Spieldokumentation: NecroDraft Arena

Maximilian Stricker

20. Juli 2025

Inhaltsverzeichnis

Kapitel 1

Installation und Setup

1.1 Systemanforderungen

- **OS:** Windows 10 (64-bit) oder höher
- **RAM:** 4 GB (empfohlen: 8 GB)
- **Speicherplatz:** 150 MB
- **Eingabe:** Maus und Tastatur

1.2 Installation für Entwickler

1.2.1 Voraussetzungen

1. **Unity Hub:** Download von https://unity.com/download
2. **Unity Editor:** Version 6000.1.4f1
3. **Git:** Optional für Versionskontrolle

1.2.2 Projekt Setup

1. Projektarchiv entpacken oder Repository klonen
2. Unity Hub öffnen → Add project from disk
3. `necrodraft-arena` Hauptordner auswählen
4. Erste Öffnung: Unity importiert Assets und Pakete (ca. 2-3 Minuten)

1.2.3 Dependencies

Das Projekt nutzt folgende Unity Packages:

- Universal Render Pipeline (17.1.0)
- 2D Animation (10.2.0)
- Input System (1.14.0)
- Visual Studio Editor (2.0.23)

1.3 Installation für Spieler

1.3.1 Windows Standalone

1. Release-Version herunterladen
2. ZIP-Archiv entpacken
3. `NecroDraftArena.exe` ausführen

1.4 Erste Schritte

1.4.1 Für Entwickler

- **Projekt-Struktur:** `Assets/Scenes/` (Spielszenen), `Assets/Scripts/` (C# Code), `Assets/ScriptableObjects/` (Part-Daten)
- **Test:** MainMenu.unity öffnen → Play-Button → Spiel testen
- **Build:** File → Build Settings → PC Standalone → Build and Run

1.4.2 Für Spieler

- **Steuerung:** Maus für UI-Navigation, Drag & Drop für Part-Assembly
- **Spielstart:** Necromancer-Klasse wählen → Parts sammeln → Combat starten

Kapitel 2

Übersicht

2.1 Spielname und Genre

NecroDraft Arena ist ein strategischer Top-Down Autobattler mit Deck-Building-Elementen. Das Spiel kombiniert taktische Positionierung mit modularem Minion-Crafting in einem Dark Fantasy Setting.

2.2 Kurzbeschreibung

Als Nekromant erschaffen Spieler einzigartige Untoten-Armeen durch das Zusammensetzen modularer Körperteile. Jedes Teil verleiht spezielle Fähigkeiten, die kombiniert noch stärker werden. In automatisierten Kämpfen bewähren sich die strategischen Entscheidungen der Aufbauphasen.

2.3 Zielgruppe

- **Primär:** Strategie-Enthusiasten (18-35 Jahre) mit Affinität zu Deck-Building und Autobattler-Spielen
- **Sekundär:** Spieler mit Interesse an Dark Fantasy und taktischen Kämpfen
- **Erfahrungslevel:** Mittlere bis hohe Gaming-Erfahrung

2.4 Plattform

- **Zielplattform:** PC (Windows 10/11, 64-bit)
- **Eingabegeräte:** Maus und Tastatur

2.5 Kern-Gameplay Loop

Der Spielzyklus besteht aus drei aufeinanderfolgenden Phasen:

1. **Combat Phase** (30-90 Sek): Automatisierte Kämpfe zwischen Minion-Armeen
2. **Assembly Phase** (2-4 Min): Ausrüstung, Merge und Verwaltung der Parts

3. **Results Phase** (10 Sek): Sieg/Niederlage Feedback und Part-Belohnungen

Nach jedem Kampf erhält der Spieler automatisch 3 zufällige Parts, die direkt ins Inventar wandern. In der Assembly-Phase können Parts an Minions ausgerüstet, miteinander zu stärkeren Varianten gemergt oder gelöscht werden - alles in derselben Szene.

2.6 Part-System Übersicht

2.6.1 Part-Aufbau

Jeder Part verfügt über ein dynamisches Punktebudget, das auf Stats verteilt wird:

- **Common Parts:** 12 Punkte Budget (1-2 Stats)
- **Uncommon Parts:** 20 Punkte Budget (2 Stats)
- **Rare Parts:** 32 Punkte Budget (2-3 Stats)
- **Epic Parts:** 50 Punkte Budget (3-4 Stats)

2.6.2 Thematische Spezialisierung

Parts gehören zu einem von drei Untoten-Themen mit distinkten Stat-Affinitäten:

- **Skeleton:** Geschwindigkeit, Kritische Treffer, Reichweite (Glass Cannon)
- **Zombie:** Gesundheit, Verteidigung, Ausdauer (Tank)
- **Ghost:** Ausgewogene Stats, Vielseitigkeit (Hybrid)

2.6.3 Merge-System

Drei identische Parts (gleicher Slot und Seltenheit) können zu einem stärkeren Part der nächsten Seltenheitsstufe verschmolzen werden.

2.7 Alleinstellungsmerkmale

- **Modulares Part-System:** Vollständig anpassbare Einheiten durch 4-Slot Ausrüstungssystem (Head/Torso/Arms/Legs)
- **Dynamische Part-Generierung:** Prozedural generierte Parts mit themenbasierten Stat-Affinitäten statt fester Items
- **Advanced Merge-Mechanik:** 3 gleiche Parts verschmelzen zu einem stärkeren Part der nächsten Seltenheitsstufe
- **Budget-basierte Stats:** Jeder Part hat ein Punktebudget, das dynamisch auf 1-4 Stats verteilt wird
- **Thematische Spezialisierung:** Skeleton (Glass Cannon), Zombie (Tank), Ghost (Hybrid) mit distinkten Stat-Profilen
- **Integrated Assembly:** Part-Management und Minion-Ausrüstung in einer einzigen, nahtlosen Szene

Kapitel 3

Gameplay

3.1 Core Gameplay Loop

NecroDraft Arena folgt einem zyklischen Gameplay-Loop mit zwei Hauptphasen:

1. **Assembly Phase** (1-3 Minuten): Strategische Optimierung der Minion-Armee
2. **Combat Phase** (30-90 Sekunden): Automatisierter Kampf zwischen den Armeen

3.1.1 Assembly Phase

Nach jedem Kampf erhält der Spieler automatisch 3 zufällige Parts ins Inventar. In dieser Phase können Parts:

- An Minions in deren 4 Ausrüstungsslots (Head/Torso/Arms/Legs) ausgerüstet werden
- Miteinander zu stärkeren Varianten gemergt werden (3 identische Parts → 1 höhere Seltenheit)
- Aus dem Inventar gelöscht werden, um Platz zu schaffen

3.1.2 Combat Phase

Vollständig automatisierte Kämpfe basieren auf:

- **Einfache KI-Regeln:** Units greifen den Gegner ganz links an, ausgenommen sind Gegnertypen die unterschiedliche Positionen angreifen.
- **Ability-Triggern:** Parts aktivieren spezielle Fähigkeiten basierend auf Kampfereignissen
- **Siegbedingung:** Die letzte überlebende Armee gewinnt die Runde

3.2 Part-System

3.2.1 Ausrüstungsslots

Jeder Minion verfügt über 4 Ausrüstungsslots mit thematischen Schwerpunkten:

- **Head:** Präzisions- und mentale Fähigkeiten (Kritische Chance, Rüstungsdurchdringung, Kritischer Schaden)
- **Torso:** Defensive und Ausdauer-Fähigkeiten (Rüstung, Gesundheit, Angriff)
- **Arms:** Offensive Fähigkeiten (Angriff, Kritische Chance, Rüsterungsdurchdringung)
- **Legs:** Bewegungs-Fähigkeiten (Ausweichen, Rüstung, Gesundheit)

3.2.2 Special Abilities

Parts können spezielle Fähigkeiten basierend auf ihrer Kategorie besitzen:

Guardian Abilities

- **Taunt:** Zwingt Gegner, diesen Minion anzugreifen
- **Shield Wall:** Blockiert Angriffe auf Verbündete hinter diesem Minion
- **Damage Sharing:** Teilt einen Prozentsatz des Schadens mit benachbarten Verbündeten

Assault Abilities

- **Flanking:** Bonusschaden wenn von Randpositionen angreifend
- **Focus Fire:** Erhöhter Schaden gegen Gegner unter 50% HP
- **Momentum:** Steigender Angriff pro Runde (zurückgesetzt bei Schaden)

Marksman Abilities

- **Range Attack:** Kann hintere Reihe von hinten angreifen
- **Overwatch:** Chance auf Konterangriff bei Verbündeten-Angriffen
- **Hunter:** Massiver Bonusschaden gegen Gegner unter 25% HP

Support Abilities

- **Healing:** Heilt verwundete Verbündete pro Runde
- **Inspiration:** Gibt benachbarten Verbündeten permanenten Angriffsbonus
- **Battle Cry:** Einmaliger kampfweiter Angriffsbuff für alle Verbündeten

Trickster Abilities

- **Mobility:** Kann Positionen mit Verbündeten tauschen
- **Phase Step:** Chance, Schaden durch Positionstausch zu vermeiden
- **Confuse:** Lässt Gegner zufällige Ziele angreifen

3.2.3 Thematische Spezialisierung

Bone Theme (Aggressiv, präzise):

- Hohe Affinität: Assault (60%) und Marksman (30%) Abilities
- Schwerpunkt: Angriff, Kritische Treffer, Rüstungsdurchdringung

Flesh Theme (Defensiv, ausdauernd):

- Hohe Affinität: Guardian (60%) und Support (30%) Abilities
- Schwerpunkt: Gesundheit, Verteidigung, Teamunterstützung

Spirit Theme (Vielseitig, unberechenbar):

- Hohe Affinität: Trickster (40%) und ausgewogene Verteilung
- Schwerpunkt: Utility-Fähigkeiten, flexible Stat-Verteilung

3.3 Steuerung

3.3.1 Maussteuerung

- **Drag & Drop:** Parts zwischen Inventar und Minionslots bewegen
- **Rechtsklick:** Partdetails und Mergeoptionen anzeigen
- **Linksklick:** Menüs bestätigen, Kampf starten

3.4 Strategische Tiefe

3.4.1 Build-Archetypen

- **Guardian Build:** Taunt + Shield Wall für maximalen Schutz
- **Assault Build:** Flanking + Focus Fire für hohen Schaden
- **Marksman Build:** Range Attack + Hunter für Fernkampfdominanz
- **Support Build:** Healing + Inspiration für Teamunterstützung
- **Trickster Build:** Mobility + Phase Step für Unberechenbarkeit

3.4.2 Strategische Entscheidungen

- **Commitment vs. Flexibilität:** Slots für Macht opfern oder Vielseitigkeit bewahren
- **Part-Verteilung:** Einen Superminion oder ausgewogenes Team erstellen
- **Seltenheits-Abwägung:** Epic Part nehmen oder auf Synergie warten
- **Counter-Strategien:** Gegnerische Builds antizipieren und kontern

Kapitel 4

Weitere Inhalte

4.1 Thematischer Rahmen

NecroDraft Arena ist im Dark Fantasy Setting angesiedelt, in dem der Spieler als Nekromant auftritt. Das Spiel verzichtet bewusst auf eine tiefe Narrative und konzentriert sich stattdessen auf mechanische Tiefe und strategisches Gameplay.

4.1.1 Setting

- **Genre:** Dark Fantasy mit strategischen Elementen
- **Protagonisten:** Verschiedene Nekromantenklassen mit unterschiedlichen Spezialisierungen
- **Zielkonflikt:** Arena basierte Kämpfe zwischen Untotenarmeen

4.2 Visueller Stil und Farbschema

4.2.1 Designprinzipien

- **Top-Down Perspektive:** Klassische strategische Übersicht
- **Modularer Stil:** Körperteile sind visuell unterscheidbar und kombinierbar

4.2.2 Farbpalette

- **Primärfarben:** Dunkle Grautöne und Blautöne für UI-Elemente
- **Akzentfarben:** Verschiedene Farben für Part-Seltenheiten (Common, Rare, Epic)
- **Thematische Farben:** Knochenweiß für Skeleton-Theme, Ätherisch-Blau für Ghost-Theme

4.3 Audio-System

Das Spiel nutzt ein zentralisiertes Audio-System mit MusicManager und SettingsManager.

4.3.1 Komponenten

- **Hintergrundmusik:** Durchgehende Musik mit Fadeübergängen
- **UI-Sounds:** Button-Clicks und Menü-Navigation
- **Kampf-Audio:** Angriffs- und Schadenssounds (geplant)

4.3.2 Audio-Einstellungen

- Separate Lautstärkeregelung für Master, Musik und Soundeffekte
- Persistente Speicherung der Audioeinstellungen

4.4 KI Verhalten

4.4.1 Combat KI

Das Kampfsystem basiert auf positionsabhängigen Targeting-Regeln mit 6 verschiedenen Gegnertypen:

- **Bruiser Enemies:** Greifen immer den Minion ganz links in der vorderen Reihe an
- **Archer Enemies:** Zielen prioritär auf die hintere Reihe (weiteste Position)
- **Assassin Enemies:** Attackieren den Minion mit der niedrigsten aktuellen HP (beliebige Position)
- **Sniper Enemies:** Fokussieren den Minion mit dem höchsten Angriffswert
- **Bomber Enemies:** Verursachen Flächenschaden auf alle benachbarten Positionen
- **Guardian Enemies:** Blockieren Angriffe und schützen andere Gegner

4.4.2 Positionssystem

- **Formation:** 2 Reihen mit je 3 Positionen (vorne/hinten)
- **Vordere Reihe:** Erhält mehr Schaden, blockiert Zugang zur hinteren Reihe
- **Hintere Reihe:** Geschützt, aber begrenzte Angriffsmöglichkeiten
- **Strategische Platzierung:** Spieler muss Formation an Gegnertypen anpassen

4.4.3 Balance Mechaniken

- Vorhersagbare Targeting-Muster ermöglichen strategische Planung
- Deterministische Kämpfe für optimale Partoptimierung
- Skalierung der Gegnerstärke basierend auf Wellenfortschritt

Kapitel 5

Technische Dokumentation

5.1 Game Engine und verwendete Tools

5.1.1 Unity Engine

- **Version:** Unity 6+ mit 2D Core Template
- **Render Pipeline:** Universal Render Pipeline (URP) 17.1.0
- **Input System:** New Unity Input System 1.14.0
- **Zielplattform:** PC Standalone (Windows 64-bit)

5.1.2 Externe Pakete

- 2D Animation (10.2.0) für Sprite-Animationen
- Visual Studio Editor (2.0.23) für Code-Integration

5.2 Code Architektur

5.2.1 Zentrale Managersysteme

Das Projekt folgt einem Manager-Pattern mit statischen Klassen für globale Systeme:

- **GameData:** Verwaltet Spielfortschritt und Wellenstatus
- **MinionManager:** Zentrale Minionroster Verwaltung
- **PlayerInventory:** Part Inventar mit Eventsystem
- **SaveSystem:** Serialisierung und Persistierung

5.2.2 ScriptableObject Architektur

Datenklassen sind als ScriptableObjects implementiert für Design-Flexibilität:

- **PartData:** Modulare Körperteil Definitionen

- **NecromancerClass:** Spielerklassen mit Boni
- **MinionData:** Basisminion Eigenschaften
- **EnemyData:** Gegnerdefinitionen

5.2.3 UI Architektur

Szenenbasierte UI Verwaltung mit Managerkomponenten:

- **MinionAssemblyManager:** Partausrüstung und Minionverwaltung
- **MainMenuManager:** Hauptmenü mit Save/Load Integration
- **SettingsManager:** Audio- und Grafikeinstellungen
- **ClassSelectionManager:** Klassenauswahl mit Preview

5.3 Part System Implementation

5.3.1 Stat Budget System

Parts nutzen ein punktebasiertes Stat-System:

- **Stat-Kosten:** HP (1 Punkt), Attack (1 Punkt), Defense (2 Punkte), Kritische Chance (2 Punkte), Kritischer Schaden (1 Punkt), Rüstungsdurchdringung (3 Punkte)
- **Rarity Budget:** Common (8 Punkte), Uncommon (14 Punkte), Rare (22 Punkte), Epic (35 Punkte)
- **Special Abilities:** Variable Kosten je nach Stärke (3-20 Punkte)
- **Dynamische Generation:** Themenbasierte Statverteilung und Ability-Zuordnung

5.3.2 Special Abilities Kosten

- **Guardian:** Taunt (4/6/8), Shield Wall (6/9/12), Damage Sharing (5/8/11)
- **Assault:** Flanking (4/7/10), Focus Fire (5/8/11), Momentum (6/9/12)
- **Marksman:** Range Attack (8/12/16), Overwatch (7/11/15), Hunter (5/8/11)
- **Support:** Healing (3/5/7), Inspiration (4/6/8), Battle Cry (8/12/16)
- **Trickster:** Mobility (10/15/20), Phase Step (12/18/24), Confuse (6/9/12)

5.4 Build und Deployment

5.4.1 Buildkonfiguration

- **Plattform:** PC Standalone (Windows x64)
- **Kompression:** LZ4 für Balance zwischen Größe und Ladezeit
- **Development Build:** Für Debug-Features in Prototyping

5.4.2 Assetmanagement

- Resources-Ordner für dynamisch geladene ScriptableObjects
- Addressables System für zukünftige Modularität
- Texturekompression für optimale Dateigröße

Kapitel 6

Entwicklungsprozess

6.1 Projektplanung und Meilensteine

6.1.1 Entwicklungsansatz

Das Projekt folgte einem iterativen Prototyping-Ansatz mit Fokus auf Core-Mechaniken:

1. **Phase 1:** Grundlegende Part System Implementierung
2. **Phase 2:** UI System und Minion Assembly Interface
3. **Phase 3:** Combatsystem und WWellenprogression
4. **Phase 4:** Savesystem und Gamebalance

6.1.2 Technologie Entscheidungen

- **Unity 6:** Für moderne 2D-Features und URP-Integration
- **ScriptableObjects:** Für datengetriebenes Design ohne Codeänderungen
- **Eventsystem:** Für lose gekoppelte Manager Kommunikation

6.2 Herausforderungen und Lösungsansätze

6.2.1 Partsystem Komplexität

Problem: Ursprünglich prozentbasierte Stats waren schwer zu balancieren und zu verstehen.

Lösung: Migration zu punktebasiertem Budget-System mit klaren Kosten pro Statpunkt. Zuerst waren auch Setboni geplant, durch das Mergesystem wurde dies doch gestrichen und durch ein Fähigkeitensystem ersetzt. Die Fähigkeiten funktionieren auch über das punktebasiertem Budget-System, was die Balance vereinfacht.

6.2.2 UI Skalierung

Problem: Inkonsistente UI Darstellung auf verschiedenen Auflösungen. Elemente wie Buttons und Tooltips waren nicht skalierbar.

Lösung: SceneConsistencyManager für automatische Canvas-Skalierung und Resolution-Management. Außerdem wurde die generelle Auflösung auf Full-HD (1920x1080) festgelegt, um das Skalieren der UI zu vereinfachen. Pixelbasierte UI-Elemente wurden mit verschiedenen Skalierungseinstellungen versehen (z.B. keine Kompression und kein Filter).

6.2.3 Savesystem Integration

Problem: Komplexe Objektreferenzen zwischen Minions, Parts und ScriptableObjects. Parts wurden nicht im Inventar gespeichert, sondern nur in der Scene. Die Darstellung der Parts im Inventar war nicht persistent, was zu Inkonsistenzen führte.

Lösung: Serialisierungsklassen mit Stringbasierten Referenzen statt direkter Objektlinks.

6.3 Designentscheidungen

6.3.1 Automatisierte Kämpfe

Entscheidung: Vollständig automatisierte Combatphase ohne Spielereingriff.

Begründung: Fokus auf strategische Partoptimierung statt Mikromanagement im Kampf.

6.3.2 Single Scene Assembly

Entscheidung: Alle Part Management Features in einer Szene kombiniert.

Begründung: Reduziert Szenenwechsel und ermöglicht dem Spieler alle Features (bspw. Mergen der Parts oder Verwalten der Minions) auf einen Blick zu nutzen.

6.3.3 Managerpattern

Entscheidung: Zentrale statische Manager für globale Spielzustände.

Begründung: Vereinfacht Datenzugriff zwischen Szenen und reduziert Singletonkomplexität.

6.4 Erreichte Ziele und Weiterentwicklung

6.4.1 Erfolgreich implementiert

- Vollständig funktionsfähiges Partsystem mit speziellen Fähigkeiten
- Intuitive Drag & Drop UI für Part Assembly
- Persistente Save/Load Funktionalität
- Skalierbare Wellenprogression mit Freischaltung von Minions

- Modulare Codebase für einfache Erweiterungen

6.4.2 Verbesserungspotential

- **Performance:** Object Pooling für UI Elemente
- **Content:** Weitere Klassen, mehr Partthemes, Ausreifung des Gameplayflows durch interaktive Entscheidungen nach jeder Welle (z.B. Pfadfindung)
- **Polish:** Animationen und Visual Effects
- **Balance:** Erweiterte Gameplaytests für Stattuning und Fähigkeitenbalancing

6.4.3 Fazit

NecroDraft Arena demonstriert erfolgreich die Kernmechaniken eines modularen Deck-Building-Autobattlers. Das fundierte technische Framework ermöglicht einfache Contenterweiterungen und bietet eine solide Basis für die Weiterentwicklung zum vollständigen Spiel.

Kapitel 7

Spielanleitung

7.1 Einleitung

Willkommen bei NecroDraft Arena! Als Nekromant erschaffst du mächtige Untoten-Armeen durch das strategische Kombinieren von Körperteilen. Jeder Teil verleiht deinen Minions einzigartige Fähigkeiten, die durch zusammenfügend immer stärker werden. In automatisierten Kämpfen bewähren sich die strategischen Entscheidungen der Aufbauphasen.

7.2 Spielstart

7.2.1 Erste Schritte

1. **Klassen-Auswahl:** Wähle deine Nekromantenklasse für verschiedene Spezialisierungen (momentan nur Bone Weaver verfügbar)
2. **Erstes Minion:** Du startest mit einem Basisminion und einigen Startteilen
3. **Assembly Phase:** Übersicht deines Minions, deines Inventars und der verfügbaren Parts
4. **Combat Phase:** Positioniere dein Minion an einer der 6 Positionen in der Arena und beobachte den automatisierten Kampf

7.3 Steuerung

7.3.1 Maussteuerung

- **Drag & Drop:** Parts aus dem Inventar auf Minionslots ziehen
- **Hover:** Tooltips für detaillierte Partinformationen
- **Buttons:** Navigation zwischen Minions und Modi (bspw. zusammenfügen der Parts oder ausrüsten der Parts an einem Minion)

7.3.2 UI Navigation

- **Pfeiltasten:** (wenn vorhanden) Zwischen Minions wechseln
- **Inventar:** Verfügbare Parts am rechten Teil des Bildschirms.
- **Part-Slots:** Head, Torso, Arms, Legs für jeden Minion

7.4 Kernmechaniken

7.4.1 Partsystem

- **4 Slot Typen:** Jeder Part passt nur in seinen spezifischen Slot
- **Seltenheiten:** Common (grau), Uncommon (grün), Rare (blau), Epic (violett)
- **Themes:** Bone (aggressiv), Flesh (defensiv), Spirit (vielseitig)

7.4.2 Special Abilities

- **Kategorien:** Guardian, Assault, Marksman, Support, Trickster
- **Aktivierung:** Parts haben individuelle Fähigkeiten basierend auf ihrem Budget
- **Beispiele:** Taunt (zwingt Gegner zum Angriff), Range Attack (Fernkampf), Healing (heilt Verbündete)

7.4.3 Partmanagement

- **Merging:** 3 identische Parts $\rightarrow$ 1 höhere Seltenheit
- **Löschen:** Unerwünschte Parts entfernen für Platz
- **Strategische Planung:** Potentiell Special Abilities über Rohstats priorisieren

7.5 Progression

7.5.1 Wellensystem

- **Welle 1-7:** Akt I - Grundlagen lernen (max. 3 Minions)
- **Welle 8-14:** Akt II - Erweiterte Strategien (max. 4 Minions)
- **Welle 15-20:** Akt III - Endgame Content (max. 5 Minions)

7.5.2 Minionunlocks

- Welle 5: 2. Minionslot
- Welle 9: 3. Minionslot
- Welle 13: 4. Minionslot
- Welle 17: 5. Minionslot

7.6 Strategische Tipps

7.6.1 Für Anfänger

- **Abilities nutzen:** Parts mit Special Abilities sind oft stärker als reine Statparts
- **Themefokus:** Spezialisiere Minions auf ein Theme für bessere Fähigkeitensynergien
- **Slotplanung:** Behalte Parts für strategische Builds und Merging Möglichkeiten

7.6.2 Fortgeschrittene Taktiken

- **Partbudgeting:** Rare und Epic Parts strategisch verteilen
- **Komposition der Armee:** Guardian, Assault, Support Rollen ausbalancieren
- **Mergetiming:** Parts upgraden vs. Fähigkeitendiversität abwägen

7.7 Häufige Fehler

- **Fähigkeiten ignorieren:** Einzelne hohe Stats ohne spezielle Fähigkeiten sind oft schwächer
- **Ungenutzte Parts:** Inventar nicht voll ausnutzen
- **Mischen der Themes:** Verschiedene Themes pro Minion verwässern Synergien
- **Zu frühes Mergen:** Parts mergen bevor strategische Builds etabliert sind

Kapitel 8

Quellen und Referenzen

8.1 Game Design Referenzen

8.1.1 Gameplayinspiration

- **Teamfight Tactics (Riot Games):** Auto-Battler Mechaniken und strategische Positioning
- **Hearthstone Battlegrounds (Blizzard):** Ebenfalls Auto-Battler Mechaniken mit Fokus auf dem Kartensystem
- **Slay the Spire (Mega Crit):** Deckbuilding Progression
- **Divinity: Original Sin 2 (Larian):** Modulare Equipmentsets und Fähigkeitenkombinationen

8.1.2 UI/UX Design

- **Auto Chess Genre:** Drag & Drop Interface
- **Path of Exile (Grinding Gear Games):** Komplexe Inventorymanagement
- **Monster Hunter (Capcom):** Equipmentslot basierte Progression

8.2 Technische Ressourcen

8.2.1 Unity Engine

- **Unity Technologies:** Unity 6 LTS Engine und Universal Render Pipeline
- **Unity Documentation:** Official Unity Scripting API Reference
- **Unity Learn Platform:** ScriptableObject Architecture Tutorials

8.2.2 Asset Herkunft und Eigenleistung

Eigenleistung

- **Code:** Vollständige C# Skripte und Unity Implementierung in Eigenarbeit
- **Game Design:** Spielmechaniken, Balancing und Systemarchitektur
- **UI/UX Design:** Interfacelayout und Benutzerführung
- **Art Assets:** Alle visuellen Elemente außer explizit aufgeführten Ausnahmen

AI-Generierte Inhalte

- **Hintergrundbilder:** AssemblyBackground (2), Background und GameplayBackground - erstellt mit Imagen 4 von AI Studio
- **Klassenbilder:** Alle Sprites im Ordner sprites/classes - erstellt mit Imagen 4 von AI Studio

Externe Ressourcen

- **Fonts:** TextMeshPro Standardschriftarten (Unity)
- **Hintergrundmusik:** von https://pixabay.com/users/clavier-music-16027823/?utm_source=link-attribution&utm_medium=referral&utm_campaign=music&utm_content=354468
- **Button-Soundeffekte:** von https://ateliermagicae.itch.io/be-not-afraid-uimenu-sfx?download

8.3 Code Bibliotheken und Tools

8.3.1 Unity Packages

- Universal Render Pipeline (com.unity.render-pipelines.universal) - Version 17.1.0
- 2D Animation (com.unity.2d.animation) - Version 10.2.0
- Input System (com.unity.inputsystem) - Version 1.14.0
- Visual Studio Editor (com.unity.ide.visualstudio) - Version 2.0.23

8.3.2 Development Tools

- **Visual Studio Code:** Primäre IDE für C# Development
- **Git:** Versionskontrolle und Source Management
- **Unity Hub:** Projekt- und Engine-Management

8.4 Externe Entwicklerressourcen

8.4.1 Community und Documentation

- **Unity Forums:** Community Support für technische Probleme
- **Stack Overflow:** C# und Unity-spezifische Programmierungsfragen
- **GitHub:** Open Source Unity-Projekte als Referenz

Kapitel 9

Demo und Zusatzmaterialien

9.1 Projekt-Repository

9.1.1 Source Code

- **Repository:** necrodraft-arena
- **Hauptentwicklung:** Unity Projekt mit vollständigem Source Code
- **Dokumentation:** LaTeXbasierte Projektdokumentation

9.1.2 Projekt-Struktur

- `Assets/Scripts/` - Gesamte C# Codebase
- `Assets/ScriptableObjects/` - Part- und Class-Definitionen
- `Assets/Scenes/` - Spielszenen
  (MainMenu, ClassSelection, MinionAssembly, Gameplay)
- `ProjectDocumentation_LaTeX/` - Diese Dokumentation

9.2 Gameplay-Features

9.2.1 Implementierte Core-Features

- ✓ Vollständiges Part-System mit Fähigkeiten
- ✓ Drag & Drop UI für intuitive Partassembly
- ✓ Automatisierte Combatsimulation
- ✓ Wellenprogression mit Minionunlocks
- ✓ Save/Load System für Spielfortschritt
- ✓ Mergesystem für Partupgrades
- ✓ Multiklassensystem mit verschiedenen Spezialisierungen

9.2.2 Demoumfang

- **Spielbare Waves:** 1-20 mit progressiver Schwierigkeit
- **Part-Varietät:** 3 Themes (Skeleton, Ghost, Zombie) mit jeweils 4 Slot-Typen (Allerdings nur Skeleton Parts implementiert)
- **Progression:** Minionunlocks und Raritysteigerung über Wellenfortschritt

9.3 Technische Demonstration

Im Repository befindet sich ein Ordner technische Demonstration, der eine ausführliche technische Präsentation der Kernfeatures enthält. Diese umfasst:

- **Gameplay-Demo:** Kurzes Gameplay-Video, das die Kernmechaniken zeigt
- **UI-Demo:** Screenshots der Benutzeroberfläche und deren Interaktionen

9.4 Weiterentwicklung

9.4.1 Geplante Features

- Erweiterte Visual Effects und Animationen
- Zusätzliche Partthemes und Implementierung der Klassen
- Balancingverbesserungen basierend auf Gameplaytests
- Verbesserung des Nutzererlebnisses durch UI-Optimierungen

9.4.2 Potentielle Erweiterungen

- Multiplayermodi für kompetitives Gameplay
- Moddingsupport durch ScriptableObjectsystem
- Mobile Platform Portierung
- Narrative Kampagne