Phasen, Techniken und Strategien der Softwareentwicklung an Hand eines Kursprojekts im Modul SWE-II*

*Thema Vorgabe laut Checkliste zum Kursprojekt im Modul Software-Engineering-II

Fabian Schiemann
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_schiemann21@stud.hwr-berlin.de

Oskar Hugo Thaute
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_thaute21@stud.hwr-berlin.de

Sharleen Schnelle
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_schnelle21@stud.hwr-berlin.de

Noah Reckhard
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_reckhard21@stud.hwr-berlin.de

Laurin Pausch
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_pausch21@stud.hwr-berlin.de

Minh Hai Nguyen
Student at Computer Science Dept
Faculty of Cooperative Studies
Berlin School of Economics and Law, Germany
s_nguyenmi21@stud.hwr-berlin.de

***Abstract*—Die Theorie der Softwareentwicklung bietet zahlreiche strukturierte Informationen und Vorstellungen von Form und Struktur eines Softwareprojektes. Die Realität innerhalb eines solchen Projektes kann davon jedoch stark variieren. Sei es der Vorgesetzte in einem alteingesessenen Familienunternehmen, der Scrum-Sprints verplant, oder der rüstige Teamleiter einer großen Abteilung, der alle seine Projekte in PASCAL geschrieben haben möchte. Dieser Artikel präsentiert eine solche Variation und zeigt, wie sich eine Gruppe von sechs Studenten mit hohen Ambitionen und frisch erlangtem Wissen einem solchen Softwareprojekt widmet. Generationsbedingt wird im Folgenden die Entwicklung eines Computerspiels mit dem Namen "Harsh Politics" vorgestellt. Ziel dieses Artikels ist die Darlegung und Erläuterung spezieller Methoden und Strategien aus der Softwareentwicklung, anhand der Entwicklung eines Spieles.**

I. Hinweise

Aus Gründen der besseren Lesbarkeit wird im Text verallgemeinernd die männliche Form verwendet. Diese Formulierungen umfassen gleichermaßen weibliche und männliche Personen.

II. Einleitung / Motivation / Einführung / Hintergründe

Grady Booch gibt in seinem Paper "The History of Software Engineering" eine gute Übersicht über die Entwicklung des Begriffs "Software Engineering". Im folgenden Absatz ist das Paper zusammengefasst. Laut Booch wurde der Begriff erstmals 1963/64 von Magaret Hamilton erwähnt. Als damalige Lead-Developerin in Skylab und Apollo wollte sie ihre Arbeit von "Hardware Engineers" unterscheiden. In den 1960er soll es drei große Faktoren gegeben haben, die die Softwareentwicklung beeinflussten, und zwar die Kommerzialisierung von Software als Produkt, die Entwicklung des SAGE-Verteidigungssystems und das US-Raumprogramm und die daraus resultierenden überlebenswichtigen Softwaressyteme. Zwischen den 60er und 80er wurden die Konzepte wie Modular Programming, Wasserfall Modell und Entity-Relationship-Modell für die Softwareentwicklung entwickelt. In den 80er und 90er veränderten sich die Ansprüche auf Softwaresysteme durch die zunehmende Globalisierung. Daher mussten neue Konzepte wie UML (Unified-Model-Language), Spiral Modell, OOP (objekt-orientierte Programmierung) eingeführt werden, um die wachsenden Anforderungen zu bewerkstelligen. Durch das Internet in den 90er erhielt die Rolle der Software abermals eine neue Stellung. Der "Consumer" hatte nun als neuer Stakeholder andere Ansprüche an die Software als Unternehmen. Zu dieser Zeit haben sich bereits Techniken der Softwareentwicklung etabliert, wie "ContinuousIntegration" und iteratives Entwickeln. Hinzu kamen auch Design Pattern auf Architektur und Code-Ebene. Das Internet-of-Things gewann durch Smartphones an Bedeutung, sodass

monolitische Software durch Microservices, Webtechnologien und Cloud Computing ersetzt wurden. [1] Wie zu sehen ist, hat das Wort "Software Engineering" eine lange Entwicklungsgeschichte. Sie zeigt, dass durch die wachsenden Anforderungen und die wachsende Komplexität, Software gut organisiert, gemanagt und durch geeignete Richtlinien, wie Patterns, Agiles Programmieren und gute Dokumentation entwickelt werden muss. Die moderne Softwareentwicklung baut auf fünf Phasen auf. Die Analyse, das Design, die Codierung, das Testen und die Wartung. [2] Jeder dieser fünf Phasen nutzt Techniken und Strategien, die in den letzten 60 Jahren entwickelt wurden. In der Analyse gibt es Stakeholderlisten und Anforderungsanalyse, aus der sich funktionale und nicht funktionale Anforderungen ableiten lassen. Aus diesen lassen sich das Design entwickeln, in dem man aus den Anforderungen UML-Diagramme und Softwarearchitekturen entwickelt. Hinzu kommen Grundsätze und Leitsätze, die die Gestaltung von Bedienoberflächen und Dialogen festlegen. Dadurch lassen sich den Stakeholderlisten nahe Software entwickeln. In der Implementierungsphase haben sich bestimmte Frameworks, und Code-Richtlinien, wie YAGNI, DRY, KISS, SOLID durchgesetzt, um eine hohe Qualität des Codes zu gewährleisten. [3]–[5] Genauso ist das Testen und Warten von Software sehr wichtig. Software spielt in vielen Lebenslagen der Menschen eine wichtige Rolle und würde bei einem Ausfall zu fatalen Folgen führen. Um die Software zu verbessern, werden daher auch oft Refactoring betrieben. Für die Realisierung von Software werden dann auch sowas wie Agiles Programmieren, Scrum und Kanban verwendet. Das Ziel dieses Projektes ist es zu zeigen, dass durch eine gute Planung und unter Einhaltung moderner Programmier-Richtlinien eine releasefähige Software im Rahmen des Studienprojektes in einem angemessenen Zeitraum erstellt werden kann. Es wurde für die Entwicklung eines Spiels namens "Harsh-Politics" entschieden. Das Spiel baut auf altbekannte Spiele auf, wie Tekken, Super Smash Bros und andere diverse Kampfspiele, in der zwei Spieler gegeneinander kämpfen und unter anderem verschiedene Fähigkeiten besitzen. In diesem Projekt sollen bekannte Politiker gegeneinander kämpfen. Dabei besitzt jeder Politiker typische Wiedererkennungsmerkmale, die unter anderem auch als Spezialfähigkeit eingesetzt werden können. Mit diesem Spiel werden die Phasen der Softwareentwicklung in den folgenden Kapiteln beispielhaft verdeutlicht.

III. Anforderungen

Als relevante Stakeholder wurden die späteren Nutzer, das Projektteam und Lara Maria Stricker identifiziert. Frau Stricker ist die Dozentin des Moduls Software-Engineering II, in dessen Umfang das Projekt zu bearbeiten war. Sie fungierte somit als Auftraggeberin.

Als wichtigste Projektziele wurden folgende Anforderungen ermittelt:

- Frau Stricker stellte nur eine Anforderung an das zu entwickelnde Programm. Die Forderung, dass das Programm lauffähig ist, wurde daher als Projektziel mit höchster Priorität definiert.
- Als weiteres Projektziel hat das Entwicklerteam von Beginn an die Anforderung gestellt, dass es sich bei dem Programm um ein zweidimensionales Spiel, im eins gegen eins Modus handeln soll.

Sowohl funktionale als auch nicht-funktionale Anforderungen wurden als Team ermittelt, indem jedes Mitglied seine speziellen Wünsche ins Spiel einbrachte. Als Ermittlungstechnik wurde überwiegend Brainstorming genutzt. Darüber hinaus wurde auch die Methode des Wechsels der Perspektive angewandt. Dabei wurde aus der Sicht der späteren Nutzer versucht, fehlende Anforderungen zu ermitteln. Einige Anforderungen entstanden nicht zu Beginn, sondern entwickelten sich während des Entwicklungsprozesses. Auch die Risikoanalyse in der frühen Phase der Entwicklung brachte einige neue Anforderungen ein.

A. Nicht-Funktionale Anforderungen

- Auswahl verschiedener Spielfiguren: Die Nutzer sollen die Möglichkeit haben, verschiedene Fähigkeiten und verschiedene Spielfiguren zu benutzen. Es muss also eine Option geben, eine der verschiedenen Spielfiguren zu wählen.
- Spielfeldauswahl: Neben der Auswahl verschiedener Charaktere soll der Benutzer auch die Möglichkeit haben zwischen verschiedenen Spielfeldern zu wählen.
- Dynamische Animationen und Darstellungen: Verschiedene Darstellungen und Animationen sollen das Spiel dynamischer und interessanter machen. Beispiele dafür sind Sprung- oder Angriffsanimationen oder auch die Spiegelung der Charaktere je nach Bewegungsrichtung.
- Skalierbare Bildschirmauflösung: Als weitere Anforderung wurde die Skalierbarkeit der Bildschirmauflösung gewünscht. Die Anwendung soll die Größe verschiedener Bildschirme oder individuell gewünschte Größen annehmen können.
- Zuverlässigkeit: Die Anwendung muss zuverlässig funktionieren, soll möglichst flüssig laufen und nicht abstürzen.
- Benutzerfreundlichkeit: Der Nutzer sollte sich schnell und einfach im Programm zurechtfinden können. Dafür muss die Steuerung möglichst selbsterklärend und leicht verständlich sein, um ein optimales Spielerlebnis zu gewährleisten.

B. Funktionale Anforderungen

- Steuerbarkeit der Figuren: Damit die Spielfigur sich nach Anweisung des Anwenders bewegen kann, muss eine Eingabe auf der Tastatur möglich sein. Die Eingabe muss vom Programm ausgewertet und umgesetzt werden können.
- Zwei Spielfiguren, mit getrennter Steuerung: Um das Projektziel des "eins gegen eins" Spiels umzusetzen, muss es zwei getrennte Figuren geben. Damit zwei Nutzer gegeneinander spielen können, müssen die Steuerungen der Figuren verschieden sein.

- Figuren können springen: Der Anwender soll die Möglichkeit haben, seine Figur springen zu lassen.
- Angriffe auf den Gegner: Da zwei Nutzer gegeneinander spielen sollen, müssen Angriffe der eigenen Figur gegen die Figur des Gegners möglich sein.
- Spezialangriffe/-fähigkeiten: Neben den normalen Angriffen sollen Spezialangriffe oder Fähigkeiten möglich sein, die je nach ausgewähltem Charakter unterschiedlich sind.
- gegnerische Angriffe abwehren: Um den Angriffen des Gegners nicht hilflos ausgeliefert zu sein, sollen die Charaktere sich durch eine Anweisung gegen die Angriffe verteidigen können.
- Menü für Einstellungen: Als weitere Anforderung wünschten sich die Entwickler ein Menü, indem über Buttons verschiedene Einstellungen vorgenommen werden können. Darunter zählen neben der bereits erwähnten Spielfeld- und Charakterauswahl, auch das Ein- oder Ausschalten der Sounds.

Die hier genannten Anforderungen sind ein Auszug aus den im Projekt ermittelten Anforderung. Sie wurden von der Gruppe als am wichtigsten identifiziert.

IV. Organisation

Zu Beginn wurde die Projektgruppe in die folgenden Aufgabenbereiche unterteilt: Management, Entwicklung und Design/Art. Hierbei gab es einen Manager/Sounddesigner, drei Entwickler und zwei Artisten. Die Artisten waren außerdem unterteilt in Environment Design und Character Design. Das Projekt wurde mit Hilfe von Prinzipien der Agilen Softwareentwicklung verwirklicht. Wichtige Bestandteile hierbei waren die Inkrementelle Entwicklung und der Scrum Prozess [6]. Der Fokus der Entwickler lag anfangs vollständig darauf, sich mit Unity vertraut zu machen und die Basisfaktoren umzusetzen. Diese fundamentalen Funktionen und das Wissen, diese zu benutzen, waren grundlegende Kenntnisse, die benötigt wurden, um jeweils eigene Teilsysteme bearbeiten und bewältigen zu können. Hierbei gab es keine Vorgaben außer dem generellen "politischen" Thema des Spiels. Innerhalb der Projektphase wurden wöchentliche Meetings durchgeführt, die den aktuellen Sprint beendeten, auswerteten und mit dem ganzen Team zusammen den neuen Sprint vorbereiteten [6]. Die genaue Umsetzung der einzelnen Schritte war hierbei den Entwicklern belassen. Bis auf die Meetings gab es keine vorgeschriebenen Arbeitszeiten, da asynchron gearbeitet wurde. Es war jedoch wichtig das vorher festgelegte Pensum im aktuellen Sprint zu erledigen. Für die Kommunikation wurde überwiegend Whatsapp genutzt. Die Meetings fanden online über Discord statt. Für den Code-Austausch und die Versionierung wurde GitHub verwendet. Im Github-Repository wurde eine Aktivitätenliste mit den noch nicht erledigten Aufgaben erstellt. Dabei wurde das Projekt in drei Phasen unterteilt. Die jeweiligen Phasen wurden des Weiteren in einzelne Teilsysteme unterteilt, die bearbeitet werden müssen. Mit Hilfe des dadurch resultierenden Backlogs wurde in den Meetings besprochen, an welchem Punkt sich das Projekt befindet und welche Teilsysteme als nächstes bearbeitet werden müssen. Die ersten dieser Teilsysteme waren "Sprungphysics", "zwei Spielfiguren, die sich unabhängig voneinander bewegen können", "getrennte Steuerung" und "ein langer schmaler Background". Zudem wurde innerhalb der Meetings festgelegt, welche Aufgaben des letzten Sprints nicht zur vollständigen Zufriedenheit bearbeitet wurden. Diese wurden dann in den nächsten Sprint mit aufgenommen. Anschließend wurde ermittelt, welche Teilsysteme für welchen Aufgabenbereich die höchste Priorität haben. Anschließend wurde der nächste Sprint eingeleitet. Bevor Aufgaben als beendet eingetragen wurden, war es wichtig, dass sie von der Gruppe verifizieren zu lassen. Daraufhin hat die Gruppe ihr Feedback gegeben und dieses wurde bestmöglich umgesetzt. Erst dann wurden sie offiziell beendet und der Bearbeiter kann an einem neuen Teilsystem arbeiten.

V. Umsetzung

Als Entwicklungsumgebung wurde die Unity Engine gewählt, da diese neben einer Grafischen Oberfläche, welche sich leicht an die Anforderungen anpassen lässt und über ein Koordinatensystem verfügt, auch über Funktionen und Komponenten verfügt, welche auf das Programmieren von Spielen ausgelegt sind. Das Ziel des Spieles ist es die Lebenspunkte der gegnerischen Spielfigur auf Null zu Reduzieren und ihn somit zu besiegen. Um der gegnerischen Spielfigur Schaden hinzuzufügen, stehen jedem Spieler verschiedene Waffen zur Verfügung, die zufällig und in regelmäßigen Intervallen dem Spiel dynamisch hinzugefügt werden. Dabei handelt es sich um die Nahkampfwaffen wie Schwert, Schild und Dolch sowie eine Pistole, die für den Fernkampf dient. Zusätzlich zu diesen Waffen kann jeder Spieler abhängig von der gewählten Spielfigur eine Spezialattacke benutzen.

A. Movementsystem

Es wird bei der Steuerung der Spielfiguren eine Funktion verwendet, die abhängig von einer Geschwindigkeitsvariable die Bewegung der Spielfiguren ermöglicht. Dabei wird mit Hilfe dieser Variable die Änderung der Koordinaten der Spielfiguren berechnet. Indem man einem Spielobjekt die Unity Komponenten Rigidbody2D und Colliderbody zuweist, erhält das Objekt in dem Spiel einen "Körper". Das bedeutet, dass das Objekt von der Physik-Engine von Unity beeinflusst werden kann, was die Kollision mit anderen Objekten ermöglicht und eine Schwerkraft simuliert. Dies ist besonders wichtig um die Spielfigur springen zu lassen, da mit Hilfe dieser Schwerkraft automatisch die Änderungen der Koordinaten auf der Y-Achse berechnet werden und die Spielfigur auf einem Boden stehenbleiben kann.

B. Animationen

Ein wichtiger Punkt, der Spiele lebendig wirken lässt, sind die Animationen. In Harsh Politics sind diese Animationen in der Bewegung der Spielfiguren zu finden. So sind sowohl Ruhezustand und Sprungzustand der Spielfiguren animiert, als auch Ruhezustand und Angriffszustand der verschiedenen

Waffen. Um ein Spielobjekt zu Animieren stellt Unity die Animator Komponente zur Verfügung, die als Manager für verschiedene Animationen dient. In dem Animator kann man festlegen, wann eine Animation abgespielt werden soll und in welche Animation sie danach übergehen soll.

C. Aufbau des Spielfelds

Das Spiel startet im Hauptmenü, in dem sich jeder Spieler eine Spielfigur aussuchen kann, und man sich auf eine Spielszene einigt. Dabei gibt es jeweils die Auswahl zwischen 6 Spielfiguren und 6 Szenen. Sobald die Auswahl abgeschlossen ist, wird das Spielfeld gebaut. Es wird der Hintergrund festgelegt, Hindernisse und Böden hinzugefügt, und die Spielfiguren an ihre Positionen gesetzt. Da es ein großer Aufwand wäre jeden dieser Schritte einzeln auszuführen, bietet Unity die Möglichkeit sogenannte Prefabs zu erstellen. Prefabs sind von den Entwicklern vorgefertigte Einstellungen. Beim Bau eines Spielfeldes werden mehrere solcher Prefabs benutzt um den Aufwand des Programmes zu minimieren. Zum einen sind die Einstellungen für die einzelnen Szenen gespeichert, also die Positionen der Spielfiguren und Hindernisse und zum Anderen sind die Einstellungen für die Spielfiguren gespeichert. Dazu zählen Sprite, Verhalten und Animationen. So ist es möglich Spielfiguren und Szenen in jeder möglichen Variation schnell und zuverlässig in das Spiel zu laden.

D. Waffen

Eine wichtige Funktion des Spieles ist es, Waffen aufzuheben und wieder abzulegen. Dazu verfügt jede Spielfigur über ein ”Hand”-Objekt. Dieses Objekt hat einen Punkt für die Waffenposition definiert. Solange die Spielfigur keine Waffe hält, wird an diesem Punkt eine Faust als Platzhalter dargestellt. Darüber hinaus hat das ”Hand”-Objekt einen Bereich definiert, in der das Aufheben von Waffen möglich ist.

VI. Evaluation

A. Projektprozess

Die in diesem Paper schon vielfach erwähnten Methoden, wie Scrum und die agile Softwareentwicklung haben die Projektplanung und Projektumsetzung nicht geschädigt, sondern vielmehr die Kommunikation, die klare Aufgabenverteilung und das Zeitmanagement positiv beeinflusst. Dies ermöglichte die Umsetzung des Projektes in einer relativ kurzen Zeitspanne, obwohl die Projektverantwortlichen keinerlei Erfahrungen mitbrachten. Dies zeigt deutlich, dass die im Projekt eingesetzten Methoden, eine sehr große Rolle bei dem Projekterfolg spielten.

B. Anforderung

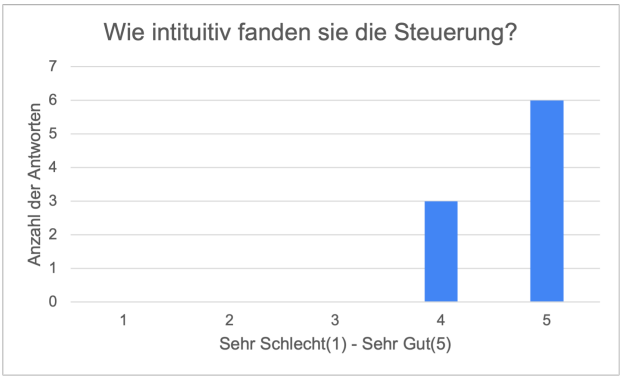
Durch die Anforderungsanalyse konnten klar definierte Implementierungsziele gesetzt werden. Durch klar formulierte Anforderungen konnten die Systemarchitektur und das UML-Diagramm erstellt werden. Die Implementierungsziele konnten dadurch in diskrete Aufgabengebiete unterteilt werden, sodass in den wöchentlichen Sprints einzelne Aufgaben in einer angemessenen Zeit bewältigt werden konnten.

C. Umsetzung

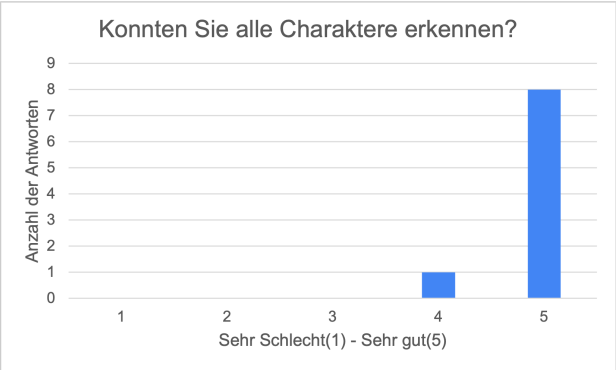
Von den im Kapitel 3 genannten Anforderungen konnte ein Großteil der Anforderungen umgesetzt werden. Die Steuerung der Spielfigur über die Pfeil- bzw. WASD-Tasten ist intuitiv, da diese Tastenbelegung ein Standard in der Branche darstellt. Das Abwehren von Angriffen konnte aus Zeitgründen nicht umgesetzt werden. Ein paar der umgesetzten Anforderungen zeigen jedoch Fehler auf. Beim Schlagen eines anderen Charakters mit einer Waffe wird die Schadensberechnung teilweise mehrfach durchgeführt. Die Ursache ist bekannt, konnte aber bisher nicht gelöst werden. Auch kann es dazu kommen, dass die Spielfiguren außerhalb des sichtbaren Bereiches des Spielfeldes gelangen können. Durch den stark begrenzten Zeitrahmen, konnten viele Code-Richtlinien nicht eingehalten werden, da dies mehrfaches Refactoring erfordert. Jedoch ist die KISS (KEEP IT SIMPLE AND STUPID) Richtlinie weitestgehend eingehalten worden. Deswegen konnten Fehler nicht im Vornherein ausgeschlossen werden, wie es z.B. bei Test-Driven-Development der Fall ist.

D. Umfrage

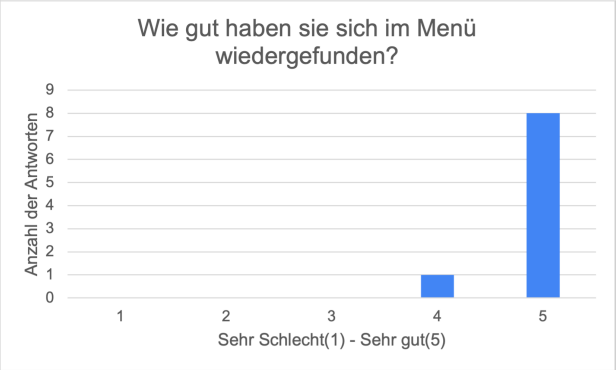
Es wurde eine Umfrage vorbereitet und durchgeführt. Es nahmen insgesamt 9 Personen an der Umfrage teil, die vor der Bearbeitung der Umfrage das Spiel unterschiedlich lang testeten. Jeder Teilnehmer konnte die Umfrage geheim durchführen und wurde nicht von den Entwicklern während der Teilnahme beeinflusst. Die Umfrage ermöglicht den Entwicklern weitere Anforderungen zu erfassen und Fehler zu erkennen. Das Ergebnis der Umfrage ist überwiegend positiv ausgefallen. Im Folgenden werden die Aspekte der Umfrage hinsichtlich der Bedienfreundlichkeit, Wiedererkennung und die allgemeine Resonanz vorgestellt. Die Auswertung der einzelnen Fragen ist rein qualitativ.

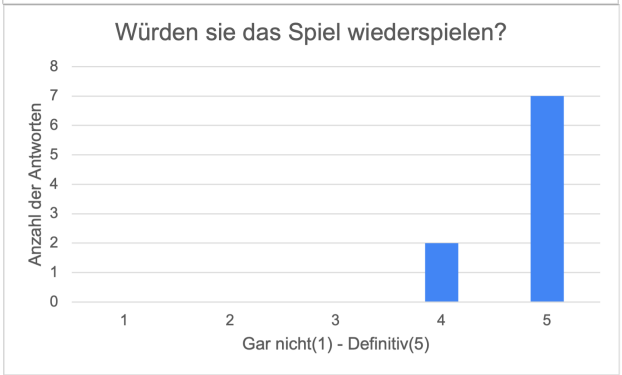


Die Steuerung wurde passend implementiert, da die meisten Tester ein intuitives Verständnis für die Tastenbelegung und die daraus resultierende Spielweise hatten.

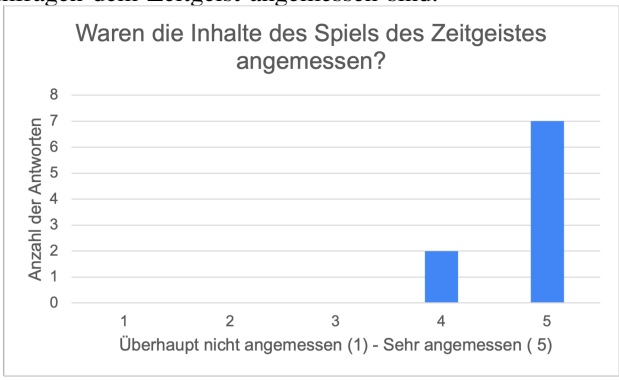


Alle Tester konnten die Spielfiguren erkennen. Das gewählte Design der Figuren war somit den Anforderungen entsprechenden und unterstützt den Erkennungswert des Spiels.





Das Spiel fand hohen Anklang bei den Umfrageteilnehmern. Der Wiederspielwert wurde von den Entwicklern als hoch interpretiert. Dies ist vermutlich unter anderem auf die humoristischen Inhalte zurückzuführen, welche laut den Umfragen dem Zeitgeist angemessen sind.



Es wurden auch Feature-Vorschläge gesammelt und Feedback zu den beliebtesten bestehenden Features. Vorgeschlagene Features waren unter Anderem eine Randomisierte Spielfigurenauswahl und Wünsche nach spezifischen Spielfiguren. Das Feedback zu den beliebtesten bestehenden Features und die oben dargestellten Umfragen, lassen darauf schließen, dass folgende Anforderung mit besonderem Erfolg von den Entwicklern umgesetzt worden sind:

- Auswahl verschiedener Spielfiguren.
- Verschiedene Spielszenen/Maps.
- Animationen und Darstellung
- Programmstabilität

Zudem wurden alle funktionale und nicht-funktionale Anforderung erfüllt, bis auf die Abwehr von gegnerischen Angriffen. Folgende Anforderungen konnten durch die Umfrage erfasst werden:

- Game-Over Event: Wird bei der Niederlage eines Spielers ausgelöst und führt die Nutzer zurück zum Menü
- Einfaches Waffen aufheben: Das Aufheben einer Waffe wurde von den Testern als zu schwierig bewertet. Waffen sollten demnach einfacher aufzuheben sein.
- Zufällige Auswahl von Spielfiguren und Spielszenen

VII. Fazit

Dieses Paper bietet eine detaillierte Beschreibung der Methodik und Arbeitsweise bei der Umsetzung des beschriebenen Projekts. Die Zusammenarbeit im Team sowie die eigenständige Konzeptionierung trugen maßgeblich zur Vertiefung der im vorherigen Fachsemester erworbenen Kenntnisse im Bereich Softwareengineering bei. Zudem bot das Projekt den Studierenden wertvolle Einblicke in den Ablauf von Softwareprojekten im späteren Berufsleben. Im Rahmen des Projektbearbeitungsprozesses haben die Studierenden Fertigkeiten und Kompetenzen erworben, die es ermöglichen, eigene Perspektiven und Ideen Teamkollegen gegenüber darzulegen und zu diskutieren. Sie durchliefen ebenfalls, auf Basis des Moduls ”Softwareengineering I”, verschiedene Etappen der Softwareentwicklung, wie Anforderungsermittlung sowie Aspekte der agilen Softwareentwicklung und lernten selbstständig Lösungen für herausfordernde Situationen zu erarbeiten. Diese Tätigkeiten beinhalteten die Modellierung von Klassen- und Sequenzdiagrammen durch den Gebrauch der Unified-Modeling-Language (UML). Außerdem haben die Studierenden gelernt, wie man ein Produkt auf eine verständliche Art und Weise mittels Projekt-Teasern gegenüber potenziellen Interessenten präsentiert. Unter Berücksichtigung der individuellen Stärken und Interessen der einzelnen Gruppenmitglieder wurden spezifische Aufgabenbereiche zugewiesen, um die Entwicklung des Projekts zu optimieren. Folglich gliederte sich das Entwicklerteam in zwei Gruppen: eine für die organisatorischen Belange und eine für die technische Umsetzung. Während die erstgenannte Gruppe für Aufgaben wie Grafikdesign, Abgabemanagement und

allgemeine Organisation verantwortlich war, konzentrierte sich die andere Gruppe hauptsächlich auf die Entwicklung und Implementierung der verschiedenen Spielmechaniken. Durch diese Aufteilung konnte die Arbeit am Projekt asynchron und zugleich effizient gestaltet werden. Dieser Ansatz betonte die Bedeutung einer gezielten Aufgabenverteilung, um die Stärken und Fähigkeiten jedes Mitglieds optimal zu nutzen und die Projektentwicklung zu maximieren. Die Strukturierung des Projekts in einzelne Phasen erwies sich im Nachhinein als vorteilhaft, da dadurch das Ziel der Erstellung eines funktionsfähigen Prototyps bereits in einem frühen Stadium des Entwicklungsprozesses erreicht werden konnte. Obwohl dieser Prototyp zunächst nur grundlegende Funktionen, wie die Steuerung oder Sprünge, enthielt, stellte er einen ersten bedeutenden Meilenstein dar, der den Übergang zur nächsten Phase des Projekts markierte. Später wurden zusätzliche Funktionen, wie das Kampfsystem oder Eingabe-Menüs, implementiert und kontinuierlich verbessert, was letztendlich zur Fertigstellung des Endprodukts führte. Zu Beginn des Projekts stellte die Entwicklungsumgebung der Unity-Engine als gänzlich neues Werkzeug eine Herausforderung dar, die eine gründliche Einarbeitung erforderte. Im weiteren Verlauf des Entwicklungsprozesses, konnte jedoch das Verständnis für die Plattform schrittweise erhöht werden, was in der Durchführung erster Testkonfigurationen anhand vordefinierter Beispielprojekte resultierte. Im Ergebnis wurde das erworbene Wissen konsolidiert und etwaige Unklarheiten innerhalb der Gruppe beseitigt. Diese Erfahrungen sind für die erfolgreiche Durchführung ähnlicher Projekte von Bedeutung und tragen dazu bei, dass ähnliche Herausforderungen in Zukunft effektiver gelöst werden können. Umfassende Gespräche trugen dazu bei, Dinge besser zu verstehen und erleichterten die Projektplanung. Wöchentliche Meetings brachten neue Meilensteine hervor, die gesetzt und schließlich durch neue Ziele ersetzt wurden. Das Projekt wurde mit einem funktionsfähigen Prototypen abgeschlossen, der fast alle Anforderungen erfüllt. Eine weiterführende Entwicklung wurde durch eine entsprechende Dokumentation und selbst erklärendem Code unterstützt, ist jedoch nicht festgelegt. Auch die Veröffentlichung des Projektes auf Platformen wie ”Steam” oder ”Good old Games” ist nicht geplant.

REFERENCES

[1] Booch, G. (2018). The History of Software Engineering. IEEE Software, 35(5), 108–114. https://doi.org/10.1109/ms.2018.3571234

[2] Sommerville, I. (2011). Software Engineering. Addison Wesley Longman.

[3] DRY vs KISS – Clean Code Prinzipien. (2018, 25. Mai). genericde. Abgerufen am 20. März 2023, von https://www.generic.de/blog/dry-vs-kiss-clean-code-prinzipien

[4] Oloruntoba, S. (2021, 19. Februar). SOLID: Die ersten 5 Prinzipien des objektorientierten Designs. DigitalOcean. Abgerufen am 20. März 2023, von https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design-de

[5] Was ist YAGNI? — Softwareentwicklung — PI-Lexikon. (2023, 23. März). pi-informatik. Abgerufen am 24. März 2023, von https://www.pi-informatik.berlin/pi-lexikon/softwareentwicklung/was-ist-yagni/

[6] Folien Lara Maria Stricker HWR-Berlin