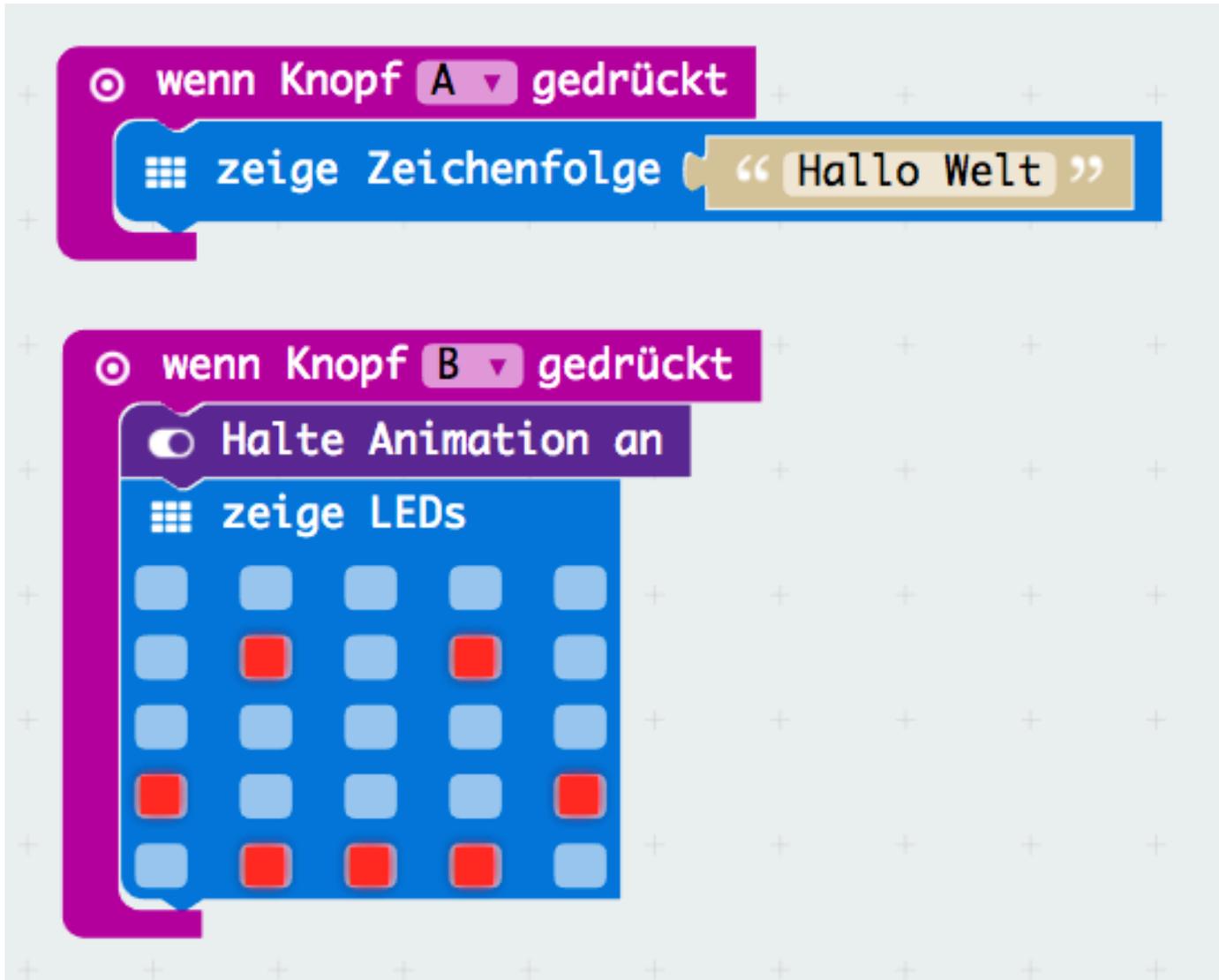


Lösung Task I: Der Welt Hallo Sagen



TASK

- › Nach Knopfdruck Displayanzeige aktivieren / wechseln

INHALTE

- › Einstieg in die PXT Programmumgebung
- › Auf Ereignisse reagieren

Benötigte Blöcke

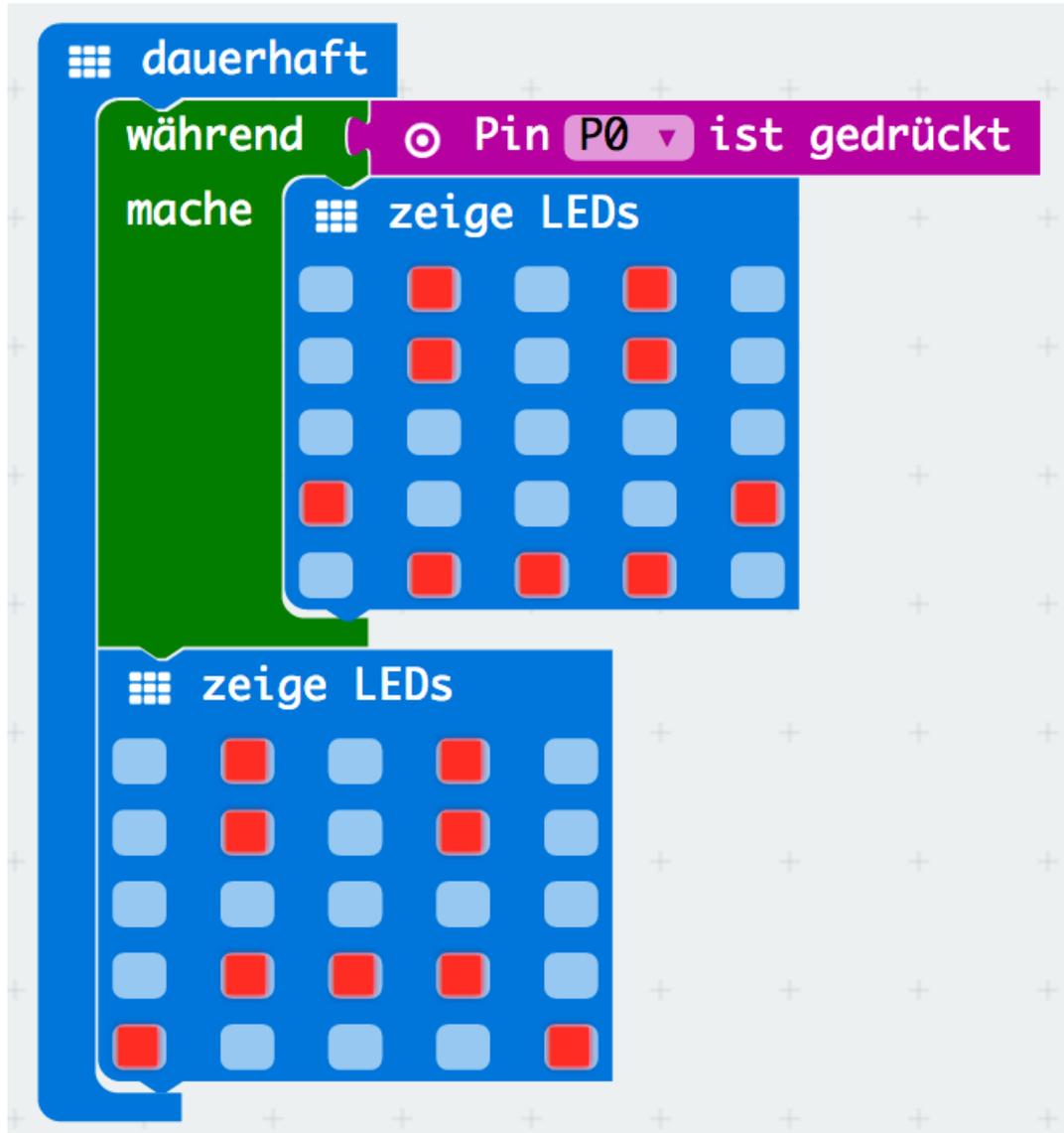
zeige Zeichenfolge []

zeige LEDs

wenn Knopf [] gedrückt

halte Animation an

Lösung TASK II: Pin-Kontakt



TASK

- › Durch den Pinkontakt ändert sich die Displayanzeige von einem Anti-Smiley zu einem Smiley

INHALTE

- › Nutzung der analogen Pins
- › Einsatz einer Schleife

Benötigte Blöcke

- dauerhaft []
- zeige LEDs
- pin [] ist gedrückt
- während [] mache []

Lösung TASK III: Zähler

```
beim Start
  ändere Nummer auf 0

dauerhaft
  zeige Nummer Nummer

wenn Knopf A gedrückt
  wenn (Nummer > 0)
  dann
    ändere Nummer auf (Nummer - 1)

wenn Knopf B gedrückt
  wenn (Nummer < 10)
  dann
    ändere Nummer um 1

wenn geschüttelt
  ändere Nummer auf (wähle eine zufällige Zahl zwischen 0 und 9)
```

TASK

- › Eine Variable mit dem Namen «Nummer» wird generiert (Platzhalter → neuen Platzhalter)
- › Je nach Knopfdruck wird der Wert der Variabel erhöht oder reduziert
- › Ein «Shake» definiert die Variable zufällig neu (1–9)

INHALTE

- › Variablen einsetzen und deren Wert verändern
- › Einsatz von «Wenn – Dann» Bedingungen
- › Fehlerkorrektur: Eingaben vor der Verarbeitung validieren

Benötigte Blöcke

beim Start dauerhaft

wenn Knopf [] gedrückt wenn [geschüttelt]

wenn [] dann

ändere [Platzhalter] auf []

[] + [] [] - [] wähle eine zufällige Zahl...

Lösung TASK IV: Kompass

```
Scratch code for a compass task. It starts with a 'dauerhaft' loop containing an 'ändere grad auf' block with 'Kompassausrichtung (°)' as the value. This is followed by a 'wenn' block with a condition: '(grad >= 315 und grad <= 360) oder grad < 45'. Inside the 'wenn' block, there are four 'dann/ansonsten wenn' branches: 1) 'dann zeige Zeichenfolge "N"', 2) 'ansonsten wenn grad < 135 dann zeige Zeichenfolge "0"', 3) 'ansonsten wenn grad < 225 dann zeige Zeichenfolge "S"', and 4) 'ansonsten zeige Zeichenfolge "W"'.
```

TASK

- › Je nach Ausrichtung der Platine wird die Himmelsrichtung angezeigt
- › Die Ausrichtung wird in Grad gemessen und wird in einer Variable gespeichert
- › 0 Grad entspricht Norden. «N» soll entsprechend zwischen 315° und 360° sowie zwischen 0° und 45° angezeigt werden

INHALTE

- › Variablen einsetzen und deren Wert durch einen Sensorwert bestimmen
- › Einsatz von «Wenn – Dann – Ansonsten» Bedingungen
- › Einsatz von logischen Operatoren

Benötigte Blöcke

- dauerhaft
- zeige Zeichenfolge []
- Kompassausrichtung
- wenn [] dann [] ansonsten
- [] und []
- [] oder []
- ändere [Platzhalter] auf []

Lösung TASK V: Game Astroids

Um ein Astroid Game zu entwickeln gibt es verschiedene Lösungsmöglichkeiten. Nachfolgend wird ein möglicher Ansatz erläutert.

Schritt 1:

Als erstes empfiehlt es sich die notwendigen Variablen (Platzhalter) anzulegen:

5 Variablen für die Astroidpixel

1 Variable für den Spielerpixel

1 Variable für den Punktestand

1 Variable für den Spielstatus (läuft = true, gam over = false)



Lösung TASK V: Game Astroids

Schritt 2:

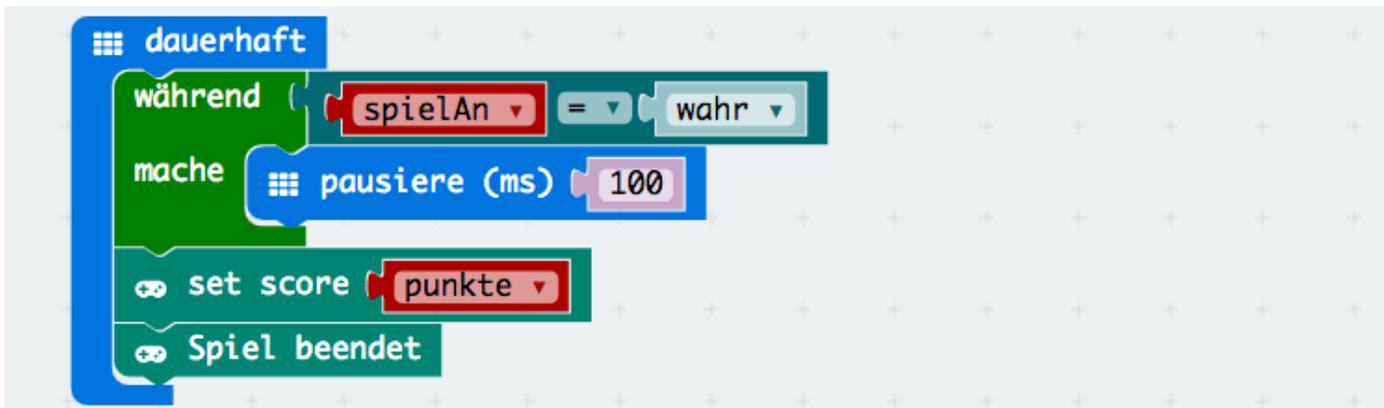
Als erstes wird beim Start das Spiel initiiert und die Variablen mit den nötigen Werten versehen.



```
beim Start
  ändere punkte auf 0
  ändere spieler auf erzeuge Sprite an Position x: 2 y: 4
  ändere spielAn auf wahr
```

Schritt 3:

Mit einer Schleife wird geprüft ob das Spiel läuft oder die Game-Over Sequenz gestartet wird.

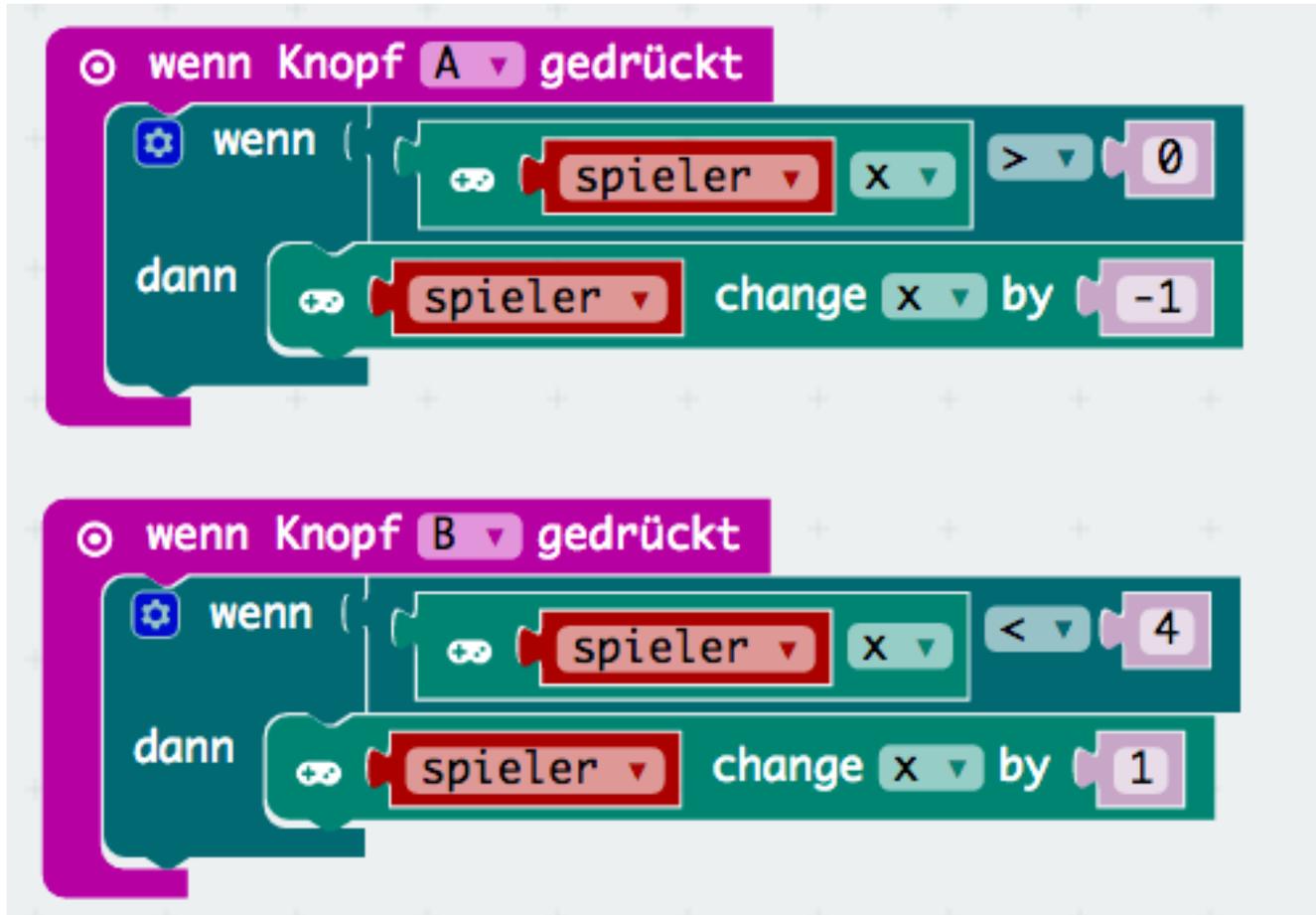


```
dauerhaft
  während (spielAn = wahr)
    mache
      pausiere (ms) 100
  set score punkte
  set score Spiel beendet
```

Lösung TASK V: Game Astroids

Schritt 4:

Nun wird die Spielsteuerung (Links-Rechts Bewegung) programmiert.



Lösung TASK V: Game Astroids

Schritt 5:

Nun folgt die Programmierung des ersten Astroiden (Spalte 0). Dieser wird zu Beginn an der Stelle X=0 und Y=0 zwischen 0 und 5000 Millisekunden «parkiert». Erst danach folgt der Fall (ansonsten würden alle Astroiden zur selben Zeit herunterfallen). Wenn der Astroid zuunterst ohne Kollision ankommt wird der Punktestand erhöht und die Schleife wird erneut abgearbeitet. Bei einer Kollision (spieler touching astroid0) wird die Spielstatus-Variable auf FALSE gesetzt. Dadurch verlässt das Programm die Schleife aus Schritt 3 und die Game-Over Sequenz wird gestartet.

```
Scratch code for spawning and moving an asteroid (astroid0):

1. Start with a 'dauerhaft' (forever) loop.
2. Inside the loop, a 'wenn (spielAn = wahr)' (if) block:
   - 'dann' (then) block:
     - 'ändere astroid0 auf (erzeuge Sprite an Position x: 0 y: 0)' (set astroid0 to create sprite at x: 0 y: 0)
     - 'pausiere (ms) wähle eine zufällige Zahl zwischen 0 und 5000' (wait 0-5000 ms)
   - 'während (spielAn = wahr)' (while) loop:
     - 'mache' (do) block:
       - 'wenn (astroid0 y = 4)' (if) block:
         - 'dann' (then) block:
           - 'wenn (spieler touching astroid0 ?)' (if) block:
             - 'dann' (then) block:
               - 'ändere spielAn auf falsch' (set spielAn to false)
             - 'ansonsten' (otherwise) block:
               - 'ändere punkte um 1' (add 1 to score)
               - 'astroid0 set y to 0' (set astroid0 y to 0)
               - 'pausiere (ms) wähle eine zufällige Zahl zwischen 0 und 5000' (wait 0-5000 ms)
         - 'ansonsten' (otherwise) block:
           - 'astroid0 change y by 1' (increase astroid0 y by 1)
           - 'pausiere (ms) 500' (wait 500 ms)
```

Dieser Schritt wird für die Astroiden 1 bis 4 mit den entsprechenden Astroid-Variablen wiederholt.