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: 20.12.2023

II Inhaltsverzeichnis

II Inhaltsverzeichnis 2
III Abbildungsverzeichnis 3
IV Changelog 4
1 Projektdokumentation 5
1.1 Softwareprozess / Vorgehensmodell 5
1.2 Technologien und Tools 5
1.3 UML-Diagramm 6
2 Projektplan 7
2.1 Ziele, Umfang und angestrebtes Ergebnis des Projekts 7
2.2 Anvisierte Zielgruppe 8
2.3 Potenzielle Projektrisiken und Gegenmaßnahmen 8
2.4 Zeitplan und Meilensteine 8
2.5 GitHub-Repository 10
3 Anforderungsdokument 10
3.1 Management Summary 10
3.2 Systemumfang und Kontext 11
3.3 Funktionale Anforderungen 12
3.4 Nicht-funktionale Anforderungen 13
3.5 Glossar 14
4 Projektdokumentation 15
4.1 Requirements 15
4.2 Tools und Technologien 15
4.3 Aktivitätsdiagramm 18
4.4 Use Case Diagramm 19
5 Testprotokolle 20

III Abbildungsverzeichnis

Abbildung 1: UML-Diagramm 7
Abbildung 2: Aktivitätsdiagramm.....19
Abbildung 3: Use Case Diagramm.....19

IV Changelog

1. Abschnitt 2.1: Beschreibung des Spielkonzepts erweitert
2. Abschnitt 3.1: Spielwelt beschrieben sowie Möglichkeit von Erweiterungen benannt.
3. Abschnitt 3.1: Anpassung der Beschreibung zu den technischen Anforderungen sowie entfernen von Docker als Containerisierung, da dies in dieser Form für den Endbenutzer einen zusätzlichen vermeidbaren Aufwand im Rahmen der Installation darstellt.
4. Abschnitt 1.3: Objekt als Abgrenzung der Spielwelt bezeichnet
5. Abschnitt 2.4: Beschreibung des Wasserfallmodells erweitert
6. Diagramm 2 sowie Zugehörige Beschreibung entfernt
7. Abschnitt 3.3 vollständige Überarbeitung der funktionalen und nicht-funktionalen Anforderungen zwecks einer klaren Darstellung von Akzeptanzkriterien sowie den daraus resultierenden Nutzen

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 UML-Klassendiagramm, welches die Zusammensetzung des Programmcodes auf Klassenebene beschreibt. Hierbei wird ersichtlich, dass die Klasse „Game“ als der Mittelpunkt zu betrachten ist, welcher durch die main-Funktion ausgelöst wird.

Hierbei ruft die erstellt die Main-Funktion eine Instanz der Klasse Game und führt die run()-Funktion aus, welche die Spielschleife startet. Konkret kommen innerhalb der Game-Klasse, Instanzen und Funktionen der anderweitigen Klassen, wie bspw. Spieler, Gegner und Button etc. zum Einsatz.

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 (nach oben, unten, links und rechts) kann. Der Charakter wird zu Anfang von einem Verfolger gejagt, zu welchem in bestimmten zeitlichen abständen

weitere hinzukommen, und muss möglichst lange entkommen, ohne vom Gegner berührt zu werden. Entsprechend dessen lässt sich das Spiel-Konzept dahingehend beschreiben, dass der/die Spieler/in im Rahmen einer fest vorgegebenen Spielwelt sich bewegen kann und vor der Herausforderung steht, diese für einen möglichst langen Zeitraum erfolgreich für das Flüchten/Entrinnen vor den Verfolgern zu nutzen. Zusätzlich der letzte Highscores im Hauptmenu 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):

In dieser Phase liegt der Schwerpunkt auf dem Verständnis der Projektaufgaben und der Vorbereitung der Anforderungsdokumentation im Rahmen des Wasserfallmodells. Es werden das Management Summary, der Systemumfang und Kontext, sowie funktionale und nicht-funktionale Anforderungen gemäß den Vorgaben erstellt und eingereicht.

2. Technische Planung (2 Tag):

Während dieser Phase des Wasserfallmodells werden die notwendigen Tools und Frameworks installiert, darunter Python, Pygame, pathfinding, sys, random, SQLite, und GitHub. Eventuell

werden zusätzliche Recherchen eingeleitet und dokumentiert, um die technische Planung zu unterstützen.

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

Im Rahmen der UML-Diagrammerstellung wird die Struktur der Anwendung visualisiert. Komponenten und Abhängigkeiten werden dokumentiert, um einen klaren Überblick über die geplante Umsetzung zu erhalten.

4. Entwicklung der Spiellogik (5 Tage):

Die Umsetzung der Charaktersteuerung, die Programmierung der Objekte und Verfolgerlogik sowie die Implementierung der Highscore-Verwaltung finden in dieser Phase statt. Hier wird die Spiellogik gemäß den Anforderungen des Wasserfallmodells entwickelt.

5. Entwicklung der Benutzeroberfläche (1 Tage):

Die Gestaltung des Startbildschirms mit Start- und Exit-Button sowie die Anzeige der letzten drei Highscores erfolgt in dieser Phase. Die Implementierung der Spieloberfläche mit allen erforderlichen Anzeigeelementen wird ebenfalls durchgeführt.

6. Spieltest und Fehlerbehebung (2 Tage):

In dieser Phase werden ausgiebige Tests des Spiels durchgeführt, um mögliche Fehler zu identifizieren. Die Fehlerbehebung und die Optimierung der Leistung stehen im Mittelpunkt, um sicherzustellen, dass die Anwendung den Anforderungen entspricht.

7. Dokumentation (1 Tag):

Die Abschließende Erstellung und Aktualisierung der Projektdokumentation erfolgt in dieser Phase. Es wird überprüft, ob alle Dokumente vollständig und genau sind, um den Anforderungen des Wasserfallmodells gerecht zu werden.

8. Projektabschluss (1 Tag):

Die letzte Testphase und die Sicherstellung, dass das Projekt den Anforderungen entspricht, stehen im Fokus. Die Vorbereitung der finalen Abgabe und Präsentation schließt die Wasserfallmodell-Phasen ab.

Gesamtdauer: 14 Tage

2.5 GitHub-Repository

Nachfolgend finden sich der GitHub-Link zur Einsicht sowie der Repository Pull-Link für das benannte Softwareprojekt.

GitHub-Link: https://github.com/kemise/IU-Game

Repository Pull-Link: https://github.com/kemise/IU-Game.git

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 sich nach oben und unten sowie links und rechts bewegen kann. Eine aufregende Verfolgungsjagd entsteht, indem der Spieler von einem Verfolger gejagt wird und dabei geschickt die Spielwelt nutzt. Dies beruht darauf, dass die Spielwelt sich auf Basis eines starren Hintergrundes sowie einer fest codierten Matrix, welche die Objekte (1 und 0) in Form von Blöcken beinhaltet, vorgehalten wird. Diese Form der Bereitstellung der Spielkarte, ermöglicht es, im Rahmen einer gegebenenfalls späteren Weiterentwicklung des Spiels, einen Kartengenerator und/oder weitere, aus einem Pool vorgehaltener Karten, Spielewelten zur Verfügung zu Stellen.

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. Für die Sicherung und Score-Verwaltung findet SQLite Verwendung. Des Weiteren findet pathfinding, in Form des A*-Algorithmus, für die Implementierung der Lauflogik des Gegners Verwendung. Zusätzlich wird im Rahmen der Entwicklung darauf Wert gelegt, dass etwaigen Nutzern eine schnelle und einfache Verwendung des Spiels ermöglicht wird. Hierfür wird im Rahmen der Entwicklung dafür Sorge getragen, dass das Ausführen des Spiels mittels einer BAT-Datei ermöglicht wird. Ein Weitere wichtiger Aspekt besteht darin, dass die entsprechende BAT-Datei vor dem Starten des Spiels, die für die Anwendung erforderlichen

Bestandteile prüft. Hierbei wird abgefragt, ob Python installiert ist und sofern nicht, ein entsprechender Hinweis ausgegeben. Des Weiteren werden die zusätzlichen Third-Party-Libraries automatisch, bei nicht vorhanden sein, installiert. Aufgrund dessen kann festgehalten werden, dass für den potenziellen Spieler, lediglich die Installation der korrekten und in der Spielbeschreibung benannten Python-Version sowie das Starten des Spiels via der BAT-Datei erforderlich ist.

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.

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

3.3 Funktionale Anforderungen

1. Charakterbewegungen:
    a. Akzeptanzkriterien: Der Charakter muss sich reibungslos vorwärts, rückwärts und seitlich bewegen können, und die Bewegungen sollten auf die Spielersteuerung ansprechend reagieren.
    b. Nutzen: Die Möglichkeit der vielseitigen Charakterbewegung erhöht die Spielerfreiheit-
2. Algorithmus-gesteuerter Verfolger:
    a. Akzeptanzkriterien: Der Verfolger muss den Spieler aktiv jagen und dabei intelligente Entscheidungen basierend auf dem Spielerstandort treffen.
    b. Nutzen: Schafft eine dynamische und herausfordernde Spielumgebung, indem ein intelligenter Verfolger für Spannung und Nervenkitzel sorgt, während der Spieler versucht, ihm zu entkommen.
3. Reaktion auf Bewegungen des Spielers:
    a. Akzeptanzkriterien: Der Verfolger soll in Echtzeit auf die Bewegungen des Spielers reagieren und seine Route entsprechend anpassen.
    b. Nutzen: Erhöht die Interaktivität des Spiels, da der Verfolger als reaktionsfähiges Element agiert und den Spieler dazu zwingt, taktische Entscheidungen zu treffen, um erfolgreich zu sein.
4. Direkter und schnellster Weg während der Verfolgung:
    a. Akzeptanzkriterien: Der Verfolger wählt stets den direkten und schnellsten Weg, um den Spieler zu jagen.

   b. Nutzen: Intensiviert die Verfolgung und sorgt für eine höhere Schwierigkeitsstufe, wodurch der Spieler ständig herausgefordert wird.
5. „Game Over“ bei Berührung durch den Gegner:
   a. Akzeptanzkriterien: Das Spiel endet mit einem "Game Over", wenn der Charakter von einem Gegner berührt wird.
   b. Nutzen: Stellt eine klare Regel für den Misserfolg auf.
6. Beendigung des Spiels durch den Spieler:
   a. Akzeptanzkriterien: Der Spieler kann das Spiel über den Start-Menü-Button oder das Rote-Kreuz im Spiel verlassen.
   b. Nutzen: Bietet dem Spieler Kontrolle über seine Spielerfahrung und ermöglicht eine einfache Navigation im Spiel.
7. Anzeige des Fortschritts:
   a. Akzeptanzkriterien: Das Spiel soll eine klare Anzeige des Fortschritts haben, die im Rahmen des "Game Over" ausgegeben wird.
   b. Nutzen: Motiviert den Spieler durch visuelles Feedback und zeigt den Erfolg seiner Bemühungen an.
8. Wiederaufnahme des Spiels nach "Game Over":
   a. Akzeptanzkriterien: Der Spieler kann das Spiel nach einem "Game Over" erneut starten.
   b. Nutzen: Erhöht die Wiederspielbarkeit und fördert die Motivation des Spielers, indem er die Möglichkeit hat, sein Glück erneut zu versuchen.
9. Musikalischer Unterschied bei "Game Over":
   a. Akzeptanzkriterien: Im Rahmen des "Game Over" erfolgt ein musikalischer Unterschied im Vergleich zum bisherigen Spiel-Sound.
   b. Nutzen: Schafft eine Differenzierung und verstärkt den Einfluss des Spielers auf die Spielererfahrung.
10. Musikalische Begleitung im Spiel-Menü und während des Spiels:
    a. Akzeptanzkriterien: Das Spiel-Menü und das Spiel selbst sollen jeweils eine eigenständige musikalische Begleitung haben.
    b. Nutzen: Steigert die Atmosphäre des Spiels und ermöglicht eine musikalische Abgrenzung des aktiven Spiels und des Menüs.

3.4 Nicht-funktionale Anforderungen

1. Flüssige Spielerfahrung und Framerate von mindestens 30 FPS:

   a. Akzeptanzkriterien: Das Spiel muss auf verschiedenen Hardwarekonfigurationen eine Framerate von mindestens 30 FPS erreichen und eine insgesamt flüssige Spielerfahrung bieten.
   b. Nutzen: Gewährleistet eine angenehme Spielerfahrung, indem es sicherstellt, dass das Spiel reibungslos läuft, unabhängig von der Hardware des Spielers.
2. Intuitive und reaktionsschnelle Steuerung des Charakters:
   a. Akzeptanzkriterien: Die Steuerung des Charakters muss intuitiv sein und auf Spielerbefehle reaktionsschnell reagieren.
   b. Nutzen: Verbessert die Spielerfreundlichkeit und ermöglicht es dem Spieler, sich nahtlos durch das Spiel zu bewegen, Aktionen präzise auszuführen und eine direkte Kontrolle über den Charakter zu haben.
3. Ansprechende und visuell ansprechende Spielumgebung:
   a. Akzeptanzkriterien: Die Spielumgebung muss visuell mit hochwertigen Grafiken ansprechend gestaltet sein.
   b. Nutzen: Erhöht die Qualität des Spiels und fördert die Spielerbindung sowie den Unterhaltungswert.
4. Leicht verständliche Anweisungen für den Spieler:
   a. Akzeptanzkriterien: Das Spiel und/oder die Anleitung sollen klare und leicht verständliche Anweisungen bieten, um den Spielern ein angenehmes Spielerlebnis zu ermöglichen.
   b. Nutzen: Verbessert die Benutzerfreundlichkeit, indem es sicherstellt, dass der Spieler das Spiel ohne Verwirrung spielen kann

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.

4 Projektdokumentation

Die nachfolgenden Kapitel beinhalten die eine Beschreibung zu den verwendeten Tools und Technologien sowie eine Beschreibung zum Quellcode.

4.1 Requirements

Für die Entwicklung des 2D Spieles SAND, findet die Programmiersprache Python in der Version 3.11.1 Verwendung. Entsprechend dessen ist erforderlich, die benannte Version, unter Berücksichtigung des eigenen Systems, von offizieller Stelle (siehe readme.md) zu beziehen sowie fachgerecht zu installieren. Da sich die technischen Details dieser Dokumentation an entsprechend fachkundige Leser und Leserinnen richtet, wird an dieser Stelle auf eine detaillierte Erklärung der einzelnen Installationsschritte verzichtet und auf die offizielle Dokumentation von Python verwiesen.

Eine weitere Voraussetzung, welche es für die Entwicklung dieses Projektes zu benennen gilt, ist die Verwendung von Visual Studio Code als Entwicklungsumgebung (IDE). Hierbei handelt es sich um eine kostenfrei IDE, welche aus dem Hause Microsoft stammt und mittel der Python-Erweiterung eingesetzt wird.

4.2 Tools und Technologien

Das Softwareprojekt, zur Erstellung eines 2D Spiels wurde in Python mittels der Pygame-Bibliothek entwickelte. Pygame ist ein Open-Source-Framework für die Entwicklung von Videospielen in Python. Zudem findet pathfinding für die optimierte Wegfindung des Gegners sowie SQLite für die Scoreverwaltung Verwendung

Nachfolgend werden die wichtigsten Tools/Technologien und Methoden aufgeführt und in kurzer Form erörtert.

Pygame-Bibliothek

Die Pygame-Bibliothek wird für die Erstellung von 2D-Spielen in Python verwendet. Sie bietet Funktionen zur Steuerung von Grafiken, Ton und Benutzereingaben.

Pathfinding-Bibliothek

Die pathfinding-Bibliothek wird für den A*-Algorithmus zur Wegfindung verwendet. Dieser Algorithmus ermöglicht es dem Gegner, den Spieler in der Spielwelt zu verfolgen.

pygame.mixer

Die pygame.mixer-Bibliothek wird für die Soundeffekte im Spiel verwendet. Es werden verschiedene Soundeffekte für das aktive Spielen, das Spielende und das Hauptmenü bereitgestellt.

Klassen und Objekte:

Das Spiel SAND wurde mittels eines Codes, welcher die objektorientierte Programmierung berücksichtigt entwickelt. Dies führt auf Grund von Klassen, wie Game, Player, Enemy, ScoringHandler und Button zu einer strukturierten Organisation, verbesserten Wartbarkeit und Lesbarkeit des Codes. Entsprechend dessen ist es etwaigen Entwicklern im Nachgang möglich, Änderungen und/oder Erweiterungen im Rahmen eines kalkulierbaren Aufwandes vorzunehmen.

Zufallszahlen

Das random-Modul wird verwendet, um zufällige Positionen für den Feind zu generieren. Dieses Modul ermöglich es somit den Gegner auf zufälligen Positionen innerhalb des Spielfeldes generiert zu werden, was zu einer erhöhten Abwechslung und Unvorhersehbarkeit führt.

Text-Rendering

Die Pygame-Funktionen render und blit werden für das Rendern der jeweiligen Texte mit ihrer spezifischen Größe und Farbe auf dem Bildschirm verwendet, hierfür lässt sich beispielsweise der Anzeigetext am Spielende benennen.

Spieldesign und Logik

Das Spiel wurde mit einer umfangreichen Spiellogik ausgestattet, welche das Verhalten des Spiels steuert. Hierzu gehört die Kollisionserkennung zwischen verschiedenen Spielelementen wie dem Spieler, dem Gegner und der Umgebung. Die Kollisionserkennung ist entscheidend, um festzustellen, ob der Spieler mit Hindernissen in Form von Blöcken oder einem Feind in Berührung kommt.

Die Steuerung des Spielablaufs beinhaltet Funktionen zur Initialisierung des Spiels, zum Zeichnen der Spielwelt und zur Aktualisierung des Bildschirms. Durch die Aktualisierung des

Bildschirms wird zudem sichergestellt, dass Änderungen im Spielgeschehen dem Spieler in Echtzeit angezeigt werden.

Game-Klasse (game_handler.py):

Die Game-Klasse stellt die Hauptklasse des Spiels dar, welche die unter anderem die Spiellogik steuert. Sie initialisiert und verwaltet verschiedene Elemente wie Spieler, Gegner, Punkte, Anzeigeeinstellungen und verarbeitet Benutzereingaben. Hierbei führt die run-Methode die Hauptschleife des Spiels aus und aktualisiert den Spielzustand. Im Rahmen der Implementierung findet das State Design Pattern Verwendung. Dies wird zur Steuerung des Spielzustandes mittels self.game_started verwendet.

Button-Klasse (buttons.py):

Mittels der Button-Klasse wird werden die Erstellung sowie Bereitstellung von Schaltflächen im Spiel gesteuert. Des Weiteren ermöglicht die Klasse das Anzeigen von interaktiven Schaltflächen auf dem Bildschirm und die Überprüfung von Benutzerinteraktionen. Zu den jeweiligen Benutzerinteraktionen zählen beispielsweise Aktionen, wie das Starten eines neuen Spiels.

Player-Klasse (player.py):

Die Player-Klasse repräsentiert die Spielfigur und steuert deren Bewegung sowie Animation. Die Klasse enthält Methoden zur Handhabung von Spieleraktionen wie Bewegungen und Kollisionen mit Objekten. Des Weiteren ermöglicht sie ebenfalls die Aktualisierung der Spielerposition basierend auf Benutzereingaben.

ScoringHandler-Klasse (scoring.py):

Auf Basis der ScoringHandler-Klasse erfolgt die Verwaltung der Spielerpunkte. Hierzu zählen unter anderem das Speichern und Abrufen von Spielerpunkten in einer SQLite-Datenbank.

Enemy-Klasse (enemy.py):

Mit der Enemy-Klasse erfolgt die Erstellung der feindlichen Figuren im Spiel. Die Klasse enthält Methoden zur Berechnung der Gegnerbewegungen, Kollisionserkennung mit dem Spieler und zur Aktualisierung ihrer Positionen. Sie stellt somit eine Schlüsselrolle bei der Schaffung von Herausforderungen im Spiel dar.

Das Spiel "SAND" kombiniert die benannten Technologien, um eine einfache, aber unterhaltsame Spielerfahrung zu bieten. Es wird somit demonstriert, dass die Anwendung von unter anderem Pygame-Funktionalitäten, pathfinding sowie sqlite für die Spieleentwicklung in Python geeignet ist.

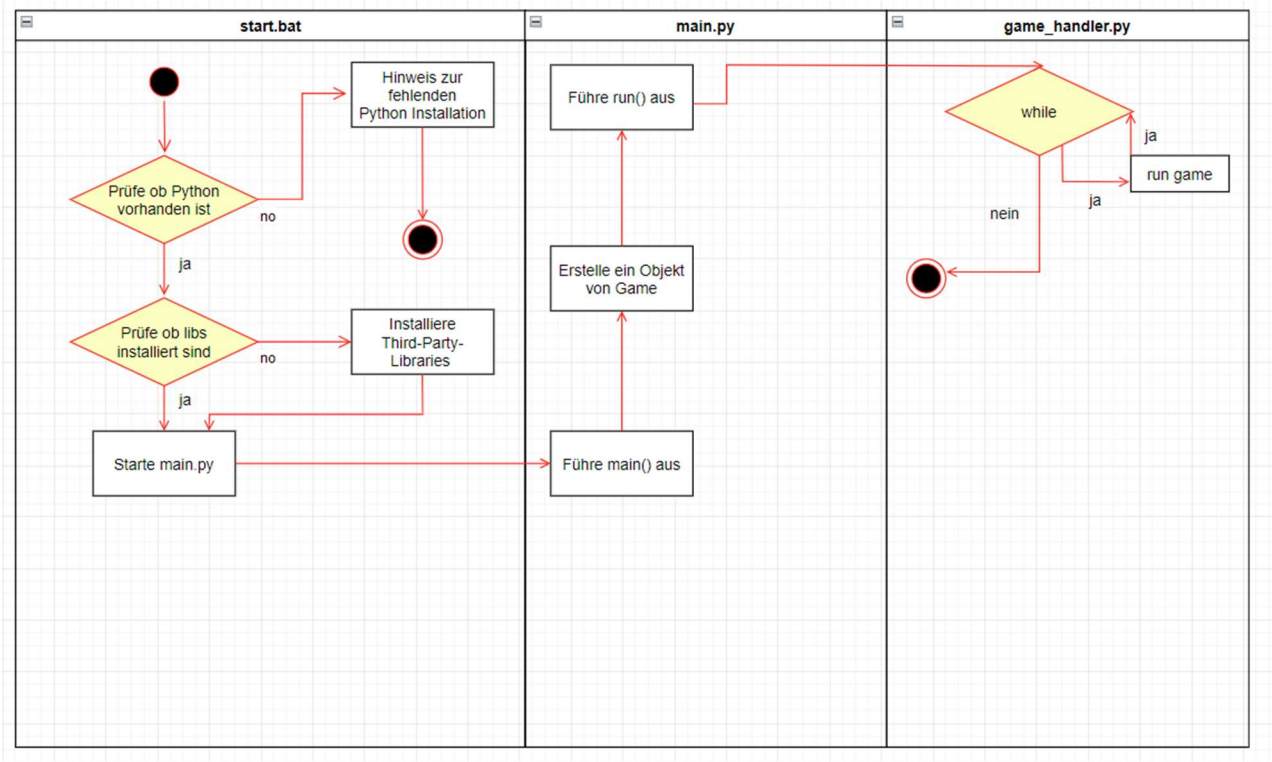
4.3 Aktivitätsdiagramm

In der Nachfolgenden Abbildung findet sich ein Aktivitätsdiagramm wieder, welches einen Überblick über den Programmablauf ermöglicht.

Grundsätzlich lässt sich dieser wie folgt in Form einer überschaubaren Erklärung darlegen.

1. Zu Anfangs wird die start.bat Datei ausgeführt, welche den initialen Start einleitet
    a. Im Rahmen der Ausführung wird überprüft, ob Python vorhanden ist und bei nicht Vorhandensein eine Meldung mit der Bitte um Installation sowie einen entsprechenden Download-Link ausgespielt. Anschließend ende das Programm.
    b. Sofern Python vorhanden ist, wird überprüft, ob die notwendigen Third-Party-Libraries vorhanden sind. Bei einem nicht Vorhandensein, werden die die benötigten Third-Party-Libraries automatisch installiert
2. Anschließend wird die main.py gestartet, welche ein Objekt der Game-Klasse instanziiert sowie die run()-Funktion ausführt.
3. Auf Basis der run()-Funktion wird die Hauptspiellogik in Form einer While-Schleife gestartet und bis zu einem negieren aktiv ausgeführt.
    a. Ein Grund für ein negieren der Schleife kann hierbei ein Event, wie beispielsweise das Auslösen des Exit-Buttons sein

Abbildung 2: Aktivitätsdiagramm

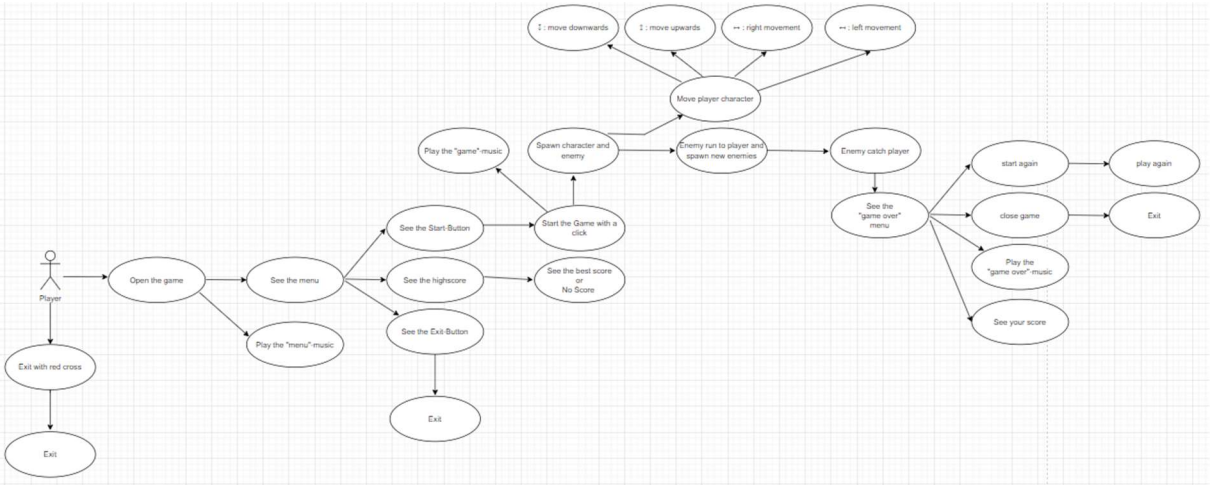


Quelle Bild: eigene Darstellung (2023)

4.4 Use Case Diagramm

Im Rahmen der anschließenden Abbildung erhält die lesende Person eine übersichtliche Darstellung über die jeweiligen Funktionen/Anforderungen des Spiels.

Abbildung 3: Use Case Diagramm



Quelle Bild: eigene Darstellung (2023)

5 Testprotokolle

Nachfolgend finden sich tabellarisch aufgeführte Testprotokelle zu den jeweiligen funktonalen Anforderungen.

Testfall-ID	SG-01
Datum	18.12.2023
Beschreibung	Überprüfung der reibungslosen Charakterbewegungen in verschiedene Richtungen.
Vorbedingungen	Das Spiel ist geöffnet sowie gestartet und der Charakter steht bereit.
Eingabedaten/Handlungen	Der Spieler bewegt den Charakter vorwärts, rückwärts und seitlich.
Erwartetes Ergebnis	Der Charakter bewegt sich ohne Verzögerungen in alle Richtungen.
Tatsächliches Ergebnis	Der Charakter bewegt sich reibungslos in alle Richtungen, die Spielersteuerung reagiert wie erwartet.

Testfall-ID	SG-02
Datum	18.12.2023
Beschreibung	Überprüfung der Verfolgung durch den Algorithmus-gesteuerten Verfolger.
Vorbedingungen	Das Spiel ist geöffnet sowie gestartet und der Gegner steht bereit.
Eingabedaten/Handlungen	Der Spieler versucht dem Verfolger zu entkommen, indem er sich bewegt.
Erwartetes Ergebnis	Der Verfolger trifft intelligente Entscheidungen, um den Spieler auf dem kürzesten Weg zu jagen.
Tatsächliches Ergebnis	Der Verfolger jagt den Spieler aktiv und trifft Entscheidungen hinsichtlich seiner Wegwahl basierend auf dem Spielerstandort.

Testfall-ID	SG-03
Datum	18.12.2023

Beschreibung	Überprüfung, ob das Spiel mit einem "Game Over" endet, wenn der Charakter vom Gegner berührt wird.
Vorbedingungen	Das Spiel ist geöffnet sowie gestartet und der Charakter sowie Gegner befinden sich im Spielfeld
Eingabedaten/Handlungen	Der Charakter wird vom Gegner berührt.
Erwartetes Ergebnis	Das Spiel endet mit einem klaren "Game Over".
Tatsächliches Ergebnis	Das Spiel endet wie erwartet mit einem "Game Over", wenn der Charakter vom Gegner berührt wird.

Testfall-ID	SG-04
Datum	18.12.2023
Beschreibung	Überprüfung der Spielbeendigung durch den Spieler über den Start-Menü-Button, das Rote-Kreuz im Spiel oder im Rahmen der „Game Over“ Sequenz.
Vorbedingungen	Das Spiel läuft, der Spieler befindet sich im Spiel.
Eingabedaten/Handlungen	Spieler beendet das Spiel einmal im Rahmen der „Game Over“ Sequenz via der vorgesehenen Eingabe, einmal über den im Start-Menü vorhandenen Button sowie einmal über das Rote-Kreuz innerhalb Spieles.
Erwartetes Ergebnis	Das Spiel wird ordnungsgemäß ohne Verzögerung oder Fehlermeldungen geschlossen.
Tatsächliches Ergebnis	Der Spieler kann das Spiel erfolgreich über die benannten Wege beenden.

Testfall-ID	SG-05
Datum	18.12.2023
Beschreibung	Überprüfung der Anzeige des Fortschritts im Rahmen der "Game Over" Sequenz.
Vorbedingungen	Das Spiel wurde gespielt und endet mit einem "Game Over".
Eingabedaten/Handlungen	Spieler betrachtet die Fortschrittsanzeige im "Game Over"-Bildschirm.
Erwartetes Ergebnis	Das Spiel visualisiert den Punktestand der gespielten Runde.
Tatsächliches Ergebnis	Der Punktestand wird korrekt im "Game Over"-Bildschirm angezeigt.

Testfall-ID	SG-06
Datum	18.12.2023
Beschreibung	Überprüfung, ob der Spieler das Spiel nach einem "Game Over" erneut starten kann.
Vorbedingungen	Das Spiel endet mit einem "Game Over".
Eingabedaten/Handlungen	Spieler tätigt die notwendige Eingabe zur erneuten Aufnahme des Spiels nach dem "Game Over".
Erwartetes Ergebnis	Das Spiel startet erneut und der Spieler kann eine neue Runde spielen.
Tatsächliches Ergebnis	Der Spieler kann das Spiel erfolgreich nach einem "Game Over" erneut spielen.

Testfall-ID	SG-07
Datum	18.12.2023
Beschreibung	Überprüfung, ob im Rahmen des "Game Over" ein musikalischer Unterschied im Vergleich zum bisherigen Spiel-Sound erfolgt.
Vorbedingungen	Das Spiel endet mit einem "Game Over".
Eingabedaten/Handlungen	Der Spieler hört auf den musikalischen Unterschied beim "Game Over".
Erwartetes Ergebnis	Der Sound beim "Game Over" unterscheidet sich klar vom bisherigen Spiel-Sound.
Tatsächliches Ergebnis	Im Rahmen des "Game Over" erfolgt, wie erwartet, ein musikalischer Unterschied.

Testfall-ID	SG-08
Datum	18.12.2023
Beschreibung	Überprüfung der musikalischen Begleitung im Spiel-Menü und während des Spiels.
Vorbedingungen	Das Spiel läuft, der Spieler befindet sich im Menü und startet anschließend das Spiel.

Eingabedaten/Handlungen	Der Spieler hört die musikalische Begleitung im Menü, startet das Spiel und hört erneut auf die musikalische Begleitung während des Spiels.
Erwartetes Ergebnis	Das Spiel-Menü und das aktive Spiel weisen jeweils eine eigenständige musikalische Begleitung auf.