# OpenGL学习脚印：投影矩阵的推导

2014-10-28 方海龙的... 摘自 csdn博客 阅 1863 转 17 分享： 微信 转藏到我的图书馆



## OpenGL学习脚印: 投影矩阵的推导

写在前面

本节内容翻译和整理自http://www.songho.ca songho的博客《OpenGL Projection Matrix》内容,以供自己和初学者熟悉投影矩阵推导过程。

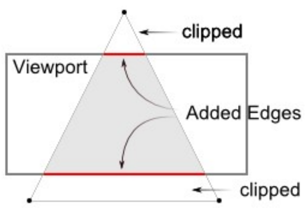
**通过本节，你可以了解到:**

投影矩阵计算的阶段
透视投影和正交投影的矩阵推导

**本节的要点就在于: 阅读时，自己拿笔推导一遍。**

**1.概览**

计算机屏幕是2维的，OpenGL渲染的3D场景必须以2D形式的图像投影到屏幕上。GL_PROJECTION 矩阵就是用来设置投影变换的。首先，它将所有顶点从眼坐标(照相机坐标)转换到裁剪坐标系下。然后，这些裁剪坐标通过透视除法，即除以裁剪坐标中w分量，转换到归一化设备坐标系(NDC)。




一个由视锥裁剪的三角形

因此，我们要记住，裁剪(视锥剔除frustum culling)和NDC转换都集成到了GL_PROJECTION 矩阵。接下来的部分描述了怎么样通过left, right, bottom, top, near and far 这6个界限参数来构造投影矩阵。

**注意:**

视锥剔除是在裁剪坐标系中进行的，并且恰好在透视除法之前进行。裁剪坐标xc, yc 和 zc 通过与wc比较来进行测试。 如果某个坐标值比Wc小或者比Wc大，那么这个顶点将被丢弃。然后，OpenGL会重新在裁剪进行的地方构造多边形的边缘。

**补充内容:**

实际上，眼坐标系下坐标在乘以投影矩阵后，裁剪测试和透视除法都是由GPU来执行的。而后面这两个过程处理的裁剪坐标系数据都是由投影矩阵变换的。



1. 裁剪测试也即视锥剔除

$-Wc < Xc,Yc,Zc < Wc$

2. NDC透视除法

$Xn = Xc / Wc \quad Yn = Yc / Wc \quad Zn = Zc / Wc$

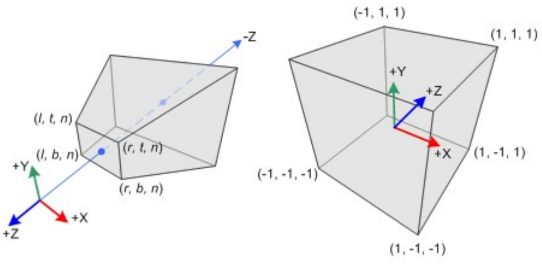
这里需要注意的是，我们在构造16个参数的投影矩阵的同时，不仅要考虑到裁剪，还要考虑到透视除法的过程。这样，最终的NDC坐标才会满足:

$-1 < Xn,Yn,Zn < 1$

在透视投影中，在眼坐标下截头椎体(a truncated pyramid frustum)内的3D点被映射到NDC下一个立方体中；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]。

**注意:**

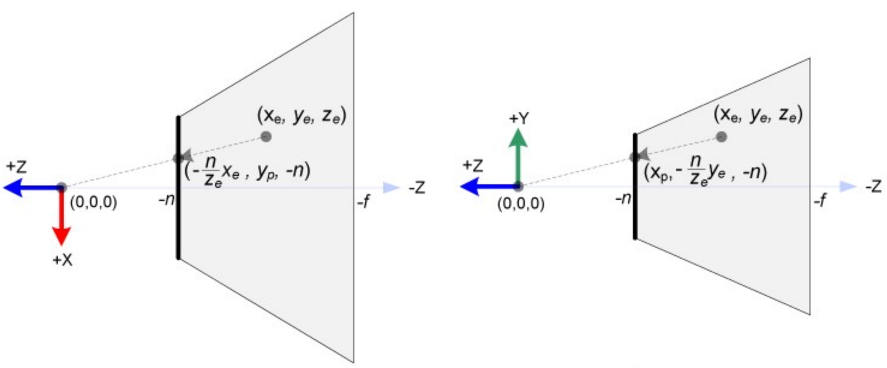
眼坐标系使用右手坐标系，而NDC使用左手坐标系。这就是说，眼坐标系下，在原点处的照相机朝着-Z轴看去，但是在NDC中它朝着+Z轴看去。因为glFrustum() 仅接受正的near和far距离，我们在构造GL_PROJECTION 矩阵时，需要取其相反数。眼坐标系和NDC坐标系如下图所示:




Perspective Frustum and Normalized Device Coordinates (NDC)




在OpenGL中,眼坐标下3D点被投影到近裁剪面(即投影平面)。下图展示了眼坐标系下点(xe, ye, ze) 如何投影到近裁剪面上的 (xp, yp, zp) 的。(左侧是视锥的俯视图，右侧是视锥的侧视图,拿出右手构成右手坐标系，然后比划比划就出来了)




Top View of Frustum　　Side View of Frustum

根据三角形的相似性，由俯视图可得出:

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

由侧视图可以得出:

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

**补充:** $x_p$ 和$y_p$其实是一个中间值，我们要找的是(Xc, Yc, Zc)和 (Xn, Yn, Zn)之间的关系，但是可以利用:

$$x_p = \frac{nx_e}{-z_e}$$
$$y_p = \frac{ny_e}{-z_e}$$
$$z_p = -n$$

这一关系做过渡，后面利用$x_p$和$y_p$，映射到NDC中$x_n$和$y_n$的线性关系就利用到了 $x_p$ 和$y_p$。这一点很重要。

赖于$z_e$，他们与 $-z_e$成反比，换句话说，他们都被 $-z_e$相除，这个事构造GL_PROJECTION 矩阵最初的线索。在眼坐标通过乘以 GL_PROJECTION 来转换时，裁剪坐标系仍然是一个齐次坐标系。通过对裁剪坐标进行透视除法得到最终的NDC坐标。下图解释了这个过程:

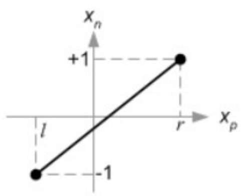
$$\begin{pmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{pmatrix} = M_{projection} \cdot \begin{pmatrix} x_{eye} \\ y_{eye} \\ z_{eye} \\ w_{eye} \end{pmatrix}, \quad \begin{pmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \end{pmatrix} = \begin{pmatrix} x_{clip}/w_{clip} \\ y_{clip}/w_{clip} \\ z_{clip}/w_{clip}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

因此我们可以把裁剪坐标系下的w分量设为$-z_e$，那么GL_PROJECTION矩阵第4行变为(0, 0, -1, 0),如下图(**求出了投影矩阵第4行**):

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

下现在我们把$x_p$和$y_p$，映射到NDC中$x_n$和$y_n$，他们之间是线性关系: [l, r] ? [-1, 1]和[b, t] ? [-1, 1].

线性关系如下图所示:




Mapping from $x_p$ to $x_n$

则可以推导出:

$$x_n = \frac{1-(-1)}{r-l} \cdot x_p + \beta$$
$$1 = \frac{2r}{r-l} + \beta \quad \text{// substitute } (x_p, x_n) \text{ with } (r, 1)$$

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

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

细节部分有删节,这个推导过程使用的就是简单的y=kx+b线性关系推导，同理利用[b, t] ? [-1, 1]可推得:

$$y_n = \frac{2y_p}{t-b} - \frac{t+b}{t-b}$$

将上面的 $x_p$ 和$y_p$带入求得:

$$\begin{aligned} x_n &= \frac{2x_p}{r-l} - \frac{r+l}{r-l} \qquad \left(x_p = \frac{nx_e}{-z_e}\right) \\ &= \frac{2 \cdot \frac{n \cdot x_e}{-z_e}}{r-l} - \frac{r+l}{r-l} \\ &= \frac{2n \cdot x_e}{(r-l)(-z_e)} - \frac{r+l}{r-l} \\ &= \frac{\frac{2n}{r-l} \cdot x_e}{-z_e} - \frac{r+l}{r-l} \\ &= \frac{\frac{2n}{r-l} \cdot x_e}{-z_e} + \frac{\frac{r+l}{r-l} \cdot z_e}{-z_e} \\ &= \left(\frac{2n}{r-l} \cdot x_e + \frac{r+l}{r-l} \cdot z_e\right) \Big/ -z_e \end{aligned}$$

$$\begin{aligned} y_n &= \frac{2y_p}{t-b} - \frac{t+b}{t-b} \qquad \left(y_p = \frac{ny_e}{-z_e}\right) \\ &= \frac{2 \cdot \frac{n \cdot y_e}{-z_e}}{t-b} - \frac{t+b}{t-b} \\ &= \frac{2n \cdot y_e}{(t-b)(-z_e)} - \frac{t+b}{t-b} \\ &= \frac{\frac{2n}{t-b} \cdot y_e}{-z_e} - \frac{t+b}{t-b} \\ &= \frac{\frac{2n}{t-b} \cdot y_e}{-z_e} + \frac{\frac{t+b}{t-b} \cdot z_e}{-z_e} \\ &= \left(\frac{2n}{t-b} \cdot y_e + \frac{t+b}{t-b} \cdot z_e\right) \Big/ -z_e \end{aligned}$$

注意这里Xn和Yn已经是NDC中的坐标了，通过这两个坐标可以求出GL_PROJECTION 的前两行来，书写如下(**求出了投影矩阵第1,2,4行**):

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

现在怎么求出第3行呢？

找出$z_n$与找出$x_n$和$y_n$不同，因为 $z_e$总是被投影近裁剪面-n上。但是我们需要唯一的Z值进行裁剪和深度测试。另外，我们还能够unproject即反向变换(inverse transform)。因为Z值不依赖于x或者y，因此我们借用w分量来找出 $z_n$ 和 $z_e$ 之间的关系。

因此我们可以这样指定第3行:

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

在眼坐标下We等于1，因此上式变为:

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

我们使用($z_e$, $z_n$)的关系(-n, -1)和 (-f, 1)来求解出系数A,B；

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

细节部分有删节，使用消元法即可求出:

$$A = -\frac{f+n}{f-n} \quad B = -\frac{2fn}{f-n}$$

我们求出了A和B，那么$z_e$和$z_n$关系如下式:

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

最终的投影矩阵如下式：

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

OpenGL Perspective Projection Matrix

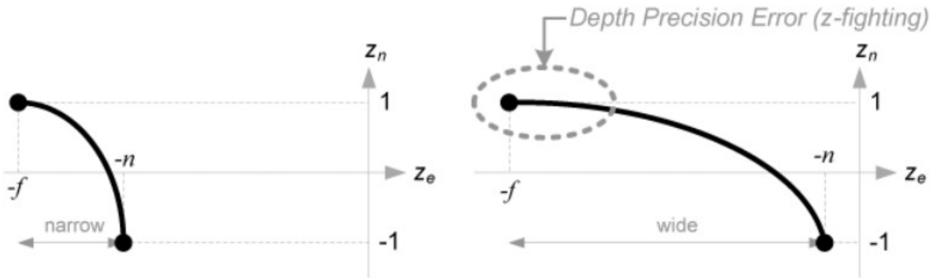
这个公式对应的是一般的视锥，如果视锥是对称的，即r = -l ,t= -b，那么有:

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

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

**z-fighting**

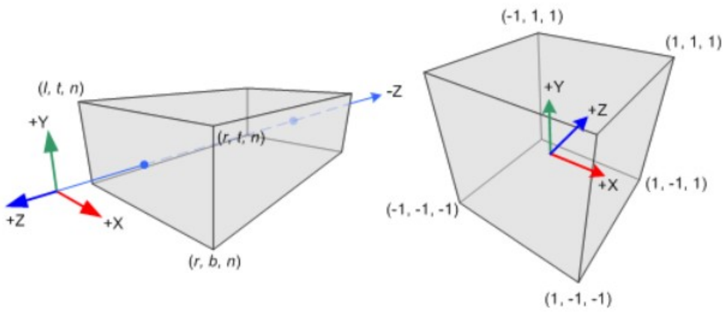
看表示$z_e$和$z_n$关系的式3.这是一个有理函数，并且$z_e$和$z_n$之间不是线性关系。这意味着，在近裁剪面的精度很高，而远裁剪面则很小。如果[-n, -f]范围变得大写，就会引起深度精度问题，即z-fighting。在远裁剪面附近，$z_e$的小变化根本不影响$z_n$值。近裁剪面和远裁剪面之间的n和f举例，应该尽可能小，来减少深度缓存的精度问题。可参考下图来帮助理解。




Comparison of Depth Buffer Precisions

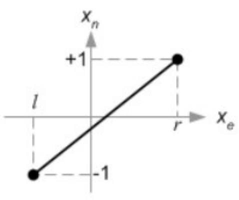
## 3.正交投影

构造正交投影的矩阵简单很多。所有的是眼坐标下$x_e$, $y_e$ 和$z_e$，都被线性的映射到NDC中。我们需要做的就是讲长方体视景体缩放为规范视见体，然后移动到原点。如下图所示：




Orthographic Volume and Normalized Device Coordinates (NDC)

以$x_e$和$x_n$之间映射关系为例，[l,r]=>[-1,1],则可以推导如下:




Mapping from $x_e$ to $x_n$

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

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

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

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

y,z也有类似推导，这里省略，最后得出投影矩阵为:

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

OpenGL Orthographic Projection Matrix

如果视锥是对称的话，即r = -l ,t= -b的话，则可以简化为:

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

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

到这里透视投影和正交投影矩阵推导完毕。



**上一篇：OpenGL学习脚印: OpenGL 坐标变换**

**下一篇：OpenGL学习脚印: 绘制移动三角形**