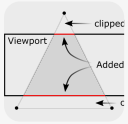
OpenGL投影矩阵

学无止境
猪八戒肚子鼓鼓的!

> 下面链接是原文的地址:

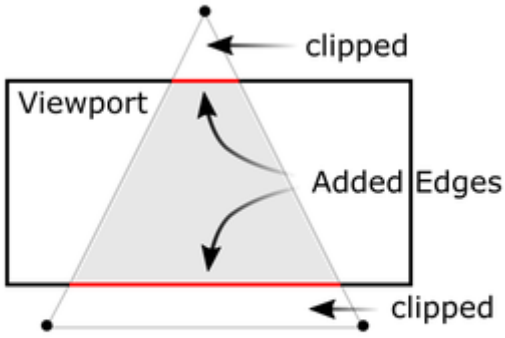
OpenGL Projection Matrix
www.songho.ca/opengl/gl_projectionmat...



- **概述**

我们知道计算机屏幕是2D的。利用OpenGL渲染3D场景会将场景投射成屏幕空间的2D图片。OpenGL的GL_PROJECTION矩阵就是用来进行投影变换的。首先需要将顶点着色器输入的顶点数据从观察坐标(eye coordinates)变换到裁剪坐标(clip coordinates)；然后将裁剪坐标变换到标准设备空间(normalized device coordinates, NDC)。
因此，我们需要知道GL_PROJECTION矩阵集合了裁剪(视椎体剔除)以及NDC坐标变换的功能。接下来的部分描述如何通过视椎体上下左右近远六个平面的边界值来构建我们的投影矩阵。
我们注意到视椎体剔除(裁剪)是在裁剪空间下进行的，而且是在进行透射除法(各个分量除以)之前进行。裁剪空间下的坐标、和需要和分量进行比较，如果裁剪坐标小于-或者大于，那么该顶点将被丢弃，也就是-Wc < Xc、Yc、Zc < Wc。所以当发生裁剪时，OpenGL将会重新构建多边形被裁减的边，如下图所示。

(注：由于在透射除法之前会进行裁剪操作，所以的值不会等于0，我们无需担心发生除以0的情况，因为近平面的z值应当大于0)

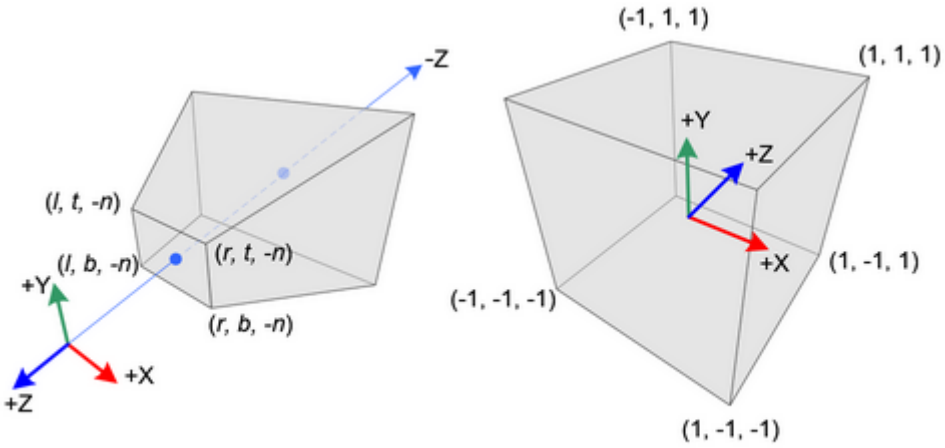


A triangle clipped by frustum

图元被视椎体裁剪的示意图

- **透射投影**

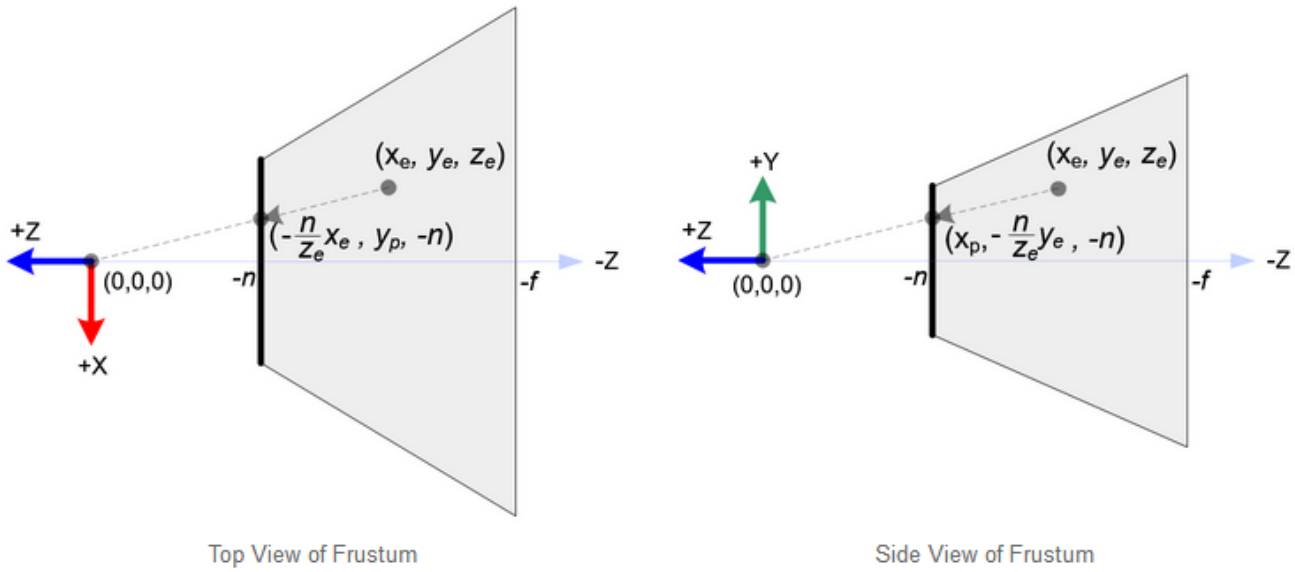
在透射投影中，投影视椎体类似一个头部被裁剪的金字塔。观察空间中位于视椎体内的3D点会被映射到NDC空间中；该映射将x坐标从[l, r]映射到[-1,1]，y坐标从[b, t]映射到[-1,1]，z坐标从[n, f]映射到[-1,1]。需要注意的是观察空间是使用的是右手坐标系，而NDC空间使用的是左手坐标系，因为摄像机在观察空间原点沿着Z轴的负方向看过去，而在NDC空间是沿着Z轴的正方向看过去。当我们利用glFrustun()函数构建GL_PROJECTION矩阵时，需要注意该函数接收的参数near和far要求是正数，所以我们需要对这两个参数进行取反的操作。



Perspective Frustum and Normalized Device Coordinates (NDC)

透射投影视椎体以及NDC坐标

在OpenGL中，观察空间的3D点会被投影到近平面(投影平面)。下面两张图展示了观察空间的点(Xe, Ye, Ze)如何被投射为近平面上的点(Xp, Yp, Zp)。
(注：文中的下标c, e, p, n分别代表的是裁剪空间、观察空间、投影空间和NDC空间)



透视投影视椎体的俯视图和侧视图

第一张图是视椎体的顶视图，从图中我们可以知道观察空间中的x坐标被映射为，我们可以通过相似三角形的方法计算：

$$\frac{x_p}{x_e} = \frac{-n}{z_e}$$

$$x_p = \frac{-n \cdot x_e}{z_e} = \frac{n \cdot x_e}{-z_e}$$

相似三角形求xp

第二张图为视椎体的侧视图，利用相同的方法我们可以计算：

$$\frac{y_p}{y_e} = \frac{-n}{z_e}$$

$$y_p = \frac{-n \cdot y_e}{z_e} = \frac{n \cdot y_e}{-z_e}$$

相似三角形求yp

我们发现和都和有关，它们与成反比，也就是说他们都除以，这是我们构建GL_PROJECTION矩阵的第一个线索。当观察坐标乘以GL_PROJECTION后得到的裁剪坐标是齐次坐标。通过除以w分量我们可以将齐次裁剪坐标变为NDC坐标。

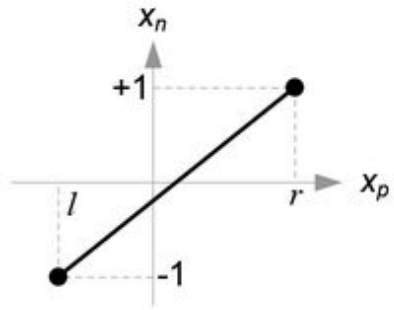
$$\begin{pmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{pmatrix} = M_{projection} \cdot \begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix}, \quad \begin{pmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{pmatrix} = \begin{pmatrix} x_{clip}/w_{clip} \\ y_{clip}/w_{clip} \\ z_{clip}/w_{clip} \end{pmatrix}$$

eye表示观察空间坐标，clip表示裁剪空间坐标，ndc表示标准设备坐标

因此，我们可以设置w分量为-，GL_PROJECTION矩阵的第四行变为(0, 0, -1, 0)。
(注：投射投影变换有线性和非线性两个部分组成，线性部分是投影矩阵变换到裁剪空间的过程，非线性变换是透射除法除以z的过程。由于我们后续的变换会对z坐标进行归一化处理，所以在进行透射除法时我们无法知道最初z坐标的值，所以我们在变换之前利用w分量存储了z坐标)

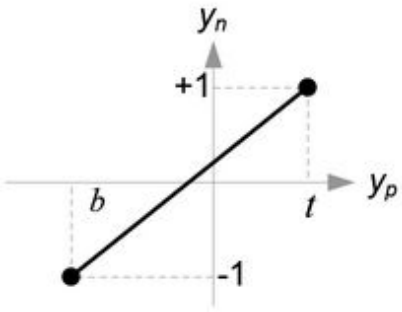
$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}, \qquad \therefore w_c = -z_e$$

接下来我们将线性的映射到，也就是[l, r] => [-1,1]、[b, t]=>[-1,1]。



Mapping from x_p to x_n

$$\begin{aligned} x_n &= \frac{1-(-1)}{r-l} \cdot x_p + \beta \\ 1 &= \frac{2r}{r-l} + \beta \qquad (\text{substitute } (r,1) \text{ for } (x_p, x_n)) \\ \beta &= 1 - \frac{2r}{r-l} = \frac{r-l}{r-l} - \frac{2r}{r-l} \\ &= \frac{r-l-2r}{r-l} = \frac{-r-l}{r-l} = -\frac{r+l}{r-l} \\ \therefore x_n &= \frac{2x_p}{r-l} - \frac{r+l}{r-l} \end{aligned}$$



Mapping from y_p to y_n

$$\begin{aligned} y_n &= \frac{1-(-1)}{t-b} \cdot y_p + \beta \\ 1 &= \frac{2t}{t-b} + \beta \qquad (\text{substitute } (t,1) \text{ for } (y_p, y_n)) \\ \beta &= 1 - \frac{2t}{t-b} = \frac{t-b}{t-b} - \frac{2t}{t-b} \\ &= \frac{t-b-2t}{t-b} = \frac{-t-b}{t-b} = -\frac{t+b}{t-b} \\ \therefore y_n &= \frac{2y_p}{t-b} - \frac{t+b}{t-b} \end{aligned}$$

视椎体和NDC坐标之间进行线性映射

然后我们将刚才得到的代入上面的方程得到：

$$
\begin{aligned}
x_n &= \frac{2x_p}{r-l} - \frac{r+l}{r-l} \qquad \left(x_p = \frac{nx_e}{-z_e}\right) \\
&= \frac{2 \cdot \frac{n \cdot x_e}{-z_e}}{r-l} - \frac{r+l}{r-l} \\
&= \frac{2n \cdot x_e}{(r-l)(-z_e)} - \frac{r+l}{r-l} \\
&= \frac{\frac{2n}{r-l} \cdot x_e}{-z_e} - \frac{r+l}{r-l} \\
&= \frac{\frac{2n}{r-l} \cdot x_e}{-z_e} + \frac{\frac{r+l}{r-l} \cdot z_e}{-z_e} \\
&= \underbrace{\left(\frac{2n}{r-l} \cdot x_e + \frac{r+l}{r-l} \cdot z_e\right)}_{x_c} \Big/ -z_e
\end{aligned}
\qquad
\begin{aligned}
y_n &= \frac{2y_p}{t-b} - \frac{t+b}{t-b} \qquad \left(y_p = \frac{ny_e}{-z_e}\right) \\
&= \frac{2 \cdot \frac{n \cdot y_e}{-z_e}}{t-b} - \frac{t+b}{t-b} \\
&= \frac{2n \cdot y_e}{(t-b)(-z_e)} - \frac{t+b}{t-b} \\
&= \frac{\frac{2n}{t-b} \cdot y_e}{-z_e} - \frac{t+b}{t-b} \\
&= \frac{\frac{2n}{t-b} \cdot y_e}{-z_e} + \frac{\frac{t+b}{t-b} \cdot z_e}{-z_e} \\
&= \underbrace{\left(\frac{2n}{t-b} \cdot y_e + \frac{t+b}{t-b} \cdot z_e\right)}_{y_c} \Big/ -z_e
\end{aligned}
$$

透射投影NDC坐标可以通过裁剪坐标加上透射除法得到

大括号中的项分别是裁剪坐标，通过透射除法分别除以-Ze(Xc/Wc, Yc/Wc，我们之前将Wc设为-Ze)得到NDC坐标。从这两个等式我们可以得到GL_PROJECTION矩阵的第一行和第二行。

$$
\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}
$$

GL_PROJECTION矩阵第一行和第二行

现在GL_PROJECTION矩阵只剩下第三行需要求解。寻找的关系跟之前的方法有些不一样，因为在观察空间总被投射到-n近平面。但是我们需要不同的z值来进行裁剪以及深度测试，也就是说我们要能够通过反向投射(反变换)得到的值。由于z的值跟x、y没有关系，我们可以借助w分量来寻找和的关系，因此可以将GL_PROJECTION矩阵的第三行设为如下。

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ w_e \end{pmatrix}, \qquad z_n = z_c/w_c = \frac{Az_e + Bw_e}{-z_e}$$

GL_PROJECTION矩阵第三行的求解过程

在观察空间中，We=1，因此等式变为：

$$z_n = \frac{Az_e + B}{-z_e}$$

为了解出系数A和B，我们利用(,)的两个关系：(-n,-1)和(-f, 1)(注：该关系其实就是近平面映射到-1，远平面映射到1)将其代入以上的方程得到：

$$\begin{cases} \frac{-An + B}{n} = -1 \\ \frac{-Af + B}{f} = 1 \end{cases} \rightarrow \begin{cases} -An + B = -n & (1) \\ -Af + B = f & (2) \end{cases}$$

我们重写方程(1)得到：

$$B = An - n \qquad (1')$$

将方程(1`)代入方程(2)中的B解除A：

$$-Af + (An - n) = f \qquad (2)$$

$$-(f - n)A = f + n$$

$$A = -\frac{f + n}{f - n}$$

将A代入方程(1)解出B：

$$\left(\frac{f+n}{f-n}\right)n + B = -n \qquad (1)$$

$$B = -n - \left(\frac{f+n}{f-n}\right)n = -\left(1 + \frac{f+n}{f-n}\right)n = -\left(\frac{f-n+f+n}{f-n}\right)n$$
$$= -\frac{2fn}{f-n}$$

我们解出了A和B，因此我们得到了和 的关系为：

$$z_n = \frac{-\frac{f+n}{f-n}z_e - \frac{2fn}{f-n}}{-z_e} \qquad (3)$$

最后，我们得到了最终版本的GL_PROJECTION矩阵，该矩阵如下：

$$\begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

最终版本的GL_PROJECTION矩阵

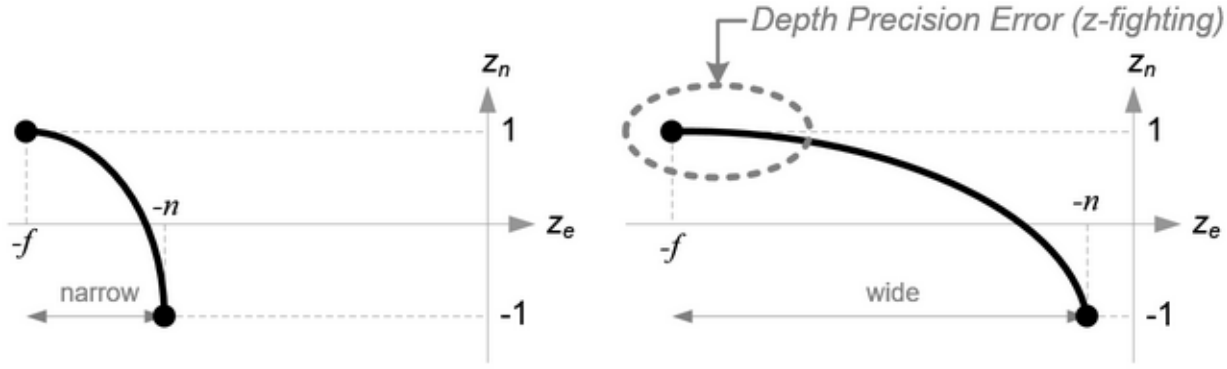
该矩阵适用于一般的视椎体，如果视椎体(viewing volume)是对称的，也就是说r = -l, t = -b，那么该矩阵可以进一步得到化解：

$$\begin{cases} r+l=0 \\ r-l=2r \ (\text{width}) \end{cases}, \quad \begin{cases} t+b=0 \\ t-b=2t \ (\text{height}) \end{cases}$$

$$\begin{pmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

对称视椎体的GL_PROJECTION矩阵

我们继续讨论之前，请再次看下方程(3)中和 的关系。你会发现该函数是一个有理函数而且和 具有非线性关系。这意味着靠近近平面具有比较好的精度，而靠近远平面具有很低的精度。如果[-n, -f]的距离不断变大，那么会带来深度精度不够的问题(z-fighting)。也就是说稍微改变远平面附近的值并不会引起值的改变。所以我们应该尽可能减少n和f之间的距离来最小化深度缓冲精度所带来的问题。

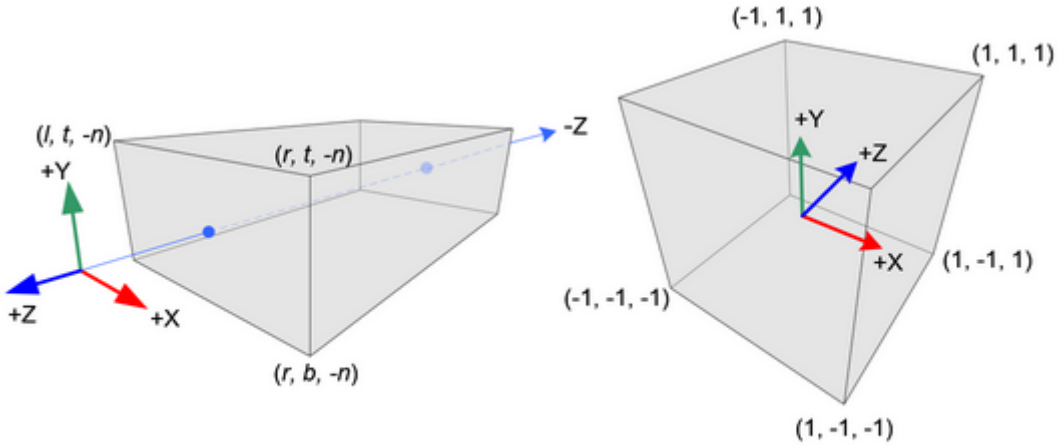


透视除法后z分量为非线性关系，可能会带来z-fighting的问题

(注：从图中可以看到我们构建的函数具有保序性，也就是说对深度值进行归一化处理之后，深度关系保持不变。所以，在实现深度缓冲算法中，我们仍能在归一化区间内正确的比较出不同点之间的深度关系)

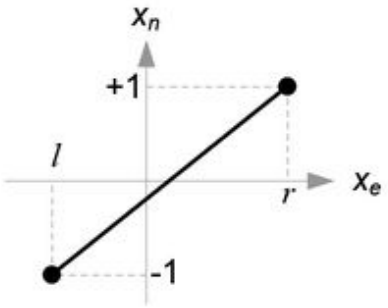
- **正交投影**

构建正交投影矩阵GL_PROJECTION相比投射投影要简单很多。所有在观察空间的线性的映射到NDC坐标。我们只需要将一个长方体包围盒缩放成一个正方体包围盒，然后平移到原点就可以。我们接下来通过线性的关系来求解GL_PROJECTION。



Orthographic Volume and Normalized Device Coordinates (NDC)

正交投影视椎体以及NDC坐标



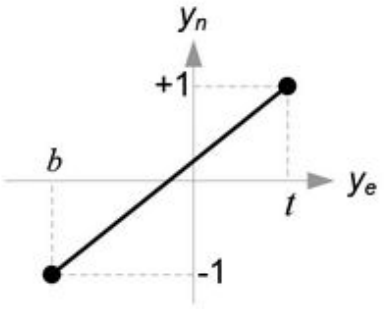
Mapping from x_e to x_n

$$x_n = \frac{1-(-1)}{r-l} \cdot x_e + \beta$$

$$1 = \frac{2r}{r-l} + \beta \qquad \text{(substitute } (r, 1) \text{ for } (x_e, x_n))$$

$$\beta = 1 - \frac{2r}{r-l} = -\frac{r+l}{r-l}$$

$$\therefore x_n = \frac{2}{r-l} \cdot x_e - \frac{r+l}{r-l}$$



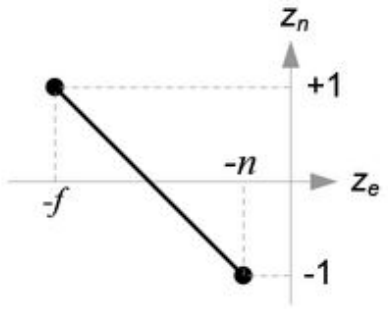
Mapping from y_e to y_n

$$y_n = \frac{1-(-1)}{t-b} \cdot y_e + \beta$$

$$1 = \frac{2t}{t-b} + \beta \qquad \text{(substitute } (t, 1) \text{ for } (y_e, y_n))$$

$$\beta = 1 - \frac{2t}{t-b} = -\frac{t+b}{t-b}$$

$$\therefore y_n = \frac{2}{t-b} \cdot y_e - \frac{t+b}{t-b}$$



Mapping from z_e to z_n

$$z_n = \frac{1-(-1)}{-f-(-n)} \cdot z_e + \beta$$

$$1 = \frac{2f}{f-n} + \beta \qquad \text{(substitute } (-f, 1) \text{ for } (z_e, z_n))$$

$$\beta = 1 - \frac{2f}{f-n} = -\frac{f+n}{f-n}$$

$$\therefore z_n = \frac{-2}{f-n} \cdot z_e - \frac{f+n}{f-n}$$

视椎体和NDC坐标之间进行线性映射

由于正交投影不需要使用到w分量，我们保留GL_PROJECTION矩阵的第四行为(0, 0, 0, 1)。因此，正交投影所使用的GL_PROJECTION矩阵最终版本如下：

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

OpenGL Orthographic Projection Matrix

正交投影GL_PROJECTION矩阵

如果视椎体是对称的话(r = -l, t = -b)，该矩阵可以进一步简化:

$$\begin{cases} r+l=0 \\ r-l=2r \ (\text{width}) \end{cases}, \begin{cases} t+b=0 \\ t-b=2t \ (\text{height}) \end{cases}$$

$$\begin{pmatrix} \frac{1}{r} & 0 & 0 & 0 \\ 0 & \frac{1}{t} & 0 & 0 \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

对称视椎体的GL_PROJECTION矩阵

- **知识扩展**

我们接下来看下与投影变换相关的一个问题：**拾取**。给出鼠标点击的2D屏幕坐标，如何确定此点对应的3D对象？我们需要做的是一个逆于投影变换的操作，通过屏幕空间变换会3D空间。
拾取操作一般分为下面4个步骤：

1. 根据屏幕的点s，求出投影窗口中对应的点p
2. 计算出位于观察空间的拾取射线。次射线以观察空间中的原点为起点，并经过p
3. 将拾取射线与场景中的物体变换到同一空间中
4. 确定与拾取射线相交的物体，与射线相交的距离摄像机最近的物体即为用户选中的物体

变换过程：

1. 通过视口变换矩阵逆矩阵将屏幕坐标变换到NDC坐标
2. 然后通过乘以W分量(投射除法的逆变换)将NDC坐标变换到裁剪坐标
3. 通过投影矩阵逆矩阵将裁剪坐标变换到观察坐标

4. 求出经过原点O以及点的拾取射线
5. 拾取射线位于观察空间，通过将拾取射线变换到局部空间进行相交行检测

编辑于 2019-07-31 20:05

计算机图形学 3D 渲染 游戏开发