第 16 章 Android 游戏开发实践——快乐数独

益智类游戏是一种比较流行的游戏，其画面大都比较简单，很少有很复杂的游戏特效，但是通常用到人工智能的算法来控制游戏的难度。而算法的优化是开发该类游戏的难点。这类游戏主要包括棋牌类游戏和智力测试类游戏，例如麻将、扫雷、五子棋、扑克牌等。

数独就是益智游戏的一种，玩法简单但数字的排列方式千变万化，很多人认为数独是训练头脑的绝佳方式。本章通过讲解数独游戏在 Android 平台上的设计与实现，使读者了解此类游戏的开发过程，掌握实用的开发技巧，学会此类游戏的开发。

16.1 游戏的背景及功能概述

本节在整体上对数独游戏进行了简单的介绍，使读者了解数独游戏的发展，知道什么是数独游戏，以及在 Android 中该游戏的玩法。

16.1.1 背景概述

数独的前身为“九宫格”，最早起源于中国。但当时的算法比现在的更为复杂，要求纵向、横向、斜向上的三个数字之和等于 15，而不只是数字不能重复。儒家典籍《易经》中的“九宫图”也是来源于此。

到了 18 世纪末，瑞士数学家莱昂哈德·欧拉又发明了一种叫做“拉丁方块”的游戏，之后不久，美国的一家数学逻辑游戏杂志开始刊登这类游戏，使此类游戏得到良好发展，之后又在日本得到了广泛的传播。2004 年，第一个“数独”游戏被刊登上了英国《泰晤士报》的封面，此时开始数独游戏才真正为世界所知晓。

由于此类游戏操作简单，不需要特定的语言基础，也不需要进行数字运算且可玩性高、锻炼思维、开发大脑，所以很快风靡全球。之后由其衍生的游戏也越来越多，例如杀手数独、角线数独等。

16.1.2　功能简介

数独游戏的规则很简单，只需在空格处填入 1～9 的数字，并保证每个数字在每个九宫格内只能出现一次，且每个数字在每一行、每一列也只能出现一次，而一般的游戏过程是系统随机生成一个棋局，然后玩家需要在空白处填上相应的数字使其满足游戏规则。该游戏的运行步骤如下。

① 启动游戏后首先进入的便是欢迎界面，效果如图 16-1 所示，在欢迎界面中，“点击屏幕继续……”会时有时无，达到提示的效果。

② 在游戏界面点击屏幕，便进入菜单界面，如图 16-2 所示。

③ 在菜单界面单击“关于游戏”菜单可进入“关于”界面，如图 16-3 所示，在“关于”界面中介绍了该游戏的目标平台及开发日期。

④ 在菜单界面单击“帮助游戏”菜单可进入“帮助”界面，如图 16-4 所示，在“帮助”界面介绍了游戏的基本规则。

图 16-1　欢迎界面

图 16-2　菜单界面

图 16-3　关于界面

图 16-4　帮助界面

⑤ 在菜单界面单击“开始游戏”菜单可进入游戏界面，如图 16-5 所示。

⑥ 当在游戏过程中单击“暂停”按钮时，便进入暂停状态，如图 16-6 所示。

⑦ 在游戏过程中随时可以通过“换题”按钮来更换题目。

⑧ 当玩家单击“放弃”按钮时，会提示玩家是否真的需要退出游戏，然后根据玩家的选择进行操作，如图 16-7 所示。

⑨ 当玩家单击“提示”按钮时，界面上会出现一个红心表示当前可以提示玩家输入数字，此时再单击空白处，便会自动填上正确答案，如图 16-8 所示。

图 16-5　游戏效果

图 16-6　暂停效果

图 16-7　退出提示

图 16-8　提示效果

⑩ 当玩家将界面中所有的空白全部填满数字时，系统会自动判断所填写的数字正确与否，当有错误时，会出现如图 16-9 所示的游戏失败界面；当全部填写正确时，会出现如图 16-10 所示的游戏胜利界面。

图 16-9　游戏失败

图 16-10　游戏胜利

⑪ 在胜利或者失败状态时，玩家可以通过点击屏幕任意位置返回欢迎界面。

⑫ 在菜单界面单击“退出游戏”便可退出该游戏。

> 提示：因为考虑到市面上很多 Google Phone 没有键盘，而屏幕普遍较大，所以在本游戏中采用全触控笔操作，以提高玩家对游戏的体验。

16.2　游戏的策划及准备工作

本节主要介绍数独游戏的策划及游戏实现前的准备工作。在商业化游戏的开发中，这些步骤还需要更细致、更具体、更全面。

16.2.1　游戏的策划

人类如果经常动用大脑，便可让自身的逻辑和抽象思维能力得到增长，因此，玩数独游戏就是一个增长智慧的好办法。通过数独题谜的解答，可以让头脑变得更聪明。用户可以把他作为锻炼逻辑思维及策略的工具，在手机上的实现便可以时时刻刻地开发思维、锻炼头脑。

数独的推理性强，一些数学的思想、推理、假设、反证等都会用到，每个题谜各不相同，不可能用一种方法解决所有问题，这也是数独的魅力所在。

开发这个游戏的目的是为读者在 Android 平台上进行游戏开发提供一个指导方案，而不是生产商业化的游戏产品。读者可以以此为范例开发出更好、更具可玩性的数独游戏。

16.2.2　Android 平台下游戏的准备工作

游戏的准备工作通常是根据游戏的策划来制作游戏所用的的图片、声音等。因为本游戏并没有添加音效设置，所以只需搜集或制作游戏过程中所用到的图片资源即可，本游戏用到的图片文件资源如表 16-1 所示。

表 16-1 图片清单

图 片 名	大小 (KB)	像素 (w×h)	用　途	图 片 名	大小 (KB)	像素 (w×h)	用　途
a1 ~a9.png	258	21×21	输入的数字	key_background.png	22	100×100	数字键盘背景图片
b1 ~b9.png	269	21×21	默认的数字	change2.png	4.47	50×30	被按下的换题
icon.png	6.6	48×48	游戏图标	change1.png	4.65	50×30	未被按下的换题
fail.png	4.42	150×80	游戏失败	select.png	1.03	24×24	选中的单元格
win.png	1.85	150×80	恭喜过关	drop1.png	4.41	50×30	未被按下的放弃
go_on.png	3.91	150×80	暂停中	drop2.png	4.41	50×30	被按下的放弃
time0.png	1.12	21×21	时间中的 0	help1.png	4.38	50×30	未被按下的提示
help2.png	4.38	50×30	被按下的提示	stop1.png	7.07	50×30	未被按下的暂停
Stop2.png	7.07	50×30	被按下的暂停	background.png	195	320×480	背景
heart.png	4.29	25×25	提示的心型	small_background.png	2.25	100×100	背景方框图片
time.png	1.26	31×31	时间的冒号	about.png	292	320×463	关于背景
exit.png	2.77	200×86	退出提示	help.png	284	320×463	帮助背景
menu.png	15.7	200×176	菜单图元	w1~30.gif	15.7	320×244	欢迎动画帧

res 是存放所有非代码资源的文件夹，其下的 drawable 文件夹一般存放图片资源。因为从 1.6r2 开始 SDK 支持各种尺寸的屏幕，所以在 res 下有多个以 drawable 开头的文件夹，mdpi 为标准图库。本程序中所有的图片资源都存储在 res\drawable-mdpi 文件夹下。

> 提示：当在程序中需要对图片进行不等比例的拉伸时，可使用 android sdk 安装目录下 tools 文件夹中的 draw9patch 工具对图片进行处理，以达到更好的拉伸效果。

16.3　游戏的架构

在正式开发代码之前，首先需要对该游戏的设计框架进行简要介绍，以帮助读者更好地理解游戏的开发过程。希望读者能够仔细阅读本节的内容，在整体上了解本游戏。这会为之后的开发带来事半功倍的功效。

16.3.1　各类的简要介绍

为了让读者更好地理解后面的代码，下面将对游戏中的各个类逐一进行简要说明。关于这些类的详细代码将在后面的章节中相继给出。

1．共有类

主类 SudukuActivity 类

该类是通过继承和扩展基类 Activity 来实现的，是整个应用程序的入口，主要是根据收到的 Handler 消息的不同切换到不同的界面。

2．欢迎界面相关类

（1）WelcomeView 类

该类为欢迎界面的实现类，主要负责欢迎动画界面的绘制，是欢迎界面的前台显示的

View，根据后台数据的不同绘制不同效果的动画。

（2）WelcomeViewDrawThread 类

该类为欢迎动画界面的刷帧线程。

（3）WelcomeViewGoThread 类

该类为欢迎动画界面的服务线程，主要负责欢迎动画的生成，通过改变表示当前帧的索引值达到动画的效果。

（4）关于界面 AboutView 类

该类是游戏关于界面的实现类，主要负责游戏关于界面的绘制。

（5）帮助界面 HelpView 类

该类是游戏帮助界面的实现类，主要负责游戏帮助界面的绘制。

3．游戏界面相关类

（1）GameView 类

GameView 类是游戏中最主要的一个类，游戏规则、游戏模型都包含在此类当中。同时该类还负责绘制游戏的画面、接收玩家的响应。

（2）GameViewDrawThread 类

GameViewDrawThread 类负责定时的刷帧操作。

（3）刷新时间线程 TimeThread 类

该线程的实现非常简单，只是定时改变游戏模型中的时间值，并不负责其他事件的处理。

（4）数字键盘渲染线程 DrawKeyThread 类

该线程的作用是数字键盘出现或消失过程中对数字键盘的渐变处理，逻辑很简单，定时改变数字键盘图片透明度即可。

（5）数独生成器 ShuDuSuanFa 类

该类是个普通的 Java 类，不继承任何类，只为游戏随机提供存放数独数字的二维数组，该类所采用算法的优劣决定了该游戏可玩性的好坏。

16.3.2　游戏的框架简介

本节的目的是让读者对 Android 平台的该游戏基本架构有一个简单了解，可能介绍得不够细致，不过请读者不要担心，本节只是对游戏整体的简单介绍，后面的各节中将详细讲解各个类的实现。简单的框架图如图 16-11 所示。

接下来将通过游戏的运行过程介绍各个类在游戏中的作用，具体运行步骤如下。

① 启动游戏后，首先在 KLSDActivity 中初始化 WelcomeView，然后将用户界面切换到欢迎动画界面 WelcomeView。

② 而在 WelcomeView 中会启动 WelcomeViewDrawThread 与 WelcomeViewGoThread 来刷新界面和生成欢迎动画。

③ 当玩家单击“帮助”菜单时，会创建 HelpView，显示“帮助”界面。

④ 当玩家单击“关于”菜单时，会创建 AboutView 并显示“关于”界面。

⑤ 当单击“开始游戏”时，则正式进入游戏界面 GameVew，同时启动 GameViewDrawThread

来刷新界面。

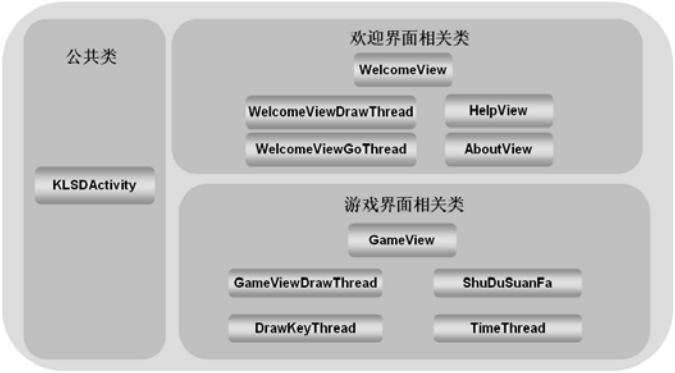

图 16-11　游戏框架图

⑥ 当玩家点击空白位置，需要绘制数字键盘时，会启动 DrawKeyThread 线程绘制数字键盘。

⑦ 在游戏过程中会根据情况启动 TimeThread 线程，来更新界面中的时间。

⑧ 其中 ShuDuSuanFa 为数独的算法生成，游戏前需要通过该类生成数独数组，游戏过程中同样需要使用该数组来判断玩家的输入是否正确。

> 提示：如果读者想优化数独的生成算法，只需更改 SudokuGenerator 类内部结构，并不需要对其他类进行更改。

16.4　欢迎界面的设计与实现

从本节开始将正式进入该游戏的开发过程。笔者始终坚信，手比眼高，举起手臂一定会高过眼睛，动手是学习程序最快的方式，所以希望读者能够跟随笔者的思路亲自动手实现这个游戏，有条件的读者可以将其移植到 Google Phone 中调试运行。

16.4.1　主类 KLSDActivity 实现

应用程序的每个屏幕的显示都通过继承和扩展基类 Activity 类来实现。重写 Activity 类的 onCreate()方法，在 onCreate()方法中做游戏的初始化工作，该类的代码如下。

```
1   package wyf.ytl;                                        //声明所在类
2   import android.app.Activity;                            //引入相关类
3   ……//该处省略了部分类的引入代码，读者可自行查阅随书光盘中的源代码
4   import android.view.WindowManager;                      //引入相关类
5   public class KLSDActivity extends Activity {
6       WelcomeView welcomeView;                            //“欢迎”动画界面的引用
7       GameView gameView;                                  //游戏界面的引用
8       AboutView aboutView;                                //“关于”界面
9       HelpView helpView;                                  //“帮助”界面
10      Handler myHandler = new Handler(){                  //用来更新 UI 线程中的控件
11        public void handleMessage(Message msg) {
```

```
12              if(msg.what == 1){                      //“欢迎”界面发送的消息
13                  if(welcomeView != null){
14                      welcomeView = null;
15                  }
16                  initGameView();                     //初始化游戏界面
17                  KLSDActivity.this.setContentView(gameView);
18              }else if(msg.what == 2){                //游戏胜利或者失败时点击屏幕
19                  if(gameView != null){
20                      gameView = null;
21                  }
22                  initWelcomeView();                  //初始化“欢迎”界面
23                  KLSDActivity.this.setContentView(welcomeView);
                                                        //切换到“欢迎”界面
24              }else if(msg.what == 3){
25                  initAboutView();                    //初始化“关于”界面
26                  KLSDActivity.this.setContentView(aboutView);//切换到“关于”界面
27              }else if(msg.what == 4){
28                  initHelpView();                     //初始化“帮助”界面
29                  KLSDActivity.this.setContentView(helpView); //切换到“帮助”界面
30              }
31          }
32       };
33      public void onCreate(Bundle savedInstanceState) {     //重写的 onCreate 方法
34          super.onCreate(savedInstanceState);
35          //设置全屏
36          requestWindowFeature(Window.FEATURE_NO_TITLE);  //去掉标题栏
37          getWindow().setFlags(
38                      WindowManager.LayoutParams.FLAG_FULLSCREEN ,
39                     WindowManager.LayoutParams.FLAG_FULLSCREEN);
40          initWelcomeView();                              //初始化“欢迎”界面
41          this.setContentView(welcomeView);               //切换到“欢迎”界面
42      }
43      public void initWelcomeView(){                      //初始化“欢迎”界面
44          welcomeView = new WelcomeView(this);
45      }
46      public void initGameView(){                         //初始化游戏界面
47          gameView = new GameView(this);
48      }
49      public void initAboutView(){                        //初始化“关于”界面
50          aboutView = new AboutView(this);
51      }
52      public void initHelpView(){                         //初始化“帮助”界面
53          helpView = new HelpView(this);
54      }
55  }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 KLSDActivity.java。

- 第 6~9 行声明为各个界面的引用。
- 第 10~32 行创建一个 Handler 类，根据接收到的信息的不同初始化不同的用户界面并切换到相应用户界面。
- 第 33~42 行为重写的 onCreate 方法，其中第 35~39 行将屏幕设置为全屏模式，而第 40~41 行初始化并切换到欢迎界面。
- 第 43~54 行为各个 View 的初始化方法，在各个方法中，对相应的界面进行初始化，然后在其他位置可以通过使用 setContentView 方法将用户界面设置成期望的界面。

16.4.2　欢迎界面 WelcomeView 类的实现

WelcomeView 类是本游戏的欢迎界面，主要负责欢迎界面的绘制。该类中最重要的方法为 onDraw 方法。该类的开发步骤如下。

① 框架的搭建，首先需要为该类搭建基本的框架，代码如下。

```
1   package wyf.ytl;                                          //声明所在包
2   import android.graphics.Bitmap;                           //引入相关类
3   ……//该处省略了部分类的引入代码，读者可自习查阅随书光盘的源代码
4   import android.view.SurfaceView;                          //引入相关类
5   public class WelcomeView extends SurfaceView implements SurfaceHolder.Callback{
6       KLSDActivity activity;                                //activity 的引用
7       WelcomeViewDrawThread welcomeViewDrawThread;          //刷帧线程
8       WelcomeViewGoThread welcomeViewGoThread;              //动画生成线程
9       int[] bitmapsID = new int[]{                          //图片 ID
10          R.drawable.w1,R.drawable.w2,
11          ……//该处省略了部分图片资源的 ID，读者可自行查阅随书光盘中的源代码
12          R.drawable.w29,R.drawable.w30,
13      };
14      Bitmap[] bitmaps;                                     //图片数组
15      Bitmap[] menuBitmap = new Bitmap[4];                  //菜单
16      int drawIndex = 0;                                    //当前绘制第几帧
17      boolean drawString = false;                           //是否绘制文字
18      int status = 0;                                       //0 欢迎状态,1 菜单状态
19      Paint paint;                                          //画笔
20      Paint paint2;
21      public WelcomeView(KLSDActivity activity) {           //构造器
22          super(activity);
23          this.activity = activity;                         //得到 activity 的引用
24          getHolder().addCallback(this);                    //添加 Callback 接口的实现
25          initBitmap();                                     //调用初始化方法
26          welcomeViewDrawThread = new WelcomeViewDrawThread(this);
27          welcomeViewGoThread = new WelcomeViewGoThread(this);
28      }
29      public void initBitmap(){                             //初始化图片方法
30          ……//该处省略了图片资源的初始化代码，将在之后给出
31      }
32      protected void onDraw(Canvas canvas) {                //绘制方法
33          ……//该处省略了界面的绘制代码，将在之后给出
34      }
35      public boolean onTouchEvent(MotionEvent event) {      //屏幕监听方法
36          ……//该处省略了屏幕监听方法的代码，将在之后给出
37      }
38      public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3){}
39      public void surfaceCreated(SurfaceHolder arg0) {
40          welcomeViewDrawThread.setFlag(true);              //设置循环标志位
41          welcomeViewGoThread.setFlag(true);
42          welcomeViewDrawThread.start();                    //启动线程
43          welcomeViewGoThread.start();                      //启动线程
44      }
45      public void surfaceDestroyed(SurfaceHolder arg0) {
46        boolean retry = true;                               //循环标志位
47        welcomeViewDrawThread.setFlag(false);               //设置循环标志位
```

```
48          welcomeViewGoThread.setFlag(false);
49          while (retry) {                                  //循环
50                 try {
51                      welcomeViewDrawThread.join();       //等待线程结束
52                      welcomeViewGoThread.join();         //等待线程结束
53                      retry = false;
54                 }
55                 catch (Exception e) {                    //不断地循环，直到刷帧线程结束
56                      e.printStackTrace();                //打印异常信息
57                 }
58            }
59       }
60   }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 WelcomeView.java。

- 第 9~12 行创建了存放动画帧图片 ID 的数组，将其存放到数组中方便管理，在之后进行系统维护时，如需新增动画帧，只需在此数组中添加新的 ID 即可。
- 第 16 行的变量记录了当前已经绘制到图片的第几帧，在之后进行绘制时，会根据该变量的值进行图片的读取。而第 18 行为界面的状态值，根据状态的不同会有不同的处理。
- 第 21~28 行为该类的构造器，先得到 activity 的引用，然后添加 Callback 接口的实现，同时初始化图片资源及相关的线程资源。
- 第 32~34 行为重写的绘制方法，代码会在之后给出，而第 35~37 行为重写的屏幕监听方法，负责根据玩家点击屏幕的事件的不同执行不同的操作，代码会在后面进行介绍。
- 第 38~57 行实现了接口中的三个抽象方法，其中第 39~44 行的 surfaceCreated 方法会在此 View 创建时被调用，在方法中启动绘制线程及动画生成线程。而其中 45~57 行的 surfaceDestroyed 方法则会在此 View 被释放时被调用，在方法中等待绘制线程及动画线程的结束。

② 接下来对图片资源的初始化方法进行介绍，代码如下所示。

```
1   public void initBitmap(){                                  //初始化图片方法
2       paint = new Paint();                                   //画笔的初始化
3       paint2 = new Paint();
4       paint2.setAlpha(125);                                  //设置透明度
5       paint.setColor(Color.WHITE);
6       paint.setAntiAlias(true);                              //打开抗锯齿
7       paint.setTextSize(18);                                 //设置字体大小
8       bitmaps = new Bitmap[bitmapsID.length];                //初始化图片数组
9       for(int i=0; i<bitmaps.length; i++){                   //循环初始化图片
10          bitmaps[i] = BitmapFactory.decodeResource(getResources(), bitmapsID[i]);
11      }
12      Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.menu);
13      for(int i=0; i<menuBitmap.length; i++){                //切成小图片
14          menuBitmap[i] = Bitmap.createBitmap(bitmap, 0,
15                  (bitmap.getHeight()/menuBitmap.length)*i,
16                  bitmap.getWidth(),
17                  (bitmap.getHeight()/menuBitmap.length));
18      }
19      bitmap = null;                                         //释放掉大图即释放掉菜单图元
20  }
```

- 第 2~3 行初始化两个画笔，一个用于正常图片的绘制，另一个用于绘制透明度不同的图片。
- 第 5~7 行对画笔进行设置，包括颜色、抗锯齿及字体大小。
- 第 8~11 行初始化欢迎动画的各个图片帧资源。
- 第 12~19 行先初始化菜单图元，然后按照一定比例将其分割成各个单个的菜单，并将其存储到图片数组中，最后释放菜单图元。

③ 绘制方法 onDraw 的完善，代码如下所示。

```
1   protected void onDraw(Canvas canvas) {                    //绘制方法
2       super.onDraw(canvas);
3       canvas.drawColor(Color.BLACK);                        //清背景
4       canvas.drawBitmap(bitmaps[drawIndex], 0, 100, paint);
5       if(status == 0){                                      //欢迎动画播放时
6           if(drawString){                                   //允许绘制文字时
7               canvas.drawText("点击屏幕继续...", 103, 380, paint);   //绘制文字
8           }
9       }
10      else if(status == 1){                                 //菜单状态时
11          canvas.drawRect(0, 0, 320, 480, paint2);          //绘制半透明矩形
12          for(int i=0; i<menuBitmap.length; i++){           //循环
13              canvas.drawBitmap(menuBitmap[i], 60, 130+50*i, null);
14          }
15      }
16  }
```

- 第 3 行将屏幕背景设成黑色。
- 第 5~9 行当状态值为 0 即欢迎动画播放时，根据标记位的不同选择是否绘制文字。
- 第 10~14 行表示当前状态值为 1 即菜单状态时，先绘制一个全屏幕的半透明矩形，然后再循环绘制菜单图片。

④ 接下来对屏幕监听方法 onTouchEvent 进行完善，代码如下所示。

```
1   public boolean onTouchEvent(MotionEvent event) {              //屏幕监听方法
2       if(event.getAction() == MotionEvent.ACTION_DOWN){         //屏幕被按下事件
3           double x = event.getX();                              //得到 X 坐标
4           double y = event.getY();                              //得到 Y 坐标
5           if(status == 0){                                      //当 status 为 0 时
6               status = 1;                                       //设置 status 为 1
7           }
8           else if(status == 1){
9               if(x>60 && x<260 && y>130 && y<160){               //开始游戏
10                  activity.myHandler.sendEmptyMessage(1);       //发送 Handler 消息
11              }else if(x>60 && x<260 && y>180 && y<210){         //关于游戏
12                  activity.myHandler.sendEmptyMessage(3);       //发送 Handler 消息
13              }else if(x>60 && x<260 && y>230 && y<260){         //帮助游戏
14                  activity.myHandler.sendEmptyMessage(4);       //发送 Handler 消息
15              }else if(x>60 && x<260 && y>280 && y<310){         //退出游戏
16                  System.exit(0);                               //直接退出游戏
17              }
18          }
19      }
20      return super.onTouchEvent(event);
21  }
```

- 第 3 行先得到事件的 *X*、*Y* 坐标。
- 第 5 行表示当欢迎动画正在播放时，玩家点击了屏幕，此时只需将当前状态设置成 1 即可，表示进入菜单状态。
- 第 8~17 行表示在菜单界面，根据玩家单击的按钮的不同执行不同的操作，一般的操作是向 Activity 发送 Handler 消息。
- 第 20 行调用父类的 onTouchEvent 方法并将返回值返回。

16.4.3 刷帧线程 WelcomeViewDrawThread 类的实现

本节将对“欢迎”动画界面的服务线程 WelcomeViewDrawThread 进行介绍，该类主要负责界面的定时刷新，代码如下所示。

```
1   package wyf.ytl;                                          //声明所在包
2   import android.graphics.Canvas;                           //引入相关类
3   import android.view.SurfaceHolder;                        //引入相关类
4   public class WelcomeViewDrawThread extends Thread{
5       WelcomeView welcomeView;                              //welcomeView 的引用
6       private int sleepSpan = 100;
7       private boolean flag = true;                          //循环标志位
8       private SurfaceHolder surfaceHolder;                  //surfaceHolder 的引用
9       public WelcomeViewDrawThread(WelcomeView welcomeView){   //构造器
10          this.welcomeView = welcomeView;                   //得到 WelcomeView 的引用
11          surfaceHolder = welcomeView.getHolder();          //得到 SurfaceHolder 的引用
12      }
13      public void run() {                                   //重写的 run 方法
14          Canvas c;                                         //声明画布
15          while (this.flag) {
16              c = null;                                     //将画布置空
17              try {
18                  //锁定整个画布
19                  c = this.surfaceHolder.lockCanvas(null);
20                  synchronized (this.surfaceHolder) {     //同步
21                      welcomeView.onDraw(c);                //调用绘制方法
22                  }
23              } finally {                                   //用finally保证一定被执行
24                  if (c != null) {
25                      //更新屏幕显示内容
26                  this.surfaceHolder.unlockCanvasAndPost(c);
27                  }
28              }
29              try{
30                  Thread.sleep(sleepSpan);                  //睡眠指定毫秒数
31              }catch(Exception e){                          //捕获异常
32                  e.printStackTrace();                      //打印异常信息
33              }
34          }
35      }
36      public void setFlag(boolean flag){                    //循环标记位的 set 方法
37          this.flag = flag;
38      }
39  }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 WelcomeViewDrawThread.java。

- 第 9~12 行为该类的构造器，在构造器中得到 WelcomeView 的引用及 SurfaceHolder 的引用。
- 第 19 行得到并锁定画布，然后在第 21 行的同步语句块中调用 WelcomeView 的 onDraw 方法进行界面的绘制。
- 第 26 行释放并更新屏幕显示的内容。
- 第 36~38 行为循环标志位 flag 的 set 方法，用于其他类对该变量的设置。

16.4.4　动画生成线程 WelcomeViewGoThread 类的实现

前面已经介绍了“欢迎”界面的绘制工作，本节将对“欢迎”动画界面的动画生成线程进行介绍，使我们的游戏的欢迎动画动起来，代码如下所示。

```
1   package wyf.ytl;                                          //声明所在包
2   public class WelcomeViewGoThread extends Thread{
3       WelcomeView welcomeView;                              //welcomeView 的引用
4       private int sleepSpan = 150;
5       private boolean flag = true;                          //循环标记位
6       public WelcomeViewGoThread(WelcomeView welcomeView) {//构造器
7           this.welcomeView = welcomeView;                   //得到 welcomeView 的引用
8       }
9       public void run() {                                   //重写的 run 方法法
10          while (flag) {                                    //循环
11              welcomeView.drawIndex++;                      //自加
12              if(welcomeView.drawIndex>welcomeView.bitmapsID.length-1){
13                  welcomeView.drawIndex = welcomeView.bitmapsID.length-10;
14              }
15              if(welcomeView.drawIndex%5 == 0){             //当对 5 取余为 0 时
16                  welcomeView.drawString = !welcomeView.drawString;
17              }
18              try{
19                  Thread.sleep(sleepSpan);                  //睡眠指定毫秒数
20              }catch(Exception e){                          //捕获异常
21                  e.printStackTrace();                      //打印异常信息
22              }
23          }
24      }
25      public void setFlag(boolean flag){                    //循环标志位的设置方法
26          this.flag = flag;
27      }
28  }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 WelcomeViewGoThread.java。

- 第 6~8 行为该类的构造器，在构造器中只需得到 WelcomeView 的引用即可。
- 第 9~24 行为该类的 run 方法，每次循环将欢迎界面的 drawIndex 加 1，然后判断是否到达数组的最后，当到达最后时，将其减 10，然后继续进行循环，第 15~17 行表示每循环 5 次改变一次 drawString 值。
- 第 25~27 行为循环标志位 flag 的 set 方法。

16.5 “帮助”与“关于”界面的设计与实现

上一节已经将欢迎动画相关类介绍完毕，本节将介绍另外两个辅助界面，“帮助”界面及“关于”界面。两个界面分别为玩家在菜单界面点击“帮助”或者“关于”菜单时进入的界面。

16.5.1 “帮助”界面 HelpView 类的实现

本节将对该游戏的“帮助”界面进行介绍。该界面的实现比较简单，只需将图片绘制到指定位置，然后绘制一个“确定”按钮即可，代码如下所示。

```
1   package wyf.ytl;                                          //声明所在包
2   import android.graphics.Bitmap;                           //引入相关类
3   ……//该处省略了部分类的引入代码，读者可自行查阅随书光盘中的源代码
4   import android.view.SurfaceView;                          //引入相关类
5   public class HelpView extends SurfaceView implements SurfaceHolder.Callback{
6       KLSDActivity activity;                                //activity 的引用
7       Bitmap helpBitmap ;                                   //背景图片的引用
8       SurfaceHolder surfaceHolder;                          //surfaceHolder 的引用
9       public HelpView(KLSDActivity activity) {              //构造器
10          super(activity);
11          this.activity = activity;                         //得到 activity 的引用
12          surfaceHolder = this.getHolder();                 //获得 surfaceHolder
13          getHolder().addCallback(this);                    //添加 Callback 接口的实现
14          helpBitmap = BitmapFactory.decodeResource(getResources(), R.drawable. help);
15      }
16      public void onDraw(Canvas canvas) {                   //绘制方法
17          canvas.drawColor(Color.WHITE);                    //背景色
18          canvas.drawBitmap(helpBitmap, 0, 10, null); //绘制图片
19      }
20      public boolean onTouchEvent(MotionEvent event) {          //键盘监听方法
21          if(event.getAction() == MotionEvent.ACTION_DOWN){     //屏幕被按下
22              double x = event.getX();
23              double y = event.getY();                          //得到坐标
24              if(x>220 && x<310 && y>430 && y<460){             //单击“确定”按钮
25                  activity.myHandler.sendEmptyMessage(2);       //发送 Handler 消息
26              }
27          }
28          return super.onTouchEvent(event);
29      }
30      public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3){}
31      public void surfaceCreated(SurfaceHolder arg0) {          //View 创建时被调用
32          Canvas c = null;                                      //声明画布
33          try {
34              //锁定整个画布
35              c = this.surfaceHolder.lockCanvas(null);
36              synchronized (this.surfaceHolder) {     //同步
37                  onDraw(c);                          //调用绘制方法
38              }
39          } finally {                                 //用 finally 保证一定被执行
40              if (c != null) {
41                  //更新屏幕显示内容
42                  this.surfaceHolder.unlockCanvasAndPost(c);
43              }
```

```
44            }
45        }
46        public void surfaceDestroyed(SurfaceHolder arg0) {}  //接口中方法的空实现
47    }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 HelpView.java。

- 第 9~15 行为“帮助”界面的构造器，在构造器中先得到 activity 以及 surfaceHolder 的引用，然后添加 Callback 接口的实现并对图片资源进行初始化。
- 第 16~19 行为重写的绘制方法，在该方法中，首先绘制白色背景，然后将帮助界面的背景图片绘制到指定位置。
- 第 20~29 行为重写的键盘监听方法，当玩家点击屏幕时，会调用该方法来处理玩家的屏幕事件，在“帮助”界面中该方法实现比较简单，只是判断玩家点击的是否为“确定”按钮即可。当玩家点击“确定”按钮时，需要向 activity 发送 Handler 消息通知控制器切换屏幕。
- 第 31~45 行实现了接口中的 surfaceCreated 抽象方法，该方法会在 HelpView 创建时被调用，在该方法中首先得到并锁定整个画布，然后调用 onDraw 方法绘制屏幕，最后释放并更新屏幕的显示内容。

16.5.2 “关于”界面 AboutView 的实现

上一节已经对帮助界面进行了介绍，本节将介绍游戏“关于”界面的实现过程，代码如下所示。

```
1   package wyf.ytl;                                          //声明所在包
2   import android.graphics.Bitmap;                           //进入相关类
3   ……//该处省略了部分类的引入代码，读者可自行查阅随书光盘中的源代码
4   import android.view.SurfaceView;                          //引入相关类
5   public class AboutView extends SurfaceView implements SurfaceHolder.Callback{
6       KLSDActivity activity;                                //activity 的引用
7       Bitmap aboutBitmap ;                                  //背景图片
8       SurfaceHolder surfaceHolder;                          //surfaceHolder 的引用
9       public AboutView(KLSDActivity activity) {             //构造器
10          super(activity);
11          this.activity = activity;                         //得到 activity 的引用
12          surfaceHolder = this.getHolder();
13          getHolder().addCallback(this);                    //添加 Callback 接口的实现
14          aboutBitmap = BitmapFactory.decodeResource(getResources(),
R.drawable.about);
15      }
16      public void onDraw(Canvas canvas) {                   //绘制方法
17          canvas.drawColor(Color.WHITE);
18          canvas.drawBitmap(aboutBitmap, 0, 10, null);          //绘制图片
19      }
20      public boolean onTouchEvent(MotionEvent event) {          //屏幕监听方法
21          if(event.getAction() == MotionEvent.ACTION_DOWN){     //屏幕被按下事件
22              double x = event.getX();
23              double y = event.getY();                          //得到坐标
24              if(x>230 && x<310 && y>310 && y<450){             //单击“确定”按钮
25                  activity.myHandler.sendEmptyMessage(2);       //发送消息
26              }
```

```
27            }
28            return super.onTouchEvent(event);
29        }
30        public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3){}
31        public void surfaceCreated(SurfaceHolder arg0) {
32            Canvas c = null;                                        //声明画布
33            try {
34                //锁定整个画布
35                c = this.surfaceHolder.lockCanvas(null);
36                synchronized (this.surfaceHolder) {                 //同步
37                    onDraw(c);                                      //调用绘制方法
38                }
39            } finally {                                       //用 finally 保证一定被执行
40                if (c != null) {
41                    //更新屏幕显示内容
42                    this.surfaceHolder.unlockCanvasAndPost(c);
43                }
44            }
45        }
46        public void surfaceDestroyed(SurfaceHolder arg0) {} //接口中方法的空实现
47    }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 AboutView.java。

- 第 9~15 行为该类的构造器，在构造器中得到 activity 及 surfaceHolder 的引用，然后对图片资源进行初始化。
- 第 16~19 行为重写的 onDraw 绘制方法，先绘制背景色，然后绘制背景图片。
- 第 20~29 行为关于界面的屏幕监听方法，同样只对“确定”按钮进行监听，当“确定”按钮被单击时，会向 Activity 发送 Handler 消息。
- 第 30~46 行实现了 SurfaceHolder.Callback 接口中的三个方法，其实现代码与帮助界面中相应方法的实现完全相同，因本书篇幅有限，在此不再赘述，读者可参见 16.5.1 节代码中相应方法的讲解自行学习。

此时运行该游戏，并在菜单界面单击“帮助”与“关于”菜单，将分别进入“帮助”与“关于”界面，可观察到如图 16-12 和图 16-13 所示的运行效果。

图 16-12 “帮助”界面

图 16-13 “关于”界面

16.6　游戏界面的框架搭建

前面已经对与“欢迎”界面相关的类进行了介绍，本节开始将进入游戏界面的开发环节。

① 首先对游戏界面 GameView 的方法框架进行介绍，代码如下所示。

```
1   package wyf.ytl;                                          //声明所在包
2   import android.graphics.Bitmap;                           //引入相关类
3   ……//该处省略了部分类的引入代码，读者可自行查阅随书光盘中的源代码
4   import android.view.SurfaceView;                          //引入相关类
5   public class GameView extends SurfaceView implements SurfaceHolder.Callback{
6       ……//该处省略了成员变量的声明代码，将在之后进行介绍
7       public GameView(KLSDActivity activity) {              //构造器
8           super(activity);
9           this.activity = activity;                         //得到 activity 的引用
10          getHolder().addCallback(this);                    //添加Callback接口的实现
11          init();                                           //调用初始化方法
12          gameViewDrawThread = new GameViewDrawThread(this);   //初始化刷帧线程
13      }
14      public void init(){                                   //初始化方法
15          ……//该处省略了图片等资源的初始化代码，将在之后进行介绍
16      }
17      private void init2(){                                 //初始化方法
18          ……//该处省略了数独数组的初始化代码，将在之后进行介绍
19      }
20      protected void onDraw(Canvas canvas) {                //绘制方法
21          ……//该处省略了界面的绘制代码，将在之后进行介绍
22      }
23      public void drawkey(){                                //数字键盘的绘制方法
24          ……//该处省略了数字键盘的绘制代码，将在之后进行介绍
25      }
26      public boolean onTouchEvent(MotionEvent event){       //触屏监听器
27          ……//该处省略了屏幕的监听代码，将在之后进行介绍
28      }
29      private boolean isFinish(){                           //判断棋盘是否填满
30          ……//该处省略了判断棋盘中是否全部填满的判断代码，将在之后进行介绍
31      }
32      private void isWin(){                          //判断是否赢，改变 status 值表示输赢
33          ……//该处省略了是否输赢的判断代码，将在之后进行介绍
34      }
35      private void mouseDown(float x, float y) {            //鼠标按下的时候
36          ……//该处省略了鼠标按下的处理代码，将在之后进行介绍
37      }
38      private void mouseUP(float x, float y){               //鼠标抬起的时候
39          ……//该处省略了屏幕抬起的处理代码，将在之后进行介绍
40      }
41      public void surfaceChanged(SurfaceHolder holder,int format,int width,int
height){}
42      public void surfaceCreated(SurfaceHolder holder) {   //View 创建时被调用
43          ……//该处省略了相关线程的启动工作，读者可自行查阅随书光盘中的源代码
44      }
45      public void surfaceDestroyed(SurfaceHolder holder) { //View 摧毁时被调用
46          ……//该处省略了相关线程的释放工作，读者可自行查阅随书光盘中的源代码
```

```
47        }
48    }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 GameView.java。

- 第 7~13 行为该类的构造器，在构造器中得到 activity 的引用并初始化相关资源。
- 第 14~19 行为资源的初始化方法，其代码将在后面进行介绍。
- 第 20~25 行为主要的绘制方法，其中 onDraw 主要负责界面的绘制，而 drawkey 负责数字键盘的绘制。
- 第 29~34 行为判断是否填满棋盘及判断是否胜利的方法。
- 第 35~40 行为屏幕事件的处理代码，其中 35~37 行为屏幕被按下事件的处理代码，而第 38~40 行为屏幕被抬起事件的处理代码。
- 第 41~47 行实现了接口中的三个抽象方法，其中 42~44 行会在 View 创建时被调用，主要负责相关线程的启动工作，而第 45~47 行的方法则会在 View 被摧毁时被调用，主要负责相关线程的释放工作。

② 接下来对游戏界面 GameView 的成员变量进行介绍，代码如下所示。

```
1   KLSDActivity activity;                  //activity 的引用
2   int alpha = 100;                        //透明度
3   float span = (float) 1;                 //难度(1-10)
4   int status = 0 ;                        //0 游戏中，1 暂停中，2 游戏胜利，3 游戏失败
5   int time=0;                             //时间
6   boolean tishi;                          //是否提示
7   Bitmap background;                      //大背景
8   Bitmap small_backgroud;                 //背景方格
9   Bitmap stop;                            //“暂停”按钮
10  Bitmap change;                          //“换题”按钮
11  Bitmap drop;                            //“放弃”按钮
12  Bitmap help;                            //“提示”按钮
13  Bitmap go_on;                           //暂停图像
14  Bitmap keyDown;                         //选中后单元格样式
15  Bitmap win;                             //胜利
16  Bitmap fail;                            //失败
17  Bitmap select;                          //选中的单元格
18  Bitmap timeBitmap;                      //时间中间的冒号
19  Bitmap heart;                           //有提示时绘制的提示
20  Bitmap exit;                            //是否退出图片
21  Bitmap key_background;                  //数字按键背景
22  Bitmap key_bitmap;                      //小数字键盘
23  Bitmap[] number_bitmap = new Bitmap[10];    //默认的数字图片
24  Bitmap[] number_input = new Bitmap[10];     //输入的数字图片
25  Bitmap[] time_bitmap = new Bitmap[10];      //时间
26  Paint paint;
27  boolean drawkey = false;
28  int[][] num;                                //用来存放生成的数字即正确答案
29  int[][] inputNum;                           //用来存放输入的数字
30  int[][] outputNum;                          //用来存放系统的数字
31  float scale = (float) 0.8;                  //数字键盘上数字缩放的比例
32  Bitmap[] small_number;
33  int r = 39;                                     //数字键盘小圆圈的半径
```

```
34  int keyx;
35  int keyy;                                   //数字键盘的位置
36  int downx;
37  int downy;
38  ShuDuSuanFa sdsf;                           //生成数独的算法类
39  TimeThread tt;                              //时间线程
40  GameViewDrawThread gameViewDrawThread;
```

> 说明：该段代码为 GameView 的成员变量的声明，各个成员变量的含义和参见代码中的注释。

16.7　计时线程与数字键盘线程的开发

经过前面几节，程序的框架基本已经开发完成，接下来将介绍游戏中两个后台线程的开发，为以后业务处理的开发做好准备。

16.7.1　计时线程的开发

TimeThread 类的实现比较简单，只需每隔一秒钟更改一下 GameView 中表示时间的属性即可，代码如下。

```
1   package wyf.ytl;                                    //声明所在包
2   public class TimeThread extends Thread{
3       GameView gameView;                              //声明 GameView 的引用
4       boolean flag=true;                              //循环标志位
5       public TimeThread(GameView gameView){           //构造器
6           this.gameView=gameView;                     //得到 GameView 的引用
7       }
8       public void run(){                              //重写的 run 方法
9           while(flag){
10              gameView.time++;                        //时间自加
11              try{
12                  Thread.sleep(1000);                 //睡眠一秒种
13              }catch(Exception e){                    //捕获异常
14                  e.printStackTrace();                //打印异常信息
15              }
16          }
17      }
18  }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 TimeThread.java。

> 说明：该段线程的实现非常简单，只需每隔一秒种将 gameView 中的 time 值加一即可。

> 提示：Android 平台中多线程开发与普通 SE 的多线程开发并没有很大的区别，主要考虑平台的能力与程序的性能。

16.7.2 数字键盘线程的开发

数字键盘线程主要对数字键盘的渐隐渐消进行处理，当玩家点击屏幕时，会根据情况绘制数字键盘，在绘制过程中该线程定时改变图片的透明度，达到渐隐渐消的效果，代码如下。

```
1   package wyf.ytl;                                          //声明所在包
2   import android.graphics.Bitmap;                           //引入相关类
3   import android.graphics.Canvas;                           //引入相关类
4   import android.graphics.Paint;                            //引入相关类
5   import android.graphics.Bitmap.Config;                    //引入相关类
6   public class DrawKeyThread extends Thread {
7       GameView gameView;                                    //gameView 的引用
8       int span = 40;                                        //透明度的更改步数
9       boolean flag = true;                                  //循环标志位
10      Bitmap key_bitmap;                                    //图片
11      int alpha=0;                                          //透明度
12      public DrawKeyThread(Bitmap key_bitmap,GameView gameView){   //构造器
13          this.key_bitmap = key_bitmap;                     //得到图片的引用
14          this.gameView = gameView;                         //得到 gameView 的引用
15      }
16      public void run(){                                    //重写的 run 方法
17          while(flag){                                      //循环
18              Bitmap b = Bitmap.createBitmap(key_bitmap.getWidth(),
19                  key_bitmap.getHeight(), Config.ARGB_8888);
20              Canvas c = new Canvas(b);                     //创建画布
21              Paint p = new Paint();                        //创建画笔
22              p.setAlpha(alpha);                            //设置透明度
23              c.drawBitmap(key_bitmap, 0, 0, p);            //绘制图片
24              gameView.key_bitmap = b;
25              alpha+=span;                                  //改变透明度
26              if(alpha>255){                                //当透明度过大时
27                  alpha = 255;
28                  flag = false;                             //停止循环
29              }
30              try{
31                  Thread.sleep(100);                        //睡眠指定毫秒数
32              }catch(Exception e){                          //捕获异常
33                  e.printStackTrace();                      //打印异常信息
34              }
35          }
36      }
37  }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 DrawKeyThread.java。

- 第 7~11 行用于声明相应对象的引用及该类中用到的属性。
- 第 12~15 行是构造器，得到 gameView 的引用及需要处理的数字键盘图片的引用。
- 第 16~36 行是重写的 run()方法，该方法是每 100 毫秒循环一次，每次循环创建一张新图片，将从 gameView 传过来的图片通过设置某个透明度画到新图片上，然后将新图片赋给 GameView 的数字键盘的图片引用。

16.8　数独生成器的开发

本节将对数独的生成器 ShuDuSuanFa 进行开发，该类基本上是数学计算，通过一定的算法产生所需要的数独数组，通过特定的接口供其他类使用。

该类的目的是创建一个二维数组来表示数独矩阵。使矩阵的每行、每列、每块都没有重复的数字。算法的简单思路是先随机取出一个 0~9 的数字，然后检查其所在的行、列、块是否都符合要求。当符合要求时继续填充下一个，而当不符合要求时，再次随机取出一个没有取出过的数字，再判断。当 9 个数字都取出过后还没有找到符合要求的数字时，进行回退处理，即将最后一个取出的符合要求的数字进行重新取值，直到所有数字全部填充完毕。接下来对该类进行详细介绍，步骤如下。

① 首先介绍的是该类对外的接口，代码如下所示。

```
1   package wyf.ytl;                                          //声明所在包
2   public class ShuDuSuanFa{
3       int[][] n = new int[9][9];                            //存储数字的数组
4       int[] num = {1,2,3,4,5,6,7,8,9};                      //生成随机数字的源数组，随机数
字从该数组中产生
5       public boolean checkLine(int col){                    //检查列是否符合要求
6           ……//该处省略了检查列是否满足要求的代码，将在之后给出
7       }
8       public boolean checkNine(int row,int col){            //检查 3×3 区域是否符合要求
9           ……//该处省略了检查 3X3 区域是否满足要求的代码，将在之后给出
10      }
11     public boolean checkRow(int row){                      //检查行是否符合要求
12          ……//该处省略了检查行是否满足要求的代码，将在之后给出
13      }
14      public int generateNum(int row,int col,int time){     //产生 1~9 之间的随机数字
15          if(time == 0){                                    //第一次尝试时，初始化随机数字源数组
16              for(int i = 0;i < 9;i++){
17                  num[i] = i + 1;
18              }
19          }
20          //第 10 次填充，表明该位置已经卡住，则返回 0，由主程序处理退回
21          if(time == 9){
22              return 0;                                     //返回 0
23          }
24          //不是第一次填充
25          //生成随机数字，该数字是数组的下标，取数组 num 中该下标对应的数字为随机数字
26          int ranNum = (int)(Math.random()*(9-time));
27          //把数字放置在数组倒数第 time 个位置，
28          int temp = num[8 - ranNum];
29          num[8 - ranNum] = num[ranNum];
30          num[ranNum] = temp;
31          return num[8 - ranNum];
32      }
33      public int[][] getShuDu(){                            //生成数字
34          for(int i = 0;i < 9;i++){
35              int time = 0;                                 //尝试填充的数字次数
36              for(int j = 0;j < 9;j++){                     //填充数字
37                  n[i][j] = generateNum(i,j,time);
```

```
38                      //如果返回值为0，则代表卡住，退回处理
39                      if(n[i][j] == 0){
40                          if(j > 0){                  //不是第一列，则倒退一列
41                              j-=2;
42                              continue;               //跳出本次循环
43                          }else{                      //是第一列，则倒退到上一行的最后一列
44                              i--;                    // i 自减
45                              j = 8;
46                              continue;               //跳出本次循环
47                          }
48                      }
49                      if(isCorret(i,j)){              //成功
50                          time = 0;                   //将 time 置 0
51                      }else{
52                          time++;                     //time 自加
53                          j--;                        //j 自减
54                      }
55                  }
56              }
57              return n;                               //返回值
58          }
59          public boolean isCorret(int row,int col){   //是否满足行、列和九宫区域不重复的要求
60              return (checkRow(row)&checkLine(col)&checkNine(row,col));//返回布尔值
61          }
62      }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 ShuDuSuanFa.java。

- 第 3~4 行分别为存储结果的数字及源数组。
- 第 5~13 行的三个方法分别检查行、列及 3×3 区域是否满足要求。
- 第 14~32 行为产生随机数的方法，原则是已经尝试过的数字将不会被取到。
- 第 33~58 行是该类对外的接口，其他类可以通过调用该类的 getSudoku()方法取得填充好满足数独要求的二维数组。
- 第 59~61 行是检查所填数字是否符合要求的方法。

② 接下来对上述代码中的三个监测方法进行介绍，代码如下所示。

```
1       public boolean checkLine(int col){              //检查列是否符合要求
2           for(int j = 0;j < 8;j++){                   //循环监测
3               if(n[j][col] == 0){                     //当为 0 时
4                   continue;                           //跳出本次循环
5               }
6               for(int k =j + 1;k< 9;k++){             //循环
7                   if(n[j][col] == n[k][col]){
8                       return false;                   //返回监测失败
9                   }
10              }
11          }
12          return true;                                //返回监测成功
13      }
14      public boolean checkNine(int row,int col){      //检查 3×3 区域是否符合要求
15          int j = row/3*3;                            //获得左上角的坐标
16          int k = col/3*3;
17          for(int i = 0;i < 8;i++){                   //循环比较
18              if(n[j + i/3][k + i % 3] == 0){
19                  continue;                               //跳出本次循环
```

```
20                }
21                for(int m = i+ 1;m < 9;m++){                    //循环
22                    if(n[j + i/3][k + i % 3] == n[j + m/3][k + m % 3]){
23                        return false;                           //返回 false
24                    }
25                }
26            }
27            return true;                                        //返回检测成功
28        }
29        public boolean checkRow(int row){                       //检查行是否符合要求
30         for(int j = 0;j < 8;j++){                              //循环
31            if(n[row][j] == 0){                                 //当为 0 时
32                continue;                                       //跳出本次循环
33            }
34            for(int k =j + 1;k < 9;k++){                        //循环
35                if(n[row][j] == n[row][k]){                     //当有相同的数字时
36                    return false;                               //返回检查失败
37                }
38            }
39        }
40        return true;                                            //返回监测成功
41    }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 ShuDuSuanFa.java。

- 第 1~13 行是检查列是否符合要求的方法，在循环中检查已经存在的数字是否有重复的数字。
- 第 14~28 行检查的是块内是否符合要求，同样是循环检查是否还有重复的数字。
- 第 29~40 行检查的是行是否符合要求。

> 提示：该类选用的算法比较简单，如果读者希望增加游戏的可玩性，可以重新开发该类，采用比较有效率的算法，只需保持对外的使用接口不变即可。

16.9　游戏界面逻辑方法的实现

在前面游戏界面框架的开发中，由于当时有很多关联类没有开发，所以业务代码无法进行开发，现在已经完成相关类的开发，本节将回到游戏界面，对其业务方法进行完善。

16.9.1　初始化方法的完善

本节将对 GameView 中的两个初始化方法进行详细介绍。

1．初始化图片方法 init()的完善

该方法只是初始化游戏中用到的所有图片，实现起来很简单，具体代码如下。

```
1   public void init(){
2       paint = new Paint();                                        //初始化画笔
3       background = BitmapFactory.decodeResource(getResources(), R.drawable.background);
4       //初始化背景小方格图元
5       small_backgroud= BitmapFactory.decodeResource(getResources(), R.drawable.
small_background);
```

```
    6        //初始化下面的四个按钮
    7        stop = BitmapFactory.decodeResource(getResources(), R.drawable.stop1); //停止
    8        change = BitmapFactory.decodeResource(getResources(), R.drawable.change1);//换题
    9        drop = BitmapFactory.decodeResource(getResources(), R.drawable.drop1);  //放弃
    10       help = BitmapFactory.decodeResource(getResources(), R.drawable.help1); //提示
    11       key_background = BitmapFactory.decodeResource(getResources(), R.drawable.
key_background);
    12       number_bitmap[0] = BitmapFactory.decodeResource(getResources(), R.drawable.b0);
    13       ……//该处省略了数字图片的初始化代码，读者可以自行查阅随书光盘的源代码
    14       number_bitmap[9] = BitmapFactory.decodeResource(getResources(), R.drawable.b9);
    15       number_input[0] = BitmapFactory.decodeResource(getResources(), R.drawable.a0);
    16       ……//该处省略了数字图片的初始化代码，读者可以自行查阅随书光盘的源代码
    17       number_input[9] = BitmapFactory.decodeResource(getResources(), R.drawable.a9);
    18       go_on = BitmapFactory.decodeResource(getResources(), R.drawable.go_on); //继续
    19       keyDown = BitmapFactory.decodeResource(getResources(), R.drawable.c0);
    20       win = BitmapFactory.decodeResource(getResources(), R.drawable.win);  //胜利
    21       fail = BitmapFactory.decodeResource(getResources(), R.drawable.fail); //失败界面
    22       select = BitmapFactory.decodeResource(getResources(), R.drawable.select); //选择
    23       heart = BitmapFactory.decodeResource(getResources(), R.drawable.heart); //提示的红心
    24       //时间的图片
    25       Matrix matrix = new Matrix();                              //创建 Matrix
    26       matrix.postScale((float)1.5, (float)1.5);                  //缩放
    27       timeBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.time);
    28       Bitmap temp0 = BitmapFactory.decodeResource(getResources(), R.drawable.time0);
    29       time_bitmap[0] = Bitmap.createBitmap(temp0, 0, 0, temp0.getWidth(), temp0.
getHeight(), matrix, true);
    30       for(int i=1; i<=9; i++){                                   //循环
    31           time_bitmap[i] = Bitmap.createBitmap(number_input[i], 0, 0,
    32               number_input[i].getWidth(), number_input[i].getHeight(), matrix, true);
    33       }
    34       //初始化数字键盘的图片
    35       Matrix matrix2 = new Matrix();                             //创建 Matrix
    36       matrix2.postScale(scale, scale);
    37       small_number = new Bitmap[10];                             //初始化图片数组
    38       for(int i=0; i<number_bitmap.length; i++){
    39           small_number[i] = Bitmap.createBitmap(number_input[i], 0, 0,
    40               number_input[i].getWidth(), number_input[i].getHeight(), matrix2, true);
    41       }
    42       exit = BitmapFactory.decodeResource(getResources(), R.drawable.exit); //退出图片
    43       init2();                                                   //调用其他初始化方法
    44   }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 GameView.java 的 init 方法。

> 说明：该段代码主要是对图片资源进行初始化，在图片初始化时，会用到 Matrix 将图片进行缩放，同时在最后调用 init2 方法来初始化数独数组。

2．初始化界面方法 initView()的完善

该方法主要对界面的数据进行初始化，代码如下。

```
    1   private void init2(){                                  //初始化数独数组
    2       //生成数独
    3       sdsf = new ShuDuSuanFa();                          //创建数独算法类
    4       num = sdsf.getShuDu();                             //得到数组数组
    5       //初始化用于存放输入数字的数组
    6       inputNum = new int[9][9];                          //创建数组
```

```
7         for(int i=0;i<inputNum.length; i++){              //循环
8             for(int j=0; j<inputNum[i].length; j++){
9                 inputNum[i][j] = 0;                       //将数组中所有的位置置 0
10            }
11        }
12        outputNum = new int[9][9];                        //创建用于显示的数组
13        for(int i=0;i<num.length; i++){
14            for(int j=0; j<num[i].length; j++){           //循环
15                if(10*Math.random()>span){                //取一定的概率
16                    outputNum[i][j] = num[i][j];
17                }else {
18                    outputNum[i][j] = 0;                  //将一部分填上 0
19                }
20            }
21        }
22        //启动时间线程
23        tt = new TimeThread(this);                        //创建线程
24        tt.start();                                       //启动线程
25    }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 GameView.java 的 init2 方法。

- 第 3~4 行初始化数独生成器，并得到数独数组。
- 第 6~11 行创建用于存放用户输入数字的数组，并对数组初始化，填充 0 代表此单元格还没有用户的数字。
- 第 12~21 行创建游戏中默认显示的数字，这里采用的算法非常简单，只是从正确答案的数组中随机去掉一部分，将需要去掉的位置用 0 代替。

16.9.2　简单逻辑方法的完善

下面对两个简单的逻辑处理方法进行完善，为之后的屏幕事件处理方法的开发做准备。之前已经将框架开发完成，此时读者只需将下面的方法代码覆盖到框架中即可。

1．判断结束方法 isFinish()的完善

该方法很容易实现，只需检查屏幕上的所有单元格是否已被填满即可，代码如下。

```
1   private boolean isFinish(){                           //判断是否填满
2       int count = 0;                                    //声明计数器
3       int temp;                                         //声明临时变量
4       for(int i=0; i<9; i++){
5           for(int j=0; j<9; j++){
6               temp = inputNum[i][j];                    //检测此位置是否有玩家输入的数字
7               if(temp != 0){
8                   count++;                              //计数器自加
9               }
10              temp = outputNum[i][j];                   //检测此位置是否有系统默认的数字
11              if(temp != 0){
12                  count++;                              //计数器自加
13              }
14          }
15      }
16      if(count<81){
17          return false;                          //当计数器小于 81 时，说明没填满
18      }
```

```
19        else{
20            return true;                          //说明已经填满，返回 true
21        }
22    }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 GameView.java。

- 第 4~15 行循环检查 inputNum、outputNum 数组中不为 0 位置的个数。
- 第 16~21 行是当不为 0 的个数不到 81 时，说明还没有填满，返回 false，否则说明已经填满，返回 true。

2．判断输赢方法 isWin()的完善

该方法是游戏结束后用来判断本次游戏是输是赢的，逻辑很简单，只需要得到用户输入的数字，然后检查与正确答案是否相同即可，代码如下。

```
1    private void isWin(){                         //判断是否赢，改变 status 值表示输赢
2         int temp;                                //临时变量的声明
3         for(int i=0; i<9; i++){
4             for(int j=0; j<9; j++){
5                 temp =  inputNum[i][j];          //取出用户输入的数字
6                 if(temp != 0){
7                     if(temp != num[i][j]){       //和正确答案不同
8                         status = 3;              //游戏失败
9                         return;                  //有一个不同的直接返回
10                    }
11                }
12            }
13        }
14        status = 2;                              //游戏胜利
15   }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 GameView.java。

> 说明：在判断输赢的逻辑方法中得到输赢结果后只改变表示状态的值，无须做其他的操作，这样在屏幕绘制或其他方法中，只需检查状态值即可。

16.9.3　屏幕事件处理方法的完善

本节是对处理屏幕事件的两个方法的完善，一个是处理屏幕被按下时的事件，另一个是处理触控笔离开屏幕时的事件，Android 平台对屏幕事件处理的方法与思路基本上与 Java ME 中的相同。

1．屏幕监听方法的完善

首先对屏幕监听方法 onTouchEvent 进行完善，代码如下所示。

```
1    public boolean onTouchEvent(MotionEvent event){          //触屏监听器
2        float x = event.getX();                              //得到事件的 X、Y 坐标
3        float y = event.getY();
4        if(status == 2 || status == 3){                  //游戏胜利或失败时
5            activity.myHandler.sendEmptyMessage(2);
6        }else if(status == 4){                           //是否退出
7            if(x>70 && x<105 && y>230 && y<265){         //单击“是”
8                System.exit(0);                          //退出游戏
```

```
9            }
10           else if(x>200 && x<235 && y>230 && y<265){  //单击“否”
11               status = 0;
12           }
13       }else{
14           switch(event.getAction()){
15               case MotionEvent.ACTION_DOWN:           //按下
16                   mouseDown(x,y);                     //调用屏幕按下的事件处理方法
17                   break;
18               case MotionEvent.ACTION_UP:             //抬起
19                   mouseUP(x,y);                       //调用屏幕抬起事件的处理方法
20                   break;
21           }
22       }
23       return true;                                    //返回 true
24   }
```

- 第 2~3 行得到事件发生的 *X*、*Y* 坐标。
- 第 4~6 行为当游戏界面胜利或失败时的屏幕监听，当玩家在胜利或失败后点击屏幕时，会向 activity 发送 Handler 消息。
- 第 6~12 行当游戏界面显示的是“询问玩家是否真正退出”的界面时，会判断玩家单击的是“是”还是“否”并进行处理。
- 第 13~22 行为其他情况的事件处理方法，根据屏幕事件的类型调用不同的方法来处理屏幕事件。

2．处理屏幕按下事件方法 mouseDown ()的完善

当玩家点击屏幕时，通过 onTouchEvent()方法对事件进行捕捉，然后判断事件的类型，调用 mouseDown()方法处理屏幕按下事件。该方法的开发步骤如下。

① 代码框架的搭建，代码如下所示。

```
1   private void mouseDown(float x, float y) {              //鼠标按下的时候
2       if(drawkey == true){                                //如果数字键盘存在
3           if(x>keyx&&x<keyx+key_background.getWidth()
4           &&y>keyy&&y<keyy+key_background.getHeight()){   //按在数字键盘上
5               for(int i=0; i<9; i++){
6                   if(x>(((keyx+r)+r*Math.sin((40.0*i)/180*Math.PI))+4)&&
7                   x<(((keyx+r)+r*Math.sin((40.0*i)/180*Math.PI))+4+small_
number[i].getWidth())
8                   &&y>(keyy+(r-r*Math.cos((40.0*i)/180*Math.PI)))
9                   &&y<(keyy+(r-r*Math.cos((40.0*i)/180*Math.PI))+small_
number[i].getHeight())){
10                      inputNum[downy][downx] = i+1;       //点击的数字
11                  }
12              }
13          }
14          if(isFinish()){                          //判断是否已经填满,填满就出输赢
15              isWin();                                    //判断输赢
16              return;
17          }
18          drawkey = false;                            //将 deawkey 标志位设成 false
19          return;
20       }
21       //计算单击的位置
22       if(x>5&&x<108){
```

```
23              downx = (int) ((x-15)/25);                  //计算 downx 值
24          }
25          if(x>108&&x<210){
26              downx = (int) ((x-123)/25)+3;               //计算 downx 值
27          }
28          if(x>210&&x<310){
29              downx = (int) ((x-225)/25)+6;               //计算 downx 值
30          }
31          if(y>25&&y<125){
32              downy = (int) ((y-35)/25);                  //计算 downy 值
33          }
34          if(y>125&&y<230){
35              downy = (int) ((y-140)/25)+3;               //计算 downy 值
36          }
37          if(y>230&&y<330){
38              downy = (int) ((y-245)/25)+6;               //计算 downy 值
39          }
40          if(y<330){                                      //如果点击位置 Y 坐标小于 330 就
显示数字键盘
41              if(x>200){
42                  this.keyx = (int) (x-key_background.getWidth());    //keyx 的大小
43              }else{
44                  this.keyx = (int) x;                    //等于 x
45              }
46              this.keyy = (int) y;                        //得到 keyy
47              if(outputNum[downy][downx] == 0){           //当为 0 时
48                  if(tishi == true){
49                      inputNum[downy][downx] = num[downy][downx];
50                      if(isFinish()){                     //判断是否已经填满,填满就出结果
51                          isWin();
52                          return;                         //返回
53                      }
54                      tishi = false;                      //将提示标志位设成 false
55                  }else{
56                      drawkey = true;                     //显示数字键盘
57                      drawkey();                          //绘制数字键盘
58                      DrawKeyThread dk = new DrawKeyThread(key_bitmap,this);
59                      dk.start();                         //启动线程
60                  }
61              }
62          }
63          ……//该处省略了屏幕下部控制面板的事件响应，将在之后进行介绍
64      }
```

代码位置：见随书光盘中源代码/第 16 章/ KLSD/src/wyf/ytl 目录下的 GameView.java。

- 第 2~20 行是当数字键盘存在时，判断所点的位置是否在数字键盘上，当点击在数字键盘上时，再判断所点击的数字，并将该数字填入相应的单元格。
- 第 21~39 行计算玩家所点击的位置处于哪个单元格中。
- 第 40 行判断点击的位置是否在游戏区域内，只有在区域内才有弹出数字键盘的必要。
- 第 41~45 行表示当玩家点击的位置在第三宫所在的列时，让数字键盘在点击位置的左侧显示，以防数字键盘显示不全。而在前两列时，使数字键盘在点击位置的右侧显示，如图 16-14 和图 16-15 所示。
- 第 48~59 行判断之前是否已经按过提示按钮，如果按过提示按钮，则直接将正确答案

填入玩家单击处，否则弹出数字键盘。

图 16-14　在左侧出现的效果图

图 16-15　在右侧出现的效果图

② 接下来将对屏幕下方控制区域的各个按钮的点击事件进行处理，将下列代码插入到上面代码的第 63 行处。

```
1    if(x>150&&x<200&&y>380&&y<410){                          //按下“暂停”按钮
2        stop = BitmapFactory.decodeResource(getResources(), R.drawable.stop2);
3        if(status == 0){
4            status = 1;                                      //暂停
5        }else {
6            status = 0;
7            //重新启动时间线程
8            tt = new TimeThread(this);                       //创建时间线程
9            tt.start();                                      //启动线程
10       }
11   }
12   if(x>230&&x<280&&y>380&&y<410){                          //按下“换题”按钮
13       change = BitmapFactory.decodeResource(getResources(), R.drawable.change2);
14       time = 0;
15       tt.flag = false;
16       status = 0;                                          //将状态值设成 0
17       init2();                                             //重新初始化界面
18   }
19   if(x>150&&x<200&&y>420&&y<450){                          //按下“放弃”按钮
20       drop = BitmapFactory.decodeResource(getResources(), R.drawable.drop2);
21       this.status = 4;                                     //状态值设置成 4
22   }
23   if(x>230&&x<280&&y>420&&y<450){                          //按下“提示”按钮
24       help = BitmapFactory.decodeResource(getResources(), R.drawable.help2);
25       tishi = true;                                        //将提示标志位设置 true
26   }
```

代码位置：见随书光盘源代码/第 16 章/ KLSD/src/wyf/ytl 目录下 GameView.java 的 mouseDown 方法。

- 第 1~11 行处理“暂停”按钮被按下的事件，只有暂停中和游戏中“暂停”按钮才可用，并实现暂停中和游戏中状态的循环。
- 第 12~18 行处理“换题”按钮被按下的事件，游戏时间重新开始计算，然后调用 init2 方法重新初始化游戏界面。
- 第 19~22 行处理“放弃”按钮被按下的事件，在代码中先初始化是否放弃的提示图片，然后将状态值设置成 4。

- 第 23~26 行处理“提示”按钮的按下事件，同样只有在暂停或游戏中才有效，实现很简单，只需将一个标志位设置成 true 即可。

3．处理屏幕触控笔抬起事件方法 mouseUP()的完善

mouseUP()方法很简单，只需判断在哪个按钮处抬起，并将相应按钮的图片改变即可，代码如下。

```
1   private void mouseUP(float x, float y) {                    //鼠标抬起的时候
2       if(x>150&&x<200&&y>380&&y<410){                         //在暂停处抬起
3           stop = BitmapFactory.decodeResource(getResources(), R.drawable.stop1);
4       }
5       if(x>230&&x<280&&y>380&&y<410){                         //“换题”按钮
6           change = BitmapFactory.decodeResource(getResources(), R.drawable.change1);
7       }
8       if(x>150&&x<200&&y>420&&y<450){                         //“放弃”按钮
9           drop = BitmapFactory.decodeResource(getResources(), R.drawable.drop1);
10      }
11      if(x>230&&x<280&&y>420&&y<450){                         //“提示”按钮
12          help =
BitmapFactory.decodeResource(getResources(),
R.drawable.help1);
13      }
14  }
```

代码位置：见随书光盘源代码/第 16 章/ KLSD/src/wyf/ytl 目录下 GameView.java 的 mouseUP 方法。

> 说明：在该方法中先判断玩家所点击的按钮，然后将相应按钮的图片更换即可。

当按钮被按下时会根据按下按钮的不同绘制不同的效果，例如当“提示”按钮被按下时，效果如图 16-16 所示。

图 16-16 “提示”按钮被按下

16.10 游戏界面绘画方法的完善

16.10.1 数字键盘的绘制方法 drawKey ()的完善

drawKey()将需要显示的数字键盘先画到一张图片上，等重画屏幕时只需将生成的图片一次性输出即可，代码如下所示。

```
1   public void drawKey(){                                      //画小数字键盘
2       Bitmap b = Bitmap.createBitmap(key_background.getWidth(),
3            key_background.getHeight(),
4            Config.ARGB_8888);                                 //创建一个空的图片
5       Canvas c = new Canvas(b);                               //得到该图片的画布
6       c.drawBitmap(key_background, 0, 0, paint);              //先将数字键盘的背景画到 b 上
7       for(int i=0; i<9; i++){                                 //绘制数字键盘上的 9 个数字
8           c.drawBitmap(small_number[i+1],
9               (float)((0+r)+r*Math.sin((40.0*i)/180*Math.PI))+2,
10              (float)(0+(r-r*Math.cos((40.0*i)/180*Math.PI))),
11              paint);
12      }
```

```
13        key_bitmap = b;                                    //将新生成的图片 b 赋给成员变量
14    }
```

代码位置：见随书光盘源代码/第 16 章/ KLSD/src/wyf/ytl 目录下 GameView.java 的 drawKey 方法。

- 第 2~5 行先创建一个空的图片并得到该图片的画布。
- 第 6~12 行先将数字键盘的背景绘制到创建的图片上，然后按照角度一一将 9 个数字绘制到创建的图片上。
- 第 13 行将创建的图片存储到成员变量 key_bitmap 中。

> 提示：双缓冲的基本思想是先将需要绘制的图像绘制到一张图片上，然后在合适的时间一次性将这张图片绘制到屏幕，相当于在绘制线程与屏幕之间增加了一个缓冲层。

16.10.2　绘画方法 onDraw()的完善

该方法主要负责游戏界面的绘制工作，刷帧线程会定时调用该方法进行屏幕刷新，该方法的开发步骤如下所示。

① 对该方法的整体架构进行介绍，代码如下所示。

```
1   protected void onDraw(Canvas canvas) {                   //重写的绘制方法
2       super.onDraw(canvas);
3       canvas.drawColor(Color.BLACK);                       //绘制背景
4       canvas.drawBitmap(background, 0, 0, paint);          //画背景图片
5       for(int i=0; i<3; i++){                              //画背景小方格
6           for(int j=0; j<3; j++){
7               canvas.drawBitmap(small_backgroud, 5+104*j, 25+104*i, paint);
8           }
9       }
10      //画下面的四个按钮
11      canvas.drawBitmap(stop, 150,380, paint);             //画“暂停”按钮
12      canvas.drawBitmap(change, 230,380, paint);           //画“换题”按钮
13      canvas.drawBitmap(drop, 150,420, paint);             //画“放弃”按钮
14      canvas.drawBitmap(help, 230,420, paint);             //画“提示”按钮
15      if(status == 2){                                     //游戏胜利
16          tt.flag = false;
17          canvas.drawBitmap(win, 40, 160, paint);
18      }else if(status == 3){                               //游戏失败
19          tt.flag = false;
20          canvas.drawBitmap(fail, 80, 180, paint);
21      }else if(status == 4){                               //是否退出
22          canvas.drawBitmap(exit, 70, 180, paint);
23      }else if(status == 0){                               //游戏中
24      ……//该处省略了游戏过程中的绘制代码，将在之后进行介绍
25      }
26      else if(status == 1){                                //暂停中
27          tt.flag = false;
28          canvas.drawBitmap(go_on, 80, 180, paint);        //绘制图片
29      }
30      //绘制时间
31      canvas.drawBitmap(timeBitmap, 60, 400, paint);
32      int temp = time/60;                                  //计算时间
33      String timeStr = temp+"";                            //转换成字符串
34      if(timeStr.length()<2){
```

```
35            timeStr = "0" + timeStr;                    //如果长度不到两位，前面加 0
36        }
37        for(int i=0;i<2;i++){                           //根据数字绘制
38            int tempScore=timeStr.charAt(i)-'0';
39            canvas.drawBitmap(time_bitmap[tempScore], 20+i*20, 400, paint);
40        }
41        //画秒钟
42        temp = time%60;                                 //计算
43        timeStr = temp+"";                              //转换成字符串
44        if(timeStr.length()<2){                         //如果长度不到两位，前面加 0
45            timeStr = "0" + timeStr;
46        }
47        for(int i=0;i<2;i++){                           //根据数字绘制
48            int tempScore=timeStr.charAt(i)-'0';
49            canvas.drawBitmap(time_bitmap[tempScore], 80+i*20, 400, paint);
50        }
51    }
```

代码位置：见随书光盘源代码/第 16 章/ KLSD/src/wyf/ytl 目录下 GameView.java 的 onDraw 方法。

- 第 3 行将背景设置为黑色。
- 第 4 行绘制背景图片，因为背景图片比较大，所以绘制在屏幕的左上角即可填充全屏幕。
- 第 5~9 行绘制背景小方格，即每宫的背景图片。
- 第 11~14 行负责绘制屏幕下方四个与玩家交互的按钮，第 15~29 行根据当前游戏的状态进行不同的绘制。
- 第 30~50 行负责绘制屏幕下方的时间，先根据得到的时间计算出有多少分钟。为了美观，如果计算出的分钟是个位数，则在计算出的时间前加上 0。然后计算秒钟，最后根据计算出的数字绘制时间图片。

② 接下来对上述方法进行完善，将下列代码插入到上述代码的第 24 行处。

```
1    else if(status == 0){                                  //游戏中
2        if(tishi == true){
3            canvas.drawBitmap(heart, 10, 340, paint);      //当有提示红心时绘制红心
4        }
5        //画全部数字
6        tt.flag = true;                                    //启动时间线程
7        int x;
8        int y;                                             //声明临时变量
9        for(int i = 0;i < 9;i++){
10           for(int j = 0;j < 9;j++){
11               x = 16+28*j;
12               y = 35+29*i;
13               if(j>=3&&j<6){                             //在第二列时调整 X 坐标
14                   x += 20;
15               }
16               if(j>=6){                                  //在第三列时调整 X 坐标
17                   x += 42;
18               }
19               if(i>=3&&i<6){                             //在第二行时调整 Y 坐标
20                   y += 18;
21               }
22               if(i>=6){                                  //在第三行时调整 Y 坐标
23                   y += 34;
24               }
25               canvas.drawBitmap(number_bitmap[outputNum[i][j]], x, y, paint);
```

```
                                                                    //绘制数字
26              }
27          }
28          //画用户输入的数字
29          for(int i = 0;i < 9;i++){
30              for(int j = 0;j < 9;j++){
31                  x = 16+28*j;
32                  y = 35+29*i;
33                  if(j>=3&&j<6){                                  //在第二宫所在的列
34                      x += 20;
35                  }
36                  if(j>=6){                                       //在第三宫所在的列
37                      x += 42;
38                  }
39                  if(i>=3&&i<6){                                  //在第二行时调整 Y 坐标
40                      y += 18;
41                  }
42                  if(i>=6){                                       //在第三行时调整 Y 坐标
43                      y += 34;
44                  }
45                  canvas.drawBitmap(number_input[inputNum[i][j]], x, y, paint);
                                                                        //绘制数字
46              }
47          }
48          //画选中的单元格
49          x = 15+28*downx;                                        //计算 X、Y 坐标
50          y = 34+29*downy;
51          if(downx>=3&&downx<6){                                  //在第二列时调整坐标
52              x += 19;                                            //将 X 坐标加 19
53          }
54          if(downx>=6){                                           //在第三列时调整坐标
55              x += 41;                                            //将 X 坐标加 41
56          }
57          if(downy>=3&&downy<6){                                  //在第二行时调整坐标
58              y += 17;                                            //将 Y 坐标加 17
59          }
60          if(downy>=6){                                           //在第三行时调整坐标
61              y += 34;                                            //将 Y 坐标 34
62          }
63          canvas.drawBitmap(select, x, y, paint);                 //画选中的单元格
64          if(drawkey == true){                                    //画数字键盘
65              canvas.drawBitmap (key_bitmap, keyx, keyy, paint);
66          }
67      }
```

代码位置：见随书光盘源代码/第 16 章/ KLSD/src/wyf/ytl 目录下 GameView.java 的 onDraw 方法。

- 第 2~4 行检查是否需要绘制表示提示的红心，需要时在指定位置绘制红心。
- 第 9~27 行根据默认灰色图片的大小计算出灰色图片绘制的坐标，因为每个宫之间有一定的距离，所以需要对计算出来的坐标进行调整。
- 第 29~47 行同样是对坐标的计算，然后按坐标绘制用户输入的数字。
- 第 48~63 行计算出点击的单元格坐标，然后绘制被选中的单元格，使玩家知道当前选中的单元格是哪个。
- 第 64~66 行当需要绘制数字键盘时，在指定位置绘制数字键盘。

16.11 游戏界面刷帧线程的实现

前面已经将游戏界面开发完毕，接下来将对游戏界面的刷帧线程进行开发，该类主要负责定时刷新游戏界面，代码如下所示。

```
1   package wyf.ytl;                                          //声明所在包
2   import android.graphics.Canvas;                           //引入相关类
3   import android.view.SurfaceHolder;                        //引入相关类
4   public class GameViewDrawThread extends Thread{
5       GameView gameView;                                    //gameView 的引用
6       private int sleepSpan = 100;                          //睡眠的时间
7       private boolean flag = true;                          //循环标记位
8       private SurfaceHolder surfaceHolder;                  //surfaceHolder 的引用
9       public GameViewDrawThread(GameView gameView){         //构造器
10          this.gameView = gameView;                         //得到 welcomeView 的引用
11          surfaceHolder = gameView.getHolder();             //得到 surfaceHolder 的引用
12      }
13      public void run() {                                   //重写的 run 方法
14          Canvas c;                                         //声明画布
15          while (this.flag) {                               //循环
16              c = null;                                     //将画布置空
17              try {
18                  //锁定整个画布
19                  c = this.surfaceHolder.lockCanvas(null);
20                  synchronized (this.surfaceHolder) {       //同步
21                      gameView.onDraw(c);                   //调用绘制方法
22                  }
23              } finally {                                   //用finally保证一定被执行
24              if (c != null) {
25                      //更新屏幕显示内容
26                      this.surfaceHolder.unlockCanvasAndPost(c);
27                  }
28              }
29              try{
30                  Thread.sleep(sleepSpan);                  //睡眠指定毫秒数
31              }catch(Exception e){                          //捕获异常
32                  e.printStackTrace();                      //打印异常信息
33              }
34          }
35      }
36      public void setFlag(boolean flag){                    //循环标记位的 set 方法
37          this.flag = flag;
38      }
39  }
```

代码位置：见随书光盘源代码/第 16 章/ KLSD/src/wyf/ytl 目录下 GameViewDrawThread.java。

- 第 5~8 行声明 gameView 的引用及线程的睡眠时间。
- 第 9~12 行为该类的构造器，在构造器中得到 WelcomeView 及 SurfaceHolder 的引用。
- 第 13~35 行为该类的 run 方法，在方法中，先得到并锁定画布，然后调用 onDraw 方法进行界面的绘制，最后释放画布，更新屏幕的显示内容。
- 第 36~38 行为循环标志位的 set 方法，外部线程会通过调用该方法进行循环变量的设置。

16.12　游戏的优化与改进

到 16.11 节为止，本游戏的基本功能已经开发完毕，但仍有很多方面可以进行优化和改进，有能力的读者可以继续对本游戏进行提升，可以提升的地方如下。

1．音效的添加

在玩家眼中，好的游戏总会是有声有色丰富多彩的，离开了生动的音效，游戏的体验就会大打折扣，所以对音效的处理是游戏开发中必不可少的。本游戏中，并没有添加对音效的处理，有能力的玩家可以在适当的地方添加上音效，以提升玩家的体验。

2．玩家的定制

游戏应该开发成尽可能地可定制，因为玩家的水平不一，过于困难或者过于简单都会使游戏失去吸引力。本游戏中，应该添加上游戏难度的选择功能，使玩家根据自己的情况选择不同的等级进行游戏，这样才更加合理。

> 提示：如果能将玩家的定制结果保存起来，不用每次运行游戏重新设置，会让游戏更加人性化。

3．成绩的存储

如果能在游戏的最后加上游戏成绩的存储与排行，更会大大提高玩家对本游戏的体验。Android 的存储技术在前面的章节中已经讲解过，读者可以参考前面的讲解，自行开发完成此功能。