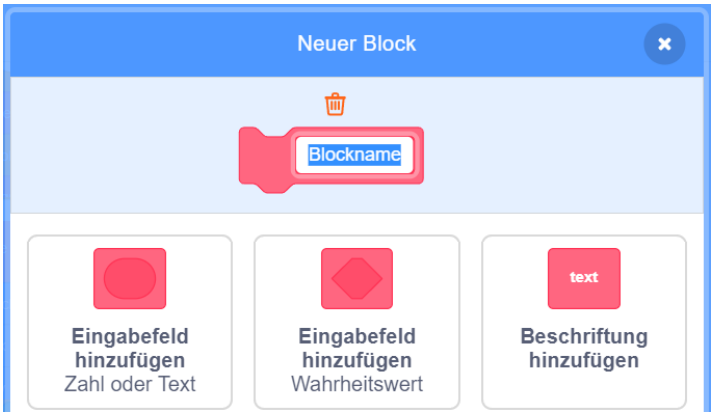
Skripte strukturieren

In Scratch gibt es die Möglichkeit, **eigene Blöcke** zu definieren. Man kann sie benutzen, um den Code besser zu strukturieren: Wenn Skripte sehr lang werden, definiert man Teile davon als eigene Blöcke, und baut die dann in das Skript ein, statt ein langes Skript zu programmieren.

Im abgebildeten Beispiel besteht das Bewegungsskript einer Figur aus Anfangseinstellungen, Bewegung und der Behandlung von Kollisionen. Für diese Bestandteile wurden hier eigene Blöcke definiert, die dann in das Skript eingesetzt wurden. Wenn einer der rosa Blöcke im Skript ausgeführt wird, werden alle Blöcke, die zur Definition des Blocks gehören, ausgeführt.

Einen eigenen Block kann man unter „Meine Blöcke“ definieren. Im folgenden Dialog gibt man dem Block einen **Namen**. Dieser sollte möglichst präzise beschreiben, was der Block tut.

Darunter kann man **Parameter** („Eingabefelder“) angeben, was wir an dieser Stelle aber nicht brauchen.



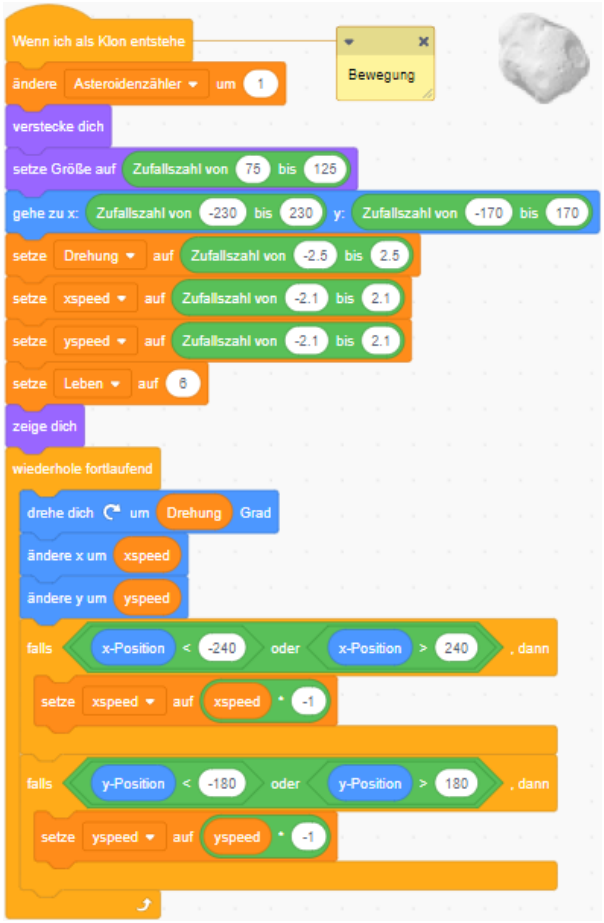
Aufgabe 1

Bearbeite die ausgeteilte Vorlage zum Spiel Asteroids (vom Abschnitt 12, ohne Nachrichten).

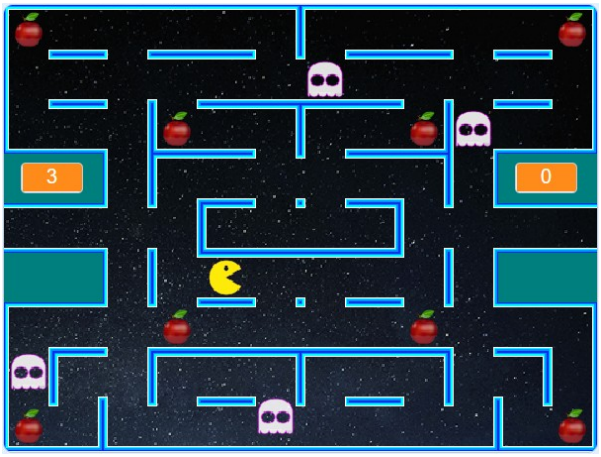
Für die Figur „Asteroid“ definiere eigene Blöcke, um das Bewegungsskript „aufzuräumen“:

- Einen Block für die Anfangseinstellungen
- Einen Block für die Bewegung
- Einen Block für das Abprallen vom Rand (das man hier selbst programmieren muss)

Anschließend baue diese selbst definierten Blöcke in das Bewegungsskript ein.



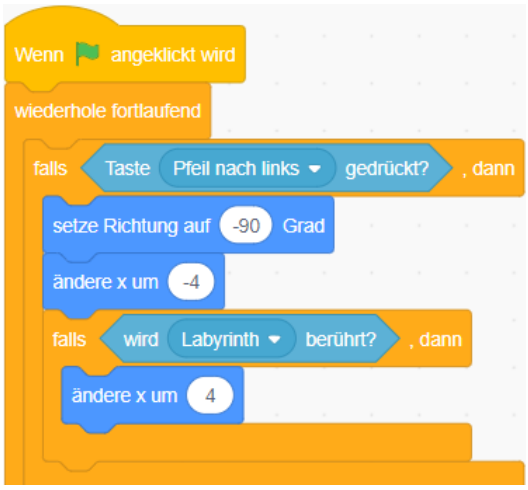
Pac-Man



Das Spiel Pac-Man wurde 1980 von der japanischen Firma Namco, unter Federführung des Designers Toru Iwatani entwickelt. Auch wenn es einfach aussieht, ist es kommerziell eines der erfolgreichsten Spiel aller Zeiten: Seit seiner Veröffentlichung hat es ca. 18 Milliarden Dollar Umsatz (inflationsbereinigt, Stand 2022) erzielt.

Das Spielprinzip ist vermutlich bekannt:
Pac-Man bewegt sich durch ein Labyrinth, um die darin liegenden Gegenstände aufzusammeln. Dabei muss er den Geistern ausweichen, die die Gänge des Labyrinths patrouillieren.

Kollision mit Hitbox



Die Hauptaufgabe bei der Programmierung von Pac-Man ist die Bewegung der Figuren: sie dürfen sich frei im Labyrinth bewegen, dabei aber nicht die Wände der Gänge überschreiten. Wie beim Spiel „Tunnelflug“ ist das Labyrinth hier eine Figur von der Größe der Bühne, bei der die Gänge durchsichtig sind.

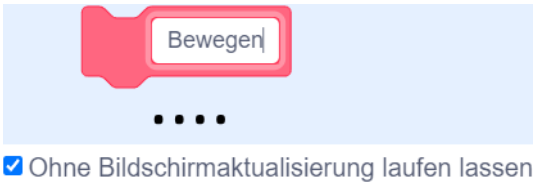
Rechts abgebildet ist ein Skript, mit dem Pac-Man sich mit der Tastatur nach links bewegt. Wenn er eine Wand berührt, wird seine Bewegung einfach rückgängig gemacht. So stoppt die Wand seine Bewegung. Für die anderen drei Richtungen funktioniert das Skript analog.

Ein Problem ist aber, dass Pac-Man **rund** ist. Dadurch bleibt er an den Ecken der Wände leicht hängen. Das Spiel würde zwar irgendwie funktionieren, aber das Hängenbleiben stört den Ablauf und wirkt nicht „natürlich“.

Mit einem rechteckigen bzw. quadratischen Kostüm könnte er sich „glatt“ durch die Gänge und um Ecken bewegen, aber Pac-Man ist eben rund! Man löst das Problem auf folgende Weise:

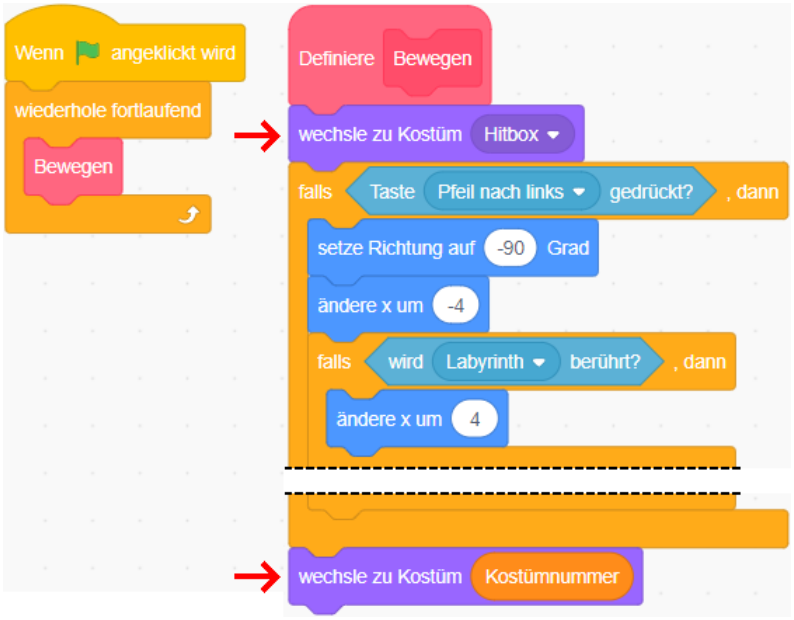
- Die Figur Pac-Man erhält ein zusätzliches, quadratisches Kostüm. Da die Kollision mit diesem Kostüm geprüft wird, nennt man dieses auch „**Hitbox**“
- Vor jeder Bewegung wechselt das Skript erst zum Hitbox-Kostüm. Es wird versucht, Pac-Man zu bewegen, und bei Kollision mit einer Wand wird die Bewegung rückgängig gemacht.
- Nachdem die neuen Koordinaten von Pac-Man bestimmt sind, wechselt das Skript zum passenden, runden Kostüm.
- Der Wechsel der Kostüme geschieht „unsichtbar“, d.h. als Spieler*in bekommt man davon nichts mit – man sieht immer nur die runden Pac-Man-Kostüme.

Block ohne Bildschirmaktualisierung



Den unsichtbaren Kostümwechsel zur „Hitbox“ kann man (nur) über einen selbst definierten Block erreichen. Wenn man z.B. einen Block namens „Bewegen“ definiert, aktiviert man die Option „Ohne Bildschirmaktualisierung laufen lassen“. Das bedeutet, dass der nächste Frame in Scratch erst gezeichnet wird, nachdem der Block vollständig abgearbeitet wurde. Kurzzeitige Kostümwechsel oder andere Veränderungen im Ablauf werden also nicht gezeichnet, nur der Zustand am Ende des Blocks.

Im abgebildeten Skript wird der Block „Bewegen“ laufend wiederholt. In jeder Wiederholung wechselt Pac-Man (unsichtbar) zum quadratischen Kostüm der Hitbox, führt die Bewegung entsprechend der Benutzereingaben und des Labyrinths durch, und wechselt am Ende des Blocks zum passenden runden Kostüm.



Um Pac-Man mit den drei runden Kostümen animieren zu können, wird das passende Kostüm hier mithilfe einer Variable dargestellt, die von einem anderen Block berechnet wird.

Aufgabe 2

Verwende die ausgeteilte Scratch-Vorlage. Sie enthält die Figuren für das Labyrinth, Pac-Man und die Geister sowie für die Äpfel. Du brauchst hier nur Pac-Man zu programmieren, die anderen Figuren sind bereits fertig.

Für Pac-Man sind vier eigene Blöcke zu definieren und in die Haupt-Wiederholschleife einzufügen:

a) **Kostümwahl**
Das Animationsskript kann hier nicht parallel neben dem Bewegungsskript laufen, da dieses ja zum Hitbox-Kostüm wechseln muss. Stattdessen berechnet der Block „Kostümwahl“ die Nummer des aktuellen Kostüms (eine Zahl zwischen 1 und 3) und speichert sie in einer Variablen. Diese wird dann vom nächsten Block „Bewegen“ für den Kostümwechsel eingesetzt. Überlege, wie du erreichen kannst, dass Pac-Man nicht in jedem Frame sein Kostüm wechselt, denn dann bewegt er seinen Mund zu schnell.

b) **Bewegen**
Nach der Anleitung von Seite 3 und 4 wechselt Pac-Man erst zum Hitbox-Kostüm. Dann wird die Bewegung entsprechend der vier Pfeiltasten und des Labyrinths berechnet. Anschließend wechselt er zum aktuell passenden Kostüm, das durch die Variable „Kostümnummer“ festgehalten wird. Wegen des Hitbox-Kostüms muss dieser Block „ohne Bildschirmaktualisierung laufen“.

c) **Teleport**
Das Labyrinth hat zwei Öffnungen, mittig am linken und rechten Rand. Wenn sie hier hineinlaufen, werden Pac-Man und die Geister zur Öffnung auf der gegenüberliegenden Seite teleportiert.

d) **Kollision**
Wenn Pac-Man einen der Geister berührt, verliert er ein Leben und, falls noch Leben übrig sind, beginnen er und die Geister wieder auf ihren Anfangspositionen.

Falls du Schwierigkeiten mit Teilen der Skripte hast, kannst du ggf. bei den Geistern „spicken“.

Der Algorithmus für die Bewegung der Geister ist relativ komplex, daher ist er in der Vorlage bereits fertig programmiert. Sie entscheiden sich an Abzweigungen im Labyrinth zufällig für eine der Wahlmöglichkeiten. Wenn du es selbst versuchen, oder ein anderes Verhalten programmieren möchtest, kannst du den Block „Bewegen“ verändern oder ihn löschen und neu programmieren.

Das Spiel ist an dieser Stelle einfach gehalten: Es gibt nur ein Level und acht Äpfel zum Aufsammeln. Wenn du Lust hast, erweitere das Projekt, z.B. um mehr Gegenstände zum Aufsammeln oder neue Level. Die Kacheln des Labyrinths findest du im Material, mit einem Grafikprogramm kannst du daraus neue Level erstellen.