



## VERTIEFUNG ALGORITHMEN

# UNTERRICHTSGANG MIT HINTERGRUNDINFORMATIONEN

Dieses Werk ist unter einem **Creative Commons 3.0 Deutschland Lizenzvertrag** lizenziert:

- Namensnennung
- Keine kommerzielle Nutzung
- Weitergabe unter gleichen Bedingungen



Um die Lizenz anzusehen, gehen Sie bitte zu <https://creativecommons.org/licenses/by-nc-sa/3.0/de>.

Monika Eisenmann – E-Mail: [eisenmann.schule@email.de](mailto:eisenmann.schule@email.de). – April 2018



## Inhaltsverzeichnis

**Vorbemerkungen..... 3**

**Ein kurzer Überblick des Unterrichtsganges..... 3**

**Visuelle Programmierumgebung..... 4**

**Unterrichtsmaterial..... 4**

**Wiederholung Klasse 7..... 5**

**Froschhüpfen..... 5**

**Froschhüpfen – Teil I (Aktion)..... 5**

**Froschhüpfen – Teil I (Programmierung)..... 6**

**Froschhüpfen – Teil II (Aktion)..... 7**

**Froschhüpfen – Teil II (Programmierung)..... 9**

**Einarmiger Bandit..... 9**

**Einarmiger Bandit (Aktion)..... 9**

**Einarmiger Bandit (Programmierung)..... 11**

**Softwareprojekt..... 14**

**Vernetzung mit dem mathematischen Teil von IMP..... 15**

**Literaturliste und Internetseiten:..... 16**

Scratch wird von der Lifelong-Kindergarten-Group am MIT-Media-Lab entwickelt. Siehe <http://scratch.mit.edu>.  
Scratch ist lizenziert unter *CC BY-SA 2.0* (<https://creativecommons.org/licenses/by-sa/2.0/deed.en>).

Der MIT App Inventor (<http://appinventor.mit.edu>) wurde ursprünglich von einem Entwicklerteam um Mark Friedman und Hal Abelson bei Google entwickelt und 2012 an das MIT übergeben.  
Der MIT App Inventor wird unter der Creative Commons Attribution-ShareAlike 3.0 Unported License veröffentlicht:  
<https://creativecommons.org/licenses/by-sa/3.0>



## Vorbemerkungen

„Aufbauend auf den in Klasse 7 kennengelernten Grundbausteinen von Algorithmen verknüpfen die Schülerinnen und Schüler diese Grundbausteine systematisch zu Programmen, die in ihrer Komplexität zunehmen. Um ihre Programme zu strukturieren und redundanten Code zu vermeiden, können sie Unterprogramme nutzen und dadurch Funktionalitäten in eigene Programmteile auslagern

Durch die Verwendung von Zufallszahlen erschließen sich eine Reihe neuer Möglichkeiten, insbesondere in den Bereichen Spieleprogrammierung und Simulation sowie zum Erzeugen von Testdaten.

Anhand geeigneter Szenarien erfahren die Schülerinnen und Schüler, dass bei der Speicherung von mehreren gleichartigen Daten der Einsatz einzelner Variablen nicht mehr sinnvoll ist. Sie lernen, dass diese Daten in einer geeigneten Struktur zusammengefasst werden können. Je nach konkreter eingesetzter (zum Beispiel visueller) Programmiersprache handelt es sich hier um indexbasierte Listen, Arrays oder ähnliches. Anhand von Algorithmen, die mit einer Schleife über alle Werte einer Datenstruktur iterieren, werden die Konzepte von Algorithmen und Datenspeicherung miteinander verknüpft.

Ein Softwareprojekt nimmt in diesem Schuljahr einen breiten Raum ein. Hierzu gehören Planungs-, Durchführungs- und Testphasen.“<sup>1</sup>

Im vorliegenden Dokument wird ein möglicher Unterrichtsgang aufbauend auf den Vorgaben des Bildungsplanes aufgezeigt, der inhaltsbezogene und prozessbezogene Kompetenzen verknüpft, und es werden kurze Einblicke in die Hintergründe der neuen Elemente und Inhalte gegeben.

Im Anschluss daran werden Einsatzmöglichkeiten der erworbenen Fähigkeiten im mathematischen Teil von IMP vorgestellt.

## Ein kurzer Überblick des Unterrichtsganges

Zu Beginn werden die in Klasse 7 erworbenen Kompetenzen wiederholt. Darauf aufbauend lernen die Schülerinnen und Schüler eine Möglichkeit kennen, mehrere gleichartige Daten in einer indexbasierten Liste zu speichern und mit der erzeugten Liste zu arbeiten. Dieses geschieht sowohl ohne den Einsatz des Computers, um die Strukturen und Algorithmen begreifen zu können, als auch in Form eines kleinen Programmierprojektes, in dem das erworbene Wissen umgesetzt wird.

In einem weiteren kleinen Projekt werden logische Verknüpfungen in Bedingungen genutzt, die vorher wieder ohne PC kennengelernt werden. In beiden gemeinsamen Projekten werden Zufallszahlen verwendet, und es kommen kleine Unterprogramme zum Einsatz, um redundanten Code zu vermeiden und die Programme zu strukturieren.

Im Anschluss wird in Zweiertteams ein Softwareprojekt geplant, durchgeführt und immer wieder reflektiert und ggf. erweitert bzw. verändert. Testphasen werden in die Arbeit integriert.

<sup>1</sup> IMP-Bildungsplan 2016 (Anhörungsfassung), Stand 29. März 2018, S. 33



## Visuelle Programmierumgebung

Es ist durchaus möglich, die Programmierphasen mit Scratch zu gestalten und auch das Softwareprojekt in Scratch durchführen zu lassen.

Eine zweite Möglichkeit wäre, die Einführungsprojekte in Scratch zu bearbeiten und beim Softwareprojekt auf eine andere Umgebung umzusteigen. Um die Nähe zu Scratch zu erhalten, bieten sich hier z.B. die Arbeit mit dem MIT-App-Inventor oder der Einsatz von mBlock in Kombination mit einem mBot-Bausatz an.

Es ist aber auch möglich, den gesamten Unterrichtsgang mit dem MIT-App-Inventor zu machen. Die Motivation der Schülerinnen und Schüler ist in dem Fall eventuell sogar noch höher, da sie tagtäglich Apps nutzen und fasziniert davon sind, selbst relativ einfach Apps erstellen zu können.

Bei der Entscheidung müssen ein paar wichtige Punkte bedacht und Vor- und Nachteile abgewogen werden.

Der MIT App Inventor ist unter Umständen anspruchsvoller, da die Möglichkeiten reichhaltiger und die Begriffe auf Englisch sind.

Scratch ist offline nutzbar, fertige Programme aber nur in Scratch (oder auf einer entsprechenden Plattform im Internet) ausführbar. Dafür entfällt die Einführungsphase, da die Schülerinnen und Schüler es aus Klasse 7 schon kennen.

Der MIT-App-Inventor ist nur mit einer Internetverbindung nutzbar und es wird ein google-Konto benötigt. Hier genügt ein von der Lehrkraft angelegtes Konto, das die gesamte Klasse nutzen kann. Dafür können die Schülerinnen und Schüler aber ihre Ergebnisse als App auf ihrem Smartphone mit nach Hause nehmen. Im Moment ist das noch eingeschränkt auf Android-Phones, das soll aber in diesem Jahr noch auf iOS-Phones erweitert werden<sup>2</sup>. Eine ausführlichere Beschreibung finden Sie im Dokument [02\\_alg\\_anleitung\\_nutzung\\_ai2](#), einen Vergleich der wichtigsten Bausteine von Scratch mit denen des MIT-App-Inventors im Dokument [06\\_alg\\_vergleich\\_scratch\\_ai2](#).

Möchte man für das Softwareprojekt Roboterbausätze nutzen, stellt sich die Kostenfrage. Wenn an Schulen schon Bausätze oder Roboter vorhanden sind, kann sich auch diese Form anbieten.

## Unterrichtsmaterial

Die Wiederholung von Algorithmen aus Klasse 7 und die jeweils von der Programmierumgebung unabhängigen Materialien finden Sie direkt im Ordner [2\\_kopiervorlagen](#).

Die Materialien, die von der Programmierumgebung abhängig sind, sind in zwei Unterordnern zu finden. Es gibt alle Materialien für Scratch ([alg\\_unterricht\\_mit\\_scratch](#)) und alle Materialien für den MIT-App-Inventor ([alg\\_unterricht\\_mit\\_ai2](#)). Für diesen finden Sie eine mögliche Einführung in die Programmierumgebung für die Schülerinnen und Schüler ebenfalls in diesem Unterordner.

Im Ordner [6\\_software](#) finden Sie Scratchprojekte zu den einzelnen Teilen. Außerdem sind dort alle Apps als aia-Dateien hinterlegt. Diese können Sie im App-Inventor importieren, verändern und ggf. auf Ihrem Smartphone installieren.

Sollten Sie mit dem App-Inventor arbeiten wollen, finden Sie im Ordner [1\\_hintergrund](#) eine Anleitung zur Nutzung, die für Sie als Lehrkraft gedacht ist. Außerdem gibt es eine Datei in den

2 Stand 19.04.2018



Kopiervorlagen, die die wichtigsten Blöcke der beiden Umgebungen nebeneinander stellt.

## Wiederholung Klasse 7

In Klasse 7 haben die Schülerinnen und Schüler Anweisungen, Sequenzen, Verzweigungen, Schleifen und Variablen kennen und nutzen gelernt. Diese Begriffe werden in einem kahoot<sup>3</sup>-Quiz abgefragt und wiederholt. Hier ist es wichtig, Vergessenes wieder in Erinnerung zu rufen und Begriffe ggf. noch einmal zu klären. Es sind einfache Fragen darunter, aber auch kompliziertere Codeabschnitte, bei denen die Schülerinnen und Schüler sehen können, wie gut sie mit Kontrollstrukturen und Variablenkonzepten umgehen können.

Das Quiz kann jederzeit verändert oder erweitert werden. Erstellen Sie sich dazu einen Account in kahoot und duplizieren Sie das Quiz. Sie finden eine Kopie in den eigenen kahoots und können diese editieren.

Im Anschluss an das Quiz bekommen die Schülerinnen und Schüler einen Arbeitsauftrag (01\_alg\_auftrag\_wiederholung\_klasse7) mit einem kleinen Scratch-Video (movie\_einstieg) und können zeigen, ob sie die Kontrollstrukturen richtig und geschickt einsetzen können.

Die Lehrkraft kann hier Zeit und Gelegenheit nutzen, die Schülerinnen und Schüler zu unterstützen, denen die Grundlagen aus Klasse 7 nicht mehr vollständig in Erinnerung sind und denen die Wiederholung durch das Quiz zu Beginn nicht ausgereicht hat.

Am Ende dieser Doppelstunde werden die Lösungen der Teams verglichen und die eleganteste und schnellste Lösung herausgesucht und besprochen. Auch hierbei kann man Einsatzmöglichkeiten und Sinn der einzelnen Kontrollstrukturen noch einmal im Plenum thematisieren.

## Froschhüpfen

Das gesamte Projekt geht über drei Doppelstunden und beinhaltet Teile ohne PC und Teile mit Programmieraufgaben für Scratch oder den MIT-App-Inventor.

Im ersten Teil wird das Variablenkonzept wiederholt und bei der Programmierung werden Zufallszahlen eingesetzt.

Im zweiten Teil wird eine neue Datenstruktur eingeführt und genutzt.

## Froschhüpfen – Teil I (Aktion)

Zunächst bekommt jede Schülerin und jeder Schüler einen Papierfrosch<sup>4</sup>. Die Frösche sind durchnummeriert von eins bis zur Anzahl der Schüler. (Wenn im zweiten Auftrag zwei oder drei Gruppen gebildet werden, weil die Klasse zu groß ist, erfolgt die Nummerierung für jede Gruppe extra, auch wenn die Schülerinnen und Schüler hier noch nicht in Gruppen aufgeteilt sind. Hier bietet es sich an, für jede Gruppe eine andere Farbe der Klebpunkte zu wählen, dass die Zugehörigkeit zur Gruppe klar erkennbar ist.)

3 URL: [www.kahoot.com](http://www.kahoot.com) - „Algorithmen in Scratch“ (by eisenmann\_ggk)

4 URL: <http://www.mittags-pause.de/origami-frosch-falten-anleitung-zum-basteln/> (abgerufen am 19.04.2018); Alternativ können Kunststoff-Chips des sog. Floh-Spiels verwendet werden.



In einem ersten Schritt bekommen alle den Auftrag, ihren Frosch soweit wie möglich springen zu lassen und die Sprungweite auf einen kleinen Zettel zu notieren. Es soll das Variablenkonzept aus Klasse 7 wiederholt werden. Deshalb bekommt jeder eine leere Streichholzschachtel mit der Aufschrift „Sprungweite“.

Der Zettel mit der notierten Sprungweite wird in das Schächtelchen gelegt. Jetzt lassen alle ihren Frosch ein weiteres Mal springen und vergleichen die neue Sprungweite mit der bisherigen. Ist die neue Sprungweite größer, so wird der alte Wert auf dem Zettel in der Streichholzschachtel durch den neuen Wert ersetzt. Wenn nicht, wird der Wert der Variable nicht verändert.

Das wird so lange fortgesetzt, bis jeder Frosch 10-mal gesprungen ist.

Die Schülerinnen und Schüler erhalten den Auftrag genau aufzuschreiben, welche Schritte sie in jedem Durchgang gemacht haben und sie tragen ihre Werte im Froschsprungprotokoll (02\_alg\_frosch\_auftrag\_1\_protokoll) ein. Ein Beispiel für ein ausgefülltes Protokoll finden Sie in den Lösungen (02\_alg\_frosch\_auftrag\_1\_beispielprotokoll).

## Hintergrund:

Die Aktion dient auf der einen Seite der Wiederholung des Variablenkonzeptes, auf der anderen Seite ist sie Grundlage für die spätere Einführung einer neuen Datenstruktur.

In der Programmierung bezeichnet man als Variable einen logischen Speicherplatz mit dessen Wert<sup>5</sup>. Jede Variable besitzt einen Namen, unter dem man sie ansprechen und ihren Wert verändern kann.

Für die Schülerinnen und Schüler bietet sich hier als Veranschaulichung ein Kästchen an. Das Kästchen bekommt einen Namen und steht in der Folge für die Variable.

Bei der Initialisierung wird der Variable das erste Mal ein Wert zugewiesen. In der Schüleraktion entspricht dies dem Einlegen des ersten beschrifteten Zettels in die Streichholzschachtel.

Im Folgenden werden in jedem Schritt (nach jedem Sprung des Frosches) der Wert der Variable erfragt und ggf. verändert. Hierbei nutzen die Schülerinnen und Schüler bei der Umsetzung in der Programmierumgebung eine Verzweigung ohne Alternative: „Falls „die neue Sprungweite“ größer ist als „die Sprungweite (in der Schachtel)“, wird der Wert der neuen Sprungweite in der Variable Sprungweite gespeichert (auf den Zettel in der Schachtel geschrieben).“

Nach den zehn Sprüngen ist bei jedem Frosch die maximale Sprungweite auf dem Zettel in der Schachtel notiert. Dies machen wir uns später bei der Einführung der neuen Datenstruktur zu nutze.

## Froschhüpfen – Teil I (Programmierung)

Der nächste Auftrag (02\_alg\_frosch\_auftrag\_1) findet am PC statt. Hier wird das Froschhüpfen von vorher programmiert:

1. Ein Frosch, der ganz links auf der Bühne sitzt, „springt“ beim Anklicken mit der Maus eine zufällige Weite nach rechts.
2. Das erste Ergebnis wird direkt in einer Variable „Sprungweite“ gespeichert.
3. Eine weitere Variable „Sprungweite\_neu“ erhält nach jedem „Sprung“ die neue Weite als

5 Schülerduden Informatik, 4. Auflage, 2003, S. 527, Begriff: Variable



Wert.

4. Ab dem zweiten Sprung wird überprüft, ob die neue Weite größer ist. Falls ja, wird der Wert der Variable „Sprungweite“ ersetzt, sonst bleibt er gleich.
5. Zur Übung kann eine weitere Variable als Zähler genutzt werden, um die Sprünge des Frosches mitzuzählen.

## Hintergrund:

Wenn die Schülerinnen und Schüler in Klasse 7 noch nicht mit Zufallszahlen gearbeitet haben, lernen sie die Nutzung hier kennen. Es ist nicht daran gedacht zu thematisieren, wie der Computer Zufallszahlen erzeugt. Es geht hier lediglich um die Eröffnung neuer Möglichkeiten, die Zufallszahlen bei der Programmierung bieten.

In unserem Fall simulieren wir die Froschsprünge und erhalten Sprungweiten in einem vorgegebenen Bereich.

Um das Ganze anschaulich zu machen, lassen die Schülerinnen und Schüler auf der Bühne in Scratch einen Frosch tatsächlich die zufällige Weite in Pixel nach vorne springen. Um das deutlich sichtbar zu machen, ist es sinnvoll, den Frosch beim Anklicken zunächst wieder an die Ausgangsposition zu setzen, dort kurz warten zu lassen und erst dann den Sprung durchzuführen.

## Froschhüpfen – Teil II (Aktion)

Der zweite Auftrag geht an die gesamte Gruppe. Ist die Klasse zu groß, kann man auch in zwei oder drei Untergruppen aufteilen. Jede Gruppe bekommt den Auftrag, die jeweils beste Sprungweite aller Frösche der Gruppe in einer neuen Datenstruktur zu speichern, um die Weiten vergleichen zu können.

Dazu bekommt jede Gruppe zunächst den Auftrag, ihre Schachteln in der Reihenfolge ihrer Froschnummern aneinander zu legen. In Schachtel Nr. 1 liegt also die maximale Sprungweite von Frosch Nr. 1, in Schachtel Nr. 2 die maximale Weite von Frosch Nr. 2, usw.

Die Schülerinnen und Schüler einer Gruppe erhalten dann den Auftrag, in einem ersten Schritt den Frosch mit der größten Weite zu finden, und in einem zweiten Schritt den Mittelwert (arithmetisches Mittel) der Sprungweiten aller Frösche in der Gruppe zu berechnen. Für beide Aufgaben gibt es folgende Vorgaben:

1. Es darf immer nur eine Schachtel geöffnet sein.
2. Die Gruppe darf sich zusätzlich zum Wert in der geöffneten Schachtel immer nur einen weiteren Wert „merken“. (Möchte man auch wissen, welcher Frosch die beste Sprungweite hatte, muss man sich dazu auch die jeweilige Position merken.)

Auch hier bekommen die Schülerinnen und Schüler wieder die Aufgabe, ihre Schritte genau aufzuschreiben. (Es macht Sinn, einen Protokollanten oder eine Protokollantin zu bestimmen.) Ein Beispielprotokoll finden Sie in den Lösungen (02\_alg\_frosch\_auftrag\_2\_beispielprotokoll).

Im Plenum wird das Vorgehen der einzelnen Gruppen vorgestellt und besprochen.

Im Anschluss daran erhalten die Schülerinnen und Schüler ein Arbeitsblatt zu Listen



(02\_alg\_frosch\_auftrag\_2\_listen), bei dem sie gruppenintern zunächst Aufgaben zur Schreibweise und danach Aufgaben zu Listen in der Programmierumgebung bearbeiten.

## Hintergrund:

Mehrere Daten des gleichen Typs lassen sich in unterschiedlichen Datenstrukturen speichern. Sowohl in Scratch als auch im MIT-App-Inventor heißt diese Datenstruktur Liste, ist aber eine Mischform aus einem dynamischen Array und einer verketteten Liste.

Bei einem Array (Feld) handelt es sich um eine lineare Anordnung von Daten desselben Grundtyps. Ein Array hat eine feste Größe, die nicht veränderbar ist. Da jedes Element mit einem Index versehen ist, kann direkt auf einzelne Elemente zugegriffen werden.

Eine verkettete Liste dagegen bietet die Möglichkeit, beliebig viele Elemente geordnet in einer Datenstruktur abzulegen, es ist aber nur das erste Element bekannt. Jedes Element kennt seinen Nachfolger. Ein Zugriff direkt auf ein Element ist aber nicht möglich.

Bei doppelt verketteten Listen ist auch das letzte Element bekannt und jedes Element kennt sowohl Vorgänger als auch Nachfolger.

Der große Vorteil von Arrays ist es, dass man direkt auf ein Element durch seinen Index zugreifen kann, der Nachteil ist die feste Größe. Hier bietet sich die Struktur des dynamischen Arrays an, bei dem die Größe des Feldes verändert werden kann.

Wie oben schon erwähnt, bieten beide Programmierumgebungen eine elegante Mischform an. Es können sowohl beliebig viele Elemente angehängt werden, ohne die Größe verändern zu müssen (wie bei verketteten Listen), man kann auch direkt auf ein Element zugreifen (wie bei Arrays).

Bei einem Array bleiben Positionen, deren Element gelöscht ist, unbesetzt. Da es eine feste Größe besitzt, wird der Wert des gelöschten Elementes auf „null“ gesetzt. Wird dagegen ein Element aus einer Liste gelöscht, wird der Nachfolger des gelöschten Elementes an dessen Vorgänger angehängt. Diese Vorgehensweise der Liste wird auch hier in beiden Umgebungen genutzt.

Zur Darstellung von Listen werden in den meisten Programmiersprachen eckige Klammern verwendet.

### Beispiele:

[23, 14, 56, 70, 50, 23] – Liste von Sprungweiten eines Frosches

[ ] - leere Liste

['Simon', 'Susi', 'Hans', 'Elke', 'Micha'] – Liste der Gruppenteilnehmer

Alle Elemente einer Liste sind durchnummeriert. In den gängigen Programmiersprachen startet die Zählung bei 0. In Scratch und dem MIT-App-Inventor startet sie – was für die jüngeren Schülerinnen und Schüler intuitiver ist – mit 1. Die Nummer eines Elementes wird Index genannt.

### Beispiel:

S = [23, 14, 56, 70, 50, 23]

Die Liste S hat 6 Elemente. Dabei beschreibt S[ i ] das i-te Listenelement.

S[ 1 ] ist 23, S[ 2 ] ist 14, usw.

Den Umgang mit Listen in Scratch und dem MIT-App-Inventor finden Sie auf dem jeweiligen





Arbeitsblatt für die Schülerinnen und Schüler (02\_alg\_frosch\_auftrag\_2\_listen).

## Froschhüpfen – Teil II (Programmierung)

Die Schülerinnen und Schüler versuchen jetzt, ihre Entdeckungen und ihr neu erworbenes Wissen über Listen auch in der Programmierumgebung umzusetzen. Die Aufträge zur Umsetzung mit Hilfestellungen für Scratch (oder den AI2) finden Sie im Material (02\_alg\_frosch\_auftrag\_2).

Zunächst wird eine neue Liste erzeugt, die maximalen Sprungweiten der Frösche aus der Gruppe dort eingegeben und es werden zwei Tastaturereignisse programmiert, die das Maximum bzw. den Mittelwert der Listenelemente bestimmen sollen.

Im Anschluss daran können die beiden Aspekte verknüpft werden. Jede zufällige Sprungweite des Frosches auf der Bühne wird in die Liste eingetragen. Dann können Maximum und Mittelwert der Elemente der Liste bestimmt und ausgegeben werden.

Als Differenzierungsmöglichkeit können schnelle Schülerinnen und Schüler sich überlegen, wie die Einträge der Liste sortiert werden können. Für die Programmierung eines Spiels mit Highscore-Liste ist das zum Beispiel eine gute Vorarbeit.

Grundsätzlich ist die Sortierung noch nicht in Klasse 8 im IMP-Bildungsplan.

In der Erprobung haben die Schülerinnen und Schüler, die ihre Listen sortieren wollten, die Suche nach dem Maximum genutzt. Sie haben zwei weitere Listen definiert. Eine neue, leere Liste und eine Hilfsliste, die anfangs ihrer Liste der Sprungweiten entspricht. Sie haben dann in der ursprünglichen Liste nach dem Maximum gesucht und dieses an die neue leere Liste angehängt. Anschließend wurde aus der Hilfsliste das Element mit dem größten Wert gelöscht. In den nächsten Schritten wurde dies so fortgesetzt und das jeweils das neue Maximum an den Beginn der Liste gesetzt.

## Einarmiger Bandit

In diesem Projekt werden erneut Zufallszahlen genutzt, einfache logische Verknüpfungen kennen und nutzen gelernt und in einer Erweiterung erste kleine Unterprogramme verwendet und anschließend eingeübt.

Das gemeinsame Projekt erstreckt sich über drei Doppelstunden.

## Einarmiger Bandit (Aktion)

Die Schülerinnen und Schüler werden wieder in Gruppen aufgeteilt und bekommen pro Gruppe drei Spielwürfel. Zu Beginn hat jedes Gruppenmitglied 100 Punkte auf dem Konto. Reihum wird gewürfelt. Die bzw. der Würfelnde wirft alle drei Würfel gleichzeitig. Zeigen alle drei Würfel eine andere Augenzahl, werden 5 Punkte vom Konto abgezogen. Sind genau zwei der drei Augenzahlen gleich, werden 10 Punkte und bei drei gleichen Augenzahlen sogar 100 Punkte gutgeschrieben.

Zum Spiel gibt es ein Arbeitsblatt mit Aufträgen (03\_alg\_bandit\_vorbereitung\_aussage). Die Schülerinnen und Schüler simulieren durch das Spiel den einarmigen Banditen und müssen sich in den Aufträgen damit beschäftigen, was bei der Auswertung bedacht werden muss.



Dabei kommen sie mit einfachen logischen Verknüpfungen in Berührung.

Übungen am Beispiel des Spiels „Mäxle“ vertiefen den Umgang (03\_alg\_bandit\_log\_verknuepfungen\_uebungen).

## Hintergrund:

Bei der Programmierung werden in Verzweigungen Bedingungen benötigt. Diese sind entweder wahr oder falsch. Je nach Wahrheitswert der Bedingung werden unterschiedliche Teile der Verzweigung ausgeführt.

Die Bedingung besteht aus einer Aussage. Diese Aussage kann wieder aus anderen Aussagen bestehen, die miteinander logisch verknüpft sind.

### Beispiel:

W1 soll für die Augenzahl des ersten Würfels stehen, W2 für die des zweiten und W3 für die des dritten Würfels.

Bedingung 1:  $W1 > 3$  („Die erste Augenzahl ist größer als 3.“ - Ergebnisse für W1: 4, 5, 6)

Bedingung 2:  $W2 = W1$  („Die Augenzahlen von Würfel 1 und 2 sind gleich.“ - Ergebnisse für W1 und W2: (1,1), (2,2), (3,3), (4,4), (5,5), (6,6))

Bedingung 3:  $(W1 > 3)$  und  $(W2 = W1)$  („Die erste Augenzahl ist größer als 3 und ist gleich wie die zweite Augenzahl.“ - Ergebnisse für W1 und W2: (4,4), (5,5), (6,6))

Genauso wie zwei Aussagen mit „und“ verknüpft werden können, kann man einer Aussage ein „nicht“ voranstellen oder zwei Aussagen mit „oder“ verknüpfen.

Achtung: Im allgemeinen Sprachgebrauch wird „oder“ oft als „entweder oder“ genutzt. In der mathematischen Logik sind das zwei unterschiedliche Verknüpfungen. Bei „entweder oder“ müssen beide Aussagen unterschiedliche Wahrheitswerte haben, bei „oder“ dürfen auch beide wahr sein.

Die logischen Verknüpfungen lassen sich in sog. Wahrheitstabellen darstellen, die Ihnen hier zur Hilfe dienen sollen (s.u.). Die Schülerinnen und Schüler lernen diese aber erst in Klasse 9 beim Thema Aussagenlogik und Graphen kennen und nutzen.

Beispiele für Wahrheitstabellen:

A und B seien Aussagen, „w“ steht für die Aussage ist „wahr“, „f“ für die Aussage ist „falsch“.

A	nicht A
f	w
w	f

A	B	A und B
f	f	f
f	w	f



w	f	f
w	w	w

A	B	A oder B
f	f	f
f	w	w
w	f	w
w	w	w

## Einarmiger Bandit (Programmierung)

Um einen einarmigen Banditen zu programmieren, brauchen die Schülerinnen und Schüler für die Auswertung genau diese logischen Verknüpfungen (03\_alg\_bandit\_auftrag\_1).

Es gibt hier unterschiedliche Lösungsmöglichkeiten. Es kann sein, dass nicht alle auf die kürzeste und eleganteste Variante kommen. Hier kann man am Ende drüber sprechen, warum auch andere Bedingungen den gleichen Wahrheitswert liefern.

Zum Beispiel genügt es, beim Vergleich auf drei gleiche Bilder folgende Aussage auf ihren Wahrheitswert zu prüfen:

(Bild 1 = Bild 2) und (Bild 1 = Bild 3)

Es folgt automatisch (Bild 2 = Bild 3), wenn die obige Aussage wahr ist.

Auch die Reihenfolge der Bedingungen entscheidet über den Umfang der Bedingungen.

**Möglichkeit 1:** Es wird zunächst überprüft, ob alle drei Bilder gleich sind

Falls (Bild 1 = Bild 2) und (Bild 1 = Bild 3)

    dann <werden die Punkte um 100 erhöht>

sonst falls ((Bild 1 = Bild 2) oder (Bild 1 = Bild 3)) oder (Bild 2 = Bild 3)

    dann <werden die Punkte um 10 erhöht>

sonst <werden Punkte abgezogen / hat man nichts gewonnen>.

**Möglichkeit 2:** Es wird zunächst überprüft, ob nur zwei der drei Bilder gleich sind

Falls ((Bild 1 = Bild 2) und nicht (Bild 1 = Bild 3)) oder ((Bild 1 = Bild 3) und nicht (Bild 1 = Bild 2)) oder ((Bild 2 = Bild 3) und nicht (Bild 2 = Bild 1))

    dann <werden die Punkte um 10 erhöht>

sonst falls (Bild 1 = Bild 2) und (Bild 1 = Bild 3)



```
dann <werden die Punkte um 100 erhöht>  
sonst ...
```

Bei der ersten Möglichkeit macht man sich zu nutze, dass die zweite Bedingung nur noch dann abgefragt wird, wenn nicht alle drei Würfel gleich sind. Deshalb kann man die Überprüfung des dritten Würfels weglassen. Statt 14 Vergleichen, braucht man nur acht.

Sollte niemand die zweite Möglichkeit nutzen, muss das nicht thematisiert werden.

Die Schülerinnen und Schüler können wieder eigenständig arbeiten oder Hilfekärtchen in Anspruch nehmen (Dabei können Sie gerne Hilfskarten ergänzen oder weglassen).

In einem weiteren Schritt soll das Programm ergänzt werden: Sind alle drei Bilder verschieden, werden ab jetzt Punkte abgezogen. Das erfordert jedes Mal die Überprüfung, ob überhaupt noch genug Punkte vorhanden sind. Sind alle Punkte weg, hat die Spielerin bzw. der Spieler verloren und das Spiel soll beendet sein.

Werden dagegen 1000 Punkte erreicht, endet das Spiel auch, nur dieses Mal mit einem Sieg.

Um die Übersichtlichkeit des Codes zu erhalten, macht es hier Sinn, die Punkteberechnung aus der Verzweigung auszulagern. Dazu bekommen die Schülerinnen und Schüler in ihrem Material (03\_alg\_bandit\_auftrag\_2\_unterprogramme) gezeigt, wie sie Unterprogramme in ihrer Programmierumgebung nutzen können.

Zur Übung und Festigung des neu Erlernen gibt es ein Übungsblatt zu Unterprogrammen (03\_alg\_uebungen\_unterprogramm), bei denen der erste Teil ohne, der zweite Teil mit Programmierumgebung zu lösen ist.

Bei diesem Projekt findet eine erste Begegnung mit dem Testen von Programmen oder Programmteilen statt. Auf einem Protokoll (04\_alg\_projekt\_testfaelle\_protokoll) finden die Schülerinnen und Schüler Anforderungen an ihr Projekt, die sie testen können. Sie können das Protokoll um weitere, selbst formulierte Anforderungen ergänzen.

## Hintergrund:

Unterprogramme entsprechen Methoden. In Scratch wird ein neuer Block erzeugt und mit Namen versehen – dies entspricht der Deklaration der Methode. An den Kopf wird ein Codeblock angehängt, was dem Schreiben der Methode entspricht. Aufgerufen werden kann die Methode dann an beliebig vielen Stellen im Programm.

Beim Erzeugen eines neuen Blocks kann man über die Einstellungen auswählen, ob man eine Methode mit oder ohne Übergabeparameter haben möchte. Man kann hier mehrere Parameter, auch von unterschiedlichem Typ einfügen.

In unserem Projekt möchten wir für jede Möglichkeit die Punkte berechnen lassen. Es bietet sich deshalb an, ein Unterprogramm mit einer Zahlvariable zu definieren, der beim Aufrufen des Unterprogramms dann der Wert 10, 100 oder -5 übergeben wird.

### Beispiel:

Definition: berechne\_punkte(p)

Aufruf z.B. berechne\_punkte(100)

Der Vollständigkeit halber sei hier erwähnt, dass im AI2 auch Methoden mit Rückgabewert „geschrieben“ und genutzt werden können. Diese werden im Blatt mit den Übungen auch



genutzt.



## Softwareprojekt

Für das Softwareprojekt stehen etwa sieben Doppelstunden zur Verfügung. Sie müssen sich vorab überlegen, ob Sie die Programmierumgebung vorgeben wollen, oder ob Sie die Entscheidung den Teams überlassen.

Es bietet sich an, Paarprogrammierung (englisch: Pair Programming) vorzuschreiben<sup>6</sup>. Dabei arbeiten immer zwei Personen zusammen an einem Projekt. Sie wechseln sich regelmäßig an der Tastatur ab. Da sie ständig im Gespräch sein können und immer vier Augen auf den Quellcode schauen, werden viele Fehler schon vorab entdeckt, die sonst im Nachhinein viel Zeit beim Testen gekostet hätten. Außerdem können sie gemeinsam ihre Gedanken und Ideen ergänzen und ausbauen.

Ergänzend dazu sollte in jeder Doppelstunde zu Beginn und am Ende ein kurzes Plenum statt finden. Alternativ dazu kann man die gemeinsame Phase auch zu einer festgelegten Zeit in der Mitte der Doppelstunde machen.

Im Plenum sollte jedes Team kurz vorstellen, an was es gerade arbeitet, was es sich für die aktuelle Stunde vorgenommen hat, welche Probleme eventuell aufgetaucht sind und wie sie vielleicht schon gelöst wurden. Die Lehrerin bzw. der Lehrer übernimmt hier die Moderation und muss ab und zu auch manches Team vor zu großen Vorhaben schützen. Außerdem kann er sich offene Fragen notieren und bis zum nächsten Mal klären, ob und eventuell wie die Problematik angegangen werden kann.

Sobald die Schülerinnen und Schüler ein Stück in ihrem Projekt vorwärts gekommen sind, kommt ein weiterer wichtiger Punkt dazu. Die Teams müssen sich überlegen, wie sie ihre Projekte geschickt auf Fehler testen können. Hier bietet es sich auch an, fremde Teams immer wieder ein Projekt eines anderen Teams testen zu lassen. Dabei werden auch Schwachstellen in der Nutzung offenbar, die den beiden Programmierern vielleicht selbst nicht aufgefallen sind.

Sie sollten vorab ein paar konkrete Bedingungen vorgeben, z.B. Nutzen von Zufallszahlen, Arbeiten mit mindestens einer Liste, sinnvolle Nutzung von Unterprogrammen. Außerdem hat die Erfahrung gezeigt, dass es Teams gibt, die Unterstützung bei der Ideenfindung brauchen und dankbar über mögliche Projektideen waren.

Eine mögliche Vorlage für ein Blatt für die Schülerinnen und Schüler finden Sie im Material (04\_alg\_projekt\_vorlage).

Erprobt wurde die Projektphase mit dem MIT-App-Inventor. Im Laufe der Zeit sind nach und nach einige Hilfekarten zusammengekommen. Sie wurden initiiert durch Fragen der Teams und wurden dann an alle ausgeteilt, um evtl. noch weitere Möglichkeiten zu eröffnen bzw. Ideen zu geben. Sie finden diese Unterlagen im Material (04\_alg\_hilfekarten\_projekte\_ai2).

<sup>6</sup> Einen Film dazu, der extra für Schülerinnen und Schüler gedreht wurde, finden Sie unter URL: <https://appcamps.de/> (abgerufen am 20.04.2018). Eine Registrierung für Lehrkräfte ist kostenlos. Sie haben dann Zugriff auf unterschiedliche Lehrgänge. Der Film ist Startfilm zur Einführung in den App Inventor.



## Vernetzung mit dem mathematischen Teil von IMP

Es bietet sich an, die erworbenen Programmierkenntnisse immer wieder einzusetzen und damit wachzuhalten. Das ist vor allem immer wieder im Bereich der Mathematik in IMP möglich.

Beispiele sind die Bestimmung des ggT mit dem euklidischen Algorithmus, das Sieb des Eratosthenes zur Bestimmung von Primzahlen oder das Finden der Primfaktorzerlegung einer Zahl.

In den Materialien finden Sie Apps zu diesen Beispielen, außerdem eine App, bei der es um Farbcodierung und Umwandlung in Hexadezimalzahlen geht und eine App mit einem Umfüllrätsel.

Scratch wird von der Lifelong-Kindergarten-Group am MIT-Media-Lab entwickelt. Siehe <http://scratch.mit.edu>.  
Scratch ist lizenziert unter *CC BY-SA 2.0* (<https://creativecommons.org/licenses/by-sa/2.0/deed.en>).



## Literaturliste und Internetseiten:

### Scratch:

Immler, Ch.; Der kleine Hacker – Programmieren für Einsteiger; München; Franzis Verlag; 2016

Sweigart, Al; Coole Spiele mit Scratch; dt. Ausgabe; Heidelberg; dpunkt.verlag; 2017

Vorderman, C., Dr. Woodcock, J.; Spiele programmieren – super easy; dt. Ausgabe; München; Dorling Kindersley Verlag; 2016

Vorderman, C., Dr. Woodcock, J.; Kreative Projekte mit Scratch – super easy programmieren; dt. Ausgabe; München; Dorling Kindersley Verlag; 2017

Dr. Woodcock, J.; Star Wars Spiele programmieren; dt. Ausgabe; München; Dorling Kinderley Verlag; 2018

<https://scratch.mit.edu/> (abgerufen am 20.04.2018)

<https://appcamps.de/> (abgerufen am 20.04.2018)

### MIT-App-Inventor 2:

Bergner, N., Leonhardt, Th.; Eigene Apps programmieren – für Dummies Junior; Weinheim; WILEY-VCH Verlag; 2016

Rollke, K.-H.; Android Apps mit AppInventor2 – Jeder kann programmieren; Sundern; Selfpublishing; 2017

<http://appinventor.mit.edu/explore/> (abgerufen am 20.04.2018)

<https://appcamps.de/> (abgerufen am 20.04.2018)

<http://appinventor.mit.edu/extensions/> (abgerufen am 14.08.2018) – Erweiterungen für den MIT App Inventor

### Allgemeines

Boockmeyer, Fischbeck, Neubert; Fit fürs Studium Informatik; Bonn; Rheinwerk Verlag; 2017

Gallenbacher, J.; Abenteuer Informatik; 4. Auflage; Heidelberg; Springer-Verlag; 2017

<https://www.inf-schule.de/> (abgerufen am 20.04.2018)