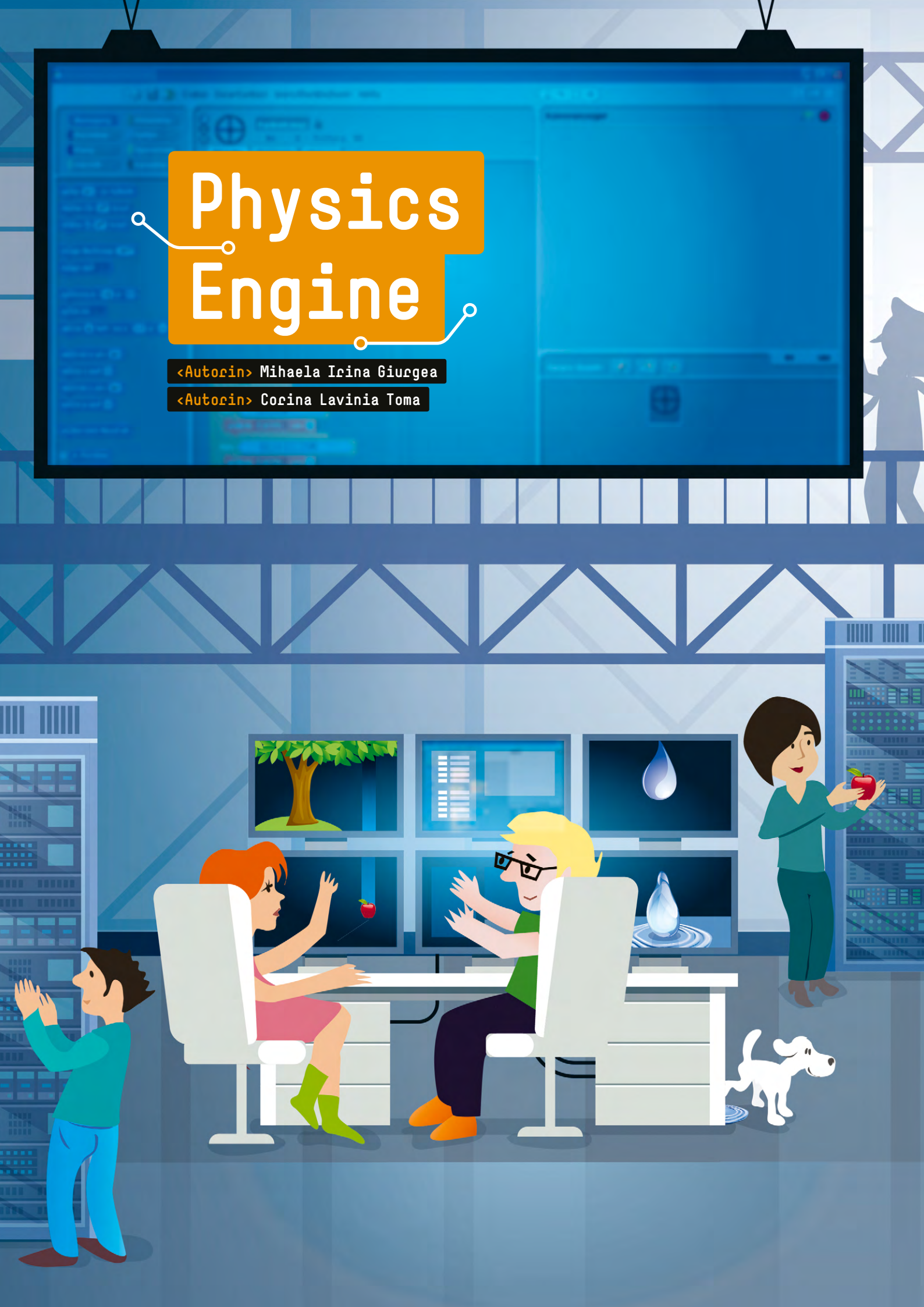
Physics Engine

<Autorin> Mihaela Irina Giurgea
<Autorin> Corina Lavinia Toma



<Info>

<Schlagwörter> Animation, Sprite, Blöcke, Schleifen, Grafiken, Gravitation, elastischer Stoß, freier Fall, Reibungskraft, schräger Wurf, Bewegung, Impuls, Operatoren, Variablen

<Unterrichtsfächer> Informatik, Physik, Mathematik, IKT

<Altersgruppe> 14–16 Jahre

<Hardware> Computer

<Programmiersprache> Scratch[1]

<Programmierniveau> leicht, mittel

<Zusammenfassung>

Was würden Sie denken, wenn wir Ihnen sagen, dass Schülerinnen und Schüler die zwei scheinbar sehr unterschiedlichen Fächer Physik und Informatik leichter und dabei auch noch gleichzeitig erlernen könnten? In dieser Einheit ist die „Engine" (Motor) – die Scratch-Programmierumgebung[1] – das magische Werkzeug, das Schülerinnen und Schüler darin unterstützt, interessante Programme zu alltäglichen Naturphänomenen zu entwickeln, um die Folgen der physikalischen Gesetze zu verstehen und ihre Programmierkenntnisse zu verbessern.

<Vorstellung des Konzepts>

Warum verwenden wir Scratch[1]? Scratch ist eine visuelle Programmierumgebung, die Figuren („Sprites") mit Blöcken auf dem Computerbildschirm animiert und es ermöglicht, Programme einfacher als mit klassischen Umgebungen (C++, Java usw.) zu entwickeln. Zudem hilft unsere „Engine" den Lehrkräften zwei Fächer zu unterrichten, die als schwierig empfunden werden: Physik und Informatik. Indem wir die Haupthindernisse, d. h. die Unmöglichkeit, sich vorzustellen (zu sehen), wie ein Phänomen eigentlich funktioniert, und die komplizierte Programmiersyntax, beseitigten, schaffen wir eine angenehme und interessante Lernumgebung.

Da die Schülerinnen und Schüler, die an der Entwicklung dieser Unterrichtseinheit beteiligt waren, schon Anfängerkenntnisse in der Programmierung besaßen, fanden sie selbst heraus, wie sie mit Scratch arbeiten können, was sowohl im regulären als auch im freiwilligen Informatikunterricht stattfand. Die erforderlichen Physikkenntnisse sind Teil des regulären Lehrplans, und es ist immer hilfreich, Gelerntes zu wiederholen und anzuwenden.

<Praktische Umsetzung>

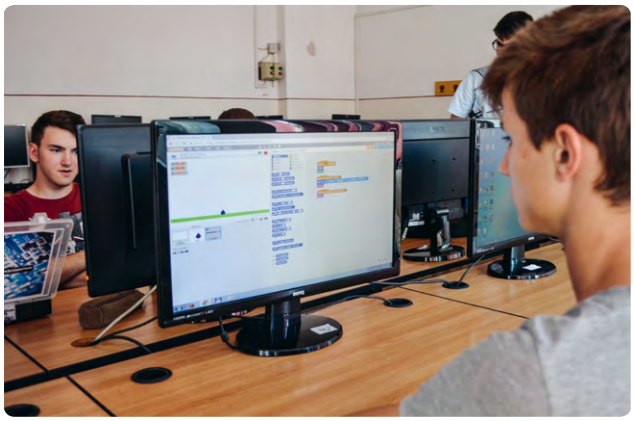
Die Einheit besteht aus alternierenden Lernsequenzen in Programmierung und Physik.

Zuerst präsentierte die Informatiklehrkraft die Grundlagen einer Projekterstellung in Scratch[1]. Die Schülerinnen und Schüler machten sich mit verschiedenen Schlüsselbegriffen der Scratch-Umgebung vertraut: Bühne („Stage"), Figuren („Sprites"), Kostüme („Costumes"), Bewegung („Movement"). Die Anleitungen und Erläuterungen zum ersten mit Scratch gelehrten Programm ohne physikalische Formel sind online verfügbar.[2]

Die Schülerinnen und Schüler müssen die Interaktionen zwischen den Sprites und ihre Synchronisation verstehen, aber auch, wie ein Koordinatensystem funktioniert. Ein komplettes Tutorial zu Scratch ist online verfügbar.[3] Für ein besseres Verständnis der Hauptalgorithmen von Scratch schauten sich die Schülerinnen und Schüler einige interessante Beispiele an und waren oft erstaunt, wie einfach die dahinterliegende Programmierung ist.

Im Physikteil wandten sie die theoretischen Grundlagen, auf denen die Phänomene ihrer Umwelt basieren, an. So schlug die Physiklehrkraft viele Themen[4] vor, die anschließend diskutiert wurden: erforderliche Formeln, mögliche Animationen, das Design usw. Dann wählten die Schülerinnen und Schüler Themen aus und nach einer Woche präsentierten sie Apps zum schrägen Wurf eines Balls, zum freien Fall eines Apfels, zum komplexeren Fall eines Wassertropfens oder zur Kollision zweier Bälle, aber auch zu Planetenbewegungen im Sonnensystem und sogar kleine Computerspiele.

Zu Beginn arbeiteten die Schülerinnen und Schüler weitgehend selbstständig. Wenn ein Projekt verbessert werden musste, erhielten sie Unterstützung von den Lehrkräften und dem Rest der Klasse. (📷 1)



📷 1: Selbstständiges Arbeiten

Danach stellten alle ihre Projekte vor. Dank der Rückmeldungen fiel es den Schülerinnen und Schülern leichter zu erkennen, welche Aspekte noch verbessert werden mussten: die Programmierung oder der Physikteil.

Am Ende unseres Projekts wurden die Schülerinnen und Schüler zu Lehrkräften für die Jüngeren (12–13 Jahre alt), indem sie geeignete Programme vorstellten und während des Physikunterrichtes testeten. Es machte ihnen Spaß, in der Rolle der Lehrkraft zu sein, und sie waren sehr stolz auf ihre Arbeit. Zudem erhielten sie von den Jüngeren Vorschläge und Anregungen. Die besten Simulationen der Schülerinnen und Schüler sind auf der Scratch-Plattform abrufbar.[2]

Nachfolgend präsentieren wir einige Beispiele zu Ansätzen wie Physik mit Programmieren verbunden werden kann.

<Programm 1: Freier Fall (Newtons Apfel)>

Vom freien Fall dieses historischen Objektes hat jeder schon einmal gehört: des Apfels des Isaac Newton. Das Programm in 📷 2 ist von einer klassischen Aufgabe inspiriert: Welche Distanz legt Newtons Apfel im freien Fall pro Sekunde zurück?

Physikalische Theorie

Annahme: Eine lineare Bewegung erfolgt mit der konstanten Erdbeschleunigung $g = 9{,}8\ \frac{\mathrm{m}}{\mathrm{s}^2}$.

Nach einer Zeit t beträgt die zurückgelegte Distanz $h(t)$ des Apfels: $h(t) = \frac{gt^2}{2}$.

Der Ausgangspunkt $h(0)$ befindet sich am Ast des Baumes, von dem sich der Apfel gelöst hat.

Dann berechnen wir die zurückgelegte Distanz über einen längeren Zeitraum $t + \Delta t$:

$$h(t+\Delta t) = \frac{g(t+\Delta t)^2}{2}.$$

Die allgemeine Formel für die Distanz $\Delta h(t)$, d. h. die Strecke, die der Apfel in der Zeit Δt zurücklegt, lautet:

$$\Delta h(t) = h(t+\Delta t) - h(t) = \frac{g(2t\Delta t + \Delta t^2)}{2}.$$

In unserem spezifischen Fall gilt: $\Delta t = 1$ s. In der ersten Sekunde folgt aus $t = t_{\mathrm{in}} = 0$, dass $\Delta h_1 = 4{,}9$ m, in der nächsten Sekunde t=1s führt zu $\Delta h_2 = 3 \cdot 4{,}9\ \mathrm{m} = 14{,}7$ m usw. Durch mathematische Induktion können wir die Berechnung für die *n-te* Sekunde und $t = (n-1)$ s vornehmen:

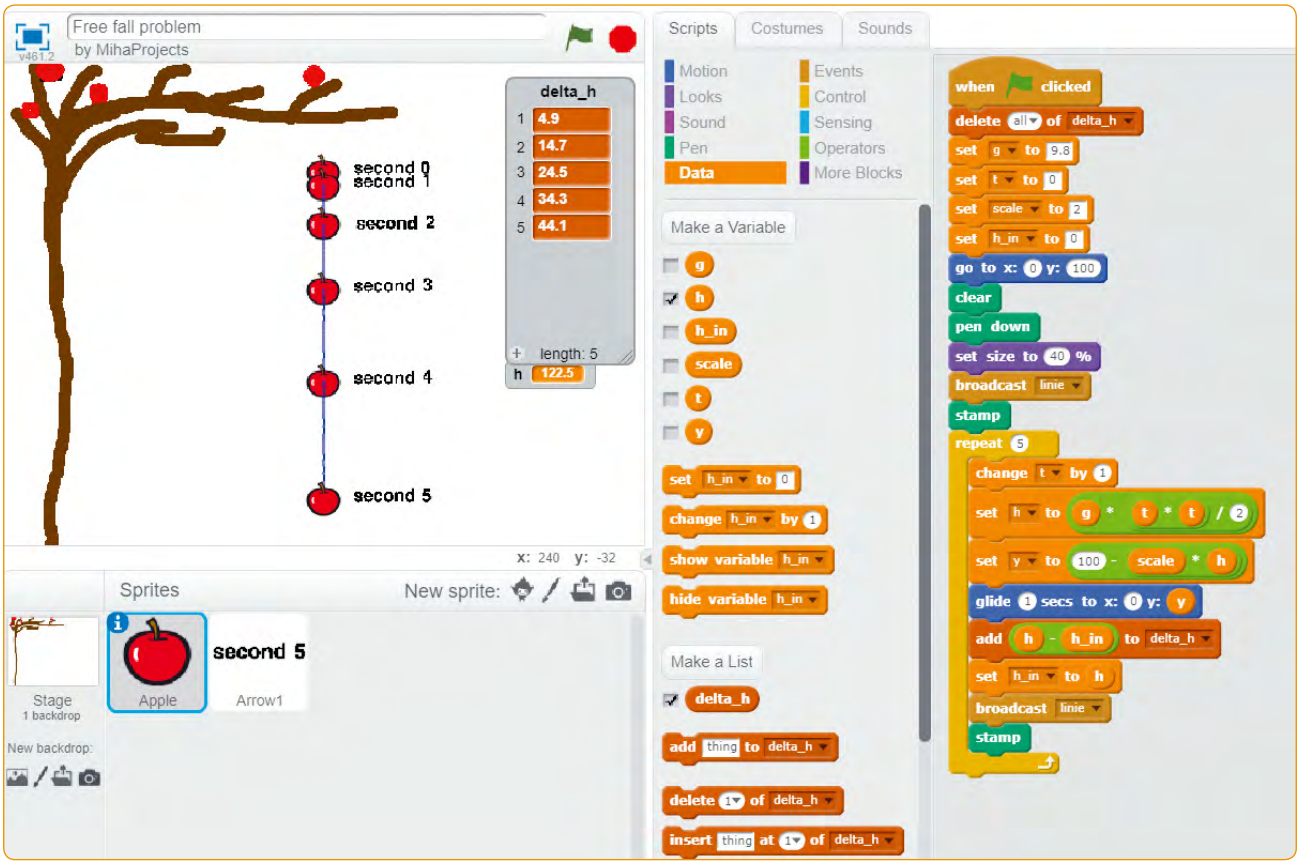
$$\Delta h_n = \frac{9{,}8(2n-1)}{2}\ \mathrm{m}.$$

Nun können die Schülerinnen und Schüler berechnen und in unserer Animation auch sofort sehen, dass sich die zurückgelegte Distanz in jeder Sekunde stets um den gleichen Betrag erhöht: 9,8 m.

Wie programmiert man das?

Verwendete Variablen:

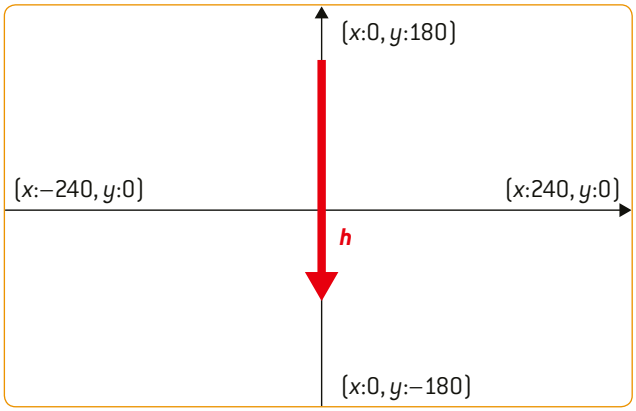
g: Erdbeschleunigung



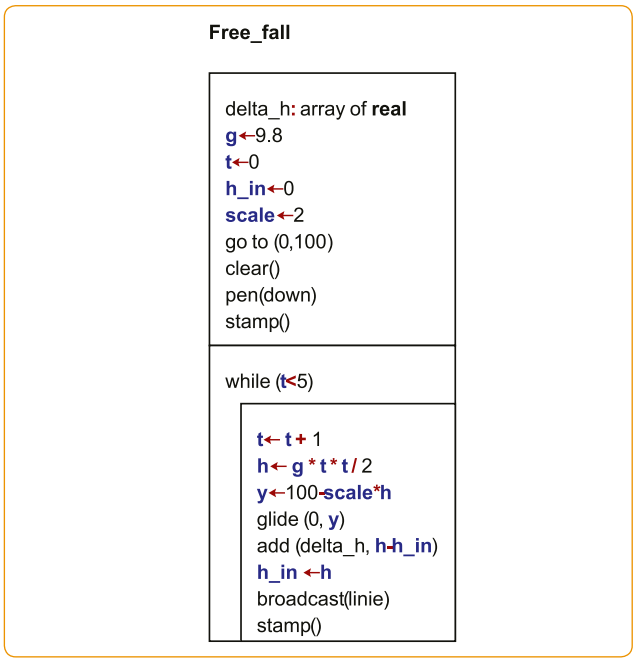
📷 2: Freier Fall

t: Zähler für Sekunden (mit den Werten: 0, 1, 2, 3, 4, 5)
h: nach *t* Sekunden zurückgelegte Distanz
h_in: ursprüngliche Position des Apfels
delta_h: Liste (Feld) mit allen in jeder Sekunde zurückgelegten Distanzen
y: *y*-Koordinate des Apfels
Bemerkung: Die *x*-Koordinate bleibt konstant = 0, damit sich die Trajektorie im Bild leichter von links nach rechts bewegen lässt.

Zu Beginn befindet sich der Apfel am Ausgangspunkt mit den Koordinaten (0,180). In 📷 3 sind der Ausgangspunkt und die Richtung für die zurückgelegte Distanz *h*(*t*) markiert.



📷 3: Orientierung im Koordinatensystem



📷 4: Programmieralgorithmus für den freien Fall

Indem man eine Schleife fünfmal durchlaufen lässt, berechnet man die Distanz, die der Apfel nach jeder Sekunde zurückgelegt hat, und bestimmt so die neue *y*-Koordinate unter Berücksichtigung der Bildschirmcharakteristika (Programmieralgorithmus in 📷 4)

Zusatzaufgabe

Die Schülerinnen und Schüler können Δ*t* und die Zeit *t* modifizieren (der Apfelbaum müsste dann sehr groß sein, vielleicht wäre ein Hochhaus besser) oder die Aufgabe auf einen Planeten mit abweichender Fallbeschleunigung verlegen. Man könnte die Reibungskraft der Luft hinzufügen und eine variable Erdbeschleunigung annehmen, indem man den Apfel aus größerer Höhe aus einem Wetterballon fallen lässt.

<Programm 2: Fallender Wassertropfen>

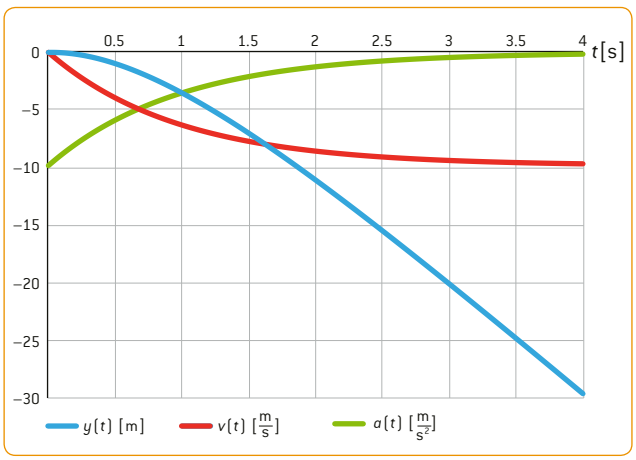
An Regentagen können wir das Fallen von Wassertropfen beobachten. Die Schülerinnen und Schüler analysierten mit einer Simulation die lineare Bewegung eines Tropfens. Sie sahen, dass sich der Fall des Wassertropfens anfangs beschleunigte, die Beschleunigung jedoch abnahm. Nach einer bestimmten Zeit erreichte die Geschwindigkeit des Tropfens ihre Grenze – die Endgeschwindigkeit v_t („terminal velocity") – wenn die Beschleunigung gleich null war. Dann fiel der Wassertropfen mit dieser konstanten Geschwindigkeit weiter. Wie erklärt man das?

Physikalische Theorie

Im beschleunigten Teil der Bewegung wirken zwei Kräfte in entgegengesetzter Richtung auf den Tropfen ein: die Gravitationskraft $G = mg$ (*m:* Masse des Tropfens, *g:* Erdbeschleunigung) und die Reibungskraft $F_f = kv$ (*k:* Proportionalitätskonstante, *v:* momentane Geschwindigkeit, Index f für „friction"). Die Beschleunigung des Tropfens wird $a = g - \frac{k}{m} v$.

In unserer Simulation nehmen wir einen großen Tropfen mit einem Durchmesser von ca. 5 mm mit einer Endgeschwindigkeit $v_t = 9{,}8 \frac{m}{s}$ an.[5] Hier beträgt die Konstante $\frac{k}{m} = \frac{1}{s}$. Die Beschleunigung nimmt ab, wenn die Geschwindigkeit zunimmt (📷 5). Die Startwerte sind $a = 9{,}8 \frac{m}{s^2}$, $v = 0$ und $y = 0$.

Um die momentane Beschleunigung und Geschwindigkeit zu berechnen, verwenden wir ein kleines Programm in C++[4], in

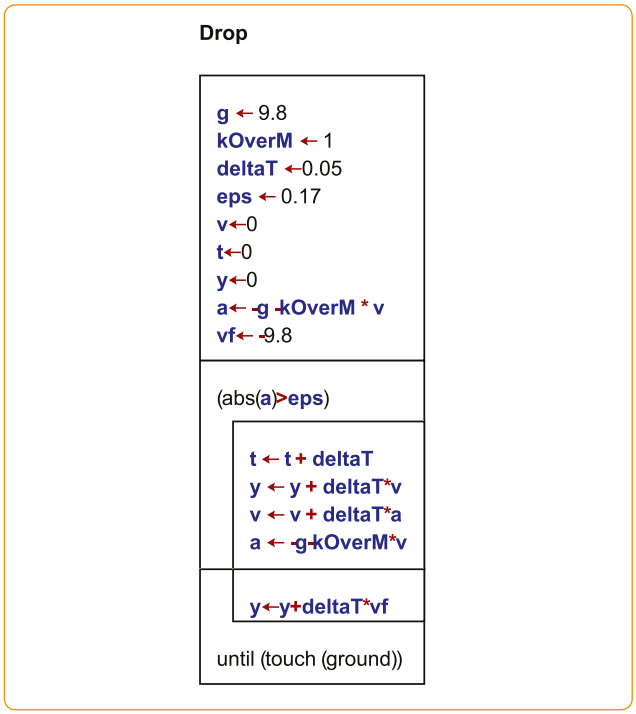


📷 5: Zusammenhang zwischen zurückgelegter Distanz, Geschwindigkeit und Beschleunigung

dem wir die Beschleunigung und die Geschwindigkeitskonstante für sehr kleine Zeitintervalle Δt (z. B. 0,05 s) betrachten. In diesem Fall erhöht sich für jedes gewählte Δt die Geschwindigkeit mit $\Delta v = a\Delta t$, und die zurückgelegte Distanz mit $\Delta y = v\Delta t$ (schrittweise Methode).

Wie programmiert man das?

1. Zeichne ein Wassertropfen-Sprite.
2. Zeichne eine horizontale grüne Linie am unteren Rand des Hintergrunds.
3. Erstelle ein Sprite mit der Meldung „Beschleunigung=0“, welche erscheint, wenn die Beschleunigung etwa 0 ist.
4. Schreibe das Programm für das Tropfen-Sprite. Der Tropfen startet bei $(0, y_{init})$. Mit einer Schleife berechnen wir *a(t), v(t), y(t)* fortlaufend neu. Wir nehmen die zurückgelegte Distanz des Tropfens nach jedem Δt, erhalten so die neue y-Koordinate und berücksichtigen dabei die Bildschirmcharakteristika. Die Beschleunigung nimmt ab, und wenn sie bei ca. 0 liegt, endet die Schleife und es erscheint die Meldung auf dem Bildschirm. Als Nächstes fällt der Tropfen mit einer konstanten Geschwindigkeit, bis er die grüne Linie des Hintergrunds berührt. 📷 6 hilft, den Code zu verstehen.



📷 6: Fallender Wassertropfen

Zusatzaufgabe

Die Schülerinnen und Schüler können dieses Programm verbessern, indem sie eine Variable für die Masse des Wassertropfens (der Durchmesser des Wassertropfens beträgt üblicherweise 1 mm bis 5 mm)[6] und einen anderen Reibungskrafttyp hinzufügen: $F_f = \frac{kv^2}{2}$.

Außerdem können sie weitere Tropfen mit unterschiedlichen Massen erstellen und ihren Fall vergleichen.

<Programm 3: Elastischer Stoß>

Es gibt viele Beispiele für kollidierende Körper. Diese Kollisionen sind oft kompliziert, aber wir betrachten hier den elastischen Stoß, welcher sich auf den Zusammenstoß von Billard- oder Stahlkugeln anwenden lässt, bzw. auf die Theorie von Molekülkollisionen im Modell des idealen Gases.

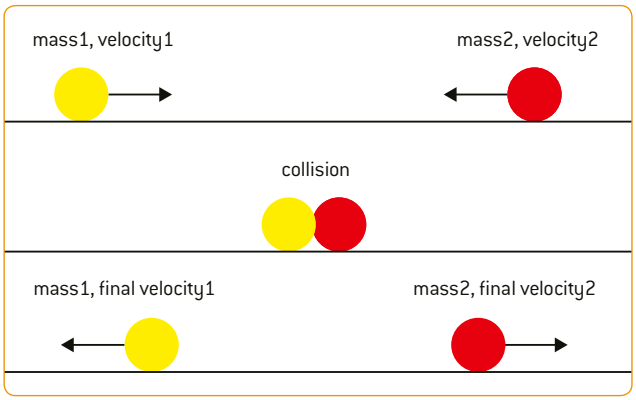
Physikalische Theorie

Wir betrachten die Erhaltungssätze für den linearen Impuls und die kinetische Energie für zwei Bälle mit den Massen m_1 und m_2, den Startgeschwindigkeiten $(\vec{v_1})$ und $(\vec{v_2})$ und den Endgeschwindigkeiten $(\vec{v_{1f}})$ und $(\vec{v_{2f}})$. (📷 7)

$$m_1\vec{v_1} + m_2\vec{v_2} = m_1\vec{v_{1f}} + m_2\vec{v_{2f}} \text{ und}$$

$$\frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 = \frac{1}{2}m_1v_{1f}^2 + \frac{1}{2}m_2v_{2f}^2$$

Wenn alle Bewegungen entlang derselben Achse (hier x-Achse) stattfinden, können wir zur Anzeige der Richtung + oder – verwenden. Die Vektornotation wird für diesen Fall nicht benötigt und die Endgeschwindigkeiten werden wie folgt berechnet:
$v_{1f} = 2\frac{m_1v_1+m_2v_2}{m_1+m_2} - v_1$ und $v_{2f} = 2\frac{m_1v_1+m_2v_2}{m_1+m_2} - v_2$.

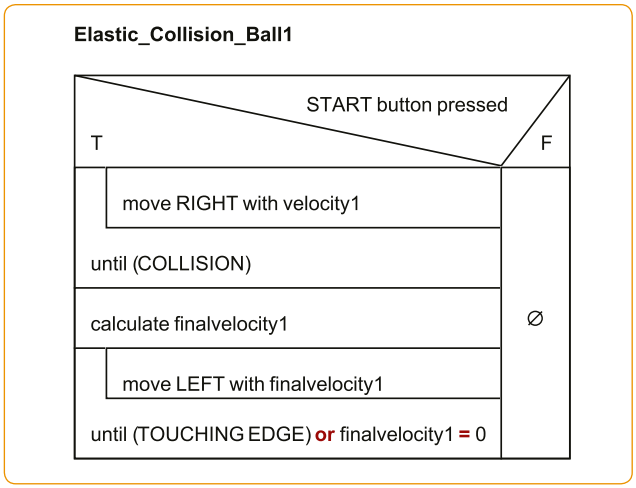


📷 7: Elastischer Stoß

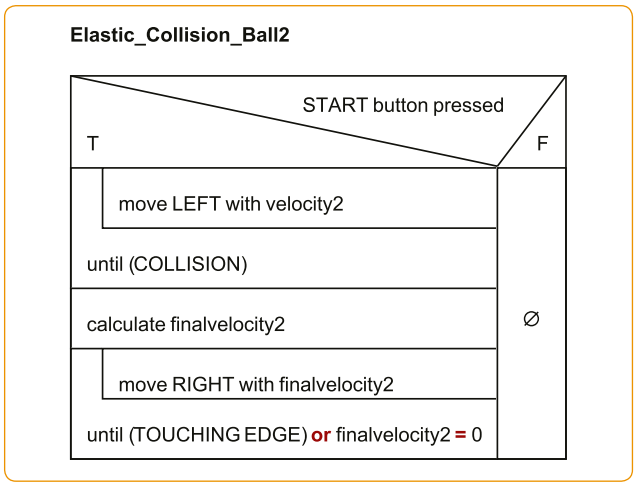
Wie programmiert man das?

1. Wähle zwei Sprites für die Bälle (Ball1 und Ball2) und ein Sprite für den **START**-Knopf (Start-Sprite).
2. Verwende die Variablen: *mass1, mass2, velocity1, velocity2* (die Masse und die Startgeschwindigkeit) für jedes Objekt. Mache die variablen Schieberegler sichtbar und bestimme ihre Minimal- und Maximalwerte.
3. Gib Masse und Startgeschwindigkeit für jedes Objekt ein.
4. Drücke den **START**-Knopf. In diesem Moment sendet das Start-Sprite eine Meldung an die Ball-Sprites. Sobald sie die Meldung erhalten, bewegen sich die Bälle mit *Distanz = Geschwindigkeit · Zeit* aufeinander zu.

5. Berechne die Endgeschwindigkeiten der Bälle und verwende sie, um diese in die richtige Richtung zu bewegen, wobei sich jeder Ball solange bewegt, bis er entweder den Rand berührt und das Bild verlässt oder stehen bleibt, weil seine neue Geschwindigkeit gleich 0 ist.
   📷 8 und 9 zeigen, wie die Bälle in Scratch[1] animiert werden.



📷 8: Elastischer Stoß für Ball1



📷 9: Elastischer Stoß für Ball2

Zwei Anwendungsbeispiele dieses Programms:

1. Wähle eine Geschwindigkeit 0 und gleiche Massen für die Bälle; nach der Kollision stoppt der sich bewegende Ball und der andere bewegt sich mit der Geschwindigkeit, die der erste vor dem Aufprall hatte.
2. Die Bälle haben unterschiedliche Geschwindigkeiten und gleiche Massen; durch die Kollision vertauschen sich die Geschwindigkeiten der Bälle.

In beiden Beispielen vertauschen sich die Impulse der Bälle.

Zusatzaufgabe

Die Schülerinnen und Schüler verändern die Größe der Bälle direkt proportional zu ihrer Masse. Außerdem können sie ein Programm für einen zweidimensionalen elastischen Stoß entwickeln (Simulation des Compton-Effekts) oder für die Kollision eines Balls mit einer Wand (mechanisches Reflexionsgesetz). Sie können ihre Untersuchungen mit einem weiteren Programm zum unelastischen Stoß fortsetzen.[4]

<Fazit>

<Für die Schülerinnen und Schüler>

Vorteile

Die Schülerinnen und Schüler erlernten physikalische Konzepte auf eine unterhaltsamere Weise und konnten Phänomene durch die Simulationen in Scratch besser verstehen, wobei sie zugleich ihre Kenntnisse in Informatik und Physik praktisch anwandten. Auch wenn nicht alle Projekte perfekt waren, verbesserten die Schülerinnen und Schüler ihre Programmierkenntnisse und ihr algorithmisches Denken.

Nachteile

Alle arbeiteten alleine und größtenteils zu Hause. In der Schule erhielten sie dann Rückmeldungen.

<Für die Lehrkräfte>

Vorteile

Wir beobachteten ein echtes Interesse an der Entwicklung eines eigenen Programms und ein besseres Lernergebnis als im klassischen Unterricht.

Nachteile

Wegen der verschiedenen physikalischen Themen und der sehr spezifischen Bugs in den einzelnen Programmen war es für uns schwierig, die gesamte Klasse zu koordinieren. Wir sind der Ansicht, dass es besser wäre, allen jeweils nur ein Thema zu geben und dieses dann je nach den Fähigkeiten der Schülerinnen und Schüler zu verbessern und zu vertiefen.

<Kooperationsmöglichkeiten>

Klassen aus verschiedenen Schulen und Ländern können die Aufgaben des Projekts lösen und neue Aufgaben mit anderen, auf das ursprüngliche Thema bezogenen Ideen entwickeln. Alle diese Programme können auf die Scratch-Plattform hochgeladen werden, worauf ein Wettbewerb unter den besten Einsendungen durchgeführt werden kann. Bei der Bewertung der Arbeit müssen die Lehrkräfte die Komplexität der Programmierung und der physikalischen Themen mitberücksichtigen.

<Quellen und Hinweise>

[1] https://scratch.mit.edu
[2] Alle zusätzlichen Materialien sind online erhältlich unter www.science-on-stage.de/coding-materialien.
[3] https://scratch-dach.info
[4] https://scratch.mit.edu/users/SonS_Coding
[5] http://hypertextbook.com/facts/2007/EvanKaplan.shtml
[6] https://journals.ametsoc.org/doi/pdf/10.1175/1520-0450%281969%29008%3C0249%3ATVORA%3E2.0.CO%3B2

<Impressum>

<Entnommen aus>
Coding im MINT-Unterricht
www.science-on-stage.de/coding

<Herausgeber>
Science on Stage Deutschland e.V.
Am Borsigturm 15
13507 Berlin

<Revision und Übersetzung>
Translation-Probst AG

<Gestaltung>
WEBERSUPIRAN.berlin

<Illustration>
Rupert Tacke, Tricom Kommunikation und Verlag GmbH

<Text- und Bildnachweise>
Die Autorinnen und Autoren haben die Bildrechte für die Verwendung in dieser Publikation nach bestem Wissen geprüft und sind für den Inhalt ihrer Texte verantwortlich.

<Bestellungen>
www.science-on-stage.de
info@science-on-stage.de

<ISBN PDF-Fassung>
978-3-942524-60-5

1. Auflage 2019

Science on Stage Deutschland -
The European Network for Science Teachers

… ist ein Netzwerk von Lehrkräften für Lehrkräfte aller Schularten, die Mathematik, Informatik, Naturwissenschaften und Technik (MINT) unterrichten.
… bietet eine Plattform für den europaweiten Austausch anregender Ideen und Konzepte für den Unterricht.
… sorgt dafür, dass MINT im schulischen und öffentlichen Rampenlicht steht.

Science on Stage Deutschland e.V. wird maßgeblich gefördert von think ING., der Initiative für den Ingenieurnachwuchs des Arbeitgeberverbandes GESAMTMETALL.

Machen Sie mit!
www.science-on-stage.de
www.facebook.com/scienceonstagedeutschland
www.twitter.com/SonS_D

Bleiben Sie informiert!
www.science-on-stage.de/newsletter

Ein Projekt von
SCIENCE ON STAGE DEUTSCHLAND

Hauptförderer von Science on Stage Deutschland
think ING.
Die Initiative für Ingenieurnachwuchs

Mit freundlicher Unterstützung von
SAP