Einführung blockbasierte Programmierung

Informationen für Lehrkräfte

Einleitung

Durch die Einführung von G9 und des damit verbundenen neuen Faches *Informatik und Medienbildung* ist angedacht, einige der Inhalte des *Aufbaukurses Informatik* aus Klasse 7 bereits in Klasse 6 zu verorten. Im *Aufbaukurs Informatik* hat sich bei den meisten Lehrkräften eine Einführung in die Programmierung mithilfe der blockbasierten Programmiersprache *Scratch* etabliert.

Der vorliegende Unterrichtsgang wurde konzipiert, um sich dem Gebiet der Programmierung in Klasse 6 altersgemäß zu nähern, wobei eine auf *Scratch* basierende, jedoch reduzierte Programmierumgebung zum Einsatz kommt. Die dort erlernten grundlegenden und zentralen Konzepte dienen in Klasse 7 als Basis und können dann auf Scratch übertragen werden.

Dieses Dokument wendet sich an Lehrkräfte ohne Ausbildung in Informatik. Es wird versucht, alle relevanten fachlichen Aspekte zu thematisieren und so weit in die Tiefe zu gehen, wie es für Unterricht in Klasse 6 notwendig ist. Hierbei wurde bewusst an einigen Stellen auf informatische Fachbegriffe und Hintergründe verzichtet, sofern sie nicht unbedingt relevant sind.

Hinweis: Es sei bereits an dieser Stelle erwähnt, dass der Unterrichtsgang Funktionen nutzt, bei denen Töne ausgegeben werden. Es sollte darauf geachtet werden, dass die verwendeten Endgeräte Ton ausgeben können und den Schüler*innen Kopfhörer zur Verfügung stehen. Sollte dies nicht möglich sein, wird eine Durchführung problematisch.

Quellen und Danksagung

Der Unterrichtsgang profitiert von und basiert auf vorhandenem Material anderer Lehrkräfte bzw. Organisationen. Diesen sei an dieser Stelle ganz herzlich gedankt!

Ein besonderer Dank gilt der *WDR mediagroup*, welche freundlicherweise die Nutzung von Grafiken der eingesetzten Programmierumgebung *Programmieren mit der Maus* gestattet hat!

Die verlinkten Quellen dienen auch der vertiefenden Auseinandersetzung mit der Thematik.

- MARTIN SCHMITT & MONIKA EISENMANN: *ZPG-Fortbildungsmaterial zum Aufbaukurs Informatik*, 2017. [CC BY-SA 2.0]
- DANIEL JUNGBLUT: *Skript Informatik 7*, 2019. (ins obige ZPG-Fortbildungsmaterial eingebettet) [CC BY-SA 2.0]
- WDR & PLANET SCHULE!: *Programmieren mit der Maus* und *Begleitmaterial für Lehrkräfte*, 2019. [© WDR – *Nutzung freundlicherweise gestattet durch WDR mediagroup*]

Lizenz

Fachlicher Hintergrund

Algorithmus und Programm

Der *Algorithmus* ist eine zentrale Idee der Informatik. Er ist – vereinfacht ausgedrückt – eine „klare, endliche Abfolge von *Anweisungen*, die ein Problem löst oder eine Aufgabe ausführt“ (WIKIPEDIA: *Algorithmus*). Diese „Definition“ ist zur Verständnisbildung in Klassenstufe 6 völlig ausreichend.

„Klar“ heißt in diesem Zusammenhang, dass jede Anweisung (für Mensch und Maschine) unmissverständlich ausgeführt werden kann. Missverständlich wäre z. B. die Anweisung „drehe dich um 45°“, da unklar ist, ob die Drehung nach links oder rechts vollzogen werden soll. Klar hingegen wäre die Anweisung „drehe dich nach links um 45°“.

Ein Algorithmus kann aus einer Anweisung bestehen, aus zehn oder einer Million. Die Anzahl an Anweisungen muss jedoch beschränkt sein – sie darf nicht unendlich sein. Dies ist mit „endliche Abfolge“ gemeint.

Zu unterscheiden ist diese Vorgabe von der Frage, ob die Ausführung eines Algorithmus jemals endet. Wir betrachten den folgenden Algorithmus: „Wiederhole für immer: Drehe dich rechtsum einmal im Kreis.“ Der Algorithmus besteht aus einer *Anweisung* („Drehe dich rechtsum einmal im Kreis.“) und einer sog. *Schleife* („Wiederhole für immer“). Die Anzahl seiner Anweisungen ist also endlich. Die Ausführung des Algorithmus hingegen endet (theoretisch) nie! Man sagt auch: Der Algorithmus terminiert nicht.

Aufgrund der klaren Anweisungen ist ein Algorithmus dafür prädestiniert, durch eine Maschine (Rechner/Computer) ausgeführt zu werden. Hierfür müssen die Anweisungen des Algorithmus in eine Sprache übersetzt werden, welche die Maschine versteht. An dieser Stelle kommen *Programmiersprachen* ins Spiel. Sie stellen eine Möglichkeit dar, der Maschine mitzuteilen, was sie tun soll. Hierzu wird der Algorithmus in einer geeigneten Programmiersprache als *Programm implementiert*, welches das ursprüngliche Problem löst bzw. die Aufgabe ausführt.

Algorithmus und Programm sind jedoch im Allgemeinen nicht identisch. Während wir uns unter „drehe dich nach links um 45°“ etwas vorstellen können, hängt es bei der Maschine bzw. verwendeten Programmiersprache davon ab, welche Anweisungen er „versteht“. Während sich die meisten Programmiersprachen hierunter nichts vorstellen können, gibt es auch solche, die über eine entsprechende Anweisung verfügen. Für erstere müsste die Anweisung des Algorithmus in der Programmiersprache in dort vorhandene Anweisungen überführt werden. Da hierbei es oft mehrere Möglichkeiten gibt, ist das Ergebnis im Allgemeinen nicht identisch. Mit anderen Worten: Ein Algorithmus kann i. d. R. auf verschiedene Weise durch ein Programm implementiert werden.

Damit Programme von der Maschine fehlerfrei verarbeitet werden können, verfügen Programmiersprachen über strenge Vorgaben. Insbesondere besitzen sie eine *Syntax*: klare Regeln, wie ihre Programme aufgebaut sein müssen.

Neben den Anweisungen stehen hierbei meist folgende drei *Kontrollstrukturen* zur Verfügung:

- *Sequenz:* Mehrere Anweisungen werden nacheinander ausgeführt.
- *Schleife:* Eine Programmabschnitt wird wiederholt. Dies kann z. B. mit einer festen Anzahl geschehen („wiederhole 4 mal“), solange eine Bedingung erfüllt ist („wiederhole solange Taste gedrückt wird“) oder (theoretisch) endlos („für immer“ – also solange das Programm ausgeführt wird).
- *Verzweigung:* Ein Programmabschnitt wird nur ausgeführt, falls eine Bedingung erfüllt ist (z. B. „falls punkte > 100, gib ‚*Gewonnen!*‘ aus; ansonsten gib ‚Verloren!‘ aus“).

Im Rahmen des vorgestellten Unterrichtsgangs werden Sequenzen, Endlos-Schleifen und solche mit einer festen Anzahl behandelt. Andere Schleifen und Verzweigungen werden nicht thematisiert.

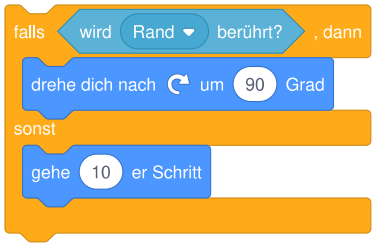
Blockbasierte Programmierung

Wie bereits angedeutet, gibt es eine Vielzahl von Programmiersprachen mit verschiedenen Eigenschaften, Zielsetzungen und Vor- und Nachteilen. Bekannte Vertreterinnen *textueller Programmiersprachen* sind *Java*, *Python* und *C++*. Textuell bedeutet, dass die Programme als Texte verfasst werden.

Textuelle Programmiersprachen besitzen insbesondere beim Einstieg den gravierenden Nachteil, dass die Einhaltung ihrer Syntax sehr herausfordernd ist. Ein vergessenes Semikolon oder Leerzeichen kann bereits dazu führen, dass ein Programm nicht übersetzt oder ausgeführt werden kann.

Aus diesem Grund haben sich seit vielen Jahren *blockbasierte Programmiersprachen* als Einstieg (und mittlerweile weit darüber hinaus) etabliert. Bekannte Vertreterinnen sind *Scratch* und *Snap!*. Ihre Sprachelemente sind Blöcke, welche dem Algorithmus entsprechend puzzleartig miteinander verbunden werden. Hierdurch werden syntaktische Fehler weitestgehend unterbunden, da ausschließlich zulässige Sprachelemente miteinander verbunden werden können. Hier ein Programmabschnitt im Vergleich:

Blockbasierte Programmiersprache



Textuelle Programmiersprache

```
if randBerührt():
    dreheRechts(90)
else:
    geheVor(10)
```

Unabhängig von der Programmiersprache selbst wird zur Programmierung meist spezielle Software (*Entwicklungs-/Programmierumgebung*) verwendet. Diese unterstützen bei der Programmierung z. B. durch das Markieren von bekannten Syntaxelementen, Hinweisen auf Fehler oder das Übersetzen und Ausführen von Programmen mit einem Klick. In manchen Fällen bilden Programmiersprache und -umgebung sogar eine konzeptionelle Einheit.

Programmieren mit der Maus

In dieser Einführung wird nicht das bereits erwähnte *Scratch* genutzt, sondern die Programmierumgebung *Programmieren mit der Maus* von der *Sendung mit der Maus*. Diese basiert auf *Scratch*, kleidet es jedoch in ein neues Gewand. Entwickelt wurde sie vom *WDR* gemeinsam mit *planet schule* im Rahmen von Schulschließungen mit dem Fokus auf eigenständiges Lernen zu Hause, weshalb kurze („Lernspiele“ genannte) Lehrgänge zentraler Bestandteil der Programmierumgebung sind. Diese umfassen zum einen Anleitungen zum Vorgehen, sodass die Schüler*innen eigenständig arbeiten können. Zum anderen ist die Palette verfügbarer Blöcke bei den Lernspielen auf jene Blöcke reduziert, die für das jeweilige Lernspiel relevant sind. Dies verhindert, sich gerade zu Beginn der Programmierung in einer Vielzahl von Blöcken zu verlieren.

Man erreicht *Programmieren mit der Maus* im Browser über

programmieren.wdrmaus.de

Die Programmierumgebung kann direkt im Browser genutzt werden, es ist keinerlei Installation auf dem Endgerät nötig. Laut eigener Angaben unterstützt die Programmierumgebung folgende Plattformen und Browser:

Computer (Windows/Linux/macOS etc.)	**Tablets**
Chrome (63+)	Mobile Chrome (62+)
Edge (15+)	Mobile Safari (11+)
Firefox (57+)	
Safari (11+)	

Darüber hinaus steht Begleitmaterial für Lehrkräfte zur Verfügung, auf dessen Idee Teile dieses Unterrichtsgang basieren und das als Inspiration für weitere Aufgaben dienen kann.

Im Unterrichtsgang wird darauf verzichtet, Programme zu speichern. Möchte man dies jedoch tun, muss man hierzu das gesamte Projekt (d. h. alle Programmabschnitte aller Figuren) über den entsprechenden Knopf oben rechts auf dem eigenen Endgerät speichern. Da die Programmierumgebung im Browser läuft, wird hierbei die Bezeichnung „Projekt herunterladen“ (statt „speichern“) verwendet. Analog kann man ein heruntergeladenes Projekt wieder „hochladen“. Die Projektdateien besitzen die Dateiendung `*.sb3`. Da die Programmierumgebung kein eigens installiertes Programm im Betriebssystem ist, kann die `sb3`-Datei nicht im Dateiexplorer durch einen Doppelklick geöffnet werden, sondern muss in der Programmierumgebung über „Projekt hochladen“ geöffnet werden. Da Schüler*innen das Öffnen von Dateien gewohnt sind, sollte das Öffnen eines Projekts unbedingt im Vorfeld demonstriert werden.

Die Programmierumgebung besitzt keinen Knopf, um Aktionen rückgängig zu machen. Die gängige Tastenkombination `Strg + Z`, welche die Schüler*innen vermutlich bereits aus der Textverarbeitung kennen, kann jedoch zu diesem Zweck verwendet werden.

Programmieren mit der Maus ist ein Open-Source-Projekt und sein Quellcode auf Github veröffentlicht.

Hinweis: Der Unterrichtsgang nutzt Funktionen der Programmierumgebung, bei denen Töne ausgegeben werden. Es sollte darauf geachtet werden, dass die verwendeten Endgeräte Ton ausgeben können und den Schüler*innen Kopfhörer zur Verfügung stehen.

Unterrichtsgang

Bei der Beschreibung des Unterrichtsgangs werden Inhalte der Arbeitsblätter nur im Ansatz wiederholt. Eine parallele Lektüre bietet sich an.

Der zeitliche Umfang hängt beträchtlich von der Lerngruppe und der konkreten Ausgestaltung ab und wird mit mindestens 3 Unterrichtsstunden eingeschätzt, kann jedoch sicherlich auf mindestens 6 Unterrichtsstunden ausgeweitet werden.

1 Programmieren unplugged (1–2 Stunden)

Lernziele

Die Schüler*innen können …

- analog *Programme* aus Anweisungen erstellen, welche eine Figur auf einem gerasterten „Programmierfeld“ steuern.
- aus Sequenzen bestehende Programme auf diesem Programmierfeld ausführen.

Hintergrundinformationen

Im Zentrum der ersten Stunde steht ein spielerischer, analoger Einstieg ins Thema („unplugged“): Auf einem gerasterten „Programmierfeld“ befinden sich die Maus und der Elefant. In allen Aufgabenstellung ist das Ziel, die Maus zum Elefanten zu bewegen. Hierzu wird sie von uns durch ein Programm wie ein Roboter gesteuert.

Auf eine Definition und die Verwendung des Begriffs *Algorithmus* wird im Unterrichtsgang verzichtet. Es hat Vorteile, ihn an späterer Stelle z. B. in Klasse 7 mit etwas Programmiererfahrung einzuführen, bei Bedarf kann er jedoch bereits hier thematisiert werden.

Zum Einsatz kommen lediglich drei Anweisungen in Form von Anweisungskarten:

V Die Maus bewegt sich in Blickrichtung ein Feld **v**or.

R Die Maus dreht sich um 90 Grad nach **r**echts.

L Die Maus dreht sich um 90 Grad nach **l**inks.

Eine erste zentrale Erkenntnis (gleichzeitig verbunden mit einer potenziellen Fehlvorstellung) ist, dass die von uns gesteuerte Maus und wir selbst verschiedene Perspektiven einnehmen. Eine Drehung nach rechts findet aus Blickrichtung der Maus nach rechts statt. Je nachdem, wie wir das Programmierfeld betrachten, kann dies für uns z. B. eine Drehung nach unten sein.

Aus diesem Grund sollte vermieden werden, die Anweisungskarten auf das Programmierfeld zu legen. Zusätzlich würde hierbei das Problem entstehen, dass auf einem Feld mehr als eine Anweisung ausgeführt werden müsste (z. B. drehen und ein Feld vor bewegen).

Die Fachbegriffe *Anweisung*, *Sequenz* und *Programm* werden eingeführt. Ein Programm kann aus einer Anweisung oder Sequenz bestehen – theoretisch auch aus weiteren, noch unbekannten Kontrollstrukturen. Wie in der Programmierumgebung werden die Programme durch die An-

weisungskarten zuerst von oben nach unten ausgeführt. Es sollte darauf geachtet werden, diese Fachbegriffe bereits konsequent zu verwenden, obwohl eine explizite Definition noch folgt.

Beim Bearbeiten des Arbeitsblattes wird schnell herausgearbeitet, dass es für die Lösung einer Aufgabe bzw. eines Problems nicht nur eine, sondern eine Vielzahl geeigneter Programme (bzw. Algorithmen) gibt. Eine Bewertung einer Lösung kann an dieser Stelle hinsichtlich der Anzahl verwendeter Anweisungen stattfinden. Perspektivisch gibt es bei komplexeren Programmen weitere potenzielle Kriterien, wie z. B. die Anzahl *ausgeführter* Anweisungen, die zusätzlich unterschiedlich „rechenintensiv“ sein können.

In weiteren Aufgaben werden die ikonischen Befehle auf den Aufgabenkarten durch Buchstaben ersetzt, die nunmehr von links nach rechts ausgeführt werden. Dieser Wechsel der Darstellung zeigt auf, dass eine Programmiersprache letztendlich durch Menschen definiert wird und verschiedene Darstellungen unterschiedliche Vor- und Nachteile haben, z. B.intuitive Nachvollziehbarkeit (ikonische vs. symbolische Darstellung), Platzbedarf, schnelles Einfügen einer Anweisung, … Darüber hinaus bietet sich ein Rückbezug zum Thema *Codierungen* an, da hier die Anweisungskarten zweckmäßig durch Buchstaben codiert werden.

Vorbereitung

- Modell des Programmierfeldes drucken, schneiden und Figurenaufsteller kleben:

  `01_Programmieren_unplugged_Druckvorlage_Klassenunterricht_A4.odt`
  oder …`Klassenunterricht_A3.odt`

- Arbeitsblatt ausdrucken:

  `01_Programmieren_unplugged.odt` (mit Anweisungskarten) **oder**
  `01_Programmieren_unplugged_ohne_Anweisungskarten.odt`

- Anweisungskarten für Arbeitsblatt ausdrucken und schneiden (Briefumschläge oder andere Behältnisse bieten sich an):

  `02_Programmieren_unplugged_Druckvorlage_Einzelarbeit.odt`

Stundenverlauf

Es bietet sich an, als Einführung die Unplugged-Programmierung auf dem Programmierfeld im Klassenunterricht gemeinsam oder in Gruppen einzuführen. Hierzu stehen Druckvorlagen für A3 und A4 (bzw. mehrseitig) zur Verfügung, die möglichst groß gedruckt, geschnitten und geklebt werden sollte.

Auf der ersten Seite des Arbeitsblattes befinden sich alle wichtigen Informationen zur Unplugged-Programmierung, welche durch die Lehrkraft im Vorfeld frontal präsentiert, im Klassenunterricht vorgelesen oder eigenständig gelesen werden.

Anschließend werden die Aufgaben des Arbeitsblattes bearbeitet. Bei Aufgabe 1 kann die Programmierung zuerst durch Anweisungskarten durchgeführt werden. Das umständliche Übertragen der Anweisungskarten der resultierenden Sequenz auf das Arbeitsblatt bereitet bereits den oben thematisierten Wechsel zu einer anderen Darstellung des Programms vor.

Erprobungen haben gezeigt, dass die „haptische Programmierung“ in Aufgabe 1 für das Verständnis der Schüler*innen – je nach Klasse – nicht notwendig ist. Aufgrund der verhältnismäßig aufwendigen Herstellung der Anweisungskarten in Klassenstärke und ihres kurzen Einsatzes im Unterricht steht auch ein alternatives Arbeitsblatt zur Verfügung, in dem auf sie verzichtet wird.

Die Lösungen der Aufgaben werden dann auf geeignete Weise gesichert.

2 Programmierumgebung, Anweisung & Sequenz (1–2 Stunden)

Lernziele

Die Schüler*innen können …

- ein Programm in einer blockbasierten Programmiersprache erstellen, starten und stoppen.
- beschreiben, was *Anweisung* und *Sequenz* sind und hierfür Beispiele in einer blockbasierten Programmiersprache nennen.
- Programmabschnitte mit Sequenzen und „für immer“-Schleife untersuchen und deren Wirkung auf der Bühne beschreiben.

Hintergrundinformationen

Nun findet der Übergang zur Programmierung am Rechner statt. Es hat sich bewährt, den Bereichen der blockbasierten Programmierumgebung explizit Namen zu geben, um sie eindeutig benennen zu können.

Die Programmierumgebung stellt viele für Schüler*innen attraktive Funktionen zur Verfügung (Kostüme, Töne, Hintergründe, …). Es ist damit zu rechnen, dass die meisten Schüler*innen diese Funktionen erkunden werden, obwohl dies durch die Anleitung nicht angewiesen wird. Dieser Neugierde darf man in einem angemessenen Rahmen durchaus Raum geben – es wäre schade, die Motivation an dieser Stelle zu schnell auszubremsen.

Die Begriffe *Anweisung* und *Sequenz* werden nun prominent definiert und mit Beispielen konkretisiert. Außerdem wird die „für immer“-Schleife eingeführt, jedoch noch nicht als solche benannt. Ihre Funktion sollte an dieser Stelle intuitiv klar werden.

Im Block „gleite in ___ Sek. zu ___“ sind auch Sekundenangaben unter einer Sekunde zulässig. Hierbei ist zu beachten, dass der Punkt als Dezimaltrenner zu verwenden ist, also z. B. 0.4 statt 0,4.

Vom Speichern der Programme wird an dieser Stelle abgesehen, da sie zukünftig nicht mehr benötigt werden.

Vorbereitung

- Arbeitsblatt ausdrucken:

  `02_Programmierumgebung_Anweisung_Sequenz.odt`

Stundenverlauf

Die Programmierumgebung kann zu Beginn des Themas durch die Lehrkraft vorgestellt werden. Hierbei werden ihre Bereiche benannt und beschrieben. Es kann auch bereits demonstriert werden, wie man Blöcke in den Programmierbereich zieht und ein erstes kurzes Programm startet und stoppt.

Anschließend werden die Aufgaben des Arbeitsblattes bearbeitet und ihre Lösungen dann auf geeignete Weise gesichert.

3 Schleifen (1–2 Stunden)

Lernziele

Die Schüler*innen können …

- beschreiben, was eine *Schleife* ist und Beispiele in einer blockbasierten Programmiersprache hierfür nennen.
- mit den ihnen bisher bekannten Sprachelementen Programme zur Lösung einfacher Probleme implementieren.
- Programmabschnitte mit den ihnen bisher bekannten Sprachelementen untersuchen und deren Wirkung auf der Bühne beschreiben.

Hintergrundinformationen

Als erstes wird aufgezeigt, wie man die Maus mithilfe eines Kostümwechsels die Augen schließen und wieder öffnen lässt, sodass sie blinzelt. Es ist sinnvoll, ein Schließen und Öffnen der Augen (Lidschlag) als ein Blinzeln aufzufassen. Hierzu besitzt die Maus bereits zwei unterschiedliche Kostüme (eines mit geöffneten und eines mit geschlossenen Augen). Bei den Übungen ist zu beachten, dass die Maus immer mit geöffneten Augen starten sollte. Ggf. muss das Kostüm vor dem Start manuell über den „nächstes Kostüm“-Block in der Blockpalette zu den geöffneten Augen gewechselt werden.

Neben der „für immer“-Schleife, welche bereits im vorherigen Abschnitt zum Einsatz kam, wird nun die Schleife mit konstanter Anzahl eingeführt. Wie bereits beschrieben, werden weitere Schleifen-Arten nicht thematisiert.

Darüber hinaus wird in beiden Lernspielen mit Tönen gearbeitet. Die Schüler*innen sollten hierzu optimalerweise mit Kopfhörern ausgestattet sein.

Die Möglichkeiten, Schleifen in den behandelten Lernspielen einzusetzen, sind begrenzt. Es besteht die Möglichkeit, mithilfe des Modells aus *1 Programmieren unplugged* Schleifen auf dem Programmierfeld zu thematisieren. Ein typisches Beispiel ist die Aufgabe, die Maus ein Quadrat laufen zu lassen. Nachdem dies durch eine Sequenz gelöst wurde (*Programm 1*), ist die Wiederholung der Sequenz VR mithilfe einer Schleife (*Programm 2*) naheliegend und wird i. d. R. durch Schüler*innen vorgeschlagen.

Programm 1

Programm 2

Dieses Erkennen gleichartiger Strukturen (in diesem Fall eine sich wiederholende Sequenz innerhalb einer Sequenz) und definieren einer kleineren Teillösung des Problems (ein Schritt vor und drehen), auf welche dann zurückgegriffen wird (hier mithilfe der Schleife) ist ein zentraler Vorgang in der Informatik (im Kleinen ist dies eine Ausprägung der sog. *Top-Down-Methode*).

Die fachlich größte Herausforderung sind sog. verschachtelte Schleifen, d. h. eine Schleife befindet sich innerhalb einer anderen Schleife. Für jeden Durchlauf der äußeren Schleife wird deren Inhalt – in diesem Fall weitere innere Schleifen – jeweils einmal ausgeführt.

Vorbereitung

- Ton auf Endgeräten überprüfen und Schüler*innen bitten, Kopfhörer mitzubringen und/ oder Leihgeräte beschaffen
- Arbeitsblatt ausdrucken:

  `03_Schleife.odt`
- Modell von *1 Programmieren unplugged* parat haben

Stundenverlauf

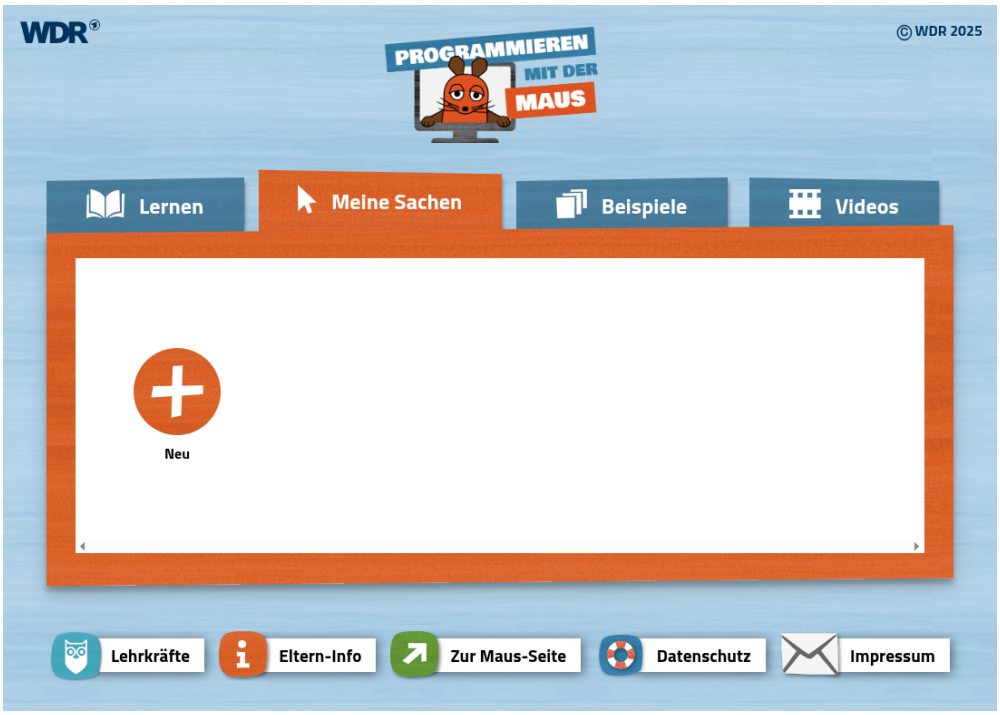
Die Lehrkraft führt das Konzept der Schleife mithilfe des Modells aus *1 Programmieren unplugged* frontal und gemeinsam mit der Klasse ein (s. Hintergrundinformationen).

Anschließend werden die Aufgaben des Arbeitsblattes bearbeitet und ihre Lösungen dann auf geeignete Weise gesichert.

Ausblick

Es besteht die Möglichkeit, das Thema je nach Schwerpunktsetzung im Unterricht und verbleibender Unterrichtszeit zu vertiefen. An dieser Stelle einige Gedanke hierzu:

- Es bietet sich an, mit Lernspiel 04 *„Wimmelbild“* fortzufahren. Hier werden folgende Inhalte behandelt:
  - Programmabschnitte ausführen, wenn Figur angeklickt wird
  - weitere Figuren ins Programm hinzufügen
  - Programmabschnitte von einer zu einer anderen Figur kopieren
- Es wird *dringend* empfohlen, Lernspiel 05 *„Mathefant“* und nachfolgende Lernspiele nicht mehr zu bearbeiten. In Lernspiel 05 werden viele weitere Programmierkonzepte eingeführt, welche den erwarteten Rahmen von Klasse 6 deutlich übersteigen. Außerdem ist zur Vorbereitung eine fundierte fachliche und didaktische Auseinandersetzung mit den Inhalten nötig.
- Mit Fokus auf Medienbildung bietet sich an, die Programmierumgebung zur Medienproduktion zu nutzen und die Schüler*innen kreativ mit ihr arbeiten und z. B. eine Animation erstellen zu lassen.
  - Von den Lernspielen muss zuerst in die vollumfängliche Programmierumgebung gewechselt werden. Es steht dann annähernd der komplette Funktionsumfang von *Scratch* zur Verfügung. Hierzu muss in der Übersicht unter *Meine Sachen* ein neues Projekt erstellt werden:

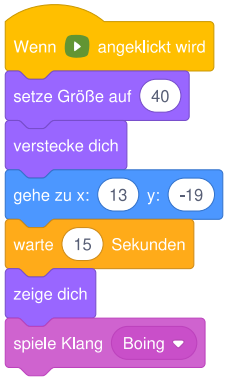


- Die Lehrkraft sollte für sich den Umfang zu nutzender Blöcke im Vorfeld festlegen und diese frontal einführen, ein Handout oder Erklärvideo erstellen. Hier ein Beispiel mit den drei Figuren *Maus*, *Elefant* und *Apfel*.

Maus

Elefant

Apfel



Es wird deutlich, dass selbst bei einer kurzen Animation diverse neue Blöcke benötigt werden. Es spricht natürlich nichts dagegen, dass Schüler*innen auch andere Blöcke nutzen, es ist jedoch sinnvoll, ihnen eine kompakte Auswahl relevanter Blöcke vorzustellen, damit sie sich nicht in der Blockpalette verlieren.

- Im bisherigen Unterrichtsgang wurde das absolute Positionieren von Figuren auf der Bühne umgangen, indem auf Zufallspositionen zurückgegriffen wurde. Bei Animationen ist dies meist nicht zielführend. Deshalb muss thematisiert werden, dass jede Figur eine Position in Form eines Punktes mit x- und y-Koordinate auf der Bühne besitzt. Koordinatensysteme sollten aus dem Mathematikunterricht zu diesem Zeitpunkt bekannt sein. Das zugrundeliegende Koordinatensystem hat seinen Ursprung in der Mitte der Bühne. Durch „gehe zu x: ___ y: ___“ „springt“ eine Figur zum Punkt (x | y).
- Zum Festigen der bereits erlernten Inhalte sollte darauf geachtet werden, dass z. B. auch eine Schleife verwendet wird.
- Inhaltlich bietet es sich an, den Rahmen der Animation vorzugeben. Z. B. könnte sie genutzt werden, um einen bereits im Unterricht behandelten, inhaltlichen Aspekt eines anderen Gebietes zu thematisieren: Regeln im Klassenchat, Recht am eigenen Bild, Codierung, Textverarbeitung, …
- In diesem Kontext sollte dann auch das Speichern und Öffnen von Projekten behandelt werden.
- Zur Sicherung der Animationen kann z. B. ein Gallery Walk durchgeführt werden.