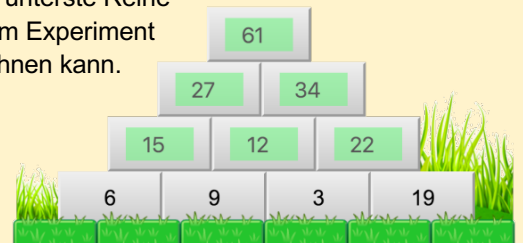


Computer rechnen sehr schnell

Eine grosse Stärke des Computers ist seine Fähigkeit sehr sehr schnell zu rechnen. Vielleicht hast du Schule bereits Zahlenmauern kennengelernt. Jeder Stein der Mauer steht für eine Zahl. Die Zahl jedes Steins wird aus der Summe der beiden darunterliegenden Steine gebildet. In der einfachsten Variante ist die unterste Reihe vorgegeben und die restlichen Zahlen sollen ausgerechnet werden. In diesem Experiment wollen wir ausprobieren, wie schnell der Computer eine Zahlenmauer berechnen kann.

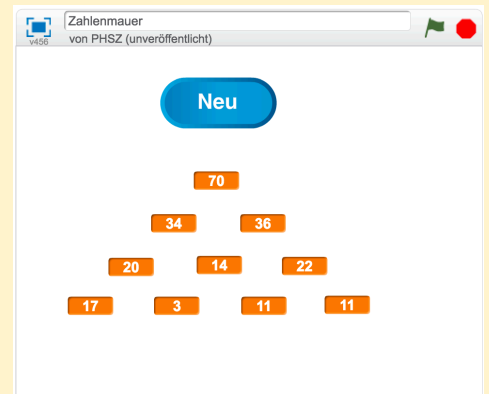
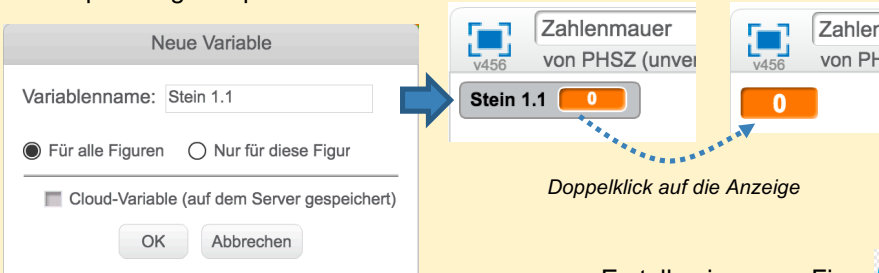
Aus Sicht der Mathematik können wir eine Zahlenmauer als Folge von einfachen Gleichungen beschreiben, unabhängig davon, welche Zahlen konkret darin stehen (Stein_{2,1} ist z.B. der erste Stein in der zweiten Reihe):

$$\begin{aligned} \text{Stein}_{2,1} &= \text{Stein}_{1,1} + \text{Stein}_{1,2} & \text{Stein}_{3,1} &= \text{Stein}_{2,1} + \text{Stein}_{2,2} \\ \text{Stein}_{2,2} &= \text{Stein}_{1,2} + \text{Stein}_{1,3} & \text{Stein}_{3,2} &= \text{Stein}_{2,2} + \text{Stein}_{2,3} \\ \text{Stein}_{2,3} &= \text{Stein}_{1,3} + \text{Stein}_{1,4} & \text{Stein}_{4,1} &= \text{Stein}_{3,1} + \text{Stein}_{3,2} \end{aligned}$$



Eine Welt aus Variablen

In Scratch definieren wir für jeden Stein eine Variable. Wechsel in der Blockpalette auf „Daten“ und klicke auf **Neue Variable**, um eine neue Variable zu erstellen. Erstelle insgesamt 10 Variablen mit passenden Namen für die einzelnen Steine. Diese erscheinen automatisch oben links auf der Bühne. Verändere die Darstellung zu einem Block und ordne die Variablen auf der Bühne per Drag&Drop als Mauer an.



Wenn ich angeklickt werde

- setze Stein 1.1 auf Zufallszahl von 1 bis 20
- setze Stein 1.2 auf Zufallszahl von 1 bis 20
- setze Stein 1.3 auf Zufallszahl von 1 bis 20
- setze Stein 1.4 auf Zufallszahl von 1 bis 20
- setze Stein 2.1 auf Stein 1.1 + Stein 1.2
- setze Stein 2.2 auf Stein 1.2 + Stein 1.3
- setze Stein 2.3 auf Stein 1.3 + Stein 1.4
- setze Stein 3.1 auf Stein 2.1 + Stein 2.2
- setze Stein 3.2 auf Stein 2.2 + Stein 2.3
- setze Stein 4.1 auf Stein 3.1 + Stein 3.2

Erstelle eine neue Figur **Neu**, welche beim Anklicken eine neue zufällige Zahlenmauer berechnet. Erstelle dazu ein Skript, welches die Variablen der unteren 4 Steine der Mauer mit einer Zufallszahl befüllt und alle weiteren Steine aus der Summe der beiden vorherigen berechnet (siehe Skript links). Wenn du alles richtig gemacht hast, sollte eine fertig gelöste Zahlenmauer entstehen. Sollte etwas nicht stimmen, prüfe nochmals das Skript und die Anordnung der Variablen auf der Bühne.

Das geht aber schnell

Für Aufgabenblätter wäre es schön Zahlenmauern mit einer bestimmten Zahl im Deckstein zu erzeugen. Erstelle eine weiteren Schaltfläche, welche Zahlenmauern mit 100 erzeugt. Zähle die Versuche, die der Computer unternommen hat, bis zufällig eine passende Mauer entstanden ist.

Wenn ich angeklickt werde

- setze Versuche auf 0
- setze Stein 4.1 auf 0
- wiederhole bis Stein 4.1 = 100
 - ... alles vom Skript links ...
 - ändere Versuche um 1
- sage verbinde Versuche: Versuche

Zusatz

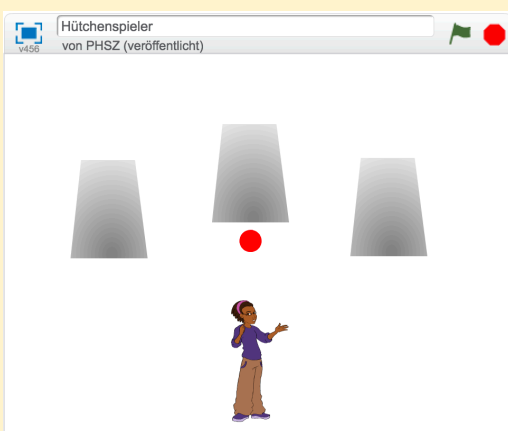
Verändere das zweite Skript so, dass Zahlenmauern mit 100'000 im Deckstein entstehen. Teste das Programm. Welche Anpassungen musst du zusätzlich vornehmen, damit eine Zahlenmauer mit 100'000 entstehen kann? Kannst du die Berechnungszeit messen? Verwende **setze Stoppuhr zurück** und **verbinde Stoppuhr Sekunden**.

Sich absprechen

Ein Computerprogramm besteht in der Regel aus vielen verschiedenen Teilen. Bei Scratch besitzt jede Figur ein oder mehrere eigene Skripte, welche grundsätzlich unabhängig von den anderen Figuren ablaufen. In der Informatik spricht man auch von paralleler Verarbeitung. Manchmal ist es nötig, dass sich Teilprogramme miteinander absprechen. Auf deinem Computer oder Smartphone müssen sich manchmal auch die Programme und Apps untereinander absprechen. Das Betriebssystem entscheidet zum Beispiel wann welches Programm arbeiten darf.

Das Hütchenspiel

Das folgende Beispiel verwendet 3 Becher, um das klassische Hütchenspiel nachzubauen. Bei diesem Spiel wird unter einen Becher eine Kugel gelegt und die Becher mehrmals schnell vertauscht. Der Spieler muss nun raten, unter welchem Becher die Kugel liegt. Wie im richtigen Spiel brauchen wir noch einen Hütchenspieler, der die Becher schnell verschiebt und damit die 3 Becher „koordiniert“. In Scratch kann der Hütchenspieler den Bechern eine Nachricht schicken, die darauf hin ihre Plätze tauschen.

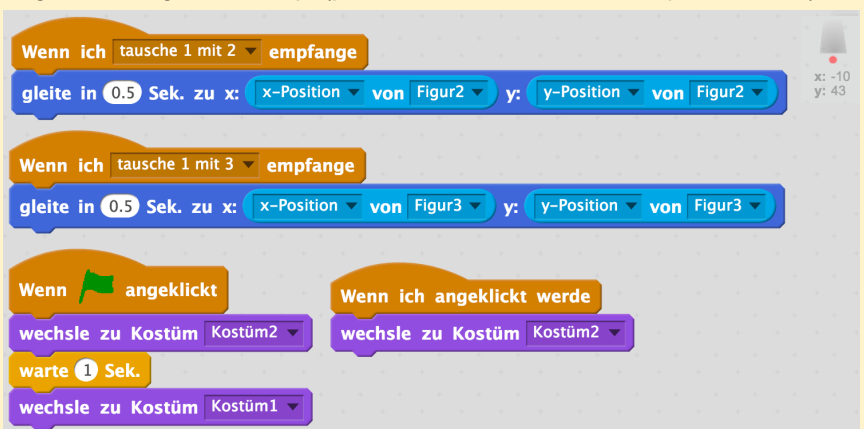


Erstelle ein neues Scratch-Projekt. Zeichne eine neue Figur für den ersten Becher (verwende den Vektor-Modus). Dupliziere das Kostüm und zeichne zusätzlich eine Kugel auf den unteren Teil des Bechers. Verschiebe dann den Becher über die Kugel. Beim Wechsel der Kostüme sollte der Becher scheinbar angehoben werden. Dupliziere die Figur 2 mal, sodass du insgesamt 3 Becher hast. Lösche bei den anderen beiden Bechern die Kugel im Kostüm2, sodass nichts darunter liegt. Ordne die Becher nebeneinander auf der Bühne an. Erstelle eine weitere Figur für den Hütchenspieler.



Hütchenspieler programmieren

Der Hütchenspieler soll die Becher einige Male tauschen. Im Skript links wird 10 Mal getauscht. In jedem Schritt kann er sich entscheiden, welche Becher er miteinander tauscht. Es gibt 3 Möglichkeiten: 1 mit 2, 1 mit 3, oder 2 mit 3. Damit das Spiel unvorhersehbar bleibt verwenden wir eine Zufallszahl. Erstelle eine Variable Z, um die gewürfelte Zahl zwischenspeichern. Je nachdem verschicken wir Nachrichten an alle anderen Figuren des Programms und damit an die Bechern. Jeder Becher wartet auf jeweils die zwei Nachrichten, die ihn betreffen. Für Becher1 ergibt sich folgendes Skript (passe bei Becher2 und 3 entsprechend an):



Zusatz

Kannst du das Programm so erweitern, dass der Spieler in jeder Runde automatisch einen Geldbetrag setzt und beim Anklicken des richtigen Bechers den doppelten Einsatz dazu gewinnt?

Warum Programme aufteilen?

In der Informatik ist „Abstraktion“ ein sehr zentrales Thema. Dabei geht es immer darum Details wegzulassen und ein allgemeineres Konzept zu beschreiben. Das machen wir auch in unserer Sprache. Zum Beispiel der Begriff „Haus“ meint kein bestimmtes Gebäude sondern ein Konzept mit Dach, Fenstern, Türen und Zimmern wobei jedes Haus in der Welt etwas anderes aussieht aber gewisse Eigenschaften überall vorhanden sind.

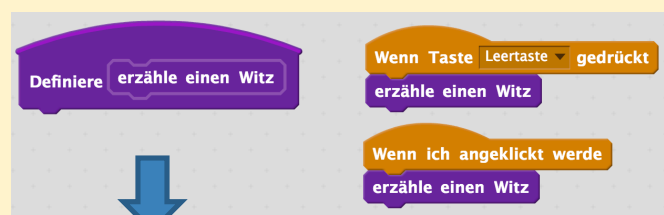
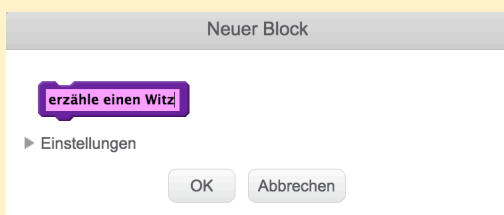
Auch beim Programmieren wenden Entwickler Abstraktion an, um einzelne Teile eines Programms unabhängig vom Rest zu entwickeln. Idealerweise kann dieser Teil dann auch in verschiedenen Programmen, oder zumindest an verschiedenen Stellen im gleichen Programm, wiederverwendet werden. Arbeiten mehrere Entwickler an einem grösseren Projekt zusammen hilft dieses Prinzip auch sich abzustimmen. Solange ein Teilprogramm sich nach den Vorgaben verhält, d.h. bestimmte Eingabe (Parameter) entgegennimmt und bestimmte Ausgaben liefert, spielt es keine Rolle wie das Teilprogramm erstellt wurde oder genau arbeitet (man spricht auch vom „Blackbox“-Prinzip).

Unterprogramme in Scratch

Jeder Befehlsblock in Scratch wie **gehe 10 er-Schritt** oder **spiele Klang meow** wird durch ein Unterprogramme beschrieben. Die genaue Definition (oder Implementation) was bei einem solchen Befehl passiert, zeigt uns Scratch aber nicht an. Solange die Figur einen Schritt läuft, müssen uns die Details aber auch nicht kümmern.

Scratch bietet aber die Möglichkeit an, selbst neue Befehlsblöcke zu erstellen. In der Blockpalette unter „weitere Blöcke“ klicken wir auf **Neuer Block**, um selbst ein Unterprogramm für die ausgewählte Figur oder die Bühne zu erstellen. Unterprogramme sind immer nur für die Figur nutzbar, in der sie erstellt wurden.

Wir möchten einen neuen Block **erzähle einen Witz** erfinden, den wir beim Drücken der Leertaste und beim Anklicken der Figur verwenden wollen. Erstelle einen neuen Block und gib den Namen in das Eingabefeld im Dialog ein:



In dem grossen, lilanen, Block müssen wir nun definieren, was passieren soll, wenn dieser verwendet wird. Damit es nicht so schnell langweilig wird, soll die Figur jedes Mal einen zufälligen Witz erzählen. Dazu erstellen wir eine Variable **Witz** (in der Blockpalette unter Daten). Erstelle ein Skript wie Rechts gezeigt und teste dein Programm. Du kannst dir natürlich eigene Witze oder Sprüche ausdenken.



Zusatz


Erweitere dein Programm so, dass beim Drücken der Taste „w“ gleich 3 Witze nacheinander erzählt werden.

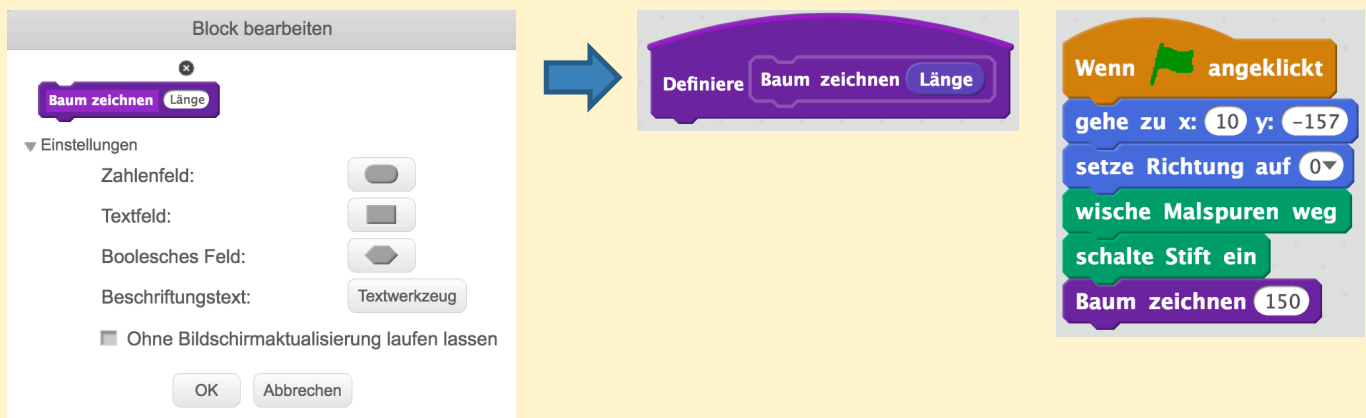
Nach jedem Witz soll die Figur noch mit **spiele Klang laugh-female** lachen. Wo musst du diesen Block einfügen?

Kannst du durch das Programm so verändern, dass garantiert niemals der gleiche Witz hintereinander erzählt wird?

Wiederholung mit Rekursion

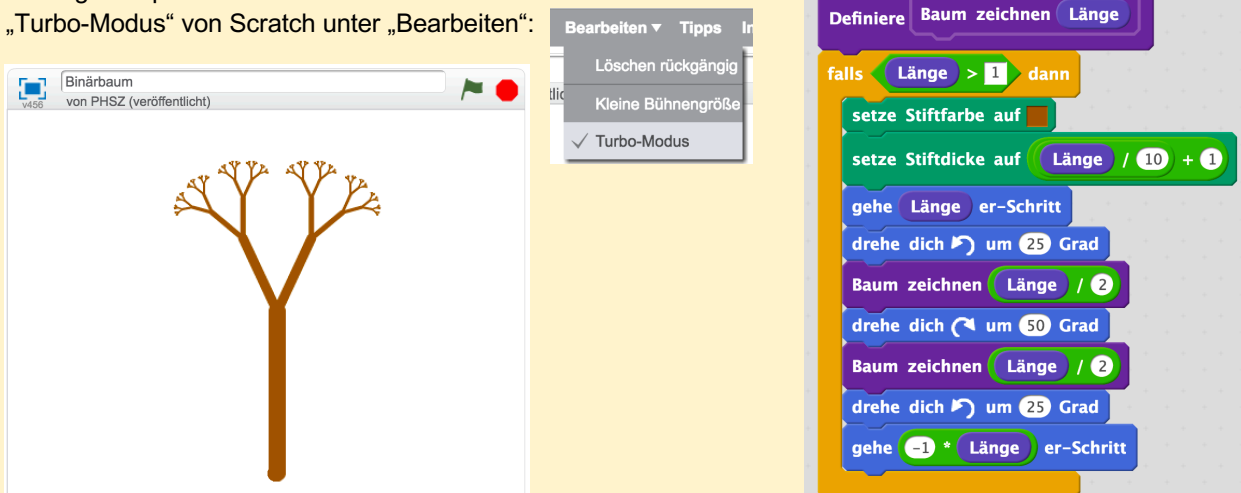
Mit Hilfe von eigenen Blöcken lassen sich Unterprogramme in Scratch abbilden. Diese Unterprogramme können sich auch selbst aufrufen, wodurch rekursive Programme möglich werden. Damit lässt sich eine Reihe von informatischen Problemen elegant lösen, jedoch ist das Konzept der Rekursion schwer zu verstehen und anzuwenden. Das nachfolgende Beispiel zum Zeichnen eines Baumes zu verstehen ist somit eine recht anspruchsvolle Aufgabe.

Erstelle ein neues Scratch-Programm. Erstelle eine neue Figur mit der Vorlage „Pencil“ und lösche die Katze aus dem Projekt. Wähle die Figur aus und klicke auf **Neuer Block** in der Blockpalette unter „weitere Blöcke“. Gib als Namen „Baum zeichnen“ in das lila Kästchen ein. Klicke auf „Einstellungen“ im Dialog, um die Parameter für den neuen Block auszuwählen. Klicke einmal auf , um ein neues Zahlenfeld hinzuzufügen. Benenne dieses mit „Länge“ und bestätige den Dialog mit OK. Erstelle das rechts abgebildete Skript, um den neuen Block mit einer Länge von 150 zu verwenden.



Der Block „Baum zeichnen“

Wir möchten einen „Binärbaum“ wie im Bild unten zeichnen. Ein Binärbaum hat an jedem Ast genau eine Verzweigung in zwei weitere Äste – daher der Name Binärbaum. Zudem ist jeder Ast genau halb so lang wie der vorherige. Der Stamm ist somit der erste Ast. Bei einer Rekursion ist es wichtig eine „Abbruchbedingung“ zu haben, bei der das immer wieder selbst Aufrufen beendet wird. Im Falle des Baums zeichnen wir nur weiter, wenn der nächste Ast noch länger als 1 ist. Erstelle das Skript rechts für „Baum zeichnen“ und überlege was dabei genau passiert. Zum schneller Zeichnen aktiviere den „Turbo-Modus“ von Scratch unter „Bearbeiten“:



Zusatz

Versuche die Zahlen bei „drehe“ von 25 – 50 – 25 zum Beispiel auf 45 – 90 – 45 zu ändern.

Was passiert wenn du statt $\text{Länge} / 2$ z.B. $\text{Länge} / 1.5$ verwendest? Achtung: Scratch kann nicht über den Rand zeichnen, sollte dein Baum verschoben erscheinen, versuchen eine kleinere Zahl bei **Baum zeichnen 150**.