**第一章：静态绘制(一)**

**图元与基本图形绘制**

从这篇文章开始，进行引擎功能的详细讲解，第一章主要讲解静态绘制的基本API，这是制作动画的基础部分。

[图元][class: geivcore.enginedata.obj.Obj]]

我承认Obj这个名字不是很好，这是历史原因造成的。Obj是图元类，任何展示在屏幕上的图像都由Obj实现，任何对图像的变换操作由ObjAPI来完成，一个图元中容纳若干个图形类（可以是0个），在一个图元中图形被编号为由0开始的整数索引，但在实际操作中，使用索引的情况并不多见。

[引擎句柄][interface: geivcore.UESI]

Obj的创建由引擎句柄（UESI）完成，UESI是引擎的上层包装接口，它向上提供引擎的全部系统调用，在游戏系统开始时，最好将UESI作为构造器参数传入或干脆将UESI做静态化单例包装，以供上层框架随时调用。总之，UESI就是引擎的全部，这点需要注意。

[例子]画一个方形：

public static void main(String[] args) {

UESI UES = new R();//初始化引擎环境

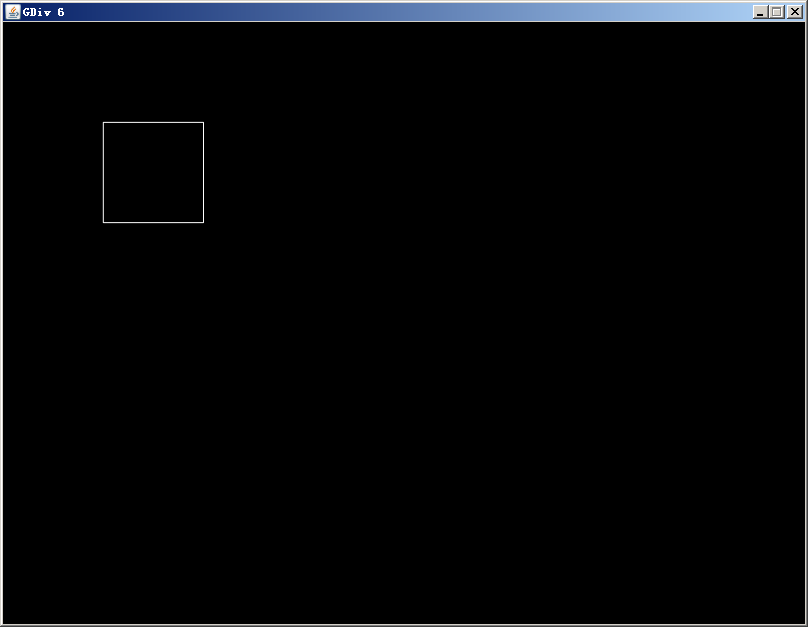
Obj rect = UES.creatObj(UESI.BGIndex);//创建一个图元，在背景层次。

rect.addGLRect("FFFFFF",0,0,100f,100f);//在图元上画一个正方形，位置（0,0），大小：100X100。颜色FFFFFF

rect.show();//显示

rect.setDx(100);//将图元x坐标移动到100位置

rect.setDy(100); //将图元y坐标移动到100位置

}

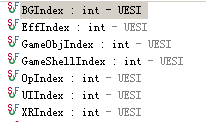
有必要进行详细的代码解释。

1）UESI UES = new R();//初始化引擎环境

UESI前者说过，是引擎包装接口，而R类是这个接口的一个具体实现，我们使用接口屏蔽R的具体实现细节。典型的游戏开发中只会使用到一个R，因此，只需要保留这个UESI引用即可，不要每次都实例化。

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

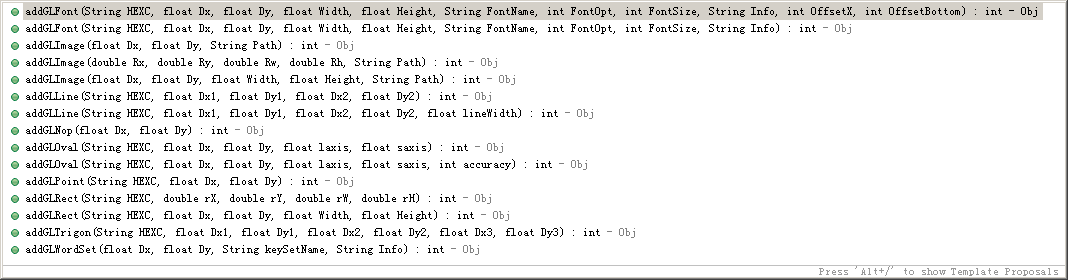
Obj为图元，图元的创建能且仅能使用UES的createObj方法，不要使用new Obj的形式。构造参数是这个图元所在的绘制层次，这个引擎由下至上保留了7个绘制层次，当多个图元出现在相同位置时，上层图元一定会覆盖下层图元，而同层图元覆盖规则依赖创建的顺序。



这里，BGIndex是最底层，一般放置游戏的背景元素。

3）rect.addGLRect("FFFFFF",0,0,100f,100f);

rect是一个图元引用，图元使用一系列addGL\*方法来创建各种图形，它们包括：



一共有十六中方法，当然包括了一些重载情况。

抛开重载不提，依次简单介绍一下：

addGLFont:向图元绘制一个静态字体，需要指定其颜色、位置、字体、字号、字形、内容、左下缩进。支持中文，当然也可以使用setFontString方法进行修改。（会重新生成文字纹理，所以不要频繁调用，字体调用会使用单独章节讲解）

addGLImage:创建一个矩形区域，填充指定的图像纹理，需要指定绘制位置和图像文件，图像文件可以是JPG\PNG\BMP等外部文件。在单独由IO读取文件流后，生成的纹理会保存在内存中，不会重复读取IO。

addGLLine：绘制一条线，需要给定颜色、两点坐标，可以指定绘制宽度。

addGLNop：什么也不绘制。Nop就是Nop。

addGLOval:绘制一个椭圆，指定位置与长短轴，由于是使用极坐标进行的多边形模拟，可以指定使用的定点数来提高图像质量。

addGLPoint:绘制顶点，指定位置与颜色。

addGLRect:绘制矩形，指定颜色与长宽。

addGLTrigon：绘制三角形，指定颜色与三组坐标。

addGLWorldSet:绘制字库，与静态字体不同，字库中每个字单独产生纹理，例如产生0~9 十个数字的纹理，就可以显示所有数字（会在字体章节详细讲解）对于易变数据的显示，推荐此种方式。

4）rect.show();

显示图元，你可以调用hide隐藏一个图元，或调用destory方法永久销毁。

5）setDx，Dy

设置图元的位置，整个图元会作为一个整体一起移动，（不只是位置，很多图元变换API会影响到图元中的全部图形，这点很有用），顺带一提，在不指定配置文件的情况下，默认长宽坐标为800X600，以左上为0坐标点，左到右为X方向，上到下为Y方向。

[另一个例子，多图形图元]

在上例中稍加改动：

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

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

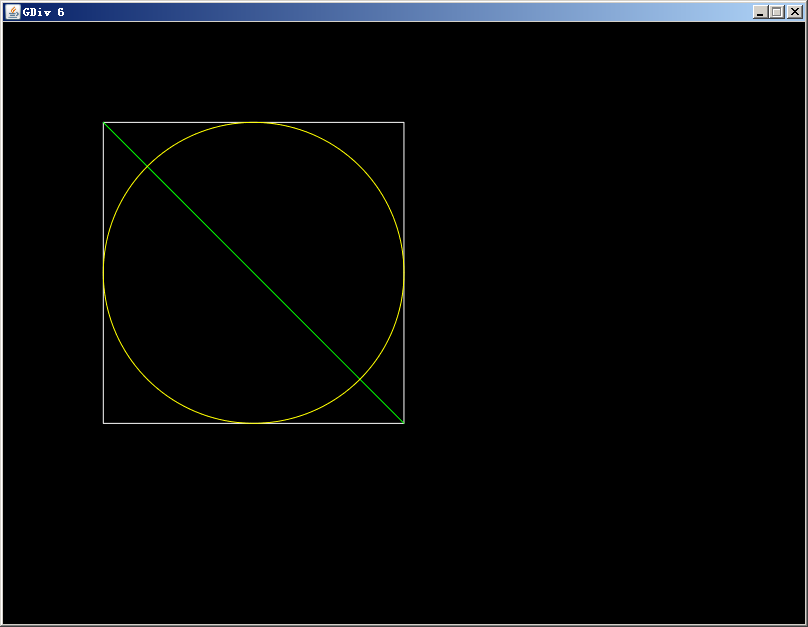
rect.addGLLine("00FF00",0,0,300f,300f);

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

rect.show();

rect.setDx(100);

rect.setDy(100);



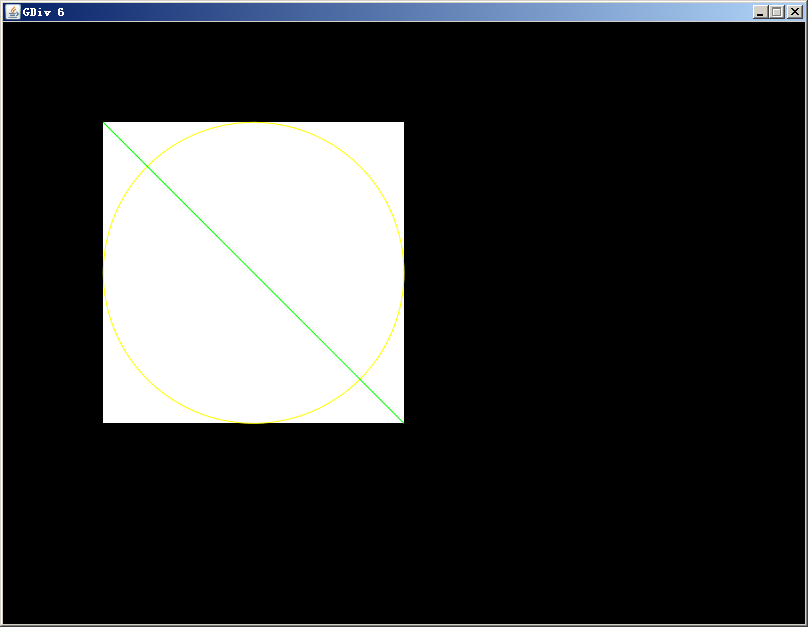
在rect图元中，创建了三个图形，分别为矩形、线、圆形。

它们分别被名为DivIndex的图元索引标记为0、1、2。

在进行图元变换过程中（包括位置、角度、通道），它们默认作为一个整体一同变换。当然，你也可以使用图元索引指定图元中要变换的某个图形。请看下例

[填充例子]

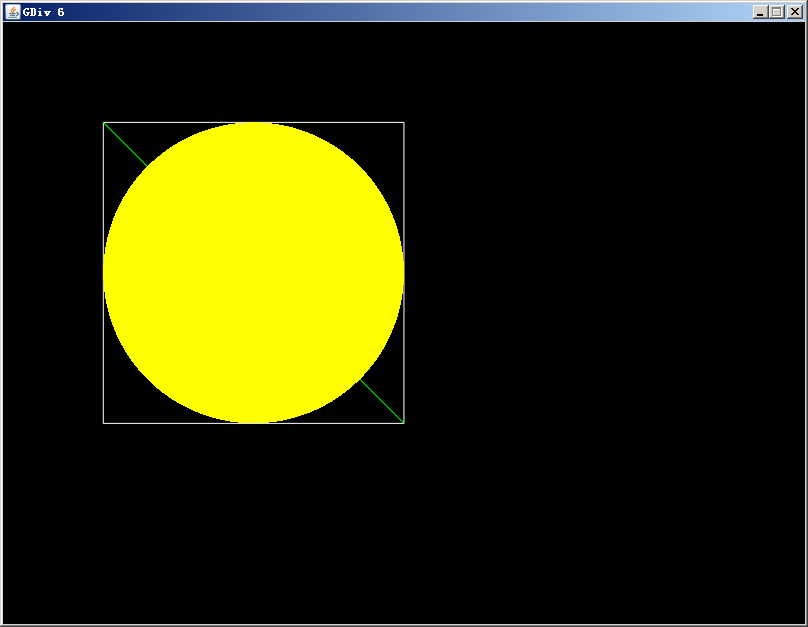
我们使用rect.setGLFill(true);方法将图元设置为填充状态。在show前加入这句后执行结果如下：



可以看到，作为底层的方形被填充了，但是圆形却没有填充。

此时若我想填充圆形而非方形该如何呢？

使用图元索引：rect.setGLFill(2,true);，其中2是圆形的图元索引，此时执行结果如下：



在图元类中，大部分set方法都由针对图元索引的重载，以提供灵活的操作方式。但图形过多时不推荐放到同一个图元中，图元单图形的方式是被推荐的。

[总结]

通过这一章，应该可以进行基本图形的绘制了。