宝贵建议请发送至：wangzhenyang@itcast.cn

黑马程序员 itheima.com -编程，始于黑马

Android 课程同步笔记

Alpha 0.01 版

By 阳哥

Android 手机卫士-05

1.自定义吐司（★★★★）

因为目前的土司比较丑而且无法控制显示的时间，又没有提供直接修改的接口，为了满足我们的需求，所以我们现在实现一个自定义的土司。

首先我们通过查阅源码可知，土司其实就是显示在 WindowManager 上面的一个 View，也就是直接显示在窗口上面。它的生命周期不随着 Activity 的生命周期变化而变化。所以我们可以直接通过 WindowManager.addView(View,Params)来显示。

步骤：我们在上一个文档中用到了自定义吐司。这里直接修改 AddressService 类。

1、声明 WindowManager 对象以及其他参数对象

```
// 窗体服务对象
private WindowManager wm;
private View view;
//窗体对象布局参数
private WindowManager.LayoutParams params;
```

2、在 onCreate 方法中，初始化参数

```
//获取 WindowManager 对象
wm = (WindowManager) getSystemService(WINDOW_SERVICE);
```

3、在类中定义 myToast 方法，在该方法中实现自定义吐司的核心逻辑

```
public void myToast(String address) {
        //从 sp 中获取主题样式的脚标值
        int which = sp.getInt("which", 0);
        //不同主题对应的背景资源
        int[] its = new int[] { R.drawable.call_locate_white,
```

```
R.drawable.call_locate_orange, R.drawable.call_locate_blue,
R.drawable.call_locate_gray, R.drawable.call_locate_green };
        //填充一个 LinearLayout 对象，作为自定义吐司的显示样式
        LinearLayout layout = (LinearLayout) View.inflate(this,
R.layout.show_address, null);
        //设置背景颜色资源
        layout.setBackgroundResource(its[which]);
        //设置 TextView 控件对象
        TextView et_address = (TextView) layout.findViewById(R.id.tv_address);
        et_address.setText(address);
        et_address.setTextColor(Color.RED);
        et_address.setTextSize(20);
        //初始化窗体布局参数
        params = new WindowManager.LayoutParams();
        //设置窗体的对齐方式
        params.gravity = Gravity.TOP+Gravity.LEFT;
        //设置窗体的尺寸
        params.height = WindowManager.LayoutParams.WRAP_CONTENT;
        params.width = WindowManager.LayoutParams.WRAP_CONTENT;
        //设置窗体对象是否允许获取焦点，同时屏幕常亮
        params.flags = WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE
        | WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON;
        //设置窗体对象的透明属性
        params.format = PixelFormat.TRANSLUCENT;
        //设置窗体类型
        params.type = WindowManager.LayoutParams.TYPE_PRIORITY_PHONE;
        view = layout;
        //设置布局的事件监听器，在该监听器中实现自定义窗体对象的移动
        view.setOnTouchListener(new OnTouchListener() {
            //初始位置
            int startX = 0;
            int startY = 0;
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    //当手指按下时获取当前坐标
                    startX = (int) event.getRawX();
                    startY = (int) event.getRawY();
                    break;
                case MotionEvent.ACTION_MOVE:
                    //移动的时候获取最新坐标
```

```
                int newX = (int) event.getRawX();
                int newY = (int) event.getRawY();
                //计算移动的位移
                int dx = newX-startX;
                int dy = newY-startY;
                //窗体对象位置相应的做改变
                params.x = params.x+dx;
                params.y = params.y+dy;
                //如下判断是不允许窗体对象跑出屏幕
                if (params.x<0) {
                    params.x = 0;
                }
                if (params.y<0) {
                    params.y=0;
                }
                if (params.x>wm.getDefaultDisplay().getWidth()-view.getWidth()) {
                    params.x = wm.getDefaultDisplay().getWidth()-view.getWidth();
                }
                if (params.y>wm.getDefaultDisplay().getHeight()-view.getHeight())
{
                    params.y = wm.getDefaultDisplay().getHeight()-view.getHeight();
                }
                //更新窗体对象的位置
                wm.updateViewLayout(view, params);
                //新坐标赋值给当前坐标
                startX = newX;
                startY = newY;
                break;
            case MotionEvent.ACTION_UP:
                //放手的时候记录当前坐标，并保存在 sp 中，这样下次再来的时候位置可以直接
使用
                Editor editor = sp.edit();
                editor.putInt("lastX", params.x);
                editor.putInt("lastY", params.y);
                editor.commit();
                break;
            default:
                break;
            }
            return true;
        }
    });
    int lastX = sp.getInt("lastX", -1);
```

```
        int lastY = sp.getInt("lastY", -1);
        if (lastX>0) {
            params.x = lastX;
        }
        if (lastY>0) {
            params.y = lastY;
        }
        //在窗体上添加自定义 view 对象
        wm.addView(view, params);
    }
}
```

2.自定义多连击事件（★★★）

2.1 自定义双击事件

双击的原理： 当第一次点击的时候记录下点击的时间 A，然后第二次点击的时候获取到第二次的时间 B。如果 B - 500 的差值小于上一次点击的时间的话，则认为这两次点击是一次双击事件。

```
private long[] mHits = new long[2];
    public void dbclick(View view) {
        // 把 mHits[1]赋给 mHits[0]
        System.arraycopy(mHits, 1, mHits, 0, mHits.length - 1);
        // 重新给 mHits[1]赋值
        mHits[mHits.length - 1] = SystemClock.uptimeMillis();
        if (mHits[0] >= (SystemClock.uptimeMillis() - 500)) {
            Toast.makeText(this, "双击", Toast.LENGTH_SHORT).show();
        }
    }
```

2.2 自定义多击事件

多击事件的原理其实跟双击事件是一样的。我们只需要多少次连击就定义一个多长数组即可。

```
public void click(View view) {
        //实际效果就是将数组第一个去掉，后面的都前移一位
        System.arraycopy(hints, 1, hints, 0, hints.length - 1);
        //给数组的最后一个赋值最新的时间
        hints[hints.length - 1] = SystemClock.uptimeMillis();
        //判断最后点击时间跟数组中保留的第一次点击时间间隔是否小于 500 毫秒，当然这个值是可以改变的
        if (hints[0] >= SystemClock.uptimeMillis() - 500) {
            Toast.makeText(this, "您实现了一个" + hints.length + "连击。", 0).show();
        }
    }
```

3.小火箭（★★★）

小火箭的实现原理：小火箭显示的页面其实是一个透明的 Activity，当拖动的小火箭达到某个位置的时候，执行一系列的动画来实现想要的效果。

为了演示小火箭，我们单独创建一个工程，工程名就叫做小火箭。

步骤：

1、创建一个新 Android 工程，工程名叫小火箭，包名：com.itheima.rocket

2、编写布局文件 activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
   <ImageView
       android:src="@drawable/desktop_rocket_launch_1"
       android:layout_height="wrap_content"
       android:layout_width="wrap_content"
       android:id="@+id/iv"
       />
   <ImageView
       android:layout_alignParentBottom="true"
```

```
        android:src="@drawable/desktop_smoke_m"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/iv_bottom"
        />
    <ImageView
        android:layout_above="@id/iv_bottom"
        android:src="@drawable/desktop_smoke_t"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:id="@+id/iv_top"
        />
</RelativeLayout>
```

3、在 AndroidManifest.xml 中设置主题为透明全屏。

```
  android:theme="@android:style/Theme.Translucent.NoTitleBar"
```

4、编写 MainActivity 类，在该类中实现核心业务逻辑

```
public class MainActivity extends Activity {
    private AnimationDrawable animationDrawable;
    private ImageView iv;
    private WindowManager wm;
    private ImageView iv_top;
    private ImageView iv_bottom;
    private int width = 0;// wm.getDefaultDisplay().getWidth();
    private int height = 0;// wm.getDefaultDisplay().getHeight();
    Handler handler = new Handler() {
        public void handleMessage(android.os.Message msg) {
            if (msg.what == RESULT_OK) {
                //火箭发射后重新让火箭显示在初始位置，同时将烟雾掩藏
                iv_bottom.setVisibility(View.INVISIBLE);
                iv_top.setVisibility(View.INVISIBLE);
                iv.requestLayout();
            } else {
                int y = (Integer) msg.obj;
                //更新火箭的坐标
                iv.layout(iv.getLeft(), y, iv.getRight(), y + iv.getHeight());
            }
        };
```

```
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // 获取底部图片 烟雾
        iv_bottom = (ImageView) findViewById(R.id.iv_bottom);
        // 获取顶部图片 烟雾
        iv_top = (ImageView) findViewById(R.id.iv_top);
        // 底部和顶部图片设置不可见
        iv_bottom.setVisibility(View.INVISIBLE);
        iv_top.setVisibility(View.INVISIBLE);
        // 获取窗体管理对象
        wm = (WindowManager) getSystemService(WINDOW_SERVICE);
        // 获取小火箭图片对象的控件
        iv = (ImageView) findViewById(R.id.iv);
        // 设置火箭的背景资源
        iv.setBackgroundResource(R.drawable.rocket);
        // ImageView 的 AnimationDrawable 对象
        animationDrawable = (AnimationDrawable) iv.getBackground();
        // 开始动画 这个动画的效果是火箭的火焰不停的变化，跟真的在喷火似的，其实是两张动画不停的轮换
        animationDrawable.start();
        // 获取屏幕宽度
        width = wm.getDefaultDisplay().getWidth();
        // 获取屏幕高度
        height = wm.getDefaultDisplay().getHeight();
        // 给火箭控件添加监听事件
        iv.setOnTouchListener(new OnTouchListener() {
            int startX = 0;
            int startY = 0;

            @Override
            public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    // 记录当前坐标
                    startX = (int) event.getRawX();
                    startY = (int) event.getRawY();
                    break;
                case MotionEvent.ACTION_MOVE:
                    // 滑动的时候记录新坐标
```

```
                int newX = (int) event.getRawX();
                int newY = (int) event.getRawY();
                int dx = newX - startX;
                int dy = newY - startY;
                iv.layout(iv.getLeft() + dx, iv.getTop() + dy, iv.getRight() + dx,
iv.getBottom() + dy);
                startX = newX;
                startY = newY;
                // 获取火箭的左侧坐标和顶部坐标
                int left = iv.getLeft();
                int top = iv.getTop();
                // 当火箭左侧坐标位于屏幕 1/3 和 2/3 中间并且顶部大于屏幕高度的 5/8 时显示
喷射的火焰
                if (left > width / 3 && left < width * 2 / 3 && top > height * 5 /
8) {
                    iv_bottom.setVisibility(View.VISIBLE);
                    iv_top.setVisibility(View.INVISIBLE);
                } else {
                    iv_bottom.setVisibility(View.INVISIBLE);
                    iv_top.setVisibility(View.INVISIBLE);
                }
                break;
            case MotionEvent.ACTION_UP:

                left = iv.getLeft();
                top = iv.getTop();
                if (left > width / 3 && left < width * 2 / 3 && top > height * 5 /
8) {
                    Toast.makeText(MainActivity.this, "火箭发射了", 0).show();
                    // 执行发射火箭方法
                    sendRocket();
                    // 喷射的烟雾延迟 1 秒逐渐消失
                    AlphaAnimation alphaAnimation = new AlphaAnimation(0f, 1f);
                    alphaAnimation.setDuration(1000);
                    alphaAnimation.setRepeatCount(1);
                    alphaAnimation.setRepeatMode(AlphaAnimation.REVERSE);
                    iv_bottom.startAnimation(alphaAnimation);
                    iv_top.startAnimation(alphaAnimation);
                }
                break;
            default:
                break;
            }
```

```
                return true;
            }
        });
    }

    // 火箭发射
    private void sendRocket() {
        new Thread(new Runnable() {

            @Override
            public void run() {
                // 为了能看清效果我们分 21 次完成火箭从底部到顶部的操作
                for (int i = 0; i < 21; i++) {
                    // 计算火箭的 y 轴方向的坐标
                    int y = height - ((i) * (height + iv.getHeight())) / 20;
                    Message msg = handler.obtainMessage();
                    msg.obj = y;
                    handler.sendMessage(msg);
                    try {
                        Thread.sleep(50);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                handler.sendEmptyMessage(RESULT_OK);
            }
        }).start();
    }
}
```

在上面的代码中，我们给 ImageView 控件设置了动画背景资源。在 drawable 目录下创建 rocket.xml，清单如下

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/desktop_rocket_launch_1"
android:duration="200" />
    <item android:drawable="@drawable/desktop_rocket_launch_2"
android:duration="200" />
</animation-list>
```

上面的动画的 android:oneshot="false"意思是动画不止播放一次，而是不停的重复播放。

4.创建快捷方式（★★★）

创建快捷方式就是在系统桌面创建一个图标，点击该图标会执行一段操作，比如打开某个原因，拨打电话、发送短信等等。

我们在我们的手机卫士项目的 SplashActivity 类中添加，创建快捷键方法，当手机卫士安全的时候创建快捷键。

```
//创建桌面快捷键，实现一键呼叫功能
    private void createShortCut() {
        sp = getSharedPreferences("config", MODE_PRIVATE);
        //将创建快捷键后的信息保存在 sp 中，这样不至于每次安全都创建快捷键
        boolean installed = sp.getBoolean("shortcutInstalled", false);
        //如果没有安装则安装
        if (!installed) {
            Intent intent = new Intent();
            intent.setAction("com.android.launcher.action.INSTALL_SHORTCUT");
            //指定图标
            intent.putExtra(Intent.EXTRA_SHORTCUT_ICON,
BitmapFactory.decodeResource(getResources(), R.drawable.app));
            //指定名称
            intent.putExtra(Intent.EXTRA_SHORTCUT_NAME, "手机卫士");
            //创建一个 intent 用于，处理点击图标的时候触发的意图
```

```
        Intent callIntent = new Intent();
        callIntent.setAction("com.ithema.mobileSafe.entryHome");
        //将新创建的意图添加到快捷键 Intent 中，其实创建快捷键本身就是一个意图
        intent.putExtra(Intent.EXTRA_SHORTCUT_INTENT, callIntent);
        //将创建快捷键意图作为广播发送出去
        sendBroadcast(intent);
        Editor editor = sp.edit();
        editor.putBoolean("shortcutInstalled", true);
        editor.commit();
    }
}
```

上面创建的快捷键的作用是点击时进入手机卫士主界面。

至此，本文档完！

2015 年 1 月 7 日 星期三 17:16:45
北京市中关村软件园国际软件大厦