Marcel Klinger

Universität Duisburg-Essen, marcel.klinger@uni-due.de

Algorithmisches Denken im Mathematikunterricht fördern: Ein Praxisbeitrag aus der Lehramtsausbildung Mathematik

Die Lehrveranstaltung „Algorithmische Mathematik" an der Universität Duisburg-Essen hat das Ziel, Studierende dazu zu befähigen, den eigenen Mathematikunterricht algorithmisch-gehaltvoll zu gestalten. Hierzu werden mathematische Algorithmen verschiedener mathematischer Teildisziplinen thematisiert und mithilfe einer blockbasierten Programmiersprache auch praktisch erschlossen. Der vorliegende Beitrag beschreibt die Konzeption des Lehrformats und hält die bei mehrmaliger Durchführung gewonnenen praktischen Erfahrungen fest.

Algorithmen haben eine immense Bedeutung für die heutige Gesellschaft. Um dem Anspruch einer allgemeinbildenden Schule gerecht zu werden, müssen Algorithmen mit all ihren Facetten zweifelsohne Bestandteil einer grundständigen Schullaufbahn sein. In diesem Kontext kann gerade das Fach Mathematik mit seinen mannigfaltigen Fachinhalten algorithmischer Natur einen besonderen Beitrag leisten.

Damit schulischer Mathematikunterricht der damit einhergehenden Verantwortung gerecht werden kann, muss sichergestellt sein, dass auch (angehende) Lehrkräfte über die notwendigen Kompetenzen verfügen, die Rolle von Algorithmen im eigenen Unterricht entsprechend hervorzuheben. Kurzum: Angehende Mathematiklehrkräfte müssen selbst über die Fähigkeit verfügen, *algorithmisch zu denken*.

Um dies zu gewährleisten, wurde an der Universität Duisburg-Essen ein entsprechendes Wahlpflichtmodul „Algorithmische Mathematik" konzipiert und mehrmalig durchgeführt. Die Vorstellung der Konzeption und Durchführung dieses Moduls soll den Fokus dieses Beitrags darstellen. Zuvor werden die Rollen von Algorithmen im Mathematikunterricht und der hiermit in Zusammenhang stehende Begriff des „Algorithmischen Denkens" jedoch genauer beleuchtet.

1. Algorithmen *im* Mathematikunterricht

Die Bedeutung von Algorithmen als Gegenstand der Mathematik sowie als Werkzeug innerhalb der Disziplin ist neben ihrer gesamtgesellschaftlichen Relevanz unbestritten. Entsprechend lassen sich algorithmische Strukturen auch im schulischen Mathematikunterricht vielfältig wiederfinden (Ziegenbalg et al., 2016). Das Spektrum reicht hierbei von schriftlichen Rechenverfahren der Grundschule bis zum Gaußschen Eleminationsverfahren oder einer strukturierten Kurvendiskussion der Oberstufe. Obwohl es somit reichlich Gelegenheiten gäbe, werden Algorithmen innerhalb schulischer Curricula nicht explizit oder nur stark unterrepräsentiert thematisiert (vgl. Stephens, 2018).

Vor dem Hintergrund ihrer fachbezogenen wie allgemein-gesellschaftlichen Bedeutung erscheint dies paradox, dränge doch mit ihrer unterrichtlichen Erfassung gleichzeitig auch ein hochaktuelles Thema in den Unterricht, das potenziell einen lebensweltlichen und sinnstiftenden Bezug zwischen mathematischen Inhalten und der Welt herzustellen vermag. In diesem Zusammenhang besteht somit eine Wechselwirkung zwischen *Algorithmischen Denken* (oder allgemeiner *Digital Literacy*) auf der einen und der Förderung eines tieferen mathematischen Verständnisses auf der anderen Seite (vgl. Stephens & Kadijevich, 2020).

1.1 Algorithmisches Denken

Während der viel ältere Begriff des „Algorithmus" gut konturiert und vielfach in voneinander nur in einzelnen Aspekten abweichenden Definitionen konsolidiert ist (s. hierzu z.B. Oldenburg, 2022; Knuth, 1997), befindet sich der Begriff „Algorithmisches Denken" sowie der verwandte und von Papert (1980) geprägte Begriff „Computational Thinking" im Mittelpunkt vielfältiger Diskussionen innerhalb der wissenschaftlichen Community. Vor diesem Hintergrund greifen Stephens und Kadijevich (2020) unterschiedliche Definitionen und Beschreibungen beider Konzepte auf und synthetisieren diese. Demnach ist Algorithmisches Denken erforderlich, „whenever one has to comprehend, test, improve, or design an algorithm […]" (ebd., S. 118). Andere Autoren, etwa Kortenkamp und Labert (2015) oder Oldenburg (2022), beschreiben das Konzept entsprechend dieses Umstands in Analogie zu Vollrath (1989) als „eine Denkweise, die typisch für den Umgang mit Algorithmen ist" (Kortenkamp & Lambert, 2015).

Der Synthese von Stephens und Kadijevich folgend ist diese „Denkweise" von den drei Eckpfeilern „decomposition" (z. B. das strategische Zerlegen von Problemen), „abstraction" (z. B. kognitive Analyse von Daten und Mustererkennung) und „algorithmization" (z. B. Strukturieren und Formulieren einzelner Verfahrensschritte zur Problembewältigung) geprägt.

Zusätzlich zu diesen Eckpfeilern umfasst das Computational Thinking demnach mit „automation" einen weiteren Eckpfeiler, der hervorhebt, dass durch einen entsprechenden Denkprozess gewonnene Produkte mit Hilfe eines Informationsverarbeitungssystems (z. B. eines Computers) effizient verarbeitbar sein sollen und somit prinzipiell *automatisierbar* sind (vgl. Wing, 2010). Insofern stellt sich das Algorithmische Denken also als substanzielle Teilmenge des Computational Thinkings dar, was auch eine häufig in der Literatur vorzutreffende weitgehend synonyme Verwendung der beiden Begriffe erklärt. Der Begriff „Algorithmisches Denken" ist hingegen erfahrungsgemäß allein aufgrund seiner begrifflichen Nähe zum Algorithmusbegriff dahingehend eingängiger, dass in der entsprechenden Denkweise versierte Menschen mit ihm in natürlicher Weise ein klar umrissenes Concept Image (z.B. Tall & Vinner, 1981; vgl. Klinger, 2019) dessen verbinden, was er zu beschreiben versucht. Aus diesem Grund soll hier das Algorithmische Denken begrifflich im Vordergrund stehen.

1.2 Blockbasierte Programmierumgebungen

Algorithmisches Denken ist eine Denkart, für deren erfolgreiche Ausprägung neben einer theoretischen Fundierung zweifelsohne das praktische Arbeiten mit Algorithmen essenziell ist. Neben klassischen textbasierten Programmierkonzepten bieten blockbasierte Programmierumgebungen hier in allen Phasen des schulischen Mathematikunterrichts gerade für unerfahrene Schülerinnen und Schüler ein besonderes Potenzial. Im Gegensatz zu klassischen Programmierkonzepten muss dabei keine Syntax erlernt werden, da sich ein sequentieller Code automatisch durch das Aneinanderreihen vorgegebener elementarer „Programmblöcke“ ergibt. Dies birgt gleichzeitig den Vorteil, dass eine Bibliothek aller verfügbaren Befehle einsehbar ist und nicht zuletzt durch Trial-and-Error-artiges Ausprobieren verinnerlicht werden kann. Gleichzeitig wirken derartig visualisierte Programmierkonzepte auch einschränkend, etwa weil mit ihnen naturgemäß eine eher eingeschränkte Funktionalität einhergeht. Im Weiteren sei indes auf spezifischere Literatur verwiesen; etwa für den praktischen Einsatz blockbasierter Programmierkonzepte auf einen Beitrag von Eppendorf und Marx (2020) sowie für die mit solchen Konzepten einhergehenden informatikdidaktischen Herausforderungen auf einen Beitrag von Jatzlau und Romeike (2017).

2. Rahmenbedingungen und Grundannahmen zur Veranstaltungskonzeption

Zunächst ergeben sich für die Konzeption einer Veranstaltung, die das Augenmerk auf Algorithmen in Mathematik und vor allem im Mathematikunterricht lenkt, die folgenden organisatorischen Rahmenbedingungen aus der Struktur der vorhandenen Lehramtsstudiengänge: Im Studiengang für das Mathematik-Lehramt an Haupt-, Real-, Sekundar- und Gesamtschulen (HRSGe) an der Universität Duisburg-Essen sind durch die Studierenden insgesamt drei fachmathematische Vertiefungswahlmodule zu belegen. Von diesen sind zwei in der Bachelor-, eines ist in der Masterphase zu belegen. Die Wahlmodule haben Vorlesungscharakter im Umfang von jeweils zwei Semesterwochenstunden und werden durch eine Übung (ebenfalls im Umfang von jeweils zwei Semesterwochenstunden) begleitet. Aus derselben Gruppe von Wahlmodulen können auch Studierende des Grundschullehramts in der Masterphase eine Veranstaltung ihrer Wahl belegen.

Insgesamt steht daher ein für beide Studiengänge und im Falle des HRSGe-Lehramtes auch für Bachelor- und Masterphase gemeinsamer Pool regelmäßig angebotener Veranstaltungen zur Auswahl. Zu diesen zählen etwa typische mathematische Vertiefungsgebiete wie „Analytische Geometrie“, „Analysis“ oder „Diskrete Mathematik“. In unregelmäßigen Abständen bieten Dozierende entsprechende Veranstaltungen erstmalig an, zuletzt etwa eine Veranstaltung „Geschichte der Mathematik“.

In diesem Sinne war eine Veranstaltung, die die Bedeutung und Relevanz von Algorithmen in der Mathematik wie im Mathematikunterricht aufzeigt und gleichsam

die Fähigkeit Studierender und somit angehender Mathematik-Lehrkräfte algorithmisch zu denken fördert, am ehesten in Form eines neuen fachmathematischen Wahlmoduls des beschriebenen Pools zu konzipieren. Hierdurch kann fachmathematischen Gegenständen der Sekundarstufen und darüberhinausgehenden mathematischen Inhalten ausreichend Raum gegeben werden. Demgegenüber lassen sich fachdidaktische Aspekte der Betrachtung von Algorithmen im Unterricht aufgrund der fachmathematisch vorgegebenen Modulprägung eher am Rande behandeln.

Überdies wurden für die Konzeption einer Veranstaltung die weitergehend geschilderten didaktischen Grundannahmen getroffen. Zu diesen zählt etwa ein besonderer Grad von Eigenständigkeit und Selbstorganisation, welcher den Studierenden abverlangt werden soll und in der späten Bachelor- bzw. sogar Masterphase auch abverlangt werden darf. Dies umfasst etwa ein eigenständiges Bearbeiten der verschiedenen Vorlesungsinhalte im Rahmen eines Flipped-Classroom-Settings. Ein besonderes Augenmerk soll auch auf die Verzahnung von Lernzielen, zugehörigen Lehr-Lern-Aktivitäten und der anschließenden Leistungsüberprüfung fallen, wie sie etwa Biggs und Tang (2011) in ihrem Konzept des *Constructive Alignments* fordern. Gleichzeitig soll besagte Verzahnung ebenso wie der geplante Veranstaltungsablauf für Lernende explizit transparent gemacht werden.

3. Konzeption und Durchführung der Veranstaltung „Algorithmische Mathematik“

Im Weiteren soll die von den zuvor aufgeführten Rahmenbedingungen und Grundannahmen ausgehende Veranstaltungskonzeption vorgestellt werden. Hierbei wird auf die einzelnen Veranstaltungselemente und -inhalte aber auch Aspekte der konkreten Veranstaltungsdurchläufe eingegangen. Mit einer gesonderten Betrachtung des „Maker-Praktikums“ soll einem Veranstaltungselement zudem spezielle Aufmerksamkeit gewidmet werden. Abschließend wird außerdem auf Aspekte der finalen Leistungsüberprüfung sowie einer abschließenden Evaluation eingegangen.

3.1 Veranstaltungsstruktur

Das Vertiefungswahlmodul sieht Vorlesungen mit einer klassischen Struktur aus einer Vorlesung und einer zugehörigen Übung vor, so dass prinzipiell zwei Slots im Umfang von je 90 Minuten Lehrveranstaltungslänge zur Verfügung stehen. Traditionell folgen mathematische Vorlesungen hierbei dem Muster, dass der jeweils wochenweise neu zu thematisierende Stoff in einer großen Plenumsveranstaltung mit idealerweise allen teilnehmenden Studierenden meist in dozierendenzentrierter Form erörtert wird (s. Abb. 1). Es folgt zumeist die Ausgabe eines Übungsblattes, das dann in Einzel- oder Kleingruppenarbeit bearbeitet und eingereicht werden muss. Abschließend werden die Lösungen (ggfs. ergänzt um weitere Präsenzübungsaufgaben) in mittlerer Gruppenstärke in einer Übungsgruppe vorgestellt.

Im Unterschied zu dieser „traditionellen“ Vorgehensweise werden in der hier vorgestellten Veranstaltung zunächst Lernvideos ausgespielt, die den initialen Kontakt mit dem jeweils zu behandelnden Stoff darstellen. Es folgt ein Treffen in mittlerer Gruppenstärke innerhalb der Übungsgruppe, welche im Kontext der Veranstaltung „Lerncafé“ genannt wird, in welcher die Studierenden aufgeteilt in Kleingruppen wöchentliche Übungsaufgaben selbstständig bearbeiten. Auch hier werden die entsprechenden Bearbeitungen der Übungsblätter modulkonform später zur Korrektur eingereicht. Als letztes wochenweises Element findet im Rahmen des Vorlesungszeitslots das „Plenum“ statt, in dem alle an der Veranstaltung beteiligten Studierenden zusammenkommen, so dass finale Fragen geklärt oder weitergehende Präsenzübungsaufgaben bearbeitet werden können. Die Grundstruktur der Veranstaltung folgt somit einem Flipped- bzw. Inverted-Classroom-Design (Akçayır & Akçayır, 2018; Werner et al., 2018)

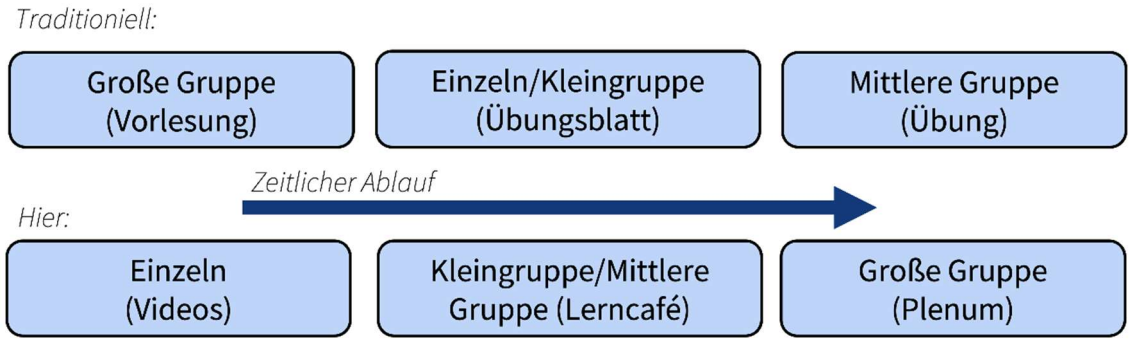

Abbildung 1: Sich wöchentlich wiederholender Veranstaltungsverlauf der vorgestellten Veranstaltung (unten) im Vergleich zum traditionellen Ablauf (oben)

3.2 Veranstaltungselemente und -inhalte

Die zuvor erläuterte grobe Struktur soll nun entlang der einzelnen Elemente der Veranstaltung detaillierter beleuchtet werden.

Bei den wöchentlich veröffentlichten Videos handelt es sich um zwei bis vier interaktive H5P-Video-Einheiten.[13] Relevante Themen, die jeweils in mehreren Videos aufbereitet werden, sind hierbei nach einer initialen Einführung etwa Turtle-Grafiken, der Euklidische Algorithmus, das Newton-Verfahren, Sortieralgorithmen, die Türme von Hanoi oder der Aufbau von Gleitkommazahlen. Eine Betrachtung der Themen erfolgt dabei im Regelfall auf drei Ebenen: einer *mathematisch-theoretischen*, einer *präimplementierenden* sowie einer *implementierenden Ebene*. Auf Ersterer wird das zugrundeliegende mathematische Konzept erarbeitet. Die präimplementierende Ebene zeichnet sich dadurch aus, dass das grundsätzliche Vorgehen in Form eines Pseudocodes programmiersprachen-unabhängig formuliert und ein entsprechender Algorithmus hierbei vorbereitet wird. Die letzte Ebene beinhaltet die konkrete Implementation eines Algorithmus in Form eines Programms. Die drei beschriebenen Ebenen werden hierbei nicht notwendigerweise

[13] Das gesamte Material ist unter https://www.klinger.nrw/lehre/algorithmische-mathematik/ zum kostenlosen Download verfügbar und steht unter einer CC-BY-NC-SA-Lizenz.

in aufbauender Reihenfolge thematisiert. Vielmehr kann etwa auch eine initiale Betrachtung der fertigen Implementationsebene aus didaktischen Gründen (z. B. im Rahmen des Black-Box/White-Box-Prinzips; Heugl, Klinger & Lechner, 1996) sinnvoll sein.

Aus mathematisch-inhaltlicher Sicht besteht die Veranstaltung somit im Wesentlichen aus Streifzügen durch vielfältige Teildisziplinen und der überblicksartigen Betrachtung einzelner mathematischer Gegenstände mit hohen algorithmischen Anteilen. Bei den Videos handelt es sich dabei um gescreencastete und vertonte Powerpoint-Präsentationen. Die Länge der einzelnen H5P-Einheiten beträgt hierbei im Regelfall nicht mehr als 15 Minuten, was einerseits einen höheren Aktivierungsgrad versprechen (Guo et al., 2014) und zudem gerade von Studierenden besser angenommen werden soll (Bordes et al., 2021). Zusätzlich folgt jede Einheit der gleichen Makrostruktur, wobei zu Beginn stets der Inhalt des Videos und bei Abschluss eine Zusammenfassung der wichtigsten Erkenntnisse stichpunktartig gelistet wird.

Im wöchentlichen Lerncafé soll regelmäßig Raum für die zu bearbeitenden Übungsblätter (im Kontext der Veranstaltung „Trainingsblätter“ genannt) gegeben werden. Im Sinne des in Abschnitt 2 erläuterten Transparenzprinzips wird hierbei auf jedem Blatt einerseits hervorgehoben, die Betrachtung welcher Videos für die nachfolgenden Aufgaben notwendig ist. Andererseits wird durch ein Sternchen an einer Aufgabe indiziert, dass zu ihrer vollständigen Bearbeitung auch ein Computerprogramm entwickelt und eingereicht werden muss. Es soll eine entspannte und ungezwungene Atmosphäre vorherrschen (worauf u. a. bereits die Bezeichnung „Café“ hindeuten soll), in welcher die Studierenden im Wesentlichen selbstständig und eigenorganisiert arbeiten. Hierbei steht der Dozent jedoch ständig als Ansprechpartner bereit.

Für die in die Veranstaltung integrierten praktischen Anteile wurde die blockbasierte Programmierumgebung „Scratch“ (Resnick et al., 2009) u. a. aus den in Abschnitt 1.2 illustrierten Gründen ausgewählt. Dies hat zur Folge, dass mit Scratch eine kostenlose und kindgerechte Plattform gewählt wurde, die die Lernenden unmittelbar in ihren späteren Unterricht integrieren können. Prinzipiell hätte auch eine professionellere bildungsorientierte Programmiersprache (etwa auf ABSIC oder Pascal basierende Konzepte) für die Veranstaltung gewählt werden können, jedoch senkt Scratch die Einstiegshürden für gänzlich unerfahrene Studierende erheblich. Scratch wird daher im Wesentlichen so genutzt, dass ein mathematischer Algorithmus als Code für eine einzelne Figur hinterlegt wird. Hierbei werden neben den grundlegenden Kontrollstrukturen wie Schleifen und Verzweigungen auch Listen, die in Scratch die wesentlichen Eigenschaften von Arrays implementieren, und „eigene Blöcke“, welche mit abgekapselten Funktionen oder Prozeduren anderer Sprachen vergleichbar sind, thematisiert. Letztere dienen hierbei vor allem dazu, rekursive Algorithmen (etwa die rekursive Variante des Euklidischen

Algorithmus oder rekursive Turtle-Grafiken) praktisch umsetzten zu können. Aspekte der objektorientierten Programmierung, wie sie Scratch etwa in Form des „Klonens“ von Figuren unterstützt, spielen hingegen keine Rolle.

3.3 Veranstaltungsdurchführung

Die Veranstaltung wurde erstmalig im Wintersemester 2020/21 durchgeführt. Bei diesem Semester handelte es sich, wie bei vielen Universitäten in Deutschland, um ein vollständiges Distanz-Semester aufgrund der damals vorherrschenden Corona-Pandemie. Aus diesem Grund konnten zwar Videos und andere Materialien ausgespielt, jedoch Lerncafés nicht wie beschrieben durchgeführt werden. Stattdessen fanden regelmäßig Besprechungen als Videokonferenz zu den Inhalten der Veranstaltung statt. Erst bei einer zweiten und dritten Durchführung im Wintersemester 2022/23 bzw. Sommersemester 2023 konnte das Konzept der beschriebenen Planung entsprechend durchgeführt werden. Hierbei kam es im Wintersemester 2022/23 aufgrund eines umfangreichen Cyberangriffs auf die Universität Duisburg-Essen zu weiteren Unregelmäßigkeiten, so dass zunächst eine zum universitären Lern-Management-System Moodle alternative Möglichkeit des Ausspielens der Videodateien gefunden werden musste. Weitere Erschwernisse ergaben sich durch einen längeren Internet- und WLAN-Ausfall in den Veranstaltungsräumen.

Erst im Sommersemester 2023 ergab sich somit eine vollständig reguläre und der Planung entsprechende Veranstaltungsdurchführung. Bei dieser wurde zusätzlich das im folgenden Abschnitt vorgestellte „Maker-Praktikum“ flankierend zum bisherigen Veranstaltungskonzept ergänzend durchgeführt.

3.4 Maker-Praktikum

Unter einem „Maker-Space“ (auch „FabLab“) ist eine Art offene Werkstatt zu verstehen, welche vielfältige analoge und digitale Werkzeuge bereitstellt, um es Privatleuten zu ermöglichen, eigene Innovationen von einer Idee ausgehend bis mindestens zu einem ersten Prototypen umzusetzen (vgl. Anderson, 2012; Akhavan, 2021). Entsprechende Werkstätten sind im Kontext der sog. *Maker-Bewegung* zu sehen (ebd.).

Das „Maker-Praktikum“ ist in diesem Sinne einem Maker-Space nachempfunden. Einzige Aufgabenstellung für die Studierenden ist es, in Kleingruppen eine Idee zu entwickeln und diese Mithilfe verschiedener bereitgestellter oder von außen hinzugezogener Werkzeuge umzusetzen. Hierbei soll einerseits ein gewisser Programmieraufwand notwendig sein, andererseits soll ein mathematischer oder mathematikdidaktischer Bezug bestehen. Dies soll dazu führen, dass im Kontext der Veranstaltung gewonnene algorithmische Kompetenzen für die Studierenden auch im größeren projektbezogenen Rahmen praktisch erfahrbar werden. Gleichzeitig erleichtert das Praktikum bereits kurz nach Veranstaltungsbeginn auch unerfahrenen Studierenden den Einstieg in generelle Ideen des Programmierens.

Zu Beginn des Praktikums geht eine Brainstorming-Phase Hand in Hand mit einer Phase des praktischen Ausprobierens und Erkundens der verschiedenen Hardware-

Optionen. Zur Auswahl stehen hierbei insbesondere „LEGO Mindstorms EV3“[14] wie auch programmierbare Einplatinencomputer wie etwa der „Calliope“[15]. Die Programmierung ist wahlweise mit Scratch oder „Microsoft MakeCode“[16] möglich. Letzteres ist Scratch hierbei durch seine blockbasierte Natur sehr ähnlich, so dass es insgesamt leichtfällt zwischen den beiden Umgebungen zu wechseln. Neben dieser bereitgestellten Hard- und Software ist es den Studierenden auch gestattet weitere Mittel hinzuzuziehen.

Der Flipped-Classroom-Charakter der Veranstaltung erlaubt es hierbei, den Übungsgruppen- oder Vorlesungsslot für die gemeinsame Umsetzung der Praktikums-Aufgabe zu verwenden. Dabei entscheiden die Studierenden eigenverantwortlich, ob gerade das aktuelle Trainingsblatt bearbeitet oder die Praktikums-Idee umgesetzt wird.

Das Praktikum schließt mit einer gemeinsamen gegenseitigen Präsentation aller Projekte (etwa 15 Minuten frei gestaltete Präsentation pro Gruppe). Die reine Teilnahme am Praktikum ist zur Klausurzulassung verpflichtend. Die einzelnen Projekte werden mit 0 bis 10 Punkten entlang der Kriterien „Kreativität / Idee“, „Mathematischer / mathematikdidaktischer Bezug“, „Umsetzung / Komplexität“, „Dokumentation“ und „Präsentation“ durch eine Fach-Jury bewertet. Besagte Kriterien wurden im Sinne eines Constructive Alignments zu Praktikumsbeginn transparent dargestellt. Die gewonnenen Punkte erhalten die Studierenden als Bonuspunkte zur abschließenden Modulklausur gutgeschrieben, sofern sie auch ohne die Punkte bestanden hätten.

3.5 Leistungsüberprüfung

Das Modul schließt mit einer Modulklausur von 90 Minuten Umfang. Eine Klausur im Sinne eines Constructive Alignments ist bei einer Veranstaltung wie der vorliegenden praktisch nur möglich, wenn das entsprechende Werkzeug (hier Scratch) auch während der Prüfung zur Verfügung steht (vgl. Küppers & Schroeder, 2017). Denkbar wäre in Analogie zu vielen Abiturprüfungen etwa ein hilfsmittelfreier sowie ein Teil mit entsprechenden Hilfsmitteln. Da dies jedoch für gewisse Restriktionen sorgt (eine Weiterarbeit am hilfsmittelfreien Teil ist z. B. nach Zulassung des Werkzeugs nicht mehr möglich) und es dem offenen und eigenverantwortlichen Charakter der Veranstaltung am ehesten entspricht, steht Scratch als Werkzeug während der Prüfung uneingeschränkt in der entsprechenden Offline-Version des Programms zur Verfügung. Hierbei handelt es sich bei einer von insgesamt fünf Aufgaben um eine explizite Programmier-Aufgabe (analog zu den Sternchen-Übungsaufgaben), sodass (von einer etwaig ausgeteilten Vorlage ausgehend) ein Scratch-Programm erstellt und zur Korrektur eingereicht werden muss. In allen anderen Aufgaben entscheiden die Studierenden selbst, ob sie

[14] Der Vertrieb wurde durch LEGO inzwischen eingestellt, siehe etwa https://www.heise.de/news/Lego-stellt-Mindstorms-ein-7322482.html.
[15] Siehe https://calliope.cc/.
[16] Siehe https://www.microsoft.com/de-de/makecode.

Scratch zur Unterstützung heranziehen oder allein Paper-and-Pencil-basiert vorgehen.

Die Prüfung wird wahlweise in einem für derartige Veranstaltung bereitstehenden Computer-Pool oder mit eigenen Geräten der Studierenden im Sinne eines BYOD-Konzepts durchgeführt. Hierbei ist dann sicherzustellen, dass während der Prüfung kein WLAN-Zugriff zur Verfügung steht bzw. erfolgt. Um Täuschungsversuchen pro aktiv zu begegnen und zudem einem unverständigen Auswendiglernen vorzubeugen, ist es den Prüflingen gestattet, ein beidseitig handschriftlich beschriebenes DIN-A4-Blatt vorzubereiten und in der Klausur zu verwenden.

3.6 Rezeption

Aufgrund der Corona-Pandemie sowie eines bereits oben beschriebenen umfangreichen IT-Ausfalls der Universität Duisburg-Essen konnte die Veranstaltung nicht mithilfe eines standardisierten Fragebogens evaluiert werden. Stattdessen wurden neben einzelnen Feedback-Gesprächen mit Studierenden Kartenabfragen vorgenommen. Hierbei hatten die Teilnehmer:innen der Veranstaltung die Gelegenheit, anonym von positiven (grüne Karten) wie negativen (rote Karten) Erfahrungen mit der Veranstaltung zu berichten wie auch konstruktive Vorschläge (gelbe Karten) zu ihrer weiteren Optimierung zu äußern.

Exemplarisch ist ein Auszug des Ergebnisses einer entsprechenden Abfrage aus dem Wintersemester 2022/23 in Abbildung 2 dargestellt. Die Kritik fiel insgesamt im beschriebenen Sinne durchweg positiv bzw. konstruktiv aus, so dass nahezu keine roten Karten abgegeben wurden.

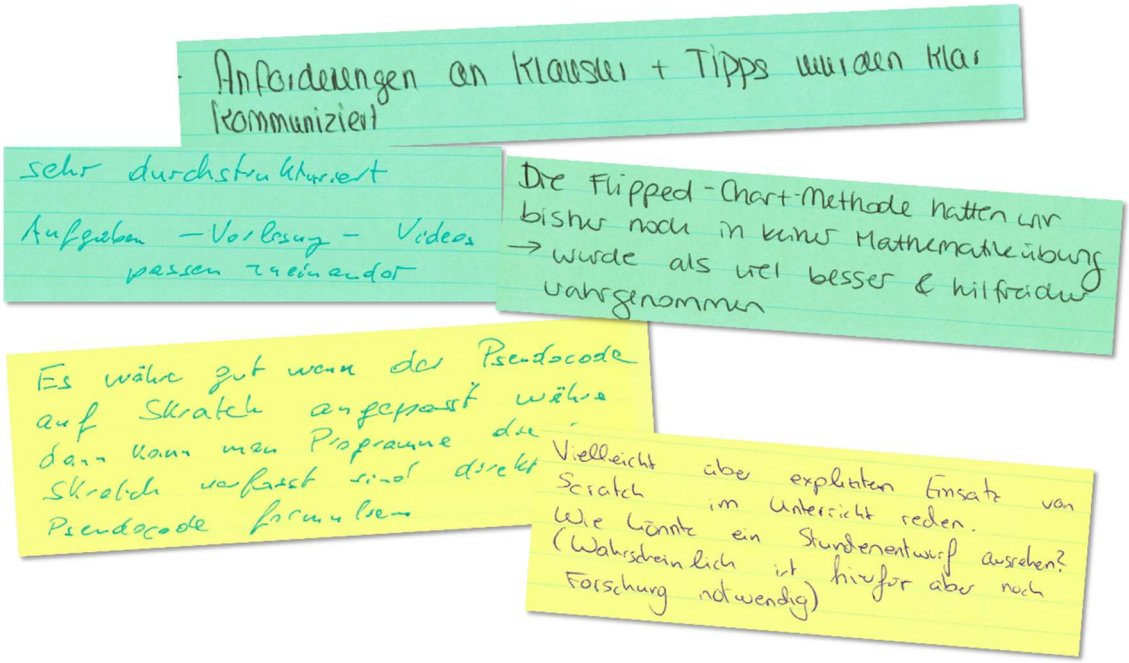

Abbildung 2: Auszug des Ergebnisses einer Kartenabfrage zu negativen, positiven und konstruktiven Rückmeldungen der teilnehmenden Studierenden im Wintersemester 2022/23

Es zeigt sich, dass die konsequente Verzahnung von Videos und Präsenzbetrieb positiv wahrgenommen wurde. Gleichzeitig wird die große Transparenz (gerade im Hinblick auf die abzuleistende Prüfung) geschätzt. Aus Sicht der Studierenden tendenziell zu verbessernde Aspekte betreffen etwa die fachmathematische und weniger unterrichtspraktische Ausrichtung der Veranstaltung, welche sich jedoch aus dem anzuwenden Wahlmodul ergibt. Auch die hier beschriebene fehlende Passung zwischen Pseudocodes und der blockbasierten Struktur von Scratch ist durchaus intendiert, um Studierenden so eine gewisse Flexibilität im algorithmischen Denken abzuverlangen.

Darüber hinaus erhält auch das Maker-Praktikum, welches im Sommersemester 2023 mit einer ähnlichen Kartenabfrage evaluiert wurde, großen Zuspruch seitens der Teilnehmenden. Hervorgehoben wird die Möglichkeit, die eigene Kreativität vollständig zu entfalten wie auch frei und individuell zu arbeiten zu können. Hierdurch habe das Programmieren deutlich an Anschaulichkeit gewonnen. Einziger Kritikpunkt ist hingegen, dass das zusätzlich zur Rahmenveranstaltung durchgeführte Praktikum die für die parallel zu erarbeitenden Vorlesungsinhalte zur Verfügung stehende Zeit zwangsläufig reduziert.

4. Fazit

Im Rahmen der mehrmaligen Veranstaltungsdurchführung hat sich das Veranstaltungskonzept, in der Form wie es auf Flipped-Classroom- und Elemente projektbasierten Lernens setzt, bewährt. Dies lässt sich gut in den unterschiedlichen Evaluationen und vor dem Hintergrund persönlicher Erfahrungen erkennen. Darüberhinausgehende empirische Erkenntnisse zur Wirksamkeit des vorgestellten Lehrveranstaltungsformats liegen nicht vor, wären aber mit Blick auf die potenziell vermittelte Fähigkeit zum algorithmischen Denken aber wünschenswert. Hierzu zählen neben einer Wirkung auf die fachlichen Kompetenzen an der Schnittstelle zwischen Mathematik und Informatik der Studierenden auch etwaig beeinflusste affektiv-motivationale Merkmale, etwa in Form von Beliefs.

Eine Fortführung und Weiterentwicklung der Lehrveranstaltung ist beabsichtigt und durch die Dokumentation und Veröffentlichung der wesentlichen Materialien in OER-Form auch dozentenunabhängig an anderen Standorten möglich. Generell wäre erstrebenswert, dass entsprechende Veranstaltungsformate durch ein Aufbrechen von zu starken Restriktionen in Form von zu restriktiven Modulbeschreibungen und unflexibler Prüfungsformate einfacher und im Sinne eines Constructive Alignments noch stimmiger realisierbar sind. Hierzu wäre etwa eine Art „Experimentierklausel“ in Prüfungsordnungen denkbar und potenziell zielführend.

Literatur:

Akçayır, G. & Akçayır, M. (2018). The flipped classroom: A review of its advantages and challenges. *Computers & Education*, 126, 334–345.

Akhavan, M. (2021). Third places for work: A multidisciplinary review of the literature on Coworking Spaces and Maker Spaces. In I. Mariotti, S. Di Vita & M. Akhavan (Hrsg.), *New workplaces – Location patterns, urbans effects and development trajectories: A worldwide investigation* (S. 13–32). Springer.

Anderson, C. (2012). *Makers: The new industrial revolution*. Crown.

Biggs, J. & Tang, C. (2011). *Teaching for quality learning at university: What the student does* (4. Aufl.). McGraw-Hill.

Bordes, S. J., Walker, D., Modica, L. J., Buckland, J. & Sobering, A. K. (2021). Impact of fully guided implant planning software training on the knowledge acquisition and satisfaction of dental undergraduate students. *Medical Education Online*, 28(1).

Eppendorf, F. & Marx, B. (2020). Blockprogrammieren im Mathematikunterricht – ein Werkstattbericht. In F. Dilling & F. Pielsticker (Hrsg.), *Mathematische Lehr-Lernprozesse im Kontext digitaler Medien: Empirische Zugänge und theoretische Perspektiven* (S. 227–245). Springer Spektrum.

Guo, P. J., Kim, J. & Rubin, R. (2014). How video production affects student engagement: An empirical study of MOOC videos. In *L@S '14: Proceedings of the first ACM conference on Learning @ scale conference* (S. 41–50). ACM.

Heugl, H., Klinger, W. & Lechner, J. (1996). *Mathematikunterricht mit Computeralgebra-Systemen: Ein didaktisches Lehrbuch mit Erfahrungen aus dem österreichischen DERIVE-Projekt*. Addison-Wesley.

Jatzlau, S. & Romeike, R. (2017). Herausforderungen durch neue Programmierkonzepte in blockbasierten Programmiersprachen. In I. Diethelm (Hrsg.), *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt: 17. GI-Fachtagung Informatik und Schule, 13.–15. September 2017, Oldenburg* (S. 383–392). Gesellschaft für Informatik e.V.

Klinger, M. (2019). Grundvorstellungen versus Concept Image? Gemeinsamkeiten und Unterschiede beider Theorien am Beispiel des Funktionsbegriffs. 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. 61–75). Springer Spektrum.

Knuth, D. E. (1997). *The art of computer programming: Fundamental algorithms* (Bd. 3). Addison-Wesley.

Kortenkamp, U. & Lambert, A. (2015). Wenn…, dann… bis…: Algorithmisches Denken (nicht nur) im Mathematikunterricht. *mathematik lehren*, Heft 188, 2–9.

Küppers, B. & Schroeder, U. (2017). Vergleich von Papierklausuren und elektronischen Prüfungen. *INFORMATIK*, 307–318. https://doi.org/10.18420/in2017_24

Oldenburg, R. (2022). Informatisches Denken im Mathematikunterricht. In G. Pinkernell, F. Reinhold, F. Schacht, & D. Walter (Hrsg.), *Digitales Lehren und Lernen von Mathematik in der Schule: Aktuelle Forschungsbefunde im Überblick* (S. 303–323). Springer Spektrum.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67.

Stephens, M. (2018). Embedding algorithmic thinking more clearly in the mathema- tics curriculum. In Y. Shimizu & R. Vithal (Hrsg.), *ICMI Study 24: School mathematics curriculum reforms: Challenges, Changes and opportunities / November 25–30, 2018, Tsukuba, Japan, University of Tsukuba* (S. 483– 490).

Stephens, M. & Kadijevich, D. M. (2020). Computational/algorithmic thinking. In S. Lerman (Hrsg.), *Encyclopedia of mathematics education* (S. 117–123). Springer.

Tall, D. & Vinner, S. (1981). Concept image and concept definition in mathematics with particular reference to limits and continuity. *Educational Studies in Mathematics*, 12(2), 151–169.

Vollrath, H.-J. (1989). Funktionales Denken. *Journal für Mathematik-Didaktik*, 10(1), 3–37.

Werner, J., Ebel, C., Spannagel, C. & Bayer, S. (Hrsg.) (2018). *Flipped Classroom – Zeit für deinen Unterricht. Praxisbeispiele, Erfahrungen und Handlungsempfehlungen*. BertelsmannStiftung.

Wing, J. M. (2010). Computational thinking: What and why? Abgerufen von https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf am 29.09.2023.

Ziegenbalg, J., Ziegenbalg, O. & Ziegenbalg, B. (2016). *Algorithmen von Hammurapi bis Gödel: Mit Beispielen aus den Computeralgebrasystemen Mathematica und Maxima*. Springer Spektrum.