Programmieren mit KI für den Mathematikunterricht

Kann Vibe Coding die Entwicklung digitaler Unterrichtsressourcen revolutionieren?

Frederik Dilling

Was ist Vibe Coding?

Die Entwicklung von generativer Künstlicher Intelligenz (KI) und im Speziellen von sogenannten Large Language Models (LLMs), auf denen beispielsweise die bekannte KI-Oberfläche ChatGPT aufbaut, ist bereits von Beginn an eng mit dem Anwendungsfeld des Programmierens verknüpft. Dies liegt auch nahe: Zum einen stammen die KI-Entwickler und die bereits früh mit dem Thema in Kontakt kommenden Personen meist aus dem IT-Bereich – KI-Modelle basieren eben auf durch Programmierung umgesetzten Algorithmen. Zum anderen sind LLMs im wahrsten Sinne des Wortes Sprachmodelle – sie simulieren Sprache auf der Basis von Wahrscheinlichkeitsmodellen. Und Programmieren nutzt eben eine besondere Form der Sprache mit klar definierten Grundbegriffen und einer klaren Logik im Aufbau – man spricht schließlich auch von einer Programmiersprache.

Im Oktober 2022, also kurz bevor OpenAI sein GPT-3.5-Modell veröffentlicht und damit den KI-Hype ausgelöst hat, haben Becker et al. (2023) unter dem Titel „Programming is hard – Or at least it used to be“ ein inzwischen viel zitiertes Positionspapier zu den Potenzialen und Herausforderungen von KI für das Programmieren bzw. das Programmierenlernen als Preprint veröffentlicht. Sie beziehen sich in dem Beitrag auf „AI-driven code generation tools“ (S. 500), fassen darunter aber auch frühere Formen von LLMs, und kommen nach der Erörterung unterschiedlicher Szenarien zu dem Schluss:

> AI-generated code is now firmly part of the education landscape, but we do not yet know how to adapt our practices to overcome the challenges and leverage the benefits. What we confidently predict is that software development of the future will include an increasing amount of auto-generated code and this includes those training for such roles and jobs, such as our students. We believe this minimally suggests a shift in emphasis towards code reading and evaluating rather than code generation. (S. 505)

An diesem kurzen Zitat ist zu erkennen, dass die Autor:innen von einem grundsätzlichen Wandel der Tätigkeit des Programmierens ausgehen, mit einer Entwicklung hin zum Lesen und Bewerten von automatisch generierten Codes oder Code-Stücken anstelle von der selbstständigen Entwicklung von Code. Diese Entwicklung hat sich in den letzten Jahren auch durchaus gezeigt und hat im Februar dieses Jahres zu einem neuen Begriff geführt – dem sogenannten *Vibe Coding*. Als erstes verwendet hat den Begriff der KI-Experte und ehemalige Tesla- und OpenAI-Mitarbeiter Andrej Karpathy in einem Post auf dem sozialen Netzwerk X (x.com/karpathy/status/1886192184808149383):

> There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding – I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

Der Begriff Vibe Coding hat sich für das Generieren von Code durch die Interaktion mit einem LLM in natürlicher Sprache schnell durchgesetzt und wurde vor kurzem sogar vom Collins Dictionary zum Wort des Jahres gekürt (blog.collinsdictionary.com/language-lovers/collins-word-of-the-year-2025-ai-meets-authenticity-as-society-shifts/), unter anderem, weil es prototypisch für die sich verändernde Beziehung zwischen Mensch und Technologie stehe. Während Karpathy das Phänomen Vibe Coding sehr anschaulich beschreibt, lässt sich in einem Beitrag von Meske et al. (2025) eine präzise wissenschaftliche Definition des Begriffs finden:

> Vibe coding is a software development paradigm where humans and generative AI engage in collaborative flow to co-create software artifacts through natural language dialogue, shifting the mediation of developer intent from deterministic instruction to probabilistic inference. (S. 1)

Hierauf aufbauend formulieren sie fünf Kernaspekte von Vibe Coding. Zunächst einmal liegt der Fokus beim Vibe Coding auf dem Ziel anstelle der Implementierung. Nutzer:innen beschreiben dabei möglichst klar, was das entstehende Programm leisten oder wie die Benutzeroberfläche aussehen soll – um die Implementierung wie zum Beispiel die Auswahl passender Programmier-Frameworks kümmert sich die KI. Ein weiteres Charakteristikum von Vibe Coding ist die schnelle dialogische Interaktion mit dem LLM. Diese ersetzt vergleichsweise starre Write-Compile-Test Loops, in denen das erste Feedback zu geschriebenem Code erst zu einem recht späten Zeitpunkt im Prozess auftritt und sich eher auf mögliche formale Fehler im Code bezieht. Als dritten Kernaspekt nennen Meske et al. (2025) die „implementation abstraction", worunter sie verstehen, dass durch Vibe Coding eine Bereitstellung ohne ein vollständiges Verständnis für den Code möglich ist. Hinzu kommt als vierter Aspekt die dynamische semantische Verfeinerung durch die KI, indem diese wesentliche Interpretationsleistungen übernimmt und Konzepte weiterdenkt. Diese verschiedenen Aspekte führen dann schließlich zum fünften Kernaspekt von Vibe Coding, der laut Meske et al. (2025) die Entstehung ko-kreativer Flow-Zustände mit einem produktiven Rhythmus zwischen Entwickler:innen und KI ist.

Meske et al. (2025) beschreiben Vibe Coding als ein komplett neues Paradigma. Auch in vielen anderen wissenschaftlichen Texten werden Vibe Coding und das klassische Programmieren mit ähnlichen Erkenntnissen verglichen (z. B. Ray, 2025; Horvat, 2025). Das Programmieren hat sich allerdings auch schon vor der verstärkten KI-Nutzung in einigen Bereichen immer wieder verändert und weiterentwickelt, z. B. um Menschen ohne oder mit nur wenig Programmiererfahrung Zugänge zu schaffen. Im Jahr 2007 hat Mitchel Resnick mit einem Team von Entwickler:innen am MIT die erste Version von Scratch entwickelt – eine Programmierumgebung, in der sich Code-Blöcke mit bestimmten Befehlen puzzleartig zu kompletten Programmen zusammensetzen lassen. Dadurch wurde es möglich, wichtige Aspekte von Algorithmen und Programmierungen wie z. B. Schleifen oder Wenn-dann-Beziehungen kennenzulernen, ohne vorher komplizierte Programmiersprachen lernen zu müssen. Das Konzept findet inzwischen in der informatischen Grundbildung starke Anwendung – so gibt es auch viele Beispiele für den Einsatz von Scratch im Mathematikunterricht (z. B. Förster, 2011; Bescherer & Fest, 2019; Dilling & Vogler, 2022a) wie auch für die Übertragung auf verschiedenste Anwendungsbereiche wie das Design von 3D-Modellen (z. B. Dilling et al., 2022) oder die Bedienung von Mikrocontrollern (z. B. Dilling & Plack, 2023). Aber auch außerhalb vom Bildungsbereich sind graphische Programmierumgebungen inzwischen üblich, um Workflows zu definieren und den Zugriff auf konkrete Hardware oder Software zu erleichtern. Beispielsweise lassen sich mit dem Programm Voiceflow blockbasiert Voice-Anwendungen für Sprachassistenzsysteme wie Amazon Alexa erstellen (Dilling, 2022). Für die professionelle Programmierung von kompletten Programmen oder Apps findet es aber eher nicht Anwendung, was an den stark eingeschränkten Umsetzungsmöglichkeiten im Vergleich zu komplexen skriptbasierten Programmiersprachen liegt. Mit Vibe Coding ist nun das erste Mal der Zugriff auf solche Programmiersprachen möglich, ohne diese selbst zu beherrschen.

Da es sich bei Vibe Coding noch um ein junges Phänomen handelt – übrigens so jung, dass die meisten aktuellen Sprachmodelle den Begriff noch gar nicht kennen – gibt es bisher nur wenige belastbare Forschungsergebnisse. Fawzy et al. (2025) haben vor diesem Hintergrund ein interessantes Vorgehen gewählt, um Vibe Coding in der Praxis genauer zu untersuchen. In einem Systematic Grey Literature Review haben sie Erfahrungsberichte von Anwender:innen aus Blogs, Foren, Medienartikeln und anderen öffentlich zugänglichen Quellen analysiert. Dabei haben sie herausgefunden, dass die Motivation zur Nutzung von Vibe Coding vor allem in der Geschwindigkeit und Effizienz liegt. Weitere wichtige Gründe sind die Zugänglichkeit und das Empowerment sowie das Lernen und Experimentieren. Eine Mehrheit der Anwender:innen beschreibt, dass sie beim Vibe Coding einen sofortigen Erfolg hatten, allerdings lassen sich in den Quellen auch Beispiele für Schwierigkeiten bei der passenden Formulierung von Prompts und für fehlerhafte Codes finden. Die Code-Qualität wird in den Quellen eher mäßig eingeschätzt. So wird in den meisten Quellen eine schnelle aber fehlerhafte Code-Generierung beschrieben, andere sprechen von einem schwachen und fehleranfälligen Code. Maßnahmen zur Qualitätssicherung fanden in vielen Fällen nicht statt, es wurde unkritisch den Ausgaben vertraut oder die Qualitätssicherung wurde an die KI delegiert. Nur etwa ein Drittel der Quellen berichtet von manuellen Tests und direkten Änderungen am Code.

Dieser Einblick in die Erfahrungen von Vibe Coding Anwender:innen zeigt, dass der Ansatz zwar Potenziale bietet, aber auch mit einigen Herausforderungen einhergeht. Im Bildungsbereich sind viele Anwendungs-

szenarien für Vibe Coding denkbar – in diesem Beitrag soll die Entwicklung von digitalen Unterrichtsressourcen als ein möglicher Use Case genauer beleuchtet werden, welcher aus Sicht des Autors bereits mit den jetzigen technischen Möglichkeiten gut umsetzbar ist. Hiermit ist die folgende Erwartung und Hoffnung verbunden:

Vibe Coding könnte es Lehrkräften ohne besondere Programmierkenntnisse ermöglichen, in kurzer Zeit individuelle, interaktive und flexible digitale Unterrichtsressourcen zu erstellen.

Im Bereich der medizinischen Bildung gibt es bereits erste Versuche, wie sich durch Vibe Coding Simulationen von medizinischen Entscheidungsprozessen erstellen lassen, um besonders praxisnahe Lerngelegenheiten für zukünftiges medizinisches Personal zu bieten (Chow & Ng, 2025). Aber auch in der schulischen Bildung insb. im Mathematikunterricht ist der Umgang mit digitalen Medien und Werkzeugen und damit erzeugbaren (mathematischen) Darstellungen ein wichtiges und zeitgemäßes Lernprinzip.

Digitale Unterrichtsressourcen mit Vibe Coding erstellen

Der Umgang mit digitalen Medien und Werkzeugen hat im Mathematikunterricht bereits eine recht lange Tradition. Unterschiedliche Formen von Medien und Werkzeugen treten dabei als Unterrichtsressourcen auf (Schmidt-Thieme & Weigand, 2015): Anschauungsmittel repräsentieren mathematische Inhalte, ohne dass Schüler:innen auf diese einwirken können. Dagegen lassen die durch Arbeitsmittel repräsentierten mathematischen Objekte Handlungen und Operationen an ihnen zu. Werkzeuge lassen sich als spezielle Arbeitsmittel mit einem besonders großen Anwendungsbereich charakterisieren.

Die Einteilung von Unterrichtsressourcen in Anschauungsmittel, Arbeitsmittel und Werkzeuge ist allerdings in der Praxis nur situationsspezifisch möglich. Das zeigt auch eine andere Unterteilung von Roth (2019). Dieser betrachtet den Einsatz von digitalen Mathematikwerkzeugen wie beispielsweise GeoGebra. Der Werkzeugcharakter wird besonders bei der freien Nutzung deutlich – in diesem Szenario wählen Schüler:innen bei der Aufgabenbearbeitung eigenständig unter vielen Funktionen eines Werkzeugs die für den eigenen Lösungsweg wichtigen aus und wenden diese an. Die freie Nutzung lässt sich aber auch bewusst einschränken, indem aufbauend auf dem Werkzeug sogenannte Applets erstellt werden. Es handelt sich hierbei um Programme, bei denen nur ausgewählte Funktionen eines digitalen Werkzeugs durch Schüler:innen genutzt werden können und häufig auch bereits Konfigurationen von mathematischen Objekten vorgegeben sind. Nach Roth (2019) dienen solche Applets als „Experimentier- und Erkundungsumgebungen für spezifische Fragestellungen des Mathematikunterrichts“ (S. 239). Sie haben bei dieser Nutzung somit eher den Status eines relativ spezifischen Arbeitsmittels als den eines Werkzeugs.

Mathematische Applets lassen sich mit verschiedenen Mitteln erstellen. Bekannt ist der Weg über GeoGebra (z. B. Elschenbroich, 2017; Stoffels, 2021). Dafür meldet man sich auf der Internetseite von GeoGebra an und kann dann beispielsweise eine geometrische Konstruktion erstellen, die über einen Schieberegler veränderbar ist. Bei der späteren Bereitstellung der Konstruktion als Material über die GeoGebra-Website lassen sich dann die verwendeten Konstruktionsschritte und sonstigen GeoGebra-Werkzeuge ausblenden, sodass die Schüler:innen nur noch über den Schieberegler Veränderungen an der Konstruktion vornehmen können.

Applets für den Mathematikunterricht müssen aber nicht zwangsläufig auf der Basis eines mathematischen Werkzeugs generiert werden. In der Schulpraxis sind beispielsweise auch Plattformen beliebt, auf denen sich bestimmte klassische Aufgaben- und Übungsformate mit konkreten Inhalten füllen lassen. Auf der Internetseite LearningApps.org können z. B. Templates für Lückentexte, Multiple-Choice-Quizzes, Zahlenstrahle oder paarweise Zuordnungen mit mathematischen Aufgabestellungen verbunden oder zum Wiederholen gelernter Inhalte und Verfahrensweisen genutzt werden (z. B. Weng & Pfeiffer, 2016). Auch die oben in diesem Beitrag bereits thematisierte Blockprogrammier-Anwendung Scratch ermöglicht die Erzeugung von Mathematik-Applets, in denen beispielsweise zufallsgeneriert bestimmte Rechenaufgaben gestellt und Antworten automatisiert überprüft werden (z. B. Dilling & Vogler, 2022b). Eine letzte nicht zu vernachlässigende Option ist das klassische skriptbasierte Programmieren von Anwendungen und die Bereitstellung als Webanwendung oder App für digitale Endgeräte (z. B. Urff, 2011).

Die verschiedenen Herangehensweisen bieten jeweils verschiedene Vor- und Nachteile. Bei der Nutzung von GeoGebra-Applets können Schüler:innen z. B. niederschwellig Erfahrungen mit dem Interface von GeoGebra sammeln und schrittweise Funktionen des Werkzeugs kennenlernen. Auf der anderen Seite ist man auch innerhalb der Applets auf die Funktionen von GeoGebra beschränkt und das Erstellen eines ansehnlichen Applets kann durchaus viel Zeit in Anspruch nehmen. Beim klassischen Programmieren hat man dagegen schier unendliche Möglichkeiten, der Zeitfaktor

ist allerdings nochmal deutlich ausgeprägter. Hinzu kommt das Problem, dass man zur Erstellung sehr gute Programmierkenntnisse braucht, was für einen Großteil der Lehrkräfte eher nicht gegeben ist.

Die Erstellung mathematischer Applets könnte für Vibe Coding ein gutes Anwendungsfeld sein, da diese ziemlich klar abgegrenzte Ziele und Funktionen haben und meist mit einer relativ überschaubaren Menge an Code umgesetzt werden können. Um ein Mathematik-Applet mit Vibe Coding zu generieren, müssen nur drei einfache Schritte ausgeführt werden:

1. *Entwicklung einer Applet-Idee und Formulierung eines Start-Prompts:* Zunächst einmal sollte man sich genau überlegen, welche Funktionen das Applet erfüllen soll. Dies umfasst neben didaktischen Überlegungen auch technische bzw. gestalterische zu der Bedienung, den Einstellungsmöglichkeiten und dem Aufbau des User Interfaces. Aus diesen Überlegungen heraus kann dann der Start-Prompt formuliert werden, der möglichst präzise die eigenen Überlegungen beschreibt (z. B. in Form eines Mega-Prompts). Das eigentliche Programmierframework muss dazu nicht vorgegeben werden, es bietet sich aber an, konkretere Angaben zum gewünschten Ausgabeformat zu machen (z. B. Webanwendung).
2. *Testen und Überarbeiten des Applets:* Die Programmierung übernimmt das LLM nach der Eingabe des Prompts ganz von selbst. Es ist allerdings klar zu empfehlen, für das Vibe Coding ein sogenanntes Reasoning-Modell oder einen Thinking-Modus zu nutzen, die speziell für komplexe Problemlöseaufgaben und logisches Schlussfolgern entwickelt wurden. Ist der Code für das Applet fertig generiert, kann er in manchen KI-Oberflächen direkt als Vorschau getestet werden. Alternativ lässt sich der Code als Datei abspeichern. Beispielsweise kann man einen HTML-Code sehr leicht testen, indem man diesen im Editor in eine TXT-Datei kopiert und beim Abspeichern mit dem Zusatz .html versieht. Öffnet man dann die Datei, wird das Applet automatisch im Browser geöffnet. Diese erste Applet-Version kann dann getestet werden. Treten Fehler auf, können diese als Überarbeitungsauftrag an die KI weitergegeben werden. Dazu kann dann z. B. auch eine Fehlermeldung direkt als Information in das Dialogfenster mit dem Chatbot kopiert werden. Auch andere Änderungswünsche, z. B. zum Design der App, lassen sich im Dialogfenster formulieren und mitteilen, sodass das Applet überarbeitet wird und erneut getestet werden kann. Dabei können auch durchaus mehrere Überarbeitungsschleifen hintereinander durchgeführt werden. Wenn die erste Applet-Version noch sehr weit von den Vorstellungen abweicht, bietet es sich statt einer Überarbeitung aber eher an, den Start-Prompt nochmal zu überarbeiten und die Anfrage in einem neuen Chatfenster erneut zu stellen.
3. *Speicherung und Bereitstellung:* Ist man mit dem erstellten Applet zufrieden, kann man es final abspeichern. Die Bereitstellung für Schüler:innen kann dann beispielsweise erfolgen, indem man die Datei in ein Learning Management System hochlädt, auf einer Website veröffentlicht oder in eine digitale Lernumgebung einbindet. Die Möglichkeiten hängen insbesondere davon ab, welches Ausgabeformat man für die Anwendung vorgesehen hat.

Beispiele für den Mathematikunterricht

Nachdem im vorherigen Abschnitt mit einem einfachen Workflow vorgestellt wurde, wie sich Mathematik-Applets als digitale Unterrichtsressourcen durch Vibe Coding genieren lassen, sollen nun ein paar Beispiele vorgestellt werden, die der Autor dieses Beitrags auf diese Weise erzeugt hat. Dabei sollte angemerkt werden, dass es sich um Werkstattmaterialien handelt, an denen die Potenziale aber auch die bestehenden Hürden des Ansatzes gut erkannt werden können, die allerdings nicht unbedingt Musterbeispiele für gute Applets darstellen. Für alle Beispiele wurde eine einfache HTML-Umgebung genutzt, da diese sich leicht in andere bestehende Systeme einbinden lässt. Alternativ lassen sich aber auch deutlich komplexere und mächtigere Programmiersprachen für das Vibe Coding nutzen. In der Tabelle sind die verwendeten Prompts und ein Screenshot des damit entstandenen Applets zu sehen. In allen gezeigten Fällen war nach dem Start-Prompt kein weiterer Dialog mit der KI notwendig, um die Applets in der dargestellten Form zu erhalten. Sie können unter dem folgenden Cloud-Link abgerufen und gerne weiterverwendet werden: uni-siegen.sciebo.de/s/JZCnbayWNeqkDBM

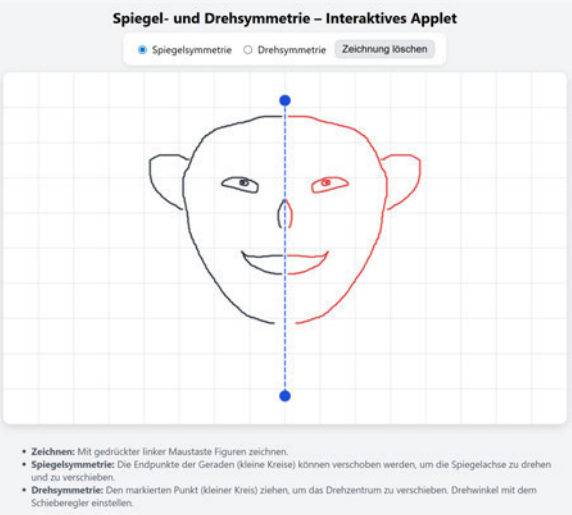
Diskussion und Ausblick

Die vier auf den folgenden Seiten gezeigten, mit Vibe Coding erstellten HTML-Applets decken unterschiedliche mathematische Teilgebiete und klassische Nutzungsszenarien von Applets ab. Mit dem Applet zur Spiegel- und Drehsymmetrie könnte man beispielsweise erste Erfahrungen mit den Themen Symmetrie und Kongruenzabbildungen machen. Darauf aufbauend ermöglicht es dann weiterführende Erkundungen, z. B. zu Fixpunkten der Abbildungen oder zur Auswirkung der Lage des Drehzentrums auf die gedrehte Figur. Mit dem Ballwurf-Applet lässt sich zunächst

Prompt

Erstelle ein HTML-Applet für den Mathematikunterricht zum Thema Spiegel- und Drehsymmetrie. Im Applet soll eine Zeichenfläche zu sehen sein. Auf der Fläche soll sich eine Gerade befinden, zunächst mittig und senkrecht von unten nach oben. Man soll die Gerade aber auch drehen und verschieben können. Zeichnet man mit der Maus eine geometrische Figur auf die Zeichenebene, soll die an der Gerade gespiegelte Figur automatisch auch erzeugt werden. In dem Applet soll man anstatt der Spiegelsymmetrie aber auch auf eine Drehsymmetrie umstellen können. Dann ist in der Mitte der Zeichenebene ein Punkt markiert, der sich aber auch verschieben lässt. Wird in diesem Szenario eine Figur mit der Maus gezeichnet, erscheint automatisch auch die durch Drehung um den Punkt erzeugte Figur. Der Drehwinkel soll sich einstellen lassen, aber erstmal standardmäßig 180 Grad sein. Das Applet soll auch auf Tablets im Browser funktionieren.

Entstandenes Applet

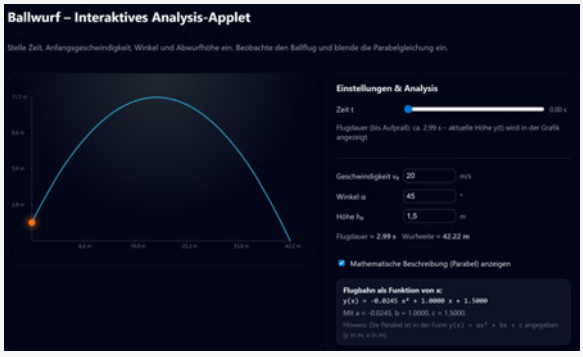


HTML-Applet für den Mathematikunterricht zum Thema Spiegel- und Drehsymmetrie

Prompt

Erstelle ein HTML-Applet zur Analysis im Mathematikunterricht. Das Applet soll den Wurf eines Balls simulieren. Mit einem Schieberegler soll sich der Zeitpunkt einstellen und somit der Ball bewegen lassen. Außerdem soll man die Abwurfgeschwindigkeit, den Abwurfwinkel und die Abwurfhöhe einstellen können. Die mathematische Beschreibung als Parabel soll sich einblenden lassen.

Entstandenes Applet



HTML-Applet zur Analysis im Mathematikunterricht

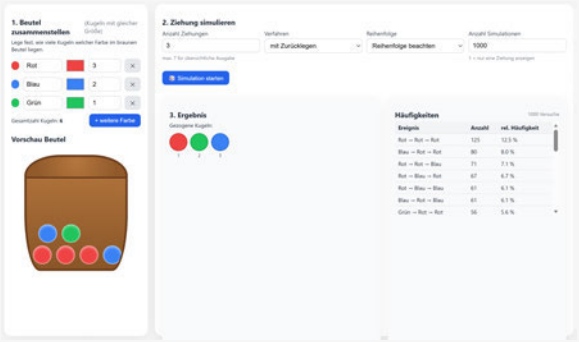
die Verbindung zu quadratischen Funktionen erarbeiten, um dann beispielsweise herauszufinden, unter welchem Winkel der Ball abgeworfen werden müsste, um eine maximale Distanz zu überwinden. Dabei ist dann auch die Verbindung zum Physik- und Sportunterricht möglich. Das Applet zum Urnenmodell kann als wiederkehrendes Arbeitsmittel verwendet werden, um Zufallsexperimente in verschiedensten Kontexten zu simulieren und diese mit dem zugrundeliegenden mathematischen Modell in Verbindung zu bringen. Es ließen sich aber beispielsweise auch tiefgehende Erfahrungen mit dem empirischen Gesetz der großen Zahlen machen, indem die Anzahl der durchgeführten Simulationen gezielt variiert und die Auswirkung auf die Wahrscheinlichkeitsverteilung untersucht wird. Das Memory-Applet ist ein einfaches spielerisches Übungsformat, das sich in ähnlicher Form auch auf andere Kontexte übertragen ließe.

Die entstandenen Applets bieten somit durchaus spannende Einsatzmöglichkeiten im Mathematikunterricht und sind technisch von einer hohen Qualität. Es ist aber auch nicht so, dass ähnliche Applets nicht bereits existieren oder mit GeoGebra oder einer anderen Anwendung nicht hätten erstellt werden können. Das Neue ist also weniger das Ergebnis als der Weg dorthin. Mit einem kurzen und wenig elaborierten Prompt konnte innerhalb von wenigen Minuten ein Applet erstellt werden, das sich früher nur mit viel Aufwand hätte erzeugen lassen. Das bedeutet im Folgeschluss auch, dass mehr Zeit für die didaktischen Ideen und die konkrete kreative Ausgestaltung bleibt. Dabei sollte aber klar sein: Vibe Coding macht aus Lehrkräften noch keine Programmierer:innen. Gerade komplexere und technisch anspruchsvolle Anwendungen lassen sich (noch) nicht auf diese Weise umsetzen, selbst wenn man viel Zeit in die Formulierung des Prompts steckt.

Prompt

Erstelle ein HTML-Applet zur Wahrscheinlichkeitsrechnung im Mathematikunterricht. Das Applet soll das Urnenmodell simulieren. In einem gezeichneten braunen Beutel sollen verschiedene Kugeln liegen. Die Kugeln haben alle die gleiche Größe, man kann aber entscheiden, wie viele Kugeln von welcher Farbe im Beutel sind. Dann wird das Ziehen der Kugeln simuliert. Hierzu soll man unterscheiden können zwischen (1) Ziehen mit Zurücklegen und (2) Ziehen ohne Zurücklegen. Außerdem soll man entscheiden können, ob die Reihenfolge beim mehrfachen Ziehen beachtet werden soll oder nicht. Die Simulation beachtet das Verhältnis der Kugeln im Beutel und die dadurch entstehenden Wahrscheinlichkeiten.

Entstandenes Applet

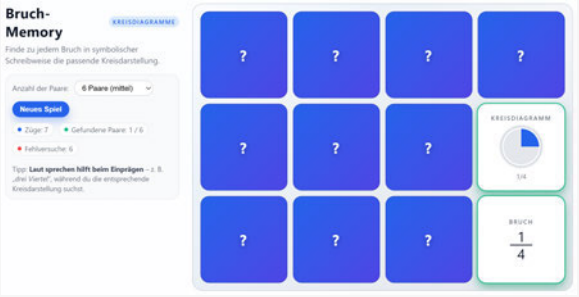


HTML-Applet zur Wahrscheinlichkeitsrechnung

Prompt

Erstelle ein HTML-Applet für den Mathematikunterricht. Es soll sich um ein Memory-Spiel handeln, bei dem Brüche in symbolischer Form mit der entsprechenden bildlichen Darstellung als Kreisdiagramm zusammengebracht werden sollen. Als Brüche sollen keine zu komplizierten oder krummen Zahlen auftreten.

Entstandenes Applet



HTML-Applet für den Mathematikunterricht

Je anspruchsvoller die Programmier-Aufgabe ist, desto fehleranfälliger ist das Vorgehen auch. Und dann ist es durchaus gut möglich, dass man auch nach vielen Versuchen keine gute oder fehlerfreie Anwendung erhält, welche die eigenen Erwartungen erfüllt. Aus diesem Grund werden professionell entwickelte digitale Unterrichtsressourcen wohl auch weiterhin eine hohe Relevanz besitzen. Dies lässt sich auch damit begründen, dass in der technischen Umsetzung und den damit verbundenen Details eines Programms auch viele didaktische Entscheidungen stecken, die man beim Vibe Coding zwangsläufig der KI überlässt.

Und dennoch lässt sich mit Blick auf den Untertitel dieses Beitrags festhalten, dass Vibe Coding ganz klar das Potenzial hat, die Entwicklung von digitalen Unterrichtsressourcen zu revolutionieren, indem es Lehrkräften im Sinne eines echten Empowerments ermöglicht, diese auf einfache Weise selbst zu erstellen. Und das kann einiges verändern: Denkbar wäre beispielsweise die Erstellung von individuellen Applets angepasst an die spezifischen mathematischen Vorstellungen der Schüler:innen. Solche Applets ließen sich sogar spontan erstellen, wenn im Unterricht konkrete Fragen auftreten. Mit der weiteren Entwicklung der KI-Modelle ist außerdem zu erwarten, dass auch die Coding-Fähigkeiten der Systeme zunehmen, was dann womöglich Programmierungen auf einem ganz anderen Level ermöglichen wird.

Im Projekt KIMADU NRW (Dilling & Witzke, im Druck) testen Lehrkräfte aktuell – als eines von vielen verschiedenen KI-Anwendungsszenarien – Vibe Coding zur Erstellung von Mathematik-Applets. Auf der Grundlage von Chats der Lehrkräfte, erstellten Applets und als Antworten auf Reflexionsfragen formulierten Erfahrungsberichten sollen dann die Beliefs und Motivationen von Lehrkräften zum Vibe Coding analysiert werden, um das Phänomen im Bereich der Bildung genauer verstehen zu können. Außerdem sollen die gesammelten Erfahrungen zu praktischen Hilfestellungen ausgearbeitet werden, mit denen dann auch ein spezieller KI-Agent für Vibe Coding zur Erstellung von mathematischen Unterrichtsressourcen gebaut wird.

Literatur

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming Is hard – or at least it used to be: Educational opportunities and challenges of AI code generation. *SIGCSE 2023 Proceesings,* 500–506. DOI: 10.1145/3545945.3569759

Bescherer, C. & Fest, A. (2019). Mathematik und informatische Bildung. Programmieren mit Scratch. In T. Junge & H. Niesyto (Hrsg.), *Digitale Medien in der Grundschullehrerbildung. Erfahrungen aus dem Projekt dileg-SL* (S. 117–130). Kopaed Verlag.

Chow, M., & Ng, O. (2025). From technology adopters to creators: Leveraging AI-assisted vibe coding to transform clinical teaching and learning. *Medical Teacher, 1–3.* DOI: 10.1080/0142159X.2025.2488353

Dilling, F. (2022). Entwicklung von Voice-Apps für den Mathematikunterricht. In B. Brandt, L. Bröll & H. Dausend (Hrsg.), *Digitales Lernen in der Grundschule III* (S. 109–120). Waxmann. DOI: 10.25656/01:25830

Dilling, F., Milicic, G. & Vogler, A. (2022). Coding in the context of 3D printing. In F. Dilling, F. Pielsticker & I. Witzke (Hrsg.), *Learning mathematics in the context of 3D printing* (S. 207–228). Springer Spektrum.

Dilling, F. & Plack, J. (2023). Zufallsexperimente mit Calliope mini simulieren und Reflexionen anregen am Beispiel des manipulierten Würfels. *MNU-Journal, 2/2023*, 139–143.

Dilling, F., & Vogler, A. (2022a). Mathematikhaltige Programmierumgebungen mit Scratch – Eine Fallstudie zu Problemlöseprozessen von Lehramtsstudierenden. In F. Dilling, F. Pielsticker & I. Witzke (Hrsg.), *Neue Perspektiven auf mathematische Lehr-Lern-Prozesse mit digitalen Medien* (S. 359–384). Springer Spektrum.

Dilling, F. & Vogler, A. (2022b). Programmieren im Mathematikunterricht. Arithmetische und geometrische Zusammenhänge mit Scratch erkunden. In B. Brandt, L. Bröll & H. Dausend (Hrsg.), *Digitales Lernen in der Grundschule III* (S. 121–137). Waxmann. DOI: 10.25656/01:25830

Dilling, F. & Witzke, I. (im Druck). Professionalisierung von Mathematiklehrkräften im Kontext generativer KI – Vorstellung von zwei Pilotprojekten. In L. Schick, M. Platz & A. Lambert (Hrsg.), *Beiträge zum Mathematikunterricht 2025* (S. 247–250). WTM.

Elschenbroich, H.-J. (2017). Anschauliche Zugänge zur Integralrechnung mit dem Integrator. *MNU-Journal, 5/2017*, 312–317.

Fawzy, A, Tahir, A. & Blincoe, K. (2025). Vibe coding in practice: Motivations, challenges, and a future outlook – a grey literature review. *arXiv.* DOI: 10.48550/arXiv.2510.00328

Förster, K.-T. (2011). Neue Möglichkeiten durch die Programmiersprache Scratch: Algorithmen und Programmierung für alle Fächer. In R. Haug & L. Holzäpfel (Hrsg.). *Tagungsband GDM 45* (S. 263–266). WTM.

Horvat, M. (2025). What is Vibe coding and when should you use it (or not)? *TechRxiv.* DOI: 10.36227/techrxiv.175459780.03758839/v1

Meske, Ch., Hermanns, T., v. d. Weiden, E., Loser, K.-U. & Berger, Th. (2025). Vibe coding as a reconfiguration of intent mediation in software development: Definition, implications, and research agenda. *arXiv.* DOI: 10.48550/arXiv.2507.21928

Ray, P. P. (2025). A review on vibe coding: Fundamentals, state-of-the-art, challenges and future directions. *TechRxiv.* DOI: 10.36227/techrxiv.174681482.27435614/v1

Roth, J. (2019) Digitale Werkzeuge im Mathematikunterricht – Konzepte, empirische Ergebnisse und Desiderate. In A. Büchter, M. Glade, R. Herold-Blasius, M. Klinger, F. Schacht & P. Scherer (Hrsg.), *Vielfältige Zugänge zum Mathematikunterricht. Konzepte und Beispiele aus Forschung und Praxis* (S. 233–248). Springer Spektrum

Schmidt-Thieme, B. & Weigand, H.-G. (2015). Medien. In R. Bruder, L. Hefendehl-Hebecker, B. Schmidt-Thieme & H.-G. Weigand (Hrsg.), *Handbuch der Mathematikdidaktik* (S. 416–490). Springer Spektrum.

Stoffels, G. (2021). Punkte erzeugen Geraden – Chancen und Herausforderungen des Einsatzes von „GeoGebra Büchern“ in der Linearen Algebra. In F. Dilling & F. Pielsticker (Hrsg.), *Mathematische Lehr-Lernprozesse im Kontext digitaler Medien* (S. 78–101). Springer. DOI: 10.1007/978-3-658-31996-0_4

Urff, Ch. (2011). Virtuelle Arbeitsmittel im Mathematikunterricht der Primarstufe. In S. Ladel & Ch. Schreiber (Hrsg.), *Lernen, Lehren und Forschen in der Primarstufe. Schriften zu Mathematikunterricht und Technologieeinsatz. Band 1.* Franzbecker.

Weng, A. & Pfeiffer, A. (2016). „Lernen durch Lehren“ in der Mathematik – Videotutorials und Apps im Praxistest. *Pedocs.* DOI: 10.25656/01:12264

Frederik Dilling, Universität Siegen
frederik.dilling@uni-siegen.de