

Lektion 6

Einfaches Jump and Run Spiel

Zum Abschluss wollen wir noch ein richtig cooles Spiel programmieren: Ein Jump and Run Game.

Der Held des Spiels muss dabei die Hindernisse überwinden ohne diese zu berühren. Für jedes Hinderniss gibt es einen Punkt. Wir bedienen uns bei dem Spiel eines coole Tricks: Statt die Figur nach vorne zu bewegen bewegen wir die Hindernisse auf die Figur zu. So wirkt es als würde die Figur laufen aber die Figur kommt nie an einem Ende an. Man nennt so ein Spiel auch Endless-Runner, also Endlos-Renn-Spiel.

The image shows the Scratch programming environment. On the left is the 'Skripte' (Scripts) palette with categories: Bewegung (Movement), Aussehen (Appearance), Klang (Sound), Ereignisse (Events), Steuerung (Control), Fühlen (Sensing), Operatoren (Operators), Variablen (Variables), and Meine Blöcke (My Blocks). The main workspace contains two event-driven scripts. The first script, triggered by 'Wenn grüner Flagge angeklickt wird' (When green flag clicked), includes: 'verstecke dich' (hide), 'warte 1 Sekunden' (wait 1 second), 'zeige dich' (show), and a 'wiederhole fortlaufend' (repeat forever) loop containing 'gehe zu x: 210 y: -130' (go to x: 210 y: -130) and 'gleite in 3 Sek. zu x: -241 y: -130' (slide in 3 seconds to x: -241 y: -130). The second script, also triggered by 'Wenn grüner Flagge angeklickt wird', includes: 'warte bis wird Ghost berührt?' (wait until Ghost is touched), 'spiele Klang pop ganz' (play sound pop at volume ganz), and 'stoppe alles' (stop all). On the right is a preview window showing a blue ghost character on a dark stage with a city skyline background and two yellow sun-like obstacles. Below the preview is the 'Figur' (Sprite) control panel for 'Sun2', showing x: 4, y: -130, size: 33, and direction: 90. The 'Bühne' (Stage) panel shows the background image and a sprite list containing 'Ghost', 'Sun', and 'Sun2'.

Die Gestaltung

- Suche dir eine Figur aus und ein Hinderniss (A), sowie einen Ort an dem das Spiel stattfinden soll.

Passe nun die Position auf der Bühne und die größe von Figur und Hinderniss an. Ähnlich wie bei (C) abgebildet.



Figur: Sun, x: 68, y: -1

Sichtbarkeit:

Größe: 100, Richtung: 90

Bühne: Hintergrundbilder: 2

Ghost, Sun



Figur: Sun, x: 220, y: -131

Sichtbarkeit:

Größe: 33, Richtung: 90

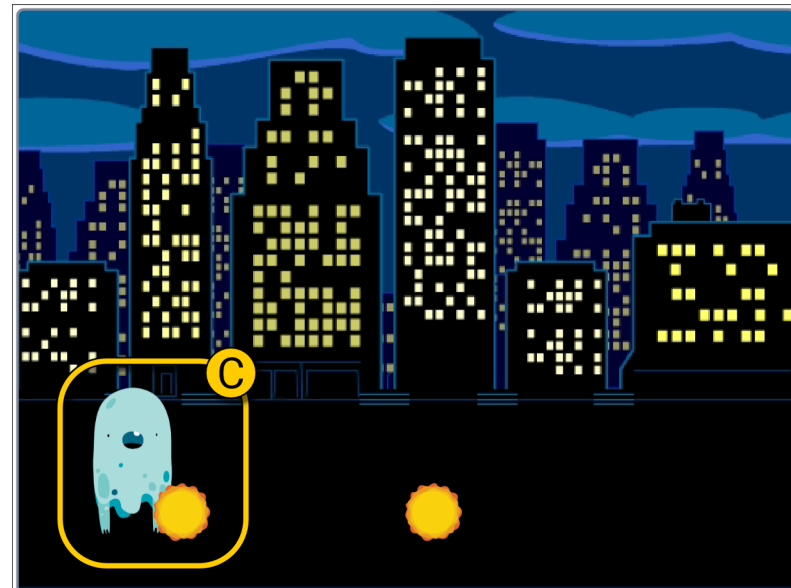
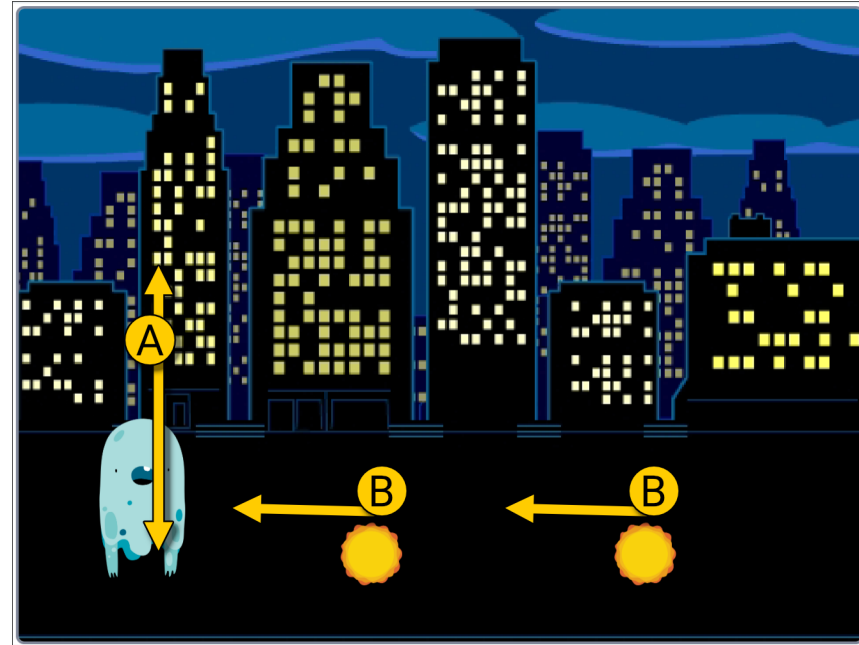
Bühne: Hintergrundbilder: 9

Ghost, Sun

Was wir programmieren wollen

Bevor man losprogrammiert sollte man sich immer genau überlegen was man eigentlich programmieren will.

- (A) Die Figur soll springen können, also sich hoch und runter bewegen.
- (B) Die Hindernisse sollen sich von rechts nach links bewegen.
- (C) Wenn die Figur ein Hinderniss berührt soll das Spiel zu Ende sein.



Figur: Springen programmieren

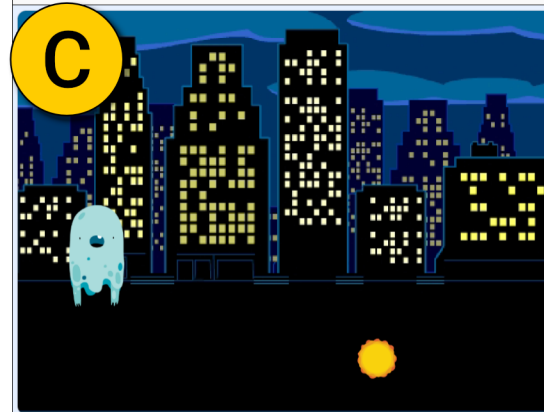
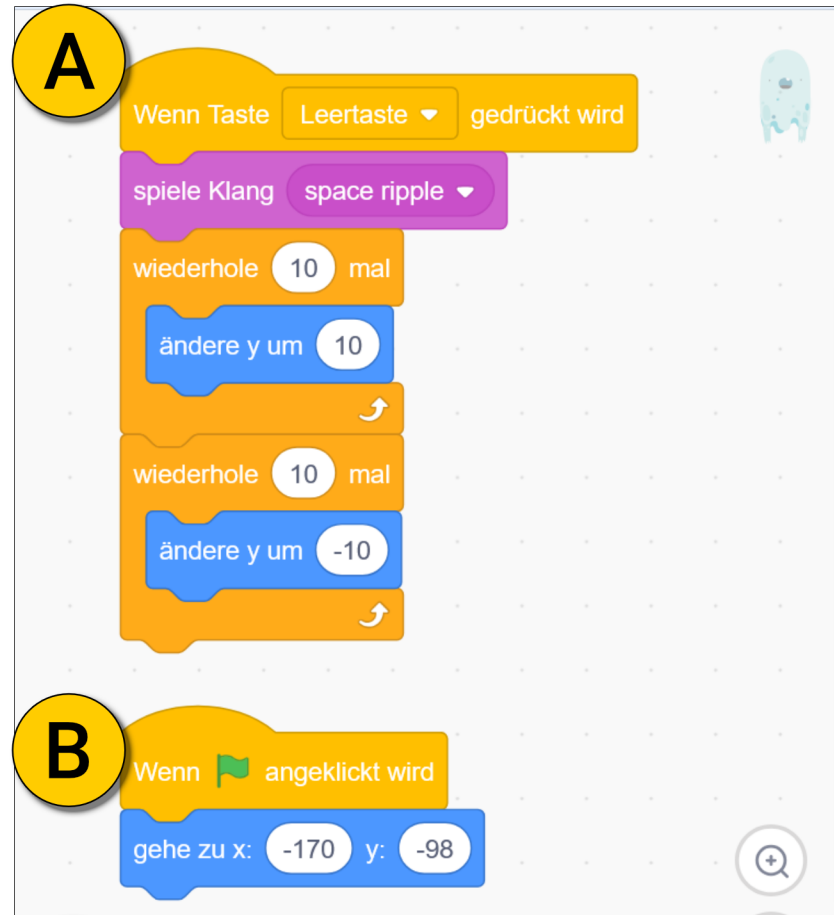
Für die Figur benötigst du zwei Skripte:

(A) Beim Drücken der Leertaste die Figur ein Geräusch machen und sich dann jeweils 10 mal in 10er Schritten nach oben und dann nach unten bewegen. Das sieht dann aus als würde die Figur springen.

Tipp: Achte darauf nicht den Block "spiele Klang ... ganz" zu nutzen. Das Wörtchen ganz führt dazu, dass die Figur erst nach abspielen des Sounds springt. Sie soll aber natürlich direkt springen während der Sound abgespielt wird.

(B) Das zweite Skript stellt sicher, dass die Figur an dem gewählten Startpunkt startet.

(C) Probiere deinen Code aus. Die Figur sollte bei Druck auf die Leertaste springen!.



Hindernisse - Bewegung und Kollision programmieren

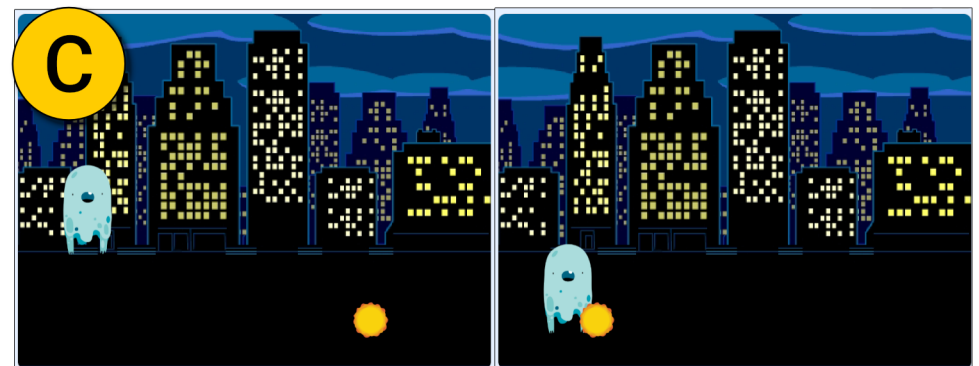
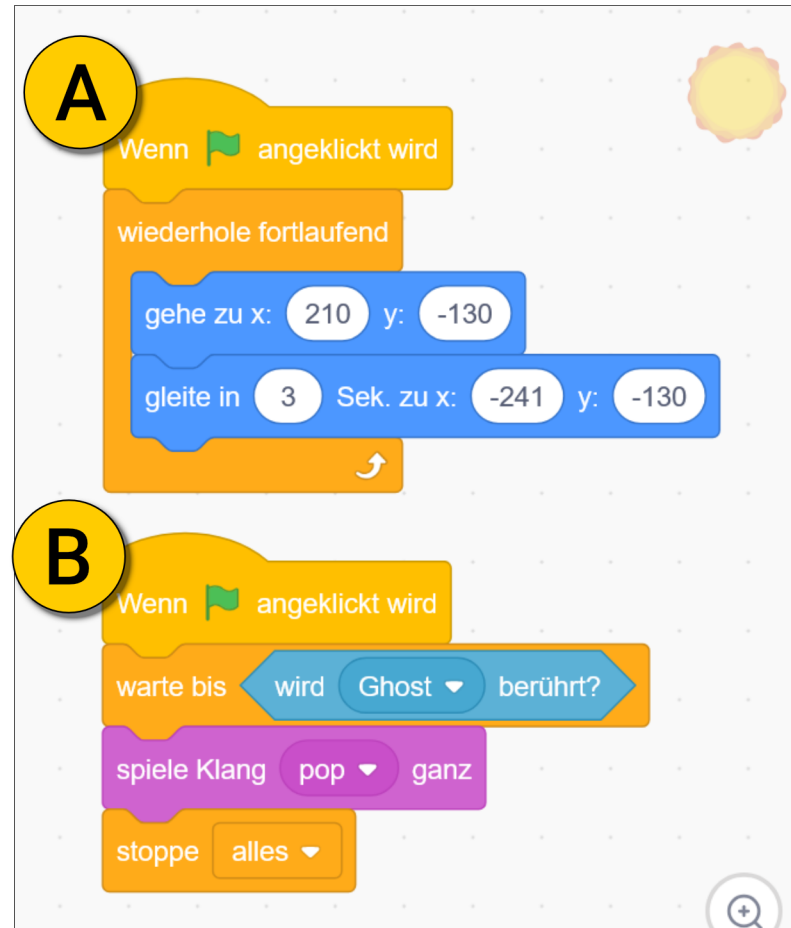
Auch die Hindernisse bekommen zwei Skripte:

(A) Das erste Skript lässt die Hindernisse von rechts nach links gleiten und wieder zurück zum Anfang springen.

Tipp: Wenn du nicht die Werte in den Bewegungsblöcken eintippen willst kannst du immer auch zuerst die Figur auf der Bühne dahinbewegen wo du sie haben möchtest. Ziehst du dann einen Bewegen-Block auf die Bühne sind bereits die aktuellen X- und Y-Werte der Figur eingetragen.

(B) Dieses Skript wartet darauf, dass das Hinderniss die Figur berührt. Passiert das, wird ein Sound abgespielt und anschließend wird alles gestoppt. Dieser block stoppt alle Skripte im Spiel und das Spiel endet

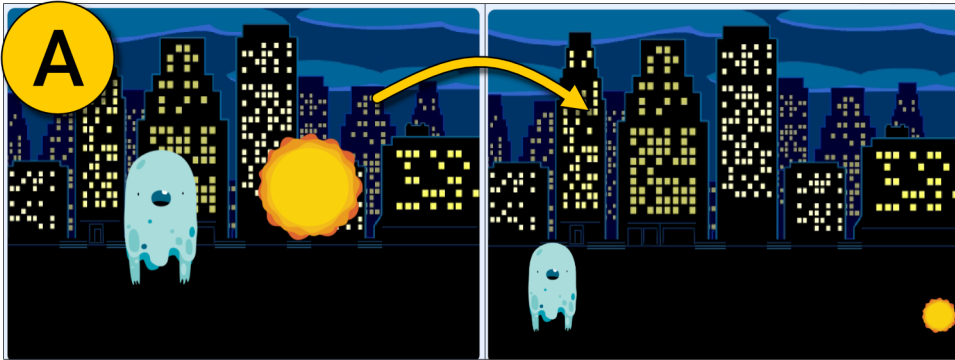
(C) Probiere den Code aus



Game Design

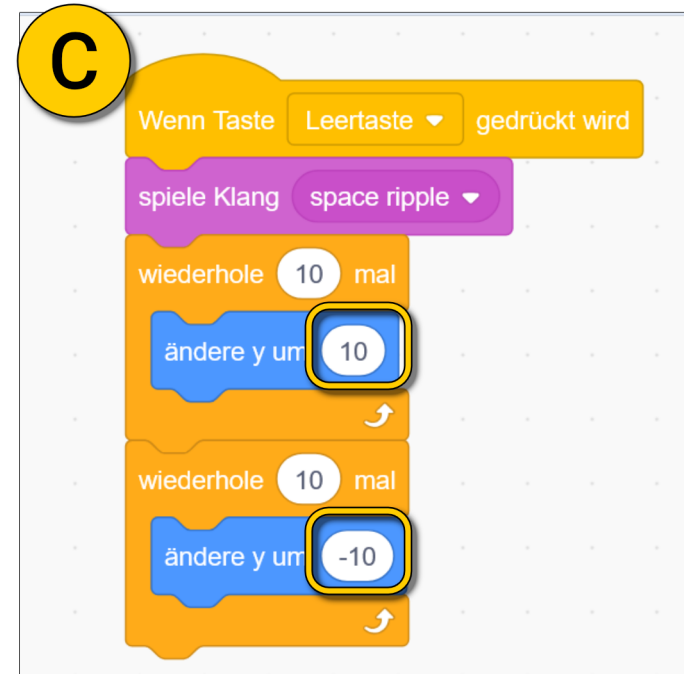
Du kannst verschiedene Aspekte des Spiel anpassen um es leichter oder schwieriger zu machen. Das Spiel soll weder langweilig noch frustrierend sein, sondern Spaß machen.

(A) Am wichtigsten ist die Größe von der Figur und dem Hinderniss so anzupassen, dass die Figur über das Hinderniss springen kann.



Außerdem kannst du mit dem Wert bei **(B)** verändern wie schnell das Hinderniss ist.

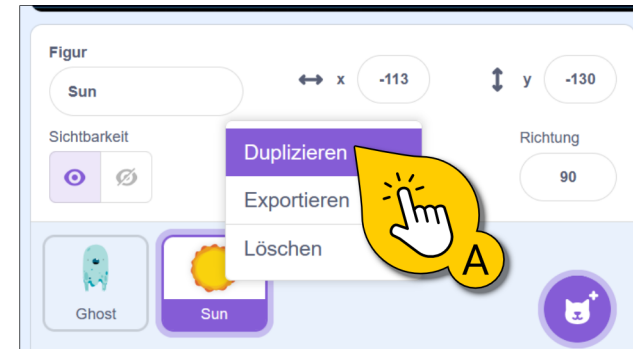
Mit den Werten bei **(C)** kannst du verändern wie ich die Figur springt. Wichtig ist, dass der untere Wert der negative Wert des oberen ist.



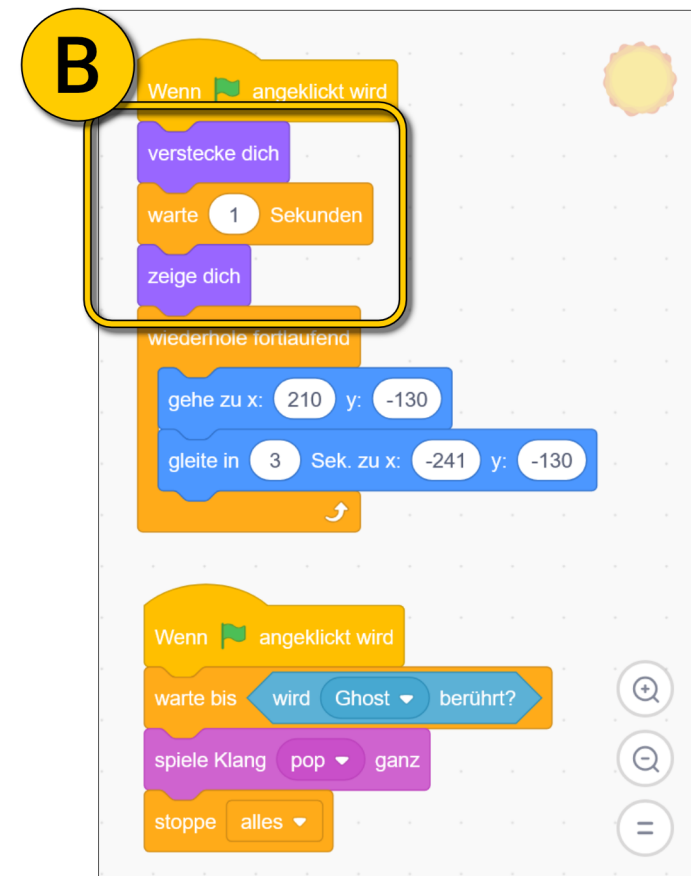
Ein zweites Hinderniss einfügen

Wir wollen jetzt ein weiteres Hinderniss hinzufügen.

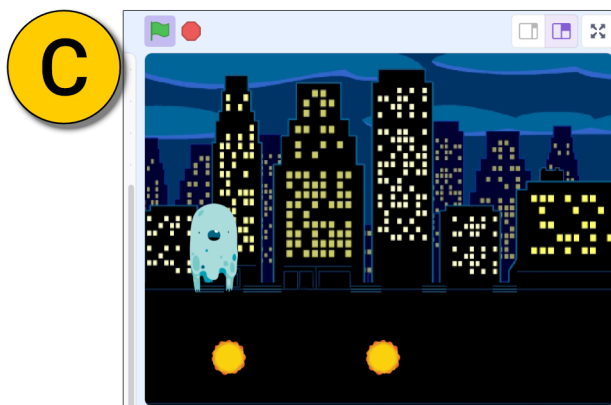
(A) Dupliziere das vorhandene Hinderniss.



(B) Füge die markierten Blöcke in das Skript des zweiten Hindernisses ein. Diese drei Blöcke sind wichtig, um das zweite Hinderniss kurz zu verstecken und es eine Sekunde später (sichtbar) loszuschicken. Ohne diese Blöcke würden beide Hindernisse immer zugleich starten.



(C) Probiere den Code aus.



Punkte zählen

Fast geschafft! Aber wir brauchen noch eine Punktezählung. Jedes Hindernis das überwunden wurde soll einen Punkt geben!

(A) Erstelle eine Variable "Punkte".

(B) Ergänze die markierten Blöcke im ersten Hindernis

(C) Im zweiten Hindernis musst du den Code auch anpassen. **Tip**: Es genügt nur in einem Hindernis die Punkte auf 0 zurückzusetzen.

The screenshot shows the Scratch code editor interface. On the left, the 'Variablen' (Variables) panel is open, showing a variable named 'Punkte' with a value of 0. A yellow callout bubble with a hand icon and the letter 'A' points to the 'Punkte' variable. In the center, two script blocks are visible. The first block is a 'Wenn angeklickt wird' (When clicked) block followed by a 'wiederhole fortlaufend' (Repeat) block. Inside the repeat block, there are three blocks: 'gehe zu x: 210 y: -130' (Go to), 'gleite in 3 Sek zu x: -241 y: -130' (Slide to), and 'ändere Punkte um 1' (Change Punkte by 1). A yellow callout bubble with a hand icon and the letter 'B' points to the 'ändere Punkte um 1' block. The second script block is another 'Wenn angeklickt wird' block followed by a 'warte bis wird Ghost berührt?' (Wait until Ghost is touched?) block. Inside the wait block, there is a 'setze Punkte auf 0' (Set Punkte to 0) block. A yellow callout bubble with a hand icon and the letter 'B' points to this 'setze Punkte auf 0' block. Below the wait block, there is a 'spiele Klang pop ganz' (Play sound pop full) block and a 'stoppe alles' (Stop all) block. On the right, the game preview window shows a cityscape at night with a ghost character and two sun-like obstacles. A score counter at the top left of the preview shows 'Punkte 3'. Below the preview, the 'Figur' (Sprite) panel is open, showing the 'Sun' sprite selected. A yellow callout bubble with a hand icon and the letter 'C' points to the 'Sun' sprite. The bottom of the screen shows a 'Lager' (Library) panel with various assets.

Auf Ereignisse warten - Der “warte auf” Block

In dieser Lektion hast du eine weitere Möglichkeit kennengelernt Ereignisse zu prüfen. Das wollen wir uns genauer anschauen.

Bei **(A)** siehst du zwei Beispiele:

- Das linke Skript wartet bis die Figur “Ghost” berührt wird, um dann **einmal** ein Game Over auszulösen.
- Das rechte Skript prüft ob der Mauszeiger berührt wird, um ein **dauerhaftes** Drehen auszulösen.

Du siehst, “warte bis ...” ist besser geeignet für **einmalige** Ereignisse.

Außerdem ist wichtig, dass “warte bis ...” immer bedeutet, dass während des Wartens nichts anderes passiert. “Falls, dann” kann man hingegen nutzen um mehrere Möglichkeiten gleichzeitig zu prüfen. Bei **(B)** siehst du wie das aussehen kann.

Das Skript bei **(B)** macht übrigens genau das Gleiche wie die beiden Skripte bei **(A)**.

Tipp: Beim Programmieren gibt es selten die eine beste Lösung. Stattdessen gibt es oft viele Lösungen.

