DOI：10.3969/j.issn.1001-8972.2011.15.057

两种投影变换下深度缓冲值的统一计算

杜剑 罗林 广州大学华软软件学院游戏系

摘 要

游戏开发中经常需要使用两种不同的投影变换，正交投影和透视投影，两种投影变换下各自 Z 坐标的转换会用来在深度缓冲中做先后的判定，在不同变换情况下，通过对两种转换公式的推导得出 Z 坐标变换情况的两种情况的互换，在游戏开发中可以使用统一标尺来度量前后关系。

关键词

透视投影；正交投影；投影变换；D3D；深度缓冲

三维图形编程中，图形渲染管线中的三步变换中投影变换把顶点的三维坐标投影到一个投影平面上，在进行投影变换时，所有的顶点会被投影到近裁剪面上，所有投影后顶点 p′ 的 z 坐标会转换成同一个值，为了渲染时区分渲染对象之间的前后关系，D3D 会使用深度缓冲（一般是 z-buffer）来实现，深度缓冲中存放的，实际就是观察坐标系下 z 坐标在投影区间的一个映射值，在 D3D 使用的左手坐标系下，深度缓冲中的 z 值的范围会被转换到[0，1]之间，而在 OPENGL 使用的是右手坐标系，深度缓冲中的 z 值的范围转换到[-1，1]之间，本文中笔者将其称为投影转换后的 z 值（实际上转换后 z 值是一个统一值，我们这里指的是写入深度缓冲的 z 值），良种开发库下转换原理差不多，我们以 D3D 下的转换为例进行探讨。

在同一种投影变换过程中，无论如何转换，转换后的 z 值大小关系是保持不变的。但在使用 D3D 进行三维游戏开发的过程中，通常会采用两种投影变换方式，正交投影变换和透视投影变换．在实现 UI 界面或者某些特殊标记会使用正交投影，而普通的游戏对象则采用透视投影，但是对于两种投影方式而言，z 坐标的转换过程并不是遵循同样的标准，所以在世界坐标下的 z 坐标并不能完全反映两种不同变换方式下他们的前后关系，而在游戏开发中，那么我们又需要一个转换统一的 z 坐标来实现渲染时的相互遮挡，这样就有必要能够判断不同变换下的 z 坐标大小。

本文我们通过研究两种不同投影变换下 z 坐标的转换过程，得出一个从正交投影相对于透视投影下的 z 坐标对应关系的公式，从而在游戏开发中可以在进行世界坐标设定时，能直接得到其转换后的关系，来更直观的设置游戏对象的先后关系。

两种投影方式中，其中正交投影的投影方式是在一个长方体观察体内进行，它不会根据远近缩放物体，而透视投影则是在一个视锥观察体中进行，会根据物体距离视点的远近缩放物体．我们先来推导出 D3D 渲染下的正交投影变换过程中 Z 坐标的转换计算。下面是一个正交投影变换的示意图：

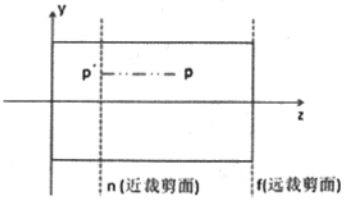

图 1

n 和 f 是正交投影过程中的远近裁减面，也就代表了我们裁剪面距离视点的最小和最大距离，投影之前，顶点 P 的坐标为(x，y，z)，投影之后 P′ 的坐标为(x，y，n)，这样每个顶点的 Z 坐标将进行变换成 n，而对于深度缓冲中保存的 z 值而言，在正交投影变换中，变换后的 Z′ 值是和距离呈线性关系，那么有线性公式为

$z' = az+b$

而变换后 Z 值的范围在[0，1]之内，这样近裁减面变换后为 0，远裁剪面变换后则是 1，则可以得到方程组

$$\begin{cases} an+b=0 \\ af+b=1 \end{cases} \Rightarrow \begin{cases} a=1/(f-n) \\ b=-n(f-n) \end{cases}$$

那么 $z' = (z-n) / (f-n)$

而对于透视投影变换来说，稍微麻烦一点，透视投影变换是在一个视锥体中进行，下面是示意图：

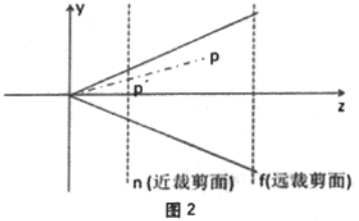

图 2

与正交投影不同，透视投影的过程中，变换后的 z′ 值与视点到平面的距离并非线性关系，在距离近裁剪面比较近时增加 z 值的精度，远时精度则减小。

透视投影变换时，可以得到其公式为 $z' = a*1/(-z)+b$

$$\begin{cases} a*1/(-n)+b=0 \\ a*1/(-f)+b=1 \end{cases} \Rightarrow \begin{cases} a=nf/(f-n) \\ b=f/(f-n) \end{cases}$$

那么 $z' = ((z-n)*f)/((f-n)*z)$

从两种投影变换中 z 的转换关系可以看出，正交投影变换 z 值和投影距离是线性变换，而透视投影则是非线性，从图可以看出在距离近裁剪面更近的地方 z 值分布更多，而在比较远的距离处 z 值就很散，这样就会导致深度缓冲采用 z-Buffer 时，在近摄像机距离会比较准确，而在比较远处则会出现一定范围无法判断的情况，而且当两个裁剪面距离更大时，这样的问题会更严重，所以尽量保持投影裁减距离在 1000 以内，相对能接受，如果深度缓冲采用 w-Buffer 就在距离分布上是均匀的，但是现在显卡基本上都支持 z-buffer 而非 w-buffer，所以游戏编程中，出于效率的考虑，还是采用 z-buffer，而且从图形效果上而言，玩家总是对近距离的境况要求更高。

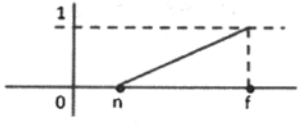

图 3 正交投影矩阵 z 线性变换

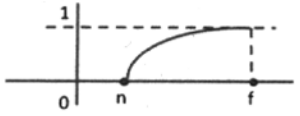

图 4 透视投影矩阵非线性变换

推导出两种投影方式 z 坐标的变换公式后，游戏开发中就能在设定使用正交投影变换的对象时，将其转换成采用透视投影的 z 坐标来设定，如此就可以在游戏中建立一个前后关系明确的 z 轴标尺。

在正交投影变换的游戏对象设定其 zo 值时，可以用透视投影相应的 zp 值来转换。在同等转换后的深度缓冲中的 z 值而言，可以得到下面等式：

$$\frac{(Z^p-n^p)*f^p}{(f^p-n^p)*z^p} = \frac{(Z^o-n^o)}{(f^o-n^o)}$$

在使用 D3D 进行游戏开发时，D3D 把具体的投影变换的细节封装起来了，但实际上在实际编程中要解决一些比较高端的问题，通常需要对变换的过程有深入了解，再探索出这样一个公式，在游戏开发中就不会困扰于不同投影变换对深度缓冲的处理。

参考文献

[1] Andr é LaMothe .3D 游戏编程大师技巧.人民邮电出版社.2005

[2]胡静斯，李霖，李维科.基于 DirectX 的地貌景渲实现机制研究[J].测绘科学.2004.(04)

[3]丁宇明.投影变换与座标变换[J].武汉大学学报(工学版).1981 年 04 期