Leistungsnachweis im Fach Programmierung 1

Maximilian von Hohenbühel
Fabian Cieslik

2020-12-07

Contents

1 Projektvorraussetzungen

Beschreibung: Die Projektaufgabe besteht darin, ein einfaches Spiel zu implementieren. Die Wahl des Spiels bleibt Ihnen überlassen, beachten Sie jedoch, dass sich im Rahmen von 36 Stunden Arbeitszeit nur sehr begrenzte Spielideen auch umsetzen lassen.
Details: Sie programmieren ein Spiel für ein sehr eingeschränktes Display. Dieses enthält nur 24×48 Bildpunkte (Pixel), d.h. 24 Reihen mit jeweils 48 Spalten. Jeder Bildpunkt kann 16 Millionen Farben annehmen, wobei die Rot, Grün und Blau-Komponente mit jeweils einem Byte angesprochen wird. Als Steuermöglichkeit stehen Ihnen vier Tasten zur Verfügung, die wie im Cursorblock üblich angeordnet sind. Es gibt nur einen Spieler. Die Zeit für eine Spielrunde sollte bei 20-30 Sekunden liegen.
Zur Ein- und Ausgabe erhalten Sie eine Klasse mit zwei Methoden:

- *public int getKeyboard()*
  Liefert die vier Cursortasten der Tastatur folgende Werte zurück:
  - 0 -> "hoch"
  - 1 -> "runter"
  - 2 -> "links"
  - 3 -> "rechts"
  - -1 -> keine Taste

- *public void showImage(short[] image)*
  Zeigt ein komplettes Bild auf dem Display an, wobei der erste Wert des Arrays die Rot-Komponente des linken oben Bildpunkts ist und der letzte Wert die Blau-Komponente von 0 bis 255 des rechten unteren Bildpunktes. Das übergebene Array muss exakt $24 * 48 * 3$ Elemente haben für die 24 Zeilen, 48 Spalten und 3 Farbkomponenten pro Pixel. Das Display wird zeilenweise durchlaufen.

Spielumfang:

- Eine *interaktive Spielerfigur*
- Eine *automatisch gesteuerte Spielerfigur*
- Einen Hintergrund
- Ein *Score-System*
- Ein *Highscore-System*
- Implementierungsvorgaben:
  - Eine generische Klasse
  - Drei davon abgeleitete Klassen (Spieler, Hintergrund, Gegner/NPC)

2 Idee

Name:
MP - Mari proelium
Spiel:
Der Spieler steuert ein Schiff und probiert so lang zu überleben wie möglich. Es existieren Gegner, die sich zufällig bewegen.
Das Spiel wird in einer Vogelperspektive gespielt, man hat dadurch jederzeit den Überblick der gesamten Karte.
Das Spielprinzip der Runden wird in dem Sinne implementiert, dass alle 30 Sekunden neue Gegner auftauchen und man für jede überlebte Runde zusätzliche Punkte bekommt.
Das Punktesystem wird von der Zeit, die man am Leben ist und der Anzahl der besiegten feindlichen Schiffe beeinflusst.
Die Steuerung wird auf die vier verfügbaren Tasten aufgeteilt, sodass man ohne Probleme sein Schiff steuern kann und zugleich auch schiessen kann. Die Kollisionsinteraktionen mit feindlichen Schiffen und eventuellen Häfen wird vom System übernommen.

3 Beschreibung

Karte:

- **Aussehen**
  Die Karte hat einen blauen Hintergrund, der den Ozean darstellt. Auf dem Ozean kann es vorkommen, dass es verschiedene Inseln geben kann. Inseln werden als braune oder grüne Pixel dargestellt. Die Platzierung der Inseln wird zufällig am Anfang des Spiels festgelegt und wird nur bei einem kompletten Neustart verändert. Auf den Inseln können Häfen generiert werden, die einem einen Bonus geben, falls man sie erreichen sollte.

Spieler:

- **Aussehen**
  Ein zwei bzw. drei Pixel langes Schiff.

- **Fähigkeiten**
  - Links: 45 Grad Drehung gegen den Uhrzeigersinn
  - Rechts: 45 Grad Drehung in den Uhrzeigersinn
  - Oben: Vorwärts Bewegung nach vorne
  - Unten: Benutzen der Schiffsinternen Kanone

Gegner:

- **Aussehen**
  Ein zwei bzw. drei Pixel langes Schiff.

- **Fähigkeiten**
  - Zufälliges Bewegen auf der Karte
  - Bei Spielersicht wird Geschossen

Punkte:

- **Punktequellen**
  - Beim treffen eines Gegners
  - Beim besiegen eines Gegners
  - Besiegen aller gerade lebender Geger
  - Fürs überleben einer Runde

- **Highscore**
  Punkte werden in der Konsole als Highscore nach jedem Tod des Spielers ausgegeben

4 Programmablauf

- **Vorbereitung**
  Es werden alle Spielnotwendigen Variablen deklariert und initialisiert. In einer *Do-While* Schleife wird daraufhin gestarted um mehrere Spiele hintereinander spielen zu Können. Am Start der Schleife wird die Karte, der Spieler und die Gegner erstellt und gezeichnet und auf eine Eingabe des Benutzers gewartet. Bei Eingabe wird der Spielablauf gestartet. Nach dem Tod des Spielers wird der Punktestand ausgegeben und die Möglichkeit geboten ein neues Spiel zu Starten.

- **Spielablauf**
  Zuerst wird der Spieler bewegt und auf Kollisionen überprüft, danach die Gegner. Anschließend wird überprüft ob man eine Runde überlebt hat.

5 Klassendiagramm

6 Programmcode

Ship.java

```
1  public class Ship extends Agent {
2      protected int hp;
3      protected int[][] pos;
4      protected int[][] oldpos;
5      protected int align;
6      protected short[][][] color = new short[3][3][3];
7      protected Bullet bullet = null;
8
9      Ship(int hp){
10         this.pos = new int[3][2];
11         this.oldpos = new int[3][2];
12         this.hp = hp;
13         this.align = 7;
14         this.pos[0][0] = 2;
15         this.pos[0][1] = 3;
16         this.pos[1][0] = 2;
17         this.pos[1][1] = 2;
18         this.pos[2][0] = 2;
19         this.pos[2][1] = 1;
20     }
21
22     Ship(int hp, int x, int y, int orient){
23         this.pos = new int[3][2];
24         this.oldpos = new int[3][2];
25         this.hp = hp;
26         if (x >= 1 && x <= 46 && y >= 1 && y <= 22) {
27             this.align = orient;
28             this.pos[1][0] = x;
29             this.pos[1][1] = y;
30             switch(this.align){
31                 case 1:
32                     this.pos[0][0] = x;
33                     this.pos[0][1] = y - 1;
34                     this.pos[2][0] = x;
35                     this.pos[2][1] = y + 1;
36                     break;
37                 case 2:
38                     this.pos[0][0] = x + 1;
39                     this.pos[0][1] = y - 1;
40                     this.pos[2][0] = x - 1;
41                     this.pos[2][1] = y + 1;
42                     break;
43                 case 3:
44                     this.pos[0][0] = x + 1;
45                     this.pos[0][1] = y;
46                     this.pos[2][0] = x - 1;
```

```
47                  this.pos[2][1] = y;
48                  break;
49              case 4:
50                  this.pos[0][0] = x + 1;
51                  this.pos[0][1] = y + 1;
52                  this.pos[2][0] = x - 1;
53                  this.pos[2][1] = y - 1;
54                  break;
55              case 5:
56                  this.pos[0][0] = x;
57                  this.pos[0][1] = y + 1;
58                  this.pos[2][0] = x;
59                  this.pos[2][1] = y - 1;
60                  break;
61              case 6:
62                  this.pos[0][0] = x - 1;
63                  this.pos[0][1] = y + 1;
64                  this.pos[2][0] = x + 1;
65                  this.pos[2][1] = y - 1;
66                  break;
67              case 7:
68                  this.pos[0][0] = x - 1;
69                  this.pos[0][1] = y;
70                  this.pos[2][0] = x + 1;
71                  this.pos[2][1] = y;
72                  break;
73              case 8:
74                  this.pos[0][0] = x - 1;
75                  this.pos[0][1] = y - 1;
76                  this.pos[2][0] = x + 1;
77                  this.pos[2][1] = y + 1;
78                  break;
79          }
80      }else {
81          this.pos[0][0] = 2;
82          this.pos[0][1] = 3;
83          this.pos[1][0] = 2;
84          this.pos[1][1] = 2;
85          this.pos[2][0] = 2;
86          this.pos[2][1] = 1;
87          this.align = 7;
88      }
89  }
90
91  protected short[] clearTrace(short[] myImage){
92      for (int i = 0; i < this.oldpos.length; i++) {
93          myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 0]
                = (short)0;
94          myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 1]
                = (short)177;
```

```
95              myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 2]
                    = (short)241;
96          }
97          return myImage;
98      }
99
100     /**
101      * This method uses the Players values to update the map and return
             it always.
102      * @param myImage the Pixel array given from the {@link GameMain}
103      * @return the updated maparray
104      * */
105     public short[] paint(short[] myImage){
106         myImage = clearTrace(myImage);
107         if(this.hp > 0) {
108             for(int i=0; i < this.pos.length; i++){
109                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
                        color[this.hp - 1][i][0];
110                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
                        color[this.hp - 1][i][1];
111                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
                        color[this.hp - 1][i][2];
112             }
113         }else {
114             for(int i=0; i < this.pos.length; i++){
115                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
                        0;
116                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
                        177;
117                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
                        241;
118             }
119             if(this.bullet != null){
120                 myImage = this.bullet.clear(myImage);
121             }
122         }
123         return myImage;
124     }
125
126     public short[] isHit(short[] myImage){
127         for(int i = 0; i < this.pos.length; i++){
128             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
129                 damage(1);
130             }
131         }
132         myImage = paint(myImage);
133         return myImage;
134     }
135
136     public boolean isAlive(){
```

```
137         return (this.hp > 0);
138     }
139 
140     protected boolean comparePixel(short r1, short g1, short b1, short
            r2, short g2, short b2){
141         return (r1 == r2 && g1 == g2 && b1 == b2);
142     }
143 
144     /**
145      * The method collide looks at the pixels of the ship and look if it
              collided with another object
146      * */
147     public int collide(short[] myImage){
148         return -1;
149     }
150 
151     /**
152      * This method takes the userinput and changes the
              position/direction of the ship
153      * @param dir represents the given userinput
154      *          0 - up
155      *          1 - down
156      *          2 - left
157      *          3 - right
158      * */
159     protected void move(int dir){
160         switch(dir){
161             case 0: // Hoch
162                 forward();
163                 break;
164             case 1: // Runter
165                 shoot();
166                 break;
167             case 2: // Links
168                 rotate(0);
169                 break;
170             case 3: // Rechts
171                 rotate(1);
172                 break;
173         }
174     }
175 
176     /**
177      * This method will be called by the move method and rotates the
              ship in the given direction.
178      * @param dir represents the direction which the ship takes to
              rotate.
179      *          0 - Left
180      *          1 - Right
181      * */
```

```
182     protected void rotate(int dir){
183         if(dir == 0){ // Left
184             switch(this.align){
185                 case 1:
186                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                            47) {
187                         saveOldPos();
188                         this.pos[0][0]--;
189                         this.pos[2][0]++;
190                         changeAlign(-1);
191                     }
192                     break;
193                 case 2:
194                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                            47) {
195                         saveOldPos();
196                         this.pos[0][0]--;
197                         this.pos[2][0]++;
198                         changeAlign(-1);
199                     }
200                     break;
201                 case 3:
202                     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                            23) {
203                         saveOldPos();
204                         this.pos[0][1]--;
205                         this.pos[2][1]++;
206                         changeAlign(-1);
207                     }
208                     break;
209                 case 4:
210                     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                            23) {
211                         saveOldPos();
212                         this.pos[0][1]--;
213                         this.pos[2][1]++;
214                         changeAlign(-1);
215                     }
216                     break;
217                 case 5:
218                     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                            0) {
219                         saveOldPos();
220                         this.pos[0][0]++;
221                         this.pos[2][0]--;
222                         changeAlign(-1);
223                     }
224                     break;
225                 case 6:
226                     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
```

```
                    0) {
227                     saveOldPos();
228                     this.pos[0][0]++;
229                     this.pos[2][0]--;
230                     changeAlign(-1);
231                 }
232                 break;
233             case 7:
234                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0) {
235                     saveOldPos();
236                     this.pos[0][1]++;
237                     this.pos[2][1]--;
238                     changeAlign(-1);
239                 }
240                 break;
241             case 8:
242                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0) {
243                     saveOldPos();
244                     this.pos[0][1]++;
245                     this.pos[2][1]--;
246                     changeAlign(-1);
247                 }
248                 break;
249         }
250     }else { // Right
251         switch(this.align){
252             case 1:
253                 if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                        0) {
254                     saveOldPos();
255                     this.pos[0][0]++;
256                     this.pos[2][0]--;
257                     changeAlign(1);
258                 }
259                 break;
260             case 2:
261                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0) {
262                     saveOldPos();
263                     this.pos[0][1]++;
264                     this.pos[2][1]--;
265                     changeAlign(1);
266                 }
267                 break;
268             case 3:
269                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0) {
270                     saveOldPos();
```

```
271                         this.pos[0][1]++;
272                         this.pos[2][1]--;
273                         changeAlign(1);
274                     }
275                     break;
276                 case 4:
277                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                            47) {
278                         saveOldPos();
279                         this.pos[0][0]--;
280                         this.pos[2][0]++;
281                         changeAlign(1);
282                     }
283                     break;
284                 case 5:
285                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                            47) {
286                         saveOldPos();
287                         this.pos[0][0]--;
288                         this.pos[2][0]++;
289                         changeAlign(1);
290                     }
291                     break;
292                 case 6:
293                     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                            23) {
294                         saveOldPos();
295                         this.pos[0][1]--;
296                         this.pos[2][1]++;
297                         changeAlign(1);
298                     }
299                     break;
300                 case 7:
301                     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                            23) {
302                         saveOldPos();
303                         this.pos[0][1]--;
304                         this.pos[2][1]++;
305                         changeAlign(1);
306                     }
307                     break;
308                 case 8:
309                     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                            0) {
310                         saveOldPos();
311                         this.pos[0][0]++;
312                         this.pos[2][0]--;
313                         changeAlign(1);
314                     }
315                     break;
```

```
316             }
317         }
318     }
319 
320     /**
321      * Method to save the ship position from one move ago.
322      * */
323     protected void saveOldPos(){
324         for(int i = 0; i < this.pos.length; i++){
325             this.oldpos[i][0] = this.pos[i][0];
326             this.oldpos[i][1] = this.pos[i][1];
327         }
328     }
329 
330     protected void damage(int amount){
331         this.hp -= amount;
332     }
333 
334     /**
335      * Used to move the ship in the direction it is aligned to.
336      * */
337     protected void forward(){
338         if(canMove()){
339             switch(this.align){
340                 case 1:
341                     saveOldPos();
342                     for (int i = 0; i < this.pos.length; i++){
343                         this.pos[i][1]--;
344                     }
345                     break;
346                 case 2:
347                     saveOldPos();
348                     for (int i = 0; i < this.pos.length; i++){
349                         this.pos[i][0]++;
350                         this.pos[i][1]--;
351                     }
352                     break;
353                 case 3:
354                     saveOldPos();
355                     for (int i = 0; i < this.pos.length; i++){
356                         this.pos[i][0]++;
357                     }
358                     break;
359                 case 4:
360                     saveOldPos();
361                     for (int i = 0; i < this.pos.length; i++){
362                         this.pos[i][0]++;
363                         this.pos[i][1]++;
364                     }
365                     break;
```

```
366                 case 5:
367                     saveOldPos();
368                     for (int i = 0; i < this.pos.length; i++){
369                         this.pos[i][1]++;
370                     }
371                     break;
372                 case 6:
373                     saveOldPos();
374                     for (int i = 0; i < this.pos.length; i++){
375                         this.pos[i][0]--;
376                         this.pos[i][1]++;
377                     }
378                     break;
379                 case 7:
380                     saveOldPos();
381                     for (int i = 0; i < this.pos.length; i++){
382                         this.pos[i][0]--;
383                     }
384                     break;
385                 case 8:
386                     saveOldPos();
387                     for (int i = 0; i < this.pos.length; i++){
388                         this.pos[i][0]--;
389                         this.pos[i][1]--;
390                     }
391                     break;
392             }
393         }
394     }
395
396     /**
397      * Used to determine if the ship can move forward.
398      * @return the returnvalue says, if the ship can move forward or if
           the ship would move outside the map.
399      * */
400     protected boolean canMove(){
401         boolean ret = false;
402         switch(this.align){
403             case 1 -> ret = (this.pos[0][1] - 1 >= 0);
404             case 2 -> ret = (this.pos[0][1] - 1 > 0 && this.pos[0][0] + 1
                    < 48);
405             case 3 -> ret = (this.pos[0][0] + 1 < 48);
406             case 4 -> ret = (this.pos[0][0] + 1 < 48 && this.pos[0][1] +
                    1 < 24);
407             case 5 -> ret = (this.pos[0][1] + 1 < 24);
408             case 6 -> ret = (this.pos[0][1] + 1 < 24 && this.pos[0][0] -
                    1 >= 0);
409             case 7 -> ret = (this.pos[0][0] - 1 >= 0);
410             case 8 -> ret = (this.pos[0][0] - 1 >= 0 && this.pos[0][1] -
                    1 >= 0);
```

```
411         }
412         return ret;
413     }
414     /**
415      * 8 1 2
416      * 7   3
417      * 6 5 4
418      * */
419     protected void shoot() {
420         int dir1 = (this.align + 2 > 8) ? (this.align + 2 - 8) :
                (this.align + 2);
421         // int dir2 = (this.align - 2 < 1) ? (8 + this.align - 2) :
                (this.align - 2);
422         if(this.bullet == null){
423             this.bullet = new Bullet(dir1, 5, this.pos[1][0],
                    this.pos[1][1]);
424         }
425         // bullets.add(new Bullet(dir2, 5, this.pos[1][0],
                this.pos[1][1]));
426     }
427
428     protected boolean hitPlayer(short[] myImage, int x, int y){
429         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
430             int idx = (y * 48 + x) * 3;
431             return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
                    myImage[idx + 2] == 36) ||
432                 (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
                        myImage[idx + 2] == 36) ||
433                 (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
                        myImage[idx + 2] == 36) ||
434                 (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
                        myImage[idx + 2] == 22) ||
435                 (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
                        myImage[idx + 2] == 22) ||
436                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
                        myImage[idx + 2] == 21) ||
437                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
                        myImage[idx + 2] == 11) ||
438                 (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
                        myImage[idx + 2] == 10) ||
439                 (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 11);
440         }else {
441             return false;
442         }
443     }
444
445     protected boolean hitEnemy(short[] myImage, int x, int y){
446         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
447             int idx = (y * 48 + x) * 3;
```

```
448             return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
                    myImage[idx + 2] == 222) ||
449                 (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
                        myImage[idx + 2] == 143) ||
450                 (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
                        myImage[idx + 2] == 74) ||
451                 (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
                        myImage[idx + 2] == 215) ||
452                 (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
                        myImage[idx + 2] == 134) ||
453                 (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 72) ||
454                 (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
                        myImage[idx + 2] == 209) ||
455                 (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
                        myImage[idx + 2] == 128) ||
456                 (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
                        myImage[idx + 2] == 74);
457         }else {
458             return false;
459         }
460     }
461 
462     protected boolean hitBullet(short[] myImage, int x, int y){
463         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
464             int idx = (y * 48 + x) * 3;
465             return (myImage[idx + 0] == 12 && myImage[idx + 1] == 13 &&
                        myImage[idx + 2] == 12);
466         }else {
467             return false;
468         }
469     }
470 
471 
472     /**
473      * This method is used to set the align variable after a succesful
                rotation
474      * @param dir the direction the ship rotates to
475      * */
476     protected void changeAlign(int dir){
477         this.align += dir;
478         if(this.align < 1){
479             this.align = 8;
480         }
481         if(this.align > 8){
482             this.align = 1;
483         }
484     }
485 
486     public short[] run(int key, short[] myImage){
```

```
487         myImage = isHit(myImage);
488         if(key != -1){
489             myImage = clearTrace(myImage);
490             move(key);
491             if (collide(myImage) == 1){
492                 resetMove();
493                 if(key == 2){
494                     this.align++;
495                 }
496                 if(key == 3){
497                     this.align--;
498                 }
499             }
500         }
501         if(this.bullet != null){
502             if(this.bullet.getRange() > 0){
503                 this.bullet.run(-1, myImage);
504             }else{
505                 this.bullet = null;
506             }
507         }
508         myImage = paint(myImage);
509         return myImage;
510     }
511 
512     protected void resetMove(){
513         for(int i=0; i < this.pos.length; i++){
514             for(int j=0; j < this.pos[i].length; j++){
515                 this.pos[i][j] = this.oldpos[i][j];
516             }
517         }
518     }
519 
520     /**
521      * Debug method to print shiplocation and locationdifference between
             the new and old location.
522      * */
523     public void print(String where){
524         System.out.println(where + "\nA: " + this.align);
525         for (int i = 0; i < this.pos.length; i++){
526             System.out.println("X: " + this.pos[i][0] + " Y: " +
                  this.pos[i][1] + " | Xo: " + this.oldpos[i][0] + " Yo: "
                  + this.oldpos[i][1]);
527             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                  - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                  this.oldpos[i][1]));
528         }
529     }
530 
531     public int[][] getPos(){
```

```
532         return this.pos;
533     }
534 
535     public int getHp(){
536         return this.hp;
537     }
538 
539     public void setHp(int hp){
540         this.hp = (hp >= 0)? hp : 0;
541     }
542 
543     protected void changeColor(short[][][] rgbs){
544         for (int i = 0; i < this.color.length; i++) {
545             for (int j = 0; j < this.color[0].length; j++){
546                 for (int k = 0; k < this.color[0][0].length; k++){
547                     this.color[i][j][k] = (rgbs[i][j][k] <= 255 &&
                            rgbs[i][j][k] >= 0)? rgbs[i][j][k] : 0;
548                 }
549             }
550         }
551     }
552 }
```

Agent.java

```
1  public abstract class Agent {
2
3      abstract short[] paint(short[] myImage);
4
5      abstract int collide(short[] myImage);
6
7      abstract void move(int dir);
8
9      abstract short[] run(int key, short[] myImage);
10 }
```

Enemy.java

```
1  import java.util.List;
2  import java.util.ArrayList;
3  public class Enemy extends Ship {
4      private int range;
5      private int dmg = 0;
6      private int PX;
7      private int PY;
8      private int RouteX = -1;
9      private int RouteY = -1;
10     private boolean detectedPlayer = false;
11     private List<int[]> routing = new ArrayList<int[]>();
12
13     Enemy(int hp){
14         super(hp);
15         short[][][] rgbs = {{{31, 222, 215},{21, 138, 134},{11, 74,
               72}},{{31, 69, 222},{19, 43, 143},{10, 22, 74}},{{153, 23,
               209},{94, 15, 128},{55, 10, 74}}};
16         changeColor(rgbs);
17         int[][] pos = { {22, 22}, {23, 22}, {24, 22} };
18         this.pos = pos;
19     }
20
21     Enemy(int hp, int x, int y, int o, int r){
22         super(hp, x, y, o);
23         short[][][] rgbs = {{{31, 222, 215},{21, 138, 134},{11, 74,
               72}},{{31, 69, 222},{19, 43, 143},{10, 22, 74}},{{153, 23,
               209},{94, 15, 128},{55, 10, 74}}};
24         changeColor(rgbs);
25         this.range = r;
26     }
27
28     public void resetDmg(){
29         this.dmg = 0;
30     }
31
32     public int getDamageReceived(){
33         return this.dmg;
34     }
35
36     /**
37      *  Create Routes
38      * */
39     private void pathFinder(){
40         this.routing = new ArrayList<int[]>();
41         /**
42          *  Start: this.pos[1][0]=x
43          *         this.pos[1][1]=y
44          *  End:   this.RouteX
```

```
45          *         this.RouteY
46          * */
47         this.PX = this.RouteX;
48         this.PY = this.RouteY;
49         int pX = this.pos[1][0];
50         int pY = this.pos[1][1];
51         while(pX != this.RouteX && pY != this.RouteY){
52             int[] ia = {this.RouteX, this.RouteY};
53             switch(routeDirection(pX, pY, ia)){
54                 case 1 -> {
55                     int[] rt = {pX, --pY};
56                     this.routing.add(rt);
57                 }
58                 case 2 -> {
59                     int[] rt = {++pX, --pY};
60                     this.routing.add(rt);
61                 }
62                 case 3 -> {
63                     int[] rt = {++pX, pY};
64                     this.routing.add(rt);
65                 }
66                 case 4 -> {
67                     int[] rt = {++pX, ++pY};
68                     this.routing.add(rt);
69                 }
70                 case 5 -> {
71                     int[] rt = {pX, ++pY};
72                     this.routing.add(rt);
73                 }
74                 case 6 -> {
75                     int[] rt = {--pX, ++pY};
76                     this.routing.add(rt);
77                 }
78                 case 7 -> {
79                     int[] rt = {--pX, pY};
80                     this.routing.add(rt);
81                 }
82                 case 8 -> {
83                     int[] rt = {--pX, --pY};
84                     this.routing.add(rt);
85                 }
86                 default -> {
87                     break;
88                 }
89             }
90         }
91     }
92 
93     public short[] run(short[] myImage){
94         if(this.RouteX != -1 && this.RouteY != -1){
```

```
95              pathFinder();
96              pR();
97              System.out.println(this.routing.get(0)[0]);
98          }
99          myImage = clearTrace(myImage);
100         if(playerInVision(myImage) && this.bullet == null){
101             shoot();
102         }
103         move();
104         if (collide(myImage) != 0){
105             resetMove();
106         }
107         if(this.bullet != null){
108             if(this.bullet.getRange() > 0){
109                 this.bullet.run(-1, myImage);
110             }else{
111                 this.bullet = null;
112             }
113         }
114         myImage = paint(myImage);
115         return myImage;
116     }
117
118     private void move(){
119         if(this.routing.size() <= 0 ){
120             if(canMove()){
121                 forward();
122             }else{
123                 if(Math.random() > 0.5){
124                     rotate(0);
125                 }else {
126                     rotate(1);
127                 }
128             }
129         }else {
130             switch(routeDirection(this.pos[1][0], this.pos[1][1],
                     this.routing.get(this.routing.size() - 1))){
131                 case 1 -> {
132                     rotateTo(1);
133                     forward();
134                 }
135                 case 2 -> {
136                     rotateTo(2);
137                     forward();
138                 }
139                 case 3 -> {
140                     rotateTo(3);
141                     forward();
142                 }
143                 case 4 -> {
```

```
144                 rotateTo(4);
145                 forward();
146             }
147             case 5 -> {
148                 rotateTo(5);
149                 forward();
150             }
151             case 6 -> {
152                 rotateTo(6);
153                 forward();
154             }
155             case 7 -> {
156                 rotateTo(7);
157                 forward();
158             }
159             case 8 -> {
160                 rotateTo(8);
161                 forward();
162             }
163             default -> {}
164         }
165         this.routing.remove(this.routing.size() - 1);
166     }
167 }
168 
169 private void pR(){
170     for(int[] i : this.routing){
171         System.out.println("| " + i[0] + " | " + i[1] + " |");
172     }
173 }
174 
175 private void rotateTo(int newOri){
176     while (this.align != newOri) {
177         rotate(1);
178     }
179 }
180 
181 private int routeDirection(int x, int y, int[] gPos){
182     if(x > gPos[0]){
183         if(y > gPos[1]){
184             return 8;
185         }else if(y < gPos[1]){
186             return 6;
187         }else {
188             return 7;
189         }
190     }else if(x > gPos[0]){
191         if(y > gPos[1]) {
192             return 2;
193         }else if(y < gPos[1]){
```

```
194                 return 4;
195             }else {
196                 return 3;
197             }
198         }else {
199             if(y > gPos[1]) {
200                 return 1;
201             }else if(y < gPos[1]){
202                 return 5;
203             }
204         }
205         return -1;
206     }
207 
208     /**
209      * Method to detect if the player is visible for the enemy ship.
210      * */
211     private boolean playerInVision(short[] myImage){
212         int difx;
213         int dify;
214         if(this.hp > 0){
215             for (int i = 0 - this.range; i <= this.range; i++) {
216                 difx = this.pos[1][0] + i;
217                 for (int j = 0 - this.range; j <= this.range; j++) {
218                     dify = this.pos[1][1] + j;
219                     if ((Math.pow(difx - this.pos[0][1], 2)+Math.pow(dify
                          - this.pos[1][1], 2)) <= Math.pow(this.range, 2))
                          {
220                         if(hitPlayer(myImage, difx,dify)) {
221                             this.detectedPlayer = true;
222                             this.PX = difx;
223                             this.PY = dify;
224                             return true;
225                         }
226                     }
227                 }
228             }
229         }
230         return false;
231     }
232 
233     public void setRouteX(int PX){
234         this.PX = PX;
235     }
236 
237     public void setRouteY(int PY){
238         this.PY = PY;
239     }
240 
241     public int getPX(){
```

```
242         return this.PX;
243     }
244 
245     public int getPY(){
246         return this.PY;
247     }
248 
249     public boolean getPlayerDetected(){
250         return this.detectedPlayer;
251     }
252 
253     /**
254      * The method collide looks at the pixels of the ship and look if it
              collided with another object
255      * */
256     public int collide(short[] myImage){
257         int ret = 0;
258         for(int i=0; i < this.pos.length; i++){
259             int idx = (this.pos[i][1] * 48 + this.pos[i][0]) * 3;
260             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
261                 damage(1);
262                 ret = 2;
263             }
264             if(hitPlayer(myImage, this.pos[i][0], this.pos[i][1])) {
265                 damage(1);
266                 this.dmg += 1;
267                 ret = 1;
268             }
269         }
270         return ret;
271     }
272 
273     public boolean includesPos(int x, int y){
274         for (int i = 0; i < this.pos.length; i++){
275             if(this.pos[i][0] == x && this.pos[i][1] == y){
276                 return true;
277             }
278         }
279         return false;
280     }
281 
282     /**
283      * Debug method to print shiplocation and locationdifference between
              the new and old location.
284      * */
285     public void print(){
286         System.out.println("Enemy ship:\nA: " + this.align);
287         for (int i = 0; i < this.pos.length; i++){
288             System.out.println("X: " + this.pos[i][0] + " Y: " +
                    this.pos[i][1]);
```

```
289             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
                    this.oldpos[i][1]);
290             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                    - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                    this.oldpos[i][1]));
291         }
292     }
293 }
```

Fleet.java

```
1  import java.util.List;
2  import java.util.ArrayList;
3  public class Fleet {
4      private List<Enemy> fleet;
5      private int PX;
6      private int PY;
7      private boolean detected = false;
8
9      Fleet(){
10         this.fleet = new ArrayList<Enemy>();
11     }
12
13     public void addFleetmember(Enemy s){
14         this.fleet.add(s);
15     }
16
17     private boolean isWater(short[] myImage, int idx){
18         return (myImage[idx] == 0 && myImage[idx + 1] == 177 &&
               myImage[idx + 2] == 241);
19     }
20
21     public short[] employFleet(short[] myImage, int num){
22         while (num > 0){
23             int i = (int)(Math.random() * 46) + 1;
24             int j = (int)(Math.random() * 22) + 1;
25             if(    isWater(myImage, (((j-1) * 48 + i ) * 3)) &&
26                    isWater(myImage, (((j-1) * 48 + (i+1)) * 3)) &&
27                    isWater(myImage, (((j-1) * 48 + (i-1)) * 3)) &&
28                    isWater(myImage, ((j   * 48 + i   ) * 3)) &&
29                    isWater(myImage, ((j   * 48 + (i+1)) * 3)) &&
30                    isWater(myImage, ((j   * 48 + (i-1)) * 3)) &&
31                    isWater(myImage, (((j+1) * 48 + i ) * 3)) &&
32                    isWater(myImage, (((j+1) * 48 + (i+1)) * 3)) &&
33                    isWater(myImage, (((j+1) * 48 + (i-1)) * 3))){
34                 addFleetmember(new Enemy(2, i, j, 4, 8));
35                 myImage = paintFleet(myImage);
36                 num--;
37                 continue;
38                     }
39         }
40         return myImage;
41     }
42
43     public int getNumberOfAliveShips(){
44         int ret = 0;
45         for (Enemy s : this.fleet){
46             if(s.isAlive()){
47                 ret++;
```

```
48              }
49          }
50          return ret;
51      }
52
53      public void resetDamageControl(){
54          for (Enemy e : this.fleet){
55              e.resetDmg();
56          }
57      }
58      public int damageControl(){
59          int ret = 0;
60          for (Enemy e : this.fleet){
61              ret += e.getDamageReceived();
62          }
63          return ret;
64      }
65
66      public void distributeDamage(int x, int y){
67          for (Enemy e : fleet){
68              if(e.includesPos(x, y)){
69                  e.damage(1);
70                  break;
71              }
72          }
73      }
74
75      public int getDead(){
76          int ret = 0;
77          for (Enemy e : fleet){
78              if(!e.isAlive()){
79                  ret++;
80              }
81          }
82          return ret;
83      }
84
85      public void printing(){
86          for(Enemy e : this.fleet){
87              e.print("text");
88          }
89      }
90
91      private void broadcastPosition(){
92          for(Enemy e : this.fleet){
93              if(e.getPlayerDetected()){
94                  this.detected = true;
95                  this.PX = e.getPX();
96                  this.PY = e.getPY();
97              }
```

```
98          }
99      }
100
101     public short[] executeOrders(short[] myImage){
102         broadcastPosition();
103         for (Enemy s : this.fleet){
104             if(s.isAlive()){
105                 s.setRouteX(this.PX);
106                 s.setRouteY(this.PY);
107                 myImage = s.run(myImage);
108             }
109         }
110         return myImage;
111     }
112
113     public short[] statusUpdate(short[] myImage){
114         for(Enemy e : this.fleet){
115             myImage = e.isHit(myImage);
116         }
117         return myImage;
118     }
119
120     public short[] paintFleet(short[] myImage){
121         for (Enemy s : this.fleet) {
122             myImage = s.paint(myImage);
123         }
124         return myImage;
125     }
126 }
```

Bullet.java

```
1  public class Bullet extends Agent {
2      private int direction;
3      private int range;
4      private int maxRange;
5      private int[] pos = new int[2];
6      private int[] oldpos = new int[2];
7      private boolean hasHit = false;
8
9      Bullet(int dir, int range, int x, int y){
10         this.direction = dir;
11         this.range = range;
12         this.maxRange = range;
13         this.pos[0] = x;
14         this.pos[1] = y;
15     }
16
17     private short[] clearTrace(short[] myImage){
18         myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 0] =
               (short)0;
19         myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 1] =
               (short)177;
20         myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 2] =
               (short)241;
21         return myImage;
22     }
23
24     public short[] clear(short[] myImage){
25             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 0] = (short)0;
26             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 1] =
                   (short)177;
27             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 2] =
                   (short)241;
28         return myImage;
29     }
30     public short[] paint(short[] myImage){
31         myImage = clearTrace(myImage);
32         if(this.range > 0) {
33             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 0] = (short)12;
34             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 1] = (short)13;
35             myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 2] = (short)12;
36         }else {
37             myImage = clear(myImage);
38         }
39         return myImage;
40     }
41
42     public int collide(short[] myImage){
43         return 0;
```

```
44      }
45
46      protected boolean hitPlayer(short[] myImage, int x, int y){
47          if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
48          int idx = (y * 48 + x) * 3;
49          return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
                    myImage[idx + 2] == 36) ||
50              (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
                    myImage[idx + 2] == 36) ||
51              (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
                    myImage[idx + 2] == 36) ||
52              (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
                    myImage[idx + 2] == 22) ||
53              (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
                    myImage[idx + 2] == 22) ||
54              (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
                    myImage[idx + 2] == 21) ||
55              (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
                    myImage[idx + 2] == 11) ||
56              (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
                    myImage[idx + 2] == 10) ||
57              (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
                    myImage[idx + 2] == 11);
58          }else {
59              return false;
60          }
61      }
62
63      protected boolean hitEnemy(short[] myImage, int x, int y){
64          if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
65          int idx = (y * 48 + x) * 3;
66          return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
                    myImage[idx + 2] == 222) ||
67              (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
                    myImage[idx + 2] == 143) ||
68              (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
                    myImage[idx + 2] == 74) ||
69              (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
                    myImage[idx + 2] == 215) ||
70              (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
                    myImage[idx + 2] == 134) ||
71              (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
                    myImage[idx + 2] == 72) ||
72              (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
                    myImage[idx + 2] == 209) ||
73              (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
                    myImage[idx + 2] == 128) ||
74              (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
                    myImage[idx + 2] == 74);
75          }else {
```

```
76              return false;
77          }
78      }
79
80      public void move(int dir){}
81
82      private void saveOldPos(){
83          this.oldpos[0] = this.pos[0];
84          this.oldpos[1] = this.pos[1];
85      }
86
87      public boolean move(){
88          if(canMove()){
89              saveOldPos();
90              switch(this.direction){
91                  case 1:
92                      this.pos[1]--;
93                      break;
94                  case 2:
95                      this.pos[0]++;
96                      this.pos[1]--;
97                      break;
98                  case 3:
99                      this.pos[0]++;
100                     break;
101                 case 4:
102                     this.pos[0]++;
103                     this.pos[1]++;
104                     break;
105                 case 5:
106                     this.pos[1]++;
107                     break;
108                 case 6:
109                     this.pos[0]--;
110                     this.pos[1]++;
111                     break;
112                 case 7:
113                     this.pos[0]--;
114                     break;
115                 case 8:
116                     this.pos[0]--;
117                     this.pos[1]--;
118                     break;
119             }
120             this.range--;
121             return true;
122         }else {
123             return false;
124         }
125     }
```

```
126
127     private boolean canMove(){
128         boolean ret = false;
129         switch(this.direction){
130             case 1 -> ret = (this.pos[1] - 1 >= 0);
131             case 2 -> ret = (this.pos[1] - 1 > 0 && this.pos[0] + 1 < 48);
132             case 3 -> ret = (this.pos[0] + 1 < 48);
133             case 4 -> ret = (this.pos[0] + 1 < 48 && this.pos[1] + 1 <
                    24);
134             case 5 -> ret = (this.pos[1] + 1 < 24);
135             case 6 -> ret = (this.pos[1] + 1 < 24 && this.pos[0] - 1 >=
                    0);
136             case 7 -> ret = (this.pos[0] - 1 >= 0);
137             case 8 -> ret = (this.pos[0] - 1 >= 0 && this.pos[1] - 1 >=
                    0);
138         }
139         return ret;
140     }
141
142     public short[] run(int key, short[] myImage){
143         if(this.range == this.maxRange){
144             if(move()){
145             myImage = paint(myImage);
146             }else {
147                 this.range = 0;
148                 myImage = clear(myImage);
149             }
150         }else{
151         if (!(hitEnemy(myImage, this.pos[0], this.pos[1]) ||
                hitPlayer(myImage, this.pos[0], this.pos[1]))){
152             if(move()){
153             myImage = paint(myImage);
154             }else {
155                 this.range = 0;
156                 myImage = clear(myImage);
157             }
158         }else{
159             this.range = 0;
160             this.hasHit = true;
161             myImage = paint(myImage);
162         }
163         }
164         return myImage;
165     }
166
167     public boolean getHasHit(){
168         return this.hasHit;
169     }
170
171     public int getRange(){
```

```
172         return this.range;
173     }
174 }
```

Harbor.java

```java
public class Harbor extends Agent{
    protected short[] color = {125, 66, 24};
    protected int orient;
    protected boolean captured = false;
    protected boolean possession = false;

    Harbor(int orient){
        this.orient = orient;
    }

    public int getOrient(){
        return this.orient;
    }

    @Override
    short[] paint(short[] myImage) {
        return new short[0];
    }

    @Override
    int collide(short[] myImage) {
        return -1;
    }

    @Override
    void move(int dir) {
    }

    @Override
    short[] run(int key, short[] myImage) {
        return new short[0];
    }
}
```

Island.java

```
1  public class Island extends Agent {
2      protected short[][] color = {{196, 156, 53},{186, 148, 48}}; //
           normale Insel, Hafeninsel
3      protected int[][][] pos;
4      protected int[] size;
5      protected Harbor harbor = null;
6
7      Island(int[] size, int x, int y){
8          this.pos = new int[size[0]][size[1]][2];
9          for(int i1 = 0; i1 < size[0]; i1++){
10             for(int i2 = 0; i2 < size[1]; i2++){
11                 this.pos[i1][i2][0] = x + i1;
12                 this.pos[i1][i2][1] = y + i2;
13             }
14         }
15     }
16
17     Island(int[] size, int x, int y, Harbor harbor){
18         this.pos = new int[size[0]][size[1]][2];
19         for(int i1 = 0; i1 < size[0]; i1++){
20             for(int i2 = 0; i2 < size[1]; i2++){
21                 this.pos[i1][i2][0] = x + i1;
22                 this.pos[i1][i2][1] = y + i2;
23             }
24         }
25         this.harbor = harbor;
26     }
27
28     @Override
29     short[] paint(short[] myImage) {
30         for(int i1 = 0; i1 < this.pos.length; i1++){
31             for(int i2 = 0; i2 < this.pos[i1].length; i2++){
32                 if(harbor == null){
33                     myImage[(this.pos[i1][i2][1] * 48 +
                           this.pos[i1][i2][0]) * 3 + 0] = color[0][0]; //
                           (y * 48 + x) * 3 + 0
34                     myImage[(this.pos[i1][i2][1] * 48 +
                           this.pos[i1][i2][0]) * 3 + 1] = color[0][1]; //
                           (y * 48 + x) * 3 + 1
35                     myImage[(this.pos[i1][i2][1] * 48 +
                           this.pos[i1][i2][0]) * 3 + 2] = color[0][2]; //
                           (y * 48 + x) * 3 + 2
36                 }
37                 else{
38                     myImage[(this.pos[i1][i2][1] * 48 +
                           this.pos[i1][i2][0]) * 3 + 0] = color[1][0]; //
                           (y * 48 + x) * 3 + 0
39                     myImage[(this.pos[i1][i2][1] * 48 +
```

```
                        this.pos[i1][i2][0]) * 3 + 1] = color[1][1]; //
                        (y * 48 + x) * 3 + 1
40                  myImage[(this.pos[i1][i2][1] * 48 +
                        this.pos[i1][i2][0]) * 3 + 2] = color[1][2]; //
                        (y * 48 + x) * 3 + 2
41                  switch(harbor.getOrient()){
42                      case 1:
43
44                          break;
45                      case 2:
46
47                          break;
48                      case 3:
49
50                          break;
51                      case 4:
52
53                          break;
54                      case 5:
55
56                          break;
57                      case 6:
58
59                          break;
60                      case 7:
61
62                          break;
63                      case 8:
64
65                          break;
66                  }
67              }
68          }
69      }
70      return myImage;
71  }
72
73  @Override
74  int collide(short[] myImage) {
75      return -1;
76  }
77
78  @Override
79  void move(int dir) {
80  }
81
82  @Override
83  short[] run(int key, short[] myImage) {
84      return myImage;
85  }
```

```
86 }
```

Player.java

```
1  public class Player extends Ship {
2      private int score = 0;
3      private boolean hit = false;
4      private int hitX;
5      private int hitY;
6
7      Player(int hp){
8          super(hp);
9          short[][][] rgbs = {{{237, 76, 36},{145, 47, 22},{74, 24,
               11}},{{237, 207, 36},{148, 129, 22},{66, 58, 10}},{{123,
               237, 36},{74, 143, 21},{38, 74, 11}}};
10         changeColor(rgbs);
11
12         int[][] pos = { {5, 5}, {6, 5}, {7, 5} };
13         this.pos = pos;
14     }
15
16     Player(int hp, int x, int y, int o){
17         super(hp, x, y, o);
18         short[][][] rgbs = {{{237, 76, 36},{145, 47, 22},{74, 24,
               11}},{{237, 207, 36},{148, 129, 22},{66, 58, 10}},{{123,
               237, 36},{74, 143, 21},{38, 74, 11}}};
19         changeColor(rgbs);
20     }
21
22     /**
23      * The method collide looks at the pixels of the ship and look if it
            collided with another object
24      * */
25     public int collide(short[] myImage){
26         int ret = 0;
27         for(int i=0; i < this.pos.length; i++){
28             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
29                 damage(1);
30                 ret = 1;
31             }
32             if(hitEnemy(myImage, this.pos[i][0], this.pos[i][1])) {
33                 this.hit = true;
34                 this.hitX = this.pos[i][0];
35                 this.hitY = this.pos[i][1];
36                 damage(1);
37                 ret = 1;
38             }
39         }
40         return ret;
41     }
42
43     public int[][] getPos(){
```

```
44          return this.pos;
45      }
46
47      public int getHitX(){
48          return this.hitX;
49      }
50      public int getHitY(){
51          return this.hitY;
52      }
53      public boolean getHit(){
54          return this.hit;
55      }
56
57      public void resetHit(){
58          this.hit = false;
59          this.hitX = -1;
60          this.hitY = -1;
61      }
62
63      public short[] run(int key, short[] myImage){
64          myImage = isHit(myImage);
65          if(key != -1){
66              myImage = clearTrace(myImage);
67              move(key);
68              if (collide(myImage) == 1){
69                  resetMove();
70                  if(key == 2){
71                      this.align++;
72                  }
73                  if(key == 3){
74                      this.align--;
75                  }
76              }
77          }
78          if(this.bullet != null){
79              if(this.bullet.getHasHit()){
80                  addScore(50);
81              }
82              if(this.bullet.getRange() > 0){
83                  this.bullet.run(-1, myImage);
84              }else{
85                  this.bullet = null;
86              }
87          }
88          myImage = paint(myImage);
89          return myImage;
90      }
91
92      public int getScore(){
93          return this.score;
```

```
94      }
95
96      public void addScore(int val){
97          this.score += val;
98      }
99
100     /**
101      * Debug method to print shiplocation and locationdifference between
             the new and old location.
102      * */
103     public void print(){
104         System.out.println("Your ship:\nA: " + this.align);
105         for (int i = 0; i < this.pos.length; i++){
106             System.out.println("X: " + this.pos[i][0] + " Y: " +
                    this.pos[i][1]);
107             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
                    this.oldpos[i][1]);
108             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                    - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                    this.oldpos[i][1]));
109         }
110     }
111 }
```

GameMain.java

```
1  import java.util.ArrayList;
2  import java.util.Collections;
3  import java.util.List;
4  import java.util.Scanner;
5  public class GameMain {
6
7      static public void main(String[] passedArgs) throws
           InterruptedException {
8          short[] myImage = new short[24*48*3];
9          List<Integer> highscore = new ArrayList<Integer>();
10         int thisKey=0;
11         int frame = 0;
12         int round = 1;
13         long startTime = System.currentTimeMillis();
14         long roundtime = 30000;
15
16         // This is initialization, do not change this
17         InternalLedGameThread.run();
18
19         // Now we show some introductory message and wait 3s before we
               switch to purple
20         System.out.println("Willkommen bei Mari proelium!\n In kuerze
               wird das Spiel beginnen und Ihr Punktestand wird mit den
               anderen Spielern verglichen!\n");
21         Thread.sleep(1000);
22
23         boolean end = false;
24         Scanner scan = new Scanner(System.in);
25         do {
26             for(int i=0; i<myImage.length; i+=3){
27                 myImage[i+0]=(short)0;
28                 myImage[i+1]=(short)177;
29                 myImage[i+2]=(short)241;
30             }
31
32             System.out.println("Sending to displayThread");
33             Player p = new Player(3, 7, 7, 5);
34             Fleet fleet = new Fleet();
35             int[] islandPosition = {3,3};
36             Island island = new Island(islandPosition,20,20);
37
38             myImage = fleet.employFleet(myImage, 3);
39             myImage = p.paint(myImage);
40             myImage = fleet.paintFleet(myImage);
41             myImage = island.paint(myImage);
42             InternalLedGameThread.showImage(myImage);
43             System.out.println("Drucken Sie eine beliebige Taste um das
                   Spiel zu starten.");
```

```
44              while(true){
45                  if(InternalLedGameThread.getKeyboard() != -1){
46                      break;
47                  }
48              }
49              while(p.isAlive()){
50                  thisKey = InternalLedGameThread.getKeyboard();
51                  myImage = p.run(thisKey, myImage);
52                  if(p.getHit()){
53                      fleet.distributeDamage(p.getHitX(), p.getHitY());
54                  }
55                  myImage = fleet.statusUpdate(myImage);
56                  if(frame % 3 == 0) {
57                      frame = 0;
58                      myImage = fleet.executeOrders(myImage);
59                  }
60                  InternalLedGameThread.showImage(myImage);
61                  frame++;
62                  Thread.sleep(100);
63                  System.out.println("+++ " + (System.currentTimeMillis() -
                        startTime) + " +++");
64                  p.damage(fleet.damageControl());
65                  fleet.resetDamageControl();
66                  if((System.currentTimeMillis() - startTime) > roundtime){
67                      myImage = fleet.employFleet(myImage, (3 -
                            fleet.getNumberOfAliveShips()));
68                      round++;
69                      startTime = System.currentTimeMillis();
70                      p.addScore(200);
71                  }
72                  if(fleet.getNumberOfAliveShips() == 0){
73                      p.addScore(50);
74                  }
75              }
76
77              p.addScore(fleet.getDead() * 50);
78              highscore.add(p.getScore());
79              Collections.sort(highscore);
80              System.out.println("(" + round + ") - Score: " +
                    p.getScore());
81              System.out.println("Highscores:");
82              for(int i = 0; i < highscore.size(); i++){
83                  System.out.println("(" + (i+1) + ") -> " +
                        highscore.get(i));
84              }
85
86              System.out.println("Wollen Sie noch eine Runde spielen?
                    (Y/N)\n> ");
87              end = (scan.next() == "Y") ? false : true;
88          }while(!end);
```

```
89      }
90  }
```