Programmieren ohne Objekte



In diesem Thema arbeiten Sie mit **virtuellen Figuren**, die sich in einer **virtuellen Welt** bewegen – wie in den meisten **Computer- und Videospielen**. Mittelpunkt des Spiels ist der von Ihnen kontrollierte **namenlose Held**, der in einem Level alle Coins einsammeln und dabei verschiedene Hindernisse überwinden muss.



Um uns auf die **wesentlichen Aspekte** der **objektorientierten Programmierung** zu konzentrieren, programmieren Sie das Spiel nicht komplett neu. Stattdessen ergänzen Sie nur Teile eines bereits vorhandenen Programms. Das nennen wir **Framework** (dt. *Rahmen*). Das ist aus zwei Gründen ähnlich zu realen Programmierarbeiten:

- Sie arbeiten oft im Team und nur an einem Teil des ganzen Programms.
- Spiele-Engines wie Unity, Unreal oder Godot erleichtern das Erstellen von Spielen, indem z. B. das Rendering (dt. Darstellen) der Welten und Figuren bereits implementiert ist.

(Hier läuft im Heft eine kleine Vorschau der Spielwelt.)

Zunächst machen wir uns mit diesem Framework vertraut. Noch gibt es darin keine Objekte: Die Figur hängt nur an ein paar Variablen Ihres Programms.

Aufgabe 1

Die Figur benötigt eine x- und eine y-Koordinate, die wir in den Variablen x und y speichern.

a) Legen Sie in Level 0A die Variablen x und y an und setzen Sie die Koordinaten auf (1|2). Führen Sie das Programm aus.

(noch nicht bearbeitet)

b) Wo befindet sich die Figur nun, und wie ist das Raster aufgebaut? In welche Richtung zählen x und y?

(noch nicht bearbeitet)

Aufgabe 2

Wir möchten die Figur nun ein bisschen bewegen und ihr dabei zusehen. Jede Änderung einer Koordinate zeigt das Framework sofort in der Welt an – mit dem Tempo-Regler unter der Welt können Sie das Abspielen verlangsamen.

a) In **Level 0B** steht Ihr Programm aus Level 0A schon bereit. Fügen Sie ihm die folgenden Zeilen hinzu und führen Sie es aus:

```
x = 2
x = 3
y = 3
```

Beobachten Sie die Figur: Jede Zuweisung bewegt sie um ein Feld. Noch ist sie nicht am Ziel – ergänzen Sie die fehlenden Zeilen selbst. Die Figur darf dabei nur **Feld für Feld** gehen.

(noch nicht bearbeitet)

b) Probieren Sie auf der **Spielwiese** (Level 0S) aus, was passiert, wenn die Figur springen soll: Setzen Sie sie auf (1|2) und schreiben Sie danach z. B. `x = 4`. Auf der Spielwiese gibt es nichts zu lösen – Ihre Lösung von Level 0B bleibt erhalten.

(noch nicht bearbeitet)

c) Warum darf die Figur nicht gleich mit `x = 5` und `y = 4` auf das Ziel gesetzt werden? Und warum ist `x = 4` direkt nach `x = 1` ein Fehler?

(noch nicht bearbeitet)

Aufgabe 3 – Eine kleine Übung

Bisher haben Sie die neue Koordinate jedes Mal selbst hingeschrieben: `x = 2`, `x = 3`, `x = 4` … Bei langen Wegen wird das mühsam. Für Wiederholungen gibt es Schleifen – aber wie bringt man eine Schleife dazu, die Figur jedes Mal ein Feld weiterzusetzen?

a) In **Level 0BR** soll die Figur einfach nur **10 Felder nach rechts** laufen – Feld für Feld. Ihr Programm darf dabei aber **höchstens 7 Zeilen** haben. Verwenden Sie eine `for`-Schleife.

Wenn Sie nicht weiterkommen, rufen Sie die **Tipps** auf – dafür sind sie da.

(noch nicht bearbeitet)

b) Jetzt selbstständig: In **Level 0C** startet die Figur unten links auf dem Feld (0|7). Lassen Sie sie im **Zickzack** nach oben laufen: jeweils einen Schritt nach rechts, dann einen nach oben, nach rechts, nach oben usw.

(noch nicht bearbeitet)

Aufgabe 4 – Simulation

Die Figur soll nun einen sogenannten **Random Walk** (Zufallsspaziergang) durchführen. Das geht so:

- Die Figur startet im Ursprung (0|0).
- In jedem Schritt wählt sie zufällig eine der vier Richtungen: hoch, runter, links oder rechts.
- Sie geht einen Schritt in diese Richtung.
- Dies wiederholt sie *n*-mal, zum Beispiel 100-mal.

a) Setzen Sie den Random Walk in Level 0D um. Am Ende zeigt das Framework in der Ausgabe an, wie weit die Figur vom Ursprung entfernt ist.

(noch nicht bearbeitet)

b) Wie weit entfernt sich die Figur vom Start? Klicken Sie auf **Ausprobieren**: Auf der Spielwiese steht Ihr Programm aus 4a. Ändern Sie die Zahl der Schritte auf 10, 100 und 1000, führen Sie das Programm jeweils mehrmals aus und notieren Sie die Entfernungen.

(noch nicht bearbeitet)

Notieren Sie Ihre Ergebnisse und vergleichen Sie. Wie groß muss n sein, damit die Figur mit großer Sicherheit ein Feld von 16 × 11 Feldern verlässt?

(noch nicht bearbeitet)

Im ursprünglichen Random Walk hat die Figur zusätzlich eine **Blickrichtung**: Sie dreht sich zufällig nach links oder rechts und geht dann einen Schritt nach vorne.

Bonuslevel 0DX (freiwillig)

Rechnen Sie die Entfernung selbst aus: Mit dem Satz des Pythagoras ist sie $d = \sqrt{(*x^{*2} + *y^{*2})}$. Geben Sie sie am Ende mit `print` aus.

Bonuslevel 0DY (freiwillig)

Aufgabe 5 – Das Duell

Wir haben die Figur erfolgreich durch die Simulation laufen lassen. Nun sollen **zwei** Figuren gegeneinander antreten: der Held und die Heldin. Die Koordinaten des Helden stehen in den Variablen `x1` und `y1`.

a) Ergänzen Sie die Variablen `x2` und `y2` für die Heldin. Sie startet auf demselben Feld wie der Held, im Ursprung (0|0). Lassen Sie dann beide **gleichzeitig** 20 Schritte der Simulation durchlaufen: In jedem Durchgang macht jede Figur einen Schritt – und jede entscheidet selbst, in welche Richtung. Am Ende zeigt das Framework für beide getrennt an, wie weit sie vom Start entfernt sind.

(noch nicht bearbeitet)

Freiwillig: Erhöhen Sie auf der Spielwiese die Zahl der Schritte nach und nach und lassen Sie das Duell mehrmals laufen. Gewinnt immer dieselbe Figur?

Bonuslevel 0ES (freiwillig)

b) Nehmen Sie Stellung dazu, noch mehr Figuren gegeneinander antreten zu lassen. Bei Computerspielen sind teilweise Hunderte von Figuren gleichzeitig aktiv. Wie viele Variablen und wie viele Programmzeilen bräuchten Sie dafür?

(noch nicht bearbeitet)

Das Duell mit **Blickrichtung**: Held und Heldin drehen sich zufällig und gehen dann in ihre Blickrichtung.

Bonuslevel 0EX (freiwillig)

Zum Weiterüben (freiwillig)

Viele Befehle lassen sich in **Unterprogramme** auslagern, um das Hauptprogramm zu entlasten. Schreiben Sie zum Beispiel eine Funktion `schritt1()`, die den Helden einen zufälligen Schritt machen lässt, und eine Funktion `schritt2()` für die Heldin. Damit die Funktionen `x1` und `y1` verändern können, müssen Sie diese darin mit `global x1, y1` deklarieren. Welche Parameter bräuchte eine gemeinsame Funktion `schritt(…)` für beide Figuren – und warum ist das so umständlich?