课程设计报告

**题目：基于android的2048游戏**

**学 院**

**专 业**

**班 级**

**学 号**

**学 生 姓 名**

**联 系 方 式**

**指 导 教 师 职称：**

**设计时间：**

**目录**

[1需求分析 3](#_Toc1573)

[1. 1背景与意义 3](#_Toc27173)

[1.1.1手机应用市场发展现状 3](#_Toc32053)

[1.1.2国内外现状 4](#_Toc31121)

[1.1.3此游戏的意义 4](#_Toc31174)

[1.2.系统需求分析 4](#_Toc6010)

[1.2.1系统功能需求分析 5](#_Toc10454)

[1.2.2游戏的基本规则 6](#_Toc9481)

[2 系统分析与设计 6](#_Toc4233)

[2.1. 系统流程设计 6](#_Toc6893)

[2.2系统模块设计 7](#_Toc3463)

[2.2.1 菜单设计 7](#_Toc22295)

[2.2.2界面设计 7](#_Toc21051)

[2.2.3算法设计 8](#_Toc25984)

[2.3本章小结 9](#_Toc14610)

[3 系统实现 10](#_Toc1005)

[3.1开始界面的实现 10](#_Toc18840)

[3.2游戏界面的实现 10](#_Toc24849)

[3.3游戏滑动卡片移动的实现 11](#_Toc4012)

[3.4 重新开始游戏功能的实现 16](#_Toc22906)

[3.5 退出游戏功能的实现 17](#_Toc18734)

[3.6 游戏帮助功能的实现 18](#_Toc4474)

[3.7 本章小结 19](#_Toc32481)

[4 测试 19](#_Toc8379)

[4.1 游戏流程测试 19](#_Toc1578)

[4.2游戏模式 20](#_Toc6483)

[4.3 本章小结 20](#_Toc26665)

[5 总结 20](#_Toc10915)

**1需求分析**

1. **1背景与意义**

**1.1.1手机应用市场发展现状**

随着4G越来越普及以及手机应用的日益丰富还有智能水平的不断提高，从便携性和随身性这两方面来考虑，电脑所带来的体验已经不能跟手机相提并论了，它已经完美的超越了电脑。

现如今Android、苹果等各智能手机已经基本占领整个手机市场，从而使得更多应用的出现，而手机游戏应用在其中占领主要位置。

随着Android智能手机的普及以及游戏种类的多元化，使得Android手机游戏用户规模保持着稳步增长之势。

**1.1.2国内外现状**

目前国内外的Android开发还是主要以应用开发为主，主要分成三类：企业应用、通用应用及游戏应用。企业应用的开发主要是一些大公司为了自己的品牌而开发的；通用应用主要是一些创业型公司或者独立开发者为了自己盈利开发的应用；游戏应用目前和通用应用相同。

2048小游戏是一款最近风靡全球的手机游戏，简单的游戏模式和趣味的玩法，几乎游戏下载排行榜的前20名都可以看到“它的身影”。

**1.1.3此游戏的意义**

现如今，手机游戏已在我们的生活中占据一席之地，并在一步步的壮大。可以说，随着它的迅猛发展，现今的手机游戏已经不单单是一种缓解压力的工具，而是形成了一种文化现象。随着游戏软件在市场的一步步壮大，与其有关的文化也随之传播。

2048游戏的制作属于电子游戏中益智类小游戏，它做到了娱乐性、趣味性、教育性相统一。益智类的游戏即是需要去开动大脑思考从而获得游戏的胜利。简单的益智类游戏可以使玩家在娱乐中不断地开发大脑。这样一来就实现了在娱乐中学习。

**1.2.系统需求分析**

**1.2.1系统功能需求分析**

系统主要实现以下的几个功能：呈现游戏界面、重新开始游戏、当前分数和最高分数、游戏帮助等功能。

重新开始游戏是当玩家无法满足当前进度时点击此按钮就会重新开始游戏，如果玩家处于不同关卡时提示重新开始游戏还是停留在此关卡。游戏帮助是当新手玩此游戏时无法知道游戏玩法时给予相应的提示。呈现游戏界面是游戏开始时主界面在游戏区域会生成4x4的矩阵同时在矩阵里面随机生成两个2或4的游戏卡片。当前分数和最高分数是显示此局玩家所获得的分数和历史上最高的分数，如果当前的分数超过最高的分数，那么最高分数显示当前的分数。如图所示：

2048游戏

开始界面

主界面

游戏帮助

重新开始

游戏关卡

游戏区域

最高分数

当前分数

选择关卡

返回主界面

**1.2.2游戏的基本规则**

在开始游戏后玩家通过滑动屏幕来操控卡片的移动方向，当卡片滑动中如果有两张卡片相同且他们中间也没有其他卡片时，在滑动的过程中这两张卡片会合并，显示为这两张卡片之和。在滑动过程中有三张卡片相同时只会合并向滑动方向两张卡片。在滑动中如果有两张卡片一样同时又有一张卡片的值跟这两张卡片相加的值时，滑动只会使那两张相同的卡片合并而不会接着让合并后的卡片和另一张卡片合并。

**2 系统分析与设计**

**2.1. 系统流程设计**

游戏进入开始页面，能够进入游戏的主界面并开始普通开局，从主界面能够重新开始游戏、查看帮助和进入关卡选择界面。当玩家点击重新开始按钮会弹出相应的对话框让玩家选择，如果玩家选择“是”时则重新开始游戏，如果选择“否”则返回游戏界面不做任何处理。在开始界面按返回按钮时则会退出游戏。游戏流程如图所示：

开始界面

随机产生两张卡片

滑屏控制卡片的移动

N

移动方向是否有卡片

卡片移到底部

Y

Y

原位置卡片清除

数字是否相等

N

N

Y

两卡片相加并消除起始位置的卡片，当前卡片值翻倍

上一个卡片数字是否相等

N

Y

卡片是否可以移动

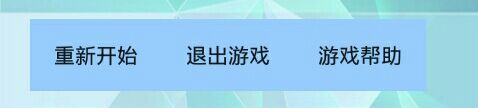
结束

**2.2系统模块设计**

从总体出发，将该系统划分为三大模块：“菜单设计”、“界面设计”和“算法设计”。

**2.2.1 菜单设计**

菜单的实现是在游戏界面，可进一步划分为三个模块，分别是：“重新开始”、“退出游戏”、“游戏帮助”，如图所示：



**2.2.2界面设计**

如图所示：图一为开始界面，图二为游戏界面

**2.2.3算法设计**

1. 当有两张卡片相同时，向它们可以碰撞的方向滑屏，卡片会移动到最底边并生成其两倍数字的卡片，并且生成一个“2”或者“4”的卡片。如图所示：

1. 当有两张卡片相同时，且在它们相同的方向有张跟他们之和的卡片，向它们可以碰撞的方向滑屏，相同的卡片会移动到无法移动的位置并生成其两倍数字的卡片，但合成的方向不会跟那两张数字的卡片合并，并且生成一个“2”或者“4”的卡片。如图所示：

1. 当界面上没有空位并且两两相邻的卡片不相同时游戏结束。如图所示：



**2.3本章小结**

本章主要对游戏所实现的功能进行需求分析，分析了图形的特点和实现的可行性。对系统的性能进行了详细的分析。对系统的流程，系统所需的图形文件，系统的总体架构和系统用例进行了设计。通过本章的分析、设计能更加具体的了解系统功能，对系统所要实现的功能和图形文件有了更深的认识。为下一章系统功能的具体实现提供了可靠的参考依据。

**3 系统实现**

**3.1开始界面的实现**

游戏的主界面是按钮，只是为了实现界面的跳转，当玩家点击开始游戏就会调用loginActivity.java，此函数让页面跳转到游戏界面开始游戏，如图所示：

**public class** loginActivity **extends** MainActivity {  
 **protected void** onCreate(Bundle savedInstanceState){  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***login***);  
 }  
 }



**3.2游戏界面的实现**

游戏界面主要是在activity\_main.xml中当前分数、最高分数、重新开始按钮、退出游戏按钮、游戏帮助按钮、帮助按钮，当跳转到游戏界面时就会调用并执行MainActivity.java函数来展示游戏界面，如图所示：

**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);

}

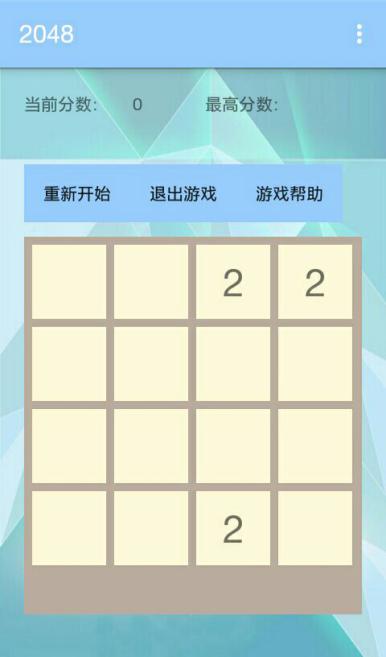
****

**3.3游戏滑动卡片移动的实现**

当玩家滑动屏幕时，主要通过GameView函数来监听玩家手指滑动的位置，先通过获取开始坐标和结束坐标，然后通过比较结束坐标跟开始坐标的差值来判断玩家是怎么滑动屏幕的。判断出玩家的滑动轨迹后，通过调用swipeLeft、swipeRight、swipeUp、swipeDown方法来实现卡片的移动，如图所示：

**private void** initGameView(){  
 setColumnCount(4);  
 *//将面板设置成4列* setBackgroundColor(0xffbbada0);  
 System.***out***.println(**"initGameView"**);  
  
 setOnTouchListener(**new** View.OnTouchListener() {  
 */\*  
 \* startX:手机刚开始在屏幕上的X坐标  
 \* startY:手机刚开始在屏幕上的Y坐标  
 \* offsetX,offsetY,分别是手指在屏幕上的X,Y上的偏移量  
 \*/* **private float startX**,**startY**,**offsetX**,**offsetY**;  
 @Override  
 **public boolean** onTouch(View v, MotionEvent event) {  
 **switch** (event.getAction()){  
 **case** MotionEvent.***ACTION\_DOWN***:  
 **startX** = event.getX();  
 **startY** = event.getY();  
 **break**;  
 **case** MotionEvent.***ACTION\_UP***:  
 **offsetX** = event.getX() - **startX**;  
 **offsetY** = event.getY() - **startY**;  
  
 **if**(Math.*abs*(**offsetX**) > Math.*abs*(**offsetY**)){  
 **if**(**offsetX** < -5){  
 swipeLeft();  
 System.***out***.println(**"Left"**);  
  
 }**else if**(**offsetX** > 5){  
 swipeRight();  
 System.***out***.println(**"Right"**);  
 }  
  
 } **else**{  
 **if**(**offsetY** < -5){  
 swipeUp();  
 System.***out***.println(**"Up"**);  
 }**else if**(**offsetY** > 5){  
 swipeDown();  
 System.***out***.println(**"Down"**);  
 }  
 }  
 **break**;  
 }  
 **return true**;  
 }  
 });  
}

*//向左滑动***public void** swipeLeft(){  
 **boolean** meger = **false**;  
 **for** (**int** y = 0; y < 4; y++) {  
 **for** (**int** x = 0; x < 4; x++) {  
 **for** (**int** x1 = x+1; x1 < 4; x1++) {  
 **if**(**cardsMap**[x1][y].getNum()>0){  
 **if**(**cardsMap**[x][y].getNum()<=0){  
 */\*  
 \* 将下标为（x，y）所在位置的卡片上的数字  
 \* 设置为，坐标为(x1,y)所在位置的卡片上的值；  
 \* 第二步，将坐标（x1，y）所在位置的卡片上的数字  
 \* 设置为0  
 \* （即：变成空卡片）  
 \*/* **cardsMap**[x][y].setNum(  
 **cardsMap**[x1][y].getNum());  
 **cardsMap**[x1][y].setNum(0);  
 x--;  
 meger =**true**;  
 **break**;  
 }**else if**(**cardsMap**[x][y].equals(**cardsMap**[x1][y])){  
 **cardsMap**[x][y].setNum(**cardsMap**[x][y].getNum()\*2);  
 **cardsMap**[x1][y].setNum(0);  
 MainActivity.*getMainActivity*().  
 addScore(**cardsMap**[x][y].getNum());  
 meger = **true**;  
 }  
 **break**;  
 }  
 }  
 }  
 }  
 **if**(meger){  
 addRandomNum();  
 checkComplete();  
 }  
}  
*//向右滑动***public void** swipeRight(){  
 **boolean** meger = **false**;  
 **for** (**int** y = 0; y < 4; y++) {  
 **for** (**int** x = 3; x >=0; x--) {  
 **for** (**int** x1 = x-1; x1 >= 0; x1--) {  
 **if**(**cardsMap**[x1][y].getNum()>0){  
 **if**(**cardsMap**[x][y].getNum()<=0){  
 */\*  
 \* 将下标为（x，y）所在位置的卡片上的数字  
 \* 设置为，坐标为(x1,y)所在位置的卡片上的值；  
 \* 第二步，将坐标（x1，y）所在位置的卡片上的数字  
 \* 设置为0  
 \* （即：变成空卡片）  
 \*/* **cardsMap**[x][y].setNum(  
 **cardsMap**[x1][y].getNum());  
 **cardsMap**[x1][y].setNum(0);  
 x++;  
 meger =**true**;  
 **break**;  
 }**else if**(**cardsMap**[x][y].equals(**cardsMap**[x1][y])){  
 **cardsMap**[x][y].setNum(**cardsMap**[x][y].getNum()\*2);  
 **cardsMap**[x1][y].setNum(0);  
 MainActivity.*getMainActivity*().  
 addScore(**cardsMap**[x][y].getNum());  
 meger =**true**;  
 }**break**;  
 }  
 }  
 }  
 }  
 **if**(meger){  
 addRandomNum();  
 checkComplete();  
 }  
}  
*//向上滑动***public void** swipeUp(){  
 **boolean** meger = **false**;  
 **for** (**int** x= 0; x< 4; x++) {  
 **for** (**int** y = 0; y < 4; y++) {  
 **for** (**int** y1 = y+1; y1 < 4; y1++) {  
 **if**(**cardsMap**[x][y1].getNum()>0){  
 **if**(**cardsMap**[x][y].getNum()<=0){  
 */\*  
 \* 将下标为（x，y）所在位置的卡片上的数字  
 \* 设置为，坐标为(x1,y)所在位置的卡片上的值；  
 \* 第二步，将坐标（x1，y）所在位置的卡片上的数字  
 \* 设置为0  
 \* （即：变成空卡片）  
 \*/* **cardsMap**[x][y].setNum(  
 **cardsMap**[x][y1].getNum());  
 **cardsMap**[x][y1].setNum(0);  
 y--;  
 meger =**true**;  
 **break**;  
 }**else if**(**cardsMap**[x][y].equals(**cardsMap**[x][y1])){  
 **cardsMap**[x][y].setNum(**cardsMap**[x][y].getNum()\*2);  
 **cardsMap**[x][y1].setNum(0);  
 MainActivity.*getMainActivity*().  
 addScore(**cardsMap**[x][y].getNum());  
 meger =**true**;  
 }  
 **break**;  
 }  
 }  
 }  
 }  
 **if**(meger){  
 addRandomNum();  
 checkComplete();  
 }  
}  
*//向下滑动***public void** swipeDown(){  
 **boolean** meger = **false**;  
 **for** (**int** x = 0; x< 4; x++) {  
 **for** (**int** y = 3; y>= 0;y--) {  
 **for** (**int** y1 = y-1; y1 >=0; y1--) {  
 **if**(**cardsMap**[x][y1].getNum()>0){  
 **if**(**cardsMap**[x][y].getNum()<=0){  
 */\*  
 \* 将下标为（x，y）所在位置的卡片上的数字  
 \* 设置为，坐标为(x1,y)所在位置的卡片上的值；  
 \* 第二步，将坐标（x1，y）所在位置的卡片上的数字  
 \* 设置为0  
 \* （即：变成空卡片）  
 \*/* **cardsMap**[x][y].setNum(  
 **cardsMap**[x][y1].getNum());  
 **cardsMap**[x][y1].setNum(0);  
 y++;  
 meger =**true**;  
 **break**;  
 }**else if**(**cardsMap**[x][y].equals(**cardsMap**[x][y1])){  
 **cardsMap**[x][y].setNum(**cardsMap**[x][y].getNum()\*2);  
 **cardsMap**[x][y1].setNum(0);  
 MainActivity.*getMainActivity*().  
 addScore(**cardsMap**[x][y].getNum());  
 meger =**true**;  
 }  
 **break**;  
 }  
 }  
 }  
 }  
 **if**(meger){  
 addRandomNum();  
 checkComplete();  
 }  
}

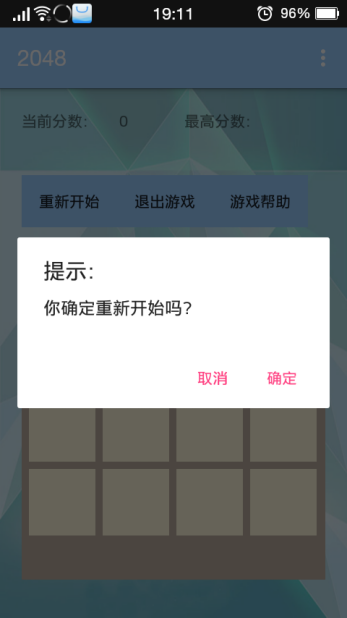
 

**3.4 重新开始游戏功能的实现**

当玩家点击游戏界面的重新开始游戏时，会弹出给玩家选择的对话框，让玩家选择“是”时游戏会重新开始，如图所示：

**public void** onClick(View view){  
 AlertDialog.Builder dialog3 = **new** AlertDialog.Builder(**this**);  
dialog3.setTitle(**"提示："**);  
dialog3.setMessage(**"你确定重新开始吗？"**);  
dialog3.setPositiveButton(**"确定"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
  
 }  
});  
dialog3.setNegativeButton(**"取消"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
  
 }  
});  
dialog3.show();

}

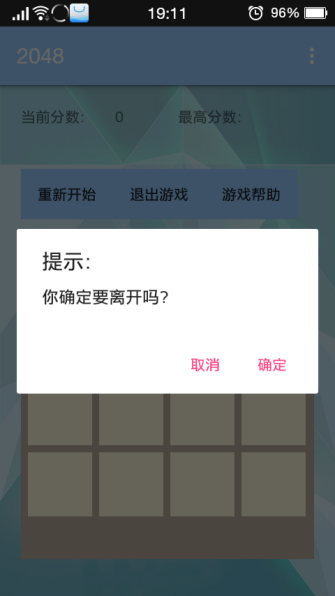


**3.5 退出游戏功能的实现**

当玩家中途有事想退出游戏时会弹出给玩家选择的对话框，让玩家选择“是”时游戏会退出，如图所示：

public void onClick(View view){  
 AlertDialog.Builder dialog = new AlertDialog.Builder(this);  
dialog.setTitle("提示：");  
dialog.setMessage("你确定要离开吗？");  
dialog.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 System.*exit*(0);  
 }  
});  
dialog.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
});  
dialog.show();

}



**3.6 游戏帮助功能的实现**

当新玩家进入到游戏且不知道此游戏玩法时，玩家可以点击游戏帮助按钮来了解游戏玩法，点击按钮时游戏会弹出对话框显示游戏玩法，如图所示：

public void onClick(View view){  
 AlertDialog.Builder dialog2 = new AlertDialog.Builder(this);  
dialog2.setTitle("hey,guy！");  
dialog2.setMessage("这么简单的游戏你确定需要帮助？");  
dialog2.setNegativeButton("继续玩~", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
});  
dialog2.show();

}



**3.7 本章小结**

本章主要阐述本游戏相关功能的实现，详细的讲述了主界面的实现和各按钮功能的实现。

1. **测试**

本章主要对系统的功能进行测试，此次测试只是进行简单的调试，来确定游戏的各项功能是否能够正常运行。

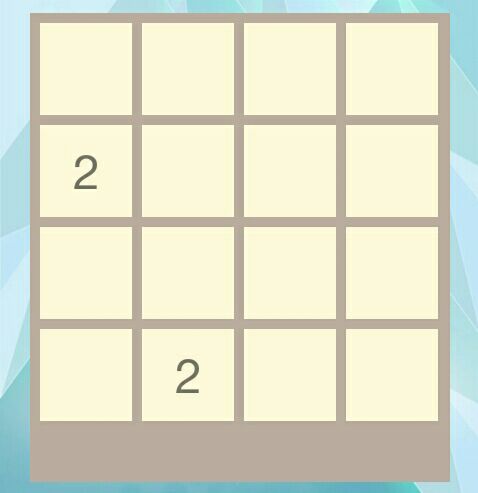
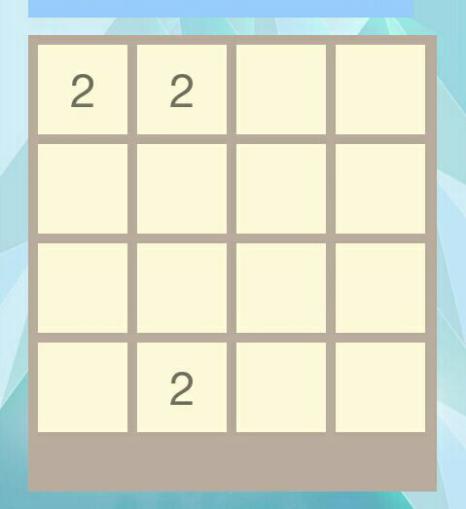
**4.1 游戏流程测试**

该测试主要验证游戏能否实现场景的切换，当界面在开始界面时只显示按钮画面，当玩家点击此界面的开始按钮时跳转到游戏界面，如图所示：

**4.2游戏模式**

该测试主要是测试游戏能否正常运行，当玩家滑动屏幕时能否正常的移动和当卡片相同时是否能够相加，还有就是测试游戏是否能正常结束。

**4.3 本章小结**

本章是对游戏系统进行简单的测试，通过测试可以看出此游戏可以正常的工作，同时一些功能也能够实现。

1. **总结**

本次课程设计的内容大部分都是参照课堂所讲以及一些网站给出的各种建议写的，在写的过程中遇到了很多问题，学到了很多东西。期间大概是因为基础不够好，只是找错误就花了很长时间。不过正因为这些错误，才能学到更多的知识，才能把知识点掌握的更牢靠，对于一些没有实现的功能，之后我一定会多花费些时间研究出来。我的课程设计优化的空间还相当大，希望老师能给出指导！