

**Debugging für Neulinge**

**I 1a**



uf/1/17

Recht häufig kommt es vor, dass ein Computer-Programm beim ersten Test alles andere macht, als was es sollte. In den seltensten Fällen ist dann der Computer schuld. Meistens hat der Scratcher einen Denkfehler oder eine Unaufmerksamkeit begangen und muss nun den Fehler suchen.

Programmierer nennen das *Debugging*. Das Wort kommt vom Wort *Bug*. In grauer Computer-Vorzeit, im Jahre 1947, wurde eine Motte als Ursache für eine Computerpanne ausgemacht und seither hält sich der Name Bug (Ungeziefer, Wanze) für Programmfehler.

Debuggen ist nicht besonders spannend, aber manchmal kommt man nicht darum herum. Die nachfolgende Checkliste kann dir beim Suchen helfen, wenn dein Programm nicht so läuft, wie du dir das vorgestellt hast:

1. Skript am falschen Ort

Hast du das Skript beim richtigen Sprite geschrieben?

2. Vergessene Blocks

Gehe für dich in Gedanken die einzelnen Blocks durch. Machen sie Sinn oder fehlt etwas? Falls du einen Code abschreibst: Zähle die Blocks nach. Vielleicht kannst du deinen Code auch mit deiner Kollegin, deinem Kollegen vergleichen.

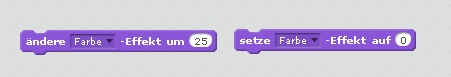
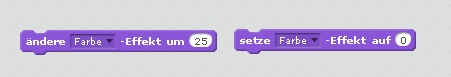
3. Verwechselte Blocks

Hast du die richtigen Blocks verwendet? Besonders häufig werden Blocks mit relativen und absoluten Anweisungen verwechselt:

↔



↔



↔



↔



↔

Auch Blocks mit ähnlichen Anweisungen:

↔



Vielleicht haben die Blocks aber auch nicht die richtige Farbe und damit eine falsche Funktion:

↔



↔



4. x und y verwechselt

Unter Umständen hast du einen X-Block statt einen Y-Block erwischt oder umgekehrt?

↔



↔



5. Falsche Zahlen

Gerade wenn du ein Skript oder ein Skript-Teil kopiert hast, ist es oft notwendig, noch die Zahlen oder das Vorzeichen zu ändern:

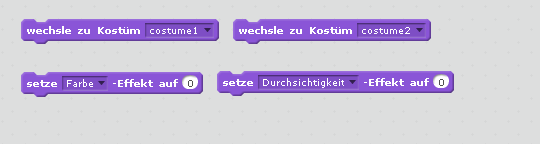
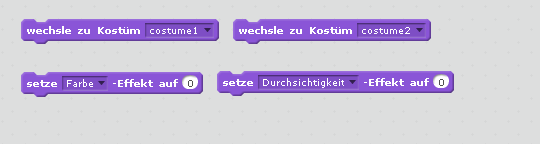
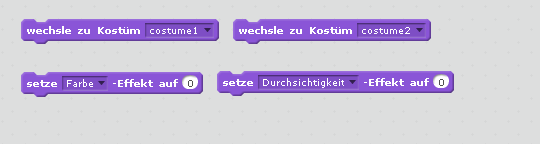
↔



6. Falsche Auswahl

Ebenfalls geht das Ändern der Dropdown-Menüs schnell vergessen:

↔



↔

7. Falsche Reihenfolge

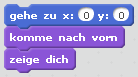
Sind die Blocks in der korrekten Reihenfolge?

→



Tipp: Sprite verschwunden?

Schreibe das folgendes Skript in den Skriptbereich des vermissten Sprites und klicke darauf: →



**Debugging für Fortgeschrittene**

**I 1b**



uf/1/17

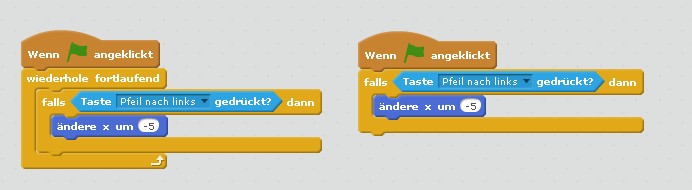
8. Schleifen

Sind Blocks innerhalb von Schleifen, wenn sie ausserhalb sein sollten oder umgekehrt?

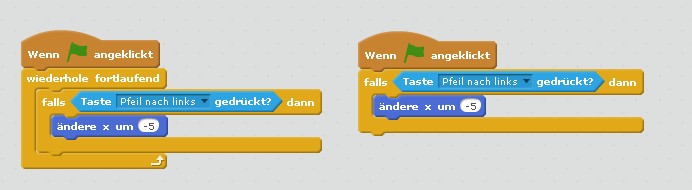
→



Oder fehlt gar eine Schleife? Statt so,



sollte das Skript so aussehen:



9. Zeitliche Begrenzung

Sind Aktionen zeitlich begrenzt, wo sie es nicht sein sollten oder umgekehrt?

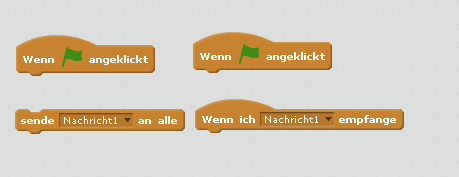
↔



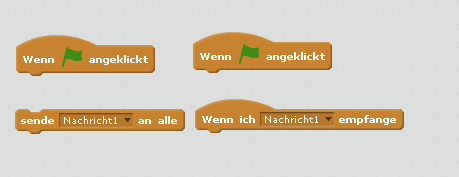
↔

10 Falsches Timing

Um einen genauen zeitlichen Ablauf mit mehreren Sprites festzulegen, sind die *sende-empfange*-Blocks den *warte*-Blocks vorzuziehen.



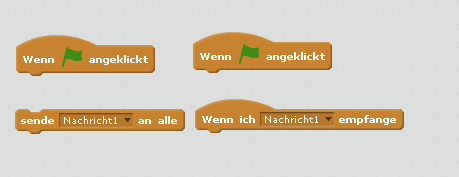
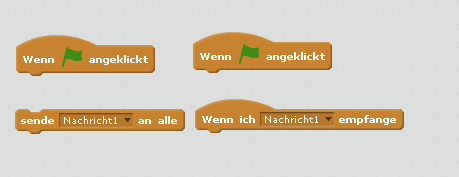
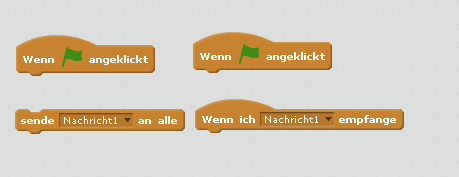
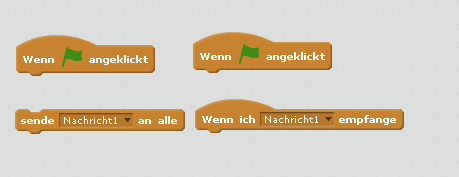
→



11. Gleichzeitigkeit

Wenn mehrere Skripts mit dem Flaggenblock beginnen, ist unter Umständen nicht klar, welches Skript zuerst abgearbeitet werden soll. *Warte*-Blocks einschieben kann da helfen. Besser ist es aber, die *sende-empfange*-Blocks zu verwenden, um den genauen Ablauf festzulegen:

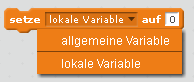
→



12. Lokale oder allgemeine Variable?

Kannst du bei einem Sprite eine Variable nicht einsetzen, hast du sie vermutlich als lokale Variable definiert. Lokale

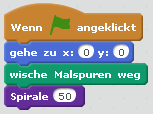
Variablen erscheinen im Dropdown-Menü unterhalb eines schwarzen Querstriches. →



13. Initialisieren

Beim Initialisieren (bei den Startvorbereitungen) musst du darauf achten, dass Spuren des letzten Skript-Gebrauchs nicht den Neustart beeinflussen.

→



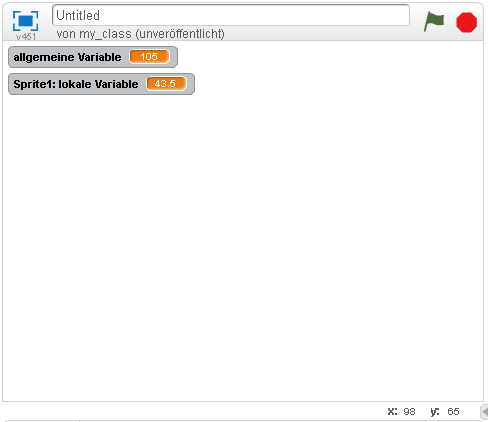
14. Zu viele Klone

Wenn dein Skript Klone erstellt, müssen einige davon unter Umständen zuerst wieder gelöscht werden, bevor neue erstellt werden. →



Tipp: Werte/Zustände testen

Um besser verstehen zu können, was das Programm gerade macht, ist es oft hilfreich, die Variablenwerte auf der Bühne anzeigen zu lassen: →



Möglich ist es aber auch, einen Wert laufend durch einen *sage*-Block anzeigen zu lassen:



Diese Methode hat den Vorteil, dass sie auch noch für andere Informationen herangezogen werden kann:

