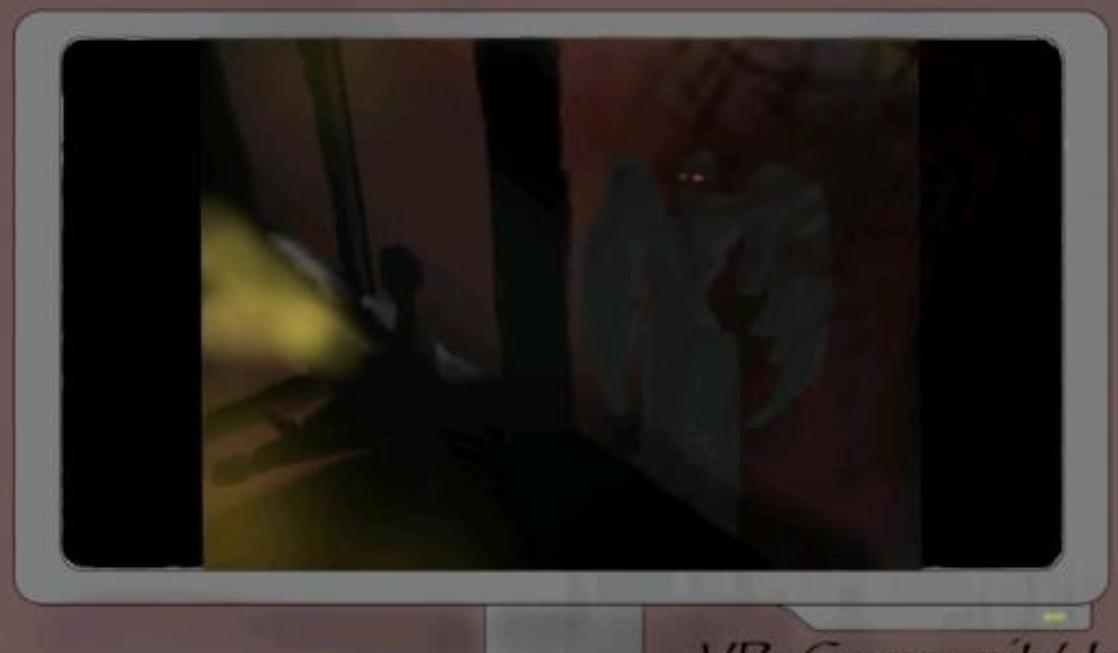


Nightmare

on Bahnhofsstreet



VR-Game mit Unreal

GE-LAB BERICHT

VR-PROJEKT MIT UNREAL

Verfasser:

Dan Eisenkrämer, Feier Jiang, Syafiqah Husna Md Sazali, Simon Kompaß, Marc von Glahn, Niklas Weinhart

Betreuer:

Prof. Dr. Bernd Dreier

Arbeit vorgelegt am:

10.02.2021

Durchgeführt an der:

Fakultät für Informatik

INHALTSVERZEICHNIS

INHALTSVERZEICHNIS	3
1 EINLEITUNG	6
2 DIE ENTSTEHUNG DER IDEE.....	6
2.1 Die ersten Treffen.....	6
2.1.1 Zoom-Meeting.....	6
2.1.2 Erstes Discord-Meeting.....	6
2.1.3 Erstes Offline-Treffen.....	7
3 DAS GAME DESIGN.....	7
3.1 Planung	7
3.1.1 Suche von Referenzmaterial	8
3.2 Skizzen	9
3.2.1 Entwurf des Spiel-Level-Designs.....	10
3.2.2 Digitale Skizzen.....	10
3.2.3 Skizze der Taschenlampe	10
3.2.4 Skizze der Statue.....	11
3.2.5 Hacker-Grundriss	11
3.3 Game Atmosphäre.....	12
3.3.1 Stimmungsbrett	13
3.4 Medien	14
3.4.1 Sound-Dateien	14
3.4.2 Intro und Outro-Video	15
3.4.3 Video-Konzept	16
3.4.4 Erstellung des Intro-Videos	16
3.4.5 Videoschnitt	17
3.4.6 Fertigstellung des Intro-Videos	17
3.4.7 Filmaufnahmen.....	17
3.4.8 Trailer	18
3.5 Spieler-Inventar	19
3.5.1 Inventar-Symbole	19
3.6 Zusammenarbeit mit den anderen Teams	20
3	

3.7	Gestaltung des Posters	21
4	GAMEPLAY	22
4.1	VR-Spieler Gameplay	22
4.1.1	Bewegungssteuerung	22
4.1.2	Taschenlampe	23
4.1.3	Tür	24
4.1.4	Einsammeln von Gegenständen	26
4.1.5	Problemlösung: Spieler hebt ab	28
4.1.6	VR-Spieler HUD	29
4.1.7	Schlüsselvergabe in S.011	30
4.1.8	Interaktion mit Schalter am Sicherungskasten	32
4.1.9	Interaktion mit den Puppen	33
4.1.10	Animation der Tür	34
4.1.11	Implementierung Musik und Sounds	35
4.1.12	Triggern der Ereignisse	37
4.2	PC-Spieler/Hacker Gameplay	38
4.2.1	Kameraimplementierung	38
4.2.2	Hidden Actor Liste	40
4.2.3	Hackermenü	40
4.2.4	Menu Items	42
5	MODELLIERUNG	47
5.1	Vorraussetzungen	47
5.2	Szenerie und Umgebung	47
5.2.1	Basislevel	47
5.3	Materialien: Gebäude	48
5.3.1	Fußboden in den Vorlesungssälen	48
5.3.2	Wände in den Vorlesungssälen	49
5.3.3	Decke in den Vorlesungssälen	50
5.3.4	Fußboden im Hauptgang	50
5.4	Licht	51
5.5	Nebel	52
5.6	Dekorationen	53
5.6.1	Plakat	53

5.6.2	Raum Beschriftung & „KEIN EINGANG“ Schild	54
5.6.3	Notausgangsschild	54
5.7	Modellierung raumfüllender Objekte.....	54
6	FAZIT	62
6.1	Dan Eisenkrämer	62
6.2	Syafiqah Husna Md Sazali	63
6.3	Feier Jiang.....	63
6.4	Simon Kompaß.....	64
6.5	Niklas Weinhart	64
6.6	Marc von Glahn	64
7	QUELLENVERZEICHNIS	66
7.1	Audioverzeichnis.....	66
7.2	Videoverzeichnis	67
7.3	Abbildungsverzeichnis	67
8	ZUSÄTZLICHE DOKUMENTE	68
9	ERKLÄRUNGEN	69
9.1	Selbstständigkeitserklärung.....	69
9.2	Ermächtigung	70

1 EINLEITUNG

Dan

Während ein Projekt zu Beginn immer sehr groß und nicht zu bewältigen erscheint, hilft es, wenn man klein beginnt und sich erst einmal um grundlegende Funktionalitäten kümmert. Wichtige Fragen, die man sich dafür stellen sollte, sind: „Was ist essenziell für dieses Spiel oder generell Spiele dieser Art?“ und „Was soll der Spieler hinterher grundsätzlich können?“ Es schadet auch nicht, sich an dieser Stelle schon Gedanken zu machen, welche Probleme durch die Art und Plattform des Spieles entstehen können.

2 DIE ENTSTEHUNG DER IDEE

2.1 DIE ERSTEN TREFFEN

2.1.1 ZOOM-MEETING

Husna

Bei unserem ersten Treffen hielten wir ein Zoom-Meeting mit unserem Professor ab, um zu besprechen, worum es bei dem Projekt gehen wird. Er erklärte uns, was seine Rolle ist und am Ende des Treffens teilten wir uns primäre Aufgabengebiete zu. Feier und Husna wurden zum Design-Team, Niklas und Marc bildeten das Tech Team und Simon und Dan kümmerten sich um die VR-Programmierung. Das Design-Team war für die Erstellung des Game Design Dokuments zuständig. In der ersten Diskussion kümmerten wir uns um verschiedene Themen bezüglich des Designs des Spiels.

2.1.2 ERSTES DISCORD-MEETING

Zu Beginn des Projekts beschlossen wir, ein kleines Treffen in Discord abzuhalten, um zu besprechen, welche Art von Spiel wir machen wollten. Dieses Treffen dauerte nicht lange und am Ende des Treffens waren wir uns einig, welches Spiel wir machen wollten: Ein Horror-VR-Spiel, welches an unserer Universität spielen sollte. Wir beschlossen auch, dass es eine interessante Idee wäre, wenn das Spiel aus zwei verschiedenen Perspektiven von zwei verschiedenen Spielern dargestellt werde. Die eine wäre die des "VR-Spielers", dessen Aufgabe es wäre, aus der Einrichtung zu entkommen, und die andere wäre die des "Hackers", dessen Aufgabe es wäre, den "VR-Spieler" daran zu hindern, aus der Einrichtung zu entkommen. Bevor das Treffen endete, beschlossen Feier, Marc und ich, dass wir uns an diesem Sonntag bei Marc zu Hause treffen würden, um das Gameplay des Spiels weiter zu diskutieren.

Feier

Am 9. Oktober war unser erstes Treffen. An diesem Tag definierten wir das allgemeine Genre dieses Spiels. Um den VR Spieler in seiner Egoperspektiv immersiv zu sein, entschieden wir, dass wir ein Horrorspiel machen möchten. Unter den vielen Vorschlägen (wie zum Beispiel Krankenhaus, Kindergarten, und so weiter) einigten wir uns darauf, das S-Gebäude von der Hochschule Kempten als Spielort nehmen.

Um mehr Konflikt zu bilden, legten wir den VR Spieler als „der Held“, und den PC-Spieler als „der Schatten“ des Spiels fest. Anstatt einer Kooperation zwischen den Spielern ließen wir sie gegeneinander antreten.

2.1.3 ERSTES OFFLINE-TREFFEN

Husna

Am 11. Oktober fuhren Feier und Husna zu Marc. Dort angekommen, besprachen wir das Gameplay-Konzept noch ein wenig. Wir hatten auch angefangen, das Game Design Dokument zu entwerfen. Wir hatten eine grobe Skizze, wie das Spiel aussehen würde, da es sich im Gebäude S und im zweiten Stock befand. Das Problem war jedoch, dass wir nicht wussten, welche Art von Rätseln für den VR-Spieler gelegt werden sollten und ob diese Aufgaben auch in den Befehlsvorrat des Hackers integriert werden konnten oder nicht. Außerdem mussten wir einen Weg finden, wie das Spiel den Hacker nicht zu übermächtig macht, wenn er den Fluchtweg des Spielers unterbricht. Wir entschieden uns dann für ein Limit für die Anzahl der Befehle, die der Hacker ausgeben kann. Dann blieb nur noch die Frage, welche Rätsel wir in das Spiel implementieren sollten. Nach einer Weile entschieden wir uns für eine Liste von Rätseln, die wir für den VR-Spieler verwenden können. Nachdem wir die letzten Teile des Entwurfs für das Spieldesign-Dokument fertiggestellt hatten, beschlossen wir, das Treffen zu beenden. Anschließend luden wir das Dokument online ins Gitlab hoch, damit die anderen Mitglieder sehen und kommentieren konnten, was wir für unser Spiel verwenden konnten und was nicht.

Feier

Am nächsten Wochenende setzten wir, das Design Team, uns mit Marc aus dem Tech-Team zusammen und die erste vorläufige Version unseres Game Design Dokumentes erarbeitet. Wir beschlossen die Zimmer S.011 und S.012 als Hauptspielorte des Spiels, weil die Zimmer für uns bekannt waren. Eine Veränderung des Stils der vertrautesten Szenen von allen kann den Spieler mehr Angst auslösen. Wir stellten einige Ideen fest, in welchem Raum das Rätsel stattfinden sollten. Und die Lösung der Rätsel sollten auch nah liegen. Das Ziel des VR Spielers war, das Rätsel zu lösen und die Zimmer zu verlassen. Im Gegenteil hatte der PC-Spieler das Hauptziel, die Flucht des VR Spielers durch Interaktion mit der Umgebung zu verhindern.

Aber die Balance des Spiels war am Anfang schwer zu meistern. Da das ganze Spiel hauptsächlich auf VR Spieler ausgerichtet war, wie man es für PC-Spieler auch fesselnd und nicht langweilig machen konnte, führten wir viele Diskussionen mit den anderen. Viele der allerersten Ideen, die wir am Anfang für PC-Spieler dachten, änderten und optimierten wir nachdem Diskussionen.

Husna

Ein Problem, das in den frühen Phasen der Entwicklung des Spiels auftrat, war, dass einige der Befehle für den Hacker als unbrauchbar erachtet wurden, da sie den Spieler nur am Weiterkommen hindern würden und keine Elemente enthielten, die zum "Spaß" des Rätsels beitragen würden. Daher wurden diese Befehle aus dem ursprünglichen Gameplay entfernt und durch einige andere Befehle ersetzt.

3 DAS GAME DESIGN

3.1 PLANUNG

Husna

Nachdem das anfängliche Dokument hochgeladen war, machten wir weitere Fortschritte, nachdem wir besprochen hatten, welche Elemente für und welche gegen das Spiel sprachen. Etwas, auf das wir uns jedoch einigten, war die Tatsache, dass das

Spiel nicht zu lange sein sollte, da es sonst zu viel Aufwand in der Erstellung für uns bedeuten würde. Daher entschieden wir, die Idee, dem Spieler und dem Hacker zu viel Hintergrundgeschichte zu geben, zu streichen und uns stattdessen mehr auf die Horror- und Gameplay-Elemente des Spiels zu konzentrieren.

3.1.1 SUCHE VON REFERENZMATERIAL

Husna

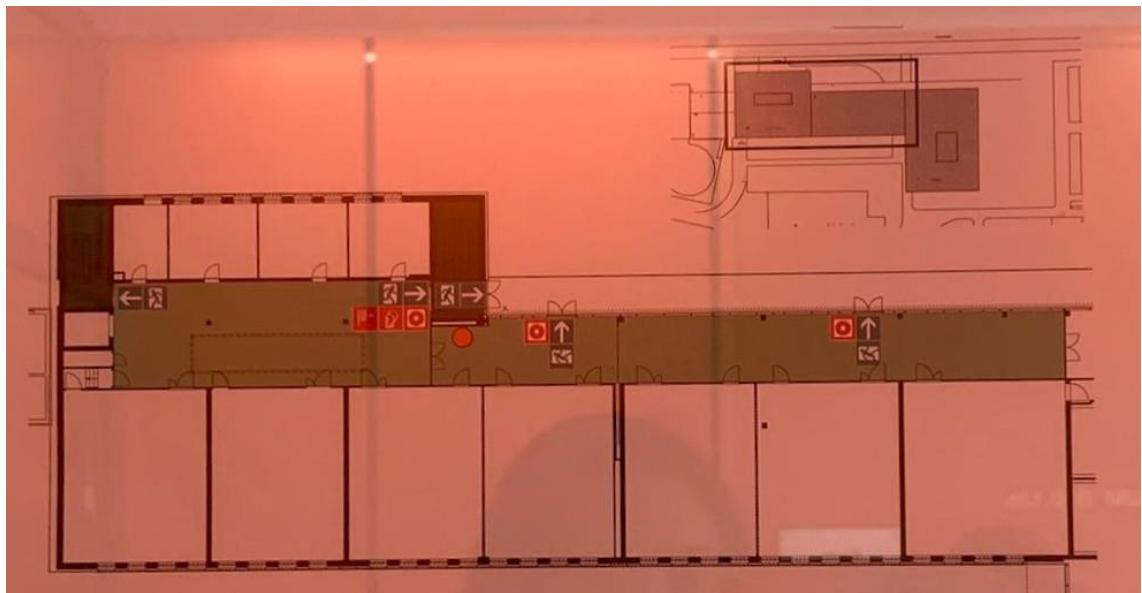
Wir beschlossen, dass einige Referenzbilder notwendig waren, damit die anderen Mitglieder besser verstehen konnten, welche Art von Spiel uns vorschwebte. Das Design-Team suchte dann ein paar Bilder, die wir dann der Gruppe mitteilten, damit die anderen ein Bild davon erhielten, was wir gemacht hatten.

Eine andere Sache, die gemacht werden musste, war es, Bilder von der Hochschule zu machen. Diese Bilder wurden dann als Referenzen für die anderen Teammitglieder verwendet. Es handelte sich um Bilder von verschiedenen Szenerien der Schule sowie um geschlossene Bilder der Objekte, die in der Schule vorhanden waren. Diese wurden dann entweder für Texturen oder einfach als Referenzen beim Entwerfen einer Szenerie der Schule verwendet.

Niklas

Für die Umgebung wurde die Hochschule Kempten, speziell das S-Gebäude in dem ein Großteil der Informatik – Game Engineering Vorlesungen stattfinden, ausgewählt. Mit dieser Entscheidung gab es eine spezifische Szenerie und das Ziel diese detailliert zu imitieren.

Dafür diente ein Fluchtplan des Erdgeschosses von Gebäude S als Vorlage. Mit diesem konnte der Grundriss des Levels eindeutig abgepaust werden.



Trotz der bekannten Umgebung erwies sich schnell, dass die Erinnerungen an das Innenleben nicht ausreichen, um es authentisch wiederzugeben. Zum Beispiel brauchte es für die korrekte Deckenhöhe mehrere Anpassungen, da diese geschätzt werden musste.

Daher brauchten wir zusätzlich weitere Bilder von der Umgebung und Einrichtung eine klare Vorstellung, wie das finale Level aussehen sollte.



3.2 SKIZZEN

Feier

Wir einigten uns auch grob auf den Grundriss des Game-Levels, den allgemeinen Stil des Monsters, die Aktivität des Spielers und das Design der Szene. Dadurch machten wir ein paar Skizzen, um den anderen besser zu verstehen.



SKIZZE VON S.011

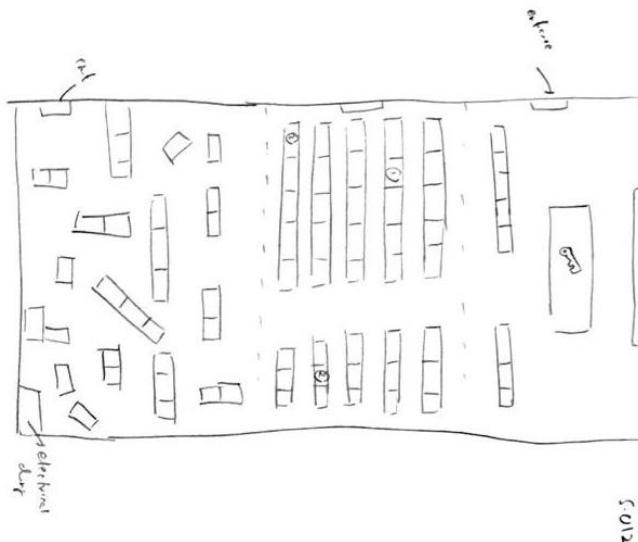


SKIZZE VON S.012

SKIZZE VON FLUR

Husna

Für das Spiel wurden auch einige Skizzen benötigt, also zeichneten wir ein paar Skizzen des Grundrisses und ein paar andere Skizzen, die zeigten, wie das Spiel aussehen würde, von Hand. Die Skizzen wurden dann an die Gruppe weitergegeben.



Grundriss-Skizze

3.2.1 ENTWURF DES SPIEL-LEVEL-DESIGNS

Das Game-Level-Design des Spiels wurde zwischen den Teammitgliedern besprochen und wir notierten uns, was die Skizze des Game-Play sowie das Game-Level-Design sein würde. Das Game-Design-Dokument wurde während dieser Zeit ebenfalls aktualisiert, um sicherzustellen, dass mehrere Elemente des Spiels in diesem Dokument aktuell waren.

3.2.2 DIGITALE SKIZZEN

Nach ein paar Wochen erkannte das Design-Team, dass handgezeichnete Skizzen nicht mehr genügten. Daher wechselten wir die Plattform und begannen stattdessen, digitale Skizzen und Bilder zu erstellen. Die Programme, für die wir uns entschieden, waren Adobe Photoshop und SketchBook. Wir mussten uns auch für einen Dateityp entscheiden, den wir beide gemeinsam nutzen konnten, da dies die spätere Bearbeitung erleichterte.

3.2.3 SKIZZE DER TASCHENLAMPE

Ein früher Entwurf, wie der "Lichtstrahl" der Taschenlampe aussehen sollte, um zu zeigen, wie hell die Taschenlampe sein würde. Es wurden auch Skizzen gezeichnet, wie die Beleuchtung der Räume S.011 und S.012 aussehen würde. Diese gaben einen Einblick, wie schummrig, beziehungsweise hell die Räume und welches Farbschema die Beleuchtungen haben sollten.



Taschenlampe und Raumskizze

3.2.4 SKIZZE DER STATUE

Erste Entwürfe, wie die Statue aussehen sollte, wurden ebenfalls angefertigt und der Gruppe mitgeteilt. Wir entschieden uns für das Konzept eines weinenden Engels. Die Statue würde aus Marmor bestehen und eine Frau darstellen. Das Gesicht der Statue würde Tränen haben, was ein weinendes Gesicht darstellen würde. Die Statue hatte auch Flügel, passend zum "Engel"-Konzept. Das Design wurde dann ein wenig geändert, so dass die Statue keine Beine haben würde, sondern nur der Torso und das Gesicht der Statue gezeigt werden würde.



Statue Skizze

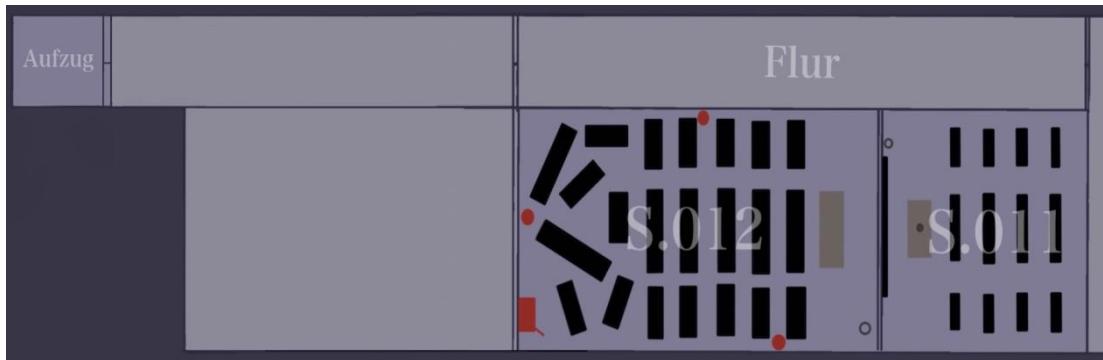
3.2.5 HACKER-GRUNDRISS

Ein grober Grundriss in der Sicht des Hackers, um zu zeigen, welche Befehle und wo die Befehle sein würden, wurde entworfen und an die Gruppe weitergegeben. Der Grundriss enthielt Symbole, die zeigten, wo der Hacker interagieren konnte, um bestimmte Befehle auszuführen, sowie eine grobe Ansicht des Bildschirms des Hackers.

Ein weiterer Grundriss wurde erstellt und an die Gruppe weitergegeben. Allerdings ergab sich ein Problem, da es ein kleines Missverständnis bei der Bodengestaltung des Spiels gab. Das Design-Team war der Meinung, dass die Position des Fahrstuhls auf der gegenüberliegenden Seite des Spielplans lag, als es eigentlich der Fall sein sollte. Daher wurden entsprechende Änderungen am Grundriss vorgenommen.

Feier

Wir diskutierten auch mit den anderen Teams, wie das Interface des PC-Spielers aussehen sollte. Wir zeigten in den Zeichnungen, welche Interaktion der PC-Spieler an welcher Stelle durchführen kann. In Kombination mit den Hacker Kommandos aus dem Game Design Dokument konnte man einfacher damit arbeiten. Weil es eine begrenzte Anzahl von Hacker Kommandos geben sollte, definierten wir „Actionpoints“ für den Hacker. Damit konnte der Hacker nicht unbeschränkt Kommandos in einem Raum durchführen. Es war großartig, dass die Technical Teams all unsere Ideen verwirklichen konnten.



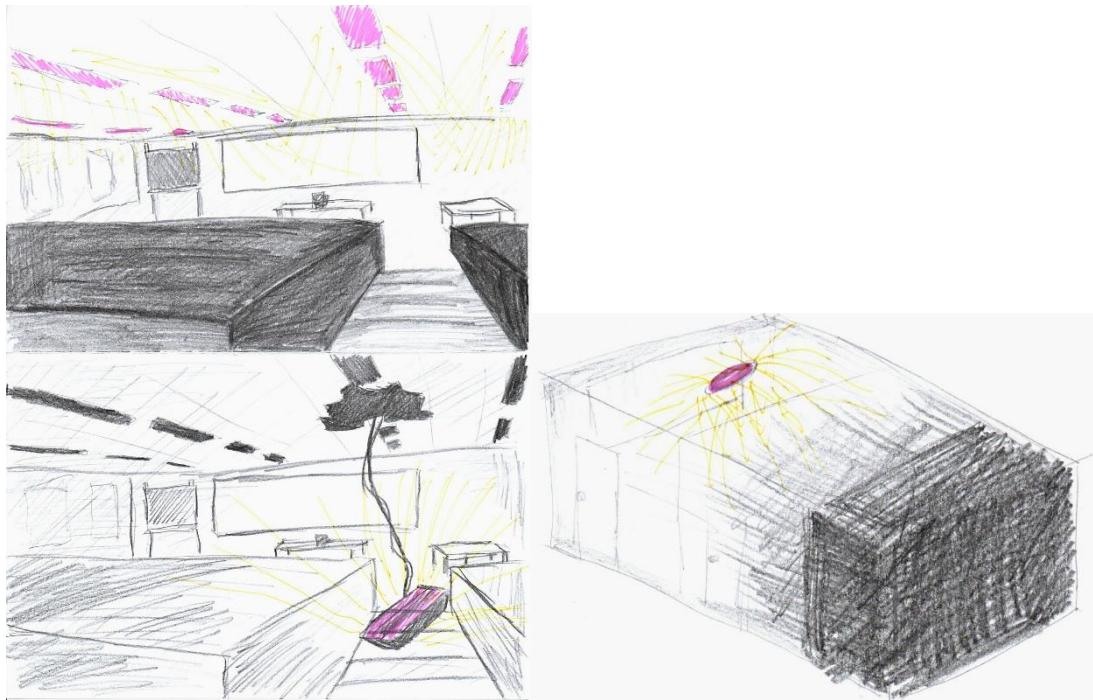
FLURPLAN

3.3 GAME ATMOSPHÄRE

Feier

Wir bevorzugten es, die gruselige Atmosphäre durch das Wetter, die Musik und die Änderung der Beleuchtung zu erzeugen anstatt durch einen „Jump-Scare“.

Die Beleuchtung spielte in unserem Spiel eine große Rolle. Die Veränderung der Lichtstärke fand sowohl in S.011 als auch in S.012 statt. Wir fanden, dass die Dunkelheit dem VR Spieler mehr Druck geben konnte. Gleichzeitig hatte der PC-Spieler in der Dunkelheit einen größeren Spielraum.



von links nach rechts: Lichtquelle S.011/S.012, Lichtquelle

Wir wollten auch die Atmosphäre durch die Veränderung des Wetters ausdrücken. Jedes Mal, wenn der VR-Spieler aus verschiedenen Räumen herauskam, änderte sich das Wetter entsprechend: von sonnig bis neblig, und am Ende bis zu dunkel. Es bedeutete sowohl den Verlauf der Zeit innerhalb des Spiels als auch den inneren psychologischen Zustand des VR-Spielers. Wir verstärkten die Atmosphäre weiter durch die Hintergrundmusik und die Engelskulptur. Jedes Mal, wenn der VR-Spieler erfolgreich aus dem Raum entkam, kam er in die nächste Phase des Spieles. Die Hintergrundmusik und die Posen der Skulptur wurden auch allmählich immer verrückter verändert. Dafür entschieden wir uns für vier verschiedenen Posen der Skulptur und zeichneten diese. Wir wollten dem VR-Spieler auf unterschiedlichen Arten mehr Druck geben.

3.3.1 STIMMUNGSBRETT

Husna

Damit die anderen Mitglieder einen Einblick in die Gesamtatmosphäre des Spiels bekommen konnten, erstellten wir ein Moodboard, das aus einigen Skizzen und Bildern bestand, wie die Spielatmosphäre, die Beleuchtung, das Farbschema und das Gesamtgefühl aussehen sollten. Das Moodboard selbst wurde auch so bearbeitet, dass es einen "Hacker"-Vibe hatte.



Stimmungsbrett

3.4 MEDIEN

3.4.1 SOUND-DATEIEN

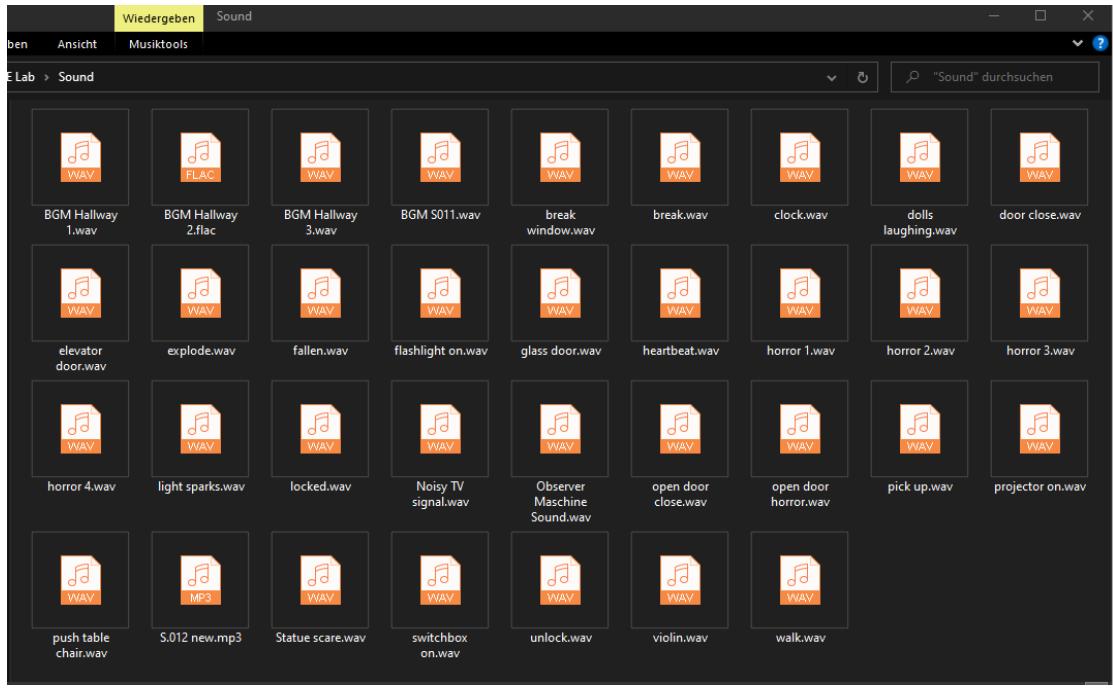
Husna

Wir begannen mit der Suche nach Sounddateien für das Spiel. Zu den Sounddateien gehörten Hintergrundmusik, kleine Soundeffekte für die Bewegungen und Interaktionen der Spieler sowie ein Game-Over-Soundeffekt. Die gesuchten und gegebenen Sounddateien waren alle lizenzenfreie Sounddateien.

Feier

Aufgrund der unterschiedlichen Umgebungen und Spielverläufe haben wir passende Musik für die jeweiligen Situationen ausgewählt.¹ Wir verglichen mit der Interaktionsliste des Spielers und der Environment Liste verglichen, um sicherzugehen, welche Interaktionen des Spielers mit Szenen und Gegenständen Geräusche erzeugen konnten, welche Dinge ohne Interaktion einen konstanten Ton erzeugen konnten und welche Räume eine Hintergrundmusik benötigten. Die Sounds und Musik sollten auch ein grausiges Gefühl in das Spiel bringen. Wir schrieben dann die Übersicht von Sounds und Musik im Game Design Dokument, um die Technical Team damit besser arbeiten zu können.

¹ ChinaZ - <https://sc.chinaz.com>, earo - <https://www.earo.com/>, freesound - <https://freesound.org>



SOUNDS UND MUSIK

3.4.2 INTRO UND OUTRO-VIDEO

Husna

Wir begannen mit der Erstellung des Intros und Outros des Spiels. Das Outro sollte ein einfaches Video sein, das die Worte "Game Over" zeigte, während das Intro ein halbminütiges Video sein sollte, das ein wenig zeigt, wie das Spiel aussehen sollte, aber es würde komplett aus Skizzen bestehen, die das Design-Team zeichnete.

Wir fingen an, das Videokonzept zu besprechen und auch, wie der Kunststil des Videos sein sollte, da wir unsere Skizzen und Bilder synchronisieren mussten, damit der Trailer nicht "seltsam" wirkte. Nachdem wir uns auf einen Art-Style einigten, besprachen wir den Ablauf des Videos und die Handlung. Das Video sollte Szenen der Hochschule enthalten, und so entschieden wir, dass mehrere Szenen, die Orte wie den Aufzug der Hochschule und den Flur der Hochschule zeigten, zum Video passen würden.

Feier

Wir haben auch einige Videos gemacht, zum Beispiel das Game-Over Video, das Intro Video und das Trailer Video. Seitdem wir uns entschieden hatten, ein 2D-Video als Intro Video zu machen, zeichneten wir dafür über 30 digitalen Zeichnungen, um ein fließendes Video zu bearbeiten (mit einer chinesischen Videobearbeitungssoftware 视频编辑王). Wir einigten uns darauf, dass das Intro Video mit einer schwarzen, roten und grauen Farbpalette gemacht werden sollte, um eine mysteriöse und gefährliche Atmosphäre zu bilden. Der Inhalt des Videos sollte möglichst den gesamten Spielablauf und die symbolischen Monster des Spiels enthalten, damit die Leute, die das Spiel noch nicht gespielt hatten, ein besseres Verständnis für das Spiel bekommen konnten. Um bei Anderen einen starken Eindruck zu hinterlassen, kombinierten wir unser Video mit einer rhythmischen Musik. Wir versuchten, das Video mit der Musik in einem schnellen Tempo zu realisieren. Deshalb begrenzten wir die Länge des Videos auf eine Zeitspanne von 30 Sekunden bis zu einer Minute. Wir fügten auch einige Filter ein, die visuell auffällig sind, damit die Aufreihung der verschiedenen Zeichnungen in diesem Video nicht zu langweilig war.



BEARBEITUNG DES INTRO VIDEOS

3.4.3 VIDEO-KONZEPT

Husna

Wir begannen, im Internet nach Videoinspirationen zu suchen und suchten nach Bildern, die auch als Referenzbilder verwendet werden konnten. Nachdem wir zu einer gemeinsamen Entscheidung gekommen waren, kamen wir zu dem Schluss, dass das Video ein kurzes 30-Sekunden-Video sein würde, wobei das Hauptkonzept ein wenig gruselig sein und einige Horror-Elemente enthalten sollte. Eine Sache, die wir früh beschlossen, war die Tatsache, dass das Video auf unbestimmte Zeit Bilder und Szenen mit der "Statue" enthalten sollte, da diese Statue eine große Rolle im Ende des Spiels spielte. Wir entschieden uns auch für ein Farbschema, das hauptsächlich aus Schwarz, Weiß und Rot bestand, um das Grusel- und Horrorgefühl des Videos noch zu verstärken.

3.4.4 ERSTELLUNG DES INTRO-VIDEOS

Wir begannen mit dem Zeichnen und Skizzieren von Bildern für das Intro. Am Anfang hatten wir noch keinen konkreten Plot, also zeichneten wir grundsätzlich Dinge, von denen wir dachten, sie würden zum Intro-Video passen. Damit sich unsere Zeichnungen oder Skizzen nicht überschnitten oder gegeneinander verstießen, sprachen wir zuerst miteinander ab, ob ein Bild oder eine Zeichnung, die wir vorhatten zu zeichnen, zum Video passte oder nicht.

Wir zeichneten dann weiter mehrere Bilder für die Zeichnung und fingen an, uns gegenseitig zu fragen, ob man bei bestimmten Zeichnungen helfen könnte, da es immer wieder vorkam, dass die Besonderheiten einer Zeichnung, wie z. B. die Schattierung oder so, unsicher waren.



Intro Video Zeichnungen

3.4.5 VIDEOSCHNITT

Wir begannen, mit der Videobearbeitung zu experimentieren, da wir beide fast keine Erfahrung auf diesem Gebiet hatten. Dabei kam auch das Programm Adobe Photoshop zum Einsatz. Dabei wurde auch ein Testvideo erstellt, aber da es nur ein Testvideo war, war das Ergebnis nicht sehr günstig und das Video wurde dann entsorgt. Weitere Experimente mit der Videobearbeitung wurden ebenfalls durchgeführt.

3.4.6 FERTIGSTELLUNG DES INTRO-VIDEOS

Die endgültigen Bilder für das Intro-Video wurden schließlich fertiggestellt und zu einem Intro-Video zusammengestellt. Wie zuvor beschlossen, sollte das Intro-Video eine unheimliche und gruselige Stimmung vermitteln, aber nur eine halbe Minute lang sein. Nachdem das Video der Gruppe vorgestellt wurde, war die Resonanz erfreulicherweise ebenfalls sehr positiv.

3.4.7 FILMAUFAHMEN

Zur abschließenden Präsentation brauchten wir noch einige Aufnahmen, um einen Trailer zusammenzustellen. Simon übernahm diese Aufgabe. Dieser besaß die nötige Ausrüstung und Umgebung, um anschauliche Bilder für den Zusammenschnitt zu liefern.



3.4.8 TRAILER

Nachdem das Intro-Video erfolgreich geschnitten wurde, wurden wir mit dem Schnitt des Trailer-Videos beauftragt. Die Videoclips für dieses Video wurden von den anderen Teammitgliedern zur Verfügung gestellt und wir mussten den Schnitt des Videos übernehmen. Zum Glück war das zur Verfügung gestellte Videomaterial mehr als ausreichend, um ein interessantes Video zu erstellen.

Feier

Simon und Marc hatten für das Trailer Video eigene Spielaufnahmen aufgenommen. Sie gaben uns eine Menge In-Game-Aufnahme und „Real Life“-Aufnahme. Durch die Videos von allen konnten die Zuschauer in die Entstehung des Spiels eintauchen. Aber wegen der Längenbegrenzung der ausgewählten Musik konnten wir leider nicht alles verwenden. Wir schnitten die Videos in kleine Fragmente und bauten die Fragmente in neuer Reihenfolge wieder zusammen. Wir wählten die „Real Life“-Aufnahmen, welche mit typischen Aktionen (zum Beispiel Tür öffnen, Taschenlampe aufheben, Hacker Kommandos durchführen und so weiter) bevorzugt aus und verbanden diese mit den angepassten Fragmenten aus der Game-Aufnahme, um dem Zuschauer ein spannendes und immersives Gefühl zu geben. Wir woben auch einige bewegte Aufnahmen der Spielszenen dazwischen, um eine gruselige Atmosphäre zu schaffen. Wir versuchten das Video mit einer Musik im Schnelltempo gut zu kombinieren. Wir bemühten uns, dass das Video jeder Person ihren ursprünglichen, persönlichen Charakter und Stil im Bearbeitungsprozess beibehalten konnte. Am Ende erhielten wir ein Video mit einer Laufzeit von einer Minute und 33 Sekunden.



Bearbeitung des Trailer-Videos



Trailer Zeichnung

3.5 SPIELER-INVENTAR

Husna

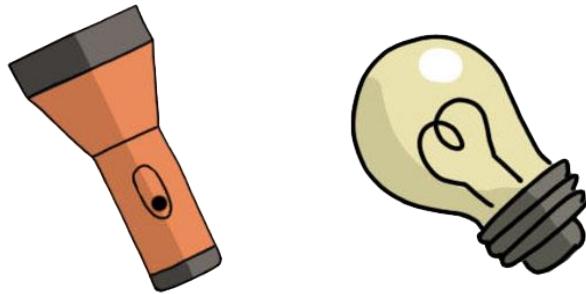
Wir begannen damit, ein Inventar für den Spieler zu zeichnen, um die Gegenstände anzuzeigen, die der Spieler bei sich trug. Wir zeichneten ein paar Entwürfe und entschieden uns schließlich für einen, den wir an die Gruppe schickten. Das anfängliche Inventarbild wurde jedoch verworfen, da wir der Meinung waren, dass es besser wäre, ein kleines Inventar zu haben, das ständig am unteren Rand des Spielerbildschirms angezeigt wird, im Gegensatz zu einer Inventarseite. Dies wurde dann berücksichtigt und eine neue Inventarseite gezeichnet. Eine Skizze, wie das Inventar auf dem Bildschirm des Spielers aussehen sollte, wurde ebenfalls gezeichnet und mit der Gruppe geteilt.



Spieler-Inventar Zeichnung

3.5.1 INVENTAR-SYMBOLE

Die Icons der Objekte, die im Inventar verfügbar sein sollten, wurden ebenfalls gezeichnet. Wir mussten uns dann auch für einen Stil für die Objekt-Icons entscheiden. Wir entschieden, dass ein einfacher "Comic"-Stil besser zum Spiel passte als ein harter "realistischer" Stil. So wurden die Icons in diesem Stil gezeichnet. Die Icons wurden dann auch in der Gruppe verteilt.



Inventar-Symbole

3.6 ZUSAMMENARBEIT MIT DEN ANDEREN TEAMS

Feier

Wir versorgten auch andere Teams (VR Team und Technical Team) mit den Materialien, die sie für das Spiel benötigten. Zum Beispiel die Fotos von der Hochschule, welche man bei Modellierung brauchte, die digitalen Zeichnungen die im Raum S.011 für Rätsel benutzt wurden, die Symbolen für das Inventar und „Actionpoints“ der PC Spieler, die einzelnen Sounds und Musik für die Interaktionen des Spielers und der Umwelt und den Flurplan.

Um dem Modellierungsteam einen besseren Einblick in die Konstruktion einiger Szenen und Gegenstände zu bieten, zeichneten wir anschließend die digitalen Zeichnungen mit Adobe Photoshop und SketchBook von iPad. Die Zeichnungen mit Farben konnten auch viele Details wie zum Beispiel die Form und die Texturen zeigen.



links: Beispiel Zeichnung, rechts: Symbol

Im Verlauf der Programmierung und Modellierung der anderen Teams korrigierten und aktualisierten wir unser Game Design Dokument nach den Bedürfnissen der anderen Gruppen. Beispielsweise veränderten wir den Flurplan, tauschten die PC-Spieler Kommandos und so weiter. Da es keinen großen Einfluss auf den Spielablauf hatte, dachten wir uns, dass es einfacher wäre, die Änderungen von unserer Seite vorzunehmen als von dem Technical Team.

Husna

Es wurden letzte Änderungen am Spieldesign-Dokument vorgenommen. Da das Dokument seit geraumer Zeit nicht mehr bearbeitet worden war, musste viel editiert und neu geschrieben werden. Außerdem wurden mehrere Aspekte des Spiels im Vergleich zum ursprünglichen Dokument geändert und diese Änderungen mussten auch in diesem Dokument aktualisiert werden.

3.7 GESTALTUNG DES POSTERS

Feier

Wir dachten viel über das Poster nach, wie wir die wichtigen Elemente, die im Spiel vorkamen, in einem Poster verpacken konnten. Und es sollte auch möglich sein, dass andere Leute auf den ersten Blick erkennen konnten, dass dieses Spiel sowohl von VR Spieler als auch von PC-Spieler gleichzeitig gespielt wurde. Wir zeichneten zwei Zeichnungen, eine für PC-Spieler und eine für VR Spieler, und kombinierten sie zu einem Poster.

Husna

Wir begannen mit ersten Entwürfen für ein Poster, das für das Spiel gemacht werden sollte. Die Idee des Plakats unterschied sich nicht sehr vom Intro-Video, da es eine gruselige und eine Art von Horror-Stimmung ausstrahlen sollte. Allerdings sollte das Plakat sowohl die Hacker-, als auch die Spielerperspektive des besagten Spiels beinhalten und nicht nur die einer Person. Während des ersten Treffens zur Besprechung des Plakats besprachen wir, wie schon zuvor, zuerst den Stil und das Konzept des Plakats. Als das erledigt war, erstellten wir einen einfachen Entwurf des Posters.

Weitere Fortschritte für das Plakat wurden gemacht, jedoch stellte sich ein Problem, da wir uns noch nicht für einen Namen für das Spiel entschieden hatten. Dieses Problem wurde dann in einem Zoom-Meeting mit dem gesamten Team angesprochen und schließlich einigten wir uns alle auf den Namen "Nightmare on Bahnhofsstreet". Damit konnten wir das Poster weiter vorantreiben.

Feier

Zu Beginn konnten wir uns noch nicht auf einen Namen festlegen. Letztendlich einigten wir uns auf Simons Vorschlag „Nightmare on Bahnhofsstreet“. Dieser Name sollte auch auf dem Poster stehen.



POSTER

4 GAMEPLAY

4.1 VR-SPIELER GAMEPLAY

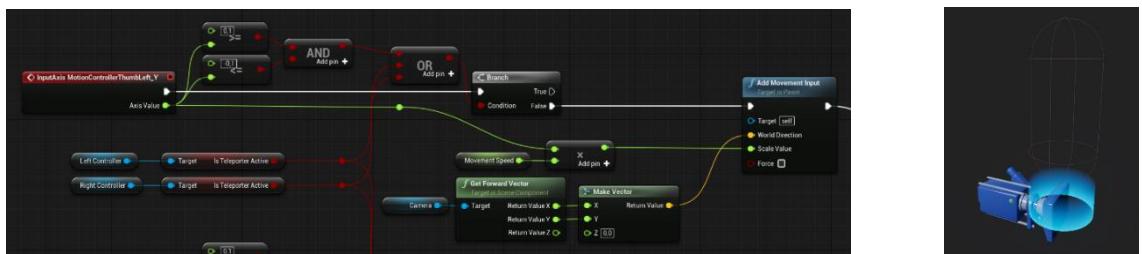
4.1.1 BEWEGUNGSSTEUERUNG

Dan

Da es sich in unserem Fall um ein Rätselspiel mit Horrorsetting in einer Virtual Reality Umgebung handeln soll, stellen sich diese Fragen bereits bei der Art der Bewegung. Aufgrund von Motion Sickness, die viele Spieler in VR erfahren, hat sich die Teleportation als ein gängiges Fortbewegungsmittel in einer solchen Umgebung durchgesetzt. Allerdings geht dadurch vor allem in Horrorspielen viel von dem Gruselfaktor verloren, da man gänsehauterregenden Situationen leicht entfliehen kann oder sogar Schreckmomente komplett überspringt. Um dieses Problem zu umgehen, haben verschiedene Spiele unterschiedliche Lösungsansätze gewählt, wobei eine der effektivsten Möglichkeiten ein System ist, in welchem man nur feste Punkte hat, an die man sich teleportieren kann. Hier legen die Entwickler vorher Stellen im Raum fest, zu denen sich der Spieler teleportieren muss, um im Spiel fortzuschreiten. Gleichzeitig kann der Spieler sich auch nicht anderweitig im Raum bewegen, was dazu führt, dass er gezwungen ist, alle Schreckmomente zu erfahren und sogar aus einem Blickwinkel zu sehen, den man vorher ganz genau festlegen konnte. Auf der anderen Seite führt dies jedoch dazu, dass der Spieler kaum noch Freiheiten hat und das Spielerlebnis sehr linear und erzwungen wird. Des Weiteren gibt eine solche Fortbewegungsmethode auch Hinweise darauf, wie Rätsel gelöst werden sollen und schränkt die Anzahl der möglichen Lösungsvorgehen und Irreführungen drastisch ein.

Mithilfe dieser Überlegungen sind wir dann zu dem Entschluss gekommen, die Motion Sickness zu Teilen in Kauf zu nehmen und dem Spieler die Möglichkeit zu geben, sich mithilfe der Thumsticks im Spiel zu bewegen. Um die Übelkeit so gut wie möglich vorzubeugen verzichten wir dabei auf abrupte Bewegungen, erlauben es dem Spieler nicht, sich zu schnell zu bewegen, und lassen Drehungen nur in 45° Schritten ausführen. Natürlich ist auch diese Fortbewegungsmethode nicht innovativ und wird schon in einigen VR-Spielen, allen voran Horrorspielen wie „Phasmophobia“, verwendet.

Umgesetzt haben wir diese Bewegungssteuerung, indem wir anhand der Orientierung und Position des Spielers mithilfe der Controller-Inputs entschieden, wie schnell sich der Spieler wohin bewegen oder rotieren sollte. Da der VR-Spieler im Spiel jedoch standardmäßig keinen Körper und daher auch keine Kollisionsabfrage besaß, haben wir dem „Motion Controller Pawn“ eine Kapsel hinzugefügt, welche diese Abfragen hervorrufen konnte. Hierdurch kamen allerdings neue Fragen und Probleme auf, die damit zusammenhängen, ob diese Kapsel an den Mittelpunkt des VR-Spielfeldes oder an die Headset-Position des Spielers angeknüpft ist.



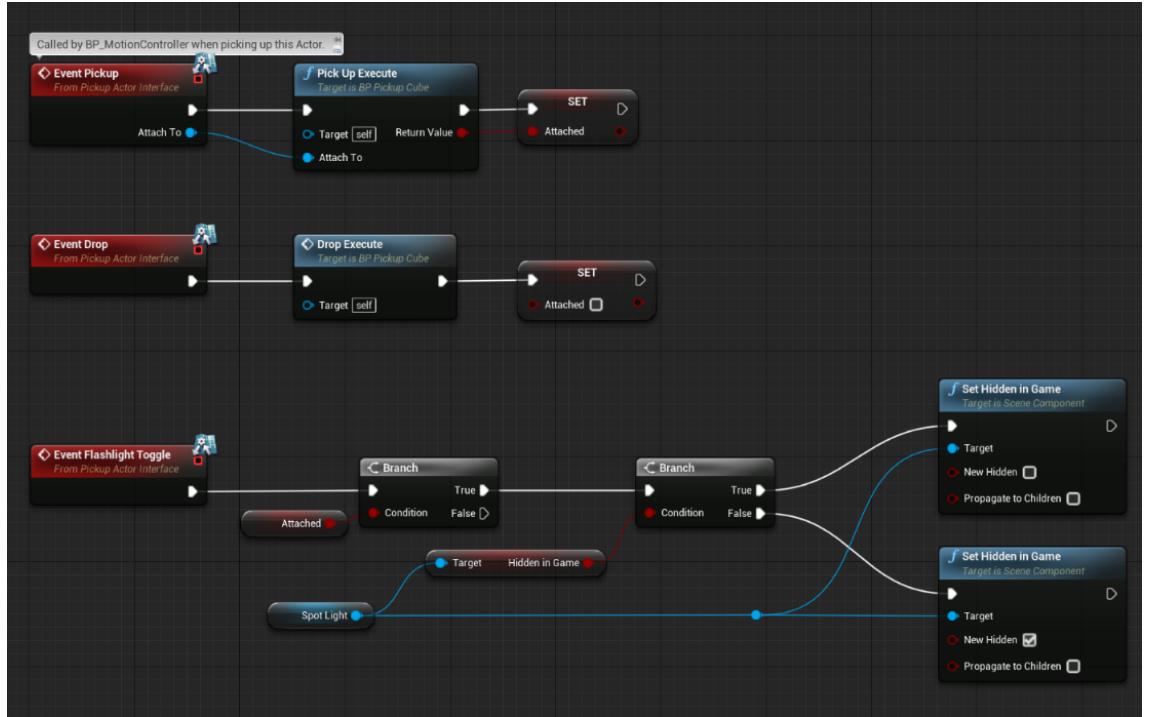
Im letzteren Fall entstanden dadurch nämlich einige Probleme, wenn der Spieler seinen Kopf in eine Wand oder in den Boden bewegt. Dies sorgte dafür, dass eine Kollisionsberechnung den Körper des Spielers von diesen Objekten wegstieß. Dadurch

entstanden unerwünschte, ruckartige Bewegungen, die sowohl unerwartet als auch unberechenbar waren. In einer Virtual Reality Umgebung sollte man so etwas unter allen Umständen vermeiden, da dies schnell zu Unwohlsein und Störungen des Gleichgewichtssinnes führen kann. Deshalb ließen wir die Kapsel und damit den virtuellen Körper des Spielers immer im Zentrum des VR-Spielbereiches. Auch, wenn dies zur Folge hatte, dass sich der Spieler zwar aus seinem Körper herausbewegen konnte, indem er sich im realen Raum umherbewegte, und auf diese Art und Weise auch durch Wände hindurchging, war dies ein Problem, über das wir bereit waren, hinwegzusehen. Selbst umsatzreiche Spiele wie „Half Life Alyx“ gingen gegen dieses Problem nur so insofern vor, dass sie den Bildschirm gelb werden ließen, wenn ein Spieler den Kopf durch Hindernisse hindurchbewegte. Um den Spieler zumindest darauf aufmerksam zu machen, wo er eigentlich stehen sollte, machten wir den dafür vorgesehenen Bereich mit einem Highlight sichtbar.

4.1.2 TASCHENLAMPE

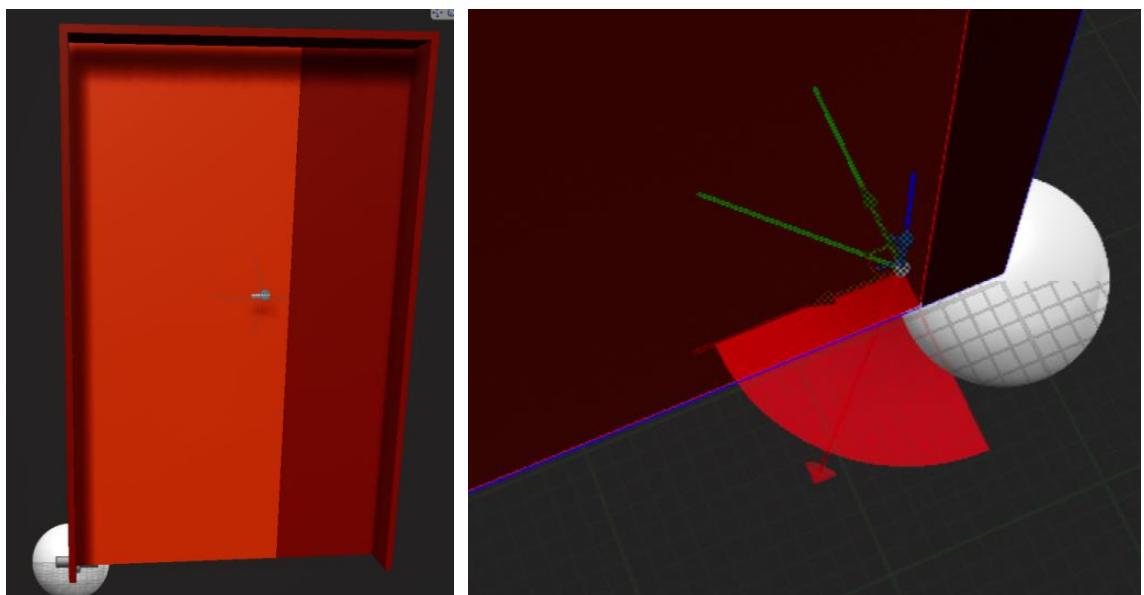
Nachdem mit der Bewegung eine essenzielle Spielmechanik implementiert wurde, beschäftigten wir uns mit der Taschenlampe. Bei ihr war es besonders wichtig, ein simples und dennoch vielseitiges Blueprint zu erstellen, damit sowohl der VR-Spieler als auch der Hacker mit ihr interagieren konnten. Durch die Erfahrungen, die wir bereits vor diesem Projekt in der VR-Programmierung mit der Unreal Engine sammeln konnten, wussten wir, dass sich dafür ein Interface anbot. Über dieses Interface konnten die Controller des VR-Spielers und später dann auch die Tastendrücke des Hackers mit dem Blueprint der Taschenlampe in der Art interagieren, dass sie sich an- und ausschalten ließ. Damit eine Taschenlampe überhaupt gehalten werden konnte, erbte sie von der Klasse „BP_PickupCube“, welche sich um eben diese Funktion kümmerte und gab ihr eine Variable „Attached“, damit auch der Hacker später abfragen konnte, ob diese Taschenlampe überhaupt gehalten wird.

Um zu überprüfen, welche Taschenlampe der Spieler betätigen möchte, wurde über das Blueprint des Controllers, der betätigt wurde, abgefragt, ob er die Taschenlampe auch wirklich hält.



4.1.3 TÜR

Nachdem die Implementierung der Taschenlampe ohne Probleme abgeschlossen werden konnte, befassten wir uns mit der nächsten grundlegenden Interaktionsmöglichkeit: Das Bewegen von Türen. Dazu sahen wir uns im Internet einige Videos² an, die die Interaktion mit Türen in Virtual Reality alle auf unterschiedlichste Arten lösten. Daraufhin machten wir uns Gedanken, wie es in unserem Fall implementiert werden sollte. Das Konzept, welches wir uns überlegten, war, mithilfe von zwei Angelpunkten an der Tür ihren Bewegungsfreiraum so einzuschränken, dass sie sich nur in einem 90 Grad um die Z-Achse drehen lässt. Wir begannen damit, aus der Tür, dem Türrahmen und dem Türgriff ein gemeinsames Blueprint zu erstellen. Da der Ursprung der Tür jedoch außerhalb des Modells lag, benötigten wir noch einen Actor, an den die Tür angehängt wurde und welcher sich dann anstelle der Tür gedreht hat, um eine passende Drehung zu simulieren. Als Nächstes fügten wir die beiden „Pivot Points“ ein, die zwischen diesem Actor und dem Türrahmen eine entsprechende Bewegungseinschränkung erschufen.



Beim Testen dieses Konstruktes, zeigte sich schnell, dass die Drehung sehr unsauber und unzuverlässig war. Außerdem ließ sich die Tür durch die Controller des VR-Spielers auch in die Richtung aufdrücken, in die sie sich eigentlich nicht bewegen sollte, und sprang danach wieder in ihre Ursprungsposition zurück. Ein weiteres Problem ergab sich darin, dass der Spieler sich nicht durch den Türrahmen hindurchbewegen konnte, auch als die Tür geöffnet war.

Um diese Probleme zu lösen, sorgten wir zuerst dafür, dass die Tür mit ihrem Türrahmen keine Kollision erzeugt. Hier kümmerte sich Niklas darum, die Einstellungen der Modelle in Unreal Engine so zu ändern, dass die Kollisionsberechnungen des Türrahmens auch wirklich nur an den Stellen berechnet wurden, an denen auch wirklich die Facetten des Modells entlangliefen.

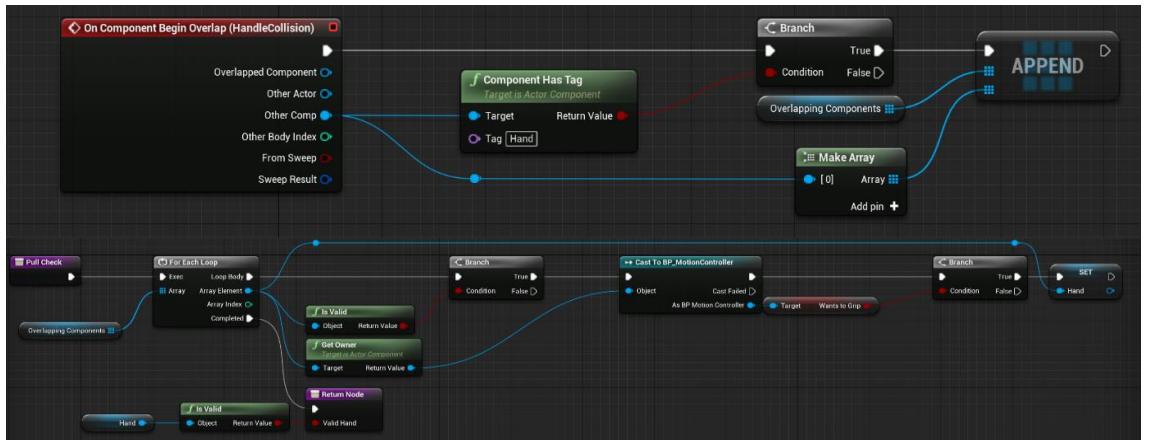
Nach zahlreichem Ausprobieren und Herumbasteln an dem Blueprint, war es dann möglich, die Tür zumindest ordentlich aufzudrücken und offen zu halten, sodass man durch den Türrahmen durchgehen konnte. Zwar ließ sich die Tür auch weiterhin minimal in die Richtung aufdrücken, in die sie sich nicht öffnen lassen sollte, doch war

² YouTube - <https://www.youtube.com/>

dies nur ein geringes Problem. Außerdem wollten wir uns nicht noch relativ am Anfang des Projektes in Probleme verrennen, die hinterher zu Zeitnot führen könnten.

Die nächste Hürde bei der Interaktion mit der Tür bestand dann darin, sie aufziehen zu können, indem man nach dem Türknauf griff. Unser erster Ansatz zur Umsetzung dieser Funktion bestand darin, die Tür in die Hand des Spielers zu legen, wenn er nach dem Türknauf griff, ohne die Position oder Rotation der Tür zu ändern, solange der Spieler die Hand nicht weiter bewegte. Entgegen unserer Hoffnung, hat der Pivot Point leider nicht dafür gesorgt, dass die Tür sich weiterhin nur innerhalb der vorgegebenen Beschränkungen bewegen konnte. Stattdessen wurde die Tür aus der Angel gerissen und an die Hand des Spielers geklebt, sodass dieser nun mit ihr spazieren gehen konnte. Auch nach längerer Recherche in verschiedenen Internetforen konnten wir keine Lösung finden, wie der Spieler die Tür wirklich greifen konnte, ohne sie von ihrem Türrahmen zu lösen. Daher verworfen wir diesen Ansatz und überlegten uns, mit den angeschauten Videos im Hinterkopf, einen neuen Plan.

Um diesen umzusetzen, fragten wir zuerst einmal ab, ob sich ein Controller in der Nähe des Türgriffes befindet, der diesen auch betätigen möchte. Dafür verwendeten wir ein Feature von Unreal Engine, welches es erlaubt, Objekten oder Klassen mehrere „Tags“ zu vergeben. So konnten wir den Controllern des Spielers den Tag „Hand“ geben und erstellten um den Türknauf eine Kugel, die sich abspeicherte, wenn eine „Hand“ in sie hineinbewegt wurde. Da der Spieler zwei Hände besaß, mit denen er nach Gegenständen greifen konnte, haben wir ein Array verwendet und in einem „Pull Check“ überprüft, ob einer dieser Controller nach dem Türgriff greifen wollte.



Um nun ein möglichst realitätsgerechtes Gefühl beim Interagieren mit der Tür zu erzeugen, erstellten wir mehrere Funktionen und Codeabschnitte, um die aktuelle Position des Controllers mit der Position im letzten Tick zu vergleichen. Darauf folgend konnten wir dann die Tür pro Tick leicht in die entsprechende Richtung drehen lassen, in die sich die Hand des Spielers bewegt hat, sofern diese sich nicht über die Bewegungseinschränkungen der Tür hinaus bewegte. Die erste Variante, mit der wir eine solche Logik implementierten, sorgte allerdings leider dazu, dass sich die Tür überhaupt nicht mehr bewegte und auch nach längeren Tüfteleien konnten wir keine Fortschritte erzielen.

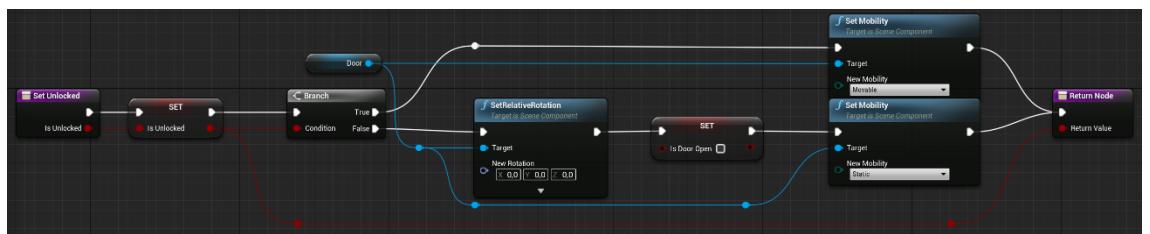
Allerdings wollten wir an der Idee festhalten, um dem Spieler ein Erlebnis bieten zu können, bei dem er möglichst realitätsnah mit der Welt interagieren können sollte. Dies war uns wichtig, um einen noch stärkeren Kontrast zwischen dem Vertrauten des Alltags und dem übernatürlichen Spuk unseres Spieles zu erschaffen.

Im nächsten Ansatz änderten wir den Zusammenhang der Abfragen komplett und ließen die Tür prinzipiell die Tick-Abfragen nur ausführen, wenn auch eine Hand nach

ihr gegriffen hat. Aber auch diese Änderungen ließen die Tür nicht rotieren und die Ideen wurden langsam knapp.

In solchen Situationen hilft es häufig, sich erst einmal von diesem Problem zu distanzieren und kleine Verschönerungen und Refactorings durchzunehmen, mit denen man hinterher dann auch leichter weiterarbeiten kann. Unter anderem war es störend, dass die Tür weiterhin über einen zusätzlichen Actor gedreht werden musste, was irgendwie unschön aussah und auch nicht zu einer intuitiven Programmierung führte. Daher passte Niklas das Modell der Tür an, sodass ihr Ursprung mit ihrem Drehpunkt übereinstimmte. Dadurch konnte auf den zusätzlichen Actor verzichtet werden und alleine schon das Aufdrücken der Tür wurde sauberer und schöner. Gleichzeitig bemerkten wir, dass sich jetzt auch die Tür auf einmal rotieren ließ, wenn man an dem Türgriff gezogen hat.

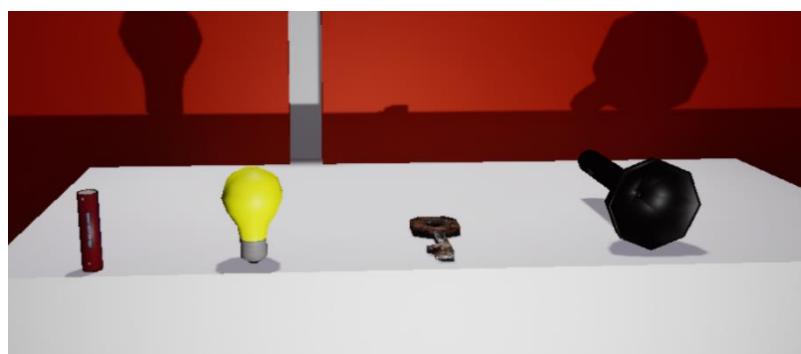
Nach drei Wochen Arbeit an der Tür scheinte es nun endlich zu funktionieren. Das Ziehen an der Tür verlief zwar nicht ganz flüssig und stockte ein wenig, aber das sollten Probleme für die Zukunft sein, wenn es darum ging, dem Projekt einen letzten Feinschliff zu verpassen. Genauso verschoben wir es auch nach hinten, uns darum zu kümmern, dass sich der VR-Spieler aufgrund seines Kollisions-Körpers sehr merkwürdig und unintuitiv bewegen musste, um die Tür beim Öffnen nicht gegen seinen Körper zu schlagen, worauf hin sie sich aufgrund der Kollision nicht weiterbewegen konnte.



Um die Arbeit an der Tür vorerst zu einem Abschluss zu bringen, fügten wir nur noch ein, dass sie zu- und aufgeschlossen werden kann, indem wir eine boolesche Variable dafür setzten und die Tür je nachdem beweglich oder statisch setzen.

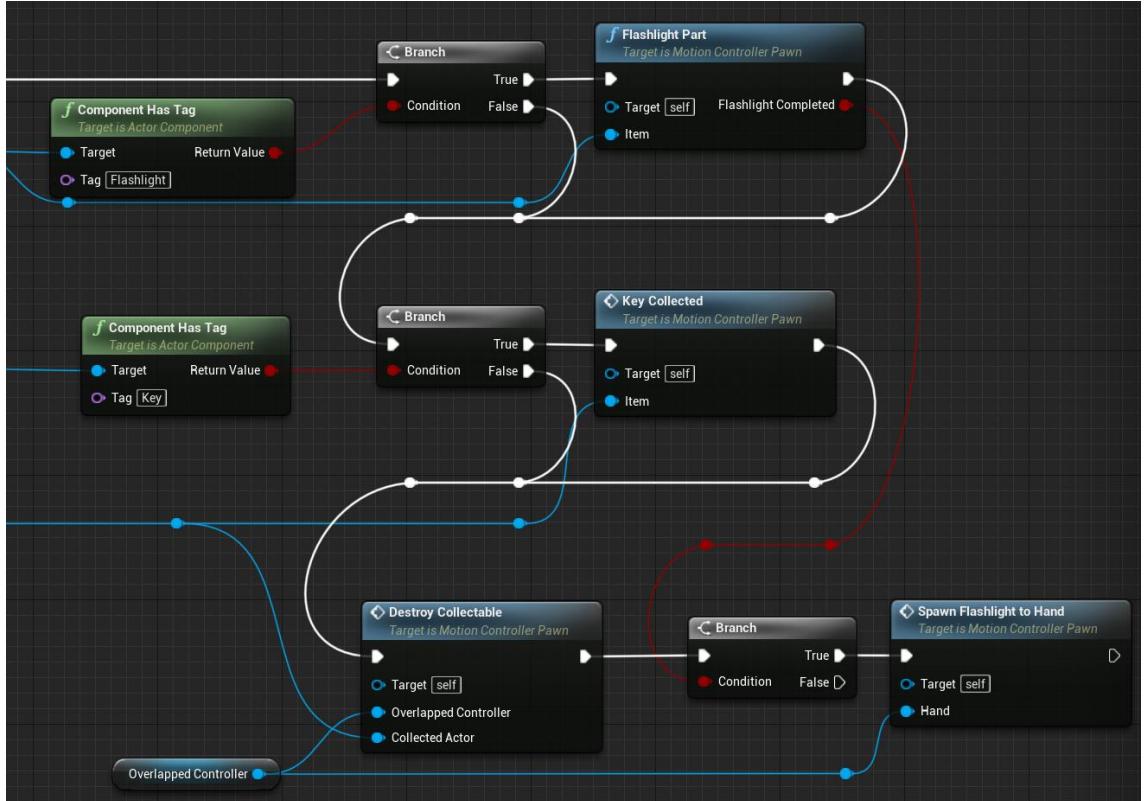
Als sich dann eine Woche später die Tür wieder nicht ziehen ließ und stattdessen bevorzugte wie im Weltall durch die Gegend zu schweben, distanzierten wir uns erst einmal komplett von der Tür und wendeten uns dem Einsammeln von Gegenständen zu.

4.1.4 EINSAMMELN VON GEGENSTÄNDEN



Für diese Funktion bot es sich an, den Kegel für die Kollisionsberechnung des Spielers wiederzuverwenden. Möchte der Spieler also einen Gegenstand einsammeln, so musste er ihn zuerst aufheben und ihn dann mit dem Controller in seinen Körper hineinbewegen. Durch das Bewegen der Hand gegen den Körper bemerkten wir jedoch,

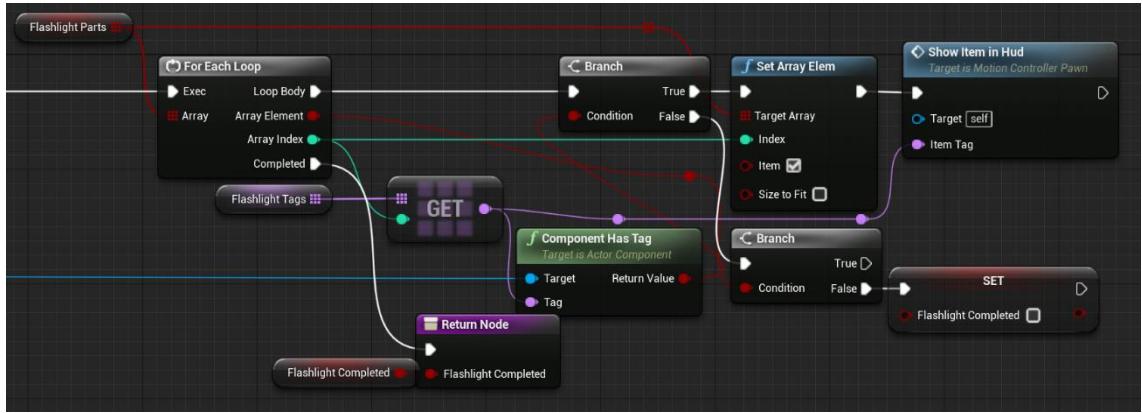
dass die Controller selbst Kollisionen erzeugten und daher den Motion Controller Pawn durch die Gegend schubsten, was die Bewegungssteuerung beeinflusste. Nachdem der Ursprung des Problems gefunden war, ließ es sich zum Glück aber schnell beheben, indem wir bei Controllern und Pick-Up-Cubes einstellten, dass diese nicht mit dem Pawn kollidieren können. Gegenstände, die der Spieler aufsammeln konnte, waren dabei nämlich auch nur Pick-Up-Cubes, welche als Tag sowohl „Collectable“ als auch ihren jeweiligen Gegenstandstypen hatten, wodurch eine Unterscheidung sehr einfach war.



Beim Einsammeln der Gegenstände wurde dann überprüft, ob der Controller, der in den Körper des Spielers bewegt wurde, auch wirklich den Gegenstand hält und ob es ein Schlüssel oder ein Teil der Taschenlampe war, welcher zum Zusammenbau benötigt wurde. Dazu wurden dann je nach Gegenstandstyp unterschiedliche Funktionen ausgeführt, bevor der eingesammelte Gegenstand zerstört und, sollte das letzte fehlende Taschenlampenteil eingesammelt worden sein, eine Taschenlampe erzeugt wurde.

Bei einem Schlüssel wurde außerdem die dazugehörige Tür benachrichtigt, dass eben dieser Schlüssel eingesammelt wurde. Damit konnte diese Tür aufgeschlossen werden, wenn der Spieler das nächste Mal mit ihr interagierte.

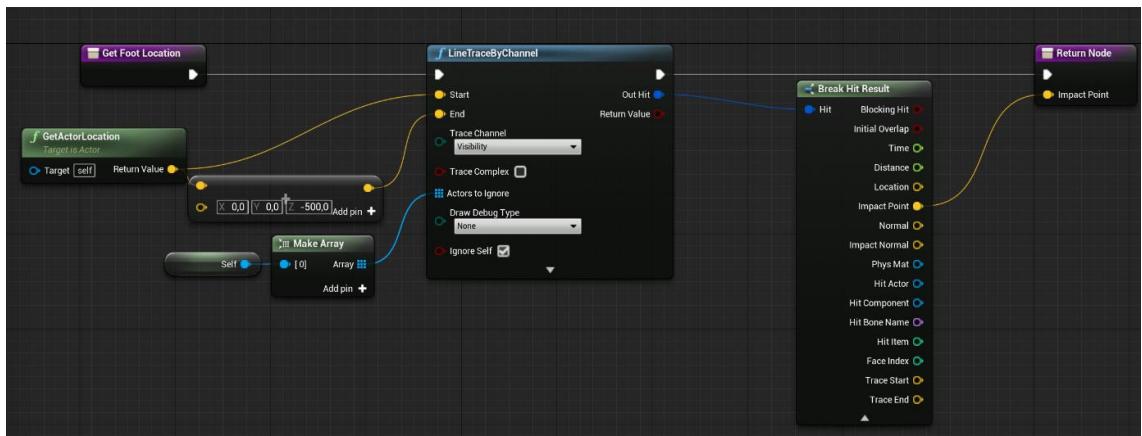
Da bei einem Taschenlampenteil gespeichert werden musste, welche Teile bereits eingesammelt wurden, erstellten wir ein Array mit booleschen Variablen und eines mit verschiedenen Tags, um zu überprüfen, welches Taschenlampenteil eingesammelt wurde und um dieses als eingesammelt abzuspeichern. Wurden alle Teile eingesammelt, so spawnten wir eine Taschenlampe in den Händen des Spielers und setzten das Array mit den booleschen Variablen auf „false“ zurück.



4.1.5 PROBLEMLÖSUNG: SPIELER HEBT AB

Nachdem das Einsammeln der Gegenstände und das Erzeugen der Taschenlampe vergleichsweise problemlos und schnell funktioniert hatte, wollten wir uns einem Problem widmen, welches schon seit der Implementierung der Bewegungssteuerung vorhanden war und zu dem wir uns nebenbei häufig Gedanken machten. Da im Raum So.11 Treppen vorhanden waren, musste es dem Spieler auch möglich sein, diese zu erklimmen und auch wieder hinunterzugehen. Das Hochlaufen auf den Treppen funktionierte zwar, war aber sehr unsauber, da man sich wiederholt nach links und rechts bewegen musste, um auf die nächste Stufe zu gelangen. Beim Heruntergehen blieb der Spieler allerdings auf derselben Höhe, auf der er die vorherige Stufe verlassen hat. Der Grund dafür war, dass die Bewegungssteuerung die Z-Koordinate unverändert ließ und das Hochlaufen der Treppe nur möglich ist, weil der Spieler durch Kollision vom Boden abgestoßen wird, während beim Heruntergehen die Höhe, auf der sich der Spieler bewegt, nicht mehr geändert wird.

Um das Problem lösen zu können, musste daher überprüft werden, auf welcher Höhe der Spieler eigentlich stehen sollte. Dafür verwendeten wir die Funktion „Line Trace By Channel“, welche eine Linie projizierte, die für das erste sichtbare Objekt, welches von ihr getroffen wurde, einen „Hit“ ausgab. Die Linie startete dabei beim Spieler und verlief die Z-Koordinate senkrecht nach unten, sodass der Auftrittspunkt mit dem Boden als „Impact Point“ gespeichert wurde. Diesen Punkt konnten wir dann verwenden, um die Kapsel des Spielers auf die passende Höhe zu setzen.



Des Weiteren verringerten wir die Höhe der Kapsel und setzten sie bündig mit dem Kopf des Spielers, sodass der Spieler etwas Puffer an den Beinen hatte, wodurch er in diesem Bereich keinerlei Kollisionen erzeugte und das Hochgehen der Treppen deutlich angenehmer verlief.

4.1.6 VR-SPIELER HUD

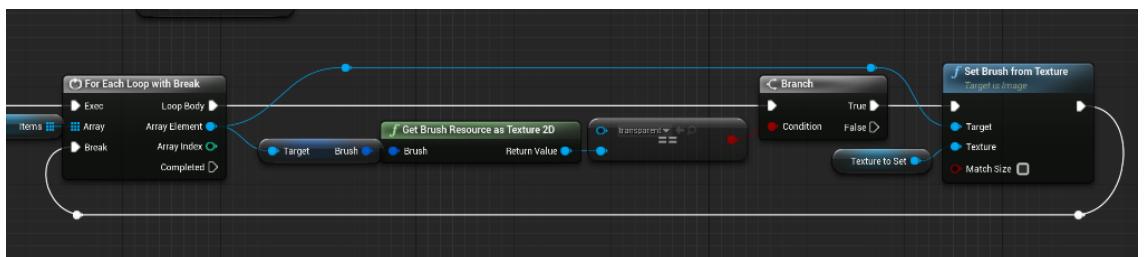
Damit der Spieler vor Augen hatte, welche Gegenstände er einsammelte, beschlossen wir, ein HUD zu erstellen, welches eben diese gesammelten Gegenstände anzeigen sollte. Da wir uns zuvor noch nie mit etwas Vergleichbarem beschäftigt hatten, suchten wir im Internet zuerst nach einigen Ideen. Allerdings wurden wir bei der Suche nicht fündig, da sich viele Anleitungen und Tipps entweder nicht auf Virtual Reality und die Umsetzung mit Blueprints beschäftigten oder es sich um keine an der Kamera des Spielers befestigten Anzeige handelte, sondern lediglich um eine Ebene im Raum.

Daher setzten wir uns ohne Vorwissen an das von Unreal Engine für HUDs vorgesehene „User Widget“ und experimentierten damit herum. Weil es in unserem Spiel eine begrenzte Anzahl an Gegenständen gab, die der Spieler aufsammeln und gleichzeitig im Inventar haben konnte, beschlossen wir, die Anzeige auf vier Inventarplätze zu beschränken. Um ein möglichst anschauliches und verständliches Inventar zu erzeugen, bastelten wir mit den verschiedenen Objekten, die es in einem User Widget geben kann, ein simples Feld, mit transparenten Hintergründen und umrahmten Feldern für die eingesammelten Gegenstände. Da sich der schwarze Rahmen dabei an der Größe der Bilder orientierte, die wir verwendeten, brauchten wir auch ein transparentes Bild in entsprechender Größe, damit das Inventar auch schön aussah, wenn man nicht alle Inventarplätze mit Gegenständen gefüllt hat.



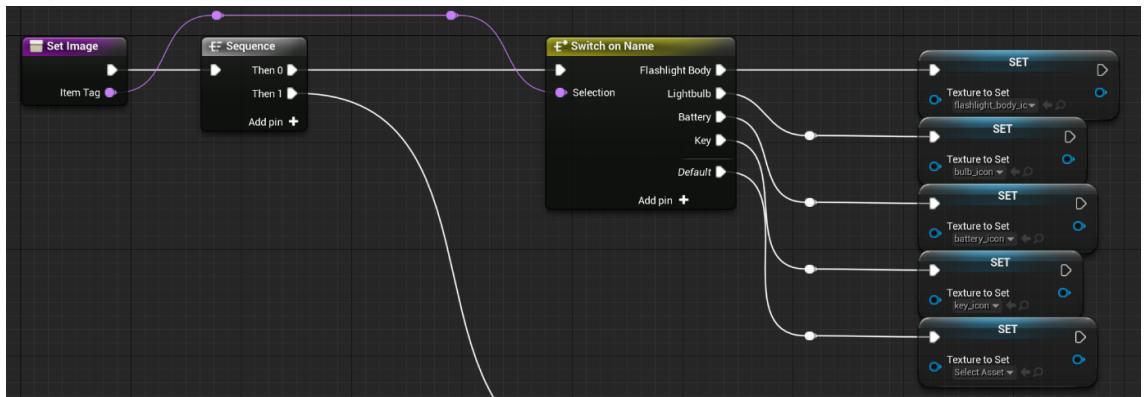
Nachdem der Aufbau des Head-up-Displays soweit feststand, ging es darum, dieses so anzupassen, dass es für den VR-Spieler leicht und angenehm sichtbar ist, ohne zu viel von der Spielumgebung zu überdecken. Dafür musste das User Widget an die Kamera des Motion Controller Pawns angehangen werden und durch zahlreiche Tests auf die entsprechende Entfernung verschoben und in passende Verhältnisse gebracht werden.

Als nächstes stellte sich dann die Frage, in welcher Reihenfolge und mit welcher Logik die Gegenstände im Inventar angezeigt werden sollten. Aufgrund dessen, dass der Spieler Gegenstände aus dem Inventar nicht direkt auswählen konnte und es lediglich als Übersicht diente, war es für das Gameplay kein Problem, wenn wir die Gegenstände im Inventar nicht immer auf ihrem ursprünglichen Platz ließen. Es lag uns daher nahe, das Inventar immer von links aus gesehen aufzufüllen und zwischen den Gegenständen keine Lücken zu erlauben. Um dies zu bewerkstelligen, legten wir ein Array für die gesetzten Bilder im HUD an und ersetzten das erste rein transparente Bild im Array mit dem Bild für den Gegenstand, der eingesammelt wurde.

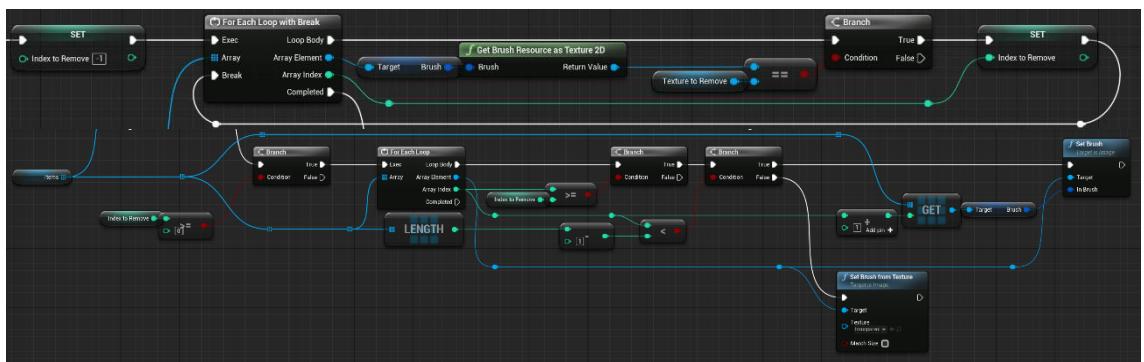


Hierbei kam jedoch eine weitere Frage auf, da wir uns nun abspeichern mussten, welches Bild denn überhaupt für einen bestimmten Gegenstand verwendet werden sollte. Hier hätte man nun bei den Gegenständen selbst eine Variable hinzufügen können, welche eben dieses Bild abspeichert und den Verweis auf das Bild dann an das HUD übergeben kann. Dies würde allerdings bedeuten, dass wir auch wirklich immer ein Objekt erzeugen müssen, damit dieses angezeigt werden kann, auch wenn wir es direkt nach der Erzeugung einsammeln und dadurch wieder sofort löschen. Dieser Fall würde zum Beispiel bei dem Interagieren mit der richtigen Puppe eintreten, da wir hier gar keinen Schlüssel erzeugen müssten, sondern diesen am liebsten einfach dem Inventar des Spielers hinzufügen würden.

Die Alternative dazu war es, dem HUD selbst die Aufgabe zur Verwaltung der Bilder zu geben. Hierfür reichte es, dem HUD den Namen des Gegenstandes zu übergeben, der angezeigt oder entfernt werden sollte und im Blueprint zu überprüfen, welches Bild dazugehörte. Da sich diese Möglichkeit leicht umsetzen ließ und auch angenehm erweiterbar war, sollten neue Gegenstände dazukommen, die der Spieler einsammeln kann, entschieden wir uns für diese Version.



Das Entfernen von Gegenständen aus der Anzeige erwies sich dann als komplexer, gelang aber ebenfalls geschwind und problemlos. Zuerst überprüften wir, an welcher Position im Inventar sich das erste Bild zu diesem Gegenstand befand und speicherten uns diese ab. Daraufhin gingen wir alle Bilder der Anzeige, angefangen mit dem herausgefundenen Index, durch und ersetzen sie mit dem nachfolgenden Bild. Beim letzten Bild brauchten wir dadurch nur noch das transparente Bild einfügen und konnten dadurch bewerkstelligen, dass weiterhin keine Lücken zwischen den Bildern in der Anzeige entstanden.



4.1.7 SCHLÜSSELVERGABE IN S.011

Marc

Für die Schlüsselvergabe stellten wir uns folgendes Konzept vor: Die Szene beinhaltet drei Bilder an der Wand. Eines der Bilder zeigt ein teilweise ausgefülltes Tic-Tac-Toe

Feld, das zweite ein Haus in drei verschiedenen Farb-Kombinationen, und das letzte eine Dame mit einer Sprechblase. Wichtig für das Rätsel waren das Haus und das Tic-Tac-Toe Feld, folgender Gründe wegen: Die Wandfarben- und Dachfarbenkombination des Hauses sollten darüber entscheiden, ob der Schlüssel in einer blond/grünen, rot/orangen oder schwarz/roten Puppe versteckt sein würde. Die Spalte, in der sich der Kreis befand, sollte darüber entscheiden, ob der Schlüssel sich in einer Puppe in der linken, mittigen oder rechten Sitzreihen-Spalte befand. Daraus ergaben sich neun verschiedene mögliche Puppen, in welcher sich der Schlüssel befinden konnte. Das Zitat-Bild sollte sich nicht verändern.

Der Schlüssel sollte zufällig einer dieser neun Puppen zugewiesen werden. Die Idee war nun, nicht jedes Mal die Puppen neu zu verteilen und einer der Puppen zufällig den Schlüssel zu geben, sondern die neun möglichen Puppen fest im Raum zu verteilen und ihnen dann zufällig den Schlüssel zuzuweisen. Dies sollte abhängig von den beiden veränderbaren Bildern geschehen, die per Zufallsalgorithmus eine der drei möglichen Materialien zugewiesen bekamen. Dies realisierten wir mit Hilfe des Tutorials „UE4 - Tutorial - Change Textures at Run Time“³. Hierbei wurde durch die Funktion „Random Integer“ eine Zufallszahl von 0 bis 2 ausgewählt und anhand dieser dann entschieden, welches Material genommen wurde.

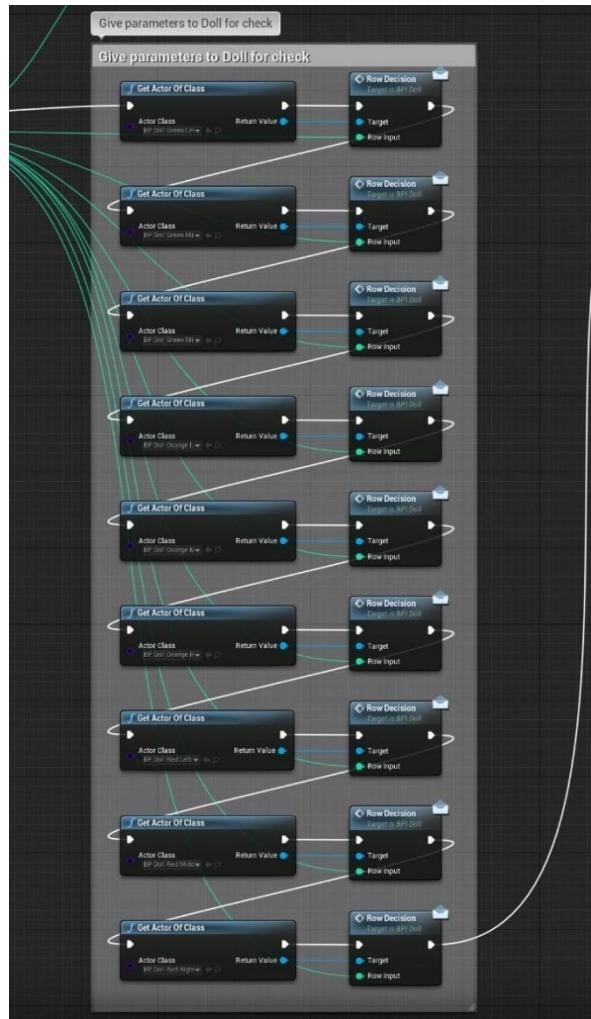
Schwieriger gestaltete es sich dann, diese beiden Zufallsauswahlen mit der Auswahl der Puppe zu verknüpfen. Die Idee war, die Zufallszahl an die Puppen weiter zu geben, und jeder der neun Puppen eine Zahlenkombination mitzugeben. Die bestand zum einen aus einer Zahl für die Sitzspalte (analog zum Tic-Tac-Toe Kreis, also beispielsweise 0 = links) und zum anderen aus einer Zahl für die Haar/Oberteil-Kombination. Zu Beginn dachten wir, man könnte das Vorhaben über Blueprint Childs realisieren, indem man also eine Parent-Puppe machte, die die beiden Variablen besaß, und aus dieser neun Child-Puppen erzeugte, die dann manuell eigene Werte je nach Lage und Farbe bekamen. Hierbei stellte sich allerdings die Herausforderung der Kommunikation zwischen den Bildern und den Puppen. So erstellten wir 9 nicht erbende Blueprints, welche alle mit beiden Bildern über Interfaces kommunizierten. Hilfestellung holten wir uns mit dem Tutorial „UE4 How to easily communicate between blueprints - Blueprint Interface Tutorial“⁴ und von Dan. In Worte gefasst funktioniert die Vergabe wie folgt:

Beide Bilder erzeugen eine Zufallszahl, mit welcher sie ihre jeweiligen Materialien entscheiden. Eben diese Zufallszahlen werden einmalig zu Beginn der Szene durch alle neun Puppen gegeben und mit deren zugewiesenen Zahlenwerten verglichen. Ist eine Übereinstimmung beider Zahlen in einer Puppe gefunden, so wird innerhalb dieser eine bool'sche Variable auf true gesetzt, um anzuseigen, dass diese Puppe den Schlüssel besitzt.

Als es ans Testen ging, fiel Simon auf, dass die Bilder nicht zu der Puppe passten, die den Schlüssel enthielten. Nach kurzer Überlegung stellten wir fest, dass wenn man zwei Branches aus einer Random Integer-Node herauszieht natürlich auch zwei verschiedene zufällige Integer-Werte erzeugt werden. Dies führte dazu, dass die Bilder-Textur-Vergabe einen Random Integer-Wert bekam, welcher unterschiedlich zu demjenigen Wert war, welcher den Puppen gegeben wurde. Dieses Problem wurde durch einfaches Zwischenspeichern in einer lokalen Variablen gelöst.

³ YouTube - <https://www.youtube.com/>

⁴ YouTube - <https://www.youtube.com/>



Logik der Schlüsselzuweisung, hier Festlegung der Reihe

Hinzu kam, dass wir die Zuordnung der Nummern der Reihen vertauscht hatten, so dass mit „links“ für die Puppen die Blickrichtung zur Tafel gemeint war, obwohl die Bilder die linke Seite mit Blickrichtung zu den Bildern meinten. Das wurde auch behoben.

4.1.8 INTERAKTION MIT SCHALTER AM SICHERUNGSKASTEN

Diese Implementierung ist auf dem Tutorial “Unreal Engine Game VR Push Button to trigger an Event - Loading different Level”⁵ aufgebaut. Die Idee war, mithilfe des VR-Motion Controllers mit einer Hitbox, welche den betreffenden Schalter umhüllte, zu interagieren, und so den Schalter umzulegen. Dies wurde innerhalb der Unreal Engine mithilfe der OnComponentBeginOverlap Funktion aktiviert, welche dann zum Motion Controller-Blueprint gecastet wurde, um zu erkennen, dass nur der Motion Controller die Hitbox aktivieren kann. Dieser Cast aktivierte wiederum eine MoveComponentTo-Funktion, welche mithilfe einiger Vektor-Nodes den Schalter nach unten drehte.

⁵ YouTube - <https://www.youtube.com/>

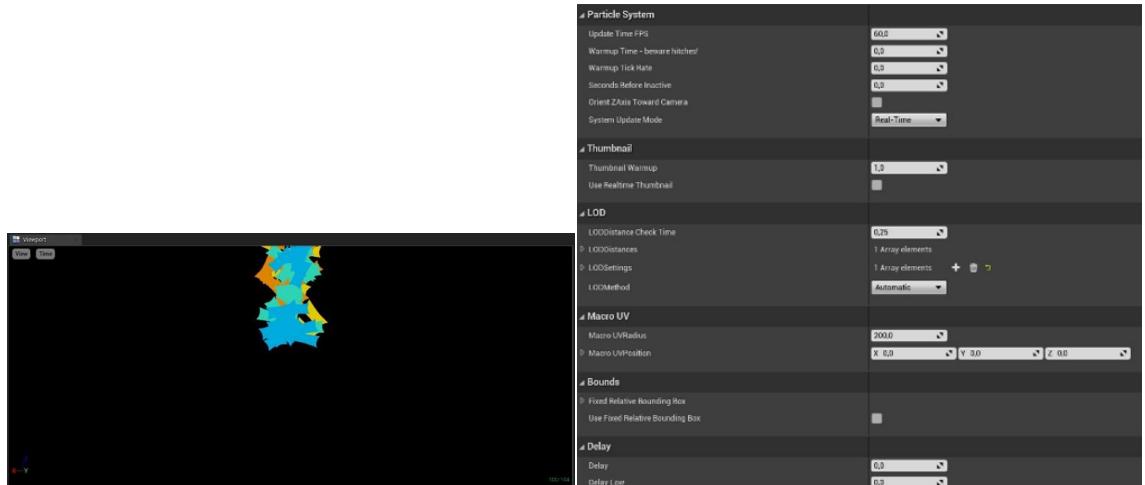
4.1.9 INTERAKTION MIT DEN PUPPEN

Zur Interaktion mit den Puppen in S.011 orientierten wir uns größtenteils an dem MotionControllerPawn-Blueprint und dessen Funktionen. Zu Beginn mussten wir den Fehler, der beim Erstellen der Puppen gemacht wurde, ausbügeln. Die neun Puppen waren nämlich alle eigenständige Blueprints und jeder einzelnen die Funktionen separat zu geben hätte viel Zeit gekostet. Dan gab den Tipp, einen leeren zusätzlichen Blueprint zu erstellen und diesen manuell bei jeder der Puppe als Parent festzulegen. Diesem zusätzlichen Blueprint gaben wir noch eine CollisionBox, mit der der Spieler interagieren konnte.

Zusammen mit Dan bauten wir im Parent-Blueprint die Logik zur Interaktion mit den Puppen auf. Wenn der MotionController mit der Kollision interagierte, dann wurde erst gecheckt, ob es sich beim kollidierenden Objekt auch um eine Hand handelte, und anschließend, ob diese Puppe einen Schlüssel enthielt. Wenn die zweite Bedingung erfüllt war, dann wurde die Hand zu einem Array von „Overlapping Components“ hinzugefügt. Wenn nicht, dann spawnte ein ParticleSystem, welches eine Konfetti Kanonen simulierte. Diese wurde nach dem Vorbild des Tutorials „UE4 - Tutorial - Confetti!“⁶ gebaut und implementiert.

Innerhalb jedes Ticks schaute der Parent-Blueprint, ob der Spieler gerade versuchte, den Schlüssel aufzuheben. Dies passierte durch Überprüfung des „Overlapping Components“-Arrays. War diese Bedingung erfüllt, so ging es weiter in die „Key Collected“-Funktion. Identisch zu Dans Funktion im MotionControllerPawn wurde eine Verbindung zu sowohl der S.011-Tür und eine Schlüssel-Referenz zur S.012-Tür hergestellt. Diese Referenzen mussten leider umständlich für jede Puppe manuell im Editor festgelegt werden.

Zum Schluss wurde der Spawn der Konfetti-Kanone hinter die Überprüfung geschoben, ob die Hand greift. Dies machte mehr Sinn, da die Kanone erst losgehen sollte, wenn der Spieler nach der Puppe greifen wollte.



Particle System der Konfetti Kanone

⁶ YouTube - <https://www.youtube.com/>

4.1.10 ANIMATION DER TÜR

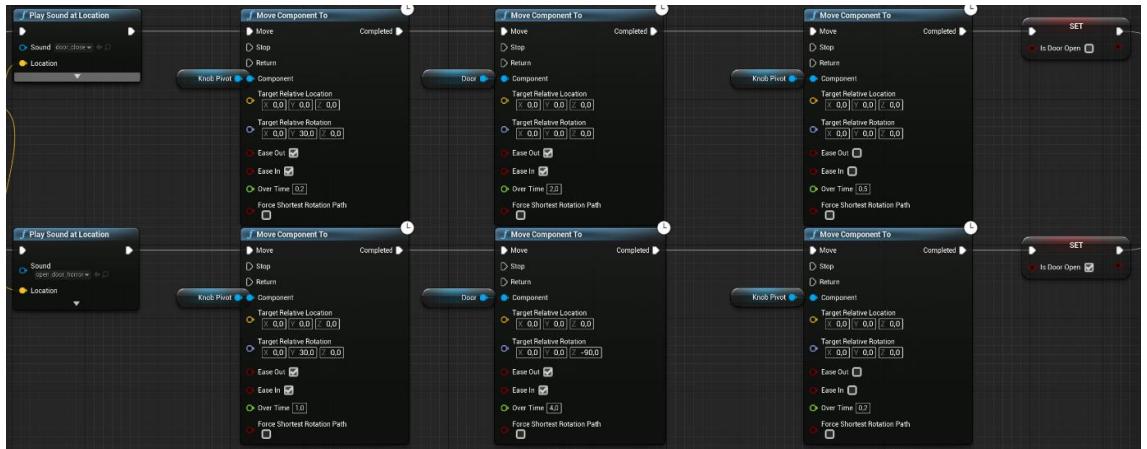
Dan

Da wir uns mehr und mehr dem Ende des Projektes näherten und nun schon die notwendigsten Funktionen implementiert waren, wurde es Zeit, uns unserem alten Erzfeind zu stellen und erneut an der Tür zu arbeiten. Weil eine realistische Interaktion mit der Tür weiterhin zu etlichen Fehlern führte und es schier unmöglich wirkte, dem Spieler ein intuitives und einfach ausführbares Öffnen der Tür zu ermöglichen, beschlossen wir, das Rotieren der Tür zu animieren.

Doch auch diese Animation sollte uns vor Probleme stellen, da wir hierzu ebenfalls nichts im Internet finden konnten. Zuerst versuchten wir es damit, mit jedem Tick die Tür ein kleines bisschen zu bewegen und dann in ihrer endgültigen Position einzurasten zu lassen. Dies war jedoch sowohl rechenaufwendig als auch schwierig für den Spieler anschaulich umzusetzen, da die Tür sich entweder zu schnell öffnete, wodurch man einen fließenden Übergang gar nicht sehen konnte, oder die Rotation überhaupt nicht angezeigt werden konnte. Der letztere Fall führt dazu, dass die Tür nach einer gewissen Zeit einfach nur in ihre Endposition sprang.

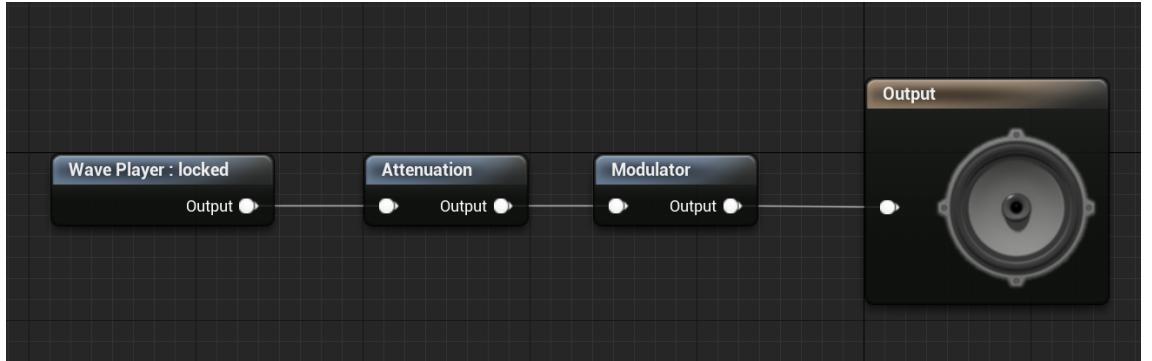
Nach nahezu einer Woche des Ausprobierens, bescherte uns Marc den Durchbruch, indem er uns auf die Funktion „Move Component To“ aufmerksam machte. Mithilfe dieser Funktion konnten wir eine fließende Bewegung des Türknaufs und der Tür simulieren. Innerhalb des Ticks brauchten wir nun lediglich abfragen, ob ein Spieler mit seiner Hand nach dem Türrahmen griff und die Tür entsprechend ihres Zustandes öffnen oder schließen lassen. Die Funktion ermöglichte des Weiteren die Angabe der Zeit, die für die Animation benötigt werden sollte, wodurch wir das Herunterdrücken der Türklinke, das Öffnen oder Schließen der Tür und das Loslassen des Türknaufs zeitlich an die Tonspuren für die Interaktion mit der Tür anpassen konnten.

Während der Animation der Tür wurde außerdem der Körper des Spielers ignoriert, was das Spielerlebnis für den VR Spieler angenehmer machte und außerdem entstanden keine unangenehmen Stoßbewegungen durch Kollisionsberechnungen während der Interaktion. Im Nachhinein lässt sich also sagen, dass es wohl von Anfang an am besten gewesen wäre, hätten wir die Tür durch eine Animation öffnen und schließen lassen. Auch, wenn die Lösung nun sehr kurz und simpel ist und wir die zahlreichen Experimente wieder löschen mussten, so haben wir durch die Probleme, denen wir begegnet sind, zahlreiche Erfahrungen gemacht und einiges dazugelernt.



4.1.11 IMPLEMENTIERUNG MUSIK UND SOUNDS

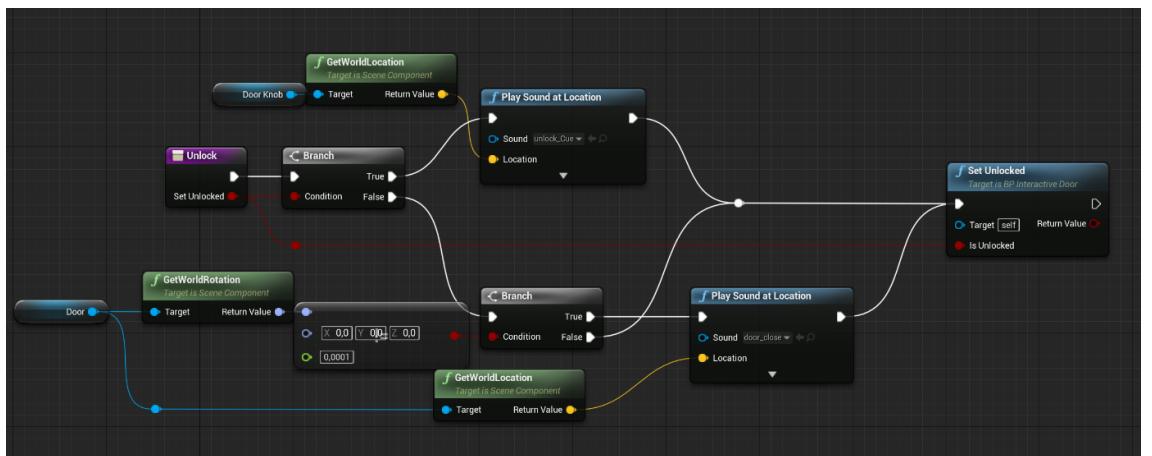
Da wir bei der Tür bereits damit begonnen haben, Sounds zu implementieren, sollte dies auch der nächste Schritt bei der Entwicklung des Spiels sein. Um die verschiedenen Töne gleichermaßen und einfach im Nachhinein bearbeiten zu können, erstellten wir als Erstes eine Attenuation, die von jedem Sound verwendet wurde. Des Weiteren ließen wir die meisten Töne durch einen Modulator laufen, welcher den Pitch und die Lautstärke bei jedem Erklingen des Tones zufällig verändert, um so Abwechslung zu kreieren und das Spiel interessanter zu halten.



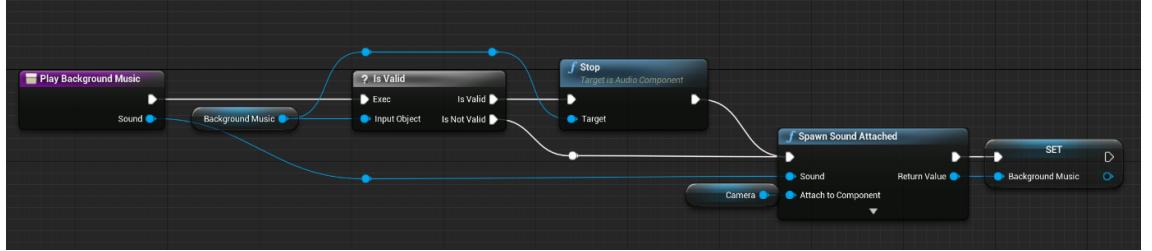
Die „Sound-Cue“, die dadurch entstanden sind, bauten wir dann ins Spiel ein. Der Aufbau der meisten Blueprints bot sich hierfür sehr gut an, sodass man bereits mit einer einzigen Funktion die Musik einfügen konnte. Hierzu sieht man Beispiele beim Aufsammeln von Gegenständen und beim Anschalten der Taschenlampe:



Um auch das Auf- und Abschließen der Tür mit den entsprechenden Sounds einfach an unterschiedlichen Stellen verwenden zu können, wurde dafür im Blueprint der Türe eine extra Funktion angelegt. Zuvor mussten die Sounds dafür in den unterschiedlichen Blueprints, die für das Auf- und Abschließen einer Türe sorgten, immer wieder aufgerufen werden, bevor die Funktion „Set Unlocked“ verwendet wurde. Da dies nicht besonders effizient war und schnell vergessen wurde, haben wir es in einer Funktion zusammengefasst.

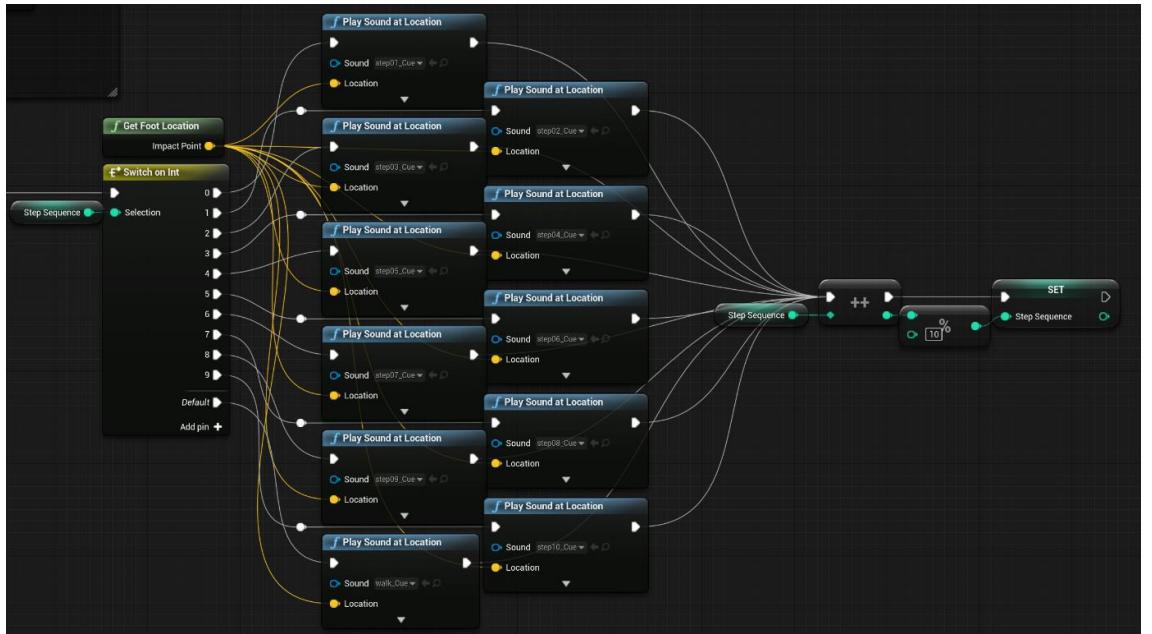


Zum Abspielen von Hintergrundmusik musste ebenfalls ein wenig mehr hinzugefügt werden. Erst einmal erstellten wir im Level einige Trigger Volumes, wodurch wir die Musik des Spielers ändern konnten, je nachdem, in welchem Raum er sich befindet oder in welcher Phase des Spiels er gerade ist. Betritt der Spieler dann einen solchen Auslöser, so gibt er dem Motion Controller Pawn weiter, welche Musik gespielt werden soll. Hierfür schrieben wir eine eigene Funktion, die die bisherige Hintergrundmusik stoppt und die neue Musik an das Headset des Spielers anhängt, sodass sie sich mit dem Spieler mitbewegt.

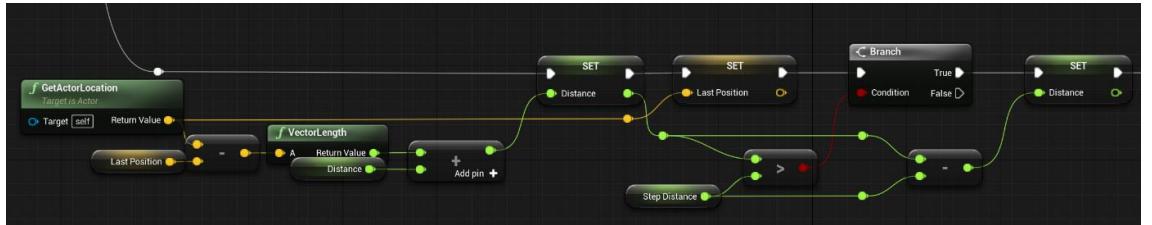


Allerdings verlief die Implementierung der Spielsounds nicht ganz ohne Probleme. Besonders die Fußstapfen, die man beim Bewegen des Spielers hören sollte, benötigten einige Zeit. Das erste Problem bestand darin, dass das Audiodokument, welches für die Fußstapfen verwendet werden sollte, mehrere Sekunden lang war und damit auch mehrere Fußstapfen beinhaltete. Dadurch konnte der Spieler seine eigenen Schritte auch dann noch hören, wenn er schon lange zum Stillstand gekommen war. An dieser Stelle hätte man einbauen können, dass der Sound aufhört, wenn der Spieler sich nicht mehr bewegt, allerdings hätten dann seine Schritte immer sehr abgehackt und unnatürlich geklungen.

Deshalb haben wir das Audiodokument genommen und in zehn einzelne Unterdokumente aufgeteilt, die jeweils ein einzelnes Schrittgeräusch beinhalten. Dadurch konnten wir nun diese Audiodateien nacheinander abspielen und nach einem beliebigen Schritt anhalten. Das Geräusch wurde dafür am Boden unter dem Spielercharakter abgespielt, wobei wir wieder unsere Funktion zur Berechnung eben dieser Position, die wir bereits bei der Bewegung des Spielers benötigten, verwenden konnten.



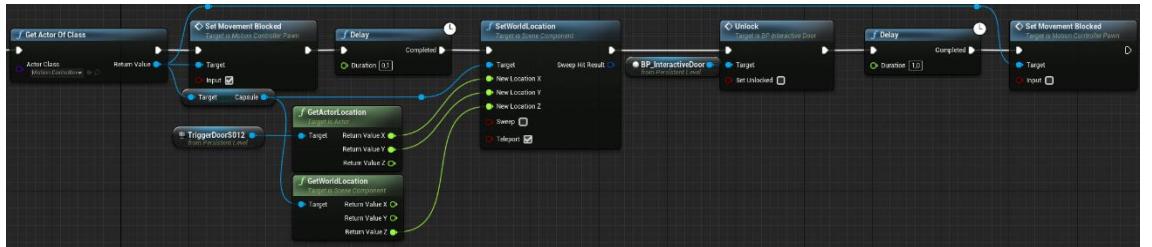
An dieser Stelle hat sich nun aber ein zweites Problem bei den Bewegungsgeräuschen ergeben, denn diese wurden innerhalb kürzester Zeit mehrfach abgespielt, wodurch sich teilweise etliche Schrittgeräusche überlappt haben. Um dieses Problem zu beheben, haben wir die Erzeugung dieser Geräusche in den Tick des Motion Controller Pawns gepackt und mithilfe einer Abfrage geschaut, wie weit sich der Spieler seit dem letzten Tick bewegt hat. Nur, wenn der Spieler eine bestimmte Entfernung zurückgelegt hat, wurde ein Schrittgeräusch erzeugt. Durch Anpassen des Parameters „Step Distance“ konnten wir diese Entfernung so festlegen, dass das Intervall zwischen den Schritten natürlich klang.



4.1.12 TRIGGERN DER EREIGNISSE

Nachdem nun alle Interaktionsmöglichkeiten des Spielers eingebaut waren und auch mit Musik und Geräuschen hinterlegt wurden, setzen wir uns unter Anleitung des Game Design Documents daran, die Interaktionsmöglichkeiten in den Rahmen des Spielablaufes zu setzen. Auf der einen Seite gehörte hier das Verschieben und Umplatzieren von Gegenständen in der Szene dazu, auf der anderen Seite musste auch noch in den Blueprints Arbeit getan werden. Mit einem Trigger Volume sorgten wir zum Beispiel dafür, dass sich der erste Rätsel-Raum des Spiels automatisch aufschließt, wenn der Spieler die Tür am Ende des Ganges untersucht.

Weitere Trigger Volumes setzten wir in die Räume So.11 und So.12 hinter den Eingang, damit wir die Türen hinter dem Spieler schließen konnten, wenn er die Räume betreten hatte. In der Spieleprogrammierung ist an solchen Stellen immer besondere Vorsicht gefordert, da durch das Schließen von Türen ganz schnell Fehler entstehen können, die dazu führen, dass das Spiel unspielbar wird. Was ist zum Beispiel, wenn der Spieler den Raum betritt, aber sich der Großteil seines In-Game-Körpers noch außerhalb des Raumes befindet oder er einfach wieder direkt aus dem Raum rausläuft? Dann würde sich die Tür vor ihm schließen und er könnte niemals den Raum betreten, in welchem er eigentlich ein Rätsel lösen müsste, um im Spiel weiterzukommen. Deshalb haben wir an dieser Stelle eingeführt, dass die Bewegung des Spielers blockiert wird, er sicherheitshalber zusätzlich in den Eingang des Raumes teleportiert und die Tür erst danach geschlossen wird. Mit diesem Vorgehen konnten wir bewerkstelligen, dass sich der Spieler auch immer in dem Raum befand, in dem wir ihn erwarteten und in dem er sich befinden musste, um im Spiel fortfahren zu können.



4.2 PC-SPIELER/HACKER GAMEPLAY

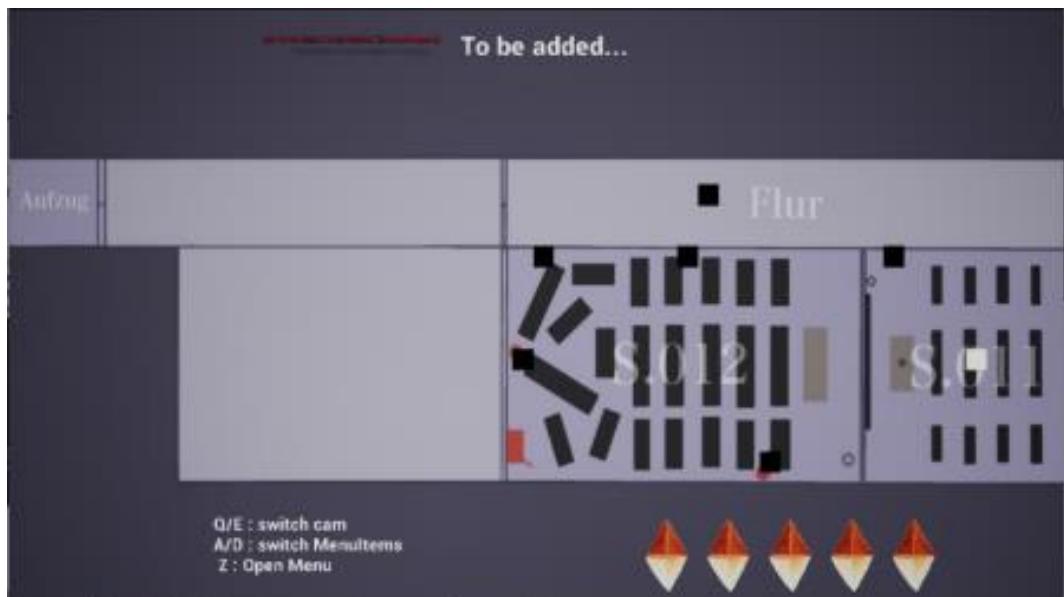
4.2.1 KAMERAIMPLEMENTIERUNG

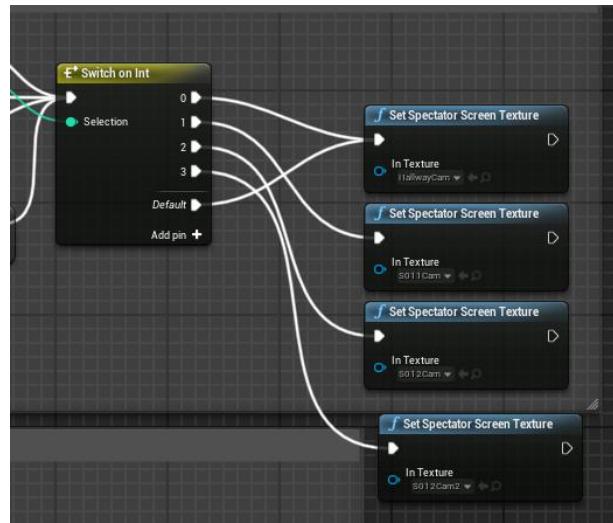
Simon

Anfangs implementierten wir die verschiedenen Hacker-Cams, durch die der PC-Spieler durchschalten konnte, um die Bewegung des VR-Spielers nachvollziehen zu können. Diese implementierten wir durch eine SpectatorView, die Unreal zur Nutzung mit VR bereitstellt. Wir zweckentfremdeten diese Funktion, die eigentlich dafür da ist eine alternative Sicht des VR-Spielers anzuzeigen, indem wir eine erzeugte Textur in die Methode „Set Spectator Screen Texture“ einspeisten, nachdem wir vorher den Modus des SpectatorScreens auf Texture setzen.

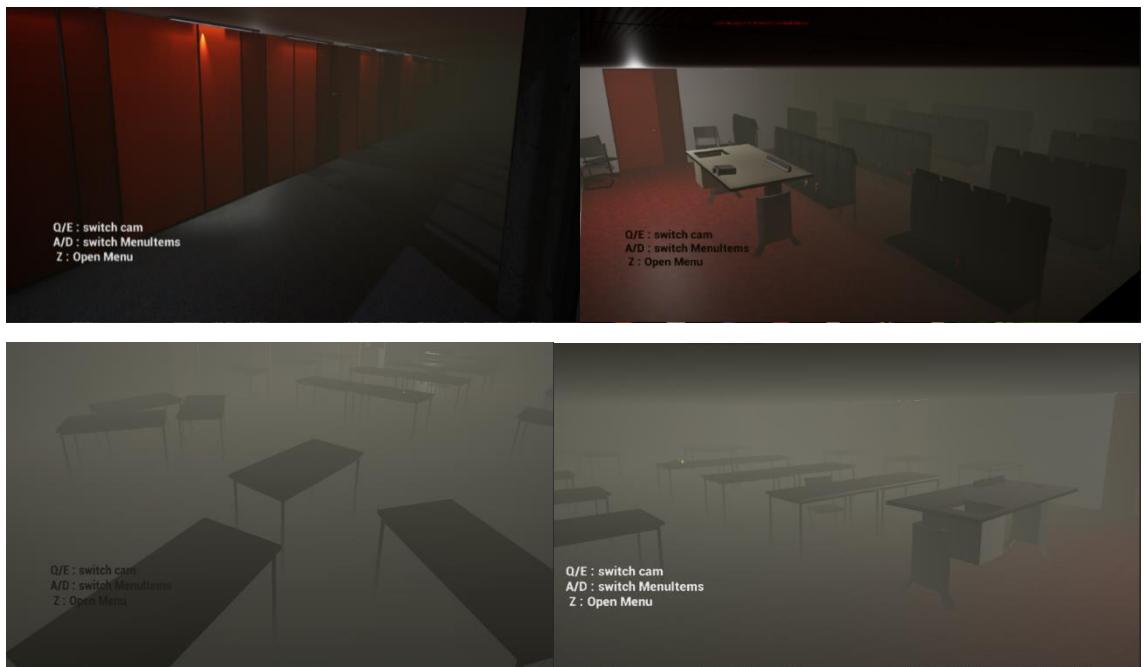


Diese Textur erstellten wir durch ein sogenanntes „Rendertarget“, das ein Bild einer Kamera in eine 2D-Textur umwandelt. Schlussendlich entschieden wir uns dazu, vier Kameras zu implementieren, um dem PC-Spieler genug Möglichkeiten zu geben, den VR-Spieler zu beobachten, eine in jedem zugänglichen Raum und eine, die wir nutzten, um das Hacker-Menü anzuzeigen. Die Kamera im Flur platzierten wir unter der abgeschlossenen Ausgangstür, da sich große Teile des VR-Gameplays damit beschäftigten mit dieser Tür zu interagieren. Bei der nächsten Kamera, die sich in Raum S.011 befand, experimentierten wir ein wenig mit verschiedenen Positionen herum und legten uns schließlich auf die Ecke neben dem Pult fest, da von dort aus sowohl der Eingang des Raumes als auch die Bilder, die zum Lösen des Rätsels nötig waren, sichtbar waren.



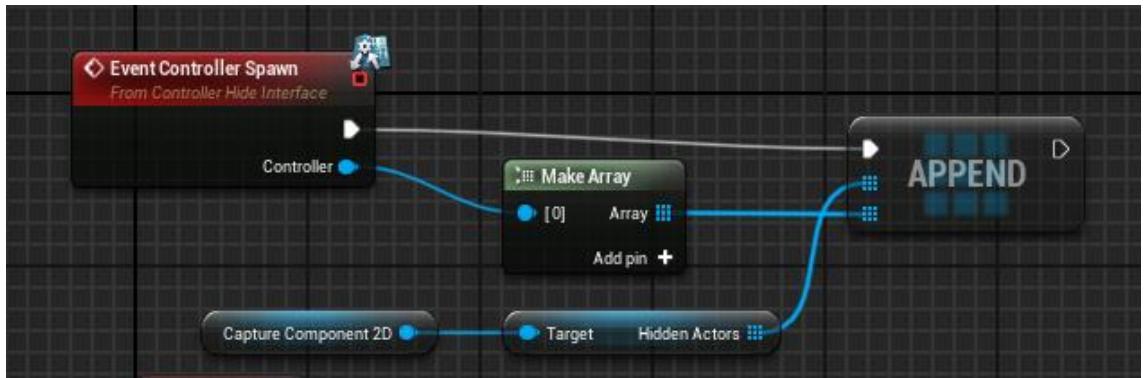


In S.012 installierten wir dann die Kamera ebenfalls neben dem Pult in der Ecke, sodass der Eingang in den Raum zu sehen war. Am Ende des Projekts stellten wir in der Gruppe jedoch fest, dass aufgrund der Größe dieses Raumes noch eine weitere Kamera von Nöten war, also platzierten wir noch am gegenüberliegenden Ende eine Kamera über dem Sicherungskasten. Um die Einrichtung der PC-Sichten vorerst abzuschließen, vergaben wir dann noch im Input-Tab der Unreal-Engine einige Hotkeys, mit denen man dann durch die Kameras durchschalten und das Hackermenü öffnen konnte. Wir erzeugten einen Integer, mit dessen Hilfe wir die aktuelle Kameransicht speicherten. Diesen ließen wir durch Blueprint auf Knopfdruck erhöhen bzw. erniedrigen. Daraufhin wurde mit dem neuen Wert die aktuell angezeigte Kameransicht gewechselt. Ebenfalls ließen wir über einen Boolean prüfen, ob das Menü offen ist oder nicht. Um die grundsätzliche Einrichtung der Kameras abzuschließen veränderten wir noch deren Gamma Wert, sodass der PC-Spieler in der dunklen Atmosphäre alles gut sehen konnte.



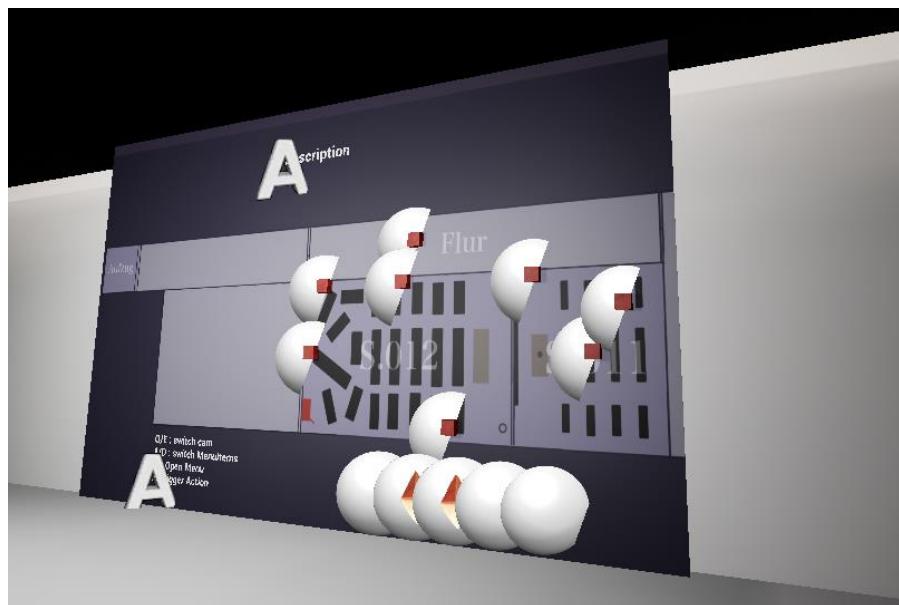
4.2.2 HIDDEN ACTOR LISTE

Anfangs stellte es sich als schwierig heraus, wie geplant den VR-Spieler für die Kameras unsichtbar zu machen. In Unreal gab es hierfür ein Array in Kameraobjekten namens „HiddenActors“. In diese Liste konnte man dann alle möglichen Szenenobjekte hinzufügen und die Kamera zeigte diese Objekte nicht mehr an. Dies war aber nicht möglich, da die Motion Controller des VR-Spielers erst zur Laufzeit erstellt wurden. Nach einigem Ausprobieren verschiedenster Lösungsansätze kamen wir dann auf die Idee, nach dem Erstellen dieser ein Event abzusetzen. Innerhalb des Kamera Blueprints konnten wir somit dann während der Laufzeit die Controller an besagtes Array anhängen. Dadurch war der VR-Spieler wie gewünscht nicht mehr für die Kameras sichtbar. Anfangs hingen wir auch den MotionControllerPawn an die HiddenActor Liste an, jedoch entschieden wir uns dann in der Gruppe dafür, dass der Raumkegel des VR-Raumes sichtbar sein sollte, um dem PC-Spieler einen Anhaltspunkt zu geben wo sich der VR-Spieler befand. Deswegen löschten wir den MotionControllerPawn wieder aus besagter Liste.

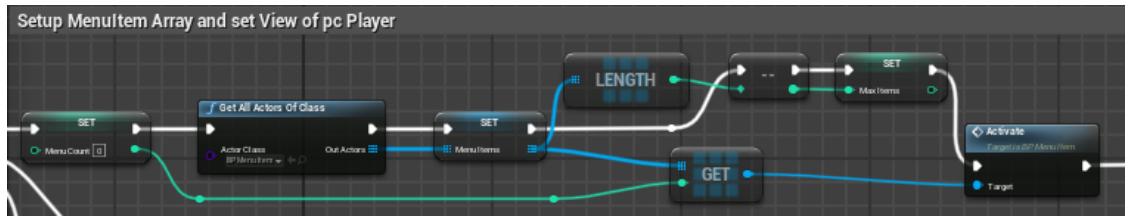


4.2.3 HACKERMENÜ

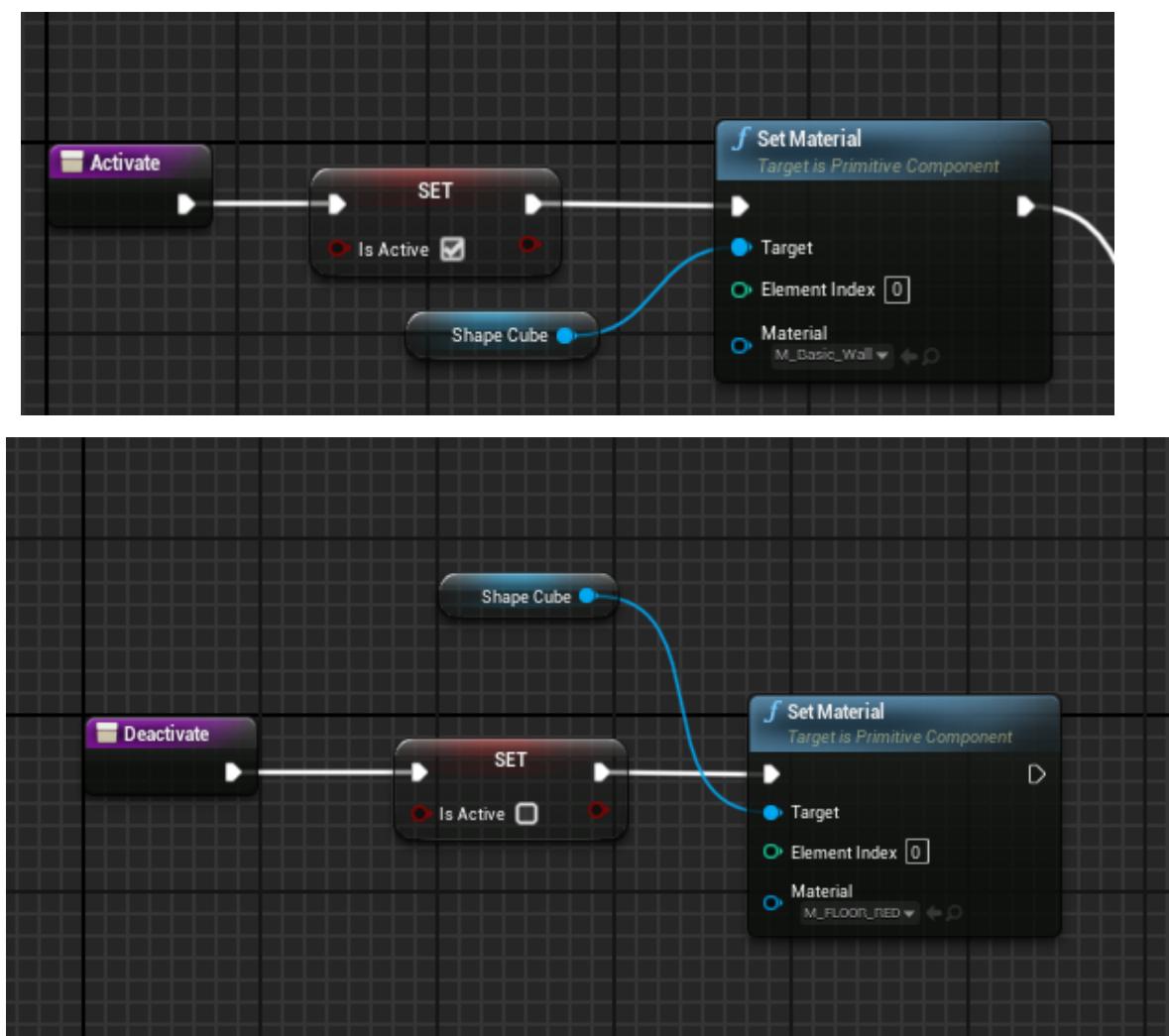
Im isolierten Hackerraum erstellten wir eine Ebene auf der dann das Hackermenü angezeigt werden sollte. Darauf legten wir den ersten Entwurf des Grundrisses, den wir vom Design-Team erhalten hatten. Als Nächstes sollten darauf die ersten Menüpunkte dargestellt werden.



Dazu erstellten wir ein neues Blueprint mit einem einfachen Würfel-Mesh und verteilten diese auf dem Hackerbildschirm. Die einzelnen Menüpunkte wurden zum Programmstart in einem Array gespeichert und konnten somit durch einen Integer, ähnlich wie beim Durchschalten der Kameras durch Knopfdruck durchgeschalten werden.



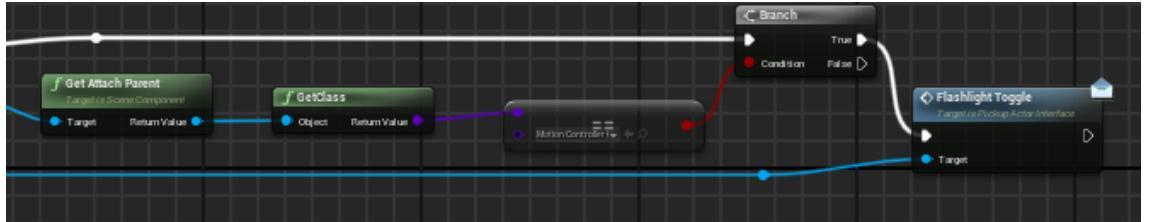
Dafür implementierten wir noch zwei kleine Methoden im Blueprint der MenuItems, wodurch sie aufleuchteten, wenn sie ausgewählt wurden und wieder dunkel wurden wenn sie abgewählt wurden.



Als Nächstes implementierten wir noch eine Methode namens „Interact“, die ausgelöst wurde, wenn der PC-Spieler die Taste „F“ drückte. Diese sollte dann dafür sorgen, dass die jeweilige Aktion des Menu Items durch ein Event ausgelöst wird. Darüber hinaus stellten wir die Hacker-View auf eine orthografische Sicht um, damit die Kuben nur noch als Rechtecke zu sehen waren.

4.2.4 MENU ITEMS

Die erste funktionierende Hacker-Aktion war es, die Taschenlampe des VR Spielers aus- oder anzuschalten. Dazu erstellten wir ein Blueprint, das vom Grundblueprint des Menu Items erbte. Innerhalb dieses wurde beim Interact-Event nach allen Taschenlampen Komponenten innerhalb der Szene gesucht. Daraufhin prüften wir, ob sich jene an der Hand des VR-Spielers befinden. Ist dies der Fall, dann wurde dieselbe Aktion gestartet, die der Spieler beim Halten der Taschenlampe durch Drücken eines Facebuttons auslösen konnte, um die Taschenlampe an- oder auszuschalten.

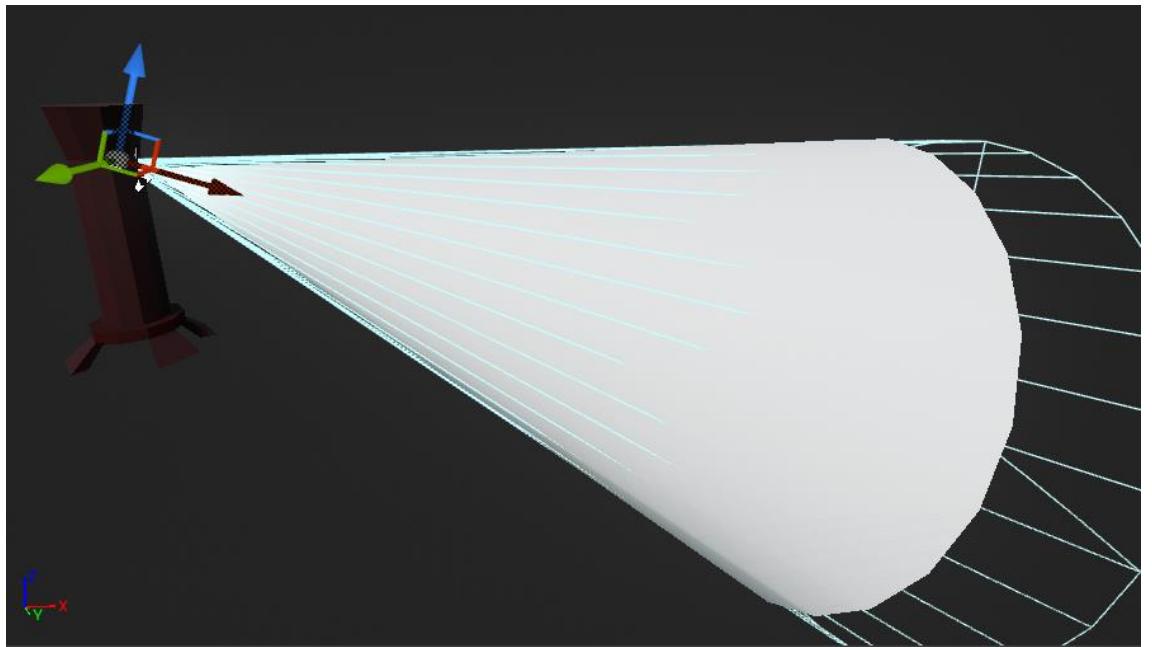
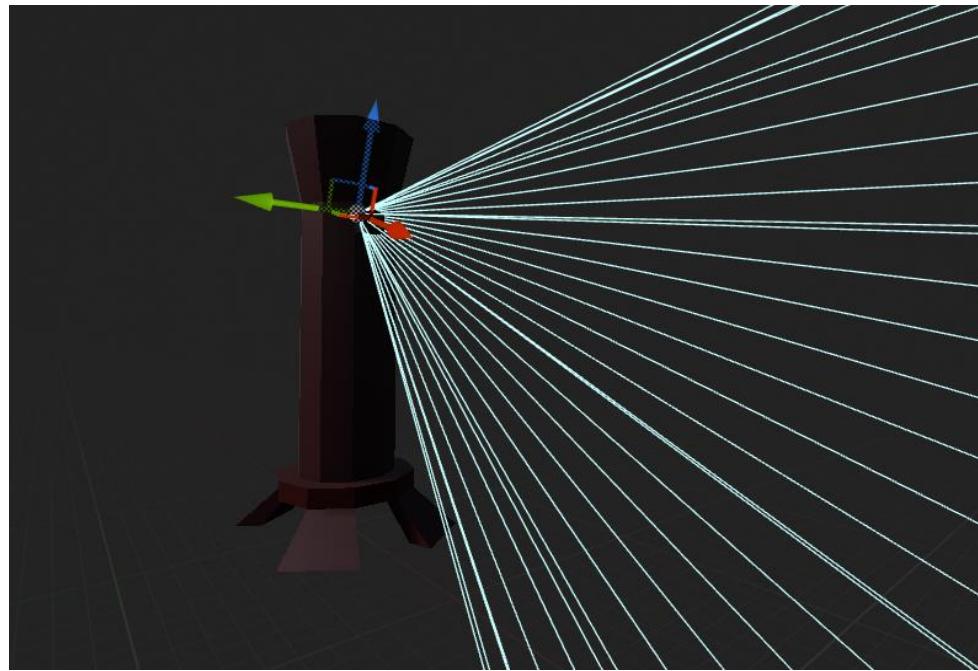


Anschließend brachten wir an jede Kamera einen Text an, der die Bewegungsoptionen für den PC-Spieler anzeigen. Diesen Text mussten wir nun jedoch vor dem VR-Spieler verstecken, was erst nicht funktionieren wollte. Als letzten Ausweg machten wir den MotionControllerPawn des VR-Spielers mit einer Unreal Methode zum „Owner“ der Texte und konnten somit bei den Texten den Boolean „OwnerNoSee“ aktivieren, wodurch das Problem gelöst wurde. Dies war zwar nicht optimal, da dies nicht der Sinn der Methode war, aber in unserem Kontext nicht weiter schlimm und es erzeugte den erwünschten Effekt.

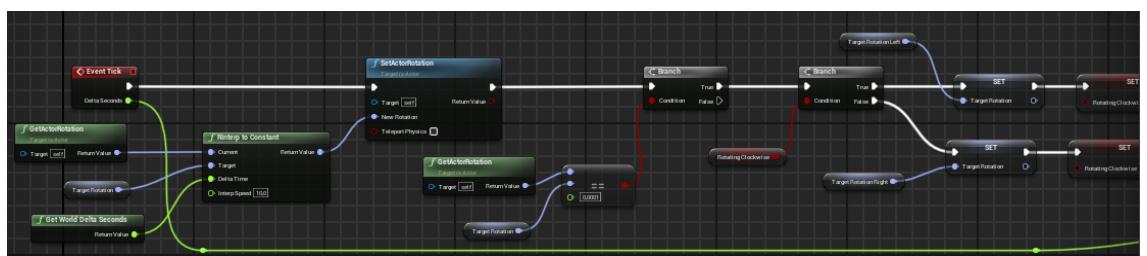
Zu diesem Zeitpunkt bekamen wir vom Design-Team einen aktualisierten Grundriss, der dem finalen Grundriss schon sehr nah kam, den wir dann an die Hacker Sicht anfügten und diesen entsprechend anpassten.



Die nächste Hacker-Aktion sollte die sogenannten Observer in Raum S.012 erscheinen lassen. Diese rüsteten wir erst mit einem Spotlight aus, damit der Sichtkegel dieser klar ersichtlich für den VR-Spieler war. Daraufhin fügten wir diesen auch noch ein Kegel-Objekt hinzu, mit dem man dann eine Kollisionsabfrage implementieren konnte.



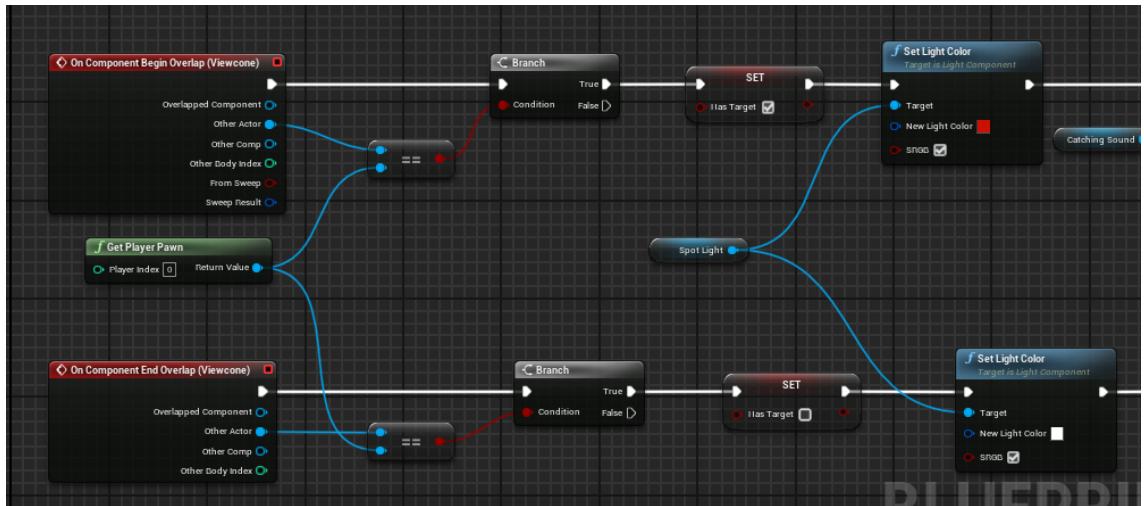
Daraufhin implementierten wir noch eine Drehung für die Observer, sodass sie sich automatisch von ihrem Startpunkt aus erst um 90 Grad gegen den Uhrzeigersinn und danach jeweils um 180 Grad in die entgegengesetzte Richtung drehen.



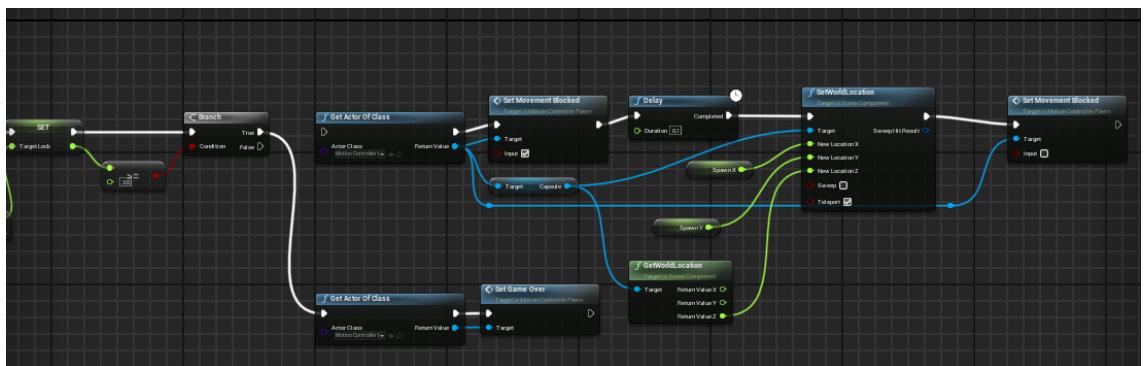
Der Sichtkegel wurde dann noch soweit angepasst, dass er zwar Collision-Events abfeuerte, wenn eine Kollision zustande kam, jedoch sonst keine Physik

beeinträchtigte, denn anfangs schob dieser noch den VR-Spieler durch die Gegend. Schlussendlich wurde er noch unsichtbar gemacht.

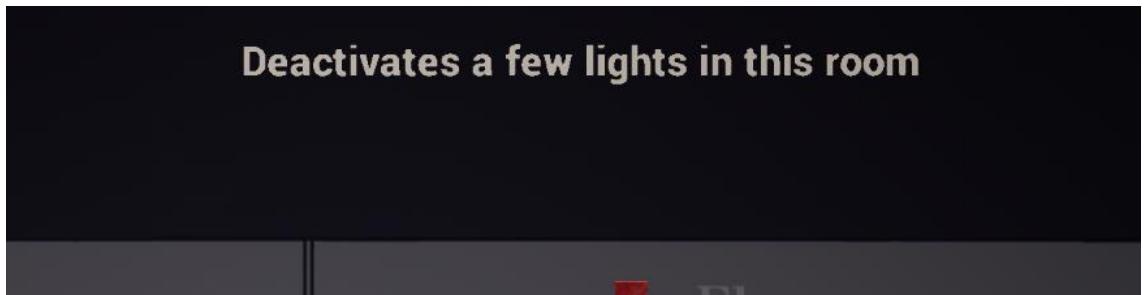
Zur Kollisionsabfrage wurden noch einige Codeabschnitte an den Blueprint des Observers angefügt, die einen Boolean setzten, wenn sich der VR-Spieler innerhalb des Sichtkegels befand. Außerdem änderte das Licht des Spotlights seine Farbe von Weiß auf Rot.



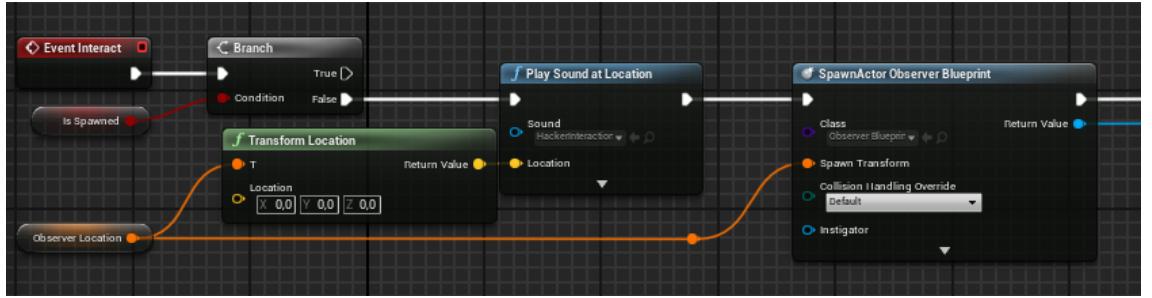
Wenn sich der Spieler zu lange im Sichtkegel eines Observers befand, sollte dies zu einem „Game Over“ führen. Da dieser Zustand aber noch nicht im Spiel vorhanden war, versuchten wir, den VR-Spieler, nachdem er sich drei Sekunden im Radius befand, an den Anfang des Raumes zu teleportieren. Dies funktionierte allerdings nicht, weswegen wir den Code vorerst abhingen. Auch eine spätere Implementierung eines Game Over Bildschirms funktionierte nicht, da er entweder gar nicht auftauchte oder garantiert auftauchte, auch wenn der Spieler den Bereich frühzeitig verließ.



Wir fügten an dieser Stelle verschiedene Beschreibungen an die Menüpunkte und ließen diese vom jeweilig ausgewählten Punkt über dem Hacker-Menü anzeigen, damit der PC-Spieler eine Übersicht über die verschiedenen Aktionen bekam.

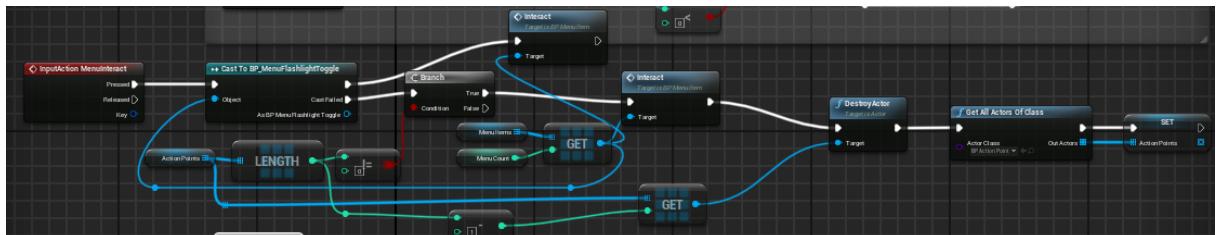
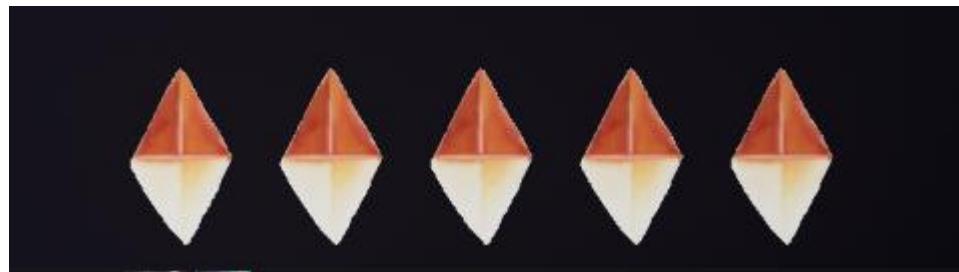


Um die Observer im Raum zu platzieren, erstellten wir wieder ein neues Blueprint, das von MenuItem erbte. Dieses erhielt dann jeweils die Position in einer globalen Variablen übergeben an deren Stelle ein Observer erstellt wurde, wenn das jeweilige Interact-Event ausgelöst wurde. Zum Schluss zerstörte sich der Menüpunkt selbst, da nur einmal an der jeweiligen Position ein Observer erstellt werden sollte.



In diesem Zuge implementierten wir dann auch eine ActionPoint Leiste, sodass der Hacker bei jedem Benutzen einer Fähigkeit einen von 5 Punkten verlor. Ausgenommen dabei war das Umschalten der Taschenlampe. Diese Aktion konnte beliebig oft ausgeführt werden, ohne einen Punkt zu verbrauchen. Sobald alle aufgebraucht waren, waren keine Kommandos mehr möglich.

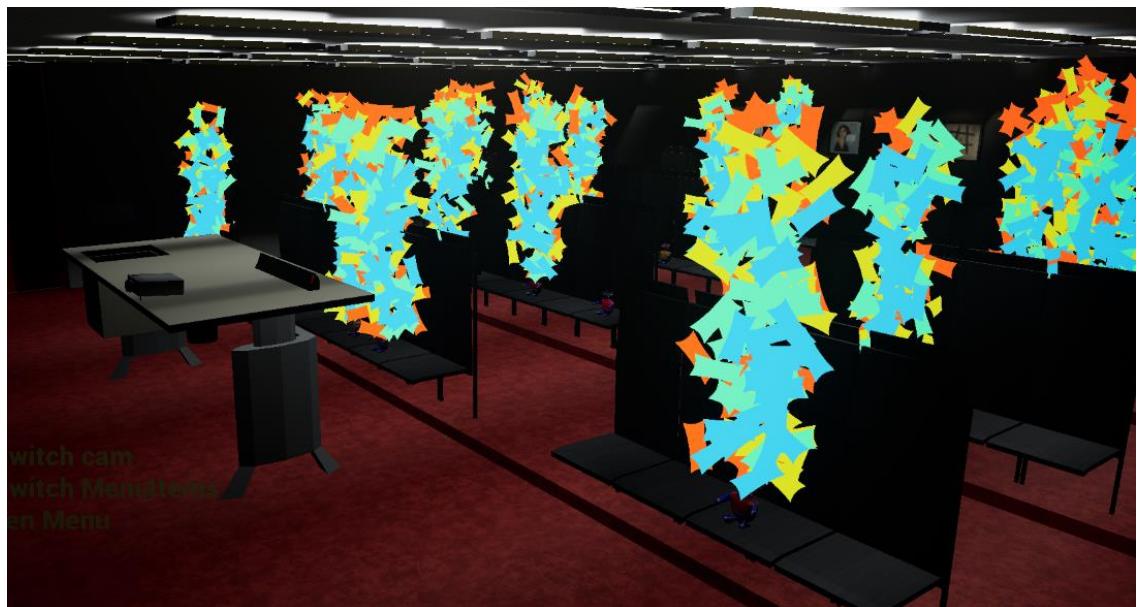
Diese wurden ähnlich wie bereits die MenuItems in Laufzeit in ein Array gelegt und dann im Event Interact einzeln gelöscht und die Liste wieder angepasst.



Die nächsten zwei implementierten Hacker-Aktionen waren das Blocken der Ausgänge der beiden Räume S.011 und So.12. Hierzu erstellten wir zwei Blueprints, die jedoch vom Aufbau sehr ähnlich waren. Beide machten Möbel für den VR-Spieler interagierbar, jedoch war eines für Stühle, die wir neben den Ausgang von S.011 platzierten und das andere war für Tische, die für den Raum S.012 bestimmt waren. Diese wurden dann durch ein neues, von MenuItem erbendes, Blueprint an neue Positionen gerückt, nachdem der PC-Spieler den entsprechenden Knopf drückte. Hierbei mussten wir noch einige Einstellungen innerhalb der Möbel anpassen, damit sie den Spieler blockierten, aber dennoch noch aufhebbar blieben und den Spieler nicht durch die Gegend schoben.



Ein weiteres Kommando, das über das Hacker-Menü ausgelöst werden sollte, war ein Schreckeffekt in S.011. Dabei sollten alle nicht interagierbaren Puppen im Raum auf Knopfdruck Konfetti feuern und einen Ton abspielen. Das Konfetti-Schießen war bereits implementiert, also tauschten wir lediglich die nicht interagierbaren Puppen durch eine neue Version der Puppen mit angehängtem Blueprint aus und ließen bei Interaktion mit einem neu angefertigten Menüpunkt die Funktion „ShootConfetti“ ausführen.



Die letzte Aktion, die wir schließlich noch für den Hacker entwickelten, war das Dimmen des Lichts in S.011. Dies erreichten wir durch Anlegen eines neuen Arrays, dem wir dann ausgewählte Lampen hinzufügten und diese bei Interaktion auf nicht mehr sichtbar stellten. Wir fügten daraufhin noch den meisten Menüpunkten eine Selbstzerstörung hinzu, damit diese nur einmal aufgerufen werden konnten. Lediglich das Umschalten der Taschenlampe ließen wir noch mehrere Male ausführbar.

Zu guter Letzt implementierten wir noch einige Kleinigkeiten für die finale Präsentation des Projekts, wie beispielsweise das Zurücksetzen der Szene auf Knopfdruck, was das Präsentieren etwas leichter machte.

5 MODELLIERUNG

5.1 VORRAUSSETZUNGEN

Marc

Der erste Schritt bestand daraus, in der 3D-Modellierungs-Software Blender ein einheitliches Maß für unsere Objekte zu bestimmen. Dies ist nötig, um Objekte innerhalb des Unreal Engine Editors nicht mehr skalieren zu müssen, da so schnell Unregelmäßigkeiten auftreten. Der erste Gedanke war, die Unit Scale zu verändern und diesen Wert allgemein festzuhalten. Nach einigen Test-Versuchen mit dem Modell des Ganges jedoch fiel Niklas auf, dass für unsere Zwecke der standardmäßige Unit Scale Wert von 1,0 perfekt für die Arbeit in der Unreal Engine ist. Zur Orientierung macht es Blender uns sogar möglich, während des Modellierens mithilfe der Viewport Overlay-Einstellungen eine dynamische Kantenlängen-Anzeige zu aktivieren und so ein direktes Feedback zu den tatsächlichen Ausmaßen zu bekommen. Aufgrund der Unit Scale von 1,0 ist dieser Kantenlängen-Wert in Meter bemessen. Mit diesem Wissen konnten wir nun genaue Meshes erstellen, die den Proportionen der realen Welt fast perfekt entsprachen.

Der zweite, schnellere Schritt sah vor, ein Export-Format für unsere Meshes in die Unreal Engine zu finden. Durch Ausprobieren stellte sich heraus, dass das Exportieren als Filmbox-Datei der beste Weg war, da so Mesh, Materialien und dazugehörige Texturen in einer großen Verpackung exportiert wurden. Nach Export aus Blender in den Projekt Ordner wurde durch Unreal die Option gegeben, neu gefundene Dateien zu importieren, was immer problemlos funktionierte.

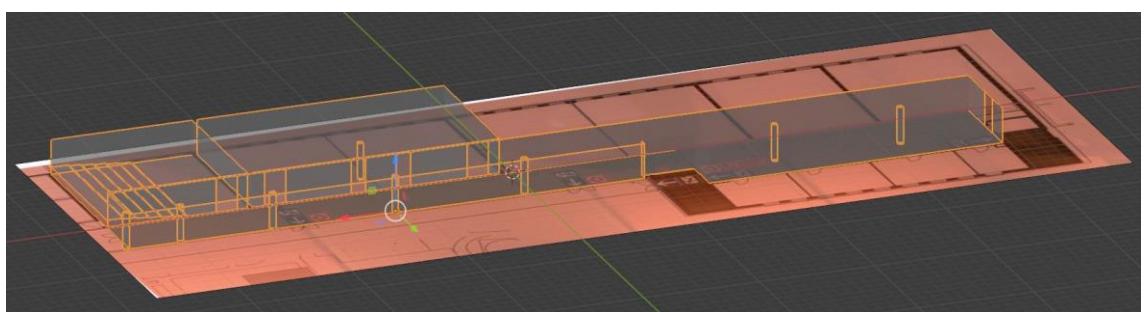
5.2 SZENERIE UND UMGEBUNG

5.2.1 BASISLEVEL

Niklas

Das erste Ziel war es, ein Basislevel zu errichten, welches dazu genutzt werden kann, um Spielmechaniken (zum Beispiel Spielerbewegung und Umgebungsinteraktion) zu testen und auf diesem weiter aufzubauen.

Im ersten Anlauf wurde das gesamte Gemäuer, samt Boden, Decke und Wänden, in einem einzelnen, zusammenhängendem Mesh konstruiert.



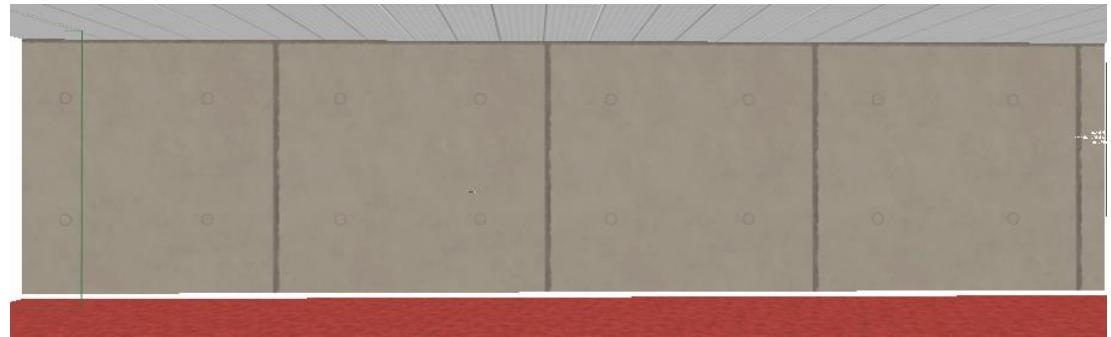
Mit diesem Prototyp wurde es schnell klar, dass ein einzelnes Mesh des Gebäudes einige Probleme mit sich bringt.

Der erste Test zeigte, dass die Unreal Engine und ihr VR Modul nicht gut mit diesem Setup klarkamen. Teleportation des Spielers war nur sporadisch möglich, obwohl das Nav Mesh keine Löcher oder Fehler aufwies. Durch das Ersetzen des Bodens mit mehreren Grundflächen, war das Problem gelöst.

Mit dieser Umstellung kam ein weiterer Vorteil. Materialien und Texturen konnten um einiges einfacher den unterschiedlichen Teilaräumen zugeordnet werden.



Um diesen Vorteil auch auf Wände und Decken anwenden zu können, wurden diese ebenfalls aufgeteilt und bekamen ihr eigenes Mesh und Material.



Dennoch diente das initiale Mesh dazu, im restlichen Levelbau alle Wände und Säulen, welche später dazukamen, originalgetreu zu platzieren.

5.3 MATERIALIEN: GEBÄUDE

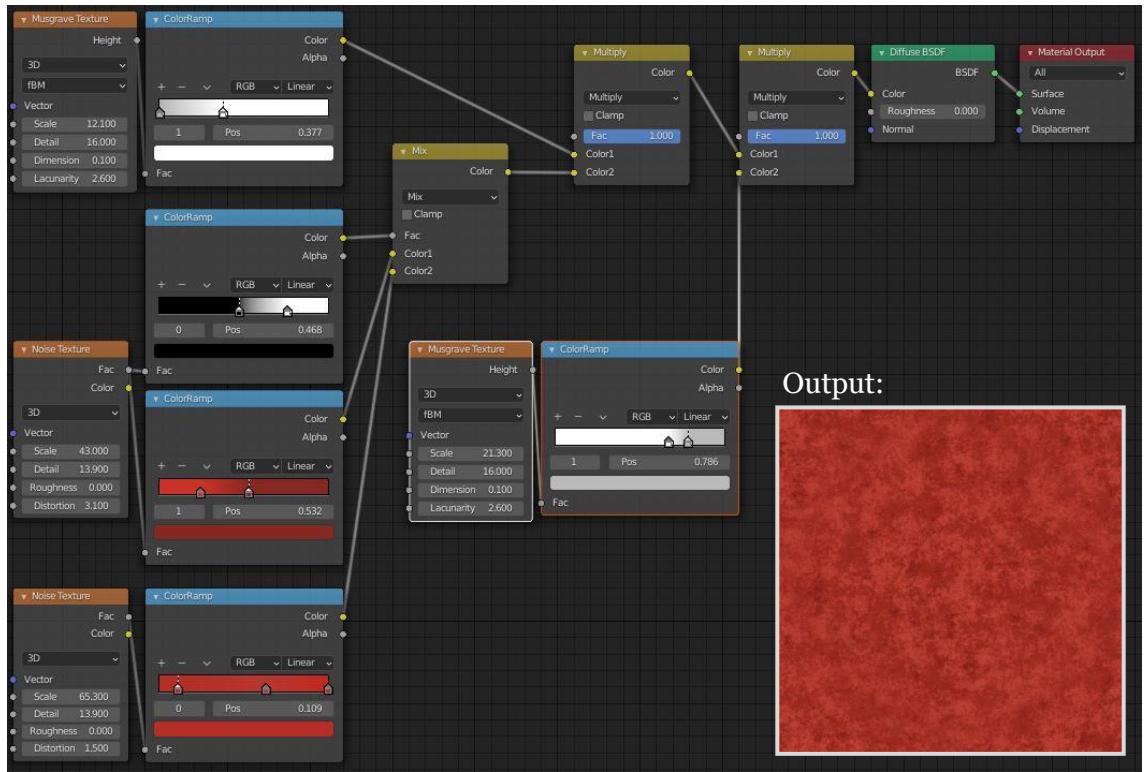
Da wir überall wo möglich versucht haben, selbsterstellte Ressourcen zu verwenden, kamen für die unterschiedlichen Oberflächen verschiedene Methoden zur Texturerstellung zum Einsatz.

5.3.1 FUßBODEN IN DEN VORLESUNGSSÄLEN

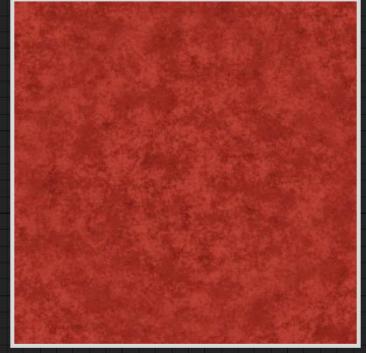


Dank der organischen Natur des Musters konnte in Blender eine prozedurale Textur erstellt werden. Dafür kam der integrierte Shader-Graph zum Einsatz. Durch das Verbinden von Nodes und Anpassen der gegebenen Parameter konnten interessante und einzigartige Muster generiert werden. Diese Methode hatte den Vorteil, ein detailliertes und gleichmäßiges Bild zu generieren.

Das Ergebnis war aber nicht nahtlos. Um das zu korrigieren, wurde in der Bildbearbeitungssoftware Gimp der 'Tile Seamless' Filter angewandt, welcher bei organischen Bildern ein gutes Ergebnis lieferte.



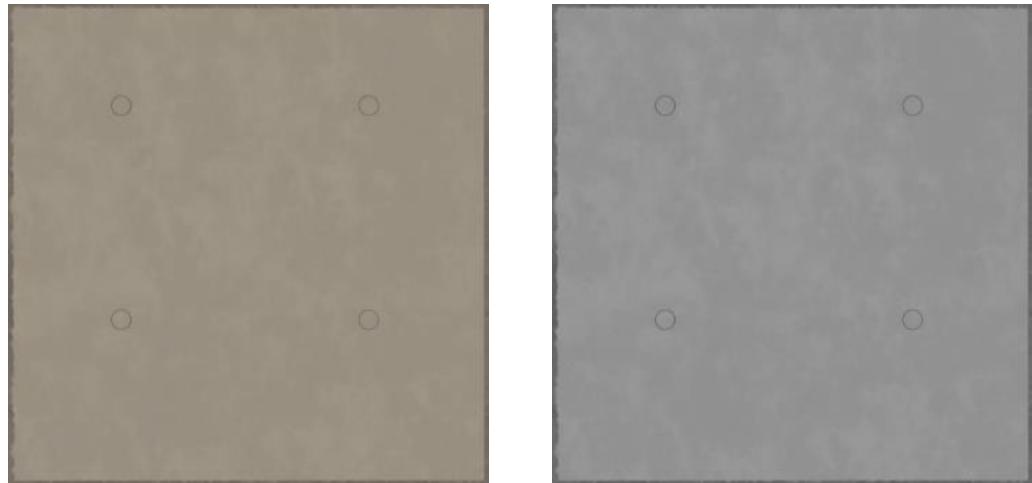
Output:



5.3.2 WÄNDE IN DEN VORLESUNGSSÄLEN

Für die Betonwände innerhalb der Vorlesungssäle bot es sich an, die Textur von Hand zu malen. Da diese nur sehr gering beleuchtet waren, war eine grobe Umsetzung der Oberflächentextur ausreichend.

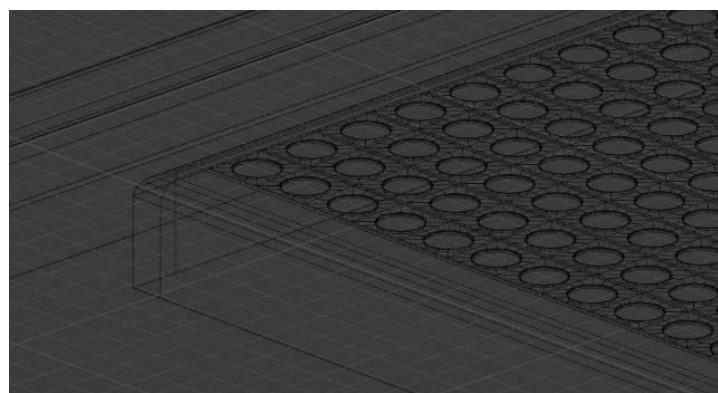
Um ein wenig Struktur zu erzeugen, diente eine schwarz-weiß Version der Textur als Specular- und Bumpmap.



5.3.3 DECKE IN DEN VORLESUNGSSÄLEN



Um die geometrische Struktur der Deckenpaneele erzeugen zu können, wurde die Textur von einem 3D Model gerendert.



Diese Methode benötigte zwar viel Prozessorstärke, um die über fünf Millionen Vertices zu berechnen, aber lieferte eine realistische Skalierung der über 22.000 Löcher.

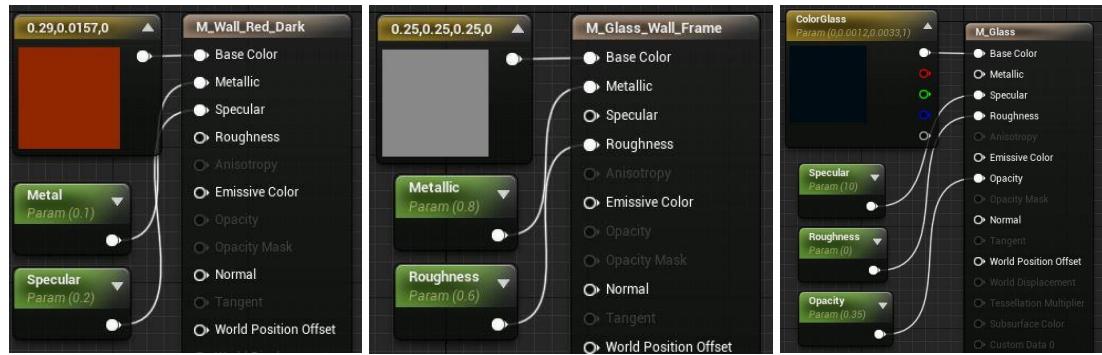
Für zukünftige Projekte würde es sich lohnen, eine mögliche Alternative in einem 2D Grafikprogramm zu finden.

5.3.4 FUßBODEN IM HAUPTGANG

Da dieses Muster mehr Details aufwies, welche schwieriger zu reproduzieren waren, wurde hierfür ein Foto des Bodens angepasst. In Gimp konnte ein passender Ausschnitt perspektivisch korrigiert und farblich ausgeglichen werden.

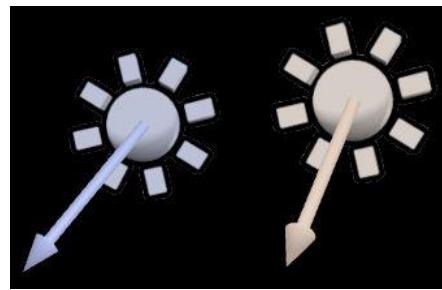


Für den Großteil der restlichen Objekte waren die UE Materialien mit angepassten Parametern ausreichend.



5.4 LICHT

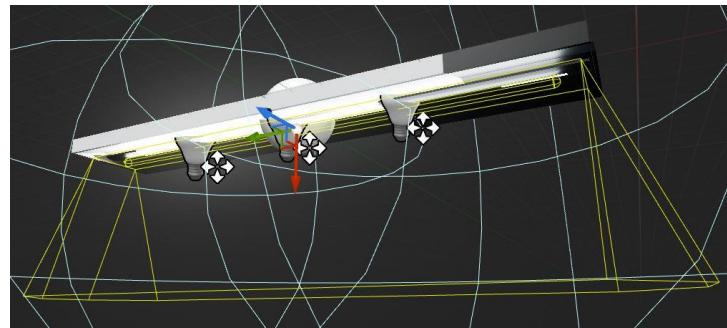
Für die Beleuchtung des Levels wurden verschiedene Arten von Lichtquellen verwendet, um eine stimmige Atmosphäre zu erzeugen.



Für die natürliche Beleuchtung von Sonne und Mond kam das „Directional Light“ zum Einsatz.

Diese Lichtquelle gab ein globales und paralleles Licht, ähnlich wie ein weit entfernter direkt oder indirekt leuchtender Himmelskörper, ab.

Der Wechsel zwischen den Tageszeiten wurde mittels zwei Quellen – mit einer warmen/hellen und mit einer kalten/dunklen Farbe – gelöst. Zwischen diesen konnte bei Bedarf gewechselt werden.

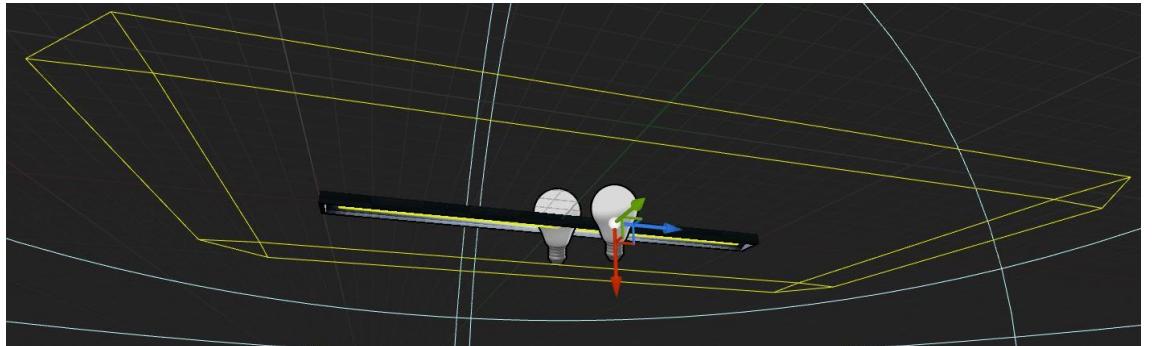


Um in den Räumen ein atmosphärisches Licht zu erzeugen, wurde eine Kombination aus einem gestrecktem „Point Light“ und einem „Rectangle Light“ verwendet.

Das „Point Light“ diente dazu, die Helligkeit der Leuchtstoffröhre auf das Gehäuse des Deckenlichtes zu emittieren.

Das „Rectangle Light“ erlaubte eine kontrollierte Beleuchtung der Umgebung mit harten Schatten. Damit konnte ein starker Kontrast zwischen hellen Flecken, direkt unter den Lampen, und dunklen Schatten mit einem geringen Übergang erzeugt werden.

Ebendiese Methode wurde auch für die Deckenbeleuchtung im Hauptgang verwendet. Nur wurde hier das „Rectangle Light“ angepasst, um eine weichere und flächendeckendere Beleuchtung zu erzeugen.



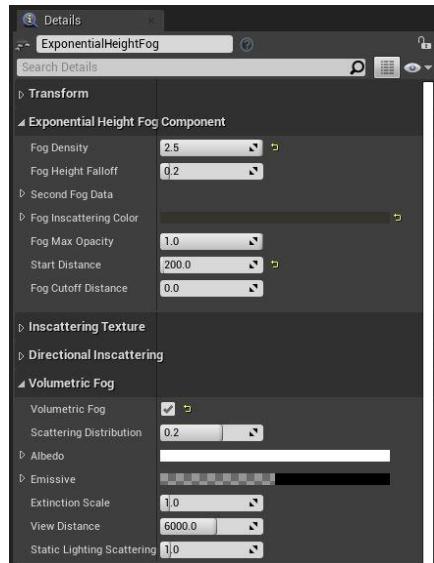
5.5 NEBEL

Um eine düstere Atmosphäre zu erzeugen, kam kurz nach Beginn des Spiels ein dunkler Nebel zum Einsatz.



Zuerst wurden dafür Lösungen mit Partikel Systemen und einem Post-Processing-Shader ausgetestet, um die Nebelschwaden auf den Außenbereich beschränken zu können. Ein Partikelsystem hätte diese Möglichkeit geboten, erwies sich jedoch als sehr ressourcenintensiv und schwer optimierbar. Der Post-Processing-Shader war dagegen sehr sparsam, aber nicht auf gewünschte Bereiche zu beschränken, ohne harte Schnitte zu erzeugen.

Schließlich fiel die Entscheidung auf den UE internen „Exponential Height Fog“. Dieser bot einen sehr stimmigen und realistischen volumetrischen Nebel, welcher über wenige Parameter einfach angepasst werden konnte.



Nach ein paar Versuchen mit diversen Farbvariationen ergab sich ein stimmiges Bild.

Dieser Nebel ließ sich leider auch nicht lokal begrenzen, aber lieferte entgegen den Erwartungen auch innerhalb der Räume eine stimmige Atmosphäre.

Zusätzlich bot er den Pluspunkt, dass seine volumetrische Eigenschaft sich auch auf Lichtquellen auswirkte und somit die Immersion enorm verstärkte.

Besonders vorteilhaft hat sich das bei den Observern gezeigt. Anstatt, dass der Lichtkegel nur auf Oberflächen zu sehen war, erzeugte er ein räumliches Feedback. Sie wirkten damit um einiges prägnanter und somit bedrohlicher. Auch, dass Sie im Nebel nicht eindeutig zu erkennen waren, trug zum Unwohlsein des Spielers bei.

5.6 DEKORATIONEN

Um dem Korridor etwas mehr Leben einzuhauen wurden kleine Details hinzugefügt.

5.6.1 PLAKAT

Das für die AK-Gaming LAN-Party im November 2020 von Uzu Baser erstellte Plakat, wurde netterweise von ihr zur Verfügung gestellt.

Dieses verzierte die beiden Glastüren an beiden Enden des Hauptganges.



5.6.2 RAUM BESCHRIFTUNG & „KEIN EINGANG“ SCHILD

Für die Beschriftung der Räume und Schilder wurde mit eigenen Texturen angelegt. Alternativ hätte man auch ein Font-Objekt der Unreal Engine verwenden können.

Zusätzlich zur Atmosphäre trugen beide Objekte zur besseren Orientierung für den VR-Spieler bei. Die Raumnummerierung war originalgetreu und die „KEIN EINGANG“ Schilder wurden nur an Türen angebracht, welche nicht geöffnet werden konnten.



5.6.3 NOTAUSGANGSSCHILD

Um die Ausgänge hervorzuheben, wurden über diesen Notausgangsschilder aufgehängt.

Es leuchtet grün.



5.7 MODELLIERUNG RAUMFÜLLENDER OBJEKTE

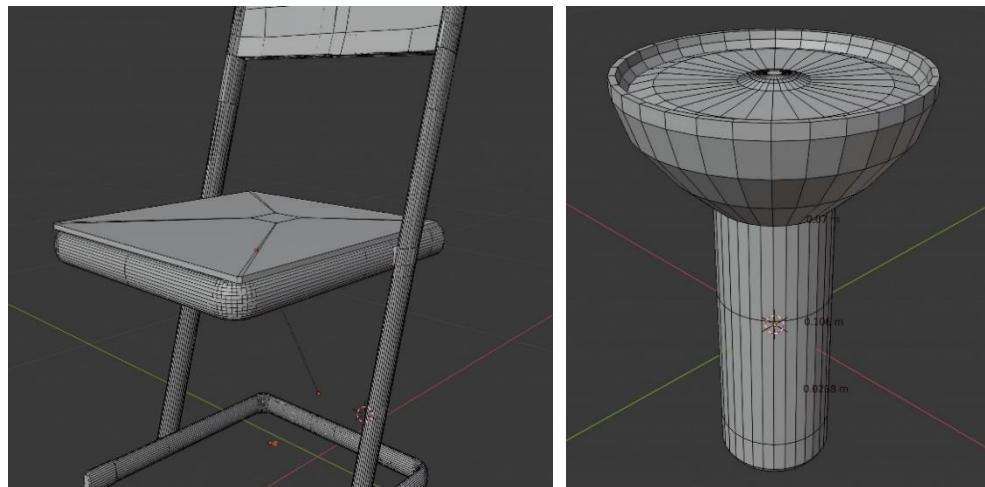
Marc

Gleich zu Beginn unseres Projektes war klar, dass wir viele Modelle von den in der „echten Welt“ vorhandenen Möbeln in den Vorlesungsräumen brauchten, da unser Spiel realistisch wirken sollte. Nach der Beschaffung von Fotoaufnahmen der Innenräume der Hochschule war es meine Aufgabe, diese Objekte nun zu modellieren und zu texturieren.

Mit den Grundthemen der Modellierung geklärt, machten wir uns in einer Besprechung Gedanken, welche Modelle wir als zu Beginn brauchen würden. Es war wichtig, dass zu Beginn die Objekte wichtig waren, mit denen der Spieler schlussendlich interagieren konnte. In unserem Fall waren das die sammelbaren Teile der Taschenlampe,

bestehend aus Batterie, Glühbirne, Gehäuse der Taschenlampe und natürlich die fertige Lampe an sich. Außerdem nahmen wir uns vor, einen der Stühle aus dem Raum S.012 zu modellieren. Als Vorlage für die Batterie nahmen wir eine eigene Batterie, die man im Haushalt hatte. Das Mesh zu erstellen, war recht einfach. Auch das UV-Mapping und das Texture-Mapping ging dank einer Internet-Tutorial-Serie des Youtubers Darrin Lile⁷ recht reibungslos. Der Prozess des Texture-Mappings der Batterie war der Punkt, an welchem wir uns dazu entschieden, unsere Texturen größtenteils selbst zu gestalten, teils innerhalb des Texture-Mapping-Tools von Blender, teils aber auch in GIMP. Das hatte den Vorteil, dass man keine Texturen im Internet suchen musste, und so blieb einem die Dokumentation einiger Bildquellen erspart. Andererseits konnten wir so von Beginn jede Textur nach unseren Vorstellungen gestalten und teilweise kleinere Easter Eggs auf den Objekten einfügen. So gestalteten wir beispielsweise den Warnhinweis auf der Batteriehülle nach unseren Vorstellungen. Natürlich versuchten wir trotzdem, möglichst realistisch zu bleiben, was das Design der Texturen anging. Fast jedes Objekt, welches wir während des Projekts modellierten und texturierten, renderten wir und ließen es dem restlichen Team zukommen, um Wünsche und Anregungen zu motivieren.

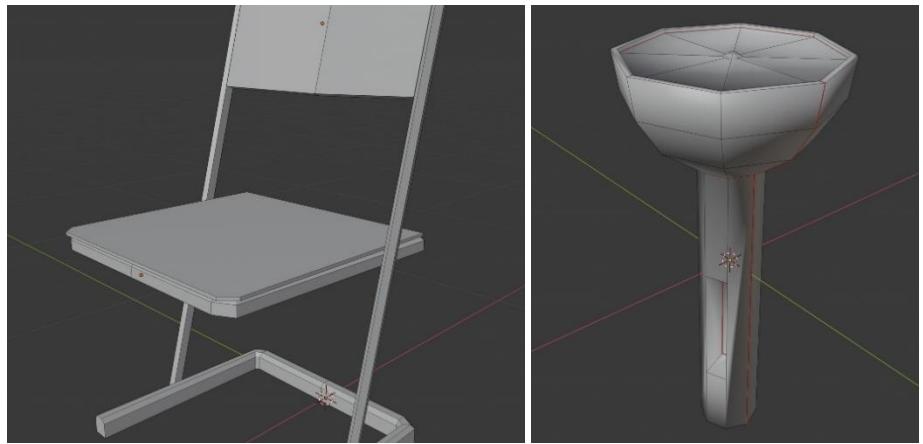
Nach der ersten Woche und den ersten fertigen Modellen äußerte Niklas die Bedenken, dass unsere Meshes zum Teil zu komplex waren, womit er vollkommen recht hatte. Das erste Mesh der Taschenlampe hatte über 1300 Faces (Oberflächen), die Glühbirne 1200 Faces und der erste Stuhl fast 3000 Faces. Besonders letzterer hätte zu einem Problem werden können, da der Plan vorsah, etwa 20 Stühle im Raum zu platzieren. Es stellte sich also die Aufgabe, die Meshes noch einmal neu zu modellieren und dabei so einfach wie möglich zu arbeiten.



Stuhl und Taschenlampe mit zu vielen Faces

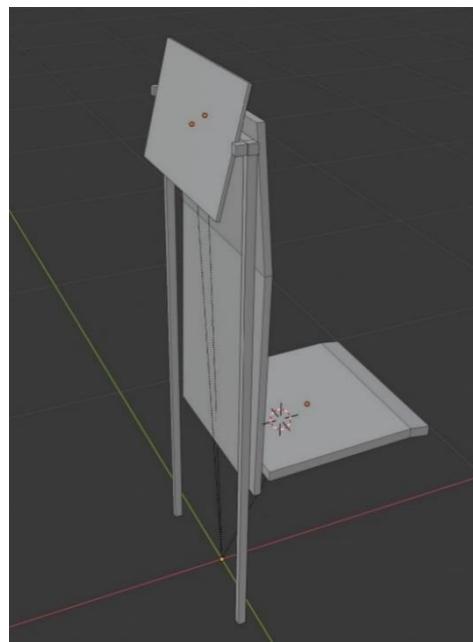
Um die Anzahl der Faces zu reduzieren, hatten wir in diesem Fall zwei Möglichkeiten. Die erste Möglichkeit war, das komplexe Mesh zu nehmen und den Decimate-Modifier darauf anzuwenden. Das birgt allerdings die Gefahr, dass sich das Mesh verformt oder an Stellen Details verloren gehen, welche man behalten will. So entschieden wir uns für die zweite Möglichkeit, das Mesh noch einmal von Grund auf neu zu modellieren. Es handelte sich zu diesem frühen Zeitpunkt im Projekt lediglich um vier Meshes, also ging nicht allzu viel Zeit verloren. Die neuen Modelle sahen nun weniger detailliert aus, doch keinesfalls zu undetailliert. Nach der Texturierung standen die interaktiven Objekte zur weiteren Arbeit bereit.

⁷ YouTube - <https://www.youtube.com/>



Stuhl und Taschenlampe mit reduzierter Face-Anzahl

Die nächsten Modelle sollten nun das grundlegende Mobiliar des Raumes S.011 werden. Dies beinhaltete die Sitzbänke, das Pult und den Projektor. Mit dem Wissen, welches wir während der Modellierung der vorherigen Meshes sammeln konnten, ging dies auch schnell und unkompliziert. Die erste Version, welche in das Projekt implementiert wurde, war allerdings zu hoch, sodass die Beine etwas gekürzt werden mussten. Dies war allerdings kein Problem in sich selbst, sondern das Aktualisieren der Filmbox-Dateien innerhalb des Unreal Editors. Uns fiel auf, dass sobald man die geupdate Filmbox-Datei eines Modelles in den gewünschten Ordner hinzufügte, dieser zwar einen Import erkannte, eben diesen aber nicht durchführen konnte. Das Problem ließ sich beheben, in dem man das alte Modell mitsamt aller Materialien und Texturen aus dem Projektordner entfernte und manuell die geupdate Filmbox-Datei in den gewünschten Ordner im Unreal Editor zog. Dies war sicherlich nicht die eleganteste Weise, das Problem zu lösen, aber so konnten wir uns sicher sein, dass alles genau so war, wie wir es haben wollten.

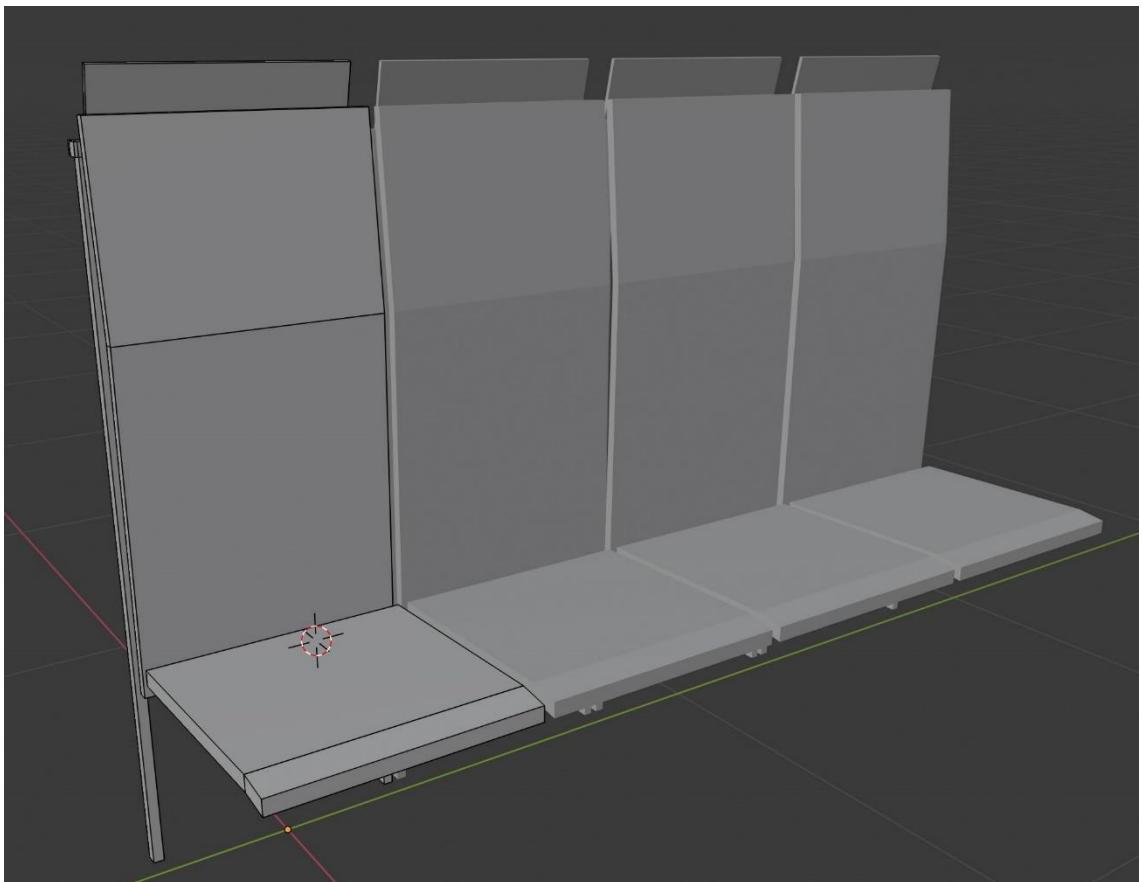


Ein Sitz der Sitzbank im Einzelnen

Die Sitzbänke entsprachen im Aufbau den Maßen der echten Sitzbank. Hierzu mussten wir uns genau überlegen, wie die verschiedenen Teile dieser Bank miteinander zusammenhingen. Jeder Sitz war nämlich in direkter Weise mit dem Tisch der jeweils hinter ihr liegenden Sitzeinheit verbunden. Hierbei waren die Referenzbilder hilfreich,

welche wir zur Verfügung hatten. Eine Sitzreihe bestand aus 4 oder 7 Sitzeinheiten, abhängig davon, ob es sich um die mittlere Spalte (7 Sitzeinheiten) oder die beiden äußersten Spalten (jeweils 4 Sitzeinheiten) handelte. Um nun nicht mehrmals denselben Sitz zu machen oder durch das Kopieren einer Sitzeinheit Unregelmäßigkeiten zu provozieren, benutzten wir in solchen Fällen den Array-Modifier. Dafür modelliert wir zu Beginn das Mesh für eine einzelne Sitzeinheit.

Diese sollte fertig so aussehen, wie man jede Sitzeinheit der Reihe schlussendlich haben wollte. Aus zeitlichen Gründen gaben wir jedem Sitz dieselbe Textur, also war UV-Mapping und Texture-Mapping ebenfalls Teil dieses Schritts. Die Sitzreihe war auch das einzige Modell, für welches eine externe Textur für die Holzoptik benutzt wurde. Der Array-Modifier übernahm nun die Kopie des Objekts. Beim Anwenden des Modifiers entscheidet man sich, auf welcher Achse nun ein Array des Objektes, also eine aufeinanderfolgende Reihe, erzeugt wurde. Dann wählte man den Abstand der Elemente für jede der drei Hauptachsen Arrays zueinander mithilfe des relativen Offsets aus. Nach dem Erstellen einer Sicherungskopie backten wir die ganze Sitzreihe und exportierten sie in den Projektordner.



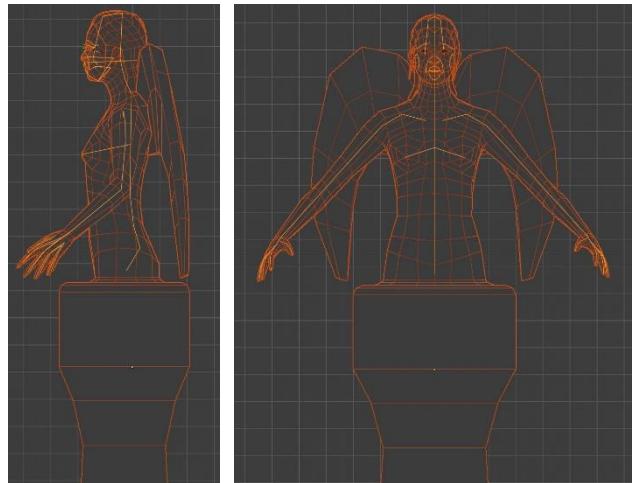
Eine Sitzreihe mit 3-fachem Array-Modifier

Der Projektor stellte keine Probleme dar und in dem folgenden Sonntags-Meeting beschlossen wir, noch einen Mülleimer und eine CD zu modellieren, wobei letzteres von beiden möglicherweise später ein „Collectible“ werden könnte.

Niklas

Die übernatürliche und angsteinflößende Macht, welche die Fachhochschule Kempten heimsuchte, sollte durch eine Engelsstatue à la „Weeping Angels“ verkörpert werden.

Um den Basiskörper, in der sogenannten A-Pose, zu erstellen, kam das open source Programm „MakeHuman“ zum Einsatz. Dieses Tool bot die Möglichkeit, humanoide in unterschiedlichsten Größen und Proportionen zu generieren. Das fertige Modell konnte in verschiedenen Detailgraden exportiert werden, was für Spiele einen Vorteil bietet.



Um die Figur in eine Statue zu verwandeln, wurden die Beine gegen einen Sockel ausgetauscht und ein fließender Übergang erzeugt. Die Haare und Flügel wurden per Hand an die Figur modelliert.

Abschließend wurde das Modell mit dem Animationsskelett, welches von „MakeHuman“ generiert wurde, in seine vier Positionen gebracht. Jede wurde als eigenes Modell dem Spiel hinzugefügt.



Im Unreal Projekt wurden die jeweiligen Events zur Positionierung des Engels, in die bestehende Logik von Dan eingebunden. Diese bestanden daraus, die Stadien des Engels Gegeneinander auszutauschen, indem der vorherige Engel unter und der aktive Engel über dem Boden platziert wurden. Zusätzlich wurden bei den letzten zwei Posen Sounds getriggert. Zum Einen ein Klirren, welches das Durchbrechen der Hand des Engels durch das Fensterglas wiedergab und zum Anderen ein Schrei, um den finalen Jump-Scare zu unterstützen.

Das Auftreten des letzten Engels macht den VR-Spieler zusätzlich bewegungsunfähig. Abschließend wird nach einer kurzen Verzögerung der ‘Game Over’ Bildschirm abgespielt.

Marc

Die Modellierung der Puppen in S.011 war eine spannende Aufgabe, da dies das erste Modell war, bei welchem wir keine Vorlage aus der Realität, sondern alleinig die Zeichnungen des Game-Design-Teams hatten. Deshalb hatten wir ein wenig Freiheit bezüglich der Proportionen, obwohl wir doch versuchten, nahe an den Referenzzeichnungen zu bleiben. Das Endergebnis war zufriedenstellend und die Texturierung geschah fast ausschließlich innerhalb der Blender-eigenen Texture-Mapping Oberfläche.



links: Vorlage der Puppe, rechts: fertiges Modell der Puppe

Das Ziel der Gestaltung der Puppe war, sie möglichst unangenehm wirken zu lassen, weshalb die Augen und Arme asymmetrisch proportioniert wurden. Für das Rätsel dieses Raumes gab es vorerst drei mögliche Kombinationen der Haar- und Oberteilfarbe. Diese waren blond/grün, rothaarig/orange und schwarzhaarig/rot (Haarfarbe/Oberteilfarbe). Die Erstellung dieser Texturen und Materialien war unkompliziert, da lediglich eine Kopie einer Standard-Textur erstellt werden musste, welche man in GIMP dann nach Belieben verändert hat. Die neue Textur exportierten wir jeweils in den Projekt-Ordner des Unreal Projekts und erstellten ein neues Material mit der neuen Textur als Parameter. Da die Puppe korrekt UV-mapped war, musste man dieser nur noch das neu erstellte Material zuweisen.

Nach den Puppen waren die Bilder an der hinteren Wand in S.011 an der Reihe, modelliert zu werden. Diese waren sehr einfach zu modellieren, zu UV-mappen und zu texturieren, insbesondere dank der Zeichnungen des Design-Teams, welche sich problemlos in die Textur einbauen ließen. Nach dieser Modellierung begannen wir, die Logik für das dazugehörige Rätsel zu implementieren. Dieser Vorgang ist in 4.1.7 näher beschrieben. Anschließend an diese Implementierung platzierten wir die restlichen Puppen im Raum, mit denen der Spieler nicht interagieren konnte. Hierzu erstellten wir nochmal sechs zusätzliche Puppentexturen und -materialien, die aus Haar/Shirt Kombinationen bestanden, welche nicht Teil des Rätsels sein konnten (zum Beispiel blond/rot).

Nach der Logik für die Schlüsselvergabe ging es daran, den Rest des Raumes S.011 und S.012 zu möblieren. In S.011 fehlten noch eine Tafel und ein Whiteboard und S.012 hatte nur einen Stuhl vom Anfang und einen Tisch, welchen wir zwischenzeitlich modelliert hatten. Auf dem Whiteboard sollte eine Projektion des Projektors zu sehen sein, welche die Anleitung für diesen Raum zeigen sollte. Dies wurde durch ein Zusammenspiel der Whiteboard-Textur und eines Rectangle-Lights umgesetzt. An der gewünschten Stelle auf der Textur ist die Projektion als Bild eingefügt worden und wurde vom Rectangle-Light so angestrahlt, dass die Illusion einer tatsächlichen

Projektion entstand. Die Tafel hingegen diente keinem genaueren Zweck, lediglich der Dekoration des Raumes und der Authentizität halber.



Whiteboard, welches mit Rectangle Light bestrahlt wird

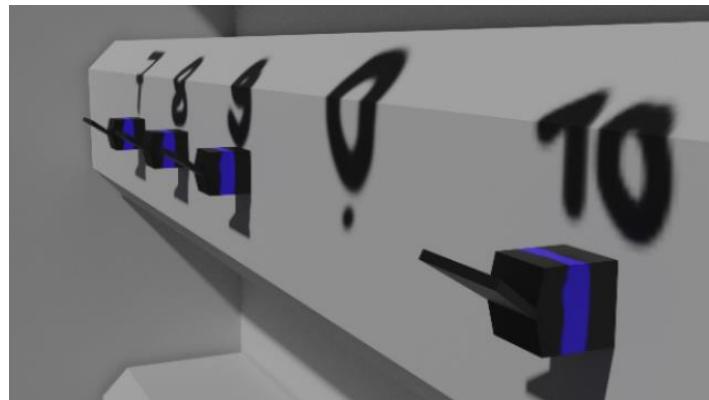
In S.012 fehlte noch viel Mobiliar, unter anderem ein Pult, mehrere realistisch angeordnete Tische und Stühle, ein Waschbecken und ein Schlüssel, welcher ein Collectible in diesem Raum darstellte. Das Waschbecken war recht schnell gemacht, besonders das Becken konnten wir dank Smooth Shading an vereinzelten Bereichen realistisch gestalten. Das Pult in S.012 ist dasselbe wie das in S.011, also traten auch hier keine Schwierigkeiten auf. Tische konnten einfach mehrfach eingefügt werden, da auch in echt alle Tische in diesem Raum gleich waren. Der Schlüssel bekam die Optik eines alten, rostigen Schlüssels, wie man sie aus Mittelalter-Geschichten kennt. Dieses Design wählten wir deshalb so, damit sich der Schlüssel etwas von der Umgebung abhebt und der Spieler intuitiv erkennen konnte, dass es sich hierbei um ein Collectible handelt.

Nach der Platzierung der Tische in S.012 stellte sich die Frage nach der Modellierung des Observers, welcher im letzten Drittel des Raumes den Spieler überwachen und ihn dazu zwingen sollte, in Bewegung zu bleiben. Hierfür gab es nur eine grobe Idee, allerdings kein richtiges Konzept. Wir einigten uns darauf, einfach zu improvisieren und das Modell später dem restlichen Team vorzustellen. Wir entschieden uns gegen eine humanoide Struktur und stattdessen für ein statisch wirkendes Objekt. Er sollte schmal genug sein, um sich problemlos zwischen den Tischen bewegen zu können und trotzdem einschüchternd wirken. Die Entscheidung fiel auf ein rundes Metallobjekt mit vier Beinen, welches auf Kopfhöhe des Spielers einen Bildschirm angebracht bekam, auf welchem in grünen Buchstaben „I am watching you“ geschrieben stehen sollte. Dieser erste Observer wurde vom Team angenommen und so konnten wir mit dem Stromkasten, einem weiteren interagierbaren Objekt im letzten Drittel des Raumes, weitermachen.



Roh-Fassung und finale Version des Observers

Das Design des Strom-Kastens war wieder simpel, da man sich zuerst an einem normalen Sicherungskasten orientieren konnte. Lediglich die Schalter mussten wir vergrößern und uns eine Möglichkeit ausdenken, die nicht interagierbaren Schalter von dem einen Interagierbaren zu unterscheiden. Dies geschah letztlich mit einer Textur, die es so aussehen ließ, als ob der Schalter mit einem farbigen Stück Klebeband markiert ist.



Rendering der Schalter

Der interagierbare Schalter fehlt hierbei, da dieser ein eigenes Objekt darstellen musste. Wie die Logik hinter diesem interagierbaren Schalter aussah, ist in 4.1.8 näher erklärt. Um den Kasten authentisch wirken zu lassen, bearbeiteten wir die Textur so, dass die Innenseite der Kastentür den Anschein hatte, als hing hier ein Zettel mit Erklärungen. Die Textur des Blattes hatten wir uns wie die Holz- und die Schlüsseltextur aus dem Internet geholt⁸.

Mit dem Abschließen des Sicherungskastens war unsere Modellierungsarbeit für das Mobiliar und die Dekoration der Räume fertig.

⁸ PNGKIT - <https://www.pngkit.com>

6 FAZIT

6.1 DAN EISENKRÄMER

Wenn man an einem Projekt arbeitet, hat man häufig klare Vorstellungen, wie das Ergebnis hinterher aussehen soll und wie man dieses erreichen möchte. An diesen Idealen hält man dann fest, bis sich irgendwann einem ein Problem in den Weg stellt, welches ein Umdenken erfordert. Meist sind es gerade die unscheinbarsten Aufgaben, die man bewerkstelligen will, die einem hinterher die größten Probleme bereiten. So war es in diesem Projekt die Tür, die mich immer wieder an den Rand der Verzweiflung brachte und mich wochenlang beschäftigte. Hierbei wollte ich ungerne meine Vorstellung aufgeben, dass der Spieler realistisch mit der Tür interagieren können soll. Durch das Projekt habe ich gelernt, in solchen Situationen erst einmal Abstand von der Aufgabe zu nehmen, mich mit anderen Problemen zu beschäftigen und mit einem neuen Blick auf die Dinge zu der kniffligen Funktion zurückzukehren. Dadurch konnte ich einsehen, dass ich mich von meinen Vorstellungen häufiger auch distanzieren sollte. So schaffte ich es, die Interaktion mit der Tür in wenigen Codeblöcken mithilfe einer Animation zu bewerkstelligen und damit ein Ergebnis zu erschaffen, was meine Vorstellungen überstiegen hat.

Ein klares Ergebnis vor Augen zu haben, ist hilfreich, um zu sehen, worauf man hinarbeitet, aber man sollte sich niemals darauf verbeissen. Dafür haben mir auch die wöchentlichen Treffen sehr geholfen, um zu sehen, wie die anderen mit ihren Aufgaben vorankommen, was ihre Vorstellungen sind und wo sie vielleicht Probleme haben. Wir hatten alle sehr unterschiedliche Stärken und Schwächen, was dazu geführt hat, dass jeder einen klaren Aufgabenbereich hatte, man sich aber dennoch gegenseitig immer ausgeholfen hat, wenn jemand vor Problemen stand. Dadurch hat man auch nicht nur der anderen Person geholfen, sondern hat gleichzeitig auch Ideen und Möglichkeiten entdeckt, die einem bei seinen eigenen Problemen helfen sollten.

Da dies mein erstes Projekt war, an dem mehr als drei Personen beteiligt sind, war nicht nur die klare Aufgabenteilung etwas, was mir geholfen hat den Überblick zu bewahren. Ich merkte schnell, wie wichtig eine klare Strukturierung innerhalb des Projektes war. Wenn immer möglich, habe ich Funktionen erstellt, die man wiederverwenden konnte und auf die auch die anderen Projektteilnehmer aufbauen konnten. Trotzdem lief hier auch nicht immer alles ideal ab und so hatte ich am Anfang des Projektes sehr wenige erbende Klassen, bevor ich erkannt habe, dass es sich lohnt, eine klare Hierarchie aufzubauen, auch wenn man noch keinen Grund dafür sieht und noch keine zusätzlichen Funktionen dadurch einbaut. Etwas, was wir während der Arbeit an dem Projekt leider nicht mehr richten konnten, war eine einheitliche Ordnerstruktur. Für mein nächstes Projekt nehme ich daher mit, dass man eine gut erkennbare Ordnerhierarchie hat, an die sich alle im Projekt halten, damit man Objekte und Blueprints schnell wiederfinden kann und jeder weiß, wo sich was befindet.

Zusammenfassend kann ich sagen, dass mir das Projekt viel Spaß gemacht hat und viel beigebracht hat. Es ist sehr schön, dass hinterher ein vollständiges und in meinen Augen sehr anschauliches Spiel dabei herumgekommen ist, welches seinen Zweck auch erfüllt und seine erschreckenden Momente hat. Die Arbeit zusammen mit den Anderen fand ich auch sehr schön, da wir gut miteinander klarkamen und uns auch sehr gut verstanden haben. Dadurch war es nie ein Problem, um Hilfe oder Tipps zu fragen, wenn man mal nicht weiterkam. Ich hoffe, dass meine zukünftigen Projektarbeiten in einer ähnlich angenehmen Atmosphäre ablaufen werden und ich meine neuen Erfahrungen dort mit einbringen kann.

6.2 SYAFIQAH HUSNA MD SAZALI

Zusammenfassend lässt sich sagen, dass die wichtigste Lektion, die wir bei diesem Projekt gelernt haben, die ist, dass man seinen Teil in einem Team leisten sollte, damit das Projekt reibungslos läuft. Wenn zum Beispiel eine Person beschließt, ihren Teil nicht zu tun, hätte das negative Auswirkungen auf das Projekt und könnte das Projekt um Tage, wenn nicht sogar Wochen zurückwerfen. Dieses Problem trat bei unserem Team jedoch nicht auf. Ich habe mit Freude festgestellt, dass jeder seine individuellen Aufgaben perfekt erledigt hat, so dass das Projekt innerhalb des vorgegebenen Zeitrahmens abgeschlossen werden konnte.

Ich denke, einer der ausschlaggebenden Faktoren, der dies möglich gemacht hat, war die Tatsache, dass wir in einem unserer allerersten Treffen jeder Person eine Aufgabe zugewiesen hatten, die sie selbst zu erledigen hatte. Damit war sichergestellt, dass es nicht zu Überschneidungen von Aufgaben kommen würde. Das heißt natürlich nicht, dass die Teammitglieder die anderen nicht um Hilfe bitten durften. Im Gegenteil, es gab den anderen Teammitgliedern eine breitere Perspektive auf das, was die anderen in der besagten Gruppe taten, und ermöglichte den Austausch von Ideen bezüglich des Projekts.

Ein weiterer Faktor war auch die wöchentlichen Meetings, in denen wir präsentieren mussten, was wir in der Woche geleistet hatten. Meiner Meinung nach waren diese Treffen sehr hilfreich, da sie verhinderten, dass jemand "nachlässt" oder seine Arbeit nicht macht. Es half uns auch zu sehen, wie weit wir mit unserem Projekt gekommen sind.

Alles in allem habe ich mich sehr gefreut, ein Teil dieses Teams zu sein, und war auch mit dem Endergebnis des Projekts sehr zufrieden. Als solches war ich auch sehr dankbar für jede Person im Team, die diese Erfahrung so wunderbar gemacht hatte.

6.3 FEIER JIANG

Abschließend möchte ich sagen, dass es wirklich ein sehr sinnvolles Projekt ist. Ich bekam meine erste praktische Erfahrung, wie man ein Spiel vom Anfang an entwickeln kann: Vom Entwurf (Design), Fertigstellung (Programmierung und Modellierung) bis hin zur Promotion (Videos und Poster). Ich finde, dass unsere Gruppe eine tolle Arbeitsweise hat. Wir haben uns jede Woche getroffen, um unsere Fortschritte zu berichten und auch Ziele für die folgende Woche zu setzen. Jeder bekommt eine klare Aufgabe zugewiesen. Was auch wichtig ist, dass wir kaum Meinungsverschiedenheit haben. Dadurch ist unser Projekt sehr effizient und reibungslos gelaufen.

Ich bedanke mich sehr bei meiner Teammitgliedern, dass sie mich keine Designsache machen lassen, weil Programmierung meine Schwäche ist. Ich habe mich eigentlich nur für Spieldesign interessiert, aber in diesem Projekt, indem ich zum ersten Mal digital Bilder gezeichnet, Musik & Sounds bearbeitet und Videos editiert habe, habe ich auch mehr Spaß entdeckt und mein Interesse an neuen Dingen geweckt. Es ist nur möglich mit der Hilfe von allen anderen und auch dem Vertrauen, das anderen Teammitgliedern mir gegeben haben.

Ursprünglich wurden wir aufgrund unserer gemeinsamen Interessen zufällig in dieser Gruppe zusammengebracht, aber durch das Zusammenarbeiten in den einigen Monaten und die Strebe nach demselben Ziel haben wir uns besser kennengelernt. Die Arbeit mit unserer Gruppe hat mir sehr viel Spaß gemacht und es war ein wirklich schönes Erlebnis.

6.4 SIMON KOMPAß

Insgesamt kann ich sagen, dass ich anfangs große Schwierigkeiten damit hatte mich in das Blueprint-System der Unreal Engine einzuarbeiten. Ich habe bereits jahrelange Programmiererfahrung, doch die Tücken der Blueprints waren mir lange Zeit ein Dorn im Auge. Jedoch habe ich mich mit der Zeit daran gewöhnt und mich sogar ein wenig damit angefreundet, da es durch die geradlinige Struktur gut nachvollziehbar ist was passiert und warum welche Probleme auftauchen. Ein wenig schwierig fand ich ebenfalls das Testen, da wir aufgrund der aktuellen Lage nur jeweils allein am Projekt arbeiten konnten, was das Debuggen deutlich erschwerte, wenn man etwas in VR testen möchte. Innerhalb des Teams gab es zudem gelegentlich einige Kommunikationsprobleme, die mich manchmal warten ließen, da gewisse Grundbausteine, die ich zum Arbeiten benötigte, fehlten. Jedoch hat mich dieses Projekt zum Schluss wirklich begeistert und die gute Teamarbeit hat mich sehr gefreut. Ich habe vieles gelernt, auch wenn ich bereits an einigen Spieleprojekten mitgearbeitet habe.

6.5 NIKLAS WEINHART

Insgesamt ist das Projekt den Umständen entsprechend sehr gut gelaufen. Trotz des bedrückenden Ausnahmezustandes, welcher weder für die Kreativität noch Produktivität förderlich ist, waren alle Gruppenmitglieder sehr engagiert und haben einen jeweils maßgeblichen Beitrag zu unserem Projekt geleistet. So konnten wir ein Projekt auf die Beine stellen, auf welches wir stolz sein können.

Von Beginn an hatten wir eine Struktur, welche sich natürlich nicht solide bis zum Ende durchgezogen hat. Diese erlaubte es uns jedoch an unserer Idee festzuhalten. Angefangen bei den wöchentlichen Meetings, die den Fortschritt und auch andauernde Probleme vor Augen geführt haben, bis hin zur Bereitschaft Aller, sich gegenseitig zu unterstützen. Die Zusammenarbeit hat mir viel Spaß gemacht.

Ich denke, wir haben alle eine Menge dazugelernt und werden zukünftige Projekte mit diesem neuen Wissen besser angehen und darauf aufzubauen.

Für mich speziell habe ich mitgenommen, meine Arbeitszeiteneinteilung neu zu überdenken und zu optimieren. Und auch den wichtigsten Punkt, den ich bei Projekten jeder Art immer wieder wahrnehme, Kommunikation ist das Wichtigste. Sei es mit sich selbst oder mit den Arbeitskollegen. Um Voranzukommen muss man den gleichen Nenner haben und sehen was das gemeinsame Ziel ist.

Rückblickend möchte ich für zukünftige Game Labs einen Verbesserungsvorschlag einbringen. Für den ersten Durchlauf verlief die Organisation gut. Die Freiheit, das Projekt und die dazugehörige Planung selbst zu gestalten, hatte einen guten Lernfaktor. Aus meiner Sicht wäre es aber unterstützend, wenn uns Studenten eine durchgängige projektbezogene Führung zur Anwendung der grundlegenden Methoden des Projektmanagements gegeben wird, wie etwa Projektstruktur und ‚problem solving‘. Damit könnten Frustfaktoren, etwa im Umgang mit Sackgassen und unsauberen Projektstrukturen, gemindert werden.

6.6 MARC VON GLAHN

Es hat sehr viel Spaß gemacht, dieses Projekt so mitzuerleben und an ihm mitzuwirken. Wir hatten gleich zu Beginn eine gute Team-Chemie, was sich immer wieder in den Donnerstags-Meeting mit Herr Dreier oder in den separaten Sonntags-Meetings der Studenten gezeigt hat. Jeder war zu aller Zeit bereit, seine Team-Kollegen zu unterstützen, was besonders zu Beginn wertvoll war, da sich jeder erst einmal in diese

neue Engine hineinarbeiten musste. Auch unsere Team-Größe war nahezu perfekt; Das Team war groß genug, um Aufgaben sinnvoll zu verteilen, aber klein genug um nicht sofort den Überblick zu verlieren.

Persönlich habe ich vor allem neue Kenntnisse im Umgang mit Game Engines mitgenommen. Den Prozess der Entwicklung eines Spiels, die Arbeit im Team und die ersten ernsthaften Schritte mit Git sind ebenfalls Erkenntnisse, die sich in Zukunft lohnen werden. Es war fantastisch zu sehen, wie unser kleines Level wächst, erweitert und gefüllt wird und schließlich spielbar ist und navigiert werden kann.

Das Corona-Semester hat es uns allerdings auch nicht leicht gemacht, das Projekt wie gewollt und gehofft anzugehen. Natürlich wäre es um einiges besser gewesen, wenn man sein Team auch mal persönlich hätte sehen können und sich wie in einem Büro hätte zusammensetzen können, um gemeinsam an einem PC an dem Spiel zu tüfteln. Besonders der Test-Aspekt wurde so natürlich leider auf diejenigen beschränkt, die eine Brille hatten und dazu dauerhaft in Kempten. Ich hatte zwar die Chance, das fertige Spiel mit der Brille zu spielen, allerdings war es für mich während der Entwicklung schwierig, Logik zu implementieren und diese dann auch sofort zu testen. Irgendwie haben wir es dann doch hinbekommen, was ein Testament für den Willen ist, dieses Spiel zu realisieren.

Dass uns dieses Fach als erster Jahrgang so zur Verfügung gestellt wurde, ist ein sehr positive Entwicklung. Von der ersten Idee bis zum spielbaren VR-Game zu kommen ist eine sehr wertvolle Erfahrung, die besonders im Kontext des anstehenden Praxissemester zusätzlich an Bedeutung gewinnt. Zusammenfassend bin ich sehr zufrieden, wie es gelaufen ist.

7 QUELLENVERZEICHNIS

7.1 AUDIOVERZEICHNIS

- 1) Unbekannter Urheber, (24.06.2020) „酒店电梯门打开关闭音效“. ChinaZ, (<https://sc.chinaz.com/yinxiao/200624304941.htm>)

Unbekannter Urheber, (06.11.2020) „走路脚步声音效“. ChinaZ, (<https://sc.chinaz.com/yinxiao/201106434910.htm>)

Krissyeliot, (30.07.2011) „glass-door“. freesound, (<https://freesound.org/people/krissyeliot/sounds/125529/>)

时光倒影, (03.04.2016) „拨弄门锁把“. earo, (<https://www.earo.com/sound/show/soundid-13746>)

Unbekannter Urheber, (25.10.2020) „门锁打开的声音“. ChinaZ, (<https://sc.chinaz.com/yinxiao/201025096461.htm>)

Unbekannter Urheber, (17.09.2020) „开门的音效在线试听“. ChinaZ, (<https://sc.chinaz.com/yinxiao/200917555211.htm>)

Unbekannter Urheber, (11.05.2015) „俩娃娃笑声音效“. ChinaZ, (<https://sc.chinaz.com/yinxiao/150511316241.htm>)

ggctuk, (24.09.2009) „exp_obj_largeo3“. freesound, (<https://freesound.org/people/ggctuk/sounds/80500/>)

Horemheb, (23.05.2020) „游戏拾取物品音效“. earo, (<https://www.earo.com/sound/show/soundid-20118>)

Unbekannter Urheber, (19.08.2020) „平稳有力的心跳声“. ChinaZ, (<https://sc.chinaz.com/yinxiao/200819295751.htm>)

Unbekannter Urheber, (02.09.2020) „房子开灯音效“. ChinaZ, (<https://sc.chinaz.com/yinxiao/200902446231.htm>)

Unbekannter Urheber, (30.09.2020) „木地板滑动木椅音效“. ChinaZ, (<https://sc.chinaz.com/yinxiao/200930012031.htm>)

小卡比兽, (22.12.2018) „科幻机械引擎声音“. earo, (<https://www.earo.com/sound/show/soundid-17982>)

Unbekannter Urheber, (08.07.2020) „机床持续运作音效“. ChinaZ, (<https://sc.chinaz.com/yinxiao/200708207020.htm>)

Unbekannter Urheber, (19.11.2020) „玻璃碎裂动画音效“. ChinaZ, (<https://sc.chinaz.com/yinxiao/201119069372.htm>)

坚果, (10.06.2014) „低保真恐怖尖叫“. earo, (<https://www.earo.com/sound/show/soundid-11038>)

Unbekannter Urheber, (21.10.2020) „摔跤倒地音效“. ChinaZ, (<https://sc.chinaz.com/yinxiao/201021425650.htm>)

Eideann, (29.12.2017) „恐怖氛围噪声“. earo, (<https://www.earo.com/sound/show/soundid-16334>)

- Lughaidh, (09.06.2017) „怪异持续音氛围“.
earo, (<https://www.earo.com/sound/show/soundid-15501>)
- 梦伴, (04.11.2015) „恐怖低音持续声景“.
earo, (<https://www.earo.com/sound/show/soundid-13127>)
- Space_Radio, (16.04.2015) „horror.mp3“.
freesound, (https://freesound.org/people/Space_Radio/sounds/270658/)
- 殇蓝, (09.11.2014) „灰暗恐怖环境音“.
earo, (<https://www.earo.com/sound/show/soundid-11660>)
- R_mac, (01.07.2016) „Horror Film Scores“.
freesound, (https://freesound.org/people/R_mac/sounds/348858/)
- LeoX, (10.08.2015) „电子舞蹈节拍循环音乐“.
earo, (<https://www.earo.com/sound/show/soundid-12774>)

7.2 VIDEOVERZEICHNIS

- 2) Sir_Fansi Gamedev, (27.04.2020) „How to make VR door and opening it”.
YouTube, (https://www.youtube.com/watch?v=_HULVPAOj6Y)
DownToCode, (05.05.2020) „Make a simple VR Door in Unreal Engine”.
YouTube, (<https://www.youtube.com/watch?v=j4MJ46jTwWg>)
Rune Berg, (22.08.2018) „Knuckles EV2 field test and implementing an open/close door in VR”.
YouTube, (<https://www.youtube.com/watch?v=h4gChzavPpY>)
Abdulkader El Rawas, (14.12.2019) „Unreal Engine 4 Beginner Tutorial Simple Open/Close Door using E key Blueprint in 5 minutes”.
YouTube, (<https://www.youtube.com/watch?v=OfJ4wK784kY>)
- 3) Dean Ashford, (08.07.2018) „UE4 - Tutorial Change - Textures at Run Time”.
YouTube, (<https://www.youtube.com/watch?v=FY5P8-9MYfg>)
- 4) TorQueMoD, (09.02.2017) „UE4 How to easily communicate between blueprints - Blueprint Interface Tutorial”.
YouTube, (<https://www.youtube.com/watch?v=9RmOaStWfNY>)
- 5) Fatty Bull, (02.10.2020) „Unreal Engine Game VR Push Button to trigger an Event - Loading different Level”.
YouTube, (<https://www.youtube.com/watch?v=NN682AGC1Yg>)
- 6) Dean Ashford, (15.02.2017) „UE4 - Tutorial - Confetti!”.
YouTube, (<https://www.youtube.com/watch?v=F229gVefDec>)
- 7) Darrin Lile, (24.09.2014) „Blender Character UV Mapping”.
YouTube, (<https://www.youtube.com/watch?v=W-ZmDKuB6HI&list=PLyelxoTsmSpdQwcx1OloZuW48bOUJQo2s>)

7.3 ABBILDUNGSVERZEICHNIS

- 8) Unbekannter Urheber, (l.A. 08.02.2021). PNGKIT.
https://www.pngkit.com/png/full/93-938660_paper-wrinkled-lines-to-write-signs-note-paper.png

8 ZUSÄTZLICHE DOKUMENTE

Dokument

Verfasser*innen

GDD_Nightmare_on_Bahnhofsstreet.pdf Feier Jiang, Syafiqah Husna Md Sazali

9 ERKLÄRUNGEN

Die Unterzeichnung der Selbstständigkeitserklärung ist obligatorisch. Die Unterzeichnung der Ermächtigung ist optional.

9.1 SELBSTSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, dass ich die studentische Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe.

Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher nicht veröffentlicht.

Kempten, 08.02.2021



Ort, Datum

Unterschrift Dan Eisenkrämer

Kempten, 09.02.2021



Ort, Datum

Unterschrift Syafiqah Husna MD Sazali

Kempten, 09.02.2021



Ort, Datum

Unterschrift Feier Jiang

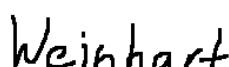


Kempten, 09.02.2021

Ort, Datum

Unterschrift Simon Kompaß

Kempten, 08.02.2021



Ort, Datum

Unterschrift Niklas Weinhart



Kempten, 08.02.2021

Ort, Datum

Unterschrift Marc von Glahn

9.2 ERMÄCHTIGUNG

Die Urheberin/Der Urheber der studentischen Arbeit kann (muss nicht) erklären, dass die Hochschule Kempten folgende Nutzungsrechte erhält.



Hiermit ermächtige ich/wir die Hochschule Kempten zur Veröffentlichung einer Kurzzusammenfassung sowie Bilder/Screenshots und ggf. angefertigte Videos meiner studentischen Arbeit z. B. auf gedruckten Medien oder auf einer Internetseite der Hochschule Kempten zwecks Bewerbung des Bachelorstudiengangs „Game Engineering“ und des Masterstudiengangs „Game Engineering und Visual Computing“.

Dies betrifft insbesondere den Webauftritt der Hochschule Kempten inklusive der Webseite des Zentrums für Computerspiele und Simulation. Die Hochschule Kempten erhält das einfache, unentgeltliche Nutzungsrecht im Sinne der §§ 31 Abs. 2, 32 Abs. 3 Satz 3 Urheberrechtsgesetz (UrhG).



Kempten, 08.02.2021

Ort, Datum

Unterschrift Dan Eisenkrämer



Kempten, 09.02.2021

Ort, Datum

Unterschrift Syafiqah Husna MD Sazali



Kempten, 09.02.2021

Ort, Datum

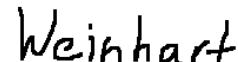
Unterschrift Feier Jiang



Kempten, 09.02.2021

Ort, Datum

Unterschrift Simon Kompaß



Kempten, 08.02.2021

Ort, Datum

Unterschrift Niklas Weinhart



Kempten, 08.02.2021

Ort, Datum

Unterschrift Marc von Glahn