中山大學
SUN YAT-SEN UNIVERSITY

Java 程序设计

期末设计报告

学 院 名 称 ：	数据科学与计算机学院
专业（班级）：	18 级软件工程 2 班
学 生 姓 名 ：	郑卓民/张泽健
学　　号	18342134/18342133
时　　间 ：	2020　年　1　月　1　日

目录

一. 项目背景介绍

伴随着互联网的发展，互联网游戏一向发展火爆。“球球大作战”便是其中一款。但市面上的同款“球球大作战”游戏大多规则复杂，功能累赘，界面过于花哨而不够简约，使得小孩与老人难以理解游戏运行逻辑而拒绝参与游戏。

本次期末项目我们便尝试运用本学期的 Java 知识制作一款简化版的“球球大作战”。使这款敏捷类游戏能够造福更多的小孩与老人，促进小孩思维拓展，延缓老人反应力衰退。

众所周知，制作一款游戏，需要考虑多个方面且不仅局限于代码层面。开始前，需要先对市场进行调研，把握潮流走向，此环节即体现在上文中对用户群体的需求分析。大致决定了制作哪款游戏后，就需要进行详尽的前期策划，定位必须明确，前期必须做好充足的准备，为代码的编写铺路，本阶段将体现在游戏功能的设计、界面 UI 的设计、关键大模块的定义等等方面。到了开发环节，就需要抓住人心，把自己了解到的目标人群喜欢的元素融入游戏，使游戏更耐玩。

因此，我们希望通过这样一个尝试，既能在游戏中实现自己想要的功能，还可以让学到的知识更加稳固，增加项目经验，其中重点目标便是一下：

1. 熟练掌握 Java 面向对象编程（重中之重）。
2. 熟练掌握 Swing 图形用户界面编程以及事件处理等，掌握 Java 绘图技术。
3. 熟练应用多线程编程的基本原理，深入理解线程创建、启动、关闭等工作。
4. 培养使用 Java 语言完整完成项目的能力。
5. 培养独立查找资料进行自学，并解决问题的能力。

二. 项目任务

设计并编程实现“球球大作战”程序，用户能通过游戏主页引导点击开始按钮进入游戏界面或者点击退出按钮关闭游戏程序。

主页与游戏页的 UI 设计贴近人心，符合大众审美，无歧义选项，精准引导用户进入与进行游戏。

用户的球体由用户鼠标控制，游戏中其他球体则遵循一下几个规律：随机颜色、随机速度、随机大小、随机运动方向，并需要设计好球体与墙壁碰撞等事件的处理。

利用文件存储技术，对用户的游戏体验与得分进行记录，达到持久化记录效果，即便用户退出游戏，当其再次进入时，仍然能看到之前在游戏中取得的成就。

三. 游戏定位

游戏简介：

球球大作战是一款很棒的敏捷小游戏。弱肉强食的时代，就是大鱼吃小鱼的社会现象。现在游戏中，玩家（用户）需要控制大球吃掉比自己小的球，让自己变得更大才能吃到更大的小球。一起来帮助小球长大吧！

游戏目标：

合理操作，避开大球与危险物，吃到更多的小球以及补给品。

四. 开发工具与平台

1. JavaSE-11

2. eclipse

五. 游戏功能设计

游戏对象及其对应任务：

1. **玩家（球）**：玩家通过鼠标操控一个属于玩家的小球，小球始终跟随于鼠标，鼠标速度移动的速度也即是玩家小球移动的速度。玩家（球）碰到游戏中的其他物件将触发对应事件。

2. **炸弹**：炸弹定义为游戏中的危险物，当玩家操控的小球撞上了黑色的炸弹，则小球立即死亡，被判定为游戏失败。

3. **胶囊**：胶囊定义为游戏中的补给品，当玩家操控的小球吃到了地图上随机生成的胶囊，游戏界面顶上的倒计时进度条则会增加少许，相当于获得了更多的生存时间。

4. **倒计时进度条**：进度条定义为玩家的“血条”，处在游戏界面的顶端，采用倒计时模式，随时间的推移，其“生命值”逐渐减少。当其生命值减少至 0 时，游戏被判定为失败。

5. **其他球**：随机产生，随机移动，颜色各异，速度不一，与玩家球进行互动。

6. **特殊的黄色小球**：黄色球定义为游戏中的陷阱（也可叫惩罚），其具体运动性质与普通球一致，但是，当玩家吃下了黄色的小球，则会变回最开始的大小。同时，倒计时进度条将**不会**重置。因此，玩家应当注意避开黄色的小球。

7. **得分与难度**：玩家吃掉更小的球不仅可使自身变大，还会使得分相应增加，随着得分的增加，游戏难度会相应增加，但设置了一个最高游戏难度。

游戏基本玩法设计：

玩家可以通过鼠标来操控移动玩家小球。玩家小球可以吃掉比自身更小的其他小球来使自身变大并得分。当不慎撞上了比自己大的球或者进度条耗尽或者碰到炸弹的时候，游戏会被判定为失败。

六. 设计模式与系统类图的设计

设计模式：

工厂模式（Factory Pattern）是 Java 中最常用的设计模式之一。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。在本项目中，考虑到游戏界面中的绝大多数对象的属性都相似（都属于球），可以考虑创建一个球类生产工厂，来生产各种各样的球类对象（玩家小球、其余小球、补给品、危险品）。

系统类图的设计：

在工厂模式中，我们在创建对象时不会对客户端暴露创建逻辑，并且是通过使用一个共同的接口来指向新创建的对象。

类图是 UML 图的一种，也是最常用的 UML 图。它可以清楚的表示程序中类的基本结构，类与类之间的结构关系，掌握 UML 图对于了解系统的总体结构和设计模式有着重大的作用。我们将提前设计好系统类图以便于游戏的代码实现。

在面对对象的 Java 编程中，类是对象的骨架，其包含三个组成部分：Java 中定义的类名、属性 Attributes、该类所提供的方法 Methods。

对于标准的 UML 类图，第一行表示类的名称，第二行表示类的属性，即其成员变量。第三行表示类的方法。

经过以上已做的项目需求分析，可以归结出我们需要的几个对象（类）分别是什么：

1. GUI、PaintPanel、ProgressUI： GUI 类为 UI 界面设计的总控制中心，决定了游戏界面的样式、球的绘制、成绩的显示、进度条的显示，而另外的两个类 PaintPanel、ProgressUI 均是服务于 GUI。

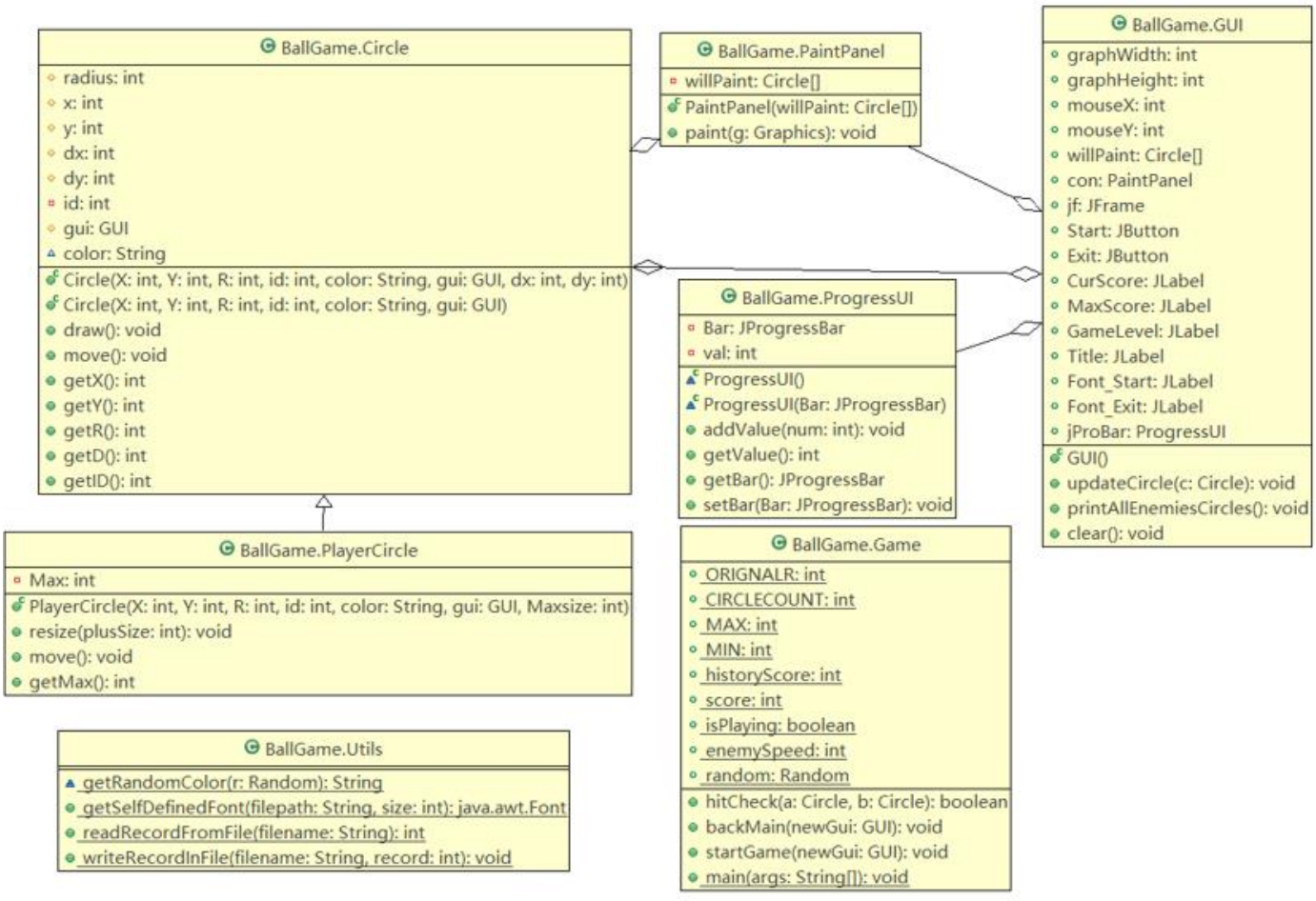
2. Circle、PlayerCircle： Circle 为一个球类，需要定义球体通用的产生、初始化、运动属性等基础信息与方法，而 PlayerCircle 是一个继承于 Circle 的子类，用于区别普通小球，

作为玩家的操作对象，与父类不同地，该类需定义相关方法获取鼠标位置并变更 PlaerCircle 相关属性。

3. Utils：该类是一个工具方法的集合，将一些常用的工具方法封装在一个工具包里，便于管理与调用，其中包含初始化球时候需要的随机颜色方法、文件存储功能中对用户数据的读取与保存。

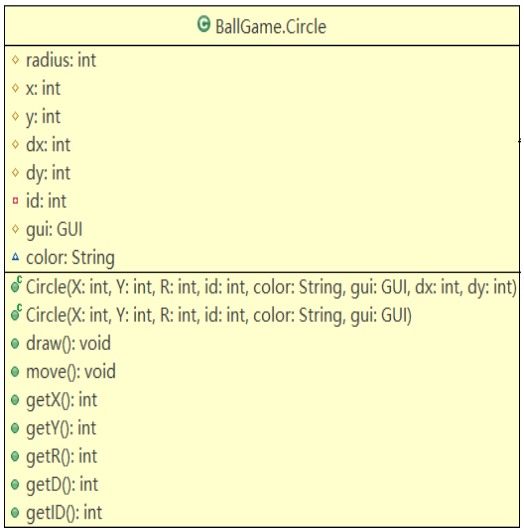
4. Game：该类为主类，包含 main 函数，控制游戏的开始结束。其中包含游戏的启动与运行时的相关逻辑处理，以及游戏结束的处理。

下面将根据上述分类，以及对应类的功能需求，设计出各个类需要的变量以及方法，并绘制本次项目的系统类图（Class Diagram），以便观察各类之间的关系与依赖：



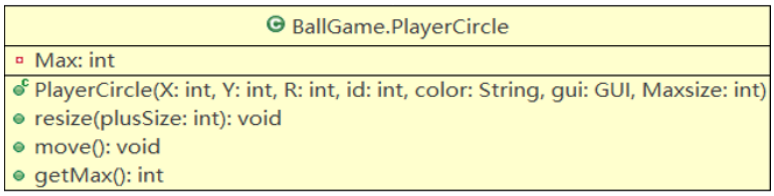
七. 各模块详细说明及其实现方法

Circle 类：



Circle 类是项目中最基础的通用类，其可在创造玩家对象，补给品对象，炸弹对象，其他小球对象中发挥作用。其包含了运动型球的基础属性（坐标（x，y）、颜色 (color)、运动矢量 (dx，dy)，同时还有一个 GUI 变量 (gui)，其作用是让每个球在反应各种突发事件时候，都有权使得用户界面产生变化（简单地说，就是帮助球类对象完成相应的事件处理方法）。

PlayerCircle 类：



该类继承于 Circle 类，用于作为玩家小球的对象，特别地，其有一个 Max 最大值，用于限定玩家球最大大小，此外还有一个 resize 方法用于更改球的大小，一个 move 方法用鼠标位置信息来决定玩家小球的位置信息。

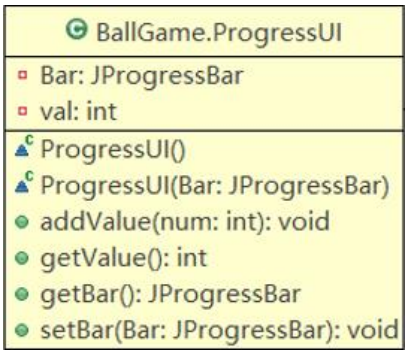
GUI 类：

该类封装了所有游戏界面需要的组件，实现 UI 界面的整体样式。成员变量里主要包含了一些窗口信息如高度和宽度、鼠标的位置、协助记录球集合的数组以及一些基本组件（如：JButton、JLabel 和自定义 UI 组件 ProgressUI)。

GUI 的初始化过程也即游戏界面的初始化过程，其中包括 JFrame 的创建，以及将各组件插入到 JPanel 中，其中这里用到的 JPanel 为自定义的 PaintPanel 类（继承于 JPanel)。

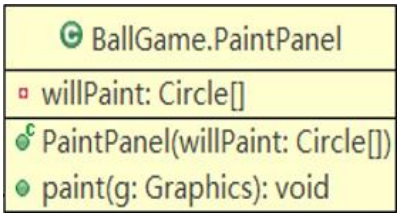
而对于界面中各小球的操作（updateCircle、printAllEnemiesCircles、clear），是通过操作小球集合数组（willPaint）并更新界面来实现。

ProgressUI：



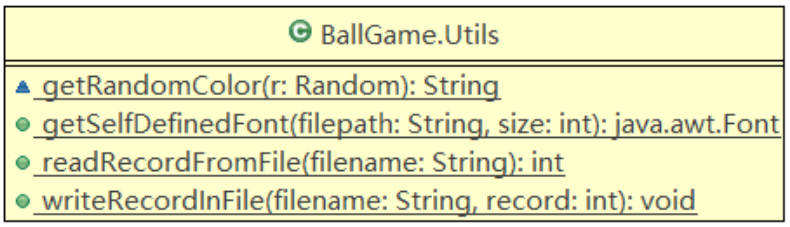
该类为自定义 UI 组件（进度条），实现原理就是通过一个 val 值来协助对 JProgressBar 组件成员变量的属性修改，达到进度条所需要随值的变化而颜色变化的效果。

PaintPanel 类：



该类继承于 JPanel，重写（Override）了 paint 方法，增加了绘制 willPaint 中所有 Circle 的功能，根据 Circle 中属性的不同，绘制药丸、炸弹、各种颜色/形状的球。

Utils：



该类用作个人工具包，其中包含了四个常用方法：

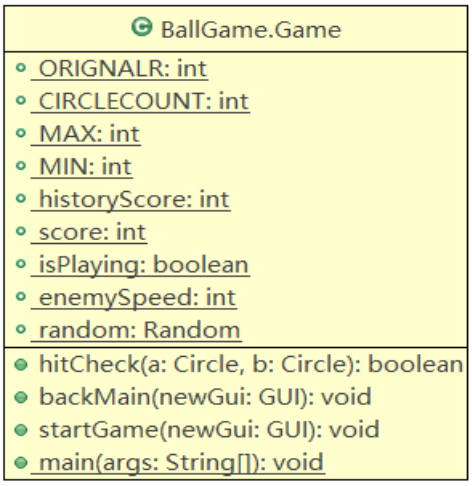
（1）获取随机颜色：通过简单处理利用 Random 包获取到随机数以及对该数的 case 语句来获得对应的颜色。

（2）获取自定义字体：该部分参考 Font 包标准用法导入从网络中下载的自定义字体包。

（3）读取信息：利用 FileReader 与 Scanner 获得 txt 文件中的数据。

（4）保存信息：利用 FileWriter 与 StringBuilder 将数据保存在 txt 文件中。

Game:



该类为游戏的控制中心，决定了游戏开始至结束的各种逻辑与事件处理方法。游戏开始时对游戏界面以及相关变量进行初始化，游戏结束时对游戏重要数据进行存储。

其中利用多线程原理保证游戏的正常运行。总体包括四个重要的线程，分别是:

（1）玩家（PlayerBall）移动线程：在该线程中，利用 while 语句时刻利用玩家的鼠标位置来更新玩家小球的位置信息。

(2)其余小球移动线程:在该线程中,利用 for 语句对每个小球对象进行成员函数(move)的调用（其中需要注意若判断当前对象为空，说明该球被玩家所吃，所以需要根据随机判断条件来决定生成一个怎么样的新球)。

（3）成绩计算线程：在该线程中，作用不只是更新玩家的得分情况，更是对玩家与其他对象交互的逻辑判断，与炸弹、药丸、大小不一的球相撞，都会对应不同的结果，以及不同的成绩变更情况。

（4）进度条线程：在该线程中，利用 sleep 函数对进度条的 val 值进行随时间递减的操作，并在 val 值为 0 时将玩家游戏状态设置为中止。

八. 知识点应用说明

1. Java 面向对象编程（类与对象、超类与继承、异常处理、多线程、文件存储等)。

2. Java 图形界面的设计与构建。

3. Java 的工厂设计模式。

九. 创新点或技术难点说明

创新点：

本次项目设计的简化版“球球大作战”的游戏规则与市面上的同款游戏有所不同，游戏的运行不只是简单的大球可吞掉小球，还增加了陷阱/惩罚（黄球）、补给品（药丸）和危险品（炸弹）的概念，增加了游戏的趣味性和可玩性，还给游戏玩家添加了一个约束时间（进度条），以防挂机消极游戏行为。

技术难点：

（1）设计出符合大众审美的游戏主页、游戏进行页、游戏结束页，并利用 Java Swing 完美地用相关组件绘制出所需要的 UI 界面。

（2）根据游戏功能需求，结合多线程程序的特点与多线程原理，分析出需要哪些线程来协助完成游戏的正常运作，并分配给每个线程合适的任务。

（3）在并发编程中存在线程安全问题，主要原因有：1.存在共享数据 2.多线程共同操作共享数据。我们必须利用多线程原理处理好并发线程中存在的各种问题，使得整体代码健壮性更强大。

十. 小组成员分工

成员 1：郑卓民

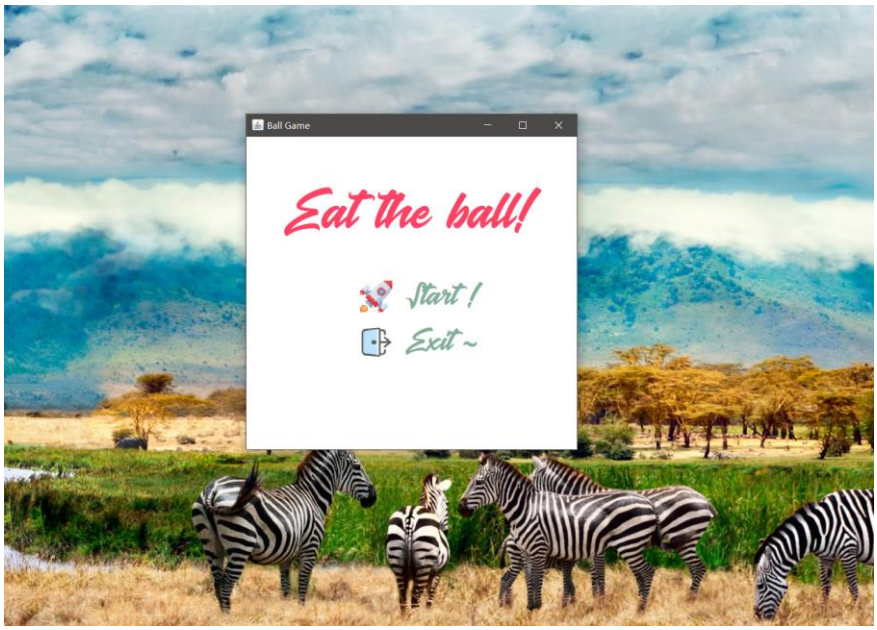
参与游戏题材选取、游戏项目初步分析与各功能模块的设计过程；负责游戏内部运行逻辑的设计与实现，对整体框架/构造利用多线程原理并发编程，构造自定义组件，满足功能需求。

成员 2：张泽健

参与游戏题材选取、游戏项目初步分析与各功能模块的设计过程；负责游戏 UI 界面的设计与构建，以及重要数据的文件读取/存储功能。

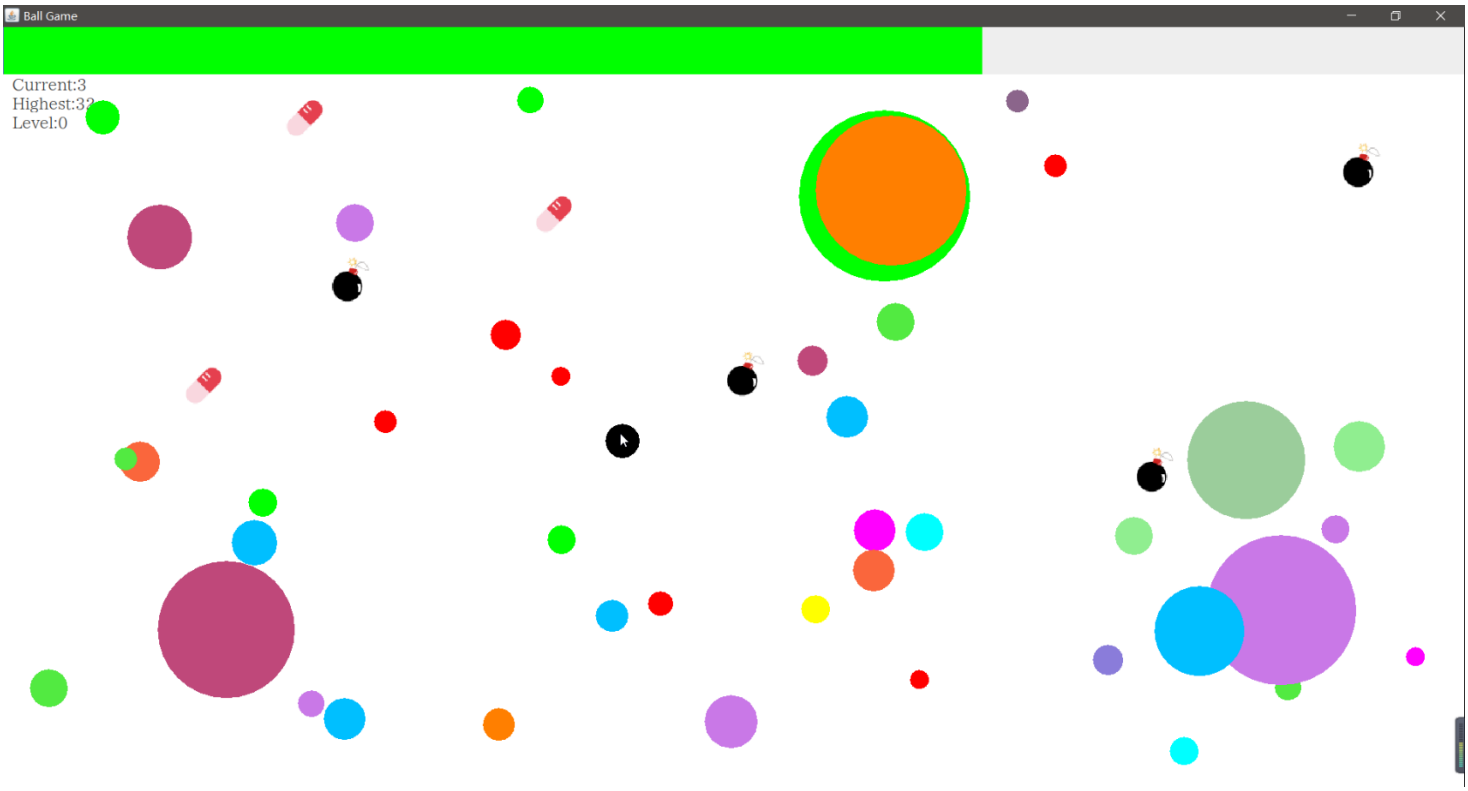
十一. 游戏运行成果展示

游戏开始界面：点击小火箭可进入游戏主界面；点击退出图标可正常退出游戏；

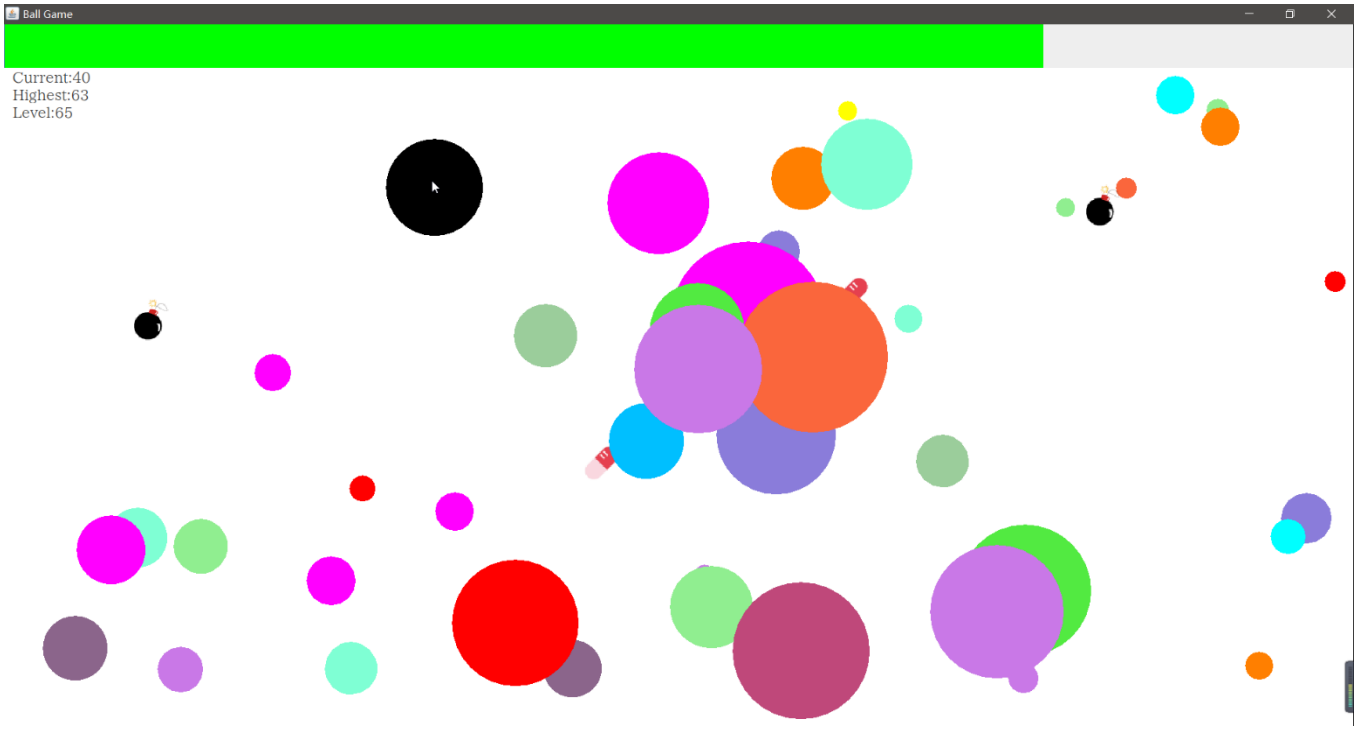


游戏运行主界面：

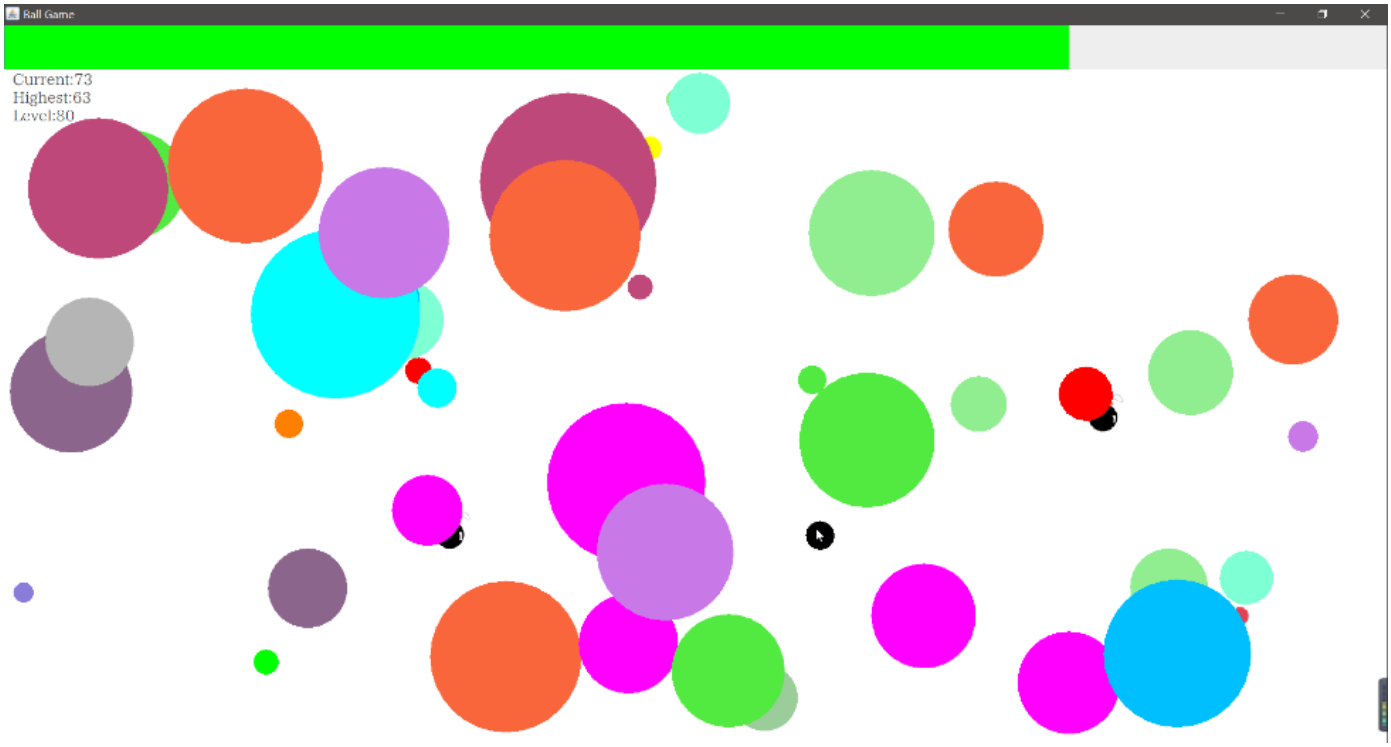
正常状态：左上角为分数栏，中间鼠标位置的黑球为玩家，窗口上方为进度条。



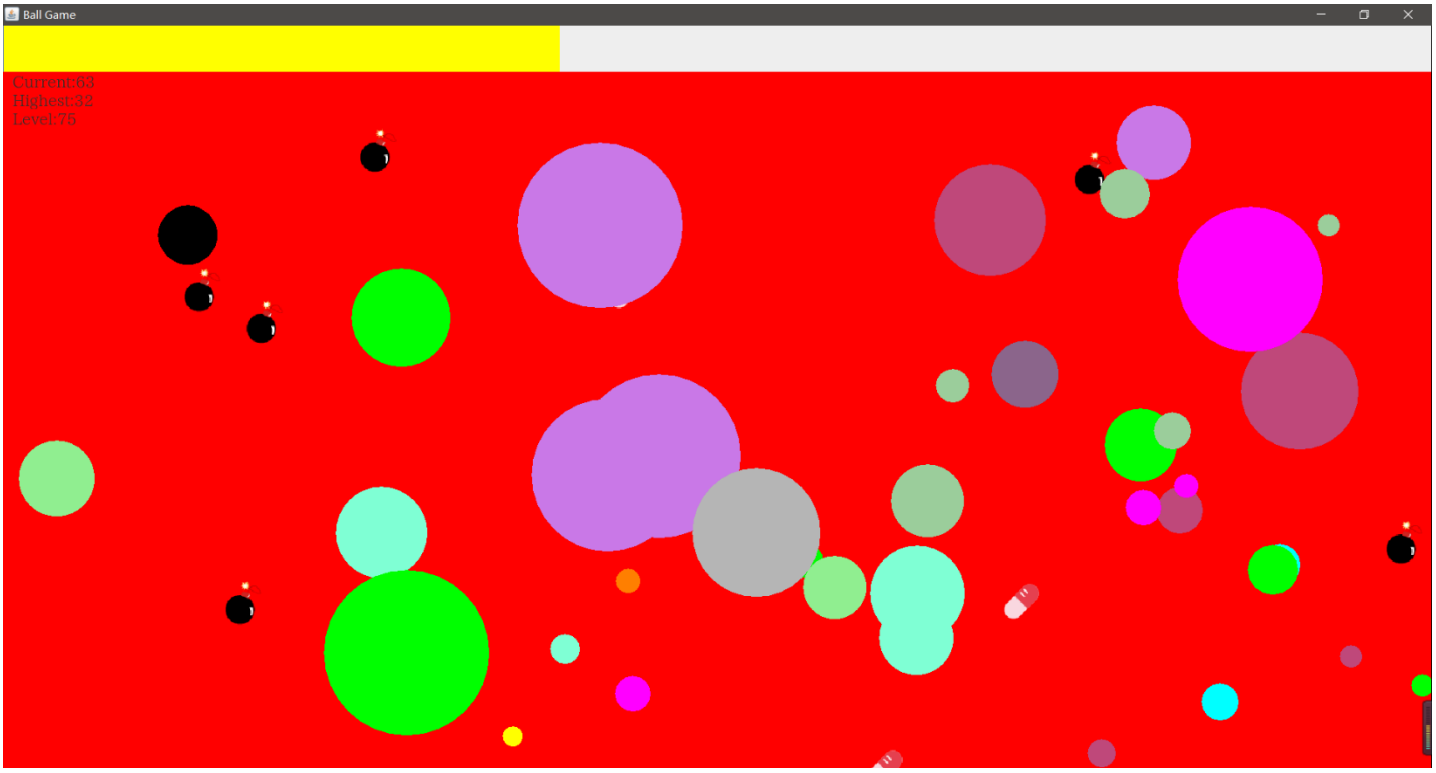
难度升级：



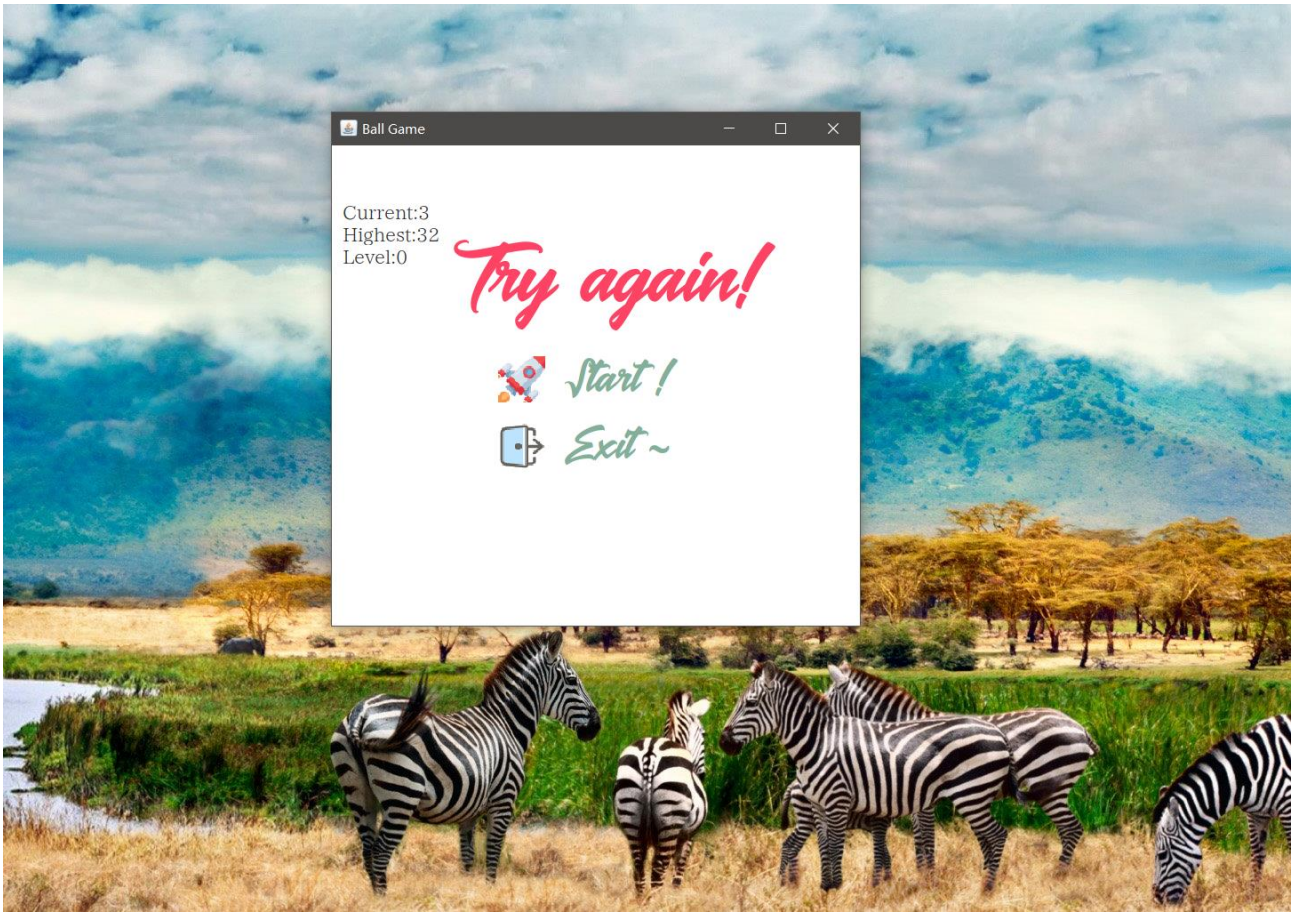
高难度下碰到陷阱变回初始大小：



玩家死亡、游戏结束：



游戏结束界面： 可选择继续游戏或者退出游戏。



十二. 项目总结

本次 Java 期末项目选取了市面上十分火爆的“球球大作战”题材，初衷即创造一个简化版的“球球大作战”，让规则简单的同时，游戏富有趣味，可玩性高。

此外，作为一款敏捷类游戏，本次简化版“球球大作战”的用户面极宽，原因在于，市面上复杂的游戏规则、花哨的道具使用与多人混乱的同台竞技可能会：让小孩与老人因难以理解而拒绝参与；让青壮年感到功能累赘。而此简化版本，可以既简约且能训练反应力的特点，赢得大多小孩与老人的欢心，同时有助于小孩提高反应力、敏捷度，从而提升其创造力与思维能力，以及有助于老人减缓大脑反应能力的衰退。

重中之重，本次项目执行让我们了解并体验利用 Java 语言开发完成完整项目的过程，学到了并熟练应用了许多编程代码之外的知识，同时使在课堂上学到的知识在脑海深处更加牢固。

十三. 完整代码展示

Game.java

```
1.  package BallGame;
2.
3.  import javax.swing.*;
4.  import java.awt.Color;
5.  import java.awt.event.ActionEvent;
6.  import java.awt.event.ActionListener;
7.  import java.awt.event.WindowAdapter;
8.  import java.awt.event.WindowEvent;
9.  import java.util.Random;
10.
11. public class Game {
12.     public static final int ORIGNALR = 15;
13.     public static final int CIRCLECOUNT = 50;
14.     public static final int MAX = 100;
15.     public static final int MIN = 10;
16.     public static int historyScore = 0;
17.     public static volatile int score = 0;
18.     public static volatile boolean isPlaying;
19.     public static int enemySpeed = 100;
20.     public static Random random;
21.
22.     public boolean hitCheck(Circle a, Circle b) {
23.         // 检查两个球是否触碰 ： 没触碰 则返回 true
24.         return a.getR()+b.getR() >= Math.sqrt(Math.pow(a.getX()-b.getX(),2)+Math.pow(a.getY()-
    b.getY(),2));
25.     }
26.
27.     public void backMain(GUI newGui){
28.         newGui.jf.getContentPane().setBackground(Color.decode("#FFFFFF"));
29.         newGui.jf.setBounds(newGui.graphWidth/2-300, newGui.graphHeight/2-300, 490, 470);
30.         newGui.Exit.setVisible(true);
31.         newGui.Start.setVisible(true);
32.         newGui.Font_Start.setVisible(true);
33.         newGui.Font_Exit.setVisible(true);
34.         newGui.Title.setText("Try again!");
35.         newGui.Title.setVisible(true);
36.         newGui.Title.setBounds(110, 70, 500, 100);
37.         newGui.jProBar.getBar().setVisible(false);
38.         newGui.MaxScore.setText("Highest:"+historyScore);
39.         newGui.clear(); newGui.jf.getContentPane().repaint();
```

```
40.     }
41.
42.     public synchronized void startGame(final GUI newGui) throws InterruptedException {
43.         // 创造游戏玩家对象
44.         final PlayerCircle[] player = {new PlayerCircle(newGui.mouseX, newGui.mouseY, ORIGNALR,
45.                                                         CIRCLECOUNT, "#000000", newGui, MAX)};
46.         final Circle[] enemies = new Circle[CIRCLECOUNT];// 创造敌人数组
47.         score = 0;
48.         isPlaying = true;
49.         random = new Random();
50.         for (int i = 0; i < CIRCLECOUNT; i++) { // 创造敌人对象
51.             if (i < CIRCLECOUNT / 4 * 3) {
52.                 int enermyR = random.nextInt(player[0].getR()) + MIN;
53.                 if (i == CIRCLECOUNT / 2) {
54.                     do {
55.                         enemies[i] = new Circle(random.nextInt(newGui.graphWidth-
    enermyR*2)+enermyR,
56.                                 random.nextInt(newGui.graphHeight-enermyR*2-
    70) + enermyR+50, enermyR,
57.                                 i, "#FFFF00", newGui, random.nextInt(10) + 1, random.nextInt(10) + 1
    );
58.                     } while (hitCheck(enemies[i], player[0])); // 判断不会一开始就触碰到 player
59.                 }else {
60.                     do {
61.                         enemies[i] = new Circle(random.nextInt(newGui.graphWidth-
    enermyR*2)+enermyR,
62.                                 random.nextInt(newGui.graphHeight-enermyR*2-
    70) + enermyR+50, enermyR,
63.                                 i,Utils.getRandomColor(random), newGui, random.nextInt(10) + 1, rand
    om.nextInt(10) + 1);
64.                     } while (hitCheck(enemies[i], player[0]));
65.                 }
66.             } else {
67.                 int enermyR = random.nextInt(MAX - player[0].getR()) + player[0].getR();
68.                 do {
69.                     enemies[i] = new Circle(random.nextInt(newGui.graphWidth - enermyR * 2) + enermy
    R,
70.                             random.nextInt(newGui.graphHeight - enermyR * 2-
    70) + enermyR+50, enermyR,
71.                             i, Utils.getRandomColor(random), newGui, random.nextInt(3) + 1, random.n
    extInt(3) + 1);
72.                 } while (hitCheck(enemies[i], player[0]));
73.             }
74.         }
```

```
75.         newGui.jf.getContentPane().repaint(); // repaint 重绘 ， 创造游戏开始最初样子
76.         // 创造进度条线程
77.         class progressUI implements Runnable {
78.             public synchronized void run() {
79.                 try {
80.                     Thread.sleep(1000);
81.                 } catch (InterruptedException e) {}
82.                 while(newGui.jProBar.getValue() > 0 && isPlaying) {
83.                     newGui.jProBar.addValue(-1);
84.                     try {
85.                         Thread.sleep(200);
86.                     } catch (InterruptedException e) {}
87.                 }
88.                 isPlaying = false;
89.             }
90.         }
91.         // 创建 player 移动的线程
92.         class playerMoving implements Runnable {
93.             @Override
94.             public synchronized void run() {
95.                 try { // 开始时候 有一个缓冲时间给用户
96.                     Thread.sleep(1000);
97.                 } catch (InterruptedException e) {}
98.                 while (isPlaying && player[0] != null) {
99.                     player[0].move(); // 更新球的坐标（鼠标位置）
100.                }
101.            }
102.        }
103.        // 创建 enemy 移动的线程
104.        class enemyMoving implements Runnable {
105.            public synchronized void run() {
106.                try { // 开始时候 有一个缓冲时间给用户
107.                    Thread.sleep(800);
108.                } catch (InterruptedException e1) {}
109.                while (isPlaying && player[0] != null) {
110.                    for (int i = 0; i < enemies.length; i++) {
111.                        if (enemies[i] == null) { // 球没了（被吃了）
112.                            int enermyR;
113.                            if (i < CIRCLECOUNT / 4 * 3) {
114.                                enermyR = random.nextInt(player[0].getR()-MIN)+MIN;
115.                            } else {
116.                                enermyR = random.nextInt(MAX-player[0].getR())+player[0].getR();
117.                            }
118.                            do{
```

```
119.                                 enemies[i] = new Circle(random.nextInt(newGui.graphWidth-
    enermyR*2)+enermyR,
120.                                         random.nextInt(newGui.graphHeight-enermyR*2-
    70) + enermyR+50, enermyR,
121.                                         i, Utils.getRandomColor(random),newGui, random.nextInt(3) +
    1, random.nextInt(3) + 1);
122.                             } while (hitCheck(enemies[i], player[0]));
123.                             if (i == CIRCLECOUNT / 2) {
124.                                 do {
125.                                     enemies[i] = new Circle(random.nextInt(newGui.graphWidth-
    enermyR*2)+enermyR,
126.                                             random.nextInt(newGui.graphHeight-enermyR*2-
    70) + enermyR+50, enermyR,
127.                                             i, "#FFFF00",newGui, random.nextInt(10) + 1, random.next
    Int(10) + 1);
128.                                 } while (hitCheck(enemies[i], player[0]));
129.                             }
130.                         }
131.                         enemies[i].move();
132.                     }
133.                     newGui.printAllEnemiesCircles(); // 重绘所有敌人球
134.                     try {
135.                         Thread.sleep(enemySpeed); // 将 speed 设置成更新时间间隔
136.                     } catch (InterruptedException e) {}
137.                 }
138.             }
139.         }
140.         // 创造成绩计算（包含运行逻辑）线程
141.         class countScore implements Runnable {
142.             public synchronized void run() {
143.                 while (isPlaying) {
144.                     for (int i = 0; i < enemies.length; i++) {
145.                         if (enemies[i] != null && player[0] != null) {
146.                             if (hitCheck(enemies[i], player[0])) {
147.                                 //炸弹检测
148.                                 if(enemies[i].color.equals("#EED5D2")) {
149.                                     isPlaying = false;  break;
150.                                 }
151.                                 //药 检测
152.                                 if(enemies[i].color.equals("#6A5ACD")) {
153.                                     newGui.jProBar.addValue(5); // 进度条加 5
154.                                     enemies[i] = null;  continue;
155.                                 }
156.                                 //普通球检测
```

```
157.                                 if (player[0].getR() > enemies[i].getR()) {
158.                                     if (i != CIRCLECOUNT / 2) {
159.                                         player[0].resize(1);
160.                                         score++;
161.                                         newGui.CurScore.setText("Current:"+score);
162.                                         if(score != 0 && score % 10 == 0) {
163.                                             if(enemySpeed > 40) {
164.                                                 enemySpeed -= 20;
165.                                             }else if(enemySpeed > 20) {
166.                                                 enemySpeed -= 5;
167.                                             }else if(enemySpeed > 1) {
168.                                                 enemySpeed -= 1;
169.                                             }
170.                                             newGui.GameLevel.setText("Level:"+(100-enemySpeed));
171.                                         }
172.                                         newGui.jProBar.addValue(3);
173.                                     } else {
174.                                         player[0].resize(-1 * (player[0].getR()-ORIGNALR));
175.                                     }
176.                                     enemies[i] = null;
177.                                 } else {
178.                                     isPlaying = false;  break;
179.                                 }
180.                             }
181.                         }
182.                     }
183.                 }
184.                 // 游戏结束 重置数组信息 更新成绩信息
185.                 for (int i = 0; i < enemies.length; i++) {
186.                     enemies[i] = null;
187.                 }
188.                 if(score > historyScore) {
189.                     historyScore = score;
190.                 }
191.                 player[0] = null;
192.                 newGui.jf.getContentPane().setBackground(Color.RED);
193.                 try {// 死亡界面缓冲
194.                     Thread.sleep(800);
195.                 } catch (InterruptedException e) {}
196.                 backMain(newGui);
197.             }
198.         }
199.         // 实例化线程
200.         playerMoving pm = new playerMoving(); Thread MC = new Thread(pm);
```

```
201.        enemyMoving em = new enemyMoving();   Thread EM = new Thread(em);
202.        countScore cs = new countScore();     Thread CS = new Thread(cs);
203.        progressUI ui = new progressUI();     Thread TP = new Thread(ui);
204.        // 启动线程
205.        MC.start(); EM.start(); CS.start(); TP.start();
206.    }
207.
208.    public static void main(String[] args) {
209.        final Game newGame = new Game(); // 开始游戏
210.        final GUI  newGui  = new GUI();
211.        isPlaying = false;
212.        historyScore = Utils.readRecordFromFile("./public/record.txt");
213.        newGui.jf.getContentPane().repaint(); // repaint 重绘 component
214.        // 开始游戏 action
215.        newGui.Start.addActionListener(new ActionListener() { // 按开始按钮
216.            @Override
217.            public void actionPerformed(ActionEvent e) {
218.                score = 0;  enemySpeed = 100;  // 初始化数据
219.                newGui.Start.setVisible(false); // 开始按钮消失
220.                newGui.Exit.setVisible(false);  // 推出按钮消失
221.                newGui.Title.setVisible(false); // 标题消失
222.                newGui.Font_Start.setVisible(false);
223.                newGui.Font_Exit.setVisible(false);
224.                newGui.CurScore.setVisible(true);  // 当前分数 显示
225.                newGui.MaxScore.setVisible(true);      // 最高成绩 显示
226.                newGui.GameLevel.setVisible(true);     // 游戏难度 显示
227.                newGui.CurScore.setText("Current:" + score);
228.                newGui.MaxScore.setText("Highest:" + historyScore);
229.                newGui.GameLevel.setText("Level:" + (100-enemySpeed));
230.                newGui.clear();
231.                newGui.jProBar.getBar().setValue(100); // 重置进度条
232.                newGui.jProBar.getBar().setVisible(true); // 进度条 显示
233.                newGui.jf.setExtendedState(JFrame.MAXIMIZED_BOTH); // 重置窗口大小
234.                try {
235.                    newGame.startGame(newGui);
236.                } catch (InterruptedException ee) {}
237.            }
238.        });
239.        // 游戏结束 action
240.        newGui.jf.addWindowListener(new WindowAdapter() {
241.            @Override
242.            public void windowClosing(WindowEvent e) { // 存储历史成绩
243.                Utils.writeRecordInFile("./public/record.txt", historyScore);
244.             }
```

```
245.         });
246.         newGui.Exit.addActionListener(new ActionListener() {
247.             @Override
248.             public void actionPerformed(ActionEvent e) { // 存储历史成绩
249.                 Utils.writeRecordInFile("./public/record.txt", historyScore);
250.                 System.exit(0);
251.             }
252.         });
253.     }
254. }
```

GUI.java

```
1.  package BallGame;
2.
3.  import javax.swing.*;
4.  import java.awt.*;
5.  import java.awt.event.MouseEvent;
6.  import java.awt.event.MouseMotionListener;
7.
8.  public class GUI {
9.      public final int graphWidth;
10.     public final int graphHeight;
11.     public int mouseX;
12.     public int mouseY;
13.     public Circle[] willPaint = new Circle[Game.CIRCLECOUNT+3];
14.     public PaintPanel con = new PaintPanel(willPaint);
15.     public JFrame jf;
16.     public JButton Start;
17.     public JButton Exit;
18.     public JLabel CurScore;
19.     public JLabel MaxScore;
20.     public JLabel GameLevel;
21.     public JLabel Title;
22.     public JLabel Font_Start;
23.     public JLabel Font_Exit;
24.     public ProgressUI jProBar;
25.
26.     public GUI(){
27.         jf = new JFrame("Ball Game");
28.
29.         // 初始化窗口
30.         Toolkit t = Toolkit.getDefaultToolkit();
31.         graphWidth = t.getScreenSize().width;
```

```
32.         graphHeight = t.getScreenSize().height-70;
33.
34.         jf.setBounds(graphWidth/2-300, graphHeight/2-300, 450, 450);
35.         jf.setLayout(null);
36.         jf.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
37.         con.setLayout(null);
38.
39.         // 初始化两个按钮 开始/退出
40.         Start = new JButton();
41.         Start.setBounds(150,190,42,42);
42.         Start.setIcon(new ImageIcon("./public/Start.png"));
43.         Start.setMargin(new Insets(0,0,0,0));
44.         Start.setIconTextGap(0);
45.         Start.setBorderPainted(false);
46.         Start.setBorder(null);
47.         Start.setFocusPainted(false);
48.         Start.setContentAreaFilled(false);
49.
50.         Exit = new JButton();
51.         Exit.setBounds(150,250,42,42);
52.         Exit.setIcon(new ImageIcon("./public/Exit.png"));
53.         Exit.setMargin(new Insets(0,0,0,0));
54.         Exit.setIconTextGap(0);
55.         Exit.setBorderPainted(false);
56.         Exit.setBorder(null);
57.         Exit.setFocusPainted(false);
58.         Exit.setContentAreaFilled(false);
59.
60.         // 设置字体
61.         Font font=Utils.getSelfDefinedFont("./public/Baksoda.otf",  85);
62.         Font font1=Utils.getSelfDefinedFont("./public/font.ttf",  17);
63.         Font font2=Utils.getSelfDefinedFont("./public/Baksoda.otf",  48);
64.
65.         CurScore = new JLabel();
66.         CurScore.setVisible(false);
67.         CurScore.setFont(font1);
68.         CurScore.setBounds(10, 50, 200, 20);
69.
70.         MaxScore = new JLabel();
71.         MaxScore.setVisible(false);
72.         MaxScore.setFont(font1);
73.         MaxScore.setBounds(10, 70, 200, 20);
74.
75.         GameLevel = new JLabel();
```

```
76.         GameLevel.setVisible(false);
77.         GameLevel.setFont(font1);
78.         GameLevel.setBounds(10, 90, 200, 20);
79.
80.         Font_Start = new JLabel("Start !");
81.         Font_Start.setVisible(true);
82.         Font_Start.setFont(font2);
83.         Font_Start.setBounds(210, 160, 500, 100);
84.         Font_Start.setForeground(Color.decode("#83af9b"));
85.
86.         Font_Exit = new JLabel("Exit ~");
87.         Font_Exit.setVisible(true);
88.         Font_Exit.setFont(font2);
89.         Font_Exit.setBounds(210, 220, 500, 100);
90.         Font_Exit.setForeground(Color.decode("#83af9b"));
91.
92.         Title = new JLabel("Eat the ball!");
93.         Title.setVisible(true);
94.         Title.setFont(font);
95.         Title.setBounds(50, 50, 500, 100);
96.         Title.setForeground(Color.decode("#FE4365"));
97.
98.         // 初始化血槽
99.         jProBar = new ProgressUI();
100.        jProBar.getBar().setSize(graphWidth, 50);
101.        jProBar.getBar().setLocation(0, 0);
102.        jProBar.getBar().setVisible(false);
103.
104.        con.add(Start);
105.        con.add(Exit);
106.        con.add(CurScore);
107.        con.add(MaxScore);
108.        con.add(GameLevel);
109.        con.add(Font_Start);
110.        con.add(Font_Exit);
111.        con.add(Title);
112.        con.add(jProBar.getBar());
113.
114.        // 处理鼠标移动事件
115.        jf.addMouseMotionListener(new MouseMotionListener() {
116.            @Override
117.            public void mouseDragged(MouseEvent e) {}
118.            @Override
119.            public void mouseMoved(MouseEvent e) {
```

```
120.                mouseX = e.getX()-4;
121.                mouseY = e.getY()-22;
122.            }
123.        });
124.        jf.setContentPane(con);
125.        jf.setVisible(true);
126.        jf.getContentPane().setBackground(Color.decode("#FFFFFF"));
127.        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
128.    }
129.
130.    public void updateCircle(Circle c){
131.        if(c != null){
132.            if(c.getID() == Game.CIRCLECOUNT) {
133.                willPaint[c.getID()] = c;
134.                jf.getContentPane().repaint();
135.            }else {
136.                willPaint[c.getID()] = c;
137.            }
138.        }
139.    }
140.
141.    public void printAllEnemiesCircles() {
142.        jf.getContentPane().repaint();
143.    }
144.
145.    public void clear(){
146.        for(int i = 0; i < willPaint.length; i++)
147.            willPaint[i] = null;
148.        jf.getContentPane().repaint();
149.    }
150.}
```

Circle.java

```
1.  package BallGame;
2.
3.  import java.util.Random;
4.
5.  public class Circle {
6.      protected int radius,x,y,dx,dy; // dx dy 代表下一步的间隔方向
7.      private int id;
8.      protected GUI gui;
9.      String color;
```

```
10.     public Circle(int X,int Y,int R,int id,String color,GUI gui,int dx,int dy){
11.         x = X; y = Y; radius = R;
12.         Random random = new Random();
13.         this.dx = random.nextBoolean() ? dx : -dx;
14.         this.dy = random.nextBoolean() ? dy : -dy;
15.         this.gui = gui;  this.id = id;   this.color = color;
16.         draw();
17.     }
18.     public Circle(int X,int Y,int R,int id,String color,GUI gui){
19.         x = X; y = Y; radius = R;
20.         this.gui = gui;
21.         this.id = id;
22.         this.color = color;
23.     }
24.     public int getX(){
25.         return x;
26.     }
27.     public int getY(){
28.         return y;
29.     }
30.     public int getR(){
31.         return radius;
32.     }
33.     public int getD(){
34.         return radius*2;
35.     }
36.     public int getID(){
37.         return id;
38.     }
39.     public void draw(){
40.         gui.updateCircle(this);
41.     }
42.     public void move(){ // 注意碰墙反弹
43.         x += dx;  y += dy;
44.         if(x+radius>gui.graphWidth || x-radius<0)   dx = -dx;
45.         if(y+radius>gui.graphHeight || y-radius<50)   dy = -dy;
46.         draw();
47.     }
48. }
```

PaintPanel.java

```
1.  package BallGame;
```

```
2.
3.  import java.awt.Color;
4.  import java.awt.Graphics;
5.  import java.awt.Image;
6.  import java.awt.Toolkit;
7.  import javax.swing.JPanel;
8.
9.  @SuppressWarnings("serial")
10. public class PaintPanel extends JPanel{
11.         private Circle[] willPaint;
12.         public PaintPanel(Circle[] willPaint){
13.             this.willPaint = willPaint;
14.         }
15.         @Override
16.         public void paint(Graphics g){
17.             super.paint(g);
18.             for(Circle CurPaint : willPaint){
19.                 int BoomsNum = 0, LifesNum = 0;
20.                 if(CurPaint != null){
21.                     if(CurPaint.color.equals("#EED5D2") && (BoomsNum <= 3)) {
22.                         Image image=Toolkit.getDefaultToolkit().getImage("./public/boom.png");
23.                         CurPaint.radius = 8;
24.                         g.drawImage(image, CurPaint.getX(), CurPaint.getY(), this);
25.                         BoomsNum++;
26.                         continue;
27.                     }else if(CurPaint.color.equals("#6A5ACD") && (LifesNum <= 3)) {
28.                         Image image=Toolkit.getDefaultToolkit().getImage("./public/life.png");
29.                         CurPaint.radius = 8;
30.                         g.drawImage(image, CurPaint.getX(), CurPaint.getY(), this);
31.                         LifesNum++;
32.                         continue;
33.                     }
34.                     g.setColor(Color.decode(CurPaint.color));
35.                     g.fillOval(CurPaint.getX()-CurPaint.getR(),
36.                             CurPaint.getY()-CurPaint.getR(),
37.                             CurPaint.getD(),CurPaint.getD());
38.                 }
39.             }
40.         }
41.     }
```

PlayerCircle.java

```java
package BallGame;

import BallGame.Circle;
import BallGame.GUI;

public class PlayerCircle extends Circle{
    private int Max;
    public PlayerCircle(int X,int Y,int R,int id,String color,GUI gui,int Maxsize){
        super(X,Y,R,id,color,gui);
        Max = Maxsize;
        draw();
    }
    public int getMax(){
        return Max;
    }
    public void resize(int plusSize){
        if(radius < Max){
            radius += plusSize;
            draw();
        }
    }
    public void move(){
        x = gui.mouseX;
        y = gui.mouseY;
        draw();
    }
}
```

ProgressUI.java

```java
package BallGame;

import java.awt.Color;
import javax.swing.JProgressBar;

public class ProgressUI {
    private JProgressBar Bar;
    private int val;

    ProgressUI() {
        this.Bar = new JProgressBar();
        this.Bar.setBorder(null);
    }
```

```
    ProgressUI(JProgressBar Bar) {
        this.Bar = Bar;
    }
    public int getValue() {
        return Bar.getValue();
    }
    public JProgressBar getBar() {
        return Bar;
    }
    public void setBar(JProgressBar Bar) {
        this.Bar = Bar;
    }
    public void addValue(int num) {
        val=this.Bar.getValue() + num;
        this.Bar.setValue(val);
        if(val<20)   this.Bar.setForeground(Color.RED);
        else if(val<40) this.Bar.setForeground(Color.YELLOW);
        else if(val<60) this.Bar.setForeground(Color.BLUE);
        else if(val<80) this.Bar.setForeground(Color.GREEN);
        else  this.Bar.setForeground(Color.CYAN);
        this.Bar.setValue(val);
    }
}
```

Utils.java

```
package BallGame;

import java.awt.FontFormatException;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.util.Random;
import java.util.Scanner;

public class Utils {
    // 工具函数类
    static String getRandomColor(Random r) {
        int count=(int) (r.nextInt(18)+1);
        String Color;
```

```
18.         switch (count) {
19.             case 1: Color = "#00FF00";break;
20.             case 2: Color = "#00FFFF";break;
21.             case 3: Color = "#00BFFF";break;
22.             case 4: Color = "#8A7CDA";break;
23.             case 5: Color = "#FF00FF";break;
24.             case 6: Color = "#FF0000";break;
25.             case 7: Color = "#FA663C";break;
26.             case 8: Color = "#B5B5B5";break;
27.             case 9: Color = "#BF487A";break;
28.             case 10:Color = "#90EE90";break;
29.             case 11:Color = "#C978E7";break;
30.             case 12:Color = "#EED5D2";break;
31.             case 13:Color = "#52EA41";break;
32.             case 14:Color = "#FF7F00";break;
33.             case 15:Color = "#8B658B";break;
34.             case 16:Color = "#7FFFD4";break;
35.             case 17:Color = "#6A5ACD";break;
36.             case 18:Color = "#9BCD9B";break;
37.             default:Color = "#8B0A50";break;
38.         }
39.         return Color;
40.     }
41.
42.     public static java.awt.Font getSelfDefinedFont(String filepath, int size){
43.         java.awt.Font font = null;
44.         File file = new File(filepath);
45.         try{
46.             font = java.awt.Font.createFont(java.awt.Font.TRUETYPE_FONT, file);
47.             font = font.deriveFont(java.awt.Font.PLAIN, size);
48.         }catch (FontFormatException e){
49.             return null;
50.         }catch (FileNotFoundException e){
51.             return null;
52.         }catch (IOException e){
53.             return null;
54.         }
55.         return font;
56.     }
57.
58.
59.     public static int readRecordFromFile(String filename) {
60.         int record = 0;
61.         try {
```

```
62.             Scanner in = new Scanner(new FileReader(filename));
63.             if(in.hasNext()) record = in.nextInt();
64.             in.close();
65.         } catch (FileNotFoundException e) {}
66.         return record;
67.     }
68.
69.     public static void writeRecordInFile(String filename, int record) {
70.         File file = new File(filename);
71.         Writer writer = null;
72.         StringBuilder outputString = new StringBuilder();
73.         try {
74.             outputString.append(record);
75.             writer = new FileWriter(file, false);
76.             writer.write(outputString.toString());
77.             writer.close();
78.         } catch (IOException e) {}
79.     }
80. }
```