Erklärung zur Bachelorarbeit

Ich versichere, dass ich die Arbeit selbständig, ohne fremde Hilfe verfasst habe. Bei der Abfassung der Arbeit sind nur die angegebenen Quellen benutzt worden. Wörtlich oder dem Sinne nach entnommene Stellen sind als solche gekennzeichnet. Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, insbesondere, dass die Arbeit Dritten zur Einsichtnahme vorgelegt oder Kopien der Arbeit zur Weitergabe an Dritte angefertigt werden.

_____ _____

Ort, Datum Unterschrift

Zusammenfassung

Im Rahmen dieser Bachelorarbeit wurde eine Webanwendung unter den Namen „Go!Farmer“ entwickelt, um der Zielgruppe (Kinder zwischen sechs bis zehn Jahre) eine spielerische Einführung in das „Computational Thinkings“ zu ermöglichen.

Den Ausgangspunkt der Entwicklung bildete eine Analyse der Zielgruppe und einer bereits existierenden ähnlichen Webanwendung (Lightbot). Die Analyse hat ergeben, dass die bereits existierende Webanwendung zwar einen spielerischen Zugang ermöglicht, indem ein Roboter durch eine Sequenz von Anweisungen Schritt für Schritt an ein Ziel geführt werden kann. Allerdings wird nur begrenzt darauf eingegangen, welche weiterführende Konzepte Programmiersprachen haben.

Die Entwicklung des Konzepts begann mit einer Analyse der Anforderungen, welche sich aus der Anlehnung an das Konzept von Lightbot, den möglichen Erweiterungen und der Analyse von Computational Thinking ergeben haben. Anschließend wurde die Webanwendung mithilfe von HTML, CSS und Javascript umgesetzt.

Der Prototyp, welcher sich aus dem Konzept ergab, wurde mithilfe von einem Usability-Test und einer Evaluation in einer Grundschule getestet und bewertet, um dementsprechend die erforderlichen Anpassungen vorzunehmen, damit das Endresultat optimal ist.

Die Implementierung der Webanwendung beweist, dass das in dieser Bachelorarbeit aufgestellt Konzept funktioniert.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während meiner Bachelorarbeit und meines Studiums unterstützt haben.

Mein Dank gilt meiner Familie für die Unterstützung während meines Studiums.

Ein besonderer Dank geht an meinen Erstprüfer Prof. Dr. rer. nat. Dipl.-Inform. Nane Kratzke und meiner Zweitprüferin Prof. Dr. rer. nat. Monique Janneck für die zahlreichen Tipps und das Lesen meiner Arbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
DOM	Document Object Model
JS	JavaScript
JSON	JavaScript Object Notation
PNG	Portable Network Graphics
PF	Produktfunktion
NFR	Nicht funktionale Requirements
MVC	Model-View-Controller
CT	Computational Thinking

Kapitel 1 – Einleitung

Heutzutage spielt die informatische Bildung eine zentrale Rolle, aber dennoch sind nicht alle davon begeistert. Der Grund dafür ist, dass den meisten Leuten gar nicht klar ist, was dieser Begriff überhaupt bedeutet. Einige denken, dass informatische Bildung gleichbedeutend sei mit Tastaturschreiben oder das Anwenden von Textverarbeitungsprogrammen. Die anderen denken bei der informatischen Bildung an eine stark programmierfokussierte Ausbildung, welche die nächste Generation von „Super-Programmieren“ und „Mega-Firmengründern“ wie Bill Gates (Microsoft) oder Mark Zuckerberg (Facebook) hervorbringen. Der Grund für diese Sichtweisen liegt in der schulischen Bildung. Da es keine frühzeitige Einführung in das Themengebiet der „informatischen Bildung“ gibt, ist es den meisten Leuten im späteren Lebensverlauf auch gar nicht klar, was dieser Begriff bedeutet. Hinzu kommt, dass obwohl sehr viele Kinder ein großes Interesse an Computertechnologien haben und Computerspiele spielen oder soziale Netzwerke nutzen, sich nur sehr wenige für das Programmieren interessieren. Grund dafür ist, dass die meisten Schüler und Schülerinnen das Programmieren als „schwierig und langweilig“ empfinden. Bedeutet dies also, dass Schulen Informatik unterrichten müssen? „Die Forschung zeigt, dass auch Länder wie die USA, die schon mehr Erfahrung mit informatischer Bildung gemacht haben, nur mühsam traditionelles Programmieren erfolgreich in den Unterricht integrieren konnten. Wie so oft liegt die Lösung in einem Kompromiss.“ [Fit17] Der Kompromiss lautet „Computational Thinking“, welches ein Ansatz ist, der sich auf Konzepte und Problemlösungsstrategien allgemeiner Relevanz fokussiert. Zudem sind die Lehrinhalte von Computational Thinking nicht nur für Informatiker relevant, sondern auch für andere Wissenschaften wie Mathematik, Naturwissenschaften, Sprachen, Kunst und Design, wodurch es noch attraktiver wird und ein Vorteil für alle Beteiligten ist und nicht nur für angehende Programmierer.

Aus den zuvor genannten Argumenten wird es also erkenntlich, dass Schüler schon frühzeitig im Grundschulalter mit den Grundlagen der informatischen Bildung vertraut gemacht werden müssen, damit diese im späteren Lebensverlauf nicht vor dem Programmieren abschrecken und die frühzeitig erlernten Grundlagen im späteren Lebensverlauf ausbauen können.

1.1 Aufgabenstellung und Zielsetzung

Im Rahmen dieser Bachelorarbeit soll eine Webapplikation erstellt werden, mit deren Hilfe Kinder spielerisch das „Computational Thinking“ erlernen können.

Hierzu soll ein Spiel in Anlehnung an eine bereits existierende Webapplikation (Lightbot) mit erweiterter Funktionalität erstellt werden. Dabei wird der Schwerpunkt nicht auf die graphische Darstellung des Spiels gelegt, sondern ergänzend zur sequentiellen Ausführung von Einzelanweisungen zusätzliche Konzepte imperativer Programmiersprachen spielerisch nutzen:

- Variablen zur Speicherung von Zuständen
- Schleifen mit Bedingungen
- Funktionen mit Parametern und Rückgabewerten
- ggf. Collection-artige Datenstrukturen wie Listen, Queues, Stacks

Zudem soll mit Hilfe einer graphischen Darstellung und Erklärung der Konzepte geholfen werden diese zu verstehen und zur Lösung von Problemen spielerisch anzuwenden. Die genannten Konzepte sollen hierzu mittels verschiedener Level und steigendem Schwierigkeitsgrad schrittweise und spielerisch eingeführt werden.

Die Applikation soll als Stand-alone Webapplikation (Single Page App) für Desktop- und Tablet-Browser mittels eines etablierten Responsive Web Frameworks konzipiert werden, die ohne eine zentrale Server-Komponente auskommt und daher flexibel in Schulen auf Laptops/Desktops oder Tablets eingesetzt werden kann.

1.2 Aufbau der Arbeit

Im Ersten Kapitel wird eine Einleitung in das Thema geliefert und zudem die Aufgabenstellung und Zielsetzung erläutert. Das zweite Kapitel beschäftigt sich mit den Grundlagen der Bachelorarbeit und stellt eine ähnliche Webanwendung vor, welche bereits existiert und außerdem wird auf die Zielgruppe eingegangen und das Spielkonzept vorgestellt. Im dritten Kapitel werden die Anforderungen, welche an die Webanwendung gestellt werden, genauer spezifiziert und im vierten Kapitel wird die Architektur und die Implementation der Webanwendung vorgestellt. Im vorletzten Kapitel werden die Tests und die Evaluation geliefert und das letzte Kapitel gibt einen Ausblick.

Kapitel 2 - Grundlagen

Dieses Kapitel beschäftigt sich mit den Grundlagen dieser Bachelorarbeit. Es wird eine Einführung in das Thema geliefert, eine bereits existierende Webanwendung (Lightbot) präsentiert, auf die Zielgruppe eingegangen und das Spielkonzept vorgestellt.

2.1 Was ist „Computational Thinking“?

Unter „Computational Thinking“ versteht man die Kompetenz ein Problem zu erkennen, das mithilfe eines Computers gelöst werden kann und anschließend in einer Abfolge von Teilschritten von einem Menschen oder einem Computer lösen zu lassen. Dabei bezieht sich das Computational Thinking auf die individuelle Fähigkeit einer Person eine Problemstellung zu identifizieren und anschließend in Teilprobleme aufzuteilen, um eine Lösungsstrategie zu erstellen. Dabei ist es wichtig die Lösungsstrategie so formalisiert darzustellen, dass sie von einem Menschen oder von einem Computer nachvollzogen werden kann und anschließend ausgeführt werden kann. Die Konzeption und Umsetzung solch einer Lösungsstrategie kann zusätzlich durch die Strukturierung und Verarbeitung von vorliegenden oder gewonnen Daten unterstützt werden. Außerdem ist die Umsetzung solch einer Lösungsstrategie unabhängig von der Programmiersprache. [Uni17]

Decomposition	Die Aufteilung der Daten, Prozesse oder Probleme in kleinere Teilprobleme.
Pattern Recognition	Das Erkennen von Mustern, Trends und Regelmäßigkeiten in Daten.
Abstraction	Die Identifizierung der allgemeinen Prinzipien, welche das Muster erzeugen.
Algorithm Design	Das Entwickeln einer Schritt für Schritt Anleitung zur Lösung dieser und ähnliche Probleme.

Tabelle 1: Grundlagen des Computational Thinkings [Com17]

2.1.1 Beispiele für das Computational Thinking

Decomposition: Bei der Berechnung des Durchschnitts eines Notenspiegels in der Schule wird das Problem in drei Teilprobleme aufgeteilt. Im erste Teil werden alle Noten aufsummiert, im zweiten wird die Gesamtanzahl der Schüler gezählt und im letzten Schritt wird die Summe aller Noten durch die Anzahl der Schüler dividiert, um den Durchschnitt zu erhalten.

Pattern Recognition: Bei der Gesichtserkennung einer Kamera wird auf das Muster eines Gesichts zurückgegriffen, welches in der Software der Kamera gespeichert ist. Dabei achtet die Kamera zur Laufzeit bei der Gesichtserkennung auf die Form des Gesichtes, die Augen, die Nase und den proportionalen Abstand zwischen den einzelnen Gesichtspartien und klassifiziert dem entsprechend ein Objekt mit diesen Eigenschaften als ein Gesicht.

Abstraction: Bei der Klassifizierung von Objekten aus dem Straßenverkehr unterscheiden sich zwar ein Auto, ein Lastkraftwagen und ein Bus in der Optik, aber im Allgemeinen können Sie alle auf den Begriff „Fahrzeug“ reduziert werden.

Algorithm Design: Wenn ein Programm eine Liste durchgehen soll, in welcher Personen mit deren Alter eingetragen sind und es entschieden werden soll, ob diese in die Kategorie Jugendliche (bis 18 Jahre), Erwachsene (von 18-62 Jahre) oder Rentner (ab 63 Jahre) gehören. Kann ein Algorithmus entwickelt werden, welcher sich das Alter der jeweiligen Person anschaut und diese dementsprechend klassifiziert. Da dieser Algorithmus allgemein entwickelt wurde, kann er anschließend für beliebige neue Datensätze (ähnliche Probleme) erneut angewendet werden.

2.2 Spielerisches Lernen

2.2.1 Begriffsbestimmung

Spiel: „… eine freiwillige Handlung oder Betätigung, die innerhalb gewisser festgesetzter Grenzen von Zeit und Raum nach freiwillig angenommenen, aber unbedingt bindenden Regeln verrichtet wird, ihr Ziel in sich selber hat und begleitet wird von einem Gefühl der Spannung und Freude und einem Bewusstsein des Andersseins als das gewöhnliche Leben“ [Hui04, S.37]

Lernen: Ein beabsichtigter, beiläufiger, individueller oder kollektiver Prozess in welchem man geistige oder körperliche Fähigkeiten und Fertigkeiten erwirbt. [Lex17]

2.2.2 „Spielerisches Lernen“ bei Grundschulkindern

Das Thema des spielenden Lernens beinhaltet viele Aspekte, zunächst gilt es aber sich mit der Frage auseinanderzusetzen was überhaupt „spielendes Lernen“ ist, wo es stattfindet und welchen Sinn es hat? Der Begriff „spielendes Lernen“ beinhaltet einen Widerspruch in sich. Denn das Lernen ist ein Mittel zum Zweck, welches auf Ergebnisse abzielt [Oer96, S.6ff.]. Das Spielen dahingegen findet aus eigener Motivation statt und kann jederzeit beendet werden, was beim Lernen nicht immer der Fall ist. Ein Vorteil des Spielens gegenüber dem Lernen ist, dass der Spielende einen Lerneffekt haben kann, ohne dies wahrzunehmen. Dies ist zudem sehr vorteilhaft, wenn der Spielende negative Erfahrungen mit herkömmlichen Lernmethoden gemacht hat. Beispielsweise das Ankämpfen gegen die Langeweile und zudem die Demotivation und der Frust bei Fehlversuchen spielen einen großen Faktor. Beim spielerischen Lernen werden alle diese Faktoren in den meisten Fällen vom Spielenden gar nicht erst wahrgenommen, da sich dieser auf den Faktor „Spaß“ konzentriert. Somit lässt sich in Bezug auf die Zielgruppe (Grundschulkinder zwischen sechs bis zehn Jahre) sagen, dass die Methode des „spielerischen Lernens“ bei den Grundschulkindern nicht nur einen Lerneffekt herbeiruft, sondern den Kindern auch noch Spaß macht und somit eine effektive Alternative zu herkömmlichen Lernmethoden ist. Dies ist ein sehr

wichtiger Aspekt und einer der Grundlagen auf welche diese Arbeit aufbaut. Ein weiteres Argument liefert der Neurobiologe und Erziehungswissenschaftler Prof. Dr. med. Henning Scheich, welcher besagt, dass „Lernprozesse bis zur Pubertät [...] zur Abspeicherung von Informationen und gleichzeitig[er] Strukturierung des noch unfertigen Gehirns im Sinne von später ausbaubaren Fähigkeiten [führe].“ [Mft17, S.10] Zudem hat „die Strukturierung in der Hirnrinde [...] ihren Höhepunkt im Vorschul- und Grundschulbereich“ [Mft17, S.10], was auch noch ein weiteres Argument dafür ist, die Methode des „spielerischen Lernens“ in dieser Altersgruppe anzuwenden. Zusammengefasst sprechen also mehrere Argumente für eine frühzeitige Einführung in das „Computational Thinking“, damit die Schüler die Grundlagen frühzeitig erlernen und im späteren Verlauf ihres Lebens auf diese ihre weiteren Fähigkeiten ausbauen können. Mit dem Ansatz des Computational Thinkings wird es den Kindern beispielsweise ermöglicht den Zusammenhang zwischen sequentiellen und parallelen Prozessen zu verstehen. Im ersten Augenblick mag das alles kompliziert erscheinen und man mag denken, dass Kinder im Grundschulalter gar nicht in der Lage sind solche komplexe Sachverhalte nachzuvollziehen. Mit einem vereinfachten Beispiel sieht das Ganze schon anders aus. Mit dem Beispiel, dass eine Großmutter einen Kuchen backen will und sie weiß, dass sie den Zuckerguss schon vorbereiten kann, während der Kuchen im Ofen ist, ist es auch für Kinder verständlich. Dabei geht es im Computational Thinking um die Formulierung eines Problems, die Repräsentation einer Lösung und die Bewertung der Lösung [Fit17]. Im zuvor genannten Beispiel beinhaltet die Problemformulierung, die Erkenntnis, dass der Zuckerguss erst am Ende auf den gebackenen Kuchen kommt. Die Repräsentation der Lösung wäre ein Backrezept für den Kuchen, welches eine detaillierte Beschreibung der einzelnen parallelen und aufeinanderfolgenden Schritte beinhaltet. Und für die Bewertung des Lösungskonzeptes (Backrezept) könnte die Großmutter das Backrezept an ihre Enkel oder jede andere beliebige Person delegieren, damit diese die Arbeitsschritte mithilfe des Rezepts selber durchführen und bewerten können.

2.3 Anlehnung an das Konzept von „Lightbot“

„Lightbot“ ist eine Webanwendung von den Lightbot Inc. Entwicklern, welche als ein Spiel realisiert wurde mit dem Zweck Kindern spielerisch das Konzept der Programmierung näherzubringen. Umgesetzt wurde die Webapplikation mithilfe von Flash und OpenFL [Wik17].
Das Ziel des Spiels ist es einen kleinen Roboter mithilfe von sequentiellen Anweisungen (gehen, umdrehen, springen und Licht einschalten) durch ein Labyrinth zur Zielposition zu führen. Zudem wird mit höherem Level die Schwierigkeitsstufe des Spiels erhöht. Dabei bietet „Lightbot“ drei verschiedene Modi an: Grundlagen, Prozeduren und Schleifen [Lig17a].

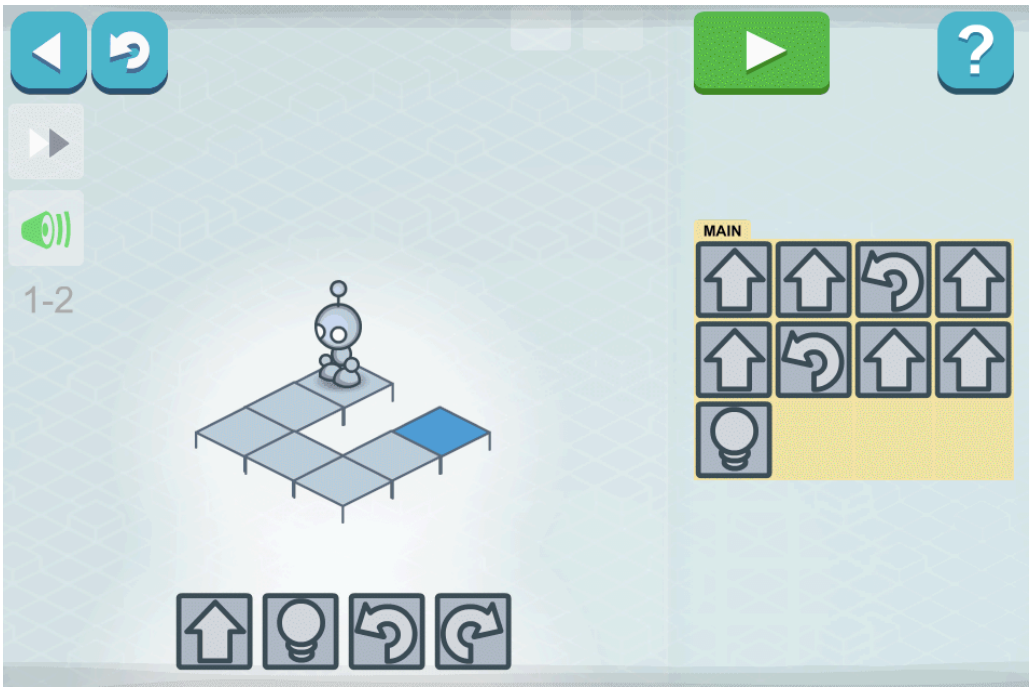


Abbildung 1: Lightbot-Modus: Grundlagen [Lig17b]

Im Grundlagen-Modus der Lightbot Applikation wird es dem Spieler ermöglicht mithilfe der Steuerungsoptionen vorwärtsgehen, Licht einschalten, nach links umdrehen und nach rechts umdrehen den Roboter von der Startposition zur Endposition zu navigieren. Zudem muss auf allen blauen Feldern die Steuerungsoption „Licht einschalten" benutzt werden, um das Level erfolgreich zu absolvieren. Dabei werden die einzelnen Steuerelemente durchs anklicken oder per Drag and Drop selektiert und erscheinen anschließend in der gelben Box, welche mit „Main" gekennzeichnet ist. Durch das anklicken des grünen Play-Buttons werden die einzelnen Anweisungen aus der Main-Box sequentiell ausgeführt.

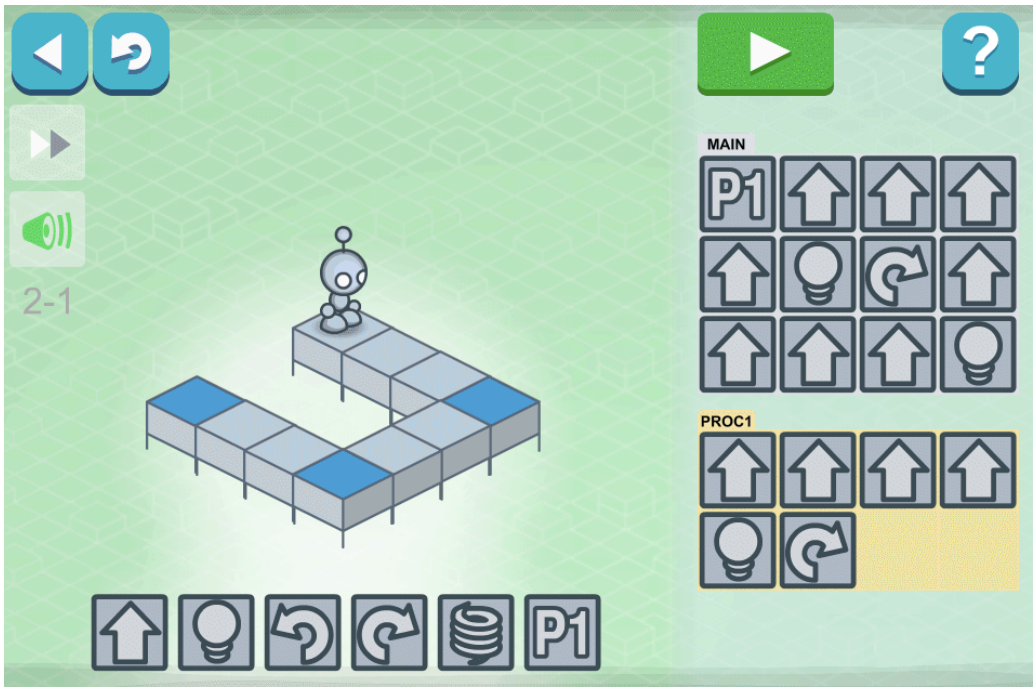


Abbildung 2: Lightbot-Modus: Prozeduren [Lig17b]

Im Prozeduren-Modus, wird dem Spieler zusätzlich zur Main noch eine weitere Box zur Verfügung gestellt, welche mit „PROC1“ gekennzeichnet ist. Diese Box stellt dem Spieler zusätzlichen Platz zur Verfügung, um mehr Anweisungen durchführen zu können. Dabei wird die Prozedur, welche in der „PROC1“ ausgelagert ist, in der Main mithilfe der „P1“ Anweisung aufgerufen.

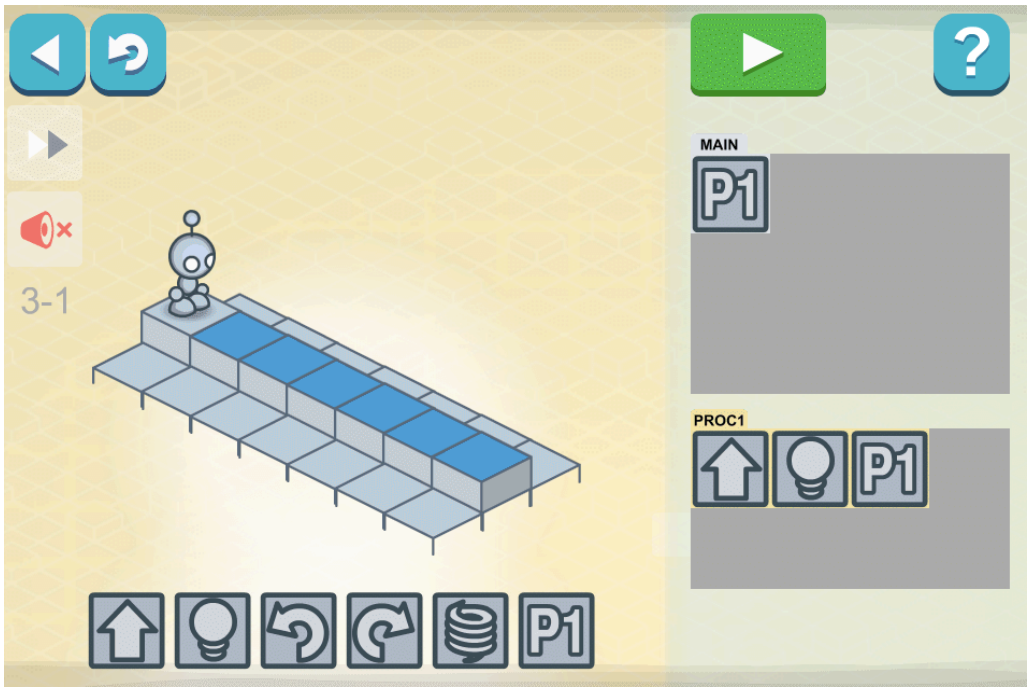


Abbildung 3: Lightbot-Modus: Schleifen [Lig17b]

Im Schleifen-Modus, wird der Spieler dazu aufgefordert in der „PROC1“ eine Schleife zu erstellen, welche in der Main rekursiv aufgerufen werden kann.

Das Konzept von Lightbot stell somit drei verschiedene Modi zur Verfügung, um dem Spieler das „Computational Thinking“ näherzubringen. Jedoch sind nicht alle Aspekte berücksichtigt worden und somit ist das Konzept erweiterbar. Beispielsweise wird dem Spieler nicht vermittelt, was eine Variable ist oder was für einfache Datenstrukturen es gibt. Zudem hat der Spieler auch keine Möglichkeit Kontrollstrukturen zu verwenden.

2.4 Das Konzept des Spieles

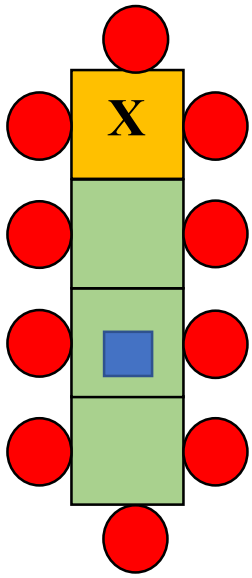
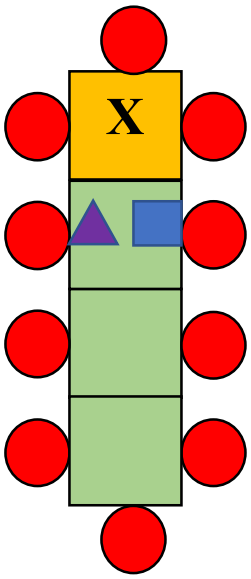
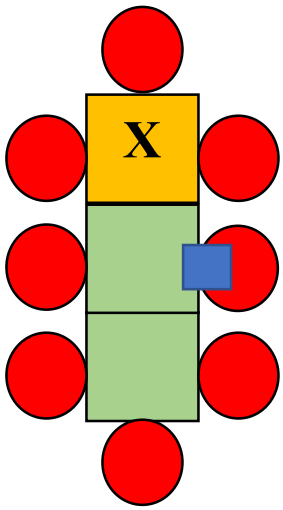
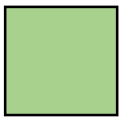
		
Spielfigur bewegt sich auf den bewegbaren Feldern.	Spielfigur ist auf dem Feld des einsammelbaren Objektes.	Spielfigur kollidiert mit Hindernis.

Abbildung 4: Konzept des Spiels

Legende:

Spielfigur	Begehbares Feld	Zielposition	Hindernis	Einsammelbares Objekt
				

Das Spiel „Go!Farmer“ basiert auf ein quadratisches Spielfeld, welches aus einer Tabelle besteht. Auf dem Spielfeld (in einer Zelle der Tabelle) befindet sich eine Spielfigur, welche die Möglichkeit hat sich auf den begehbaren Feldern von der Startposition bis zu Endposition (Zielposition) des Spielfeldes zu bewegen. Des Weiteren sind die begehbaren Felder von Hindernissen umgeben, welche dazu dienen, dass der Spieler nur auf den begehbaren Feldern bis zur Zielposition gelangen kann und anderenfalls mit den Hindernissen kollidiert. Die Spielfigur hat die Möglichkeiten: in Blickrichtung bewegen, nach links drehen, nach rechts drehen und Objekt einsammeln. Diese Aktionen werden durch das Anklicken in der Webanwendung des Spiels in die Aktionsauswahl hinzugefügt. Nachdem die Sequenz der einzelnen Aktionen erstellt wurde (alle Teilaktion in die Aktionsauswahl hinzugefügt) und der Start-Button geklickt wurde, werden alle Aktionen sequentiell nacheinander ausgeführt, bis der Spieler die Zielposition erreicht oder gegen ein Hindernis stößt.

Das Spiel endet und lädt das Level von vorne, wenn folgendes passiert:

- Die Spielfigur hat nicht die Endposition (Zielposition) des Levels erreicht.
- Die Spielfigur hat nicht alle einsammelbare Objekte eingesammelt.
- Die Spielfigur stößt gegen ein Hindernis.

„Go!Farmer“ kann auf vielfältige Arten variiert werden:

- Die Spielfigur, die Hindernisse und die einsammelbaren Objekte könnten einfach optisch verändert werden. (andere Bilder)
- Es könnte eine neue Spieler-Aktion ergänzt werden (beispielsweise Früchte auf Feldern hinlegen statt aufheben)
- Das Spielfeld könnte beliebig gestaltet werden mit neuen Feldern (beispielsweise könnte ein Teleportationsfeld hinzugefügt werden, von welchem sich die Spielfigur auf ein anderes Feld teleportiert, um Hindernisse zum umgehen)
- usw.

Die nachfolgende Architektur wurde so entworfen, dass solche Variationen möglich sind und nachträglich ergänzt werden können.

Kapitel 3 – Anforderungsanalyse

In diesem Kapitel wird auf die Anforderungen eingegangen, welche an die Webanwendung gestellt werden. Dazu wird zunächst auf die Anforderungen eingegangen, welche sich aus der bereits existierenden Webanwendung (Lightbot) ergeben haben und anschließend auf die Erweiterungen.

3.1 Anforderungen an die Webapplikation

Die Anforderungen an die Webapplikation ergeben sich aus der Anlehnung an das Konzept von Lightbot, den möglichen Erweiterungen und der Analyse von Computational Thinking.

3.1.1 Produktfunktionen (Funktionale Requirements)

PF1 - Steuerung

Aus der Anlehnung an das Konzept von Lightbot ergibt sich die Produktfunktion der Steuerung, welche die Steuerungselemente „in Blickrichtung bewegen“, „nach links drehen“ und nach „rechts drehen beinhaltet“ beinhaltet.

- PF1.1 in Blickrichtung bewegen
- PF1.2 nach links drehen
- PF1.3 nach rechts drehen

PF2 – Objekt einsammeln

Aus dem ersten Punkt der Aufgabenstellung „Variablen zur Speicherung von Zuständen“ ergibt sich die Produktfunktion „Objekte einsammeln“, welche dem Nutzer die Inkrementierung einer Variable näherbringen soll, indem ein Objekt auf dem Spielfeld eingesammelt wird und die dazugehörige Variable in der Anzeige aktualisiert wird.

PF3 – Schleife durchlaufen

Aus dem zweiten Punkt der Aufgabenstellung „Schleifen mit Bedingungen“ in Bezug auf „Schleifen“ und dem Punkt „Pattern Recognition“ des Computational Thinkings ergibt sich die Produktfunktion eine Schleife zu durchlaufen, in welcher die Spielfigur ein erkanntes Muster von Anweisungen bis zum Schleifenende wiederholt.

PF4 – Bis zum nächsten Hindernis gehen

Aus dem zweiten Punkt der Aufgabenstellung „Schleifen mit Bedingung“ in Bezug auf „Bedingung“ ergibt sich die Produktfunktion „bis zum nächsten Hindernis gehen“, welche es der Spielfigur ermöglicht sich solange in Blickrichtung zu bewegen bis ein Hindernis kommt.

PF5- Routine (Prozedur) definieren

Aus dem dritten Punkt der Aufgabenstellung „Funktionen mit Parametern und Rückgabewerten“ und den Punkt „Algorithm Design“ des Computational Thinkings ergibt sich die Produktfunktion „Routine (Prozedur) definieren“, welche es dem Spieler ermöglicht eine Prozedur zu erstellen und je nach Level die erstellte Prozedur (Unterprogramm) beliebig oft im Hauptprogramm aufzurufen.

PF6- Teilaktionen auswählen und Sequenz starten

Aus dem Punkt „Decomposition“ des Computational Thinkings ergibt sich die Produktfunktion, die ausgewählten Teilaktionen zu starten, damit diese sequentiell hintereinander ausgeführt werden. Der Bezug zum Computational Thinking dabei ist die Aufteilung des Gesamtproblems (Spielfigur vom Start bis zum Ende des Spielfeldes navigieren) in Teilprobleme (gehen, links drehen, rechts drehen, Feld überprüfen und Objekt aufheben).

PF-7 Level Konzept mit dynamischer Map-Erstellung

Aus dem Punkt „Algorithm Design“ des Computational Thinkings ergibt sich die Produktfunktion der dynamischen Spielfelderstellung, um verschiedene Level zu erstellen. Der Bezug zum Computational Thinking dabei ist es eine Lösung nicht nur für ein Problem (Level) anwenden zu können, sondern auch für ähnliche Probleme (höhere Level).

3.1.2 Nicht funktionale Requirements

NFR1- Single Player Game

Das nicht funktionale Requirement die Webapplikation als ein Single Player zu realisieren ergibt sich aus der Anlehnung an das Konzept von Lightbot.

NFR2- 2D Game

Die Webapplikation soll als ein 2D Game umgesetzt werden, damit der Spieler sich nicht zu sehr von der graphischen Darstellung des Spieles ablenken lässt, sondern sich auf die Lösungsfindung der einzelnen Level konzentriert.

NFR3- Desktop Browser

Die Webanwendung soll unabhängig von der Hardware in allen gängigen Desktopbrowsern spielbar sein, um dem Spieler einen uneingeschränkten Zugriff zu ermöglichen und die Nutzung der Webanwendung in der Schule zu ermöglichen (Computer-Raum der Schule).

NFR5 – Tablet Browser

Die Webanwendung soll in allen Tablet-Browsern spielbar sein, damit die Schüler zu Hause auch das Tablet nutzten können und nicht nur den PC und um die Zugriffsmöglichkeiten zu erweitern.

NFR6 – Mobile Browser

Die Webanwendung soll in allen gängigen Mobile-Browsern spielbar sein, damit die Schüler die Webanwendung mobil nutzen können und unterwegs die Webanwendung wie eine Spiel-App nutzen können.

NFR7- Stand-alone Webapplikation

Die Webapplikation soll mithilfe eines etablierten Responsive Web Frameworks umgesetzt werden, um eine dynamische Ansicht zu liefern und die Orientierung auf der Seite der Webanwendung zu erleichtern.

3.2 Abdeckung der Anforderungen

ID	Kurztitel	Erfüllt	Teilw. erfüllt	Nicht erfüllt
[PF1]	**Steuerung**	**X**		
[PF2]	**Objekt einsammeln**	**X**		
[PF3]	**Schleife durchlaufen**	**X**		
[PF4]	**Bis zum nächsten Hindernis gehen**	**X**		
[PF5]	**Routine (Prozedur) definieren**	**X**		
[PF6]	**Teilaktionen auswählen und Sequenz starten**	**X**		
[PF7]	**Level Konzept mit dynamischer Map-Erstellung**	**X**		
[NFR1]	**Single Player Game**	**X**		

[NFR2]	**2D Game**	**X**		
[NFR4]	**Desktop Browser**	**X**		
[NFR5]	**Tablet Browser**	**X**		
[NFR6]	**Mobile Browser**	**X**		
[NFR7]	**Stand-alone Webapplikation**	**X**		

Tabelle 2: Abdeckung der Anforderungen

Kapitel 4 – Architektur/Implementation

In diesem Kapitel wird die Architektur (Model View Controller) beschrieben, das Levelkonzept vorgestellt, die Implementation der in Kapitel 3 spezifizierten Webanwendung beschrieben und außerdem werden noch einige nennenswerte Ausschnitte des Quellcodes erläutert.

4.1 MVC-Pattern

Die Architektur der Webapplikation wurde mithilfe des Model-View-Controller Patterns (MVC) umgesetzt. Die Vorteile der MVC Architektur sind, dass das Model, welches die einzelnen Entitäten beinhaltet, die View welche für die Aktualisierung der graphischen Darstellung zuständig ist und der Controller, welcher für die Interaktion des Users zuständig ist, modularisiert arbeiten. Das Model arbeitet somit für sich selbst, wobei der Controller nach User Interaktion das Model manipulieren kann. Außerdem kann die View auf das Model zugreifen, um die graphische Darstellung zu aktualisieren, meistens geschieht dies nach einer User Interaktion und einer Aufforderung des Controllers an die View eine Aktualisierung vorzunehmen.

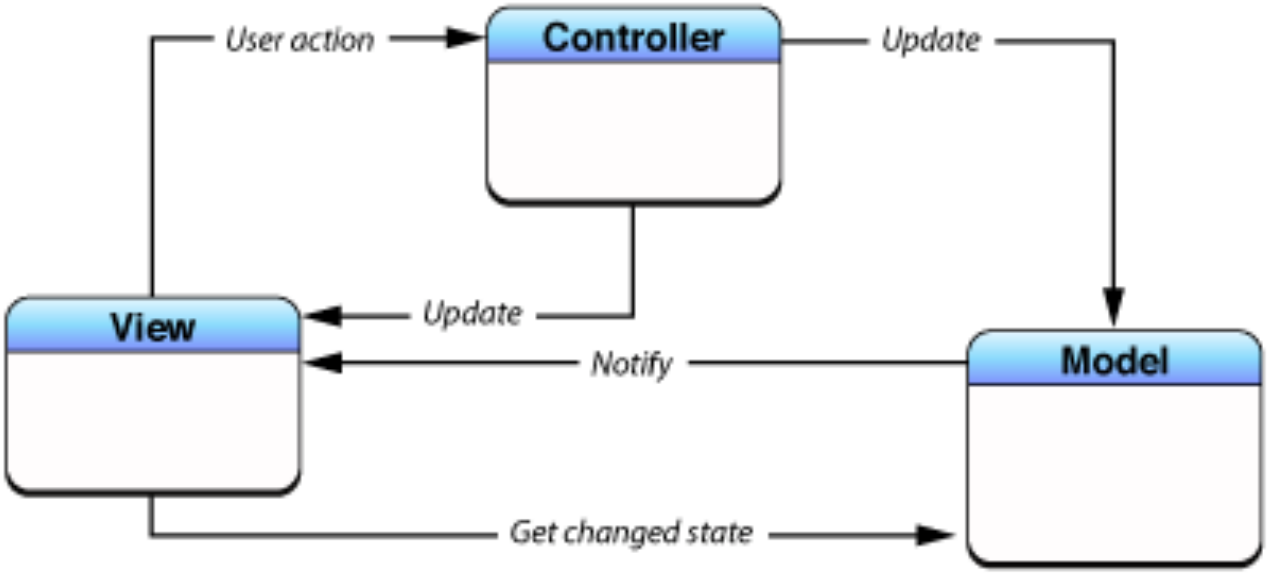


Abbildung 5: MVC Pattern [Inf17]

4.1.1 Architektur der Webanwendung

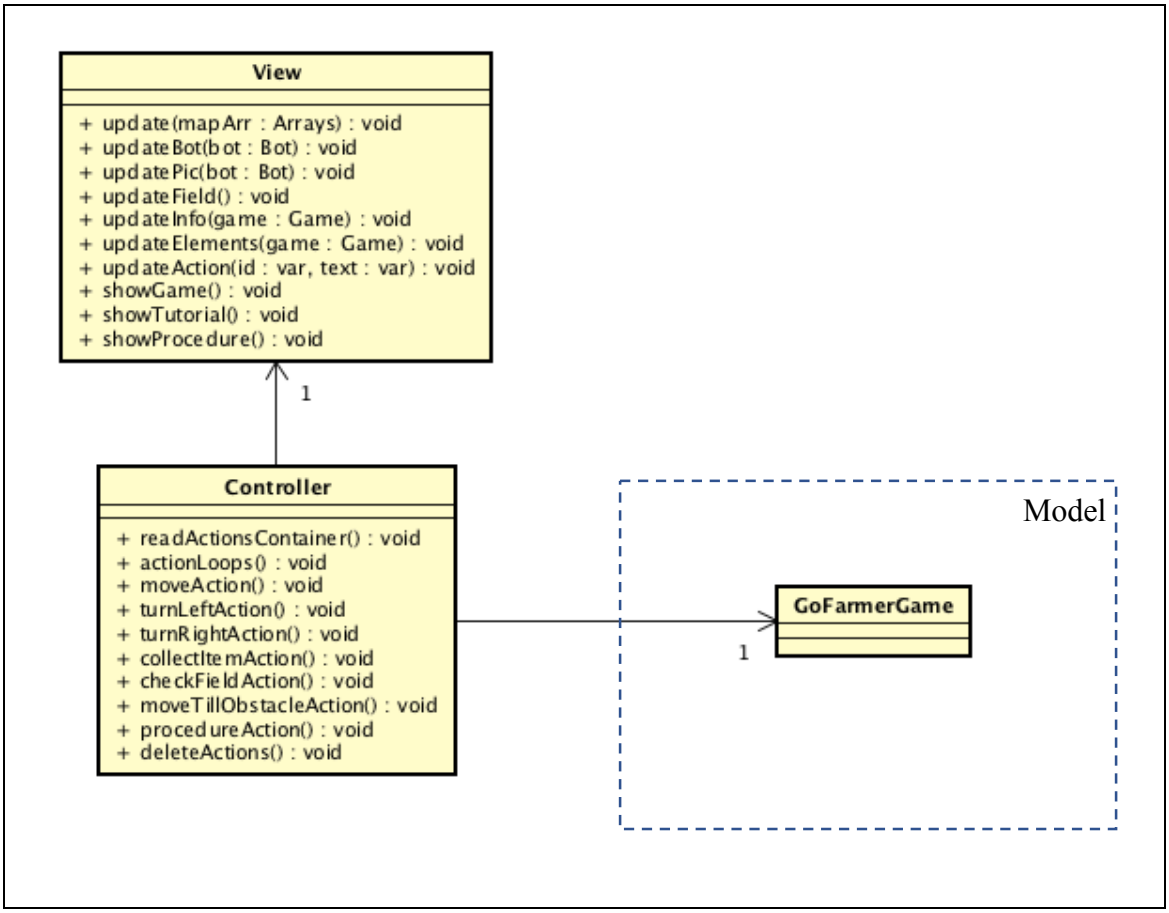


Abbildung 6: Architektur der Webanwendung

4.1.2 Methoden der View

update(mapArr) – Erstellt aus dem mapArray (Array mit den einzelnen Einträgen für den Zelleninhalt der Tabelle) das Spielfeld und update es (beispielsweise beim Aufruf eines neuen Levels).

updateBot(bot) – Updatet die Position des Bots auf dem Spielfeld.

updatePic(bot) – Updatet das Bild des Bots der Blickrichtung entsprechend.

updateField() – Updatet den Zelleninhalt. Beispielsweise, wenn ein einsammelbares Objekt aufgehoben wurde, wird es in der View von der jeweiligen Zelle entfernt.

updateInfo(game) – Updatet die Informationsanzeige (Anzahl der einsammelbaren Objekte und Anzahl der Wiederholungen in der graphischen Darstellung der View).

updateElements(game) – Update die Sichtbarkeit der einzelnen Elemente in der View (zum Beispiel das Ein- und Ausblenden von Aktionen in verschiedenen Level).

updateAction(id,text) – Updatet die Informationsanzeige der Aktionsauswahl und zeigt die ausgewählte Aktion in der View.

showGame() – Zeigt das Spiel in der View und verbirgt die Erklärung des Spiels.

showTutorial() – Zeigt die Erklärung des Spiels in der View und verbirgt das Spiel.

showProcedure() – Zeigt den Prozedur-Tab (Routinen-Tab) in der View, in welcher eine Routine erstellt werden kann.

4.1.3 Methoden des Controllers

readActionsContainer() – Geht die Aktionsauswahl durch und fügt für jede ausgewählte Aktion eine Aktion in das Actions-Array des Bots (Array mit Einträgen für Aktionen, welche ausgeführt werden sollen).

actionLoops() – Erstellt eine Schleife indem die Kind-Elemente der Aktionsauswahl (Aktionen) kopiert werden und für jede Wiederholung am Ende der Aktionsauswahl hinzugefügt werden. (Beispiel: „Aktionsauswahl = gehen, linksumdrehen & Wiederholung = 2“ würde gleichbedeutend sein mit „Aktionsauswahl = gehen, linksumdrehen, gehen, linksumdrehen“).

deleteActions() – Geht die Aktionsauswahl durch und entfernt alle ausgewählten Elemente.

Controller-Methoden, welche mit einem onclick-eventListener (ein Listener, welcher prüft, ob ein gewisses Element angeklickt wurde) ausgestattet wurden:

- **moveAction()**
- **turnLeftAction()**
- **turnRightAction()**
- **collectItemAction()**
- **checkFieldAction()**
- **moveTillObstacleAction()**
- **procedureAction()**

Diese Methoden benachrichtigen die View und fordern diese dazu auf die ausgewählten Aktionen in der View zu aktualisieren (nach dem Anklicken einer Aktion, soll diese in der View angezeigt werden). Als Parameter an die View wird eine ID vergeben, welche der View mitteilt welche Aktion hinzugefügt werden soll und ein Text, welcher von der View angezeigt wird. Zum Beispiel wird mit einer ID: „move“ die ausgewählte Aktion in die Aktionsauswahl hinzugefügt und mit einem Text: „geradeaus gehen“ in der Anzeige in der View als „Du hast (geradeaus gehen) hinzugefügt“ dargestellt.

4.2 Architektur des Models

Die vorherige Beschreibung des MVC-Patterns hat gezeigt, dass der Controller für die Interaktion mit dem User zuständig ist und die View für das Updaten der graphischen Ansicht. Im Folgenden wird auf die Architektur des Models eingegangen, welches für die Modellierung des Spieles zuständig ist.

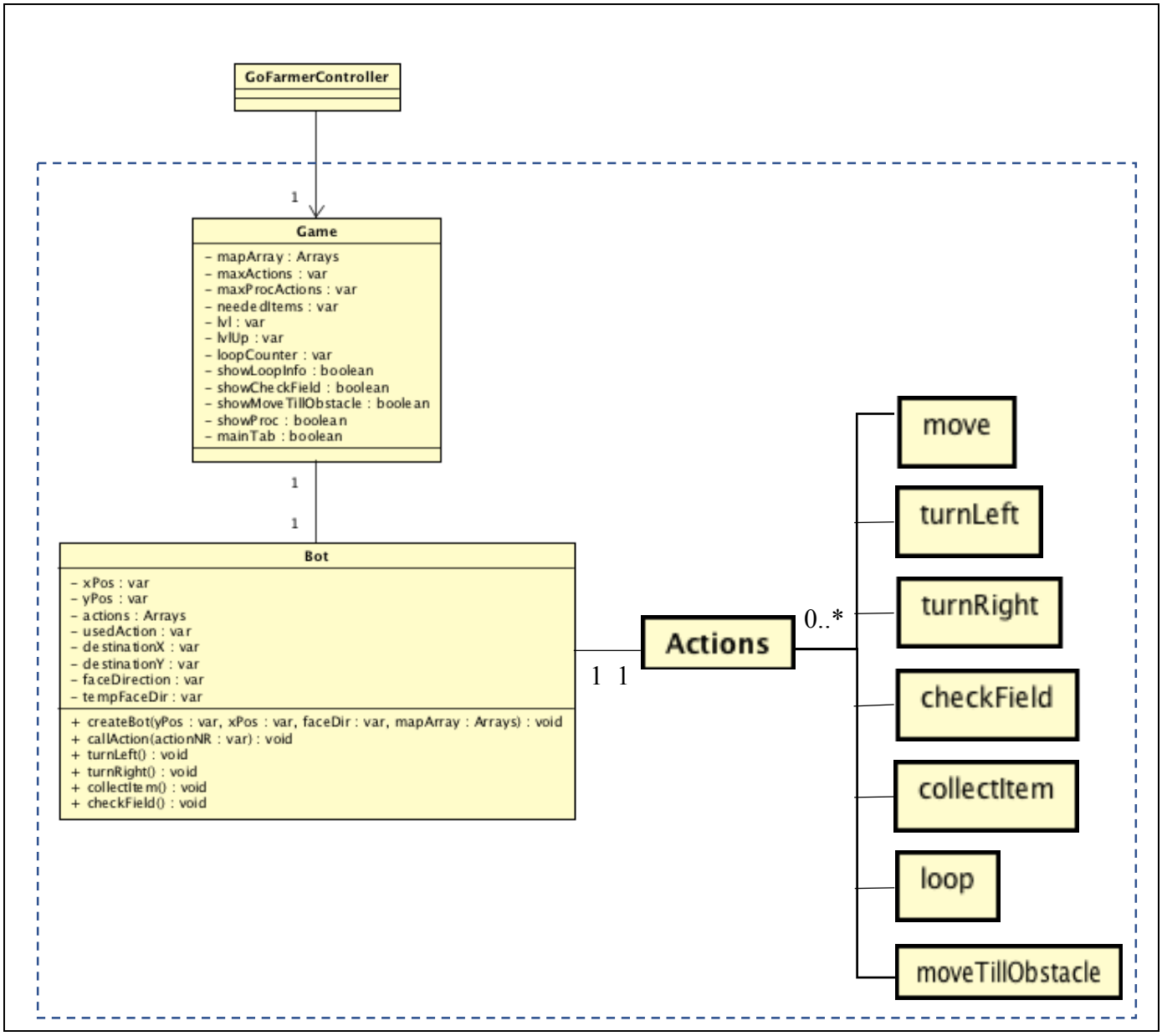


Abbildung 7: Architektur des Models

4.2.1 Variablen und Methoden des Models

Variablen des Game-Objekts:

- **mapArray** (Ein zweidimensionales Array mit den einzelnen Einträgen für den Tabelleninhalt).
- **maxActions** (Die Anzahl der maximal auswählbaren Aktionen für die Main/Hauptprogramm).
- **maxProcActions** (Die Anzahl der maximal auswählbaren Aktionen für die Prozedur).
- **neededItems** (Die Anzahl der benötigten einsammelbaren Objekte, um ein Level abzuschließen).
- **lvl** (Das aktuelle Level).
- **loopCounter** (Die Anzahl der Schleifen-Wiederholungen).
- **showLoopInfo** (Ein Boolean, welcher angibt, ob das Element für die Wiederholungen in einem Level angezeigt werden soll).
- **showCheckField** (Ein Boolean, welcher angibt, ob die Aktionen „Feld überprüfen“ und „neue Frucht aufheben“ in einem Level angezeigt werden sollen).
- **showMoveTillObstacle** (Ein Boolean, welcher angibt, ob die Aktion „bis zum nächsten Hindernis gehen“ in einem Level angezeigt werden soll).
- **showProc** (Ein Boolean, welcher angibt, ob in einem Level Prozeduren erstellt werden können).
- **mainTab** (Ein Boolean, welcher angibt in welchem Tab der Spieler sich gerade befindet. Zum Beispiel bei „mainTab = true“ im Hauptprogramm und bei „mainTab = false“ im Prozeduren-Tab).

Variablen des Bot-Objekts:

- **xPos** (Die aktuelle Position des Bots auf der X-Achse).
- **yPos** (Die aktuelle Position des Bots auf der Y-Achse).
- **actions** (Ein Array mit Einträgen für die Aktionen, welche vom Bot ausgeführt werden sollen).
- **usedAction** (Die zu Letzt genutzte Aktion des Bots).

- **destinationX** (Die angestrebte Position des Bots auf der X-Achse nach der Auswahl einer Bewegungs-Aktion).
- **destinationY** (Die angestrebte Position des Bots auf der Y-Achse nach der Auswahl einer Bewegungs-Aktion).
- **faceDirection** (Die aktuelle Blickrichtung des Bots).
- **tempFaceDir** (Die temporäre Blickrichtung des Bots).

Methoden des Bots:

createBot(yPos, xPos, faceDir, mapArray) – Erstellt einen Bot mit der ausgewählten Blickrichtung (faceDir) und Position (xPos & yPos) in dem zweidimensionalen Array (mapArray), aus welchem später das Spielfeld erstellt wird.

callAction(actionNr) – Führt die ausgewählte Aktionen aus dem „actions-Array“ aus.

turnLeft() – Ändert die Blickrichtung des Bots in Abhängigkeit von der aktuellen Position nach links.

turnRight() – Ändert die Blickrichtung des Bots in Abhängigkeit von der aktuellen Position nach rechts.

collectItem() – Ändert den Eintrag im mapArray von 6 (Eintrag für einen Apfel im dem Spielfeld) in 1 (Eintrag für ein begehbares Feld) um, was nach dem Update der View dazu führt, dass das einsammelbare Objekt (der Apfel) verschwindet.

checkField() – Ändert den Eintrag im mapArray von 7 (Eintrag für ein überprüfbares Feld im Spielfeld) in 6.5 (Eintrag für eine neue Frucht) um, was nach dem Update der View dazu führt, dass das überprüfbare Feld verschwindet und dort eine neue Frucht erscheint.

4.3 Levelkonzept

Das Levelkonzept basiert darauf, dass es mit zunehmenden Level immer schwieriger wird und die Wege bis zum Ziel immer komplexer werden, beispielsweise können mehr Abzweigungen und mehr einsammelbare Objekt in höheren Level auftauchen, zudem wird die Anzahl der Aktionen begrenzt, damit der Spieler nicht beliebig viele Aktionen benutzen kann, um irgendwie zum Ziel zu gelangen, sondern den schnellsten Weg erkennen muss und dann die richtigen Aktionen auswählen muss.

```
{
"mapArray": [
[2, 2, 2, 2, 2, 2, 2, 2, 2],
[0, 0, 0, 3, 3, 3, 0, 0, 0],
[0, 0, 0, 3, 5, 3, 0, 0, 0],
[0, 0, 0, 3, 1, 3, 0, 0, 0],
[0, 0, 0, 3, 1, 3, 0, 0, 0],
[0, 0, 0, 3, 1, 3, 0, 0, 0],
[0, 0, 0, 3, 3, 3, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0]
],
  "maxActions": 3,
  "neededItems": 0,
  "playerYpos": 5,
  "playerXpos": 4,
  "playerFaceDir": "up",
  "showLoopInfo": false,
  "showCheckField": false,
  "showMoveTillObstacle": false,
  "showProc": false
}
```

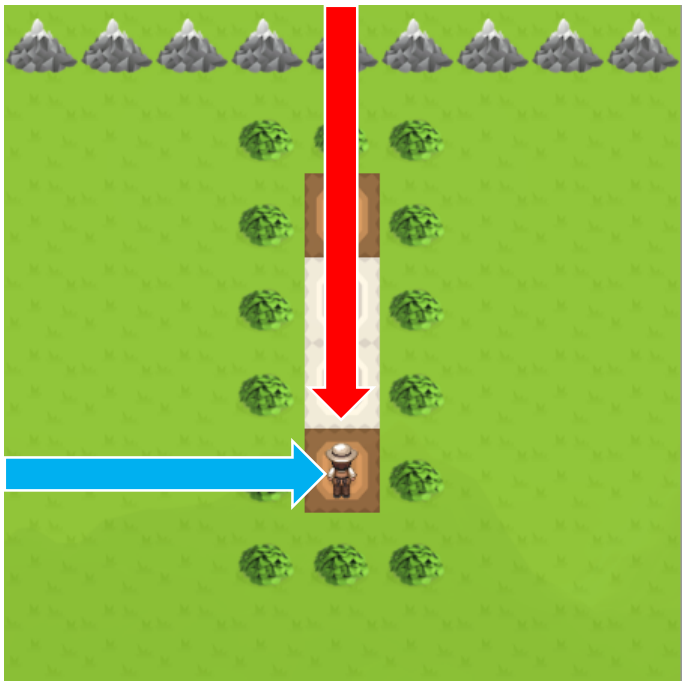


Abbildung 8: lvl0.json (Level Beispiel)

Der Startzustand des Bots (Spielfigur) ist in den einzelnen Levels ausgelagert (JSON-Datei), wie im obigen Beispiel (Abbildung 8). Dabei steht "playerYpos" für die Position der Spielfigur auf der Y-Achse und "playerXpos" für die Position der Spielfigur auf der X-Achse und "playerFaceDir" für die Blickrichtung (up, right, left und down) der Spielfigur. Die Einträge des mapArrays beschreiben den Zelleninhalt der Tabelle an der jeweiligen Position.

Einträge des mapArrays und deren Bedeutung:

1 = begehbares Feld

2 = Berg (Hindernis)

3 = Bush (Hindernis)

4 = Startposition des Bots

5 = Zielposition

6 = Apfel (einsammelbares Objekt)

6.5 = neue einsammelbare Frucht (nach dem überprüfen des „prüfbaren Feldes“)

7 = prüfbares Feld

4.4 Model

Das Model beinhaltet einen Konstruktor für das Game, welche die allgemeinen Informationen beinhaltet, wie zum Beispiel die Anzahl der maximal auswählbaren Aktionen (maxActions) und die Anzahl der benötigten einsammelbaren Objekte (neededItems), um ein Level zu bestehen. Außerdem ist im Model auch der Konstruktor des Bots (Spielfigur), welcher den Zustand des Bots und die Bot-Aktionen beinhaltet.

```
1  // creates the bot in the map at the start-position
2  this.createBot = function (yPos, xPos, faceDir, mapArray) {
       if (xPos < 0 || xPos > mapArray.length - 1) return;
3      if (yPos < 0 || yPos > mapArray.length - 1) return;
4      this.faceDirection = faceDir;
       this.tempFaceDir = faceDir;
5      this.xPos = xPos;
       this.yPos = yPos;
6      mapArray[yPos][xPos] = START_POSITION;
7  };
```

Abbildung 9: Die createBot-Methode (Bot-Methode)

Mithilfe der createBot-Methode (Abbildung 9) wird im mapArray an der ausgewählten Indexposition (yPos, xPos) eine „vier“ eingetragen, welche für die Startposition des Bots steht und außerdem wird mithilfe des „faceDir“-Parameters die Blickrichtung des Bots bestimmt.

```
1  // turns the bot to the left depending on the former faceDirection
2  this.turnLeft = function () {
3      if (this.faceDirection === "left") {
4          this.faceDirection = "down";
5      } else if (this.faceDirection === "down") {
6          this.faceDirection = "right";
7      } else if (this.faceDirection === "right") {
           this.faceDirection = "up";
       } else if (this.faceDirection === "up") {
           this.faceDirection = "left";
       }
   };
```

Abbildung 10: Die turnLeft-Methode (Bot-Methode)

Die turnLeft-Methode (Abbildung 10) des Bots greift auf die aktuelle Blickrichtung des Bots zu und setzt sie dementsprechend einen weiter nach links. Nach demselben Prinzip funktioniert auch die turnRight-Metode des Bots.

4.5 View

4.5.1 HTML-Dokument

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4     <meta charset="UTF-8">
5     <title>Go!Farmer</title>
6     <meta name="viewport" content="width=device-width, initial-scale=0.4">
7     <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
8     <link rel="stylesheet" href="styles.css">
9     <script src="jquery-3.2.1.min.js"></script>
10 </head>
11 <body>
12
13 <!--NAVIGATION-->
14 <header>
15    <div id="navigation" class="w3-bar w3-black"...>
16 </header>
17 <!--GAME-->
18 <section>
19    <div id="level" class="w3-animate-opacity"...>
20 </section>
21 <!--TUTORIAL-->
22 <section>
23    <div id="tutorial"...>
24 </section>
25
```

```
18  <!--FOOTER-->
19  <footer>
      <div class="footer w3-container w3-black w3-center">
20      <span class="fa fa-envelope"> Schazahdah Mokhlis, s.mokhlis@web.de</span>
      </div>
21  </footer>
22  <script src="model.js"></script>
23  <script src="view.js"></script>
    <script src="controller.js"></script>
24  <script src="main.js"></script>
25  </body>
    </html>
```

Abbildung 11: HTML Basis-Dokument des Spiels

Die Umsetzung des Spiels ist mithilfe von JavaScript, jQuery, HTML (Hypertext Markup Language) und CSS (Cascading Style Sheets) erfolgt. In der Abbildung 11 ist das HTML-Dokument zu sehen, welches für die Einblendung des Spiels benutzt wird. Dabei beruht das Grundgerüst des Spiels auf ein HTML Table Element, welches dazu dient das Spielfeld darzustellen. Das Table Element, was eine Tabelle ist, besteht aus mehreren Zeilen und Spalten. Diese wiederum beinhaltenen Zellen (TD Elemente) mit individuellen IDs, welche mit Bildern von Objekten, Hindernissen oder einem Bild der Spielfigur befüllt werden können. Die ID der Zellen beinhaltet einen Bezeichner „cell“ und zudem noch zwei aufeinanderfolgende Zahlen, welche für Zeile und Spalte stehen. Zudem ist jede Zelle einer Klasse zugeordnet, um eine bestimmtes Hintergrundbild fürs Spielfeld zu liefern.

CELL00 CLASS = GRASS	CELL10 CLASS = GRASS	CELL20 CLASS = GRASS
CELL01 CLASS = GRASS	CELL11 CLASS = SAND	CELL21 CLASS = GRASS
CELL02 CLASS = GRASS	CELL12 CLASS = SAND	CELL22 CLASS = GRASS

Tabelle 3: Beispiel Spielfeld

Legende: Spalte, Zeile

Die Bewegungen der Spielfigur von einer Zelle in eine andere erfolgt mithilfe JavaScript und jQuery indem das DOM (Document Object Model) manipuliert wird. Dabei wird das Model (Game und Bot) nach einer Manipulation (zum Beispiel die Änderung eines Eintrags im mapArray) in der View mithilfe der update-Methode aktualisiert.

4.5.2 Update der View

Die update-Methode der View greift auf das „mapArray“ (siehe Abbildung 12) zu und geht das Array Reihe für Reihe durch und erstellt in einer Tabelle für jede Reihe des Arrays ein „TR-Element“ (table row element). Anschließend wird für jeden Eintrag in der Reihe des Arrays ein „TD-Element“ (table data element) in der jeweiligen Tabellen-Reihe erzeugt und je nach Wert des mapArrays an der jeweiligen Stelle mit den gewünschten Bildern und einer ID ausgestattet.

```
1  // classes of the cells, which are needed for the background-image of
2  the cell
   var cellClasses = ["decorationBackground", "walkableArea", "start",
3  "finish", "obstacle", "fruitField"];
4
   //images and class of the obstacles
5  var obstacleImages = ["pictures/mountain.png", "pictures/bush.png"];
6  var obstacleClass = ["obstacle0", "obstacle1"];
7
   //images and ids of the collectible Items
8  var collectibleItemImg = ["pictures/apple.png", "pictures/banana.png",
   "pictures/orange.png", "pictures/pear.png", "pictures/strawberry.png",
9  "pictures/watermelon.png"];
10 var collectibleItemId = "collectible";
11
   //images and the id of the bot (from left to down)
12 var botImages = ["pictures/farmerLeft.png", "pictures/farmerUp.png",
   "pictures/farmerRight.png", "pictures/farmerDown.png"];
13 var botId = "player";
14
15
   //creates and updates the table (map of the game)
16 this.update = function (mapArr) {
       $("#tbody").empty();
17     mapArr.forEach(function (subArr, rowIndex) {
18         var row = document.createElement("tr");
           tbody.appendChild(row);
19         subArr.forEach(function (number, colIndex) {
20             var cell = document.createElement("td");
               row.appendChild(cell);
21             cell.className = number;
               cell.id = "cell" + colIndex + rowIndex;
```

```
22              if (number === 0) {
23                  cell.className = cellClasses[0];
24              } else if (number === 1) {
25                  cell.className = cellClasses[1];
26              } else if (number === 2) {
27                  cell.className = cellClasses[4];
28                  var obstacleImage = document.createElement("img");
29                  obstacleImage.src = obstacleImages[0];
30                  obstacleImage.className = obstacleClass[0];
31                  cell.appendChild(obstacleImage);
32              } else if (number === 3) {
33                  cell.className = cellClasses[4];
34                  var obstacleImg = document.createElement("img");
35                  obstacleImg.src = obstacleImages[1];
36                  obstacleImg.className = obstacleClass[1];
37                  cell.appendChild(obstacleImg);
38              }
39              else if (number === 4) {
40                  cell.className = cellClasses[2];
41                  var botImg = document.createElement("img");
42                  if (bot.tempFaceDir === "left") {
43                      botImg.src = botImages[0];
44                  } else if (bot.tempFaceDir === "up") {
45                      botImg.src = botImages[1];
46                  } else if (bot.tempFaceDir === "right") {
47                      botImg.src = botImages[2];
48                  } else if (bot.tempFaceDir === "down") {
49                      botImg.src = botImages[3];
50                  }
51                  botImg.id = botId;
52                  cell.appendChild(botImg);
53              } else if (number === 5) {
54                  cell.className = cellClasses[3];
55              } else if (number === 6) {
56                  cell.className = cellClasses[1];
57                  var itemImg = document.createElement("img");
58                  itemImg.id = collectibleItemId;
59                  itemImg.src = collectibleItemImg[0];
60                  cell.appendChild(itemImg);
61              } else if (number === 7) {
62                  cell.className = cellClasses[5];
63              }
64          });
65      });
66      document.getElementById("map").appendChild(table);
67  };
68
69
70
```

Abbildung 12: Update des Spielfeldes (View-Methode)

Des Weiteren beinhaltet die View eine updateBot-Methode, welche die ausgewählten Aktionen (Einträge im actions-Array) des Bots alle 500 Millisekunden aktualisiert und somit als Bewegungsverlauf in der View darstellt, eine updateInfo-Methode, um die Anzeige der eingesammelten Objekte zu aktualisieren und eine updateElements-Methode, um je nach Level bestimmte Elemente ein- und auszublenden (Beispiel: Abbildung 8, showLoopInfo & showCheckField, welche in höheren Level benötigt werden).

4.6 Controller

4.6.1 Schleifen nutzen

Abbildung 13: Screenshot Go!Farmer (Level 21)

Der Controller ist dafür zuständig auf User Input zu reagieren und um das Model danach zu manipulieren. In der graphischen Darstellung des Spiels (Abbildung 13) sieht man in der Aktionsauswahl die einzelnen anklickbaren Aktionen (gehen, links drehen, rechts drehen, Apfel aufheben, Feld überprüfen, neue Frucht aufheben und Auswahl löschen), welche mit einem actionListener (ein Eventlistener der prüft, ob ein Element angeklickt wurde) ausgestattet sind. Nachdem der User die einzelnen Teilaktionen ausgewählt hat, werde diese in das „actions"-Array des Bots geladen und danach nach dem Anklicken des Startbuttons sequentiell nach einander ausgeführt.

```
1  /*
2   goes through the actionsContainer and creates for every loopThroug a new element
   with the same id at the end
3   of the actionsContainer
4   */
   this.actionLoops = function () {
5       var loopTroughs = game.loopCounter - 1;
6       var length = document.getElementById("actionsContainer").children.length *
   loopTroughs;
7       for (var i = 0; i < length; i++) {
8           var newElem = document.createElement("A");
9           newElem.id = document.getElementById("actionsContainer").children[i].id;
            document.getElementById("actionsContainer").appendChild(newElem);
10      }
   };
```

Abbildung 14: Wiederholungen im Spiel (Controller-Methode)

Außerdem beinhaltet der Controller die actionLoops-Methode, diese Methode dient dazu Wiederholungen der ausgewählten Aktionen in Go!Farmer nutzen zu können. Dabei greift diese Methode auf den actionsContainer (Container mit den ausgewählten Aktionen) zu und kopiert für jede ausgewählte Aktion eine Kopie am Ende des actionsContainer hinzu. Je nach Anzahl der Wiederholungen wird dieser Vorgang wiederholt. Die Abbildung 13 zeigt eine Auswahl von vier Wiederholungen, heißt die ausgewählten Aktionen aus dem actionsContainer werden vier Mal hintereinander ausgeführt.

4.6.2 Schleifen mit Bedingungen

Abbildung 15: Screenshot Go!Farmer (Level22)

In der Abbildung 15 kann man erkennen, dass man Schleifen auch mit Bedingungen benutzen kann. In diesem Fall ist die Bedingung (gehen bis ein Hindernis kommt). In der Aktionsauswahl befindet sich „nach links drehen“ und „bis zum nächsten Hindernis gehen“. Wenn man diese Aktionen mit den ausgewählten Wiederholungen (2 Wiederholungen) kombiniert, dann dreht sich der Bot nach links und geht bis zum nächsten Hindernis in der ersten Wiederholung und in der zweiten führt er die ausgewählte Aktion noch einmal aus.

```
else if (document.getElementById("actionsContainer").children[j].id
=== "moveTillObstacle") {

//a move action is added into the actions-array, while the bot can
move forward
    while (tempMap[tempYpos][tempXpos] === START_POSITION ||
tempMap[tempYpos][tempXpos] === WALKABLE_AREA) {
        if (bot.tempFaceDir === "left") {
            bot.actions.push(0);
            tempXpos--;
            lastAction = 0;
        } else if (bot.tempFaceDir === "right") {
            bot.actions.push(1);
            tempXpos++;
            lastAction = 1;
        } else if (bot.tempFaceDir === "up") {
            bot.actions.push(2);
            tempYpos--;
            lastAction = 2;
        } else if (bot.tempFaceDir === "down") {
            bot.actions.push(3);
            tempYpos++;
            lastAction = 3;
        }
    }
```

Abbildung 16: Auswahl der „gehen bis zum Hindernis“-Bedingung

Der Controller geht in der readActionsContainer-Methode die Aktionsauswahl (actionsContainer) durch und prüft, ob ein Element mit der ID „moveTillObstacle“ hinzugefügt wurde, falls dies der Fall ist, wird in einem temporären Array (Kopie des mapArrays) geprüft wie viele Felder der Bot sich in die jeweilige Blickrichtung bewegen und dem entsprechend die Bewegungsaktionen in das actions-Array des Bots eingetragen.

4.6.3 Prozedur erstellen

Abbildung 17: Screenshot Go!Farmer (Level 28)

Im Prozeduren-Tab (Abbildung 17) kann eine Prozedur (Unterprogramm) erstellt werden, welche dann im Hauptprogramm aufgerufen werden kann.

```
1  // creates a procedure and adds it in the actionsContainer at the
2  index of the "proc1" element
3  } else if
   (document.getElementById("actionsContainer").children[j].id ===
4  "proc1") {
5      // creates a temporary procedure for every procedure which is
   added to the actionsContainer
6      var tempProc = document.createElement("A");
7      tempProc.id = "tempProc";
8
8      var proc1 = document.getElementById("proc1");
9      proc1.replaceWith(tempProc);
10
10     // adds the elements of the procedureContainer to the temporary
   procedure
11     var length =
12 document.getElementById("procedureContainer").children.length;
13     if(length > 0) {
14         var elem = document.createElement("A");
15         elem.id =
   document.getElementById("procedureContainer").children[0].id;
16         document.getElementById("tempProc").appendChild(elem);
17     }
18     for (var k = 0; k < length; k++) {
           var newElem = document.createElement("A");
           newElem.id =
```

```
19  document.getElementById("procedureContainer").children[k].id;
20          document.getElementById("tempProc").appendChild(newElem);
    }
21
22  // replaces the temporary procedure with it's children (the
actions)
24  var element = document.getElementById("tempProc");
24  var fragment = document.createDocumentFragment();
    while(element.firstChild) {
25      fragment.appendChild(element.firstChild);
26  }
    element.parentNode.replaceChild(fragment, element);
}
```

Abbildung 18: Prozedur erstellen

Der Controller geht in der readActionsContainer-Methode die Aktionsauswahl (actionsContainer) durch und prüft, ob ein Element mit der ID „proc1“ hinzugefügt wurde, falls dies der Fall ist, wird das Element mit der ID „proc1“ durch ein Element mit der ID „tempProc“ ersetzt und anschließend werden alle Kind-Elemente des Prozedur-Tabs (Element mit der ID „procedureContainer“) kopiert und an das Element mit der ID „tempProc“ gehängt. Darauffolgend wird das Element mit der ID „tempProc“ mit dessen Kind-Elementen ersetzt. Somit werden die einzelnen Aktionen an der Position eingefügt, wo das Element mit der ID „proc1“ im actionsContainer hinzugefügt wurde (die Prozedur ausgewählt wurde, siehe Abbildung 17).

Kapitel 5 – Evaluation und Test

Dieses Kapitel beinhaltet einen Usability-Test und eine Evaluation in Form von einer Umfrage innerhalb der Zielgruppe. Sowohl der Test als auch die Evaluation wurden anonym innerhalb der Johannes-Gutenberg-Schule in Bargteheide (vierte Klasse) durchgeführt.

5.1 Usability-Test

5.1.1 Definition

Usability:

„Usability bezeichnet das Ausmaß, in dem ein Produkt, System oder Dienst durch bestimmte Benutzer in einem bestimmten Anwendungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen.“ [Usa17]

User Experience:

„User Experience erweitert den Begriff Usability um ästhetische und emotionale Faktoren wie eine ansprechende, „begehrenswerte” Gestaltung, Aspekte der Vertrauensbildung oder Spaß bei der Nutzung (Joy of use).“ [Usa17]

5.1.2 Durchführung

Es wurde einer klassischer (Szenario-basierter) Nutzertest durchgeführt, um die Gebrauchstauglichkeit der Anwendung zu bestimmten. Dabei haben die Nutzer aus der spezifischen Zielgruppe den Prototypen während der Implementations-Phase auf Schwachstellen und Verbesserungsmöglichkeiten getestet (induktiver Test). Der Test beinhaltet einzelne Teilaufgaben, welche innerhalb bestimmter Zeit von den Probanden erfüllt werden mussten. Da der Test in einer vierten Grundschulklasse mit 22 Schülern stattfand, wo es kein Usability-Labor gab, musste auf den Einsatz von Eye-Tracking verzichtet werden. Stattdessen wurde der Prototyp mithilfe eines MacBooks benutzt und die Ergebnisse auf einem Notizblock notiert, um diese im späteren Verlauf auszuwerten.

5.2 Test-Ergebnisse

Bei der Durchführung der Tests wurde bei der sprachlichen Formulierung der Teilaufgaben Rücksicht auf die Zielgruppe (vierte Schulklasse) genommen. Zudem beziehen sich die Testergebnisse auf den Prototypen und nicht auf das Endprodukt.

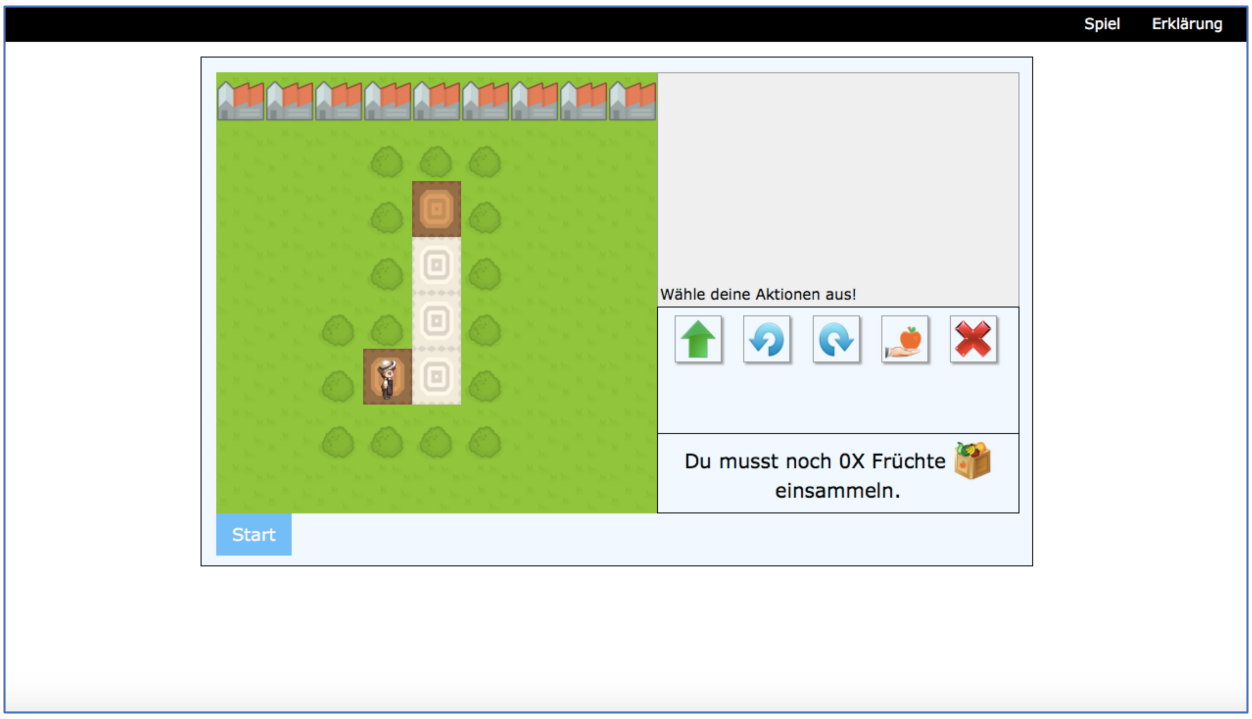


Abbildung 19: Screenshot (Prototyp)

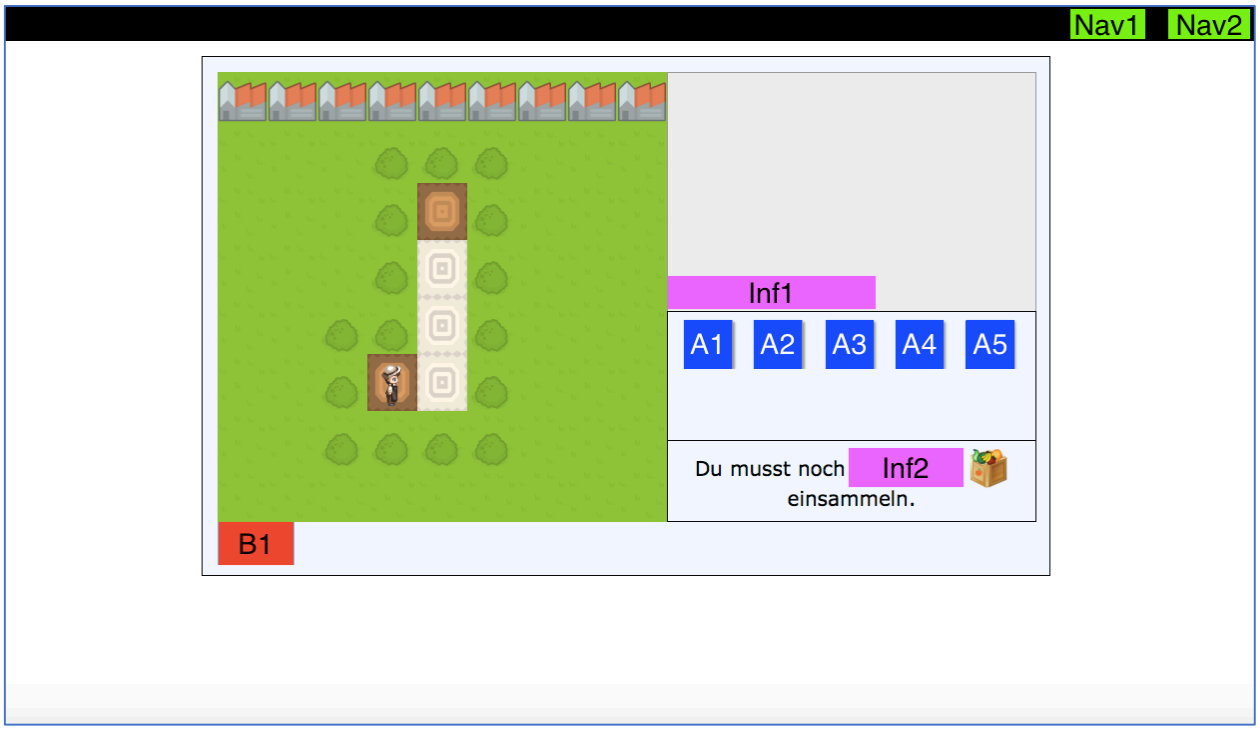


Abbildung 20: Screenshot (Prototyp) mit Zuweisung

Legende:

A = Aktion

B = Button

Inf = Information

Nav = Navigation

5.2.1 Intuitive Nutzung der Webanwendung

Sind die unterschiedlichen Navigationselemente klar erkennbar dargestellt und positioniert?
Vorgegebene Zeit: 5 Sekunden

Abbildung 21: Usability-Test Aufgabe 1

Test-Aufgabe 1.1: „Klicke auf die Erklärung des Spieles.“ (Nav2)
Mittelwert: 4,95 Sekunden

Test-Aufgabe 1.2: „Klicke auf das Spiel (zurück zum Spiel)“ (Nav1)
Mittelwert: 3,27 Sekunden

Test-Ergebnis: Sowohl der Mittelwert der Test-Aufgabe 1.1 mit 4,95 Sekunden als auch der Test-Aufgabe 1.2 mit 3,27 Sekunden sind im vorgegeben Zeitraum (5 Sekunden). Somit sind die unterschiedlichen Navigationselemente klar erkennbar dargestellt und positioniert und aufgrund dessen auch intuitiv bedienbar.

Sind die Buttons, semantisch korrekt und verständlich bezeichnet?

Vorgegebene Zeit: 5 Sekunden

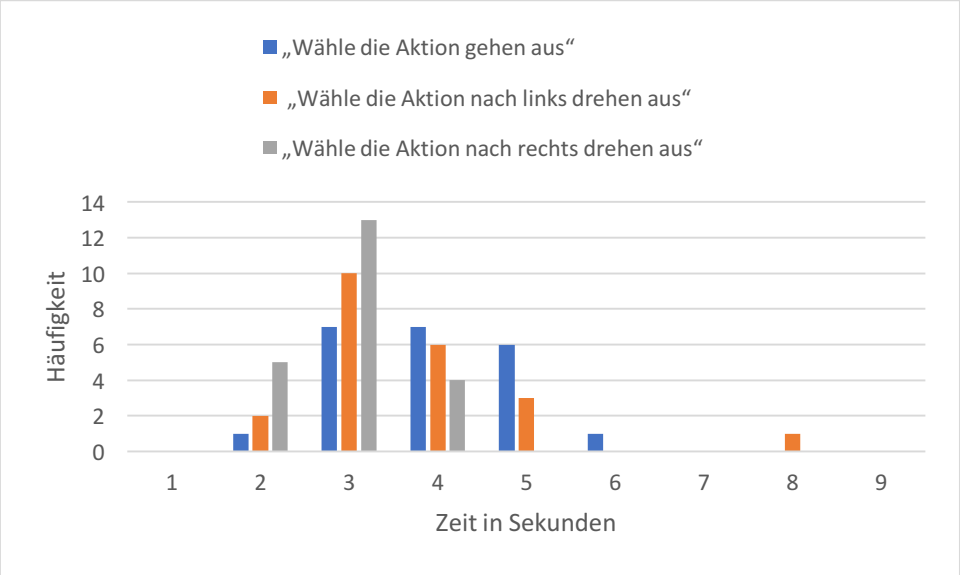


Abbildung 22: Usability Aufgabe 2.1, 2,2 & 2.3

Test-Aufgabe 2.1: Wähle die Aktion „gehen“ aus (A1)

Mittelwert: 3,95 Sekunden

Test-Aufgabe 2.2: Wähle die Aktion „nach links drehen“ aus (A2)

Mittelwert: 3,68 Sekunden

Test-Aufgabe 2.3: Wähle die Aktion „nach rechts drehen“ aus (A3)

Mittelwert: 2,95 Sekunden

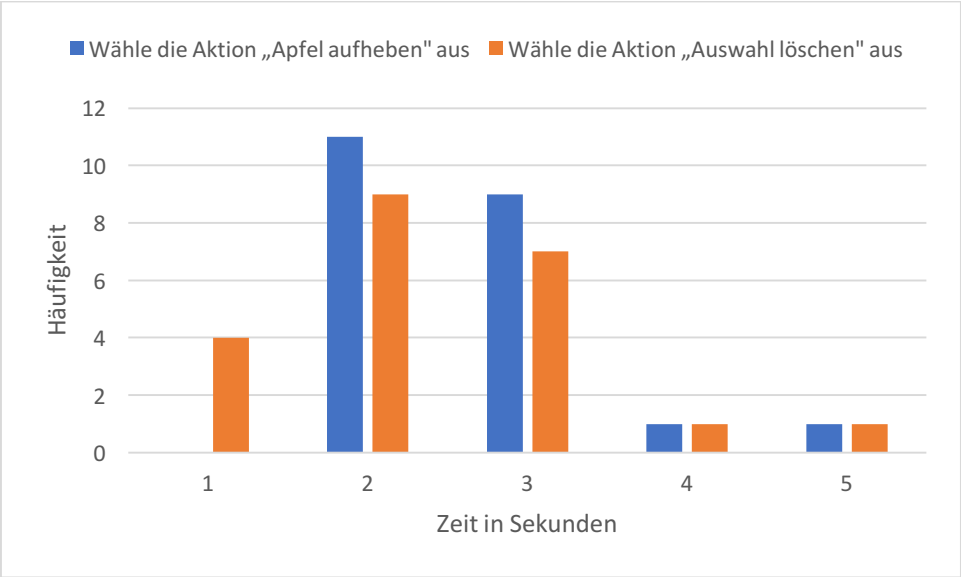


Abbildung 23: Usability Test-Aufgabe 2.4 & 2.5

Test-Aufgabe 2.4: Wähle die Aktion „Apfel aufheben“ aus (A4)

Mittelwert: 2,63 Sekunden

Test-Aufgabe 2.5: Wähle die Aktion „Auswahl löschen“ aus (A5)

Mittelwert: 2,36 Sekunden

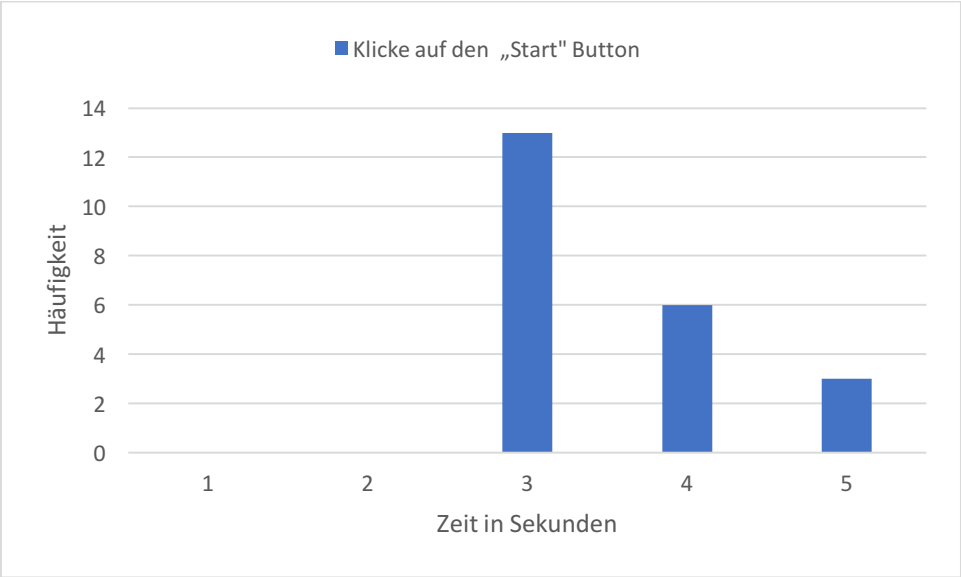


Abbildung 24: Usability Test-Aufgabe 2.6

Test-Aufgabe 2.6: Klicke auf den „Start“ Button (B1)
Mittelwert: 3,55 Sekunden

Test-Ergebnis: Alle Test-Aufgaben von 2.1 bis 2.5, welche sich auf die Aktionen (A1-A5) beziehen als auch die Test-Aufgabe 2.6, welche sich auf den Startbutton (B1) bezieht, wurden innerhalb kürzester Zeit erledigt. Die Mittelwerte der einzelnen Aktionen weisen auch darauf hin, dass mit steigender Nutzung der Aktionen die benötigte Zeit sinkt. Der Grund dafür ist, dass ähnliche Elemente (die Aktionen A1 bis A5) in der Webanwendung nebeneinander positioniert sind und da der Proband nach der Nutzung der ersten Aktion den Bereich der Aktions-Elemente auf einen kleineren Teilbereich reduzieren kann, statt die ganze Seite zu durchsuchen. Aufgrund dessen sinkt die Zeit von der ersten Aktion (A1) mit einem Mittelwert von 3,95 Sekunden bis zur letzten Aktion (A5) auf einen Mittelwert von 2,36 Sekunden. Somit waren alle Mittelwert-Zeiten unter der vorgegebenen Zeit von 5 Sekunden, was darauf hinweist, dass die Buttons semantisch korrekt und verständlich bezeichnet sind.

Sind Informationen an aufmerksamkeitsstarken Stellen platziert?

Vorgegebene Zeit: 5 Sekunden

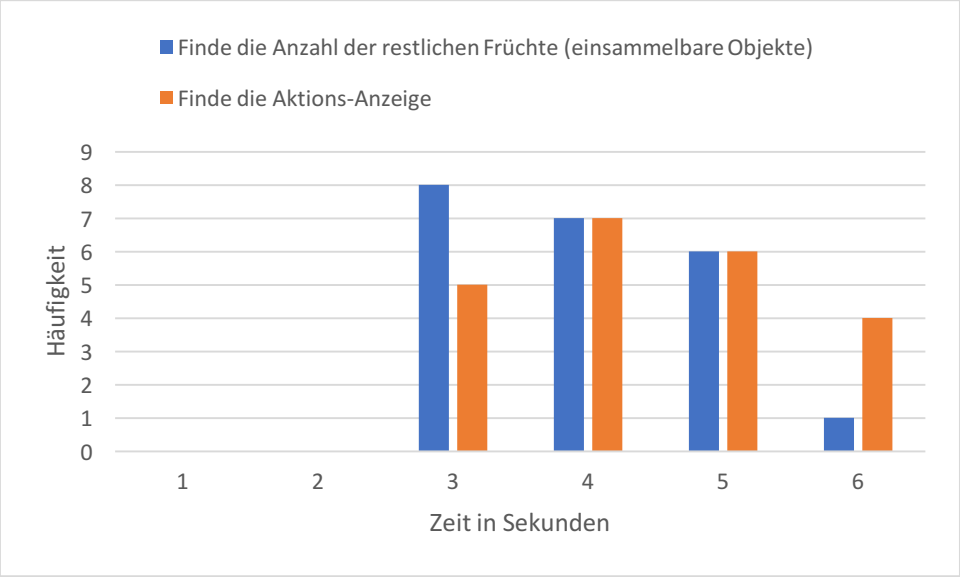


Abbildung 25: Usability Test-Aufgabe 3.1, 3.2 & 3.3

Test-Aufgabe 3.1: Finde die Anzahl der restlichen Früchte (Inf2)

Mittelwert: 4,00 Sekunden

Test-Aufgabe 3.2: Finde die Aktions-Anzeige (Inf1)

Mittelwert: 4,41 Sekunden

Test-Ergebnis: Die Test-Aufgabe 3.1 und 3.2 wurden ebenfalls innerhalb der vorgegeben Zeit von 5 Sekunden erledigt und somit sind die Informationen Inf1 & Inf2 an aufmerksamkeitsstarken Stellen platziert.

5.3 Evaluations-Ergebnisse

Die Formulierung der Fragen wurde an die Zielgruppe (vierte Schulklasse) angepasst.

Frage 1: „Spielst du gerne?“

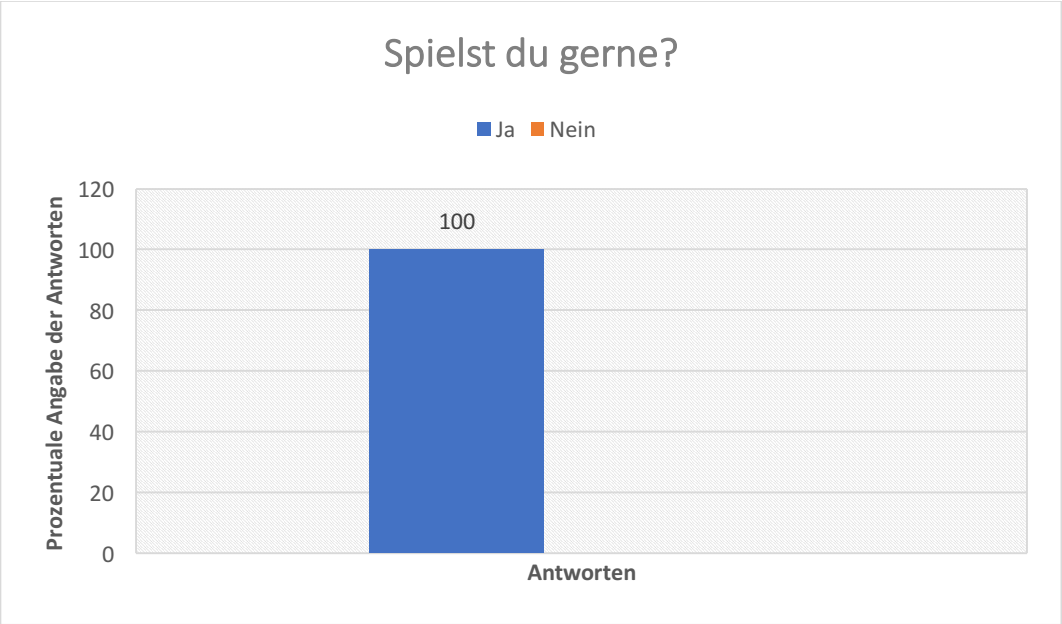


Abbildung 26: Evaluation Frage 1

Frage 2: „Wie lernst du zu Hause?“ (normalerweise)

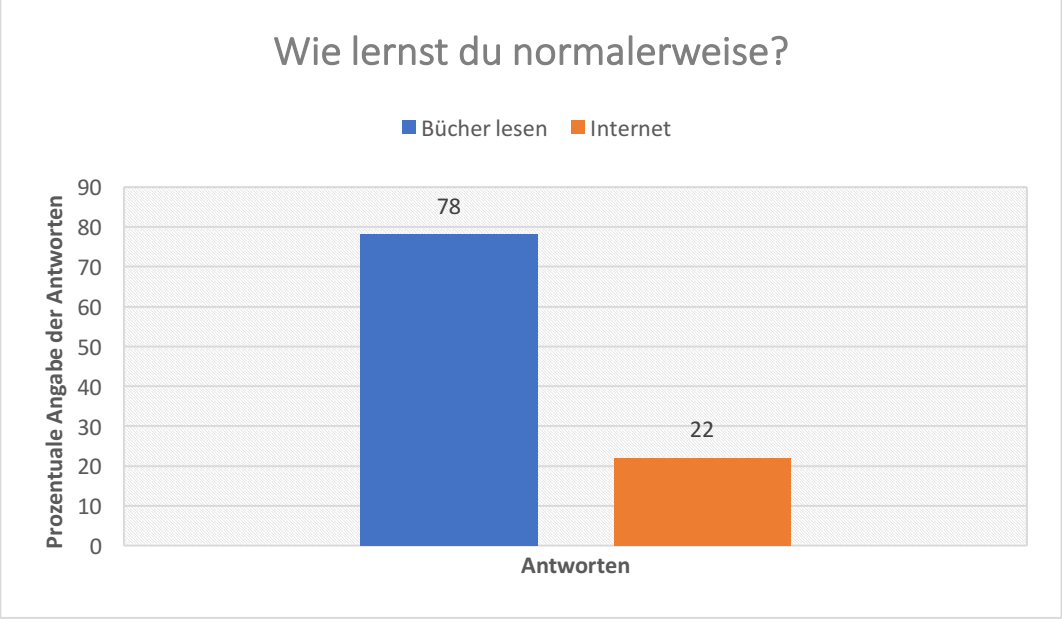


Abbildung 27: Evaluation Frage 2

Frage 3: „Findest du das Spiel übersichtlich?“

Frage 4: „Konntest du das Spiel einfach benutzen (bedienen)?“

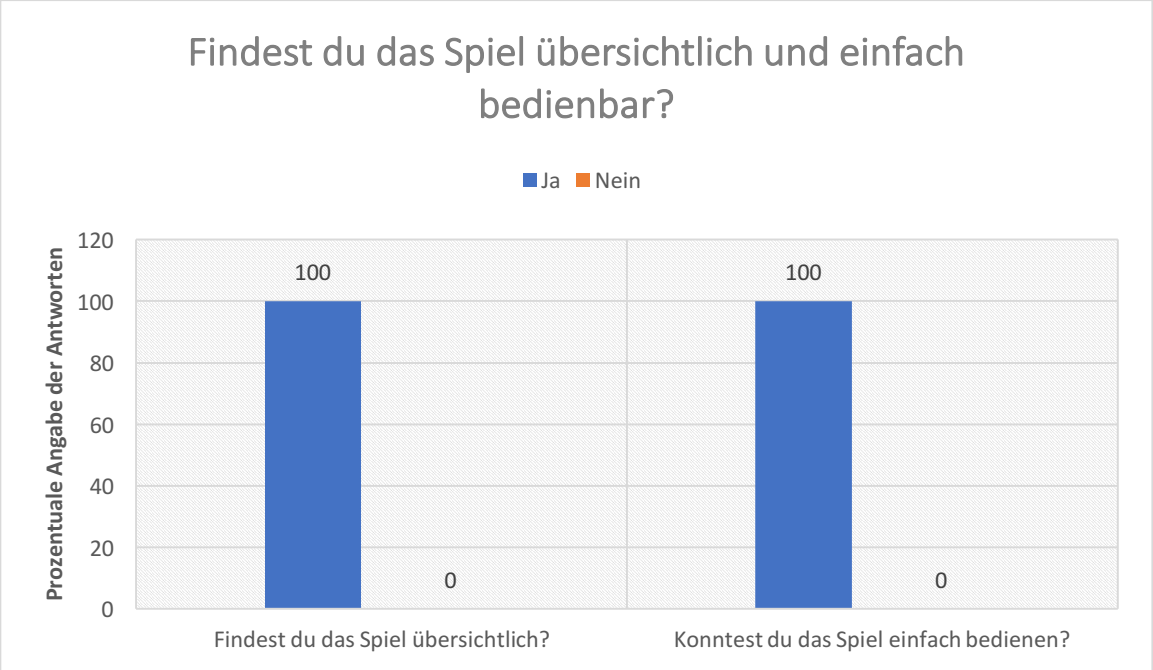


Abbildung 28: Evaluation Frage 3 & 4

Frage 5: „Findest du das Spiel schwierig?“

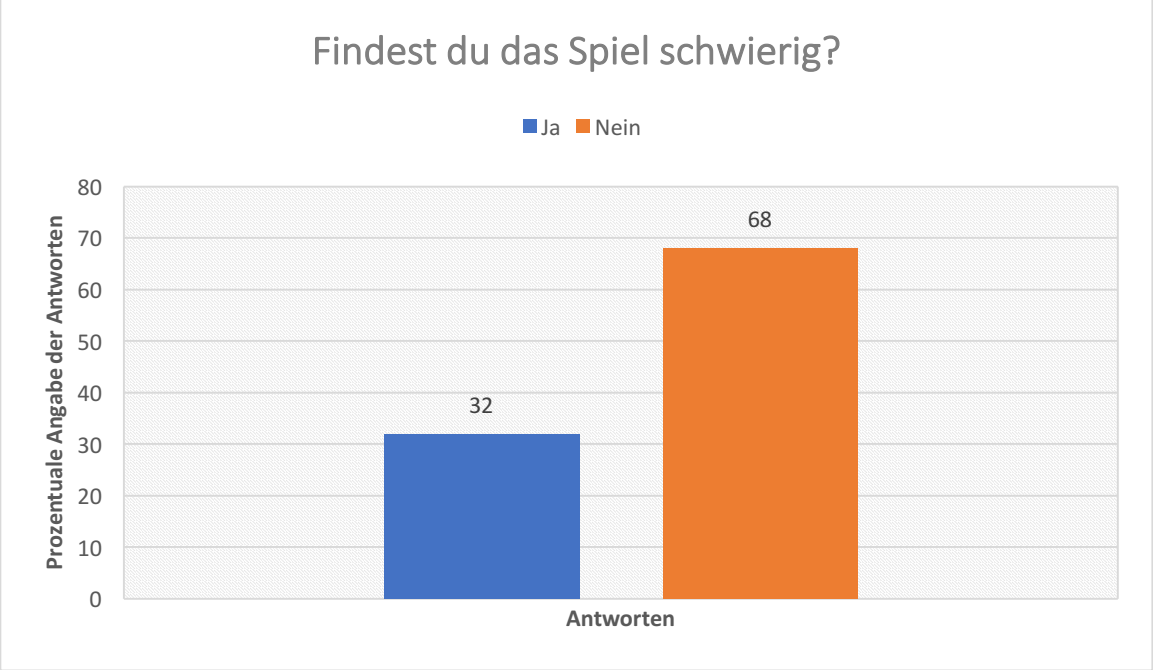


Abbildung 29: Evaluation Frage 5

Frage 6: „Hat dir das Spiel Spaß gemacht?“

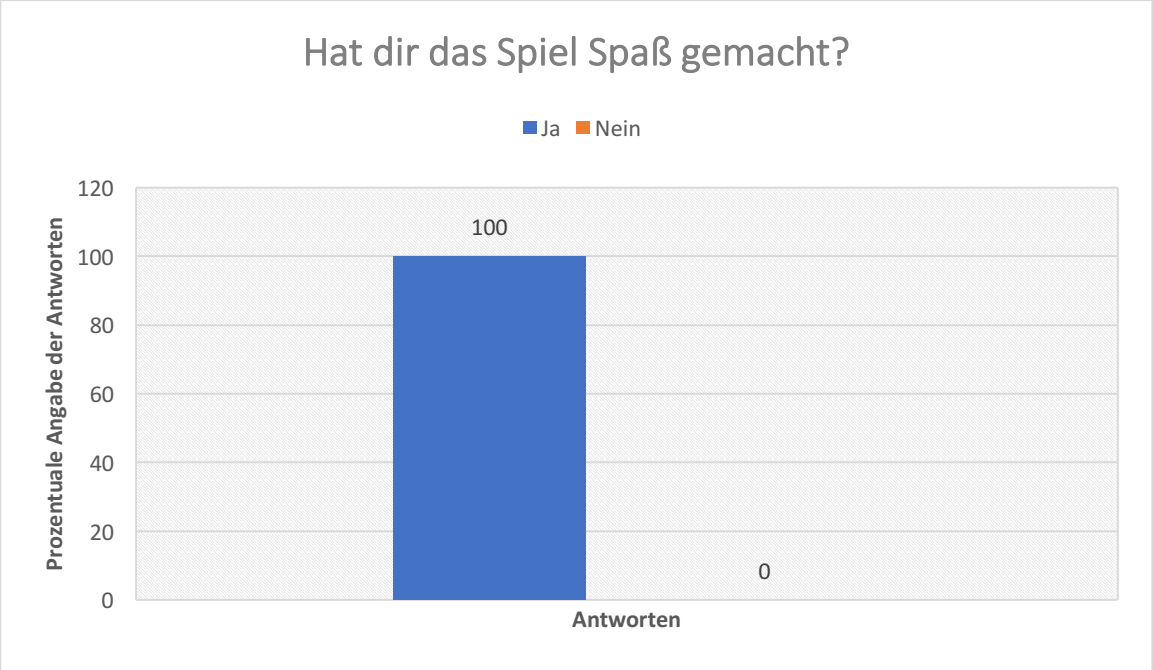


Abbildung 30: Evaluation Frage 6

5.4 Gesamtergebnis

Sowohl die Ergebnisse aus dem Usability-Test als auch die Evaluation sind positiv ausgefallen. Der Usability-Test beweist aufgrund der positiven Ergebnisse, dass die Anwendung intuitiv bedienbar ist. Ein weiteres Indiz für die intuitive Bedienung der Anwendung und ein positives „Look & Feel“-Erlebnis liefern die Ergebnisse der Frage 3 und 4 aus der Evaluation. Mit einer einstimmigen Meinung der Probanden ist der Screen- & Content-Design als übersichtlich und einfach bedienbar empfunden worden. Die Frage 1 bestätigt, dass es den Schülern Spaß macht etwas zu spielen und somit ein allgemeines Interesse fürs „spielerische Lernen“ vorhanden ist. Die Frage 5 „Findest du das Spiel schwierig?“ zeigt, dass eine gewisse Herausforderung in den einzelnen Leveln vorhanden ist und es somit nicht als zu einfach angesehen wird. Den wichtigsten Beweis liefert die Frage 6 aus der Evaluation, in welcher die Probanden die Frage: „Hat dir das Spiel Spaß gemacht?“ zu 100 Prozent mit „Ja“ beantwortet haben. In Bezug auf die User Experience kann somit bestätigt werden, dass das „Joy of Use“ vorhanden ist, da es den Probanden aus der Zielgruppe Spaß gemacht hat die Webanwendung zu nutzen.

Kapitel 6 – Fazit und Ausblick

Als Fazit dieser Bachelorarbeit kann aufgrund der positiven Ergebnisse der Tests und der Umfrage gesagt werden, dass das Lernen auf diese Art bei der Zielgruppe als positiv angesehen wird. Es hat den Probanden bei den Tests nicht nur Spaß gemacht Probleme spielerisch zu lösen, sondern auch ohne den Betreuer zu fragen, selbstständig mögliche Lösungswege durchzugehen. In Bezug auf das problemorientierte „Superlearning“ (selbständiges Lernen in einem Zustand von tiefer Entspannung, um beide Gehirnhälften zu beanspruchen) hat sich ebenfalls gezeigt, dass die Probanden eine emotionale Bindung zum Spiel hatten und statt verkrampft vor einem Buch zu sitzen, um den Inhalt zu lernen, sie entspannt spielerisch gelernt haben. Des Weiteren war der Spaßfaktor dafür ausschlaggebend, dass die Probanden bei einem Erfolg (Levelabschluss) gejubelt haben und bei einem Misserfolg dazu geführt hat, dass die Probanden aus ihren Fehlern lernen wollten, um das nächste Erfolgserlebnis zu haben, statt frustriert zu sein. Das spielerische Lernen hat sich auch als sehr Vorteilhaft in Bezug auf das langfristige Abspeichern von Informationen erwiesen, da die Wahrscheinlicht einer langfristigen Abspeicherung steigt, wenn eine emotionale Bewertung vorhanden ist [Mft17, Seite: 9]. Als Ausblick ist es wünschenswert, dass die in dieser Bachelorarbeit erstellte Webanwendung im Schulunterricht zum Einsatz kommt. Beispielsweise könnte die Webanwendung im Matheunterricht eingesetzt werden, um den Schüler den Zusammenhang zwischen der Mathematik und dem Computational Thinking zu verdeutlichen. Außerdem verfügen einige Grundschulen schon über einen Computerraum, in welchem den Schülern beigebracht wird, wie man mit einem Computer eine Text-Datei erstellt oder wie man einen Browser nutzt. Dieser Computerraum könnte auch dazu eingesetzt die Webanwendung zu nutzen, um den Kindern eine Einführung in das Computational Thinking zu ermöglichen. Des weiter könnte der Einsatz solcher Anwendungen im Schulunterricht dazu führen, dass das Interesse der Schüler geweckt wird und diese nicht nur im Schulunterricht das Computational Thinking nachvollziehen wollen, sondern auch privat, was ein großer Schritt in Bezug auf die informatische Bildung der Schüler wäre.

Literaturverzeichnis

[App17] „Lightbot Statistics“ - https://www.appbrain.com/dev/Lightbot/ (Version: Juli 2017)

[Com17] „What is Computational Thinking?“ - https://computationalthinkingcourse.withgoogle.com/unit?lesson=8&unit=1 (Version: Mai 2017)

[Fit17] Alexander Repenning - „Computational Thinking in der Lehrerbildung“ http://www.fit-in-it.ch/sites/default/files/downloads/schrift_repenning-1411-gzd_deutsch_0.pdf (Version: Mai 2017)

[Hui04] Johan Huizinga – „Homo Ludens: Vom Ursprung der Kultur im Spiel“ 2004 Verlag: Rowohlt

[Inf17] „Model-View-Controller“ - https://informatik.bildungrp.de/fileadmin/user_upload/informatik.bildung-rp.de/Fortbildung/html/D1-MVCNotenstatistik/mvc0_0.html (Version: Juni 2017)

[Lex17] „Definition: Lernen“ - http://lexikon.stangl.eu/551/lernen/ (Version: April 2017)

[Lig17a] Danny Yaroslavski - „How does Lightbot teach programming?“ https://lightbot.com/Lightbot_HowDoesLightbotTeachProgramming.pdf (Version: Juli 2017)

[Lig17b] „Lightbot Flashanwendung“ - https://lightbot.com/flash.html (Version: Mai 2017)

[Mft17] Henning Scheich - „Wie lernt der Mensch?“
http://www.mft-online.de/files/12.25_fr_omft2013_scheich.pdf
(Version: Mai 2017)

[Oer96] Rolf Oerter – „Spielendes Lernen, gibt es das?“ In: Praxis Schule, H. 4

[Uni17] „Definition: Computational Thinking“ -
https://kw.uni-paderborn.de/institut-fuer-erziehungswissenschaft/arbeitsbereiche/schulpaedagogik/forschung/forschungsprojekte/computational-thinking/
(Version: Mai 2017)

[Usa17] „Definition: Usability und User Experience“ -
https://www.usability.de/usability-user-experience.html (Version Juni 2017)

[Wik17] „Allgemeine Information zu Lightbot“ - https://en.wikipedia.org/wiki/Lightbot
(Version: Mai 2017)