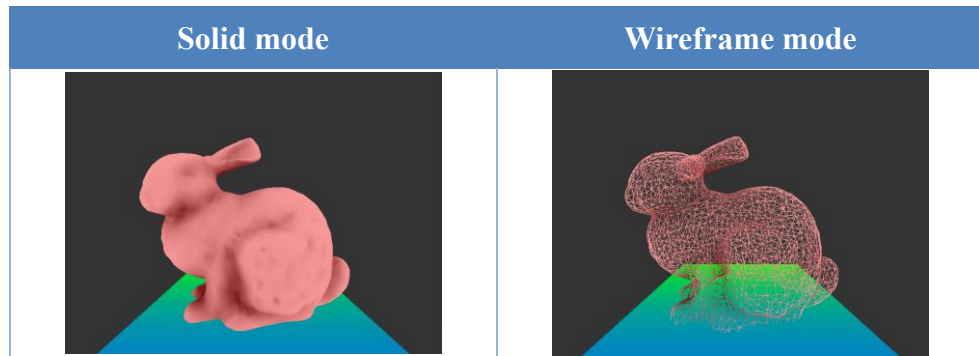


Computer Graphics Homework1 Report

107062313 黃寶瑩

◆ Key mapping

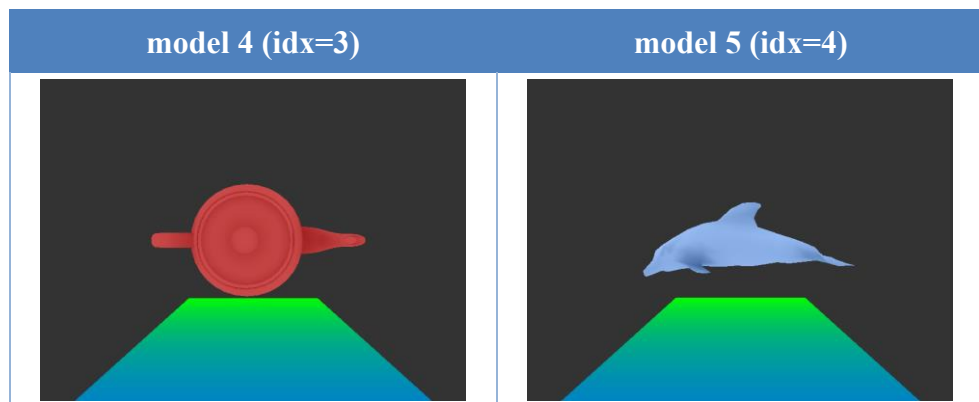
1. W : switch between solid and wireframe mode



新增一個 boolean 變數 `isSolid` 來記為 Solid mode 或 Wireframe mode，按下 W 鍵時切換 `isSolid` 的值，在 `RenderScene()` 要做 rendering 時去判斷 `isSolid` 的 value，若 `isSolid` 為 false 便利用 `glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)` 設定只要畫 primitives 的邊框，反之則設定為 `GL_FILL`。

```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);  
if (!isSolid) glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
```

2. Z/X : switch the model



在 `setupRC()` 中利用 for loop 將 5 個 models load 進來，`cur_idx` 介於 0 ~ 4 之間，按下 Z/W 鍵 update `cur_idx` 的 value，每次 render 都是根據 `cur_idx` 的值畫出對應的 model。

```
else if (key == GLFW_KEY_Z && action == GLFW_PRESS) {  
    if (cur_idx == 0) cur_idx = 4;  
    else cur_idx = cur_idx - 1;  
}  
else if (key == GLFW_KEY_X && action == GLFW_PRESS) {  
    cur_idx = (cur_idx + 1) % 5;  
}
```

3. O : switch to Orthogonal projection

P : switch to NDC Perspective projection

Orthogonal projection matrix

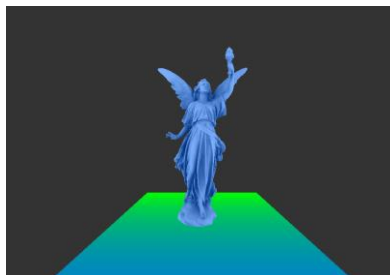
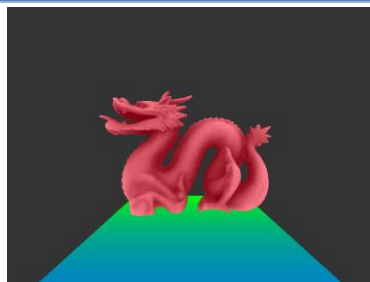
```
x_diff = proj.right - proj.left;
x_sum = proj.right + proj.left;
y_diff = proj.top - proj.bottom;
y_sum = proj.top + proj.bottom;
project_matrix = Matrix4(
    2 / x_diff, 0, 0, (-1)*x_sum / x_diff,
    0, 2 / y_diff, 0, (-1)*y_sum / y_diff,
    0, 0, (-1) * 2 / (f - n), (-1)*(f + n) / (f - n),
    0, 0, 0, 1
);
```

Perspective projection matrix

```
GLfloat angle;
GLfloat ar;
GLfloat n, f;
GLfloat radian = proj.fovy / 180.0 * 3.1415926;
angle = tan(radian / 2);
ar = proj.aspect;
n = proj.nearClip;
f = proj.farClip;

if (proj.aspect > 1) {
    project_matrix = Matrix4(
        1 / (ar*angle), 0, 0, 0,
        0, 1 / angle, 0, 0,
        0, 0, -(f + n) / (f - n), -2 * f*n / (f - n),
        0, 0, -1, 0
    );
}
else {
    project_matrix = Matrix4(
        1 / angle, 0, 0, 0,
        0, ar / angle, 0, 0,
        0, 0, -(f + n) / (f - n), -2 * f*n / (f - n),
        0, 0, -1, 0
    );
}
```

Perspective Projection



Orthogonal Projection



- setPerspective()

有三種情況會呼叫 setPerspective()，分別是 initialize、ChangeSize() 以及按下 P 鍵。

Perspective projection matrix 需要考慮到 FOV，因此先將 FOV 從角度轉換成弧度，再將數值帶入建立 projection matrix。但如果直接使用老師上課講義中 projection matrix 寫法並無法正確投影，試過網路上不同的寫法才拼湊出可以正確投影的矩陣。

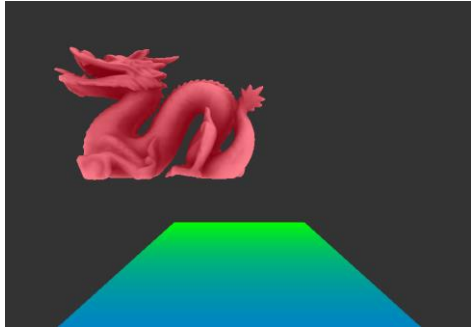
- setOrthogonal()

按下 O 鍵以及 ChangeSize() 時會呼叫 setOrthogonal()。

4. T : translation mode

Translation matrix

```
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;
}
```



分別根據 cursor 水平與垂直移動距離 update `models[cur_idx].position.x` 與 `models[cur_idx].position.y`，z 軸方向則是根據滑鼠滾輪做調整。

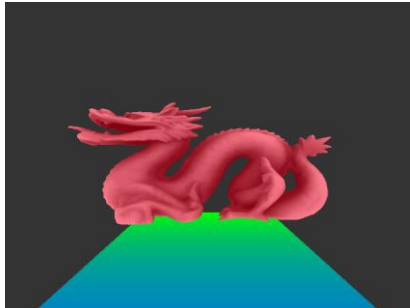
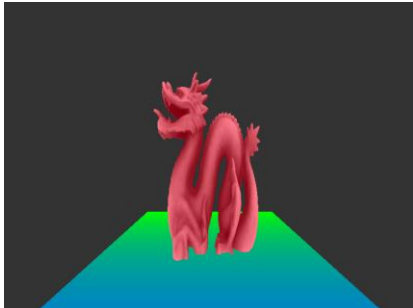
< 滑鼠往右(左/上/下)移動：物體往右(左/上/下)移動 >

```
static void cursor_pos_callback(GLFWwindow* window, double xpos, double ypos)
{
    // [TODO] cursor position callback function=====
    if (starting_press_x == -1 && starting_press_y == -1) {
        starting_press_x = xpos;
        starting_press_y = ypos;
    }
    if (isPress == true) {
        float diffx = xpos - starting_press_x;
        float diffy = ypos - starting_press_y;
        //T, S, R, E, C, U
        if (cur_trans_mode == GeoTranslation) {
            models[cur_idx].position.x += diffx * 0.008;
            models[cur_idx].position.y -= diffy * 0.008;
        }
    }
}

void scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
{
    // [TODO] scroll up positive, otherwise it would be negative=====
    // xoffset : The scroll offset along the x - axis
    // yoffset : The scroll offset along the y - axis

    if (cur_trans_mode == GeoTranslation) {
        models[cur_idx].position.z += yoffset * 0.05;
    }
}
```

5. S : scale mode

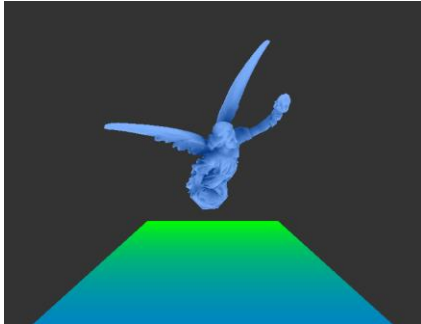
Scaling matrix	
<pre>Matrix4 scaling(Vector3 vec) { Matrix4 mat; //===== mat = Matrix4(vec.x, 0, 0, 0, 0, vec.y, 0, 0, 0, 0, vec.z, 0, 0, 0, 0, 1); //===== return mat; }</pre>	
	

分別根據 cursor 水平與垂直移動距離 update `models[cur_idx].scale.x` 與 `models[cur_idx].scale.y`，z 軸方向則是根據滑鼠滾輪做調整。

< 滑鼠往右(左)移動: 物體水平方向壓縮(放大) >

< 滑鼠往上(下)移動: 物體垂直方向放大(壓縮) >

6. R : rotation mode

rotateX	rotateY
<pre>mat = Matrix4(1, 0, 0, 0, 0, cos(val), -sin(val), 0, 0, sin(val), cos(val), 0, 0, 0, 0, 1);</pre>	<pre>mat = Matrix4(cos(val), 0, sin(val), 0, 0, 1, 0, 0, -sin(val), 0, cos(val), 0, 0, 0, 0, 1);</pre>
rotateZ	
<pre>mat = Matrix4(cos(val), -sin(val), 0, 0, sin(val), cos(val), 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);</pre>	

```
Matrix4 rotate(Vector3 vec)
{
    return rotateX(vec.x)*rotateY(vec.y)*rotateZ(vec.z);
}
```

Rotation matrix 分成 x, y, z 三軸方向，分別寫三個矩陣在相乘。

分別根據 cursor 水平與垂直移動距離 update `models[cur_idx].rotation.y` 與 `models[cur_idx].rotation.x`，z 軸方向則是根據滑鼠滾輪做調整。

< 滑鼠往右(左)移動：物體左右旋轉 >

< 滑鼠往上(下)移動：物體上下旋轉 >

7. E : translate eye position mode

C : translate viewing center position mode

U : translate camera up vector position mode

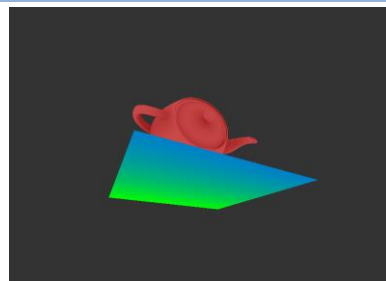
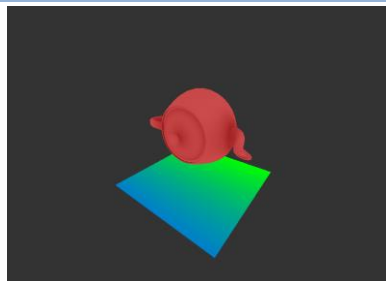
Viewing matrix

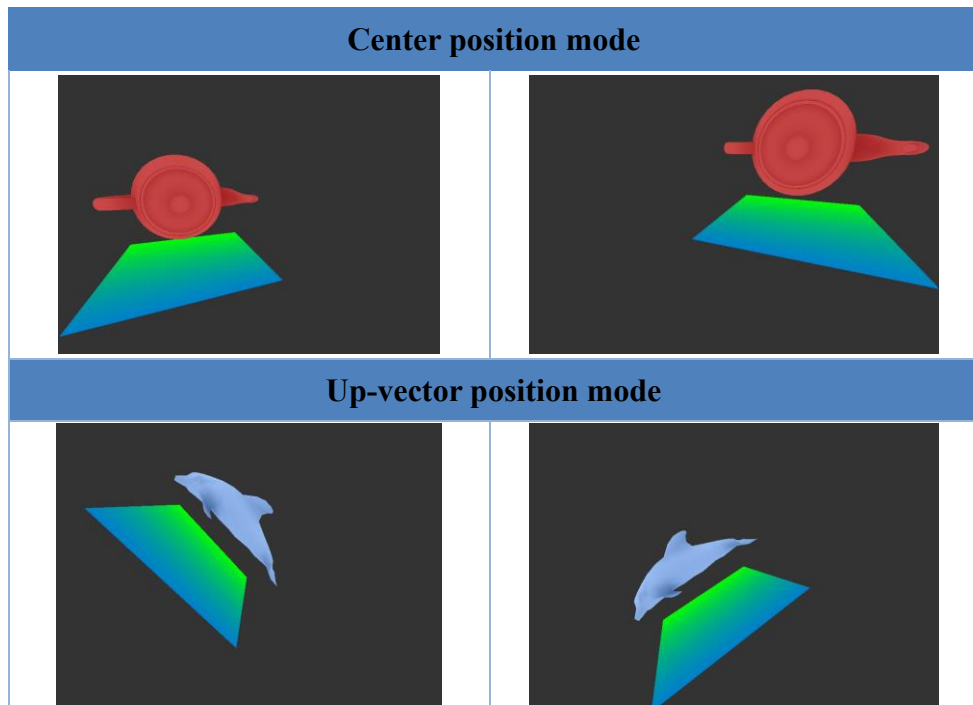
```
A = Matrix4(
    Rx[0], Rx[1], Rx[2], 0,
    Ry[0], Ry[1], Ry[2], 0,
    Rz[0], Rz[1], Rz[2], 0,
    0, 0, 0, 1
);

// translate camera position to the origin
B = Matrix4(
    1, 0, 0, (float)(-1)*main_camera.position.x,
    0, 1, 0, (float)(-1)*main_camera.position.y,
    0, 0, 1, (float)(-1)*main_camera.position.z,
    0, 0, 0, 1
);

view_matrix = A * B;
```

Eye position mode





切換 E、C、U 三種視角皆會呼叫 `setViewingMatrix()` 重新調整 view matrix。

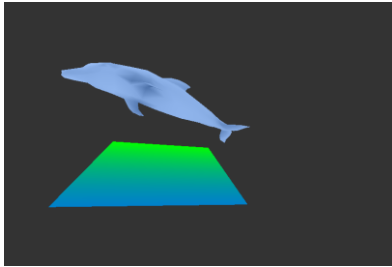
- Eye position :
分別根據 cursor 水平與垂直移動距離 update `main_camera.position.x` 與 `main_camera.position.y`，z 軸方向則是根據滑鼠滾輪做調整。
- Center position :
分別根據 cursor 水平與垂直移動距離 update `main_camera.center.x` 與 `main_camera.center.y`，z 軸方向則是根據滑鼠滾輪做調整。
- Up vector position :
分別根據 cursor 水平與垂直移動距離 update `main_camera.up_vector.x` 與 `main_camera.up_vector.y`，z 軸方向則是根據滑鼠滾輪做調整。

```

if (cur_trans_mode == ViewCenter) {
    main_camera.center.x -= diffx * 0.008;
    main_camera.center.y -= diffy * 0.008;
    setViewingMatrix();
}
if (cur_trans_mode == ViewEye) {
    main_camera.position.x -= diffx * 0.008;
    main_camera.position.y += diffy * 0.008;
    setViewingMatrix();
}
if (cur_trans_mode == ViewUp) {
    main_camera.up_vector.x -= diffx * 0.008;
    main_camera.up_vector.y -= diffy * 0.008;
    setViewingMatrix();
}

```

8. I : print information

example	Show all matrices
	<pre> cur_idx: 4 Translation Matrix: (1, 0, 0, 0.112) (0, 1, 0, 0.168) (0, 0, 1, 0) (0, 0, 0, 1) Rotation Matrix: (0.914713, 0, 0.404104, 0) (-0.29656, 0.679291, 0.671279, 0) (-0.274504, -0.733869, 0.621356, 0) (0, 0, 0, 1) Scaling Matrix: (1.234, 0, 0) (0, 1.042, 0) (0, 0, 1, 0) (0, 0, 0, 1) Viewing Matrix: (0.972068, 0.132864, 0.19347, -0.500511) (-0.0736189, 0.955342, -0.286185, 0.452277) (-0.222853, 0.263949, 0.938439, -2.47617) (0, 0, 0, 1) Projection Matrix: (0.810499, 0, 0, 0) (0, 1.19175, 0, 0) (0, 0, -1.00002, -0.00200002) (0, 0, -1, 0) </pre>

按下 I 鍵印出當下的 cur_idx、Translation matrix、Rotation matrix、Scaling matrix、Viewing matrix 以及 Projection matrix。

◆ Render Scene

1. Multiply all matrices (projection, view, translation, rotation, scaling)

```

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=====
// [TODO] row-major --> column-major
MVP = project_matrix * view_matrix * T * R * S;

```

2. Change to 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];

```

3. Pass column-major matrix to vertex shader

```

// use uniform to send mvp to vertex shader
glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);

```

4. Check whether is Solid or wireframe

```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);  
if (!isSolid) glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
```

5. 即可畫出 model，並呼叫 drawPlane()

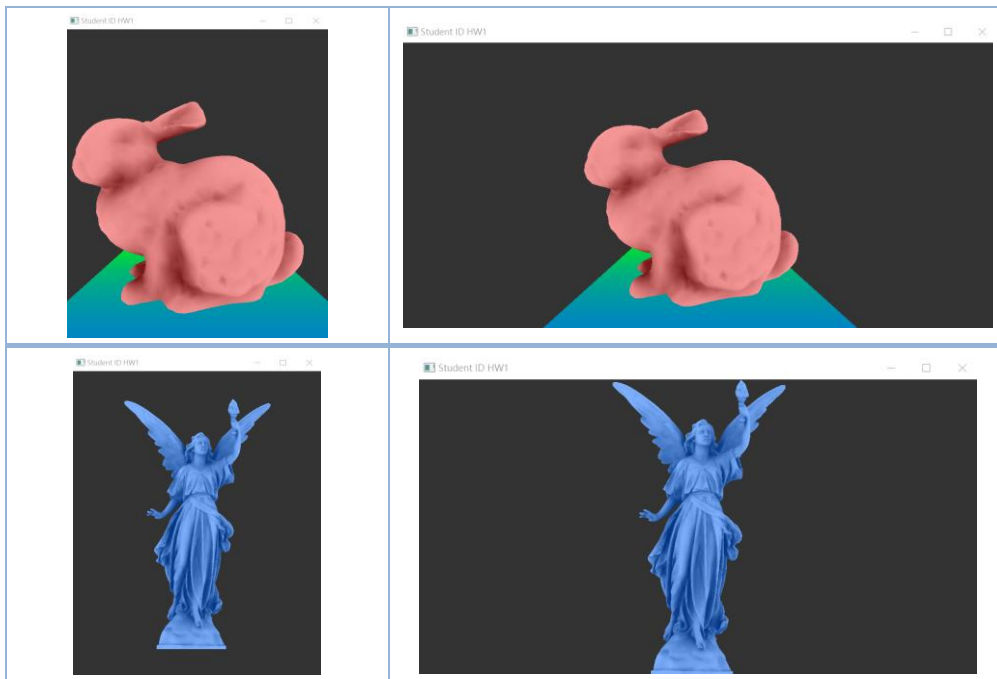
```
glBindVertexArray(m_shape_list[cur_idx].vao);  
glDrawArrays(GL_TRIANGLES, 0, m_shape_list[cur_idx].vertex_count);  
glBindVertexArray(0);  
drawPlane();
```

◆ Draw Plane

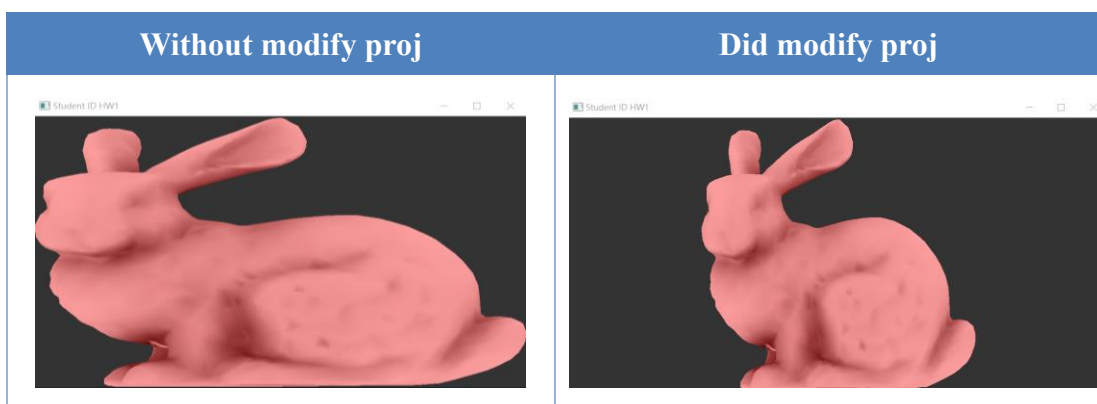
1. 做法與上面畫 model 的方式相似，但因為 plane 不需要跟著做 translation、rotation 以及 scaling，因此只需要將 projection matrix 與 view_matrix 相乘，再以同樣的方式轉成 column-major。
2. 關於 plane 的 vertex 與 color attributes 我新增一個 **Shape plane** 變數來存資料，將 vertex data 存在 plane.vbo、color data 存在 plane.p_color，即可畫出 plane。
3. Plane 不會切換 solid/wireframe mode，因此皆設定為 GL_FILL。

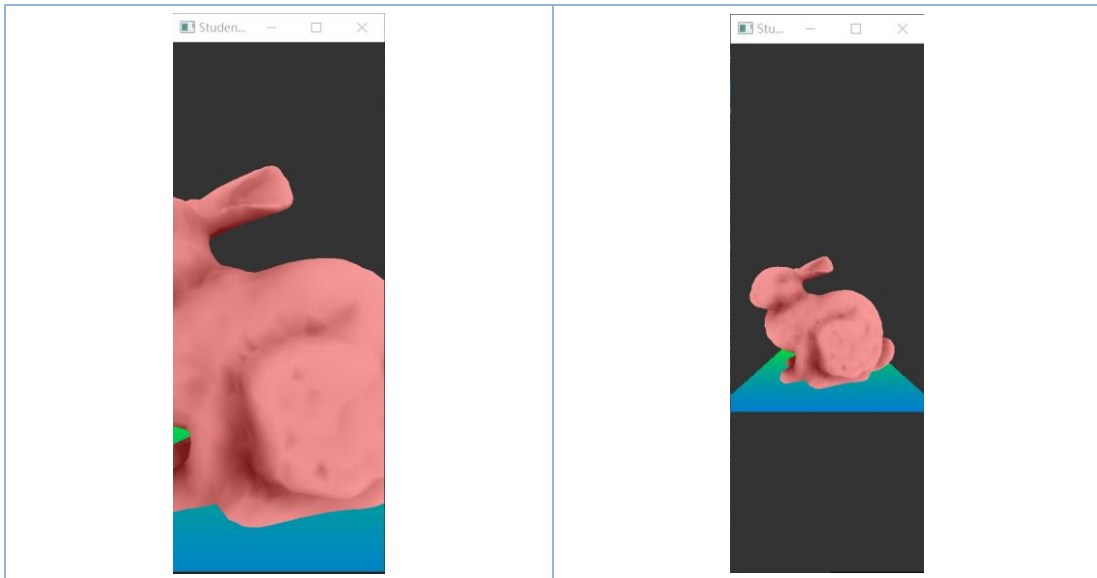
```
glGenVertexArrays(1, &plane.vao);  
glGenBuffers(1, &plane.vbo);  
glGenBuffers(1, &plane.p_color);  
  
glBindVertexArray(plane.vao);  
  
glBindBuffer(GL_ARRAY_BUFFER, plane.vbo);  
glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);  
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);  
glEnableVertexAttribArray(0);  
  
glBindBuffer(GL_ARRAY_BUFFER, plane.p_color);  
glBufferData(GL_ARRAY_BUFFER, sizeof(colors), colors, GL_STATIC_DRAW);  
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);  
glEnableVertexAttribArray(1);  
  
Matrix4 MVP;  
GLfloat mvp[16];  
MVP = project_matrix * view_matrix;  
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];  
  
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);  
glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);  
glBindVertexArray(plane.vao);  
glDrawArrays(GL_TRIANGLES, 0, 6);
```


◆ Change Size



1. 先判斷更改後的 window height 是否為 0，若為 0 則設成 1
2. Update proj.aspect：更改投影的長寬比
3. 在 orthogonal mode 時為了維持物體原本的長寬比不會受到 window size 影響，因此會先調整 proj 的 left、right、top、bottom 的值，再呼叫 setOrthogonal() 重新調整 projection matrix，可以參考下圖比較。
4. 若為 perspective mode 則呼叫 setPerspective() 調整 projection matrix，判斷當下的 $\text{proj.aspect} > 1$ 或 $\text{proj.aspect} \leq 1$ 使用不同的投影矩陣，使 window 長寬比不同時 model 仍然可以保持正常的比例。





```
void ChangeSize(GLFWwindow* window, int width, int height)
{
    if (height == 0) height = 1;
    glViewport(0, 0, width, height);
    // [TODO] change your aspect ratio=====
    // default size (800x600)
    // glViewport : bottom-left point
    //glm::ortho(float left, float right, float bottom, float top, float zNear, float zFar)
    //glm::perspective(float fovy, float aspect, float zNear, float zFar)
    proj.aspect = (float)width / (float)height;
    if (width >= height) {
        proj.left = -((float)width / (float)height);
        proj.right = (float)width / (float)height;
    }
    else {
        proj.bottom = -((float)height / (float)width);
        proj.top = (float)height / (float)width;
    }
    if (cur_proj_mode == Perspective) setPerspective();
    if (cur_proj_mode == Orthogonal) setOrthogonal();
    //=====
}
```

◆ Other work : Press a key to enable self-rotate in y-axis

按下鍵盤 K 鍵即可讓 model 自動旋轉。

利用 boolean 變數 **selfRotate** 紀錄 model 是否要自動旋轉，按下 K 鍵時，將 **selfRotate** 設為 true，在 render 時判斷 **selfRotate** 的 value，若該值為 true 便 update **models[cur_idx].rotation.y**，使物體可以自動旋轉，再按一次 K 鍵即可停止旋轉。

```
else if (key == GLFW_KEY_K && action == GLFW_PRESS) {
    selfRotate = !selfRotate;
}

if (selfRotate==true) {
    models[cur_idx].rotation.y -= 0.003;
}
```