

# 案例11：在星空背景下的小球冒泡

2022年7月12日 8:37

## 一、案例需求

1. 需求：创建图形化窗口，设置为可视化之后；在窗口上添加画布，绘制300颗位置随机的星星作为背景，另外再绘制一个粉色的小球，小球能够跟随鼠标不停的移动，当点击、按压鼠标粉色小球能够向上冒出橙色的小泡泡，如果持续按压则不停的冒出橙色的小泡泡
2. 分析：
  - a. 此案例就是结合了“满天星”和“冒泡小球”两个案例来实现即可；
  - b. 仅仅将两个案例的代码重构到一起，则画布类会非常的复杂；
  - c. 经过比较原先的代码会发现，其实星星和会冒泡的小球之间没有太大区别的，只是绘制的图形不相同；因此可以将这两个抽象成一个类，在该类上添加一个属性用于标识该对象是星星还是会冒泡的小球。
  - d. 其余代码参考冒泡小球与满天星即可

## 二、具体实现步骤以及代码

1. 创建窗口类以及画布类，并绘制一个会冒泡的大球，鼠标控制其移动；参考案例10即可
2. 创建小泡泡类Bubble类
3. 激活鼠标的时候，小球可以向上冒泡
4. 在画布中添加300颗星星作为背景：

- a. 类中添加属性：

```
// 背景的星星
private int [] xx,yy;
// 星星的个数
private int starCount;
// 随机数
private Random ran;
```

- b. 构造方法中初始化：

```
starCount = 300;
ran = new Random();
xx = new int[starCount];
yy = new int[starCount];
for (int i = 0; i < starCount; i++) {
    xx[i] = ran.nextInt(1024);
    yy[i] = ran.nextInt(768);
}
```

- c. paint方法中绘制：

```
// 绘制漫天星星背景
g.setColor(Color.YELLOW);
for (int i = 0; i < starCount; i++) {
    g.drawString("*", xx[i], yy[i]);
}
```

}

### 三、案例知识点总结

#### 1. 集合中的泛型：

- a. 将原先可以存储任意类型的集合约束为只能存储泛型所指定的集合；
- b. 如果一个集合未定义泛型则可以存储任意类型的元素（基本数据类型的值、引用类型的对象），但是在取值的时候无法直接遍历将其中的元素取出并赋值给某个变量
- c. 格式：例如：`Vector<Bubble> allBubbles = new Vector<Bubble>();`

## 案例12：泡泡大战

2022年7月12日 9:33

### 一、案例需求：

1. 需求：窗口、画布。在画布上先绘制不停落下的泡泡，出现泡泡的位置是随机的，并且能不停的落下；然后另外再绘制一个较大的小球，小球能够跟随鼠标光标位置的移动不停的移动，当单击、按压鼠标的时候小球可以向上冒出小泡泡；当落下的泡泡和小球冒出的小泡泡相遇时，两个泡泡均消失
2. 分析：此案例其实就是一个冒泡小球、下落的泡泡（类似雪花落下）、冒泡小球产生的泡泡和落下泡泡相遇时消失（类似碰撞的小球）的整合案例
  - a. 下落的泡泡和冒出的泡泡没有区别，都是一个实心圆，只是运动方向不同而已
  - b. 判断两个小泡泡发生碰撞： $(x_2-x_1)^2+(y_2-y_1)^2=(r_2+r_1)^2$

### 二、具体实现步骤以及代码：

1. 创建窗口和画布，将之前案例10当中的Bubble类复制到本案例的包下
2. 参考案例11完成在星空背景下小球会发射泡泡的功能
3. 完成下落的泡泡：

- a. 在原先的Bubble类中加上小球向下运动的方法：

- i. 类中添加常量：`public static final int DOWN = 5;`

- ii. move方法的switch中添加下落的case：

case DOWN:

y++;

break;

- b. 原先雪花下落是采用数组存储了雪花的坐标，此处可以考虑用数组来存储下落的泡泡，与发射的泡泡存在同样的问题，不知道应该为数组开辟多大的初始空间；所以此处同样采用集合的方式来存储下落的泡泡，在panel类中添加

// 定义存储下落的泡泡（敌机）的集合；

private Vector<Bubble> downBubbles;

并且在其构造方法中完成初始化：

// 初始化

downBubbles = new Vector<Bubble>();

- c. 构造方法：下落的泡泡是在游戏开始之后就立刻出现的，因此在构造方法中完成下落泡泡的创建，由于是在构造中来遍历创建下落的泡泡，此时集合中还没有泡泡对象的，要注意循环条件：

- i. 可以考虑设置为`i < downBubbles.capacity();`

- ii. 或者设置一个固定的下落泡泡的数量（类似星星的数量）：`private int dbCount;`

- iii. 并完成创建：

//敌机：下落泡泡初始化

dbCount = 10;

downBubbles = new Vector<Bubble>();

for (int i = 0; i < dbCount; i++) {

Bubble b = new Bubble();

b.setColor(Color.RED);

```

        b.setR(12);
        b.setOrientation(Bubble.DOWN);
        b.setX(ran.nextInt(1024));//[0,1024)
        b.setY(-ran.nextInt(768));//[ -768,0)
        downBubbles.add(b);

```

- d. paint方法：绘制出敌机的下落泡泡：

```

//绘制下落的泡泡
for (int i = 0; i < dbCount; i++) {
    Bubble dwBubble = downBubbles.get(i);
    dwBubble.draw(g);
}

```

- e. run方法中：添加敌机的运动方法：

```

//敌机：下落泡泡的移动
for (int i = 0; i < downBubbles.size(); i++) {
    Bubble b = downBubbles.get(i);
    b.move();
}

```

- f. 运行程序之后，发现的第一个问题：下落的泡泡落了一会儿之后就停下了，原因是因为在Bubble类中运动方向是DOWN时y++，当下落的泡泡的y都大于了窗口的高度之后，泡泡运动出去了窗口所能呈现的范围，所以想实现泡泡一直下落，就需要把Bubble类中DOWN方向上的y++到超出边框之后重置：

case DOWN:

```

//判断下落是否超出了窗口高度，超出则重置y为0
if(y > 768) {
    y = 0;
}else {
    y++;
}
break;

```

- g. 第二个问题：子弹泡泡和下落敌机泡泡目前没有发生碰撞；

需要实现碰撞并消失，开发一个判断两种泡泡是否发生碰撞的方法，根据之前的分析可知，如果两个泡泡的原型举例等于他们的半径的和，就说明两个泡泡发生了碰撞；在BallBattlePanel中添加isHit方法：

```

/*
 * 判断小球发射的泡泡是否击中下落的泡泡
 * ub：upBubble向上的泡泡
 * db：downBubble向下的泡泡
 */
public boolean isHit(Bubble ub,Bubble db) {
    boolean isHit = false;
//    计算两个泡泡ub\db圆心之间x轴和y轴的距离
    int xDistance = (ub.getX()+ub.getR())-(db.getX()+db.getR());
    int yDistance = (ub.getY()+ub.getR())-(db.getY()+db.getR());

```

```

//      计算两个泡泡圆心之间的距离的平方和
int xyDis = xDistance*xDistance + yDistance*yDistance;
//      计算两个泡泡半径的和的平方
int rDis = (ub.getR()+db.getR())*(ub.getR()+db.getR());
//      根据勾股定理
if(xyDis <= rDis) {
    isHit = true;
}
return isHit;

```

接着在run方法的while(true)中调用此方法，判断即可

```

//      判断小球发射的泡泡是否击中下落的泡泡
for (int i = 0; i < allBubbles.size(); i++) {
//      遍历取出发射的泡泡
    Bubble ub = allBubbles.elementAt(i);
    for (int j = 0; j < downBubbles.size(); j++) {
//      遍历取出下落的泡泡
        Bubble db = downBubbles.get(j);
//      调用判断的方法：让两个泡泡消失
        if(isHit(ub,db)) {
//      发射的泡泡：从集合中移除
            allBubbles.remove(i);
//      下落的泡泡：重置y轴
            db.setY(-db.getY());
            j--;
            i--;
            break;
        }
    }
}

```

4. 至此，案例12泡泡大战完成！

# 案例13：雷霆战机

2022年7月12日 16:31

## 一、案例需求

1. 游戏说明：从屏幕上方不断出现敌机，玩家控制一架飞机，单机鼠标可以使得飞机发射子弹，玩家按住鼠标不放的话，那么飞机将不断的发射子弹，当子弹射中敌机之后，敌机将消失；玩家操作的飞机默认有100滴血，承受一次伤害之后减少10滴血，血量为0时，游戏结束。
2. 案例分析：
  - a. 此案例其实就是上一个“泡泡大战”案例的延伸，将小球搞成为一架飞机、小球发射的泡泡改为了子弹、下落的泡泡改为了敌机，并且将绘制小球、泡泡、背景雪花等等改为绘制相应的图片；
  - b. 当英雄飞机发射的子弹碰到敌机的时候，敌机爆炸，己方得分增加
  - c. 当敌机碰到英雄飞机，英雄飞机血量减少；默认初始血量为100滴血，被一架敌机碰到一次掉10滴血，当英雄的剩余血量为0时，GG！

## 二、具体实现步骤以及代码：

1. 经过分析可知，本案例需要设计四个类：英雄战机类、敌机&子弹类、窗口和画布类
2. 窗口和画布类：设置基本属性以及应当实现的接口&方法
3. 英雄战机类HeroPlane：

- ★a. 将课堂资料的img文件夹拷贝到当前工程目录下
  - b. 静态常量：初始血量、战机图片
  - c. 成员变量属性：x\y坐标、图片宽高、剩余生命值、总分、击中一次得分、所在面板
- ```
// 定义英雄战机的总生命值
public final static int ALL_BLOOD = 100;

// 定义英雄战机图片
public final static Image hero = new ImageIcon("img/hero.png").getImage();

// 定义英雄战机的坐标以及图片宽高
private int x, y, size;

// 定义英雄战机的剩余生命值
private int blood;

// 定义英雄战机的总得分
private int score;

// 定义英雄战机击中一架敌机的得分
private int oneScore;

// 定义英雄战机所在画布
private JPanel panel;

// 记得要生成setter&getter的方法！！
```
- d. draw方法，完成英雄战机图片的绘制以及英雄战机血量和得分绘制：



```
// draw方法，完成绘制
```

```
public void draw(Graphics g) {  
    //      战机绘制  
        g.drawImage(hero, x, y, size, size, panel);  
    //      战机的生命值绘制  
        Font font = new Font("宋体",Font.BOLD,12);  
        g.setFont(font);  
        g.setColor(Color.RED);  
    //      显示血量的title  
        g.drawString("剩余血量", 0, 25);  
    //      显示总血量所在的矩形框  
        g.setColor(Color.WHITE);  
        g.fillRect(60, 16, ALL_BLOOD, 10);  
    //      显示剩余血量所在的矩形框  
        g.setColor(Color.RED);  
        g.fillRect(60, 15, this.blood, 10);  
    //      显示得分  
        g.drawString("得分 : "+this.score, 0, 40);  
}
```