最近一直在研究关于游戏编程，人在深圳，工作不好找啊！

发一个斗地主游戏的牌桌实现。

为了节约内存资源，每张扑克牌都是剪切形成的，当然这也是当前编程的主流方法。

1、主Activity

[java] view plain copy print ?

```java
<span style="font-size:18px;color:#3333ff;">package com.bison;

import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;

/**
 * 求某公司包养
 *
 * @author Bison
 *
 */
public class PukeActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 这个事隐藏标题栏，不解释
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        // 隐藏状态栏，你懂的
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        /*
         * 开始有考虑使屏幕上扑克的排列随屏幕的分辨率变动 结果貌似不好做，注释掉了 Display display =
         * getWindowManager().getDefaultDisplay(); int screenWidth =
         * display.getWidth(); int screenHeight = display.getHeight();
         */

        // 使用代码锁定横屏
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
        // setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);这个是竖屏
        setContentView(new GameView(this));
    }
}</span>
```

2、牌桌页面

[java] view plain copy print ?

```java
<span style="color:#3333ff;">package com.bison;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Rect;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

import com.bison.utils.Person;

/**
 * 牌桌，会被老婆骂，最好不要上去，你懂的
 *
 * 扑克图片来源，和牌桌背景在文章的下面。 扑克背面图等我没上传，玩家自行百度
 *
 * @author Bison
 *
 */
public class GameView extends SurfaceView implements SurfaceHolder.Callback {
    private FlushThread thread = null;// 刷帧线程
    private Bitmap sourceBitmap = null;// 扑克图片来源
    private Bitmap backgroundDesk = null;// 牌桌背景
    private Bitmap backgroundPuke = null;// 扑克背面

    private final Person person;
    private int pukeWidth = 0;// 扑克的宽
    private int pukeHeight = 0;// 扑克的高
    private int deskWidth = 0;// 牌桌的宽
    private int deskHeight = 0;// 牌桌的高
    private int left = 0;// 我自己首张牌左距离

    public GameView(Context context) {
        super(context);
        getHolder().addCallback(this);
        this.thread = new FlushThread(getHolder(), this);// 实例化线程
        initBitmap();// 实例化图片
        this.person = new Person();// 实例化Person类
        this.left = deskWidth / 2 - (16 * 25 + pukeWidth) / 2;// 左距开始时赋值
    }

    private void initBitmap() {// 初始化图片
        sourceBitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.smallcard);
        pukeWidth = sourceBitmap.getWidth() / 14;// 每张扑克的宽高
        pukeHeight = sourceBitmap.getHeight() / 4;

        backgroundDesk = BitmapFactory.decodeResource(getResources(),
                R.drawable.gameback2);

        deskWidth = backgroundDesk.getWidth();// 牌桌的宽高
        deskHeight = backgroundDesk.getHeight();

        backgroundPuke = BitmapFactory.decodeResource(getResources(),
                R.drawable.cardback);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        // 绘制牌桌
        canvas.drawBitmap(backgroundDesk, 0, 0, null);
        personPaint(canvas, pukeWidth, pukeHeight);
        deskthreePukes(canvas, pukeWidth, pukeHeight);
    }

    /** 绘制每个玩家手里的牌 */
    public void personPaint(Canvas c, int pukeWidth, int pukeHeight) {
        Rect src = new Rect();
        Rect dst = new Rect();

        // 遍历数组
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 17; j++) {
                if (i == 0) {// 左手边玩家，不用绘出正面
                    // src = person.cardRect(person.person1[j], pukeWidth,
                    // pukeHeight);
                    // dst.set(10, j * 20, 10 + pukeWidth, j * 20 + pukeHeight);
                    c.drawBitmap(backgroundPuke, 35, 85, null);
                }
                if (i == 1) {// 自己
                    src = person.cardRect(person.person2[j], pukeWidth,
                            pukeHeight);
                    dst.set(left + j * 25, this.deskHeight - 20 - pukeHeight,
                            left + j * 25 + pukeWidth, deskHeight - 20);
                    c.drawBitmap(sourceBitmap, src, dst, null);
                }
                if (i == 2) {// 右手边玩家，同样不用绘出正面
                    // src = person.cardRect(person.person3[j], pukeWidth,
                    // pukeHeight);
                    // dst.set(this.screenWidth - 10 - pukeWidth, j * 20,
                    // this.screenWidth - 10, j * 20 + pukeHeight);
                    c.drawBitmap(backgroundPuke, deskWidth - 35 - pukeWidth,
                            85, null);
                }
            }
        }
    }

    /** 绘制三张底牌 */
    private void deskthreePukes(Canvas c, int pukeWidth, int pukeHeight) {
        Rect src = new Rect();
        Rect dst = new Rect();
        for (int i = 0; i < 3; i++) {
            src = person.cardRect(person.threePukes[i], pukeWidth, pukeHeight);
            dst.set(280 + i * pukeWidth, 12, 280 + (i + 1) * pukeWidth,
                    12 + pukeHeight);
            c.drawBitmap(sourceBitmap, src, dst, null);
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        // 正在研究点击弹出相应的扑克
        return super.onTouchEvent(event);
    }

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

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        this.thread.setFlag(true);
        this.thread.start();
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        boolean retry = true;
        this.thread.setFlag(false);
        while (retry) {
            try {
                thread.join();
                retry = false;
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

    }

    // 刷帧线程，这个不解释，实在看不懂，M我：289302487@qq.com
    class FlushThread extends Thread {
        private boolean flag = false;
        private final int span = 500;
        private final GameView gameView;
        private final SurfaceHolder holder;

        public FlushThread(SurfaceHolder holder, GameView gameView) {
            this.gameView = gameView;
            this.holder = holder;
        }

        @Override
        public void run() {
            Canvas canvas;
            while (this.flag) {
                canvas = null;
                try {
                    canvas = this.holder.lockCanvas(null);
                    synchronized (this.holder) {
                        this.gameView.onDraw(canvas);
                    }
                } finally {
                    if (canvas != null) {
                        this.holder.unlockCanvasAndPost(canvas);
                    }
                }

                try {
                    Thread.sleep(span);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }

        public boolean isFlag() {
            return flag;
        }

        public void setFlag(boolean flag) {
            this.flag = flag;
        }

    }

}
</span>
```

3、相关实体类

扑克牌类：

[java] view plain copy print ?

```java
<span style="font-size:18px;color:#3333ff;">package com.bison.utils;

import java.util.Random;

/**
 * 生成一副洗好的牌，并且 设计为单例模式
 *
 * @author Bison
 *
 */
public class Cards {
    // 声明一副扑克牌
    public int[] pukes = new int[54];

    private static Cards cardsInstance = null;

    private Cards() {
        setPuke();
        shuffle();
    }

    public static Cards getInstance() {
        if (cardsInstance == null) {
            cardsInstance = new Cards();
        }
        return cardsInstance;
    }

    /** 给54张扑克牌赋值 ：1~54 */
    private void setPuke() {
        for (int i = 0; i < 54; i++) {
            pukes[i] = i + 1;
        }
    }

    /** 洗牌 */
    private void shuffle() {
        Random rdm = new Random();
        for (int i = 0; i < 54; i++) {
            // random.nextInt();是个前闭后开的方法：0~53
            int rdmNo = rdm.nextInt(54);
            int temp = pukes[i];
            pukes[i] = pukes[rdmNo];
            pukes[rdmNo] = temp;
        }
    }
}
</span>
```

玩家类：

[java] view plain copy print ?

```java
<span style="font-size:18px;color:#3333ff;">package com.bison.utils;

import android.graphics.Rect;

/**
 * 这个是玩家的实体类
 *
 * @author Bison
 *
 */
public class Person {
    private final Cards mCards = Cards.getInstance();

    public int[] person1 = new int[17];
    public int[] person2 = new int[17];
    public int[] person3 = new int[17];

    // 余下三张属于地主的
    public int[] threePukes = new int[3];

    public Person() {
        personHold(mCards.pukes);
    }

    /** 分牌 */
    private void personHold(int[] pukes) {
        int k = 0;
        for (int i = 0; i < 3; i++) {
            if (i == 0) {
                for (int j = 0; j < 17; j++) {
                    person1[j] = pukes[k++];
                }
                // 将其排序
                sort(person1);
            }
            if (i == 1) {
                for (int j = 0; j < 17; j++) {
                    person2[j] = pukes[k++];
                }
                // 将其排序
                sort(person2);
            }
            if (i == 2) {
                for (int j = 0; j < 17; j++) {
                    person3[j] = pukes[k++];
                }
                // 将其排序
                sort(person3);
            }
        }

        threePukes[0] = pukes[51];
        threePukes[1] = pukes[52];
        threePukes[2] = pukes[53];
    }

    /** 对每个玩家手里的牌排序:使用冒泡排序 */
    private void sort(int[] ary) {
        for (int i = 0; i < ary.length; i++) {
            for (int j = 0; j < ary.length - i - 1; j++) {
                if (ary[j] > ary[j + 1]) {
                    int temp = ary[j];
                    ary[j] = ary[j + 1];
                    ary[j + 1] = temp;
                }
            }
        }
    }

    /**
     * 对应扑克所在图片上的位置
     * 1 5 9 ……… 53
     * 2 6 10 ……… 54
     * 3 7 11
     * 4 8 12
     */
    public Rect cardRect(int cardValue, int width, int height) {
        int x = 0, y = 0;
        if (cardValue % 4 == 0) {
            x = cardValue / 4 - 1;
            y = 4;
        } else {
            x = cardValue / 4;
            y = cardValue % 4;
        }

        int left = x * width;
        int top = (y - 1) * height;
        int right = (x + 1) * width;
        int bottom = (y) * height;
        return new Rect(left, top, right, bottom);
    }
}
</span>
```

PS：斗地主还是可以做成很复杂的。相关图片