分享到

Android平台下OpenGL初步

作者：hilarysky ，发布于2012-8-6

本文只关注于如何一步步实现在Android平台下运用OpenGl。

1、GLSurfaceView

GLSurfaceView是Android应用程序中实现OpenGl画图的重要组成部分。GLSurfaceView中封装了一个Surface。而android平台下关于图像的现实，差不多都是由Surface来实现的

2、Renderer

有了GLSurfaceView之后，就相当于我们有了画图的纸。现在我们所需要做的就是如何在这张纸上画图。所以我们需要一支笔。

Renderer是GLSurfaceView的内部静态接口，它就相当于在此GLSurfaceView上作画的笔。我们通过实现这个接口来“作画”。最后通过GLSurfaceView的setRenderer(GLSurfaceView.Renderer renderer)方法，就可以将纸笔关联起来了。

实现Renderer需要实现它的三个接口：onSurfaceCreated(GL10 gl, EGLConfig config)、onSurfaceChanged(GL10 gl, int width, int height)、onDrawFrame(GL10 gl)。下面就这三个接口的具体意义做个简单的介绍。

2.1、onSurfaceCreated

此方法看名字就知道它是在Surface创建的时候被调用的。因此我们可以在这个函数的实现中做一些初始化的工作。例如取出文房四宝、铺好画布、调好颜料之类的。它的函数原型如下：

```
public abstract void onSurfaceCreated (GL10 gl, EGLConfig config)
```

第二个参数在文档中没有关于它的任何public方法和域。因此我们可以不用管它。

第一个参数非常重要。如果说Renderer是画笔的话，那么这个gl参数，就可以说是我们的手了。如何操作这支画笔，都是它说了算！所以我们绝大部分时候都是通过这个叫做gl的手来指挥Renderer画图的。

2.2 onSurfaceChanged

当GLSurfaceView大小改变时，对应的Surface大小也会改变。值得注意的是，在Surface刚创建的时候，它的size其实是0，也就是说在画第一次图之前它也会被调用一次的。（而且对于很多时候，

Surface的大小是不会改变的，那么此函数就只在创建之初被调用一次而已）

原型如下：

```
public abstract void onSurfaceChanged (GL10 gl, int width, int height)
```

同样的，画图的手是必需的。

另外值得注意的是，它告诉了我们这张纸有多高多宽。这点很重要。因为在onSurfaceCreated的时候我们是不知道纸的宽高的，所以有一些和长宽相关的初始化工作还得在此函数中来做。

2.3 onDrawFrame

好了，我们的初始化工作做得差不多了，那么现在就是该真刀真枪画画的时候了！此函数就是真正给你画画用的。每调用一次就画一幅图。可能的疑问是这个函数什么时候会被调用呢？最开始的时候肯定是会被调用的。以后会有两种模式供你选择：

1.RENDERMODE_CONTINUOUSLY

相关文章

android人机界面指南
Android手机开发（一）
Android手机开发（二）
Android手机开发（三）
Android手机开发（四）
iPhone消息推送机制实现探讨
手机软件测试用例设计实践
手机客户端UI测试分析
手机软件自动化测试研究报告
更多...

相关培训课程

Android高级移动应用程序
Android应用开发
Android系统开发
手机软件测试
嵌入式软件测试
Android软、硬、云整合
更多课程...

成功案例

阿尔卡特 Linux内核驱动
艾默生 嵌入式软件架构设计
西门子 嵌入式架构设计
某国际通信公司 嵌入式需求
爱立信 嵌入式系统分析设计
丹佛斯 UML在嵌入式系统中
霍尼韦尔 嵌入式架构设计
更多...

2.RENDERMODE_WHEN_DIRTY

第一种模式（RENDERMODE_CONTINUOUSLY）：

连续不断的刷，画完一幅图马上又画下一幅。这种模式很明显是用来画动画的；

第二种模式（RENDERMODE_WHEN_DIRTY）：

只有在需要重画的时候才画下一幅。这种模式就比较节约CPU和GPU一些，适合用来画不经常需要刷新的情况。多说一句，系统如何知道需要重画了呢？当然是你要告诉它……

调用GLSurfaceView的requestRender ()方法，告诉它，你脏了。

这两种模式在什么地方设置呢？GLSurfaceView的setRenderMode(int renderMode)方法。 可以供你设置你需要的刷新模式。

还是来看看这个函数的原型吧： public abstract void onDrawFrame (GL10 gl) 很简单，只有手。

3、 Android下OpenGL绘图基本流程：

我们从画一个三角形开始说起：

3.1 MyRender

经过前面的介绍，我们应该知道现在需要做的事，就是写好Renderer的三个接口方法。

我们需要重新写一个类实现它，然后重写这三个方法。

class MyRender implements GLSurfaceView.Renderer

OK，笔已经拿好了。“铺好纸”是非常关键的一步。虽然我们说GLSurfaceView就是我们作图的纸，但是“铺”好这张纸，却也非常的重要。

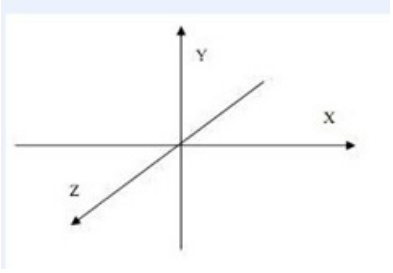
下面我们重点讲下，这纸该怎么铺。OpenGL这张纸可不是一般的纸啊。最起码，它是三维的。然而实际的显示屏幕却是一个平面。所以这纸还不好“铺”。

首先，不一定整张纸都用来作画吧，Surface不一定全部都用上（当然，一般情况下我们是全部用上的）。那么我们需要计划好，哪部分区域用来作画。

gl.glViewport(0, 0, width, height);

根据这里的参数width和height，我们可以知道这个宽高需要在onSurfaceChanged里得知。

然后，这一步很关键。如何在平面上画三维坐标的点或图形呢？OpenGL有一个坐标系，如下图：



我们需要将这个坐标系和我们的GLSurfaceView里的Surface做一个映射关系。

glMatrixMode(GL10.GL_PROJECTION);

gl.glLoadIdentity();

gl.glFrustumf(-400, 400, -240, 240, 0.3f, 100);

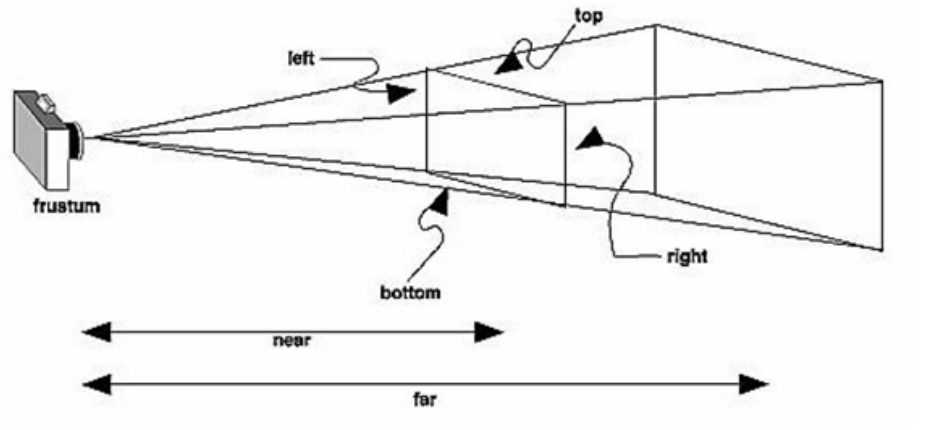
glMatrixMode(GL10.GL_PROJECTION); 是说我们现在改变的是坐标系与Surface的映射关系（投影矩阵）。

下一句 gl.glLoadIdentity(); 是将以前的改变都清掉（之前对投影矩阵的任何改变）。

glFrustumf (float left, float right, float bottom, float top, float zNear, float zFar) 这个函数非常Powerful。它实现了Surface和坐标系之间的映射关系。它是以透视投影的方式来进行映射的。

透视投影的意思见下图：

透视投影的想想见下图：



映射说明：

1、 前面一个矩形表示的是我们平面作图的范围。即Surface的作图范围。它有四条边，我们将它们暂时命名edgeLeft、edgeRight、edgeTop、edgeBottom。

2、 glFrustumf 参数中的left、right、bottom、top指的是作图范围的四条edge在OpenGL x = -400的位置。同理top、right、bottom的值表示的是edgeRight、edgeTop、edgeBottom这几条边在坐标系中的位置。

3、 上面第二点定出了作图范围的x和y的范围。那么对于3D的OpenGL这张纸来说，我们还需要定出z的范围。首先，要想象一下，相机或者眼睛在坐标系的哪个位置？

默认的眼睛的位置在OpenGL坐标的原点处(0,0,0)。视线方向是平行于Z轴往里看。

near表示的是眼睛到作图平面的距离(绝对值哦！)，far表示眼睛到最远可见处的平面范围。于是，默认情况下z的作图范围就是在-near到-far的位置。

4、 好了，我们已经确定好x、y、z方向的作图范围了。回过头来，就可以发现，这张“立体”的纸，是一个方椎体切去头部的平截头体。我们所画的物体坐标落在这个区域范围内的部分将可以被我们看到（即在屏幕里画出来）。OK，至此，我们把纸终于铺好了。

glMatrixMode函数的选项(参数)有后面三种：GL_PROJECTION，GL_MODELVIEW和GL_TEXTURE；

- GL_PROJECTION，是投影的意思，就是要对投影相关进行操作，也就是把物体投影到一个平面上，就像我们照相一样，把3维物体投到2维的平面上。这样，接下来的语句可以是跟透视相关的函数，比如glFrustum()或gluPerspective()；
- GL_MODELVIEW，是对模型视景的操作，接下来的语句描绘一个以模型为基础的适应，这样来设置参数，接下来用到的就是像gluLookAt()这样的函数；
- GL_TEXTURE，就是对纹理相关进行操作；

顺便说下，OpenGL里面的操作，很多是基于对矩阵的操作的，比如位移，旋转，缩放，所以， 这里其实说的规范一点就是glMatrixMode是用来指定哪一个矩阵是当前矩阵，而它的参数代表要操作的目标：

- GL_PROJECTION是对投影矩阵操作；
- GL_MODELVIEW是对模型视景矩阵操作；
- GL_TEXTURE是对纹理矩阵进行随后的操作；

3.2 画图之前先构图

Android 的 OpenGL 作图，不同于一般的作图，这点我们不得不感慨。它将数据和画完全分开来。例如，我们要画一个三角形。很显然，三角形有三个点。我们在画图之前首先要构图，比如每个点在哪个地方。我们将这些数据放在一个一个数组缓冲区中，放好这些数据之后，再统一一起画出来。

下面，主要讲下，如何将顶点数据和颜色数据放入符合 Android OpenGL 的数组缓冲区中。

首先我们要明白的是，OpenGL 是一个非常底层的画图接口，它所使用的缓冲区存储结构是和我们的 java 程序中不相同的。Java 是大端字节序(BigEdian)，而 OpenGL 所需要的数据是小端字节序(LittleEdian)。所以，我们在将 Java 的缓冲区转化为 OpenGL 可用的缓冲区时需要作一些工作。

byte 数据缓冲区

不管我们的数据是整型的还是浮点型的，为了完成 BigEdian 到 LittleEdian 的转换，我们都首先需要一个 ByteBuffer。我们通过它来得到一个可以改变字节序的缓冲区。

```
ByteBuffer mBuffer = ByteBuffer.allocateDirect(pointCount*dimension*4);

mBuffer.order(ByteOrder.nativeOrder());
```

注意，我们应该用“allocateDirect”来分配内存空间，因为这样才可以 order 排序。最后我们就可以通过：

```
resultBuffer = mBuffer.asFloatBuffer();

resultBuffer = mBuffer.asIntBuffer();
```

将字节缓冲区转化为整型或者浮点型缓冲区了。

3.3 终于画图了！

有了前面所有的准备工作之后，有一个好消息可以告诉你，我们终于可以画图了！而且，有了前面这么多工作，真正画图的工作其实比较简单。

3.3.1、清理好你的纸

前面我们说过了，在画图之前，一定要把纸给清理干净喽：

```
gl.glClear(GL10.GL_COLOR_BUFFER_BIT);
```

另外，之前我们在映射坐标系的时候，用了glMatrixMode(GL10.GL_PROJECTION);来指定改变的是投影矩阵。那么现在要画图了，所以我们需要指定改变的是“视图矩阵”：

```
gl.glMatrixMode(GL10.GL_MODELVIEW);

gl.glLoadIdentity();
```

3.3.2、启用数组

我们的前面说过，画图的数据都放在数组缓冲区里，最后再一起传过来作画。那么我们首先要告诉OpenGL，我们需要用到哪些数组。例如我们需要顶点数组和颜色数组：

```
gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);

gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
```

3.3.3、指定数组数据

我们前面已经构造好了我们的数据缓冲区，floatBuffer（或 IntBuffer）。现在我们只需要将这个数据缓冲区和对应的功能绑定起来就好了：

```
gl.glVertexPointer(3, GL10.GL_FLOAT, 0, VertexBuffer);

gl.glColorPointer(4, GL10.GL_FLOAT, 0, colorBuffer);
```

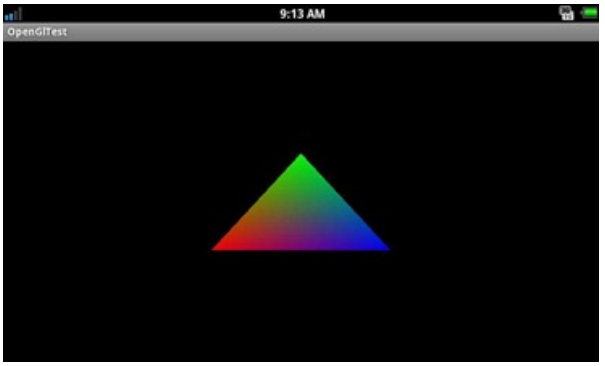
这两句话分别绑定了顶点数据数组和颜色数据数组。其中第一个参数表示的是每个点有几个坐标。例如顶点，有 x、y、z值，所以是 3；而颜色是 r、g、b、a 值，所以是 4。

3.3.4、画图！

```
gl.glDrawArrays(GL10.GL_TRIANGLES, 0, 3);
```

第一个参数指明了画图的类型——三角形（android 似乎只支持画三角形、点、线，不支持画多边形）。后面两个参数指明，从哪个顶点开始画，画多少个顶点。

OK！至此，我们的第一个三角形就画出来了，来看看效果吧。



相关文章

- 深度解析：清理烂代码
- 如何编写出拥抱变化的代码
- 重构-使代码更简洁优美
- 团队项目开发"编码规范"系列文章

相关文档

- 重构-改善既有代码的设计
- 软件重构v2
- 代码整洁之道
- 高质量编程规范

相关课程

- 基于HTML5客户端、Web端的应用开发
- HTML 5+CSS 开发
- 嵌入式C高质量编程
- C++高级编程

分享到

希望我们的资料可以帮助你学习，也欢迎投稿&提建议
给我
频道编辑：winner
邮　　件：winner@uml.net.cn