Ein Zeichenprogramm mit *Snap!*[1]- Eigene Blöcke

Eigene Blöcke erzeugen

Snap! bietet uns eine Reihe von fertigen Bausteinen an. Wir können aber auch unsere selbst geschriebenen Skripte zu einem eigenen Baustein (Block) zusammenfassen. Das hat den Vorteil, dass wir sie einfach wiederverwenden können und unsere Skripte kürzer und damit übersichtlicher werden.

Um einen eigenen Baustein zu erzeugen, klicken wir im Bereich **Variables** auf *Make a block*. Es öffnet sich das Fenster, das Sie in Abbildung 1 sehen. Damit wir unsere Blöcke leicht finden, ordnen wir sie immer der Kategorie *Other* zu. In das leere Textfeld geben wir den Namen unseres Blocks ein, hier z. B. *zeichneQuadrat*.

Abbildung 1: Erzeugen eines eigenen Blocks

Es werden drei Arten von Blöcken unterschieden:

- **Command**: Ein solcher Block enthält eine Abfolge von Anweisungen, die von dem Sprite ausgeführt werden.
- **Reporter**: Ein solcher Block liefert zusätzlich nach der Ausführung einen Ergebniswert zurück.
- **Predicate**: Ein solcher Block liefert nach der Ausführung einen Wahrheitswert zurück.

Da zum Zeichnen eines Quadrats nur Anweisungen ausgeführt werden müssen, wählen wir den Typ *Command*. Schließlich können wir noch angeben, ob alle Sprites diesen Block verwenden dürfen oder nur der aktuell ausgewählte.

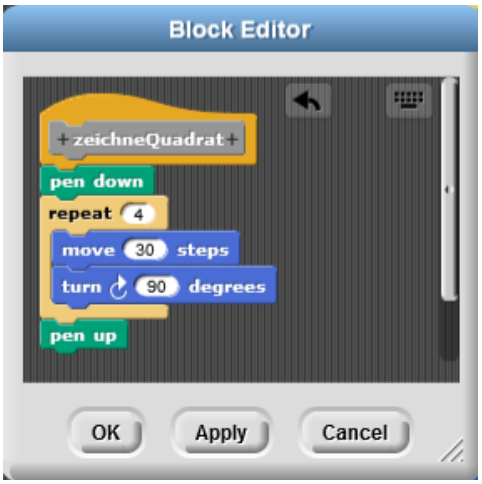

Abbildung 2: Der Block Editor

Nach der Bestätigung mit *ok*, öffnet sich ein weiteres Fenster (s. Abbildung 2). Hier fügen wir nun die Befehlsfolge ein, die in dem Block zusammengefasst werden soll. Haben wir das Skript bereits geschrieben, können wir es einfach hier hineinziehen.

Aufgabe 1: Erstellen Sie für das Zeichnen eines Dreiecks, eines Quadrats und eines Kreises jeweils einen entsprechenden eigenen Block. Ersetzen Sie in Ihren Skripten die entsprechenden Stellen durch diese Blöcke.

Eigene Blöcke mit Parametern

Wir hatten unser Programm bereits so erweitert, dass der Anwender die Größe der Figuren bestimmen kann. Dazu könnten wir den *ask*-Baustein hier mitaufnehmen oder in dem *move*-Baustein die Variable verwenden, in der wir die entsprechende Größe gespeichert haben. Diese Lösung hat jedoch einen Nachteil. Vielleicht möchten wir unseren *zeichneQuadrat*-Baustein später einmal in einem anderen Programm verwenden. Man kann nämlich auch einzelne Blöcke exportieren und in ein anderes *Snap!*-

[1] Snap*!* wird von der University of California, Berkeley zur Verfügung gestellt: https://snap.berkeley.edu

Projekt importieren. Wir wissen aber noch gar nicht, wie dort die Variable heißt, in der wir den Wert speichern, und ob wir den Anwender nach der Größe fragen oder sie anderweitig bestimmen. Daher wäre es günstig, wenn wir unserem Block die gewünschte Größe beim Aufruf mitteilen könnten. Wie das geht, schauen wir uns jetzt an.

Im Block Editor sehen wir vor und hinter dem Namen des Blocks kleine Pluszeichen. Wenn wir sie anklicken, können wir an dieser Stelle ein Eingabefeld für den Block hinzufügen. Es öffnet sich ein Fenster, in dem wir den Namen des Eingabefeldes eingeben können, hier z. B. *breite* (*s.* Abbildung 3). Nach Bestätigen mit *ok* wird in der Hutkachel eine Variable mit diesem Namen angezeigt. Diese Variable enthält den Wert, der beim Aufruf des Blocks eingetragen wird. Für den Aufruf 

, würde die Variable `breite` z. B. den Wert 40 enthalten. Eine solche Eingabe für einen Block nennt man auch Parameter.

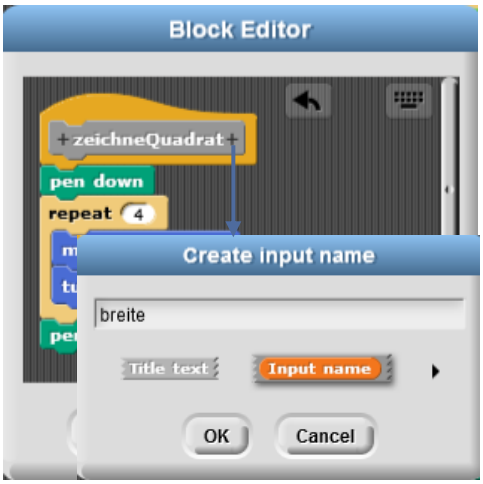

Abbildung 3: Ergänzen eines Parameters im Block Editor

In unserem Skript können wir den übergebenen Wert in Form der Variablen verwenden. Dazu ziehen wir die Variable aus der Hutkachel einfach an die entsprechende Stelle in unserem Skript. Dort entsteht dann eine passende Kopie (s. Abbildung 4).

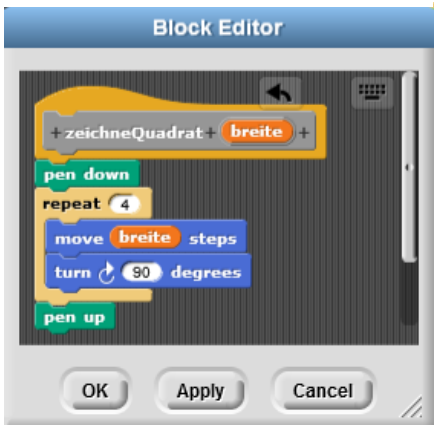

Abbildung 4: Verwenden eines Parameters

Aufgabe 2: Ändern Sie Ihre Blöcke für das Zeichnen des Quadrats und des Dreiecks so ab, dass man ihnen die Größe der Figuren als Parameter übergeben kann. Der jeweilige Parameter muss dann natürlich auch innerhalb des Blocks verwendet werden.

Ergänzen Sie auch einen entsprechenden Block zum Zeichnen des Rechtecks.

Eigene Blöcke mit Rückgabewert

Blöcke vom Typ *Predicate*

Für unser Skript des Stiftes hatten wir uns eine Bedingung zusammengesetzt, die prüft, ob sich der Mauszeiger im Zeichenbereich befindet, wenn die Maustaste gedrückt wird. Für diese Bedingung können wir uns einen *Predicate*-Block erzeugen. Innerhalb unseres Blocks wird die entsprechende Bedingung überprüft. Wenn sie wahr ist, gibt der Block den Wert *true* zurück und ansonsten *false*. Die Rückgabe eines Wertes erfolgt mithilfe des Bausteins *report*.

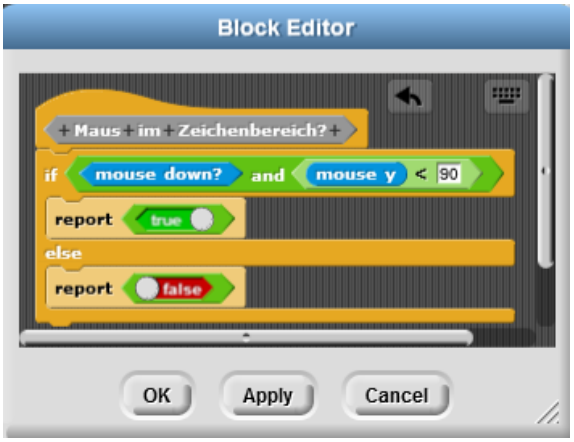

Abbildung 5: Erzeugen eines Predicate-Blocks zum Überprüfen einer Bedingung

Im Skript für den Stift können wir die Bedingung nun durch unseren Baustein ersetzen.

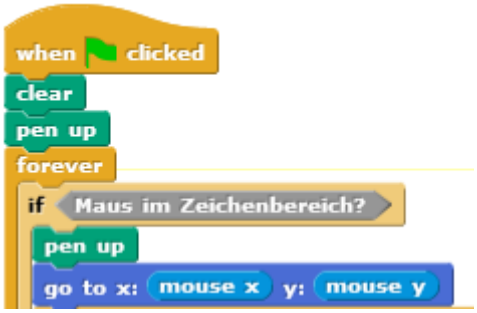

Abbildung 6: Verwenden des selbst erstellten Predicate-Blocks

Aufgabe3:

a) Erzeugen Sie einen *Predicate*-Block wie in Abbildung 5 und setzen Sie ihn in Ihrem Skript für den Stift ein. Passen Sie dabei die zu überprüfende Bedingung an die Gegebenheiten in Ihrem Programm an.

b) Erstellen sie verschiedene Varianten des Blocks aus Abbildung 5. Der gültige Zeichenbereich soll mithilfe von Parametern variabel an den Baustein übergeben werden können. Für das Beispiel aus Abbildung 5 wird nur ein Parameter für den maximalen Wert der y-Koordinate benötigt. Andere Varianten könnten auch eine Begrenzung nach unten oder zu den Seiten erlauben.

Blöcke vom Typ *Reporter*

Ein *Predicate*-Block kann nur die Werte *true* oder *false* zurückgeben. Ein Reporter-Block kann hingegen beliebige Rückgabewerte liefern. Schauen wir uns als Beispiel einen Block *berechneFlaecheRechteck* an, der als Parameter die Breite und die Höhe eines Rechtecks erhält. Der Block berechnet den Flächeninhalt des Rechtsecks und gibt den entsprechenden Wert zurück. Die Rückgabe erfolgt mitilfe des Bausteins *report*.

Abbildung 7: Eigener Block zur Berechnung des Flächeninhalts eines Rechtecks

Aufgabe 4:

a) Erstellen Sie den Baustein *berechneFlaecheRechteck* aus Abbildung 7 und setzen Sie ihn an geeigneter Stelle in Ihrem Zeichenprogramm ein.

b) Erstellen Sie weitere Bausteine für die Berechnung des Flächeninhalts eines gleichseitigen Dreiecks und eines Quadrates.

Skriptvariablen

Wenn unsere Skripte innerhalb eines Blocks komplexer werden, benötigen wir ggf. Hilfsvariablen, die nur innerhalb des Blocks von Bedeutung sind. Deshalb schauen wir uns noch kurz den Baustein *script variables* an.

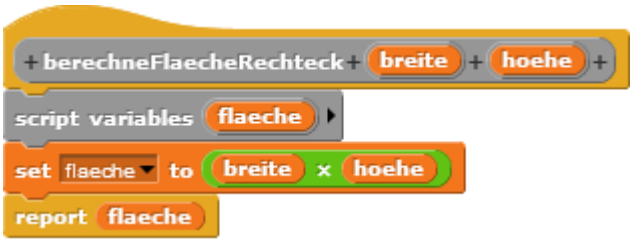

Abbildung 8: Berechnung der Fläche mithilfe einer Skriptvariablen

Mit diesem Baustein können wir eine oder, wenn wir auf den schwarzen Pfeil klicken, auch mehrere Variablen erzeugen, die nur innerhalb des Skriptes verwendet werden können. Für die Skriptvariablen stehen die gleichen Bausteine zur Verfügung wir für andere Variablen. Wenn wir eine Skriptvariable im Programm einsetzen wollen, ziehen wir sie einfach aus dem *script variables*-Baustein an die entsprechende Stelle. Abbildung 8 zeigt, wie der Baustein für die Flächenberechnung des Rechtecks aussehen würde, wenn wir die Fläche zunächst in einer Skriptvariablen speichern.

Aufgabe 5: Im Zusammenhang mit Variablen haben wir gelernt, wir man eine Schnecke zeichnet. Erstellen Sie einen eigenen Block *zeichneSchnecke*. Wählen Sie selbst, ob und ggf. welche Parameter der Block erhält, aber es dürfen keine globalen Variablen verwendet werden. Nutzen Sie innerhalb der Definition des Blocks nur Skriptvariablen.

Aufgabe 6: Untersuchen Sie, ob es in Ihrem Skript weitere Abschnitte gibt, die sich sinnvoll in einem eigenen Block zusammenfassen lassen. Erzeugen Sie ggf. entsprechende Blöcke und setzen Sie diese ein.

Lizenz