

Maturitätsarbeit 2023
Documentation

Laneclash

2022



Version: 1.0
Date: 2022-10-21 14:07:29+02:00

Team: Marc Honegger (C6a)
Elia Schürpf (C6a)

Begleitperson: Martin Hunziker

Gymnasium
KZO - Kantonsschule Zürcher Oberland

Inhaltsverzeichnis

I Ziel	1
1 Vision	2
2 Anforderungen	3
2.1 KZO	3
2.2 Unsere eigenen Anforderungen	3
2.2.1 Notwendig	4
2.2.2 Anvisiert	5
2.2.3 Möglichkeiten	6
2.2.4 Bereits früh verworfen	7
3 Skizzen	9
3.1 Erste Skizzen	9
3.2 Erstes Mindmap	10
4 Risiko	11
4.1 Hoch	11
4.2 Mässig	12
4.3 Tief	12
II Produkt	14
5 Aktuelle Features	15
5.1 KZO	15
5.2 Unsere eigenen Anforderungen	15
5.2.1 Notwendig	15
5.2.2 Anvisiert	20
5.2.3 Möglichkeiten	21
5.2.4 Fürs Erste verworfen	21
6 Architektur	24

7 Zukunft	25
7.1 Ein weisses Blatt	25
7.2 Alpha- und Closed Beta-Testing	25
7.3 Veröffentlichung	26
III Organisation	27
8 Technologien	28
8.1 Unity	28
8.2 Github	29
8.3 Jetbrains Rider	29
8.4 Jira	29
8.5 LaTeX	29
8.6 Stable-Diffusion	30
8.7 Visual Studio Code	30
8.8 Gource	30
9 Team Management	31
9.1 Anfänge	31
9.2 Betreuungsperson	31
9.3 Kommunikation	32
10 Time Management	33
10.1 Roadmap	33
10.2 Time-Tracking	34
IV Reflexion	35
11 Team	36
11.1 Rückblick	36
11.2 Nächstes Mal	36
12 Elia	37
12.1 Rückblick	37
12.2 Nächstes Mal	37
13 Marc	38
13.1 Rückblick	38
13.2 Nächstes Mal	38

V Glossar **39**

Glossar **40**

Teil I

Ziel

Kapitel 1

Vision

Ein funktionsfähiges Videospiel, welches jede vergleichbare Maturitätsarbeit in den Schatten stellt. Die Ideen sind nicht neu, sondern waren schon vor Jahren in Marcs Kopf. Jedoch war für die Umsetzung ein Team notwendig. Uns war dennoch von Anfang an bewusst, dass wir auch zu zweit nicht alle Funktionen hinkriegen werden, dennoch haben wir uns ein hohes Ziel gesetzt und für die Betreuung uns für Martin Hunziker entschieden. Vieles konnten wir schlussendlich auch umsetzen. Das grösste Wagnis war der Multiplayer und das haben wir dann auch an eigenem Leib erfahren. Alleine für den Multiplayer wurden über 100h investiert. Alles in allem kam ein funktionsfähiges und lustiges Spiel heraus. Einige Kommentare:

'Sick Game' - Marc Honegger
'Absolut Krass' - Elia Schürpf
'Masterpiece' - Martin Hunziker

Wie das Spiel jetzt aussieht, wie es zu diesem Punkt kommen konnte, aber auch wie es in Zukunft damit weiter geht, ist in dieser Arbeit beschrieben.

Kapitel 2

Anforderungen

2.1 KZO

Die Anforderungen der Schule halten sich in Grenzen. Zum einen gibt es fast keine genauen Anforderungen und die Anforderungen, die es gibt, sind relativ tief gehalten. Die drei Anforderungen der KZO sind:

1. Zeit

Geplant ist ungefähr eine Lektion pro Woche, denn diese wir dem Stundenplan für das Semester abgezogen. Jedoch ist diese Anforderung bei uns nicht relevant, denn unser Projekt ist eine riesige Maturitätsarbeit. Wir werden ohne Probleme diese Zeitanforderung erreichen und auch bei Weitem überschreiten.

2. Begleitperson

Jede Maturitätsarbeit braucht eine Begleitperson. Also eine Lehrperson, welche die Arbeit bewertet und einen unterstützt. Letzteres, also Unterstützung, war bei uns sehr schwierig, denn keine Lehrperson kennt sich mit Unity und C# genug gut aus. Wir sind bereits früh auf Herrn Kern, ein Informatiklehrer, zugegangen. Er hat uns an Martin Hunziker, Leiter der IT, weitergeleitet. Dieser nahm uns sehr gerne auf, mit der Anmerkung, dass unsere Arbeit sehr schwierig sein wird, umzusetzen.

3. Plagiat

Arbeiten dürfen nicht einfach kopiert werden. Wir haben uns zwar bei gewissen Problemen online inspirieren lassen, jedoch sind diese keine Plagiate. Der grösste Teil unseres Codes und alles unserer schriftlichen Arbeit sind von uns geschrieben.

2.2 Unsere eigenen Anforderungen

Gewisse Features sind für das Funktionieren des Spiels wichtiger als andere. Zum Beispiel sind Einstellungen, Truppen und Multiplayer notwendiger als die Monetarisierung, zumindest nach unserer Ansicht. Deshalb ist dieser Abschnitt in 4 Gruppen unterteilt:

1. **Notwendig:** Ohne diese Features läuft das Spiel nicht oder ist sehr mangelhaft, voller Fehler und unspielbar. Ohne diese Features ist Idee des Spieles nicht erkennbar.
2. **Anvisiert:** Diese Features sind unser Ziel. Sie wurden Herr Hunziker, also der Begleitperson, angekündigt und sind alle in einem Plan festgehalten. Diese Ziele sollen dazu führen, dass das Spiel gut spielbar ist und es Spass macht. Sie sind auch dafür essenziell, das Spiel nicht zu monoton zu gestalten und sollten alle bis zum Abgabetermin der schriftlichen Arbeit vollendet sein.
3. **Möglichkeiten:** Möglichkeiten das Spiel noch auf die Spitze zu bringen. Keine dieser Anforderungen ist nötig für das Spielen, jedoch können sie das Spielerlebnis spannender und ausgereifter machen. Diese Ziele sind entweder in den Sternen geschrieben oder zu erreichen bis zur mündlichen Präsentation. Vereinzelt können diese, wie die anvisierten Anforderungen, noch bis zur schriftlichen Abgabe erreicht werden.
4. **Verworfen:** Weitere Ideen, welche wir hatten, jedoch als unmöglich oder nicht sinnvoll erachteten.

2.2.1 Notwendig

- **Systemanforderungen**

Unser Spiel sollte auf den meisten modernen Geräten funktionieren. Es soll auf macOS und Windows fliessend gespielt werden können. Also mit mindestens 60FPS.

- **Benutzeroberfläche**

Eine Startseite mit einem "Spielen"-Knopf, einem "Einstellungen"-Knopf, einem "Credits"-Knopf und einem "Verlassen"-Knopf. Sie soll intuitiv sein und einigermassen schön aussehen. Die Benutzeroberfläche soll mit Touch bedienbar sein, andere Eingabemöglichkeiten werden nicht unterstützt.

- **Lokaler Multiplayer**

Das Spiel soll im lokalen Netz gespielt werden können. So zum Beispiel mit Freunden oder Familie. Es sollte mithilfe der IP-Adresse innerhalb des gleichen Netzes zusammengespielt werden können.

- **Einstellungen**

Auflösung, Vollbild und Lautstärke müssen eingestellt werden können.

- **Kamera**

Die Kamera ist beweglich und das ganze Schlachtfeld ist sichtbar.

- **Truppen**

Es braucht Truppen, die für einen kämpfen können. Diese sind notwendig für den Verlauf des Spieles und ohne sie hat das Spiel keinen Sinn.

- **Gewinnmöglichkeit**

Eine Chance das Spiel zu gewinnen oder verlieren und es somit zu beenden. Dies kann sehr simpel mit einer Linie vollendet werden, welche beim Überschreiten das Spiel beendet.

- **Lanes**

Unser Spiel ist zwar 2.5D, aber hat dennoch eine Tiefe.

- **Synchronisation**

Unser Spiel soll in Echtzeit spielbar sein, deswegen müssen die beiden Spieler Synchronisiert das Gleiche sehen. Eine De-Synchronisation wäre verheerend.

- **Crossplay** Die Möglichkeit sich mit einem Windows Rechner und einem macOS Rechner zu verbinden und zusammenzuspielen.

2.2.2 Anvisiert

- **Helden**

Beide Spieler haben eine Art Truppe, welche ihre Siegeslinie beschützt. Sie haben einen Racheeffekt, welcher die Lane ausradiert, damit das Spiel nicht sofort vorbei ist.

- **Design**

Das Spiel sollte vom Aussehen her was hergeben. Es sollte nicht wie ein Prototyp, sondern wie ein vollendetes Spiel aussehen.

- **Deck**

Zu Beginn Spieler könne ihr eigenes Deck aus einer Auswahl von Karten zusammenstellen. Im Spiel werden davon per Zufall Karten gezogen.

- **Bot**

Ein sehr simpler Algorithmus um alleine gegen den Computer zu spielen auf den Schwierigkeitsstufen '*Einfach*', '*Mittel*', '*Schwierig*'. Es sollen rein zufällig Truppen geschickt werden, bei höherer Schwierigkeit mehr.

- **Truppen**

- **Nahkampf** Eine Truppe mit wenig Reichweite.

- **Fernkampf** Eine Truppe die auf Reichweite angreift. Sie schießt Projektive, zum Beispiel ein Bogenschütze, der mit Pfeilen schießt.

- **Suizid** Eine Truppe die bei Berührung mit einem Gegner stirbt und einen Effekt auslöst.

- **Effekte**

- **Gift:** Zeitlich limitierter und wiederholender Schadenseffekt. Eine visuelle Markierung soll vorhanden sein und das gleiche Gift, also der gleichen Truppe, soll nur einmal auf jemanden sein.

- **Dornen:** Truppen, welche Charaktere mit Dornen attackieren, erhalten schaden. Soll nach Auswahl auf Nahkämpfer, Fernkämpfer oder beides wirken.
- **Rache:** Truppen mit Rache haben einen Effekt nach ihrem Tod. Zum Beispiel eine Truppe beschwören oder Schaden verursachen.

2.2.3 Möglichkeiten

- **Zauber**

Als Alternative sollen nicht alle Karten einfach eine Truppe herbeirufen. Gewisse Karten sollten nur einen Effekt auslösen. Beispiele für Zauber:

- ”Heile deine Helden um 5 leben”
- ”Gib deinen Helden 5 Rüstung”
- ”Verursache allen gegnerischen Truppen 5 Schaden”
- ”Ziehe 3 Karten”

- **Monetarisierung**

Hierfür sind uns drei Möglichkeiten eingefallen, hier jeweils die Vor- und Nachteile:

1. **Kaufbare Gegenstände**

- + Vielseitige und Nachhaltige Monetarisierung. Neue Features, bedeuten auch neue Geldmöglichkeit. Die Inhalte sollten auch erspielbar sein, somit kann bezahlen zwar zu Vorteilen führen, jedoch mit viel Spielen trotzdem erreichbar sein.
- Pay-to-Win

2. **Skins**

- + Weit verbreitet und sehr beliebt bei Spielern. Gibt keinem Spieler Vorteile
- Wenig Nutzen und sehr aufwändig. Neue Designs zu erstellen, wäre bei uns nicht so sinnvoll. Wir haben noch sehr viel Features zu implementieren und sind nicht Designer. Für uns braucht es sehr viel Zeit eine brauchbare Darstellung zu erstellen und der Mehrwert, welcher dem Spiel beigetragen wird, ist marginal.

3. **Kostenpflichtiges Spiel**

- + Einmalige Bezahlung, was für viele Nutzer lukrativer ist. Jedoch gilt dies vor allem bei Singleplayer Spielen.
- Wir nehmen an, dass wenige Leute bereit wären, Geld für unser Spiel zu bezahlen. Dafür wird es nicht genug ausgereift sein. Auch ist diese Methodik nicht nachhaltig und führt nur zu einer einmaligen Geldspritze. Viele grössere Spiele führen deshalb später DLCs ein, um das Spiel zu erweitern. Jedoch ist dies bei einem Multiplayer Spiel Pay-to-Win. Abschliessen müssten wir höchstwahrscheinlich selbst Geld vorauswerfen, um unser Spiel anbieten zu können, z.B. auf Steam.

- **Helden**

Eine Auswahl von Helden, mit unterschiedlichem Schaden, Leben und Effekten, würde das Spiel nochmals spannender machen. Auch sollten gewisse Truppen auf bestimmte Helden limitiert sein.

- **Design**

Wenn die Zeit vorhanden ist, kann das Design immer verbessert werden. Hier geht es aber um den Feinschliff, z.B. mehr Hintergründe, und mehr Dinge selbst zu designen.

- **Tutorial**

Eine Anleitung für Anfänger wäre sehr schön. Sie soll neuen Spielern das Anfangen erleichtern. Es sollte aber auch überspringbar sein, damit alte Spieler, es nicht nochmals spielen müssen. Es soll nicht lange sein und nicht schwierig. Dennoch soll es alle Mechaniken des Spiels erklären, am besten Anhand von Gameplay.

- **Truppen** Weitere Truppen können jederzeit erstellt werden. Hier ist kein Limit gesetzt. Leben, Schaden, Darstellung, Effekt und vieles mehr kann angepasst werden.

- **Effekte**

- **Wiederbelebung:** Die Truppe wird wiederbelebt, sofort an Ort und Stelle oder mit einer Verzögerung am Startpunkt.
- **Rüstung:** Die Truppe hat zusätzlich zu den Leben auch Rüstung. Die Rüstung wird zuerst abgezogen und hat im Gegensatz zum Leben keine Lü mite.
- **Aura:** Die Truppe fügt gegnerischen Truppen in einem gewissen Radius permanent Schaden zu.

- **Mehrsprachig**

Das Spiel soll in Englisch, Deutsch und Französisch spielbar sein.

- **Ingame-Chatfenster**

Ein Textfeld, in dem man sich mit dem Gegner unterhalten kann.

2.2.4 Bereits früh verworfen

- **Online Multiplayer**

Besser als ein Multiplayer, der limitiert auf dasselbe Netz ist, ist ein Multiplayer, der weltweit verfügbar ist. Jedoch ist von einem Computer auf einen anderen zu verbinden dank der Firewall von Router und Computer, nahezu unmöglich. Es würden sich viele weitere Probleme ergeben, wie zum Beispiel Port-Forwarding. Dies ist einer der Gründe, weshalb ein Server sehr praktisch ist. Mit diesem wäre dieses Problem gelöst. Server sind aber teuer und aufwändig.

- **Shop**

Nicht alle Karten sind von Beginn an freigeben, sondern müssen freigespielt werden. So kann zum Beispiel eine Ingame Währung erspielt werden und damit im Shop Karten oder sonstige Dinge gekauft werden. Der Shop kann mit Zahlungsmethoden ausgestattet werden und so zur eventuellen Monetarisierung beitragen.

- **Kampagne**

Singleplayer gegen bestimmte vorprogrammierte Gegner. Sie sollen immer schwieriger werden und bestimmte Herausforderungen mit sich bringen. Bei Vollendung werden Belohnungen verteilt.

- **Anti-Cheat**

Eine sehr komplexe Software um das Schummeln zu verhindern. Ist in allen grösseren Spielen vorhanden. Teilweise sogar auf Systemebene, oder sogar auf Kernel-Level, gespeichert und ausgeführt.

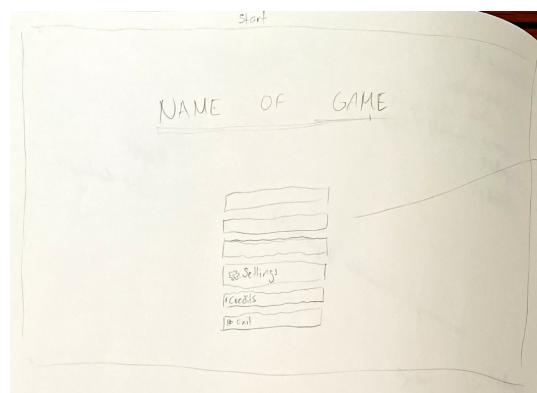
- **Errungenschaften**

Eine Übersicht und die Möglichkeit die Errungenschaften zu erfüllen. Gegebenenfalls Belohnungen verteilen, wie zum Beispiel bestimmte Karten freischalten. Diese Erweiterung ist keines Falls notwendig und ein absolutes nice-to-have Feature.

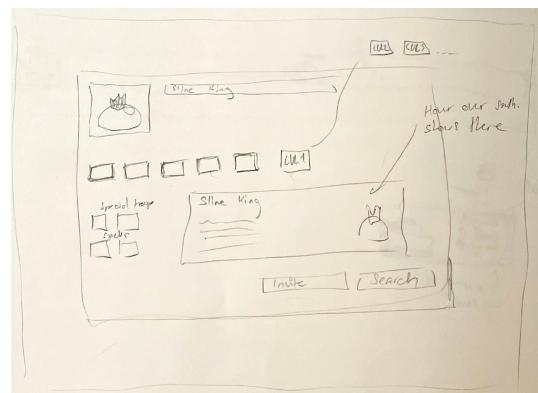
Kapitel 3

Skizzen

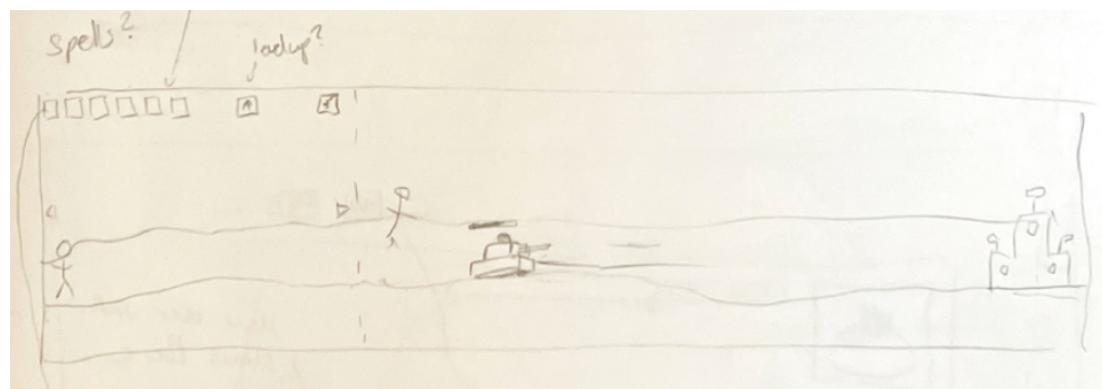
3.1 Erste Skizzen



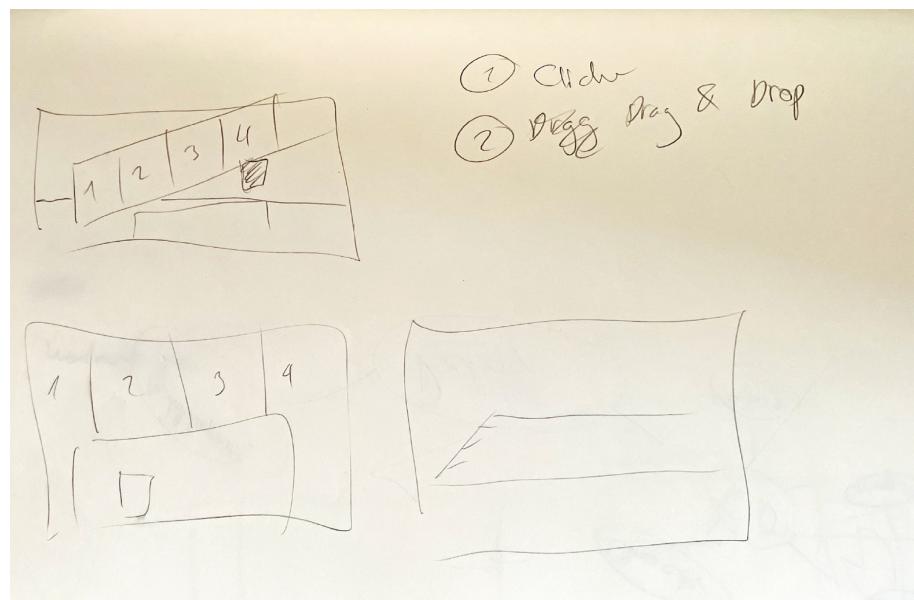
Startseite



Auswählen von Karten und Helden



Spielszene mit UI



Mechanik der Karten / Spawning von Truppen

3.2 Erstes Mindmap



Kapitel 4

Risiko

Wir haben die Risiken eingeteilt in verschiedene Gefahrenstufen. Die Einstufung geschah basierend auf Wahrscheinlichkeit, wie schlimm ist der Worstcase und wie gut kann das Risiko reduziert werden. Jedes Risiko ist dann unterteilt in:

1. Risiko (Wahrscheinlichkeit und Schaden)
 - (a) Reduktionsstrategie: Strategie um die Wahrscheinlichkeit und/oder Schaden zu reduzieren.
 - (b) Reduziertes Risiko: Risiko nach der Umsetzung der Reduktionsstrategie.
 - (c) Tatsächliches Outcome: Das tatsächliche Endresultat

4.1 Hoch

1. Vergange Maturitätsarbeiten, welche ein Videospiel als Ziel hatten, sind gescheitert (Wahrscheinlich und Kritisch)
 - (a) Reduktionsstrategie: Sich des Risikos bewusst sein und bereit sein, viel Zeit und Arbeit zu investieren
 - (b) Reduziertes Risiko: Das Risiko ist reduziert immer noch sehr hoch. Wir haben uns beide an Multiplayer fest gefressen und werden es höchst wahrscheinlich trotzdem umsetzen
 - (c) Tatsächliches Outcome: Wie erwartet ist die Maturaarbeit nach 250h immer noch weit entfernt von vollständig. Dennoch ist das Ziel erreicht von einem funktionierendem Videospiel.
2. Beide Teammitglieder haben keine Ahnung von der Entwicklung einer Multiplayer-Funktion (Wahrscheinlich und Kritisch)
 - (a) Reduktionsstrategie: Andere Spielideen erarbeiten, die ohne Multiplayer funktionieren.

- (b) Reduziertes Risiko: Das Risiko ist reduziert immer noch sehr hoch. Wir haben uns beide an Multiplayer fest gefressen und werden es höchst wahrscheinlich trotzdem umsetzen
 - (c) Tatsächliches Outcome: Multiplayer ist mit Abstand der grösste Zeitfresser. Der Multiplayer alleine hat über 100h Arbeit, also fast ein Drittel unserer Arbeit ausgemacht.
3. Es könnten sich Bugs in der abgabebereiten Version verstecken (Wahrscheinlich und Kritisch)
- (a) Reduktionsstrategie: So viel testen wie nur möglich. Sich außerdem auf die kritischen Bugs konzentrieren.
 - (b) Reduziertes Risiko: Das Risiko ist selbst reduziert sehr hoch. Wir wollen ein perfektes Game abgeben.
 - (c) Tatsächliches Outcome: Wir haben unser Möglichstes gegeben. Wir können nur die Daumen drücken, dass Herr Hunziker keine neuen Bugs findet.
4. Die Version für die Abgabe könnte aus unbekannten Gründen crashen (Unwahrscheinlich und Kritisch)
- (a) Reduktionsstrategie: So viel testen wie nur möglich.
 - (b) Reduziertes Risiko: Das Risiko ist selbst reduziert sehr hoch. Man konnte bereits des Öfteren beobachten, wie sich eine schlechte erste Version auf eine Firma auswirken kann.
 - (c) Tatsächliches Outcome: Wir haben unser Möglichstes gegeben. Wir könnten nur noch die Daumen drücken und hoffen, dass es keine fatalen Fehler gibt.

4.2 Mässig

1. Elia hat noch nie mit Unity gearbeitet (Wahrscheinlich und Unbedenklich)
- (a) Reduktionsstrategie: Elia befasst sich zu Beginn mit Unity
 - (b) Reduziertes Risiko: Weiterhin kann angenommen werden, dass Elia auf gewisse Schwierigkeiten mit Unity stossen wird. Jedoch sollten diese nicht viel Zeit oder Mühe benötigen.
 - (c) Tatsächliches Outcome: Elia konnte alles Nötige zu Beginn erlernen und bei allfälligen Fragen auf das Internet oder Marc zurückgreifen.

4.3 Tief

1. Wir werden selbst nicht mit unserem Spiel zufrieden sein. (Wahrscheinlich und Unbedenklich)

- (a) Reduktionsstrategie: Sich bewusst sein, dass wir nur 2 Schüler sind, die keine bisherigen Erfahrungen hatten. Wir dürfen uns nicht an etwas "festfressen".
- (b) Reduziertes Risiko: Es stellt sich das Risiko, dass wir unsere Zeit verschwenden. Wir müssen teilweise einfach weitermachen.
- (c) Tatsächliches Outcome: Man ist nie richtig fertig mit einem Spiel. Es gibt immer Dinge, die man Verbessern könnte. So könnten auch wir noch tausende weitere Stunden investieren.

Teil II

Produkt

Kapitel 5

Aktuelle Features

Kurz gesagt; wir konnten fast alle unsere Anforderung erfüllen.

5.1 KZO

1. Zeit

Geplant war ungefähr eine Lektion pro Woche. Wir haben die geschätzten 100 Stunden (50 Stunden pro Person) um einen Faktor von drei überschritten. Unser gemessener Zeitaufwand, während aktivem Programmieren, beträgt über 330 Stunden. Hier sind die Stunden, in denen wir uns in der Schule über das Projekt unterhalten haben, gar nicht notiert.

2. Begleitperson

Wir hatten zum Glück kein Problem eine Begleitperson zu finden. Martin Hunziker war gerne bereit unsere Arbeit im Blick zu behalten.

3. Plagiat

Zwar sind einige Grafiken aus dem Internet, jedoch sind alle Nutzungsfrei. Auch sonst haben wir keinen Code geklaut und verwendet.

5.2 Unsere eigenen Anforderungen

5.2.1 Notwendig

- Systemanforderungen

Unser Spiel funktioniert sowohl auf macOS als auch auf Windows. Dies ausserdem mit einer FPS von 300, bis teilweise 500, bei den meisten modernen Geräten

- Benutzeroberfläche

- Startseite



Unsere Startseite hat einen "Spielen"-Knopf, einen "Einstellungen"-Knopf, einen "Credits"-Knopf und einen "Verlassen"-Knopf. Unsere vereinzelten Tester bewerteten die Startseite als intuitiv verständlich. Bedienbar mit der Maus, hatten die Tester keine Probleme sich zurechtzufinden.

– Credits



Es gibt die Möglichkeit die Credits vorzeitig zu Verlassen. Dies geschieht durch das Drücken des Knopfes mit dem Haus als Icon. Die Tester fanden dies auch sehr intuitiv.

– Settings

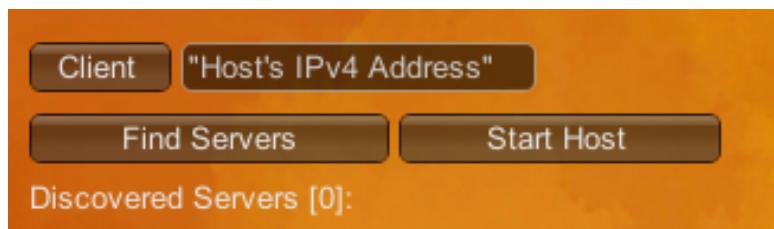


Man kann hier die Auflösung ändern und auswählen, ob das Spiel im Vollbild dargestellt werden soll. Auch gibt es hier die Möglichkeit die Musik lauter, leiser zu stellen oder zu muten. Wenn man Einstellungen geändert hat, muss man das Speichern bestätigen.



Wenn man die Auflösung ändert und dies bestätigt, werden die Änderungen angewendet und man hat zehn Sekunden Zeit erneut zu bestätigen. Sonst werden die Änderungen zurückgesetzt. Dies wurde implementiert, um eventuellen Problemen vorzubeugen.

– Server Management



Nach Interaktion mit dem "Start"-Knopf gibt es die Möglichkeit einen Server zu Hosten oder einer bereits existierenden Lobby als Client beizutreten. Außerdem kann man automatisch nach bereits existierenden Servern suchen.



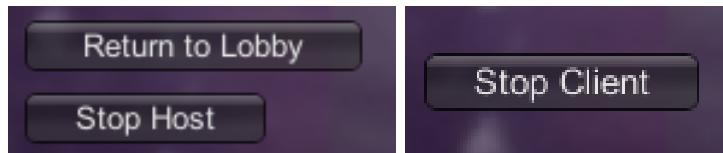
Den gewollten Server kann man mit einem einfachen Linksklick auswählen.

– Lobby Management



Player (1) ist dabei immer der Host, und Player (2) immer der Client. Das eigentliche Spiel beginnt erst, sobald beide Spieler bereit, also "ready", sind. Dieser Status "ready" wird dabei immer aktualisiert dargestellt. Der Host hat außerdem die Möglichkeit den Spieler (2) aus der Lobby zu entfernen. Wir wollten zuerst eine Möglichkeit für einen "Schlüssel" hinzufügen, allerdings stellte sich dies als überflüssig heraus, da alle Spieler unter einem Dach sitzen.

– InGame-UI



Der Host hat im Spiel die Möglichkeit, den Server zurück zur Lobby zu schicken, und das Spiel somit neu zu starten. Er kann auch direkt aufhören zu hosten. Auch der Client hat durchgehend die Möglichkeit den Server zu verlassen.



Im Pausemenu kann man außerdem auf die Schnelle die Musiklautstärke und die Stärke des Schneefalls ändern. Dieses Menu kann mit ESC geöffnet und geschlossen werden.

• Lokaler Multiplayer

Das Spiel kann im lokalen Netz gespielt werden. Zum Testen auch auf demselben Gerät. Als Protokoll haben wir TCP gewählt. Dies, da TCP viel zuverlässiger ist. Folgende vier Punkte waren entscheidend für die Wahl von TCP über UDP:

– Zuverlässigkeit

Keine Probleme mit verloren gegangenen Paketen. Ging ein Paket verloren, sendet TCP es einfach erneut. Entweder werden alle Daten erfolgreich übermittelt, oder man bekommt einen Error.

– Sequenziert

TCP stellt sicher, dass jede Nachricht in der gleichen Reihenfolge eintrifft, in der sie gesendet wurde. Wenn man "a" dann "b" sendet, erhält man auf der anderen Seite garantiert auch zuerst "a" dann "b".

– Verbindungsorientiert

TCP hat sich auf das Konzept einer Verbindung konzentriert. Eine Verbindung bleibt so lange offen, bis entweder der Client oder der Server sich dazu entscheiden, sie zu schliessen. Anschliessend werden sowohl Client als auch Host benachrichtigt, dass die Verbindung beendet wurde.

– Überlastungskontrolle

Wenn ein Server überlastet wird, drosselt TCP selbstständig die Daten, um einen Zusammenbruch durch Überlastung zu verhindern.

- Einstellungen

Auflösung, Vollbild, Lautstärke und Stärke des Schneefalls können eingestellt werden.

- Kamera

Die Kamera ist beweglich und das ganze Schlachtfeld ist sichtbar. Mit den Pfeiltasten kann man die Kamera frei bewegen und hinein- bzw. hinauszoomen.

- Truppen

- **Gewinnmöglichkeit**

Um zu Gewinnen, muss eine eigene Truppe die Portale des Gegners durchschreiten. Dies verhindern zuerst sogenannte Helden. Sterben jene Helden, erledigt ihr Racheeffekt die ganze Linie auf der sich der Held befand.

- **Lanes**

Unser Spiel hat 4 sogenannten Linien ("Lanes"). Diese unterschiedlichen Linien wurden in der Tiefe verschoben.

- **Synchronisation**

Unser Spiel Synchronisiert alle Daten in Echtzeit. Dadurch sehen alle Spieler immer das Gleiche. Wird der Client aus einem unbekannten Grund disconnected kann er versuchen sich erneut zu verbinden und der Server wird dann direkt alle Daten synchronisieren. Auch hier hilft uns das gewählte Protokoll TCP. Geht ein Paket verloren, wird es automatisch erneut gesendet. Dies hilft uns dabei, die Synchronisation zu jeder Zeit aufrecht zu erhalten.

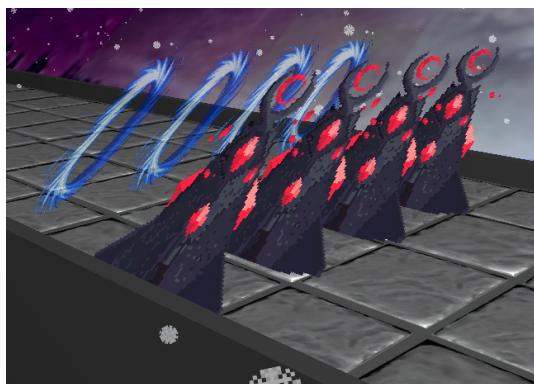
- **Crossplay** Es ist möglich sich mit einem Windows Rechner und einem macOS Rechner zu verbinden. Allerdings muss dem Spiel dazu die Kommunikation durch die Firewall teilweise erlaubt sein. Bei einer eher aggressiveren Firewall wird zudem teilweise der Server-Broadcast geblockt. Dies hätte zur Folge, dass man die Server nicht automatisch finden kann. Verbinden sollte man sich allerdings trotzdem noch können.

5.2.2 Anvisiert

- **Helden**

Beide Spieler haben im Moment den gleichen Helden. Dies ist ein magischer Turm. Er hat keine Angriffsmöglichkeiten. Allerdings kann er sich passiv heilen und so als eine temporäre Grenze zwischen den gegnerischen Truppen und den eigenen Portalen agieren. Die eigenen Truppen werden durch diese Portale hinter den Helden gespawned.

- **Design**



Wir haben uns für ein 2.5d Design entschieden. Dabei ist der Vorteil, dass wir 2d Grafiken benutzen können. Die Sprites werden dabei um 45 Grad gedreht. Auch die Kamera schaut in einem 45 Grad Winkel auf das Geschehen. Dadurch ergibt sich die Illusion einer Perspektive, in der man von der Seite auf das Geschehen schaut.

- **Truppen**

- **Nahkampf** Eine Truppe mit wenig Reichweite.
- **Fernkampf** Eine Truppe die auf Reichweite angreift. Sie schießt Projektile, zum Beispiel ein Bogenschütze, der mit Pfeilen schießt.
- **Suizid** Eine Truppe die bei Berührung mit einem Gegner stirbt und einen Effekt auslöst.

- **Effekte**

- **Gift:** Zeitlich limitierter und wiederholender Schadenseffekt. Eine visuelle Markierung soll vorhanden sein und das gleiche Gift, also der gleichen Truppe, soll nur einmal auf jemanden sein. (Hier: Pilz-Truppe)
- **Rache:** Truppen mit Rache haben einen Effekt nach ihrem Tod. Zum Beispiel eine Truppe beschwören oder Schaden verursachen. (Hier: Helden mit Feuer)

5.2.3 Möglichkeiten

- **Monetarisierung**

1. Kostenpflichtiges Spiel

- Wir haben eventuell vor, unser Spiel nach Vollendung der Dev Phase zu veröffentlichen. Allerdings auf keinen Fall für einen hohen Preis.

- **Design**

Wir werden auch in Zukunft das Design noch verbessern. Hier geht es aber um den Feinschliff, z.B. mehr Hintergründe, und mehr Dinge selbst zu designen.

- **Truppen** Weitere Truppen können jederzeit erstellt werden. Hier ist kein Limit gesetzt. Leben, Schaden, Darstellung, Effekt und vieles mehr kann angepasst werden. Wir haben alle Programme mit dem Hintergedanken geschrieben, dass es jederzeit ohne größere Anpassungen möglich ist, neue Truppen hinzuzufügen.

5.2.4 Fürs Erste verworfen

- **Online Multiplayer**

Um sich mit Rechnern ausserhalb des eigenen Netzwerkes verbinden zu können, müssten wir das ganze Multiplayer System unseres Spieles anpassen, was einen

kompletten Recode zur Folge hätte. Ausserhalb würde sich die Frage nach der Finanzierung stellen, falls wir einen externen Server kaufen / mieten würden.

- **Shop**

Ein Shop ohne Anti-Cheat und/oder Server ist die perfekte Angriffsfläche für Cheater und hat somit nicht viel Sinn. Wir müssten dann ausserdem ein Ranking-System hinzufügen, damit wir die Spieler belohnen können.

- **Kampagne**

Unser Spiel ist vor allem auf den Multiplayer ausgelegt. Eine Kampagne setzt ausserdem voraus, dass man einen offline Gegner ("Bot") hat. Wir haben uns bereits früh gegen eine Kampagne entschieden, da dies nach unserer Meinung nach in diesem Stadium leider eine komplette Zeitverschwendungen wäre.

- **Anti-Cheat**

Dies ist für uns ohne Erfahrung in diesem Bereich und einem lokal berechneten Spiel ein Ding der Unmöglichkeit. Wir haben keine Erfahrung in diesem Bereich und es ist ein extrem komplexes Thema. Allerdings hat der Host sozusagen über fast alles die Macht, der Client kann also fast gar nichts kaputt machen. Wir hoffen ausserdem auf die Menschen, die unser Spiel spielen, und darauf dass sie nicht versuchen zu schummeln.

- **Errungenschaften**

Belohnungen zu erhalten ist immer toll. Wenn es eine Währung wäre, bräuchten wir allerdings auch einen Shop. Man müsste ausserdem neue Dinge hinzufügen, die man zuerst erspielen muss. Es könnten auch Pokale sein, allerdings bräuchten wir dann bereits wieder ein Ranking-System.

- **Bot**

Ein guter Bot passt sich der Spielweise des Gegners an. Einen so guten Bot zu programmieren ist eine eigene Herausforderung für sich. Wir hatten zwar die Idee, dass der Bot immer zufällige Truppen zu zufälligen Zeiten spielt. Allerdings würde dies auch eine Menge Balancing benötigen, wofür wir im Moment leider zu wenig Zeit haben.

- **Helden**

Wir hatten als Ziel, mehrere verschiedene Helden zu gestalten und zu programmieren. Allerdings stellte sich relativ früh heraus, dass wir die nötigen Ressourcen lieber an einer anderen Stelle investieren. Wir haben nun einen Helden, allerdings würde sich das Ausbauen der Helden nach der Abgabe als spannend gestalten.

- **Ingame-Chatfenster**

Wir hatten eine funktionierende Version, allerdings stellte sich dies als überflüssig heraus. Der Grund ist, dass man normalerweise neben der Person sitzt, mit der man im Moment spielt.

- **Mehrsprachig**

Das Spiel ist im Moment nur in Englisch verfügbar. Eventuell werden wir dies nach der Abgabe der schriftlichen Arbeit noch anpassen.

- Effekte

- **Wiederbelebung:** Die Möglichkeit einer Wiederbelebung würde sich nur für spezielle Truppen anbieten. Der Effekt Wiederbelebung würde wahrscheinlich ein Vielfaches davon kosten, wenn man die Truppe erneut losschicken würde.
- **Rüstung:** Uns schien dies nicht nötig zu sein. Dies da es sich erst lohnen würde, sobald wir mehr Truppen, Effekte und mehr hinzugefügt hätten.
- **Aura:** Das Prinzip einer Aura hört sich ganz gut an. Allerdings müssten wir sehr wahrscheinlich einen Recode des Damage-Systems machen. Dies würde uns mehr Zeit kosten, als es im Moment wert wäre.

- **Tutorial**

Ein Tutorial ist ein immenser Zeitaufwand. Es muss so gestalten sein, dass der Spieler gar nicht bemerkt, dass es ein Tutorial ist. Wir haben uns ausserdem darauf geachtet, das ganze Spiel so intuitiv wie möglich zu gestalten.

- **Monetarisierung**

Ohne Anti-Cheat leider vollkommen nutzlos.

- 1. **Kaufbare Gegenstände**

- Neben Schummelmöglichkeiten, stellt sich hier ausserdem die Gefahr eines möglichen Pay-to-Win.

- 2. **Skins**

- Der Spieler bräuchte ein Inventar, das mit einem Server kontrolliert und synchronisiert wird.

- **Zauber**

Wir hatten zu einem frühen Zeitpunkt eine einigermassen funktionierende Testversion. Der Zauber wurde dem Spieler auch auf einer Karte verteilt. Eine erneute Nutzung davon würde allerdings viele Änderungen des gesamten Spiels bedeuten. Dies hätte sehr wahrscheinlich auch einen Recode zur Folge.

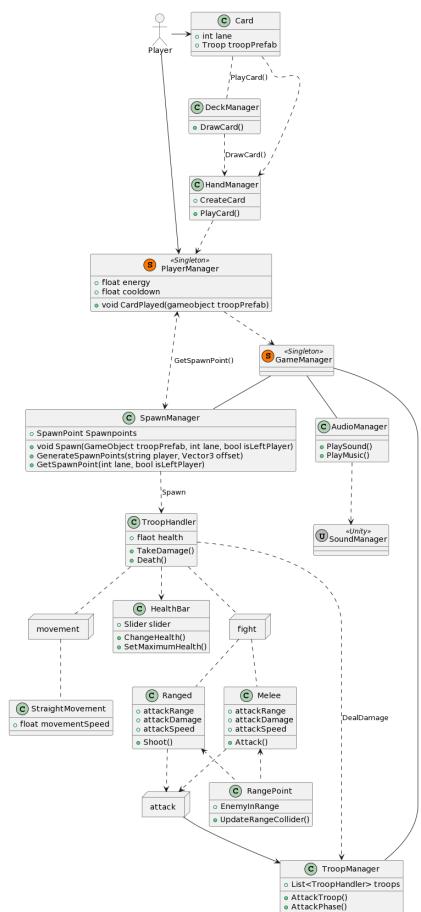
- **Dornen:** Nachdem wir den Zauber "Gift" hinzugefügt hatten, sahen wir den Grund nicht, nun auch noch Dornen hinzuzufügen.

- **Deck**

Die Möglichkeit sich ein eigenes Deck zusammenzustellen, aus dem man dann die Karten zieht, wurde leider aus Zeitgründen fürs Erste verworfen. Allerdings können wir fast schon garantieren, dass wir dies bis zu der Präsentation noch hinzufügen werden.

Kapitel 6

Architektur



github file system

<https://www.youtube.com/watch?v=IQT37uwpcg4>

Kapitel 7

Zukunft

”Der beste Weg, die Zukunft vorauszusagen, ist, sie selbst zu gestalten.” (Abraham Lincoln)

7.1 Ein weisses Blatt

- Das Spiel wird bis zu der Präsentation nicht nur Staub ansetzen. Wir haben fest vor, das Spiel noch weiterzuentwickeln und zu verbessern. Denn trotz grossem Fortschritt ist das Spiel noch lange nicht fertig. Obwohl schon sehr viel existiert, haben wir noch sehr viele Ideen, die wir gerne umsetzen möchten.
- Viele Ideen wurden aus Zeitgründen fürs Erste verworfen oder verzögert. Dies bedeutet allerdings, dass es uns auf keinen Fall an Ideen mangelt. Wir wollen diese fehlenden Features auf jeden Fall noch hinzufügen.
- Das Hinzufügen der meisten Ideen sollte sich dabei nicht sehr schwierig herausstellen. So existiert zum Beispiel die Mechanik für den Effekt ”Dornen“ bereits. Es benötigt lediglich noch einen Feinschliff.
- Hätten wir unendlich viel Zeit, würden wir einen kompletten Recode des ganzen Spiels tätigen. Im Nachhinein wissen wir, dass wir viele Stellen besser schreiben hätten können. Zudem hätten wir dann den Vorteil, dass wir bereits ungefähr wissen, wie alles aufgebaut sein soll.

7.2 Alpha- und Closed Beta-Testing

- Geplant war ein intensives Testing ab der ersten Woche der Sommerferien. Allerdings waren wir zu diesem Zeitpunkt gar nicht zufrieden mit unserem Produkt. Wir wollten auf keinen Fall etwas unserer Meinung nach Ungenügendes aus der Hand geben. So wurde das Datum des Testing immer weiter verschoben, bis es gar kein Datum mehr gab.

- Dies soll sich in der Zukunft allerdings ändern. Es gab zwar ein geschlossenes Alpha-Testing, allerdings war dies nur minimal vorhanden. Bis zu der Präsentation wird das Alpha-Testing ausgeweitet und ein geschlossenes Beta-Testing ist bereits in Planung.
- An Testern wird es auf keinen Fall mangeln. Alle Personen in unserem Umfeld wollen das Spiel ausprobieren. Ob Schulkollegen, Freunde oder Verwandte; sie alle wollen unser Spiel in den Händen halten. Wir sind froh, in einem solchen Umfeld arbeiten zu können.
- Wir hatten bereits die Idee, dass sich mögliche Tester auf einer Warteliste eintragen können. Dafür wurde bereits besprochen, ob wir einen kleinen Server mieten wollen, um eine kleine Webseite hochzuschalten. Die Beta-Tester würden dann per Los ausgewählt werden.

7.3 Veröffentlichung

- Wir planen die erste vollständige Version für die Öffentlichkeit zugänglich zu machen. V 1.0 wird entweder auf itch.io (<https://itch.io/>) oder auf Steam (<https://store.steampowered.com/>) veröffentlicht. Welchen Preis dafür festgelegt wird, steht noch in den Sternen geschrieben.
- Vielleicht werden wir auch nur einen "Buy Me A Coffee"-Button (<https://www.buymeacoffee.com/>) hinzufügen, über den man uns unterstützen kann.
- Der Source-Code bleibt allerdings ziemlich sicher privat. Es ist allerdings möglich, dass wir noch darüber diskutieren werden.

Teil III

Organisation

Kapitel 8

Technologien

8.1 Unity

<https://unity.com/>

”Unity ist mehr als eine Engine”, schreibt Unity auf der eigenen Webseite <https://unity.com/pages/more-than-an-engine>. Fakt ist, Unity ist marktführend. Die Laufzeit- und Entwicklungsumgebung ist vor allem bei Indie-Developern sehr beliebt. Deswegen ist es auch kein Wunder, dass über 50 Prozent aller Spiele mit Unity erstellt wurden. Doch auch für Grafikdesigner, Architekten, Filmentwickler und viele weitere ist Unity ein sehr mächtiges Programm. Und wenn selbst die Streitkräfte der Vereinigten Staaten Unity nutzen, wieso sollten wir noch weiter suchen? Trotzdem haben wir einige für uns wichtige Vorteile herausgesucht:

- Physik
 - Unity berechnet einen grossen Teil der physikalischen Eigenschaften automatisch. Dies ist für uns sehr vom Vorteil, da wir uns zum Glück nicht mit Kollisionen herumschlagen müssen. Ausserdem erlaubt uns Unity die Physik nach Bestreben zu ändern und ganz nach unserem Willen zu formen.
- Grafiken
 - Die Animation von Grafiken ist dank Unity ziemlich einfach. So kann man zum Beispiel eine Bildersequenz in Unity laden, und Unity formt automatisch eine Animation daraus. Jedes Spielobjekt kann seinen eigenen Animator haben. In diesem kann man dann verschiedenen Zuständen des Spielobjekts verschiedene Animationen zuweisen. Greift die Truppe zum Beispiel an, wird von der normalen Animation zur Angriffs-Animation gewechselt.
- Programmierung

- In Unity kann man Spielobjekten selbst geschriebene Programme, sogenannte Scripts, hinzufügen. Diese werden mit C# geschrieben. Dies war von Vorteil, da wir bereits Erfahrung mit C# hatten.

8.2 Github

<https://github.com/>

Github ist die grösste webbasierte open-source Plattform. Auf ihr kann man Source Code speichern und teilen. Ein grosser Vorteil von Github ist, dass man immer zu vergangenen Versionen zurückgelangen kann. Ausserdem kann man verschiedene Versionen des Source Codes in sogenannten Branches speichern und verändern. Github speichert ausserdem jede Änderung am Code, die man durchgeführt hat. So kann man auch nachschauen, wer genau was am Code verändert und geschrieben hat. Ein sehr grosser Vorteil von Github ist, dass man im Team in Echtzeit am gleichen Code schreiben kann. Dazu kommt das grösste Pro-Argument. Backup-Backup-Backup; alles wird gespeichert, und es gibt keine Gründe sich die Festplatte mit Backups vom Code vollzuschreiben.

Eine Visualisierung von unserem Repository kann man hier finden: <https://www.youtube.com/watch?v=IQT37uwpcg4>

8.3 JetBrains Rider

<https://www.jetbrains.com/rider/>

”Rider ist eine unglaubliche .NET-IDE mit der Power von ReSharper”, schreibt JetBrains auf der eigenen Webseite. Dies können wir nur bestätigen. Rider hat zudem die Unity API integriert. Dadurch hat Rider eine spezialisierte Codeinspektion für Unity. Treten Errors im Unity Editor auf, kann Rider direkt den Ursprung des Fehlers anzeigen. Das Debugging ist dadurch um einiges leichter und müheloser. Rider warnt User ausserdem vor schlecht geschriebenen Funktionen, die einen Performance-kritischen Effekt auf das Spiel haben könnten.

8.4 Jira

<https://www.atlassian.com/software/jira>

Jira ist ein webbasiertes Programm für Projektmanagement und Zeiterfassung. Dies ermöglichte uns genau zu erfassen, wie viel Zeit wir an einem bestimmten Problem gearbeitet haben. Uns war klar, dass wir eine Art der Zeiterfassung brauchten, denn im Nachhinein ist dies ein Ding der Unmöglichkeit.

8.5 LaTeX

<https://www.latex-project.org/>

LaTeX verwendet das Textsatzsystem TeX. LaTeX ist eine Software, die sich von anderen

Textprogrammen unterscheidet. Im Gegensatz zu den gewohnten WYSIWYG ("What You See is what you get") Editoren (wie z.B. Word), braucht man zur Benutzung von LaTeX einen separaten Editor. LaTeX vereinfacht den Umgang mit TeX durch einfachere Befehle. Der grösste Vorteil an TeX ist, dass man sehr genau einstellen kann, wie alles aussehen soll.

8.6 Stable-Diffusion

<https://stability.ai/>

Stable-Diffusion ist ein State-of-the-Art, Deep-Learning Text-zu-Bild Modell. Es wurde 2022 vom Start-up Stability AI veröffentlicht. Es wird für die Generation von detaillierten Bildern verwendet. Wir wollten seit Beginn der Arbeit eine sehr neue Technologie verwenden. Dann als SD veröffentlicht wurde, war uns klar, dass wir diese Chance nicht verpassen dürfen. Hier ergab sich, dass wir SD verwendet haben, um einen individuellen Hintergrund zu generieren. Wir nutzen Ideen, eine schlechte Skizze und viele Stunden an Testen und Verbessern, um den Hintergrund nach unserem Geschmack zu generieren.

8.7 Visual Studio Code

<https://code.visualstudio.com/>

Visual Studio Code ist ein kostenloser Editor von Microsoft. Ein grosser Vorteil von VS Code ist, dass man Erweiterungen hinzufügen kann. Die Erweiterung "LaTeX Workshop" (<https://marketplace.visualstudio.com/items?itemName=James-Yu.latex-workshop>) hat dabei das spezifische Syntax-highlighting hinzugefügt. Ohne jene Erweiterung sich die Benutzung von LaTeX um einiges schwerer gewesen.

8.8 Gource

<https://gource.io/>

Gource ist eine gratis open-source Software, die perfekt für die Visualisierung von Github Repositories geeignet ist. Wir haben Gource für die Erstellung eines Videos genutzt, das als Visualisierung unseres Datei-systems dient (<https://www.youtube.com/watch?v=IQT37uwpcg4>).

Kapitel 9

Team Management

9.1 Anfänge

- Bereits Mitte Januar fragte Marc Elia an, ob sie nicht zusammen die Maturitätsarbeit schreiben wollten. Wir beide hatten vor, etwas zu programmieren als Maturaarbeit. Da ergab sich die Idee, dass man die Kräfte vereinigte und zusammen ein Projekt programmiert.
- Die nächste Frage war, was wir als Projekt genau programmieren wollten. Es stellte sich sehr schnell heraus, dass sich ein Videospiel perfekt eignen würde. Dies, da man sich bei einem Videospiel unter anderem frei ausleben kann. Als feststand, dass wir ein Team bilden würden, mussten wir noch eine Lehrperson finden, die uns betreuen würde.
- Marc brachte am Anfang direkt den Vorschlag, dass wir unsere Zeit mit Jira tracken würden. Er setzte zudem Github und Unity für unseren ersten Gebraucht auf.

9.2 Betreuungsperson

- Uns wurde vor Beginn der Arbeit mehrfach gesagt, dass einige Schüler Probleme haben, eine Betreuungsperson zu finden. Um sicherzugehen, dass dies uns nicht geschehen würde, suchten wir bereits sehr früh nach einer Lehrperson.
- Der erste Vorschlag war Albert Kern. Elia hatte ihn am Anfang für eine individuelle Arbeit angefragt. Was lag näher, nun ihn auch für eine Gruppenarbeit anzufragen. Wie es der Zufall wollte, begegneten sich Herr Kern und Elia Ende Februar zufälligerweise auf dem Gang und tauschten einige Worte aus. Herr Kern gab allerdings zu, dass er keine Erfahrung mit der Entwicklung von Videospielen hatte und schlug Elia Martin Hunziker vor. Dieser sei ein begnadeter Videospiel-Fanatiker und wahrscheinlich besser für die Betreuung geeignet.
- Am dritten März schrieben wir eine kurze Anfrage an Herr Hunziker per E-Mail. Herr Kern hatte zuvor im Gespräch mit Elia erwähnt, dass er dabei gerne ein gutes

Wort bei Herr Hunziker einlegen würde. Ob dies tatsächlich stattfand, entzieht sich unserem Wissen. Allerdings sassen wir 4 Tage später, am siebten März bereits bei Herr Hunziker im Büro und besprachen mit ihm unsere Idee.

- Eine Woche später stand seine Unterschrift auf unseren beiden Blättern bezüglich der Maturitätsarbeit. Allerdings hat selbst Herr Hunziker gestanden, dass er uns bei den meisten Problemen nicht helfen können, weil er keine grosse Erfahrung hat.

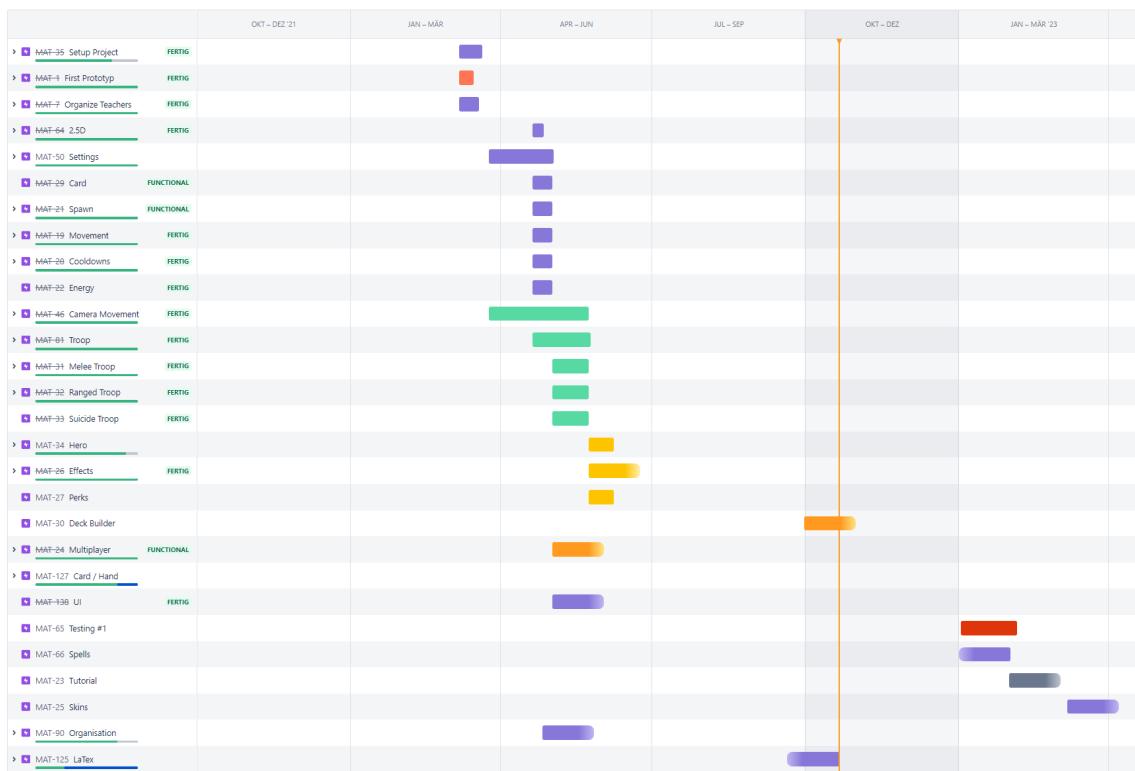
9.3 Kommunikation

- Unsere Kommunikation fand vor allem über WhatsApp statt. Dies geschah allerdings nicht nur in Textform, sondern des Öfteren auch in Telefonaten. Zudem konnte man auf Jira nachschauen, auf welche Aufgabe der Partner seine Zeit ge-tracked hat.
- Wir müssen allerdings zugeben, dass die Kommunikation teilweise sehr gering vorhanden war. Teilweise hatte man keine Ahnung, woran genau der Partner arbeitete. Wir wussten zwar das Ungefährre, allerdings manchmal keine Einzelheiten.
- Die Kommunikation zwischen Schüler und Lehrer war unserer Meinung nach mit sechs Treffen genügend gut. Zudem trafen wir uns dreimal neben der Schule, um unsere Arbeit zu organisieren.

Kapitel 10

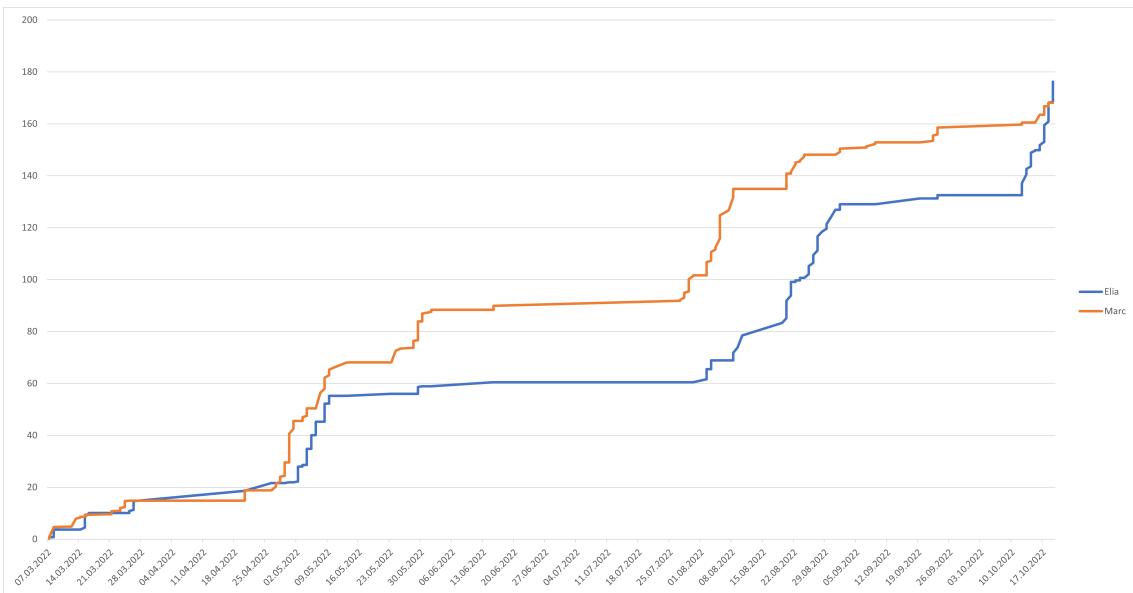
Time Management

10.1 Roadmap



(Abgerundetes Ende bedeutet, dass kein genaues Datum festgelegt wurde)

10.2 Time-Tracking



DA NO EXPECTED IN GRAPH

TEAM HOURS

	getrackte Zeit	erwartete Zeit
Elia		120h
Marc		120h
Team		240h

Teil IV

Reflexion

Kapitel 11

Team

11.1 Rückblick

11.2 Nächstes Mal

meh fix, eimal im moment und notize mache gewisse abschnitte wo der andere keine ahnung davon hat kein 4 augen prinzip

Kapitel 12

Elia

12.1 Rückblick

Das Produkt dieser Arbeit erfreut mich sehr. Obwohl dem Prototyp unseres Videospiels noch einige Funktionen fehlen, bin ich doch grundsätzlich mit unserem Spiel zufrieden. Dies ist sehr wahrscheinlich das grösste und zeitaufwendigste Projekt, an dem ich jemals gearbeitet habe. Unser Github Repository enthält mehr als 40'000 Zeilen an C# Source-Code. Das ganze Repository ist ca. 1.1 Millionen Zeilen lang und mehrere Gigabytes gross (einschliesslich DLL's etc.).

Alleine hätte ich dies nie geschafft, deshalb bin ich besonders froh, dass ich Marc's Unterstützung hatte. Die Gruppenarbeit hat mir viel Druck von den Schultern genommen, da ich mich bei Problemen immer auf Marc verlassen konnte. Zudem führte die Zusammenarbeit zu vielen guten Ideen, die wir auch nur zusammen als Team verwirklichen konnten.

Obwohl von Anfang an klar war, dass dies eine sehr aufwendige Arbeit sein würde, habe ich sie doch unterschätzt. Dies merkte ich auch daran, dass sich mein Zimmer durch die Nutzung des Computers im Laufe eines Tages trotz geöffnetem Fensters um mehrere Grade erwärmt.

Am meisten unterschätzt habe ich allerdings den Arbeitsaufwand der schriftlichen Arbeit. Diese stellte sich als ziemlich grosser Zeitschlucker heraus.

12.2 Nächstes Mal

Für das nächste Mal möchte ich mir zu Beginn des Semesters einen Plan erstellen, der festlegt, dass ich pro Woche mindestens eine Stunde arbeiten werde. Mit diesem Plan wäre es auf keinen Fall zu dieser sehr langen Pause(von Mai bis August) gekommen. Wenn ich erneut die Chance hätte, dieses Projekt in einer Gruppenarbeit zu lösen, würde ich diese Chance ohne zu zögern ergreifen.

Kapitel 13

Marc

13.1 Rückblick

13.2 Nächstes Mal

Teil V

Glossar

Glossar

AAA-Spiel Ausgesprochen "Tripple A", sind Videospiele mit maximalem Budget und meist erstellt von den grössten Gamestudios .

Anti-Cheat Software um Schummeln zu verhindern. Ist sehr komplex und bei allen grösseren online Videospielen vorhanden. Teilweise sehr tief in das System vernetzt.

Cheater Jemand, der unerlaubte Methoden oder Programme verwendet, um sich einen unfairen Vorteil zu verschaffen. Auf Deutsch *ein Schummller*.

Client Die Person, die dem Server einer anderen Person als Mitspieler beitritt.

Crossplay Plattformübergreifendes Spielen. Die Möglichkeit das gleiche Spiel auf zwei unterschiedlichen Plattformen zu spielen (hier: Windows und macOS).

Feature Eine bestimmte Funktion, welche die Software weiterbringt.

Firewall Schutzsystem eines technischen Systems um Viren fernzuhalten. Blockiert aus Sicherheit manchmal mehr als nötig und erschwert so zum Beispiel das Verbinden von zwei Computern über das Internet.

FPS Frames Per Second. Anzahl Bilder pro Sekunde. Höhere FPS ist meistens gleichbedeutend zu einem flüssigeren Spielerlebnis.

Game Studio Ein Unternehmen, dass Videospiele entwickelt .

Github Repository Ein Repository enthält alle Ordner und Dateien des aktuelles Projekts. Innerhalb dieses Repository kann man die ganze bisherige Arbeit anschauen und kontrollieren. Dabei speichert das Repository auch eine Vergangenheit jeder Datei, sodass man Änderungen einfach zurücksetzen kann..

Helden In Videospiele der wichtigste Charakter. In gewissen Spielen ist der Held der Charakter, den man spielt. Der Tod des Helden bedeutet einen massiven Nachteil.

Host Die Person die das Spiel hostet. Sie ist Server und Client.

Indie Developer Ein Indie Developer entwickelt Indie Games, kurz für "Independent Game". Diese werden von einer einzelnen Person oder kleinen Game Studios entwickelt. Der Gegensatz sind AAA-Spiele .

Ingame Im laufendem Spiel, nicht auf dem Computer oder beim Start oder Schliessen des Spiels.

Kamera Die Kamera nimmt in Unity auf, was der Spieler sieht. Mit Kamera ist also das gemeint, was der Spieler sieht.

Lane Bezeichnet generell eine Linie oder Bahn auf der gegangen werden kann. Sehr prägnant in MOBAs wie League of Legends oder Dota. In unserem Spiel eine Bahn der vier, auf welcher Truppen beschwört werden können.

Lokal Im selben Netz oder Computer.

Multiplayer Mehrspieler, also mehrere Spieler spielen zusammen das gleiche Spiel.

Online nicht nur lokal auf dem eigenen Rechner, sondern im Internet.

Pay-to-Win Übersetzt: Bezahlen-zum-Gewinnen, bedeutet, dass mit echter Währung Ingame Vorteile erkauf werden können.

Recode Komplette Umprogrammierung und Reprogrammierung einer Funktion.

Shop ein Ort im Spiel, in dem Dinge gekauft werden können. Ein Ingame-Laden.

Singleplayer Einzelspieler, das Spiel wird alleine, meist nur lokal, gezeigt.

Skins Kaufbare Darstellungen, welche von dem Standard abweichen. In vielen Spielen nur im Tausch mit echter Währung zu erhalten und bringen nur ästhetische Unterschiede.

Steam Steam ist der grösste Videospiel-Anbieter der Welt. Gehört dem Unternehmen Valve.

Tutorial Eine Anleitung, in Videospielen meist eine kurze Spielsequenz, welche Tipps und eine fixe Abfolge besitzt.