**Unity3D教程：计算机3D图形基础在Unity中的实现**

Posted on 2013年06月21日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 96 次

**Vector3:**

Vector3这词最早是谁发明的无从考证了，但是Unity3D里出现了这个名词的时候，我才意识到这个不是D3D和openGL才有的变量名，D3D里叫做 D3DXVECTOR3继承自\_D3DVECTOR结构体，其中\_D3DVECTOR结构体只有3个单精度数X，Y，Z。用于表示向量。

Unity3D中Vector3类定义（只写有用的）：

**属性：**

x，y，z：表示一个空间向量。

this：用于访问x，y，z三个数据使用数组的方式访问，比如[0][1][2]

normalized：返回单位化向量后的值（只读）。单位化向量是个很重要的概念，他常常是各种计算的基点，他保证了在不改变方向的前提下，使向量长度变为1。公式如下：

magnitude：返回向量长度值（只读），这词取得真怪，反正就是向量的模运算，得到向量的长度。公式就是x平方+y平方+z平方然后根号便是。这都不记得了赶紧回家面壁。

sqrmagnitude：返回向量的平方长度（只读）。上面那个不开根号就是了。

**方法：**

Scale(a:vector3,b:vector3):vertor3 缩放，返回a的每个坐标乘以b的相对应的每个坐标。注意他是静态函数。

Normalize() 向量化，注意magnitude向量长度会变为1.

类特有的一些属性（直接粘help的了）：

zero：Shorthand for writing Vector3(0, 0, 0)

one：Shorthand for writing Vector3(1, 1, 1)

forward：Shorthand for writing Vector3(0, 0, 1)

up：Shorthand for writing Vector3(0, 1, 0)

right：Shorthand for writing Vector3(1, 0, 0)

Maxtrix4\*4:

矩阵也是3D图形学一个重要的概念，在D3D里用的很平凡，但是U3D里好像都已经封装到各个Object上去了，所以很容易忽视掉，但不能忽视它的存在。在3D世界里，每个物体均有自身的世界矩阵，摄像机有摄像机矩阵，投影场景有projection矩阵，对顶点、向量、物体实施各种平移、旋转、缩放都是通过矩阵来完成的。计算机3D物体的标准4×4矩阵是这样定义的：(表示不出来矩阵大括号，请读者就当左4行的[和右4行的]当成一对大括号)

[ x, y ,z ,0] <- 物体的右方向向量

[ x, y ,z ,0] <- 物体的上方向向量

[ x, y ,z ,0] <- 物体的前方向向量

[ x, y ,z ,1] <- 物体所在的坐标

注意：前三行的坐标一定是互有90度的，（除非特殊情况，例如空间扭曲之类的=.=）因为他们是物体的三个坐标轴。第四列的 0 0 0 1 是为了补齐4\*4矩阵，为了计算方便而已。

矩阵一样有单位矩阵和0矩阵，

[1 0 0]

[0 1 0] 其他的我就不说了，在Unity里头是完全封装的（见Transform）

[0 0 1]

**Transform：**

这个就是U3D所封装的矩阵运算了，用于缩放，平移，还有定位(这个囧，他把矩阵给放这用了，所有物体都可以用transform类型来存放)。Transform所实现的功能不过就是物体矩阵的运算罢了，具体如下：

**Variables**

position：Vector3类型，物体位置，相对于世界坐标系的值。就是矩阵的最后一行的值。

localPosition：Vector3类型，物体相对于父元素的位置.

eulerAngles：Vector3类型，旋转后的欧拉角相对世界坐标的值。简单理解为各向量坐标所转动的值就好了。

localEulerAngles：Vector3类型，相对父元素的欧拉角。

right：Vector3类型，表示x轴（U3里红色的轴）方向的单位向量。

up：Vector3类型，表示y轴（U3里绿色的轴）方向的单位向量。

forward：Vector3类型，表示z轴（U3里蓝色的轴）方向的单位向量。

rotation：Quaternion类型(quatermion就是个四维数，比vector3多一个w变量表示旋转角度的)，世界坐标旋转。.

localRotation：Quaternion类型，本地坐标的旋转度数（相对父元素）。

localScale：Vector3类型，相对本地坐标系缩放（就是矩阵的乘法）

parent：Transform类型，他的父元素。

worldToLocalMatrix：Matrix4\*4类型，把一个点从世界坐标系的位置转换为本地坐标系位置。（终究得用矩阵，哈哈）只读

localToWorldMatrix：Matrix4\*4类型，把一个点从本地坐标系转换为世界坐标系。（只读）

root：Transform类型，返回最上层的那个Transform，就是他父亲的父亲的父亲的。（如果有的话）

childCount：int型，你子元素的数量。。。

lossyScale：Vector3类型，相对世界坐标系缩放。

Functions

Translate (translation : Vector3, relativeTo : Space=Space.Self) : void 平移物体，space可以是Self和World，这个不用说了吧

Rotate (eulerAngles : Vector3, relativeTo : Space = Space.Self) : void 绕欧拉角度的向量旋转

RotateAround (point : Vector3, axis :Vector3, angle : float) : void 点point按axis的轴旋转angle度

LookAt (target : Transform, worldUp :Vector3 ) : void 让这个物体的z轴面向target物体

TransformDirection (direction : Vector3) : Vector3 将direction从本地坐标转换为世界坐标

InverseTransformDirection (direction : Vector3): Vector3 将derection从世界坐标转换为本地坐标

TransformPoint (position :Vector3): Vector3 同理

InverseTransformPoint (position : Vector3) : Vector3

DetachChildren () : void 解除所有子对象

Find (name : string) : Transform 按名字寻找并返回子对象

IsChildOf (parent : Transform) : bool 看他是不是别人的子对象