学　号	
成　绩	

移动终端开发技术

课程设计报告

题　　目	**2048 游戏**
班　　级	软件二班
学　　号	**201825040239**
姓　　名	岳震波
小组成员	岳震波 吴兆千 史鑫泽 乔中阳
指导教师	陈媛媛

2019 年 **7** 月 **2** 日

目　录

第一章 项目简介

1.1 项目背景

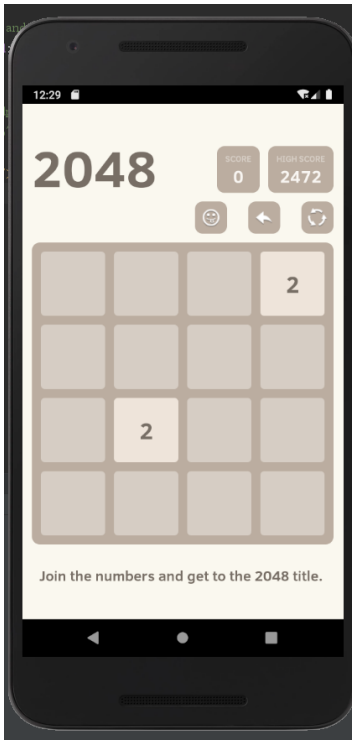
当今市场,已经出现了各种各样,适合不同年龄段和不同人群的游戏。比如:CS、极品飞车、大富翁、魔兽等一些大型游戏,再比如连连看、贪吃蛇、找茬等一些小游戏,都是现在受大众欢迎的娱乐休闲游戏。并且,随着互联网的发展,许多游戏不仅有单机版的, 而且还有网络版的,甚至还有近两年流行起来的网页游戏。迄今为止,游戏产业在我国已经发展为一个庞大的产业,为许多相关行业带来了巨大的经济效益。

2048 这款小游戏是一款流行的数字游戏。第一款 2048 小游戏是由 Gabriele?Cirulli 首度发布在 GitHub 上。2048 是当时基于 1024 和小三传奇这两款的玩法开发的新型数字游戏。

1.2 项目内容介绍

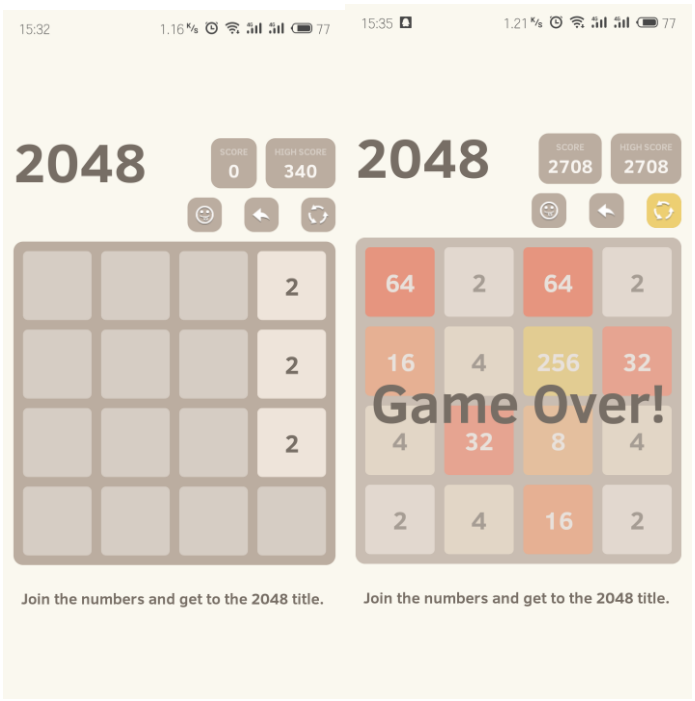
1.2.1 项目完成图

下图为项目完成截图：



1.2.2 真机测试图

下图为真机测试截图：



第二章 详细设计与实现

2.1 运行环境

2048 Android 游戏开发平台为 Android Studio+ JDK1.8+Mysql8.0，模拟测试平台为 Android SDK 模拟器，真机测试平台为 MEIZU16PLUS。

2.2 开发工具及技术简介

以下内容是对本系统所采用的开发工具和技术进行的简单介绍：

2.2.1 开发工具简介

Android Studio 是谷歌推出一个 Android 集成开发工具，基于 IntelliJ IDEA.类似 Eclipse ADT，Android Studio 提供了集成的 Android 开发工具用于开发和调试。

架构组成：在 IDEA 的基础上，Android Studio 提供基于 Gradle 的构建支持、Android 专属的重构和快速修复、提示工具以捕获性能、可用性、版本兼容性等问题支持 ProGuard 和应用签名。

基于模板的向导来生成常用的 Android 应用设计和组件，功能强大的布局编辑器，可以让你拖拉 UI 控件并进行效果预览。

Java 语言的前身是 Sun Microsystems 公司开发的一种用于智能化家电的名为 Oak（橡树）的语言，它的基础是当时最流行的 C 和 C++语言（Sun 公司于 2009 年 4 月被 Oracle 公司收购）。但是，由于一些非技术上的原因，Oak 语言并没有得到迅速的推广。知道 1993 年，WWW（万维网）迅速发展，Sun 公司发现可以利用 Oak 语言的技术来创造含有动态内容的 WWW 网页，于是已受人冷落了的 Oak 语言又被重新的开发和改造。于是便将改造后的 Oak 语言改名为 Java 语言，Java 是太平洋上的一个盛产咖啡的岛屿的名字。终于，在 1995 年，Java 这个被定位于网络应用的程序设计语言被正式推出。

Mysql 是一种采用 T-SQL 语言，基于 C/S 模式的关系型数据库管理系统 Mysql 存储和管理数据有以下优点：

（1）每个数据项都存储在中央位置，所有用户都可在这个位置使用它们；

（2）各个客户端上不单独存储数据项复本，从而消除了因用户不得不确保使用的信息相同所带来的麻烦。系统不需要确保使用当前值更新所有数据复本，因为中央位置仅有一个复本；

（3）可以在服务器上一次性定义业务和安全规则，并对所有的用户平等执行；

（4）可以在数据库内通过使用约束、存储过程和触发器来强制执行规则。还可在服务器应用程序中执行规则，因为这些应用程序也是许多客户端访问的中央资源；

（5）关系数据库服务器只返回应用程序所需要的数据，优化了网络流量；

（6）最大程度地降低硬件的成本，由于数据不是存储在每个客户端上，客户端不必耗费磁盘空间来存储数据。客户端无需在本地增加管理数据的功能，同时，服务器不需将处理能力耗费在显示数据上；

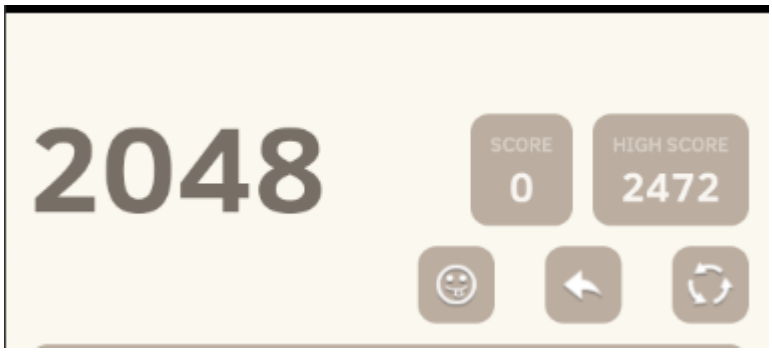
（7）可以配置服务器以优化检索数据所需的磁盘输入/输出容量，配置客户端以优化从服务器检索数据的格式；

（8）可以将服务器存储在一个相对安全的位置，并配备如不间断电源供应系统这样的设备，这比完全保护每个客户端更经济；

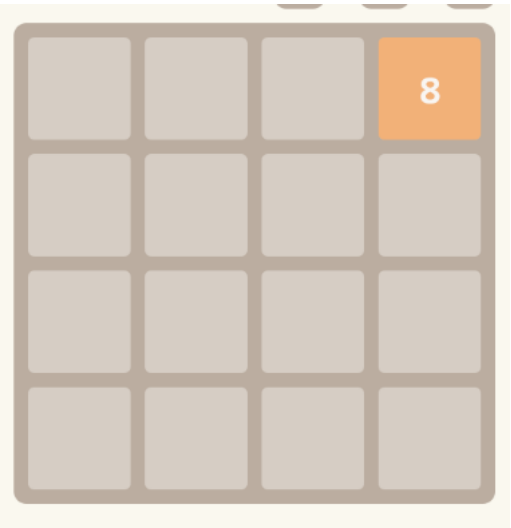
（9）维护任务（例如备份和恢复数据）得到简化，因为这些任务都可以集中在中央服务器上执行。

2.3 游戏界面设计

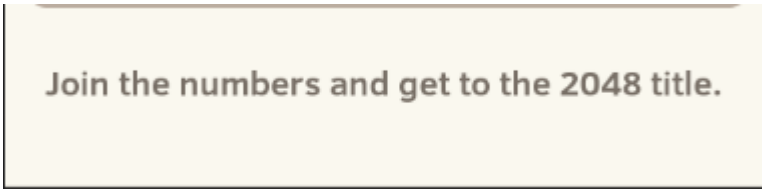
本游戏的主界面设计结构采用了上中下的结构，上部分为主功能菜单区，主要是显示游戏的积分系统，左侧游戏名字，右侧为分数与历史最高分数，右下侧部分为清空格子、返回上一步、初始化游戏的功能按钮。中间部分为游戏的主棋盘，用来显示方块与让玩家进行游戏。下部分添加了游戏说明。主要是为方便用户提供了一些简单方便的操作，界面设计如图所示：



界面（上）



界面（中）



界面（下）

2.4 项目代码

本节对项目代码进行汇总：

2.4.1 背景-background_rectangle

代码如下所示：

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#bbada0"/>
```

```
    <corners
        android:bottomRightRadius="10dp"
        android:bottomLeftRadius="10dp"
        android:topLeftRadius="10dp"
        android:topRightRadius="10dp"/>
</shape>
```

2.4.2 方块-cell_rectangle

（1）cell_rectangle.xml

```
    <shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#d6cdc4"/>

    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（2）cell_rectangle_2.xml

```
    <shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#eee4da"/>
    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
```

```
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（3）cell_rectangle_4.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#ede0c8"/>
    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（4）cell_rectangle_8.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle" android:padding = "10dp">
<solid android:color="#f2b179"/>
<corners
    android:bottomRightRadius="5dp"
    android:bottomLeftRadius="5dp"
    android:topLeftRadius="5dp"
    android:topRightRadius="5dp"/>
</shape>
```

（5）cell_rectangle_16.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle" android:padding = "10dp">
<solid android:color="#f59563"/>
<corners
    android:bottomRightRadius="5dp"
    android:bottomLeftRadius="5dp"
    android:topLeftRadius="5dp"
    android:topRightRadius="5dp"/>
</shape>
```

（6）cell_rectangle_32.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#f67c5f"/>
    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（7）cell_rectangle_64.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#f65e3b"/>
    <corners
```

```
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（8）cell_rectangle_128.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#edcf72"/>
    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（9）cell_rectangle_256.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#edcc61"/>

    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
```

```
        android:topRightRadius="5dp"/>
</shape>
```

（10）cell_rectangle_512.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#edc850"/>

    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（11）cell_rectangle_1024.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#edc53f"/>

    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（12）cell_rectangle_2048.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#edc22e"/>

    <corners
        android:bottomRightRadius="5dp"
        android:bottomLeftRadius="5dp"
        android:topLeftRadius="5dp"
        android:topRightRadius="5dp"/>
</shape>
```

（13）fade_rectangle.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
    <solid android:color="#d6cdc4"/>

    <corners
        android:bottomRightRadius="10dp"
        android:bottomLeftRadius="10dp"
        android:topLeftRadius="10dp"
        android:topRightRadius="10dp"/>
</shape>
```

（14）light_up_rectangle.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding = "10dp">
```

```
    <solid android:color="#edcf72"/>
    <corners
        android:bottomRightRadius="10dp"
        android:bottomLeftRadius="10dp"
        android:topLeftRadius="10dp"
        android:topRightRadius="10dp"/>
</shape>
```

2.4.3 Java

（1）AnimationCell

```
public class AnimationCell extends Cell {
    private int animationType;
    private long timeElapsed;
    private long animationTime;
    private long delayTime;
    public int[] extras;

    public AnimationCell (int x, int y, int animationType, long length,
long delay, int[] extras) {
        super(x, y);
        this.animationType = animationType;
        animationTime = length;
        delayTime = delay;
        this.extras = extras;
    }

    public int getAnimationType() {
```

```
        return animationType;
    }

    public void tick(long timeElapsed) {
        this.timeElapsed = this.timeElapsed + timeElapsed;
    }

    public boolean animationDone() {
        return animationTime + delayTime < timeElapsed;
    }

    public double getPercentageDone() {
        return Math.max(0, 1.0 * (timeElapsed - delayTime) /
animationTime);
    }

    public boolean isActive() {
        return (timeElapsed >= delayTime);
    }

}
```

（2）AnimationGrid

```
public class AnimationGrid {
   public ArrayList<AnimationCell>[][] field;
   int activeAnimations = 0;
   boolean oneMoreFrame = false;
   public ArrayList<AnimationCell> globalAnimation = new
```

```
ArrayList<AnimationCell>();

   @SuppressWarnings("unchecked")
   public AnimationGrid(int x, int y) {
      field = new ArrayList[x][y];

      for (int xx = 0; xx < x; xx++) {
         for (int yy = 0; yy < y; yy++) {
            field[xx][yy] = new ArrayList<AnimationCell>();
         }
      }
   }

   public void startAnimation(int x, int y, int animationType, long
length,
         long delay, int[] extras) {
      AnimationCell animationToAdd = new AnimationCell(x, y,
animationType,
            length, delay, extras);
      if (x == -1 && y == -1) {
         globalAnimation.add(animationToAdd);
      } else {
         field[x][y].add(animationToAdd);
      }
      activeAnimations = activeAnimations + 1;
   }

   public void tickAll(long timeElapsed) {
      ArrayList<AnimationCell> cancelledAnimations = new
```

```
ArrayList<AnimationCell>();
      for (AnimationCell animation : globalAnimation) {
         animation.tick(timeElapsed);
         if (animation.animationDone()) {
            cancelledAnimations.add(animation);
            activeAnimations = activeAnimations - 1;
         }
      }

      for (ArrayList<AnimationCell>[] array : field) {
         for (ArrayList<AnimationCell> list : array) {
            for (AnimationCell animation : list) {
               animation.tick(timeElapsed);
               if (animation.animationDone()) {
                  cancelledAnimations.add(animation);
                  activeAnimations = activeAnimations - 1;
               }
            }
         }
      }

      for (AnimationCell animation : cancelledAnimations) {
         cancelAnimation(animation);
      }
   }

   public boolean isAnimationActive() {
      if (activeAnimations != 0) {
         oneMoreFrame = true;
```

```
            return true;
        } else if (oneMoreFrame) {
            oneMoreFrame = false;
            return true;
        } else {
            return false;
        }
    }

    public ArrayList<AnimationCell> getAnimationCell(int x, int y) {
        return field[x][y];
    }

    public void cancelAnimations() {
        for (ArrayList<AnimationCell>[] array : field) {
            for (ArrayList<AnimationCell> list : array) {
                list.clear();
            }
        }
        globalAnimation.clear();
        activeAnimations = 0;
    }

    public void cancelAnimation(AnimationCell animation) {
        if (animation.getX() == -1 && animation.getY() == -1) {
            globalAnimation.remove(animation);
        } else {
            field[animation.getX()][animation.getY()].remove(animation);
        }
```

```
    }

}
```

（3）BuildConfig

```
public final class BuildConfig {
  public static final boolean DEBUG = Boolean.parseBoolean("true");
  public static final String APPLICATION_ID = "me.veryyoung.game2048";
  public static final String BUILD_TYPE = "debug";
  public static final String FLAVOR = "";
  public static final int VERSION_CODE = 1;
  public static final String VERSION_NAME = "1.0";
}
```

（4）Cell

```
public class Cell {
    private int x;
    private int y;

    public Cell(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return this.x;
    }
```

```
    public int getY() {
        return this.y;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }
}
```

（5）Grid

```
public class Grid {

   public Tile[][] field;
   public Tile[][] undoField;
   private Tile[][] bufferField;

   public Grid(int sizeX, int sizeY) {
      field = new Tile[sizeX][sizeY];
      undoField = new Tile[sizeX][sizeY];
      bufferField = new Tile[sizeX][sizeY];
      clearGrid();
      clearUndoGrid();
   }
```

```
public Cell randomAvailableCell() {
   ArrayList<Cell> availableCells = getAvailableCells();
   if (availableCells.size() >= 1) {
      return availableCells.get((int) Math.floor(Math.random()
            * availableCells.size()));
   }
   return null;
}

public ArrayList<Cell> getAvailableCells() {
   ArrayList<Cell> availableCells = new ArrayList<Cell>();
   for (int xx = 0; xx < field.length; xx++) {
      for (int yy = 0; yy < field[0].length; yy++) {
         if (field[xx][yy] == null) {
            availableCells.add(new Cell(xx, yy));
         }
      }
   }
   return availableCells;
}

public ArrayList<Cell> getNotAvailableCells() {
   ArrayList<Cell> notAvailableCells = new ArrayList<Cell>();
   for (int xx = 0; xx < field.length; xx++) {
      for (int yy = 0; yy < field[0].length; yy++) {
         if (field[xx][yy] != null) {
            notAvailableCells.add(new Cell(xx, yy));
         }
```

```
            }
        }
        return notAvailableCells;
    }

    public boolean isCellsAvailable() {
        return (getAvailableCells().size() >= 1);
    }

    public boolean isCellAvailable(Cell cell) {
        return !isCellOccupied(cell);
    }

    public boolean isCellOccupied(Cell cell) {
        return (getCellContent(cell) != null);
    }

    public Tile getCellContent(Cell cell) {
        if (cell != null && isCellWithinBounds(cell)) {
            return field[cell.getX()][cell.getY()];
        } else {
            return null;
        }
    }

    public Tile getCellContent(int x, int y) {
        if (isCellWithinBounds(x, y)) {
            return field[x][y];
        } else {
```

```
            return null;
        }
    }

    public boolean isCellWithinBounds(Cell cell) {
        return 0 <= cell.getX() && cell.getX() < field.length
                && 0 <= cell.getY() && cell.getY() < field[0].length;
    }

    public boolean isCellWithinBounds(int x, int y) {
        return 0 <= x && x < field.length && 0 <= y && y < field[0].length;
    }

    public void insertTile(Tile tile) {
        field[tile.getX()][tile.getY()] = tile;
    }

    public void removeTile(Tile tile) {
        field[tile.getX()][tile.getY()] = null;
    }

    public void saveTiles() {
        for (int xx = 0; xx < bufferField.length; xx++) {
            for (int yy = 0; yy < bufferField[0].length; yy++) {
                if (bufferField[xx][yy] == null) {
                    undoField[xx][yy] = null;
                } else {
                    undoField[xx][yy] = new Tile(xx, yy,
                            bufferField[xx][yy].getValue());
```

```
            }
        }
    }
}

public void prepareSaveTiles() {
    for (int xx = 0; xx < field.length; xx++) {
        for (int yy = 0; yy < field[0].length; yy++) {
            if (field[xx][yy] == null) {
                bufferField[xx][yy] = null;
            } else {
                bufferField[xx][yy] = new Tile(xx, yy,
                        field[xx][yy].getValue());
            }
        }
    }
}

public void revertTiles() {
    for (int xx = 0; xx < undoField.length; xx++) {
        for (int yy = 0; yy < undoField[0].length; yy++) {
            if (undoField[xx][yy] == null) {
                field[xx][yy] = null;
            } else {
                field[xx][yy] = new Tile(xx, yy,
                        undoField[xx][yy].getValue());
            }
        }
    }
```

```
    }

    public void clearGrid() {
        for (int xx = 0; xx < field.length; xx++) {
            for (int yy = 0; yy < field[0].length; yy++) {
                field[xx][yy] = null;
            }
        }
    }

    public void clearUndoGrid() {
        for (int xx = 0; xx < field.length; xx++) {
            for (int yy = 0; yy < field[0].length; yy++) {
                undoField[xx][yy] = null;
            }
        }
    }
}
```

（6）InputListener

```
public class InputListener implements View.OnTouchListener {

    private static final int SWIPE_MIN_DISTANCE = 0;
    private static final int SWIPE_THRESHOLD_VELOCITY = 25;
    private static final int MOVE_THRESHOLD = 250;
    private static final int RESET_STARTING = 10;

    private float x;
```

```
private float y;
private float lastdx;
private float lastdy;
private float previousX;
private float previousY;
private float startingX;
private float startingY;
private int previousDirection = 1;
private int veryLastDirection = 1;
private boolean hasMoved = false;

MainView mView;

public InputListener(MainView view) {
   super();
   this.mView = view;
}

public boolean onTouch(View view, MotionEvent event) {
   switch (event.getAction()) {

   case MotionEvent.ACTION_DOWN:
      x = event.getX();
      y = event.getY();
      startingX = x;
      startingY = y;
      previousX = x;
      previousY = y;
      lastdx = 0;
```

```
            lastdy = 0;
            hasMoved = false;
            return true;
        case MotionEvent.ACTION_MOVE:
            x = event.getX();
            y = event.getY();
            if (mView.game.isActive()) {
                float dx = x - previousX;
                if (Math.abs(lastdx + dx) < Math.abs(lastdx) + Math.abs(dx)
                        && Math.abs(dx) > RESET_STARTING
                        && Math.abs(x - startingX) > SWIPE_MIN_DISTANCE) {
                    startingX = x;
                    startingY = y;
                    lastdx = dx;
                    previousDirection = veryLastDirection;
                }
                if (lastdx == 0) {
                    lastdx = dx;
                }
                float dy = y - previousY;
                if (Math.abs(lastdy + dy) < Math.abs(lastdy) + Math.abs(dy)
                        && Math.abs(dy) > RESET_STARTING
                        && Math.abs(y - startingY) > SWIPE_MIN_DISTANCE) {
                    startingX = x;
                    startingY = y;
                    lastdy = dy;
                    previousDirection = veryLastDirection;
                }
                if (lastdy == 0) {
```

```
                lastdy = dy;
            }
            if (pathMoved() > SWIPE_MIN_DISTANCE * SWIPE_MIN_DISTANCE)
{
                boolean moved = false;
                if (((dy >= SWIPE_THRESHOLD_VELOCITY &&
previousDirection == 1) || y
                        - startingY >= MOVE_THRESHOLD)
                        && previousDirection % 2 != 0) {
                    moved = true;
                    previousDirection = previousDirection * 2;
                    veryLastDirection = 2;
                    mView.game.move(2);
                } else if (((dy <= -SWIPE_THRESHOLD_VELOCITY &&
previousDirection == 1) || y
                        - startingY <= -MOVE_THRESHOLD)
                        && previousDirection % 3 != 0) {
                    moved = true;
                    previousDirection = previousDirection * 3;
                    veryLastDirection = 3;
                    mView.game.move(0);
                } else if (((dx >= SWIPE_THRESHOLD_VELOCITY &&
previousDirection == 1) || x
                        - startingX >= MOVE_THRESHOLD)
                        && previousDirection % 5 != 0) {
                    moved = true;
                    previousDirection = previousDirection * 5;
                    veryLastDirection = 5;
                    mView.game.move(1);
```

```
                } else if (((dx <= -SWIPE_THRESHOLD_VELOCITY &&
previousDirection == 1) || x
                        - startingX <= -MOVE_THRESHOLD)
                        && previousDirection % 7 != 0) {
                    moved = true;
                    previousDirection = previousDirection * 7;
                    veryLastDirection = 7;
                    mView.game.move(3);
                }
                if (moved) {
                    hasMoved = true;
                    startingX = x;
                    startingY = y;
                }
            }
        }
        previousX = x;
        previousY = y;
        return true;
    case MotionEvent.ACTION_UP:
        x = event.getX();
        y = event.getY();
        previousDirection = 1;
        veryLastDirection = 1;
        // "Menu" inputs
        if (!hasMoved) {
            if (iconPressed(mView.sXNewGame, mView.sYIcons)) {
                mView.game.newGame();
            } else if (iconPressed(mView.sXUndo, mView.sYIcons)) {
```

```
                mView.game.revertUndoState();
            } else if (iconPressed(mView.sXCheat, mView.sYIcons)) {
                mView.game.cheat();
            } else if (isTap(2)
                    && inRange(mView.startingX, x, mView.endingX)
                    && inRange(mView.startingY, x, mView.endingY)
                    && mView.continueButtonEnabled) {
                mView.game.setEndlessMode();
            }
        }
    }
    return true;
}

private float pathMoved() {
    return (x - startingX) * (x - startingX) + (y - startingY)
            * (y - startingY);
}

private boolean iconPressed(int sx, int sy) {
    return isTap(1) && inRange(sx, x, sx + mView.iconSize)
            && inRange(sy, y, sy + mView.iconSize);
}

private boolean inRange(float starting, float check, float ending) {
    return (starting <= check && check <= ending);
}

private boolean isTap(int factor) {
```

```
        return pathMoved() <= mView.iconSize * factor;
    }

}
```

（7）MainActivity

```
MainView view;
    public static final String WIDTH = "width";
    public static final String HEIGHT = "height";
    public static final String SCORE = "score";
    public static final String HIGH_SCORE = "high score temp";
    public static final String UNDO_SCORE = "undo score";
    public static final String CAN_UNDO = "can undo";
    public static final String UNDO_GRID = "undo";
    public static final String GAME_STATE = "game state";
    public static final String UNDO_GAME_STATE = "undo game state";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        view = new MainView(getBaseContext());

        SharedPreferences settings = PreferenceManager
                .getDefaultSharedPreferences(this);
        view.hasSaveState = settings.getBoolean("save_state", false);
```

```
    if (savedInstanceState != null) {
        if (savedInstanceState.getBoolean("hasState")) {
            load();
        }
    }
    setContentView(view);
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_MENU) {
        // Do nothing
        return true;
    } else if (keyCode == KeyEvent.KEYCODE_DPAD_DOWN) {
        view.game.move(2);
        return true;
    } else if (keyCode == KeyEvent.KEYCODE_DPAD_UP) {
        view.game.move(0);
        return true;
    } else if (keyCode == KeyEvent.KEYCODE_DPAD_LEFT) {
        view.game.move(3);
        return true;
    } else if (keyCode == KeyEvent.KEYCODE_DPAD_RIGHT) {
        view.game.move(1);
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
   savedInstanceState.putBoolean("hasState", true);
   save();
}

protected void onPause() {
   super.onPause();
   save();
}

private void save() {
   SharedPreferences settings = PreferenceManager
         .getDefaultSharedPreferences(this);
   SharedPreferences.Editor editor = settings.edit();
   Tile[][] field = view.game.grid.field;
   Tile[][] undoField = view.game.grid.undoField;
   editor.putInt(WIDTH, field.length);
   editor.putInt(HEIGHT, field.length);
   for (int xx = 0; xx < field.length; xx++) {
      for (int yy = 0; yy < field[0].length; yy++) {
         if (field[xx][yy] != null) {
            editor.putInt(xx + " " + yy, field[xx][yy].getValue());
         } else {
            editor.putInt(xx + " " + yy, 0);
         }

         if (undoField[xx][yy] != null) {
            editor.putInt(UNDO_GRID + xx + " " + yy,
```

```
                    undoField[xx][yy].getValue());
            } else {
                editor.putInt(UNDO_GRID + xx + " " + yy, 0);
            }
        }
    }
    editor.putLong(SCORE, view.game.score);
    editor.putLong(HIGH_SCORE, view.game.highScore);
    editor.putLong(UNDO_SCORE, view.game.lastScore);
    editor.putBoolean(CAN_UNDO, view.game.canUndo);
    editor.putInt(GAME_STATE, view.game.gameState);
    editor.putInt(UNDO_GAME_STATE, view.game.lastGameState);
    editor.commit();
}

protected void onResume() {
    super.onResume();
    load();
}

private void load() {
    // Stopping all animations
    view.game.aGrid.cancelAnimations();

    SharedPreferences settings = PreferenceManager
            .getDefaultSharedPreferences(this);
    for (int xx = 0; xx < view.game.grid.field.length; xx++) {
        for (int yy = 0; yy < view.game.grid.field[0].length; yy++) {
            int value = settings.getInt(xx + " " + yy, -1);
```

```
                if (value > 0) {
                    view.game.grid.field[xx][yy] = new Tile(xx, yy, value);
                } else if (value == 0) {
                    view.game.grid.field[xx][yy] = null;
                }

                int undoValue = settings.getInt(UNDO_GRID + xx + " " + yy,
-1);
                if (undoValue > 0) {
                    view.game.grid.undoField[xx][yy] = new Tile(xx, yy,
                            undoValue);
                } else if (value == 0) {
                    view.game.grid.undoField[xx][yy] = null;
                }
            }
        }

        view.game.score = settings.getLong(SCORE, view.game.score);
        view.game.highScore = settings.getLong(HIGH_SCORE,
view.game.highScore);
        view.game.lastScore = settings.getLong(UNDO_SCORE,
view.game.lastScore);
        view.game.canUndo = settings.getBoolean(CAN_UNDO,
view.game.canUndo);
        view.game.gameState = settings.getInt(GAME_STATE,
view.game.gameState);
        view.game.lastGameState = settings.getInt(UNDO_GAME_STATE,
                view.game.lastGameState);}
}
```

第三章 项目总结

在 Android 的学习过程中，遇到许许多多的问题，环境配置、系统环境等，但通过网上、书籍等渠道，慢慢学习了许多新的知识，也解决了诸多问题，最后做出项目。