移动应用开发



传感器应用实践报告

|  |  |
| --- | --- |
| 学 院： | 计算机与信息技术学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | 计科1602 |
| 姓 名： | 麻锦涛 |
| 学 号： | 16281262 |

2019年 6月 2日

1. **功能介绍**

“Gravity Sensor”是一款实现**手机重力传感器**的值自动报数以及提示用户手机出于水平/向左翻/向右翻/向上翻/向下翻以及屏幕向上/向下状态的APP

1. **具体实现**
2. MainActivity

package com.himi;

import android.app.Activity;

import android.os.Bundle;

import android.view.Window;

import android.view.WindowManager;

public class MainActivity extends Activity {

    public static MainActivity ma;

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        ma = this;

        getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);

        this.requestWindowFeature(Window.FEATURE\_NO\_TITLE);

        setContentView(new MySurfaceView(this));

    }

}

1. Activity\_Main\_xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

>

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/hello"

/>

</LinearLayout>

1. MySurfaceView：

这里的要点如下：（1）继承SurfaceView父类和SurfaceHolder.Callback接口，并实现surfaceCreated(SurfaceHolder holder)，surfaceChanged，surfaceDestroyed方法来管理surfaceview的生命周期。

（2）构造方法中通过SurfaceHolder来设置this为callback。

        sfh = this.getHolder(); // 获得SurfaceHolder对象

        sfh.addCallback(this); // 为SurfaceView添加状态监听

**具体实现的步骤如下：**

**1）创建SurfaceView**

创建自定义的SurfaceView继承自SurfaceView,并实现两个接口:SurfaceHolder.Callback和Runnable.代码如下：

public class MySurfaceView extends SurfaceView implements Callback, Runnable {

通过实现这两个接口，就需要在自定义的SurfaceView中实现接口的方法，对于SurfaceHolder.Callback方法，需要实现如下方法，其实就是SurfaceView的生命周期：

    public void surfaceCreated(SurfaceHolder holder) {

        arc\_x = this.getWidth() / 2 - 25;

        arc\_y = this.getHeight() / 2 - 25;

        th.start();

    }

    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {

    }

    public void surfaceDestroyed(SurfaceHolder holder) {

    }

对于Runnable接口，需要实现run()方法：

    public void run() {

        // TODO Auto-generated method stub

        while (true) {

            draw();

            try {

                Thread.sleep(100);

            } catch (Exception ex) {

            }

        }

    }

**（2）初始化SurfaceView**

在自定义的MySurfaceView的构造方法中，需要对SurfaceView进行初始化，包括SurfaceHolder的初始化、画笔的初始化等。在自定义的SurfaceView中，这里需要定义以下几个成员变量:

    private Thread th = new Thread(this);

    private SurfaceHolder sfh;

    private Canvas canvas;//绘图的画布

    private Paint paint;

    private SensorManager sm;

    private Sensor sensor;//创建传感器管理器

    private SensorEventListener mySensorListener;//创建传感器监听器

    private int arc\_x, arc\_y;// 圆形的x,y位置

    private float x = 0, y = 0, z = 0;//重力传感器的(x,y,z)

在SurfaceHolder里面保存了一个对Surface对象的引用，而我们执行绘制方法本质上就是操控Surface。SurfaceHolder因为保存了对Surface的引用，所以使用它来处理Surface的生命周期。（SurfaceView的生命周期其实就是Surface的生命周期）例如使用 SurfaceHolder来处理生命周期的初始化。

初始化代码如下：

    public MySurfaceView(Context context) {

        super(context);

        this.setKeepScreenOn(true);

        sfh = this.getHolder();

        sfh.addCallback(this);

        paint = new Paint();

        paint.setAntiAlias(true);

        setFocusable(true);

        setFocusableInTouchMode(true);

        //传感器管理对象

        sm = (SensorManager) MainActivity.ma.getSystemService(Service.SENSOR\_SERVICE);

        sensor = sm.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER);

//创建一个重力传感器实例

        //TYPE\_ACCELEROMETER 加速度传感器的类型——重力传感器

        //TYPE\_ALL 所有类型的传感器。

        mySensorListener = new SensorEventListener() {

            @Override

            //传感器获取值发生改变时函数发生响应

            public void onSensorChanged(SensorEvent event) {//备注1

                //这里处理传感器获取值发生的改变

                x = event.values[0]; //手机左右翻转

                //x>0 左翻 x<0右翻

                y = event.values[1]; //手机上下翻转

                //y>0 下翻 y<0上翻

                z = event.values[2]; //屏幕朝上/朝下

                //z>0 朝上 z<0 朝下

                arc\_x -= x;

                arc\_y += y;

            }

            //传感器精度发生改变时函数响应

            public void onAccuracyChanged(Sensor sensor, int accuracy) {

                // TODO Auto-generated method stub

            }

        };

        sm.registerListener(mySensorListener, sensor, SensorManager.SENSOR\_DELAY\_GAME);

        //第一个参数指的是传感器监听器，第二个是需要监听的传感实例

        //最后一个参数指的是监听的传感器速率类型： 一共有以下四种形式

        //SENSOR\_DELAY\_NORMAL 正常

        //SENSOR\_DELAY\_UI 适合界面

        //SENSOR\_DELAY\_GAME 适应游戏

        //SENSOR\_DELAY\_FASTEST 最快

    }

（3）使用SurfaceView

通过SurfaceHolder对象的lockCanvans()方法，我们可以获取当前的Canvas绘图对象。接下来的操作就和自定义View中的绘图操作一样了。需要注意的是这里获取到的Canvas对象还是继续上次的Canvas对象，而不是一个新的对象。因此，之前的绘图操作都会被保留，如果需要擦除，则可以在绘制前，通过drawColor()方法来进行清屏操作。

绘制的时候，在surfaceCreated()方法中进行绘制（x，y的坐标值）

    public void surfaceCreated(SurfaceHolder holder) {

        arc\_x = this.getWidth() / 2 - 25;

        arc\_y = this.getHeight() / 2 - 25;

        th.start();

    }

Canvas.drawColor() 使用srcover porterduff模式使用指定的颜色填充整个画布的位图

            if (canvas != null) {

                canvas.drawColor(Color.BLACK);//背景设置为黑色

                paint.setColor(Color.RED);//移动的点为红色

  paint.setColor(Color.YELLOW);//提示语句的颜色设置为黄色

Canvas.drawArc() 是Canvas绘制圆弧的方法，这里绘制了一个红色的圆形（360），即填充圆弧并且含圆心（**true**）

   canvas.drawArc(new RectF(arc\_x, arc\_y, arc\_x + 50,arc\_y + 50), 0, 360, true, paint);

//绘制圆弧，含圆心

Canvas.drawText() 是 Canvas 最基本的绘制文字的方法：给出文字的内容和位置， Canvas 按要求去绘制文字。方法的参数很简单： text 是文字内容，x 和 y 是文字的坐标。drawText() 方法参数中的 y 值，就是指定的基线（**baseline**）的位置。

 canvas.drawText("当前重力传感器的值:", arc\_x - 50, arc\_y-30, paint);

                canvas.drawText("x=" + x + ",y=" + y + ",z=" + z,

                        arc\_x - 50, arc\_y, paint);

我在这里又声明三个变量，名为temp\_str，temp\_str2，temp\_str3，类型为**String**，根据**(x,y,z)值的变化**来提示：

（1）手机是否处于水平放置/向左翻/向右翻/向上翻/向下翻的状态；

（2）手机是否屏幕向上/向下；

String temp\_str = "Himi提示： ";

                String temp\_str2 = "";

                String temp\_str3 = "";

                if (x < 1 && x > -1 && y < 1 && y > -1) {

                    temp\_str += "当前手机处于水平放置的状态";

                    if (z > 0) {

                        temp\_str2 += "并且屏幕朝上";

                    } else {

                        temp\_str2 += "并且屏幕朝下,提示别躺着玩手机，对眼睛不好哟~";

                    }

                } else {

                    if (x > 1) {

                        temp\_str2 += "当前手机处于向左翻的状态";

                    } else if (x < -1) {

                        temp\_str2 += "当前手机处于向右翻的状态";

                    }

                    if (y > 1) {

                        temp\_str2 += "当前手机处于向下翻的状态";

                    } else if (y < -1) {

                        temp\_str2 += "当前手机处于向上翻的状态";

                    }

                    if (z > 0) {

                        temp\_str3 += "并且屏幕朝上";

                    } else {

                        temp\_str3 += "并且屏幕朝下,提示别躺着玩手机，对眼睛不好哟~";

                    }

                }

paint.setTextSize(20);

                canvas.drawText(temp\_str, 0, 50, paint);

                canvas.drawText(temp\_str2, 0, 80, paint);

                canvas.drawText(temp\_str3, 0, 110, paint);

            }

        } catch (Exception e) {

            Log.v("Himi", "draw is Error!");

        }

同时我通过调用Thread.Sleep可以暂停/阻塞线程。调用Sleep()和意味着线程将不再得到CPU时间，当前线程将休眠100ms

    @Override

    public void run() {

        // TODO Auto-generated method stub

        while (true) {

            draw();

            try {

                Thread.sleep(100);

            } catch (Exception ex) {

            }

        }

    }

此前在绘制的逻辑中通过sfh.lockCanvas()方法获取Canvas对象进行绘制，最后通过sfh.unlockCanvasAndPost(mCanvas)方法对画布内容进行提交。

finally {

            sfh.unlockCanvasAndPost(canvas);

//保证每次可以提交绘图的内容

        }

1. **特色之处**

**1.安卓重力传感器详解：**

重力传感器简称GV-sensor，输出重力数据。在地球上，重力数值为9.8，单位是m/s^2。根据我们的初中物理知识，坐标系统与加速度传感器相同。当设备复位时，重力传感器的输出与加速度传感器相同。

假设当地的重力加速度值为g

当手机正面朝上的时候，z的值为g，反面朝上的时候，z的值为-g

当手机右侧面朝上的时候，x的值为g,右侧面朝上的时候,x的值为-g

当手机上侧面朝上的时候，y的值为g,右侧面朝上的时候,y的值为-g

private float x = 0.0F;

private float y = 0.0F;

private float z = 0.0F;

首先我们创建一个传感器管理器和一个传感器监听器，管理器用来管理传感器以及创建各种各样的传感器，监听器用来监视传感器的变化并且进行相应的操作

private SensorManager sm; //创建传感器管理器

private Sensor sensor;

private SensorEventListener mySensorListener; //创建传感器监听器

在页面中使用setKeepScreenOn保持屏幕常亮

this.setKeepScreenOn(true);

这个监听器是我自己定义的，在重力感应器感应（应该可以说叫“监听”）到手机位置有变化的时候，就可以采取相应的操作将x,y,z的值打印出来

this.sm = (SensorManager)MainActivity.ma.getSystemService("sensor");

this.sensor = this.sm.getDefaultSensor(1);//

this.mySensorListener = new SensorEventListener() {

//实现SensorEventListen接口，重写onSensorChanged和onAccuracyChanged的方法

public void onSensorChanged(SensorEvent event) {

MySurfaceView.this.x = event.values[0];

MySurfaceView.this.y = event.values[1];

MySurfaceView.this.z = event.values[2];

MySurfaceView.this.arc\_x = (int)((float)MySurfaceView.this.arc\_x - MySurfaceView.this.x);

MySurfaceView.this.arc\_y = (int)((float)MySurfaceView.this.arc\_y + MySurfaceView.this.y);

}

//可以得到传感器实时测量出来的变化值

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}

};

this.sm.registerListener(this.mySensorListener, this.sensor, 1);

**2. SurfaceView**

在Android系统中，有一种特殊的视图，称为SurfaceView，它拥有独立的绘图表面，即它不与其宿主窗口共享同一个绘图表面。由于拥有独立的绘图表面，因此SurfaceView的UI就可以在一个独立的线程中进行绘制。又由于不会占用主线程资源，SurfaceView一方面可以实现复杂而高效的UI，另一方面又不会导致用户输入得不到及时响应。这里我总结归纳了一下SurfaceView和View不同之处有以下几点：

（1） SurfaceView允许其他线程更新视图对象(执行绘制方法)而View不允许这么做，它只允许UI线程更新视图对象。

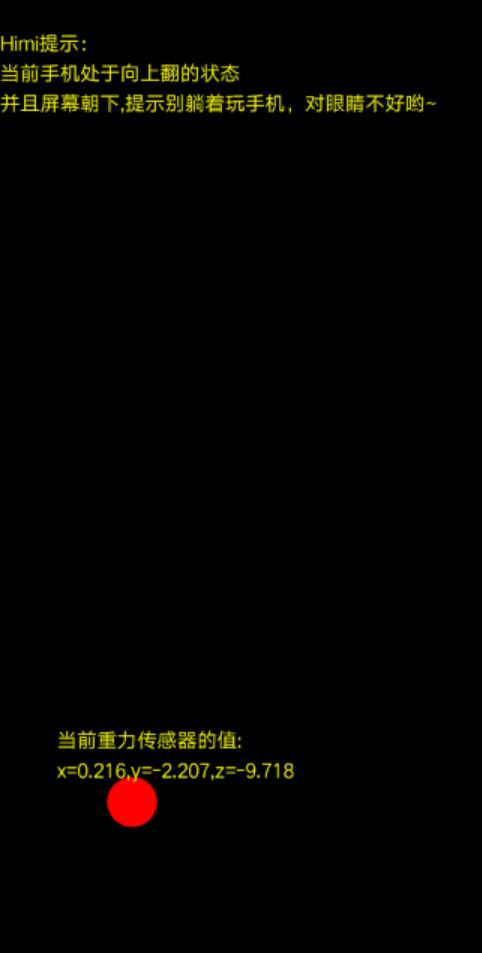
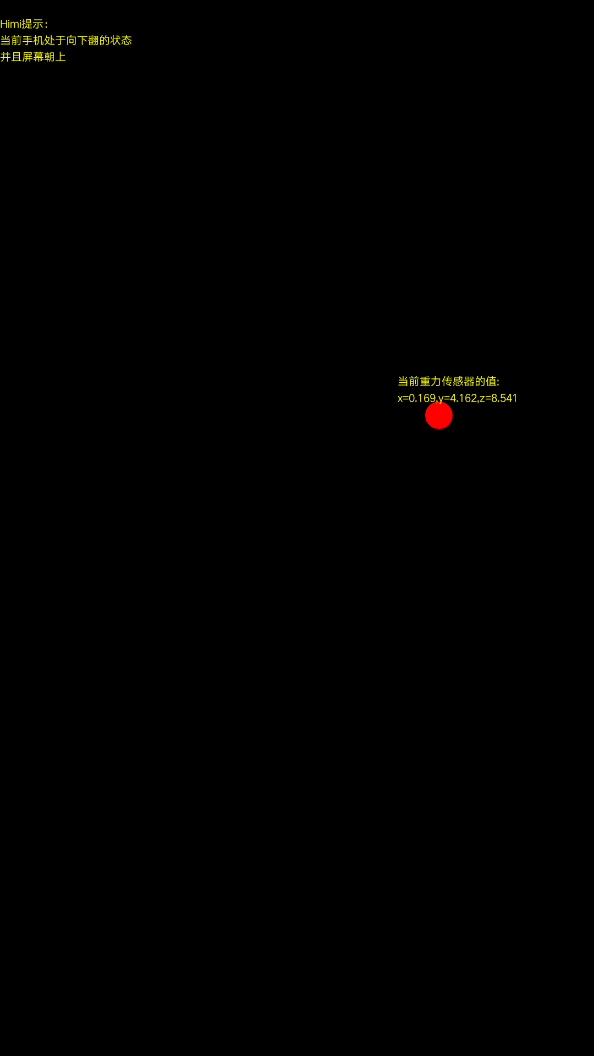
（2） SurfaceView是放在其他最底层的视图层次中，所有其他视图层都在它上面，所以在它之上可以添加一些层，而且它不能是透明的。

（3）它执行动画的效率比View高，而且你可以控制帧数。

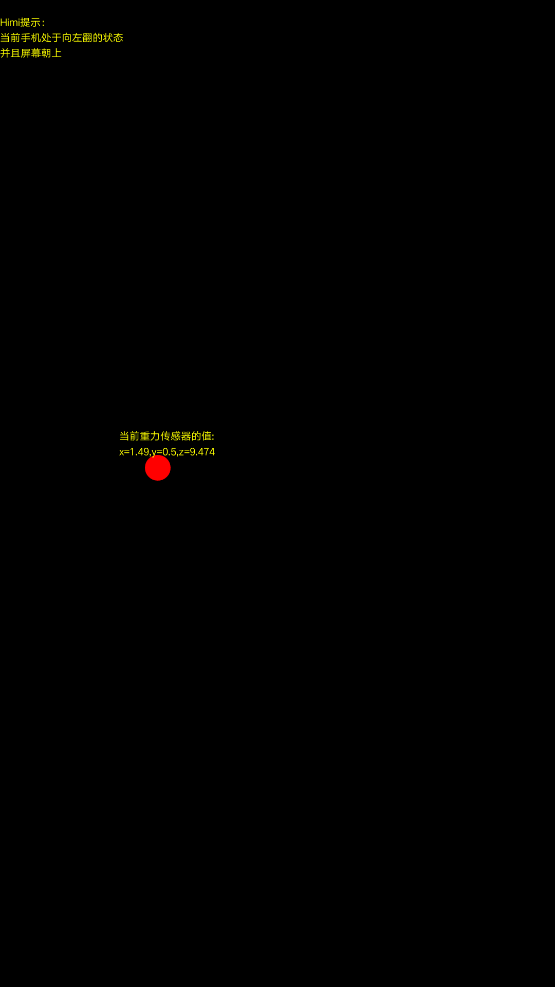
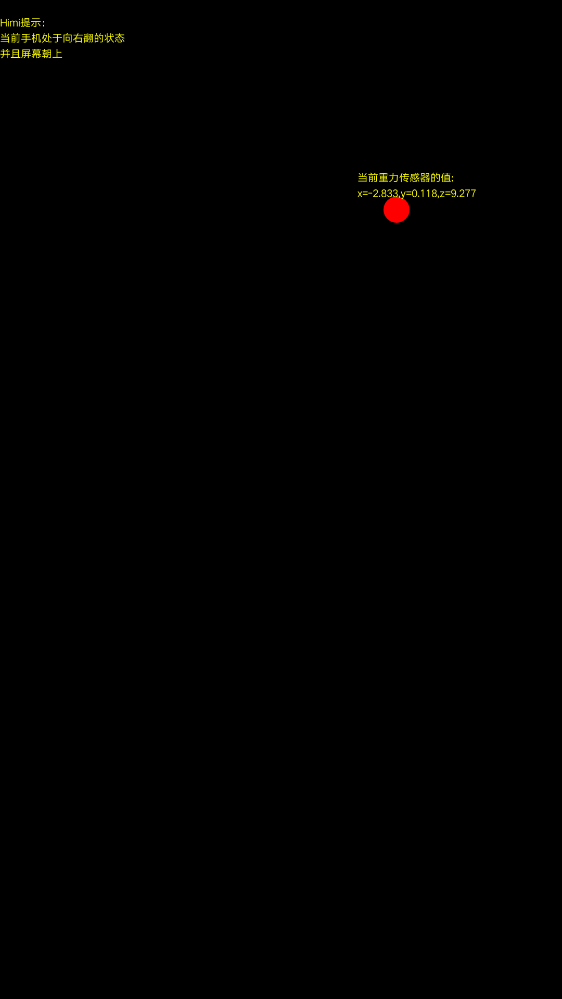
（4）SurfaceView在绘图时使用了双缓冲机制，而View没有。

1. **效果展示**

主要实现了相册的3D跳转效果，这里以本人最爱的女星苍井空小姐姐为例：

向上翻转 向下翻转

向左翻转 向右翻转

屏幕朝上水平放置 屏幕朝下水平放置

**个人认为这项技术可以用于矫正当代大学生玩手机的正确姿势。比如说，有人在床上玩手机总喜欢躺着玩，长期躺着打游戏或者看电子书会导致近视。如果锁屏加上这项技术，小伙伴们就可以坐起来玩手机啦！！！**

**还可以开发一款锻炼小朋友平衡感的小游戏！！！**

