Projektplan – Projekt Software Engineering

Erstellt von: Jan Wunderlich
Matrikelnummer: 92205002
Studiengang: Master of Science Informatik
Tutor: Prof. Dr. Markus Kleffmann
Datum: 23.11.2023

Inhaltsverzeichnis

1 Ziele, Umfang und angestrebtes Ergebnis des Projekts

Das Ziel des Projekts besteht in der Entwicklung eines sogenannten Endless-Runner- oder Infinite-Runner-Spiels. In diesem Genre übernimmt der Spieler die Kontrolle über eine Figur, die ununterbrochen läuft, und versucht, so lange wie möglich zu überleben. Dies erreicht er, indem er geschickt sämtlichen Hindernissen ausweicht und auftauchende Gegner bezwingt. Der Spieler verliert, sobald die Spielfigur von einem Hindernis oder Gegner getroffen wird. In diesem Moment endet das Spiel bzw. der aktuelle Versuch. Folgende Komponenten sind integraler Bestandteil des Projekts und sollen in der finalen Anwendung nahtlos integriert sein:

1. Grundgerüst:
    - Initialisierung des Spiels durch Pygame, Erschaffung eines Fensters und Laden aller erforderlicher Ressourcen, einschließlich Grafiken für Hintergrund, Spieler, Gegner, Power-Ups und mehr.
    - Das Design soll über alle Spielelemente hinweg, wie den Hintergrund, die Spieler, die Gegner und das Menü im Cyberpunk-Style gehalten werden.
    - Definition von Kernparametern wie die Scrollgeschwindigkeit des Hintergrunds, die Sprunghöhe des Spielers, die Schussgeschwindigkeit und die Häufigkeit von Hindernissen oder Gegnern.
2. Spieler:
    - Spielerbewegungen einschließlich links, rechts, springen und ducken oder sliden ermöglichen.
    - Umsetzung der Schussmechanik, wobei die Möglichkeit geschaffen wird, mehrere Kugeln gleichzeitig abzufeuern, welche ein Spieler mithilfe eines Power-Ups freischalten kann.
3. Gegner:
    - Zufällige Erzeugung von Gegnern auf der rechten Seite des Bildschirms.
    - Definition und Implementierung von unterschiedlichen Fähigkeiten für Gegner wie Laufen, Springen oder das Abfeuern von Waffen.
4. Power-Ups und Items:
    - Zufällige Generierung von Power-Ups, die der Spieler einsammeln kann.
    - Power-Ups bieten verschiedene Effekte, wie ein zusätzliches Leben oder verbesserte Waffen.
5. Spiellogik und Fortschrittsanzeige:
    - Anzeige von zurückgelegten Metern sowohl während des Spiels als auch im Endbildschirm mithilfe eines Meterzählers.
    - Umsetzung einer Highscore-Verwaltung, um Spielerergebnisse zu speichern und zu aktualisieren.

- Spielende-Logik, um ein Spiel zu beenden, wenn der Spieler von einem Gegner getroffen wird oder gegen ein Hindernis läuft (Kollisionserkennung).

6. Startbildschirm und Optionen:
   - Hauptmenü-Bildschirm, über den der Spieler das Spiel starten kann.
   - Integration von Optionen zur Steuerung der Musik und Soundeffekten.
7. Benutzerinteraktion:
   - Tastatursteuerung, um Interaktion mit der Spielfigur zu realisieren.
   - Maussteuerung für die Navigation im Spielmenü.
8. Spielneustart, -pause und -beendigung:
   - Option zum Neustart des Spiels nach dem Tod des Spielers.
   - Spieler können Spiel pausieren und zum Hauptbildschirm zurückkehren.

Diese Komponenten bilden den groben Umfang des zu entwerfenden Endless-Runner-Spiels und können im Laufe des Projekts, sollte ein entsprechender Bedarf erkannt werden, ggf. um weitere Elemente ergänzt werden. Das angestrebte Ergebnis dieses Software Engineering Projekts ist die Entwicklung einer voll funktionsfähigen, ansprechenden und unterhaltsamen Endless-Runner-Anwendung, die alle definierten Komponenten umfasst sowie dem Nutzer ein fesselndes Spielerlebnis beschert.

2 Anvisierte Zielgruppe

Bei der Definition der Zielgruppe für das Endless-Runner-Spiel sind mehrere entscheidende Überlegungen anzustellen, die maßgeblich in die Planung und Entwicklung des Spiels einfließen. Neben der Altersgruppe, die mit dem Spiel angesprochen wird, spielen auch die Interessen und Vorlieben der Nutzer eine besonders wichtige Rolle. Ebenso darf die gewünschte Spielerfahrung nicht außer Acht gelassen werden.

Obwohl das Endless-Runner-Spiel grundsätzlich von einem vielfältigen Publikum gespielt werden kann, richtet es sich primär an Gamer und Personen, die zumindest gelegentlich Computer- oder Videospiele nutzen. Die Hauptzielgruppe sind folglich Jugendliche und junge Erwachsene. Die Spielmechanik und -logik werden bewusst so gestaltet, dass das Spiel leicht zugänglich ist, aber dennoch eine gewisse Herausforderung bietet. Eine progressiv zunehmende Schwierigkeit ermöglich es sowohl Einsteigern als auch erfahrenen Spielern, sich auf ihrem eigenen Niveau zu verbessern und Fortschritte zu erzielen.

Das im Cyberpunk-Style gehaltene Design des Spiels spricht darüber hinaus insbesondere Science-Fiction-Fans an. Eine futuristische Ästhetik, gepaart mit innovativen Spielelementen, bietet eine einzigartige Spielerfahrung. Die spielerischen, visuellen und inhaltlichen Merkmale der Anwendung werden dabei gezielt ausgerichtet, um die Interessen der beschriebenen Zielgruppe bestmöglich zu bedienen.

3 Potenzielle Projektrisiken und Gegenmaßnahmen

Für die Realisierung eines adäquaten Risikomanagements sind drei aufeinanderfolgende Schritte erforderlich. Zunächst erfolgt im Rahmen der Risikoidentifikation eine umfassende Analyse, um potenzielle Risiken während der Projektdurchführung zu identifizieren. Anschließend erfolgt eine Risikobewertung, die unter anderem die angenommene Eintrittswahrscheinlichkeit sowie die möglichen Auswirkungen des Risikos auf Kosten oder Ergebnisqualität ermittelt. Der letzte Schritt besteht in der Maßnahmenplanung, bei der geeignete Gegenmaßnahmen zur Prävention oder Korrektur der Risiken entwickelt werden. Die Ergebnisse dieser Analyse werden in einem Risikoinventar zusammengefasst, das in der nachfolgenden Tabelle abgebildet ist. Die Risikokennzahl setzt sich dabei aus dem Produkt der Eintrittswahrscheinlichkeit und der Schadenshöhe des jeweiligen Risikos zusammen. Die Gegenmaßnahmen sind in 2 Arten, nämlich präventive (p) und korrektive (k) Maßnahmen, untergliedert.

Name des Risikos	Tragweite	Schadenshöhe (1-5)	Eintrittswahrscheinlichkeit (1-5)	Risikokennzahl	Maßnahmen (k oder p)
Technische Komplexität	Schwierigkeiten bei der Implementierung	4	3	12	Entwicklungsprozess in kleine Komponenten unterteilen (p), Funktionalitäten priorisieren und ggf. Abstriche machen (k)
Ressourcenknappheit	Verzögerungen bei Implementierung oder Dokumentation	2	2	4	Effiziente Planung und Priorisierung der Aufgaben (p), Projektdauer verlängern (k)
Unvorhergesehene Änderungen im Projektumfang	Zusätzlicher Arbeitsaufwand und Zeitbedarf	3	4	12	Detaillierte Anforderungsanalyse (p), flexible Anpassung der Anforderungen / Aufgaben (k)
Qualitätsprobleme	Fehler in der Anwendung und ggf. Leistungseinbußen	2	2	4	Einzelnen Funktionen testen (p), intensive Fehleranalyse und -korrektur (k)
Ungenügende Planung	Unklare Ziele und Projektverzögerungen	3	2	6	Durchführung einer gründlichen Projektplanung (p), flexibles Projektmanagement (k)
Ungenügende Testabdeckung	Unentdeckte Fehler und Qualitätsprobleme	4	1	4	Festlegung klarer Testziele und -kriterien (p), Erweiterung der Testabdeckung (k)

Tabelle 1: Risikoinventar

4 Zeitplan und Meilensteine

Die Erstellung eines klaren und verständlichen Zeitplans mitsamt Meilensteinen ist entscheidend für den Erfolg eines jeden Projekts. Um diese Komplexität zu bewältigen und den zeitlichen Ablauf optimal zu planen, wird in diesem Software Engineering Projekt auf ein Gantt-Diagramm zurückgegriffen. Dieses visuelle Instrument ermöglicht nicht nur eine übersichtliche Präsentation der Projektzeitleiste, sondern auch eine gezielte Identifikation von kritischen Meilensteinen sowie Abhängigkeiten und ist aus Übersichtlichkeitsgründen auf der folgenden Seite im Querformat abgebildet.

Bei genauerer Betrachtung des Gantt-Diagramms wird deutlich, dass für die Implementierung des Quellcodes in der Erarbeitungs- und Reflexionsphase zwar eine grobe Zeitspanne vorgesehen ist, jedoch keine spezifischen Startdaten, Dauerangaben oder Priorisierungen für einzelne Komponenten festgelegt sind. Dies resultiert aus der Entscheidung für ein agiles Vorgehensmodell in der Entwicklung, bei dem die exakte Reihenfolge und Priorisierung der Funktionen innerhalb des geplanten Zeitrahmens erst im Verlauf des Projekts festgelegt wird. Weitere Details zu diesem Vorgehensmodell werden in der Projektdokumentation ausführlich erläutert.

Projekt: Software Engineering

Endless-Runner-Game

Projektanfangsdatum: 13.11.23

Meilensteinbeschreibung	Fortschritt	Start	Tage
Konzeptionsphase			
Projektplan erstellen	100%	13.11.23	10
Anforderungsdokument erstellen	100%	13.11.23	10
Projektdokumentation erstellen (erste Version)	100%	13.11.23	10
Erarbeitungs- und Reflexionsphase			
GitHub Projekt einrichten	0%	04.12.23	1
Tools, Libraries aufsetzen und konfigurieren	0%	04.12.23	1
Quellcode schreiben	0%	05.12.23	20
Grundgerüst	0%	?	?
Spieler	0%	?	?
Benutzerinteraktion / Steuerung	0%	?	?
Gegner	0%	?	?
Spiellogik und Fortschrittsanzeige	0%	?	?
Power-Ups und Items	0%	?	?
Startbildschirm und Optionen	0%	?	?
Spielneustart, -pause und -beendigung	0%	?	?
Design Pattern in Projektdokumentation ergänzen	0%	18.12.23	1
2 UML-Diagramme erstelllen und in Projektdokumentation aufnehmen	0%	19.12.23	2
Testfälle aufstellen, durchführen und Testprotokoll schreiben	0%	22.12.23	2
Finalisierungsphase			
Anwendung fertigstellen	0%	30.12.23	5
Dokumentationen aktualisieren / fertigstellen	0%	03.01.24	7
Benutzeranleitung schreiben	0%	07.01.24	1
Abstract schreiben	0%	09.01.24	2
Puffer	0%	10.01.24	7

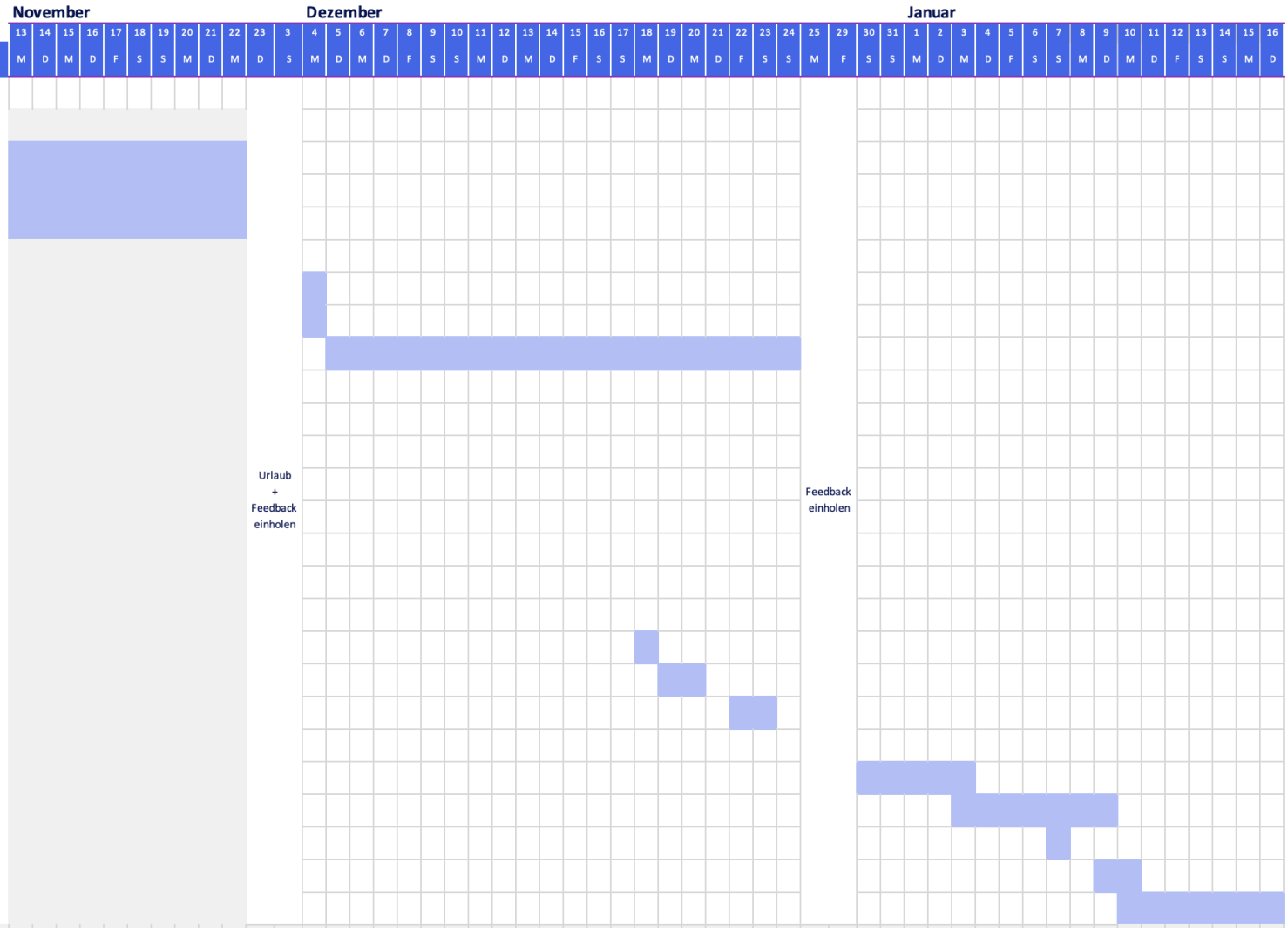

Abbildung 1: Zeitplanung als Gantt-Diagramm

Anforderungsdokument – Projekt Software Engineering

Erstellt von: Jan Wunderlich
Matrikelnummer: 92205002
Studiengang: Master of Science Informatik
Tutor: Prof. Dr. Markus Kleffmann
Datum: 23.11.2023

Inhaltsverzeichnis

1 Management Summary

Dieses Software Engineering Projekt konzentriert sich auf die Entwicklung eines fesselnden Endless-Runner-Spiels, bei dem der Spieler das Ziel verfolgt, eine möglichst große Distanz zu überwinden, ohne von Hindernissen oder Gegnern getroffen zu werden. Das Herzstück des Spiels liegt in seinem ansprechenden Spielerlebnis, welches die Nutzer dazu ermutigt, ihre eigenen Fähigkeiten zu verbessern und ihren Highscore immer wieder zu übertreffen.

Die Funktionalitäten des Spiels erstrecken sich über verschiedene Aspekte. Der Spieler hat die Möglichkeit, die Spielfigur präzise zu steuern, geschickt Hindernissen auszuweichen und Gegner durch den Einsatz einer Waffe zu besiegen. Das Sammeln unterschiedlichster Power-Ups während des Spiels eröffnet dem Spieler zusätzliche Strategien und Möglichkeiten, seinen aktuellen Versuch zu optimieren. Während eines laufenden Versuchs erhält der Spieler kontinuierlich Rückmeldung über seinen Fortschritt, angezeigt in Form der zurückgelegten Meter und gesammelten Punkte. Diese transparente Darstellung ermöglicht es dem Spieler, seine Leistung zu überwachen und den Anreiz zu schaffen, stetig besser zu werden.

Abseits des eigentlichen Spielgeschehens bietet das übersichtliche Hauptmenü dem Spieler eine Vielzahl von Interaktionsmöglichkeiten. Hier kann er nicht nur seine letzten Läufe und Statistiken einsehen, sondern auch die Laustärke von Ton und Musik anpassen. Zusätzlich hat der Spieler die Option, gesammelte Punkte gegen temporäre Power-Ups einzutauschen, die über mehrere Runden hinweg wirksam sind. Ebenso startet der Spieler von hier unkompliziert ein neues Spiel respektive einen neuen Versuch.

In Ergänzung zu den funktionalen Anforderungen liegt ein besonderer Fokus auf nicht-funktionalen Aspekten wie Leistung, Zuverlässigkeit, Ladezeiten und Benutzeroberfläche. Das Ziel ist es, den Nutzern eine umfassende und zufriedenstellende Spielerfahrung zu bieten, die durch hohe Qualität in der Benutzerinteraktion, eine intuitive Benutzeroberfläche und reibungslose Performance geprägt ist. Die Entwicklung zielt darauf ab, ein durchweg ansprechendes und unterhaltsames Spielerlebnis zu garantieren, das den Erwartungen der Spieler gerecht wird.

2 Systemumfang und Kontext

Die Anwendung eines UML-Anwendungsfalldiagramms stellt eine effektive Methode dar, um die Abgrenzungen eines Systems zu definieren und den Umfang des Systems auf prägnante Weise zu veranschaulichen. Die klare visuelle Strukturierung von Akteuren und deren Interaktion mit den verschiedenen Anwendungsfällen schafft Transparenz über die Funktionen und Möglichkeiten, die das System bietet. In der nachfolgenden Abbildung ist daher ein UML-Anwendungsfalldiagramm dargestellt, das die Kernaspekte des zu entwerfenden Endless-Runner-Spiels umfasst und dementsprechend Auskunft über die möglichen Aktionen, die der Spieler ausführen kann, gibt.

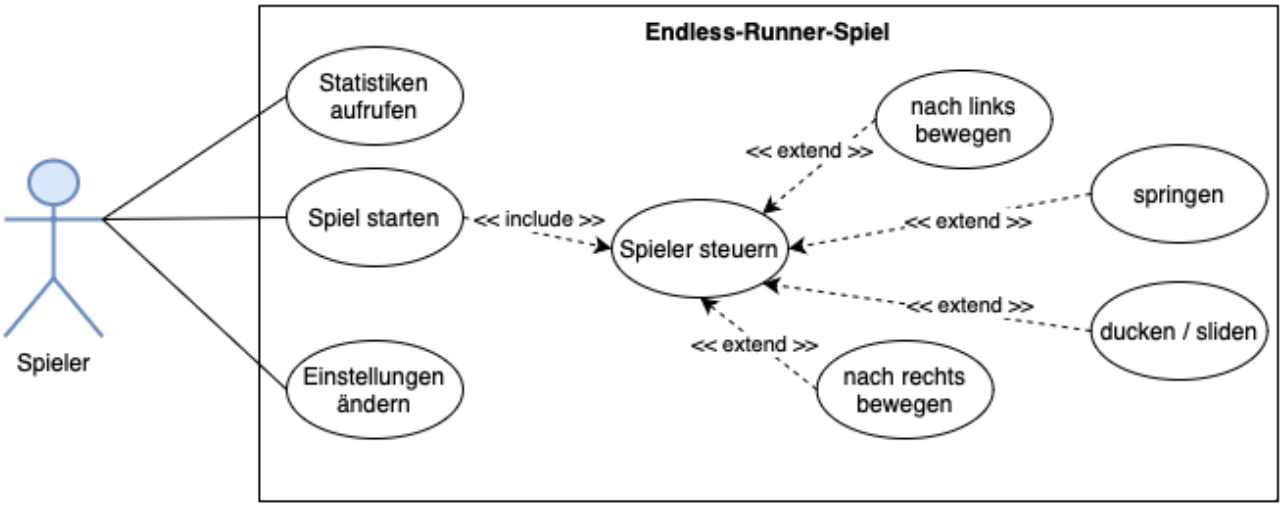

Abbildung 1: UML-Anwendungsfalldiagramm

In diesem UML-Anwendungsfalldiagramm repräsentiert das Endless-Runner-Spiel die klaren Grenzen des Systems. Der einzige Akteur ist der Spieler bzw. Nutzer, der durch die Interaktion mit dem System verschiedene Anwendungsfälle auslöst. Diese Anwendungsfälle umfassen "Statistiken aufrufen", "Einstellungen ändern" und "Spiel starten".

Der Anwendungsfall "Spiel starten" nimmt eine zentrale Rolle ein, da er den Anwendungsfall "Spieler steuern" einschließt. Dieser wiederum koordiniert sämtliche Bewegungen des Spielcharakters, wie springen, ducken oder in eine Richtung laufen, und stellt daher die Schlüsselkomponente während der Spielausführung dar. Zusammengefasst ermöglicht dieses Anwendungsfalldiagramm eine prägnante Darstellung der Systemgrenzen. Es verdeutlicht auf einen Blick, wie der Spieler mit der Endless-Runner-Anwendung interagieren kann, welche Optionen im Menü für den Nutzer verfügbar sind und welche Aktionen im Kontext eines laufenden Spiels ausgeführt werden können.

3 Funktionale Anforderungen

Zur präzisen Beschreibung der funktionalen Anforderungen werden sogenannte User Stories verwendet, welche im Zuge des Projektmanagements häufig für diesen Zweck eingesetzt werden. User Stories konzentrieren sich auf spezifische Benutzerinteraktionen und Bedürfnisse, wodurch die Entwicklung unmittelbar auf die Anforderungen des Endnutzers ausgerichtet wird. Definierte Akzeptanzkriterien für jede User Story dienen als klare Maßstäbe zur Beurteilung der erfolgreichen Umsetzung, indem sie messbare Ergebnisse und klare Erwartungen festlegen. Die Einführung von Story Points ermöglicht zudem eine relative Schätzung des Aufwands jeder einzelnen User Story. Dieses Schätzverfahren unterstützt dabei, den Entwicklungsaufwand abzuschätzen und Prioritäten festzulegen. Die nachfolgenden Abbildungen repräsentieren die wichtigsten User Stories dieses Projekts in Form von Karten. Die Story Points reichen hierbei von eins bis fünf und spiegeln eine Bandbreite von geringstem bis höchstem Aufwand wider.

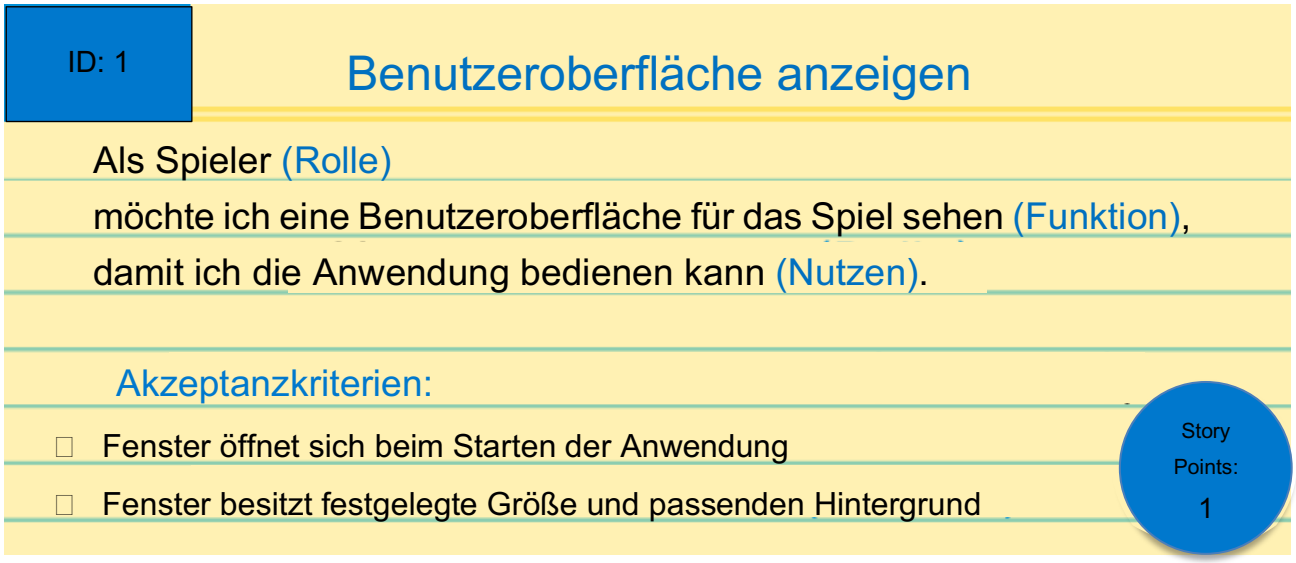

Abbildung 2: User Story - Benutzeroberfläche anzeigen

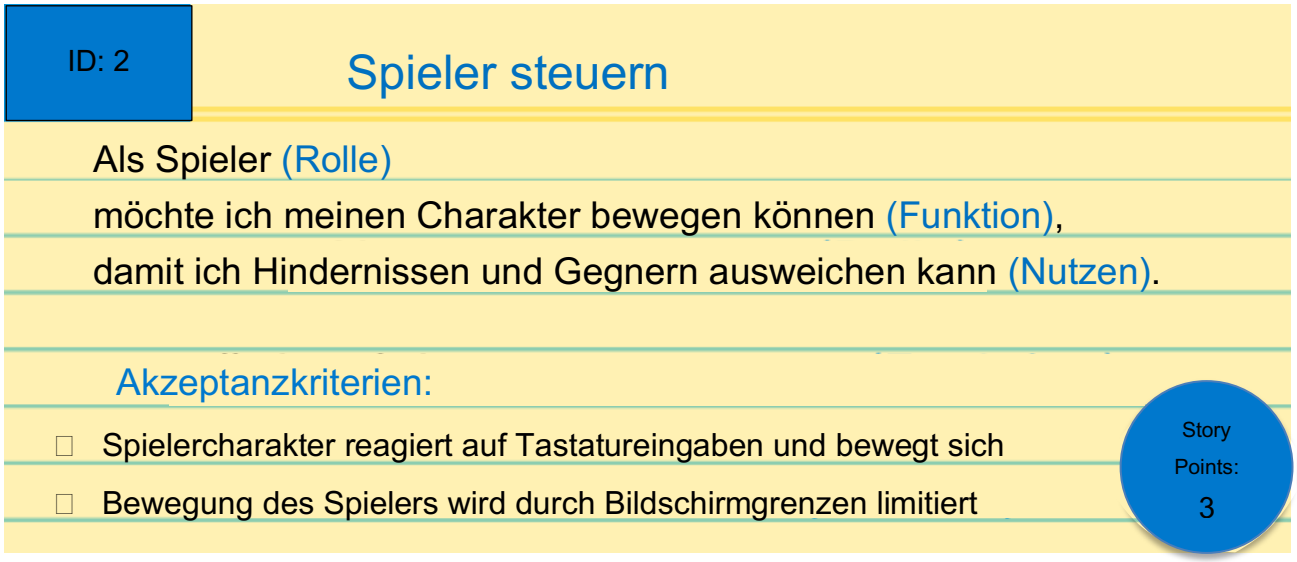

Abbildung 3: User Story - Spieler steuern

ID: 3

Gegner mit verschiedenen Attacken

Als Spieler (Rolle)
möchte ich die Fähigkeiten meiner Gegner kennenlernen (Funktion),
damit ich meine Strategie anpassen kann (Nutzen).

Akzeptanzkriterien:

- Gegner sind sichtbar und visuell einfach zu unterscheiden
- Gegner besitzen verschiedene Fähigkeiten (Springen, Schießen)

Story Points: 4

Abbildung 4: User Story - Gegner mit verschiedenen Attacken

ID: 4

Funktionale Spiellogik

Als Spieler (Rolle)
möchte ich das ein Spiel korrekt beendet und neugestartet wird (Funktion),
damit ich mich verbessern kann (Nutzen).

Akzeptanzkriterien:

- Spiel wird bei Kollision mit Gegner oder Hindernis beendet
- Neustart startet Spiel / Versuch neu

Story Points: 5

Abbildung 5: User Story - Funktionale Spiellogik

ID: 5

Fortschritt anzeigen

Als Spieler (Rolle)
möchte ich einen Meterzähler sehen (Funktion),
damit ich meinen Fortschritt verfolgen kann (Nutzen).

Akzeptanzkriterien:

- Meterzähler erhöht sich basierend auf zurückgelegter Strecke
- Korrekte Zurücksetzung bei Spielneustart

Story Points: 1

Abbildung 6: User Story - Fortschritt anzeigen

ID: 6

Hauptmenü anzeigen

Als Spieler (Rolle)
möchte ich ein Hauptmenü sehen (Funktion),
damit ich das Spiel starten oder Optionen auswählen kann (Nutzen).

Akzeptanzkriterien:

- Hauptmenü wird bei Anwendungsstart angezeigt
- Spieler kann spiel starten oder Einstellungen anpassen

Story Points: 2

Abbildung 7: User Story - Hauptmenü anzeigen

ID: 7

Power-Ups einsammeln

Als Spieler (Rolle)
möchte ich Power-Ups einsammeln können (Funktion),
damit die Verbesserung des Highscores leichter fällt (Nutzen).

Akzeptanzkriterien:

- Power-Ups erscheinen zufällig auf dem Bildschirm
- Einsammeln eines Power-Ups schaltet jeweilige Funktion frei

Story Points: 3

Abbildung 8: User Story - Power-Ups einsammeln

ID: 8

Belohnungen für Leistungen

Als Spieler (Rolle)
möchte ich Belohnungen für meine Leistungen erhalten (Funktion),
damit ich Anreize für kontinuierliches Spielen habe (Nutzen).

Akzeptanzkriterien:

- Spieler erhält Punkte oder Münzen für erreichte Distanz
- Spieler kann Münzen für Upgrades oder temporäre Power-Ups nutzen

Story Points: 2

Abbildung 9: User Story - Belohnungen für Leistungen

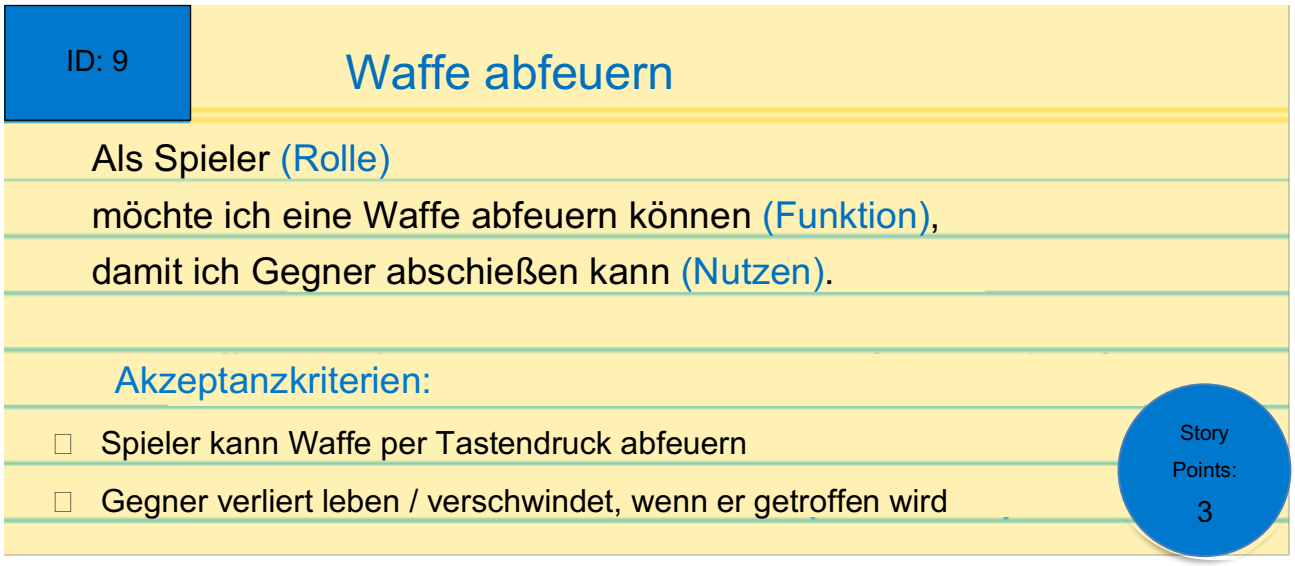

Abbildung 10: User Story - Waffe abfeuern

4 Nicht-funktionale Anforderungen

Während funktionale Anforderungen direkt mit bestimmten Features oder Funktionen in Verbindung stehen, beziehen sich nicht-funktionale Anforderungen auf andere Aspekte eines Systems wie qualitative Eigenschaften, Randbedingungen oder Leistungsmerkmale. Hier werden also keine Handlungen oder Ergebnisse definiert, sondern stattdessen wird beschrieben, wie das System bestimmte Qualitäten erfüllen soll. Nachfolgend sind die wichtigsten Punkte aufgelistet, die das Endless-Runner-Spiel, welches in diesem Software Engineering Projekt entwickelt wird, erfüllen soll.

- Leistung
    - Das Spiel sollte flüssig und reaktionsschnell laufen, ohne Verzögerungen oder Ruckeln, um ein optimales Spielerlebnis zu gewährleisten.
- Benutzeroberfläche
    - Die Benutzeroberfläche sollte benutzerfreundlich und intuitiv gestaltet sein, um eine einfache Interaktion und Navigation für Spieler zu ermöglichen.
- Zuverlässigkeit
    - Das Spiel sollte stabil und zuverlässig sein, ohne häufige Abstürze, Fehler oder Bugs, um eine positive Spielerfahrung sicherzustellen.
- Barrierefreiheit
    - Das Spiel sollte barrierefrei und für Spieler mit unterschiedlichen Fähigkeiten oder Erfahrungen geeignet sein.
- Plattformkompatibilität
    - Die Anwendung sollte sowohl unter Windows 10 als auch unter Windows 11 vollumfänglich und ohne Probleme funktionieren.
- Wartbarkeit
    - Der Quellcode des Spiels sollte gut dokumentiert und strukturiert sein, um zukünftige Wartungen und Aktualisierungen bzw. Anpassungen zu erleichtern.
- Ladezeiten
    - Das Spiel sollte schnelle Ladezeiten aufweisen, um die Wartezeit für die Spieler zu minimieren und ein reibungsloses Spielerlebnis zu realisieren.

5 Glossar

Begriff	Beschreibung / Erklärung
Akzeptanzkriterien	Klar definierte Kriterien, die erfüllt sein müssen, damit eine User Story als abgeschlossen bzw. erfüllt gilt.
Cyberpunk	Ein Genre in der Science-Fiction, das oft eine dystopische, technologisch fortschrittliche Welt mit starken visuellen Elementen betont.
Endless-Runner	Ein Spielgenre, bei dem der Spieler eine Figur steuert, die unendlich lange durch eine sich ständig verändernde Umgebung läuft.
Gantt-Diagramm	Eine visuelle Darstellung eines Projektzeitplans, die die zeitliche Abfolge von Aufgaben und Meilensteinen zeigt.
Highscore	Die höchste erreichte Punktzahl eines Spielers.
Kollisionserkennung	Die Fähigkeit des Spiels, Kollisionen zwischen verschiedenen Objekten im Spiel zu erkennen, wie z.B. der Spielerfigur und Hindernissen.
Power-Ups	Objekte oder Elemente im Spiel, die dem Spieler temporäre Vorteile oder Fähigkeiten verleihen.
Story Points	Ein Maß für den Aufwand, der benötigt wird, um eine bestimmte User Story abzuschließen.
User Story	Eine kurze, benutzerzentrierte Beschreibung einer Funktion oder eines Features aus der Sicht des Benutzers.

Projektdokumentation – Projekt Software Engineering

Erstellt von: Jan Wunderlich
Matrikelnummer: 92205002
Studiengang: Master of Science Informatik
Tutor: Prof. Dr. Markus Kleffmann
Datum: 23.11.2023

Inhaltsverzeichnis

1 Vorgehensmodell

Im Rahmen der Entwicklungsphase für das vorliegende Projekt wird die agile Methode Scrum eingesetzt. Scrum ist ein bewährtes und flexibles Vorgehensmodell, das sich besonders gut für Softwareprojekte eignet, bei denen sich Anforderungen ändern können. Die Wahl von Scrum als Vorgehensmodell für die Erstellung des Endless-Runner-Spiels erfolgt aufgrund spezifischer Überlegungen, die die besonderen Anforderungen dieses Vorhabens berücksichtigen.

Scrum zeichnet sich durch eine hohe Flexibilität aus, die besonders der Natur von Endless-Runner-Games entgegenkommt. Diese Spiele unterliegen häufig sich ändernden Anforderungen, sei es zur Verbesserung des Gameplays, zur Integration neuer Features oder zur Anpassung aufgrund von Testergebnissen. Scrum ermöglicht dementsprechend eine agile Reaktion auf diese Veränderungen, indem sie in den Product-Backlog integriert und in den kurzen Sprints priorisiert werden.

Die iterative und inkrementelle Struktur von Scrum stellt sicher, dass die Implementierung des Endless-Runner-Spiels in handhabbare Sprints unterteilt werden kann. Dies ermöglicht nicht nur eine kontinuierliche Entwicklung, sondern auch die schnelle Sichtbarkeit von Fortschritten. In kurzen Entwicklungszyklen können so regelmäßig spielbare Teile des Spiels erstellt, getestet und verbessert werden. Die Sprintdauer für dieses Projekt wird auf 5 Tage festgelegt, um eine ausgewogene Balance zwischen Effizienz, Flexibilität und regelmäßiger Leistungsbewertung sicherzustellen. Ein weiterer Vorteil von Scrum liegt im klaren Fokus auf die Benutzererfahrung und die kontinuierliche Anpassung des Produkts. Diese Aspekte sind besonders relevant für die Entwicklung eines Endless-Runner-Games, bei dem die Spielerfahrung im Mittelpunkt steht. Die Möglichkeit, Gameplay, Grafiken und Benutzeroberfläche iterativ zu verbessern, sorgt dafür, dass das Spiel am Ende den Erwartungen der Spieler entspricht.

Obwohl Scrum ursprünglich für die Teamarbeit konzipiert wurde, bietet es auch für Einzelentwickler klare Vorteile. Eine transparente Kommunikation sowie regelmäßige Selbstreflexion ermöglichen es, den eigenen Fortschritt zu bewerten, Schwierigkeiten zu identifizieren und gezielte Anpassungen vorzunehmen. Die Verwendung eines Kanban-Boards und des Product-Backlogs innerhalb des Scrum-Frameworks unterstützt zudem die effiziente Verwaltung von Aufgaben und Prioritäten. Das Kanban-Board verbessert die Übersichtlichkeit und zeigt den Status aller Aufgaben des aktuellen Sprints an. Das Product-Backlog dient als zentrale Sammelstelle für Ideen und Aufgaben, die in den kommenden Sprints organisiert werden.

Insgesamt schafft Scrum eine ausgewogene und flexible Umgebung, maßgeschneidert für die Herausforderungen der Einzelentwicklung eines Endless-Runner-Spiels. Es ermöglicht nicht nur eine effiziente und iterative Entwicklung, sondern auch die stetige Anpassung des Spiels, um eine optimale Spielerfahrung zu gewährleisten. Die kurzen, regelmäßigen Iterationen fördern dabei eine dynamische Anpassung an sich verändernde Anforderungen und eine kontinuierliche Verbesserung der Spielerzufriedenheit im Verlauf des Entwicklungsprozesses.

2 Technologien und Tools

Um eine effiziente und erfolgreiche Umsetzung des Projekts zu gewährleisten, werden in der Entwicklungsphase verschiedene Technologien, Tools und Frameworks eingesetzt. Als Programmiersprache wird Python gewählt, da persönliche Erfahrungen mit dieser Sprache vorliegen. Python bietet eine klare und leserliche Syntax, was die Entwicklung und Wartung erleichtert. Zudem verfügt Python über umfangreiche Bibliotheken, die speziell für die Spieleprogrammierung genutzt werden können. Pygame (siehe https://www.pygame.org/wiki/about) wird als Bibliothek für die Spieleentwicklung integriert, da es sich als äußerst geeignet für die Umsetzung eines Endless-Runner-Games erwiesen hat. Pygame bietet Funktionen und Werkzeuge, die besonders auf die Anforderungen von 2D-Spielen zugeschnitten sind. Die Bibliothek vereinfacht die Handhabung von Grafiken, Animationen, Kollisionserkennung und andere relevante Aspekte der Spieleentwicklung. Die einfache Integration von Bildern, Sounds und weiteren Ressourcen macht Pygame zu einer optimalen Wahl für die Umsetzung eines Endless-Runner-Spiels.

Für die IDE bzw. Entwicklungsumgebung wird PyCharm verwendet. Diese Wahl beruht auf persönlichen Erfahrungen und Vorlieben im Umgang mit PyCharm. PyCharm bietet verschiedenen Funktionen wie Code-Intelligenz, Debugging-Tools oder eine Codeformatierung unter Beachtung des integrierten Standards PEP-8 für Python, die bei der Entwicklung und Fehlersuche helfen. Als zentrales Tool für die Umsetzung des Scrum-Vorgehensmodells wird JIRA eingesetzt. JIRA vereint verschiedene Funktionen, die für die agile Softwareentwicklung entscheidend sind. Es ermöglicht die Verwaltung des Product-Backlogs, die Planung von Sprints und die Organisation bzw. Nachverfolgung von Aufgaben durch ein Kanban-Board. Durch die Integration von JIRA wird eine transparente und strukturierte Umsetzung des Scrum-Frameworks gewährleistet, was die effektive Planung und Durchführung des Entwicklungsprozesses ermöglicht. Für das vorliegende Projekt respektive das zu implementierende Endless-Runner-Game wurde ein neues eigenständiges Projekt in JIRA angelegt, welches unter folgendem Link zu erreichen ist: https://pse-endlessrunnergame.atlassian.net/jira/software/projects/PSE/boards/1/backlog.

Als Versionierungstool wird GitHub bzw. Git eingesetzt. GitHub ist eine Plattform, die auf Git basiert und eine zentrale sowie kollaborative Umgebung für die Codeverwaltung bietet. Die Integration von Git ermöglicht eine effiziente Verwaltung von Code, das Nachvollziehen von Änderungen sowie das Wechseln von verschiedenen Entwicklungsständen, sollte ein entsprechender Bedarf bestehen. Für das Endless-Runner-Game wurde bereits ein neues GitHub Repository erstellt, das unter https://github.com/jan-wun/PSE_Endless-Runner-Game betrachtet werden kann.

Die bewusste Auswahl von Technologien und Tools, darunter Python als Programmiersprache, Pygame für die Spieleentwicklung, PyCharm als IDE, JIRA für das Scrum-Vorgehensmodell und GitHub als Codeversionierungstool, schafft eine harmonische Tool-Landschaft. Diese Tools unterstützen nicht nur die spezifischen Anforderungen des Projekts, sondern gewährleisten auch einen reibungslosen Entwicklungsprozess durch effektive Codeverwaltung und agile Planung.

3 Klassendiagramm

Zur Visualisierung der Struktur des geplanten Endless-Runner-Spiels wird ein UML-Klassendiagramm verwendet. Dieses Diagramm bietet einen klaren und verständlichen Überblick über die Klassen, ihre Attribute, Methoden und die Beziehungen zwischen ihnen. Durch die Nutzung eines UML-Klassendiagramms kann folglich ein schneller Einblick in die Architektur des Softwareprojekts gewonnen werden. Aus Übersichtlichkeitsgründen ist das Diagramm für das vorliegende Software Engineering Projekt im Querformat auf der nächsten Seite dargestellt.

Die Struktur des Endless-Runner-Spiels wurde im UML-Klassendiagramm sorgfältig entworfen, um die verschiedenen Komponenten und ihre Interaktionen übersichtlich darzustellen. Die Hauptklassen sind *Game*, *Player*, *Enemy*, *PowerUp*, *Obstacle* und *Weapon*. Das *Game*-Objekt koordiniert alle Abläufe im Spiel, während *Player*, *Enemy*, *PowerUp* und *Obstacle* spezifische Spielentitäten repräsentieren. Die Klasse *Weapon* modelliert die verschiedenen Waffen im Spiel.

Zusätzlich dazu zeigt das Diagramm, dass verschiedene Manager-Klasse wie *AudioManager*, *Menu* und *ScoreManager* existieren, die für Audioeffekte, Menüs und die Punkteverwaltung bzw. den Highscore verantwortlich sind. Die *Entity*-Klasse dient als gemeinsame Abstraktion für alle spielbezogenen Objekte oder Entitäten. Die Verwendung von Vererbungen und Assoziationen wird genutzt, um die Beziehungen zwischen den Klassen zu verdeutlichen. Zum Beispiel erbt *Player* von *Entity*, was die gemeinsamen Eigenschaften und Methoden definiert.

Darüber hinaus beinhaltet das Diagramm die Enums *GameState* und *PlayerState*, die den aktuellen Zustand des Spiels bzw. des Spielers repräsentieren. Diese Zustände werden innerhalb des *Game*- und *Player*-Objekts verwaltet und beeinflussen das Spielgeschehen entsprechend. Diese klaren Hierarchien und Verbindungen ermöglichen eine bessere Strukturierung und Wartbarkeit des Codes während der Entwicklung.

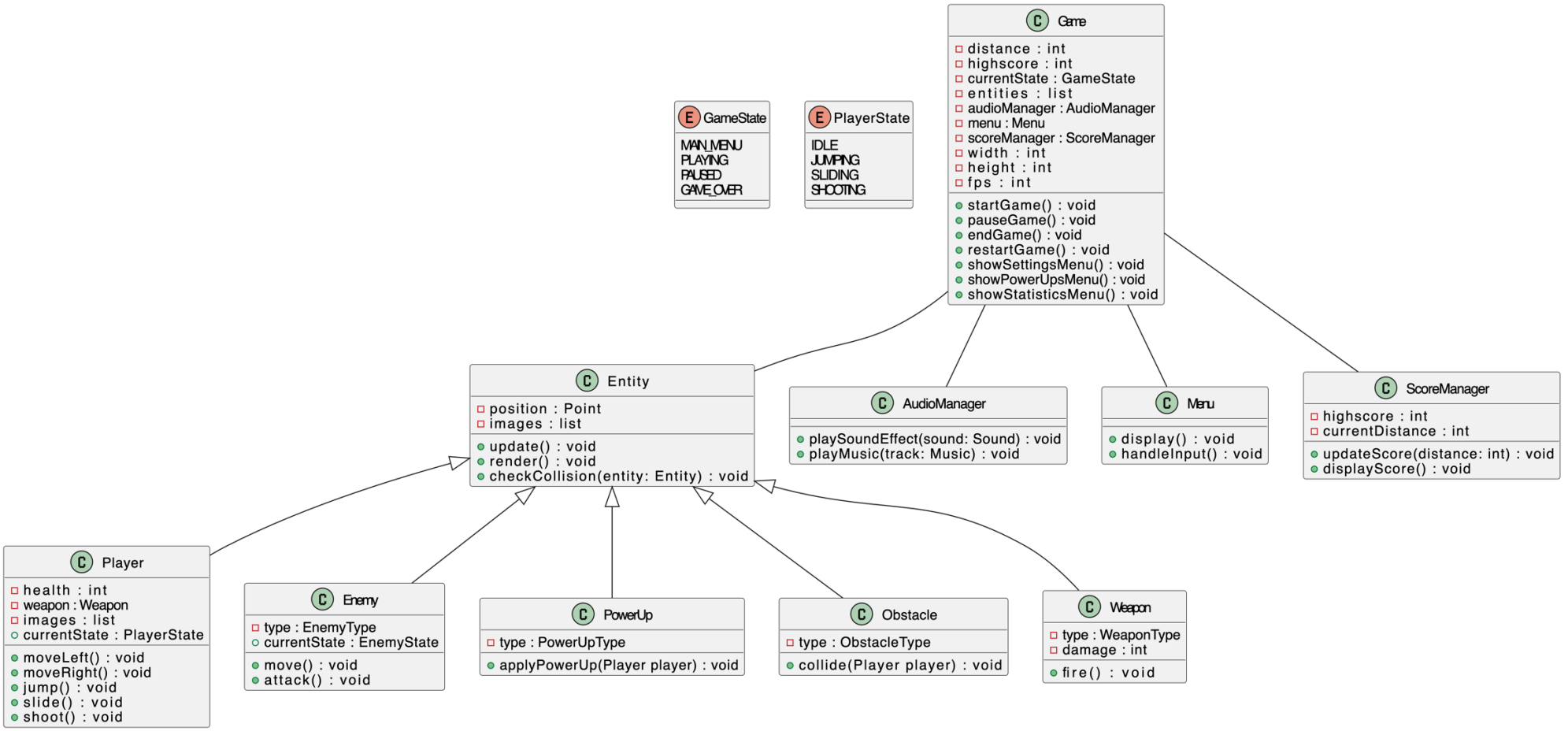

Abbildung 1: UML-Klassendiagramm