Simulationsarbeit 2024

Dokumentation zur Simulationsarbeit

Dungeoncrawler

Abgabedatum: 02.02.2024

Bastian Lucht

FIN 11

Inhaltsverzeichnis

1 Einleitung

Diese Projektdokumentation wurde im Rahmen der Berufsschulinhalte in Form eines Softwareprojektes angefertigt. Sie bietet einen detaillierten Einblick in die definierten Ziele, den Verlauf des Entwicklungsprozesses sowie die erzielten Ergebnisse. Ziel ist es, einen Überblick über das Projekt und seine relevanten Aspekte zu vermitteln.

1.1 Projektumfeld

Der Auftraggeber war die Berufliche Schule der Hanse- und Universitätsstadt Rostock für Technik. Der Projektzeitraum wurde vom 04.09.2023 bis zum 02.02.2024 festgelegt. Der Projektverlauf wurde durch die Lernfelder 10, 11 und 12 begleitet und diente der Festigung der dort vermittelten Grundlagen.
Als Ansprechpartner für Fragen zu den Inhalten und potentiell auftretenden Problemen standen die Lehrer der Lernfelder zur Verfügung.

1.2 Projektziel

Das Ziel des Projektes war es, den Ablauf einer selbst gewählten Situation zu simulieren.
Eine Simulation wurde wie folgt definiert:
- Eine Anwendung, die ein durch Parameter beeinflusstes Ergebnis liefert.
- Das Ergebnis ist auch bei gleichbleibenden Parametern in jedem Durchlauf unterschiedlich.

Zur Umsetzung der Simulation wurde sich für einen vereinfachten Dungeoncrawler entschieden.
Innerhalb eines Durchlaufes trifft der Charakter auf verschiedene Arten von Gegnern und findet unterschiedliche Gegenstände um seinen Durchlauf zu bestreiten.

1.3 Projektbegründung

Die Inhalte der Lernfelder 10 – 12 können innerhalb des Schulunterrichtes überwiegend nur theoretisch vermittelt werden. Daher dient die Simulationsarbeit dazu, die praktische Anwendung dieser Inhalte besser zu verstehen und die Umsetzung eines Softwareprojektes zu üben.
Im Spezifischen betrifft das die Aspekte der Projektdokumentation, eines Benutzerhandbuches, einer Softwarelösung und der Ablaufplanung eines Projektes. Dies dient außerdem der Vorbereitung für das Abschlussprojekt der IHK.

2. Projektplanung

2.1 Vorgehensmodell

Ich habe mich für das Erweiterte Wasserfallmodell als mein Vorgehensmodell entschieden. Dadurch habe ich den Vorteil einer klaren und strukturierten Entwicklung. Ich beginne mit einer detaillierten Planung, definiere Phasen und Meilensteine. Dies ermöglicht eine bessere Kontrolle über den Entwicklungsprozess und erleichtert die frühzeitige Identifizierung von potenziellen Problemen. Ebenfalls habe ich beim Erweiterten Wasserfallmodell eine bessere Möglichkeit auf diese Probleme einzugehen, da ich stets in der Lage bin zu einer vorheriger Phase zurück zu kehren. Das Erweiterte Wasserfallmodell eignet sich besonders gut für Projekte mit stabilen Anforderungen, bei denen Änderungen während der Entwicklung eher unwahrscheinlich sind – wie es bei diesem Projekt der Fall ist. Außerdem ist es ein mir bereits vertrautes Modell aus meinem Arbeitsumfeld.

2.2 Ablaufplanung

Die Umsetzung des Projektes fand außerhalb der betrieblichen Arbeitsstunden und an gezielten Projekttagen (Donnerstag + Freitag) während der Berufsschul-Turni

statt. Im Zuge des gewählten Erweiterten Wasserfallmodells habe ich meine Zeit wie folgt eingeplant:

Phase	Zeit
Analyse	5h
Entwurf	10h
Implementierung	50h
Test	10h
Dokumentation	15h
Gesamt	**90h**

3. Analysephase

3.1 Ist-Analyse

Die geforderte Anwendung besteht bisher nicht und ist daher in vollem Umfang von mir umzusetzen.
Die Anwendung knüpft an kein bestehendes System an und wird auch kein zukünftiges System erweitern.

3.2 Anwendungsfälle

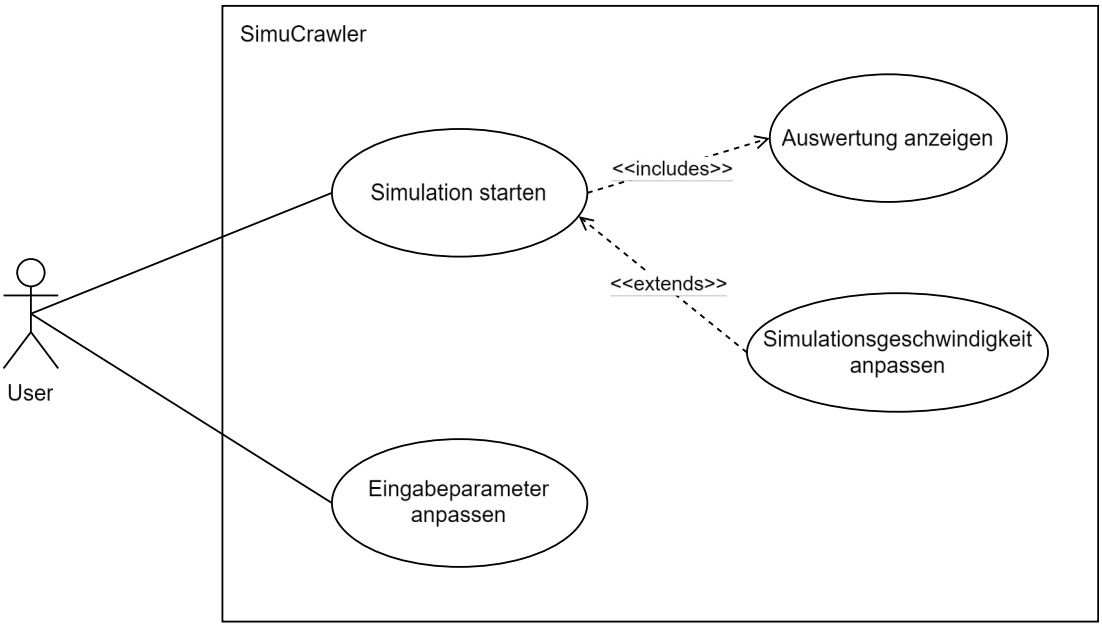

Schaubild 1: Anwendungsfalldiagramm

3.3 Lastenheft

3.3.1 Zielsetzung

Ziel des Projektes ist es, eine Anwendung zu fertigen, die einen selbstgewählten simulierten Ablauf darstellen kann. Die Anwendung muss der Definition einer Simulation genügen:

- Eine Anwendung, die ein durch Parameter beeinflusstes Ergebnis liefert.
- Das Ergebnis ist auch bei gleichbleibenden Parametern in jedem Durchlauf unterschiedlich.

3.3.2 Anforderungen

Folgende Anforderungen wurden an die Simulation gestellt:

- soll über eine GUI ansprechbar sein

- besitzt mind. 5 Eingabeparameter, die über eine UI einstellbar sind
- die Simulationsgeschwindigkeit soll mind. 3 verschieden Stufen annehmen können
- besitzt mind. 2 Zufallsparameter, die die Simulation während eines Durchlaufs beinflussen
    - mind. einer dieser Parameter soll per Normalverteilung verteilt sein
- enthält mind. eine Situationsbezogene Animation
- es erfolgt eine grafische Auswertung
- die Simulation muss zeitabhängig sein

3.3.3 Benutzeroberflächen

Die Benutzeroberfläche soll den im Lernfeld vermittelten UI-Prinzipien nach Nielsen genügen.

3.3.4 Leistungsanforderungen

Die Anwendung soll auf einem Windows System laufen.
Die Anwendung soll ohne Performance Beeinträchtigungen auf der von der Schule bereit gestellten Hardware laufen.

3.3.5 Technische Anforderungen

Die zu verwendende Programmiersprache ist frei wählbar, solange es sich dabei um eine objektorientierte Sprache handelt. Die Gestaltung einer Web-Anwendung ist ebenfalls möglich. Für die Umsetzung zusätzlich verwendete Frameworks oder Libraries sind ebenfalls frei wählbar.

4. Entwurfsphase

4.1 Zielplattform

Das Projekt wird als Desktopanwendung unter Berücksichtigung der Anforderungen

umgesetzt. Zur Umsetzung wird die Godot Engine in der Version 4.2 und die dazugehörige Sprache GDScript verwendet. Für die Version 4.2 habe ich mich entschieden, da es die latest stable Version zum Zeitpunkt des Projektstartes ist. Ich habe mich für diese Technologien entschieden, da ich an der Verwendung einer Game Engine interessiert war und diese Möglichkeit nutzen wollte um darin Erfahrung zu sammeln. Der Einstieg in so eine Technologie ist durch die vereinfachte Bedienung der Godot Engine erleichtert. Die Verwendung einer Game Engine ermöglicht eine effiziente Umsetzung eines durch Parameter beeinflussten Ablaufes, da diese in der Spielentwicklung üblich sind. Außerdem eignet sich die Godot Engine gut dafür, eine GUI, dessen Navigation und zur Simulation passende Animationen umzusetzen. Die Verwendung von GDScript ermöglicht eine klare Trennung im Backend durch Modularität der Skripte.

4.2 Architekturdesign

Die Godot Engine verbindet mehrere Patterns (wie z.B. das Observer Pattern, Singletons oder das Protoype Pattern) und ermöglicht während der Entwicklung ein dynamisches Nutzen dieser bereits in der Engine eingebundenen Designs. Das überliegende Prinzip ist der SceneTree - das Zusammenspiel von Nodes in der Oberfläche mit der Möglichkeit Skripts in diese einzubinden. Über Skripts können Nodes miteinander kommunizieren, entweder durch direkte Referenzen oder über Signale, die ausgesendet werden können.

4.3 Benutzeroberfläche

Die Benutzeroberfläche wurde in Form einer GUI mittels des Godot Engines umgesetzt. Die GUI besteht aus mehreren einzelnen Ansichten die ineinander überleiten und über Buttons angesprochen werden. Die Inhalte der Ansichten begrenzen sich ebenfalls auf bekannte Bedienelemente wie Buttons, Dropdowns, Slider oder Spinboxen zum intuitiven Einstellen von Parametern oder beeinflussen

der dargestellten Abläufe. Die Benutzeroberfläche wurde angelehnt an die UI-Prinzipen von Nielsen designed.

4.4 Geschäftslogik

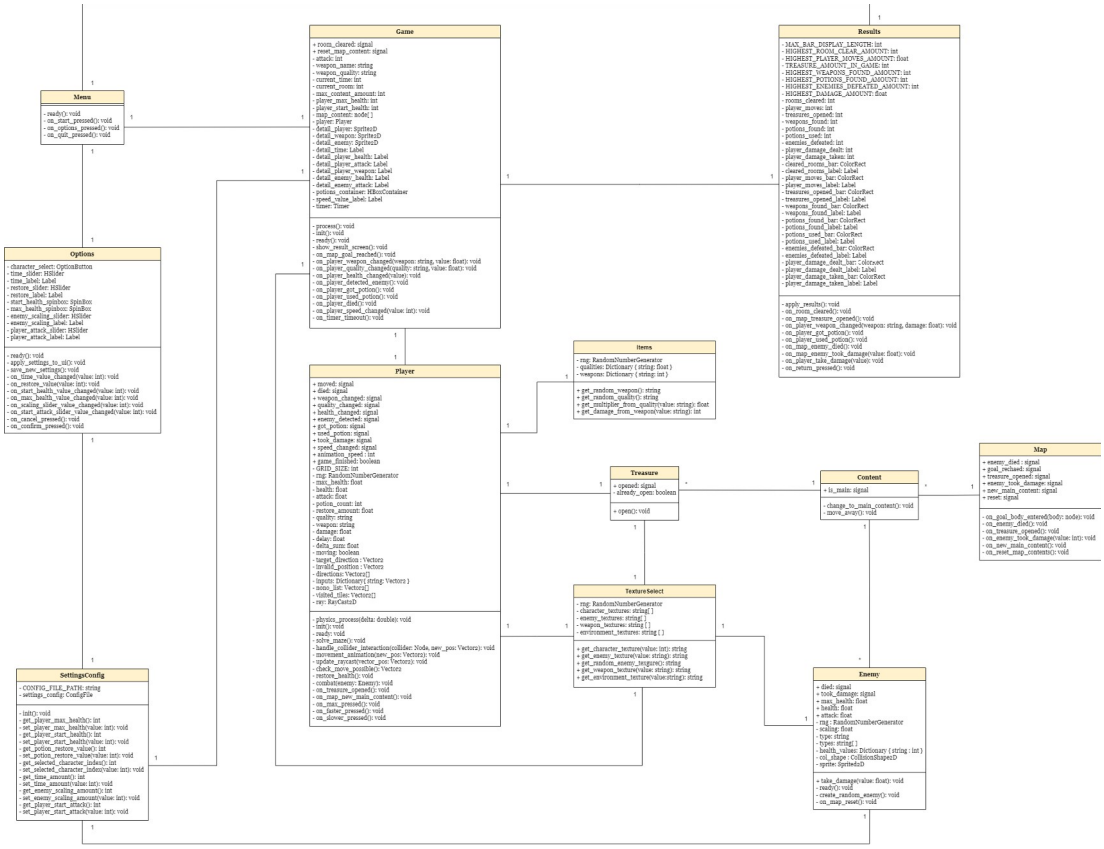

Schaubild 2: Klassendiagramm

4.5 Pflichtenheft

4.5.1 Zielsetzung

Entwicklung einer Simulation in Form eines Dungeoncrawlers unter Verwendung der Godot Engine.

4.5.2 Anforderungen

- Eine GUI wird mit der Godot Engine mittels Szenen umgesetzt
- 6 veränderbare Eingabeparameter werden in der GUI durch Spinboxen und Slider einstellbar sein
- Zufallsparameter:
    - Innerhalb eines Durchlaufes wird die Art der kreierten Gegner zufällig ausgewählt. Ebenso sind die Gegenstände, die aus einer Truhe gewonnen werden können zufällig.
    - Die Qualität und Art des generierten Equipments ist normalverteilt.
- Die Simulationsgeschwindigkeit ist über 3 Buttons in insgesamt 20 Stufen einstellbar.
- Die Animation wird durch die Bewegung des Spieler Charakters umgesetzt.
- Die grafische Auswertung erfolgt über eine Ansicht, die gesammelte Daten mit repräsentativen Graphen sowie deren Werten darstellt.
- Die Simulationsdauer hängt von einem einstellbaren Parameter ab. Nach Ablauf dieser Zeit wird der aktuelle Durchlauf beendet und die Daten ausgewertet.

4.5.4 Abnahme

Die Abnahme erfolgt, indem die angefertigten Dokumente sowie die Anwendung in das Schulnetzwerk geladen werden. Dort wird durch den Fachlehrer die Funktionalität der Anwendung und Vollständigkeit der Dokumente geprüft.

5. Implementierungsphase

5.1 Implementierung der Benutzeroberfläche

Die Umsetzung der GUI mittels der Godot Engine fand über das Erstellen von Szenen statt, welche mit relevanten Oberflächenelementen und für die Darstellung des Simulationsdurchlaufes nötigen Nodes gefüllt wurden. Jede Szene hatte ihre spezifische Funktion, um die Interaktionen des Benutzers zu erleichtern. Erstellte Szenen sind das Hauptmenü, das Optionsmenü, die Ansicht, welche den Simulationsablauf darstellt und die Auswertung. Die Flexibilität der Godot Engine ermöglichte eine einfache Anpassung der Szenen. Die Szenen wurden über Skripts miteinander verbunden, die den Wechsel zwischen Ansichten ermöglichen sowie relevante Informationen von einer Szene zur anderen übertragen. Ebenfalls regelten die mit den Szenen verbundenen Skripte die Logik für Benutzerinteraktionen mit der Oberfläche, wie z.B. das Aktualisieren von Labeltexten oder speichern der gewählten Parameter. Die Oberfläche wurde in einem Pixeldesign erstellt. Dafür wurden zusätzliche Fonts importiert und auf repräsentive Elemente sowie Tooltips angewendet. Ebenso wurden Assets zur Gestaltung der Dungeon Layouts im Pixel Design implementiert. Der Farbton der Oberflächen wurde im dunkleren Farbspektrum gehalten um der „Dungeon“ Ästhetik des Projektthemas gerecht zu werden.

5.2 Implementierung der Geschäftslogik

Zuerst wurde die Szene sowie Node für das überliegende Element der Spiellogik implentiert. In diese Node wurden alle für einen Ablauf relevanten Nodes eingebunden. Diese Nodes sind die Player, Enemy, Map, Goal und Treasure Nodes. Zunächst wurden die jeweiligen Nodes mit ihren Attributen und Methoden gefüllt, welche sich aus den Vorüberlegungen der Projektplanung ergeben haben wie z.B.

die Player Health und dessen Möglichkeit sich selbstständig durch eine Labyrinth-Struktur bewegen zu können. Im Falle der Map Node wurden die Layouts der verschiedenen Räume des Dungeons designed und mit Kollisionslogik ausgestattet sowie mit Gegnern, Truhen und dem jeweiligen Ausgang gefüllt. Anschließend wurden die jeweiligen Nodes mit Signalen ausgestattet, um mit anderen Nodes interagieren zu können. Ein Großteil dieser Kommunikation war ein Weiterleiten eines Prozesses an die überliegende Spiellogik zum Updaten der Statistiken und zum Sammeln der Daten für die Auswertung. Einige Global verfügbare Skripte wurden erstellt für Assets und Daten, auf die von überall Zugriff erlaubt sein darf. Dies betrifft die Texturen für die in der Map dargestellten Elemente, die Logik für das Generieren von Equipment und das einholen zugehöriger Attribute wie der Multiplikator einer bestimmten Qualität und die Logik für das Interagieren mit der Config Datei, in der der die eingestellten Parameter gespeichert und ausgelesen werden. Abschließend wurde die Logik für das Verarbeiten und Darstellen der Daten aus der Spiellogik für die Auswertung implementiert.

6. Qualitätssicherung

Um die Qualität des Codes zu sichern wurde das Backend an in gezielten Interaktionen getestet. Für die Umsetzung der Tests wurden 3 verschiedene Test-Libraries in Betracht gezogen:

1) Die Builtin Godot Test Suite:
Die Builtin Library von Godot lässt Testausführung ausschließlich über Konsoleneingaben zu und erfordert, dass die anzulegenden Tests in C++ geschrieben werden. Um das Backend einheitlich in GDScript zu halten habe ich mich gegen diese Library entschieden.

2) Die GUT Library (Godot Unit Testing):
Die GUT Library ist derzeit noch inkompatibel mit der Godot Version 4.2 somit habe ich mich ebenfalls gegen diese entschieden.

3) GdUnit4:

GdUnit4 ermöglicht die Testausführung direkt innerhalb des Godot Editors und bringt direkt gezielte Funktionen für das Testen mit sich. Außerdem ermöglicht es das Schreiben von Tests in GDScript. Aufgrund dieser Qualitäten habe ich mich für die Umsetzung mit GdUnit4 entschieden.

Ich habe mich dazu entschieden, die 4 Scripte zu testen, die am häufigsten agieren oder mit anderen Nodes interagieren:

Player:

Der Spieler Charakter bewegt sich durchgehend durch den Dungeon und interagiert somit mit den Inhalten der einzelnen Räume wie Gegner, Truhen und den Ausgängen. Die Möglichkeit seine Position zu verändern, Kollisionen zu verarbeiten, Schaden zu nehmen, Leben zu regenerieren und Signale an die Spiellogik zu schicken wurden durch Tests abgedeckt.

Enemy:

Die Gegner bilden Hindernisse für den Spieler, die beseitigt werden können. Beim Initialisieren eines Gegners werden dessen Attribute festgelegt. Durch Tests werden die Zuweisung neuer Attribute bei Initialisierung, die Darstellung eines Hindernisses durch aktive Kollision, das beseitigen der Kollision sobald der Gegner besiegt wurde sowie das korrekte Verhalten beim Erhalt von Schaden abgedeckt.

Treasure:

Truhen werden Spielern geöffnet, allerdings dürfen diese nur ein mal geöffnet werden und bleiben auch nach einem Loop in diesem Zustand. Die Tests decken ab, dass eine Truhe nur ein mal geöffnet werden kann, dessen Textur nach Öffnung angepasst wurde um zu vermitteln, dass diese bereits geöffnet wurde und dass das Signal des erfolgreichen Öffnens an den Spieler übermittelt wurde.

Game:

Die Spiellogik ist dafür zuständig die Layouts des Dungeons zu verändern, sobald ein Spieler den jeweiligen Ausgang eines Raumes erreicht hat. Durch die Tests wird

sichergestellt, dass sich das Layout nach beenden eines Raumes verändert, dass die Layouts Loopen, wenn die Maximalanzahl der Layouts erreicht wurde und dass den Gegnern vermittelt, dass ein Loop stattgefunden hat um diese zu reinitialisieren.