网上的答案，但感觉理解不透彻，明天抽时间自己具体从头算一下

Eigen::Matrix4f get\_model\_matrix(float rotation\_angle)

{

// TODO: Implement this function

// Create the model matrix for rotating the triangle around the Z axis.

// Then return it.

Eigen::Matrix4f model = Eigen::Matrix4f::Identity();//定义 4\*4 单位矩阵

float r = rotation\_angle / 180.0 \* MY\_PI;//定义旋转的弧度

Eigen::Matrix4f translate;//初始化模型变换矩阵

translate << cos(r), -sin(r), 0, 0,

sin(r), cos(r), 0, 0,

0, 0, 1, 0,

0, 0, 0, 1;

model = translate \* model;

return model;

}

Eigen::Matrix4f get\_projection\_matrix(float eye\_fov, float aspect\_ratio,

float zNear, float zFar)

{

// Students will implement this function

Eigen::Matrix4f projection = Eigen::Matrix4f::Identity();//定义 4\*4 单位矩阵

// TODO: Implement this function

// Create the projection matrix for the given parameters.

// Then return it.

Eigen::Matrix4f persp\_to\_ortho = Eigen::Matrix4f::Identity();

persp\_to\_ortho << -zNear, 0, 0, 0,

0, -zNear, 0, 0,

0, 0, -zNear + -zFar, -zNear \* zFar,

0, 0, 1, 0;

float half\_eye\_fovY = eye\_fov / 2 / 180.0 \* MY\_PI;

float top = zNear \* tan(half\_eye\_fovY);

float bottom = -top;

float right = aspect\_ratio \* top;

float left = -right;

Eigen::Matrix4f ortho\_translate = Eigen::Matrix4f::Identity();

ortho\_translate << 1, 0, 0, -(right + left) / 2,

0, 1, 0, -(top + bottom) / 2,

0, 0, 1, -(zNear + zFar) / 2,

0, 0, 0, 1;

Eigen::Matrix4f ortho\_scale = Eigen::Matrix4f::Identity();

ortho\_scale << 2 / (right - left), 0, 0, 0,

0, 2 / (top - bottom), 0, 0,

0, 0, 2 / (zFar - zNear), 0,

0, 0, 0, 1;

projection = ortho\_scale \* ortho\_translate \* persp\_to\_ortho;

return projection;

}