**毕业论文**

**Android平台基于openGL的图形渲染与实现**

**吉林大学**

**李志启**

**计算机科学与技术学院**

**摘要**

目录

1. 绪论

1.1 课题背景及研究的目的和意义

1.2 课题相关国内外研究状况

1.2.1 Android 的国内外研究状况

1.2.2 OpenGL 的国内外研究状况

1. Android 体系结构

2.1 android 系统架构简介

2.2 android 四大基本组件

2.3 avtivity 生命周期

2.4 activity 启动模式

2.5 android 消息分发机制简介

1. OpenGL 与 计算机图形原理

3.1 基本几何图形绘制

3.2 立方体的三维建模

3.3 单张3D 旋转效果

3.4 多张复合三维效果

3.5 本章小结

1. Android 之 OpenGL ES应用开发实践

4.1 coco2d-x 初体验

4.2 基于coco2d-x 编写游戏

4.3 android 平台 人体骨骼 3D 模型展示

1. OpenGL 建模测试

5.1 测试背景及意义

5.2 如何利用工具测试

5.3 如何编写用例测试程序

1. 结论

参考文献

致谢

1. 绪论
2. Android体系结构

2.1 android系统架构简介

Android是一种基于[Linux](http://baike.baidu.com/view/1634.htm" \t "http://baike.baidu.com/_blank)的自由及开放源代码的[操作系统](http://baike.baidu.com/view/880.htm" \t "http://baike.baidu.com/_blank)，主要使用于[移动设备](http://baike.baidu.com/view/8323830.htm" \t "http://baike.baidu.com/_blank)，如[智能手机](http://baike.baidu.com/view/535.htm" \t "http://baike.baidu.com/_blank)和[平板电脑](http://baike.baidu.com/view/74538.htm" \t "http://baike.baidu.com/_blank)，由[Google](http://baike.baidu.com/view/105.htm" \t "http://baike.baidu.com/_blank)公司和[开放手机联盟](http://baike.baidu.com/view/1245202.htm" \t "http://baike.baidu.com/_blank)领导及开发。

Android的系统采用分层结构，从高到低分为四层，依次为应用程序层，应用框架层，系统运行库层和linux 内核层。

2.2 Android四大基本组件

说到四大组件，对于android 开发者来说再熟悉不过了，分别是Activity ,Service,BroadcastReceiver和ContentProvider。

Activity 是一种展示型组件，用于界面的展示以及人机交互。是android 开发中最重要的组件。

Service 是一种计算型组件，用于在后台执行一系列的计算任务。其分为两种：一种是非绑定的service;另一种是跟activity 绑定的，其生命周期跟相应的activity关联起来。

BroadcastReceiver 是一种消息型组件，用于不同组件甚至不用应用间的消息传递，在应用开发中可以使用它来解耦程序。BroadcastReceiver没有生命周期这一概念。

ContentProvider 是一种数据共享型组件，用于组件或者应用间的数据共享，其主要跟本地数据库相关联，也没有生命周期这一概念。

2.3 Activity生命周期

Activity是android开发最重要的组件，充分理解activity的生命周期是每个android开发者必须掌握的基本技能，也为以后开发打下坚实的基础。

如图所示，在一般正常情况下，activity 会经历一下生命周期过程：

1. OnCreate: 在activity 创建时会调用该方法

我们需要在activity 创建时初始化很多对象。例如在后面章节提到的Glserfaceview ，渲染器等

1. OnRestart:表示activity 正在重启
2. OnStart:表示activity正在启动
3. onResume:表示activity已经可见，并显示在前台。注意其跟onStart的区别。onResume和onStart都示avtivity 可见，但onStart的时候表示activity 还在后台，onResume的时候avtivity 已经显示到前台了。
4. onPause:表示avtivity 正在停止
5. onStop: 表示avtivity 即将停止

在activity 停止时我需要停止与该avtivity相关的所有内容，否者可能会出现异常，导致程序崩溃。

1. onDestroy: 表示activity即将被销毁

在activity被销毁的时候，我们要主动回收所有相关的内存，特别是在openGL 编程中，会使用直接内存，不受java 内存的回收机制的处理。这部分内存若不能及时手动回收，会导致内存泄漏甚至内存溢出。

2.4 activity 启动模式

在android 应用程序中，Activity 可以简单理解为对应每个页面，每次在启动activity时，会创建activity的实例，然后放入任务栈中，当我们点击back 回退键时，任务栈就会出栈，直到为空。开发者会根据不同的业务需求制定不同的启动模式。Activity 共有四种启动模式（launchMode）。

1. Standard: 标准模式，每次启动activity 都会创建实例，放入当前任务栈中。谁启动了这个activity，那么activity 就会运行在启动它的activity所在的任务栈中。
2. SingleTop:栈顶复用模式，如果新启动的activity 在任务栈栈顶，就不会重新创建新的实例，而是直接复用栈顶的实例。
3. SingleTask: 栈内复用模式，如果新启动的实例在栈中存在，则复用这个实例，并弹出该实例上面的所有实例，让该实例置于栈顶。否则创建新的实例。
4. SingleInstance:单实例模式，当启动新activity 会创建一个新的任务在，然后把activity 实例押入栈中。以后就一直复用这个实例，该栈中也一直只保存一个实例，直到被销毁。

在本论文中，我们可以通过设置mainActivity的luanchMode 为singleTask ,这样避免在测试中反复启动activity而重复创建实例，造成内存空间浪费，影响测试结果。

2.5 Activity 消息分发机制简介

本文第四章会涉及android 人机交互，尽管android官方就有像onclick() 等基本交互接口框架，但是在基于openGL的开发中，这些基本框架是满足不了所有需求的。令人可喜的是，android系统是基于事件分发设计的，人机交互事件也是如此。这里介绍一下android消息事件分发机制，理解透彻后，在以后的开发中即使没有现成的API ,我们也能轻松地处理各种自定义的事件交互。

2.3.1 handler

Handler 是android 中的一种处理线程中消息循环的机制。主要涉及thread、MessageQueue、Looper 。

如图所示：

Thread

Looper

MessageQueue

Handler

其中MessageQueue是消息队列，Looper 是消息循环，handler 可以理解为消息处理器，他们都是构建在thread 上。我们用一张图来更清楚地理解他们之间的关系。

Handler.send(Message)

handler.dispatchMessage

thread

Looper

Message

messageQueue

我们把android 中消息队列看成传送带，把每个消息看成是传送带上传送的一个货物，把Looper看成是传送带电机，把线程看成是电源。Handler 作为消息的处理者，他通过send 方法向消息队列不断发送消息，而在传送带的另一端，handler 又通过dispatch 方法将消息从消息队列中不断的取出，分发给其他组件。这样整个消息的分发机制就这样运作了起来。

那么问题来了：为什么不直接将消息给需要他的组件，而不是搞个消息队列这么大费周章的处理消息？

原因是这样的：①. android的消息的发生很难由开发者控制，比如点击事件的消息，对于开发者来说根本不能预先知道用户做出来了怎样的点击事件；②. 还有如果消息数量很多，并且是高并发，这时系统就没法处理所有消息了；③. 假如我们想要的消息有延迟，比如网络请求，我们不可能一直等待消息的到来，这是就需要单独开一个子线程来处理，我们就可以先做其他事情了。

以上3种情况我们都可以使用android的这种消息分发机制解决问题。他具有良好的可控制性，效率也非常高效。相当于把消息都加入到一个缓存队列里面，按需慢慢来处理消息。

1. OpenGL ES for android

3.1 初识OpenGL ES

正如本文第二章所述，在android系统架构层次中的系统运行库层中，专门用于图像处理的是系统运行库是OpenGL ES。OpenGL（全写Open Graphics Library）是个定义了一个跨编程语言、[跨平台](http://baike.baidu.com/view/469855.htm" \t "http://baike.baidu.com/_blank)的[编程接口](http://baike.baidu.com/view/897136.htm" \t "http://baike.baidu.com/_blank)的规格，它用于三维图象（二维的亦可）。OpenGL是个专业的图形程序接口，是一个功能强大，调用方便的开源的底层图形库。而OpenGL ES (OpenGL for Embedded Systems) 是 [OpenGL](http://baike.baidu.com/view/9222.htm" \t "http://baike.baidu.com/_blank)三维图形 API 的子集，针对手机、PDA和游戏主机等[嵌入式设备](http://baike.baidu.com/view/2778983.htm" \t "http://baike.baidu.com/_blank)而设计。

OpenGL ES 有诸多版本。版本1.0提供了一个不够灵活的、固定功能的管道。版本2.0 则推出了可编程的管道，可一解决任何图形学问题。版本3.0 在2.0 的基础上添加了一些新的特性，但是还未被广泛使用。本文全篇均以OpenGL 2.0 为准。

3.2 顶点与着色器

3.2.1 点、直线、三角形

首先介绍一下OpenGL中的基本概念。在OpenGL中，只能绘制点、直线、三角形。

三角形是最基本的几个图形，它有如此稳定的结构，并且在生活中随处可见。点和直线可以用于某些效果，但只有三角形才能用来构建复杂的对象和纹理场景。

在OpenGL中，我们把单独的点放在一个组里构建出一个三角形，再告诉OpenGL 怎样连接这些点。我们想要构建的所有东西都要使用点、直线、三角形来定义。如果需要构建更复杂的对象，就需要有足够的点来拟合对象中的曲线。（在定义三角形的时候，约定以逆时针的顺序排列顶点）

3.2.2 着色器介绍

在把定义的对象画到屏幕上之前，需要找OpenGL的管道中传递，这里就需要使用着色器（shader），它会告示GPU 如何绘制数据。用两种shader:

1. 顶点着色器（vertex shader）生成每个顶点的最终位置
2. 片段着色器（fragment shader）为图像基本元素的每个片段生成最终的颜色，一个片段类似于计算机屏幕上的一个像素。

当最终的颜色生成后，OpenGL就会把他们写到帧缓冲区（frame buffer 内存的某部分）中，接着android就会把这个帧缓冲区显示在屏幕上。

基本流程如下：

读取顶点数据-->执行顶点着色器-->组装图元-->光栅化图元-->执行片段着色器-->写入帧缓冲区-->显示在屏幕上

3.2.2 编译着色器

1. 加载着色器
2. 创建着色器对象
3. 上传和编译着色器源代码
4. 验证编译状态并返回shader对象ID

3.2.3 链接进OpenGL 程序

1. OpenGL program 简介

这里指的是OpebGL 的一个组件，与一般的应用程序不同。一个OpenGL program就是把一个vertex shader 和一个fragment shader 链接在一起变成单个对象。他们两总是一起工作的。

1. 新建程序对象并附上着色器
2. 链接程序

3.3 在屏幕上绘制

1. 使用创建的OpenGL 程序

glUseProgram(program);

1. 获取着色器相关属性
2. 关联属性与顶点数据
3. 使能顶底数组
4. 绘制在屏幕上

3.4 OpenGL三维世界

3.4.1坐标变换

3.4.2 三维数组定义

3.4.3 透视投影

3.4.4 矩阵变换

3.5 触控交互

1. 对activity增加触控交互
2. 渲染器处理交互事件
3. 拖拽移动物体
4. OpenGL ES应用实战