**第一章：静态绘制(二)**

**图元的位置变换**

这篇文章主要介绍图元位置变换方法。

[位置变换]:

再次提醒：默认分辨率为800X600，这里使用的图形单位以分辨率为准，而非OpenGl上下文式的坐标，这点请注意。

我们继续使用上节中的例子：

public static void main(String[] args) {

UESI UES = new R();

Obj rect = UES.creatObj(UESI.BGIndex);

rect.addGLRect("FFFFFF",0,0,300f,300f);

rect.addGLLine("00FF00",0,0,300f,300f);

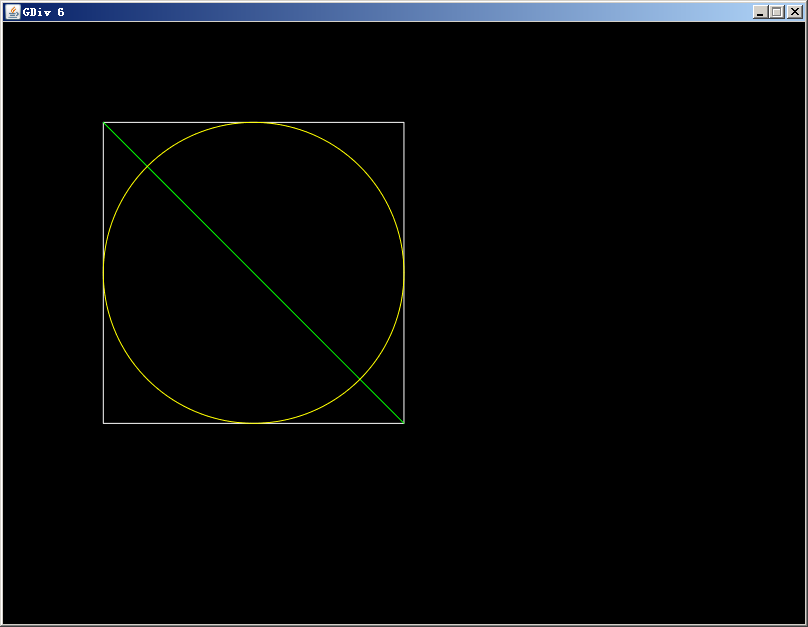
rect.addGLOval("FFFF00",0,0,300f,300f);

rect.show();

rect.setDx(100);

rect.setDy(100);

}



其实我们已经使用了位置变换方法中的setDx与setDy了，它们会将图元位置设置在制定的坐标位置，我们绘制图形时都是以原点为起点绘制的，之后使用set方法将其偏离原点100个像素，因为以原点绘制图形较为容易。

**抛弃坐标:** 但没人愿意校准坐标的，相信我；假如此时我们希望将这个图元放到屏幕中心，该如何呢？计算坐标么？我在很多游戏UI设计时花费了大量的时间校准坐标，这相当不值得，因此才有了校准API：

使用rect.setPosition(CANExPos.POS\_CENTER);方法替换掉两个set方法。

Obj rect = UES.creatObj(UESI.BGIndex);

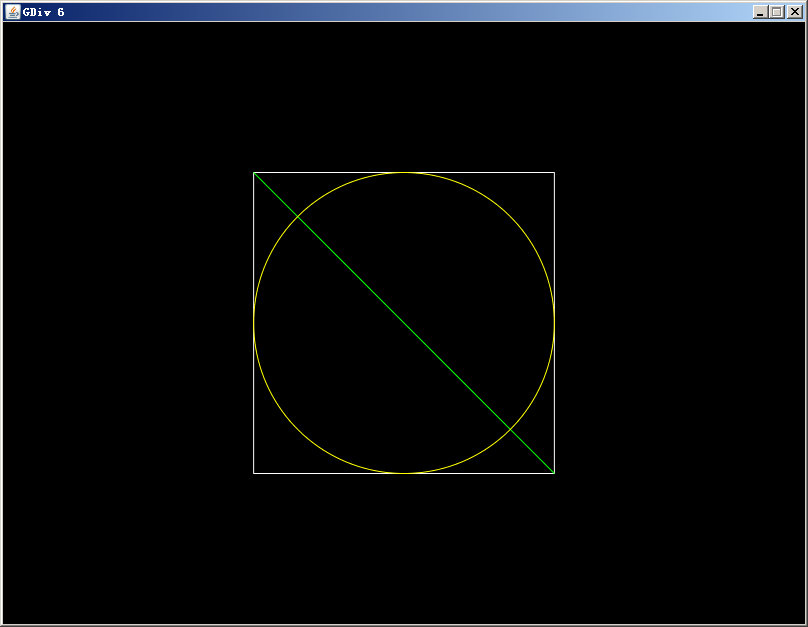
rect.addGLRect("FFFFFF",0,0,300f,300f);

rect.addGLLine("00FF00",0,0,300f,300f);

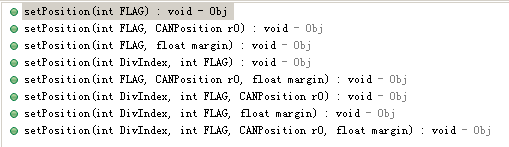
rect.addGLOval("FFFF00",0,0,300f,300f);

rect.show();

rect.setPosition(CANExPos.POS\_CENTER);



校准API详解：



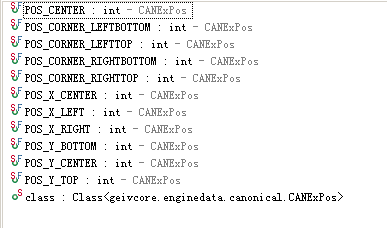
其中，FLAG参数是CANExPos或CANRefPos接口中的静态量。CAN开头的接口是图形标准，Obj已经将他们全部实现。

在上述setPosition方法中分为两类，一种是绝对定位类，它定位相对于屏幕的位置，Flag使用CANExPos中的值。

另一种是相对坐标类（其参数包含了rO），它用于相对其他图元进行定位。Flag使用CANRefPos中的值。

=====================================================

绝对定位FLAG：



POS\_CENTER:图元中心校准在屏幕中心，也就是上例所用。

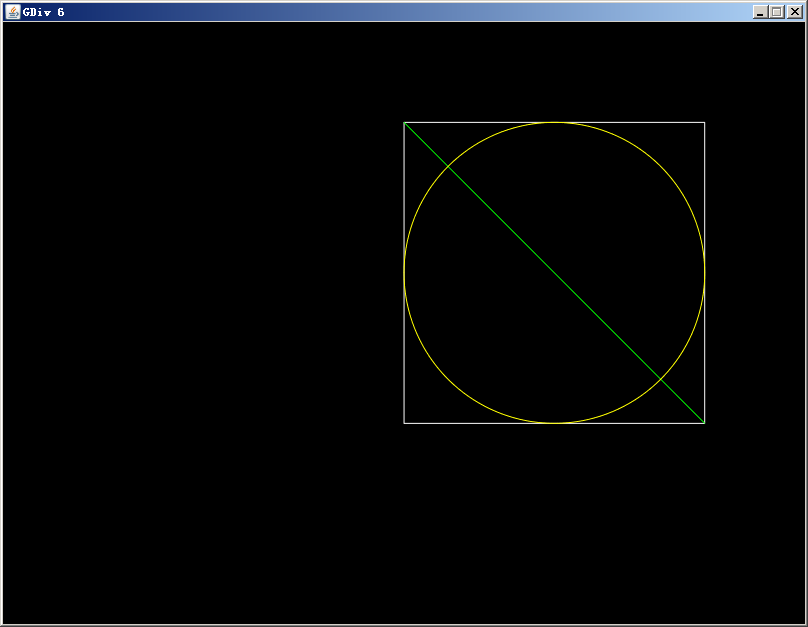
POS\_CORNER系列对应于屏幕四个角

POS\_X、Y系列对应于X、Y轴位置。

当然，我们可以使用包含margin参数的重载进行缩进，例如，我们希望把图形放置在屏幕右上角，并进行缩进100个单位：

只需要将：rect.setPosition(CANExPos.POS\_CENTER);

替换为：rect.setPosition(CANExPos.POS\_CORNER\_RIGHTTOP,100f);

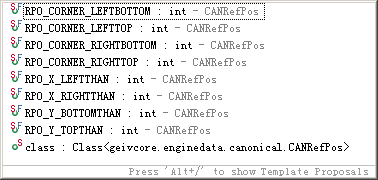
结果：

缩进指的是横纵向缩进，而非斜向；

setPosition除了提供margin重载外，还提供了图形索引重载，可以对图元中的指定图形进行缩进。

Margin参数是float型，如果大意写成了Int型，则会进行视图索引重载方法，而产生潜在的越界异常。

相对定位Flag



这些常量的名字比较容易理解：

CORNER系列为角相对位置，LEFTTHAN\RIGHTTHAN\BOTTOMTHAN\TOPTHAN分别对应在rO左、右、下、上的位置，当然你也可以使用margin进行相对缩进。

相对定位需要一个相对的标准（rO），它可以是一个图元Obj。

例子：

Obj rect = UES.creatObj(UESI.BGIndex);

rect.addGLRect("FFFFFF",0,0,300f,300f);

rect.addGLLine("00FF00",0,0,300f,300f);

rect.addGLOval("FFFF00",0,0,300f,300f);

rect.show();

rect.setPosition(CANExPos.POS\_CENTER);

//以上是中心位置的例子，在下面进行拓展。

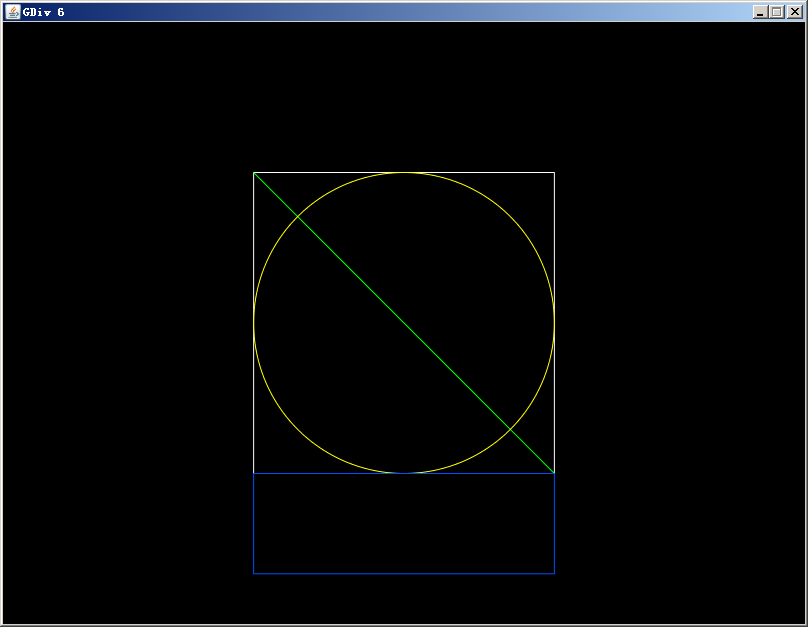
Obj rect2 = UES.creatObj(UESI.BGIndex);

rect2.addGLRect("004FFF",0,0,300f,100f);

rect2.setPosition(CANRefPos.RPO\_Y\_BOTTOMTHAN, rect);

rect2.show();

结果：



我们创建了300X100的蓝色矩形rect2，并将其放置在rect1的下方。

获得图元坐标：

使用setDx()\setDy()可以设置图元坐标。

同理，使用getDx()\getDy()可以获得图元当前坐标。

当然，它们包含有图元索引重载，你可以获得图元中某一图形的位置坐标。

**除了位置坐标**

一个图元图形如果包括了两个以上的点，例如矩形，除了标示位置的坐标DxDy外，还有其四个点坐标（注意位置坐标并不是第一个顶点的坐标，它们仅是在未进行旋转变换前恰巧重合着）；对于多点图形的顶点坐标，可使用getDxs\getDys方法获得

[例子]

在相对定位例子中末尾，加入如下语句：

for(int i = 0;i < 4;i++){

System.out.println(rect2.getDxs(i) + "," + rect2.getDys(i));

}

控制台打印结果：

250.0,450.0

550.0,450.0

550.0,550.0

250.0,550.0

同样，你可以使用setDxs\setDys方法设置这些坐标，它们同样具备图元索引重载。

[总结]

通过这一节内容，我们知道了改变图元位置的各种方法，包括使用setDx、setDy这样的直接设置法，也包括校准API中的绝对校准与相对校准。最后，我们介绍了获得图元坐标或图元内图形坐标的方法，以及除去位置坐标，图形顶点的坐标的设置及获得方法。