湖南科技大学课程教案
（章节、专题首页）

授课教师：王志喜　　职称：副教授　　单位：计算机科学与工程学院

课程名称	计算机图形图像技术
章节、专题	三维图形变换
教学目标及基本要求	（1）掌握三维物体的多边形表示和三维基本变换的原理。 （2）掌握三维变换的复合方法和三维坐标变换。 （3）掌握三维对象的观察流程及投影变换和裁剪算法。
教学重点	三维物体的多边形表示，三维变换的复合，投影变换，规范化变换，三维裁剪。
教学难点	三维变换的复合，投影变换，规范化变换。
教学内容与时间分配	（1）三维物体的多边形表示（0.4课时） （2）三维基本变换以及反射与旋转（1课时） （3）三维变换的复合与三维坐标变换（1课时） （4）三维观察流程与观察体的设置（0.8课时） （5）投影变换与规范化变换（1.5课时） （6）三维裁剪（0.8课时） 共计5.5课时。
习　　题	第5.10.1节（基础知识题）。

第5章　三维图形变换

5.1　三维物体的多边形表示

场景中的对象一般用一组多边形面片来描述。如图5-1所示。

- 多面体。多边形表示精确地描述了物体的表面特征。
- 其他物体。表面嵌入到物体生成一个多边形网格逼近。

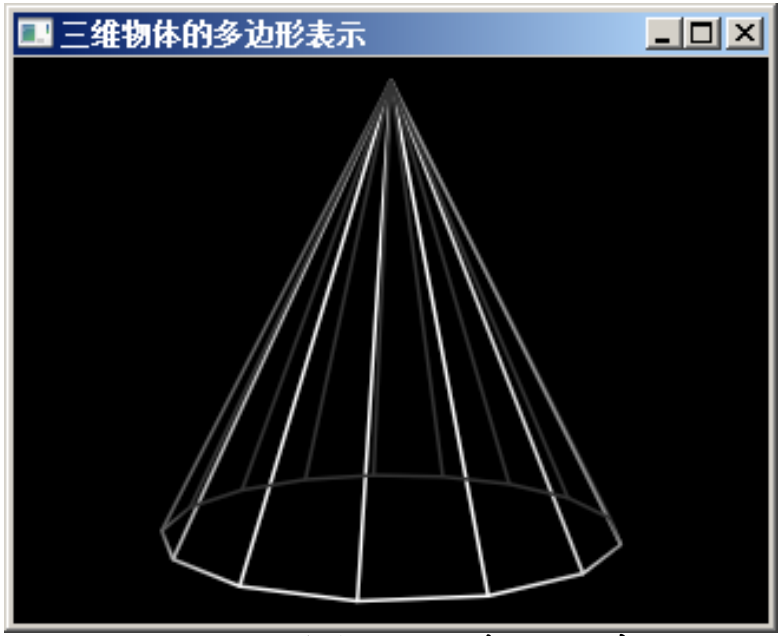


图5-1　三维物体的多边形表示

5.1.1 常用的多边形网格

图形软件包一般使用多边形网格形式描述表面形状，常用的多边形网格有三角形带和四边形网格，如图5-2所示。

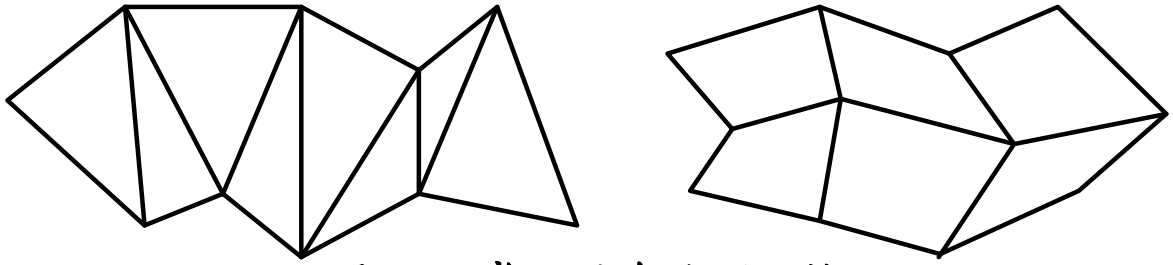


图5-2 常用的多边形网格

5.1.2 平面方程

1. 方程形式

$$Ax + By + Cz + D = 0$$

2. 法向量

$$\overrightarrow{N} = (A, B, C)$$

3. 表面方向

（1）内侧面与外侧面。多边形表面向着对象内部的一侧称为内侧（后向面），可见或朝外的一侧称为外侧（前向面）。

（2）表面方向的描述方法。多边形表面的空间方向可以用所在平面的法向量描述，法向量从平面的内侧指向外侧，也就是从多边形的后方指向前方。如图5-3所示。

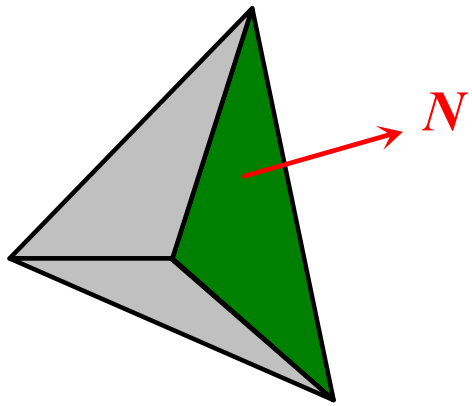


图5-3　多边形的法向量

4. 方程系数

可以使用向量的叉积计算平面的法向量。

已知三点 V_1、V_2 和 V_3 不共线，且按照逆时针方向排列，则

$$\overrightarrow{N}=\overrightarrow{V_1V_2}\times\overrightarrow{V_2V_3}=(V_2-V_1)\times(V_3-V_2)$$

由此可计算出平面法向量 $\overrightarrow{N}=(A,B,C)$。

在计算出平面法向量以后，可以在已知的三点中任意选取一点代入平面方程就可以算出 D。

【注】也可以由 $\overrightarrow{N}=\overrightarrow{V_1V_2}\times\overrightarrow{V_1V_3}$ 计算出 $\overrightarrow{N}=(A,B,C)$。

5. 空间中的点与平面的位置关系

$$\begin{cases}Ax+By+Cz+D=0, & (x,y,z)\ \text{On Plane}\\ Ax+By+Cz+D<0, & (x,y,z)\ \text{Inside Plane (Back)}\\ Ax+By+Cz+D>0, & (x,y,z)\ \text{Outside Plane (Front)}\end{cases}$$

6. 平面方程举例

【问题】已知三个顶点 $V_1(3,4,5)$、$V_2(0,9,8)$ 和 $V_3(3,6,0)$，从里向外以右手系形成逆时针方向。请构造出这三个顶点所确定的平面方程。

【解答】由 $\overrightarrow{V_1V_2}=(-3,5,3)$ 和 $\overrightarrow{V_2V_3}=(3,-3,-8)$ 可得

$$\vec{N}=\overrightarrow{V_1V_2}\times\overrightarrow{V_2V_3}=(-31,-15,-6)$$

设平面方程为 $\vec{N}\bullet\vec{P}+D=0$，即 $-31x-15y-6z+D=0$。将 V_1 代入该方程，可得 $-183+D=0$，从而 $D=183$。所以该平面的方程为

$$-31x-15y-6z+183=0$$

【注】也可以由 $\overrightarrow{V_1V_2}=(-3,5,3)$ 和 $\overrightarrow{V_1V_3}=(0,2,-5)$ 得到

$$\vec{N}=\overrightarrow{V_1V_2}\times\overrightarrow{V_1V_3}=(-31,-15,-6)$$

$$(u_1,u_2,u_3)\times(v_1,v_2,v_3)=\begin{vmatrix}\vec{i}&\vec{j}&\vec{k}\\u_1&u_2&u_3\\v_1&v_2&v_3\end{vmatrix}=\vec{i}\begin{vmatrix}u_2&u_3\\v_2&v_3\end{vmatrix}+\vec{j}\begin{vmatrix}u_3&u_1\\v_3&v_1\end{vmatrix}+\vec{k}\begin{vmatrix}u_1&u_2\\v_1&v_2\end{vmatrix}$$
$$=(u_2v_3-v_2u_3,u_3v_1-v_3u_1,u_1v_2-v_1u_2)$$

5.2 三维基本变换

三维的平移、旋转和缩放是三种基本的三维变换。其他变换通常可以通过这三种基本变换复合得到。

5.2.1 平移

用 $T(t_x, t_y, t_z)$ 表示。

1. 变换方程及其矩阵形式

$$\begin{cases} x' = x + t_x \\ y' = y + t_y \\ z' = z + t_z \end{cases} \Rightarrow \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

2. 逆变换

$$T^{-1}(t_x, t_y, t_z) = T(-t_x, -t_y, -t_z)$$

3. OpenGL中的相关函数

- glTranslatef(tx, ty, tz);
- glTranslated(tx, ty, tz);

5.2.2 缩放

用 $S(s_x, s_y, s_z)$ 表示。

1. 变换方程及其矩阵形式

$$\begin{cases} x' = x \times s_x \\ y' = y \times s_y \\ z' = z \times s_z \end{cases} \Rightarrow \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

2. 逆变换

$$S^{-1}(s_x, s_y, s_z) = S(1/s_x, 1/s_y, 1/s_z)$$

3. OpenGL中的相关函数

- glScalef(sx, sy, sz);
- glScaled(sx, sy, sz);

5.2.3 绕坐标轴旋转

分别用 $R_x(\theta)$、$R_y(\theta)$ 和 $R_z(\theta)$ 表示绕 x 轴、y 轴和 z 轴的旋转。

1. 正旋转方向的确定

使用右手螺旋方法确定，即右手大拇指指向旋转轴的正半轴方向，其余四指弯曲方向就是正旋转方向。如图5-4所示。

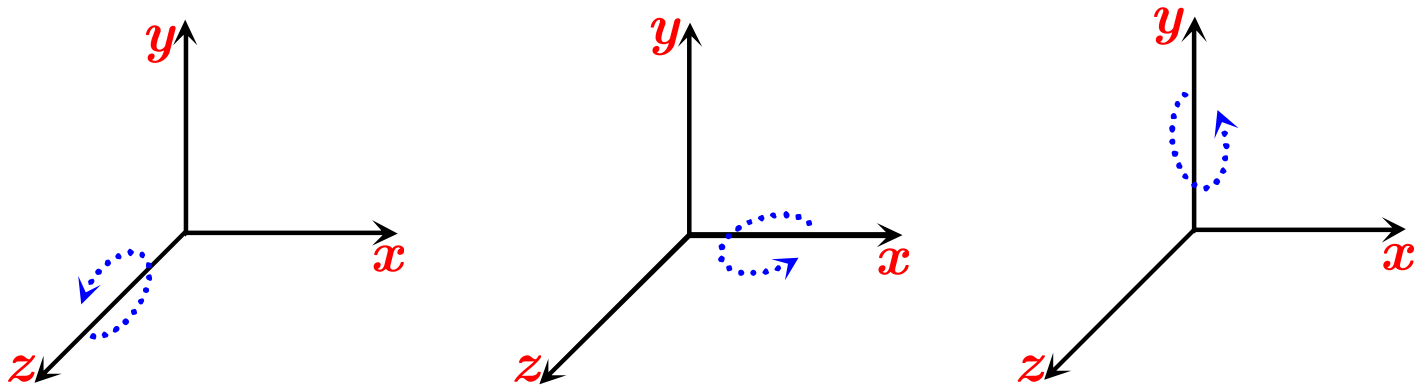


图5-4 正旋转方向的确定

2. 变换方程及其矩阵形式

绕 z 轴旋转的变换方程可由二维的绕原点旋转改装而成。绕 x 轴和 y 轴旋转的变换方程可由绕 z 轴旋转的变换方程通过 x 、 y 和 z 循环替换得到，即 $x \longrightarrow y \longrightarrow z \longrightarrow x$ 。

$$R_z(\theta):\begin{cases}x'=x\cos\theta-y\sin\theta\\ y'=x\sin\theta+y\cos\theta\\ z'=z\end{cases}\Rightarrow\begin{pmatrix}x'\\y'\\z'\\1\end{pmatrix}=\begin{pmatrix}\cos\theta&-\sin\theta&0&0\\ \sin\theta&\cos\theta&0&0\\ 0&0&1&0\\ 0&0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}$$

$$R_x(\theta):\begin{cases}x'=x\\ y'=y\cos\theta-z\sin\theta\\ z'=y\sin\theta+z\cos\theta\end{cases}\Rightarrow\begin{pmatrix}x'\\y'\\z'\\1\end{pmatrix}=\begin{pmatrix}1&0&0&0\\ 0&\cos\theta&-\sin\theta&0\\ 0&\sin\theta&\cos\theta&0\\ 0&0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}$$

$$R_y(\theta):\begin{cases}x'=z\sin\theta+x\cos\theta\\ y'=y\\ z'=z\cos\theta-x\sin\theta\end{cases}\Rightarrow\begin{pmatrix}x'\\y'\\z'\\1\end{pmatrix}=\begin{pmatrix}\cos\theta&0&\sin\theta&0\\ 0&1&0&0\\ -\sin\theta&0&\cos\theta&0\\ 0&0&0&1\end{pmatrix}\begin{pmatrix}x\\y\\z\\1\end{pmatrix}$$

$$\begin{cases}x'=x\cos\theta-y\sin\theta\\ y'=x\sin\theta+y\cos\theta\end{cases}$$

3. 逆变换

$$R^{-1}(\theta) = R(-\theta) = R^T(\theta)$$

4. OpenGL中的相关函数

- glRotatef(th, x, y, z);
- glRotated(th, x, y, z);

表示绕旋转轴 $(0,0,0) \sim (x,y,z)$ 旋转 th 角度。

显然，$(x,y,z) = (1,0,0)$ 表示旋转轴为 x 轴，$(x,y,z) = (0,1,0)$ 表示旋转轴为 y 轴，$(x,y,z) = (0,0,1)$ 表示旋转轴为 z 轴。

【注】变换公式的构造见“三维变换的复合”一节。

5.3 反射和旋转

一些三维变换虽然不是三维基本变换，但是因为这些变换形式简单，又很常用，我们可以将它们当做扩充的三维基本变换。这里只介绍最常用的反射变换和旋转矩阵。

5.3.1 反射

只介绍关于 z 轴的反射和关于 xy 平面的反射。

1. 关于z轴的反射

改变 x 和 y 的符号，而 z 不变。相当于绕 z 轴旋转 180° 或 $S(-1,-1,1)$ 。

$$\begin{cases} x' = -x \\ y' = -y \\ z' = z \end{cases} \qquad R_z(180^\circ) = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2. 关于xy平面的反射

改变 z 的符号，而 x 和 y 不变。相当于 $S(1,1,-1)$ 。

$$\begin{cases} x' = x \\ y' = y \\ z' = -z \end{cases} \qquad RF_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5.3.2 旋转

1. 两两正交的单位向量组变换到坐标轴方向

将两两正交的单位向量组 $\vec{u}=(u_1,u_2,u_3)$、$\vec{v}=(v_1,v_2,v_3)$ 和 $\vec{n}=(n_1,n_2,n_3)$ 分别变换成沿 x 轴、y 轴和 z 轴的单位向量。求该变换的变换矩阵。

2. 变换矩阵的构造

因为 $\vec{u}=(u_1,u_2,u_3)$、$\vec{v}=(v_1,v_2,v_3)$ 和 $\vec{n}=(n_1,n_2,n_3)$ 是两两正交的单位向量组，所以 $|\vec{u}|=|\vec{v}|=|\vec{n}|=1$ 且 $\vec{u}\bullet\vec{v}=\vec{v}\bullet\vec{n}=\vec{n}\bullet\vec{u}=0$。考虑变换矩阵

$$R=\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

因为

$$\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ 1 \end{pmatrix}=\begin{pmatrix} u_1u_1+u_2u_2+u_3u_3 \\ v_1u_1+v_2u_2+v_3u_3 \\ n_1u_1+n_2u_2+n_3u_3 \\ 1 \end{pmatrix}=\begin{pmatrix} |\vec{u}|^2 \\ \vec{v}\bullet\vec{u} \\ \vec{n}\bullet\vec{u} \\ 1 \end{pmatrix}=\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad (u_1,u_2,u_3)\Rightarrow x$$

$$\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ 1 \end{pmatrix} = \begin{pmatrix} u_1v_1 + u_2v_2 + u_3v_3 \\ v_1v_1 + v_2v_2 + v_3v_3 \\ n_1v_1 + n_2v_2 + n_3v_3 \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{u}\bullet\vec{v} \\ |\vec{v}|^2 \\ \vec{n}\bullet\vec{v} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \qquad (v_1, v_2, v_3) \Rightarrow y$$

$$\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ 1 \end{pmatrix} = \begin{pmatrix} u_1n_1 + u_2n_2 + u_3n_3 \\ v_1n_1 + v_2n_2 + v_3n_3 \\ n_1n_1 + n_2n_2 + n_3n_3 \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{u}\bullet\vec{v} \\ \vec{v}\bullet\vec{n} \\ |\vec{n}|^2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \qquad (n_1, n_2, n_3) \Rightarrow z$$

所以该变换的变换矩阵就是 R。

3. OpenGL中的相关函数

- void glMultMatrixd(const GLdouble *m);
- void glMultMatrixf(const GLfloat *m);

变换

$$R=\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

的调用形式为（列主向存储）

```
glMultMatrixd((const double[])
{   u1, v1, n1, 0,
    u2, v2, n2, 0,
    u3, v3, n3, 0,
    0, 0, 0, 1
});
```

4. 旋转矩阵的特性

当 $\vec{u}=\vec{v}\times\vec{n}$ 、 $\vec{v}=\vec{n}\times\vec{u}$ 或 $\vec{n}=\vec{u}\times\vec{v}$ 之一成立时，上述变换矩阵 R 实际上代表一个复合的旋转变换（连续两次关于坐标轴的旋转），称为旋转矩阵。

（1）逆变换。$R^{-1}=R^T$。这是因为

$$\begin{pmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}=\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ 1 \end{pmatrix} \qquad x \Rightarrow (u_1, u_2, u_3)$$

$$\begin{pmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}=\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ 1 \end{pmatrix} \qquad y \Rightarrow (v_1, v_2, v_3)$$

$$\begin{pmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}=\begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ 1 \end{pmatrix} \qquad z \Rightarrow (n_1, n_2, n_3)$$

实际上，旋转变换的逆变换也是一个旋转变换（证明略）。

（2）正交性。行向量 $\vec{u}=(u_1,u_2,u_3)$、 $\vec{v}=(v_1,v_2,v_3)$ 和 $\vec{n}=(n_1,n_2,n_3)$ 是两两正交的单位向量组。列向量 $\vec{u}'=(u_1,v_1,n_1)$， $\vec{v}'=(u_2,v_2,n_2)$ 和 $\vec{n}'=(u_3,v_3,n_3)$ 也是两两正交的单位向量组。因为由 $R^{-1}=R^T$ 可知 $R^T\times R=I$，即

$$R^T\times R=\begin{pmatrix}u_1 & v_1 & n_1 & 0\\ u_2 & v_2 & n_2 & 0\\ u_3 & v_3 & n_3 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}u_1 & u_2 & u_3 & 0\\ v_1 & v_2 & v_3 & 0\\ n_1 & n_2 & n_3 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$=\begin{pmatrix}u_1^2+v_1^2+n_1^2 & u_1u_2+v_1v_2+n_1n_2 & u_1u_3+v_1v_3+n_1n_3 & 0\\ u_2u_1+v_2v_1+n_2n_1 & u_2^2+v_2^2+n_2^2 & u_2u_3+v_2v_3+n_2n_3 & 0\\ u_3u_1+v_3v_1+n_3n_1 & u_3u_2+v_3v_2+n_3n_2 & u_3^2+v_3^2+n_3^2 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$\left(\begin{array}{c|c|c|c}u_1 & u_2 & u_3 & 0\\ v_1 & v_2 & v_3 & 0\\ n_1 & n_2 & n_3 & 0\\ 0 & 0 & 0 & 1\end{array}\right)$$

所以

$$\begin{cases} u_i^2+v_i^2+n_i^2=1,\ \ i=1,2,3 \\ u_iu_j+v_iv_j+n_in_j=0,\ \ 1\le i<j\le 3 \end{cases}$$

即

$$|\vec{u}'|=|\vec{v}'|=|\vec{n}'|=1,\qquad \vec{u}'\bullet\vec{v}'=\vec{v}'\bullet\vec{n}'=\vec{n}'\bullet\vec{u}'=0$$

$$\begin{pmatrix} u_1^2+v_1^2+n_1^2 & u_1u_2+v_1v_2+n_1n_2 & u_1u_3+v_1v_3+n_1n_3 & 0 \\ u_2u_1+v_2v_1+n_2n_1 & u_2^2+v_2^2+n_2^2 & u_2u_3+v_2v_3+n_2n_3 & 0 \\ u_3u_1+v_3v_1+n_3n_1 & u_3u_2+v_3v_2+n_3n_2 & u_3^2+v_3^2+n_3^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

（3）循环外积性。当 R 是旋转矩阵时，R 的行向量 $\vec{u}$、$\vec{v}$ 和 $\vec{n}$ 满足循环外积性，也就是说 $\vec{u}=\vec{v}\times\vec{n}$、$\vec{v}=\vec{n}\times\vec{u}$ 和 $\vec{n}=\vec{u}\times\vec{v}$ 都成立（如图5-5所示），即

$$
\begin{aligned}
(u_1,u_2,u_3)&=(v_2n_3-n_2v_3,v_3n_1-n_3v_1,v_1n_2-n_1v_2)\\
(v_1,v_2,v_3)&=(n_2u_3-u_2n_3,n_3u_1-u_3n_1,n_1u_2-u_1n_2)\\
(n_1,n_2,n_3)&=(u_2v_3-v_2u_3,u_3v_1-v_3u_1,u_1v_2-v_1u_2)
\end{aligned}
$$

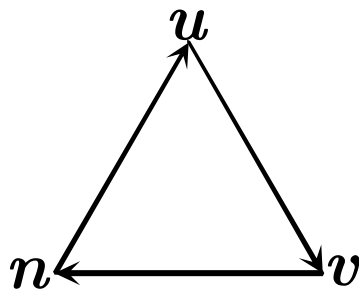


图5-5　循环外积示意图

$$
R=\begin{pmatrix} u_1 & u_2 & u_3 & 0\\ v_1 & v_2 & v_3 & 0\\ n_1 & n_2 & n_3 & 0\\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
\begin{aligned}
(u_1,u_2,u_3)\times(v_1,v_2,v_3)&=\begin{vmatrix} \vec{i} & \vec{j} & \vec{k}\\ u_1 & u_2 & u_3\\ v_1 & v_2 & v_3 \end{vmatrix}=\vec{i}\begin{vmatrix} u_2 & u_3\\ v_2 & v_3 \end{vmatrix}+\vec{j}\begin{vmatrix} u_3 & u_1\\ v_3 & v_1 \end{vmatrix}+\vec{k}\begin{vmatrix} u_1 & u_2\\ v_1 & v_2 \end{vmatrix}\\
&=(u_2v_3-v_2u_3,u_3v_1-v_3u_1,u_1v_2-v_1u_2)
\end{aligned}
$$

5.3.3 3种变换矩阵的对比

平移、旋转、缩放的对比。

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5.4 三维变换的复合与三维坐标变换

首先介绍三种通用的三维变换，然后介绍三维坐标变换。

5.4.1 通用三维旋转

1. 问题

已知 $P_0=\{x_0,y_0,z_0\}$， $P_1=\{x_1,y_1,z_1\}$，旋转轴为 P_0P_1，旋转角为 θ。求该旋转变换的变换矩阵。

2. 变换矩阵的构造

（1）使旋转轴通过坐标原点：$T=T(-x_0,-y_0,-z_0)$。

（2）使旋转轴与某坐标轴（通常是 z 轴）重合：R。

将 $\overline{P_0P_1}$ 单位化，得 $\vec{n}=\overline{P_0P_1}/|\overline{P_0P_1}|=(n_1,n_2,n_3)$。设 $\overline{V}$ 是一个与 $\vec{n}$ 不平行的向量，令 $\vec{u}=\overline{V}\times\vec{n}/|\overline{V}\times\vec{n}|=(u_1,u_2,u_3)$，　　$\vec{v}=\vec{n}\times\vec{u}=(v_1,v_2,v_3)$。由此可得

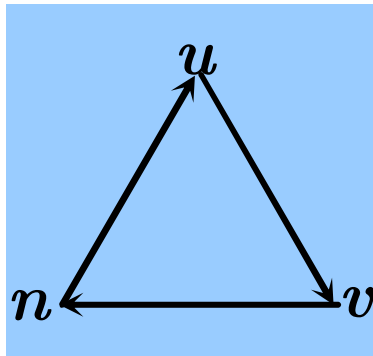
$$R=\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

（3）绕坐标轴（z 轴）完成指定的旋转：$R_z(\theta)$。

（4）使旋转轴回到原来的方向：R^{-1}。

（5）使旋转轴回到原来的位置：T^{-1}。

完整变换为



$$M = T^{-1} \times R^{-1} \times R_z(\theta) \times R \times T$$

$$= \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times$$

$$\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} & b_x \\ a_{yx} & a_{yy} & a_{yz} & b_y \\ a_{zx} & a_{zy} & a_{zz} & b_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

其中

$$\begin{cases} a_{xx} = n_1^2(1-\cos\theta)+\cos\theta \\ a_{xy} = n_1 n_2(1-\cos\theta)-n_3\sin\theta \\ a_{xz} = n_1 n_3(1-\cos\theta)+n_2\sin\theta \\ b_x = +(1-a_{xx})x_0 - a_{xy}y_0 - a_{xz}z_0 \end{cases}$$

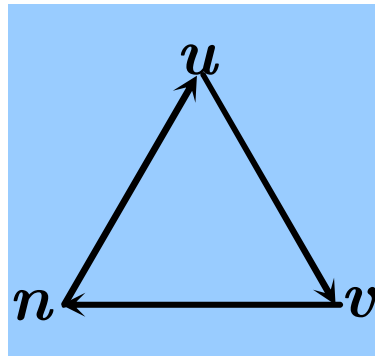
$$\begin{cases} a_{yx} = n_2 n_1(1-\cos\theta)+n_3\sin\theta \\ a_{yy} = (1-\cos\theta)n_2^2+\cos\theta \\ a_{yz} = n_2 n_3(1-\cos\theta)-n_1\sin\theta \\ b_y = -a_{yx}x_0 + (1-a_{yy})y_0 - a_{yz}z_0 \end{cases}$$

$$\begin{cases} a_{zx} = n_3 n_1(1-\cos\theta)-n_2\sin\theta \\ a_{zy} = n_3 n_2(1-\cos\theta)+n_1\sin\theta \\ a_{zz} = n_3^2(1-\cos\theta)+\cos\theta \\ b_z = -a_{zx}x_0 - a_{zy}y_0 + (1-a_{zz})z_0 \end{cases}$$

可以看到，最终结果与 $\vec{u}$ 和 $\vec{v}$ 的取值无关。

【注】根据 $\vec{u}$ 、 $\vec{v}$ 和 $\vec{n}$ 的构造可知， $\vec{u}$ 、 $\vec{v}$ 和 $\vec{n}$ 两两正交且 $\vec{v}=\vec{n}\times\vec{u}$ ，所以 R 是一个旋转矩阵。由旋转矩阵的下列特性可以得到上述结果。

- $n_1=u_2v_3-v_2u_3$ ， $n_2=u_3v_1-v_3u_1$ ， $n_3=u_1v_2-v_1u_2$ （循环外积性）。
- $u_i^2+v_i^2+n_i^2=1$ ， $i=1,2,3$ （列向量为单位向量）。
- $u_iu_j+v_iv_j+n_in_j=0$ ， $1\le i<j\le 3$ （列向量两两正交）。



3. 程序片段

对于该例，OpenGL中对应的程序片段如下（使用double类型参数）。

```
glTranslated(x0, y0, z0); // (5) 使旋转轴回到原来的位置
glMultMatrixd((const double[]) // (4) 使旋转轴回到原来的方向
{   u1, u2, u3, 0,
    v1, v2, v3, 0,
    n1, n2, n3, 0,
    0, 0, 0, 1
});
glRotated(th, 0, 0, 1); // (3) 绕坐标轴完成指定的旋转
glMultMatrixd((const double[]) // (2) 使旋转轴与某坐标轴重合
{   u1, v1, n1, 0,
    u2, v2, n2, 0,
    u3, v3, n3, 0,
    0, 0, 0, 1
});
glTranslated(-x0, -y0, -z0); // (1) 使旋转轴通过坐标原点
```

5.4.2 通用反射轴反射♫

1. 问题

已知反射轴为 P_0P_1，其中 $P_0=(x_0,y_0,z_0)$，$P_1=(x_1,y_1,z_1)$。求该反射变换的变换矩阵。

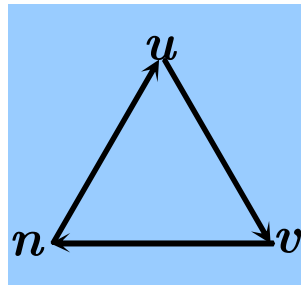
2. 变换矩阵的构造

（1）使反射轴通过原点： $T=T(-x_0,-y_0,-z_0)$。

（2）使反射轴与 z 轴重合： R。

将 $\overline{P_0P_1}$ 单位化，得 $\vec{n}=\overline{P_0P_1}\,/\,|\overline{P_0P_1}|=(n_1,n_2,n_3)$。选取一个与 $\vec{n}$ 不平行的向量 $\overline{V}$，令 $\vec{u}=\overline{V}\times\vec{n}\,/\,|\overline{V}\times\vec{n}|=(u_1,u_2,u_3)$，$\vec{v}=\vec{n}\times\vec{u}=(v_1,v_2,v_3)$，则

$$R=\begin{pmatrix} u_1 & u_2 & u_3 & 0\\ v_1 & v_2 & v_3 & 0\\ n_1 & n_2 & n_3 & 0\\ 0 & 0 & 0 & 1 \end{pmatrix}$$



（3）关于 z 轴反射： $R_z(180°)$。

（4）使反射轴回到原来的方向： R^{-1}。

（5）使反射轴回到原来的位置： T^{-1}。

完整变换为

$$(u_1,u_2,u_3)\times(v_1,v_2,v_3)=\begin{vmatrix}\vec{i} & \vec{j} & \vec{k}\\ u_1 & u_2 & u_3\\ v_1 & v_2 & v_3\end{vmatrix}=\vec{i}\begin{vmatrix}u_2 & u_3\\ v_2 & v_3\end{vmatrix}+\vec{j}\begin{vmatrix}u_3 & u_1\\ v_3 & v_1\end{vmatrix}+\vec{k}\begin{vmatrix}u_1 & u_2\\ v_1 & v_2\end{vmatrix}$$
$$=(u_2v_3-v_2u_3,u_3v_1-v_3u_1,u_1v_2-v_1u_2)$$

$$M = T^{-1} \times R^{-1} \times R_z(180°) \times R \times T$$

$$= \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times$$

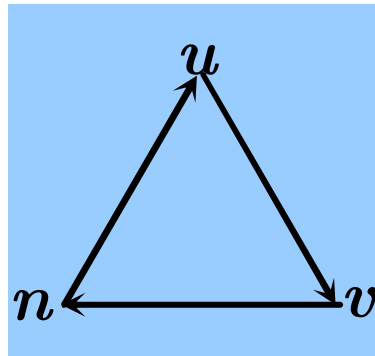
$$\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 2n_1^2 - 1 & 2n_1n_2 & 2n_1n_3 & +(2-2n_1^2)x_0 - 2n_1n_2y_0 - 2n_1n_3z_0 \\ 2n_2n_1 & 2n_2^2 - 1 & 2n_2n_3 & -2n_2n_1x_0 + (2-2n_2^2)y_0 - 2n_2n_3z_0 \\ 2n_3n_1 & 2n_3n_2 & 2n_3^2 - 1 & -2n_3n_1x_0 - 2n_3n_2y_0 + (2-2n_3^2)z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

可以看到，最终结果与 $\vec{u}$ 和 $\vec{v}$ 的取值无关。

【注】根据 $\vec{u}$ 、 $\vec{v}$ 和 $\vec{n}$ 的构造可知， $\vec{u}$ 、 $\vec{v}$ 和 $\vec{n}$ 两两正交且 $\vec{v}=\vec{n}\times\vec{u}$ ，所以 R 是一个旋转矩阵。由旋转矩阵的下列特性可以得到上述结果。

- $n_1=u_2v_3-v_2u_3$ ， $n_2=u_3v_1-v_3u_1$ ， $n_3=u_1v_2-v_1u_2$ （循环外积性）。
- $u_i^2+v_i^2+n_i^2=1$ ， $i=1,2,3$ （列向量为单位向量）。
- $u_iu_j+v_iv_j+n_in_j=0$ ， $1\le i<j\le 3$ （列向量两两正交）。



3. 程序片段

对于该例，OpenGL中对应的程序片段如下（使用double类型参数）。

```
glTranslated(x0, y0, z0); // (5) 使反射轴回到原来的位置
glMultMatrixd((const double[]) // (4) 使反射轴回到原来的方向
{   u1, u2, u3, 0,
    v1, v2, v3, 0,
    n1, n2, n3, 0,
    0, 0, 0, 1
});
glScaled( -1, -1, 1); // (3) 关于坐标轴完成指定的反射
glMultMatrixd((const double[]) // (2) 使反射轴与某坐标轴重合
{   u1, v1, n1, 0,
    u2, v2, n2, 0,
    u3, v3, n3, 0,
    0, 0, 0, 1
});
glTranslated(-x0, -y0, -z0); // (1) 使反射轴通过坐标原点
```

5.4.3 通用反射面反射

1. 问题

已知反射面的方程为 $ax+by+cz+d=0$，且点 (x_0,y_0,z_0) 在反射面上。求该反射变换的变换矩阵。

2. 变换矩阵的构造

（1）使反射面通过原点：$T=T(-x_0,-y_0,-z_0)$。

（2）使反射面与 xy 平面重合：R。

将反射面的法向量 $\overline{N}=(a,b,c)$ 单位化，得 $\vec{n}=\overline{N}\,/\,|\overline{N}|=(n_1,n_2,n_3)$。选取一个与 $\vec{n}$ 不平行的向量 $\overline{V}$，令 $\vec{u}=\overline{V}\times\vec{n}\,/\,|\overline{V}\times\vec{n}|=(u_1,u_2,u_3)$，$\vec{v}=\vec{n}\times\vec{u}=(v_1,v_2,v_3)$，则

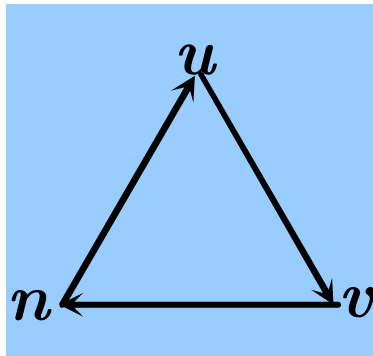
$$R=\begin{pmatrix} u_1 & u_2 & u_3 & 0\\ v_1 & v_2 & v_3 & 0\\ n_1 & n_2 & n_3 & 0\\ 0 & 0 & 0 & 1 \end{pmatrix}$$

（3）关于 xy 面反射：RF_z。

（4）使反射面回到原来的方向：R^{-1}。

（5）使反射面回到原来的位置：T^{-1}。

完整变换为



$$M = T^{-1} \times R^{-1} \times RF_z \times R \times T$$

$$= \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times$$

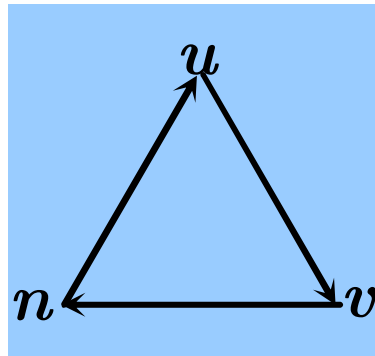
$$\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1-2n_1^2 & -2n_1n_2 & -2n_1n_3 & 2n_1^2x_0 + 2n_1n_2y_0 + 2n_1n_3z_0 \\ -2n_2n_1 & 1-2n_2^2 & -2n_2n_3 & 2n_2n_1x_0 + 2n_2^2y_0 + 2n_2n_3z_0 \\ -2n_3n_1 & -2n_3n_2 & 1-2n_3^2 & 2n_3n_1x_0 + 2n_3n_2y_0 + 2n_3^2z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

可以看到，最终结果与 $\vec{u}$ 和 $\vec{v}$ 的取值无关。

【注】根据 $\vec{u}$ 、 $\vec{v}$ 和 $\vec{n}$ 的构造可知， $\vec{u}$ 、 $\vec{v}$ 和 $\vec{n}$ 两两正交且 $\vec{v}=\vec{n}\times\vec{u}$ ，所以 R 是一个旋转矩阵。由旋转矩阵的下列特性可以得到上述结果。

- $n_1=u_2v_3-v_2u_3$ ， $n_2=u_3v_1-v_3u_1$ ， $n_3=u_1v_2-v_1u_2$ （循环外积性）。
- $u_i^2+v_i^2+n_i^2=1$ ， $i=1,2,3$ （列向量为单位向量）。
- $u_iu_j+v_iv_j+n_in_j=0$ ， $1\le i<j\le 3$ （列向量两两正交）。



3. 程序片段

对于该例，OpenGL中对应的程序片段如下（使用double类型参数）。

```
glTranslated(x0, y0, z0); // (5) 使反射面回到原来的位置
glMultMatrixd((const double[]) // (4) 使反射面回到原来的方向
{   u1, u2, u3, 0,
    v1, v2, v3, 0,
    n1, n2, n3, 0,
    0, 0, 0, 1
});
glScaled(1, 1, -1); // (3) 关于坐标平面完成指定的反射
glMultMatrixd((const double[]) // (2) 使反射面与某坐标平面重合
{   u1, v1, n1, 0,
    u2, v2, n2, 0,
    u3, v3, n3, 0,
    0, 0, 0, 1
});
glTranslated(-x0, -y0, -z0); // (1) 使反射面通过坐标原点
```

5.4.4 三维坐标变换

1. 已知条件

（1）新坐标系的原点：$P_0(x_0, y_0, z_0)$

（2）新坐标轴的单位向量：新 x 方向为 $\vec{u}=(u_1, u_2, u_3)$，新 y 方向为 $\vec{v}=(v_1, v_2, v_3)$，新 z 方向为 $\vec{n}=(n_1, n_2, n_3)$。

2. 变换矩阵的构造

（1）使新坐标系的原点与旧坐标系的原点重合： $T = T(-x_0, -y_0, -z_0)$

（2）使新坐标系的坐标轴与旧坐标系的坐标轴重合： R

$$R = \begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

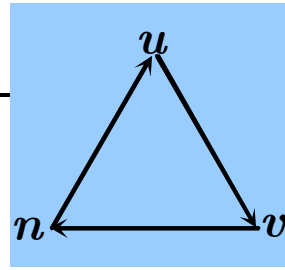
完整变换为

$$
\begin{aligned}
M &= RT \\
&= \begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} u_1 & u_2 & u_3 & -u_1x_0 - u_2y_0 - u_3z_0 \\ v_1 & v_2 & v_3 & -v_1x_0 - v_2y_0 - v_3z_0 \\ n_1 & n_2 & n_3 & -n_1x_0 - n_2y_0 - n_3z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

3. 举例

【问题】已知在原坐标系中某个平面的方程为 $7x-24y+10=0$，试求变换矩阵 M，使该平面方程在新坐标系下变成 $z=0$。其中，新坐标系的 y 方向为 $(24,7,0)$，且新坐标系的原点 $(2,1,0)$ 在该平面上[①]。

① 该平面可理解为照相机的镜头平面，新原点是照相机镜头平面的中心，新 y 方向是照相机的顶部方向。



【解答】

平面法向量在原坐标系中的坐标为 $(7,-24,0)$，而在新坐标系中的坐标为 $(0,0,1)$，所以 $(7,-24,0)$ 是新坐标系的 z 方向。由新坐标系的 y 和 z 方向可计算出新坐标系的 x 方向。

（1）使新原点与旧原点重合：$T=T(-2,-1,0)$

（2）使新坐标轴与旧坐标轴重合：R

新 y 方向为 $\overline{V}=(24,7,0)$，新 z 方向为 $\overline{N}=(7,-24,0)$，新 x 方向为 $\overline{U}=\overline{V}\times\overline{N}=(0,0,-625)$。将 $\overline{U}$、$\overline{V}$、$\overline{N}$ 单位化，得 $\vec{u}=\overline{U}/|\overline{U}|=(0,0,-1)$，$\vec{v}=\overline{V}/|\overline{V}|=(0.96,0.28,0)$，$\vec{n}=\overline{N}/|\overline{N}|=(0.28,-0.96,0)$，从而

$$R=\begin{pmatrix} 0 & 0 & -1 & 0 \\ 0.96 & 0.28 & 0 & 0 \\ 0.28 & -0.96 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

完整变换为

$$
\begin{aligned}
M &= RT \\
&= \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0.96 & 0.28 & 0 & 0 \\ 0.28 & -0.96 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0.96 & 0.28 & 0 & -2.2 \\ 0.28 & -0.96 & 0 & 0.4 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

5.5 三维观察流程和三维观察变换

5.5.1 三维观察流水线

如图5-6所示。

（1）模型变换：$MC \rightarrow WC$。

（2）观察变换：$WC \rightarrow VC$。

（3）投影变换：$VC \rightarrow PC$。

（4）规范化变换：$PC \rightarrow NC$。

（5）工作站变换：$NC \rightarrow DC$。

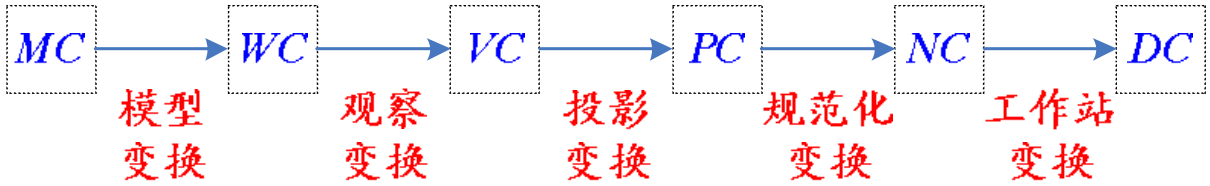


图5-6 三维观察流水线

5.5.2 模型变换

1. 已知条件

使用模型坐标系建立物体模型，建立世界坐标系以后，模型坐标系的原点位于世界坐标系的 $P_0(x_0, y_0, z_0)$ 位置，模型坐标系的 x 轴、y 轴和 z 轴在世界坐标系中的的单位向量分别为 $\vec{u}=(u_1, u_2, u_3)$、$\vec{v}=(v_1, v_2, v_3)$ 和 $\vec{n}=(n_1, n_2, n_3)$。

2. 变换矩阵的构造

首先构造从世界坐标系到模型坐标系的变换，然后计算该变换的逆变换。

（1）使模型坐标系的原点与世界坐标系的原点重合：$T=T(-x_0, -y_0, -z_0)$

（2）使模型坐标系的坐标轴与世界坐标系的坐标轴重合：R

$$R=\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

（3）计算上述2个变换复合以后的逆变换。

完整变换为

$$
\begin{aligned}
M &= (RT)^{-1} = T^{-1}R^{-1} \\
&= \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \\
&= \begin{pmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 & v_1 & n_1 & 0 \\ u_2 & v_2 & n_2 & 0 \\ u_3 & v_3 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} u_1 & v_1 & n_1 & x_0 \\ u_2 & v_2 & n_2 & y_0 \\ u_3 & v_3 & n_3 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

5.5.3 三维视图的生成过程

（1）建立观察坐标系 $x_v y_v z_v$（也称为投影坐标系）。

（2）建立垂直于 z_v 轴的观察面（也称为投影面）。

（3）将景物中的世界坐标转换为观察坐标。

（4）观察坐标投影到观察面。

5.5.4 三维观察坐标系

1. 建立观察坐标系

（1）挑选观察参考点。在世界坐标系中挑选一个点作为观察参考点，用作观察坐标系的原点。

（2）指定观察面的法向量。指定观察面的法向量 $\overline{N}$，由 $\overline{N}$ 确定 z_v 轴的正方向和观察面的方向。

【注】有些教材观察坐标系的使用前后不一致，本书统一使用右手坐标系，沿负 z_v 轴观察，即负 z_v 方向是远离观察者的方向。

（3）指定观察向上向量。指定一个观察向上向量 $\overline{V}$，用 $\overline{V}$ 来建立 y_v 轴的正方向。可以选定任何合适的方向作为观察向上向量 $\overline{V}$，只要 $\overline{V}$ 与 $\overline{N}$ 不平行。

（4）确定 x_v 轴的正方向。用 $\overline{U}=\overline{V}\times\overline{N}$ 定义 x_v 轴的正方向。

（5）确定 y_v 轴的正方向。用 $\overline{V}=\overline{N}\times\overline{U}$ 定义 y_v 轴的正方向。

2. 建立观察面

- 沿 z_v 轴指定一个离原点的距离来选择观察面的位置，如图5-7所示。
- 由于观察面平行于 $x_v y_v$ 平面，所以物体到观察面的投影与在输出设备上显示的场景视图一致。

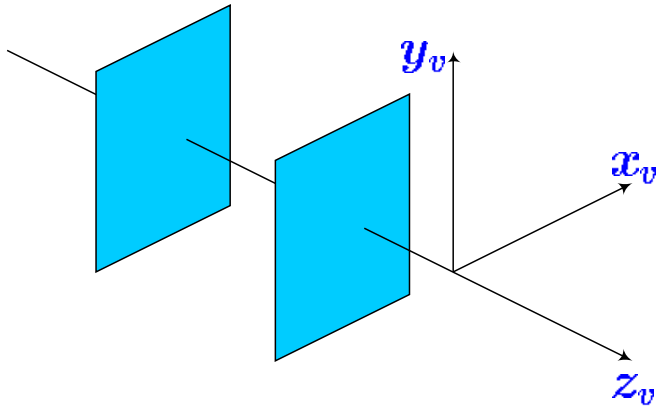


图5-7　建立观察面

5.5.5 观察变换

1. 已知条件

观察参考点为 $P_0=(x_0,y_0,z_0)$，观察面的法向量为 $\overline{N}$，观察向上向量为 $\overline{V}$。

2. 变换矩阵的构造

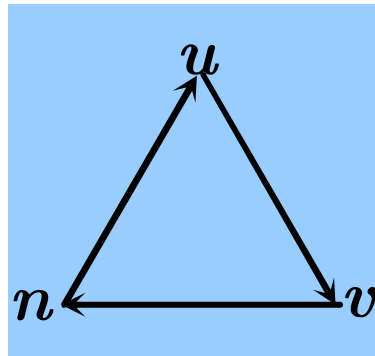
（1）将观察参考点平移到世界坐标原点：$T=T(-x_0,-y_0,-z_0)$

（2）使 x_v,y_v,z_v 轴与 x_w,y_w,z_w 轴重合：R

令 $\overline{U}=\overline{V}\times\overline{N}$，$\overline{V}=\overline{N}\times\overline{U}$。将 $\overline{U}$、$\overline{V}$、$\overline{N}$ 单位化，得 $\vec{u}=\overline{U}/|\overline{U}|=(u_1,u_2,u_3)$，$\vec{v}=\overline{V}/|\overline{V}|=(v_1,v_2,v_3)$，$\vec{n}=\overline{N}/|\overline{N}|=(n_1,n_2,n_3)$，从而

$$R=\begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

完整变换为



$$
\begin{aligned}
M &= RT \\
&= \begin{pmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} u_1 & u_2 & u_3 & -u_1x_0 - u_2y_0 - u_3z_0 \\ v_1 & v_2 & v_3 & -v_1x_0 - v_2y_0 - v_3z_0 \\ n_1 & n_2 & n_3 & -n_1x_0 - n_2y_0 - n_3z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

5.6 投影的类型与观察体的设置

5.6.1 投影的分类

1. 分类图

如图5-8所示。

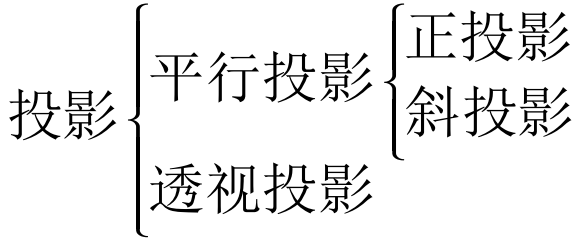


图5-8 投影的分类

2. 平行投影

如图5-9所示。

- 物体位置沿平行线变换到观察面。
- 投影视图由投影线与观察面的交点构成。
- 保持物体的有关比例不变。
- 没有给出三维物体外表的真实感表示。

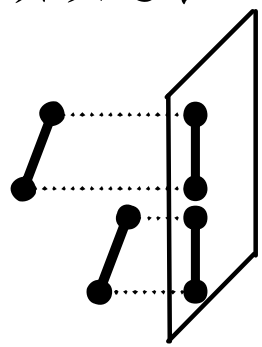
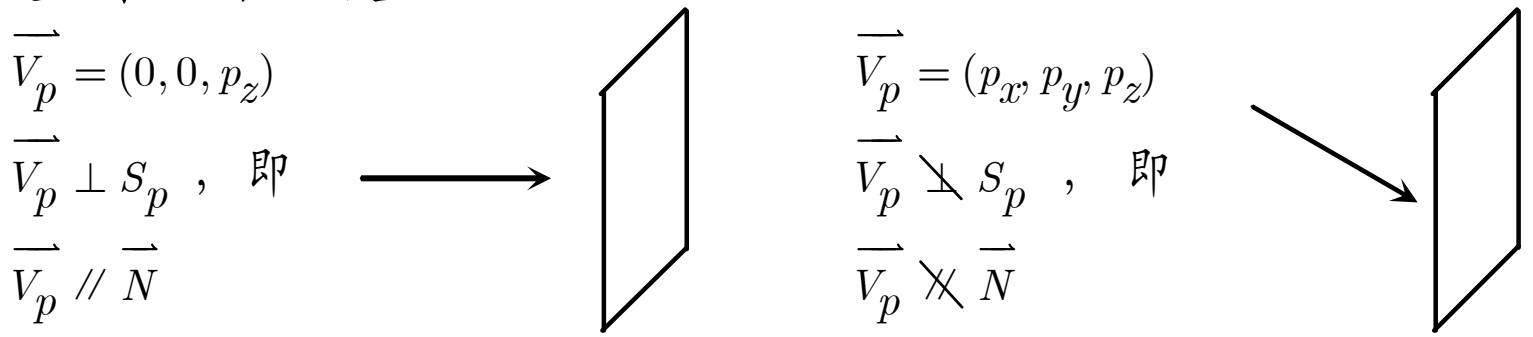


图5-9　平行投影

3. 正投影和斜投影

（1）平行投影的投影向量。定义投影线方向。

（2）两类平行投影的比较。如图5-10所示，其中，$\overrightarrow{V_p}$ 是投影向量，S_p 是观察面，$\vec{N}$ 是观察面的法向量。



[1] 正投影　　[2] 斜投影

图5-10　两类平行投影的比较

4. 透视投影

如图5-11所示。

- 物体位置沿收敛于投影中心的直线变换到观察面。
- 投影视图由投影线与观察面的交点构成。
- 不能保持物体的有关比例不变。
- 距离投影面较远的物体的投影视图比距离投影面较近的物体的投影视图要小一些。
- 给出了三维物体外表的真实感表示。

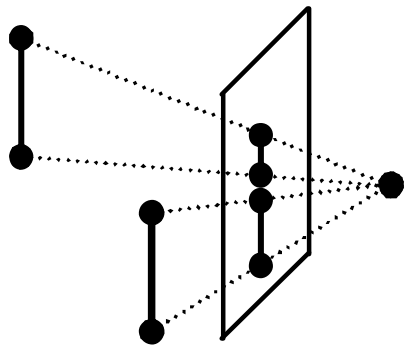


图5-11　透视投影

5.6.2 用投影窗口设置观察体

1. 指定窗口边界

在观察面上定义投影窗口（就是裁剪窗口），只需指定窗口边界，如图5-12所示。

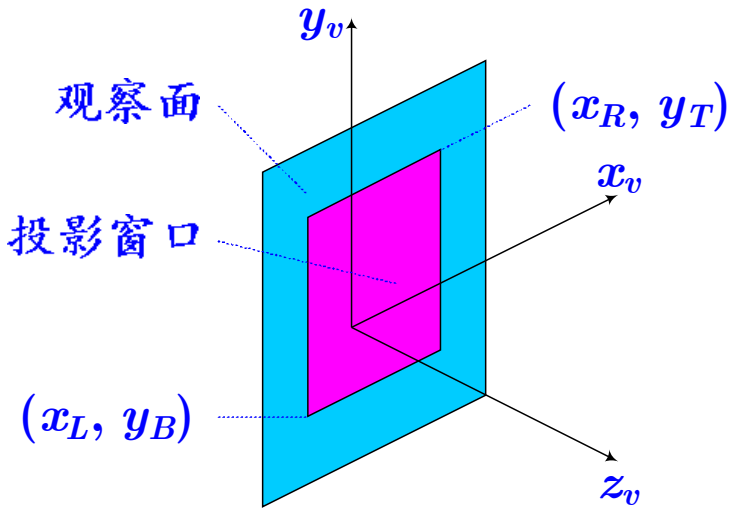
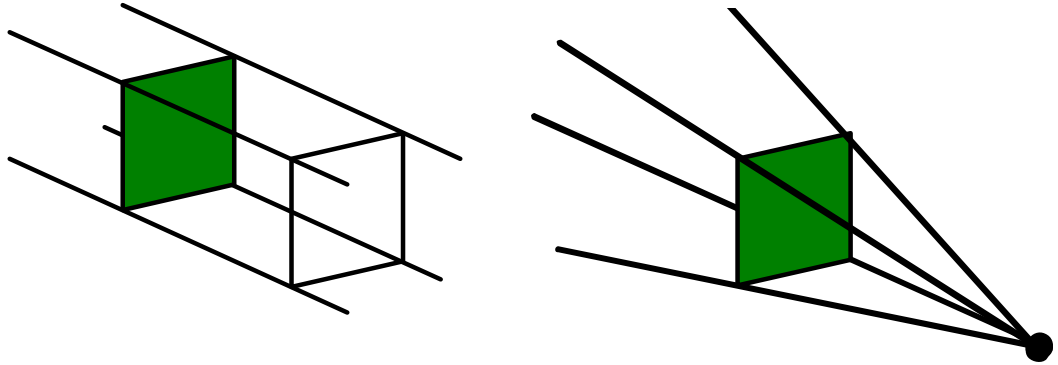


图5-12 窗口边界

2. 观察体的性质

如图5-13所示。

- 大小。依赖于投影窗口的大小。
- 形状。依赖于投影类型。
  - 平行投影。观察体的4个侧面形成无限管道，如图5-13[1]所示。
  - 透视投影。观察体是顶点在投影中心的无限棱锥，如图5-13[2]所示。
- 侧面。通过投影窗口边界的平面。



[1] 平行投影　　[2] 透视投影

图5-13　两类投影的观察体

5.6.3 有限观察体

在 z_v 方向限制容量。

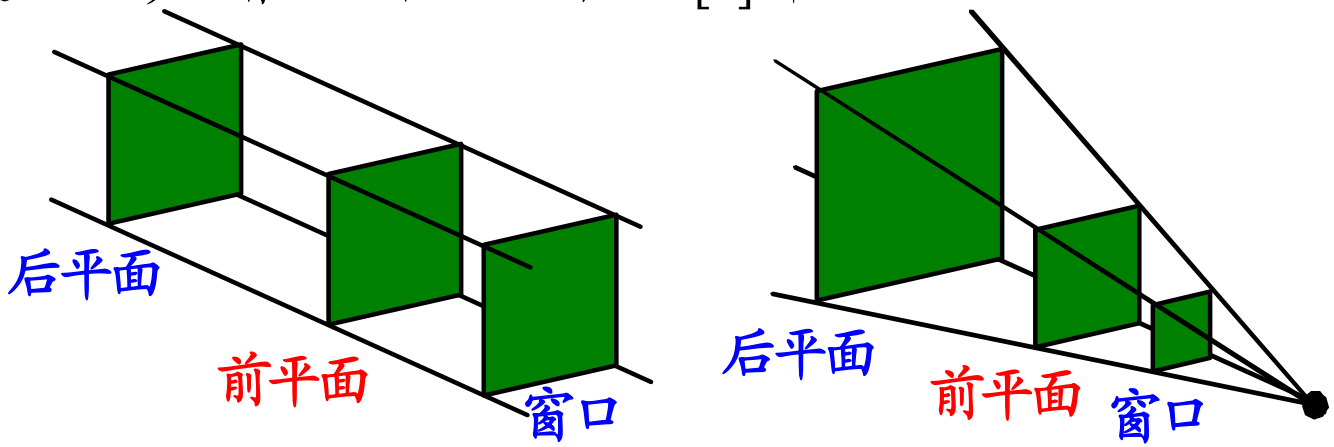
1. 实现方法

在投影中心同一侧附加两个边界平面。其中，离投影中心较近的称为前平面（近平面），离投影中心较远的称为后平面（远平面）。

2. 有限观察体的形状

如图5-14所示，有限观察体共有6个面。

- 平行投影。有限的平行管道观察体，如图5-14[1]所示。
  - 正平行投影。有限的矩形管道。
  - 斜平行投影。有限的平行四边形管道。
- 透视投影。有限棱台，如图5-14[2]所示。



[1] 平行管道观察体　　[2] 棱台观察体

图5-14　观察体形状

5.6.4 投影以后的观察体

投影以后，观察体变为矩形管道（长方体，如图5-15所示）。

【注】投影以后，需要保留原 z 坐标的有关信息（通常保持 z_F 和 z_N 不变）作为深度提示。

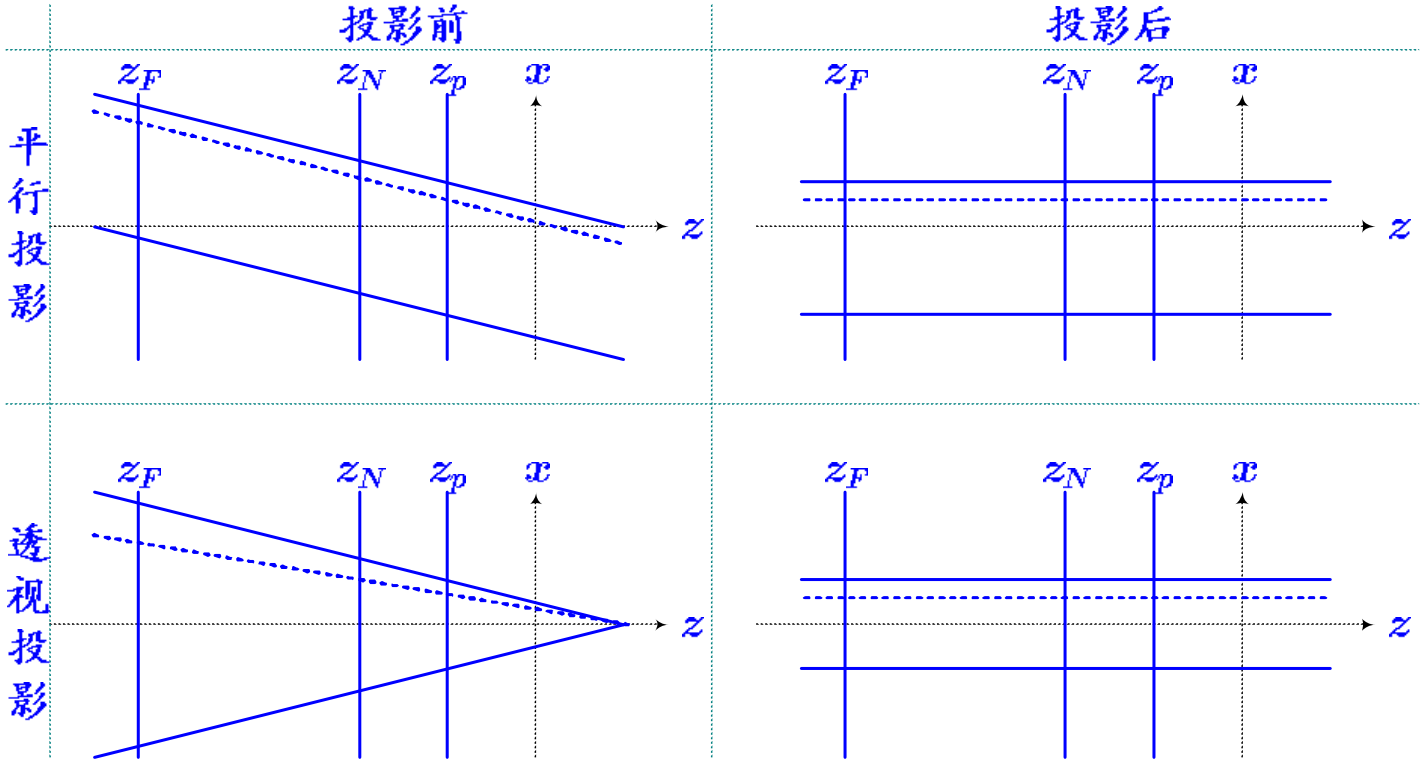


图5-15 投影以后的观察体

5.7 投影变换

5.7.1 平行投影变换

1. 已知条件

投影向量为 $\overline{V_p}=(p_x,p_y,p_z)$，投影面为 $z_v=z_p$。

2. 变换方程

如图5-16所示，由 $(x'-x, y'-y, z_p-z)=u(p_x, p_y, p_z)$ 可得

$$\begin{cases} x'=x+up_x \\ y'=y+up_y \\ z_p=z+up_z \end{cases}$$

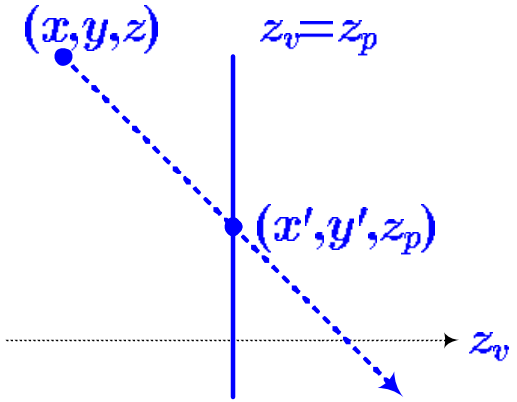


图5-16　平行投影变换方程推导

由 $z_p = z + up_z$ 可得 $u = (z_p - z) / p_z$。由此可得平行投影的变换方程为

$$\begin{cases} x' = x + \dfrac{p_x}{p_z}(z_p - z) = x - \dfrac{p_x}{p_z} z + \dfrac{p_x}{p_z} z_p \\ y' = y + \dfrac{p_y}{p_z}(z_p - z) = y - \dfrac{p_y}{p_z} z + \dfrac{p_y}{p_z} z_p \end{cases}$$

将原始 z 坐标作为深度信息保存。

$$\begin{cases} x' = x + up_x \\ y' = y + up_y \\ z_p = z + up_z \end{cases}$$

3. 变换矩阵

由平行投影的变换方程

$$\begin{cases} x' = x - \dfrac{p_x}{p_z} z + \dfrac{p_x}{p_z} z_p \\ y' = y - \dfrac{p_y}{p_z} z + \dfrac{p_y}{p_z} z_p \end{cases}$$

可得平行投影的变换矩阵为

$$M_{\text{para}} = \begin{pmatrix} 1 & 0 & -\dfrac{p_x}{p_z} & \dfrac{p_x}{p_z} z_p \\ 0 & 1 & -\dfrac{p_y}{p_z} & \dfrac{p_y}{p_z} z_p \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. 正投影变换

对于正投影，因为 $p_x = p_y = 0$，所以变换矩阵为

$$M_{\text{ortho}} = \begin{pmatrix} 1 & 0 & -\dfrac{p_x}{p_z} & \dfrac{p_x}{p_z} z_p \\ 0 & 1 & -\dfrac{p_y}{p_z} & \dfrac{p_y}{p_z} z_p \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5.7.2 透视投影变换

可以假定投影中心为观察坐标原点。若投影中心不是观察坐标原点，则可以先对场景进行整体平移。更加方便的是，直接将观察坐标原点设置为投影中心（在实际工作中一般都是这样做的）。后续章节只考虑这种情况。

1. 已知条件

在观察坐标系中，投影中心为 $R(0,0,0)$，观察面为 $z_v = z_p$。

2. 变换方程

如图5-17所示，由 $(x'-x, y'-y, z_p-z) = u(0-x, 0-y, 0-z)$ 可得

$$\begin{cases} x' = x - xu \\ y' = y - yu \\ z_p = z - zu \end{cases}$$

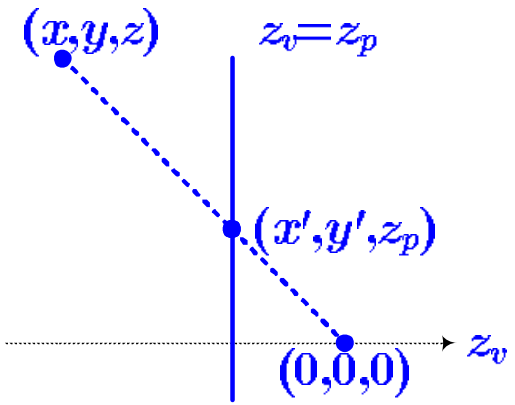


图5-17　透视投影变换方程推导

由 $z_p = z - zu$ 可得 $u = (z - z_p)/z$。由此可得透视变换方程

$$\begin{cases} x' = x - \dfrac{z - z_p}{z} x = \dfrac{z_p}{z} x \\ y' = y - \dfrac{z - z_p}{z} y = \dfrac{z_p}{z} y \end{cases}$$

$$\begin{cases} x' = x - xu \\ y' = y - yu \\ z_p = z - zu \end{cases}$$

3. 变换矩阵

选取 $h=-z$，由 $x'=\frac{z_p}{z}x$ 和 $y'=\frac{z_p}{z}y$ 可得下列使用齐次坐标的透视变换方程。

$$\begin{cases} x_h = x' \bullet h = -z_p x \\ y_h = y' \bullet h = -z_p y \\ z_h = z' \bullet h = -z_p z \\ h = -z \end{cases}$$

由该变换方程可得透视变换的变换矩阵为

$$M_{\text{pers}} = \begin{pmatrix} -z_p & 0 & 0 & 0 \\ 0 & -z_p & 0 & 0 \\ 0 & 0 & -z_p & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

4. 投影坐标的计算

由

$$\begin{cases} x_h = -z_p x \\ y_h = -z_p y \\ z_h = -z_p z \\ h = -z \end{cases}$$

得

$$\begin{cases} x' = \dfrac{x_h}{h} = \dfrac{z_p}{z} x \\ y' = \dfrac{y_h}{h} = \dfrac{z_p}{z} y \end{cases}$$

$$M_{\text{pers}} = \begin{pmatrix} -z_p & 0 & 0 & 0 \\ 0 & -z_p & 0 & 0 \\ 0 & 0 & -z_p & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

5.7.3 带深度提示的透视变换

1. 直接构造变换方程的缺陷

- 变换后 z 坐标为常数（ $z'=z_h/h=-z_pz/(-z)=z_p$ ），不能保留深度信息。
- 若使用原始 z 坐标作为深度信息保留，即 $z'=z$ ，则 $z_h=z'h=-z^2$ ，不能由此构造变换矩阵。

2. 解决办法

使用齐次坐标通过其他途径（远近平面）构造变换方程，使得该变换与其他变换复合以后能够保留深度信息。

$$\begin{cases} x_h=-z_px \\ y_h=-z_py \\ z_h=-z_pz \\ h=-z \end{cases}$$

3. 变换矩阵的改造

已知：投影中心为 $R(0,0,0)$，观察面为 $z_v=z_p$，窗口范围为 $(x_L,y_B)\sim(x_R,y_T)$，远近截面为 $z_F\sim z_N$。

为了在投影变换中保留深度提示（实际上，深度提示只需要保持原始的远近关系，而不需要保持具体的深度值），引入 s 和 t，将计算 z_h 的方程由 $z_h=-z_pz$ 修改为 $z_h=sz+t$（加入常数项），即将投影变换矩阵作如下修改，以避免变换后的 z 坐标为常数。

$$\begin{pmatrix}-z_p & 0 & 0 & 0\\ 0 & -z_p & 0 & 0\\ 0 & 0 & -z_p & 0\\ 0 & 0 & -1 & 0\end{pmatrix}\Rightarrow\begin{pmatrix}-z_p & 0 & 0 & 0\\ 0 & -z_p & 0 & 0\\ 0 & 0 & s & t\\ 0 & 0 & -1 & 0\end{pmatrix}$$

只需考虑变换以后的 z 坐标。

由 $z_h = sz + t$ 和 $h = -z$ 可得 $z' = z_h / h = -s - t / z$。考虑保持远近面的深度值不变，可得如下方程组

$$\begin{cases} z_F = -s - t / z_F \\ z_N = -s - t / z_N \end{cases}$$

解得 $s = -(z_F + z_N)$， $t = z_F z_N$。由此可得带深度提示的透视变换矩阵为

$$M_{\text{pers}} = \begin{pmatrix} -z_p & 0 & 0 & 0 \\ 0 & -z_p & 0 & 0 \\ 0 & 0 & -(z_F + z_N) & z_F z_N \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

从而， $z' = -s - t / z = (z_F + z_N) - z_F z_N / z$。

因为远近面位于投影中心同一侧，所以 $z_F z_N \geq 0$，从而 z' 是 z 的增函数，能够达到保留深度信息的目的。

$$\begin{pmatrix} -z_p & 0 & 0 & 0 \\ 0 & -z_p & 0 & 0 \\ 0 & 0 & s & t \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

4. 改造后的变换方程

根据上述透视投影变换矩阵，有

$$\begin{cases} x_h = -z_p x \\ y_h = -z_p y \\ z_h = -(z_F + z_N)z + z_F z_N \\ h = -z \end{cases}$$

从而

$$\begin{cases} x' = z_p x / z \\ y' = z_p y / z \\ z' = (z_F + z_N) - z_F z_N / z \end{cases}$$

$$M_{\text{pers}} = \begin{pmatrix} -z_p & 0 & 0 & 0 \\ 0 & -z_p & 0 & 0 \\ 0 & 0 & -z_F - z_N & z_F z_N \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

5.8 规范化变换

5.8.1 观察体的变化

如图5-18所示。投影变换完成以后，观察体已经变换成矩形管道。对于规范化变换，有

- 矩形管道变换为规范化观察体。
- 矩形管道中的点（PC）变换为三维视口中的点（NC）。
- 三维视口范围不超过规范化观察体范围。

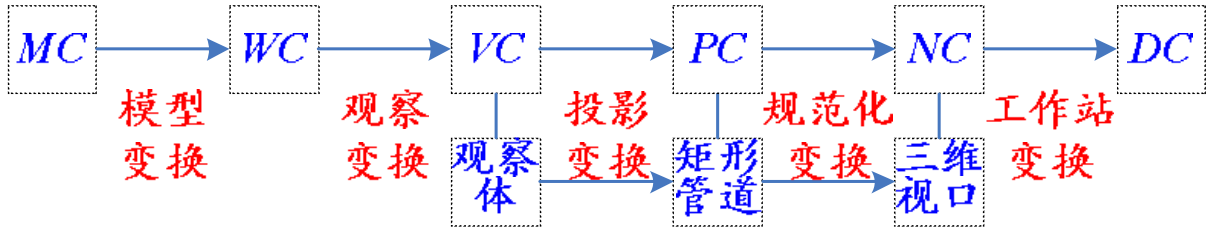


图5-18 观察体的变化

5.8.2 一般规范化变换

1. 已知条件

如图5-19所示。

- 窗口范围：$(x_L, y_B) \sim (x_R, y_T)$。
- 视口边界：$(x_L', y_B') \sim (x_R', y_T')$。
- 近远截面：z_N, z_F。
- 深度范围：z_N', z_F'。

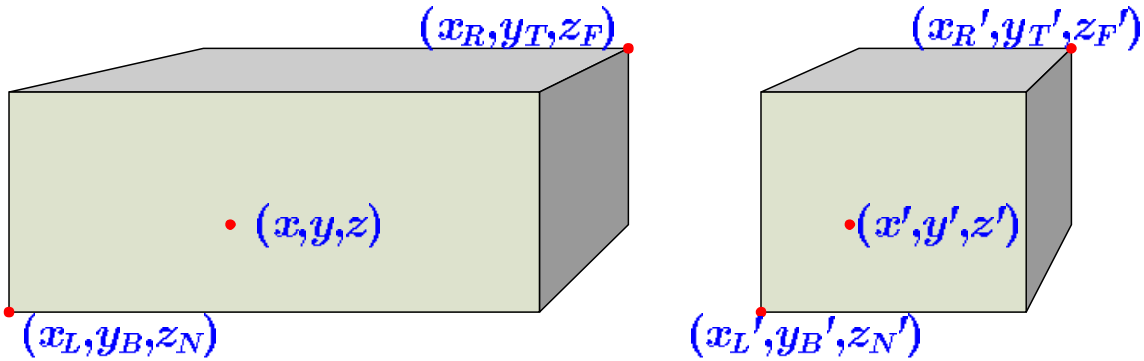
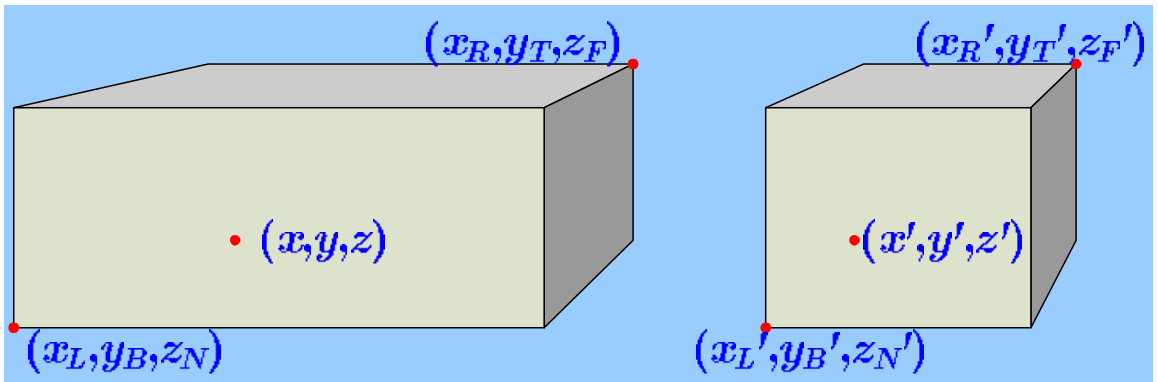


图5-19 规范化变换的推导

2. 变换矩阵的构造

规范化变换将矩形观察体中位于 (x,y,z) 的点映射到视口中坐标为 (x',y',z') 的点，为了使视口与矩形观察体中的对象有同样的相对位置，必须满足

$$\begin{cases} \dfrac{x'-x_L'}{x_R'-x_L'}=\dfrac{x-x_L}{x_R-x_L} \\ \dfrac{y'-y_B'}{y_T'-y_B'}=\dfrac{y-y_B}{y_T-y_B} \\ \dfrac{z'-z_N'}{z_F'-z_N'}=\dfrac{z-z_N}{z_F-z_N} \end{cases}$$



解得

$$\begin{cases} x' = x_L' + (x - x_L)\dfrac{x_R' - x_L'}{x_R - x_L} \\ y' = y_B' + (y - y_B)\dfrac{y_T' - y_B'}{y_T - y_B} \\ z' = z_N' + (z - z_N)\dfrac{z_F' - z_N'}{z_F - z_N} \end{cases}$$

从而可得变换方程为

$$\begin{cases} x' = \dfrac{x_R' - x_L'}{x_R - x_L}x + \dfrac{x_R x_L' - x_L x_R'}{x_R - x_L} \\ y' = \dfrac{y_T' - y_B'}{y_T - y_B}y + \dfrac{y_T y_B' - y_B y_T'}{y_T - y_B} \\ z' = \dfrac{z_F' - z_N'}{z_F - z_N}z + \dfrac{z_F z_N' - z_N z_F'}{z_F - z_N} \end{cases}$$

$$\begin{cases} \dfrac{x' - x_L'}{x_R' - x_L'} = \dfrac{x - x_L}{x_R - x_L} \\ \dfrac{y' - y_B'}{y_T' - y_B'} = \dfrac{y - y_B}{y_T - y_B} \\ \dfrac{z' - z_N'}{z_F' - z_N'} = \dfrac{z - z_N}{z_F - z_N} \end{cases}$$

由变换方程可知变换矩阵为

$$M_{\text{norm}}=\begin{pmatrix} \dfrac{x_R{}'-x_L{}'}{x_R-x_L} & 0 & 0 & \dfrac{x_Rx_L{}'-x_Lx_R{}'}{x_R-x_L} \\ 0 & \dfrac{y_T{}'-y_B{}'}{y_T-y_B} & 0 & \dfrac{y_Ty_B{}'-y_By_T{}'}{y_T-y_B} \\ 0 & 0 & \dfrac{z_F{}'-z_N{}'}{z_F-z_N} & \dfrac{z_Fz_N{}'-z_Nz_F{}'}{z_F-z_N} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{cases} x{}'=\dfrac{x_R{}'-x_L{}'}{x_R-x_L}\,x+\dfrac{x_Rx_L{}'-x_Lx_R{}'}{x_R-x_L} \\ y{}'=\dfrac{y_T{}'-y_B{}'}{y_T-y_B}\,y+\dfrac{y_Ty_B{}'-y_By_T{}'}{y_T-y_B} \\ z{}'=\dfrac{z_F{}'-z_N{}'}{z_F-z_N}\,z+\dfrac{z_Fz_N{}'-z_Nz_F{}'}{z_F-z_N} \end{cases}$$

5.8.3 OpenGL中的规范化变换

在OpenGL中，规范化变换分解成两个变换，分别是矩形管道到规范化立方体的变换和规范化立方体到三维视口的变换。其中，矩形管道到规范化立方体的变换称为规范化变换，规范化立方体到三维视口的变换称为视口变换，规范化立方体规定为 $(-1,-1,-1)\sim(1,1,1)$，使用左手坐标系。

1. 矩形管道到规范化立方体的变换

此时

$$\begin{cases} x_L' = -1 \\ x_R' = 1 \end{cases} \qquad \begin{cases} y_B' = -1 \\ y_T' = 1 \end{cases} \qquad \begin{cases} z_N' = -1 \\ z_F' = 1 \end{cases}$$

从而得到矩形管道到规范化立方体的变换矩阵为

$$M_{\text{norm}} = \begin{pmatrix} \dfrac{x_R{}' - x_L{}'}{x_R - x_L} & 0 & 0 & \dfrac{x_R x_L{}' - x_L x_R{}'}{x_R - x_L} \\ 0 & \dfrac{y_T{}' - y_B{}'}{y_T - y_B} & 0 & \dfrac{y_T y_B{}' - y_B y_T{}'}{y_T - y_B} \\ 0 & 0 & \dfrac{z_F{}' - z_N{}'}{z_F - z_N} & \dfrac{z_F z_N{}' - z_N z_F{}'}{z_F - z_N} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \dfrac{2}{x_R - x_L} & 0 & 0 & -\dfrac{x_R + x_L}{x_R - x_L} \\ 0 & \dfrac{2}{y_T - y_B} & 0 & -\dfrac{y_T + y_B}{y_T - y_B} \\ 0 & 0 & \dfrac{2}{z_F - z_N} & -\dfrac{z_F + z_N}{z_F - z_N} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{cases} x_L{}' = -1 \\ x_R{}' = 1 \end{cases} \quad \begin{cases} y_B{}' = -1 \\ y_T{}' = 1 \end{cases} \quad \begin{cases} z_N{}' = -1 \\ z_F{}' = 1 \end{cases}$$

2. 规范化立方体到三维视口的变换

OpenGL将三维视口规定为 $(x_L', y_B', -1) \sim (x_R', y_T', 1)$，其中 z 坐标用来保存深度信息。此时，

$$\begin{cases} x_L = -1 \\ x_R = 1 \end{cases} \quad \begin{cases} y_B = -1 \\ y_T = 1 \end{cases} \quad \begin{cases} z_N = -1 \\ z_F = 1 \end{cases} \quad \begin{cases} z_N' = -1 \\ z_F' = 1 \end{cases}$$

所以，规范化立方体到三维视口的变换矩阵为

$$M_{\text{view}} = \begin{pmatrix} \frac{x_R{}' - x_L{}'}{x_R - x_L} & 0 & 0 & \frac{x_R x_L{}' - x_L x_R{}'}{x_R - x_L} \\ 0 & \frac{y_T{}' - y_B{}'}{y_T - y_B} & 0 & \frac{y_T y_B{}' - y_B y_T{}'}{y_T - y_B} \\ 0 & 0 & \frac{z_F{}' - z_N{}'}{z_F - z_N} & \frac{z_F z_N{}' - z_N z_F{}'}{z_F - z_N} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{x_R{}' - x_L{}'}{2} & 0 & 0 & \frac{x_R{}' + x_L{}'}{2} \\ 0 & \frac{y_T{}' - y_B{}'}{2} & 0 & \frac{y_T{}' + y_B{}'}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{cases} x_L = -1 \\ x_R = 1 \end{cases} \quad \begin{cases} y_B = -1 \\ y_T = 1 \end{cases} \quad \begin{cases} z_N = -1 \\ z_F = 1 \end{cases} \quad \begin{cases} z_N{}' = -1 \\ z_F{}' = 1 \end{cases}$$

在OpenGL中，使用glViewport(L, B, W, H)定义一个视口，并生成规范化立方体到三维视口的变换。

此时，因为 $x_L' = L$， $y_B' = B$， $x_R' = L + W$， $y_T' = B + H$，所以该变换矩阵就是

$$M_{\text{view}} = \begin{pmatrix} \mathrm{W}/2 & 0 & 0 & L + W/2 \\ 0 & H/2 & 0 & B + H/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \frac{x_R' - x_L'}{2} & 0 & 0 & \frac{x_R' + x_L'}{2} \\ 0 & \frac{y_T' - y_B'}{2} & 0 & \frac{y_T' + y_B'}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

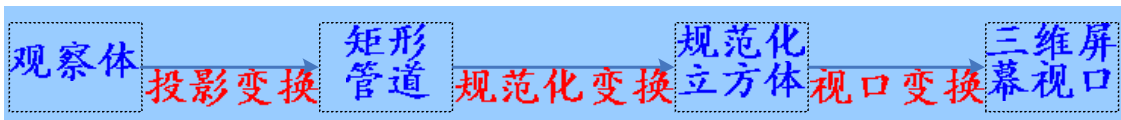
5.8.4 投影变换的规范化

在OpenGL中，投影变换和规范化变换（矩形管道到规范化立方体的变换）是由一个函数完成的，称为投影变换的规范化。

1. 平行投影的规范化

OpenGL只提供了正投影。对于正投影，因为 $M_{\text{para}} = I$，所以

$$M_{\text{proj}} = M_{\text{norm}} = \begin{pmatrix} \dfrac{2}{x_R - x_L} & 0 & 0 & -\dfrac{x_R + x_L}{x_R - x_L} \\ 0 & \dfrac{2}{y_T - y_B} & 0 & -\dfrac{y_T + y_B}{y_T - y_B} \\ 0 & 0 & \dfrac{2}{z_F - z_N} & -\dfrac{z_F + z_N}{z_F - z_N} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



在OpenGL中，正投影通过调用glOrtho(L, R, B, T, N, F)实现。其中，参数 N 和 F 使用深度值，从而 $z_N = -N$， $z_F = -F$，所以

$$M_{\text{proj}} = \begin{pmatrix} \frac{2}{R-L} & 0 & 0 & -\frac{R+L}{R-L} \\ 0 & \frac{2}{T-B} & 0 & -\frac{T+B}{T-B} \\ 0 & 0 & \frac{-2}{F-N} & -\frac{F+N}{F-N} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

该矩阵就是glOrtho()生成的变换矩阵。

$$\begin{pmatrix} \frac{2}{x_R-x_L} & 0 & 0 & -\frac{x_R+x_L}{x_R-x_L} \\ 0 & \frac{2}{y_T-y_B} & 0 & -\frac{y_T+y_B}{y_T-y_B} \\ 0 & 0 & \frac{2}{z_F-z_N} & -\frac{z_F+z_N}{z_F-z_N} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2. 透视投影的规范化

$$\frac{2(-z_F-z_N)}{z_F-z_N}+\frac{z_F+z_N}{z_F-z_N}$$

$$M_{\text{proj}}=M_{\text{norm}}M_{\text{pers}}$$

$$=\begin{pmatrix}\frac{2}{x_R-x_L} & 0 & 0 & -\frac{x_R+x_L}{x_R-x_L}\\ 0 & \frac{2}{y_T-y_B} & 0 & -\frac{y_T+y_B}{y_T-y_B}\\ 0 & 0 & \frac{2}{z_F-z_N} & -\frac{z_F+z_N}{z_F-z_N}\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}-z_p & 0 & 0 & 0\\ 0 & -z_p & 0 & 0\\ 0 & 0 & -z_F-z_N & z_Fz_N\\ 0 & 0 & -1 & 0\end{pmatrix}$$

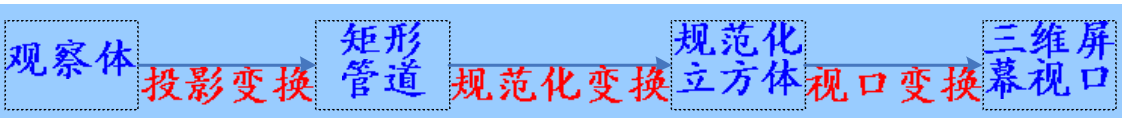
$$=\begin{pmatrix}\frac{-2z_p}{x_R-x_L} & 0 & \frac{x_R+x_L}{x_R-x_L} & 0\\ 0 & \frac{-2z_p}{y_T-y_B} & \frac{y_T+y_B}{y_T-y_B} & 0\\ 0 & 0 & -\frac{z_F+z_N}{z_F-z_N} & \frac{2z_Fz_N}{z_F-z_N}\\ 0 & 0 & -1 & 0\end{pmatrix}$$

（2）OpenGL中的变换矩阵。在OpenGL中，透视投影通过调用glFrustum(*L*, *R*, *B*, *T*, *N*, *F*)实现。其中，投影中心为观察坐标原点，观察面为近平面，参数 N 和 F 使用深度值，从而 $z_N=-N$，$z_F=-F$，$z_p=-N$，所以

$$M_{\text{proj}}=\begin{pmatrix} \dfrac{2N}{R-L} & 0 & \dfrac{R+L}{R-L} & 0 \\ 0 & \dfrac{2N}{T-B} & \dfrac{T+B}{T-B} & 0 \\ 0 & 0 & -\dfrac{F+N}{F-N} & -\dfrac{2FN}{F-N} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

该矩阵就是glFrustum()生成的变换矩阵。

$$\begin{pmatrix} \dfrac{-2z_p}{x_R-x_L} & 0 & \dfrac{x_R+x_L}{x_R-x_L} & 0 \\ 0 & \dfrac{-2z_p}{y_T-y_B} & \dfrac{y_T+y_B}{y_T-y_B} & 0 \\ 0 & 0 & -\dfrac{z_F+z_N}{z_F-z_N} & \dfrac{2z_Fz_N}{z_F-z_N} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$



5.9 三维裁剪

5.9.1 三维裁剪的一般思路

1. 三维裁剪算法的功能

（1）识别并保留在观察体以内的部分以在输出设备中显示，所有在观察体以外的部分被丢弃。

（2）按照观察体边界平面进行裁剪。

2. 线段的裁剪

设某边界平面的方程为 $Ax+By+Cz+D=0$。

（1）将线段端点代入该方程，判断端点与边界的位置关系。

- $Ax+By+Cz+D>0$，该端点在边界以外。
- $Ax+By+Cz+D<0$，该端点在边界以内。

（2）两端点均在同一边界以外，舍弃；两端点均在所有边界以内，保留。

（3）其他，求交点（将直线方程和边界平面方程联立）。

3. 多边形面的裁剪

（1）测试物体的坐标范围。

- 若坐标范围在所有边界内部，则保留该对象。
- 若坐标范围在某一边界外部，则舍弃该对象。

（2）求多边形每一边与观察体边界平面的交点。

4. 裁剪边界的特征

如图5-20所示。

- 边界平面的方向：决定于投影类型、投影窗口、投影中心。
- 前后面：平行于观察面，z坐标为常数。
- 4个侧面：方向任意，不便于计算直线与边界的交点。
- 先投影后裁剪：裁剪前通过投影变换将观察体变成矩形管道。

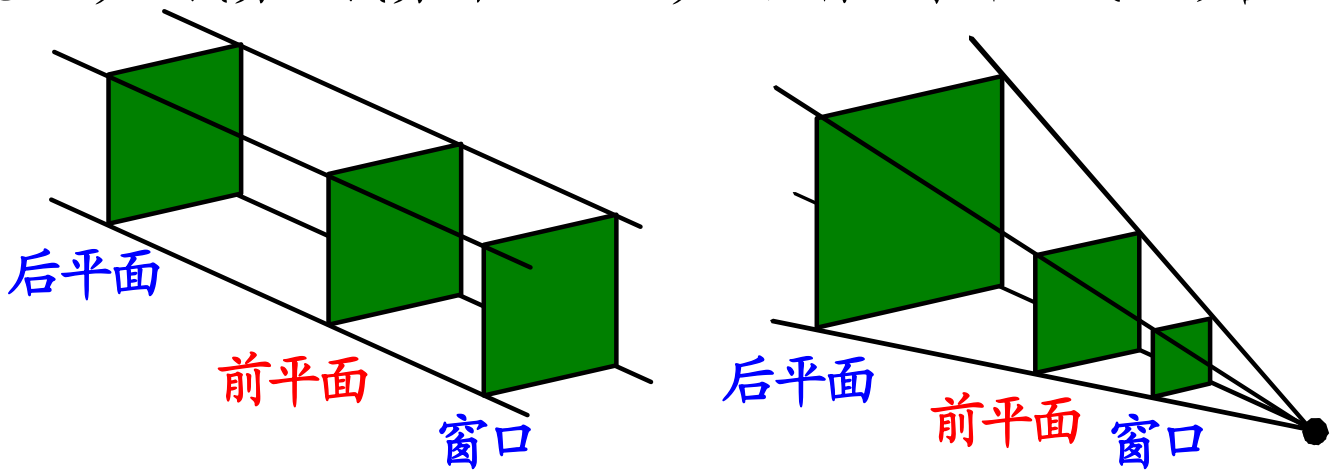


图5-20　裁剪边界的特征

5. 使用矩形管道裁剪的优越性

- 上下面：y 坐标为常数。
- 左右面：x 坐标为常数。
- 前后面：z 坐标为常数。

6. 先投影与先裁剪的比较

先投影后裁剪需要变换的顶点数并不一定比先裁剪后投影多，且裁剪工作比较简单。例如，对于如图5-21所示的裁剪，先裁剪后投影，需要变换8个顶点，而先投影后裁剪，只需变换4个顶点。

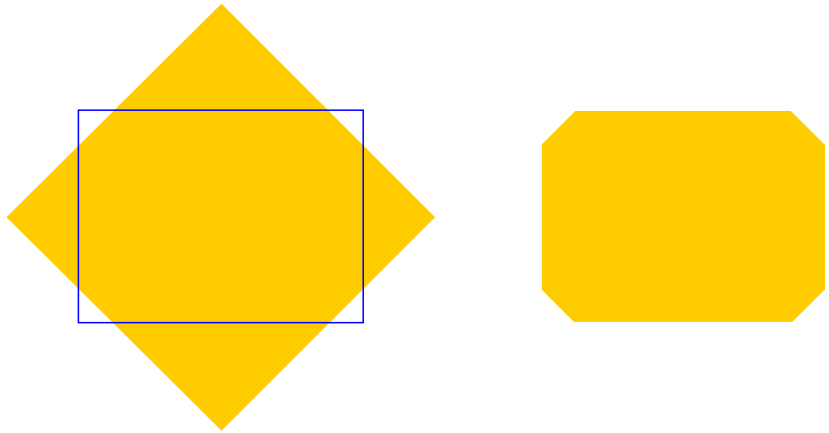


图5-21　先投影与先裁剪的比较

5.9.2 线段的裁剪算法

采用视口裁剪。

1. 区域码

标识端点相对于视口边界的位置。

2. 坐标区域与区域码各位的关系

从右到左编码：$b_T b_B b_R b_L b_N b_F$（上、下、右、左、近、远）。

3. 如何对端点编码

$$\begin{cases} b_L = 1, & \text{if } x < x_L \\ b_R = 1, & \text{if } x > x_R \end{cases} \qquad \begin{cases} b_B = 1, & \text{if } y < y_B \\ b_T = 1, & \text{if } y > y_T \end{cases} \qquad \begin{cases} b_N = 1, & \text{if } z < z_N \\ b_F = 1, & \text{if } z > z_F \end{cases}$$

4. 算法描述

（1）对端点编码。

（2）是否完全在内：两端点区域码均为0，保留，退出。

（3）是否完全在外：两端点区域码按位与不等于0，舍弃，退出。

（4）不能确定：两端点区域码按位与等于0。

- 找到线段的一个外端点，将线段与相应边界求交，舍弃外端点与交点之间的部分。
- 对剩余部分重复上述过程。
- 直到线段被舍弃或找到保留部分为止。

5. 交点计算

设端点为 $P_1(x_1, y_1, z_1)$ 和 $P_2(x_2, y_2, z_2)$。

（1）线段 P_1P_2 的参数方程为

$$\begin{cases} x = x_1 + (x_2 - x_1)u \\ y = y_1 + (y_2 - y_1)u \\ z = z_1 + (z_2 - z_1)u \end{cases} \quad (0 \le u \le 1)$$

（2）计算交点。将选定的边界坐标代入合适的方程，解出 u，然后将 u 代入参数方程，求得交点坐标。例如，由 $z = c$ 可得

$$u = (c - z_1)/(z_2 - z_1)$$
$$x = x_1 + (x_2 - x_1)u$$
$$y = y_1 + (y_2 - y_1)u$$

5.9.3 多边形面的裁剪算法♫

可以将Sutherland-Hodgeman算法改写成如下三维算法。

（1）用左边界裁剪多边形，按顺序产生新顶点序列。

（2）用右边界裁剪多边形，按顺序产生新顶点序列。

（3）用下边界裁剪多边形，按顺序产生新顶点序列。

（4）用上边界裁剪多边形，按顺序产生新顶点序列。

（5）用近边界裁剪多边形，按顺序产生新顶点序列。

（6）用远边界裁剪多边形，按顺序产生新顶点序列。

【注】如果使用上述算法裁剪凹多边形，可能得不到希望的结果，这时可将凹多边形分割为凸多边形以后再进行裁剪。

5.10 练习题

5.10.1 基础知识题

1. 已知三个顶点$A(1, 2, 1)$、$B(3, 4, 2)$和$C(2, 5, 3)$，从里向外以右手系形成逆时针方向。请构造出这三个顶点所确定的平面方程。

2. 已知旋转轴为z轴，旋转角为θ。请使用齐次坐标写出该旋转变换的变换矩阵和变换方程。

3. 已知：$A(3, 3, 5)$和$B(6, 7, 5)$，旋转轴为AB，旋转角为θ。请使用齐次坐标写出该旋转变换的变换矩阵和变换方程。

4. 已知旋转轴为直线AB，其中A=(0, 0, 0)，B=(3, 4, 0)，请构造绕AB旋转90度的旋转变换。

5. 已知旋转角为60度，旋转轴为AB，请构造该三维旋转变换的变换矩阵M，结果至少保留3位小数，其中A=(1, 2, 0)，B=(1, 2, 1)。

6. 已知缩放系数为s_x、s_y和s_z，固定点位置为(x_f, y_f, z_f)，请构造该缩放变换的变换矩阵。

7. 已知新坐标系统的原点位置定义在旧坐标系的(x_0, y_0, z_0)处，且单位轴向量分别为(u_1, u_2, u_3)、(v_1, v_2, v_3)和(n_1, n_2, n_3)，分别对应新的x、y和z轴。请构造完整的从旧坐标系到新坐标系的坐标变换矩阵。

8. 已知：观察参考点为(1, 1, 1)，观察面法向量为(4, 3, 0)，观察向上向量为(-3, 4, 0)。请构造从世界坐标到观察坐标的变换，写出变换矩阵。

9. 已知在原坐标系中某个平面的方程为$3x+4y-10=0$，试求变换矩阵M，使该平面方程在新坐标系下变成$z=0$。其中，新坐标系的y方向为(-4, 3, 0)，且新坐标系的原点(2, 1, 0)在该平面上。

10. 已知投影向量为(3, 4, 1)，投影面为xy平面，请根据定义计算该平行投影的变换矩阵。

11. 求经过平行投影变换后点(1, 2, 3)的坐标。已知：观察面为$z=-4$，投影向量为(1, 1, 1)。

12. 已知投影中心为原点，投影面为$z=-1$，请根据定义计算该透视投影的变换矩阵。

13. 求经过透视投影变换后点(1, 2, 3)的坐标。已知：观察面为$z=-4$，投影中心为(0, 0, 0)。

14. 已知矩形管道观察体为(1, 1, 1)~(10, 10, 10)，规范化立方体为(-1, -1, -1)~(1, 1, 1)，要将窗口中位于(x, y, z)的点映射到规范化立方体坐标为(x', y', z')的点，请构造变换公式和变换矩阵。

5.10.2 阶段实习题

编写一个程序，显示如图所示的两个相交长方体。要求自己构造透视变换函数，不能调用图形软件包提供的透视变换函数。其中，较近的长方体用红色显示，较远的长方体用蓝色显示。