


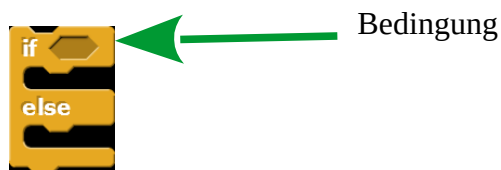


Zweiseitig bedingte Anweisung


 Aufgabe 1	<div>Die Simpsons</div> <div>Öffne die Datei Simpsons.xml aus der letzten Stunde in Snap! .</div> <div></div> <div>Bisher ist es den beiden Figuren noch möglich, sich aus dem Zeichenfenster heraus zu bewegen - zu „verschwinden“. Dies möchten wir im Folgenden unterbinden.</div> <div> Verändere dein Programm derart, dass die Figuren sich <i>nicht mehr aus dem Zeichenfenster heraus</i> bewegen können.</div>
---	--




Bei genaueren Überlegungen stellen wir fest, dass die Reaktion auf die Tastatur **nur dann** funktionieren soll, **wenn** die Figur nicht den Rand berührt. **Ansonsten** muss sie vom Rand wieder zurück ins Spielfeld bewegt werden und kann erst dann wieder auf die Tastatur reagieren.




Für solche Fälle gibt es einen weiteren wichtigen Baustein für Algorithmen - die **zweiseitig bedingte Anweisung**. Diese wird verwendet, um **in Abhängigkeit von einer Bedingung eine von zwei möglichen Sequenzen auszuführen**. Es wird also nicht nur eine Programmteil ausgeführt, falls die Bedingung eintrifft, sondern es ist auch ein alternativer Programmteil vorhanden, der ausgeführt wird, falls die Bedingung nicht eintrifft.






Das Programm überprüft – wie bei der einseitig bedingten Anweisung – an der entsprechenden Stelle, ob die Bedingung erfüllt wird. Ist dies der Fall – und die Bedingung damit wahr –, so wird die erste Sequenz ausgeführt. Ansonsten – falls die Bedingung falsch ist – wird die zweite Sequenz ausgeführt.

	<p>Bedingung → führe den folgenden Programmcode aus, falls der Rand berührt wird</p> <p>Programcode, der ausgeführt wird, falls der Rand berührt wird</p> <p>Programcode, der ausgeführt wird, falls der Rand nicht berührt wird</p>
---	--

 Aufgabe 2	<div> <div>Die Simpsons</div> <div>  </div> </div> <p>Ändere das Verfolgungsspiel mit den Simpsons mit Hilfe der zweiseitig bedingten Anweisung derart ab, dass die Spieler die Zeichenfläche nicht mehr verlassen können.</p> <div>  <div>Teste dein Programm anschließend.</div> </div>
---	--

 Aufgabe 3	<div> <div>Die Simpsons – Der Donut</div> <div>  </div> </div> <p>Um das Spiel für den Homer-Spieler etwas schwieriger zu gestalten, soll nun ein Hindernis in Gestalt eines Donuts an beliebiger Stelle im Zeichenfenster eingebaut werden. Dazu steht dir in den Vorlagen der Ordner Simpsons mit den erforderlichen Dateien zur Verfügung.</p> <p>Wird der Donut von Homer berührt, so beginnt Homer zu essen: Er ändert sein Aussehen entsprechend, schwärmt vom Donut und wartet 1 Sekunde. Erst dann kann er sich  wieder bewegen.</p> <p>Tipp:</p> <ul style="list-style-type: none"> • Homer muss sich erst wieder vom Donut entfernen, um sich wieder bewegen zu können. • Vergiss nicht, Homer wieder das ursprüngliche Aussehen zu geben, wenn er sich wieder bewegen kann.
---	---

 Aufgabe 4	<div> <div>Die Simpsons – Der Donut</div> <div>  </div> </div> <p>Im Folgenden soll der Donut nun derart programmiert werden, dass er sich zufällig immer wieder neu im Zeichenfenster platziert.</p> <p>Tipp:</p> <ul style="list-style-type: none"> • Verwende zum Erzeugen einer Zufallszahl folgenden Baustein:  • Es bietet sich an, Werte zwischen -250 und 250 zu erlauben. Damit wird das ganze Zeichenfenster abgedeckt.
---	---



Tipps und Tricks

- Falls du eine **Bedingung** ins **Gegenteil** umkehren möchtest, kannst du den Operator **not** verwenden.



- Bedingte Anweisungen (einseitig oder zweiseitig) und Wiederholungen (mit fester Anzahl oder bedingt) können **beliebig oft ineinander verschachtelt** werden.

Beispiel:

