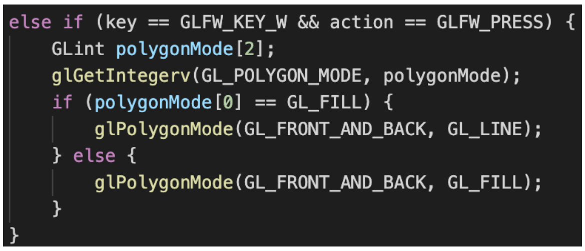
Computer Graphics Assignment #1

I. Implementation of key mapping

- W: switch between solid and wireframe mode

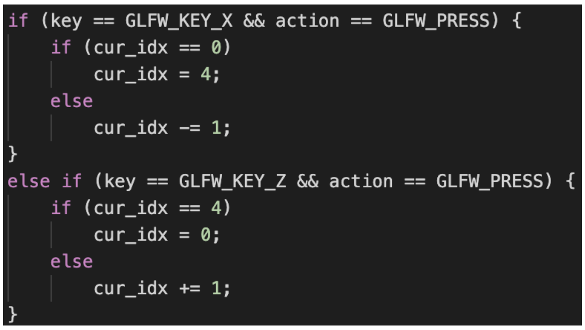
  Use glGetIntegerv to check out polygonMode is GL_FILL or GL_LINE.
  If the mode is GL_FILL than translate into GL_LINE, otherwise translate into GL_FILL.



```
else if (key == GLFW_KEY_W && action == GLFW_PRESS) {
    GLint polygonMode[2];
    glGetIntegerv(GL_POLYGON_MODE, polygonMode);
    if (polygonMode[0] == GL_FILL) {
        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    } else {
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
    }
}
```

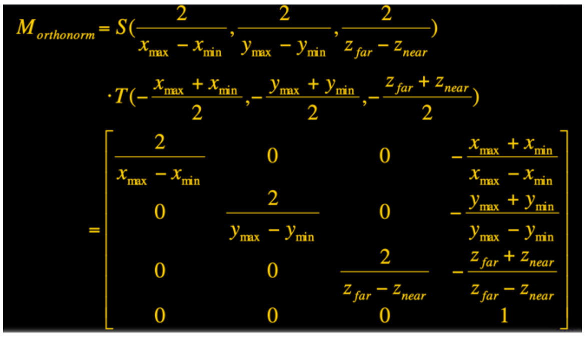
- Z/X: switch the model

  Change cur_idx to change the current model. (cur_idx = 0 ~ 4)



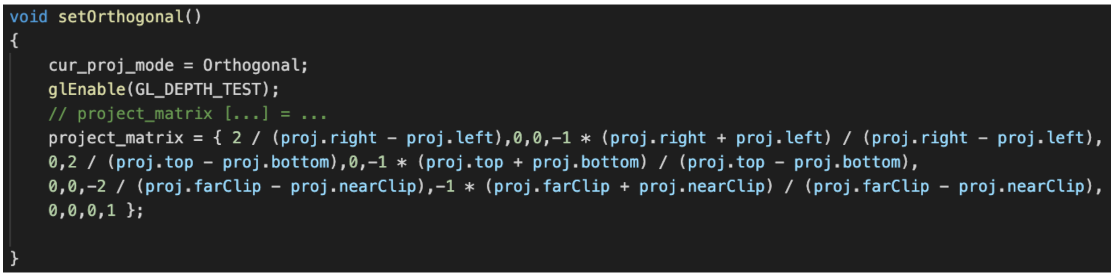
```
if (key == GLFW_KEY_X && action == GLFW_PRESS) {
    if (cur_idx == 0)
        cur_idx = 4;
    else
        cur_idx -= 1;
}
else if (key == GLFW_KEY_Z && action == GLFW_PRESS) {
    if (cur_idx == 4)
        cur_idx = 0;
    else
        cur_idx += 1;
}
```

- O: switch to Orthogonal projection



$$M_{orthonorm} = S\left(\frac{2}{x_{\max} - x_{\min}}, \frac{2}{y_{\max} - y_{\min}}, \frac{2}{z_{far} - z_{near}}\right) \cdot T\left(-\frac{x_{\max} + x_{\min}}{2}, -\frac{y_{\max} + y_{\min}}{2}, -\frac{z_{far} + z_{near}}{2}\right)$$

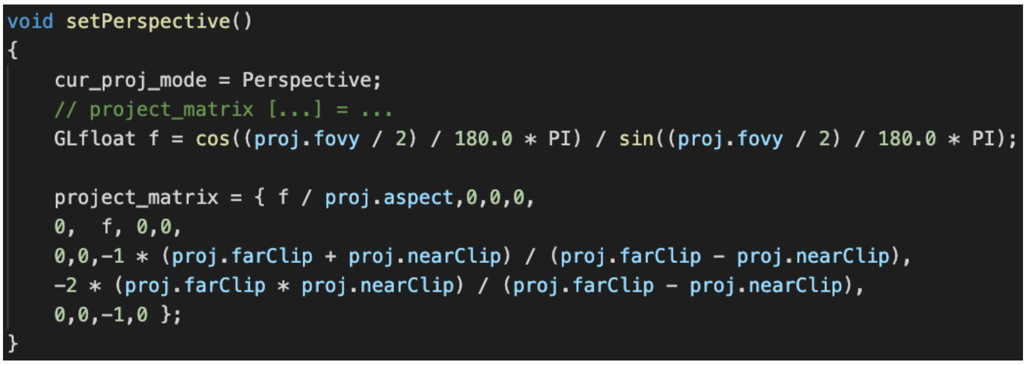
$$= \begin{bmatrix} \frac{2}{x_{\max} - x_{\min}} & 0 & 0 & -\frac{x_{\max} + x_{\min}}{x_{\max} - x_{\min}} \\ 0 & \frac{2}{y_{\max} - y_{\min}} & 0 & -\frac{y_{\max} + y_{\min}}{y_{\max} - y_{\min}} \\ 0 & 0 & \frac{2}{z_{far} - z_{near}} & -\frac{z_{far} + z_{near}}{z_{far} - z_{near}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



```
void setOrthogonal()
{
    cur_proj_mode = Orthogonal;
    glEnable(GL_DEPTH_TEST);
    // project_matrix [...] = ...
    project_matrix = { 2 / (proj.right - proj.left),0,0,-1 * (proj.right + proj.left) / (proj.right - proj.left),
    0,2 / (proj.top - proj.bottom),0,-1 * (proj.top + proj.bottom) / (proj.top - proj.bottom),
    0,0,-2 / (proj.farClip - proj.nearClip),-1 * (proj.farClip + proj.nearClip) / (proj.farClip - proj.nearClip),
    0,0,0,1 };

}
```

- P: switch to NDC Perspective projection proj.fovy: stands for the “field of view y-axis” and is the vertical angle of the camera's lens

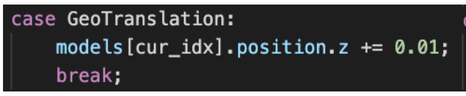
$$M = \begin{bmatrix} \frac{1}{ar*tan(\frac{\alpha}{2})} & 0 & 0 & 0 \\ 0 & \frac{1}{tan(\frac{\alpha}{2})} & 0 & 0 \\ 0 & 0 & \frac{NearZ+FarZ}{NearZ-FarZ} & \frac{2*FarZ*NearZ}{NearZ-FarZ} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$



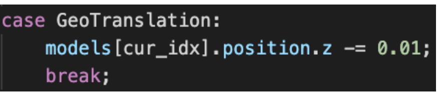
```
void setPerspective()
{
    cur_proj_mode = Perspective;
    // project_matrix [...] = ...
    GLfloat f = cos((proj.fovy / 2) / 180.0 * PI) / sin((proj.fovy / 2) / 180.0 * PI);

    project_matrix = { f / proj.aspect,0,0,0,
    0,  f, 0,0,
    0,0,-1 * (proj.farClip + proj.nearClip) / (proj.farClip - proj.nearClip),
    -2 * (proj.farClip * proj.nearClip) / (proj.farClip - proj.nearClip),
    0,0,-1,0 };
}
```

- T: switch to translation mode

  Scroll up or scroll down

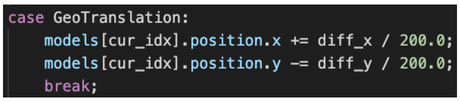


```
case GeoTranslation:
    models[cur_idx].position.z += 0.01;
    break;
```



```
case GeoTranslation:
    models[cur_idx].position.z -= 0.01;
    break;
```

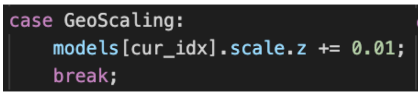
Cursor position



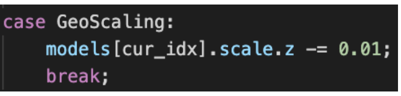
```
case GeoTranslation:
    models[cur_idx].position.x += diff_x / 200.0;
    models[cur_idx].position.y -= diff_y / 200.0;
    break;
```

- S: switch to scale mode

  Scroll up or scroll down

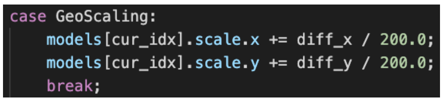


```
case GeoScaling:
    models[cur_idx].scale.z += 0.01;
    break;
```



```
case GeoScaling:
    models[cur_idx].scale.z -= 0.01;
    break;
```

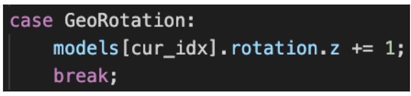
Cursor position



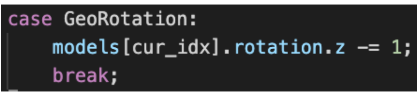
```
case GeoScaling:
    models[cur_idx].scale.x += diff_x / 200.0;
    models[cur_idx].scale.y += diff_y / 200.0;
    break;
```

- R: switch to rotation mode

  Scroll up or scroll down

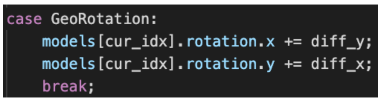


```
case GeoRotation:
    models[cur_idx].rotation.z += 1;
    break;
```



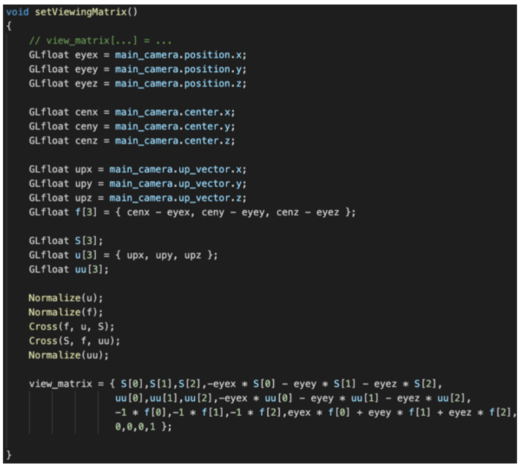
```
case GeoRotation:
    models[cur_idx].rotation.z -= 1;
    break;
```

Cursor position



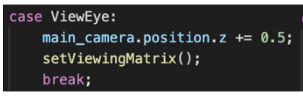
```
case GeoRotation:
    models[cur_idx].rotation.x += diff_y;
    models[cur_idx].rotation.y += diff_x;
    break;
```

- E/C/U: setViewingMatrix()

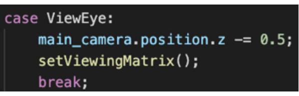


```
void setViewingMatrix()
{
    // view_matrix[...] = ...
    GLfloat eyex = main_camera.position.x;
    GLfloat eyey = main_camera.position.y;
    GLfloat eyez = main_camera.position.z;

    GLfloat cenx = main_camera.center.x;
    GLfloat ceny = main_camera.center.y;
    GLfloat cenz = main_camera.center.z;

    GLfloat upx = main_camera.up_vector.x;
    GLfloat upy = main_camera.up_vector.y;
    GLfloat upz = main_camera.up_vector.z;
    GLfloat f[3] = { cenx - eyex, ceny - eyey, cenz - eyez };

    GLfloat S[3];
    GLfloat u[3] = { upx, upy, upz };
    GLfloat uu[3];

    Normalize(u);
    Normalize(f);
    Cross(f, u, S);
    Cross(S, f, uu);
    Normalize(uu);

    view_matrix = { S[0],S[1],S[2],-eyex * S[0] - eyey * S[1] - eyez * S[2],
                    uu[0],uu[1],uu[2],-eyex * uu[0] - eyey * uu[1] - eyez * uu[2],
                    -1 * f[0],-1 * f[1],-1 * f[2],eyex * f[0] + eyey * f[1] + eyez * f[2],
                    0,0,0,1 };

}
```

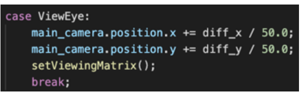
- E: switch to translate eye position mode



```
case ViewEye:
    main_camera.position.z += 0.5;
    setViewingMatrix();
    break;
```

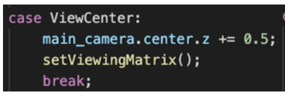


```
case ViewEye:
    main_camera.position.z -= 0.5;
    setViewingMatrix();
    break;
```

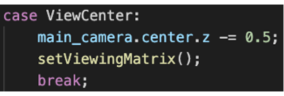


```
case ViewEye:
    main_camera.position.x += diff_x / 50.0;
    main_camera.position.y += diff_y / 50.0;
    setViewingMatrix();
    break;
```

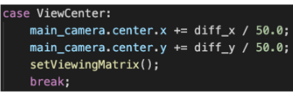
- C: switch to translate viewing center position mode



```
case ViewCenter:
    main_camera.center.z += 0.5;
    setViewingMatrix();
    break;
```

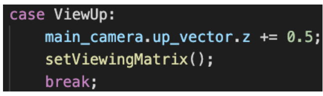


```
case ViewCenter:
    main_camera.center.z -= 0.5;
    setViewingMatrix();
    break;
```

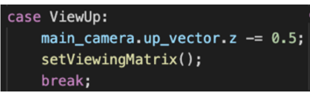


```
case ViewCenter:
    main_camera.center.x += diff_x / 50.0;
    main_camera.center.y += diff_y / 50.0;
    setViewingMatrix();
    break;
```

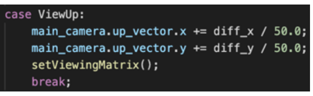
- U: switch to translate camera up vector position mode



```
case ViewUp:
    main_camera.up_vector.z += 0.5;
    setViewingMatrix();
    break;
```

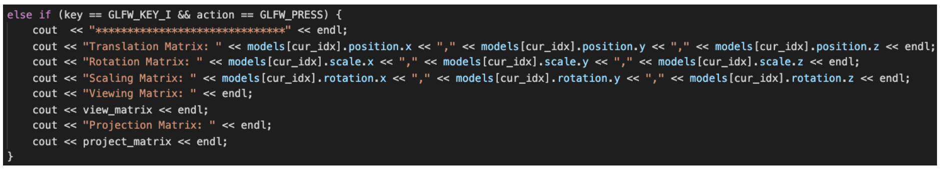


```
case ViewUp:
    main_camera.up_vector.z -= 0.5;
    setViewingMatrix();
    break;
```



```
case ViewUp:
    main_camera.up_vector.x += diff_x / 50.0;
    main_camera.up_vector.y += diff_y / 50.0;
    setViewingMatrix();
    break;
```

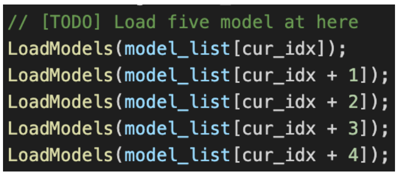
- I: print information



```
else if (key == GLFW_KEY_I && action == GLFW_PRESS) {
    cout  << "*******************************" << endl;
    cout << "Translation Matrix: " << models[cur_idx].position.x << "," << models[cur_idx].position.y << "," << models[cur_idx].position.z << endl;
    cout << "Rotation Matrix: " << models[cur_idx].scale.x << "," << models[cur_idx].scale.y << "," << models[cur_idx].scale.z << endl;
    cout << "Scaling Matrix: " << models[cur_idx].rotation.x << "," << models[cur_idx].rotation.y << "," << models[cur_idx].rotation.z << endl;
    cout << "Viewing Matrix: " << endl;
    cout << view_matrix << endl;
    cout << "Projection Matrix: " << endl;
    cout << project_matrix << endl;
}
```

II. Implementation and explanation of the pipline

- setupRC() (State initialization) → setShaders() → initParameter() → LoadModels()

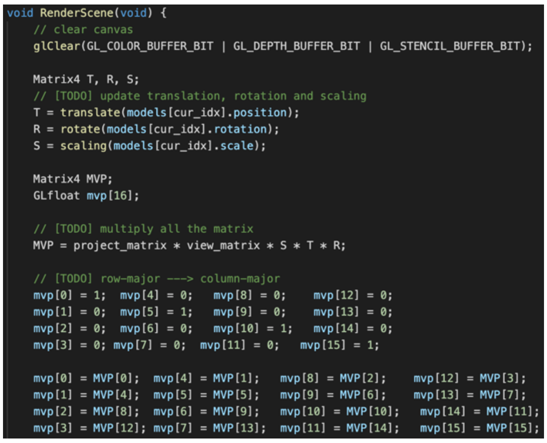


```
// [TODO] Load five model at here
LoadModels(model_list[cur_idx]);
LoadModels(model_list[cur_idx + 1]);
LoadModels(model_list[cur_idx + 2]);
LoadModels(model_list[cur_idx + 3]);
LoadModels(model_list[cur_idx + 4]);
```

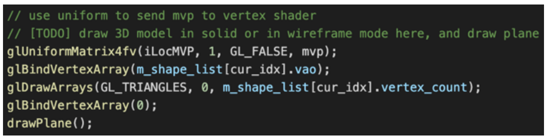
- Event handling loop: Renderscene() (Randering tasks) → glfwSwapBuffers() → glfwPollEvents() (Swap buffers and poll IO events)

  a. Renderscene()

     Update the translation, rotation and scaling matrixes, then multiply all the matrix by the reverse order.



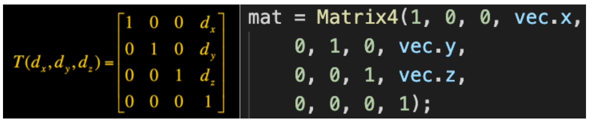
```
void RenderScene(void) {
    // clear canvas
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);

    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);

    Matrix4 MVP;
    GLfloat mvp[16];

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

    // [TODO] row-major ---> column-major
    mvp[0] = 1;  mvp[4] = 0;   mvp[8] = 0;    mvp[12] = 0;
    mvp[1] = 0;  mvp[5] = 1;   mvp[9] = 0;    mvp[13] = 0;
    mvp[2] = 0;  mvp[6] = 0;   mvp[10] = 1;   mvp[14] = 0;
    mvp[3] = 0; mvp[7] = 0;  mvp[11] = 0;   mvp[15] = 1;

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

b. Translate, rotate, scaling

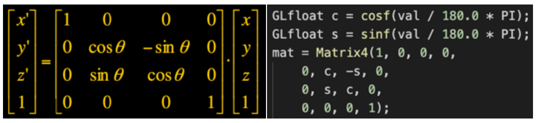
translate()



$$T(d_x,d_y,d_z)=\begin{bmatrix}1&0&0&d_x\\0&1&0&d_y\\0&0&1&d_z\\0&0&0&1\end{bmatrix}$$

```
mat = Matrix4(1, 0, 0, vec.x,
    0, 1, 0, vec.y,
    0, 0, 1, vec.z,
    0, 0, 0, 1);
```

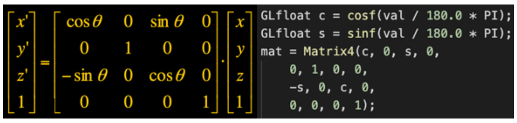
rotateX()



$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}1&0&0&0\\0&\cos\theta&-\sin\theta&0\\0&\sin\theta&\cos\theta&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

```
GLfloat c = cosf(val / 180.0 * PI);
GLfloat s = sinf(val / 180.0 * PI);
mat = Matrix4(1, 0, 0, 0,
    0, c, -s, 0,
    0, s, c, 0,
    0, 0, 0, 1);
```

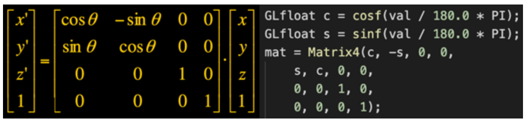
rotateY()



$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}\cos\theta&0&\sin\theta&0\\0&1&0&0\\-\sin\theta&0&\cos\theta&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

```
GLfloat c = cosf(val / 180.0 * PI);
GLfloat s = sinf(val / 180.0 * PI);
mat = Matrix4(c, 0, s, 0,
    0, 1, 0, 0,
    -s, 0, c, 0,
    0, 0, 0, 1);
```

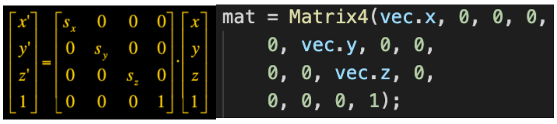
rotateZ()



$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}\cos\theta&-\sin\theta&0&0\\\sin\theta&\cos\theta&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

```
GLfloat c = cosf(val / 180.0 * PI);
GLfloat s = sinf(val / 180.0 * PI);
mat = Matrix4(c, -s, 0, 0,
    s, c, 0, 0,
    0, 0, 1, 0,
    0, 0, 0, 1);
```

scaling()



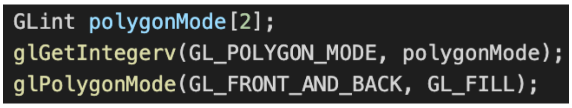
$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}s_x&0&0&0\\0&s_y&0&0\\0&0&s_z&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

```
mat = Matrix4(vec.x, 0, 0, 0,
    0, vec.y, 0, 0,
    0, 0, vec.z, 0,
    0, 0, 0, 1);
```

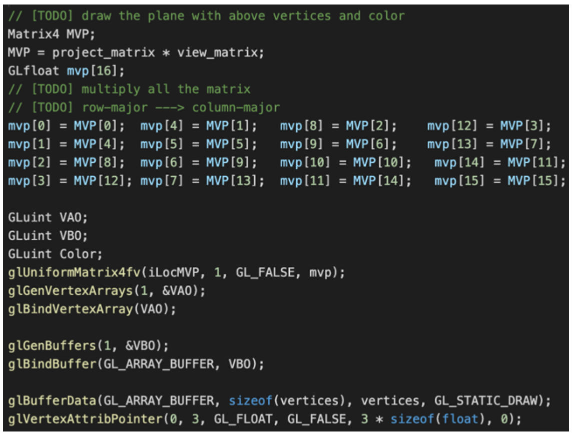
c. drawPlane()

Draw the plane with the vertices and the colors in **solid mode**.

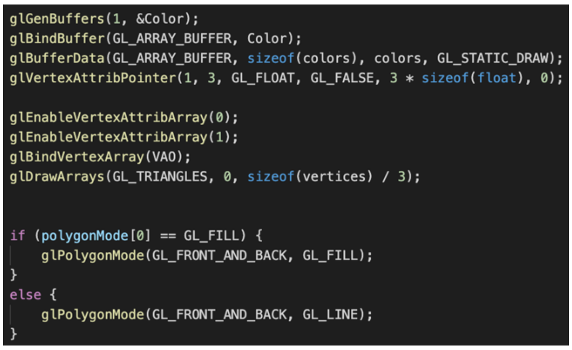
Use glGetIntegerv() to check current mode (solid or wireframe).



```
GLint polygonMode[2];
glGetIntegerv(GL_POLYGON_MODE, polygonMode);
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
```

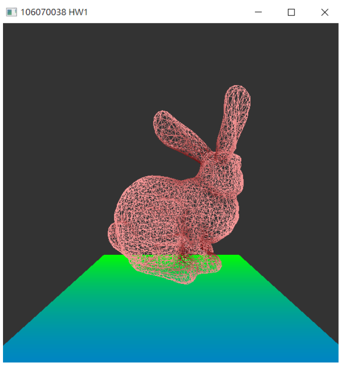


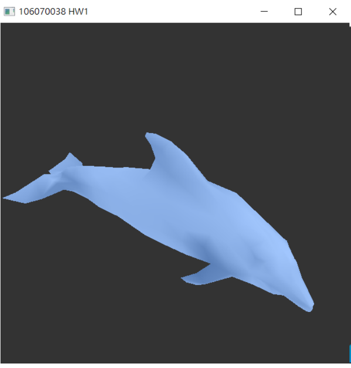
```
// [TODO] draw the plane with above vertices and color
Matrix4 MVP;
MVP = project_matrix * view_matrix;
GLfloat mvp[16];
// [TODO] multiply all the matrix
// [TODO] row-major ---> column-major
mvp[0] = MVP[0];  mvp[4] = MVP[1];   mvp[8] = MVP[2];    mvp[12] = MVP[3];
mvp[1] = MVP[4];  mvp[5] = MVP[5];   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];

GLuint VAO;
GLuint VBO;
GLuint Color;
glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);
glGenVertexArrays(1, &VAO);
glBindVertexArray(VAO);

glGenBuffers(1, &VBO);
glBindBuffer(GL_ARRAY_BUFFER, VBO);

glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), 0);
```

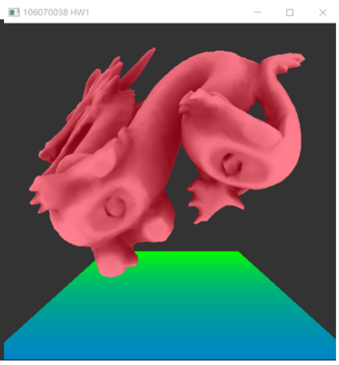


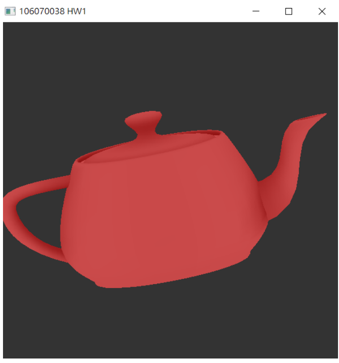
```
glGenBuffers(1, &Color);
glBindBuffer(GL_ARRAY_BUFFER, Color);
glBufferData(GL_ARRAY_BUFFER, sizeof(colors), colors, GL_STATIC_DRAW);
glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), 0);

glEnableVertexAttribArray(0);
glEnableVertexAttribArray(1);
glBindVertexArray(VAO);
glDrawArrays(GL_TRIANGLES, 0, sizeof(vertices) / 3);


if (polygonMode[0] == GL_FILL) {
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
}
else {
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
}
```

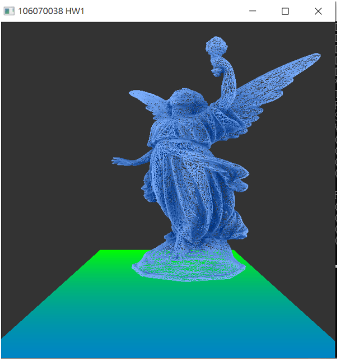
III. Some screen shot

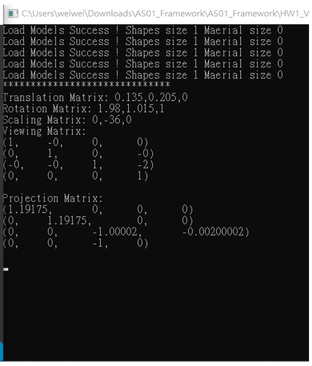











C:\Users\weiwei\Downloads\AS01_Framework\AS01_Framework\HW1_V

```
Load Models Success ! Shapes size 1 Maerial size 0
Load Models Success ! Shapes size 1 Maerial size 0
Load Models Success ! Shapes size 1 Maerial size 0
Load Models Success ! Shapes size 1 Maerial size 0
Load Models Success ! Shapes size 1 Maerial size 0
******************************
Translation Matrix: 0.135,0.205,0
Rotation Matrix: 1.98,1.015,1
Scaling Matrix: 0,-36,0
Viewing Matrix:
(1,      -0,      0,       0)
(0,      1,       0,       -0)
(-0,     -0,      1,       -2)
(0,      0,       0,       1)

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