GLIDE

BY LUKAS OLESCH

3. SEMESTER | GE-LAB

CASUAL CASINO

MATR. NR: 406251

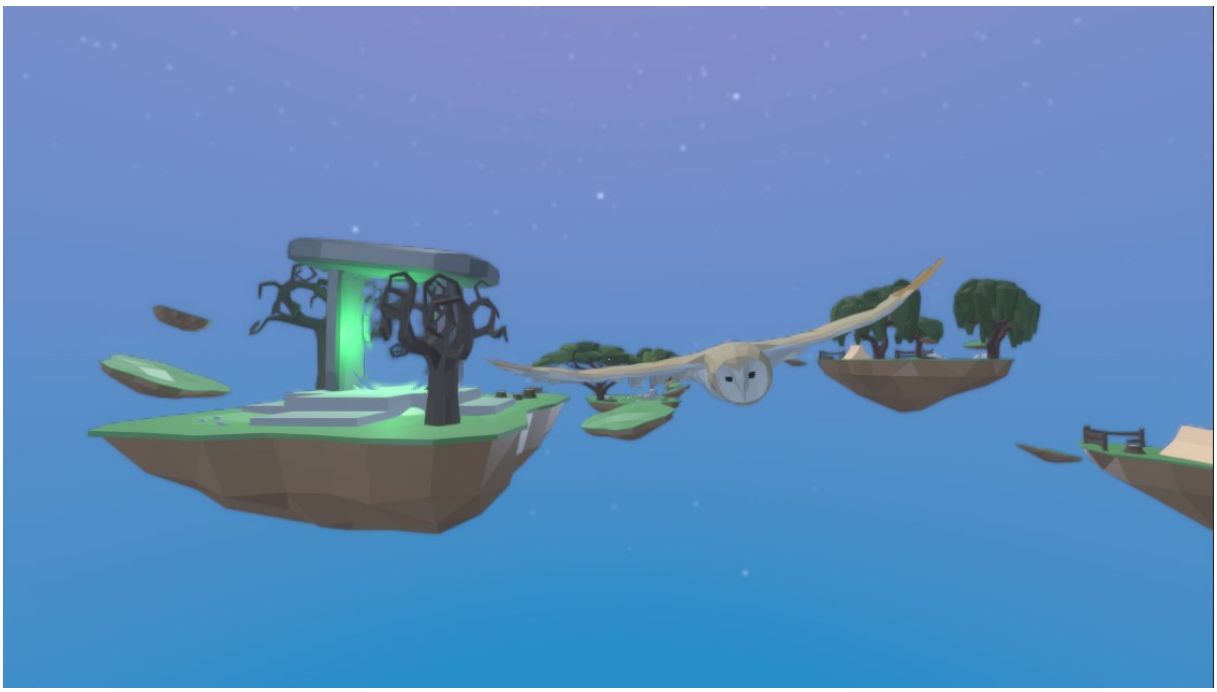

Abb. 1

Entspanne dich während du in einer endlosen Traumwelt durch die Gegend fliegst

Glide ist ein Endless-Runner, bei dem es in erster Linie um Entspannung geht, welche durch ambiente Sounds und Visuelle Ästhetik erreicht wird. Um das Ziel, einen Highscore aufzustellen, zu erreichen, ist man gezwungen geschickt Hindernissen auszuweichen und durch Portale zu fliegen, welche zufällig auf Inseln spawnen.

INHALTSVERZEICHNIS

1.0 KONZEPT

1.1 ANFANGSGEDANKEN

Abb. 2 (Mindmap Poolfight, erstes aufgegebenes Spiel)

Nach den ersten Brainstorming-Sessions ist mir ziemlich schnell klar geworden wie viel Arbeit in einem Spiel steckt. Es gibt eine Menge an Faktoren, die das Feeling dramatisch ändern können. Vergisst man zum Beispiel die Musik, kann es sein das keine Atmosphäre entsteht, und sich das Spiel nur wie eine „billige Demo“ anfühlt. Meine erste Mindmap war sehr klein und auch sehr spezifisch, aber bei den ersten Übungsstunden habe ich dann gesehen wie die anderen es gemacht haben, und wie so eine Mindmap wirklich aussehen soll. Auf dem Bild kann man meine Mindmap erkennen. Sie ist auf den ersten Blick recht unübersichtlich, und eigentlich nur für mich gedacht. Falls ich in einem Team arbeiten sollte, muss ich mir mehr Wert auf Organisation und Lesbarkeit legen.

1.2 SKIZZE

Im nachfolgenden Bild sieht man verschiedene Skizzen, oben links das Spiel zur Mindmap. Sie haben mir extrem dabei geholfen das Spiel zu verbildlichen und somit detailliertere Ideen mit in die Mindmap einzubringen.

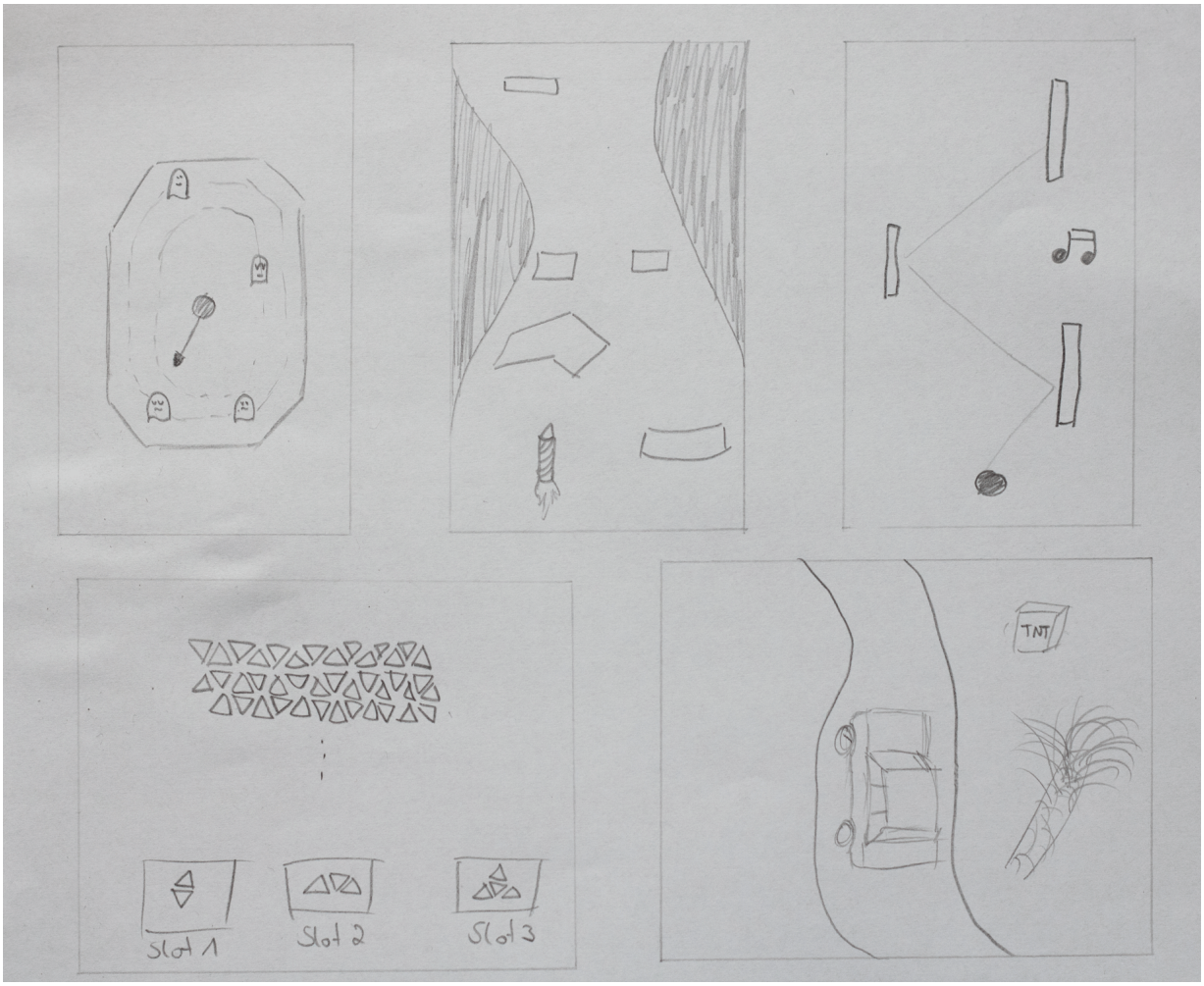

Abb. 3 (Skizzen der Spiele, zwischen denen ich mich entscheiden sollte)

1.3 FEHLSCHLAG

In den ersten paar Tagen habe ich versucht die Grundtechniken in Unity umzusetzen. Mir ist ziemlich schnell aufgefallen, dass das Spiel eher langweilig ist. Es hat kein Spaß gemacht, was bei einem Spiel natürlich suboptimal ist. Daraufhin hätte ich spannende Features mit einbauen können, aber mir ist eingefallen, dass wir das Spiel ja für WebGL bauen. Nachdem ich nachgeschlagen habe, was das Performance-mäßig bedeutet, habe ich mir vorgenommen ein graphisch anspruchsvolleres Spiel zu gestalten. Nach ein paar weiteren Stunden Recherche bin ich den verschiedenen Render Pipelines begegnet. Ich habe mich für die Universal Render Pipeline (URP) entschieden, da sie von der Performance her am flüssigsten läuft, wobei sie graphisch mehr unterstützt als die Standard Pipeline. Für die SRP hatte ich zu wenig wissen, und die HDRP unterstützt WebGL nicht.

1.4 ZWEITVERSUCH

Nachdem ich mich also für die Render Pipeline entschieden habe, habe ich mir überlegt, in welchem Stil ich sie voll auslasten könnte, ohne über Board zu gehen. Daraufhin vielen mir zwei Spiele ein, die Ästhetisch, aber nicht zu anspruchsvoll schienen. (Links Altos Adventure, rechts Aer Memories of Old)

https://adventuregamers.com/articles/view/33873 https://stadt-bremerhaven.de/altos-adventure-und-altos-odyssey-kostenlos/

Diese zwei Stile haben mich sehr inspiriert. Da ich allein war, war es eine gute Entscheidung den Low Poly Art Style zu wählen. Er sah schön aus, aber auch sehr minimalistisch. Und da ich schon seit langem ein Vogelspiel machen wollte, in dem man frei fliegen kann, war das meine Gelegenheit.

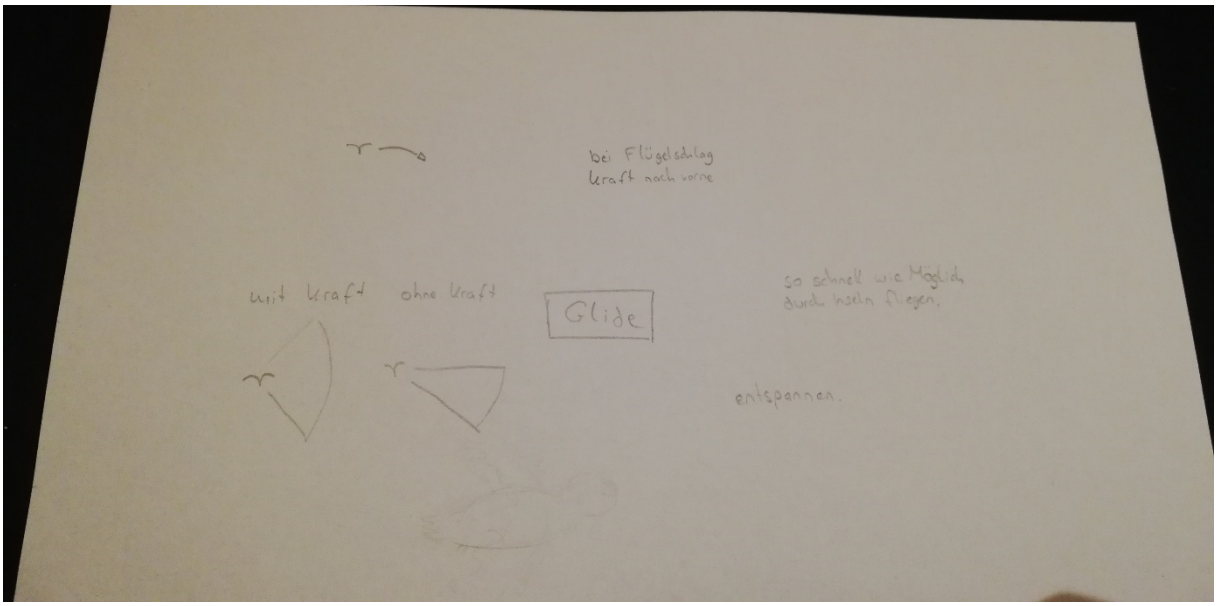

Abb. 4 (kleine Skizze zu Glide)

Wie man auf den Skizzen erkennen kann, habe ich anfangs Physik mit eingeplant. Leider ist das im laufe der Zeit als ein zu Zeitaufwendiges Thema markiert worden, weswegen ich es leider nicht als Mechanik implementieren konnte. Mein Ziel war aber von Anfang an ein Spiel zu kreieren, das wie meine beiden Inspirationen den Spieler beruhigt. Weitere Stichworte sind Endless-Runner, Ambient Sound, Aesthetic Low Poly Visuals. Man sollte als Vogel Spaß

dabei haben die Gegend zu erkunden, während man gleichzeitig versucht einen hohen Highscore aufzustellen.

2.0 MAKING-OF

2.1 SKY BOX SHADER

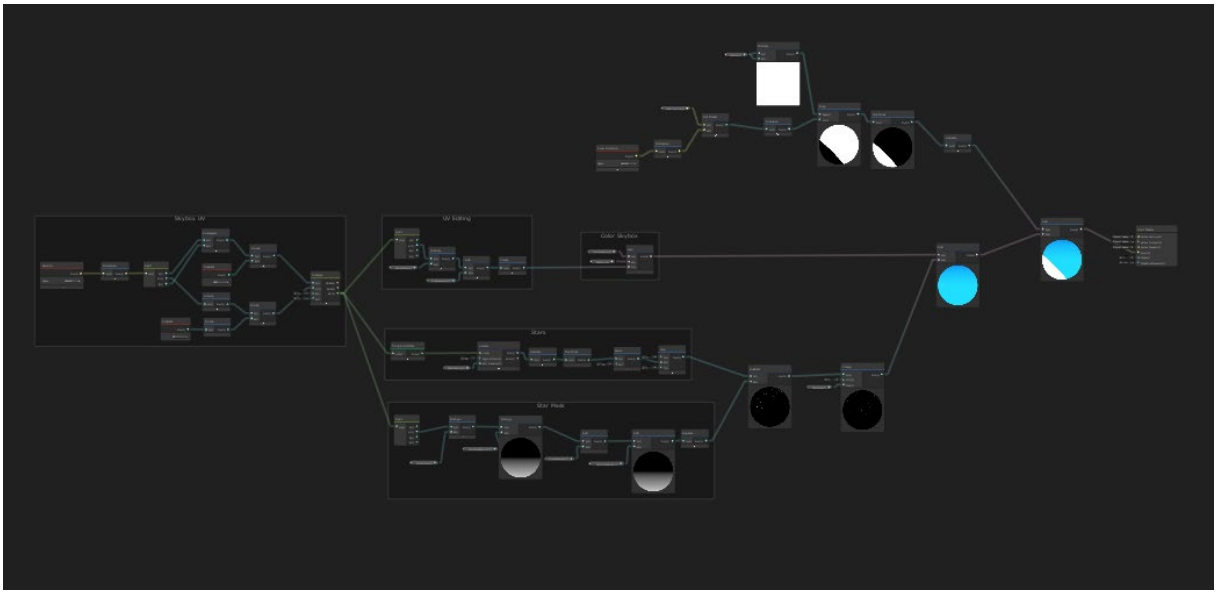

Abb. 5 (Unity Shadergraph der Himmelstextur generiert)

Oben kann man einen Shadergraph in Unity erkennen. Da ich schon ein bisschen Erfahrung mit Shadern habe, war das nicht allzu schwer. Ich habe natürlich ein bisschen Hilfe gebraucht *(Siehe Quellen)*, aber letztendlich ist es so. Der Shader erstellt über Sinus und Cosinus Funktionen eine Textur, die so auf den Skycube gemappt wird, dass genau am Horizont eine Trennung zwischen zwei Farben ist, welche natürlich als Gradient entweder einen Harten oder einen weichen Übergang haben können. Die zwei Farben kann ich dann in den Shader eingeben, was aber von einem normalen Skript aus geht. Dazu kommen auf der oberen Halbkugel bei Nacht Voronoi-Sterne am Himmel, welche man auch per Skript erscheinen oder verschwinden lassen kann. Im Skript habe ich dann einen Timer erstellt, der den Realen Tag simuliert und sowohl Farben als auch Stern Transparenz und Environmental Light für die Szene verändert. Nachdem ich ein Eulen-Model gefunden, und eine Partikeltextur erstellt habe, ist die Hauptmenü Szene entstanden.

2.2 UI

Natürlich fehlt da noch was. Ein Menü ohne User Interface ist kein Menü. Hier dachte ich mir, dass es Sinnvoll ist, ein einfaches, aber smoothes Interface zu machen, indem ich zwischen allen User Inputs Übergänge einfüge. Somit wird ruckartiges auftauchen von UI-Elementen verhindert, was den Spieler natürlich relaxt. Weiche Übergänge machen deshalb bei diesem Spiel viel Sinn. Abgesehen davon wollte ich dem Spieler die Steuerung zeigen, und da ein Tutorial zu zeitaufwendig für mich als Solo-Developer wäre, blende ich die Tasten einfach ins Menü ein und lasse den Spieler grübeln wozu sie nützlich sind.

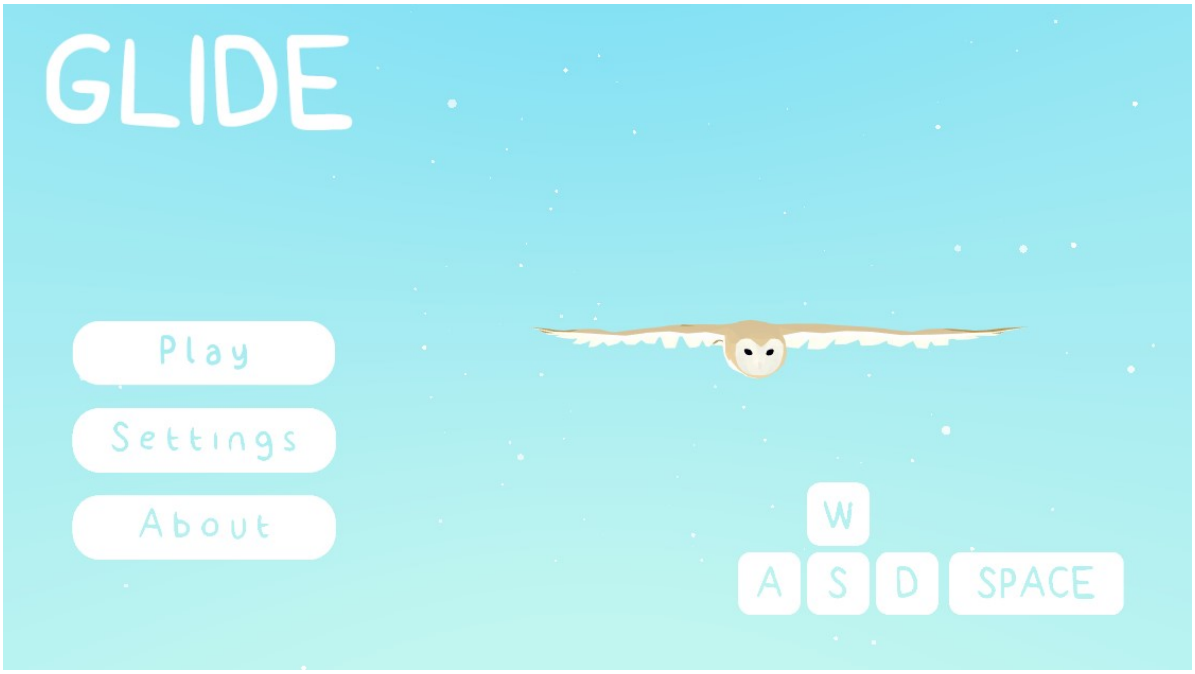

Abb. 7 (Fertiges Hauptmenü)

2.3 CONTROLS UND KAMERAVERFOLGUNG

Was bringt ein Menü ohne ein Spiel? Wir kommen zur Grundmechanik, die Steuerung der Eule und die Kameraverfolgung. And diesem Thema saß ich wohl am längsten, da dies der Großteil des Spiels ist. Angefangen habe ich mit einer einfachen Physik. Die Eule bewegt sich vor sich hin. Eine Kompliziertere Physik mit Flügelschlagen und Runterfallen, wenn man zu langsam ist sollte dann am Ende noch folgen, aber aus Gründen die ich in #1.4 geschildert habe ist daraus nichts geworden. Da ich das Modell der Eule gekauft habe, sind ein paar Animationen mitgeliefert gekommen, aber keine davon war wirklich schön und passend. Außerdem brauchte ich noch eine Zusatzanimation für die Rolle. Ich habe den Entwickler angeschrieben und er

hat mir gerne weitergeholfen. Das allein hat mir aber nicht gereicht. Die Animation an sich sah nicht gut und selbst mit Animation Blending sehr ruckartig aus. Außerdem trat ein anderes Problem auf. Die Rotation der Eule Lokal um ihr eigenes Zentrum wurde mit den Tasten WASD gesteuert, und weil die Eule sich die ganze Zeit vorwärtsbewegt (force * eule.forward), konnte man nun kontrolliert mit der Eule rumfliegen. Allerdings hat die Kamera sich selbst mit Damping und Smoothing zu ruckartig, und vor allem zu schnell bewegt. Die Kamera war nämlich einfach immer in einem gewissen abstand hinter der Eule (Kamera.pos = eule.forward * -CamDistance)

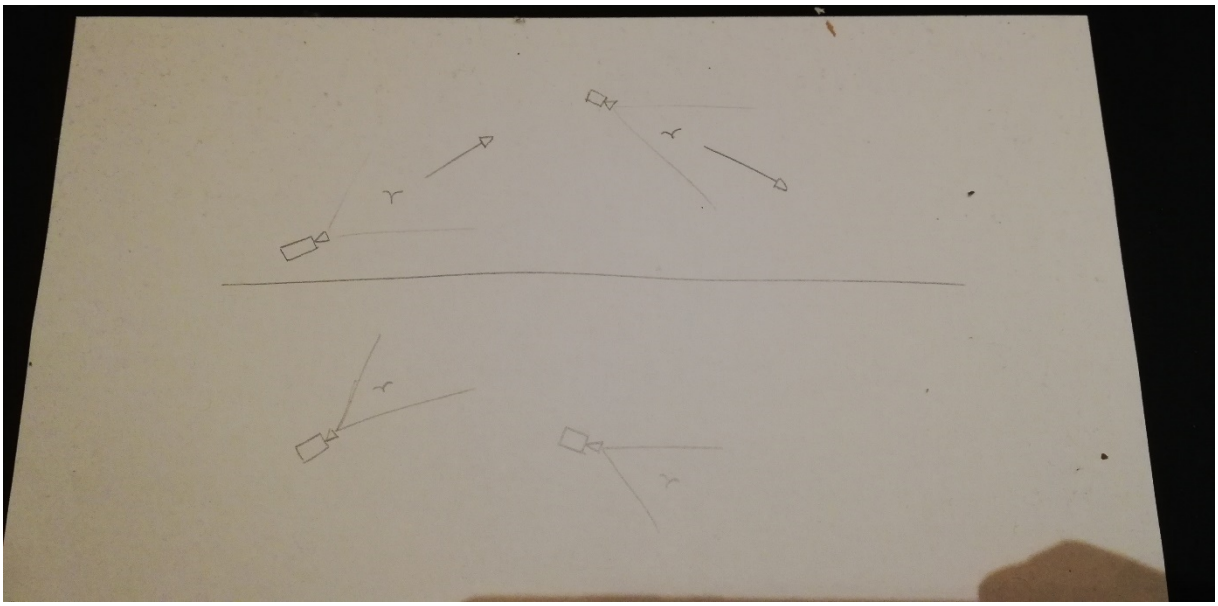

Abb. 8 (Schilderung des Problems)

Somit hat sich die Position der Kamera die ganze Zeit verändert. Wenn man also schnell zwischen W und S gewechselt hat, bewegte sich die Eule wenig, die Kamera aber sehr viel. Die Lösung zu diesem Problem hat mich mehrere Monate gekostet, und ist komplett ausdokumentiert. Man erstellt einfach einen Rig. Falls man die Eule jetzt rotieren will, dann tut man das um einen versetzten Punkt, bzw. den Rig. In Unity heißt das, man erstellt ein leeres GameObject, setzt die Eule als Child, verschiebt sie ein bisschen nach vorne. Die Kamera ist auf der Position des leeren GameObjects und schaut einfach in die Richtung der Eule. Jetzt kann man zwischen W und S wechseln so viel man will, die Kamera wird an der fast gleichen Position sein, während die Eule sich um die Kamera drehen wird. Das alles liest sich so einfach, aber es in Unity mit Quaternions umzusetzen, und die Sicht so zu clampen dass die Kamera nicht kopfüber sein kann ist auch eine Kunst für sich. Um die Animation der Eule schwunghafter darzustellen, addiere ich die normale Animation der Eule aus dem Animation Controller, mit einer Drehung der Eule um sich selbst, wobei hier jede Achse auf eine andere Distanz geclamped ist. Drückt man also nach rechts oder links, dreht sich die Eule nur um 20 Grad, drückt man aber nach oben, ist die Eule auf 90 Grad beschränkt (Leider keine

Loopings möglich D;). Gibt man die Drehung der Eule um den Rig auch noch dazu, kommt man auf das Ergebnis wie es im Spiel Glide realisiert wurde. Nebenbei habe ich auch noch einen Flügelschlag eingefügt. Die Eule hat also eine minimale Geschwindigkeit, und falls man mit den Flügeln schlägt oder durch ein Portal fliegt wird man schneller, aber nicht schneller als die Maximale Geschwindigkeit.

Abb. 9 (So sieht das am Ende dann aus)

2.4 MODELS

Nun gut, da fehlt aber immer noch was, nicht? Richtig. Das Level! Ich muss zugeben, dass ich Code mehr Liebe als das Grafische, aber nun gut. Ich hab alle Inseln selbst gemodelt, sowie ein paar Kleinigkeiten, aber auch ein paar offene Low Poly Assets aus dem Asset Store genommen. Ohne sie wären die Inseln zu leer gewesen, und sie selbst zu machen wäre leider zu viel für mich allein.

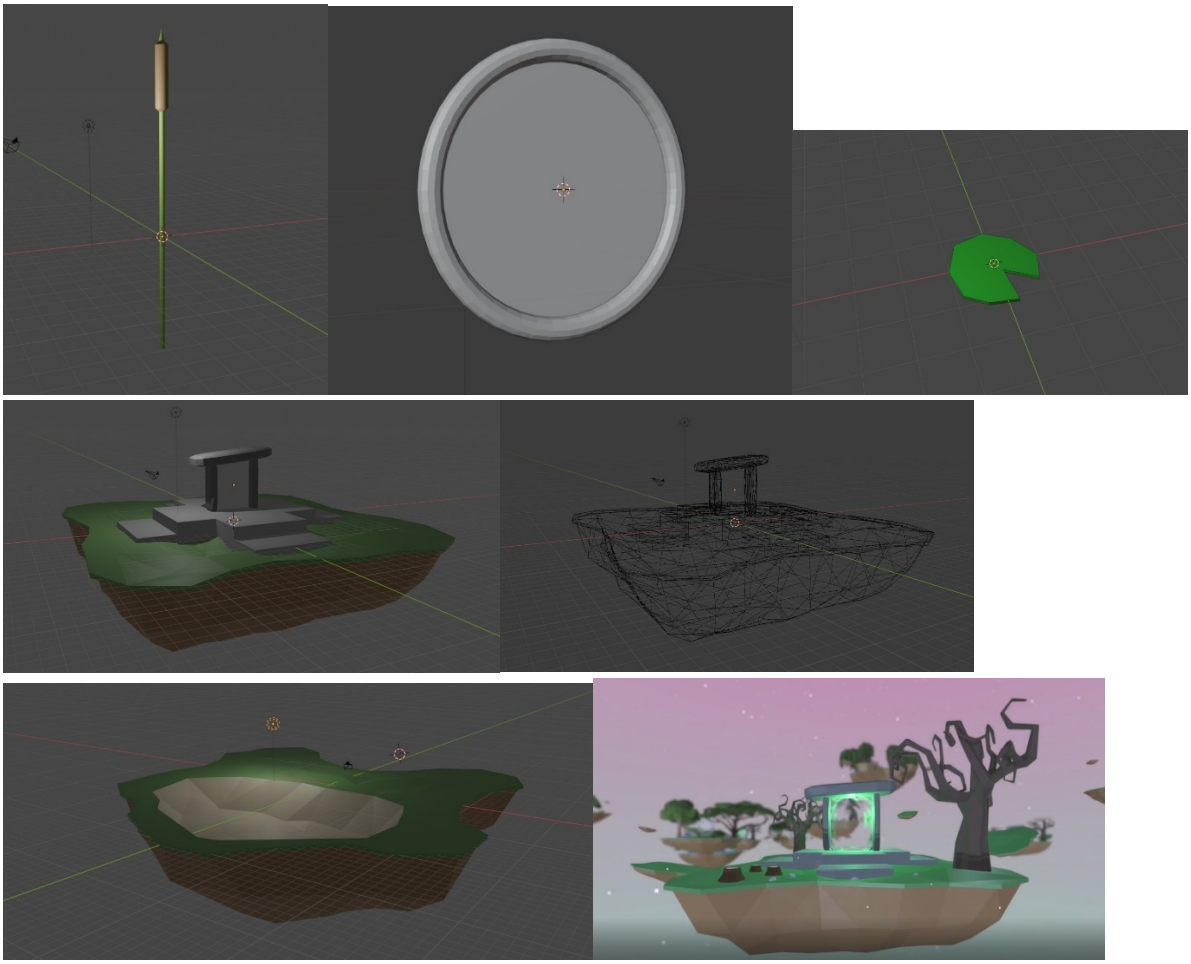

Abb. 10 (Ansammlung modellierter Sachen)

2.5 PORTAL SHADER

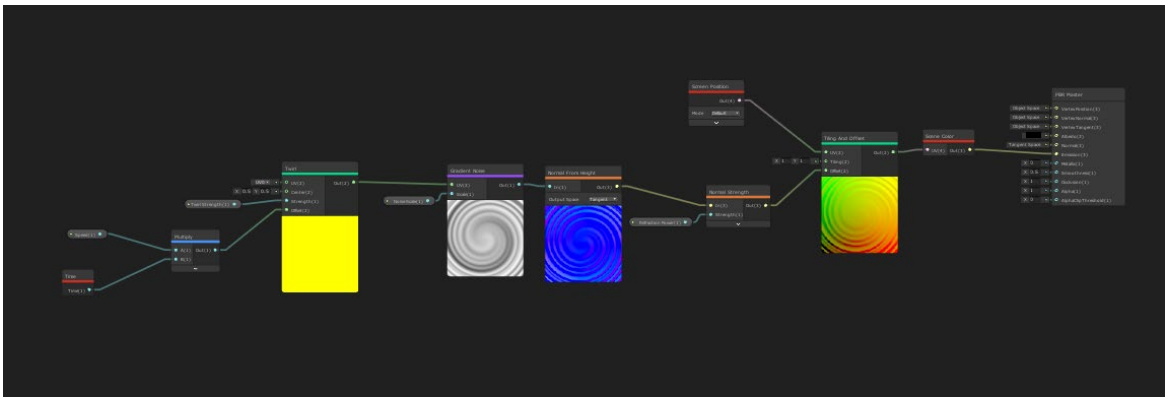

Abb. 11 (Unity Shader Graph Portal Shader)

Der Portal Shader ist einfacher als er aussieht, er nimmt einen Twirl (Spiraleffekt) der durch die Zeit animiert ist, setzt diesen in eine Normal Map um, verändert also die Geometrie des Objektes, welches dieses Material trägt, und lässt durch das Transparente Scene Color durch, also das Bild hinter dem Portal.

2.6 INSEL SPAWN

Am Anfang des Spiels werden im Level 3 „Inselgruppen“ gespawned. Jedes Mal, wenn ein Spieler durch ein Portal fliegt wird vorne eine neue Inselgruppe gespawned und hinten die letzte gelöscht. Alles ist variabel festgelegt, man kann also die Inseln, die am Anfang spawnen über eine Zahl im Inspektor verändern, genau so wie die Zahl der maximal lebenden Inseln, und man kann auch neue Inselgruppen an das Skript hängen. ´

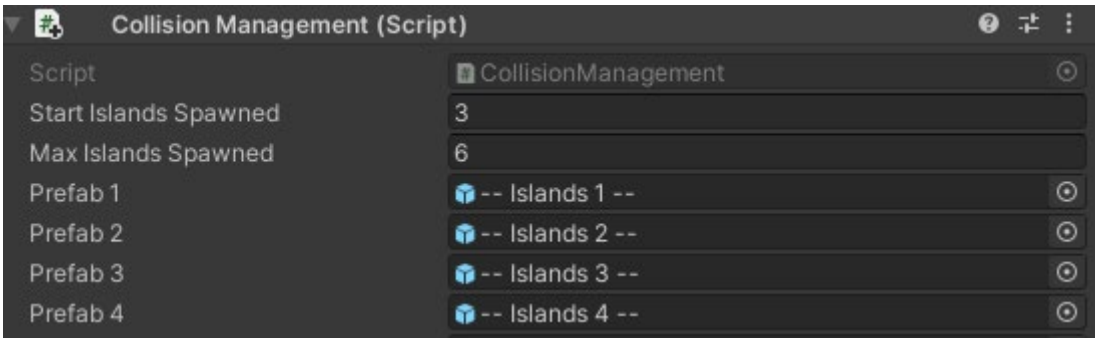

Abb. 12 (Skript im Inspektor)

2.7 MUSIK

So, nun haben wir ein fertiges Spiel. Moment, da fehlt ja noch der größte Teil! Das Spiel muss zum Entspannen eine Atmosphäre erzeugen. Dies geschieht durch ein Wind Geräusch, welches lauter wird je schneller man fliegt. Dazu kommt ein Geräusch, wenn man mit den Flügeln schlägt und eine Hintergrundmusik, die endlos zufällige Lieder spielt. Mir war die Musik wie am Anfang schon erwähnt sehr wichtig, da sie die Hälfte der Atmosphäre ausmacht. Deswegen habe ich meinen Lieblings Künstler Vexento genommen, welcher relaxende Musik produziert, sie aber auch ohne Copyright anderen zur Verfügung stellt.

3.0 ABSCHLUSS

Leider habe ich die Physik nicht mehr geschafft, und es gibt auch viele andere Aspekte, die ich noch verbessern könnte, seien es nur neue Inseln, Fine Tuning, oder sogar ganz neue Features. Abschließend muss ich sagen, dass ich zufrieden bin. Alles, was ich wirklich im Spiel haben wollte, habe ich, wenn auch nur mit viel Mühe, geschafft. Für die Zukunft weiß ich jetzt dafür, dass ich lieber weniger plane als es später zu bereuen. Mit Unity, Blender, Illustrator und After Effekts (fürs Video) bin ich gut zurechtgekommen, liegt vielleicht auch daran, dass ich Vorerfahrung habe.

4.0 QUELLEN

Music by Vexento
(https://www.youtube.com/channel/UCYZ9rknEmE4R1J_HBJ2yBlQ)

Shifting Winds: https://www.youtube.com/watch?v=OOzTZ06v520

Cloud Nine: https://www.youtube.com/watch?v=ILe2yxO-TPI

Lucid: https://www.youtube.com/watch?v=f18i2_B7av4

Sound Effects:

Wind SFX: https://www.zapsplat.com/sound-effect-category/wind/

Wing Flap SFX: https://www.youtube.com/watch?v=ydIFlbrDs-w

Bilder / Inspiration:

Aer, Memories of Old: https://adventuregamers.com/articles/view/33873

Alto's Adventure: **https://stadt-bremerhaven.de/altos-adventure-und-altos-odyssey-kostenlos/**

Shader Hilfe:

Wasser: https://alexanderameye.github.io/simple-water.html

Day/Night: https://medium.com/@jannik_boysen/procedural-skybox-shader-137f6b0cb77c

Portal: https://www.youtube.com/watch?v=CIvrWnxWG-k

Model Assets:

Low poly tree pack: https://brokenvector.itch.io/low-poly-tree-pack

Simple Low Poly Nature Pack: https://assetstore.unity.com/packages/3d/environments/landscapes/simple-low-poly-nature-pack-157552

Simple Nature Pack: https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153