# 贪吃蛇

## 游戏简介

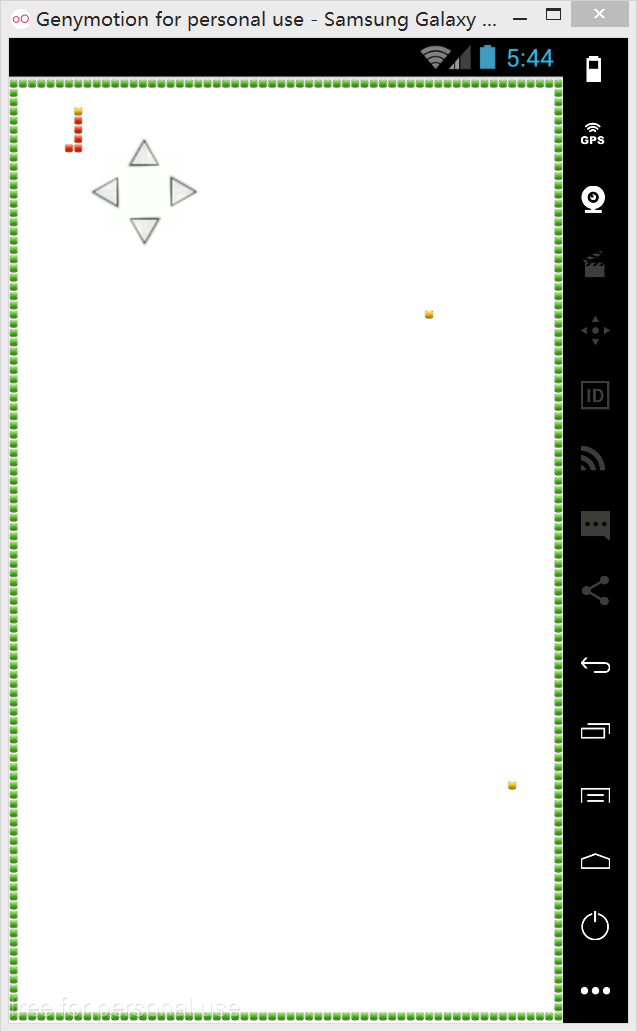
贪吃蛇是一款经典的手机游戏，规则也很简单：一条小蛇在屏幕中来回游走，吃随机出现的苹果，每吃一个蛇的身长就会增加。在游走过程中不能碰到周围的墙壁也不能碰到自己的身子，不然立刻游戏结束。

## 案例设计与实现

### 2.1 项目简介

整个Snake工程包括三个源文件，分别是：TileView.java、SnakeView.java、TileView.java。TileView是基于Android的View类实现的方块图类，用来支撑上层类的调用，绘制方块图的显示界面。 SnakeView继承了TileView，封装了数据的操作实现了游戏的功能。Snake是这个游戏的入口。Coordinate，RefreshHandler是2个辅助类，其中，Coordinate是一个包括两个参数，用于记录X轴、Y轴的简单类；RefreshHandler将RefreshHandler对象绑定某个线程并给它发送消息更新视图。游戏演示图如下：





### 2.2 需求分析

游戏实现的功能：

1.小蛇的移动、身长的增加

2.苹果的随机出现

3.碰到墙壁和自己立刻结束游戏

4.刷新

### 2.3 界面设计

2.3.1 Snake\_Layout

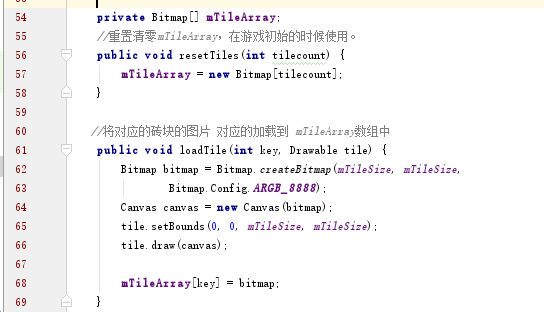
本游戏只有一个主界面：Snake\_Layout.xml:

布局采用FrameLayout（层布局）。接下来上面是一个SnakeView的视图，再上面是RelativeLayout（相对布局），里面放了开始按钮Button和说明文字TextView，最上面是一个绝对布局（AbsoluteLayout），该布局放置了四个ImageButton是上、下、左、右四个方向的按钮。其中com.xmobileappSnake.SnakeView是一个自定义的View。代码如下：

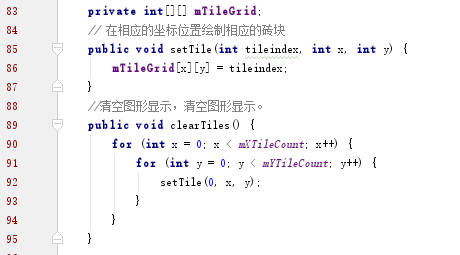
*<?***xml version="1.0" encoding="utf-8"***?>*<**FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 tools:context="com.zhbit.www.snake.Snake"**>  
 <**com.zhbit.www.snake.SnakeView  
 android:id="@+id/snake"  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"  
 tileSize="12"  
 android:background="@color/teneight"**/>  
 <**RelativeLayout  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"** >  
 <**TextView  
 android:id="@+id/text"  
 android:visibility="visible"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:gravity="center\_horizontal"  
 android:textColor="#FF0033"  
 android:textSize="24sp"**/>  
 <**Button android:id="@+id/play"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:layout\_marginTop="10px"  
 android:gravity="center\_horizontal"  
 android:text="点击这里开始"**/>  
 </**RelativeLayout**>  
 <**AbsoluteLayout  
 android:layout\_width="fill\_parent"  
 android:layout\_height="fill\_parent"**>  
 <**ImageButton  
 android:id="@+id/left"  
 style="?android:attr/buttonStyleSmall"  
 android:layout\_width="100px"  
 android:layout\_height="100px"  
 android:layout\_x="75px"  
 android:layout\_y="100px"  
 android:src="@drawable/left"** />  
 <**ImageButton  
 android:id="@+id/right"  
 style="?android:attr/buttonStyleSmall"  
 android:layout\_width="100px"  
 android:layout\_height="100px"  
 android:layout\_x="175px"  
 android:layout\_y="100px"  
 android:src="@drawable/right"** />  
 <**ImageButton  
 android:id="@+id/up"  
 style="?android:attr/buttonStyleSmall"  
 android:layout\_width="100px"  
 android:layout\_height="100px"  
 android:layout\_x="125px"  
 android:layout\_y="50px"  
 android:src="@drawable/up"** />  
 <**ImageButton  
 android:id="@+id/down"  
 style="?android:attr/buttonStyleSmall"  
 android:layout\_width="100px"  
 android:layout\_height="100px"  
 android:layout\_x="125px"  
 android:layout\_y="150px"  
 android:src="@drawable/down"** />  
 </**AbsoluteLayout**>  
</**FrameLayout**>

2.3.2 游戏画面绘制

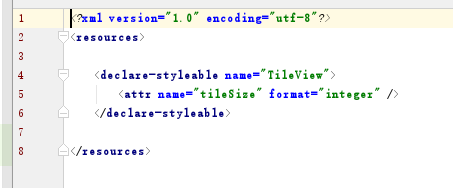
（1）Bitmap是Android中图像处理的最重要类之一。mTileArray存储着不同种类的bitmap图，通过resetTiles（重置清零，在游戏初始的时候使用），loadTile（加载相应的图片到mTileArray数组中），将游戏中的方块加载到这个数组。 其中在loadTile（）中做了一个 Drawable 到 bitmap 的转换，由于外部程序使用的时候是直接读取资源文件中的图片，  是drawable格式，而我们的数组是bitmap格式，方便最终的绘制，所以，需要进行一次到 bitmap的转换。

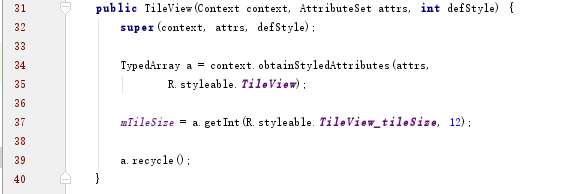


1. mTileGrid相当于一个二维数组的画布，通过setTile（）在相应的坐标位置绘制相应的砖块，而clearTiles（）函数是初始化mTileGrid，更新整个画布。

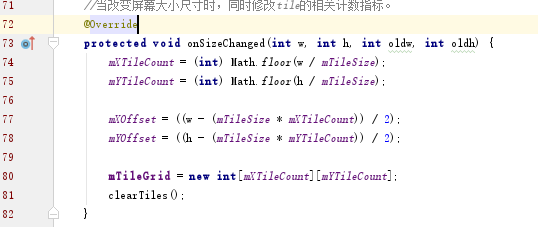


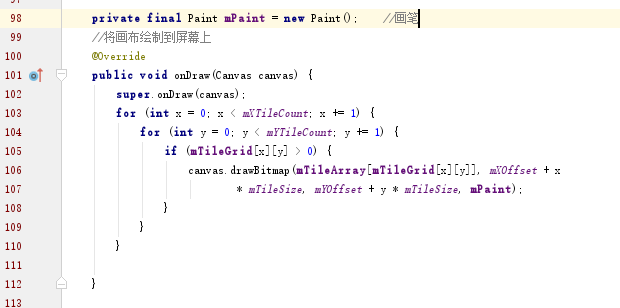
1. Public SnakeView(Context context , AttributeSet  attrs ,int defStyle) 中使用TypedArray，获取在attrs.xml中为TileView定义的新属性tileSize。





1. 当改变屏幕大小时，onSizeChanged（）方法会将相关计数指标同时改变。onDraw（）方法就是将这个画布绘制在手机上，其中Canvas就是绘制图形，需要画笔Paint实现。



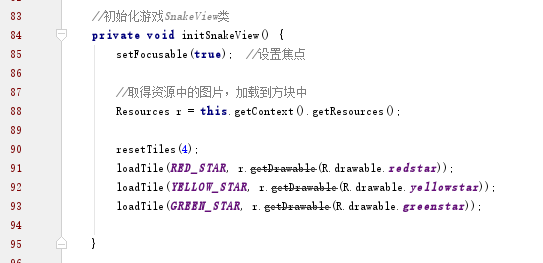


### 2.4功能实现

2.4.1 核心代码

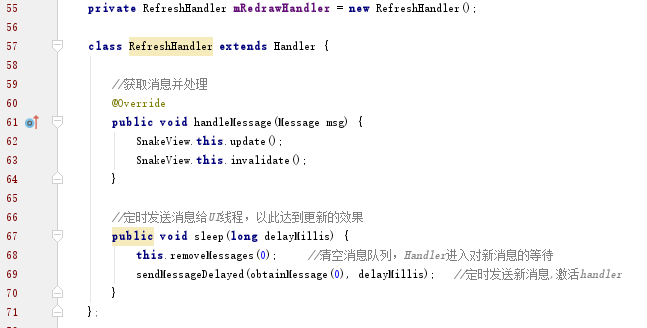
1. 初始化

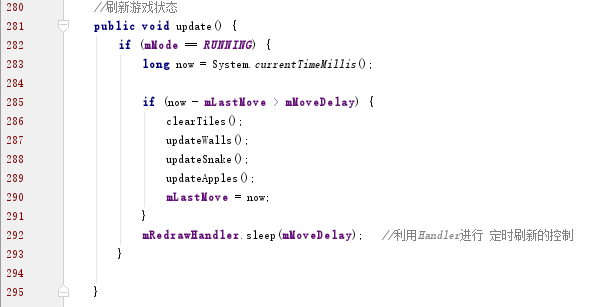
定义initSnakeView()初始化SnakeView类。由于存在文字和游戏之间的跳转，需要设置一个focus焦点来确定鼠标位置，还设置了mTileArray的长度，并且加载了三张图片。



2. 刷新

在Snake这个游戏中，辅助类RefreshHandler继承自Handler，用来把RefreshHandler与当前线程进行绑定，从而可以直接给线程发送消息并处理。Handle对消息的处理都是异步。RefreshHandler在Handler的基础上增加sleep（）接口，用来每隔一段时间后给当前线程发送一个消息。handleMessage（）方法在接受消息后，根据当前的游戏状态重绘界面。而update（）又与sleep（）相互调用构成一个循环，保证了游戏的及时刷新。





3. 保存数据

在setMode（）中共有RUNNING、PAUSE、READY、LOSE四种模式。其中在PAUSE时，我们可以保存暂停时的数据，以便再次玩时可以有上一次的记录。那么我们用Bundle方式保存数据，saveState（）保存数据，restoreState（）函数用来还原数据。

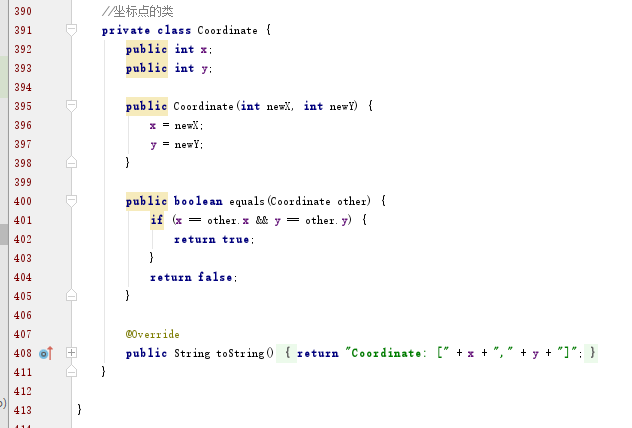


4. 点击事件按键的监听

如果按键的方向 跟蛇本身的运动方向完全相反，则无法执行。

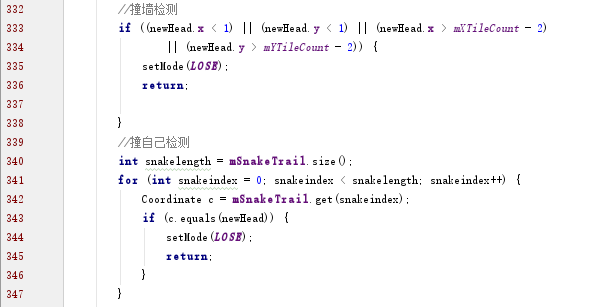


1. 坐标点的类



6. 游戏结束的检测

当碰到周围墙壁或蛇的头部碰到自己的身子则游戏结束，游戏模式setMode变为LOSE。



吃过果子的蛇身子会变长，用growSnake变量来记录，初始值设为FALSE。当蛇的头部碰到果子时，这个苹果会立即消失，同时用addRandomApple()方法再随机出现苹果，此时mMoveDelay打个九折，用来实现速度越来越快，growSnake的值同时变为TURE，蛇的身长加1。



## 3. 功能流程图

开始

初始化蛇和苹果

屏幕显示蛇和苹果

蛇开始移动，后坐标覆盖前坐标坐标

是否有按键

根据按键不同改变蛇的坐标

更改方向

是否越界或碰到蛇身

是否吃到苹果

蛇身加1

游戏结束

## 项目心得

（1）从资源中获取Bitmap

Resources res=getResources();

Bitmap bmp=BitmapFactory.decodeResource(res, R.drawable.pic);

（2）RefreshHandler是与当前线程进行绑定，然后就可以直接给线程发送消息并处理消息，而Handler对消息的处理是异步。RefreshHandler在Handler的基础上增加sleep（）借口，其主要作用就是每隔一段时间给当前线程发送一个消息，而hanleMessage（）则就是在接受消息后，让界面对当前的状态作出处理。

## github地址

Github地址:https://github.com/NN1007/Lay

## 参考文献

android Bundle的应用（用Bundle绑定数据,便于数据处理）:

<http://blog.sina.com.cn/s/blog_629b701e0100qr4l.html>

Bitmap用法总结：

<http://www.cnblogs.com/rongxh7/archive/2011/05/09/2041517.html>

RefreshHandler和Handler：

<http://qiangwei27.blog.163.com/blog/static/44899053201071871414275/>

代码的参考：

<http://blog.csdn.net/biaobiaoqi/article/details/6618313>

<http://blog.sina.com.cn/s/blog_6ca887bb0100xzo4.html>