

Leistungsnachweis im Fach Programmierung 1

Maximilian von Hohenbühel
Fabian Cieslik

2020-12-07 - 2020-01-17

Contents

| | | |
|---|--------------------------------|------|
| 1 | Projektvoraussetzungen | III |
| 2 | Idee | IV |
| 3 | Beschreibung | V |
| 4 | Programmablauf | VII |
| 5 | Arbeitsverteilung | VIII |
| 6 | Klassendiagramm | X |
| 7 | Kommentare und Zusammenfassung | XI |
| 8 | Programmcode | XII |

1 Projektvoraussetzungen

Beschreibung: Die Projektaufgabe besteht darin, ein einfaches Spiel zu implementieren. Die Wahl des Spiels bleibt Ihnen überlassen, beachten Sie jedoch, dass sich im Rahmen von 36 Stunden Arbeitszeit nur sehr begrenzte Spielideen auch umsetzen lassen.

Details: Sie programmieren ein Spiel für ein sehr eingeschränktes Display. Dieses enthält nur 24×48 Bildpunkte (Pixel), d.h. 24 Reihen mit jeweils 48 Spalten. Jeder Bildpunkt kann 16 Millionen Farben annehmen, wobei die Rot, Grün und Blau-Komponente mit jeweils einem Byte angesprochen wird. Als Steuermöglichkeit stehen Ihnen vier Tasten zur Verfügung, die wie im Cursorblock üblich angeordnet sind. Es gibt nur einen Spieler. Die Zeit für eine Spielrunde sollte bei 20-30 Sekunden liegen.

Zur Ein- und Ausgabe erhalten Sie eine Klasse mit zwei Methoden:

- *public int getKeyboard()*
Liefert die vier Cursortasten der Tastatur folgende Werte zurück:
 - 0 -> "hoch"
 - 1 -> "runter"
 - 2 -> "links"
 - 3 -> "rechts"
 - -1 -> keine Taste
- *public void showImage(short[] image)*
Zeigt ein komplettes Bild auf dem Display an, wobei der erste Wert des Arrays die Rot-Komponente des linken oben Bildpunkts ist und der letzte Wert die Blau-Komponente von 0 bis 255 des rechten unteren Bildpunktes. Das übergebene Array muss exakt $24 * 48 * 3$ Elemente haben für die 24 Zeilen, 48 Spalten und 3 Farbkomponenten pro Pixel. Das Display wird zeilenweise durchlaufen.

Spielumfang:

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

2 Idee

Name:

MP - Mari proelium

Spiel:

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 Kollision-sinteraktionen 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 gestartet 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 oder das Program zu Beenden.

- **Spielablauf**

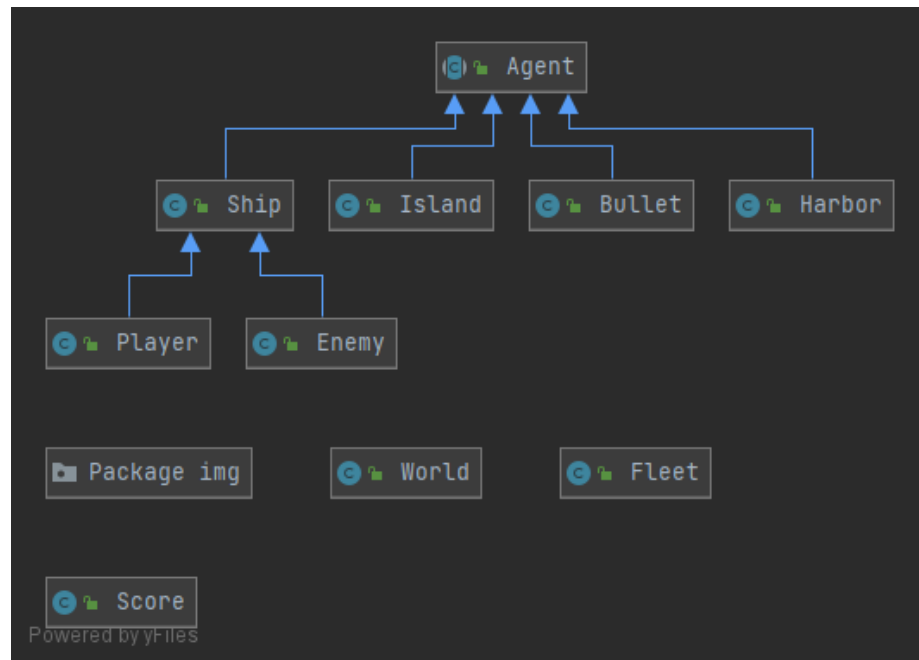
Zuerst wird der Spieler bewegt und auf Kollisionen überprüft, danach die Gegner. Anschließend wird überprüft ob man eine Runde überlebt hat und falls ja werden die Punkte zugewiesen und neue Gegner erstellt. Alle drei Runden wird das Spiel schwerer und die Positionen der Inseln neu gewählt.

5 Arbeitsverteilung

| Projektphase | Idee/Entwurf | Programmierung | Dokumentation |
|---|---|---|---|
| Fabian Cieslik (Keine Vorkenntnisse im Programmieren) | 1 Std. | 20 Std. | 8 Std. |
| Aufgaben | <ul style="list-style-type: none"> - Erstellung des Konzepts - Erweiterung durch Ideen - Herangehensweise an die Programmierstruktur | <ul style="list-style-type: none"> - Inseln - Häfen und Funktionen - Karte | <ul style="list-style-type: none"> - Beschreibungsgrafik - Idee - Beschreibung |
| Maximilian von Hohenbühel (Vorkenntnisse im Programmieren) | 1 Std. | 30 Std. | 8 Std. |
| Aufgaben | <ul style="list-style-type: none"> - Erstellung des Konzepts - Erweiterung durch Ideen - Herangehensweise an die Programmierstruktur | <ul style="list-style-type: none"> - Schiffe und Funktionen - Grafiken - Spielablauf - Kollisionsberechnung | <ul style="list-style-type: none"> - Idee- Beschreibung- Klassendiagramm |

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 Kommentare und Zusammenfassung

Allgemein stellt das Programmieren eines Spiels ein ideales Ende des Moduls Programmieren 1 dar. Die erlernten Programmierkenntnisse werden dabei optimal abgefragt und angewendet, dadurch werden auf vielseitige Art und Weise die Kenntnisse vertieft und erweitert.

Ein wesentlicher Kritikpunkt der individuellen Gruppeneinteilung hat sich bei uns gezeigt. Ein Mitglied (Maximilian) ist ein bereits erfahrener bzw. sehr fortgeschrittener Programmierer in diversen Sprachen. Das andere Mitglied (Fabian) dagegen besitzt kaum Vorkenntnisse in diesem Bereich. Die Schwierigkeit lag also darin, ein Spiel zu finden, was beiden Ansprüchen genügt, d.h. denjenigen ohne Kenntnisse nicht überfordert, als auch denjenigen mit fortgeschrittenen Kenntnissen unterfordert. Dieses Ziel zu erreichen gelang leider gar nicht. Die Überforderung des Teammitglieds ohne Kenntnisse war deutlich zu erkennen. Dies führen wir beide auf die teils einseitigen und zu praxisfernen Übungen während des Semesters und das nicht optimale Einleiten der Herangehensweise eines Projekts zurück. Besser/effektiver wäre gewesen, mehrere kleinere Übungen zu jedem Teilgebiet und eine Vorlesung/ein Skript anzubieten, das einen roten Faden von Anfang bis Ende besitzt. Die Struktur der Vorlesung war ebenfalls nicht unbedingt von Vorteil. Diese sollte mehr Schritt für Schritt aufgebaut sein (z.B. sollten nicht Klassen vor Zugriffsmodifikatoren gelehrt werden). Dies hat teilweise für Verwirrung gesorgt und musste ausführlicher vom Teammitglied mit Kenntnissen erklärt werden. Insgesamt kann man sagen, dass wir ein einfacheres Spiel hätten wählen sollen, um die Herangehensweise an ein solches Projekt Schritt für Schritt erklären, genügend Zeit in die Erklärung des Stoffes investieren und die Wissens-/Kenntnislevel zu einem gewissen Teil aneinander anpassen zu können.

Das Spiel betreffend sind wir durchaus zufrieden. Das Ergebnis hat uns letzten Endes sehr überrascht, da wir nicht gedacht hatten, dass ein so einfaches Spiel so Spaß machen kann. Insgesamt ist diese Idee, wie bereits beschrieben, eine tolle Endaufgabe im Modul Programmieren 1, benötigt aber hier und da etwas Feinschliff. Vor allem was die Gruppeneinteilung angeht, sollte man sich etwas mehr Gedanken machen, da der Unterschied der jeweiligen Level teilweise deutlich auseinander liegt.

8 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]
108                 = (short)0;
109             myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 1]
110                 = (short)177;
111             myImage[(this.oldpos[i][1] * 48 + this.oldpos[i][0]) * 3 + 2]
112                 = (short)241;
113         }
114         return myImage;
115     }
116
117     protected int getShipColor(){
118         double ps = (double)this.hp / (double)this.MAXHP;
119         if(ps > 0.67){
120             return 2;
121         }else if(ps > 0.34){
122             return 1;
123         }else {
124             return 0;
125         }
126     }
127
128     /**
129     * This method uses the Players values to update the map and return
130     * it always.
131     * @param myImage the Pixel array given from the {@link GameMain}
132     * @return the updated maparray
133     */
134     public short[] paint(short[] myImage){
135         myImage = clearTrace(myImage);
136         if(this.hp > 0) {
137             for(int i=0; i < this.pos.length; i++){
138                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
139                     color[getShipColor()][i][0];
140                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
141                     color[getShipColor()][i][1];
142                 myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
143                     color[getShipColor()][i][2];
144             }
145         }else {
146             for(int i=0; i < this.pos.length; i++){

```

```

140         myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 0] =
141             0;
142         myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 1] =
143             177;
144         myImage[(this.pos[i][1] * 48 + this.pos[i][0]) * 3 + 2] =
145             241;
146     }
147     if(this.bullet != null){
148         myImage = this.bullet.clear(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
166     r2, short g2, short b2){
167     return (r1 == r2 && g1 == g2 && b1 == b2);
168 }
169
170 /**
171  * The method collide looks at the pixels of the ship and look if it
172  * collided with another object
173  */
174 public int collide(short[] myImage){
175     return -1;
176 }
177
178 /**
179  * This method takes the userinput and changes the
180  * position/direction of the ship
181  * @param dir represents the given userinput
182  * 0 - up
183  * 1 - down
184  * 2 - left
185  * 3 - right
186  */

```

```

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
203     * ship in the given direction.
204     * @param dir represents the direction which the ship takes to
205     * rotate.
206     * 0 - Left
207     * 1 - Right
208     * @param save decides if the old ship position shall be saved
209     */
210     protected void rotate(int dir, boolean save, short[] myImage){
211         if(dir == 0){ // Left
212             switch(this.align){
213                 case 1:
214                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
215                         47 &&
216                         !(hitIsland(myImage, this.pos[0][0] - 1,
217                             this.pos[0][1], false) > 0) &&
218                         !(hitIsland(myImage, this.pos[2][0] + 1,
219                             this.pos[2][1], false) > 0)) {
220                         if(save){saveOldPos();}
221                         this.pos[0][0]--;
222                         this.pos[2][0]++;
223                         changeAlign(-1);
224                     }
225                     break;
226                 case 2:
227                     if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
228                         47 &&
229                         !(hitIsland(myImage, this.pos[0][0] - 1,
230                             this.pos[0][1], false) > 0) &&
231                         !(hitIsland(myImage, this.pos[2][0] + 1,
232                             this.pos[2][1], false) > 0)) {
233                         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 <=
233         23 &&
234         !(hitIsland(myImage, this.pos[0][0],
235             this.pos[0][1] - 1, false) > 0) &&
236         !(hitIsland(myImage, this.pos[2][0],
237             this.pos[2][1] + 1, false) > 0)) {
238         if(save){saveOldPos();}
239         this.pos[0][1]--;
240         this.pos[2][1]++;
241         changeAlign(-1);
242     }
243     break;
244 case 4:
245     if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
246         23 &&
247         !(hitIsland(myImage, this.pos[0][0],
248             this.pos[0][1] - 1, false) > 0) &&
249         !(hitIsland(myImage, this.pos[2][0],
250             this.pos[2][1] + 1, false) > 0)) {
251         if(save){saveOldPos();}
252         this.pos[0][1]--;
253         this.pos[2][1]++;
254         changeAlign(-1);
255     }
256     break;
257 case 5:
258     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
259         0 &&
260         !(hitIsland(myImage, this.pos[0][0] + 1,
261             this.pos[0][1], false) > 0) &&
262         !(hitIsland(myImage, this.pos[2][0] - 1,
263             this.pos[2][1], false) > 0)) {
264         if(save){saveOldPos();}
265         this.pos[0][0]++;
266         this.pos[2][0]--;
267         changeAlign(-1);
268     }
269     break;
270 case 6:
271     if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
272         0 &&
273         !(hitIsland(myImage, this.pos[0][0] + 1,
274             this.pos[0][1], false) > 0) &&
275         !(hitIsland(myImage, this.pos[2][0] - 1,

```

```

265         this.pos[2][1], false) > 0)) {
266         if(save){saveOldPos();}
267         this.pos[0][0]++;
268         this.pos[2][0]--;
269         changeAlign(-1);
270     }
271     break;
272 case 7:
273     if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
274         0 &&
275         !(hitIsland(myImage, this.pos[0][0],
276             this.pos[0][1] + 1, false) > 0) &&
277         !(hitIsland(myImage, this.pos[2][0],
278             this.pos[2][1] - 1, false) > 0)) {
279         if(save){saveOldPos();}
280         this.pos[0][1]++;
281         this.pos[2][1]--;
282         changeAlign(-1);
283     }
284     break;
285 case 8:
286     if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
287         0 &&
288         !(hitIsland(myImage, this.pos[0][0],
289             this.pos[0][1] + 1, false) > 0) &&
290         !(hitIsland(myImage, this.pos[2][0],
291             this.pos[2][1] - 1, false) > 0)) {
292         if(save){saveOldPos();}
293         this.pos[0][1]++;
294         this.pos[2][1]--;
295         changeAlign(-1);
296     }
297     break;
298 }
299 }else { // Right
300     switch(this.align){
301     case 1:
302         if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
303             0 &&
304             !(hitIsland(myImage, this.pos[0][0] + 1,
305                 this.pos[0][1], false) > 0) &&
306             !(hitIsland(myImage, this.pos[2][0] - 1,
307                 this.pos[2][1], false) > 0)) {
308             if(save){saveOldPos();}
309             this.pos[0][0]++;
310             this.pos[2][0]--;
311             changeAlign(1);
312         }
313         break;
314     case 2:

```

```

305         if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
306             0 &&
307                 !(hitIsland(myImage, this.pos[0][0],
308                     this.pos[0][1] + 1, false) > 0) &&
309                 !(hitIsland(myImage, this.pos[2][0],
310                     this.pos[2][1] - 1, false) > 0)) {
311             if(save){saveOldPos();}
312             this.pos[0][1]++;
313             this.pos[2][1]--;
314             changeAlign(1);
315         }
316         break;
317     case 3:
318         if (this.pos[0][1] + 1 <= 23 && this.pos[2][1] - 1 >=
319             0 &&
320                 !(hitIsland(myImage, this.pos[0][0],
321                     this.pos[0][1] + 1, false) > 0) &&
322                 !(hitIsland(myImage, this.pos[2][0],
323                     this.pos[2][1] - 1, false) > 0)) {
324             if(save){saveOldPos();}
325             this.pos[0][1]++;
326             this.pos[2][1]--;
327             changeAlign(1);
328         }
329         break;
330     case 4:
331         if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
332             47 &&
333                 !(hitIsland(myImage, this.pos[0][0] - 1,
334                     this.pos[0][1], false) > 0) &&
335                 !(hitIsland(myImage, this.pos[2][0] + 1,
336                     this.pos[2][1], false) > 0)) {
337             if(save){saveOldPos();}
338             this.pos[0][0]--;
339             this.pos[2][0]++;
340             changeAlign(1);
341         }
342         break;
343     case 5:
344         if (this.pos[0][0] - 1 >= 0 && this.pos[2][0] + 1 <=
345             47 &&
346                 !(hitIsland(myImage, this.pos[0][0] - 1,
347                     this.pos[0][1], false) > 0) &&
348                 !(hitIsland(myImage, this.pos[2][0] + 2,
349                     this.pos[2][1], false) > 0)) {
350             if(save){saveOldPos();}
351             this.pos[0][0]--;
352             this.pos[2][0]++;
353             changeAlign(1);
354         }
355     }

```

```

343         break;
344     case 6:
345         if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
346             23 &&
347             !(hitIsland(myImage, this.pos[0][0],
348                 this.pos[0][1] - 1, false) > 0) &&
349             !(hitIsland(myImage, this.pos[2][0],
350                 this.pos[2][1] + 1, false) > 0)) {
351             if(save){saveOldPos();}
352             this.pos[0][1]--;
353             this.pos[2][1]++;
354             changeAlign(1);
355         }
356         break;
357     case 7:
358         if (this.pos[0][1] - 1 >= 0 && this.pos[2][1] + 1 <=
359             23 &&
360             !(hitIsland(myImage, this.pos[0][0],
361                 this.pos[0][1] - 1, false) > 0) &&
362             !(hitIsland(myImage, this.pos[2][0],
363                 this.pos[2][1] + 1, false) > 0)) {
364             if(save){saveOldPos();}
365             this.pos[0][1]--;
366             this.pos[2][1]++;
367             changeAlign(1);
368         }
369         break;
370     case 8:
371         if (this.pos[0][0] + 1 <= 47 && this.pos[2][0] - 1 >=
372             0 &&
373             !(hitIsland(myImage, this.pos[0][0] + 1,
374                 this.pos[0][1], false) > 0) &&
375             !(hitIsland(myImage, this.pos[2][0] - 1,
376                 this.pos[2][1], false) > 0)) {
377             if(save){saveOldPos();}
378             this.pos[0][0]++;
379             this.pos[2][0]--;
380             changeAlign(1);
381         }
382         break;
383     }
384 }
385
386 protected void rotateTo(int newOri, short[] myImage){
387     saveOldPos();
388     while (this.align != newOri) {
389         rotate(1, false, myImage);
390     }
391 }

```

```

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
466         the ship would move outside the map.
467  */
468 protected boolean canMove(short[] myImage){
469     boolean ret = false;
470     switch(this.align){
471         case 1 -> {
472             if(this.pos[0][1] - 1 >= 0 &&
473                 !(hitIsland(myImage, this.pos[0][0], this.pos[0][1]-1,
474                     false) > 0)){
475                 if(!hitEnemy(myImage, this.pos[0][0], this.pos[0][1]-1)){
476                     return true;
477                 }else {
478                     damage(1);
479                 }
480             }
481         }
482         case 2 -> {
483             if(this.pos[0][1] - 1 > 0 && this.pos[0][0] + 1 < 48 &&

```

```

481         !(hitIsland(myImage, this.pos[0][0]+1, this.pos[0][1]-1,
482         false) > 0)){
483         if(!hitEnemy(myImage, this.pos[0][0]+1, this.pos[0][1]-1)){
484             return true;
485         }else {
486             damage(1);
487         }
488     }
489 }
490 case 3 -> {
491     if(this.pos[0][0] + 1 < 48 &&
492     !(hitIsland(myImage, this.pos[0][0]+1, this.pos[0][1],
493     false) > 0)){
494         if(!hitEnemy(myImage, this.pos[0][0]+1, this.pos[0][1])){
495             return true;
496         }else {
497             damage(1);
498         }
499     }
500 }
501 case 4 -> {
502     if(this.pos[0][0] + 1 < 48 && this.pos[0][1] + 1 < 24 &&
503     !(hitIsland(myImage, this.pos[0][0]+1, this.pos[0][1]+1,
504     false) > 0)){
505         if(!hitEnemy(myImage, this.pos[0][0]+1, this.pos[0][1]+1)){
506             return true;
507         }else {
508             damage(1);
509         }
510     }
511 }
512 case 5 -> {
513     if(this.pos[0][1] + 1 < 24 &&
514     !(hitIsland(myImage, this.pos[0][0], this.pos[0][1]+1,
515     false) > 0)){
516         if(!hitEnemy(myImage, this.pos[0][0], this.pos[0][1]+1)){
517             return true;
518         }else {
519             damage(1);
520         }
521     }
522 }
523 case 6 -> {
524     if(this.pos[0][1] + 1 < 24 && this.pos[0][0] - 1 >= 0 &&
525     !(hitIsland(myImage, this.pos[0][0]-1, this.pos[0][1]+1,
526     false) > 0)){
527         if(!hitEnemy(myImage, this.pos[0][0]-1, this.pos[0][1]+1)){
528             return true;
529         }else {
530             damage(1);
531         }
532     }
533 }

```

```

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 &&
565             myImage[idx + 2] == 36) ||
566            (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
567             myImage[idx + 2] == 36) ||
568            (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
569             myImage[idx + 2] == 36) ||
570            (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
571             myImage[idx + 2] == 22) ||
572            (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
573             myImage[idx + 2] == 22) ||
574            (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
575             myImage[idx + 2] == 21) ||
576            (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
577             myImage[idx + 2] == 11) ||
578            (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
579             myImage[idx + 2] == 10) ||
580            (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
581             myImage[idx + 2] == 11);
582     }else {
583         return false;
584     }
585 }
586
587 protected int hitIsland(short[] myImage, int x, int y, boolean
588     harbor){
589     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
590         int idx = (y * 48 + x) * 3;
591         if(harbor){
592             return ((myImage[idx + 0] == 125 && myImage[idx + 1] ==
593                     66 && myImage[idx + 2] == 24) ||
594                    (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
595                     myImage[idx + 2] == 35) ||
596                    (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
597                     myImage[idx + 2] == 221)) ? 1 : 0;
598         }else {
599             return ((myImage[idx + 0] == 196 && myImage[idx + 1] == 156
600                     && myImage[idx + 2] == 53) ||
601                    (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
602                     myImage[idx + 2] == 48) ||
603                    (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
604                     myImage[idx + 2] == 24) ||
605                    (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
606                     myImage[idx + 2] == 35) ||
607                    (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
608                     myImage[idx + 2] == 221)) ? 1 : 0;
609         }
610     }else {
611         return 0;
612     }
613 }

```

```

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 &&
601                 myImage[idx + 2] == 222) ||
602                 (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
603                 myImage[idx + 2] == 143) ||
604                 (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
605                 myImage[idx + 2] == 74) ||
606                 (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
607                 myImage[idx + 2] == 215) ||
608                 (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
609                 myImage[idx + 2] == 134) ||
610                 (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
611                 myImage[idx + 2] == 72) ||
612                 (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
613                 myImage[idx + 2] == 209) ||
614                 (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
615                 myImage[idx + 2] == 128) ||
616                 (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
617                 myImage[idx + 2] == 74);
618         }else {
619             return false;
620         }
621     }
622
623     protected boolean hitBullet(short[] myImage, int x, int y){
624         if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
625             int idx = (y * 48 + x) * 3;
626             return (myImage[idx + 0] == 12 && myImage[idx + 1] == 13 &&
627                 myImage[idx + 2] == 12);
628         }else {
629             return false;
630         }
631     }
632
633     /**
634     * 0 - Enemy
635     * 1 - Player
636     */
637     private boolean isCollectable(short[] myImage, int idx, int who){
638         if(who == 0){
639             return (myImage[idx] == 29 && myImage[idx + 1] == 67 &&
640                 myImage[idx + 2] == 220);
641         }else {
642             return (myImage[idx] == 121 && myImage[idx + 1] == 235 &&
643                 myImage[idx + 2] == 34);
644         }
645     }

```

```

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
651     * rotation
652     * @param dir the direction the ship rotates to
653     */
654     protected void changeAlign(int dir){
655         this.align += dir;
656         if(this.align < 1){
657             this.align = 8;
658         }
659         if(this.align > 8){
660             this.align = 1;
661         }
662     }
663
664     public short[] run(int key, short[] myImage){
665         myImage = isHit(myImage);
666         if(key != -1){
667             myImage = clearTrace(myImage);
668             System.out.println("k " + key + " |a " + this.align);
669             print("beforeif");
670             move(key,myImage);
671             if (collide(myImage) == 1){
672                 resetMove();
673                 if(key == 2){
674                     this.align++;
675                 }
676                 if(key == 3){
677                     this.align--;
678                 }
679             }
680         }
681     }

```

```

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
710  * the new and old location.
711  */
712 public void print(String where){
713     System.out.println(where + "\nA: " + this.align);
714     for (int i = 0; i < this.pos.length; i++){
715         System.out.println("X: " + this.pos[i][0] + " Y: " +
716             this.pos[i][1] + " | Xo: " + this.oldpos[i][0] + " Yo: "
717             + this.oldpos[i][1]);
718         // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
719             - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
720             this.oldpos[i][1]));
721     }
722 }
723
724 public int[] [] getPos(){
725     return this.pos;
726 }

```

```
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 &&
736                                             rgbs[i][j][k] >= 0)? rgbs[i][j][k] : 0;
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,
18                                72}},{31, 69, 222},{19, 43, 143},{10, 22, 74}},{153, 23,
19                                209},{94, 15, 128},{55, 10, 74}}};
20        changeColor(rgbs);
21        int[][] pos = { {22, 22}, {23, 22}, {24, 22} };
22        this.pos = pos;
23    }
24
25    Enemy(int hp, int x, int y, int o, int r){
26        super(hp, x, y, o);
27        short[][][] rgbs = {{{31, 222, 215},{21, 138, 134},{11, 74,
28                                72}},{153, 23, 209},{94, 15, 128},{55, 10, 74}},{31, 69,
29                                222},{19, 43, 143},{10, 22, 74}}};
30        changeColor(rgbs);
31        this.range = r;
32    }
33
34    public void resetDmg(){
35        this.dmg = 0;
36    }
37
38    public int getDamageReceived(){
39        return this.dmg;
40    }
41
42    protected int hitIsland(short[] myImage, int x, int y, boolean
        harbor){
43        if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
44            int idx = (y * 48 + x) * 3;
45            if(harbor){
46                return ((myImage[idx + 0] == 125 && myImage[idx + 1] ==
47                        66 && myImage[idx + 2] == 24) ||
```

```

43         (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
44             myImage[idx + 2] == 35) ||
45         (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
46             myImage[idx + 2] == 221)) ? ((myImage[idx + 0]
47             == 29 && myImage[idx + 1] == 67 && myImage[idx +
48             2] == 220)) ? 2 : 1) : 0;
49     }else {
50         return ((myImage[idx + 0] == 196 && myImage[idx + 1] == 156
51             && myImage[idx + 2] == 53) ||
52             (myImage[idx + 0] == 186 && myImage[idx + 1] == 148 &&
53             myImage[idx + 2] == 48) ||
54             (myImage[idx + 0] == 125 && myImage[idx + 1] == 66 &&
55             myImage[idx + 2] == 24) ||
56             (myImage[idx + 0] == 122 && myImage[idx + 1] == 236 &&
57             myImage[idx + 2] == 35) ||
58             (myImage[idx + 0] == 30 && myImage[idx + 1] == 68 &&
59             myImage[idx + 2] == 221)) ? ((myImage[idx + 0]
60             == 29 && myImage[idx + 1] == 67 && myImage[idx +
61             2] == 220)) ? 2 : 1) : 0;
62     }
63 }else {
64     return 0;
65 }
66 }
67
68 /**
69  * Create Routes
70  * */
71 private void pathFinder(){
72     this.routing = new ArrayList<int[]>();
73     /**
74      * Start: this.pos[1][0]=x
75      *       this.pos[1][1]=y
76      * End:   this.RouteX
77      *       this.RouteY
78      * */
79     this.PX = this.RouteX;
80     this.PY = this.RouteY;
81     int pX = this.pos[1][0];
82     int pY = this.pos[1][1];
83     while(pX != this.RouteX && pY != this.RouteY){
84         switch(routeDirection(pX, pY, this.RouteX, this.RouteY)){
85             case 1 -> {
86                 int[] rt = {pX, --pY};
87                 this.routing.add(rt);
88             }
89             case 2 -> {
90                 int[] rt = {++pX, --pY};
91                 this.routing.add(rt);
92             }
93         }
94     }
95 }

```



```

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],
205                                 this.PX, this.PY);
206     if(this.bullet == null){
207         this.bullet = new Bullet(orient, 8, this.pos[1][0],
208                                 this.pos[1][1], 1);
209     }
210 }
211
212 private void move(short[] myImage){
213     // if(this.routing.size() <= 0 ){
214     if(canMove(myImage)){
215         forward(myImage);
216     }else{
217         if(Math.random() > 0.5){
218             rotate(0, true, myImage);
219         }else {
220             rotate(1, true, myImage);
221         }
222     }
223     // }else {
224     //     switch(routeDirection(this.pos[1][0], this.pos[1][1],
225     //                             this.routing.get(this.routing.size() - 1)[0],
226     //                             this.routing.get(this.routing.size() - 1)[1])){
227     //         case 1 -> {
228     //             rotateTo(1);
229     //             forward(myImage);
230     //         }
231     //         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
324     * collided with another object
325     */
326     public int collide(short[] myImage){
327         int ret = 0;
328         for(int i=0; i < this.pos.length; i++){
329             int idx = (this.pos[i][1] * 48 + this.pos[i][0]) * 3;
330             if (hitIsland(myImage, this.pos[i][0], this.pos[i][1], false)
331                 > 0){
332                 ret = 1;
333             }
334             if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
335                 damage(1);
336                 ret = 2;
337             }
338             if(hitPlayer(myImage, this.pos[i][0], this.pos[i][1])) {
339                 damage(1);
340                 this.dmg += 1;
341                 ret = 1;
342             }
343         }
344         return ret;
345     }
346
347     public boolean includesPos(int x, int y){
348         for (int i = 0; i < this.pos.length; i++){
349             if(this.pos[i][0] == x && this.pos[i][1] == y){
350                 return true;
351             }
352         }
353         return false;
354     }
355
356     /**
357     * Debug method to print shiplocation and locationdifference between
358     * the new and old location.
359     */
360     public void print(){
361         System.out.println("Enemy ship:\nA: " + this.align);
362         for (int i = 0; i < this.pos.length; i++){
363             System.out.println("X: " + this.pos[i][0] + " Y: " +
364                 this.pos[i][1]);
365             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
366                 this.oldpos[i][1]);
367             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
368                 - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
369                 this.oldpos[i][1]));
370         }
371     }
372 }

```

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 &&
20            myImage[idx + 2] == 241);
21    }
22
23    public short[] employFleet(short[] myImage, int num, int hp){
24        while (num > 0){
25            int i = (int)(Math.random() * 46) + 1;
26            int j = (int)(Math.random() * 22) + 1;
27            if(
28                isWater(myImage, (((j-1) * 48 + i ) * 3)) &&
29                isWater(myImage, (((j-1) * 48 + (i+1)) * 3)) &&
30                isWater(myImage, (((j-1) * 48 + (i-1)) * 3)) &&
31                isWater(myImage, ((j * 48 + i ) * 3)) &&
32                isWater(myImage, ((j * 48 + (i+1)) * 3)) &&
33                isWater(myImage, ((j * 48 + (i-1)) * 3)) &&
34                isWater(myImage, (((j+1) * 48 + i ) * 3)) &&
35                isWater(myImage, (((j+1) * 48 + (i+1)) * 3)) &&
36                isWater(myImage, (((j+1) * 48 + (i-1)) * 3))) {
37                addFleetmember(new Enemy(hp, i, j, (int)(Math.random() *
38                    7 + 1), 15));
39                myImage = paintFleet(myImage);
40                num--;
41                continue;
42            }
43        }
44        return myImage;
45    }
46
47    public void resetHadBonus(){
48        for(Enemy e : this.fleet){
49            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] =
28            (short)0;
29        myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 1] =
30            (short)177;
31        myImage[(this.oldpos[1] * 48 + this.oldpos[0]) * 3 + 2] =
32            (short)241;
33        return myImage;
34    }
35
36    public short[] clear(short[] myImage){
37        myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 0] = (short)0;
38        myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 1] = (short)177;
39        myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 2] = (short)241;
40        return myImage;
41    }
42
43    public short[] paint(short[] myImage){
44        myImage = clearTrace(myImage);
45        if(this.range > 0) {
46            myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 0] = (short)12;
47            myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 1] = (short)13;
48            myImage[(this.pos[1] * 48 + this.pos[0]) * 3 + 2] = (short)12;
49        }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 &&
59             myImage[idx + 2] == 36) ||
60             (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
61                 myImage[idx + 2] == 36) ||
62             (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
63                 myImage[idx + 2] == 36) ||
64             (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
65                 myImage[idx + 2] == 22) ||
66             (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
67                 myImage[idx + 2] == 22) ||
68             (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
69                 myImage[idx + 2] == 21) ||
70             (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
71                 myImage[idx + 2] == 11) ||
72             (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
73                 myImage[idx + 2] == 10) ||
74             (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
75                 myImage[idx + 2] == 11);
76     }else {
77         return false;
78     }
79 }
80
81 protected boolean hitEnemy(short[] myImage, int x, int y){
82     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
83         int idx = (y * 48 + x) * 3;
84         return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
85             myImage[idx + 2] == 222) ||
86             (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
87                 myImage[idx + 2] == 143) ||
88             (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
89                 myImage[idx + 2] == 74) ||
90             (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
91                 myImage[idx + 2] == 215) ||
92             (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
93                 myImage[idx + 2] == 134) ||
94             (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
95                 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 <
155             24);
156         case 5 -> ret = (this.pos[1] + 1 < 24);
157         case 6 -> ret = (this.pos[1] + 1 < 24 && this.pos[0] - 1 >=
158             0);
159         case 7 -> ret = (this.pos[0] - 1 >= 0);
160         case 8 -> ret = (this.pos[0] - 1 >= 0 && this.pos[1] - 1 >=
161             0);
162     }
163     return ret;
164 }
165
166 public short[] run(int key, short[] myImage){
167     if(this.range == this.maxRange){
168         if(move()){
169             myImage = paint(myImage);
170         }else {
171             this.range = 0;
172             myImage = clear(myImage);
173         }
174     }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,
4         221}};
5     protected int hasBonus = 0;
6     protected int orient;
7     protected boolean captured = false;
8     protected int possession = -1;
9     protected int[] pos;
10    protected int[] enemyPos;
11    protected Bullet bullet = null;
12
13    Harbor(int orient){
14        this.orient = orient;
15        this.pos = new int[2];
16        this.enemyPos = new int[2];
17    }
18
19    public int getOrient(){
20        return this.orient;
21    }
22
23    public short[][] getColor() {
24        return this.color;
25    }
26
27    public int getPossession(){
28        return this.possession;
29    }
30
31    public void setPos(int y, int x){
32        if(x >= 0 && x <= 47){
33            this.pos[0] = x;
34        }
35        if(y >= 0 && y <= 23){
36            this.pos[1] = y;
37        }
38    }
39
40    public short[] reset(short[] myImage){
41        this.possession = -1;
42        this.captured = false;
43        this.hasBonus = 0;
44        return myImage;
45    }
46
47    @Override
48    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 &&
71                 myImage[idx + 2] == 12);
72         }else {
73             return false;
74         }
75     }
76
77     public int getHasBonus(){
78         return this.hasBonus;
79     }
80
81     public void setHasBonus(int hasBonus) {
82         this.hasBonus = hasBonus;
83     }
84
85     @Override
86     short[] run(int key, short[] myImage) {
87         if(this.captured){
88             myImage = isHit(myImage);
89             if(this.captured && detectShip(15, myImage) ==
90                 this.possession) {
91                 shoot();
92             }
93             if(this.captured && detectShip(1, myImage) ==
94                 Math.abs(this.possession - 1)){
95                 this.hasBonus = 0;
96             }
97         } 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],
143                                 this.enemyPos);
144     if(this.bullet == null){

```

```

144         this.bullet = new Bullet(orient, 8, this.pos[0], this.pos[1],
145                                   1);
146     }
147 }
148 protected boolean hitPlayer(short[] myImage, int x, int y){
149     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
150         int idx = (y * 48 + x) * 3;
151         return (myImage[idx + 0] == 237 && myImage[idx + 1] == 76 &&
152               myImage[idx + 2] == 36) ||
153               (myImage[idx + 0] == 237 && myImage[idx + 1] == 207 &&
154               myImage[idx + 2] == 36) ||
155               (myImage[idx + 0] == 123 && myImage[idx + 1] == 237 &&
156               myImage[idx + 2] == 36) ||
157               (myImage[idx + 0] == 145 && myImage[idx + 1] == 47 &&
158               myImage[idx + 2] == 22) ||
159               (myImage[idx + 0] == 148 && myImage[idx + 1] == 129 &&
160               myImage[idx + 2] == 22) ||
161               (myImage[idx + 0] == 74 && myImage[idx + 1] == 143 &&
162               myImage[idx + 2] == 21) ||
163               (myImage[idx + 0] == 74 && myImage[idx + 1] == 24 &&
164               myImage[idx + 2] == 11) ||
165               (myImage[idx + 0] == 66 && myImage[idx + 1] == 58 &&
166               myImage[idx + 2] == 10) ||
167               (myImage[idx + 0] == 38 && myImage[idx + 1] == 74 &&
168               myImage[idx + 2] == 11);
169     }else {
170         return false;
171     }
172 }
173 protected boolean hitEnemy(short[] myImage, int x, int y){
174     if (x <= 47 && y <= 23 && x >= 0 && y >= 0) {
175         int idx = (y * 48 + x) * 3;
176         return (myImage[idx + 0] == 31 && myImage[idx + 1] == 69 &&
177               myImage[idx + 2] == 222) ||
178               (myImage[idx + 0] == 19 && myImage[idx + 1] == 43 &&
179               myImage[idx + 2] == 143) ||
180               (myImage[idx + 0] == 10 && myImage[idx + 1] == 22 &&
181               myImage[idx + 2] == 74) ||
182               (myImage[idx + 0] == 31 && myImage[idx + 1] == 222 &&
183               myImage[idx + 2] == 215) ||
184               (myImage[idx + 0] == 21 && myImage[idx + 1] == 138 &&
185               myImage[idx + 2] == 134) ||
186               (myImage[idx + 0] == 11 && myImage[idx + 1] == 74 &&
187               myImage[idx + 2] == 72) ||
188               (myImage[idx + 0] == 153 && myImage[idx + 1] == 23 &&
189               myImage[idx + 2] == 209) ||
190               (myImage[idx + 0] == 94 && myImage[idx + 1] == 15 &&
191               myImage[idx + 2] == 128) ||
192               (myImage[idx + 0] == 55 && myImage[idx + 1] == 10 &&
193               myImage[idx + 2] == 74);
194     }

```

```

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 -
187                     this.pos[1], 2)) <= Math.pow(range, 2)) {
188                     if(hitPlayer(myImage, difx, dify)) {
189                         this.enemyPos[0] = difx;
190                         this.enemyPos[1] = dify;
191                         return 1;
192                     }
193                     if(hitEnemy(myImage, difx, dify)) {
194                         this.enemyPos[0] = difx;
195                         this.enemyPos[1] = dify;
196                         return 0;
197                     }
198                 }
199             }
200         }
201         return -1;
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}}; //
6         normale Insel, Hafeninsel
7     protected int[][] pos;
8     protected int[] size;
9     protected Harbor harbor = null;
10
11     Island(int[] size, int x, int y){
12         this.pos = new int[size[0]][size[1]][2];
13         for(int i1 = 0; i1 < size[0]; i1++){
14             for(int i2 = 0; i2 < size[1]; i2++){
15                 this.pos[i1][i2][0] = x + i1;
16                 this.pos[i1][i2][1] = y + i2;
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 +
37                         this.pos[i1][i2][0]) * 3 + 0] = this.color[0][0];
38                     // (y * 48 + x) * 3 + 0
39                     myImage[(this.pos[i1][i2][1] * 48 +
40                         this.pos[i1][i2][0]) * 3 + 1] = this.color[0][1];
41                     // (y * 48 + x) * 3 + 1
42                     myImage[(this.pos[i1][i2][1] * 48 +
43                         this.pos[i1][i2][0]) * 3 + 2] = this.color[0][2];
44                     // (y * 48 + x) * 3 + 2
45                 }
46                 else{
47                     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)(
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)(
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 *

```

```

79         48 + x) * 3 + 2
80         harbor.setPos(this.pos[(int)(this.pos.length-1)][0][1], this.pos[(int)(this.p
81         break;
82     case 7:
83         myImage[(this.pos[(int)(this.pos.length/2)][0][1]
84             * 48 +
85             this.pos[(int)(this.pos.length/2)][0][0])
86             * 3 + 0] =
87             (short)(harbor.getColor()[harbor.getPossession()
88             + 1][0] - harbor.getHasBonus()); // (y *
89             48 + x) * 3 + 0
90         myImage[(this.pos[(int)(this.pos.length/2)][0][1]
91             * 48 +
92             this.pos[(int)(this.pos.length/2)][0][0])
93             * 3 + 1] =
94             (short)(harbor.getColor()[harbor.getPossession()
95             + 1][1] - harbor.getHasBonus()); // (y *
96             48 + x) * 3 + 1
97         myImage[(this.pos[(int)(this.pos.length/2)][0][1]
98             * 48 +
99             this.pos[(int)(this.pos.length/2)][0][0])
100             * 3 + 2] =
101             (short)(harbor.getColor()[harbor.getPossession()
102             + 1][2] - harbor.getHasBonus()); // (y *
103             48 + x) * 3 + 2
104         harbor.setPos(this.pos[(int)(this.pos.length/2)][0][1], this.pos[(int)(this.p
105         break;
106     case 8:
107         myImage[(this.pos[0][0][1] * 48 +
108             this.pos[0][0][0]) * 3 + 0] =
109             (short)(harbor.getColor()[harbor.getPossession()
110             + 1][0] - harbor.getHasBonus()); // (y *
111             48 + x) * 3 + 0
112         myImage[(this.pos[0][0][1] * 48 +
113             this.pos[0][0][0]) * 3 + 1] =
114             (short)(harbor.getColor()[harbor.getPossession()
115             + 1][1] - harbor.getHasBonus()); // (y *
116             48 + x) * 3 + 1
117         myImage[(this.pos[0][0][1] * 48 +
118             this.pos[0][0][0]) * 3 + 2] =
119             (short)(harbor.getColor()[harbor.getPossession()
120             + 1][2] - harbor.getHasBonus()); // (y *
121             48 + x) * 3 + 2
122         harbor.setPos(this.pos[0][0][1], this.pos[0][0][0]);
123         break;
124     }
125 }
126 }
127 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,
12            11}},{237, 207, 36},{148, 129, 22},{66, 58, 10}},{123,
13            237, 36},{74, 143, 21},{38, 74, 11}};
14        changeColor(rgbs);
15
16        int[][] pos = { {5, 5}, {6, 5}, {7, 5} };
17        this.pos = pos;
18    }
19
20    Player(int hp, int x, int y, int o){
21        super(hp, x, y, o);
22        this.score = new Score();
23        short[][] rgbs = {{237, 76, 36},{145, 47, 22},{74, 24,
24            11}},{237, 207, 36},{148, 129, 22},{66, 58, 10}},{123,
25            237, 36},{74, 143, 21},{38, 74, 11}};
26        changeColor(rgbs);
27    }
28
29    /**
30     * The method collide looks at the pixels of the ship and look if it
31     * collided with another object
32     */
33    public int collide(short[] myImage){
34        int ret = 0;
35        for(int i=0; i < this.pos.length; i++){
36            if (hitIsland(myImage, this.pos[i][0], this.pos[i][1], false)
37                > 0){
38                ret = 1;
39            }
40            if (hitBullet(myImage, this.pos[i][0], this.pos[i][1])){
41                damage(1);
42                ret = 1;
43            }
44            if (hitEnemy(myImage, this.pos[i][0], this.pos[i][1])) {
45                this.hit = true;
46                this.hitX = this.pos[i][0];
47                this.hitY = this.pos[i][1];
48                damage(1);
49            }
50        }
51        return ret;
52    }
53}
```

```

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]

```

```

82         == 121 && myImage[idx + 1] == 235 && myImage[idx
83         + 2] == 34)) ? 2 : 1) : 0;
84     }
85     }else {
86         return 0;
87     }
88 }
89
90 public short[] run(int key, short[] myImage){
91     myImage = isHit(myImage);
92     if(collectBonus(myImage, 1) && !this.hadBonus){
93         this.hp += (this.hp < this.MAXHP) ? 1 : 0;
94         this.hadBonus = true;
95     }
96     if(key != -1){
97         myImage = clearTrace(myImage);
98         move(key,myImage);
99         if (collide(myImage) == 1){
100             resetMove();
101             if(key == 2){
102                 this.align++;
103             }
104             if(key == 3){
105                 this.align--;
106             }
107         }
108     }
109     if(this.bullet != null){
110         if(this.bullet.getHasHit()){
111             addScore(50);
112         }
113         if(this.bullet.getRange() > 0){
114             myImage = this.bullet.run(-1, myImage);
115         }else{
116             this.bullet = null;
117         }
118     }
119     if(this.bullet2 != null){
120         if(this.bullet2.getHasHit()){
121             addScore(50);
122         }
123         if(this.bullet2.getRange() > 0){
124             myImage = this.bullet2.run(-1, myImage);
125         }else{
126             this.bullet2 = null;
127         }
128     }
129     myImage = paint(myImage);
130     return myImage;
131 }

```

```

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
141      * the new and old location.
142      */
143     public void print(){
144         System.out.println("Your ship:\nA: " + this.align);
145         for (int i = 0; i < this.pos.length; i++){
146             System.out.println("X: " + this.pos[i][0] + " Y: " +
147                 this.pos[i][1]);
148             System.out.println("Xo: " + this.oldpos[i][0] + " Yo: " +
149                 this.oldpos[i][1]);
150             // System.out.println("(" + i + ") -> X: " + (this.pos[i][0]
151                 - this.oldpos[i][0]) + " Y: " + (this.pos[i][1] -
152                 this.oldpos[i][1]));
153         }
154     }
155     private static boolean isWater(short[] myImage, int idx){
156         return (myImage[idx] == 0 && myImage[idx + 1] == 177 &&
157             myImage[idx + 2] == 241);
158     }
159     public static Player spawn(short[] myImage, int num){
160         Player ret = null;
161         while (num > 0){
162             int i = (int)(Math.random() * 46) + 1;
163             int j = (int)(Math.random() * 22) + 1;
164             if(
165                 isWater(myImage, (((j-1) * 48 + i) * 3)) &&
166                 isWater(myImage, (((j-1) * 48 + (i+1)) * 3)) &&
167                 isWater(myImage, (((j-1) * 48 + (i-1)) * 3)) &&
168                 isWater(myImage, ((j * 48 + i) * 3)) &&
169                 isWater(myImage, ((j * 48 + (i+1)) * 3)) &&
170                 isWater(myImage, ((j * 48 + (i-1)) * 3)) &&
171                 isWater(myImage, (((j+1) * 48 + i) * 3)) &&
172                 isWater(myImage, (((j+1) * 48 + (i+1)) * 3)) &&
173                 isWater(myImage, (((j+1) * 48 + (i-1)) * 3))){
174                 ret = new Player(3,i,j,(int)(Math.random()*7)+1);
175                 num--;
176                 continue;
177             }
178         }
179         return ret;
180     }

```


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
12         InterruptedException {
13         short[] myImage = new short[24*48*3];
14         List<Integer> highscore = new ArrayList<Integer>();
15         int thisKey=0;
16         int frame = 0;
17         int round = 1;
18         int diff = 1;
19         int enemyHealth = 2;
20         long startTime = System.currentTimeMillis();
21         long roundtime = 30000;
22
23         // This is initialization, donot change this
24         InternalLedGameThread.run();
25
26         System.out.println("Willkommen bei Mari proelium!\n In kuerze
27             wird das Spiel beginnen und Ihr Punktestand wird mit den
28             anderen Spielern verglichen!\n");
29         Thread.sleep(1000);
30
31         boolean end = false;
32         Scanner scan = new Scanner(System.in);
33         do {
34             Fleet fleet = new Fleet();
35             World world = new World();
36             myImage = world.parseImage("intro.mvh");
37             InternalLedGameThread.showImage(myImage);
38             Thread.sleep(1500);
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 -
100         fleet.getNumberOfAliveShips()), enemyHealth);
101     myImage = world.resetHarbor(myImage);
102     startTime = System.currentTimeMillis();
103     p.addScore(200);
104     String s = "round" + round + ".mvh";
105     System.out.println(s);
106     myImage = world.parseImage(s);
107     InternalLedGameThread.showImage(myImage);
108     Thread.sleep(1000);
109     myImage = world.clear();
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: " +
119     p.getScore().getScore());
120 System.out.println("Highscores:");
121 for(int i = 0; i < highscore.size(); i++){
122     System.out.println("(" + (i+1) + ") -> " +
123         highscore.get(i));
124 }
125 myImage = world.parseImage("commands.mvh");
126 InternalLedGameThread.showImage(myImage);
127 Thread.sleep(1500);
128 thisKey = -1;
129 while(true){
130     thisKey = InternalLedGameThread.getKeyboard();
131     if(thisKey != -1){
132         break;
133     }
134 }
135 switch(thisKey){
136     case 0 -> {
137         end = false;
138         thisKey=0;
139         frame = 0;
140         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 &&
27            myImage[idx + 2] == 241);
28    }
29
30    private boolean canPlaceIsland(short[] myImage, int rad, int x, int
31        y){
32        boolean ret = true;
33        for(int i = -rad; i <= rad; i++){
34            for(int j = -rad; j <= rad; j++){
35                if((y+i) >= 0 && (y+i) <= 23 && (x+j) <= 47 && (x+j) >= 0
36                    && isWater(myImage, (((y+i) * 48 + (x+j)) * 3)) &&
37                        ret){
38                    ret = true;
39                }else {
40                    return false;
41                }
42            }
43        }
44        return ret;
45    }
46
47    public short[] createIsland(short[] myImage, int num){
48        this.islands.clear();
49    }
50}
```

```

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
106             File("Game"+File.separator+"src"+File.separator+"de"+File.separator+"thdeg"+File.separator+"as
107             while (myReader.hasNextLine()) {
108                 String data = myReader.nextLine();
109                 if(data.length() > 1 && !data.substring(0,1).equals("/")){
110                     if(data.substring(0,1).equals("B")){
111                         for(int i = 0; i < ret.length; i += 3){
112                             String[] s = data.substring(1).split("-");
113                             ret[i] = Short.parseShort(s[0]);
114                             ret[i + 1] = Short.parseShort(s[1]);
115                             ret[i + 2] = Short.parseShort(s[2]);
116                         }
117                     }else{
118                         String[] pos = data.substring(0,
119                             data.indexOf(';')).split("-");
120                         String[] rgb =
121                             data.substring(data.indexOf(';')+1).split("-");
122                         ret[(Integer.parseInt(pos[1]) * 48 +
123                             Integer.parseInt(pos[0])) * 3 + 0] =
124                             Short.parseShort(rgb[0]);
125                         ret[(Integer.parseInt(pos[1]) * 48 +
126                             Integer.parseInt(pos[0])) * 3 + 1] =
127                             Short.parseShort(rgb[1]);
128                         ret[(Integer.parseInt(pos[1]) * 48 +
129                             Integer.parseInt(pos[0])) * 3 + 2] =
130                             Short.parseShort(rgb[2]);
131                     }
132                 }
133             }
134         catch (FileNotFoundException e) {
135             // e.printStackTrace();
136             ret = clear();
137             return ret;
138         }
139     }
140     myReader.close();
141     return ret;
142 }

```

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 }
```
