目录

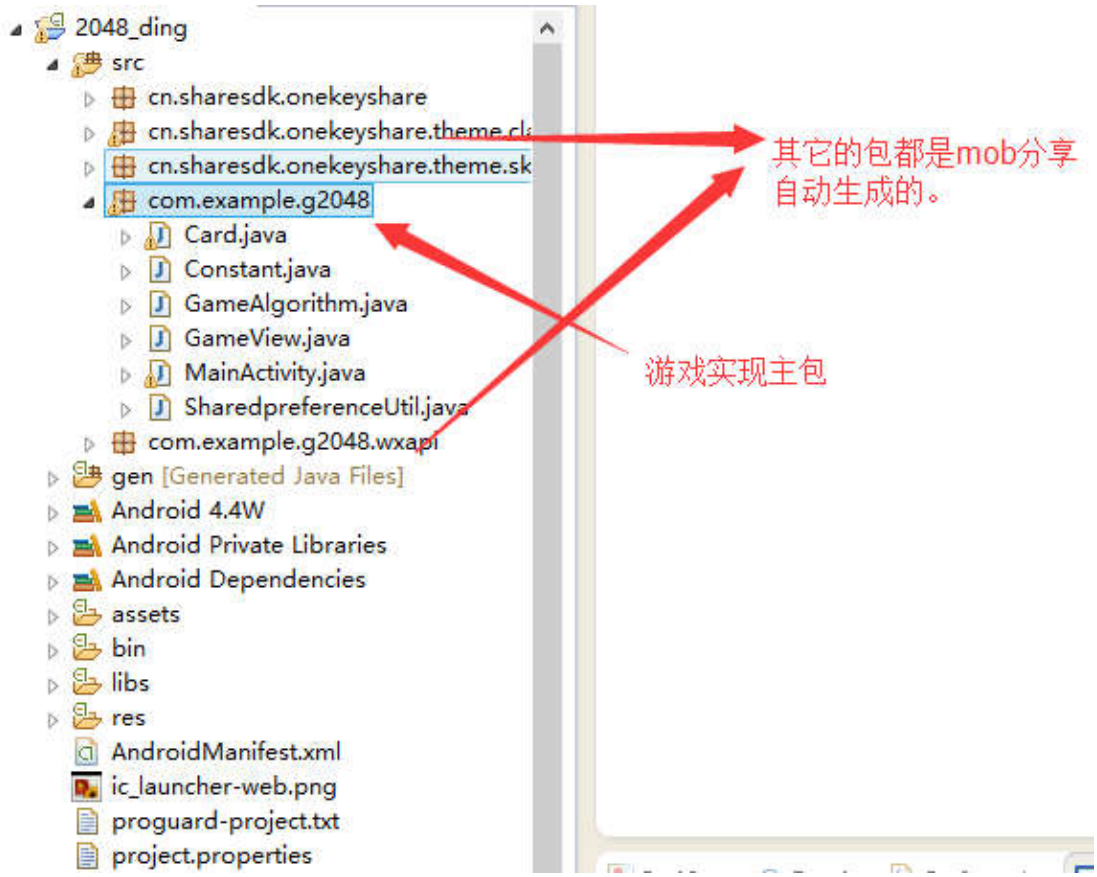
1 项目简介

主要是对 android-2048 游戏的实际编写，在熟悉了相关的游戏规则后，实际编写一个可以操作的 2048 游戏，并加入了分享功能（未完全实现），熟悉 android 编程。

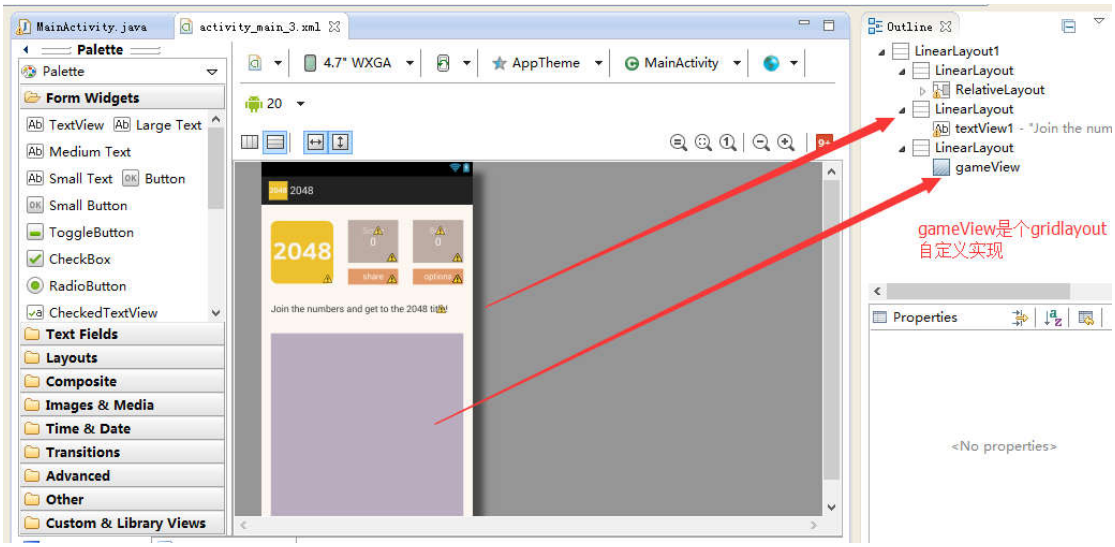
我的 2048 源码下载地址：https://github.com/doctording/android_2048

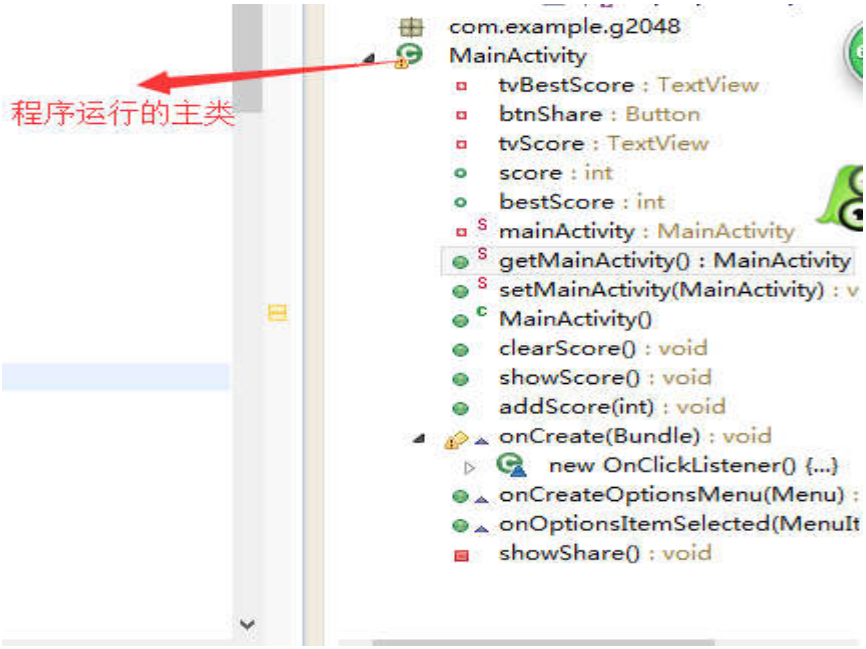
2 项目结构说明

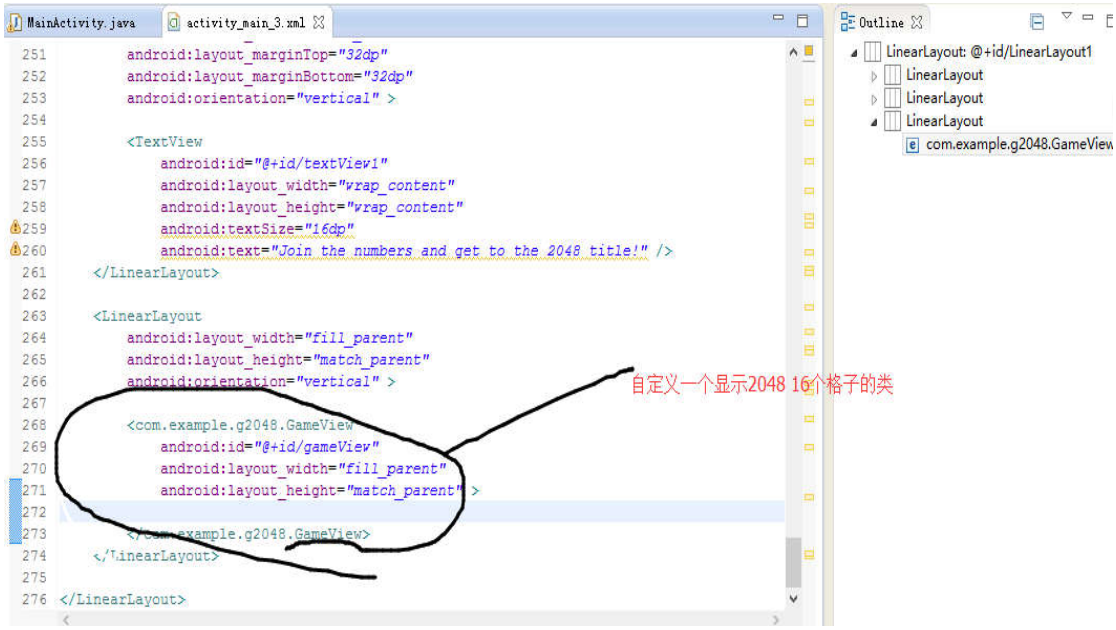
2.1 项目整体结构

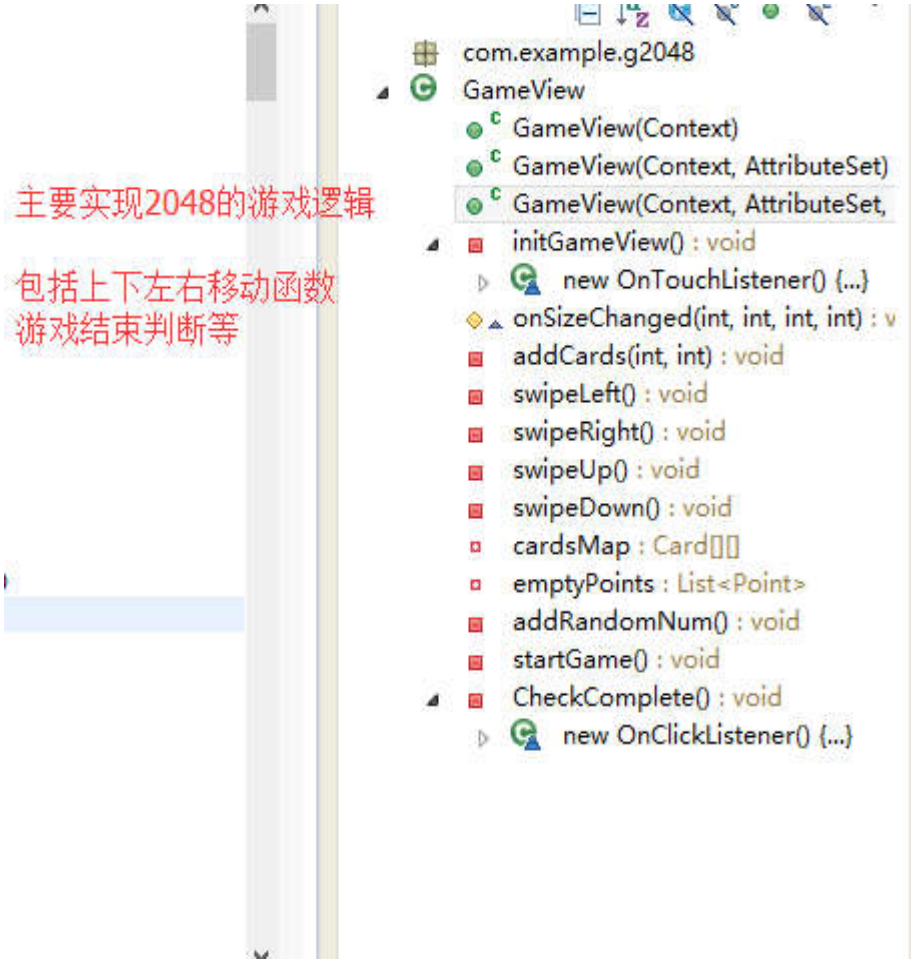


游戏主界面：


MainActivity.java
activity_main_3.xml
Palette
Form Widgets
TextView
Large Text
Medium Text
Small Text
Button
Small Button
ToggleButton
CheckBox
RadioButton
CheckedTextView
Text Fields
Layouts
Composite
Images & Media
Time & Date
Transitions
Advanced
Other
Custom & Library Views
4.7" WXGA
AppTheme
MainActivity
20
2048
Join the numbers and get to the 2048 tile!
Outline
LinearLayout1
LinearLayout
RelativeLayout
LinearLayout
textView1 - "Join the num
LinearLayout
gameView
gameView是个gridlayout
自定义实现
Properties
<No properties>


程序运行的主类
com.example.g2048
MainActivity
tvBestScore : TextView
btnShare : Button
tvScore : TextView
score : int
bestScore : int
mainActivity : MainActivity
getMainActivity() : MainActivity
setMainActivity(MainActivity) : v
MainActivity()
clearScore() : void
showScore() : void
addScore(int) : void
onCreate(Bundle) : void
new OnClickListener() {...}
onCreateOptionsMenu(Menu) :
onOptionsItemSelected(MenuIt
showShare() : void


MainActivity.java
activity_main_3.xml
Outline
LinearLayout: @+id/LinearLayout1
LinearLayout
LinearLayout
LinearLayout
com.example.g2048.GameView
自定义一个显示2048 16个格子的类


主要实现2048的游戏逻辑
包括上下左右移动函数
游戏结束判断等
com.example.g2048
GameView
GameView(Context)
GameView(Context, AttributeSet)
GameView(Context, AttributeSet,
initGameView() : void
new OnTouchListener() {...}
onSizeChanged(int, int, int, int) : v
addCards(int, int) : void
swipeLeft() : void
swipeRight() : void
swipeUp() : void
swipeDown() : void
cardsMap : Card[][]
emptyPoints : List<Point>
addRandomNum() : void
startGame() : void
CheckComplete() : void
new OnClickListener() {...}

2.2 分享功能说明：

借助了 mob.com 平台，参看我写的博客：
http://blog.csdn.net/qq_26437925/article/details/49704317

2.3 项目中各个 JAVA 类的说明

2.3.1 Card.java

Card 类是每一个数字卡片类，包括 num 表示卡片上的数字，label 就是卡片 ，可以设置颜色，数字样式等。

```
/**
 * 判断两个卡片是否可以合并（即数字相等）
 * @param c
 * @return
 */
public boolean equals(Card c)
{
    return (getNum() == c.getNum());
}
```

2.3.2 Constant.java

Constant 类 主要设置相应的常量，方便程序的扩展。我就定义了卡片数字颜色。

```
public class Constant {

/**
 * 2 - 2048 数字的颜色
 */
public static String[] numColor ={
    "#CCC0B3", // 0
    "#EEE4DA", // 2
```

```
        "#EDE0C8", // 4
        "#F2B179", // 8
        "#F49563", // 16
        "#F5794D", // 32
        "#F55D37" , //64
        "#EEE863" , //128
        "#EDB04D" , // 256
        "#ECB04D" , // 512
        "#EB9437" , // 1024
        "#EA7821" // 2048
    };

}
```

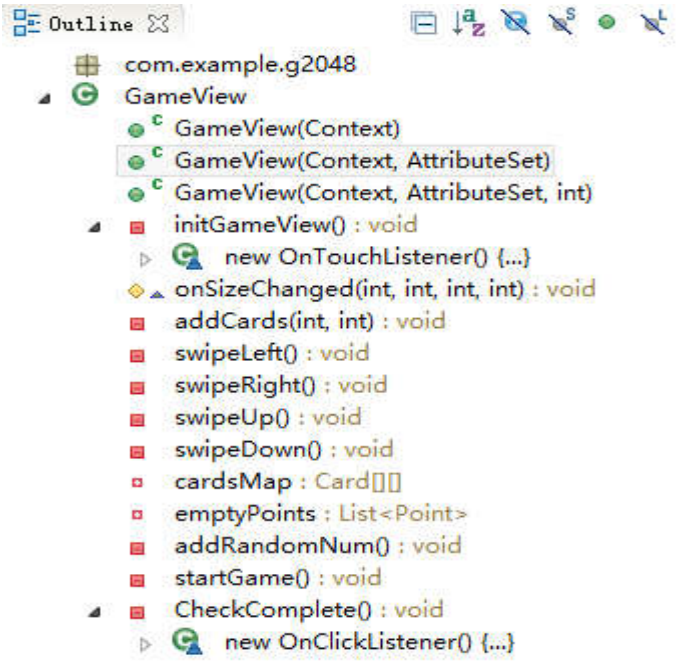
2.3.3 SharedpreferenceUtil.Java

利用sharedpreference存储数据到本地，主要是存储2048游戏的最高分。
sharedpreference存储一些简单的数据比较的方便,参考如下：
http://blog.csdn.net/qq_26437925/article/details/46399973

2.3.4 GameView.java

```
public class GameView extends GridLayout
```

GameView 继承了GridLayout ， 包括对游戏进行的操作。



在MainActivity中定义了自身变量，以便GameView中可以拿到相应的变量进行游戏操作

```
    private static MainActivity mainActivity = null ;

    public static MainActivity getMainActivity() {
        return mainActivity;
    }

    public static void setMainActivity(MainActivity mainActivity) {
        MainActivity.mainActivity = mainActivity;
    }
```

再初始化GameView，定义了手势操作

```
    private void initGameView()
    {
        setColumnCount(4);
        // 主体背景色 // // #bda6ae // 0xffbbadc0
        setBackgroundColor(0xffbbadc0);

        setOnTouchListener(new View.OnTouchListener() {

            private float startx , offsetx ;
            private float starty , offsety ;

            @Override
            public boolean onTouch(View v, MotionEvent event) {
```

```
                switch(event.getAction())
                {
                case MotionEvent.ACTION_DOWN:
                    startx = event.getX();
                    starty = event.getY();
                    break;
                case MotionEvent.ACTION_UP:
                    offsetx = event.getX() - startx;
                    offsety = event.getY() - starty;

                    if(Math.abs(offsetx) > Math.abs(offsety))
                    {
                        if(offsetx < -5)
                        {
                            //  left
                            swipeLeft();
                        }else if(offsetx > 5){
                            // right
                            swipeRight();
                        }
                    }else{
                        if(offsety < -5)
                        {
                            // up ;
                            swipeUp();
                        }else if(offsety > 5){
                            // down
                            swipeDown();
                        }
                    }
                    break;
                }
                return true;
            }
        });


    }
```

关于手势，主要是利用GridView自身的onSizeChanged方法，和OnTouchListener，进行上下左右的判断，在从而进行相应的操作

```
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    super.onSizeChanged(w, h, oldw, oldh);

    int cardWidth = (Math.min(w, h) - 10) / 4 ;
    addCards(cardWidth,cardWidth);

    startGame();  // 开始game
}

setOnTouchListener(new View.OnTouchListener() {

        private float startx , offsetx ;
        private float starty , offsety ;

        @Override
        public boolean onTouch(View v, MotionEvent event) {

            switch(event.getAction())
            {
            case MotionEvent.ACTION_DOWN:
                startx = event.getX();
                starty = event.getY();
                break;
            case MotionEvent.ACTION_UP:
                offsetx = event.getX() - startx;
                offsety = event.getY() - starty;

                if(Math.abs(offsetx) > Math.abs(offsety))
                {
                    if(offsetx < -5)
                    {
                        //  left
                        swipeLeft();
                    }else if(offsetx > 5){
                        // right
                        swipeRight();
                    }
                }else{
                    if(offsety < -5)
                    {
                        // up ;
```

```
                    swipeUp();
                }else if(offsety > 5){
                    // down
                    swipeDown();
                }
            }
            break;
        }
        return true;
    }
});
```

2.4 以左移为例说明 2048 游戏逻辑

```
private void swipeLeft()
{
    boolean merge = false ;  // 是否要产生新的
    for(int y = 0 ; y < 4 ; y++)
    {
        for(int x = 0 ; x < 4 ; x++)
        {

            for(int x1 = x + 1; x1 < 4; x1++)
            {
                if(cardsMap[x1][y].getNum() > 0){
                    if(cardsMap[x][y].getNum() <= 0)
                    {

cardsMap[x][y].setNum(cardsMap[x1][y].getNum());
                        cardsMap[x1][y].setNum(0);
                        x-- ;
                        merge = true ;
                    }else
if(cardsMap[x][y].equals(cardsMap[x1][y])){

cardsMap[x][y].setNum(cardsMap[x][y].getNum()*2);
                        cardsMap[x1][y].setNum(0);


MainActivity.getMainActivity().addScore(cardsMap[x][y].getNum());
                        merge = true ;
                    }
                    break ;
```

```
                }
            }

        }
    }
    if(merge)
    {
        addRandomNum();
        CheckComplete();
    }
}
```

在游戏运行时，左滑手机界面
根据2048逻辑，判断是否有合并的卡片。合并之后，要产生新的卡片，还要判断游戏是否结束。
产生新的卡片就是随机在空的卡片产生数字2和4的卡片。

```
private void addRandomNum()
{
    emptyPoints.clear();
    for(int y = 0 ; y < 4 ; y++)
    {
        for(int x = 0 ; x < 4 ; x++)
        {
            if(cardsMap[x][y].getNum() <= 0)
            {
                emptyPoints.add(new Point(x,y));
            }
        }
    }

    Point p = emptyPoints.remove(
            (int)(Math.random()*emptyPoints.size()));
    cardsMap[p.x][p.y].setNum(Math.random()>0.1?2:4);
}
```

再添加卡片

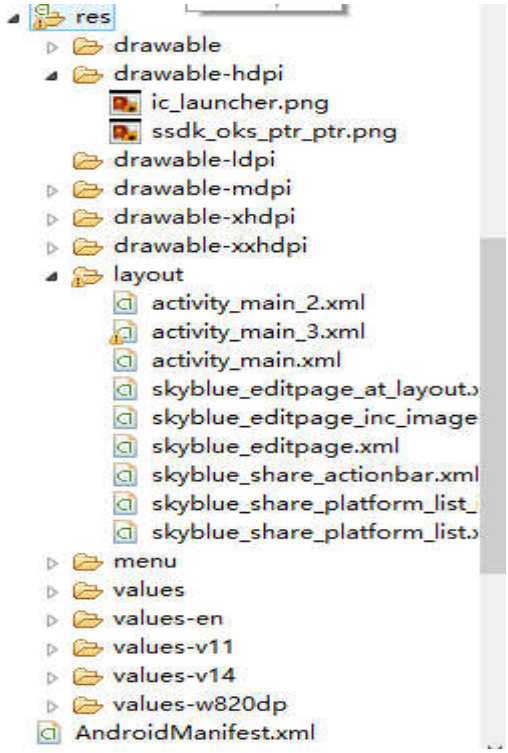
```
private void addCards(int cardWidth,int cardHeight)
{
    Card c ;
    for(int y = 0 ; y < 4 ; y++)
    {
        for(int x = 0 ; x < 4 ; x++)
        {
```

```
                c = new Card(getContext());
                c.setNum(0);
                addView(c, cardWidth, cardHeight);

                cardsMap[x][y] = c ;
            }
        }
    }
```

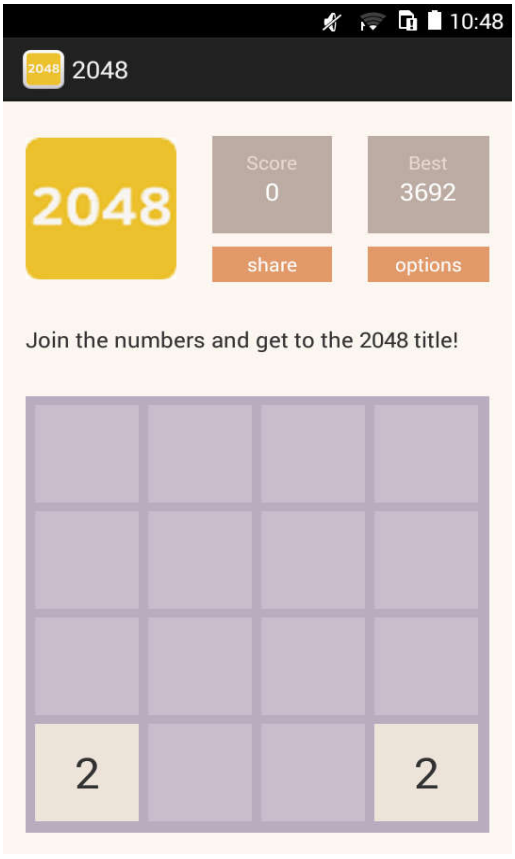
3 游戏资源

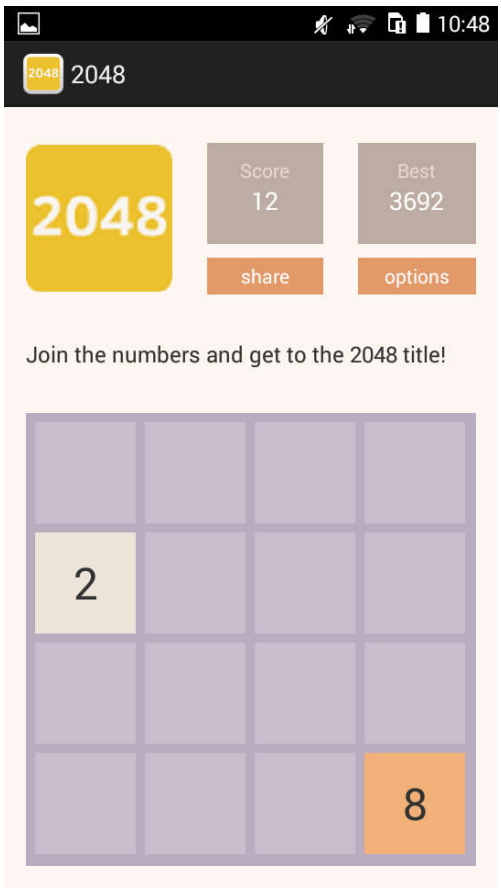
主要是网上参考，页面设计部分需要不断优化（自己设计的不是太好）；关于游戏颜色，使用了拾色器等工具进行相关的操作。

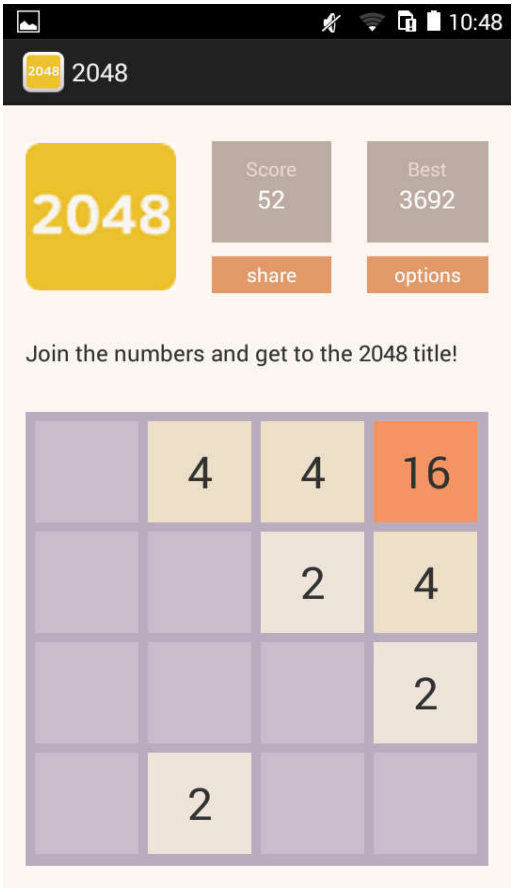


4 游戏运行截图：


10:36
授权管理
360超级ROO
ShareSDK
Root Explore
2048
2048

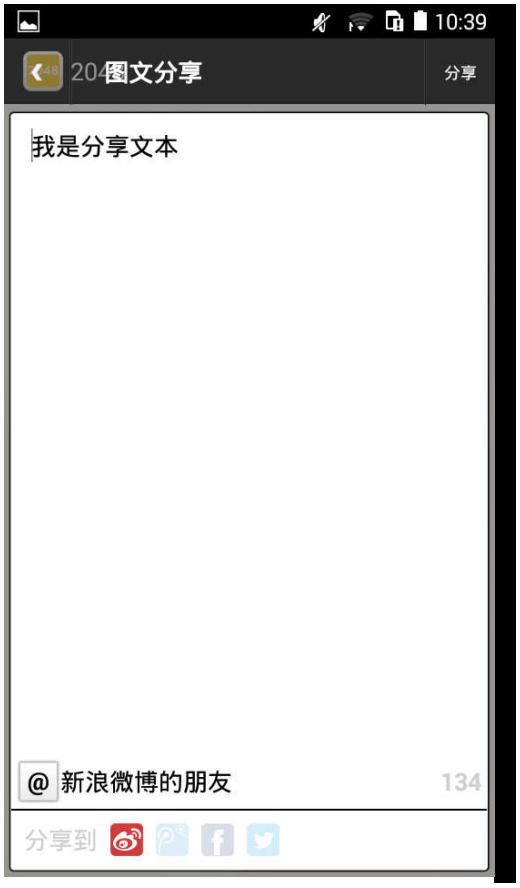

10:48
2048
2048
Score
0
Best
3692
share
options
Join the numbers and get to the 2048 title!
2
2


10:48
2048
2048
Score
12
Best
3692
share
options
Join the numbers and get to the 2048 title!
2
8


10:48
2048
2048
Score
52
Best
3692
share
options
Join the numbers and get to the 2048 title!
4
4
16
2
4
2
2


10:49
2048
2048
Score
52
Best
3692
share
options
新浪微博
腾讯微博
QQ空间
微信好友
微信朋友圈
微信收藏
QQ
Facebook
Twitter
取消


10:49
选择联系人
QQ好友
群
讨论组
QQ空间
最近聊天
我的电脑
分享
我是分享文本
评论一番吧
取消
发送


10:39
204图文分享
分享
我是分享文本
@
新浪微博的朋友
134
分享到