[**Flash与3D编程探秘（二）- 静态长方体**](http://www.cnblogs.com/yangzhou1030/archive/2008/10/30/1322652.html)

日期：2008年10月

#### ****静态长方体****

那么接着上一节所讨论的内容，来制作一个静态的长方体，并且让它在屏幕上来回的运动。这个例子和上一节的例子非常的相似，同样主要关心3D空间。不 同的是，上一节我们利用了一个事先画好的小球，这一节改变方式，在程序执行时，计算出长方体的每个顶点的位置然后使用Flash的图形API绘制出一个长 方体，这样在一系列的绘制后，所看到的就是长方体移动的动画！

**移动的静态长方体**

#### ****动画制作步骤****

1. 在这一个例子中，需要发挥一下空间想象力去定位长方体的顶点。至于场景上的网格，我想大家可以先把它放在场景的底部，等动画完成后再把它向上移动到合适的位置。

2. 开始和以前一样，定义原点以及焦距。另外也要初始几个常量。长方体围绕y轴公转，R是长方体公转的半径。

var PI = 3.1415926535897932384626433832795;

var R = 100;

var origin = new Object();  
origin.x = stage.stageWidth/2;  
origin.y = stage.stageHeight/2;  
origin.z = 0;  
  
var focal\_length = 300;  
var angular\_velocity = PI/180;  
var \_angle\_xz = 0;

[复制代码](javascript:void(0);)

3. 接下来定义一个3D空间的点，长方体就围绕这个点进行公转。

var spin\_center = new Object();  
spin\_center.x = 0;  
spin\_center.y = 0;  
spin\_center.z = 100;

[复制代码](javascript:void(0);)

4. 创建一个场景舞台，用来盛放绘制的长方体。创建一个长方体的空Sprite并且把它添加到舞台上。

var scene = new Sprite();  
scene.x = origin.x;  
scene.y = origin.y  
this.addChild(scene);  
  
var box:Sprite = new Sprite();  
scene.addChild(box);

[复制代码](javascript:void(0);)

5. 下面需要一个函数来创建3D空间的点，那么下一次需要描述一个3D空间的点时，只需要执行这个函数，并且传入相应的x，y和z值即可。

function vertex(x, y, z):Object  
{  
    var point3d = new Object();  
    point3d.x = x;  
    point3d.y = y;  
    point3d.z = z;  
    return point3d;  
}

[复制代码](javascript:void(0);)

6. 你肯定猜到下一步要做的，那就是把一个3D空间的点转换成Flash能够理解的2D空间的点。转换的原理和第一节是一样的，这里就不解释了。

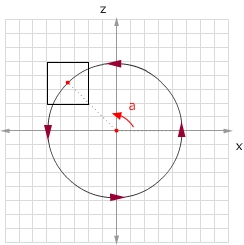
function convert(point3d, focal\_length):Object  
{  
    var point2d = new Object();  
    var scale\_ratio = focal\_length/(focal\_length+point3d.z);  
    point2d.x = point3d.x \* scale\_ratio;  
    point2d.y = point3d.y \* scale\_ratio;  
    return point2d;  
}

[复制代码](javascript:void(0);)

7. 下一步也是主要关心的循环函数。首先我想你明确这个函数要做的工作，那就是计算出长方体每个顶点的3D坐标，然后把它们转化成2D坐标，最后用Flash 的绘制API建立图形。每一次函数执行，把长方体围绕y轴旋转的角度增加。然后根据这个角度，便可以计算出长方体中心点（对角线的交 点）cente.x，center.y，center.z。很好，下面就可以确定这个长方体的每个点的坐标了。在这里我使用了一个边长为80正方体，你可 以更改参数调试出你喜欢的长方体。

#### ****注意****

再第一篇到第六篇文章里，物体的运动都是Frame Based的运动，不过长方体的运动你可以尝试改为Time Based的运动。由于文章旨在让读者体会3D的乐趣，因此关于Time Based和Frame Based的运动的区别暂不关系，不过这个课题将会在第七篇讨论。



**俯瞰长方体的移动**

那么紧接着，使用之前写的函数，把长方体的8个3D顶点转换成2D点。然后用Flash的绘制API绘制出来长方体。有一点想说明：假如你的正方体 的中心的x是a，正方体的边长是80，那么可以得到它的左前上顶点的x为(a-40)，它的右前上的x为(a+40)，依此类推；进而可以得出其他顶点的 坐标以及它们的y和z。需要注意的是，在这里是对空间中的点进行操作而不是对一个Sprite进行操作。

function move(e:Event):void  
{  
    var screen\_points = new Array();  
      
    \_angle\_xz += angular\_velocity;  
    if (\_angle\_xz > 360)  
    {  
        \_angle\_xz -= 360;  
    }  
  
    var center = new Object();  
    center.x = R\*Math.cos(\_angle\_xz) + spin\_center.x;  
    center.y = 0 + spin\_center.y;  
    center.z = R\*Math.sin(\_angle\_xz) + spin\_center.z;  
      
    var points = [  
            vertex(center.x-40, -40, center.z-40),      
            vertex(center.x+40, -40, center.z-40),          
            vertex(center.x+40, -40, center.z+40),  
            vertex(center.x-40, -40, center.z+40),  
                      
            vertex(center.x-40, 40, center.z-40),  
            vertex(center.x+40, 40, center.z-40),  
            vertex(center.x+40, 40, center.z+40),  
            vertex(center.x-40, 40, center.z+40)  
             ];  
  
    for (var i = 0; i < points.length; i++)  
    {  
        screen\_points[i] = convert(points[i], focal\_length);  
    }  
      
    with (box.graphics)  
    {  
        clear();  
        lineStyle(.5, 0x000000, 1);  
        moveTo(screen\_points[0].x, screen\_points[0].y);      
        lineTo(screen\_points[1].x, screen\_points[1].y);  
        lineTo(screen\_points[2].x, screen\_points[2].y);  
        lineTo(screen\_points[3].x, screen\_points[3].y);  
        lineTo(screen\_points[0].x, screen\_points[0].y);  
          
        moveTo(screen\_points[4].x, screen\_points[4].y);      
        lineTo(screen\_points[5].x, screen\_points[5].y);  
        lineTo(screen\_points[6].x, screen\_points[6].y);  
        lineTo(screen\_points[7].x, screen\_points[7].y);  
        lineTo(screen\_points[4].x, screen\_points[4].y);  
          
        moveTo(screen\_points[0].x, screen\_points[0].y);      
        lineTo(screen\_points[4].x, screen\_points[4].y);  
        moveTo(screen\_points[1].x, screen\_points[1].y);  
        lineTo(screen\_points[5].x, screen\_points[5].y);  
        moveTo(screen\_points[2].x, screen\_points[2].y);  
        lineTo(screen\_points[6].x, screen\_points[6].y);  
        moveTo(screen\_points[3].x, screen\_points[3].y);  
        lineTo(screen\_points[7].x, screen\_points[7].y);  
    }  
}

[复制代码](javascript:void(0);)

8. 最后在场景上添加一个循环执行事件。

this.addEventListener(Event.ENTER\_FRAME, move);

我希望你能够明确制作这个动画的思路：建立场景，创建3D物体，把顶点转换成为2D点，使用Flash绘制。So far so good? Great! 后面的文章将开始介绍摄像机以及相关的要素，希望你能跟上。

#### ****建议：****

在你的脑海中建立一个长方体应该不是很困难。我的方法是首先勾勒出正面，然后后面，上面，下面，左面和右面。如果你觉得用脑海想象比较困难的话，可以试着用笔和纸画出一个长方体，然后标出坐标。

#### ****建议：****

在Flash里，2D空间的原点在左上角(0,0)。在这上面的两节的例子中，在程序一开始就把3D空间的原点向右再向下移动，那么在屏幕上的映射面也就 到了屏幕的中间。我是刻意这样写的，好处在于如果你是新接触3D空间这个课题的话，想象物体在原点附近对你来说要容易一些。当然你可以尝试不移动原点，取 而代之移动物体，看看和原程序有什么不同。

#### ****一个简单的Particle Sytem的例子****

这里所说的Particle System，指得是大量的物体的聚集，我就把它叫做Particle Sytem。下面这个例子主要展示了物体的旋转运动，你应该可以看到一个大量分子（400个位置随机产生的星星）组成的旋转球体，可以使用WSAD移动你 的摄像机找到一个合适的观位置。Looks Cool！不过由于这个例子涉及到摄像机以及相关的内容，在这篇文章中我就不再详细解释了。不过不要担心，下一篇将介绍摄像机及其相关的内容，等你了解了 程序运行原理后再看这个例子，你一定会觉得简单了。

**旋转的星星球体，使用WASD键控制观看角度**