Cocos2d-android 学习笔记

# 前言

最近在做公司项目中要做动画特效，但是在复杂的动画导致开发周期长，性能差。

之前也考虑过引擎，但是看了很多引擎都感觉不好用。

忽然想试一下，之前看过的Mars的这个教程。想试试看。

另外在自己写动画的时候忽然想到cocos2d的这种类的结构。即使这个不能用于项目中，也可以学习一下代码。

可能存在的问题：

GLSurfaceView是否可以绘制在View上。

---参考自己github上之前对cocos2d-x的研究，我记得是解决了这个问题了。

就因为没有笔记所以需要去翻代码了。还是写笔记好啊，方便以后查阅。

# 二、Mars视频重新学习

## 1、创世纪

Github下载cocos2d，导入到eclipse，运行例子的过程就不写了。

创建测试项目。导入库到libs下。

|  |
| --- |
|  |

导演类的属性：

|  |
| --- |
|  |

**实验：**

由于这次学习是为了在android的项目中使用cocos2d，比如房间中动画的一部分动画用cocos2d来做，这样的话和游戏不同的是游戏是作为程序所有内容的，也是全屏的，而项目中一部分是要绘制在项目中一部分的。

由于Cocos2d是绘制在CCGLSurfaceView上的。

如果我们想要实现一部分，可以将这个CCGLSurfaceView作为布局中的一部分，或者添加到FrameLayout作为视图的一个图层来显示。

在这里添加jufan的子包。实验将聚范礼物动画用cocos2d来做

经过测试是可以的。具体看代码。

需要注意的是，需要是设置屏幕的渲染的宽高。

**director.setScreenSize()**

**这里进行了0.9倍缩放，如果设成全高会看不到，具体原因还不知道**

|  |
| --- |
| director.setScreenSize(WindowsDPUtils.*getWindowsWidthPX*(**this**) \* 0.9f,WindowsDPUtils.*getWindowsHightPX*(**this**) \* 0.9f); |

下面要做就是将CCGLSurfaceView背景透明就可以了，为了通用性切不改源代码。可以继承CCGLSurfaceView一个类。

将PixCCGLSurfaceView设置成透明的

为了让背景可以透明，需要将cocos2d的源码作为库的形式引入，因为需要改里面的源代码。

改CCDirector.java

|  |
| --- |
|  |

SurfaceView中的代码：

|  |
| --- |
| **public** **class** PixCCGLSurfaceView **extends** CCGLSurfaceView {  **public** PixCCGLSurfaceView(Context context) {  **super**(context);  // //背景透明  setZOrderOnTop(**true**);  getHolder().setFormat(PixelFormat.*TRANSLUCENT*);  setEGLConfigChooser(8, 8, 8, 8, 16, 0);  }  } |

将CCDirector.java中设置透明度的地方，封装成方法

|  |
| --- |
| **public** **float** bgAlpha = 1.0f;  /\*\*  \* 设置背景的透明度  \* **@param** alpha  \*/  **public** **void** setBGAlpha(**float** alpha) {  bgAlpha = alpha;  } |

更改的地方

|  |
| --- |
| // set other opengl default values  gl.glClearColor(0.0f, 0.0f, 0.0f,bgAlpha); |

调用的地方

|  |
| --- |
| //得到导演对象  CCDirector director = CCDirector.*sharedDirector*();  director.setBGAlpha(0.0f); |

## 2、精灵初体验

|  |
| --- |
|  |

简单的精灵与动作对象

|  |
| --- |
| //初始化精灵对象  mPlayerSprite = CCSprite.*sprite*("player.png");  //设置精灵对象的位置  // mPlayerSprite.setPosition(100, 100);  //通常用于标识坐标，或者向量  CGPoint point = CGPoint.*ccp*(100f,100f);  mPlayerSprite.setPosition(point);  //将精灵对象添加到布景层中  addChild(mPlayerSprite);  CGPoint targetPoint = CGPoint.*ccp*(400, 100);  //生成一个跳跃动作对象  CCJumpTo jumpTo = CCJumpTo.*action*(3, targetPoint, 200, 3);  //使用精灵对象执行该动作  mPlayerSprite.runAction(jumpTo); |

## 3、动作类

|  |
| --- |
|  |

动作类的方法：

|  |
| --- |
|  |

基础的瞬时动作：

|  |
| --- |
|  |

|  |
| --- |
| //第二次测试=================================================  /\*  //初始化精灵对象  mPlayerSprite = CCSprite.sprite("player.png");  CGPoint playerPos = CGPoint.ccp(100, 100);  mPlayerSprite.setPosition(playerPos);  addChild(mPlayerSprite);  //1、生成动作对象  CCFlipX flipX = CCFlipX.action(true);  //2、执行动作对象  mPlayerSprite.runAction(flipX);    //隐藏测试  CCHide hide = CCHide.action();  //执行动作  mPlayerSprite.runAction(hide);  \*/    //另外还有CCFlipY和CCShow类似 |

延时动画：

|  |
| --- |
|  |

|  |
| --- |
| //第三次测试==================================================  /\*  mPlayerSprite = CCSprite.sprite("player.png");  this.addChild(mPlayerSprite);  mPlayerSprite.setPosition(100, 100);  CGPoint movePoint = CGPoint.ccp(400,400);  // //移动动作  // CCMoveTo moveTo = CCMoveTo.action(2, movePoint);  // //执行动作  // mPlayerSprite.runAction(moveTo);    //旋转动画,角度制  CCRotateTo rotateTo = CCRotateTo.action(3,180);  //执行动作  mPlayerSprite.runAction(rotateTo);  \*/  //另外还有 CCScaleTo和CCBlink类似 |

## 4、向量

带方向的量

CGPoint向量，CGPoint坐标相对原点方向的量

|  |
| --- |
|  |

CCMoveBy与CCJumpBy用法，参考代码

组合动画：

|  |
| --- |
|  |

|  |
| --- |
| mPlayerSprite = CCSprite.sprite("player.png");  // CGPoint initPoint = CGPoint.ccp(100, 100);  // this.addChild(mPlayerSprite);  // mPlayerSprite.setPosition(initPoint);  //  // CGPoint targetPoint = CGPoint.ccp(300, 300);  // CCMoveTo moveTo = CCMoveTo.action(2, targetPoint);  //  // CCRotateTo rotateTo = CCRotateTo.action(2, 180);  //  // CCScaleTo scaleTo = CCScaleTo.action(2, 2);  //顺序执行  // CCSequence sequence = CCSequence.actions(moveTo, rotateTo,scaleTo);  //同步执行  //CCSpawn spawn = CCSpawn.actions(moveTo, rotateTo,scaleTo);    //回调函数  // CCCallFuncN func = CCCallFuncN.action(this, "onActionFinished");  //  // CCSequence seq = CCSequence.actions(moveTo, func);    //跟随动画 CCFollow    // mPlayerSprite.runAction(seq); |

其他一些动画：

|  |
| --- |
| // //淡入淡出动画  mPlayerSprite = CCSprite.*sprite*("player.png");  CGPoint initPoint = CGPoint.*ccp*(100, 300);  mPlayerSprite.setPosition(initPoint);  **this**.addChild(mPlayerSprite);  // //从完全透明到不透明  // CCFadeIn fadeIn = CCFadeIn.action(3);  // //从不透明到透明  // CCFadeOut fadeOut = CCFadeOut.action(3);  // mPlayerSprite.runAction(fadeOut);    //颜色变换动画  // ccColor3B color3B = ccColor3B.ccc3(-120, 255, -120);  // CCTintTo tintTo = CCTintTo.action(3, color3B);  // //在当前精灵颜色的基础上再加一个值  // CCTintBy tintBy = CCTintBy.action(3, color3B);    //重复动画  CGPoint targetPoint = CGPoint.*ccp*(400, 300);  CCMoveTo moveTo = CCMoveTo.*action*(2, targetPoint);  CCMoveTo moveTo2 = CCMoveTo.*action*(2, initPoint);  CCSequence seq = CCSequence.*actions*(moveTo, moveTo2);  //用于指定动作的次数  CCRepeat repeat = CCRepeat.*action*(seq, 3);  CCRepeatForever repeatFE = CCRepeatForever.*action*(seq);  mPlayerSprite.runAction(repeatFE); |

## 5、触摸事件

|  |
| --- |
|  |

//构造方法设置接受触摸

|  |
| --- |
| //设置接受  **this**.setIsTouchEnabled(**true**); |

|  |
| --- |
| //当用户开始触摸屏幕时，此方法被回调  //ccoverToGL可以将以左上角为原点的坐标，转换成以左下角为原点的坐标系坐标  @Override  **public** **boolean** ccTouchesBegan(MotionEvent event) {  Log.*i*(*TAG*, "ccTouchesBegan(),x:" + event.getX() + ",y:" + event.getY());  CGPoint p1 = CGPoint.*ccp*(event.getX(),event.getY());  CGPoint p2 = CCDirector.*sharedDirector*().convertToGL(p1);  Log.*i*(*TAG*, "ccTouchesBegan(),ccx:" + p2.x + ",ccy:" + p2.y);  **return** **super**.ccTouchesBegan(event);  }  //当用户手指离开屏幕时，此方法被回调  @Override  **public** **boolean** ccTouchesEnded(MotionEvent event) {  Log.*i*(*TAG*, "ccTouchesEnded(),x:" + event.getX() + ",y:" + event.getY());  CGPoint p1 = CGPoint.*ccp*(event.getX(),event.getY());  CGPoint p2 = CCDirector.*sharedDirector*().convertToGL(p1);  Log.*i*(*TAG*, "ccTouchesEnded(),ccx:" + p2.x + ",ccy:" + p2.y);  **return** **super**.ccTouchesEnded(event);  }    //当用户手指在屏幕上在屏幕上移动时，执行该方法  @Override  **public** **boolean** ccTouchesMoved(MotionEvent event) {  Log.*i*(*TAG*, "ccTouchesMoved(),x:" + event.getX() + ",y:" + event.getY());  CGPoint p1 = CGPoint.*ccp*(event.getX(),event.getY());  CGPoint p2 = CCDirector.*sharedDirector*().convertToGL(p1);  Log.*i*(*TAG*, "ccTouchesMoved(),ccx:" + p2.x + ",ccy:" + p2.y);  **return** **super**.ccTouchesMoved(event);  } |

由于oppo这个手机有个模拟键盘，导致竖屏时View显示不全。

所以测试下从CCDirector类可以获得的屏幕大小值：

|  |
| --- |
| //测试屏幕大小  CGSize winSize = CCDirector.*sharedDirector*().winSize();  CGSize refSize = CCDirector.*sharedDirector*().winSizeRef();  CGSize disSize = CCDirector.*sharedDirector*().displaySize();  org.cocos2d.opengl.GLSurfaceView v = CCDirector.*sharedDirector*().getOpenGLView();  Log.*d*(*TAG*, "winSize(),width:" + winSize.width + ",height:" +winSize.height);  Log.*d*(*TAG*, "winSizeRef(),width:" + refSize.width + ",height:" +refSize.height);  Log.*d*(*TAG*, "displaySize(),width:" + disSize.width + ",height:" +disSize.height);  Log.*d*(*TAG*, "getOpenGLView(),width:" + v.getWidth() + ",height:" +v.getHeight()); |
| Log：  winSize(),width:972.0,height:1620.0  winSizeRef(),width:972.0,height:1620.0  displaySize(),width:1080.0,height:1800.0  getOpenGLView(),width:0,height:0 |

总结：winSize()和winSizeRef()值相同，都表示OpenGL显示的尺寸

而displaySize()是屏幕的实际大小。

getOpenGLView()获得View的大小不可以用。

所以cocos2d显示大小用winSize()

屏幕大小用:displaySize();

如果场景和图层大小等于显示大小，也可以用图层大小或场景大小来获取：

|  |
| --- |
| CGSize layerSize = **this**.getContentSize();  Log.*d*(*TAG*, "layer size:width:" + layerSize.width + ",height:" + layerSize.height); |
| layer size:width:972.0,height:1620.0 |

## 6、时间调度