GamesLab Handbuch (Lehrerhandbuch)

Gregor Walter, KidsLab.de

Montag, 13. Januar 2025

Übersicht

Willkommen zum GamesLab-Handbuch für Lehrerinnen und Lehrer.

Liebe Lehrerinnen und Lehrer,

schön, dass Sie beim GamesLab dabei sind und Ihren Schülern die Chance geben, die ersten Schritte in der Spieleentwicklung mit Scratch zu gehen.

Im GamesLab Workshop haben Ihre Schüler einen ersten Eindruck davon bekommen, wie viel Spaß Programmieren machen kann und was man damit alles anstellen kann. Aber ein Workshop alleine reicht natürlich nicht aus, um wirklich fit zu werden.

Jetzt gibt es eine Challenge für sie: *schaffen sie es, das Interesse und die Neugier, die im GamesLab geweckt wurden, aufzugreifen und im regulären Unterricht weiter anzufachen?*

Sie kennen Ihre Schüler am besten und wissen, wie man sie motiviert und unterstützt: Gehen Sie mit gutem Beispiel voran und zeigen Sie, dass es sich lohnt, an einer Sache dranzubleiben, auch wenn es mal schwierig wird.

Mit Scratch lernen die Jugendlichen, wie man Probleme in kleine Teile zerlegt und Schritt für Schritt löst - eine Fähigkeit, die in allen Lebensbereichen Gold wert ist. Und nebenbei entstehen auch noch richtig coole Spiele und Animationen, auf die man stolz sein kann. Da macht Lernen doch gleich mal Spaß!

Als Starthilfe haben wir für Sie einen Lehrplan mit konkreten Unterrichtseinheiten zusammengestellt. Er baut direkt auf dem Schüler-Handbuch auf und enthält Zeitpläne, Lernziele und jede Menge praktische Tipps. Angepasst an Ihre Gegebenheiten lassen sich die Inhalte so ganz einfach in Ihren Unterricht integrieren.

Viel Freude und Erfolg dabei wünscht

Ihr GamesLab-Team

Wir freuen uns auf Ihr Feedback: was können wir besser machen, was hat funktioniert, welche Probleme hatten Sie? War der Zeitplan zu lang oder zu kurz? Was würden Sie sich sonst noch wünschen einfach eine Mail an team@kidslab.de oder per WhatsApp (siehe Kapitel Debug & Hilfe) schreiben.

Übersicht / GamesLab Lehrplan

Doppelstunde	Thema	Ziele	Aktivitäten	Hinweise
1	Einführung & Movie Director	Erste Schritte in Scratch, Kennenlernen der Oberfläche	Begrüßung, Accounts erstellen, Movie Director Tutorial, Experimentieren, Präsentation	Hausaufgabe: Scratch-Projekte erkunden
2	Katzenfreund & Sound Master	Bewegungssteuerung und Sound-Effekte	Projektvorstellung, Wiederholung, Katzenfreund Tutorial, Debugging, Präsentation	Debugging-Tipps im Handbuch nutzen
3	Cookie King	Variablen und Spielmechaniken verstehen	Einführung Variablen, Cookie King Entwicklung, Spielbalancing, Präsentation	Schüler ermutigen, Werte selbst auszubalancieren
4	Dino Runner Grundlagen	Komplexere Spielmechaniken	Einführung Spielkonzept, Grundmechaniken programmieren, Testing, Zwischenstand	Fortsetzung in nächster Stunde
5	Dino Runner Finalisierung & Effekte	Spiel vervollständigen und mit Effekten aufwerten	Dino Runner fertigstellen, Präsentation	-
6	Effekte	Spiele erweitern und verbessern	Effekte erstellen, Arbeit mit Klonen	-

Allgemeine Hinweise für Lehrkräfte

- Jede Einheit ist für eine 90-minütige Doppelstunde konzipiert
- Die Schüler arbeiten am besten jeder an einem Computer, sind nicht genügend Computer vorhanden können sich auch zwei Schüler einen teilen
- Nach jeder Stunde sollten die Projekte gespeichert und im Scratch-Account der Schüler veröffentlicht werden

- Ermutigen Sie die Schüler, sich gegenseitig zu helfen und voneinander zu lernen. Oft gibt es Schüler, die schneller fertig sind und auch die Zusatzaufgaben erledigt haben – wenn man sie darauf anspricht, helfen Sie gerne ihren Mitschülern

Scratch und Lehrplan

Die Doppelstunden sind für **Mittelschule / Realschule ab 7. Klasse** ausgelegt. Sie können sie auch in anderen Klassen nutzen - bitte beachten sie, dass sie dann mehr / weniger Zeit veranschlagen pro Einheit veranschlagen müssen.

Mittelschule:

- Wahlpflichtfächer ab 7. Klasse: “Informatik und digitales Gestalten”
- Scratch als Einstieg in algorithmisches Denken und Programmierung

Realschule:

- Pflichtfach Informationstechnologie (7. Klasse)
- Wahlpflichtfach Informatik (8.-10. Klasse)
- Scratch als Block-basierte Programmiersprache für erste Programmiererfahrungen

Gymnasium:

- Natur und Technik (6. Klasse): Einführung in algorithmisches Denken
- Informatik (9. Klasse): Block-basierte Programmierung
- Scratch als optionale Entwicklungsumgebung für den Einstieg

Der **LehrplanPLUS** sieht Scratch als geeignetes Werkzeug für:

- Entwicklung von Algorithmen
- Grundlagen der Programmierung
- Medienkompetenz
- Computational Thinking

Ist Scratch eine “echte” Programmiersprache?

Scratch ist keine “Spielzeug-Programmiersprache”, sondern ein tolles Werkzeug, das dieselben fundamentalen Konzepte wie professionelle Programmiersprachen vermittelt: Variablen, Schleifen, Funktionen, Ereignissteuerung und objektorientierte Programmierung.

Viele erfolgreiche Entwickler haben mit Scratch ihre ersten Coding-Erfahrungen gesammelt. Die visuelle Block-basierte Oberfläche eliminiert lediglich die Syntax-Hürden, die Anfänger oft frustrieren - die zugrundeliegende Logik und Komplexität entspricht echter Softwareentwicklung. Projekte wie Multiplayer-Spiele, Physik-Simulationen oder KI-Anwendungen beweisen das enorme Potenzial.

Nicht umsonst wird Scratch an Universitäten wie Harvard und dem MIT im Informatik-Grundstudium eingesetzt.

Motivation der Schüler

Was motiviert die Schüler
Nach ihrem Workshop im GamesLab führen wir eine Evaluation durch, um positive und negative Aspekte zu sammeln, vielleicht können Sie dies auch in ihrem Unterricht nutzen.
Aus den Schüler-Rückmeldungen wird klar, am GamesLab und Scratch begeistert vor allem:

1. Eigene Spiele entwickeln und kreativ sein
2. Selbstständig am Computer arbeiten
3. Praktisches Lernen statt “normaler Unterricht”
4. Gemeinsam programmieren und Erfolge teilen

Zusätzliche Tipps für Lehrkräfte:

1. **Differenzierung:**
    - Schnellere Schüler können zusätzliche Challenges aus den “NerdY Side Quests” bearbeiten
    - Langsamere Teams können sich auf die Grundfunktionen konzentrieren
2. **Problemlösung:**
    - Ermutigen Sie die Schüler, erst selbst nach Lösungen zu suchen
    - Nutzen Sie die Debug-Tipps aus dem Handbuch
    - Lassen Sie Schüler sich gegenseitig helfen
3. **Motivation:**
    - Regelmäßiges Präsentieren der Zwischenergebnisse
    - Positive Verstärkung auch für kleine Fortschritte
    - Fehler als Lernchancen nutzen

4. **Organisation:**
    - Projekte am Ende jeder Stunde speichern & veröffentlichen
    - Schüler dokumentieren ihre Fortschritte im Handbuch: **Achievements**
    - Erfolge mit Achievement-Stickern würdigen

Benötigte Materialien pro Doppelstunde:

- Computer mit Internetzugang (1 pro 2 Schüler)
- iPad oder Tablet ist auch möglich
- GamesLab Handbuch für jeden Schüler
- Kopfhörer für Sound-Tutorials
- Achievement-Sticker (optional)

Benötigte technische Ausstattung

Nicht jede Schule ist perfekt mit Computern, Computerräumen und Internet ausgestattet.

Scratch bietet eine Vielfalt von Möglichkeiten:

Benutzung mit dem iPad

Scratch läuft ohne Probleme auch auf iPads. Hier benötigen Sie allerdings zwingend eine Internetverbindung. Rufen Sie einfach in einem Browser ihrer Wahl die Scratch Webseite auf:

Https://scratch.mit.edu

Benutzung ohne Internet

Sollten Sie für die Arbeit mit ihren Schülern keine zuverlässige Internetverbindung zur Verfügung haben, gibt es auch eine offline Version von Scratch. Diese ist für Windows, Linux und macOS verfügbar. Einmal runtergeladen, funktioniert sie auch ohne Internet.

Natürlich können die Schüler dann ihre Projekte nicht direkt auf Scratch veröffentlichen und auch andere Sachen funktionieren nicht 100-prozentig, im Großen und Ganzen jedoch sehr gut.

https://scratch.mit.edu/download

Anzahl von Schüler pro Computer

Grundsätzlich ist es möglich, alleine oder zu zweit an einem Computer die Programme zu programmieren. Beides hat vor und Nachteile, die ich Ihnen hier kurz auflisten:

Programmieren alleine:

Vorteile:

- Eigenes Lerntempo bestimmen
- Ungestörtes Experimentieren
- Volle Kontrolle über das Projekt
- Keine Abstimmung nötig

Nachteile:

- Keine direkte Hilfe bei Problemen
- Gefahr des “Feststeckens”
- Weniger Austausch von Ideen
- Keine soziale Interaktion

Programmieren zu zweit (Pair Programming):

Vorteile:

- Gegenseitige Hilfe bei Problemen
- Austausch von Ideen und Kreativität
- Soziales Lernen
- Weniger Frustration durch gemeinsame Fehlersuche
- Bessere Codequalität durch “4-Augen-Prinzip”

Nachteile:

- Mögliche Dominanz eines Partners
- Unterschiedliche Arbeitsgeschwindigkeiten
- Koordinationsaufwand
- Geteilte Computerzeit

Hintergrund und Studien

Es gibt viele Studien, die beim Pair-Programming von Schülern viele Vorteile erkannt haben, zum Beispiel: https://repository.nie.edu.sg/server/api/core/bitstreams/2f9c05e5-b118-4357-9675-f52031d2dcaa/content
Diese wichtige Studie von 2002 untersuchte die Auswirkungen von Pair Programming in einem Einführungskurs zur Programmierung mit etwa 600 Studierenden. Die Hauptergebnisse:
Programmierleistung:

- Paare erzielten deutlich bessere Programmiernoten (86% vs. 67%)
- Auch im Vergleich zu den besten 50% der Einzelprogrammierer schnitten Paare besser ab (86% vs. 77%)

Abschlussquoten:

- Deutlich höhere Kursabschlussquote bei Paaren (92% vs. 76%)
- Die höhere Abschlussquote deutet darauf hin, dass Pair Programming Studierenden hilft durchzuhalten

Prüfungsleistung:

- Vergleichbare Ergebnisse in der Abschlussprüfung
- Bei Berücksichtigung der unterschiedlichen Abbruchquoten schnitten Paare sogar besser ab

Empfehlung für den Unterricht:

- Pair Programming bevorzugen
- Regelmäßiger Rollenwechsel (Pilot/Navigator)
- Bei Bedarf auch Einzelarbeit ermöglichen
- Flexible Gruppenzusammensetzung je nach Aufgabe

Aber ja - das ist geschackssache, probieren sie aus, was bei ihnen und ihren Schülern besser funktioniert.

Teilen & Weitermachen erwünscht! 🔁

Downloads von Unterlagen

Sie können dieses Handbuch als PDF, Word und E-Book unter folgender Adresse herunterladen: https://kids-lab.de/gameslab-lehrer

Auch die Handtücher für die Schüler sind in den gleichen Formaten hier verfügbar: https://kidslab.de/gameslab-handbuch

Dieses Handbuch wurde mit❤erstellt von deinem KidsLab-Team Version 1.0 - 2024

Doppelstunde 1: Einführung & Movie Director

Ziel: Erste Schritte in Scratch, Kennenlernen der Oberfläche

Das Movie Director Kapitel ist der perfekte Einstieg für Schüler in die Welt der Programmierung mit Scratch. In dieser ersten Doppelstunde geht es vor allem darum, die Schüler behutsam an die **Scratch-Oberfläche heranzuführen** und erste **Erfolgserlebnisse** zu schaffen.

Zeit	Aktivität	Material & Hinweise
10 min	**Begrüßung & Einstieg** - Vorstellung Scratch - Fun Fact über Scratch Community - Motivation: Was können wir mit Scratch machen?	- Handbuch S.1-3 - Demo-Projekte auf Beamer - Kurze Präsentation erfolgreicher Schülerprojekte
15 min	**Scratch-Accounts** - Erklärung der Passwort-Regeln - Paarweise Account-Erstellung - Zugangsdaten im Handbuch notieren	- Computer mit Internet - Handbuch Seite “Zugangsdaten” - Klassen-Verwaltung: Lehrer kann Passwörter zurücksetzen
20 min	**Scratch-Oberfläche kennenlernen** - Gemeinsame Erkundung der Bereiche - Blöcke-Kategorien verstehen - Erste Experimente mit der Katze	- Handbuch “Übersicht IDE” - Beamer für Live-Demo
20 min	**Erste Animation** - Schritt-für-Schritt “Tanze-Katze” - Grundlegende Konzepte erklären - Erste Debug-Erfahrungen	- Programmcode im Handbuch - Debug-Tipps griffbereit
15 min	**Freies Experimentieren** - Eigene Ideen umsetzen - Verschiedene Blöcke testen - Gegenseitige Hilfe fördern	- “NerdY Side Quests” - Zusatzaufgaben für schnelle Teams
10 min	**Abschluss & Ausblick** - Kurze Präsentationen - Erfolge würdigen - Hausaufgabe erklären	- Achievement-Sticker - Dokumentations-Seite im Handbuch

Differenzierung

- Grundstufe: Die Katze macht eine einfache Animation wenn die grüne Flagge geklickt wird.
- Mittelstufe: Die Katze reagiert auf Mausklicks mit verschiedenen Kostümen.
- Expertenstufe: Die Katze führt eine komplexe Animation mit mehreren Kostümen und Bewegungen aus.

Hausaufgabe “Scratch Safari”

Die Schüler erkunden die Scratch-Plattform und suchen sich drei interessante Projekte aus, die sie in der nächsten Stunde kurz vorstellen möchten. Sie sollen dabei besonders auf Dinge achten, die sie selbst gerne programmieren würden. Siehe auch *Seite 56* im Handbuch.

Die Schüler schreiben am besten die ID des Scratch-Projektes und den Namen des Projekts auf:

scratch.mit.edu/projects/**935749619** <- diese Zahl ist die eindeutige ID

Alternativ können sie die Projekte auch min einem Stern in ihrem Account als Lesezeichen speichern.

Ein paar didaktische Hinweise für den Unterricht:
Der erste Kontakt mit dem Programmieren sollte Spaß machen. Das Handbuch unterstützt Sie dabei mit seinem lockeren Ton und den spielerischen Elementen.

- Die “NerdY Side Quests” am Ende des Kapitels bieten zusätzliche Herausforderungen für schnellere Schüler. So können Sie gut differenzieren, ohne dass sich langsamere Schüler unter Druck gesetzt fühlen.
- Planen Sie am Ende der Stunde unbedingt Zeit für kurze Präsentationen ein. Dies motiviert die Schüler und gibt ihnen die Möglichkeit, voneinander zu lernen. Achten Sie darauf, dass die Präsentationen kurz und positiv bleiben.

Bei Problemen finden die Schüler Hilfe in dieser Reihenfolge:

1. Handbuch
2. Mitschüler
3. Lehrkraft

Weitere Infos und Beispiele

Abbildung 1: https://pad.kidslab.de/p/GamesLab-Movie

Doppelstunde 2: Katzenfreund & Sound Master

Ziel: Bewegungssteuerung und Sound-Effekte

Diese Doppelstunde markiert den Übergang von einfachen Animationen zu echten interaktiven Spielelementen. Die Schüler entwickeln ihr erstes kleines Spiel und lernen dabei, wie Sound das Spielerlebnis bereichert.

Zeit	Aktivität	Material & Hinweise
10 min	**Einstieg mit Projekt-Vorstellung** - Schüler zeigen Scratch-Projekte (Hausaufgabe) - Diskussion erfolgreicher Elemente - Überleitung zu Spielesteuerung	- Digitale Tafel für Präsentation - Vorbereitete Beispielprojekte als Backup - Beobachtungsbogen für gelungene Lösungen
15 min	**Einführung Bewegungssteuerung** - Demo des Koordinatensystems - Gemeinsames Programmieren einer Grundbewegung - Erklärung der Pfeiltasten-Steuerung	- Whiteboard für Koordinaten-Erklärung - Koordinaten-System Hintergrund zeigen - Code-Vorlagen im Handbuch - Debugging-Checkliste
25 min	**Katzenfreund Entwicklung** - Figurenauswahl und -anpassung - Implementation der Bewegungssteuerung - Erste Tests und Fehlerbehebung	- Handbuch Kapitel 3 - Debug-Tipps für Bewegung - Hilfestellung bei Bedarf
20 min	**Sound-Integration** - Einführung der Sound-Blöcke - Unterschied zwischen Sound-Befehlen erklären (siehe Handbuch, spiele Klang *ganz*)- Sound-Effekte einbauen	- Kopfhörer für alle - Sound-Bibliothek - Beispiele für gutes Audio-Feedback
15 min	**Test & Optimierung** - Gegenseitiges Testen der Spiele - Feedback sammeln - Verbesserungen einbauen	
5 min	**Abschluss & Ausblick** - Erfolgreiche Lösungen vorstellen - Achievement-System aktualisieren - Vorschau auf nächste Stunde	- Achievement-Sticker - Handbuch Achievements-Seiten

Differenzierung

- Grundstufe: Einfache Bewegung mit Pfeiltasten und ein Sound-Effekt.
- Mittelstufe: Geschwindigkeitsänderungen und verschiedene Sounds für verschiedene Aktionen.
- Expertenstufe: Komplexe Bewegungsmuster und Sound-Effekte die sich je nach Spielsituation ändern.

Hausaufgabe: “Sound Designer”

Die Schüler nehmen zu Hause drei eigene Geräusche auf (zum Beispiel mit dem Handy) und überlegen sich, wie sie diese in ihr Spiel einbauen könnten. Dies können Umgebungsgeräusche, selbst gemachte Effekte oder kurze Melodien sein.

Didaktischer Ansatz

Die Kombination von Bewegungssteuerung und Sound-Effekten ist bewusst gewählt. Die Schüler lernen zunächst, wie sie eine Figur mit den Pfeiltasten steuern können - ein fundamentales Konzept der Spieleentwicklung. Mit der Ergänzung von Sound-Effekten erleben sie direkt, wie audiovisuelles Feedback das Spielgefühl verbessert.

Wichtige Unterrichtshinweise

1. Bewegungssteuerung

    - Lassen Sie die Schüler erst die einfache Bewegung programmieren
    - Erklären Sie das Koordinatensystem anhand der Bühne
    - Weisen Sie auf häufige Fehler bei der Tastatursteuerung hin

2. Kollisionserkennung

    - Dies ist ein wichtiges neues Konzept - nehmen Sie sich Zeit dafür
    - Nutzen Sie die Debug-Tipps für typische Probleme
    - Lassen Sie die Schüler verschiedene Berührungsszenarien testen

3. Koordinatensystem

    Nutzen sie einen fertigen Hintergrund, um das Koordinaten-System einfacher erklären zu können:

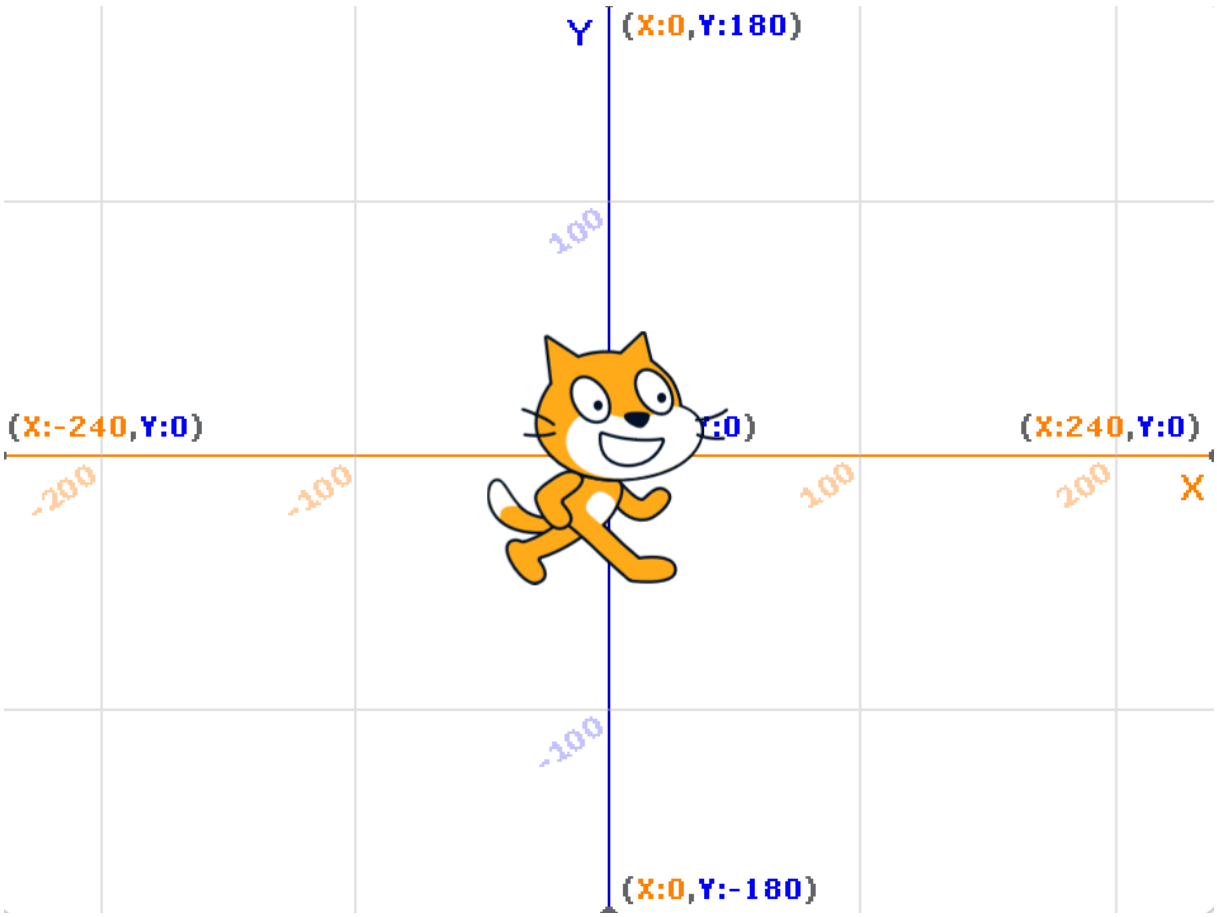

Abbildung 2: scratch-koordinaten

4. Sound-Integration
    - Stellen Sie sicher, dass alle Kopfhörer haben
    - Erklären Sie den Unterschied zwischen “spiele Klang” und “spiele Klang ganz”
    - Ermutigen Sie zum Experimentieren mit verschiedenen Sounds
5. Differenzierung
    - Schnellere Teams können die “NerdY Side Quests” angehen
    - Zusatzaufgabe: Eigene Sounds aufnehmen (Mikrofon vorhanden?)
    - Alternative Steuerungsmöglichkeiten entwickeln (Erweiterung Video)

Häufige Herausforderungen:

- Figuren verschwinden außerhalb der Bühne

- Timing-Probleme bei Sound-Effekten (Spiele Klang vs. spiele klang ganz)
- Ungewollte Endlosschleifen bei der Bewegung

Beachten Sie besonders: - Ausreichend Zeit für gegenseitiges Ausprobieren und Präsentation der Spiele - Dokumentation der Fortschritte im Achievement-System (Handbuch Seite 6-9)

Weitere Infos und Beispiele

Abbildung 3: https://pad.kidslab.de/p/GamesLab-Katzenfreund#

Doppelstunde 3: Cookie King - Einführung in Variablen und Spielbalancing

Ziel: Variablen und Spielmechaniken verstehen

Das Cookie King Kapitel ist der Übergang von einfachen Aktions-Reaktions-Spielen zu komplexeren Spielmechaniken mit Fortschrittssystem. Die Schüler lernen hier erstmals, wie sie Spielstände speichern und damit das Spielerlebnis über längere Zeit interessant gestalten können.

Zeit	Aktivität	Material & Hinweise
15 min	**Einführung in Variablen** - Box-Metapher für Variablen erklären (Seite 24)- Live-Demo einer Punktezählung - Gemeinsames Erstellen einer ersten Variable	- Handbuch mit Box-Metapher - Beispiel-Cookie-Clicker
25 min	**Grundmechanik entwickeln** - Cookie-Sprite erstellen und animieren - Klick-Mechanik programmieren - Punktezähler einrichten - Visuelle Feedback-Effekte	- Cookie-Vorlagen - Code-Beispiele im Handbuch - Animation-Tutorial
25 min	**Upgrade-System** - Erste Upgrade-Variable erstellen - Multiplikatoren implementieren - Automatische Klicker programmieren (Back-Oma)	- Upgrade-System Übersicht (Seite 28)
15 min	**Balancing-Workshop** - Spiele gegenseitig testen - Werte optimieren - Feedback sammeln und umsetzen	- Feedback-Bögen - Balance-Checkliste - Optimierungs-Tipps
10 min	**Präsentation & Reflexion** - Erfolgreiche Spiele vorstellen - Unterschiedliche Balance-Ansätze vergleichen - Beste Praktiken sammeln	

Beispiel: https://scratch.mit.edu/projects/1111026687/

Cookie Clicker - das sinnloseste Spiel der Welt?
Man kann sich echt wundern – warum sind Cookie Clicker Spiele so beliebt? Wir haben bei uns eine Umfrage unter Kindern das Workshops gemacht und alle haben gesagt, sie lieben Cookie Clicker. Was macht die Faszination aus? Probieren Sie es selbst mal aus: es ist eine entspannende Erfahrung. Man klickt immer nur, man muss sich nicht gross kümmern, aber man hat immer das Gefühl belohnt werden.

Differenzierung

- Grundstufe: Ein funktionierender Klick-Zähler mit einem einfachen Upgrade.
- Mittelstufe: Mehrere Upgrades mit unterschiedlichen Effekten. Expertenstufe:
- Komplexes Upgrade-System mit Auto-Clicker und Multiplikatoren.

Hinweis: Ermutigen Sie die Schüler, die Werte für ihr Spiel selbst auszubalancieren

Hausaufgabe: “Balance Master”

Die Schüler spielen verschiedene Clicker-Spiele (z.B. Cookie Clicker original) und notieren sich, was das Spiel motivierend macht. Sie sollen drei konkrete Ideen aufschreiben, wie sie ihr eigenes Spiel verbessern können.

NerdY Erkläromat: Was ist eigentlich Balancing? 🎮

Stell dir vor, du backst einen Kuchen: Zu viel Zucker macht ihn zu süß, zu wenig und er schmeckt fade. Genauso ist es bei Spielen - sie müssen perfekt “ausbalanciert” sein!
Ein Beispiel aus deinem Cookie Clicker:

- Wenn ein Upgrade nur 10 Cookies kostet aber 1000 neue Cookies pro Klick bringt = zu einfach = langweilig
- Wenn ein Upgrade 1.000.000 Cookies kostet aber nur 1 Cookie pro Klick bringt = zu schwer = frustrierend
- Wenn ein Upgrade 100 Cookies kostet und 5 Cookies pro Klick bringt = genau richtig = macht Spaß!

Die goldenen Regeln des Balancing:

- Jedes Upgrade sollte sich “lohnen” - aber nicht zu sehr
- Der Spieler soll immer ein Ziel vor Augen haben
- Schwieriger werden ist ok, unfair werden nicht
- Verschiedene Wege zum Ziel machen mehr Spaß

Denk dran: Ein perfekt ausbalanciertes Spiel fühlt sich an wie eine Treppe - jede Stufe ist ein bisschen höher, aber noch gut zu erreichen! 🎯

Didaktischer Ansatz:

Der Cookie Clicker ist ein ideales Projekt zur Einführung von Variablen, da die Schüler deren Notwendigkeit direkt nachvollziehen können: Ohne Variablen keine Punktezählung. Der “NerdY Fun-Fact” über den originalen Cookie Clicker zeigt den Schülern, dass auch sehr simple Spielkonzepte enormen Erfolg haben können, wenn sie gut umgesetzt sind.

Weitere Infos und Beispiele

Abbildung 4: https://pad.kidslab.de/p/GamesLab-CookieMaster

Doppelstunde 4: Dino Runner Grundlagen

Ziel: Komplexere Spielmechaniken

Diese Doppelstunde führt die Schüler an eine der wichtigsten Mechaniken der Spieleentwicklung heran: präzise Spielersteuerung und unmittelbares visuelles Feedback. Der Chrome-Dino dient als perfektes Beispiel für ein einfaches aber motivierendes Spielprinzip.

Zeit	Aktivität	Material & Hinweise
15 min	**Inspiration & Planung** - Demo des Chrome Dino - Gemeinsame Analyse der Sprungbewegung - Erstellen eines Entwicklungsplans	- Chrome Browser für Demo: https://chromedino.com/- Handbuch Kapitel 6 - Whiteboard für Bewegungsanalyse
30 min	**Grundlegende Sprungmechanik** - Variable “Sprungkraft” einführen - Aufwärtsbewegung programmieren - Schwerkraft implementieren - Boden-Kollision einbauen	- Code-Vorlagen im Handbuch - Debug-Tipps für Sprungverhalten - Wertetabelle für Feintuning
25 min	**Feintuning & Experimentieren** - Sprunghöhe anpassen - Fallgeschwindigkeit optimieren - Verschiedene Werte testen - Spielgefühl verbessern	- Experimentier-Protokoll - Verschiedene Testszenarien - Optimierungs-Checkliste
15 min	**Test & Verbesserung** - Gegenseitiges Testen - Feedback sammeln - Werte optimieren	- Feedback-Bögen - Vergleichswerte für gutes “Game Feel”
5 min	**Abschluss & Sicherung** - Erfolgreiche Werte notieren - Projekte speichern - Ausblick auf nächste Stunde	- Notizzettel für Erfolgswerte - Speicher-Checkliste

Hausaufgabe: “Game Research”

Die Schüler sollen den Chrome Dino und ähnliche Jump’n’Run Spiele spielen und analysieren: Wie hoch springt die Figur? Wie schnell bewegt sich der Hintergrund? Was macht das Spiel herausfordernd aber fair?

Differenzierung:

- Grundstufe: Funktionierender Sprung und einfache Hindernisse.
- Mittelstufe: Variable Geschwindigkeit und verschiedene Hindernistypen.
- Expertenstufe: Fortgeschrittene Mechaniken wie Doppelsprung oder Power-Ups.

Didaktischer Schwerpunkt:

Der Fokus dieser ersten Einheit liegt auf der Sprungmechanik. Dies ist bewusst gewählt, da hier die Schüler lernen, wie man Physik simuliert und gleichzeitig ein befriedigendes Spielgefühl erzeugt. Die Variable “Sprungkraft” macht abstrakte Physikkonzepte greifbar.

Zentrale Konzepte der Sprungmechanik:

Das Handbuch führt die Sprungmechanik über die “Sprungkraft”-Variable ein. Diese simuliert auf vereinfachte Weise die Schwerkraft: - Positiver Wert = Dino bewegt sich nach oben - Wert wird kontinuierlich kleiner = Dino verlangsamt sich - Negativer Wert = Dino fällt wieder - Bodenkontakt = Zurücksetzen der Sprungkraft

Häufige Herausforderungen: - Schüler vergessen die Sprungkraft zurückzusetzen - Der Sprung fühlt sich “schwammig” an - Dino bleibt am oberen Bildschirmrand hängen - Mehrfachsprünge sind möglich (kann als Feature oder Bug gesehen werden)

Tipps für gelungenes Springen:

- Sprungkraft-Startwert: etwa 10
- Schwerkraft (Verringerung): etwa -0.5 pro Schritt
- Bodenposition konstant halten
- Sprung nur erlauben, wenn Bodenkontakt besteht

Remember: Ein präzises Sprungsystem ist die Basis für ein motivierendes Spielerlebnis. Nehmen Sie sich die Zeit, bis es sich für jeden Schüler “richtig” anfühlt.

NerdY Erklärомat: Warum ist die Sprungmechanik so wichtig?
Stell dir vor, du spielst Basketball. Was macht mehr Spaß? - Ein Ball, der sofort dort landet, wo du hin zielst - Oder ein Ball, der sich wie ein echter Ball anfühlt, mit Schwung und Gewicht
Genauso ist es in Spielen! Ein guter Sprung sollte sich “echt” anfühlen: - Am Anfang schnell nach oben - Dann langsamer werden - Kurz “schweben” am höchsten Punkt - Und wieder nach unten fallen
Das nennt man “Game Feel” - und das macht den Unterschied zwischen einem okay Spiel und einem, das man stundenlang spielen kann! 🎮

Die nächste Einheit baut direkt auf dieser Grundmechanik auf. Ein gut funktionierender Sprung ist daher essentiell für den weiteren Erfolg des Projekts.

Weitere Infos und Beispiele

Abbildung 5: https://pad.kidslab.de/p/GamesLab-Dino

Beispiele:

- https://scratch.mit.edu/projects/1102220876/
- https://scratch.mit.edu/projects/1120295179/

Doppelstunde 5: Dino Runner Finalisierung & Effekte

Ziel: Spiel vervollständigen und mit Effekten aufwerten

In dieser Doppelstunde verwandeln wir die Sprungmechanik in ein vollwertiges Spiel. Die Schüler lernen, wie man durch geschicktes Kombinieren einfacher Elemente ein spannendes Spielerlebnis erschafft und dieses dann mit visuellen Effekten aufwertet.

Zeit	Aktivität	Material & Hinweise
15 min	**Einstieg & Wiederholung** - Präsentation funktionierender Sprungmechaniken - Kurze Erklärung des Konzepts von Klonen - Vorstellung der Paralax-Demo	- Digitale Tafel für Demo - Beispielprojekt mit Paralax-Effekt
30 min	**Grundmechaniken** - Kaktus-Klone erstellen - Bewegung nach links programmieren - Kollisionserkennung einbauen - Game Over implementieren	- Handbuch Kapitel 6 - Debug-Tipps für Kollisionen - Checkliste für Klon-Verhalten
25 min	**Differenzierte Arbeitsphase** *Basis:* - Feintuning der Grundmechanik *Fortgeschritten:* - Steine am Boden implementieren *Experten:* - Wolken mit Paralax-Effekt	- Zusatzmaterial für Paralax - Vorlagen für verschiedene Kakteen - Beispiele für Bodendekoration
10 min	**Test & Debug** - Gegenseitiges Testen - Fehlersuche & Optimierung - Anpassung der Schwierigkeit	- Testprotokoll im Handbuch - Debug-Checkliste
10 min	**Abschluss & Ausblick** - Kurzpräsentationen - Sammeln von Verbesserungsideen - Vorschau auf Effect Wizard	- Achievement-Sticker - Notizzettel für Ideen

Hinweis: Flexibel Zeit für individuelle Hilfestellung einplanen, besonders bei der Kollisionserkennung und dem Paralax-Effekt.

Hausaufgabe: “Speed Runner”

Die Schüler dokumentieren ihre persönlichen Highscores im Chrome Dino und anderen Jump’n’Run Spielen. Sie sollen analysieren, was den Unterschied zwischen einem guten und einem perfekten Run ausmacht.

Differenzierung für “Dynamische Spielwelt”:

1. Wolken-Paralax:
    - Erstelle verschiedene Wolkenformen
    - Lass sie in unterschiedlichen Geschwindigkeiten fliegen
    - Je weiter hinten, desto langsamer!
2. Bodendekoration:
    - Füge verschiedene Steine hinzu
    - Lass sie zufällig erscheinen
    - Experimentiere mit verschiedenen Größen
3. Kaktus-Variation:
    - Erstelle verschiedene Kaktus-Arten
    - Lass sie zufällig erscheinen
    - Mache größere Kakteen schwieriger zu überspringen
4. Vögel:
    - im Original kommen ab 200 Punkte auch Vögel, bei denen sich der Dino ducken muss
    - Abwechselnd mit den Kakteen
    - Andere Taste, um sich zu ducken

NerdY Extra Side Quest: Weather Wizard 🌤️

Baue ein dynamisches Wettersystem:

- Wolken ziehen unterschiedlich schnell
- Regentropfen fallen bei schlechtem Wetter
- Blitze erhellen kurz den Hintergrund
- Wetterwechsel beeinflusst die Spielgeschwindigkeit

Didaktischer Schwerpunkt:

Der Fokus liegt auf dem Konzept der Klone - einer mächtigen Scratch-Funktion, die es erlaubt, Spielobjekte dynamisch zu erzeugen und zu steuern. Die Schüler lernen dabei auch das wichtige Konzept der Kollisionserkennung.

Kernkonzepte:

1. Hindernisse mit Klonen
    - Der Original-Kaktus ist die “Fabrik”, die Klone erzeugt
    - Klone bewegen sich automatisch nach links
    - Klone löschen sich selbst am linken Bildschirmrand
2. Kollisionserkennung
    - Prüfen, ob der Dino den Kaktus berührt
    - Game Over bei Berührung auslösen
    - Neustart ermöglichen
3. Visuelle Verbesserungen (Differenzierung)
    - Steine am Boden für mehr Atmosphäre
    - Wolken mit Paralax-Effekt für Tiefenwirkung
    - Verschiedene Kaktus-Varianten

Diese Doppelstunde ist entscheidend für das Spielgefühl. Ein gut getimter Kaktus-Spawn und präzise Kollisionserkennung machen den Unterschied zwischen frustrierend und motivierend.

Ermutigen Sie die Schüler, viel zu experimentieren und das Feedback ihrer Mitschüler einzuholen.

Doppelstunde 6: Effekte

Ziel: Spiele erweitern und verbessern

Diese Doppelstunde lehrt nicht nur technische Fähigkeiten, sondern entwickelt auch ein Gespür für ästhetisches Design und Spielgefühl. Die Schüler lernen, wie kleine visuelle Details große Wirkung haben können.

Die verschiedenen Effekte bauen aufeinander auf und können je nach Fähigkeitsniveau der Schüler angepasst werden. Der modulare Aufbau ermöglicht es, einzelne Effekte zunächst einfach zu implementieren und später zu verfeinern.

Das abschließende Einbauen der Effekte in die eigenen Spiele gibt den Schülern die Möglichkeit, das Gelernte direkt praktisch anzuwenden und die Verbesserung ihrer Spiele zu erleben.

Zeit	Aktivität	Material & Hinweise
15 min	**Einführung in Spieleffekte** - Demonstration verschiedener Effekte - Diskussion: Warum machen Effekte Spiele besser? - Analyse von Beispielen	- Effekt-Demo-Projekt - Handbuch “Effect Wizard” - Videoclips von Spieleffekten
25 min	**Paralax-Effekt** - Konzept der Bewegungsebenen - Implementation von Wolkenschichten - Geschwindigkeiten abstimmen - Feintuning der Bewegung	- Paralax-Tutorial im Handbuch - Fertige Grafiken für Wolken - Debug-Tipps für Bewegung
20 min	**Regenbogen-Power** - Partikel-System verstehen - Klone für Effekte nutzen - Farbverläufe programmieren - Animation optimieren	- Code-Vorlagen - Beispiele für Partikeleffekte - Performance-Checkliste
20 min	**Sparkle-Effekt** - Lokale Variablen für Bewegung - Zufallsgesteuerte Partikel - Timing und Lebensdauer - Effekt-Trigger einbauen	- Sparkle-Tutorial - Vorlagen für Partikel - Mathematische Grundlagen
10 min	**Integration & Präsentation** - Effekte in eigene Spiele einbauen - Ergebnisse vorstellen - Feedback und Optimierung	- Integrations-Checkliste - Achievement-Sticker - Präsentations-Leitfaden

Hausaufgabe: “Effect Hunter”

Die Schüler sollen in ihren Lieblingsspielen nach coolen visuellen Effekten suchen und diese mit dem Handy aufnehmen. In der nächsten Stunde analysieren wir gemeinsam, wie man diese Effekte in Scratch nachbauen

könnte.

Differenzierung

- Grundstufe: Einfacher Paralax-Effekt mit zwei Ebenen.
- Mittelstufe: Regenbogen-Trail und einfache Partikeleffekte.
- Expertenstufe: Komplexe Partikelsysteme mit verschiedenen Verhaltensweisen.

Konzeptverständnis für jede Phase:

Einführungsphase: - Effekte als Mittel zur Spielverbesserung verstehen - Unterscheidung verschiedener Effekttypen - Bedeutung von visuellem Feedback

Paralax-Implementation: - Verständnis für Bewegungsebenen - Geschwindigkeitsverhältnisse - Endlosschleifen für Hintergründe

Partikel-Systeme: - Klone als Grundlage für Partikel - Bewegungsmathematik - Ressourcenmanagement

Weitere Infos und Beispiele

Weitere **Effekte** und wie sie funktionieren findet sie hier:

Scratch-Lehreraccount

Als Lehrkraft können Sie einen speziellen “Educator-Account” bei Scratch erstellen, der Ihnen zusätzliche Funktionen für den Unterricht bietet.

Abbildung 6: Scratch Logo

Vorteile des “Educator-Accounts”

- Klassen/Kurse anlegen und verwalten
- Schülerkonten ohne E-Mail-Adressen erstellen
- Schülerprojekte und -aktivitäten überwachen
- Vereinfachte Passwort-Verwaltung
- Zugriff auf spezielle Unterrichtsmaterialien

Weitere Informationen finden Sie hier: https://scratch.mit.edu/educators/.

So richten Sie einen Lehrkräfte-Account ein

1. Besuchen Sie: https://scratch.mit.edu/educators/register
2. Füllen Sie das Formular mit Ihren Daten aus:
    - Name und E-Mail
    - Schulname und -adresse
    - Kurze Beschreibung Ihrer Lehrtätigkeit
3. Sie erhalten eine Mail mit einem Link, den sie zur Bestätigung klicken
4. Nach der Verifizierung (ca. 1 Tag) erhalten Sie Zugang zu den zusätzlichen Funktionen

Klasse und Klassenlink erstellen

Legen Sie vor dem GamesLab-Workshop eine Klasse in Ihrem Account an - so haben Sie später die Kontrolle über die Schüler-Accounts und können z.B. verlorene Passwörter zurücksetzen.

Sollten Sie Schwierigkeiten haben, können wir Ihnen auch am Workshop-Tag dabei helfen.

So gehen sie vor:

1. Melden Sie sich mit ihrem Lehrkraft-Account bei Scratch an: https://scratch.mit.edu

2. Klicken sie auf “meine Klassen” und erstellen Sie eine neue Klasse

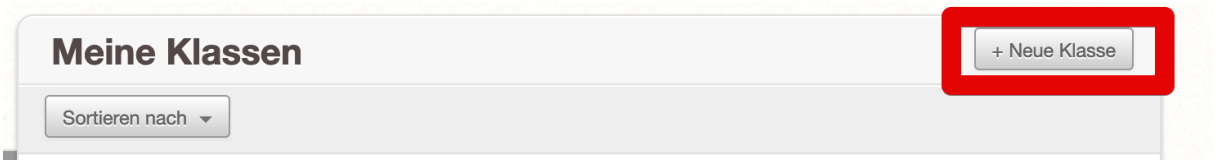

Abbildung 7: Neue Klasse anlegen

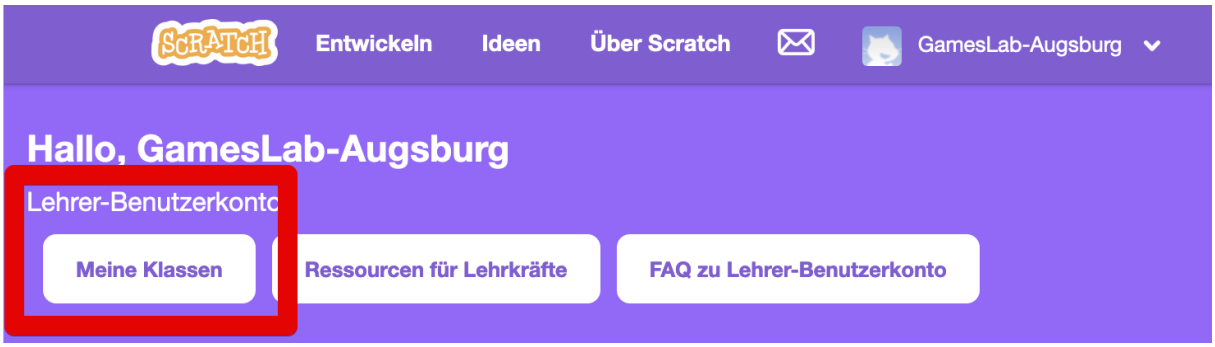

Abbildung 8: Meine Klassen

3. Erstellen Sie den Schüler-Registrierungslinks

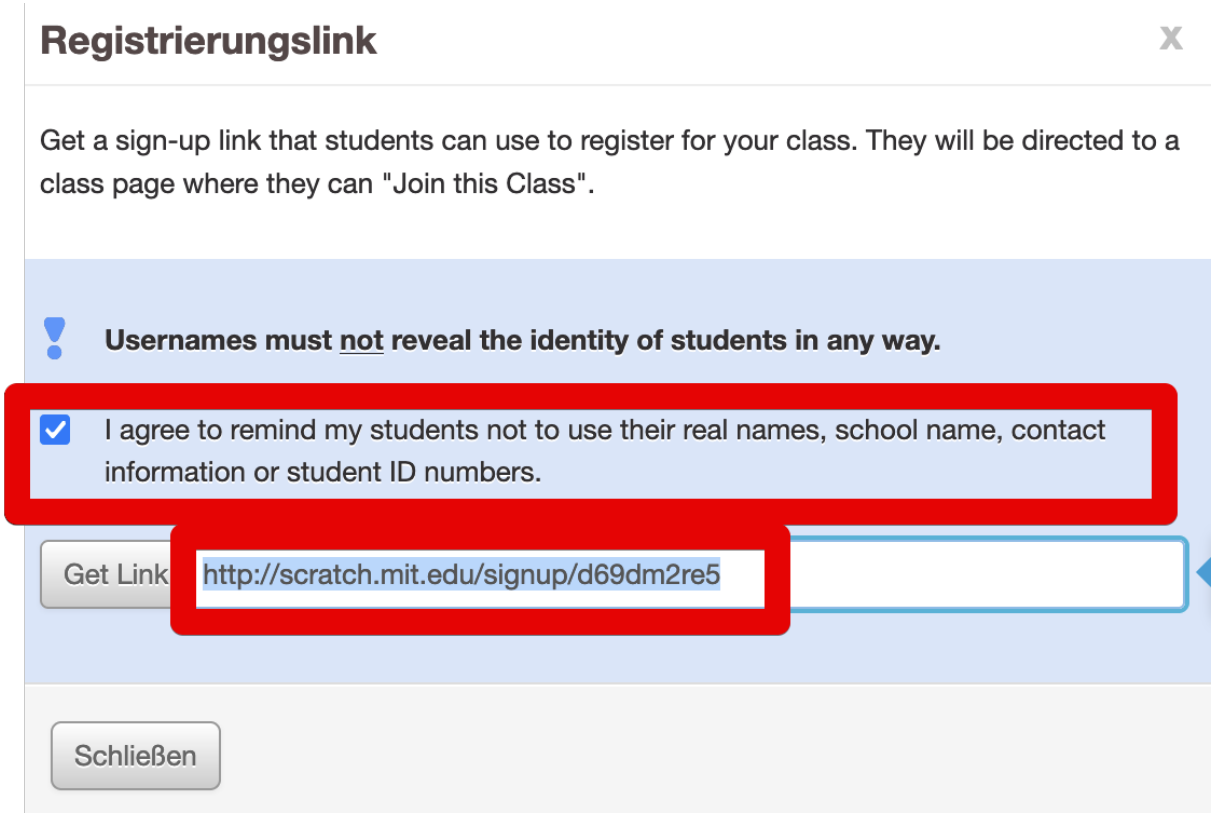

Abbildung 9: Registrierungslink

4. Kopieren Sie den Link und mailen diesen an: team@kidslab.de

Weiterführende Informationen und Angebote für Lehrkräfte

Raspberry Pi Foundation

- Umfangreiche Scratch-Projektsammlung in deutscher Sprache
- URL: https://projects.raspberrypi.org/de-DE/collections/scratch
- Ideal für strukturiertes Lernen mit steigendem Schwierigkeitsgrad
- Enthält detaillierte Schritt-für-Schritt Anleitungen

Google CS-First

- Kostenloser Informatik-Kurs für Schüler*innen und Lehrkräfte
- URL: https://csfirst.withgoogle.com/c/cs-first/de/curriculum.html
- Spezieller Game Design Kurs: https://csfirst.withgoogle.com/c/cs-first/de/game-design/overview.html

Besondere Merkmale:

- Etwa 1h Vorbereitungszeit pro Unterrichtseinheit
- Keine Anmeldung erforderlich
- Ausdruckbare Unterrichtsmaterialien verfügbar
- Verschiedene Spielkonzepte (Rennspiele, Fangspiele, etc.)
- Material zum Ausdrucken: https://csfirst.withgoogle.com/c/cs-first/de/game-design/materials.html

Die Schüler können die Einheiten auch selbständig mit den Videos im Kurs machen oder zusammen mit einer Lehrkraft.

Eine **Anmeldung** als Lehrkraft oder Schüler ist möglich, aber **optional**: Sie können auf alle Kursinhalte auch ohne ein Google-Konto zugreifen und sie benutzen.

Es gibt neben den Online-Inhalten auch viel Material zum Ausdrucken und direktem Einsatz im Unterricht: https://csfirst.withgoogle.com/c/cs-first/de/game-design/materials.html

GamesLab: Curriculare Einordnung und Kompetenzen

Besondere Merkmale:

- Spiralcurriculum: Konzepte werden wiederholt und vertieft
- Differenzierung durch Schwierigkeitsgrade (⭐ bis ⭐⭐⭐⭐⭐)
- Integrierte Debug-Hilfen und Troubleshooting-Guides
- Selbstständiges Lernen durch klare Dokumentation
- Motivierende Achievements und Fortschrittssystem
- QR-Codes für zusätzliche Online-Ressourcen

Methodische Hinweise:

- Projekt- und handlungsorientierter Ansatz
- Individuelles Lerntempo möglich
- Peer-Learning durch Partnerarbeit
- Unmittelbare Erfolgserlebnisse
- Kreative Freiräume in jedem Kapitel

Kapitel	Technische Konzepte	Informatische Kompetenzen	Methodik & Didaktik	Fächerübergreifende Aspekte
1. Movie Director	- Grundlagen der Entwicklungsumgebung- Event-Handler- Einfache Animationen	- Algorithmisches Denken- Sequenzierung von Befehlen- Grundlegende Programmierprinzipien	- Niedrigschwelliger Einstieg- Visuelle Erfolgserlebnisse- Kreatives Arbeiten	- Medienkompetenz- Kunst (Animation)- Kreatives Gestalten
2. Katzen Freund	- Tastatureingaben- Bewegungssteuerung- Kollisionserken- nung	- Ereignissteuerung- Bedingte Anweisungen- Koordinatensystem	- Spielerisches Lernen- Trial & Error- Problemlösung	- Physik (Bewegung)- Mathematik (Koordinaten)

Kapitel	Technische Konzepte	Informatische Kompetenzen	Methodik & Didaktik	Fächerübergreifende Aspekte
3. Sound Master	- Audio-Integration- Event-Sound-Synchronisation- Mehrkanal-Audio	- Multimedia-Integration- Timing und Synchronisation- Ereignissteuerung	- Multimediales Lernen- Auditive Wahrnehmung- Kreative Expression	- Musik- Physik (Schall)- Mediengestaltung
4. Cookie King	- Variablen- Arithmetische Operationen- Zähler und Timer	- Datentypen und -strukturen- Mathematische Modellierung- Wirtschaftssimulation	- Spieleökonomie- Balancing- Systematisches Denken	- Mathematik- Wirtschaft- Statistik
5. Dino Runner	- Gravitation & Sprungmechanik- Endlosschleifen- Kollisionserkennung	- Physikalische Simulation- Komplexe Bedingungen- Spielemechanik	- Iteratives Entwickeln- Gamedesign-Grundlagen- Motivation durch Herausforderung	- Physik- Gamedesign- Spieltheorie
6. High-Score Master	- Datenbanken (Cloud-Variablen)- Persistente Speicherung- Datenverwaltung	- Datenmanagement- Client-Server-Konzepte- Netzwerkkommunikation	- Wettbewerbselemente- Soziales Lernen- Leistungsmotivation	- Mathematik (Statistik)- Sozialwissenschaften- Datenschutz
7. Effect Wizard	- Partikelsysteme- Komplexe Animationen- Shader & Effekte	- Modularisierung- Optimierung- Parallele Prozesse	- Kreatives Problemlösen- Ästhetische Gestaltung- Experimentelles Lernen	- Kunst- Physik (Optik)- Mediendesign
8. Intro Designer	- Szenenmanagement- UI/UX-Design- Mehrstufige Animationen	- User Interface Design- Storyboarding- Mediengestaltung	- User Experience- Narrative Gestaltung- Ästhetisches Empfinden	- Deutsch (Narration)- Kunst- Mediengestaltung

Kapitel	Technische Konzepte	Informatische Kompetenzen	Methodik & Didaktik	Fächerübergreifende Aspekte
9. Extension King	- Externe Bibliotheken - API-Integration - Modularisierung	- Systemintegration - Erweiterbarkeit - Bibliotheksnutzung	- Ressourcenmanagement - Modulares Denken - Technologieintegration	- Informatik-Systemarchitektur - Modularität
10. Game Master	- Projektmanagement - Komplexe Spielemechaniken - Code-Organisation	- Projektplanung - Qualitätssicherung - Debugging	- Selbstständiges Arbeiten - Projektmanagement - Problemlösung	- Fächerübergreifende Integration - Projektmanagement - Kreativität

Glossar für Lehrkräfte - Scratch und Programmierung

Debug-Konsole
Ein Werkzeug, das Programmierer nutzen, um Fehler zu finden. Zeigt Meldungen über den Programmablauf.

IDE (Integrated Development Environment)
Die Programmierumgebung - bei Scratch ist das die Webseite mit Bühne, Codeblöcken und Figuren.

Loop (Schleife)
Ein Programmteil, der mehrmals wiederholt wird, z.B. “wiederhole 10 mal” oder “wiederhole fortlaufend”.

Open Source
Software, deren Quellcode öffentlich ist und von allen genutzt werden kann. Scratch ist Open Source.

Quellcode
Die Anweisungen eines Programms - bei Scratch sind das die zusammengesteckten Blöcke.

Repository
Ein Speicherort für Programm-Code, oft online. Scratch-Projekte werden im Scratch-Repository gespeichert.

Runtime
Die Laufzeit eines Programms - der Moment, wenn das Programm tatsächlich ausgeführt wird.

Script
Eine Folge von Programmanweisungen - bei Scratch eine Reihe verbundener Blöcke.

Syntax
Die Regeln, wie Programmanweisungen aufgebaut sein müssen. Bei Scratch durch die Blöcke vorgegeben.

UI (User Interface)
Die Benutzeroberfläche - alles was man sieht und womit man interagieren kann.

Achievement
Ein Belohnungssystem für erreichte Lernziele, ähnlich wie Abzeichen oder Sterne.

Bug
Ein Fehler im Programm. Zum Beispiel wenn die Spielfigur nicht wie gewünscht reagiert.

Debugging
Die systematische Suche und Behebung von Programmfehlern.

Effekte
Visuelle oder akustische Elemente, die das Spiel lebendiger machen (z.B. Explosionen, Glitzer).

Game Feel
Beschreibt, wie sich ein Spiel “anfühlt” - ob Bewegungen und Reaktionen natürlich und angenehm wirken.

Klone

Kopien von Figuren, die das Programm automatisch erstellt und wieder löscht (z.B. für Hindernisse oder Effekte).

Koordinatensystem

Das Raster auf der Bühne, mit dem die Position von Figuren bestimmt wird.

Pair Programming

Zwei Schüler arbeiten gemeinsam an einem Computer - einer tippt, einer beobachtet und gibt Hinweise.

Paralax-Effekt

Verschiedene Bildebenen bewegen sich unterschiedlich schnell, um Tiefe zu erzeugen (z.B. Wolken und Berge).

Sprite

Eine Figur oder ein Objekt in Scratch, das programmiert werden kann.

Variable

Ein “Behälter” für Zahlen oder Text, z.B. für Punktestände oder Namen.

Debugging-Tipps für dein Scratch-Projekt

Bugs? No Problem! 🐛

Hey Gamedev! Dein Spiel macht nicht das, was es soll? Die **Katze** dreht sich in die falsche Richtung, der **Punktezähler** spinnt oder dein **Dino** schwebt plötzlich im Weltraum? Willkommen im Club!

Jeder - wirklich **JEDER** - Programmierer kämpft mit kleinen und großen **Fehlern** im Code. Das ist völlig normal und gehört zum **Spieleentwickeln** dazu wie Pommes zu Ketchup! Die gute Nachricht: Mit ein paar coolen **Tricks** und Kniffen findest du fast jeden **Bug**.

In diesem Kapitel zeige ich dir, wie du deinen **Code** auf Vordermann bringst und die häufigsten **Probleme** ganz easy löst. Und das Beste: Mit jeder **Fehlersuche** lernst du etwas Neues und wirst ein besserer Programmierer!

Also: Lass uns auf **Bug-Jagd** gehen! 🔍

1. Laut vorlesen

Lies deinen **Code** laut vor und denk dabei wie ein **Computer**. Sind alle **Schritte** wirklich im **Code**? Sind die **Anweisungen** klar?

2. In Teile zerlegen

Teile große **Programme** in kleine Häppchen auf. Teste jedes **Teil** einzeln und füge sie dann wieder zusammen.

3. Verlangsamen

Füge "**warte**"-Blöcke ein, um zu sehen, was genau passiert. Entferne sie wieder, wenn alles läuft.

4. Sound-Checkpoints

Baue **Sounds** als Kontrollpunkte ein. Kein **Sound** = Fehler davor, **Sound** = Fehler danach.

5. Block-Reihenfolge checken

- Was muss zuerst passieren?
- Was kommt danach?
- Muss etwas zurückgesetzt werden?

6. Schleifen prüfen

Checke deine “**wiederhole**”- und “**fortlaufend**”-Blöcke:

- Sollen wirklich alle **Blöcke** in der **Schleife** sein?
- Fehlt ein “**warte**”-Block?
- Braucht es überhaupt eine **Schleife**?

7. Timing beachten

Wenn mehrere Dinge gleichzeitig passieren sollen, kann es **Chaos** geben. Baue kleine **Pausen** oder **Klick-Events** ein.

8. Mach Pausen!

Manchmal hilft es, einfach mal 5 Minuten vom **Computer** wegzugehen. Mit frischem **Kopf** sieht man **Fehler** oft sofort!

Remember: Jeder Programmierer macht **Fehler** - das Geheimnis ist, sie zu finden! 🐞✨

Nutze den Telefon-Joker! Auch für Lehrkräfte :)

Alles ausprobiert, aber es geht immer noch nicht?

- **Veröffentliche** dein Projekt
- Schicke uns die **URL** des Projekts
- **Foto** reicht auch
- Und natürlich: Was dein Bug oder Problem ist?

Entweder per **Mail**: **team@kidslab.de**

Oder per **WhatsApp**:

Abbildung 10: whatsapp