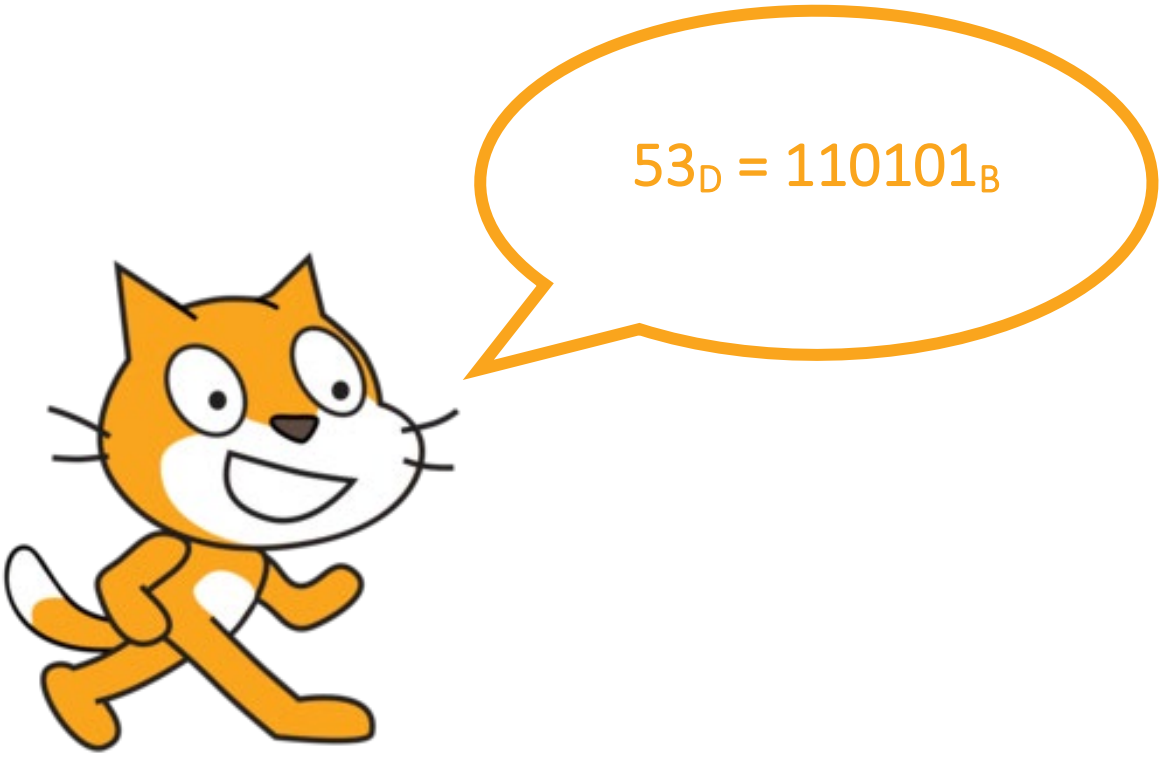
HOPP FOUNDATION
for computer literacy & informatics

> Scratch im Informatikunterricht – mehr als Spielerei



Quelle: https://upload.wikimedia.org/wikipedia/commons/6/66/Scratch_Cat_Pictture.png (CC BY-SA)

Autor: Christian Stemberg (2023)

Kontakt: cstemberg@gmx.net

Inhaltsverzeichnis

1 Einführung

Seit der Einführung des Faches Informatik in Baden-Württemberg ab Klasse 7 unterrichte ich das Fach an einer Gesamtschule in Baden-Württemberg. Im Austausch mit Informatik-Lehrkräften ist mir aufgefallen, dass im Teilbereich „Algorithmen“ des Bildungsplans[1] bei der Programmierung meist darauf gesetzt wird, dass Schüler[2] Spiele programmieren. Während ich diesen Ansatz selbst längere Zeit im Unterricht verfolgt habe, setze ich im Bereich der Programmierung in den Klassen 7 und 8 andere Schwerpunkte und verzichte im Fachunterricht darauf, mit Schülern Spiele zu programmieren. Dieses Skript soll interessierten Informatik- und Mathematiklehrern diesen anderen Ansatz – keine Spiele im Informatikunterricht – erläutern (vgl. 1.2).

1.1 Spiele programmieren im Informatikunterricht: Bildungsplanbezug

Das Programmieren von Spielen stellt zwar im Informatikunterricht einen durchaus motivierenden Zugang für Schüler zur Programmierung dar, doch es fehlt dann die in 1.2 formulierte Umsetzung von Aufgabenstellungen aus anderen Teilbereichen der Informatik, z. B. der Codierung, in die Algorithmen.

Um nicht falsch verstanden zu werden: Beide Ansätze können je nach Perspektive ihre Berechtigung haben, z. B. im Hinblick auf Differenzierung. Denn auch mit dem Programmieren von Spielen lassen sich die im o. g. Bildungsplan formulierten Kompetenzen erreichen. So lauten die für den Teilbereich „Algorithmen“ formulierten Kompetenzen im M-Niveau: „1) die algorithmischen Grundbausteine Anweisung, Sequenz, Schleife/Wiederholung, Verzweigung und Bedingung benennen“; „(2) Algorithmen als Verknüpfung von Anweisungen und Kontrollstrukturen beschreiben“; „(3) Beispiele für die Verwendung von Variablen (z. B. als Speicher für Punktestand, Rundenzähler in Spielen etc.) nennen“; „(4) an gegebenen Algorithmen Veränderungen durchführen“ sowie „(5) einfache Algorithmen in einer geeigneten (z.

[1] https://www.bildungsplaene-bw.de/,Lde/LS/BP2016BW/ALLG/SEK1

[2] nachfolgend der Einfachheit halber nur „Schüler“ genannt, ohne dabei ein Geschlecht bewusst zu diskriminieren (die falsche Schreibweise „SchülerInnen“ lehne ich als Deutschlehrer ab).

B. visuellen) Programmierumgebung implementieren und dabei Variablen und algorithmische Grundbausteine zielorientiert anwenden“[3].

Zusammenfassend lässt sich also sagen, dass der spielerische Zugang zur Programmierung kein „falscher“ ist. Meiner Meinung nach aber eröffnet er Schülern häufig nicht die Einblicke in Scratch, die nötig sind, um die einzelnen Teilbereiche des Informatikunterreichs besser zu verzahnen und/oder Schüler in die Lage zu versetzen, informatische und mathematische Probleme durch Programmierung zu lösen. Ich plädiere dafür, bereits in Klasse 7 einen anderen Ansatz zu verfolgen, den ich nachfolgend genauer beschreiben möchte.

1.2 Alternativer Ansatz: Keine Spiele im Informatikunterricht

Der alternative Ansatz setzt konsequent darauf, Aufgaben aus den Teilbereichen der Informatik, die in der Regel theoretisch bzw. mit Hilfe von Arbeitsblättern bearbeitet werden, mit Hilfe von Scratch in der Programmierung umzusetzen. Dabei müssen sich die Schüler zunächst intensiv mit der Logik des Programmablaufs, z. B. zunächst in Form eines Programmablaufplans[4] (PAP) auseinandersetzen, um die Aufgabe dann in Scratch zu programmieren. Nicht wenigen Schüler fällt es meiner Erfahrung nach schwer, sich in die Logik eines Algorithmus einzuarbeiten. Indem die Schüler immer Aufgaben aus dem Informatikunterricht in Scratch programmieren, setzen sie sich meiner Erfahrung nach auch intensiver mit diesen auseinander, sodass der Lernerfolg größer ist. Gleichzeitig wird das Programmieren im Laufe des Schuljahres immer wieder geübt und nicht – wie es manchmal der Fall ist – als isolierter einzelner Block im Fachunterricht abgehandelt.

Haben die Schüler einmal die unter „Operatoren“ bei Scratch zu findenden Bausteine erlernt, lassen sich auch mathematische Rätsel umsetzen, an denen die Schüler ihre Programmierkenntnisse vertiefen können; der Ehrgeiz, diese Aufgaben zu knacken und

[3] Bildungsplan 2016 „Aufbaukurs Informatik, S. 19“, verfügbar unter: https://www.bildungsplaene-bw.de/,Lde/LS/BP2016BW/ALLG/SEK1/INF7

[4] vgl. https://de.wikipedia.org/wiki/Programmablaufplan

mit einem Algorithmus eine Lösung für ein mathematisches Problem zu finden, kann mindestens genauso viel Spaß machen wie das Programmieren eines Spiels und fördert dabei – häufig unterbewusst – informatische und mathematische Kompetenzen.

Man kann sich nun fragen, weshalb dieser Ansatz im Informatikunterricht nicht häufiger verfolgt wird. Eine Vermutung ist, dass sich für die Spieleprogrammierung mit Scratch im Internet zahlreiche „fertige“, also einsatzbereite Tutorials und Kurse mit direkt einsetzbarem Unterrichtsmaterial finden. Vielleicht aber haben Lehrkräfte auch Bedenken, dass die Motivation der Schüler leidet, wenn man sie mit diesen scheinbar langweiligeren Themen begegnet. Und nicht zuletzt könnte es auch die Lehrkraft selbst sein, die sich das nicht zutraut oder zu wenig Kenntnisse im Programmieren hat. Es gibt aufgrund des Lehrermangels gerade im Bereich des Informatikunterrichts zahlreiche Lehrkräfte, die das Fach Informatik selbst nicht studiert haben und fachfremd unterrichten (müssen). Vielleicht trauen sie sich zwar die Betreuung der Schüler beim Programmieren von (einfachen) Spielen zu, kennen aber selbst vielleicht die Funktion der Operator-Bausteine in Scratch nicht, die benötigt werden, um informatische und/oder mathematische Problemstellungen durch Programmierung zu lösen.

Dieses Skript soll Lehrkräften dabei helfen, mit Hilfe von Beispielen das Konzept in den eigenen Unterricht zu übertragen. Diese Beispiele orientieren sich am Bildungsplan Informatik des Landes Baden-Württemberg für die Fächer „Aufbaukurs Informatik 7“ und Wahlfach Informatik 8.

Die Links am Ende des Skriptes sollen helfen, weitere Aufgaben mit verschiedenen Schwierigkeitsgraden für Schüler (oder Lehrkräfte selbst) zu finden. Meine eigenen Erfahrungen mit dem Konzept sind durchweg positiv, sodass ich dazu ermuntern möchte, die Programmieraufgaben im Unterricht umzusetzen.

2 Bildungsplanbezüge Informatik (Klassen 7 und 8)

2.1 Bildungsplanbezug Klasse 7

Im Bildungsplan für das Fach „Aufbaukurs Informatik Klasse 7[5]“ sollen die Schüler Algorithmen als „formalisierte Handlungsanweisungen zur Lösung von Problemen kennen“, Algorithmen „mithilfe elementarer Anweisungen und Kontrollstrukturen beschreiben“ und „Variablen als änderbaren Wertespeicher kennen“. Die Schüler sollen dabei die Funktionsweise eines Computers als eine Maschine, die Algorithmen ausführt, erfahren und dabei den „Gesamtprozess von der Problemanalyse über das Aufstellen von Algorithmen bis hin zur Implementierung und Verifizierung (…) kennen.“ Konkret können die Schüler am Ende von Klasse 7 (M-Niveau) …

- ✓ „(1) die algorithmischen Grundbausteine Anweisung, Sequenz, Schleife/Wiederholung, Verzweigung und Bedingung erläutern“;
- ✓ „(2) Algorithmen als Verknüpfung von Anweisung und Kontrollstrukturen beschreiben“;
- ✓ „(3) Variablen als änderbaren Wertespeicher (…) erläutern“;
- ✓ „(4) Algorithmen zu gegebenen einfachen Problemstellungen entwerfen“;
- ✓ „(5) Algorithmen (…) implementieren und dabei Variablen und algorithmische Grundbausteine zielorientiert anwenden“;
- ✓ „(6) in grafischer Form (z. B. als Flussdiagramm) dargestellte Algorithmen erklären“;
- ✓ „(7) Codeabschnitte schrittweise untersuchen und deren Wirkung interpretieren“.

2.2 Bildungsplanbezug Klasse 8

Der Bildungsplan für das Fach „Informatik (Wahlfach)“[6] für Klasse 8 baut auf den Inhalten von Klasse 7 auf; die Schüler sollen nun die gelernten Grundbausteine zu

[5] https://www.bildungsplaene-bw.de/,Lde/LS/BP2016BW/ALLG/SEK1/INF7
[6] https://www.bildungsplaene-bw.de/,Lde/LS/BP2016BW/ALLG/SEK1/INFWF

immer komplexeren Programmen zusammenfügen. So kommen beispielsweise logische Operatoren, Zufallszahlen, aber auch Arrays und Listen hinzu. Auch sollen Unterprogramme genutzt werden, um die Redundanz von Code zu vermeiden.

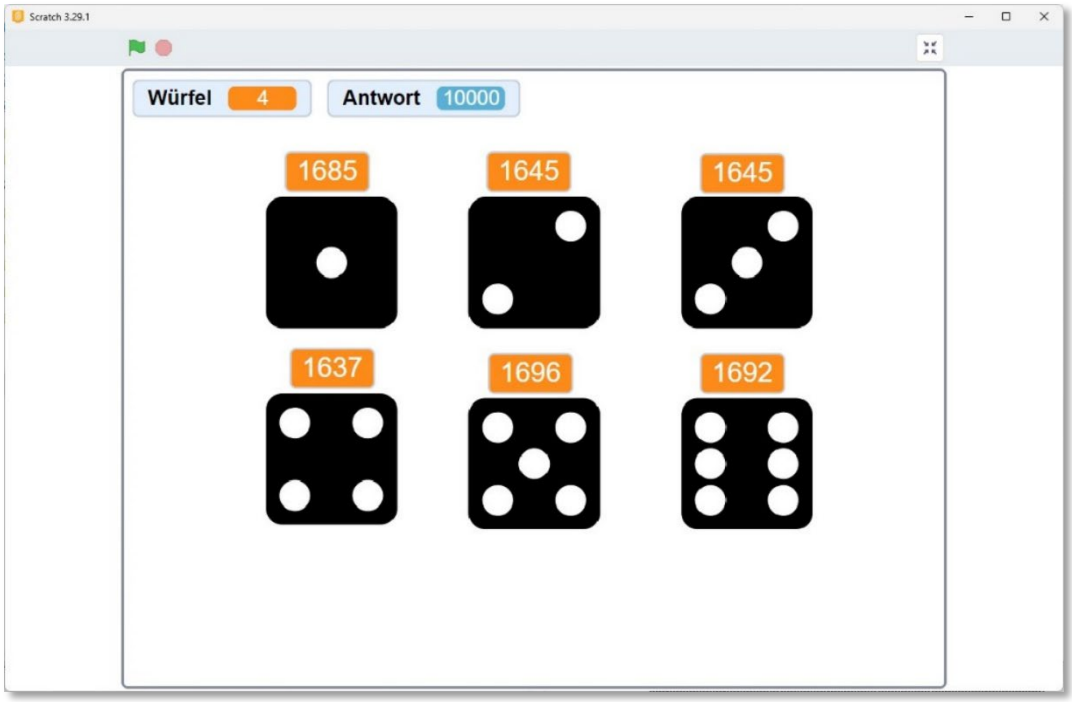
Konkret können die Schüler am Ende von Klasse 8 (M-Niveau) …

- ✓ „(1) Logische Verknüpfungen (UND, ODER, NICHT) in Bedingungen von Schleifen und Verzweigungen verwenden“;
- ✓ „(2) Zufallszahlen in eigenen Programmen verwenden (z. B. um Würfelergebnisse zu simulieren oder einen Spielverlauf abwechslungsreicher zu gestalten)“;
- ✓ „(3) eine indexbasierte Datenstruktur zur Speicherung und Verarbeitung gleichartiger Daten (auch per Iteration) verwenden“;
- ✓ „(4) grundlegende Algorithmen auf einer indexbasierten Datenstruktur (z. B. Füllen mit Werten, Maximumsuche, Summenbildung) beschreiben und implementieren“;
- ✓ „(5) Unterprogramme verwenden, um Programmcode zu strukturieren und redundanten Code zu vermeiden“.

Die nachfolgenden Projekte sollen helfen, die in den Bildungsplänen für die Klassen 7 und 8 formulierten Kompetenzen bei den Schülern anzubahnen. Die Projekte werden zunehmend komplexer; auch in den Beispielen für Klasse 7 werden Kompetenzen für die 8. Klasse angebahnt (z. B. logische Operatoren).

3 Scratch-Projekt 1: Häufigkeitsverteilung von Würfeln (Klasse 7)

Im ersten Projekt wird die Häufigkeitsverteilung bei einer beliebigen Anzahl von Würfen mit einem 6-seitigen Standardwürfel ermittelt; im Beispiel 10000 Würfe:



3.1 Angebahnte Kompetenzen

Der Schwerpunkt dieses Programmierprojektes liegt wiederum im Lesen des Programmablaufplanes (PAP) sowie in der Umsetzung des Algorithmus in Scratch mit Hilfe von Eingaben, Schleifen, Variablen, Zufallszahlen, Vergleichsoperatoren sowie Bedingungen[7].

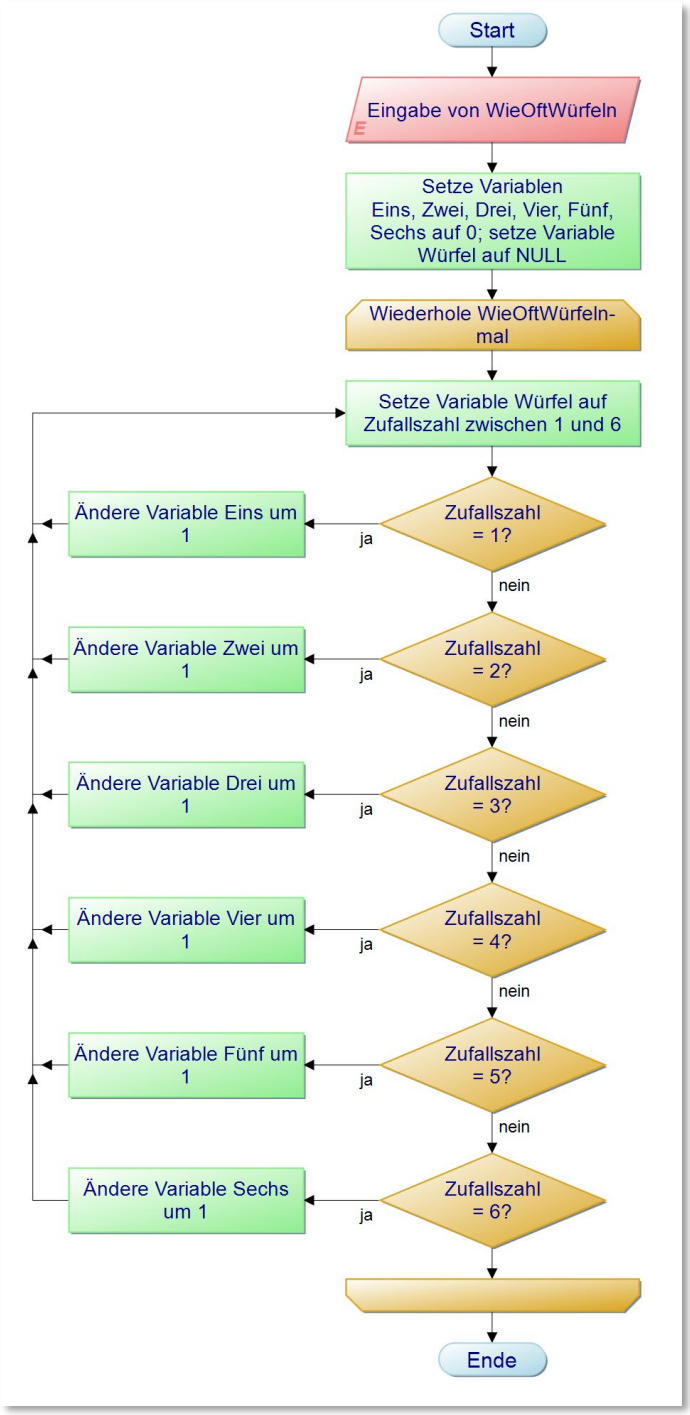
3.2 Programmablaufplan

Der Programmablaufplan (PAP) macht deutlich, wie einfach der Algorithmus aufgebaut und zu verstehen ist. Mit Schülern lässt sich dieser PAP schnell erarbeiten und im Anschluss in Scratch implementieren.

Für die Implementierung müssen Schüler mit der Anlage von Variablen vertraut gemacht werden. Darüber hinaus sollte die Funktion einer Variablen erläutert werden;

[7] Bildungsplan für das Fach Informatik (M-Niveau) in Klasse 7 (vgl. 2.1, 1 und 3).

im Beispiel werden Variablen für die gewürfelten Augenzahlen (Eins, Zwei, Drei, Vier, Fünf und Sechs) benötigt. Auch der Zufallszahlen-Baustein spielt eine wichtige Rolle, auch er sollte erläutert werden. Mit Hilfe einer Eingabe und den Kontrollstrukturen (Wiederholen-Schleife, Falls-Baustein) kann das Programm dann extrem schnell die Häufigkeitsverteilung einer beliebigen Anzahl von Würfen darstellen.



Eingabe: Wie oft soll gewürfelt werden?

Variablen initialisieren.

Kontrollstruktur: Wiederhole so oft wie Eingabe.

Variable Würfel auf Zufallszahl zwischen 1 und 6 setzen.

Verzweigung: falls Zufallszahl == 1, ändere Variable Eins um 1.

Verzweigung: falls Zufallszahl == 2, ändere Variable Zwei um 1.

Verzweigung: falls Zufallszahl == 3, ändere Variable Zwei um 1.

Verzweigung: falls Zufallszahl == 4, ändere Variable Zwei um 1.

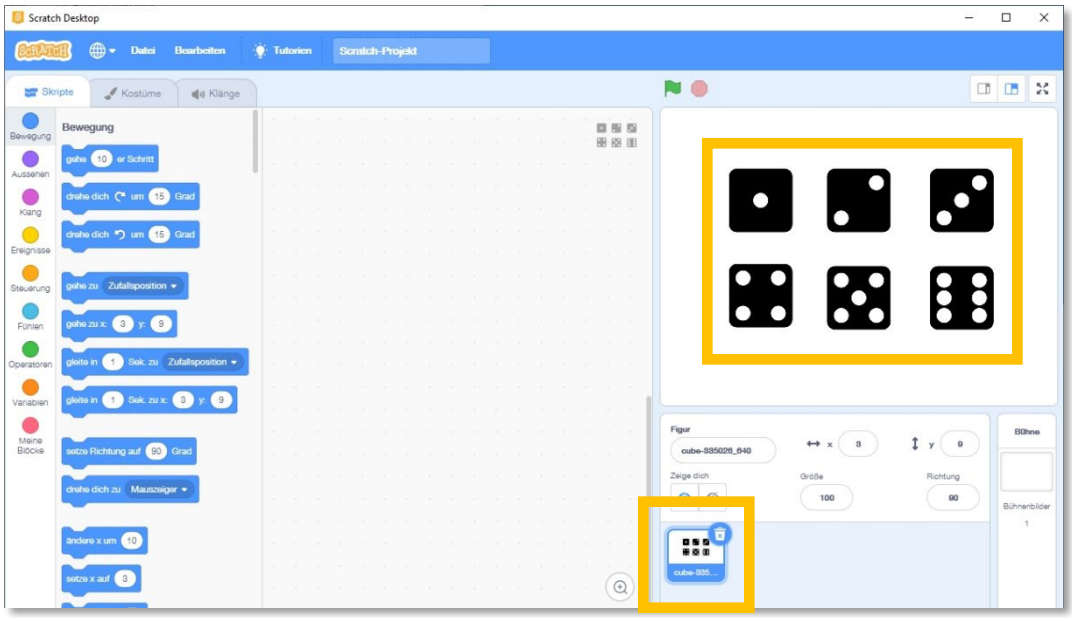
Verzweigung: falls Zufallszahl == 5, ändere Variable Zwei um 1.

Verzweigung: falls Zufallszahl == 6, ändere Variable Zwei um 1.

Nachfolgend wird die Implementierung mit Scratch ausführlich erläutert.

3.3 Scratch starten, Figur und Hintergrund einfügen

Nach dem Start von Scratch sollten zunächst eine passende(re) Figur und ggf. ein Hintergrund eingefügt werden. Im Internet gibt es zahlreiche Anlaufstellen für freie Grafiken; ich verwende in meinem Unterricht gerne Cliparts der englischsprachigen Seite openclipart.org[8] oder von pixabay[9]. Hier ist die Würfelgrafik[10] als Figur in Scratch eingefügt worden:



3.4 Variablen anlegen und bei Programmstart initialisieren

Im nächsten Schritt werden die Variablen angelegt, welche die gewürfelten Augenzahlen zählen. Es werden sieben Variablen benötigt: Eine Variable „Würfel“, die bei jedem Durchlauf der Schleife auf einen zufälligen Wert zwischen 1 und 6 gesetzt wird und sechs Variablen „Eins“ bis „Sechs“, die zählen, wie oft die jeweilige Augenzahl gewürfelt wurde.

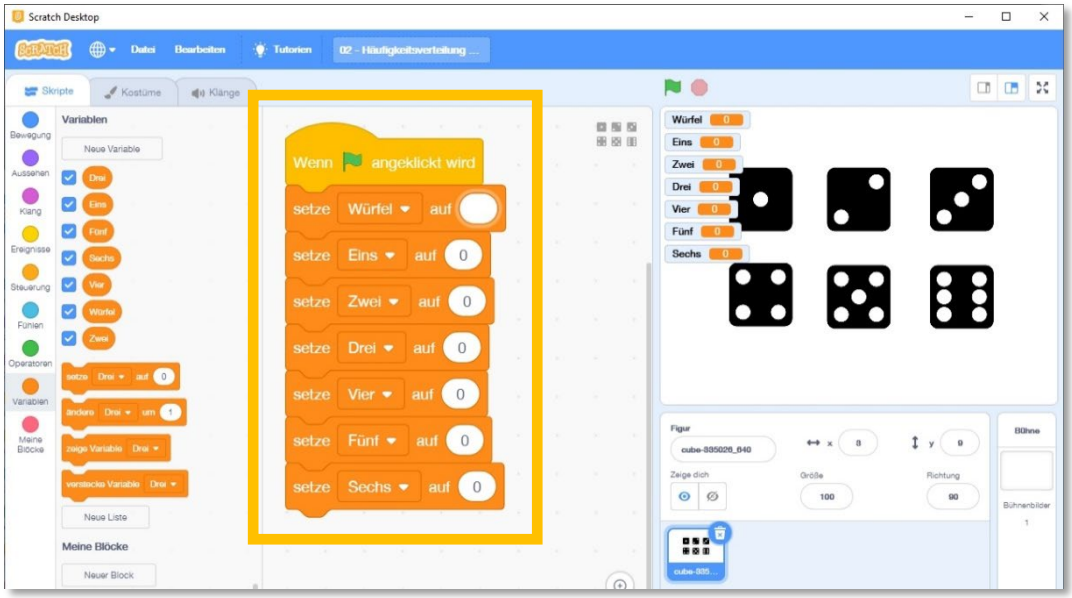
Bei Programmstart werden die Variablen „Eins“ bis „Sechs“ auf den Wert 0 gesetzt, die Variable „Würfel“ auf NULL (leer).

---

[8] openclipart.org: https://openclipart.org/

[9] pixabay: https://pixabay.com/de/

[10] https://pixabay.com/vectors/dice-one-two-three-four-five-six-335026/ (11.10.2022)



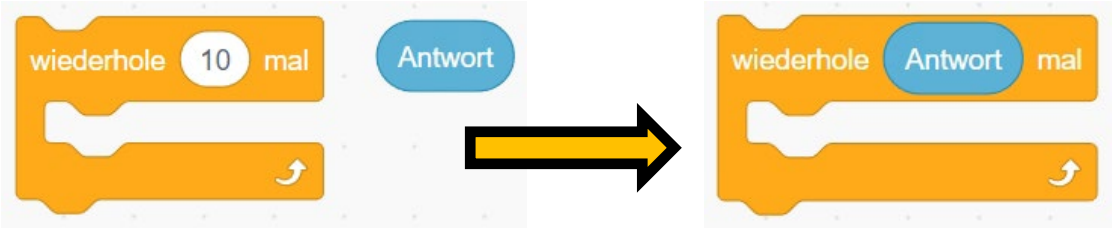
Die Variablen auf der Bühne lassen sich durch einen Rechtsklick in die sogenannte Großanzeige umschalten und über der eingefügten Grafik arrangieren:

3.5 Eingabe: Wie oft soll gewürfelt werden?

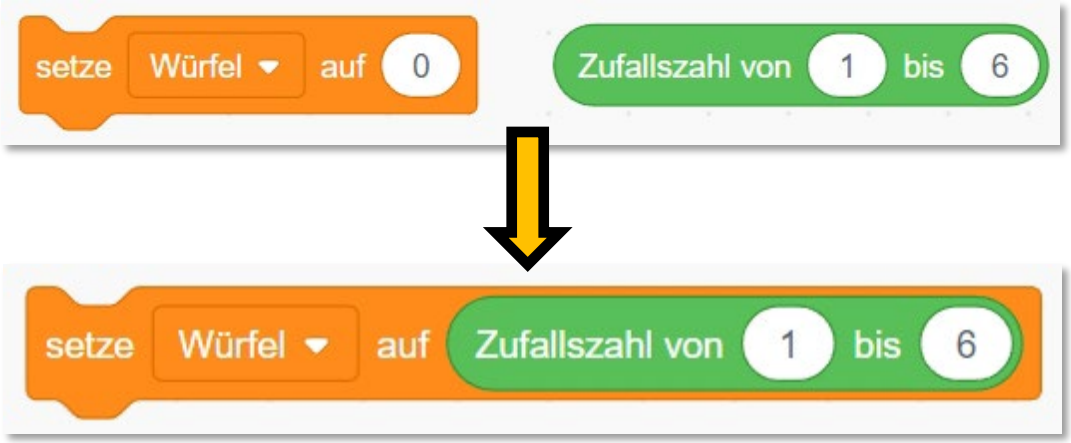
Im nächsten Schritt erfolgt die Abfrage, wie oft gewürfelt werden soll. Die Eingabe wird in der unter „Fühlen“ zu findenden Variablen „Antwort“ gespeichert:

3.6 Wiederholen-Schleife und Wertzuweisung an Variablen

Nun fehlt noch die Schleife, die so oft durchlaufen wird wie bei der Eingabe angegeben:



Innerhalb der Schleife wird bei jedem Durchlauf die Variable „Würfel“ auf eine Zufallszahl[11] zwischen 1 und 6 gesetzt:



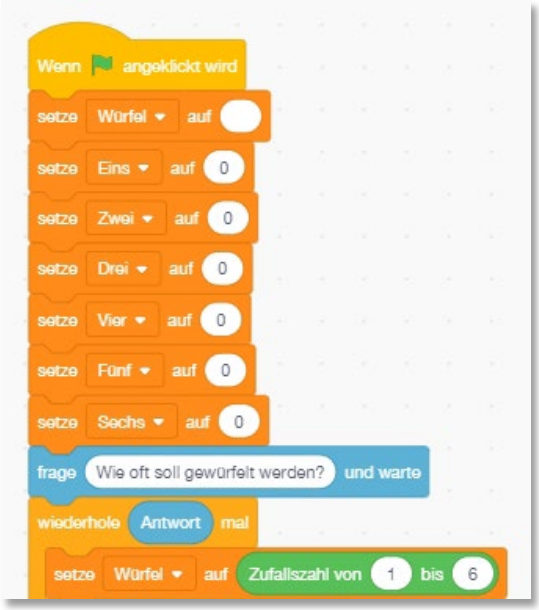
3.7 Verzweigungen („Falls“) einfügen

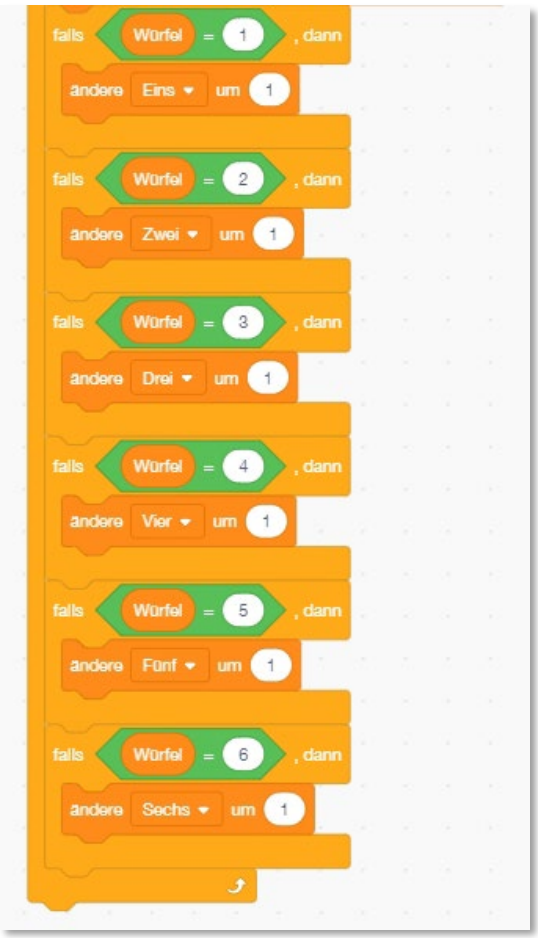
Wenn das Programm nun gestartet wird, kann man sehen, dass die Variable „Würfel“ sehr schnell hintereinander verschiedene Werte zwischen 1 und 6 annimmt. Um zu zählen, wie oft eine Eins, eine Zwei, eine Drei usw. ermittelt wurde, muss mit Hilfe von Verzweigungen gearbeitet werden. Nachfolgend nur 2 der 6 zu unterscheidenden Fälle; für die anderen wird für die anderen Möglichkeiten der Zufallszahl entsprechend unterschieden:

[11] Baustein „Zufallszahl“: Unter „Operatoren“ zu finden.

3.8 Das fertige Programm (Scratch-Code)

Der Übersichtlichkeit halber wurde der Screenshot des fertigen Programms auf zwei Tabellenzeilen aufgeteilt.



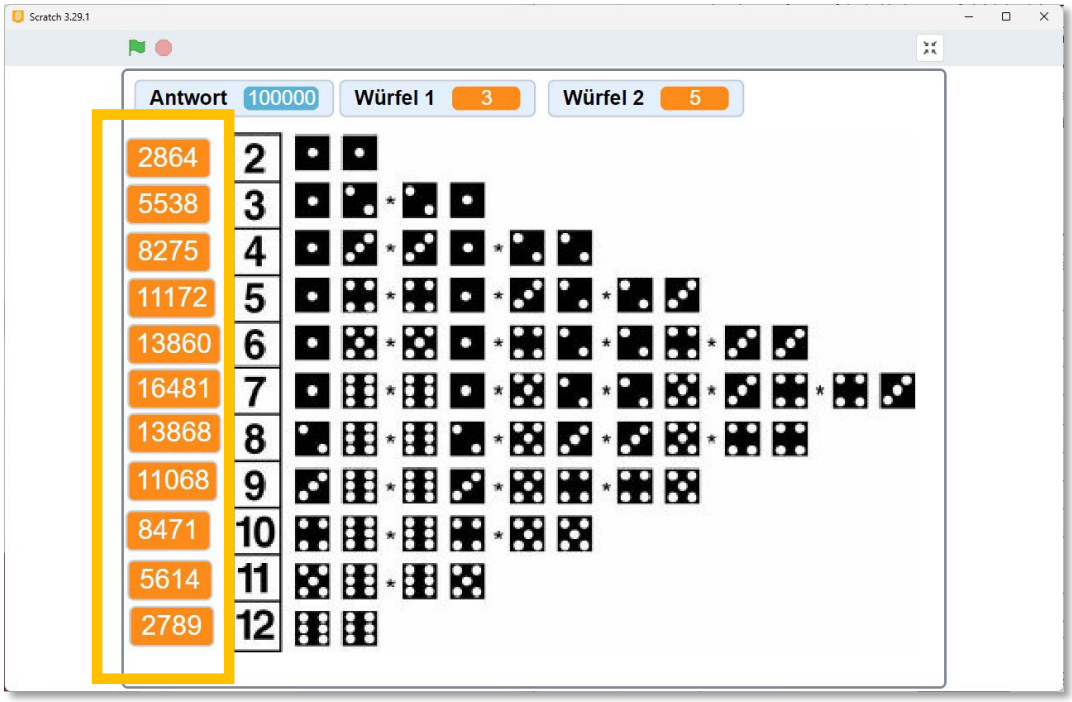


3.9 Differenzierung/Idee zum Weiterarbeiten

Es stellt nicht viel Aufwand dar, das Programm zu erweitern. Bei der Häufigkeitsverteilung der Augenzahl bei einem Würfel sieht man, dass das Verhältnis zwischen allen Möglichkeiten ausgeglichen ist, da die Wahrscheinlichkeit, eine Eins oder eine Fünf zu würfeln, bei jedem Wurf gleich hoch ist.

Anders verhält es sich aber beim Würfeln mit 2 Würfeln, denn hier sind die Wahrscheinlichkeiten ganz anders verteilt. So kann die Würfelsumme „2“ nur dann erreicht werden, wenn beide Würfel eine Eins anzeigen. Die Chance auf eine Eins liegt bei einem Würfel bei 1/6. Und beide zeigen entsprechend bei 1/6 * 1/6 = 1/36 eine Eins an. Genauso groß ist die Chance auf die Würfelsumme von „12“ bei zwei Sechsen (1/36). Größer sind die Chancen bei anderen Würfelsummen. Die größte Wahrscheinlichkeit besteht bei der Würfelsumme „7“, denn hier gibt es sechs verschiedene Kombinationen für die genannte Würfelsumme[12].

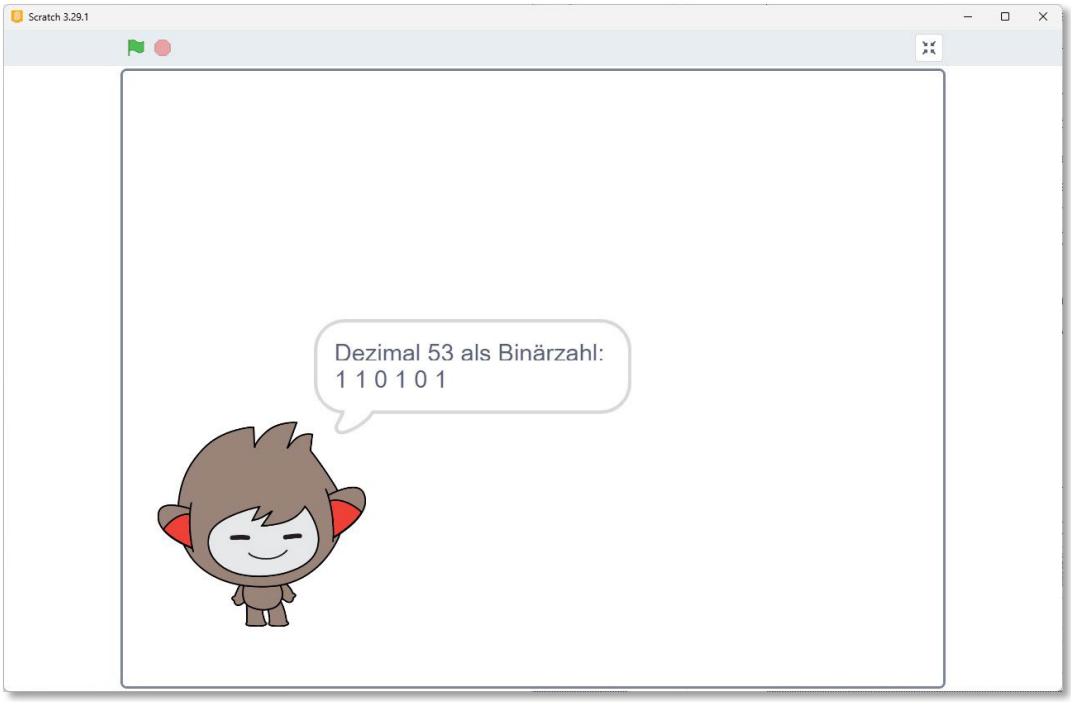
Das sieht man deutlich, wenn man häufig würfeln lässt und diese Verteilung z. B. mit Hilfe von Variablen zählen lässt.



[12] Würfelsumme „7“ lässt sich würfeln durch: Eins + Sechs; Zwei + Fünf; Drei + Vier; Vier + Drei; Fünf + Zwei; Sechs + Eins = 6 Kombinationen.

4 Scratch-Projekt 2: Dezimalzahl in Binärzahl umwandeln (Klassen 7/8)

Beim zweiten Projekt wird eine eingegebene Dezimalzahl in eine Binärzahl umgewandelt:



Dieses Programmierprojekt passt zum Themenbereich „Daten und Codierung“ im Bildungsplan für den Aufbaukurs Informatik in Klasse 7[13] und 8[14] nach dem Schüler u. a. „natürliche Zahlen (…) mithilfe des Binärsystems als Bitfolge darstellen und Bitfolgen als Zahlen interpretieren“ und „Zahlen in Hexadezimaldarstellung identifizieren und mit geeigneten Hilfsmitteln (…) in Dezimalzahlen umwandeln“ können. Nicht selten wird dieser Themenbereich rein theoretisch behandelt. Durch die intensive Auseinandersetzung mit dem Algorithmus zur Umwandlung von Dezimal- in Binärzahlen und der sich daran anschließenden Programmierung werden die beiden Themenbereiche „Daten und Codierung“ sowie „Algorithmen“ verknüpft und das bereits theoretisch Erlernte und Geübte vertieft.

[13] https://www.bildungsplaene-bw.de/,Lde/LS/BP2016BW/ALLG/SEK1/INF7
[14] https://www.bildungsplaene-bw.de/,Lde/LS/BP2016BW/ALLG/SEK1/INFWF

Das zweite Projekt ist anspruchsvoller als das erste, weil hier unter anderem der für Schüler nicht immer einfach zu verstehende Modulo-Operator[15] ins Spiel kommt. Mit diesem wird der ganzzahlige Rest einer Division ermittelt. Auch kommen neben Variablen erstmals Listen zum Einsatz.

4.1 Angebahnte Kompetenzen

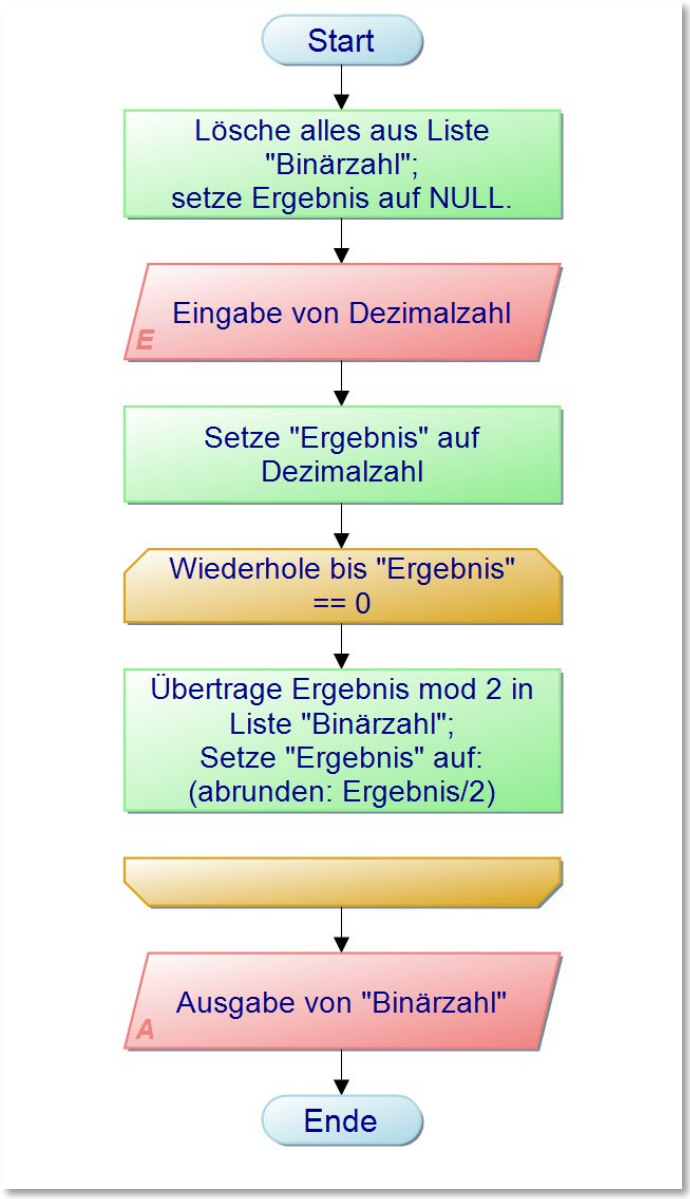
Der Schwerpunkt dieses Programmierprojektes liegt wiederum im Lesen des Programmablaufplanes sowie in der Umsetzung des Algorithmus in Scratch mit Hilfe von Eingaben, Schleifen, Variablen, Listen sowie dem Modulo-Befehl[16].

4.2 Programmablaufplan

Der Programmablaufplan macht deutlich, dass der Umwandlung einer Dezimal- in eine Binärzahl ein sehr einfacher Algorithmus zugrunde liegt. Die Dezimalzahl, die umgewandelt werden soll, wird immer wieder durch 2 geteilt und der ganzzahlige Rest notiert. Sobald man beim Teilen beim Ergebnis „0“ angekommen ist, terminiert der Algorithmus. Die notierten ganzzahligen Reste ergeben – richtig herum abgelesen – die umgewandelte Binärzahl:

[15] vgl. https://de.wikipedia.org/wiki/Division_mit_Rest#Modulo

[16] Bildungsplan für das Fach Informatik (M-Niveau) in Klasse 7 (vgl. 2.1, 1 und 3).



Eingabe von Dezimalzahl.

Eingabe in die Variable „Ergebnis“ schreiben; mit dieser Variable wird gerechnet.

Setze Ergebnis auf Dezimalzahl (→ Eingabe).

Wiederholen, bis Ergebnis == 0 …

Eintrag in Liste: Ergebnis modulo 2; setze Ergebnis auf: abrunden(Ergebnis:2);

Ausgabe Liste Binärzahl.

Nachfolgend wird die Implementierung mit Scratch ausführlich erläutert.

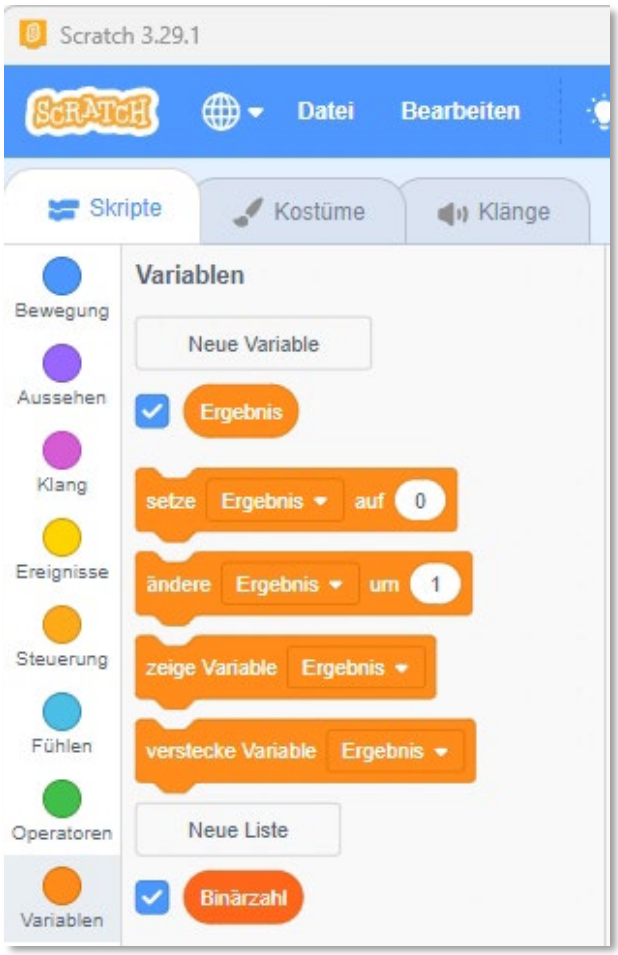
4.3 Scratch starten, Figur und Hintergrund einfügen

Nach dem Start von Scratch sollten zunächst wieder eine passende(re) Figur und ggf. ein Hintergrund eingefügt werden.

4.4 Variablen und Listen anlegen und bei Programmstart initialisieren

Im nächsten Schritt werden die Variablen angelegt. Es wird eine Variable „Ergebnis“ benötigt, mit der bei jedem Durchlauf der Schleife gerechnet wird. Darüber hinaus wird eine Liste „Binärzahl“ benötigt, die im Gegensatz zu einer Variablen mehrere Werte

speichern kann. Sie speichert in diesem Projekt den ganzzahligen Rest der Division vom aktuellen Wert der Variable „Ergebnis“ mit dem Teiler 2.



Beim Programmstart wird die Variable „Ergebnis“ auf NULL (leer) gesetzt und alle Einträge aus der Liste „Binärzahl“ werden gelöscht:

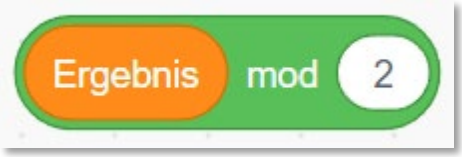
4.5 Eingabe: Dezimalzahl

Im nächsten Schritt wird die Eingabe einer Dezimalzahl benötigt. Die benötigten Bausteine sind unter „Fühlen“ zu finden; die Eingabe ist in der Variablen „Antwort“ gespeichert.

4.6 Wiederholen-Schleife

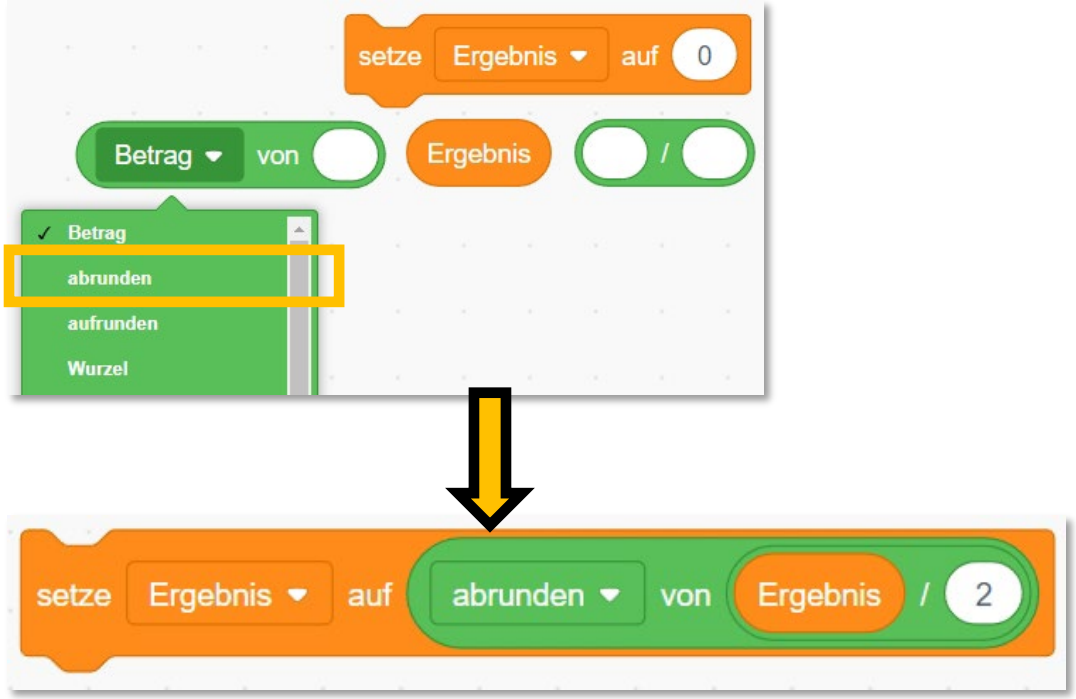
Nach den Eingaben wird nun eine Schleife benötigt, die so lange läuft, bis der Wert der Variablen „Ergebnis“ 0 ist.

Innerhalb der Schleife folgen nun 2 Berechnungen: Zunächst wird der aktuelle Wert der Variablen „Ergebnis“ modulo 2 gerechnet, um den ganzzahligen Rest dieser Division zu erhalten. Bei der Division durch 2 beträgt der ganzzahlige Rest entweder 0 oder 1, je nachdem, ob man eine gerade oder eine ungerade Zahl teilt.



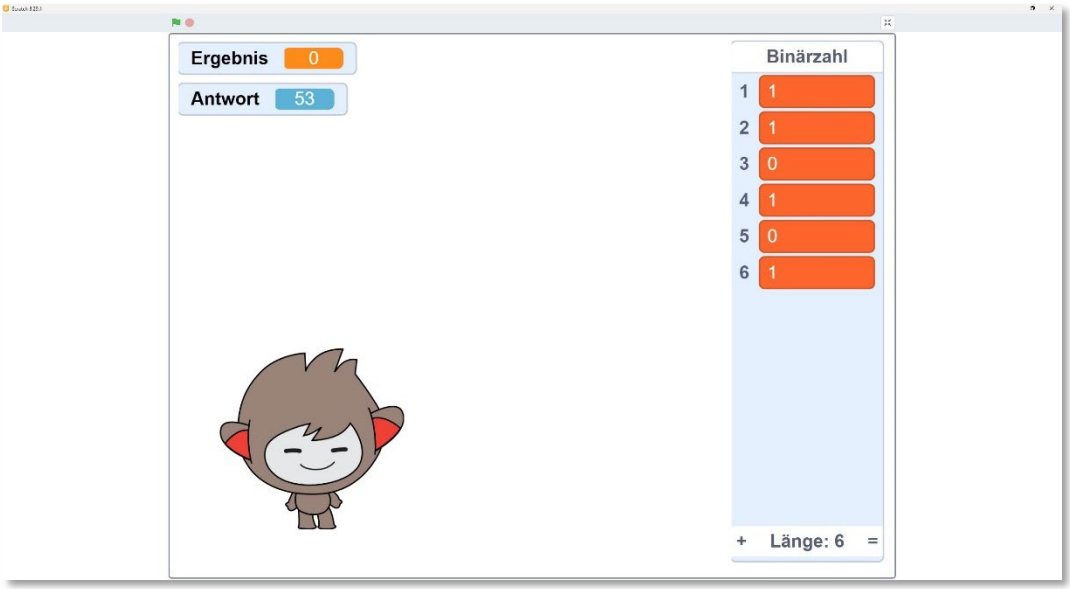
Der durch den Modulo-Baustein gefundene Rest wird nun an erster Stelle in die Liste „Binärzahl“ eingefügt.

Danach muss der aktuelle Wert der Variablen „Ergebnis“ durch 2 geteilt und der Wert von „Ergebnis“ auf diesen neuen Wert gesetzt werden. Um ganzzahlige Werte zu erhalten, muss bei der Division abgerundet werden. Hierfür werden mehrere Bausteine benötigt. Der Befehl „abrunden“ ist etwas versteckt und nicht intuitiv zu finden; hier bedarf es ggf. der Unterstützung durch die Lehrkraft.



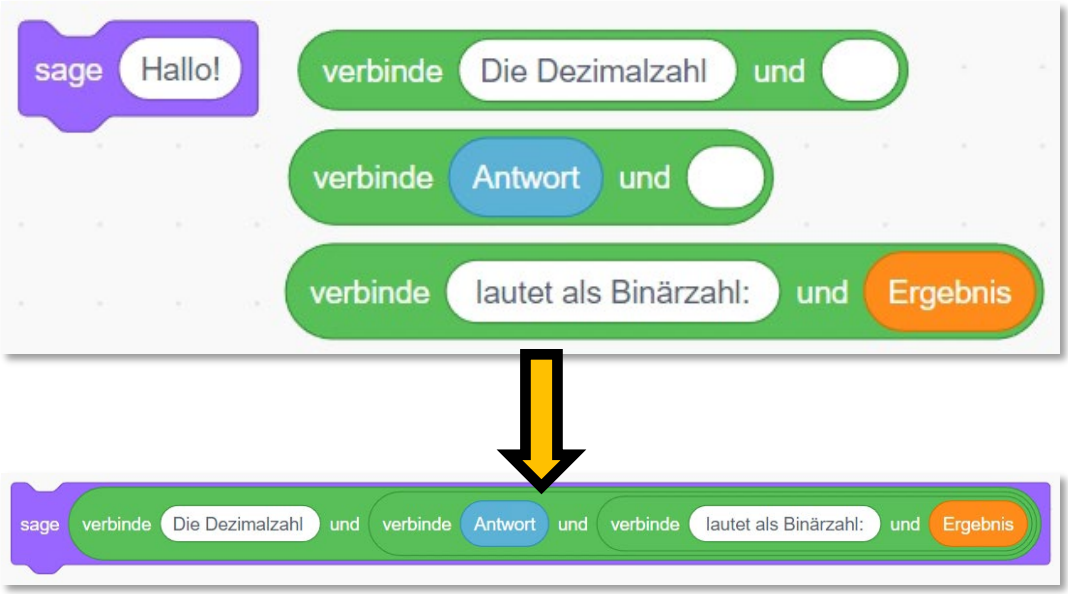
Die Schleife mit den beiden Befehlen sieht nun so aus:

Damit ist das Programm schon fast fertig. Startet man es, wird nach einer Dezimalzahl gefragt und nach deren Eingabe erscheinen die Reste in der Liste „Binärzahl“. Von oben nach unten gelesen ergeben diese Reste die eingegebene Dezimalzahl in Binärschreibweise, hier im Beispiel für die Zahl 53_D:



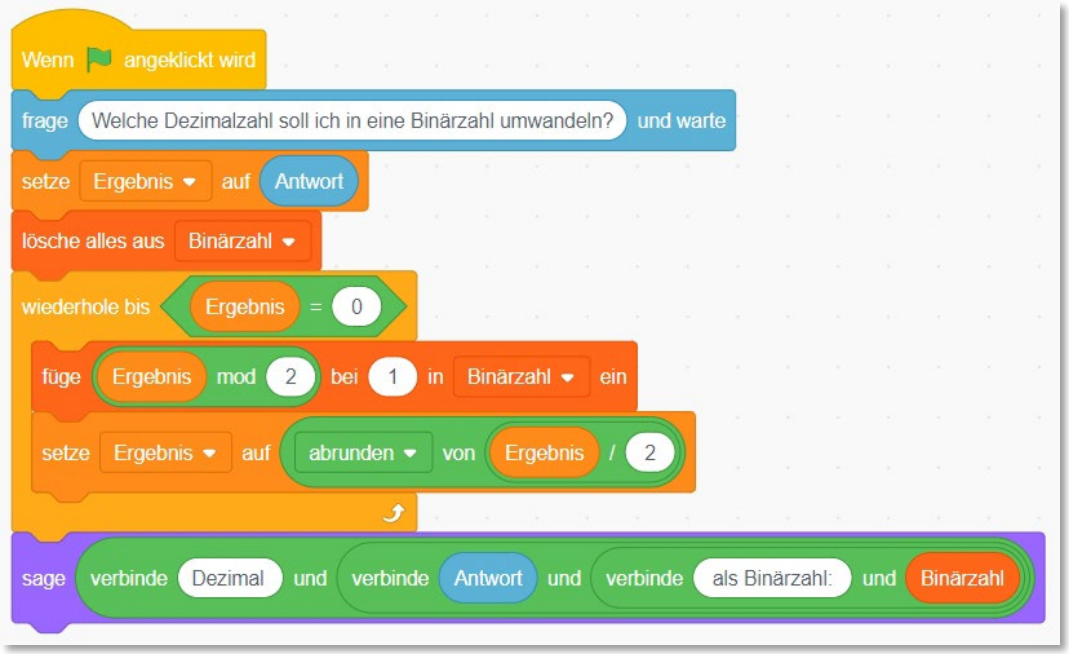
4.7 Ausgabe

Nun fehlt nur noch eine Codierung der Ausgabe. Diese darf ruhig eine ausführliche sein. Dies kann mit mehreren „Verbinden“-Bausteinen erreicht werden, die miteinander verknüpft werden. Solche Ausgaben wirken viel freundlicher, als wenn nur das Ergebnis als Binärzahl zurückgegeben wird und es lohnt sich, das Bewusstsein von Schülern dafür zu schärfen.



Zum Schluss können nicht benötigte Variablen und Listen ggf. versteckt werden.

4.8 Das fertige Programm (Scratch-Code)



4.9 Differenzierung/Idee zum Weiterarbeiten

Schüler, die das Prinzip der Umwandlung vom Dezimalsystem ins Binärsystem verstanden haben, können mit sehr kleinen Änderungen das Programm so modifizieren, dass statt in eine Binär- z. B. in eine Oktalzahl umgewandelt wird. Etwas mehr Aufwand stellt die Umwandlung in eine Hexadezimalzahl dar, denn hierfür müssen die Reste 10 bis 15 noch in die Buchstaben A bis F codiert werden.

Darüber hinaus sind viele weitere Varianten denkbar, z. B. das gleichzeitige Zählen dreier Figuren in Dezimal-, Binär-, Oktal- und Hexadezimalweise.

5 Scratch-Projekt 3: Lottozahlen ziehen (Klasse 8)

Beim dritten Projekt werden 6 Lottozahlen zwischen 1 und 49 gezogen:

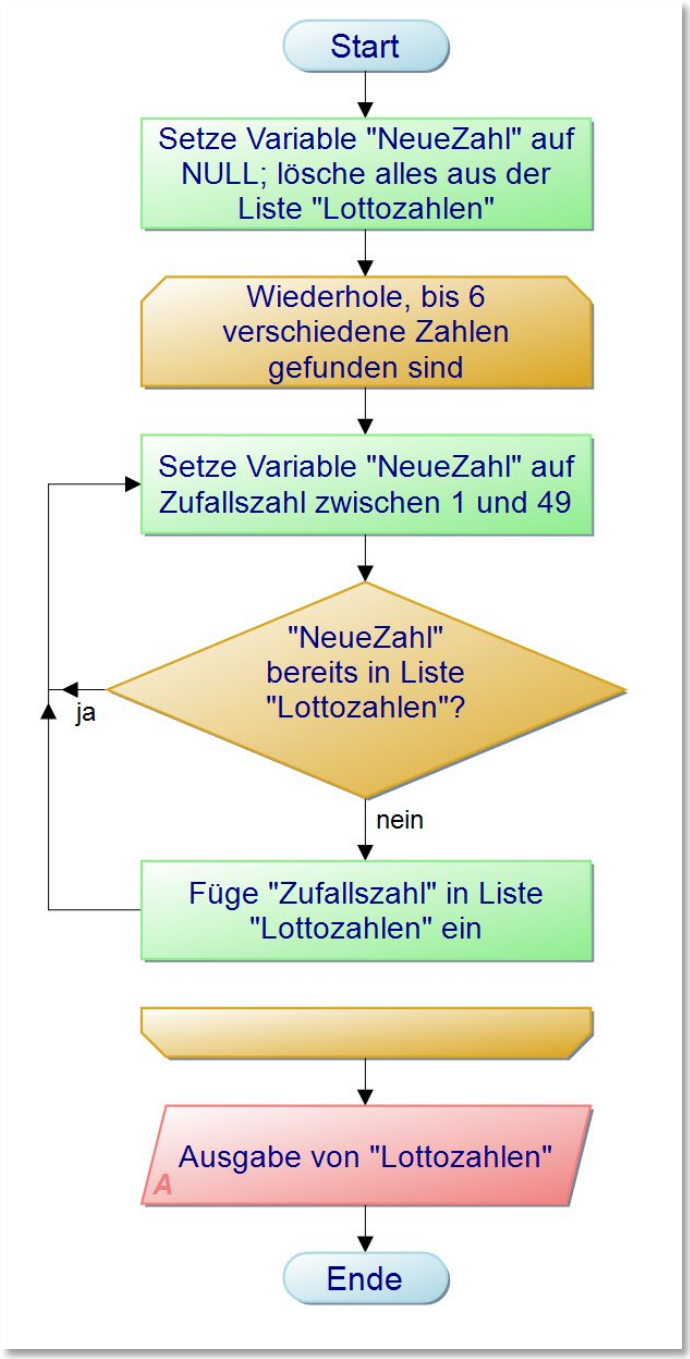
Die Schüler wenden bei diesem Projekt das bislang Erlernte (Schleifen, Variablen, Bedingungen) an und lernen darüber hinaus den Einsatz logischer Operatoren und Listen.

5.1 Angebahnte Kompetenzen

Das Projekt lässt sich zur Einführung von logischen Operatoren und von Listen beispielsweise in Klasse 8 einsetzen, da mehrere der geforderten Kompetenzen des Bildungsplans (vgl. 2.2) angebahnt werden. So müssen logische Verknüpfungen (hier: NICHT) als Bedingung in einer Schleife, darüber hinaus Zufallszahlen und die Datenstruktur Liste genutzt werden.

5.2 Programmablaufplan

Der Programmablaufplan macht deutlich, dass die Ziehung von Lottozahlen kein komplizierter Ablauf darstellt; er kann leicht als Algorithmus dargestellt und in einer Programmiersprache umgesetzt werden.



Variable „Zufallszahl“ initialisieren; Liste „Lottozahlen“ löschen.

Wiederholen, bis 6 Zahlen gezogen sind …

Variable „NeueZahl“ auf eine Zufallszahl zwischen 1 und 49 setzen.

Falls die Liste mit den Lottozahlen bereits diese Zahl enthält, wird eine neue Zufallszahl ermittelt …

… falls nicht, wird die Zufallszahl aus der Variablen „Neue Zahl“ in die Liste „Lottozahlen“ übertragen.

Ausgabe der Liste „Lottozahlen“.

Nachfolgend wird die Implementierung mit Scratch ausführlich erläutert.

5.3 Scratch starten, Figur und Hintergrund einfügen

Nach dem Start von Scratch sollten zunächst wieder eine passende(re) Figur und ggf. ein Hintergrund eingefügt werden.

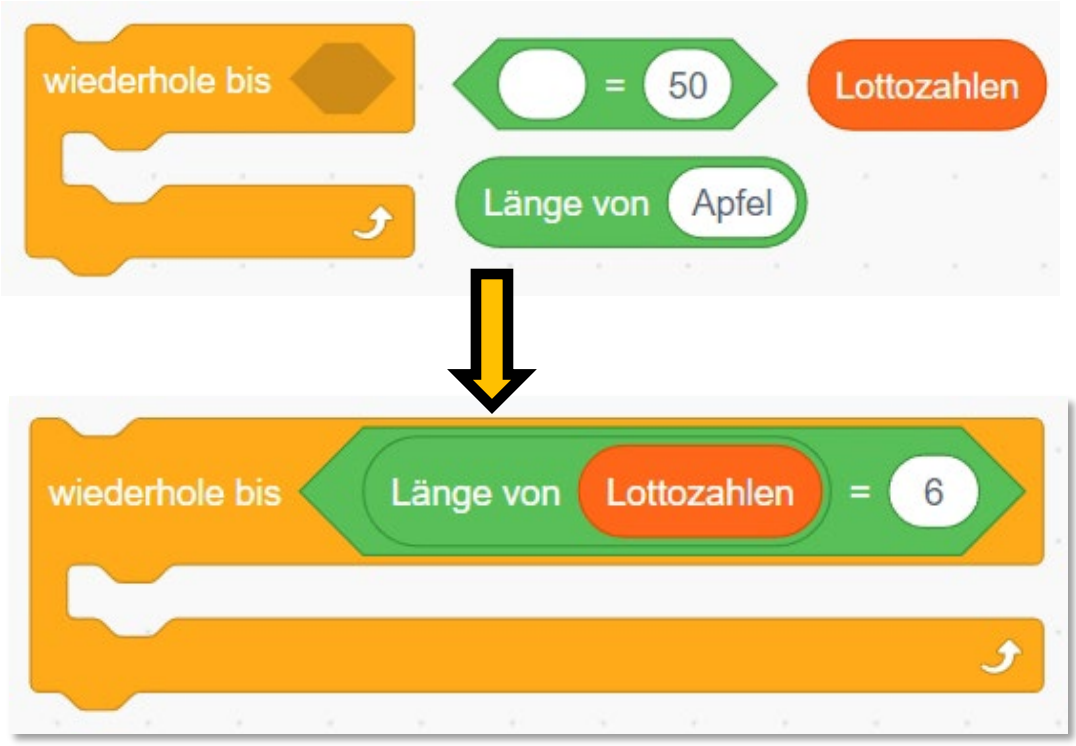
5.4 Variablen und Listen anlegen und bei Programmstart initialisieren

Im nächsten Schritt werden die Variablen angelegt. Es wird eine Variable „NeueZahl“ benötigt, die bei jedem Schleifendurchlauf mit einer Zufallszahl zwischen 1 und 49

befüllt wird. Darüber hinaus wird eine Liste „Lottozahlen“ benötigt, die im Gegensatz zu einer Variablen, mehrere Werte speichern kann. Sie speichert in diesem Projekt die 6 verschiedenen Lottozahlen, um sie später auszugeben.

5.5 Wiederholen-Schleife

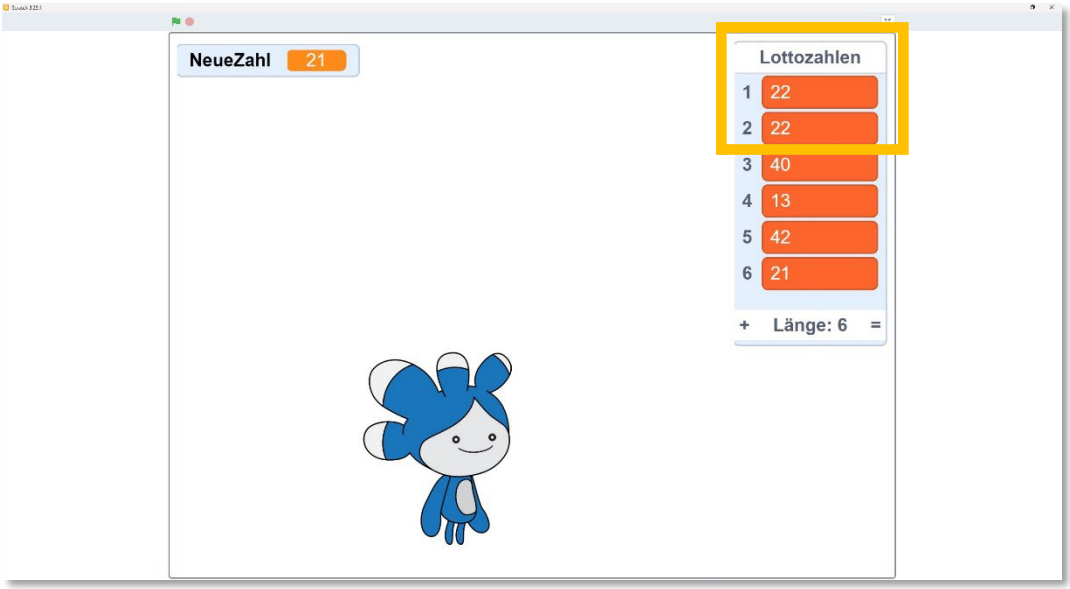
Im nächsten Schritt sorgt eine Schleife dafür, dass sechs verschiedene Zahlen zwischen 1 und 49 gezogen werden. Dabei ermittelt der Baustein „Länge von Apfel“ die Länge eines Strings, einer eingefügten Variable oder – wie im Projekt hier – die Anzahl der Elemente einer Liste.



In der Schleife wird bei jedem Durchlauf die Variable „NeueZahl“ auf eine Zufallszahl zwischen 1 und 49 gesetzt, die man dann der Liste „Lottozahlen“ hinzufügen kann.

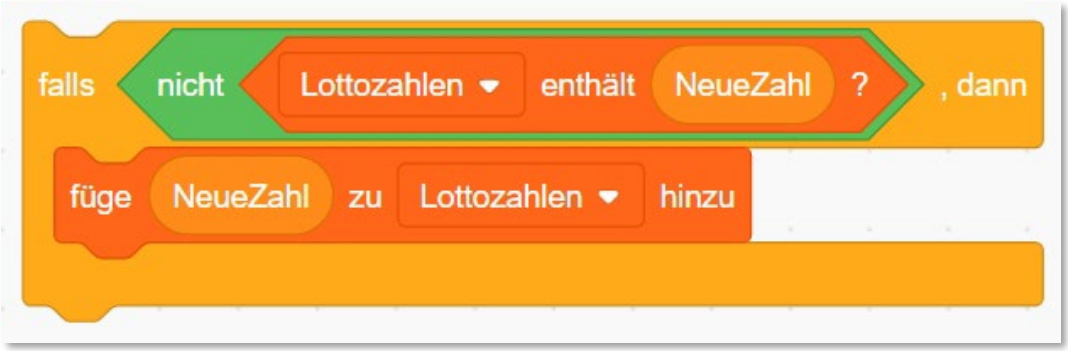
5.6 Testlauf

Startet man das Programm nun, wird die Liste mit 6 Zufallszahlen gefüllt. Es kann jedoch passieren, dass Zahlen doppelt gezogen werden, da keine Überprüfung stattfindet, ob eine Zahl bereits in der Liste „Lottozahlen“ vorhanden ist.



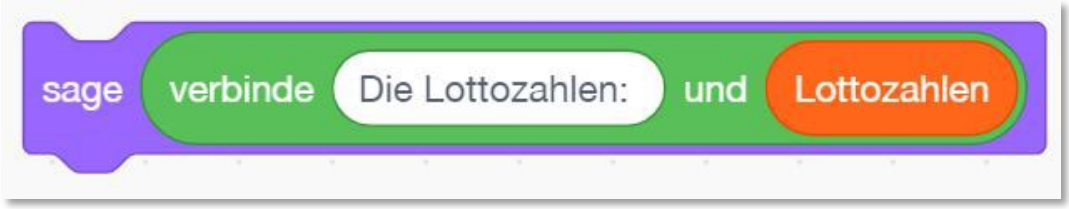
5.7 Fehlerbehebung

Der Fehler aus 5.6 lässt sich durch eine einfache Überprüfung, ob der Wert der Variablen „NeueZahl“ bereits in der Liste „Lottozahlen“ vorhanden ist, beheben. Hierfür ist der logische Operator „NICHT“ wichtig, der dafür sorgt, dass ein Wert nur dann der Liste hinzugefügt wird, wenn er darin noch nicht enthalten ist.

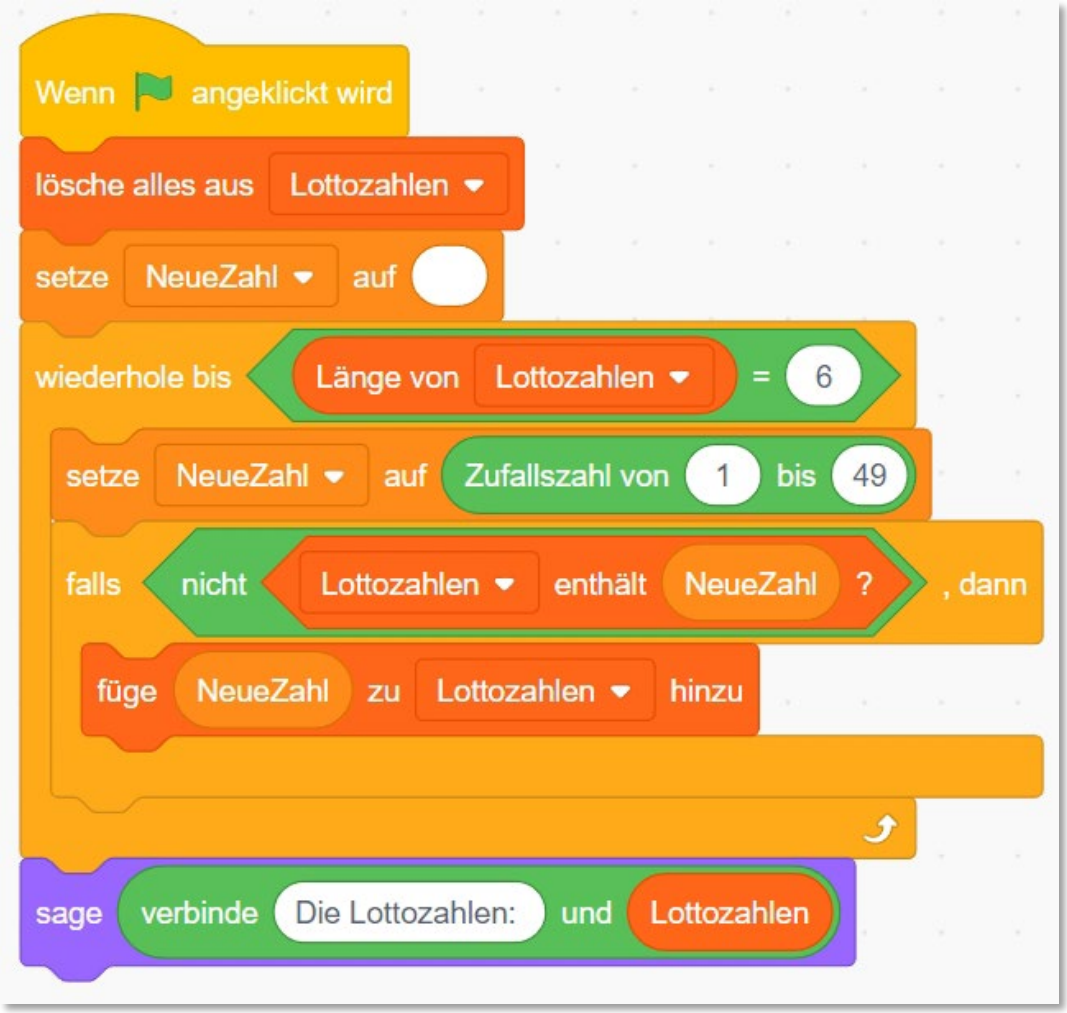


5.8 Ausgabe

Die Codierung der Ausgabe sollte auch bei diesem Programm „sprechend“ sein, sodass man besser nachvollziehen kann, welche Ausgabe man erhält. Auch eine Liste kann an den „Sage“-Baustein von Scratch übergeben werden, sodass die Ausgabe der 6 Zahlen direkt erfolgen kann.



5.9 Das fertige Programm



5.10 Differenzierung/Idee zum Weiterarbeiten

Die Ausgabe der 6 Zahlen erfolgt unsortiert. In Scratch gibt es leider keinen Befehl, der Elemente sortiert einfügen kann oder Elemente einer Liste sortiert. Man muss das Programm nicht unerheblich erweitern, um eine sortierte Ausgabe zu erhalten.

Hier werden weitere Variablen und Listen benötigt. Gleichzeitig aber erfüllt die Umsetzung der Sortierung dann die im Bildungsplan geforderte Kompetenz, nach der Schüler „(4) grundlegende Algorithmen auf einer indexbasierten Datenstruktur (z. B. Füllen mit Werten, Maximumsuche, Summenbildung) beschreiben und implementieren“[17] können.

[17] vgl. 2.2

6 Scratch-Projekt 4: Zahlenrätsel „Wie groß ist die Herde?“ (Klassen 7/8)

Als einer der bedeutendsten Universalgelehrten des islamischen Mittelalters gilt Ali al-Hasan Ibn al-Haitham, der in Europa auch unter dem Namen „Alhazen“[18] bekannt ist. Er war Mathematiker, Optiker und Astronom und verfasste über 200 Werke, in denen er sich u. a. mit mathematischen Fragen befasste[19]. In einer seiner Schriften findet man unter anderem folgendes mathematische Rätsel, in dem ein Schäfer nach der Größe seiner Herde gefragt wird und antwortet: „Es sind weniger als 500 Schafe. Wenn ich die Zahl meiner Schafe durch 2, 3, 4, 5 oder 6 teile, dann bleibt immer ein Schaf übrig. Teile ich meine Herde aber durch 7, dann geht die Rechnung genau auf.“

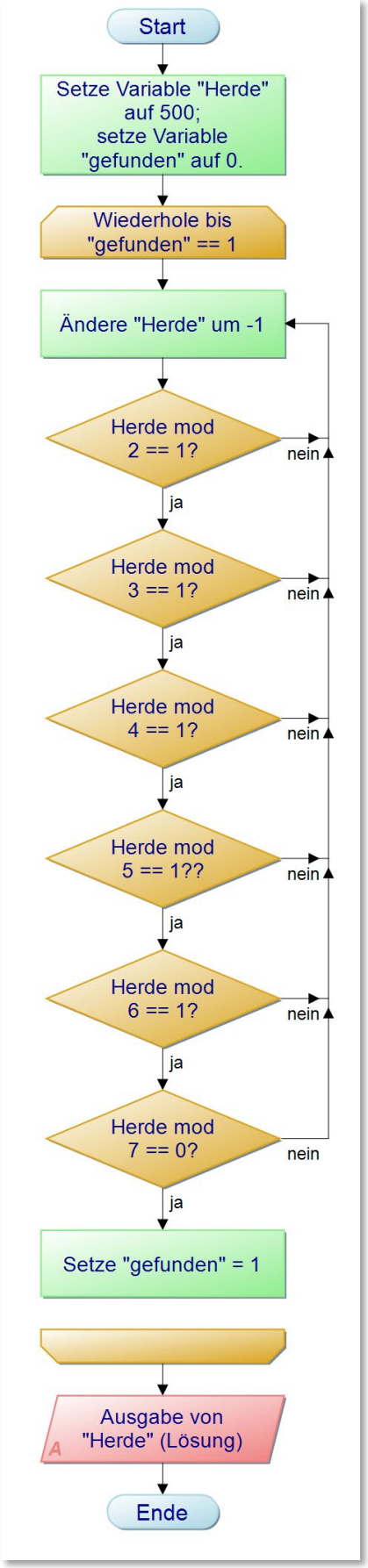
6.1 Angebahnte Kompetenzen

Das Projekt lässt sich zur Vertiefung von Bedingungen (vgl. 2.1) nutzen. Für die Lösung werden nur wenige Bausteine benötigt, die Hauptarbeit besteht im richtigen Einsatz von ineinander verschachtelten Verzweigungen. Alternativ lässt sich das Programm auch nutzen, um logische Operatoren einzuführen oder zu vertiefen (vgl. 2.2).

[18] vgl. https://de.wikipedia.org/wiki/Alhazen

[19] vgl. https://www.spektrum.de/wissen/abu-ali-al-hasan-ibn-al-haitham-965-1039/893657

6.2 Programmablaufplan



Initialisierung der Variablen:
Die Herde ist kleiner als 500, daher Startwert der Variablen „Herde“ = 500; die Variable „gefunden“ wird als Abbruchbedingung in einer Schleife genutzt.

Kontrollstruktur: Die Schleife läuft so lange, bis „gefunden“ == 1.

Ändere Wert der Variablen „Herde“ um 1.

Verzweigung: Bleibt ein Rest von 1, wenn „Herde“ durch 2 geteilt wird?

Verzweigung: Bleibt ein Rest von 1, wenn „Herde“ durch 2 geteilt wird?

Verzweigung: Bleibt ein Rest von 1, wenn „Herde“ durch 2 geteilt wird?

Verzweigung: Bleibt ein Rest von 1, wenn „Herde“ durch 2 geteilt wird?

Verzweigung: Bleibt ein Rest von 1, wenn „Herde“ durch 2 geteilt wird?

Verzweigung: Bleibt ein Rest von 1, wenn „Herde“ durch 2 geteilt wird?

Setze „gefunden“ auf 1.

Ausgabe der Lösung.

6.3 Scratch starten, Figur und Hintergrund einfügen

Wie bei den vorherigen Projekten können ggf. Figuren und Hintergründe eingefügt werden.

6.4 Variablen anlegen und bei Programmstart initialisieren

Für das Programm wird nur eine Variable „Herde“ benötigt. Diese speichert die aktuelle Größe der Herde, für die geprüft wird, ob eine Lösung vorliegt. Darüber hinaus wird eine Variable „gefunden“ benötigt, die als Abbruchbedingung für eine Schleife dient.

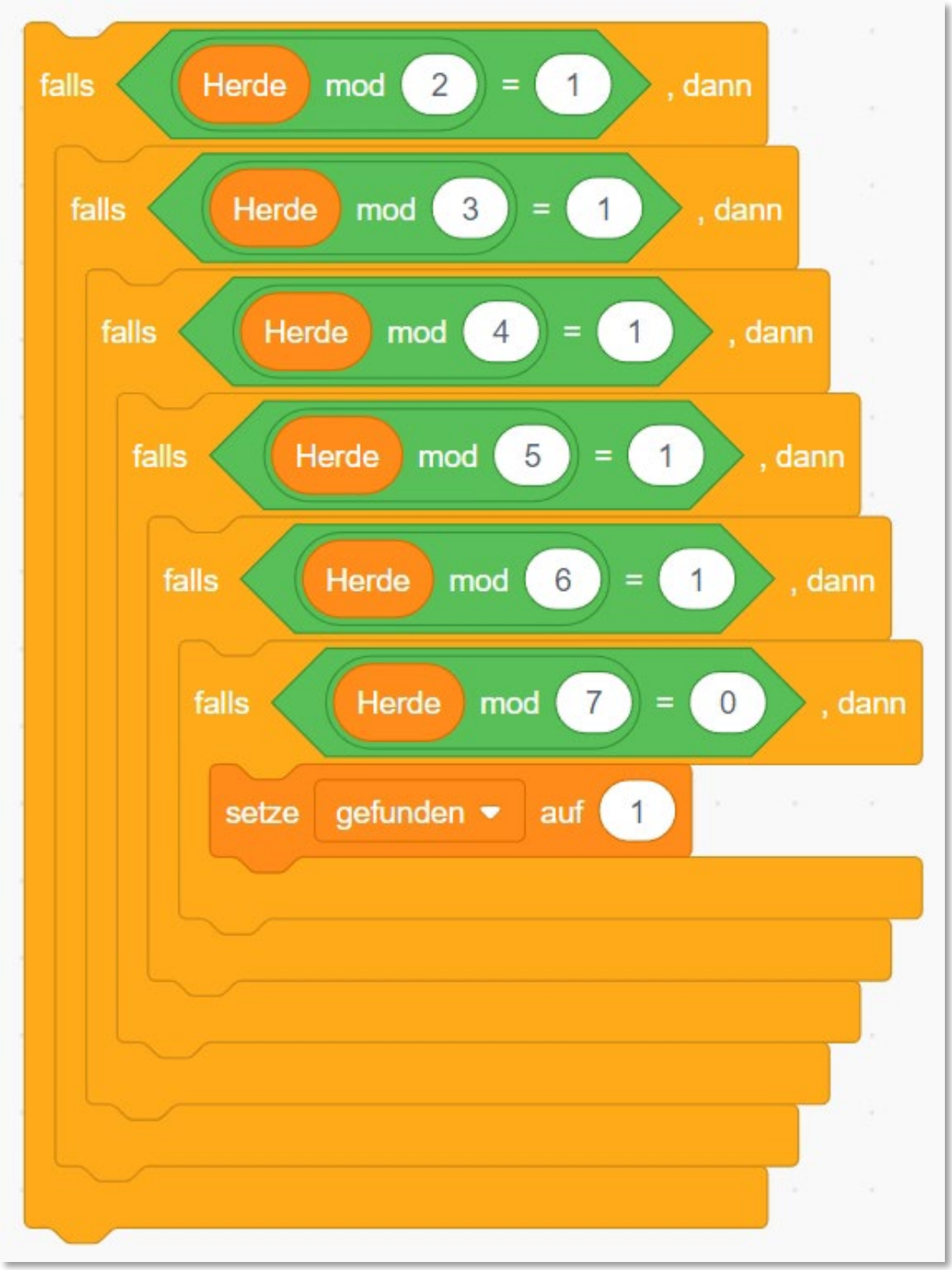
6.5 Wiederholen-Schleife

Der Kern des Programms ist die Wiederholen-Schleife mit den verschachtelten Verzweigungen. Die Schleife läuft so lange und ändert den Wert der Variablen „Herde“ um -1, solange die Variable „gefunden“ den Wert 0 hat.

Innerhalb der Schleifen werden Verzweigungen ineinander verschachtelt, in denen der Modulo-Befehl[20] (vgl. auch 4) genutzt wird, um den Rest der Divisionen durch 2, 3, 4,

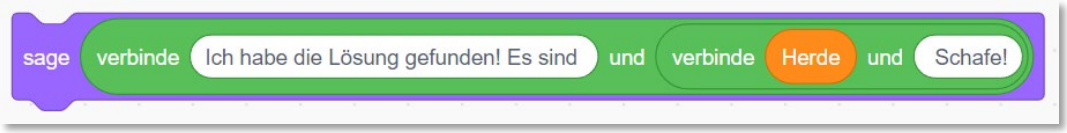
[20] vgl. https://de.wikipedia.org/wiki/Division_mit_Rest#Modulo

5, 6, und 7 zu ermitteln; beträgt dieser beim Teilen durch 1, 2, 3, 4, 5, und 6 jeweils 1, beim Teilen durch 7 aber 0, wird die Variable „gefunden“ auf 1 gesetzt.

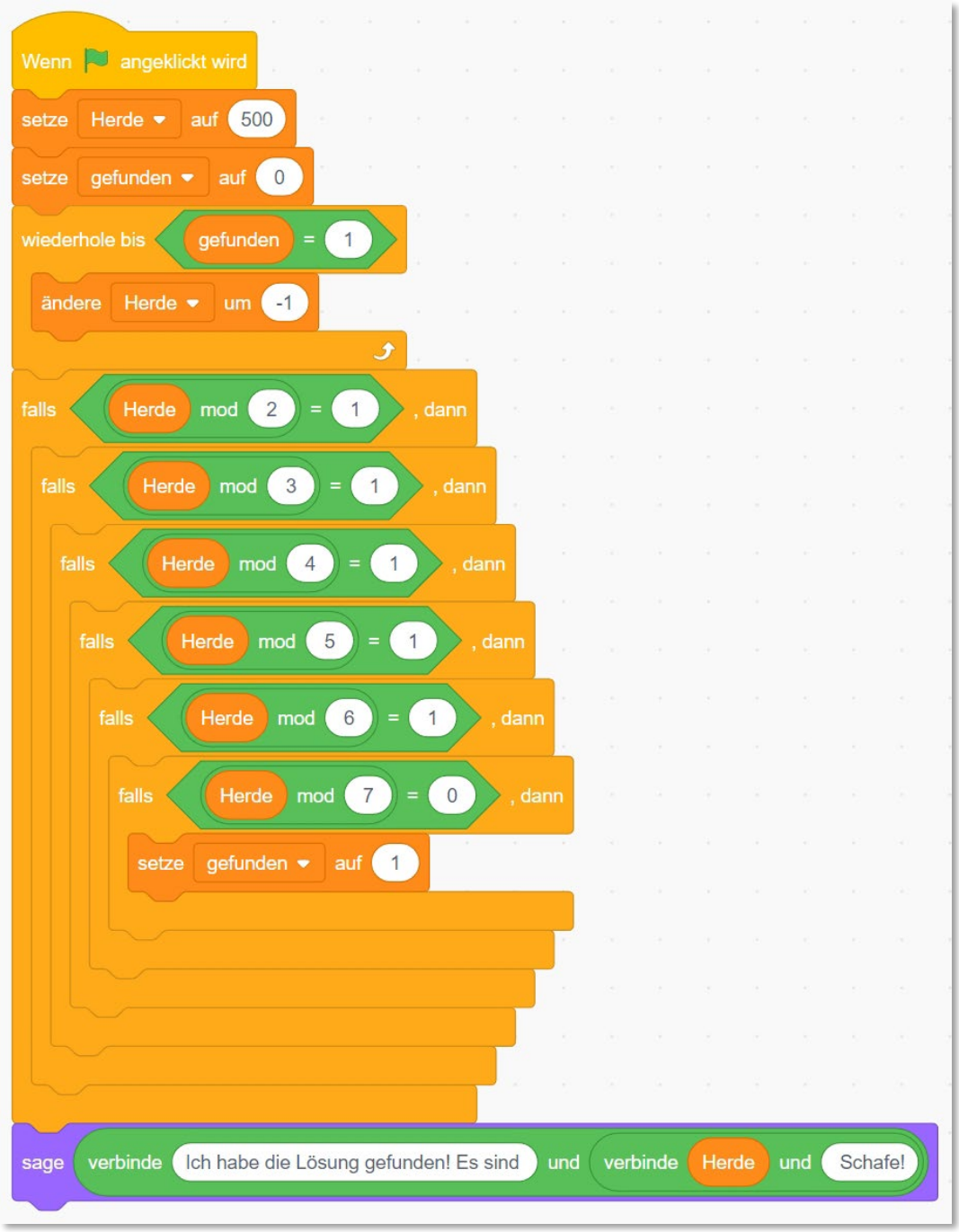


6.6 Ausgabe

Eine „sprechende“ Ausgabe gibt die Lösung aus.



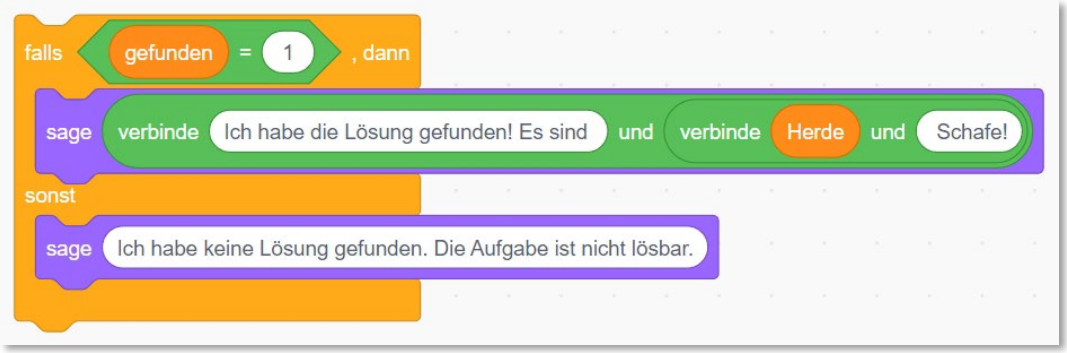
6.7 Das fertige Programm



6.8 Differenzierung/Idee zum Weiterarbeiten

Das Programm lässt sich z. B. so erweitern, dass auch die Möglichkeit abgefangen wird, dass es keine Lösung des Problems gibt. Hierfür sind Änderungen am Programmcode nötig, indem die Abbruchbedingung der Schleife mit Hilfe eines logischen Operators erweitert wird.

Auch die Ausgabe muss dann erweitert werden, sodass ggf. „Keine Lösung gefunden“ ausgegeben wird, wenn der Wert der Variablen „gefunden“ auch beim Abbruch der Schleife 0 beträgt.



Auch dieses Problem lässt durch Bedingungen lösen, die mit logischem „UND“ miteinander verknüpft werden (vgl. 7).

Hier sind 6 einzeln formulierbare Bedingungen nötig, die mit logischem „UND“ verknüpft werden. Sind alle Bedingungen erfüllt, ist eine Lösung gefunden und kann ausgegeben werden.

7 Scratch-Projekt 5: Zahlenrätsel „Fünfstellige Zahl gesucht“ (Klassen 7/8)

Bei diesem Projekt wird eine besondere fünfstellige Zahl gesucht, auf die folgende Bedingungen zutreffen:

a) die 1. Ziffer der gesuchten Zahl entspricht der Hälfte der 4. Ziffer;

b) die 2. Ziffer entspricht einem Viertel der letzten Ziffer;

c) die 3. Ziffer entspricht der 1. Ziffer + 4;

d) die 4. Ziffer entspricht dem Dreifachen der 2. Ziffer;

7.1 Angebahnte Kompetenzen

Das Projekt lässt sich zur Vertiefung von logischen Operatoren (vgl. 2.2) einsetzen. Gelöst besteht der Programmcode nur aus wenigen Bausteinen; die Hauptarbeit besteht im richtigen Einsatz von Bedingungen und deren Verknüpfung miteinander.

7.2 Programmablaufplan (PAP)

Initialisierung der Variablen:
Die kleinste fünfstellige Zahl ist 10000, auf diesen Wert wird die Variable zum Programmstart gesetzt.

Die Schleife läuft so lange, wie die formulierten Bedingungen nicht erfüllt sind.

Sind die Bedingungen nicht erfüllt, wird die Variable „FünfstelligeZahl“ um 1 erhöht.

Ausgabe der gefundenen Lösung.

7.3 Scratch starten, Figur und Hintergrund einfügen

Wie bei den vorherigen Projekten können ggf. Figuren und Hintergründe eingefügt werden.

7.4 Variablen und Listen anlegen und bei Programmstart initialisieren

Für das Programm wird nur eine Variable „FünfstelligeZahl“ benötigt. Diese speichert jeweils den aktuellen Wert einer fünfstelligen Zahl. Da die kleinste fünfstellige Zahl 10000 ist, wird die Variable bei Programmstart auf diesen Wert gesetzt.

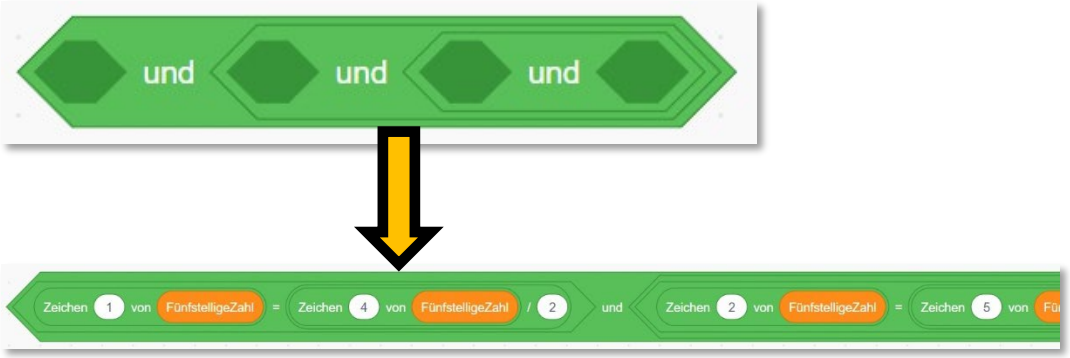
7.5 Wiederholen-Schleife

Die Schleife stellt den Kern des Projekts dar. Sie läuft solange, wie die vier Bedingungen (vgl. 6) nicht zutreffen. Die Grundstruktur des Programms sieht wie folgt aus:

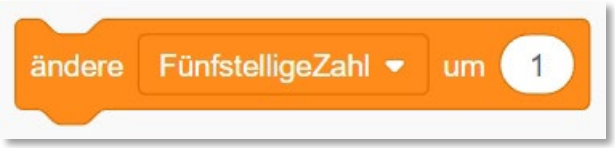
Entscheidend für das Finden einer Lösung ist die Formulierung der Bedingungen; sie werden mit dem logischen Operator „UND“ verknüpft, sodass alle Bedingungen

zutreffen müssen, damit die Schleife nicht mehr durchlaufen wird und eine Lösung ausgegeben werden kann. Die vier Bedingungen lassen sich mit den Operator-Bausteinen mit etwas Übung bzw. „Know-how“ recht leicht formulieren. Da sich aber auch leicht Fehler einschleichen können, ist ggf. eine Unterstützung der Schüler an dieser Stelle sinnvoll.

Die gerade formulierten Bedingungen werden nun in die Grundstruktur des Programms eingefügt, sodass sich durch die „UND“-Verknüpfung eine lange Kette von Bedingungen ergibt (und die sich hier nicht gänzlich darstellen lässt).



Bei jedem Durchlauf der Schleife muss die Variable „FünfstelligeZahl“ noch um 1 erhöht werden.



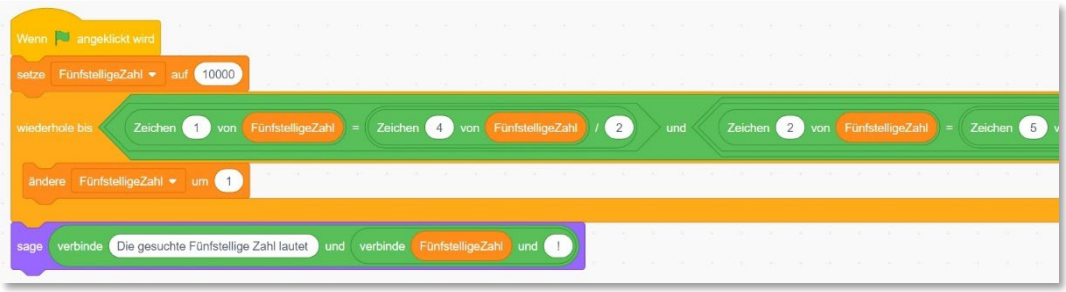
7.6 Testlauf

Startet man das Programm, sollte es recht schnell stoppen, weil nur nach einer fünfstelligen Zahl gesucht wird. Falls der Algorithmus nicht stoppt, liegt vermutlich ein Fehler bei den formulierten Bedingungen vor.

7.7 Ausgabe

Die Codierung der Ausgabe sollte auch bei diesem Programm „sprechend“ sein, sodass man gut nachvollziehen kann, zu welcher Problemstellung man ein Ergebnis erhält.

7.8 Das fertige Programm



7.9 Differenzierung/Idee zum Weiterarbeiten

Das Problem, die fünfstellige Zahl zu finden, lässt sich auch mit einer zusätzlichen Variable und verschachtelten „Wenn – Dann“-Bausteinen lösen. Für gute Schüler kann es eine Herausforderung sein, die Lösung auch ohne Verwendung des „UND“-Operators zu lösen. Und man kann mit der alternativen Lösung so deutlich machen, dass in der Programmierung viele Wege zum Ziel führen können.

Allerdings können bei komplexeren Algorithmen durchaus erhebliche Unterschiede in der Laufzeit auftreten, die hier aber vernachlässigt werden können.

```
Wenn [Flagge] angeklickt wird
setze gefunden auf 0
setze FünfstelligeZahl auf 9999
wiederhole bis <gefunden = 1>
    ändere FünfstelligeZahl um 1
    falls <Zeichen 1 von FünfstelligeZahl = (Zeichen 4 von FünfstelligeZahl / 2)>, dann
        falls <Zeichen 2 von FünfstelligeZahl = (Zeichen 5 von FünfstelligeZahl / 4)>, dann
            falls <Zeichen 3 von FünfstelligeZahl = (Zeichen 1 von FünfstelligeZahl + 4)>, dann
                falls <Zeichen 4 von FünfstelligeZahl = (Zeichen 2 von FünfstelligeZahl * 3)>, dann
                    setze gefunden auf 1
sage <verbinde (Die gesuchte Fünfstellige Zahl lautet) und (verbinde (FünfstelligeZahl) und (!))>
```

8 Scratch-Projekt 6: Zahlenrätsel „Besondere 6-stellige Zahlen“ (Klassen 7/8)

Drei Frauen treffen sich zufällig einige Jahre nach ihrem Mathematikstudium bei einem Klassentreffen wieder. Sie verstehen sich gut und damit sie künftig besser Kontakt halten und sich bald wieder treffen können, tauschen sie ihre Telefonnummern aus. Die Nummern sind sechsstellig; den mathematikbegeisterten Frauen fällt direkt auf, dass diese Nummern eine besondere Eigenschaft haben: Wenn man bei den sechsstelligen Nummern die letzten beiden Ziffern abschneidet und sie links vor die Zahl setzt, verdreifachen sich die Nummern, wenn sie als Zahl gelesen werden! Wie lauten die 3 Telefonnummern der Frauen?

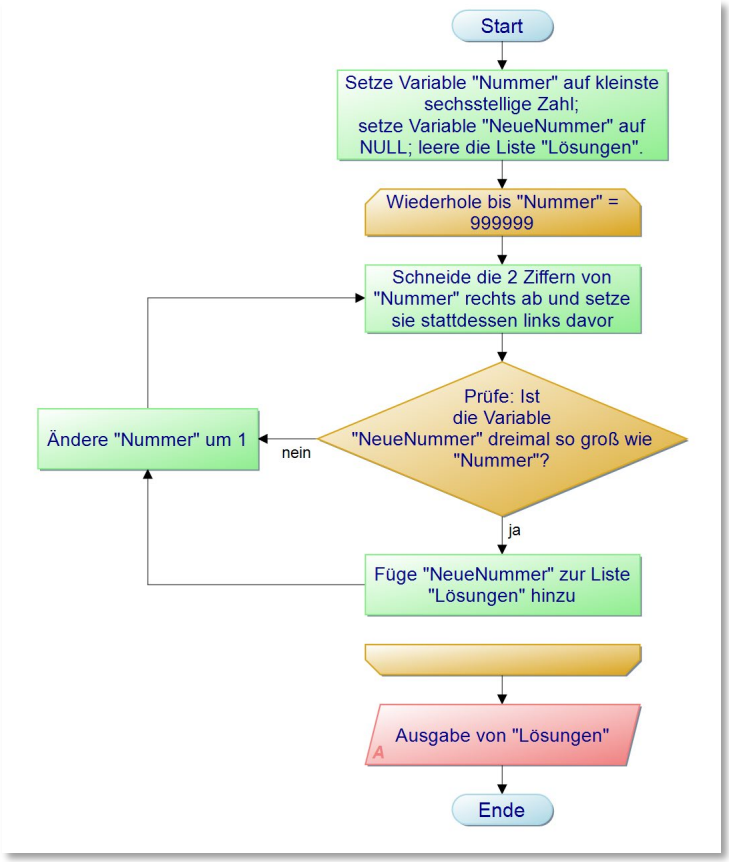
8.1 Angebahnte Kompetenzen

Schüler entwickeln vermutlich die Strategie, dass man alle Nummern der Reihe nach durchprobieren muss. Angefangen bei der kleinsten sechsstelligen Zahl (100000) bis hin zur größten (999999). Würde man alle diese Nummern durch Probieren auf mögliche Lösungen prüfen, wäre man sehr lange beschäftigt. Für den Computer stellt eine solche Aufgabe allerdings kein großes Problem dar, da er über genügend Rechenkraft verfügt. Insofern stellt diese Aufgabe eine ideale Lösung dar, um Schülern das Konzept der Brute-Force-Methode[21] deutlich zu machen, die auf dem Ausprobieren aller möglichen Lösungen beruht.

Um die Lösung für dieses Zahlenrätsel zu finden, müssen die Schüler verschiedenste Scratch-Elemente einsetzen: Variablen und Listen, Kontrollstrukturen, Verzweigungen und Bedingungen. Darüber hinaus müssen einzelne Stellen von Variablen ausgelesen und neu kombiniert werden; dies wurde in den bisherigen Aufgabenstellungen noch nicht benötigt, erweitert aber die Kompetenz von Schülern so, dass sie mit diesem Wissen künftig auch deutlich komplexere Zahlenrätsel lösen können.

[21] vgl. https://de.wikipedia.org/wiki/Brute-Force-Methode

8.2 Programmablaufplan



Initialisierung der Variablen „Nummer“ und „NeueNummer“.

Kontrollstruktur: Die Schleife läuft so lange, bis „Nummer“ == 999999.

Von „Nummer“ wird nach der Vorgabe eine „NeueNummer“ erstellt.

Falls „NeueNummer“ dreimal so groß ist wie „Nummer“, ist eine Lösung gefunden; in diesem Fall wird „NeueNummer“ in die Liste mit den Lösungen übernommen.

In beiden Fällen: „Nummer“ wird um 1 erhöht.

Ausgabe von „Lösungen“…

8.3 Scratch starten, Figur und Hintergrund einfügen

Wie bei den vorherigen Projekten können ggf. Figuren und Hintergründe eingefügt werden.

8.4 Variablen und Listen anlegen und bei Programmstart initialisieren

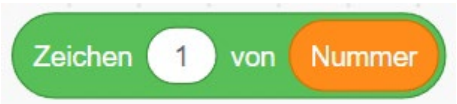
Für das Programm werden zwei Variablen „Nummer“ und „NeueNummer“ benötigt. „Nummer“ speichert die aktuelle Zahl, die in „NeueNummer“ nach dem benötigten Muster zusammengesetzt wird – die letzten beiden Stellen von „Nummer“ werden die ersten beiden von „NeueNummer“, danach folgen die Stellen 1 bis 4 von „Nummer“ als 3. bis 6. von „NeueNummer“ (vgl. 8).

Mögliche Lösungen werden in einer Liste „Lösungen“ gespeichert, deren Inhalt beim Programmstart gelöscht werden muss.

8.5 Wiederholen-Schleife

Den Kern des Programms bildet die Schleife, die so lange läuft, bis „Nummer“ den Wert 999999[22] erreicht hat.

Innerhalb der Schleife müssen nun die Stellen von „Nummer“ ausgelesen werden, um sie in „NeueNummer“ neu zusammenzusetzen. Das lässt sich mit Hilfe des in Scratch unter „Operatoren“ zu findenden Bausteins „Zeichen 1 von Apfel“ umsetzen. Fügt man die Variable „Nummer“ in den Baustein ein, lässt sich jede beliebe Stelle des aktuellen Wertes auslesen, hier im Beispiel das 1. Zeichen bzw. die 1. Stelle.

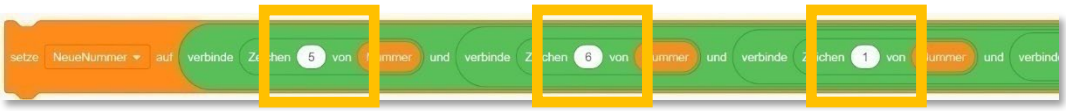


Mit Hilfe von sechs dieser Bausteine (Zeichen 1 von Nummer; Zeichen 2 von Nummer; Zeichen 3 von Nummer usw.) kann nun die aktuelle Nummer zu einer neuen

[22] Mathematisch bietet sich hier eine deutliche Optimierung für die Laufzeit an, vgl. 8.8

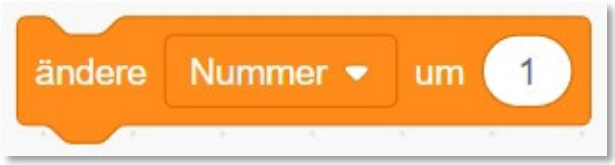
zusammengesetzt werden, indem verschachtelte „Verbinden“-Bausteine genutzt werden.

Im Beispiel wird der Wert der Variablen „NeueNummer“ auf das benötigte Muster gesetzt; „NeueNummer“ bekommt die letzten beiden Ziffern von „Nummer“ vorangestellt (vgl. 8). Durch einen Klick auf den Baustein kann man direkt überprüfen, ob das Muster von „NeueZahl“ dem für die Lösung der Aufgabe benötigten entspricht. (Die Darstellung an dieser Stelle ist nur verkürzt möglich.)



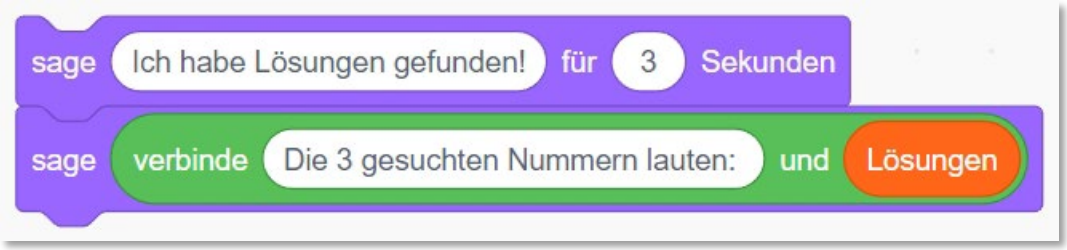
Nachdem die „NeueNummer“ erstellt wurde, muss geprüft werden, ob ihr Wert dreimal so groß ist wie der Wert von „Nummer“ – in diesem Fall hat man eine Lösung gefunden und man kann „Nummer“ in die Liste mit den Lösungen übertragen.

Nun muss nur noch die Variable „Nummer“ um 1 erhöht werden.

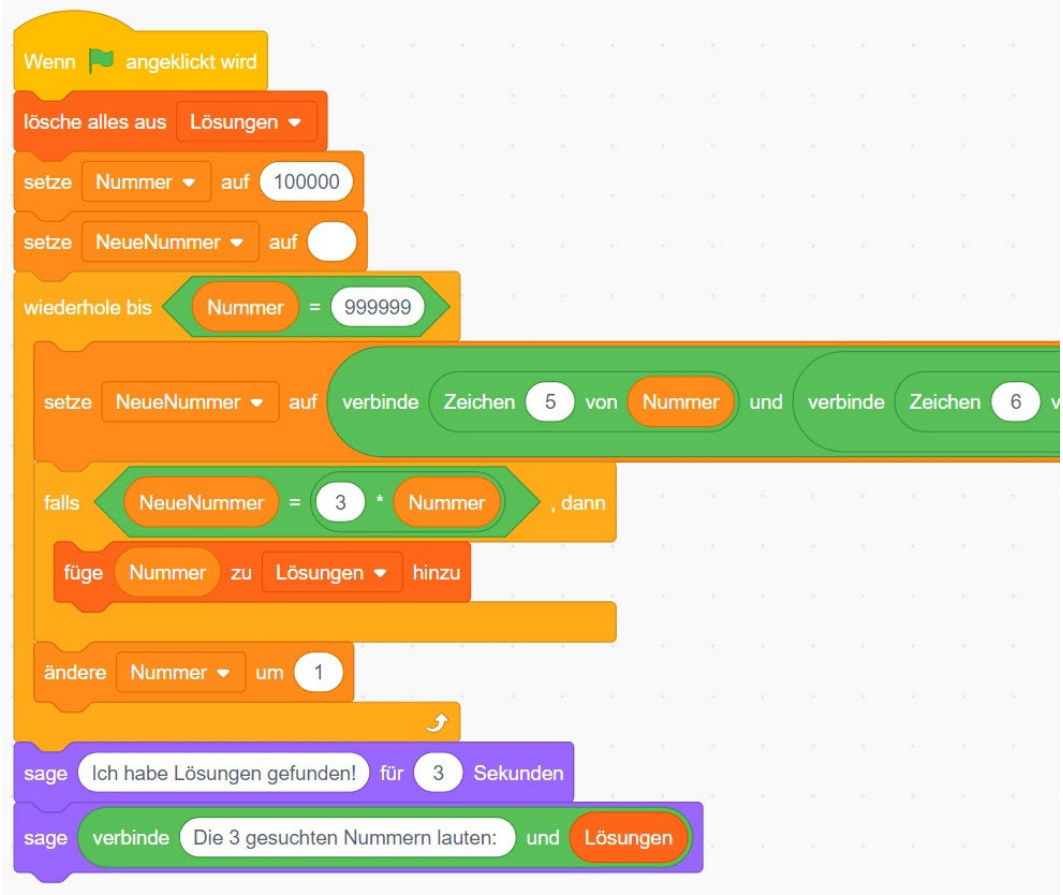


8.6 Ausgabe

Die Codierung der Ausgabe sollte auch bei diesem Programm „sprechend“ sein, sodass man gut nachvollziehen kann, zu welcher Problemstellung man ein Ergebnis erhält. In diesem Fall kann die Ausgabe wie folgt aufgebaut werden.



8.7 Das fertige Programm



8.8 Differenzierung/Idee zum Weiterarbeiten

Rein mathematisch lässt sich die Anzahl der Schleifendurchläufe mit der Brute-Force-Methode deutlich verkürzen, denn die Variable „NeueZahl“ kann maximal 1/3 der Größe von 999999 haben. Gute Schüler kommen ggf. selbst auf die Lösung, aber der Lehrer sollte das unter dem Gesichtspunkt der Laufzeitoptimierung thematisieren.

9 Literaturempfehlungen

Die folgenden Bücher eignen sich, um die Scratch-Grundlagen zu erlernen. Sie legen allerdings – bis auf das von Erik Bartman – den Schwerpunkt auf das Programmieren von Spielen.

 *Mit Scratch 3 programmieren lernen. Bombini Verlag. ISBN: 978-3-94649-611-3*	 *Der kleine Hacker. Programmieren für Einsteiger. Franzis Verlag. ISBN: 978-3-6456-0278-5*
 *Coole Spiele mit Scratch. dpunkt.verlag. ISBN: 978-3-86490-447-9*	 *Spiele Programmieren supereasy. Coole Games mit Scratch. DK-Verlag. ISBN: 978-3-8310-3095-8*

10 Weiterführende Internet-Links

Im Internet gibt es zahlreiche Seiten, die sich mit der Programmiersprache Scratch befassen. Auf vielen sind Schritt-für-Schritt-Anleitungen zu finden, die helfen, die Beispiele nachzubauen und so Scratch immer besser zu verstehen.

https://scratch.mit.edu/	*Die Anlaufstelle für Scratch-Fans. Hier gibt es Scratch zum Download sowie unzählige Projekte zum Anschauen, Ausprobieren, Nachbauen und Abändern.*
https://projekteuler.de/	*„Project Euler ist eine Serie von herausfordernden mathematischen bzw. Programmier-Problemen, die mehr als nur mathematische Einblicke zum Lösen erfordern.“*[23]
https://www.spektrum.de/raetsel/	*Mathematische Rätsel des Spektrum-Verlags. Einige der durchaus kniffligen Rätsel sind durch Programmierung lösbar.*
https://www.spiegel.de/thema/raetsel_der_woche/	*Beim „Rätsel der Woche“ vom Spiegel finden sich immer wieder mathematische Rätsel bzw. Aufgaben, die durch Programmierung lösbar sind.*
http://coderdojo-linz.github.io/infos/uebungsbeispiele.html	*Seiten des Coderdojo Linz, einem Programmierclub für Kinder und Jugendliche, die programmieren lernen wollen. Hier sind zahlreiche Beispiele zu Scratch zu finden. Auch Einblicke in andere Programmiersprache werden für Interessierte gegeben.*
https://studio.code.org/	*Seite, welche die Grundlagen der Informatik vermittelt. Selbstlernkurse für Schüler und Beispiele bringen u. a. die Programmierung näher.*

[23] https://projekteuler.de/about/info

11 Lizenz und Rechte

Ich möchte an dieser Stelle auf die Lizenzbedingungen von Scratch sowie die in diesem Skript verwendeten Bilder und Grafiken hinweisen.

> *„Scratch is a project of the Scratch Foundation, in collaboration with the Lifelong Kindergarten Group at the MIT Media Lab. It is available for free at https://scratch.mit.edu."*[24]

Scratch ist kostenlos und darf auch in Kursen oder Schulen – auch für kommerzielle Zwecke – eingesetzt werden. Grafiken und Screenshots in diesem Skript sowie den im Kurs eingesetzten Präsentationen dürfen ebenfalls kostenfrei – auch für kommerzielle Zwecke – verwendet werden.

Bitte beachten Sie ggf. die Hinweise zu den Lizenzen und Rechte von Scratch.

[24] https://scratch.mit.edu/info/faq#scratch3

12 Danksagung und Ausblick

12.1 Danksagung

Danken möchte ich der Hopp Foundation Weinheim für die tollen Workshops, an denen ich teilnehmen durfte, für die Unterstützung meiner Schule und natürlich auch für die Unterstützung meiner Idee für dieses Skript.

Ebenfalls danken möchte ich meinem Freund und Informatiklehrer Georg Busch für das Korrekturlesen, für das kritische Hinterfragen und letztlich für die Ermutigung, mit dieser Idee nach außen zu treten und dieses Skript zu veröffentlichen.

Darüber hinaus danke ich meinen Kindern und meiner Frau für das Verständnis, wenn ich „nur mal schnell“ noch etwas an diesem Skript zu Ende schreiben wollte und daraus wieder ein abendfüllendes Projekt wurde.

Nicht zuletzt danke ich allen Schülerinnen und Schülern, die unfreiwillig zu „Versuchskaninchen“ wurden, wenn ich die Umsetzung der Algorithmen in meinem Informatikunterricht getestet habe. Zu erkennen, dass auch der in diesem Skript formulierte Ansatz - die Abkehr von Spielen hin zum Lösen informatischer bzw. mathematischer Problemstellungen - im Unterricht funktioniert und Schülerinnen und Schüler dabei sogar „Spaß haben“, ist Motivation für jeden Lehrer!

12.2 Ausblick

Über dieses Skript hinaus sind dem Programmieren mit Kindern und Jugendlichen praktisch keine Grenzen gesetzt. Im Unterricht – das Fach Informatik wird in Baden-Württemberg nur 1-stündig unterrichtet – stellt jedoch der limitierende Faktor Zeit eine erhebliche Rolle. Deshalb bieten sich Aufgaben aus der Programmierung, gerade auch Zahlenrätsel, als Projekte und/oder GFS in der Schule an.

Ideen zum Vertiefen, weitere Zahlenrätsel und Ideen sind in diesem Skript im Kapitel 10 zu finden.