Ausführlicher Unterrichtsentwurf

Für das Fach Informatik

Vorgelegt an der

Pädagogische Hochschule Weingarten

Thema der Einheit:

Entwicklung und Implementierung eines eigenen 2D-Computerspiels. Von der Spielidee über die Planung bis hin zur Umsetzung.

Thema der Stunde:

Entwicklung eines Programmablaufplans für den Spielablauf eines eigenen 2D-Spiels.

Bildungsgang:	Technisches Gymnasium
Klasse:	Eingangsklasse / 11. Klasse
Matrikelnummer:	7153537
Name:	Paola Verena Heinzelmann
Datum:	21.07.2026

1. Einordnung in die Unterrichtseinheit

In der Unterrichtseinheit wird ein eigenes 2D-Spiel entwickelt und implementiert. Dabei umfasst die Einheit alle Prozessschritte von der Spielidee über die Planung bis hin zum fertigen Spiel (Bildungsplaneinheit 11). Die in der Unterrichtseinheit angestrebten Kompetenzen werden schrittweise aufgebaut und durch passende Lernaktivitäten gefördert. Während zunächst die Analyse und Modellierung von Spielabläufen im Vordergrund steht, werden diese Kompetenzen anschließend in der Implementierung und Reflexion eines eigenen Spiels angewendet.

Stunde	**Thema**	**Lernziel der Stunde**	**Bezug Bildungsplan**	**Prozessbezogene Kompetenz**
1	Einführung in die Spielprogrammierung	Die SuS identifizieren bestehende 2D-Spiele hinsichtlich Spielziel, Spielobjekten, Benutzereingaben, Zustände und Spielregeln.	Konzeption und Entwurf eines 2D Spiels	Analysieren, Bewerten, Kommunizieren
2	Programmablaufplan erstellen	Die SuS können den Ablauf eines Spiels mithilfe eines Programmablaufplans unter Berücksichtigung von Ereignissen, Entscheidungen und Zustandsübergängen modellieren.	Planung, Entwurf, Konzeption eines Spielablaufs für ein 2D Spiel	Modellieren, Analysieren
3	Einführung/Wiederholung zentraler Programmierkonzepte/ Frameworks	Die SuS können zentrale Elemente der Entwicklungsumgebung und des verwendeten Frameworks gezielt anwenden (z.B. Schleifen, Eingaben, …).	Implementierung eines Spiels mit Entwicklungsumgebungen & Framework	Modellieren, Implementieren

4	Projektphase 1 – Projektstart: Spielkonzept & Implementierung	Die SuS können Spielkonzepte erstellen, zentrale Anforderungen festlegen und die technische Umsetzung vorbereiten.	Konzeption, Planung, Entwurf eines 2D Spiels	Modellieren, Implementieren, Strukturieren, Kommunizieren, Kooperieren
5	Projektphase 2 – Spiellogik entwickeln: Implementierung der Grundfunktionen und Spiellogik	Die SuS können grundlegende Bestandteile der Spiellogik eigenständig implementieren.	Implementierung des Spiels mit einer Hochsprache; Spielablauf; Zustandsvariable	Implementieren, Problemlösung, Kommunizieren, Kooperieren
6	Projektphase 3 – Benutzeroberfläche gestalten: Gestaltung der Oberfläche und Erweiterung der Spiellogik	Die SuS können Benutzereingaben, grafische Elemente und ein Punktesystem mit Punkte- und Ergebnisanzeige implementieren und diese zu einer funktionierenden Benutzeroberfläche integrieren.	Gestaltung der Oberfläche	Implementieren, Problemlösung, Kommunizieren, Kooperieren
7	Projektphase 4 – Spielabschluss und Punktewertung: Spielabschluss implementieren	Die SuS können das Spielende abhängig vom Spielzustand implementieren sowie Punkte und Abschlussmeldungen ausgeben.	Gestaltung der Oberfläche/Anzeige, Spielergebnisse	Begründen, Bewerten, Implementieren, Kommunizieren, Kooperieren
8	Projektphase 5 – Testen: Testen, Debugging und iterative Optimierung des Spiels	Die SuS können ihr Spiel systematisch testen, Fehler identifizieren und geeignete Korrekturen implementieren.	Eigenständige Implementierung eines 2D-Spiels	Problemlösung, Analysieren, Begründen, Bewerten, Kommunizieren, Kooperieren

9	Projektphase 6 – Dokumentation: Dokumentation und Reflexion des Entwicklungsprozesses	Die SuS können den Entwicklungsprozess und wesentliche Programmierentscheidungen dokumentieren und begründen.		Kommunizieren, Kooperieren, Analysieren, Bewerten
10	Präsentation und Bewertung von Lösungsansätzen	Die SuS können ihr Spiel darstellen, ihre Lösungswege erläutern und die Ergebnisse des Entwicklungsprozesses auswerten.		Kommunizieren, Kooperieren, Analysieren, Bewerten[1]

[1] Vgl. Land Baden-Württemberg, o.J.

Stellung der ausgearbeiteten Stunde innerhalb der Unterrichtssequenz:

Die ausgearbeitete Stunde stellt die zweite Unterrichtsstunde der Unterrichtsreihe zur Entwicklung eines 2D-Spiels dar. Nachdem die Schülerinnen und Schüler in der ersten Stunde grundlegende Merkmale eines 2D-Spiels kennengelernt haben, liegt der Schwerpunkt dieser Stunde auf der strukturierten Planung eines Programmablaufs.

Die Lernenden entwickeln ausgehend von einer eigenen Spielidee einen zentralen Spielablauf und stellen diesen in Form eines Programmablaufplans dar.

Die Stunde nimmt innerhalb der Unterrichtsreihe eine Schlüsselfunktion ein, da sie den Übergang von der Analyse eines Spiels zur strukturierten Modellierung algorithmischer Abläufe vorbereitet und bildet damit die Grundlage für das spätere Entwicklungsprojekt. Durch die grafische Darstellung des Programmablaufplans werden algorithmische Strukturen zunächst auf einer abstrakten Ebene entwickelt, strukturiert und reflektiert, bevor sie in Programmcode umgesetzt werden. Der Programmablaufplan dient dabei als Modellierungsebene zwischen der Beschreibung einer Spielidee und der späteren Umsetzung in Programmcode.[2]

Die in dieser Stunde erworbenen Kompetenzen im Modellieren algorithmischer Abläufe bilden die fachliche und methodische Grundlage für die anschließenden Unterrichtsstunden. Nach der Einführung in die Entwicklungsumgebung und das verwendete Framework entwickeln die Schülerinnen und Schüler ab der Projektphase ihr eigenes 2D-Spiel. Dabei greifen sie die in dieser Stunde erlernte Vorgehensweise zur strukturierten Planung von Spielabläufen bei der Konzeption und Implementierung ihres Spiels erneut auf.[3]

2. Bedingungsanalyse

Die Unterrichtsstunde ist für eine Eingangsklasse des Technischen Gymnasiums. Es wird angenommen, dass die Lerngruppe 24 Schülerinnen und Schüler (5 Mädchen, 19 Jungen) umfasst. Der Unterricht findet im Fach Informatik statt. Die Schülerinnen und Schüler verfügen über unterschiedliche Vorerfahrungen im Bereich der Informatik und Programmierung. Einige Lernende haben bereits erste Erfahrungen mit blockbasierten Programmiersprachen oder mit ersten Programmierkonzepten gesammelt, während andere nur über grundlegende Kenntnisse verfügen. Ein einheitlicher Kenntnisstand im Bereich der Programmierung kann daher nicht vorausgesetzt werden. Insbesondere beim algorithmischen Denken und bei der strukturierten Planung von Programmen zeigen sich daher Unterschiede hinsichtlich der Selbstständigkeit und Problemlösefähigkeit.

[2] Vgl. Hubwieser, 2001, S. 85-97
[3] Vgl. Ministerium für Kultus, 2020

Es wird angenommen, dass ein grundlegendes Interesse an praxisnahen und kreativen Aufgabenstellungen besteht. Die Entwicklung eines eigenen 2D-Spiels bietet daher einen motivierenden Kontext, da die Lernenden ihre Ergebnisse direkt anwenden und sichtbar machen können. Die Zusammenarbeit in Partner- und Gruppenarbeitsphasen wird als grundsätzlich möglich angenommen.
Der Unterricht findet in einem Informatikraum statt, der mit Computern für die Arbeit ausgestattet ist. Dadurch können die Lernenden geeignete digitale Werkzeuge zur Darstellung von Programmabläufen nutzen und ihre entwickelten Modelle übersichtlich dokumentieren.

3. Sachanalyse

Im Kontext der Entwicklung eines 2D-Spiels werden Spielideen zunächst als Algorithmen beschrieben und mit Hilfe eines Programmablaufplan (kurz: PAP) grafisch modelliert. Dabei wird die logische Struktur des Spielablaufs unabhängig von der konkreten Programmiersprache dargestellt und für die spätere Implementierung vorbereitet.[4]
Die Grundlage der Programmierung bilden Algorithmen. Sie beschreiben eine eindeutige, endliche und ausführbare Folge von Anweisungen zur Lösung eines Problems oder zur Durchführung einer bestimmten Aufgabe. Ein Algorithmus muss dabei nicht zwingend als Programmcode vorliegen, sondern kann zunächst in einer abstrakten Form beschrieben werden.[5] Bei Computerspielen können Algorithmen beispielsweise die Bewegung von Spielfiguren, die Verarbeitung von Benutzereingaben oder die Reaktion auf bestimmte Ereignisse beschreiben. Damit eine solche Spielidee programmiert werden kann, muss der Ablauf zunächst analysiert und in einzelne Handlungsschritte zerlegt werden.[6]
Zur Planung und Darstellung solcher Algorithmen werden in der Informatik verschiedene Modellierungstechniken eingesetzt. Ein PAP stellt dabei Algorithmen mithilfe standardisierter Symbole grafisch dar und ist in der DIN 66001 normiert. Mithilfe von Symbolen für Start und Ende, Verarbeitungsschritte, Ein- und Ausgaben sowie Entscheidungen lassen sich Programmabläufe übersichtlich und eindeutig darstellen. Dadurch wird die logische Struktur eines Algorithmus sichtbar.[7]
Für die Umsetzung einer Spielidee ist insbesondere die Darstellung von Steuerungsabläufen von Bedeutung, da Spiele durch verschiedene Programmstrukturen geprägt sind. Bei der

[4] Vgl. dhbw Stuttgart, o.J.
[5] Vgl. Meyer, Hilpert, & Schüller-Ruhl, 2021, S. 13
[6] Vgl. Hubwieser, 2001
[7] Vgl. Franke, o.J.

Planung einer Spielidee müssen die grundlegenden Kontrollstrukturen eines Programms berücksichtigt werden. Ein Spielablauf besteht beispielsweise aus aufeinanderfolgenden Aktionen, Entscheidungen aufgrund bestimmter Ereignisse oder Bedingungen sowie wiederkehrenden Abläufen, die durch Schleifen umgesetzt werden können. Diese Strukturen bilden die Grundlage für die spätere Umsetzung der Spiellogik in Programmcode.[8]

4. Didaktische Analyse

Bildungsplanbezug und didaktischer Begründungszusammenhang:

Die Unterrichtsstunde orientiert sich an den Vorgaben des Bildungsplans für das technische Gymnasium. Ein Bestandteil des Informatikunterrichts ist hierbei die Fähigkeit Probleme zu analysieren, geeignete Modelle zu entwickeln und diese anschließend in informatische Lösungen zu überführen. Die vorliegende Stunde greift insbesondere die prozessbezogenen Kompetenzen des Modellierens, Analysierens und Bewertens auf.[9]

Die Lernenden analysieren zunächst einen Spielablauf hinsichtlich notwendiger Aktionen, Bedingungen und Zustände. Anschließend entwickeln sie mithilfe eines Programmablaufplans ein informatisches Modell, das die logische Struktur des Spielablaufs unabhängig von einer konkreten Programmiersprache darstellt. Dabei wird ein wesentlicher Schritt informatischer Problemlösung sichtbar. Die zunächst abstrakte Spielidee wird durch Analyse und Strukturierung in ein eindeutiges, überprüfbares Modell überführt. Dadurch wird die Grundlage geschaffen, den geplanten Ablauf später in Programmcode zu überführen.

Im Kontext der Unterrichtsreihe zur Entwicklung eines 2D-Spiels unterstützt die Stunde insbesondere die Kompetenz zur Konzeption und zum Entwurf eines Spiels, indem Spielabläufe zunächst geplant und modelliert werden. Diese Planungsphase bildet eine wesentliche Grundlage für die spätere Entwicklung eigener Spiele, bei der die Lernenden ihre Konzepte mithilfe eines Anforderungskatalogs weiter konkretisieren und umsetzen.

Die eigenständige Modellierung von Spielabläufen wird bewusst vor der eigentlichen Projektphase durchgeführt, damit die Lernenden die Vorgehensweise der algorithmischen Planung zunächst unabhängig von der technischen Umsetzung kennenlernen. Die erworbenen Modellierungskompetenzen können anschließend in der Projektphase auf die Entwicklung des eigenen 2D-Spiels übertragen werden. Um die Anforderungen in der Projektphase jedoch transparent zu machen, erhalten die Lernenden einen Anforderungskatalog sowie einen darauf abgestimmten Bewertungsbogen. Dabei werden nicht nur technische Aspekte der Umsetzungen

[8] Vgl. Gesellschaft der Informatik, 2016, S. 11 ff.
[9] Vgl. Land Baden-Württemberg, o.J.

berücksichtigt, sondern auch die Planung, Strukturierung und Reflexion des Entwicklungsprozesses um die Lernenden im gesamten Prozess zu unterstützen. Dadurch werden die in dieser Unterrichtsstunde angebahnten Kompetenzen im weiteren Verlauf der Unterrichtsreihe aufgegriffen und weiterentwickelt.

Die Bedeutung des Unterrichtsinhalts dieser Stunde ergibt sich aus der grundlegenden Vorgehensweise informatischer Problemlösung. Modellierung zählt dabei zu den wichtigsten Vorgehensweisen in der Informatik, da sie es ermöglicht, „... die Komplexität der realen Welt in aussagekräftige Modelle zu übertragen. Im Unterricht müssen diese Fähigkeiten durch schrittweises Vorgehen, visuelle Hilfsmittel, spezialisierte digitale Werkzeuge und praxisnahe Projekte gezielt gefördert werden.“[10]

Die Schülerinnen und Schüler erfahren, dass Programme nicht unmittelbar durch das Schreiben von Quellcode entstehen, sondern Vorarbeiten notwendig sind. Diese Vorgehensweise ist nicht nur für die Entwicklung von Computerspielen relevant, sondern stellt eine allgemeine Arbeitsweise bei der Entwicklung von Software dar.

Didaktische Reduktion:

Die Entwicklung eines vollständigen Computerspiels beinhaltet zahlreiche komplexe Inhalte, beispielsweise objektorientierte Programmierung, Klassenstrukturen, grafische Benutzeroberflächen, sowie umfangreiche Verwaltung von Spielzuständen. Diese Aspekte werden in der vorliegenden Unterrichtsstunde bewusst nicht behandelt, da sie den Schwerpunkt der Stunde übersteigen würden.

Die didaktische Reduktion erfolgt durch die Konzentration auf die Modellierung eines Spielablaufs und dessen Darstellung mithilfe eines Programmablaufplans. Die Schülerinnen und Schüler beschäftigen sich somit zunächst mit der Struktur eines Ablaufs, bevor die konkrete Implementierung erfolgt. Der PAP dient hierbei als geeignetes Werkzeug, da er eine visuelle Darstellung algorithmischer Strukturen ermöglicht und somit einen niedrigschwelligen Zugang zur späteren Programmierung schafft. Gerade für Lernende mit unterschiedlichen Programmiererfahrungen bietet die Modellierung mithilfe eines PAPs die Möglichkeit, algorithmische Denkweisen zunächst unabhängig von den syntaktischen Anforderungen einer Programmiersprache zu entwickeln. Dadurch wird die kognitive Belastung reduziert und der Fokus auf die Entwicklung einer nachvollziehbaren Problemlösung gelegt.

[10] Hartmann , Jäckel , Näf, & Reichert, 2026, S. 16

Auch die Auswahl der Fachbegriffe wird an den Lernstand der Eingangsklasse angepasst. Statt komplexer Begriffe aus der Softwareentwicklung stehen zunächst verständliche Begriffe wie „Spielidee“, „Spielmechanik“, „Spielablauf“ und „Ablaufplanung“ im Vordergrund. Diese dienen als Zugang zu den dahinterliegenden informatischen Konzepten und ermöglichen eine schrittweise Erweiterung des Fachwissens.

Kompetenzen und Lernziel:

Kompetenz:

- Fachkompetenz:
  Die SuS können einen Spielablauf analysieren, strukturieren und mithilfe eines Programmablaufplans darstellen.
- Methodenkompetenz:
  Die SuS nutzen eine Modellierungstechnik zur strukturierten Planung algorithmischer Abläufe.
- Sozialkompetenz:
  Die SuS tauschen sich über Lösungswege aus, begründen eigene Entscheidungen und entwickeln gemeinsam Programmabläufe.
- Personalkompetenz:
  Die SuS überprüfen ihre eigenen Lösungsansätze, erkennen Verbesserungspotenziale und übernehmen Verantwortung für die Planung und Reflexion ihrer Lösungsansätze.

Lernziel:

- Übergeordnetes Lernziel:
  - Die SuS können einen Spielablauf analysieren, strukturieren und mithilfe eines Programmablaufplans modellieren.
- Teilziele:
  - Die SuS können einen Ablauf eines Spiels strukturieren und notwendige Entscheidungen identifizieren.
  - Die SuS können den geplanten Ablauf mithilfe eines Programmablaufplans modellieren.
  - Die SuS können ihre Modellierungsentscheidungen erläutern und hinsichtlich der Umsetzbarkeit überprüfen.

Antizipierte Schülerfehler und Gegenmaßnahmen:

Bei der Modellierung von Spielabläufen können verschiedene Schwierigkeiten auftreten. Eine mögliche Fehlvorstellung besteht darin, dass Schülerinnen und Schüler generell direkt mit der Umsetzung in Programmcode beginnen möchten, ohne den Ablauf zunächst zu planen und strukturieren. Diesem Problem wird entgegengewirkt, indem der Programmablaufplan als notwendiger Zwischenschritt zwischen Spielidee und späterer Implementierung eingeführt wird.

Eine weitere Herausforderung besteht darin, dass Lernende bei der Entwicklung eigener Spielideen zu umfangreichen oder technisch schwer umsetzbare Spielideen neigen, die zahlreiche Funktionen voraussetzen. Da die Entwicklung eines Computerspiels viele unterschiedliche Aspekte umfasst, kann dies dazu führen, dass der eigentliche Spielablauf nur schwer überschaubar bleibt. Dabei ist es wichtig, die Kreativität und Motivation der Lernenden aufzugreifen und ihre Ideen wertschätzend anzuerkennen, statt diese direkt einzuschränken oder abzulehnen. Um die Machbarkeit sicherzustellen, werden strukturierte Impulse und Hilfestellungen in Form von Hilfekarten gegeben, um die Lernenden dabei zu unterstützen wesentliche Abläufe herauszuarbeiten und auf eine modellierbare Struktur zu reduzieren. Weitere Ideen können bei Bedarf in einer Ausbauliste gesammelt werden und später hinsichtlich ihrer Umsetzbarkeit bewertet werden, sodass die Kreativität der Lernenden erhalten bleibt und gleichzeitig eine umsetzbare Planung entsteht.

Im Umkehrschluss könnte auch vorkommen, dass die Schülerinnen und Schüler Schwierigkeiten haben, eine eigene Spielidee oder einen geeigneten Ablauf für die Modellierung zu entwickeln. Dies kann dazu führen, dass keine konkreten Abläufe formuliert werden können. Um diese Schwierigkeit zu reduzieren, werden Anregungen in Form von Impulsfragen (Hilfekarten) oder vereinfachten Spielideen bereitgestellt. Dadurch erhalten die Lernenden eine Orientierung, ohne dass die eigenständige Entwicklung einer Spielidee eingeschränkt wird.

Zusätzlich können Schwierigkeiten beim Einsatz der Kontrollstrukturen auftreten. Lernenden berücksichtigen beispielsweise keine Wiederholungen oder Entscheidungen, sondern erstellen ausschließlich lineare Abläufe. Hier greift die Kriterienliste ein, welche die Lernenden zu Beginn der Arbeitsphase erhalten. Darüber hinaus unterstützt hier der Austausch der Lernenden über unterschiedliche Lösungswege.

Auch der Umgang mit den standardisierten Symbolen des Programmablaufplans kann zu Unsicherheiten führen. Eine Übersicht der wichtigsten PAP-Symbole sowie eine gemeinsame Besprechung eines Beispiels dienen dazu, diese Unsicherheiten zu reduzieren.

5. Methodische Analyse

Die methodische Gestaltung orientiert sich an einer kompetenzorientierten Planung des Informatikunterrichts. Im Mittelpunkt steht dabei die Frage, wie Lernende eine eigene Spielidee strukturiert planen und mithilfe eines Programmablaufplans darstellen können. Dabei steht nicht die Vermittlung einzelner Programmierschritte im Vordergrund, sondern die Planung und Modellierung eines Spielablaufs. Diese Vorgehensweise entspricht dem Charakter informatischer Problemlösung, bei der Problemstellungen analysiert, modelliert und anschließend in Lösungen überführt werden.[11]

Die Unterrichtsstunde folgt dabei einem schrittweisen Entwicklungsprozess. Ausgehend von einer eigenen Spielidee identifizieren die Schülerinnen und Schüler zunächst zentrale Spielabläufe und modellieren diesen anschließend in Form eines PAP. Der Programmablaufplan übernimmt hierbei eine vermittelnde Funktion zwischen der Beschreibung einer Spielidee und der späteren Umsetzung in Programmcode. Durch diese Modellierung wird die Komplexität der späteren Programmierung reduziert, da die Schülerinnen und Schüler zunächst die logische Struktur eines Ablaufs unabhängig von der Programmiersyntax entwickeln können. Die Trennung zwischen der Planung eines Algorithmus und der späteren Implementierung unterstützt dabei die Entwicklung eines strukturierten Vorgehens bei der Lösung informatischer Probleme.

Die Modellierung des Programmablaufs erfolgt zunächst angeleitet. Die Lehrkraft stellt die grundlegenden Symbole und Elemente vor und entwickelt gemeinsam mit den Schülerinnen und Schülern einen Beispielablauf. Diese gemeinsame Modellierung unterstützt die Lernenden bei der anschließenden eigenständigen Erstellung eines eigenen Programmablaufplans und reduziert Unsicherheiten bei der Anwendung der Modellierungsmethode.

Die zentrale Erarbeitungsphase erfolgt in Partnerarbeit. Die Sozialform ermöglicht es den Lernenden sich über unterschiedliche Lösungswege auszutauschen und gemeinsame Modell zu entwickeln. Durch die kooperative Arbeitsform, können unterschiedliche Vorkenntnisse innerhalb der Lerngruppe ausgeglichen und der Austausch über algorithmische Strukturen gefördert werden. Dies ist insbesondere in einer heterogenen Klasse gewinnbringend, da die Lernenden ihre individuellen Erfahrungen einbringen und gemeinsame Lösungsstrategien entwickeln können. Durch die gemeinsame Planung und Diskussion ihrer APA lernen die

[11] Vgl. Hartmann , Jäckel , Näf, & Reichert, 2026, S. 13 ff.

Schülerinnen und Schüler, ihre Überlegungen zu begründen und weiterzuentwickeln. Dadurch werden insbesondere die Kompetenzen Kommunizieren und Kooperieren gefördert.[12]
Während dieser Phase übernimmt die Lehrkraft überwiegend eine unterstützende und beratende Rolle und begleitet diesen Prozess durch gezielte Fragen und Impulse. Dadurch werden die Lernenden dazu angeregt, ihre eigenen Lösungswege zu reflektieren und mögliche Fehler innerhalb ihres Modells selbstständig zu erkennen.
Im Sinne der Binnendifferenzierung werden zur Unterstützung der heterogenen Lerngruppe verschiedene Differenzierungsmaßnahmen eingesetzt. Schülerinnen und Schüler, die Schwierigkeiten bei der Entwicklung oder Strukturierung ihrer Spielidee haben, erhalten Hilfekarten, die sie bei der Reduktion ihrer Spielidee auf wesentliche Elemente sowie bei der schrittweisen Erstellung eines Programmablaufplans unterstützen. Ergänzend stehen vereinfachte Vorgaben zu möglichen Spielabläufen zur Orientierung zur Verfügung. Leistungsstärkere Schülerinnen und Schüler können ihre Abläufe durch zusätzliche Bedingungen (z.B. Zeitbegrenzungen) oder komplexere Zustände erweitern. Dadurch arbeiten alle Lernenden am gleichen fachlichen Gegenstand, jedoch auf unterschiedlichen Anforderungsniveaus.
Während der Erarbeitungs- und Sicherungsphase überprüfen die Lernenden ihre Programmabläufe anhand vorgegebener Kriterien hinsichtlich Vollständigkeit und Umsetzbarkeit. Die Überprüfung der entwickelten Programmablaufpläne erfolgt anhand einer Kriterienliste, die sich an den Anforderungen einer vollständigen und nachvollziehbaren Modellierung orientiert. Dadurch werden Lernziel, methodisches Vorgehen und Bewertung miteinander verknüpft. Durch die gemeinsame Betrachtung ausgewählter Ergebnisse erhalten sie Rückmeldung zu ihren Modellierungen und können diese bei Bedarf weiterentwickeln. Dadurch wird der Modellierungsprozess als schrittweise Entwicklung und Verbesserung einer Lösung verstanden.
Ein weiterer methodischer Schwerpunkt liegt in der Steuerung des Umfangs und der Machbarkeit der entwickelten Spielidee. Da die Entwicklung eines vollständigen Computerspiels eine hohe Komplexität besitzt, werden die Schülerinnen und Schüler dabei unterstützt, ihre Spielidee auf einen zentralen und modellierbaren Spielablauf zu reduzieren. Durch Kriterien und Hilfestellungen (z.B. beispielhafte Spielideen) wird sichergestellt, dass die entwickelten Spielabläufe innerhalb des zeitlichen Rahmens modellierbar bleiben. Dabei wird der kreative Gestaltungsspielraum der Lernenden bewusst erhalten. Die Hilfestellungen dienen nicht dazu

[12] Vgl. Gesellschaft der Informatik, 2016, S. 5-7

die Spielidee einzuschränken, sondern unterstützten die Lernenden dabei Ideen zu strukturieren und auf wesentliche Abläufe zu fokussieren.
Die Unterrichtsstunde ist handlungsorientiert gestaltet, da die Schülerinnen und Schüler nicht nur theoretische Modelle betrachten, sondern selbst eine Spielidee entwickeln, planen und als Grundlage für eine spätere praktische Umsetzung vorbereiten. Dadurch entsteht ein unmittelbarer Bezug zwischen Konzept und einem konkreten Anwendungsproblem.
Die Sicherung der Ergebnisse erfolgt durch die Vorstellung ausgewählter Programmablaufpläne im Plenum. Die Schülerinnen und Schüler erläutern ihre Entscheidungen und vergleichen unterschiedliche Lösungswege hinsichtlich ihrer Verständlichkeit und Umsetzbarkeit. Dadurch werden die entwickelten Modelle bewertet und mögliche Verbesserungen gemeinsam reflektiert.
Durch die Verbindung aus problemorientiertem Einstieg, kooperativer Erarbeitung, individueller Modellierung und Reflexion werden die Schülerinnen und Schüler gezielt auf die anschließende Implementierung ihres Spiels vorbereitet.

6. Tabellarischer Verlaufsplan

Übergeordnetes Ziel:

Zeit	Phasen	Lehrer-Aktivität	Schüler*innen-Aktivität	Sozialform	Medien
5‘	Einstieg: Problemorientierung und Aktivierung des Vorwissens	Zeigt eine kurze Spielsituation bzw. Spielidee und stellt die Problemfrage: „Wie kann der Ablauf eines Spiels geplant und dargestellt werden, bevor es programmiert wird?“	Beschreiben notwendige Aktionen, Bedingungen und Wiederholungen der Spielidee und aktivieren ihr Vorwissen aus den vorherigen Stunden.	Plenum	Präsentation/Tafel
10‘	Erarbeitung I: Einführung Programmablaufplan	Stellt den PAP als Modellierungsmethode vor und erläutert die grundlegenden Symbole. Anschließend entwickelt sie gemeinsam mit den SuS einen einfachen Beispielablauf.	Ordnen einzelne Handlungsschritte den PAP-Symbolen zu und erkennen die Bedeutung von Entscheidungen und Wiederholungen innerhalb eines Algorithmus.	Unterrichtsgespräch	Präsentation, Arbeitsblatt PAP-Symbole, draw.io, Computer
20‘	Erarbeitung II: Erstellung eines eigenen PAPs	Begleitet die Partnerarbeit beratend und unterstützt die SuS durch Impulse. Bei zu umfangreichen Spielideen unterstützt sie die Eingrenzung auf einen zentralen und realisierbaren Spielablauf. SuS mit	Erstellen in Partnerarbeit einen Programmablauf für eine ausgewählte Spielidee. Sie diskutieren Lösungswege und begründen algorithmische Entscheidungen.	Partnerarbeit	PAP-Vorlage, Hilfekarten, Kriterienliste, Computer

		Unterstützungsbedarf erhalten Hilfekarten oder Beispiel-PAPs.			
8‘	Transfer & Sicherung: Überprüfung der Modelle	Lässt ausgewählte Ergebnisse vorstellen und stellt Bezüge zur späteren Umsetzung in Java her.	Erläutern ihre Entscheidungen, vergleichen unterschiedliche Lösungswege und erkennen den PAP als Grundlage für spätere Programmierung.	Plenum	Schülerergebnisse, Computer
2‘	Abschluss und Ausblick	Fasst die Bedeutung der Ablaufplanung zusammen und gibt einen Ausblick auf die Übertragung in Java.	Reflektieren die Funktion des PAP für die weitere Entwicklung des Spiels.	Plenum	Tafel/Präsentation, Computer

7. Reflexion der Umsetzungsphase des Projekts

Die praktische Umsetzung eines eigenen 2D-Spiels stellt für die Lernenden eine motivierende, aber gleichzeitig komplexe Aufgabe dar. Eine zentrale Herausforderung besteht darin, den Umfang der entwickelten Spielidee so zu steuern, dass innerhalb des verfügbaren Zeitrahmens ein funktionsfähiges Ergebnis entstehen kann. Gerade bei offenen und kreativen Aufgabenstellungen besteht die Gefahr, dass Lernende zunächst sehr umfangreiche Vorstellungen entwickeln, die technisch oder zeitlich nur schwer umsetzbar sind. Die Aufgabe der Lehrkraft besteht daher darin, die Kreativität der Schülerinnen und Schüler zu erhalten und gleichzeitig die Machbarkeit der Projekte sicherzustellen. Dabei werden Spielideen nicht grundsätzlich eingeschränkt, sondern gemeinsam mit den Lernenden auf einen realisierbaren Kern reduziert (Scope-Steuerung). Ein wichtiges Instrument zur Steuerung des Projektumfangs ist dabei das Scope-Gespräch. Dieses beginnt mit der Wertschätzung der entwickelten Spielidee. Die Lehrkraft nimmt die Kreativität und Motivation der Lernenden ernst und gibt zunächst eine positive Rückmeldung, indem sie beispielsweise hervorhebt, welche Überlegungen bereits in die Idee eingeflossen sind. Eine direkte Ablehnung oder Abwertung umfangreicher Ideen wird vermieden, da dadurch die Motivation und Eigenverantwortung der Lernenden beeinträchtigt werden könnte. Stattdessen wird die Idee gemeinsam weiterentwickelt und hinsichtlich ihrer Umsetzbarkeit betrachtet.[13]

Zur Steuerung des Projektumfangs wird zunächst der kleinste spielbare Kern des Spiels betrachtet (Minimum Viable Product, MVP). Die Schülerinnen und Schüler identifizieren die wesentlichen Spielmechaniken und konzentrieren sich auf die Funktionen, die für ein grundlegendes funktionierendes Spiel notwendig sind.[14] Erweiterungen und zusätzliche Ideen können in einer Ausbauliste gesammelt werden (Backlog)[15] und bei ausreichender Zeit ergänzt werden. Dadurch bleibt die ursprüngliche Spielidee erhalten, während gleichzeitig verhindert wird, dass sich die Lernenden in zu komplexen Anforderungen verlieren.

Eine wichtige Unterstützung bei diesem Prozess stellen Skizzen und Prototypen dar (Game Design Documentation, Wireframe/Mockup und Prototypen). Durch eine erste grafische Darstellung des Spiels können Lernende ihre Vorstellungen konkretisieren und zentrale Elemente wie Spielfiguren, Spielobjekte, Benutzeroberfläche und Spielablauf sichtbar machen. Diese Vorplanung ermöglicht es, mögliche Schwierigkeiten frühzeitig zu erkennen und Änderungen

[13] Vgl. Franke, o.J.
[14] Vgl. t2 Informatik - Minimum Viable Product, o.J.; Franke, o.J.
[15] Vgl. Kleczewski, o.J.

vorzunehmen, bevor umfangreiche Programmierarbeit entsteht. Gleichzeitig unterstützten Skizzen und Prototypen die Kommunikation innerhalb der Gruppe, da gemeinsame Vorstellungen über Gestaltung und Funktion des Spiels entwickelt werden können. Durch die gemeinsame Betrachtung und Überarbeitung des Mockups/Prototyps reflektieren die Lernenden ihre gestalterischen Entscheidungen und überprüfen, ob die geplanten Elemente zur Spielidee und den Anforderungen des Spiels passen. Dadurch können notwendige Anpassungen frühzeitig erkannt und umgesetzt werden. Hier sollte allerdings berücksichtigt werden, dass im Informatikunterricht ein Projekt häufig nicht mit einem produktiven System endet, sondern mit einem Prototyp.[16]

Der PAP übernimmt in der Umsetzungsphase eine zentrale Vermittlungsfunktion zwischen Spielidee und Programmcode (Modellierungsebene zwischen Konzept und Implementierung). Nachdem die grundlegenden Spielmechaniken festgelegt wurden, wird der Ablauf des Spiels strukturiert modelliert. Der PAP ermöglicht es den Schülerinnen und Schülern, Entscheidungen, Wiederholungen und Zustandsänderungen zunächst unabhängig von einer konkreten Programmiersprache zu planen. Dadurch wird die logische Struktur des Spiels überprüft, bevor die eigentliche Implementierung beginnt. Während der Programmierphase dient der PAP als Orientierung und kann bei Bedarf weiterentwickelt und angepasst werden.

Eine weitere Unterstützung bei der Entwicklung stellt eine Referenz-Implementierung dar. Diese dient nicht der Übernahme fertiger Programmteile, sondern unterstützt die Analyse bestehender Lösungen. Die Lernenden untersuchen dabei, wie bestimmte Spielmechaniken technisch umgesetzt werden können und übertragen diese Erkenntnisse auf die Entwicklung eigener Lösungen. Dabei kann reflektiert werden, welche Bestandteile der Referenz-Implementierung für das eigene Projekt geeignet sind, welche Anpassungen notwendig werden und welche alternativen Umsetzungen möglich sind. Dadurch wird der Übergang von der Modellierung zur Implementierung unterstützt und gleichzeitig die Fähigkeit gefördert, bestehenden Lösungen kritisch zu bewerten und weiterzuentwickeln.[17]

Die Rolle der Lehrkraft verändert sich dabei während der Umsetzungsphase von einer erklärenden zu einer begleitenden und beratenden Funktion (Lernbegleitung und Coaching). Die Lehrkraft gibt keine vollständigen Lösungen vor, sondern unterstützt die Lernenden gezielt durch Fragen, Feedback und methodische Hilfestellungen. Bei Problemen hilft sie dabei, Ursachen zu analysieren und geeignete Lösungsstrategien zu entwickeln. Gleichzeitig begleitet

[16] Vgl. Hartmann , Jäckel , Näf, & Reichert, 2026, S. 112
[17] Vgl. Franke, o.J.

sie die Gruppen bei der Priorisierung von Anforderungen und achtet darauf, dass die Projekte innerhalb eines realistischen Umfangs bleiben.
Durch die Kombination aus Planung, Modellierung, Prototyp und iterativer Implementierung erfahren die Schülerinnen und Schüler Softwareentwicklung als einen strukturierten Entwicklungsprozess (Iterativer Entwicklungsprozess). Sie lernen, dass erfolgreiche Lösungen in der Informatik nicht unmittelbar durch Programmcode entstehen, sondern durch wiederholtes Planen, Umsetzen, Überprüfen und Weiterentwickeln.[18]

[18] Vgl. Franke, o.J.

8. Anhang

Anhang 1: Arbeitsblatt „Programmablaufplan“

Programmablaufplan *(PAP)*

Programmablaufplan in der Programmierung

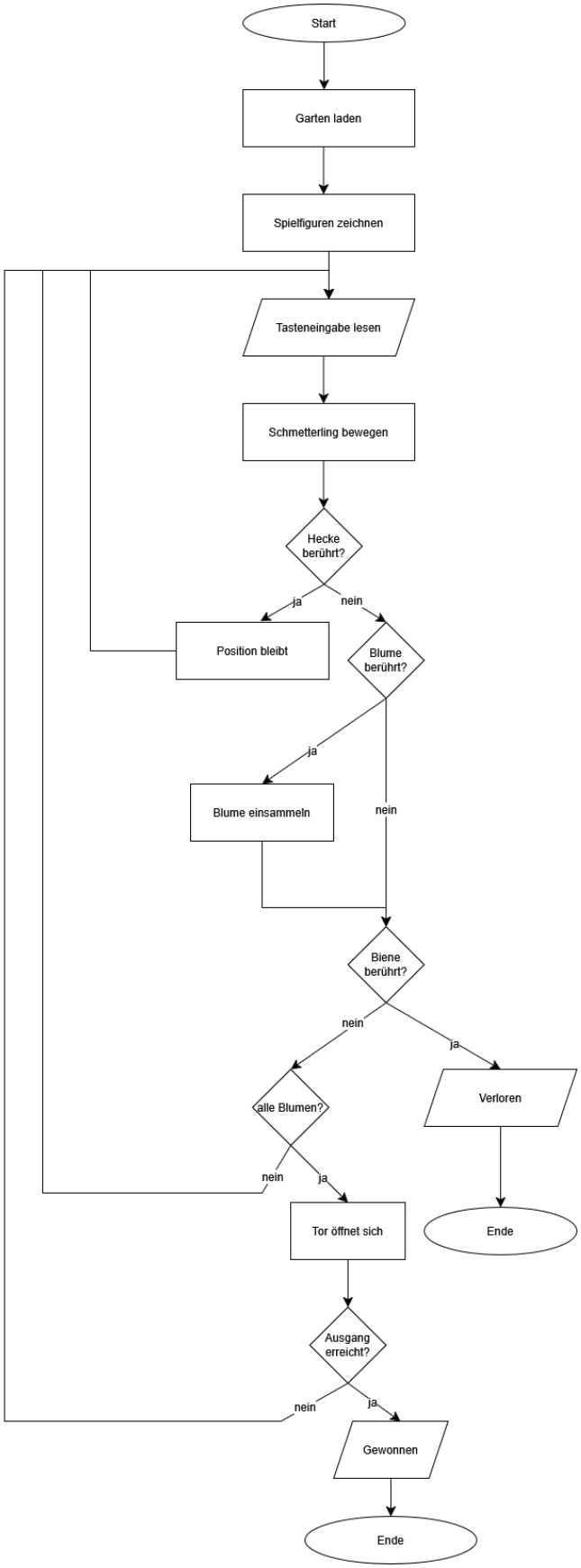
Bevor ein Spiel programmiert wird, muss der Ablauf geplant werden. Ein Programmablaufplan (PAP) stellt einen Algorithmus grafisch dar. Er zeigt, welche Schritte nacheinander ausgeführt werden und wie das Programm auf Ereignisse reagiert.

Die wichtigsten PAP-Symbole und ihre Bedeutung:

Symbol	Bedeutung	Beispiel aus einem 2D-Spiel
	Start/Ende	Spiel starten, Spiel beendet
	Ein-/Ausgabe	Taste drücken, Punktestand anzeigen
	Verarbeitung	Spieler bewegen, Punkte erhöhen
	Verzweigung	Ist der Spieler mit einem Gegner zusammengestoßen?
	Flusslinie	Reihenfolge der Schritte

Beispiel PAP aus der Unterrichtsstunde:

Anhang 2: Vorlage Programmablaufplan (PAP)



Anhang 3: Kriterienliste zur Überprüfung des Spielablaufs

Kriterienliste – Ist unser Programmablaufplan vollständig?

Überprüft euren PAP gemeinsam und setzt ein Häkchen.

Kriterium	**Ja**	**Teil-weise**	**Nein**
Unser PAP besitzt einen Start und ein Ende.	☐	☐	☐
Die einzelnen Schritte sind in einer logischen Reihenfolge dargestellt.	☐	☐	☐
Alle wichtigen Aktionen des Spiels sind enthalten.	☐	☐	☐
Mindestens eine notwendige Entscheidung wird berücksichtigt.	☐	☐	☐
Wiederholungen bzw. Schleifen wurden eingezeichnet (mindestens eine), wenn sie benötigt werden.	☐	☐	☐
Die verwendeten PAP-Symbole sind richtig eingesetzt.	☐	☐	☐
Der Ablauf ist übersichtlich dargestellt und gut lesbar.	☐	☐	☐
Eine andere Gruppe könnte unseren Ablauf nachvollziehen.	☐	☐	☐

Reflexionsfragen:

1. Was funktioniert an unserm Programmablaufplan bereits gut?

   _____

   _____

   _____

2. Was können wir noch verbessern?

   _____

   _____

   _____

3. Fehlt noch eine Entscheidung oder eine Wiederholung?

   _____

   _____

   _____

4. Ist unsere Spielidee realistisch umsetzbar/wo könnten Schwierigkeiten in der technischen Umsetzung auftreten?

   _____

   _____

   _____

Anhang 4: Hilfekarten

Zur Entwicklung und Modellierung eines PAP

Herausforderung:

Meine Spielidee ist zu groß und ich weiß nicht wo ich anfangen soll:

1. **Spielziel:**
   - Was muss das Spiel erreichen? – z.B. Alle Gegenstände sammeln)
   - Wann hat der Spieler gewonnen? – z.B. eine Bestimmte Punktzahl erreicht
   - Wann ist das Spiel vorbei? – z.B. Ein Ziel erreicht/Bestimmte Zeit überschritten
2. **Spieleraktion:**
   - Was kann die Spielfigur tun? – z.B. laufen, springen, schießen, sammeln
3. **Spielobjekt:**
   - Welche Objekte gibt es? – z.B. Spielfigur, Gegner, Hindernisse

MERKE: Ein gutes Spiel braucht nicht viele Funktionen. Plane zuerst eine einfache Version, die funktionieren kann.

Herausforderung:

Ich weiß nicht, welche Schritte in meinen PAP gehören. Gehe Schritt für Schritt vor:

1. **Start**
   - Was passiert am Anfang des Spiels? – z.B. Spielfeld laden, Spiel positionieren
2. **Aktionen**
   - Was macht der Spieler? – z.B. Spieler bewegt sich
   - Was passiert automatisch? – z.B. Punkte werden gesammelt, Gegner bewegt sich
3. **Entscheidungen**
   - Gibt es Situationen, auf die das Spiel reagieren muss?

     Signalwörter: Wenn…dann
4. **Wiederholungen**
   - Was passiert immer wieder? – z.B. Spielfigur bewegen, Eingabe überprüfen

Herausforderung:

Entscheidungen im Spiel erkennen (Erweiterung)

Viele Spiele bestehen aus Entscheidungen:

Formuliere Bedingungen:

Wenn… passiert, dann:

Beispiel:

Ereignis	**Reaktion**
Spieler berührt Gegner	Leben verlieren
Spieler sammelt Objekte	Punkte erhöhen

Herausforderung:

Dein Spiel funktioniert bereits? Ergänze weitere Zustände

Mögliche Erweiterungen:

- Mehrer Level
- Verschiedenen Schwierigkeitsstufen
- Zusätzliche Gegner
- Zeitbegrenzung
- Verschiedene Spielzustände

ACHTUNG: Mehr Funktionen bedeuten:

- Mehr Entscheidungen
- Mehr Zustände
- Mehr Programmieraufwand

MERKE: Plane nur Erweiterungen, die realistisch umsetzbar sind.

Unterstütz zur Entwicklung einer Spielidee

Beispiel: Hindernislauf

Spielidee:

Eine Spielfigur bewegt sich durch eine Spielwelt und muss Hindernisse ausweichen.

Mögliche Abläufe:

- Spielfigur bewegt sich nach links und rechts.
- Bei einer Berührung mit einem Hindernis wird ein Ereignis ausgelöst.
- Das Spiel endet nach einer bestimmten Anzahl an Treffern.

Mögliche Zustände:

- Position der Spielfigur
- Anzahl der Leben
- Punktestand

Beispiel: Sammelspiel

Spielidee:

Eine Spielfigur sammelt Gegenstände ein und erhält dafür Punkte.

Mögliche Abläufe:

- Spielfigur bewegt sich durch die Spielwelt.
- Bei Kontakt mit einem Gegenstand wird dieser entfernt.
- Der Punktestand wird erhöht.
- Wenn alle Gegenstände gesammelt werden, endet das Spiel.

Mögliche Zustände:

- Position der Spielfigur
- Anzahl gesammelter Gegenstände
- Punktestand
- Punktestand

Beispiel: Hindernislauf

Spielidee:

Die Spielerin oder der Spieler muss auf bestimmte Ereignisse schnell reagieren.

Mögliche Abläufe:

- Objekt erschein zufällig.
- Die Spielfigur oder der Spieler reagiert darauf.
- Bei einer richtigen Reaktion werden Punkte vergeben.
- Nach einer bestimmten Zeit endet das Spiel.

Mögliche Zustände:

- Zeit
- Punktestand
- Aktuelles Ereignis

Anhang 5: Hilfekarten für das Projekt

Herausforderung:

Das Spiel startet nicht – Beim Start ist etwas falsch: Figuren fehlen oder das Spiel beginnt nicht im Startbildschirm

- Fragen:
  - Welche Figur muss beim Starten sichtbar sein?
  - Welche Position soll die Figur am Anhang haben?
  - Muss eine Variable zurückgesetzt werden?

Tipp: Wenn Grüne Flagge angeklickt wird, wird eine Startbedingung benötigt

Herausforderung:

Meine Figur bewegt sich nicht – Die Figur reagiert nicht auf die Pfeiltasten

- Fragen:
  - Gibt es einen Block: „Wenn grüne Flagge angeklickt wird?“
  - Liegt die Bewegung in einer „wiederhole fortlaufend“ Schleife?
  - Sind die richtigen Tasten ausgewählt?

Tipp: Ohne Schleife wird die Taste nur ein Mal geprüft

Herausforderung:

Meine gesammelten Objekte verschwinden nicht –

Die Objekte können mit der Figur gesammelt werden, aber sie bleibt sichtbar.

- Fragen:
  - Gibt es eine Berührungsabfrage?
  - Wird das Objekt danach versteckt?
  - Wird der Zähler erhöht?

Tipp: Überlege genau, was passieren soll, wenn der Spieler die Blume berührt.

Herausforderung:

Das Ziel öffnet sich nicht – Alle Objekte wurden gesammelt, aber das Ziel öffnet sich nicht.

- Fragen:
  - Ist die Variable wirklich auf 3?
  - Prüft das Tor die richtige Variable?
  - Gibt es den Block: „falls Blumen = 3“

Tipp: Zeige dir die Variable auf der Oberfläche an und beobachte den Wert.

Herausforderung:

Verloren/Gewonnen funktioniert nicht – Das Spiel endet nicht richtig Fragen:

- Fragen:
  - Wird die Bedingung wirklich geprüft?
  - Steht „stoppe alle“ innerhalb der Bedingung?
  - Wird die Nachricht vor dem Stoppen angezeigt

Herausforderung:

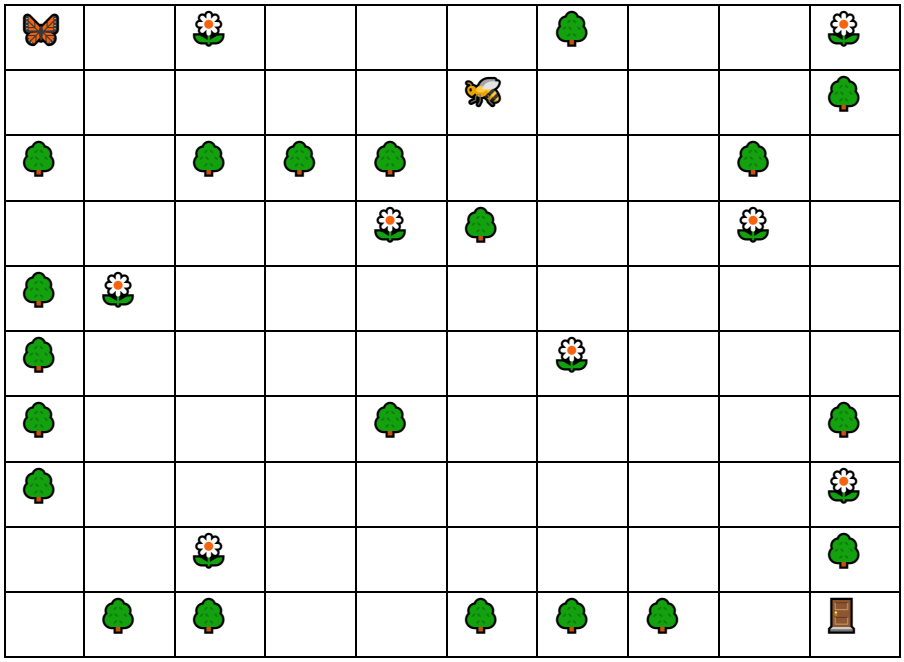
Meine Figur geht durch Hindernisse – Die Figur fliegt durch die Hindernisse

Tipp: Nach einer Bewegung muss geprüft werden: „Bin ich jetzt in einem Hindernis?“

Anhang 6: Anforderungskatalog für die Spielentwicklung

- Genutzt wird Java – geeignet für den Fokus Projektarbeit, Spielentwicklung und Problemlösung – Fokus nicht auf komplexer Syntax
- 2D-Spiel mit dem Spielfigur Objekte sammeln und Hindernisse ausweichen – muss – dafür gibt es Punkte
    - Optional: es läuft ein Counter mit, in dem so viel Punkte wie möglich erreicht werden sollen
- Anforderungen
    - Steuerung – Spielfigur muss per Tastatur steuerbar sein
    - Variable – Mindestens 2 Variable müssen verwendet werden (z.B. Punkte, Leben, Zeit)
    - Schleifen - das Spiel muss 2 Schleifen beinhalten
    - Bedingungen – mind. 3 Bedingungen
    - Figuren - mindestens 3 unterschiedliche Figuren (Hindernisse, Sammelobjekte, …)
    - Nachrichten – mindestens 2 Nachrichten müssen ausgegeben werden
    - Funktionalität – Spiel muss spielbar sein
    - Benutzeroberfläche – Startbild und Endbildschirm einbauen
    - Spielende – es gibt ein Szenario zum Gewinnen und Verlieren
    - Benutzerfreundlich – verständliche Steuerung & klare Bedienung
    - Gestaltung – Verwendung passender Grafik, Farben, Hintergründe
    - Sinnvoller Spielablauf
    - Erweiterungen möglich – zusätzliche Funktionen (z.B Timer, Sound, mehrere Level)
- Dokumentation
    - Titel des Spiels
    - Beschreibung Spielidee
    - Erklärung Spielregeln
    - Zustandsdiagramm
        - Startzustand
        - Spiel läuft
        - Pause
        - Gewinn/Verlust
    - Aufgabenverteilung
    - Reflexion

Anhang 7: Mockup für das Spieldesign



Anhang 8: Bewertungsbogen für das Spielprojekt

Anforderung	**Punkte**
Technisch	**50**
Spielfigur muss per Tastatur steuerbar	4
2 Variable sinnvoll verwendet	2
2 Schleifen korrekt verwendet	4
3 Bedingungen korrekt verwendet	6
2 verschiedene Figuren eingesetzt	3
2 Nachrichten müssen ausgegeben werden – z.B. „Du hast verloren“	2
Spiel ist spielbar und funktionsfähig	10
Szenario für Gewinner und Verlierer berücksichtigt	6
Benutzerfreundliche Steuerung/Bedienung	2
Sinnvoller & nachvollziehbarer Spielablauf	6
Saubere Programmierstruktur – Übersichtlich, sinnvolle Aufteilung, kein überflüssiger Code, nachvollziehbar	3
Verwendung von sinnvollen Kommentaren	2
Kreativ	15
Eigenständige und kreative Spielidee	5
Passende Grafiken, Figuren, Hintergründe	4
Farbgestaltung unterstützt Spielverständnis (Grün – Gewinn)	2
Zusätzliche Funktionen (z.B Timer, Sound, Level) • Keine Erweiterung • Erweiterung nur teilweise Umgesetzt • Eine sinnvolle Erweiterung • Mehrere sinnvolle Erweiterungen	 0 1 2 4
Präsentation	15
Verständliche Vorstellung des Spiels (Aufbau der Präsentation)	3
Spielidee und Regeln erläutert	2
Technische Umsetzung erläutert	2
Demo des Spiels	2

Fragen beantworten	1
Auftreten (deutliche Sprache, Blickkontakt, angemessen Lautstärke)	5
Dokumentation	20
Titel des Spiels vorhanden	1
Spielidee verständlich beschrieben	3
Spielregeln erläutert	3
Zustandsdiagramm vorhanden • Startzustand • Spiel läuft • Pause • Gewinn • Verlust	5
Aufgabenverteilung dokumentiert	2
Reflexion • Gruppenarbeit • Technische Herausforderungen	5
Übersichtlichkeit der Dokumentation	2
Gesamtpunktzahl	100

9. Literaturverzeichnis

dhbw Stuttgart. (kein Datum). *Flussdiagramm/Programmablaufplan (PAP)*. Von https://wwwlehre.dhbw-stuttgart.de/~fritzsch/Programmieren/Material/PAP%20Struktogramm.pdf abgerufen

Franke, S. (kein Datum). *Praktische Umsetzung - ein Programmierprojekt mit Schüler:innen steuern*. Von https://franke-lab.de/lehre/sose/fadiinf/u9.html abgerufen

Gesellschaft der Informatik. (29. 01 2016). Abgerufen am 19. 07 2026 von https://dl.gi.de/server/api/core/bitstreams/b9724584-0fad-475d-8f07-b8742a37a248/content

Hartmann , W., Jäckel , S., Näf, M., & Reichert, R. (2026). *Informatikunterricht planen und durchführen.* Springer. Abgerufen am 07. 20 2026

Hubwieser, P. (2001). *Didaktik der Informatik Grunlagen, Konzepte, Beispiele.* Heidelber: Springer-Verlag.

Kleczewski, C. (kein Datum). *Agile Scrum Group*. Von Product Backlog: https://scrumguide.de/product-backlog/ abgerufen

Land Baden-Württemberg. (kein Datum). *Baden-Württemberg Bildungsplände* . Von Informatik TG: https://www.bildungsplaene-bw.de/In_OS abgerufen

Meyer, D., Hilpert, W., & Schüller-Ruhl, T. (12 2021). *Algorithmus*. (B. f. /bpb, Herausgeber) Von https://www.bpb.de/system/files/dokument_pdf/einfach_POLITIK_Lexikon_barrierefrei_2021_12.pdf abgerufen

Ministerium für Kultus, J. u.-W. (Hrsg.). (2020). *Bildungsplan des Beruflichen Gymnasiums*. Von https://www.bildungsplaene-bw.de/In_OS abgerufen

t2 Informatik - Minimum Viable Product. (kein Datum). Von https://t2informatik.de/wissen-kompakt/minimum-viable-product/ abgerufen

TüftelLab. (kein Datum). Von https://digital.tueftellab.de/mod/page/view.php?id=143 abgerufen