PICOBOY COLOR PLUS

Arbeitsblätter

Programmieren mit Blöcken

Klasse 5–7 · 23 Arbeitsblätter · pbcplus.de

Das Lehrerhandbuch gibt es kostenlos auf pbcplus.de/schule.
Musterlösungen und Programmdateien: E-Mail an info@picoboy.de.

Hinweis: Diese Arbeitsblätter wurden mit KI generiert.
Texte und Beispielprogramme stammen von einer künstlichen Intelligenz.
Die Programme wurden getestet – Fehler sind trotzdem möglich.
Rückmeldungen gern an info@picoboy.de.

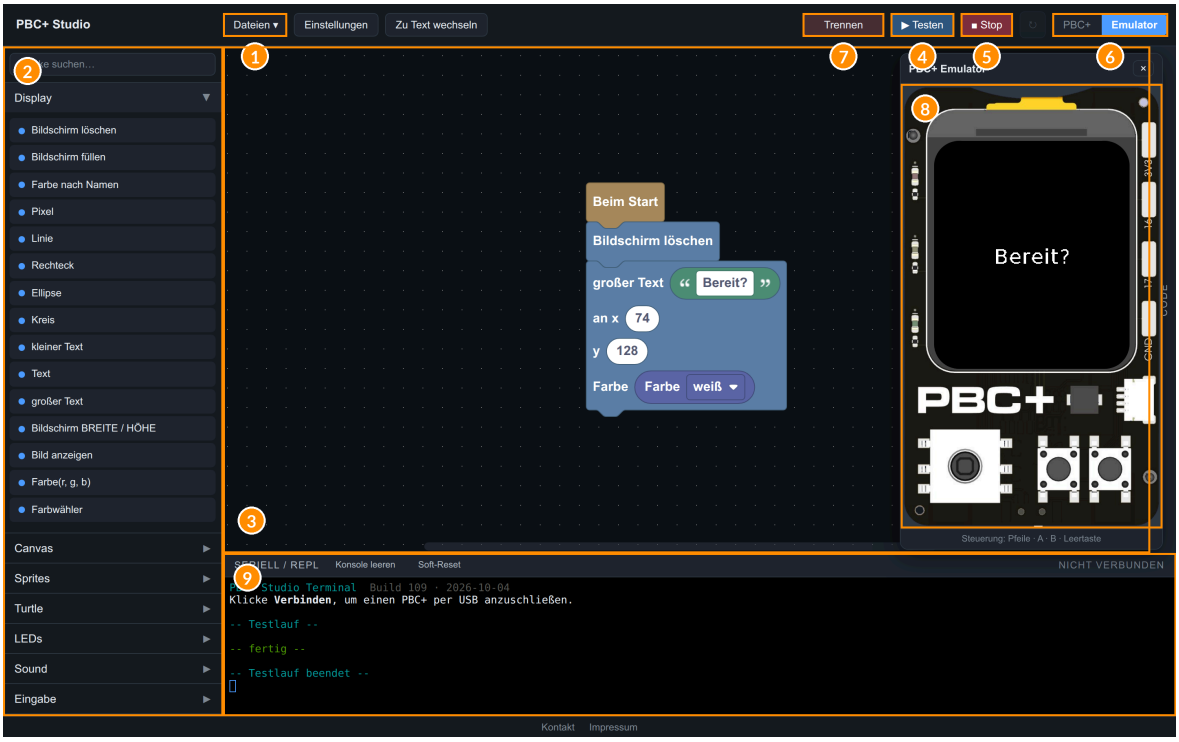
Name: _____ Klasse: _____ Datum: _____

PBC+ Studio kennenlernen

Ziel: Du lernst PBC+ Studio kennen, testest dein erstes Programm im Emulator, änderst es und speicherst es.

Öffne **pbcstudio.de** in Chrome oder Edge.

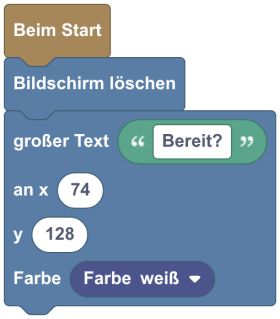
Aufgabe 1 ⭐ Wo ist was?



Schreibe zu jeder Nummer den passenden Namen. Diese Wörter helfen dir: *Arbeitsfläche* · *Blockleiste* · *Dateien* · *Emulator* · *Stop* · *Terminal* · *Testen* · *Verbinden* · *Ziel (Emulator oder PBC+)*

1 _____	4 _____	7 _____
2 _____	5 _____	8 _____
3 _____	6 _____	9 _____

Aufgabe 2 ⭐ Dein erstes Programm



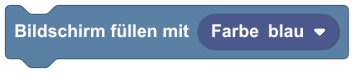
Dieses Programm liegt schon auf der Arbeitsfläche.

1. Wähle oben rechts **Emulator** und klicke auf ▶ **Testen**. Was erscheint im Emulator? _____
2. Klicke in das Feld mit „Bereit?“ und schreibe „Hallo,“ und deinen Namen hinein. Wähle bei **Farbe** eine andere Farbe. Teste wieder.

Tipp: Passt der Text nicht ganz auf den Bildschirm, mach die Zahl bei **an x** kleiner.

Aufgabe 3 ⭐ Bunter Hintergrund

1. Lösche den Block **„Bildschirm löschen“**: Klicke ihn an und drücke die Taste **Entf**. Der Text-Block rückt nach oben.
2. Hole aus der Kategorie **Display** diesen Block und hänge ihn direkt unter „Beim Start“:



3. Wähle eine Farbe und teste.



Aufgabe 4 ⭐ Speichern und öffnen

1. Speichere dein Programm: **Dateien → Auf den Computer…** In welchem Ordner liegt die Datei jetzt?

   _____
2. Klicke auf **Dateien → Neu**. Dein Programm ist weg!
3. Hol es zurück: **Dateien → Vom Computer…**

Aufgabe 5 ⭐⭐ Ein Ton zum Schluss

Suche in der Kategorie **Sound** diesen Block. Hänge ihn ganz unten an dein Programm und teste. Probiere andere Zahlen aus: Was ändert sich bei der ersten Zahl, was bei der zweiten?



_____

Aufgabe 6 ⭐⭐⭐ Auf das echte Gerät *(wenn ihr PicoBoys habt)*

Schließe den PicoBoy mit dem USB-Kabel an. Wähle als Ziel **PBC+**, klicke auf **Verbinden** und wähle den PicoBoy in der Liste aus. Teste dein Programm jetzt auf dem echten Gerät.

Das habe ich gelernt:

- ☐ Ich kenne die Teile von PBC+ Studio.
- ☐ Ich kann ein Programm im Emulator testen und anhalten.
- ☐ Ich kann Blöcke holen, verbinden, löschen und ihre Werte ändern.
- ☐ Ich kann ein Programm speichern und wieder öffnen.

Mehr dazu im Buch: Kapitel 1 – Der PicoBoy Color Plus und PBC+ Studio

Name: _____ Klasse: _____ Datum: _____

Hallo PBC+!

Ziel: Du schreibst Texte an genau die Stelle auf dem Bildschirm, die du willst. Dabei lernst du das Koordinatensystem des PBC+ kennen und die drei Text-Blöcke „kleiner Text“, „Text“ und „großer Text“.

Öffne PBC+ Studio, wähle oben rechts **Emulator** und klicke auf **Dateien → Neu**.

Aufgabe 1 ⭐ Hallo!



Baue dieses Programm. Den Block „großer Text“ findest du in **Display**. Teste mit ▶ **Testen**.

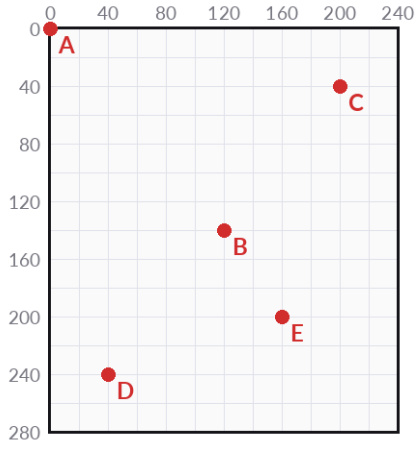
Setze **x** auf 0 und teste. Wo steht der Text jetzt?

_____

Setze **y** auf 0 und teste. Wo steht der Text jetzt?

_____

Aufgabe 2 ⭐ Koordinaten ablesen



Der Bildschirm ist **240** Pixel breit und **280** Pixel hoch. Jeder Punkt hat zwei Koordinaten: **x** zählt nach rechts, **y** zählt nach **unten**. (0, 0) ist oben links.

Schreibe die Koordinaten der Punkte auf:

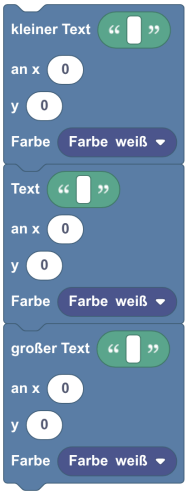
A (,) _____ **B** (,) _____

C (,) _____ **D** (,) _____

E (,) _____

Prüfe eine Antwort: Schreibe mit „kleiner Text“ ein **E** an die Stelle von Punkt E. Der Punkt ist die Ecke oben links des Buchstabens.

Aufgabe 3 ⭐ Drei Größen



In **Display** gibt es drei Text-Blöcke. Schreibe deinen Namen dreimal untereinander: klein, mittel und groß. Die Texte dürfen sich nicht überlappen!

Tipp: Ein kleiner Text ist 16 Pixel hoch, ein mittlerer 24 und ein großer 32 Pixel.

Meine y-Werte: klein _____ mittel _____ groß _____

Aufgabe 4 ⭐⭐ Meine Visitenkarte

Baue eine Visitenkarte: dein Name groß und bunt, darunter deine Klasse (mittel), darunter ein Satz über dein Hobby (klein). Setze über jeden Teil einen Kommentar-Block (ganz unten in der Blockleiste unter **Kommentar**), z. B. „Name“, „Klasse“, „Hobby“: 

Achtung: Die Schrift kennt kein ä, ö, ü und ß. Schreibe `ae`, `oe`, `ue` und `ss` – also `Fussball` statt „Fußball“.

Aufgabe 5 ⭐⭐⭐ Vier Ecken

Schreibe mit „großer Text“ ein **A** in jede Ecke des Bildschirms – so nah an die Ecke wie möglich, aber ganz zu sehen. Was fällt dir an den Ecken des Bildschirms auf?

_____

Nachgedacht: Schreibe „Grüße“ auf den Bildschirm. Was erscheint? Warum?

Das habe ich gelernt:

- ☐ Ich kann einen Text an eine bestimmte Stelle schreiben.
- ☐ Ich weiß: (0, 0) ist oben links, x zählt nach rechts, y nach unten.
- ☐ Ich kenne die drei Textgrößen und weiß, wie hoch sie sind.
- ☐ Ich kann Kommentare in mein Programm schreiben.

Mehr dazu im Buch: Kapitel 2 – Das erste Programm

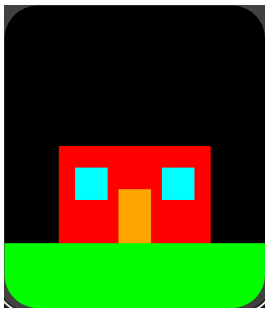
Name: _____ Klasse: _____ Datum: _____

Ein Bild aus Formen

> **Ziel:** Du zeichnest ein Haus und ein Gesicht aus Rechtecken, Linien, Kreisen und Ellipsen. Dabei lernst du, Farben selbst zu mischen.

Öffne PBC+ Studio, wähle **Emulator** und klicke auf **Dateien → Neu**. Alle Blöcke findest du in der Kategorie **Display**.

Aufgabe 1 ⭐ Das Haus nach Bauplan



Beim Rechteck sind **x** und **y** die Ecke **oben links**, **b** die Breite, **h** die Höhe. Ganz vorne wählst du **gefüllt**. So sieht die Wiese aus:



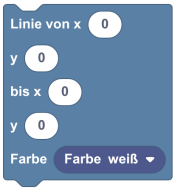
Baue unter „Bildschirm löschen“ alle fünf Rechtecke:

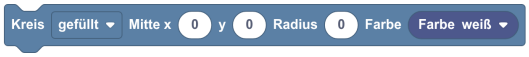
Teil	x	y	b	h	Farbe
Wiese	0	220	240	60	grün
Mauer	50	130	140	90	rot
Tür	105	170	30	50	orange
Fenster links	65	150	30	30	cyan
Fenster rechts	145	150	30	30	cyan

Tipp: Klicke mit der rechten Maustaste auf ein Rechteck und wähle **Kopieren** – dann musst du nur noch die Zahlen ändern.

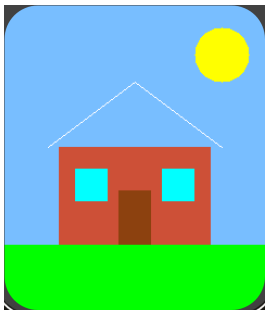
Aufgabe 2 ⭐ Dach und Sonne

Das Dach sind zwei Linien: von (40, 130) bis (120, 70) und von (120, 70) bis (200, 130). Die Sonne ist ein gelber Kreis mit der Mitte (200, 45) und dem Radius 25. Ergänze diese drei Blöcke (zwei Linien, ein Kreis) am Ende deines Programms:





Aufgabe 3 ⭐⭐ Eigene Farben



1. Ersetze „Bildschirm löschen“ durch „Bildschirm füllen“ mit einem hellblauen Himmel. Die Farbe mischst du mit „Farbe(r, g, b)“:

2. Mische für die Mauer ein Ziegelrot: rot 200, grün 80, blau 60.
3. Gib der Tür mit dem „Farbwähler“ ein Braun:

Was passiert, wenn „Bildschirm füllen“ ganz **unten** im Programm steht?

Aufgabe 4 ⭐⭐ Ein Smiley

Starte ein neues Programm (**Dateien → Neu**). Zeichne ein Smiley:

- Gesicht: gelber Kreis, Mitte (120, 140), Radius 80
- Augen: zwei schwarze Ellipsen, Mitte (90, 115) und (150, 115), Breite 20, Höhe 30
- Mund: zuerst eine **schwarze** Ellipse, Mitte (120, 170), Breite 100, Höhe 50 – danach eine **gelbe** Ellipse, Mitte (120, 160), Breite 100, Höhe 40. Warum entsteht so ein lachender Mund?

Aufgabe 5 ⭐⭐⭐ Mein eigenes Bild

Zeichne selbst ein Bild aus Formen: einen Roboter, ein Auto, einen Schneemann, eine Blume … Plane zuerst: Skizziere das Bild auf Kästchenpapier und schreibe die Koordinaten dazu.

Das habe ich gelernt:

☐ Ich kann Rechtecke, Linien, Kreise und Ellipsen zeichnen.

☐ Ich weiß: Beim Rechteck ist x, y die Ecke oben links, bei Kreis und Ellipse die Mitte.

☐ Ich kann Farben aus Rot, Grün und Blau mischen.

☐ Ich weiß: Was später gemalt wird, liegt oben drauf.

Mehr dazu im Buch: Kapitel 3 – Zeichnen und Farben

Name: _____ Klasse: _____ Datum: _____

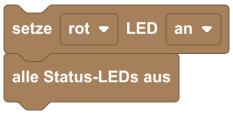
Die Ampel

Ziel: Du programmierst eine Ampel mit den drei Lämpchen des PBC+. Dabei lernst du, wie wichtig die **Reihenfolge** der Blöcke ist und wie du dein Programm **warten** lässt.



So geht's: Links neben dem Bildschirm hat der PBC+ drei kleine Lämpchen (LEDs): oben **rot**, in der Mitte **gelb**, unten **grün**. Im Emulator siehst du sie an derselben Stelle.

Die Blöcke dafür findest du in der Kategorie **LEDs**. Im Block „setze … LED …" wählst du die Farbe und **an** oder **aus**:



Die Warte-Blöcke findest du in der Kategorie **Zeit**: „warte … Sekunden" und „warte … ms" (1000 ms = 1 Sekunde).

Aufgabe 1 ⭐ Erst Rot, dann Grün



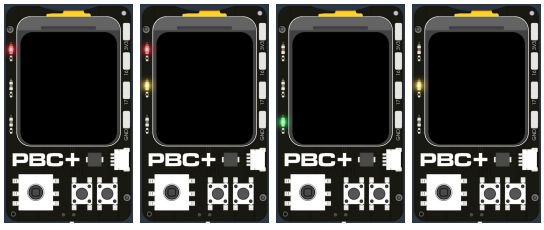
Wähle **Emulator**, klicke auf **Dateien → Neu** und baue dieses Programm. Teste es und beobachte die Lämpchen.

Lösche den Block „setze rot LED aus" und teste noch einmal. Was ist jetzt anders?

_____

Aufgabe 2 ⭐ Die Verkehrsampel

Eine echte Ampel schaltet so. Programmiere sie! Beginne mit dem Block „alle Status-LEDs aus". Überlege bei jeder Phase: Welche LED muss **an**, welche muss **aus**?



Phase	Lämpchen	Dauer
1	Rot	3 s
2	Rot und Gelb	1 s
3	Grün	3 s
4	Gelb	1 s
5	Rot	–

Tipp: Schreibe vor jede Phase einen Kommentar, zum Beispiel „Rot und Gelb".

Aufgabe 3 ⭐⭐ Für Fußgänger

Zeige zusätzlich auf dem Bildschirm, was Fußgänger tun sollen: Wenn die Ampel grün ist, steht in Grün **GEHEN** da, sonst in Rot **STOP**. Lösche den Bildschirm vor jedem neuen Wort. *Tipp:* Mit „großer Text“ bei y = 124 steht STOP bei x = 86 und GEHEN bei x = 74 in der Mitte.

Aufgabe 4 ⭐⭐ Wie lange dauert das?

Zähle alle Wartezeiten deiner Ampel zusammen: _____ Sekunden = _____ Millisekunden.

Hänge ganz unten einen „kleiner Text“ an. Stecke in sein Textfeld den Block „Zeit seit Programmstart (ms)“ aus der Kategorie **Zeit**. Welche Zahl erscheint am Ende? _____ Warum ist sie etwas größer als deine Rechnung?

_____

Aufgabe 5 ⭐⭐⭐ Nachts blinkt Gelb

Nachts sind manche Ampeln aus und blinken nur gelb. Lass die gelbe LED dreimal blinken: eine halbe Sekunde an, eine halbe Sekunde aus. *Tipp:* Klicke mit der rechten Maustaste auf einen Block und wähle **Kopieren**. Wie viele Blöcke brauchst du? _____

Das habe ich gelernt:

- ☐ Ich kann die drei Status-LEDs an- und ausschalten.
- ☐ Ich kann mein Programm Sekunden oder Millisekunden warten lassen.
- ☐ Ich weiß: Die Blöcke laufen genau in der Reihenfolge, in der sie hängen.
- ☐ Ich kann messen, wie lange mein Programm gelaufen ist.

Mehr dazu im Buch: Kapitel 4 – Schritt für Schritt: Reihenfolge und Zeit, Kapitel 9 – Ausgabe: LEDs und Töne

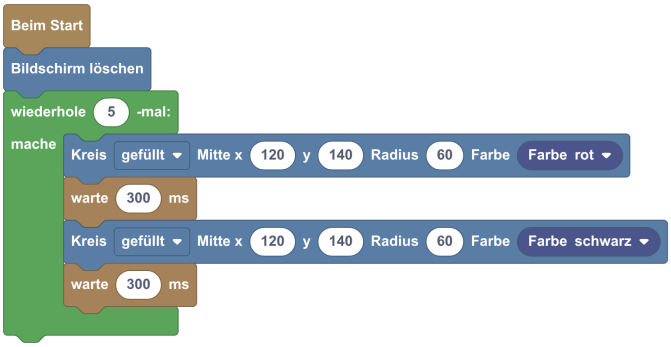
Name: _____ Klasse: _____ Datum: _____

Muster mit Schleifen

Ziel: Mit **Schleifen** lässt du Blöcke wiederholen. Du zeichnest damit Streifen, eine Treppe und ein Schachbrett – mit nur wenigen Blöcken.

Die Schleifen findest du in der Kategorie **Kontrollstrukturen**. Die Blöcke, die wiederholt werden sollen, ziehst du **in** die Schleife hinein.

Aufgabe 1 ⭐ Blinklicht



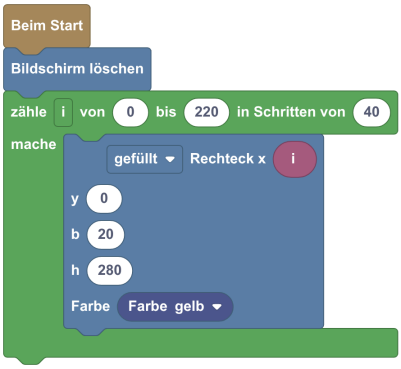
Baue dieses Programm und teste es. Wie oft blinkt der Kreis?

_____

Lass ihn 10-mal blinken, doppelt so schnell und in einer anderen Farbe.

Achtung: Ein neuer Block „wiederhole … -mal“ steht auf **0** – dann passiert gar nichts.

Aufgabe 2 ⭐ Streifen



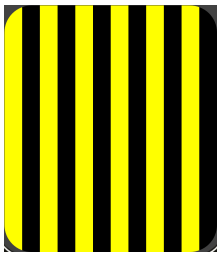
Neues Programm! Die Schleife „für i von … bis…“ zählt mit: `i` ist erst 0, dann 40, dann 80 … Den Block `i` findest du in **Variablen**: (i)

Baue das Programm. Welche Werte hat `i`?

_____

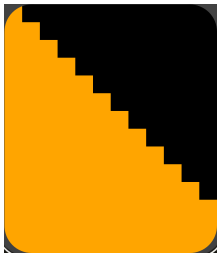
Ändere „in Schritten von“ auf **20**. Was passiert?

_____



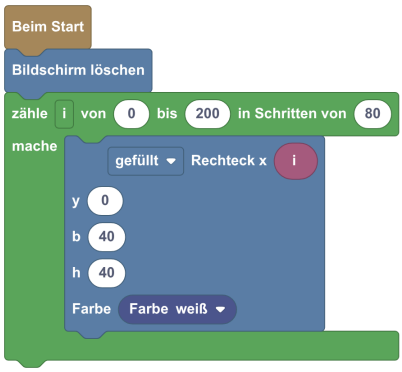
Aufgabe 3 ⭐ Die Treppe

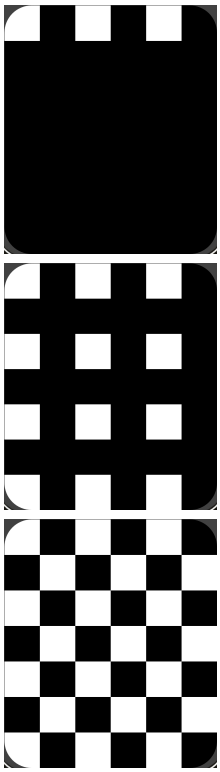
Ändere dein Streifen-Programm: Setze beim Rechteck auch für **y** den Block `i` ein, stelle die Schrittweite auf 20 und die Farbe auf Orange. Es entsteht eine Treppe! Warum? _____



Aufgabe 4 ⭐⭐ Das Schachbrett

1. **Eine Reihe:** Baue dieses Programm. Es zeichnet drei weiße Quadrate nebeneinander.

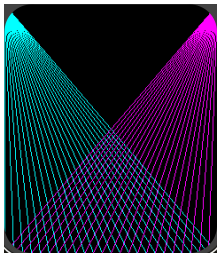




2. **Viele Reihen:** Hole eine zweite Zählschleife. Sie zählt automatisch mit `j`. Stelle sie auf „von 0 bis 240 in Schritten von 80“ und ziehe die Schleife aus Schritt 1 **hinein**. Setze beim Rechteck für y den Block `j` ein (aus **Variablen**).
3. **Die Lücken füllen:** Kopiere die ganze äußere Schleife (rechte Maustaste → **Kopieren**) und hänge die Kopie darunter. Stelle in der Kopie bei **beiden** Schleifen „von“ auf 40.

Aufgabe 5 ⭐⭐⭐ Mein Muster

Erfinde ein eigenes Muster: Ringe mit „Kreis“ (Radius `i`), Gitter aus Linien, ein Fächer … Ein Beispiel: Linien von (0, 0) bis (`i`, 280) und von (240, 0) bis (`i`, 280), mit `i` von 0 bis 240 in Schritten von 10.



Nachgedacht: Wie oft läuft „zähle i von 0 bis 220 in Schritten von 40“? Warum ist `i` nie 220?

Das habe ich gelernt:

☐ Ich kann Blöcke mit „wiederhole … -mal“ wiederholen lassen.

☐ Ich kann mit „für i von … bis…“ zählen und `i` als Zahl einsetzen.

☐ Ich kann eine Schleife in eine andere Schleife stecken.

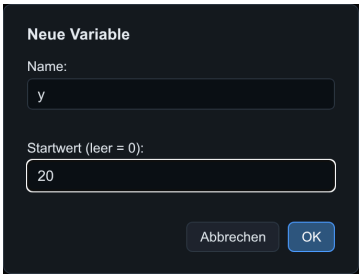
Mehr dazu im Buch: Kapitel 5 – Wiederholungen

Name: _____ Klasse: _____ Datum: _____

Der springende Ball

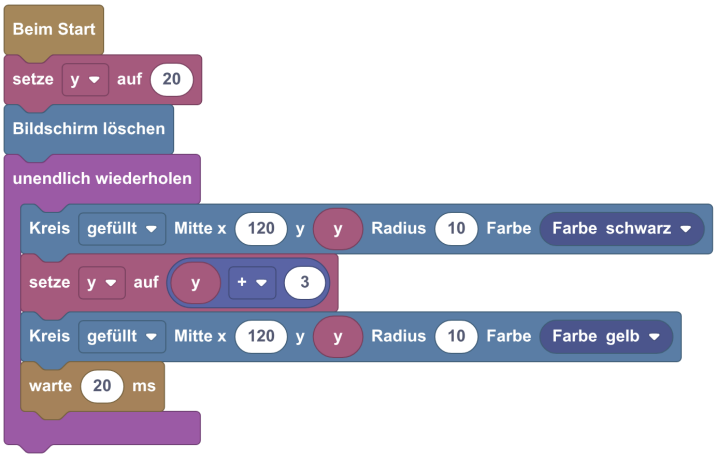
Ziel: Du bringst einen Ball zum Springen. Dabei lernst du **Variablen** kennen und erfährst, wie eine Bewegung ohne Flackern gelingt.

Vorbereitung: eine Variable anlegen



Eine **Variable** ist wie eine Schachtel mit Namen, in der eine Zahl liegt. Klappe die Kategorie **Variablen** auf und klicke auf **„+ neue Variable…“**. Gib als Name `y` und als Startwert `20` ein und klicke **OK**. Unter „Beim Start“ erscheint jetzt der Block „setze y auf 20“.

Aufgabe 1 ⭐ Der Ball fällt



Baue das Programm nach. Den Kreis findest du unter **Display**, das Rechnen „y + 3“ unter **Mathe**, den runden Block `y` unter **Variablen**.

In jeder Runde wird der Ball **schwarz übermalt**, `y` wird um 3 größer, und der Ball wird **gelb** an der neuen Stelle gemalt.

Was passiert, wenn der Ball unten ankommt?

_____

Aufgabe 2 ⭐ Warum schwarz übermalen?

1. Lösche den ersten Block in der Schleife (den schwarzen Kreis): anklicken und **Entf** drücken. Teste. Was siehst du? _____
2. Hole an seine Stelle **„Bildschirm löschen“**. Teste. Was siehst du jetzt?

   _____
3. Mache alles wieder rückgängig (**Strg + Z**), bis der schwarze Kreis wieder da ist.

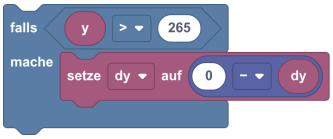
Aufgabe 3 ⭐ Der Ball springt zurück

Der Ball soll unten abprallen und wieder hochfliegen – und oben auch.

1. Lege eine zweite Variable `dy` mit dem Startwert `3` an. Sie sagt, wie weit der Ball in einem Schritt fliegt.
2. Ersetze „y + 3“ durch „y + dy“:



3. Baue diesen Block (Kategorie **Kontrollstrukturen**) in die Schleife – direkt unter „setze y auf y + dy“:

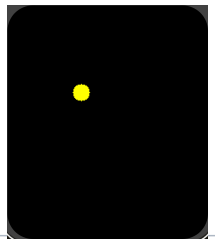


   Aus 3 wird so −3: Der Ball fliegt nach oben.

4. Baue einen zweiten solchen Block für oben ein: falls `y` < 15.

Aufgabe 4 ⭐⭐ Schräg hüpfen

Jetzt fliegt der Ball auch nach links und rechts.



1. Lege die Variablen `x` (Startwert 120) und `dx` (Startwert 2) an.
2. Setze in **beide** Kreis-Blöcke bei **Mitte x** die Variable `x` statt 120.
3. Ändere `x` in jeder Runde um `dx` – so wie `y` um `dy`.
4. Lass den Ball auch am linken und rechten Rand abprallen (x > 225 und x < 15).

Aufgabe 5 ⭐⭐⭐ Plopp!

Bei jedem Aufprall soll ein kurzer Ton erklingen (Kategorie **Sound**). Und noch schwieriger: Ändere die Zahlen so, dass der Ball schneller wird. Welche Variablen musst du dafür ändern?

Das habe ich gelernt:

- ☐ Ich kann eine Variable anlegen, setzen und lesen.
- ☐ Ich kann mit „setze y auf y + 3“ eine Variable verändern.
- ☐ Ich weiß, warum ich den Ball erst übermale und dann neu zeichne.
- ☐ Ich kann einen Ball am Rand abprallen lassen.

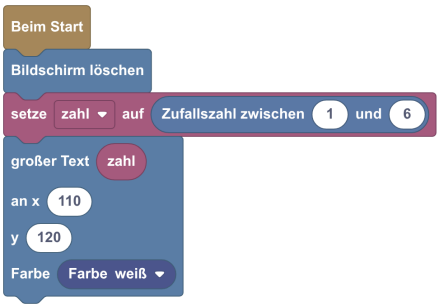
Mehr dazu im Buch: Kapitel 6 – Variablen und Kapitel 8.4 – Einen Punkt mit dem Joystick steuern

Name: _____ Klasse: _____ Datum: _____

Der Würfel

Ziel: Du baust einen elektronischen Würfel. Dabei lernst du die **Zufallszahl** kennen und fragst ab, ob die Taste A gedrückt wurde.

Aufgabe 1 ⭐ Eine Zufallszahl



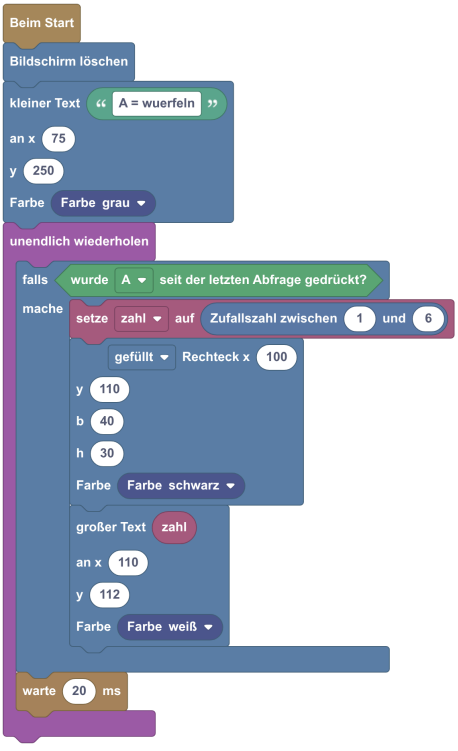
Lege unter **Variablen** die Variable `zahl` an (Startwert leer lassen). Baue dann das Programm nach. „Zufallszahl zwischen 1 und 6“ findest du unter **Mathe**.

Teste fünfmal mit ▶ **Testen**. Notiere die Zahlen:

_____

Welche Zahlen können vorkommen? Was ändert sich, wenn du „zwischen 1 und 6“ in „zwischen 0 und 6“ änderst? _____

Aufgabe 2 ⭐ Würfeln mit A



Jetzt würfelt der PBC+ jedes Mal, wenn du **A** drückst (im Emulator: Taste A auf der Tastatur).

Der Block „wurde A seit der letzten Abfrage gedrückt?“ steht in der Kategorie **Eingabe**. Er ist genau **einmal** wahr, wenn du A drückst.

Das schwarze Rechteck übermalt die alte Zahl. Ohne es würden die Zahlen übereinander stehen.

Baue das Programm nach und teste.

Aufgabe 3 ⭐ Mit Text und Ton

Statt nur der Zahl soll „Du hast eine 4“ erscheinen, und beim Würfeln erklingt ein kurzer Ton.

1. Ersetze den großen Text durch diesen Block (verbinde … mit… findest du unter **Text**):



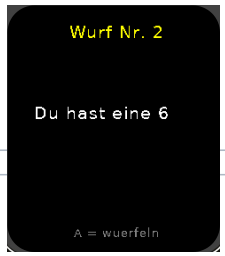
2. Mach das schwarze Rechteck breiter: x = 0, b = 240.
3. Hänge einen Block „Ton 600 Hz für 100 ms“ (Kategorie **Sound**) an.

Aufgabe 4 ⭐⭐ „wird“ oder „wurde“?

Tausche „wurde A seit der letzten Abfrage gedrückt?“ gegen „wird A gedrückt?“. Halte A eine Sekunde lang fest. Was passiert? Warum?

Aufgabe 5 ⭐⭐ Würfe zählen

Lege eine Variable `wuerfe` an. Bei jedem Wurf wird sie um 1 größer: „setze wuerfe auf wuerfe + 1“. Zeige oben auf dem Bildschirm „Wurf Nr. 3“ an. Denke an das schwarze Rechteck!



Aufgabe 6 ⭐⭐⭐ Der Würfel rollt

Bevor die richtige Zahl erscheint, sollen schnell hintereinander zehn zufällige Zahlen in Grau aufblitzen, jede mit einem kurzen Klick-Ton – wie ein rollender Würfel. *Tipp:* „wiederhole 10-mal“ mit „warte 60 ms“.

Das habe ich gelernt:

- ☐ Ich kann mit „Zufallszahl zwischen … und …“ würfeln.
- ☐ Ich kann eine Zufallszahl in einer Variable speichern und anzeigen.
- ☐ Ich kann mit „wurde A … gedrückt?“ auf einen Tastendruck reagieren.
- ☐ Ich kenne den Unterschied zwischen „wird … gedrückt?“ und „wurde … gedrückt?“.

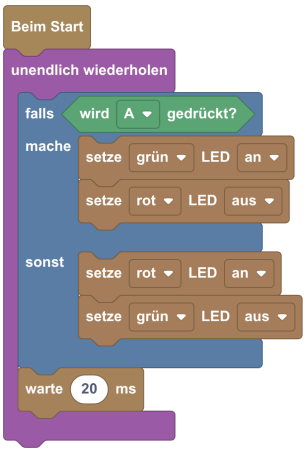
Mehr dazu im Buch: Kapitel 6.3 – Zufallszahlen und Kapitel 8 – Eingabe: Tasten und Joystick

Name: _____ Klasse: _____ Datum: _____

Wenn – dann – sonst

Ziel: Dein Programm trifft Entscheidungen. Du lernst **„falls / sonst“**, **Vergleiche** und die Verknüpfungen **und** / **oder** kennen.

Aufgabe 1 ⭐ Die Fußgängerampel



Baue das Programm nach. „falls / sonst“ findest du unter **Kontrollstrukturen**, „wird A gedrückt?“ unter **Eingabe**, die LEDs unter **LEDs**.

Teste. Die LEDs leuchten links neben dem Bildschirm.

Wann ist die Ampel grün? _____

Wann ist sie rot? _____

Aufgabe 2 ⭐⭐ und / oder

Jetzt sollen zwei Leute drücken. Hole aus **Mathe** den Block „und / oder“ und stecke zwei Tasten-Fragen hinein: 

Setze ihn in das „falls“ ein. Probiere dann auch **oder** aus (Feld ▾). Wann wird die Ampel grün? Kreuze an.

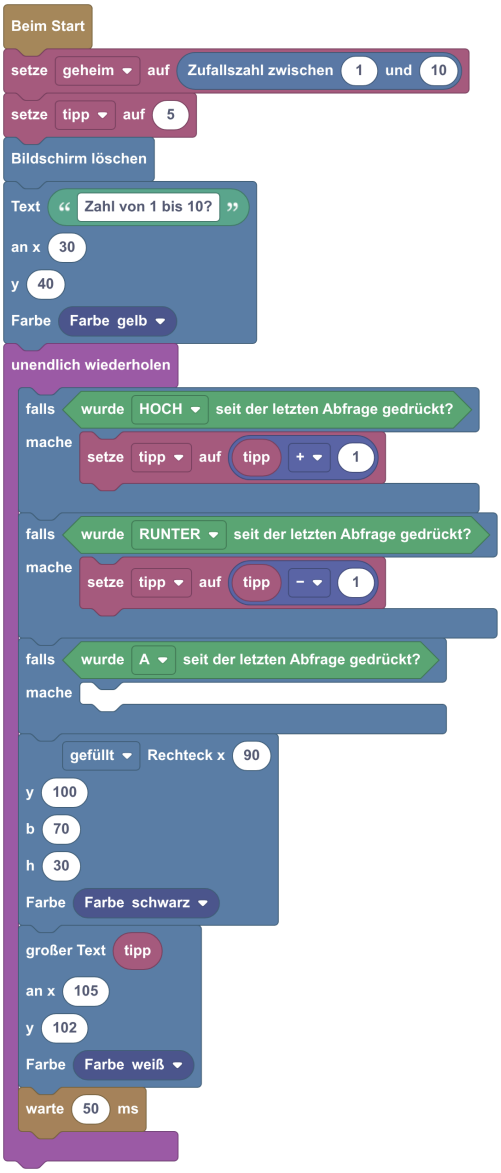
gedrückt	nur A	nur B	A und B	keine
mit **und** grün?	☐	☐	☐	☐
mit **oder** grün?	☐	☐	☐	☐

Aufgabe 3 ⭐ Wahr oder falsch?

Schreibe **w** (wahr) oder **f** (falsch) dahinter.

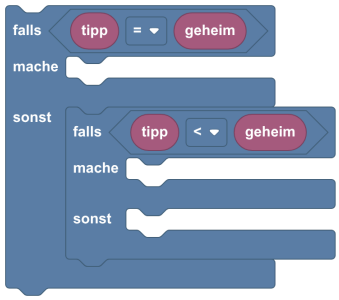
7 > 3	5 = 6	4 < 4	4 ≤ 4	3 < 5 und 8 < 6	3 < 5 oder 8 < 6

Aufgabe 4 ⭐⭐ Zahlenraten



Der PBC+ denkt sich eine Zahl `geheim` von 1 bis 10 aus. Mit **HOCH** und **RUNTER** stellst du deinen Tipp `tipp` ein, mit **A** gibst du ihn ab.

1. Lege die Variablen `geheim` und `tipp` an und baue die Vorlage nach.
2. Noch passiert nichts, wenn du A drückst. Baue in den letzten „falls“-Block ein „falls / sonst“ mit einem zweiten „falls / sonst“ darin ein:



3. Ergänze in den drei Lücken: „Richtig!“, „zu klein“ oder „zu gross“. Übermale vorher die alte Meldung mit einem schwarzen Rechteck (x 0, y 170, b 240, h 24).

Aufgabe 5 ⭐⭐⭐ Profi-Raten

a) Zähle in einer Variable `versuche`, wie oft du A gedrückt hast, und zeige sie unten an. b) Sorge mit **und** dafür, dass `tipp` nicht über 10 und nicht unter 1 rutschen kann.

Das habe ich gelernt:

- ☐ Ich kann mit „falls / sonst“ zwischen zwei Wegen entscheiden.
- ☐ Ich kann Zahlen mit = < > vergleichen.
- ☐ Ich weiß, wann „und“ und wann „oder“ wahr ist.
- ☐ Ich kann ein „falls / sonst“ in ein anderes stecken.

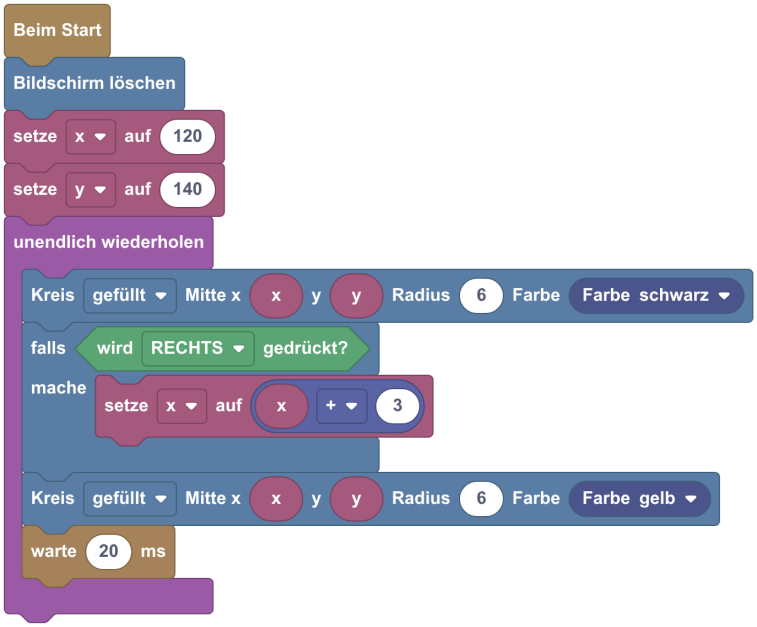
Mehr dazu im Buch: Kapitel 7 – Entscheidungen

Name: _____ Klasse: _____ Datum: _____

Steuern mit dem Joystick

Ziel: Du steuerst einen Punkt mit dem Joystick über den Bildschirm, lässt ihn am Rand anhalten und ins Ziel fahren. Neue Blöcke: **„wird … gedrückt?“** und **„wiederhole solange“**.

Aufgabe 1 ⭐ Der Punkt fährt



Lege die Variablen `x` (Startwert 120) und `y` (Startwert 140) an. Baue das Programm nach und teste: Mit der Pfeiltaste → fährt der Punkt nach rechts.

Ergänze drei weitere „falls“-Blöcke für **LINKS**, **HOCH** und **RUNTER**. *Tipp:* Nach oben heißt: `y` wird kleiner.

Merke: In jeder Runde übermalt das Programm den Punkt an der alten Stelle schwarz und malt ihn dann an der neuen Stelle gelb. Würde es den ganzen Bildschirm löschen, würde es flackern.

Aufgabe 2 ⭐ Halt am Rand!

Fahre mit dem Punkt nach rechts aus dem Bildschirm. Was passiert?

_____

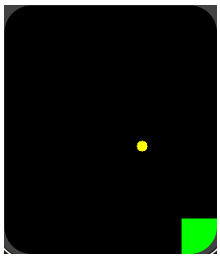
Der Punkt soll nur weiterfahren, wenn er noch Platz hat. Hole aus **Mathe** den Block „und“ und baue diese Bedingung in den „falls“-Block für RECHTS:



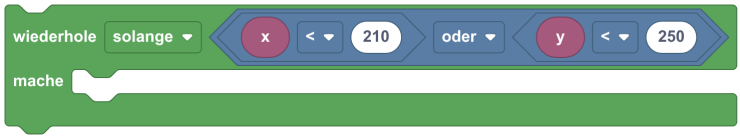
Mache das auch für die anderen Richtungen: LINKS nur, wenn `x` > 10; HOCH nur, wenn `y` > 10; RUNTER nur, wenn `y` < 270.

Aufgabe 3 ⭐⭐ Ab ins Ziel

Unten rechts liegt ein grünes Zielfeld. Das Spiel soll vorbei sein, wenn der Punkt drin ist.



1. Setze die Startwerte von `x` und `y` auf 20.
2. Tausche „unendlich wiederholen“ gegen diese Schleife aus **Kontrollstrukturen**:



3. Male als ersten Block **in** der Schleife das Zielfeld: gefülltes Rechteck, x 200, y 240, b 40, h 40, grün.
4. Hänge **unter** die Schleife einen Ton und den großen Text „Ziel!“.

Warum steht hier **oder** und nicht **und**?

Aufgabe 4 ⭐⭐ Turbo

Solange **B** gedrückt ist, soll der Punkt schneller fahren. Lege eine Variable `schritt` an. Setze sie in jeder Runde mit „falls / sonst“ auf 6 (B gedrückt) oder 2 (sonst) und nimm `schritt` statt 3 beim Bewegen.

Aufgabe 5 ⭐⭐⭐ Auf Zeit

Zeige im Ziel an, wie viele Sekunden du gebraucht hast, z. B. „Zeit: 7 s“. *Tipp:* „Zeit seit Programmstart (Sekunden)“ (Kategorie **Zeit**), „... runden“ und „verbinde ... mit...“.

Das habe ich gelernt:

- ☐ Ich kann einen Punkt mit „wird ... gedrückt?“ in vier Richtungen steuern.
- ☐ Ich kann einen Punkt ohne Flackern bewegen.
- ☐ Ich kann mit „und“ verhindern, dass er aus dem Bildschirm fährt.
- ☐ Ich kann mit „wiederhole solange“ ein Spiel beenden.

Mehr dazu im Buch: Kapitel 8 – Eingabe: Tasten und Joystick

Name: _____ Klasse: _____ Datum: _____

Musik!

Ziel: Du lässt den PBC+ Töne spielen, baust eine Sirene und programmierst ein Lied. Neue Blöcke: **„Ton“**, **„Dauerton“** und **„Ton stoppen“** (Kategorie **Sound**).

Im Emulator hörst du die Töne über die Lautsprecher oder Kopfhörer deines Computers.

Aufgabe 1 ⭐ Hoch und tief

Hänge diesen Block unter „Beim Start“ und teste: 

Die erste Zahl ist die Tonhöhe in **Hertz** (Hz), die zweite die Dauer in Millisekunden. Probiere 100 Hz, 1000 Hz und 4000 Hz aus.

Je größer die Zahl in Hz, desto _____ der Ton.

Aufgabe 2 ⭐ Die Tonleiter

Ton	c’	d’	e’	f’	g’	a’	h’	c’’
Hz	262	294	330	349	392	440	494	523



So beginnt die Tonleiter. Baue sie mit allen acht Tönen zu Ende. *Tipp:* Rechtsklick auf einen Block → **Kopieren**.

Der Ton-Block **wartet nicht**, bis der Ton fertig ist. Lösche einmal alle „warte“-Blöcke und teste. Was hörst du?

_____

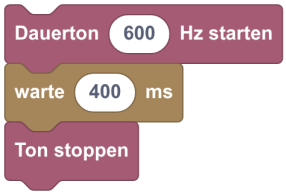
Aufgabe 3 ⭐ Hänschen klein

Eine Note = ein Ton und ein „warte“. Viertelnote: Ton 350 ms, warte 400 ms. Halbe Note: Ton 750 ms, warte 800 ms. Programmiere die erste Zeile:

Häns-	chen	klein	ging	al-	lein	in	die	wei-	te	Welt	hin-	ein
g’	e’	e’	f’	d’	d’	c’	d’	e’	f’	g’	g’	g’
¼	¼	½	¼	¼	½	¼	¼	¼	¼	¼	¼	½

Für Schnelle: Die zweite Zeile „Stock und Hut steht ihm gut, ist gar wohlgemut“: g’ e’ e’(½) · f’ d’ d’(½) · c’ e’ g’ g’ · c’ (1200 ms).

Aufgabe 4 ⭐⭐ Die Sirene



„Dauerton … starten“ klingt so lange, bis „Ton stoppen“ kommt. Baue eine Sirene: In einer Schleife, die **4-mal** läuft, wechseln sich 600 Hz und 800 Hz ab, jeder Ton 400 ms lang. Nach der Schleife: „Ton stoppen“.

Aufgabe 5 ⭐⭐⭐ Mini-Klavier

Solange du **LINKS** drückst, klingt c’ als Dauerton, bei **RECHTS** e’, bei **A** g’. Drückst du nichts, ist Ruhe. Spiele damit eine kleine Melodie! *Tipp:* „unendlich wiederholen“, „falls / sonst“ ineinander, „wird … gedrückt?“ (Kategorie **Eingabe**), am Ende „warte 20 ms“.

Aufgabe 6 ⭐⭐ Meine eigene Melodie

Erfinde eine eigene Melodie aus mindestens acht Noten oder nimm ein Lied, das du kennst. Trage Töne und Längen ein und programmiere sie.

Note	1	2	3	4	5	6	7	8	9	10
Ton										
Hz										
Länge (ms)										

Nachgedacht: Warum ist der Ton in einer Note 50 ms kürzer als das „warte“? Probiere „e’ e’“ einmal mit Ton 400 ms und warte 400 ms aus.

Das habe ich gelernt:

- ☐ Ich weiß: Je mehr Hertz, desto höher der Ton.
- ☐ Ich weiß, warum nach einem Ton ein „warte“ kommen muss.
- ☐ Ich kann eine Melodie aus Tönen und Pausen bauen.
- ☐ Ich kann einen Dauerton starten und stoppen.

Mehr dazu im Buch: Kapitel 9 – Ausgabe: LEDs und Töne

Name: _____ Klasse: _____ Datum: _____

Turtle 1: Vielecke

Ziel: Du lässt eine Turtle (Schildkröte) mit ihrem Stift Quadrate, Dreiecke und Sterne malen. Dabei entdeckst du, um wie viel Grad sie sich an jeder Ecke drehen muss.

So funktioniert die Turtle: Sie startet in der Mitte des Bildschirms und blickt nach **oben**. „vorwärts“ malt eine Linie. „rechts drehen um 90 Grad“ ist eine Vierteldrehung im Uhrzeigersinn – dabei malt sie nichts.

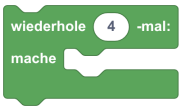
Aufgabe 1 ⭐ Das erste Quadrat



1. Öffne PBC+ Studio, wähle **Emulator**.
2. Klicke in der Kategorie **Turtle** auf **„neue Turtle erstellen“** und gib den Namen `kroete` ein. Hänge den Block unter „Beim Start“.
3. Baue das Programm nach. Die Turtle-Befehle findest du auch in der Kategorie **Turtle**. Teste.

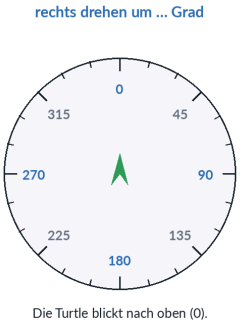
In welche Richtung läuft die Turtle zuerst? _____

Aufgabe 2 ⭐ Kürzer mit Schleife

Im Programm stehen zwei Blöcke viermal hintereinander. Nimm stattdessen diesen Block aus **Kontrollstrukturen** 

und lege nur **einmal** „vorwärts 80“ und „rechts drehen um 90“ hinein.

Lösche die übrigen Blöcke. Sieht das Quadrat genauso aus?





Hilfe: So weit dreht sich die Turtle bei „rechts drehen um …“.

Aufgabe 3 ⭐ Vielecke entdecken

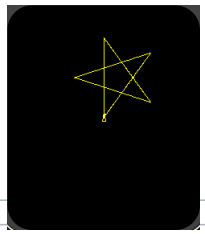
Ändere die Zahlen in deiner Schleife (Wiederholungen und Drehwinkel). Finde heraus, wie sich die Turtle drehen muss, damit das Vieleck sich genau schließt. Nimm 50 Pixel als Seitenlänge.

Vieleck	Ecken (Wiederholungen)	Drehwinkel	Ecken × Drehwinkel
Quadrat	4	90°	360°
Dreieck	3		
Fünfeck	5		
Sechseck	6		
Achteck	8		

Was fällt dir in der letzten Spalte auf? Ergänze die Regel: Drehwinkel = _____ ÷ Anzahl der Ecken

Aufgabe 4 ⭐⭐ Ein Stern

Lass die Turtle 5-mal 100 Pixel vorwärts laufen und sich jedes Mal um **144 Grad** nach rechts drehen. Mach den Stern gelb mit „Stiftfarbe auf“.



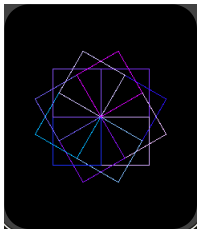
Rechne: 5 × 144° = _____ Wie oft hat sich die Turtle also ganz herum gedreht?

_____

Aufgabe 5 ⭐⭐⭐ Ein Muster aus Quadraten

Lege deine Quadrat-Schleife in eine zweite Schleife: 12-mal ein Quadrat malen und danach um 30 Grad drehen. Mit „Geschwindigkeit 10“ geht es schneller. Gib jedem Quadrat eine Zufallsfarbe:





Erfinde danach dein eigenes Muster, zum Beispiel aus Sechsecken.

Nachgedacht: Die Ecken eines Dreiecks sind spitz (60°). Warum muss sich die Turtle trotzdem um 120° drehen? *Tipp:* Spiel selbst Turtle!

Das habe ich gelernt:

- ☐ Ich kann eine Turtle erstellen und mit „vorwärts“ und „drehen“ steuern.
- ☐ Ich kann ein Vieleck mit einer Schleife malen.
- ☐ Ich weiß: Drehwinkel = 360° ÷ Anzahl der Ecken.
- ☐ Ich kann Schleifen ineinander legen, um Muster zu malen.

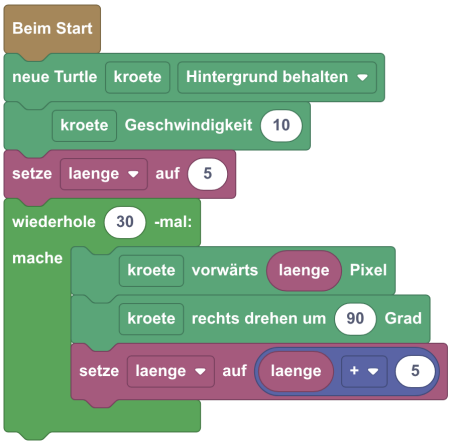
Mehr dazu im Buch: Kapitel 10 – Turtle-Grafik

Name: _____ Klasse: _____ Datum: _____

Turtle 2: Spiralen und Muster

Ziel: Die Turtle malt Spiralen. Dafür wird eine Variable in jeder Runde der Schleife ein bisschen größer. Mit Farben und Geschwindigkeit machst du daraus bunte Muster.

Aufgabe 1 ⭐ Die eckige Spirale



1. Erstelle eine Turtle mit dem Namen `kroete` (Kategorie **Turtle**, „neue Turtle erstellen“).
2. Lege unter **Variablen** die Variable `laenge` an.
3. Baue das Programm nach und teste es.

Die Turtle läuft zuerst 5 Pixel, dann 10, dann 15 … Wie lang ist die letzte Linie? _____

Aufgabe 2 ⭐ Andere Winkel

Ändere nur die Zahl bei „rechts drehen um“. Was für eine Spirale entsteht?

Winkel	Was siehst du?
120	
91	
144	
eigener:	

Aufgabe 3 ⭐ Bunte Spirale

Lege diesen Block an den Anfang der Schleife. Jetzt hat jede Linie eine zufällige Farbe:



Der Block „Farbe rot … grün … blau …“ mischt eine Farbe aus Rot, Grün und Blau. Jede Zahl geht von 0 bis 255.

Aufgabe 4 ⭐⭐ Wie schnell ist die Turtle?

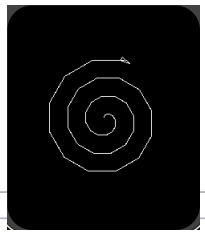
Ändere die Zahl im Block 

Probiere 1, 5, 10 und 0. Was bedeutet 0?

_____

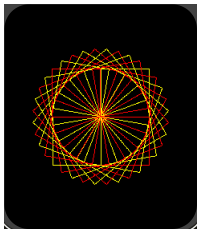
Aufgabe 5 ⭐⭐ Die runde Spirale

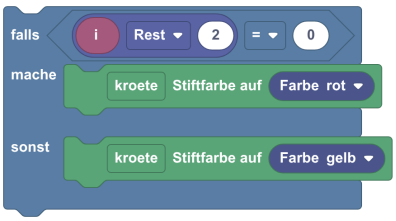
Baue eine Spirale, die fast rund aussieht: `laenge` startet bei 1, die Turtle dreht sich um 30 Grad, und `laenge` wird in jeder Runde nur um 1 größer. Wie viele Wiederholungen brauchst du, damit die Spirale nicht über den Bildschirm hinausgeht?



Aufgabe 6 ⭐⭐⭐ Die rot-gelbe Blume

Male 36 Quadrate (Seite 60). Nach jedem Quadrat dreht sich die Turtle um 10 Grad. Die Quadrate sind abwechselnd rot und gelb. *Tipp:* Nimm die Zählschleife „zähle i von 1 bis 36“. Ist der Rest von i ÷ 2 gleich 0, ist i gerade:





Nachgedacht: In der Schleife stehen in jeder Runde dieselben Blöcke. Warum wird die Linie trotzdem jedes Mal länger?

Das habe ich gelernt:

- ☐ Ich kann eine Variable in einer Schleife immer größer werden lassen.
- ☐ Ich kann mit „vorwärts laenge“ eine Spirale malen.
- ☐ Ich kann die Stiftfarbe ändern, auch zufällig.
- ☐ Ich weiß, was die Geschwindigkeit 0, 1 und 10 bedeutet.

Mehr dazu im Buch: Kapitel 10 – Turtle-Grafik

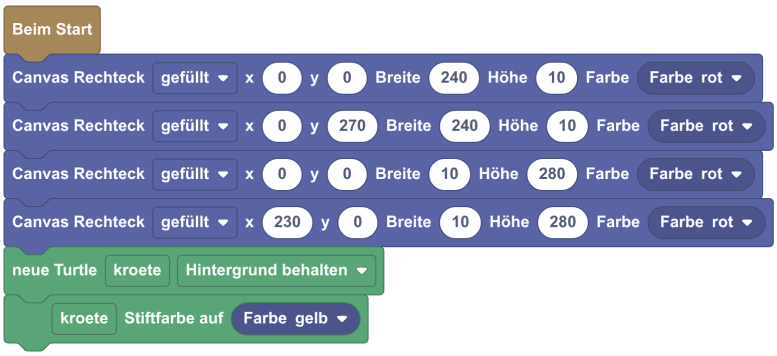
Name: _____ Klasse: _____ Datum: _____

Turtle 3: Die Turtle sieht Farben

Ziel: Die Turtle fährt bis zur Wand und dreht um – wie ein Staubsauger-Roboter. Dafür lernst du den Frage-Block **„Turtle … berührt Farbe …“** kennen.

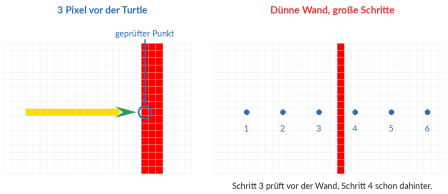
Aufgabe 1 ⭐ Ein Zimmer mit roten Wänden

Baue diese Vorlage. Die vier Rechtecke findest du in der Kategorie **Canvas**, die Turtle-Blöcke in der Kategorie **Turtle** (Name: `kroete`). Teste: Du siehst vier rote Wände und die Turtle in der Mitte.

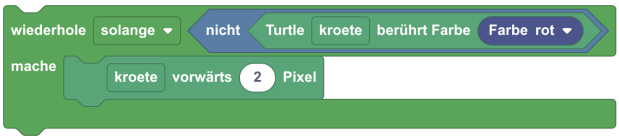


Aufgabe 2 ⭐ Bis zur Wand und zurück

Der Block „Turtle berührt Farbe“ prüft den Punkt **3 Pixel vor der Turtle**. Ist er genau rot, antwortet er mit **wahr**.



Hänge diese Schleife (Kategorie **Kontrollstrukturen**, **Mathe**, **Turtle**) an deine Vorlage:



Was tut die Turtle? _____

Lass sie danach mit „rechts drehen um 180 Grad“ umdrehen. Klicke mit der rechten Maustaste auf die Schleife, wähle **Kopieren** und hänge die Kopie darunter. Was passiert jetzt?

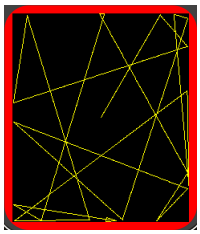
_____

Aufgabe 3 ⭐ Immer hin und her

Ersetze deine Blöcke aus Aufgabe 2 durch „unendlich wiederholen“. Darin: dieser „falls“-Block, dann „kroete vorwärts 2 Pixel“ und „warte 10 ms“.



Aufgabe 4 ⭐⭐ Der Staubsauger-Roboter

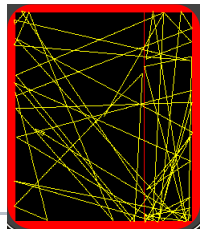


1. Dreh die Turtle vor der Schleife um 30 Grad, damit sie schräg losfährt.
2. Im „falls“-Block soll sie sich nicht immer um 180 Grad drehen, sondern um eine Zufallszahl:



Lass den Roboter eine Minute fahren. Wird der ganze Raum „gesaugt“?

Aufgabe 5 ⭐⭐ Die dünne Wand



Male mit diesem Block eine dünne rote Wand quer durch den Raum:

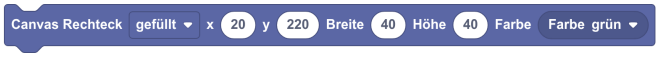


Ändere „vorwärts 2“ in „vorwärts 5“.

Beobachte die dünne Wand. Was passiert manchmal? Warum?

Aufgabe 6 ⭐⭐⭐ Ab ins Ziel!

Male unten links ein grünes Zielfeld:



Der Roboter fährt nur, **solange** er nicht Grün sieht. Zähle in einer Variable `stoesse`, wie oft er gegen die Wand stößt. Am Ziel versteckt sich die Turtle, und auf dem Canvas steht, wie oft sie angestoßen ist. *Tipp:* Nach „Text auf Canvas“ brauchst du „Canvas zeigen“.

Das habe ich gelernt:

- ☐ Ich kann mit „Turtle berührt Farbe“ prüfen, ob vor der Turtle eine Farbe ist.
- ☐ Ich kann die Frage in „solange“ oder „falls“ benutzen.
- ☐ Ich weiß, warum Wände dicker sein müssen als ein Schritt der Turtle.

Mehr dazu im Buch: Kapitel 10 – Turtle-Grafik (10.5)

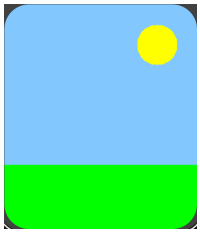
Name: _____ Klasse: _____ Datum: _____

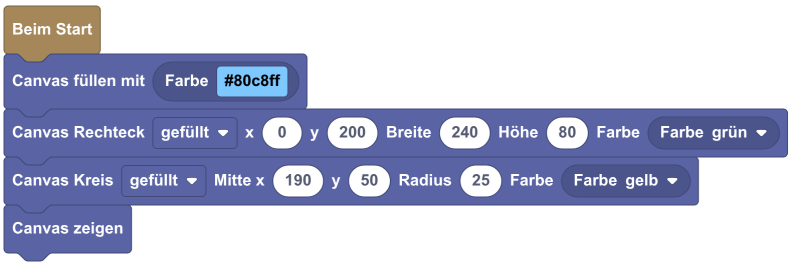
Mein Hintergrund

Ziel: Du malst ein Bild auf den **Canvas** – ein Bild im Speicher – und zeigst es dann auf einmal. Danach legst du ein fertiges Bild als Hintergrund auf den Canvas. Später laufen deine Sprites vor diesem Hintergrund.

Aufgabe 1 ⭐ Eine Landschaft

Baue dieses Programm nach. Alle Blöcke findest du in der Kategorie **Canvas**. Die hellblaue Farbe wählst du mit dem Farbwähler aus.





Aufgabe 2 ⭐ Ohne „Canvas zeigen“

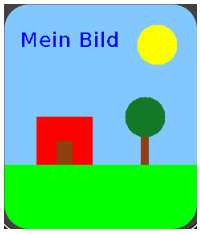
Lösche den letzten Block „Canvas zeigen“ und teste. Was siehst du? _____

Warum ist das so? _____

Hänge „Canvas zeigen“ danach wieder an.

Aufgabe 3 ⭐ Mein Bild

Male weitere Dinge **vor** „Canvas zeigen“: ein Haus, einen Baum, eine Wolke … Diese Blöcke helfen dir:





Achtung: Der Text-Block für den Canvas sieht aus wie der Text-Block aus **Display**. Nimm ihn aus der Kategorie **Canvas**! Umlaute zeigt der PBC+ nicht an – schreibe „ue“ statt „ü“.

Was du später malst, liegt oben. Was musst du also zuerst malen? _____

Aufgabe 4 ⭐⭐ Ein Hintergrundbild



1. Lade das Bild `weltraum.png` hoch: **Dateien → Bilder verwalten…**
2. Beginne ein neues Programm. Lege das Bild mit diesem Block auf den Canvas:



3. Schreibe einen Text darauf und vergiss „Canvas zeigen“ nicht.

Das Bild ist 240 × 280 Pixel groß – genau so groß wie der Bildschirm.

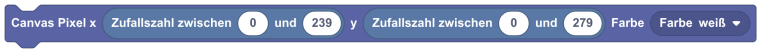
Aufgabe 5 ⭐⭐ Selbst gemalt

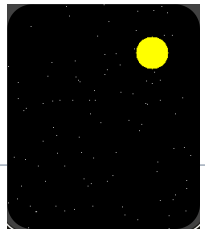
Male ein kleines Bild (zum Beispiel 32 × 32 Pixel) mit **Dateien → Bild zeichnen…** und speichere es. Lege es mit „Bild auf Canvas“ fünfmal an verschiedene Stellen auf deinen Weltraum.



Aufgabe 6 ⭐⭐⭐ Sternenhimmel

Male 100 weiße Sterne an zufällige Stellen und einen gelben Mond. Lege diesen Block in „wiederhole 100-mal“:





Nachgedacht: Wenn du den Bildschirm löschst und neu malst, flackert das Bild. Warum flackert es nicht, wenn du auf den Canvas malst und dann „Canvas zeigen“ benutzt?

Das habe ich gelernt:

- ☐ Ich kann Formen und Text auf den Canvas malen.
- ☐ Ich weiß, dass ich den Canvas erst mit „Canvas zeigen“ sehe.
- ☐ Ich kann ein Bild hochladen und mit „Bild auf Canvas“ als Hintergrund benutzen.

Mehr dazu im Buch: Kapitel 11 – Der Canvas – ein Hintergrund im Speicher

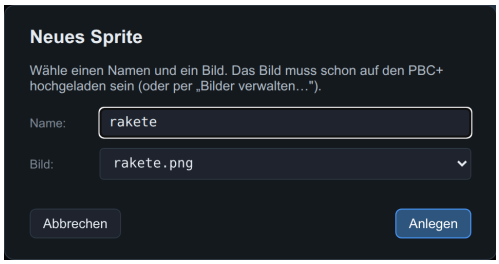
Name: _____ Klasse: _____ Datum: _____

Sprites in Bewegung

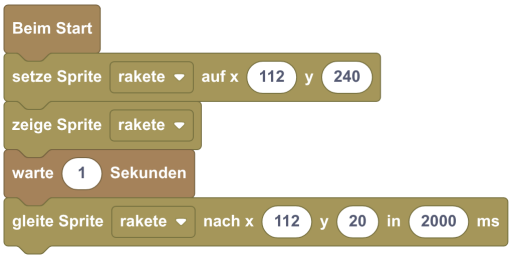
Ziel: Du legst Sprites an – kleine Bilder mit einem Namen –, lässt eine Rakete starten und steuerst sie mit dem Joystick durch das Weltall.

Vorbereitung

1. Öffne PBC+ Studio und wähle **Emulator**.
2. Lade die Bilder `rakete.png`, `ufo.png` und `weltraum.png` hoch: **Dateien → Bilder verwalten…**
3. Klicke in der Kategorie **Sprites** auf **„+ neues Sprite…“**. Name: `rakete`, Bild: `rakete.png`, dann **Anlegen**.
4. Lege genauso das Sprite `ufo` mit dem Bild `ufo.png` an.



Aufgabe 1 ⭐ Start!



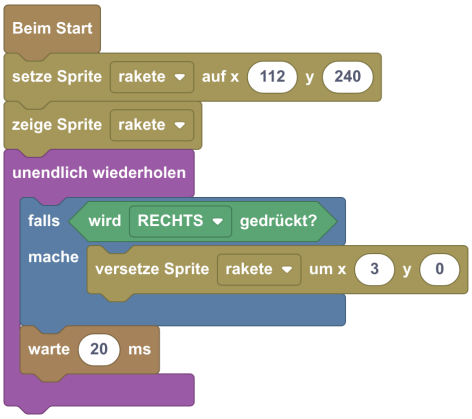
Baue das Programm nach und teste es.

Was passiert, wenn du den Block „zeige Sprite“ weglässt?

_____

x und y gehören zur **linken oberen Ecke** des Bildes. Wie bei allen Bildschirm-Koordinaten zählt y nach **unten**.

Aufgabe 2 ⭐ Steuern mit dem Joystick



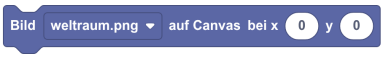
Baue ein neues Programm: Mit der Pfeiltaste → fliegt die Rakete nach rechts.

Ergänze drei weitere „falls“-Blöcke für **links**, **hoch** und **runter**.

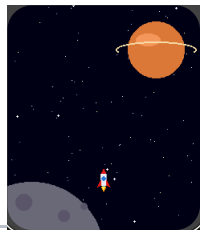
Tipp: Nach links: x wird kleiner. Nach oben: y wird kleiner.

Aufgabe 3 ⭐ Ab ins Weltall

Hänge diesen Block aus der Kategorie **Canvas** direkt unter „Beim Start“:

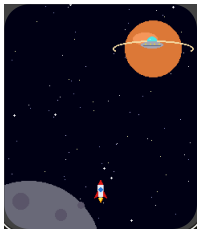


Was passiert mit dem Hintergrund, wenn die Rakete darüber fliegt?



Aufgabe 4 ⭐⭐ Das Ufo patrouilliert

Das Ufo soll oben hin und her fliegen, während du die Rakete steuerst.



1. Lege die Variable `schritt` an und setze sie vor der Schleife auf 2.
2. Setze das Ufo auf x = 0, y = 40 und zeige es.
3. Lege diese Blöcke in die Schleife:



Warum 212? Das Ufo ist 28 Pixel breit: 240 − 28 = 212.

Aufgabe 5 ⭐⭐⭐ Nicht aus dem Bild!

Die Rakete soll am Rand stehen bleiben. Ändere deine vier „falls“-Blöcke so:



Die Rakete ist 16 Pixel breit und 28 Pixel hoch. Welche Grenzen brauchst du für links, oben und unten?

Nachgedacht: Ersetze im Programm aus Aufgabe 4 die Blöcke für das Ufo durch „gleite Sprite ufo nach x 212 y 40 in 2000 ms“. Warum kannst du die Rakete jetzt nicht mehr flüssig steuern?

Das habe ich gelernt:

- ☐ Ich kann ein Sprite anlegen, setzen und zeigen.
- ☐ Ich kann ein Sprite gleiten lassen und mit dem Joystick versetzen.
- ☐ Ich kann mit „x von Sprite“ abfragen, wo ein Sprite steht.
- ☐ Ich weiß, dass der Canvas der Hintergrund der Sprites ist.

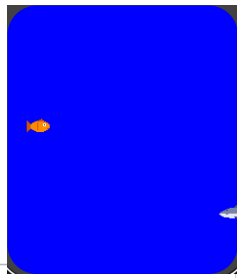
Mehr dazu im Buch: Kapitel 12 – Sprites – Figuren, die sich bewegen

Name: _____ Klasse: _____ Datum: _____

Kollision 1: Achtung, Hai!

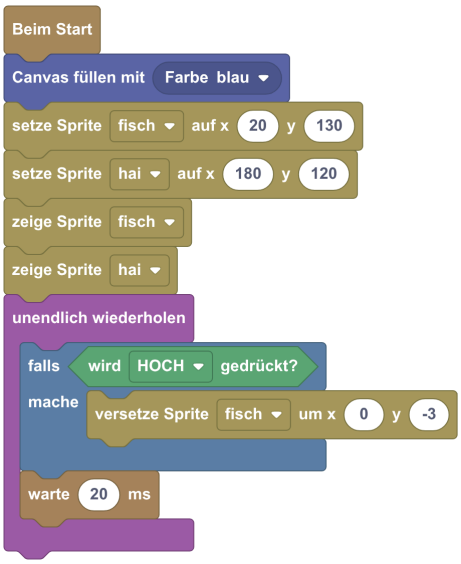
Ziel: Du programmierst ein kleines Spiel: Ein Fisch muss einem Hai ausweichen. Dabei lernst du den Block **„Sprite … berührt Sprite …“** kennen.

Vorbereitung



1. Öffne PBC+ Studio und wähle oben rechts **Emulator**.
2. Lade die Bilder `fisch.png` und `hai.png` hoch: **Dateien → Bilder verwalten…**
3. Lege in der Kategorie **Sprites** mit **„+ neues Sprite…“** zwei Sprites an: `fisch` mit dem Bild `fisch.png` und `hai` mit dem Bild `hai.png`.

Aufgabe 1 ⭐ Der Fisch schwimmt



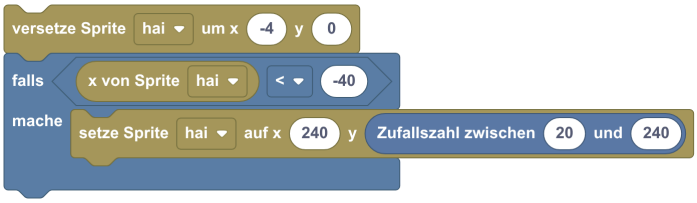
Baue dieses Programm nach und teste es mit ▶ **Testen**. Mit der Pfeiltaste ↑ schwimmt der Fisch nach oben.

Ergänze einen zweiten „falls“-Block, damit der Fisch auch nach **unten** schwimmen kann.

Tipp: Nach unten heißt: y wird größer.

Aufgabe 2 ⭐ Der Hai kommt

Baue diese Blöcke **in die Schleife** ein – direkt über „warte 20 ms“:



Was macht der „falls“-Block? *Tipp:* Der Hai ist 40 Pixel breit.

Aufgabe 3 ⭐ Erwischt!

Baue auch diesen Block in die Schleife ein, direkt über „warte 20 ms“:



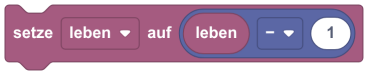
Teste dein Spiel. Was passiert, wenn der Hai den Fisch erwischt?

Aufgabe 4 ⭐⭐ Drei Leben

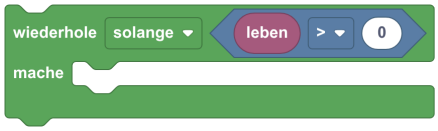
Nach drei Treffern ist das Spiel vorbei.



1. Lege unter **Variablen** die Variable `leben` an. Setze sie gleich nach „Beim Start“ auf 3.
2. Baue diesen Block in den „falls“-Block aus Aufgabe 3 ein:



3. Ersetze „unendlich wiederholen“ durch diese Schleife:



4. Hänge die Blöcke rechts **unter** die Schleife.

Aufgabe 5 ⭐⭐⭐ Immer schneller

Jedes Mal, wenn der Hai wieder rechts auftaucht, soll er 1 Pixel schneller schwimmen. *Tipp:* Lege eine Variable `tempo` an (Start: 4). Der Hai schwimmt um „0 − tempo“ nach links.

Nachgedacht: Manchmal ist der Fisch „erwischt“, obwohl sich die Bilder gar nicht berühren. Woran liegt das?

Das habe ich gelernt:

- ☐ Ich kann mit „berührt Sprite“ prüfen, ob sich zwei Sprites berühren.
- ☐ Ich weiß, warum die Frage in der Schleife stehen muss.
- ☐ Ich kann mit einer Variable Leben zählen und das Spiel beenden.
- ☐ Ich weiß, dass für „berührt Sprite“ die Rechtecke der Bilder zählen.

Mehr dazu im Buch: Kapitel 13 – Kollisionen

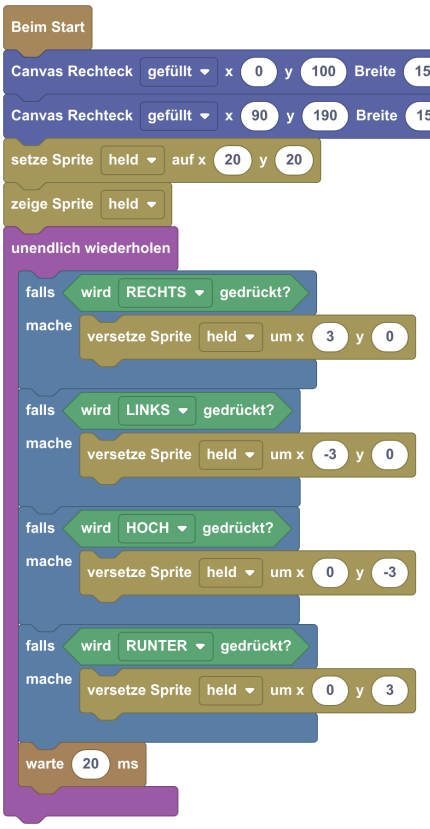
Name: _____ Klasse: _____ Datum: _____

Kollision 2: Vorsicht, Lava!

Ziel: Du malst Lava auf den Hintergrund. Dein Held muss heil hindurch bis ins Ziel. Dabei lernst du die Blöcke **„Sprite … berührt Farbe …“** und **„Sprite … berührt Rand“** kennen.

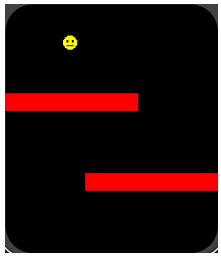
Vorbereitung: Wähle **Emulator**. Lade das Bild `held.png` hoch (**Dateien → Bilder verwalten…**) und lege das Sprite `held` mit diesem Bild an.

Aufgabe 1 ⭐ Lava malen



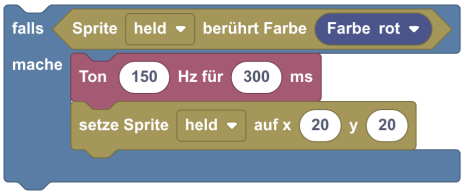
Baue dieses Programm nach und teste es. Die beiden roten Rechtecke sind die Lava. Du findest die Blöcke in den Kategorien **Canvas**, **Sprites**, **Kontrollstrukturen**, **Eingabe** und **Zeit**.

Was passiert, wenn der Held über die Lava läuft?



Aufgabe 2 ⭐ Autsch, heiß!

Baue diesen Block **in die Schleife** ein, direkt über „warte 20 ms“. Teste.

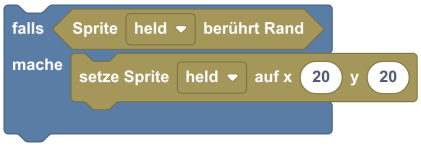


Ändere jetzt im Kollisions-Block „Farbe rot“ in „Farbe orange“. Was passiert? Warum? Stelle danach wieder rot ein.

Aufgabe 3 ⭐ Nicht weglaufen!

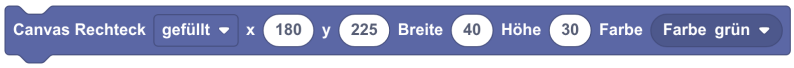
Lauf mit dem Held aus dem Bildschirm. Was passiert? _____

Baue diesen Block in die Schleife ein. Er schickt den Held zurück zum Start, sobald er den Rand berührt:



Aufgabe 4 ⭐⭐ Ins Ziel

1. Male unten rechts ein grünes Zielfeld. Hänge dazu diesen Block unter die beiden Lava-Blöcke:



2. Ersetze „unendlich wiederholen“ durch diese Schleife (Kategorien **Kontrollstrukturen**, **Mathe** und **Sprites**):



3. Unter der Schleife: Verstecke den Held und schreibe „Geschafft!“ mit „großer Text“ auf den Bildschirm.

Aufgabe 5 ⭐⭐⭐ Superkraft

Baue diesen Block in die Schleife ein: Mit Taste **A** springt der Held 60 Pixel nach unten.



Stell den Held knapp über die obere Lava und drücke A. Was passiert? Warum merkt das Programm die Lava nicht?

_____

_____

Das habe ich gelernt:

- ☐ Ich kann mit „berührt Farbe“ gemalte Hindernisse erkennen.
- ☐ Ich weiß: Die Farbe im Block muss genau die gemalte Farbe sein.
- ☐ Ich kann mit „berührt Rand“ prüfen, ob ein Sprite den Bildschirm verlässt.
- ☐ Ich kann ein Spiel mit „solange nicht …“ beenden, wenn das Ziel erreicht ist.

Mehr dazu im Buch: Kapitel 13 – Kollisionen

Name: _____ Klasse: _____ Datum: _____

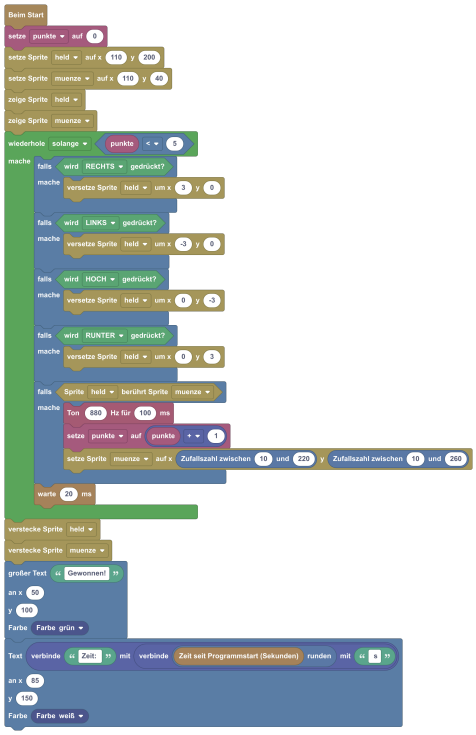
Ordnung mit Funktionen

Ziel: Du räumst ein langes Programm auf. Du teilst es in **Funktionen** mit guten Namen auf. Danach liest sich „Beim Start“ wie ein Inhaltsverzeichnis.

Vorbereitung

1. Wähle oben rechts **Emulator**. Lade die Bilder `held.png` und `muenze.png` hoch: **Dateien → Bilder verwalten…**
2. Öffne die Datei `ab18_vorlage.pbcp`: **Dateien → Vom Computer…**

Aufgabe 1 ⭐ Was macht welcher Teil?



Teste das Spiel mit ▶ **Testen**. Es ist das Münzspiel aus AB 16.

Das Programm hat vier Teile. Schreibe zu jedem Teil, welche Blöcke dazu gehören (z. B. „die ersten 5 Blöcke“).

Vorbereiten (Startwerte, Sprites):

Steuern (Held bewegen):

Münze prüfen (Treffer, Punkte):

Ende zeigen (Gewonnen, Zeit):

Aufgabe 2 ⭐ Die erste Funktion: `steuern`



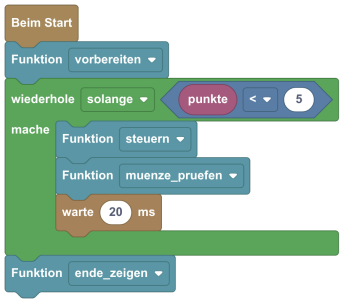
1. Kategorie **Funktionen** → **„+ neue Funktion…“**, Name `steuern`.
2. Ziehe den ersten „falls wird RECHTS gedrückt?“-Block in die neue Funktion. **Achtung:** Alle Blöcke darunter kommen mit!
3. Ziehe den Block „falls Sprite held berührt Sprite muenze“ (mit „warte“ darunter) zurück in die Schleife.
4. Klicke in der Kategorie **Funktionen** auf `steuern()` und setze den Aufruf oben in die Schleife:



5. Teste: Läuft der Held noch?

Aufgabe 3 ⭐ Alles aufräumen

Baue genauso die Funktionen `vorbereiten`, `muenze_pruefen` und `ende_zeigen`. Am Ende soll „Beim Start“ so aussehen:



Teste nach **jeder** neuen Funktion. Speichere dein Programm: **Dateien → Auf den Computer…**

Ziehe den Aufruf `steuern` kurz aus der Schleife heraus und teste. Was passiert? Warum?

Aufgabe 4 ⭐⭐ Eine Funktion mit Parameter: `piep`



Lege eine Funktion `piep` an. Klicke auf das grüne **+**: Ein Parameter `x` erscheint. Klicke hinein und nenne ihn `hoehe`. Den Block `hoehe` findest du in der Kategorie **Funktionen** unter `piep(hoehe)`. Ersetze in `muenze_pruefen` den Ton-Block durch den Aufruf (unten links) und spiele in `ende_zeigen` einen tiefen Piep mit 440.

Aufgabe 5 ⭐⭐⭐ Ein Geist kommt dazu

Lade `geist.png` hoch und lege ein Sprite `geist` an. Schreibe eine Funktion `geist_bewegen`: Der Geist schwebt hin und her (wie in Kapitel 13). Berührt er den Helden, muss der Held zurück an den Start. Welche Funktionen musst du noch ändern?

Das habe ich gelernt:

- ☐ Ich kann eine Funktion anlegen und Blöcke hineinziehen.
- ☐ Ich kann eine Funktion aufrufen.
- ☐ Ich kann einer Funktion einen Parameter geben.
- ☐ Ich finde gute Namen, die sagen, was eine Funktion tut.

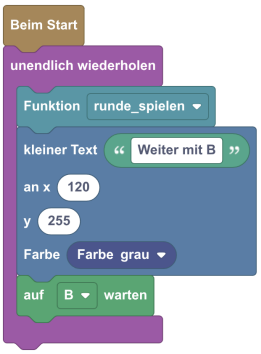
Mehr dazu im Buch: Kapitel 14 – Funktionen

Name: _____ Klasse: _____ Datum: _____

Listen 1: Meine Bestenliste

Ziel: Beim Klick-Rennen drückst du 5 Sekunden lang so oft wie möglich die Taste A. Dein Programm merkt sich alle Ergebnisse in einer **Liste**, zeigt sie an und findet deinen Rekord.

Vorbereitung: Wähle **Emulator** und öffne die Datei `ab19_vorlage.pbcp` (**Dateien → Vom Computer…**). Die Funktion `runde_spielen` spielt eine Runde und merkt sich das Ergebnis in der Variable `klicks`. „Beim Start" ruft sie immer wieder auf:





Nach drei Runden

Aufgabe 1 ⭐ Wie schnell bist du? Teste das Spiel: **B** startet, dann drückst du **A** (im Emulator: Taste A). Spiele drei Runden. Deine Ergebnisse: _____ _____ _____ Was ist das Problem, wenn du dir alle Ergebnisse merken willst?

_____

Aufgabe 2 ⭐ Fächer ab 0

Die Liste `ergebnisse` sieht so aus. Fülle aus:

Element Nr. 1 von Liste ergebnisse: _____

Länge von Liste ergebnisse: _____

Nummer des Elements 17: _____

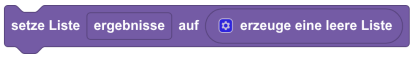
Nach „hänge 30 an" hat die 30 die Nummer: _____

Liste ergebnisse

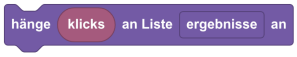
14	22	17
Nr. 0	Nr. 1	Nr. 2

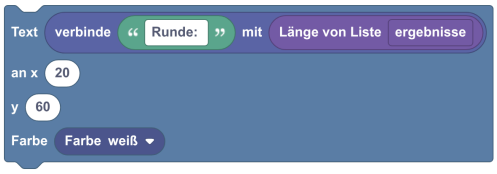
Aufgabe 3 ⭐ Ergebnisse sammeln

1. Kategorie **Listen** → **„+ neue Liste…“**, Name `ergebnisse`.
2. Setze **vor** „unendlich wiederholen“ diesen Block:



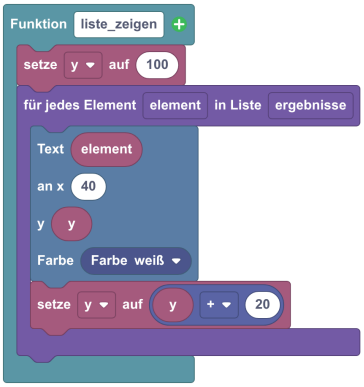
3. Setze direkt **unter** den Aufruf `runde_spielen` diese beiden Blöcke:





Teste: Zählt „Runde“ richtig mit?

Aufgabe 4 ⭐ Alle Ergebnisse anzeigen



Lege die Funktion `liste_zeigen` an und baue sie so nach. Rufe sie unter dem Block „Text Runde: …“ auf.

„für jedes Element“ läuft einmal für **jedes** Ergebnis. In `element` steht dabei immer das aktuelle Ergebnis. Warum wird `y` jedes Mal um 20 größer?

Aufgabe 5 ⭐⭐ Mein Rekord



Lege die Variable `rekord` an. Setze sie in `liste_zeigen` vor der Schleife auf 0. Baue den „falls“-Block **in** die Schleife. Zeige nach der Schleife „Rekord: …“ in Grün an (x 120, y 100).

Aufgabe 6 ⭐⭐⭐ Nur die letzten 8



Nach vielen Runden passen nicht mehr alle Ergebnisse auf den Bildschirm. Sorge dafür, dass die Liste höchstens 8 Ergebnisse hat: Ist sie länger, wird das älteste (Nr. 0) entfernt. Wohin gehört dieser Block?

Das habe ich gelernt:

- ☐ Ich kann eine Liste anlegen und Werte anhängen.
- ☐ Ich weiß, dass das erste Element die Nummer 0 hat.
- ☐ Ich kann mit „für jedes Element“ alle Werte einer Liste anzeigen.
- ☐ Ich kann den größten Wert einer Liste finden.

Mehr dazu im Buch: Kapitel 15 – Listen

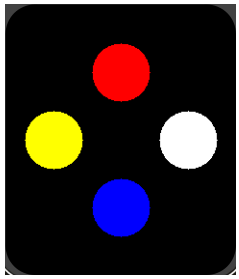
Name: _____ Klasse: _____ Datum: _____

Listen 2: Merk dir die Folge

Ziel: Du programmierst das Merkspiel „Simon sagt“. Der PicoBoy lässt farbige Felder in einer Folge aufleuchten, du spielst sie mit dem Joystick nach. In jeder Runde wird die Folge um eine Farbe länger. Die Folge steckt in einer **Liste**.

So geht das Spiel: Vier Felder: oben rot, rechts grün, unten blau, links gelb. Jedes Feld hat eine **Nummer** von 0 bis 3 und einen eigenen Ton. Drückst du den Joystick in eine Richtung, leuchtet das Feld dort weiß auf.

Vorbereitung: Wähle **Emulator** und beginne ein neues Programm (**Dateien → Neu**). Speichere nach jeder Aufgabe (**Dateien → Auf den Computer…**).

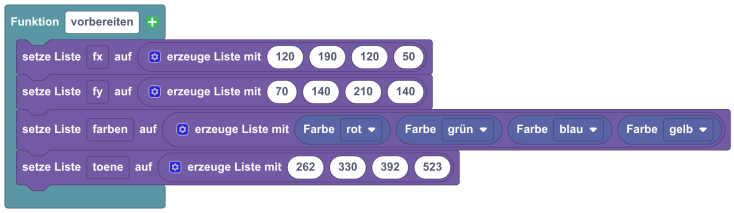


Rechts (grün) leuchtet gerade auf.

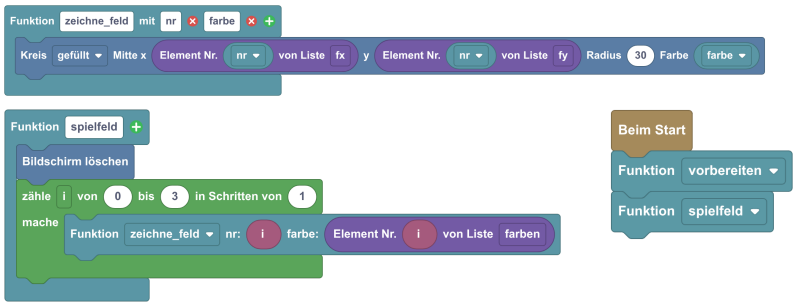
Stunde 1: Spielfeld und Folge

Aufgabe 1 ⭐ Das Spielfeld

Lege die Listen `fx`, `fy`, `farben` und `toene` an (Kategorie **Listen**, „+ neue Liste…“). Baue die Funktion `vorbereiten`. Die Lücken im Block „erzeuge Liste mit“ stellst du mit dem Zahnrad ein.



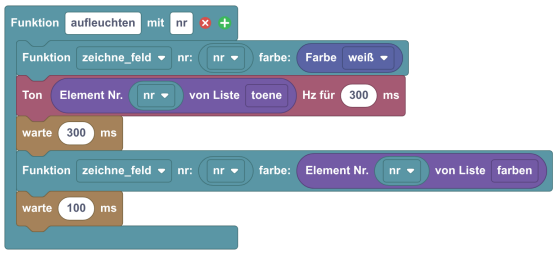
Baue dann die Funktion `zeichne_feld` mit zwei Parametern (grünes **+**, umbenennen in `nr` und `farbe`) und die Funktion `spielfeld`:



Rufe in „Beim Start“ `vorbereiten` und `spielfeld` auf (rechts) und teste.

Welche Werte holt `zeichne_feld` für das Feld Nr. 3 aus den Listen? x = _____ y = _____ Farbe = _____

Aufgabe 2 ⭐ Aufleuchten



Die Funktion `aufleuchten` malt das Feld weiß, spielt seinen Ton und malt es wieder in seiner Farbe.

Baue die Funktion. Teste sie, indem du unter „Beim Start“ zwei Aufrufe anhängst: „aufleuchten nr: 0“ und „aufleuchten nr: 2“.



Warum steht nach dem Ton-Block noch „warte 300 ms“?

Aufgabe 3 ⭐ Die Folge

Die Folge, die du dir merken musst, ist eine Liste von Feld-Nummern. Diese Folge bedeutet: unten, oben, links, links.

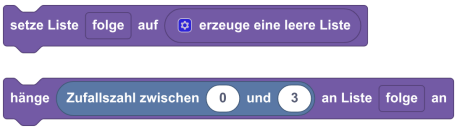
Welche Farben leuchten bei der Folge **[1, 1, 3, 0]**?

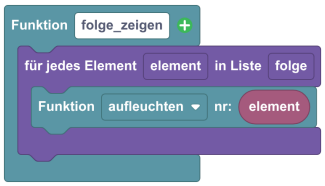
Liste folge

2	0	3	3
Nr. 0	Nr. 1	Nr. 2	Nr. 3

0 = oben (rot) 1 = rechts (grün)
2 = unten (blau) 3 = links (gelb)

Lege die Liste `folge` an. Baue die Funktion `folge_zeigen`. Ersetze die Test-Aufrufe aus Aufgabe 2 durch diese Blöcke und hänge darunter „warte 1000 ms“ und den Aufruf `folge_zeigen`:

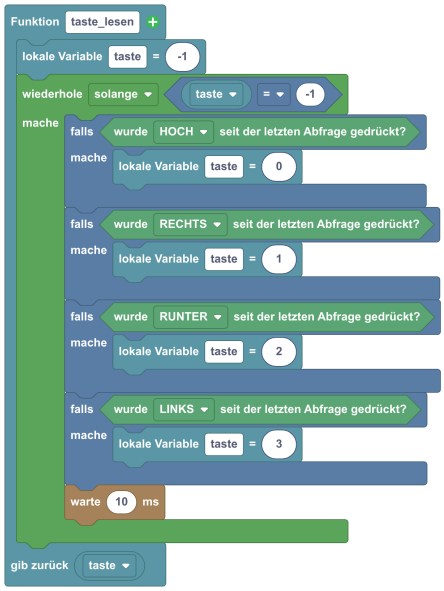




Hänge die Zufallszahl **dreimal** an. Teste mehrmals: Ist die Folge jedes Mal anders?

Stunde 2: Nachspielen

Aufgabe 4 ⭐ Den Joystick lesen

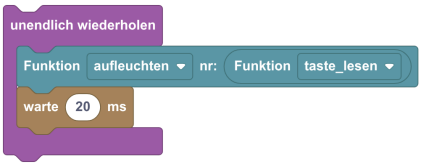


`taste_lesen` ist eine **Funktion mit Rückgabe**: „+ neue Funktion mit Rückgabe…“. Sie wartet, bis du den Joystick in eine Richtung drückst, und gibt die Nummer dieser Richtung zurück.

Die **lokale Variable** `taste` legst du unter `taste_lesen()` mit „+ neue lokale Variable…“ an. Den zweiten bis fünften Block „lokale Variable taste =“ bekommst du mit Rechtsklick → **Kopieren**.

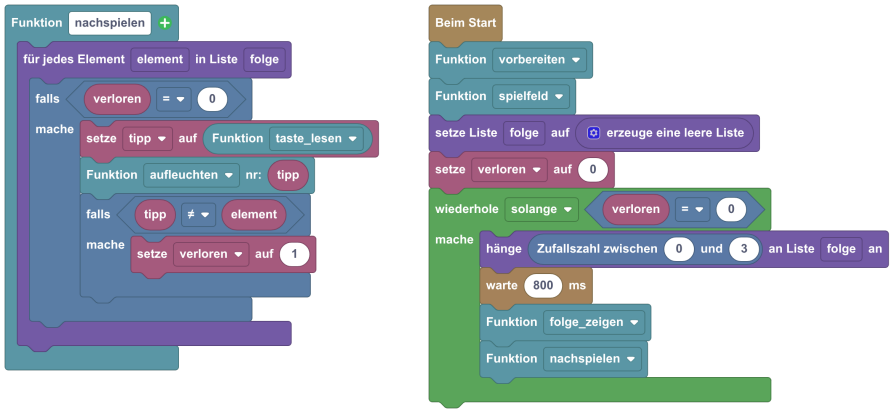
`taste` startet mit −1. Warum −1 und nicht 0?

Test: Ersetze in „Beim Start“ alles unter `spielfeld` durch diese Schleife. Jetzt leuchtet das Feld auf, in dessen Richtung du drückst.



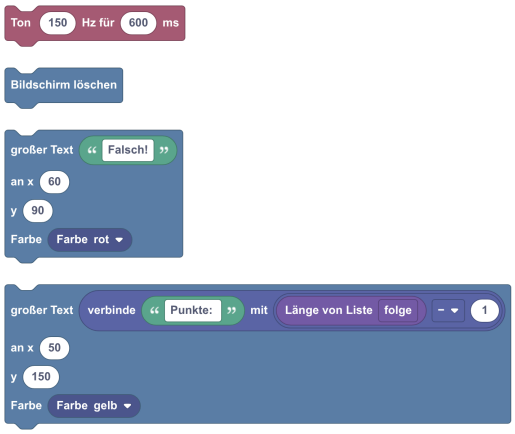
Aufgabe 5 ⭐⭐ Das Spiel

Lege die Variablen `verloren` und `tipp` an. Baue die Funktion `nachspielen` und ändere „Beim Start“ wie im Bild unten:



Erkläre: Warum wird in `nachspielen` geprüft, ob `verloren` noch 0 ist?

Aufgabe 6 ⭐⭐ Game over



Hänge diese Blöcke **unter** die „solange“-Schleife. Warum steht dort „Länge von Liste folge − 1“ und nicht einfach die Länge?

Mein Rekord: _____ Punkte

Aufgabe 7 ⭐⭐⭐ Immer schneller

Mit jeder Runde soll das Aufleuchten kürzer werden: 400 ms minus 30 ms pro Element der Folge, aber nie kürzer als 100 ms. Lege eine Variable `tempo` an und setze sie am Anfang jeder Runde auf

„Maximum von 100 und (400 − Länge von Liste folge × 30)“

(Block „Minimum / Maximum“ aus **Mathe**). Benutze `tempo` in `aufleuchten` statt der beiden 300. Weitere Ideen: ein Fehler-Ton, eine Bestenliste wie in AB 19, ein fünftes Feld für die Taste A.

Das habe ich gelernt:

- ☐ Ich kann Werte, die zusammengehören, in mehreren Listen mit gleicher Nummer speichern.
- ☐ Ich kann mit „Element Nr. …“ einen bestimmten Wert aus einer Liste holen.
- ☐ Ich kann eine Liste mit „hänge … an“ wachsen lassen und ihre Länge benutzen.
- ☐ Ich kann eine Funktion mit Rückgabe und einer lokalen Variable bauen.

Mehr dazu im Buch: Kapitel 15 – Listen (und Kapitel 14 – Funktionen)

Name: _____ Klasse: _____ Datum: _____

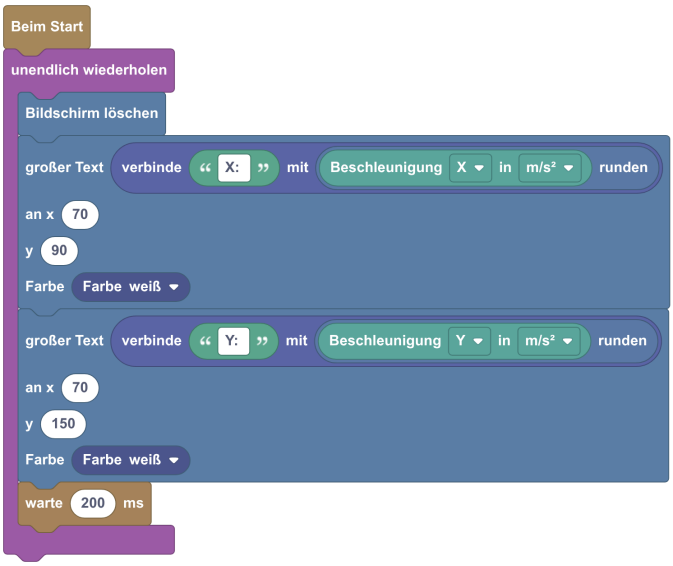
Kipp mich!

Ziel: Du liest den **Beschleunigungssensor** aus und steuerst eine Kugel, indem du den PicoBoy kippst. Neuer Block: **„Beschleunigung … in …“** (Kategorie **Sensor**).

Du brauchst das echte Gerät! Der Emulator hat keinen Sensor. Dort ist immer X = 0 und Y = 0 – die Kugel bewegt sich nicht.

Vorbereitung: Verbinde den PBC+ mit dem USB-Kabel, wähle als Ziel **PBC+** und klicke auf **Verbinden**. Lade das Bild `kugel.png` hoch (**Dateien → Bilder verwalten…**) und lege unter **Sprites** mit „+ neues Sprite…“ das Sprite `kugel` an.

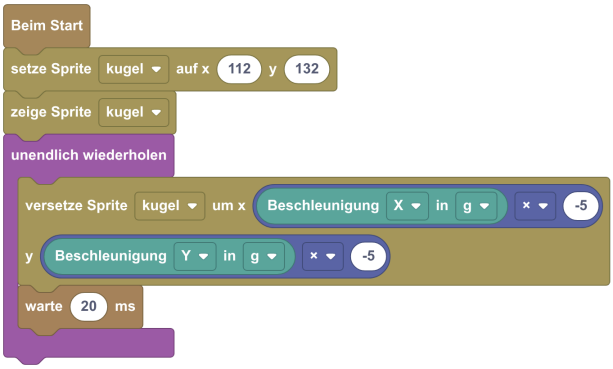
Aufgabe 1 ⭐ Messen



Baue das Programm nach und teste es auf dem PBC+. Stelle bei „Beschleunigung“ die Achse (X oder Y) und die Einheit **m/s²** ein. Halte das Gerät so, wie in der Tabelle steht, und trage die Werte ein.

Haltung	X	Y
flach auf dem Tisch		
rechte Seite nach unten		
linke Seite nach unten		
obere Kante nach unten		
untere Kante nach unten		

Aufgabe 2 ⭐ Die Kugel rollt



Baue dieses Programm. Stelle die Einheit **g** ein. Kippe den PBC+ vorsichtig in alle Richtungen.

Kippst du die rechte Seite nach unten, ist X negativ. Warum steht im Programm **−5** und nicht 5? Probiere es aus.

Aufgabe 3 ⭐⭐ Aus!



Rollt die Kugel aus dem Bildschirm, soll ein Ton kommen und die Kugel zurück in die Mitte springen. Baue diesen Block **in die Schleife** ein.

Aufgabe 4 ⭐⭐ Ins Loch

1. Lege eine Variable `punkte` an und setze sie am Anfang auf 0.
2. Male gleich nach „Beim Start“ ein grünes Loch (Kategorie **Canvas**):



3. Baue in die Schleife ein: Liegt die Kugel auf Grün, gibt es einen Ton, einen Punkt, und die Kugel startet wieder in der Mitte.



Aufgabe 5 ⭐⭐⭐ Drei Treffer

Nach 3 Treffern ist das Spiel vorbei: Die Kugel verschwindet, und „3 Treffer!“ und die Zeit in Sekunden erscheinen. Versuche dann, das Loch schwerer erreichbar zu machen: kleiner, weiter in der Ecke – oder mit roter Lava davor.

Nachgedacht: Hältst du den PBC+ nur ein kleines bisschen schief, bleibt die Kugel liegen. Erst bei stärkerem Kippen rollt sie los. Woran könnte das liegen? *Tipp:* Rechne 0,1 × −5. Kann ein Sprite um einen halben Pixel versetzt werden?

Das habe ich gelernt:

- ☐ Ich kann den Beschleunigungssensor auslesen und die Werte anzeigen.
- ☐ Ich weiß, wie sich X und Y beim Kippen ändern.
- ☐ Ich kann eine Kugel durch Kippen steuern.
- ☐ Ich weiß, warum das im Emulator nicht geht.

Mehr dazu im Buch: Kapitel 18 – Der Beschleunigungssensor

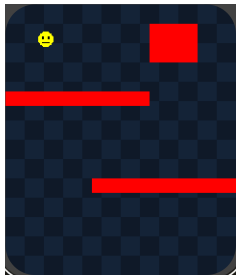
Name: _____ Klasse: _____ Datum: _____

Projekt Schatzjagd 1: Spielfeld, Held und Lava

Ziel: In drei Stunden baust du ein ganzes Spiel: Dein Held sammelt fünf Schätze ein und weicht dabei Lava und einem Geist aus. Heute baust du das Spielfeld, die Steuerung und die Lava. Dabei teilst du das Programm in **Funktionen** auf.

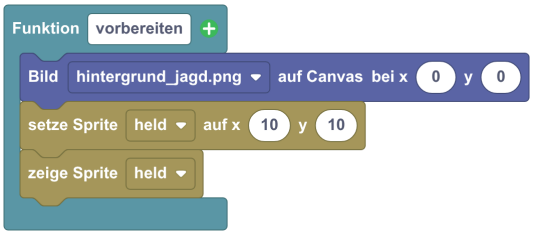
Vorbereitung

1. Öffne PBC+ Studio und wähle **Emulator** (oder verbinde deinen PBC+).
2. Lade die Bilder `hintergrund_jagd.png`, `held.png`, `muenze.png` und `geist.png` hoch: **Dateien → Bilder verwalten…**
3. Lege unter **Sprites** mit „+ neues Sprite…“ drei Sprites an: `held` (Bild `held.png`), `schatz` (Bild `muenze.png`) und `geist` (Bild `geist.png`). Die beiden letzten brauchst du erst nächste Stunde.



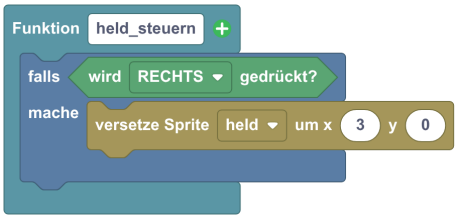
So sieht es heute am Ende aus.

Aufgabe 1 ⭐ Die Funktion `vorbereiten`



Klicke in der Kategorie **Funktionen** auf „+ neue Funktion…“ und gib den Namen `vorbereiten` ein. Baue die Blöcke hinein. „Bild auf Canvas“ findest du unter **Canvas**.

Aufgabe 2 ⭐ Die Funktion `held_steuern`



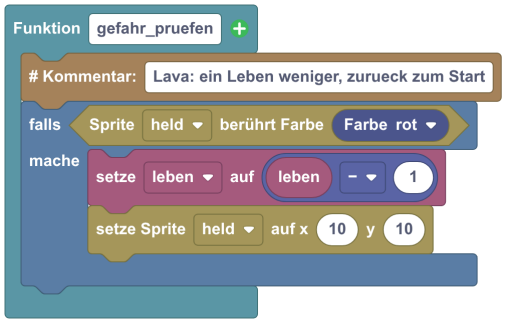
Lege die Funktion `held_steuern` an. Baue den Block für **rechts** nach und ergänze drei weitere „falls“-Blöcke für links, hoch und runter.



Eine Funktion tut nichts, bis sie **aufgerufen** wird. Klicke in der Kategorie **Funktionen** auf ihren Namen – dann bekommst du den Aufruf-Block. Baue so „Beim Start“ zusammen und teste: Der Held läuft. Auch durch die Lava – noch!

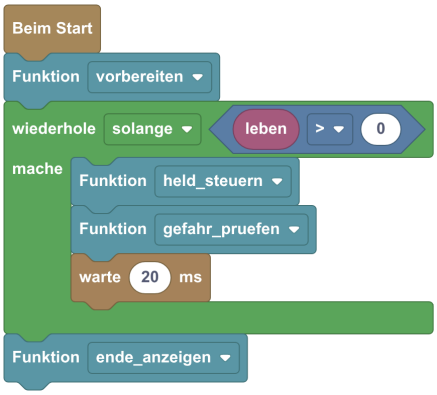
Aufgabe 3 ⭐⭐ Vorsicht, Lava!

1. Lege die Variable `leben` an. Setze sie in `vorbereiten` auf 3 (direkt unter „Bild auf Canvas“).
2. Lege die Funktion `gefahr_pruefen` an:



3. Lege die Funktion `ende_anzeigen` an und ändere „Beim Start“:





Teste: Nach drei Mal Lava erscheint „Game Over“.

Aufgabe 4 ⭐⭐⭐ Leben-Anzeige

Am Anfang leuchten alle drei Status-LEDs. Bei jedem verlorenen Leben geht eine aus (erst grün, dann gelb, dann rot) und ein tiefer Ton erklingt.

Nachgedacht: Das Programm ist in Funktionen aufgeteilt. Was ist daran praktisch? Schreibe einen Grund auf.

Speichern nicht vergessen! Dateien → Auf den Computer… – nenne die Datei `schatzjagd.pbcp`. Nächste Stunde baust du daran weiter.

Das habe ich gelernt:

- ☐ Ich kann eine Funktion anlegen und aufrufen.
- ☐ Ich kann ein Spielfeld mit „Bild auf Canvas“ zeigen.
- ☐ Ich kann mit „berührt Farbe“ Lava erkennen und Leben zählen.

Mehr dazu im Buch: Kapitel 20 – Projekt: Schatzjagd (20.1–20.3)

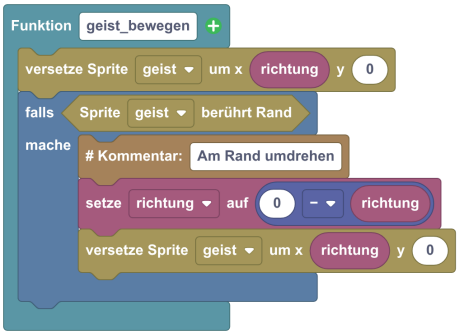
Name: _____ Klasse: _____ Datum: _____

Projekt Schatzjagd 2: Geist und Schätze

Ziel: Ein Geist macht das Spielfeld unsicher, und fünf Schätze warten auf dich. Die Plätze der Schätze speicherst du in zwei **Listen**.

Vorbereitung: Öffne deine Datei `schatzjagd.pbcp` (**Dateien → Vom Computer…**). Fehlen Bilder oder Sprites, lege sie wie in Teil 1 an.

Aufgabe 1 ⭐ Der Geist

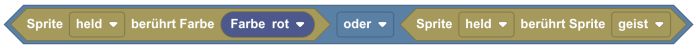


1. Lege die Variable `richtung` an.
2. Hänge in `vorbereiten` unter „setze Sprite held …“ an:



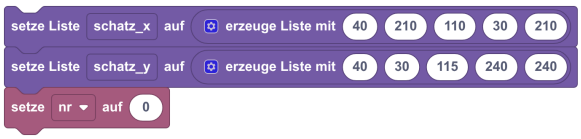
3. Lege die Funktion `geist_bewegen` an (links).
4. Rufe sie in der Schleife unter `held_steuern` auf.

5. Der Geist soll gefährlich sein. Ändere in `gefahr_pruefen` die Bedingung mit „oder“ (**Mathe**). Holst du „Sprite … berührt Sprite …“, steht in beiden Feldern dasselbe Sprite: Stelle das zweite auf `geist` um.



Aufgabe 2 ⭐ Wo liegen die Schätze?

1. Lege unter **Listen** mit „+ neue Liste…“ die Listen `schatz_x` und `schatz_y` an, unter **Variablen** die Variable `nr`.
2. Baue in `vorbereiten` unter „setze leben auf 3“ ein (Block „Liste aus Werten“; weitere Plätze bekommst du mit dem Zahnrad ⚙):

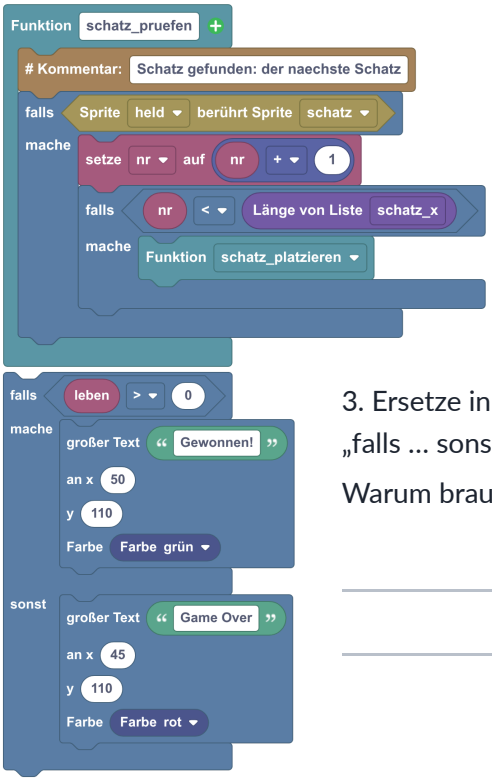


3. Lege die Funktion `schatz_platzieren` an:



4. Rufe in `vorbereiten` am Ende `schatz_platzieren` auf und zeige das Sprite `schatz`.

Aufgabe 3 ⭐⭐ Schatz gefunden!



1. Lege die Funktion `schatz_pruefen` an (links). Rufe sie in der Schleife vor „warte 20 ms“ auf.
2. Die Schleife läuft jetzt, solange Schätze übrig sind **und** Leben da sind:



3. Ersetze in `ende_anzeigen` den Block „großer Text Game Over“ durch das „falls … sonst“ links. Teste dein Spiel!

Warum braucht `schatz_pruefen` das innere „falls nr < Länge …“?

_____

_____

Aufgabe 4 ⭐⭐⭐ Deine Schatzkarte

Denk dir eigene Plätze für die Schätze aus – und einen sechsten Schatz dazu. Liegt keiner in der Lava? Muss die Schleife geändert werden?



Nachgedacht: Listen zählen ab 0. Wo liegt Schatz Nr. 3? x = _____ y = _____

Speichern! Dateien → Auf den Computer… als `schatzjagd.pbcp`.

Das habe ich gelernt:

☐ Ich kann Bedingungen mit „und“ und „oder“ verbinden.

☐ Ich kann Positionen in zwei Listen speichern und mit einer Nummer abrufen.

☐ Ich weiß, dass Listen ab 0 zählen.

Mehr dazu im Buch: Kapitel 20 – Projekt: Schatzjagd (20.3–20.4)

Name: _____ Klasse: _____ Datum: _____

Projekt Schatzjagd 3: Zeiten und dein eigenes Spiel

Ziel: Dein Spiel stoppt die Zeit für jeden Schatz und zeigt am Ende alle Zeiten an. Dann machst du die Schatzjagd zu **deinem** Spiel – mit einer eigenen Erweiterung.

Vorbereitung: Öffne deine Datei `schatzjagd.pbcp` (**Dateien → Vom Computer…**) und teste kurz, ob alles noch läuft.

Aufgabe 1 ⭐ Die Stoppuhr

1. Lege die Liste `zeiten` und die Variable `start` an.
2. Hänge an das **Ende** von `vorbereiten` an:



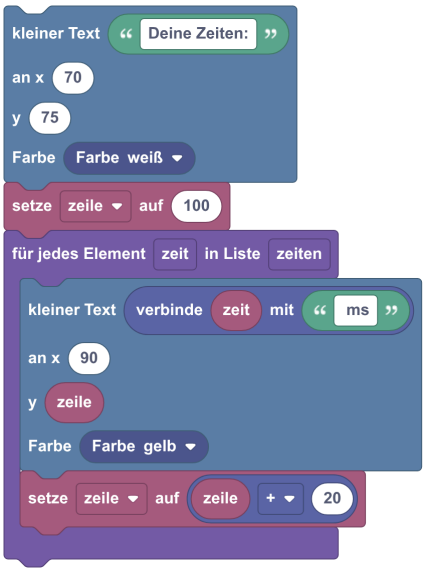
„setze Liste … auf“ ist schon leer, wenn du ihn holst.

3. Baue in `schatz_pruefen` diese zwei Blöcke **als erste** in das „falls … berührt Sprite schatz“ ein:



Was passiert hier? Ergänze: Wenn der Held einen Schatz findet, wird die _____ an die Liste `zeiten` angehängt, und `start` beginnt _____.

Aufgabe 2 ⭐ Alle Zeiten anzeigen



1. Lege die Variablen `zeile` und `zeit` an.
2. Stelle in `ende_anzeigen` bei „Gewonnen!“ y auf **30**.
3. Hänge die Blöcke links **unter** „Gewonnen!“ an – also noch in den „mache“-Teil von „falls leben > 0“.



„für jedes Element“ findest du unter **Listen**. Wähle darin `zeit` und `zeiten` aus. Bei jedem Durchlauf steht in `zeit` die nächste Zeit.

Spiele und trage deine Zeiten ein:

0	1	2	3	4

Aufgabe 3 ⭐⭐ Gesamtzeit

Unter den Zeiten soll „Gesamt: … ms“ stehen. Lege dafür die Variable `summe` an. Sie startet bei 0. Eine zweite „für jedes Element“-Schleife zählt jede Zeit dazu. Die Zeile kommt an die Stelle `zeile`.

Aufgabe 4 ⭐⭐ Mein eigenes Spiel

Wähle mindestens eine Erweiterung aus (ankreuzen) und baue sie ein:

☐ eigene Schatzkarte: andere Plätze, mehr Schätze

☐ Töne: beim Schatz, bei der Lava, beim Sieg

☐ Leben-Anzeige mit den drei Status-LEDs

☐ Der Geist wird nach jedem Schatz schneller.

☐ Ein zweiter Geist schwebt auf und ab.

☐ eigenes Spielfeld mit **Dateien → Bild zeichnen…** (Lava in „Farbe rot“!)

☐ eigene Idee: _____

Welche Funktion(en) musst du dafür ändern? _____

Aufgabe 5 ⭐ Spieltest

Tauscht die Plätze: Spielt das Spiel eures Nachbarn. Schreibt auf, was euch gefällt und was ihr ändern würdet.

_____

_____

Speichern! Dateien → Auf den Computer… – und wer einen PBC+ hat: **Dateien → Auf den PBC+…** mit der Wahl **Python-Programm**, dann kannst du dein Spiel überall spielen.

Das habe ich gelernt:

☐ Ich kann mit „hänge … an Liste an“ Werte sammeln.

☐ Ich kann mit „für jedes Element“ alle Werte einer Liste anzeigen.

☐ Ich kann ein Programm mit Funktionen gezielt erweitern.

Mehr dazu im Buch: Kapitel 20 – Projekt: Schatzjagd (20.5–20.6)

Name: _____ Klasse: _____ Datum: _____

Projekt: Mein eigenes Spiel

Ziel: Du entwickelst ein eigenes Spiel für den PicoBoy – von der Idee über den Plan bis zum fertigen Programm. Du baust es in Stufen, testest es sorgfältig und stellst es am Ende der Klasse vor.

So läuft das Projekt ab

Schritt	Was tust du?	Seite
1. Idee und Plan	Spielidee finden, Planungsbogen ausfüllen, mit der Lehrkraft besprechen	1–2
2. Bauen und testen	in Stufen bauen, nach jeder Stufe testen und ins Logbuch schreiben	3
3. Vorstellen	Spiel der Klasse zeigen, Rückmeldung geben und bekommen	4

Aufgabe 1 ⭐ Eine Idee finden

Wähle eine Idee aus der Ideenbörse – oder erfinde eine eigene. Wichtig: Das Spiel soll **klein** sein. Ein kleines Spiel, das läuft, ist besser als ein großes, das nicht fertig wird.

Idee	Darum geht es	Wichtige Blöcke
Fang den Apfel	Ein Korb fängt fallende Äpfel.	Sprites, berührt Sprite, Zufallszahl
Meteoritenregen	Ein Raumschiff weicht Meteoriten aus.	Sprites, berührt Sprite, LEDs
Lava-Labyrinth	Eine Figur läuft durch ein gemaltes Labyrinth ins Ziel.	Canvas, berührt Farbe
Reaktionstest	Wer drückt am schnellsten, wenn die LED angeht?	Zeit, Zufallszahl, Tasten
Simon sagt	Eine Farbfolge merken und nachspielen	Listen, Töne, LEDs
Kipp-Kugel	Eine Kugel durch Kippen des Geräts ins Loch rollen	Beschleunigung, Sprites

Meine Idee: _____

Das gehört in jedes Spiel

- mindestens **eine Figur**, die man steuern kann
- mindestens **zwei Regeln** („Wenn …, dann …“), z. B. Punkte oder Treffer
- eine **Schleife**, in der die Regeln immer wieder geprüft werden
- ein **Ende**: gewonnen, verloren oder ein Punktestand
- mindestens **zwei eigene Funktionen** mit sprechenden Namen

Aufgabe 2 ⭐ Der Planungsbogen

Name des Spiels	
Worum geht es? 1–2 Sätze	
Figuren (Sprites) Name und Bild	
Steuerung Welche Taste macht was?	
Wie endet das Spiel? gewonnen / verloren	
Variablen Name und Startwert	

Die Regeln: Wenn ..., dann... (mindestens 4)

Wenn ...	dann ...

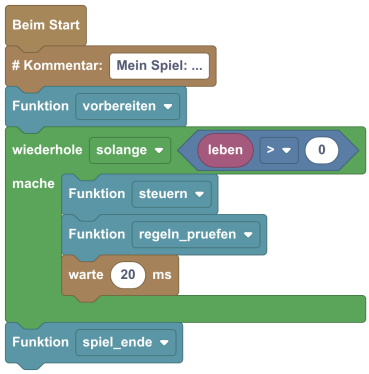
Skizze des Bildschirms

Funktionen sprechende Namen ohne Umlaute	
Muss ohne das geht es nicht	
Kann wenn noch Zeit ist	

Tipp: Ein ausgefülltes Beispiel findest du im Buch, Kapitel 21.1.

☐ Plan mit der Lehrkraft besprochen Unterschrift: _____

Aufgabe 3 ⭐ Bauen in Stufen – mit Bau-Logbuch



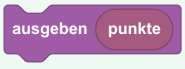
Beginne mit dem Hauptprogramm und leeren Funktionen – so wie links. (Deine Lehrkraft kann dir dieses Gerüst als Datei geben.) Baue dann **eine Stufe nach der anderen**: zuerst die Steuerung, dann die erste Regel, dann die nächste …

Nach **jeder** Stufe: testen, ins Logbuch schreiben, **speichern** – am besten als neue Datei (`spiel_stufe1.pbcp`, `spiel_stufe2.pbcp` …).

Datum	Was habe ich gebaut?	Was soll passieren?	Was passiert?	ok?

Ein Fehler, den ich gefunden habe – und wie ich ihn behoben habe:

Wenn etwas nicht klappt: ☐ Hängt der Block wirklich im Programm (unter „Beim Start“ oder in einer Funktion)? ☐ Steht der „falls“-Block in der Schleife? ☐ Wird die Funktion irgendwo aufgerufen? ☐ Steht „warte 20 ms“ in der Schleife? ☐ Meldung lesen – Erklärungen im Buch, Anhang B. ☐ Werte im Terminal anzeigen lassen:



Aufgabe 4 ⭐⭐ Testen lassen

Gib dein Spiel jemandem aus der Klasse. Sag nichts – schau nur zu. Was hat die Person anders gemacht, als du dachtest? Was hast du danach verbessert?

Aufgabe 5 ⭐ Vorstellen (etwa 3 Minuten)

1. Sag, wie dein Spiel heißt und worum es geht.
2. Zeig es im Emulator oder auf dem PicoBoy.
3. Zeig dein Hauptprogramm und **eine** Funktion, auf die du stolz bist.
4. Erzähl von einem Fehler, den du gefunden hast.

Aufgabe 6 ⭐ Rückmeldung: Zwei Sterne und ein Wunsch

Für das Spiel von: _____

⭐	
⭐	
Wunsch	

Aufgabe 7 ⭐⭐⭐ Version 2

Baue etwas aus deiner **Kann**-Liste ein oder setze den Wunsch aus der Rückmeldung um. Vergiss das Logbuch nicht.

Rückblick: Was ist an deinem Spiel richtig gut geworden? Was würdest du beim nächsten Mal anders machen?

Das habe ich gelernt:

☐ Ich kann ein Spiel mit einem Planungsbogen planen.

☐ Ich kann ein Programm in Funktionen aufteilen.

☐ Ich baue in Stufen und teste nach jeder Stufe.

☐ Ich kann einen Fehler systematisch suchen und beheben.

Mehr dazu im Buch: Kapitel 21 – Ein eigenes Spiel entwickeln · Anhang B – Fehlermeldungen verstehen