

课程设计

**课程名称： Java程序课程设计**

**设计题目： 飞机大战游戏**

**学 院： 信息工程与自动化学院**

**专业年级： 计科173**

**学 号： 201710405328**

**学生姓名： 方子希**

**指导教师： 尚振宏**

**日 期： 2019-9-1**

目录

[摘要 3](#_Toc18520755)

[功能说明 3](#_Toc18520756)

[设计 3](#_Toc18520757)

[关键代码 4](#_Toc18520758)

[测试及测试结果 22](#_Toc18520759)

[总结 27](#_Toc18520760)

### 摘要

利用Java的相关只是设计完成“雷霆战机”游戏。

### 功能说明

1. 首先是点击游戏开始，需要有游戏开始界面
2. 其次是消灭一个敌机要加分，消灭大敌机加的分要更加的多
3. 再然后是消灭蜜蜂加一倍的火力或者加一条生命，如果是宝箱就直接四倍火力加两条命
4. 当英雄机碰到其他任何飞行物（包括宝箱或蜜蜂），都掉一条命
5. 火力5秒钟减少一条子弹
6. 当生命值为0的时候游戏结束
7. 再次点击可以重新开始游戏

### 设计

1. 拿到素材后首先对界面的大小进行设置，界面需要显示的素材有三张图片，开始（start）、结束（pause）、背景（background），、、、然后当然选择了400\*654作为界面的长宽，因为如果用背景图480\*852的长宽作为界面的宽度，那么会发生暂停界面出现的时候，边缘会露出背景图片。
2. 首先想清楚有哪些对象、有蜜蜂、大宝箱、敌机、大敌机、子弹、英雄机、然后呢蜜蜂和大宝箱可以作为一类，都是获取奖励的，敌机和大敌机可以作为一类、都是获取分数的、所以就想到了接口，给前一种类型，写一个接口去定义一个抽象函数，用来返回奖励类型，进行在子类中实现，后一种类型、写一个接口定义一个抽象函数，用来返回得分，在子类中实现，然后，因为当子弹碰到蜜蜂、大宝箱、敌机或者是大敌机，都会被消灭，而英雄机碰到蜜蜂、大宝箱、敌机或者是大敌机，都会掉一条命，而子弹碰上敌方单位也会消失，英雄机也需要有以下的功能，所以这所有的类可以给他们建立一个飞行物的父类，因为他们都同样有x，y坐标、有速度、有图片、并且有图片的宽高、并且要为他们定义一个outOfBorder()是否出界的方法、和一个step()每下一秒移动改变位置的方法。
3. 在老师最初的要求上我加入了大宝箱和大敌机还有大宝箱和蜜蜂出来的方向我希望他随机，所以写了一个随机方法、所以在大宝箱和蜜蜂，除了实现父类的方法和接口外，添加了初始运动方向的随机变量，就没有了。
4. 而大敌机和敌机仅仅是实现父类的方法和接口外就没有添加额外的方法了
5. 子弹实现了父类的方法后就没有进行添加了
6. 英雄机不仅实现了父类的方法和接口后，需要有鼠标的位置值、还需要有鼠标移动的方法moveTo(int x, int y)、需要有判断撞到敌机的方法hitEnemy(FlyingObject flyingObject)、需要还需要两个set和get函数去获取私有变量的火力和生命值。
7. 最终的便是带有main函数的主类、主类就需要获取图片、需要使用Java Swing进行显示游戏、需要监听鼠标、需要有自动生成敌人对象、自动生成子弹、自动进行下一次屏幕刷新后，所有飞行物飞行的下一个位置的计算、需要一个游戏规则（进行子弹与敌方单位碰撞，敌方单位与英雄机碰撞会发生什么）的方法。

### 关键代码

#### ShootGame主类

1. **package** com.AirplaneWar;
3. **import** javax.imageio.ImageIO;
4. **import** javax.swing.\*;
5. **import** java.awt.\*;
6. **import** java.awt.event.MouseAdapter;
7. **import** java.awt.event.MouseEvent;
8. **import** java.awt.image.BufferedImage;
9. **import** java.util.\*;
10. **import** java.util.Timer;
12. **public** **class** ShootGame **extends** JPanel {
13. **public** **static** BufferedImage hero0;
14. **public** **static** BufferedImage hero1;
15. **public** **static** BufferedImage bee;
16. **public** **static** BufferedImage airplane;
17. **public** **static** BufferedImage pause;
18. **public** **static** BufferedImage start;
19. **public** **static** BufferedImage gameover;
20. **public** **static** BufferedImage bullet;
21. **public** **static** BufferedImage background;
22. **public** **static** BufferedImage treasure;
23. **public** **static** BufferedImage BigBoss;

26. **public** **static** **int** score;
27. **public** **static** **final** **int** START = 0;
28. **public** **static** **final** **int** RUNNING = 1;
29. **public** **static** **final** **int** PAUSE = 2;
30. **public** **static** **final** **int** GAMEOVER = 3;
31. **public** **static** **int** state = START;

34. **static** {
36. **try** {
37. hero0 = ImageIO.read(ShootGame.**class**.getResource("/images/hero0.png"));
38. hero1 = ImageIO.read(ShootGame.**class**.getResource("/images/hero1.png"));
39. bee = ImageIO.read(ShootGame.**class**.getResource("/images/bee.png"));
40. airplane = ImageIO.read(ShootGame.**class**.getResource("/images/airplane.png"));
41. pause = ImageIO.read(ShootGame.**class**.getResource("/images/pause.png"));
42. start = ImageIO.read(ShootGame.**class**.getResource("/images/start.png"));
43. gameover = ImageIO.read(ShootGame.**class**.getResource("/images/gameover.png"));
44. bullet = ImageIO.read(ShootGame.**class**.getResource("/images/bullet.png"));
45. background = ImageIO.read(ShootGame.**class**.getResource("/images/background.png"));
46. treasure = ImageIO.read(ShootGame.**class**.getResource("/images/treasure.png"));
47. BigBoss = ImageIO.read(ShootGame.**class**.getResource("/images/BigBoss.png"));
49. } **catch** (Exception e) {
50. e.printStackTrace();
51. }
53. }
55. **public** **static** **final** **int** WIDTH = start.getWidth();
56. **public** **static** **final** **int** HEIGHT = start.getHeight();
57. **private** Heroplane heroplane = **new** Heroplane(start.getWidth() / 2, start.getHeight());
58. //        private FlyingObject[] flyings = {};
59. **private** LinkedList flyings = **new** LinkedList();
60. //        private Bullet[] bullets = {};
61. **private** LinkedList bullets = **new** LinkedList();

64. **public** **void** paint(Graphics g) {
65. g.drawImage(background, 0, 0, **null**);
66. g.drawImage(heroplane.image, heroplane.x, heroplane.y, **null**);
67. //        for (int i = 0; i < flyings.length; i++) {
68. //            g.drawImage(flyings[i].image, flyings[i].x, flyings[i].y, null);
69. //        }
70. **for** (Iterator iter = flyings.iterator(); iter.hasNext(); ) {
71. FlyingObject f = (FlyingObject) iter.next();
72. g.drawImage(f.image, f.x, f.y, **null**);
73. }
74. //        for (int i = 0; i < bullets.length; i++) {
75. //            g.drawImage(bullets[i].image, bullets[i].x, bullets[i].y, null);
76. //        }
77. **for** (Iterator iter = bullets.iterator(); iter.hasNext(); ) {
78. FlyingObject f = (FlyingObject) iter.next();
79. g.drawImage(f.image, f.x, f.y, **null**);
80. }
81. //画火力画生命
82. g.drawString("得分：" + score + "  " + "生命值：" + heroplane.getLife() + "  " + "火力：" + heroplane.getDoubleFire(), 10, 20);
83. **switch** (state) {
84. **case** START:
85. g.drawImage(start, 0, 0, **null**);
86. **break**;
87. //            case RUNNING:
88. //                break;
89. **case** PAUSE:
90. g.drawImage(pause, 0, 0, **null**);
91. **break**;
92. **case** GAMEOVER:
93. g.drawImage(gameover, 0, 0, **null**);
94. **break**;
95. **default**:
96. **break**;
97. }
99. }//显示
101. //    int refresh = 10;//调整整体刷新率（1~10）,而英雄机的刷新率要在英雄机类中改变
103. **int** wc = 0;//调整敌机或蜜蜂的生成速率
104. //    int cc = 150; //控制敌机生成速率
106. **public** **void** enermyMaker() {
107. wc++;
108. //        if (cc >= 10)
109. //            cc--;
110. //        if (wc % (100 / refresh) == 0) {
111. **if** (wc % (30) == 0) {
112. FlyingObject f = nextOne();
113. flyings.add(f);
114. //            flyings = Arrays.copyOf(flyings, flyings.length + 1);
115. //            flyings[flyings.length - 1] = f;
116. //            System.out.println(flyings.length);
117. }
118. }//生成敌机
120. **public** FlyingObject nextOne() {
121. Random random = **new** Random();
122. **int** type = random.nextInt(100);
123. **if** (0 <= type && type < 5) {
124. **return** **new** BigEnemyPlane();
125. } **else** **if** (5 <= type && type < 7) {
126. **return** **new** Treasure();
127. } **else** **if** (7 <= type && type < 10) {
128. **return** **new** Bee();
129. } **else** {
130. **return** **new** EnemyPlane();
131. }
132. }//生成下一个敌机单位
134. **int** wk = 0;//调整每次生成子弹的速率
136. **public** **void** bulletMaker() {
137. wk++;
138. //        if (wk % (300 / refresh) == 0) {
139. **if** (wk % (30) == 0) {
140. Bullet[] bs = nextBullet();
141. //            bullets = Arrays.copyOf(bullets, bullets.length + bs.length);
142. //            for (int i = 0; i < bs.length; i++) {
143. //                bullets[bullets.length - bs.length + i] = bs[i];
144. //            }
145. **for** (**int** i = 0; i < bs.length; i++)
146. bullets.add(bs[i]);
148. }
149. }//生成子弹
151. **public** Bullet[] nextBullet() {
152. **int** n = heroplane.getDoubleFire();//几条子弹
153. //        System.out.println(n);
155. Bullet[] bs = **new** Bullet[n];
156. **for** (**int** i = 0; i < n; i++) {
157. bs[i] = **new** Bullet(heroplane.x - 3 + (i + 1) \* (heroplane.width / (1 + n)), heroplane.y);
158. }
159. **return** bs;
160. }//生成下一个子弹
162. **int** ww = 0;//调整每次运动的刷新率
164. **public** **void** stepAction() {
165. heroplane.step();
166. ww++;
167. **if** (heroplane.outOfBorder()) {
168. //Pause
169. }
170. //        if (ww % (5000 / refresh) == 0) {
171. **if** (ww % (500) == 0) {
172. **if** (heroplane.getDoubleFire() > 1)
173. heroplane.setDoubleFire(heroplane.getDoubleFire() - 1);
174. }
175. //        if (ww % (10 / refresh) == 0) {
176. **if** (ww % (1) == 0) {
177. //            FlyingObject[] f = new FlyingObject[flyings.length];
178. //            int fcont = 0;
179. //            Bullet[] b = new Bullet[bullets.length];
180. //            int bcont = 0;
181. //            for (int i = 0; i < flyings.size(); i++) {
182. //                flyings[i].step();
183. //                if (!flyings[i].outOfBorder()) {
184. //                    //如果没有越界就复制
185. //                    f[fcont] = flyings[i];
186. //                    fcont++;
187. //                }
188. **for** (Iterator iter = flyings.iterator(); iter.hasNext(); ) {
189. FlyingObject f = (FlyingObject) iter.next();
190. f.step();
191. **if** (f.outOfBorder())
192. iter.remove();
194. }
195. //            for (int i = 0; i < bullets.length; i++) {
196. //                bullets[i].step();
197. //                if (!bullets[i].outOfBorder()) {
198. //                    //如果没有越界就复制
199. //                    b[bcont] = bullets[i];
200. //                    bcont++;
201. //                }
202. //            }
203. //            flyings = Arrays.copyOf(f, fcont);
204. //            bullets = Arrays.copyOf(b, bcont);
205. **for** (Iterator iter = bullets.iterator(); iter.hasNext(); ) {
206. Bullet f = (Bullet) iter.next();
207. f.step();
208. **if** (f.outOfBorder())
209. iter.remove();
211. }
212. }
213. }//所有单位下一步
215. **public** **void** gameRule() {
216. **for** (Iterator iter1 = bullets.iterator(); iter1.hasNext(); ) {
217. Bullet b = (Bullet) iter1.next();
218. **for** (Iterator iter2 = flyings.iterator(); iter2.hasNext(); ) {
219. //                Bullet b = (Bullet) iter1.next();
220. FlyingObject f = (FlyingObject) iter2.next();
221. **if** (b.hit(f)) {
222. //撞击成功
223. **if** (f **instanceof** EnemyPlane) {
224. //撞上了敌机
225. EnemyPlane enemyPlane = (EnemyPlane) f;
226. score += enemyPlane.getScore();//返回得分
227. }
228. **if** (f **instanceof** BigEnemyPlane) {
229. //撞上了敌机
230. BigEnemyPlane bigenemyPlane = (BigEnemyPlane) f;
231. score += bigenemyPlane.getScore();//返回得分
232. }
233. **if** (f **instanceof** Bee) {
234. //撞上了蜜蜂
235. Bee bee = (Bee) f;
236. **if** (bee.getAwardType() == 1) {//如果是加火力
237. //                        if(true){
238. //                            if (heroplane.getDoubleFire() <= 4) {
239. heroplane.setDoubleFire(2 \* heroplane.getDoubleFire());//火力\*2
240. //                            }
241. } **else** **if** (bee.getAwardType() == 0) {//如果是加生命
242. heroplane.setLife(heroplane.getLife() + 1);
243. //                            System.out.println(heroplane.getLife());
244. }
245. }
246. **if** (f **instanceof** Treasure) {
247. Treasure treasure = (Treasure) f;
248. heroplane.setDoubleFire(4 \* heroplane.getDoubleFire());
249. heroplane.setLife(heroplane.getLife() + 2);
250. }
251. //                    for (int k = i; k < bullets.length - 1; k++) {//子弹清除
252. //                        bullets[k] = bullets[k + 1];
253. //                    }
254. //                    bullets = Arrays.copyOf(bullets, bullets.length - 1);
255. //                    for (int k = j; k < flyings.length - 1; k++) {//飞行物清楚
256. //                        flyings[k] = flyings[k + 1];
257. //                    }
258. //                    flyings = Arrays.copyOf(flyings, flyings.length - 1);
259. //                    i--;
260. //                    j--;
261. iter1.remove();
262. iter2.remove();
264. }
265. }
266. }
267. **for** (Iterator iter = flyings.iterator(); iter.hasNext(); ) {
268. FlyingObject f = (FlyingObject) iter.next();
269. **if** (heroplane.hitEnemy(f)) {
270. //Hit!
271. //                System.out.println("Game Over!");
272. heroplane.setLife(heroplane.getLife() - 1);
273. heroplane.setDoubleFire(1);
274. //                for (int k = i; k < flyings.length - 1; k++) {//飞行物清楚
275. //                    flyings[k] = flyings[k + 1];
276. //                }
277. //                flyings = Arrays.copyOf(flyings, flyings.length - 1);
278. //                i--;
279. iter.remove();
280. }
281. }
282. }
284. **public** **void** checkGameOverAction() {
285. **if** (isGameOver()) {
286. state = GAMEOVER;
287. }
288. }
290. **public** **boolean** isGameOver() {
291. **if** (heroplane.getLife() <= 0)
292. **return** **true**;
293. **else**
294. **return** **false**;
295. }
297. **public** **void** action() {
298. //这里是鼠标监听事件
299. MouseAdapter l = **new** MouseAdapter() {
301. @Override
302. **public** **void** mouseEntered(MouseEvent e) {
303. //                super.mouseEntered(e);
304. **if** (state == PAUSE) {
305. state = RUNNING;
306. }
307. }
309. @Override
310. **public** **void** mouseExited(MouseEvent e) {
311. //                super.mouseExited(e);
312. **if** (state == RUNNING) {
313. state = PAUSE;
314. }
315. }
317. @Override
318. **public** **void** mouseClicked(MouseEvent e) {
319. //                super.mouseClicked(e);
320. **if** (state == START) {
321. state = RUNNING;
322. }
323. **if** (state == GAMEOVER) {
324. //清除上一局数据
325. heroplane = **new** Heroplane(start.getWidth() / 2, start.getHeight());
326. //                    bullets = new Bullet[0];
327. bullets.clear();
328. //                    flyings = new FlyingObject[0];
329. flyings.clear();
330. score = 0;
331. state = RUNNING;
332. }
333. }
335. @Override
336. **public** **void** mouseMoved(MouseEvent e) {
337. //                super.mouseMoved(e);
338. **if** (state == RUNNING) {
339. **int** x = e.getX();
340. **int** y = e.getY();
341. heroplane.moveTo(x, y);
342. }
343. }
344. };
346. **this**.addMouseListener(l);//侦听鼠标点击操作
347. **this**.addMouseMotionListener(l);//侦听鼠标滑动操作
348. //这里是定时事件
349. Timer time = **new** Timer();
350. time.schedule(**new** TimerTask() {
351. @Override
352. **public** **void** run() {
353. **if** (state == RUNNING) {
354. enermyMaker();//生产敌人对象并添加进数组
355. bulletMaker();//子弹产生器
356. stepAction();
357. gameRule();
358. checkGameOverAction();
359. }
360. repaint();
361. }
362. //        }, refresh, refresh);//屏幕的刷新率
363. }, 10, 10);//屏幕的刷新率
364. }//所有单位的刷新
366. **public** **static** **void** main(String[] args) {
367. JFrame frame = **new** JFrame("飞机大战");//创建框架
368. ShootGame s = **new** ShootGame();//创建画板
369. frame.add(s);
370. frame.setSize(WIDTH, HEIGHT);
371. frame.setAlwaysOnTop(**true**);
372. frame.setDefaultCloseOperation(frame.EXIT\_ON\_CLOSE);
373. frame.setLocationRelativeTo(**null**);
374. frame.setVisible(**true**);
375. s.action();//调用定时器进行添加敌人or蜜蜂
376. }
378. }

#### FlyingObject 飞行物类（所有飞行物的父类）

1. **package** com.AirplaneWar;
3. **import** java.awt.image.BufferedImage;
5. **public** **abstract** **class** FlyingObject {
6. **protected** BufferedImage image;    //图像
8. **protected** **int** x;  //x坐标
9. **protected** **int** y;  //y坐标
11. **protected** **int** width;  //图片宽
12. **protected** **int** height; //图片高
14. **public** **abstract** **void** step();
15. //飞行物的飞行方法
17. **public** **abstract** **boolean** outOfBorder();
18. //判断是否越界的方法
19. }

#### Award接口（奖励）

1. **package** com.AirplaneWar;
3. **public** **interface** Award {//用于整合大奖励和小奖励
4. //    public int LIFE = 0;
5. //    public int DOUBLEFIRE = 1;
6. **public** **int** getAwardType();
7. }

#### Bee 蜜蜂类（奖励）

1. **package** com.AirplaneWar;
3. **import** java.util.Random;
5. **public** **class** Bee **extends** FlyingObject **implements** Award {
6. **private** **int** xspeed = 2;     //蜜蜂x方向的速度
7. **private** **int** yspeed = 4;     //蜜蜂y方向的速度
8. **private** **int** awardType;      //决定被击中后给英雄机的奖励类型，0--加命，1--加火力
10. **public** Bee() {
11. image = ShootGame.bee;
12. width = image.getWidth();
13. height = image.getHeight();
14. Random r = **new** Random();
15. x = r.nextInt(ShootGame.WIDTH - width + 1);
16. y = -height;
17. awardType = r.nextInt(2);
18. //      定义一个出蜜蜂 随机向左或者向右飞行的算法
19. Random random = **new** Random();
20. **int** temp = random.nextInt(2);
21. **if** (temp == 0) {
22. xspeed = Math.abs(xspeed);
23. temp = 3;
24. } **else** **if** (temp == 1) {
25. xspeed = 0 - Math.abs(xspeed);
26. temp = 3;
27. }
28. }
30. @Override
31. **public** **int** getAwardType() {
32. //        return 1;
33. //        return 0;
34. **return** awardType;
35. }

38. @Override
39. **public** **void** step() {
41. **if** (x <= 0) {
42. xspeed = Math.abs(xspeed);
43. }
44. **if** (x >= ShootGame.WIDTH - width) {
45. xspeed = -Math.abs(xspeed);
46. }
47. y = y + yspeed;
48. x = x + xspeed;
50. }
52. @Override
53. **public** **boolean** outOfBorder() {
54. **if** (y >= ShootGame.HEIGHT) {
55. //            System.out.println("小蜜蜂越界");
56. **return** **true**;
57. } **else**
58. **return** **false**;
60. }
61. }

#### Treasure宝箱类（奖励）

1. **package** com.AirplaneWar;
3. **import** java.util.Random;
5. **public** **class** Treasure **extends** FlyingObject **implements** Award {
6. **private** **int** xspeed = 4;     //蜜蜂x方向的速度
7. **private** **int** yspeed = 4;     //蜜蜂y方向的速度
8. **private** **int** awardType;      //决定被击中后给英雄机的奖励类型，0--加命，1--加火力
10. **public** Treasure() {
11. image = ShootGame.treasure;
12. width = image.getWidth();
13. height = image.getHeight();
14. Random r = **new** Random();
15. x = r.nextInt(ShootGame.WIDTH - width + 1);
16. y = -height;
17. awardType = r.nextInt(2);
18. //      定义一个出蜜蜂 随机向左或者向右飞行的算法
19. Random random = **new** Random();
20. **int** temp = random.nextInt(2);
21. **if** (temp == 0) {
22. xspeed = Math.abs(xspeed);
23. temp = 3;
24. } **else** **if** (temp == 1) {
25. xspeed = 0 - Math.abs(xspeed);
26. temp = 3;
27. }
28. }
30. @Override
31. **public** **int** getAwardType() {
32. **return** awardType;
33. }
35. @Override
36. **public** **void** step() {

39. **if** (x <= 0) {
40. xspeed = Math.abs(xspeed);
41. }
42. **if** (x >= ShootGame.WIDTH - width) {
43. xspeed = -Math.abs(xspeed);
44. }
45. y = y + yspeed;
46. x = x + xspeed;

49. }
51. @Override
52. **public** **boolean** outOfBorder() {
53. **if** (y >= ShootGame.HEIGHT) {
54. //            System.out.println("小蜜蜂越界");
55. **return** **true**;
56. } **else**
57. **return** **false**;
59. }
60. }

#### Enemy接口（敌方单位）

1. **package** com.AirplaneWar;

4. **public** **interface** Enemy {///用于整合之后的大敌机和小敌机的得分
5. **public** **int** getScore();
6. }

#### EnemyPlane敌机类（敌方单位）

1. **package** com.AirplaneWar;
3. **import** java.util.Random;
5. **public** **class** EnemyPlane **extends** FlyingObject **implements** Enemy {
6. **private** **int** speed = 3;      //敌机的速度
8. **public** EnemyPlane() {
9. image = ShootGame.airplane;
10. width = image.getWidth();
11. height = image.getHeight();
12. Random r = **new** Random();
13. x = r.nextInt(ShootGame.WIDTH - width + 1);
14. y = -height;
15. }
17. @Override
18. **public** **void** step() {
19. y = y + speed;
20. }
22. @Override
23. **public** **boolean** outOfBorder() {
24. **if** (y >= ShootGame.HEIGHT) {
25. //            System.out.println("敌机越界");
26. **return** **true**;
27. } **else**
28. **return** **false**;
30. }
32. @Override
33. **public** **int** getScore() {
34. **return** 10;
35. }
36. }

#### BigEnemyPlane大敌机类（敌方单位）

1. **package** com.AirplaneWar;
3. **import** java.util.Random;
5. **public** **class** BigEnemyPlane **extends** FlyingObject **implements** Enemy {
6. **private** **int** speed = 6;
8. **public** BigEnemyPlane() {
9. image = ShootGame.BigBoss;
10. width = image.getWidth();
11. height = image.getHeight();
12. Random r = **new** Random();
13. x = r.nextInt(ShootGame.WIDTH - width + 1);
14. y = -height;
15. }
17. @Override
18. **public** **int** getScore() {
19. **return** 50;
20. }
22. @Override
23. **public** **void** step() {
24. y = y + speed;
26. }
28. @Override
29. **public** **boolean** outOfBorder() {
30. **if** (y >= ShootGame.HEIGHT) {
31. //            System.out.println("敌机越界");
32. **return** **true**;
33. } **else**
34. **return** **false**;
35. }
36. }

#### Bullet子弹类

1. **package** com.AirplaneWar;
3. **public** **class** Bullet **extends** FlyingObject {
4. **private** **int** speed = 3;      //子弹的速度
6. **public** Bullet(**int** x, **int** y) {
7. image = ShootGame.bullet;
8. image = ShootGame.bullet;
9. width = image.getWidth();
10. height = image.getHeight();
11. **this**.x = x;
12. **this**.y = y;
13. }
15. @Override
16. **public** **void** step() {
17. y = y - speed;
18. }
20. @Override
21. **public** **boolean** outOfBorder() {
22. **if** (y <= -14) {
23. //            System.out.println("子弹越界");
24. **return** **true**;
25. } **else**
26. **return** **false**;
28. }
30. **public** **boolean** hit(FlyingObject flyingObject) {//传入敌方单位坐标
31. **if** (flyingObject.x <= x && x <= (flyingObject.x + flyingObject.width) && flyingObject.y <= y && y <= (flyingObject.y + flyingObject.height)) {
32. **return** **true**;
33. }**else**{
34. **return** **false**;
35. }
36. }
37. }

#### Heroplane英雄机类

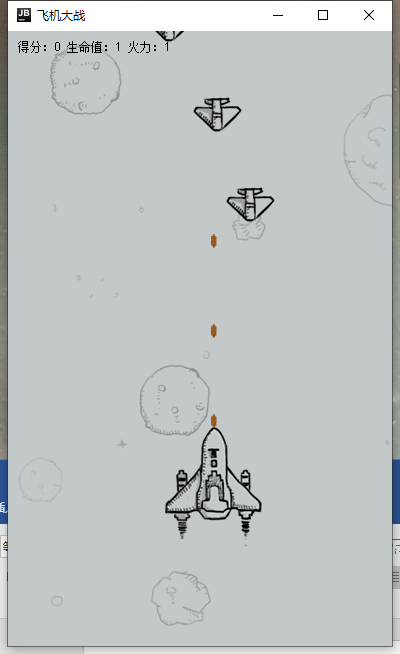
1. **package** com.AirplaneWar;
3. **public** **class** Heroplane **extends** FlyingObject {
4. **private** **int** speed;      //英雄机的速度
5. **private** **int** life = 1;       //生命值
6. **private** **int** doubleFire = 1;     //
7. **private** **int** wk = 0;
8. **private** **int** mouseX;
9. **private** **int** mouseY;
11. **public** Heroplane(**int** x, **int** y) {
12. **this**.x = x;
13. **this**.y = y;
14. image = ShootGame.hero0;
15. width = image.getWidth();
16. height = image.getHeight();
17. }
19. @Override
20. **public** **void** step() {
21. wk++;
22. **if** ((wk / 100) % 2 == 0) {
23. image = ShootGame.hero1;
24. } **else** {
25. image = ShootGame.hero0;
26. }
27. }
29. @Override
30. **public** **boolean** outOfBorder() {
31. mouseX = x + (image.getWidth() / 2);
32. mouseY = y + (image.getHeight() / 2);
33. **if** (mouseX <= 0 || mouseY <= 0 || mouseX >= ShootGame.WIDTH || mouseY >= ShootGame.HEIGHT) {
34. //            System.out.println("英雄机越界！");
35. **return** **true**;
36. } **else**
37. **return** **false**;
38. }
40. **public** **int** getDoubleFire() {
41. **return** doubleFire;
42. }
44. **public** **void** setDoubleFire(**int** doubleFire) {
45. **if** (doubleFire < 8) {
46. **this**.doubleFire = doubleFire;
47. }
48. }
50. **public** **void** moveTo(**int** x, **int** y) {
51. **this**.x = x - (image.getWidth() / 2);
52. **this**.y = y - (image.getHeight() / 2);
53. }
55. **public** **boolean** hitEnemy(FlyingObject flyingObject) {
56. **if** (**this**.x <= flyingObject.x + flyingObject.width && **this**.x + **this**.width >= flyingObject.x && **this**.y <= flyingObject.y + flyingObject.height && **this**.y + **this**.height >= flyingObject.y)
57. **return** **true**;
58. **else**
59. **return** **false**;
60. }
62. **public** **int** getLife() {
63. **return** life;
64. }
66. **public** **void** setLife(**int** life) {
67. **this**.life = life;
68. }
69. }

### 测试及测试结果

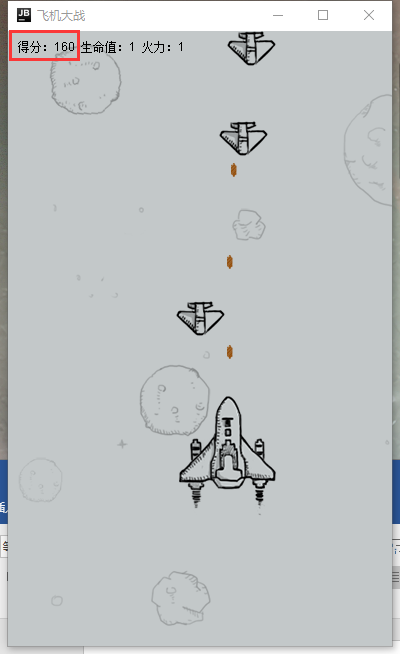
1. 单击点击开始、需要有开始界面



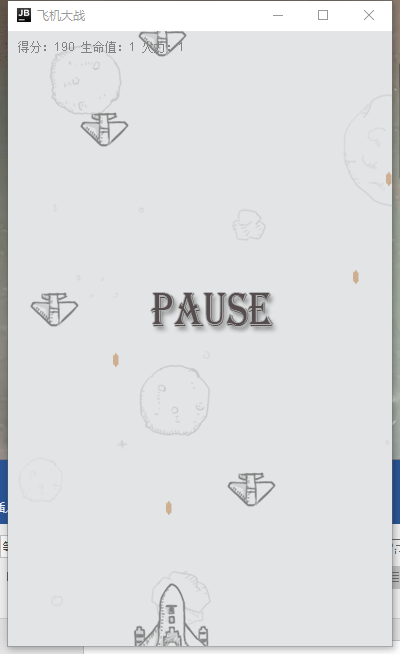
1. 正常运行



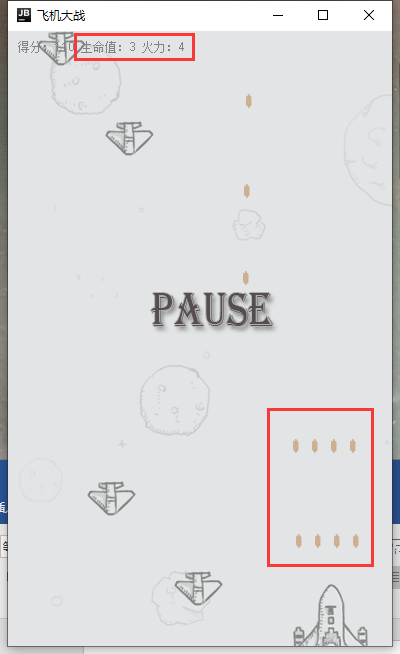
1. 消灭敌机or大敌机会加分



1. 鼠标移出会暂停



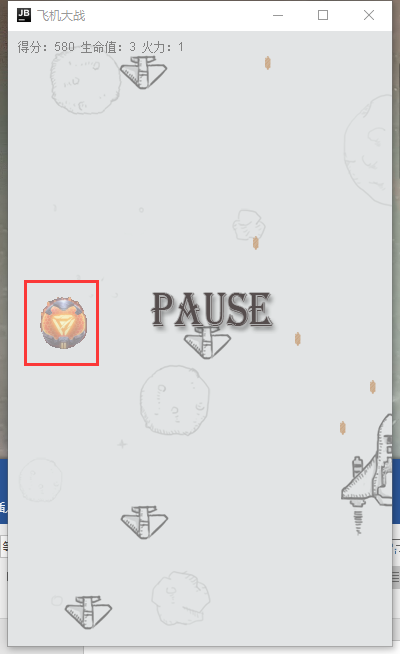
1. 消灭奖励物品 加火力或者是生命(这一次是消灭了宝箱，所以生命+2，火力×4)



1. 验证生成的单位（蜜蜂）



1. 验证生成的单位（宝箱）



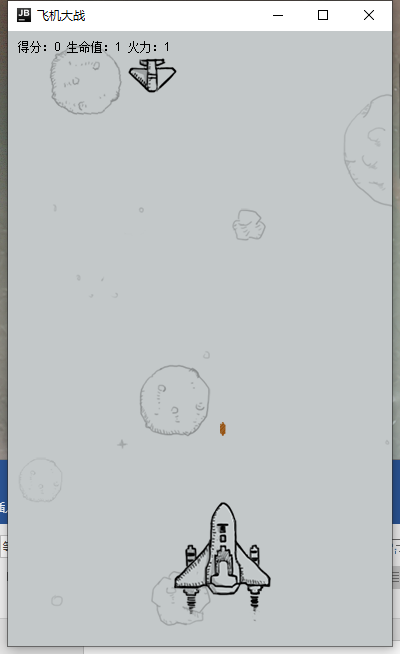
1. 验证生成的单位（大敌机）



1. 生命值为0结束游戏



1. 再次点击重新开始



很多动态的测试无法展示，功能均都已实现。

### 总结

本次实训，让我从仅仅是“纸上谈兵“到可以编写出一个小游戏的过程，令我受益匪浅，让我知道了有Java适配器并且使用这个东西，非常的巧妙，就像手机是Type-c接口，但是只有一个传统的MicroUSB的充电线，那么就弄一个转接头就可以充电了，而适配器就像这么一个转接头，还知道了我若想实现一个接口而不想把所有的方法实现，可以通过匿名内部类去只实现其中的一个接口，比如Timer中的schedule，还初步了解了Swing的用法，通过自学将数组改成集合，让我知道了Java集合有多么的好用，多么的巧妙，而令我印象最深刻的便是我深刻地理解到了面向对象编程，和多态的美妙，那种使用父类或者一个接口去归类几种类，在主函数中统一调用，若是A类对象则用A方法，若是B类对象则用B方法。面向对象去编程，需要哪一些对象，去为之做一个类，去编写他的行为，还有封装，面对一项功能，我们可以尽可能封装在外面或者别的类内，从而更加有层次，在代码量特别多的时候更好去理解，去梳理思路等等，我认为这是我收获最大的，让我深刻的了解了面向对象的三大特性：封装、继承、多态，从“纸上谈兵”，到实操，让我收获了很多很多。