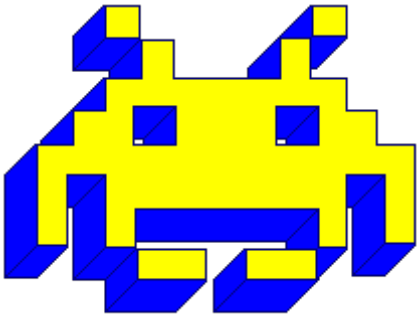
Inoffizielle Lösungen und Erklärungen - inf-schule (*Lizenz*)

7.1 Objektorientierte Programmierung mit Java

2.4

Space Invaders



Benötigte IDEs:

Greenfoot

Verfasser:

Niko Diamadis (Cyb3rKo)

Erstellungs-/ Änderungsdatum

11. September 2021

Inhaltsverzeichnis

1 Spielbeschreibung

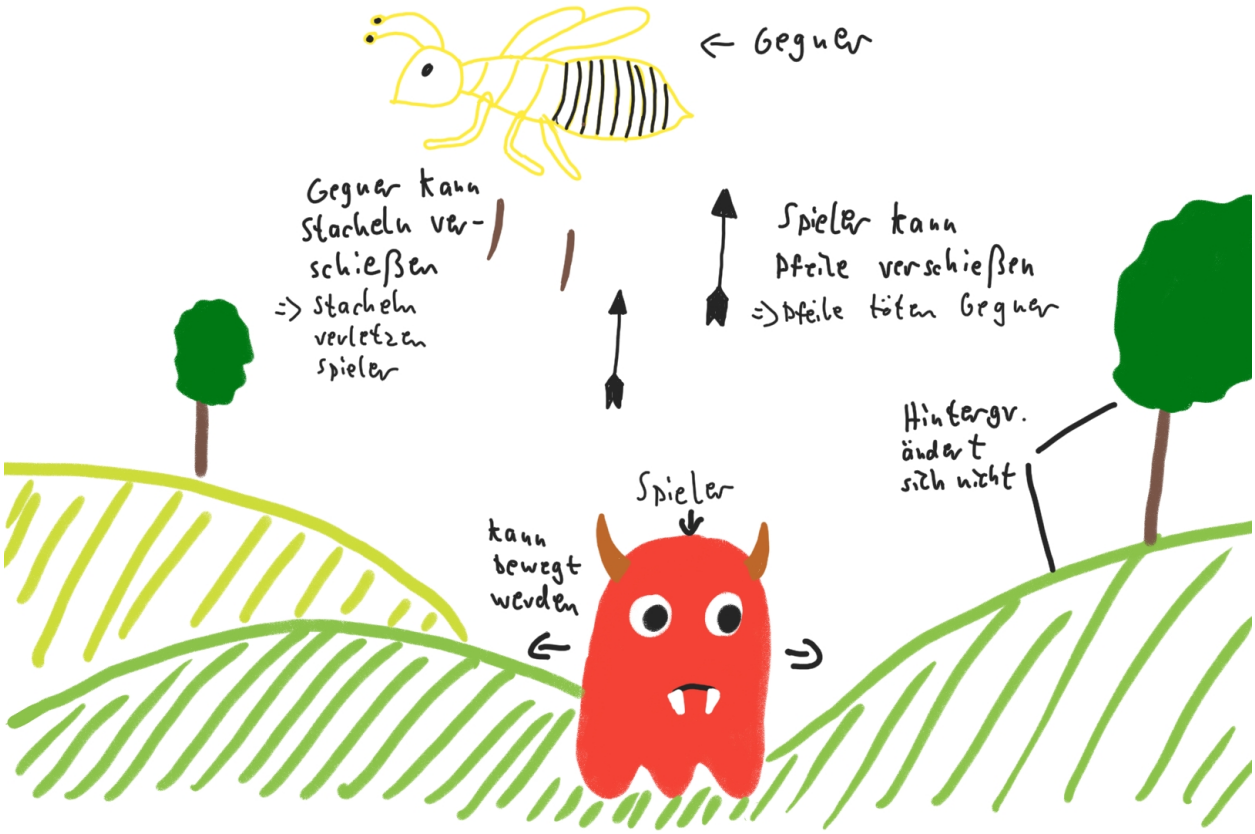
Zu dieser Seite sind meines Erachtens nach keine Anleitungen und/oder Erläuterungen nötig.

Wenn doch Fragen aufkommen, schreib' einfach an **niko@cyb3rko.de**.

2 Entwicklung des Spiels

2.1 Pflichtenheft

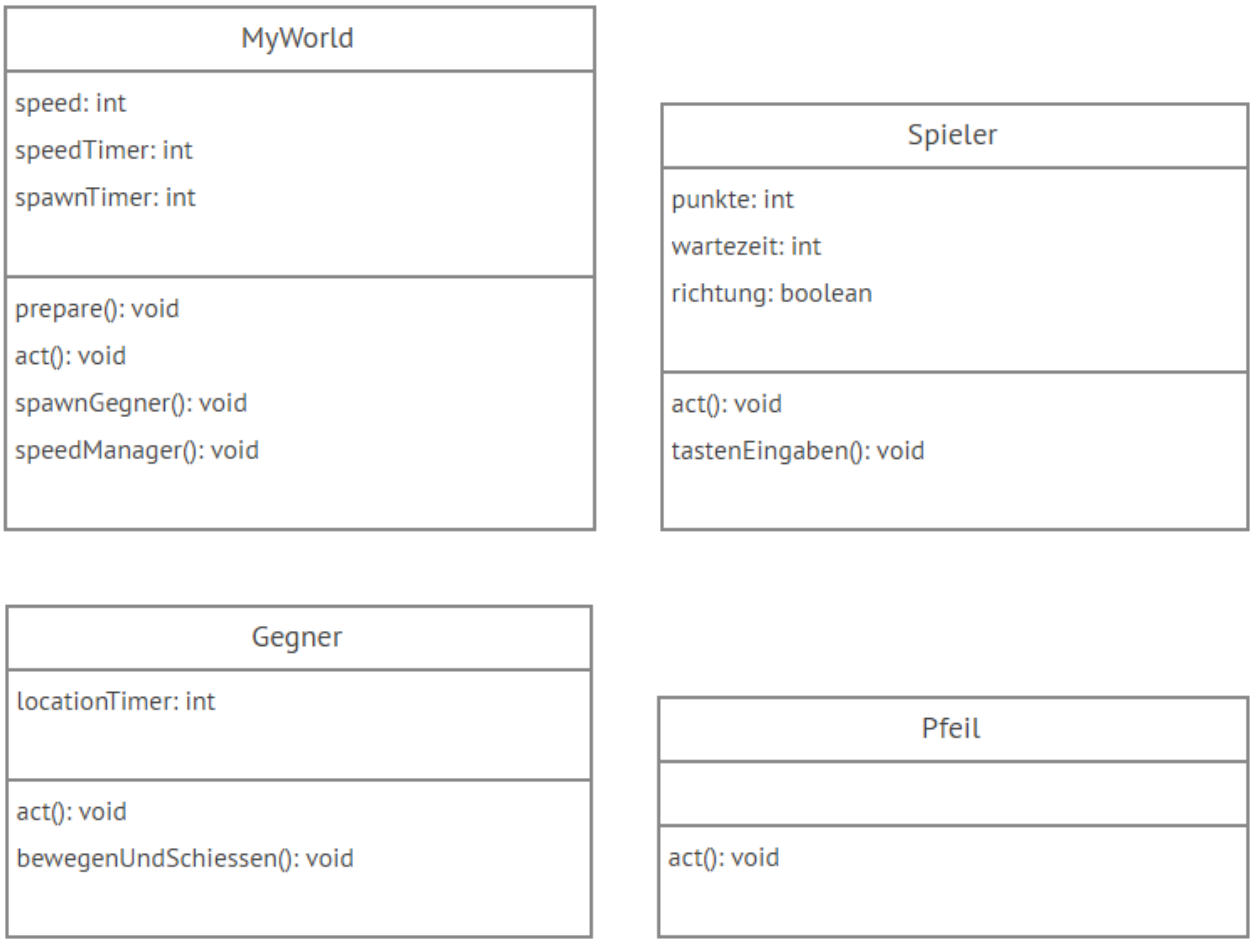
Hier erstmal meine Skizze zu meiner Spiel-Idee:

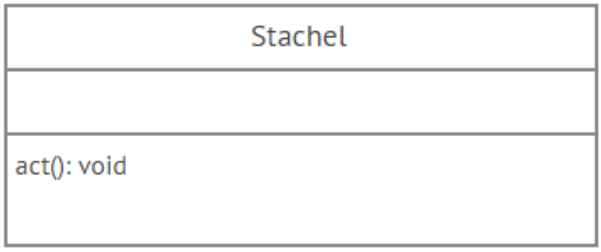


- je länger man überlebt, desto mehr Punkte erhält man
  - Ziel: möglichst viele Punkte sammeln
- mit der Leertaste kann ein Pfeil geschossen werden
  - ein Schuss kostet einen Teil der Punktzahl
  - wenn der Gegner getroffen wird, stirbt / verschwindet er

- der Gegner schießt Stacheln
  - wird der Spieler von einem Stachel getroffen, ist das Spiel vorbei
- [die Geschwindigkeit von Greenfoot wird immer schneller, sodass sowohl alle Bewegungen als auch die Geschosse schneller werden]
- [wenn der Spieler stirbt, wird die erreichte Punktzahl ausgegeben]

2.2 Klassenmodell





➤ **MyWorld**:

- `speed`: wird als Geschwindigkeit von Greenfoot gesetzt
- `speedTimer`: legt die Häufigkeit fest, wann die Geschwindigkeit sich erhöhen soll
- `spawnTimer`: legt die Häufigkeit vom Spawnen neuer Gegner fest
- `prepare`: fügt Spieler an festgelegter Stele hinzu
- `act`: hier wird die Geschwindigkeitsänderung und das Spawnen der Gegner aufgerufen
- `spawnGegner`: spawnt Gegner in einem von `spawnTimer` festgelegten Intervall
- `speedManager`: lässt die Geschwindigkeit immer schneller werden

➤ **Spieler**:

- `punkte`: Punktestand des Spielers
- `wartezeit`: Intervall zwischen Schüssen
- `richtung`: passt sich der Bewegungsrichtung an
- `act`: aktualisiert Punktestand; wartet auf Tasten-Eingaben
- `tastenEingaben`: lässt bei Tastendruck etwas passieren (Bewegen, Schießen)

➤ **Gegner**

- `locationTimer`: legt die Häufigkeit fest, wie oft sich Gegner bewegen und schießen
- `act`: ruft Bewegung und Schießvorgang auf
- `bewegenUndSchiessen`: Bewegung und Schießvorgang

➤ **Pfeil / Stachel**

- `act`: überprüft Berührung mit Gegner / Spieler oder ob es am Ende der Welt ist

2.3 Programm

Da dieses Projekt schon etwas komplexer ist, ist mein fertiger Projektordner HIER herunterzuladen.

Beim Implementieren der gestellten Funktionen gehen wir am besten chronologisch durch.

Angefangen habe ich mit dem simplen Bewegen des Spielers, was auch schon Teil vorheriger Projekte war:

```
1  void tastenEingaben() {
2      if (Greenfoot.isKeyDown("left")) {
3          if (richtung) {
4              getImage().mirrorHorizontally();
5          }
6
7          richtung = false;
8          setLocation(getX() - 10, getY());
9      }
10
11     if (Greenfoot.isKeyDown("right")) {
12         if (!richtung) {
13             getImage().mirrorHorizontally();
14         }
15
16         richtung = true;
17         setLocation(getX() + 10, getY());
18     }
19 }
```

Damit es nicht so seltsam aussieht, wird der Spieler je nach Laufrichtung an der Y-Achse gespiegelt.

Der Rest müsste klar sein...

Der nächste Punkt wäre die Punkteverwaltung:

Bei jedem Aufruf der `act`-Methode wird die Punktzahl also einfach um eins erhöht, außerdem wollen wir es direkt anzeigen lassen.

Eingefügt in den Code sieht das in der Spieler-Klasse dann so aus:

```
1  public void act() {
2      getWorld().showText("Punktestand: " + punkte, 110, 30);
3      punkte++;
4  }
```

Mit den Koordinaten der Textausgabe muss man einfach ein wenig herumprobieren, bis man etwas Sinnvolles gefunden hat.

Damit wäre schon die zweite Funktion fertig, die nächste folgt zugleich. Diese fordert, dass ein Pfeil bei Drücken der Leertaste nach oben geschossen wird.
Der Schuss soll Punkte kosten und bei Kontakt mit einem Gegner soll dieser sterben / verschwinden.
Wir fügen zu der Abfrage der Tasteneingaben noch die Leertaste hinzu und erstellen bei Klick dieser einen neuen Pfeil über dem Spieler, außerdem ziehen wir ein paar Punkte ab:

```
1  void tastenEingaben() {
2          ...
3
4          if (Greenfoot.isKeyDown("space") && wartezeit > 20) {
5              getWorld().addObject(new Pfeil(), getX(), 555);
6
7              punkte -= 30;
8              wartezeit = 0;
9          }
10     }
```

Damit der Pfeil sich nun auch bewegt und bei Kontakt etwas ausführt, wechseln wir zu der `Pfeil`-Klasse. Dort wird in der `act`-Methode bei Kontakt mit einem Gegner der Gegner und der Pfeil entfernt. Solange der Rand der Welt nicht erreicht ist, bewegt sich der Pfeil stetig nach oben. Bei Berühung des Randes der Welt wird der Pfeil auch entfernt:

```
public void act() {
        if (isTouching(Gegner.class)) {
            removeTouching(Gegner.class);
            getWorld().removeObject(this);
        } else if (!isAtEdge()) {
            setLocation(getX(), getY() - 10);
        } else {
            getWorld().removeObject(this);
        }
    }
```

Ziemlich ähnlich läuft das auch mit den Stacheln der Gegner ab.

Zuerst wird ein Stachel in gewissen Abständen von einem Gegner gespawnt:

```
void bewegenUndSchiessen() {
    setLocation(random.nextInt(1280), getY());
    getWorld().addObject(new Stachel(), getX(), 150);
}
```

Die Bewegung des Stachels steht in der `Stachel`-Klasse:

```
public void act() {
    if (!isAtEdge()) {
        setLocation(getX(), getY() + 10);
    } else {
        getWorld().removeObject(this);
    }
}
```

Dieser Code müsste auch verständlich sein...

Nun wollen wir noch das Spiel beenden lassen, falls ein Stachel den Spieler trifft, dazu packen wir folgenden Code einfach noch mit in die `act`-Methode des Spielers:

```
if (isTouching(Stachel.class)) {
    Greenfoot.stop();
}
...
```

Nun haben wir die Hauptfunktionen des Spiels fertig.

Widmen wir uns noch den beiden optionale Funktionen. Die erste ist das Größerwerden der Geschwindigkeit.

Dieses Problem gehen wir in der `MyWorld`-Klasse an, da diese sozusagen die oberste Instanz ist, weil sie die anderen Objekte wie z.B. Spieler, Gegner etc. zur Welt hinzufügt.

Wir ergänzen dort dann folgende Methode:

```
void speedManager() {
    speedTimer--;

    if (speedTimer == 0) {
        speed++;
        Greenfoot.setSpeed(speed);
        speedTimer = 200;
    }
    }
```

Wenn diese Methode nun immer wieder (indirekt über die `act`-Methode) von Greenfoot aufgerufen wird, passt sich die Geschwindigkeit von alleine an.

Der letzte Punkt ist die Ausgabe der Punktzahl, wenn das Spiel beendet wird.

Dazu fügen wir einen Schnipsel aus dem Bereich "Tipps und Tools" direkt auf `inf-schule.` in die `act`-Methode ein:

```
1 if (isTouching(Stachel.class)) {
2     JOptionPane.showMessageDialog(null, "Du hast " + punkte + " Punkte
          erreicht.");
3     Greenfoot.stop();
4 }
5 ...
```

2.4 Spielen

Zu dieser Aufgabe sind meines Erachtens nach keine Anleitungen und/oder Erläuterungen nötig. Wenn doch Fragen aufkommen, schreib' einfach an **niko@cyb3rko.de**.

3 Fachkonzept-Softwareentwicklungsphasen

Zu dieser Seite sind meines Erachtens nach keine Anleitungen und/oder Erläuterungen nötig. Wenn doch Fragen aufkommen, schreib' einfach an **niko@cyb3rko.de**.