三维向量

Mathf.Pow(名称.x,2)　x的平方
Mathf.Sqrt（）开平方

名称.magnitude　求名称的向量的模长

单位向量
v3 n01= 名称/名称.magnitude
v3 n02=名称.normalized　　获取向量的方向

t3.Transform(relativeDirection.normalized); 朝着relativeDirection.normalized的方向移动。

角度转换弧度　　弧度=角度*PI/180;*
*r1=d1*Mathf.PI/180;　r2=d1*Mathf.Deg2Rad;

弧度转化为角度　角度=弧度*180/PI;*
*d1=r1*180/PI;　d2=r1*Mathf.Rad2Deg;

1弧度=180度/pi　1角度/180度

反三角函数
反正弦，反余弦，反正切等函数的总称。
可用于根据两边长,计算角度。
公式:
反正弦arcsina/C=X;
反余弦arccosb / c= X ;
反正切arctana/b = X ;
API：　求角度
Mathf.Asin(float radian)
Mathf.Acos(float radian)
Mathf.Atan(float radian)

eg; 角度30，对边x，斜边10，临边y
x=mathf.Sin(30*mathf.dEB2Rad)*10;

三角函数
建立了直角三角形中角与边长比值的关系
可用于根据一边一 角，计算另外-边长。
公式:
正弦sinx=a/C;
余弦cOSx=b/C;
正切tanx=a/b;

API：

Mathf. Sin(float radian)

Mathf.Cos(float radian)

Mathf. Tan(float radian)

将自身坐标系转化为世界坐标系。

v3 worldPoint=transform.TranslatePoint(0,0,10);

点乘

又称点积或”内积“

公式:各分量乘积和

[x1,y1,z1] . [x2,y2,z2] = x1x2+y1y2+z1z2

几何意义: ab=|a| . |b| cos<a,b>

两个向量的单位向量相乘后再乘以二者夹角的余弦值.

API : float dot=Vector3.Dot(va, vb);

eg:求角度 float dot=aVector3.Dot(ti.transform.Normalized,t2.normalized);

angle =Mathf.Acos(dot)*Mathf.Rad2Deg;

叉乘

又称“叉积”或"外积”

公式: [x1,y1,z1] x [x2,y2,z2] =

[Y1*Z2-Z1★Y2，Z1*x2-x1*z2，x1y2-y1x2]

几何意义:结果为两个向量所组成面的垂直向量，模长为两向量模长乘积再乘夹角的正弦值.

脚本: Vector vector=Vector3. Cross (a, b);

欧拉角

使用三个角度来保存方位。

X与Z沿自身坐标系旋转，Y沿世界坐标系旋转.

API : Vector3 eulerAngle = this transform.eulerAngles;、、欧拉角 没有方向，大小 表示的旋转角度

Vector3 eulerAngle = this transform.position 有方向，大小 表示各个轴向上有位移

优点

仅使用三个数字表达方位，占用空间小。

沿坐标轴旋转的单位为角度，符合人的思考方式。

任意三个数字都是合法的,不存在不合法的欧拉角。

缺点

对于一个方位，存在多个欧拉角描述，因此无法判断多个欧

拉角代表的角位移是否相同。

例如:

角度0,5,0与角度0,365,0

角度0,-5,0与角度0,355,0

角度250,0,0与角度290,180,180
为了保证任意方位都只有独一无二的表示, Unity引|擎限制了角度范围,即沿X轴旋转限制在90到90之间，沿Y与Z轴旋转限制在0到360之间。

什么是四元数
●●Quaternion在3D图形学中代表旋转,由一一个三维向量(XNY/Z)和一个标量(W)组成。

旋转轴为V ,旋转弧度为0 ,如果使用四元数表示,则四个分量为
x=sin(θ /2)*V.x
y=sin(θ /2)*V.y
z=sin(θ /2)*V.z
w=cos(O /2)
X、Y、Z、W的取值范围是- 1到1。
API : Quaternion qt = this.transform.rotation;
优点:
避免万向节死锁
this.transform.rotation * = Quaternion.Euler(O, 1, 0);
可使物体沿自身坐标Y轴旋转
this.transform.Rotate(Vector3 eulerAngles)
内部就是使用四元数相乘实现

vect向量根据当前物体的旋转而旋转
vect=this.transform.rotation * new Vector3(0,0,10);
vect向量沿y轴旋转30度
vect=Quaternion.Euler(0,30,0)*vect;
vect向量移动到当前物体位置
vect = this .transform.position+vect;

magnitude 模长

Local Space --> World Space
transform.forward在世界坐标系中表示物体正前方。
transforn.right在世界坐标系中表示物体正右方。
transform.up在世界坐标系中表示物体正上方。
transform. TransformPoint
转换点,受变换组件位置、旋转和缩放影响。
transform. TransformDirection
转换方向，受变换组件旋转影响。

transform. TransformVector

转换向量，受变换组件旋转和缩放影响。