



AUREA

Ausarbeitung SS 21

Von:	Matrikelnummer:	Rolle:
Dennis Hawran	262357	Entwicklung/Projektmanagement
Nick Häcker	262144	Entwicklung
Jonathan Schnee	260363	Entwicklung
Leonie Schwall	262476	Modeling/Konzeption
Sebastian Pfeiffer	262068	Modeling/Konzeption
Lukas Blea	260610	Modeling/Konzeption

Stand: 09.07.2021

Vorgelegt am: 09.07.2021

Inhaltsverzeichnis

Trailer	3
Autor: Lukas Blea	3
Rendering	10
Autor: Jonathan Schnee	10
User Interface	14
Autoren: Leonie Schwall und Nick Häcker	14
Modeling	19
Autoren: Leonie Schwall und Sebastian Pfeiffer	19
VFX	23
Autor: Sebastian Pfeiffer und Lukas Blea	23
Shopsystem	29
Autor: Nick Häcker	29
Augmented Reality	36
Autor: Dennis Hawran	36
Netzwerk	39
Autor: Dennis Hawran	39
Kampfsystem	41
Autor: Dennis Hawran	41
Inventarsystem	43
Autor: Nick Häcker	43
Soundsystem	45
Autor: Jonathan Schnee	45

Trailer

Autor: Lukas Blea

Planung

Um den Trailer für Aurea zu planen muss man sich erst einmal überlegen, welche Elemente des Spiels man in dem Video unterbringen möchte. Für mich war es wichtig einen schönen Übergang zwischen realen Aufnahmen und virtuellen Aufnahmen zu finden, um den Zuschauer auf eine kleine Reise mitzunehmen. Die virtuellen Aufnahmen sollen starten, nachdem der Protagonist des Videos durch ein Portal läuft. Dies ist im Einklang mit unserer geschriebenen Story, da auch dort der Protagonist in die Welt der Aurea gerufen wird, indem sie ihm ein Portal in die echte Welt setzen womit er/sie dann in diese Welt eintreten kann.

Nachdem also klar ist, was im Video passieren soll, müssen jetzt Entscheidungen getroffen werden wie: Wo sollen die Aufnahmen stattfinden und welche Musik passt zu dem Trailer?

Location

Für die Location war es wichtig darüber nachzudenken, wo sich der Charakter im Spiel befinden wird, nachdem er durch das Portal läuft. Natürlich sollen die Szenen der echten Welt sich nicht mit den Szenen der virtuellen Welt streiten. Heißt soviel wie: Wenn er Tagsüber durch das Portal läuft, sollen die virtuellen Aufnahmen nicht dunkel sein, da es auch in der Welt der Aurea Tags ist. Da auf Sky Island eine kleine Burg steht, war es klar, dass die Aufnahmen im echten Leben auch bei einer Burgruine stattfinden sollten und dort dann das Portal erscheint. Somit sind wir auf die Burgruine Kirneck gestoßen. Nachdem wir dort vorbeigefahren sind um uns darüber schlau zu machen ob Aufnahmen dort Möglich sind, gab es einen entscheidenden Faktor über den man sich bewusst werden musste: Wo stellen wir über After Effects das virtuelle Portal hin, durch das der Protagonist laufen muss?

Glücklicherweise haben wir den perfekten Eingang gefunden mit einem großen Rahmen und Treppen die in die Burgruine führen. Dieser Eingang, stellt sich heraus, war perfekt geeignet für diese Aufnahmen.



Durch die dicke des Rahmens am Eingang war es mir möglich, das virtuelle Portal realistischer wirken zu lassen, da ich alle drei Dimensionen gut ausnutzen konnte. Vor allem der Tiefeneffekt ist besser umzusetzen da, wie vorher schon erwähnt, der Rahmen des Eingangs besonders tief geht und man das Portal somit mittig setzen kann (Siehe Bilder der nächsten Seite).

Portal Implementierung

Um das fertige Portal nun in die entsprechenden Szenen einzufügen, muss man den 3D-Kamera Tracker von AE benutzen. Dieses Tool erlaubt es dir in einem Video 2D Objekte einzusetzen und mit Hilfe der 3D-Kamera mitbewegen zu lassen. Nachdem du "Kamera verfolgen" drückst, erstellt das Programm Punkte auf dem Video, welche man auswählen kann. Diese Punkte werden über das Programm getrackt und man hat die Möglichkeit Bilder & Videos an die entsprechenden Punkte einzusetzen. Somit muss man nicht selbst die räumliche Formatierung des Portals vornehmen und auch nicht per einzelne Keyframes das Portal an jede Kamerabewegung anpassen. Dies passiert nun automatisch über den Tracker.

Zu schön wäre es allerdings, wenn das schon alles ist. Leider wird man oft mit Problemen konfrontiert und muss improvisieren. Hier zum Beispiel war das Portal zwar richtig gesetzt, allerdings ragte es über den Rand des Eingangs der Burgruine hinweg. Wenn ein 3D Effekt für den Zuschauer entstehen soll, muss man nun mit Masken arbeiten um bestimmte Bildelemente verschwinden, oder auch auftauchen zu lassen.

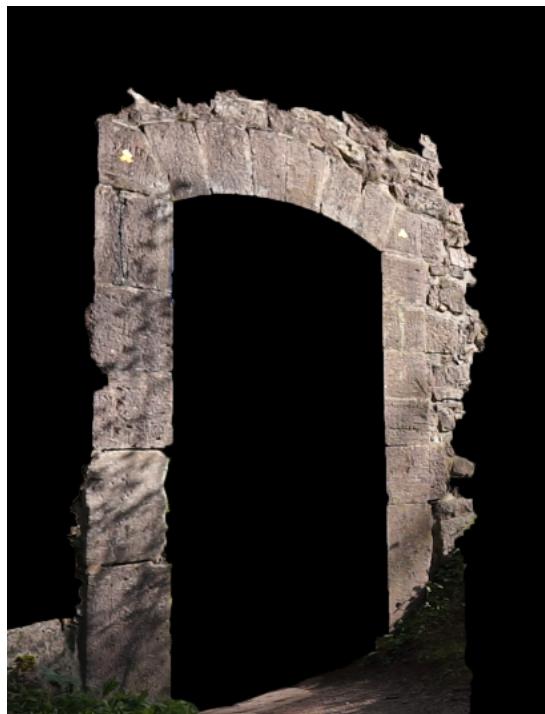
OHNE MASKE:



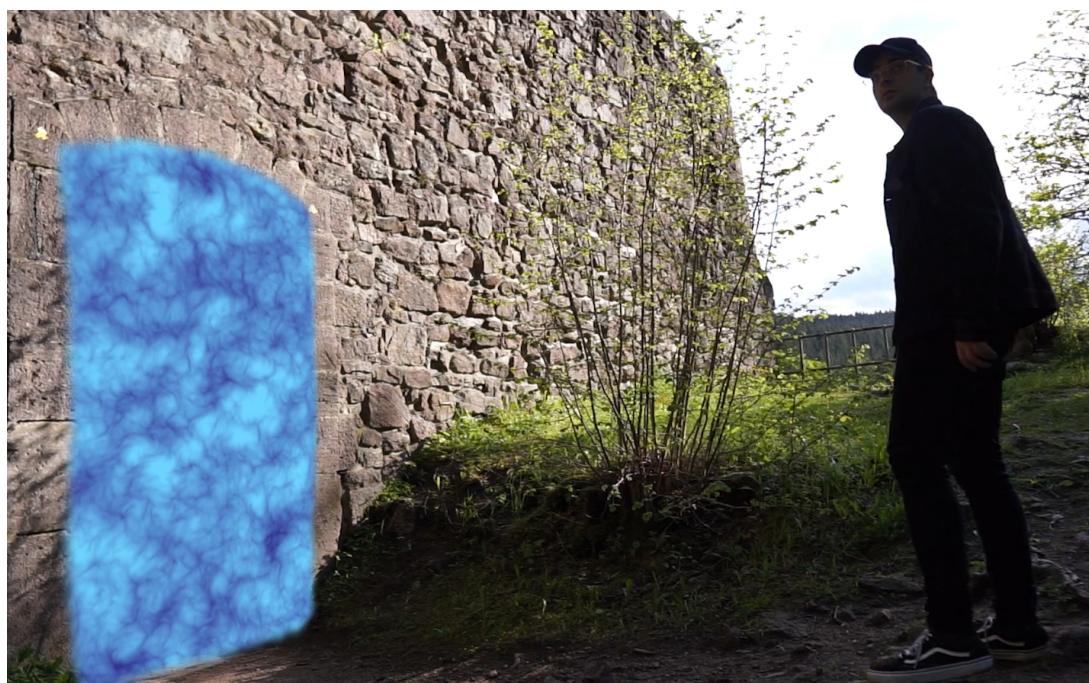
MIT MASKE:



MASKE:



ERSTER VERSUCH:



Im ersten Versuch kann man gut erkennen was ich meine. Das Portal ist zwar im Video, erzielt allerdings keinen großen Erfolg darin real oder überzeugend zu wirken. Das Portal muss, in einem 2D Video, weiter in die 3. Dimension, also die

Tiefe. Deshalb habe ich mich dazu entschieden das Portal weiter in die Einwölbung des Eingangs zu setzen.

Diese ganzen kleinen Improvisationen sind notwendig, da das Programm nicht alles für einen erledigt. Man muss sich stets überlegen wie man die Szene anpassen kann und muss flexibel sein. Offenheit für neues ist hier wichtig und auch die Umsetzung bei anderen Filmen zu sehen hilft sehr dabei neue denkanstöße zu finden.

Weitere Informationen zur Erstellung des Portals in After Effects kannst du im Kapitel VFX finden.

Drehbuch

Nachdem also klar war, dass diese Location eine gute Wahl für das Filmen unseres Trailers ist, erstellte ich ein kleines Drehbuch indem ich mir visuell versucht habe vorzustellen, wie die einzelnen Szenen gedreht werden könnten.

DREHBUCH AUREA

Wald-Szene (Real Life)
[Fade In]

#1 Kamera (Nah): Aufnahme der Füße, langsame Kamerafahrt bis an den Kopf.
Alles seitlich
(3-5 Sekunden)

#2 Kamera (Amerikanisch): Aufnahme von Hinten (Obersicht). Kamera folgt dem Mensch durch den Wald.
(3-5 Sekunden)

#3 Kamera (Nah): Aufnahme von Seite (Untersicht). Mensch holt Handy aus seiner Hosentasche nachdem ein Notifikation-sound zu hören ist.
(2 Sekunden)

#4 Kamera (Groß): Aufnahme des Gesichts des Menschen. Verwundert & Verirrt
(1 Sekunde)

#5 Kamera (Groß): Bildschirm des Handys. NOTIFICATION VON AUREA: „Wir brauchen deine Hilfe!!“
(1-2 Sekunden)

#6 Kamera (Normale + Obersicht): Mensch tippt auf Notifikation.
(1-2 Sekunden)

#7 Kamera (Totale): Mensch steht dort, nachdem er auf Notifikation getippt hat. 2 Sekunden passiert nichts, bis auf einmal ein helles Licht/Portal aus dem Handy erscheint und ihn hereinzieht.
(5-10 Sekunden)

[Fade Out]

Nicht alles davon wurde umgesetzt und manches habe ich dann vor Ort anders gemacht. Aber es ist wichtig sich eine klare Struktur im vorhinein zu überlegen um dann vor Ort zu wissen, welche Szenen essentiell sind und mit welchen man noch etwas Spielraum hat. Auch unser Protagonist Lennart hat sich in den Dreh mit eingebunden und eigene kleine Ideen eingebracht. Erst am Ort der Aufnahme fallen einem manchmal Dinge ein, die besser funktionieren

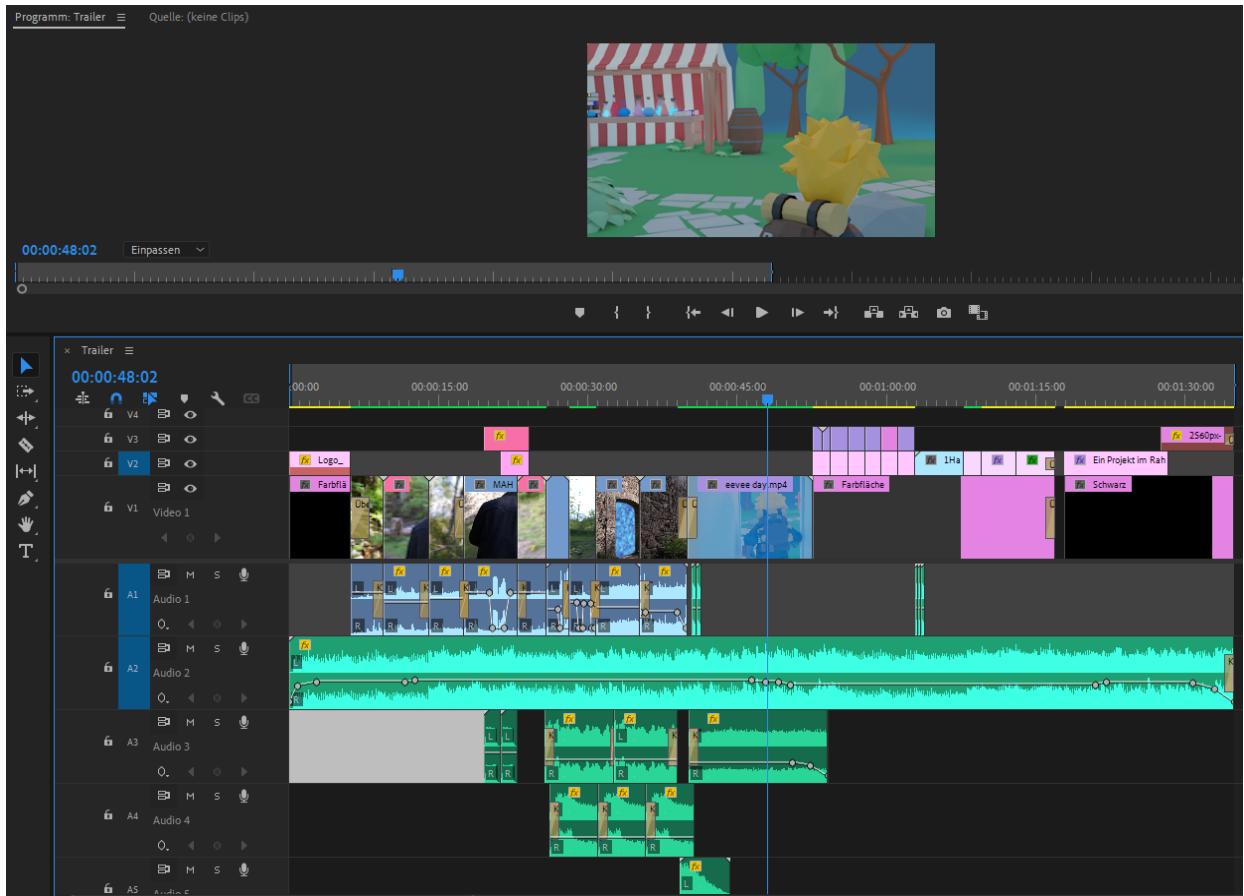
könnten wenn man sie anders umsetzt. Geplant waren Aufnahme seiner Füße während er am laufen ist. Stattdessen haben wir aus dem Gras heraus, aus einer stillen Position gefilmt, um dem Zuschauer ein besseres räumliches Gefühl zu geben wo er gerade ist und was gerade passiert.

Nachdem die Aufnahmen abgeschlossen sind geht es nun darum, sie in einen sinnvollen Kontext anzureihen und über Adobe Premiere Pro zusammenzufügen.

Soundtrack

Bei einem Videospiel ist das Auditive Erlebnis ein ausschlaggebender Faktor darüber, wie sehr sich ein Spieler in diese virtuelle Fantasy Welt einfinden kann. Deshalb war es mir wichtig die Atmosphäre des Spiels über den richtigen Soundtrack im Trailer zum Leben erwecken zu lassen. Meinen ausgewählten Soundtrack habe ich, mit einer entsprechenden Lizenzierung, aus Envato Elements heruntergeladen. Vieles gibt es dazu nicht zu sagen, außer das es sich richtig angefühlt hat. Man sollte sich nicht beim ersten Soundtrack festlegen sondern immer flexibel sein und sich auf neues einlassen. Geplant war am Anfang noch ein ganz anderer Soundtrack der Fröhlicher und Heiterer wirkt. Allerdings spiegelt dieser neue Track ein besseres Gefühl des Abenteuers und der Erkundung wieder. Auch war es mir möglich die virtuellen Szenen besser auf den Soundtrack anzupassen und dementsprechend das Video zu schneiden. In der Szene im Trailer, bei der alle unsere Aurea vorgestellt werden, kann man das gut sehen. Jeder Schnitt ist auf den Sound des Trailers angepasst.

Fertiger Trailer



Zu sehen ist hier der fertige Trailer in seinen einzelnen Komponenten auf Premiere Pro. Hier bekommt man nochmal einen Einblick auf die kleinen Feinheiten die bei so einem Trailer mitwirken.

Rendering

Autor: Jonathan Schnee

Da der Trailer Aufnahmen vom Spiel enthalten soll, wurden diese in Blender modelliert und animiert. Dann kam aber die Frage auf wie rendern wir diese Animationen am besten? Wichtige Kriterien waren für uns die Renderzeiten und Qualität. Die drei Render Engines die wir zur Auswahl hatten waren:

Eevee

Eevee steht für Extra Easy Virtual Environment Engine. Eevee ist im Gegensatz zu den anderen beiden keine Raytrace Render Engine. Das bedeutet anstatt jeden Lichtstrahl einzeln zu berechnen nutzt es eine Rasterisierung die abschätzt, wie das Licht in Art und Weise mit Objekten und Materialien interagiert. Dies wird mithilfe zahlreicher Algorithmen ermöglicht. Der große Nachteil daran ist, dass es nie physikalisch so genaue Renderings wie die anderen liefern kann. Trotzdem ist Eevee gut genug für Low-Poly-Objekte und andere „einfachen“ Gegenstände. Der größte Pluspunkt ist die Renderzeit von Eevee, da es die Lichter abschätzt, ist es viel schneller als die anderen Kandidaten. Eevee selbst ist in der normalen Blender Version einstell- und renderbar.

Cycles

Cycles ist das Gegenstück von Eevee und ist ein Raytrace Render der alle Lichtstrahlen in einem Renderframe berechnet. Er wurde entwickelt um physikalisch basierte Ergebnisse „out of the box“ zu liefern. Da er alles genau berechnet pro Frame dauert er aber deswegen viel länger als Eevee. Dafür sind seine Ergebnisse schöner und physikalisch korrekter. Auch er ist in normalen Blender Version einstell und renderbar.

CyclesX

Cycles X eine Neuauflage von Cycles. Es wurden Verbesserungen vorgenommen, die das Weiterentwickeln vereinfachen und am wichtigsten die

Renderzeit verbessern sollte. Aber CyclesX ist noch in der Alphaphase, bedeutet das viele Features noch nicht oder nur zum Teil implementiert sind. Wie zum Beispiel das Rendern von Volumen. Da es ein Alphaprojekt ist muss man auch die Experimentelle Blender Version nutzen an der täglich entwickelt wird.

Renderzeiten Pro Frame



EEVEE: 0 MIN 0 SEC 840 MSEC



CYCLES: 17 MIN 13 SEC 130 MSEC



CYCLESX: 00 MIN 37 SEC 550 MSEC

Der CyclesX Render ist mit Google Colab gerendert, damit es noch schneller geht. Google Collab ist eine Webseite von Google auf der man Python Code im Browser schreiben und ausführen kann. Dazu ist es auch möglich auf

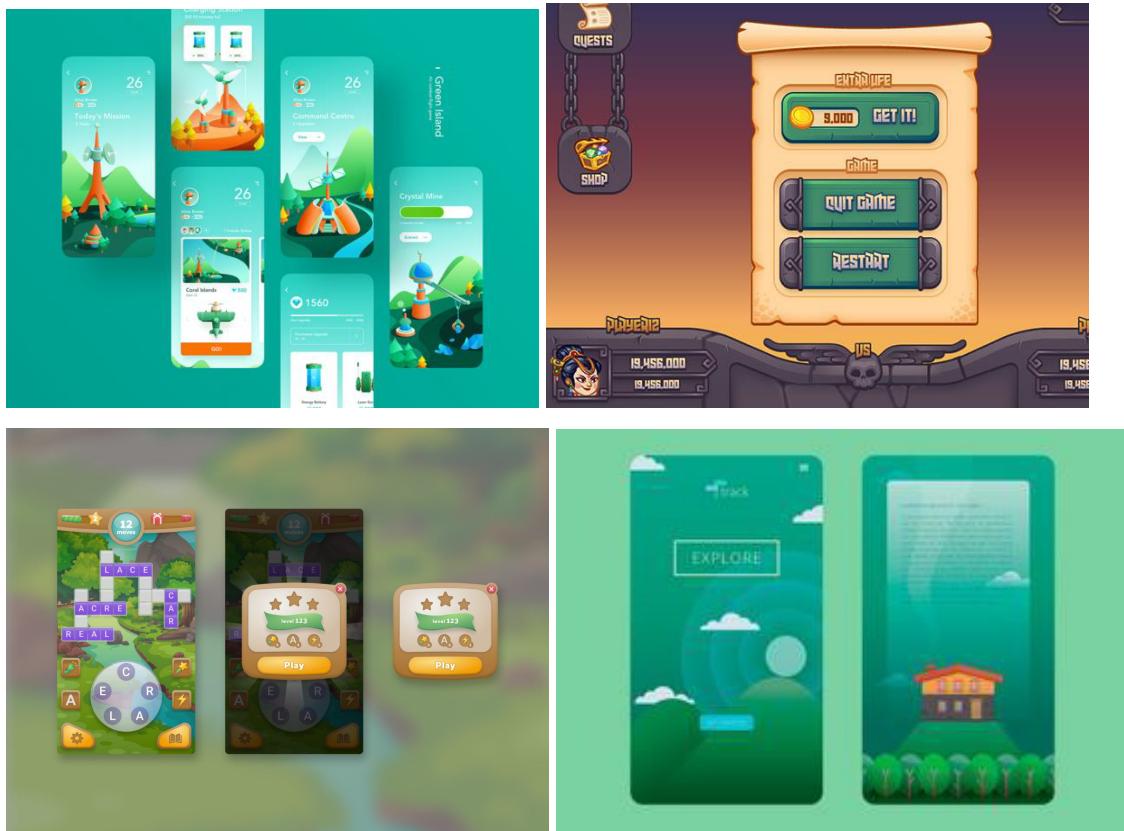
Ressourcen von Google zuzugreifen wie zum Beispiel der GPU mit der man dann Blender Render erstellen kann, da Blender auch auf Python geschrieben ist und auch ohne Programm Interface renderfähig ist.

Da die Render im Trailer Nebel haben sollten, konnten wir leider nicht CyclesX nehmen, da er kein Volumen anzeigen kann, aus dem der Nebel besteht. Cycles selbst ist wegen der Zeit ausgeschieden da die Render mehr als 30h in Anspruch genommen hätten. Somit blieb dann Eevee übrig für das wir uns entschieden haben. Also haben wir mit den Einstellungen rumgespielt um so viel wie möglich rauszuholen. Wir haben smooth Shadow eingeschaltet, um die harten Schatten zu entfernen und haben ein Irradiance Volume Objekt benutzt um den Bloom Effekt zu simulieren der auf andere Objekte fällt. Somit kam ein guter Render heraus der auch schnell gerendert ist. Die Inseln selbst hatten wir als wir Zeit haben aber in Cycles gerendert damit die schöner aussehen.

User Interface

Autoren: Leonie Schwall und Nick Häcker

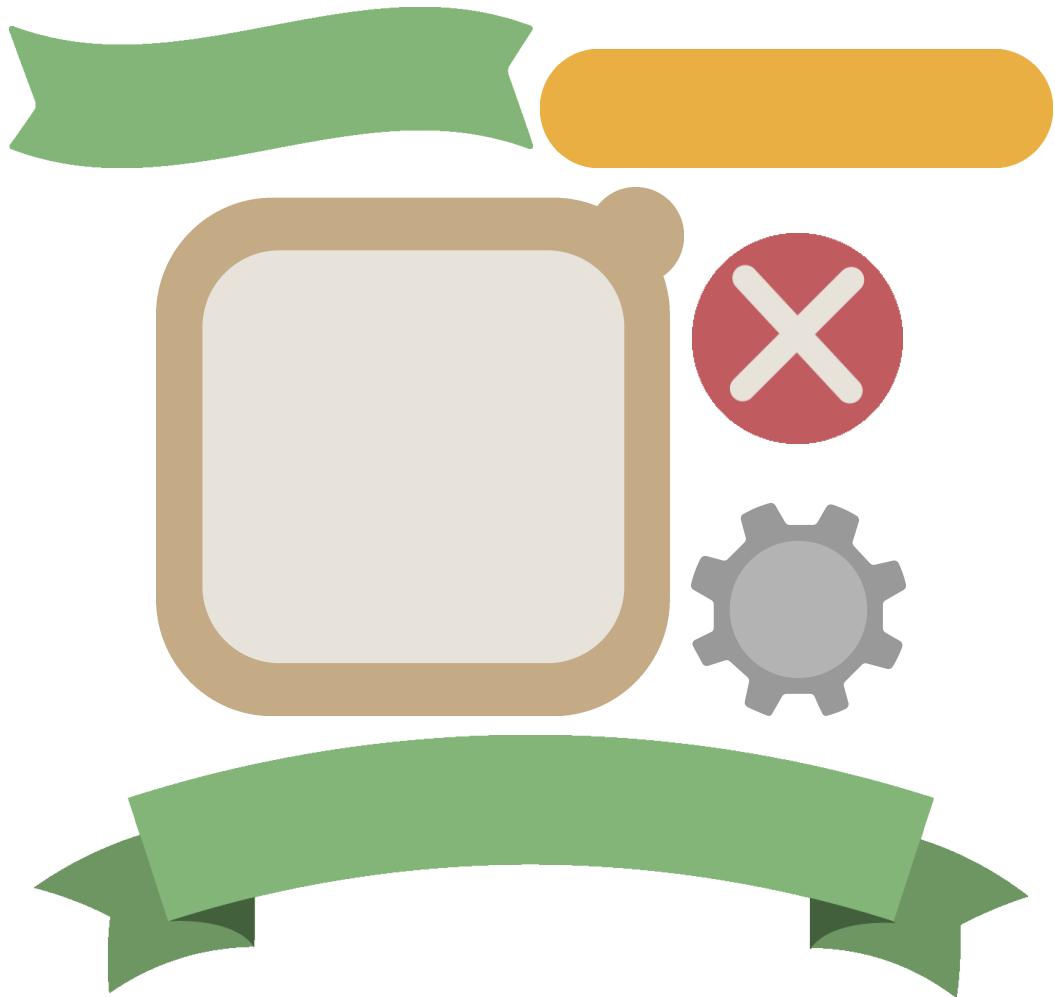
Das User Interface ist im Flatdesign gestaltet. Dieses ist derzeit viel genutzt unter anderem von Google. Flatdesign bedeutet, einfache Flächen und Formen, keine realistische Darstellung von Texturen und Dimension und der Einsatz von Farben, also ein minimalistischer Gestaltungsstil.



Wie im Moodboard zu erkennen ist, haben wir uns an einfachen Farben und Formen orientiert, die gut zu den gestalteten Inseln passen.

Erstellt haben wir daraufhin in Adobe Illustrator einzelne Komponenten auf Vektorbasis. Vektorgrafiken beruhen auf mathematischen Formeln, dadurch kann man die Grafiken beliebig groß skalieren, ohne das es sich auf die Qualität der Grafik auslöst. Anders wie z.B. bei Photoshop welches auf Pixelbasis

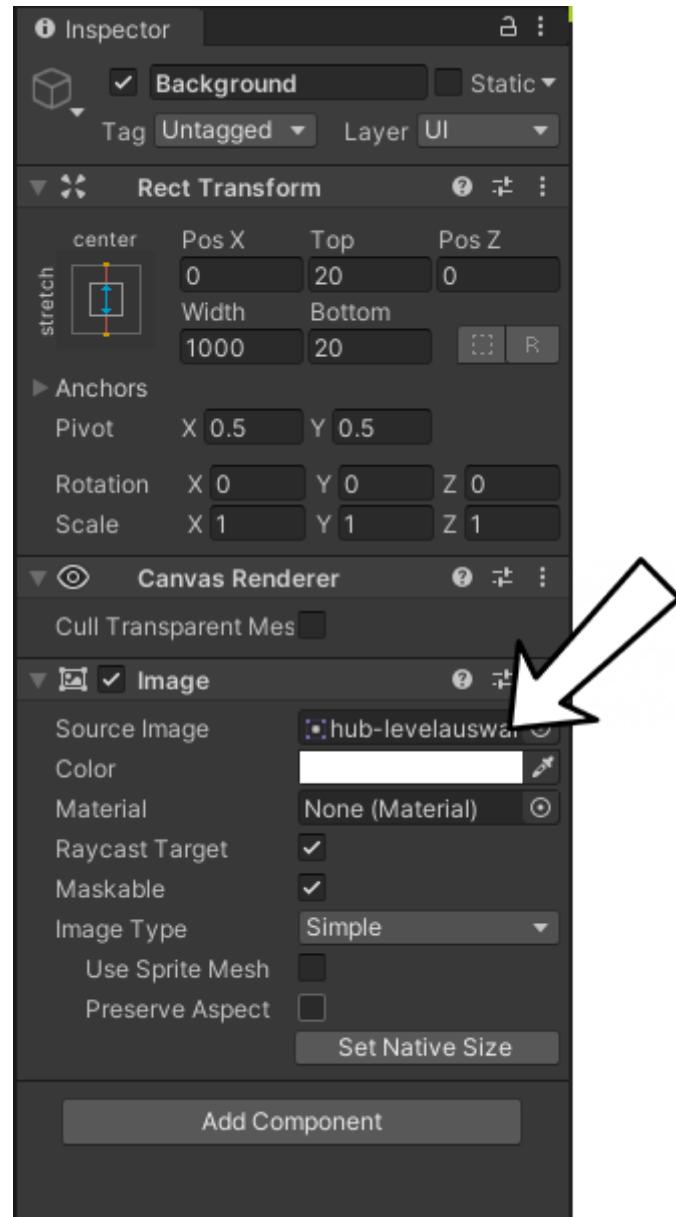
funktioniert. Die einzelnen Komponenten, wie der Hintergrund, Buttons, Banner, Einstellungs-Zahnrad, Exit-Button wurden erstellt und erst im Anschluss in Unity zu einem vollständigen Screen angeordnet, um das ganze flexibel gestalten zu können.



Umsetzung in Unity

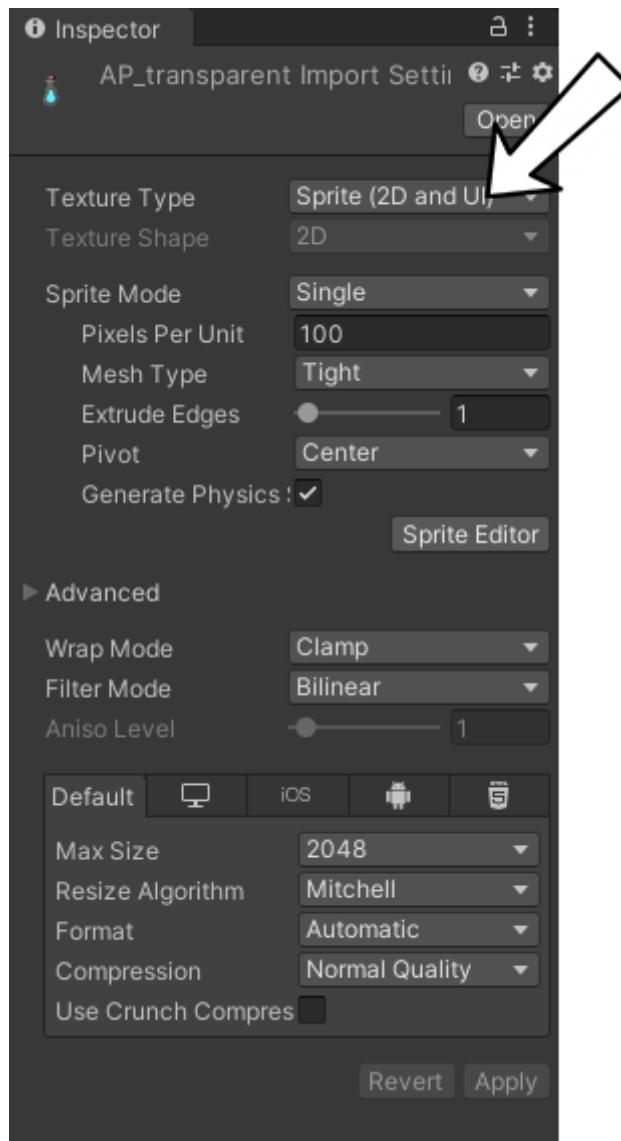
Um User Interfaces (UI) in Unity zu realisieren, werden 2D Canvas Objekte erstellt, auf denen einzelne UI Elemente wie Bilder, Videos, Texte oder Buttons angezeigt und nutzbar gemacht werden können.

Die erstellen Segmente der einzelnen UI Elemente werden in Unity in der Image Komponente eines Game Objektes als Source Image angegeben.



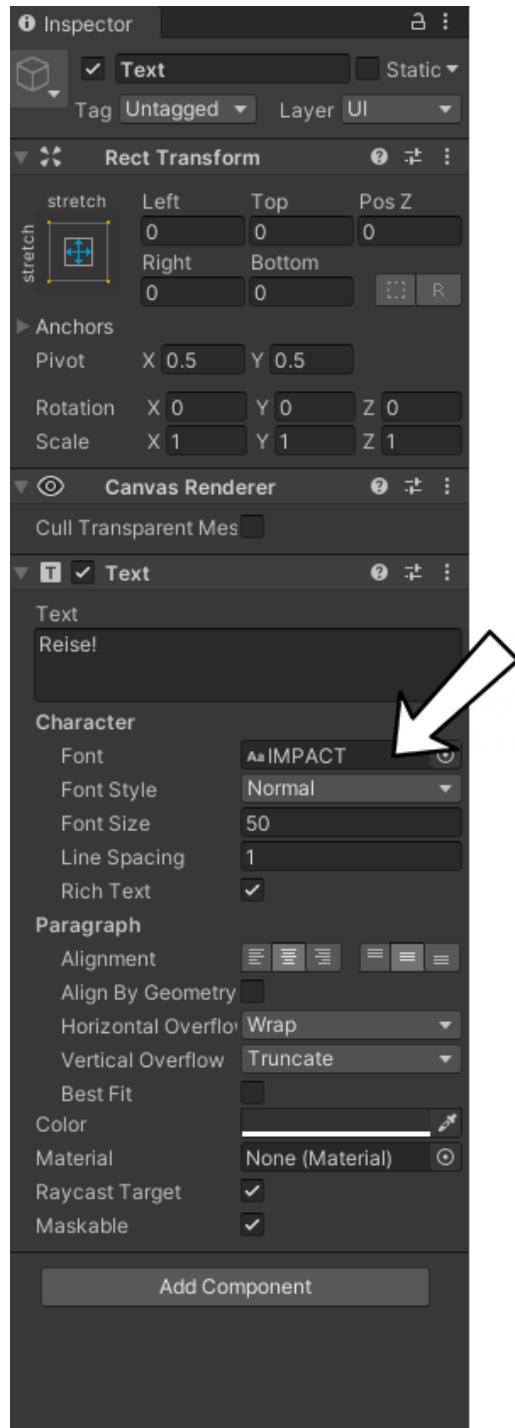
ZUWEISUNG VON HUD IN IMAGE KOMPONENTE

Damit die einzelnen Segmente dafür genutzt werden können, müssen diese als „2D and UI Sprite“ Texture Type importiert werden. Macht man dies nicht, so erhalten die Bilder plötzlich einen schwarzen Hintergrund und erhalten eine andere .meta Signatur. Das hat zur Folge, dass man dieses Sprite nicht auf einer Unity 2D Canvas nutzen kann. Deshalb wählt man nach dem Import den richtigen Texture Typ aus:



IMPORT VON TEXTURE TYP VON 2D SPRITES

Zusätzlich zu den Bildern ist es wichtig, dass der ausgesuchte Font für die Schrift genauso richtig in Unity importiert wird. Dafür muss zunächst sichergestellt sein, dass der Font auf dem PC installiert ist, sodass man ihn in sein Unity Projekt importieren kann. Hat man ihn in sein Projekt importiert, so kann man ihn bei Textfeldern als Font auswählen:



IMPORT VON FONT IN TEXT KOMPONENTE

Modeling

Autoren: Leonie Schwall und Sebastian Pfeiffer

Items modellieren

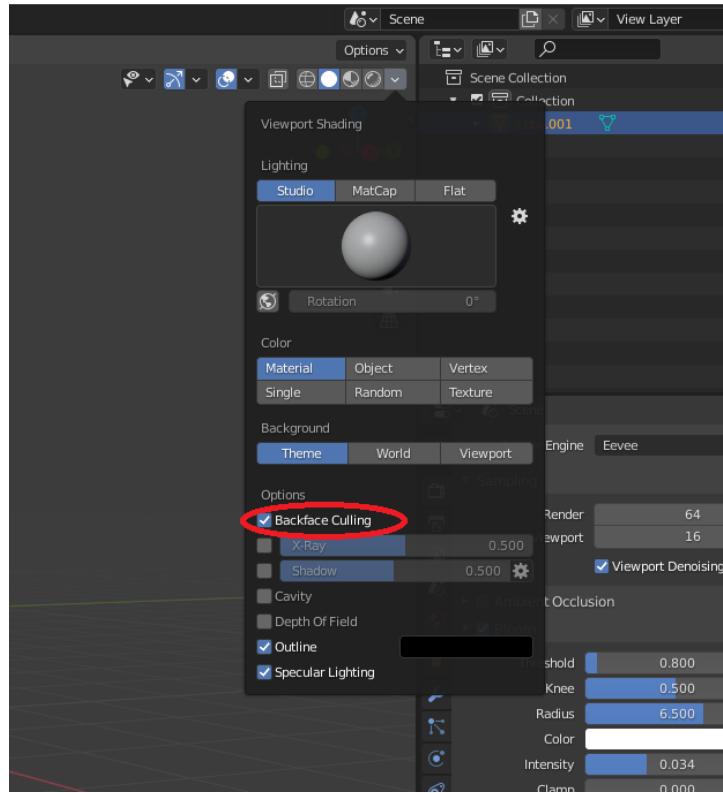
Die letzten Dinge, die in diesem Semester noch neu modelliert wurden, sind die Items für den Marktstand, um diese in das Shopsystem integrieren zu können. Dafür wurden hauptsächlich Zaubertränke und Bücher modelliert. Die Bücher sollen dabei für neues Wissen stehen und die Zaubertränke um Aktionspunkte, die Rüstung und weiteres aufzufüllen.



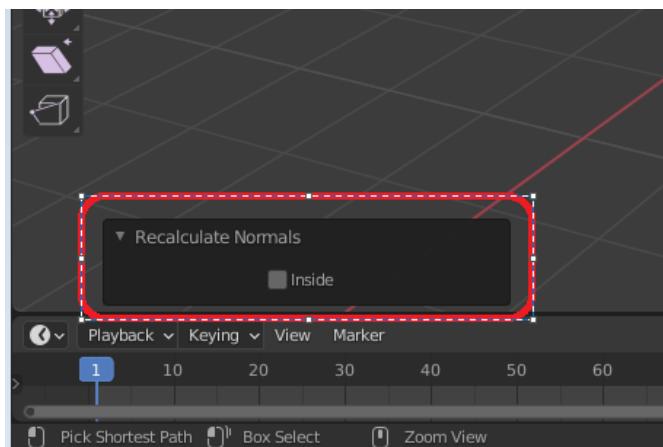
Normalenfehler

Unity betreibt Echtzeitrendering, dadurch werden Flächen nur von einer Seite betrachtet. Anders als in Blender. Sind die Normalen nun nicht richtig in Unity ausgerichtet, entstehen Normalen Fehler. Diese machen sich nur dadurch bemerkbar, dass die Fläche durchsichtig ist und es einem vorkommt, als würde man in das innere eines Models sehen können. Die Fehler entstehen beim Modellieren in Blender, entweder durch fünfeckige Flächen oder durch eine falsche Normalen Ausrichtung. Dies ist aber einfach zu beheben. Zum einen sollte man von Anfang an darauf achten keine fünfeckigen Flächen zu kreieren und zum anderen kann man ein paar Einstellungen vornehmen, um die

Normalen für Unity auszurichten. Dafür muss man unter den Einstellungen das Backface Culling einstellen.

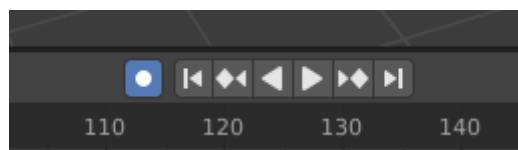


Mit dem aktivieren des Backface Culling, welches man im ausgeklappten Viewport des Shadings findet, werden die Flächen sichtbar die in Unity nicht angezeigt werden können. Um dies nun zu beheben kann man das Modell im Edit Mode auswählen und die Tastenkombination Shift + N drücken, mit dem ein Feld aufgeht, dort darf das Feld für die Normalen nicht ausgewählt sein. Somit müssten die Fehler behoben worden sein.



Umsetzung der Szenen für den Trailer

Für die Umsetzung von Filmszenen in Blender wurden alle einzelnen Modelle, die für eine Szene gebraucht wurden in eine Datei gepackt und dann so ausgerichtet wie sie benötigt wurden. Nach dem Setzen der Kamera und des Lichtes in Blender, kann animiert werden. Dabei müssen ein paar Dinge beachtet werden. Am leichtesten ist es Objekte oder Charakter mit Auto Keying zu animieren, dabei wird automatisch ein neuer Keyframe gesetzt, nachdem das Objekt bewegt wurde.

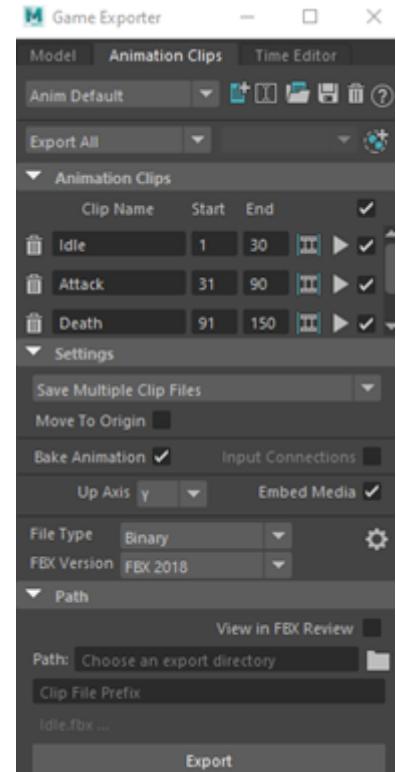


Zu beachten ist hierbei, dass dies nicht immer zum gleichen Zeitpunkt/ Frame passiert, da sonst der Keyframe und somit die Bewegung überschrieben wird. Zudem sollte darauf geachtet werden Auto Keying wieder auszustellen wenn gerade nicht animiert wird, da sonst unbeabsichtigt Objekte Millimeterweise verrücken oder zu einer anderen Stelle springen und das evtl. erst beim Rendern entdeckt wird.

Nach dem Animieren, sollten Proberender von dem Video kurz in Eevee (die schnellste Renderengine in Blender, da Echtzeitrendering) gemacht werden, um zu sehen wie flüssig die Animation in einem Video aussieht und alles so im Bild ist wie man sich das vorgestellt hat. Für die Lichtverhältnisse sollte ein einzelnes Frame ausgewählt werden, in dem man die Lichter am besten erkennen kann und diesen Frame mit Cycles rendern, um dies zu überprüfen.

Model Export

Nach dem Modellieren aller Assets ging es dann um den Export der Assets. Hierzu hatte Maya einen Game Exporter mit dem man Modelle direkt im richtigen Format(.fbx) exportieren kann. Alle Animationen werden hier auch direkt baked, was dafür sorgt dass sie auf dem Skelett gespeichert werden und im nachhinein nicht mehr verändert werden können. Der Game exporter bietet einem auch die Möglichkeit die Animationen und das Model in einem einzigen File zu speichern oder auch in mehreren. Von den Programmierern wurde ein einziges File empfohlen. Hier kam es dann zu Schwierigkeiten da Blender über keinen direkten Game exporter verfügt hatte es auch nicht die Funktion alle Animationen in einem File als fbx zu exportieren. Ein Lösungsversuch dafür war es das Blender File in Maya zu öffnen und dort über den game exporter zu exportieren, was nicht funktioniert hat da die Animation beim Übertragen von Blender nach Maya kaputt gegangen ist. Die Lösung war dann das Blender File direkt nach Unity zu exportieren, ohne ein fbx File daraus zu machen.



Baking

Das Baking einer Animation sorgt dafür dass eine Animation auf das Skelet des Models gespeichert wird, sodass man besser mit den Models in Unity arbeiten kann. Bei dem Baking Vorgang kam es manchmal zu Problemen, wenn das Modell fehlerhafte Oberflächen wie Löcher o.ä. hatte.

Skalierung

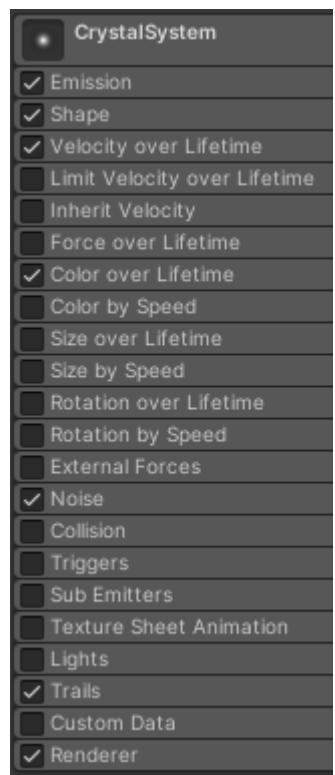
Da Unity eine andere Skalierung Basis hat als Maya und Blender kam es beim Übertragen von Blender/Maya zu Unity zu Skalierung Problemen. Die Modelle wurden in Unity viel zu groß dargestellt und mussten deshalb manuell runter skaliert werden. Da es bei der Skalierung in Unity manchmal zu Problemen führt, wenn man Modelle zu stark skaliert, mussten hier die richtigen Skalierungen in Blender/Maya gefunden werden sodass diese direkt richtig in Unity importiert wurden. Dazu haben wir Beispiels Modelle genommen, die wir aus Unity in der richtigen Größe exportiert haben und dann in Maya/Blender importiert haben. Anhand der Beispiel Modelle haben wird dann die Größe der Aurea angepasst.

VFX

Autor: Sebastian Pfeiffer und Lukas Blea

Partikel in Unity

Dieses Semester war ein großer teil meiner Arbeit die Partikelsysteme für die Aurea Attacken zu erstellen. Die Partikelsysteme wurden in Unity selber erstellt da der Import von Partikel Systemen, die in Blender erstellt wurden, nicht möglich war. Als Basis hatten wir immer die bereits ausgedachten Attacken der Aurea. Wir haben uns dann überlegt wie das Partikelsystem aussehen muss, um der Beschreibung zu entsprechen. Nach der Planung wurde dann ein Partikelsystem erstellt und durch verschiedene Parameter zur gewünschten form gebracht. Meistens haben wir mit Parametern wie: Velocity Over Lifetime oder Rotation. Wenn die Bewegungen dann alle so aussahen wie wir uns das vorgestellt hatten wurde noch die Skalierung und die Farben angepasst.



Probleme

Bei der Skalierung der Partikel kam es dann zu Problemen vor allem wenn die Partikel im AR Modus angezeigt werden sollten hier waren die Partikel deutlich zu groß da diese nicht mit skaliert wurden hier wurde dann erst versucht die Partikel mit Hilfe von selbst geschriebenem Code zu skalieren. Das hat auch nicht perfekt funktioniert. Wir haben dann bemerkt das man in Unity nur den Scaling Mode der Partikel von lokal auf Hierarchie setzen und dann wurden die Partikel auch im AR Modus richtig angezeigt.

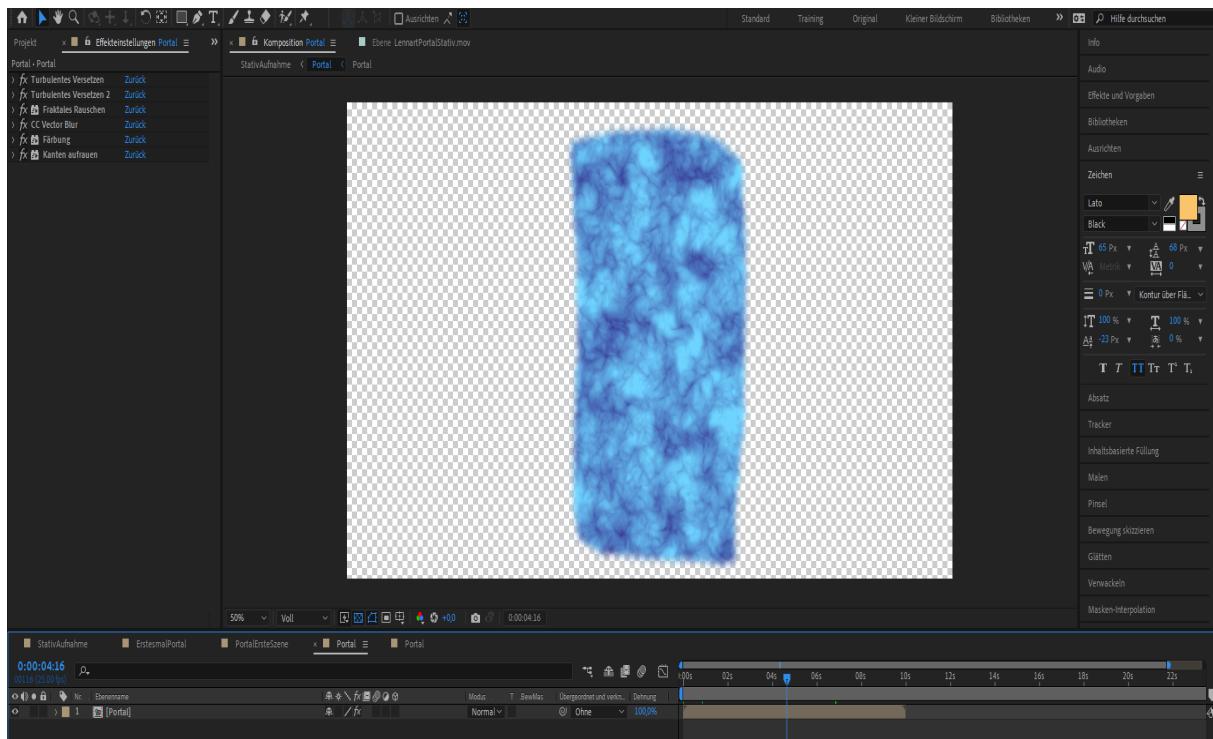
Partikel in Blender

Für den Trailer und Bilder wollten wir dann auch noch Partikel für das Portal einbauen. Da wir die das Rendern in Blender machen wollten und nicht in Unity da die Ergebnisse in Blender besser sind mussten diese Partikel nochmal neu in Blender gemacht werden. Die Vorgehensweise in Blender war ein bisschen anders da man statt mit Velocity Over Lifetime mit wind Objekten oder vortext Objekten arbeiten musste.

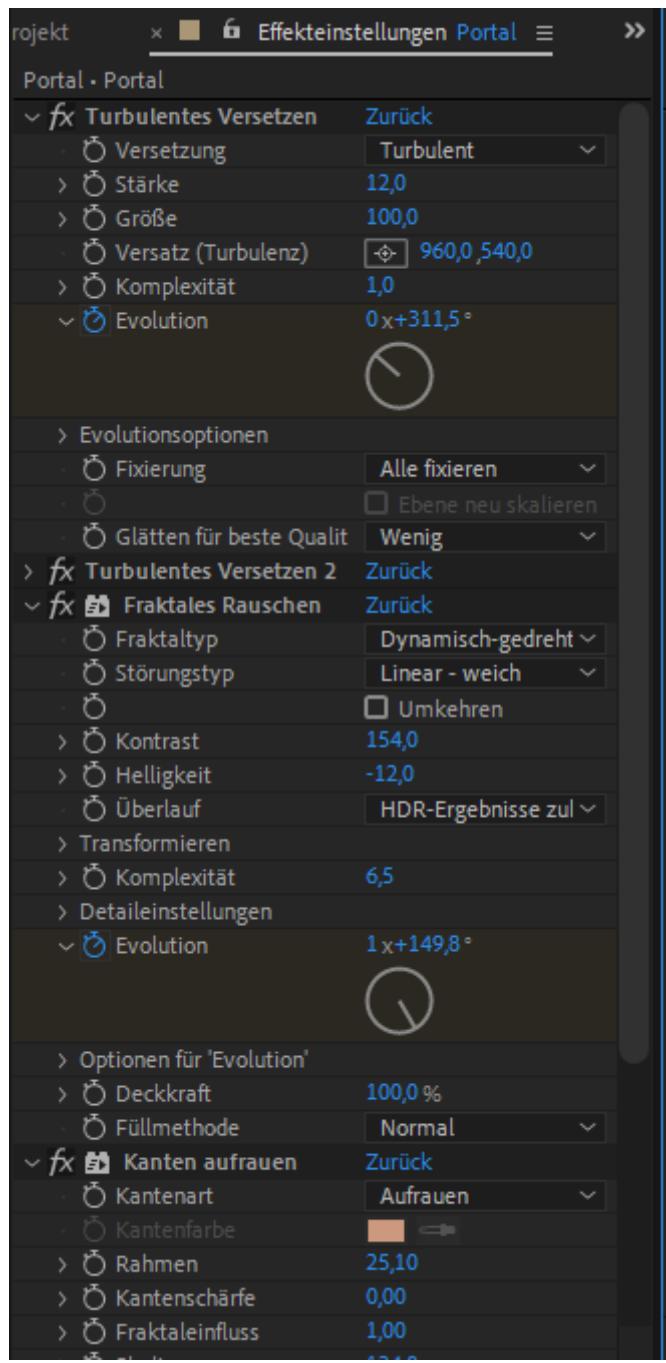


Portal Erstellung

Für die Erstellung des Portals musste ich Adobe After Effects benutzen. Zwar gibt es Vorlagen, bzw. Templates die man benutzen könnte, allerdings war es mir wichtig ein Portal zu haben das ich beliebig an die entsprechenden Szenen anpassen und verändern kann.

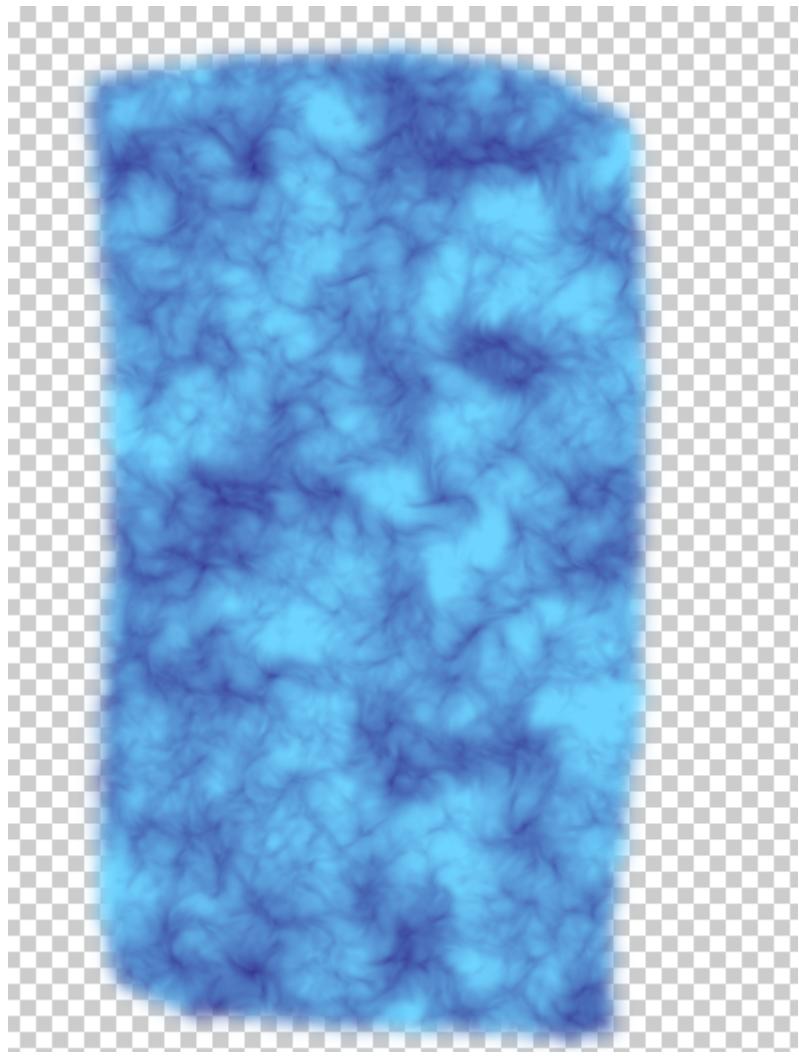


Im Bild zu sehen ist das fertige Portal. Man startet erst damit, eine Viereckige Form auf After Effects zu erstellen. Auf diese Form, wendet man dann Effekte an.



Turbulentes Versetzen steht hier an oberster Stelle. Durch diesen Effekt bleibt die Vierecks-Form nie still, sondern bewegt sich. Dadurch hat man nicht das Gefühl, dass man gerade ein stilles Bild betrachtet sondern ein wirkliches Objekt in der echten Welt.

Fraktales Rauschen gibt dem Portal das nötige Aussehen. Die leichten Akzente und Wellenformen im Portal selbst kommen damit zum Vorschein.

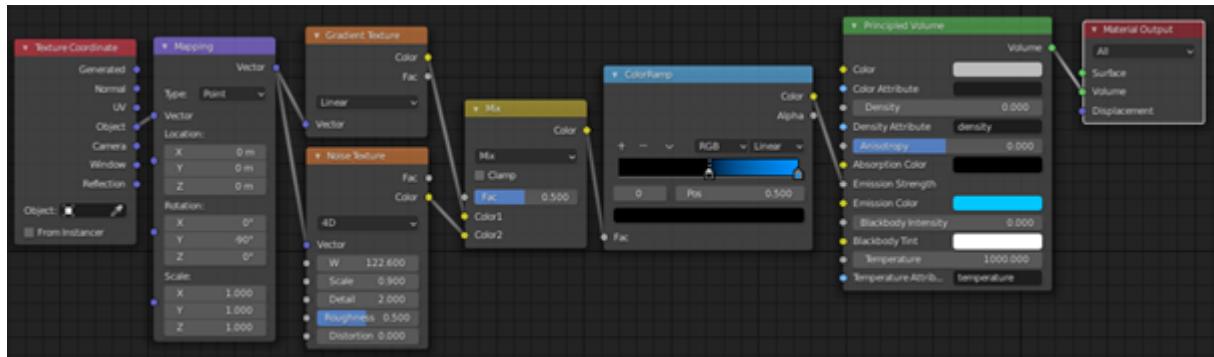


Kanten aufrauen lässt das Portal keine harten, sondern weiche Kanten haben. Durch harte Kanten wird der Betrachter sofort aus der Illusion genommen, dass dieses Portal wirklich in der echten Welt steht, da es so aussieht als würde es nicht dort hingehören. Vor allem an dem Punkt, wenn das Portal mit dem Boden in Berührung kommt, war es wichtig weiche Kanten zu haben um das Portal nicht vom rest abzuheben.

Andere VFX Elemente

Neben der Partikel haben wir auch Nebel modelliert, um den Aufnahmen das gewisse etwas zu geben. Der Nebel entstand durch einen normalen Würfel, der zum Volumen Objekt gemacht wurde durch einige Änderungen an der

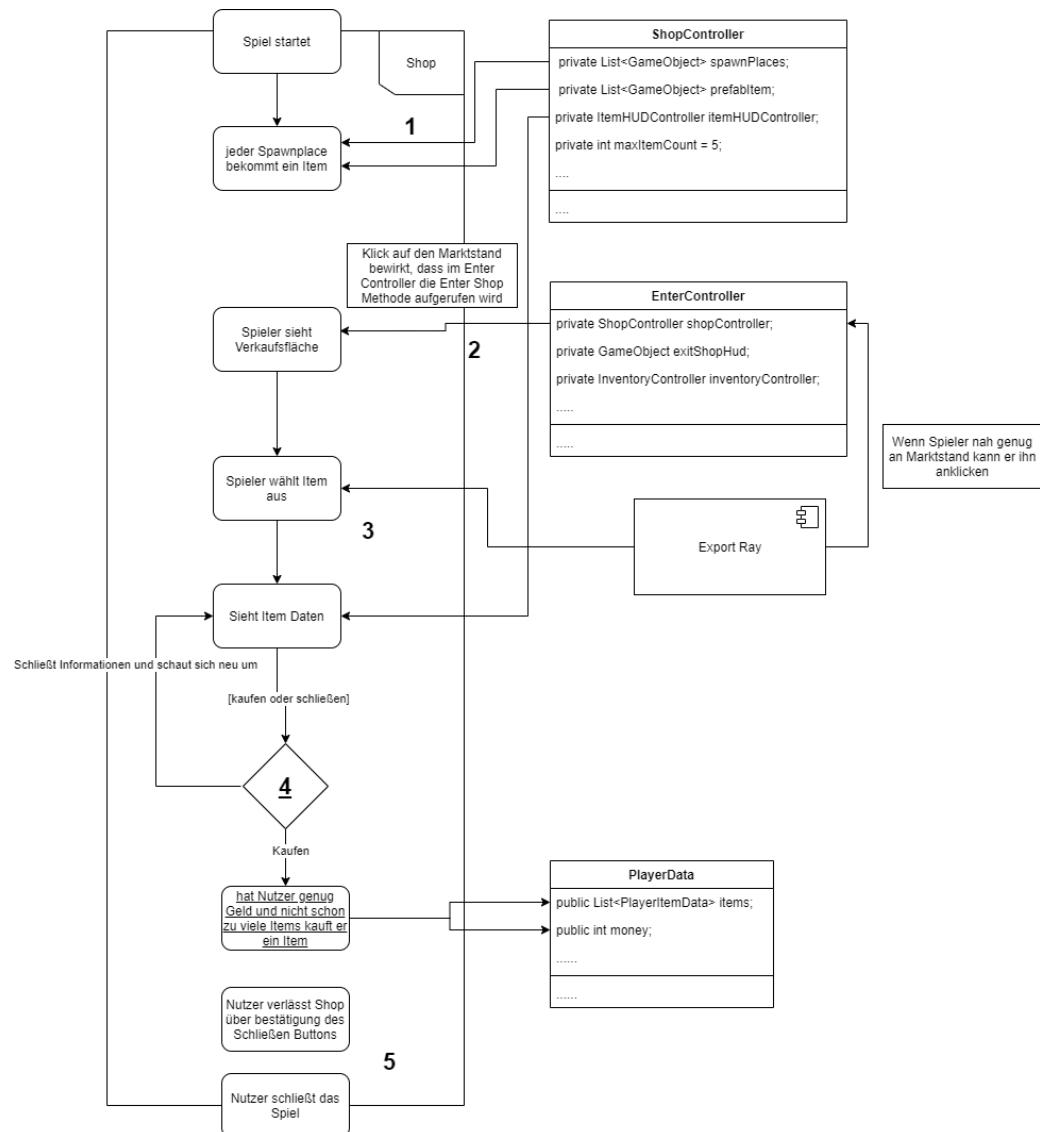
Nodestruktur wurde dann lediglich ein durchsichtiger Nebel, den man auch die Inseln setzen konnte.



Shopsystem

Autor: Nick Häcker

Wir hatten uns dafür entschieden, dass man nicht nur mitsamt seiner Aurea einen Kampf gewinnen kann, sondern auch mithilfe von Gegenständen, die während des Kampfes eingesetzt werden können. Diese Gegenstände kann der Nutzer am Marktstand erwerben. Dazu habe ich eine kleine Skizze erstellt, die eine Mischung aus Klassen- und Aktivitätsdiagramm ist, mit der die Mechanik näher erklärt wird.



ÜBERSICHT ÜBER SHOP MECHANIK

1.Schritt:

Sobald das Spiel gestartet wurde spawnt der Shop auf seinen 4 Spawn Slots pro Slot je ein Item, von dem der Shop Kenntnis hat. Derzeit sind dem Shop die AP - und HP Potion bekannt, sowie ein Buch mit Rezepten für weitere Potions. Die Anzahl der sichtbaren Items bleibt immer gleich, jedoch sollen sich sobald ein Tag rum ist die Items unterscheiden, sodass es täglich unterschiedliche Items gibt.

2.Schritt:



COLLIDER (GELB) UND KOLLISIONS WEGE (ROT) VOM CHARAKTER UND INTERAKTIONSOBJEKten

Steuert der Spieler seinen Charakter nah genug an den Marktstand heran, so kollidieren die Collider des Charakters und des Shops (Collider als gelbe Markierung, Kollision wenn in Bewegung in rote Pfeilrichtung erfolgt) und es wird ein Event ausgelöst, sodass der Spieler die Möglichkeit hat auf den

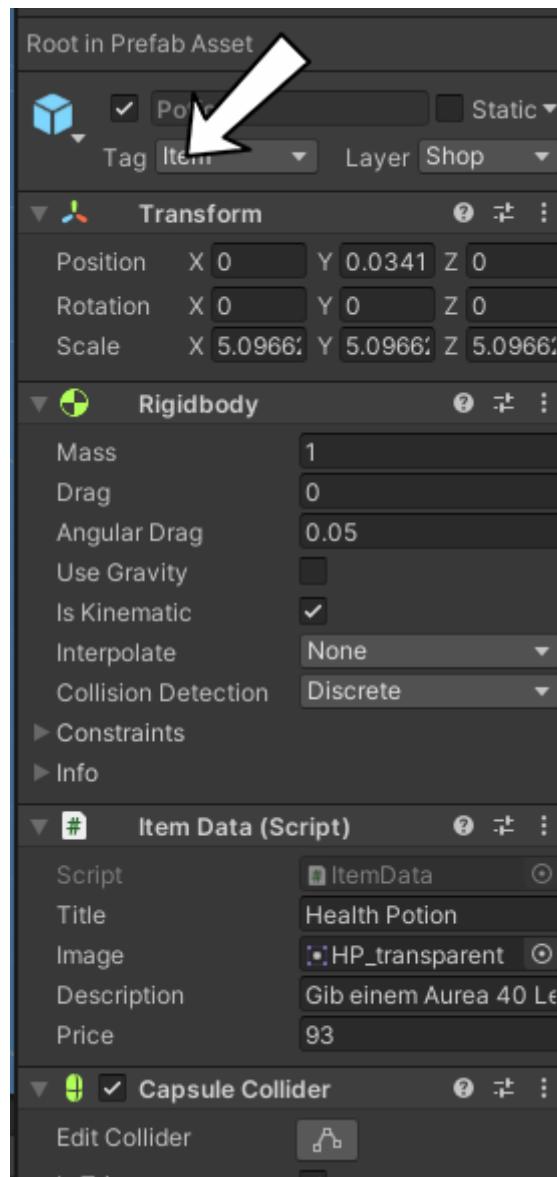
Marktstand zu klicken und sich den Tresen des Marktes näher anzuschauen.



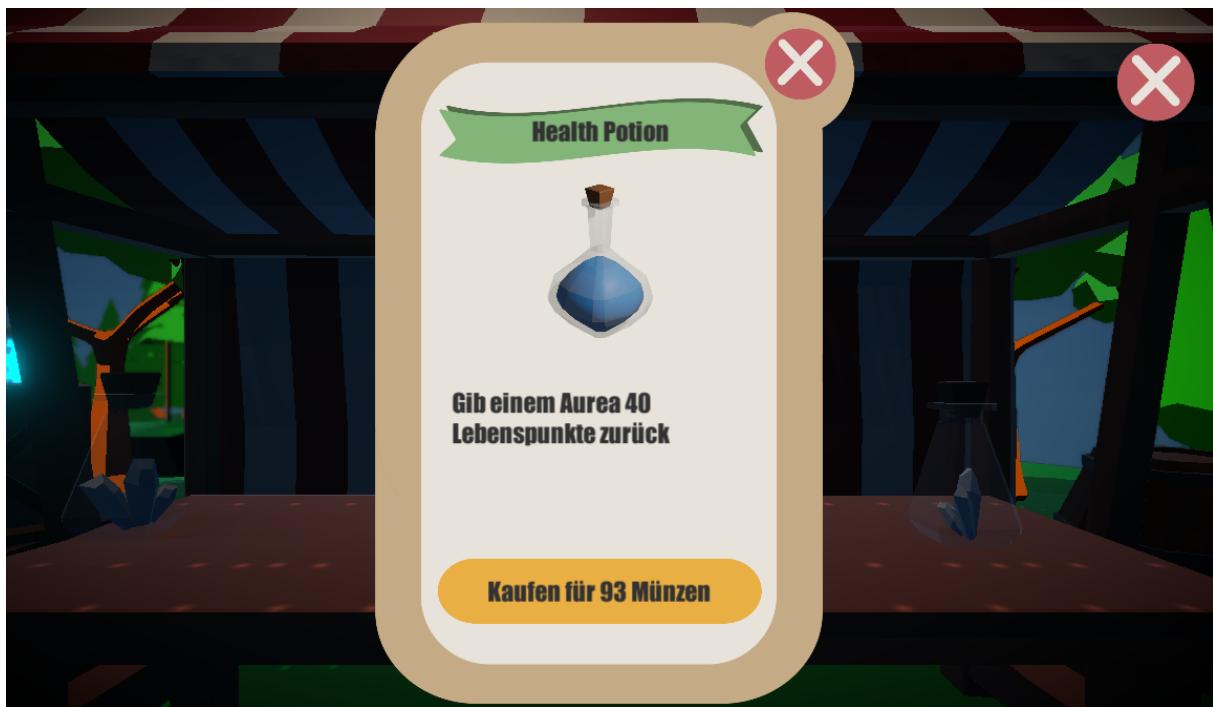
TRESEN MIT ITEMS (GELBE UMRANDUNG: COLLIDER)

3. Schritt:

Die im Shop erhältlichen Gegenstände besitzen alle auch einen Collider (Gelb umkreist). Durch anklicken eines Gegenstandes erhält der Nutzer Einsicht auf den Titel, Beschreibung und Preis. Die einzelnen Gegenstände erhalten auch einen Tag, allerdings keinen Layer Tag für die Kamera, sondern einen Ordnungstag, der der Unity Game Engine sagt, dass das Item ein Item ist. Die durch Touchinput erzeugten Rays schießen auf das Item und erkennen es durch den Tag als Item an.



ITEM ERHÄLT ITEM TAG FÜR DIE ABFRAGE AUF WELCHES OBJEKT DER RAY GETROFFEN IST



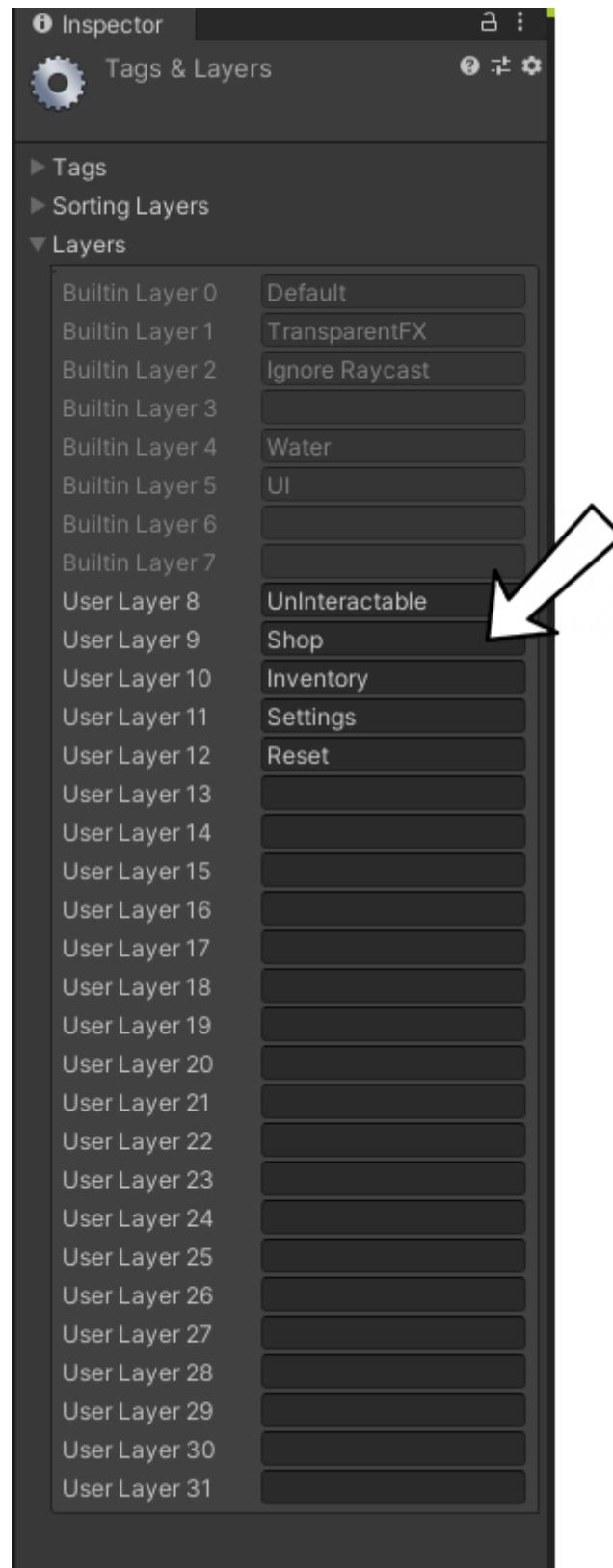
ERSCHEINENDES ITEM HUD

4.Schritt

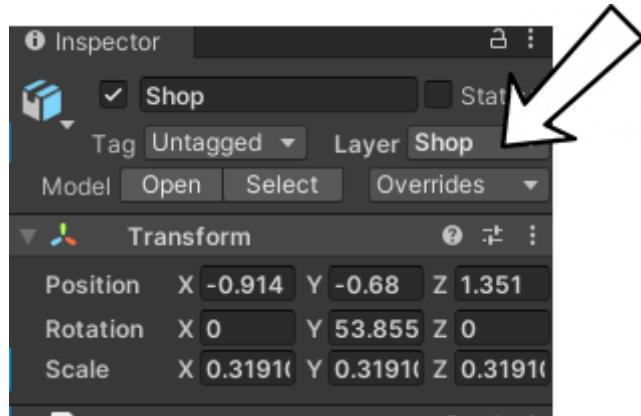
Klickt der Spieler auf den Kaufen Button, so wird der Preis des Gegenstandes vom Konto des Spielers abgerechnet und das Item wird in die Itemliste des Spielers hinzugefügt.

Betrachtungsmodus in AR:

Eine besonderheit des Betrachtungsmodus' im AR Modus ist es, dass alle weiteren Objekte außer der Marktstand und seine Gegenstände ausgeblendet werden. Dies wird mit den Culling Musk Layern der AR Kamera realisiert. Dazu wird zunächst ein Kamera Layer definiert welcher dem Shop als Tag mitgeteilt wird.



SHOP TAG WIRD AUF KAMERA LAYER 9 GESETZT



SHOP OBJEKT ERHÄLT SHOP LAYER ALS MARKIERUNG FÜR DIE

Sobald der Spieler den auf den Marktstand geklickt hat bekommt er nur noch den Marktstand zu sehen. Mit dem Befehl aus dem Bild verschiebt man die derzeitige Culling Mask der Kamera, die die alle Kamera Layer anzeigen lässt, auf den 9. Layer. Man sagt, die Culling Mask wird um 9 Bit verschoben.

```

if (armode == true)
{
    if (exitShopHud != null && skyIslandController != null)
    {
        skyIslandController.SetStaticmode(true);
        exitShopHud.SetActive(true);
        arCamera.cullingMask = 1 << 9;
    }
}

```

CODE SNIPPET ZUR VISUALISIERUNG DER VERSCHIEBUNG DER CULLING MASK DER KAMERA

Im Ar Modus würde man dadurch den Shop als einziges Objekt sehen, da aber die Skalierung aller Objekte in Unity auf Minimum gesetzt wird, ist der Marktstand und sein Tresen nur sehr klein zu betrachten.

Augmented Reality

Autor: Dennis Hawran

Die Augmented Reality (AR) Funktionen waren eine der wichtigsten Komponenten des Spiels. Um diese zu implementieren gab es zwei Möglichkeiten. Zum einen hätte der Service von Vuforia benutzt werden können. Vuforia ist in der Industrie weit verbreitet und bietet eine Vielzahl an Möglichkeiten um AR Funktionen zu implementieren. Die zweite Option war das In-Built System von Unity mit dem Namen AR Foundation. Wir haben mit beiden Systemen einen Prototyp gebaut um die Vor- und Nachteile abwägen zu können. Entschieden haben wir uns dann für AR Foundation aus mehreren Gründen.

- Da AR Foundation ein Unity eigenes Package ist, steht es kostenlos zur Verfügung.
- Es ist robuster, da es direkt für Unity-Anwendungen entwickelt wurde.
- Da es Open-Source ist wären benötigte Änderungen einfach umzusetzen
- Wie alle Unity-Packages ist es Plattformunabhängig und somit auf jedem Gerät nutzbar.

Anschließend mussten wir uns überlegen welche Einstellungsmöglichkeiten wir dem User bieten möchten. Damit das Spiel in jeder möglichen Situation spielbar ist, mussten wir dafür sorgen dass der User Position und Größe der Inseln anpassen kann. Somit kann AUREA gespielt werden wenn der User sich in einem kleinen Raum oder draußen auf einem großen Feld befindet.



AUREA IM FREIEN



AUREA AUF DEM SCHREIBTISCH

Eine der größten Herausforderungen war es über verschiedene Szenen hinweg die Insel an die gleiche Stelle im Raum zu projizieren. Der Grund dafür war, dass AR Foundation sich bei jedem Szenenwechsel neu kalibriert. Dies konnten wir umgehen indem wir das gesamte Spiel in einer Szene programmiert haben. Jedoch hat das zu einem Trade-off zwischen Ladezeit am Anfang und robustheit

des Spiels geführt. Außerdem konnten wir so die Ladezeiten im Spiel selbst auf null reduzieren da alle Inseln beim Start geladen wurden und dann nur Ein- oder Ausgeblendet werden mussten.

In Unity steht eine Entfernungseinheit für einen Meter in der realen Welt. Dies macht es normalerweise sehr einfach, Objekte maßstabsgerecht zu modellieren und intuitiver mit Physikalischen Berechnungen umzugehen. Für uns war dies jedoch nicht vorteilhaft, da die Objekte dann sehr groß in die echte Welt projiziert wurden. Um dies zu umgehen hätten wir zwei verschiedene Inseln im Spiel einbauen können. Eine normal große Insel für den Fall dass die AR Funktionen ausgeschaltet sind und eine kleine für die Projektion in die echte Welt. Dies hätte jedoch dazu geführt, dass entweder der komplette Spielstand auf beiden Versionen synchronisiert werden muss, oder der User nur außerhalb eines Kampfes zwischen den Modi wechseln kann. Gelöst haben wir dieses Problem, indem wir jedes einzelne Objekt in der Szene herunterskaliert haben. Dadurch mussten wir dann zwar auch noch die Geschwindigkeit des Spielers anpassen, jedoch sonst nichts mehr.

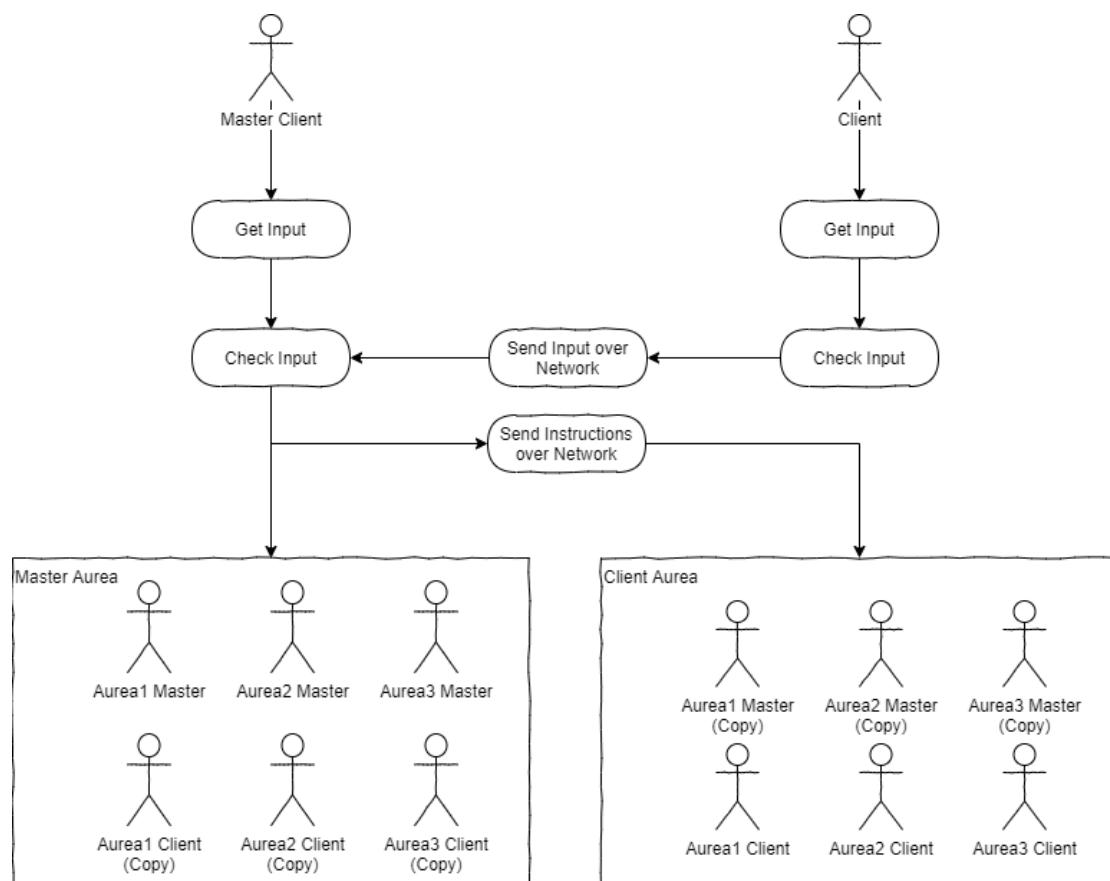
Netzwerk

Autor: Dennis Hawran

Wie bei der Implementierung des AR Systems hatten wir auch bei der Netzwerkkomponente verschiedene Möglichkeiten zur Auswahl. Auch hier haben wir wieder verschiedene Prototypen gebaut um die Möglichkeiten vergleichen zu können. Die erste Wahl war wieder Unitys In-Built System. Dies heißt MLAPI. Da Unity aktuell ein neues System entwickelt, wurde die Weiterentwicklung von MLAPI leider gestoppt. Durch die fehlende weiterentwicklung von MLAPI haben wir uns schnell gegen die Verwendung dieses Systems entschieden, da viele Dinge einfach nicht funktioniert haben oder sehr aufwändige Workarounds gebraucht hätten. Danach standen noch 2 Services von Drittanbietern zur Verfügung. Der erste Service heißt Mirror und ist ein Open-Source Projekt der Unity Community. Der zweite Service ist von der Firma Photon und nennt sich PUN. Aus Kostengründen und auf Grund der Flexibilität durch die Open-Source Anwendung hätten wir uns für Mirror entschieden. In unserer Situation allerdings haben wir uns für den Service von Photon entschieden. AUREA wurde ursprünglich nicht als Multiplayer Spiel konzeptioniert. Im zweiten Semester wurde dann entschieden das dieses Feature implementiert werden soll. Dadurch musste die Einbindung schnell gehen. Es hat sich herausgestellt ,dass sich die Einbindung mit PUN leichter gestaltet und schneller zu Ergebnissen führt. Deshalb haben wir uns letzten Endes für PUN entschieden. Da wir schnell einen Prototypen brauchten, haben wir in Kauf genommen dass der Service in der kostenlosen Variante nur für weniger als Zehn User zur Verfügung steht.

Da es unser erstes Spiel mit Multiplayer Funktionalität war mussten wir erstmal lernen umzudenken. In klassischen Spielen übernimmt jedes Script eine Aufgabe und es ist immer eindeutig wo und wann dieses Script ausgeführt wird. In einem Multiplayer System ist ein anderes Vorgehen erforderlich. Die Skripte existieren jetzt mehrere male verteilt auf den verschiedenen Geräten. Jedes Script muss nun also erst herausfinden, ob es sich auf dem Master Client befindet oder ob es lediglich eine Kopie ist welche synchronisiert werden muss. Wenn ein User gerade nicht der Master Client ist, muss sein Input zuerst

verarbeitet werden und dann beim Master Client nachgefragt werden ob dieser Input zulässig ist. Wenn ja gibt der Master Client die Information an alle kopierten Instanzen weiter. Außerdem muss der Input bei einem Multiplayer Spiel mehrere male überprüft werden, da zwischen dem Input auf dem Gerät und dem Auswerten auf dem Master Client Zeit vergangen ist. Dies hat zur Folge das sich der Status des Spiels in der Zwischenzeit geändert haben könnte.



ABLAUF DES NETZWERKES

Kampfsystem

Autor: Dennis Hawran

Wie bereits im Kapitel [Netzwerk](#) beschrieben wurde, haben wir uns erst im zweiten Semester dazu entschieden ein Multiplayer Spiel aus AUREA zu machen. Zuerst war geplant daraus ein Offline Game zu machen in welchem man gegen Gegner spielt welche mittels Reinforcement Learning trainiert wurden. Diese Änderung hat das Kampfsystem deutlich betroffen und es musste komplett neu geschrieben werden. Lediglich das Skillssystem konnte übernommen werden.

Das Skillssystem war eine sehr große Herausforderung. Als dieses System erarbeitet wurde, waren unsere AUREA noch nicht konzipiert. Dies bedeutete, dass noch nicht klar war welche Fähigkeiten die AUREA später besitzen werden. Aus diesem Grund musste das Kampf und das Skillssystem so generisch wie nur möglich gehalten werden. Wir haben daraus ein Modifier System entwickelt welches sich komplett selbst verwaltet. Das bedeutet dass ein Skill nur seinen Besitzer und sein Ziel kennen muss.

```
public abstract void Use(Damage _dmg);
public abstract bool IsTargetValid(Aurea _aurea, Aurea _sender);
public abstract bool CheckTargets(List<Aurea> _targets, Aurea _sender);
```

DIE WICHTIGEN FUNKTION UNSERER SKILLKLASSE

Der Skill macht dann alle Änderungen, setzt sich selbst alle nötigen Event Listener und zerstört sich am Ende auch selbst. Durch das Event System von C# konnten Skills auch auf alle Änderungen der Ziele oder seines Besitzers reagieren.

```
public Action<Damage> StartAttack;
public Action<Damage> BeforeGettingHit;
public Action<Damage> AfterGettingHit;
public Action<int> TookDamage;
public Action<float> BeforeChangeLifepoints;
public Action ChangedLifepoints;
public Action<Aurea> Died;
public Action<Aurea> Selected;
public Action<List<Aurea>> ChangedTargets;
public Action SkillCancled;
public Action GotHit;
```

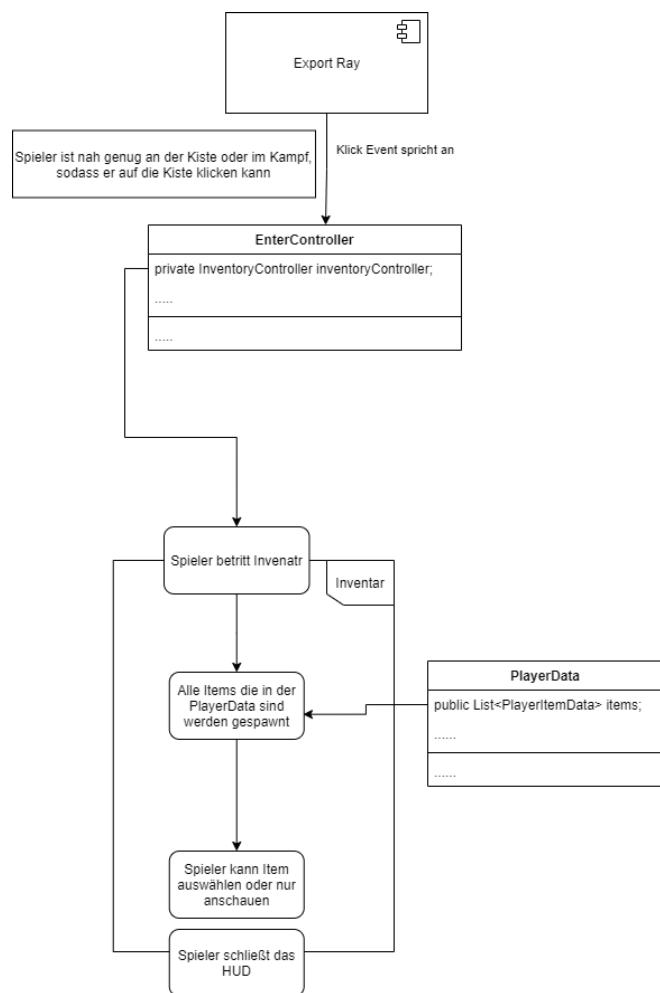
EINE AUSWAHL AN EVENTS AUF WELCHE EIN SKILL REAGIEREN KANN

So war es beispielsweise möglich dass ein Skill nur dann in Kraft tritt, wenn ein bestimmter Gegner besiegt wurde oder wenn eine bestimmte Anzahl an Runden vergangen war. Uns war es wichtig solche Skills implementieren zu können damit die User gezwungen sind so taktisch wie möglich zu spielen.

Inventarsystem

Autor: Nick Häcker

Wie der Spieler neue Gegenstände kaufen kann wurde bereits beim [Shopsystem](#) näher beleuchtet. Nun behandeln wir das Verwalten und Nutzen von Items:



ÜBERSICHT ÜBER DIE INVENTAR MECHANIK

1. Schritt:

Steuert der Spieler seinen Charakter nah genug an die Truhe heran, so wird wieder ein Event, ähnlich dem Shopsystem ausgelöst, wodurch der Nutzer die Möglichkeit erhält durch das Klicken auf die Truhe sein Inventar zu öffnen.



SCREENSHOT DER INVENTARANSICHT

2.Schritt:

Der Inhalt des Inventars wird von den verschiedenen Gegenständen bestimmt, die der Nutzer bereits gekauft hat. Er erhält nun die Möglichkeit einzelne Items auszuwählen.

Befindet sich der Nutzer in einem Kampf gegen einen anderen Nutzer, so erhalten der Nutzer oder seine Aurea die beschriebenen Effekte. Die Gegenstände funktionieren genauso wie die Skills der Aurea als Modifier und sind für sich selbst verantwortlich.

Soundsystem

Autor: Jonathan Schnee

Wir haben uns Sounds auf der Seite Envato Elements herausgesucht. Da Lukas eine Subscription hat dafür sind diese alle direkt Lizenziert und herunterladbar. Die Sounds werden dann in Unity eingefügt und mit einem selbstgeschriebenen Audiocontroller abgespielt wird.



Hier sieht man die Einstellungen. Man kann zwischen Hintergrund und SFX Musik entscheiden SFX sind Sounds wie zum Beispiel das Angriff Geräusch eines Aurea diese kann man dann im Programmcode mit dem Play Befehl ausführen, indem man den Namen als Parameter mitgibt. Des Weiteren kann man den Pitch eines Sounds einstellen und die maximale Lautstärke des Sounds. Die Lautstärke kann man dann auch im Spiel anpassen, diese wird dann gespeichert damit man nicht immer von neuem seinen Sound anpassen muss