[**Flash与3D编程探秘（六）- 全方位旋转摄像机**](http://www.cnblogs.com/yangzhou1030/archive/2008/11/06/1328146.html)

日期：2008年11月

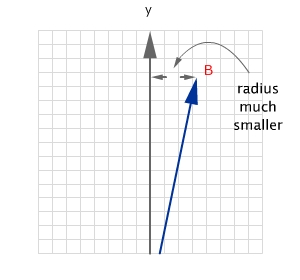
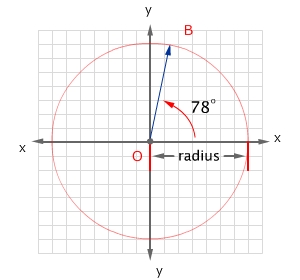
前面讨论过了如何横向旋转和移动摄像机，希望你已经完全理解，因为本文中的内容紧接着上一篇。回想一下，在前面制做的动画中，摄像机的旋转一直是围 绕着y轴（竖直向上的轴）旋转，然而现实中我们可以上下旋转摄像机，甚至可以把摄像机倾斜一定角度，这就提醒了大家还需要更深入的研究旋转这个课题。下面 几个动画演示了摄像机（简单的摄像机轮廓）3种旋转模式，从左到右分别是横向旋转，纵向旋转，倾斜。从3D空间角度来说，分别是沿y，x和z轴旋转。

**横向和纵向旋转摄像机**

**倾斜摄像机**

#### ****再介绍一些三角函数方程****

事情变得复杂起来了！不过请你还是保持头脑清醒，这一篇文章你的任务就是学会纵向旋转和倾斜摄像机。你也许会想，我已经学会了横向旋转摄像机，算法中使用 了一个panning的变量代表旋转角度，那么我再加两个变量，然后按顺序先沿x旋转，然后再沿y，最后z旋转不就好了。Well，这种想法很接近但却是 错的（不要把3D数学想的那么简单）。来看一下下面的两个图，还是以3D空间在2D平面上的投影举例，把线段OB沿着z轴（也就是2D平面上 的原点）旋转大约78度，可以看到OB的长度就是我们的旋转半径。那么接下来，把OB沿y轴旋转，看一下图中的旋转半径是多少？不难看出，旋转半径明显的 变小了。继续，如果你沿y旋转后，再打算沿x轴旋转，你还是会有同样“半径变化”的问题。



**先沿z轴旋转再试图沿y旋转**

在上面的例子中，如果想要保持旋转半径不变，那么一开始就不要有旋转角度。不过拿着摄像机左转右转，怎么可能保持角度不变！？因此你要在每一次摄像机旋转后，计算新的旋转半径。可是如果每一次都把旋转半径计算出来，那一定是很头疼！

于是人们聪明的想到了省力的方法。先来再看一下2D的三角函数（又是三角函数），根据旋转半径和旋转角度可以得到x和y：

x = Math.cos(angle)\*radius;  
y = Math.sin(angle)\*radius;

[复制代码](javascript:void(0);)

需要注意一点，上面的方程式旋转点为原点，并且之前旋转角度为0。如果之前就有旋转角度a，再旋转b，那么方程式就成了：

x = Math.cos(a+b)\*radius;  
y = Math.sin(a+b)\*radius;

[复制代码](javascript:void(0);)

看起来眼熟，但是不知道是什么了？别担心:

cos(a+b) = cos(a)\*cos(b) - sin(a)\*sin(b);  
sin(a+b) = sin(a)\*cos(b) + sin(b)\*cos(a);

[复制代码](javascript:void(0);)

把cos(a+b)和sin(a+b)带入上面的x和y求值方程我们就有：

x = radius\*cos(a)\*cos(b) - radius\*sin(a)\*sin(b);  
y = radius\*sin(a)\*cos(b) + radius\*sin(b)\*cos(a);

[复制代码](javascript:void(0);)

化简一下得到：

x = x\_before\*cos(b) - y\_before\*sin(b);  
y = x\_before\*sin(b) + y\_before\*cos(b);

[复制代码](javascript:void(0);)

这样，使用上面两个方程，不用担心你在其他平面（xz或者yz平面）的旋转角度，也不用每一次旋转后再去计算物体新的旋转半径了，只要关心旋转后的x和y，并且把它们作为下一次旋转的x\_before和y\_before，问题就解决了。下面我把相应的方程式写上：

围绕y轴旋转pan角度：

x = Math.cos(pan)\*x\_before - Math.sin(pan)\*z\_before;  
z = Math.sin(pan)\*x\_before + Math.cos(pan)\*z\_before;

[复制代码](javascript:void(0);)

围绕x轴旋转pitch角度：

y = Math.cos(pitch)\*y\_before - Math.sin(pitch)\*z\_before;  
z = Math.sin(pitch)\*y\_before + Math.cos(pitch)\*z\_before;

[复制代码](javascript:void(0);)

围绕z轴旋转tilt角度：

x = Math.cos(tilt)\*x\_before - Math.sin(tilt)\*y\_before;  
y = Math.sin(tilt)\*x\_before + Math.cos(tilt)\*y\_before;

[复制代码](javascript:void(0);)

那么基本的知识已经说完了。总结一下，当你在使用这些方程式操作摄像机全方位旋转的时候，只要取得相应的变量，然后替换在方程里就可以了。是不是看起来有 点难理解？不要担心，适应这些东西是需要花一点时间（特别是这些对你来说还是新课题的话），不过适应以后你应该就觉得很简单了。坦白的说，其实你并不需要 知道到底这些是怎样得来的，你只要知道如何使用它们，得到想要得结果就可以了（当然完全理解会对你以后的学习有一些帮助）。这些方程你可以写成一个函数， 然后命名它为“给我旋转”方程，当你需要摄像机旋转的时候，只要呼唤“给我旋转”就好了，至于“给我旋转”怎么做的工作，你就不需要担心了。

#### ****全方位旋转摄像机****

只说这些理论的东西，你肯定会觉得乏味，那么我举个例子来说明。下面这个程序演示了摄像机的全方位旋转，运行程序你会看到你置身在一个巨大的正方体 中，这个正方体是由很多我们的朋友小P组成的，不过这回用不同颜色的小P来代表不同的边。使用WS键控制纵向旋转，AD键控制横向旋转，QE控制倾斜角 度，鼠标点击屏幕禁止或者允许鼠标移动控制纵向和横向旋转。

**全方位旋转摄像机，使用WS纵向旋转，AD横向旋转，QE倾斜角度，鼠标点击禁止或者允许鼠标控制**

#### ****制作步骤：****

1. 首先定义几个常量，MAX\_OBJ是每条边上的物体数量，CUBE\_WIDTH是正方体的边长。

// constants  
var MAX\_OBJ = 6;  
var PI = 3.1415926535897932384626433832795;  
var CUBE\_WIDTH = 300;

[复制代码](javascript:void(0);)

2. 下面还是设置原点，场景，焦距等。

// same as usual  
var origin = new Object();  
origin.x = stage.stageWidth/2;  
origin.y = stage.stageHeight/2;  
origin.z = 0;  
  
var scene = new Sprite();  
scene.x = origin.x;  
scene.y = origin.y;  
this.addChild(scene);  
  
var focal\_length = 400;

[复制代码](javascript:void(0);)

3. 设置摄像机，这回的摄像机多了几个新的属性，pitching是纵向旋转角度，tilt是倾斜的角度。

var camera = new Object();  
camera.x = 0;  
camera.y = 0;  
camera.z = 0;  
camera.panning = -PI/8;                // init pan angle of our camera, pan left  
camera.pitching = -PI/8;               // pitch up  
camera.tilt = 0;                           // and no tilt

[复制代码](javascript:void(0);)

4. 设置一些全局变量，在处理键盘和鼠标事件时会用到。

// global booleans for our keyboard control  
var pan\_left;  
var pan\_right;  
var pitch\_up;  
var pitch\_down;  
var mouse\_ctl = true;

[复制代码](javascript:void(0);)

5. 下面布置一个正方体，你完全不必要明白我是怎么布置场景的，因为布置场景的方式并不唯一，所以如果你愿意的话，你可以自己动手布置3D场景，当然我也不介 意直接拷贝我的设置场景的代码去用。总之，这些代码就是初始化一些小P，然后把它们摆放在合适位置（围绕着摄像机）。

// ok, here you don't have to know how i acutally setup the cube  
// cause every body has a different way of doing that, if you really  
// interested in how i did it, then you may have a look  
// you can just copy my code and it will set it up for you  
var len = CUBE\_WIDTH/2;  
for (var seg = 0; seg < 3; seg++)  
{  
    var line\_h;  
    var line\_v;      
    var line\_z;  
      
    switch (seg)  
    {  
        case 0:  
            line\_h = true;  
            line\_v = false;  
            line\_z = false;  
            break;  
        case 1:  
            line\_h = false;  
            line\_v = true;  
            line\_z = false;  
            break;  
        case 2:  
            line\_h = false;  
            line\_v = false;  
            line\_z = true;  
            break;  
    }  
      
    for (var i = 0; i < MAX\_OBJ; i++)  
    {  
        if (line\_h || (i == 0 || i == MAX\_OBJ-1))  
        {  
            for (var j = 0; j < MAX\_OBJ; j++)  
            {  
                if (line\_v || (j == 0 || j == MAX\_OBJ-1))  
                {  
                    for (var k = 0; k < MAX\_OBJ; k++)  
                    {  
                        if (line\_z || (k == 0 || k == MAX\_OBJ-1))  
                        {  
                            var ball;  
                            if (line\_h)  
                            {  
                                ball = new SphereHorizontal();  
                                if ((i == 0 || i == MAX\_OBJ-1))  
                                {  
                                    ball = new SphereVertex();  
                                }  
                                else  
                                {  
                                    ball = new SphereHorizontal();  
                                }  
                            }  
                            if (line\_v)  
                            {  
                                ball = new SphereVertical();  
                                if ((j == 0 || j == MAX\_OBJ-1))  
                                {  
                                    continue;  
                                }  
                            }  
                            if (line\_z)  
                            {  
                                ball = new SphereStraight();  
                                if ((k == 0 || k == MAX\_OBJ-1))  
                                {  
                                    continue;  
                                }  
                            }  
                            ball.x\_3d = -len + (i)\*(CUBE\_WIDTH/MAX\_OBJ);  
                            ball.y\_3d = -len + (j)\*(CUBE\_WIDTH/MAX\_OBJ);  
                            ball.z\_3d = -len + (k)\*(CUBE\_WIDTH/MAX\_OBJ);  
                            scene.addChild(ball);  
                        }  
                    }  
                }  
            }  
        }  
    }  
}

[复制代码](javascript:void(0);)

6. 下面的函数就是刷新小P位置和大小的函数，这也是这篇文章主要讲述的内容，所以，请集中。OK，首先要得出小P（其中一个小P）到摄像机的x，y和z 距离。对于横向旋转角度panning，使用本文前面讲述的方程，把相应的x距离和z距离带入，然后得出新的x距离和z距离。使用相同的方法得出 纵向旋转角度pitching后的y和z距离，对摄像机倾斜角度tilt再次使用上述方程。这样就得到围绕三个轴旋转后新的x，y和z距离，继而便可以使 用老办法算出物体的缩放和移动。最后别忘记加一个z\_near变量，存储小P到摄像机的距离，以便于对所有小P进行z排序。

// update ball size and position  
// here is what we really care about, so concentrate  
function display(obj)  
{      
    var x\_distance = obj.x\_3d - camera.x;         // first we determine x distance ball to camera  
    var y\_distance = obj.y\_3d - camera.y;        // y  
    var z\_distance = obj.z\_3d - camera.z;        // z distance  
      
    var tempx, tempy, tempz;                        // some temporary variables  
      
    // two more trig you need to know about, suppose a is the previous angle  
    // cos(a+b) = cos(a)\*cos(b) - sin(a)\*sin(b)  
    // sin(a+b) = sin(a)\*cos(b) + cos(a)\*sin(b)  
    // thus we have the following  
    var angle = camera.panning;  
    tempx = Math.cos(angle)\*x\_distance - Math.sin(angle)\*z\_distance;  
    tempz = Math.sin(angle)\*x\_distance + Math.cos(angle)\*z\_distance;  
    x\_distance = tempx;  
    z\_distance = tempz;  
      
    angle = camera.pitching;                    // the same thing we have for pitch angle  
    tempy = Math.cos(angle)\*y\_distance - Math.sin(angle)\*z\_distance;  
    tempz = Math.sin(angle)\*y\_distance + Math.cos(angle)\*z\_distance;  
    y\_distance = tempy;  
    z\_distance = tempz;  
      
    angle = camera.tilt;                          // and tilt angle  
    tempx = Math.cos(angle)\*x\_distance - Math.sin(angle)\*y\_distance;  
    tempy = Math.sin(angle)\*x\_distance + Math.cos(angle)\*y\_distance;  
    x\_distance = tempx;  
    y\_distance = tempy;  
      
    if (z\_distance > 0)                                           // if the ball isin front of the camera  
    {  
        if (!obj.visible)                                  
            obj.visible = true;                                    // make the ball visible anyway  
              
        var scale = focal\_length/(focal\_length+z\_distance);   // cal the scale of the ball  
        obj.x = x\_distance\*scale;                             // calcualte the x position in a camera view   
        obj.y = y\_distance\*scale;                            // and y position  
        obj.scaleX = obj.scaleY = scale;                    // scale the ball to a proper state  
    }  
    else  
    {  
        obj.visible = false;  
    }  
      
    obj.z\_near = z\_distance;            // keep track of z distance to our camera  
}

[复制代码](javascript:void(0);)

7. 写一个循环函数，不停的调用第6步的函数刷新所有的小P。如果你一直有看文章的话那么这些对你来说应该不难。

// loop to update the screen  
function run(e:Event)  
{  
    for (var i = 0; i < scene.numChildren; i++)                  // update all the balls on the screen  
    {  
        display(scene.getChildAt(i));  
    }  
      
    swap\_depth(scene);  
}  
  
// bubble sort algo  
function swap\_depth(container:Sprite)  
{  
    for (var i = 0; i < container.numChildren - 1; i++)  
    {  
        for (var j = container.numChildren - 1; j > 0; j--)  
        {  
            if (Object(container.getChildAt(j-1)).z\_near < Object(container.getChildAt(j)).z\_near)  
            {  
                container.swapChildren(container.getChildAt(j-1), container.getChildAt(j));  
            }  
        }  
    }  
}

[复制代码](javascript:void(0);)

8. 最后是设置一些键盘和鼠标事件响应函数，完成我们的程序。

function key\_down(e:KeyboardEvent):void  
{  
    if (e.keyCode == 65)            // a  
        pan\_left = true;  
    if (e.keyCode == 68)            // d  
        pan\_right = true;  
    if (e.keyCode == 87)            // w  
        pitch\_up = true;  
    if (e.keyCode == 83)            // s  
        pitch\_down = true;  
}  
function key\_up(e:KeyboardEvent):void  
{  
      
    if (e.keyCode == 65)  
        pan\_left = false;  
    if (e.keyCode == 68)  
        pan\_right = false;  
    if (e.keyCode == 87)  
        pitch\_up = false;  
    if (e.keyCode == 83)  
        pitch\_down = false;  
}  
function key\_response(e:Event):void  
{  
    if (pan\_left)  
        camera.panning -= 0.01;  
    if (pan\_right)  
        camera.panning += 0.01;  
    if (pitch\_up)  
        camera.pitching -= 0.01;  
    if (pitch\_down)  
        camera.pitching += 0.01;  
          
    if (mouse\_ctl)                                // if allow mouse control pan and pitch  
    {  
        camera.panning += scene.mouseX/22000;  
        camera.pitching += scene.mouseY/22000;  
    }  
      
    // limit the pitch and tilt  
    if (camera.pitching < -1\*PI/3)  
        camera.pitching = -1\*PI/3;  
    if (camera.pitching > PI/3)  
        camera.pitching = PI/3;  
}  
function clicked(e:Event)                        // toggle mouse control  
{  
    mouse\_ctl = !mouse\_ctl;  
}  
  
// setup event listeners  
this.addEventListener(Event.ENTER\_FRAME, run);  
this.addEventListener(Event.ENTER\_FRAME, key\_response);  
stage.addEventListener(KeyboardEvent.KEY\_DOWN, key\_down);  
stage.addEventListener(KeyboardEvent.KEY\_UP, key\_up);  
stage.addEventListener(MouseEvent.CLICK, clicked);

[复制代码](javascript:void(0);)

总结一下，这篇文章中的三角函数部分可能有些抽象，因为是在3D空间中完成的，如果你把所有的例子首先映射到2D平面上会想对好理解一些。不过还是那句话，不要担心，你只要知道如何使用这些方程就可以了。

#### ****注意：物体自身围绕中心的3D旋转****

有一点不知道你有没有注意，那就是在本文的开头，我做了几个摄像机的旋转演示。在演示里，摄像机（物体）都是本身在旋转，而并不是我们的眼睛（摄像机）在 旋转。虽然到目前为止的文章里，还没有讨论到如何让物体自身旋转，不过很快就会看到一些例子。注：我做这些演示唯一想说明的就是三种旋转的机制，所以请不 要着急那些演示是怎么做出来的。

其实，关于Flash和3D空间的基本知识的介绍到这里我想应该结束了，从下篇文章开始就要关注3D物体，因此，如果对前面文章中的基础知识还是模模糊糊 的话，完全可以不必担心。不过我还是建议你自己多实验一些小例子，增加自己的空间感。相信你在不久的将来就可以开发自己的3D Engine了，加油，ALL THINGS ARE POSSIBLE！

http://www.cnblogs.com/yangzhou1030/archive/2008/11/06/1328146.html