作业1 - MVP变换

要求

填写一个旋转矩阵和一个透视投影矩阵。给定三维下三个点：

$v_0(2.0, 0.0, -2.0), v_1(0.0, 2.0, -2.0), v_2(-2.0, 0.0, -2.0)$

需要将这三个点的坐标变换为屏幕坐标并在屏幕上绘制出对应的线框三角形 (在代码框架中已经提供了 draw_triangle 函数，所以只需要去构建变换矩阵即可)。简而言之，需要进行模型Model、视图View、投影Projection、视口等变换来将三角形显示在屏幕上。在提供的代码框架中，需要完成模型变换和投影变换的部分。

思路

- 实现如下`main.cpp`中如下接口：

```
// Create the model matrix for rotating the triangle around the Z axis.
Matrix4f get_model_matrix(float rotation_angle);

// Create the projection matrix for the given parameters.
Matrix4f get_projection_matrix(float eye_fov,
                float aspect_ratio,
                float zNear,
                float zFar);
```

即 MVP 变换中的 Model Transform 和 Project Transform。只需要分别实现对应的矩阵即可。由于View Transform已在框架中已实现(`draw_triangle()`)，故不需要知道摄像机位置。

Model Transform模型变换

- `Matrix4f get_model_matrix(float rotation_angle)`
  功能：将逐个元素地构建模型变换矩阵并返回该矩阵。在此函数中，只需要实现三维中绕 z 轴旋转的变换矩阵，而不用处理平移与缩放。
  参数：
  - `float rotation_angle` 旋转角度 需要转化为弧度形式
    c++ 的标准接口 sin 和 cos 接收的参数是弧度
- **实现**

$$\mathbf{R}_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

创建新的旋转矩阵`rotation`,套公式即可。

```
Matrix4f model = Matrix4f::Identity();
double angle = (double)rotation_angle / 180.0 * MY_PI;//转换旋转角度
Matrix4f rotation = Matrix4f::Identity();//绕z轴的旋转矩阵

rotation << cos(angle), -sin(angle), 0, 0,
            sin(angle), cos(angle), 0, 0,
            0, 0, 1, 0,
            0, 0, 0, 1;

model = rotation * model;
return model;
```

Project Transform投影变换

- `Matrix4f get_projection_matrix(float eye_fov, float aspect_ratio,float zNear, float zFar)`

  功能：使用给定的参数逐个元素地构建透视投影矩阵并返回该矩阵。

  参数：

  - `eye_fov`：视角宽度/角度（上下or左右视角），框架中width/height = 1：1
  - `aspect_ratio`：宽高比
  - `zNear`：近平面Z值,此处给正 需要取负值（往z负方向看）
  - `zFar`：原平面Z值

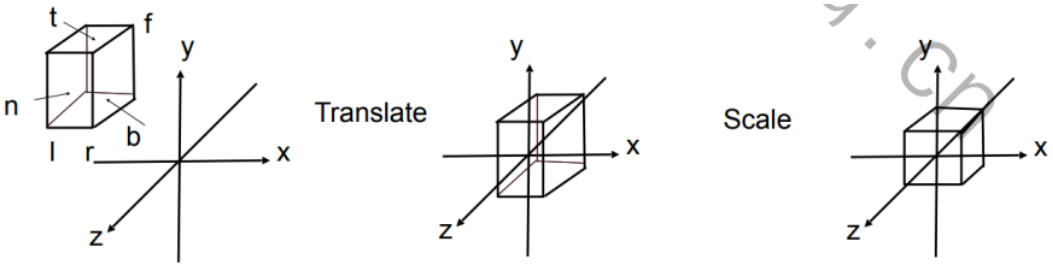
- **实现**：

  做一次透视投影，再做正交投影，将物体缩放到合适比例并移动到合适位置

Rotation by angle α around axis $\mathbf{n}$

$$\mathbf{R}(\mathbf{n},\alpha) = \cos(\alpha)\,\mathbf{I} + (1-\cos(\alpha))\,\mathbf{n}\mathbf{n}^T + \sin(\alpha)\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

正交变换矩阵



对空间中 [l,r]×[b,t]×[f,n] 的体积成像，将其变换到 [−1,1]^3 ，那么它的变换可以分解为：先平移这个空间的中心到原点，再放缩

*注：此处看向z轴负方向

只求出找到左边&右边矩阵对应参数即可：

$$n = -zNear,\ f = -zFar,\ t = tan(alpha/2) * abs(n), r = t * aspect\_ratio$$

$$b = -t, l = -r$$

提高

思路

完成绕任意过原点轴的旋转变换矩阵`Matrix4f get_rotation(Vector3f axis, float angle)`

需要参考如下公式：

Rotation by angle α around axis n

$$\mathbf{R}(\mathbf{n},\alpha) = \cos(\alpha)\mathbf{I} + (1-\cos(\alpha))\mathbf{n}\mathbf{n}^T + \sin(\alpha)\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

罗德里格旋转公式：表示任意旋转后向量n

注意：公式中矩阵皆为三维，由于项目统一将坐标使用齐次化，返回四维坐标，故需要转换。

实现

```
Matrix4f get_rotation(Vector3f axis, float angle) {//返回绕任意过原点轴的旋转变换矩阵

    float alpha = angle / 180 * MY_PI;
    Matrix3f N = Matrix3f::Identity();
    N << 0, -axis.z(), axis.y(),
         axis.z(), 0, -axis.x(),
         -axis.y(), axis.x(), 0;

    Matrix3f rod = cos(alpha) * Matrix3f::Identity() + (1 - cos(alpha)) * axis *
axis.transpose() + sin(alpha) * N;

    Matrix4f model = Matrix4f::Identity();

    model << rod(0, 0), rod(0, 1), rod(0, 2), 0,
             rod(1, 0), rod(1, 1), rod(1, 2), 0,
             rod(2, 0), rod(2, 1), rod(2, 2), 0,
             0, 0, 0, 1;
    return model;
}
```

还需要在`main()`&`rasterizer.cpp`中补全相关操作逻辑，这部分仿照着做即可

```
while (key != 27) {
    r.clear(rst::Buffers::Color | rst::Buffers::Depth);

    r.set_model(get_model_matrix(angle));
    r.set_view(get_view_matrix(eye_pos));
    r.set_projection(get_projection_matrix(45, 1, 0.1, 50));

    if (rflag)//区分
    {
        r.set_rodigues(get_rotation(raxis, rangle));
    }
    else
    {
        r.set_rodigues(get_rotation({0,0,1},0));
    }
    r.draw(pos_id, ind_id, rst::Primitive::Triangle);

    cv::Mat image(700, 700, CV_32FC3, r.frame_buffer().data());
    image.convertTo(image, CV_8UC3, 1.0f);
    cv::imshow("image", image);
    key = cv::waitKey(10);

    cout << "frame count: " << frame_count++ << '\n';

    if (key == 'a') {//按下a，逆时针旋转10°
```

效果：