INFORMATIK · GREENFOOT · BLOCK 1: METEOR II · ZUSATZÜBUNG 3B

Tiefsee

Taucher, Fische und Luftblasen – Übung zu den Folgen 1 bis 3 und zur Klassenarbeit 1

Worum geht es?

Ein Taucher erkundet die Tiefsee. Von links schwimmen Fische vorbei, vom Meeresboden steigen Luftblasen auf. Sie programmieren die Szene Schritt für Schritt und üben dabei alle Themen der Klassenarbeit 1.

Die Übung wiederholt, was Sie in den **Folgen 1 bis 3 von Meteor II** gelernt haben: ein Objekt in die Welt setzen, es mit W, A, S, D steuern, Objekte zufällig erzeugen und am Rand verschwinden lassen. Neu ist nur die Kulisse – und an einigen Stellen läuft es andersherum als bei Rakete und Meteoren.



So kann die fertige Szene aussehen.

Öffnen Sie in Greenfoot das Projekt **meteor-3b-tiefsee-aufgabe**. Die Bilder sind den Klassen schon zugeordnet.

Klasse	Typ	Was sie tun soll	Entspricht in Meteor II
`Meer`	World	Welt mit 900 × 600 Pixeln, erzeugt Fische und Blasen	`Weltraum`
`Taucher`	Actor	wird mit den Tasten W, A, S, D gesteuert	`Rakete`
`Fisch`	Actor	schwimmt von links nach rechts	`Meteor` (Gegenrichtung)
`Blase`	Actor	steigt vom Meeresboden nach oben	neu

Koordinaten der Welt: x geht von 0 (links) bis 899 (rechts), y geht von 0 (oben) bis 599 (unten).

1 Den Taucher in die Welt setzen *Meteor II: Folge 1*

Setzen Sie im Konstruktor der Klasse `Meer` einen Taucher in die Mitte der Welt, also an die Position (450, 300).

- ☐ Ein neues Taucher-Objekt wird mit `new` erzeugt.
- ☐ Der Taucher wird mit `addObject` an (450, 300) gesetzt.
- ☐ Nach dem Kompilieren ist der Taucher sofort zu sehen, ohne dass Sie ihn per Hand einsetzen.

2 Tastensteuerung mit eigenen Methoden *Meteor II: Folge 2*

Der Taucher soll sich mit W, A, S, D in alle vier Richtungen bewegen, 3 Pixel pro Schritt.

- ☐ Schreiben Sie in der Klasse `Taucher` vier eigene Methoden: `schwimmeHoch()`, `schwimmeRunter()`, `schwimmeLlinks()`, `schwimmeRechts()`.
- ☐ Jede Methode verändert die Position mit `setLocation`, `getX()` und `getY()`.
- ☐ In `act()` rufen Sie die passende Methode auf, wenn die Taste gedrückt ist.

Tipp: Nach oben heißt: y wird kleiner. Bei der Rakete hießen diese Methoden `moveUp()`, `moveDown()`, `moveLeft()` und `moveRight()`.

3 Fische zufällig erzeugen *Meteor II: Folge 3*

In der `act()`-Methode der Welt `Meer` soll mit 2 % Wahrscheinlichkeit pro Schritt ein neuer Fisch erscheinen.

- ☐ Der Fisch erscheint am **linken** Rand (x = 0).
- ☐ Die Höhe (y) ist zufällig zwischen 0 und 599.
- ☐ In der Klasse `Fisch` schreiben Sie eine eigene Methode `schwimme()`: Der Fisch bewegt sich 2 Pixel nach **rechts**.

Tipp: `Greenfoot.getRandomNumber(100) < 2` ist in 2 von 100 Fällen wahr. Achtung: Die Meteore kamen von rechts und flogen nach links – die Fische machen es genau umgekehrt.

4 Fische verschwinden lassen *Meteor II: Folge 3*

Ein Fisch soll aus der Welt entfernt werden, sobald er den rechten Rand erreicht hat.

- ☐ Schreiben Sie dafür eine eigene Methode in der Klasse `Fisch`.
- ☐ Testen Sie genau: Verschwindet der Fisch wirklich – oder bleibt er am Rand hängen?

Denkanstoß: Der Meteor verschwand am linken Rand bei `getX() < 1`. Welche x-Werte kann ein Actor in einer 900 Pixel breiten Welt überhaupt haben – und was heißt das für den rechten Rand?

5 Luftblasen *neu: senkrechte Bewegung*

In der Welt `Meer` soll mit 3 % Wahrscheinlichkeit pro Schritt eine Luftblase erscheinen.

- ☐ Die Blase erscheint am **unteren** Rand (y = 599) an einer zufälligen x-Position.
- ☐ Sie steigt 1 Pixel pro Schritt nach **oben** (eigene Methode `steige()`).
- ☐ Sie verschwindet, sobald sie den oberen Rand erreicht.

Befehle, die Sie brauchen

Befehl	Bedeutung
`this.getX() / this.getY()`	liest die aktuelle Position aus
`this.setLocation(x, y)`	setzt den Actor an eine neue Position
`Greenfoot.isKeyDown("w")`	ist wahr, solange die Taste gedrückt ist
`this.addObject(objekt, x, y)`	setzt ein Objekt in die Welt (in der World-Klasse)
`Greenfoot.getRandomNumber(n)`	Zufallszahl von 0 bis n – 1
`this.getWorld().removeObject(this)`	der Actor entfernt sich selbst aus der Welt

Selbstcheck

- ☐ Ich kann eine eigene Methode schreiben und sie aus `act()` aufrufen.
- ☐ Ich weiß, in welche Richtung sich x und y ändern, wenn sich ein Objekt nach oben, unten, links oder rechts bewegt.
- ☐ Ich kann Objekte zufällig in der Welt erzeugen.
- ☐ Ich weiß, warum ein Fisch am Rand hängen bleiben kann – und wie ich das verhindere.
- ☐ Ich erkenne, welche Teile aus Meteor II ich hier wiederverwenden konnte und was ich anpassen musste.