附件 1：

学　号	201725230516
成　绩	

移动终端开发技术

课程设计报告

题　　目	五子棋
班　　级	软件技术五班
学　　号	**201725230516**
姓　　名	李文仙
小组成员	李文仙
指导教师	毛晓娜

2018 年 **6** 月 **28** 日

目　录

1 引言.....3
1.1 研究背景意义.....3
1.2 游戏的简介.....3
2 设计功能分析.....3
2.1 游戏界面设计.....3
2.2 游戏整体结构.....4
3 XML 布局.....4
3.1 编写 XML.....4
4 初始化测绘.....5
4.1 自定义 view.....5
4.2 绘制棋盘.....7
5 初始化棋盘棋子.....8
5.1 初始化棋盘棋子.....8
5.2 绘制棋子.....9
6 落子.....10
7 绘制棋子.....12
8 逻辑判断.....13
8.1 游戏结束.....13
8.2 五子连珠的条件.....14
9 存储与恢复.....18
10 再来一局的设置.....18
10.1 调用 start 方法.....18
10.2 疑问.....18
总结.....24

1 引言

1.1 研究背景意义

随着经济的发展，社会竞争越来越激烈，现代社会进入了竞争时代。上班族为了完成公司业务，每天超负荷的工作；学生为了不落后他人每天早起晚睡不断地学习，压力巨大。所以为了缓解大家的压力，使大家在工作、学习之余娱乐一下，活跃大脑，提高工作效率，学习效率，因此益智游戏越来越受人们的关注，五子棋作为益智游戏之一，备受人们的喜爱。

1.2 游戏的简介

在古代，五子棋棋具虽然与围棋相类同，但是下法却是完全不同的。正如《辞海》中所言，五子棋是“棋类游戏，棋具与围棋相同，两人对局，轮流下子，先将五子连成一行者为胜”。

2 设计功能分析

2.1 游戏界面设计

随着经济的发展，社会竞争越来越激烈，现代社会进入了竞争时代。上班族为了完成公司业务，每天超负荷的工作；学生为了不落后他人每天早起晚睡不断地学习，压力巨大。所以为了缓解大家的压力，使大家在工作、学习之余娱乐一下，活跃大脑，提高工作效率，学习效率，因此益智游戏越来越受人们的关注，五子棋作为益智游戏之一，备受人们的喜爱。

2.2 游戏整体结构

随着经济的发展，社会竞争越来越激烈，现代社会进入了竞争时代。上班族为了完成公司业务，每天超负荷的工作；学生为了不落后他人每天早起晚睡不断地学习，压力巨大。所以为了缓解大家的压力，使大家在工作、学习之余娱乐一下，活跃大脑，提高工作效率，学习效率，因此益智游戏越来越受人们的关注，五子棋作为益智游戏之一，备受人们的喜爱。

3 XML 布局

3.1 编写 XML

编写 XML

1、设置背景 方便看 view 的大小 设置红色 半透明

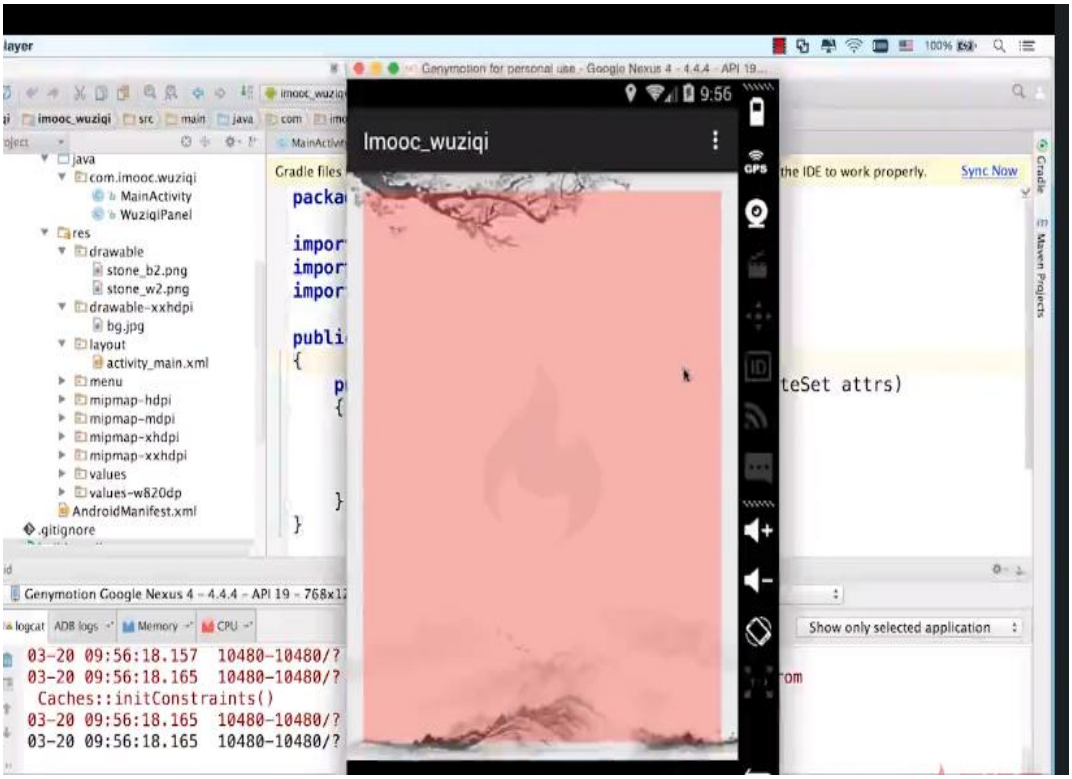
去掉 padding

测量五子棋

measure

2、自定义 view

background 具体的看到 view 大小

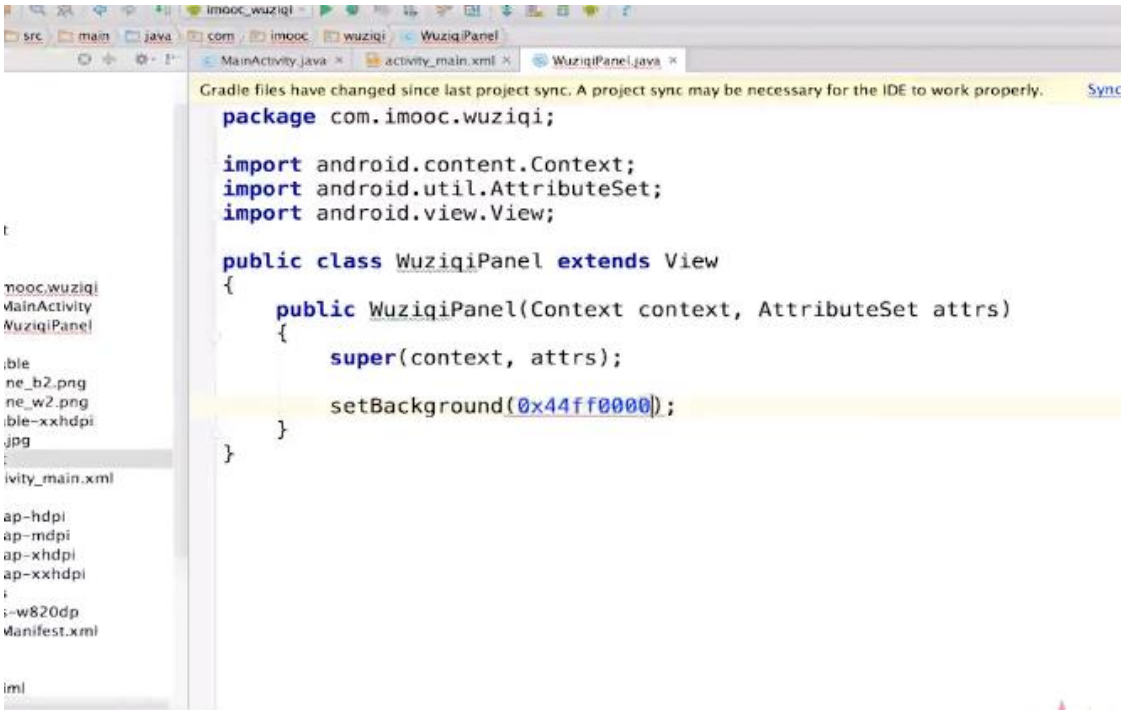


4 初始化测绘

4.1 自定义 view

因为是自定义 view 所以继承 view

然后实现他的方法



只实现两个构造方法

并在 activity-main 中进行使用

在自定义 view 中 首先考虑的就是测量 的使用情况

width 辅助类 MeasureSpec.getSize

widthSize 和 widthMode

设置宽高

正方形 Math.min(widthSize,heightSize)

使用确定的值

onMeasure 部分代码

```java
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec)
{
    int widthSize = MeasureSpec.getSize(widthMeasureSpec);
    int widthMode = MeasureSpec.getMode(widthMeasureSpec);

    int heightSize = MeasureSpec.getSize(heightMeasureSpec);
    int heightMode = MeasureSpec.getMode(heightMeasureSpec);


    int width = Math.min(widthSize, heightSize);

    if(widthMode == MeasureSpec.UNSPECIFIED)
    {
        width = heightSize;
    }else  if(heightMode == MeasureSpec.UNSPECIFIED)
    {
        width = widthSize;
    }
    setMeasuredDimension(width, width);
}
```

4.2 绘制棋盘

声明两个成员变量和一个静态常量

```java
private int mPanelWidth;
private float mLineHeight;
private int MAX_LINE=10;
```

绘制棋盘的方法 drawBroard

成员变量 棋盘宽度 mpanelWidth

高 lineheight

成员变量 用局部变量代替一下 然后 直接使用局部变量

for 循环

横坐标：

定义起点左边 （int）lineheihgt/2

终止坐标用的是 （int） w（宽度）-lineheight/2

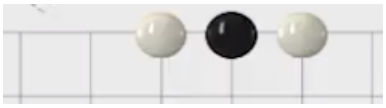
效果：左右两边各是放歌的一半即半个 lineheight

纵坐标：

每行的纵坐标一致

考虑横坐标左边可以下棋

所以第一行的纵坐标是 0.5 个 lineheight



则第二行是 1.5 个 lineheight
规律：首先至少有一个 lineheight
横线的绘制
(0.5+i)*lineheight

绘制竖线
起点 0.5 个 lineheight
第二行 1.5 个 lineheight
起点和终点同上 所以 差不多 只是命名不同
效果图

注意：
1、中间有 9 个 lineheight 上下有 0.5 个 lineheight 因为棋子可以下在边界上 （共十行）

5 初始化棋盘棋子

5.1 初始化棋盘棋子

onSizeChanged

布置一些跟尺寸相关的成员变量

*1.0f 装换成浮点数

绘制棋盘

ondrow

绘制的时候需要一个 paint

```
private Paint mPaint=new Paint();
```

构造方法中对 paint 进行初始化

创建一个 init 方法

设置 paint 颜色 （灰色半透明）0x88000000

画线

```
private void init() {
    mPaint.setColor(0x88000000);
    mPaint.setAntiAlias(true);
    mPaint.setDither(true);
    mPaint.setStyle(Paint.Style.STROKE);
```

5.2 绘制棋子

引入棋子的图片

```
private Bitmap mWhitePiece ;
private Bitmap mBlackPiece ;
```

在 init 中对棋子进行初始化

```
mWhitePiece = BitmapFactory.decodeResource(getResources() , R.drawable.stone_w2);
mBlackPiece = BitmapFactory.decodeResource(getResources() , R.drawable.stone_w2);
```

限定棋子的大小为行高的 3/4

```
private float ratioPieceOfLineHeight = 3 * 1.0f / 4;
```

改变棋子的大小

onSizeChange

Bitmap 方法 createscaledBitmap（加载的图片、宽度、）

声明一个目标的宽度

```
int pieceWidth = (int) (mLineHeight * ratioPieceOfLineHeight);
```

黑色棋子同上

6 落子

6.1 onTouch 处理

存储坐标事涉及到 x 和 y

声明一个集合 List<> 专门放用户所点击的一个坐标

对黑白棋子都需要这个范形

```
private List<Point> mWhiteArray = new ArrayList<>()
private List<Point> mBlackArray = new ArrayList<>()
```

声明一个变量 表明现在是黑子下棋还是白子下棋

```
//白棋先手，当前轮到白棋
private boolean mIsWhile=true;
```

在 onTouch 方法中调用并把 x,y 封装成 point

对黑白棋子进行判断

如果是白子放入 mWhiteArray

黑子放入 mBlackArray

```
if(mIsWhite)
{
    mWhiteArray.add(p);
}else
{
    mBlackArray.add(p);
}
```

调用 invalidate 然后请求重绘

作用：捕获了手势，用户点击以后选取坐标并进行重绘

每一个 point 对象对应一个位置

然后根据 x，y 去确定哪一个点

然后就在哪一个点绘制棋子

注意事项：

（1）点击落子的范围

点击时 x，y 会有一个细小的误差 eg：（0，0）和（0，0.1）

所以存储方式不必用点 改为(0,0),(1,0),(2,0)

原来

```
Point p = new Point(x, y);
```

改变方案

```
Point p = getValidPoint(x)
```

并创建一个 getValidPoint 方法 返回一个 point

return new point

```
private Point getValidPoint(int x, int y)
{
    return new Point((int)(x/mLineHeight),(int)(y/mLineHeight));


}
```

因此实现了（0，0）（0，1）（0，2）的坐标效果（偏离不大的话就是落在了同一个点上）

（2）棋子不能下在同一个位置

判断棋子是不是落在了同一个位置

onTouch 处理

防止在滚动时间落子 action_down 和 action_move 都不不合适

action_down 改为 action_up

```
int action = event.getAction();
if (action == MotionEvent.ACTION_DOWN)
{

    int x = (int) event.getX();
    int y = (int) event.getY();

    Point p = getValidPoint(x, y);
    if (mWhiteArray.contains(p) || mBlackArray.contains(p))
    {
        return false;
    }
```

7 绘制棋子

OnDraw 方法　绘制棋子 for 循环

```
{
    for(int i = 0 , n = mWhiteArray.size(); i < n   ; i++)
}
```

棋子之间相差 1/4lineheight

因为左右都有空隙 所以处以 2

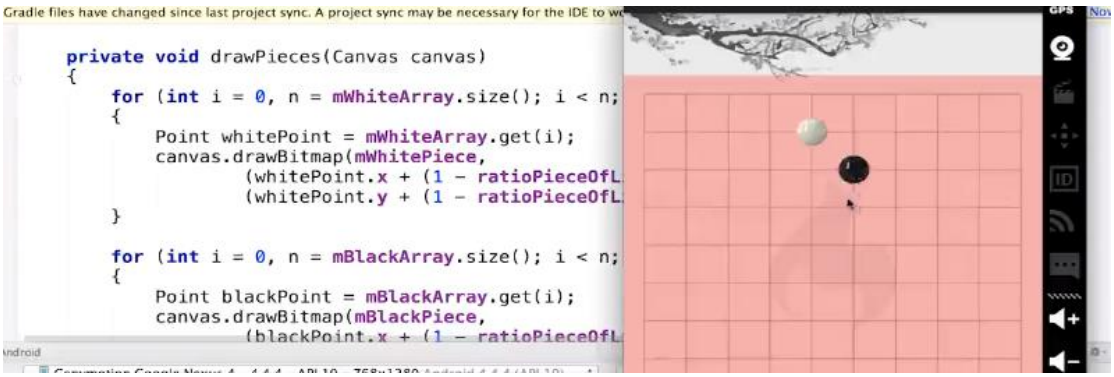
```
private void drawPieces(Canvas canvas)
{
    for(int i = 0 , n = mWhiteArray.size(); i < n   ; i++)
    {
        Point whitePoint = mWhiteArray.get(i);
        canvas.drawBitmap(mWhitePiece , (whitePoint.x +(1-ratioPieceOfLineHeight )/2)*mLineHeight);
    }
}
```

Cannot resolve method 'drawBitmap(android.graphics.Bitmap, float)

纵坐标同横坐标一致

黑子的绘制同白子

效果图：



8 逻辑判断

8.1 游戏结束

创建成员变量　游戏结束

假设白子胜利

True 白子赢 fouse 黑赢

```
private boolean mIsGameOver ;
private boolean mIsWhiteWinner ;
```

1、游戏结束后不能再落子

2、游戏结束的要求 横向纵向斜向连五个棋

```
private void checkGameOver()
{
    boolean whiteWin = checkFiveInLine(mWhiteArray);
    boolean blackWin = checkFiveInLine(mBlackArray);

    if(whiteWin || blackWin)
    {
        mIsGameOver = true;
    }

}

private boolean checkFiveInLine(List<Point> mWhiteArray)
{
    return false;
}
```

增加判断条件

编辑 Game over

```
private void checkGameOver()
{
    boolean whiteWin = checkFiveInLine(mWhiteArray);
    boolean blackWin = checkFiveInLine(mBlackArray);

    if(whiteWin || blackWin)
    {
        mIsGameOver = true;
        mIsWhiteWinner = whiteWin ;

        String text =mIsWhiteWinner?"白棋胜利":"黑棋胜利"
    }

}
```

8.2 五子连珠的条件

分别是横向纵向左斜和右斜；

判断 X，Y 位置的棋子，是否有相邻的五个一致；

横向原理：以当前棋子为中心往左右相邻的棋子数，每数到一个同色棋子加一，直至数到五个。

```
//判断 X，Y 位置的棋子，是否横向有相邻的五个棋子
    private boolean checkHorizontal(int x, int y, List<Point> points)
    {
        int count=1;
        //横向左
        for(int i=1;i<MAX_COUNT_IN_LINE;i++)
        {
            if (points.contains(new Point(x-i,y)))
            {
                count++;
            }else
            {
                break;
            }
        }
        if(count==MAX_COUNT_IN_LINE)return true;

        for(int i=1;i<MAX_COUNT_IN_LINE;i++)
        {
            if (points.contains(new Point(x+i,y)))
            {
                count++;
            }else
```

```
        {
            break;
        }
    }
    if(count==MAX_COUNT_IN_LINE)return true;
    return  false;
}



private boolean checkVertical(int x, int y, List<Point> points)
{
    int count=1;
    //上下
    for(int i=1;i<MAX_COUNT_IN_LINE;i++)
    {
        if (points.contains(new Point(x,y-i)))
        {
            count++;
        }else
        {
            break;
        }
    }
    if(count==MAX_COUNT_IN_LINE)return true;

    for(int i=1;i<MAX_COUNT_IN_LINE;i++)
    {
        if (points.contains(new Point(x,y+i)))
        {
            count++;
        }else
        {
            break;
        }
    }
    if(count==MAX_COUNT_IN_LINE)return true;
    return  false;
}
```

```
private boolean checkLeftDiagonal (int x, int y, List<Point> points)
{
    int count=1;
    //左斜
    for(int i=1;i<MAX_COUNT_IN_LINE;i++)
    {
        if (points.contains(new Point(x-i,y+i)))
        {
            count++;
        }else
        {
            break;
        }
    }
    if(count==MAX_COUNT_IN_LINE)return true;

    for(int i=1;i<MAX_COUNT_IN_LINE;i++)
    {
        if (points.contains(new Point(x+i,y-i)))
        {
            count++;
        }else
        {
            break;
        }
    }
    if(count==MAX_COUNT_IN_LINE)return true;
    return  false;
}


private boolean checkRightDiagonal(int x, int y, List<Point> points)
{
    int count=1;
    //右斜
    for(int i=1;i<MAX_COUNT_IN_LINE;i++)
    {
        if (points.contains(new Point(x-i,y-i)))
        {
            count++;
        }else
        {
            break;
        }
```

```
        }
        if(count==MAX_COUNT_IN_LINE)return true;

        for(int i=1;i<MAX_COUNT_IN_LINE;i++)
        {
            if (points.contains(new Point(x+i,y+i)))
            {
                count++;
            }else
            {
                break;
            }
        }
        if(count==MAX_COUNT_IN_LINE)return true;
        return  false;
    }




    private void drawPieces(Canvas canvas)
    {

        for (int i=0,n=mWhiteArray.size();i<n;i++)
        {
          Point whitePoint=mWhiteArray.get(i);
            canvas.drawBitmap(mWhitePiece,
                    (whitePoint.x+(1-ratioPieceOfLineHeight)/2)*mLineHeight,
                    (whitePoint.y+(1-ratioPieceOfLineHeight)/2)*mLineHeight,null);
        }
        for (int i=0,n=mBlackArray.size();i<n;i++)
        {
            Point blackPoint=mBlackArray.get(i);
            canvas.drawBitmap(mBlackPiece,
                    (blackPoint.x+(1-ratioPieceOfLineHeight)/2)*mLineHeight,
                    (blackPoint.y+(1-ratioPieceOfLineHeight)/2)*mLineHeight,null);
        }
    }
```

9 存储与恢复

存储黑白棋子和 Game over

```
private static final String INSTANCE = "instance";
private static final String INSTANCE_GAME_OVER ="instance_game_over";
private static final String INSTANCE_WHITE_ARRAY ="instance_game_over";
private static final String INSTANCE_BLACK_ARRAY ="instance_game_over";
```

在 onSaveInstanceState 方法里面

```
@Override
protected Parcelable onSaveInstanceState()
{
    Bundle bundle = new Bundle();
    bundle.putParcelable(INSTANCE,super.onSaveInstanceState());
    bundle.putBoolean(INSTANCE_GAME_OVER,mIsGameOver);
    bundle.putParcelableArrayList(INSTANCE_WHITE_ARRAY,mWhiteArray);
```

10 再来一局的设置

10.1 调用 start 方法

逻辑原理：调用 start 方法重新启动游戏

```
public void start()
{
    mWhiteArray.clear();
    mBlackArray.clear();
    mIsGameOver=false;
    ismIsWhiteWinner=false;
    invalidate();
}
```

10.2 疑问

问题 1、

图片的安放问题

xxhtpl

和 xxhtpi 干扰

问题 2、什么是自定义 view

zidingyiview 就是凡事能被用户看到的小控件都是一种 view，也可以自定义 view

测量 View 的高宽的基本步骤：

1、 从 MeasureSpec 对象中提取拒听的测量模式与大小

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int wMode = MeasureSpec.getMode(widthMeasureSpec);//获取测量模式
    int hMode = MeasureSpec.getMode(heightMeasureSpec);
    int wSize = MeasureSpec.getSize(widthMeasureSpec);//获取测量的大小
    int hSize = MeasureSpec.getSize(heightMeasureSpec);
}
```

2、 通过判断测量的模式，给出不同的测量值。

```
int mHegith;
int mWidht;

@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {

    int wMode = MeasureSpec.getMode(widthMeasureSpec);//获取测量模式
    int hMode = MeasureSpec.getMode(heightMeasureSpec);
    int wSize = MeasureSpec.getSize(widthMeasureSpec);//获取测量的大小
    int hSize = MeasureSpec.getSize(heightMeasureSpec);
    if (wMode == MeasureSpec.EXACTLY) {
        mWidht = wSize;
    } else {
        mWidht = 400;
        mWidht = Math.min(mWidht, wSize);
    }
    if (hMode == MeasureSpec.EXACTLY) {
        mHegith = hSize;
    } else {
        mHegith = 400;
        mHegith = Math.min(mHegith, hSize);
    }
    setMeasuredDimension(mHegith, mHegith);
}
```


ViewGroup 测量

ViewGroup 在管理其子 View 时，其中一个管理项目就是负责子 View 的显示大小。当 ViewGroup 的大小为 wrap_content 时，ViewGroup 就需要对子 View 进行遍历,以便获取所有子 View 的大小，从而决定自己的大小。而在其他模式下则会通过具体指定的值来设置自身的大小。

ViewGroup 在测量时通过遍历所有的子 View，从而调用子 View 的 Measure 方法来获取每一个 View 的测量结果。当子 View 测量完毕之后,就需要将子View放到合适的位置,这个过程就是View的Layout过程。ViewGroup 在执行 Layout 过程时，同样是使用遍历来调用子 View 的方法，并指定其具体的显示位置，从而来决定其布局位置。在自定义 ViewGroup 时，通常会去重写 onLayout()方法来控制其子 View 显示位置的逻辑 。同样，如果需要支持 wrap_content 属性，则必须重写 onMeasure 方法。

ViewGroup 通常情况下不需要绘制，因为它本身就没有需要绘制的东西，如果不是指定了 ViewGroup 的背景颜色，那么 ViewGroup 的 onDraw()方法都不会调用。当时，ViewGroup 会调用 dispatchDraw()方法来绘制其子 View，其过程同样是通过遍历所有的子 View，并调用子 View 的绘制方法来完成绘制工作。

在自定义 View 时，我们通常会去重写 onDraw()方法来绘制 View 的显示内容。如果该 View 使用 wrap_content 属性，那么还必须重写 onMeasure()方法。在自定义 View 时比较几个重要的回调方法：

onFinishInflate():从 Xml 加载组件后回调。

onSizeChaged():组件大小改变的时候回调

onMeasure():回调该方法进行测量（ViewGrop 测量时会调用子 View 的 Measure 子 View 的测量）

onLayout():回调该方法来确定显示位置。

onTouchEvent():监听到触摸事件回调。

通常情况下，有以下三种方法来实现自定义的控件。

1、 对现有控件进行拓展。

2、 通过组合来实现新控件。

3、 重写 View 来实现全新的控件

问题三 3、onMeasure 知识点

Android 系统调用 onMeasure 来定义 view 的大小，很长时间理解不是很透彻，今天花了些时间打日志来理解它。总结如下。

1. widthMeasureSpec 和 heightMeasureSpec 这两个值是 android:layout_width="200dp" android:layout_height="80dp"来定义的，它由两部分构成，可通过 int specModeHeight = MeasureSpec.getMode(heightMeasureSpec); int specSizeHeight = MeasureSpec.getSize(heightMeasureSpec)来得到各自的值。

如果 android:layout_width="wrap_content"或 android:layout_width="fill_parent"，哪么得到的 specMode 为 MeasureSpec.AT_MOST，如果为精确的值则为 MeasureSpec.EXACTLY。另外，specSize 要想得到合适的值需要在 Androidmanifest.xml 中

添加<uses-sdk android:minSdkVersion="10" />

2.系统默认的 onMeasure 调用方法是 getDefaultSize 来实现，有时候在自定义控件的时候多数采用

```
int widthSize = MeasureSpec.getSize(widthMeasureSpec);
int heightSize = MeasureSpec.getSize(heightMeasureSpec);
setMeasuredDimension(widthSize, heightSize);
```

getDefaultSize 代码如下：

```
public static int getDefaultSize(int size, int measureSpec) {
int result = size;
int specMode = MeasureSpec.getMode(measureSpec);
int specSize = MeasureSpec.getSize(measureSpec);

switch (specMode) {
case MeasureSpec.UNSPECIFIED:
result = size;
break;
case MeasureSpec.AT_MOST:
case MeasureSpec.EXACTLY:
result = specSize;
break;
}
return result;
}
```

3.onMeasure 好像会调用两次,这点我没有到代码中具体跟踪了，如果该 view 的父 view 是 RelativeLayout，则其父 view 的 onMeasur 也要测量一下。我在测试中用的 xml 为:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.goso.ui.record.RecordView
    android:id="@+id/uvMeter"
    android:layout_width="200dp"
    android:layout_height="80dp"
    android:layout_centerInParent="true" />
    </RelativeLayout>
    <!-- RelativeLayout LinearLayout-->

    protected void onMeasure(int widthMeasureSpec, int
heightMeasureSpec) {
    // TODO Auto-generated method stub
    int specModeHeight = MeasureSpec.getMode(heightMeasureSpec);
    int specSizeHeight = MeasureSpec.getSize(heightMeasureSpec);
    Log.e("HJJ", "****specModeHeight:" + getModeStr(specModeHeight) + ",
specSizeHeight:" + specSizeHeight );

    int widthSize = MeasureSpec.getSize(widthMeasureSpec);
    int heightSize = MeasureSpec.getSize(heightMeasureSpec);
    setMeasuredDimension(widthSize, heightSize);
    }
```

打印出的结果为：

```
01-06 12:47:53.170: ERROR/HJJ(5639): ****specModeHeight:AT_MOST,
specSizeHeight:714
01-06 12:47:53.170: ERROR/HJJ(5639): ****specModeHeight:EXACTLY,
specSizeHeight:120
01-06 12:47:53.190: ERROR/HJJ(5639): ****specModeHeight:AT_MOST,
specSizeHeight:714
01-06 12:47:53.190: ERROR/HJJ(5639): ****specModeHeight:EXACTLY,
specSizeHeight:120
```

如果将 RelativeLayout 换成 LinearLayout 则为

01-06 12:51:17.980: ERROR/HJJ(5779): ****specModeHeight:EXACTLY, specSizeHeight:120

01-06 12:51:18.010: ERROR/HJJ(5779): ****specModeHeight:EXACTLY, specSizeHeight:120

问题 4、canvas.drawLine(startX)..... 什么意思

问题 5、不会进入 point.java

百度搜索 查找资料 请教学长

结果是按住 ctrl 点击 point

总结

通过这个项目的实战，说实话收获还真的是不小。首先是专业的知识的巩固和学习，真的是感觉学了这么长时间，都没有做这一边记得扎实。五子棋这个项目虽然不是很困难，但是对于我们只是学了一个学期来说，这个程序并没有那么的简单，而且好多的知识点我们也没有接触到，看似简单，实际上，自己打出来并调试成功，并没有那么的容易。在逻辑思维上，更是考虑到了方方面面，比如说棋子之间的间隙 1/4 个 lineheight 和棋盘的横纵坐标的起点和终点坐标的选择，以及下棋时顶端多出的 1/2 个 lineheight 距离，都需要考虑到并绘制完整。棋子的绘制是调用了 ondraw 方法和 for 循环对棋子进行重绘，每下一个棋子对棋子进行重新绘制，在这之前，对棋子也是有要求的,棋子的大小的限定和棋盘方格的大小相匹配。落子的时候，黑白棋子不能落在同一个位置，以及滑动屏幕时不能落子，不能说是滑动一下屏幕就落下了棋子。这就对 onTouch 指尖检测有了要求。以

及下棋的位置（x，y）的限定，在某个范围内，棋子落子规定的交叉点上，不能让棋子偏离太大。程序虽然没有调用数据库，但是也考虑到了，程序的存储问题，避免在接电话或者不小心关闭程序时，清空原来运行的程序。最后设计的一个再来一局也是启动调用 start 方法，清空棋盘，起到再来一局的效果。这些看似简单的东西，让我们自己想思路、设计和运作方案的话，我觉得还是比较困难的。

其次通过这次实战我还发现真的是有好多的东西等待着我去学习，总以为自己已经会了，明白了，到头来，才发现懂的只是一些皮毛。经过这次的实战，我也是深深的体会到了程序员的辛苦，总有人仰望他们的薪水，可又有谁知道，他们背后的艰辛和努力。在这个信息化、人工智能迅速发展的时代，我觉得学习新知识是我们最大的任务和使命，因为科技发展的速度，永远超乎我们的想象，落后就会挨打这句亘古不变的真理，随着时间的流逝也是证实了它的意义。

所以说，学习才是王道。只有不断地学习，才能紧跟时代的潮流。不说了，学习去。