1. **向量代数**

**1，左手坐标系和右手坐标系**

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

**2，点积**

点积的大小：a\*b=|a||b|cosθ

点积的应用：

1，求解向量间的夹角

2，正交投影

**3，叉积**

叉积的大小：|a×b|=|a||b|sinθ

叉积结果的方向：两个3D向量u与v的叉积得到的是：既正交于u也正交于v的向量w。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略指向的即为w=uxv的方向。

叉积的应用：

1. 判断点是否在三角形内
2. 计算平行四边形面积

**4，规范正交化**

定义：如果向量集中的每个向量都是互相正交且皆具单位长度。

正交化方法：

1. 格拉姆-施密特正交化

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{w0,…,wn-1}的基本步骤：

设w0=v0

对于1≤i≤n-1，令wi=vi -

对于wi进行规范化

1. 通过叉积来进行正交化处理

令w0=v0/|v0|

令w2=w0xv1/|w0xw1|

令w1=w2xw0

**5，利用DirectXMath库进行向量运算**

**5.1 XMVECTOR概念**

SIMD技术：借助128位宽的单指令多数据寄存器，利用一条SIMD指令即可同时对4个32位浮点数或整数进行运算。向量类型XMVECTOR将被映射到SIMD硬件寄存器中。

头文件：<DirectXMath.h>和<DirectXPackedVector.h>

命名空间：DirectX和DirectX::PackedVector

**5.2 XMVECTOR与XMFLOATn**

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或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。

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

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

float XM\_CALLCONV XMVectorGetX(FXMVECTOR V);

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

**5.2 XMVECTOR参数用法**

**传递 XMVECTOR参数的规则如下：**

1，前3个XMVECTOR参数应当用类型FXMVECTOR；

2，第4个XMVECTOR参数应当用类型GXMVECTOR；

3，第5，6个XMVECTOR参数应当用类型HXMVECTOR；

4，其余的XMVECTOR参数应当用类型CXMVECTOR。

在32位的windows系统上，编译器将根据\_\_fastcall调用约定将前3个XMVECTOR参数传递到寄存器中，而把其余参数都存在栈上。

typedef const XMVECTOR FXMVECTOR;

typedef const XMVECTOR& GXMVECTOR;

typedef const XMVECTOR& HXMVECTOR;

typedef const XMVECTOR& CXMVECTOR;

在32位的windows系统上，编译器将根据\_\_vectorcall调用约定将前6个XMVECTOR参数传递到寄存器中，而把其余参数都存在栈上。

typedef const XMVECTOR FXMVECTOR;

typedef const XMVECTOR GXMVECTOR;

typedef const XMVECTOR HXMVECTOR;

typedef const XMVECTOR& CXMVECTOR;

在编写构造函数时，前3个XMVECTOR参数用FXMVECTOR类型，其余XMVECTOR参数则用CXMVECTOR类型。

**5.3 常向量**

XMVECTOR类型的常量实例应当用XMVECTORF32类型表示。

static const XMVECTORF32 g\_vHalfVector = { 0.5f, 0.5f, 0.5f, 0.5f}

XMVECTORF32是一种按16字节对齐的结构体，数学库中还提供了将它转换至XMVECTOR类型的运算符。也可以通过XMVECTORU32类型来创建由整型数据构成的XMVECTOR常向量。

**6，其他库函数**

**6.1 Setter函数**

XMVECTOR XM\_CALLCONV XMVectorZero();

XMVECTOR XM\_CALLCONV XMVectorSplatOne();

XMVECTOR XM\_CALLCONV XMVectorSet(float x, float y, float z, float w);

XMVECTOR XM\_CALLCONV XMVectorReplicate(float Value);

XMVECTOR XM\_CALLCONV XMVectorSplatX(FXMVECTOR V);

XMVECTOR XM\_CALLCONV XMVectorSplatY(FXMVECTOR V);

XMVECTOR XM\_CALLCONV XMVectorSplatZ(FXMVECTOR V);

**6.2 向量函数**

XMVECTOR XM\_CALLCONV XMVector3Length(FXMVECTOR V);

XMVECTOR XM\_CALLCONV XMVector3LengthSq(FXMVECTOR V);

XMVECTOR XM\_CALLCONV XMVector3Dot(FXMVECTOR V1, FXMVECTOR V2);

XMVECTOR XM\_CALLCONV XMVector3Cross(FXMVECTOR V1, FXMVECTOR V2);

XMVECTOR XM\_CALLCONV XMVector3Normalize(FXMVECTOR V);

XMVECTOR XM\_CALLCONV XMVector3Orthogonal(FXMVECTOR V);

XMVECTOR XM\_CALLCONV XMVector3AngleBetweenVectors(FXMVECTOR V1, FXMVECTOR V2);

void XM\_CALLCONV XMVector3Dot(XMVECTOR\* pParallel, XMVECTOR\* pPerpendicular,

FXMVECTOR v, FXMVECTOR Normal);

bool XM\_CALLCONV XMVector3Equal(FXMVECTOR V1, FXMVECTOR V2);

bool XM\_CALLCONV XMVector3NotEqual(FXMVECTOR V1, FXMVECTOR V2);

**6.3 浮点数误差**

XMFINLINE bool XM\_CALLCONV XMVector3NearEqual(FXMVECTOR U, FXMVECTOR V, FXMVECTOR Epsilon);