1. **向量代数**

**1，坐标系**

**左手坐标系的判断方法**

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**

**点积的大小公式**

点积的大小：a\*b=|a||b|cosθ

**点积的两个应用**

1，求解向量间的夹角

2，正交投影

**3，叉积**

**叉积的大小公式**

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。

**叉积的两个应用**

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**

**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。

**规范正交化的两种方法**

1，格拉姆-施密特正交化

2，通过叉积来进行正交化处理

**5，DirectXMath库**

**5.1 XMVECTOR**

**XMVECTOR的性能优势**

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。

**XMVECTOR的头文件和命名空间**

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

命名空间：DirectX和DirectX::PackedVector

**5.2 XMVECTOR与XMFLOATn**

**XMVECTOR和XMFLOATn的使用规则**

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。

**5.3 XMVECTOR参数用法**

**在编写普通函数时，传递 XMVECTOR参数的规则**

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。

**在编写构造函数时，传递 XMVECTOR参数的规则**

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。

**XMVECTOR类型的常量实例应当用什么类型表示？**

XMVECTOR类型的常量实例应当用XMVECTORF32类型表示。

**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 XMVECTOR与XMFLOATn**

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);

**6.4 浮点数误差**

XMFINLINE bool XM\_CALLCONV XMVector3NearEqual(FXMVECTOR U,

FXMVECTOR V, FXMVECTOR Epsilon);