# OpenGL绘制旋转三角形实例Study

世界坐标系：在OpenGL中，世界坐标系是以屏幕中心为原点(0, 0, 0)，且是始终不变的。你面对

           屏幕，你的右边是x正轴，上面是y正轴，屏幕指向你的为z正轴。长度单位这样来定：

           窗口范围按此单位恰好是(-1,-1)到(1,1)，即屏幕左下角坐标为（-1，-1），右上角

           坐标为（1,1）。

当前绘图坐标系：是绘制物体时的坐标系。程序刚初始化时，世界坐标系和当前绘图坐标系是重合

           的。当用glTranslatef()，glScalef(), glRotatef()等对当前绘图坐标系进行平移、

           伸缩、旋转变换之后，世界坐标系和当前绘图坐标系不再重合。注意，这里的平移旋

           转是将当前绘图坐标系看做一个整体在世界坐标系中进行旋转平移。然后，改变以

           后，再用glVertex3f()等绘图函数绘图时，都是在当前绘图坐标系进行绘图，所有的

           函数参数也都是相对当前绘图坐标系来讲的。

右手坐标系：让右手拇指指向x轴的正方向，食指指向y轴的正方向，如果中指能指向z轴的正方向，则称这个坐标系为右手直角坐标系（同理）

**OpenGL**中的**6**种坐标系

OpenGL中存在6种坐标系，

1. Object or model coordinates

2. World coordinates

3. Eye (or Camera) coordinates

4. Clip coordinates

5. Normalized device coordinates

6. Window (or screen) coordinates

从object coordainates到world coordinates再到camera coordinate的变换，在OPENGL中统一称为model-view转换，初始化的时候，object coordinates和world coordinates还有camera coordinates坐标重合在原点，变换矩阵都为Identity。model-view matix转换points,vectorsd到camera坐标系。

世界坐标是OpenGL中用来描述场景的坐标，Z+轴垂直屏幕向外，X+从左到右，Y+轴从下到上，是右手笛卡尔坐标系统。我们用这个坐标系来描述物体及光源的位置。

将物体放到场景中也就是将物体平移到特定位置、旋转一定角度，这些操作就是坐标变换。OpenGL中提供了glTranslate\*/glRotate\*/glScale\*三条坐标变换命令，利用OpenGL的矩阵运算命令，则可以实现任意复杂的坐标变换。

**非常重要：**OpenGL中有一个坐标变换矩阵栈(ModelView)，栈顶就是当前坐标变换矩阵，进入OpenGL管道的每个坐标(齐次坐标)都会先乘上这个矩阵，结果才是对应点在场景中的世界坐标。OpenGL中的坐标变换都是通过矩阵运算完成的，与图形学课本的描述完全一致。要注意的是变换中的矩阵乘法是左乘，而矩阵乘法与算术乘法不同，不符合交换律(万一不明白去看矩阵代数书好了)。

glTranslate\*(x,y,z)：平移，参数为各轴向的移动量。  
glRotate(d,x,y,z)：旋转，第一个参数为转动的度数，后三个参数表明是否绕该轴旋转。通常x,y,z中只有一个为1，其余为0，用连续几条旋转命令完成复杂旋转。由于矩阵运算的左乘特点，旋转命令的顺序与旋转动作的顺序正好相反。

物体坐标是以物体某一点为原点而建立的“世界坐标”，该坐标系仅对该物体适用，用来简化对物体各部分坐标的描述。物体放到场景中时，各部分经历的坐标变换相同，相对位置不变，所以可视为一个整体，与人类的思维习惯一致。

眼坐标是以视点为原点，以视线的方向为Z+轴正方向的坐标系中的方向。OpenGL管道会将世界坐标先变换到眼坐标，然后进行裁剪，只有在视线范围(视见体)之内的场景才会进入下一阶段的计算。

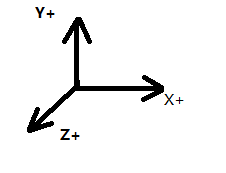
同样的，有投影变换矩阵栈(Projection)，栈顶矩阵就是当前投影变换矩阵，负责将场景各坐标变换到眼坐标，由所得到的结果是裁剪后的场景部分，称为裁剪坐标。前面提到过的视见体设定其实就是在建立该矩阵。

设备坐标：OpenGL 的重要功能之一就是将三维的世界坐标经过变换、投影等计算，最终算出它在显示设备上对应的位置，这个位置就称为设备坐标。在屏幕、打印机等设备上的坐标是二维坐标。值得一提的是，OpenGL可以只使用设备的一部分进行绘制，这个部分称为视区或视口(viewport)。投影得到的是视区内的坐标(投影坐标)，从投影坐标到设备坐标的计算过程就是设备变换了。

矩阵栈切换：glMatrixMode(GL\_MODELVIEWING或GL\_PROJECTION);本命令执行后参数所指矩阵栈就成为当前矩阵栈，以后的矩阵栈操纵命令将作用于它。

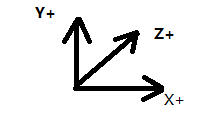
矩阵栈操纵命令：  
glPushMatrix(); 当前矩阵入栈，这时矩阵栈将栈顶值压入栈。  
glPopMatrix(); 栈顶出栈，通常与上一条命令配合使用。  
glLoadIdentity(); 将栈顶设为不变矩阵(就是对角线全为1其它为0的那个)。  
glMultMatrix(M);将栈顶T设为M·T。

1.OpenGL坐标系：



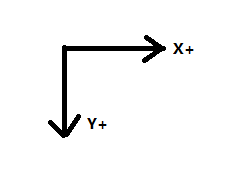
默认的OpenGL视点都是指向Z轴负方向。

2.相机局部坐标系：

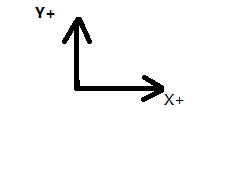


**因为常用的相机局部坐标系和OpenGL的坐标系是不同的，故用OpenGL显示的时候，要特别注意这点。**

3.常用的图片存储和读取坐标系，以及MFC窗口的坐标系：



4.CxImage图像库（第三方库）坐标系：



**void gluLookAt(GLdouble eyex,GLdouble eyey,GLdouble eyez,**

**GLdouble centerx,GLdouble centery,GLdouble centerz,**

**GLdouble upx,GLdouble upy,GLdouble upz);**

该[函数](http://baike.baidu.com/view/15061.htm" \t "_blank)定义一个视图[矩阵](http://baike.baidu.com/view/10337.htm)，并与当前矩阵相乘。

第一组eyex, eyey,eyez 相机在世界坐标的位置

第二组centerx,centery,centerz 相机镜头对准的物体在世界坐标的位置

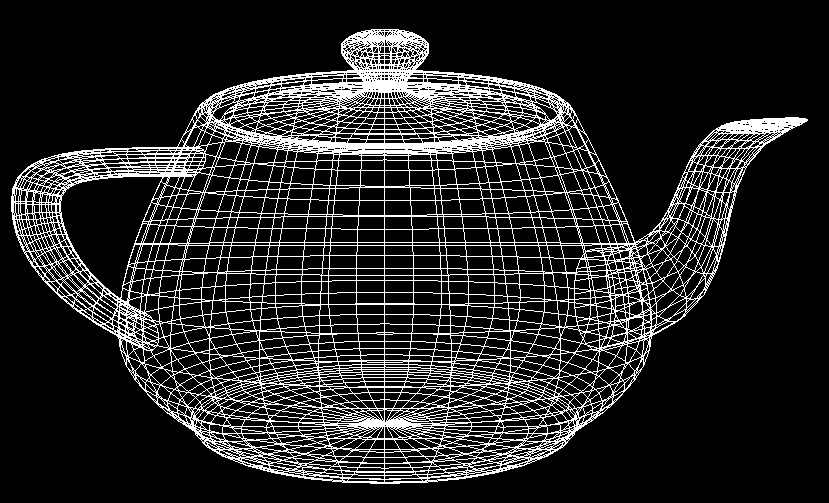
第三组upx,upy,upz 相机向上的方向在世界坐标中的方向

你把相机想象成为你自己的脑袋：

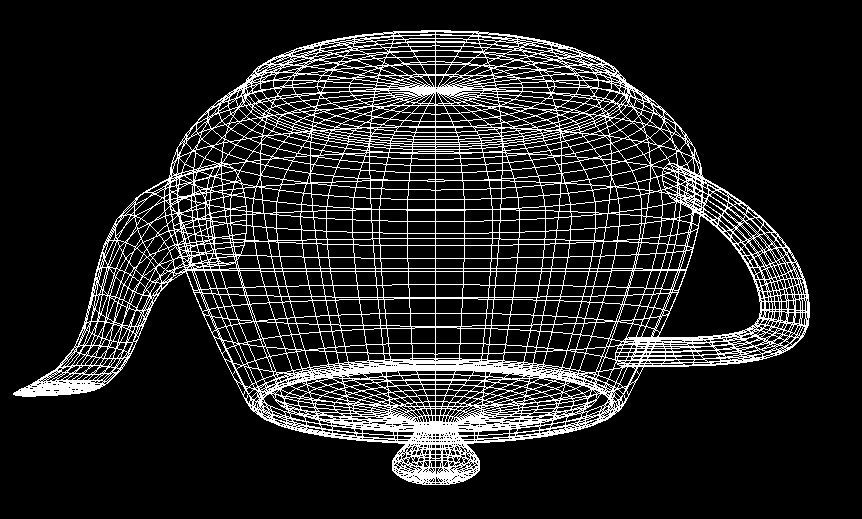
第一组数据就是脑袋的位置

第二组数据就是眼睛看的物体的位置

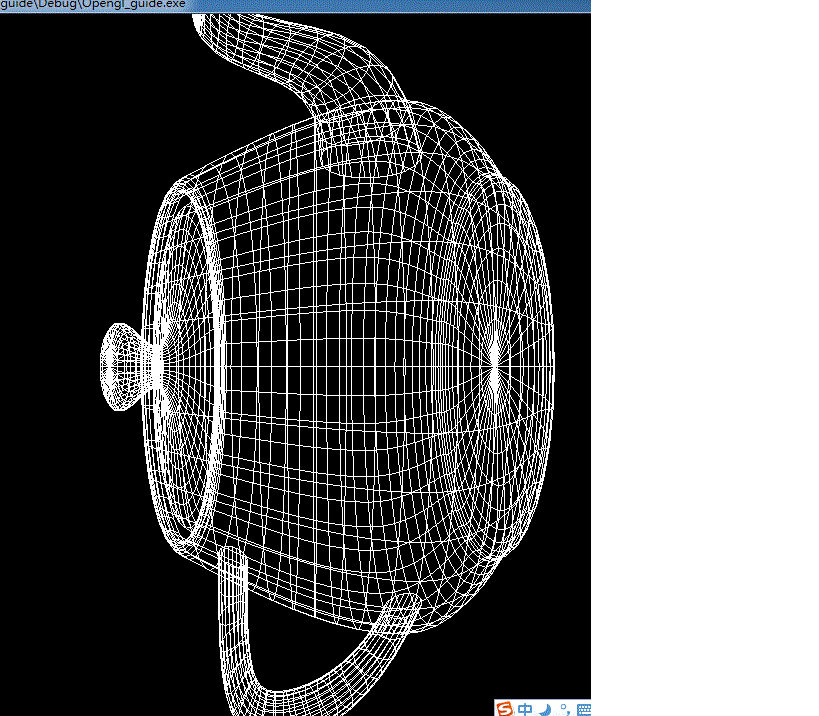
第三组就是头顶朝向的方向（因为你可以歪着头看同一个物体）。

一、上面的display()函数中：gluLookAt(0.0,0.0,5.0, 0.0,0.0,0.0, 0.0,1.0,0.0); 相当于我们的脑袋位置在(0.0,0.0,5.0)处，眼睛望向(0.0,0.0,0.0),即原点。后面的三个参数(0.0,1.0,0.0),y轴为1，其余为0，表示脑袋朝上，就是正常的情况。看到的情况如下图：

壶嘴在右，壶柄在坐，壶底在下，壶盖在上。

二、若将gluLookAt的后三个参数设置为（0.0,-1.0,0.0）,即y轴为-1,其余为0。这样表示脑袋向下，即人眼倒着看，看到的效果如下图：

三、再次修改gluLookAt的后三个参数为（1.0,0.0,0.0）;x轴为1，其余为0.即人的脑袋像右歪90度来看，即顺时针转90度（换个角度思考就是壶逆时针转90度），猜想看到的结果应该是壶嘴在上，壶盖在右，壶底在左，壶柄在下。如下图：



        如果并没有调用gluLookAt(),那么照相机就被设置为默认的位置和方向。**在默认情况下，照相机位于原点，指向z轴的负方向，朝上向量为(0,1,0)。**

       可以修改原来的代码。把视图变换函数gluLookAt()函数，改为模型变换函数glTranslatef(),并使用参数(0.0,0.0,-5.0)。这个函数的效果和使用gluLookAt()函数的效果是完全相同的，原因：gluLookAt()函数是通过移动照相机（使用试图变换）来观察这个立方体，而glTranslatef()函数是通过移动茶壶（使用模型变换）。另外注意：视图变换要在模型变换之前进行。

http://blog.csdn.net/wangqinghao/article/details/14002077