- 模型变换 goe_model_matrix

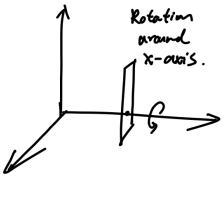
★绕 z轴旋转的变换矩阵

Rodrigues's Rotation Formula.

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

Rotation around x- y- or z-axis

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



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

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

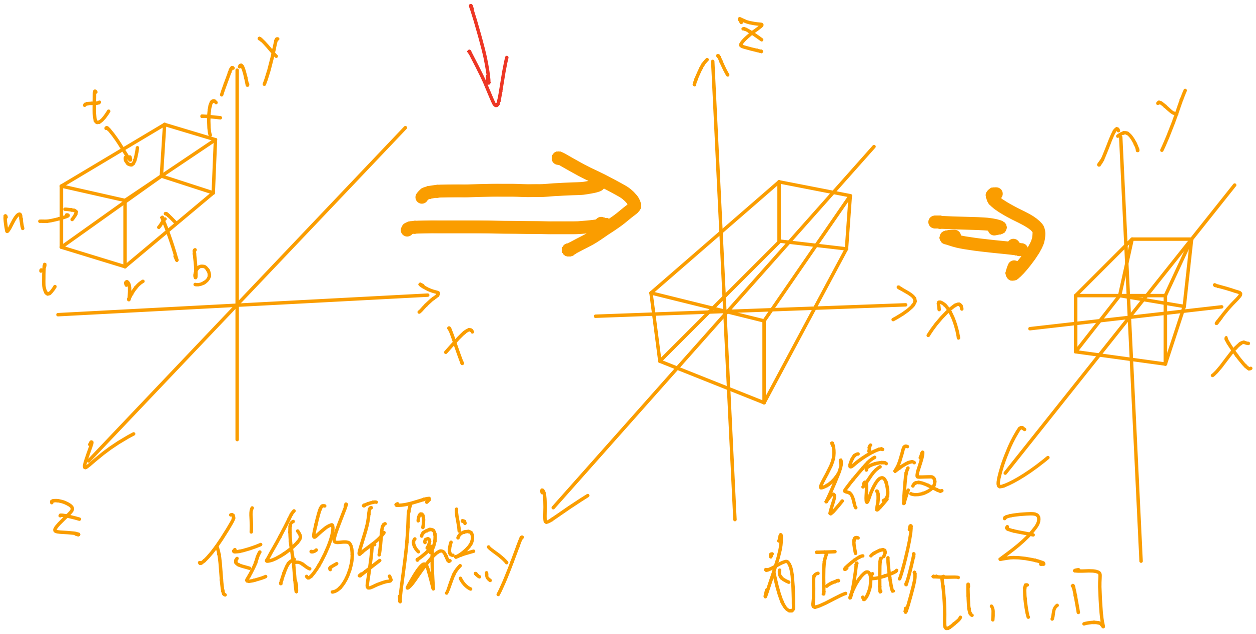
三个轴的旋转矩阵合为一个公式表达.

◦ 透视投影变换矩阵.

透视投影矩阵 = 正交矩阵 · 透视矩阵

$$M_{persp} = M_{ortho} M_{persp \to ortho}.$$

$$M_{ortho} = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

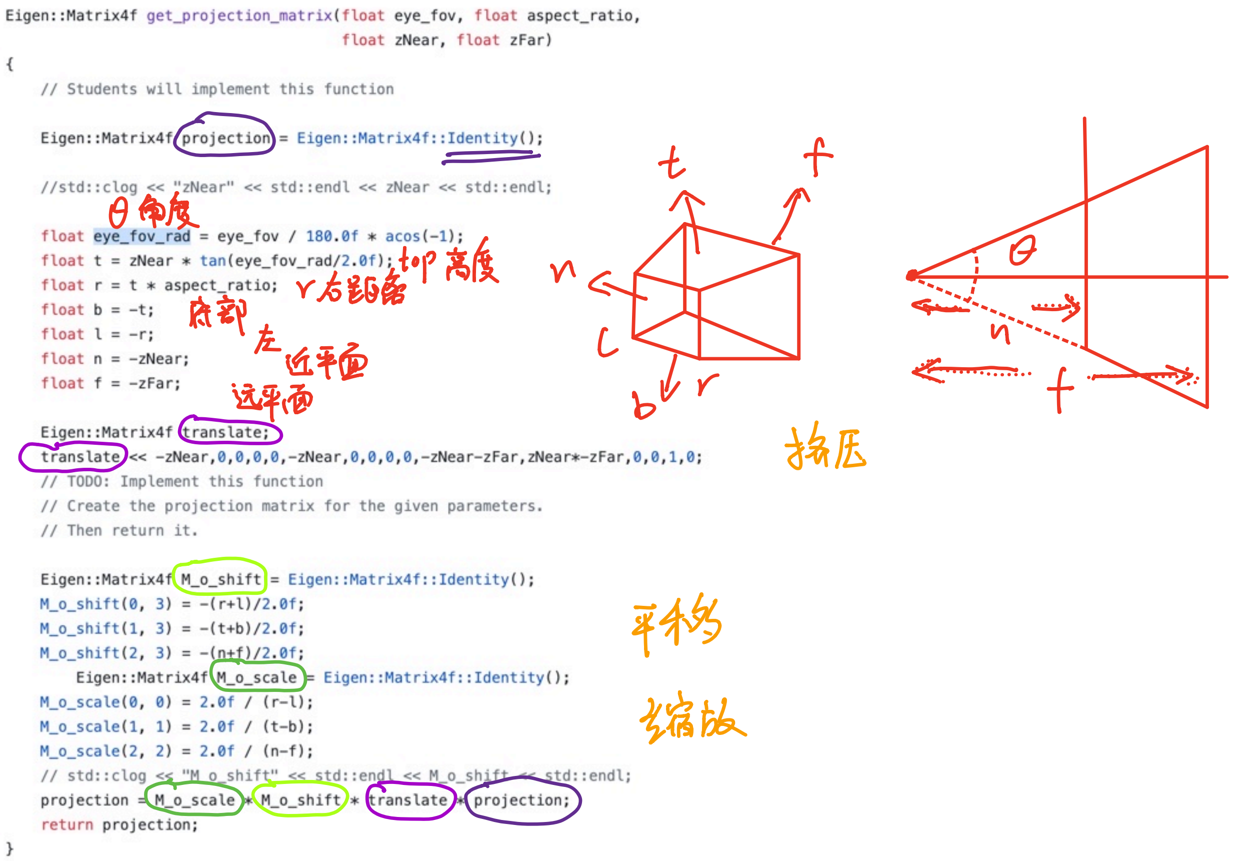


$M_{persp \to ortho}$

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$M_{persp} = M_{ortho} M_{persp \to ortho}.$$

$$\underset{\text{平移}}{\begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \quad \underset{\text{缩放}}{\begin{bmatrix} 1 & 0 & 0 & -\frac{r+l}{2} \\ 0 & 1 & 0 & -\frac{t+b}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix}} \quad \underset{\text{挤压}}{\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -nf \\ 0 & 0 & 1 & 0 \end{bmatrix}}$$



```
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();

    //std::clog << "zNear" << std::endl << zNear << std::endl;

    float eye_fov_rad = eye_fov / 180.0f * acos(-1);
    float t = zNear * tan(eye_fov_rad/2.0f);
    float r = t * aspect_ratio;
    float b = -t;
    float l = -r;
    float n = -zNear;
    float f = -zFar;

    Eigen::Matrix4f translate;
    translate << -zNear,0,0,0,0,-zNear,0,0,0,0,-zNear-zFar,zNear*-zFar,0,0,1,0;
    // TODO: Implement this function
    // Create the projection matrix for the given parameters.
    // Then return it.

    Eigen::Matrix4f M_o_shift = Eigen::Matrix4f::Identity();
    M_o_shift(0, 3) = -(r+l)/2.0f;
    M_o_shift(1, 3) = -(t+b)/2.0f;
    M_o_shift(2, 3) = -(n+f)/2.0f;
        Eigen::Matrix4f M_o_scale = Eigen::Matrix4f::Identity();
    M_o_scale(0, 0) = 2.0f / (r-l);
    M_o_scale(1, 1) = 2.0f / (t-b);
    M_o_scale(2, 2) = 2.0f / (n-f);
    // std::clog << "M_o_shift" << std::endl << M_o_shift << std::endl;
    projection = M_o_scale * M_o_shift * translate * projection;
    return projection;
}
```