# Computer Graphics Assignment 1 Report

一、Program control instruction

1. 切換 model

   使用 Z、X 鍵來左右變換 model。

2. 切換 solid, wireframe mode

   使用 W 鍵在 solid, wireframe mode 間做切換。

3. 切換 projection mode

   使用 O 鍵變成 orthogonal projection

   使用 P 鍵變成 perspective projection

4. 切換 Geometric Transformation

   使用 T 鍵後，藉由讓滑鼠與滾輪的移動來分別更改 x, y, z 軸的 translation。若按著滑鼠左鍵往左、右移，model 會跟著左右移動。若往上下移，model 也會跟著上下移動。若滾動滾輪，以兩指往上的方式，model 會靠近我們，以兩指往下，model 會遠離我們。

   使用 R 鍵後，藉由讓滑鼠與滾輪的移動來分別更改 x, y, z 軸的 Rotation。若按著滑鼠左鍵往左、右移，model 會跟著左右轉動。若往上下移，model 也會跟著下上轉動。若滾動滾輪，以兩指往上的方式，model 會順時針轉動，以兩指往下，model 會逆時針轉動。

   使用 S 鍵後，藉由讓滑鼠與滾輪的移動來分別更改 x, y, z 軸的 Scaling。若按著滑鼠左鍵往左、右移，model 會跟著以正負 x 軸方向的放大。若往上下移，model 也會跟著以正負 y 軸方向的放大。若滾動滾輪，以兩指往上的方式，model 會在 z 軸負向放大，以兩指往下，model 會在 z 軸正向放大

5. 切換 Viewing Transformation

   使用 E 鍵後，藉由讓滑鼠與滾輪的移動來分別更改 x, y, z 軸的 eye position。

   使用 C 鍵後，藉由讓滑鼠與滾輪的移動來分別更改 x, y, z 軸的 viewing

center position

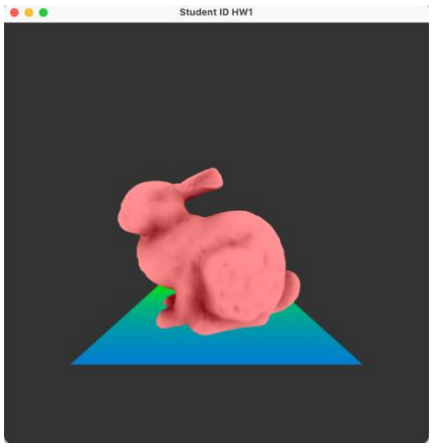
使用 U 鍵後，藉由讓滑鼠的移動來分別更改 x, y, z 軸的 camera up vector position。

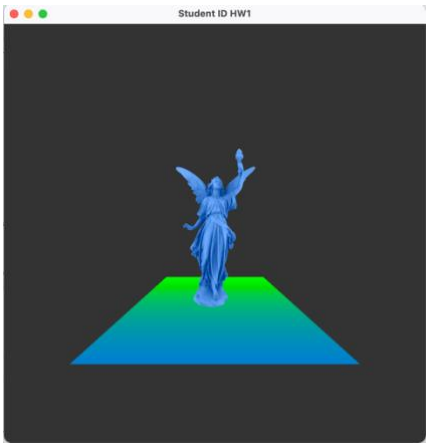
6. Print 出 matrix 資訊

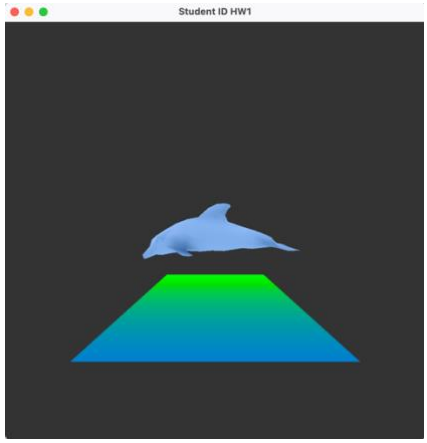
使用 I 鍵，可以在 terminal 看到 translation, rotation, scaling, viewing, projection matrix 的值。

二、Screen shots

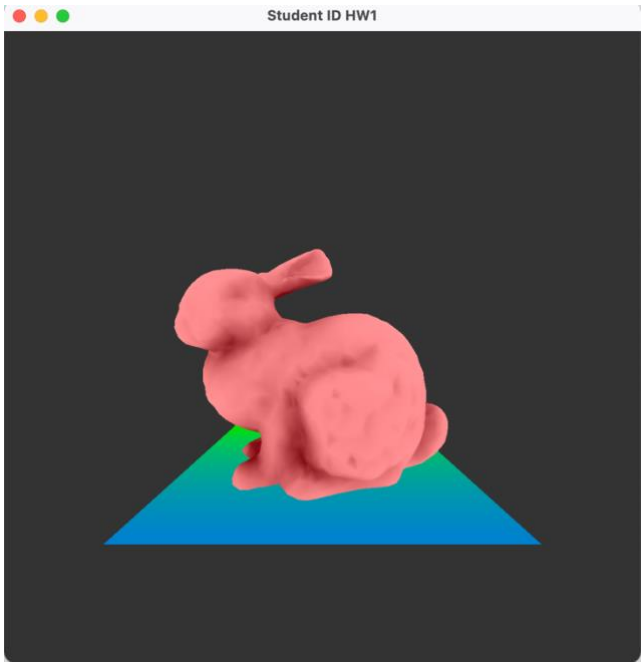
1. 切換 model

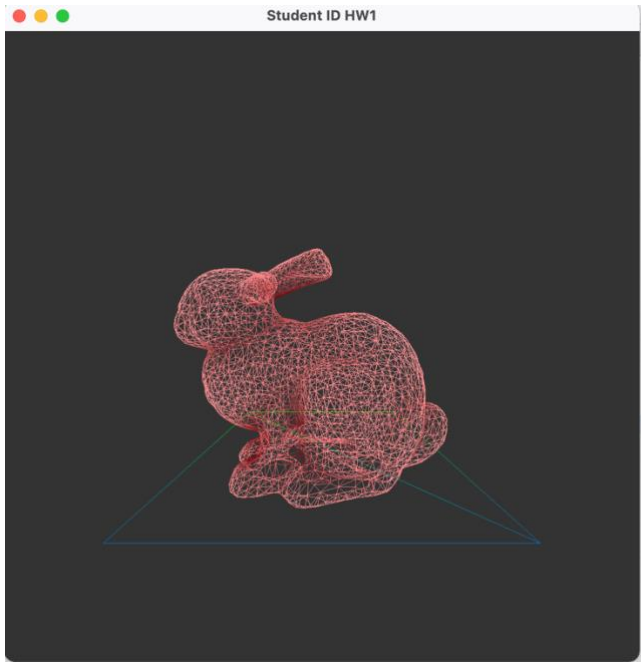




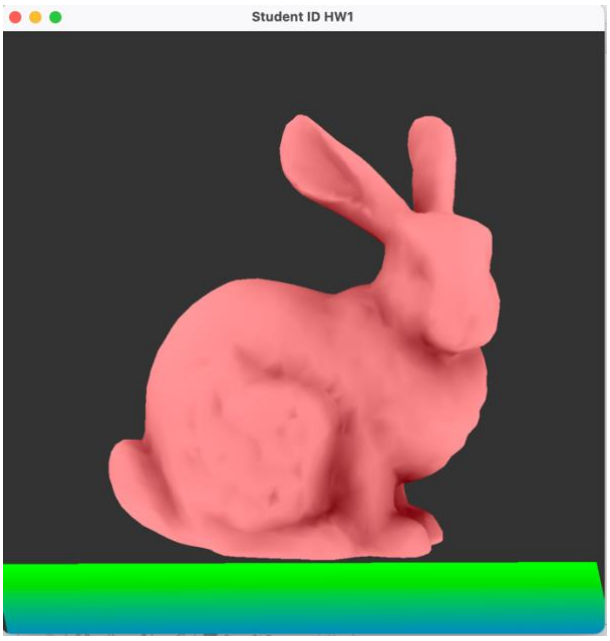


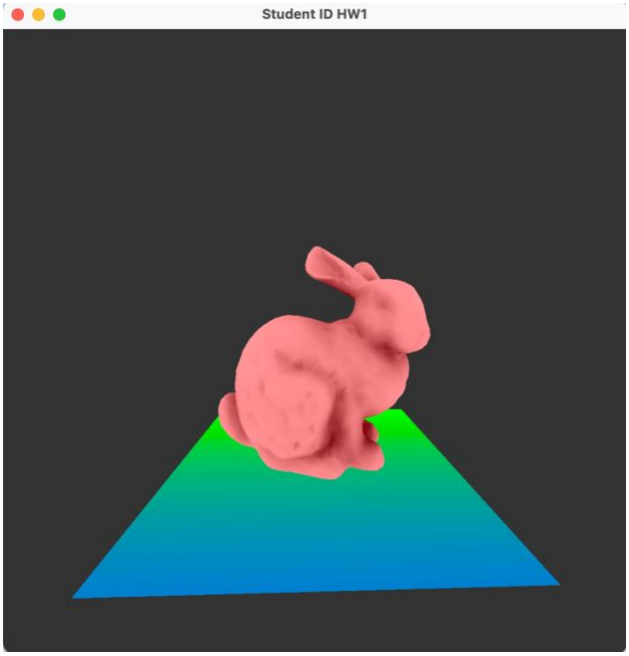

2. Solid vs Wireframe



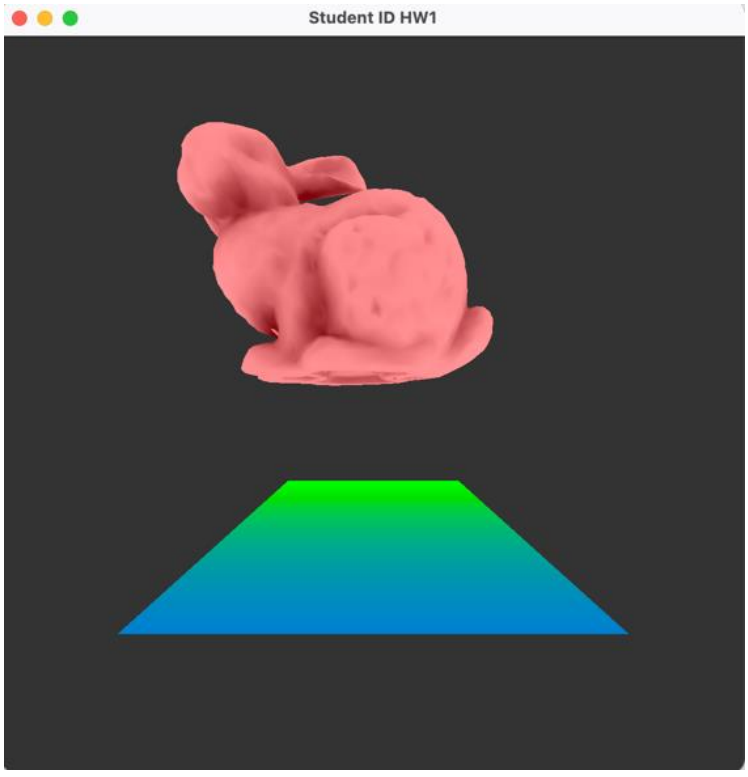


3. Orthogonal vs Perspective

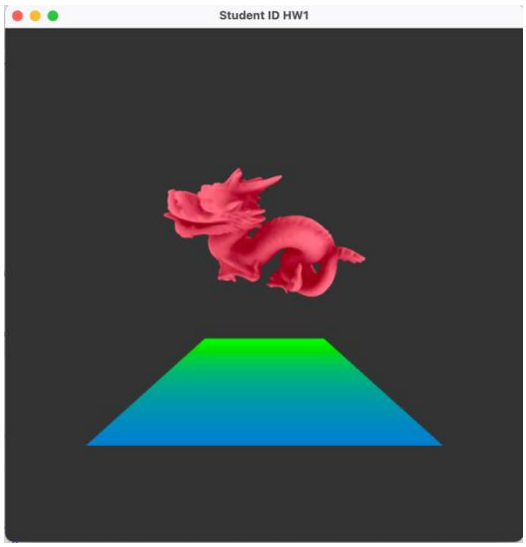


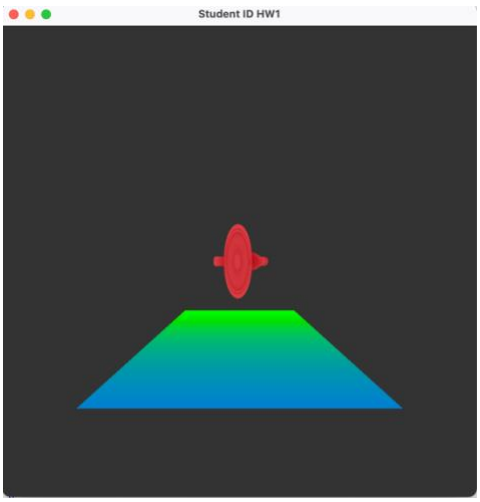

4. Translation

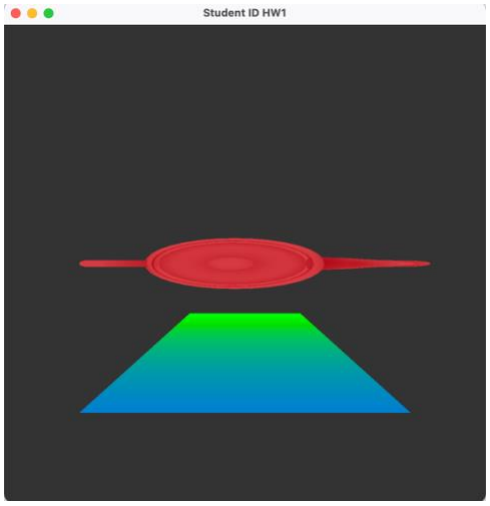


5. Rotation

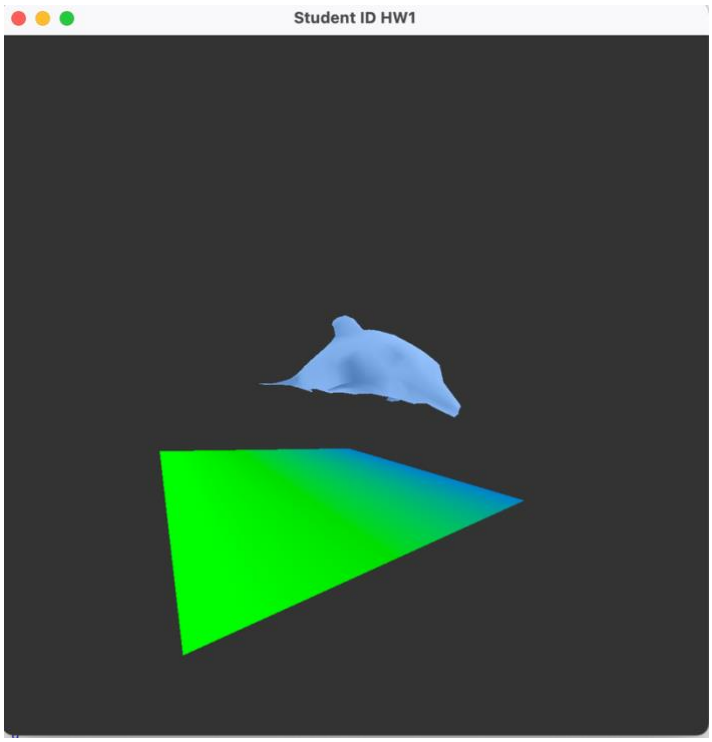



6. Scaling

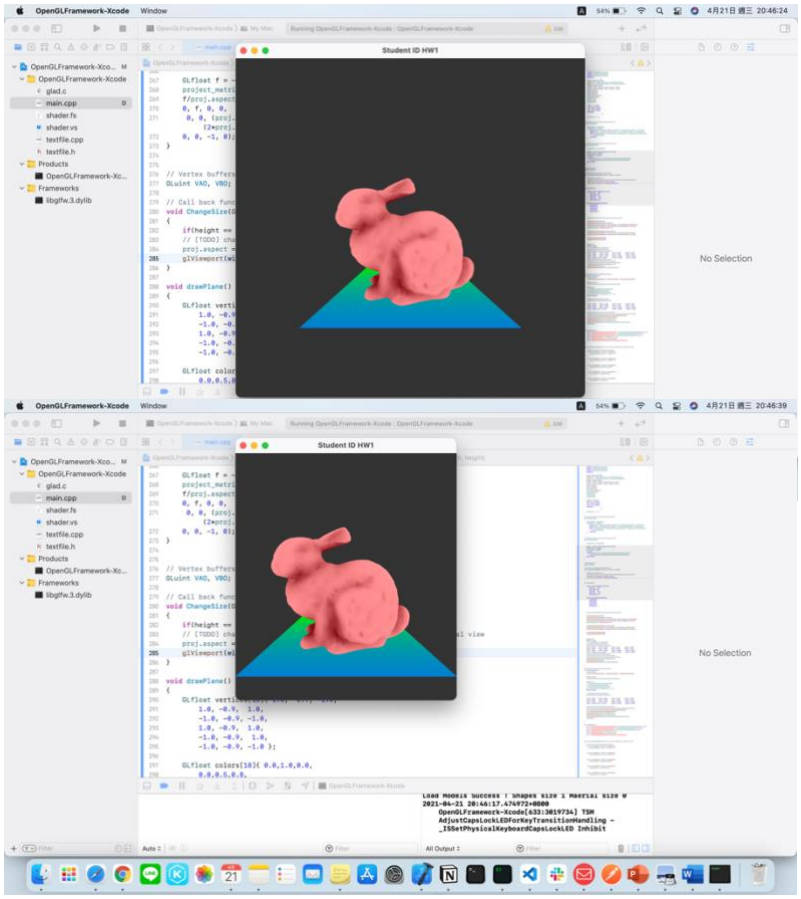


7. Viewing transformation

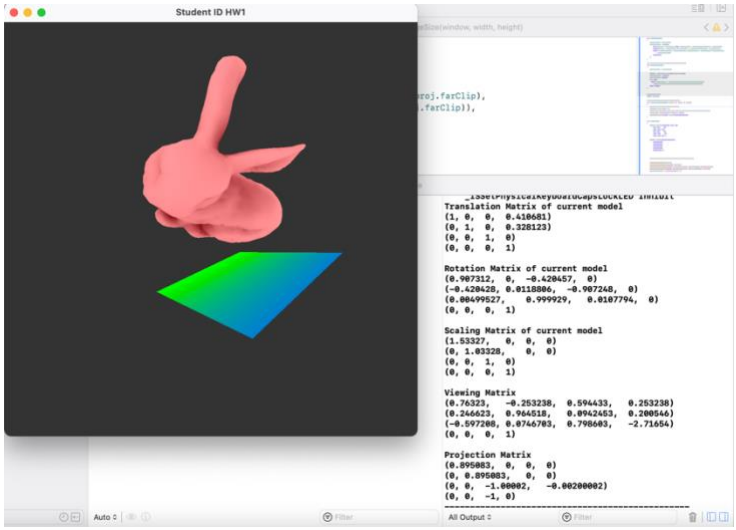



8. Change size



我在網路上有查到，若把視窗最小化的話，會導致 height 變成 0 會有錯誤，不過不知道是不是 xcode 比較不一樣，我將視窗最小化後，也不會進入到 change size 的 callback function，不過在這邊我還是有將若 height = 0 的話，將它設成 1。

9. Output information




## 三、Some code

1. Viewing matrix

```
void setViewingMatrix()
{
    Matrix4 T = Matrix4(
    1, 0, 0, -main_camera.position.x,
    0, 1, 0, -main_camera.position.y,
    0, 0, 1, -main_camera.position.z,
    0, 0, 0, 1
                        );
    GLfloat Rx[3];
    GLfloat p1p2[3], up[3], tmp[3];
    Vector3 P1P2 = main_camera.center - main_camera.position;
    p1p2[0] = P1P2.x; p1p2[1] = P1P2.y; p1p2[2] = P1P2.z;
    tmp[0] = p1p2[0]; tmp[1] = p1p2[1]; tmp[2] = p1p2[2];
    Normalize(tmp);
    Vector3 up_v = main_camera.up_vector;
    up[0] = up_v.x; up[1] = up_v.y; up[2] = up_v.z;
    Normalize(up);
    Cross(tmp, up, Rx);
    Cross(Rx, p1p2, up);
    Normalize(Rx);
    Normalize(up);

    Matrix4 R = Matrix4 (
    Rx[0], Rx[1], Rx[2], 0,
    up[0], up[1], up[2], 0,
    -tmp[0], -tmp[1], -tmp[2], 0,
    0, 0, 0, 1
                        );

    view_matrix = R * T;
}
```

2. Orthogonal project matrix

```
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
);
```

3. Perspective project matrix

```
GLfloat f = -cos(proj.fovy/2)/sin(proj.fovy/2);
project_matrix = Matrix4(
f/proj.aspect, 0, 0, 0,
0, f, 0, 0,
 0, 0, (proj.farClip + proj.nearClip)/(proj.nearClip-proj.farClip),
     (2*proj.farClip*proj.nearClip/(proj.nearClip-proj.farClip)),
0, 0, -1, 0);
```

這部分是與同學討論出來的，雖然講義上的公式是要找 cot，但我們覺得要加一個負號看起來比較合理

4. Wireframe mode

```
if(isDrawWireframe)glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
else glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
```