Untersuchung eines virtuellen 3D-Modells des Campus Finkenau mit Laserscan und Fotogrammetrie

auf Realisierbarkeiten und die Visualisierungen von Lichtshows und Videomapping

Bachelor-Thesis

zur Erlangung des akademischen Grades B.Sc.

Mirco Hülsemann

[redacted]

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Design, Medien und Information
Department Medientechnik

Erstprüfer: Prof. Dr.-Ing. Roland Greule

Zweitprüfer: Prof. Dr.-Ing. Sabine Schumann

Hamburg, 2. August 2019

Inhaltsverzeichnis

Abstract

This bachelor thesis analyses the possibility to simulate light shows and video mapping in a game engine. All 3D models are created based on data of the Finkenau Campus, collected with photogrammetry and laser scanning. The simulation is as physically accurate as possible. The finished scene is played in real time on a desktop computer and in virtual reality.

The simulation is done in the Unreal Engine 4 and implements the most common elements in light shows and video mappings. If several possible implementations are available, all are presented, and the advantages and disadvantages are discussed. In a few cases, to create the simulation, physical accuracy is compromised.

Zusammenfassung

Diese Bachelorarbeit untersucht die Möglichkeit, Lichtshows und Videomapping mit einem 3D-Modell in einer Game-Engine zu simulieren. Die 3D-Objekte werden dabei mit Daten aus der Fotogrammetrie und Laserscanning vom Campus Finkenau erstellt. Die Simulation ist physikalisch so akkurat wie möglich. Als Endprodukt wird die Szene in Echtzeit auf Desktop PCs und in der Virtuellen Realität dargestellt.

Die Simulation erfolgt in der Unreal Engine 4 und setzt erfolgreich die wichtigsten Lichtshowelemente und Videomappings um. Wenn mehrere Methoden zur Umsetzung vorhanden sind, werden diese beschrieben und die Vor- und Nachteile aufgetragen. Bei manchen Methoden müssen mangels Umsetzbarkeit in der Engine allerdings Abstriche in der physikalisch akkuraten Darstellung gemacht werden.

1 Einleitung

Moderne Game-Engines ermöglichen neue immersive Darstellungsmöglichkeiten für Spiele und Simulationen in der virtuellen Realität (VR). In Kombination mit den detailgetreuen Möglichkeiten, die Fotogrammetrie und Laserscanning bieten, entstehen realitätsnahe Erfahrungen in der VR.

Bereits seit Anfang des 20. Jahrhunderts gibt es Fotogrammetrie. Seit den 1980er Jahren wird diese teilweise von Laserscanning ersetzt.

Abgesehen von den klassischen Anwendungsbereichen ermöglichen diese Messsysteme das Festhalten von temporären Installationen und einmaligen Situationen. Diese können so für die Ewigkeit digital festgehalten werden.

VR als Anwendungsbereich ist noch verhältnismäßig jung. Die Verwendung von 3D-Szenen aus Laserscannern und Fotogrammetrie in der Augmented Reality (AR) und VR bietet viele Möglichkeiten. Es können Produktionsabläufe, Arbeitsplätze und Notfallsituationen simuliert werden, ohne die realen Räume und Objekte zu benötigen. So kann zum Beispiel ein Einsatz auf einem Windrad in der sicheren Umgebung eines Wohnzimmers geübt werden.

Kombiniert man beide Einsatzbereiche, ermöglicht dies einmalige Situationen so realistisch wie möglich wieder zu erleben. Doch nicht nur vergangene Geschehnisse können so veranschaulicht werden, sondern auch Simulationen von Konzepten oder zukünftigen Veranstaltungen.

Besonders die Inhalte beim Videomapping müssen weit im Voraus geplant und erstellt werden. Eine Möglichkeit, die Show zu testen, gibt es meistens nicht, da die Projektionsflächen nicht frei zugänglich, die Projektoren teuer in der Miete und kompliziert aufzubauen sind. Bei Lichtshows hat der Designer noch vor Ort die Möglichkeit, seine Inszenierung an die Situation anzupassen. Einmal positionierte Scheinwerfer lassen sich aber nicht mehr so schnell umbauen.

Lassen sich diese Shows vorweg an einem Modell simulieren, können noch rechtzeitig Anpassungen vorgenommen werden. Je detaillierter dabei das 3D-Modell ist, desto genauer kann auch simuliert und geplant werden. Es gibt bereits viele Programme, die eine Lichtsimulation umsetzen können. Bei vielen würde ein hohe Anzahl von Polygonen aber schnell problematisch werden. Eine Lösung bieten die Game-Engines wie Unity und Unreal, die aber bisher kaum Anwendung gefunden haben.

Im Rahmen dieser Arbeit wird der Innenhof des Kunst- und Mediencampus Hamburg mit Laserscan und Fotogrammetrie vermessen. Aus den gewonnenen Daten wird

ein texturiertes 3D-Modell erstellt. Das Ziel ist es, mit diesem Modell Videomapping und Lichtshowelemente zur Veranschaulichung in Echtzeit auf einem Desktop PC und in VR zu simulieren. Dafür wird die Unreal Engine 4 verwendet, da sie diverse Funktionen zur realitätsnahen Lichtsimulation anbietet.

Diese Arbeit beginnt mit den Grundlagen zur Fotogrammetrie (Kapitel 2) und Laserscanning (Kapitel 3). Beim Laserscanning wird dabei auf die unterschiedlichen Messprinzipien und Gerätetypen eingegangen. Auch der in diesem Projekt verwendete Scanner FARO Focus^S 350 wird kurz vorgestellt (Kapitel 3.4). Im Anschluss folgt die Beschreibung des Scanobjekts – dem Campus Finkenau – und die Durchführung der Datenerfassung mit terrestrischem Laserscanning und Fotogrammetrie (Kapitel 4).

Die Entwicklung des 3D-Modells mit FARO Scene, Adobe Bridge, RealityCapture, ZBrush, Maya und Unreal wird in dem Kapitel „Datenauswertung“ zusammengefasst (Kapitel 5). Die Meshes und Texturen der fertigen 3D-Objekte werden kurz analysiert (Kapitel 5.5). Anhand der 3D-Szene wird die Licht- und Videomapping-Simulation in Unreal schrittweise erläutert (Kapitel 6). Umgesetzt werden Videomapping, Spotlights mit Gobos, Washlights und Beamlights. Dabei wird auf die Verwendung von Lichtstärkeverteilungskurven und die Umsetzung von Farbmischungen, Animationen und der Umgebungsbeleuchtung eingegangen.

Abschließend erfolgt eine Zusammenfassung der Ergebnisse im „Fazit“(Kapitel 7).

2 Fotogrammetrie

Fotogrammetrie (auch Photogrammetrie) bezeichnet das Verfahren, die Lage und Dreidimensionalität eines Objektes im Raum aus zweidimensionalen Bildern oder Videos zu bestimmen. Diese Methode zählt zu den passiven Messverfahren, da das von den Objekten reflektierte Licht fotografisch festgehalten und berührungslos vermessen wird. Die Anwendungsbereiche werden häufig in Luftbildfotogrammetrie und terrestrische- bzw. Nahbereichsfotogrammetrie unterteilt.

Bei der Luftbildmessung werden Fotografien mit Messbildkameras aus Flugzeugen heraus erstellt. Die Aufnahmen bilden ein geordnetes, sich stark überlappendes Raster. Endprodukt der Luftbildfotogrammetrie sind meistens topografische Karten.

In der Nahbereichsfotogrammetrie können dahingegen beliebige Aufnahmepositionen gewählt werden. Dafür werden meist handelsübliche, hochauflösende Digitalkameras verwendet. Die Genauigkeit der Messungen liegt zwischen 0,1 mm bei Dimensionen von etwa 1 m, und im cm-Bereich bei ca. 200 m großen Objekten. Die erreichbare Genauigkeit hängt dabei nicht nur von der Bildqualität, sondern auch von der Wertigkeit der Kamera ab (innere Orientierung). Aus der Nahbereichsfotogrammetrie werden Messwerte, Bestandsaufnahmen oder 3D-Modelle gewonnen.[1]

	Sensoren	Aufnahmeentfernung	Genauigkeit im Objektraum (Lage)
Satellit Raumschiff	optoelektronischer Scanner Messkamera	200 km bis 400 km	±2 m bis ±20 m
Flugzeug	Messkamera optoelektronischer Scanner	0,3 km bis 10 km	±3 cm bis ±50 cm
Stativ Hand	terrestrische Messkamera digitale Kamera Amateurkamera	0,1 m bis 200 m	±0,01 mm bis ±100 mm

Tabelle 2.1: Sensoren, Aufnahmebereiche und Genauigkeitspotential der Fotogrammetrie (nach Spektrum Akademischer Verlag, 2000)

[1] Luhman, 2018.

2.1 Funktionsprinzip

Das Messverfahren basiert auf dem mathematischen Modell der zentralprojektiven Abbildung. Die untenstehende Abbildung 2.1 stellt das Prinzip beispielhaft dar.

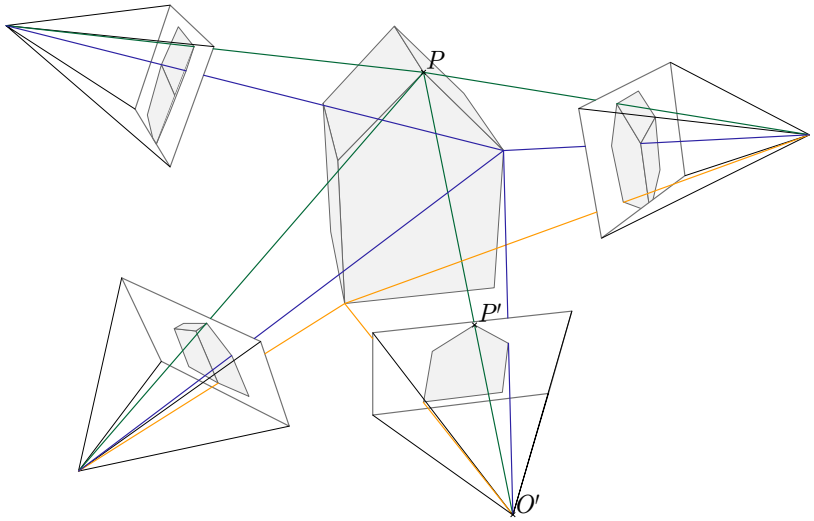


Abbildung 2.1: Messprinzip der Fotogrammetrie (in Anlehnung an Mason, 2015)

Jeder Bildpunkt P' legt eine Raumrichtung vom Projektionszentrum O' zum Objektpunkt P dar. Die Länge des Richtungsstrahls ist dabei anfangs noch nicht bekannt. Zur dreidimensionalen Bestimmung sind mindestens zwei sich teilweise überlappende Aufnahmen von unterschiedlichen Positionen nötig. So ergibt sich, durch den Schnitt von mindestens zwei Bildstrahlen, die Position des Objektpunktes im Raum.

Sowohl die Dimensionen innerhalb der Kamera (innere Orientierung) und die Lage der Kamera im Raum (äußere Orientierung) müssen zur erfolgreichen Berechnung bekannt sein. Zur inneren Orientierung gehören diverse Größen, die für Fotogrammetrie benötigt werden. Außerdem beschreibt sie Werte für die Korrektur von Abbildungsfehlern und den Abweichungen vom idealen System. Die innere Orientierung kann, im Gegensatz zur äußeren Orientierung, nicht bei der Auswertung berechnet, sondern muss extra gemessen werden. Dafür können die Herstellerangaben verwendet oder, für genauere Werte, eigene Messungen im Labor oder vor Ort vorgenommen werden.

Für die äußere Orientierung wird die räumliche Lage der Kamera durch Bemessung bekannter Objektpunkte berechnet. Vor allem in der Geodäsie (Ausmessung und Abbildung der Erdoberfläche) kommen dafür auch GPS (Global Positioning System) und INS (Trägheitsnavigationssystem, engl. Inertial Navigation System) zum Einsatz. Jedes Bild wird mit je drei Translations- und Rotationswerten in dem übergeordneten Koordinatensystem eingeordnet.

Objektbereiche, die in den zweidimensionalen Bildern nicht zu sehen sind, können auch nicht durch die Fotogrammetrie rekonstruiert werden. Dazu zählen nicht nur verdeckte, sondern auch kontrastarme Bereiche und Elemente, die zu klein sind.[2]

[2]Luhman, 2018.

3 Laserscanning

Zum Vermessen eines Raumes oder Objektes per Laserscanning werden diese mit Laserstrahlen beleuchtet. Aus der reflektierten Strahlung wird die Distanz gemessen. Häufig werden dabei auch die Intensitätswerte der Oberfläche berechnet. Dies ermöglicht eine Darstellung, die ähnlich der eines Schwarzweißfotos ist. Das Laserscanning ergänzt oder ersetzt in vielen Anwendungen die Fotogrammetrie.

Laserscanning gehört zu den aktiven Messverfahren, da der Laserscanner die Flächen aktiv beleuchtet. Das hat den Vorteil, dass unabhängig vom Umgebungslicht auch nicht beleuchtete Flächen gemessen werden können. Problematisch sind allerdings, je nach Messverfahren, sowohl spiegelnde als auch stark absorbierende oder streuende Oberflächen.[1]

3.1 Baugruppen

Die verschiedenen Laserscanner-Gerätetypen (siehe Kapitel 3.3) basieren alle auf denselben Prinzipien. Der Aufbau wird hier beispielhaft anhand eines terrestrischen Laserscanners erklärt.

Die Hauptbauteile eines terrestrischen Laserscanners sind das elektronische Distanzmessgerät (EDM), je ein vertikales und ein horizontales Ablenkungssystem, ein integrierter Rechner und eine Farbkamera (siehe Abb. 3.1).

Das EDM funktioniert wie ein handelsübliches Laserentfernungsmessgerät. Jeder Messpunkt wird durch einen Laserstrahl beleuchtet. Das vom Objekt reflektierte Licht wird in dem Laserkopf des Scanners gemessen und in ein elektrisches Signal umgewandelt. Das Licht muss dabei eine Filterscheibe im Laserkopf durchdringen, mit der das Umgebungslicht blockiert wird.

Das vertikale Ablenksystem sorgt dafür, dass der Laserstrahl in vertikaler Richtung auf den gewünschten Messpunkt abgelenkt wird. Das System besteht aus einem Drehspiegel, der in einem Winkel von 45° angebracht ist, und einem Motor, der diesen dreht.

Für die horizontale Ablenkung steht ein zweiter Motor zur Verfügung, der den gesamten Scanner um die eigene Achse dreht. Lediglich der Gerätefuß ist fest auf einem Stativ montiert.

Die Messwerte des EDM beinhalten keine Farbinformationen. Deshalb nimmt im Anschluss eine integrierte Farbkamera mehrere Farbfotos der Umgebung auf.

[1] Die Ausführungen in diesem Kapiteln beruhen weitgehend auf Mettenleiter u. a., 2015

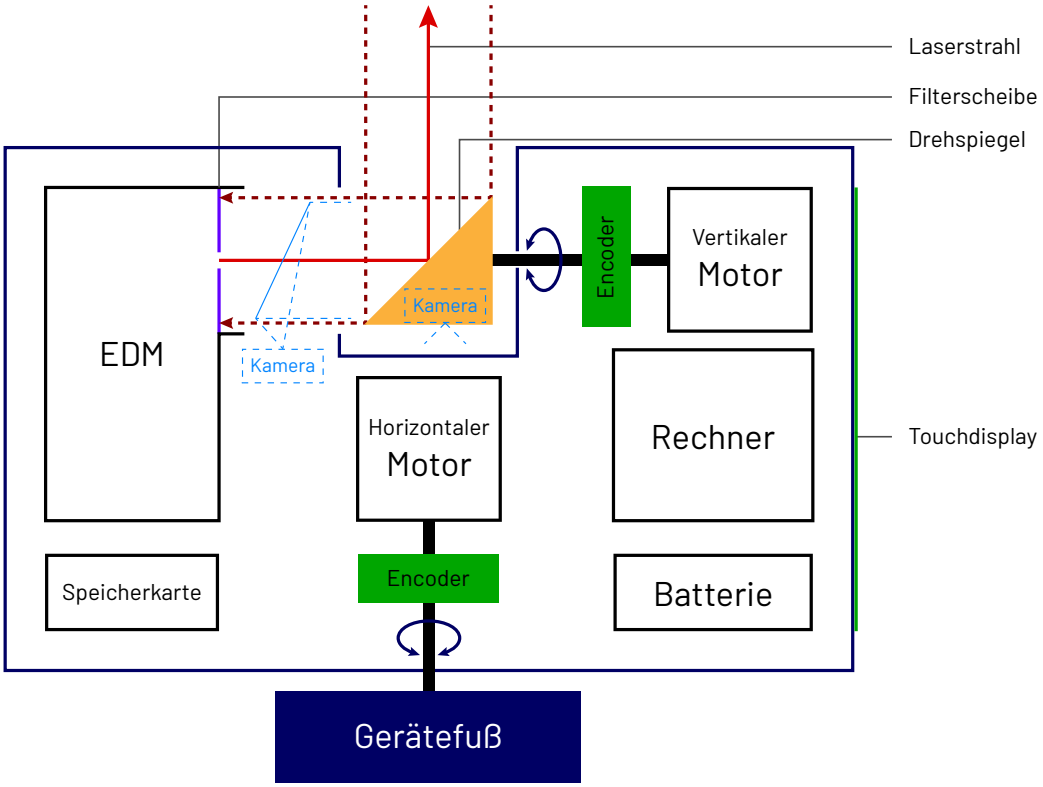


Abbildung 3.1: Funktionsprinzip eines terrestrischen Laserscanners (in Anlehnung an Mettenleiter u. a., 2015, 14ff.)

Dabei entstehen etwa 40 bis 250 Einzelaufnahmen. Das dauert, je nach Lichtverhältnissen, drei bis zehn Minuten. Die Zuordnung der Farbpunkte zu den Messpunkten erfolgt automatisch im Rechner. Die Abweichungen liegen dabei bei unter einem Pixel. Die Farbkamera wird entweder mit einem Spiegel in den Strahlengang des Lasers geschaltet oder sie ist direkt mit im Rotor des Drehspiegels verbaut.

Die Winkelmessung der Rotoren und die Messwerte des Messpunktes werden bereits im EDM zusammengefasst und anschließend an den integrierten Rechner übertragen. Dieser verarbeitet die Daten und speichert sie auf einer Speicherkarte. Der Rechner steuert zudem die Motoren, die integrierte Farbkamera und vermisst diverse Parameter, wie Gerätetemperatur, Neigung und Ladezustand der Batterie. Über ein Touchdisplay kann mit dem Gerät interagiert und die Einstellungen verändert werden. Das Display visualisiert zudem die gemessenen Daten und gibt Auskunft über den Gerätestatus. Der Rechner kann auch über WLAN oder Ethernet mit einem externen PC verbunden werden.

3.2 Messprinzipien

Für die Distanzmessungen stehen mehrere Messverfahren zur Verfügung. Die drei verbreitetsten werden im Folgenden kurz erläutert.

Verfahren	Messbereich	Bemerkung
Laufzeitmessung	1 m - etliche km	schnell, neue Maßstäbe durch Multi-Waveform-Methode
Triangulation	1 µm - 30 m	stark abhängig von der Oberfläche, günstig, robust für kleinere Entfernungen
Phasendifferenz-verfahren	max. 300 m	Kosten überschaubar für hohe Genauigkeiten, sehr schnell

Tabelle 3.1: Auswahlkriterien für das Messprinzip (nach Mettenleiter u. a., 2015)

3.2.1 Impulslaufzeitverfahren

Zum Messen der Distanz zum Objekt wird beim Impulslaufzeitverfahren ein kurzer Lichtpuls ausgesendet und das reflektierte Signal empfangen. Die gemessene Zeitdifferenz ist proportional zur Streckenlänge. Die resultierende Distanz zum Objekt berechnet sich aus der Laufzeit Δt und der Lichtgeschwindigkeit c.

$$d = \frac{c \cdot \Delta t}{2}$$

Weil der Laserstrahl mit Lichtgeschwindigkeit reflektiert wird, muss das Messgerät im Bereich von 10^{-11} s zählen können, um Messgenauigkeiten im Millimeterbereich zu erreichen.[2] Dies umzusetzen ist technisch sehr aufwendig.

Multi-Waveform-Verfahren

Viele Laserscanner mit dem Prinzip des Impulslaufzeitverfahrens verwenden das Multi-Waveform-Verfahren. Dabei wird die Intensität des gesamten reflektierten Signals

[2] Möllring, 2018a.

aufgenommen. Je nach Bedarf kann der erste, letzte oder intensivste Messwert zur Auswertung verwendet werden. So können Störobjekte, wie zum Beispiel Blätter eines Baumes, ausgefiltert werden.

Ein Vorteil des Impulslaufzeitverfahrens ist die stetige Eindeutigkeit über weite Distanzen. Im Vergleich zu anderen Verfahren ist die Messung aber ungenau.

3.2.2 Triangulationsverfahren

Ein Triangulationssensor beleuchtet das Objekt mit einem Laser, das reflektierte Signal wird von einem Lichtsensor gemessen. Aus der Position des Lichtflecks auf dem Sensor kann der Winkel α bestimmt werden. Da die inneren Größen, wie der Abstand zwischen Laser und Sensor (Basislänge) und der Austrittswinkel des Lasers, bekannt sind, lässt sich damit die Entfernung zum gemessenen Objekt berechnen.

Das Triangulationsverfahren bei Lasersystemen ist, mit hoher Genauigkeit, geeignet für kurze Distanzen von 1 µm bis 30 m.

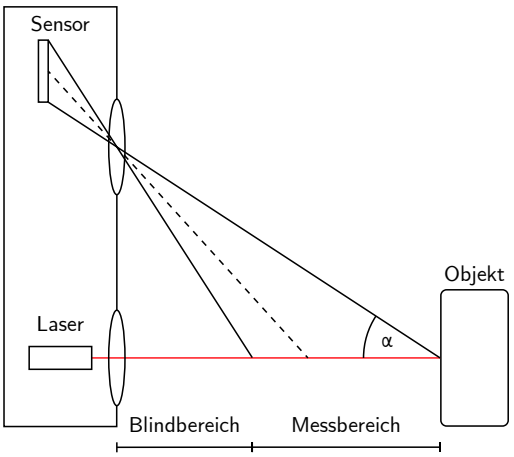


Abbildung 3.2: Triangulationsverfahren (eigene Darstellung)

3.2.3 Phasendifferenzverfahren

Beim Phasendifferenzverfahren (auch Phasenvergleichsverfahren) wird ein amplitudenmoduliertes Laserlicht kontinuierlich ausgesendet. Das vom Objekt reflektierte Signal wird auf seine Phasenlage gemessen. Die Phasendifferenz zwischen dem gesendeten und empfangenen Signal ist proportional zur Entfernung des Messobjektes.

Die Strecke wird durch die Messgröße der Phasendifferenz $\Delta\varphi$ und der Anzahl der ganzen Wellenlängen a zwischen Sender und Empfänger berechnet.[3]

$$2d = a \cdot \lambda + \frac{\Delta\varphi}{2\pi} \cdot \lambda$$

Die Wellenlänge ergibt sich aus der Lichtgeschwindigkeit und der Modulationsfrequenz.

$$\lambda = \frac{c}{f}$$

Um die Anzahl der ganzen Wellen (a) zu ermitteln, wird der Messvorgang mit verschiedenen Wellenlängen wiederholt. Zuerst werden die Feinheiten mit einer kurzen Wellenlänge gemessen. Anschließend wird mit immer längeren Wellenlängen, die mit

[3] Möllring, 2018b.

immer kleineren Frequenzen moduliert sind, die gesamte Länge ermittelt. Aus der Kombination dieser Messwerte ergibt sich die Streckendistanz. Tabelle 3.2 zeigt eine beispielhafte Messreihe. Die Gesamtlänge entsteht, indem für jede Stelle immer der jeweils genauste Messwert verwendet wird.

Messung	Messfrequenz	Wellenlänge	Reststrecke
1	10,0 MHz	20 m	8,437
2	1,0 MHz	200 m	88,2
3	0,1 MHz	2000 m	789
Gesamtlänge			788,437

Tabelle 3.2: Messbeispiel mit Phasendifferenzverfahren (nach Przybilla, 2019)

Das Phasendifferenzverfahren ist sehr schnell und präzise bei Messbereichen von bis zu 300 m.

3.3 Gerätetypen

Laserscanner werden im Allgemeinen in drei, beziehungsweise vier Gerätetypen eingeteilt. Der Aufbau und die Funktionsweise der unterschiedlichen Typen ähnelt sich weitestgehend.

3.3.1 Airborne-Laserscanning-Systeme

Airborne-Laserscanning-Systeme, kurz ALS, sind Laserscanner zum Erfassen der Erdoberfläche. Die Geräte werden in Hubschraubern oder Flugzeugen eingesetzt und ersetzen die Methode der Fotogrammetrie (siehe Kapitel 2).

Im Vergleich zu anderen Gerätetypen brauchen ALS nur einen kleinen Winkelbereich. Dieser ist nach unten, in Richtung Erdoberfläche ausgerichtet – typischerweise zwischen 40° bis 70° statt den sonst üblichen 360°. Verwendet wird ausschließlich das Impulslaufzeitverfahren, meistens mit dem Multi-Waveform-Verfahren (siehe Kapitel 3.2). Grund sind die benötigte große Reichweite, wegen der typischen Flughöhe von bis zu 4 km, und die geringe benötigte Distanzauflösung von einigen Zentimetern. Außerdem ist es vorteilhaft Störobjekte, wie Äste von Bäumen, ausfiltern zu können.

ALS sind mit Navigationseinheiten ausgestattet, welche die Position, Geschwindigkeit und Winkelbewegungen des Flugzeugs genauestens messen. So kann die räumliche Lage der Scans eindeutig zugewiesen werden.

3.3.2 Mobile-Mapping-Systeme

Mobile-Mapping-Systeme, kurz MMS, funktionieren ähnlich wie ALS. Allerdings haben sie ein Blickfeld von 360° auf einer Achse. MMS werden auf bewegten Plattformen, wie Autos oder Schienenfahrzeugen, eingesetzt. Der Laserscanner hat nur eine vertikale Strahlenablenkung und nimmt so immer nur einzelne „Scheiben“ oder „Profile“ auf. Durch die Vorwärtsbewegung des Fahrzeugs werden die Profile zu einem Helix auseinandergezogen.

Die Abstände Δx zwischen den einzelnen Profilen hängen dabei von der Fahrtgeschwindigkeit v und der Drehzahl des Scanners ω ab. Bei Autos im Straßenverkehr sind damit Abstände von bis zu 11 cm nicht ungewöhnlich.

$$\Delta x = \frac{v}{3,6 \cdot \omega} \quad [\mathrm{m}]$$

MMS besitzen, genau wie ALS, Navigationseinheiten zur räumlichen Bestimmung des Fahrzeugs.

3.3.3 Handgeführte 3D-Laserscanner

Handgeführte 3D-Laserscanner, kurz HLS, sind kleine, handgeführte Varianten der Laserscanner. Sie bieten eine flexiblere Handhabung für Scans im Bereich von bis zu drei Metern dar. Zum Verwenden ist ein angeschlossener Laptop oder ein Tablet notwendig. Das Objekt wird in Echtzeit auf dem angeschlossenen Gerät angezeigt. HLS werden häufig zu den terrestrischen Laserscannern gezählt.

3.3.4 Terrestrische Laserscanner

Laserscanner, die an festen Standpunkten – meist auf Stativen – operieren, werden als terrestrische Laserscanner, kurz TLS, bezeichnet. Anders als ALS, MMS und HLS wird der Laser vertikal und horizontal abgelenkt, wodurch die Umgebung sphärisch erfasst werden kann.

3.4 FARO Focus^S 350

„Der FARO Focus^S 350 wurde speziell für Outdoor-Anwendungen entwickelt, dank seinen kleinen Abmessungen, seinem geringen Gewicht und seiner erweiterten Reichweite.“ FARO, 2019

Die Laserscanner der Focus^S Serie von FARO sind terrestrische Scansysteme, die auf dem Phasendifferenzverfahren basieren. Die Modelle sind ca. $23 \times 18,3 \times 10$ cm (Länge, Breite, Höhe) groß und 4,2 kg schwer. Sie besitzen neben der Scaneinheit

eine Kamera mit einem hohen dynamischen Umfang (HDR) und bis zu 165 Megapixeln Farbauflösung. Integriert sind ein Zweiachskompensator, Höhensensor, Kompass, GPS und GLONASS (Globales Satellitennavigationssystem).[4]

Das Modell Focus^S 350 hat mit 0,6 m bis 350 m die längste Reichweite der Serie. Die in der Praxis erreichbaren Entfernungen hängt allerdings von der Oberfläche des Objektes ab. Tabelle 3.3 listet die erreichbaren Reichweiten abhängig von dem Reflexionsgrad der Oberfläche auf.

Reflexionsgrad der Oberfläche	90% (weiß)	10% (dunkel-grau)	2% (schwarz)
Reichweite	0,6 - 350 m	0,6 - 150 m	0,6 - 50 m

Tabelle 3.3: Angaben zur Reichweite bei Lambertscher Streuung (Quelle: FARO, 2016)

Das Distanzrauschen ist ebenfalls Abhängig von dem Reflexionsgrad der Oberfläche. Je nach eingestellter Scanqualität, Reflexionsgrad und Entfernung der Oberfläche reicht das Distanzrauschen 0,3 mm bis 2 mm.
Der Systematische Distanzfehler bei 10 m bis 25 m liegt bei ±1 mm und die Winkelgenauigkeit beträgt 19″ (Bogensekunden) für vertikale und horizontale Winkel.
Die resultierende 3D-Positionsungenauigkeit beträgt 2 mm bei 10 m Entfernung, 3,5 mm bei 25 m und 36 mm bei 350 m.[5]

Abbildung 3.3: FARO Focus^S 350 (Quelle: FARO, 2016)

[4] FARO, 2016.
[5] FARO, 2016.

4 Campus Finkenau

Der Kunst- und Mediencampus Hamburg, der auf dem Gelände der ehemaligen Frauenklinik Finkenau liegt, besteht aus mehreren Ausbildungsinstitutionen des Medienbereiches. In den Gebäuden befinden sich die Fakultät Design, Medien und Information der Hochschule für Angewandte Wissenschaften Hamburg (HAW), Teile der Hochschule für bildende Künste Hamburg (HFBK), die Hamburg Media School (HMS), die Miami Ad School (MAS), das Multimedia Kontor Hamburg (MMKH) und der Hörfunk- und Fernseh-Stadtsender Tide (TIDE). Zum Campus gehören der Altbau und ein OP-Bunker[1], die früher die Frauenklinik beherbergten, der Neubau der HAW, das TIDE-Studio und das Gebäude der MAS im Innenbereich.

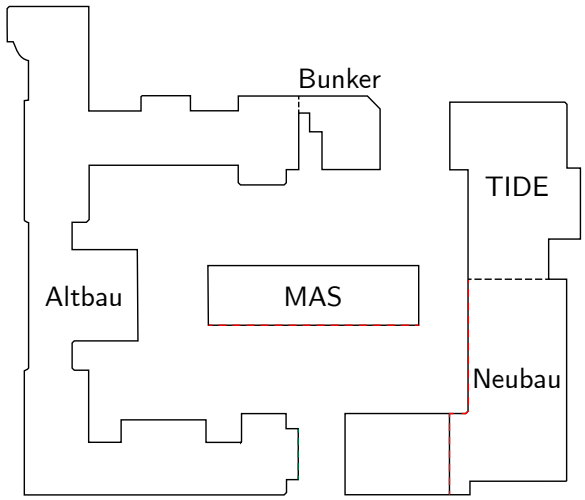


Abbildung 4.1: Grundriss des Campus Finkenau (eigene Darstellung)

Im Rahmen des Projektes dieser Bachelorarbeit wird hauptsächlich der Neubau und das MAS-Gebäude für die Lichtsimulation benötigt. Zum korrekten Positionieren der virtuellen Lichtquellen im 3D-Modell wird auch der dem Neubau am nächsten liegende Teil des Altbaus benötigt. Damit beim Betrachten in VR ein vollständiger Raumeindruck entsteht, wurde der Rest des Campus ebenfalls erfasst. Die weniger relevanten Teile wurden in der Datenverarbeitung mit etwas geringerer Qualität aufgearbeitet und teilweise komplett weggelassen.

[1] Ärztekammer Hamburg, 2015.

4.1 Datenerfassung

Der terrestrische Laserscan und die Aufnahme der Daten für die Fotogrammetrie wurden zeitgleich innerhalb eines Tages durchgeführt.

(a) FARO Focus^S 350 **(b)** Canon EOS 80D **(c)** DJI Osmo RAW

Abbildung 4.2: Aufnahmen mit dem Laserscanning-System und mit Digitalkameras für die Fotogrammetrie (eigene Darstellungen)

4.1.1 Terrestrisches Laserscanning

Für die Aufnahme der Scandaten wurde der Laserscanner FARO Focus^S 350 verwendet. Insgesamt wurden an 23 Standpunkten Scanaufnahmen gemacht. Die Standpunkte wurden vorher nicht festgelegt, sondern wurden spontan vor Ort gewählt. Zwei der ursprünglich in Erwägung gezogenen Standpunkte zur Aufnahme von oben konnten leider nicht realisiert werden, weil der Zugang zu den Balkonen verschlossen war. Stattdessen wurde aus Fenstern in der Nähe gescannt.

Die eingestellte Auflösung betrug 10330×4268 Punkte, also insgesamt 44 Millionen Punkte mit 4-facher „Qualität“. Das heißt, jeder Scanpunkt wurde viermal bemessen und anschließend der Mittelwert gebildet. Die Scandauer pro Standpunkt betrug ca. 15 Minuten, wobei die Erfassung der Punkte mit dem Laser ungefähr ein Drittel und die Aufnahme der HDR-Farbfotos den Rest der Zeit beanspruchte.

4.1.2 Fotogrammetrie

Die Fotos für die Fotogrammetrie wurden mit einer handelsüblichen Spiegelreflexkamera – der Canon EOS 80D – und einer Handheld-Kamera – der DJI Osmo RAW – aufgenommen. Aufnahmen mit einer Drohne waren nicht nötig, da hohe Standpunkte auch zu Fuß erreichbar waren und kein besonderer Anspruch auf die wirklichkeitsgetreue Rekonstruktion der Dachflächen liegt.

Die DSLR-Kamera verfügt über einen CMOS-Sensor mit einer effektiven Pixelzahl von ca. 24,2 Megapixeln. Zur lückenlosen Bestandsaufnahme wurden mehr als tausend Fotos aufgenommen. Allerdings stellte sich beim Sichten der Fotos heraus, dass auf einige Bildern der Kontrast zwischen Schatten- und Sonnenflächen zu stark war. Deswegen wurde ein paar Tage später, bei bewölktem Wetter, nochmal mehrere hundert Aufnahmen angefertigt.

Mit der DJI Osmo RAW Handheld-Kamera wurden zusätzlich 360°-Panorama Aufnahmen erstellt, um eine absolut lückenlose Datenerfassung zu garantieren. Dafür stand die, am Gimbal befestigte, Kamera auf einem Stativ. Auf Knopfdruck nahm die Kamera dann vollautomatisch in allen 360° mehrere Einzelbilder auf.

5 Datenauswertung

5.1 Entwicklung der Rohdaten

5.1.1 FARO Scene

Die Datensätze des FARO Focus^S können in der hauseigenen Software FARO Scene entwickelt werden. Auch andere Programme, wie zum Beispiel die freie und plattformunabhängige Software CloudCompare, können die Scandaten auswerten. Allerdings werden nur in FARO Scene die Farbwerte aus den Bildern der integrierten Fotokamera den dazugehörigen Scanpunkten zugeordnet.

Abbildung 5.1: Screenshot aus der FARO Scene Software (eigene Darstellung)

Die einzelnen Scansätze werden automatisiert zueinander ausgerichtet. Bei Bedarf kann der Benutzer die dazu verwendeten Parameter anpassen oder die Scans händisch positionieren. Zur Analyse der Passgenauigkeit zweier Scansätze sind Ungenauigkeitswerte angegeben. Diese Werte ergeben sich aus den Distanzen zwischen zwei identischen Punkten aus den jeweiligen Scans. Bei dem Laserscan vom Campus Finkenau liegt der Ungenauigkeiten-Mittelwert bei 8,21 mm. 35,7 % der Distanzen liegen unter 4 mm. Der kleinste Mittelwert liegt bei 2,32 mm und der höchste bei 29,22 mm.

Das sind im Vergleich zu den theoretisch möglichen Genauigkeiten große Distanzen. Dies kann viele Gründe haben. Eine mögliche Ursache könnte die große Glasfläche des Neubaus sein. In diesen Flächen werden die Entfernungen zu den gespiegelten Objekten gemessen. Diese tauchen aus verschiedenen Blickwinkeln unterschiedlich auf. Dadurch können die Messwerte nicht genau zugeordnet werden. Wegen der großen Glasfläche schlägt sich dieser Messfehler stark auf den Durchschnittsfehler aus. In diesem Fall sind die betroffenen Messpunkte uninteressant, da es nur Spiegelungen sind. Sie werden somit als Datenmüll aussortiert. In dem Screenshot aus Abbildung 5.1 ist rechts zu erkennen, wie sich das MAS-Gebäude in dem Neubau spiegelt. Der Laserscanner kann nicht wissen, dass diese Scanpunkte Spiegelungen sind. Eine andere Ursache ist vermutlich das kleinteilige Geländer der Terrasse, da besonders in diesem Bereich große Durchschnittsfehler vorkommen.

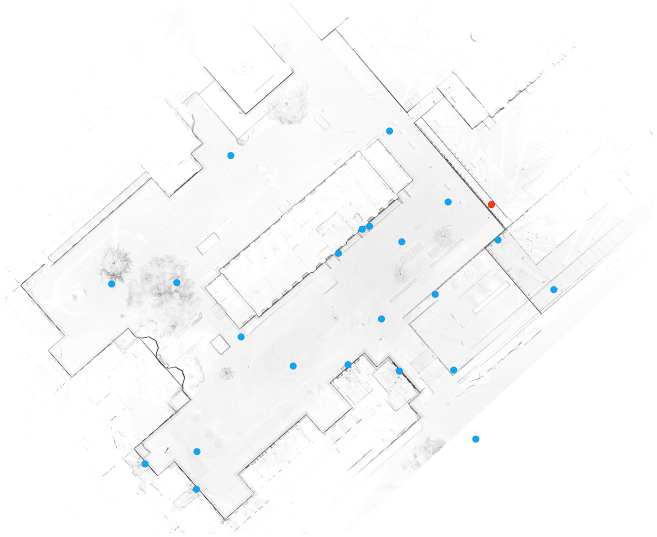


Abbildung 5.2: Aus den Laserscannerdaten generierter Grundriss mit Scanpositionen, der nicht registrierte Scan ist in Rot markiert (eigene Darstellung)

Abbildung 5.3: Blick aus dem Fenster vom nicht registriertem Scan (eigene Darstellung)

Einer der 23 Datensätze konnte nicht automatisch zugeordnet werden. Bei diesem Scansatz gibt es zu wenig Messpunkte, die mit den anderen Scans übereinstimmen. Der betroffene Scan wurde aus einem Fenster heraus aufgenommen, wodurch der Sichtbereich auf den Innenhof zu klein war (siehe Abbildung 5.3). Es wäre möglich, den Scan manuell zu positionieren. Da der zusätzliche Ausschnitt aber klein ist, wurde der Datensatz verworfen.

FARO Scene bietet diverse Funktionen zur Auswertung der Scandaten an. Die Punktewolke kann direkt in VR angezeigt werden und es können Entfernungen

gemessen und Grundrisse generiert werden (siehe Abbildung 5.2). In diesem Projekt wird die Punktewolke zur Weiterverarbeitung im ASTM-Industriestandard-Format E57 (.e57) exportiert.

5.1.2 Adobe Bridge

Die Rohdaten der Spiegelreflexkamera und der Handheld-Kamera werden zum Vorbereiten für die Fotogrammetrie als erstes in Adobe Bridge geladen. Durch das Fotografieren im Rohdatenformat-Format (RAW) stehen hier sämtliche Messwerte des Kamerasensors für die Entwicklung zur Verfügung.

Hauptaufgabe beim Entwickeln ist zu verhindern, dass Bildbereiche im absoluten schwarzen oder weißen Bereich verschwinden, wenn diese in ein Standardbildformat exportiert werden. Dafür werden die dunklen Bereiche erhellt und die hellen dunkler gezogen. Besonders bei starker Sonneneinstrahlung, wie es bei der Datenerfassung der Fall war, ist dies nötig.

Wenn im fertigen Produkt die Texturen des 3D-Modells lichtneutral sein sollen, muss der Sonnen- und Schattenanteil ausgeglichen werden. Auch dazu müssen die hellen, von der Sonne beschienen Flächen verdunkelt und die Schattenflächen erhellt werden. Am Ende müssen beide Bereiche gleich aussehen. Bei der ursprünglichen Datenerfassung waren die Lichtverhältnisse so extrem, dass die Anpassung nur schwer oder gar nicht möglich war. Deswegen wurden an einem bewölkten Tag neue Fotos aufgenommen.

Adobe Bridge, zusammen mit Camera Raw, ermöglicht das Anwenden von Entwicklungseinstellungen auf alle Fotos einer Serie, so dass nur noch kontrolliert werden muss, ob alle Bilder zueinander passen.

Die entwickelten Bilder werden, je nach Qualitätsanforderung, in einem verlustfreien oder verlustbehafteten Bildformat exportiert. Aufgrund der hohen Datenmenge wurde in diesem Fall das verlustbehaftete JPEG Format gewählt, um eine größere Komprimierung der Daten zu ermöglichen.

5.2 RealityCapture

RealityCapture ist eine Software zum Generieren von 3D-Modellen aus Fotos und Laserscans. Es können sowohl terrestrische Bilder als auch Luftbilder verwendet werden. Die Fotos müssen hierfür nicht sortiert sein.

Die Projektpunktewolke im E57 Format und die entwickelten Fotos werden in RealityCapture importiert.

Die Software ordnet automatisch die Positionen der importierten Bilder untereinander an. Anschließend werden die Bilder und die Laser-Punktewolke aneinander ausgerichtet. Zur Berechnung der Fotogrammetrie können die Objektiv-Verzeichnungen korrigiert werden, sofern dies vorher noch nicht vorgenommen wurde.

RealityCapture kombiniert die Laserscan-Punktewolke und fotogrammetrischen Daten und generiert daraus ein hochauflösendes 3D-Modell. Zur Generierung kann zwischen einem *Normal*- oder *High*-Detail gewählt werden. Das *High*-Detail generiert die genausten Ergebnisse, braucht aber eine längere Berechnungszeit und erheblich mehr Arbeitsspeicher. Beim *Normal*-Detail wird die Auflösung aller Bilder vor der Berechnung halbiert.

Abbildung 5.4 zeigt eine automatisch ausgerichtete Fotoreihe des MAS-Gebäudes und die daraus generierte Punktewolke.

Abbildung 5.4: Screenshot aus der RealityCapture Software (eigene Darstellung)

Das fertige 3D-Modell kann so entweder verwendet und exportiert oder vorher noch in einem Zwischenschritt vereinfacht werden. Dafür wird das Modell für eine bessere Handhabung in einzelne Objekte mit je ca. 35 Millionen Polygonen unterteilt. Zur Vereinfachung stehen viele Programme zur Verfügung, in diesem Fall wurde ZBrush verwendet.

Nach der Bearbeitung in ZBrush (siehe Kapitel 5.3) werden die Objekte wieder in RealityCapture importiert. Auf die neuen Modelle werden nun die Texturen projiziert. Verwendet werden hauptsächlich die Fotos der Fotogrammetrie. Es können nach Bedarf aber auch die Farbwerte des Laserscanners verwendet werden. Wenn nötig können die Texturen noch in einer 3D-Mal- und Texturierungssoftware, wie Mari bearbeitet werden. Anschließend werden die texturierten 3D-Modelle wieder aus RealityCapture exportiert.

Exportiert und importiert werden die Meshes im offenen Wavefront OBJ (.obj) Format.

5.3 ZBrush

Die Meshteile aus RealityCapture werden in ZBrush importiert. Unerwünschte Teile werden entfernt und Unebenheiten behoben, die bei der Objektgenerierung entstanden sind („smoothing“). Durch das Wegschneiden der unerwünschten Elemente wird die Anzahl der Polygone bereits reduziert. Um auf die erforderte Komplexität zu kommen, müssen in der Regel aber noch weitere Reduzierungen angewendet werden. Dazu bietet ZBrush diverse Methoden und Optionen an, die je nach Objektart unterschiedlich gute Ergebnisse vorweisen können.

Außerdem kann in ZBrush das UV-Mapping des Objektes vorgenommen werden. Je nach Objekt und Anforderung werden die UVs komplett automatisch generiert oder händisch angepasst.

Am Ende werden die Meshes für die Texturierung in RealityCapture wieder als OBJ exportiert.

5.4 Maya

Zur weiteren Verarbeitung werden die Meshes in eine 3D-Modellierungs-Software geladen. Dafür können alle gängigen 3D-Grafiksuiten verwendet werden. In diesem Projekt wurde Autodesk Maya gewählt.

Die importierten Meshes werden zuerst an die „Real World Scale“ angepasst, damit die Skalierung der Realität entsprechen. In Maya sind 100 Einheiten ein Meter.

Die Hauptaufgabe in der 3D-Modellierungs-Software ist, fehlerhafte Meshes und Texturen zu korrigieren. Die generierten Objekte können, je nach Größenordnung und Qualität der Laserscans und Fotogrammetrie, unterschiedliche Fehler aufweisen.

Verhältnismäßig kleinteilige Bereiche, wie Geländer oder Bäume werden bei dem Laserscanning und der Fotogrammetrie kaum erfasst und können gar nicht oder nur sehr schlecht in ein 3D-Objekt umgewandelt werden.

An Stellen, wo weder der Laserscan noch die Fotogrammetrie Messungen vornehmen konnten, können die Objekte Löcher oder extreme Abweichungen von der realen Geometrie aufweisen – zum Beispiel an Dachflächen oder verdeckten Bereichen.

Stark spiegelnde Flächen, wie Fenster oder Wasserflächen, können kaum erfasst werden.

Generell können die Objekte große Ungenauigkeiten aufweisen. Besonders bei ungünstigen Lichtverhältnissen oder monotonen Flächen kann die Fotogrammetrie nur minderwertige Ergebnisse aufweisen. Auch der Lasercan kann bei zu wenigen Messpunkten und flachen Blickwinkeln nur ungenau messen. So kann es vorkommen, dass große Teile eines 3D-Objektes hügelig sind. Je nach Anforderung ist dies störend oder vernachlässigbar.

Diese und andere Fehler müssen händisch korrigiert werden. Dieser Vorgang beansprucht bei der Datenauswertung den meisten Arbeitsaufwand.

Kleinteilige Objekte, wie Geländer, können entweder anhand von den vorhandenen Daten rekonstruiert oder basierend auf Messwerten nachmodelliert werden. Die neuen Objekte können anhand der vorliegenden Daten exakt in der Szene positioniert werden (siehe Abbildung 5.5).

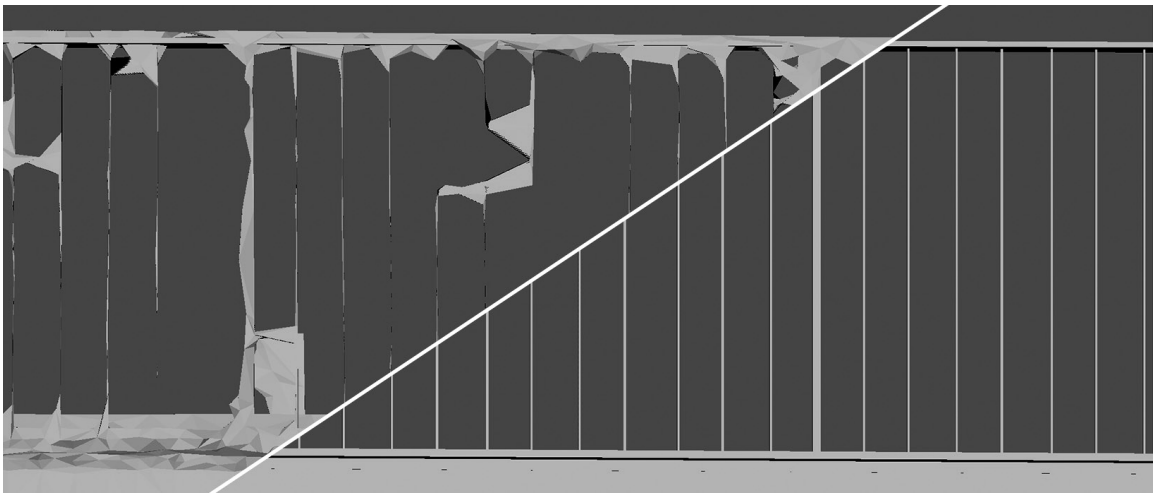
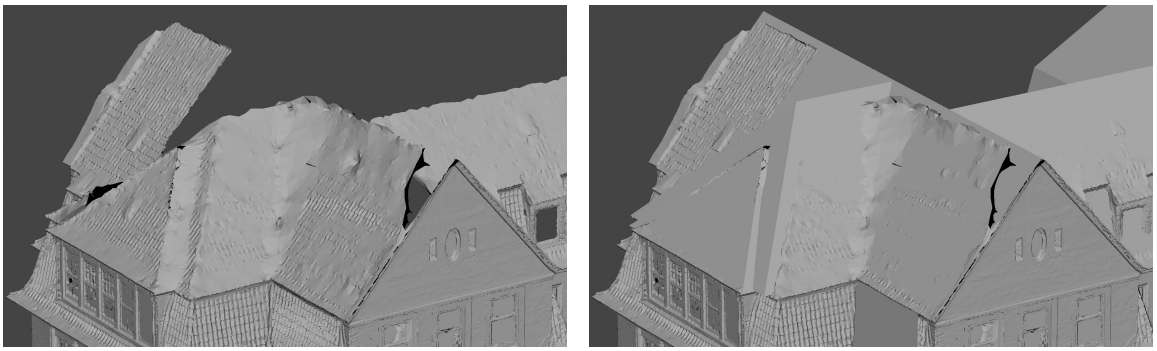


Abbildung 5.5: Originales 3D-Objekt (oben links) und Nachbildung (unten rechts) (eigene Darstellung)

Für die Glasscheiben werden einfache Flächen in den Fensterrahmen eingezogen. Beim texturieren bekommen diese ein spiegelndes oder gläsernes Material.

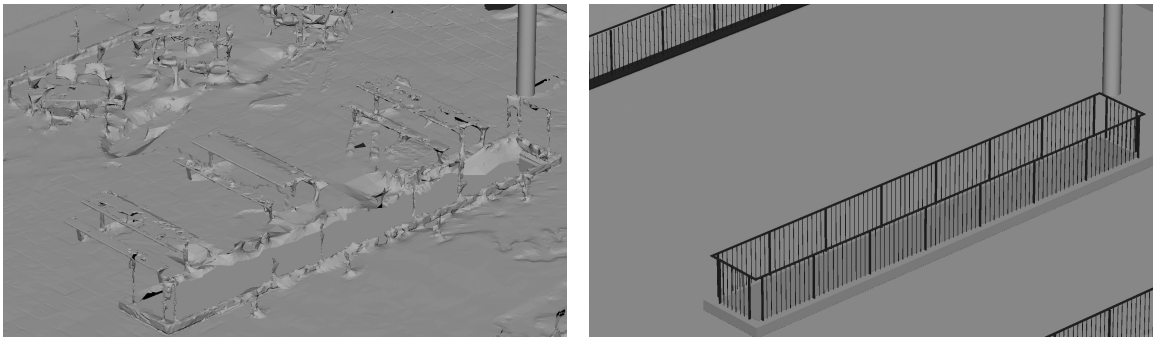
Bei komplett fehlenden oder fehlerhaften Bereichen muss ein passendes Objekt modelliert werden, das diese Stelle abdeckt. Bei den Dachflächen können dafür öffentliche Luftbildaufnahmen zu Hilfe genommen werden. Je nach Anforderung muss hier mehr oder weniger genau gearbeitet werden. In diesem Projekt lag nur wenig Gewicht auf der genauen Darstellung der Dächer, da das 3D-Modell primär aus der Bodenperspektive betrachtet werden soll.

Bei unebenen Flächen sollte entweder die bestehende Fläche korrigiert oder das komplette Objekt nachmodelliert werden. Die Fläche kann dafür in Maya oder anderen Programmen – wie ZBrush – geglättet werden. Ob das rentabel ist oder nicht, hängt von der Art der Fläche ab. Wenn ein Objekt neu modelliert wird, kann das alte Modell zur Orientierung genutzt werden. Dadurch behält das neue Objekt die genauen Dimensionen und Positionen des ursprünglichen Modells. Ein großer Vorteil des neu-modellieren ist, dass die Anzahl der Polygone minimiert wird, was besonders bei aufwendigen Szenen ausschlaggebend ist.



(a) Original **(b)** Original mit Lückenfüllern

Abbildung 5.6: Lücken im Dach des Altbaus (eigene Darstellungen)



(a) Original **(b)** minimalisierter Nachbau

Abbildung 5.7: Rekonstruktion der Terrasse des Neubaus (eigene Darstellungen)

Bäume und andere Pflanzen können in der Regel nicht durch Fotogrammetrie oder Laserscanner erfasst werden, da jede kleinste Bewegung des Objektes Fehler verursacht. Deswegen werden diese am Anfang der Datenverarbeitung als Datenmüll verworfen. Wenn in dem fertigen Modell trotzdem Bäume und Pflanzen vorkommen sollen, müssen sie extra eingesetzt werden. Dazu können entweder die originalen Bäume nachmodelliert oder vorhandene Modelle genutzt werden. Diese werden entweder jetzt oder gegebenenfalls in einem anschließenden Programm eingebaut.

Ein weiterer Arbeitsschritt ist das Erstellen von Materialien. Jedes Objekt bekommt ein passendes Material und wird mit der entsprechenden Textur verknüpft. In diesem Projekt wird das Textur-Streaming Plugin Granite von Graphine Software verwendet. Das Plugin ermöglicht die Verwendung hochauflösender 16K-Texturen.

Falls der Raum hinter Fensterscheiben als 3D-Modell zur Verfügung steht, können die Glasflächen ein Glas-Material zugewiesen bekommen. Dies würde einen realitäts-

nahen Eindruck vermitteln. Wenn dies nicht möglich ist, wird ein reflektierendes Material verwendet. Gegebenenfalls kann der dahinterliegende Raum schemenhaft, als Textur hinter die Glasscheibe gelegt werden, wodurch ein räumlicher Eindruck entsteht. In diesem Projekt sind zwar keine 3D-Raummodelle vorhanden, aber teilweise wurden Texturen hinter die Scheiben gelegt.

Falls Texturen fehlerhaft sind, müssen diese korrigiert oder ersetzt werden. Das kann entweder in diesem Programm oder in einer speziellen 3D-Mal- und Texturierungssoftware wie Mari durchgeführt werden.

Fehlende Texturen können durch öffentliche Luftaufnahmen oder Standard-Texturen ersetzt werden.

Abbildung 5.8: Texturierte Objekte in Maya (eigene Darstellung)

Nach der Bearbeitung in der 3D-Grafiksuite ist die Modellierung der 3D-Objekte beendet. Das Projekt ist jetzt möglicherweise bereits fertiggestellt. Es kann aber noch weiter verarbeitet werden. Gegebenenfalls wird die Szene dafür in eine andere Software exportiert. Dafür wird wieder das Wavefront OBJ Dateiformat oder direkt ein Format für die Zielsoftware verwendet. In den meisten Fällen ist es nötig, die Materialien und Texturen in der Zielsoftware neu zu erstellen bzw. neu zu verlinken.

In diesem Projekt wird zur weiteren Verarbeitung die Unreal Engine verwendet. Zum Übertragen der Daten wird das Autodesk FBX (.fbx) Format mit dem Granite Plugin für das Textur-Streaming verwendet. Wegen der großen Datenmengen und des Ziels, die Szene in VR darstellen zu können, wurden die Bereiche hinter dem MAS Gebäude entfernt. Das TIDE-Gebäude, der Bunker und die Hälfte des Altbaus sind weggefallen.

5.5 Analyse der 3D-Szene

5.5.1 Meshes

Die 3D-Modelle wurden aus 22 Laserscansätzen und über 1000 Bildern für die Fotogrammetrie erstellt. Die Informationen aus der Fotogrammetrie dienten dabei zum Korrigieren von Fehlern, die in der Auswertung der Laserscandaten entstanden sind. Hauptsächlich wurden die Bilder aber für die Texturierung verwendet.

Ein eindeutiger Vorteil der Verwendung von Laserscannern ist die detailgetreue Messung. Der FARO Focus^S 350 hat eine 3D-Positionsungenauigkeit von 2 mm bei 10 m Entfernung bis 36 mm bei 350 m (siehe Kapitel 3.4). Das ist, bei einer Gesamtgröße des Ausschnittes vom Campus Finkenau von 133 m × 65 m × 23 m (L, B, H), eine geringe Fehlergröße.

Die aus diesen Daten entstehenden 3D-Modelle haben somit alle eine zentimetergenaue Form und Größe. Zwar könnte man die Objekte theoretisch auch händisch modellieren, dies würde aber eine enorme Arbeit erfordern und voraussetzen, dass alle Größen exakt bekannt sind.

Allerdings ist die Qualität der automatisch generierten 3D-Objekte sehr schlecht. Selbst nach vielen Optimierungsarbeiten ist die Anzahl der Polygone immer noch deutlich von einer optimalen Menge entfernt. Außerdem haben die Meshes keine geordnete geometrische Topologie. Dies verursacht nicht zwingend Probleme, sorgt aber für einen höheren Rechenaufwand und könnte bei Interaktion des Spielers mit den 3D-Objekten zu Fehlern (Glitches) führen. Um Letzteres zu verhindern, muss der Boden und der Bereich, in dem sich der Spieler bewegen darf, extra erstellt werden.

Außerdem weisen die meisten Oberflächen Unebenheiten auf, die eine größere Entfernung oder flachen Winkel zu den Scanstandorten haben. Die meisten wurden zwar ausgeglichen, teilweise sind sie aber immer noch zu sehen.

Um diesen Problemen zu entgehen, wurde das Gebäude des Neubaus komplett nachmodelliert. Das verursacht zwar zusätzliche Arbeit, behebt aber alle oben beschriebenen Probleme. Nachmodellierte Objekt behalten die Genauigkeit des Originals, haben aber eine deutlich geringere Anzahl an Polygonen und eine saubere geometrische Topologie. Das neue Modell des Neubaus hat, zusammen mit der Mensa und der Terrasse, ca. 20 000 Polygone. Im Vergleich dazu hat das nicht neu modellierte MAS-Gebäude über 487 000 Polygone. Die 3D-Szene umfasst insgesamt über 1,5 Mio. Polygone. Das ist angesichts der Komplexität eine akzeptable Anzahl und sollte keine Leistungsprobleme verursachen. Abbildungen 5.9 und 5.10 zeigen zur Veranschaulichung der Topologie das Drahtgittermodell des Neubaus und des MAS-Gebäudes.

Die nachmodellierten Modelle können trotzdem Texturen aus der Fotogrammetrie erhalten, hier entsteht also kein Unterschied zu den originalen Objekten.

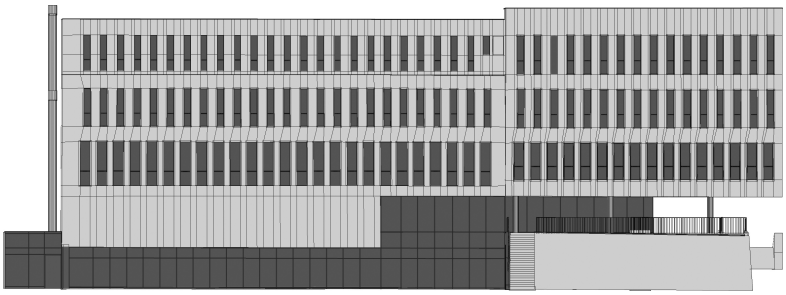


Abbildung 5.9: Drahtgitter-Ansicht des nachmodellierten Neubaus (eigene Darstellung)

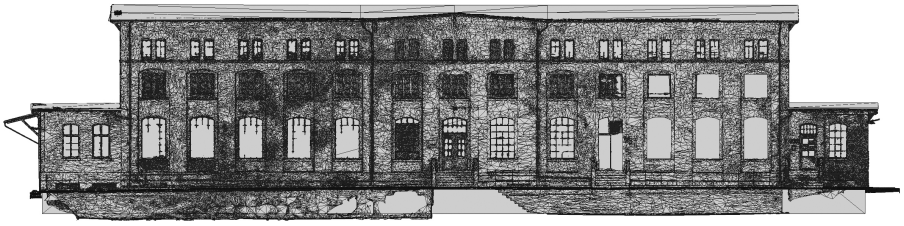
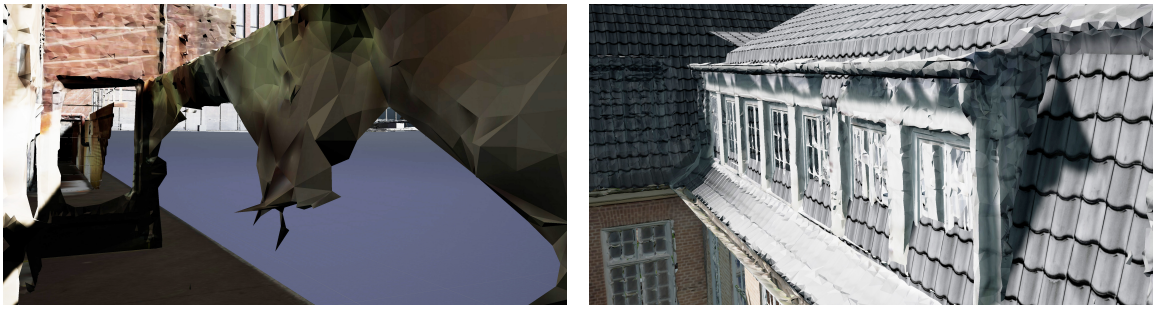


Abbildung 5.10: Drahtgitter-Ansicht des originalen MAS-Gebäudes (eigene Darstellung)

Generell sehen die 3D-Objekte in einer normalen Ansicht, vor allem in VR, sehr gut aus. Nur wenn die Objekte aus sehr kleiner Distanz begutachtet werden, fallen die Unebenheiten auf. Genau so verhält es sich mit den Texturen, die von sehr nahem betrachtet werden müssen, damit die Auflösung zu gering ist. Störend ist aber das Fehlen von anderen Textur-Maps (siehe Kapitel 5.5.2). Problematisch werden die Unebenheiten beim Beleuchten aus flachen Winkeln, wenn zum Beispiel an einer Wand flach empor geleuchtet wird. Hier werfen die Beulen in den Meshes teilweise starke Schatten auf.



(a) Löcher im Schacht vor dem MAS-Gebäude **(b)** Überlappende Meshes im Dachgeschoss des Altbaus

Abbildung 5.11: Fehler im 3D-Modell (eigene Darstellungen)

An einigen Stellen weisen die 3D-Objekte noch Fehler wie Löcher oder Überlappungen auf. Diese befinden sich aber an gar nicht oder nur schlecht sichtbaren Bereichen. Zum Beispiel im Schacht beim MAS-Gebäude befindet sich ein großes Loch und an den Dächern des Altbaus überlappen sich einige Flächen (siehe Abbildung 5.11).

5.5.2 Materialien

Die aus der Fotogrammetrie und Laserscanning generierten 3D-Objekte wurden in RealityCapture UV-unwrapped und texturiert. Die Texturen wurden in Mari überarbeitet und in Maya mit dem Granite Plugin verknüpft. Mit Granite werden die Texturen auch in Unreal auf die Meshes gemappt. Das ermöglicht größere Texturauflösungen und eine bessere Performance.

Die Materialien dieser Objekte bestehen aber nur aus einem diffusen Farbanteil. Die diffuse Textur hat zwar eine hohe Auflösung und damit eine verhältnismäßig hohe Detailreiche, aber ansonsten beinhalten die Materialien keine Informationen. Da keine anderen Maps mit dem Material verknüpft sind, werden für das Shading die Standardwerte verwendet. Der *Metallic*-Anteil ist (großteils korrekterweise) null, der *Specular*- und der *Roughness*-Anteil liegen bei je 0,5. Die *Normal*-Map ist überall gleich, also flach.

(a) Wandmaterial mit reinem Diffus-Anteil **(b)** Glasscheiben mit Textur im Hintergrund

Abbildung 5.12: Wand- und Glasmaterialien des MAS-Gebäudes (eigene Darstellungen)

Der entstehende Eindruck der Gebäudeflächen ist deswegen planar und monoton (siehe Abbildung 5.12a). Für einen besseren Eindruck müsste zumindest eine *Roughness*- und eine *Normal*-Map generiert werden. Die Qualität der UV-Maps ist allerdings problematisch. Da die UVs automatisch generiert wurden, sind sie größtenteils chaotisch und unsortiert (siehe Abbildung 5.13). Dadurch lassen sich sowohl automatisiert, als auch händisch nur schwer die benötigten Maps generieren.

Für eine *Normal*-Map ist die vorhandene Auflösung im Verhältnis zur Oberfläche noch zu gering. Es sind zum Beispiel kaum Pixel zwischen den einzelnen Ziegelsteinen vorhanden, um eine Delle zu vermitteln. So ist bei dem MAS-Gebäude die Lücke zwischen zwei Ziegelsteinen nur vier Pixel breit.

(a) MAS-Gebäude **(b)** Boden im Innenhof und Teile des Altbaus

Abbildung 5.13: Texture Maps (eigene Darstellungen)

Für die zwei Hauptflächen, also die Wände des Neubaus und MAS-Gebäudes, wurden *Roughness*-Maps generiert. Die dabei entstandenen Texturen bieten aber kaum einen Mehrwert, da die Werte ausschließlich zwischen 80% und 100% *Roughness* liegen. Um die Rauheit der Oberflächen genauer darzustellen, müssten die einzelnen Bestandteile händisch festgelegt werden. Das Messen des Brechungsindexes wäre nicht das Problem, allerdings müssten sämtliche Fugen, Ziegelsteine und andere Flächen händisch aus den Texturen herausgefiltert werden.

Bei den meisten Fenstern wurden Flächen hinter die Glasscheiben gesetzt. Diese haben, wenn vorhanden eine abgedunkelte Textur des Raumes dahinter. Andernfalls wurde eine generische Textur verwendet. In beiden Fällen entsteht ein deutlich realistischerer Eindruck (siehe Abbildung 5.12b) im Vergleich zu ausschließlich spiegelnden Flächen.

5.6 Unreal Engine

Die weitere Verarbeitung des Projekts sieht eine Visualisierung des Finkenau Campus in VR und die Simulation von Videomapping und Lichtshows vor. Dafür wird die Unreal Engine 4.21.2, eine Spiele-Engine von Epic Games, verwendet.

Die Unreal Engine unterstützt die derzeit verbreitetsten HMD-VR-Geräte (Head Mounted Display). Dadurch ist die VR-Visualisierung schnell zu realisieren. Für die Lichtsimulation ermöglicht Unreal das Verwenden von physikalischen Größen und IES-Lichtprofilen (Illuminating Engineering Society of North America, amerikanisches Format von Lichtstärkeverteilungskurven). Für das Videomapping gibt es zwar keine direkte Unterstützung, aber mehrere indirekte Umsetzungsmöglichkeiten.

Um die 3D-Szene in Unreal verwenden zu können, muss lediglich die korrekt aus Maya exportierte FBX-Datei in ein Unreal Projekt importiert werden. Ein zusätzliches Plugin macht es möglich, hochauflösende Texturen zu verwenden. Nativ unterstützt wird eine Pixelzahl von 8192×8192 (8K).[1] Die Texturen in RealityCapture wurden allerdings mit 16384×16384 (16K) Pixeln erstellt. Damit diese Texturen auch in Unreal verwendet werden können, wird das Textur-Streaming Plugin Granite von Graphine verwendet. Testversion-Lizenzen können kostenfrei auf der Internetseite des Herstellers beantragt werden. Das Projekt wird dann mit einem Wasserzeichen in der unteren rechten Ecke markiert.

Die importierten 3D-Objekte müssen gegebenenfalls in der Skalierung angepasst werden. Diese kann, vor allem beim Betrachten in VR, von der realen Skalierung abweichen. Wenn die Größeneinheit in Maya auf Zentimeter eingestellt ist, sollte die Skalierung korrekt sein. Beim Betrachten in VR scheint die Größe oft dennoch falsch. In dem Fall muss der Entwickler die Skalierung nach Gefühl anpassen.

Neue Projekte werden automatisch mit einer Standard-Lichtquelle und einem Standard-Himmel (*Skybox*) erstellt. Mit diesen können unterschiedliche Lichtstimmungen und Tageszeiten simuliert werden (siehe Kapitel 6.4). Wenn ein neues, leeres Level generiert wird, können diese aus anderen Leveln kopiert oder neu erstellt werden.

5.6.1 Virtual Reality

Beim Erstellen von neuen Projekten bietet die Unreal Engine 4 diverse Vorlagen an. Für VR gibt es das *Virtual Reality*-Preset. In diesem werden automatisch die für verschiedene VR-Brillen notwendigen Umgebungen generiert. In dem neuen Projekt stehen zwei Level zur Verfügung:

[1] *Unreal Engine 4 Documentation* 2019: *Texture Support and Settings*

1) „HMDLocomotionMap“ und 2) „MotionControllerMap“.
Das erste Level ist für „HMD (+ Gamepad Optional)“, also für einfache VR-Brillen ohne Motiontracking im Raum, wie die Occulus Rift und Gear VR-Brillen. Das zweite ist für „HMD + Motion Controller“, also für VR-Brillen mit Motion Controller, wie die HTC Vive.

Falls Virtual Reality Kompatibilität in ein bestehendes Projekt implementiert werden soll, ist es am einfachsten, ein neues VR-Projekt zu erstellen und die VR-Elemente in das bestehende Projekt zu übertragen. Gegebenenfalls müssen auch die Tastenbelegungen mit kopiert werden. Diese sind in den *Project Settings* zu finden.[2]

In den jeweiligen Leveln sollte das abspielen und interagieren mit der entsprechenden VR-Hardware funktionieren. Eventuell muss noch festgelegt werden, in welchen Bereichen sich der Spieler bewegen und teleportieren darf. Dafür muss eine Bodenfläche mit Kollision vorhanden sein. Normalerweise hat jede Fläche automatisch eine Kollisionsmaske, die zum Teleportieren verrechnet wird. Bei den 3D-Objekten des Campus Finkenau ist dies nicht der Fall. Stattdessen muss eine eigene Fläche zum Teleportieren eingezogen werden, die sich mehr oder weniger am Boden der Szene orientiert. Diese Flächen können in der Spielumgebung ausgeblendet werden.

[2]Looman, 2018.

6 Simulation

Für die Simulation sollen typische Lichtshow- und Videomapping-Situationen dargestellt werden. Physikalisch korrekte Simulationen sind nicht zwingend notwendig und auch nicht immer umsetzbar. Möglichst realitätsnahe Darstellungen sind aber erwünscht.

Am Ende des Abschnitts zeigen die Abbildungen 6.17 und 6.18 ein fertiggestelltes Simulationsbeispiel mit Lichtshowelementen und Videomapping.

6.1 Videomapping

Für das Videomapping ist wichtig, dass verschiedene Bilder- und vor allem Videoformate abgespielt werden können. Außerdem ist es vorteilhaft, die Wirkung unterschiedlicher Farbsituationen auf den roten Backsteinwänden zu visualisieren.

Für die Simulation stehen drei grundlegend unterschiedliche Methoden zur Verfügung. Das Bildmaterial kann als Textur auf das 3D-Mesh gemappt werden, als Kameraprojektion oder mit einer gerichteten Lichtquelle projiziert werden. Alle Methoden haben unterschiedliche Vor- und Nachteile.

Bei allen drei Methoden können sowohl Bilder als auch Videos verwendet werden. Das Seitenverhältnis spielt dabei keine Rolle, muss aber eventuell händisch angegeben werden. Neue Materialien können einfach per Drag-and-Drop in einen Mediaplayer gezogen und abgespielt werden. Auch das Erstellen von Playlisten ist möglich.

Bildmaterial als Textur

Die einfachste Methode ist, das Bildmaterial als Textur direkt auf das 3D-Mesh zu legen. Wenn dabei die eigentliche Textur komplett ersetzt wird, verliert die Simulation komplett an realistischen Eindruck. Stattdessen sollte entweder die Ziegelstein-Textur in der Videodatei unter das Video gelegt oder in Unreal die beiden Texturen mit einer Alpha-Maske vermischt werden. Wenn die Textur nur als diffusen Anteil in dem Material eingebunden wird und die Umgebungsbeleuchtung dunkel ist, erscheint das Bild düster (siehe Abbildung 6.1). Statt dessen muss das Bildmaterial als Emissionswert angegeben werden. Dann leuchtet das Material von selber und wird nicht von der Umgebungsbeleuchtung beeinflusst (ähnlich wie Abbildung 6.2).

Vorteil dieser Methode ist, dass sie kaum Rechenaufwand benötigt. So kann auch auf leistungsschwachen Systemen schnell eine Simulation verwirklicht werden. So lässt sich einfach erkennen, ob das Bildmaterial auf die Fassade des Gebäudes passt oder ob Elemente, wie zum Beispiel Fenster, im Weg sind.

Allerdings funktioniert diese Variante nur, wenn das Videomapping auf einer einzelnen, planaren Wand projiziert wird. Sobald eine kompliziertere Fläche oder um Ecken herum gemappt werden soll, wäre das so nicht oder nur mit einem erheblichen Aufwand möglich. Auch alle anderen Aspekte des Videomappings sind nicht simulierbar.

Abbildung 6.1: Videomapping mit Bildmaterial als Textur (eigene Darstellung)

Decals und Kameraprojektion

Die Unreal Engine ermöglicht das Projizieren von Texturen mittels der *Decals*-Funktion. Das *Deferred Decal* projiziert eine ihm zugeordnete Textur auf alle Objekte in seinem Umkreis. Dabei stehen in einem *Deferred Decal*-Material elf verschiedene Methoden zur Verfügung, die festlegen wie die Textur projiziert wird. Für das Videomapping sollte die Textur entweder als diffuse Farbe oder als Lichtfarbe projiziert werden.[1]

Ein Problem bei *Decals* ist, dass alle Objekte im Umkreis, unabhängig von ihrer Position, die Textur aufprojiziert bekommen. Das bedeutet, dass auf den projizierten Flächen keine Schatten entstehen, wenn Objekte zwischen ihnen und der *Decal*-Quelle stehen. Deswegen haben seit August 2015 mehrere Benutzer im Unreal Forum

[1] *Unreal Engine 4 Documentation* 2019: *Decals*

ein Spielelement-Skript (*Blueprint*) erstellt, dass der Projektion eine Raumrichtung gibt.[2] Zuletzt hat Christopher Remde im Rahmen einer Bachelorarbeit das Blueprint weiterentwickelt.[3]

Das inzwischen umfangreiche *Blueprint* ordnet das *Decal* einer Kamera zu. So können Blickrichtung, Tiefe, Brennweite und viele weitere Eigenschaften aus der Kamera entnommen und bei der Berechnung der Projektion berücksichtigt werden.

Hauptfunktion ist, dass Objekte im Strahlengang einen Schatten werfen. Zudem erhält der Nutzer viele Einstellungsmöglichkeiten wie Projektionsgröße und Kantenunschärfe.

Allerdings ist der Strahlengang nicht kegelförmig, sondern immer parallel. So werfen Objekte einen Schatten, die eigentlich gar nicht im Lichtkegel wären. Dies kann Probleme verursachen, wenn knapp an Objekten vorbei projiziert wird, wie es bei der Neubaufläche über der Terrasse der Fall ist. Wenn die Kameraprojektion aus der ersten Etage simuliert wird, wirft die Terrasse einen Schatten auf die halbe Wand (siehe Abbildung 6.2). Eigentlich befindet lediglich das Geländer im Strahlengang.

Bei beiden Methoden (*Decals* oder mit Kameraprojektion) entstehen perspektivische Verzerrungen, wenn die Projektion nicht orthogonal zur Fläche steht. Diese könnten – wenn gewünscht – im Material entzerrt werden.

Abbildung 6.2: Videomapping mit Kameraprojektion-*Blueprint* (eigene Darstellung)

Nachteil dieser Projektionen ist, dass die Textur auf das Objekt selber gelegt wird (ähnlich wie bei der oben beschriebenen Methode „Bildmaterial als Textur“). Das Bildmaterial wird also nicht auf die Objektoberfläche „geleuchtet“, sondern wird

[2]Der Forumsbeitrag ist im Unreal Forum abrufbar (Unreal Engine Forum, 2015).

[3]Das *Blueprint* ist verfügbar auf GitHub (Remde, 2019).

quasi als Textur mit in das Material vom bestrahlten Objekt eingemischt. Die Textur wird deswegen immer mit derselben Intensität beziehungsweise Helligkeit dargestellt. Wenn ein helles Bild projiziert wird, verschwindet so gegebenenfalls die Textur der Wand. Um dies zu verhindern, kann entweder das Bildmaterial abgedunkelt oder leicht transparent gemacht werden. Physikalisch korrekte Ergebnisse sind allerdings nicht erreichbar.

Lichtquelle mit Textur

Lichtquellen können in der Unreal Engine mit Materialien versehen werden, die dessen Leuchtstärke beeinflussen. Dabei entsteht ein Effekt, der dem von Gobos ähnelt. Dafür wird eine Textur mit dem *Emissive Color* Ausgang eines *Light Function*-Materials verbunden. Anders als der Name vermuten lässt, wird allerdings nur die Helligkeit und nicht die Farbe des Lichts beeinflusst.[4] Die Farbe kann nur in der Lichtquelle selber geändert werden und dabei auch nur ein Farbwert zur selben Zeit. Das heißt, es kann mit der *Light Function* ein einfarbiges Muster projiziert werden.

Um dennoch ein farbiges Bild darstellen zu können, kann die additive Farbmischung von Licht genutzt werden. Dafür muss das Bildmaterial in seine roten, grünen und blauen Farbkanäle unterteilt werden. Mit jedem Kanal wird eine eigene Textur erstellt und diese jeweils einer eigenen gerichteten Lichtquelle (*Spot Light*) zugeordnet. Die Lichtquellen müssen die Farbe des jeweiligen Kanals haben. Wenn alle drei *Spot Lights* übereinander liegen, entsteht auf der beleuchteten Fläche wieder das originale Farbbild.

Ein weiteres Problem der *Light Function* ist, dass eine rechteckige Textur auf einen runden Lichtkegel gelegt wird. Dadurch werden die Ecken der Textur abgeschnitten. Um das zu verhindern, muss das Material kleiner skaliert werden. Allerdings entsteht dann das Problem, dass das Bildmaterial in alle Richtungen wiederholt wird. Alternativ können nur die äußersten Pixel gestreckt werden. Damit dabei nicht die äußeren Farbwerte wiederholt werden, muss das Bildmaterial einen schwarzen oder transparenten Rand haben. Dafür wird die Textur mit einem Bild überlagert, dass einen 1 Pixel großen schwarzen Rand hat.

Damit das Material beim Skalieren nicht verzerrt wird, muss das Seitenverhältnis beziehungsweise die Auflösung angegeben werden. Andernfalls geht die Skalierung von einer quadratischen Datei aus, wie es bei Texturen üblich ist.

Diese Videomapping-Methode liefert die physikalisch akkuratesten Ergebnisse. Den *Spot Lights* können, wie allen Lichtquellen in der Unreal Engine 4, auf physikalischen Größen basierende Einheiten zugeordnet werden. Die Helligkeit kann

[4] *Unreal Engine 4 Documentation* 2019: *Light Functions*

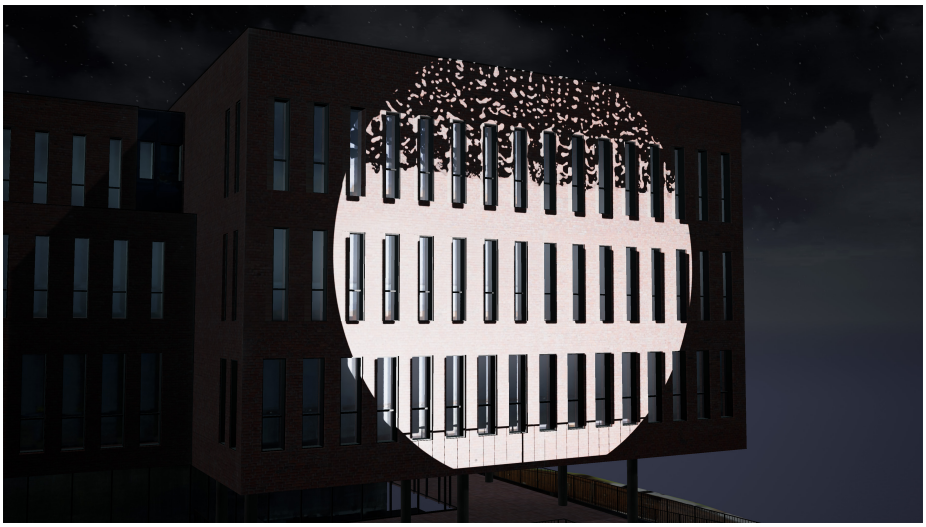


Abbildung 6.3: Abgeschnittene Ecken wegen des runden Lichtkegels (eigene Darstellung)

entweder in Candela oder Lumen angegeben werden.[5] Außerdem ist es möglich, ein IES-Lichtprofil einzubinden. Es besteht auch die Möglichkeit, die Beleuchtungsstärke basierend auf dem fotometrischen Entfernungsgesetz (*Inverse Square Falloff*) abfallen zu lassen. Bei den anderen beiden Methoden stehen diese Funktionen nicht zur Verfügung.

Außerdem wird das Bild, ähnlich wie bei einem Projektor, kegelförmig von einer Lichtquelle aus ausgestrahlt. Dadurch werden diverse Problematiken aufgezeigt, die auch in einer echten Mapping-Situation entstehen:

Die Helligkeit nimmt vom Mittelpunkt des Lichtkegels aus ab und wird zum Rand hin komplett dunkel.

Objekte, die im Strahlengang stehen, hinterlassen einen Schatten auf der Wand. So lässt sich schnell feststellen, ob die gewählte Beamer-Position sinnvoll ist.

Wenn die Lichtquellen nicht orthogonal zur bespielten Fläche angeordnet sind, entstehen perspektivische, trapezförmige Verzerrungen. Diese können, ähnlich wie beim realen Videomapping, wieder entzerrt werden. Allerdings steht dafür kein Interface zur Verfügung. Die Verzerrungen müssen mathematisch in der Textur der *Light Function* vorentzerrt werden. Ungleichmäßige Verzerrungen, zum Beispiel um Häuserecken herum, können so nur schlecht oder auch gar nicht korrigiert werden.

Tiefenschärfe kann generell nicht simuliert werden. Genau wie bei der Gobo-Simulation, sind die *Light Function* Muster immer scharf (siehe Kapitel 6.2.5).

[5] *Unreal Engine 4 Documentation* 2019: *Physical Lighting Units*

Nachteil dieser Videomapping-Methode ist die erforderliche hohe Rechenleistung. Es müssen immer drei Lichtquellen auf einmal berechnet werden. Und diese müssen in den meisten Fällen eine lange Reichweite und Helligkeit haben, damit das Bildmaterial auch zu erkennen ist.

Abbildung 6.4: Videomapping mit drei (RGB) *Light Functions* (eigene Darstellung)

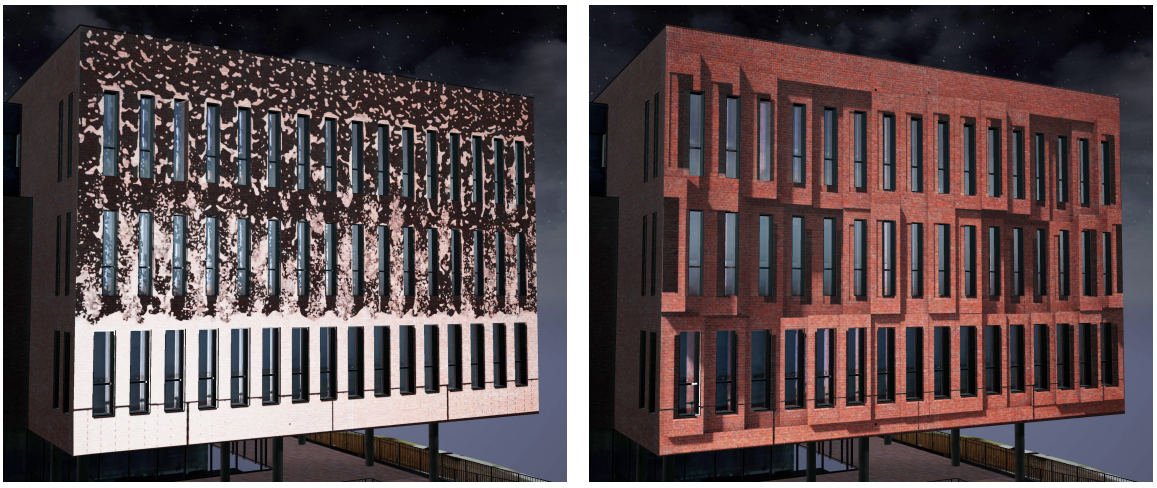


Abbildung 6.5: Beispiele aus einer Videomapping-Show mit drei (RGB) *Light Functions* (eigene Darstellungen)

6.2 Lichtshows

Bei der Simulation der Lichtshows soll es möglich sein, den generellen Eindruck der Show zu visualisieren und die Positionierung der Scheinwerfer zu testen.

Dafür sollten die gängigen Scheinwerfervarianten und deren typischen Verwendungen umsetzbar sein. Die verwendeten Scheinwerfertypen sind Spotlights, Washlights und Beamligths beziehungsweise jeweils ihre konventionellen, unbewegten Gegenstücke. Als Effekte sollen Gobos, Bewegungen, Farbänderung und Lichtstrahlen in der Luft (Haze-Effekt) visualisiert werden.

6.2.1 Lichtstärkeverteilungskurven

Wie bereits in Kapitel 6.1 beschrieben, ermöglicht die Unreal Engine 4 das Verwenden von physikalischen Größen für alle Lichtquellen und das Einbinden von IES-Lichtprofilen für gerichtete und punktförmige („Omni-“) Lichter.

Um ein IES-Profil in Unreal nutzen zu können, muss die .ies-Datei in das Unreal Projekt geladen werden. Diese erstellt dann automatisch aus dem Profil eine zweidimensionale IES-Textur. Abbildung 6.6 zeigt die IES-Textur und den entstehenden Lichtschein des SGM G·4 Wash-Beam Scheinwerfers.[6] In dieser Textur stehen diverse Variablen, die manuell angepasst werden können. Insbesondere ist hier die Lichtstärke in Candela eingestellt. Es besteht derzeit (in Version 4.21.2) ein Bug, wodurch dieser Wert verändert wird. Wenn die Lichteinheit einer Lichtquelle von Einheitslos (*Unitless*) auf Candela geändert wird, wird bei der ihr zugeordneten IES-Textur die Lichtstärke auf einen deutlich kleineren Wert verändert (ca. Faktor 1000). In dem Fall muss der korrekte Candela-Wert manuell zurückgestellt oder das IES-Profil neu generiert werden.

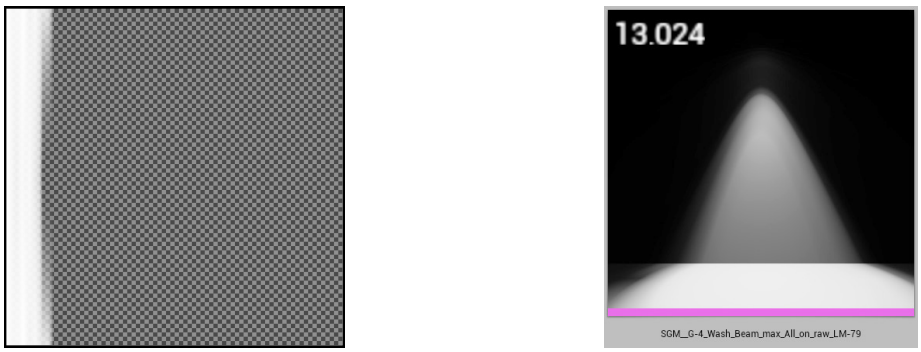


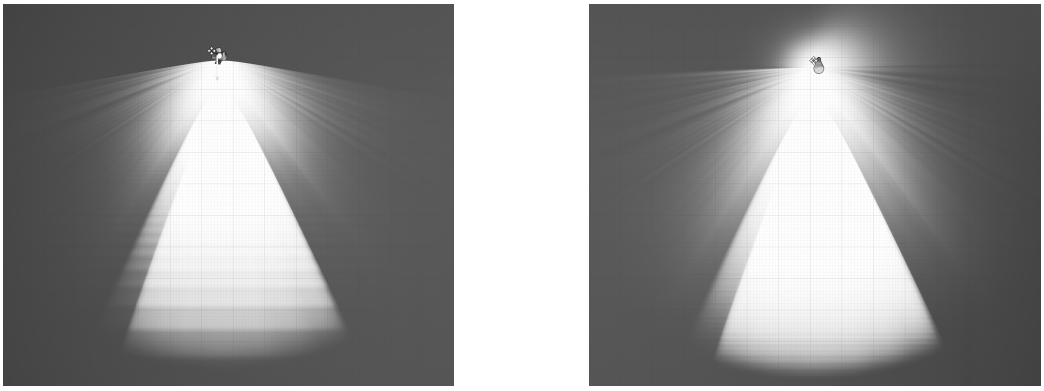
Abbildung 6.6: IES-Textur und der entstehende Lichtschein des SGM G·4 Wash-Beam Scheinwerfers (eigene Darstellungen)

[6] SGM, 2019.

Wenn die *Use IES Intensity*-Option aktiviert ist, verwendet die Lichtquelle anstelle ihres eigenen Helligkeitswertes die Lichtstärke der IES-Textur. Mit dem *IES Brightness Scale*-Slider kann die Skalierung von diesem Wert – also die Helligkeit des Scheinwerfers – prozentual eingestellt werden.

IES-Texturen können sowohl auf punkt- als auch auf kegelförmigen Lichtquellen (*Point Lights* und *Spot Lights*) angewendet werden. Bei beiden entsteht der gleiche Lichtschein. Allerdings beschneidet der äußere Kegel der *Spot Lights* den Lichtkegel. Dieser Kegel ist auf maximal 179° beschränkt. Alle IES-Daten jenseits dieses Winkels werden verworfen. So wird zum Beispiel Streulicht hinter dem Scheinwerfer komplett weggeschnitten.[7]

Dieser Effekt ist in manchen Situationen aber auch vorteilhaft, da so der Lichtkegel verändert werden kann.



(a) *Spot Light* **(b)** *Point Light*

Abbildung 6.7: Lichtschein eines *Spot Lights* und eines *Point Lights* mit derselben IES-Textur (eigene Darstellung)

Die von Unreal generierten IES-Texturen sind zweidimensional und werden um die vertikale Achse der Lichtquelle gedreht. Es entsteht also immer ein symmetrischer Lichtschein (siehe Kapitel 6.2.4).

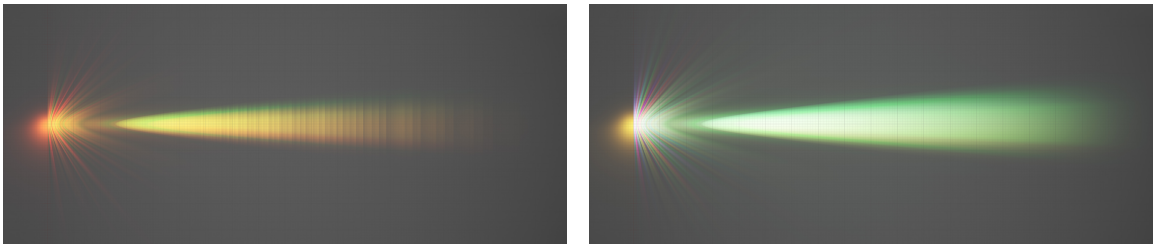
6.2.2 Farbmischung

Die Farben der Lichtquellen können über die *Light Color*-Funktion angepasst werden. Die Farbwerte sind in sRGB angegeben. Beim Ändern der Lichtfarbe bleibt die Helligkeit der Lichtquellen konstant. Bei realen Scheinwerfern würde diese

[7] *Unreal Engine 4 Documentation* 2019: *IES Light Profiles*

allerdings, entweder durch Hinzufügen/Entfernen von Filterscheiben oder durch das Weg-/Dazuschalten von Lichtquellen, verändert werden.

Für LED-Scheinwerfer mit RGB(W)-Farbmischung gibt es den Lösungsansatz, statt einer Lichtquelle drei beziehungsweise vier Lichtquellen einzusetzen. Anstelle von einer Lichtstärkeverteilungskurve werden dann die IES-Profile für die jeweilige Farb-LED eingebunden. Dann kann die Farbmischung vorgenommen werden, indem jede einzelne Lichtquelle in ihrer Helligkeit verändert wird. So verändert sich automatisch auch die Gesamthelligkeit. Allerdings erhöht dies die Komplexität des Farbmischens enorm. Auch der Rechenaufwand wird so verdrei- oder vervierfacht. Wenn alle RGB-Lichtquellen auf volle Helligkeit eingestellt sind, sollte annähernd ein Weiß entstehen. Allerdings hat die resultierende additive Farbmischung einen Farbstich zu der Farbe, deren Lichtquelle eine höhere Lichtstärke hat (zum Beispiel einen Grünstich). Das ist zwar grundsätzlich auch in der Realität ein Problem, fällt in der Simulation aber besonders stark auf.



(a) Orangene Farbmischung (Rot auf 100 %, Grün auf 25 %)

(b) „Weiße“ Farbmischung (alle Lichtquellen auf 100 %)

Abbildung 6.8: Farbmischung mit drei Lichtquellen mit jeweiligen RGB-IES-Profilen (eigene Darstellungen)

Eine Möglichkeit, bei nur einer Lichtquelle die Helligkeit mit der Farbänderung anzupassen, ist, die Lichtstärke manuell anzugeben. Die Lichtstärken eines Scheinwerfers sollten aus dem Datenblatt oder aus den IES-Profilen entnehmbar sein. Diese Werte können in der Lichtquelle angegeben werden (wenn nicht die Helligkeitsübernahme aus dem IES-Profil aktiviert ist). Allerdings können nur die Lichtstärken herangezogen werden, die bekannt sind. Also in der Regel volle Rot-, Grün- oder Blauwerte. Andere Farbmischungen müssten berechnet oder geschätzt werden.

Zur reinen Visualisierung einer Lichtshow kann auf diese Helligkeitsanpassungen möglicherweise auch verzichtet werden.

6.2.3 Spotlights und Gobos

Gobos können, wie die Bilder beim Videomapping, als *Light Function* eingebaut werden. Dafür muss das Gobo-Muster als Schwarz-Weiß-Bild vorliegen. Dieses wird als Textur der *Emissive Color* zugeordnet. Einer gerichteten Lichtquelle (*Spot Light*) wird dann dieses Material zugewiesen. Es müssen *Spot Lights* verwendet werden, da die *Light Function* auf punktförmigen Lichtquellen sphärisch gemappt wird. Weitere Einstellungen müssen nicht vorgenommen werden.

Die Größe des Gobos richtet sich an dem äußeren Lichtkegel (*Outer Cone Angle*, in Grad) des *Spot Lights* aus. Dieser beschränkt auch die Breite des Lichtkegels. Wenn er zu klein gezogen wird, wird also das IES-Profil verfälscht

Es gibt zwei Möglichkeiten, das Gobo zu rotieren. Die Textur des Gobo-Materials kann mit dem *CustomRotator*-Node gedreht werden. Damit würden sich allerdings alle Instanzen dieses Gobo-Materials gleichzeitig drehen. Alternativ kann sich die Lichtquelle um die eigene Achse drehen. Dies kann unabhängig für jede Lichtquelle vorgenommen werden.

Tiefenschärfe von *Light Functions* kann in der Unreal Engine nicht simuliert werden. Die entstehenden Muster sind immer scharf, egal in welchem Abstand zur Lichtquelle sie stehen. Besonders bei Gobos ist Unschärfe aber ein gewünschter Effekt. Diese lässt sich – physikalisch unkorrekt – simulieren, indem die Textur selber unscharf gezeichnet wird. In der Unreal Engine geht dies mit der *SpiralBlur-Texture*-Funktion. Diese Funktion ist allerdings extrem leistungsaufwendig und sollte deswegen nicht verwendet werden.

Alternativ kann das Schwarz-Weiß-Bild des Gobo-Musters in einem Bildbearbeitungsprogramm unscharf gezeichnet und als eigene *Light Function*-Textur implementiert werden (siehe Abbildung 6.9, rechts). Dies erhöht zwar den Rechenaufwand nicht, verhindert aber das Anpassen der Unschärfe in der Engine selbst. Allerdings besteht immer dieselbe Unschärfe, unabhängig von der Entfernung zur Lichtquelle.

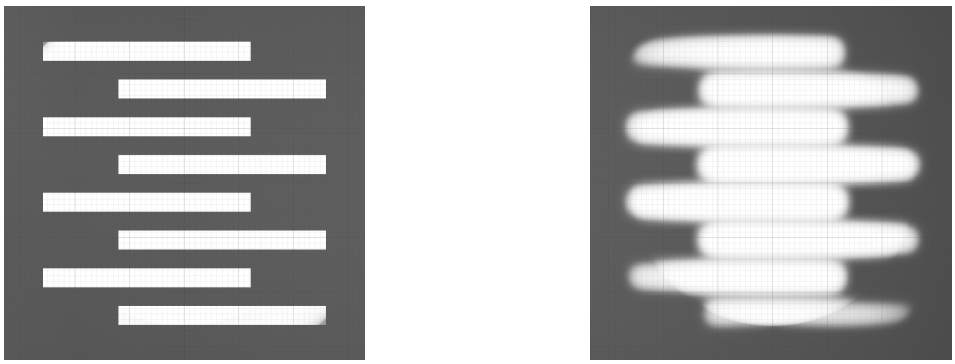


Abbildung 6.9: *Light Fuction*-Gobo ohne und mit Unschärfe (eigene Darstellung)

6.2.4 Washligths und unsymmetrische Lichtkegel

Washlights besitzen meistens eine bewegbare Linse, mit der der Austrittswinkel des Lichts variiert wird. Bei der Verwendung von IES-Profilen ist der Nachteil, dass der Lichtkegel fest vorgeben ist. Zwar sind oft mehrere Profile für verschiedene Winkel verfügbar, eine stufenlose Winkeländerung ist so trotzdem nicht möglich.

Ersatzweise kann das IES-Profil mit dem größten Winkel verwendet werden. Der Lichtkegel wird dann mit dem *Outer Cone Angle* der Lichtquelle abgeschnitten. Das Ergebnis ist zwar nicht physikalisch korrekt, erfüllt jedoch seinen Zweck.

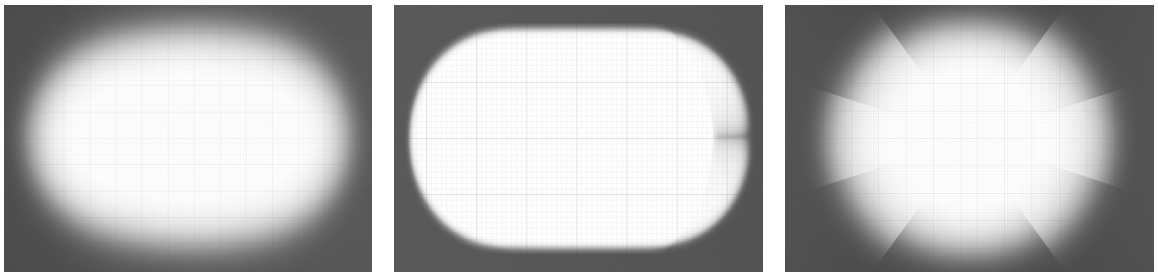
Problematischer ist es allerdings mit einem asymmetrischen Lichtkegel, wie es bei den meisten Washlights der Fall ist. Die Unreal Engine generiert aus den IES-Profilen zweidimensionale Texturen. Diese Texturen werden um die vertikale Achse der Lichtquelle rotiert. So entsteht immer ein symmetrischer Lichtkegel. Lichtquellen können auch nicht gestaucht werden. Allerdings gibt es zwei beziehungsweise drei Lösungsansätze.

Wenn der Scheinwerfer in mehrere Lichtquellen unterteilt wird, können diese für ein asymmetrisches Licht sorgen. Statt einer Lichtquelle werden zum Beispiel zehn Lichtquellen eingesetzt. Diese können alle mit demselben IES-Profil versehen werden, verwenden aber nur je zehn Prozent der Helligkeit. Werden diese Leichtquellen nun in der Position verschoben und in einer Reihe auf dem hypothetischen Scheinwerfer verteilt, entsteht eine leicht asymmetrische Lichtverteilung. Der Effekt wird allerdings nur relativ gering sein, da die einzelnen Lichtquellen nur wenige Zentimeter voneinander entfernt sind. Um den Effekt zu verstärken, müssen die äußeren Lichtquellen nach außen geschwenkt werden. Der Winkel muss allerdings geschätzt werden.

Diese Methode ermöglicht nur eine Näherung an den eigentlichen Lichtkegel und ist mit viel Aufwand und Rechenleistung verbunden.

Eine anderer Ansatz ist, den Lichtkegel mit einer *Light Function* unsymmetrisch zu beschränken. Dadurch entsteht ein Lichtfleck, der dem eines asymmetrischen Scheinwerfers nachempfunden werden kann. Der Lichtkegel ist dann aber, anders als in der Realität, gerade abgeschnitten.

Ein dritter Ansatz ist, diese beiden Techniken zu verbinden. So wie Lichtstärkeverteilungskurven in Scheiben unterteilt sind, können auch die Unreal Lichtquellen in Scheiben unterteilt werden, wenn *Light Functions* verwendet werden. Der Scheinwerfer wird wieder in mehrere, in diesem Beispiel zehn, Lichtquellen unterteilt. In jeder Lichtquelle wird, mit einer *Light Function*, alles bis auf ein 36° Kreisausschnitt abgeschnitten. Jede Lichtquelle wird um 36° gedreht und bekommt ein IES-Profil für den jeweiligen Ausschnitt zugeordnet. So entsteht näherungsweise die physikalisch korrekte Lichtstärkeverteilungskurve.



(a) durch Aufteilen, Verteilen und Schwenken **(b)** durch Abschneiden mit *Light Functions* **(c)** durch Aufteilen in Kreisausschnitte

Abbildung 6.10: Asymmetrische Lichtkegel (eigene Darstellung)

6.2.5 Beamlights und Haze

Beamlights werden hauptsächlich dafür eingesetzt, einen Lichtstrahl in der Luft zu erzeugen. Um diesen sichtbar zu machen, wird in der echten Welt, wie auch in der Unreal Engine, Nebel oder Haze benötigt. In Unreal wird dafür ein Nebel-Element (*Exponential Height Fog*) verwendet. Dieses Element simuliert einen Nebel, der nach oben hin abnimmt. Es können diverse Einstellungen vorgenommen werden, wie Farbe, Reichweite, Dichte und viele weitere. Um Rechenleistung zu sparen, sollte der Startabstand und die Reichweite beschränkt werden, da der Nebel selber hier nicht interessant ist. Um Lichtkegel sichtbar zu machen, ist die Einstellung *Volumetric Fog* relevant. Standardmäßig ist diese Option deaktiviert. Hier können unter anderem noch einmal Intensität, Farbe und Sichtweite eingestellt werden.

Der Lichtkegel, der im Exponential Height Fog entsteht, verwendet dabei nicht die Information aus dem IES-Profil. Eine Lichtquelle wird genau den gleichen Lichtschein mit oder ohne einer IES-Textur erzeugen. Lediglich die Lichtstärke kann aus dem IES-Profil übernommen werden.[8]

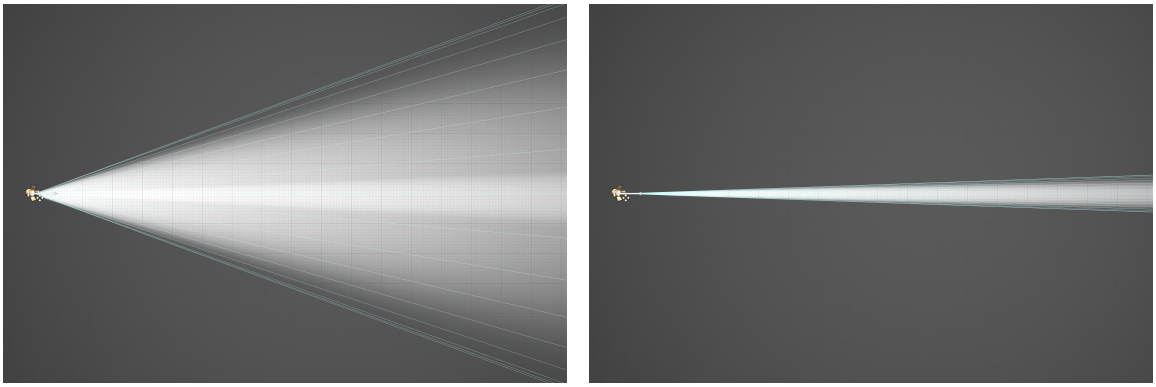
Um dennoch einen annähernd korrekten Lichtkegel zu erstellen, kann der äußere Kegel (*Outer Cone*) einer *Spot Light*-Lichtquelle verwendet werden. Dazu muss der Winkel des Kegels an den Abstrahlwinkel des simulierten Scheinwerfers angepasst werden. Dazu kann eine Winkelangabe aus dem Datenblatt verwendet werden. Alternativ kann das *Spot Light* parallel zu einer Fläche positioniert und dann der Kegelwinkel an den entstehenden Lichtkegel angepasst werden.

In Abbildung 6.11a und b ist mittig der Lichtschein der IES-Textur auf dem Boden zu sehen. Drumherum ist der Lichtschein in der Luft zu erkennen, dieser wird vom *Outer Cone*-Kegel (hellblaue Linien) begrenzt.

Die Reichweite wird, wenn aktiviert, nach dem fotometrischen Entfernungsgesetz abfallen, kann aber auch mit dem *Attenuation Radius* (in cm) beschränkt werden.

[8] *Unreal Engine 4 Documentation* 2019: *Exponential Height Fog User Guide*

Allerdings wird der Lichstrahl nicht unterbrochen, wenn er auf ein Objekt trifft (siehe Abbildung 6.16 auf Seite 51).



(a) mit weitem Kegelwinkel **(b)** mit angepasstem Kegelwinkel

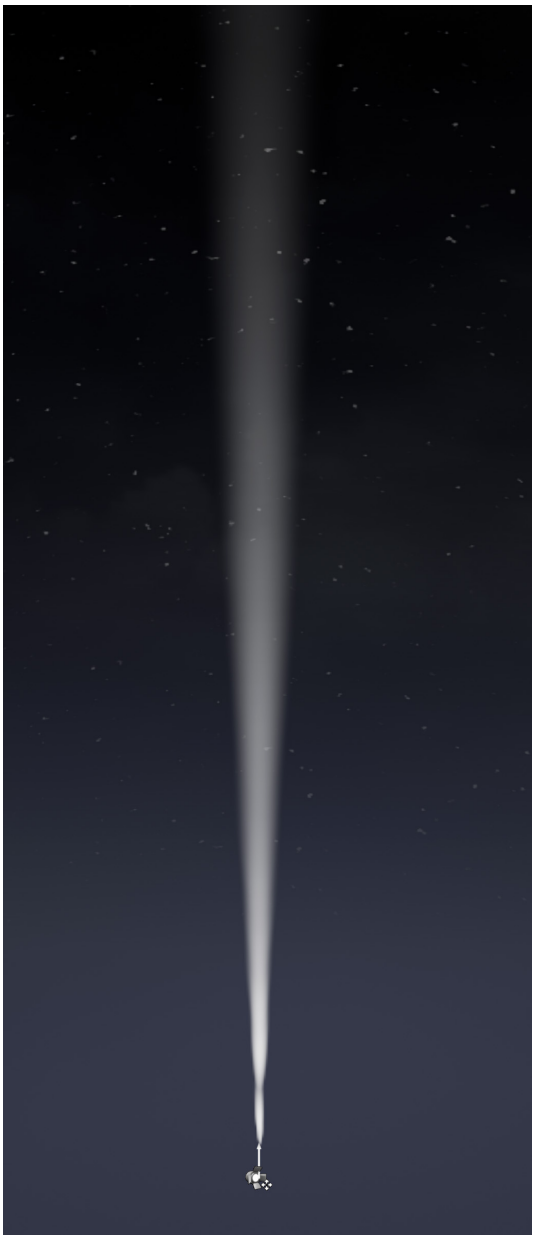
Abbildung 6.11: Lichtschein mit IES-Textur auf einer Fläche und Lichtkegel in der Luft (eigene Darstellungen)

Die Qualität des *Volumetric Fogs* hängt von der eingestellten *Volume Texture Resolution* ab. Diese wird in den *Engine Shadow Scalability*-Optionen eingestellt.[9] Dadurch werden zwei für die Lichtkegel relevante Werte geändert. Diese können aber auch direkt per Kommandozeilen-Befehl verändert werden, so hat der Benutzer mehr Kontrolle über die Qualität. Tabelle 6.1 listet beide Befehle, Effekte und den verwendeten Wert auf. Bei beiden Einstellungsgrößen erhöht sich, mit steigender Qualität, auch die Rechenleistung. Für die *GridPixelSize* wäre ein kleinerer Wert (zum Beispiel 2) optimaler, die Leistungsanforderungen steigen aber zu stark.

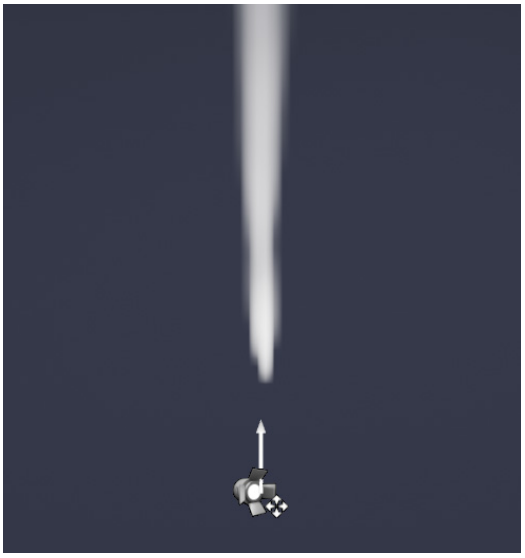
Befehl	Effekt	Wert in diesem Projekt
r.VolumetricFog.GridPixelSize	kleinere Werte führen zu höherer Qualität	4
r.VolumetricFog.GridSizeZ	höhere Werte führen zu höherer Qualität	64

Tabelle 6.1: Kommandozeilen-Befehle zur Volumetric Fog Qualitätseinstellung (*Unreal Engine 4 Documentation* 2019: Volumetric Fog)

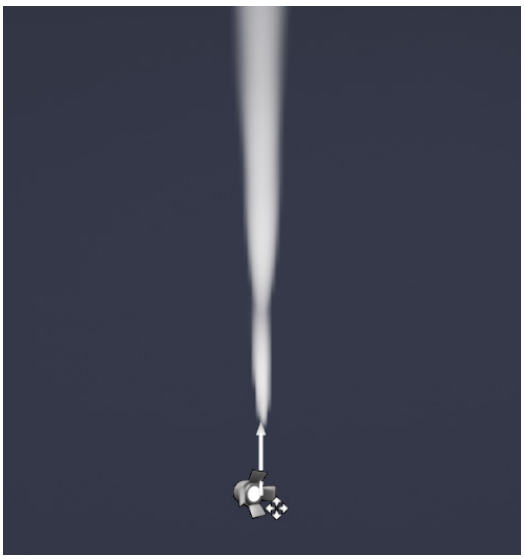
[9] *Unreal Engine 4 Documentation* 2019: *Scalability Reference*



(a) Abnehmender Lichtschein nach dem fotometrischen Entfernungsgesetz



(b) Normale Qualität



(c) Verbesserte Qualität

Abbildung 6.12: Lichtscheine in der Luft (eigene Darstellungen)

6.3 Animation

Jedes Element in der Unreal Engine Spielumgebung kann um je drei Achsen in der Position, Rotation und Skalierung transformiert werden. Um diese Transformationen über den zeitlichen Raum hinweg zu verändern, bietet die Unreal Engine diverse Möglichkeiten. Die zwei verwendeten Funktionen werden im folgenden kurz vorgestellt.

Sequencer

Der *Sequencer Editor* ist ein Tool zum Erstellen und Bearbeiten von filmischen Sequenzen in der Unreal Engine. Wenn eine *Level Sequence* zum Projekt hinzugefügt wird, erhält der Benutzer Zugriff auf die *Sequencer Editor* Zeitleiste. Dieser Editor ist ein vollwertiges Animationstool, in dem fast alle Variablen eines Unreal Levels animiert werden können.[10]

Für die Lichtshow-Simulation sind hauptsächlich die Roll-, Nick- und Gierwinkel, die Helligkeit und die Farbe der Lichtquelle relevant. Um ein Objekt animieren zu können, muss dieses erst zum Editor hinzugefügt werden. Bei einer Lichtquelle steht anschließend direkt die Intensität (Helligkeit) und der Farbwert zum Animieren zur Verfügung. Wenn die Helligkeit aus der IES-Textur übernommen wird, muss die *IES Intensity Scale* verändert werden, um die Helligkeit zu beeinflussen. Diese Variable und die Transformationsvariablen können mit zur Zeitleiste hinzugefügt werden.

Die Werte können nun nach Belieben angepasst und animiert werden. Dafür stehen diverse Interpolationsverfahren zur Verfügung. Standardmäßig werden alle Variablen, auch die Farbwerte, kubisch interpoliert. Zum Drehen von Gobos kann hier der Rollwinkel verändert werden. So wird zwar die ganze Lichtquelle gerollt, da die IES-Texturen aber symmetrische Lichtkegel verursachen, hat das denselben Effekt, als wenn nur das Gobo gedreht werden würde.

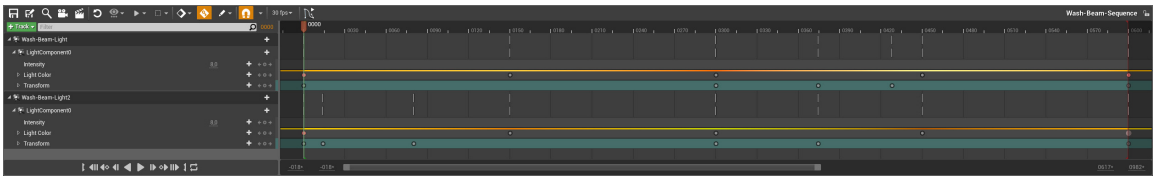


Abbildung 6.13: Benutzeroberfläche des *Sequencer Editors* (eigene Darstellung)

Animationsspuren können kopiert, auf andere Objekte angewandt und dort wieder bearbeitet werden. So lassen sich schnell Choreografien erstellen.

[10] *Unreal Engine 4 Documentation* 2019: *Sequencer Editor*

Zum Animieren einer kompletten Lichtshow können einzelne Sequenzen erstellt und diese in eine Master Sequenz geladen werden. Dann können alle Sequenzen, auch mehrfach, aneinander gereiht werden. Außerdem gibt es die Möglichkeit, Kameraeinstellungen und -fahrten zu animieren und alles als Videodatei zu exportieren.

Material

Im Material Editor steht das *Time*-Node zur Verfügung. Dieses zählt die Zeit seit dem Starten des Spiels beziehungsweise des Unreal Editors in Sekunden. Die Geschwindigkeit kann beeinflusst werden, indem der Zeitwert mit einer Variabel multipliziert wird.

Mit diesem stetig steigendem Wert können Texturen animiert werden. Zum Beispiel kann damit die Textur eines Gobos gedreht werden, indem der Wert als Drehwinkel angegeben wird. Häufig wird mit dem Zeitwert ein Sinus generiert. Wird der Sinuswert als Helligkeit der Licht-Textur verwendet, entsteht ein pulsierender Lichteffekt.[11]

Eventuell von Nachteil ist, dass alle Lichtquellen, die dieses Material verwenden, die identische Animation haben. Um dies zu umgehen, kann für jede Instanz das Material dupliziert und der Zeitwert verändert werden.

6.4 Umgebungsbeleuchtung

In der Unreal Engine gibt es eine Himmelskugel (*Sky Sphere Blueprint*), die verschiedene Tages- und Nachthimmel darstellen kann. Zusätzlich zu diesem *Blueprint* wird noch eine gerichtete Lichtquelle benötigt. Die Drehung dieser Lichtquelle stellt die Position der Sonne dar. Steht diese zum Beispiel bei $-23°$ (über dem Horizont), zeigt der Himmel einen typischen Taghimmel an. Ist die Lichtquelle parallel zum Boden, wird ein Sonnenuntergang dargestellt. Videomapping und Lichtshows werden meistens in der Dämmerung oder bei Nacht durchgeführt. Dafür wird die Lichtquelle auf einen Winkel zwischen 30° und 150° (unter dem Horizont) eingestellt. Der Himmel zeigt dann eine Dämmerung beziehungsweise einen Nachthimmel an.

In diesem Winkel sollte die Lichtquelle allerdings keine Helligkeit haben, weil das Licht sonst von unten durch den Boden scheint. Stattdessen wird eine zweite gerichtete Lichtquelle hinzugefügt, die den Mond darstellt. Diese kann nach Belieben positioniert werden. Als Beleuchtungsstärke kann eine typische Vollmond-Situation mit ca. 0,27 lx gewählt werden. Falls die Gesamtsituation zu dunkel ist, kann mit einem *Sky Light*[12] die Szene etwas erhellt werden, da die *Sky Spehere* selber kein Licht ausstrahlt. Hier liegen die Werte für Nachtsituationen zwischen $1\,\mathrm{cd/m^2}$ und $5\,\mathrm{cd/m^2}$.

[11] *Unreal Engine 4 Documentation* 2019: *Constant Expressions - Time*
[12] *Unreal Engine 4 Documentation* 2019: *Sky Light*

Eventuell notwendig ist das Verändern der Einstellungen in einem *Post Process* Modul. Hier ist ein relativ starkes Blooming voreingestellt. Dieses sollte entweder deaktiviert oder zumindest reduziert werden.

Hier kann aber auch eine Helligkeitsanpassung (*Exposure*) eingestellt werden. Mit dieser wird simuliert, wie sich das menschliche Auge an Helligkeiten gewöhnt. Dies ist nützlich, wenn die Simulation sowohl bei Tageslicht als auch bei Nachtlicht durchgeführt werden soll. Allerdings verschwindet gegebenenfalls der Himmel, wenn eine helle (Tages-) Umgebungsbeleuchtung mit der *Exposure*-Einstellung kompensiert wird (siehe Abbildung 6.14). Das kann mit zwei *Post Process* Modulen behoben werden, wenn ein Modul die Helligkeit des Himmels und das andere die Umgebungshelligkeit steuert.[13] Das erfordert aber die doppelte Rechenleistung.

Durch das Verwenden von physikalisch korrekten Größen für die Umgebungsbeleuchtung entsteht ein realitätsnaher Eindruck der Helligkeitssituation zu verschiedenen Tageszeiten (siehe Abbildungen 6.14 bis 6.16). So verblassen das Videomapping und die Scheinwerferstrahlen bei einer helleren Umgebungsbeleuchtung schnell.

Abbildung 6.14: Sonnenstand bei $-60°$ und $19\,000\,\mathrm{lx}$ mit Tag-Himmel (eigene Darstellung)

[13] *Unreal Engine 4 Documentation* 2019: *Post Process Materials*

Abbildung 6.15: Sonnenstand bei $-15°$ und $3\,\mathrm{lx}$ mit Abend-Himmel (eigene Darstellung)

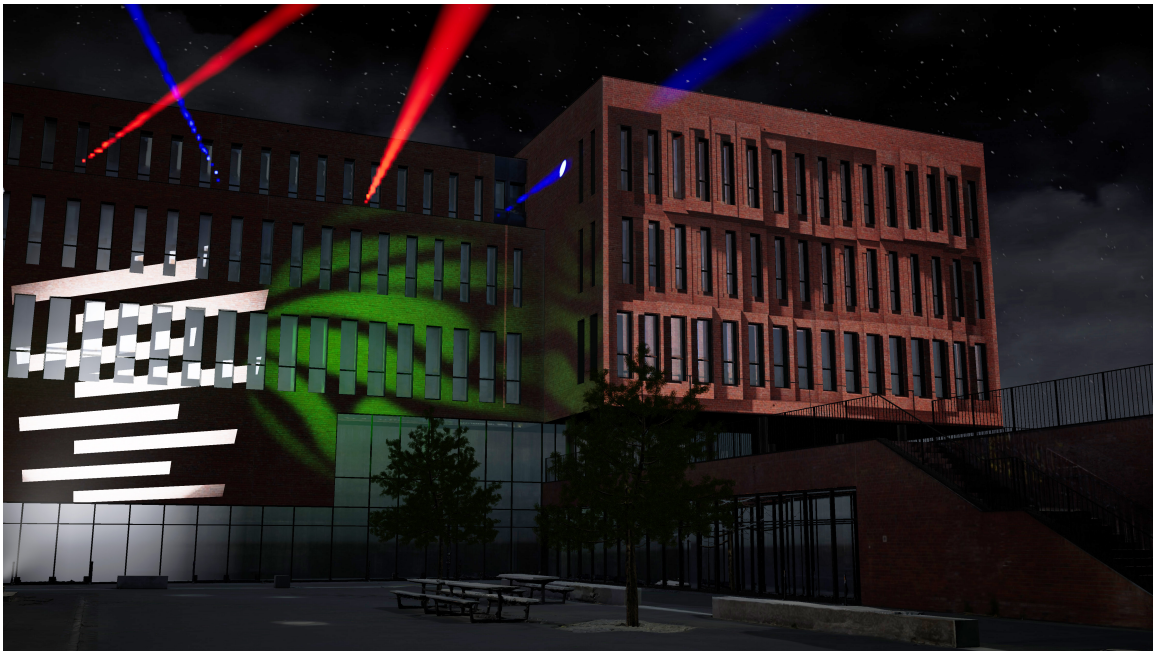


Abbildung 6.16: Sonnenstand bei $90°$ und $0{,}27\,\mathrm{lx}$ mit Nacht-Himmel (eigene Darstellung)

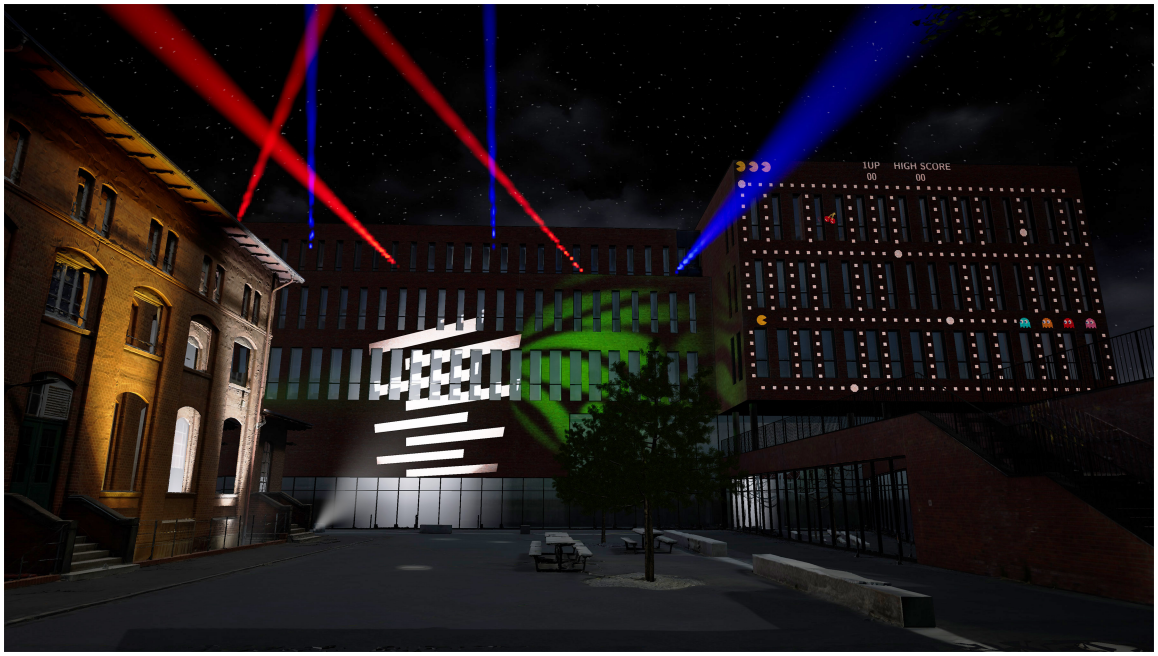


Abbildung 6.17: Fertiggestellte Szene mit Lichtshowelementen und Videomapping mit Blick vom Innenhof (eigene Darstellung)

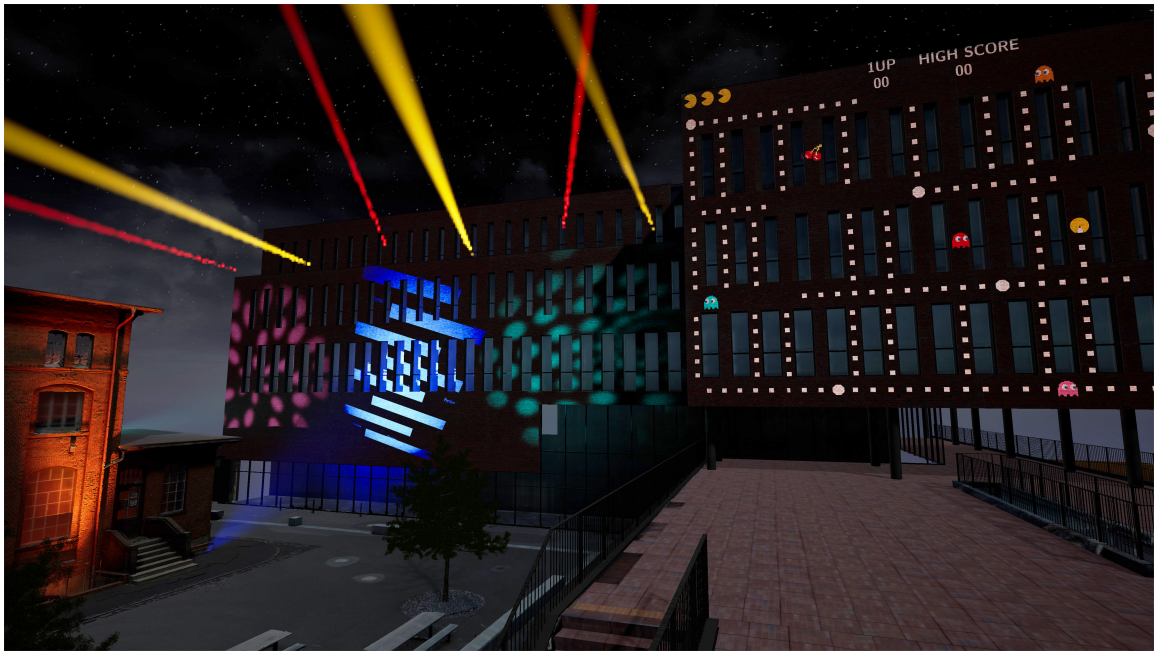


Abbildung 6.18: Fertiggestellte Szene mit Lichtshowelementen und Videomapping mit Blick von der Terrasse (eigene Darstellung)

7 Fazit

7.1 Zusammenfassung

Das Ziel dieser Bachelorarbeit war die Untersuchung der Realisierbarkeit einer Simulation von Lichtshows und Videomapping mit einem 3D-Modell in einer Game-Engine. Das 3D-Modell wurde dabei mit Fotogrammetrie und Laserscanning erstellt.

Die Simulation in der Unreal Engine war ein voller Erfolg. Sowohl Videomapping als auch die wichtigsten Lichtshow-Elemente lassen sich, sobald alles einmal eingerichtet ist, unkompliziert umsetzen.

Für alle Lichtquellen können fotometrische Einheiten in Candela oder in Lux angegeben werden. Omni- und Spot-Lights können Lichtstärkeverteilungskurven im IES-Format zugewiesen bekommen. So können sämtliche Scheinwerfer, deren IES-Profil vorhanden ist, physikalisch korrekt simuliert werden.

Um Gobos zu simulieren, werden die Gobo-Muster als Schwarz-Weiß-Bilder einer *Light Function* zugeordnet. Ein *Spot Light* mit dieser Textur projiziert das Muster dann wie ein Gobo. Allerdings haben die *Light Function*-Muster keine Tiefenschärfe, das Muster ist also immer gestochen scharf. Um dennoch – physikalisch unkorrekte – Unschärfe zu erzeugen, muss die Textur selber unscharf gezeichnet werden.

Damit Lichtstrahlen in der Luft sichtbar sind, wird ein *Exponential Height Fog*-Nebel benötigt. Dieser verbraucht allerdings viel Rechenleistung. Er sollte also nur aktiviert werden, wenn er benötigt wird, und auch dann nur von den Lichtquellen, deren Strahlen in der Luft sichtbar sein sollen. Leider werden auch keine Lichtkurven aus den IES-Profilen verwendet. Der Lichtkegel muss stattdessen in Grad angegeben werden. Immerhin fällt die Intensität des Lichtstrahls mit dem fotometrischen Entfernungsgesetz ab. Die Qualität des Lichtkegels lässt sich manuell einstellen, der benötigte Rechenaufwand steigt aber schnell an.

Problematischer ist das Umsetzen von asymmetrischen Lichtstärkeverteilungen, wie es bei Washlights oft der Fall ist. Es gibt aber Lösungsansätze, welche eine mehr oder weniger realistische Umsetzung ermöglichen. Der Scheinwerfer wird dafür mit mehreren Lichtquellen simuliert. Diese werden entweder asymmetrisch angeordnet, oder mit einer *Light Function* in Kreisausschnitte mit unterschiedlichen IES-Profilen unterteilt. Beide Varianten erfordern einiges an Aufwand und Rechenleistung. Die entstehenden Lichtkegel könnten so aber der Realität angenähert werden. Zur reinen Visualisierung von Lichtshows reichen diese Methoden aber auf jeden Fall aus.

Die Lichtfarbe kann in der Lichtquelle in sRGB angegeben werden. Soll sich die Helligkeit zusammen mit der Farbe ändern, gibt es zwei Möglichkeiten. Entweder wird der Helligkeitswert entsprechend händisch angepasst oder der Scheinwerfer wird

in drei Lichtquellen unterteilt, die jeweils ein IES-Profil für Rot, Grün oder Blau bekommen. Dann wird die Farbe geändert, indem die Helligkeit der entsprechenden Lichtquelle verändert wird.

Mit dem *Sequencer* lassen sich die Scheinwerfer bewegen, Farbwerte verändern und Gobos drehen. So kann eine komplette Lichtshow inszeniert werden.

Für das Videomapping bietet sich am ehesten die Projektion per *Light Function* an. Zwar werden drei Lichtquellen für farbige Projektionen benötigt, der Einfluss auf die Rechenleistung ist aber nicht essentiell und die Vorteile überwiegen deutlich. Durch das Verwenden von Lichtquellen ergibt sich auch hier die Möglichkeit, fotometrische Einheiten, das fotometrische Entfernungsgesetz und – falls vorhanden – IES-Profile zu nutzen. So entstehen realitätsnahe Projektionshelligkeiten.

Objekte, die sich im Strahlengang befinden, werfen einen Schatten und die Probleme, die beim Projizieren auf Gebäudekanten und Fenstern entstehen, werden korrekt dargestellt. Damit perspektivische Verzerrungen kompensiert werden können, wurde im Rahmen des Projektes in der *Light Function* unter anderem eine Option zum mathematischen Vorentzerren erstellt. So muss nur noch angegeben werden, wie stark die Verzerrung ist. Es können sowohl Bilder als auch Videos mit beliebigen Seitenverhältnissen für die Projektion verwendet werden.

Für die Umgebungsbeleuchtung können ebenfalls Candela oder Lux angegeben werden. Ein Himmel wird entsprechend des Sonnenwinkels automatisch generiert. So lassen sich schnell unterschiedliche Tagessituationen erstellen. Durch das Verwenden von realistischen Umgebungshelligkeiten erscheinen auch das Licht der Scheinwerfer verhältnismäßig hell oder dunkel.

Das Abspielen eines Unreal Projektes in VR funktioniert in den entsprechenden Leveln problemlos. Gegebenenfalls muss der Teleportationsbereich einmalig manuell erstellt werden. Da sonst keine Interaktion mit Spielelementen vorkommt, muss nichts weiter eingerichtet werden.

Das Verwenden von Laserscanning und Fotogrammetrie zum Erstellen der 3D-Szene hat diverse Vor- und Nacheile. Vorteilhaft ist vor allem die zentimetergenaue Form und Position der Objekte. Allerdings haben die Meshes eine schlechte Topologie und unebene Oberflächen. Bei verhältnismäßig kleinteiligen Objekten funktionieren Laserscanning und Fotogrammetrie nur beschränkt. Anhand der vorhandenen Daten lassen sich allerdings einfach neue Objekte modellieren. Die Form kann dann perfekt an die alten Meshes angepasst werden.

Die Texturen sind durch die Fotogrammetrie sehr detailgetreu und hochwertig. Bei der Größe der Szene muss aber ein Kompromiss mit der Texturauflösung eingegangen werden, damit der Rechenaufwand nicht zu hoch wird. Die Generierung von anderen Maps, wie Normal- oder Roughness-Maps wird durch die Größe der Szene und die automatisch generierten UVs erschwert.

7.2 Ausblick

Die Ergebnisse zeigen, dass sich die Kombination von Fotogrammetrie und Laserscanning zum Erstellen einer detailgetreuen 3D-Szene eignet. Allerdings müssen ein paar Kompromisse eingegangen werden. Die Unreal Engine bietet bereits einige Funktionen zum Simulieren von Videomapping und Lichtshows. Die wenigen Limitationen lassen sich, mit Abstrichen in der physikalischen Exaktheit, umgehen. Das größte Problem stellen hier bisher unsymmetrische Lichtstärkeverteilungskurven dar. Die beschriebenen Lösungsansätze könnten in zukünftigen Arbeiten noch weiter untersucht werden.

Das in dieser Arbeit entwickelte Unreal Projekt bietet eine solide Grundlage zum Simulieren, Testen und Präsentieren von Lichtshows und Videomapping am Campus Finkenau. Es bieten sich außerdem viele Möglichkeiten zur Weiterentwicklung auch in anderen Anwendungsbereichen an.

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

Ärztekammer Hamburg. „Hamburger Ärzteblatt“. In: *04/2015* (2015).

FARO. *FARO Laser Scanner Focus S 350 technisches Datenblatt*. Techn. Ber. 2016.

FARO. *Faro Focus 3D Laser Scanner*. 2019. URL: https://www.faro.com/de-de/produkte/bausektor-bim-cim/faro-focus/ (besucht am 03. 06. 2019).

Looman, T. *VR Template Guide for Unreal Engine 4*. 2018. URL: https://www.tomlooman.com/vrtemplate/ (besucht am 19. 07. 2019).

Luhman, Thomas. *Nahbereichsphotogrammetrie Grundlagen - Methoden - Beispiele*. 4. Auflage. 2018.

Luhmann, Thomas, Schumacher, Christina und Oldenburger 3D-Tage (2016: Oldenburg). *Photogrammetrie - Laserscanning - Optische 3D-Messtechnik: Beiträge der Oldenburger 3D-Tage 2016*. 2016.

Mason, Austin. *The Haskins Society - 3D Photogrammetry with PhotoScan*. 2015. URL: https://thehaskinssociety.wildapricot.org/photogrammetry (besucht am 25. 05. 2019).

Mettenleiter, Markus. u. a. *Laserscanning – Phasenbasierte Lasermesstechnik für die hochpräzise und schnelle dreidimensionale Umgebungserfassung*. 2015.

Möllring, Lara. *Impulslaufzeitverfahren zur Streckenmessung*. 2018. URL: https://messpanda.de/strecken/impulsverfahren/ (besucht am 03. 06. 2019).

Möllring, Lara. *Phasenvergleichsverfahren zur Streckenmessung*. 2018. URL: https://messpanda.de/strecken/phasenvergleichsverfahren/ (besucht am 03. 06. 2019).

Przybilla, H.-J. *Elektronische Distanzmessung und Tachymetrie*. 2019. URL: http://www.przybilla.biz/bv/Skripte/BV7-Elektronische_Distanzmessung_und_Tachymetrie.pdf.

Remde, Christopher. *GitHub - ChristopherRemde/Projection_shadow*. 2019. URL: https://github.com/ChristopherRemde/Projection_shadow (besucht am 28. 05. 2019).

SGM. *G-4 Wash-Beam Micro-Fresnel LED from SGM Light*. 2019. URL: https://sgmlight.com/Default.aspx?ID=2410&ProductId=18497 (besucht am 07. 07. 2019).

Spektrum Akademischer Verlag, Heidelberg. *Photogrammetrie*. 2000. URL: https://www.spektrum.de/lexikon/geowissenschaften/photogrammetrie/12220 (besucht am 16. 06. 2019).

Unreal Engine 4 Documentation. 2019. URL: https://docs.unrealengine.com/en-US/ (besucht am 27. 07. 2019).

Constant Expressions - Time: /Engine/Rendering/Materials/ExpressionReference/Constant/#time

Decals: /Resources/ContentExamples/Decals/

Exponential Height Fog User Guide: /Engine/Actors/FogEffects/HeightFog/

IES Light Profiles: /Engine/Rendering/LightingAndShadows/IESLightProfiles/

Light Functions: /Engine/Rendering/LightingAndShadows/LightFunctions/

Physical Lighting Units: /Engine/Rendering/LightingAndShadows/PhysicalLightUnits/

Post Process Materials: /Engine/Rendering/PostProcessEffects/PostProcessMaterials/

Scalability Reference: /Engine/Performance/Scalability/ScalabilityReference/

Sequencer Editor: /Engine/Sequencer/

Sky Light: /Engine/Rendering/LightingAndShadows/LightTypes/SkyLight/

Texture Support and Settings: /Engine/Content/Types/Textures/SupportAndSettings/

Volumetric Fog: /Engine/Rendering/LightingAndShadows/VolumetricFog/.

Unreal Engine Forum. *Projection Mapping technique to projects video files on objects? - Unreal Engine Forums*. 2015. URL: https://forums.unrealengine.com/development-discussion/rendering/103582-projection-mapping-technique-to-projects-video-files-on-objects (besucht am 08. 07. 2019).

Wunderlich, Prof Thomas u. a. *Objektivierung von Spezifikationen Terrestrischer Laserscanner – Ein Beitrag des Geodätischen Prüflabors der Technischen Universität München*. Bd. 20. 02. 2013.

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht.

Ort, Datum Mirco Hülsemann