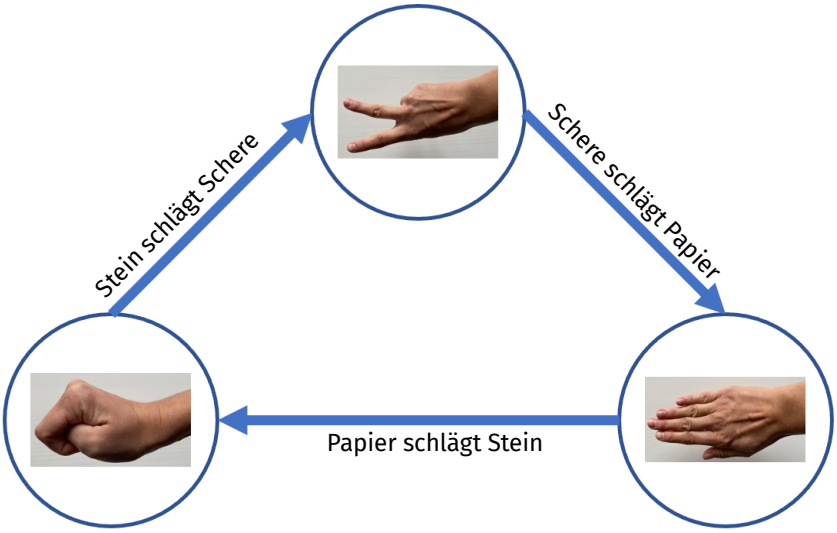
Calliope mini – Online-Modul

Projekt: Schnick, Schnack, Schnuck

Nach der Bearbeitung dieses Arbeitsblatts könnt ihr den Calliope mini nutzen, um *Schnick, Schnack, Schnuck* zu spielen.

In diesem Zusammenhang lernt oder wiederholt ihr, was man in der Informatik unter Variablen und bedingten Verzweigungen versteht.

Das Spiel *Schnick, Schnack, Schnuck* kennt wohl jede*r. Zwei Spieler*innen wählen im Kopf je eins von drei möglichen Symbolen und zeigen dieses mittels ihrer Hände dann gleichzeitig. Da die drei Symbole *Schere*, *Stein* und *Papier* unterschiedliche Werte besitzen und jedes Symbol gegen ein anderes gewinnen kann, steht am Ende jeder Runde ein Sieger oder eine Siegerin fest. Hier noch einmal die Zusammenfassung, denn diese Regeln gelten auch später für euer Spiel gegen den Calliope mini:



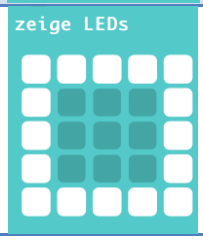
Ihr werdet den Calliope mini nun als euren Gegenspieler programmieren. Das Spiel wird dann so funktionieren:

- Ihr beginnt, indem ihr ganz normal eines der Symbole mit eurer Hand zeigt.
- Gleichzeitig (oder kurz danach) drückt ihr Knopf A: Beim Drücken von Knopf A soll ein zufälliges Symbol (Schere, Stein oder Papier) auf dem LED-Display des Calliope mini angezeigt werden.
- Der Gewinner oder die Gewinnerin der Runde steht fest: ihr oder der Calliope mini.
- Um den Spielzug zu beenden, drückt ihr Knopf B: Beim Drücken von Knopf B soll die Bildschirm-Anzeige **gelöscht** werden.

Bevor ihr loslegt: Denkt daran, dass ihr ein **neues Projekt** anlegt SchnickSchnackSchnuck und es unter einem passenden Namen **speichert.**

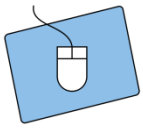
Als Erstes müssen die Symbole *Schere, Stein, Papier* in eine für den Calliope mini darstellbare Form gebracht werden. Die Symbole können mit Hilfe des LED-Bildschirms abgebildet werden. Wie man den LED-Bildschirm programmiert, wisst ihr bereits aus den Herausforderungen. In der Tabelle findet ihr ein paar Vorschläge für die Darstellung der Symbole. Ihr könnt aber auch andere Darstellungen wählen, solange ihr diese später im Spiel wiedererkennt.

Projekt: Schnick, Schnack, Schnuck

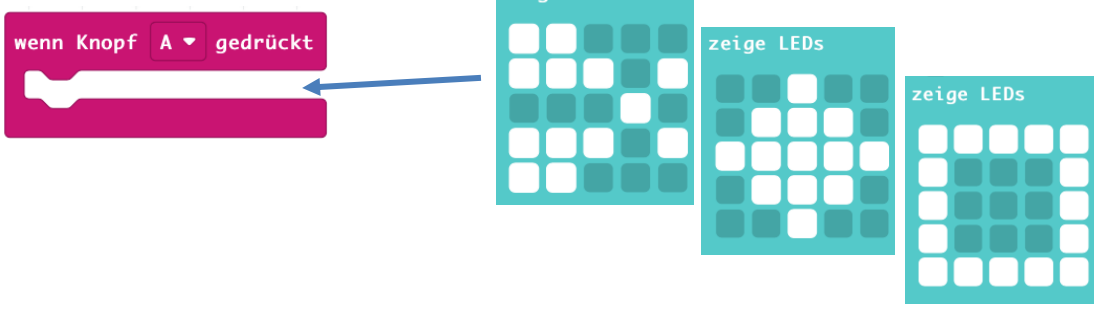
0		
1		
2		

Computer zählen anders

Die Symbole der Tabelle sind nummeriert, beginnend mit der 0. Das ist so, weil Computer im Gegensatz zu uns Menschen bei 0 anfangen zu zählen. Später in der Programmierung werdet ihr jedem Symbol eine Zahl (hier zwischen 0 und 2) zuweisen. Wählt der Calliope mini dann per Zufall eine der Zahlen zwischen 0 und 2, kann er das entsprechende Symbol zur Zahl aufrufen.



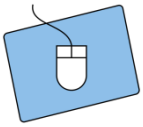
Stellt die Symbole mit Hilfe der LED-Blöcke dar. Sie sollen angezeigt werden, wenn Knopf A gedrückt wird.



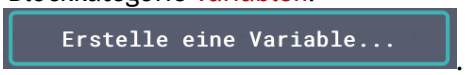
Testet das Programm.

Wahrscheinlich habt ihr gemerkt, dass die Symbole nun immer in derselben Reihenfolge und nacheinander angezeigt werden. Ziel ist aber, dass nur ein zufälliges Symbol aufgerufen und angezeigt wird. Um dies programmieren zu können, legt ihr nun eine sogenannte Variable für die Symbole an.

Projekt: Schnick, Schnack, Schnuck



Erstellt eine Variable für eure Symbole:

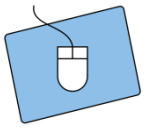
1. Öffnet dazu die Blockkategorie Variablen.
2. Klickt dann auf 

.
3. Gebt der Variable einen passenden Namen, zum Beispiel symbol. Klickt dann auf „Ok“. Es erscheinen dann neue Programmierblöcke für eure Variable.

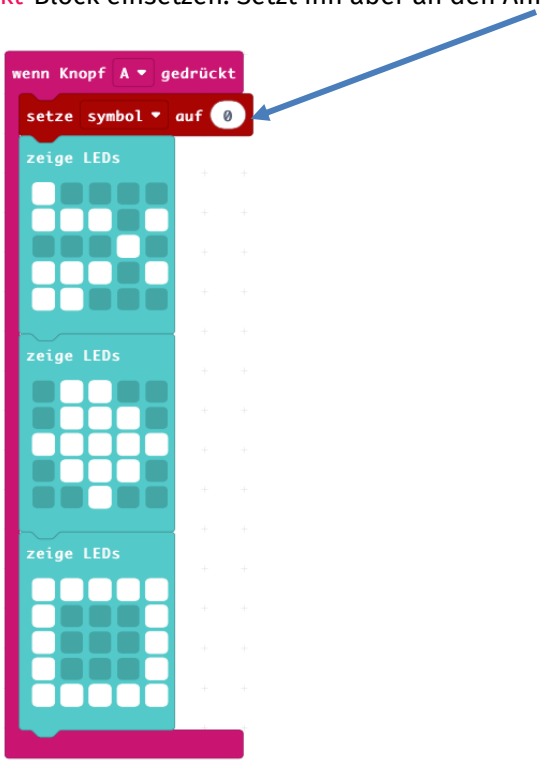
Variable

Im Programm benutzt ihr eine sogenannte Variable. In dieser Variablen speichert der Calliope mini die Zahlen zu den Symbolen (siehe Tabelle).

Eine Variable könnt ihr euch in etwa wie eine große **Kiste** vorstellen, in der etwas aufbewahrt wird – in diesem Fall die Zahlen zu den Symbolen. Variablen erhalten einen Namen, damit man sie auseinanderhalten und ansprechen kann.



Die Variable müsst ihr nun in euer Programm einbauen. Da die Variable erst relevant wird, wenn ihr Knopf A drückt, um ein Symbol aufzurufen, könnt ihr den entsprechenden Befehl in den wenn-Knopf-A-gedrückt-Block einsetzen. Setzt ihn aber an den Anfang.



Projekt: Schnick, Schnack, Schnuck

Als Nächstes werdet ihr das Symbol der Schere der **0** zuweisen.

Wenn die Variable gleich 0 ist,
dann soll die Schere angezeigt werden.
Ansonsten soll eines der anderen Symbole angezeigt werden.

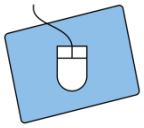
Um dies im Programm verwirklichen zu können, sind die folgenden Blöcke nötig:

Wenn eine bestimmte **Bedingung** wahr ist,
dann wird der Befehl ausgeführt, der in der **Lücke** steht.
Falls nicht, geht es weiter mit den Befehlen, die in der **Lücke** hinter ansonsten stehen.

Diesen Block braucht ihr, um überprüfen zu können, ob die Variable = 0 ist.

Diesen Block braucht ihr für die Variable symbol.

Der Block für die Darstellung der Schere ist schon in eurem Programm, erhält nun jedoch einen neuen Platz.



Kombiniert die Blöcke, um abzufragen, ob die Variable symbol = **0** ist. Falls die Bedingung wahr ist, soll die Schere angezeigt werden. Testet das Programm.

Bedingte Verzweigung
Der Wenn-dann-(ansonsten)-Block wird euch in der Programmierung häufiger begegnen.

Wenn die Variable für das Symbol gleich 0 ist,
dann soll die Schere angezeigt werden.

Oder allgemeiner:

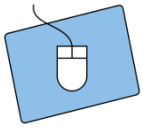
Wenn eine **Bedingung wahr** ist,
dann wird ein **Befehl ausgeführt**.

Deshalb spricht man in der Informatik auch von einer **bedingten Verzweigung**. Mit dem Wenn wird erst einmal abgefragt, ob eine bestimmte Bedingung erfüllt ist (hier: symbol = 0). Wenn dies der Fall ist, die Bedingung also wahr ist, dann wird der Befehl hinter dann ausgeführt. Falls nicht, werden die Befehle ausgeführt, die hinter ansonsten stehen.

Projekt: Schnick, Schnack, Schnuck

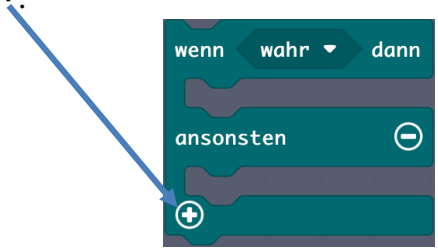
Wenn ihr euer Programm jetzt testet, wird euch beim Drücken von Knopf A immer die Schere angezeigt. Damit auch eines der anderen Symbole erscheinen kann, fehlen noch zwei Schritte.

1. Ihr müsst bedingte Verzweigungen für die anderen beiden Symbole (Stein und Papier) anlegen.
2. Ihr müsst dafür sorgen, dass nicht immer nur die Variable mit dem Wert **0** aufgerufen wird, sondern auch die mit dem Wert **1** (Stein) und **2** (Papier) zufällig aufgerufen werden können.

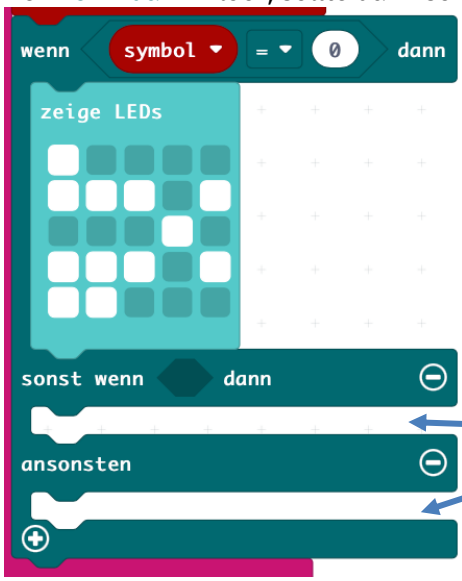


Beginnt mit den bedingten Verzweigungen für die anderen beiden Symbole.

1. Erweitert den wenn-dann-Block, damit nicht nur eine, sondern **zwei weitere bedingte Verzweigungen** verwirklicht werden können. Klickt dazu im wenn-dann-Block auf das **+**.



Der wenn-dann-Block, sollte dann so aussehen:



Hier ist nun Platz für die Bedingungen zu den anderen beiden Symbolen.

2. Versucht, die Bedingungen für die Symbole Stein und Papier zu programmieren. Das funktioniert im Wesentlichen genauso wie bei der Schere.

Tipp: In der Lücke hinter ansonsten braucht ihr zusätzlich noch einen 

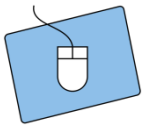
-Block.

3. Testet euer Programm.

Projekt: Schnick, Schnack, Schnuck

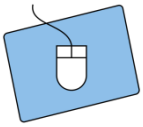
Euer Programm sollte jetzt ein zufälliges Symbol (Schere, Stein, Papier) anzeigen, wenn ihr Knopf A drückt. Klappt das? Falls ja, ist das super! Sollte noch nicht alles reibungslos funktionieren, bittet einen Betreuer oder eine Betreuerin um Hilfe, bevor ihr fortfahrt.

Auf geht's zum Endspurt. Eine Runde *Schnick, Schnack, Schnuck* sollte man auch beenden können. Der Calliope mini soll schließlich nicht ewig ein zufälliges Symbol anzeigen.



Programmiert daher ein Ende der Runde: Wenn ihr Knopf B drückt, dann soll die Anzeige des LED-Bildschirms gelöscht werden.

Funktionieren die Knöpfe A und B wie sie sollen? Sehr gut, denn dann ist euer Schnick-Schnack-Schnuck-Spiel fertig programmiert.



Es ist an der Zeit, das *Schnick-Schnack-Schnuck*-Spiel ausgiebig zu testen. Spielt ein paar Runden, und notiert die Ergebnisse in der Tabelle, indem ihr ein Kreuz beim Gewinner oder der Gewinnerin setzt und eine 0 beim Verlierer oder der Verliererin. Wer hat am Ende gewonnen – der Calliope mini oder ihr? Notiert den Endstand.

Runde	Spieler*in	Calliope mini
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
Endstand:		

Quellenverzeichnis:
Abbildungen der Hände (Schere, Stein, Papier) – *Quelle: InfoSphere*
Programmierblöcke – *Quelle: Screenshots des MakeCode-Editors (https://makecode.calliope.cc)*
 angefertigt vom InfoSphere-Team