Realisierung eines universitären Games auf Basis der Unreal 4 Engine

Gavin Barnes, Jan Brinkmann, Anh Minh Nguyen, Eugenie Rerich, Sina Mostafawy
Medieninformatik
Hochschule Düsseldorf
Josef – Gockeln Straße 9
Tel.-Nr.:
Email:

Alex Ostrowski
Concept Art
Hochschule Düsseldorf
Josef – Gockeln Straße 9
Tel.-Nr.:
Email:

Abstract: In diesem Beitrag beschreiben wir die Entwicklung eines Spiels und legen hierbei den Schwerpunkt auf dem Weg von einer anfänglichen Idee zum fertigen Konzept. Es gibt viele verschiedene Ansätze zur Spieleentwicklung. Einige davon werden hier näher betrachtet und bei der Spieleentwicklung berücksichtigt. Trotzdem ist es nicht möglich ausschließlich einem Ansatz zu folgen und somit stellt dieser Beitrag gleichzeitig auch einen eigenen Lösungsansatz für kleine Entwicklergruppen dar. Das Vorgehen wird beschrieben und ein aktueller Stand der Entwicklung wird vermittelt, sowie ein Ausblick auf die noch ausstehenden Aufgaben gegeben. Die Entwicklung des Spiels erfolgt im Kontext der Bachelorarbeiten der meisten Beteiligten.

Keywords: Modellierung, Animation, Game Design, Concept Art, Unreal 4

1 Einleitung

Die Entwicklung von Spielen blickt auf eine lange Tradition zurück, zumal es Spiele schon immer gab. Alle diese Spiele folgen bestimmten Regeln und in ihrer Entstehung liegt ein bestimmtes Konzept zugrunde, unabhängig davon, ob es sich hierbei um ein Computerspiel, ein Brettspiel oder andere Spielarten (Fußballspiel, etc.) handelt. Das Konzept eines Spiels kann noch so primitiv oder kompliziert sein, ob es erfolgreich sein wird, hängt von einer positiven Spielerfahrung ab. Das Ziel jeder Spieleentwicklung sollte also sein, dem Spieler möglichst eine positive Spielerfahrung zu ermöglichen[…]. Nun gibt es allerdings auch innerhalb eines Obergenres wie dem des Computerspiels zahlreiche verschiedene Untergenres. Da die riesigen Unternehmen der Spielindustrie natürlich möglichst viel Profit mit dem Spiel machen wollen, sind viele Spielkonzepte sehr ähnlich, oft mit demselben Spielprinzip und lediglich mit einem

Unterschied in Story und Design[…]. Die Story ist meistens jedoch sehr oberflächlich und das Design richtet sich nach bestimmten Stereotypen. Spiele, die dieser Klassifizierung nicht folgen und ein innovatives Spielprinzip oder – design aufweisen, sind dann meistens Indie Games von kleineren Unternehmen oder Gruppen. Mit diesem Projekt haben wir auch versucht ein innovativeres Spielkonzept in einer verhältnismäßig kurzen Zeit zu entwickeln. Ziel des Beitrags ist die Beschreibung eines noch laufenden Projekt, in dem ein Spiel entwickelt wird, welches in seiner Idee und seinem Concept Art stark stilisiert ist. Nicht nur das Design soll sich von den typischen Mainstream Games unterscheiden, sondern auch die Implementation einer spezifischen Spielmechanik führt zu einem andersartigen Spielerlebnis. Das Spiel selbst ist eine Art Hybrid aus Puzzle Game und Adventure Game, d.h. der Spieler navigiert durch die Umgebung und sieht sich mit Rätseln konfrontiert, die er lösen muss, um weiter im Spiel voran zu schreiten. Auf seinem Weg lauern Gefahren, die ihn davon abhalten können und er hat Helfer, die ihm helfen diese Gefahren zu überwinden und die Rätsel zu lösen. Durch das Voranschreiten im Spiel erfährt man auch immer mehr über die Hintergrundgeschichte des Characters und über die Beschaffenheit seiner Umgebung. Der Beitrag fokussiert sich hier vor allem auf dem Weg von der Idee zum Spiel und seiner Realisierung in der Unreal 4 Engine. Er vermittelt einen Eindruck darüber, wie das Spiel im weiteren Verlauf seiner Entwicklung aussehe könnte. Die Entwicklung dieses Spiels verläuft im Rahmen mehrerer Bachelorarbeiten und wissenschaftlicher Vertiefungen, wobei jeder Teilnehmer seinen eigenen Fachbereich hat, für den er zuständig ist. Zunächst beschreibt dieser Beitrag jedoch die Grundlagen des Game Designs und verwandte Arbeiten, während im Folgenden dann das Konzept als solches und der aktuelle Prototyp vorgestellt werden.

2 Verwandte Arbeiten

Zum Thema Game Design gibt es viel Literatur. Oft ist es jedoch so, dass diese Literatur lediglich beschreibt, wie man die Entwicklung eines Spieles angehen sollte. Damit wird ein Tutorial bereitgestellt, welches das Vorgehen bei der Spieleprogrammierung erläutert. Ähnlich verhält es sich bei der wissenschaftlichen Recherche nach der aktuellen Spieleengine. Viele der wissenschaftlichen Quellen beschäftigen sich entweder mit Guidelines für den Entwickler […] oder die Rolle der Engine wird im Kontext von anderen Bereichen, oft Serious Games, diskutiert[…]. Dieses Projekt beschreibt jedoch die Entwicklung eines unterhaltungsbasierten Spieles mit der Unreal Engine und stellt somit einen etwas anderen Aspekt in den Vordergrund.

3 Beitrag

Es gibt zahlreiche Literatur darüber, wie man die Entwicklung eines Spiels am besten angehen sollte. Gerade, weil auch immer mehr Studiengänge zu diesem Fachbereich entstehen, die sich explizit nur mit Game Design auseinandersetzen, existieren viele Annäherungen an dieses Thema. Die meisten Ansätze sind sich jedoch darin einig, dass eine bestimmte Organisationsstruktur in der Entwicklung eines Spieles vorhanden sein muss. So lässt sich die Entwicklung eines Games in drei Phasen unterteilen [P07].

Die Vorproduktionsphase umfasst alles, was das Konzept und das Design anbetrifft, insofern ist dies die Phase, um die es in diesem Beitrag hauptsächlich geht. Zuerst wurden mithilfe von Brainstorming verschiedene Ideen in unterschiedlichste Richtungen gesammelt. Es stellte sich schnell heraus, dass es in eine ästhetisch stark stilisierte Richtung gehen sollte und das das Spiel ins Genre des *Art Driven Designs*[RA03] gehören würde. Im Gespräch mit dem Betreuer des Projekts legte man sich dann auf die Kernidee fest. Zu den essentiellen Bestandteilen des Game Designs gehört auch das Aufsetzen eines *Design Documents*. Obwohl das Design Document in verschiedenen Ansätzen unterschiedlich definiert wird und teilweise noch unterteilt wird in verschiedene Dokumenttypen, lässt es sich zusammenfassend als eine Art Referenz beschreiben, in dem alle wichtigen strukturellen und konzeptuellen Ideen des Spiels zusammengefasst sind[RA03]. Dies wurde organisatorisch über ein Spieleigenes Wiki gelöst. Anhand der Kernidee und der Inspirationen für das Spiel fand ein Briefing mit dem Concept Artist statt, wobei die Zusammenarbeit größtenteils über die Kollaborationsplattform *Trello* und über regelmäßige Meetings und Briefings geregelt wurde.

Character Design

Die Priorität lag erstmal in der Konzepterstellung des Characters. In dem Briefing des Concept Artist wurden dann die Grundidee und Inspirationen für das Spiel vorgestellt. Aus den ersten Entwürfen des Concept Artist, wurde dann das grobe Aussehen des Characters festgelegt. In einem iterativen Design Prozess [SZ04], welcher darin bestand, dass der Concept Artist

Entwürfe vorlegt, welche dann im Team besprochen wurden und anhand derer neue Entwürfe entstanden, wurde dann immer mehr das finale Design des Characters entwickelt.

Der Zeitaufwand bis zur Fertigstellung des finalen Character Concepts betrug in etwa 4 Wochen. Für die Animation des Characters wurden dann Animation Cycles auf Basis der geplanten Steuerungsmöglichkeiten entwickelt.

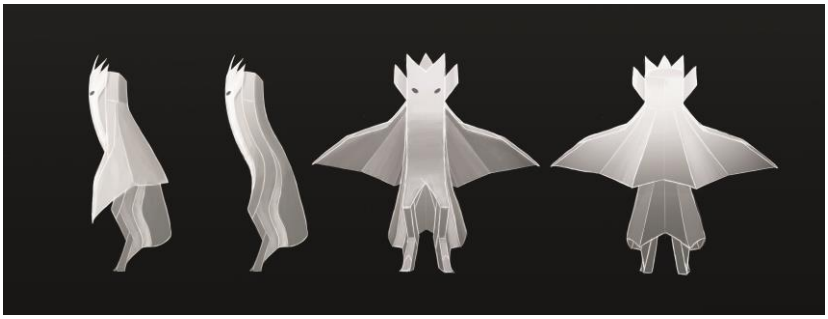

Abbildung 1: Finales Character Concept Art aus verschiedenen Perspektiven (Design by Alex Ostrowski)

Story

Die Story selbst wurde um den Character herum aufgebaut, d.h. es wurden Kernbedingungen für das Spiel erstellt:

- Die Welt des Spiels besteht aus Papier ebenso wie die Hauptfigur
- Die Welt ist weiß
- Mithilfe einer besonderen Spielmechanik kann der Character durch RGB – Anteile Sachen in der farblosen Welt sichtbar machen die sonst nicht zu sehen sind

Aus diesen Kernbedingungen wurden dann folgende Fragen aufgestellt:

- Wieso sollte man versuchen Dinge durch die Spielmechanik sichtbar zu machen?
- Wie würde diese Spielmechanik im Spiel eingesetzt werden?

Die Beantwortung dieser Fragen resultierte in dem Konzept der Orbs, einer Art Naturgeist, der zutiefst mit der Papierwelt verbunden ist. Die Orbs kommen zum Character Gami, um ihn um Hilfe zu bitten, da die Papierwelt von einer dunklen Gefahr bedroht wird. Um die Papierwelt zu retten, leihen sie Gami ihre Macht die Dinge durch Farben sichtbar zu machen. Auch hier stellten sich wieder eine Reihe von Fragen, die in der Entwicklung einer Backstory für den Character und die Orbs resultierten. Gami wurde dadurch zum Nomaden, der durch die unendlichen Weiten der Papierwelt wandert. Die Orbs haben sich deswegen Gami ausgesucht, um ihre Papierwelt zu retten, weil er ein Wesen ist, welches die Papierwelt und ihre Reinheit respektiert. Die Orbs selbst können die Papierwelt nicht retten, da sie zu schwach sind und sich

schnell fürchten.

Jetzt musste man nur noch definieren, wodurch die Papierwelt bedroht wird. Zu diesem Zweck wurde eine Sammlung darüber aufgestellt, was Papier in der Realität zerstören könnte. Die Papierwelt wird charakterisiert durch ihre Reinheit und deswegen wurden nicht nur Elemente betrachtet, die die Papierwelt vollends auslöschen (bspw. Feuer), sondern auch Elemente, die die Charakteristiken der Papierwelt ändern. So ergab sich das Feindbild der Tinte. An sich ist Tinte nicht schädlich für Papier, aber da die Papierwelt als rein beschrieben wird,

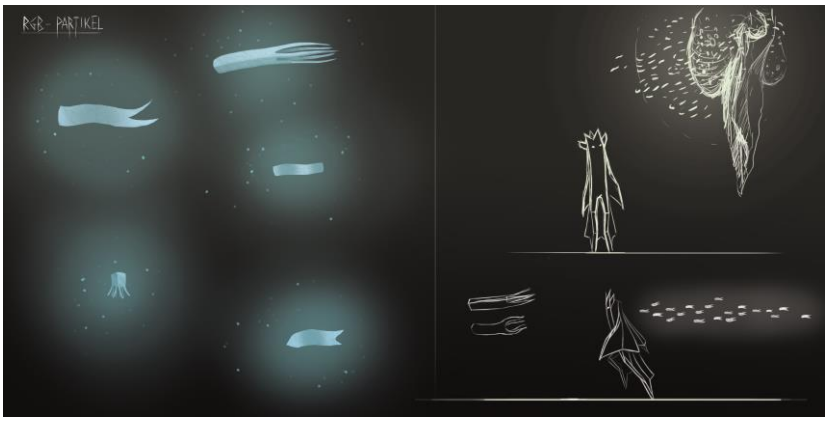

Abbildung 2: Concept Art der Orbs - Festlegung auf das Design links oben (Design by Alex Ostrowski)

stellt das Benutzen von Tinte eine ‚Verschmutzung' dar. Zu viel Tinte hat außerdem den Effekt von Wasser: Es macht das Papier wellig und schwer. Der Gedanke für die Tinte als Feindbild wurde außerdem dadurch inspiriert, dass interessante Effekte im Design des Spiels entstehen könnten. Auch die Tinte bedurfte einer Hintergrundgeschichte: Sie lebt einsam in der Tintenwelt und entwickelt Neid gegenüber der Reinheit der Papierwelt. Tinte kann nicht durch das Attribut der Reinheit charakterisiert werdend und deswegen beschließt die Tinte die Reinheit der Papierwelt zu zerstören. Während Gami als Bestandteil der Papierwelt ein kantiges bzw. faltiges Design hat, muss die Tinte als sein Gegenspieler etwas flüssiges, nicht statisches besitzen. Das Design der Tinte ist zum aktuellen Zeitpunkt noch in Bearbeitung.

Level Design

Das Level Design führte insbesondere zu 3 Fragestellungen:

- Wie kann der Spieler am einfachsten (ohne ausführliche Erklärungen) in das Spiel eingeführt werden?
- Welche Hindernisse begegnet er auf seinem Weg und wie können diese gelöst werden?
- Wie wird ersichtlich, dass sich die Papierwelt in Gefahr befindet?

Aus diesen drei Fragestellungen ergab sich die Einteilung in drei Story Arcs: Einem Einführungs - Arc, in dem der Spieler mit der Spielmechanik vertraut gemacht wird, einem weiter ausgearbeiteten Rätsel – Arc und einem Endkampf – Arc. Des Weiteren wurde festgelegt, dass nach oder innerhalb der Arcs einige Sequenzen vorkommen werden, die die

Story näher beschreiben, sowie eine Startsequenz. Jedes Arc bis auf das Endkampf – Arc soll aus je drei Levels bestehen.

Im ersten Arc wird die RGB – Mechanik der Orbs eingeführt, d.h., dass es je ein Level gibt, in dem der Spieler die grüne, die blaue und die rote Funktionalität der Orbs kennen lernt. Jede dieser Farben hat eine eigene Funktion, die dem Spieler im ersten Arc vermittelt wird. So werden mit dem Benutzen der grünen Farbe Objekte sichtbar gemacht, die statisch sind. Mit der blauen Farbe werden Objekte sichtbar gemacht, die man bewegen kann und mit der roten Farbe werden Gefahren sichtbar gemacht. Damit die Farben mehr zur Geltung kommen, wurde das Setting des Spiels in eine Höhle der Papierwelt verlegt, die je weiter man zur Tinte vordringt immer dunkler wird, so dass man schließlich nur auf die Orbs und mögliche Geräusche angewiesen ist. Mithilfe von *Mood Paintings*, Bildern, die die allgemeine Stimmung des Spiels vermitteln, wurde erstmals die Funktionalität der Orbs visualisiert.

Abbildung 3: Mood Painting zur Visualisierung des Orb - Konzeptes und der Spielstimmung (Design by Alex Ostrowski)

Das erste Arc ist dabei so aufgebaut, dass der Spieler lernt[…], d.h. die ersten Orbs trifft der Spieler ganz automatisch ohne, dass er irgendetwas machen muss. Die Orbs zeigen ihm die Funktionalität der grünen Farbe (siehe Abb. 3) und heften sich an den Character. Für das Kennenlernen der blauen und roten Farbe muss der Charakter dann die grünen Orbs einsetzen. Das User Interface soll dabei ganz minimal gestaltet werden (es wird nur eine Taste angezeigt), um den ästhetischen Charakter des Spiels nicht zu stören.

In einem Meeting wurden dann die konkreten Herausforderungen, die der Charakter antrifft besprochen. Es wurden mögliche Wege konkretisiert, die der Spieler einschlagen kann und festgelegt, wann der Spieler ‚stirbt'. Die Einteilung in drei Arcs erwies sich als sehr umfangreich, sodass die genauen Hindernisse in Arc 2 und der Weg zum Endgegner in Arc 3 zwar festgelegt sind, aber noch unklar ist, ob im Rahmen dieses Projekts wirklich alle Arcs bis zum Ende ausgearbeitet werden können. In jedem Fall ist eine Weiterführung des Projekts auch außerhalb der Bachelorarbeiten geplant.

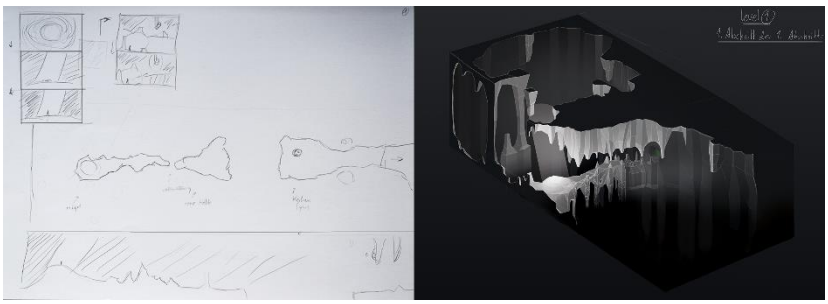

Abbildung 4: Darstellung des 1. Levels als Skizze mit Hindernissen und dem ausgearbeiteten Concept Art

Environment Design

Das Setting des Spiels wurde auf eine Höhle in der Papierwelt festgelegt. Dementsprechend ergab sich die Frage, wie eine Höhle in der Papierwelt aussehen könne. In der Realität gibt es verschiedene Arten von Höhlen. Als Inspiration für dieses Spiel wurde eine Tropfsteinhöhle genommen, weswegen im Environment Design viele Stalaktit ähnlichen Gesteinsformationen von der Höhlendecke hängen. Am Anfang des Spiels ist man erst am Eingang der Höhle, d.h., dass hin und wieder Lichtstrahlen noch durchdringen. Je weiter man in die Höhle hinein geht, desto dunkler wird es. Weitere Elemente der Umgebung stellen Pflanzen und Flüsse dar. Alle Elemente haben jedoch gemeinsam, dass sie auch, wenn sie organisch sind, in einer Papieroptik designt sind, um das Concept Art des Spiels zu unterstreichen. Manche der Objekte sind rein zur Optik dar, andere Objekte können durch die Orbs sichtbar gemacht werden und benutzt werden.

Weitere Game Design Phasen

In der *Produktionsphase* wird die Implementation des Spiels begonnen. Das Konzept des *Rapid Prototyping* ist hierbei sehr hilfreich. Es beschreibt die Erstellung von Prototypen zu einem möglichst frühen Zeitpunkt[RA03]. Eine frühe Erstellung von Prototypen hat den Vorteil, dass Änderungen im Konzept mit relativ wenig Aufwand und Kosten durchgeführt werden können im Vergleich zu Änderungen im fertigen Produkt. Mit den Prototypen kann außerdem schon die Spielerfahrung getestet werden, die bei einem Spiel im Vordergrund steht. In diesem Projekt wurde dieses Konzept umgesetzt. Nähere Informationen dazu in Kapitel 4: Implementation und aktuelles Konzept.

Die *Nachproduktionsphase* beschreibt schließlich das Testen und die Fehlerbehebung im Spiel. Diese Phase ist in diesem Projekt zwar geplant, jedoch erst zu einem späteren Zeitpunkt umsetzbar, da der aktuelle Stand sich am Anfang der Produktionsphase befindet, insofern existieren hier noch keine konkreten Anforderungen oder Problemstellungen.

4 Implementation und aktuelles Konzept

Im aktuellen Entwicklungsstand [Juni 2015] ist die Konzeptionsphase weitgehend beendet.
Aufbauend auf den Designs aus der Konzeptionsphase entstehen momentan die ersten Prototypen. Mit diesen soll in erster Linie das Gameplay getestet werden.

4.1 Modelling

Der erste Prototyp stellt ein Model für den Character dar. Durch seine Bewegung kann ein erster Eindruck darüber vermittelt werden, wie sich das Spiel für den Spieler anfühlen wird. Zu diesem Zweck wurde ein grobes *Block – Out* angefertigt, um die Proportionen des Characters abzubilden und eine erste Topologie zu erstellen. Daraufhin mussten einige der Concept Arts überarbeitet werden, da bei der Realisation der Zeichnung, aufgefallen ist, dass einige Charakteristiken des Characters anatomisch nicht funktionieren. Die größte Herausforderung bestand aber darin, dem Modell auch ohne Textur schon das Gefühl von Papier zu geben. Die verschiedenen Eigenschaften des Papiers (weiche/scharfe Kanten, glatt, gebogen) sind alle im Konzept des Characters vorhanden und sollten auch demgemäß umgesetzt werden. Animationstechnisch ist zu beachten dem Modell genau dort genug Geometrie zu geben, wo es im Spiel ‚einknicken' wird, um eine glaubhafte Deformation zu gewährleisten.

Das weitere Vorgehen umfasste die Erstellung Texturierung des Modells und damit die Erstellung einer UV Map. Das schwierige war hierbei die Erstellung der zugehörigen Normal Map, welche in diesem Fall dafür benutzt wurde, um eine Illusion von Detailgenauigkeit zu erzeugen ohne die Polygonanzahl zu überschreiten. Damit ist es also möglich ein hochaufgelöstes Modell in eine Normal Map zu *baken* und auf ein niedrig polygonales Mesh anzuwenden, so dass dieses trotzdem noch die Detailgenauigkeit des hochaufgelösten Modells besitzt, aber spielegeeigneter ist. Die Herausforderung bestand hier größtenteils darin die Rundungen, Kanten und Spitzen des Characters in dem High – Poly Modell zu modellieren, um daraus eine vernünftige Normal Map zu erstellen. Dafür musste eine neue Topologie erstellt werden, die von dem Low – Poly Modell abwich.

Innerhalb von Unreal werden die Materialien mithilfe des Material Editors erstellt und bearbeitet. Der Editor besitzt eine sehr benutzerfreundliche Oberfläche. Per Drag & Drop können Attribute und Funktion einfach rein gezogen und verknüpft werden[…]. Prinzipiell bietet der Material Editor sehr viele Möglichkeiten, bedarf aber einer intensiveren Einarbeitungszeit.

4.2. Animation

Auf Basis des 3D Modell – Prototyps wurde ein Rig für das Modell erstellt, indem die Joints platziert und das Rig an den Character *geskinnt* wurden. Eine Herausforderung stellten hier insbesondere die für den Character charakteristischen Flügel und sein Mantel dar. Die Flügel besitzen viele Knickkanten, die deren Beschaffenheit und Bewegung bestimmen. An jeder Knickkante wurde ein Joint platziert, um die Bewegung jeder einzelnen Falte besser kontrollieren zu können. Ein ähnliches Problem bestand beim Mantel, was auch durch eine Joint – Kette gelöst wurde. Nur der gebogene Übergang zwischen den Falten hinten und der Seite bedarf keiner eigenen Joints. Hier ergibt die lineare Verteilung der Gewichtung der Vertices zwischen den anliegenden Joints das gewünschte Verhalten. Der Rest des Körpers hat einen menschenähnlichen Aufbau. Eine kleine Besonderheit stellen die digitigraden Füße des Characters da, da dort das Fußgelenk viel höher liegt als beim Menschen.

Für die Animation - Cycles dienen die Keyframes der Konzeptzeichnungen als Vorlage. Zu beachten war hier, dass in dem Konzept des Characters sowohl menschliche, als auch tierische Elemente vorhanden sind, woraus folgte, dass die Bewegungen dementsprechend angepasst werden mussten. Bei der Animation des Gehens beispielsweise bewegen sich die Flügel wie menschliche Arme, während die Beine durch ihren digitigraden Aufbau eher dem Gang eines Pferdes ähneln. Beim Laufen oder Rennen verhalten sich die Flügel dann eher wie ein Blatt Papier, welches leicht flattert. In der Sprung – Animation werden diese dann ausgebreitet, damit der Character besser gleiten kann.

Zusammengefasst ergeben diese durch den Körperbau des Character bedingten Besonderheiten ein ganz eigenes, spezifisches Bewegungsmuster. Zusätzlich verleihen subtile, sekundäre Animationen wie das Zucken der Ohren oder deren Ausfahren bei Gefahr dem Character eine Persönlichkeit.

Abbildung 5: Keyframes eines Animation Jump – Cycles (Design by Alex Ostrowski)

Für die Implementation in der Unreal Engine ist es notwendig das Rig mit seinem *Skinned Mesh* und den dazugehörigen Animationen zu importieren. Für den Austausch zwischen Unreal und Maya wird eine FBX – Datei verwendet. In Maya müssen das Rig und das Mesh über das FBX 2014 Plugin exportiert werden. Für den Character müssen alle Joints sowie das Skinned Mesh exportiert werden, während bei den Animationen nur die Joints ausreichen. Auf die Joints müssen jedoch Keyframes *gebaked* werden, da beim Animieren mit Animation Controls in Unreal Keyframes verwendet werden. Beim Import wird dann für das Skinned Mesh ein *Skeletal Mesh* und für die Joint Hierarchie ein *Skeleton* erstellt. Im Import – Dialogfenster wird dann der Skeleton ausgewählt und angegeben, welchen Zeitraum die Animation umfasst. Dieser wird dann in Unreal als *AnimSequenze* importiert [U..].

Um auf die Eingaben des Users während des Spiels reagieren zu können, benutzt man die von der Engine bereitgestellten Klassen *PlayerController*, welche die Eingaben des Users annimmt, und Character, welche die Eingaben verarbeitet und Eigenschaften für die Steuerung einer Figur bietet. Die weitere Verarbeitung der Eingabe wird dann im Code oder über Blueprints geskripted. In diesem Projekt wurde das Character Blueprint von Unreals Third Person Template mit dem Skeletal Mesh von unserem Modell verwendet.

Das Animation Blueprint stellt die Verbindung zwischen der Character Klasse und den Animationen dar. Der *EventGraph* nimmt die Eingabwerte der Character Klasse entgegen und verarbeitet diese, so dass sie im *AnimGraph* verwendet werden können. Der AnimGraph steuert dann die *Animation State Machines* und *BlendSpaces*. Eine State Machine ist eine visuelle Darstellung der Animation als States, deren Überblendungen durch Regeln und Änderung der

Parameter im EventGraph gesteuert werden kann. Ein BlendSpace ist ein Asset, welches die Möglichkeit bietet Übergänge zwischen mehreren Animationen anhand von zwei Werten zu erstellen. Aktuell werden bei diesem Projekt Übergänge zwischen Stehen, Gehen und Rennen mithilfe der Steuerung der Laufgeschwindigkeit in dem BlendSpace realisiert. Schließlich muss in der *GameMode* Klasse die Character Klasse referenziert werden, um sie im Spiel verwenden zu können. In den Levels, in denen der Character verwendet wird, werden dann in den World Setting der entsprechende GameMode gesetzt.

4.3 Softwareintegration und Programmierung

Als Game Engine wurde von Anfang an die Unreal Engine bevorzugt, da sie viele Funktionen bietet und grafisch qualitativ sehr hochwertig ist. Ausschlaggebend war u.a. der Punkt, dass Unreal neben C++ auch eine visuelle Skriptsprache integriert, die in sogenannten *Blueprints*[U..] umgesetzt wird. Dadurch können bestehende C++ - Klassen mit Blueprints erweitert werden und für unerfahrene Nutzer können Blueprints eingesetzt werden, um eine Spiellogik aufbauen zu können, ohne eine bestimmte Programmiersprache zu beherrschen. Durch die Programmierung in C++ bieten sich außerdem äußerst viele Anpassungsmöglichkeiten innerhalb von Unreal selbst, wobei Unreal zum Teil integrierte Funktionalitäten bietet wie bspw. die Garbage Collection[U..].

Ein weiterer Prototyp wurde in der Entwicklung der Orbs umgesetzt. Im Vergleich zum Character erforderte dieser Prototyp kein kompliziertes 3D Modell als Grundlage, sondern musste einem bestimmten Verhalten folgen. Wie bereits beschrieben sind Orbs kleine Naturgeist ähnliche Lebewesen, die den Character um Hilfe bitten, ihm aber gleichzeitig ihre Macht verleihen. Sie kreisen um den Character, wenn er durch die Papierwelt schreitet, wobei diese kreisende Bewegung auch Abweichungen beinhalten soll, um es realistischer zu gestalten. Gleichzeitig müssen sie vom Character weg und zum Character hin fliegen können.

Für die Integration der Orbs gäbe es mehrere Möglichkeiten in Unreal. Typischerweise wäre hier *Flocking* ein geeigneter Algorithmus, allerdings bietet Unreal bis jetzt keine native Unterstützung dafür und müsste von uns implementiert werden[S..]. Eine Implementation über ein Plugin, wäre möglich, wobei möglichst versucht wird Third – Party Komponenten auszuschließen. Die Implementation über ein Partikelsystem erwies sich als ästhetisch unzureichend, da die Partikel konstant neu gespawnt werden und eine gewisse Lebenszeit besitzen. Der aktuelle Ansatz verfolgt daher die Steuerung der Orbs über einen Pfad als zusammenhängende Gruppe. Es gibt eine Klasse *AOrbGroup*, der sich darum kümmert eine vorher festgelegte Anzahl an Orbs zu spawnen. Es können mehrere Orb - Gruppen im Level gespawnt werden und an die unterschiedlichen Actors angehängt werden. Die einzelnen Orbs bestehen aus einem *UStaticMeshComponent*, welcher das Mesh für die Orbs lädt. An den Component wird jeweils noch ein Partikelsystem (mit einem Ribbon Trail) angehängt, das dafür sorgt, dass die Orbs mit einem Schweif dargestellt werden. Beim Spawning der Orbs musste festgelegt werden, wie diese gespawnt werden. Unreal bietet hier eine Funktionalität die Position innerhalb einer Bounding Box zufallsbedingt zurück zu geben. Diese Funktionalität wird dafür benutzt eine Position für jeden Orb zu erstellen und ihn dorthin zu setzen. Zusätzlich

müssen die Orbs sich zwischen aktuell zwei *AttachmentTypes* entscheiden können. *Attached* bedeutet, dass die Orbs an einem Actor angehängt sind, während *FreeRoam* einen Modus bezeichnet, in dem die Bewegung der Orbs nicht von einem Actor beeinflusst werden. Die Bewegung selbst wird dann mittels einmalig dynamisch generierten *Splines* realisiert, welche bestimmen, wie sich die Orb - Gruppe um den Actor bewegt. Mithilfe von Tags unterscheidet die Orb – Gruppe, um welchen Actor (Character, Kokon, etc.) es sich aktuell handelt.

```
void AOrbGroup::FollowPath(float deltaSeconds)
{
    if (!this->OrbPath)
        return;

    float pathDistance = this->OrbPath->GetSplineLength();
    this->TravelledDistanceOnPath = FMath::FInterpConstantTo(this->TravelledDistanceOnPath, pathDistance, deltaSeconds, this->MovementSpeed);

    if (this->TravelledDistanceOnPath >= pathDistance)
        this->TravelledDistanceOnPath = 0.0f;

    const FVector location = this->OrbPath->GetWorldLocationAtDistanceAlongSpline(TravelledDistanceOnPath);
    const FRotator rotation = this->OrbPath->GetWorldRotationAtDistanceAlongSpline(TravelledDistanceOnPath);

    this->OrbsSceneComponent->SetWorldLocationAndRotation(location, rotation, true);
}
```

Abbildung 6: Codeausschnitt für die Steuerung der OrbGruppe über Splines

5 Fazit

In diesem Beitrag wurden die Entwicklung eines Spiels und dessen Weg von Idee zum Spiel und die Implementation des ersten Prototyps in der Unreal 4 Engine beschrieben.

6 Literatur

[RA03] Andrew Rollings, Ernest Adams. *Andrew Rollings and Ernest Adams on Game Design*. New Riders Publishing, 2003.

[SZ04] Katie Salen, Eric Zimmerman. *Rules of Play. Game Design Fundamentals*. The MIT Press, 2004.

[D10] Daniel Dumont. *Von der Idee zum Konzept*. http://www.makinggames.biz/features/von-der-idee-zum-konzept,2596.html, 2010.

[H13] Tina O'Hailey. *Rig it Right! Maya Animation Rigging Concepts*. Focal Press, 2013.

[G09] Jason Gregory. *Game Engine Architecture*. Taylor and Francis Ltd., 2009.

[S..] Daniel Shiffmann. *Implementation of Craig Reynolds Boids program*. https://processing.org/examples/flocking.html, ..

[P07] Purdue University. *Game Development Process*. http://www.e-games.tech.purdue.edu/GameDevProcess.asp, 2007.

[U] Unreal 4 Engine Documentation. *Setting Up a Character*. https://docs.unrealengine.com/latest/INT/Engine/Animation/CharacterSetupOverview/index.html