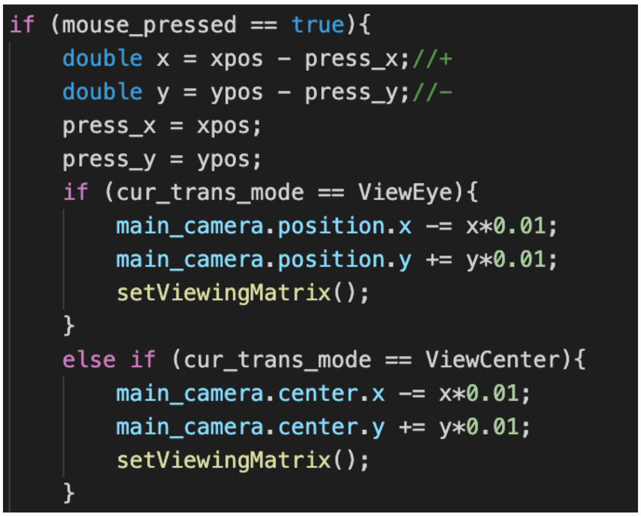
```
if (yoffset>0){
    if (cur_trans_mode == ViewEye){
        main_camera.position.z += 0.05;
        setViewingMatrix();
    }
    else if (cur_trans_mode == ViewCenter){
        main_camera.center.z += 0.05;
        setViewingMatrix();
    }
```

- mouse_button_callback()：當偵測到滑鼠按下時將mouse_pressed設為true，在全域開press_x、press_y的變數，記住按下按鍵時的滑鼠位置，放開則將mouse_pressed設為false。



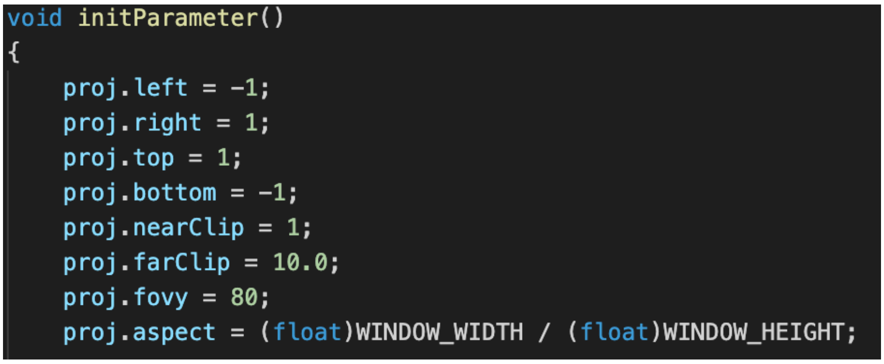
```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    // [TODO] mouse press callback function
    if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_PRESS){
        mouse_pressed = true;
        glfwGetCursorPos(window, &press_x, &press_y);
    }
    if (button == GLFW_MOUSE_BUTTON_LEFT && action == GLFW_RELEASE)
        mouse_pressed = false;
}
```

- cursor_pos_callback()：滑鼠事件用來處理x、y軸的變換，當滑鼠是按著時算出移動後及移動前的差，並根據cur_trans_mode做x、y軸的加減，而每個加減都乘上一個常數，使移動速度合理。其中當cur_trans_mode跟View有關時要重設view_matrix。以下以ViewEye、ViewCenter的code為示範。



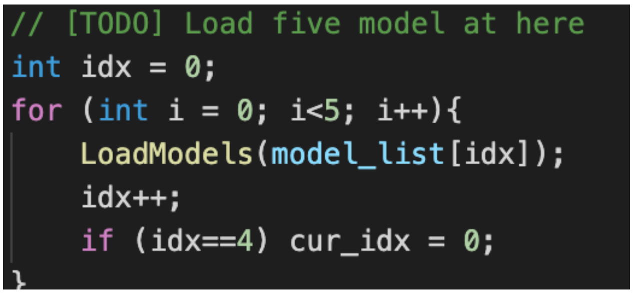
```
if (mouse_pressed == true){
    double x = xpos - press_x;//+
    double y = ypos - press_y;//-
    press_x = xpos;
    press_y = ypos;
    if (cur_trans_mode == ViewEye){
        main_camera.position.x -= x*0.01;
        main_camera.position.y += y*0.01;
        setViewingMatrix();
    }
    else if (cur_trans_mode == ViewCenter){
        main_camera.center.x -= x*0.01;
        main_camera.center.y += y*0.01;
        setViewingMatrix();
    }
```

- initParameter()：原先implement完後圖片會在很遠的地方，必須改變initParameter圖片才會在正確的地方。



```
void initParameter()
{
    proj.left = -1;
    proj.right = 1;
    proj.top = 1;
    proj.bottom = -1;
    proj.nearClip = 1;
    proj.farClip = 10.0;
    proj.fovy = 80;
    proj.aspect = (float)WINDOW_WIDTH / (float)WINDOW_HEIGHT;
```

- setupRC()：利用for-loop將5個model load進來。



```
// [TODO] Load five model at here
int idx = 0;
for (int i = 0; i<5; i++){
    LoadModels(model_list[idx]);
    idx++;
    if (idx==4) cur_idx = 0;
}
```

shader.vs:

- main：將mvp矩陣乘入

```
void main()
{
    // [TODO]
    gl_Position = mvp * vec4(aPos.x, aPos.y, aPos.z, 1.0);
    vertex_color = aColor;
}
```