# Leistungsnachweis im Fach Programmierung 1

Maximilian von Hohenbühel
Fabian Cieslik

2020-12-07 - 2020-01-17

# 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 $24 \times 48$ 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:**
Idee war es, ein Spiel zu programmieren, dass in der Vogelperspektive gespielt wird, um dem Spieler die maximale Übersicht über das Spiel zu geben, d.h. man sieht ständig die vollständige Karte.
Der Spieler steuert dabei ein Schiff, welches drei Leben besitzt und probiert so lang zu überleben wie nur möglich. Es existieren sowohl Gegner, die sich zufällig bewegen und nach jeder Runde neu erzeugt werden und Inseln, mit optionalen Häfen, die bei Eroberung sowohl dem Gegner als auch dem Spieler zugewandt sein können.
Das Spiel basiert auf dem klassischen Levelprinzip, d.h. dass alle 30 Sekunden neue Gegner auftauchen, jede dritte Runde ein weiterer Gegner erzeugt wird 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 und zugleich auch schießen kann. Die Kollisionsinteraktionen mit feindlichen Schiffen und eventuellen Häfen wird vom System übernommen.

# 3 Beschreibung

**Karte:**

- **Darstellung**
  Der Hintergrund der Karte ist blau gefärbt und soll das Meer/den Ozean simulieren. Auf der gesamten Karte werden jede dritte Runde fünf Inseln zufällig/randomisiert erstellt. Diese lassen sich sowohl vom Gegner als auch vom Spieler einnehmen. Diese unterstützen, in Form von einer Kanone, die auf den jeweiligen Feind schießt, den jeweiligen Besitzer. Inseln werden durch gelbe und Häfen durch braune Pixel dargestellt. Die Häfen können den Besitzer „heilen“, d.h. +1 Leben verleihen. Die Spieler und Gegner können sich auf der Karte vollständig frei bewegen.

**Spieler:**

- **Darstellung**
  Ein drei Pixel langes Schiff in grüner, gelber oder roter Farbe (Je nach Anzahl der Leben).
- **Fähigkeiten**
  - Schussrichtung senkrecht zur Fahrtrichtung (beidseitig)
  - 3 Leben
  - 4x schneller als Gegner

**Gegner:**

- **Darstellung**
  Ein drei Pixel langes Schiff in blauer, violetter Farbe (Je nach Anzahl der Leben).
- **Fähigkeiten**
  - Schussrichtung 360°
  - 2 Leben (jede 15. Runde +1 Leben)
  - Werden jede Runde neu erzeugt (sofern gestorben in vorangegangener Runde)
  - Jede 3. Runde erhöht sich die Gegneranzahl um +1

**Insel ohne Hafen**

- **Darstellung**
  Drei mal drei große Raster in gelb.
- **Fähigkeiten**
  - Kugeln werden von Inseln geblockt
  - Kollision verursacht keinen Schaden

**Insel mit Hafen**

- **Darstellung**
  Drei mal drei große Raster in gelb mit einem braunen Pixel als Hafen.
- **Fähigkeiten**
  - Können eingenommen werden
  - Können durch Zerstörung zurückerobert werden
  - Schussrichtung 360°
  - Geben beim Andocken jede 3. Runde +1 Leben

**Steuerung**

- ↑ Bewegung um 1 Pixel in Fahrtrichtung
- ← Drehung um 45° gegen den Uhrzeigersinn
- → Drehung um 45° im Uhrzeigersinn
- ↓ Schießen (beidseitig)

**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

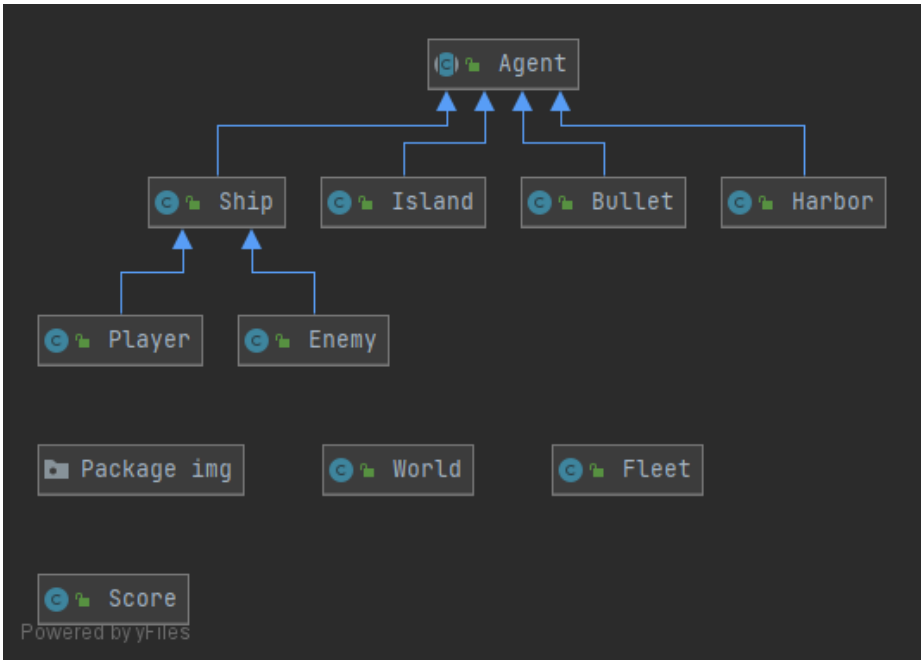
- **Vorbereitung**
  Es werden alle Spielnotwendigen Variablen deklariert und initialisiert. In einer *Do-While* Schleife wird daraufhin gestarted um mehrere Spiele hintereinander spielen zu Können. Am Start der Schleife wird die Karte, der Spieler und die Gegner erstellt und gezeichnet und auf eine 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 Arbeitsaufteilung

# 6 Klassendiagramm

Das Diagramm stellt einen groben Ausschnitt des Klassendiagramms, inklusive der Vererbung der Klassen, dar. Die ausführliche Version des Klassendiagramms befindet sich im *Documentation* Ordner.



# 7 Programmcode

### Ship.java

```
1  package de.thdeg.game.assets;
2  public class Ship extends Agent {
3      protected int hp;
4      protected final int MAXHP;
5      protected int[][] pos;
6      protected int[][] oldpos;
7      protected int align;
8      protected short[][][] color = new short[3][3][3];
9      protected Bullet bullet = null;
10     protected Bullet bullet2 = null;
11     protected boolean hadBonus = false;
12
13     Ship(int hp){
14         this.pos = new int[3][2];
15         this.oldpos = new int[3][2];
16         this.hp = hp;
17         this.MAXHP = hp;
18         this.align = 7;
19         this.pos[0][0] = 2;
20         this.pos[0][1] = 3;
21         this.pos[1][0] = 2;
22         this.pos[1][1] = 2;
23         this.pos[2][0] = 2;
24         this.pos[2][1] = 1;
25     }
26
27     Ship(int hp, int x, int y, int orient){
28         this.pos = new int[3][2];
29         this.oldpos = new int[3][2];
30         this.hp = hp;
31         this.MAXHP = hp;
32         if (x >= 1 && x <= 46 && y >= 1 && y <= 22) {
33             this.align = orient;
34             this.pos[1][0] = x;
35             this.pos[1][1] = y;
36             switch(this.align){
37                 case 1:
38                     this.pos[0][0] = x;
39                     this.pos[0][1] = y - 1;
40                     this.pos[2][0] = x;
41                     this.pos[2][1] = y + 1;
42                     break;
43                 case 2:
44                     this.pos[0][0] = x + 1;
45                     this.pos[0][1] = y - 1;
46                     this.pos[2][0] = x - 1;
```

```
47                  this.pos[2][1] = y + 1;
48                  break;
49              case 3:
50                  this.pos[0][0] = x + 1;
51                  this.pos[0][1] = y;
52                  this.pos[2][0] = x - 1;
53                  this.pos[2][1] = y;
54                  break;
55              case 4:
56                  this.pos[0][0] = x + 1;
57                  this.pos[0][1] = y + 1;
58                  this.pos[2][0] = x - 1;
59                  this.pos[2][1] = y - 1;
60                  break;
61              case 5:
62                  this.pos[0][0] = x;
63                  this.pos[0][1] = y + 1;
64                  this.pos[2][0] = x;
65                  this.pos[2][1] = y - 1;
66                  break;
67              case 6:
68                  this.pos[0][0] = x - 1;
69                  this.pos[0][1] = y + 1;
70                  this.pos[2][0] = x + 1;
71                  this.pos[2][1] = y - 1;
72                  break;
73              case 7:
74                  this.pos[0][0] = x - 1;
75                  this.pos[0][1] = y;
76                  this.pos[2][0] = x + 1;
77                  this.pos[2][1] = y;
78                  break;
79              case 8:
80                  this.pos[0][0] = x - 1;
81                  this.pos[0][1] = y - 1;
82                  this.pos[2][0] = x + 1;
83                  this.pos[2][1] = y + 1;
84                  break;
85          }
86      }else {
87          this.pos[0][0] = 2;
88          this.pos[0][1] = 3;
89          this.pos[1][0] = 2;
90          this.pos[1][1] = 2;
91          this.pos[2][0] = 2;
92          this.pos[2][1] = 1;
93          this.align = 7;
94      }
95  }
96
```

```
97      public boolean getHadBonus(){
98          return this.hadBonus;
99      }
100
101     public void resetHadBonus(){
102         this.hadBonus = false;
103     }
104
105     protected short[] clearTrace(short[] myImage){
106         for (int i = 0; i < this.oldpos.length; i++) {
107             myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 0]
                    = (short)0;
108             myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 1]
                    = (short)177;
109             myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 2]
                    = (short)241;
110         }
111         return myImage;
112     }
113
114     protected int getShipColor(){
115         double ps = (double)this.hp / (double)this.MAXHP;
116         if(ps > 0.67){
117             return 2;
118         }else if(ps > 0.34){
119             return 1;
120         }else {
121             return 0;
122         }
123     }
124
125     /**
126      * This method uses the Players values to update the map and return
           it always.
127      * @param myImage the Pixel array given from the {@link GameMain}
128      * @return the updated maparray
129      * */
130     public short[] paint(short[] myImage){
131         myImage = clearTrace(myImage);
132         if(this.hp > 0) {
133             for(int i=0; i < this.pos.length; i++){
134                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
                        color[getShipColor()][i][0];
135                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
                        color[getShipColor()][i][1];
136                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
                        color[getShipColor()][i][2];
137             }
138         }else {
139             for(int i=0; i < this.pos.length; i++){
```

```
140                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
                        0;
141                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
                        177;
142                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
                        241;
143             }
144             if(this.bullet != null){
145                 myImage = this.bullet.clear(myImage);
146             }
147         }
148         return myImage;
149     }
150 
151     public short[] isHit(short[] myImage){
152         for(int i = 0; i < this.pos.length; i++){
153             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
154                 damage(1);
155             }
156         }
157         myImage = paint(myImage);
158         return myImage;
159     }
160 
161     public boolean isAlive(){
162         return (this.hp > 0);
163     }
164 
165     protected boolean comparePixel(short r1, short g1, short b1, short
            r2, short g2, short b2){
166         return (r1 == r2 && g1 == g2 && b1 == b2);
167     }
168 
169     /**
170      * The method collide looks at the pixels of the ship and look if it
              collided with another object
171      * */
172     public int collide(short[] myImage){
173         return -1;
174     }
175 
176     /**
177      * This method takes the userinput and changes the
              position/direction of the ship
178      * @param dir represents the given userinput
179      *          0 - up
180      *          1 - down
181      *          2 - left
182      *          3 - right
183      * */
```

```
184     protected void move(int dir, short[] myImage){
185         switch(dir){
186             case 0: // Hoch
187                 forward(myImage);
188                 break;
189             case 1: // Runter
190                 shoot();
191                 break;
192             case 2:// Links
193                 rotate(0, true, myImage);
194                 break;
195             case 3: // Rechts
196                 rotate(1, true, myImage);
197                 break;
198         }
199     }
200 
201     /**
202      * This method will be called by the move method and rotates the
            ship in the given direction.
203      * @param dir represents the direction which the ship takes to
            rotate.
204      *          0 - Left
205      *          1 - Right
206      * @param save decides if the old ship position shall be saved
207      * */
208     protected void rotate(int dir, boolean save, short[] myImage){
209         if(dir == 0){ // Left
210             switch(this.align){
211                 case 1:
212                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                            47 &&
213                             !(hitIsland(myImage, this.pos[0][0] - 1,
                                    this.pos[0][1], false) > 0) &&
214                             !(hitIsland(myImage, this.pos[2][0] + 1,
                                    this.pos[2][1], false) > 0)) {
215                         if(save){saveOldPos();}
216                         this.pos[0][0]--;
217                         this.pos[2][0]++;
218                         changeAlign(-1);
219                     }
220                     break;
221                 case 2:
222                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                            47 &&
223                             !(hitIsland(myImage, this.pos[0][0] - 1,
                                    this.pos[0][1], false) > 0) &&
224                             !(hitIsland(myImage, this.pos[2][0] + 1,
                                    this.pos[2][1], false) > 0)) {
225                         if(save){saveOldPos();}
```

```
226                     this.pos[0][0]--;
227                     this.pos[2][0]++;
228                     changeAlign(-1);
229                 }
230                 break;
231             case 3:
232                 if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                        23 &&
233                     !(hitIsland(myImage, this.pos[0][0],
                            this.pos[0][1] - 1, false) > 0) &&
234                     !(hitIsland(myImage, this.pos[2][0],
                            this.pos[2][1] + 1, false) > 0)) {
235                     if(save){saveOldPos();}
236                     this.pos[0][1]--;
237                     this.pos[2][1]++;
238                     changeAlign(-1);
239                 }
240                 break;
241             case 4:
242                 if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                        23 &&
243                     !(hitIsland(myImage, this.pos[0][0],
                            this.pos[0][1] - 1, false) > 0) &&
244                     !(hitIsland(myImage, this.pos[2][0],
                            this.pos[2][1] + 1, false) > 0)) {
245                     if(save){saveOldPos();}
246                     this.pos[0][1]--;
247                     this.pos[2][1]++;
248                     changeAlign(-1);
249                 }
250                 break;
251             case 5:
252                 if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                        0 &&
253                     !(hitIsland(myImage, this.pos[0][0] + 1,
                            this.pos[0][1], false) > 0) &&
254                     !(hitIsland(myImage, this.pos[2][0] - 1,
                            this.pos[2][1], false) > 0)) {
255                     if(save){saveOldPos();}
256                     this.pos[0][0]++;
257                     this.pos[2][0]--;
258                     changeAlign(-1);
259                 }
260                 break;
261             case 6:
262                 if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                        0 &&
263                     !(hitIsland(myImage, this.pos[0][0] + 1,
                            this.pos[0][1], false) > 0) &&
264                     !(hitIsland(myImage, this.pos[2][0] - 1,
```

```
                        this.pos[2][1], false) > 0)) {
                    if(save){saveOldPos();}
                    this.pos[0][0]++;
                    this.pos[2][0]--;
                    changeAlign(-1);
                }
                break;
            case 7:
                if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                    0 &&
                        !(hitIsland(myImage, this.pos[0][0],
                            this.pos[0][1] + 1, false) > 0) &&
                        !(hitIsland(myImage, this.pos[2][0],
                            this.pos[2][1] - 1, false) > 0)) {
                    if(save){saveOldPos();}
                    this.pos[0][1]++;
                    this.pos[2][1]--;
                    changeAlign(-1);
                }
                break;
            case 8:
                if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                    0 &&
                        !(hitIsland(myImage, this.pos[0][0],
                            this.pos[0][1] + 1, false) > 0) &&
                        !(hitIsland(myImage, this.pos[2][0],
                            this.pos[2][1] - 1, false) > 0)) {
                    if(save){saveOldPos();}
                    this.pos[0][1]++;
                    this.pos[2][1]--;
                    changeAlign(-1);
                }
                break;
        }
    }else { // Right
        switch(this.align){
            case 1:
                if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                    0 &&
                        !(hitIsland(myImage, this.pos[0][0] + 1,
                            this.pos[0][1], false) > 0) &&
                        !(hitIsland(myImage, this.pos[2][0] - 1,
                            this.pos[2][1], false) > 0)) {
                    if(save){saveOldPos();}
                    this.pos[0][0]++;
                    this.pos[2][0]--;
                    changeAlign(1);
                }
                break;
            case 2:
```

```
305                 if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0 &&
306                     !(hitIsland(myImage, this.pos[0][0],
                            this.pos[0][1] + 1, false) > 0) &&
307                     !(hitIsland(myImage, this.pos[2][0],
                            this.pos[2][1] - 1, false) > 0)) {
308                 if(save){saveOldPos();}
309                 this.pos[0][1]++;
310                 this.pos[2][1]--;
311                 changeAlign(1);
312             }
313             break;
314         case 3:
315             if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
                        0 &&
316                     !(hitIsland(myImage, this.pos[0][0],
                            this.pos[0][1] + 1, false) > 0) &&
317                     !(hitIsland(myImage, this.pos[2][0],
                            this.pos[2][1] - 1, false) > 0)) {
318                 if(save){saveOldPos();}
319                 this.pos[0][1]++;
320                 this.pos[2][1]--;
321                 changeAlign(1);
322             }
323             break;
324         case 4:
325             if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                        47 &&
326                     !(hitIsland(myImage, this.pos[0][0] - 1,
                            this.pos[0][1], false) > 0) &&
327                     !(hitIsland(myImage, this.pos[2][0] + 1,
                            this.pos[2][1], false) > 0)) {
328                 if(save){saveOldPos();}
329                 this.pos[0][0]--;
330                 this.pos[2][0]++;
331                 changeAlign(1);
332             }
333             break;
334         case 5:
335             if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
                        47 &&
336                     !(hitIsland(myImage, this.pos[0][0] - 1,
                            this.pos[0][1], false) > 0) &&
337                     !(hitIsland(myImage, this.pos[2][0] + 2,
                            this.pos[2][1], false) > 0)) {
338                 if(save){saveOldPos();}
339                 this.pos[0][0]--;
340                 this.pos[2][0]++;
341                 changeAlign(1);
342             }
```

```
343                 break;
344             case 6:
345                 if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                        23 &&
346                     !(hitIsland(myImage, this.pos[0][0],
                            this.pos[0][1] - 1, false) > 0) &&
347                     !(hitIsland(myImage, this.pos[2][0],
                            this.pos[2][1] + 1, false) > 0)) {
348                     if(save){saveOldPos();}
349                     this.pos[0][1]--;
350                     this.pos[2][1]++;
351                     changeAlign(1);
352                 }
353                 break;
354             case 7:
355                 if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
                        23 &&
356                     !(hitIsland(myImage, this.pos[0][0],
                            this.pos[0][1] - 1, false) > 0) &&
357                     !(hitIsland(myImage, this.pos[2][0],
                            this.pos[2][1] + 1, false) > 0)) {
358                     if(save){saveOldPos();}
359                     this.pos[0][1]--;
360                     this.pos[2][1]++;
361                     changeAlign(1);
362                 }
363                 break;
364             case 8:
365                 if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
                        0 &&
366                     !(hitIsland(myImage, this.pos[0][0] + 1,
                            this.pos[0][1], false) > 0) &&
367                     !(hitIsland(myImage, this.pos[2][0] - 1,
                            this.pos[2][1], false) > 0)) {
368                     if(save){saveOldPos();}
369                     this.pos[0][0]++;
370                     this.pos[2][0]--;
371                     changeAlign(1);
372                 }
373                 break;
374         }
375     }
376 }
377
378 protected void rotateTo(int newOri, short[] myImage){
379     saveOldPos();
380     while (this.align != newOri) {
381         rotate(1, false, myImage);
382     }
383 }
```

```
384
385     /**
386      * Method to save the ship position from one move ago.
387      * */
388     protected void saveOldPos(){
389         for(int i = 0; i < this.pos.length; i++){
390             this.oldpos[i][0] = this.pos[i][0];
391             this.oldpos[i][1] = this.pos[i][1];
392         }
393     }
394
395     protected void damage(int amount){
396         this.hp -= amount;
397     }
398
399     /**
400      * Used to move the ship in the direction it is aligned to.
401      * */
402     protected void forward(short[] myImage){
403         if(canMove(myImage)){
404             switch(this.align){
405                 case 1:
406                     saveOldPos();
407                     for (int i = 0; i < this.pos.length; i++){
408                         this.pos[i][1]--;
409                     }
410                     break;
411                 case 2:
412                     saveOldPos();
413                     for (int i = 0; i < this.pos.length; i++){
414                         this.pos[i][0]++;
415                         this.pos[i][1]--;
416                     }
417                     break;
418                 case 3:
419                     saveOldPos();
420                     for (int i = 0; i < this.pos.length; i++){
421                         this.pos[i][0]++;
422                     }
423                     break;
424                 case 4:
425                     saveOldPos();
426                     for (int i = 0; i < this.pos.length; i++){
427                         this.pos[i][0]++;
428                         this.pos[i][1]++;
429                     }
430                     break;
431                 case 5:
432                     saveOldPos();
433                     for (int i = 0; i < this.pos.length; i++){
```

```
434                     this.pos[i][1]++;
435                 }
436                 break;
437             case 6:
438                 saveOldPos();
439                 for (int i = 0; i < this.pos.length; i++){
440                     this.pos[i][0]--;
441                     this.pos[i][1]++;
442                 }
443                 break;
444             case 7:
445                 saveOldPos();
446                 for (int i = 0; i < this.pos.length; i++){
447                     this.pos[i][0]--;
448                 }
449                 break;
450             case 8:
451                 saveOldPos();
452                 for (int i = 0; i < this.pos.length; i++){
453                     this.pos[i][0]--;
454                     this.pos[i][1]--;
455                 }
456                 break;
457         }
458     }else{
459
460     }
461 }
462
463 /**
464  * Used to determine if the ship can move forward.
465  * @return the returnvalue says, if the ship can move forward or if
        the ship would move outside the map.
466  * */
467 protected boolean canMove(short[] myImage){
468     boolean ret = false;
469     switch(this.align){
470         case 1 -> {
471             if(this.pos[0][1] - 1 >= 0 &&
                  !(hitIsland(myImage,this.pos[0][0],this.pos[0][1]-1,
                  false) > 0)){
472                 if(!hitEnemy(myImage,this.pos[0][0],this.pos[0][1]-1)){
473                     return true;
474                 }else {
475                     damage(1);
476                 }
477             }
478         }
479         case 2 -> {
480             if(this.pos[0][1] - 1 > 0 && this.pos[0][0] + 1 < 48 &&
```

```
                !(hitIsland(myImage,this.pos[0][0]+1,this.pos[0][1]-1,
                false) > 0)){
481             if(!hitEnemy(myImage,this.pos[0][0]+1,this.pos[0][1]-1)){
482                 return true;
483             }else {
484                 damage(1);
485             }
486         }
487     }
488     case 3 -> {
489         if(this.pos[0][0] + 1 < 48 &&
                !(hitIsland(myImage,this.pos[0][0]+1,this.pos[0][1],
                false) > 0)){
490             if(!hitEnemy(myImage,this.pos[0][0]+1,this.pos[0][1])){
491                 return true;
492             }else {
493                 damage(1);
494             }
495         }
496     }
497     case 4 -> {
498         if(this.pos[0][0] + 1 < 48 && this.pos[0][1] + 1 < 24 &&
                !(hitIsland(myImage,this.pos[0][0]+1,this.pos[0][1]+1,
                false) > 0)){
499             if(!hitEnemy(myImage,this.pos[0][0]+1,this.pos[0][1]+1)){
500                 return true;
501             }else {
502                 damage(1);
503             }
504         }
505     }
506     case 5 -> {
507         if(this.pos[0][1] + 1 < 24 &&
                !(hitIsland(myImage,this.pos[0][0],this.pos[0][1]+1,
                false) > 0)){
508             if(!hitEnemy(myImage,this.pos[0][0],this.pos[0][1]+1)){
509                 return true;
510             }else {
511                 damage(1);
512             }
513         }
514     }
515     case 6 -> {
516         if(this.pos[0][1] + 1 < 24 && this.pos[0][0] - 1 >= 0 &&
                !(hitIsland(myImage,this.pos[0][0]-1,this.pos[0][1]+1,
                false) > 0)){
517             if(!hitEnemy(myImage,this.pos[0][0]-1,this.pos[0][1]+1)){
518                 return true;
519             }else {
520                 damage(1);
```

```
521                 }
522             }
523         }
524         case 7 -> {
525             if(this.pos[0][0] - 1 >= 0 &&
                    !(hitIsland(myImage,this.pos[0][0]-1,this.pos[0][1],
                    false) > 0)){
526                 if(!hitEnemy(myImage,this.pos[0][0]-1,this.pos[0][1])){
527                     return true;
528                 }else {
529                     damage(1);
530                 }
531             }
532         }
533         case 8 -> {
534             if(this.pos[0][0] - 1 >= 0 && this.pos[0][1] - 1 >= 0 &&
                    !(hitIsland(myImage,this.pos[0][0]-1,this.pos[0][1]-1,
                    false) > 0)){
535                 if(!hitEnemy(myImage,this.pos[0][0]-1,this.pos[0][1]-1)){
536                     return true;
537                 }else {
538                     damage(1);
539                 }
540             }
541         }
542     }
543     return ret;
544 }
545 /**
546  * 8 1 2
547  * 7   3
548  * 6 5 4
549  * */
550 protected void shoot() {
551     int dir1 = (this.align + 2 > 8) ? (this.align + 2 - 8) :
            (this.align + 2);
552     int dir2 = (this.align - 2 < 1) ? (8 + this.align - 2) :
            (this.align - 2);
553     if(this.bullet == null){
554         this.bullet = new Bullet(dir1, 5, this.pos[1][0],
                this.pos[1][1], 0);
555     }
556     if(this.bullet2 == null){
557         this.bullet2 = new Bullet(dir2, 5, this.pos[1][0],
                this.pos[1][1], 0);
558     }
559 }
560
561 protected boolean hitPlayer(short[] myImage, int x, int y){
562     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
```

```
563             int idx = (y * 48 + x) * 3;
564             return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
                    myImage[idx + 2] == 36) ||
565                 (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
                        myImage[idx + 2] == 36) ||
566                 (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
                        myImage[idx + 2] == 36) ||
567                 (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
                        myImage[idx + 2] == 22) ||
568                 (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
                        myImage[idx + 2] == 22) ||
569                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
                        myImage[idx + 2] == 21) ||
570                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
                        myImage[idx + 2] == 11) ||
571                 (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
                        myImage[idx + 2] == 10) ||
572                 (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 11);
573         }else {
574             return false;
575         }
576     }
577
578     protected int hitIsland(short[] myImage, int x, int y, boolean
            harbor){
579         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
580             int idx = (y * 48 + x) * 3;
581             if(harbor){
582                 return ((myImage[idx + 0] == 125 && myImage[idx + 1] ==
                        66 && myImage[idx + 2] == 24) ||
583                     (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
                            myImage[idx + 2] == 35) ||
584                     (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
                            myImage[idx + 2] == 221)) ? 1 : 0;
585             }else {
586             return ((myImage[idx + 0] == 196 && myImage[idx + 1] == 156
                    && myImage[idx + 2] == 53) ||
587                     (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
                            myImage[idx + 2] == 48) ||
588                     (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
                            myImage[idx + 2] == 24) ||
589                     (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
                            myImage[idx + 2] == 35) ||
590                     (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
                            myImage[idx + 2] == 221)) ? 1 : 0;
591             }
592         }else {
593             return 0;
594         }
```

```
595     }
596
597     protected boolean hitEnemy(short[] myImage, int x, int y){
598         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
599             int idx = (y * 48 + x) * 3;
600             return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
                        myImage[idx + 2] == 222) ||
601                 (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
                        myImage[idx + 2] == 143) ||
602                 (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
                        myImage[idx + 2] == 74) ||
603                 (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
                        myImage[idx + 2] == 215) ||
604                 (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
                        myImage[idx + 2] == 134) ||
605                 (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 72) ||
606                 (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
                        myImage[idx + 2] == 209) ||
607                 (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
                        myImage[idx + 2] == 128) ||
608                 (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
                        myImage[idx + 2] == 74);
609         }else {
610             return false;
611         }
612     }
613
614     protected boolean hitBullet(short[] myImage, int x, int y){
615         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
616             int idx = (y * 48 + x) * 3;
617             return (myImage[idx + 0] == 12 && myImage[idx + 1] == 13 &&
                        myImage[idx + 2] == 12);
618         }else {
619             return false;
620         }
621     }
622
623     /**
624      *  0 - Enemy
625      *  1 - Player
626      * */
627     private boolean isCollectable(short[] myImage, int idx, int who){
628         if(who == 0){
629             return (myImage[idx] == 29 && myImage[idx + 1] == 67 &&
                        myImage[idx + 2] == 220);
630         }else {
631             return (myImage[idx] == 121 && myImage[idx + 1] == 235 &&
                        myImage[idx + 2] == 34);
632         }
```

```
633     }
634
635     protected boolean collectBonus(short[] myImage, int who){
636         boolean ret = false;
637         for (int k = 0; k < this.pos.length; k++){
638             for(int i = -1; i <= 1; i++){
639                 for(int j = -1; j <= 1; j++){
640                     if((this.pos[k][1]+i) >= 0 && (this.pos[k][1]+i) <= 23
                            && (this.pos[k][0]+j) <= 47 && (this.pos[k][0]+j)
                            >= 0 && isCollectable(myImage,
                            (((this.pos[k][1]+i) * 48 + (this.pos[k][0]+j)) *
                            3), who)){
641                         ret = true;
642                     }
643                 }
644             }
645         }
646         return ret;
647     }
648
649     /**
650      * This method is used to set the align variable after a succesful
            rotation
651      * @param dir the direction the ship rotates to
652      * */
653     protected void changeAlign(int dir){
654         this.align += dir;
655         if(this.align < 1){
656             this.align = 8;
657         }
658         if(this.align > 8){
659             this.align = 1;
660         }
661     }
662
663     public short[] run(int key, short[] myImage){
664         myImage = isHit(myImage);
665         if(key != -1){
666             myImage = clearTrace(myImage);
667             System.out.println("k " + key + " |a " + this.align);
668             print("beforeif");
669             move(key,myImage);
670             if (collide(myImage) == 1){
671                 resetMove();
672                 if(key == 2){
673                     this.align++;
674                 }
675                 if(key == 3){
676                     this.align--;
677                 }
```

```
678             }
679             System.out.println("k " + key + " |a " + this.align);
680             print("afterif");
681         }
682         if(this.bullet != null){
683             if(this.bullet.getRange() > 0){
684                 myImage = this.bullet.run(-1, myImage);
685             }else{
686                 this.bullet = null;
687             }
688         }
689         if(this.bullet2 != null){
690             if(this.bullet2.getRange() > 0){
691                 myImage = this.bullet2.run(-1, myImage);
692             }else{
693                 this.bullet2 = null;
694             }
695         }
696         myImage = paint(myImage);
697         return myImage;
698     }
699 
700     protected void resetMove(){
701         for(int i=0; i < this.pos.length; i++){
702             for(int j=0; j < this.pos[i].length; j++){
703                 this.pos[i][j] = this.oldpos[i][j];
704             }
705         }
706     }
707 
708     /**
709      * Debug method to print shiplocation and locationdifference between
             the new and old location.
710      * */
711     public void print(String where){
712         System.out.println(where + "\nA: " + this.align);
713         for (int i = 0; i < this.pos.length; i++){
714             System.out.println("X: " + this.pos[i][0] + " Y: " +
                    this.pos[i][1] + " | Xo: " + this.oldpos[i][0] + " Yo: "
                    + this.oldpos[i][1]);
715             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                    - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                    this.oldpos[i][1]));
716         }
717     }
718 
719     public int[][] getPos(){
720         return this.pos;
721     }
722 
```

```
723     public int getHp(){
724         return this.hp;
725     }
726 
727     public void setHp(int hp){
728         this.hp = (hp >= 0)? hp : 0;
729     }
730 
731     protected void changeColor(short[][][] rgbs){
732         for (int i = 0; i < this.color.length; i++) {
733             for (int j = 0; j < this.color[0].length; j++){
734                 for (int k = 0; k < this.color[0][0].length; k++){
735                     this.color[i][j][k] = (rgbs[i][j][k] <= 255 &&
                            rgbs[i][j][k] >= 0)? rgbs[i][j][k] : 0;
736                 }
737             }
738         }
739     }
740 }
```

**Agent.java**

```
1  package de.thdeg.game.assets;
2  public abstract class Agent {
3
4      abstract short[] paint(short[] myImage);
5
6      abstract int collide(short[] myImage);
7
8      abstract void move(int dir,short[] myImage);
9
10     abstract short[] run(int key, short[] myImage);
11 }
```

**Enemy.java**

```
1
2  package de.thdeg.game.assets;
3  import java.util.List;
4  import java.util.ArrayList;
5  public class Enemy extends Ship {
6      private int range;
7      private int dmg = 0;
8      private int PX;
9      private int PY;
10     private int RouteX = -1;
11     private int RouteY = -1;
12     private boolean detectedPlayer = false;
13     private List<int[]> routing = new ArrayList<int[]>();
14
15     Enemy(int hp){
16         super(hp);
17         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}}};
18         changeColor(rgbs);
19         int[][] pos = { {22, 22}, {23, 22}, {24, 22} };
20         this.pos = pos;
21     }
22
23     Enemy(int hp, int x, int y, int o, int r){
24         super(hp, x, y, o);
25         short[][][] rgbs = {{{31, 222, 215},{21, 138, 134},{11, 74,
               72}},{{153, 23, 209},{94, 15, 128},{55, 10, 74}},{{31, 69,
               222},{19, 43, 143},{10, 22, 74}}};
26         changeColor(rgbs);
27         this.range = r;
28     }
29
30     public void resetDmg(){
31         this.dmg = 0;
32     }
33
34     public int getDamageReceived(){
35         return this.dmg;
36     }
37
38     protected int hitIsland(short[] myImage, int x, int y, boolean
           harbor){
39         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
40             int idx = (y * 48 + x) * 3;
41             if(harbor){
42                 return ((myImage[idx + 0] == 125 && myImage[idx + 1] ==
                       66 && myImage[idx + 2] == 24) ||
```

```
43                  (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
                        myImage[idx + 2] == 35) ||
44                  (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
                        myImage[idx + 2] == 221)) ? (((myImage[idx + 0]
                        == 29 && myImage[idx + 1] == 67 && myImage[idx +
                        2] == 220)) ? 2 : 1) : 0;
45              }else {
46              return ((myImage[idx + 0] == 196 && myImage[idx + 1] == 156
                    && myImage[idx + 2] == 53) ||
47                  (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
                        myImage[idx + 2] == 48) ||
48                  (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
                        myImage[idx + 2] == 24) ||
49                  (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
                        myImage[idx + 2] == 35) ||
50                  (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
                        myImage[idx + 2] == 221)) ? (((myImage[idx + 0]
                        == 29 && myImage[idx + 1] == 67 && myImage[idx +
                        2] == 220)) ? 2 : 1) : 0;
51              }
52          }else {
53              return 0;
54          }
55      }
56
57      /**
58       *  Create Routes
59       * */
60      private void pathFinder(){
61          this.routing = new ArrayList<int[]>();
62          /**
63           *  Start:  this.pos[1][0]=x
64           *          this.pos[1][1]=y
65           *  End:    this.RouteX
66           *          this.RouteY
67           * */
68          this.PX = this.RouteX;
69          this.PY = this.RouteY;
70          int pX = this.pos[1][0];
71          int pY = this.pos[1][1];
72          while(pX != this.RouteX && pY != this.RouteY){
73              switch(routeDirection(pX, pY, this.RouteX, this.RouteY)){
74                  case 1 -> {
75                      int[] rt = {pX, --pY};
76                      this.routing.add(rt);
77                  }
78                  case 2 -> {
79                      int[] rt = {++pX, --pY};
80                      this.routing.add(rt);
81                  }
```

```
82                case 3 -> {
83                    int[] rt = {++pX, pY};
84                    this.routing.add(rt);
85                }
86                case 4 -> {
87                    int[] rt = {++pX, ++pY};
88                    this.routing.add(rt);
89                }
90                case 5 -> {
91                    int[] rt = {pX, ++pY};
92                    this.routing.add(rt);
93                }
94                case 6 -> {
95                    int[] rt = {--pX, ++pY};
96                    this.routing.add(rt);
97                }
98                case 7 -> {
99                    int[] rt = {--pX, pY};
100                   this.routing.add(rt);
101               }
102               case 8 -> {
103                   int[] rt = {--pX, --pY};
104                   this.routing.add(rt);
105               }
106               default -> {
107                   break;
108               }
109           }
110       }
111   }
112
113   public short[] run(short[] myImage){
114       // if(this.RouteX != -1 && this.RouteY != -1){
115       //     pathFinder();
116       //     pR();
117       //     System.out.println(this.routing.get(0)[0]);
118       // }
119       if(collectBonus(myImage, 0) && !this.hadBonus){
120           this.hp += (this.hp < this.MAXHP) ? 1 : 0;
121           this.hadBonus = true;
122       }
123       myImage = clearTrace(myImage);
124       if(inVision(myImage, 0) && this.bullet == null){
125           shoot();
126       }
127       move(myImage);
128       if (collide(myImage) != 0){
129           resetMove();
130       }
131       if(this.bullet != null){
```

```
132             if(this.bullet.getRange() > 0){
133                 myImage = this.bullet.run(-1, myImage);
134             }else{
135                 this.bullet = null;
136             }
137         }
138         if(this.bullet2 != null){
139             if(this.bullet2.getRange() > 0){
140                 myImage = this.bullet2.run(-1, myImage);
141             }else{
142                 this.bullet2 = null;
143             }
144         }
145         myImage = paint(myImage);
146         return myImage;
147     }
148 
149     private int routeDirection(int x, int y, int gx, int gy){
150         if(x > gx){
151             if(y > gy){
152                 return 8;
153             }else if(y < gy){
154                 return 6;
155             }else {
156                 return 7;
157             }
158         }else if(x < gx){
159             if(y > gy) {
160                 return 2;
161             }else if(y < gy){
162                 return 4;
163             }else {
164                 return 3;
165             }
166         }else {
167             if(y > gy) {
168                 return 1;
169             }else if(y < gy){
170                 return 5;
171             }
172         }
173         return -1;
174     }
175 
176     // private int routeDirection(int x, int y, int[] gPos){
177     //     if(x > gPos[0]){
178             // if(y > gPos[1]){
179             //     return 8;
180             // }else if(y < gPos[1]){
181             //     return 6;
```

```
182             // }else {
183             //     return 7;
184             // }
185         // }else if(x < gPos[0]){
186             // if(y > gPos[1]) {
187             //     return 2;
188             // }else if(y < gPos[1]){
189             //     return 4;
190             // }else {
191             //     return 3;
192             // }
193         // }else {
194             // if(y > gPos[1]) {
195             //     return 1;
196             // }else if(y < gPos[1]){
197             //     return 5;
198             // }
199         // }
200         // return -1;
201     // }
202 
203     public void shoot(){
204         int orient = routeDirection(this.pos[1][0], this.pos[1][1],
                this.PX, this.PY);
205         if(this.bullet == null){
206             this.bullet = new Bullet(orient, 8, this.pos[1][0],
                    this.pos[1][1], 1);
207         }
208     }
209 
210     private void move(short[] myImage){
211         // if(this.routing.size() <= 0 ){
212         if(canMove(myImage)){
213             forward(myImage);
214         }else{
215             if(Math.random() > 0.5){
216                 rotate(0, true, myImage);
217             }else {
218                 rotate(1, true, myImage);
219             }
220         }
221         // }else {
222         //    switch(routeDirection(this.pos[1][0], this.pos[1][1],
                this.routing.get(this.routing.size() - 1)[0],
                this.routing.get(this.routing.size() - 1)[1])){
223         //        case 1 -> {
224         //            rotateTo(1);
225         //            forward(myImage);
226         //        }
227         //        case 2 -> {
```

```
228         //            rotateTo(2);
229         //            forward(myImage);
230         //        }
231         //        case 3 -> {
232         //            rotateTo(3);
233         //            forward(myImage);
234         //        }
235         //        case 4 -> {
236         //            rotateTo(4);
237         //            forward(myImage);
238         //        }
239         //        case 5 -> {
240         //            rotateTo(5);
241         //            forward(myImage);
242         //        }
243         //        case 6 -> {
244         //            rotateTo(6);
245         //            forward(myImage);
246         //        }
247         //        case 7 -> {
248         //            rotateTo(7);
249         //            forward(myImage);
250         //        }
251         //        case 8 -> {
252         //            rotateTo(8);
253         //            forward(myImage);
254         //        }
255         //        default -> {}
256         //    }
257         // this.routing.remove(this.routing.size() - 1);
258         // }
259     }
260 
261     private void pR(){
262         for(int[] i : this.routing){
263             System.out.println("| " + i[0] + " | " + i[1] + " |");
264         }
265     }
266 
267 
268     /**
269      * Method to detect if the player is visible for the enemy ship.
270      * */
271     private boolean inVision(short[] myImage, int who){
272         int difx;
273         int dify;
274         if(this.hp > 0){
275             for (int i = 0 - this.range; i <= this.range; i++) {
276                 difx = this.pos[1][0] + i;
277                 for (int j = 0 - this.range; j <= this.range; j++) {
```

```
278                     dify = this.pos[1][1] + j;
279                     if ((Math.pow(difx - this.pos[0][1], 2)+Math.pow(dify
                            - this.pos[1][1], 2)) <= Math.pow(this.range, 2))
                            {
280                         switch(who){
281                             case 0 -> {
282                                 if(hitPlayer(myImage, difx, dify)) {
283                                     this.detectedPlayer = true;
284                                     this.PX = difx;
285                                     this.PY = dify;
286                                     return true;
287                                 }
288                             }
289                             case 1 -> {
290                                 if(hitIsland(myImage, difx, dify, true) >
                                        0){
291                                     rotateTo(routeDirection(this.pos[1][0],
                                         this.pos[1][1], difx, dify),
                                         myImage);
292                                 }
293                             }
294                         }
295                     }
296                 }
297             }
298         }
299         return false;
300     }
301
302     public void setRouteX(int PX){
303         this.PX = PX;
304     }
305
306     public void setRouteY(int PY){
307         this.PY = PY;
308     }
309
310     public int getPX(){
311         return this.PX;
312     }
313
314     public int getPY(){
315         return this.PY;
316     }
317
318     public boolean getPlayerDetected(){
319         return this.detectedPlayer;
320     }
321
322     /**
```

```
323      * The method collide looks at the pixels of the ship and look if it
             collided with another object
324      * */
325     public int collide(short[] myImage){
326         int ret = 0;
327         for(int i=0; i < this.pos.length; i++){
328             int idx = (this.pos[i][1] * 48 + this.pos[i][0]) * 3;
329             if (hitIsland(myImage, this.pos[i][0], this.pos[i][1], false)
                    > 0){
330                 ret = 1;
331             }
332             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
333                 damage(1);
334                 ret = 2;
335             }
336             if(hitPlayer(myImage, this.pos[i][0], this.pos[i][1])) {
337                 damage(1);
338                 this.dmg += 1;
339                 ret = 1;
340             }
341         }
342         return ret;
343     }
344 
345     public boolean includesPos(int x, int y){
346         for (int i = 0; i < this.pos.length; i++){
347             if(this.pos[i][0] == x && this.pos[i][1] == y){
348                 return true;
349             }
350         }
351         return false;
352     }
353 
354     /**
355      * Debug method to print shiplocation and locationdifference between
             the new and old location.
356      * */
357     public void print(){
358         System.out.println("Enemy ship:\nA: " + this.align);
359         for (int i = 0; i < this.pos.length; i++){
360             System.out.println("X: " + this.pos[i][0] + " Y: " +
                    this.pos[i][1]);
361             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
                    this.oldpos[i][1]);
362             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                    - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                    this.oldpos[i][1]));
363         }
364     }
365 }
```

**Fleet.java**

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

```
47                  e.resetHadBonus();
48              }
49          }
50      }
51
52      public boolean getHadBonus(){
53          boolean ret = false;
54          for(Enemy e : this.fleet){
55              if(e.isAlive() && e.getHadBonus()){
56                  ret = true;
57              }
58          }
59          return ret;
60      }
61
62      public int getNumberOfAliveShips(){
63          int ret = 0;
64          for (Enemy s : this.fleet){
65              if(s.isAlive()){
66                  ret++;
67              }
68          }
69          return ret;
70      }
71
72      public void resetDamageControl(){
73          for (Enemy e : this.fleet){
74              e.resetDmg();
75          }
76      }
77      public int damageControl(){
78          int ret = 0;
79          for (Enemy e : this.fleet){
80              ret += e.getDamageReceived();
81          }
82          return ret;
83      }
84
85      public void distributeDamage(int x, int y){
86          for (Enemy e : fleet){
87              if(e.includesPos(x, y)){
88                  e.damage(1);
89                  break;
90              }
91          }
92      }
93
94      public int getDead(){
95          int ret = 0;
96          for (Enemy e : fleet){
```

```
97              if(!e.isAlive()){
98                  ret++;
99              }
100         }
101         return ret;
102     }
103 
104     public void printing(){
105         for(Enemy e : this.fleet){
106             e.print("text");
107         }
108     }
109 
110     private void broadcastPosition(){
111         for(Enemy e : this.fleet){
112             if(e.getPlayerDetected()){
113                 this.detected = true;
114                 this.PX = e.getPX();
115                 this.PY = e.getPY();
116             }
117         }
118     }
119 
120     public short[] executeOrders(short[] myImage){
121         broadcastPosition();
122         for (Enemy s : this.fleet){
123             if(s.isAlive()){
124                 s.setRouteX(this.PX);
125                 s.setRouteY(this.PY);
126                 myImage = s.run(myImage);
127             }else {
128                 myImage = s.paint(myImage);
129             }
130         }
131         return myImage;
132     }
133 
134     public short[] statusUpdate(short[] myImage){
135         for(Enemy e : this.fleet){
136             if(e.isAlive()){
137                 myImage = e.isHit(myImage);
138             }
139         }
140         return myImage;
141     }
142 
143     public short[] paintFleet(short[] myImage){
144         for (Enemy s : this.fleet) {
145             if(s.isAlive()){
146                 myImage = s.paint(myImage);
```

```
147             }
148         }
149         return myImage;
150     }
151 }
```

**Bullet.java**

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

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

```
81              (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
                    myImage[idx + 2] == 209) ||
82              (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
                    myImage[idx + 2] == 128) ||
83              (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
                    myImage[idx + 2] == 74);
84          }else {
85              return false;
86          }
87      }
88
89      protected boolean hitIsland(short[] myImage, int x, int y){
90          if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
91              int idx = (y * 48 + x) * 3;
92              return (myImage[idx + 0] == 196 && myImage[idx + 1] == 156 &&
                    myImage[idx + 2] == 53) ||
93              (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
                    myImage[idx + 2] == 48) ||
94              (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
                    myImage[idx + 2] == 24);
95          }else {
96              return false;
97          }
98      }
99
100
101     public void move(int dir,short[] myImage){}
102
103     private void saveOldPos(){
104         this.oldpos[0] = this.pos[0];
105         this.oldpos[1] = this.pos[1];
106     }
107
108     public boolean move(){
109         if(canMove()){
110             saveOldPos();
111             switch(this.direction){
112                 case 1:
113                     this.pos[1]--;
114                     break;
115                 case 2:
116                     this.pos[0]++;
117                     this.pos[1]--;
118                     break;
119                 case 3:
120                     this.pos[0]++;
121                     break;
122                 case 4:
123                     this.pos[0]++;
124                     this.pos[1]++;
```

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

```
172             if(this.origin == Origin.SHIP){
173                 if (!(hitEnemy(myImage, this.pos[0], this.pos[1]) ||
                        hitPlayer(myImage, this.pos[0], this.pos[1]) ||
                        hitIsland(myImage, this.pos[0], this.pos[1]))){
174                     if(move()){
175                         myImage = paint(myImage);
176                     }else {
177                         this.range = 0;
178                         myImage = clear(myImage);
179                     }
180                 }else{
181                     this.range = 0;
182                     this.hasHit = true;
183                     myImage = paint(myImage);
184                 }
185             }else if(this.origin == Origin.ISLAND){
186                 if (!(hitEnemy(myImage, this.pos[0], this.pos[1]) ||
                        hitPlayer(myImage, this.pos[0], this.pos[1]))){
187                     if(move()){
188                         myImage = paint(myImage);
189                     }else {
190                         this.range = 0;
191                         myImage = clear(myImage);
192                     }
193                 }else{
194                     this.range = 0;
195                     this.hasHit = true;
196                     myImage = paint(myImage);
197                 }
198             }
199         }
200         return myImage;
201     }
202 
203     public boolean getHasHit(){
204         return this.hasHit;
205     }
206 
207     public int getRange(){
208         return this.range;
209     }
210 }
```

**Harbor.java**

```
1  package de.thdeg.game.assets;
2  public class Harbor extends Agent{
3      protected short[][] color = {{125, 66, 24},{122, 236, 35},{30, 68,
           221}};
4      protected int hasBonus = 0;
5      protected int orient;
6      protected boolean captured = false;
7      protected int possession = -1;
8      protected int[] pos;
9      protected int[] enemyPos;
10     protected Bullet bullet = null;
11
12     Harbor(int orient){
13         this.orient = orient;
14         this.pos = new int[2];
15         this.enemyPos = new int[2];
16     }
17
18     public int getOrient(){
19         return this.orient;
20     }
21
22     public short[][] getColor() {
23         return this.color;
24     }
25
26     public int getPossession(){
27         return this.possession;
28     }
29
30     public void setPos(int y, int x){
31         if(x >= 0 && x <= 47){
32             this.pos[0] = x;
33         }
34         if(y >= 0 && y <= 23){
35             this.pos[1] = y;
36         }
37     }
38
39     public short[] reset(short[] myImage){
40         this.possession = -1;
41         this.captured = false;
42         this.hasBonus = 0;
43         return myImage;
44     }
45
46     @Override
47     short[] paint(short[] myImage) {
```

```
48          return new short[0];
49      }
50
51      @Override
52      int collide(short[] myImage) {
53          return -1;
54      }
55
56      public short[] isHit(short[] myImage){
57          if (hitBullet(myImage, this.pos[0], this.pos[1])){
58              myImage = reset(myImage);
59          }
60          return myImage;
61      }
62
63      @Override
64      void move(int dir,short[] myImage) {
65      }
66
67      protected boolean hitBullet(short[] myImage, int x, int y){
68          if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
69              int idx = (y * 48 + x) * 3;
70              return (myImage[idx + 0] == 12 && myImage[idx + 1] == 13 &&
                        myImage[idx + 2] == 12);
71          }else {
72              return false;
73          }
74      }
75
76      public int getHasBonus(){
77          return this.hasBonus;
78      }
79
80      public void setHasBonus(int hasBonus) {
81          this.hasBonus = hasBonus;
82      }
83
84      @Override
85      short[] run(int key, short[] myImage) {
86          if(this.captured){
87              myImage = isHit(myImage);
88              if(this.captured && detectShip(15, myImage) ==
                        this.possession) {
89                  shoot();
90              }
91              if(this.captured && detectShip(1, myImage) ==
                        Math.abs(this.possession - 1)){
92                  this.hasBonus = 0;
93              }
94          } else {
```

```
95              int poss = detectShip(7, myImage);
96              if(poss != -1) {
97                  this.captured = true;
98                  this.hasBonus = 1;
99                  if(poss == 0) {
100                     this.possession = 1;
101                 }else {
102                     this.possession = 0;
103                 }
104             }
105         }
106         if(this.bullet != null){
107             if(this.bullet.getRange() > 0){
108                 myImage = this.bullet.run(-1, myImage);
109             }else{
110                 this.bullet = null;
111             }
112         }
113         return myImage;
114     }
115     private int routeDirection(int x, int y, int[] gPos){
116         if(x > gPos[0]){
117             if(y > gPos[1]){
118                 return 8;
119             }else if(y < gPos[1]){
120                 return 6;
121             }else {
122                 return 7;
123             }
124         }else if(x < gPos[0]){
125             if(y > gPos[1]) {
126                 return 2;
127             }else if(y < gPos[1]){
128                 return 4;
129             }else {
130                 return 3;
131             }
132         }else {
133             if(y > gPos[1]) {
134                 return 1;
135             }else if(y < gPos[1]){
136                 return 5;
137             }
138         }
139         return -1;
140     }
141     protected void shoot() {
142         int orient = routeDirection(this.pos[0], this.pos[1],
                this.enemyPos);
143         if(this.bullet == null){
```

```
144             this.bullet = new Bullet(orient, 8, this.pos[0], this.pos[1],
                    1);
145         }
146     }
147     protected boolean hitPlayer(short[] myImage, int x, int y){
148         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
149             int idx = (y * 48 + x) * 3;
150             return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
                        myImage[idx + 2] == 36) ||
151                 (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
                        myImage[idx + 2] == 36) ||
152                 (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
                        myImage[idx + 2] == 36) ||
153                 (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
                        myImage[idx + 2] == 22) ||
154                 (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
                        myImage[idx + 2] == 22) ||
155                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
                        myImage[idx + 2] == 21) ||
156                 (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
                        myImage[idx + 2] == 11) ||
157                 (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
                        myImage[idx + 2] == 10) ||
158                 (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 11);
159         }else {
160             return false;
161         }
162     }
163     protected boolean hitEnemy(short[] myImage, int x, int y){
164         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
165             int idx = (y * 48 + x) * 3;
166             return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
                        myImage[idx + 2] == 222) ||
167                 (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
                        myImage[idx + 2] == 143) ||
168                 (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
                        myImage[idx + 2] == 74) ||
169                 (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
                        myImage[idx + 2] == 215) ||
170                 (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
                        myImage[idx + 2] == 134) ||
171                 (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
                        myImage[idx + 2] == 72) ||
172                 (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
                        myImage[idx + 2] == 209) ||
173                 (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
                        myImage[idx + 2] == 128) ||
174                 (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
                        myImage[idx + 2] == 74);
```

```
175         }else {
176             return false;
177         }
178     }
179     protected int detectShip(int range, short[] myImage){
180         int difx;
181         int dify;
182         for (int i = -range; i <= range; i++) {
183             difx = this.pos[0] + i;
184             for (int j = -range; j <= range; j++) {
185                 dify = this.pos[1] + j;
186                 if ((Math.pow(difx - this.pos[0], 2)+Math.pow(dify -
                        this.pos[1], 2)) <= Math.pow(range, 2)) {
187                     if(hitPlayer(myImage, difx, dify)) {
188                         this.enemyPos[0] = difx;
189                         this.enemyPos[1] = dify;
190                         return 1;
191                     }
192                     if(hitEnemy(myImage, difx, dify)) {
193                         this.enemyPos[0] = difx;
194                         this.enemyPos[1] = dify;
195                         return 0;
196                     }
197                 }
198             }
199         }
200         return -1;
201     }
202 }
```

**Island.java**

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

```
                    this.pos[i1][i2][0]) * 3 + 0] = this.color[1][0];
                    // (y * 48 + x) * 3 + 0
42                  myImage[(this.pos[i1][i2][1] * 48 +
                    this.pos[i1][i2][0]) * 3 + 1] = this.color[1][1];
                    // (y * 48 + x) * 3 + 1
43                  myImage[(this.pos[i1][i2][1] * 48 +
                    this.pos[i1][i2][0]) * 3 + 2] = this.color[1][2];
                    // (y * 48 + x) * 3 + 2
44                  switch(harbor.getOrient()){
45                      case 1:
46                          myImage[(this.pos[0][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length/2)][0])
                                * 3 + 0] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][0] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 0
47                          myImage[(this.pos[0][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length/2)][0])
                                * 3 + 1] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][1] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 1
48                          myImage[(this.pos[0][(int)(this.pos[0].length/2)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length/2)][0])
                                * 3 + 2] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][2] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 2
49                          harbor.setPos(this.pos[0][(int)(this.pos[0].length/2)][1],this.pos[0][(int)(t
50                          break;
51                      case 2:
52                          myImage[(this.pos[0][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length-1)][0])
                                * 3 + 0] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][0] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 0
53                          myImage[(this.pos[0][(int)(this.pos[0].length-1)][1]
                                * 48 +
                                this.pos[0][(int)(this.pos[0].length-1)][0])
                                * 3 + 1] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][1] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 1
54                          myImage[(this.pos[0][(int)(this.pos[0].length-1)][1]
                                * 48 +
```

```
                    this.pos[0][(int)(this.pos[0].length-1)][0])
                    * 3 + 2] =
                    (short)(harbor.getColor()[harbor.getPossession()
                    + 1][2] - harbor.getHasBonus()); // (y *
                    48 + x) * 3 + 2
55                  harbor.setPos(this.pos[0][(int)(this.pos[0].length-1)][1],this.pos[0][(int)(t
56                  break;
57              case 3:
58                  myImage[(this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][1]
                    * 48 +
                    this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][0])
                    * 3 + 0] =
                    (short)(harbor.getColor()[harbor.getPossession()
                    + 1][0] - harbor.getHasBonus()); // (y *
                    48 + x) * 3 + 0
59                  myImage[(this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][1]
                    * 48 +
                    this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][0])
                    * 3 + 1] =
                    (short)(harbor.getColor()[harbor.getPossession()
                    + 1][1] - harbor.getHasBonus()); // (y *
                    48 + x) * 3 + 1
60                  myImage[(this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][1]
                    * 48 +
                    this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)][0])
                    * 3 + 2] =
                    (short)(harbor.getColor()[harbor.getPossession()
                    + 1][2] - harbor.getHasBonus()); // (y *
                    48 + x) * 3 + 2
61                  harbor.setPos(this.pos[(int)(this.pos.length/2)][(int)(this.pos[0].length-1)]
62                  break;
63              case 4:
64                  myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][1]
                    * 48 +
                    this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][0])
                    * 3 + 0] =
                    (short)(harbor.getColor()[harbor.getPossession()
                    + 1][0] - harbor.getHasBonus()); // (y *
                    48 + x) * 3 + 0
65                  myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][1]
                    * 48 +
                    this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][0])
                    * 3 + 1] =
                    (short)(harbor.getColor()[harbor.getPossession()
                    + 1][1] - harbor.getHasBonus()); // (y *
                    48 + x) * 3 + 1
66                  myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][1]
                    * 48 +
                    this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)][0])
                    * 3 + 2] =
```

```
                        (short)(harbor.getColor()[harbor.getPossession()
                        + 1][2] - harbor.getHasBonus()); // (y *
                        48 + x) * 3 + 2
67                  harbor.setPos(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length-1)]
68                  break;
69              case 5:
70                  myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][1]
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][0])
                        * 3 + 0] =
                        (short)(harbor.getColor()[harbor.getPossession()
                        + 1][0] - harbor.getHasBonus()); // (y *
                        48 + x) * 3 + 0
71                  myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][1]
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][0])
                        * 3 + 1] =
                        (short)(harbor.getColor()[harbor.getPossession()
                        + 1][1] - harbor.getHasBonus()); // (y *
                        48 + x) * 3 + 1
72                  myImage[(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][1]
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)][0])
                        * 3 + 2] =
                        (short)(harbor.getColor()[harbor.getPossession()
                        + 1][2] - harbor.getHasBonus()); // (y *
                        48 + x) * 3 + 2
73                  harbor.setPos(this.pos[(int)(this.pos.length-1)][(int)(this.pos[0].length/2)]
74                  break;
75              case 6:
76                  myImage[(this.pos[(int)(this.pos.length-1)][0][1]
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][0][0])
                        * 3 + 0] =
                        (short)(harbor.getColor()[harbor.getPossession()
                        + 1][0] - harbor.getHasBonus()); // (y *
                        48 + x) * 3 + 0
77                  myImage[(this.pos[(int)(this.pos.length-1)][0][1]
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][0][0])
                        * 3 + 1] =
                        (short)(harbor.getColor()[harbor.getPossession()
                        + 1][1] - harbor.getHasBonus()); // (y *
                        48 + x) * 3 + 1
78                  myImage[(this.pos[(int)(this.pos.length-1)][0][1]
                        * 48 +
                        this.pos[(int)(this.pos.length-1)][0][0])
                        * 3 + 2] =
                        (short)(harbor.getColor()[harbor.getPossession()
                        + 1][2] - harbor.getHasBonus()); // (y *
```

```
                                48 + x) * 3 + 2
79                          harbor.setPos(this.pos[(int)(this.pos.length-1)][0][1],this.pos[(int)(this.po
80                          break;
81                      case 7:
82                          myImage[(this.pos[(int)(this.pos.length/2)][0][1]
                                * 48 +
                                this.pos[(int)(this.pos.length/2)][0][0])
                                * 3 + 0] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][0] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 0
83                          myImage[(this.pos[(int)(this.pos.length/2)][0][1]
                                * 48 +
                                this.pos[(int)(this.pos.length/2)][0][0])
                                * 3 + 1] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][1] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 1
84                          myImage[(this.pos[(int)(this.pos.length/2)][0][1]
                                * 48 +
                                this.pos[(int)(this.pos.length/2)][0][0])
                                * 3 + 2] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][2] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 2
85                          harbor.setPos(this.pos[(int)(this.pos.length/2)][0][1],this.pos[(int)(this.po
86                          break;
87                      case 8:
88                          myImage[(this.pos[0][0][1] * 48 +
                                this.pos[0][0][0]) * 3 + 0] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][0] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 0
89                          myImage[(this.pos[0][0][1] * 48 +
                                this.pos[0][0][0]) * 3 + 1] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][1] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 1
90                          myImage[(this.pos[0][0][1] * 48 +
                                this.pos[0][0][0]) * 3 + 2] =
                                (short)(harbor.getColor()[harbor.getPossession()
                                + 1][2] - harbor.getHasBonus()); // (y *
                                48 + x) * 3 + 2
91                          harbor.setPos(this.pos[0][0][1],this.pos[0][0][0]);
92                          break;
93                  }
94              }
95          }
96      }
97      return myImage;
```

```
98      }
99
100     public boolean hasHarbor(){
101         return (this.harbor != null) ? true : false;
102     }
103
104     public Harbor getHarbor(){
105         return this.harbor;
106     }
107
108     @Override
109     int collide(short[] myImage) {
110         return -1;
111     }
112
113     @Override
114     void move(int dir,short[] myImage) {
115     }
116
117     @Override
118     short[] run(int key, short[] myImage) {
119         return myImage;
120     }
121
122 }
```

**Player.java**

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

```
43                  ret = 1;
44              }
45          }
46          return ret;
47      }
48
49      public int[][] getPos(){
50          return this.pos;
51      }
52
53      public int getHitX(){
54          return this.hitX;
55      }
56      public int getHitY(){
57          return this.hitY;
58      }
59      public boolean getHit(){
60          return this.hit;
61      }
62
63      public void resetHit(){
64          this.hit = false;
65          this.hitX = -1;
66          this.hitY = -1;
67      }
68
69      protected int hitIsland(short[] myImage, int x, int y, boolean
            harbor){
70          if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
71              int idx = (y * 48 + x) * 3;
72              if(harbor){
73                  return ((myImage[idx + 0] == 125 && myImage[idx + 1] ==
                        66 && myImage[idx + 2] == 24) ||
74                      (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
                            myImage[idx + 2] == 35) ||
75                      (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
                            myImage[idx + 2] == 221)) ? (((myImage[idx + 0]
                            == 121 && myImage[idx + 1] == 235 && myImage[idx
                            + 2] == 34)) ? 2 : 1) : 0;
76              }else {
77              return ((myImage[idx + 0] == 196 && myImage[idx + 1] == 156
                    && myImage[idx + 2] == 53) ||
78                      (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
                            myImage[idx + 2] == 48) ||
79                      (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
                            myImage[idx + 2] == 24) ||
80                      (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
                            myImage[idx + 2] == 35) ||
81                      (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
                            myImage[idx + 2] == 221)) ? (((myImage[idx + 0]
```

```
                    == 121 && myImage[idx + 1] == 235 && myImage[idx
                    + 2] == 34)) ? 2 : 1) : 0;
82          }
83      }else {
84          return 0;
85      }
86  }
87
88  public short[] run(int key, short[] myImage){
89      myImage = isHit(myImage);
90      if(collectBonus(myImage, 1) && !this.hadBonus){
91          this.hp += (this.hp < this.MAXHP) ? 1 : 0;
92          this.hadBonus = true;
93      }
94      if(key != -1){
95          myImage = clearTrace(myImage);
96          move(key,myImage);
97          if (collide(myImage) == 1){
98              resetMove();
99              if(key == 2){
100                 this.align++;
101             }
102             if(key == 3){
103                 this.align--;
104             }
105         }
106     }
107     if(this.bullet != null){
108         if(this.bullet.getHasHit()){
109             addScore(50);
110         }
111         if(this.bullet.getRange() > 0){
112             myImage = this.bullet.run(-1, myImage);
113         }else{
114             this.bullet = null;
115         }
116     }
117     if(this.bullet2 != null){
118         if(this.bullet2.getHasHit()){
119             addScore(50);
120         }
121         if(this.bullet2.getRange() > 0){
122             myImage = this.bullet2.run(-1, myImage);
123         }else{
124             this.bullet2 = null;
125         }
126     }
127     myImage = paint(myImage);
128     return myImage;
129 }
```

```
130
131     public Score getScore(){
132         return this.score;
133     }
134
135     public void addScore(int val){
136         this.score.add(val);
137     }
138
139     /**
140      * Debug method to print shiplocation and locationdifference between
             the new and old location.
141      * */
142     public void print(){
143         System.out.println("Your ship:\nA: " + this.align);
144         for (int i = 0; i < this.pos.length; i++){
145             System.out.println("X: " + this.pos[i][0] + " Y: " +
                    this.pos[i][1]);
146             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
                    this.oldpos[i][1]);
147             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
                    - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
                    this.oldpos[i][1]));
148         }
149     }
150     private static boolean isWater(short[] myImage, int idx){
151         return (myImage[idx] == 0 && myImage[idx + 1] == 177 &&
                 myImage[idx + 2] == 241);
152     }
153     public static Player spawn(short[] myImage, int num){
154         Player ret = null;
155         while (num > 0){
156             int i = (int)(Math.random() * 46) + 1;
157             int j = (int)(Math.random() * 22) + 1;
158             if(     isWater(myImage, (((j-1) * 48 + i ) * 3)) &&
159                     isWater(myImage, (((j-1) * 48 + (i+1)) * 3)) &&
160                     isWater(myImage, (((j-1) * 48 + (i-1)) * 3)) &&
161                     isWater(myImage, ((j   * 48 + i   ) * 3)) &&
162                     isWater(myImage, ((j   * 48 + (i+1)) * 3)) &&
163                     isWater(myImage, ((j   * 48 + (i-1)) * 3)) &&
164                     isWater(myImage, (((j+1) * 48 + i ) * 3)) &&
165                     isWater(myImage, (((j+1) * 48 + (i+1)) * 3)) &&
166                     isWater(myImage, (((j+1) * 48 + (i-1)) * 3))){
167                 ret = new Player(3,i,j,(int)(Math.random()*7)+1);
168                 num--;
169                 continue;
170             }
171         }
172         return ret;
173     }
```

```
174 }
```

**GameMain.java**

```
1  package de.thdeg.game.assets;
2
3  import de.thdeg.game.runtime.InternalLedGameThread;
4  import de.thdeg.game.assets.Fleet;
5  import java.util.ArrayList;
6  import java.util.Collections;
7  import java.util.List;
8  import java.util.Scanner;
9  public class GameMain {
10
11     static public void main(String[] passedArgs) throws
           InterruptedException {
12         short[] myImage = new short[24*48*3];
13         List<Integer> highscore = new ArrayList<Integer>();
14         int thisKey=0;
15         int frame = 0;
16         int round = 1;
17         int diff = 1;
18         int enemyHealth = 2;
19         long startTime = System.currentTimeMillis();
20         long roundtime = 30000;
21
22         // This is initialization, donot change this
23         InternalLedGameThread.run();
24
25         System.out.println("Willkommen bei Mari proelium!\n In kuerze
               wird das Spiel beginnen und Ihr Punktestand wird mit den
               anderen Spielern verglichen!\n");
26         Thread.sleep(1000);
27
28         boolean end = false;
29         Scanner scan = new Scanner(System.in);
30         do {
31             Fleet fleet = new Fleet();
32             World world = new World();
33             myImage = world.parseImage("intro.mvh");
34             InternalLedGameThread.showImage(myImage);
35             Thread.sleep(1500);
36             myImage = world.fade(myImage);
37             InternalLedGameThread.showImage(myImage);
38             Thread.sleep(500);
39             myImage = world.fade(myImage);
40             InternalLedGameThread.showImage(myImage);
41             Thread.sleep(500);
42             myImage = world.fade(myImage);
43             InternalLedGameThread.showImage(myImage);
44             Thread.sleep(500);
45             myImage = world.parseImage("round1.mvh");
```

```
46              InternalLedGameThread.showImage(myImage);
47              Thread.sleep(500);
48              myImage = world.clear();
49              myImage = world.createIsland(myImage, 5);
50              myImage = fleet.employFleet(myImage, diff, enemyHealth);
51              Player p = Player.spawn(myImage,1);
52              myImage = p.paint(myImage);
53              myImage = fleet.paintFleet(myImage);
54              InternalLedGameThread.showImage(myImage);
55              System.out.println("Drucken Sie eine beliebige Taste um das
                    Spiel zu starten.");
56              while(true){
57                  if(InternalLedGameThread.getKeyboard() != -1){
58                      break;
59                  }
60              }
61              startTime = System.currentTimeMillis();
62              while(p.isAlive()){
63                  thisKey = InternalLedGameThread.getKeyboard();
64                  myImage = p.run(thisKey, myImage);
65                  if(p.getHit()){
66                      fleet.distributeDamage(p.getHitX(), p.getHitY());
67                  }
68                  myImage = fleet.statusUpdate(myImage);
69                  if(frame % 4 == 0) {
70                      frame = 0;
71                      myImage = fleet.executeOrders(myImage);
72                  }
73                  if(frame % 2 == 0 || p.getHadBonus() ||
                        fleet.getHadBonus()){
74                      myImage = world.runHarbor(myImage);
75                  }
76                  myImage = world.paintIslands(myImage);
77                  InternalLedGameThread.showImage(myImage);
78                  frame++;
79                  Thread.sleep(100);
80                  System.out.println("+++ " + (System.currentTimeMillis() -
                        startTime) + " +++");
81                  p.damage(fleet.damageControl());
82                  fleet.resetDamageControl();
83                  if((System.currentTimeMillis() - startTime) > roundtime
                        || fleet.getNumberOfAliveShips() == 0){
84                      if(fleet.getNumberOfAliveShips() == 0){
85                          p.addScore(50);
86                      }
87                      round++;
88                      if(round % 3 == 0){
89                          diff++;
90                          p.resetHadBonus();
91                          fleet.resetHadBonus();
```

```
92                      myImage = world.createIsland(myImage, 5);
93                  }
94                  if(round == 16){
95                      round = 1;
96                      enemyHealth++;
97                      diff = 1;
98                  }
99                  myImage = fleet.employFleet(myImage, (diff -
                        fleet.getNumberOfAliveShips()), enemyHealth);
100                 myImage = world.resetHarbor(myImage);
101                 startTime = System.currentTimeMillis();
102                 p.addScore(200);
103                 String s = "round" + round + ".mvh";
104                 System.out.println(s);
105                 myImage = world.parseImage(s);
106                 InternalLedGameThread.showImage(myImage);
107                 Thread.sleep(1000);
108                 myImage = world.clear();
109             }
110         }
111         world.clear();
112         myImage = world.parseImage("gameover.mvh");
113         InternalLedGameThread.showImage(myImage);
114         p.addScore(fleet.getDead() * 50);
115         highscore.add(p.getScore().getScore());
116         Collections.sort(highscore);
117         Collections.reverse(highscore);
118         System.out.println("(" + round + ") - Score: " +
                p.getScore().getScore());
119         System.out.println("Highscores:");
120         for(int i = 0; i < highscore.size(); i++){
121             System.out.println("(" + (i+1) + ") -> " +
                    highscore.get(i));
122         }
123         myImage = world.parseImage("commands.mvh");
124         InternalLedGameThread.showImage(myImage);
125         Thread.sleep(1500);
126         thisKey = -1;
127         while(true){
128             thisKey = InternalLedGameThread.getKeyboard();
129             if(thisKey != -1){
130                 break;
131             }
132         }
133         switch(thisKey){
134             case 0 -> {
135                 end = false;
136                 thisKey=0;
137                 frame = 0;
138                 round = 1;
```

```
139                     diff = 1;
140                     enemyHealth = 2;
141                 }
142                 case 1 -> {
143                     end = true;
144                     System.exit(0);
145                 }
146                 case 2 -> {
147                     end = true;
148                     System.exit(0);
149                 }
150                 case 3 -> {
151                     end = true;
152                     System.exit(0);
153                 }
154             }
155         }while(!end);
156     }
157 }
```

**World.java**

```
1  package de.thdeg.game.assets;
2  import java.util.ArrayList;
3  import java.util.List;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.util.Scanner;
7  public class World {
8      private List<Island> islands;
9
10     World(){
11         this.islands = new ArrayList<Island>();
12     }
13
14     public void addIslands(Island s){
15         this.islands.add(s);
16     }
17
18     public short[] paintIslands(short[] myImage){
19         for(Island i : this.islands){
20             myImage = i.paint(myImage);
21         }
22         return myImage;
23     }
24
25     private boolean isWater(short[] myImage, int idx){
26         return (myImage[idx] == 0 && myImage[idx + 1] == 177 &&
                myImage[idx + 2] == 241);
27     }
28
29     private boolean canPlaceIsland(short[] myImage, int rad, int x, int
           y){
30         boolean ret = true;
31         for(int i = -rad; i <= rad; i++){
32             for(int j = -rad; j <= rad; j++){
33                 if((y+i) >= 0 && (y+i) <= 23 && (x+j) <= 47 && (x+j) >= 0
                       && isWater(myImage, (((y+i) * 48 + (x+j)) * 3)) &&
                       ret){
34                     ret = true;
35                 }else {
36                     return false;
37                 }
38             }
39         }
40         return ret;
41     }
42
43     public short[] createIsland(short[] myImage, int num){
44         this.islands.clear();
```

```
45          while (num > 0){
46              int i = (int)(Math.random() * 45) + 1;
47              int j = (int)(Math.random() * 21) + 1;
48              if(canPlaceIsland(myImage,5,i,j)){
49                  int[] size = {3,3};
50                  if(Math.random() < 0.5){
51                      addIslands(new Island(size,i,j));
52                  } else {
53                      addIslands(new Island(size,i,j,new
                            Harbor((int)(Math.random()*7)+1)));
54                  }
55                  myImage = paintIslands(myImage);
56                  num--;
57              }
58          }
59          return myImage;
60      }
61
62      public short[] resetHarbor(short[] myImage){
63          for(Island i : this.islands){
64              if(i.hasHarbor()){
65                  myImage = i.getHarbor().reset(myImage);
66              }
67          }
68          return myImage;
69      }
70
71      public short[] runHarbor(short[] myImage){
72          for(Island i : this.islands){
73              if(i.hasHarbor()){
74                  myImage = i.getHarbor().run(-1,myImage);
75              }
76          }
77          return myImage;
78      }
79
80      public short[] clear(){
81          short[] ret = new short[24*48*3];
82          for(int i=0; i<ret.length; i+=3){
83              ret[i+0]=(short)0;
84              ret[i+1]=(short)177;
85              ret[i+2]=(short)241;
86          }
87          return ret;
88      }
89
90      public short[] fade(short[] myImage){
91          for (int i = 0; i < myImage.length; i += 3){
92              double gr = myImage[i + 1] / (myImage[i]+1);
93              double br = myImage[i + 2] / (myImage[i]+1);
```

```
94              myImage[i] = (short) (myImage[i] * 0.7);
95              myImage[i + 1] = (short) (myImage[i] * gr);
96              myImage[i + 2] = (short) (myImage[i] * br);
97          }
98          return myImage;
99      }
100
101     public short[] parseImage(String what){
102         short[] ret = new short[24*48*3];
103         Scanner myReader = null;
104         try {
105     myReader = new Scanner(new
            File("Game"+File.separator+"src"+File.separator+"de"+File.separator+"thdeg"+File.separator+"as
106             while (myReader.hasNextLine()) {
107                 String data = myReader.nextLine();
108                 if(data.length() > 1 && !data.substring(0,1).equals("/")){
109                 if(data.substring(0,1).equals("B")){
110                     for(int i = 0; i < ret.length; i += 3){
111                         String[] s = data.substring(1).split("-");
112                         ret[i] = Short.parseShort(s[0]);
113                         ret[i + 1] = Short.parseShort(s[1]);
114                         ret[i + 2] = Short.parseShort(s[2]);
115                     }
116                 }else{
117                     String[] pos = data.substring(0,
                            data.indexOf(':')).split("-");
118                     String[] rgb =
                            data.substring(data.indexOf(':')+1).split("-");
119                     ret[(Integer.parseInt(pos[1]) * 48 +
                            Integer.parseInt(pos[0])) * 3 + 0] =
                            Short.parseShort(rgb[0]);
120                     ret[(Integer.parseInt(pos[1]) * 48 +
                            Integer.parseInt(pos[0])) * 3 + 1] =
                            Short.parseShort(rgb[1]);
121                     ret[(Integer.parseInt(pos[1]) * 48 +
                            Integer.parseInt(pos[0])) * 3 + 2] =
                            Short.parseShort(rgb[2]);
122                 }
123                 }
124             }
125         } catch (FileNotFoundException e) {
126             // e.printStackTrace();
127             ret = clear();
128             return ret;
129         }
130         myReader.close();
131         return ret;
132     }
133 }
```

**Score.java**

```
1  package de.thdeg.game.assets;
2  public class Score {
3      int score;
4
5      Score(){}
6
7      public void add(int val){
8          this.score += val;
9      }
10
11     public void setScore(int score){
12         this.score = score;
13     }
14
15     public int getScore(){
16         return this.score;
17     }
18 }
```