SCRATCH

Ein Handbuch für Lehrende

Erstellt im Zuge der Diplomarbeit „Einführung in die Programmierung anhand der visuellen Programmiersprache Scratch“

René Leutgeb

Wien, 2017

Vorwort

Liebe Kolleginnen und Kollegen,

ich habe dieses Einführungshandbuch für Scratch im Zuge meiner Diplomarbeit geschrieben. Ziel des Handbuches ist es, Lehrerinnen und Lehrer mit wenig Scratch Erfahrung eine Hilfestellung für einen Programmier-Einführungsunterricht mit Scratch zu geben und sie auch für den Unterricht mit Scratch zu motivieren.

Das Arbeiten mit Scratch ist für Schülerinnen und Schüler relativ schnell und einfach zu erlernen, was dazu führt, dass die Schülerinnen und Schüler innerhalb kürzester Zeit sichtbare Programmiererfolge haben und dementsprechend motivierter für die Materie sind. Scratch bietet somit eine optimale Möglichkeit den Einstieg in die Programmierung zu gestalten.

Eingehender wird der Zusammenhang zwischen visueller Programmierung und der Motivation von Schülerinnen und Schülern in meiner Diplomarbeit „Einführung in die Programmierung anhand der visuellen Programmiersprache Scratch“ behandelt. Diese ist online abrufbar bzw. findet sich in der Bibliothek der Universität Wien.

Viel Vergnügen mit Scratch!

René ☺

Inhaltsverzeichnis

1 Einleitung

1.1 Was ist Scratch?

Scratch ist eine visuelle Programmiersprache und wurde im Mai 2007 von der „Lifelong Kindergarten Group“ am „MIT (Massachusetts Institute of Technology) Media Lab“ herausgegeben. Bei Scratch handelt es sich um eine imperative Programmiersprache. Ein mit Scratch erstelltes Programm besteht dahingehend aus einer Reihenfolge von Anweisungen, die bestimmen, wie das Programm funktioniert. Diese Anweisungen werden im Gegensatz zur textbasierten Programmierung nicht geschrieben, sondern mittels sogenannter (visueller) Blöcke definiert.

1.2 Warum Scratch?

Wie der Einstieg in die Programmierung möglichst einfach, verständlich und motivierend für Schülerinnen und Schüler gestalten werden kann, ist eine zentrale Fragestellung bei der Gestaltung des Informatikunterrichtes. Schon in den 70iger Jahren beschäftigte sich der amerikanische Erziehungswissenschaftler und Informatiker Seymour Papert mit dem Thema Programmieren und Unterricht. Als Antwort darauf entwickelte er mit seinem Team die visuelle Programmiersprache LOGO, die die Grundlage für Scratch bildet.

Der Grundgedanke hinter Scratch ist Kinder und Jugendliche unter dem Motto „imagine, program and share“ für die Programmierung zu motivieren und sie mit den Grundkonzepten der Programmierung vertraut zu machen.

Durch den visuellen Charakter von Scratch fällt es den Schülerinnen und Schülern einfacher diese Grundkonzepte der Programmierung zu erkennen, zu verstehen und auch anwenden zu können. Nicht zu vernachlässigen ist, dass durch die spielerischen Elemente von Scratch die Schülerinnen und Schüler deutlich motivierter für die Informatik und das Thema der Programmierung sind.

Scratch eignet sich dahingehend optimal für die Einführung in die Programmierung.

1.3 Aufbau des Handbuches

1. Einleitung

   Was ist Scratch und warum sollte Scratch für einen Einführungsunterricht in die Programmierung verwendet werden?

2. Ein mögliches Unterrichtskonzept

   Zu Beginn wird ein mögliches Unterrichtskonzept für die Einführung in die Programmierung vorgestellt. Dieses umfasst 4 Doppelstunden Informatik.

3. Erste Schritte mit Scratch

   Im dritten Punkt werden die unterschiedlichen Versionen / Entwicklungsumgebungen von Scratch behandelt und die nötigen Vorbereitungen für das Unterrichten mit Scratch erklärt.

4. Die Grundlagen zum Arbeiten mit Scratch

   Im Punkt Grundlagen von Scratch werden die zentralen Elemente (Oberfläche, Blöcke, Skripte) der Programmiersprache Scratch vorgestellt und erklärt.

5. Ein erstes Spiel

   Das erste Spiel dient in weiterer Folge als mögliches Einstiegsspiel für den Einführungsunterricht in die Programmierung. Die einzelnen Programmierschritte werden dabei äußerst detailliert beschrieben, um somit auch den Einstieg für Lehrende zu erleichtern.

6. Weitere Spiele für den Unterricht

   Für den Unterricht mit Scratch werden hier weitere Aufgabenstellungen für Schülerinnen und Schüler in ihrer Basisimplementierung vorgestellt. Diese Spiele sind dahingehend konzipiert, dass ihre Grundstruktur innerhalb von zwei Unterrichtseinheiten von Schülerinnen und Schülern programmiert werden kann.

7. Von Scratch zu Python

   In diesem Punkt soll eine Möglichkeit aufgezeigt werden, wie der Übergang von der visuellen Programmierung in die textbasierte Programmierung gestaltet

werden kann. Dazu wird ein Spiel in Scratch und in Python implementiert und verglichen.

8. Quellen und weitere Literatur

   Um die Übersichtlichkeit und den Charakter eines Handbuches zu erhöhen, wurde teilweise darauf verzichtet, Quellen direkt anzugeben. Sofern nicht direkt angegeben, werden diese jedoch im Punkt *Quellen und weitere Literatur* entsprechend angeführt. Ergänzend wird in diesem Kapitel noch weitere und vertiefender Literatur zum Thema Scratch angeführt.

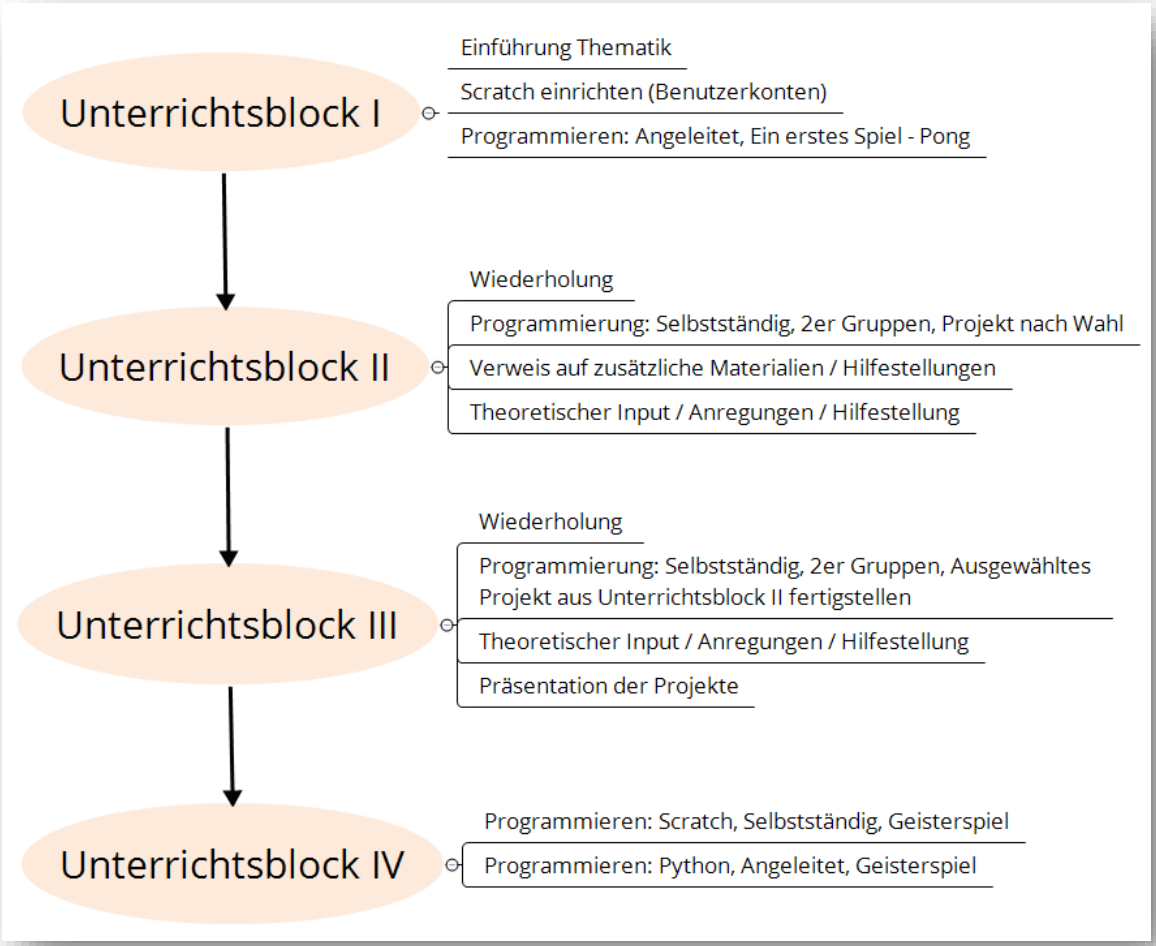
2 Ein mögliches Unterrichtskonzept

2.1 Beschreibung

Im Folgenden soll ein mögliches Unterrichtskonzept für die Einführung in die Programmierung anhand der visuellen Programmiersprache Scratch vorgestellt werden. Das Unterrichtskonzept zielt auf einen kompakten und schnellen Einstieg in die visuelle Programmierung ab. Einerseits soll dieser Einstieg die Schülerinnen und Schüler motivieren und zum anderen auch einen hohen Lernerfolg gewährleisten.

Angelehnt ist das Unterrichtskonzept an den Grundgedanken von Scratch („imagine, program, share“). Die ersten drei Unterrichtsblöcke (je zwei Doppeleinheiten) gliedern sich demnach in drei Schwerpunkte: Einführung und Motivation, Selbstständiges Arbeiten und Präsentation. Im viertem Unterrichtsblock wird ein fließender Übergang von Scratch zu Python vorgenommen.

2.2 Aufbau



2.2.1 Erster Unterrichtsblock

Zu Beginn des ersten Unterrichtsblockes werden die Benutzerkonten mit den Schülerinnen und Schülern erstellt, danach wird eine kurze Einführung in die Thematik und ein Ausblick auf die folgenden Unterrichtseinheiten gegeben. Im Anschluss wird mit den Schülerinnen und Schülern gemeinsam das Spiel Pong (siehe Seite 16) programmiert.

2.2.2 Zweiter Unterrichtsblock

Zu Beginn des zweiten Unterrichtsblockes werden die erarbeiteten Grundelemente von Scratch aus dem Einführungsspiel wiederholt. Zentraler Punkt des zweiten Unterrichtblockes ist der Übergang vom angeleiteten zum selbständigen Lernen. Dazu werden die Schülerinnen und Schüler ein Spiel ihrer Wahl selbständig programmieren.

Dieses Spiel können sie entweder aus den Spielvorgaben auswählen (ab Seite 23) oder eine eigene Idee umsetzen. Für das selbstständige Arbeiten werden den Schülerinnen und Schülern in 2er Gruppen eingeteilt. Während der selbstständigen Arbeitsphase bietet die Lehrkraft Unterstützung an und gibt punktuell theoretischen Input bzw. Anregungen. Im Weiteren werden den Schülerinnen und Schüler Online-Materialien (https://scratch.mit.edu/) zur Verfügung gestellt.

2.2.3 Dritter Unterrichtsblock

Der dritte Unterrichtsblock ist von der Grundstruktur wie der zweite Unterrichtsblock aufgebaut. Zu Beginn wird eine kurze Wiederholung gegeben, aufgetretene Fragen in der Klasse beantwortet/besprochen und danach können die Schülerinnen und Schüler wieder selbstständig an ihren Projekten weiterarbeiten. Den Abschluss des dritten Unterrichtsblockes bildet die Präsentation der einzelnen Projekte.

2.2.4 Vierter Unterrichtsblock

Im letzten Unterrichtsblock wird ein fließender Übergang von der visuellen Programmierung hin zur textbasierten Programmierung vorgenommen. Dazu werden die Schülerinne und Schüler zuerst ein Spiel in Scratch selbstständig programmieren. Im Anschluss daran wird dasselbe Spiel in Python gemeinsam programmiert. (ab Seite 29) Die Schülerinnen und Schüler sollen somit ein erstes Gefühl für die textbasierte Programmierung bekommen und erkennen, dass die Konzepte der Programmierung von der Programmiersprache unabhängig und universell anwendbar sind.

3 Erste Schritte mit Scratch

3.1 Versionen

3.1.1 Scratch 1.4

Scratch in der Version 1.4 wurde am 2. Juli 2009 veröffentlicht und am 9. Mai 2013 durch die Version 2.0 ersetzt. Da in vielen Schulen (und auch an Universitäten) diese ältere Version noch installiert ist, soll diese Version nicht unerwähnt bleiben.

Scratch 1.4 kann nach wie vor unter https://scratch.mit.edu/scratch_1.4/ kostenlos für MAC OS X, Windows und Debian/ Ubuntu heruntergeladen werden. Erstellte Projekte können direkt aus dem Programm auf die Scratch-Homepage hochgeladen werden und ermöglichen so einen Austausch mit der Community. Projekte die mit Scratch 2.0 erstellt wurden können jedoch nicht mit Scratch 1.4 geöffnet werden.

Weiter und vertiefender Informationen zu Scratch 1.4 finden sich unter http://scratch-dach.info/wiki/Scratch_1.4.

3.1.2 Scratch 2.0

Die aktuelle Version von Scratch ist online wie offline nutzbar. Wie für Scratch 1.4 wird eine Offline-Version für MAC OS X, Windows und einige Linux-Distributionen (32 Bit) zum Download unter https://scratch.mit.edu/scratch2download/ angeboten.

Unter dem Punkt 2.2 und 2.3. finden sich detaillierte Anleitungen zur Installation der Offline-Version bzw. zum Arbeiten mit der Online-Version.

Weiter und vertiefender Informationen zu Scratch 2.0 finden sich unter http://scratch-dach.info/wiki/Scratch_2.0.

3.1.3 Scratch 3.0

Die neueste Version von Scratch wird voraussichtlich Ende 2017 in einem Alpha-Release veröffentlicht.

Weiter und vertiefender Informationen zu Scratch 3.0 finden sich unter http://scratch-dach.info/wiki/Scratch_3.0.

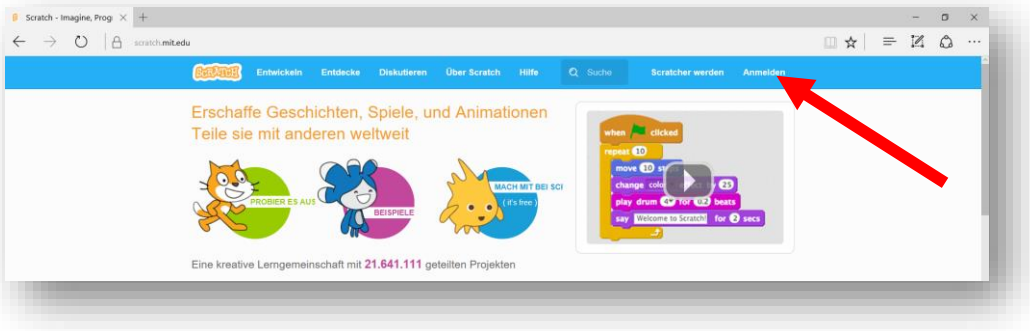
3.2 Scratch 2.0 Online

Um mit Scratch 2.0 online arbeiten zu können, ist es notwendig, dass die Schülerinnen und Schüler ein Scratch-Konto besitzen.

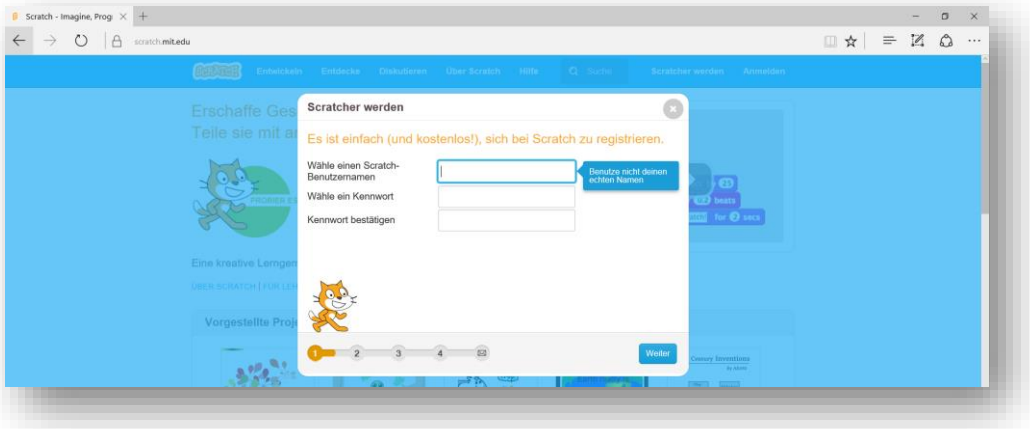
Es gibt zwei Möglichkeiten wie ein Benutzerkonto für die Schülerinnen und Schüler erstellt werden kann. Zum einen selbstständig durch die Schülerinnen und Schüler und zum anderen zentral von der Lehrkraft.

3.2.1 Erstellen des Benutzerkontos durch die Schülerinnen und Schüler

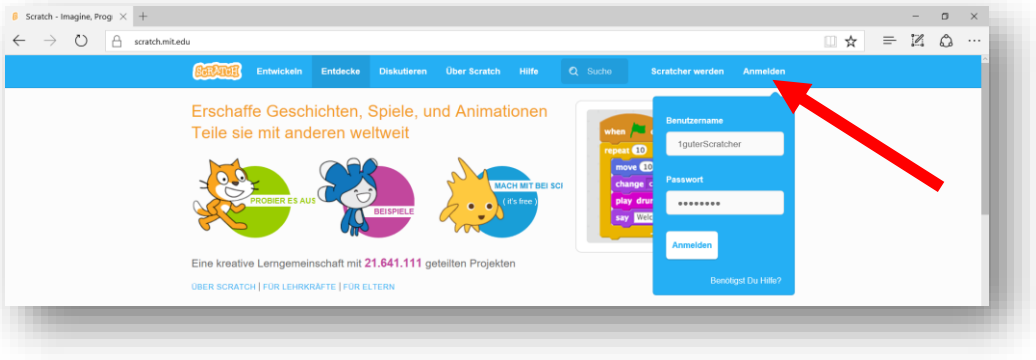
1. Ein Benutzerkonto für Scratch zu erstellen ist auf https://scratch.mit.edu/ unter dem Menüpunkt *Scratcher werden* möglich.



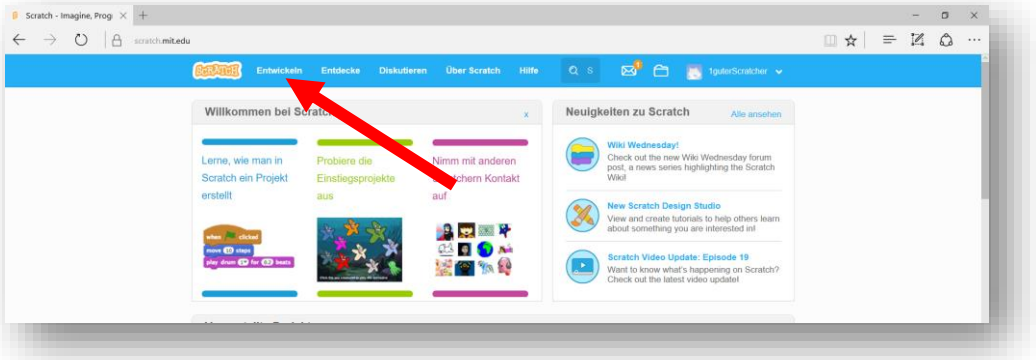
2. Ein neues Konto erstellen. Die Schülerinnen und Schüler müssen zur Erstellung des Kontos eine E-Mail-Adresse angeben. Mittels dieser Adresse wird das Konto verifiziert. Darum ist es unbedingt erforderlich, dass die Schülerinnen und Schüler Zugriff auf ihr Mail-Konto haben.

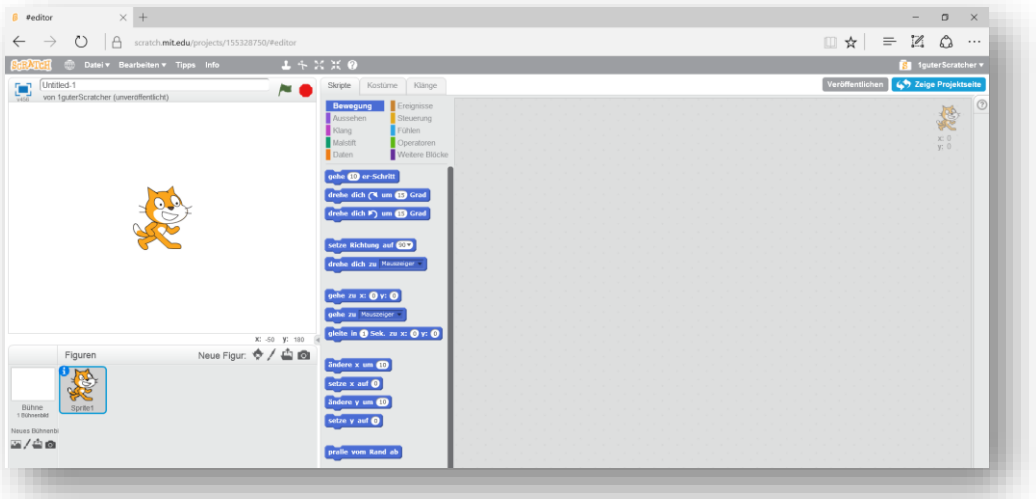


3. Unter dem Punkt *Anmelden* kann man sich mit seinem Benutzernamen auf https://scratch.mit.edu/ anmelden.



4. Nach der erfolgreichen Anmeldung ist es möglich unter dem Menüpunkt *Entwickeln* ein neues Scratch-Projekt zu beginnen bzw. bei einem bestehenden Projekt weiterzuarbeiten.

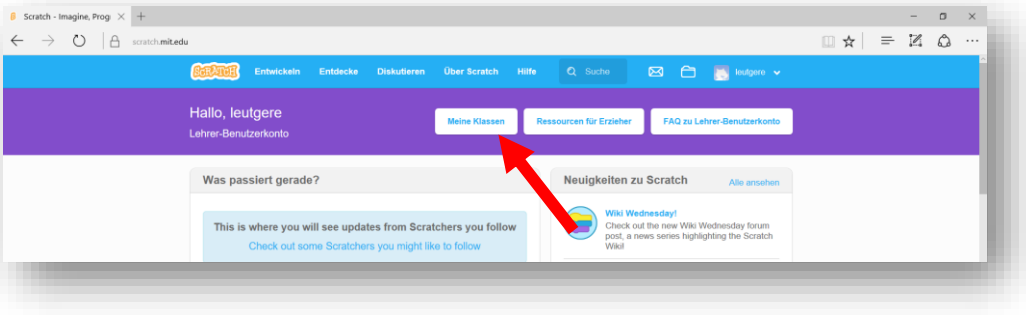




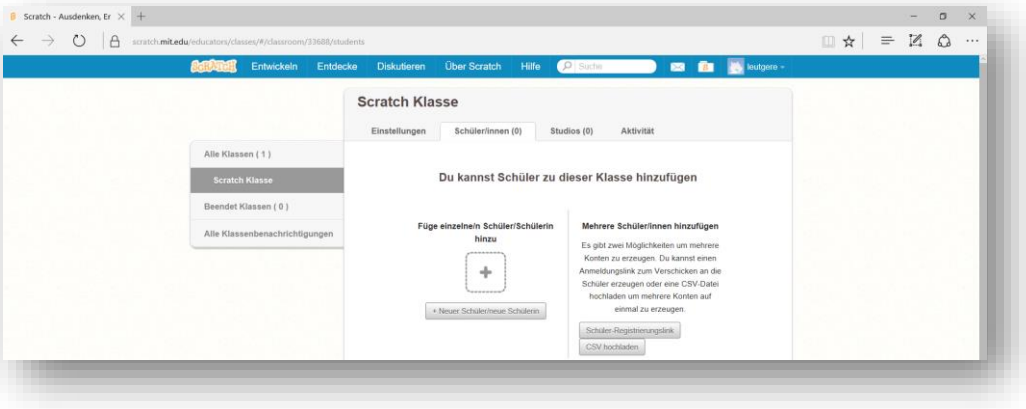
3.2.2 Erstellen der Benutzerkonten durch die Lehrkraft

1. Beantragen eines Lehrerkontos unter https://scratch.mit.edu/educators/register. Die Freischaltung des Kontos kann bis zu 24 Stunden dauern.

2. Bevor Benutzerkonten angelegt werden können, muss eine neue Klasse erstellt werden.



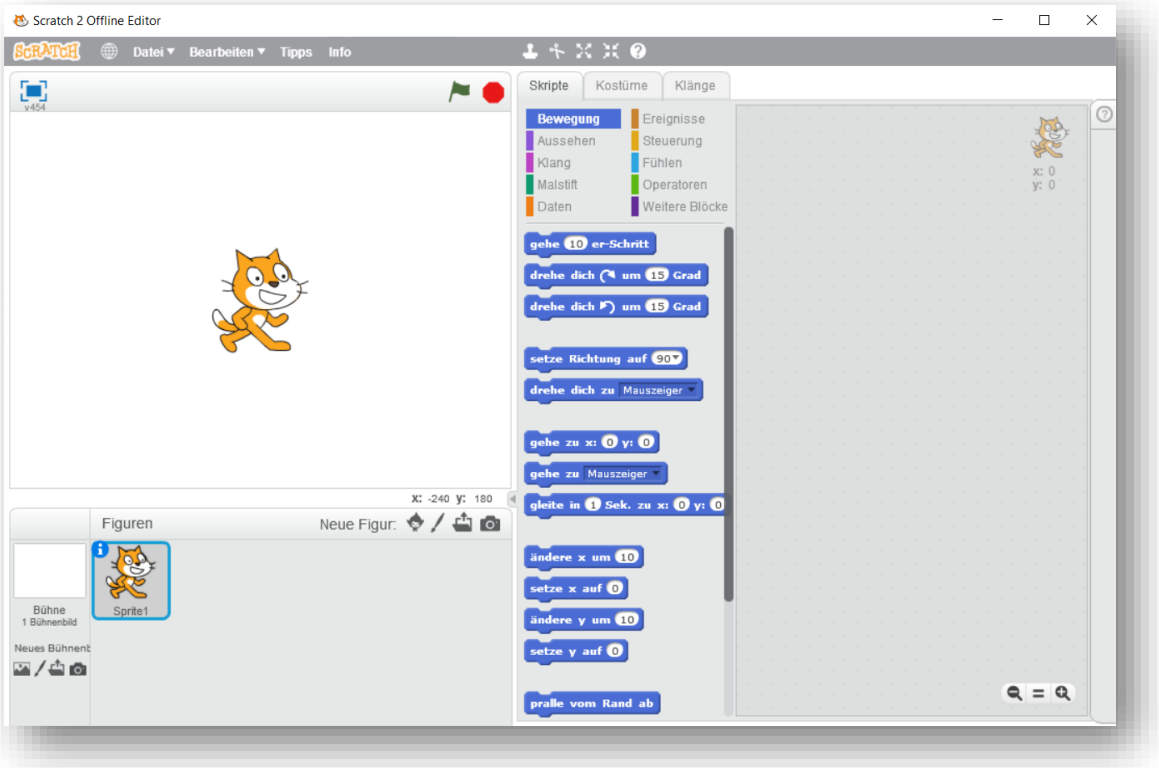
3. Benutzerkonten können auf drei unterschiedliche Arten von der Lehrkraft angelegt werden:
   a. Manuelles hinzufügen
   b. Import einer CSV Datei
   c. Mittels Registrierungslink an die Schülerinnen und Schüler (empfohlen)



4. Unter https://scratch.mit.edu/ können die Schülerinnen und Schüler sich mit ihrem Benutzerkonto anmelden.

3.3 Scratch 2.0 Offline

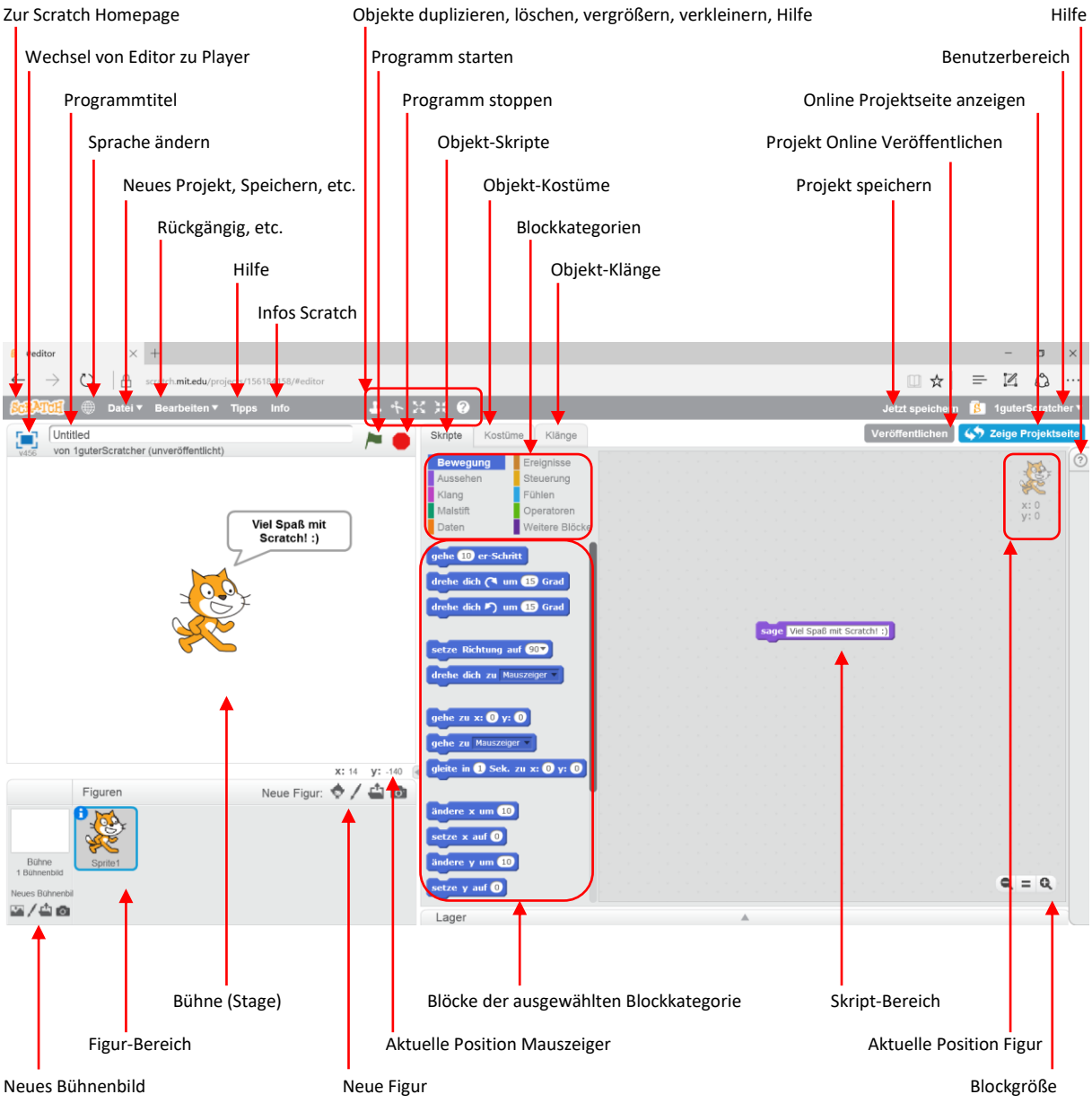
Der Scratch 2.0 Offline Editor kann unter https://scratch.mit.edu/scratch2download/ heruntergeladen werden. Für das Arbeiten mit dem Scratch 2.0 Offline Editor ist keine bestehende Internetverbindung und kein Benutzerkonto notwendig. Ist das Netzwerk in der Schule instabil oder langsam, empfiehlt sich die Installation des Offline Editors. Im Weiteren ist er Schülerinnen und Schülern zu empfehlen, die privat keinen bzw. nur beschränkten Internetzugang haben.



Empfehlung: Arbeite im Unterricht mit Scratch 2.0 in der Online-Version und erstelle die Benutzerkonten mittels eines Lehrer-Accounts (Registrierungslink).

4 Die Grundlagen zum Arbeiten mit Scratch

4.1 Oberfläche



4.2 Bühne

Die Bühne ist der Bereich, in dem ein Scratch-Projekt abgespielt wird. Auf ihr bewegen sich die Figuren und können dort ihre Malspuren hinterlassen. Das Koordinatensystem der Bühne reicht von x = -240 bis x = 240 und y = -180 bis y = 180. Ähnlich wie eine Figur kann die Bühne Skripte, Kostüme (Bühnenbilder) und Klänge enthalten, die man erreicht, wenn keine Figur, sondern das außerhalb des Figuren-Auswahlbereiches gelegenes Bühnen-Icon angewählt wird. Bestimmte Blöcke können in den Skripten der

Bühne nicht verwendet werden, wie z.B. Bewegung oder Größe, da die Bühne eine feste Position und Größe hat. Weil die Bühne die hinterste Ebene des Projekts ist, kann sich keine Figur hinter sie bewegen.

4.3 Figuren

Als Figur wird in Scratch alles bezeichnet, das ein Kostüm (Bild) hat und sich auf der Bühne bewegen kann. Ein Beispiel ist die Katze, die als erste Figur erscheint, wenn man ein neues Projekt erstellt. Die Bühne selber ist einmalig, unbeweglich und immer die Hintergrund-Ebene für alle Figuren, welche sich im Gegensatz dazu auf ihr bewegen, drehen und ihre Ebene ändern können.

4.4 Blöcke

Abbildung 1

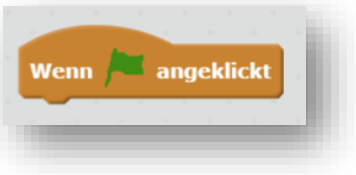
In Scratch werden die Blöcke einerseits durch ihre Farbe kategorisiert und zum anderen durch ihre Form. Die unterschiedlichen Farben bestimmen die jeweilige Blockkategorie. Gesamt gibt zehn verschiedene Blockkategorien (siehe Abbildung 1). Jede Blockkategorie umfasst eine bestimmte Anzahl von Funktionsblöcken. Diese Funktionsblöcke haben unterschiedliche Formen und können wie Puzzleteile zusammengesetzt werden. Die Formen haben den Vorteil, dass sich nur zusammensetzten lässt, was einen syntaktisch korrekten Code ergibt. Somit sind Syntaxfehler von vorhinein ausgeschlossen.

Mehrere zusammengesetzte Blöcke ergeben ein ausführbares Skript. Im Folgenden werden die einzelnen Blockformen und Blockkategorien erklärt. Im Anhang finden sich noch detailliertere Erklärungen zu den einzelnen Blöcken der Blockkategorien.

4.4.1 Blockformen

Kopf-Blöcke

Jedes Skript muss mit einem Kopf-Block beginnen. Der Kopf-Block gibt an, unter welchen Bedingungen das Skript startet.



Klammer-Blöcke

Klammer-Blöcke sind Steuerungs-Blöcke. Sie werden überall da eingesetzt, wo etwas beliebig oft oder nur passieren soll, wenn eine bestimmte Bedingung erfüllt wird.

Wert-Blöcke

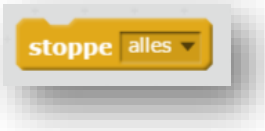
Ein Wert-Block gibt einen Wert zurück. Dieser Wert kann sowohl eine Zahl, als auch eine Zeichenkette (zum Beispiel ein Wort) sein.

Wahrheits-Blöcke

Wahrheits-Blöcke sind eine spezielle Form von Wert-Blöcken, die entweder "wahr" oder "falsch" zurückgeben.

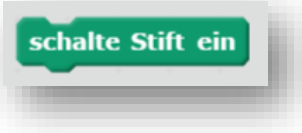
Abschluss-Blöcke

Mit Abschluss-Blöcken kann das Skript oder das ganze Programm beendet werden.



Stapel-Blöcke

Stapel-Blöcke sind nach oben und nach unten geöffnet und können somit beliebig eingesetzt werden. Stapel-Blöcke besitzen unterschiedliche Funktionen und finden sich daher in jeder Blockkategorie (Ausnahme Daten).



4.4.2 Blockkategorien

Bewegung

Die Bewegungs-Blöcke sind dunkelblau und dienen der Bewegung von Objekten.
In Scratch 2.0 umfasst diese Kategorie 14 Stapel-Blöcke und drei Wertblöcke.

Aussehen

Die Aussehen-Blöcke sind lila und steuern das Aussehen einer Figur.
In Scratch 2.0 umfasst diese Kategorie 16 Stapel-Blöcke und drei Wertblöcke.

Klang

Die Klang-Blöcke sind pink und steuern die Klänge, welche eine Figur oder manchmal auch die Bühne abspielt.
In Scratch 2.0 umfasst diese Kategorie 11 Stapel-Blöcke und zwei Wert-Blöcke.

Malstift

Die Malstift-Blöcke sind grün und steuern die Malfunktion in Scratch, mit dem jede Figur Malspuren auf der Bühne hinterlassen kann.
In Scratch 2.0 umfasst diese Kategorie 11 Stapel-Blöcke

Daten

Die Daten-Blöcke sind orange und mittels diesen Blöcken lassen sich Werte wie Zahlen oder Zeichenketten (Strings) speichern, verwalten und anzeigen.
In Scratch 2.0 umfasst diese Kategorie 2 Blöcke: Neue Variable und Neue Liste.

Ereignisse

Die Ereignisse-Blöcke sind braun und beeinflussen, wann Skripte beginnen und wie sie (durch die Nachrichtenblöcke) mit anderen Skripten reagieren.
In Scratch 2.0 umfasst diese Kategorie 8 Kopf-Blöcke.

Steuerung

Die Steuerungs-Blöcke sind gelb und werden verwendet, um Skripte zu steuern.
In Scratch 2.0 umfasst diese Kategorie einen Kopf-Block, fünf Klammer-Blöcke, drei Stapel-Blöcke und zwei Abschluss-Blöcke.

Fühlen

Die Fühlen-Blöcke sind hellblau. Sie können bestimmte Zustände feststellen und messen.
In Scratch 2.0 umfasst diese Kategorie fünf Wahrheitsblöcke, vier Stapel-Blöcke und acht Wert-Blöcke.

Operatoren

Die Operatoren-Blöcke sind grün und können mathematische Funktionen und Textverarbeitung ausführen.
In Scratch 2.0 umfasst diese Kategorie sechs Wahrheits-Blöcke und 11 Wert-Blöcke.

Weitere Blöcke

Weitere Blöcke ist eine spezielle Kategorie von Blöcken, die selbst erstellt wird. Je nachdem kann diese Kategorie also auch leer sein. Alle enthaltenen, violetten Blöcke sind Stapelblöcke und können je nach Wunsch mit verschiedenen Parametern ausgestattet werden.

5 Ein erstes Spiel - Pong

Zielsetzung

Ziel dieses ersten Spieles ist es den Schülerinnen und Schülern einen ersten Einblick in die Welt von Scratch zu geben und sie spielerisch für den folgenden Einstieg in die Programmierung zu motivieren.

Beschreibung

Beim Spiel Pong wird versucht mittels eines Balken zu verhindern, dass der Ball den Boden berührt. Das Spiel beinhaltet drei Level. Je höher die Level, desto schneller bewegt sich der Ball und desto kleiner wird der Balken. Der Aufstieg in das nächste Level wird durch das Erreichen einer bestimmten Zeit ohne Fehler erreicht.

Aufbau

Im Folgenden werden die einzelnen Schritte zur Erstellung des Programmes zuerst grob und dann detailliert beschrieben. Die Reihenfolge der einzelnen Schritte ist so gewählt, dass es für die Schülerinnen und Schüler einfacher ist die Zusammenhänge zu verstehen. Um die Verständlichkeit noch zu erhöhen, wird zu Beginn nur die Grundstruktur des Spieles mit den Schülerinnen und Schülern erstellt und in weiterer Folge erst Erweiterungen vorgenommen.

Link zum Spiel

https://scratch.mit.edu/projects/155913773/

Quellen

https://www.youtube.com/watch?v=LXQfs7qEXfc

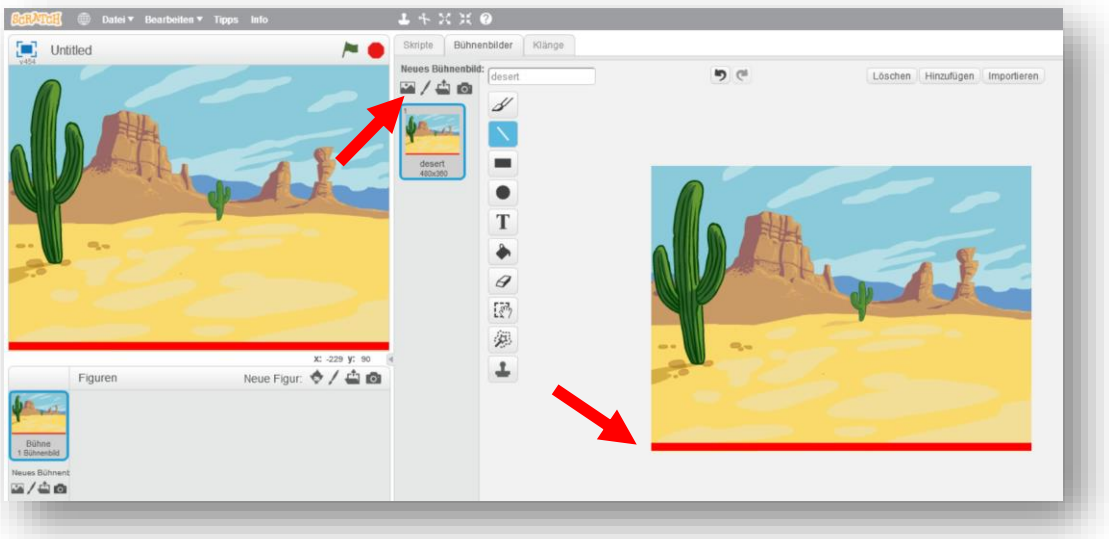
https://www.youtube.com/watch?v=YFYrnaGfaog&t=353s

https://scratch.mit.edu/projects/155871328/#editor

Programmierschritte

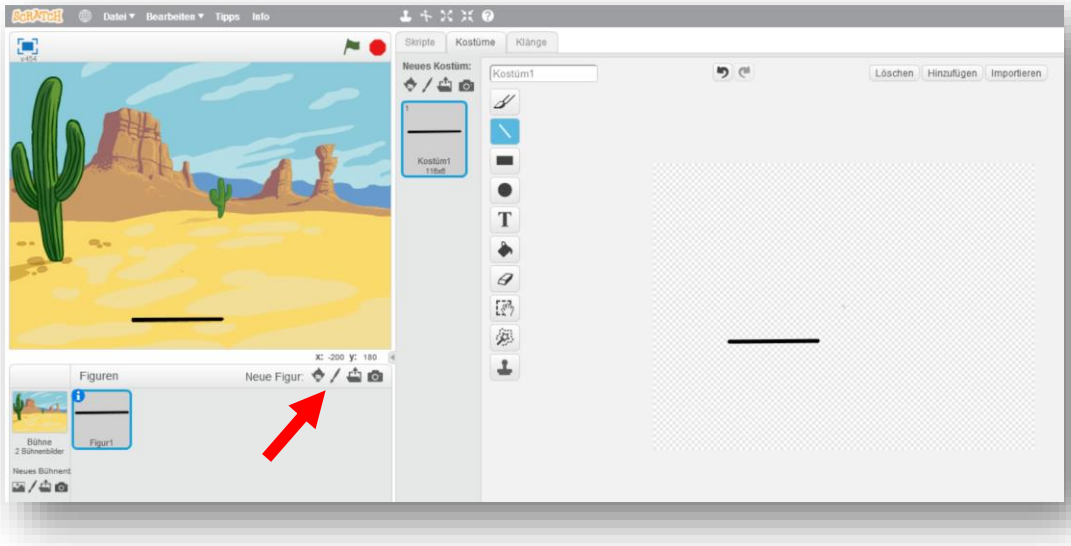
Grundstruktur	1. Bühnenbild erstellen 2. Figur Balken zeichnen 3. Figur Ball zeichnen 4. Skript für Figur Balken schreiben 5. Skript für Figur Ball schreiben
Erweiterungen	6. Bühnenbild für Level 2 und Level 3 erstellen 7. Skript für Bühnenbild schreiben 8. Verkleinern des Balkens bei nächstem Level 9. Verkleinern des Balles und Erhöhung der Geschwindigkeit bei nächstem Level

1. Bühnenbild erstellen

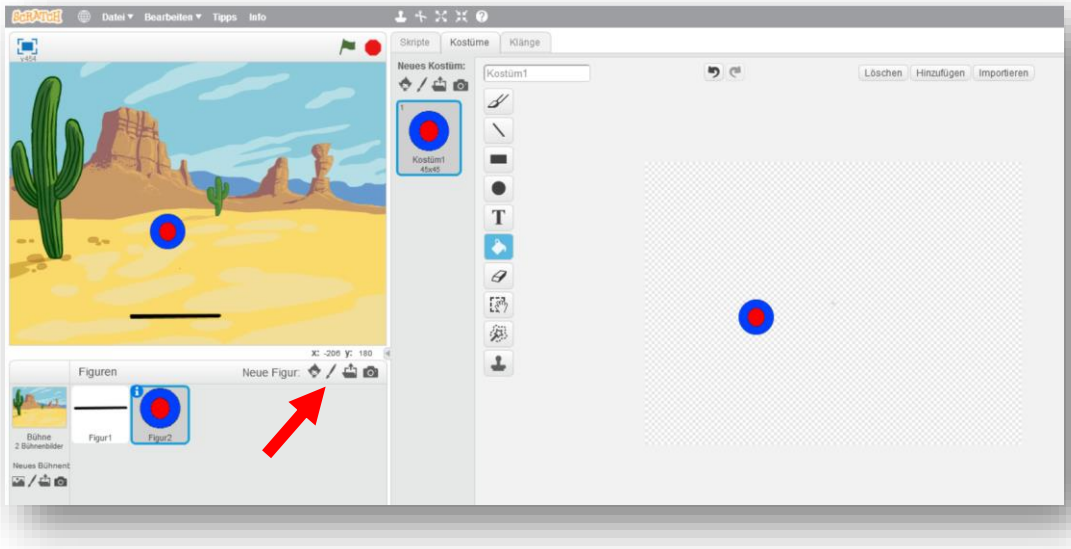


- Rote Grundlinie einzeichnen, diese dient später als Bedingung für das Ende des Spieles.

2. Figur Balken zeichnen



3. Figur Ball zeichnen



4. Skript für Figur Balken schreiben

Die x-Richtung des Balkens wird mit der Mausbewegung gesteuert. Der y-Wert wird zu Beginn definiert.

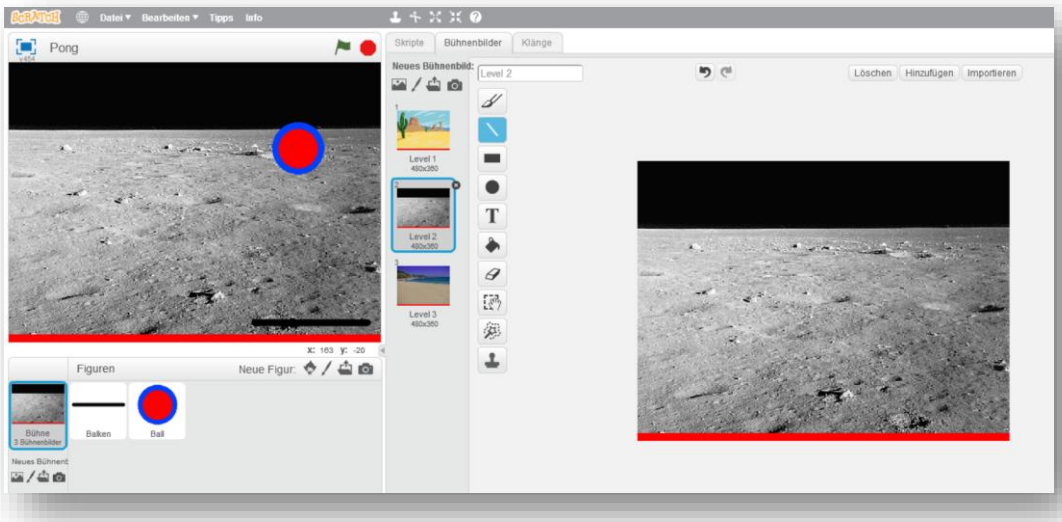
5. Skript Figur Ball schreiben

Startposition und Richtung des Balles setzen. Wenn der Ball den Spielfeldrand berührt, soll er abprallen.

Wenn der Ball den Balken berührt prallt er ab und macht einen Klang. Wichtig: In den Figur-Eigenschaften muss der Drehmodus auf Punkt eingestellt sein.

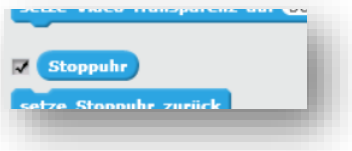
Berührt der Ball den Boden ist das Spiel zu Ende.

6. Bühnenbild für Level 2 und Level 3 erstellen



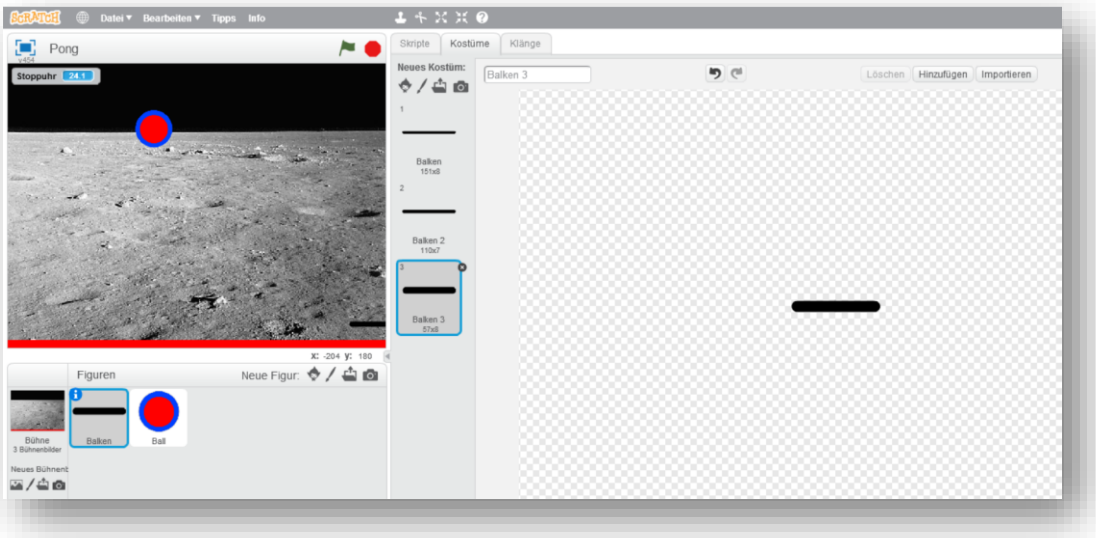
7. Skript für das Bühnenbild schreiben

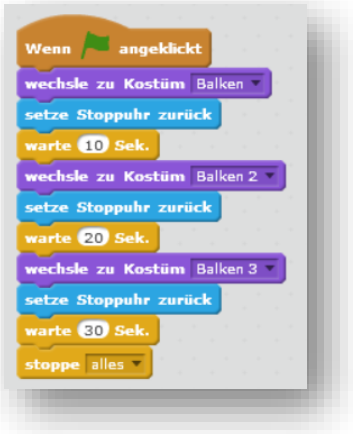
Nach Ablauf der eingestellten Zeit wird auf das nächste Bühnenbild bzw. auf das nächste Level umgeschaltet.



Zum Anzeigen der Stoppuhr auf dem Bühnenbild muss diese unter der Blockkategorie Fühlen aktiviert werden.

8. Verkleinern des Balkens bei nächstem Level





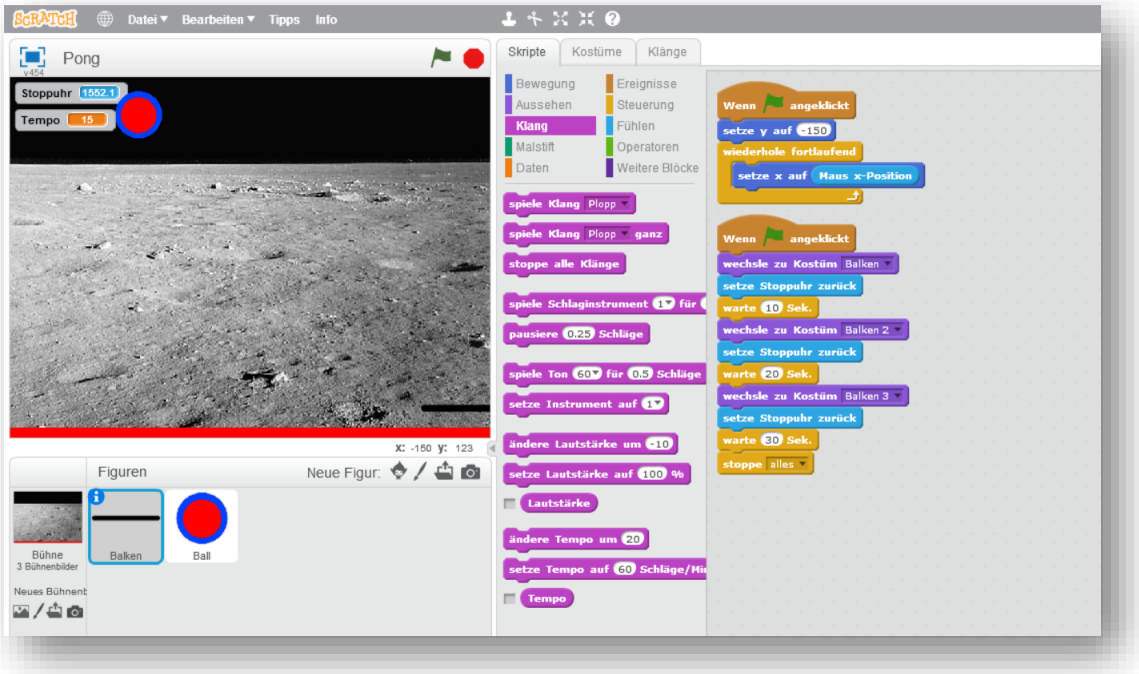
Das selbe Schema wie bei der Veränderung des Bühnenbildes. Eine andere Möglichkeit Objekte zu verkleinern wird beim nächsten Punkt (Verkleinern Ball) gezeigt.

9. Verkleinern des Balles und Erhöhung der Geschwindigkeit bei nächstem Level

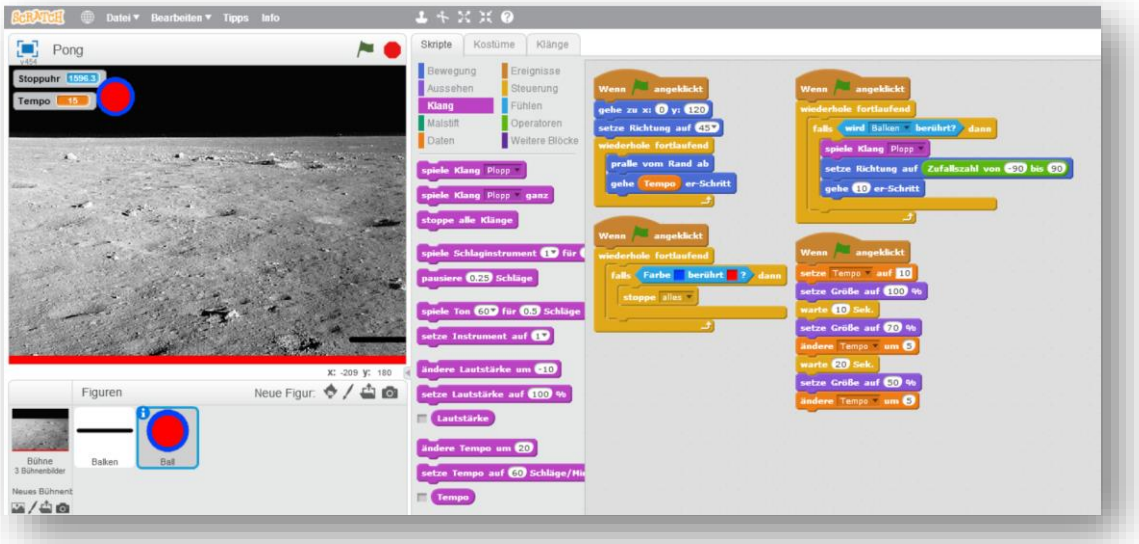
Die Größe lässt sich auch mit dem Block *setze Größe auf x %* verändern.
Zu Beginn wird das Tempo mit 10 definiert und dann jeweils um 5 geändert.

Die Variable Tempo wird eingeführt. Diese muss überall entsprechend eingesetzt werden.

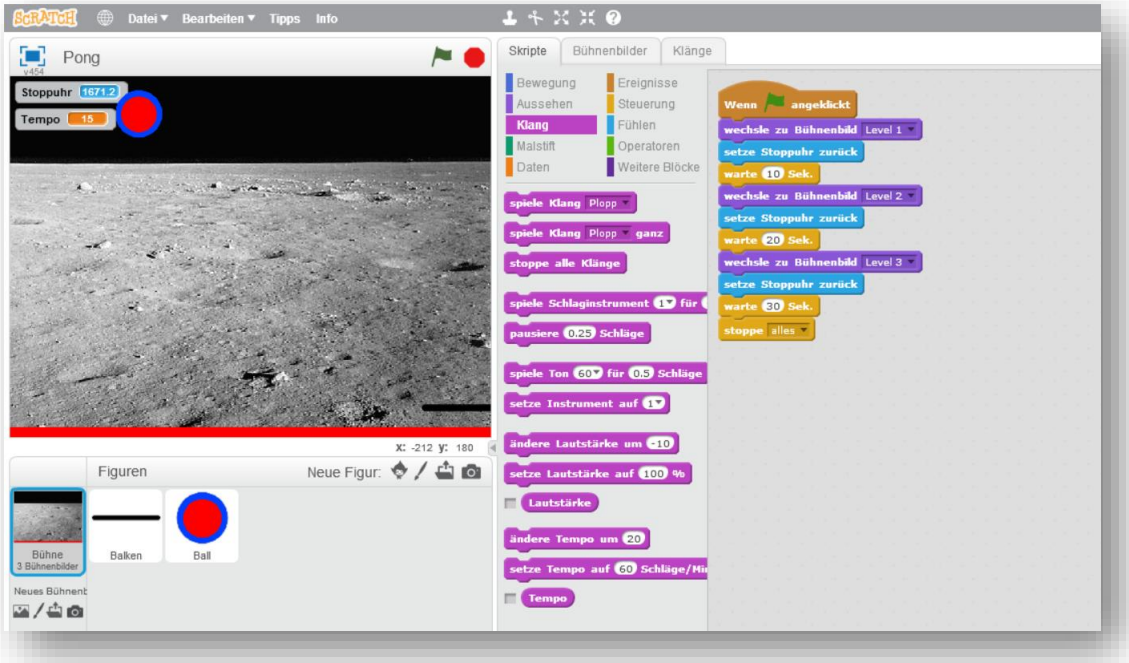
Programmcode



Skript 1 - Balken



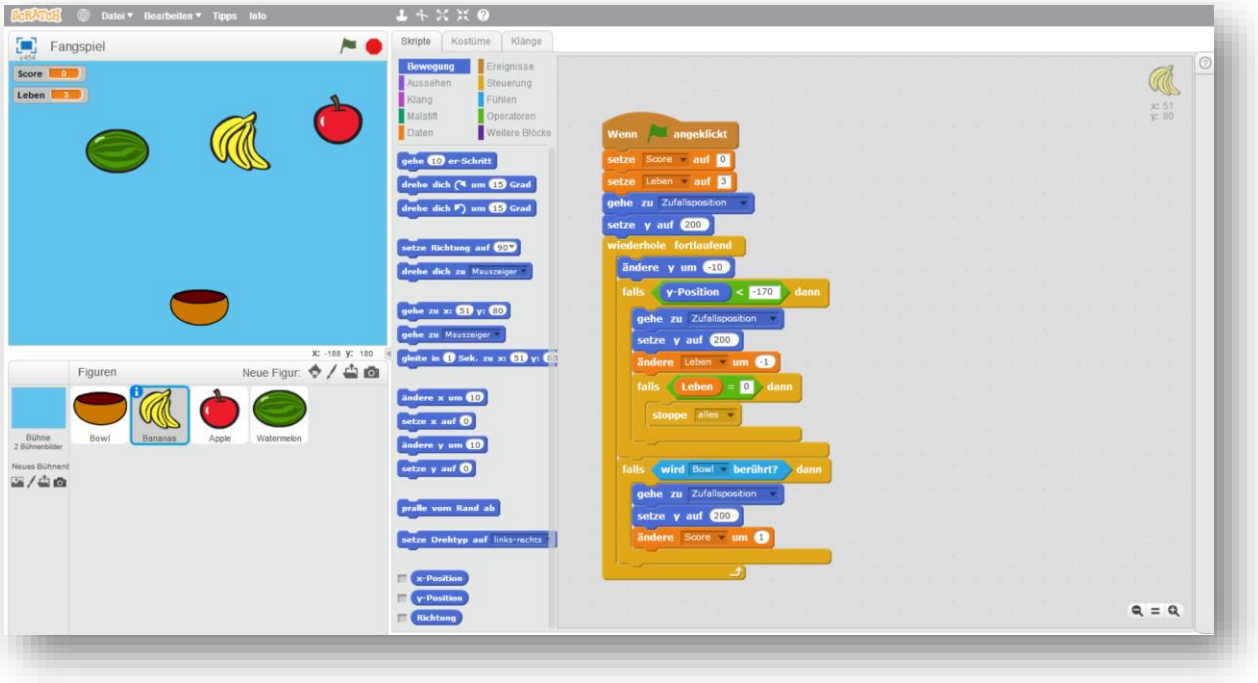
Skript 2 – Ball



Skript 3 - Bühne

6 Weitere Spiele für den Unterricht - Aufgabenstellungen

6.1 Fangspiel



Aufgabenstellung

Programmiere zuerst das Spiel in der Basis-Implementierung. Orientiere dich dabei an den vorgegebenen Programmierschritten. Erweitere dann dein Spiel. Nutze dazu die Ideen für Erweiterungen oder lass dir selbst etwas einfallen.

Beschreibung

Das Ziel bei diesem Spiel ist es das fallende Obst mit einer Schale aufzufangen. Für jedes Obststück das gefangen wird, erhöht sich die Variable Score um 1. Für jedes Obststück das nicht gefangen werden kann, verringert sich die Variable Leben um 1. Zu Beginn wird die Variable Leben auf einen beliebigen Wert gesetzt. Erreicht dieser Wert im Laufe des Spiels den Wert 0, ist das Spiel zu Ende.

Link zum Spiel

https://scratch.mit.edu/projects/156279942/

Programmierschritte

1. Neues Bühnenbild erstellen
2. Neue Figuren erstellen
3. Skript für die Schale
4. Skript für die fallenden Oststücke

Basis – Implementierung

Skript Schale

Die X-Position der Schale wird mit der Mausbewegung gesteuert.

Skript Figuren Oststücke

Ausgangssituation: Variable Score wird auf 0 gesetzt, Variable Leben wird auf 3 gesetzt und die Starposition für die Figur definiert.

Schleife: Obst fällt

Bedingung 1: Wenn das fallende Obst den Boden berührt wird Leben um 1 reduziert.

Bedingung 2: Falls Leben gleich 0, ist das Spiel vorbei.

Bedingung 3: Wenn das Obst die Schale berührt, erhöht sich der Score um 1 und Obst starte wieder von oben.

Ideen für Erweiterungen

Soundeffekte einfügen

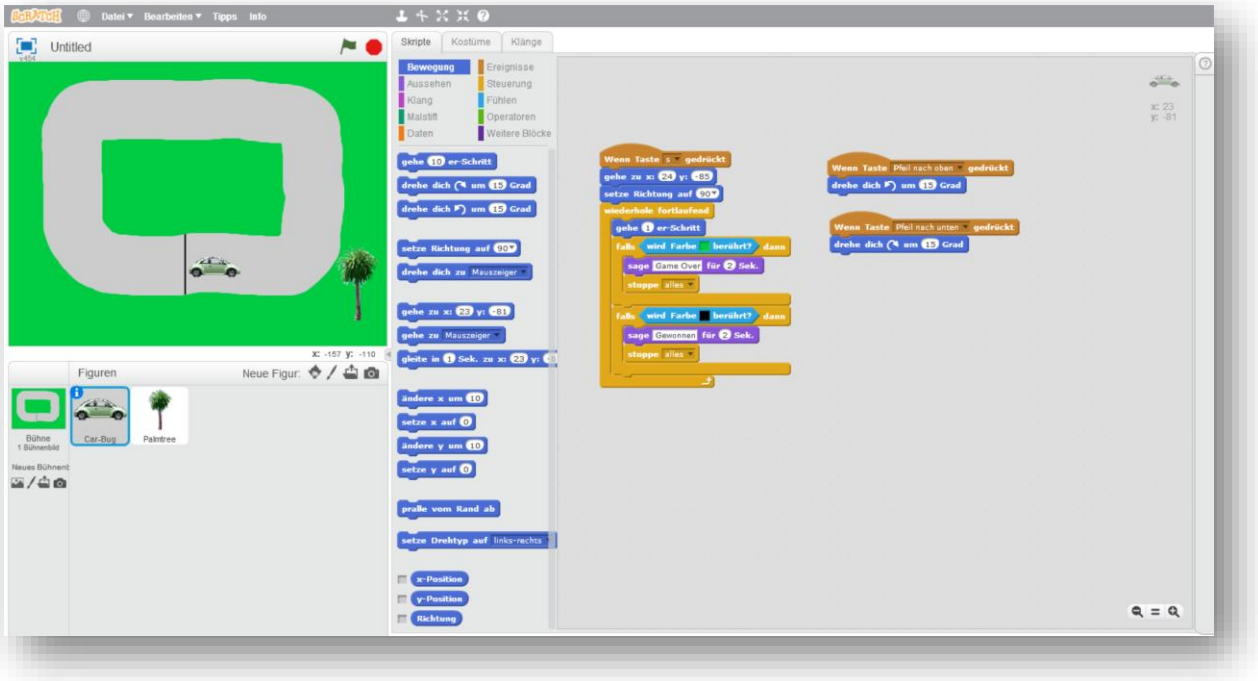
Weitere Bühnenbilder (Level)

Veränderung der Geschwindigkeit

Weiter Figuren mit Funktion (z.B.: Herz für +1 Leben)

Steuerung der Schale mit der Tastatur

6.2 Rennspiel



Aufgabenstellung

Programmiere zuerst das Spiel in der Basis-Implementierung. Orientiere dich dabei an den vorgegebenen Programmierschritten. Erweitere dann dein Spiel. Nutze dazu die Ideen für Erweiterungen oder lass dir selbst etwas einfallen.

Beschreibung

Ziel dieses Rennspieles ist es die graue Rennstrecke abzufahren ohne in die grüne Wiese zu fahren. Gesteuert wird das Rennauto mit den Pfeiltasten. Berührt das Rennauto die schwarze Ziellinie, ist das Spiel gewonnen. Das Spiel wird mit der Taste s gestartet.

Link zum Spiel

https://scratch.mit.edu/projects/158155907/

Programmierschritte

1. Neues Bühnenbild zeichnen (Hintergrund, Rennstrecke, …)
2. Neue Figur (Rennauto) erstellen
3. Skript für das Rennauto schreiben

Basis – Implementierung

Steuerung des Rennautos

Das Spiel wird mit der Taste s gestartet.

Startposition für das Rennauto festlegen.

Bedingung 1: Game-Over

Bedingung 2: Gewinn

Ideen für Erweiterungen

Soundeffekte einfügen

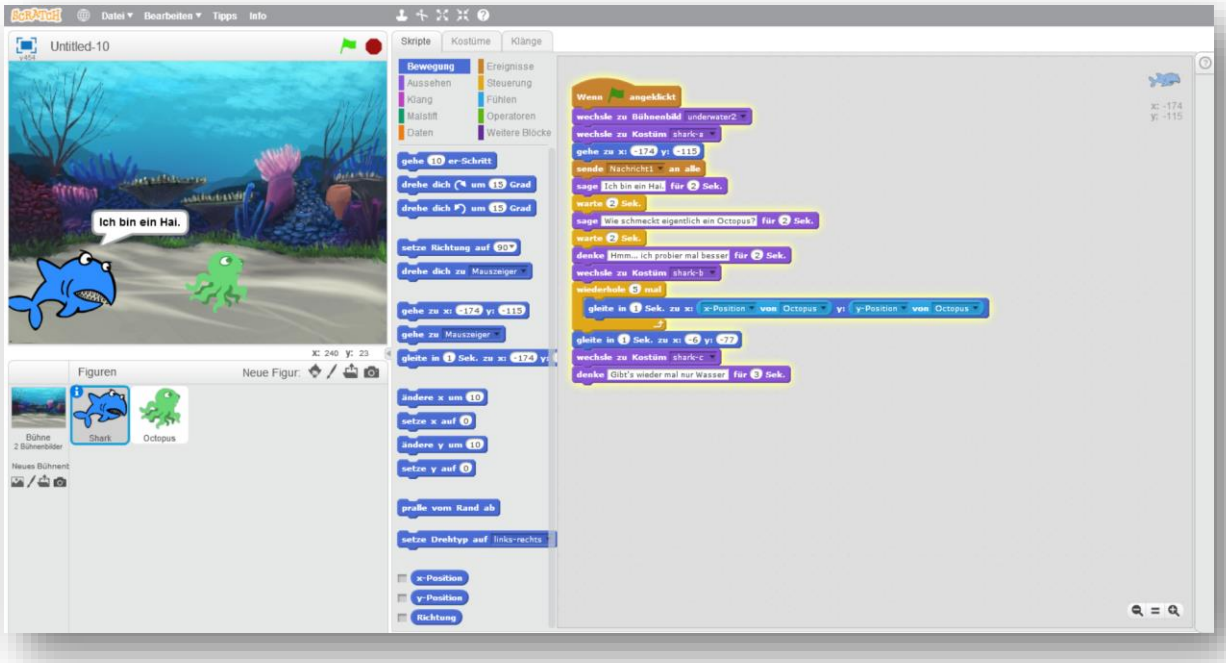
Weitere Bühnenbilder (Level)

Veränderung der Geschwindigkeit

Hindernisse auf der Rennstrecke einfügen

Zeitanzeige und Rundenanzeige

6.3 Eine Geschichte – Der Hai und der Oktopus



Aufgabenstellung

Programmiere eine Geschichte. Überlege dir dazu eine eigene Geschichte oder orientiere dich an dieser Hai-Geschichte und verändere bzw. erweitere diese.

Beschreibung

Bei dieser Geschichte treffen ein Hai und ein Oktopus aufeinander. Nach einer kurzen Unterhaltung beschließt der Hai den Oktopus zu fressen. Der Oktopus flieht, jedoch nimmt der Hai die Verfolgung auf. Vergeblich, der Oktopus war zu schnell.

Link zum Spiel

https://scratch.mit.edu/projects/158220230/

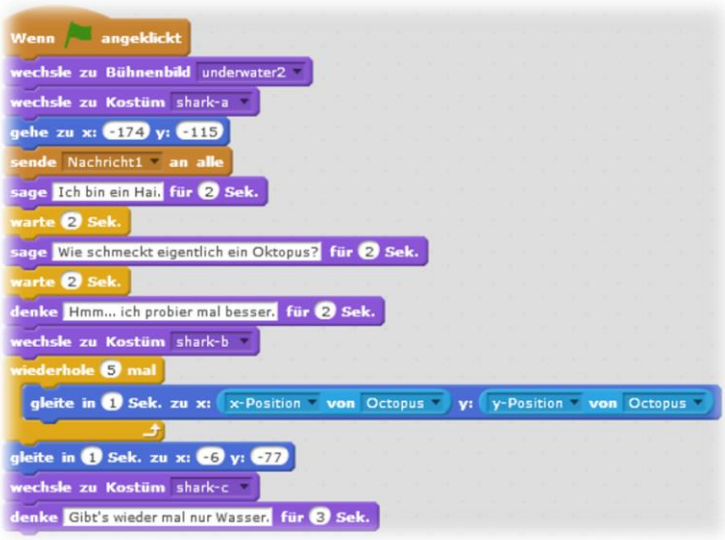
Programmierschritte

1) Erstelle ein neues Bühnenbild (Hintergrund)

2) Erstelle verschiedene Figuren

3) Schreibe für jede Figur ein Skript

Basis - Implementierung



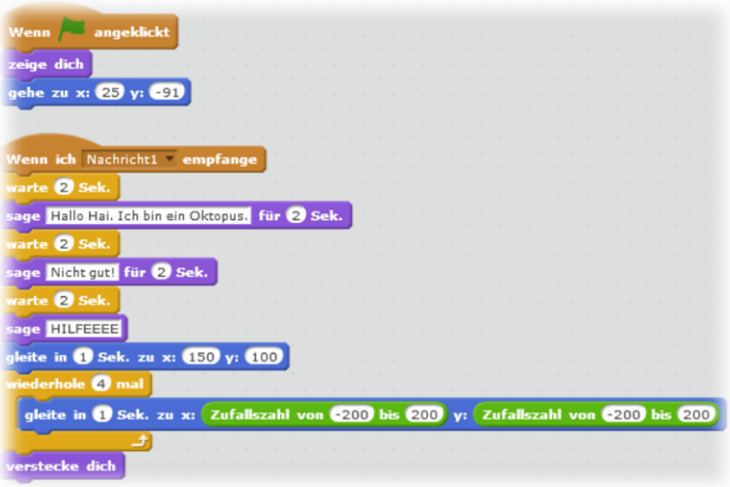
Skript Figur Hai

Ausgangssituation: Bühnenbild, Figur und Position

Start der Unterhaltung

Verfolgung des Oktopusses

Endposition der Figur und Abschlussworte



Skript Figur Oktopus

Ausgangssituation: Erscheinen, Position

Start Dialog (Wenn Nachricht 1 von Hai)

Fluchtposition

Zufällige Flucht
Verstecken

Ideen für Erweiterungen

Soundeffekte einfügen

Weitere Bühnenbilder / Figuren

Veränderung der Geschwindigkeit

Ereignis, falls eine Figur die andere fängt

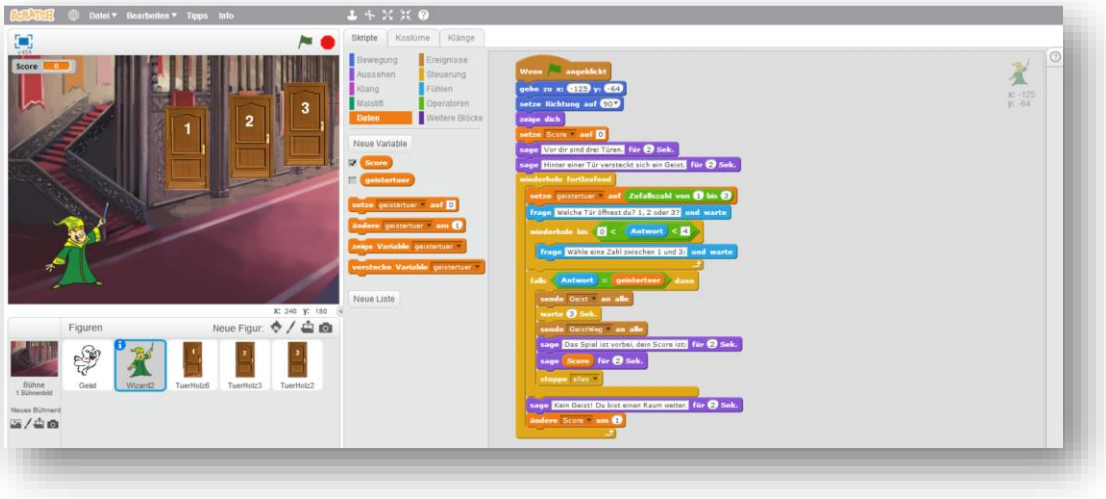
7 Von Scratch zu Python

Da die visuelle Programmierung bei komplexeren Aufgabenstellungen schnell an ihre Grenzen stößt, empfiehlt es sich, mit den Schülerinnen und Schülern nach einem Programmier-Einführungskurs mit Scratch auf eine textbasierte Programmiersprache wie Python umzusteigen. Dieser Übergang sollte fließend gestaltet werden, um die Motivation der Schülerinnen und Schüler hoch zu halten und sie nicht kognitiv zu überfordern.

Eine Möglichkeit den Umstieg der visuellen auf die textbasierte Programmierung zu gestalten bietet die Entwicklung eines Spieles. Dazu wird zuerst wird ein beliebiges Spiel in Scratch entwickelt. Dieses Spiel wird in weiterer Folge Schritt für Schritt in Python implementiert und immer wieder auf die Scratch Implementierung verwiesen. Die Schülerinnen und Schüler haben somit einen direkten Vergleich der visuellen und der textbasierten Programmierung. Durch diesen direkten Vergleich wird es den Schülerinnen und Schülern erleichtert die zentralen Elemente der Programmierung, unabhängig der Programmiersprache, verstehen und erkennen zu können.

Ein exemplarischer Übergang von Scratch nach Python bildet das folgende *Geisterspiel*.

7.1 Geisterspiel



Beschreibung In jeder Runde des Spieles steht man vor drei Türen. Hinter einer dieser Türen versteckt sich ein gefährlicher Geist. Ziel des Spieles ist es diesem

Geist zu entgehen. Errät man eine Türe ohne Geist, bekommt man einen Punkt gutgeschrieben und man kann weiterspielen. Das Spiel wiederholt sich so lange, bis man sich für eine Geistertüre entscheidet.

Link zum Spiel https://scratch.mit.edu/projects/155489827/

Quelle Die ursprüngliche Idee für dieses Projekt stammt aus dem Buch Programmieren supereasy. (Vorderman und Woodcock 2016, S. 96)

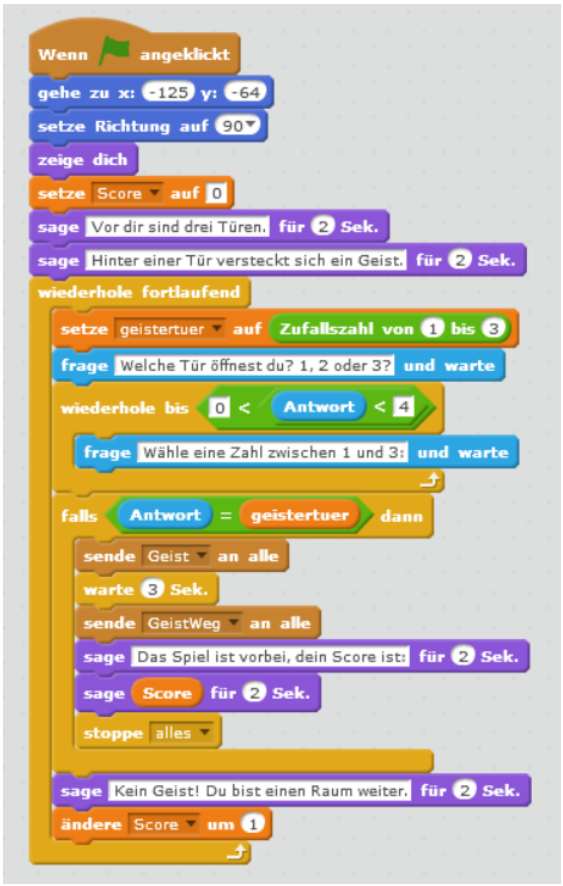
Die Umsetzung in Scratch wurde teilweise von https://scratch.mit.edu/projects/93423132/ übernommen.

Implementierung

Python

```python
#Geisterspiel

from random import randint
print ("Geisterspiel")
du_bist_mutig = True
score = 0
print("Vor dir sind drei Türen.")
print("Hinter einer Tür versteckt sich ein Geist.")

while du_bist_mutig:
    geistertuer = randint(1,3)
    tuer = int(input("Welche Tür öffnest du? 1,2 oder 3?"))
    tuer_nummer = tuer
    while (tuer < 1 or tuer > 3):
        tuer = int(input("Wähle eine Zahle zwischen 1 und 3:"))

    if tuer_nummer == geistertuer:
        print("Hinter der Tür", tuer_nummer, "ist ein Geist!")
        du_bist_mutig = False
    else:
        print("Kein Geist!")
        print("Du bist einen Raum weiter.")
        score = score + 1


print("Das Spiel ist vorbei, dein Score ist:", score)
```

Scratch



8 Quellen und weitere Literatur

8.1 Verwendete Quellen

Vorrderman, Carol; Woodcock, Jon (2016): Spiele programmieren supereasy. München: Dorling Kindersley Verlag GmbH.

Online:

Blockkategorien: http://scratch-dach.info/wiki/Alle_Bl%C3%B6cke_%3D_%C3%9Cbersichts-Liste#F.C3.BChlen [Stand 01.05.2017]

Bühne: http://scratch-dach.info/wiki/B%C3%BChne [Stand 23.04.2017]

Einleitung: https://de.wikipedia.org/wiki/Scratch_(Programmiersprache) [Stand 01.05.2017]

Fangspiel: https://www.youtube.com/watch?v=lgCsdH6GWsQ [Stand 01.05.2017]

Figuren: http://scratch-dach.info/wiki/Figur [Stand 23.04.2017]

8.2 Weitere Literatur

Ford, Jerry Lee (2009): Scratch programming for teens. Boston Mass.: Course Technology PTR.

Marji, Majed (2014): Learn to program with Scratch. San Francisco Calif.: No Starch Press.

Ulwer, Jürgen (2015): Scratch 2.0. 1. Aufl., 1. Aktualisierung. Bodenheim, Dübendorf: Herdt.

Wainewright, Max (2016): Spielend programmieren lernen. Mit Scratch, Logo, Python, HTML und JavaScript. Ravensburg: Ravensburger Buchverlag.

Online:

Scratch-Wiki: http://scratch-dach.info/wiki/Scratch-Wiki:%C3%9Cber [Stand 02.05.2017]

Offizielle Scratch Seite (deutsch): https://scratch.mit.edu/ [Stand 02.05.2017]