1 向量代数

* 1. **向量**

向量是一种兼具大小和方向的量。

向量的绘制位置之于其自身是无足轻重的。平移向量完全不改变它的几何意义。

* + 1. 向量与坐标系

同一个向量在不同的坐标系中有着不同的坐标表示。

* + 1. **左手坐标系与右手坐标系**

Direct3D采用左手坐标系。

如果我们伸出左手，并拢手指，假设它们指向的是x轴的正方向，再弯曲四指指向y轴正方向，则最后伸直拇指的方向大约就是z轴的正方向。

* + 1. 向量的基本运算
  1. 长度与单位向量
  2. **点积**

点积是一种计算结果为标量值的向量乘法运算，因此有时也称为标量积。设向量u = ( ux, uy, uz)，v = ( vx, vy, vz)，则点积的定义为：

u·v = uxvx + uyvy + uzvz

可见，点积就是向量间对应分量的乘积之和。

两个向量点积的几何关系：

u·v = |u||v|cosθ (0 ≤ θ ≤ π)

两向量的点积为两向量夹角的余弦值乘以这两个向量的模。如果向量u和向量v都是单位向量，那么u·v就等于两向量夹角的余弦值，u·v = cosθ。

p在向量v上的正交投影，通常表示为：p = projn(v)，垂直投影表示为w = perpn(v)。

如果n为单位向量，向量v在n上的投影：p = (v·n) n。

n不是单位向量，p = projn(v) = (v· n / |n|2) n

正交化

如果向量集{v0,……,vn-1}中的每个向量都是互相正交且皆具单位长度，那么我们就称此集合是规范正交的。

对于具有n个向量的一般集合{v0,……,vn-1}而言，为了将其正交化为规范正交集，我们就要使用格拉姆--施密特正交化方法进行处理。

基本步骤：设w0 = v0

对于1 ≤ i ≤ n-1，令

规范化步骤：令wi = wi / |wi|

直观上来说，在将给定集合内的向量vi添加到规范正交集中时，我们需要令vi减去它在现有规范正交级中其他向量{w0,……,wi-1}方向上的分量(投影)。

* 1. **叉积**

叉积的计算结果亦为向量。此外，只有3D向量的叉积有定义，不存在2D向量叉积。假设3D向量u和v的叉积得到的是另一个向量w，则w与向量u、v彼此正交。

叉积的计算方法为：

w = u x v = ( uyvz - uzvy, uzvx - uxvz, uxvy - uyvx)

如果伸出左手，使并拢的左手手指指向向量u的方向，再以0 ≤ θ ≤ π的角度弯曲四指，使之指向向量v的方向，那么最后伸直的大拇指约指向的即为w = u x v的方向。这就是所谓的左手拇指法则(左手定则)。

一般来说u x v ≠ v x u，我们同时能够证明 u x v = - v x u。

* + 1. 2D向量的伪叉积
    2. 通过叉积来进行正交化处理

对于3D情况：

1. 令w0 = v0 / |w0|；
2. 令w2 = w0 x v1 / |w0 x v0|；
3. 令w1 = w2 x w0；

此时，向量集{w0,w1,w2}是规范正交的。

* 1. 点
  2. 利用DirectXMath库进行向量运算

DirectXMath数学库采用了SIMD流指令扩展2指令集，借助128位宽的单指令多数据寄存器，利用一条SIMD指令即可同事对4个32位浮点数或整数进行运算。

u + v = ( ux + vx, uy + vy, uz + vz)按普通的计算方式只能对分量逐个相加。通过SIMD技术，仅用一条SIMD加法指令来取代4条普通的标量指令。

* + 1. 向量类型

在DirectXMath库中，核心的向量类型是XMVECTOR，它将被映射到SIMD硬件寄存器。通过SIMD指令的配合，利用这种具有128位的类型能一次性处理4个32位的浮点数。在向量计算过程中，必须通过此类型才可充分利用SIMD技术。

XMVECTOR类型的数据需要按16字节对齐，这对于局部变量和全局变量而言都是自动实现的。至于类中的数据成员，建议分别使用XMFLOAT2、XMFLOAT3和XMFLOAT4类型来加以代替。

但是，如果直接把上述这些类型用于计算，却依然不能充分发挥出SIMD技术的搞笑特性。为此，我们还需要将这些类型的实例转换为SMVECTOR类型。

总结一下：

1. 局部变量或全局变量用XMVECTOR类型；
2. 对于类中的数据成员，使用XMFLOAT2、XMFLOAT3和XMFLOAT4类型；
3. 在运算之前，通过加载函数将XMFLOATn类型转换为XMVECTOR类型；
4. 用XMVECTOR实例来进行运算；
5. 通过存储函数将XMVECTOR类型转换为XMFLOATn类型；
   * 1. 加载方法和存储方法

XMVECTOR XM\_CALLCONV XMLoadFloat3(const XMFLOAT3 \*pSource);

void XM\_CALLCONV XMStoreFloat3(XMFLOAT3 \*pDestination, FXMVECTOR v);

存取一个分量的方法：

float XM\_CALLCONV XMVectorGetX(FXMVECTOR V);

XMVECTOR XM\_CALLCONV XMVectorSetX(FXMVECTOR V, float x);

* + 1. 参数的传递

传递XMVECTOR参数的规则如下：

1. 前3个XMVECTOR参数应当用FXMVECTOR；
2. 第4个XMVECTOR参数应当用GXMVECTOR；
3. 第5，6个XMVECTOR参数应当用类型HXMVECTOR；
4. 其余的XMVECTOR参数应用用类型CXMVECTOR；

在编写构造函数时，前3个XMVECTOR参数用FXMVECTOR**类型，其余**XMVECTOR参数用CXMVECTOR类型。另外，对于构造函数不用使用XM\_CALLCONV注解。

* + 1. 常向量

XMVECTOR类型的常量实例应当用XMVECTORF32类型来表示。

* + 1. 重载运算符
    2. 杂项
    3. Setter函数
    4. 向量函数
    5. 浮点数误差

XMVector3NearEqual近视相等函数。