INTERNATIONALE HOCHSCHULE

Portfolio

DLMCSPSE01_D

Projekt: Software Engineering

2D Spiel „SAND“

Erstellt von: Keanu Semmel

Matrikelnummer: 32005464

Studiengang: Informatik (M.Sc.)

Tutor: Herr Dr. Markus Kleffmann

Datum: 25.07.2023

II Inhaltsverzeichnis

III Abbildungsverzeichnis

1 Projektdokumentation

1.1 Softwareprozess / Vorgehensmodell

Unter Berücksichtigung des erstellten Zeitplans, welcher sich aus Milestones und Arbeitspaketen zusammensetzt, findet für das Projekt der Entwicklung eines 2D-Spiels das Wasserfallmodell Verwendung. Dies geht darauf zurück, dass es sich bei dem Wasserfallmodell um einen sequenziellen und strukturierten Ansatz handelt, bei dem jede Phase auf der vorherigen aufbaut und klare Meilensteine Verwendung finden, um den Fortschritt zu überwachen. Unter Berücksichtigung des begrenzten Zeitrahmens, welcher für das Projekt eingeräumt wurde, und der bereits im Vorfeld festgehaltenen Anforderungen sowie der zugehörigen Projektierung, wird es ermöglicht mittels des Wasserfallmodells eine geordnete Entwicklung sowie ein effizientes Projektmanagement zu gewährleisten. Die klare Planung sowie Dokumentation unterstützen dahingehend, dass das Projekt innerhalb des vorgegebenen Zeitrahmens erfolgreich abgeschlossen werden kann.

1.2 Technologien und Tools

In der Entwicklungsphase des 2D-Spiels werden bestimmte Technologien, Tools und Third-Party-Libraries verwendet, um eine erfolgreiche Umsetzung des Projekts sicherzustellen. Die gewählte Programmiersprache ist Python aufgrund ihrer Benutzerfreundlichkeit und Eignung für die schnelle Prototypentwicklung und Spieleentwicklung.

Als Hauptframework wird Pygame eingesetzt, eine kostenfreie Bibliothek für die Entwicklung von 2D-Spielen in Python, die umfangreiche Funktionen zur Grafikdarstellung, Audio-Verarbeitung und Eingabe bietet für die Entwicklung von Spielen bietet.

Zur Sicherung der jeweiligen Speicherstände beziehungsweise High-Scores und Ausgabe dieser, wird SQLite als leichtgewichtige und einfache Datenbanklösung verwendet. Mittels des Versionskontrollsystem GitHub wird, die Verwaltung und Versionierung des Quellcodes während und nach dem Entwicklungsprozess unterstützt.

Für die Visualisierung der Anwendungsstruktur und Dokumentation der Komponenten und Abhängigkeiten wird das Online-Tool "app.diagrams.net" genutzt. Hiermit wird es ermöglicht, die geforderten Diagramme auszuarbeiten.

Darüber hinaus werden externe Quellen wie beispielsweise "OpenGameArt" (https://opengameart.org/) für den lizenzfreien Bezug sowie "Piskel" (https://www.piskelapp.com/) zur Anpassung der jeweiligen Assets verwendet. Die Nutzung von Assets mit freien Lizenzen ermöglicht die Integration hochwertiger Grafiken und Soundeffekte ohne urheberrechtliche Gefährdungen beziehungsweise Risiken einzugehen.

Mittels der Bearbeitung von Assets, unter Verwendung von Piskel, wird es ermöglicht die Anforderung eines ästhetischen Erscheinungsbilds zu erfüllen.

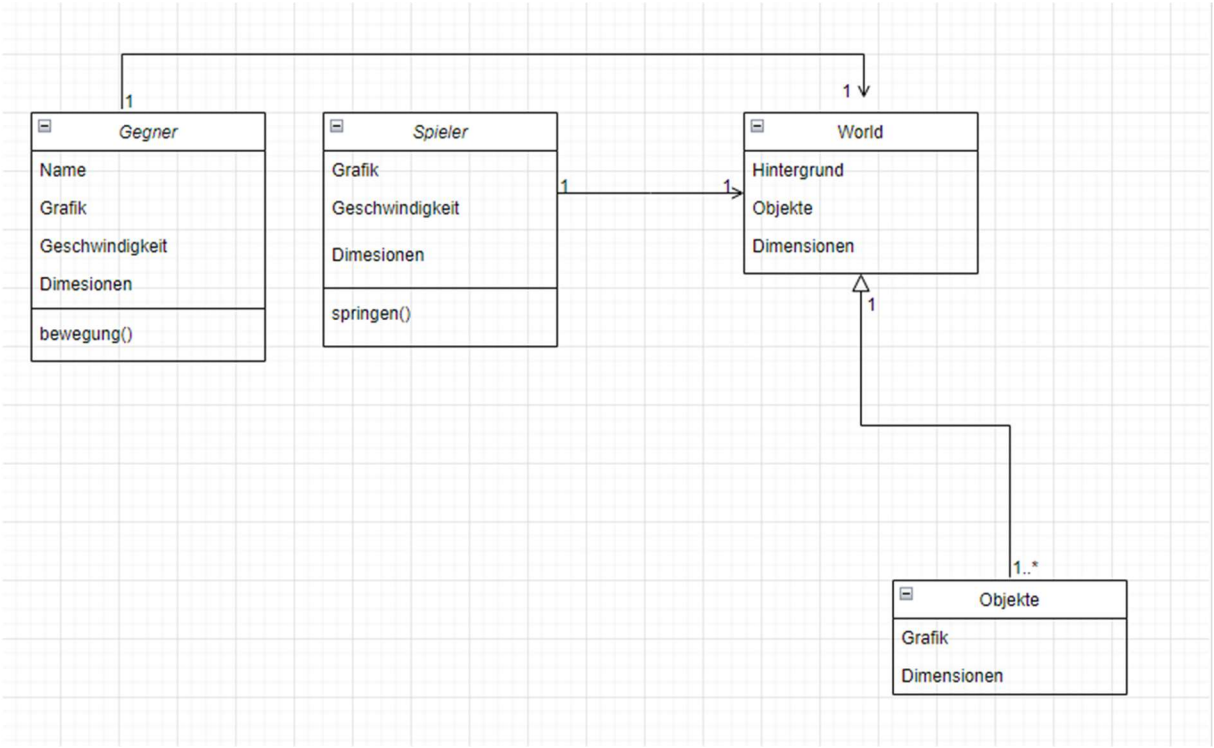
Es gilt jedoch zu beachten, dass unvorhergesehene Schwierigkeiten/Änderungen zu Abweichungen innerhalb des angewendeten Technologie-Stacks führen kann. Um einen transparenten sowie nachvollziehbaren Entwicklungsprozess zu gewährleisten, werden die gegebenenfalls aufkommenden Abweichungen sorgfältig dokumentiert und in der laufenden Projektdokumentation festgehalten. Als eine zentrale Referenz kann die Projektdokumentation gesichtet werden. Hier werden sämtliche Änderungen, Herausforderungen sowie Lösungen dokumentiert und der vollständige Entwicklungsverlauf festgehalten.

1.3 UML-Diagramm

Nachfolgend findet sich ein erster Entwurf eines UML-Diagramms, welches eine Struktur des zu entwickelnden 2D-Spieles beinhaltet.

Hierbei lässt sich erkennen, dass der Gegner, die Objekte sowie der Spieler innerhalb der Welt vorkommen werden. Es gilt jedoch zu berücksichtigen, dass sowohl der Gegner als auch der Spieler jeweils lediglich einmal in der Welt vorkommen können. Des Weiteren gilt es zu erwähnen, dass die Welt mindestens ein Objekt aufweisen muss. Ein weiterer wichtiger Aspekt ist, dass es sich hierbei lediglich um eine grobe Skizzierung/vorläufigen Entwurf handelt, welchen es im Rahmen des weiteren Projektes zu verfeinern gilt.

Abbildung 1: UML-Diagramm



Quelle Bild: eigene Darstellung (2023)

2 Projektplan

2.1 Ziele, Umfang und angestrebtes Ergebnis des Projekts

Das Ziel dieses Projekts ist die Entwicklung eines 2D-Spiels mit einem laufenden Charakter, der sich in einer 2D-Welt bewegen kann (Vor- und Rückwärts sowie links und rechts) kann. Der Charakter wird von ein bzw. mehrere Verfolger/n gejagt und muss möglichst lange entkommen, ohne vom Gegner berührt zu werden. Zusätzlich sollen die letzten drei Highscores angezeigt werden. Der Fokus liegt auf Indie-Game-Liebhabern sowie Personen mit eingeschränkter Hardware und Zeit.

2.2 Anvisierte Zielgruppe

Die Zielgruppe des Spiels spiegelt sich in Indie-Game-Liebhabern wider, die Interesse an unterhaltsamen 2D-Spielen haben. Darüber hinaus richtet sich das Spiel an Personen mit eingeschränkter Hardware, um sicherzustellen, dass das Spiel auf einer Vielzahl von Geräten reibungslos verwendbar ist. Zudem gilt es mittels des Spiels Menschen anzusprechen, welche über wenig Zeit verfügen, da das zu entwickelnde Spiel in kurzen, aufregenden Spielsitzungen absolviert werden kann

2.3 Potenzielle Projektrisiken und Gegenmaßnahmen

Potenzielle Projektrisiken können technische Schwierigkeiten bei der Implementierung der Logik sowie in Zeitplanprobleme auftreten. Um diesen Risiken entgegenzuwirken, gilt es den erstellten Projektplan mit den benannten Milestones sowie den jeweiligen Arbeitspaketen zu berücksichtigen. Frühzeitig sollen, um etwaige Verzögerungen zu verhindern, mögliche technische Herausforderungen identifizieren und entsprechende Recherchen durchgeführt werden.

2.4 Zeitplan und Meilensteine

1. Konzeptionsphase (1 Tag):

Projektaufgaben verstehen und Anforderungsdokumentation vorbereiten. Management Summary, Systemumfang und Kontext, funktionale und nicht-funktionale Anforderungen sowie das Glossar gemäß Vorgabe erstellen und einreichen.

2. Technische Planung (2 Tag):

Installation der benötigten Tools und Frameworks (Python, Pygame, pathfinding, sys, random, Docker, SQLite, GitHub etc.). Eventuell zusätzliche Recherchen einleiten und dokumentieren.

3. Erstellung des UML-Diagramms (1 Tag):

UML-Diagramm erstellen, um die Struktur der Anwendung zu visualisieren und die Komponenten und Abhängigkeiten zu dokumentieren.

4. Entwicklung der Spiellogik (5 Tage):

Implementierung der Charaktersteuerung (Laufen). Programmierung der Objekte und Verfolgerlogik. Umsetzung der Highscore-Verwaltung zur Speicherung der Rekorde.

5. Entwicklung der Benutzeroberfläche (1 Tage):

Gestaltung des Startbildschirms mit einem Start- sowie Exit-Button und Anzeige der letzten drei Highscores.

Implementierung der Spieloberfläche mit allen erforderlichen Anzeigeelementen.

6. Spieltest und Fehlerbehebung (2 Tage):

Ausgiebige Tests des Spiels.

Behebung von Fehlern und Optimierung der Leistung.

7. Dokumentation (1 Tag):

Abschließende Erstellung und Aktualisierung der Projektdokumentation.

Überprüfung der Vollständigkeit und Genauigkeit aller Dokumente.

8. Projektabschluss (1 Tag):

Letzte Tests und Sicherstellung, dass das Projekt den Anforderungen entspricht.

Vorbereitung der finalen Abgabe und Präsentation

Gesamtdauer: 14 Tage

3 Anforderungsdokument

3.1 Management Summary

Dieses Anforderungsdokument beschreibt die Entwicklung eines fesselnden 2D-Spiels für PC-Plattformen. Das Ziel ist es, ein Spiel zu schaffen, das Indie-Game-Liebhaber, Menschen mit eingeschränkter Hardware und solche, die nach kurzweiliger Unterhaltung suchen, anspricht.

Das Spielerlebnis umfasst einen Charakter mit Laufbewegung, der Vor- und Rückwärts sowie links und rechts gehen kann. Eine aufregende Verfolgungsjagd entsteht, indem der Spieler von einem Verfolger gejagt wird und dabei geschickt die Spielwelt nutzt.

Für eine hochwertige Darstellung werden PNG-Dateien für die Grafiken verwendet. Die Entwicklung des Spiels erfolgt nach dem strukturierten Wasserfallmodell, das klare Phasen für Konzeption, Implementierung, Test und Abschluss umfasst.

Die Entwicklung erfolgt eigenständig, wobei Python und Pygame für die 2D-Spielefunktionen genutzt werden. Docker wird eingesetzt, um das Projekt zu containerisieren, und SQLite wird verwendet, um die Score-Verwaltung zu ermöglichen. Des Weiteren findet pathfinding für die Implementierung der Lauflogik des Gegners Verwendung.

Das Spiel wird speziell für PC-Plattformen entwickelt und soll auf verschiedenen Hardwarekonfigurationen reibungslos funktionieren, um eine breite Zielgruppe zu erreichen.

Funktionale Anforderungen umfassen die intuitive Steuerung des Charakters in einer 2D-Spielewelt und das Bewältigen der Verfolgung durch einen Algorithmus gesteuerten Gegner. Die letzten drei Highscores werden dem Spieler auf dem Startbildschirm angezeigt, um einen entsprechenden Wettbewerbsanreiz zu schaffen. Nicht-funktionale Anforderungen umfassen eine flüssige Spielerfahrung mit einer Framerate von mindestens 30 FPS.

Ein Glossar mit Fachbegriffen wird erstellt, um ein einheitliches Verständnis über etwaige Begrifflichkeiten sicherzustellen.

Da das Projekt allein entwickelt wird, sind klare Meilensteine im Zeitplan gesetzt. Kontinuierliches Feedback und transparente Kommunikation während der Tests sowie im Rahmen dieser Dokumentation sind entscheidend, um die Qualität des Spiels sicherzustellen und sicherzustellen, dass das Endprodukt die Spieler begeistert. Die Entwicklung erfolgt mit Begeisterung und Liebe zum Detail, um ein unterhaltsames und ansprechendes Spielerlebnis zu gewährleisten.

3.2 Systemumfang und Kontext

Das System umfasst ein eigenständiges 2D-Spiel für PC-Plattformen, das von einer Einzelperson entwickelt wird. Das Spiel verwendet PNG-Dateien für die Grafiken von Charakteren, Objekten/Wände und dem Verfolger, um eine ansprechende visuelle Darstellung zu ermöglichen.

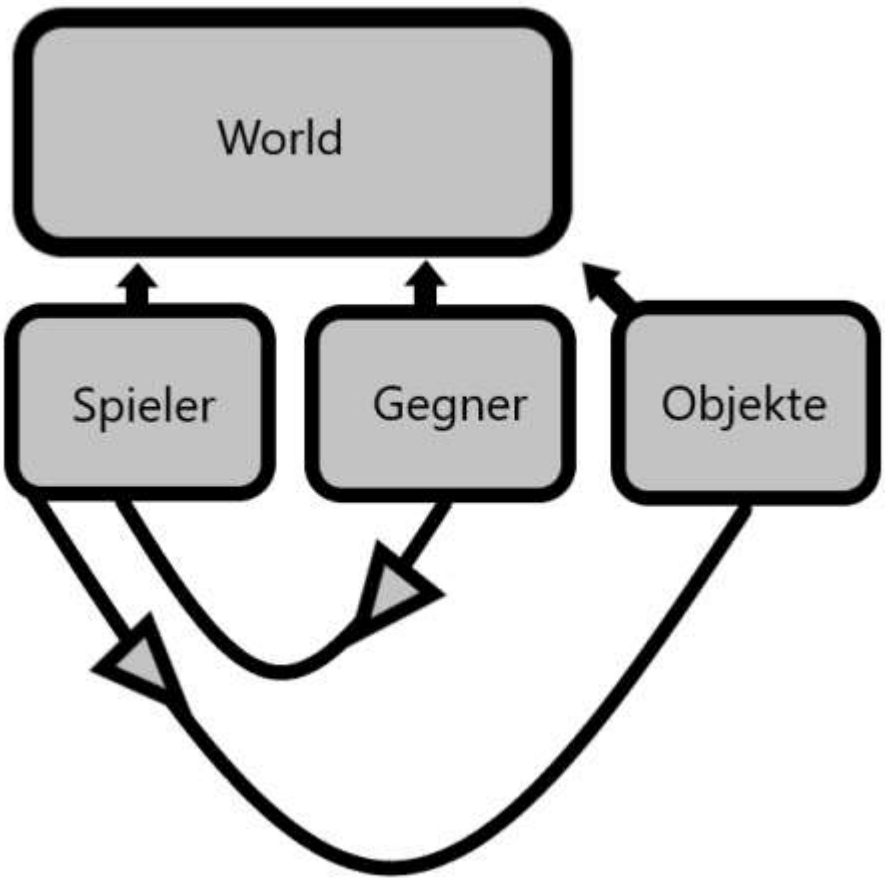
Der Umfang des Systems lässt sich wie folgt aufführen:

- Einen Charakter mit Laufbewegung, der sich unter anderem vorwärtsbewegen kann.
- Eine verzweigte Karte, die vom Spieler genutzt werden muss, um dem Gegner möglichst lange zu entkommen.
- Einen durch einen Algorithmus gesteuerten Verfolger, der den Spieler automatisch jagt, wenn dieser erzeugt wird.

Das Spielziel ist es, für eine möglichst lange Dauer vor dem Verfolger zu fliehen ohne, dass dieser in Kontakt mit dem Spieler kommt. Der Fortschritt des Spielers wird in Form der Zeit, welche der Spieler dem Gegner entkommen ist, festgehalten (Highscore). Das Spiel wird als eigenständige Anwendung entwickelt, die keine externen Ressourcen (bspw. Internet) benötigt, um eine reibungslose Ausführung auf verschiedenen Hardwarekonfigurationen zu gewährleisten. Der Spieler übernimmt die Steuerung des Charakters, um in der Karte zu manövrieren und das Spielziel zu erreichen. Das Gameplay wird von der Interaktion zwischen dem Charakter, der Welt und dem Verfolger geprägt, was für eine spannende und herausfordernde Spielerfahrung sorgt.

Nachfolgende Abbildung zeigt einen Entwurf eines Kontextdiagrames beziehungsweise skizierte erste Zusammenhänge. Hieraus wird ersichtlich, dass sich der Spieler, Gegner sowie Objekte (Wände) innerhalb der Spielwelt befinden und darin agieren. Des Weiteren besteht seitens des Gegners zu Lasten des Spielers eine Beziehung, welche sich darin ausdrückt, dass der Gegner versucht den Spieler einzuholen beziehungsweise zu kollidieren und somit den Game Over auszulösen. Ein weiterer Kontext besteht zwischen dem Objekt und dem Spieler. Die Objekte sind Teil der Welt und finden sich in Form von Wänden wieder, welche letztlich wie ein Labyrinth wirken.

Abbildung 2: Kontextdiagramm



Quelle Bild: eigene Darstellung (2023)

3.3 Funktionale Anforderungen

1. Der Charakter soll sich vorwärts-, rückwärts und seitlich bewegen können.
2. Der Algorithmus-gesteuerte Verfolger soll den Spieler automatisch jagen.
3. Wenn der Charakter von einem Gegner berührt wird, endet das Spiel mit einem "Game Over".
4. Der Spieler soll das Spiel beenden können, indem er es über den Button im Start-Menü oder im Spiel über das Rote-Kreuz verlässt.
5. Das Spiel soll eine klare Anzeige haben, wie weit der Spieler gekommen ist.
6. Der Spieler soll die Möglichkeit haben, das Spiel nach einem "Game Over" oder einem Sieg erneut zu starten.

3.4 Nicht-funktionale Anforderungen

1. Das Spiel soll eine flüssige Spielerfahrung bieten und eine Framerate von mindestens 30 FPS auf verschiedenen Hardwarekonfigurationen unterstützen.
2. Die Steuerung des Charakters soll intuitiv und reaktionsschnell sein.
3. Die Spielumgebung soll ansprechend und visuell ansprechend gestaltet sein.
4. Das Spiel soll leicht verständliche Anweisungen für den Spieler bieten, um das Gameplay zu erleichtern.

3.5 Glossar

- Charakter: Die spielbare Figur, die vom Spieler gesteuert wird.
- Verfolger: Die KI-gesteuerte Figur, die den Spieler jagt.
- FPS: Frames per Second, die Anzahl der Einzelbilder pro Sekunde, die das Spiel darstellen kann.
- Pygame: Ein Python-Framework, das zur Entwicklung von 2D-Spielen verwendet wird.
- SQLite: Eine leichte, serverlose Datenbank-Engine, die zur Speicherung von Highscore-Daten verwendet wird.
- Highscore: Die besten erzielten Ergebnisse der Spieler, die als Rekorde in der Datenbank gespeichert werden.
- Pathfinding: Finden des kürzesten Weges zwischen zwei Punkten.