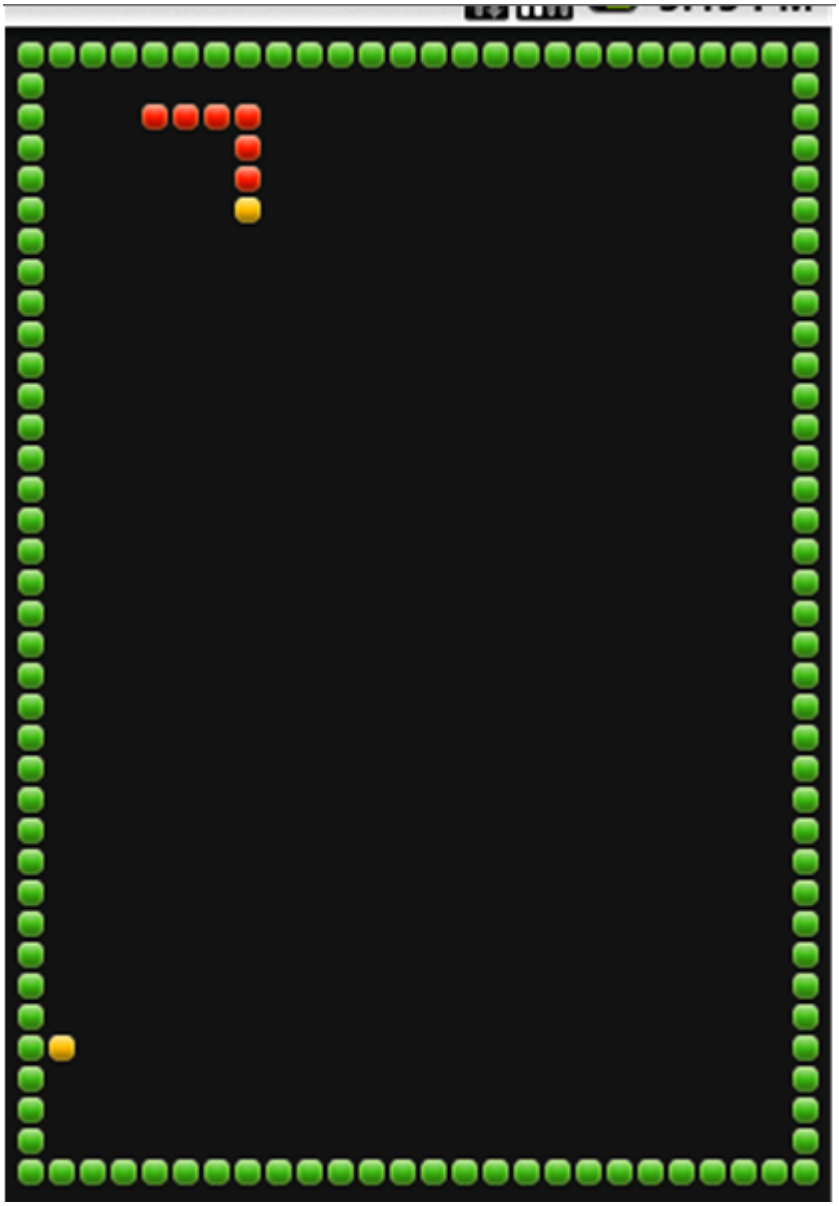
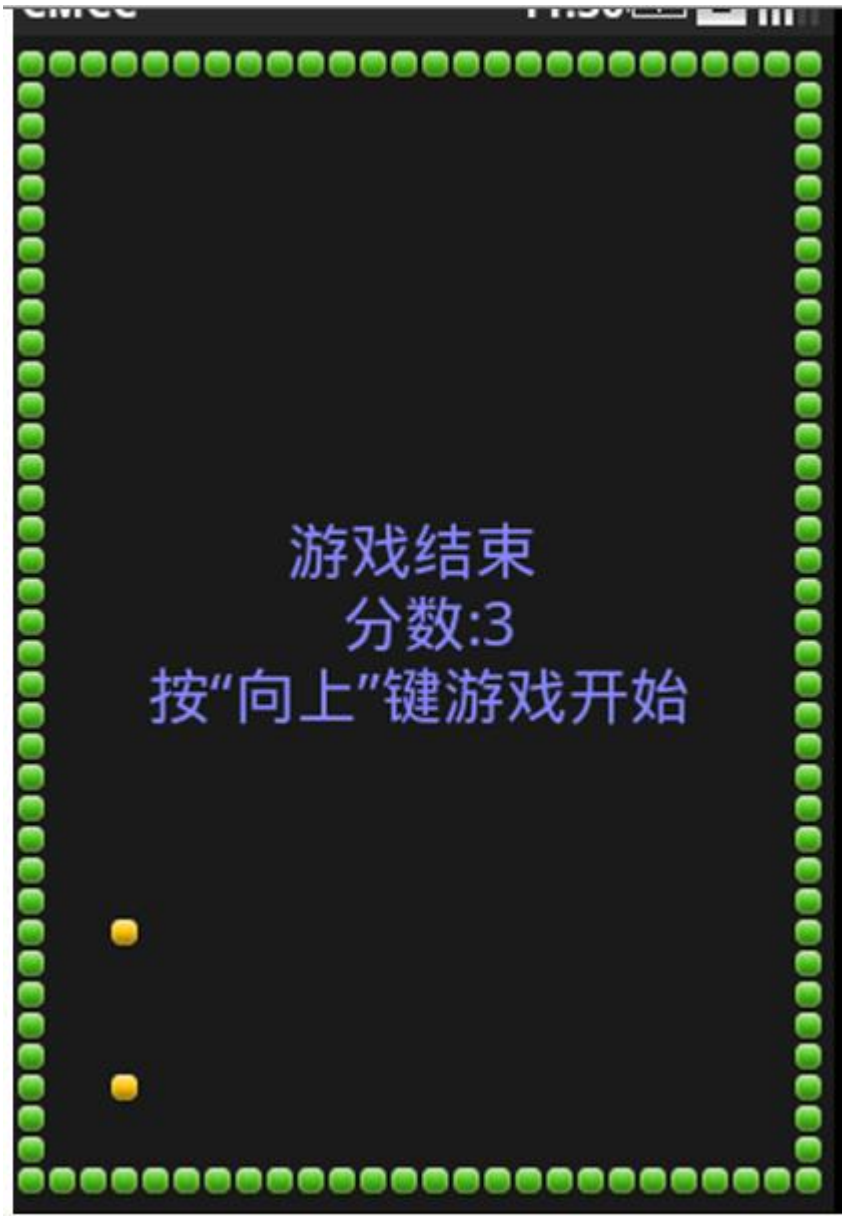
Android 游戏开发示例　贪吃蛇

效果图：



、



Snake 是游戏的实现类，通过控制小蛇在花园中游走寻找苹果，每吃掉一个苹果后，小蛇身体就会变的更长，如果小蛇撞上四周的墙或是碰到自己，游戏就会结束。 一般的游戏画面都是由地图， 精灵组成,Snake 结构也不外乎如此:
Snake ： Activity
SnakeView ： Game View，实现游戏的主体类
TileView ： Tile 类，可以实现小蛇的身体块，或者出现的苹果。

Snake Activity

在 res 布局资源创建 snake_layout ,当然，布局已经创建好了，大家打开 snake_layout.

```
<COM.GOOGLE.ANDROID.SNAKE.SNAKEVIEW
id="@+id/snake"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
tileSize="12"/>
```

这是自有的 View 类，cn.gswift.david.snake 不能写错，不然就找不到了。

TileView

TileView 使用了 Android 平台的显示基类 View，View 类是直接从 java.lang.Object 派生出来的.

TileView 类定义了 5 个 int 型全局的变量

mTileSize 小块的数量；

mXTileCount 小块 X 水平方向的数量;

mYTileCount 小块 Y 垂直方向的数量;

mXoffset 相对偏移位置；

mYOffset 相对偏移位置；

对于 View 的重写，我们需要通过 onSizeChanged 方法类实现。

```
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
mXTileCount = (int) Math.floor(w / mTileSize);
mYTileCount = (int) Math.floor(h / mTileSize);

mXOffset = ((w - (mTileSize * mXTileCount)) / 2);
mYOffset = ((h - (mTileSize * mYTileCount)) / 2);

mTileGrid = new int[mXTileCount][mYTileCount];
clearTiles();
}
```

在每次 View 移动之后，通过 clearTiles 方法清除显示，通过一个二维数组保存一个 gird 网格型的运动轨迹。

```
public void onDraw(Canvas canvas) {
super.onDraw(canvas);
for (int x = 0; x < mXTileCount; x += 1) {
```

```
for (int y = 0; y < mYTileCount; y += 1) {
if (mTileGrid[x][y] > 0) {
canvas.drawBitmap(mTileArray[mTileGrid[x][y]],
mXOffset + x * mTileSize,
mYOffset + y * mTileSize,
mPaint);
}
}
}
```

通过 onDraw 一次运动的轨迹画出来。这当中牵涉到一个 bitmap 的转化生成。

SnakeView

SnakeView 是游戏的主体，完成以下几个任务。

1. 随机产生小苹果，apples 这里是复数，当然是是大于 1 个苹果，所以代码中产生了两个苹果。
2. 游戏状态管理
3. 画蛇，view 的更新
4. 吃掉苹果后小蛇状态的变化
5. 画围墙

SnakeView 类首先定义好程序的当前状态:游戏准备，游戏中，游戏结束，并且设置静态变量。

SnakeView 对于游戏状态的控制，是通过 SaveInstanceState 实现了对对当前状态保存。在系统处理电话等之后，返回时，还可以继续进行游戏。

```
        if (savedInstanceState == null) {
// We were just launched -- set up a new game
mSnakeView.setMode(SnakeView.READY);
} else {
// We are being restored
Bundle map = savedInstanceState.getBundle(ICICLE_KEY);
if (map != null) {
mSnakeView.restoreState(map);
} else {
mSnakeView.setMode(SnakeView.PAUSE);
}
}
```

类中涉及到小蛇头部方向、苹果的颜色、游戏得分。

mSnakeTrail：坐标数组构成蛇的身体
mAppleList ：苹果的一个秘密地点
通过坐标的传递来实现的， 只要把头部的坐标点依次赋给下一个点， 后面的每一个点都走过了头部所走过的点，而蛇的头部就是负责去获取坐标，整个蛇的行走起来就很自然和连贯。
在蛇所经过的每一个坐标， 他们都要在苹果所在的（ArrayList mAppleList = new ArrayList()）坐标集里面集依次判断，若是坐标相同，那个这个苹果就被蛇吃了 。
RefreshHandler 类（嵌套在 SnakeView 类中），通过 Handler 获取消息，更新视图，
执行以下 2 行代码：

```
SnakeView.this.update();
SnakeView.this.invalidate();
```

```
public SnakeView(Context context, AttributeSet attrs) {
super(context, attrs);
initSnakeView();
}
```

主方法

要点提示:游戏中关键在于如何判断游戏的结束

1． 吃苹果

```
// Look for apples
int applecount = mAppleList.size();
for (int appleindex = 0; appleindex < applecount; appleindex++) {
Coordinate c = mAppleList.get(appleindex);
if (c.equals(newHead)) {
mAppleList.remove(c);
addRandomApple();

mScore++;
mMoveDelay *= 0.9;

growSnake = true;
}
}
```

2． 碰到了自己

```
// Look for collisions with itself
```

```
int snakelength = mSnakeTrail.size();
for (int snakeindex = 0; snakeindex < snakelength; snakeindex++) {
Coordinate c = mSnakeTrail.get(snakeindex);
if (c.equals(newHead)) {
setMode(LOSE);
return;
}
}
```

3．碰到墙

```
// Collision detection
// For now we have a 1-square wall around the entire arena
if ((newHead.x < 1) || (newHead.y < 1) || (newHead.x > mXTileCount - 2)
|| (newHead.y > mYTileCount - 2)) {
setMode(LOSE);
return;
}
```

按键事件处理

```
public boolean onKeyDown(int keyCode, KeyEvent event)
// TODO Auto-generated method stub
if (keyCode == KeyEvent.KEYCODE_DPAD_UP) {
Log.i(TAG, "KEYCODE_DPAD_UP");
}
return super.onKeyDown(keyCode, event);
}
```

小方块的运动

实现小方块动起来的秘密在于 view 的 public void invalidate ()
大家可以参看 SDK 文档中关于 View 中 Drawing 中的一小段话
To force a view to draw, call invalidate().//为了让 view 重画，可以调用 invalidate 函数

方法：

1. 在 DrawView 类中添加两个成员：

```
private int x,y;
```

同时实现 get,set 方法，

2. 在构造函数中添加他们的初始值，

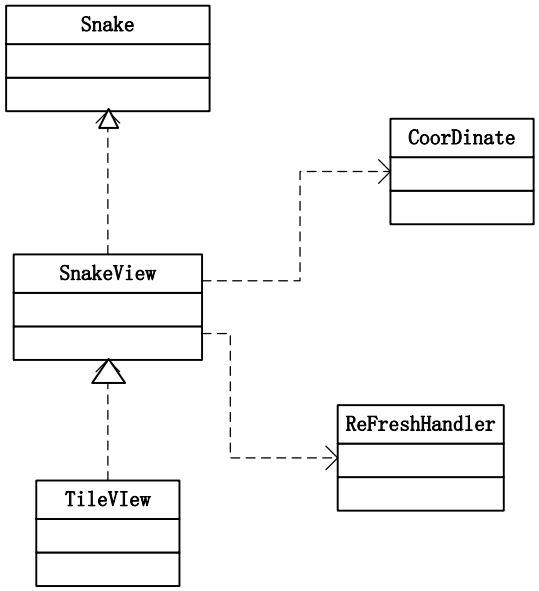
3. 修改 onDraw

```
@Override
protected void onDraw(Canvas canvas) {
super.onDraw(canvas);
Log.i(TAG, "onDraw 1");
canvas.drawBitmap(mTileArray, x, y, pa);
}
```

4. 修改 onKeyDown 函数

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
// TODO Auto-generated method stub
if (keyCode == KeyEvent.KEYCODE_DPAD_UP) {
Log.i(TAG, "KEYCODE_DPAD_UP");
dv.setX(dv.getX()+10);
dv.invalidate();
}
return super.onKeyDown(keyCode, event);
}
```

snakeUML 结构图



ReFreshHandler 类和 CoorDinate 类嵌套在 SnakeView 类里。