B 基于 Android 的数独游戏

B.1 需求分析

数独游戏是一款比较传统的游戏，它由 81 个（9 行*9 列）单元格组成，玩家要试着在这些单元格中填入 1~9 的数字，使数字在每行、每列和每区（3 行*3 列的部分）中都只出现一次，游戏开始时，部分单元格中已经填入一些已知的数字，玩家只需要在剩下的空单元格中填入数字。

一道正确的数独谜题只有一个答案。

B.2 程序开发及运行环境

数独游戏的软件开发环境及运行环境具体如下。

- 操作系统：Windows 7。
- JDK 环境：Java SE Development KET(JDK) version 7。
- 开发工具：Eclipse 4.4.2+Android 5.0。
- 开发语言：Java、XML。
- 运行平台：Windows、Linux 各版本。

B.3 程序文件夹组织结构

在编写项目代码之前，需要制定好项目的文件夹组织结构，如不同的 Java 包存放不同的窗体、公共类、数据模型、工具类或者图片资源等，这样不但可以保证团队开发的一致性，也可以规范系统的整体架构。创建完程序中可能用到的文件夹或者 Java 包之后，在开发时，只需将创建的类文件或者资源文件保存到相应的文件夹中即可。数独游戏的文件夹组织结构如图 B-1 所示。

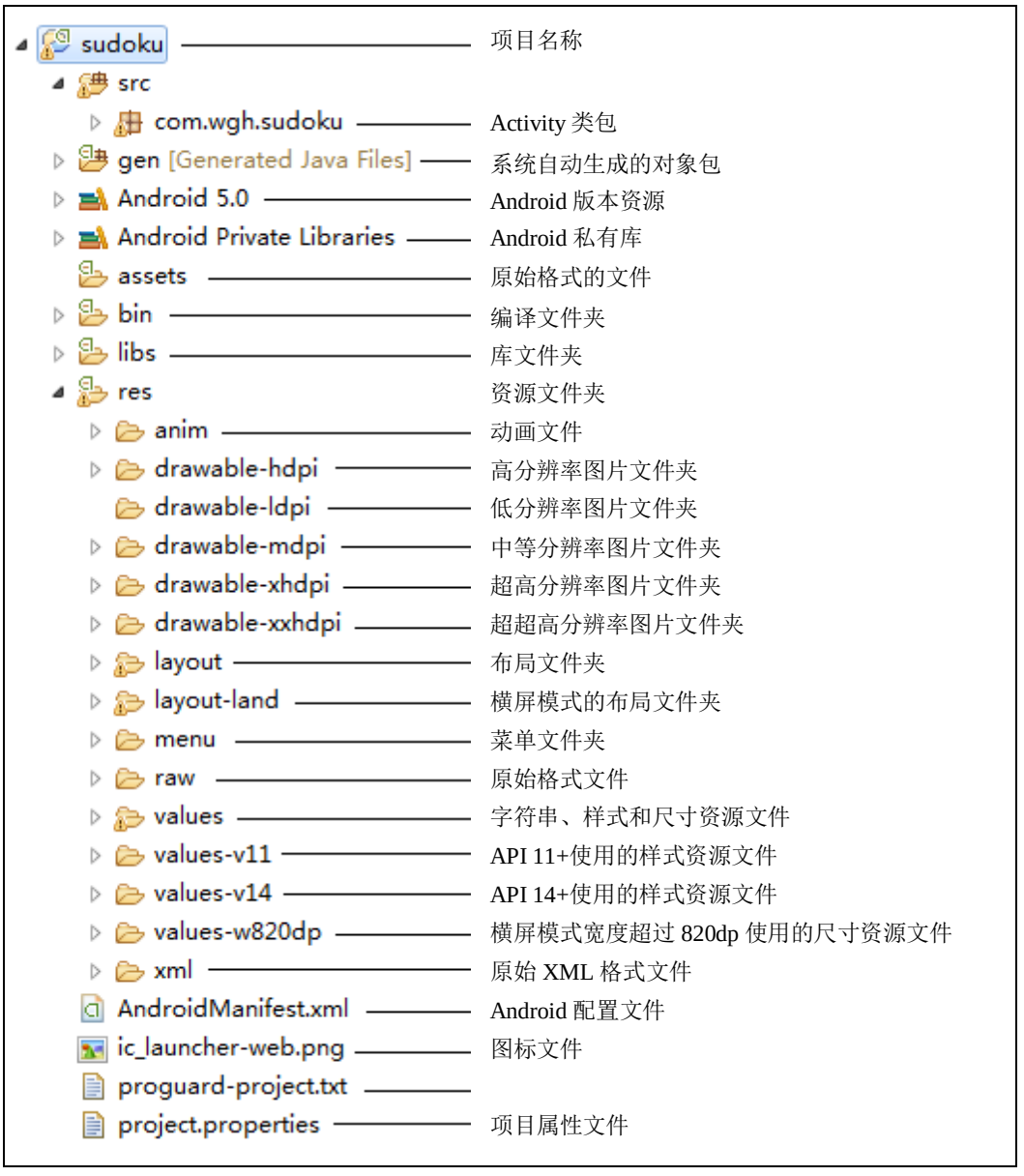

图 B-1　文件夹组织结构

B.4　公共资源文件

数独游戏中的公共资源文件主要有字符串资源文件、数组资源文件和颜色资源文件，设置完公共资源文件之后，在开发程序时，用户即可很方便的进行调用。本节将对数独游戏中的公共资源文件进行讲解。

B.4.1 字符串资源文件

字符串资源存储在 strings.xml 文件中，主要定义游戏中用到的公共字符串，主要代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Android 版的数独游戏</string>
    <string name="app_name">数独</string>
    <string name="btn1">继续</string>
    <string name="about_text">数独游戏是一款比较传统的游戏，它由 81 个（9 行*9 列）单元格组成，
玩家要试着在这些单元格中填入 1~9 的数字，
        使数字在每行、每列和每区（3 行*3 列的部分）中都只出现一次，游戏开始时，部分单元格中已经填入一些
已知的数字，玩家只需要在剩下的空单元格中填入数字。
        一道正确的数独谜题只有一个答案。
    </string>
    <string name="about_title">关于数独游戏</string>
    <string name="settings_label">设置...</string>
    <string name="settings_title">游戏设置</string>
    <string name="settings_shortcut">s</string>
    <string name="music_title">音乐</string>
    <string name="music_summary">播放背景音乐</string>
    <string name="hints_title">提示</string>
    <string name="hints_summary">是否显示提示</string>
    <!-- 开始游戏  -->
    <string name="new_game_title">难度</string>
    <string name="easy_label">简单</string>
    <string name="medium_label">一般</string>
    <string name="hard_label">高级</string>
    <string name="game_title">数独游戏</string>
    <string name="no_moves_label">不能填充任何数字</string>
    <string name="keypad_title">键盘</string>
</resources>
```

B.4.2 数组资源文件

数组资源存储在 arrarys.xml 文件中，主要定义数独游戏中的 3 种难易程度，主要代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <array name="difficulty">
     <item>@string/easy_label</item>
     <item>@string/medium_label</item>
     <item>@string/hard_label</item>
  </array>
</resources>
```

B.4.3 颜色资源文件

颜色资源存储在 colors.xml 文件中，主要定义游戏中用到的各种背景色，比如主界面背景色、

填充数字的单元格背景色、提醒背景色等，主要代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
        <color name="background">#75FF6600</color>
        <color name="puzzle_background">#ffe6f0ff</color>
        <color name="puzzle_hilite">#FFFFFFFF</color>
        <color name="puzzle_light">#64c6d4ef</color>
        <color name="puzzle_dark">#6456648f</color>
        <color name="puzzle_foreground">#ff000000</color>
        <color name="puzzle_hint_0">#64ff0000</color>
        <color name="puzzle_hint_1">#6400ff80</color>
        <color name="puzzle_hint_2">#2000ff80</color>
        <color name="puzzle_selected">#64ff8000</color>
    </resources>
```

B.5　游戏主窗体设计

主窗体是程序操作过程中必不可少的，它是与用户交互中的重要环节。通过主窗体，用户可以调用系统相关的各子模块，快速掌握本系统中所实现的各个功能。数独游戏的主窗体主要为用户提供继续游戏、新建游戏、查看数据游戏规则及退出游戏的链接按钮。主窗体运行结果如图 B-2 所示。

图 B-2　数独游戏主窗体

B.5.1　设计系统主窗体布局文件

数独游戏的主窗体有两种布局方式，一种针对竖屏，一种针对横屏，其中，针对竖屏的布局文件存放在 res/layout 目录下，实现代码如下。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="30dip"
```

```
    >
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <Button android:id="@+id/button1"
         android:text="@string/btn1"
         android:layout_height="wrap_content"
         android:layout_width="wrap_content"/>
    <Button android:id="@+id/button2"
         android:text="新游戏"
         android:layout_height="wrap_content"
         android:layout_width="wrap_content"/>
    <Button android:id="@+id/button3"
         android:text="关于"
         android:layout_height="wrap_content"
         android:layout_width="wrap_content"/>
    <Button android:id="@+id/button4"
         android:text="退出"
         android:layout_height="wrap_content"
         android:layout_width="wrap_content"/>
    </LinearLayout>
</LinearLayout>
```

针对横屏的布局文件存放在 res/layout-land 目录下，实现代码如下。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="15dip"
    >
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:paddingLeft="20dip"
        android:paddingRight="20dip"
            >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:layout_marginBottom="20dip"
        android:textSize="24.5sp"
        />
    <TableLayout
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:stretchColumns="*"
         >
         <TableRow>
             <Button android:id="@+id/button1"
                 android:text="@string/btn1"
                 android:layout_height="wrap_content"
                 android:layout_width="wrap_content"/>
             <Button android:id="@+id/button2"
                 android:text="新游戏"
                 android:layout_height="wrap_content"
                 android:layout_width="wrap_content"/>
             <Button android:id="@+id/button3"
                 android:text="关于"
                 android:layout_height="wrap_content"
                 android:layout_width="wrap_content"/>
             <Button android:id="@+id/button4"
                 android:text="退出"
                 android:layout_height="wrap_content"
                 android:layout_width="wrap_content"/>
         </TableRow>
     </TableLayout>
     </LinearLayout>
</LinearLayout>
```

B.5.2 为界面中的按钮添加监听事件

在 com.wgh.sudoku 包中创建一个 SudokuActivity.java 文件，该文件中主要是为界面中的按钮添加监听事件，代码如下。

```
public class SudokuActivity extends Activity implements OnClickListener {
     private static final String TAG="Sudoku";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        View continueButton=this.findViewById(R.id.button1);   //为继续按钮绑定单击事件
        continueButton.setOnClickListener(this);
        View newButton=this.findViewById(R.id.button2);
        newButton.setOnClickListener(this);
        View aboutButton=this.findViewById(R.id.button3);
        aboutButton.setOnClickListener(this);
        View exitButton=this.findViewById(R.id.button4);  //为退出按钮添加单击事件监听
        exitButton.setOnClickListener(this);
    }
     @Override
     public void onClick(View v) {
          Intent i;
          switch (v.getId()){
              case R.id.button1:
                  StartGame(GameActivity.DIFFICULTY_CONTINUE);
                  break;
              case R.id.button2:
```

```
                openNewGameDialog();
                break;
            case R.id.button3:
                i=new Intent(this,About.class);
                startActivity(i);
                break;
            case R.id.button4:
                finish();
                break;
        }
    }
    @Override
    protected void onPause() {
        super.onPause();
        Music.stop(this);
    }
    @Override
    protected void onResume() {
        super.onResume();
        Music.play(this,R.raw.jasmine);
    }
    private void openNewGameDialog() {
        new AlertDialog.Builder(this)
        .setTitle(R.string.new_game_title)
        .setItems(R.array.difficulty,new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int i) {
                StartGame(i);
            }
        })
        .show();
    }
    private void StartGame(int i) {
        Log.d(TAG,"clicked on "+i);
        Intent intent=new Intent(this,GameActivity.class);
        intent.putExtra(GameActivity.KEY_DIFFICULTY, i);
        startActivity(intent);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        MenuInflater inflater=getMenuInflater();
        inflater.inflate(R.menu.menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        super.onOptionsItemSelected(item);
        switch (item.getItemId()){
        case R.id.settings:
            startActivity(new Intent(this,SettingsActivity.class));
            return true;
        }
        return false;
    }
}
```

B.5.3 绘制数独游戏界面

在com.wgh.sudoku包中创建一个PuzzleView.java文件，该文件中主要是绘制数独游戏的界面，代码如下。

```
public class PuzzleView extends View{
    private static final String TAG="sudoku";
    private final GameActivity game;
    private float width;
    private float height;
    private int selX;
    private int selY;
    private final Rect selRect=new Rect();
    //记录当前位置
    private static final String SELX="selX";
    private static final String SELY="selY";
    private static final String VIEW_STATE="viewState";
    private static final int ID=42;
    public PuzzleView(Context context) {
        super(context);
        this.game=(GameActivity)context;
        setFocusable(true);
        setFocusableInTouchMode(true);
        setId(ID);                                          //设置 ID 用于记录当前位置
    }
    //**************用于记录当前位置***************/
    @Override
    protected void onRestoreInstanceState(Parcelable state) {
        Log.d(TAG, "onRestoreInstanceState");
        Bundle bundle=(Bundle)state;
        select(bundle.getInt(SELX),bundle.getInt(SELY));
        super.onRestoreInstanceState(bundle.getParcelable(VIEW_STATE));
        return;
    }
    @Override
    protected Parcelable onSaveInstanceState() {
        Parcelable p=super.onSaveInstanceState();
        Log.d(TAG, "onSaveInstanceState");
        Bundle bundle=new Bundle();
        bundle.putInt(SELX, selX);
        bundle.putInt(SELY, selY);
        bundle.putParcelable(VIEW_STATE, p);
        return bundle;
    }
    /*********************************************/
    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        width=w/9f;
        height=h/9f;
        getRect(selX,selY,selRect);
        Log.d(TAG,"onSizeChanged:width"+width+"height"+height);
        super.onSizeChanged(w, h, oldw, oldh);
    }
    @Override
```

```
protected void onDraw(Canvas canvas) {
    Paint background=new Paint();
    background.setColor(getResources().getColor(R.color.puzzle_background));
    canvas.drawRect(0,0,getWidth(),getHeight(),background);
    //绘制网格线
    Paint dark=new Paint();
    dark.setColor(getResources().getColor(R.color.puzzle_dark));
    Paint hilite=new Paint();
    hilite.setColor(getResources().getColor(R.color.puzzle_hilite));
    Paint light=new Paint();
    light.setColor(getResources().getColor(R.color.puzzle_light));
    //绘制次要网格线
    for(int i=0;i<9;i++){
        canvas.drawLine(0, i*height, getWidth(), i*height, light);
        canvas.drawLine(0, i*height+1, getWidth(), i*height+1, hilite);
        canvas.drawLine(i*width, 0, i*width, getHeight(), light);
        canvas.drawLine(i*width+1, 0, i*width+1, getHeight(), hilite);
    }
    //绘制主要网格线
    for(int i=0;i<9;i++){
        if(i%3!=0){
            continue;
        }else{
            canvas.drawLine(0, i*height, getWidth(), i*height, dark);
            canvas.drawLine(0, i*height+1, getWidth(), i*height+1, hilite);
            canvas.drawLine(i*width, 0, i*width, getHeight(), dark);
            canvas.drawLine(i*width+1, 0, i*width+1, getHeight(), hilite);
        }
    }
    //输出数字
    Paint foreground=new Paint(Paint.ANTI_ALIAS_FLAG);
    foreground.setColor(getResources().getColor(R.color.puzzle_foreground));
    foreground.setStyle(Style.FILL);
    foreground.setTextSize(height*0.75f);
    foreground.setTextScaleX(width/height);
    foreground.setTextAlign(Align.CENTER);              //设置文字居中
    FontMetrics fm=foreground.getFontMetrics();
    float x=width/2;
    float y=height/2-(fm.ascent+fm.descent)/2;
    for(int i=0;i<9;i++){
        for(int j=0;j<9;j++){
            canvas.drawText(this.game.getTileString(i,j),          i*width+x,
j*height+y, foreground);
        }
    }
    //绘制 hints
    if(SettingsActivity.getHints(getContext())){        //判断是否显示高亮提示
        Paint hint=new Paint();
        int c[]={getResources().getColor(R.color.puzzle_hint_0),
                 getResources().getColor(R.color.puzzle_hint_1),
                 getResources().getColor(R.color.puzzle_hint_2)};
        Rect r=new Rect();
        for(int i=0;i<9;i++){
            for(int j=0;j<9;j++){
```

```
                    int mouseleft=9-game.getUsedTiles(i,j).length;
                    if(mouseleft<c.length){
                        getRect(i,j,r);
                        hint.setColor(c[mouseleft]);
                        canvas.drawRect(r,hint);

                    }
                }
            }
        }
        //绘制选定区
        Log.d(TAG,"selRect"+selRect);
        Paint selected=new Paint();
        selected.setColor(getResources().getColor(R.color.puzzle_selected));
        canvas.drawRect(selRect,selected);
        super.onDraw(canvas);
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        Log.d(TAG,"onKeyDown:keycode="+keyCode+"event="+event);
        switch(keyCode){
            case KeyEvent.KEYCODE_DPAD_UP:
                select(selX,selY-1);
            case KeyEvent.KEYCODE_DPAD_DOWN:
                select(selX,selY+1);
            case KeyEvent.KEYCODE_DPAD_LEFT:
                select(selX-1,selY);
            case KeyEvent.KEYCODE_DPAD_RIGHT:
                select(selX+1,selY);
                break;

            case KeyEvent.KEYCODE_0:
            case KeyEvent.KEYCODE_SPACE:
                setSelectedTile(0);
                break;
            case KeyEvent.KEYCODE_1:
                setSelectedTile(1);
                break;
            case KeyEvent.KEYCODE_2:
                setSelectedTile(2);
                break;
            case KeyEvent.KEYCODE_3:
                setSelectedTile(3);
                break;
            case KeyEvent.KEYCODE_4:
                setSelectedTile(4);
                break;
            case KeyEvent.KEYCODE_5:
                setSelectedTile(5);
                break;
            case KeyEvent.KEYCODE_6:
                setSelectedTile(6);
                break;
            case KeyEvent.KEYCODE_7:
                setSelectedTile(7);
```

```
                    break;
                case KeyEvent.KEYCODE_8:
                    setSelectedTile(8);
                    break;
                case KeyEvent.KEYCODE_9:
                    setSelectedTile(9);
                    break;
                case KeyEvent.KEYCODE_ENTER:
                case KeyEvent.KEYCODE_DPAD_CENTER:
                    game.showKeyPadOrError(selX,selY);
                    break;
                default:
                    return super.onKeyDown(keyCode, event);
            }
            return true;
        }
        public void setSelectedTile(int tile) {
            if(game.setTileIfValid(selX,selY,tile)){
                invalidate();
            }else{
                Log.d(TAG,"setSelectedTile:invalid"+tile);
                startAnimation(AnimationUtils.loadAnimation(game, R.anim.shake));
            }
        }
        @Override
        public boolean onTouchEvent(MotionEvent event) {
            if(event.getAction()!=MotionEvent.ACTION_DOWN){
                return super.onTouchEvent(event);
            }
            select((int)(event.getX()/width),(int)(event.getY()/height));
            game.showKeyPadOrError(selX,selY);
            Log.d(TAG,"onTouchEvent:x"+selX+",y"+selY);
            return true;
        }
        private void select(int x, int y) {
            invalidate(selRect);
            selX=Math.min(Math.max(x, 0), 8);
            selY=Math.min(Math.max(y, 0), 8);
            getRect(selX,selY,selRect);
            invalidate(selRect);
        }
        private void getRect(int x, int y, Rect rect) {

    rect.set((int)(x*width),(int)(y*height),(int)(x*width+width),(int)(y*height+height
));
        }
    }
```

B.5.4 数独游戏的实现算法

在 com.wgh.sudoku 包中创建一个 GameActivity.java 文件，该文件中实现的功能主要有：根据难易程度显示不同的游戏界面、保存并继续当前游戏、数独游戏的算法实现等，代码如下。

```
    public class GameActivity extends Activity {
        private static final String TAG = "sudoku";
        public static final String KEY_DIFFICULTY = "difficulty";
```

```
public static final int DIFFICULTY_EASY = 0;
public static final int DIFFICULTY_MEDIUM = 1;
public static final int DIFFICULTY_HARD = 2;
private int puzzle[] = new int[9 * 9];
private PuzzleView puzzleView;
private final int used[][][] = new int[9][9][];
private final String easyPuzzle
        = "360000000004230800000004200"
        + "070460003820000014500013020"
        + "001900000007048300000000045";
private final String mediumPuzzle
        = "650000070000506000014000005"
        + "007009000002314700000700800"
        + "500000630000201000030000097";
private final String hardPuzzle
        = "009000000806050205010078000"
        + "000000700706040102004000000"
        + "000720903090301080000000600";
boolean success = false;                                    // 判断是否成功
//继续前一游戏
private static final String PREF_PUZZLE="puzzle";
protected static final int DIFFICULTY_CONTINUE=-1;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.d(TAG, "onCreate");
    int diff = getIntent().getIntExtra(KEY_DIFFICULTY, DIFFICULTY_EASY);
    puzzle = getPuzzle(diff);                       // 接收难度级别并返回一次数独游戏
    Log.d(TAG, "onCreate11" + diff);
    calculateUsedTiles();                                   // 实现真正的游戏逻辑
    Log.d(TAG, "onCreate22" + diff);
    puzzleView = new PuzzleView(this);
    setContentView(puzzleView);
    puzzleView.requestFocus();
     getIntent().putExtra(KEY_DIFFICULTY, DIFFICULTY_CONTINUE);//恢复已保存的游戏
}
//获取游戏的难易程序
private int[] getPuzzle(int diff) {
    String puz;
    switch (diff) {
    case DIFFICULTY_CONTINUE:
        puz=getPreferences(MODE_PRIVATE).getString(PREF_PUZZLE,easyPuzzle);
        break;
    case DIFFICULTY_HARD:
        puz = hardPuzzle;
        break;
    case DIFFICULTY_MEDIUM:
        puz = mediumPuzzle;
        break;
    case DIFFICULTY_EASY:
    default:
        puz = easyPuzzle;
        break;
    }
```

```
        return fromPuzzleString(puz);
    }
    public void showKeyPadOrError(int x, int y) {
        int tiles[] = getUsedTiles(x, y);
        if (tiles.length == 9) {
            Toast toast = Toast.makeText(this, R.string.no_moves_label,
                    Toast.LENGTH_SHORT);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
        } else {
            Log.d(TAG, "showKeyPad:used=" + toPuzzleString(tiles));
            Dialog v = new KeyPad(this, tiles, puzzleView);
            v.show();
        }
    }
    private String toPuzzleString(int[] puz) {
        StringBuilder buf = new StringBuilder();
        for (int element : puz) {
            buf.append(element);
        }
        return buf.toString();
    }
    public boolean setTileIfValid(int x, int y, int value) {
        int tiles[] = getUsedTiles(x, y);
        if (value != 0) {
            for (int tile : tiles) {
                if (tile == value) {
                    return false;
                }
            }
        }
        setTile(x, y, value);
        calculateUsedTiles();                                    // 实现真正的游戏逻辑
        /*************** 判断游戏是否成功 ************************/
        success = true;
        label: for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                if (getTile(i, j) == 0) {
                    success = false;
                    break label;
                }
            }
        }
        if (success) {
            Log.d(TAG, "数独游戏成功! ");
            // 弹出带确定按钮的提示对话框
            new AlertDialog.Builder(GameActivity.this)
                    .setTitle(TAG)
                    .setMessage("恭喜您，成功! ")
                    .setPositiveButton("确定",
                            new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog,
                                        int which) {
```

```
                                    finish();          // 返回游戏主界面
                                }
                            }).show();
        }
        /*****************************************************/
        return true;
    }
    private void calculateUsedTiles() {
        for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                used[i][j] = calculateUsedTiles(i, j);
            }
        }
    }
    private int[] calculateUsedTiles(int x, int y) {
        int c[] = new int[9];
        // 水平方向
        for (int i = 0; i < 9; i++) {
            if (i == y) {
                continue;
            }
            int t = getTile(x, i);
            if (t != 0) {
                c[t - 1] = t;
            }
        }
        // 垂直方向
        for (int i = 0; i < 9; i++) {
            if (i == x) {
                continue;
            }
            int t = getTile(i, y);
            if (t != 0) {
                c[t - 1] = t;
            }
        }
        int startx = (x / 3) * 3;
        int starty = (y / 3) * 3;
        for (int i = startx; i < startx + 3; i++) {
            for (int j = starty; j < starty + 3; j++) {
                if (i == x && j == y) {
                    continue;
                }
                int t = getTile(i, j);
                if (t != 0) {
                    c[t - 1] = t;
                }
            }
        }
        int nused = 0;
        for (int t : c) {
            if (t != 0) {
                nused++;
            }
        }
```

```
        int cl[] = new int[nused];
        nused = 0;
        for (int t : c) {
            if (t != 0) {
                cl[nused++] = t;
            }
        }
        return cl;
    }
    /**
     * 功能：获取指定单元格中的数字
     * @param x
     * @param y
     * @return
     */
    private int getTile(int x, int y) {
        return puzzle[y * 9 + x];
    }
    /**
     * 功能：设置指定单元格中的数字
     * @param x
     * @param y
     * @param value
     */
    private void setTile(int x, int y, int value) {
        puzzle[y * 9 + x] = value;
    }
    protected int[] getUsedTiles(int x, int y) {
        return used[x][y];
    }
    public String getTileString(int x, int y) {
        int v = getTile(x, y);
        if (v == 0) {
            return "";
        } else {
            return String.valueOf(v);
        }
    }
    static protected int[] fromPuzzleString(String string) {
        int[] puz = new int[string.length()];
        for (int i = 0; i < puz.length; i++) {
            puz[i] = string.charAt(i) - '0';
        }
        return puz;
    }
    @Override
    protected void onPause() {                                    //暂停游戏
        super.onPause();
        Music.stop(this);
        getPreferences(MODE_PRIVATE).edit()
        .putString(PREF_PUZZLE, toPuzzleString(puzzle)).commit();//保存游戏当前状态
    }
    @Override
    protected void onResume() {                                   //恢复游戏
```

```
            super.onResume();
            Music.play(this,R.raw.lhydd);
        }
    }
```

B.6　虚拟键盘模块设计

用户在数独游戏的游戏界面中填写数字时，单击空白处，会出现一个虚拟键盘，以便提示用户可以填写哪些数字。虚拟键盘的运行效果如图 B-3 所示。

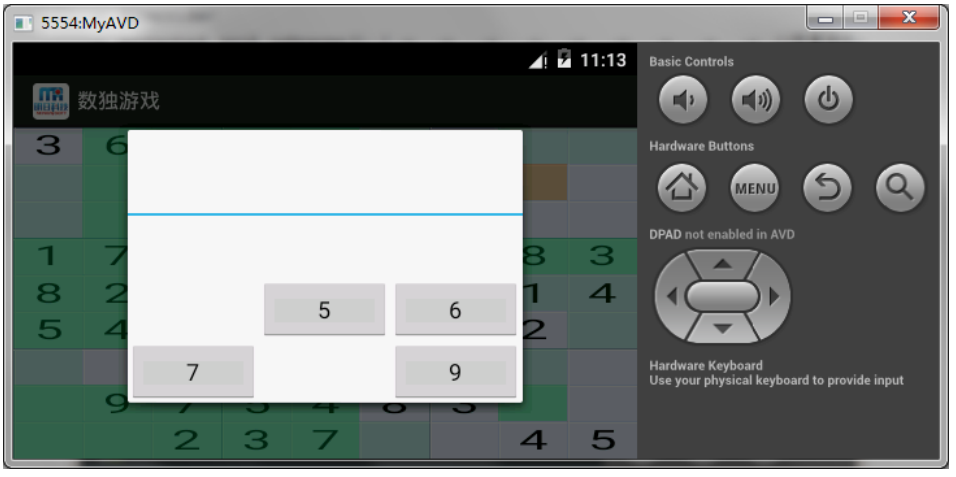

图 B-3　虚拟键盘

B.6.1　设计模拟键盘布局文件

在 res/layout 目录下新建一个 keypad.xml，用来作为虚拟键盘的布局文件，该布局文件使用 TableLayout 进行布局，并添加 9 个 Button 组件，分别表示 9 个数字按钮，实现代码如下。

```
<TableLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/keypad"
  android:orientation="vertical"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:stretchColumns="*">
  <TableRow>
     <Button android:id="@+id/keypad_1" android:text="1"/>
     <Button android:id="@+id/keypad_2" android:text="2"/>
     <Button android:id="@+id/keypad_3" android:text="3"/>
  </TableRow>
  <TableRow>
     <Button android:id="@+id/keypad_4" android:text="4"/>
     <Button android:id="@+id/keypad_5" android:text="5"/>
     <Button android:id="@+id/keypad_6" android:text="6"/>
  </TableRow>
    <TableRow>
     <Button android:id="@+id/keypad_7" android:text="7"/>
```

```
        <Button android:id="@+id/keypad_8" android:text="8"/>
        <Button android:id="@+id/keypad_9" android:text="9"/>
    </TableRow>
</TableLayout>
```

B.6.2 在虚拟键盘中显示可以输入的数字

在 com.wgh.sudoku 包中创建一个 KeyPad.java 文件，该文件的布局文件设置为 keypad.xml。在 KeyPad.java 文件中，主要根据其他单元格的数字和数独游戏规则，在虚拟键盘中显示当前单元格可以输入的数字，代码如下。

```
public class KeyPad extends Dialog{
    private static final String TAG="sudoku";
    private final View keys[]=new View[9];
    private View keypad;
    private final int useds[];
    private final PuzzleView puzzleView;
    public KeyPad(Context context,int useds[],PuzzleView puzzleView){
        super(context);
        this.useds=useds;
        this.puzzleView=puzzleView;
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.keypad);
        findViews();
        for(int element:useds){
            keys[element-1].setVisibility(View.INVISIBLE);
        }
        setListeners();
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        int tile=0;
        switch(keyCode){
        case KeyEvent.KEYCODE_0:
        case KeyEvent.KEYCODE_SPACE:tile=0;break;
        case KeyEvent.KEYCODE_1:tile=1;break;
        case KeyEvent.KEYCODE_2:tile=2;break;
        case KeyEvent.KEYCODE_3:tile=3;break;
        case KeyEvent.KEYCODE_4:tile=4;break;
        case KeyEvent.KEYCODE_5:tile=5;break;
        case KeyEvent.KEYCODE_6:tile=6;break;
        case KeyEvent.KEYCODE_7:tile=7;break;
        case KeyEvent.KEYCODE_8:tile=8;break;
        case KeyEvent.KEYCODE_9:tile=9;break;
        default:
            return super.onKeyDown(keyCode, event);
        }
        if(isValid(tile)){
            returnResult(tile);
        }
        return true;
    }
    /**
```

```
     *提取并保存软键盘的所有键和软键盘主窗口的视图
     */
    private void findViews() {
        keypad=findViewById(R.id.keypad);
        keys[0]=findViewById(R.id.keypad_1);
        keys[1]=findViewById(R.id.keypad_2);
        keys[2]=findViewById(R.id.keypad_3);
        keys[3]=findViewById(R.id.keypad_4);
        keys[4]=findViewById(R.id.keypad_5);
        keys[5]=findViewById(R.id.keypad_6);
        keys[6]=findViewById(R.id.keypad_7);
        keys[7]=findViewById(R.id.keypad_8);
        keys[8]=findViewById(R.id.keypad_9);
    }
    private void setListeners() {
        for(int i=0;i<keys.length;i++){
            final int t=i+1;
            keys[i].setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    returnResult(0);

                }
            });
        }
    }
    private boolean isValid(int tile){
        for(int t:useds){
            if(tile==t){
                return false;
            }
        }
        return true;
    }
    private void returnResult(int tile) {
        puzzleView.setSelectedTile(tile);
        dismiss();
    }
}
```

B.7 游戏设置模块设计

游戏设置模块主要对背景音乐和是否显示提示进行设置,该模块中主要通过两个复选框实现。游戏设置模块运行结果如图 B-4 所示。

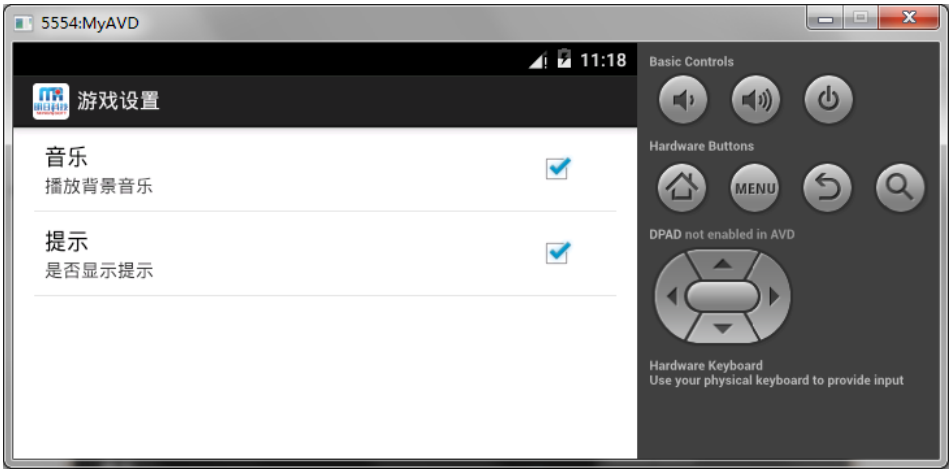

图 B-4　游戏设置

B.7.1　设计游戏设置布局文件

在 res/xml 目录下新建一个 settings.xml，用来作为游戏设置窗体的布局文件，该布局文件中主要使用两个 CheckBoxPreference 组件，用来作为复选框，实现代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
        android:key="music"
        android:title="@string/music_title"
        android:summary="@string/music_summary"
        android:defaultValue="true"/>
    <CheckBoxPreference
        android:key="hints"
        android:title="@string/hints_title"
        android:summary="@string/hints_summary"
        android:defaultValue="true"/>
</PreferenceScreen>
```

B.7.2　设置是否播放背景音乐和显示提示

在 com.wgh.sudoku 包中创建一个 SettingsActivity.java 文件，该文件的布局文件设置为 settings.xml。在 SettingsActivity.java 文件中，主要定义了两个方法，分别设置是否播放背景音乐和是否显示提示，代码如下。

```
public class SettingsActivity extends PreferenceActivity {
    private static final String OPT_MUSIC="music";
    private static final boolean OPT_MUSIC_DEF=true;
    private static final String OPT_HINTS="hints";
    private static final boolean OPT_HINTS_DEF=true;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
         addPreferencesFromResource(R.xml.settings);
    }
    public static boolean getMusic(Context context){
```

```
        return PreferenceManager.getDefaultSharedPreferences(context)
        .getBoolean(OPT_MUSIC,OPT_MUSIC_DEF);
    }
    public static boolean getHints(Context context){
        return PreferenceManager.getDefaultSharedPreferences(context)
        .getBoolean(OPT_HINTS,OPT_HINTS_DEF);
    }
}
```

B.7.3 控制背景音乐的播放与停止

在 com.wgh.sudoku 包中创建一个 Music.java 文件，该文件主要控制背景音乐的播放与停止，代码如下。

```
public class Music {
    private static MediaPlayer mp = null;
    public static void play(Context context, int resource) {
        stop(context);
        if (SettingsActivity.getMusic(context)) {            // 判断是否播放背景音乐
            mp = MediaPlayer.create(context, resource);
            mp.setLooping(true);                             // 是否循环播放
            mp.start();                                      // 开始播放
        }
    }
    public static void stop(Context context) {
        if (mp != null) {
            mp.stop();
            mp.release();
            mp = null;
        }
    }
}
```

B.8 关于模块设计

关于模块主要显示数独游戏的相关规则，关于窗体运行结果如图 B-5 所示。

图 B-5　关于模块

B.8.1　设计关于窗体布局文件

在 res/layout 目录下新建一个 about.xml，用来作为关于窗体的布局文件，该布局文件中，主要使用一个 TextView 组件现实数独游戏的相关规则，实现代码如下。

```
<ScrollView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:padding="10dip"
  >
   <TextView
         android:id="@+id/about_content"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/about_text"
      />
</ScrollView>
```

B.8.2　显示关于信息

在 com.wgh.sudoku 包中创建一个 About.java 文件，该文件中加载 about.xml 布局文件，以便显示关于数独游戏的规则信息，代码如下。

```
public class About extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.about);
    }
}
```

B.9　小结

本章重点讲解了数独游戏的实现过程及安装过程。通过对本章的学习，读者应该能够熟悉 Android 应用的开发流程，并重点掌握数独游戏的游戏规则和实现算法。