THM | CAMPUS GIESSEN | MNI
Mathematik, Naturwissenschaften und Informatik
TECHNISCHE HOCHSCHULE MITTELHESSEN

Bachelorthesis

Analyse von Kriterien zum Vergleich von Game Engines

zur Erlangung des akademischen Grades

Bachelor of Science

eingereicht im Fachbereich Mathematik, Naturwissenschaften und Informatik an der Technischen Hochschule Mittelhessen

von

Anastasia Kunst

17. Oktober 2023

Referent: Dr. Dennis Priefer

Korreferent: Prof. Dr. Peter Kneisel

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Gießen, den 17. Oktober 2023 Anastasia Kunst

Die vorliegende Arbeit untersucht die Evaluation und den Vergleich von Game-Engines anhand zentraler Kriterien im Kontext der Spieleentwicklung. Angesichts der Vielzahl von verfügbaren Game-Engines auf dem Markt und der unterschiedlichen Funktionalitäten stehen Entwickler vor der Herausforderung, die am besten geeignete Engine für ihr Projekt auszuwählen. Ziel der Arbeit ist es, relevante Kriterien zum Vergleich von Game-Engines zu identifizieren und eine Strategie zu entwickeln, um die am besten geeignete Engine für ein spezifisches Projekt auszuwählen. Die Analyse stützt sich auf die Betrachtung von Game-Engine-Komponenten, Softwarequalitätskriterien und existierenden Vergleichsarbeiten. Die Anwendung eines Nutzwertanalyse-Konzepts in Verbindung mit den identifizierten Kriterien ermöglicht die Entwicklung einer Strategie zum Vergleich von Game-Engines. Die Auswahl einer Game-Engine erfordert die Berücksichtigung verschiedener Funktionen und Eigenschaften. Je besser diese zu den spezifischen Projektanforderungen passen, desto positiver beeinflussen sie den Entwicklungsprozess und das Endergebnis.

Inhaltsverzeichnis

1 Einführung

In den vergangenen Jahren hat sich die Videospielindustrie zu einer der größten und am schnellsten wachsenden Branchen im Bereich der Unterhaltung entwickelt [Sta23]. Der Umsatz von Videospielen überstieg 2021 den Umsatz, der durch Filme und Musik gemeinsam generiert wurde. [Zan22].

Die Produktion erfolgreicher Spiele erfolgt oft durch große, etablierte Entwicklungsstudios, die als AAA (Triple-A) Videospiele bezeichnet werden. Diese Spiele haben ein hohes Produktionsbudget und tendieren daher dazu, eine höhere Qualität als andere Videospiele aufzuweisen. Beispiele für AAA-Spiele sind ”The Witcher 3: Wild Hunt”, ”Assassin’s Creed Valhalla” und ”Red Dead Redemption 2”.
Allerdings gewinnen auch Indie-Spiele (independent video games) immer mehr an Popularität [SN21]. Diese werden von einzelnen Personen oder kleinen Teams entwickelt und sind unabhängig von finanzieller und technischer Unterstützung großer Publisher, weshalb sie mit einem kleineren Budget auskommen müssen. Indie-Spiele zeichnen sich oft durch ihre kreativen und originellen Konzepte aus, die sich von den AAA-Titeln unterscheiden und deshalb immer beliebter werden [Fle22]. Beispiele für Indie-Spiele sind ”Undertale”, ”Stardew Valley” und ”Stray”.

In dem Zusammenhang spielen Game-Engines eine bedeutende Rolle. Eine Game-Engine ist eine Software, die Entwickler bei der Entwicklung von Videospielen unterstützt, indem sie grundlegende Funktionalitäten für die Entwicklung von Videospielen bereitstellt.
Studios, darunter AAA-Entwicklungsstudios, entwickeln und verwenden ihre eigenen Game-Engine, um die Kontrolle über die technische Umsetzung ihrer Spiele zu behalten, wie zum Beispiel: ”The Witcher 3: Wild Hunt” von CD Projekt Red verwendet die ”REDengine” und ”Red Dead Redemption 2” von Rockstar Games verwendet die ”Rockstar Advanced Game-Engine (RAGE)”.

Durch die Entwicklung von Open-Source-Software und die Verfügbarkeit von kostengünstigen Game-Engines haben Indie-Entwickler heute leichteren Zugang zu besseren Werkzeugen als früher [Uni22]. Dadurch wird nicht nur das Entwickeln von Indie-Games möglich, sondern auch der generelle Einstieg in die Videospielentwicklung [Fle22]. Es gibt eine wachsende Anzahl an frei verfügbaren/lizenzierbaren Game-Engines wie die Unreal Engine, Unity Engine, Godot Engine und die CryEngine.

1.1 Problembeschreibung und Motivation

Es gibt eine Vielzahl von verfügbaren Game-Engines auf dem Markt, die von Entwicklern genutzt werden können, um ihre Videospiele zu entwickeln. Jede dieser Engines hat ihre eigenen spezifischen Vor- und Nachteile, die einen Einfluss auf das Ergebnis und den Entwicklungsprozess des Spiels haben. Game-Engine werden für bestimmte Genres und Plattformen optimiert. Dies bedeutet, dass Entwickler die spezifischen Funktionen und Eigenschaften der Engines berücksichtigen müssen, wenn sie eine Engine für ihr Projekt wählen, um so gut wie möglich bei der Entwicklung unterstützt zu werden.

Zwei der am häufigsten verwendeten Game-Engines sind Unreal Engine und Unity. Diese bieten umfangreiche Funktionen für die Entwicklung von Videospielen [TIG19]. Unreal Engine ist eine leistungsstarke Engine, die von Epic Games entwickelt wurde und auch für erfolgreiche AAA-Titel wie Fortnite und Hogwarts Legacy eingesetzt wird [EpioD]. Unity hingegen wird von vielen Indie-Entwicklern eingesetzt und ist weiter verbreitet. Im Steam-Shop werden fast viermal so viele Spiele angeboten, die auf Unity basieren, als Spiele, die mit der Unreal Engine entwickelt wurden [Ste23c].

Am 12. September 2023 kündigte Unity Technologies eine Änderung in ihrem Preismodell an. Ab dem 1. Januar 2024 plant Unity, unter anderem eine Gebühr für jede Installation eines Unity-Spiels zu erheben [Uni23].
Diese Ankündigung hat erhebliche Auswirkungen auf die Entwickler-Community, insbesondere auf Indie-Entwickler und kleine Studios, und hat Diskussionen in der Community ausgelöst. Studios und Entwickler äußerten ihre Sorgen über die Auswirkungen dieser Preisänderung auf ihre Projekte und Geschäftsmodelle. Einige Entwickler haben begonnen, Alternativen wie die Unreal Engine oder die Godot Engine in Betracht zu ziehen und erwägen die Möglichkeit, ihre laufenden Projekte von Unity auf andere Engines zu migrieren oder diskutierten die Möglichkeit, ihre Spiele vom Markt zu nehmen [Usl23].

Diese jüngsten Entwicklungen betonen die Dringlichkeit der in dieser Arbeit behandelten Thematik und unterstreichen die Notwendigkeit einer sorgfältigen Analyse der Auswahlkriterien für Game-Engines. Angesichts der Vielzahl von verfügbaren Game-Engines auf dem Markt und der unterschiedlichen Funktionalitäten stehen Entwickler vor der Herausforderung, die am besten geeignete Engine für ihre Projekte auszuwählen.

1.2 Ziele dieser Arbeit

Ziel dieser Arbeit ist es, Spieleentwicklern Einblicke in die Funktionalitäten von Game-Engines zu vermitteln und herauszufinden, welche Kriterien bei der Bewertung und dem Vergleich von Game-Engines von besonderer Relevanz sind. Des Weiteren wird untersucht, wie anhand dieser Kriterien eine geeignete Game-Engine für ein spezifisches Projekt ausgewählt werden kann.

Um diese Ziele zu erreichen, wurden folgende Forschungsfragen definiert, die im Verlauf der Arbeit beantwortet werden:

[F1] Welche Kriterien sind bei der Gegenüberstellung von Game-Engines von besonderer Bedeutung?

[F2] Wie kann systematisch ermittelt werden, welche Game-Engine für ein spezifisches Projekt besser geeignet ist?

[F3] In welchen Aspekten unterscheiden sich Game-Engines hinsichtlich ihrer Funktionen, und welche Gemeinsamkeiten lassen sich identifizieren?

Durch die Ausarbeitung dieser Fragen soll Spieleentwicklern dabei geholfen werden, eine geeignete Game-Engine für ihr nächstes Projekt auszuwählen.

1.3 Vorgehensweise

Um die definierten Ziele zu erreichen, gliedert sich diese Arbeit in einen theoretischen und einen praktischen Teil.

Der erste Schritt besteht darin, eine Literaturrecherche durchzuführen, mit dem Ziel, die Anforderungen an Game-Engines sowie deren Funktionen im Kontext der Spieleentwicklung zu identifizieren, einschließlich der Identifikation bestehender Kriterien, für den Vergleich von Game-Engines. Die Recherche bezieht sich besonders auf wissenschaftliche Artikel, Fachbücher und relevante Online-Ressourcen zu den Themen Game-Design Elemente, Game-Engine Architektur und Game-Engine Vergleiche. Darauf aufbauend erfolgt, unter der Berücksichtigung von Softwarequalitätskriterien, die Ableitung von Vergleichskriterien für den Vergleich von Game-Engines, wodurch die erste Forschungsfrage F1 beantwortet wird.

Im nächsten Schritt erfolgt die Ausarbeitung einer Strategie zur Auswertung der Vergleichskriterien, um Game-Engines anhand der festgelegten Kriterien systematisch vergleichen und bewerten zu können, mit dem Schwerpunkt eine Game-Engine für ein

spezifisches Projekt auszuwählen, wodurch die zweite Forschungsfrage F2 beantwortet wird.

Die Beantwortung der dritten Forschungsfrage F3 erfolgt im praktischen Teil der Arbeit. Hier wird ein Vergleich zwischen Unreal Engine 5 und Unity 2022 anhand der entwickelten Strategie durchgeführt. Dafür werden spezifische Anforderungen an ein Spiel definiert, um die Eignung beider Game-Engines für einen konkreten Anwendungsfall zu evaluieren. Dafür werden die Dokumentationen der Engines herangezogen und prototypische Tests durchgeführt, um den Auswahlprozess einer Engine realitätsnah zu gestalten und um Gemeinsamkeiten und Unterschiede im Kontext des Anwendungsbeispiels zu identifizieren. Die Beispielanwendung wird dabei ein Open-World-Single-Player-3D-Third-Person Action-Adventure als Desktop-Anwendung sein.

1.4 Abgrenzung

In vorangegangenen Untersuchungen lag der Fokus oft darauf, einzelne Funktionen zu bewerten und spezifische Anwendungsfälle zu untersuchen, sowie allgemeine Gegenüberstellungen von Game-Engines durchzuführen. Diese Arbeit integriert die bereits erarbeiteten Vergleichskriterien dieser Arbeiten, erweitert sie um fehlende Aspekte und präsentiert eine Methode, wie diese Kriterien systematisch bewertet werden können, um eine fundierte Wahl einer Engine zu ermöglichen.
Diese Arbeit konzentriert sich auf die Ausarbeitung von Kriterien zum Vergleich von Game-Engines, ohne sich in die technischen Details und Architekturen einzelner Engines und Funktionen zu vertiefen.
Die Vergleichsanalyse dieser Arbeit dient als exemplarisches Beispiel und beschränkt sich auf einen Vergleich zwischen Unreal Engine 5 und Unity 2022 für einen spezifischen Anwendungsfall. Andere Game-Engines werden in dieser Untersuchung nicht berücksichtigt.
Der Anwendungsfall bezieht sich auf ein Open-World-Single-Player-3D-Third-Person Action-Adventure als Desktop-Anwendung. Andere Spielgenres, Plattformen sowie Multiplayer-Aspekte und Künstliche Intelligenz (KI) im Kontext von Deep Learning werden nicht behandelt.
Es wird darauf hingewiesen, dass die entwickelten Spiele lediglich Prototypen sind. Der Fokus liegt darauf, die erarbeiteten Kriterien vergleichen zu können und nicht in der Entwicklung vollständiger Spiele, um den Auswahlprozess einer Engine realitätsnah zu gestalten.
Diese Arbeit gibt keine konkreten Empfehlungen für die Entwicklung von Videospielen in bestimmten Genres. Vielmehr sollen die erarbeiteten Vergleichskriterien sowie die präsentierte Auswahlmethode den Entwicklern dabei helfen, selbstständig zu entscheiden, welche Engine für ihr Projekt am besten geeignet ist. Die Arbeit wird jedoch bewerten, welche Engine für das Anwendungsbeispiel besser geeignet ist.

1.5 Struktur der Arbeit

Die Bachelorarbeit unterteilt sich in fünf Kapitel.
Das zweite Kapitel, "Grundlagen", vermittelt Hintergrundwissen zum Thema Videospiel und den verschiedenen Aufgabenbereichen in einem Entwicklungsstudio. Hierbei wird ein Überblick über die Elemente eines Videospiels und die Aufgaben von Videospielentwicklern gegeben, aus denen sich die Anforderungen an eine Game-Engine ableiten lassen. Zusätzlich dazu werden der Aufbau und die Funktionen einer Game-Engine thematisiert sowie der Zweck, den sie in der Spieleentwicklung erfüllt.
Das dritte Kapitel präsentiert das Konzept für den praktischen Teil der Arbeit. Es werden die Vergleichskriterien zum Vergleichen von Game-Engines identifiziert und erläutert. Anschließend wird eine Methode beschrieben, um diese auszuwerten, mit dem Ziel, eine Game-Engine für ein spezifisches Projekt auswählen zu können.
Im vierten Kapitel wird das Konzept angewendet, um eine geeignete Game-Engine für ein spezifisches Anwendungsbeispiel zu bestimmen. Zudem werden die Gemeinsamkeiten und Unterschiede der Engines im Kontext des Beispiels identifiziert. Vorher erfolgt eine kurze Vorstellung von Unreal Engine und Unity, gefolgt von der Festlegung der genauen Anforderungen an das Anwendungsbeispiel.
Im letzten Kapitel, "Zusammenfassung und Fazit", werden die Ergebnisse zusammengefasst und ein Fazit gezogen. Die wichtigsten Erkenntnisse der Arbeit werden kurz aufgeführt, und ein Ausblick auf mögliche weitere Forschungen wird gegeben.

2 Grundlagen

In diesem Kapitel erfolgt zunächst die Definition von Videospielen, gefolgt von einer Erläuterung der unterschiedlichen Aufgabenbereiche eines Entwicklungsstudios. Das Kapitel bietet einen Überblick über die Anforderungen und Aufgaben von Videospielentwicklern. Es dient dazu, die Zusammenhänge zwischen den Aufgabenbereichen eines Entwicklungsstudios, den Anforderungen an ein Videospiel und den Funktionen einer Game-Engine im weiteren Verlauf besser zu verstehen. Im letzten Abschnitt dieses Kapitels erfolgt die Definition dessen, was eine Game-Engine ist, sowie eine Betrachtung des Aufbaus und der Funktionen. Dieses Kapitel bildet die Grundlage für die Ausarbeitung der Vergleichskriterien im nächsten Kapitel.

2.1 Elemente und Merkmale von Videospielen

Dieser Abschnitt befasst sich mit den Elementen und Merkmalen von Videospielen. Es erfolgt eine Ausarbeitung der Merkmale eines Spiels sowie eine Untersuchung dessen, was ein Videospiel ausmacht. Zudem wird die Rolle des Gamedesigns beleuchtet und erläutert, wie eine Player Experience entsteht.

2.1.1 Merkmale von Spielen

Spiele sind interaktive Aktivitäten, die durch festgelegte Regeln und Ziele gekennzeichnet sind. Die Spielenden lernen und meistern dabei eine fortlaufend herausfordernde Abfolge von Mustern. Um eine Herausforderung zu bewältigen oder ein bestimmtes Ziel zu erreichen, setzen sie verschiedene Fähigkeiten und Strategien ein. Spiele können sowohl in physischer Form wie Brettspielen, Kartenspielen oder Sportarten stattfinden als auch in digitaler Form wie Videospielen und dienen hauptsächlich der Unterhaltung [JunoD]. Trotz der unterschiedlichen Arten von Aktivitäten haben sie alle Gemeinsamkeiten, die als wichtige Merkmale eines Spiels definiert werden können. Zu diesen Merkmalen gehören:

- **Ziele**: Spiele haben ein oder mehrere Ziele, die erreicht werden müssen. Diese sind messbar und gelten als erreicht oder nicht erreicht [Sal03, K. 7 S. 11].

- **Regeln**: Spiele folgen bestimmten Regeln, die den Ablauf und die Interaktionen innerhalb des Spiels strukturieren. Diese Regeln bestimmen erlaubte Handlungen, den Spielverlauf und mögliche Konsequenzen [Sal03, K. 7 S. 11].

- **Interaktion**: Spiele sind interaktiv, was bedeutet, dass die Spielenden aktiv am Geschehen teilnehmen und auf das Spielgeschehen Einfluss nehmen können. Die Interaktion kann zwischen den Spielenden stattfinden (Mehrspielermodus) oder zwischen einem Spieler und der Spielumgebung ([JunoD]; [Sal03, K. 7 S. 11]).

- **Herausforderung**: Spiele bieten Herausforderungen/Konflikte, die es zu bewältigen gilt. Spieler müssen ihre Fähigkeiten und Strategien einsetzen, um Schwierigkeiten zu überwinden oder Ziele zu erreichen ([JunoD]; [Sal03, K. 7 S. 11]).

- **Vergnügen und Unterhaltung**: Spiele dienen primär der Unterhaltung und sollen den Spielern Freude bereiten. Sie bieten eine Möglichkeit, dem Alltag zu entfliehen, neue Erfahrungen zu machen und eine Form der sinnvollen Unproduktivität zu erleben, indem sie eine künstliche Welt schaffen ([JunoD]; [Sal03, K. 7 S. 11]).

Spiele lassen sich als ein System zusammenfassen, das Regeln, Spieler, künstliche Welten und Strukturen umfasst, um eine organisierte und interaktive Umgebung zu schaffen, in der Spieler aktiv am Spielgeschehen teilnehmen und ein messbares Ergebnis in Form von Sieg oder Niederlage zu erreichen [Sal03, K. 7 S. 14]).

2.1.2 Videospiele als interaktive Echtzeitsimulationen

Ein Videospiel kann allgemein als interaktive digitale Medienform beschrieben werden, bei der ein Spieler mit einer Maschine interagiert, zum Beispiel einem Computer oder einer Spielkonsole. Videospiele erfüllen die Merkmale eines Spiels, wie im Abschnitt 2.1.1 beschrieben. Durch einen fiktionalen Kontext wird dem Spieler eine bedeutungsvolle Umgebung vermittelt, in der er aktiv teilnimmt. Während des Spiels entwickelt der Spieler eine emotionale Bindung zu den Auswirkungen seiner Aktionen innerhalb dieser fiktiven Welt [Ber17].
Im Zusammenhang mit Videospielen wird das Wort ”Spiel” in der Regel mit einer dreidimensionalen virtuellen Welt assoziiert, in der der Spieler die Kontrolle über eine Entität, oft eine Figur, hat [Gre18, S. 9].

Aus technischer Sicht können die meisten zweidimensionalen und dreidimensionalen Videospiele als ”soft real-time interactive agent-based computer simulations” bezeichnet werden [Gre18, S. 9].

In einer Vielzahl von Videospielen wird ein Teil der realen oder einer imaginären Welt mathematisch modelliert, sodass sie von einem Computer manipuliert werden kann. Das Modell ist eine Annäherung und Vereinfachung der Realität (dabei kann es sich auch um eine fiktive Realität handeln). Es ist nicht möglich und nicht nötig, jedes Detail bis zur Ebene der Atome abzubilden. Das mathematische Modell ist dementsprechend eine **Simulation** der realen oder imaginären Spielwelt [Gre18, S. 9].
In einer **agent-based** Simulation interagieren verschiedene Entitäten, die als "Agenten" bezeichnet werden, miteinander[Gre18, S. 9]. Diese Agenten umfassen Non-playable characters (NPCs), die computergesteuerte Charaktere darstellen [Sch23]. Bei diesen Agenten kann es sich um verschiedene Entitäten handeln, wie zum Beispiel: Fahrzeuge, Charaktere oder auch Feuerbälle [Gre18, S. 9].
Interaktive Videospiele sind zeitliche Simulationen. Das bedeutet, dass das virtuelle Spielweltmodell dynamisch ist und der Zustand der Spielwelt sich im Laufe der Zeit verändert. Das Spiel reagiert dabei in Echtzeit auf die Eingaben des menschlichen Spielers, sodass dessen Aktionen unmittelbare Auswirkungen haben. Zum Beispiel kann der Spieler laufen, springen oder etwas einsammeln, indem bestimmte Tasten auf der Tastatur gedrückt werden. All diese Aktionen werden sofort und direkt als Reaktion auf die Eingaben des Spielers umgesetzt, wodurch sich die Spielwelt in Echtzeit anpasst. Dadurch werden interaktive Videospiele zu **interaktiven Echtzeitsimulationen**, da sie eine dynamische virtuelle Welt bieten, in der der Spieler aktiv handeln und Einfluss nehmen kann [Gre18, S. 10].
Echtzeitsysteme können in harte und weiche Echtzeitsysteme eingeteilt werden. Bei Videospielen handelt es sich um ein weiches Echtzeitsystem (**soft real-time** system). Ein weiches Echtzeitsystem ist ein System, in dem verpasste Fristen (ein Zeitlimit, innerhalb dessen ein Prozess abgeschlossen werden muss) keine kritischen Konsequenzen zufolge haben [Zö19, S. 3]. Ein Beispiel dafür ist, dass der Bildschirm mindestens 24 Mal pro Sekunde aktualisiert werden muss, um die Illusion von flüssiger Bewegung aufrechtzuerhalten. Wenn die Bildwiederholrate sinkt, hat das keinen schwerwiegenden Einfluss, sondern lediglich Ruckler in der Darstellung ([Gre18, S. 10]; [Zö19, S. 3]).
Im Gegensatz dazu können harte Echtzeitsysteme (hard real-time system) bei einer Zeitüberschreitung Probleme verursachen. In solchen Systemen sind verpasste Fristen nicht tolerierbar und können zu schwerwiegenden Konsequenzen führen. Ein Beispiel hierfür sind Steuerungssysteme in Flugzeugen. Wenn die Reaktion auf eine Eingabe zu lange dauert, kann Menschenleben gefährdet werden [Zö19, S. 3].

Diese Erkenntnisse verdeutlichen, was ein Videospiel ist, mit Hinblick auf die Verbindung zur Technologie als digitale Medienform. Weitere Merkmale von Spielen wie Ziele, Regeln, Herausforderungen und Unterhaltung sowie die bereits erwähnte emotionale Bindung und immersive Spielerfahrung werden durch das Game Design bestimmt [Ste19].

2.1.3 Game Design und Player Experience

Werke wie Filme und Bücher haben die Fähigkeit, die Betrachter zu fesseln und ihnen eine andere Perspektive auf die Welt zu ermöglichen. Videospiele besitzen diese Fähigkeit ebenfalls, indem sie den Spielenden die Möglichkeit geben, in die Rolle anderer Charaktere einzutauchen und deren Leben, Welt und Geschichte zu erleben. Eine besondere Eigenschaft von Videospielen besteht darin, dass sie den Spielenden ermöglichen, die Welt nicht nur zu beobachten, sondern auch mit ihr zu interagieren. Sie erleben die Konsequenzen ihrer Handlungen und lernen aus ihren Erfahrungen, wie diese Welt funktioniert [Zub20, S. 1].

Game Design ist ein multidisziplinärer Bereich in der Spieleentwicklung und beschäftigt sich mit der Konzeption und Planung von Spielen, einschließlich der Schaffung von Regeln, Mechanismen und Inhalten, die ein interaktives und immersives Spielerlebnis ermöglichen ([Ste19]; [Zub20, S. 1-2]).
Für ein solches Erlebnis sind drei Elemente von besonderer Bedeutung:
Mechaniken, Gameplay und Player Experience.

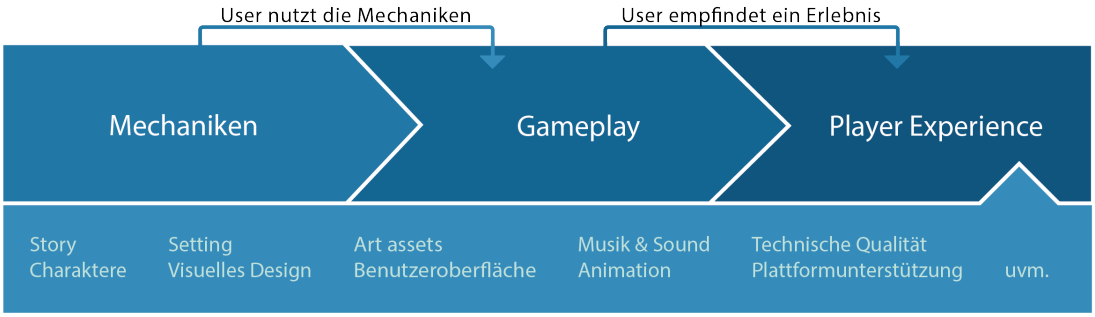

Abbildung 2.1: Game Design Elemente, in Anlehnung an [Zub20, S. 11]

Die **Mechaniken** beziehen sich auf die Spielobjekte und Aktionen, mit denen der Spieler interagiert. Sie bestimmen die Regeln, Bedingungen und Einflussmöglichkeiten des Spielers auf den Spielzustand. Im Beispiel von Poker wären die Spielkarten und die möglichen Züge der Spieler als Mechaniken zu sehen [Zub20, S. 5].
Das **Gameplay** ist der eigentliche Prozess oder die Art und Weise, wie ein Spieler mit den Spielmechaniken interagiert. Es umfasst die aktive Handlung und Entscheidungsfindung des Spielers, während er die Spielmechaniken nutzt. Im Falle von Poker wäre das Gameplay das eigentliche Kartenspiel und wie die Spieler ihre Züge planen und taktische Entscheidungen treffen [Zub20, S. 5].
Die **Player Experience** ist die subjektive Wahrnehmung und Empfindung der Spieler während des Spiels. Es ist das, was Spieler fühlen und erleben, während sie das Gameplay durchlaufen. Es umfasst die Emotionen, die während des Spielens entstehen, wie Freude, Spannung, Frustration oder Erfüllung [Zub20, S. 5].

Es gibt weitere Faktoren, die einen Einfluss auf das Spielererlebnis haben und die drei zentralen Elemente bei der Schaffung der Player Experience unterstützen [Zub20, S. 8]. Das **visuelle Design** spielt eine wichtige Rolle und umfasst unter anderem die Gestaltung der Spielwelt, der Charaktere und Umgebungen sowie die Details in 3D-Modellen und auch Aspekte wie die Wahl geeigneter Schriftarten. Eng damit verbunden ist das **Setting** des Spiels. Das bezieht sich auf die allgemeine Atmosphäre, einschließlich Ästhetik, Kultur und Handlungsort des Spiels, so wie auch die Rolle, die der Spieler darin einnimmt.
Die Qualität der **Benutzeroberfläche** sowie die Integration von **Musik und das Sounddesign** tragen ebenfalls zur Immersion bei. Die **Geschichte** ist vor allem in storybasierten Spielen wichtig, ebenso wie Charaktere und ihre Motivationen, die Situationen, in denen sie sich befinden, und die Handlung, die der Spieler erlebt, sowie die Qualität der Dialoge. Zusätzlich haben auch **technische Designelemente** wie die Integration von KI oder Multiplayer-Funktionen einen Einfluss auf die Player Experience [Zub20, S. 9-10].
Im Kontext von Videospielen umfasst der Begriff ”Künstliche Intelligenz” nicht nur den Bereich des Machine Learning, sondern bezieht sich speziell auf die ”Spiele KI” (künstliche Intelligenz im Spiel). Das Hauptziel der Spiele KI besteht darin, glaubwürdige Welten zu erschaffen, um eine immersive Player Experience zu ermöglichen. Dafür werden unter anderem die zuvor genannten Design-Elemente eingesetzt. Der Schwerpunkt liegt dabei auf der Gestaltung des Verhaltens der Agenten und der Erzeugung der Illusion von Intelligenz sowie das Vermeiden von künstlicher Dummheit - Aktionen, die offensichtlich fehlerhaft wirken oder keinen Sinn im Kontext des Spiels ergeben. Beispielsweise sollten die Agenten nicht unkontrolliert in Wände laufen oder den Spieler ignorieren [Rab13, S. 3-5]. Um die gewünschte Immersion zu erreichen, ist es entscheidend, dass die Agenten reaktiv agieren können und in Echtzeit auf die Umgebung und die Handlungen des Spielers reagieren. Es gibt verschiedene Ansätze, um das zu erreichen, wie zum Beispiel endliche Zustandsautomaten und Verhaltensbäume [Rab13, S. 5-6].

Die Player Experience eines Videospiels entsteht aus der Kombination und Wechselwirkung der genannten Elemente. Game Designer gestalten dafür die Spielmechaniken und unterstützenden Elemente, um eine bestimmte Player Experience zu erzeugen. Diese Erfahrung entfaltet sich jedoch erst, wenn der Spieler aktiv mit den Spielmechaniken interagiert und das Gameplay erlebt [Zub20, S. 7].

2.2 Aufgabenbereiche im Entwicklungsstudio

Die Untersuchung der verschiedenen Aufgabenbereiche in einem Entwicklungsstudio ermöglicht es, Funktionalitäten einer Game-Engine abzuleiten, und gewährt einen weiteren Einblick in die Komponenten eines Videospiels.

In einem typischen Entwicklungsstudio sind fünf grundlegende Aufgabenbereiche vertreten: Ingenieure/Entwickler, Künstler, Game Designer, Produzenten so wie Management- und Support-Personal. Jeder dieser Bereiche kann in verschiedene Teilbereiche unterteilt werden [Gre18, S. 5]. Die konkrete Struktur variiert in Abhängigkeit von den Projektanforderungen und dem Umfang, wobei bestimmte Bereiche entfallen oder neue hinzukommen können. Je nach Art des Spiels oder des Entwicklungsprozesses werden beispielsweise Komponisten, 3D-Modellierer oder die Entwicklung einer eigenen Engine nicht benötigt. Aufgabenbereiche können auf verschiedene Personen und Abteilungen aufgeteilt werden. Es kann auch sein, dass mehrere Aufgabenbereiche von einer Person übernommen werden, was vor allem in kleinen Teams der Fall ist [Zub20, S. 12]. Die Aufgabenbereiche kooperieren eng miteinander und können ineinander übergehen.
Im Kontext der Funktionalitäten einer Game-Engine sind insbesondere die Ingenieure, Künstler und Game Designer von Interesse, da sie aktiv an der Entwicklung des Spiels beteiligt sind [Zub20, S. 13].

Ingenieure sind für die Konzeption und Umsetzung der Software zuständig. In der Regel können Ingenieure in zwei Gruppen unterteilt werden.
Laufzeit-Programmierer arbeiten an der Laufzeitkomponente. Sie implementieren die Komponenten, die das Spiel während der Laufzeit steuern und ausführen. Dazu gehören die technische Umsetzung des Spiels, wie Gameplay-Mechaniken, KI-Verhalten, Spielsteuerung und Interaktionen mit der Spielwelt, so wie die Erweiterung und Anpassung der Kernfunktionalitäten der Game-Engine.
Tool-Programmierer hingegen entwickeln Software-Werkzeuge, um den Entwicklungsprozess effizienter zu gestalten und die internen Abläufe des Teams zu unterstützen. Dazu gehört die Entwicklung von Tools, die es ermöglicht, Inhalte für das Spiel zu erstellen und zu bearbeiten, wie zum Beispiel Texturen, Modelle oder Animationen [Gre18, S. 5].

Die Künstler sind für die Produktion aller visuellen und akustischen Inhalte verantwortlich. Künstler können in verschiedene Teilbereiche unterteilt werden, wie zum Beispiel:
Konzeptkünstler übernehmen die Aufgabe, Skizzen und Bilder zu erstellen, die dem Team eine klare Vorstellung davon geben, wie das fertige Spiel aussehen soll.
Die **3D-Modellierer** sind für die Erstellung von dreidimensionalen Objekten verantwortlich, die in der virtuellen Spielwelt vorkommen, wie Charaktere, Fahrzeuge, Gebäude und andere Objekte.

Texture Artists erstellen zweidimensionale Bilder, sogenannte Texturen, die auf die Oberflächen der 3D-Modelle aufgebracht werden, um sie realistisch und detailliert aussehen zu lassen.
Die **Lighting Artists** sind dafür zuständig, die dynamischen und statischen Lichtquellen in der Spielwelt zu platzieren und zu gestalten, um eine bestimmte Atmosphäre zu erzeugen.
Animatoren sind dafür verantwortlich, Animationen für Gegenstände und Charaktere zu erstellen, um ihnen Leben und Bewegung einzuhauchen.
Sounddesigner sind für die akustische Atmosphäre zuständig und gestalten die Musik und Soundeffekte des Spiels [Gre18, S. 6].

Die Hauptaufgabe der Game Designer besteht darin, das Gameplay zu gestalten und die dafür notwendigen Mechaniken und Systeme zu entwerfen, um die gewünschte Player Experience zu erzeugen. Es gibt verschiedene Arten von Designern. Grob unterteilt sich der Beruf in System Designer und Content Designer, wobei die Grenzen dazwischen verschwimmen können [Zub20, S. 11-12].
System Designer konzentrieren sich auf die Mechaniken und Systeme des Spiels. Sie entwerfen die grundlegenden Regeln und Mechaniken, wie beispielsweise die Spielregeln, Kampfsysteme und Wirtschatfssysteme.
Content Designer konzentrieren sich auf die Gestaltung von spezifischen Spielobjekten und Aktionen. Sie sind für das Level-Design verantwortlich, das die Umgebungen definiert, in der das Spiel stattfindet, sowie das Charakterdesign, das die Identitäten und Handlungen der Charaktere im Spiel beschreibt, oder die Gestaltung der Welt [Zub20, S. 11-12].

2.3 Game-Engines

In diesem Abschnitt wird der Aufbau und die Funktionen einer Game-Engine thematisiert. Zu Beginn wird definiert, worum es sich bei einer Game-Engine handelt und welchen Zweck sie in der Spieleentwicklung erfüllt. Anschließend wird der grundlegende Aufbau einer Game-Engine beleuchtet, wobei insbesondere auf die verschiedenen Komponenten und ihre Aufgaben eingegangen wird.

2.3.1 Game-Engine Definition

Eine Game-Engine ist eine Software, die Entwickler bei der Entwicklung von Videospielen unterstützt, indem sie grundlegende Funktionalitäten bereitstellt.

Im Jahr 1993 veröffentlichte die Firma id Software den First-Person-Shooter ”Doom” und prägte den Begriff ”Game-Engine” durch den Einsatz ihrer neuen Technologie. ”Doom” zeichnete sich durch eine deutliche Trennung zwischen seinen Kernsoftwarekomponenten (wie Grafik-Rendering, Kollisionserkennung und Audio-Verarbeitung) und den spiel-spezifischen Inhalten (wie Spielwelten, Animationen und Regeln) aus. Diese Herangehensweise ermöglichte es, eine einheitliche Code-Basis zu schaffen, die für neue Spiele wiederverwendet werden konnte [Gre18, S. 11]. Diese Basis bildet die Game-Engine.

Spiele, die ohne die Verwendung einer Game-Engine entwickelt werden, verfolgen oft einen monolithischen Ansatz. Dabei werden alle spiel-spezifischen Inhalte und Kernsoftwarekomponenten direkt im Programmcode des Spiels eingebettet [Gre18, S. 1024]. Diese Komponenten sind dadurch eng miteinander verknüpft und können als ein einziger Block betrachtet werden. Somit gibt es keine klare Trennung der Komponenten. Das erschwert die Wiederverwendbarkeit des Codes und führt zu mehr Aufwand, da Spiele, einschließlich der grundlegenden technischen Funktionalitäten, immer wieder von Grund auf neu entwickelt werden müssen. Zusätzlich dazu können Änderungen Auswirkungen auf verschiedene Teile des Systems haben.
Im Gegensatz dazu basiert eine Game-Engine auf einem modularen Ansatz mit einem datengetriebenen Architekturkonzept. Dieser Ansatz ermöglicht eine klare Trennung zwischen den eigentlichen Kernsoftwarekomponenten und den spiel-spezifischen Inhalten. Die einzelnen Elemente/Komponenten werden nicht mehr direkt im Code der Engine verankert, sondern in separaten Datenformen gespeichert [Gre18, S. 12].

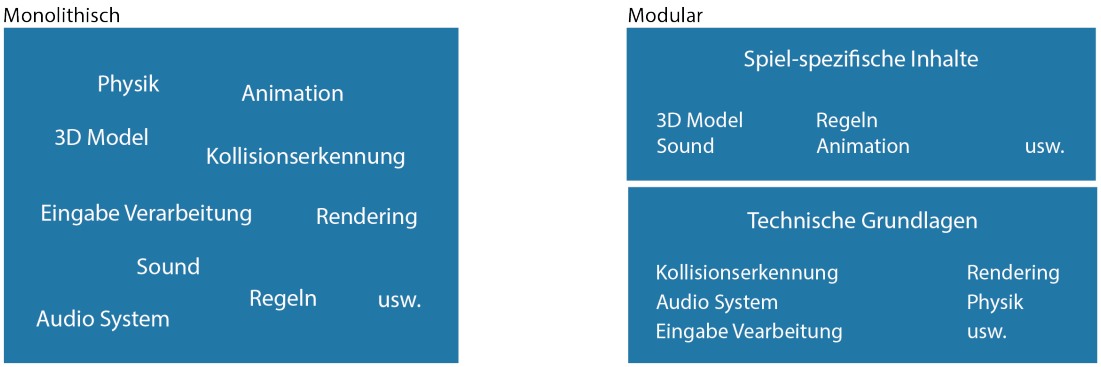

Abbildung 2.2: Beispiel für einen monolithischen und modularen Ansatz

Eine Game-Engine fungiert somit als ein Framework, das das Grundgerüst/Kernkomponenten eines Spiels bereitstellt, während die spezifischen Game-Design Elemente, die das Spiel ausmachen, wie Grafiken, Modelle, Level-Design und Regeln, in separaten Daten gespeichert werden.
Künstler und Spieledesigner können so Game-Assets in externen Programmen wie Blen-

der und Photoshop erstellen oder mit mitgelieferten Tools in der Engine. Diese Daten können dann von der Engine geladen und verwendet werden [Gre18, S. 1024-1025].

Wenn Entwickler ein neues Spiel erstellen wollen, können sie durch die modulare Struktur die Kernfunktionalitäten der Engine als Basis wiederverwenden. Durch das Erstellen von neuen spiel-spezifischen Inhalten, die von der Engine geladen werden können, entstehen so neue Spiele, ohne Veränderungen am Engine-Code vornehmen zu müssen. Dies fördert die Flexibilität und ermöglicht eine effiziente Entwicklung, da Kernkomponenten nicht für jedes Spiel komplett neu entwickelt werden müssen. Der modulare Ansatz verbessert zusätzlich Aspekte wie Wartung und Zusammenarbeit.
Die Abgrenzung zwischen der spiel-spezifischen Ebene und der technischen Grundlage kann jedoch unscharf sein. Einerseits könnte direkt in der Engine definiert sein, wie ein bestimmter Charakter in der Spielwelt gerendert wird und wie seine Bewegungen animiert werden. Andererseits kann es auch sein, dass die Engine nur allgemeine Rendering-Funktionen bereitstellt, während die spezifischen Eigenschaften und Bewegungen des Charakters ausschließlich in den Spieldaten definiert sind [Gre18, S. 12]. Das ist von der Struktur der Engine abhängig.

Eine Game-Engine wird für bestimmte Genre und Hardware-Plattformen optimiert, weshalb sie nicht absolut allgemein sein kann. Je allgemeiner eine Game-Engine ist, desto weniger optimal ist sie für die Ausführung eines bestimmten Spiels auf einer bestimmten Plattform. Je besser die Engine zu den Anforderungen des Spiels passt, desto positiver wirkt sich dies auf das Ergebnis und den Entwicklungsprozess aus [Gre18, S. 12-13].
Studios wie zum Beispiel Electronic Arts und Naughty Dogs entwickeln und verwenden ihre hauseigene Game-Engine, um so die Kontrolle über die technische Umsetzung ihrer Spiele zu behalten. Heute können Entwickler Game-Engines wie Unity oder Unreal Engine lizenzieren, um damit Spiele zu entwickeln. Viele kommerziell lizenzierte Game-Engines, wie Source, Unreal Engine 4 und CRYENGINE, haben ihren Ursprung als firmeneigene Engines [Gre18, S. 35-36].

2.3.2 Komponenten einer 3D-Game-Engine

Die Komponenten einer Game-Engine lassen sich in zwei Kategorien unterteilen, dem **Toolset** und der **Laufzeitkomponente** [Gre18, S. 38].

Das Toolset ist eine Sammlung von Entwicklerwerkzeugen, die von Entwicklern verwendet werden können, um Inhalte für das Spiel zu erstellen und zu bearbeiten. Dazu gehören beispielsweise das Erstellen und Bearbeiten von 3D-Modellen, Texturen, Animationen und das Erstellen von Landschaften. Diese Tools, darunter Bildbearbeitungsprogramme, Level-Editoren, Animationstools und Audio-Editoren, können sowohl Teil der Engine sein als auch externe Softwareanwendungen wie Photoshop und Maya [Gre18, S. 65-67].

Die Laufzeitkomponente ist der Teil der Spiele-Engine, der während der Ausführung des Spiels aktiv ist. Dieser Teil der Engine ist für die Ausführung des Spiels verantwortlich, während der Spieler es spielt. Die Laufzeitkomponente enthält die Kernfunktionalitäten der Engine, wie zum Beispiel das Grafik-Rendering, die Physiksimulation, die KI-Verarbeitung und Eingabeverarbeitung.
Die Laufzeitkomponente enthält alle wichtigen Bestandteile, die für die Ausführung eines Spiels erforderlich sind. Diese Komponenten sind als Schichten organisiert und können sich in ihrer genauen Implementierung zwischen den verschiedenen Game-Engines unterscheiden. Abbildung 2.3 zeigt eine vereinfachte Darstellung der verschiedenen Schichten. Jede Schicht erfüllt eine spezifische Aufgabe.

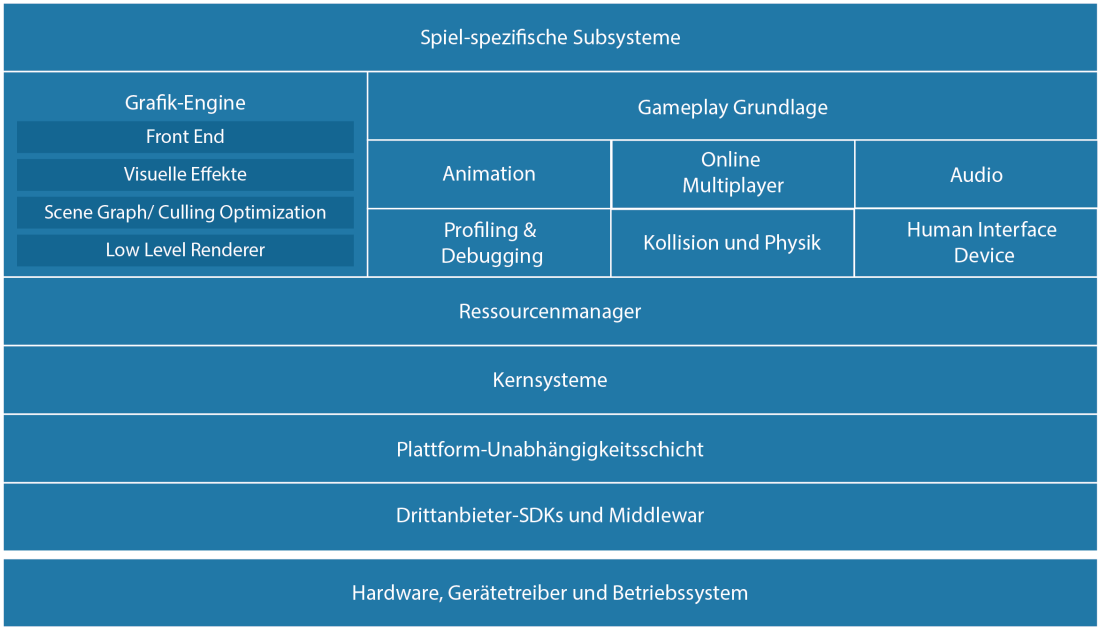

Abbildung 2.3: Vereinfachte Darstellung der Laufzeitkomponente einer Game-Engine, in Anlehnung an [Gre18, S. 39]

Hardware, Gerätetreiber und Betriebssystem repräsentiert die Hardware, die Gerätetreiber und das Betriebssystem, auf dem das Videospiel/die Engine ausgeführt werden soll, wie zum Beispiel Spielekonsolen und Smartphones [Gre18, S. 38-40].

Drittanbieter-SDKs und Middleware werden von Game-Engines häufig genutzt, um vorgefertigte Softwarekomponenten von externen Anbietern einzubinden und so zusätzliche Funktionen zu implementieren. Diese Software Development Kits (SDKs) und Middleware bieten abstrahierte Schnittstellen (Application Programming Interfaces (APIs)) über die Entwickler auf vorgefertigte Funktionen zugreifen können [Gre18, S. 40].

Einige Beispiele:
Grafik-Engines verwenden SDKs wie Glide, OpenGL, DirectX und Vulkan, um 3D-Grafiken darzustellen. Für Physiksimulationen und Kollisionserkennung werden unter anderem SDKs wie Havok, PhysX und Open Dynamics Engine eingesetzt.
Der Einsetzt der jeweiligen SDKs kann von der Anwendung und der Zielplattform abhängig sein.

Die **Plattform-Unabhängigkeitsschicht** ermöglicht, dass eine Game-Engine auf verschiedenen Hardwareplattformen wie beispielsweise PlayStation, Computern und Smartphones funktioniert. Sie stellt Schnittstellen zwischen der Engine und den Plattformen bereit und dient als Vermittler, der es der Engine ermöglicht, auf verschiedenen Systemen zu laufen, ohne dass die Engine für jede Plattform angepasst werden muss [Gre18, S. 43].

Game-Engines verwenden eine Vielzahl an Softwareutilities, die als **Kernsysteme** zusammengefasst werden können. Dazu gehören Funktionen wie Speicherverwaltung, mathematische Bibliothek, Datenstrukturen und Algorithmen [Gre18, S. 44].

Der **Ressourcenmanager** hat die Aufgabe, Game-Assets und andere Ressourcen zu verwalten und den Zugriff darauf zu ermöglichen. Dazu gehören beispielsweise 3D-Modelle, Texturen, Animationen und Sound-Dateien [Gre18, S. 45].

Die **Grafik-Engine** ist die umfangreichste und komplexeste Komponente der Engine. Sie ist für die Erzeugung der visuellen Elemente im Spiel verantwortlich. Dieser Prozess beinhaltet die Umwandlung digitaler Daten in Bilder. Hierbei kommen Grafik-SDKs wie zum Beispiel OpenGL oder DirectX zum Einsatz, um 2D- und 3D-Grafiken zu generieren und auf dem Bildschirm darzustellen.
Die Grafik-Engine kann in vier Schichten unterteilt werden: den Low-Level-Renderer, den Scene Graph/Culling Optimization Layer, Visual Effects und das Front-End [Gre18, S. 45-49].

Der Low-Level-Renderer übernimmt grundlegende Rendering-Funktionen, insbesondere das schnelle Rendern geometrischer Primitiven (grundlegende geometrische Formen), unabhängig davon, ob sie sich im Sichtbereich befinden. Ein zentraler Bestandteil des Low-Level-Renderers ist die Grafik-API, welche die Kommunikation zwischen der Engine und der Grafikhardware ermöglicht.

Die Scene Graph/Culling-Optimization-Schicht hat die Aufgabe, die Anzahl der zu rendernden Primitiven, basierend auf ihrer Sichtbarkeit, zu begrenzen, wodurch die Performance gesteigert wird.
Ein Scene Graph ist eine Datenstruktur, der die Hierarchie und Beziehungen von Objekten in einer Szene darstellt. Das umfasst auch Informationen wie Position, Rotation und Skalierung.

Bei der Culling-Optimization handelt es sich um eine Sichtbarkeitsprüfung, die feststellt, welche Objekte sich im sichtbaren Bereich befinden und gerendert werden müssen, und welche nicht. Dies dient dazu, die Rechenleistung der Hardware effizienter zu nutzen und die Frames per Sekund (FPS) zu erhöhen. Dafür gibt es verschiedene Techniken wie, Frustum Culling (Ausschluss von Objekten außerhalb des Sichtfelds), Occlusion Culling (Ausschluss von Objekten, die von anderen verdeckt werden) und Level of Detail (LOD), bei dem verschiedene Detailstufen eines Modells verwendet werden, abhängig von der Entfernung der Kamera.
Abbildung 2.4 verdeutlicht die Culling Techniken. Die blauen Objekte befinden sich im Sichtfeld und werden gerendert. Die roten Objekte werden überdeckt (Occlusion Culling) und die gelben Objekte befinden sich außerhalb des Sichtfeldes (Frustum Culling) und werden dementsprechend nicht gerendert.

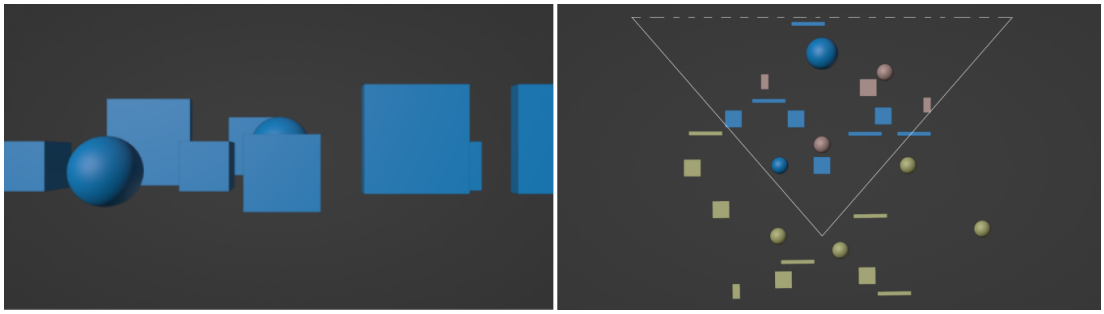

Abbildung 2.4: Culling-Optimization Techniken

Der Visual Effects Layer ist dafür zuständig, visuelle Effekte zu erzeugen. Hierzu gehören unter anderem Partikelsysteme für Effekte wie Rauch oder Feuer, Dekalsysteme (Aufkleber-System) zur Platzierung von Details wie Einschusslöchern und Fußabdrücken auf Oberflächen sowie Techniken wie Light Mapping, dynamische Schatten und Full-Screen Post-Effects.

Die Front-End-Schicht ist für die Darstellung von 2D-Grafiken über der 3D-Szene verantwortlich. Dazu gehören Elemente wie das Heads-up-Display (HUD), das während des Spiels wichtige Informationen wie Lebensanzeigen anzeigt und die Grafische Benutzeroberfläche (GUI), über die der Spieler beispielsweise ein Inventar verwalten kann. Diese Schicht umfasst außerdem das In-Game-Cinematic (IGC)-System, das kinematische Sequenzen innerhalb des Spiels ermöglicht, sowie das Full-Motion-Video (FMV)-System, das vorher aufgezeichnete Vollbildfilme abspielt.

Die **Profiling- und Debugging** Ebene umfasst Tools, die Entwicklern helfen, die Performance des Spiels zu überwachen und Probleme zu beheben [Gre18, S. 50].

Die **Kollision- und Physik-Engine** hat die Aufgabe, die Interaktion von Objekten innerhalb der Spielwelt realistisch zu simulieren. Sie ermöglicht die Erkennung von Kollisionen zwischen Objekten, berechnet physikalische Kräfte und Bewegungen sowie die Umsetzung von Regeln, die die Bewegung und Interaktion beeinflussen [Gre18, S. 51-52].

Die **Animations**-Schicht ist dafür zuständig, die Bewegungen/Animationen von Charakteren, Objekten und anderen Elementen in der Spielwelt zu steuern. Diese Schicht umfasst auch die Erstellung und Verwaltung von Animationen sowie das Abspielen von Animationen oder die Steuerung der Übergänge zwischen ihnen [Gre18, S. 39].

Die **Human Interface Devices** Schicht verwaltet Eingabegeräte wie Tastaturen, Controller, Lenkräder oder Tanzmatten. Sie erfasst die Eingaben des Spielers über das Eingabegerät und übersetzt sie in Befehle für die Engine, wodurch Interaktionen ermöglicht werden [Gre18, S. 53-54].

Die **Audio-Engine** ist für die Wiedergabe und Verwaltung akustischer Elemente verantwortlich. Zu ihren Funktionen gehören die Umsetzung von räumlichem Klang, die nahtlose Integration von Musik sowie dynamische Anpassungen. Diese Aufgaben umfassen die Steuerung von Soundeffekten, die Einbindung von Hintergrundmusik und die Verwaltung von Dialogen im Spiel [Gre18, S. 54].

Die **Online Multiplayer/Netzwerk** Schicht beinhaltet Funktionalitäten, die die Netzwerkkommunikation für Multiplayer-Spiele ermöglicht [Gre18, S. 55].

Die **Gameplay Grundlagen** Schicht stellt grundlegende Funktionalitäten zur Umsetzung der Spiellogik bereit. Sie ermöglicht die Definition und Verwaltung verschiedener Arten von Spielobjekten wie NPCs, Fahrzeugen und Waffen. Durch ein implementiertes Event-System wird die Kommunikation zwischen diesen Objekten ermöglicht. Zusätzlich enthält diese Schicht auch das Skripting-System sowie grundlegende Elemente für KIs-Implementierungen, wie beispielsweise das Pathfinding [Gre18, S. 56-58].

Die **spiel-spezifische Subsysteme** umfassen die Funktionen und Mechaniken, die spezifisch für das jeweilige Spiel sind. Hier arbeiten Programmierer und Designer zusammen, um Systeme zu entwickeln, wie zum Beispiel Steuerung des Spielercharakters, KI für NPCs, Wirtschaftssystem und andere spiel-spezifische Funktionen. Diese Systeme sind vielfältig und individuell für jedes Spiel [Gre18, S. 58-59].

3 Konzept

Obwohl Game-Engines ähnliche Grundstrukturen aufweisen, können sie sich stark voneinander unterscheiden. Je besser eine Game-Engine zu den Anforderungen und Bedürfnissen der Entwickler und des Projekts passt, desto positiver wirkt sich dies auf das Ergebnis sowie den Entwicklungsprozess aus [Gre18, S. 12-13].
In diesem Kapitel werden Kriterien identifiziert, um Game-Engines miteinander vergleichen zu können. Zudem wird eine Strategie entwickelt, um diese Kriterien systematisch zu evaluieren, mit dem Ziel, eine geeignete Game-Engine für ein spezifisches Projekt auszuwählen.

3.1 Identifikation von Kriterien

Die Auswahl der passenden Game-Engine erfordert einen sorgfältigen Vergleich. In diesem Abschnitt werden Kriterien identifiziert, die für den Vergleich von Game-Engines relevant sein können.

3.1.1 Aspekte der Softwarequalität nach ISO 25010

Verschiedene Ansätze stehen zur Verfügung, um die Qualität von Software zu bewerten. Die ISO 25010 ist eine internationale Norm für Qualitätskriterien im Bereich der Softwareentwicklung. Innerhalb dieser Norm werden Qualitätsmerkmale und Kriterien definiert, um Softwareprodukte zu bewerten. Diese Norm ist in acht Hauptmerkmale unterteilt, die sich in weitere Untermerkmale aufteilen [Int11].

Abbildung 3.1: Überblick der Hauptkategorien der ISO 25010

Diese Kriterien sind nicht ausschließlich für die Überprüfung der Qualität einer Game-Engine geeignet, sondern können ebenso zur Bewertung der einzelnen Funktionalitäten sowie des finalen Spiels herangezogen werden.

Die **Funktionale Eignung** bezieht sich darauf, ob die Software die erforderlichen Funktionen bereitstellt und diesen den festgelegten Spezifikationen und Anforderungen entspricht. Zum Beispiel sollte eine Game-Engine alle notwendigen Funktionen für die Erstellung eines Spieles bereitstellen.
Die **Leistungseffizienz** betrachtet die Leistung der Software hinsichtlich des Zeitverhaltens, der Ressourcennutzung und der Kapazität. Zum Beispiel sollte die Engine in der Lage sein, eine Vielzahl von komplexen 3D-Modellen in Echtzeit zu rendern, ohne dabei die Leistung der Hardware übermäßig zu belasten.
Kompatibilität bezieht sich auf die Fähigkeit der Software, mit anderen Systemen und Komponenten zu interagieren, ohne dabei Konflikte zu verursachen.
Die **Benutzerfreundlichkeit** umfasst verschiedene Aspekte. Die Benutzeroberfläche sollte klar und angemessen gestaltet sein. Die Software sollte leicht bedienbar und einfach zu erlernen sein, sei es durch eine intuitive Gestaltung oder klare Anleitungen. Sicherheitsmechanismen sollten vorhanden sein, um Nutzer davon abzuhalten, schädliche Aktionen durchzuführen. Zudem werden die visuelle Gestaltung der Benutzeroberfläche und die Zugänglichkeit betrachtet.
Zuverlässigkeit bezieht sich auf den Umgang der Software mit Ausfällen, Fehlern und unerwarteten Situationen. Eine zuverlässige Software sollte keine Ausfälle oder Fehler aufweisen. Sie sollte fehlertolerant sein und bei auftretenden Fehlern nicht abstürzen oder beschädigt werden. Zusätzlich wird die Wiederherstellbarkeit betrachtet, die bewertet, wie effektiv die Software nach einem Fehler oder einem Absturz in einen stabilen und funktionsfähigen Zustand zurückkehren kann. Ein Beispiel hierfür sind Mechanismen zur Wiederherstellung des Zustandes, wie das automatische Erstellen von Sicherungskopien.
Sicherheit bezieht sich auf Aspekte wie den Umgang mit sensiblen Daten und den Schutz vor unbefugtem Zugriff.
Wartbarkeit bewertet verschiedene Aspekte, darunter die Modularität, Wiederverwendbarkeit, Analysierbarkeit, Modifizierbarkeit und Testbarkeit der Software.
Portabilität beurteilt die Fähigkeit der Software, auf unterschiedlichen Plattformen oder Umgebungen zu laufen. Eine Game-Engine sollte beispielsweise auf verschiedenen Systemen wie Windows und Linux für die Entwicklung einsatzfähig sein. Diese Bewertung bezieht sich auch auf die Installierbarkeit der Software sowie deren Austauschbarkeit. [Int11]

3.1.2 Game-Engine spezifische Kriterien

Die Identifikation spezifischer Vergleichskriterien für Game-Engines erfolgte durch eine Analyse verschiedener Quellen, um einen umfassenden Überblick über relevante Aspekte zu gewinnen. Die Identifikation basiert auf drei Hauptquellen: Eine umfassende Analyse von Vergleichskriterien [Ali16] (Quelle 1), eine weitere Arbeit, die ein Framework zur Auswahl von Game-Engines für Serious Applications vorstellt [Pet12](Quelle 2), sowie die Informationen zu Funktionen und Komponenten aus Kapitel 2, die sich auf [Gre18] stützen (Quelle 3). Tabelle 3.1 zeigt die in den Quellen genannten Kriterien.

Quelle 1	Quelle 2	Quelle 3
Audiovisuelle Wiedergabetreue	Audiovisuelle Wiedergabetreue	
2D/ 3D Unterstützung		
Grafik-Rendering	Rendering (Texturen, Beleuchtung, Schatten, Visuelle Effekte)	Grafik (Visual Effekte, Frontend, Culling-Optimization)
	Animation	Animation
		Cinematic-Tools
	Sound	Sound
Scripting	Scripting	Scripting
Künstliche Intelligenz	Künstliche Intelligenz	Künstliche Intelligenz
Physik	Physik	Physik und Kollision
Import/ Export von Ressourcen	Import/ Export von Ressourcen	Ressourcenmanagement
Inhaltserstellung		
Welt-(Level-) Editor		Welt Editor
Developer Toolkits	Developer Toolkits	Developer Toolkits
Zugänglichkeit	Zugänglichkeit	
Lernkurve	Lernkurve	
	Dokumentation und Support	
Lizenzierung	Lizenzierung	
	Kosten	
Netzwerkfunktionalitäten	Netzwerkfunktionalitäten (Client-Server/Peer-to-Peer)	Netzwerkfunktionalitäten
Bereitstellungsplattformen		Bereitstellungsplattformen
Entwicklungsplattformen		Entwicklungsplattformen
	Multiplattform-Unterstützung	
		Eingabegeräte
		Profiling und Debugging

Tabelle 3.1: Identifikation von Game-Engine spezifischen Vergleichskriterien

Die extrahierten Informationen wurden zusammengefasst, kategorisiert und präzisiert, um einen Überblick zu schaffen. Basierend auf den genannten Arbeiten, dem Abgleich mit den Komponenten einer Game-Engine und unter Berücksichtigung der Qualitätskriterien nach ISO 25010 ergeben sich folgende zentrale Kriterien für den Vergleich von Game-Engines. Diese werden im Folgenden aufgeführt und in Abschnitt 3.2 detailliert erläutert. Zusätzlich dazu werden Editoren zum Erstellen von Assets separiert.

Kategorie	Kriterie
Zugänglichkeit	Plattformkompatibilität, Erlernbarkeit, Lizenzmodell und Kosten
Entwicklungsumgebung und Workflow	Unterstütze Genre, Programmiersprache und Skripting, Plattformunterstützung, Kompatibilität, Versionskontrolle und kollaborative Arbeit, Profiling- und Debugging-Tools, Oberfläche und Szenen-Editor, Quellcode Zugriff
Ressourcen	Marketplace, Asset-Management und Workflow
Grafik	Grafik-SDK, Licht und Schatten, Materialien und Objekte, Visuale Effekte und Partikel, UI-Editor, Culling-Optimierung
Physik und Kollision	Physik-SDK, Umgang mit Physik und Kollision
Animation	Animations-Editor, Umgang mit Animation, Cinematic-Tools
Sound	Sound-Editor, Umgang mit Sound
World building	World building-Tools
Künstliche Intelligenz	KI-Techniken
Netzwerkfunktionalitäten	Netzwerkfunktionalitäten
Funktionalität	Funktionalität
Leistung	Leistung
Benutzerfreundlichkeit	Benutzerfreundlichkeit

Tabelle 3.2: Kriterien zum Vergleichen von Game-Engines

3.2 Kriterien zum Vergleichen von Game-Engines

Die nachfolgenden Kriterien wurden in Abschnitt 3.1.2 entwickelt und gliedern sich in 13 Hauptkategorien. Diese umfassen 10 Vergleichskategorien, die wiederum in spezifischere Kriterien unterteilt sind. Innerhalb dieser Vergleichskriterien werden Game-Engine spezifische Funktionalitäten und Eigenschaften betrachtet. Ergänzt werden diese durch drei Bewertungskriterien, die sich aus Software-Qualitätskriterien zusammensetzen.

Abbildung 3.2: Überblick der Hauptvergleichskategorien

3.2.1 Zugänglichkeit

Plattformkompatibilität und Installation: Dieses Kriterium untersucht die unterstützten Plattformen, auf denen die Engine zur Spieleentwicklung genutzt werden kann. Es bezieht auch den Installationsprozess und die Projekterstellung mit ein.
Erlernbarkeit: Erlernbarkeit bewertet die Verfügbarkeit von Dokumentationen, Tutorials und anderen Ressourcen, die Entwicklern helfen, sich in der Engine zurechtzufinden.
Lizenzmodell und Kosten: Hier wird das Lizenzmodell der Engine und die damit verbundenen Kosten betrachtet, so wie mögliche Zugriffseinschränkungen.

3.2.2 Entwicklungsumgebung und Workflow

Unterstütze Genre: Dieses Kriterium ermittelt, welche Genres von der Engine unterstützt werden, darunter 2D- und 3D-Spiele so wie auch Extended Reality (XR).
Programmiersprache und Skripting: Dieses Kriterium identifiziert die verwendete Programmiersprache und bewertet das Skripting-System der Engine.
Plattformunterstützung: Untersucht die verfügbaren Zielsysteme, darunter Desktop, Konsole und Mobil-Plattformen für die Spieleentwicklung und bewertet die Fähigkeit der Engine zur Cross-Plattform-Entwicklung.

Kompatibilität: Dieses Kriterium untersucht die Möglichkeiten zur Migration von Projekten zwischen verschiedenen Versionen der Engine, um sicherzustellen, dass Projekte mit Updates der Engine kompatibel bleiben, insbesondere im Hinblick auf große Versionssprünge, wie zum Beispiel von Unreal Engine 3 auf Unreal Engine 4. Es bezieht auch den Aktualisierungszyklus der Engine und die historische Entwicklung mit ein, um sicherzustellen, dass sie zukunftsfähig ist.
Versionskontrolle und kollaborative Arbeit: Untersucht die Fähigkeit der Engine zur Integration von Versionskontrollsystemen und Kollaborationstools für Entwicklerteams, wie z.B. Git, um eine effiziente Zusammenarbeit und die Nachverfolgung von Code-Änderungen zu ermöglichen.
Profiling- und Debugging-Tools: Bewertet die Verfügbarkeit von Profiling- und Debugging-Werkzeuge zur Code-Analyse, um Fehler zu beheben und die Performance zu optimieren.
Oberfläche und Szenen-Editor: Bewertet die Oberfläche der Engine sowie die Möglichkeiten zur Erstellung und Bearbeitung von Szenen.
Quellcode Zugriff: Beurteilt die Möglichkeit zum Zugriff auf den Quellcode der Engine, um diesen einzusehen oder zu modifizieren.

3.2.3 Ressourcen

Marketplace: Die Bewertung bezieht sich auf die Verfügbarkeit und den Umfang des Asset-Stores sowie die Integration von Ressourcen daraus.
Asset-Management und Workflow: Dieser Aspekt bewertet das allgemeine Asset-Management innerhalb der Engine und schließt Aspekte wie Import, Export und die Betrachtung der unterstützten Dateiformate der Engine mit ein.

3.2.4 Grafik

Grafik-SDK: Dieses Kriterium identifiziert das verwendete Grafik-SDK der Engine.
Licht und Schatten: Dieses Kriterium bewertet die Möglichkeiten zur Erstellung von Licht- und Schatteneffekten sowie Beleuchtungssystemen wie zum Beispiel Global Illumination.
Materialien und Objekte: Dieses Kriterium bewertet die Möglichkeiten zur Erstellung von Materialien und Objekten innerhalb der Engine.
Visuale Effekte und Partikel: Dieses Kriterium bewertet die Möglichkeiten zur Erstellung von visuellen Effekten und Partikelsystemen innerhalb der Engine.
UI-Editor: Dieses Kriterium untersucht die verfügbaren Werkzeuge zur Erstellung und Bearbeitung von Benutzeroberflächen innerhalb der Engine, einschließlich HUDs und GUIs.

Culling-Optimierung: Untersucht die Verwendung von Culling-Techniken sowie LOD-Funktionen zur Optimierung der Performance.

3.2.5 Physik und Kollision

Physik-SDK: Dieses Kriterium identifiziert das verwendete Physik-SDK der Engine.
Umgang mit Physik und Kollision: Dieses Kriterium bewertet, wie physikalische Eigenschaften simuliert werden können, einschließlich des Umgangs mit Kollisionserkennung.

3.2.6 Animation

Animations-Editor: Dieses Kriterium untersucht die verfügbaren Werkzeuge zur Erstellung und Bearbeitung von Animationen innerhalb der Engine. Beispiele hierfür sind der Einsatz von Key-Frame-Animationen sowie die Unterstützung von Skelettanimationen für Charaktere.
Umgang mit Animation: Untersucht, wie Animationen gesteuert werden können, einschließlich der Möglichkeiten zur Gestaltung von Übergängen und anderen Animationseffekten.
Cinematic-Tools: Dieses Kriterium bezieht sich auf Werkzeuge für die Erstellung von kinematografischen Szenen und Zwischensequenzen.

3.2.7 Sound

Sound Editor: Dieses Kriterium untersucht die Werkzeuge, die in der Engine verfügbar sind, um akustische Elemente zu erstellen und zu bearbeiten.
Umgang mit Sound: Untersucht die Handhabung von Musik und Soundeffekten, einschließlich 3D- und 2D-Sounds sowie spezieller Effekte. Dieses Kriterium bezieht sich darauf, wie akustische Elemente wie Dialoge, Hintergrundmusik und Effekte in das Spiel implementiert werden können.

3.2.8 World building

Erstellung von Landschaften: Untersucht die verfügbaren Werkzeuge zur Erstellung und Gestaltung von Landschaften.

3.2.9 Künstliche Intelligenz

KI Editor: Dieses Kriterium untersucht die verfügbaren Werkzeuge zur Erstellung und Bearbeitung von KI-Verhalten, um Agenten zu steuern.

3.2.10 Netzwerk

Netzwerkfunktionalitäten: Dieses Kriterium untersucht die verfügbaren Werkzeuge zur Umsetzung von Netzwerk- und Mehrspielerfunktionen.

3.2.11 Funktionalität

Dieses Kriterium beurteilt, ob die betrachtete Funktionalität die geforderten Funktionen und Eigenschaften bereitstellt, um die beabsichtigten Aufgaben zu erfüllen. Die Bewertung kann sowohl für jedes Kriterium einzeln als auch für die Engine durchgeführt werden.

3.2.12 Leistung

Hier wird die Leistungsfähigkeit der betrachteten Funktionalität bewertet, einschließlich Aspekte wie Geschwindigkeit, mögliche Konflikte und deren Bewältigung sowie die Skalierbarkeit bei steigender Kapazität. Die Bewertung kann sowohl für die separate Funktionalität als auch für die Engine erfolgen.

3.2.13 Benutzerfreundlichkeit/ Bedienung

Dieses Kriterium betrachtet die Benutzerfreundlichkeit der jeweiligen Funktionalität. Der Fokus liegt darauf, wie leicht verständlich, zugänglich und einfach zu bedienen die Funktionalität ist, um die beabsichtigten Aufgaben zu erfüllen. Die Bewertung kann sowohl für jedes Kriterium einzeln als auch für die Engine angewendet werden.

3.3 Strategie zur Evaluierung von Game-Engines

In diesem Abschnitt wird eine Strategie zur Evaluierung der Vergleichskriterien vorgestellt, die darauf abzielt, die am besten geeignete Game-Engine für ein Projekt zu identifizieren. Die Strategie basiert auf einer Nutzwertanalyse, die es ermöglicht, verschiedene Alternativen anhand von definierten Kriterien zu bewerten. Dabei werden

Punkte entsprechend ihrer Leistung vergeben, um eine fundierte Entscheidung treffen zu können [Sol23].
Die Vergleichsmatrix in Tabelle 3.3 bietet einen Überblick über alle zu bewertenden Kriterien, die in Abschnitt 3.2 erläutert wurden, und ermöglicht eine systematische Bewertung einer Game-Engine. Hierbei erfolgt die Bewertung der engine-spezifischen Kriterien anhand der definierten Bewertungskriterien (siehe Abbildung 3.2).

Zu Beginn müssen die Projektanforderungen festgelegt werden, um die relevanten Kriterien zu identifizieren. Nach der Festlegung der Anforderungen erfolgt eine Gewichtung der einzelnen Kriterien entsprechend ihrer Bedeutung für das Projekt. Einige Kriterien können aufgrund ihrer Relevanz stärker gewichtet werden oder wegfallen, zusätzlich dazu können Ausschlusskriterien definiert werden. Die Festlegung der Gewichtung erfolgt anhand einer Skala, wie in Tabelle 3.4 dargestellt. Die Vergleichsmatrix kann bei Bedarf erweitert werden, um spezifische Funktionalitäten zu berücksichtigen. Anschließend werden die Alternativen, die zur Wahl stehen, ausgewählt.

Im nächsten Schritt werden die Kriterien bewertet. Dafür werden Informationen zu den jeweiligen Kriterien recherchiert. Die Informationen können aus verschiedenen Quellen bezogen werden, wie zum Beispiel: Berichte und Testergebnisse, technische Dokumentation und durch die Umsetzung von Prototypen, wobei Beobachtung und Messung einbezogen werden können. Dadurch können Vor- und Nachteile, so wie auch Unterschiede und Gemeinsamkeiten der Engines festgestellt werden. Für die Kriterien werden Punkte anhand einer Skala, wie sie in Tabelle 3.4 festgelegt sind, vergeben. Es ist zu beachten, dass einige Kriterien nicht in Bezug auf die drei Aspekte bewertet werden können, da es sich um Eigenschaften und nicht um Funktionen handelt. Damit diese dennoch in die Wertung einfließen können, wird ein Punkt für jeden der drei Aspekte vergeben, wenn die jeweilige Eigenschaft den Anforderungen des Projektes entspricht.

Nachdem die Bewertungen für die drei Aspekte für alle relevanten Kriterien erfasst wurden, erfolgt die Berechnung und Interpretation der Gesamtpunktzahlen. Tabelle 3.5 beschreibt die notwendigen Rechnungen. Die Engine mit der höheren gewichteten Gesamtpunktzahl ist besser für das Projekt geeignet. Durch die Aufteilung der Vergleichs- und Bewertungskriterien können auch einzelne Aspekte der Engines miteinander verglichen werden. Ein höherer Wert entspricht einer besseren Eignung für das Projekt.

Kriterium	Gewichtung	Engine				
		Funktionalität	Leistung	Benutzerfreundlichkeit	Gesamtpunkte	Gewichtete Punkte
Zugänglichkeit						
Plattformkompatibilität und Installation	W	F	L	B	P	WP
Erlernbarkeit	W	F	L	B	P	WP
Lizenzmodell und Kosten	W	F	L	B	P	WP
Entwicklungsumgebung und Workflow						
Unterstütze Genre	W	F	L	B	P	WP
Programmiersprache und Skripting	W	F	L	B	P	WP
Plattformunterstützung	W	F	L	B	P	WP
Kompatibilität	W	F	L	B	P	WP
Versionskontrolle und kollaborative Arbeit	W	F	L	B	P	WP
Profiling- und Debugging-Tools	W	F	L	B	P	WP
Oberfläche und Szenen-Editor	W	F	L	B	P	WP
Quellcode Zugriff	W	F	L	B	P	WP
Ressourcen						
Marketplace	W	F	L	B	P	WP
Asset-Management und Workflow	W	F	L	B	P	WP
Grafik						
Grafik-SDK	W	F	L	B	P	WP
Licht und Schatten	W	F	L	B	P	WP
Materialien und Objekte	W	F	L	B	P	WP
Visuale Effekte und Partikel	W	F	L	B	P	WP
UI-Editor	W	F	L	B	P	WP
Culling-Optimierung	W	F	L	B	P	WP
Physik und Kollision						
Physik-SDK	W	F	L	B	P	WP
Umgang mit Physik und Kollision	W	F	L	B	P	WP
Animation						
Animations-Editor	W	F	L	B	P	WP
Umgang mit Animation	W	F	L	B	P	WP
Cinematic-Tools	W	F	L	B	P	WP
Sound						
Sound-Editor	W	F	L	B	P	WP
Umgang mit Sound	W	F	L	B	P	WP
World building						
Erstellung von Landschaften	W	F	L	B	P	WP
Künstliche Intelligenz						
KI-Editor	W	F	L	B	P	WP
Netzwerk						
Netzwerkfunktionalitäten	W	F	L	B	P	WP
Gesamtpunkte		**GF**	**GL**	**GB**	**GP**	**GWP**

Tabelle 3.3: Vergleichsmatrix - Überblick der Vergleichs- und Bewertungskriterien

Skala	Bedeutung
Gewichtung	
0 - Nicht relevant:	Das Kriterium trägt nicht zur Gesamtbewertung der Engine bei, da es in diesem Kontext irrelevant oder nicht erforderlich ist.
1 - Neutral:	Das Kriterium hat eine neutrale Bedeutung und wird in der Bewertung gleich gewichtet.
2 - Relevant:	Das Kriterium besitzt eine signifikante Relevanz für die Umsetzung des Projektes und wird stark gewichtet.
Funktionalität	
0 - Schlecht:	Die Engine erfüllt die funktionalen Anforderungen nicht oder weist erhebliche Defizite auf, wodurch festgelegte Ziele nicht erreicht werden können.
1 - Gut:	Die Engine erfüllt die funktionalen Anforderungen und weist keine oder nur geringfügige Mängel auf. Festgelegte Ziele können erreicht werden.
2 - Sehr gut:	Die Engine erfüllt die funktionalen Anforderungen und bietet erweiterte Funktionen, die die Zielerreichung wesentlich vereinfachen.
Leistung	
0 - Schlecht:	Die Engine weist erhebliche Leistungsprobleme auf, ist langsam, instabil und von zahlreichen Konflikten betroffen.
1 - Gut:	Die Leistung der Engine ist zufriedenstellend und weist keine oder nur geringfügige Schwächen auf.
2 - Sehr gut:	Die Engine zeigt eine hervorragende Leistung, sie zeichnet sich durch Schnelligkeit, Stabilität und Effizienz aus und zeigt keine Konflikte.
Benutzerfreundlichkeit	
0 - Schlecht:	Die Benutzerfreundlichkeit ist erheblich eingeschränkt, die Funktionalität ist schwer verständlich und kompliziert zu bedienen. Es erfordert umfangreiche Recherchen, um ein Ziel zu erreichen.
1 - Gut:	Die Benutzerfreundlichkeit ist zufriedenstellend, mit wenigen oder keinen Verständnis-/Bedienungsproblemen. Es erfordert dennoch Recherche, um ein Ziel zu erreichen.
2 - Sehr gut:	Die Engine ist äußerst benutzerfreundlich, leicht verständlich und einfach zu bedienen. Wenig Recherche ist erforderlich, um ein Ziel zu erreichen.

Tabelle 3.4: Punkteskala für die Bewertung der Kriterien

Kriterium	**Gewichtung**	**Engine**				
		Funktionalität	Leistung	Benutzerfreundlichkeit	Gesamtpunkte	Gewichtete Punkte
Kategorie						
Kriterium	W_K	F_K	L_K	B_K	$P_K = F_K + L_K + B_K$	$WP_K = W_K \times P_K$
...						
Gesamtpunkte		$\sum(F_K)$	$\sum(L_K)$	$\sum(B_K)$	$\sum(P_K)$	$\sum(WP_K)$

Tabelle 3.5: Übersicht zur Berechnung der Werte

4 Anwendung

In diesem Kapitel wird ein praktisches Beispiel durchgeführt, bei dem die Game-Engines Unity 2022 und Unreal Engine 5 anhand der in Abschnitt 3.3 vorgestellten Strategie verglichen werden. Das Ziel besteht darin, herauszufinden, welche Engine besser für die Umsetzung eines spezifischen Projektes geeignet ist. Dieses Beispiel dient dazu, die Evaluierungsstrategie für Game-Engines in einer realen Entwicklungssituation anzuwenden. Zu Beginn dieses Kapitels werden die Anforderungen und Rahmenbedingungen für das Projekt definiert, und die Kriterien werden entsprechend ihrer Bedeutung gewichtet. Daraufhin erfolgt eine kurze Vorstellung der beiden Game-Engines. Für beide Engines werden anschließend alle Kriterien, die für das Projekt relevant sind, anhand der Dokumentationen, so wie durch prototypische Tests in den jeweiligen Engines, analysiert. Abschließend erfolgt die Auswertung der Ergebnisse anhand der Vergleichsmatrix aus Abschnitt 3.3.

4.1 Anforderungen und Rahmenbedingungen an das Projekt

Das zu entwickelnde Projekt soll folgende Anforderungen erfüllen:

- Genre: Ein magisches Single-Player-3D-Third-Person Action-Adventure mit Schwerpunkt auf Erkundung und Kampf.
- Spielwelt: Eine große offene Welt mit detaillierter Gestaltung, die von Spielern frei erkundet werden kann.
- Plattform: Eine Desktopanwendung für Windows-Systeme.
- Spielmodus: Ein Offline-Singleplayer ohne Netzwerkfunktionalitäten.
- Grafik und Ästhetik: Der Grafikstil vereint Realismus und Stilisierung, mit lebendigen Farben, atmosphärischer Beleuchtung und begleitender Hintergrundmusik sowie Soundeffekten.
- Visuelle Effekte und Partikel: Verwendung von visuellen Effekten und Partikeln zur Darstellung magischer Kräfte.

- Benutzeroberfläche: Benötigt wird ein Hauptmenü und ein Pausemenü. Das HUD zeigt Lebenspunkte, Erfahrungspunkte und das aktuelle Level des Charakters an.
- Assets: Assets werden vorrangig in externen Programmen erstellt und in die Engine importiert. Ressourcen können auch von Drittanbietern/Marketplaces bezogen werden.
- Spielmechaniken: Der Spieler kontrolliert einen Charakter, der laufen, springen, angreifen und mit Gegenständen interagieren kann. Magische Fähigkeiten werden durch das Sammeln von Erfahrungspunkten verstärkt.
- Interaktion mit Gegnern: Gegner können in verschiedenen Zuständen wie Idle, Run und Attack auftreten und auf den Spieler reagieren. Die Kämpfe sollen dynamisch und abwechslungsreich sein.
- Kosten: Die Entwicklungskosten sollen möglichst gering gehalten werden.
- Versionskontrolle: Git(Hub) wird als das bevorzugte Versionskontrollsystem verwendet.
- Entwicklungsplattform: Die Entwicklung erfolgt auf einem Windows-System.

Abbildung 4.1: Konzeptillustration des Spiels

Kriterien, die für die Wahl der Engine keine Relevanz haben: Kompatibilität, Profiling und Debuggen-Tools, Quellcode Zugriff, verwendete SDKs, der Animations-Editor, Cinematic-Tools, Sound-Editor und Netzwerkfunktionalitäten.

4.2 Zuvergleichende Engines

Unity und Unreal Engine sind etablierte Game-Engines, die häufig in der Entwicklung eingesetzt werden [Ste23c]. Beide bieten eine umfangreiche Palette an Funktionen, weshalb sie sich gut für die Umsetzung des Spiels eignen.

Die Unity Engine wurde im Jahr 2005 von Unity Technologies veröffentlicht. In diesem Vergleich wird die Version Unity 2022 LTS verwendet, da es sich um die aktuellste Version mit langfristiger Unterstützung handelt. Unity ist darauf spezialisiert, sowohl 2D- als auch 3D-Anwendungen für verschiedene Plattformen zu entwickeln [UnioD]. Sie wird häufig für die Entwicklung von Mobile-Applikationen eingesetzt [App22]. Die Engine soll besonders nutzerfreundlich sein. Beispiele für Spiele, die mit Unity entwickelt wurden: Ori and the Blind Forest, The Forest, RimWorld und Among Us [Ste23a].

Die Unreal Engine wurde 1998 von Epic Games veröffentlicht, und die neueste Version, Unreal Engine 5, wurde im April 2022 herausgebracht. Für den Vergleich wird Version 5.3 verwendet. Unreal Engine ist bekannt für hochwertige Spiele, insbesondere für die Grafikqualität. Beispiele für Spiele, die mit der Unreal Engine entwickelt wurden, sind Fortnite und Hogwarts Legacy, aber auch Indie-Titel wie Stray und Kena: Bridge of Spirits [Ste23b].

4.3 Bewertung von Unity 2022

In diesem Abschnitt erfolgt die Anwendung der Kriterien aus Kapitel 3.2 auf Unity 2022, um dessen Eignung für das geplante Spiel zu bewerten.

4.3.1 Zugänglichkeit

Plattformkompatibilität und Installation
Die Unity Engine kann auf verschiedenen Betriebssystemen zur Entwicklung eingesetzt werden, darunter Windows, macOS und Linux (Ubuntu und CentOS). Die Installation der Engine erfolgt über den offiziellen Unity Hub, der von der Unity-Website heruntergeladen und installiert werden kann. Nach der Installation des Unity Hubs muss im Hub die gewünschte Unity-Version heruntergeladen werden (in diesem Fall 2022.3.9f1). Zusätzlich können Module wie Android Build Support, iOS Build Support, WebGL-Unterstützung oder Visual Studio als Entwicklungsumgebung hinzugefügt werden. Zum Schluss muss eine Lizenz hinzugefügt werden, was durch die Erstellung eines Unity-Accounts und die Auswahl der entsprechenden Lizenzoption erfolgt. Sobald der Unity Hub eingerichtet ist, kann ein neues Projekt erstellt werden. Beim Anlegen eines Projekts kann eine Vorlage

ausgewählt werden, die bestimmte Konfigurationen bereitstellt, wie zum Beispiel die Ansicht oder die Renderpipeline. Dies hilft zusätzlich bei der Optimierung des Builds für die Zielplattform.
Die für das Projekt benötigte Plattform (Windows 11) wird unterstützt, und der Installationsprozess sowie die Einrichtung sind einfach durchzuführen.

Erlernbarkeit
Unity stellt auf der Plattform "Unity Learn" eine Vielzahl von Ressourcen bereit, darunter Tutorials, Kurse, Beispielprojekte und Dokumentationen. Ein Großteil dieser Inhalte ist kostenfrei verfügbar. Für jede Version der Engine gibt es eine gut strukturierte Dokumentation. Neben der offiziellen Dokumentation stehen auch Foren und FAQs zur Verfügung, die zusätzliche Hilfestellung bieten. Darüber hinaus sind auf externen Plattformen wie YouTube und Udemy zahlreiche Tutorials und Kurse verfügbar.
Die Bereitstellung dieser Ressourcen erleichtert den Einstieg und den Entwicklungsprozess mit der Engine.

Lizenzmodell und Kosten
Unity bietet verschiedene Lizenzen an, die vom Brutto-Umsatz/Finanzmitteln abhängig sind. Außerdem werden für jede höhere Stufe zusätzliche Funktionalitäten zur Verfügung gestellt. Alle haben Zugriff auf die Unity Engine, Unity Visual Scripting und die Unity-Versionskontrolle so wie auf Monetarisierungsmöglichkeiten.

- Personal: Kostenloser Zugriff auf Unity, solange der jährliche Brutto-Umsatz unter 100.000 US-Dollar liegt.
- Unity Plus: 369 € pro Person, solange der jährliche Brutto-Umsatz unter 200.000 US-Dollar liegt.
- Unity Pro: 1.877 € pro Person, ab einem jährlichen Bruttoumsatz über 200.000 US-Dollar. Unity Pro bietet erweiterte Funktionen, darunter die Möglichkeit, auf Spielekonsolen zu veröffentlichen und Zugriff auf Havok Physics (ein Physik-SDK).
- Unity Enterprise: Individuelle Preise, ab einem jährlichen Bruttoumsatz über 200.000 US-Dollar. Unity Enterprise bietet spezielle Vorteile wie erweiterten LTS-Support und die Einsicht in den Quellcode.
- Unity Industry: 4.554 € pro Person, ermöglicht zusätzlichen Zugriff auf branchenspezifische Toolkits.

Für dieses Projekt ist die Personal-Lizenz passend.

4.3.2 Entwicklungsumgebung und Workflow

Unterstütze Genre
Unity unterstützt alle gängigen Genres, darunter 2D-Spiele, 3D-Spiele, sowie Virtual Reality (VR) und Augmented Reality (AR).
Für dieses Projekt werden nur 3D-Funktionalitäten benötigt.

Programmiersprache und Skripting
Unity verwendet hauptsächlich C# als Programmiersprache für die Spieleentwicklung. Alternativ dazu kann Visual Scripting eingesetzt werden.

Plattformunterstützung
Unity unterstützt die meisten der führenden Desktop-, Web- und Mobilplattformen. Darunter Windows, macOS, Linux, iOS, visionOS, Android, WebGL, Dedicated Server, tvOS. Es können Spiele für Konsolen (Nintendo Switch, PlayStation, Xbox) erstellt werden. Dafür wird eine Zustimmung des jeweiligen Plattformbetreibers benötigt, so wie die Unity Pro Lizenz. Unity bietet Multiplatform-Unterstützung an.
Für dieses Projekt wird nur die Unterstützung für Windows-Plattformen benötigt.

Versionskontrolle und kollaborative Arbeit
Unity bietet "Plastic SCM" für die Zusammenarbeit in Projekten an (für 3 Nutzer und 5 GB Speicherplatz). Alternativ dazu kann Git verwendet werden.

Oberfläche und Szenen Editor
Die Standardoberfläche von Unity teilt sich in sechs Hauptbereiche. In der "Scene View" kann die aktuelle Szene bearbeitet werden. Die "Game View" simuliert das Spiel. Durch das Anklicken des Playbuttons kann das Spiel gestartet werden. Im Projektfenster werden alle Ressourcen des Projektes angezeigt. Das Hierarchiefenster zeigt die Struktur der Szene und alle GameObjects, die sich darin befinden und wie sie miteinander verknüpft sind. Im Inspektorenfenster werden die Eigenschaften eines GameObjects angezeigt und können dort auch bearbeitet werden. Über das Menü können Projekteinstellungen vorgenommen und weitere Fenster sowie Funktionen aufgerufen werden.
Um die Szene zu gestalten, können die benötigten Assets per Drag-and-drop in die Szene gezogen und dort platziert werden. Selbiges gilt für Materialien und andere Ressourcen. Per Rechtsklick im Hierarchiefenster kann ein neues GameObject erstellt und der Szene hinzugefügt werden, zum Beispiel Licht, Audio und 3D-Objekte.
Die Benutzeroberfläche von Unity ist übersichtlich strukturiert und die Ansicht ist individuell anpassbar, einige benötigte Funktionen/Editoren lassen sich nicht finden, da sie erst über den Package Manager hinzugefügt werden müssen.

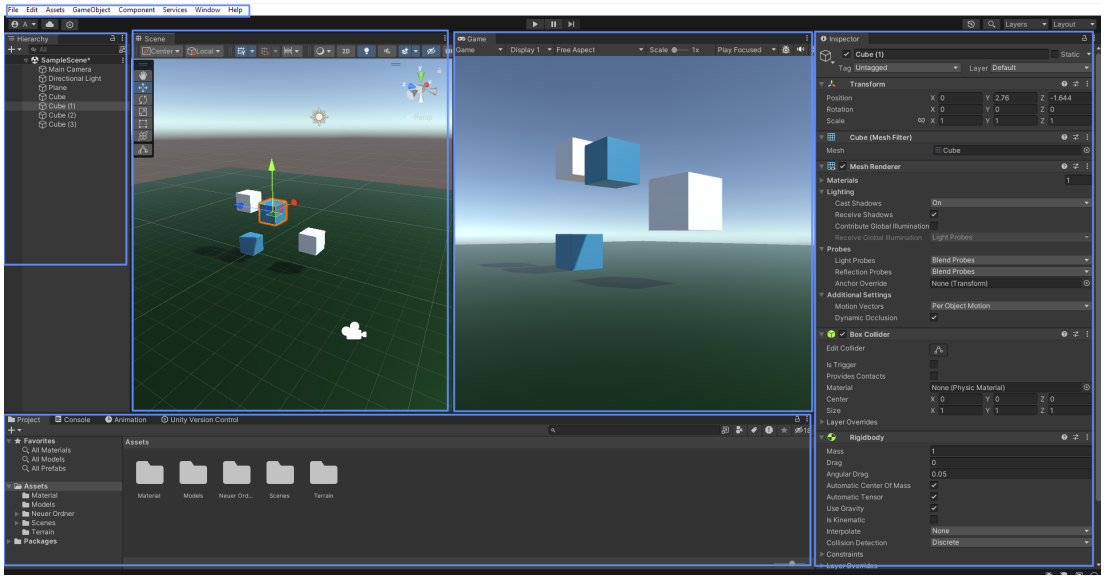

Abbildung 4.2: Unity 2022 Oberfläche und Szenen Editor

4.3.3 Ressourcen

Marketplace

Unity verfügt über einen Asset-Store mit einer Vielzahl an unterschiedlichen Assets, Plugins, Tools und anderen Ressourcen. Die Assets können über die offizielle Webseite erworben und im Projekt unter dem Menüpunkt ”Package Manager” in das Projekt integriert werden.
Trotz der großen Auswahl an Assets, ist die Anzahl der geeigneten Assets für das Projekt sehr begrenzt.

Asset-Management und Workflow

Assets, die außerhalb von Unity erstellt wurden oder von Drittanbietern bezogen werden, können in Unity importiert oder per Drag-and-Drop eingefügt werden. Unity unterstützt die meisten grundlegenden Asset-Typen, wie 3D-Modelle, Bilddateien und Audiodateien. Assets können exportiert und für andere Projekte wiederverwendet werden.
Die Drag-and-Drop-Funktionalität sowie die visuelle Unterscheidung zwischen den Dateitypen vereinfacht die Handhabung im Projekt.

4.3.4 Grafik

Licht und Schatten

Unity bietet verschiedene Möglichkeiten zur Gestaltung der Beleuchtung. Dabei werden direkte und indirekte Beleuchtung sowie dynamische und statische Beleuchtung unterstützt. Unity verfügt über Systeme für die globale Beleuchtung, die direkte und indirekte Beleuchtung kombinieren, um realistische Beleuchtungseffekte zu erzeugen.

Diese Systeme sind als ”Baked Global Illumination” und ”Realtime Global Illumination” bekannt. Bei ”Realtime Global Illumination” wird die Beleuchtung zur Laufzeit während des Spiels berechnet, was dynamische Änderungen der Beleuchtung ermöglicht. Bei ”Baked Global Illumination” werden vorab berechnete Beleuchtungsinformationen durch einen Backing-Prozess generiert und zur Laufzeit geladen.
Unity bietet eine Vielzahl vordefinierter Lichtquellen, die angepasst und in die Szene integriert werden können. Darüber hinaus können individuelle Materialien für die Beleuchtung erstellt werden.
Die Gestaltung einer realistischen Beleuchtung ist möglich, erfordert jedoch ein tiefes Verständnis für die verschiedenen Einstellungen, um die gewünschten visuellen Ergebnisse für das Projekt zu erzielen.

Materialien und Objekte
Unity bietet grundlegende 3D-Modellierungsfunktionen, die sich für die Erstellung einfacher geometrischer Formen, für Prototypen und einfachen Objekten eignen. Für die Erstellung von Texturen und komplexen Modellen muss eine externe Software wie Blender verwendet werden. Der Shader-Graph kann zur Erstellung von komplexen Materialien und Effekten genutzt werden, wobei Physically Based Rendering unterstützt wird.
Die gewünschten Ziele lassen sich mithilfe dieser Funktionen erreichen.

Visuale Effekte und Partikel
Mit Unity können verschiedene Arten von Effekten erstellt werden. Dazu gehören Post-Processing-Effekte, Full-Screen-Effekte, die Verwendung von Partikel-Systemen und die Erstellung von benutzerdefinierten Shadern. Um Post-Processing nutzen zu können, muss über den Package-Manager das entsprechende Paket hinzugefügt werden.
Die Umsetzung der benötigten Effekte im Projekt ist grundsätzlich möglich, erfordert jedoch ein tiefes Verständnis der vielfältigen Einstellungsmöglichkeiten, um spezifische Effekte zu erzielen.

UI-Editor
Benutzeroberflächen können mithilfe des ”UI Toolkit”, erstellt werden. Über das Menü kann der Editor geöffnet werden, um die Benutzeroberfläche zu gestalten. Diese wird anschließend als UXML-Datei gespeichert. Zusätzlich dazu können Style-Attribute/Klassen für die Elemente definiert werden, die als USS-Dateien abgelegt werden. Das System orientiert sich an HTML/XML und CSS. Um die UI in der Szene verwenden zu können, muss in der Szenen-Hierarchie ein GameObject mit einer UI-Komponente erstellt werden, der die erstellte UI zugewiesen wird. Durch das Hinzufügen einer Skriptkomponente zu diesem GameObject kann das Verhalten der UI und ihrer Elemente in einem Script definiert werden.

Culling-Optimierung
Unity verwendet Frustum-Culling und Occlusion Culling, um Culling-Optimierung durchzuführen. Um diese Culling-Techniken in Unity zu nutzen, muss die Szene einem Backing-Prozess unterzogen werden, der dazu dient, Culling-Daten zu generieren. Der Backing-Prozess teilt die Szene in verschiedene Bereiche auf und erzeugt Daten, die beschreiben, welche Teile der Szene von der Kamera sichtbar sind und welche nicht. Zu beachten ist, dass Occlusion Culling nicht effektiv für Objekte funktioniert, die zur Laufzeit generiert werden, da diese im Baking-Prozess nicht berücksichtigt werden können. Für solche dynamisch generierten Objekte müssen manuell alternative Sichtbarkeitsprüfungsprozesse implementiert werden.
Einem GameObject kann eine LOD Group-Komponente hinzugefügt werden. Darin werden LOD-Level in Abhängigkeit zur Distanz der Kamera eingestellt. Jede Stufe bekommt ein entsprechendes Modell zugeordnet, welches in Abhängigkeit von der Entfernung zur Kamera gerendert wird. Die Modelle für die einzelnen Detaillevel müssen manuell erstellt und importiert werden.

4.3.5 Physik und Kollision

Umgang mit Physik und Kollision
Einem Objekt kann eine Physikkomponente hinzugefügt werden, um es für physikalische Simulationen bereitzustellen. Ebenso kann eine Collider-Komponente hinzugefügt werden, um Kollisionen zwischen Objekten zu erfassen. Durch das Hinzufügen einer Skriptkomponente kann auf diese Informationen zugegriffen werden, um das Verhalten des Objekts zu beeinflussen.

4.3.6 Animation

Umgang mit Animationen
Es ist möglich, Animationsclips zu importieren. Mithilfe des Animators können Abläufe und Übergänge erstellt werden, die an Bedingungen geknüpft werden. Der Animator verhält sich dabei wie ein Zustandsautomat. Zusätzlich dazu können Animationen mithilfe von Blend Trees gesteuert werden, um komplexe Abläufe zu erstellen.

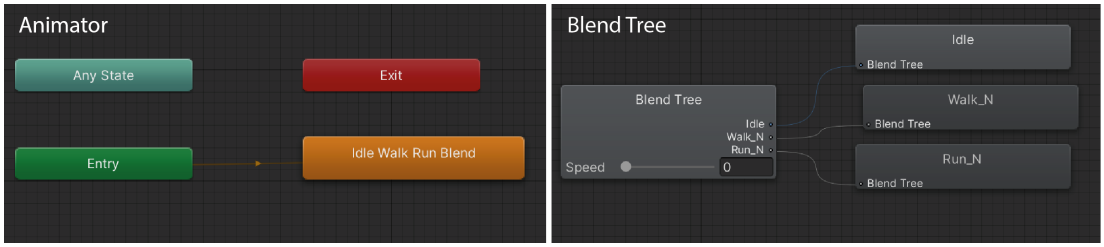

Abbildung 4.3: Unity 2022 Animator und Blend Tree

4.3.7 Sound

Umgang mit Sound

Es können Sounddateien importiert werden, um sie als Hintergrundmusik, Soundeffekte oder Sprachausgaben in das Spiel zu integrieren. Um diese Sounds im Spiel zu verwenden, wird einem GameObject in der Szene eine Audiokomponente hinzugefügt, der wiederum die Audioressource zugewiesen wird. Die Audioquellen können mithilfe von Skripten gesteuert werden. Es werden sowohl 2D- als auch 3D-Soundeffekte unterstützt.

4.3.8 World building

Unity bietet einen Terrain-Editor, der es ermöglicht, Landschaften zu erstellen. Hierfür muss ein Terrain erstellt werden, das mithilfe verschiedener Werkzeuge geformt und mit Texturen bemalt werden kann. Zusätzlich dazu können Bäume und Gras hinzugefügt werden. Durch das Hinzufügen des ”Terrain Tools” über den Package Manager stehen zusätzliche Werkzeuge zur Landschaftsgestaltung zur Verfügung.
Die Handhabung der Werkzeuge gestaltet sich unkompliziert.

4.3.9 Künstliche Intelligenz

Um das Verhalten der NPCs zu steuern und eine Illusion von Intelligenz zu erschaffen, kann das ”AI Navigation package” in Unity verwendet werden. Dieses Paket muss über den Package Manager hinzugefügt werden. Es ermöglicht unter anderem die Erstellung von NavMeshes für verschiedene Charaktere, um Bereiche zu definieren, in denen sie sich bewegen dürfen. Es können Key Patrol Points gesetzt werden, um den Agenten Pfade vorzugeben, die sie entlanglaufen sollen. Das Verhalten der Agenten kann durch den zuvor genannten Animator gesteuert werden, der als Zustandsautomat fungiert. Darüber hinaus werden Skripte und Kollisionserkennung verwendet, um das Verhalten der Agenten zu steuern.

4.4 Bewertung von Unreal Engine 5

In diesem Abschnitt erfolgt die Anwendung der Kriterien aus Kapitel 3.2 auf Unreal Engine 5, um dessen Eignung für das geplante Spiel zu bewerten.

4.4.1 Zugänglichkeit

Plattformkompatibilität und Installation
Unreal Engine 5 kann auf verschiedenen Betriebssystemen zur Entwicklung eingesetzt werden, darunter Windows, macOS und Linux. Um die Engine verwenden zu können, muss der Epic Games Launcher von der offiziellen Webseite heruntergeladen und installiert werden. Anschließend ist die Erstellung eines Epic Games-Kontos erforderlich. Im Epic Games Launcher können verschiedene Versionen der Engine heruntergeladen und verwaltet werden, ebenso wie alle Projekte und erworbene Assets. Für dieses Projekt wird die Version 5.3 benötigt. Vor der Installation können Pakete ausgewählt werden, die zusätzlich installiert werden sollen, um Unterstützung für Plattformen zu erhalten. Nach der Installation kann die Engine gestartet und ein neues Projekt kann angelegt werden. Bei der Erstellung eines neuen Projekts kann ein Template ausgewählt werden, welches die Spielsicht vordefiniert. Zusätzlich dazu kann festgelegt werden, ob das Blueprint-System oder C++ verwendet werden soll, ob es sich um eine mobile oder Desktop-Anwendung handelt, die Qualitätseinstellungen (Maximum oder Scalable Quality), ob Raytracing verwendet werden soll und ob Starter Content hinzugefügt werden soll.
Die benötigte Plattform (Windows 11) wird unterstützt, und der Installationsprozess sowie die Einrichtung sind unkompliziert durchzuführen.

Erlernbarkeit
Epic Games stellt umfangreiche Lernressourcen zur Verfügung, darunter Tutorials, Dokumentationen, Beispielprojekte und Foren. Diese Ressourcen sind kostenfrei und bieten sowohl Anfängern als auch erfahrenen Entwicklern die Möglichkeit, sich mit der Engine vertraut zu machen und ihre Fähigkeiten zu erweitern. Für jede Version der Engine gibt es eine ausführliche Dokumentation. Darüber hinaus existiert eine aktive Community von Entwicklern, die Foren und Supportmöglichkeiten bereitstellt, um zusätzliche Unterstützung zu bieten. Externe Plattformen wie YouTube und Udemy bieten ebenfalls zahlreiche Lernressourcen und Schulungskurse an.

Lizenzmodell und Kosten
Alle Funktionen der Unreal Engine sind für die Entwicklung von Videospielen kostenfrei, bis die Bruttoeinnahmen des Titels 1 Million USD übersteigen. Danach muss eine Lizenz-

gebühr von 5% gezahlt werden. Für kommerzielle Spiele muss vor der Veröffentlichung ein Veröffentlichungsformular ausgefüllt werden.

4.4.2 Entwicklungsumgebung und Workflow

Unterstütze Genre
Unreal Engine 5 unterstützt alle gängigen Genres, darunter 2D-Spiele, 3D-Spiele, sowie Virtual Reality (VR) und Augmented Reality (AR).
Für dieses Projekt werden nur 3D-Funktionalitäten benötigt.

Programmiersprache und Skripting
Unreal Engine verwendet C++ als Programmiersprache, zusätzlich dazu wird mit Blueprints ein Visual Scripting System angeboten.

Plattformunterstützung
Unreal Engine 5 bietet Unterstützung für eine Vielzahl an Desktop-, Konsolen- und Mobilplattformen, darunter Windows, macOS, Linux, iOS- und Android-Mobilgeräte und Konsolen wie Nintendo Switch, PlayStation, Xbox und das Steam Deck. Um für Konsolen entwickeln zu können, wird die Zustimmung des jeweiligen Plattformbetreibers benötigt, um Zugriff auf die notwendigen Ressourcen zu erhalten. Die Engine bietet Multiplattform-Entwicklungsfunktionen, um Spiele auf verschiedenen Plattformen zu veröffentlichen.
Für dieses Projekt wird nur die Unterstützung von Windows-Plattformen benötigt.

Versionskontrolle und kollaborative Arbeit
Unreal Engine verfügt über zwei integrierte Systeme zur kollaborativen Arbeit, Perforce und Subversion. Es ist möglich, alternativ Git zu verwenden.

Oberfläche und Szenen Editor
Die Standardoberfläche von Unreal teilt sich in fünf Hauptbereiche. Im ”Level Viewport” kann die aktuelle Szene/Level bearbeitet werden. Die Toolbar enthält Shortcuts für die gängigsten Tools der Engine sowie Shortcuts zum Aufrufen des Play-Modus und zum Bereitstellen des Projekts auf anderen Plattformen. Das Outliner-Fenster gibt einen Überblick über die Struktur der Szene und zeigt an, wie die verschiedenen Elemente (Actors) miteinander verbunden sind. Das Details-Panel bietet detaillierte Informationen über ausgewählte Objekte und erlaubt deren Bearbeitung. Im Content Drawer werden alle Ressourcen des Projekts angezeigt. Über die Menü-Bar lassen sich Projekteinstellungen vornehmen und zusätzliche Fenster oder Funktionen aufrufen.
Um die Szene zu gestalten, können die benötigten Assets per Drag-and-Drop in die Szene gezogen und dort platziert werden. Dasselbe gilt für Materialien und andere Ressourcen. Über die Toolbar können neue Actors erstellt und der Szene hinzugefügt

werden, zum Beispiel Licht, Audio und 3D-Objekte.
Die Benutzeroberfläche von Unreal ist übersichtlich strukturiert, und die Ansicht ist individuell anpassbar. Benötigte Funktionen/Editoren lassen sich einfach finden.

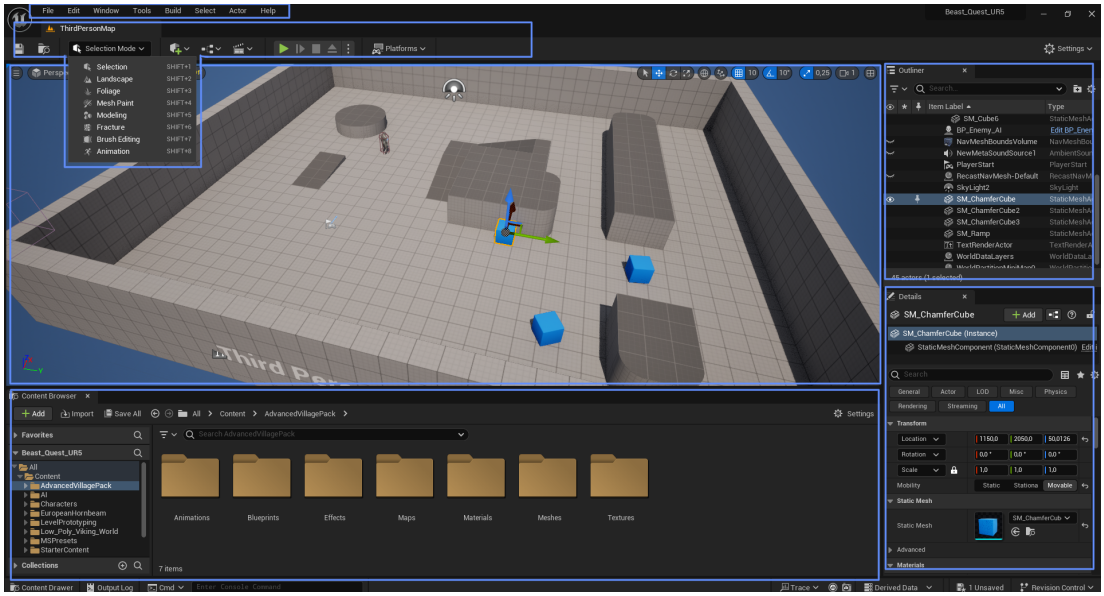

Abbildung 4.4: Unreal Engine 5 Oberfläche und Szenen Editor

4.4.3 Ressourcen

Marketplace
Unreal verfügt über einen Asset-Store mit einer Vielzahl an unterschiedlichen Assets, Plugins, Tools und anderen Ressourcen. Die Assets können über den Marketplace im Epic Games Launcher erworben werden. In der Bibliothek des Launchers finden sich alle Assets, die in dem offiziellen Marketplace erworben wurden, und können dort einem Projekt hinzugefügt werden.
Trotz der großen Auswahl an Assets ist die Anzahl der geeigneten Assets für das Projekt sehr begrenzt.

Asset-Management und Workflow
Assets, die außerhalb von Unreal erstellt wurden oder von Drittanbietern bezogen werden, können importiert oder per Drag-and-Drop eingefügt werden. Unreal unterstützt die meisten grundlegenden Asset-Typen.
Die Drag-and-drop-Funktionalität sowie die visuelle Unterscheidung zwischen den Dateitypen durch Bilder und Farben vereinfacht die Handhabung im Projekt.

4.4.4 Grafik

Licht und Schatten
Mit dem Lumen Global Illumination System von Unreal 5 lassen sich realistische Lichteffekte erstellen, so wie es für das Projekt vorgesehen ist. Lumen ermöglicht die Erzeugung von direktem und indirektem Licht, ohne dass Lightmaps erstellt werden müssen. Alternativ besteht die Möglichkeit, auf Baked Global Illumination zurückzugreifen, indem Lightmaps gebaked werden. Unreal bietet eine Vielzahl vordefinierter Lichtquellen die angepasst und in die Szene integriert werden können. Darüber hinaus können individuelle Lichtquellen erstellt werden. Die Gestaltung erfordert ein tiefes Verständnis für die verschiedenen Einstellungen, um die gewünschten Ergebnisse für das Projekt zu erzielen.

Materialien und Objekte
Unreal Engine bietet 3D-Modellierungsfunktionen. Im Modellierungsmodus können komplexe Objekte ähnlich wie in 3D-Modellierungssoftware erstellt und gestaltet werden. Der Material-Graph ermöglicht die Erstellung von Materialien mithilfe eines Node-Systems und unterstützt Physically Based Rendering. Zusätzlich dazu können mithilfe von "Fracture" zerstörbare Objekte erstellt werden, die beispielsweise bei Kollisionen in Einzelteile zerbrechen.
Die Erstellung von komplexen Materialien und Objekten erfordert Erfahrung, jedoch ist das Erstellen von Objekten für Prototypen unkompliziert.

Visuale Effekte und Partikel
Unreal Engine ermöglicht die Erstellung verschiedener Effekte, darunter Post-Processing-Effekte, Full-Screen-Effekte und die Verwendung von Partikelsystemen. Einfache Post-Processing-Effekte können mit wenig Aufwand durch das Hinzufügen von Volume Actors erstellt werden. Das Niagara VFX-System erlaubt die Erzeugung verschiedener Arten von Effekten und stellt Templates bereit.
Die Umsetzung der benötigten Effekte im Projekt ist möglich, erfordert jedoch ein tiefes Verständnis der vielfältigen Einstellungsmöglichkeiten, um spezifische Effekte zu erzielen.

UI-Editor
Benutzeroberflächen können in Unreal Engine 5 mithilfe von Widget Blueprints erstellt werden. Der UI-Editor ermöglicht die Gestaltung von Widgets, indem vordefinierte Elemente wie Buttons, Slider und viele weitere platziert, angepasst und animiert werden. Das Verhalten der Benutzeroberfläche wird über den Graphen des Blueprints gesteuert.

Culling-Optimierung
Unreal verwendet eine Kombination aus verschiedenen Culling-Methoden, darunter Frustum-Culling, Occlusion-Culling und Distanz-Culling. Dynamische Okklusionsprozesse arbeiten automatisch im Hintergrund. Es können Vorabberechnungen durchgeführt

werden (Precomputed Visibility), indem Visibility Volumes (Bereiche) dem Level hinzugefügt werden, um so Sichtbarkeitsinformationen für statische Objekte innerhalb des definierten Bereichs zu generieren. Darüber hinaus ist es möglich, durch Distanz-Culling-Einstellungen zu definieren, ab welcher Entfernung zur Kamera bestimmte Bereiche oder Actors sichtbar werden.
Modelle besitzen LOD-Level, denen zuvor erstellte Modelle zugeordnet werden können, um verschiedene Detailstufen in abhängig von der Entfernung zur Kamera zu rendern. Eine Besonderheit von Unreal Engine 5 ist die Integration von Nanite. Nanite ermöglicht die Darstellung extrem detaillierter und hochauflösender Modelle ohne den Einsatz von manuell erstellten LODs. Hierzu werden Meshes in hierarchische Cluster zerlegt. Während des Renderings werden die Cluster dynamisch auf verschiedenen Detailstufen ausgetauscht (Cluster Culling). Ein Static Mesh kann in ein Nanite-Mesh konvertiert werden.

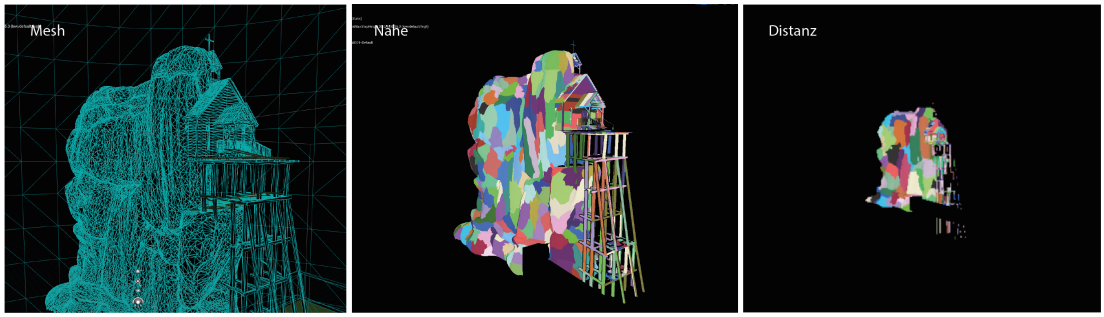

Abbildung 4.5: Unreal Engine 5 - Nanite Cluster

4.4.5 Physik und Kollision

Umgang mit Physik und Kollision
Einfache physikalische Eigenschaften eines Actors können direkt im Detail-Fenster angepasst werden. Ebenso kann eine Collider-Komponente hinzugefügt werden, um Kollisionen zwischen Objekten zu erfassen. Durch das Hinzufügen eines Skripts kann auf diese Informationen zugegriffen werden, um das Verhalten des Objekts zu beeinflussen. Chaos Destruction ermöglicht die realistische Zerstörung von Objekten.

4.4.6 Animation

Umgang mit Animationen
In einem Animation-Blueprint werden Zustände, Übergänge und Logik der Animationen definiert. Im Animationsgraph können Animationen miteinander verbunden werden. Mithilfe von BlendSpaces können Übergänge zwischen Animationen erstellt werden, die abhängig von Variablen sind (zum Beispiel die Laufgeschwindigkeit). Im Event-

Graph kann dem Blueprint Logik hinzugefügt werden. Animation-Blueprints können als Zustandsautomaten betrachtet werden.

4.4.7 Sound

Umgang mit Sound
Sounddateien können importiert und per Drag-and-Drop direkt in die Szene eingefügt werden. Sounds können außerdem durch Blueprints integriert werden, ohne sie in der Szene zu platzieren. Durch die Verwendung des MetaSound-Systems können komplexe Verhalten, so wie Klänge und Audioeffekte für die Klangquelle definiert werden.
Mit zunehmender Erfahrung können komplexe Effekte und Sound-Systeme erstellt werden.

4.4.8 World building

Über den Landscape-Modus, der über die Toolbar erreichbar ist, lassen sich Landschaften erstellen, indem das Terrain mit verschiedenen Werkzeugen geformt und gestaltet wird. Mithilfe des Foliage-Tools können dem Terrain Bäume und Pflanzen hinzugefügt werden. Die Engine bietet zudem ein Wasser-System zur Erstellung von Meeren und Flüssen. Unreal Engine 5 integriert das World Partition-System, das eine effiziente Verwaltung großer und offener Welten ermöglicht. Dabei wird die Welt in Zellen unterteilt, die abhängig von der Spielerposition dynamisch geladen oder entladen werden. Der Landescape-Modus kann in Kombination mit dem Nanite-System verwendet werden.
Die Bedienung dieser Werkzeuge ist benutzerfreundlich und bietet umfangreiche Funktionen zur Erstellung großer, offener Welten, wie es für das Projekt geplant ist.

4.4.9 Künstliche Intelligenz

Um das Verhalten der Agenten zu steuern, können verschiedene Techniken kombiniert werden. Es können NavMeshes erstellt werden, die sich automatisch der Umgebung anpassen, um Bereiche zu definieren, in denen sie sich bewegen dürfen. Agenten können mithilfe von Blueprints gesteuert werden. Animation-Blueprints fungieren als Zustandsautomaten. Zusätzlich dazu bietet Unreal Engine ein System zur Erstellung von Behavior Trees.

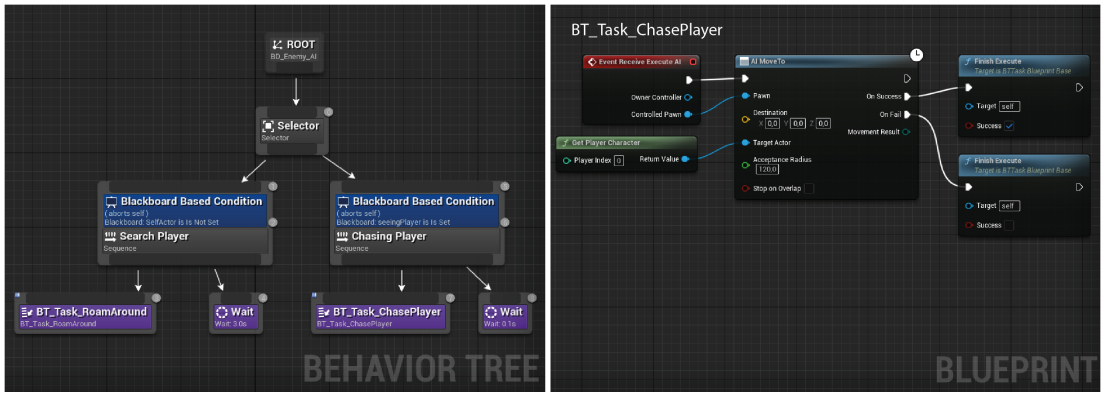

Abbildung 4.6: Unreal Engine 5 - Behavior Tree

4.5 Auswertung

Unreal Engine 5 führt in der Gesamtwertung mit 106 Punkten, während Unity 2022 insgesamt 95 Punkte erreicht. Dabei führt Unreal Engine in Hinblick auf die Funktionen mit 30 zu 22 Punkten. Unity führt hingegen mit 25 zu 23 Punkten, was die Benutzerfreundlichkeit angeht. Beide Engines erhalten für die Kriterien ”Leistung” 22 Punkte.

Sowohl Unity 2022 als auch Unreal Engine 5, bieten die erforderlichen Funktionen, um das Projekt umzusetzen. Die Vergleichsmatrix verdeutlicht jedoch, dass Unreal Engine 5 als die potentere Option hervorgeht. Neben den grundlegenden Funktionen bietet Unreal Engine 5 zusätzliche Features, die die Umsetzung des Projektes erleichtern können und das Endresultat näher an das definierte Ziel heranbringen. Besonders zeichnet es sich in den Bereichen Grafik, World-Building und künstliche Intelligenz aus.
In Hinblick auf die Benutzerfreundlichkeit führt Unity aufgrund der größeren Auswahl an Lernmaterial und einer übersichtlicheren Oberfläche.
Während des Tests traten keine Leistungsprobleme auf. Es ist jedoch zu beachten, dass es sich lediglich um prototypische Tests handelt, und daher keine Aussagen darüber getroffen werden können, wie sich die Engines mit steigender Komplexität verhalten würden.

Kriterium	Gewichtung	Unity 2022					Unreal Engine 5				
		F	L	B	P	WP	F	L	B	P	WP
Zugänglichkeit											
Plattformkompatibilität und Installation	1	1	1	2	4	4	1	1	2	4	4
Erlernbarkeit	2	2	1	2	5	10	2	1	1	4	8
Lizenzmodell und Kosten	2	1	1	1	3	6	1	1	1	3	6
Entwicklungsumgebung und Workflow											
Unterstütze Genre	1	1	1	1	3	3	1	1	1	3	3
Programmiersprache und Skripting	1	1	1	1	3	3	1	1	1	3	3
Plattformunterstützung	1	1	1	1	3	3	1	1	1	3	3
Kompatibilität	0	-	-	-	-	-	-	-	-	-	-
Versionskontrolle und kollaborative Arbeit	2	1	1	1	3	6	1	1	1	3	6
Profiling- und Debugging-Tools	0	-	-	-	-	-	-	-	-	-	-
Oberfläche und Szenen-Editor	1	1	1	2	4	4	1	1	1	3	3
Quellcode Zugriff	0	-	-	-	-	-	-	-	-	-	-
Ressourcen											
Marketplace	1	1	1	2	4	4	1	1	2	4	4
Asset-Management und Workflow	1	2	1	2	5	5	2	1	2	5	5
Grafik											
Grafik-SDK	0	-	-	-	-	-	-	-	-	-	-
Licht und Schatten	2	1	1	0	2	4	2	1	0	3	6
Materialien und Objekte	2	1	1	1	3	6	2	1	1	4	8
Visuale Effekte und Partikel	2	1	1	0	2	4	2	1	0	3	6
UI-Editor	1	1	1	2	4	4	1	1	2	4	4
Culling-Optimierung	2	1	1	1	3	6	2	1	1	4	8
Physik und Kollision											
Physik-SDK	0	-	-	-	-	-	-	-	-	-	-
Umgang mit Physik und Kollision	1	1	1	1	3	3	2	1	1	4	4
Animation											
Animations-Editor	0	-	-	-	-	-	-	-	-	-	-
Umgang mit Animation	1	1	1	1	3	3	1	1	1	3	3
Cinematic-Tools	0	-	-	-	-	-	-	-	-	-	-
Sound											
Sound-Editor	0	-	-	-	-	-	-	-	-	-	-
Umgang mit Sound	1	1	1	1	3	3	2	1	1	4	4
World building											
Erstellung von Landschaften	2	1	1	2	4	8	2	1	2	5	10
Künstliche Intelligenz											
KI-Editor	2	1	1	1	3	6	2	1	1	4	8
Netzwerk											
Netzwerkfunktionalitäten	0	-	-	-	-	-	-	-	-	-	-
Gesamtpunkte		22	20	25	67	**95**	30	20	23	73	**106**

Tabelle 4.1: Vergleichsmatrix - Bewertung von Unity 2022 und Unreal Engine 5

5 Zusammenfassung und Fazit

In diesem abschließenden Kapitel werden die Ergebnisse der Arbeit zusammengefasst und ein Fazit gezogen. Dabei werden die wichtigsten Erkenntnisse der Arbeit aufgeführt, mögliche weitere Schritte werden beleuchtet, und es wird ein Ausblick auf mögliche weitere Forschungen gegeben.

5.1 Ergebnisse

In diesem Abschnitt werden die Ergebnisse dieser Arbeit zusammengefasst. Dabei wird auf die Herangehensweise der Beantwortung der Forschungsfragen, sowie deren Ergebnisse eingegangen.

Um die relevanten Kriterien für den Vergleich von Game-Engines zu identifizieren, erfolgte eine umfassende Analyse verschiedener Quellen. Dabei wurden Kriterien aus bisherigen Untersuchungen zusammengeführt und anhand der Ausarbeitung von Game-Engine-Komponenten und Software-Qualitätskriterien erweitert. Aus der Analyse resultieren sowohl Game-Engine spezifische Kriterien, die Funktionen und Eigenschaften einer Game-Engine umfassen, als auch allgemeinere Software-Qualitätskriterien. Die resultierenden Kriterien sind in Tabelle 3.2 detailliert aufgeführt und in Abschnitt 3.2 erläutert. Besonders relevant für den Vergleich von Game-Engines sind Kriterien, die mit den Projektanforderungen zusammenhängen.

Um eine Game-Engine für ein spezifisches Projekt auswählen zu können, wurde eine Strategie anhand einer Nutzwertanalyse erarbeitet und eine Vergleichsmatrix auf Basis der identifizierten Vergleichskriterien erstellt. Zusätzlich dazu wurden Bewertungskalen definiert, um die Funktionalität, Leistung und Benutzerfreundlichkeit der jeweiligen Engine bewerten zu können. Das genaue Vorgehen wird in Abschnitt 3.3 beschreiben. Die Nutzwertanalyse wurde angewendet, um zu entscheiden, ob Unreal Engine 5 oder Unity 2022 für die Entwicklung eines Open-World-Single-Player-3D-Third-Person Action-Adventure die geeignetere Wahl ist. Die Analyse ergab, dass Unreal Engine 5 aufgrund von zusätzlichen Funktionen, die das Potenzial haben, den Entwicklungsprozess zu unterstützen, die geeignetere Wahl für die Umsetzung des Spiels ist. Zusätzlich zu den Erläuterungen in Kapitel 4, sind die vorteilhafteren Funktionen in Tabelle 5.1 markiert.

Um Gemeinsamkeiten und Unterschiede zwischen Game-Engines hinsichtlich ihrer Funktionen zu identifizieren, wurde ein Vergleich zwischen Unreal Engine 5 und Unity 2022 im Kontext eines Beispielprojekts durchgeführt. Dafür wurden die Dokumentationen der Engines eingesetzt und prototypische Tests durchgeführt. Es stellt sich heraus, dass beide Engines ähnliche Funktionalitäten vorweisen. Es ist deutlich geworden, dass Unreal Engine 5 in den Bereichen World-Building, Sound, KI, Culling-Optimierung und Modellierungsfunktionen umfangreichere Funktionen bietet.
Es zeigt sich, dass beide Engines die Grundkomponenten besitzen, wie in Abschnitt 2.3.2 beschrieben, und einzelne Aspekte als Synonyme betrachtet werden können. Allerdings wird auch deutlich, dass sie sich trotz der ähnlichen Grundstruktur in ihren Funktionen stark unterscheiden können, sei es aufgrund von zusätzlichen oder umfangreicheren Features.

Kriterium	Unity 2022	Unreal Engine 5
Plattformkompatibilität und Installation	Windows, macOS und Linux	Windows, macOS und Linux
Erlernbarkeit	**sehr großer Umfang an Lernressourcen**	großer Umfang an Lernressourcen
Lizenzmodell und Kosten	Stufenmodell mit Zugriffsbeschränkungen, Kostenfrei bis zu einem Umsatz von 100.000 USD innerhalb eines Jahrs	keine Zugriffsbeschränkungen, Kostenfrei bis zu einem Umsatz von 1 Million USD
Unterstütze Genre	2D, 3D und XR	2D, 3D und XR
Programmiersprache und Skripting	C# und Visual Scripting	C++ und Blueprints (Visual Sctipting)
Plattformunterstützung	Vielzahl an Desktop-, Konsolen- und Mobilplattformen	Vielzahl an Desktop-, Konsolen- und Mobilplattformen
Versionskontrolle und kollaborative Arbeit	Plastic SCM, Unterstützt externe Plattformen wie Git	Perforce und Subversion, Unterstützt externe Plattformen wie Git
Oberfläche und Szenen-Editor	**Ähnlicher Aufbau**	Ähnlicher Aufbau
Marketplace	Besitzt einen umfangreichen Marketplace	Besitzt einen umfangreichen Marketplace
Licht und Schatten	Realtime Global Illumination, Baked Global Illumination	**Lumen Global Illumination** Baked Global Illumination
Materialien und Objekte	einfache Modellierungsfunktionen, Shader-Graph	**Umfangreiche Modellierungsfunktionen**, **Fracure**, Material-Graph
Visuale Effekte und Partikel	Post-Processing, Full-Screen-Effekte, Partikel-Systemen	Post-Processing, Full-Screen-Effekte, **Niagara VFX-System**
UI-Editor	UI Toolkit	Widget Blueprints
Culling-Optimierung	Frustum-Culling, Occlusion Culling, LOD	Frustum-Culling, Occlusion-Culling, **Distanz-Culling**, **Nanite**, LOD
Umgang mit Physik und Kollision	-	-
Umgang mit Animation	Animator und Blend Tree	Animation-Blueprint und BlendSpace
Umgang mit Sound	kein spezielles Audio System	**MetaSound-System**
Erstellung von Landschaften	Besitzt Landscape-Tools	**Umfangreiche Landscape-Tools, World Partition, Wasser-System**
KI-Editor	NavMeshes, Key Patrol Points	NavMeshe, Pathfinding, **Behavior Trees**

Tabelle 5.1: Gemeinsamkeiten und Unterschiede der Funktionen von Unreal Engine 5 und Unity 2022

5.2 Auswertung

Die Nutzwertanalyse erweist sich als ein effektives Werkzeug, bei dem viele verschiedene Kriterien subjektiv bewertet werden können, um eine individuelle Entscheidungsfindung zu ermöglichen. Die Methode kann für den Vergleich verschiedener Engines eingesetzt werden, wobei die Ergebnisse individuell sind und sich aufgrund von unterschiedlichen Projektanforderungen und Erfahrungen unterscheiden können.
Während der Durchführung der Nutzwertanalyse ist aufgefallen, dass es schwierig ist, die Leistung und Benutzerfreundlichkeit zu bewerten. Dies liegt daran, dass es sich um Recherchen und prototypische Tests handelt, sodass keine Aussage darüber getroffen werden kann, wie sich die Engines mit steigender Komplexität verhalten. Zusätzlich ist zu beachten, dass die Benutzerfreundlichkeit sich während des Entwicklungsprozesses aufgrund zunehmender Erfahrung verändern kann. Daher ist der Fokus auf die Funktionen am wichtigsten, da diese direkt gegenübergestellt werden können.

Um einen allgemeinen Vergleich zwischen Engines durchführen zu können, können die identifizierten Vergleichskriterien als Grundlage verwendet werden. Die präsentierten Unterschiede und Gemeinsamkeiten zwischen Game-Engines beziehen sich auf Unreal Engine 5 und Unity 2022, in einem spezifischen Kontext, wodurch keine allgemeinen Aussagen über Game-Engines getroffen werden können. Anhand der identifizierten Kriterien können jedoch die jeweiligen Funktionen der Engines ermittelt und gegenübergestellt werden, um Unterschiede und Gemeinsamkeiten zwischen ihnen festzustellen.

5.3 Fazit

Die vorliegende Arbeit hat sich damit befasst, relevante Kriterien für den Vergleich von Game-Engines zu identifizieren und eine Strategie zur Auswahl der am besten geeigneten Engine. Die Analyse basierte auf einer umfassenden Betrachtung von Game-Engine-Komponenten, Softwarequalitätskriterien und existierenden Vergleichsarbeiten. Besonders relevant für den Vergleich von Game-Engines sind Kriterien, die mit den Projektanforderungen zusammenhängen.
Die Nutzwertanalyse wurde mit den identifizierten Kriterien kombiniert, um eine Strategie zu entwickeln, um Game-Engines vergleichen zu können, mit dem Ziel, die Eignung von Engine für ein spezifisches Projekt zu beurteilen. Dadurch können individuelle Anforderungen in die Entscheidung einfließen.
Die erarbeiteten Kriterien und die angewendete Methode ermöglichen einen strukturierten Vergleich von Game-Engines, der die Entscheidungsfindung bei der Auswahl einer Engine erleichtert. Sie können als Leitfaden für zukünftige Projekte dienen. Jedoch wurde auch deutlich, dass das Bewertungsverfahren aufwändig sein kann.

5.4 Nächste Schritte

Eine mögliche Weiterentwicklung dieser Arbeit könnte in der Erweiterung der bestehenden Kriterien um spezifischere Funktionen bestehen. Durch eine tiefere Analyse und Identifikation von Funktionen von Game-Engines könnten zusätzliche Kriterien aufgenommen werden. Das kann dazu beitragen, den Auswahlprozess zu präzisieren und zu beschleunigen, da nicht eigenständig die einzelnen Funktionen ermittelt werden müssten.
Ein weiterer Schritt wäre die Ausarbeitung der Bewertungsskala, um eine präzisere Bewertung der Engines zu ermöglichen.
Um die Ergebnisse der Analyse in der Praxis zu validieren, wäre es sinnvoll, das beschriebene Projekt vollständig in beiden Engines zu entwickeln. Dies ermöglicht eine direkte Beurteilung, ob die in der Analyse identifizierten Stärken und Schwächen im Entwicklungsprozess tatsächlich deutlich werden.

5.5 Ausblick

Die Videospielindustrie und die damit verbundenen Anforderungen an Videospiele unterliegen einem kontinuierlichen Wandel. Ebenso befinden sich die Technologien von Game-Engines in einem stetigen Entwicklungsprozess. Daher ist es unvermeidlich, dass die in dieser Arbeit identifizierten Vergleichskriterien im Laufe der Zeit angepasst und erweitert werden müssen.
Zukünftige Entwicklungen könnten neue Dimensionen in der Funktionalität von Game-Engines einführen, die möglicherweise nicht nur die Leistung, sondern auch Aspekte wie künstliche Intelligenz, Virtual Reality und immersive Nutzererlebnisse umfassen. Die Integration von neuen Technologien, Trends und Entwicklungen sollte daher aufmerksam verfolgt werden, um die Relevanz der Vergleichskriterien aufrechtzuerhalten.
Diese Dynamik erfordert eine flexible Herangehensweise an die Auswahl und Bewertung von Game-Engines, um stets die bestmöglichen Entscheidungen im Entwicklungsprozess zu treffen.

Literaturverzeichnis

[Ali16] Ali, Zulfiqar und Usman, Muhammad: A framework for game engine selection for gamification and serious games, in: *2016 Future Technologies Conference (FTC)*, S. 1199–1207, URL https://ieeexplore.ieee.org/document/7821753

[App22] Appfigures: Most used engines by mobile app and games developers worldwide as of July 2022, Statista (2022), URL https://www.statista.com/statistics/1326121/top-app-dev-engines-worldwide/, (09.09.2023)

[Ber17] Bergonse, Rafaello: Fifty Years on, What Exactly is a Videogame? An Essentialistic Definitional Approach, Springer Science+Business Media (2017), URL https://link.springer.com/article/10.1007/s40869-017-0045-4, (24.06.2023)

[EpioD] Epic Games: A world of possibilities, Epic Games Inc (o.D.), URL https://www.unrealengine.com/en-US/feed/all/games, (08.09.2023)

[Fle22] Fleck, Anna: Indie Games Gaining Ground Against the Grain in the U.S., Statista (2022), URL https://www.statista.com/chart/27207/percentage-point-change-in-the-share-of-us-gamers-saying-they-played-selected-game-genre/, (08.09.2023)

[Gre18] Gregory, Jason: *Game Engine Architecture*, A K PETERS (2018)

[Int11] International Organization for Standardization (ISO), International Electrotechnical Commission (IEC): ISO/IEC 25010:2011(en) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, Vogel Communications Group (2011), URL https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en, (19.08.2023)

[JunoD] Junge, Jens: Definitionen zum Spielen, Institut für Ludologie (o.D.), URL https://www.ludologie.de/spielforschung/definitionen-zum-spielen/, (18.06.2023)

[Pet12] Petridis, Panagiotis; Dunwell, Ian; Panzoli, David; Arnab, Sylvester; Protopsaltis, Aristidis; Hendrix, Maurice und de Freitas, Sara: Game engines selection framework for high-fidelity serious applications, URL https://api.semanticscholar.org/CorpusID:17642630

[Rab13] Rabin, Steven: *Game AI Pro*, CRC Press (2013)

[Sal03] Salen, Katie und Zimmerman, Eric: *Rules of Play: Game Design Fundamentals*, The MIT Press (2003)

[Sch23] Schmidt, Peer: NPC: Der Begriff „Non-Player Charakter“ im Gaming – Erklärung und Bedeutung, ingame (2023), URL https://www.ingame.de/news/npc-der-begriff-non-player-charakter-im-gaming-erklaerung-und-bedeutung-92091410.html, (19.07.2023)

[SN21] Saeidy-Nory, Sabine: Indie Games werden immer beliebter, game – Verband der deutschen Games-Branche e.V. (2021), URL https://www.game.de/indie-games-werden-immer-beliebter/, (26.02.2023)

[Sol23] Solis, Tobias: Nutzwertanalyse – Komplexe Entscheidungen rational treffen, Scribbr (2023), URL https://www.scribbr.de/methodik/nutzwertanalyse/, (05.10.2023)

[Sta23] Statista: Video game market revenue worldwide from 2017 to 2027, Statista (2023), URL https://www.statista.com/statistics/1344668/revenue-video-game-worldwide/, (20.01.2023)

[Ste19] Stefyn, Nadia: What is game design and how to become a game designer, CG Spectrum (2019), URL https://www.cgspectrum.com/blog/what-is-game-design, (29.06.2023)

[Ste23a] SteamDB: Engine · Unity · Technologies · SteamDB, SteamDB (2023), URL https://steamdb.info/tech/Engine/Unity/, (08.09.2023)

[Ste23b] SteamDB: Engine · Unreal · Technologies · SteamDB, SteamDB (2023), URL https://steamdb.info/tech/Engine/Unreal/, (08.09.2023)

[Ste23c] SteamDB: What are games built with and what technologies do they use?, SteamDB (2023), URL https://steamdb.info/tech/, (08.09.2023)

[TIG19] TIGA: What game engines do you currently use?, Statista (2019), URL https://www.statista.com/statistics/321059/game-engines-used-by-video-game-developers-uk/, (20.01.2023)

[Uni22] Unity Technologies: Unity gamimg report 2022, Unity Technologies (2022), URL https://images.response.unity3d.com/Web/Unity/%7B10460b81-b6e7-4784-a735-e7347afdf06e%7D_Unity-Gaming-Report-2022.pdf?utm_source=demand-gen&utm_medium=ceros&utm_campaign=acquisition&utm_content=2022-gaming-report-ebook&elqTrackId=d813896edf9f48e6af45a569577d8845&elqaid=4085&elqat=2, (20.01.2023)

[Uni23] Unity Technologies: Unity plan pricing and packaging updates, Unity Technologies (2023), URL https://blog.unity.com/news/plan-pricing-and-packaging-updates, (26.09.2023)

[UnioD] Unity Technologies: Unity Echtzeit-Entwicklungsplattform | 3D-, 2D-, VR- und AR-Engine, Unity Technologies (o.D.), URL https://unity.com/

de, (09.09.2023)

[Usl23] Uslenghi, Fabiano: Spieleentwickler in Sorge um finanziellen Ruin: Verbreitete Engine schockt mit Preisänderung, erntet massive Kritik, GameStar (2023), URL https://www.gamestar.de/artikel/entwickler-engine-preisaenderung-unity,3400568.html, (26.09.2023)

[Zan22] Zandt, Florian: Videospiele sind das lukrativste Unterhaltungsmedium, Statista (2022), URL https://de.statista.com/infografik/28970/geschaetzter-weltweiter-jahresumsatz-mit-videospielen-buechern-film-serie-musik/, (20.03.2023)

[Zub20] Zubek, Robert: *Elements of game design*, MIT Press (2020)

[Zö19] Zöbel, Dieter: *Echtzeitsysteme: Grundlagen der Planung*, Springer Vieweg (2019)

Abkürzungsverzeichnis

API Application Programming Interface
FPS Frames per Sekund
FMV Full-Motion-Video
GUI Grafische Benutzeroberfläche
HUD Heads-up-Display
IGC In-Game-Cinematic
ISO International Organization for Standardization
KI Künstliche Intelligenz
LOD Level of Detail
NPC Non-playable character
SDK Software Development Kit
XR Extended Reality

Abbildungsverzeichnis

Tabellenverzeichnis