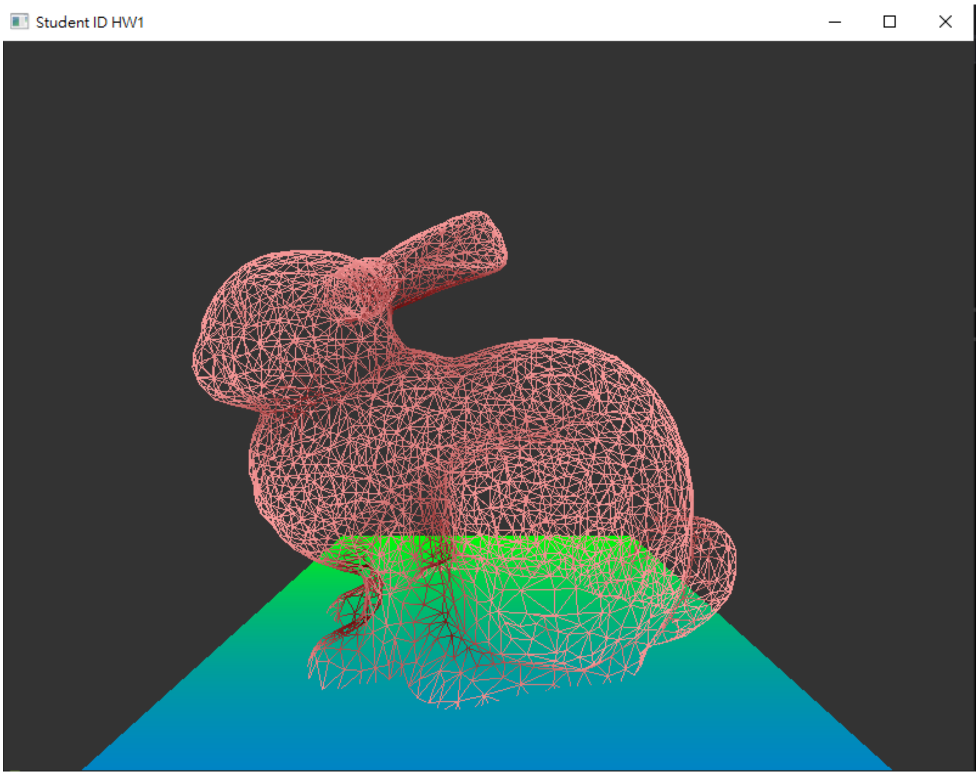
Computer Graphics

功能 Demo

W: switch between solid and wireframe mode



當按下 key W時，就去set `wireframe_mode` 的值：
這邊使用一個 global 的變數 `wireframe_mode` ，初始值為 false，並在 `RenderScene` 時，去 `check_model_mode()` 檢查現在狀況。

In KeyCallback ：

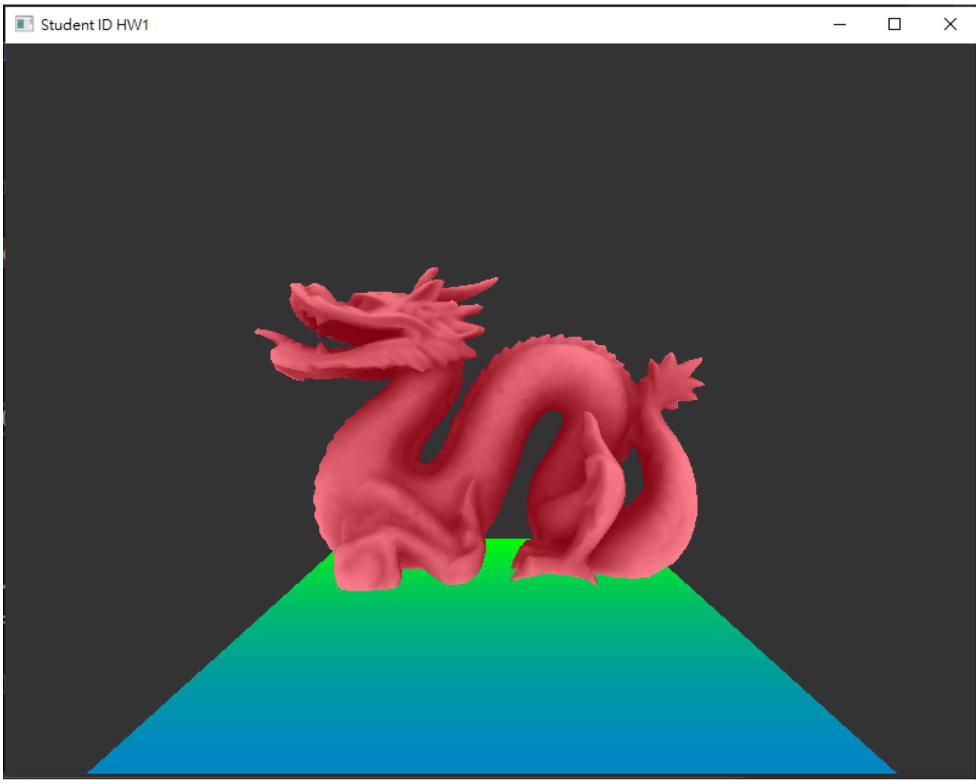
```
case GLFW_KEY_W:
    wireframe_mode = !wireframe_mode;
    break;
```

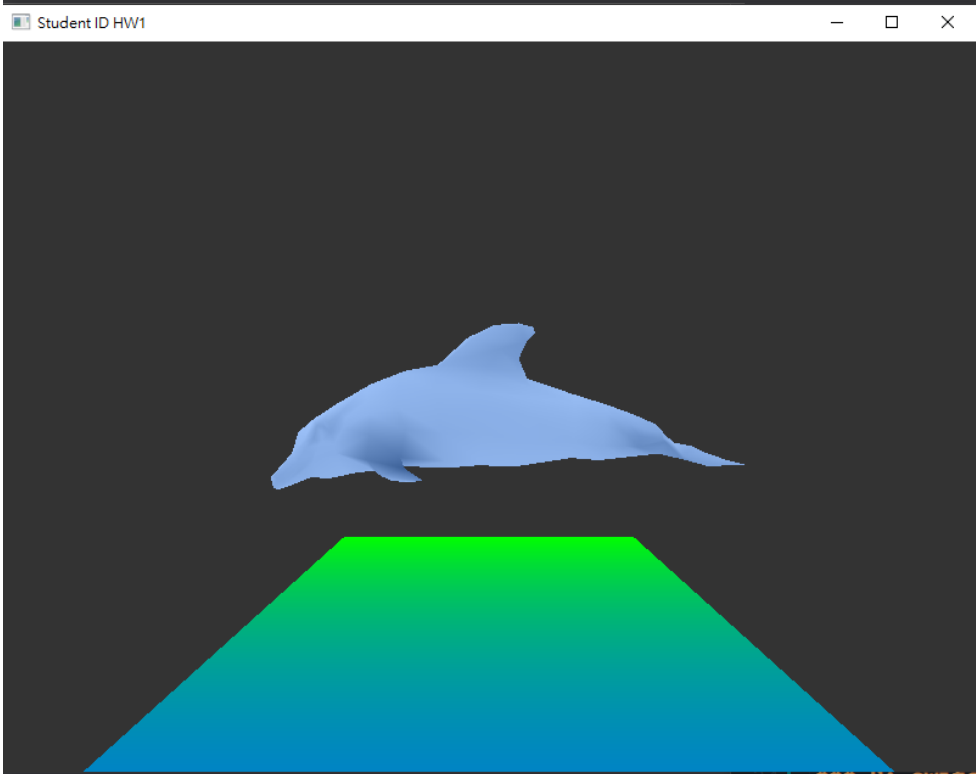
check_model_mode() ：

```
void check_model_mode() {
        if (wireframe_mode == false)
                glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
        else
                glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
}
```

並且此判斷需要加在畫Plane之前，並且在畫Plane時需要設定為 `GL_FILL` ，不然Plane也會成為wireframe。

Z/X: switch the model

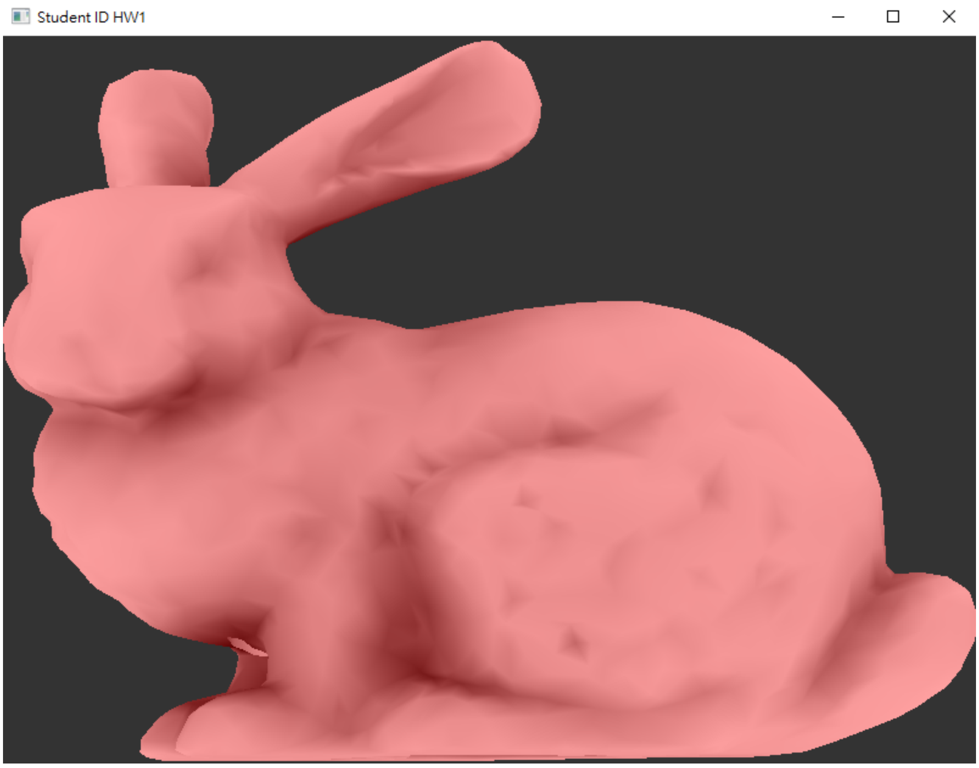




```
// Total size of model
const int MODEL_NUM = 5;

// Previous model
case GLFW_KEY_Z:
    cur_idx--;
    if (cur_idx < 0)
        cur_idx = MODEL_NUM - 1;
    break;
// Next model
case GLFW_KEY_X:
    cur_idx++;
    cur_idx %= MODEL_NUM;
    break;
```

Z: 設定為前一個model，並且檢查idx range。
X: 設定為下一個model，並且檢查idx range (不可超過總共laod進來的model)。

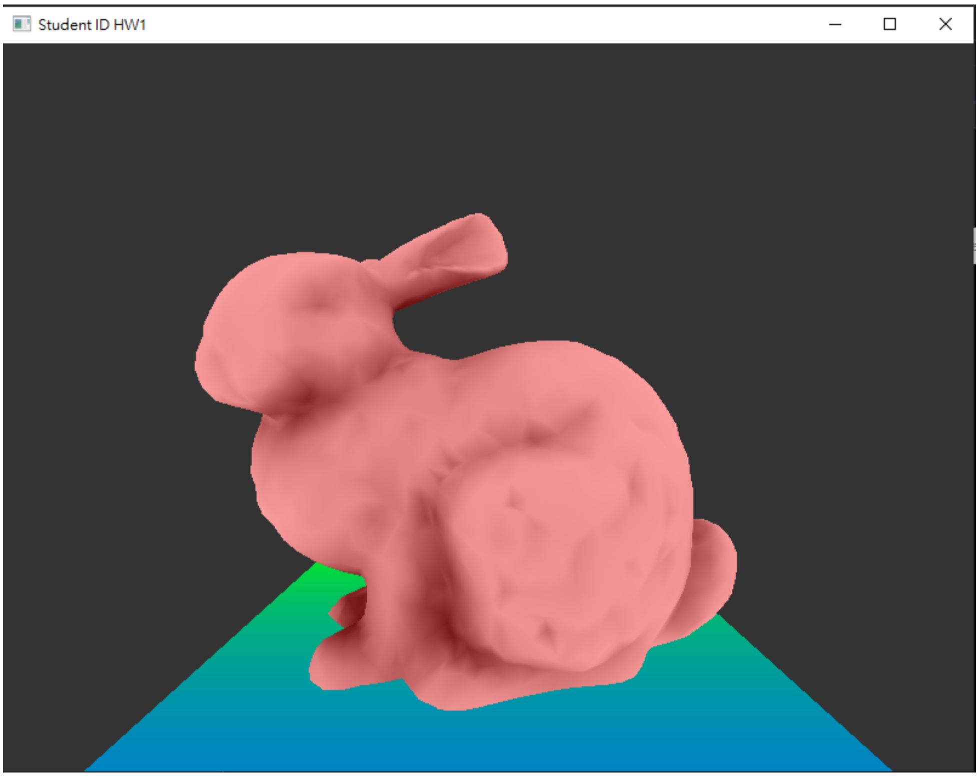
O: switch to Orthogonal projection



根據上課所提供的公式，我們可以得出 project_matrix 為以下的值：

```
void setOrthogonal()
{
    cur_proj_mode = Orthogonal;
    GLfloat x_width = proj.right - proj.left;
    GLfloat y_width = proj.top - proj.bottom;
    GLfloat z_width = proj.farClip - proj.nearClip;
    project_matrix.set(2 / x_width, 0, 0, -(proj.right + proj.left) / x_width,
                       0, 2 / y_width , 0, -(proj.top + proj.bottom) / y_width,
                       0, 0, -2 / z_width, -(proj.farClip + proj.nearClip) / z_widt
                       0, 0, 0, 1);
}
```

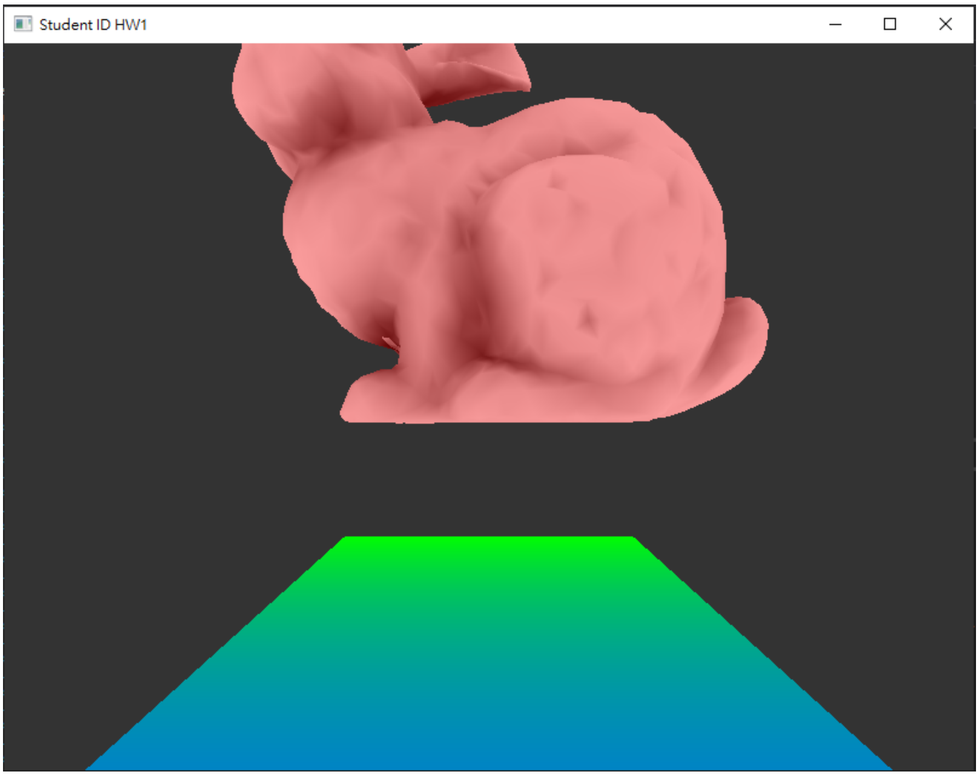
P: switch to NDC Perspective projection



根據上課所提供的公式，我們可以得出 project_matrix 為以下的值：

```
// [TODO] compute persepective projection matrix
void setPerspective()
{
    cur_proj_mode = Perspective;
    GLfloat F = tan(proj.fovy * PI / 180.0 / 2.0);
    GLfloat f = 1.0 / F;

    GLfloat z_width = proj.nearClip - proj.farClip;
    project_matrix.set(f / proj.aspect, 0, 0, 0,
                       0, f, 0, 0,
                   0, 0, (proj.farClip + proj.nearClip) / z_width,
                2.0 * proj.nearClip * proj.farClip / z_width,
               0, 0, -1, 0);

}
```

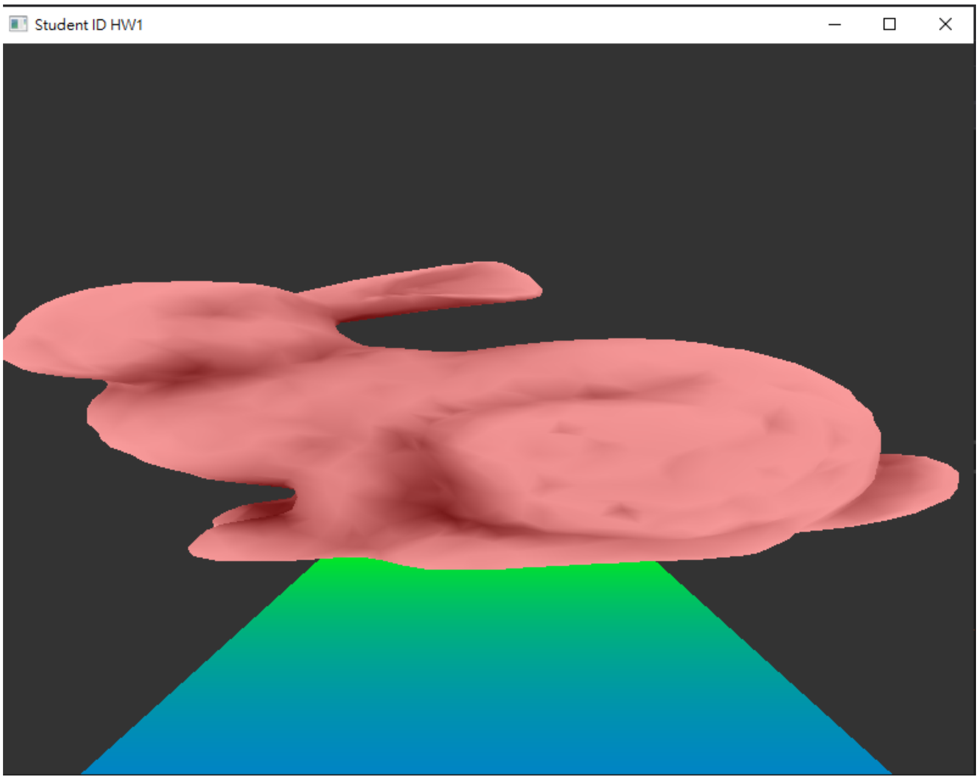
T: switch to translation mode



根據 `translate` 去寫對應的公式：

```
// [TODO] given a translation vector then output a Matrix4 (Translation Matrix)
Matrix4 translate(Vector3 vec)
{
        Matrix4 mat;

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

        return mat;
}
```

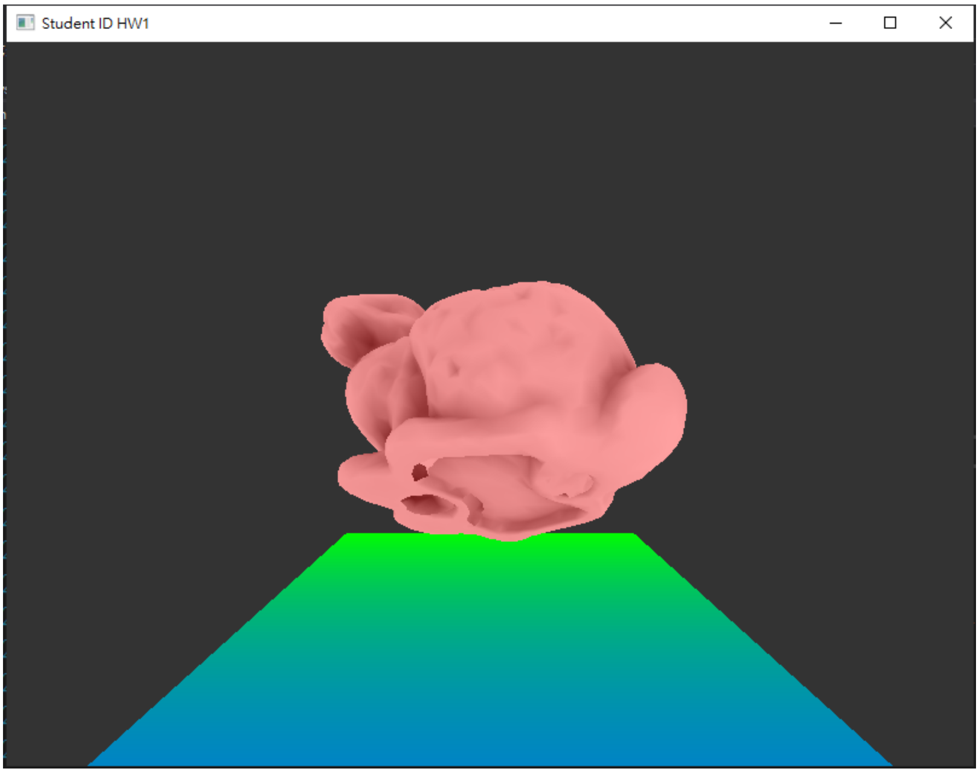
S: switch to scale mode



根據 scaling 去寫對應的公式：

```
// [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;
}
```

R: switch to rotation mode



根據 `rotate` 去寫對應的公式：

```
// [TODO] given a float value then ouput a rotation matrix alone axis-X (rotate alone a
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 a
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 a
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;
}

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

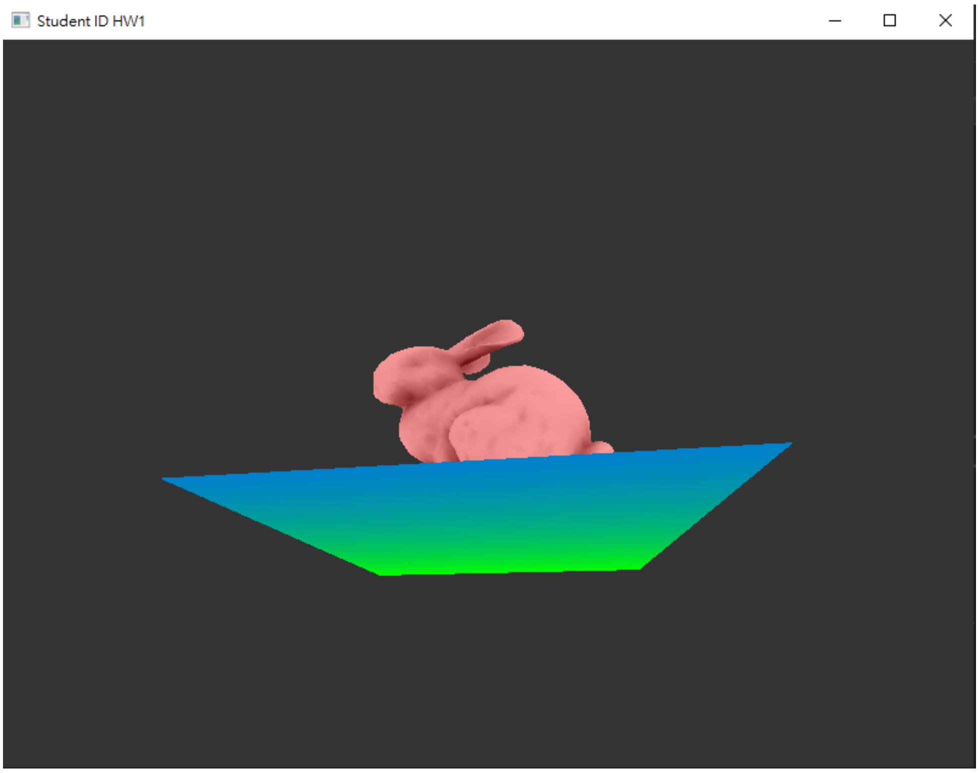
Viewing Matrix

在介紹攝影機模式之前，需要先介紹 `viewing_matrix` ，裏頭會去計算攝影機的位置等。

In `Viewing Matrix` ：

```
// [TODO] compute viewing matrix accroding to the setting of main_camera
void setViewingMatrix()
{
	// view_matrix[...] = ...
	Vector3 dir = main_camera.center - main_camera.position;
	Vector3 camera_z = dir.normalize();
	Vector3 u_norm = main_camera.up_vector.normalize();
	Vector3 camera_x = camera_z.cross(u_norm);
	Vector3 camera_y = camera_x.cross(dir);
	view_matrix = Matrix4(camera_x[0], camera_x[1], camera_x[2], 0,
		camera_y[0], camera_y[1], camera_y[2], 0,
		-camera_z[0], -camera_z[1], -camera_z[2], 0,
		0, 0, 0, 1) *
		Matrix4(1, 0, 0, -main_camera.position[0],
			0, 1, 0, -main_camera.position[1],
			0, 0, 1, -main_camera.position[2],
			0, 0, 0, 1);
}
```

E: switch to translate eye position mode



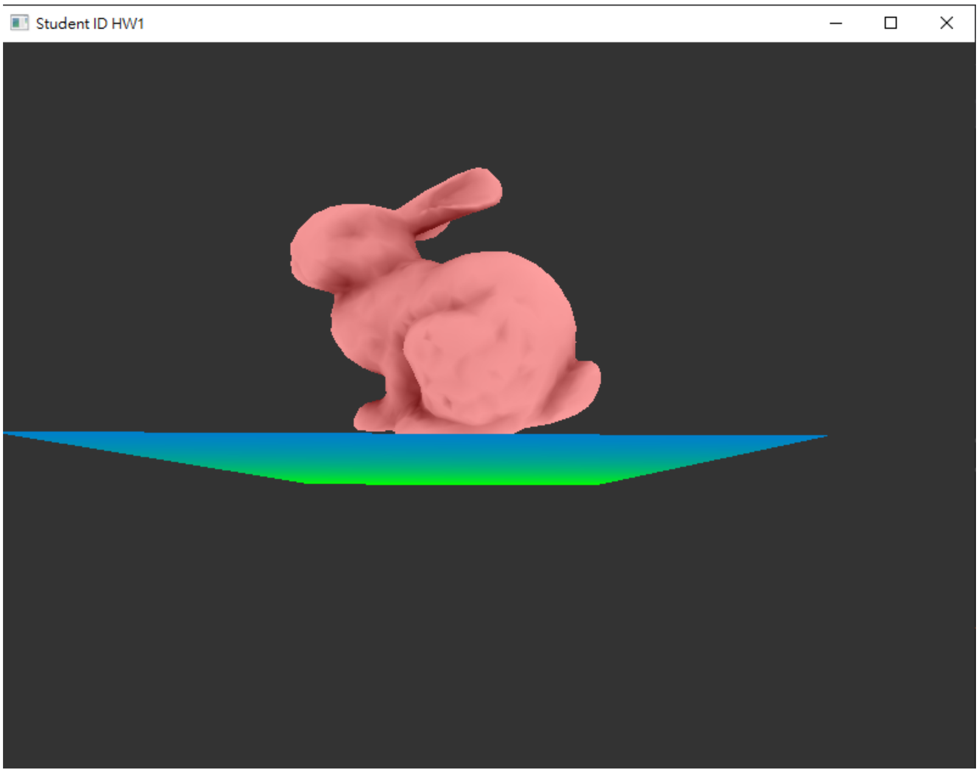
以 `cursor_pos_callback` 為例子，這邊需要去設定一些參數：

`starting_press_x` 為當下按下去的點座標，並且去計算變動量值 `diff` ，這邊需要去設置滑鼠敏感度 `diff_range = 0.01` ，取決於變動量需要多敏感。

In `cursor_pos_callback`：

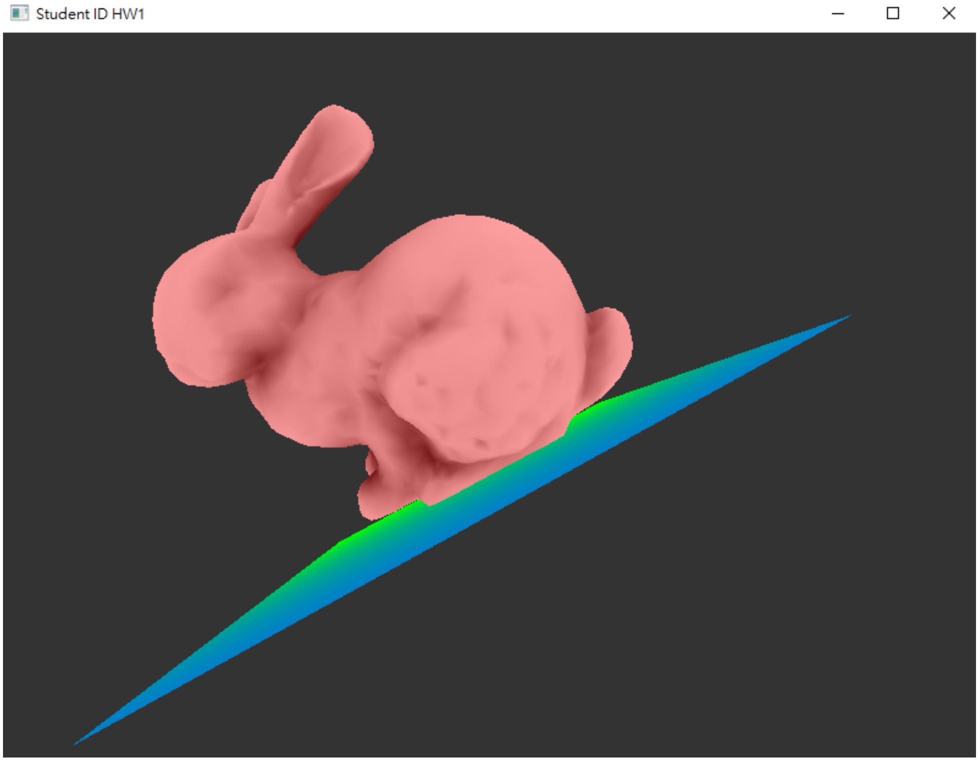
```
if (!mouse_pressed) {
		starting_press_x = xpos;
		starting_press_y = ypos;
		return;
	}

double x_diff, y_diff, diff_range = 0.01;

x_diff = xpos - starting_press_x;
y_diff = ypos - starting_press_y;

starting_press_x = xpos;
starting_press_y = ypos;

// 省略部分的switch and case
case ViewEye:
    main_camera.position.x += x_diff * diff_range;
    main_camera.position.y += y_diff * diff_range;
    setViewingMatrix();
    break;
```

C: switch to translate viewing center position mode



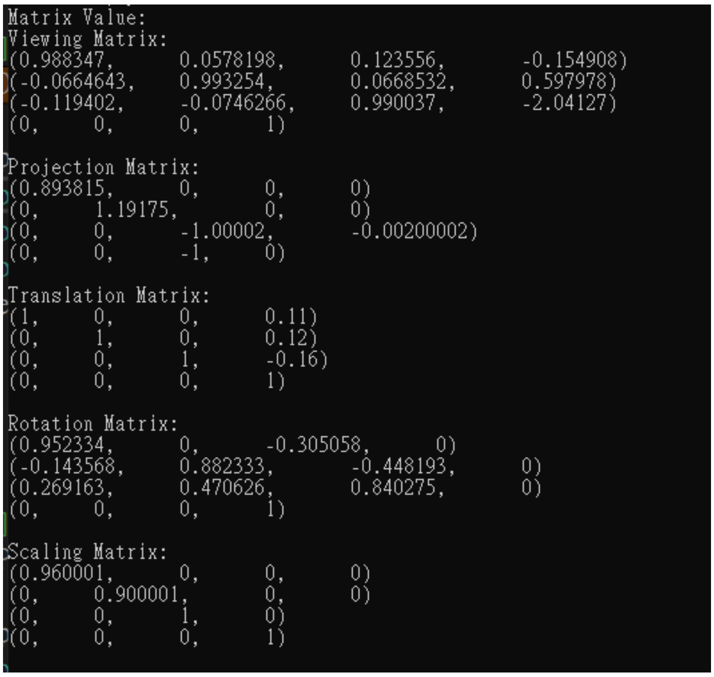
```
case ViewCenter:
    main_camera.center.x += x_diff * diff_range;
    main_camera.center.y += y_diff * diff_range;
    setViewingMatrix();
    break;
```

U: switch to translate camera up vector position mode



```
case ViewUp:
    main_camera.up_vector.x += x_diff * diff_range;
    main_camera.up_vector.y += y_diff * diff_range;
    setViewingMatrix();
    break;
```

I: print information



```
Matrix Value:
Viewing Matrix:
(0.988347,      0.0578198,      0.123556,       -0.154908)
(-0.0664643,    0.993254,       0.0668532,      0.597978)
(-0.119402,     -0.0746266,     0.990037,       -2.04127)
(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,      0.11)
(0,     1,      0,      0.12)
(0,     0,      1,      -0.16)
(0,     0,      0,      1)

Rotation Matrix:
(0.952334,      0,      -0.305058,      0)
(-0.143568,     0.882333,       -0.448193,      0)
(0.269163,      0.470626,       0.840275,       0)
(0,     0,      0,      1)

Scaling Matrix:
(0.960001,      0,      0,      0)
(0,     0.900001,       0,      0)
(0,     0,      1,      0)
(0,     0,      0,      1)
```

```
void print_Matrix(string matrix_name, Matrix4 matrix) {
        cout << matrix_name << ":" << endl;
        cout << matrix << endl;
}

case GLFW_KEY_I:
    cout << "Matrix Value:" << endl;
    print_Matrix("Viewing Matrix", view_matrix);
    print_Matrix("Projection Matrix", project_matrix);
    print_Matrix("Translation Matrix", translate(models[cur_idx].position));
    print_Matrix("Rotation Matrix", rotate(models[cur_idx].rotation));
    print_Matrix("Scaling Matrix", scaling(models[cur_idx].scale));
    break;
```

Callback function

根據螢幕大小調整 `ChangeSize` ：

這邊只有寫到當 `cur_proj_mode` 為 `Perspective` ，需要特別注意，因為這邊的 aspect 會改變，而 `Orthogonal` 時跟 aspect 沒有關係，所以不用再去 set 。

```
// Call back function for window reshape
void ChangeSize(GLFWwindow* window, int width, int height)
{
    glViewport(0, 0, width, height);
    // [TODO] change your aspect ratio
    proj.aspect = (float) width / (float) height;
    //cout << "proj.aspect:" << proj.aspect << endl;
    if (cur_proj_mode == Perspective) {
        setPerspective();
    }
}
```

按鍵 `KeyCallback` 完整程式碼:

第一步先去檢查 `action` ，因為還有action總共有這些值 `GLFW_PRESS` , `GLFW_REPEAT` or `GLFW_RELEASE` ，而如果沒有進行檢查的話，會一直被call，導致變動太快。
所以我們這邊只在當按鈕被按下 `GLFW_PRESS` 時，才去做對應的事件，如果其他是其他 action 就直接return。

```
void change_model_mode() {
    if (wireframe_mode == false)
        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    else
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    wireframe_mode = !wireframe_mode;
}

void print_Matrix(string matrix_name, Matrix4 matrix) {
    cout << matrix_name << ":" << endl;
    cout << matrix << endl;
}

void KeyCallback(GLFWwindow* window, int key, int scancode, int action, int mods)
{
    // [TODO] Call back function for keyboard
    // Filter out other action.
    // Without doing so, the cur_idx will change fastly.
    if (action != GLFW_PRESS)
        return;

    switch (key) {
    // Switch between solid and wireframe mode
    case GLFW_KEY_W:
        change_model_mode();
        break;
    // Next model
    case GLFW_KEY_Z:
        cur_idx++;
        cur_idx %= MODEL_NUM;
        break;
    // Previous model
    case GLFW_KEY_X:
        cur_idx--;
        if (cur_idx < 0)
            cur_idx = MODEL_NUM - 1;
        break;
    // Switch to Orthogonal projection
    case GLFW_KEY_O:
        setOrthogonal();
        break;
    // Switch to NDC Perspective projection
    case GLFW_KEY_P:
        setPerspective();
        break;
    case GLFW_KEY_T:
        cur_trans_mode = GeoTranslation;
        break;
    case GLFW_KEY_S:
        cur_trans_mode = GeoScaling;
        break;
    case GLFW_KEY_R:
        cur_trans_mode = GeoRotation;
        break;
    case GLFW_KEY_E:
        cur_trans_mode = ViewEye;
        break;
    case GLFW_KEY_C:
        cur_trans_mode = ViewCenter;
        break;
    case GLFW_KEY_U:
        cur_trans_mode = ViewUp;
        break;
    case GLFW_KEY_I:
        cout << "Matrix Value:" << endl;
        print_Matrix("Viewing Matrix", view_matrix);
        print_Matrix("Projection Matrix", project_matrix);
        print_Matrix("Translation Matrix", translate(models[cur_idx].position));
        print_Matrix("Rotation Matrix", rotate(models[cur_idx].rotation));
        print_Matrix("Scaling Matrix", scaling(models[cur_idx].scale));
        break;
    }
}
```

當滑鼠滾輪滾動時 `scroll_callback` ：

這邊需要去設置一個 `diff_range` ，因為如果沒設這個 range，這樣會導致變化量太大。
那當 `mode` 為跟 `viewing_matrix` 有相關時，這邊需要 call `setViewingMatrix` ，並且重新運算 `viewing_matrix` 的值。

```
switch (cur_trans_mode) {
case GeoTranslation:
    models[cur_idx].position.z += yoffset * diff_range;
    //cout << "model x:" << models[cur_idx].position.z  << endl;
    break;
case GeoRotation:
    models[cur_idx].rotation.z += yoffset * diff_range;
    break;
case GeoScaling:
    models[cur_idx].scale.z += yoffset * diff_range;
    break;
case ViewCenter:
    main_camera.center.z += yoffset * diff_range;
    setViewingMatrix();
    break;
case ViewEye:
    main_camera.position.z -= yoffset * diff_range;
    setViewingMatrix();
    break;
case ViewUp:
    main_camera.up_vector.z += yoffset * diff_range;
    setViewingMatrix();
    break;
}
```

當滑鼠按下時 `mouse_button_callback` ：

當左邊的滑鼠按下時，就去設定 `mouse_pressed` 的值。

```
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;
    }
    else
        mouse_pressed = false;
}
```

當滑鼠移動時 `cursor_pos_callback` ：

需要先去檢查，當按鈕還沒按下時，開始位置的x, y為鼠標移動到的地方，如果只是當 moused_pressed時才去改變，會導致位置瞬間移動。
這邊一樣需要去設定 `diff_range` ，讓物體變動量不要太大。

```
static void cursor_pos_callback(GLFWwindow* window, double xpos, double ypos)
{
    // [TODO] cursor position callback function
    if (!mouse_pressed) {
        starting_press_x = xpos;
        starting_press_y = ypos;
        return;
    }

    double x_diff, y_diff, diff_range = 0.01;

    x_diff = xpos - starting_press_x;
    y_diff = ypos - starting_press_y;

    starting_press_x = xpos;
    starting_press_y = ypos;

    switch (cur_trans_mode) {
    case GeoTranslation:
        models[cur_idx].position.x += x_diff * diff_range;
        models[cur_idx].position.y -= y_diff * diff_range;
        break;
    case GeoRotation:
        // drag vertically -> apply in x axis
        models[cur_idx].rotation.x -= PI * 60 / 180.0 * y_diff * diff_range;
        // drag horizontally -> apply in y axis
        models[cur_idx].rotation.y -= PI * 60 / 180.0 * x_diff * diff_range;
        break;
    case GeoScaling:
        models[cur_idx].scale.x -= x_diff * diff_range;
        models[cur_idx].scale.y -= y_diff * diff_range;
        break;
    case ViewCenter:
        main_camera.center.x -= x_diff * diff_range;
        main_camera.center.y -= y_diff * diff_range;
        setViewingMatrix();
        break;
    case ViewEye:
        main_camera.position.x -= x_diff * diff_range;
        main_camera.position.y += y_diff * diff_range;
        setViewingMatrix();
        break;
    case ViewUp:
        main_camera.up_vector.x -= x_diff * diff_range;
        main_camera.up_vector.y -= y_diff * diff_range;
        setViewingMatrix();
        break;
    }
}
```

Other thing: Set Up

這邊需要做的只是，將每個 model 都給 load 進來。

```
void setupRC()
{
    // setup shaders
    setShaders();
    initParameter();

    // OpenGL States and Values
    glClearColor(0.2, 0.2, 0.2, 1.0);
    vector<string> model_list{ "../ColorModels/bunny5KC.obj", "../ColorModels/dragon10KC
    // [TODO] Load five model at here
    for (string cur_item : model_list) {
        LoadModels(cur_item);
    }
}
```

In `shader.vs` :

這邊需要接收傳過來的 `mvp` 值，並且乘上座標。

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

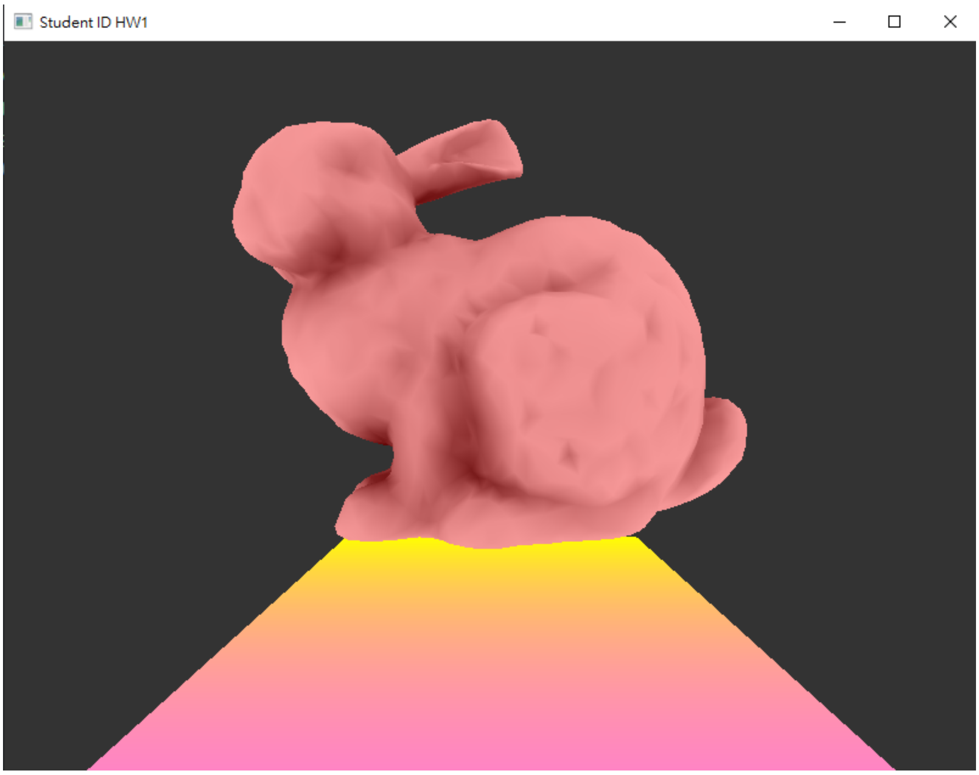
重畫場景 RenderScene

這邊的 `MVP` 是 `projection_matrix`, `view_matrix`, `Transpose`, `Rotation`, `Scaling` 互相相乘積，並且給 `mvp` 值的時候需要 *row-majoin -> column-major*。

```
// Render function for display rendering
void RenderScene(void) {
    // clear canvas
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);

    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);

    // [TODO] multiply all the matrix
    Matrix4 MVP = project_matrix * view_matrix * T * R * S;
    GLfloat mvp[16];

    // [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];

    // draw the model first becuz the Plane should not be wireframe!!
    check_model_mode();

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

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

其他功能：

嘗試改變地板顏色：


Student ID HW1