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：$(x2-x1)^2+(y2-y1)^2=(r2+r1)^2$

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类中添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！