Computer Graph HW1 104062241 莫雅雯

1. Transformation

(1) Implement MVP matrices

```
void drawModel(model* model)
{
    Matrix4 T, R, S;
    T = translate(model->position);
    R = rotate(model->rotation);
    S = scaling(model->scale);

    // [TODO] Assign MVP correct value
    Matrix4 M = T*S*R;
    Matrix4 MVP;
    if (projection)
    {
        setOrthogonal();
    }
    else
    {
        setPerspective();
    }
    setViewingMatrix();

    MVP = project_matrix * view_matrix*M;
```

在 drawModel function 裡

(1) call translate、rotate、scaling function 並分別傳入 model 的 position、rotation、scale。
得到矩陣 T、R、S，並將它們乘起來。

(2) 根據 projection 是 true or false 判斷 call setOrthogonal 或 setPerspective。
改變 project_matrix

(3)call setViewingMatrix。
改變 view_matrix

(4) MVP = project_matrix * view_matrix*M

(2) Geometrical transformation

a. Translation

```
Matrix4 translate(Vector3 vec)
{
    Matrix4 mat;
    mat = Matrix4(
        1, 0 , 0,vec.x,
        0, 1,  0, vec.y,
        0, 0,  1, vec.z,
        0, 0,  0, 1
    );
    return mat;
}
```

把 translation 的轉換 matrix 填入 translate function。

$$\begin{aligned} x' &= x + d_x \\ y' &= y + d_y \\ z' &= z + f_z \end{aligned} \qquad T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b. Rotation

```
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;
}
```

把對應的的轉換 matrix 填入 rotateX、rotateY、rotateZ function。

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

```
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;
}
```

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

```
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;
}
```

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

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

將 rotateX、rotateY、rotateZ 得到的 matrix 乘起來。

c. Scaling

```
Matrix4 scaling(Vector3 vec)
{
    Matrix4 mat;
    mat = Matrix4(
    vec.x, 0, 0, 0,
    0, vec.y, 0, 0,
    0, 0, vec.z, 0,
    0, 0, 0, 1
    );
    return mat;
}
```

把 scaling 的轉換 matrix 填入 scaling function。

$$x' = s_x \cdot x$$
$$y' = s_y \cdot y$$
$$z' = s_z \cdot z$$

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

(3) Viewing transformation

a. center、eye

```
void setViewingMatrix()
{
    Vector3 p1 = main_camera.center - main_camera.position;
    Vector3 rz = (p1).normalize();
    Vector3 p2 = main_camera.up_vector - main_camera.position;
    Vector3 rx = p1.cross(p2).normalize();
    Vector3 ry = rx.cross(rz).normalize();

    Matrix4 R=Matrix4(
        rx[0], rx[1], rx[2], 0,
        ry[0], ry[1], ry[2], 0,
        -rz[0], -rz[1], -rz[2], 0,
        0, 0, 0, 1);
    Matrix4 T= 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);
    view_matrix = R*T;
}
```

實作 viewing matrix 的部分，求出 view_matrix

因為作業跟講義 z 軸差一個負號，所以講義的公式要修改成:

$$R_z = [r_{1z} \quad r_{2z} \quad r_{3z}]^T = \frac{-\overrightarrow{P_1P_2}}{|\overrightarrow{P_1P_2}|}$$

$$R_x = [r_{1x} \quad r_{2x} \quad r_{3x}]^T = \frac{\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}}{|\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}|}$$

$$R_y = [r_{1y} \quad r_{2y} \quad r_{3y}]^T = R_z \times R_x$$

(Rz 紅色部分負號拿掉)

$$M_{view} = R \bullet T = \begin{bmatrix} r_{1x} & r_{2x} & r_{3x} & 0 \\ r_{1y} & r_{2y} & r_{3y} & 0 \\ r_{1z} & r_{2z} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(r1z,r2z,r3z 部分加負號)

(4) Projection transformation

a. Orthogonal

```
void setOrthogonal()
{
    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.nearClip-proj.farClip), -(proj.farClip+proj.nearClip)/(proj.farClip-proj.nearClip),
        0, 0, 0, 1
    );
}
```

實作 project_matrix 的部分:

把 Orthogonal 的 matrix 填入 setOrthogonal function。

因為作業跟講義 z 軸差一個負號，所以講義的公式要修改成:

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

(2/zfar-znear)那部分要加個負號。

b. Perspective

```
void setPerspective()
{
    project_matrix = Matrix4(
        2 * proj.nearClip /(proj.right-proj.left), 0, -(proj.right+proj.left)/(proj.right-proj.left), 0,
        0, 2 * proj.nearClip /(proj.top -proj.bottom), -(proj.top +proj.bottom)/(proj.top -proj.bottom), 0,
        0, 0,(proj.farClip + proj.nearClip)/(proj.nearClip-proj.farClip),-(2*proj.farClip*proj.nearClip)/(proj.farClip-proj.nearClip),
        0, 0, -1, 0
    );
}
```

實作 project_matrix 的部分:

把 Perspective 的 matrix 填入 setPerspective function。

因為作業跟講義 z 軸差一個負號，所以講義的公式要修改成:

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

(zfar+znear/znear- zfar)那部分要加個負號。

1 也要變成-1。

2. Control

(1) 設一個全域 int mode 紀錄現在處於哪種轉換模式

```
int mode = 0;
/*
0: default
1:  translate mode
2:  rotate mode
3:  scale mode
4:  center translate mode
5:  eye translate mode
*/
```

(2) 設一個全域 bool projection(預設=true)，用來判斷現在處於 Orthogonal 或 Perspective。

```
case 'o':
    projection = true;
    printf("o : go to orthogonal projection mode\n");
    break;
case 'p':
    projection = false;
    printf("p : go to perspective projection mode\n");
    break;
```

(3) 鍵盤&滑鼠

根據按下甚麼按鍵改變 mode 值，然後在滑鼠那裏判斷 mode 值改變對應的轉換的數值。

原本 onKeyboard function 裡只有判斷小寫字母，我多加了大寫的判斷。

按下 i/I 鍵會顯示 model&camera 的各種資訊、位於甚麼模式之下。

```
mode position : x = -0.250000 , y = 0.250000 , z = 0.000000
mode rotation : x = 0.000000 , y = 0.000000 , z = 0.000000
mode scaling : x = 1.000000 , y = 1.000000 , z = 1.000000
camera position : x = 0.000000 , y = 0.000000 , z = 2.000000
center position : x = 0.000000 , y = 0.000000 , z = 0.000000
projection : orthogonal projection
you are in object translate mode
```