**课 程 设 计 报 告**

**题 目**  基于Java的Smove游戏设计

**姓 名**  柳宏森

**2015年12月**

目录

[绪论 1](#_Toc421792819)

[1.背景 2](#_Toc421792820)

[1.1 应用背景 2](#_Toc421792821)

[1.2 技术背景 2](#_Toc421792822)

[1.2.1双缓冲机制 2](#_Toc421792823)

[1.2.2游戏运行框架 4](#_Toc421792824)

[1.3开发环境 4](#_Toc421792825)

[2需求分析 5](#_Toc421792826)

[2.1 需求概述 5](#_Toc421792827)

[2.2 功能描述 5](#_Toc421792828)

[3整体设计 6](#_Toc421792829)

[3.1类图视图（见图1） 6](#_Toc421792830)

[3.2类模块设计 6](#_Toc421792831)

[4详细设计 6](#_Toc421792832)

[4.1函数实现部分 6](#_Toc421792833)

[4.2实现中的难题 8](#_Toc421792834)

[5编码测试 8](#_Toc421792835)

[6系统评估 10](#_Toc421792836)

[6.1 收获心得 10](#_Toc421792837)

[6.2 改进之处 10](#_Toc421792838)

# 绪论

在当今互联网高速发展的时代，以互联网为基础所带来的一系列网络产品与商品模式正在不断冲击着传统的经销模式。其中的电子商务与网游作为主力军，更是深入到消费者的心中。我所主要关注的是网游方面。

所谓的网游，英文名称为Online Game，又称 “在线游戏”，简称“网游”。指以互联网为传输媒介，以游戏运营商服务器和用户计算机为处理终端，以游戏客户端软件为信息交互窗口的旨在实现娱乐、休闲、交流和取得虚拟成就的具有可持续性的个体性多人在线游戏。

网络的发展，又随之带来了一股移动端的热潮，以移动设备接入网络，达到“即接即用，随时随地上网”的效果，吸引着无数电子厂商在这块领域崭露头角，也激发了消费者的极大购买率。

在移动端，“手游”的发展更是迅猛。随着智能手机的迅速普及，与3G网络的快速发展，移动互联网的相关业务，已逐渐成为国内各厂商关注的焦点，中国手机游戏市场目前仍然以单机版游戏为主，网游版产品所占比重较低。从整体行业来看，中国手机游戏行业处于快速发展期，近两年来，在电信运营商和手机游戏开发商的联合推动下，中国的手机游戏行业，取得了较快的发展。中国与日本、韩国、新加坡、马来西亚、台湾等国家和地区在3G业务上深入合作，使中国乃至亚洲地区逐渐发展成为全球范围内潜力最大的移动互联网市场，截至2010年年底，中国手机网民规模达3.03亿。预计在未来一两年内，手机上的游戏用户数量将超过传统PC游戏用户。

在苹果手机中，经常可以看到许多小游戏可以下载体验，这些游戏都有着一些共同的特点：简单、易懂、易上手、有趣味性。这些游戏大都界面朴实简单又不失大方，玩一遍就能轻松地掌握规则，并且有比较大的趣味性，使玩家爱不释手。比如曾经风行一时的 FlappyBird、

别踩白块、SMOVE等等。

总的来说，游戏方面的发展是迅速的，而且消费者市场也是有增无减，未来的游戏会朝着更加智能化、物理化的方向发展。

# 1.背景

## 1.1 应用背景

全球在使用的移动电话已经超过10亿部，而且这个数字每天都在不断增加。在除美国之外的各个发达国家，手机用户都比计算机用户多。手机游戏潜在的市场比其他任何平台，比如PlayStation和GameBoy都要大。

手机游戏作为移动互联网的杀手级应用，在全球范围内流行，预计在2012 全球手机游戏市场规模达到70 亿美元。日本手机游戏市场经历了多年的稳定增长，市场规模已经超过了850 亿日元。手机游戏作为美国手机用户最喜爱的移动应用之一，在IPhone 和Android 等智能终端推出之后发展迅速。2009 年，中国手机游戏总体市场规模18 亿元，同比增长38.5%；手机网游市场规模3.2 亿元，同比增长88.2%。从2007 年到2012 年，中国手机游戏市场规模复合增长率为45.4%，预计未来3 年复合增长率超过50%，2013 年市场规模超过50 亿元。手机游戏活跃用户在2009 年超过1100 万人，有望在2012 年达到3200 万

## 1.2 技术背景

### 1.2.1双缓冲机制

双缓冲即在内存中创建一个与屏幕绘图区域一致的对象，先将图形绘制到内存中的这个对象上，再一次性将这个对象上的图形拷贝到屏幕上，这样能大大加快绘图的速度。当数据量很大时，绘图可能需要几秒钟甚至更长的时间，而且有时还会出现闪烁现象，为了解决这些问题，可采用双缓冲技术来绘图。

双缓冲实现过程如下：

1、在内存中创建与画布一致的缓冲区

2、在缓冲区画图

3、将缓冲区位图拷贝到当前画布上

4、释放内存缓冲区

在JAVA中，双缓冲的实现，要和 repaint()，update(Graphics g)，paint(Graphics g)这三个函数打交道，其中在你每次绘图的时候，首先调用repaint()，这个函数会激发系统去调用update函数，在这个函数中有一条语句叫做的clearRect()的方法，这个方法会使当前绘制区域失效，并用背景色填充，正是由于这条语句导致了屏幕的双缓冲，我们可以重写（override）这个方法，在这个里面直接调用paint()，并且在paint()函数中创建一块画布在内存，把自己想绘制的东西都放在上面去，最后再一起绘制到屏幕上。值得注意的一点是，这是在awt组件中是这样的，在新的swing中，已经帮你实现了一部分的双缓冲，你只需要直接改写paint()函数就行了。

具体步骤如下：

当我们应用程序的逻辑要对系统界面进行更新时，调用repaint() 方法来通知AWT线程进行刷新操作。repaint() 方法实际会让AWT线程去调用另外一个方法，update。update方法在默认情况下会做两件事，一是清除当前区域内容，二是调用其paint()方法完成实际绘制工作。

第一个优化工作就是重写update() 方法，也就是不对当前区域进行清除工作，而直接进行绘制.

paint(g)中的g代表屏幕对象,直接对代表屏幕对象的Graphics对象进行操作，而这个操作是比较费时的操作。解决的办法是采用“双缓冲”，即我们创建一个绘制缓冲区，以bufImage表示，先将主要的图形元素一个一个地绘制到此缓冲图像上，再将此缓冲图像一次性绘到代表屏幕的Graphics对象，即paint() 方法传入的“g”上。

paint(g);直接对代表屏幕对象的Graphics 对象进行操作，而这个操作是比较费时的操作。

重写update()步骤:

得到缓冲图象this.offScreenImage = this.createImage(800,600);

看APIcreateImage用来创建一幅用于双缓冲的、可在屏幕外绘制的图像。

得到缓冲图象的画笔Graphics offg = this.offScreenImage.getGraphics();

绘制缓冲图象offg.setColor(Color.GREEN);

调用paint(),将缓冲图象的画笔传入

再将此缓冲图像一次性绘到代表屏幕的Graphics对象，即该方法传入的“g” 上. g.drawImage(offScreenImage, 0, 0, null);

### 1.2.2游戏运行框架

所谓的游戏的运行框架，就是作为一个游戏的开发者，你必须明白一个动态类型的游戏是怎样“动”起来的。在游戏制作中，有一个专业术语叫做“帧”。所谓的“帧”，就是屏幕中出现的某一刻的图像，注意这是一个图像，是静止的。当很多的图像在短时间内在同一个区域内连续的叠加出现，就会出现连续的动画效果。这是利用的人眼的视觉暂留时间。一般的，当每秒中的绘制帧的速度达到三十帧及以上时，人就会觉得画面动了起来。

当开始编代码时，还需要了解的就是一个游戏是怎样通过代码逻辑转化为可动的动画的。首先，要做动画，肯定必须要用到多线程，这是必然的。这个线程类似一个定时器的作用，来通知屏幕进行定期的重绘。一般像一些高级语言，C#、java都有一个叫做Timer的类，

这个类本身就是定时器类。

然后在一个游戏驱动中，我们一般都是有四个部分组成的：

1. 初始化方法（initialize）
2. 游戏运行方法（run）
3. 场景各类属性更改方法（changeProperties）
4. 渲染方法（render）

这是四个大体功能，将整个游戏带动起来。大型游戏需要做的就是对这些功能进行割、细化到每一个细小的耦合度小的模块中。对于初始化部分，一般是在刚启动游戏以及游戏结束后，再接着开始时来进行调用的。运行方法是整个游戏的核心，由它带动真个游戏的流畅运行。属性更改方法，是最最重要的方法，你的物体运动、碰撞检测、粒子的物理模拟都是在这里进行。最后一个渲染就是把属性改变之后的各物体按照某种顺序依次的绘制到屏幕上，最终呈现给玩家。这四个方法应该在一个线程里运行，一般对于一个不是太大的游戏，一个运行线程已经足矣。

## 1.3开发环境

我这里所用的开发环境主要是：

1. IDE是 eclipse(luna版)
2. JDK是1.8版的
3. PLATFORM:windows 8.1

# 2需求分析

## 2.1 需求概述

纵观现在国内市场，手游小游戏比较火，开发一个游戏，市场效益比较高，而且发现这个游戏，在自身的技术上，是可以很快实现的，也能够预想会有许多的玩家会青睐这个游戏，故决定开发这款游戏。（对于一般的公司，是这么考虑的。对于在读大学生的我，只是当时我玩了这个游戏，然后对它蛮感兴趣的，所以就去实现它了）

## 2.2 功能描述

SMOVE这个游戏，规则非常简单，所有的球都在一个九宫格中移动，一次只能上、下、左、右移动一个格子，不能越出九宫格。玩家操纵自己的球，来吃到目标球。同时从屏幕的四面八方会有障碍球出现，玩家必须通过操纵键盘的上、下、左、右来躲避障碍球，并且吃到目标球。目标球每次只会出现一次，一旦被玩家球吃到后，会立刻出现另一个在九宫格内，玩家通过不断地吃目标球来得分，吃一个得一分，得分越高越厉害。

# 3整体设计

## 3.1类图视图（见图1）

H:\工作文件\类图.tif

（图1）

## 3.2类模块设计

根据上面的类图，每个类中最重要的几个特性与方法都已经列出，这里就不在赘述了。

# 4详细设计

## 4.1函数实现部分

这里仅贴出最重要的几部分代码：

这个方法是游戏的运行核心：

public void Run()

{

MoveAllElement();

}

public void MoveAllElement()

{

//TODO 玩家球的碰撞检测

for(BlockBall ball:blockBalls){

if(Constant.IsBallCrashed(playBall.centreCircle, ball.centreCircle)){

IsStarted=false;

}

}

if(keyEvent.IsInFrame && frame!=4 ){

frame+=1;

playBall.move(keyEvent.keyInfo);

}

else {

frame=0;

keyEvent.IsInFrame=false;

}

//TODO 玩家球的吃球情况

if(rewardBall.IsLife&&Constant.IsBallCrashed(playBall.centreCircle, rewardBall.centreCircle)){

eatFlag=true;

rewardBall.IsLife=false;

eatBallCnt++;

}

if(eatFlag)

eatFrame++;

//blockball 的碰撞检测

for(BlockBall ball:blockBalls)

if(Constant.IsBallCrashed(playBall.centreCircle, ball.centreCircle))

IsStarted=false；

for(BlockBall ball:blockBalls){

if(!ball.IsLife){

pos[ball.srcID]=0;

bianNum[Math.floorDiv(ball.srcID, 3)]--;

ball.autoProduce(eatBallCnt,pos,bianNum,blockBalls);

ball.IsLife=true;

ball.IsVisible=false;

}

ball.moveAuto();

}

}

这个方法是渲染的核心，把背景、目标球、玩家球、障碍球、计数字符绘到内存的画布上

//渲染所有的物理元素

public void RenderAllElement(Graphics g) {

paintBackGround(g);

paintRewardBall(g);

paintPlayBall(g);

paintBlockBalls(g);

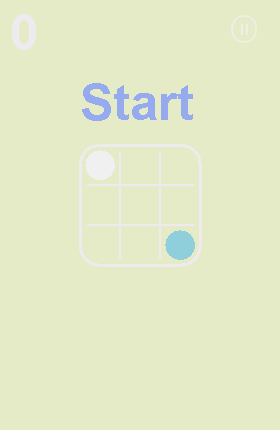
paintString(g);

}

## 4.2实现中的难题

1. 对双缓冲技术的实现，在一开始实验时，总是出现不了理想的效果
2. 对timer的理解不够，使得画面运行的不流畅
3. 一些绘制的小的细节重视不够，总是出现一些莫名奇怪的bug
4. 对障碍球出现的动态掌控很难做到尽善尽美

# 5编码测试

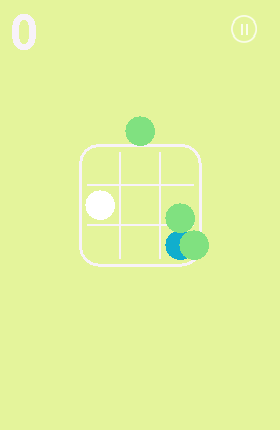
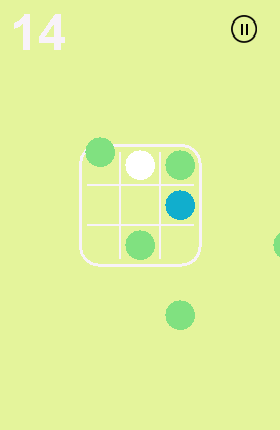
在得到前面的详细设计后，按照设计的指示，上机敲代码。码好代码后，进行一些必要的黑白盒测试，加强产品的健壮性。

下面是游戏界面的一个展示与测试。

开始界面：（见右图2）

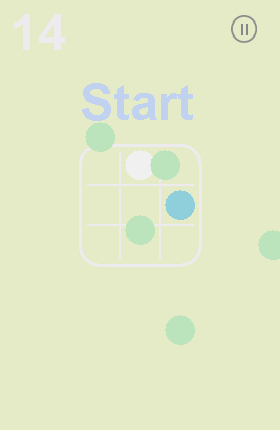
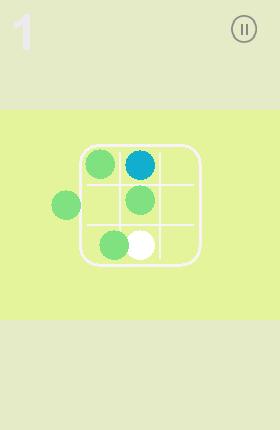
（图2）

游戏运行界面：（见图3，图4）

（图3） （图4）

游戏的结束界面：（见图5、图6）

（图5） （图6）

# 6系统评估

## 6.1 收获心得

经过这次的课程设计，我了解了一个应用程序从可行性分析、需求分析到整体设计、详细设计等一些系列制作软件流程，更加深刻的体会到，开发软件不是一个人可以胜任的事情，必须有一个好的团队，每一个人各司其职，各尽所能，完成好每一个部分，最后再一整合，得到一个像样的产品。但是有了这一次的设计，从零行代码到最终可以演示的这一全程体验，加深了我对软件工程这一概念的认识。并且也对一些较小游戏的开发，有了一个大致的开发流程思想。这是一个宝贵的经验。所谓“高楼建成非一日之功”，只有在不断地学习他们的代码，不断地掌握新的理论，并且将自己的理论投于实践，才能真正的学到东西。

## 6.2 改进之处

由于开发的比较仓促，游戏里面还是有许多不足的地方。具体体现在：

1. 游戏里面没有添加一个关闭按钮，游戏运行后，不能正常退出。
2. 游戏的界面做的还是不够精美，后期需要多多加工
3. 游戏的关数设置不够，这个游戏是以10个球为一关，但是我只编码了前面的20个球，所以还是要再加工一番
4. 游戏的窗口没有实现拖动的效果