CS550000 Computer Graphics

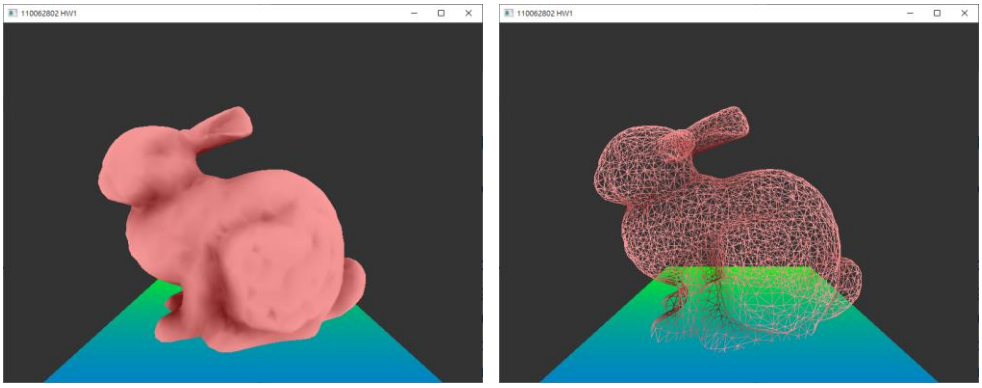
Homework 1 Report: Transformation

110062802 呂宸漢

A. Control instructions and screen shot

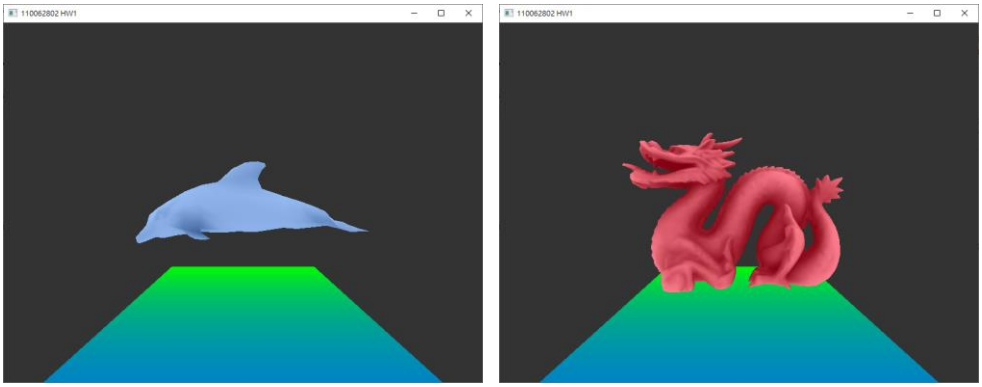
1. 切換 solid mode 和 wireframe mode：

   按 W 鍵在 solid mode 和 wireframe mode 間做切換。



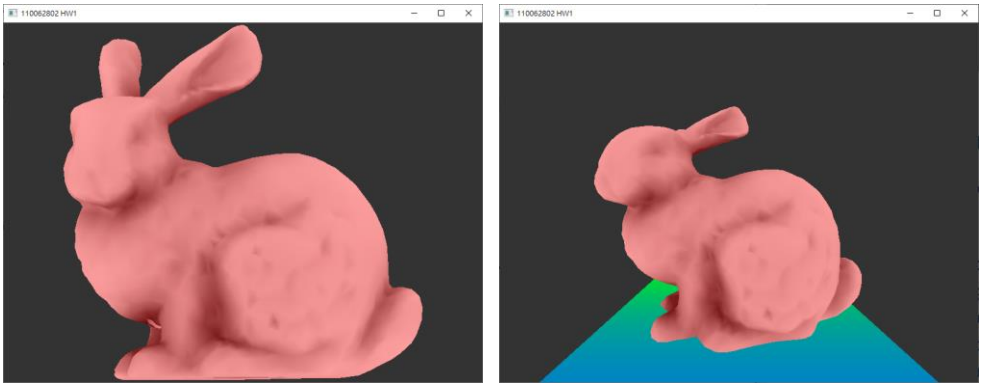
2. 切換 model：

   按 Z 鍵切換成前一個 model；按 X 鍵切換成後一個 model。



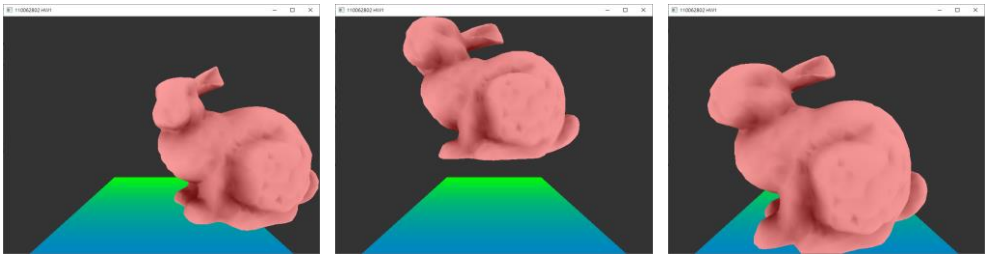
3. 切換 projection mode：

   按 O 鍵切換成 orthogonal projection；按 P 鍵切換成 perspective projection。



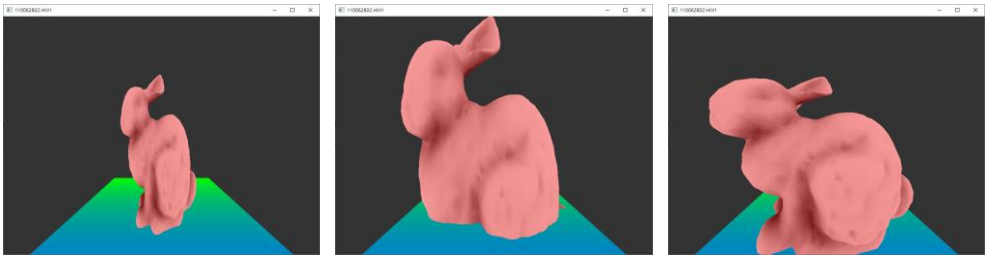
4. 移動 model：

按 T 鍵後使用滑鼠左鍵拖曳 model 改變 x 及 y 軸的 translation，使用滑鼠滾輪改變 z 軸的 translation。如：滑鼠向右拖曳 model 向右移動、滑鼠向上拖曳 model 向上移動、滾輪向前滾動 model 向前移動。



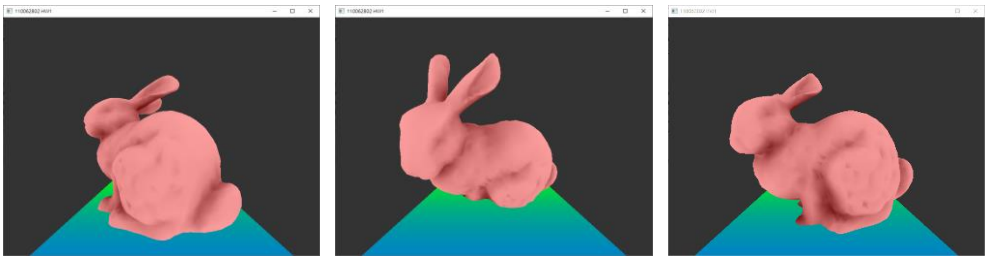
5. 縮放 model：

按 S 鍵後使用滑鼠左鍵拖曳 model 改變 x 及 y 軸的 scale，使用滑鼠滾輪改變 z 軸的 scale。如：滑鼠向右拖曳 model 以 x 軸縮小、滑鼠向上拖曳 model 以 y 軸放大、滾輪向前滾動 model 以 z 軸放大。



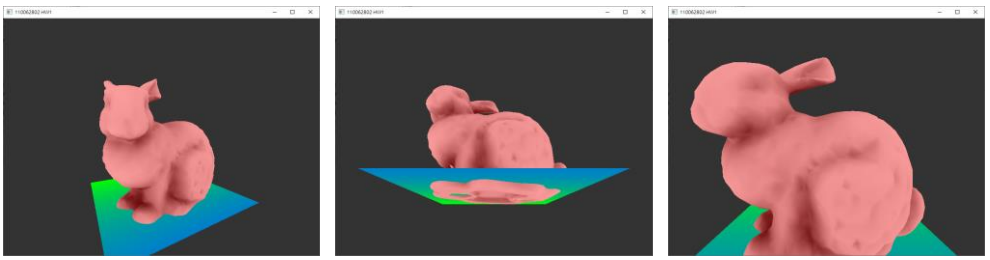
6. 旋轉 model：

按 R 鍵後使用滑鼠左鍵拖曳 model 讓 model 沿 x 軸及 y 軸旋轉，使用滑鼠滾輪讓 model 沿 z 軸旋轉。如：滑鼠向右拖曳 model 向右旋轉、滑鼠向上拖曳 model 向前旋轉、滾輪向前滾動 model 逆時針旋轉。



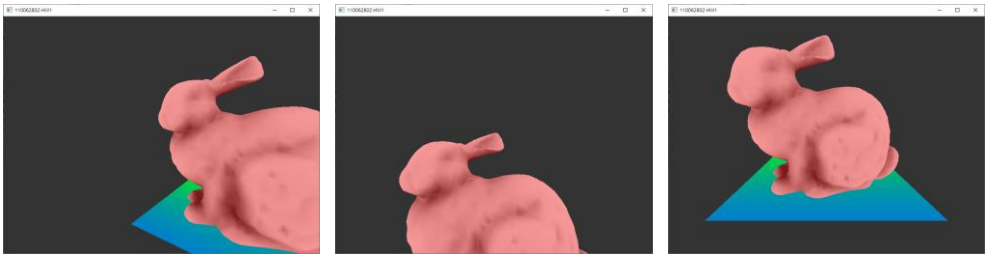
7. 改變 eye position：

按 E 鍵後使用滑鼠左鍵拖曳 model 改變 x 及 y 軸的 eye position，使用滑鼠滾輪改變 z 軸的 eye position。如：向右拖曳、向上拖曳、向前滾動。



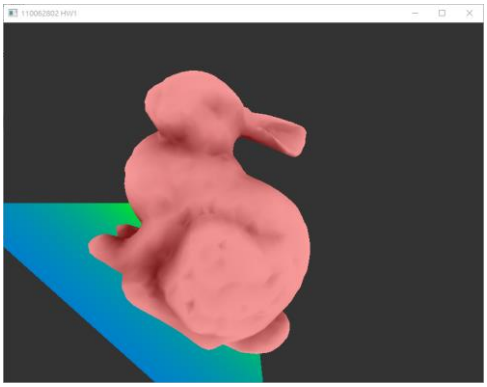
8. 改變 viewing center position：

按C鍵後使用滑鼠左鍵拖曳model改變x及y軸的viewing center position，使用滑鼠滾輪改變z軸的 viewing center position。如：向右拖曳、向上拖曳、向前滾動。



9. 改變 camera up vector position：

按 U 鍵後使用滑鼠左鍵拖曳 model 改變 x 及 y 軸的 camera up vector position，使用滑鼠滾輪改變 z 軸的 camera up vector position。如：向右拖曳。



10. 列印 matrix 資訊：

按 I 鍵列印當前的 matrix 資訊。

```
Matrix Value:
Viewing Matrix:
(1,      0,      0,      0)
(0,      1,      0,      0)
(0,      0,      1,      -2)
(0,      0,      0,      1)

Projection Matrix:
(0.893815,       0,      0,      0)
(0,      1.19175,        0,      0)
(0,      0,      -1.00002,       -0.00200002)
(0,      0,      -1,     0)

Translation Matrix:
(1,      0,      0,      0)
(0,      1,      0,      0)
(0,      0,      1,      0)
(0,      0,      0,      1)

Rotation Matrix:
(1,      0,      0,      0)
(0,      1,      0,      0)
(0,      0,      1,      0)
(0,      0,      0,      1)

Scaling Matrix:
(1,      0,      0,      0)
(0,      1,      0,      0)
(0,      0,      1,      0)
(0,      0,      0,      1)
```

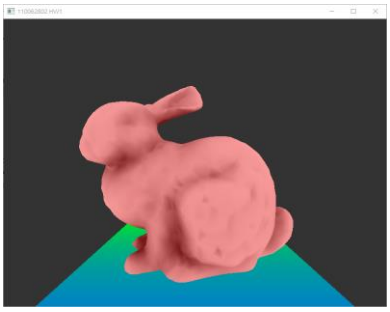
B. Special things

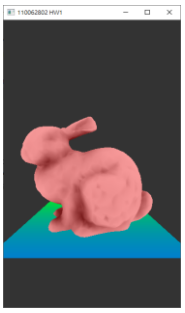
1. 在 drawPlane 內把 load plane 的 code 與 draw plane 的 code 分開。在 loadPlane 時模仿 LoadModels 建立 array 及 buffer，將對應的 vertex 與 color 傳上去，由於這個部分只需要執行一次就好，不用放在 drawPlane 內一直 call，因此我將 loadPlane 放在 setupRC 內呼叫。在 drawPlane 時模仿 RenderScene 計算 MVP 再將資料傳給 render 即可。

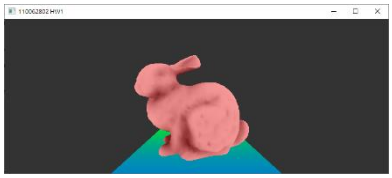
```
void loadPlane()
{
    GLfloat vertices[18]{ 1.0, -0.9, -1.0,
                          1.0, -0.9,  1.0,
                         -1.0, -0.9, -1.0,
                          1.0, -0.9,  1.0,
                         -1.0, -0.9,  1.0,
                         -1.0, -0.9, -1.0};

    GLfloat colors[18]{0.0, 1.0, 0.0,
                       0.0, 0.5, 0.8,
                       0.0, 1.0, 0.0,
                       0.0, 0.5, 0.8,
                       0.0, 0.5, 0.8,
                       0.0, 1.0, 0.0};

    /* modify from "LoadModels" */
    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 = sizeof(vertices) / sizeof(GLfloat) / 3;

    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);
}
```

```
void drawPlane()
{
    // [TODO] draw the plane with above vertices and color
    /* For loading plane, please refer to "loadPlane". */

    /* let plane be solid */
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

    /* modify from "RenderScene" */
    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];

    glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);
    glBindVertexArray(quad.vao);
    glDrawArrays(GL_TRIANGLES, 0, quad.vertex_count);
}
```

2. 改變視窗大小 model 必須隨之縮放。在視窗最小化時視窗的 width 與 height 會是 0，所以在算 aspect ratio 時要特別注意。

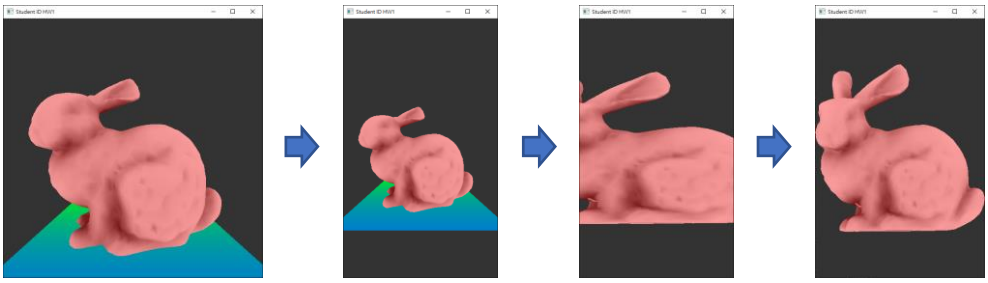






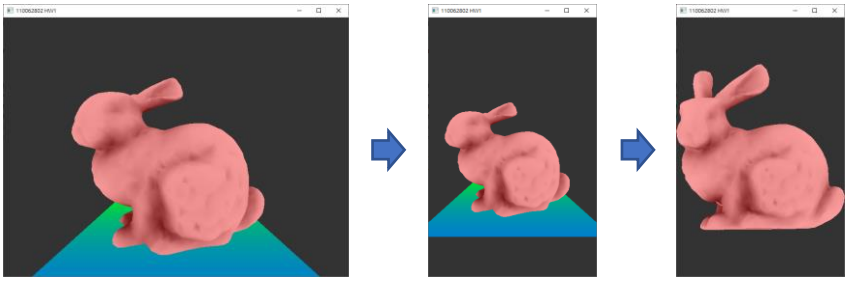
```
// Call back function for window reshape
void ChangeSize(GLFWwindow *window, int width, int height)
{
    glViewport(0, 0, width, height);
    // [TODO] change your aspect ratio
    if (width == 0 || height == 0)
        return;

    proj.aspect = static_cast<float>(width) / height;
    if (cur_proj_mode == ProjMode::Orthogonal)
    {
        if (proj.aspect >= 1)
            project_matrix[0] = 2 / (proj.right - proj.left) / proj.aspect;
        else
            project_matrix[5] = 2 / (proj.top - proj.bottom) * proj.aspect;
    }
    else if (cur_proj_mode == ProjMode::Perspective)
    {
        float f = cos(degree2radian(proj.fovy) / 2) / sin(degree2radian(proj.fovy) / 2);
        if (proj.aspect >= 1)
            project_matrix[0] = f / proj.aspect;
        else
            project_matrix[5] = f * proj.aspect;
    }
}
```

3. 在與助教的 result 比對結果時我發現助教的 code 在 perspective projection 下先改變視窗大小再換成 orthogonal projection 時 model 會變形，直到再次改變視窗大小才會變成正常的形狀。如下圖所示：



由於我的 code 也會有一樣的問題，因此我便 trace 我的 code 以找出有問題的地方。在 trace 完我的 code 後我認為是因為在變更視窗大小時，我們會利用當前視窗的 width 與 height 計算 aspect ratio 並更新對應的 projection matrix，可是在換成別的 projection mode 時，新的 projection matrix 並沒有依照當前視窗大小的 aspect ratio 更新 projection matrix 才導致 model 會在換 mode 後變形。

為了解決此問題，我在 `setOrthogonal` 與 `setPerspective` 內額外加入 `ChangeSize` 內用 aspect ratio 更新 projection matrix 的 function，這樣就可以在更換 mode 時即時更新新的 projection matrix。



```
// [TODO] compute orthogonal projection matrix
void setOrthogonal()
{
    cur_proj_mode = Orthogonal;

    float x = 2 / (proj.right - proj.left);
    float y = 2 / (proj.top - proj.bottom);
    float z = -2 / (proj.farClip - proj.nearClip);
    float t_x = -(proj.right + proj.left) / (proj.right - proj.left);
    float t_y = -(proj.top + proj.bottom) / (proj.top - proj.bottom);
    float t_z = -(proj.farClip + proj.nearClip) / (proj.farClip - proj.nearClip);

    project_matrix = Matrix4(
        x, 0, 0, t_x,
        0, y, 0, t_y,
        0, 0, z, t_z,
        0, 0, 0,   1);

    if (proj.aspect >= 1)
        project_matrix[0] = x / proj.aspect;
    else
        project_matrix[5] = y * proj.aspect;
}
```

```
// [TODO] compute persepective projection matrix
void setPerspective()
{
    cur_proj_mode = Perspective;

    float f = cos(degree2radian(proj.fovy) / 2) / sin(degree2radian(proj.fovy) / 2);
    float z = (proj.farClip + proj.nearClip) / (proj.nearClip - proj.farClip);
    float t_z = (2 * proj.farClip * proj.nearClip) / (proj.nearClip - proj.farClip);

    project_matrix = Matrix4(
        f, 0,  0,   0,
        0, f,  0,   0,
        0, 0,  z, t_z,
        0, 0, -1,   0);

    if (proj.aspect >= 1)
        project_matrix[0] = f / proj.aspect;
    else
        project_matrix[5] = f * proj.aspect;
}
```