Co-funded by the Erasmus+ Programme of the European Union

Modul 2

Allgemeine Einführung in Computational Thinking

Ein Basismodul, das für alle Lehrkräfte geeignet ist

Autoren: Radboud University (Niederlande)
Maria Kallia,
Sjaak Smetsers,
Erik Barendsen,
Christos Chytas

Rezensenten:
Arnold Pears (KTH),
Valentina Dagienė (VU)

Externe Rezensenten:
Piret Luik (Estland),
Renate Motschnig (Österreich)

Pilotierung:
Ankara University (Türkei), KTH Royal Institute of Technology (Schweden), Radboud University (Niederlande), University of Paderborn (Deutschland), Vienna University of Technology (Österreich)

Design:
Vaidotas Kinčius (Litauen)

Modul *2* basiert auf der Arbeit im Rahmen des Projekts "Future Teachers Education: Computational Thinking and STEAM" (TeaEdu4CT). Koordination: Prof. Valentina Dagienė, Universität Vilnius, Litauen. Partner: Technische Universität Wien (Österreich), CARDET (Zypern), Universität Tallinn (Estland), Universität Turku (Finnland), Universität Paderborn (Deutschland), CESIE (Italien), Radboud Universität (Niederlande), KTH Royal Institute of Technology (Schweden), Universität Ankara (Türkei). Das Projekt wurde durch das Erasmus+-Programm KA2 kofinanziert.

Inhalt

Allgemeiner Überblick und Ziel

Dieses Modul zielt darauf ab, angehenden Lehrkräften ein konkretes Verständnis von Computational Thinking (CT, Informatisches Denken) und Kenntnisse über die Lehr- und Lernprinzipien für CT zu vermitteln und sie mit Computertools vertraut zu machen, die das Lehren und Lernen unterstützen können.

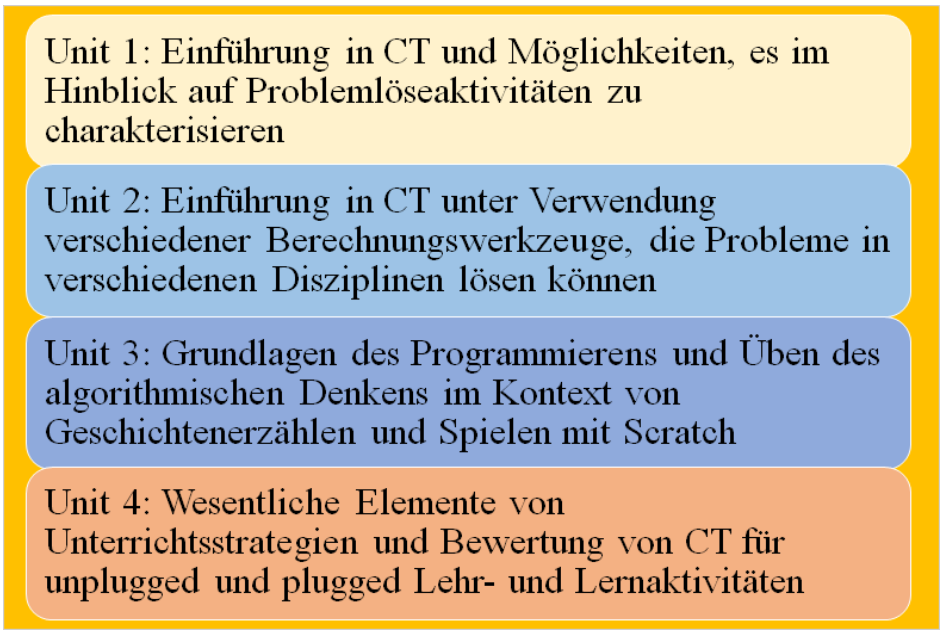
In diesem Modul werden die Lernenden:

- das Konzept des Computational Thinking erforschen und verschiedene Möglichkeiten vergleichen, wie es in Bezug auf Problemlösungsaktivitäten charakterisiert werden kann
- sich mit Aspekten, Praktiken und Werkzeugen des Computational Thinking auseinandersetzen und die Rolle des Computational Thinking innerhalb der Disziplinen verstehen
- sich mit den Grundlagen der Programmierung vertraut machen und algorithmisches Denken im Kontext von Geschichten und Spielen üben
- sowohl unplugged als auch plugged Lehr- und Lernaktivitäten erleben und Gestaltungsprinzipien für Lehrstrategien für Computational Thinking kennenlernen

Das Modul betrachtet das Computational Thinking als Rahmen für die Entwicklung fächerübergreifender Fähigkeiten und Kompetenzen, die sich für Lehrkräfte aller Fächer eignen.

Die Modulstruktur

Zu diesem Zweck ist das Modul in die folgenden vier Units gegliedert: Unit 1 führt in das Konzept des Computational Thinking ein und vergleicht Möglichkeiten, es im Hinblick auf Problemlösungsaktivitäten zu charakterisieren; Unit 2 führt in das CT ein, indem verschiedene Computerwerkzeuge (Ngrams, NetLogo, Excel) verwendet werden, und zeigt, wie diese Werkzeuge eingesetzt werden können, um Probleme in verschiedenen Disziplinen (Geschichte, Biologie, Geographie) zu lösen; Unit 3 konzentriert sich auf die Grundlagen des Programmierens und auf das Üben des algorithmischen Denkens im Kontext des Geschichtenerzählens und von Spielen mit Hilfe von Scratch. Das Modul schließt mit Unit 4 ab, die unplugged und plugged Lehr- und Lernaktivitäten hervorhebt und wesentliche Elemente von Unterrichtsstrategien und Bewertungen für das CT erörtert.



Zielgruppen und Voraussetzungen

Dieses Modul richtet sich an angehende Lehrerinnen und Lehrer, die ein Lehramtsstudium absolvieren, sowie an Lehrerinnen und Lehrer in der beruflichen Weiterbildung, die sich für das Thema "Computational Thinking" interessieren. Das Modul ist für den Präsenzunterricht konzipiert, kann aber leicht als Fernlernmodul angepasst werden.

Für das Studium dieses Moduls gibt es keine besonderen Voraussetzungen. Es wäre ratsam, dass die Studierenden über ein Grundwissen über die verwendeten Werkzeuge verfügen und das vorherige Modul 1 "Rahmen für die Unterstützung der Module“ abgeschlossen haben.

Lernergebnisse und Bewertungsmethoden

Studierende, die das gesamte Modul abgeschlossen haben, werden in der Lage sein:

- das Konzept des Computational Thinking zu erläutern und verschiedene Möglichkeiten zur Charakterisierung dieses Denkens im Hinblick auf Problemlösungsaktivitäten zu vergleichen;
- Problemlösungsaufgaben unter Verwendung von Konzepten, Praktiken und Werkzeugen des Computational Thinking durchzuführen und die Rolle des Computational Thinking innerhalb verschiedener Fächer zu erklären;
- einfache Programme in Scratch zu erstellen und algorithmisches Denken im Kontext von Geschichten erzählen und Spielen anzuwenden;
- unplugged und plugged Lehrstrategien und Lernaktivitäten für Computational Thinking anzuwenden und die zugrunde liegenden Gestaltungsprinzipien zu beschreiben.

Spezifischere Lernziele sind in den Strukturbeschreibungen der einzelnen Units zu finden.

Bewertungsstrategie

Die Aufgaben in den einzelnen Modulen bieten Gelegenheit zur formativen Bewertung und zum Feedback. Es gibt eine optionale abschließende (Gesamt-)Bewertung, die in einem separaten Dokument zu finden ist (siehe Lernressourcen: Modul Abschlussbewertung).

Modulplan und didaktische Ansätze

Das Modul besteht aus 4 Units, die face-to-face stattfinden sollen. Jede Unit umfasst mehrere Aktivitäten, die in der Regel mit einer Aufwärmaktivität beginnen und mit einer Reflexionsaktivität enden. Die Studierenden werden sich mit einer Vielzahl von Lernansätzen beschäftigen, die das Lesen von Artikeln und bereitgestellter Literaturübersichten, Gruppendiskussionen, Problemlösungsaufgaben und Reflexionsaktivitäten umfassen.

Units und Aktivitäten

Unit 1: Einführung in das Computational Thinking

Aktivität 1.1 Einführung in das CT

- Computational Thinking als Problemlösungsaktivität
- Charakterisierung von Elementen des Computational Thinking
- Schlussfolgerung

Dauer: 2 Stunden

Unit 2: Werkzeuge für das Computational Thinking

Aktivität 2.1: Google Ngrams verwenden

Aktivität 2.2: Modellierung und Simulation mit NetLogo

Aktivität 2.3 Microsoft Excel verwenden

Dauer: 8 Stunden

Unit 3: Computational Thinking durch Programmieren erleben

Aktivität 3.1: Geschichtenerzählen in Scratch

Aktivität 3.2: Ein Labyrinthspiel erstellen

Dauer: 10 Stunden

Unit 4: Lehr- und Lernstrategien für Computational Thinking

Aktivität 4.1: Bebras Aufgaben

Aktivität 4.2: Weg finden: Pfade in einem Labyrinth

Aktivität 4.3: Unterrichtsstrategien für Computational Thinking

Dauer: 8 Stunden

UNIT 1: Einführung in das Computational Thinking

In dieser Unit werden Sie das Konzept des Computational Thinking erforschen und verschiedene Möglichkeiten vergleichen, es in Bezug auf Problemlösungsaktivitäten zu charakterisieren.

Lernergebnisse

Lernergebnisse: Studierende können nach der Unit 1
• Computational Thinking als Bindeglied zwischen einem Fachgebiet und der IT beschreiben
• Computational Thinking in Form von Problemlösungsschritten und -aktivitäten charakterisieren, die zu einer operationellen, ausführbaren Lösung führen
• Elemente des Computational Thinking in Szenarien für Schüleraktivitäten erkennen

Aktivität 1.1 Einführung in das Computational Thinking

Um das volle Potenzial von Computern in einer Vielzahl von Fächern und Aktivitäten nutzen zu können, sind mehr digitale Kompetenzen erforderlich als die Fähigkeit, ein Programm wie Word zu bedienen oder ein Social-Media-Konto zu pflegen. Um eine solche Verbindung zwischen Fachinhalten und Informationstechnologie herzustellen, sind spezifische Problemlösungskompetenzen erforderlich. Diese Fähigkeiten fallen in den Bereich des Computational Thinking.

Computational Thinking als Problemlösungsaktivität

Leseaufgabe

Lesen Sie das Dokument Material A "Eine kurze Einführung in das Computational Thinking", das Sie bei den Lernressourcen finden. In diesem Text werden drei Hauptschritte des Computational Thinking unterschieden: die Pfeile (1), (2) und (3) im Diagramm.

Paardiskussion

Diskutieren Sie die beiden folgenden Unterrichtsszenarien, die Yadav et al. (2018, S. 381) entnommen sind. Welche Aktivitäten würden Sie als Computational Thinking bezeichnen? Wie hängen sie mit den Schritten (1), (2) und (3) in Material A zusammen?

Szenario 1:

Die Westwood-Grundschule wird das nächste Schuljahr mit einer 1:1 iPad-Initiative beginnen. Herr Nowak hat beschlossen, dass seine Schüler der 2. Klasse mit ihren iPads das Wetter (Temperatur, Niederschlag und Wind) für eine Woche vorhersagen sollen. Jeder Schüler malt ein Bild davon, wie das Wetter seiner Meinung nach aussehen wird. Sara, eine Schülerin, wollte auch die Temperaturen festhalten, die jeder vorausgesagt hat. Herr Nowak erstellte eine Google-Tabelle, in die jeder Schüler seine vorausgesagten Temperaturen eintrug. Am nächsten Tag zeichneten sie das tatsächliche Wetter auf, indem sie die Accuweather App auf ihren iPads benutzten und die Informationen in die Google-Tabelle eintrugen. Olivia wollte auch die tatsächliche Temperatur in Saras Tabelle eintragen, damit sie vergleichen konnten, wie ihre Vorhersagen im Vergleich zum tatsächlichen Wetter waren. Nach einer Woche projizierten sie die Google-Tabelle auf das Smartboard und zogen die Unterschiede zwischen den beobachteten und vorhergesagten Temperaturen ab. Herr Nowak zeigte ihnen, wie man ein Balkendiagramm dieser Unterschiede erstellt.

Szenario 2:

Alle Klassen der zweiten Klasse machen einen Ausflug! Die Schulcafeteria hat für alle PB&J-Mittagessen in identischen Papiertüten gepackt, außer für Sara und Olivia, die eine Erdnussallergie haben. Die Papiertüten wurden mit den Namen aller Schülerinnen und Schüler beschriftet und in 10 Boxen mit je 10 Pausenbroten aufgeteilt. Die Pausenbrote wurden in alphabetischer Reihenfolge nach Nachnamen in die Boxen gelegt. Herr Nowak möchte sich vergewissern, dass Sara und Olivia erdnussfreie Pausenbrote erhalten. Sie helfen ihm beim Durchsuchen der Kartons. Olivia Velazquez weiß, dass ihr Pausenbrot wahrscheinlich ganz am Ende stehen wird, also sieht sie sich das erste Pausenbrot in jeder Schachtel an, bis sie eine findet, die mit einem Buchstaben beginnt, der nahe am Ende des Alphabets liegt. Als sie die Schachtel findet, die mit dem Mittagessen von Jemal Summer beginnt, sieht sie sich das letzte Mittagessen in dieser Schachtel an. Es ist das von Billy Wagner, also weiß sie, dass sie nah dran sein muss! Sie sieht sich das Mittagessen an, das direkt neben Billys Mittagessen liegt, und es ist ihres. Glücklich stellt sie fest, dass die Cafeteria daran gedacht hat, ihr ein Käsesandwich und Karotten einzupacken.

Charakterisierung von Elementen des Computational Thinkings

Die Schritte der Problemlösung mit CT lassen sich global wie folgt beschreiben.

1. Dekontextualisierung: Übersetzung des Problems oder der Frage in einem Fachgebiet ("Kontext") in informatische Begriffe;
2. informatische Problemlösung: Konstruktion einer ausführbaren Lösung;
3. (Re-)Kontextualisierung: Rückübersetzung der Lösung in den Fachbereich.

Die Definitionen von Computational Thinking unterscheiden sich in der Art und Weise, wie sie die in den Schritten (1), (2) und (3) ausgeführten Aktivitäten betonen. Selby und Woollard (2013, S. 5) beschreiben beispielsweise Computational Thinking als "eine oft produktorientierte Aktivität, die mit Problemlösung verbunden, aber nicht darauf beschränkt ist. Es ist ein kognitiver oder gedanklicher Prozess, der Folgendes widerspiegelt

- die Fähigkeit, in *Abstraktionen* zu denken,
- die Fähigkeit, in Kategorien der *Zerlegung* zu denken,
- die Fähigkeit zu *Algorithmischem Denken*,
- die Fähigkeit, in Form von *Bewertungen* zu denken,
- die Fähigkeit, in *Verallgemeinerungen* zu denken".

Obwohl es einige Überschneidungen gibt, können wir die oben genannten Elemente global mit den Schritten in unserem Modell in Verbindung bringen. Die ersten beiden Elemente haben hauptsächlich mit Schritt (1) zu tun: die Analyse von Mustern innerhalb eines Problems oder einer Situation (Abstraktion) und die Zerlegung eines Problems in kleinere Teilprobleme (Dekomposition). Algorithmisches Denken kommt vor allem in Schritt (2) zum Einsatz, während die Bewertung einer Lösung und die Untersuchung, wie sie verallgemeinert werden kann, also die informatische Lösung mit dem Fachgebiet verbindet, was in Schritt (3) geschieht. Wir können dies in einer Tabelle zusammenfassen:

	(1)	(2)	(3)
Selby & Woollard (2013)	Abstraktion, Zerlegung	algorithmisches Denken	Bewertung, Verallgemeinerung

Paaraufgabe

Drei weitere bekannte Operationalisierungen finden Sie in dem Dokument "B. Definitionen von CT". Erweitern Sie die obige Tabelle um drei weitere Zeilen. Kategorisieren Sie die Elemente, die Sie in den drei Spalten finden. Erkennen Sie Elemente aus den Unterrichtsszenarien 1 und 2 wieder?

Schlussfolgerung

Allgemeine Diskussion

Vergleichen Sie Ihre Ergebnisse mit Ihren Mitstudierenden in ihrem Kurs. Erstellen Sie gemeinsam eine kurze Arbeitsdefinition von Computational Thinking in Form von Schritten und Aktivitäten.

Lernressourcen

A. Kurze Einführung in CT

B. Definitionen von CT

Referenzen

Selby, C. & Woollard, J. (2013) Computational thinking: the developing definition, verfügbar im Internet: http://eprints.soton.ac.uk/356481, Zugriff: 10. Februar 2021

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: measuring teacher understanding of computational ideas for teaching science. *Computer Science Education*, 28(4), 371-400.

UNIT 2: Werkzeuge des Computational Thinking

In dieser Unit werden Sie praktische Erfahrungen mit Lernaktivitäten zum Computational Thinking sammeln, bei denen einfache, aber leistungsstarke digitale Werkzeuge für reale Problemlösungen in verschiedenen Disziplinen zum Einsatz kommen - im Moment ist keine Programmierung erforderlich.

Die Unit stellt drei Tools vor, nämlich NetLogo, Microsoft Excel und Google Ngrams, und präsentiert Aktivitäten, die zeigen, wie diese Tools verwendet werden können, um Computational Thinking in verschiedene Schulfächer einzubinden. Die Beschreibung der einzelnen Tools sowie ihre Anwendungen in verschiedenen Fächern werden in den jeweiligen Abschnitten detailliert beschrieben. Während Sie diese Aktivitäten durchlaufen, setzen Sie sich mit den Aspekten, Praktiken und Werkzeugen des Computational Thinking auseinander und verstehen die Rolle des Computational Thinking in den verschiedenen Disziplinen.

Beitrag zu den Lernergebnissen

Lernergebnisse: Studierende können am Ende der Unit 2
• Computational Thinking in der Praxis anhand einer Reihe von Fallbeispielen anwenden • Probleme als informatisch lösbar klassifizieren • informatische Werkzeuge zur Problemlösung einsetzen • detaillierte Schritt-für-Schritt-Lösungen für Problemstellungen entwickeln • über Daten nachdenken, sie interpretieren und visualisieren • Datenanalyse zum besseren Verständnis natürlicher Systeme nutzen • bewerten, welche Art von Problemen mit Hilfe von Modellierung und Simulation gelöst werden können • verstehen und beschreiben, wie Modellierung und Simulation zur Lösung eines Problems eingesetzt werden können • verstehen und beschreiben, wie Modellierung und Simulation zur Lösung eines Problems eingesetzt werden können • Daten analysieren und Muster durch Modellierung und Simulation erkennen • mit Kolleginnen und Kollegen zur Lösung eines Problems zusammenarbeiten und kommunizieren

Tätigkeit 2.1: Google Ngrams verwenden

Gesamtzeit: 2 - 3 Stunden

Der Google Ngram Viewer ist ein Graphik-Tool, das Worthäufigkeiten aus einem großen Buchkorpus darstellt und die Verwendung eines Begriffs oder einer Phrase im Laufe der Zeit visualisiert. Das Tool verwendet Literaturquellen, die zwischen 1500 und 2008 in amerikanischem Englisch, britischem Englisch, Französisch, Deutsch, Italienisch, Spanisch, Russisch, Hebräisch und Chinesisch gedruckt wurden, und kann somit eingesetzt werden, um Sprachveränderungen im Laufe der Jahre zu untersuchen. Schwankungen im Sprachgebrauch können soziokulturelle Veränderungen aufzeigen und somit ein Verständnis dafür vermitteln, wie sich verschiedene soziokulturelle Veränderungen im Laufe der Geschichte entwickeln. In dieser Aktivität werden wir Google Ngrams verwenden, um kulturelle und soziale Veränderungen im Laufe der Zeit zu untersuchen, wie sie sich in der Verwendung bestimmter Begriffe in Büchern widerspiegeln.

Einführende Diskussion

Bevor Sie mit der Arbeit an dieser Aktivität beginnen, klicken Sie auf den folgenden Link, um den Google Ngram Viewer (https://books.google.com/ngrams) aufzurufen. Sie werden feststellen, dass das Tool bereits ein Beispiel vorschlägt, das in der folgenden Abbildung dargestellt ist. In diesem Beispiel sehen wir die Ergebnisse für drei Phrasen, "Albert Einstein", "Sherlock Holmes" und "Frankenstein". Das Tool ermittelt die Häufigkeit dieser Ausdrücke, wie sie in Büchern zwischen 1800-2008 vorkommen. Wie Sie sehen können, wird der Begriff Frankenstein seit den frühen 1800er Jahren in der Literatur erwähnt, während die beiden anderen Begriffe erst später in der Literatur auftauchen.

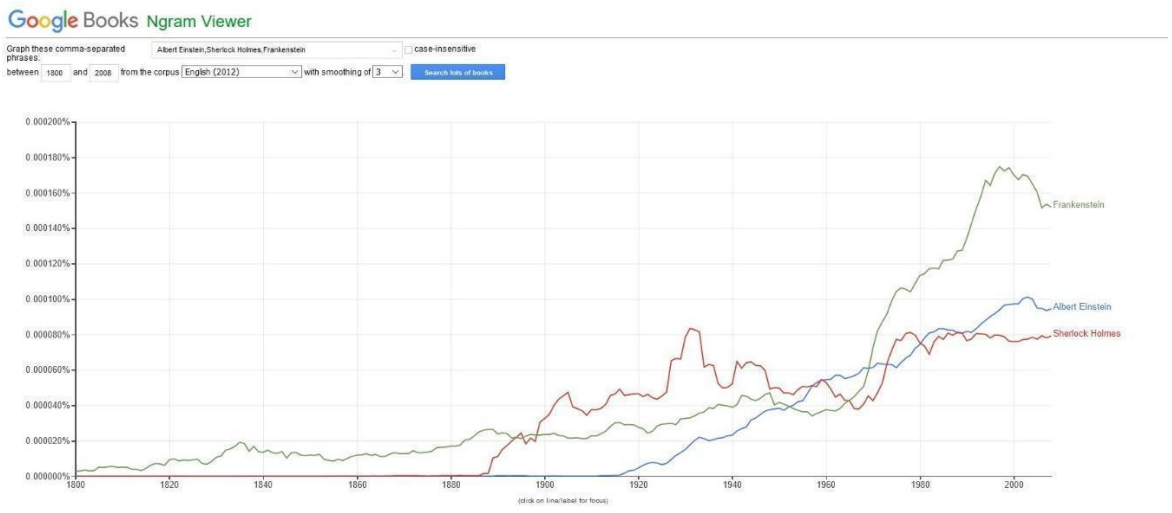


Abbildung 1 Google Ngram Beispiel für die Begriffe: "Albert Einstein, Sherlock Holmes und Frankenstein".

Diskussion

Welche anderen Tendenzen stellen Sie hinsichtlich der Häufigkeit der Begriffe in Abbildung 1 fest?

Phase 1: Problemstellung

Die Frage, der wir in dieser Aktivität nachgehen werden, bezieht sich auf die kulturellen Veränderungen in den Vereinigten Staaten in den letzten zwei Jahrhunderten. Das soziokulturelle theoretische Objektiv, das wir verwenden werden, wird im folgenden Absatz beschrieben, gefolgt von unserer Frage.

Die Theorie des sozialen Wandels und der menschlichen Entwicklung sagt einen globalen Wandel von der *Gemeinschaft* zur *Gesellschaft* voraus, der auf soziodemografischen Veränderungen beruht. Diese beiden Begriffe wurden 1887 von Tönnies eingeführt und zur Untersuchung des sozialen Wandels verwendet. Gemeinschaft" beschreibt verbindliche, primäre Interaktionsbeziehungen, die auf Gefühlen beruhen, und ist durch ein ländliches Umfeld, einfache persönliche Beziehungen, ein niedriges technologisches Niveau, begrenzte Bildung und eher durch Bedürftigkeit als durch Wohlstand gekennzeichnet. Gesellschaft hingegen beschreibt ein Interaktionssystem, das von Eigeninteresse, Wettbewerb und ausgehandelter Anpassung geprägt ist und sich durch ein städtisches Umfeld, eine modernisierte Gesellschaft mit einem hohen Maß an Technologie und Wohlstand auszeichnet" (Quelle: Younes und Reips, 2018, S.1; Christenson, 1984, S.160)

In den letzten Jahrhunderten hat die Verstädterung weltweit drastisch zugenommen. In dieser Fallstudie untersuchen wir, ob dieser Wandel mit einer Bewegung von der *Gemeinschaft* zum *Gesellschaft*ssystem einhergeht. Konkret werden wir der folgenden Frage nachgehen:

"Haben sich die Vereinigten Staaten in den letzten zwei Jahrhunderten von der Gemeinschaft zur Gesellschaft entwickelt und steht dies im Einklang mit dem Übergang von einer ländlichen zu einer städtischen Gesellschaft?"

Zur Beantwortung dieser Frage werden wir die Muster der soziokulturellen Veränderungen untersuchen, die sich in den Worthäufigkeiten der letzten zwei Jahrhunderte in den Vereinigten Staaten widerspiegeln. Der Google Ngram Viewer ist zu diesem Zweck besonders nützlich, da sich kulturelle Werte in Wörtern/Begriffen widerspiegeln, die in Schriften verwendet werden, und so können wir den langfristigen kulturellen Wertewandel durch die Untersuchung von Worthäufigkeiten in Büchern nachweisen.

Phase 2: Auswahl der Suchbegriffe

Der erste Schritt zur Beantwortung unserer Frage besteht darin, geeignete Suchbegriffe zu identifizieren, die mit den Begriffen Gemeinschaft und Gesellschaft verknüpft sind und den kulturellen Wertewandel in den Vereinigten Staaten abbilden können. In diesem speziellen

Beispiel sind folgende Kriterien für die Auswahl repräsentativer Suchbegriffe wichtig: a. eine hohe Häufigkeit des Begriffs und b. eine enge Bandbreite an semantischen Interpretationen.

Diskussion

Warum glauben Sie, dass die oben genannten Kriterien bei dieser Tätigkeit wichtig sind?

*

Die Auswahl von Begriffen mit hoher Häufigkeit ist wichtig, damit die Diagrammlinien tatsächlich kulturelle Veränderungen im Laufe der Zeit darstellen können und weil Wörter mit hoher Häufigkeit Merkmale der Kultur eines Landes sind. Ebenso wichtig ist in diesem Beispiel die Auswahl von Wörtern mit einer engen Bandbreite an semantischen Interpretationen. Denn Wörter mit breit gefächerten Bedeutungen können in verschiedenen Kontexten verwendet werden, die nicht immer für kulturelle Werte relevant sind. In der folgenden Tabelle sehen Sie Wörter, die die oben genannten Kriterien erfüllen und das System Gemeinschaft und Gesellschaft widerspiegeln.

Tabelle 1: Begriffe in Verbindung mit Gemeinschaft und Gesellschaft

Gemeinschaft	obliged, duty, give, benevolence, act, deed
Gesellschaft	choose, decision, get, acquisition, feel, emotion

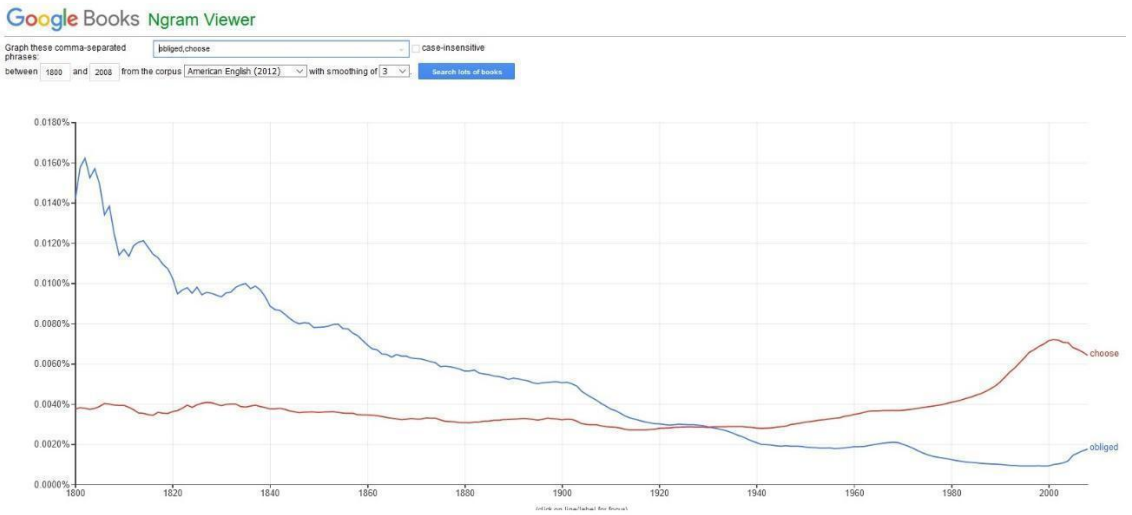
Diskussion

Wie können wir die obige Liste und den Google Ngram Viewer nutzen, um unsere Frage zu beantworten?

Wir haben besprochen, dass Google Ngram eingesetzt werden kann, um die Häufigkeit von Wörtern, die in der Literatur vorkommen, im Zeitverlauf darzustellen. Für unser Beispiel bedeutet dies, dass wir Begriffe verwenden und vergleichen können, die gegensätzliche Haltungen, Eigenschaften und Merkmale widerspiegeln, und beobachten können, wie ihre Häufigkeit im Laufe der Zeit schwankt.

Diskussion

Die folgende Abbildung zeigt die Häufigkeit der Wörter obliged(verbunden mit dem System Gemeinschaft) und choose (verbunden mit dem System Gesellschaft). Was fällt Ihnen an der Häufigkeit dieser beiden Begriffe im Laufe der Zeit auf?



Phase 3: Vergleich der Begriffe und Interpretation

In dieser Phase werden wir Google Ngrams verwenden, um weitere Begriffe im Zusammenhang mit dem System Gemeinschaft und Gesellschaft zu vergleichen und zu beobachten, wie sich ihre Häufigkeit in den letzten zwei Jahrhunderten verändert hat.

Gruppenaufgabe

Klicken Sie auf den folgenden Link, um auf Google Ngrams zuzugreifen: https://books.google.com/ngrams.

1. Benutzen Sie das Suchfeld (löschen Sie alle Suchbegriffe, die sich bereits im Feld befinden) und geben Sie die folgenden zwei durch ein Komma getrennten Wörter ein: Duty, Decision.
2. Wählen Sie aus dem Dropdown-Menü "from the corpus" American English (2012) und klicken Sie dann auf die Schaltfläche «search lots of books».
3. Was fällt Ihnen an der Häufigkeit dieser beiden Begriffe im Laufe der Zeit auf?
4. Untersuchen Sie auf die gleiche Weise die Häufigkeit der folgenden Wörter (wählen Sie mindestens zwei aus):
    a. Give and Get,
    b. Benevolence and Acquisition,
    c. Act and Feel,
    d. Deed and Emotion
5. Wie schwanken die Häufigkeiten dieser Wörter im Laufe der Zeit?

Bestätigen Ihre Beobachtungen den Übergang von der Gemeinschaft zur Gesellschaft in den letzten zwei Jahrhunderten?

*

Wie von der Theorie des sozialen Wandels und der menschlichen Entwicklung vorhergesagt, haben Begriffe wie obliged, duty, give, benevolence, act und deed, die allesamt ein ländliches und gemeinschaftliches Umfeld widerspiegeln, in den letzten zwei Jahrhunderten abgenommen. Im gleichen Zeitraum haben Begriffe wie choice, decision, get, acquisition, feel, und emotion, die allesamt ein städtisches und gesellschaftliches Umfeld widerspiegeln, zugenommen.

Verallgemeinerung (optional)

Dabei zeigte sich, dass in dem Maße, wie sich die Vereinigten Staaten in Richtung Gesellschaft bewegten, die kulturellen Merkmale der Gesellschaft (die sich in den relevanten Wörtern im Korpus der amerikanischen Bücher widerspiegeln) quantitativ zunahmen, während die kulturellen Merkmale der Gemeinschaft (die sich in den relevanten Wörtern im Korpus der amerikanischen Bücher widerspiegeln) quantitativ abnahmen. Lassen sich diese Beziehungen und Trends auf andere Teile der Welt verallgemeinern, wie es die Theorie des sozialen Wandels und der menschlichen Entwicklung vorhersagen würde?

Gruppenaufgabe

Wiederholen Sie die in Phase 3 beschriebenen Schritte; wählen Sie diesmal die Literatur Ihrer Wahl (z. B. britische oder deutsche), um die gleiche Frage für das Land Ihrer Wahl zu beantworten.

Hinweis: Für die deutsche Literatur können Sie die folgenden Begriffe verwenden (Quelle: Younes & Reips, 2018):

1. versprechen und auswählen
2. Pflicht und Entscheidung
3. geben und bekommen
4. Güte und Kauf
5. handeln und spüren
6. Handlung und Emotion

Lernressourcen

Von der Gemeinschaft zur Gesellschaft.pdf (optional)

B Die Psychologie der Kultur im Wandel von 1800 bis 2000 (optional)

C Die Psychologie der Kultur in den deutschsprachigen Ländern im Wandel (optional)

Referenzen

Christenson, J.A., (1984). Gemeinschaft and gesellschaft: Testing the spatial and communal hypotheses. *Social Forces*, 63(1), pp.160-168.

Greenfield, P.M., (2013). The changing psychology of culture from 1800 through 2000. *Psychological Science*, 24(9), pp.1722-1731.

Younes, N. and Reips, U.D., (2018). The changing psychology of culture in German‐speaking countries: A Google Ngram study. *International Journal of Psychology*, 53, pp.53-62.

Aktivität 2.2 Modellierung und Simulation mit NetLogo

Gesamtzeit: 2.30 - 3 Stunden

NetLogo ist eine programmierbare Multi-Agenten-Modellierungsumgebung zur Simulation natürlicher und sozialer Phänomene und zur Darstellung ihrer zeitlichen Entwicklung. Mit diesem Tool können Sie eine Welt aus rechteckigen Feldern erstellen und Agenten (die Schildkröten aus der Logowelt) parametrisieren, die sich bewegen und miteinander und mit ihrer Umgebung interagieren. In dieser Aktivität werden wir NetLogo für eine Fallstudie über Epidemiologie verwenden und Modellsimulationen durchführen, um zu untersuchen, wie sich große und kleine Veränderungen auf eine Umgebung auswirken können.

Einführende Diskussion

Bevor Sie mit der Arbeit an dieser Aktivität beginnen, klicken Sie auf den Link http://www.netlogoweb.org/launch#http://www.netlogoweb.org/assets/modelslib/Sample%20Models/Art/Fireworks.nlogo, um das Tool aufzurufen. Wählen Sie aus dem Dropdown-Menü "Search the Models Library" die Option sample models/Biology/Flocking. Das Beispiel, das geladen wird, ist in der folgenden Abbildung dargestellt.

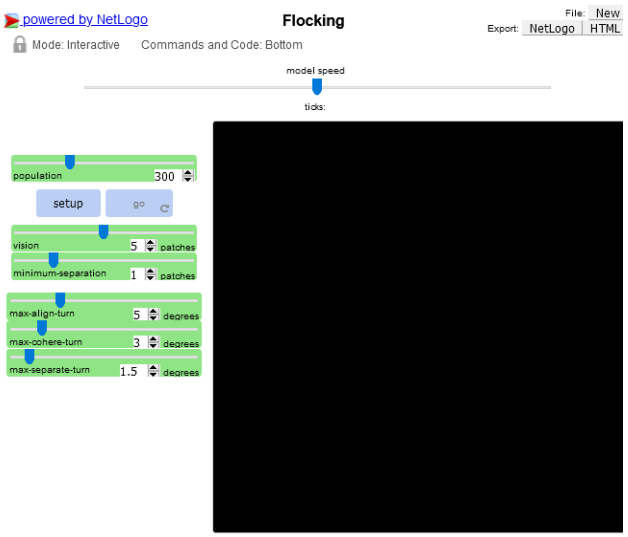


Abbildung 1 Beispiel Flocking in NetLogo

Dieses Modell ist ein Versuch, das Schwärmen von Vögeln zu imitieren. Jeder Vogel folgt genau denselben Regeln: "alignment", "separation" und "cohesion". "alignment", gesteuert durch den Schieberegler “max-align-turn”, bedeutet, dass ein Vogel dazu neigt, sich so zu drehen, dass er sich in dieselbe Richtung bewegt wie die Vögel in der Nähe. "separation", gesteuert durch den Schieberegler “max-separate-turn”, bedeutet, dass ein Vogel sich dreht, um einem anderen Vogel auszuweichen, der ihm zu nahe kommt. "cohesion", gesteuert durch den Schieberegler “max-cohere-turn”, bedeutet, dass ein Vogel sich auf andere Vögel in der Nähe zubewegt (es sei denn, ein anderer Vogel ist zu nah). Wenn zwei Vögel zu nahe beieinander sind, hat die "separation"-Regel Vorrang vor den beiden anderen, die deaktiviert werden, bis der Mindestabstand erreicht ist. Der Schieberegler “vision” ist die Entfernung, die jeder Vogel 360 Grad um sich herum sehen kann.

Die grünen Schieberegler (z.B. max-fireworks) auf der linken Seite dienen der Parametrisierung Ihres Modells. Mit der Schaltfläche SETUP wird das Modell entsprechend den durch die Schieberegler angegebenen Werten eingestellt. Die Schaltfläche GO führt das Modell aus.

Paaraufgabe

Bestimmen Sie zunächst die Anzahl der Vögel in der Simulation und stellen Sie den Schieberegler “population” auf diesen Wert ein. Drücken Sie zuerst SETUP und dann GO, um die Simulation zu starten. Die Standardeinstellungen für die Schieberegler ergeben ein recht gutes Schwarmverhalten. Sie können jedoch mit ihnen spielen, um Variationen zu erhalten. Passen Sie die Schieberegler an, um zu sehen, wie Sie dichtere, lockerere, weniger oder mehr Schwärme erhalten können.

Phase 1: Problemstellung

Die Frage, der wir in dieser Aktivität nachgehen werden, bezieht sich auf die Epidemiologie und insbesondere auf das Zika-Virus.

Das Zika-Virus ist ein durch Mücken übertragenes Flavivirus, das erstmals 1947 in Uganda bei Affen nachgewiesen wurde. Die Inkubationszeit (die Zeit von der Exposition bis zum Auftreten von Symptomen) der Viruserkrankung wird auf 3-14 Tage geschätzt. Die Mehrheit der mit dem Zika-Virus infizierten Personen entwickelt keine Symptome. Die Symptome sind in der Regel mild und umfassen Fieber, Hautausschlag, Bindehautentzündung, Muskel- und Gelenkschmerzen, Unwohlsein und Kopfschmerzen und dauern in der Regel 2-7 Tage an. Das Virus wird hauptsächlich durch den Stich einer infizierten Mücke der Gattung *Aedes*, hauptsächlich *Aedes aegypti*, in tropischen und subtropischen Regionen übertragen. Aedes-Mücken stechen in der Regel tagsüber, am häufigsten am frühen Morgen und am späten Nachmittag/Abend. Es handelt sich um dieselbe Mücke, die auch Dengue, Chikungunya und Gelbfieber überträgt. Das Virus wird auch von der Mutter auf den Fötus während der Schwangerschaft, durch sexuellen Kontakt, Transfusionen von Blut und Blutprodukten und Organtransplantationen übertragen (Quelle: Weltgesundheitsorganisation[1]).

In dieser Fallstudie werden wir an einem hypothetischen epidemiologischen Szenario arbeiten und versuchen, das Risiko der Virusausbreitung zu verstehen und zu bewerten. Das Szenario, an dem wir arbeiten werden, ist das folgende:

In einer kleinen Stadt in Südafrika wurden zwei Personen positiv auf das Zika-Virus getestet. Die Regierung möchte vorhersagen, wie ernst die Situation ist, und daher die notwendigen Präventivmaßnahmen ergreifen. Die Gesamtbevölkerung dieser Stadt beträgt 800 Einwohner, und nach einer ersten Schätzung liegt die Zahl der Mücken in dem Gebiet bei 1040, während die Zahl der Räuber (Organismen, die sich von Mücken ernähren) in dem Gebiet 21 beträgt.

Die Frage, die wir beantworten müssen, lautet wie folgt:

Wie viele Menschen werden infiziert werden, und was wäre die beste Strategie zur Verhinderung des Virus?

Um diese Frage zu beantworten, werden wir NetLogo verwenden und Simulationen an einem Modell zum Zika-Virus durchführen.

Phase 2: Simulation

In dieser Phase werden wir mit einem bereits implementierten Modell arbeiten, das uns helfen wird zu verstehen, wie Modellierung und Simulation eingesetzt werden können, um neue Erkenntnisse und Wissen über ein Phänomen zu gewinnen und wie sie die Entscheidungsfindung beeinflussen können.

Klicken Sie auf den folgenden Link, um auf NetLogo zuzugreifen:

[1] https://www.who.int/news-room/fact-sheets/detail/zika-virus

http://www.netlogoweb.org/launch#http://www.netlogoweb.org/assets/modelslib/Sample%20Models/Art/Fireworks.nlogo

Klicken Sie auf "browse", um das Modell zikavirusmodel.nlogo hochzuladen. Sobald das Modell erfolgreich geladen wurde, sollte das folgende Beispiel zu sehen sein.

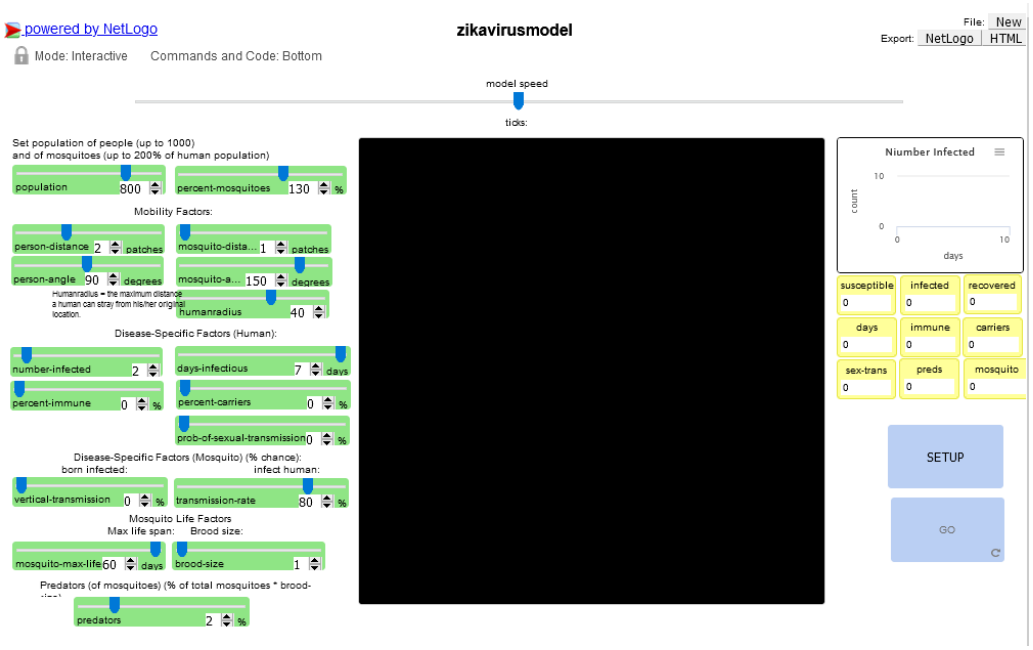


Abbildung 2 Zika-Virus Modell in NetLogo

In dem Modell würden Moskitos Zika-infizierte Menschen stechen und dann die Krankheit weiter verbreiten, indem sie andere, nicht-infizierte Menschen stechen. Es gibt viele Variablen, die Sie in diesem Modell manipulieren können. Für diese Aktivität werden wir uns nur auf die folgenden konzentrieren:

- *person-dist*ance, die Mobilität der Menschen - wie weit sie sich bewegen können; wie viele Felder (patches) sie an einem Tag zurücklegen können
- *number-infected*, die Anzahl der ursprünglich infizierten Personen
- *days-infectious*, die Dauer, die eine typische Person infiziert ist
- *percent-immune,* der Prozentsatz der Bevölkerung, der immun ist (= geimpft)
- *predators*, die Anzahl der Räuber
- *percent-mosquitoes*, die Anzahl der Moskitos

Bei jedem Durchlauf des Modells müssen Sie die folgenden drei Dinge nacheinander tun: erstens die Schieberegler einstellen (falls Sie sie anpassen wollen), zweitens auf SETUP klicken (in der unteren rechten Ecke der Anzeige) und drittens auf GO klicken, um das Modell auszuführen.

Das Diagramm in der oberen rechten Ecke und die Monitore direkt darunter zeigen Ihnen, wie sich die Population unter Ihrem Schiebereglerregime in Bezug auf das Virus entwickelt.

Paaraufgabe

Lassen Sie den Simulator mit den Standardwerten laufen, die der in Phase 1 dargestellten Problemstellung entsprechen. Sobald die Simulation beendet ist, beantworten Sie die folgende Frage: Wie viele Personen wurden insgesamt infiziert und in wie vielen Tagen?

*

Wenn Sie den Simulator ausführen, erhalten Sie eine Schätzung, wie viele Personen unter bestimmten Umständen mit dem Virus infiziert werden. In diesem Beispiel liegt diese Zahl irgendwo zwischen 740 und 775. Mit unserem Modell konnten wir also einen Teil unserer Frage nach der Anzahl der Menschen, die sich mit dem Zika-Virus infizieren würden, beantworten. Was wir noch nicht beantwortet haben, ist die Frage, welche Strategie am besten geeignet wäre, um die Ausbreitung zu verhindern. Dies ist der Schwerpunkt der nächsten Phase.

Phase 3: Weitere Untersuchung - Interpretation

In diesem Modell gibt es mehrere Variablen, die eingestellt werden können, um zu untersuchen, wie sich die Krankheit unter verschiedenen Bedingungen in der Bevölkerung ausbreiten würde. Um die beste Strategie zur Verhinderung der Ausbreitung zu ermitteln, werden wir drei Variablen berücksichtigen: Anzahl Räuber, Personen-Distanz und Prozentsatz der Immunität.

Gruppenaufgabe

In dieser Aufgabe werden wir untersuchen, wie sich die folgenden Veränderungen auf die Ausbreitung des Virus auswirken und welche davon effektiv sind.

1. Erhöhen Sie den Wert der Räuber auf 6 % und klicken Sie dann auf SETUP und dann auf GO. Notieren Sie sich die Anzahl der genesenen Tiere aus den Monitoren in der oberen rechten Ecke. Bevor Sie zu Punkt 2 weitergehen, stellen Sie den Schieberegler für die Räuber wieder auf 2% (Standardwert).
2. Erhöhen Sie den Wert für immun auf 6%, klicken Sie auf SETUP und dann auf GO. Notieren Sie sich die Anzahl der Genesenen in den Monitoren in der oberen rechten Ecke. Bevor Sie zu Punkt 3 weitergehen, stellen Sie den Schieberegler für die Immunen wieder auf 0 (Standardwert).
3. Verringern Sie die Personenbewegung auf 1 patch und klicken Sie dann auf SETUP und dann auf GO. Notieren Sie sich die Anzahl der genesenen Personen auf den Monitoren in der oberen rechten Ecke.

*

Sie werden feststellen, dass die Zahl der infizierten Personen zwischen 400 und 680 liegt, wenn Sie die Personenbewegung auf 1 patch verringern. Erhöht man die Zahl der Räuber auf 6 %, dann liegt die Zahl der Infizierten zwischen 140 und 300, während bei einer Erhöhung des Prozentsatzes der Immunen auf 6 % die Zahl der Infizierten zwischen 630 und 740 liegt. Unter den derzeitigen Bedingungen, die unser Modell berücksichtigt, ist die beste Strategie zur Verhinderung der Ausbreitung auf die gesamte Bevölkerung eine Erhöhung der Räuber auf 6 %.

Phase 4: Abschließende Frage

In dieser Aktivität arbeiteten wir an einem hypothetischen Szenario in der Epidemiologie und untersuchten mit Hilfe von Modellierung und Simulation die Dynamik der Ausbreitung eines Virus, indem wir Variablen manipulierten und die Ergebnisse auswerteten.

Diskussion

Wie können Modellierung und Simulation die Entscheidungsfindung in anderen Disziplinen als der Epidemiologie beeinflussen?

Lernressourcen

zikavirusmodel.nlogo

Aktivität 2.3 Microsoft Excel verwenden, um Daten zu bearbeiten und darzustellen

Gesamtzeit: 2:30 - 3 Stunden

Excel ist ein Tabellenkalkulationsprogramm zum Organisieren und Berechnen von Daten sowie zum Analysieren und Darstellen von Daten in Form eines Diagramms oder einer Grafik. In dieser Aktivität werden wir Excel für die Analyse und Darstellung von Daten verwenden, Trends in den Diagrammen erkennen und Informationen miteinander in Beziehung setzen, um eine Frage über Klimaunterschiede in Biomen und deren Auswirkungen auf ihre Oberflächenfarben zu beantworten.

Warm up - Diskussion

Bevor Sie mit dieser Aktivität beginnen, öffnen Sie die Arbeitsmappe Earth colour workbook und navigieren Sie zum Arbeitsblatt Mapping biomes, um ein Bild der Erdoberfläche zu sehen, das aus Satellitenbildern erstellt wurde (Abbildung 1).

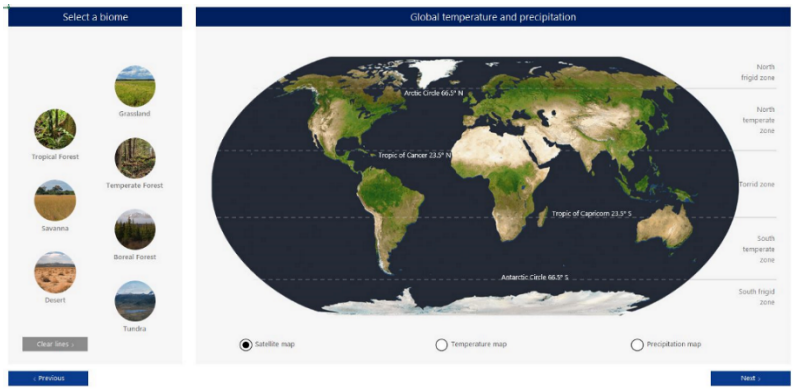


Abbildung 1: Arbeitsblatt Mapping biomes

Paaraufgabe

Klicken Sie auf die Biome-Schaltflächen auf der linken Seite des Satellitenbildes der Erde, um die Standorte der verschiedenen Biome zu markieren.

- Welche Farbe beschreibt Ihrer Meinung nach die einzelnen Biome am besten?
- Wie erklären sich die unterschiedlichen Farben in den verschiedenen Biomen?

Tipp: Wählen Sie die Schaltfläche Temperatur- und Niederschlagskarte (Temperature bzw. Precipitation) unter dem Satellitenbild, um Verbindungen zwischen den Farben und den physikalischen Bedingungen herzustellen.

*

Wie Sie feststellen können, sind Regionen mit viel Niederschlag grüner, während höhere Temperaturen und geringere Niederschläge zu Regionen führen, die brauner erscheinen können. Die Pole haben sehr niedrige Durchschnittstemperaturen, was die weiße Farbe von Schnee und Eis erklärt.

Phase 1: Problemstellung

In dieser Aktivität werden wir Daten nutzen, um besser zu verstehen, warum Farbveränderungen in Biomen auftreten. Wir werden die sieben terrestrischen Biome und ihre charakteristischen Farben aus dem Weltraum untersuchen, die von Faktoren wie Temperatur und Niederschlag abhängen. Die Frage, mit der wir uns beschäftigen werden, ist die folgende:

Wie verändern sich die Temperatur und der Niederschlag in den verschiedenen Biomen im Laufe des Jahres und wie spiegelt sich diese Veränderung in den Farben wider?

Phase 2: Graphische Darstellung von Temperatur und Niederschlag

In der Aufwärmübung haben Sie sich die durchschnittlichen globalen Temperatur- und Niederschlagsdaten angeschaut und begonnen, Verbindungen zwischen den Farben der Erdoberflächen, die im Satellitenbild dargestellt sind, und den durchschnittlichen jährlichen Niederschlägen und Temperaturen herzustellen. In dieser Phase werden wir die monatlichen Veränderungen von Niederschlag und Temperatur im Jahresverlauf in den Biomen untersuchen.

Navigieren Sie für die folgende Aufgabe zum Arbeitsblatt "Erstellen einer Biome-Klimakarte", das in der folgenden Abbildung dargestellt ist.

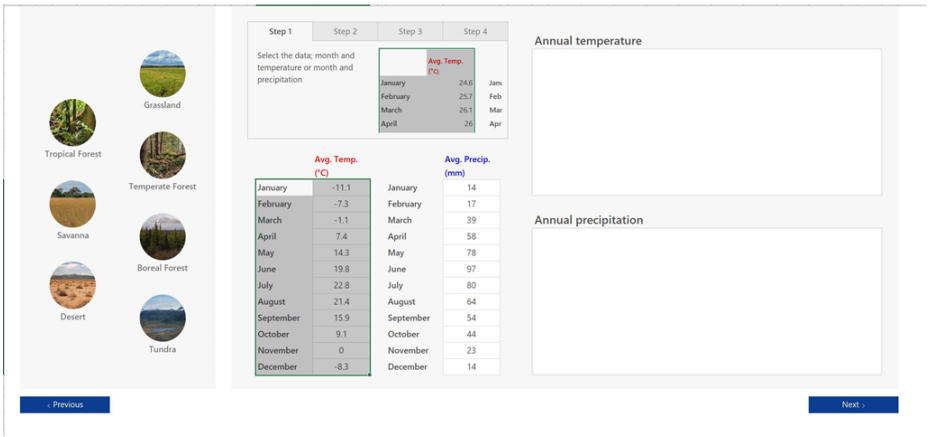


Abbildung 2: Arbeitsblatt zur Erstellung eines Biome-Klimadiagramms

Paaraufgabe

Wählen Sie für diese Aufgabe ein Biom aus, das Sie näher untersuchen möchten. Wenn Sie auf die Schaltflächen des Bioms auf der linken Seite klicken, sehen Sie die monatlichen Temperatur- und Niederschlagsdaten unter den Bezeichnungen "Avg. Temp." und "Avg. Precip." angezeigt.

1. Wählen Sie die monatlichen Temperaturdaten aus und klicken Sie dann auf die Option Einfügen in der Menüleiste von Excel. Klicken Sie auf die Option Empfohlene Diagramme und wählen Sie dann ein Diagramm aus, das Sie für die Darstellung der monatlichen Temperaturdaten des Bioms verwenden möchten.
2. Wiederholen Sie die gleichen Schritte für die Niederschlagsdaten.

Wie schwanken die Temperatur- und Niederschlagsdaten für Ihr Biom im Laufe des Jahres? Beobachten Sie, welche(r) Monat(e) die höchste(n) und niedrigste(n) Temperatur und Niederschlag aufweisen.

Wählen Sie ein anderes Biom und wiederholen Sie diese Aufgabe, indem Sie ein Diagramm erstellen, das die monatliche Temperatur und den Niederschlag darstellt. Vergleichen Sie die beiden Biome, indem Sie beobachten, welches Biom im Laufe des Jahres mehr oder weniger Schwankungen bei Temperatur und Niederschlag aufweist.

Optionale Aufgabe

Wiederholen Sie die obige Aufgabe für alle Biome und wählen Sie verschiedene Arten von Diagrammen aus. Beobachten Sie, welche Diagramme für die Darstellung von Temperatur- und Niederschlagsveränderungen und für die Erkennung von Trends, Unterschieden und Ähnlichkeiten zwischen zwei oder mehreren Biomen am nützlichsten sind.

*

An dieser Stelle haben wir den ersten Teil unserer Frage nach der Art und Weise, wie sich Temperatur und Niederschlag im Laufe eines Jahres für ein Biom verändern, beantwortet. Indem wir die Temperatur- und Niederschlagsdaten für ein Biom grafisch dargestellt haben, konnten wir diese Schwankungen beobachten und auch Biome miteinander vergleichen. Was wir jedoch noch weiter untersuchen müssen, ist, wie sich diese Veränderungen der Temperatur- und Niederschlagsdaten in den Farben widerspiegeln.

Phase 3: Vergleich von Farben mit Temperatur und Niederschlag in Biomen

In der vorangegangenen Phase haben wir anhand von Diagrammen die monatlichen Veränderungen der Temperatur- und Niederschlagsdaten für zwei Biome dargestellt und verglichen. In dieser Phase werden wir Biomeigenschaften, Temperatur- und Niederschlagsdaten miteinander in Beziehung setzen und untersuchen, wie sich Veränderungen in der saisonalen Vegetation in der auffälligsten Farbe der Landschaft eines Bioms in einem bestimmten Monat des Jahres widerspiegeln.

Paaraufgabe

Navigieren Sie zum Arbeitsblatt "Informationen über Biome", das in der folgenden Abbildung dargestellt ist. Wählen Sie erneut die Biome aus, die Sie in der vorherigen Phase ausgewählt haben, und lesen Sie die Informationen zu ihrer Vegetation. Was fällt Ihnen in Bezug auf Temperatur, Niederschlag und Vegetation eines Bioms auf?

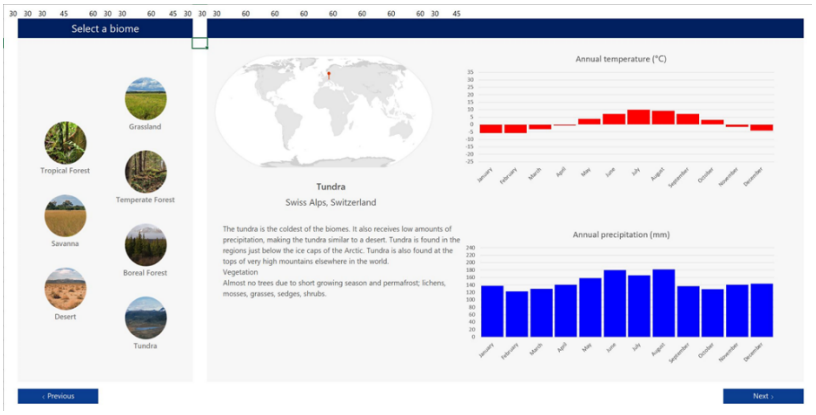


Abbildung 3: Arbeitsblatt "Informationen über Biome"

Wenn Sie die Temperatur, den Niederschlag und die Vegetation eines Bioms vergleichen, werden Sie feststellen, dass die Vegetation einer bestimmten Region speziell an die Klimabedingungen der Biomregion angepasst ist. Sie werden auch feststellen, dass, selbst wenn die Temperaturen in einer Region ähnlich sind, die Unterschiede im Niederschlag einen großen Einfluss auf die Art der Vegetation in einem Biom und damit auf die Farben seiner Oberfläche haben.

Navigieren Sie für die folgende Frage zum Arbeitsblatt "*Biome-Klimadaten vergleichen*", das in der folgenden Abbildung dargestellt ist.

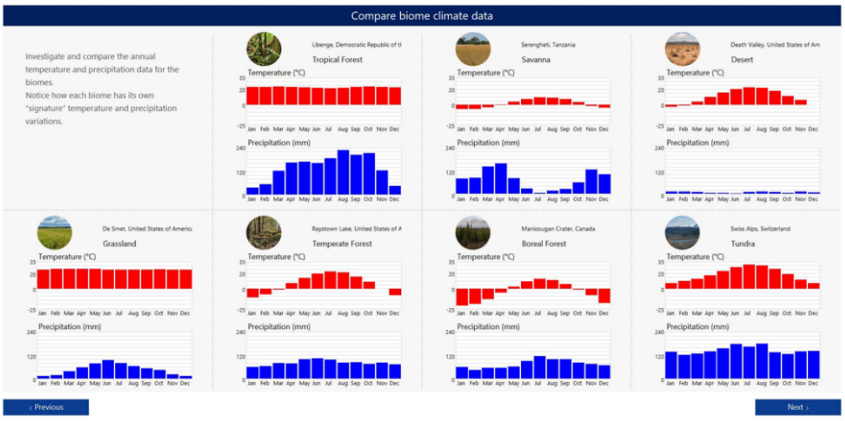


Abbildung 4: Arbeitsblatt Biome-Klimadaten vergleichen

Paaraufgabe

Basierend auf den Temperatur- und Niederschlagsdaten,

- Welche Biomregionen werden Ihrer Meinung nach die gleichmäßigsten jährlichen Oberflächenfarben aufweisen?
- Welche der Oberflächenfarben wird Ihrer Meinung nach im Laufe eines Kalenderjahres am stärksten schwanken?

Erläutern Sie Ihre Überlegungen.

*

Die Farbe kann zur Beschreibung der Veränderungen eines Bioms im Laufe des Jahres verwendet werden; die sich verändernde Vegetation eines Bioms ist für seine Oberflächenfarbe verantwortlich und spiegelt sich in der auffälligsten Farbe wider, die in der Landschaft des Bioms in einem bestimmten Monat des Jahres zu sehen ist.

Navigieren Sie für die folgende Frage zum Arbeitsblatt "Biomfarben im Jahresverlauf", das in der folgenden Abbildung dargestellt ist.

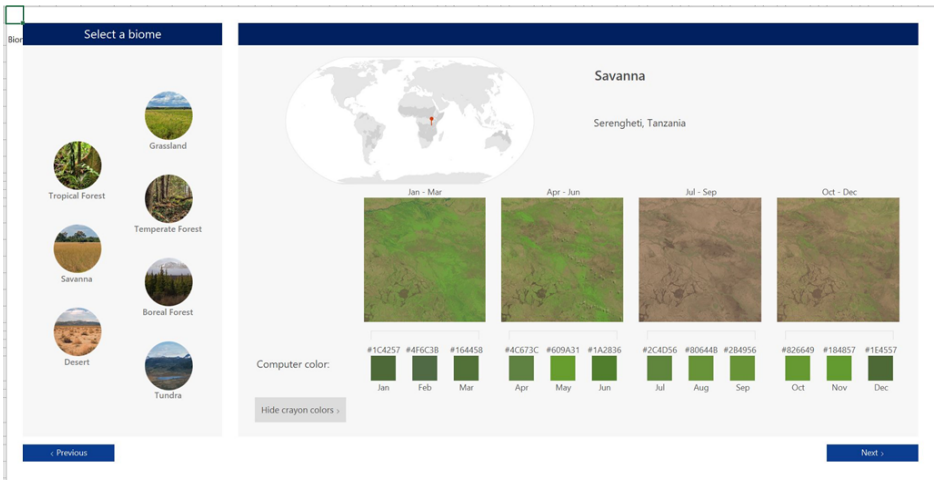


Abbildung 5: Arbeitsblatt Biomfarben im Jahresverlauf

Paaraufgabe

Basierend auf den Temperatur- und Niederschlagsdaten,

- Welche Biomregionen werden Ihrer Meinung nach die gleichmäßigsten jährlichen Oberflächenfarben aufweisen?
- Welche der Oberflächenfarben wird Ihrer Meinung nach im Laufe eines Kalenderjahres am stärksten schwanken?

Erläutern Sie Ihre Überlegungen.

*

Navigieren Sie für die folgende Frage zum Arbeitsblatt "Biom-Farbdaten vergleichen", das in der folgenden Abbildung dargestellt ist.

Abbildung 6: Arbeitsblatt Biom-Farbdaten vergleichen

Paaraufgabe

Wie hängen die monatlichen Signaturfarben für die verschiedenen Biomregionen mit Temperatur und Niederschlag zusammen? Nennen Sie Beispiele aus den Daten.

Tipp: Beachten Sie, dass die unterschiedlichen Farben in den verschiedenen Biomen mit dem Niederschlag und der Temperatur zusammenhängen.

*

Vielleicht haben Sie schon bemerkt, dass die blaugrüne Farbe für Gebiete mit Schnee steht, während Gebiete mit viel Niederschlag grüner sind und höhere Temperaturen und weniger Niederschlag Regionen entsprechen, die brauner erscheinen können.

Bei der Beantwortung des zweiten Teils unserer Frage - *wie sich Niederschlags- und Temperaturveränderungen in den Farben widerspiegeln* - haben wir festgestellt, dass sich die durch Temperatur- und Niederschlagsveränderungen bedingte Veränderung der Vegetation eines Bioms in der Farbe widerspiegelt, die in einem bestimmten Monat des Jahres in der Landschaft des Bioms am stärksten ausgeprägt ist, und somit die Oberflächenfarbe des Bioms bestimmt.

60 Phase 4: Vorhersage des Klimawandels - Erweiterte Aktivität (optional)

Die Aufgaben und Fragen in den vorangegangenen Phasen haben Sie dazu angeleitet, die Frage zu beantworten, wie sich Temperatur und Niederschlag in verschiedenen Biomen im Laufe des Jahres verändern und wie sich diese Veränderung in den Farben widerspiegelt. In Phase 4 werden Sie untersuchen, wie sich der Klimawandel auf eine Biome-Region Ihrer Wahl auswirkt. Zu diesem Zweck sammeln und analysieren Sie Niederschlags- und Temperaturdaten für dieses Biom und untersuchen, wie sich seine charakteristischen Farben verändern könnten, wenn der Klimawandel nicht bekämpft wird.

Gruppenaufgabe

Recherchieren Sie ein Problem des Klimawandels, das sich auf eine Biome-Region Ihrer Wahl auswirkt.

1. Sammeln Sie monatliche Temperatur- und Niederschlagsdaten der letzten 10 Jahre, um zu sehen, ob es irgendwelche Trends oder interessante Muster gibt. Halten Sie die Daten in Ihrer Excel-Arbeitsgruppe in einem geeigneten Format fest.
2. Erstellen Sie Temperatur- und Niederschlagsdiagramme für die Biome-Region, um Trends zu erkennen.
3. Überlegen Sie, was der Klimawandel für Ihre Biome-Region bedeutet und wie er sich in den von Ihnen gesammelten Daten widerspiegelt. Diskutieren Sie, wie sich die Farben in dieser Biome-Region infolge des Klimawandels verändert haben könnten, und machen Sie Vorhersagen darüber, wie sich die Farben in dieser Region in den nächsten 10 Jahren verändern könnten, wenn der Klimawandel nicht gestoppt wird.

Lernressourcen

Arbeitsbücher: Arbeitsbuch Farbe der Erde.xlsx

UNIT 3: Computational Thinking durch Programmierung erleben

Gesamtzeit: 10 Stunden

In dieser Unit werden Sie mit den Grundlagen der Programmierung vertraut gemacht und üben algorithmisches Denken im Kontext von Geschichten und Spielen. Sie werden auch lernen, Flussdiagramme zur Darstellung von Algorithmen zu verwenden.

Beitrag zu den Lernergebnissen

Lernergebnisse: Studierende können am Ende der Unit 3
• Ideen für Geschichten auf standardisierte Art und Weise entwickeln und aufschreiben • die wichtigsten Unterschiede bei der Entwicklung elektronischer interaktiver Geschichten und traditioneller Bücher auf Papier erkennen • den Algorithmus für ein bestehendes Programm identifizieren und erklären • Änderungen an einem bestehenden Programm vornehmen • Fehlersuche und Korrektur eines bestehenden Programms vornehmen • ein neues Programm zur Produktion einer interaktiven Geschichte planen und entwerfen • eine interaktive Geschichte mit Hilfe von programmierbaren Elementen erstellen und entwickeln

- Schleifen, Variablen, Broadcast-Nachrichten, IF-Anweisungen und sequentielle Anweisungen in einem Programm verwenden
- die Bedeutung von korrekten Anweisungen verstehen
- verstehen, was ein Algorithmus ist
- einen Algorithmus in einem Flussdiagramm darstellen
- die Qualität eines Algorithmus bewerten
- einen zuvor geschriebenen Algorithmus oder Flussdiagramm in Scratch implementieren
- erklären, was mit dem Begriff "Variable" gemeint ist
- Variablen in Ihrem Programm erstellen und verwenden
- erklären, was mit den Begriffen "Auswahl" und "Schleife" gemeint ist
- Auswahl- und Schleifenanweisungen innerhalb eines Algorithmus oder Programms verwenden

Aktivität 3.1: Geschichtenerzählen mit Scratch

Beim digitalen Geschichtenerzählen werden digitale Medien (Bilder, Sprache, Musik, Bewegung) verwendet, um eine Geschichte zu erzählen. In den letzten Jahren hat sich das digitale Geschichtenerzählen zu einer zunehmend beliebten und effektiven Lernaktivität entwickelt, um eine Reihe von Lernzielen zu erreichen, insbesondere die Lernziele im Zusammenhang mit Computational Thinking.

In dieser Unit üben Sie das digitale Geschichtenerzählen in Scratch. Scratch ist eine kostenlose, visuelle und blockbasierte Programmierumgebung. In Scratch können die SchülerInnen ihre Ideen in Form von Projekten mit programmierbaren Medien wie Videos, Bildern, Spielen und Animationen entwickeln.

Paaraufgabe

Sehen Sie sich das Video "Einführung in Scratch" an.

Sehen Sie sich dieses Beispielprojekt an, das von Achtklässlern erstellt wurde, die sich mit dem Periodensystem aus der Chemie beschäftigen.

*

Mit Scratch

- formulieren sie eine Aufgabenstellung (Problem) und finden heraus, wie Sie die Elemente (Blöcke) von Scratch nutzen können, um Ihre Geschichte zu konstruieren: Handlung, Schauplatz, Abfolge und Perspektive.
- organisieren und analysieren Sie Daten, durch Erstellen von Codeblöcken, um Figuren und ihre Umgebung zu animieren.
- repräsentieren Sie den Inhalt von Geschichten durch die Bewegung von Figuren (Sprites).
- schulen Sie algorithmisches Denken, indem Sie Code entwickeln, der Figuren bewegt und kommunizieren lässt.

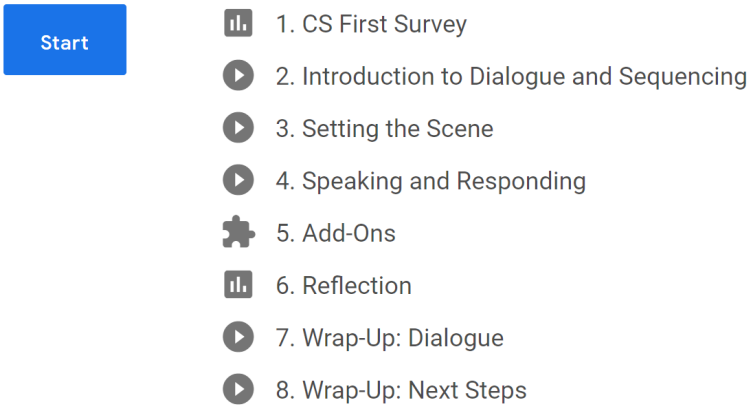
Vorbereitung: Bereiten Sie alles vor

Wir gehen davon aus, dass Ihr Dozent Ihnen die CS First-Benutzernamen und -Passwörter mitgeteilt hat. Bevor Sie loslegen können, müssen Sie Folgendes tun:

- Öffnen Sie ein neues Fenster in Ihrem Browser und gehen Sie zu g.co/CSFirst
- Klicken Sie oben rechts auf "Sign in"
- Klicken Sie auf "I am a student"
- Klicken Sie auf "Sign in with CS First"
- Klicken Sie auf "Enter class code"
- Klassencode 3h24s3 eingeben
- Geben Sie Ihren Benutzernamen und Ihr Passwort ein

Dialog

Lernen Sie etwas über CS First und schreiben Sie dann eine Geschichte, in der sich zwei Personen unterhalten, ohne Fragen zu stellen.



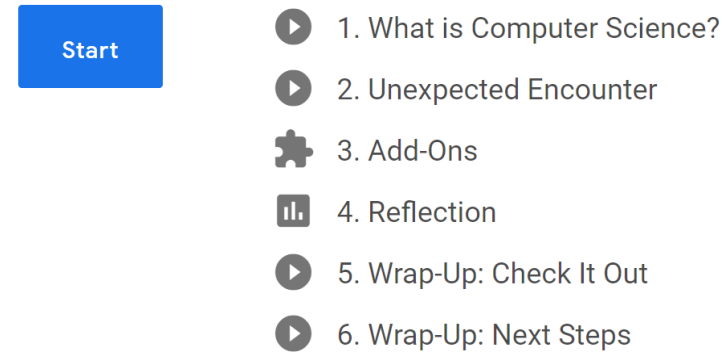
Paaraufgabe

- Drücken Sie die Start-Taste.
- Überspringen Sie die Umfrage und wählen Sie 2: Introduction to Dialogue and Sequencing
- Fügen Sie die folgende Registerkarte zu Ihrem Browserfenster hinzu: https://scratch.mit.edu/
- Wenn Sie noch kein Scratch-Konto haben, registrieren Sie sich.
    - Tipp: Sie können schnell zwischen den Registerkarten wechseln, indem Sie Strg-Tab drücken.
- Starten Sie das Video bei 2:30 und folgen Sie den Anweisungen am Ende.
- next drücken
- Schauplatz der Handlung (3)
    - Zu diesem Zeitpunkt sollten Sie ein neues Scratch-Projekt erstellt haben, zum Beispiel mit dem Namen myFirstStory
- Starten Sie das Video und folgen Sie den Anweisungen am Ende: Sie müssen das Video nicht zwischendurch unterbrechen.
- Weiter drücken
    - Sprechen und Reagieren (4)
- Starten Sie das Video und folgen Sie den Anweisungen am Ende.
- Weiter drücken
    - Zusätze (5)

- Wählen Sie unten auf der Seite "Bewegung hinzufügen".
    - Tipp: Sie müssen die Koordinaten nicht selbst eingeben, wenn Sie die Figur zuerst an der gewünschten Stelle platzieren, bevor Sie den entsprechenden Bewegungs-Block auswählen.
- Wählen Sie "Eine dritte Figur hinzufügen".

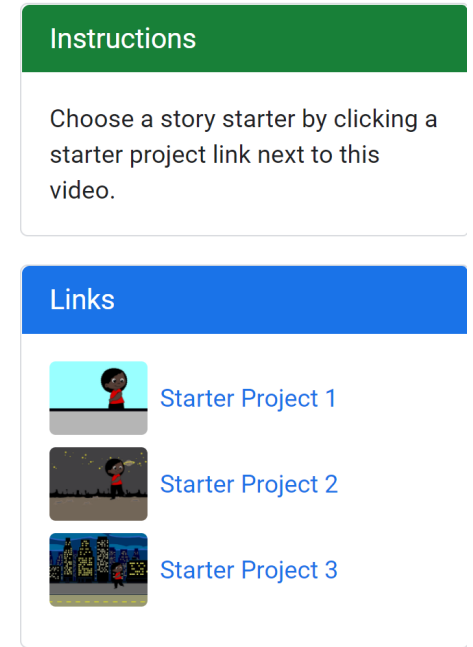
Überprüfen Sie es

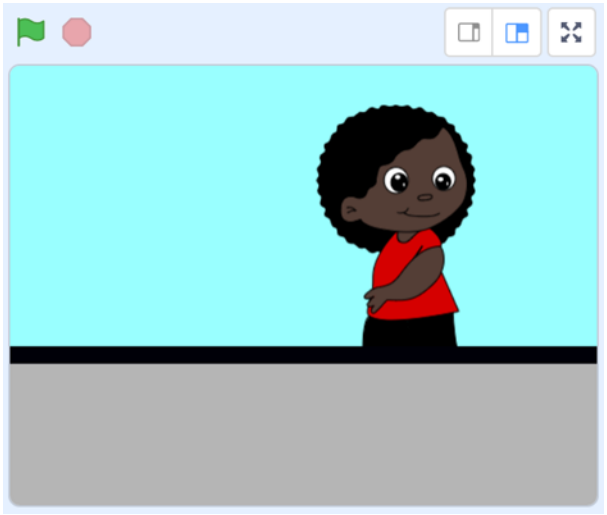
Erzählen Sie eine Geschichte, in der eine Figur durch eine Szene geht und beschreibt, was sie sieht.



Paaraufgabe

- Drücken Sie die Taste Start.
    - Sie müssen sich das erste Video nicht ansehen, sondern sollten stattdessen auf den Link mit Starter Project 1 klicken. Dadurch wird Scratch mit einem Starterprojekt geöffnet, das bereits etwas Code enthält.





- Weiter drücken
    - Unerwartete Begegnungen (2)
- Folgen Sie den Anweisungen im Video
- Weiter drücken
    - Zusätze (3)
- Wählen Sie "Ton hinzufügen" und folgen Sie den Anweisungen im Video.

Einstellung

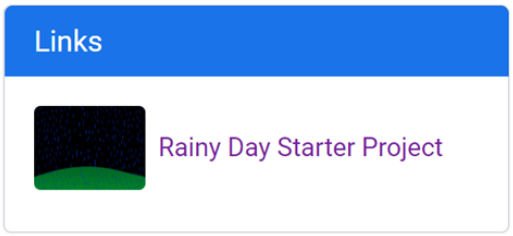
Start

Erstellen Sie eine dynamische Gewitterkulisse mit Regen und Blitzen.

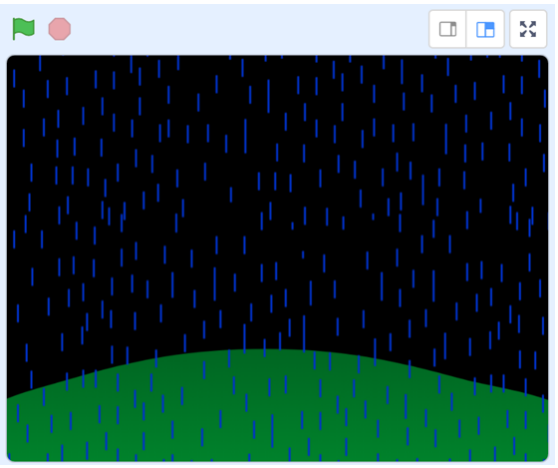
1. Introduction to Setting and Randomness
2. Make it Rain
3. Lightning Flash
4. Random Lightning
5. Making Your “Stormy Day” Setting into a Story
6. Add-Ons
7. Reflection
8. Wrap-Up: Setting
9. Wrap-Up: Next Steps

Paaraufgabe

- Drücken Sie die Taste Start.
- Öffnen Sie das Rainy Day Starter Projekt



- Gehe zu
  - Einführung in das Setting und die Zufälligkeit (1)
- Das Video ansehen
- Weiter drücken
  - Lass es regnen (2)
- Das Video erklärt, wie man Regen in sein Szenario einbaut. Dazu wird ein spezielles einzelnes Regen-Sprite verwendet, das alle Regentropfen enthält und so programmiert ist, dass es sich von oben nach unten bewegt.



- Folgen Sie den Anweisungen im Video
- Weiter drücken
  - Blitzschlag (3)
- Folgen Sie den Anweisungen im Video
- Weiter drücken
  - Zufällige Blitze (4)
- Folgen Sie den Anweisungen im Video
- Gehe zu
  - Add-Ons (6)
- Sehen Sie sich das kurze Video an, und wählen Sie unten auf der Seite eine oder zwei der möglichen Erweiterungen aus.

Aktivität 3.2 Erstellung eines Labyrinthspiels

Einführung: Algorithmen

Ein Algorithmus ist eine Liste von Anweisungen: ein schrittweiser Plan. Wenn ein Algorithmus genau genug formuliert ist, sollte jemand anderes in der Lage sein, die Schritte genau so auszuführen, wie Sie es gemeint haben. Vor allem Algorithmen, die von einem Computer ausgeführt werden sollen, werden in einer sehr präzisen Sprache, dem Programmcode, formuliert, der dem Computer Schritt für Schritt genau sagt, was er tun soll.

Paaraufgabe

Sehen Sie sich das folgende Video an: Was ist ein Algorithmus und warum sollten Sie sich dafür interessieren?

Phase 1: Untersuchung von Algorithmen anhand von Flussdiagrammen

Betrachten Sie den folgenden Algorithmus.

1. Zeichnen Sie eine vertikale Linie
2. Ziehen Sie eine horizontale Linie darüber
3. Ziehen Sie eine diagonale Linie von der Spitze der vertikalen zur Spitze der horizontalen Linie
4. Wiederholen Sie Anleitung 3 für alle übrigen Ecken.
5. Ziehen Sie eine Wellenlinie von der unteren Spitze

Gruppenaufgabe

Zeichnen Sie ein Bild nach dieser Anleitung.

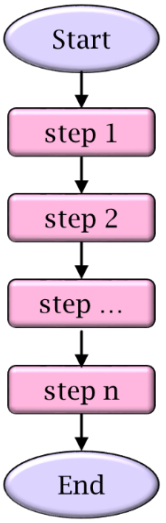
Vergleichen Sie Ihre Zeichnung mit der von anderen. Diskutieren Sie, warum es einfach war oder nicht und erklären Sie, warum.

*

Das beabsichtigte Ergebnis des Algorithmus war die Zeichnung eines Drachens (siehe Anhang A für eine mögliche Darstellung). Das Beispiel zeigt, dass, wenn die Anweisungen unklar oder unvollständig sind, das Ergebnis mehrdeutig sein kann. Manchmal wird vom Ausführenden erwartet, dass er Mehrdeutigkeiten selbst auflöst. Überlegen Sie, was passieren würde, wenn wir die Anweisungen immer genau so befolgen würden, wie sie gegeben werden.

Paaraufgabe

Sehen Sie sich das Video mit den genauen Anweisungen an.



Wenn wir einen Algorithmus beschreiben wollen, müssen wir uns auf eine Sprache einigen, die wir dafür verwenden. Eine solche Sprache muss hinreichend präzise sein: Es steht fest, was zu tun ist, wenn wir einem in dieser Sprache geschriebenen Algorithmus folgen. Ein Algorithmus kann visuell in einem Flussdiagramm dargestellt werden. Dies hilft, den Überblick zu behalten, was wiederum das Schreiben, Lesen und Analysieren von Anweisungen erleichtert.

Die einfachsten Algorithmen bestehen aus einer Reihe von Anweisungen, die nacheinander ausgeführt werden. In einem Flussdiagramm stellen wir eine solche Abfolge von Anweisungen wie rechts dargestellt dar. Sie beginnen oben bei 'Start'. Der Teil zwischen "Start" und "Ende" (der Hauptteil des Flussdiagramms) beschreibt, was der Algorithmus tatsächlich tut, dieser Mittelteil besteht aus einer Abfolge von mehreren Einzelschritten (steps).

Aufgabe zum Selbststudium

Zeichnen eines Quadrats

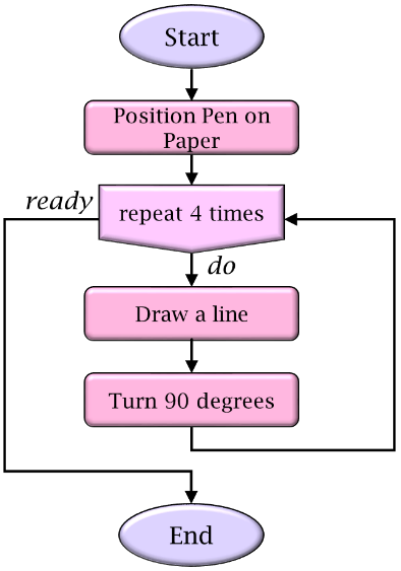
Gehen wir davon aus, dass Sie die folgenden grundlegenden Befehle verwenden können:

- **Zeichnen Sie eine Linie**
- **Um 90 Grad drehen**
- **Stift auf Papier positionieren**

Wählen Sie eine Folge von geeignete Anweisungen aus und platzieren Sie diese in einem Flussdiagramm, um ein Quadrat zu zeichnen.

*

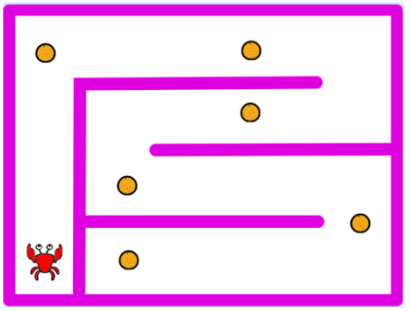
Ihre Lösung besteht wahrscheinlich aus einer langen Folge von Anweisungen, die immer wieder gleiche Abfolgen des selben Codes enthalten. Wir können eine solche Folge kompakter und damit übersichtlicher aufschreiben, indem wir ein repeat/while-Konstrukt verwenden: Solange wir noch keine 4 Wiederholungen durchgeführt haben, wird der do-Zweig verfolgt. Nach der vierten Iteration folgen wir dem ready-Pfeil und der Algorithmus endet.



Paaraufgabe

- Ein Schüler wählt ein Bild aus und schreibt Anweisungen, um seinem Partner zu erklären, wie das Bild zu zeichnen ist.
- Einigt euch auf Grundbefehle (Anweisungen), die in der Beschreibung verwendet werden können
- der andere Schüler befolgt diese Anweisungen genauestens
- Sie diskutieren das Ergebnis und korrigieren Fehler in den Anweisungen

Phase 2: Ein Labyrinthspiel mit Sprites erstellen



In dieser Phase entwerfen Sie ein Spiel in Scratch, in dem der Spieler eine Krabbe mit den Pfeiltasten steuert. Das Ziel des Spiels ist es, den Weg zur anderen Seite des Labyrinths zu finden, ohne die Wände zu berühren. In dem Labyrinth sind Orangen als “Schätze” verstreut. Der Spieler sammelt Punkte, indem er die Krabbe die Orangen aufsammeln lässt.

Lernressourcen

Scratch-Datei: Krabbe-im-Labyrinth-Start.sb3

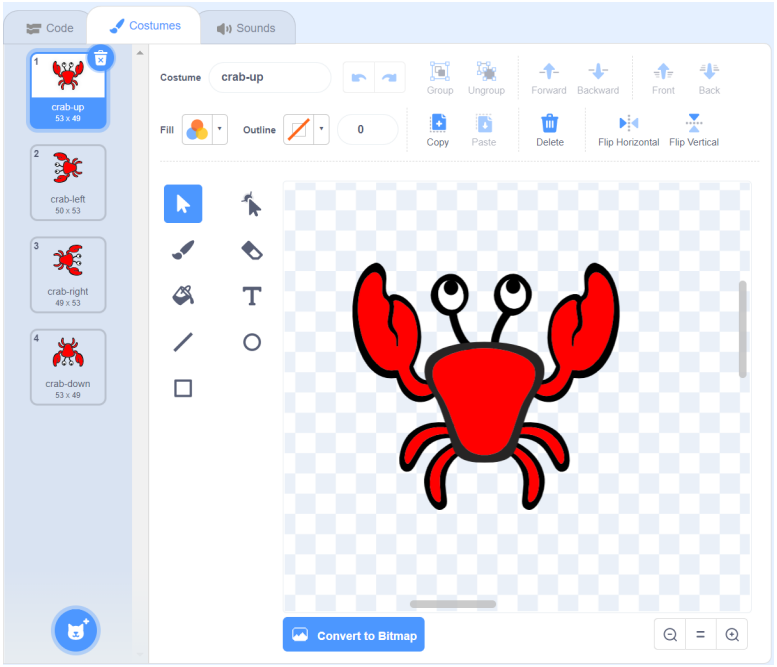
Paaraufgabe

Versuchen Sie herauszufinden, wie Ihr Spiel ablaufen könnte. Machen Sie eine grobe Skizze Ihres Spiels. Überlegen Sie, wie das Spiel aussieht und wie die Hauptfigur vom Spieler gesteuert wird. Versuchen Sie, Bedingungen zu formulieren, die anzeigen, ob das Spiel beendet ist. Welche "Schätze" fügen Sie in Ihr Spiel ein? Wo befinden sie sich und wie hebt der Spieler sie auf?

*

Um dieses Spiel zu programmieren, werden Sie lernen, wie man

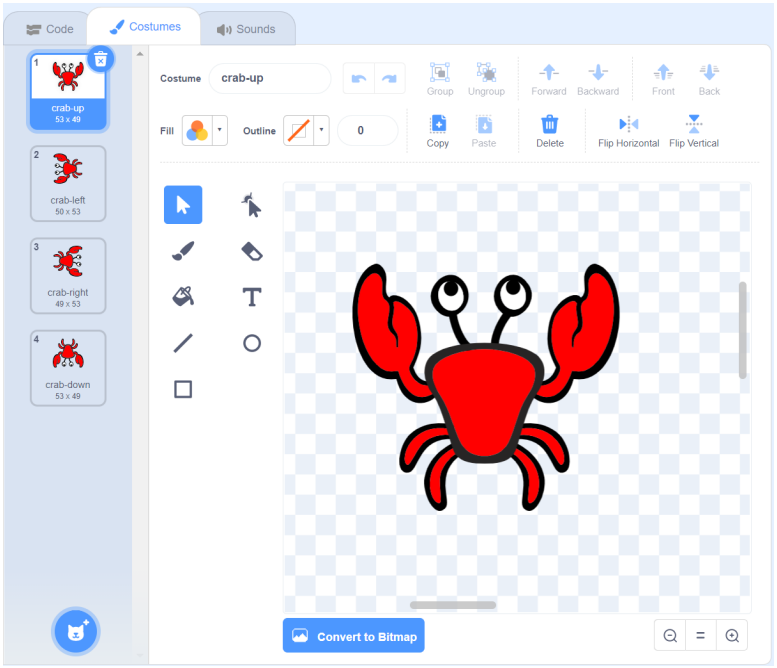
- Sprite-Kostüme, Positionierung und Bewegung definiert
- Tastatureingaben verarbeitet
- Interaktion des Sprites mit der Umgebung festlegt
- Sprites programmiert, damit sie miteinander interagieren
- Variablen verwendet, um einen Spielstand anzuzeigen



Wir beginnen mit der Anpassung des Aussehens der Hauptfigur des Spiels: der Krabbe (in Scratch als Sprite implementiert).

Paaraufgabe

- Gehe zu Scratch und öffne die Datei crab-in-maze-start.sb3 (indem du im Menü "Datei" die Option "von deinem Computer laden" wählst).
- Wähle das Krabben-Sprite und wähle die Registerkarte Kostüme.
- Entferne das zweite Bild, wähle das erste Bild, verkleinere die Krallen der Krabbe und skaliere das gesamte Bild so, dass die Krabbe bequem in das Labyrinth passt (so dass die Wände nicht berührt werden)
- Duplizieren Sie das Bild dreimal (indem Sie mit der rechten Maustaste auf das Bild klicken und "Duplizieren" wählen) und drehen Sie diese Kopien, um Ihre Krabbe aus der Vogelperspektive zu sehen: links, rechts, oben und unten.

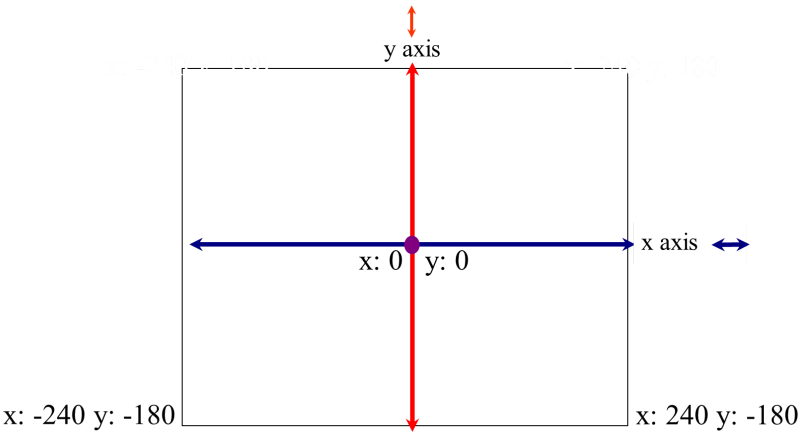


- Ändern Sie die Namen entsprechend.

*

Um die Bewegung der Krabbe zu programmieren, muss man wissen, wie Sprites in Scratch positioniert werden. Scratch stellt jeden Punkt auf dem Spielfeld als ein Paar von zwei Zahlen dar: Die erste gibt den horizontalen Abstand von der Mitte des Feldes an, die zweite den vertikalen Abstand.

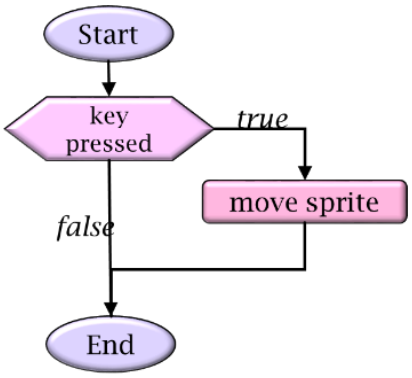
Technisch gesehen verwendet Scratch ein sogenanntes Koordinatensystem mit dem Ursprung in der Mitte des Spielfelds. Jeder Punkt wird durch ein Paar (x, y) dargestellt, wobei x die horizontale Entfernung vom Ursprung und y die vertikale Entfernung ist.



Das gesamte Spielfeld ist 480 Pixel breit und 360 Pixel hoch. Die obere linke Ecke und die obere rechte Ecke haben die Koordinaten (-240,180) bzw. (240,180).

Zurück zu unserem Spiel. Der Spieler bewegt die Krabbe mit den Pfeiltasten. Die Frage ist nun, wie wir in unserem Programm erkennen können, ob der Spieler eine Taste gedrückt hat und wie wir dann die Position der Krabbe entsprechend der gedrückten Taste anpassen können,

Der Algorithmus für die Behandlung der gedrückten Pfeiltasten kann in einem Flussdiagramm wie folgt beschrieben werden:

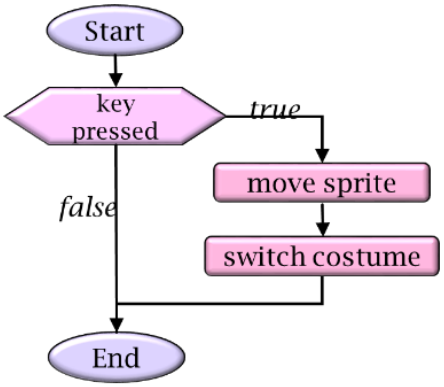


Wir haben hier eine Auswahl verwendet, die auch als Wenn-Dann-Konstrukt (If-Then) bezeichnet wird. In einer solchen Auswahl wird anhand eines bedingten Ausdrucks (im sechseckigen Block angegeben) bestimmt, ob der dann-Zweig (der Zweig mit der Bezeichnung true) ausgeführt werden soll. Letzteres geschieht nur, wenn die Bedingung wahr ist.

Paaraufgabe

- Versuchen Sie herauszufinden, welchen *Ereignis-Block* Sie verwenden können, um auf gedrückte Pfeiltasten zu reagieren
- Stellen Sie die Position der Krabbe richtig ein. Welchen *Bewegungs-Block* brauchen Sie?

  Um das Spiel realistischer zu machen, werden wir das Bild der Krabbe der Bewegungsrichtung anpassen. Wir tun dies, indem wir immer das richtige Kostüm auswählen. Ausgedrückt in einem Flussdiagramm:

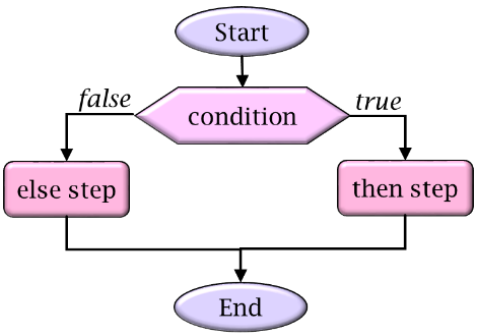


- Bestimmen Sie, welchen *Aussehen-Block* Sie dafür benötigen.

- Führen Sie Ihren Code aus und testen Sie ihn.

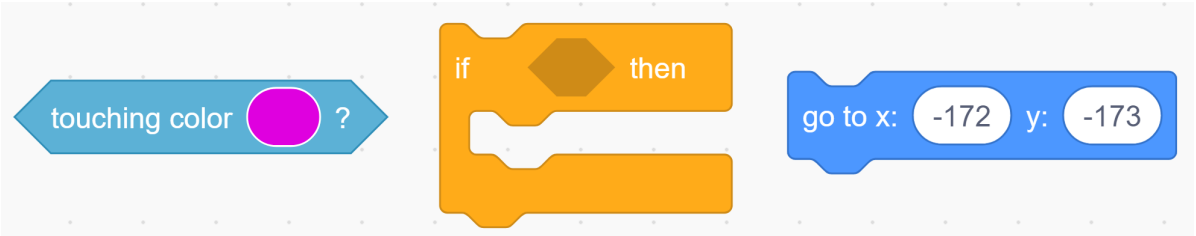
*

Die allgemeine Variante einer Auswahlanweisung bietet die Möglichkeit, auch eine Aktion auszuführen, wenn die Bedingung falsch ist. Im vorherigen Beispiel ist bei einer falschen Bedingung nichts passiert. In einem Flussdiagramm wird eine falls-dann-sonst-Anweisung (if-then-else) wie folgt dargestellt.



Wenn die Bedingung wahr ist, wird der 'then-Zweig' (mit *true* gekennzeichnet) ausgeführt. Andernfalls wird der "else-Zweig" (mit *false* gekennzeichnet) ausgeführt.

Scratch hat die folgenden Blöcke, um die Maus zu bewegen und zu überprüfen, ob die Krabbe die Wände des Labyrinths trifft.



Paaraufgabe

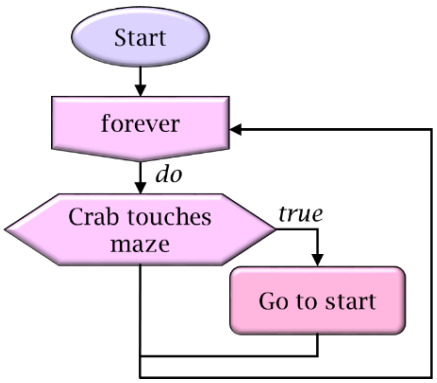
Bestimmen Sie, welche Blöcke Sie benötigen und wie die Reihenfolge der Blöcke sein muss.

- Wenn die Krabbe die Seiten des Labyrinths berührt, wie man dann die Krabbe zurück zum Anfang bewegt.
- Fügen Sie die entsprechenden Blöcke zu Ihrem Code hinzu
- Führen Sie Ihren Code aus und testen Sie ihn.
    - Funktioniert es? Wenn nicht, versuchen Sie herauszufinden, was falsch läuft, und beheben Sie es.

*

Wir haben bereits gesehen, wie wir in einem Flussdiagramm ausdrücken können, dass ein bestimmter Teil eines Algorithmus mehrmals wiederholt werden soll. Es ist auch möglich

anzugeben, dass ein Algorithmus immer wiederholt ausgeführt werden muss und daher nie aufhört. In unserem Krabbenspiel können wir dies nutzen, um immer wieder zu überprüfen, ob die Krabbe die Wände berührt und ggf. wieder an den Anfang gesetzt werden soll. Im Flussdiagramm:



Beachten Sie, dass dieses Flussdiagramm keinen Endknoten hat.

Paaraufgabe

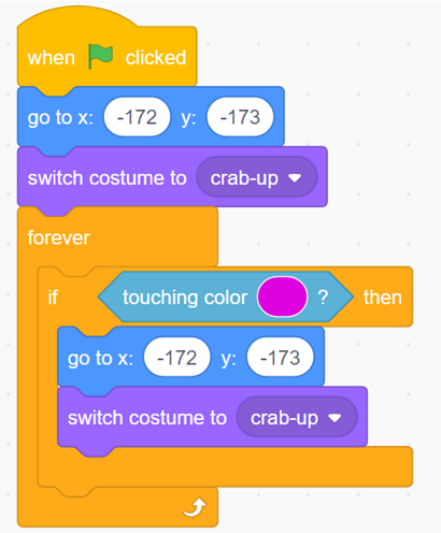
- Verwenden Sie die Endlosschleife "wiederhole fortlaufend" (aus den *Steuerung-Blöcken*), um die Bewegung der Krabbe zu steuern.
- Finden Sie heraus, was zu tun ist, wenn die Krabbe eine Wand trifft.
- Führen Sie Ihren Code aus und testen Sie ihn.

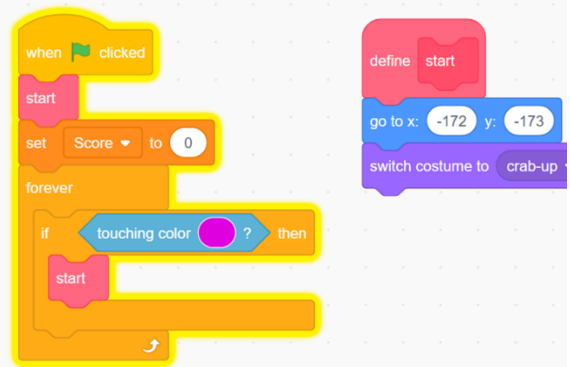
*

Ein natürlicher Weg, größere Aufgabenstellungen zu lösen, besteht darin, sie in eine Reihe von Teilproblemen zu zerlegen, die mehr oder weniger unabhängig voneinander gelöst und dann kombiniert werden, um zu einer vollständigen Lösung zu gelangen. In der Entwurfsphase eines Algorithmus sollte der Problemlöser bei der Unterteilung eines Problems in Teilaufgaben nur berücksichtigen, was ein Teilalgorithmus global tun soll, und sich nicht um die Details dieser Teilaufgaben kümmern. Diese Trennung der Aufgaben wird als Abstraktion bezeichnet. Durch den Prozess der Abstraktion kann ein Programmierer alle Details über Teilalgorithmen ausblenden, um die Komplexität zu verringern und das Programm verständlicher zu machen.

Scratch verwendet Abstraktionen, um die Programmierung zu erleichtern, indem es eine Vielzahl von Grundbausteinen anbietet, deren komplexe zugrundeliegende Implementierungen verborgen bleiben. Scratch-Programmierer können Abstraktionen einführen, indem sie dem Programm selbst neue Blöcke hinzufügen.

Betrachten Sie zum Beispiel die folgenden Programmschnipsel, die beide eine Lösung für das vorhergehende Problem sein können. Auf der linken Seite sehen Sie die Lösung, bei der alles direkt ausgearbeitet ist. Rechts wird die Abstraktion genutzt, indem das Bewegen der Krabbe und das Wechseln des Kostüms als neuer Baustein (mit dem Namen Start) eingeführt und dann im Hauptprogramm verwendet wird.





Paaraufgabe

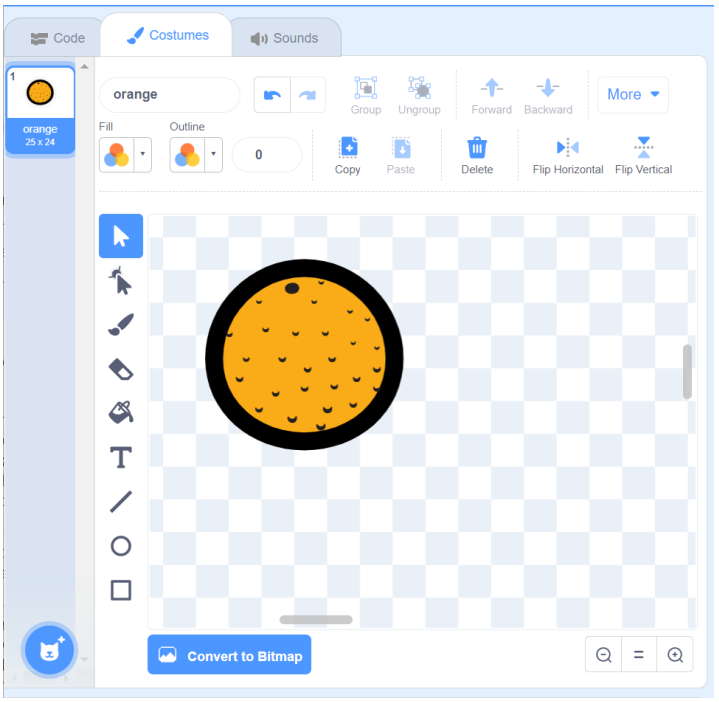
Gehen Sie Ihr Scratch-Szenario durch und wenden Sie Abstraktion an, indem Sie (detaillierte) Codefragmente durch selbst definierte Blöcke ersetzen.

*

Ein Spieler sollte nun in der Lage sein, die Krabbe mit den Pfeiltasten zu steuern, und die Krabbe reagiert richtig, wenn eine Wand des Labyrinths getroffen wird. Wir werden nun das Spiel attraktiver machen, indem wir Herausforderungen hinzufügen. Zunächst beschränken wir uns auf Orangen, die wir über das Labyrinth verteilen und die von der Krabbe gefressen werden. Dies geschieht, sobald die Krabbe sie berührt. Wer eine Orange isst, erhält einen Bonuspunkt. Das Ziel des Spiels ist es nun, so viele Bonuspunkte wie möglich zu sammeln.

Paaraufgabe

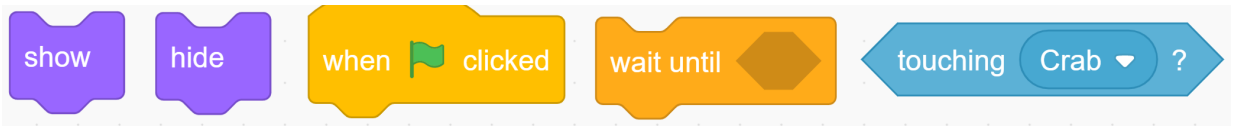
- Wählen Sie das Orange-Sprite aus und skalieren Sie die Orange so, dass die Größe im



  Verhältnis zur Krabbe steht.

  Wir stoßen nun auf ein interessantes Problem: Wen machen wir für das Erkennen eines Zusammentreffens und und die Handhabung der Interaktion zwischen Krabbe und Orange verantwortlich? In der realen Welt wird dies immer die Krabbe sein, aber hier haben wir eine Wahl. Wir können der Orange mehr Verantwortung übertragen als in der Realität. Und diese interessante Option werden wir nun nutzen: Wir lassen die Orange erkennen, ob sie von der Krabbe gefunden wurde und lassen sie sich fressen. Letzteres erreichen wir, indem wir die Orange verstecken.

- Unten sehen Sie die Blöcke, die Sie (wahrscheinlich) zur Programmierung der Orangen benötigen. Verwenden Sie diese Blöcke, um das gewünschte Verhalten zu erreichen.

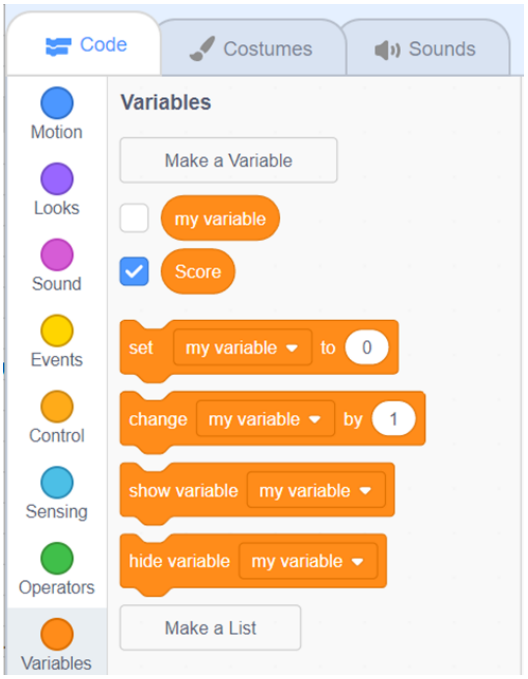


- Führen Sie Ihren Code aus und testen Sie ihn!
  - Gibt es noch etwas, das zu diesem Zeitpunkt geschehen sollte?

*

Wir wollen auch eine Möglichkeit in das Programm einbauen, um den Spielstand festzuhalten. Zu diesem Zweck werden wir eine sogenannte Variable verwenden. In Programmiersprachen ist eine Variable ein Behälter, der genau eine Information aufnehmen kann, z. B. ein Wort oder eine Zahl. Durch die Möglichkeit, diese Information zu speichern, können wir sie an

verschiedenen Stellen im Programm verwenden oder auch verändern. Diese Fähigkeit macht Variablen unglaublich nützlich.



Wie erstellt man eine Variable in Scratch? Bevor wir eine Variable verwenden können, müssen wir sie zunächst mit der Schaltfläche "Neue Variable" in der Blockpalette erstellen (bei den Variablen-Blöcken). Der Wert, den eine Variable in Scratch enthalten kann, ist entweder ein Text oder eine Zahl. Außerdem gibt es Blöcke, um einer Variablen einen (anfänglichen) Wert zu geben und um den Wert während der Ausführung des Programms zu ändern. Auf der rechten Seite sehen Sie, welche Blöcke Sie zur Manipulation von Variablen verwenden können. Übrigens muss das Wertefeld z.B. im Set-Block nicht unbedingt eine Zahl sein: Auch komplexere Ausdrücke, die mit Operatoren aus der Operatorenpalette gebildet werden, sind hier erlaubt. Nehmen wir z.B. an, Score hat den Wert 4. Nach der Ausführung von

hat Score den Wert 3 * 4 + 2 = 14.

Paaraufgabe

- Fügen Sie eine Variable mit dem Namen Score in Ihr Programm ein.
- Überlegen Sie, welchen Wert diese Variable zu Beginn hat, und überlegen Sie, wie Sie diesen Wert einstellen können.
- Überlegen Sie auch, wo und wie der Wert von Score angepasst werden sollte und welcher Block dafür verwendet werden kann!
- Führen Sie Ihren Code aus und testen Sie ihn!
- Duplizieren Sie die Orange (im Sprites-Fenster) mehrmals (z.B. 6) und platzieren Sie diese Kopien irgendwo im Labyrinth.

*

UNIT 4: Lehren und Lernen von Computational Thinking

Gesamtzeit: 8 Stunden

Das Lehren und Lernen im Bereich des Computational Thinking kann im Wesentlichen auf zwei Arten erfolgen: ohne Computer ("unplugged") und mit Computern ("plugged"). In dieser Unit werden Sie beide Arten kennenlernen und sich mit den Gestaltungsprinzipien von Lehrstrategien für Computational Thinking vertraut machen.

Beitrag zu den Lernergebnissen

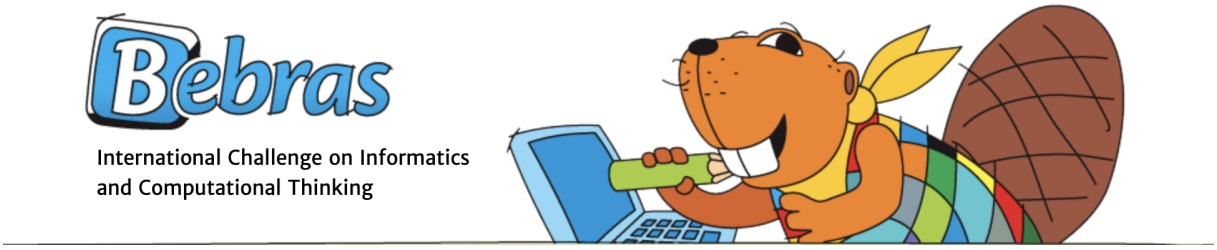
Lernergebnisse: Studierende können am Ende der Unit 4
• die Merkmale von Lehr- und Lernaktivitäten für Computational Thinking beschreiben, die sowohl ohne als auch mit Computertechnik durchgeführt werden können • wesentliche Elemente von Unterrichtsstrategien für Computational Thinking beschreiben • solche Elemente in konkreten Unterrichtsmaterialien und Aktivitäten erkennen

Aktivität 4.1 Bebras Aufgaben

45

Bebras

Bebras ist ein Online-Wettbewerb und eine Challenge, die von einer internationalen Gemeinschaft organisiert wird. Die Elemente des Wettbewerbs sind sogenannte Bebras-Aufgaben, die jeweils ein oder mehrere informatische Konzepte abdecken. In dieser Aktivität werden Sie Bebras-Aufgaben sowohl als Einheiten des Wettbewerbs als auch als "unplugged"-Lernaktivitäten für Computational Thinking untersuchen.



Paaraufgabe

- Gehen Sie auf die lokale Bebras-Website und wählen Sie bis zu fünf Bebras-Aufgaben aus (versuchen Sie, etwas Abwechslung einzubauen).
- Führen Sie die Aufgaben einzeln aus und vergleichen und besprechen Sie sie anschließend mit Ihrem Partner.
- Können Sie die Aufgaben in Bezug auf die Schritte und Aktivitäten des Computational Thinking, die Sie in diesem Modul gelernt haben, einordnen?

Bebras Aufgaben als Lernaktivitäten

Die Aufgaben von Bebras sind nach den zu behandelnden CT Konzepten kategorisiert. Zu jeder Aufgabe gibt es eine Erläuterung der Verbindung zwischen der Aufgabe und der Informatik.

Valentina Dagienė (die "Mutter von Bebras") und Sue Sentance untersuchten den Einsatz von Bebras-Aufgaben als COMPUTATIONAL THINKING-Lernaktivitäten:

Dagienė, V., & Sentance, S. (2016). It's Computational Thinking! Bebras tasks in the curriculum. In *International conference on informatics in schools: Situation, Evolution and Perspectives* (S. 28-39).

Paaraufgabe

- Lesen Sie den Artikel.

- Überprüfen Sie die Konzepte und Aktivitäten, die Sie in den Units 2 und 3 kennen gelernt haben. Finden Sie für jedes dieser Units ein Beispiel für ein informatisches Konzept und eine passende Bebras-Aufgabe.

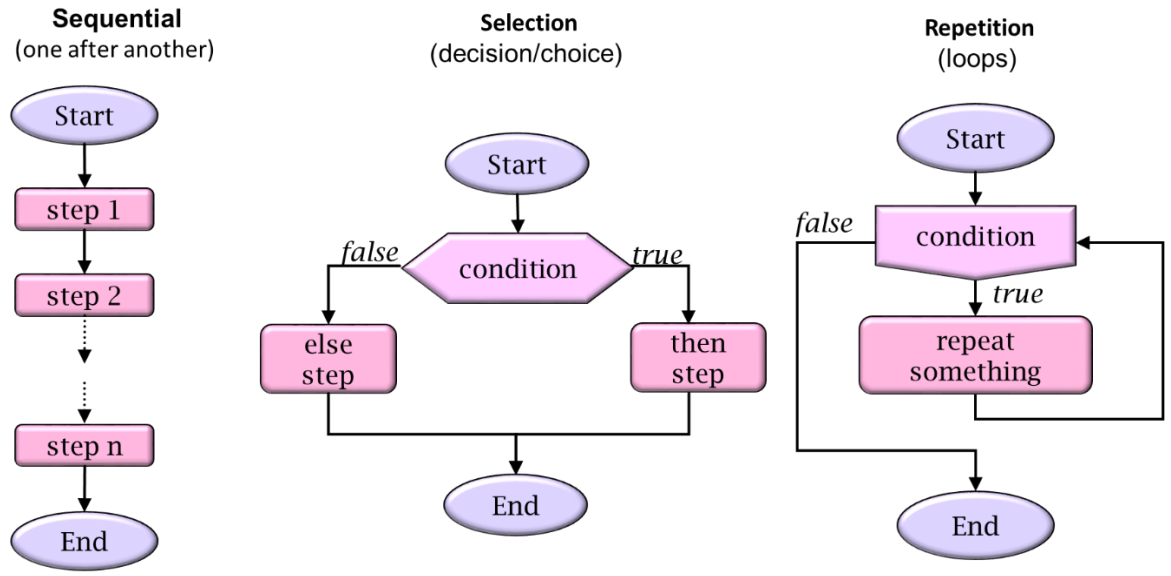
Aktivität 4.2 Wegfinden: Wege in einem Labyrinth

Das Lösen eines Labyrinths ist der Prozess, einen Weg durch das Labyrinth vom Anfang bis zum Ende zu finden. Einige Methoden zur Lösung von Labyrinthen sind so konzipiert, dass sie innerhalb des Labyrinths von einem Reisenden ohne Vorkenntnisse des Labyrinths verwendet werden können, während andere für eine Person (oder ein Computerprogramm) gedacht sind, die das gesamte Labyrinth auf einmal sehen kann. Das Lösen von Labyrinthen ist eine Variante einer allgemeineren Klasse von Problemen, die unter dem Namen "Pathfinding" bekannt ist. Bei einer Reihe von Orten, die miteinander verbunden sein können (z. B. eine Reihe von Städten, die durch Straßen miteinander verbunden sind), sucht ein Wegfindungsalgorithmus nach einer Route, die diese Orte miteinander verbindet, mit dem Ziel, die beste Route (z. B. die kürzeste oder billigste Route) zu ermitteln.

Unsere Absicht ist es, mit diesen Wegfindungsproblemen einen ersten Einblick in die Problemanalyse und den Algorithmenentwurf ohne detaillierte Kenntnisse der Computerprogrammierung und Programmiersprachen zu gewinnen. Um jedoch mögliche Lösungen hinreichend präzise formulieren zu können, werden wir zunächst wieder Flussdiagramme verwenden.

Elemente von Flussdiagrammen

Unten sehen Sie die drei Grundbausteine, mit denen Flussdiagramme erstellt werden können.



Jedes Flussdiagramm wird aus den folgenden drei Grundelementen aufgebaut

1. **Sequenz**: eine geordnete Folge von Anweisungen, die nacheinander ausgeführt werden.
2. **Auswahl**: Auf der Grundlage eines bedingten Ausdrucks wird bestimmt, ob der dann-Zweig (der Zweig mit der Bezeichnung *true*) oder der sonst-Zweig (der Zweig mit der Bezeichnung *falsch*) ausgeführt werden soll.
3. **Wiederholung**: Der Hauptteil des Elements (der Zweig mit der Bezeichnung *true*) wird so lange wiederholt, wie die Bedingung als true ausgewertet wird. Dann wird das Programm mit der Verzweigung mit der Bezeichnung false fortgesetzt.

Problem des Handlungsreisenden

Das sogenannte Problem des Handlungsreisenden (Travelling Salesperson Problem) ist ein klassisches Beispiel für eine Aufgabe, bei der eine optimale Lösung gesucht wird.

"Gegeben ist eine Liste von Städten und für je 2 Städte deren Entfernungen zueinander. Was ist die kürzest mögliche Route, die ausgehend von einer Stadt zu jeder anderen Stadt führt und zur Ausgangsstadt zurückführt?"

Wir werden uns nun eine Lösung für dieses Problem ansehen, die zwar nicht immer die aller kürzeste Route findet, aber intuitiv und leicht verständlich ist, nämlich den *Algorithmus des nächsten Nachbarn*. Der Algorithmus des nächsten Nachbarn (Nearest Neighbour Algorithm NNA) lässt den Handelsreisenden stets die nächstgelegene noch unbesuchte Stadt als nächste besuchen.

Paaraufgabe

Biber schwimmen beginnend bei der Stadt A, besuchen die Städte B, C und D und kommen dann wieder in A an. Sie wollen den kürzesten Weg finden.

1. Welchen Weg finden sie, wenn sie NNA nutzen würden?
2. Gibt NNA immer die richtige Antwort (d.h. die **kürzeste** Rundreise für eine beliebige Liste von Städten und Entfernungen zwischen diesen)? Wenn nicht, geben Sie ein konkretes Gegenbeispiel!

*

Obwohl NNA intuitiv klar ist, kann es schwierig sein, diesen Algorithmus präzise zu formulieren. Wir versuchen, diesen Algorithmus in einem Flussdiagramm auszudrücken, wobei wir zunächst festlegen müssen, welche primitiven Anweisungen wir dafür verwenden können. Diese Primitive müssen einerseits mächtig genug sein, andererseits aber auch ausreichend abstrakt, damit wir uns nicht in allerlei Details verheddern.

Paaraufgabe

Geben Sie den NNA in Form eines Flussdiagramm an.

Tipp: Denken Sie zunächst an die grundlegenden Anweisungen/Primitive, die Sie verwenden dürfen.

Einen Weg in einem Labyrinth finden

Die Suche nach einem Weg in einem Labyrinth ist ein weiteres Beispiel für eine Aufgabe, die leicht zu verstehen, aber auch schwierig genug zu lösen ist. Es gibt mehrere Varianten dieses Problems: Finde einen Weg aus einem Labyrinth heraus, finde einen Weg durch ein Labyrinth oder finde einen Weg zu einem bestimmten Ort innerhalb eines Labyrinths. Zunächst können

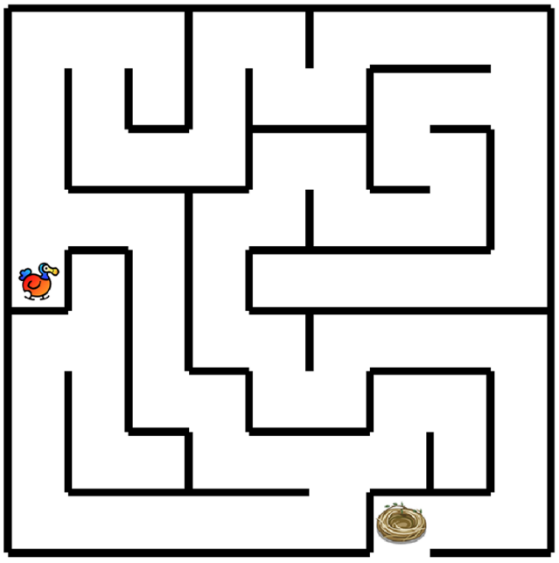
wir feststellen, dass diese Varianten verallgemeinert werden können zu: Finde einen Weg von Position A zu Position B.

Es ist auch wichtig, was wir über das Labyrinth wissen, während wir nach der Zielposition B suchen. In dieser Aufgabe gehen wir davon aus, dass wir nichts über die Größe und Struktur des Labyrinths wissen. Wir können durch die Korridore des Labyrinths gehen und dabei jeweils einen Schritt nach vorne schauen. Wir gehen also davon aus, dass wir an jedem Punkt des Labyrinths überprüfen können, ob wir geradeaus gehen können (und somit nicht vor einer Wand stehen) und eventuell unsere Bewegungsrichtung anpassen können.

Die bekannteste Regel für das Durchqueren eines Labyrinths ist die Wandfolge-Regel (auch bekannt als Linke-Hand-Regel oder Rechte-Hand-Regel). Wenn man das Labyrinth betritt, legt man z.B. die linke Hand an die linke Wand und folgt der Wand bis man den Ausgang erreicht. Wenn alle Wände eines Labyrinths miteinander verbunden sind, kann man mit dieser Methode auch von jeder Stelle im Inneren einen Ausgang finden.

Um das Problem konkreter zu machen, verwenden wir das folgende Szenario, in dem der Labyrinth-Reisende durch einen Dodo dargestellt wird, der sich irgendwo im Labyrinth befindet und versucht, sein anderswo verstecktes Nest zu finden.

Bevor wir einen Suchalgorithmus angeben können, müssen wir wieder angeben, mit welchen primitiven Anweisungen wir den Dodo steuern können.

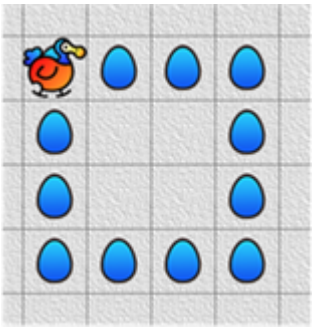


Wir unterscheiden zwischen *Befehlen* (damit der Dodo einen Schritt vorwärts macht oder die Bewegungsrichtung ändert) und *Abfragen* (mit denen wir den Dodo lokale Informationen über seine Umgebung bekommt).

- Befehle:
  - *links abbiegen*: um 90 Grad gegen den Uhrzeigersinn drehen
  - *rechts abbiegen*: um 90 Grad im Uhrzeigersinn drehen
  - *bewegen*: ein Feld vorwärts gehen
  - *Ei legen*: ein Ei legen (an der aktuellen Position)
- Abfragen/Tests:
  - *nach vorne gehen möglich?* Kann Dodo einen Schritt nach vorne machen?
  - *Nest gefunden?* Hat Dodo das Nest gefunden?

Paaraufgabe

Zeigen Sie anhand eines Flussdiagramms einen Algorithmus, mit dem der Dodo das folgende Eiermuster erzeugt. Tipp: Verwenden Sie Wiederholungen.



Paaraufgabe

Beschreiben Sie in einem Flussdiagramm die "Linke-Hand-Regel".

Pfadfindung unplugged

Vorbereitung

Stecken Sie zum Beispiel mit Malerband ein Labyrinth auf dem Boden ab. Achten Sie darauf, dass die Gänge breit genug sind, damit eine Person problemlos hindurchgehen kann. Das Labyrinth kann etwas einfacher sein als das Labyrinth auf dem Bild. Tauschen Sie untereinander die Flussdiagramme aus der vorherigen Aufgabe.

Paaraufgabe

Bestimmen Sie einen Ort im Labyrinth als Endziel und eine Person soll irgendwo im Labyrinth weit genug von diesem Endziel entfernt stehen.

- Die andere Person liest die Anweisungen aus dem Flussdiagramm vor, und derjenige, der sich im Labyrinth befindet, folgt diesen Anweisungen akribisch.
- Wird das Endziel erreicht? Wenn ja, probieren Sie aus, ob dies auch für andere Startorte im Labyrinth gilt. Wenn nicht, was stimmt mit dem Algorithmus nicht? Verbessern Sie gegebenenfalls den Algorithmus und prüfen Sie, ob die Verbesserung den gewünschten Effekt hat.
- Geben Sie die verbesserte Version des Flussdiagramms an ihre Besitzer zurück.

Lernressourcen

- Scratch-Datei: PathThroughMaze.sb3

Pfadfinden in Scratch

Schließlich werden Sie den Algorithmus aus der vorherigen Aufgabe in Scratch implementieren. Wir haben ein Startszenario erstellt, das Sie dafür verwenden können. Öffnen Sie das Startszenario PathThroughMaze.sb3.

Es folgt nun eine kurze Erklärung. Die primitiven Dodo-Anweisungen wurden dem Szenario als separate Blöcke hinzugefügt. Der Befehl *layEgg* fehlt, da er für diese Aufgabe nicht benötigt wird. Alle anderen Befehle funktionieren wie zuvor beschrieben. Die Abfragen wurden auf eine besondere Weise realisiert, auch weil selbstdefinierte Blöcke in Scratch keine Ergebnisse liefern können. Der Test, um zu prüfen, ob der Dodo das Nest gefunden hat, kann leicht mit einem vordefinierten Block aus der Fühl-Palette realisiert werden. Wir haben einen neuen Block für den canMove-Test eingeführt. Um ein Ergebnis zurückzugeben, verwenden wir eine Variable, die wir, wie den Block selbst, ebenfalls *canMove* nennen. Nach dem Ausführen des canMove-Blocks können Sie also den Wert der entsprechenden Variablen überprüfen, um festzustellen, ob Dodo einen Schritt machen kann oder nicht. Die Variable *canMove* enthält entweder den Wert 0 oder 1. Der Wert 0 bedeutet, dass kein Schritt gemacht werden kann, während der Wert 1 anzeigt, dass dies möglich ist.

Paaraufgabe

Das Startszenario enthält einen Block namens *tryToFindExit*. Dieser Block tut noch wenig: Er bringt nur die Meldung auf den Bildschirm, dass der Ausgang gefunden wurde.

- Implementieren Sie diesen Block, indem Sie das Flussdiagramm in Scratch-Code umwandeln.
- Führen Sie Ihr Programm auch für andere Startpositionen des Dodos aus und testen Sie es. Passen Sie den Code bei Bedarf an.

Aktivität 4.3 Unterrichtsstrategien

In dieser Aktivität werden Sie Strategien und pädagogische Prinzipien für den Computational Thinking Unterricht erkunden.

Rückblick

Paaraufgabe

Diskutieren Sie Ihre Erfahrungen mit der Pfadfindeaktivität:

- Was sind die wesentlichen Unterschiede zwischen der "unplugged"- und der "plugged"-Variante?
- Welche Herausforderungen erwarten Sie für Ihre künftigen Schülerinnen und Schüler in jedem dieser Bereiche?

Unterrichtsprinzipien für das Computational Thinking

In einem Übersichtsartikel analysieren Lye und Koh (2014) 27 empirische Studien zum Thema "Computational Thinking", insbesondere Studien zu Unterrichtsaktivitäten im Zusammenhang mit dem Programmieren. Unter anderem identifizieren Lye und Koh Unterrichtsansätze, die das Computational Thinking fördern.

Lye und Koh haben die Ansätze in vier Kategorien eingeteilt, die oft in Kombinationen auftreten: *Betonung von Computerkonzepten*, *Reflexion*, *Informationsverarbeitung* und *Konstruktion von Programmen mit Hilfestellungen.*

Paaraufgabe

1. Lesen Sie die Zusammenfassung der vier Kategorien des Dokuments "Instructional strategies for computational thinking" (in den Lernressourcen zu finden).
2. Analysieren Sie die Aktivitäten zum Computational Thinking, die Sie selbst in diesem Modul durchgeführt haben. Welche der Strategien erkennen Sie wieder? Diskutieren Sie.

Paaraufgabe

1. Lesen Sie die Zusammenfassung über die Beurteilung des Computational Thinking (siehe das Dokument "Assessment in CT").
2. Diskutieren Sie, welche Beurteilungsansätze für die Aktivitäten, die Sie in Unit 2 oder Unit 3 durchgeführt haben, geeignet wären. Wählen Sie für jede Aktivität einen möglichen Ansatz aus. Wie würden Sie diese Ansätze nutzen, um die Fähigkeiten der Schülerinnen und Schüler im Bereich des Computational Thinking zu bewerten?

Abschluss

Diskussion

Vergleichen Sie die Lern-Ergebnisse in Ihrem Kurs. Welche Aspekte sind Ihnen leicht gefallen, welche waren schwierig? Was haben Sie gelernt?

Lernressourcen

- A. Instructional strategies for computational thinking (Zusammenfassung von Lehrstrategien für das Lehren und Lernen von Computational Thinking, basierend auf Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51-61).
- B. Assessment in CT

Überblick über die Lernressourcen des Moduls 2

- Aktivität 1.1 - A. Eine kurze Einführung in das Computational Thinking
- Aktivität 1.1 - B. Definitions of CT
- Aktivität 2.1 - A. Gemeinschaft to Gesellschaft.pdf
- Aktivität 2.1 - B. The changing psychology of culture from 1800 through 2000
- Aktivität 2.1 - C. The changing psychology of culture in German‑speaking countries
- Aktivität 2.2 - zikavirusmodel.nlogo
- Aktivität 2.4 - Earth colour workbook.xlsx
- Aktivität 3.2 - crab-in-maze-start.sb3
- Aktivität 4.1 - Dagiene and Sentance 2016 (Dagienė, V., & Sentance, S. (2016). It's Computational Thinking! Bebras tasks in the curriculum. In *International conference on informatics in schools: Situation, evolution, and perspectives* (pp. 28–39)).
- Aktivität 4.2. PathThroughMaze.sb3
- Aktivität 4.3 - A. Instructional strategies for computational thinking (Summary of instructional strategies for CT teaching and learning, based on Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61.)
- Aktivität 4.3 - B. Assessment in CT