



UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik



UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

DIN: New Technologies

Entwicklung eines Augmented Reality-Rollenspiels für Android

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Martin Groppe und Malte Kremer

Erstgutachter: Prof. Dr.-Ing. Stefan Müller

(Institut für Computervisualistik, AG Computergraphik)

Zweitgutachter: Bastian Krayer

Koblenz, im April 2017

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ja Nein

Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.

.....
(Ort, Datum)

(Unterschrift)

Inhaltsverzeichnis

1 Abstract	1
2 Abstract (english)	1
3 Motivation	2
4 Grundlagen	3
4.1 Android	3
4.1.1 Android-Programmierung	3
4.1.2 AR-Funktionen von Android	4
4.2 Unity	6
4.3 Photoshop	7
5 Recherche von Rollenspiel und AR-Hybriden auf Android	8
5.1 Pok��mon Go	8
5.2 Parallel Kingdom	10
5.3 Motion Tennis Cast	11
6 Recherche von Rollenspielen auf dem PC und auf Konsolen	12
6.1 klassische Rollenspiele	12
6.2 Open-World Titel und MMORPGs	13
6.3 Action-Rollenspiele	13
6.4 Taktik-Rollenspiele	13
6.4.1 Final Fantasy Tactics	14
6.4.2 The Banner Saga	14
6.5 Strategie-Rollenspiel-Hybriden	15
7 Konzeption	16
7.1 Fr��he Konzeption	16

7.2	Zweite Konzept-Phase	18
7.3	Finales Konzept/Details	20
8	Implementierung	22
8.1	Karte und Menüs auf Android	22
8.1.1	Haupt-Activity und Karte	23
8.1.2	Quest-Fragment	25
8.1.3	Gruppen-Bildschirm	26
8.1.4	Statuswerte-Bildschirm	27
8.1.5	Activity zum Erschaffen von Charakteren	28
8.1.6	Unity/Kampf-Activity	29
8.2	Erstellung des Kampfes in Unity	30
8.3	A*-Pfadfindung	31
8.4	Der Sieg-Bildschirm	35
8.5	Grafik und Stil	37
8.6	Die Figuren	40
8.6.1	Der erste Wikinger	40
8.6.2	Der Morgenstern-Wikinger	43
8.6.3	Der Berserker	44
8.6.4	Die Priesterin des Ra	45
8.6.5	Der Schakalskrieger	49
8.7	Animationen	52

1 Abstract

Das Mobile-Game Pokémon Go wurde in den ersten drei Monaten nach seiner Veröffentlichung 500 Millionen mal heruntergeladen, als Hybrid aus Rollen- und Augmented Reality-Spiel bleibt es aber eine Ausnahme. Außerdem hat es einige Schwächen.

Vor dem Hintergrund des großen Erfolges und der großen Begeisterung, die Pokémon Go ausgelöst hat, ist es Ziel dieser Arbeit, Pokémon Go und vergleichbare Produkte zu analysieren und gute Aspekte daraus aufzugreifen, um einen Prototypen für ein AR-Rollenspiel zu entwickeln. Dabei ist es unser Ziel, anders als Pokémon Go Spieltiefe und langfristigen Spaß stärker in den Vordergrund zu stellen. Anschließend wird dieser von Testpersonen darauf geprüft, ob er Spaß macht und das Potenzial hätte, langfristig zu fesseln um allgemeine Erkenntnisse über die Qualität des Prototypen zu gewinnen.

2 Abstract (english)

Pokémon Go was an enormous success, it was downloaded 500 million times in the first three months. Nevertheless it remains a rare case of a combination of Augmented reality and role-playing game features. It's also lacking in terms of actual depth and long time progression.

This thesis intends to develop a role-playing game on Android that uses Google Maps features for augmented reality purposes and is fun and engaging. Our goal is also to have more complexity in fights and to make continuous playing interesting. The implementation will then be tested and evaluated.

3 Motivation

Spätestens seit dem massiven Erfolg von Pokémon Go sind Smartphones als Spieleplatform nicht mehr wegzudenken. Dennoch bleibt Pokémon Go eher eine Ausnahme insofern, als die AR-Möglichkeiten des Smartphones selten mit klassischen Spielkonzepten kombiniert werden. Dies ist vor dem Hintergrund überraschend, dass Pokémon Go gezeigt hat, dass es relativ einfach ist, Elemente der realen Welt in Rollenspiele zu integrieren.

In der vorliegenden Arbeit haben wir es uns somit zum Ziel gesetzt zu erforschen, inwieweit es möglich ist, in einem relativ kurzen Zeitraum und mit geringem Budget einen Prototyp zu entwickeln, der AR-Elemente verwendet, kurzfristig Spaß macht und das Potenzial hat langfristig zu fesseln.

4 Grundlagen

Da das Spiel auf dem Betriebssystem Android laufen soll, wurde die Entscheidung gefällt, Interface und Basis-Programm in Java zu programmieren. Als Entwicklungsumgebung für den Java-Code wird Google's eigenes Android Studio verwendet, da es gut dokumentiert und weit verbreitet ist.

Die Kämpfe hingegen werden mit der Unity-game-engine erstellt, sodass für die Erstellung der Grafik nur ein Bildbearbeitungsprogramm benötigt wird. Dazu wird Adobe Photoshop verwendet.

4.1 Android

Android ist ein von Google entwickeltes Betriebssystem für Smartphones. Es basiert auf dem Linux-Kernel und ist eine freie, quellöffentliche Software. Wir haben uns dafür entschieden unser Projekt für Android zu entwickeln, weil es einen sehr hohen Marktanteil von 87.5% besitzt und wir ein möglichst großes Zielpublikum erreichen wollten.

4.1.1 Android-Programmierung

Android Applikationen bestehen aus einer oder mehreren Komponenten. Es gibt vier verschiedene Komponententypen.

- Activities in Android sind Oberflächen, in denen beschrieben wird, was momentan angezeigt wird und wie mit Benutzereingaben umzugehen ist. Android Applikationen haben meistens mehrere Activities. Zum Beispiel könnte es in einer Email-App eine Activity geben, die eine Liste neuer E-Mails anzeigt, eine weitere, in der man neue E-Mails schreiben kann, und eine dritte zum lesen von E-Mails.

- Services laufen im Hintergrund ab ohne dem Benutzer dargestellt zu werden. Während sie am laufen sind können andere Apps im Vordergrund sein. Services werden zum Beispiel zum Abspielen von Musik verwendet.
- Broadcast Receiver sind Komponenten die Systemnachrichten von außerhalb der eigenen Applikation entgegennehmen und behandeln. Solche Nachrichten sind beispielsweise Benachrichtigungen über einen niedrigen Akkustand.
- Content Provider ermöglichen es Daten in Datenbanken oder Dateien abzuspeichern und für andere Applikationen sichtbar zu machen. Außerdem können über sie solche Daten von anderen Apps abgerufen werden.

Um zwischen den Komponenten zu kommunizieren werden sogenannte Intents verwendet. Außerdem werden Activities, Services und Broadcast Receiver über Intents gestartet. In einer E-Mail App würde der Activity zum lesen von E-Mails direkt beim Start über den Intent mitgeteilt werden, welche E-Mail angezeigt werden soll.

4.1.2 AR-Funktionen von Android

Im Gegensatz zu Desktop-Computern erlauben uns Smartphones auch die Umwelt in Spiele mit einzubinden. Mithilfe der Positionserkennung durch Gps, Wifi und Mobilfunkmasten stellt einem Android die Möglichkeit zur Verfügung die Position des Spielers Teil des Spiels werden zu lassen. Ein Service hierfür wird von Google direkt bereitgestellt, so dass man sich nicht für eine der drei Möglichkeiten entscheiden muss, sondern direkt das beste Ergebnis bekommt.

Auch die Sensoren des Smartphones können in ein Spiel eingebaut werden. Durch das Accelerometer kann man versuchen Gesten zu erkennen und das Spiel darüber steuern. Im Spiel kann es Events geben die von der Temperatur abhängen und über den Orientierungssensor kann abgefragt werden ob ein Spieler auf virtuelle Objekte oder andere Mitspieler zeigt. Eine weitere Möglichkeit die reale Welt in Spiele mit einzubinden ist die Kamera des Smartphones. Durch sie können virtuelle Objekte in der realen Welt angezeigt werden.

4.2 Unity

Unity, eine von Unity Technologies entwickelte Game engine ist seit ihrer Veröffentlichung 2005 zunehmend erfolgreich. Sie wird von Indie-Entwicklern wie William Chyr Studios ebenso wie von Publisher-abhängigen Entwicklungsstudios wie z.B. Square Enix Montreal benutzt. Unity ermöglicht einfache Entwicklung für die Smartphone-Betriebssysteme iOS, Android, Windows Phone, aber auch für Computer-Betriebssysteme und für Konsolen wie die Playstation 4 oder den 3DS.

Insgesamt sind mit Unity über 238.000 Spiele¹ für den mobilen Markt und etwa 34% der Top 1.000 mobilen Spiele entwickelt worden, es ist die meistbenutzte nicht-inhouse-Engine. Allein im dritten Quartal 2016 hatte Unity über fünf Milliarden downloads. Außerdem ist es benutzerfreundlich und gut dokumentiert. Darüber hinaus vereinfacht die große Verbreitung den Einstieg zusätzlich.

Unity bietet die Möglichkeit Spiele in 2D sowie in 3D zu programmieren und bietet übersichtliche objektorientierte Verknüpfungen von Scripts und Game-Objekten. Unitys Script Compiler arbeitet mit C# oder JavaScript. Darüber hinaus hat es Automatismen für die Erstellung von Animationen aus Basis-Data wie 3D-Modellen oder Spritesheets.

¹laut unity analytics

4.3 Photoshop

Das Bildbearbeitungsprogramm Photoshop wurde 1988 von Adobe entwickelt. Seitdem wurden mehrfach neue Versionen entwickelt. Mit Photoshop werden Bilder auf Pixel-basis bearbeitet. Dies ermöglicht relativ einfache Erstellung von sogenannten Sprites, also Bildern die als Basis für 2D-Figuren dienen.

Photoshop ist unter Game-Artists weit verbreitet und wird für die Erstellung von Storyboards, Sketches und Sprites benutzt. Animation Career Review, eine Website, die verschiedene Hilfestellungen für Animatoren und Game-Artists wie Jobangebote, Beschreibungen von Jobs und Studiengänge, Stipendien und Interviews bietet, listet es als die essenziellste Software für Künstler und Designer. Bloopanimation, eine Website die Tutorials, Artikel und Guides zu Animation im Allgemeinen und zu Filmanimation im Speziellen bietet, empfiehlt Photoshop ebenfalls für 2D-Animation und bezeichnet es als "großartige Wahl"("great choice"). 2005 lag Photoshops' Marktanteil zwischen 60% und 70%² im Bereich Bildbearbeitungssoftware für Rastergrafik, 2010 hatte Photoshop über 10 Millionen Nutzer weltweit. Außerdem ist das Programm gut dokumentiert und leicht zu lernen (eine Google-Suche nach Photoshop Tutorial lieferte rund 14.500.000 Ergebnisse). All diese Faktoren machten es zu einer guten Wahl für die Erstellung der Grafik.

²Bundeskartellamt, http://www.bundeskartellamt.de/SharedDocs/Entscheidung/DE/Entscheidungen/Fusionskontrolle/2005/B7-162-05.pdf?__blob=publicationFile&v=3

5 Recherche von Rollenspiel und AR-Hybriden auf Android

5.1 Pokémon Go

Pokémon Go ist ein freemium/free-to-play Smartphone-Spiel für iOS und Android. Es wurde von Niantic im Juli 2016 veröffentlicht und benutzt GPS sowie situational das Gyroskop und die Kamera für AR-Funktionen.

Pokémon Go ist bis heute das erfolgreichste Spiel, das für Smartphones erschienen ist, mit über 650 Millionen Downloads (Stand 27.02.2017) und über 500 Millionen Downloads in den ersten zwei Monaten. In 2016 haben gleichzeitig an einem Tag 23 Millionen Nutzer gespielt.

Im Folgenden wird grob die Funktionsweise von Pokémon Go erklärt. Pokémon Go benutzt GPS, um die Position der Spieler zu erfassen und auf einer Google Maps-ähnlichen Karte anzuzeigen. Niantics' Server erzeugt Pokémon (Monster, die man fangen, trainieren und zum kämpfen benutzen kann) an mehr oder weniger zufälligen realen GPS-Positionen. Dabei berücksichtigt der Server eine Reihe von Faktoren. Pokémon vom Typ 'Wasser' werden z.B. eher in der Nähe von Flüssen, Seen und Brunnen erschaffen, während Pokémon vom Typ 'Geist' eine erhöhte Chance haben, bei Nacht aufzutauchen. Wenn der Spieler nah genug an ein Pokémon heran kommt, kann er das Pokémon sehen und auf es tippen, um einen Bildschirm aufzurufen, in dem er versuchen kann, es mit einem Pokéball zu fangen. Sollte es dem Spieler nicht gelingen, dass Pokémon innerhalb einer bestimmten Zeit zu fangen, entkommt es.

Außerdem gibt es an festgelegten Orten sogenannte Poké-Stops, an denen Spieler Items wie die bereits erwähnten Pokébälle bekommen können. Zusätzlich gibt es Arenen, an denen Spieler gegen die Pokémon von anderen

Spielern kämpfen können um Items zu bekommen. Dies führt dazu, dass Spieler häufig in der realen Welt bestimmte Routen ablaufen und sich an festgelegten Orten immer wieder begegnen, was mehr oder weniger automatisch zu neuen Kontakten und Freundschaften führt. Neben den realen Kontakten, mit denen man das Spiel spielen kann, und den Orten, die man besucht, sorgt auch die Kamera-Anzeige, in der Pokémon vor einem realen Hintergrund angezeigt werden und die Anpassung von Pokémon in Gebieten für eine Verknüpfung von realer und virtueller Welt.

Das Spiel ist kostenlos herunterladbar, im Shop kann man z.B. Pokébälle oder Inventar-Erweiterungen für eine Ingame-Währung kaufen, die man auch mit realem Geld kaufen kann. Nutzer die reales Geld ausgeben, haben dementsprechend Vorteile, die einen schnelleren Fortschritt ermöglichen. Laut Schätzungen hat Pokémon Go 2016 etwa 950 Millionen US-Dollar Einnahmen erzeugt.

Auch wenn Pokémon Go alle Smartphone-Spiel-Rekorde gebrochen hat, gibt es doch einige Kritik. Die Interaktionsmöglichkeiten mit anderen Spielern hielten sich stark in Grenzen, das Monetisierungs-Modell war recht unterentwickelt und es gab kein wirklich forderndes Element. Arena-Kämpfe erwiesen sich als recht repetitiv und anspruchslos und echtes Player versus Player-Gameplay, von jetzt an PvP, bei dem beide Seiten aktiv Einfluss auf das Kampfgeschehen nehmen können, gibt es nicht. Es gibt auch keinen anderen fordernden Endgame-Inhalt, sodass viele Spieler Pokémon erst trainierten, schließlich aber feststellen mussten, dass sie nicht viel mit den trainierten Pokémon machen konnten.

5.2 Parallel Kingdom

Das von PerBlue 2008 auf den Markt gebrachte Parallel Kingdom war ein Smartphone-Spiel, das Elemente aus verschiedenen Spieltypen. So verwendet es Elemente von MMORPGs (s.u.), Geocache-Spielen, also Spiele die ihre Spieler auf der realen Welt nach virtuellen Gegenständen suchen lassen, und von Echtzeit-Browser-basierten-Multiplayer-Titeln in denen man Gebiete einnimmt, sie verteidigt und bebaut. Es lief server-basiert und wurde im November 2016 abgeschaltet.

So konnte man z.B. Ressourcen und Items sammeln und tauschen, Dungeons erforschen und Monster bekämpfen. Das Spiel spielte auf einer realen Weltkarte, auf der man sich (im Gegensatz zu Pokémon Go und unserem Spiel) auch virtuell fortbewegen konnte und Territorium beanspruchen konnte. OGame Features wie Gebäude, die stetigen Ertrag bringen und PvP-Kriege um Gebiete boten zusätzliche Langzeitmotivation.

Das Spiel war mit über 1.000.000 Spielern recht erfolgreich, wurde mehrfach für Awards nominiert und hatte mit acht Jahren eine für ein Android Spiel sehr lange Lebensdauer. Die Abschaltung war laut Entwickler hauptsächlich wegen der veralteten Technik notwendig. Die lange Zeit, über die das Spiel lief, zeigt, dass die Konzepte gut ineinander griffen und es genug Inhalte gab, um Spieler auf lange Sicht zu fesseln. PerBlue machte mehrere große Updates, die zusätzliche Features und Gebiete brachten. Leider waren viele Bereiche im Kern recht simpel, Kämpfe bestanden häufig daran, dass Spieler nebeneinander liefen und anschließend Schläge tauschten. Am Ende wurden sie oft durch Items und Level entschieden und nicht von den Entscheidungen der Spieler. Als wir anfingen, das Spiel zu testen, war es bereits wenige Monate vor der Abschaltung und Fortschritt war in dem Spiel recht langsam, sodass wir nicht dazu kamen, die großen PvP-

Funktionen zu testen, da die Nutzerzahl schon stark geschrumpft war.

5.3 Motion Tennis Cast

In dem 2007 veröffentlichten Motion Tennis Cast spielt man Tennis mit seinem Smartphone. Das Spielkonzept ist ähnlich wie der Tennis-Teil von Wii Fit, das Smartphone dient als Controller und wird mit einem Fernseher oder Computer verbunden, der dann als Bildschirm dient. Dabei benutzt das Spiel die Bewegungssensoren des Smartphones für Motion Capturing und der Spieler simuliert mit seinem Smartphone Schläge mit einem Tennisschläger. Leider benötigt das Spiel zusätzliche Software und laut User-Reviews ist das Game instabil und ungenau.

6 Recherche von Rollenspielen auf dem PC und auf Konsolen

6.1 klassische Rollenspiele

Klassische Rollenspiele erzählen häufig eine Geschichte aus Sicht einiger weniger Charaktere und schildern dabei meistens, wie Charaktere stärker werden und ihre Wichtigkeit in der virtuellen Welt zunimmt. Ziel ist es, dass der Hauptcharakter sich mit der Figur/den Figuren assoziiert. Dabei unterscheiden sich westliche und östliche klassische Rollenspiele, während westliche klassische Rollenspiele wie Knights of the old Republic, die Dragon Age-Reihe oder die Witcher-Serie oft einen größeren Fokus auf spielerische Freiheit und Erkundung legen, priorisieren östliche Rollenspiele, wie die Tales-, oder die Final Fantasy-Serie meist das Erzählen einer epischen, oft tragischen Geschichte. Kämpfe, Level (eine Funktionalität zur Steigerung der Stärke des Charakters) und Items sind in nahezu allen klassischen Rollenspielen vorhanden, stehen aber im Verhältnis zu z.B. Action-Rollenspielen eher im Hintergrund. Gelegentlich werden kleine Rätsel eingebaut.

Das Genre nähert sich wieder zunehmend den Open-World-Titeln an, Spiele welche ihren Spielern durch große virtuelle Gebiete ohne Ladezeiten das Gefühl von Freiheit geben. So besaßen sowohl der dritte Teil der Witcher-Reihe The Witcher 3: Wild Hunt (2015) als auch der dritte Teil von Bioware's Dragon Age Serie Dragon Age: Inquisition (2014) eine frei begehbar Welt und das Open-World-Spiel The Elder Scrolls 5: Skyrim (2011) hatte ausgereiftere Levelmechaniken, eine bessere Präsentation seiner Geschichte und mehr Dialog-Sprecher (83) als seine Vorgänger. Im Gegensatz dazu haben die meisten klassischen Rollenspiele, die älter als fünf Jahre sind zwar ge-

legentlich weitläufige Gebiete, aber meistens kurze Level-Abschnitte.

6.2 Open-World Titel und MMORPGs

Open-World-Spiele setzen den Spieler in eine riesige, frei erkundbare Welt. Dabei ist zu unterscheiden zwischen Einzelspieler-Spielen wie der Elder Scrolls Serie (zuletzt The Elder Scrolls 5: Skyrim (2011)) und sogenannten MMORPGs wie World of Warcraft (2004, letzte Erweiterung 2016), Spielen in denen oft hunderte Spieler auf dem gleichen Server in der gleichen Instanz der Welt spielen. Da unser Spiel allerdings ein AR-Spiel sein sollte, war die Kreierung einer großen virtuellen Welt für uns eher uninteressant, außerdem ziehen MMORPGs oft ihren Reiz aus der Interaktion mehrerer Spieler miteinander, was eine gute abschließende Auswertung erschweren würde.

6.3 Action-Rollenspiele

Action-RPGs wie die Diablo-Serie von Blizzard, Titan Quest, die Sacred-Reihe oder das Freemium-Spiel Path of Exile konzentrieren sich eher auf schnelles Gameplay mit viel Action. Der Held erlebt in der Regel eine eher oberflächliche Geschichte und besiegt hunderte Feinde in effektgeladenen Kämpfen. Dabei sorgt ein stetiger Strom an Gegnern für einen guten Spielfluss. Langzeit-Motivation wird häufig durch zufällig generierte Items, Klassen mit unterschiedlichen Fähigkeiten und Mehrspieler-Inhalte erzeugt.

6.4 Taktik-Rollenspiele

Taktik-Rollenspiele sind Rollenspiele, in denen man ein eher strategisches Gameplay hat. In Spielen wie der Disgaea-Serie, der Final Fantasy Tactics-Reihe oder der Fire Emblem-Serie spielt man rundenbasiert größere Grup-

pen auf Schachbrett-artigen Feldern. Dabei hat jede Figur Items und ein eigenes Level und gehört häufig einer Klasse an, die eigene Stärken und Schwächen mit sich bringt. So können Magier in der Regel über größere Distanzen viel Schaden anrichten, sind aber eher sehr gefährdet wenn Gegner an sie herankommen. Der 2014 erschienene Indie-Titel The Banner Saga fällt ebenfalls in diese Kategorie.

6.4.1 Final Fantasy Tactics

Das 1997 veröffentlichte Playstation-Spiel Final Fantasy tactics und sein 2003 erschienener Nachfolger für den Gameboy Advance sind Klassiker unter den Taktik-Rollenspielen. Ausgereifte und komplexe Level- und Klassensysteme sorgen für viele Möglichkeiten bei der Charakter-Entwicklung und für Langzeitmotivation und die Kämpfe sind immer ein bisschen anders. Stil und Grafik sind bei beiden Titeln gut, aber während Final Fantasy Tactics eine ausgereifte und komplexe Geschichte hat, steht diese bei Final Fantasy Tactics Advance eher im Hintergrund.

6.4.2 The Banner Saga

The Banner Saga ist ein von Stoic entwickeltes Taktik-Rollenspiel. Es wurde von drei ehemaligen BioWare-Mitarbeitern produziert, die über Crowdfunding Geld gesammelt haben, um ihr Projekt zu finanzieren. Banner Saga wurde über Steam und GOG.com (Good old Games) vertrieben und wurde alleine in Steam über 600.000 mal verkauft.

In Banner Saga spielt man zwei Gruppen, die versuchen in einem apokalyptischen Szenario zu überleben. Banner Saga orientiert sich in Stil und Handlung stark an der nordischen Mythologie im allgemeinen und an Ragnarök, dem Weltuntergang. Eine Gruppe besteht hauptsächlich aus soge-

nannten Varls, gehörnte Riesen, die versuchen zu ihrer Hauptstadt zu kommen, während die andere Gruppe hauptsächlich aus menschlichen Flüchtlingen besteht und nach Süd-Westen zieht, um der Invasion der sogenannten Dredge aus dem Norden zu entkommen. Banner Saga erzählt eine tragische Handlung, die vom Spieler beeinflusst werden kann. Kämpfe werden auf einem Schachbrett ausgetragen und Figuren besitzen Spezialfähigkeiten, die im Laufe des Spiels stärker werden.

6.5 Strategie-Rollenspiel-Hybriden

Strategiespiele sind Spiele, in denen man Armeen bewegt und mit ihnen Schlachten schlägt. Häufig muss man diese Armeen erst in Basen aufbauen, indem man Rohstoffe abbaut und Ausbildungsgebäude baut. Spiele wie die Heroes of Might and Magic- oder die Disciples Serie sind eher im Strategie-Bereich als im Rollenspiel-Sektor anzusiedeln, haben aber starke Rollenspiel-Komponenten. So haben diese Spiele, anders als die vorher erwähnten Taktik-Rollenspiele Basen, die man ausbaut und in denen man Truppen baut, sowie eine Weltkarte, über die man frei zieht. Allerdings wird der Anführer der Armee im Laufe des Spieles ebenfalls stärker und kann Items finden und in Disciples bekommt auch die Armee Erfahrung.

7 Konzeption

7.1 Frühe Konzeption

Unsere ersten Konzepte legten, ähnlich wie Pokémon Go, einen stärkeren Fokus auf Augmented Reality. Ursprünglich sollte der Spieler eine einzelne Figur steuern und z.B. mithilfe von Motion-Capturing über Gesten in der Realität auswählen, welche Fähigkeit die Figur ausführen soll. So wollten wir uns im Kampfsystem stärker an Action-Rollenspielen orientieren, was auch Mehrspieler-Modi ermöglicht hätte.

Unsere Recherche hat allerdings ergeben, dass bisher existierende Beispiele für Motion Capturing mit dem Smartphone durch Ungenauigkeiten Probleme verursachen und das Projekt wahrscheinlich viel Feinabstimmung bräuchte, sodass es fragwürdig war, ob innerhalb von sechs Monaten ein vorzeigbares Ergebnis zu schaffen ist. Außerdem brauchten wir einen zweiten portablen Bildschirm, andernfalls müsste der Spieler zwischen seinen Bewegungen und dem Blick auf das Smartphone abwechseln, was allerdings den Spielfluss zu stark belasten würde. Das einzige Motion-Capturing Android Spiel, dass das Smartphone wie eine Wiimote (Wii-Controller) benutzte, erforderte für diesen Zweck einen Fernseher oder einen Laptop. Diese Lösung erlaubt jedoch kaum Mobilität.

Vorstellbar wäre ein solches Konzept, wenn die Smartphone-VR-Technologie und die Smart-Watch weiter entwickelt wären, aber selbst dann wäre es schwer bis unmöglich, innerhalb von sechs Monaten einen gut funktionierenden Prototyp zu kreieren, solange Motion-Capturing nicht von einem anderen Programm übernommen wird.

Daher fokussierten wir uns eher auf anspruchsvolles Gameplay und funktionierende Rollenspiel-Systeme als eine stärkere AR-Anbindung. Dazu war

es sinnvoll, sich Rollenspiele auf anderen Plattformen stärker anzusehen und zu überlegen, welche Ideen wir übernehmen und erweitern können.

7.2 Zweite Konzept-Phase

Klassische Rollenspiele haben in der Regel mehrere Autoren für die Geschichte und Sprecher, was im Rahmen einer Bachelor-Arbeit mit zwei Personen nicht produzierbar ist. Außerdem fehlt uns Erfahrung im Schreiben größerer Geschichten mit vielen Charakteren. Daher entschieden wir uns, einen Hybriden aus den Stärken anderer Subgenres zu erzeugen, der in absehbarer Zeit produzierbar ist und gleichzeitig Spaß macht.

Unsere Kämpfe orientieren sich an denen von Taktik-Rollenspielen, da einer der großen Kritikpunkte an Pokémon Go die mangelnde Spieltiefe war und Kämpfe in Taktik-Rollenspiele durch ihr Schachbrett oft Tiefe erzeugen, ohne so hektisch wie die von Action-Rollenspielen zu sein.

Tests zeigten, dass Kämpfe in Taktik-Rollenspielen zwar durchaus lange dauern konnten, diese in den Taktik-Rollenspielen sehr ähnlichen Rollenspiel-Strategiespiel-Hybriden wie die Heroes of Might and Magic-Serien aber unter einer Minute liegen konnten, also immer noch unterwegs gut spielbar wären. Die längere Dauer in Spielen wie Fire Emblem oder Disgaea liegt häufig an der Spielfeldgröße und der größeren Menge an Kampfteilnehmern (in der Fire Emblem Serie können durchaus 40 Charaktere an einer Schlacht teilnehmen). Da wir aber schnelle Kämpfe benötigen, damit Spieler nicht nur auf der Stelle stehen und kämpfen, sondern auch immer wieder weitergehen, entschieden wir uns für mehrere Optimierungen. So wird die Anzahl an Kampfteilnehmer sowie das Schlachtfeld klein gehalten und Schaden ist im Verhältnis zu Lebenspunkten relativ hoch, sodass Kämpfe nach ungefähr drei Runden vorbei sind. Um den Strategiefaktor nicht zu vernachlässigen und das Spiel komplex zu halten, entschieden wir uns für mehrere Klassen mit verschiedenen Stärken und Schwächen.

Level- und Itemsysteme werden dagegen stärker an Action-Rollenspielen

und MMORPGs angelehnt. Sie sollen eine Motivationsquelle für Spieler sein und für ein zusätzliches Belohnungsgefühl und für ein Gefühl des stärker werden sorgen. Unsere Items werden Genre-typisch automatisch generiert und kommen in verschiedenen Seltenheitsstufen und Leveln/Stärken. Unser Ziel war es, dass seltene Items große Vorteile bringen, damit Spieler sich freuen, wenn sie sie bekommen.

Zusätzlich sieht unser Konzept ein Questsystem vor, welches Spieler Belohnungen gibt und sie dazu motivieren soll, sich ihre Kämpfe genauer auszusuchen. Dabei unterteilen wir in eine kleine und eine große Quest, während die kleine Quest in der Regel innerhalb einer Stunde erledigbar sein soll, wird die große Quest dazu gedacht sein, Spieler über einen längeren Zeitraum zu beschäftigen. Dadurch wird für eine gewisse Langzeit-Motivation gesorgt.

Die Optik soll sich dabei an Taktik-Rollenspielen und Strategie-Rollenspiel-Hybriden orientieren, wobei versucht wird, anders als Pokémon stärker erwachsenes Publikum zu erreichen, da unser Spiel stärker auch Langzeit-Spieler ansprechen soll. Gleichzeitig soll das Spiel aber weiterhin für junges Publikum spielbar bleiben.

In Tests sorgten 3D-Modelle insbesondere bei Animationen häufig für Ecken an Stellen die eigentlich rund sein sollten, sodass jedes Keyframe einer Animation erneut nachbearbeitet werden musste. Daher haben wir uns für 2D-Sprites und gegen 3D-Modelle entschieden, da diese in relativ kurzer Zeit bessere Ergebnisse lieferten.

7.3 Finales Konzept/Details

Am Ende haben wir uns entschieden, vier unterschiedliche Klassen zu designen. Dies erlaubt es, die Klassen allein durch ihre Statuswerte zu spezia-

lisieren. Eine Klasse kann viel Schaden überleben, eine auf lange Distanzen angreifen, eine macht viel Schaden und eine hat eine große Bewegungsreichweite. Dadurch wird ein Schere-Stein-Papier-gameplay ermöglicht. So kann unser Fernkämpfer mehrere Schüsse auf die Klassen mit hohem Schaden und viel Leben abfeuern, bevor diese zurück angreifen können, während die Figur mit der großen Bewegungsreichweite gegen die Fernkampf-Klasse leichtes Spiel hat, aber den Kampf gegen die Klassen mit hohem Schaden und viel Leben verliert. Die Klasse mit viel Schaden ist kampfstärker als die Klasse mit viel Leben, während im Gegenzug die Klasse mit viel Leben mit gutem Spiel einen Großteil des Schadens einstecken kann und damit gegen die künstliche Intelligenz, die nach einem Muster bei der Wahl ihrer Ziele vorgeht, einen großen Vorteil erzeugen kann.

Spieler steuern im Kampf eine vorher ausgewählte Gruppe aus drei Figuren. Mehr hätten bedeutet, dass das Spielfeld größer sein müsste, um nicht überfüllt zu werden. Weniger hätte Tiefe aus der Gruppenerstellung und dem Kampf genommen. Aufgestellte Figuren bekommen Erfahrung und nach jedem Kampf wird ein Item aus einer zufälligen Itemgruppe und von zufälliger Stärke fallengelassen, wobei stärkere Items seltener sind, um Spielern das Gefühl zu geben, dass epische Items etwas besonderes sind und sich freuen, wenn sie welche bekommen.

Gegnergruppen setzen sich aus zufälligen Konstellationen der Klassen zusammen und bekommen zunehmend mehr Statuspunkte, um auszugleichen, dass sie keine Items tragen. Das Level der Gegner hängt dabei vom Level der Spielergruppe ab. Anfangs sind Gegner schwächer, weil Spieler noch keine Items haben. Damit Kämpfe sich abwechslungsreich spielen, wird nicht nur drei gegen drei gekämpft, sondern es können auch einzelne oder auch zwei Gegner auftauchen. Damit die Kämpfe nicht trivial werden,

haben zwei Gegner jeweils 25% mehr Leben, Angriff und Initiative (Figuren mit mehr Initiative sind früher dran) als einzelne Gegner, während ein einzelner Gegner eine Herausforderung darstellen soll und als Boss auftritt, mit massiven Boni, einem größeren Modell und besseren Belohnungen. Damit Boss-Kämpfe besonders bleiben, treten sie deutlich seltener auf.

Bei den Items wurde sich für drei unterschiedliche Typen entschieden. Waffen erhöhen den Schaden und tauchen in unterschiedlichen Formen auf, die jeweils nur von einer Klasse getragen werden können. Dies ermöglicht uns, Waffen für die Klassen einzeln anzupassen und zu verhindern, dass z.B. die Fernkampf-Klasse auf lange Distanzen nahezu den gleichen Schaden macht, wie die Klasse mit hohem Schaden. Rüstungen erhöhen die Lebenspunkte und kommen ebenso wie unsere dritte Kategorie, Schuhe, in einer Form, die von allen Klassen tragbar ist. Schuhe erhöhen die Initiative und gibt einen kleinen Bonus auf Lebenspunkte, während seltenere Formen zusätzlich die Bewegungsreichweite erhöhen.

8 Implementierung

Die Implementierung umfasste mehrere verschiedene Teilespekte und Sprachen, sowie das Erstellen von Grafik und Menüs, wobei Martin sich hauptsächlich auf die Android Seite und das eigentliche Kampfmodul konzentrierte und Malte auf die Grafik, Design-Entscheidungen und die Erstellung des Siegesbildschirms in Unity.

Die App wurde nach und nach modular erzeugt. So wurde am Anfang der Fokus auf den Unity-Teil gelegt, in dem unser Kampf abläuft. Dabei übernahm Martin die Programmierung des Schachbrettes, der Figuren, der Wegfindung und der künstlichen Intelligenz, während Malte sich auf Stil und Grafik konzentrierte. Danach erzeugte Martin das Android-Modul und Malte Animationen, Buttons, Hintergründe und einen Siegesbildschirm in Unity. Beide waren an allen Entscheidungen hinsichtlich des Gamedesigns beteiligt.

8.1 Karte und Menüs auf Android

Unser Android-Code besteht aus vier Activities. Diese werden im folgenden nun einzeln beschrieben.

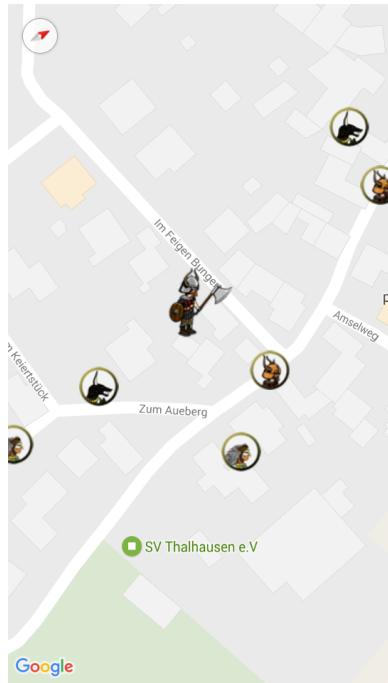


Abbildung 1: Karte mit Spielfigur in der Mitte. Die Portraits repräsentieren Gegnergruppen

8.1.1 Haupt-Activity und Karte

Unsere Haupt-Activity, auf der der Benutzer startet, besteht aus drei Fragmenten, zwischen denen man hin und her tauschen kann, indem man mit dem Finger über den Bildschirm wischt. In der Mitte befindet sich unsere Karte. Auf unserer Karte wird die Position des Spielers, die über GPS abgerufen wird, sowie die Position von Gegnergruppen, gegen die man kämpfen kann, angezeigt. Wird eine Gegnergruppe angeklickt, werden in einem Fenster Informationen zu den Gegnern angezeigt. Über dieses Fenster kann man den Kampf starten, wenn man nah genug dran ist. Die notwendigen Daten für den Kampf werden als Json Strings serialisiert und der Activity, mit der Unity ausgeführt wird, über den Intent mitgeteilt. Json, kurz für Java Script Object Notation, ist ein kompaktes Datenformat in dem man

leicht Java Objekte serialisieren kann. Wir verwenden es auch um die Daten unserer Heldengruppe abzuspeichern.

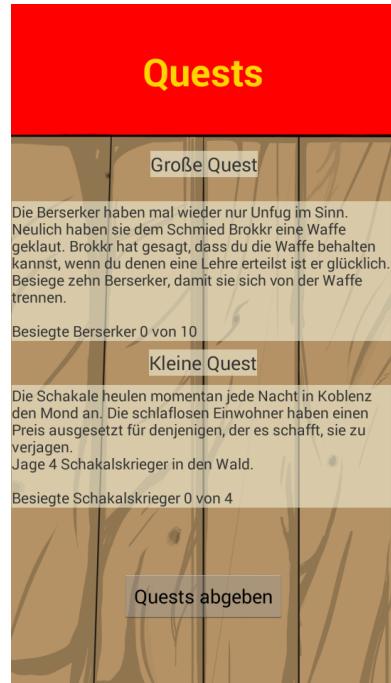


Abbildung 2: Questfragment mit momentan aktiven Aufgaben

8.1.2 Quest-Fragment

Im rechten Fragment der Haupt-Activity befindet sich eine Anzeige für momentane Aufgaben, die der Spieler erfüllen kann, um zusätzliche Belohnungen zu erhalten. Wenn der Spieler die Aufgabe erfüllt hat, kann er diese über den Button abgeben und er bekommt einen zufälligen Gegenstand auf dem Level seiner Gruppe als Belohnung.



Abbildung 3: Charakter Fragment. Die aktive Party ist in der roten Leiste. Waterloo wurde angeklickt um ihn auszutauschen. Die möglichen Ziele für den Tausch sind Leipzig und Aspern

8.1.3 Gruppen-Bildschirm

Im linken Fragment befindet sich unser Gruppen-Bildschirm. Auf ihm wird dem Spieler die eigene Gruppe angezeigt und er kann auswählen, welche Charaktere am Kampf teilnehmen sollen. Die Charaktere in der oberen Leiste sind momentan aktiv und man kann sie indem man sie anklickt gegen einen beliebigen inaktiven Charakter austauschen. Wenn man auf einen Charakter im unteren Teil klickt, wird die Activity zur Darstellung der Statuswerte gestartet und ihr werden mit einem Intent die Daten des angeklickten Charakters übergeben. Über den Button unten rechts wird eine Activity gestartet, die dem Spieler ermöglicht, neue Charaktere zu erstellen.



Abbildung 4: Statusbildschirm von Waterloo. Unten kann man die momentan ausgerüstete Waffe austauschen. Über die drei Symbole kann man zwischen Waffen, Rüstungen und Schuhen wechseln.

8.1.4 Statuswerte-Bildschirm

Die Activity zur Darstellung der Statuswerte zeigt alle Werte der Figur an, über die sie aufgerufen wird. Außerdem kann man hier der Figur gefundene Gegenstände anziehen um ihre Statuswerte zu verbessern.

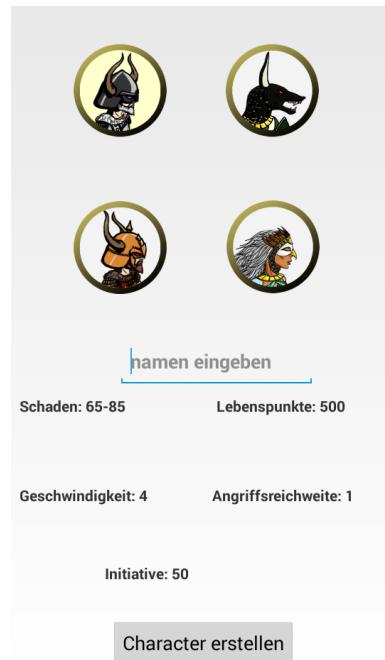


Abbildung 5: Hier kann man einen neuen Charakter erstellen.

8.1.5 Activity zum Erschaffen von Charakteren

In der vierten Activity kann man eigene Charaktere erstellen. Über die vier Bilder kann man die Klasse des Charakters aussuchen und über das Textfeld einen Namen eingeben. Die Activity zeigt einem zusätzlich die Statuswerte der momentan ausgewählten Klasse an.

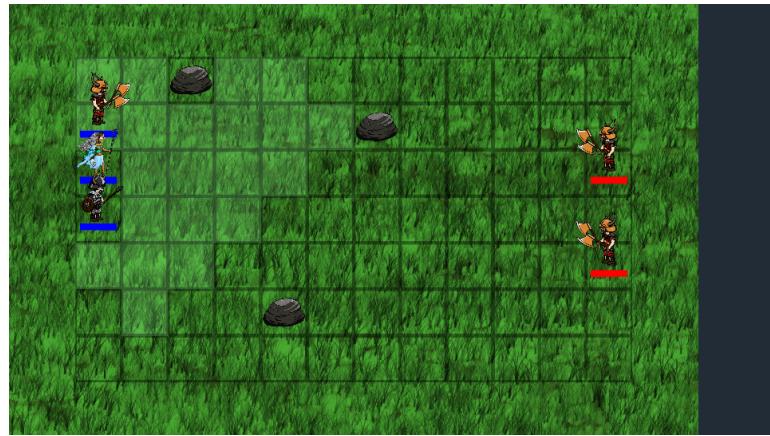


Abbildung 6: Kampfbildschirm in Unity

8.1.6 Unity/Kampf-Activity

In der auf der Karte aufgerufenen Kampf-Activity kann der von Unity kompilierte Code abgespielt werden. Um zwischen Java und Unity-Code zu kommunizieren verwenden wir Java Native Interface, kurz JNI. JNI ist eine Programmierschnittstelle, die es uns erlaubt, von Unity aus Java Funktionen aufzurufen, die sich in unserer Kampf-Activity befinden. Wir benutzen JNI, um die Kampfinformationen, die der Activity mitgeteilt werden, im Unity-Code abzurufen und um, nachdem der Kampf fertig ist, zurück zur MainActivity zu kommen.

8.2 Erstellung des Kampfes in Unity

In Unity werden Gameobjects als Basiseinheit für die Erstellung von Spielen verwendet. Gameobjects sind Behälter denen man verschiedene Komponenten wie Sprites und Skripte hinzufügen kann. Für die Steuerung unseres rundenbasierten Kampfes generieren wir in Unity ein Gitter aus Quadraten, die sich anklicken lassen und ihre Position im Gitter kennen. Damit der Kampf sich nicht immer gleich anfühlt, erzeugen wir zufällig auf unseren Quadraten Steine als Hindernisse, die Felder blockieren. Anschließend werden aus unseren sogenannten Prefabs, Schablonen aus denen man ein Gameobject instanziieren kann, die am Kampf beteiligten Spielfiguren erzeugt und an die Ränder des Spielfeldes gesetzt. Figuren sind nun reihum am Zug. Wenn eine Figur, die dem Spieler gehört an der Reihe ist, markieren wir alle Felder, zu denen sie sich bewegen kann. Wird eines dieser Felder angeklickt, generieren wir wie in 8.3 beschrieben einen Pfad, den unsere Figur dann abläuft. In unseren Prefabs sind die zugehörigen Animationen für die verschiedenen Befehle vorgemerkt. Wenn der Benutzer auf eine gegnerische Figur in Reichweite der aktiven klickt, wird diese angegriffen und der Zug beendet. Alternativ kann man die aktive Figur anklicken, um den Zug ohne Angriff zu beenden. Wenn die aktive Figur dem Gegner gehört, übernimmt eine simple künstliche Intelligenz (fortan AI genannt) die Entscheidungen, die sonst der Spieler trifft. Die AI überprüft, ob eine feindliche Figur in Reichweite ist und greift diese an, falls möglich. Wenn nicht, bewegt sie sich auf die nächste feindliche Figur auf dem Feld zu. Falls sie nun in Reichweite ist, greift sie die Figur an, andernfalls beendet sie ihren Zug.

8.3 A*-Pfadfindung

Um Figuren auf unserem Spielfeld zu bewegen, benötigen wir einen Suchalgorithmus, um einen Pfad zu finden. Wir haben dazu den sogenannten A*-Algorithmus implementiert. Im Gegensatz zu uninformierten Suchalgorithmen wie z.B. Dijkstra wird in A* eine Heuristik benutzt, um zielgerichtet zu suchen. Alle Felder werden in drei Gruppen unterteilt. Die erste Gruppe sind die noch nicht gefundenen Felder. Am Anfang sind dies alle außer dem Startfeld. Die zweite Gruppe sind die Felder, zu denen mindestens ein Weg bekannt ist. Diese werden, zusammen mit einem Wert $f(x)$, in einer Liste gespeichert. Der Wert von $f(x)$ ergibt sich aus der Summe der Kosten $g(x)$, um den Knoten zu erreichen und den geschätzten Kosten $h(x)$, um von dort zum Ziel zu kommen. Im Falle von unserem Spiel kostet es einen Punkt, um sich orthogonal und zwei, um sich diagonal zu bewegen. Die Schätzfunktion nimmt einfach die Summe der Abstände in x- und y-Koordinate zwischen einem Feld und dem Zielfeld. Zusätzlich wird für jedes Feld in dieser sogenannten Open List vermerkt, von welchem Feld ausgehend der momentane Weg dort ankommt. Die dritte Gruppe besteht aus den Feldern, zu denen bereits ein optimaler Weg gefunden wurde. Diese Felder werden in der sogenannten Closed List ebenfalls mit dem Feld, von dem der kürzeste Weg zu ihm ausgeht, gespeichert.

Zu Beginn befindet sich nur das Startfeld in der Open List und die Closed List ist leer. Nun wird in jedem Schritt das Feld a mit dem geringsten f -Wert aus der Open List in die Closed List aufgenommen.

Für jedes Nachbarfeld b wird nun $g(b)$ berechnet aus $g(a)+1$, falls es ein orthogonaler Nachbar, und $g(a)+2$, falls es ein diagonaler Nachbar ist. Zusammen mit der oben beschriebenen Heuristik wird nun $f(b) = g(b)+h(b)$ berechnet und b wird in die Open List mit a als Vorgänger aufgenommen,

falls es dort nicht schon mit einem gleichgroßen oder kleineren f-Wert vorhanden ist. Der Algorithmus endet, sobald das Zielfeld in die Closed List aufgenommen wird. Der Weg ergibt sich nun, indem man über die gemerkten Vorgänger die Felder bis zum Start zurückverfolgt. Zum besseren Verständnis folgt nun ein Beispiel für die Pfadfindung mit Bildern.



Abbildung 7: Der Schakalskrieger sucht einen Weg zum Ziel. Im ersten Schritt werden die Nachbarfelder mit ihrem f,g und h Werten in die Open List eingetragen.



Abbildung 8: Die Felder mit $f = 4$ wurden nun in die Closed List übernommen (grün markierte Felder). Für ihre Nachbarn wurde der f Wert berechnet und sie wurden in die Open List aufgenommen



Abbildung 9: Die kleinsten f-Werte aus der Open List wurden wieder in die Closed List aufgenommen.



Abbildung 10: Das Zielfeld ist jetzt in der Open List angekommen. Da es den kleinsten f-Wert hat sind wir im nächsten Schritt fertig.

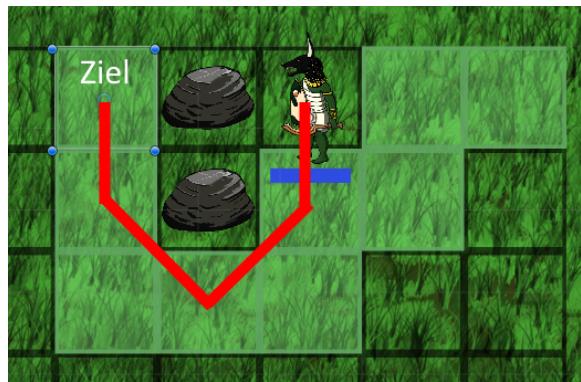


Abbildung 11: Das Ziel ist nun in der Closed List. Der Pfad ergibt sich nun dadurch, dass sich jedes Feld gemerkt hat von welchem Vorgänger aus der kürzeste Weg zu ihm führt.

8.4 Der Sieg-Bildschirm

Nachdem eine Seite keine Figuren mehr hat, endet der Kampf. Hat der Spieler verloren, ruft die Unity-Activity wieder die Karte auf und wird beendet. Hat der Spieler gewonnen, wird ihm das mitgeteilt und er bekommt Erfahrung und ein Item als Belohnung. Zur Darstellung dessen wird in Unity eine neue Scene aufgerufen und die alte verworfen. Der Siegesbildschirm liest aus dem Json-Objekt die notwendigen Daten aus und stellt den Erfahrungsfortschritt mit Balken dar, die neben den Namen und Gesichtern der Figuren angezeigt werden.



Abbildung 12: die erste Anzeige des Sieg-Bildschirm

Danach löscht Unity die Erfahrungsbalken, Namen und Gesichter und stellt das ebenfalls aus dem Json-Objekt ausgelesene Item dar. Nach sechs Sekunden wird die Unity-Activity beendet und die Karte wieder aufgerufen.



Abbildung 13: die Präsentation des Items

8.5 Grafik und Stil

Der Stil sollte sowohl auf Smartphones als auch auf Tablets gut aussehen, ein breites Publikum ansprechen und sich vom normalen etwas abheben.

Das hieß, dass es mehrere Probleme zu lösen gab:

1. Die größten Tablet-Bildschirme wie die vom Google Pixel C können über 25cm Bildschirmdiagonale haben, während unser Test-Smartphone z.B. gerade mal 11cm hat. Unser Spiel sollte also auf 13cm hohen Bildschirmen erkennbar sein, auf 30cm breiten Bildschirmen aber immer noch gut aussehen. Realistischere Darstellungen (Abb.14) waren auf dem Test-Smartphone nicht gut erkennbar (Abb.15), sodass wir uns für einen Comic-Look mit unrealistischen Proportionen entschieden haben. Dabei galt die Balance zu halten zwischen zu stark unterschiedlichen Proportionen, was insbesondere ältere Spieler häufig abschreckt, und Erkennbarkeit.



Abbildung 14: früher Entwurf eines Speerträgers

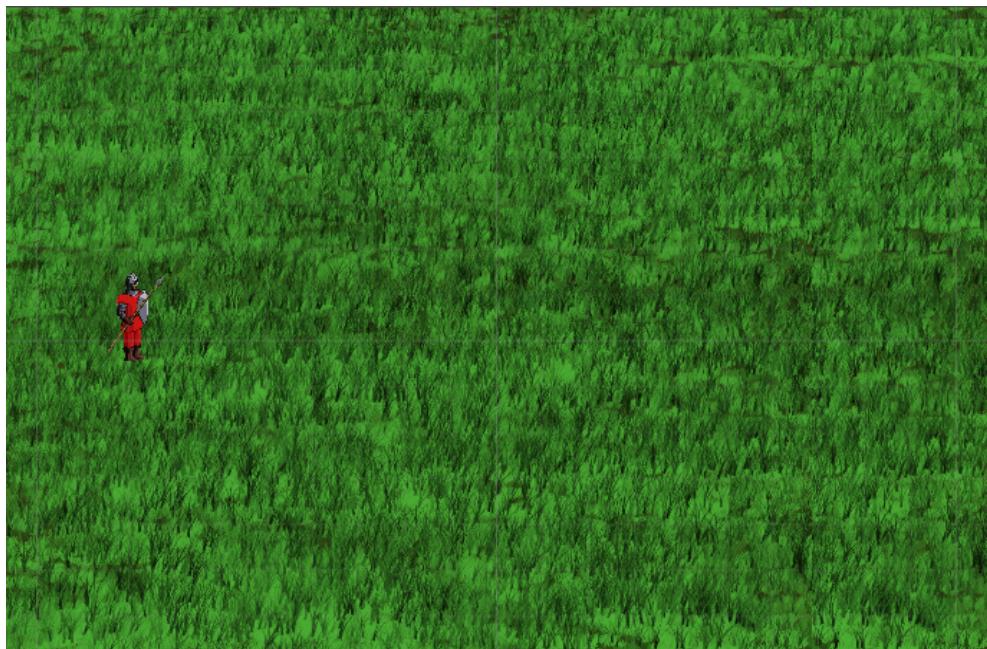


Abbildung 15: Speerträger auf unserem Schlachtfeld

2. Das Spiel sollte Erwachsene und Jugendliche anziehen, aber immer noch bedenkenlos von Kindern spielbar sein. Das hieß, dass der Grafikstil nicht zu düster sein durfte, aber auch nicht zu kindisch, ohne generisch zu wirken.
3. Die hohe Detaildichte, welche bei der Entwicklung für Tablets erforderlich ist, erfordert einen hohen Zeitaufwand. Folglich mussten Animationen vom Grundmodell aus einfach zu erzeugen sein und eine Ansicht gewählt werden, in der das nicht stark auffällt.

Die meisten Taktik-Rollenspiele haben sehr abwechslungsreiche und häufig eher zufällige Karten und teils zufällige Startpositionen mit verschiedenen Terrain-Höhen. Da allerdings unser Fokus stärker auf strategischen als zufälligen Kämpfen lag, entschieden wir uns bei dem Aufbau des Schlachtfeldes für ein flaches Feld mit Gegnern auf der einen und Spielern auf der anderen Seite, wie es in Strategie-Spielen wie Schach oder der Heroes-

Serie zum Einsatz kommt. Dies erleichterte auch das Grafikdesign, da Figuren sich hauptsächlich auf der horizontalen Achse bewegen, sodass eine 3/4-Ansicht mit situationaler Spiegelung für fast alle möglichen Lauf-Richtungen gut aussieht. Zusätzlich haben wir uns entschieden, das Feld breiter als hoch zu machen, was auf das Gameplay positive Auswirkungen hatte, aber auch bedeutete, dass zum einen das Schlachtfeld besser der 16:9-Auflösung der meisten Smartphones entspricht, zum anderen die optisch schlechter aussehenden Bewegungen nach gerade oben oder gerade unten seltener sind.

Danach wurden mehrere Skizzen erstellt, um eine geeignete Balance zwischen realistisch und übertrieben/Comic-Look zu finden. Erste Versuche (Abb.16) waren stärker überzeichnet und an den sogenannte Chibi-Zeichenstil angelehnt, da ein vergrößerter Kopf leicht auszumachen ist und eine gute Differenzierung verschiedener Figuren ermöglichte. Es gab allerdings Bedenken, dass dies Hardcore-Spieler abschrecken könnte, die wir ansprechen wollten. Daher wurde der Kopf etwas weniger überzeichnet, der restliche Körper etwas normaler proportioniert und zusätzlich die Bewaffnung etwas vergrößert. Da geplant war, einige Figuren umzufärben, einige Details abzuändern und diese dann als neues Modell zu benutzen, erlaubte eine größere Waffe eine gute Erkennbarkeit von Elementen, die für Statuswerte wichtig sein sollten. So sollte es für den Spieler intuitiv verständlich sein, dass die Figur mit dem Schild mehr Schaden aushält und die Figur mit den zwei Äxten mehr Schaden macht, was bei purem Umfärben häufig ein Problem darstellt.

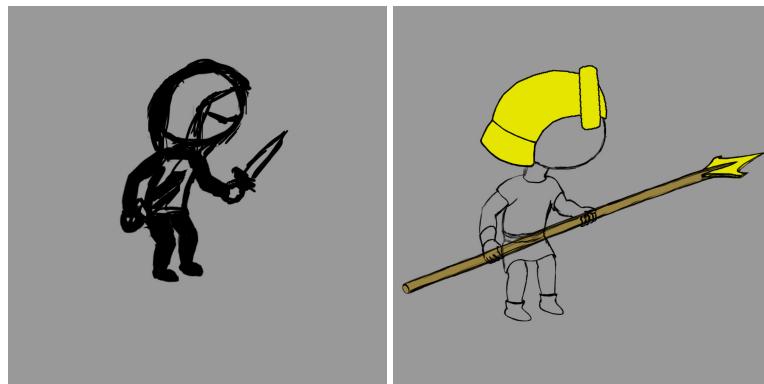


Abbildung 16: frühe Entwürfe

8.6 Die Figuren



Abbildung 17: der finale Wikinger

8.6.1 Der erste Wikinger

Da die klassischen Fantasy-Skizzen zu kindlich oder zu generisch gerieten, wurde bei den Motiven zunehmend weniger Mainstream-Motive verwandt. So wurde sich für ein Wikinger-Motiv entschieden, was es erlaubt.

te, eine Figur düster und trotzdem niedlich aussehen zu lassen. Außerdem sind die gehörnten Helme in Medien ikonisch (wenn auch historisch falsch), sodass Spieler auch auf sehr kleinen Bildschirmen erkennen sollten, dass es sich um einen Wikinger handelt.

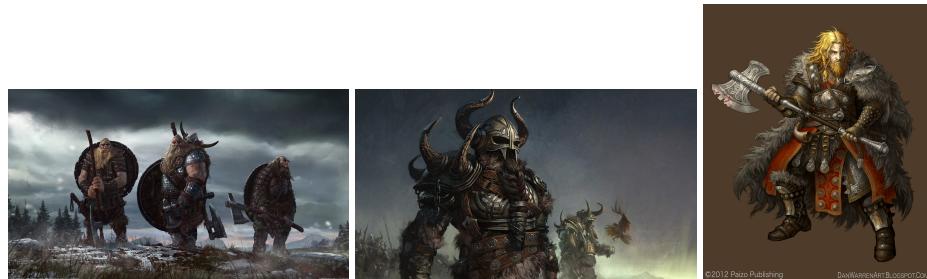


Abbildung 18: Vorbilder für den finalen Wikinger

Wikinger sind seit dem 18. Jahrhundert als noble Wilde verklärt, sie werden aber seit dem frühen 20. Jh. auch häufig als heidnische, brutale Piraten dargestellt. Attribute, die ihnen häufig zugeschrieben werden sind Wildheit, Unabhängigkeit, Kraft, Aggressivität. Darstellungen zeigen sie meistens als Krieger oder Seefahrer, oft mit Äxten und bemalten Holzschilden. Ihre Kleidung setzt sich in modernen Darstellungen aus mehreren Lagen Kleidungen oder Pelze über leichten Rüstungen wie Kettenhemden zusammen (was historisch akkurat ist), dazu die ikonischen gehörnten Helme (die historisch falsch sind). In fantastischeren Darstellungen tragen sie oft mehrere Gürtel und Medallions und gelegentlich Lamellenpanzer (Metallplättchen, die auf Leder aufgenäht werden). Wikinger tragen in modernen Darstellungen nahezu immer lange Haare und Bärte, beides häufig zu Zöpfen geflochten.

Dementsprechend setzte ich den ersten Wikinger (Abb.17), der es in das finale Spiel schaffte aus mehreren dieser Elemente zusammen. Er trägt einen auffälligen gehörnten Helm, eine Axt und ein hölzernes Rundschild. Die

Axt und der Helm sind bewusst vergrößert, um auch auf einem Handy-Bildschirm klar sichtbar zu sein. Die Rüstung setzt sich aus mehreren Gürtel mit Metallplättchen über einem roten Wams zusammen und wird durch Lamellenpanzer über Schulter und Waffenrock komplementiert. Die Detaildichte der Rüstung stört nicht auf dem Smartphone, da Axt, Schild und Helm den Wikinger klar erkennbar machen, sieht aber auf Tablets besser aus. Durch die Gürtel und Medallions setzt sich die Rüstung klar von "normaler" mittelalterlicher Kleidung und Rüstung ab und der Lamellenpanzer lässt ihn kriegerisch aussehen. Zusätzlich trägt er einen Pelz um die Schultern und auf den Stiefelrändern, sowie eine aus Pelz gemachte Hose, damit er etwas Wildes hat. Die Stiefel haben im Stile von Beinschienen einen metallenen Clip, um interessanter auszusehen. Dabei wurden Ränder der Figur bewusst schwarz und die Schatten mit geringen Farbverlauf gelassen, um die Figur vom Hintergrund abzuheben und einen Comic-Eindruck entstehen zu lassen. Später ist die Figur unser Avatar Essling auf der Karte geworden.

8.6.2 Der Morgenstern-Wikinger



Abbildung 19: die kühle Kolorierung mit dem Morgenstern

Da sich im Rahmen der Kolorierung des ersten Wikingers stärker mit Farblehre auseinandergesetzt wurde, entstanden danach zwei Umfärbungen, eine mit warmen und eine mit kalten Farben, unter Anderem mit dem Ziel, sie optisch stärker voneinander abzusetzen. Die Version mit den kühlen Farben bekam einen Morgenstern, eine Waffe des Spätmittelalters, um zwar aggressiv aber spezialisierter zu wirken, also bedachter. Dieser Eindruck wird verstärkt durch den weißen Bart und die generell kühlen Farben, da kühle Farben häufig mit Rationalität in Verbindung gebracht wird.



Abbildung 20: Der Berserker

8.6.3 Der Berserker

Die bekannteste Figur, die mit Wikingern assoziiert wird ist der Berserker, ein Krieger der sich in einen Kampfrausch versetzte. Dieser kommt in vielen Spielen vor, nicht nur in Fantasy-Wikinger-Varianten wie den Chaos-Berserkern des populären Tabletops Warhammer, auf dem einige Spiele basieren, sondern auch in eher historisch angehauchten Titeln wie dem bekannten Strategie-Spiel Age of Empires 2 und in mehreren Teilen der hochgelobten Civilization-Reihe. Berserker werden häufig als blutrünstig und wahnsinnig dargestellt. Dementsprechend wurde die warme Kolorierung der Berserker mit viel rot erzielt, da rot häufig mit Aggression in Verbindung gebracht wird. Er bekam zwei Äxte, um aggressiver zu wirken und der Helm wurde nach einem Drachenkopf modelliert um die rote Assoziation mit Feuer zu verstärken und ihn stärker von den anderen beiden abzuheben. Es wurde sich dagegen entschieden, den originalen Wikinger in die finalen Kämpfe aufzunehmen, da er einen Mittelweg zwischen beiden darstellte und denkbar war, dass es zu Verwechslungen kommen könnte.



Abbildung 21: Die Priesterin des Ra

8.6.4 Die Priesterin des Ra

Nach den Wikingern brauchten wir einen starken Kontrast, um das Spiel optisch abwechslungsreich und interessant zu halten. Mit dem ägyptischen Motiv, mit dem vorher schon gespielt wurde, war ein geeignetes Motiv gefunden, da es eher selten in Spielen verwendet findet und sehr gegensätzlich zum nordischen ist. Während Wikinger aufgrund ihrer militärischen Expansion und Piraterie im Mittelalter mit roher Kraft, Militarisierung und Barbarei verbunden werden und Unabhängigkeit und Seefahrt eine große Rolle spielen, waren die Ägypter eine spirituelle, künstlerische Hochkultur die über Jahrtausende wenig expandierte und stark hierarchisch aufgebaut war. Außerdem sind die optischen Unterschiede stark, bedingt u.A. durch die unterschiedlichen Klima-Bedingungen und den Zeitabstand.

Die Entwicklung der Priesterin des Ra dauerte aus mehreren Gründen ungewöhnlich lange und war komplex. Nach den Wikingern gab es eine Reihe Konzepte, die abgedeckt werden sollten. So sollten die Ägypter aufgrund ihrer kulturellen Unterschiede mystischer aussehen und sich weniger direkt spielen lassen, Konzepte sahen neben einem Nahkämpfer einen

Fernkämpfer vor, der möglicherweise göttliche Kräfte nutzte. Außerdem sollten die Figuren um eine weibliche Figur erweitert werden, sowohl um weiblichen Spielern eine Identifikationsfigur zu geben, als auch aus Gründen optischer Abwechslung. Die Idee war ursprünglich, einen Falken-Krieger als Nahkämpfer zu zeichnen und da die Waffe bereits vergrößert war, bot sich an, eine andersartige, auffällige Waffe zu konzipieren. Da die Bewaffnung der Wikinger eher realistisch gehalten wurde (abgesehen von der Größe), sollte die Waffe allerdings nicht zu fantastisch sein. So wurden verschiedene Konzepte skizziert, zunehmend als eine am Arm befestigte Waffe nach dem Vorbild einer Klaue oder eines Schnabels, bis die Entscheidung getroffen wurde, den Arm selbst zu einer Vogelklaue zu machen. Die Ägypter stellten ihre Götter mit Tierköpfen dar, was immer wieder Vorbild für moderne Webkunst ist, ein anderes Körperteil abzuwandeln war also naheliegend. Da die Figur damit angreifen können sollte und mit Flügeln und Klauen den Harpyien (ein griechisches Fabelwesen mit Vogelkörper und Frauenkopf, häufig mit weiblichem Oberkörper abgesehen von den Flügeln) zu ähnlich wäre, wurde entschieden, einen Arm durch eine Klaue zu ersetzen.

Danach wurde die Entscheidung getroffen, die Figur weiblich zu machen. Eine zierlichere Figur würde einen stärkeren Kontrast zu der monströsen Vogelklaue darstellen und konnte damit sowohl tragisch als auch monströs sein, was es leichter machte, sie auf der Seite des Spielers aber auch auf Seite des Gegners zu erklären. Das bedeutete aber, dass die Proportionsierung neu erstellt werden musste, die Figur musste insgesamt schmäler, aber weiterhin gut erkennbar sein, d.h. Kopf und Arm müssten weiterhin groß und präsentieren sein, ohne falsch zu wirken. Zum anderen bedeutete das ägyptische Motiv mehr komplexe Materialien, die zu zeichnen waren,

insbesondere Muskulatur, beweglicher Stoff und Goldschmuck. Die populärsten ägyptischen Vogelgötter sind Horus, der Gott des Himmels und Ra, Gott der Sonne. Ich entschied mich für Ra, da bereits angedacht war, die Priesterin zu einer Fernkämpferin zu machen und Elemente des Himmels wie Blitze schwerer umzusetzen gewesen wären als z.B. ein Feuerball. Die Priesterin bekam eine Vogelmaske, ein Konzept das noch aus der Zeit des Falken-Kriegers existierte und eine einfache Möglichkeit darstellte, das Gesicht interessanter zu machen. Die Maske wurde mit zusätzlichen Federn versehen, um den Vogel-Eindruck zu verstärken. Außerdem bekam sie Bänder aus lila Stoff mit aufgesetztem Gold nach dem losen Vorbild der Todesmaske von Tutanchamun, die einem von Pharaonen getragenen Kopftuch (Nemes) nachempfunden ist. Auf der Stirn wurde eine große rotgoldene Platte als Symbol für die Sonne gezeichnet, eine Abwandlung der historischen Darstellungen von Ra, der oft mit einer gewaltigen roten Scheibe über seinem Kopf abgebildet ist.

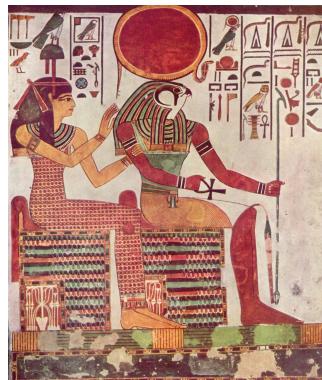


Abbildung 22: historische Darstellung des Ra mit Sonnenscheibe

Die Augen wurden golden eingefärbt, um darzustellen, dass sie aus den Augen leuchtet. Die Haare sollten im Wind wehen und auffällig sein, daher weiß, sehr lang und mit vielen Strähnen. Um den Hals zeichnete ich den auf

Pharaonen- und Götterdarstellungen basierenden typisch ägyptischen goldenen Halsschmuck. Der Vogel-Arm wurde mit Federn versehen und die Klaue der eines realen Falken nachempfunden. Darüber wurde ein zerissener Ärmel aus Stoff gezeichnet, da dies den kritischen Übergang zur Schulter zu erleichterte, flatternder Stoff gut mit der Vogel-Assoziation funktionierte und weil es wahrscheinlich wäre, dass eine reale Person den Arm verstecken wollen würde. Ursprünglich war für die Priesterin ein einärmiges Kleid geplant, aber nachdem verschiedene Konzepte für den Oberkörper ausprobiert wurden und sich für den goldenen geflügelten Skarabäus im Brustbereich entschieden wurde (ein weiteres Symbol für Ra, das ebenfalls ägyptisch und bekannt ist), wurde das Kleid verworfen und sich für einen Rock und Bandagen entschieden und der Ärmel am Hals schmuck "befestigt". Der rechte Arm war simpler, als Stab wurde sich für eine verlängerte Version des Stabs der Könige entschieden und der Armschmuck dem auf ägyptischen Darstellungen von Göttern, Göttinnen und Pharaonen nachempfunden. Der Gürtel sollte wie gewebtes Gold aussenhen, um trotz Gold noch leicht auszusehen und brauchte wegen der Größe ein auffälliges Kopfteil. Der Rock bestand erneut aus flatterndem, Himmel farbigem Stoff.



Abbildung 23: Der finale Schakalskrieger

8.6.5 Der Schakalskrieger

Nach der Priesterin des Ra wollte ich eine etwas weniger fantastische Figur, die sich stärker an der historischen Realität orientierte und martialischer aussah. Da mit der Priesterin eine ägyptische Fernkämpferin existierte, war für die nächste ägyptische Figur ein Nahkämpfer vorgesehen. Erste Überlegungen beinhalteten einen Streitwagen, einen Pharao und einen normalen Soldaten. Da die Einheit in Massen auftreten können sollte, fiel der Pharao weg und der Streitwagen wäre wahrscheinlich so groß geworden, dass er andere Einheiten verdeckt hätte. Daher orientierte ich mich am historischen Soldaten des dritten Zeitalters.



Abbildung 24: Der originale Entwurf

Der Soldat war anfangs als Mensch gezeichnet und trug ein mit einer Schlange verziertes, vorher erwähntes Nemes als Kopftuch. Kopftücher dieser Art waren als Schutz gegen die Hitze recht normal und sahen interessanter aus, als nur Haare. Die Schlange wurde hinzugefügt (als Zeichen der Königswürde von Totenmasken bekannt, wird es häufig mit Ägypten in Verbindung gebracht), um den Kopf interessanter aussehen zu lassen. Zusätzlich zeichnete ich eine mehrschichtige Rüstung aus Leinen, inklusive eines auffälligen Lendenschurzes, den ich so oder ähnlich auf einigen Bildern gefunden hatte. In der Realität trugen Ägypter wegen der Hitze lange keine oder wenig Rüstung, Leinen wurde allerdings u.A. von Kriegern in Streitwagen benutzt. Das mit Kuhfell bespannte Holzschild war auffällig und von der Form her exotisch, da es allerdings sehr groß war, zeichnete ich es anders als bei den Wikingern auf die vordere Hand, da es sonst zu viel vom Körper verdeckt hätte. Als Waffe wurde ein sogenanntes Khopesh ausgewählt, eine exotische Waffe, die Eigenschaften einer Axt und eines Schwertes kombiniert und in der Moderne häufig mit Ägypten assoziiert wird. Sie war lange (aber nicht ausschließlich) eine Zeremonie-Waffe der Pharaonen und wurde etwas vergrößert gezeichnet. Zusätzlich trug der Krieger ähnlichen, aber massiveren Schmuck um Arme und Hals

wie die Priesterin, zum einen um ein typisches ägyptisches Element unter meinen Figuren herzustellen, zum anderen weil es wie oben erwähnt für viele Menschen typisch ägyptisch ist.

Um die Figur interessanter aussehen zu lassen, wurde der Kopf gegen einen Schakalskopf ausgetauscht, der Anubis nachempfunden war. Der ägyptische schakalsköpfige Gott der Einbalsamierung Anubis ist sehr bekannt und der Kopf gab dem Krieger ein aggressiveres Aussehen, etwas das dem originalen Entwurf gefehlt hatte. Nachdem mehrere Skizzen mit unterschiedlichem Kopfschmuck und Haaren erstellt wurde, wurde sich für schwarze Rhasta-Locken mit goldenen Bändern entschieden. Dies sorgte für einen außergewöhnliches Äußeres und funktionierte gut mit dem anderen Goldschmuck. Da hier, anders als bei der Priesterin, der Arm mit dem Reif im Vordergrund steht, wurde die Armschiene stärker ausgearbeitet. Die Haut wurde dunkler gemacht, was besser zu dem Kopf passte und einen starken Kontrast zu dem beigen Leinen erzeugte.

8.7 Animationen

2D-Animationen funktionieren sowohl in Spielen, als auch in Filmen im Allgemeinen nach dem Daumenkino-Prinzip: Wenn man eine gewisse Zahl an Bildern (Frames genannt) pro Sekunde aneinander hängt, die Momente einer Bewegung zeigen, entsteht bei dem Betrachter das Gefühl einer Bewegung. Hierbei sind 24 Frames pro Sekunde optimal und kommen z.B. in Disney-Filmen zum Einsatz, zwölf sind auch für Menschen mit guten Augen nicht als einzelne Bilder erkennbar und sechs reichen, um den Eindruck von Bewegung entstehen zu lassen und kommen häufig bei Projekten mit geringem Budget, wie z.B. in den meisten 2D-Indie-Spielen zum Einsatz.

Da wir mit sechs Monaten etwas unter Zeitdruck standen, sind die Animationen sechs bis zwölf Frames lang. Insgesamt sind die Animationen eher simpel gehalten, teilweise wurden sie allerdings aufpoliert, z.B. indem Licht und Schatten korrigiert, oder Muskulatur und sich bewegende Elemente wie Stofffalten neu gezeichnet wurden.



Abbildung 25: Ein Spritesheet für eine Laufanimation

Unsere Figuren bekamen Lauf- und Angriffsanimationen, wobei die Laufanimationen im Spiel solange wiederholt werden, bis die Figur ihr Ziel

erreicht hat und die Angriffsanimationen einmal ausgeführt werden. Die Animationen wurden in sogenannten Spritesheets gespeichert, die in Unity dann wieder in einzelne Bilder zerlegt werden. Spritesheets sind mehrere Bilder, die in einem Bild gespeichert werden, sodass nicht jedes Bild einzeln geladen werden muss. Unity ist in der Lage, aus mehreren Bildern automatisch Animationen von einer Sekunde Dauer zu generieren, wobei an einigen Stellen die Animationen beschleunigt oder verlangsamt wurden, um bessere Ergebnisse zu erzielen. So wurden z.B. diverse Angriffsanimationen in eine langsam abgespielte Aushol-Phase und eine schneller abgespielte Schlag-Phase aufgeteilt. Die Anbindung der Animationen an den Code geschieht dabei mithilfe des sogenannten Animator Controller, eines Zustandsautomaten, der für verschiedene Auslöser Animationen wechselt oder abspielt.

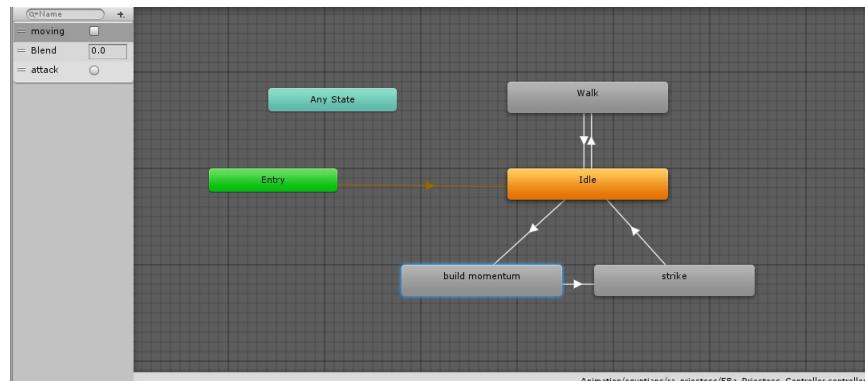


Abbildung 26: Unity's Animator Controller



Abbildung 27: Ein Spritesheet für eine Angriffsanimation