Hochschule Aalen

Spieleprogrammierung
Studiengang : Informatik

Spieleentwicklung
Projektdokumentation

bei

Hochschule Aalen

von

Veronika Scheller 79888
Simon Ruttmann 80751
Michael Ulrich 77607

Einreichungsdatum : 28. Juni 2022

Kurzfassung

Im Rahmen der Vorlesung Spieleprogrammierung wurde ein 2,5D Spiel entwickelt. Dies bedeutend, dass die Anwendung mittels 3D Elementen entwickelt wurde und der Benutzer über eine 2D Sicht mit dem Spiel interagiert. Das Spielkonzept der Anwendung ist hierbei rundenbasiert.

Der Benutzer sowie ein KI-gesteuerter Gegner erhält zu Beginn des Spiels eine Reihe von Figuren. Ziel des Spiels ist es durch Bewegung und Angriff der eigenen Figuren alle gegnerischen Figuren zu besiegen.

Die Art des Designs ist am ehesten dem Cel Shading Art zuzuordnen. Im Spiel wurden Low und High Resolution Elemente verwendet.

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Innerhalb der Entwicklung eines Spiels sollen die wichtigsten Phasen der Spielentwicklung durchlaufen werden. Im Projekt soll dabei unter anderem in den Phasen Ideenfindung, Konzepterarbeitung, Implementierung, Testing & Debugging sowie der Inbetriebnahme praktische Erfahrung gesammelt werden.

In den begleitenden Vorlesungen werden darüber hinaus grundlegende Methoden und weitergehende theoretische und praktische Konzepte vermittelt.

1.2. Problemstellung und -abgrenzung

Die Herausforderung des Projektes ist es, innerhalb einer Ideenfindungsphase mögliche Spielkonzepte zu ermitteln, diese anschließend zu evaluieren und nach einer Gegenüberstellung die Geeignetste auszuwählen.

Folgend muss ein klares Spielkonzept definiert werden und eine Eingrenzung des Umsetzungsumfangs erfolgen.

Abschließen soll innerhalb einer Implementierungsphase das entwickelte Spielkonzept umgesetzt werden.

Im Rahmen der oben genannten Punkte muss berücksichtigt werden, dass zu einem vorgegebenen Abgabezeitpunkt ein möglichst rundes Produkt entsteht.

1.3. Ziel der Arbeit

Das Ziel des Projekts ist die Entwicklung und Implementierung eines Spielkonzepts. Im Rahmen der Umsetzung sollen die gelernten Grundlagen der Vorlesung beachtet werden.

Zudem soll innerhalb einer Zwischenpräsentation der aktuelle Entwicklungsstand präsentiert werden. In einer weiteren Präsentation zum Ende des Projekts sollen die, aus dem Projekt hervorgegangen, Ergebnisse vorgestellte werden.

Die entwickelte Anwendung ist in einer ausführbaren Form, zusammen mit einer beigelegten Spielanleitung abzugeben.

1.4. Vorgehen

Das im Projekt angewandte Vorgehen für die Entwicklung des Spiels erfolgt auf Basis der in der Vorlesung vermittelten Phasen der Spieleentwicklung. Die zu durchlaufenen Phasen sind in der folgenden Abbildung dargestellt.

In der ersten Phase, werden sich Gedanken über die Motivation und Ziel des Spiels gemacht. Die Motivation zur Entwicklung der Anwendung liegt in diesem Projekt darin begründet, das mehrere Gruppenmitglieder bereits konzeptähnliche Spiele gespielt haben, jedoch aufgrund von Mängeln diese nicht den gewünschten Spielspaß erbrachten. Innerhalb dieses Projekts bietet sich daher die Möglichkeit ein 'besseres' Spiel zu entwickeln.

Bei der Ideenentwicklung werden mögliche Vorschläge aller Mitglieder gesammelt. Daraufhin erfolgen mehrere Gruppentreffen. Alle von den Gruppenmitgliedern gesammelten Ideen werden über einem Pitch vorgestellt. Durch Mehrheitsabstimmung wird anschließend eine Entscheidung getroffen.

Beim zweiten Teil der Ideenentwicklung wird das Konzept für das Spiel erstellt. Die grundlegenden Spielregeln und Spielabläufe werden festgelegt. Die Anforderungen an die grafische Darstellung werden durch Bildreferenzen festgehalten.

Bei der Implementierung in der dritten Phase wird die Spiellogik und Darstellung parallel entwickelt und in einem zweiten Schritt die beiden Komponenten zu einem Spiel zusammengefügt.

In der vierten Phase findet eine umfangreiche Test- und Balancingphase statt. Außerdem erfolgt hier die benötigten Schritte um das Projekt bauen zu können und eine ausführbare Datei zu erhalten.

Im letzten Schritt erfolgt eine Verteilung der in Phase 4 erstellten ausführbaren Datei an ausgewählte Nutzer.

Mit der Fertigstellung des Spiels wurde damit der Spielentwicklungszyklus erfolgreich beendet.

2. Game Idea

In einer initialen Ideensammlung wurden mittels einer Mindmap Konzepte und Spieleideen aufgeschrieben.

Die aus dieser Ideensammlung hervorgegangenen Ideen und Konzepte wurden in einer Diskussionsrunde besprochen. Dabei konnten einige Ideen durchgestrichen werden, während sich andere als Favoriten herauskristallisierten. Diese wurden in einer zweiten Diskussionsrunde noch einmal mit einem Beispiel genauer definiert.

In einem Abstimmungsprozess gewann das ’2D Rundenbasierte Strategiespiel’ am Ende knapp die Ideenfindung.

Die Spielinspiration hierfür stammt grundlegend von dem Browserspiel Elvenar. Der Designprozess begann mit der Idee, ein Strategiespiel zu entwickeln, bei dem Figuren zweier Lager in einem rundenbasierten Kampf auf einem hexagonalen Feld kämpfen.

3. Game Design

3.1. Spielablauf

Startet der Spieler die Anwendung, so gelangt er in die UI. Über diese kann er ein neues Spiel starten oder die Anwendung beenden.

Wird das Spiel gestartet, so gelangt der Spieler in eine neue Ansicht. Diese ist folgend dargstellt.

Der Spieler hat eine Reihe von unterschiedlichen Figuren. Diese sind:

- 2 Krieger
- 2 Bogenschützen
- 1 Magier
- 1 Paladin

Der Gegner besitzt die gleiche Anzahl und die gleichen Typen an Figuren wie der Spieler.

Das Spiel highlighted zu Beginn eine Figur, sowie alle möglichen Züge dieser. Durch klicken auf ein Feld bewegt sich die Figur auf dieses. Befindet sich eine gegnerische Figur auf diesem, so greift die Figur die gegnerische Figur an.

Bei einer Bewegung wird eine Sequenz an Animationen ausgeführt, welche die Figur auf das angeklickte Feld bewegt.

Bei einem Angriff wird ebenfalls eine Sequenz an Animationen ausgeführt. In dieser wird z.B. gezeigt, wie die angreifende Figur einen Bogenschuss auf die verteidigende Figur feuert und diese ein Schmerzgeräusch von sich gibt und zurückschreckt. Zu Beginn des Angriffs wird die Lebensanzeige der angegriffenen Figur, abhängig von dem Schaden der angreifenden Figur reduziert. Fällst diese unter 0, so stirbt diese.

Nachdem der Spieler mit allen Figuren gezogen hat führt die gegnerische KI ihre Züge aus. Daraufhin ist der Spieler wieder an der Reihe. Figuren, welche nicht die Möglichkeit haben einen Angriff oder eine Bewegung durchzuführen, werden übersprungen.

Besitzt einer der Spieler keine Figuren mehr erscheint ein End-Screen. Über dieses kann man ein neues Spiel starten oder die Anwendung beenden.

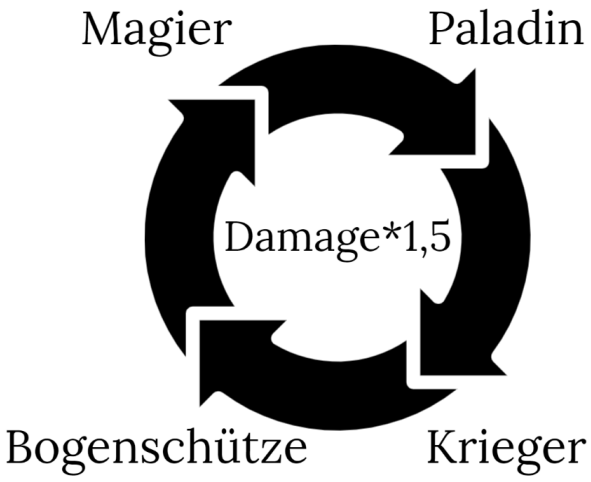
Während des Spiels kann mittels der ESC-Taste, ebenfalls das Menü geöffnet werden um das Spiel vorzeitig beenden zu können.

3.2. Mechaniken

Im Spiel sind 4 verschiedene Figuren implementiert. Jede dieser Figuren besitzt unterschiedliche Werte. In der folgenden Tabelle sind diese dargestellt.

Figurentyp	Anzahl	Leben	Schaden	Angriffsreichweite	Bewegungsreichweite
Magier	1	Niedrig (30)	Hoch (20)	Mittel (3)	Niedrig (1)
Bogenschütze	2	Mittel (50)	Niedrig (10)	Mittel (3)	Hoch (3)
Paladin	1	Sehr hoch (150)	Niedrig (10)	Niedrig (1)	Niedrig (1)
Krieger	2	Hoch (85)	Hoch (20)	Niedrig (1)	Mittel (2)

Darüber hinaus ist eine Figur immer gegen eine andere Figur effektiv. Im folgenden zeigt eine Grafik, wie die Figuren in Zusammenhang stehen.

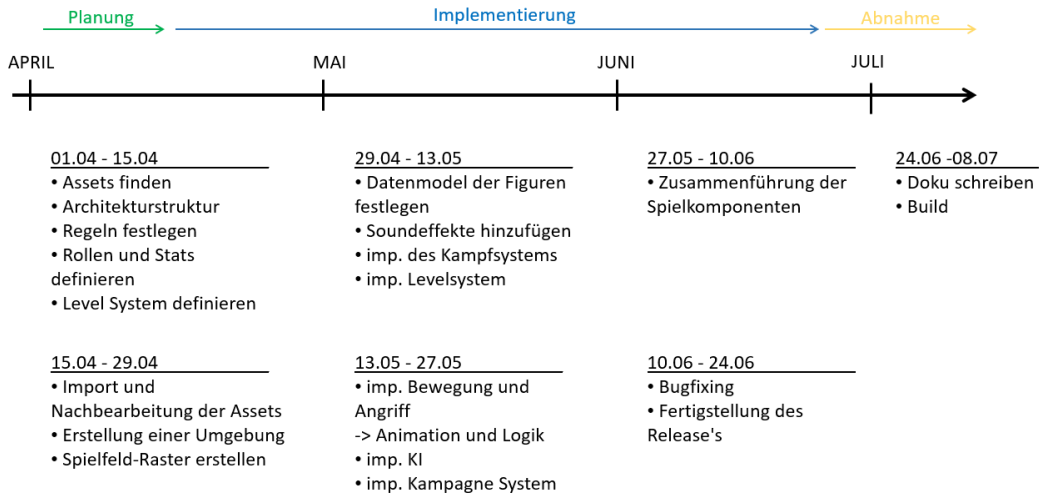


Durch die beiden genannten Mechaniken erhält das Spiel eine strategische Tiefe, wodurch der Benutzer gefordert wird.

4. Game Production

Nach erfolgreicher Festlegung der Idee und des Game Designs werden die Meilensteine des Projektes festgelegt. Um die Zeitplanung zu erleichtern, erfolgt die Formulierung von konkreten Zielen.

Die zeitlichen Aufteilung dieser Ziele ist aus der folgenden Grafik, eingegliedert in zwei-Wochen-Sprints, zu entnehmen.



Für das Projektmanagement wird das Aufgaben-Verwaltung-Tool Trello verwendet. Dieses bietet einen Überblick des aktuellen Entwicklungsstands.

Im Verlauf der Implementierungphase erfolgten Änderungen am Zeitplan. Der Grund dafür war meist programmiertechnisch vom aktuellen Stand sinnvoller.

Zum Beispiel wird nach importieren und nachbearbeiten der Assets, die Zusammenführung der Spielkomponenten vorgezogen um erste funktionale Tests durchzuführen.

Während der Projektdurchführung ist die Entscheidung gefallen, sich mehr auf das Kampfsystem des Spiels zu konzentrieren und dieses weiter auszubauen. Dies liegt darin begründet, dass durch den größeren Fokus auf das Kampfsystem der Spielspaß gesteigert werden kann, indem z.B. mehr Möglichkeiten dem Spieler eingeräumt werden.

Darüber hinaus sind wir durch diese Entscheidung zeitlich flexibler. Dies ist hierbei ein wichtiger Punkt, da aufgrund des verlassen eines Gruppenmitglieds zum Zeitpunkt des Zwischenpräsentation, an die verbleibenden Gruppenmitglieder geisterte Anforderungen anfallen.

5. Game Programming

5.1. Verwendete Technologien

Das Spiel wurde mit der Spiele-Engine Unity implementiert, unter der Verwendung der Programmiersprache C#.

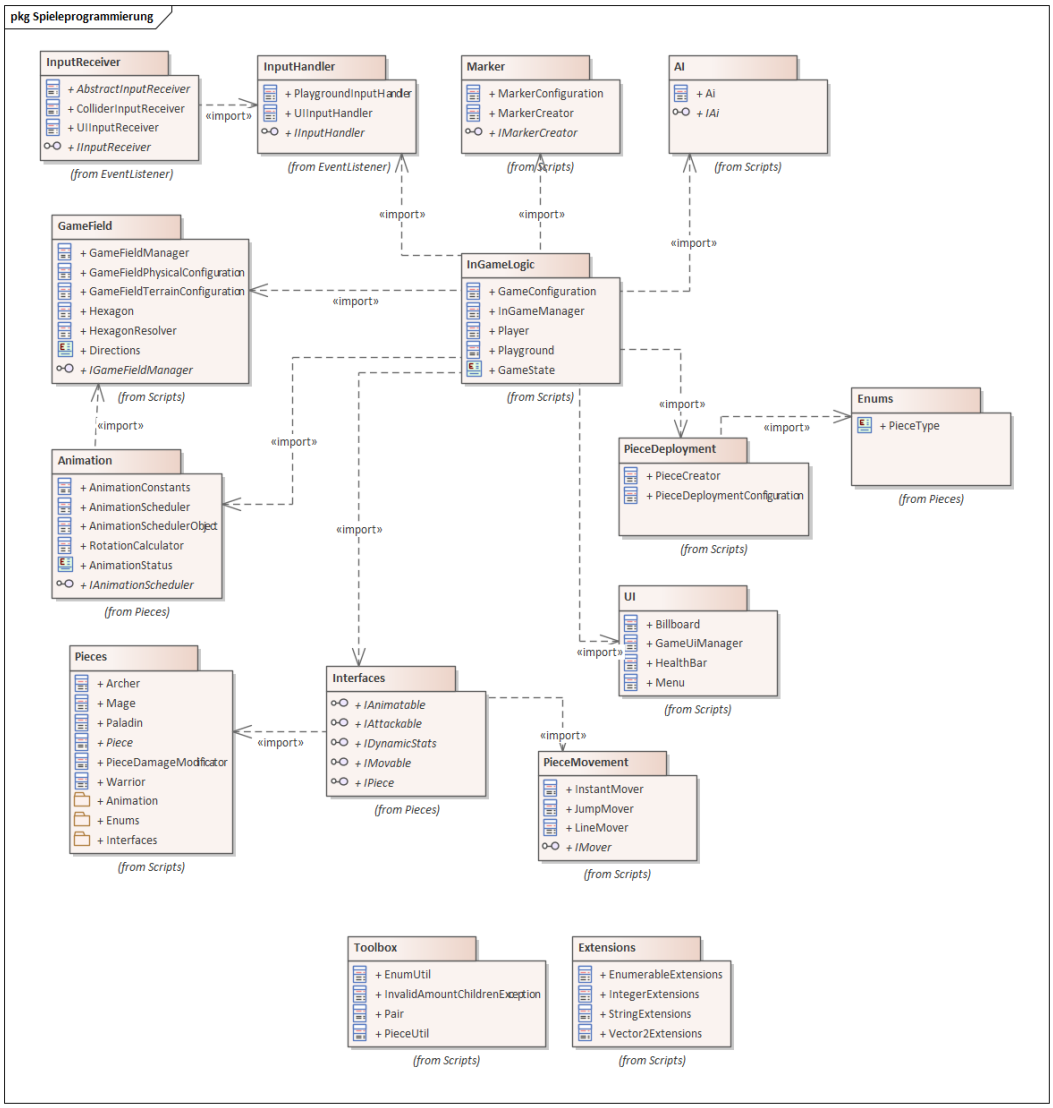
Für die Bearbeitung der Audiospuren wurde die Videoschnittsoftware Adobe Premiere Pro verwendet.

5.2. Spielelogik

5.2.1. Aufbau der Anwendung

Die Implementierung der Anwendung auf logischer Ebene erfolgt überwiegend mittels objektorientierter Programmierung. Mittels dem in Unity angebotenen Insepctor und Monobehaviour-Klassen werden einige Klassen als sogenannte Skripte mit GameObjects, welche innerhalb der Szene liegen, verknüpft.

Der objektorientierte Anteil der Anwendung folgt hierbei den gängigen Architektur- und Designprinizpien. In der folgenden Grafik zeigt das Package-Diagram den Aufbau der domain-driven Architektur.



5.2.2. Ablauf der Anwendung

Der prinzipielle Ablauf der Anwendung wurde bereits im Punkt Game Design beschrieben.

Im folgenden wird auf logischen Ablauf der Anwendung in Bezug auf die Implementierung eingegangen.

Der Kern der Anwendung ist das Paket InGameLogik. Dieses steuert mittels Zugriff auf weitere Module den wesentlichen Ablauf des Spiels.

Zu Beginn des Spiels wird die Anwendung mittels sogenannten ScriptableObjects initialisiert. Diese Objekte beinhalten z.B. Informationen über den physikalischen Aufbau des Spielfeldes, die Art der verfügbaren Animationen oder Referenzen zu Prefabs, welche zur Laufzeit instantiiert werden.

Im Anschluss daran werden auf Basis der übergebenen Informationen Figuren erstellt und weitere Klassen instanziert. Darauf folgend wird die Spielroutine gestartet. Diese regelt den generellen Spielablauf. Im Wesentlichen besteht diese aus einer Methode.

Diese Methode ist im folgenden vereinfacht als Pseudo-Code dargestellt.

```
1  Function NextTurn(ActivePlayer)
2
3      While (True)
4          If (ActivePlayer.Moves = Nothing) Then
5
6              ActivePlayer = ChangeActiveTeam()
7              PrepareNextTurn()
8          End If
9
10         NewPiece =
               GetNextPieceOfPlayerWhichDidNotMove(ActivePlayer)
11
12         If (ActivePlayer = RealPlayer) Then
13
14             AvailableMoves = SelectPiece(NewPiece)
15
16             //Player can select hexagon on leave
17             //After hexagon selection NextTurn will be called
18             If(AvailableMoves Not Nothing) Return
19
20             //Skip piece, as it has no moves
21             Continue
22
23         Else
24             NewAIMove = AI.GetNextMove()
25
26             //Skip pice, as it has no moves
27             If (NewAIMove = Nothing) Continue
28
29             //Calls AI moves, which will
30             //call NextTurn on End
31             ScheduleAiMove()
32             Return
33         End If
34
35     End While
36 End Function
```

Die gezeigte Methode wird hierbei von Event-Handlern innerhalb des Packages InputHandler aufgerufen, welche wiederum durch InputReceiver aufgerufen werden. Diese sind direkt an das Szenen-Elemente geknüpft und reagieren auf Benutzerinterkationen.

Die Implementierung der Bewegung oder eines Angriffs einer Figur liegt hierbei innerhalb weiterer Module.

5.2.3. Animationen

Animationskontoller

Für die Ausführung der Animationen ist jedem Figurenprefab die Komponente Animator hinzugefügt worden. Außerdem besitzt jede der vier Klassen Archer, Mage, Paladin und Warrior einen eigenen Animationscontroller. Jede Klasse hat dabei fünf Animationen:

- Idle
- Bewegung
- Angriff
- Getroffen
- Sterben

Es ist im Controller festgelegt, dass sich jede Figur zu Beginn im Idle-Zustand befindet und in diesem, nach Ausführung einer anderen Animation, zurückkehrt. Die Ausnahme bildet hier die Sterbeanimation.

Animationsausführung

Die Zustände werden durch Trigger ausgelöst, die in der Klasse 'Piece' gesetzt werden. Nach Ausführung der Animation kehrt die Figur in den Idle-Zustand zurück, da die Verknüpfung dafür im Animationscontroller vorliegt. Bei der Bewegungsanimation gilt die folgende Besonderheit. Der Start und das Ende der Animation muss durch Trigger kontrolliert werden. Das hat den Vorteil, das sich die Bewegungsanimation erst beendet, wenn die Figur ihr Zielfeld erreicht hat.

Durch die implementierte Softwarekomponente 'AnimationScheduler' können die Animationen mit einem Zeitverzug ausgeführt werden.

Bevor eine Angriffssituation starten kann, werden die zwei beteiligten Figuren zueinander gedreht. Der richtige Drehwinkel wird dabei von der Klasse 'Rotations-Calculator' berechnet. Nach der Drehung führt der Angreifer seine Animation aus. Wenn der Angriff die Lebensleiste der anderen Figur auf 0 gebracht hat, führt diese ihre Sterbeanimation aus. Ist dies nicht der Fall so wird die Getroffenanimation ausgeführt. Nach Vollendung aller beteiligten Animationen wird die Figur in ihre ursprüngliche Blickrichtung zurückrotiert. Diese ist abhängig von der jeweiligen Teamzugehörigkeit.

Die Rotation findet ebenfalls vor und nach der Bewegunsanimation einer Figur statt. Dabei wird die Spielfigur in Richtung des Zielfeldes gedreht.

Alle Figuren im Spiel haben außerdem zu jeder der fünf Animationen eine Audiosource. Sobald eine Animation getriggert ist, wird ein entsprechender Sound abgespielt.

5.2.4. Menü

Das Menü besteht aus drei Komponenten, dem Titelbildschirm, dem In-Spiel Menü und einem Endbildschirm. Der Titelbildschirm ermöglicht den Start eines neuen Spiels und das Beenden der Anwendung.

Das In-Spiel Menü kann innerhalb des Spiels über die ESC-Taste aufgerufen werden. Es erlaubt dem Spieler ein zurückgehen auf den Titelbildschirm, von dem aus ein Spiel neugestartet werden kann. Des Weiteren kann ein Spieler das In-Spiel Menü auch wieder über 'Continue' verlassen.

Der Endbildschirm wird bei Abschluss einer Partie eingeblendet und lässt den Spieler zurück auf den Titelbildschirm wechseln.

5.2.5. KI

Die KI berechnet ihre Züge weitgehend auf Basis der 'Piece' Methoden 'GeneratePossibleAttackMovements' und 'GeneratePossibleMoveMovements'. Basierend auf den so berechneten möglichen Spielzügen trifft die KI ihre Entscheidungen.

Prinzipiell werden immer zuerst Angriffsmöglichkeiten priorisiert, dabei erhöht sich die Priorität eines Ziels abhängig von ihrem aktuellen Leben und Angriffskraft. Da es bei Angriffen sowohl wichtig ist, Ziele auszuschalten, als auch gefährliche Ziele früher zu eliminieren, gilt: Je niedriger das Leben und höher die Angriffskraft, desto wahrscheinlicher ein Angriff.

Bei möglichen Bewegungsfeldern priorisiert die KI abhängig von der Entfernung zur nächsten gegnerischen Figur. Sie versucht bei der Auswahl möglichst auf ihre maximale Angriffsdistanz zu einem Ziel zu kommen, um eine Bedrohung für gegnerische Figuren darzustellen, gleichzeitig aber möglichst nicht in unnötige Gefahr zu geraten.

6. Game Art

Da das Spiel die Spieler überwiegend auf strategische Art herausfordert, ist es wichtig, das Informationen, welche der Benutzer zum Treffen strategischer Entscheidungen benötigt, möglichst zugänglich vermittelt werden.

Stilistisch entschieden wir uns aus diesem Grund zum einen dafür Pixelart für Hintergrundelemente und Bilder einzusetzen. Zum andern Cel Shade, einem 3D comichaften Look, für die Figuren.

Aus diesem Ansatz heraus wurden die folgenden Designentscheidungen getroffen.

6.1. Größe und Aufbau des Spielfeldes

Das Spielfeld wurde mittels einer Unity Tilemap gebaut, um eine klare Hexagonstruktur zu erhalten.

Die Auswahl der Felder liegt der logischen und visuellen Abgrenzung zugrunde. Die abstrahierten Grafiken dienen dazu den Fokus des Spielers auf die wichtigen Elemente im Spiel zu lenken. Aus diesem Grund sind die einzelnen Hexagonfeldtypen von der Darstellung leicht voneinander zu unterscheiden. So bilden sich grafisch 'natürliche' Trennlinen zwischen den verschiedenen Typen und dem Spieler ist klar wo zum Beispiel Bergfelder beginnen und enden.

Logisch ist auch, dass Truppen auf Berg-/ und Wasserfeldern nicht rasten können, weshalb sie sich für blockierende Felder gut eignen.

6.2. Teamzugehörigkeit einer Figur

Figuren sind nach Team in männlich und weiblich unterteilt.

Zusätzlich unterscheiden sich UI Elemente wie die Lebensleiste und der Zugindikator farblich. Im eigenen Team wird grün für den Lebensbalken und Orange für den Zugindikator verwendet. Im gegnerischen Team blau und gelb.

6.3. Klassenzugehörigkeit einer Figur

Bei den Figuren unterstützt der leicht comichafte Look der Figuren eine einfache Unterscheidung zwischen ihnen. Zusätzlich grenzen sie sich durch ihre Waffenauswahl voneinander ab.

Alle Klassen tragen eine traditionell mit dieser Klasse assoziierte Waffe. So trägt zum Beispiel der Paladin ein Schild oder der Bogenschütze einen Bogen. Der Magier ist davon ausgenommen.

6.4. Welche Zugmöglichkeiten bestehen

Durch blaue UI Indikatoren werden zum Zugzeitpunkt jeder Figur alle Felder markiert, die begehbar sind. Gleiches gilt für Felder, auf die eine Angriffsmöglichkeit besteht. Diese werden mit rot markiert.

6.5. Welche Figur ist am Zug

Ein Zugindikator markiert die Figur die am Zug ist. Dieser ist im eigenen Team in oranger Farbe dargestellt und im Gegnerischen in gelber Farbe.

7. Inbetriebnahme

Zum Abschluss der Entwicklung wurde im Unity Editor eine unabhängig ausführbare Version des Spiels gebaut. Diese besteht aus einem Ordner. In diesem befindet sich eine ausführbare Spieldatei (.exe), ein Data Ordner mit den verwendeten Ressourcen, ein Ordner namens 'MonoBleedingEdge', eine CrashHandler.exe und eine .dll File. Damit das Spiel ausgeführt werden kann, müssen alle Dateien vorhanden sein und dürfen nicht verändert werden.

Um das Spiel zu starten, muss die Datei 'Des Kriegers Erinnerung.exe' innerhalb des Ordners 'Game' ausgeführt werden.

Alle Kernsysteme des Spiels sind in der ausführbaren Datei erfolgreich umgesetzt worden. Das Kampfsystem ist vollumfänglich umgesetzt und spielbar. Aufgrund der zeitlichen Einschränkung und dem unerwarteten Austreten eines Teammitgliedes, mussten die Features Storykampagne und Levelsystem gestrichen werden.

8. Verwendete Assets

- (20.06.2022) Hintergrund vom Spielfeld von https://cdn.wallpapersafari.com/51/57/SLqNDl.jpg, im Wasserzeichen ist psdgraphics.com vermerkt
- (20.06.2022) Titelscreen Bild von https://art.ngfiles.com/images/520000/520185_marccobley_pixel-mountains.png?f1524749198
- (20.06.2022) Endscreen Bild https://i.redd.it/vqhezs2arkt21.png
- (20.06.2022) Healthbar Sprites und Idee von https://github.com/Brackeys/Health-Bar/tree/master/Health%20Bar/Assets/Sprites
- (20.06.2022) Hintergrundmusik von https://www.youtube.com/watch?v=tf5qwoCE6eY&list=PLCFkL_mbfwe3bx6Eim5VMIauSlq88T1Pf&index=31
- (20.06.2022) Charaktere gekauft von https://assetstore.unity.com/packages/3d/characters/humanoids/all-star-character-collection-43181
- (20.06.2022) Hexagonuntergrund von https://assetstore.unity.com/packages/2d/environments/hex-basic-set-painted-2d-terrain-52258
- (20.06.2022) Folgenede Grundsounds von https://freesound.org/ , wurden für die Verwendung noch bearbeitet
  - 'Female Battle Cries v1 wav.wav' by Volonda
  - 'Dramatic Sound #1' by AUDACITIER
  - 'Sword Clash and Slide.MP3' by FunWithSound
  - 'Teleport High' by GameAudio

9. Ausblick

9.1. Erweiterbarkeit der Ergebnisse

Auf der bestehenden Grundlage lässt sich das Spiel gut erweitern.

Wie bereits ursprünglich geplant, kann das Spiel um eine Storykampagne erweitert werden. Diese würde die Form einer Übersichtsszene mit verschiedenen Levels, die in Reihenfolge abgeschlossen werden müssen, annehmen. Die einzelnen Levels können unterschiedliche Karten, Aufstellungen und KI-Stärken enthalten, um den Spieler auf verschiedene Weisen zu fordern. Als Belohnung könnten dem Spieler nach Abschluss kurze Geschichtssegmente in Form eines Textes und Erfahrungsgewinn der Figuren winken.

In einem zusätzlichen Personalisierungssystem können den Figuren neben einem Namen nach genügendem Erfahrungserhalt auch Level verteilt werden. So könnte jeder Spieler seine Figuren in individueller Weise seinem Spielstil anpassen. Eine weitere Idee ist es, bereits bestehende Systeme des Spiels zu erweitern. So könnte der Vorteilskreis neben einer Schadensverstärkung in Drehrichtung auch eine Schadensminimierung in die entgegengesetzte Richtung verursachen, um einen klugen Spielstil weiter zu fördern.

Des Weiteren kann der Hexagonuntergrund weiter ins Spielerlebnis integriert werden. Nicht nur Berge und Seen würden sich dann auf eine Karte auswirken, sondern auch andere Felder könnten Vor-/ oder Nachteile beim betreten von diesen Feldern bewirken.

9.2. Übertragbarkeit der Ergebnisse

Einige der Konzepte des Spiels lassen sich auf zukünftige Projekte übertragen.

So besteht zum Beispiel das Menüsystem aus einem Spielobjekt und einem Skript, welches nur über feste Schnittstellen wie dem Menüaufruf mit dem restlichen Spiel verbunden ist. Es ist vorstellbar, dass wir in Zukunft dieses Menüfundament auch in anderen Spielen angepasst nutzen werden.

Ein weiteres übertragbares Konzept, ist die mathematische Grundlage für Bewegungen auf einem hexagonalen Feld.

Das Konzept der dynamischen Objekterstellung während der Spiellaufzeit ist wichtig und übertragbar. So ist es zwar in einem Strategiespiel mit fester Aufstellung und Feld vorstellbar, dass Figuren im Voraus in der Szene platziert werden. Es ist aber für die Entwicklung nicht praktisch und begrenzt mögliche Spielmechaniken, da dynamisches erzeugen von Figuren verhindert wird.

Als letztes Beispiel für ein übertragbares Konzept aus dieser Spielentwicklung, ist der Animation Scheduler, den wir im Verlauf des Semesters entwickelt haben. Er ermöglicht es eine beliebige Anzahl an Aktionen verzögert durchzuführen. So können zum Beispiel Animationen oder Rotationen mit zeitlicher Verzögerung asynchron durchgeführt werden.

A. Anhang Entwickelte Skripte

Hochauflösende und vollständige Abbildung der Skripte und Abhängigkeiten dieser zueinander.