第3章　矩阵变换

本章讲解三维变换的基础知识，让读者了解矩阵变换的数学表示以及在图形学中的意义。

目标：

- 学习矩阵的基本变换：平移、缩放和旋转。
- 学习 XNA 库中矩阵的变换函数。
- 学习几何变换的组合。

3.1　基本变换

本节介绍 3D 图形学中最为基础的几种变换：平移、旋转和缩放。一种变换对应一类矩阵，这些矩阵各有其鲜明的特点。

首先需要知道的是，本书使用 4×4 矩阵来表示三维变换，基本的矩阵变换形式如下：

$$\boldsymbol{pM} = (x,y,z,w)\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = (x',y',z',w') = \boldsymbol{p}'$$

上式表示对坐标 $\boldsymbol{p}$ 进行 $\boldsymbol{M}$ 变换后得到坐标 $\boldsymbol{p}'$，其中矩阵 $\boldsymbol{M}$ 叫做变换矩阵，$\boldsymbol{p}$ 点或向量为齐次坐标。从上述公式可以看到，$\boldsymbol{M}$ 为一个 4×4 矩阵，那么为什么不用 3×3 矩阵来进行三维变换呢？因为三维矩阵的表达能力有限，它不能用来表示平移变换、透视投影变换等。而 4×4 矩阵能够很好地使用上述形式表示包括平移、透视投影在内的几乎所有常用的变换。

3.1.1　平移变换

平移变换用于将点从一个地方移到另一个地方，假设需要将点 $\boldsymbol{p}=(x,y,z,1)$ 移动 $\boldsymbol{u}=(u_x,u_y,u_z)$ 到 $\boldsymbol{p}'=(x+u_x,y+u_y,z+u_z,1)$，如图 3.1 所示。

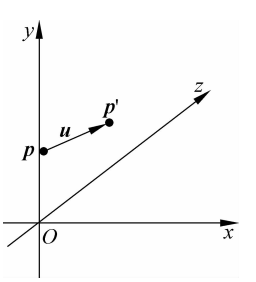

图 3.1　平移的几何描述

该平移过程可以使用如下变换矩阵完成：

$$\boldsymbol{M}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ u_x & u_y & u_z & 1 \end{bmatrix}$$

即：

$$\boldsymbol{pM}_t=(x,y,z,1)\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\u_x&u_y&u_z&1\end{bmatrix}=(x+u_x,y+u_y,z+u_z,1)=\boldsymbol{p}+\boldsymbol{u}=\boldsymbol{p}'$$

读者可以自行验证上述等式的正确性。那么，很明显，平移变换矩阵的规律就是将平移向量 $\boldsymbol{u}=\overrightarrow{\boldsymbol{pp}'}=(u_x,u_y,u_z)$ 的 3 个分量分别放入 4×4 单位矩阵的前三列的末尾，即可得到相应的变换矩阵。

平移变换矩阵的逆矩阵可以通过几何平移过程的逆过程得到，从 $\boldsymbol{p}$ 移动到 $\boldsymbol{p}'$ 需要移动 $\boldsymbol{u}=(u_x,u_y,u_z)$，那么从 $\boldsymbol{p}'$ 移动到 $\boldsymbol{p}$，则需要移动 $\boldsymbol{v}=(-u_x,-u_y,-u_z)$，便可以得到 $\boldsymbol{M}_t$ 的逆矩阵：

$$\boldsymbol{M}_t^{-1}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\-u_x&-u_y&-u_z&1\end{bmatrix}$$

即 $\boldsymbol{p}'\boldsymbol{M}_t^{-1}=\boldsymbol{p}$。那么，求平移变换矩阵的逆矩阵的规律就是，将 $\boldsymbol{M}_t$ 中 u_x,u_y,u_z 的符号求反即可。

3.1.2 旋转变换

经常需要将几何体绕某一轴旋转某一角度。绕任意轴旋转的矩阵比较复杂，故而首先介绍绕 x 轴、y 轴、z 轴旋转的变换矩阵，绕这三个轴旋转角度 θ 的变换矩阵分别为

$$\boldsymbol{M}_x=\begin{bmatrix}1&0&0&0\\0&\cos\theta&\sin\theta&0\\0&-\sin\theta&\cos\theta&0\\0&0&0&1\end{bmatrix}$$

$$\boldsymbol{M}_y=\begin{bmatrix}\cos\theta&0&-\sin\theta&0\\0&1&0&0\\\sin\theta&0&\cos\theta&0\\0&0&0&1\end{bmatrix}$$

$$\boldsymbol{M}_z=\begin{bmatrix}\cos\theta&\sin\theta&0&0\\-\sin\theta&\cos\theta&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

这几个矩阵都是很有规律的，不过读者并不需要死记硬背，只需要学会简单的推导即可。拿绕 z 轴旋转的矩阵举例，假设需要将点 $\boldsymbol{p}(x,y,z)$ 绕 z 轴旋转 θ（如图 3.2 所示），从几何学的角度分析，可以很容易地得到旋转后点的位置（常规坐标）为 $\boldsymbol{p}'=((x\cos\theta-y\sin\theta),(x\sin\theta+y\cos\theta),z)$。

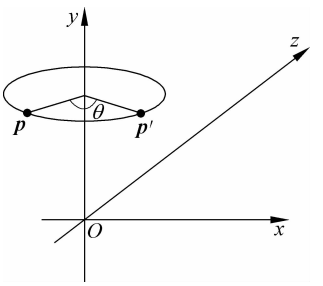

图 3.2　将 $\boldsymbol{p}$ 绕 z 轴旋转 θ

然后，结合简单的矩阵乘法知识，将上式中对应的系数放

到矩阵中对应的位置即可得到旋转矩阵 $\boldsymbol{M}_z$，使得 $\boldsymbol{p}'=\boldsymbol{p}\boldsymbol{M}_z$。

注意：关于旋转的方向，在左手坐标系中，正的旋转方向为从对应轴的正方向（从正半轴朝向负半轴看）看过去，绕逆时针方向旋转的方向为正。在右手坐标系中，以同样的方式看过去，绕顺时针方向旋转的方向为正。

从几何学的角度可以得出，求旋转变换矩阵的逆矩阵，只需要将旋转矩阵中 θ 的符号反向即可，因为将某个几何体绕某个轴旋转 θ 的逆过程就是将该几何体绕这个轴逆向旋转 θ，也就是旋转 $-\theta$。比如有：

$$\boldsymbol{M}_z^{-1}=\begin{bmatrix}\cos(-\theta) & \sin(-\theta) & 0 & 0\\ -\sin(-\theta) & \cos(-\theta) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

注意：旋转矩阵有一个很有用的性质，即旋转矩阵都是正交矩阵，而对于正交矩阵，有 $\boldsymbol{M}^T=\boldsymbol{M}^{-1}$，正交矩阵的逆矩阵和转置矩阵相同，这就提供了另一种求旋转矩阵的逆矩阵的方法，即旋转矩阵的逆矩阵即是它的转置矩阵。另外，关于正交矩阵的概念，感兴趣的读者可以参考线性代数相关书籍。

上面介绍了绕三个坐标轴旋转，而绕任意旋转轴旋转 θ 可以由绕坐标轴的旋转组合而成。

旋转轴方向矢量：$\boldsymbol{v}=\boldsymbol{p}_2-\boldsymbol{p}_1=(x_2-x_1,y_2-y_1,z_2-z_1)$。

旋转轴单位向量：$\boldsymbol{u}=\dfrac{\boldsymbol{v}}{\|\boldsymbol{v}\|}=(a,b,c)$，其中 a、b、c 分别是旋转轴的三个方向余弦：$a=\dfrac{x_2-x_1}{\|\boldsymbol{v}\|}$，$b=\dfrac{y_2-y_1}{\|\boldsymbol{v}\|}$，$c=\dfrac{z_2-z_1}{\|\boldsymbol{v}\|}$。

那么要得到绕任意轴旋转 θ 角的变换矩阵，有以下 5 个基本步骤。

（1）平移物体，使旋转轴通过坐标原点。即将 $\boldsymbol{p}_1$ 点平移到坐标原点，平移矩阵 $\boldsymbol{T}$ 为

$$\boldsymbol{T}=\begin{bmatrix}1 & 0 & 0 & -x_1\\ 0 & 1 & 0 & -y_1\\ 0 & 0 & 1 & -z_1\\ 0 & 0 & 0 & 1\end{bmatrix}。$$

（2）旋转物体使得旋转轴与某一坐标轴重合。

① 绕 x 轴旋转，使旋转轴单位向量变换到 xz 平面上。

确定旋转角 α：相当于向量 $\boldsymbol{u}$ 在 yoz 平面的投影 $\boldsymbol{u}'$ 与 z 轴的夹角。

$\boldsymbol{u}=(a,b,c)$，那么 $\boldsymbol{u}$ 在 yoz 平面上的投影向量为 $\boldsymbol{u}'=(0,b,c)$。则

$\cos\alpha=\dfrac{\boldsymbol{u}'\cdot\boldsymbol{u}}{\|\boldsymbol{u}'\|\cdot\|\boldsymbol{u}\|}$，$\sin\alpha=\dfrac{b}{d}$，其中 $d=\sqrt{b^2+c^2}$。

那么，绕 x 轴的旋转矩阵为

$$\boldsymbol{R}_x(\alpha)=\begin{bmatrix}1 & 0 & 0 & -x_1\\ 0 & c/d & -b/d & -y_1\\ 0 & b/d & c/d & -z_1\\ 0 & 0 & 0 & 1\end{bmatrix}$$

② 绕 y 轴旋转，使旋转轴单位向量变换到与 z 轴重合。

确定旋转角 β：相当于向量 $\boldsymbol{u}$ 在 xoz 平面的投影 $\boldsymbol{u}''$ 与 z 轴的夹角。$\boldsymbol{u}''=(a,0,d)$，那么

$\sin\beta=-a$，$\cos\beta=d$。

绕 x 轴的旋转矩阵为：

$$\boldsymbol{R}_y(\beta)=\begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(3) 绕坐标轴完成指定旋转。

将物体绕 z 轴旋转 θ 角，旋转矩阵为

$$\boldsymbol{R}_z(\theta)=\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(4) 利用逆旋转使旋转轴回到原始方向。

旋转矩阵为 $\boldsymbol{R}_x^{-1}(\alpha)$ 和 $\boldsymbol{R}_y^{-1}(\beta)$。

(5) 利用逆平移使旋转轴回到原始位置。

转换矩阵为 $\boldsymbol{T}^{-1}$。

那么，就可以得到绕任意轴旋转 θ 角的变换矩阵为

$$\boldsymbol{M}(\theta)=\boldsymbol{T}^{-1}\cdot\boldsymbol{R}_x^{-1}(\alpha)\cdot\boldsymbol{R}_y^{-1}(\beta)\cdot\boldsymbol{R}_z(\theta)\cdot\boldsymbol{R}_y(\beta)\cdot\boldsymbol{R}_x(\alpha)\cdot\boldsymbol{T}$$

3.1.3 缩放变换

缩放变换用来改变几何体的大小，如图 3.3 所示。

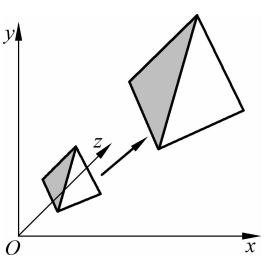

图 3.3 缩放的几何描述

将点 $\boldsymbol{p}(x,y,z)$ 沿 x 轴、y 轴、z 轴分别缩放 s_x，s_y，s_z 的变换矩阵为

$$\boldsymbol{S}=\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

使用该变换矩阵得到变换后点的位置为

$$\boldsymbol{p}'=\boldsymbol{pS}=(x,y,z,1)\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=(s_xx,s_yy,s_zz,1)$$

即缩放后点的位置为 $\boldsymbol{p}'(s_xx,s_yy,s_zz)$，显然，这是正确的。有一点需要说明，这种缩放是相对于原点进行点的位置的缩放，对于几何体的缩放实际上是对组成几何体的网格中的顶点分别进行缩放。

缩放矩阵的逆矩阵的计算也是非常方便的，只需要将各个轴的缩放因子分别取倒数即可

$$\boldsymbol{S}^{-1}=\begin{bmatrix} 1/s_x & 0 & 0 & 0 \\ 0 & 1/s_y & 0 & 0 \\ 0 & 0 & 1/s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2 XNA 矩阵变换函数

XNA 数学库提供了一些矩阵变换支持函数，这些函数使用方便并且效率很高，此处列出了这些变换函数的原型并加以说明：

1. 平移变换矩阵

根据 $x/y/z$ 轴三个方向的平移量创建平移矩阵。

```
XMMATRIX  XMMatrixTranslation(FLOAT OffsetX, FLOAT OffsetY, FLOAT OffsetZ);
```

根据平移向量创建平移矩阵。

```
XMMATRIX  XMMatrixTranslationFromVector(FXMVECTOR Offset);
```

2. 缩放变换矩阵

根据 $x/y/z$ 轴的缩放系数创建缩放矩阵。

```
XMMATRIX  XMMatrixScaling(FLOAT ScaleX, FLOAT ScaleY, FLOAT ScaleZ);
```

根据缩放向量(描述了 3 个轴各自的缩放系数)创建缩放矩阵。

```
XMMATRIX  XMMatrixScalingFromVector(FXMVECTOR Scale);
```

3. 旋转变换矩阵

根据旋转角度创建绕 x 轴旋转的矩阵。

```
XMMATRIX  XMMatrixRotationX(FLOAT Angle);
```

根据旋转角度创建绕 y 轴旋转的矩阵。

```
XMMATRIX  XMMatrixRotationY(FLOAT Angle);
```

根据旋转角度创建绕 z 轴旋转的矩阵。

```
XMMATRIX  XMMatrixRotationZ(FLOAT Angle);
```

根据欧拉角(pitch，yaw，roll)创建旋转矩阵。

```
XMMATRIX  XMMatrixRotationRollPitchYaw(FLOAT Pitch, FLOAT Yaw, FLOAT Roll);
```

根据记录了欧拉角(pitch，yaw，roll)的向量创建旋转矩阵。

```
XMMATRIX  XMMatrixRotationRollPitchYawFromVector(FXMVECTOR Angles);
```

根据法线和角度创建绕任意方向旋转的旋转矩阵。

```
XMMATRIX  XMMatrixRotationNormal(FXMVECTOR NormalAxis, FLOAT Angle);
```

根据轴向量和角度创建绕任意轴旋转的旋转矩阵。

```
XMMATRIX  XMMatrixRotationAxis(FXMVECTOR Axis, FLOAT Angle);
```

根据 4 元数创建旋转矩阵。

```
XMMATRIX  XMMatrixRotationQuaternion(FXMVECTOR Quaternion);
```

XMMatrixRotationAxis 和 XMMatrixRotationNormal 函数作用相同，二者的区别在于前者只需要输入任意的旋转轴向量，后者则要求输入规范化的旋转轴向量，后者比前者速度快。

以上函数能够根据给定条件创建变换矩阵，另外还有一个很重要的函数就是前面介绍过的向量-矩阵相乘函数：

```
XMVECTOR  XMVector * Transform(FXMVECTOR V, CXMMATRIX M);
```

其中的 * 可以为 2、3、或 4，此函数负责将变换矩阵应用到对应的点或向量。

3.3 几何变换的组合

使用 4×4 矩阵的形式表示所有变换的优点是：可以借助于矩阵乘法将多个相同类型或不同类型的变换组合成为一个变换矩阵。请看下面这段代码：

```
XMVECTOR u = XMVectorSet(3.0f, 4.0f, 1.0f, 1.0f);
XMMATRIX T, R, S;

//创建沿三个坐标轴各自缩放 2.0 的缩放矩阵
S = XMMatrixScaling(2.0f, 2.0f, 2.0f);
//创建绕 z 轴旋转 π/2 的旋转矩阵
R = XMMatrixRotationZ(XM_PI/2);
//创建沿三个坐标轴分别平移 2.0、0.0、4.0 的平移矩阵
T = XMMatrixTranslation(2.0f, 0.0f, 4.0f);

//对点 u 应用缩放变换
u = XMVector4Transform(u, S);
//对点 u 应用旋转变换
u = XMVector4Transform(u, R);
//对点 u 应用平移变换
u = XMVector4Transform(u, T);
```

以上代码的功能是，对点 $\boldsymbol{u}$ 沿三个坐标轴方向放大 2 倍，然后绕 z 轴旋转 $\pi/2$ 弧度，再沿三个坐标轴分别移动 2，0，4 个单位。同时，将上面代码中最后三行代码改为如下代码，也可以实现同样的逻辑：

```
//组合各种变换
XMMATRIX M = S * R * T;
//对点 u 直接应用组合变换
u = XMVector4Transform(u, M);
```

对比以上两种实现方式，就可以看出用矩阵乘法的好处了。对于第一种实现方式，分别对点 $\boldsymbol{u}$ 进行缩放、旋转、平移操作，共需 3 次向量-矩阵乘法。而对于第二种实现方式，则是首先将所有变换组合为一个变换，然后一次性应用到点 $\boldsymbol{u}$，需两次矩阵乘法和一次向量-矩阵乘法。对一个点进行变换还看不出组合变换的优势，现在假设对一个三维物体进行上述变换过程，此物体包含 1000 个点。如果采用第一种方式，则需要按顺序对所有点分别进行 3 次变换，共需要 3000 次向量-矩阵乘法。而按照第二种方式，首先一次性计算出组合变换矩阵，然后分别应用到所有点，则只需要 1000 次向量-矩阵乘法外加两次矩阵乘法。孰优孰劣，可见一斑。

3.4 本章小结

本章讲解了三维变换所需的矩阵基础知识，并且了解 XNA 库中相应的变换函数的用法，具体包括以下知识点：

- 向量-矩阵乘法是矩阵变换的基础，在三维坐标系中，使用点或向量的齐次坐标乘以一个 4×4 矩阵来表示三维变换，该 4×4 矩阵叫做变换矩阵。
- 最基础的三类矩阵变换：平移、旋转、缩放，它们的变换矩阵各具规律，读者需学会简单的推导。
- XNA 数学库提供了支持平移、旋转、缩放变换的相应函数。
- 因为矩阵乘法符合结合律，所以可以对矩阵变换进行组合，即$(\boldsymbol{pS})\boldsymbol{R})\boldsymbol{T}=\boldsymbol{p}(\boldsymbol{SRT})$，因此，可以将对一个对象的多种变换，组合成一个变换矩阵，然后一次性应用到该对象上。

第二部分
Direct3D基础

此部分将对 Direct3D 及其图形渲染管线进行简要说明，并对 HLSL 绘制语言进行介绍。最后对 3D 场景绘制中的一些基本主题进行描述。