

Überblick über die Materialien zur algorithmischen Problemlösung mit Scratch sowie didaktische Überlegungen

Im Folgenden wird ein kurzer Überblick über die Materialien zur algorithmischen Problemlösung mit Scratch gegeben. Die Materialien enthalten insbesondere auch Vorschläge zu kontextbezogenen und offenen Aufgaben sowie ein Beispiel für ein arbeitsteiliges Projekt. Zu mit * gekennzeichneten Materialien gibt es für Lehrkräfte Lösungsvorschläge.

Didaktische Anmerkungen zum Einsatz der Materialien

Die Materialien für Schüler:innen sind unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#) lizenziert. Daher können und sollen sie an den eigenen Unterricht und die eigene Lerngruppe angepasst werden. In der tabellarischen Übersicht über die einzelnen Materialien werden hierzu zum Teil Vorschläge für mögliche Anpassungen skizziert. Für den Einsatz im Unterricht sind vielfältige weitere Differenzierungsmöglichkeiten denkbar, beispielsweise:

- Stundeneinstiege oder Angebot für Besprechung mit Interessierten: Interpretation von fertigen Programmen: Die Skripte eines Scratch-Programmes werden gezeigt, ohne das Programm zunächst zu starten. Die Lernenden beschreiben und interpretieren die Skripte – anschließend können diese Vermutungen durch Starten des Programms überprüft werden. Ergänzend bzw. als Sicherung des Besprochenen können Vermutungen über nötige Änderungen am Programm aufgestellt werden, um eine bestimmte andere Funktionalität zu erhalten.
- Als Hilfestellung/zu Beginn der Einheit werden Teillösungen bereits zur Verfügung gestellt. Die Lernenden können darin Eingaben variieren und weitere Blöcke ergänzen, um die gewünschte Funktionalität zu erreichen.
- Hilfreiche Blöcke zur Verfügung stellen – hier sind verschiedene Abstufungen denkbar:
 - alle Blöcke, korrekte Reihenfolge muss von Lernenden gefunden werden
 - nur zentrale Blöcke
 - „neue“ Blöcke, die bisher kaum verwendet wurden
- Einzelne Materialien können als Zusatzmaterialien „für Schnelle“ eingesetzt werden.

Einstieg in das algorithmische Problemlösen mit Scratch 3.0

Die folgenden Materialien bieten einen ersten Einstieg in das algorithmische Problemlösen mit Scratch 3.0. Im Vordergrund steht hierbei die Entwicklung eigener Algorithmen auf experimentelle Weise. Elementare Kontrollstrukturen wie Anweisung, Sequenz, Schleife und Verzweigung werden anhand von Beispielen vorgestellt und zur Problemlösung intuitiv verwendet. Ergänzt werden die Materialien um kontextbezogene und offene Aufgaben.

Im Dokument L2_Beiispiel_Unterrichtsverlauf_Jg9 wird beispielhaft ein möglicher Unterrichtsverlauf im Unterricht im Pflichtfach Informatik unter Verwendung der folgenden Materialien skizziert.

Name	Kategorie	Beschreibung	Ideen für Differenzierungsmaßnahmen/Anpassungen an den eigenen Unterricht
01_Ueberblick_Scratch3	Beschreibung (für Lehrkräfte oder als Ergänzungsmaterial)	Kurzer Überblick über die Programmieroberfläche, eher relevant für Lehrkräfte oder als Material zum Nachschlagen	kein direkter Einsatz im Unterricht, höchstens Ergänzungsmaterial zum Nachlesen. Ideen für mögliche Einstiege in Scratch vgl. Tabelle oben
02_Programmieren_in_Scratch	Leittext	<p>Leittext mit kleinen Aufgaben zum Einstieg in die Programmierung unter Nutzung von algorithmischen Grundstrukturen</p> <p>Mithilfe des Leittextes können Lernende neue Sachverhalte in individuellem Lerntempo erarbeiten. Dabei geben die Programme in Scratch eine direkte Rückmeldung, so dass so für Lernende schnell ersichtlich wird, ob ihre Programme die jeweiligen Aufgabenstellungen erfüllen oder nicht, und nicht unbedingt jede</p>	<ul style="list-style-type: none"> • Aufteilung in einzelne Arbeitsblätter/Lernkarten, z.B.: <ul style="list-style-type: none"> ○ Schleifen ○ Ereignisse ○ Verzweigungen • Reduktion des Textanteils bzw. Anpassung der Sprache • Weglassen einzelner Abschnitte bzw. Bereitstellung dieser als Zusatzaufgaben „für Schnelle“ (z.B. „Steuern von Objekten mit der Maus“, „Verstecken von Objekten“ oder

		<p>Aufgabe vollständig im Plenum besprochen werden muss.</p> <p>In Gesprächsphasen im Unterricht bietet sich beispielsweise die Thematisierung von beobachteten Fehlvorstellungen, Schwierigkeiten etc. an.</p>	„Zufallszahlen + Objekte berühren“ mit den zugehörigen Aufgaben)
Interaktionsgeschichte	Aufgabe	Einfache Aufgabe zum Steuern eines Programms über Nachrichten; spricht eher jüngere SuS an, da kreativ Geschichten erzählt werden können	offene Aufgabe mit individuellem Gestaltungsspielraum, daher selbstdifferenzierend
Bienenschwarm	Aufgabe	Übungsaufgabe/kleines Spiel	<p>ggf. Bereitstellung hilfreicher Blöcke oder Diskussion über möglichen ersten Ansatz;</p> <p>vgl. allgemeine Bemerkungen zu Differenzierungsmöglichkeiten;</p>
Scratch-Translator	Aufgabe	Verwenden der Erweiterungen „Übersetzen“ und „Text to Speech“ für ein kleines Übersetzungsprogramm	<p>z.B. Aufteilung in Pflicht- und Wahlaufgabe:</p> <p>1+2: für alle (diese Aufgabenteile enthalten bereits Hinweise zu hilfreichen Blöcken)</p> <p>Wahl zwischen 3 oder 4: diese Aufgaben sollen zum Ausprobieren/Reflektieren anregen, dabei kann 3 durch Ausprobieren gelöst werden, in 4 ist ein stärkerer Transfer erforderlich</p>
Farberkenner*	Aufgabe	Kontextbezogene Aufgabe zur automatisierten Erkennung der Hintergrundfarbe; eher leichtere Aufgabe mit Angebot zur Reflexion für leistungstärkere Lernende	Aufgabe enthält bereits Hilfsblöcke sowie Zusatzaufgaben zur Reflexion „für Schnelle“;

Vorlese-Trainer*	Aufgabe	Kontextbezogene und offene Aufgabe zur Interpretation der aktuellen Lautstärke	<p>Aufgabe enthält bereits Anregungen zu Erweiterungen des Programms bzw. zur Reflexion „für Schnelle“.</p> <p>Als Hilfsangebot könnten zusätzlich weitere hilfreiche Blöcke wie etwa eine Verzweigung oder eine Vorlage mit zwei verschiedenen Kostümen angeboten werden.</p>
PinNachbauen*	Aufgabe	Offene kleine Aufgaben zur Rekonstruktion von Zugangskontrolle und Co beim Smartphone, diese kann später um eine Variable zum Ändern der Pin erweitert werden	<p>Aufteilen in zwei Aufgaben oder Kennzeichnung des Schwierigkeitsgrades der Vorschläge: das Ausschalten des Bildschirms ist deutlich einfacher zu realisieren als das Freischalten des Bildschirms. Die Änderung einer Pin ist anspruchsvoll und eher eine Zusatzaufgabe für Fortgeschrittene.</p> <p>Durch die Offenheit der Abfrage nach einer Pin ist die Aufgabe zum Teil selbstdifferenzierend: hier sind Lösungen in verschiedenen algorithmischen Niveaus denkbar (z.B. eine einfache Verzweigung / Verzweigung in Schleife und Berücksichtigung, wenn falsch eingegeben / bei dreimaliger Falscheingabe Sperrung,...)</p>
Interaktiver Adventskalender	Arbeits- teiliges Projekt	Arbeitsteilig werden die verschiedenen Türen eines Adventskalenders zunächst in Einzelarbeit oder Teams (in Form einer offenen Aufgabe) implementiert und anschließend zu einem Gesamtprodukt zusammengeführt. Je nach Vorerfahrungen der Lerngruppe kann dieses	<p>falls keine Realisierung in einer ganzen Lerngruppe gewünscht ist, kann ein ähnliches arbeitsteiliges Projekt auch in kleineren Gruppen durchgeführt werden. Ideen wären zum Beispiel:</p> <ul style="list-style-type: none"> - arbeitsteilige Vorstellung verschiedener Berufe (je Beruf eine Figur auf der Bühne)

		<p>Projekt bereits in einer Einstiegsphase zum algorithmischen Problemlösen (dann mit stärkerer Vorstrukturierung) oder zu einem späteren Zeitpunkt (dann mit stärkerer Selbstorganisation innerhalb der Lerngruppe) durchgeführt werden.</p>	<ul style="list-style-type: none"> - arbeitsteilige Vorstellung verschiedener Sportarten - arbeitsteilige Vorstellung verschiedener Musikinstrumente/chemischer Elemente/historischer Persönlichkeiten... <p>Dieses oder ein ähnliches arbeitsteiliges Projekt bietet sich auch als Wieder-Einstieg in das algorithmische Problemlösen an. Die Implementierung eines Objektes wie etwa eines Türchens eines Adventskalenders ist eine offene Aufgabe und stark selbstdifferenzierend, so dass Lernende damit gut wieder in das algorithmische Problemlösen hineinfinden können.</p>
Simulation Rasenmäher*	Aufgabe	<p>Offene und kontextbezogene Aufgabe zur Simulation des Verhaltens eines Rasenmähroboters; Variablen können zum Zählen der Durchgänge verwendet werden, sind aber nicht notwendig;</p> <p>Als Einstieg oder alternativ als noch offenere Variante ohne Arbeitsblatt kann das Video „Video_Robby_maeht“ zum Rasenmähroboter gezeigt werden</p>	<p>Als Einstieg oder alternativ als noch offenere Variante ohne Arbeitsblatt kann das Video „Video_Robby_maeht“ zum Rasenmähroboter gezeigt werden.</p> <p>Das Arbeitsblatt selbst enthält bereits hilfreiche Blöcke. Ergänzend könnte im Einstieg beispielsweise</p> <ul style="list-style-type: none"> • gezeigt werden, wie man im Block „wird Farbe .. berührt“ die Farbe über die Pipette festlegen kann • das grundsätzliche Fahrverhalten eines Roboters besprochen werden (Grundprinzip: Schleife, gerade aus fahren, regelmäßig

			<p>überprüfen, ob auf „Rand“, etwas zurück fahren und dann drehen)</p> <p>Für Schnelle bieten sich Zusatzaufgaben an wie</p> <ul style="list-style-type: none"> • Änderung der Rasenfläche, d.h. des Bühnenbildes (z.B. auf nicht stärker verwinkelte Rasenflächen, Bäume oder andere Objekte in der Rasenfläche) • Einfügen einer Ladestation, zu der der Rasenmähroboter regelmäßig zurückkehren muss • Zählen der Durchläufe nach Start, um sichtbar zu machen, wie viele benötigt werden, um die gesamte Fläche zu mähen • ...
Reflexion_Algorithmen	Aufgaben	<p>Am Beispiel eines gegebenen Algorithmus können die Lernenden über Eigenschaften von Algorithmen reflektieren.</p> <p>Das Material kann als Überleitung zum EVA-Prinzip genutzt werden, als Abschluss einer Einheit zum Algorithmischen Problemlösen oder auch in einer Vertiefung zum Algorithmischen Problemlösen</p>	<p>beispielsweise</p> <ul style="list-style-type: none"> • Ersetzen des abgebildeten Scratch-Programms durch ein oder zwei weniger umfangreiche Beispiele, z.B. ein Skript mit nur einer Verzweigung und eines mit einer Schleife • Ergänzung eines weiteren Algorithmus aus Makecode (wenn im Unterricht Calliopes mit Makecode programmiert wurden) • Aufgabenteil 3: ein Beispiel bereits nennen, z.B. „Kuchen backen“

Vertiefung des algorithmischen Problemlösens mit Scratch 3.0 – Verwendung von Variablen

Die folgenden Materialien bieten eine erste Vertiefung des algorithmischen Problemlösens mit Scratch 3.0. Algorithmen können zunehmend zielgerichtet unter Verwendung der elementaren Kontrollstrukturen entwickelt werden. Zur Implementierung werden nun auch Variablen und logische Verknüpfungen genutzt.

Name	Kategorie	Beschreibung	Ideen für Differenzierungsmaßnahmen/Anpassungen an den eigenen Unterricht
Einstiegsbeispiel_Countdown*	Aufgabe	Einstiegsbeispiel für die Verwendung von Variablen (entspricht erstem Blatt des Leittextes)	z.B. Sortierung der Vorschläge zu Aufgabe 3 nach Schwierigkeitsgrad und Angabe des Schwierigkeitsgrades
03_Variablen	Leittext	Leittext mit kleinen Aufgaben zu Variablen, Operatoren, Datentypen	<ul style="list-style-type: none"> • Aufteilung in einzelne Arbeitsblätter/Lernkarten, z.B.: <ul style="list-style-type: none"> ○ S. 1 Variablen in Scratch ○ S. 2 Hintergrundwissen (ggf. nur als ergänzendes Material zur Verfügung stellen) ○ Aufgaben S. 3, ggf. nur Aufgabe 1 und Aufgabe 2 dann „für Schnelle“ ○ Abschnitt „Operatoren“ ab S. 3 ○ ... • Reduktion des Textanteils bzw. Anpassung der Sprache • Weglassen einzelner Abschnitte bzw. Bereitstellung dieser als Zusatzaufgaben „für Schnelle“ (z.B. „Datentypen in Scratch“, „Operatoren“, ...)

Quizmoderator	Aufgabe	Verwendung von Verzweigungen und einer Variablen zum Zählen der korrekten Antworten	Aufgabe 5: „Für Schnelle“
Labyrinth	Aufgabe	Offenere Aufgabe zur Erstellung eines kleinen Spiels unter Verwendung von Variablen; spricht eher jüngere Lernende an	
Kassenautomat*	Aufgabe	Offene Aufgabe zur Rekonstruktion eines Kassenautomaten	<ul style="list-style-type: none"> • Reduktion des abgebildeten Beispiels auf eine Preiskategorie • siehe allgemeine Anmerkungen oben
Captcha-Creator*	Aufgabe	Erklärung von Captchas, Unterschied Mensch – Maschine; Analyse und Erweiterung eines gegebenen Programms, einfaches Arbeiten mit einer Variablen	<p>Aufteilen in zwei Arbeitsblätter, die jeweils nur von einem Teil der Lerngruppe bearbeitet werden:</p> <p>1. Teil: bis einschließlich Aufgabenteil c) → Reflexion des Einsatzes von Captchas</p> <p>2. Teil: Ab „Das Beispiel-Captcha... → Implementierung eines Captcha-Creators</p> <p>Aufgabe „für Schnelle“: Die Ergänzung um die Überprüfung einer Nutzereingabe (letzter Absatz)</p>
PinNachbauen	Aufgabe	Offene kleine Aufgaben zur Rekonstruktion von Zugangskontrolle und Co beim Smartphone, diese kann jetzt um eine Variable zum Ändern der Pin erweitert werden	<p>Wenn die Aufgaben noch nicht im Unterricht behandelt wurde, kann die Aufgabe arbeitsteilig implementiert werden:</p> <ul style="list-style-type: none"> • etwas leichter: die Implementierung des Freischaltens des Bildschirms • etwas schwieriger: Ändern der Pin, ggf. unter vorheriger Abfrage der alten Pin
Simulation Rasenmäher*	Aufgabe	Offene Aufgabe zur Simulation des Verhaltens eines Rasenmähroboters;	s.o.

		<p>Variablen können zum Zählen der Durchgänge verwendet werden, sind aber nicht notwendig;</p> <p>Als Einstieg oder alternativ als noch offenere Variante ohne Arbeitsblatt kann das Video „Video_Robby_maeht“ zum Rasenmäroboter gezeigt werden</p>	
Reflexion_Algorithmen _Variablen	Aufgaben	<p>Am Beispiel eines gegebenen Algorithmus können die Lernenden über Eigenschaften von Algorithmen reflektieren. Dabei ist das hier abgebildete Scratch-Programm deutlich komplexer als das im Material „Reflexion_Algorithmen“ –</p> <p>Das Material kann als Überleitung zum EVA-Prinzip genutzt werden, als Abschluss einer Einheit zum Algorithmischen Problemlösen oder auch in einer Vertiefung zum Algorithmischen Problemlösen</p>	<p>beispielsweise</p> <ul style="list-style-type: none"> • Bereitstellung beider Varianten „Reflexion_Algorithmen“ und „Reflexion_Algorithmen_Variablen“ und Entscheidung der Lernenden, welches Programm sie untersuchen • Ersetzen des abgebildeten Scratch-Programms durch ein oder zwei weniger umfangreiche Beispiele, z.B. ein Skript mit nur einer Verzweigung und eines mit einer Schleife • Ersetzen des Scratch-Programms durch eines unter Verwendung von Variablen • Ergänzung eines weiteren Algorithmus aus Makecode (wenn im Unterricht Calliopes mit Makecode programmiert wurden) <p>Aufgabenteil 3: ein Beispiel bereits nennen, z.B. „Kuchen backen“</p>

Vertiefung des algorithmischen Problemlösens mit Scratch 3.0 und Snap! – Verwendung von Listen

Die folgenden Materialien bieten eine weitergehende Vertiefung des algorithmischen Problemlösens. Sie eignen sich beispielsweise für Lerngruppen, die bereits sicher Variablen und elementare Kontrollstrukturen zur algorithmischen Problemlösung verwenden. Die hier aufgelisteten Materialien eignen sich insbesondere auch für Lerngruppen der Einführungsphase der Sek II.

Zur Entwicklung und Implementierung von Algorithmen werden neben den elementaren Kontrollstrukturen, Variablen und logischen Verknüpfungen nun auch Listen verwendet, die die Verarbeitung mehrerer Variablen ermöglichen. Dabei kommen auch einfache algorithmische Standardstrategien wie das systematische Durchlaufen einer Liste zum Einsatz.

Didaktische Anmerkung: Allgemein sollte bei der Unterrichtsgestaltung auf eine Vermittlung einer breiten Auswahl verschiedener informatischer Kompetenzen geachtet werden. Hat eine Lerngruppe bereits viel Zeit in das algorithmische Problemlösen investiert und noch kaum andere informatische Kompetenzen erworben, so sind im Zweifel andere Inhalte den folgenden vorzuziehen sind.

Name	Kategorie	Beschreibung
04_Listen*	Leittext	Leittext zur Einführung von Listen mit kleinen Aufgaben zu typischen Algorithmen auf Listen
Passwortwechsel*	Aufgabe	Eher einfachere kontextbezogene Aufgabe
Sprachen vergleichen*	Aufgabe	Aufgabe zur Erzeugung und Auswertung mehrerer Listen; Zur Bearbeitung dieser Aufgabe ist es hilfreich (aber nicht unbedingt notwendig), vorher die Aufgabe Scratch-Translator gelöst zu haben. In der Zusatzaufgabe kommen eigene Blöcke und geschachtelte Verzweigungen zum Einsatz.
Hühnersprachengenerator*	Aufgabe	Als Einstieg bietet sich die Audio-Datei Huehnersprache.mp3 an. Dort spricht ein Kind eine kurze Sequenz in „Hühnersprache“. Diese Aufgabe kann mit oder ohne Listen bearbeitet werden (vgl. Lösung Huehnersprache.sb3). Ohne die Verwendung von Listen wird die

		Bedingung in der Verzweigung jedoch sehr lang. Daher kann die Aufgabe auch als alternativer Einstieg zur Motivation von Listen genutzt werden.
Stimmanalyse*	Aufgabe	Kontextbezogene Aufgabe zur Stimmanalyse mit Scratch: Dabei liegen die Frequenzen in Textdateien zur Verfügung und werden in Scratch importiert. Sollen die Frequenzen selbst aufgezeichnet werden, bietet sich stattdessen die Verwendung von Snap! an, wo diese direkt über den Block Mikrofon der Kategorie Fühlen abgerufen und in eine Liste gespeichert werden können.
Playlist_erstellen*	Aufgabe	Kontextbezogene Aufgabe zur Rekonstruktion von Teilen einer Musik-App. Scratch 3.0 stellt bereits eine Fülle von Klängen zur Verfügung. In einer Vorlage sind als „Songs“ geeignete Klänge bereits in einem Projekt eingefügt. Lernende können so einzelne Funktionalitäten wie etwa das Hinzufügen oder Löschen von Songs zu einer eigenen Playlist implementieren. Weitere Funktionalitäten wie das Abspielen aller Lieder, Zufallsauswahl, Wechsel zum nächsten Lied und vieles mehr sind ebenso denkbar und ermöglichen so differenzierte algorithmische Herangehensweisen.
05_Listen_Snap!	Lehrerhinweise mit verschiedenen Aufgabenvorschlägen*	In diesem Dokument wird auf ausgewählte Aspekte zu Listen in Snap! eingegangen, die für interessierte Schüler:innen auch im Unterricht in der Sekundarstufe I relevant sein können. Zudem werden Aufgabenbeispiele wie etwa zur Interpretation von Listen von Zahlen als Musikstück, zu Listen von Skripten oder zur Analyse von zweidimensionalen Listen in Snap! vorgestellt und Lösungen bereitgestellt.

Trace-Tabellen*	Aufgabe + Didaktische Anmerkungen	Voraussetzung: Mündliche Einführung von Trace-Tabellen zur Darstellung der Werte von Variablen bei der Ausführung eines Algorithmus; In den didaktischen Anmerkungen werden alternative Strategien zur Analyse von möglicherweise fehlerhaften Programmen und Ausgabe von Zwischenergebnissen thematisiert.
-----------------	---	---

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht kommerziell - Keine Bearbeitungen 4.0 International Lizenz](#).

Alle Abbildungen von Scratch-Bausteinen und -Objekten sind lizenziert unter einer [Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#).

Scratch wurde entwickelt von der Lifelong Kindergarten Group, MIT Media Lab,
<http://scratch.mit.edu>