第 3 章　三维仿射变换

电子课件
三维仿射变换

几何变换是对图形进行平移(translation)、缩放(scale)、旋转(rotation)、反射(reflection)和错切(shear),这 5 种变换可以统一用仿射变换(affine transformation)来表示。三维物体的运动和变形是通过三维仿射变换来实现的。例如,用一组平移、缩放和旋转的变换,可以描述物体从建模坐标系向世界坐标系的变换。图 3-1 为由球体类对象构成的三维场景,其中平移变换用于确定球体对象在世界坐标系中的位置,缩放变换用于建立不同大小的球体,代表“太阳”“月亮”和“地球”。旋转变换用于设置“地球”的自转和公转,以及“月亮”环绕“地球”的旋转。

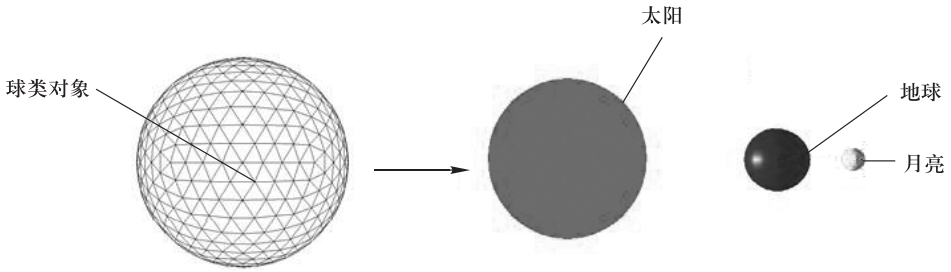


图 3-1　球体类三维场景

3.1　三 维 变 换

设 $P(x,y,z)$ 为变换前的三维点,$P'(x',y',z')$ 为变换后的三维点。这里,变换前后的三维点用列矩阵表示。

3.1.1　平移变换

平移是一种不产生变形而移动物体的变换,物体上每个点移动相同数量的坐标。平移变换将 P 点沿直线路径移动到 P' 位置,如图 3-2 所示。

平移变换的坐标表示为

$$\begin{cases} x' = x + T_x \\ y' = y + T_y \\ z' = z + T_z \end{cases}$$

其中，T_x、T_y 和 T_z 为平移系数。这里，T 代表 Translate。

如果使用列向量表示三维点，相应的矩阵表示为

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3-1}$$

公式 3-1 中，$\boldsymbol{P}' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$，$\boldsymbol{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$，$\boldsymbol{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$。

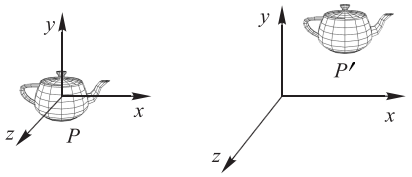


图 3-2　平移变换

3.1.2　缩放变换

缩放变换是指 P 点相对于坐标原点 O，沿 x 方向缩放 S_x 倍，沿 y 方向缩放 S_y 倍，沿 z 方向缩放 S_z 倍得到 P'点的过程，如图 3-3 所示。

缩放变换的坐标表示为

$$\begin{cases} x' = x \cdot S_x \\ y' = y \cdot S_y \\ z' = z \cdot S_z \end{cases}$$

其中，S_x、S_y 和 S_z 为缩放系数。这里，S 代表 Scale。

相应的矩阵表示为

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3-2}$$

缩放变换可以改变三维图形的形状。当 $S_x = S_y = S_z$ 且 S_x、S_y、S_z 大于 1 时，图形等比放大；当 $S_x = S_y = S_z$ 且 S_x、S_y、S_z 大于 0 且小于 1 时，图形等比缩小；当 $S_x \neq S_y \neq S_z$ 时，图形发生形变。

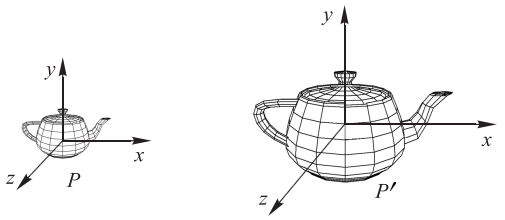


图 3-3 缩放变换

3.1.3 旋转变换

旋转变换是指 P 点相对于坐标原点 O 旋转一个角度 β,得到 P'点的过程。图 3-4 表示的是二维旋转变换,P 和 P'为变换前后的二维点,α 为起始角,β 为旋转角,r 为旋转半径。

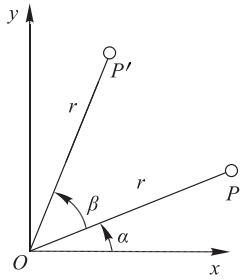


图 3-4 二维旋转变换

P 点的极坐标表示为

$$\begin{cases} x = r\cos\ \alpha \\ y = r\sin\ \alpha \end{cases}$$

P'点表示为

$$\begin{cases} x' = r\cos\ (\alpha + \beta) = r\cos\ \alpha\cos\ \beta - r\sin\ \alpha\sin\ \beta = x\cos\ \beta - y\sin\ \beta \\ y' = r\sin\ (\alpha + \beta) = r\cos\ \alpha\sin\ \beta + r\sin\ \alpha\cos\ \beta = x\sin\ \beta + y\cos\ \beta \end{cases}$$

相应的矩阵表示为

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\ \beta & -\sin\ \beta \\ \sin\ \beta & \cos\ \beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

因此,二维旋转变换矩阵

$$R = \begin{bmatrix} \cos\ \beta & -\sin\ \beta \\ \sin\ \beta & \cos\ \beta \end{bmatrix} \tag{3-3}$$

三维变换矩阵可以看作二维旋转变换矩阵的组合:分别取 x、y、z 为旋转轴,绕每个旋转轴的

三维旋转变换可以看成是在另两个坐标轴组成的二维平面上进行的二维旋转变换。将绕 3 个坐标轴的二维旋转变换组合起来，就可以得到总的三维旋转变换。需要说明的是：当沿坐标轴往坐标原点方向看过去时，沿逆时针方向的旋转角为正向旋转角，即满足右手定则：大拇指指向旋转轴的轴向，四指弯曲的方向为正向。反向旋转将旋转轴取反即可。以下的讲解中，以 β 为正向旋转角。

1. 绕 x 轴旋转

x 坐标不变，而 y、z 坐标发生变化，如图 3-5 所示。

绕 x 轴旋转变换的坐标表示为

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

因此，绕 x 轴的三维旋转变换矩阵

$$\boldsymbol{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \tag{3-4}$$

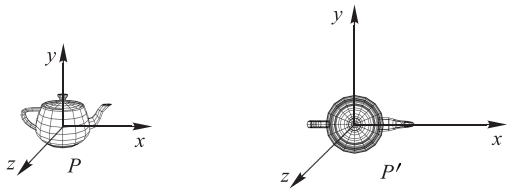


图 3-5　绕 x 轴的三维旋转变换

2. 绕 y 轴旋转

y 坐标不变，而 x、z 坐标发生变化，如图 3-6 所示。

绕 y 轴旋转变换的坐标表示为

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

因此，绕 y 轴的三维旋转变换矩阵

$$\boldsymbol{R}_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{3-5}$$

3. 绕 z 轴旋转

z 坐标不变，而 x、y 坐标发生变化，如图 3-7 所示。

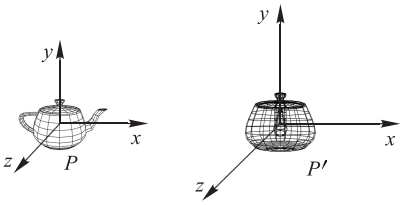


图 3-6 绕 y 轴的三维旋转变换

绕 z 轴旋转变换的坐标表示为

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

因此,绕 z 轴的三维旋转变换矩阵

$$\boldsymbol{R}_z = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3-6}$$

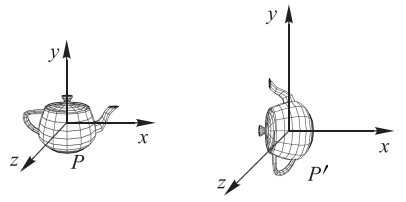


图 3-7 绕 z 轴的三维旋转变换

3.1.4 反射变换

反射变换也称为对称变换,是指 P 点关于原点或某个坐标轴、某个坐标面反射得到 P'点的过程。反射变换可以细分为关于原点反射、关于 x 轴反射、关于 y 轴反射等几种情况。

1. 关于坐标轴的反射

(1) 关于 x 轴反射

关于 x 轴反射变换的坐标表示为

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

因此,关于 x 轴的三维反射变换矩阵

$$\boldsymbol{RE}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3-7}$$

（2）关于 y 轴反射

关于 y 轴反射变换的坐标表示为

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

因此，关于 y 轴的三维反射变换矩阵

$$\boldsymbol{RE}_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3-8}$$

（3）关于 z 轴反射

关于 z 轴反射变换的坐标表示为

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

因此，关于 z 轴的三维反射变换矩阵

$$\boldsymbol{RE}_z = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3-9}$$

2. 关于坐标面的反射

与坐标面相关的反射有关于 xOy 坐标面反射、关于 yOz 坐标面反射、关于 zOx 坐标面反射等几种情况。

（1）关于 xOy 坐标面反射

关于 xOy 坐标面反射，如图 3-8 所示。

坐标表示为

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

变换矩阵

$$\boldsymbol{RE}_{xOy} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3-10}$$

（2）关于 yOz 坐标面反射

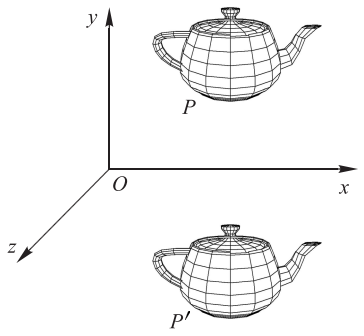


图 3-8 关于 xOy 坐标面反射

关于 yOz 坐标面反射,如图 3-9 所示。

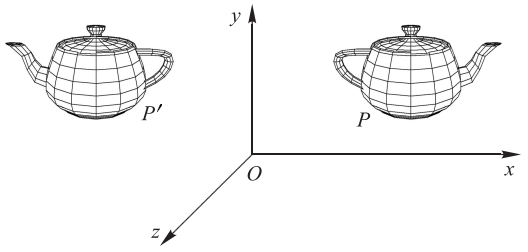


图 3-9 关于 yOz 坐标面反射

坐标表示为

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

变换矩阵

$$\boldsymbol{RE}_{yOz} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3-11}$$

(3) 关于 zOx 坐标面反射

关于 zOx 坐标面反射,如图 3-10 所示。

坐标表示为

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

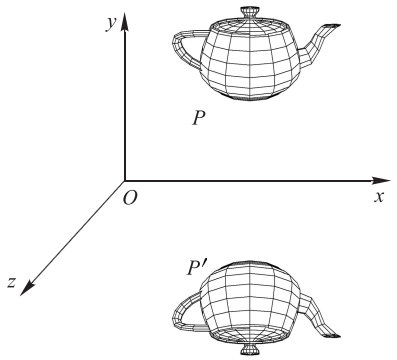


图 3-10　关于 zOx 面反射

变换矩阵

$$\boldsymbol{RE}_{zOx}=\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3-12}$$

3.1.5　错切变换

三维错切变换的坐标表示为

$$\begin{cases}x'=x+by+cz\\ y'=dx+y+fz\\ z'=gx+hy+z\end{cases}$$

因此,三维错切变换矩阵

$$\boldsymbol{SH}=\begin{bmatrix}1 & b & c\\ d & 1 & f\\ g & h & 1\end{bmatrix} \tag{3-13}$$

三维错切变换中,一个坐标的变化受另外两个坐标变化的影响。如果变换矩阵第 1 行中元素 b 和 c 不为 0,则产生沿 x 轴方向的错切;如果第 2 行中元素 d 和 f 不为 0,则产生沿 y 轴方向的错切;如果第 3 行中元素 g 和 h 不为 0,则产生沿 z 轴方向的错切。

1. 沿 x 方向错切

此时,$d=0$,$f=0$,$g=0$,$h=0$。因此,沿 x 方向错切变换矩阵

$$\boldsymbol{SH}_x=\begin{bmatrix}1 & b & c\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3-14}$$

当 $b=0$ 时，错切平面离开 z 轴，沿 x 方向移动 cz 距离；当 $c=0$ 时，错切平面离开 y 轴，沿 x 方向移动 by 距离。

2. 沿 y 方向错切

此时，$b=0, c=0, g=0, h=0$。因此，沿 y 方向错切变换矩阵

$$\boldsymbol{SH}_y = \begin{bmatrix} 1 & 0 & 0 \\ d & 1 & f \\ 0 & 0 & 1 \end{bmatrix} \tag{3-15}$$

当 $d=0$ 时，错切平面离开 z 轴，沿 y 方向移动 fz 距离；当 $f=0$ 时，错切平面离开 x 轴，沿 y 方向移动 dx 距离。

3. 沿 z 方向错切

此时，$b=0, c=0, d=0, f=0$。因此，沿 z 方向错切变换矩阵

$$\boldsymbol{SH}_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ g & h & 1 \end{bmatrix} \tag{3-16}$$

当 $g=0$ 时，错切平面离开 y 轴，沿 z 方向移动 hy 距离；当 $h=0$ 时，错切平面离开 x 轴，沿 z 方向移动 gx 距离。

3.1.6 仿射变换

三维仿射变换的坐标表示为

$$\begin{cases} x' = ax + by + cz + l \\ y' = dx + ey + fz + m \\ z' = gx + hy + iz + n \end{cases} \tag{3-17}$$

矩阵表示为

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} l \\ m \\ n \end{bmatrix} \tag{3-18}$$

变换后的坐标 x'、y'和 z'都是变换前坐标 x、y 和 z 的线性函数。参数 a、b、c、d、e、f、g、h、i 是变形系数，l、m、n 是平移系数。仿射变换具有平行线变换为平行线，有限点映射为有限点的一般特性。平移、缩放、旋转、反射和错切 5 种变换都是三维仿射变换的特例，任何一组三维仿射变换都可以表示为这 5 种变换的组合。

三维图形的仿射变换可以通过点变换来完成。例如直线的变换可以通过对两个顶点坐标进行变换，连接新顶点得到变换后的新直线；多边形的变换可以通过对每个顶点进行变换，连接新顶点得到变换后的新多边形。自由曲线的变换可通过变换控制多边形的控制点后，重新生成

曲线来实现。

从式 3-18 可以看出,三维仿射变换包含了矩阵乘法和矩阵加法,而矩阵加法出现在平移变换上。为了统一用矩阵乘法来完成三维图形的几何变换,引入了齐次坐标。

3.2　基于齐次坐标的三维变换

3.2.1　齐次坐标

所谓齐次坐标就是用 $n+1$ 维向量表示 n 维向量。例如,在二维平面中,点 $P(x,y)$ 的齐次坐标表示为 $(X,Y,W)=(wx,wy,w)$。如果 w 不等于零,就可以用 w 去除齐次坐标来得到原坐标,即 $x=X/W,y=Y/W$。类似地,在三维空间中,点 $P(x,y,z)$ 的齐次坐标表示为 $(X,Y,W)=(wx,wy,wz,w)$。这里,w 为任意不为 0 的缩放系数。如果 $w=1$,就是规范化的齐次坐标。为了避开齐次坐标的除法运算,工程中一般使用规范化齐次坐标点进行几何变换。二维点 $P(x,y)$ 的规范化齐次坐标为 $(x,y,1)$,三维点 $P(x,y,z)$ 的规范化齐次坐标为 $(x,y,z,1)$。

我们知道,如果直接用三维顶点来进行三维变换,平移变换用的是加法,而缩放变换和旋转变换用的是乘法。定义了齐次坐标以后,平移变换就可以用乘法来表示。图形仿射变换可以表示为图形顶点集合的齐次坐标矩阵与变换矩阵相乘的统一形式。

程序清单 3-1:二维齐次坐标点类

```
class CP2
{
public:
    CP2(void);
    virtual ~CP2(void);
    CP2(double x, double y);
public:
    double x, y, w;
};
```

程序说明:对比程序清单 2-1,这里在二维点类中增加了齐次坐标 w。CP3 公有派生自二维齐次坐标点类 CP2,自然继承了齐次坐标 w。

3.2.2　三维几何变换形式

设变换前物体顶点集合的齐次坐标矩阵为

$$\boldsymbol{P}=\begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \\ y_0 & y_1 & \cdots & y_{n-1} \\ z_0 & z_1 & \cdots & z_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

假设有 n 个顶点参与变换,用 $\boldsymbol{P}$ 表示变换前的三维顶点矩阵(4×n),用 P'表示变换后的三维顶点矩阵(4×n),$\boldsymbol{M}$ 表示变换矩阵(4×4),变换后新的物体顶点集合的齐次坐标矩阵如下:

$$\boldsymbol{P}'=\begin{bmatrix} x'_0 & x'_1 & \cdots & x'_{n-1} \\ y'_0 & y'_1 & \cdots & y'_{n-1} \\ z'_0 & z'_1 & \cdots & z'_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

三维变换矩阵

$$\boldsymbol{M}=\begin{bmatrix} a & b & c & l \\ d & e & f & m \\ g & h & i & n \\ p & q & r & s \end{bmatrix}$$

则三维变换公式 $\boldsymbol{P}'=\boldsymbol{M}\cdot\boldsymbol{P}$ 可以写成

$$\begin{bmatrix} x'_0 & x'_1 & \cdots & x'_{n-1} \\ y'_0 & y'_1 & \cdots & y'_{n-1} \\ z'_0 & z'_1 & \cdots & z'_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}=\begin{bmatrix} a & b & c & l \\ d & e & f & m \\ g & h & i & n \\ p & q & r & s \end{bmatrix}\begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \\ y_0 & y_1 & \cdots & y_{n-1} \\ z_0 & z_1 & \cdots & z_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{3-19}$$

3.2.3 三维几何变换矩阵

引入齐次坐标后,三维变换矩阵是一个 4×4 方阵。

$$\boldsymbol{M}=\begin{bmatrix} a & b & c & l \\ d & e & f & m \\ g & h & i & n \\ p & q & r & s \end{bmatrix} \tag{3-20}$$

其中,$\boldsymbol{M}_1=\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ 为 3×3 阶子矩阵,对物体进行缩放、旋转、反射和错切变换;$\boldsymbol{M}_2=\begin{bmatrix} l \\ m \\ n \end{bmatrix}$,为 3×1 阶子矩阵,对物体进行平移变换;$\boldsymbol{M}_3=[p \quad q \quad r]$,为 1×3 阶子矩阵,对物体进行投影变换;$\boldsymbol{M}_4=[s]$,为 1×1 阶子矩阵,对物体进行整体比例变换。

3.2.4　三维基本几何变换矩阵

下面直接给出各种变换的齐次坐标矩阵。

1. 平移变换

平移变换齐次坐标矩阵

$$\boldsymbol{T}=\begin{bmatrix}1 & 0 & 0 & T_x\\ 0 & 1 & 0 & T_y\\ 0 & 0 & 1 & T_z\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-21}$$

2. 缩放变换

缩放变换齐次坐标矩阵

$$\boldsymbol{S}=\begin{bmatrix}S_x & 0 & 0 & 0\\ 0 & S_y & 0 & 0\\ 0 & 0 & S_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-22}$$

3. 旋转变换

绕 x 轴旋转的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\beta & -\sin\beta & 0\\ 0 & \sin\beta & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-23}$$

绕 y 轴旋转的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}\cos\beta & 0 & \sin\beta & 0\\ 0 & 1 & 0 & 0\\ -\sin\beta & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-24}$$

绕 z 轴旋转的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}\cos\beta & -\sin\beta & 0 & 0\\ \sin\beta & \cos\beta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-25}$$

4. 反射变换

关于 x 轴反射的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-26}$$

关于 y 轴反射的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-27}$$

关于 z 轴反射的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-28}$$

关于 xOy 面反射的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-29}$$

关于 yOz 面反射的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-30}$$

关于 zOx 面反射的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-31}$$

5. 错切变换

沿 x 方向错切的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & b & c & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-32}$$

沿 y 方向错切的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & 0 & 0 & 0\\ d & 1 & f & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-33}$$

沿 z 方向错切的齐次坐标矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ g & h & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3-34}$$

3.3　三维复合变换

三维基本变换是相对于坐标原点或坐标轴进行的几何变换。三维复合变换是指对物体做一次以上的基本变换，总变换矩阵是每一步变换矩阵相乘的结果。

$$\boldsymbol{P}'=\boldsymbol{M}\cdot\boldsymbol{P}=\boldsymbol{M}_n\cdot\boldsymbol{M}_{n-1}\cdots\cdots\boldsymbol{M}_2\cdot\boldsymbol{M}_1\cdot\boldsymbol{P}\qquad(n>1) \tag{3-35}$$

其中，$\boldsymbol{M}$ 为复合变换矩阵，$\boldsymbol{M}_1,\boldsymbol{M}_2,\cdots,\boldsymbol{M}_n$ 为 n 个单一的基本变换矩阵。

3.3.1　相对于任一参考点的三维变换

在三维基本变换中，旋转变换和缩放变换是与参考点相关的。相对于任一参考点的缩放变换和旋转变换应表达为复合变换形式。变换方法是首先将参考点平移到坐标系原点，相对于坐标系原点做缩放变换或旋转变换，然后再进行反平移，将参考点平移回原位置。

3.3.2　相对于任意方向的三维变换

相对于任意方向的变换方法是首先对“任意方向”做旋转变换，使“任意方向”与某个坐标轴重合，然后对该坐标轴进行三维基本变换，最后做反向旋转变换，将“任意方向”还原回原来的方向。三维变换中需要进行两次旋转变换，才能使“任意方向”与某一坐标轴重合。一般做法是先将“任意方向”旋转到某个坐标平面内，然后再旋转到与该坐标平面内的某一个坐标轴重合。这里“某一个坐标轴”可以选取 3 个坐标轴中的任意一个，z 轴是个不错的选择。

变换步骤：

（1）平移变换，使得旋转轴通过坐标系原点；

（2）旋转变换，使得旋转轴与某一坐标轴重合；

（3）绕坐标轴完成指定的旋转；

（4）利用逆旋转变换使旋转轴回到其原始方向；

（5）利用逆平移变换使旋转轴回到其原始方向。

程序清单 3-2：三维几何变换类

```
    #include "P3.h"                                          //包含三维齐次坐标点
1   class CTransform3
    {
    public:
        CTransform3(void);
5       virtual ~CTransform3(void);
        void SetMatrix(CP3 * P,int ptNumber);                //三维顶点数组初始化
        void Identity(void);                                 //单位矩阵初始化
        void Translate(double tx,double ty,double tz);       //平移变换
        void Scale(double sx,double sy,double sz);           //缩放变换
10      void Scale(double sx,double sy,double sz,CP3 p);     //相对于任意点的缩放变换
        void Scale(double s);                                //整体缩放变换
        void Scale(double s, CP3 p);                         //相对于任意点的整体缩放变换
        void RotateX(double beta);                           //绕 X 轴旋转变换
        void RotateY(double beta);                           //绕 Y 轴旋转变换
15      void RotateZ(double beta);                           //绕 Z 轴旋转变换
        void RotateX(double beta,CP3 p);                     //相对于任意点的绕 X 轴旋转变换
        void RotateY(double beta,CP3 p);                     //相对于任意点的绕 Y 轴旋转变换
        void RotateZ(double beta,CP3 p);                     //相对于任意点的绕 Z 轴旋转变换
        void ReflectX(void);                                 //关于 X 轴反射变换
20      void ReflectY(void);                                 //关于 Y 轴反射变换
        void ReflectZ(void);                                 //关于 Z 轴反射变换
        void ReflectXOY(void);                               //关于 XOY 面反射变换
        void ReflectYOZ(void);                               //关于 YOZ 面反射变换
        void ReflectZOX(void);                               //关于 ZOX 面反射变换
25      void ShearX(double b,double c);                      //沿 X 方向错切变换
        void ShearY(double d,double f);                      //沿 Y 方向错切变换
        void ShearZ(double g,double h);                      //沿 Z 方向错切变换
        void MultiplyMatrix(void);                           //矩阵相乘
    private:
30      double M[4][4];                                      //三维变换矩阵
        CP3 * P;                                             //三维顶点数组名
        int   ptNumber;                                      //三维顶点个数
    };
    CTransform3::CTransform3(void)
```

```
}
CTransform3:: ~CTransform3(void)
{}
void CTransform3::SetMatrix(CP3 * P,int ptNumber)              //顶点数组初始化
{
    this->P = P;
    this->ptNumber = ptNumber;
}
void CTransform3::Identity(void)                               //单位矩阵
{
    M[0][0] = 1.0, M[0][1] = 0.0, M[0][2] = 0.0, M[0][3] = 0.0;
    M[1][0] = 0.0, M[1][1] = 1.0, M[1][2] = 0.0, M[1][3] = 0.0;
    M[2][0] = 0.0, M[2][1] = 0.0, M[2][2] = 1.0, M[2][3] = 0.0;
    M[3][0] = 0.0, M[3][1] = 0.0, M[3][2] = 0.0, M[3][3] = 1.0;
}
void CTransform3::Translate(double tx,double ty,double tz)     //平移变换
{
    Identity();
    M[0][3] = tx, M[1][3] = ty, M[2][3] = tz;
    MultiplyMatrix();
}
void CTransform3::Scale(double sx,double sy,double sz)         //缩放变换
{
    Identity();
    M[0][0] = sx, M[1][1] = sy, M[2][2] = sz;
    MultiplyMatrix();
}
void CTransform3::Scale(double sx,double sy,double sz,CP3 p)   //相对于任意点的缩放变换
{
    Translate(-p.x, -p.y, -p.z);
    Scale(sx, sy, sz);
    Translate(p.x, p.y, p.z);
}
void CTransform3::Scale(double s)                              //整体缩放变换
{
    Identity();
    M[0][0] = s, M[1][1] = s, M[2][2] = s;
    MultiplyMatrix();
```

```
}
void CTransform3::Scale(double s, CP3 p)                    //相对于任意点的整体缩放变换
{
    Translate(-p.x, -p.y, -p.z);
    Scale(s);
    Translate(p.x, p.y, p.z);
}
void CTransform3::RotateX(double beta)                      //绕 X 轴的旋转变换
{
    Identity();
    beta = beta * PI/180;
    M[1][1] = cos(beta), M[1][2] =-sin(beta);
    M[2][1] = sin(beta), M[2][2] = cos(beta);
    MultiplyMatrix();
}
void CTransform3::RotateY(double beta)                      //绕 Y 轴的旋转变换
{
    Identity();
    beta = beta * PI/180;
    M[0][0] = cos(beta), M[0][2] = sin(beta);
    M[2][0] =-sin(beta), M[2][2] = cos(beta);
    MultiplyMatrix();
}
void CTransform3::RotateZ(double beta)                      //绕 Z 轴的旋转变换
{
    Identity();
    beta = beta * PI/180;
    M[0][0] = cos(beta), M[0][1] =-sin(beta);
    M[1][0] = sin(beta), M[1][1] = cos(beta);
    MultiplyMatrix();
}
void CTransform3::RotateX(double beta,CP3 p)                //相对于任意点的绕 X 轴的旋转变换
{
    Translate(-p.x,-p.y,-p.z);
    RotateX(beta);
    Translate(p.x,p.y,p.z);
}
void CTransform3::RotateY(double beta,CP3 p)                //相对于任意点的绕 Y 轴的旋转变换
```

```
    {
        Translate(-p.x,-p.y,-p.z);
        RotateY(beta);
        Translate(p.x,p.y,p.z);
}
    void CTransform3::RotateZ(double beta,CP3 p)                    //相对于任意点的绕 Z 轴的旋转变换
    {
        Translate(-p.x,-p.y,-p.z);
        RotateZ(beta);
        Translate(p.x,p.y,p.z);
    }
    void CTransform3::ReflectX(void)                                //关于 X 轴的反射变换
    {
        Identity();
        M[1][1] = -1, M[2][2] = -1;
        MultiplyMatrix();
    }
    void CTransform3::ReflectY(void)                                //关于 Y 轴的反射变换
    {
        Identity();
        M[0][0] = -1;M[2][2] = -1;
        MultiplyMatrix();
    }
    void CTransform3::ReflectZ(void)                                //关于 Z 轴的反射变换
{
        Identity();
        M[0][0] = -1;M[1][1] = -1;
        MultiplyMatrix();
    }
    void CTransform3::ReflectXOY(void)                              //关于 XOY 面的反射变换
    {
        Identity();
        M[2][2] = -1;
        MultiplyMatrix();
}
    void CTransform3::ReflectYOZ(void)                              //关于 YOZ 面的反射变换
    {
        Identity();
```

```
        M[0][0] = -1;
150     MultiplyMatrix();
    }
    void CTransform3::ReflectZOX(void)                          //关于 ZOX 面的反射变换
    {
        Identity();
155     M[1][1] = -1;
        MultiplyMatrix();
    }
    void CTransform3::ShearX(double b,double c)                 // 沿 X 方向的错切变换
    {
160     Identity();
        M[0][1] = b;M[0][2] = c;
        MultiplyMatrix();
    }
    void CTransform3::ShearY(double d,double f)                 //沿 Y 方向错切变换
165 {
        Identity();
        M[1][0] = d;M[1][2] = f;
        MultiplyMatrix();
    }
170 void CTransform3::ShearZ(double g,double h)                 //沿 Z 方向错切变换
    {
        Identity();
        M[2][0] = g;M[2][1] = h;
        MultiplyMatrix();
175 }
    void CTransform3::MultiplyMatrix(void)                      //矩阵相乘
    {
        CP3 * PTemp = new CP3[ptNumber];
        for(int i = 0;i < ptNumber; i++)
180         PTemp[i] = P[i];
        for(int i = 0;i < ptNumber; i++)
        {
            P[i].x = M[0][0] * PTemp[i].x+M[0][1] * PTemp[i].y+M[0][2] * PTemp[i].z
                     +M[0][3] * PTemp[i].w;
            P[i].y = M[1][0] * PTemp[i].x+M[1][1] * PTemp[i].y+M[1][2] * PTemp[i].z
                     +M[1][3] * PTemp[i].w;
```

```
185             P[i].z = M[2][0] * PTemp[i].x+M[2][1] * PTemp[i].y+M[2][2] * PTemp[i].z
                        +M[2][3] * PTemp[i].w;
                P[i].w = M[3][0] * PTemp[i].x+M[3][1] * PTemp[i].y+M[3][2] * PTemp[i].z
                        +M[3][3] * PTemp[i].w;
        }
        delete [ ]PTemp;
    }
```

程序说明:第 38~42 行语句读入参与变换的三维顶点矩阵,这是三维变换类的接口。基于齐次坐标的三维顶点使用一维数组 P 表示,元素类型为 CP3,ptNumber 是数组元素个数。第 176~189 行语句实现三维变换矩阵与顶点矩阵的乘法,计算结果仍然存放在三维顶点矩阵中。

程序清单 3-3:三维变换类示例

以立方体正交投影为例,讲解 CTransform3 类的使用方法。在第 2 章的程序清单 2-4 中,已经定义过单位立方体类,名称为 CCube。这里,定义立方体对象名为 cube。

```
1   CTestView::CTestView( )
    {
        // TODO: add construction code here
        Alpha = 0.0, Beta = 0.0;
5       bPlay = FALSE;
        cube.ReadVertex( );
        cube.ReadFacet( );
        transform.SetMatrix(cube.P, 8);
        int nEdge = 300;
10      transform.Scale(nEdge);
        transform.Translate(-nEdge/2, -nEdge/2, -nEdge/2);
    }
```

程序说明:在 TestView.h 文件中添加如下声明:

```
    CCube cube;                 //立方体对象
    CTransform3 transform;      //定义变换对象
15  double Alpha, Beta;         //绕 x 轴旋转 α 角, 绕 y 轴旋转 β 角
    BOOL bPlay;                 //动画开关
```

程序说明(续):第 4 行语句初始化旋转角。第 5 行语句初始化动画状态为“停止”。第 6~7 行语句使用 cube 对象读入立方体的点表与面表。第 8 行语句用立方体的顶点数组 P(顶点个数 8)初始化三维变换类对象 transform。第 9~10 行语句将立方体放大 nEdge 倍,这里使用的是整体缩放函数。第 11 行语句将立方体的体心移到世界坐标系原点处,为围绕世界坐标系原点旋转立方体做准备。代码如下:

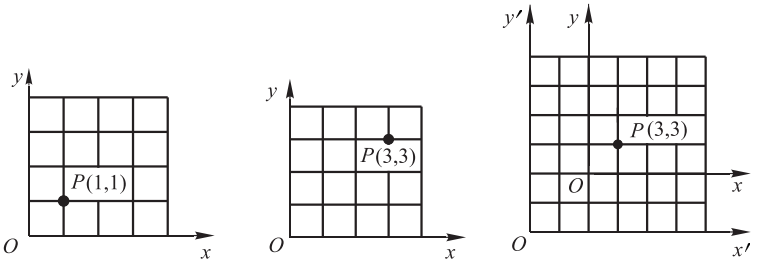
```
void CTestView::OnTimer(UINT_PTR nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    Alpha = 5, Beta = 5;
    transform.RotateX(Alpha);
    transform.RotateY(Beta);
    Invalidate(FALSE);
    CView::OnTimer(nIDEvent);
}
```

程序说明(续):在 WM_TIMER 消息的响应函数 OnTimer 中,绕指定坐标轴旋转立方体。第 20 行语句设置绕 x 轴和绕 y 轴的旋转角度 α 和 β。第 21~22 行语句通过三维旋转类对象 transform,调用 RotateX 和 RotateY 函数实现立方体绕世界坐标系的 x 轴和 y 轴的旋转变换。第 23 行语句使客户无效,系统强制刷新屏幕来绘制变换后的图形。

3.4 坐标系变换

前面讲解的变换都是点变换。在实际应用中,经常需要将物体从一个坐标系变换到另一个坐标系。例如在进行三维观察时,需要将物体的描述从世界坐标系变换到观察坐标系。同一种变换既可以看作是点变换,也可以看作是坐标系变换。点变换是物体顶点位置发生改变,但坐标系位置固定不动。坐标系变换是建立新坐标系描述旧坐标系内的顶点,坐标系位置发生改变,但物体顶点的物理位置固定不动。

在二维坐标系 xOy 下,对于点变换,如将 $P(1,1)$ 点平移到 $P(3,3)$ 点,如图 3-11(a)和(b)所示。如果看作是坐标系变换,则结果如图 3-11(c)所示。图 3-11(c)中,在旧坐标系 xOy 中,P 点的坐标是(1,1),在新坐标系 $x'O'y'$ 中,P 点的坐标为(3,3)。这两种变换方式表明:在解决实际问题时,采用选择点变换还是选择坐标系变换只是处理问题的策略不同,而点与坐标原点



(a) 原图　(b) 点变换　(c) 坐标系变换

图 3-11　二维平移变换

的相对位置是相同的。

对于点变换，在 xyz 三维坐标系中，设平移参数为 (T_x, T_y, T_z)，则 P 点变换到 P' 点。用坐标系变换表示上述平移变换，则是 P 点不动，将 xyz 坐标系的原点从 O 点平移到 $x'y'z'$ 坐标系的原点 O' 点，坐标系平移参数为 $(-T_x, -T_y, -T_z)$。下面给出三维坐标系变换矩阵。

1. 平移变换齐次坐标矩阵

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & -T_x \\ 0 & 1 & 0 & -T_y \\ 0 & 0 & 1 & -T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3-36}$$

相对于点变换而言，坐标系变换的平移参数需要取为负值。

2. 旋转变换矩阵

三维坐标系的旋转变化矩阵应使用与点变换相反方向的旋转变换矩阵表示。

（1）绕 x 轴逆时针方向旋转的齐次坐标矩阵

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3-37}$$

（2）绕 y 轴逆时针方向旋转的齐次坐标矩阵

$$\boldsymbol{M} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3-38}$$

（3）绕 z 轴逆时针方向旋转的齐次坐标矩阵

$$\boldsymbol{M} = \begin{bmatrix} \cos\beta & \sin\beta & 0 & 0 \\ -\sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3-39}$$

上式中，β 取为顺时针旋转角。

3. 反射变换矩阵

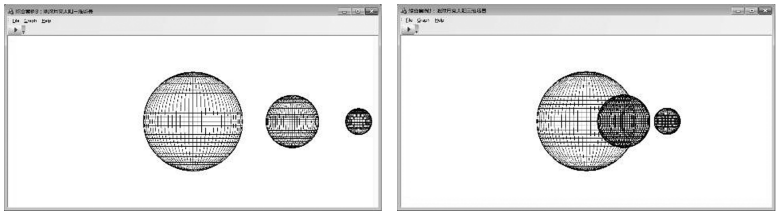
对于三维坐标系的反射变换，直接采用点变换的反射变换矩阵。

3.5 综合案例 3:三维球体类动画

微课视频
三维球体类动画

1. 案例描述

在世界坐标系中构建三维场景。设计球体 CSphere 类,构造太阳、地球和月亮线框模型对象。由大到小分别代表太阳、地球、月亮。太阳位于场景中心静止不动。地球起始位置位于太阳正右方,地球自转的同时围绕太阳逆时针方向公转。月亮起始位置位于地球正右方,围绕地球旋转。试基于三维仿射变换类,编程演示太阳、地球和月亮的运行动画,效果如图 3-12 所示。



(a) 状态1　　(b) 状态2

图 3-12　地球月亮太阳

动画
三维球体类动画

2. 案例设计

计算机图形学中的三维场景(scene)是指由物体、光源、视点等构成的一个世界坐标系,用于展示相关的图形对象。一般在 CTestView 类内定义三维场景,x 轴水平向右为正,y 轴垂直向上为正,z 轴指向观察者。在建模坐标系内使用地理划分法建立球体模型,球心位于建模坐标系原点。声明球类对象 sphere0、sphere1 和 sphere2,用于定义太阳、地球和月亮。请注意,球心坐标是 CShere 类的一个重要数据成员,用于设置太阳、地球和月亮在世界坐标系中的初始位置,如图 3-13 所示。

按照题目要求,月亮围绕地球旋转是绕世界坐标系原点的旋转,属于三维基本变换;月亮的自转是相对于任意参考点的旋转,该参考点是地球的球心,属于复合变换。

3. 案例实现

(1) 声明球类对象和三维变换对象

在 TestView.h 文件中,声明 CSphere 类和 CTransform3 类对象为 protected 成员。

```
protected:
        CSphere sphere[3];                    //球类对象
        CTransform3 transform[3];             //变换对象
        double Beta0, Beta1, Beta2;           //旋转角
```

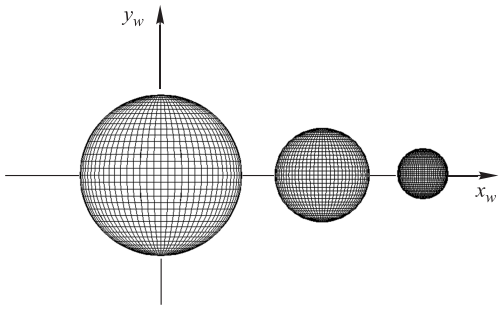


图 3-13　球类对象的初始位置

程序说明：sphere[0]代表太阳，相应的三维变换对象为 transform[0]；sphere[1]代表地球，相应的三维变换对象为 transform[1]；sphere[2]代表月亮，相应的三维变换对象为 transform[2]。球体公转和自转的旋转角用 β_0、β_1、β_2 表示。

（2）初始化球体对象

在 CTestView 的构造函数中，定义球体对象的大小和初始位置。

```
CTestView::CTestView()
{
    // TODO: add construction code here
    bPlay = FALSE;
    for(int i = 0; i < 3; i++)                                    //读入顶点表和表面表
    {
        sphere[i].ReadVertex();
        sphere[i].ReadFacet();
        transform[i].SetMatrix(sphere[i].P, (sphere[i].N1 - 1) * sphere[i].N2 + 3);
    }
    int nRadius1 = 150;                                           //太阳半径
    transform[0].Scale(nRadius1, nRadius1, nRadius1);             //缩放系数
    transform[0].Translate(0, 0, 0);                              //初始位置
    int nRadius2 = 80;                                            //地球半径
    transform[1].Scale(nRadius2, nRadius2, nRadius2);             //缩放系数
    transform[1].Translate(300, 0, 0);                            //初始位置
    int nRadius3 = 40;                                            //月亮半径
    transform[2].Scale(nRadius3, nRadius3, nRadius3);             //缩放系数
    transform[2].Translate(500, 0, 0);                            //初始位置
}
```

程序说明:CSphere 类定义的是单位球体,需要通过三维变换来进行缩放和设置球体出场的初始位置。太阳、地球、月亮的半径依次缩小。太阳的中心与世界坐标系原点重合,其余 2 个球体中心位于 x 轴正向。地球的中心与世界坐标系原点相距 300 像素,月亮的中心与世界坐标系原点相距 500 像素。

(3) 设置旋转方式

在 WM_TIMER 消息的映射函数 OnTimer 中,设置地球和月亮的旋转方式。

```
1    void CTestView::OnTimer(UINT_PTR nIDEvent)
     {
         // TODO: Add your message handler code here and/or call default
         Beta0 = 2;
5        transform[1].RotateY(Beta0);
         Beta1 = 40;
         transform[1].RotateY(Beta1, sphere[1].P[(sphere[1].N1 - 1) * sphere[1].N2 + 2]);
         Beta2 = 20;
         transform[2].RotateY(Beta2, sphere[1].P[(sphere[1].N1 - 1) * sphere[1].N2 + 2]);
10       Invalidate(FALSE);
         CView::OnTimer(nIDEvent);
     }
```

程序说明:第 4~5 行语句中 β_0 是地球公转的角度,地球绕太阳(三维坐标系原点)逆时针方向旋转。第 6~7 行语句中 β_1 是地球自转的角度,地球相对于自身运动的中心逆时针方向旋转。第 8~9 行语句中 β_2 是月亮公转的角度,月亮相对于地球运动的中心逆时针方向旋转。

4. 案例总结

一般情况下,将 CTestView 看作是三维场景空间。物体从建模坐标系中导入三维场景中的过程是建模坐标系向世界坐标系变换的过程。导入场景中的球体对象需要借助三维变换类对象进行缩放和平移变换。虽然本案例基于三维仿射变换实现了球体的相对运动,但也未能将球体的大小与球体细分程度联系到一起。根据 LOD 技术的要求,球体半径越大,细分程度越高;球体越小,细分程度越低,请读者在编程时考虑这个问题。

3.6 本章小结

三维仿射变换的作用有两个:一个是将物体从建模坐标系中导入三维场景中,一般使用平移变换和缩放变换;另一个是旋转物体观察全貌,相对于坐标系原点的旋转可以直接采用基本变换矩阵实现,相对于参考点的旋转变换需要在变换前后对参考点进行平移和反平移。

习　题　3

3.1　二维仿射变换是三维仿射变换的特例，旋转变换只有绕坐标原点的基本变换和绕任意点的复合变换。二维旋转变换可以看作是三维变换中的绕 z 轴的旋转变换。试建立自定义二维坐标系 xOy，原点 O 位于窗口客户区中心，x 轴水平向右为正，y 轴垂直向上为正。在原点下方 $B(0,b)$ 点，悬挂一边长为 $a(|ob|\geqslant\frac{\sqrt{2}}{2}a)$ 的正方形，如图 3-14 所示。试设计二维变换类 CTransform2，实现正方形绕自身中心（B 点）逆时针方向的旋转变换。

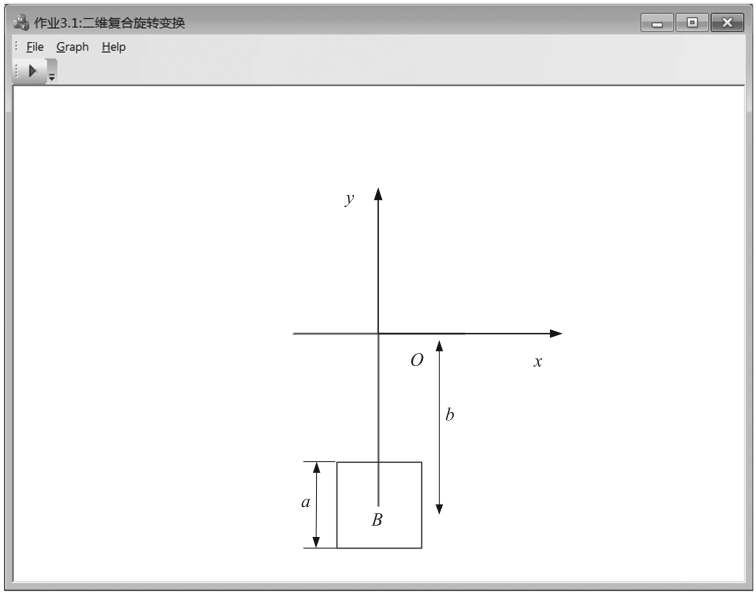


图 3-14　二维复合旋转变换

程序运行结果
习题 3.1

3.2　将建模坐标系原点设在立方体中心点，并将立方体边长减半，得到小立方体。在世界坐标系中，使用 8 个小立方体构造如图 3-15 所示的八叉树。试基于三维变换类 CTransform3，编程实现八叉树绕坐标原点的旋转变换，效果如图 3-16 所示。

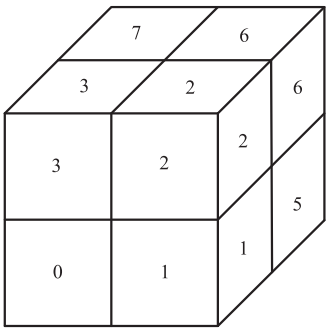


图 3-15　八叉树

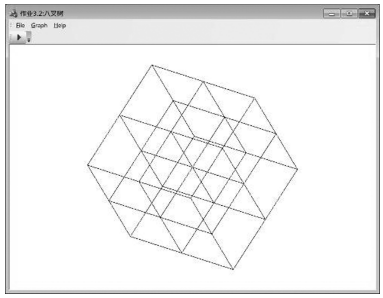
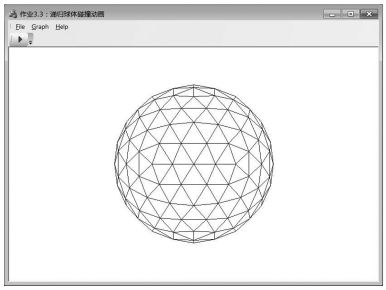


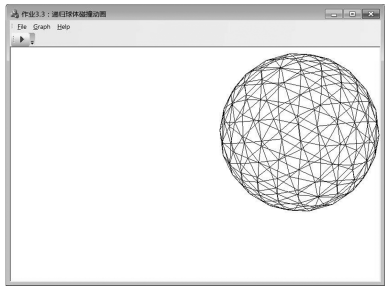
图 3-16　习题 2 效果图

程序运行结果
习题 3.2

3.3　基于正二十面体模型生成递归球体。该球体一边绕 z 轴逆时针方向旋转,一边在窗口客户区内移动。碰撞客户区边界后,球体改变运动方向。试制作递归球体运动的动画,效果如图 3-17 所示。



(a) 状态1



(b) 状态2

图 3-17　球体碰撞动画

程序运行结果
习题 3.3