Medienimpulse
ISSN 2307-3187
Jg. 64, Nr. 3, 2026
doi: 10.21243/mi-26-03-11

Zwischen Haptik und Algorithmus: Eine Reflexion über analoges Programmieren in der Sekundarstufe I.

Kathrin Zwanziger

Angelika Rusznak

Der Beitrag reflektiert das Unterrichtsmaterial „Programmieren mit der Maus“ anhand des durchgeführten fächerübergreifenden Unterrichtskonzepts „Vom Weg zur Programmierung“ in einer 1. Klasse Mittelschule. Ausgehend von Wegbeschreibungen wurden grundlegende Prinzipien des algorithmischen Denkens im analogen „Unplugged“-Setting handelnd erschlossen. Die Unterrichtserprobung zeigt Potenziale insbesondere für Sequenzierung, Eindeutigkeit, Debugging und Mustererkennung, verweist zugleich aber auf sprachliche, räumlich-kognitive und organisatorische Herausforderungen. Der Beitrag diskutiert daraus resultierende didaktische Konsequenzen für den Einsatz und die

Adaption von Unplugged-Material in der Digitalen Grundbildung der Sekundarstufe I.

This article examines the teaching resource 'Programming with the Mouse' in the context of the interdisciplinary teaching approach 'The Path to Programming', which was implemented in a secondary school. Using route descriptions as a starting point, pupils actively explored the fundamental principles of algorithmic thinking in an analogue 'unplugged' setting. The classroom trial demonstrates potential, particularly in terms of sequencing, unambiguity, debugging and pattern recognition, whilst also highlighting linguistic, spatial-cognitive and organisational challenges. The article discusses the resulting didactic implications for the use and adaptation of unplugged materials in digital literacy education at lower secondary level.

1. Einleitung

Digitale Technologien prägen zunehmend die Lebens- und Arbeitswelt von Kindern und Jugendlichen (Education Group GmbH, 2025). Schule steht damit vor der Aufgabe, Lernende nicht nur zur Nutzung digitaler Werkzeuge zu befähigen, sondern ihnen auch grundlegende Konzepte digitaler Systeme zugänglich zu machen (Bundesministerium für Bildung, o. J.). Eine zentrale Bedeutung kommt dabei dem algorithmischen Denken zu. Die Fähigkeit, Probleme zu strukturieren, in Teilprobleme zu zerlegen, Handlungsabläufe zu planen und eindeutige Anweisungen zu formulieren, bildet eine wesentliche Grundlage informatischer Bildung (Fehrmann & Zeinz, 2021).

Für Lernende stellt dieser Zugang allerdings eine besondere Herausforderung dar. Algorithmen sind abstrakte Strukturen, deren Funktionsweise zunächst nicht unmittelbar sichtbar ist. Wird Programmieren ausschließlich über digitale Endgeräte und Programmiersprachen eingeführt, kommen zu den eigentlichen informatischen Konzepten weitere Anforderungen hinzu: Bedienung der Software, Verständnis der Benutzeroberfläche, Syntax und technische Problemlagen. Gerade beim Einstieg kann dadurch die Aufmerksamkeit von der zugrunde liegenden Logik des Programmierens abgelenkt werden.

Eine Möglichkeit, diesen Einstieg zu entlasten, bietet Unplugged Computing beziehungsweise Unplugged Programming. Darunter werden Lernarrangements verstanden, in denen informatische Konzepte ohne digitale Endgeräte durch analoge, haptische, spielerische oder körperbezogene Aktivitäten vermittelt werden. Befehle können beispielsweise durch Karten repräsentiert, Algorithmen auf Spielfeldern ausgeführt oder Programmabläufe durch Bewegungen dargestellt werden. Die technische Oberfläche tritt dabei zunächst in den Hintergrund; im Mittelpunkt stehen die informatischen Strukturen selbst.

Für die Schule bietet dieser Ansatz mehrere Anknüpfungspunkte. Einerseits kann er eingesetzt werden, wenn digitale Endgeräte nur eingeschränkt verfügbar sind. Andererseits eröffnet die analoge Darstellung informatischer Konzepte auch unabhängig von technischen Rahmenbedingungen didaktische Möglichkeiten. Unplugged Programming kann somit nicht nur als Ersatz für digitales

Programmieren, sondern als eigenständige Lernform verstanden werden (Threekunprapa & Yasri, 2020).

Der vorliegende Beitrag setzt an dieser Perspektive an und untersucht das Unterrichtsmaterial „Programmieren mit der Maus“ anhand einer konkreten Unterrichtserprobung. Das Material führt Lernende über spielerische Aufgaben an grundlegende Programmierkonzepte heran. Die Progression reicht von einem analogen Einstieg mit Befehlskarten über digitale Lernspiele bis hin zur Möglichkeit, eigene Programme beziehungsweise Lernspiele zu entwickeln (Lüdeke, 2019). Für die vorliegende Unterrichtserprobung wurde speziell die analoge Phase aufgegriffen und in ein eigenes fächerübergreifendes Unterrichtskonzept integriert.

Das Unterrichtskonzept „Vom Weg zur Programmierung“ wurde in einer ersten Klasse einer Mittelschule mit 26 Schülerinnen und Schülern und mit Deutsch als Fremdsprache-/Deutsch als Zweitsprache-Schwerpunkt (DaF/DaZ-Schwerpunkt) im Teamteaching der Fächer Deutsch und Digitale Grundbildung durchgeführt. Die Unterrichtseinheit umfasste zwei Doppelstunden. In der ersten Doppelstunde wurden Wegbeschreibungen als sprachliche Handlung erarbeitet. Die zweite Doppelstunde knüpfte daran an und übertrug die Erfahrungen auf das analoge Programmieren.

Ausgangspunkt war die Überlegung, dass zwischen einer Wegbeschreibung und einem Algorithmus eine strukturelle Verbindung besteht: Beide beschreiben eine Abfolge von Handlungen, die für eine andere Person beziehungsweise ein ausführendes System nachvollziehbar sein muss. Eine Wegbeschreibung kann damit als

lebensweltlich vertrautes Modell für algorithmisches Denken dienen.

Die zentrale Fragestellung des Beitrags lautet daher:

Welche Potenziale und Herausforderungen zeigen sich beim Einsatz von Unplugged Programming in der 5. Schulstufe, wenn das Material „Programmieren mit der Maus“ in ein fächerübergreifendes Unterrichtskonzept von Deutsch und Digitaler Grundbildung eingebettet wird?

Dabei geht es nicht um eine empirische Wirksamkeitsprüfung des Materials. Vielmehr wird die konkrete Unterrichtserprobung als Fallbeispiel analysiert. Ziel ist es, aus den beobachteten Lernprozessen und Herausforderungen didaktische Konsequenzen für zukünftige Lehrpersonen abzuleiten.

2. Unplugged Programming als Zugang zu algorithmischem Denken

2.1 Vom konkreten Handeln zum abstrakten Konzept

Ein wesentlicher Vorteil von Unplugged Programming liegt in der Möglichkeit, informatische Konzepte auf einer konkreten Handlungsebene zu erschließen. Ein Algorithmus muss nicht zunächst definiert werden, sondern kann beispielsweise als Folge von Bewegungsanweisungen erfahren werden. Die Lernenden planen einen Weg, legen Befehle in einer bestimmten Reihenfolge aus und beobachten anschließend, welche Konsequenzen diese Befehlsfolge hat.

Dadurch entsteht eine unmittelbare Verbindung zwischen Handlung und Ergebnis. Wird ein Befehl falsch gewählt oder an der falschen Stelle platziert, erreicht die Spielfigur das Ziel nicht. Der Fehler wird sichtbar und kann gemeinsam analysiert werden.

Für den Einstieg in algorithmisches Denken ist dies von besonderer Bedeutung. Die Lernenden müssen zunächst nicht mit einer Programmiersprache umgehen. Sie müssen vielmehr verstehen, dass eine Aufgabe in einzelne Schritte zerlegt werden kann und dass die Reihenfolge dieser Schritte für das Ergebnis relevant ist.

Unplugged Programming schafft damit eine Zwischenebene zwischen Alltagshandlung und digitalem Programmcode. Diese Zwischenebene kann dazu beitragen, abstrakte informatische Konzepte zunächst unabhängig von technischen Oberflächen zu thematisieren (Threekunprapa & Yasri, 2020).

2.2 Algorithmisches Denken als Problemlöseprozess

Im Rahmen des Unterrichtskonzepts wurde algorithmisches Denken vorranging über vier miteinander verbundene Prozesse erschlossen:

1. *Zerlegen*: Ein komplexerer Weg wird in einzelne Schritte aufgeteilt.
2. *Sequenzieren*: Die einzelnen Schritte werden in eine sinnvolle Reihenfolge gebracht.
3. *Überprüfen*: Die erstellte Befehlsfolge wird ausgeführt und auf ihre Funktionalität überprüft.
4. *Optimieren*: Eine funktionierende Lösung wird gegebenenfalls verkürzt oder verbessert.

Diese Prozesse entsprechen grundlegenden Tätigkeiten beim Programmieren. Entscheidend ist dabei, dass sie zunächst nicht unter Verwendung einer Programmiersprache stattfinden müssen.

Gerade die Möglichkeit, Fehler unmittelbar sichtbar zu machen, bietet einen wichtigen Zugang zum Debugging (Michaeli & Romeike, 2019). Wenn die Maus nicht am vorgesehenen Ziel ankommt, muss die Befehlsfolge überprüft werden. Die Lernenden können einzelne Anweisungen verändern, entfernen oder ergänzen und anschließend erneut testen.

Der Fehler wird dadurch nicht als Scheitern verstanden, sondern als Information über die Qualität des eigenen Lösungsweges. Diese Perspektive ist für die Entwicklung einer produktiven Haltung gegenüber Problemlöseprozessen von Bedeutung.

3. Das Unterrichtsmaterial „Programmieren mit der Maus“

3.1 Aufbau und Lernprogression

Das Unterrichtsmaterial „Programmieren mit der Maus“ verfolgt einen spielerischen Zugang zu grundlegenden Programmierkonzepten. Die Lernenden werden schrittweise von einfachen Befehlsfolgen zu komplexeren Programmierstrukturen geführt.

Besonders relevant für die vorliegende Untersuchung ist die analoge Phase des Materials. Beim „Maus-Hindernislauf“ wird eine Spielfigur mithilfe von Befehlskarten über ein Spielfeld gesteuert. Ziel ist es, die Maus so zu programmieren, dass sie einen vorgegebenen Zielpunkt erreicht und dabei Hindernisse überwindet.

Bereits diese einfache Situation enthält mehrere grundlegende informatische Konzepte. Die Lernenden müssen Bewegungsbefehle auswählen, in eine geeignete Reihenfolge bringen und anschließend überprüfen. Darüber hinaus werden wiederkehrende Befehlsfolgen als Schleifen beziehungsweise zusammengefasste Abläufe thematisiert. Funktionen können als Kombination mehrerer Befehle eingeführt werden.

Eine Besonderheit des Materials besteht darin, dass die Befehlskarten eine sichtbare Repräsentation eines Programms darstellen. Das Programm liegt gewissermaßen vor der ausführenden Figur. Die Lernenden können es betrachten, verändern und gemeinsam analysieren.

Die Progression des Gesamtmaterials geht über diese analoge Phase hinaus. Im Anschluss ist eine Überleitung zu einer digitalen Programmierumgebung vorgesehen. Lernspiele führen schrittweise in weitere Programmierkonzepte ein; Aufgabenkarten ermöglichen die Anwendung und Festigung, während Kontrollkarten eine selbstständige Überprüfung der Ergebnisse unterstützen. Langfristig können Lernende eigene kleine Programme beziehungsweise Lernspiele entwickeln.

Damit ist eine Lernprogression angelegt, die sich vereinfacht als *Handeln → Visualisieren → Programmieren → Gestalten* beschreiben lässt.

Für die vorliegende Unterrichtserprobung wurde diese Progression bewusst reduziert. Im Mittelpunkt stand zunächst die Frage,

welche informatischen Grundideen bereits in der analogen Phase erschlossen werden können.

4. Vom Weg zur Programmierung: Konzeption der Unterrichtseinheit

4.1 Fächerübergreifender Ausgangspunkt

Das Unterrichtskonzept „Vom Weg zur Programmierung“ entstand aus der Verbindung zweier fachlicher Perspektiven. Im Deutschunterricht sollte die Textsorte beziehungsweise kommunikative Handlung der Wegbeschreibung erarbeitet werden. In der Digitalen Grundbildung sollte daran anschließend die Idee des Algorithmus entwickelt werden.

Die Verbindung basiert auf einer strukturellen Gemeinsamkeit: Eine Wegbeschreibung besteht aus aufeinanderfolgenden Handlungsanweisungen. Damit jemand den beschriebenen Weg tatsächlich ausführen kann, müssen diese Anweisungen ausreichend verständlich und in einer sinnvollen Reihenfolge formuliert sein.

Für algorithmische Anweisungen gilt dies in besonderer Weise. Ein Computer oder eine Spielfigur kann eine unklare Anweisung nicht durch Kontextwissen oder Nachfragen kompensieren. Die Anweisung muss so formuliert beziehungsweise codiert sein, dass sie eindeutig ausgeführt werden kann.

Die Wegbeschreibung wurde somit als didaktische Brücke genutzt. Die Lernenden begegneten dem Prinzip schrittweiser An-

weisungen zunächst in einem sprachlichen und lebensweltlichen Kontext und übertrugen es anschließend auf eine informatische Problemstellung.

4.2 Lerngruppe und Rahmenbedingungen

Die Unterrichtserprobung fand in einer ersten Klasse einer Mittelschule mit 26 Schülerinnen und Schülern statt. In der Lerngruppe lag ein DaF/DaZ-Schwerpunkt vor. Die Unterrichtseinheit wurde im Teamteaching der Fächer Deutsch und Digitale Grundbildung durchgeführt.

Die sprachlichen Voraussetzungen der Lerngruppe waren für die Planung von besonderer Bedeutung. Einerseits bot die Arbeit mit Wegbeschreibungen einen sinnvollen Anlass zur Erweiterung und Anwendung des Wortschatzes. Andererseits musste berücksichtigt werden, dass räumliche und algorithmische Anweisungen sprachlich präzise verstanden und formuliert werden müssen.

Das Unterrichtsdesign verband daher sprachliche Unterstützung mit visuellen und handelnden Elementen.

5. Durchführung

5.1 Erste Phase: Wegbeschreibungen

Die erste Doppelstunde diente der Vorbereitung der späteren Programmierphase. Zunächst wurden zentrale Richtungsbegriffe erarbeitet und visualisiert. Die Lehrperson modellierte anschlie-

ßend eine Wegbeschreibung und machte dabei insbesondere die Reihenfolge einzelner Schritte sichtbar.

Die Lernenden formulierten danach eigene Wegbeschreibungen. Dabei wurden sprachliche Strukturen durch Satzstarter und visuelle Unterstützung vorentlastet. In der anschließenden Sicherungsphase wurden Beispiele gemeinsam betrachtet und hinsichtlich ihrer Verständlichkeit diskutiert.

Eine zentrale Erkenntnis entstand aus der Frage, was passiert, wenn eine Wegbeschreibung zwar ungefähr verständlich, aber nicht ausreichend präzise ist. Die Lernenden konnten feststellen, dass eine andere Person eine Anweisung unterschiedlich interpretieren kann.

Damit war die Grundlage für den späteren informatischen Transfer geschaffen.

5.2 Zweite Phase: Transfer zum Programmieren

Zu Beginn der zweiten Doppelstunde wurde die Verbindung explizit hergestellt:

> Was hat eine Wegbeschreibung mit Programmieren zu tun?

Die Schülerinnen und Schüler sammelten zunächst Gemeinsamkeiten. Im Zentrum standen klare Schritte, Reihenfolge und eindeutige Anweisungen.

Anschließend wurde der „Maus-Hindernislauf“ zunächst am großen Spielfeld eingeführt. Die Lehrperson modellierte eine erste

Befehlsfolge und ließ die Klasse beobachten, wie die Maus auf die einzelnen Befehle reagiert.

Danach arbeiteten die Schülerinnen und Schüler in Partnerarbeit mit eigenen Spielfeldern und Befehlskarten. Zunächst wurden Hindernisse überwunden. Anschließend wurden wiederkehrende Befehlsfolgen betrachtet und Schleifen thematisiert.

In einer weiteren Aufgabe übernahm jeweils ein Team die Rolle der „programmierenden" Personen, während ein anderes Team die Befehlsfolge ausführte. Dadurch wurde überprüfbar, ob die formulierten Anweisungen tatsächlich eindeutig waren.

Abschließend wurde das Konzept der Funktion eingeführt und die verschiedenen Elemente miteinander verbunden. Die Lernenden sollten nicht nur eine funktionierende Lösung finden, sondern auch überlegen, wie eine Befehlsfolge effizienter gestaltet werden kann.

6. Analyse der Unterrichtserprobung

Die Analyse der Durchführung orientiert sich an vier zentralen Lerngelegenheiten, die für zukünftige Einsätze von Unplugged Programming besonders relevant erscheinen: *Sequenzierung und Eindeutigkeit, Debugging, Mustererkennung und Optimierung* sowie sprachliches und fächerübergreifendes Lernen. Ergänzend werden räumliche und organisatorische Herausforderungen betrachtet.

6.1 Sequenzierung und Eindeutigkeit: Der Algorithmus wird sichtbar

Eine wesentliche Stärke des analogen Materials liegt in der Sichtbarkeit der Befehlsfolge. Die Schülerinnen und Schüler legen die einzelnen Befehle physisch vor sich aus. Dadurch wird ein Programm als strukturierte Abfolge von Anweisungen greifbar.

Die Bedeutung der Reihenfolge konnte unmittelbar erfahren werden. Eine einzelne Anweisung kann für sich korrekt sein und dennoch zu einem falschen Ergebnis führen, wenn sie an der falschen Position steht.

Gerade dieser Zusammenhang ist für Anfängerinnen und Anfänger bedeutsam. Im Alltag werden sprachliche Äußerungen häufig kontextabhängig interpretiert. Wenn jemand sagt „Geh dort hinüber“, können situative Hinweise helfen, die Aussage zu verstehen. Ein Algorithmus kann sich auf diese Art von implizitem Kontext jedoch nicht verlassen.

Das analoge Spielfeld macht diese Differenz sichtbar. Die Lernenden erleben, dass die Maus nur das „ausführt“, was durch die Befehle festgelegt ist.

Damit entsteht eine zentrale Einsicht:

Ein Ziel zu kennen reicht nicht aus. Ein Algorithmus muss den Weg zum Ziel in ausführbare Einzelschritte übersetzen.

Diese Erfahrung stellt eine wichtige Grundlage für das spätere Verständnis von Programmcode dar.

6.2 Debugging: Fehler als Lerngelegenheiten

Eine zweite zentrale Lerngelegenheit ergibt sich aus dem Umgang mit fehlerhaften Befehlsfolgen.

Wenn die Maus das Ziel nicht erreicht, liegt ein Problem im entwickelten Programm vor. Die Lernenden können nun einzelne Befehle überprüfen und verändern. Der Prozess folgt damit einer einfachen Schleife:

Planen → Ausführen → Prüfen → Fehler erkennen → Verändern → erneut ausführen.

Diese Struktur entspricht einem grundlegenden Vorgehen beim Programmieren.

Bemerkenswert ist dabei die soziale Dimension. Im Peer-Compiler-Prinzip wird eine Person zum „ausführenden System“. Dadurch wird die Frage nach der Eindeutigkeit einer Anweisung unmittelbar relevant. Wenn die ausführende Person eine andere Handlung vornimmt als erwartet, muss geklärt werden, ob die Anweisung tatsächlich eindeutig war.

Die Lernenden erhalten damit eine Rückmeldung, die über „richtig“ oder „falsch“ hinausgeht. Sie müssen begründen, warum ein Programm nicht funktioniert und an welcher Stelle die Ursache liegt.

Für zukünftige Unterrichtsszenarien erscheint es daher sinnvoll, Fehler nicht möglichst schnell durch die Lehrperson zu korrigie-

ren. Vielmehr sollte bewusst Zeit für die Analyse der Befehlsfolge eingeplant werden.

6.3 Mustererkennung, Schleifen und Funktionen

Eine weitere Stärke des Materials liegt darin, dass wiederkehrende Abläufe sichtbar gemacht werden können.

Wenn mehrere Befehle in gleicher Weise wiederholt auftreten, entsteht die Frage, ob diese Wiederholung verkürzt oder zusammengefasst werden kann. Dadurch kann die Idee einer Schleife aus einer konkreten Handlungssituation heraus entwickelt werden.

Auch Funktionen können zunächst als Zusammenfassung mehrerer Einzelschritte verstanden werden. Eine Befehlsfolge wird zu einer benannten Einheit, die später erneut aufgerufen werden kann.

Für die Lernenden ergibt sich dadurch ein Wechsel der Perspektive:

Zunächst steht die Frage im Vordergrund:

> Welche Befehle brauche ich?

Später lautet die Frage:

> Welche Struktur steckt hinter diesen Befehlen?

Dieser Wechsel vom einzelnen Befehl zur Struktur ist ein wichtiger Schritt hin zu abstrakterem algorithmischem Denken.

7. Sprache als Bestandteil des Programmierens

Die fächerübergreifende Konzeption erwies sich vor allem deshalb als interessant, weil die sprachliche Präzision nicht lediglich vorbereitende Funktion hatte. Sprache wurde selbst zu einem Bestandteil des informatischen Lernprozesses.

7.1 Wegbeschreibung als algorithmisches Modell

Die Wegbeschreibung stellt ein alltagsnahes Beispiel für sequenzielle Anweisungen dar. Schülerinnen und Schüler wissen aus ihrem Alltag grundsätzlich, was eine Wegbeschreibung leisten soll. Dadurch entsteht eine inhaltliche Anschlussfähigkeit, die bei der Einführung eines abstrakten Begriffs wie „Algorithmus“ möglicherweise fehlen würde.

Die Übertragung kann schrittweise erfolgen:

Wegbeschreibung: „Gehe geradeaus. Biege rechts ab. Gehe bis zur Kreuzung“.

Algorithmische Perspektive: „Eine Handlung wird in einzelne, geordnete und ausführbare Anweisungen zerlegt“.

Die Schülerinnen und Schüler müssen dabei erkennen, dass eine Wegbeschreibung nicht nur einen Zielort nennen darf. Sie muss den Weg zwischen Ausgangspunkt und Ziel beschreiben.

7.2 Sprachliche Anforderungen in einer DaF/DaZ-Lerngruppe

Die Durchführung zeigte zugleich, dass Unplugged Programming keineswegs automatisch sprachlich niedrigschwellig ist.

Begriffe wie „links“, „rechts“, „geradeaus“, „drehen“, „Schritt“, „vor“, „hinter“ oder „wiederholen“ müssen sicher verstanden werden. Darüber hinaus müssen die Lernenden in der Lage sein, entsprechende Anweisungen selbst zu formulieren.

Für eine Lerngruppe mit DaF/DaZ-Schwerpunkt ist daher eine sprachsensible Planung erforderlich. Visuelle Darstellungen, Pfeile, Bewegungen, Satzstarter und modellierte Beispiele können dazu beitragen, die sprachlichen Anforderungen zu reduzieren.

Gleichzeitig bietet gerade das Programmieren einen authentischen Anlass, sprachliche Präzision zu üben. Die Lernenden erfahren unmittelbar, weshalb eine Formulierung nicht eindeutig genug sein kann.

Damit entsteht eine interessante Verbindung:

Sprachliche Präzision wird nicht nur geübt, weil sie ein Lernziel des Deutschunterrichts ist, sondern weil sie für die Lösung des informatischen Problems notwendig ist.

8. Räumliche Orientierung als zusätzliche Voraussetzung

Neben sprachlichen Anforderungen zeigte sich eine weitere Herausforderung: die räumliche Orientierung auf dem Spielfeld.

Für einige Lernende war es nicht selbstverständlich, die Position der Spielfigur im Gitternetz zu erfassen und die Bewegungsrichtung korrekt zu berücksichtigen. Insbesondere die Verbindung von Drehen und anschließender Bewegung stellte eine zusätzliche kognitive Anforderung dar.

Diese Beobachtung ist didaktisch relevant, weil die Aufgabe auf den ersten Blick sehr einfach erscheint. Tatsächlich müssen mehrere Informationen gleichzeitig verarbeitet werden:

- Wo befindet sich die Spielfigur?
- In welche Richtung zeigt sie?
- Was bedeutet der nächste Befehl?
- Welche Position ergibt sich daraus?
- Welcher Befehl führt anschließend zum Ziel?

Für zukünftige Unterrichtsplanungen empfiehlt es sich daher, räumliche Orientierung gegebenenfalls vorab zu thematisieren. Bewegungsübungen im Klassenraum oder ein großes Bodenraster können dazu beitragen, die kognitive Komplexität des späteren Spielfeldes zu reduzieren.

Gerade bei heterogenen Lerngruppen sollte die Lehrperson daher nicht davon ausgehen, dass die informatische Aufgabe die einzige relevante Schwierigkeit darstellt.

9. Organisation und Klassenmanagement

Unplugged Programming reduziert zwar die technische Komplexität, erhöht aber teilweise den organisatorischen Aufwand.

Für die Durchführung mussten Spielfelder, Figuren und Befehlskarten vorbereitet und für mehrere Gruppen bereitgestellt werden. Bei einer Klassengröße von 26 Lernenden gewinnt daher die Frage nach Materialorganisation und Gruppengröße an Bedeutung.

Die Partnerarbeit erwies sich als besonders geeignet, weil sie eine aktive Beteiligung beider Lernender ermöglicht. Bei größeren Gruppen besteht hingegen die Gefahr, dass einzelne Schülerinnen und Schüler in eine beobachtende Rolle geraten.

Für zukünftige Einsätze bietet sich deshalb eine klare Rollenstruktur an, beispielsweise:

- Programmierer/in: entwickelt die Befehlsfolge.
- Compiler/Maus: führt die Befehle strikt aus.
- Debugger/in: beobachtet und sucht Fehler.
- Dokumentierer/in: hält die Lösung fest.

Die Rollen können anschließend gewechselt werden. Dadurch wird verhindert, dass einzelne Lernende dauerhaft dieselbe Funktion übernehmen.

Auch die Materialvorbereitung sollte bei der Planung berücksichtigt werden. Ein einmal hergestellter und dauerhaft verwendbarer Klassensatz kann den Aufwand für spätere Unterrichtseinheiten deutlich reduzieren.

10. Hinweise zur Adaption des Materials

Die Unterrichtserprobung legt nahe, dass „Programmieren mit der Maus“ für die 5. Schulstufe grundsätzlich anschlussfähig ist. Eine direkte Übernahme des Materials erscheint jedoch weniger sinnvoll als eine didaktisch begründete Adaption.

Für Lehrpersonen lassen sich zusammenfassend sieben Empfehlungen ableiten.

10.1 Mit einem vertrauten Problem beginnen

Der Einstieg sollte möglichst an vorhandenes Wissen anschließen. Wegbeschreibungen, Kochrezepte, Spielregeln oder Bauanleitungen können als Beispiele für strukturierte Handlungsanweisungen dienen.

10.2 Zunächst handeln, dann benennen

Begriffe wie Algorithmus, Sequenz, Schleife oder Funktion sollten nicht ausschließlich definitorisch eingeführt werden. Sinnvoller ist es, zunächst eine Handlungssituation zu schaffen und anschließend den beobachteten Prozess zu benennen.

10.3 Fehler ausdrücklich zulassen

Fehlerhafte Programme sind keine Störung des Unterrichts, sondern zentrale Lerngelegenheiten. Die Frage sollte daher nicht nur lauten:

> Ist die Lösung richtig?

sondern beispielsweise:

> An welcher Stelle funktioniert der Ablauf nicht und warum?

10.4 Unterschiedliche Lösungswege vergleichen

Wenn mehrere Lösungen möglich sind, sollte dies ausdrücklich thematisiert werden. Dadurch können die Lernenden erkennen, dass Programmieren nicht nur darin besteht, irgendeine funktionierende Lösung zu finden.

Interessant werden Fragen wie:

- Welche Lösung benötigt weniger Befehle?
- Welche Lösung ist leichter verständlich?
- Welche Lösung lässt sich besser verändern?
- Welche Befehlsfolge könnte als Schleife zusammengefasst werden?

10.5 Sprache gezielt unterstützen

Insbesondere bei DaF/DaZ-Lerngruppen sollten Satzstarter, Visualisierungen und Wortschatzhilfen von Beginn an vorgesehen werden.

Beispielsweise:

Gehe _____.

Drehe dich nach _____.

Danach _____.

Wiederhole _____-mal.

Damit kann sprachliche Unterstützung geleistet werden, ohne die informatische Problemstellung vorwegzunehmen.

10.6 Räumliche Anforderungen diagnostizieren

Vor dem eigentlichen Programmieren sollte überprüft werden, ob die Lernenden mit dem verwendeten Koordinaten- beziehungsweise Bewegungsprinzip vertraut sind.

10.7 Den Transfer zum digitalen Programmieren vorbereiten

Die analogen Befehle sollten später mit digitalen Programmierblöcken in Beziehung gesetzt werden. So kann beispielsweise ein

Pfeil-Befehl mit einem entsprechenden Block in einer blockbasierten Programmierumgebung verglichen werden.

Damit wird deutlich:

Das analoge Programm ist nicht das Ende des Programmierens, sondern ein Modell dafür, was später digital codiert wird.

11. Unplugged Programming als eigenständiger Zugang – nicht nur als Ersatzlösung

Ein zentraler Befund der Unterrichtserprobung betrifft die Frage, wie Unplugged Programming grundsätzlich eingeordnet werden sollte.

Wird der Ansatz ausschließlich als Möglichkeit verstanden, Programmieren ohne Computer zu unterrichten, bleibt ein wesentlicher Teil seines didaktischen Potenzials unberücksichtigt. Die analoge Darstellung ermöglicht nämlich eine Konzentration auf algorithmische Strukturen, ohne dass gleichzeitig technische Bedienhandlungen bewältigt werden müssen.

Dies kann insbesondere in der Einstiegsphase hilfreich sein. Die Schülerinnen und Schüler können sich zunächst darauf konzentrieren,

- Probleme zu zerlegen,
- Abläufe zu planen,
- Befehle zu ordnen,
- Fehler zu erkennen,

- Muster zu finden und
- Lösungen zu optimieren.

Erst anschließend wird die Frage relevant, wie diese Strukturen in einer Programmiersprache umgesetzt werden.

Unplugged Programming kann damit als konzeptuelle Vorstufe des digitalen Programmierens verstanden werden. Gleichzeitig ist es aber auch als eigenständiges Lernarrangement sinnvoll, wenn beispielsweise die Zielsetzung nicht darin besteht, unmittelbar eine Programmiersprache zu erlernen, sondern algorithmische Denkweisen aufzubauen.

12. Diskussion

Die Unterrichtserprobung zeigt, dass das Material „Programmieren mit der Maus“ mehrere Lerngelegenheiten für die Entwicklung algorithmischen Denkens bereitstellt. Besonders deutlich wurden Sequenzierung, Eindeutigkeit und Debugging. Die Lernenden konnten Befehlsfolgen nicht nur betrachten, sondern handelnd erzeugen und unmittelbar überprüfen.

Die zentrale Stärke des Materials liegt damit in seiner Externalisierung von Denkprozessen. Ein Programm befindet sich nicht nur „im Kopf“ der Lernenden, sondern liegt sichtbar als Reihe von Karten vor. Es kann verschoben, verändert, verkürzt und mit anderen Lösungen verglichen werden. Dadurch werden Denkprozesse für Lernende und Lehrpersonen beobachtbar.

Gleichzeitig darf aus diesen Beobachtungen keine pauschale Aussage über die Wirksamkeit des Materials abgeleitet werden. Die Unterrichtserprobung war zeitlich begrenzt und wurde nicht durch standardisierte Kompetenzmessungen begleitet. Die vorliegenden Erkenntnisse sind daher als didaktische Reflexion einer konkreten Unterrichtssituation zu verstehen.

Gerade diese Perspektive macht jedoch sichtbar, welche Bedingungen für einen erfolgreichen Einsatz relevant sind. Das Material allein erzeugt noch kein informatisches Verständnis. Entscheidend ist die didaktische Rahmung.

Die Lehrperson übernimmt dabei drei Funktionen:

Erstens: Sie schafft eine Verbindung zwischen Handlung und informatischem Begriff.

Zweitens: Sie initiiert Reflexion über Lösungswege und Fehler.

Drittens: Sie unterstützt den Transfer vom analogen Modell auf digitale Programmierumgebungen.

Ohne diese Reflexions- und Transferleistung besteht die Gefahr, dass das Unterrichtsgeschehen auf das erfolgreiche Bewältigen eines Spiels reduziert bleibt.

Für die 5. Schulstufe erscheint deshalb eine Unterrichtsstruktur besonders sinnvoll, die drei Ebenen miteinander verbindet:

Handeln – Reflektieren – Abstrahieren

Auf der Handlungsebene führen die Lernenden Befehle aus. Auf der Reflexionsebene diskutieren sie, warum ein Ablauf funktio-

niert oder nicht funktioniert. Auf der Abstraktionsebene werden die Erfahrungen mit informatischen Begriffen wie Algorithmus, Sequenz, Schleife oder Funktion verbunden.

13. Schlussfolgerungen für die Unterrichtspraxis

Aus der Unterrichtserprobung lassen sich mehrere Schlussfolgerungen für zukünftige Lehrpersonen ableiten.

Erstens eignet sich Unplugged Programming besonders für den Einstieg in algorithmisches Denken, wenn die Lernenden zunächst konkrete Erfahrungen mit Befehlsfolgen sammeln sollen.

Zweitens sollte die analoge Lernumgebung nicht isoliert eingesetzt werden. Eine vorbereitende oder anschließende sprachliche, fachliche oder digitale Einbettung erhöht das Potenzial des Materials.

Drittens ist die Fehlerkultur entscheidend. Fehlerhafte Befehlsfolgen sollten nicht möglichst schnell korrigiert werden. Vielmehr sollten Lernende dazu angeleitet werden, Fehler systematisch zu lokalisieren und Lösungen selbstständig zu verbessern.

Viertens bietet die Verbindung mit Deutsch besondere Möglichkeiten. Wegbeschreibungen, Rezepte, Spielregeln oder Arbeitsanweisungen können als Ausgangspunkt dienen, um die Gemeinsamkeiten zwischen Sprache und algorithmischer Kommunikation zu thematisieren.

Fünftens sollten sprachliche und räumliche Voraussetzungen berücksichtigt werden. Gerade in heterogenen Lerngruppen kann

die eigentliche informatische Aufgabe durch zusätzliche Anforderungen überlagert werden.

Sechstens ist eine Progression vom Analogen zum Digitalen sinnvoll. Nach der Arbeit mit Befehlskarten kann beispielsweise ein entsprechendes Programm in einer blockbasierten Umgebung erstellt werden. Die Lernenden erkennen dadurch, dass die digitalen Programmierblöcke nicht völlig neue Inhalte darstellen, sondern eine andere Repräsentation bereits bekannter Strukturen.

14. Fazit

Die Unterrichtserprobung des Konzepts „Vom Weg zur Programmierung“ verdeutlicht das Potenzial von Unplugged Programming für die Digitale Grundbildung in der 5. Schulstufe. Das Material „Programmieren mit der Maus“ ermöglicht es, zentrale Prinzipien algorithmischen Denkens zunächst unabhängig von digitalen Endgeräten handelnd zu erschließen.

Besonders gewinnbringend erwiesen sich die Möglichkeiten, Sequenzen sichtbar zu machen, die Bedeutung eindeutiger Anweisungen zu erfahren, Fehler systematisch zu suchen und wiederkehrende Strukturen zu erkennen. Die Schülerinnen und Schüler konnten dabei erleben, dass Programmieren nicht primär aus dem Bedienen eines Computers besteht, sondern aus Planen, Strukturieren, Überprüfen und Optimieren.

Die fächerübergreifende Verbindung mit dem Deutschunterricht erwies sich als besonders anschlussfähig. Die Wegbeschreibung

fungierte als lebensweltlich vertrautes Modell für einen Algorithmus und ermöglichte eine Verbindung von sprachlicher Präzision und informatischem Denken. Gerade in einer DaF/DaZ-geprägten Lerngruppe zeigte sich jedoch auch, dass sprachliche Unterstützung einen festen Bestandteil der Planung darstellen muss.

Gleichzeitig wurden Grenzen sichtbar. Räumliche Orientierung, sprachliche Voraussetzungen, Gruppengröße und Materialorganisation können die Bearbeitung beeinflussen. Unplugged Programming ist daher keineswegs voraussetzungslos. Der Verzicht auf digitale Geräte reduziert zwar technische Anforderungen, ersetzt diese aber teilweise durch organisatorische und didaktische Anforderungen.

Für zukünftige Lehrpersonen ergibt sich daraus eine zentrale Konsequenz: Unplugged Programming sollte weder als vereinfachte Version des „richtigen“ Programmierens noch als bloße Ersatzlösung für fehlende Geräte verstanden werden. Sein besonderer Wert liegt vielmehr darin, informatische Konzepte auf einer konkreten Handlungsebene zugänglich zu machen und damit eine Brücke zwischen Alltagshandlung, algorithmischem Denken und digitaler Programmierung zu schaffen.

Das Beispiel „Vom Weg zur Programmierung“ zeigt, dass diese Brücke auch fächerübergreifend gestaltet werden kann. Eine Wegbeschreibung wird zum Algorithmus, die Befehlskarte zum Programmierbaustein und der Fehler im Spielfeld zum Anlass für Debugging. Gerade diese Übersetzungsprozesse zwischen Sprache, Handlung und Informatik machen Unplugged Programming

zu einem interessanten Ansatz für die Digitale Grundbildung der Sekundarstufe I.

Die analoge Phase kann somit als eigenständiger Lernraum verstanden werden, in dem Lernende zunächst erfahren, was Programmieren bedeutet, bevor sie sich damit auseinandersetzen, wie Programmieren digital umgesetzt wird. Für die Unterrichtspraxis liegt darin eine wesentliche Chance: Digitale Bildung beginnt nicht notwendigerweise am Bildschirm.

Literatur

Bundesministerium für Bildung. (o. J.). *Empfehlungen zur Nutzung digitaler Technologie an Schulstandorten*. https://sb726cd41480af32d.jimcontent.com/download/version/1664560472/module/10144288085/name/EmpfehlungenZurNutzungDigitalerTechnologieAnSchulstandorten_2018.pdf

Education Group GmbH. (2025). 9. *Oö. Jugend-Medien-Studie 2025: Das Medienverhalten der 11- bis 18-Jährigen*. https://www.edugroup.at/fileadmin/user_upload/50_Forschung/Medienstudien/Jugend-Medien-Studie/2025/edugroup-jugend-medien-studie2025-jugendliche.pdf

Fehrmann, R., & Zeinz, H. (2021). Erst denken, dann lenken: Computational Thinking: Problemlösekompetenz erwerben mithilfe von Lernrobotern. *On lernen digital*, (7), 22–25. https://www.friedrich-verlag.de/friedrich-plus/schule-paedagogik/digitale-schule/digital-unterrichten/erst-denken-dann-lenken-10654

Lüdeke, C. (2019). *Programmieren mit der Maus | Unterricht*. Planet Schule. https://www.planet-schule.de/thema/programmieren-mit-der-maus-unterricht-100.html

Michaeli, T., & Romeike, R. (2019). Debuggen im Unterricht – Ein systematisches Vorgehen macht den Unterschied. In *Lecture Notes in Informatics (LNI)*. Gesellschaft für Informatik. https://computingeducation.de/pub/2019_Michaeli-Romeike_INFOS19.pdf

Threekunprapa, A., & Yasri, P. (2020). Patterns of Computational Thinking Development while Solving Unplugged Coding Activities Coupled with the 3S Approach for Self-Directed Learning. European Journal of Educational Research, 9(3), 1025–1045. https://doi.org/10.12973/eu-jer.9.3.1025