**3 变换**

**线性变换**

**What:**

我们称t为线性变换，当且仅当函数有下列性质：

(u + v) = t(u) + t(v)

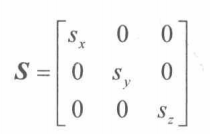
(ku) = kt(u)

**Why:**

缩放和旋转

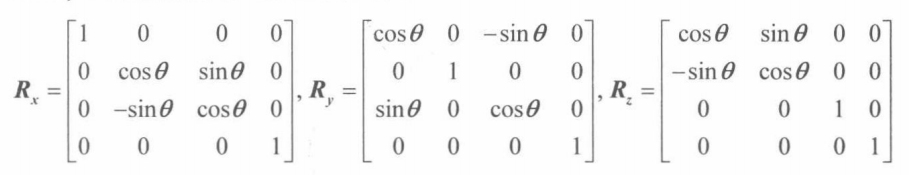
**How:**

**缩放矩阵**



**旋转矩阵**

旋转矩阵有个有趣的性质：每个行向量都为单位长度且两两正交。也就是说，这些行向量都是规范正交的。



**仿射变换**

**What:**

仿射变换是由一个线性变换与一个平移变换组合而成的

**Why:**

线性变换不能表示移动变换。

**How:**

**齐次坐标**

**What：**

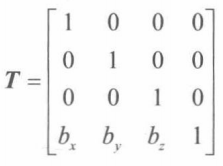
我们可以方便地对点和向量进行统一处理。(x,y,z,0)表示向量，(x,y,z,1)表示点。

**Why：**

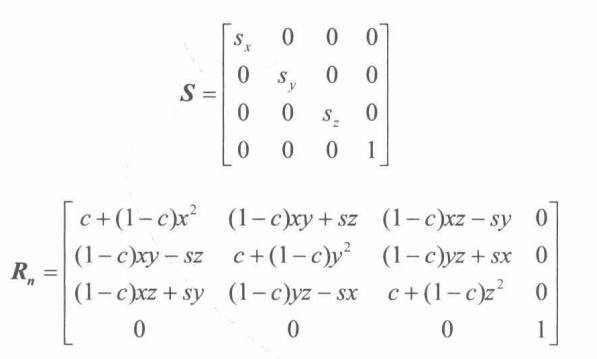
改进仿射变换

**How：**

**平移矩阵**



**缩放和旋转的仿射矩阵**



**变换的顺序**

缩放，旋转，移动

**坐标变换**

**What：**

不同标架间的坐标的转换称为坐标变换。

**Why：**

**How：**

向量的坐标变换

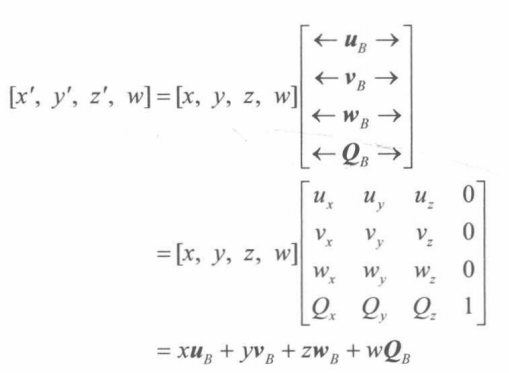
pB = xu + yv + zw

其中u，v和w分别是指向标架A中x轴，y轴和z轴正方向上的单位向量。

点的坐标变换

pB = xu + yv + zw + Q

坐标变换的矩阵表示



**DirectXMath库提供的变换函数**

**What：**

**Why：**

**How：**

// 构建一个缩放矩阵:

XMMATRIX XM\_CALLCONV XMMatrixScaling(float ScaleX, float ScaleY, float ScaleZ);

// 用一个3D向量中的分量来构建缩放矩阵:

XMMATRIX XM\_CALLCONV XMMatrixScalingFromVector(FXMVECTOR Scale);

// 构建一个绕x轴旋转的矩阵Rx:

XMMATRIX XM\_CALLCONV XMMatrixRotationX(float Angle);

// 构建一个绕任意轴旋转的矩阵Rn:

XMMATRIX XM\_CALLCONV XMMatrixRotationAxis(FXMVECTOR Axis,float Angle);

// 构建一个平移矩阵:

XMMATRIX XM\_CALLCONV XMMatrixTranslation(float OffsetX, float OffsetY, float OffsetZ);

// 用一个3D向量中的分量来构建平移矩阵:

XMMATRIX XM\_CALLCONV XMMatrixTranslationFromVector(FXMVECTOR Offset);

// 计算向量与矩阵的乘积vM，此函数为针对点的变换，即总是默认令vw = 1:

XMVECTOR XM\_CALLCONV XMVector3TransformCoord(FXMVECTOR V,CXMMATRIX M);

// 计算向量与矩阵的乘积vM，此函数为针对向量的变换，即总是默认令vw = 0:

XMVECTOR XM\_CALLCONV XMVector3TransformNormal(FXMVECTOR V,CXMMATRIX M);