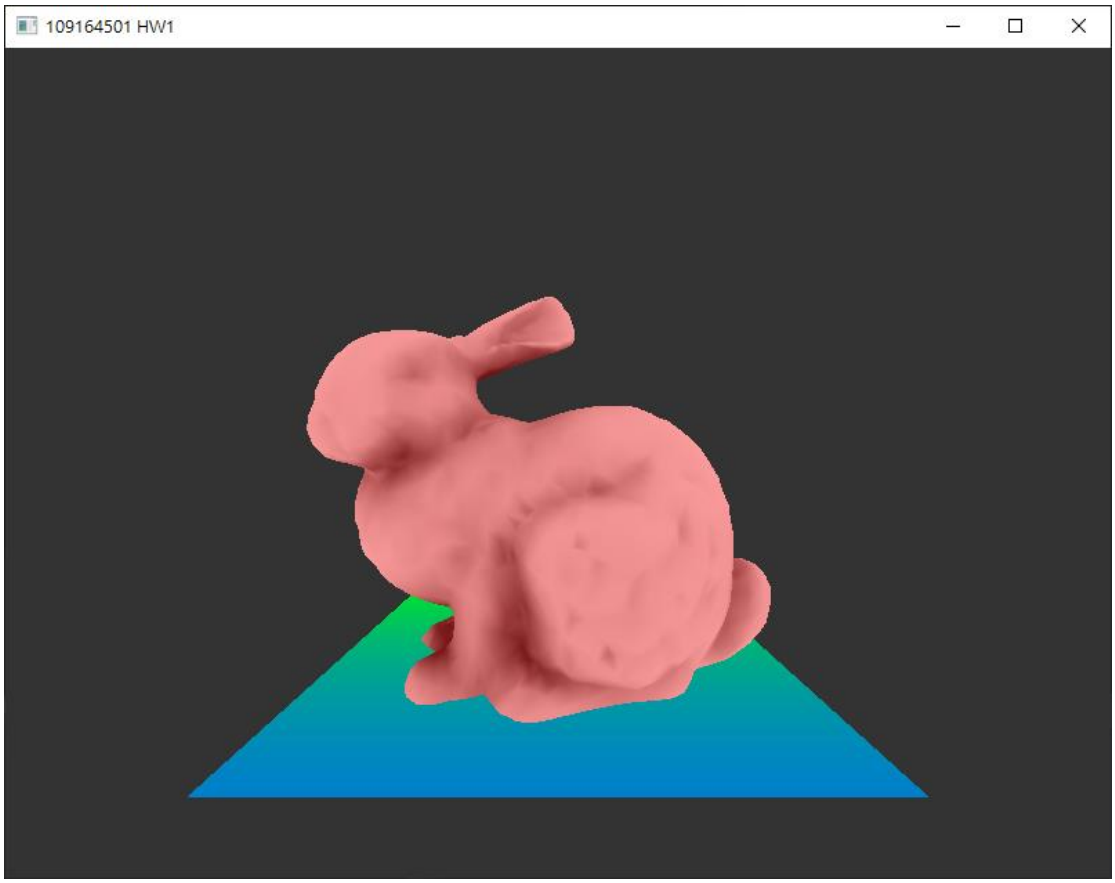
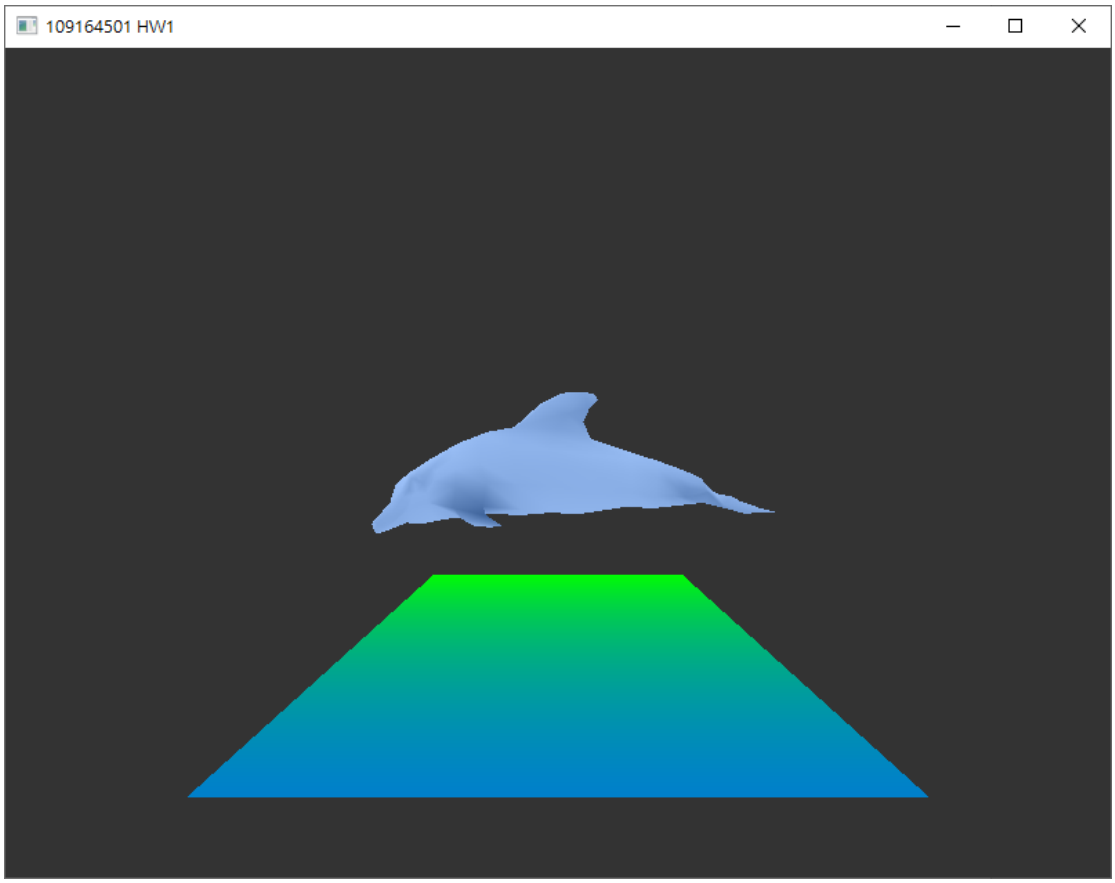
109164501 陸柏宏 計算機圖學 HW1 Report

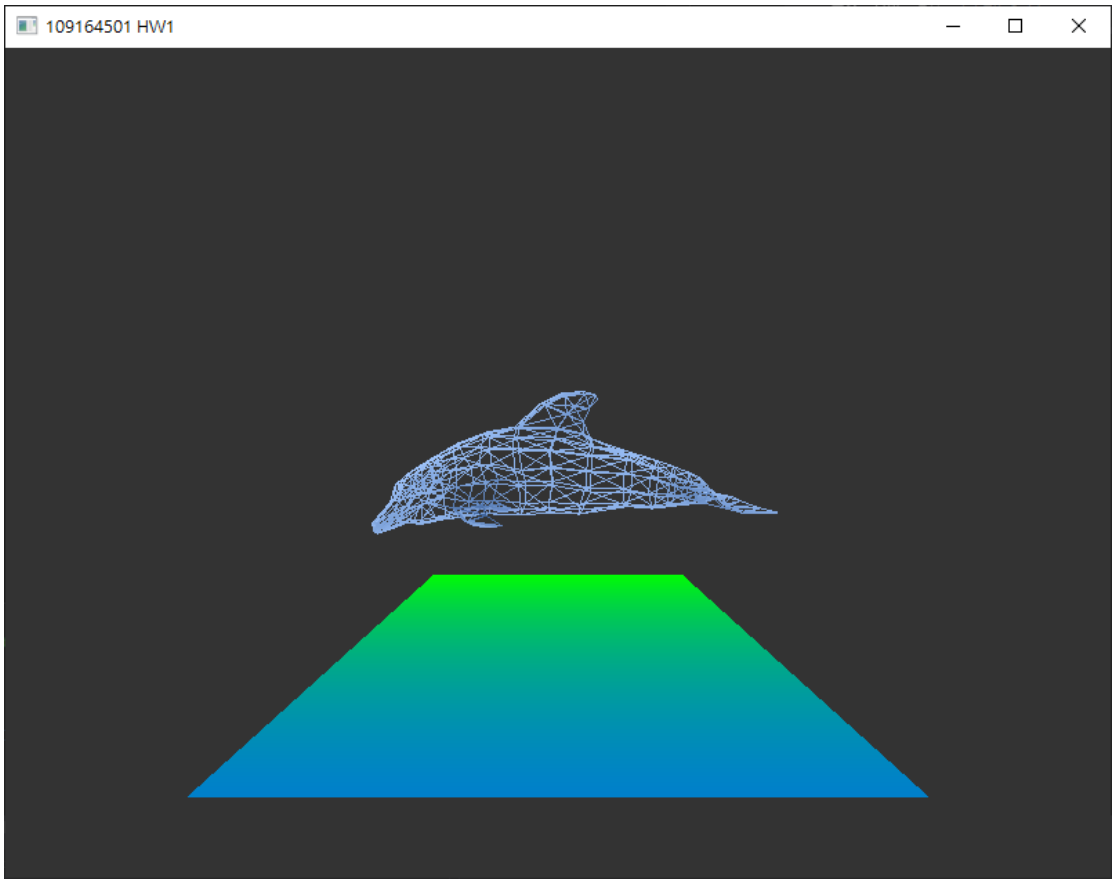
初始畫面



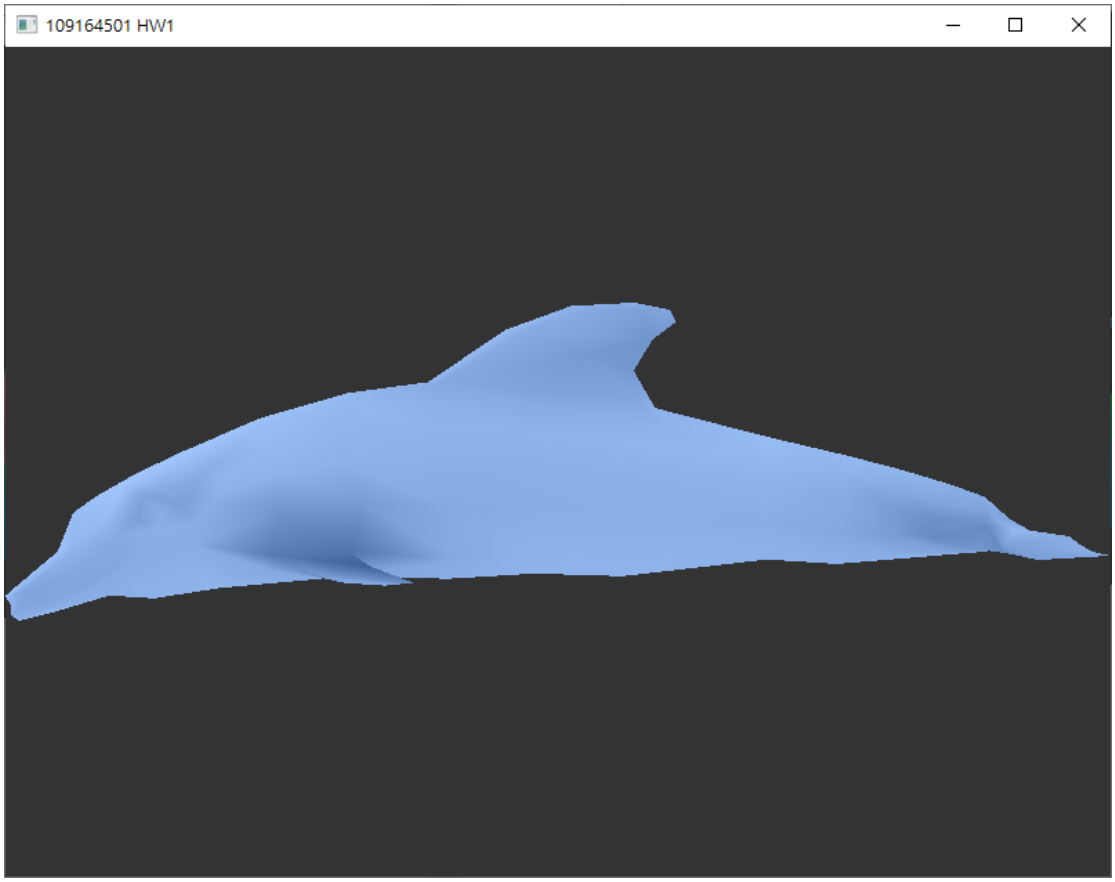
z/x 切換模型



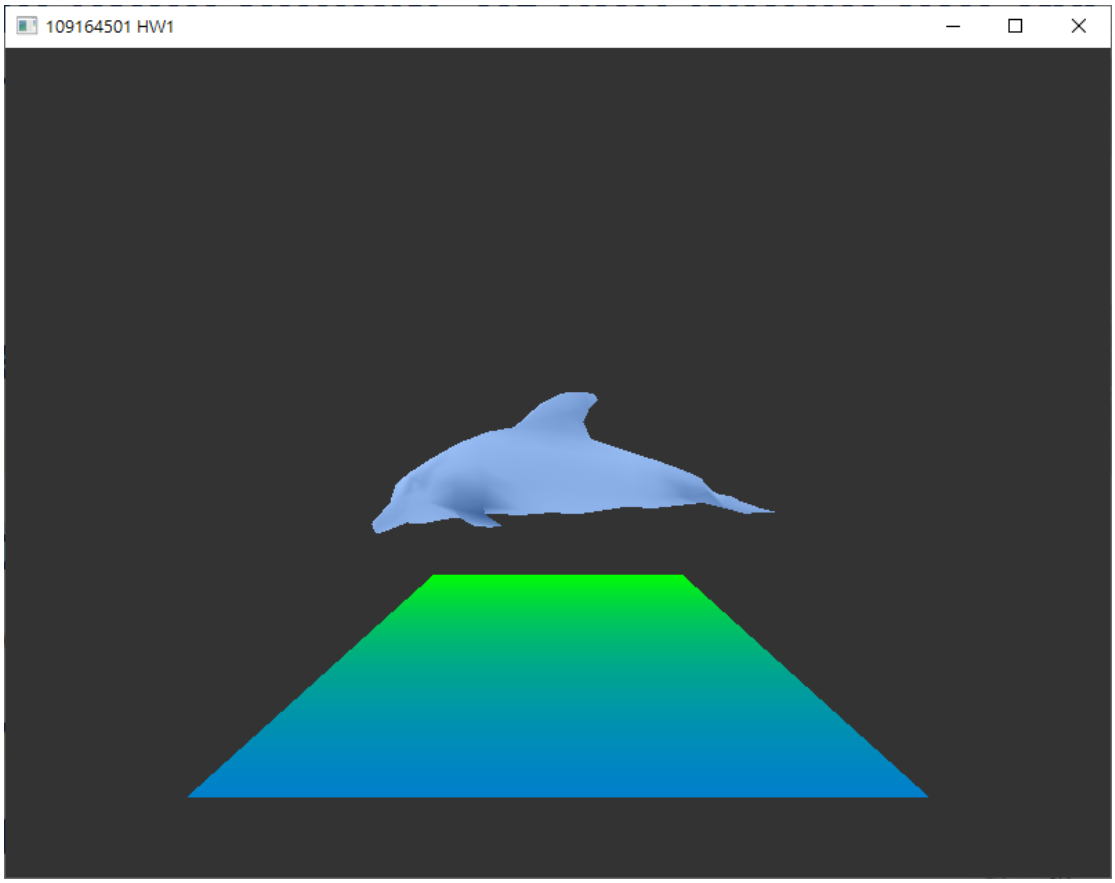
Wireframe



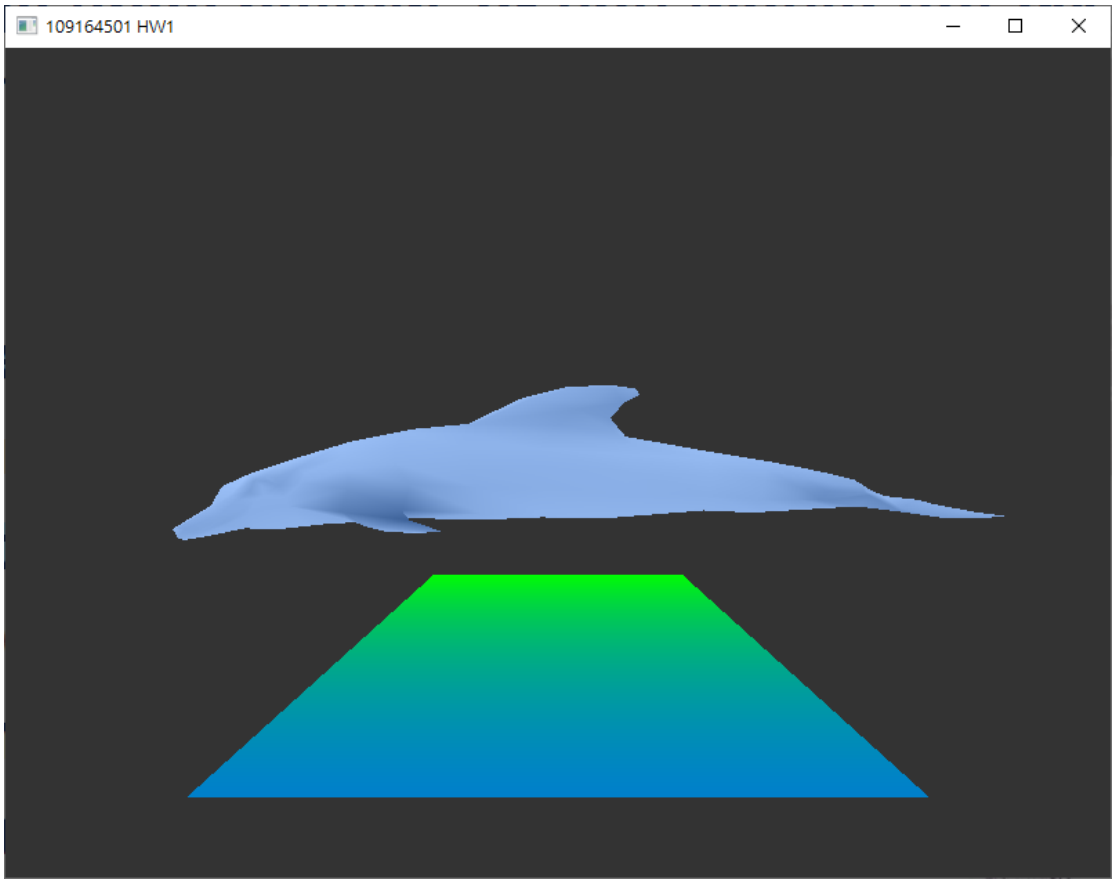
Orthogonal



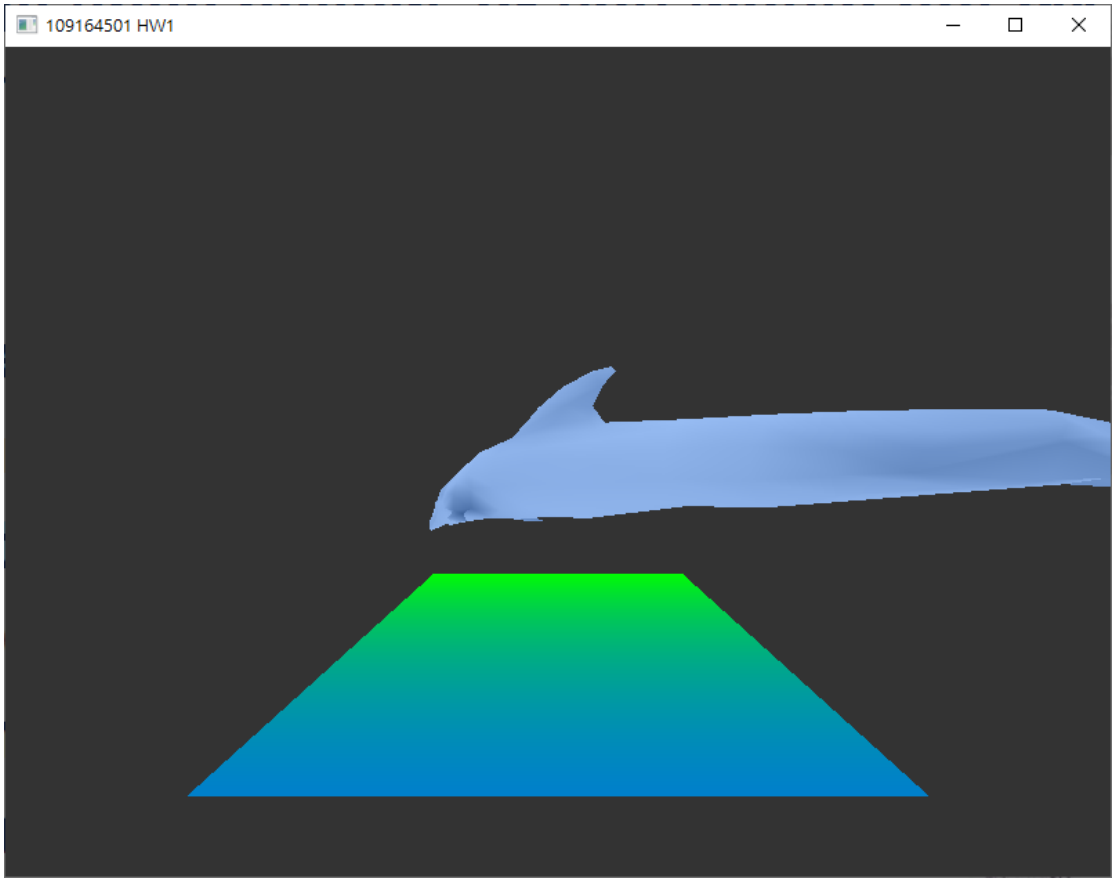
NDC Perspective



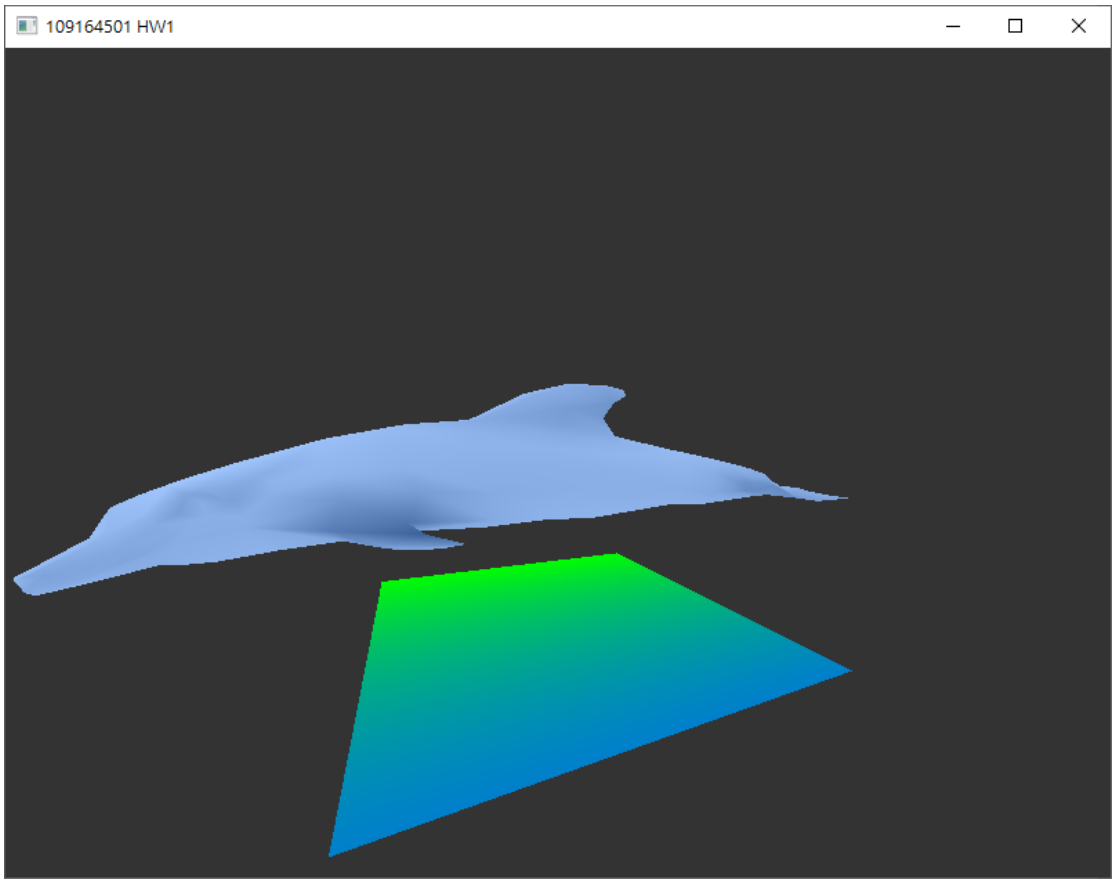
Scale



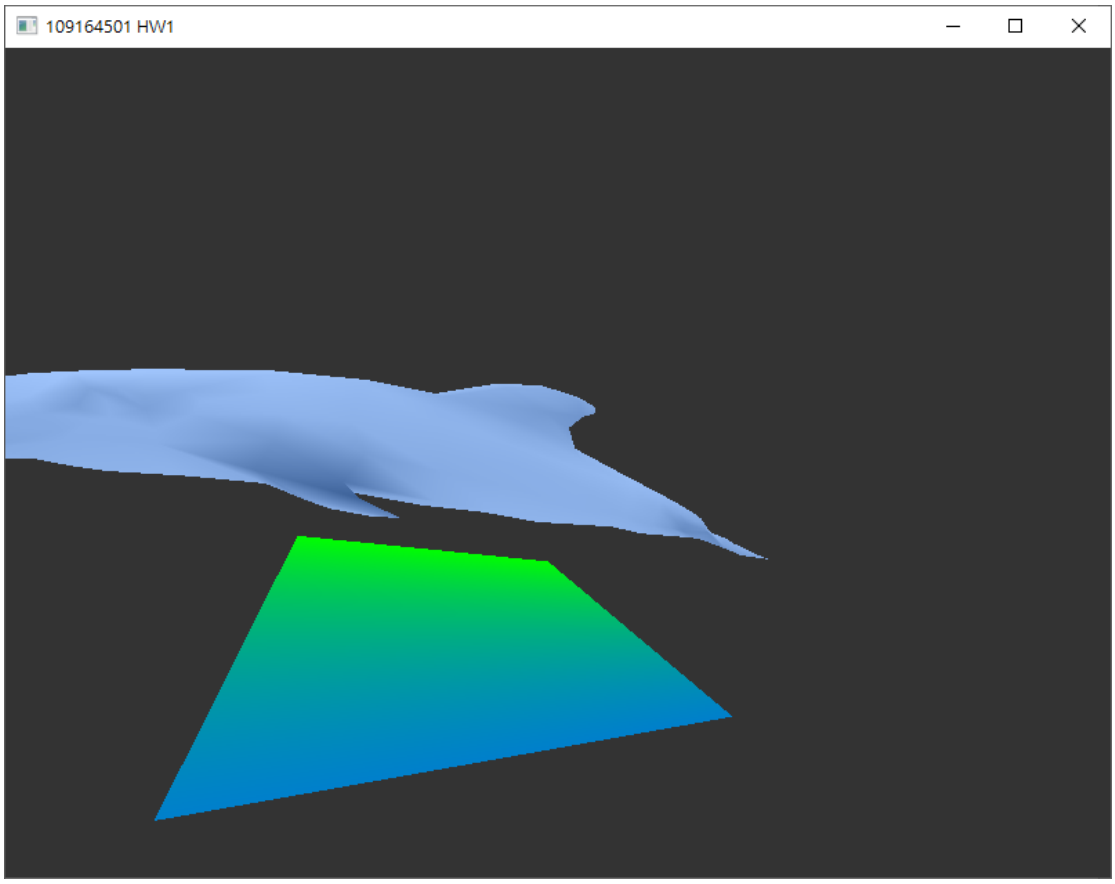
Rotation



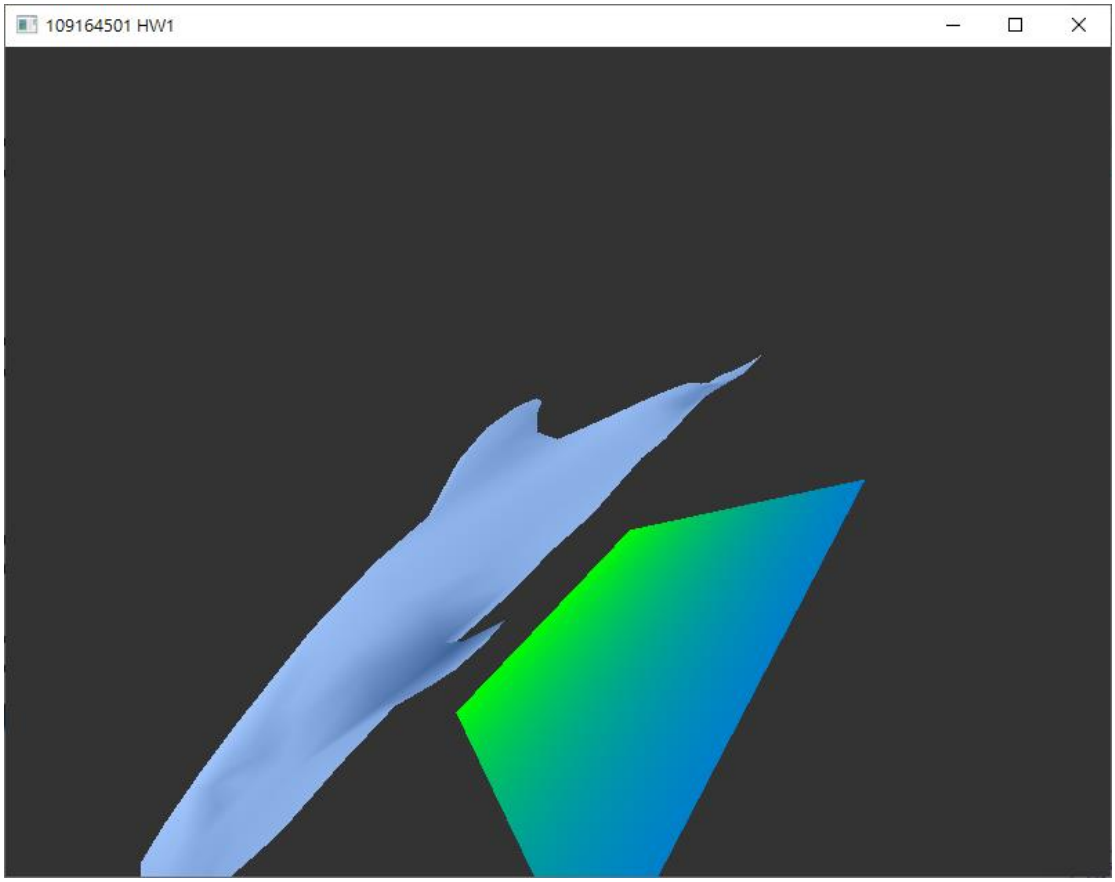
Eye position mode



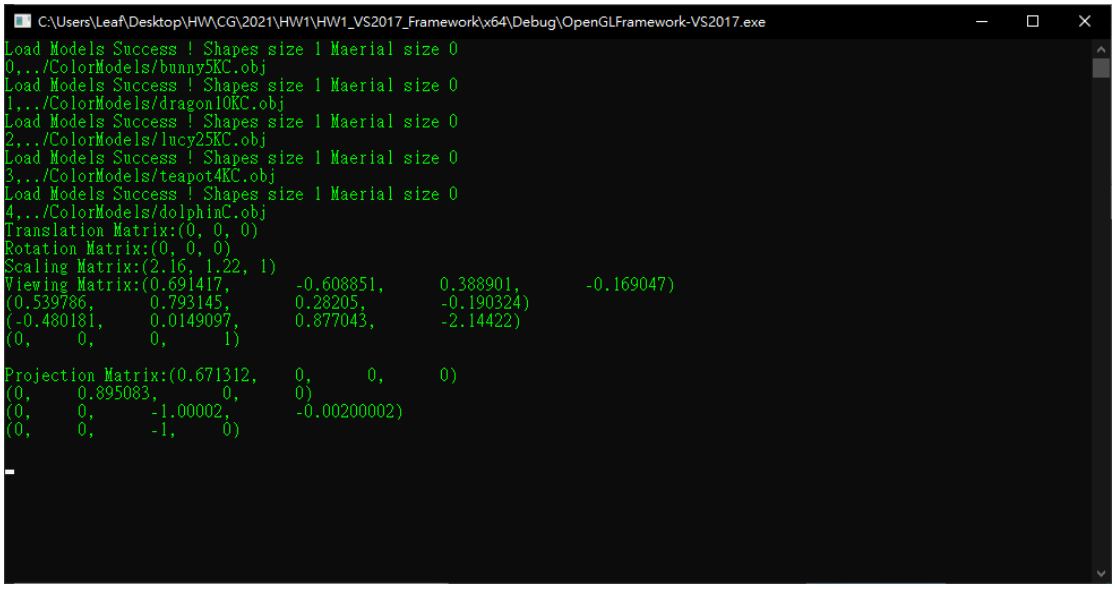
Viewing center position mode



translate camera up vector position mode



Information



鍵盤 Keys

```
switch (key) {
    case GLFW_KEY_W:
        /*solid and wireframe mode*/
        if(action == 1) isDrawWireframe = !isDrawWireframe;
        break;
    case GLFW_KEY_Z:
        /*switch the model (-1)*/
        if (action == 1)
            cur_idx = (cur_idx - 1 >= 0) ? cur_idx - 1 : model_cnt - 1;
        break;
    case GLFW_KEY_X:
        /*switch the model (+1)*/
        if(action == 1)
            cur_idx = (cur_idx + 1 < model_cnt) ? cur_idx + 1 : 0;
        break;
    case GLFW_KEY_O:
        /*switch to Orthogonal projection*/
        setOrthogonal();
        break;
    case GLFW_KEY_P:
        /*switch to NDC Perspective projection*/
        setPerspective();
        break;
    case GLFW_KEY_T:
        /*switch to translation mode*/
        cur_trans_mode = GeoTranslation;
        break;
    case GLFW_KEY_S:
```

```
    /*switch to scale mode*/
    cur_trans_mode = GeoScaling;
    break;
case GLFW_KEY_R:
    /*switch to rotation mode*/
    cur_trans_mode = GeoRotation;
    break;
case GLFW_KEY_E:
    /*switch to translate eye position mode*/
    cur_trans_mode = ViewEye;
    break;
case GLFW_KEY_C:
    /*switch to translate viewing center position mode*/
    cur_trans_mode = ViewCenter;
    break;
case GLFW_KEY_U:
    /*switch to translate camera up vector position mode*/
    cur_trans_mode = ViewUp;
    break;
case GLFW_KEY_I:
    /*print information*/
    if (action == 1)
        info();
    break;
case GLFW_KEY_ESCAPE:
    if(action == 1)
        glfwSetWindowShouldClose(window, GL_TRUE);
    break;
```

Scale 矩陣

```
// [TODO#] given a scaling vector then output a Matrix4 (Scaling Matrix)
Matrix4 scaling(Vector3 vec)
{
    Matrix4 mat;


    mat = Matrix4(
        vec[0],0,0,0,
        0,vec[1],0,0,
        0,0,vec[2],0,
        0,0,0,1
    );


    return mat;
}
```

Rotate 矩陣

```
  // [TODO#] given a float value then ouput a rotation matrix alone axis-X (rotate alone axis-X)
Matrix4 rotateX(GLfloat val)
{
    Matrix4 mat;


    mat = Matrix4(
        1,0,0,0,
        0,cos(val),-sin(val),0,
        0,sin(val),cos(val),0,
        0,0,0,1
    );


    return mat;
}
```

```
// [TODO#] given a float value then ouput a rotation matrix alone axis-Y (rotate alone axis-Y)
Matrix4 rotateY(GLfloat val)
{
    Matrix4 mat;


    mat = Matrix4(
        cos(val),0,sin(val),0,
        0,1,0,0,
        -sin(val),0,cos(val),0,
        0,0,0,1
    );


    return mat;
}
```

```
// [TODO#] given a float value then ouput a rotation matrix alone axis-Z (rotate alone axis-Z)
Matrix4 rotateZ(GLfloat val)
{
    Matrix4 mat;


    mat = Matrix4(
        cos(val),-sin(val),0,0,
        sin(val),cos(val),0,0,
        0,0,1,0,
        0,0,0,1
    );


    return mat;
}
```

Viewing matrix

```
// [TODO] compute viewing matrix accroding to the setting of main_camera
void setViewingMatrix()
{
    Vector3 p1_p2, p1_p3;
    Vector3 Rx, Ry, Rz;
    Matrix4 T;

    p1_p2 = (main_camera.center - main_camera.position);
    p1_p3 = (main_camera.up_vector - main_camera.position);

    Rz = p1_p2;
    Rz = -Rz / Rz.length();

    Rx = p1_p2.cross(p1_p3);
    Rx = Rx / Rx.length();

    Ry = Rz.cross(Rx);


    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
    );

    view_matrix[0] = Rx[0];
    view_matrix[1] = Rx[1];
    view_matrix[2] = Rx[2];
    view_matrix[3] = 0.0;

    view_matrix[4] = Ry[0];
    view_matrix[5] = Ry[1];
    view_matrix[6] = Ry[2];
    view_matrix[7] = 0.0;

    view_matrix[8] = Rz[0];
    view_matrix[9] = Rz[1];
    view_matrix[10] = Rz[2];
    view_matrix[11] = 0.0;

    view_matrix[12] = view_matrix[13] = view_matrix[14] = 0.0;
    view_matrix[15] = 1.0;
    view_matrix = view_matrix * T;
}
```

Perspective

```
// [TODO] compute persepective projection matrix
void setPerspective()
{
    cur_proj_mode = Perspective;
    GLfloat f;
    f = -1 / tan(proj.fovy / 2);

    project_matrix[0] = f / proj.aspect;
    project_matrix[1] = project_matrix[2] = project_matrix[3] = 0;

    project_matrix[5] = f;
    project_matrix[4] = project_matrix[6] = project_matrix[7] = 0;

    project_matrix[10] = (proj.farClip + proj.nearClip) / (proj.nearClip - proj.farClip);
    project_matrix[11] = 2 * proj.farClip * proj.nearClip / (proj.nearClip - proj.farClip);
    project_matrix[8] = project_matrix[9] = 0;

    project_matrix[14] = -1;
    project_matrix[12] = project_matrix[13] = project_matrix[15] = 0;
}
```

Change Size

```
// Call back function for window reshape
void ChangeSize(GLFWwindow* window, int width, int height)
{
    glViewport(0, 0, width, height);
    // [TODO] change your aspect ratio in both perspective and orthogonal view
    proj.aspect = (float)width / height;
    if ((float)width / height > 1) {
        proj.left = -(float)width / height;
        proj.right = (float)width / height;
        proj.top = 1.0;
        proj.bottom = -1.0;
    }
    else {
        proj.left = -1;
        proj.right = 1;
        proj.top = (float)height / width;
        proj.bottom = -(float)height / width;
    }

    if (cur_proj_mode == Perspective) {
        setPerspective();
    }
    else if (cur_proj_mode == Orthogonal) {
        setOrthogonal();
    }
    setViewingMatrix();
}
```

Draw Plane

新增一個 Shape plane 處理

```
// [TODO] draw the plane with above vertices and color
Matrix4 MVP;
GLfloat mvp[16];
// [TODO] multiply all the matrix

// [TODO] row-major ---> column-major
// 固定平面
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);
//please refer to LoadModels function
//glGenVertexArrays..., glBindVertexArray...
//glGenBuffers..., glBindBuffer..., glBufferData...
glGenVertexArrays(1, &plane.vao);
glBindVertexArray(plane.vao);

glGenBuffers(1, &plane.vbo);
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(GLfloat), (void*)0);

glGenBuffers(1, &plane.p_color);
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(GLfloat), (void*)0);
```

Render Scene 使用 isDrawWireFrame 的布林值來處理 w 的事件

```
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);
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];

// use uniform to send mvp to vertex shader
// [TODO] draw 3D model in solid or in wireframe mode here, and draw plane
glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);
glBindVertexArray(m_shape_list[cur_idx].vao);
// 不讓plane也wireframe
if (isDrawWireframe) {
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    glDrawArrays(GL_TRIANGLES, 0, m_shape_list[cur_idx].vertex_count);
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
}
else {
    glDrawArrays(GL_TRIANGLES, 0, m_shape_list[cur_idx].vertex_count);
}
glBindVertexArray(0);
drawPlane();
```

新增一個 function 來印出資訊

```
void info() {
    /*Translation Matrix, Rotation Matrix, Scaling Matrix, Viewing Matrix, Projection Matrix*/
    vector<string> names{ "Translation Matrix","Rotation Matrix","Scaling Matrix","Viewing Matrix","Projection Matrix" };
    cout << names[0] << ":" << models[cur_idx].position << endl;
    cout << names[1] << ":" << models[cur_idx].rotation << endl;
    cout << names[2] << ":" << models[cur_idx].scale << endl;
    cout << names[3] << ":" << view_matrix << endl;
    cout << names[4] << ":" << project_matrix << endl;
}
```

Scroll callback

```
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
{
    // [TODO] scroll up positive, otherwise it would be negtive
    switch (cur_trans_mode) {
    case GeoTranslation:
        models[cur_idx].position += Vector3(0, 0, 0.1*yoffset);
        break;
    case GeoScaling:
        models[cur_idx].scale += Vector3(0, 0, 0.1* yoffset);
        break;
    case GeoRotation:
        models[cur_idx].rotation += Vector3(0, 0, 0.1 * yoffset);
        break;
    case ViewEye:
        main_camera.position -= Vector3(0, 0, 0.1 * yoffset);
        setViewingMatrix();
        break;
    case ViewCenter:
        main_camera.center -= Vector3(0, 0, 0.01 * yoffset);
        setViewingMatrix();
        break;
    case ViewUp:
        main_camera.up_vector -= Vector3(0, 0, 0.01 * yoffset);
        setViewingMatrix();
        break;
    }
}
```

Mouse

```
void mouse_button_callback(GLFWwindow* window, int button, int action, int mods)
{
    // [TODO] Call back function for mouse
    mouse_pressed = !mouse_pressed;
}
```

使用 0.01、0.001 來控制上下拉靈敏度

```
if (mouse_pressed == true) {
    switch (cur_trans_mode) {
    case GeoTranslation:
        models[cur_idx].position += Vector3(0.01 * (xpos - starting_press_x), 0.01 * (ypos - starting_press_y), 0);
        break;
    case GeoScaling:
        models[cur_idx].scale += Vector3(0.01 * (xpos - starting_press_x), 0.01 * (ypos - starting_press_y), 0);
        break;
    case GeoRotation:
        models[cur_idx].rotation += Vector3(0.01 * (ypos - starting_press_y), 0.01 * (xpos - starting_press_x), 0);
        break;
    case ViewEye:
        main_camera.position += Vector3(0.01 * (xpos - starting_press_x), 0.01 * (ypos - starting_press_y), 0);
        setViewingMatrix();
        break;
    case ViewCenter:
        main_camera.center += Vector3(-0.001 * (xpos - starting_press_x), 0.001 * (ypos - starting_press_y), 0);
        setViewingMatrix();
        break;
    case ViewUp:
        main_camera.up_vector += Vector3(-0.01 * (xpos - starting_press_x), 0.01 * (ypos - starting_press_y), 0);
        setViewingMatrix();
        break;
    }
```

Model size 新增 int model_cnt 來存 Model 數量

```
// [TODO#] Load five model at here
model_cnt = model_list.size();
for (int i = 0; i < model_cnt; i++) {
    LoadModels(model_list[i]);
    cout << i << "," << model_list[i] << endl;
}
```

另外新增了按 Esc 可以關閉

發現到:
要用到 Shader.vs、Matrix 要先乘 MVP 才會出現平面
鍵盤事件需要 action == 1 來讓它不會跳(Ex: w: 不會反覆切換 wireframe 跟 solid mode)