# HoloLens for BIM

**Bachelorthesis**


| | |
|---|---|
| Studiengang: | BSc in Informatik |
| Autor: | Lukas Knöpfel, Vincent Genecand |
| Betreuer: | Prof. Marcus Hudritsch |
| Experten: | Dr. Harald Studer |
| Datum: | 18. Januar 2018 |

# Management Summary

Unsere Bachelorarbeit wurde mit der Augmented Reality Brille von Microsoft, der HoloLens, realisiert. Die HoloLens ist eine Brille, welche die reale Umgebung nicht versteckt, sondern sie lediglich mit digitalen Informationen ergänzt. Wie es der Name «Augmented Reality» sagt, wird die eigene Realität erweitert.

Das Ziel der Arbeit ist es, diese Technologie zu benutzen, um sich in einem bekannten Gebäude zu orten. Dieses Orten soll dann genutzt werden, um die Umgebung mit Building Information Modeling (BIM) Daten anzureichern. BIM ist eine Methode, bei welcher mit Software alle relevanten Daten eines Gebäudes in ein Modell kombiniert und erfasst werden. Dies können nebst den Raumgrössen auch z.B. die Position von elektrischen Leitungen in den Wänden sein. Das Problem dabei ist, dass die HoloLens über keine Technologie verfügt, welche es ihr erlaubt, sich innerhalb eines Gebäudes zu orten. Sie ist lediglich in der Lage, sich im Verhältnis zu platzierten Hologrammen und gescannten Umgebungen zu positionieren.

Unsere Arbeit macht diese Funktionalität möglich, indem wir unterscheiden, welcher Teil der gescannten Daten einen Raum repräsentiert und was zur restlichen Umgebung gehört. Dazu filtern wir die Flächen heraus, welche Wände und/oder Fenster sind. Dies gibt uns die Masse eines Raumes, welche wir mit einer Sammlung von bekannten Raum Modellen vergleichen können. Diese bekannten Räume sind 3D-Modelle, welche wir anhand von Plänen und eigenen Messungen erstellt haben. Bei einer konkreten Anwendung müssten das Ziel-Gebäude und die zusätzlichen Informationen in Form eines 3D-Modelles zur Verfügung stehen.

Das Resultat ist eine HoloLens-App, welche den Standort der Brille in einem Gebäude erkennt. In anderen Worten, sie weiss wie gross der Raum ist und wo sich allenfalls Fenster befinden. Mit diesen Angaben kann die App unter den bekannten Räumen ein Modell finden, welches mit den Eigenschaften des aktuellen Raumes übereinstimmt. Falls das Modell zusätzliche Eigenschaften des Raumes besitzt, wie z.B. versteckte Leitungen, so können diese nun im Sichtfeld des Betrachters an der korrekten Stelle eingeblendet werden.

# Inhaltsverzeichnis


| | | |
|---|---|---|
| | Management Summary | 2 |
| | Inhaltsverzeichnis | 3 |
| 1 | Einleitung | 5 |
| 2 | Die HoloLens | 6 |
| | 2.1 Display / Optics | 7 |
| | 2.1.1 Waveguides | 7 |
| | 2.1.2 Light Engines | 8 |
| | 2.1.3 Interpupillary Distance (IPD) | 8 |
| | 2.2 Holographic Processing Unit (HPU) | 8 |
| | 2.3 Prozessor | 9 |
| | 2.4 Sensoren | 9 |
| | 2.4.1 Inertial Measurement Unit (IMU) | 9 |
| | 2.4.2 Depth Camera | 10 |
| | 2.4.3 Environment Understanding Cameras | 10 |
| | 2.4.4 Photo / Video Camera | 10 |
| | 2.5 Audio | 10 |
| | 2.6 Zukünftige Entwicklung | 10 |
| | 2.6.1 Prozessor | 11 |
| | 2.6.2 Field of View | 11 |
| | 2.7 Hardware Spezifikationen | 12 |
| 3 | BIM / Modelle | 13 |
| | 3.1 Building Information Modeling (BIM) | 13 |
| | 3.2 Erstellung der Modelle | 13 |
| | 3.2.1 Wahl des Gebäudes und der Räume | 13 |
| | 3.2.2 Werkzeuge | 14 |
| | 3.2.3 Umsetzung | 14 |
| 4 | Mesh Analyse | 16 |
| | 4.1 Point Cloud Library (PCL) | 17 |
| | 4.1.1 Analyse / Versuche | 17 |
| | 4.1.2 Fazit | 17 |
| | 4.2 Spatial Understanding | 18 |
| | 4.2.1 Voxel | 18 |
| | 4.2.2 Fazit | 19 |
| | 4.3 Plane Finding API | 20 |
| | 4.3.1 Fazit | 20 |
| | 4.4 Entscheid | 20 |
| 5 | Die App | 21 |
| | 5.1 Gemeinsamkeiten | 21 |
| | 5.2 Übersicht über die Module | 21 |
| | 5.3 ScanManager | 22 |
| | 5.3.1 Spatial Mapping API der HoloLens | 22 |
| | 5.3.2 Implementation | 23 |
| | 5.3.3 Baking Priorisierung | 23 |
| | 5.3.4 PriorityQueue | 23 |
| | 5.3.5 Aktualisierung visualisieren | 23 |
| | 5.4 ScanProgress | 24 |
| | 5.4.1 Sensoren | 24 |
| | 5.4.2 Wann wurde genug gescannt? | 24 |
| | 5.4.3 Strategie | 24 |
| | 5.4.4 Benutzerführung | 25 |
| | 5.5 MeshAnalyzer | 26 |

| | | |
|---|---|---|
| | 5.5.1 Kategorisierung der Normalen | 26 |
| | 5.5.2 Decke und Boden | 26 |
| | 5.5.3 Wände | 26 |
| | 5.5.4 Dimensionen | 27 |
| | 5.5.5 Raumecken | 27 |
| | 5.6 RoomIdentifier | 28 |
| | 5.6.1 Virtuelle Räume | 28 |
| | 5.6.2 Identifizierung | 28 |
| | 5.6.3 Modell positionieren | 30 |
| | 5.7 MiniMap | 31 |
| 6 | Projektmanagement | 32 |
| | 6.1 Planung | 32 |
| | 6.2 Arbeitsjournal | 32 |
| | 6.2.1 Woche 1, 22. September 2017 | 32 |
| | 6.2.2 Woche 2, 29. September 2017 | 32 |
| | 6.2.3 Woche 3, 6. Oktober 2017 | 32 |
| | 6.2.4 Woche 4, 13. Oktober 2017 | 32 |
| | 6.2.5 Woche 5, 20. Oktober 2017 | 32 |
| | 6.2.6 Woche 6, 27. Oktober 2017 | 32 |
| | 6.2.7 Woche 7, 3. November 2017 | 33 |
| | 6.2.8 Woche 8, 10. November 2017 | 33 |
| | 6.2.9 Woche 9, 17. November 2017 | 33 |
| | 6.2.10 Woche 10, 24. November 2017 | 33 |
| | 6.2.11 Woche 11, 1. Dezember 2017 | 33 |
| | 6.2.12 Woche 12, 8. Dezember 2017 | 33 |
| | 6.2.13 Woche 13, 15. Dezember 2017 | 33 |
| | 6.2.14 Woche 14, 22. Dezember 2017 | 33 |
| | 6.2.15 Woche 15, 29. Dezember 2017 | 33 |
| | 6.2.16 Woche 16, 5. Januar 2018 | 33 |
| | 6.2.17 Woche 17, 12. Januar 2018 | 33 |
| 7 | Entwicklungsumgebung | 34 |
| | 7.1 Unity | 34 |
| | 7.1.1 Genutzte Version | 34 |
| | 7.2 MixedRealityToolkit-Unity | 34 |
| | 7.2.1 Genutzte Version | 34 |
| | 7.3 Visual Studio | 34 |
| | 7.3.1 Genutzte Version | 34 |
| | 7.4 Installationsanleitung | 35 |
| 8 | Schlussfolgerungen/Fazit | 38 |
| 9 | Abbildungsverzeichnis | 39 |
| 10 | Tabellenverzeichnis | 39 |
| 11 | Literaturverzeichnis | 40 |

# 1 Einleitung

In dieser Arbeit geht es darum das Potenzial der HoloLens im BIM-Umfeld auszutesten. BIM steht dabei für Building Information Modelling. Neben der Kommunikation und Präsentation eines Gebäudes in der Vor- und Projektphase könnte die HoloLens auch in fertigen Gebäuden eingesetzt werden. In wartungsintensiven Gebäuden ist z.B. das Wissen über die verdeckten Installationen wichtig für ein wirtschaftliches Facility Management. Mit der HoloLens können unsichtbare Installationen in bestehenden Räumen sichtbar gemacht werden. Ein wichtiger Teilaspekt davon ist die korrekte Lokalisierung innerhalb eines Gebäudes, von dem die HoloLens nur digitale Plandaten besitzt und noch nie vor Ort war.

Die Arbeit kann auf GitHub gefunden werden: https://github.com/shylux/HoloLens-BIM

# 2 Die HoloLens

Microsoft führt mit der HoloLens die Augmented Reality Revolution an. Sie bieten mit dem 3000$ Gerät ein Stück Hardware an, welches einem den Eindruck gibt, in der Zukunft angekommen zu sein. Die Realität kann mit der virtuellen Welt auf eine Art kombiniert werden, welche man bislang fast nur aus der Science-Fiction kannte.

In diesem Kapitel beschreiben wir vor allem die Hardware-Eigenschaften der HoloLens. Dies ist eine nicht sehr einfache Aufgabe, da Microsoft selbst keine sehr tiefe Auskunft über die einzelnen Komponenten gibt. Man ist also oft auf externe Berichte angewiesen, falls man Genaueres über ein bestimmtes Bauteil erfahren will und dabei stösst man nicht immer auf eindeutige Angaben. Dieser Bericht enthält deshalb Angaben, die von mehreren Quellen bestätigt wurden.

Abbildung 1: Die HoloLens von Microsoft

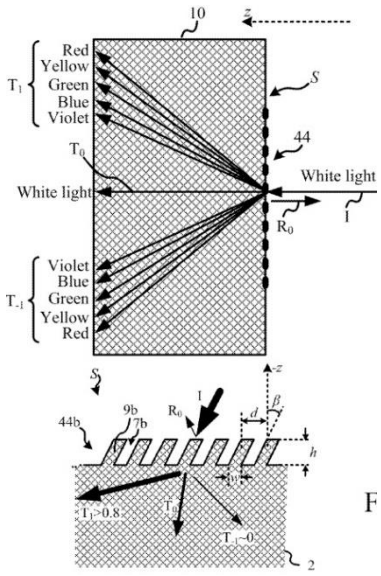
## 2.1 Display / Optics

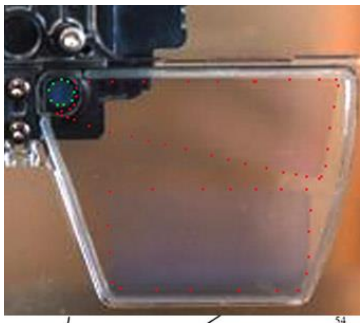
Abbildung 2: Frontansicht der HoloLens

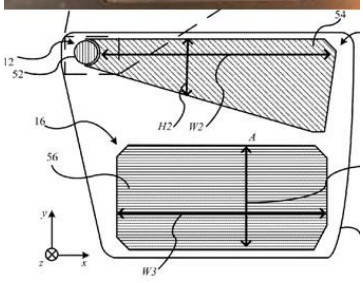
### 2.1.1 Waveguides

Die HoloLens benutzt sogenannte Optical Waveguides, um die reale Welt mit der virtuellen Welt zu verbinden.

Genutzt wird ein Diffraction Grating, welches ein Diffractive Optical Element ist. Dieses Element hat eine sehr feine lineare Struktur, mit Abständen im Bereich von Wellenlängen. Damit wirkt das Diffraction Grating wie eine Linse/Prisma, um das Licht zu biegen. Ein Nebeneffekt davon ist, dass das Licht in die verschiedenen Wellenlängen (Farben) unterteilt wird. Bei simple Grating würde das Licht symmetrisch in zwei Richtungen aufgeteilt werden. Jedoch wird im Patent von Microsoft gezeigt, dass durch eine Schieflage der Struktur mehr Licht in die gewünschte Richtung umgelenkt wird.






Die Optical Waveguides nutzen die Tatsache, dass Licht, welches in einem bestimmten Winkel auf eine transparente Oberfläche auftrifft, vollständig reflektiert wird. Total Internal Reflection (TIR).

Die HoloLens nutzt das Diffraction Grating (52, links) um das Licht so zu biegen, dass es in einem TIR-Bereich ist. Das Licht trifft so auf die rechteckige Fold Zone (54, links), was es dazu bringt, um 90 Grad nach unten Richtung Exit Zone (56, links) zu biegen. Die Exit Zone reduziert den Winkel des Lichtstrahles wieder, so dass es nicht mehr im TIR-Bereich ist und das Glas Richtung Auge verlassen kann.

Dieses Verfahren wird besonders gut im Artikel von Karl Guttag beschrieben und wird dort auch mit dem Patent von Microsoft in Verbindung gebracht.[1]

[1] http://www.kguttag.com/2016/10/27/armr-combiners-part-2-hololens/

### 2.1.2 Light Engines

Die HoloLens verwendet zwei HD 16:9 Light Engines. Diese könnte man auch einfacher als Projektoren bezeichnen. Es sind zwei kleine Liquid Crystal on Silicon (LQoD), welche man auch in regulären Projektoren finden kann.

Diese Light Engines projizieren die Farbbilder, welche durch die Waveguides mit der realen Welt kombiniert werden. Wie genau diese Technologie bei der HoloLens funktioniert, ist nicht bekannt, da Microsoft diese Information nicht herausgegeben hat.

### 2.1.3 Interpupillary Distance (IPD)

Die Distanz zwischen den Pupillen des Users ist in VR und AR sehr wichtig. Die HoloLens kann diese Distanz sowohl horizontal wie auch vertikal anpassen. Durch einen Kalibrierungsprozess ermittelt die HoloLens den korrekten Wert. Dieser Wert wird aber durch das Nutzen einer App ermittelt und nicht durch z.B. einer Eye-Tracking-Kamera. Der Wert müsste also für jeden User wieder neu ermittelt werden, indem man den Kalibrierungsprozess neu startet.

## 2.2 Holographic Processing Unit (HPU)

Die Holographic Processing Unit (HPU) wurde von Microsoft entwickelt, weil die CPU und GPU nicht die benötigte Rechenpower für die HoloLens liefern konnten.

Die Aufgaben der HPU beinhalten die Berechnung der Position und Blickrichtung der Brille, Erkennung von Gesten und das Ausmappen des Raumes.[2] Die Resultate werden dann den 3D Programmen auf der CPU zur Verfügung gestellt.

Bevor die HPU gebaut wurde, besass die HoloLens ein externes Gerät um die nötige Rechenpower zu liefern. Das Gerät bestand aus FPGAs und wurde am Körper getragen. Es war aktiv gekühlt und wurde per Kabel mit der HoloLens verbunden. FPGAs sind programmierbare Hardware Schaltkreise. Schaltkreise sind um ein Vielfaches schneller als eine CPU oder GPU und sind essentiell in Geräten wie der HoloLens. Sie werden oft bei der Entwicklung neuer Hardware verwendet, weil sie flexibel an neue Anforderungen angepasst werden können. Ist die Entwicklung des Geräts genügend fortgeschritten, wird dann das Design von den FPGAs in festen Chips umgesetzt.
Microsoft hat die HPU mittels Tensilica[3] DSP Kernen umgesetzt. Das sind anpassbare Kerne mit dem Ziel, dass Firmen ihre eigenen Chips herstellen können. Microsoft hat in den Kernen ihre eigenen Instruktionen von den FPGAs umgesetzt.[4]

Die HPU erreicht etwa eine Trillion FLOPS und ist damit 200-mal schneller als eine Software Implementation. Bis heute wird die HPU nicht mehr als 50% ausgelastet, was ein zukünftiges Wachstum sichert.

[2] https://blogs.windows.com/devices/2015/04/30/build-2015-a-closer-look-at-the-microsoft-hololens-hardware/
[3] https://en.wikipedia.org/wiki/Tensilica
[4] https://arstechnica.com/information-technology/2016/08/microsoft-sheds-some-light-on-its-mysterious-holographic-processing-unit/

### 2.3 Prozessor

Die HoloLens besitzt einen Intel Broxton System on Chip, auch Atom genannt. Dieser Chip wurde für Tablets entwickelt und ist die Antwort auf die beliebten Mobile Chips von AMD. Die Atom Reihe wurde speziell für den geringen Stromverbrauch entwickelt und unterbietet damit auch die AMD Chips.

Wieso sich Microsoft für die Intel Chips und nicht für den Platzhirsch AMD entschieden hat, ist nicht offiziell bekannt. Vermutlich rührt es daher, dass Microsoft langjährige Geschäftsbeziehungen mit Intel pflegt. Zudem ist es sehr wahrscheinlich, dass Intel Microsoft bei der Entwicklung des HPU unterstützt hat.[5]

Überraschend kommt daher die Nachricht, dass die Atom Reihe von Intel abgesetzt wird. Damit ist nicht klar, was die HoloLens in der Consumer Version enthalten wird.

Obwohl der Chip 64 Bit Architektur ermöglicht, läuft auf der HoloLens eine 32 Bit Windows 10 Variante. Damit wird auf der HoloLens die x86 Architektur verwendet.

### 2.4 Sensoren


Abbildung 3: Übersicht der HoloLens-Sensoren

#### 2.4.1 Inertial Measurement Unit (IMU)

Die Inertial Measurement Unit oder kurz IMU wird verwendet, um die Beschleunigung und Rotation der Brille zu bestimmen. Die IMU ist oberhalb der Nase in der Mitte der Brille untergebracht.
Die IMU der HoloLens besitzt Accelerometer, Gyroskope und Magnetometer. Jeweils einen Sensor pro Achse im 3D Raum. Das Gyroskop und der Magnetometer ermitteln die Rotation und der Accelerometer die Bewegung.
Diese Daten können in unter 10ms geliefert werden, was essenziell ist, um eine genügend schnelle Reaktionszeit des Displays zu erreichen. Würde dies nicht erreicht, würden die Hologramme "driften". Die IMU ist aber nicht alleine zuständig, um die Brillenposition zu aktualisieren. Die Accelerometer können nicht alleine verwendet werden, weil durch die Umrechnung Beschleunigung → Bewegung → Position der Fehler quadratisch zunimmt.[6] Die Accelerometer werden nur benutzt wegen ihrer schnellen Reaktionszeit.
Die Daten der IMU werden primär der HPU zur Verfügung gestellt - sind aber auch dem Benutzer zugänglich.

[5] https://buildhololens.com/2016/05/05/hololens-in-danger-intel-cancels-atom-cpu/#more-115
[6] https://www.youtube.com/watch?v=_q_8d0E3tDk

### 2.4.2 Depth Camera

Die HoloLens besitzt eine Tiefenkamera, welche sich direkt über dem Brillenglas auf der Frontseite des Gerätes befindet. Diese Kamera ist im Grunde eine miniaturisierte Version der von Microsoft hergestellten Kinect, welche nur ca. einen Zehntel so viel Strom verbraucht. Diese Komponente wurde direkt von Microsoft entworfen und entwickelt, etwas was für die Firma nicht üblich ist.

Die Tiefenkamera erkennt die Fingerbewegungen des Benutzers, weiss genau, wo sich der Finger befindet und kann unterscheiden, ob es sich dabei um die linke oder rechte Hand handelt.

### 2.4.3 Environment Understanding Cameras

Die Tiefenkamera arbeitet zusammen mit vier sogenannten Environment Understanding Cameras, also "Umweltverstehende" Kameras. Davon sind jeweils zwei Stück auf jeder Seite der Brille angebracht.

Diese Kameras nehmen die Umwelt in Graustufen auf und sind vor allem dazu da, um die Kopfposition zu bestimmen und die Tiefenkamera beim Mapping der Umwelt zu unterstützen.

Wie das Mesh der Umwelt aufgebaut wird, ist sehr undurchsichtig. Microsoft selbst gibt keine näheren Angaben. Der Prozess des Aufbauens und Aktualisieren des Meshes ist eigenständig. Als Entwickler erhält man nur Snapshots des aktuellen Meshes.

### 2.4.4 Photo / Video Camera

Nebst den Kameras, die verwendet werden um die Umgebung zu mappen und die Position des Gerätes zu bestimmen, gibt es noch weitere Kameras, welche speziell für Foto- und Videoaufnahmen bestimmt sind. Dabei handelt es sich um eine 2.4 MP (2048×1152) Foto-Kamera und eine 1.1 MP (1408×792) Video-Kamera, welche Aufnahmen mit bis zu 30 FPS erstellen kann.

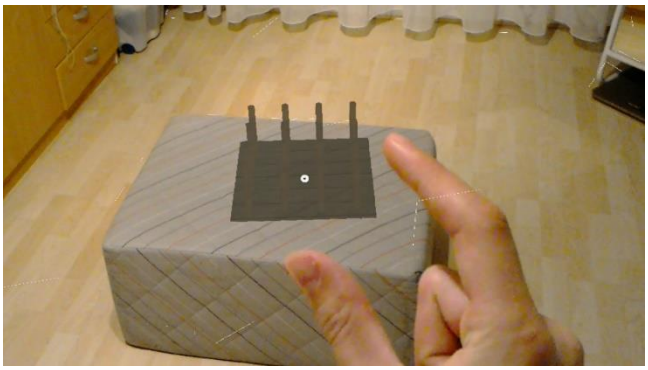


Abbildung 4: Beispiel einer Fotoaufnahme mit der HoloLens

Diese Kameras nehmen jeweils die aktuelle Sicht des Benutzers auf, inklusive der sichtbaren Hologramme. Diese kombinierte Aufnahme von realer und virtueller Umwelt ist was Microsoft als „Mixed Reality Capture" bezeichnet.

## 2.5 Audio

Die HoloLens besitzt zwei Lautsprecher, welche sich auf den Seiten, direkt über den Ohren, befinden. Anders als herkömmliche Kopfhörer verhindern diese nicht das Hören von externen Geräuschen, doch die Tonausgabe ist erstaunlich gut. Personen, die daneben stehen können selbst bei maximaler Lautstärke kaum hören, dass die HoloLens Audio wiedergibt.

Dank der Mixed Reality Audio Engine von Microsoft, werden Töne entsprechend der Position im Raum wiedergegeben. Der Zeitpunkt und die Lautstärke der Wiedergabe werden angepasst, je nachdem wo man sich in Bezug auf die Audioquelle befindet.[7]

## 2.6 Zukünftige Entwicklung

Wie es mit der Entwicklung der HoloLens weitergeht, ist weitgehend unklar. Microsoft lässt sich nicht gerne in die Karten schauen.

[7] https://developer.microsoft.com/en-us/windows/mixed-reality/spatial_sound

Klar ist, dass die erste Version der HoloLens ein voller Erfolg ist. Microsoft hat bekannt gegeben, dass ein Nachfolgemodell nicht vor 2019 zu erwarten ist. Das zeigt auch, dass Microsoft kurzfristig nicht mit ernst zu nehmender Konkurrenz rechnet.

### 2.6.1 Prozessor

Die grösste Frage ist was, Microsoft nach der Absetzung der Atom Chip Reihe machen wird. Möglicherweise wird die nächste Version der HoloLens mit einem ARM Chip ausgerüstet sein.

### 2.6.2 Field of View

Die grösste Kritik an der HoloLens ist das sehr kleine Field of View. Mit dem jetzigen Display hat der Benutzer ein Sichtfeld von nicht einmal 40 Grad. Um dieses Problem zu lösen, hat Microsoft ein Patent angemeldet, das den Einsatz von mehreren Displays beschreibt.

Es gibt die Lightfield Displays, die ein grösseres Sichtfeld ermöglichen und die Möglichkeit bieten, Objekte vollständig zu verdecken. In der HoloLens sind jedoch Waveguide Displays verbaut, weil diese höhere Auflösung, Schärfe und Kontrast bieten.

Das Patent beschreibt nun, wie diese beiden Displays hintereinander eingebaut werden. Damit können die Vorteile beider Techniken genutzt werden.[8] Das hört sich vielversprechend an, billiger wird die HoloLens dadurch aber sicher nicht.

[8] https://mspoweruser.com/microsoft-aims-solve-hololens-field-view-issue-combining-wave-guide-lightfield-displays-patent/

### 2.7 Hardware Spezifikationen

| | |
|---|---|
| **RELEASE DATE** | 30.03.2016 (Development Edition) |
| **PRICE** | $ 3'000 |
| **WEIGHT** | 579g |
| **OPERATING SYSTEM** | Windows 10 |
| **APP MEMORY USAGE LIMIT** | 900 MB |
| **INSTALLED RAM** | 2 GB |
| **INTERNAL STORAGE CAPACITY** | 64GB (54.09 GB available) Toshiba |
| **CPU MODEL** | Intel(R) Atom(TM) x5-Z8100P CPU @ 1.04Ghz |
| **CORE ARCHITECTURE** | Intel Airmont |
| **MANUFACTURING PROCESS** | 14 nm |
| **LOGICAL PROCESSORS** | 4 |
| **HPU MODEL** | Custom made for Microsoft HoloLens |
| **HPU RAM** | 1 GB |
| **GPU MODEL** | HoloLens Graphics |
| **DEDICATED VIDEO MEMORY** | 114 MB |
| **SHARED SYSTEM MEMORY** | 980 MB |
| **PHOTO RESOLUTION** | 2.4 MP (2048 x 1152) |
| **VIDEO RESOLUTION** | 1.1 MP (1408 x 792) |
| **VIDEO SPEED** | 30 fps |
| **BATTERY** | 16,500 mWh |
| **BATTERY LIFE (ACTIVE)** | 2h |
| **WI-FI** | Broadcom 802.11ac Wireless PCIE Full Dongle Adapter |
| **BLUETOOTH** | Bluetooth 4.1 Low Energy (LE) |

Tabelle 1: Hardware Spezifikationen der HoloLens

*Angaben von Wikipedia und buildhololens.com, welche sich auf Daten der Settings App und der AIDA64 App auf der HoloLens stützen.*

# 3 BIM / Modelle

Im Rahmen unserer Arbeit soll der Raum, in dem man sich befindet, mit einem 3D-Model in Verbindung gebracht werden. In diesem Kapitel wird erklärt, worum es sich bei diesen Modellen handelt und wie wir zu unseren Modellen gekommen sind.

## 3.1 Building Information Modeling (BIM)

Building Information Modeling, kurz BIM, ist ein Vorgang, bei dem es darum geht, ein Gebäude und dessen Funktionalitäten digital zu generieren und zu verwalten. BIM wird vor allem im Bauwesen und in der Bauplanung verwendet, sowie im darauffolgenden Facility-Management. Beispiele davon sind: Architektur, Ingenieurwesen, Haustechnik, Tiefbau, Städtebau, Eisenbahnbau, Strassenbau, Wasserbau, Geotechnik, usw.

Das *US National Building Information Model Standard Project Committee* definiert BIM folgendermassen:

> *Building Information Modeling (BIM) is a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition.*[9]

## 3.2 Erstellung der Modelle

In einer idealen, produktiven Anwendung unserer Applikation wären die verwendeten Modelle die offiziellen 3D-Pläne des Gebäudes. Diese würden idealerweise auch zusätzliche Angaben, wie z.B. die Positionen von Stromleitungen, beinhalten. In den meisten Fällen sind dies komplexe CAD (computer-aided design) Pläne, welche in vielen verschiedenen 2D- und 3D-Formaten existieren.

Jeder grössere Hersteller einer CAD-Software hat ein eigenes Dateiformat. Dazu kommen die verschiedenen sogenannten neutralen oder standardisierten Formate. Das sind Formate, die von mehreren CAD-Programmen gelesen und geschrieben werden können. Sie werden vor allem für den Austausch der Pläne verwendet. In der Schweiz werden vor allem die Formate DWG und DXF von Autodesk (AutoCAD) verwendet. DWG ist das proprietäre Format von AutoCAD und DXF das neutrale Format.[10]

Um also eine Lokalisierung mit der HoloLens direkt vom Plan zu ermöglichen, müsste man ein populäres neutrales Format unterstützen. Sollte ein CAD Programm dieses neutrale Format nicht unterstützen, gibt es immer die Möglichkeit über ein zweites Programm die Dateien zu konvertieren.

### 3.2.1 Wahl des Gebäudes und der Räume

Da es bei unserer Arbeit darum geht, das Potenzial der HoloLens im BIM-Umfeld auszutesten, hatten wir kein konkretes Ziel-Gebäude. Somit gab es auch kein bereits vorhandenes Modell eines Gebäudes. Aus diesem Grund haben wir verschiedene Schulungsräume der BFH, an der Wankdorffeldstrasse, selbst modelliert.

Bei der Wahl der Räume haben wir zu Beginn der Arbeit jeweils den Raum benutzt, in dem wir uns gerade befanden. Als es darum ging, die Räume für das Endprodukt zu wählen, haben wir uns für die Räume entschieden, welche bei der Ausstellung der Arbeit genutzt werden. Somit kann die Applikation auch live genutzt werden.

[9] FREQUENTLY ASKED QUESTIONS ABOUT THE NATIONAL BIM STANDARD-UNITED STATES [Link]
[10] Auf Anfrage bei Luca Wyss, Hochbauzeichner und Architekt i.A.

### 3.2.2 Werkzeuge
Um die benötigten 3D-Pläne zu erstellen, haben wir auf verschiedene Hilfsmittel zurückgegriffen. Einerseits auf bereits vorhandene Angaben des Gebäudes und anderseits auf eigene Messungen. Diese Angaben wurden dann zu einer 3D-Repräsentation der Räume zusammengefügt.

#### 3.2.2.1 Software
Als Software, zur Erstellung der 3D-Pläne, wurde Blender verwendet. Blender ist eine freie 3D-Grafiksoftware, welche ideal ist um auf genügend einfache Art, ein genügend komplexes Modell eines Raumes zu erstellen.

Zudem ist die Integration mit Unity sehr einfach, so dass wir erstellte Modelle auf einfachste Art und Weise in unser HoloLens-Projekt importieren konnten. Ein simples *Drag & Drop* des Modells in die Unity Szene genügt und allfällige Änderungen am Modell werden auch direkt in Unity übernommen.

#### 3.2.2.2 Pläne & Messungen
Um an möglichst genaue Angaben zu den Räumen zu gelangen, haben wir vom Sekretariat der BFH die offiziellen Baupläne des Gebäudes erhalten. Diese waren eine gute Grundlage, um die Umrisse der Räume in Blender zu erstellen. Später dienten sie auch, um Teile des Gebäudes zu modellieren, welche zur Orientierung und nicht zur Ortung genutzt wurden, wie z.B. die Gänge zwischen den einzelnen Räumen.

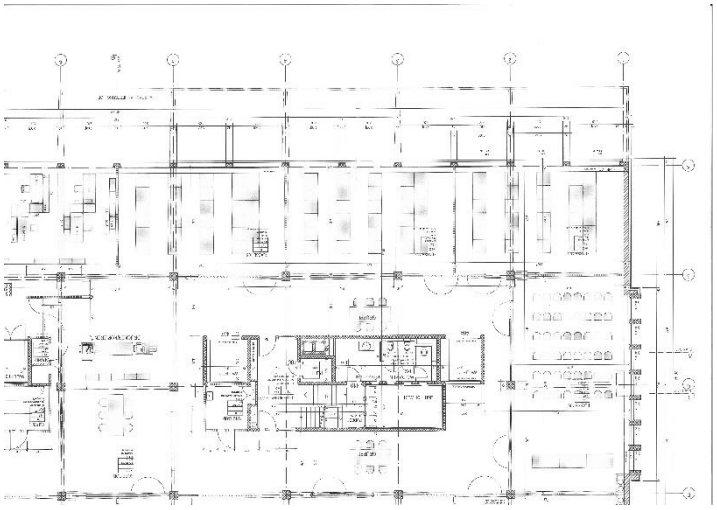


Abbildung 5: Plan der BFH-Räume an der Wankdorffeldstrasse

Da die Räume jedoch mehr Details verlangen, als nur die Umrisse, haben wir die Angaben mit eigenen Messungen ergänzt. Mit einem Lasermessgerät haben wir Details, wie z.B. die Grösse der Fenster oder die Position eines Balkens, zusätzlich ausgemessen und in das Modell integriert.

Wichtig war bei der Erstellung der Modelle vor allem der Unterschied zwischen festen Wänden und Fenstern. Da die HoloLens eine durchsichtige Scheibe als ein Loch erkennt, bzw. durch die Fläche hindurch scannt, ist ein solches “Loch” im Modell ein wichtiges Erkennungsmerkmal des Raumes.

### 3.2.3 Umsetzung
Um die gesammelten Daten in das gewünschte 3D-Model zu kombinieren, wurden mehrere Versuche benötigt.

#### 3.2.3.1 Erster Versuch
Im ersten Anlauf wurden die Pläne des Gebäudes gescannt und in Blender als Hintergrund eingeblendet. Dies sollte erlauben das Modell direkt auf den Grundrissen der Pläne zu erstellen. Somit würden die Verhältnisse der verschiedenen Wände gewährleistet sein.

Dieses Vorgehen erwies sich jedoch als ein weniger praktischer Ansatz. Die gescannten Linien waren teilweise zu ungenau oder nicht genügend gerade ausgerichtet. Es verlangte zu viel Nacharbeit, die Pläne korrekt auszurichten und es entstanden trotzdem Ungenauigkeiten, welche wir nicht riskieren wollten.

Aus diesem Grund haben wir zu diesem Zeitpunkt diesen Ansatz verworfen und einen anderen Weg eingeschlagen.

#### 3.2.3.2 Zweiter Versuch

Im zweiten Anlaufen entschieden wir uns dafür Blender so einzustellen, dass wir direkt mit metrischen Einheiten arbeiten konnten. Standardmässig nutzt Blender nämlich eine nicht definierte Einheit.

Danach wurde eine Wand nach der anderen erstellt, wobei wir jeweils deren Längen und Winkel zu einander aus den Plänen herausgelesen haben. Wir haben also den Umriss der Räume anhand der Pläne nachgestellt, statt sie über die Pläne nachzuziehen. Dies erwies sich als die präzisere und schnellere Variante.

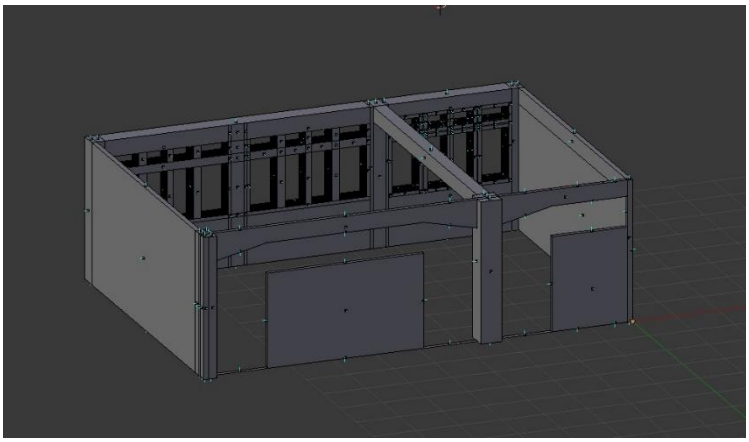


Abbildung 6: Model eines Raumes, erstellt in Blender

Mit den kompletten Umrissen konnten wir das noch flache Modell nun in die dritte Dimension ziehen. Dazu haben wir die Höhe des Raumes mithilfe des Messgerätes determiniert und das Modell um die entsprechende Höhe dicker gemacht.

Weiter mussten wir die nun vorhandenen Wände an den korrekten Stellen unterbrechen. An jeder Stelle, an der die Wände aus einem durchsichtigen Material bestehen, wie z.B. Glaswände oder Fenster, darf kein Teil des Modells vorhanden sein.

# 4 Mesh Analyse

Früh im Entwicklungsprozess unserer Applikation, war uns nicht klar wie bzw. auf welcher Ebene wir die gescannte Umgebung analysieren und bearbeiten wollten.

Das Ergebnis einer durch die HoloLens gescannten Umgebung ist ein Mesh bestehend aus den Punkten (Vertices) der Punktewolke und den Kanten (Edges), welche diese Punkte verbinden. Dieses Mesh kann einerseits als solches angeschaut werden oder als reine Punktewolke. Eine Punktewolke ist dabei ein in der Augmented Reality verbreitetes Standard, für welches es verschieden Libraries und Frameworks gibt.

Die Frage war deshalb, ob wir mit einer spezialisierten Library direkt mit der Punktewolke arbeiten wollten oder ob wir auf eine der MixedRealityToolkit-Komponenten aufbauen sollten.

### 4.1 Point Cloud Library (PCL)[11]

Als spezialisierte Library haben wir uns entschieden, die Point Cloud Library (PCL) zu analysieren. Die PCL ist eine Open-Source Library, welche speziell zur Verarbeitung von Punktwolken entwickelt wurde. Als solches beinhaltet die Library eine grosse Sammlung von optimierten Algorithmen, welche im Umgang mit Punktwolken häufig verwendet werden.

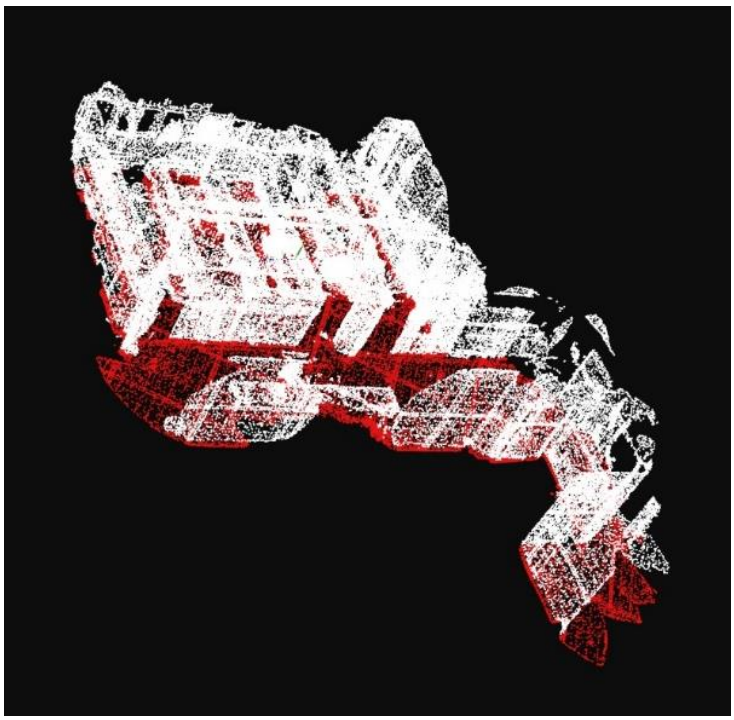


Abbildung 7: Erreichtes Resultat mit der PCL. In Rot sind die Punkte, welche den Boden ausmachen.

#### 4.1.1 Analyse / Versuche

Damit wir uns möglichst schnell ein Bild der PCL machen konnten, haben wir eine vor-kompilierte Version der Library genutzt. Diese Installation konnte als eigenständiges C++ Projekt ausgeführt werden, was uns die Möglichkeit gab auf einfachste Art und Weise die verschiedenen Tutorials durchzuspielen.

Die Idee war herauszufinden, wie wir innerhalb der Punktewolke Wände, Böden und Decken herausfiltern können. Dazu wurde ein gescannter Raum von der HoloLens exportiert. Die liefert einem eine Datei im OBJ-Format. Dieses Format beinhaltet die Koordinaten aller Punkte, deren Normalen und der Flächen, welche sie repräsentieren.

Dieses Format ist eines der Formate, welches mit der PCL importiert werden kann. Wir konnten somit ein kleines Programm schreiben, welches uns eine Idee der möglichen Ergebnisse liefern konnte, ohne die PCL bereits in die HoloLens-App integrieren zu müssen.

Mit einer Kombination aus verschiedenen Tutorials[12] gelang es auch, wahlweise die Punktegruppen aus Böden, Decken und Wänden anzuzeigen.

#### 4.1.2 Fazit

Das Resultat entsprach zwar dem, was wir uns vorgestellt hatten. Doch wir entschieden, dass der Zugang zu all den vorhandenen Algorithmen der PCL, den Zusatzaufwand, welcher die Integration in die HoloLens-App mit sich bringen würde, nicht wert war. Die Zeit die wir mit der Integration verbringen würden, würde uns fehlen, um die eigentliche Applikation schreiben zu können. Zudem würde das Verwenden der PCL einen Schritt zurück bedeuten, da wir nur mit der Punktewolke arbeiten müssten. Die HoloLens bietet uns jedoch ein Mesh an, welches bereits Kanten zwischen verbundenen Punkten besitzt.

[11] http://www.pointclouds.org/
[12] http://www.pointclouds.org/documentation/tutorials/

### 4.2 Spatial Understanding

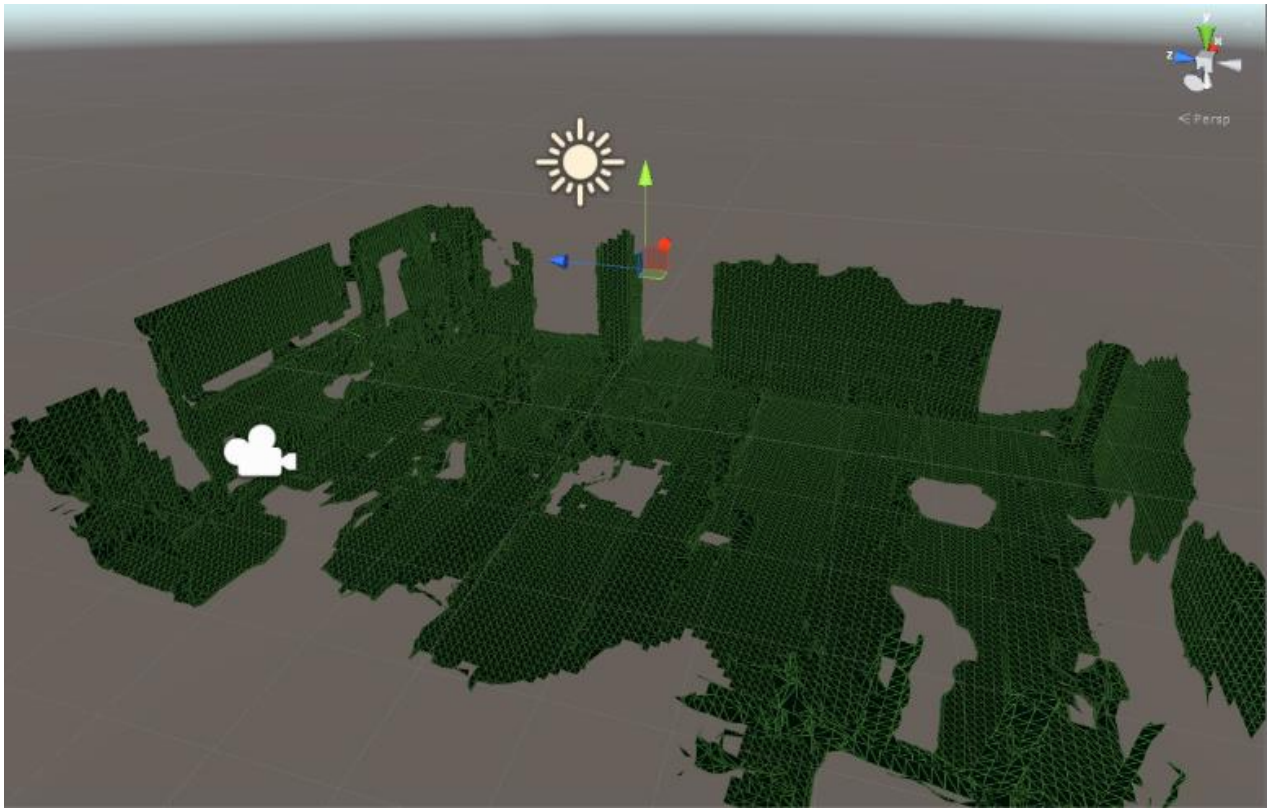


Abbildung 8: Das Mesh nach der Verwendung der Spatial Understanding API.

Das Spatial Understanding API im HoloToolkit wurde von Asobo Studios für ihre zwei Spiele Conker und Fragments entwickelt.[13] Das API ermöglicht einen sehr High-Level Zugriff auf Rauminformationen. Beispielsweise kann ein freier Platz auf dem Sofa oder eine zwei Quadratmeter grosse Fläche an der Wand abgefragt werden.

Für unsere Arbeit ist dieses API interessant, weil es das Umgebungs-Mesh verarbeitet und dabei Wände, Boden und Decke ausglättet.

#### 4.2.1 Voxel

Das Spatial Understanding API arbeitet mit Voxels. Das ist eine alternative Datenstruktur zum Umgebungs-Mesh. Es unterteilt den Raum in viele gleich grosse Kuben.

Um diese Voxel Struktur aufzubauen, muss zuerst wie in unserer finalen Applikation, die Umgebung gescannt werden. Dann berechnet das API aus den Mesh Daten das Voxel Grid. Erst ab diesem Zeitpunkt sind dann Abfragen möglich.

[13] https://github.com/Microsoft/MixedRealityToolkit/wiki/HoloToolkit.SpatialUnderstanding

#### 4.2.2 Fazit

Das Spatial Understanding API sah sehr vielversprechend aus. Leider ist während dem Austesten dann eine gravierende Limitation aufgetreten: Das Voxel Grid ist in seiner Grösse beschränkt. Es kann maximal einige Meter in alle Richtungen gross sein. Falls der Startpunkt in der Mitte des Raumes ist, reicht es gerade, um die Schulzimmer des Gründerzentrums einzuschliessen. Ist der Startpunkt aber an einem Ende des Raumes, so endet das Voxel Grid mitten im Raum.

Ein zweiter Kritikpunkt ist, dass das API nicht für unseren Use Case gedacht ist. Wir wollen vor allem wissen, wo der Boden, die Decke und die Wände sind. Das API ist dafür gedacht freie Flächen an solchen zu finden.

Aus diesen Gründen haben wir uns entschieden, dieses API nicht zu verwenden. Das Risiko ist zu gross, dass diese Limitationen später das Projekt gefährden.

### 4.3 Plane Finding API

Das Plane Finding API macht genau das, was der Titel sagt: Es nimmt als Input das Umgebungs-Mesh und gibt als Output parametrisierte Flächen zurück.

Für unsere Arbeit ist dieses API sehr interessant. Wir benötigen Oberflächen, die je nach Orientierung Boden, Decke oder Wände darstellen können.

Eine interessante Eigenschaft des API ist, dass für denselben Input immer andere Flächen gefunden werden. Das ist auch nötig, weil die Anzahl an gefundenen Flächen sehr fluktuiert. Manchmal werden zwei Flächen gefunden, manchmal zwanzig. Da jeder Call ans API immer wieder andere Resultate liefert, können wir es so viele Male aufrufen, bis genügend Flächen vorhanden sind.

#### 4.3.1 Fazit

Das Plane Finding API passt genau auf unseren Use Case. Unglücklicherweise stellte es sich aber auch als instabil heraus. Das bedeutet, durch das Plane Finding API stürzte unsere Applikation etwa jedes zweite Mal ab. Das passierte im Unity Editor, wo es den ganzen Editor schliesst und auf der HoloLens.

Das API ist open source. Wir hatten also die Möglichkeit den Fehler zu finden und zu beheben. Wir haben uns aus mehreren Gründen dagegen entschieden. Zum einen ist es in C++ geschrieben und keiner von uns hat viel Erfahrung in dieser Sprache. Dann muss eine C++ Library speziell eingebunden werden um mit C# in Unity zu verwenden. Wir kennen diesen Prozess nicht.

### 4.4 Entscheid

Nach der Analyse der verschiedenen Optionen haben wir uns dazu entschieden eine eigene Lösung anzustreben. Keiner der untersuchten Ansätze bot einen deutlichen Mehrwert gegenüber dem nötigen Zeitaufwand, um sie anzuwenden.

Die Analyse der verschiedenen Libraries und APIs bot uns jedoch die Möglichkeit, uns mit dem Mesh der gescannten Umgebung auseinanderzusetzen. Dadurch wurde uns bewusst, dass die verfügbaren Angaben genügen, um einen eigenen Ansatz anzustreben. Das Mesh gibt uns Zugang zur Position aller Punkte und Normalen der gescannten Umgebung, was uns erlaubt, mit eigens geschriebenen Algorithmen zu definieren, wo sich eine Wand befindet.

# 5 Die App

Der Code ist in Module nach Funktionalität unterteilt. Dieses Kapitel ist nicht dazu da, jede Zeile zu erklären. Der Code enthält viele Kommentare, die den aktuellen Code-Block erklären und teilweise auch, wieso es so gemacht wurde. Das Kapitel dient dazu, die Schlüsselfunktionen der Module aufzuzeigen und die Gedanken hinter den Designentscheidungen näher zu bringen.

## 5.1 Gemeinsamkeiten

Die Module sind im Projekt unter dem Ordner "Scripts" abgelegt.

Die Module sind meist Singleton und State Machines um die intermodale Kommunikation zu vereinfachen. In Unity erben die Scripts meist von der MonoBehaviour Klasse, um einen einfachen Zugriff auf das Unity API zu ermöglichen. Auch unsere Module erben von dieser Klasse.

Als Folge davon werden sie bereits beim Start der Applikation instanziiert. Da die Module bereits instanziiert wurden, kann der Datenaustausch über Module.Instance realisiert werden.

Die Module sind jeweils nicht immer aktiv. Generell warten sie zuerst, bis die Daten vom vorherigen Modul verfügbar sind, verarbeiten die Daten und geben sie dann weiter. Diese Stadien sind als State Machine implementiert.

## 5.2 Übersicht über die Module

- **ScanManager**
  Interagiert mit dem HoloLens API um das Umgebungs-Mesh zu bilden. Enthält Optimierungen um den Scan Prozess möglichst kurz zu halten.

- **ScanProgress**
  Entscheidet beim Scannen des Raumes, wann genügend Daten für die weitere Verarbeitung gesammelt wurden. Unterstützt den Benutzer beim Scan Prozess, indem er ihn per Sprachausgabe zu den, noch nicht gescannten Bereichen, des Raumes führt.

- **MeshAnalyzer**
  Analysiert das Umgebungs-Mesh und extrahiert Boden, Decke und Wände des Raumes. Daraus werden dann die Dimensionen und die Ecken des Raumes für die weitere Verarbeitung berechnet.

- **RoomIdentifier**
  Dieses Modul analysiert den gefundenen Raum und versucht ihn einem seiner eingespeicherten Modellen zuzuordnen. Gelingt das, bewegt es das Modell an die Stelle des realen Raumes, um die Augmented Reality zu ermöglichen.

- **MiniMap**
  Ermöglicht es, das Modell per Sprachkommandos zu schrumpfen und zu bewegen.

## 5.3 ScanManager

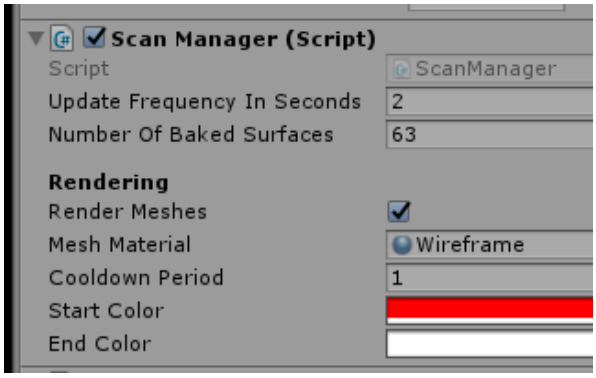



Abbildung 9: Scan Manager in Unity

### 5.3.1 Spatial Mapping API der HoloLens

Das Spatial Mapping API der HoloLens ist dazu da die Geometrie realer Objekte in den virtuellen Raum zu bringen.

Die Objekte existieren ja bereits und können durch die transparenten Gläser der HoloLens betrachtet werden. Um nun mit diesen Objekten interagieren zu können, benötigen wir nur die Geometrie. Damit kann ein virtuelles Spielbrett auf einen Tisch gestellt werden oder ein virtueller Ball eine Rampe herunterrollen.

#### 5.3.1.1 Datenstruktur

Um das API besser zu verstehen, ist es hilfreich zu wissen, wie die resultierenden Spatial Mapping Daten aussehen werden.

Die HoloLens liefert kein zusammenhängendes Mesh! Während des Scanning-Prozesses läuft der Benutzer mit der HoloLens herum und schaut umher, damit die Sensoren die Oberfläche erfassen können. Die HoloLens kann dabei die eigene Position perfekt bestimmen. Die Tiefenkamera, die hauptsächlich für die Erstellung der Geometrie verwendet wird, hat aber einen eingeschränkten Sichtbereich. Sie sieht nur Ausschnitte des Raumes - sogenannte Surfaces (Oberflächen). Diese Surfaces sind in der Regel nicht grösser als 1m und werden jeweils einzeln verwaltet.

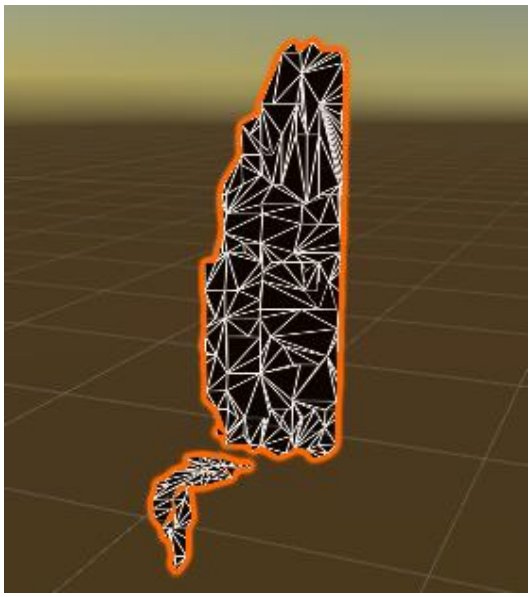


Abbildung 10: Beispiel einer Surface (Oberfläche)

#### 5.3.1.2 Observer

Um an Surface Daten zu kommen, muss zuerst ein Observer registriert werden. Dem Observer wird ein räumliches Volumen mitgegeben, das er überwachen soll. Es werden dann nur Oberflächen an den Observer weitergegeben, die sich innerhalb dieses Volumens befinden.

Das Volumen kann verschieden definiert werden. Als Box, als Sphäre oder als Frustum. Welche Art verwendet wird, kommt auf die Applikation darauf an. Wird beispielsweise ein Hologramm fest platziert, ist nur die nähere Umgebung interessant. Das Volumen kann also als AxisAlignedBoundingBox um das Hologramm definiert werden. Hingegen eine Applikation, die innerhalb der Sicht des Trägers operiert, würde das Volumen als Frustum definieren. Und da sich der Träger bewegt, muss das Volumen (Frustum) immer wieder neu gesetzt werden.

#### 5.3.1.3 Baking

Der Observer gibt über die Updatemethode Oberflächen-Events an die Applikation weiter. Mögliche Events sind 'Added', 'Updated' oder 'Removed'.

Diese Updates enthalten aber keine Oberflächen-Mesh-Information. Nur wo sich die Oberfläche befindet und wie gross sie ist. Um an das Mesh zu gelangen, muss es zuerst gebacken werden. *Baking* ist der offizielle Begriff für diesen Vorgang.

Dieser Prozess vom Update und vom Backen wird aufgeteilt, weil das Backen ein sehr aufwändiger Vorgang ist. Durch die Aufteilung kann die Applikation selbst bestimmen, welche Oberflächen nun zuerst gebacken werden sollen.

### 5.3.2 Implementation

In unserem Fall wollen wir den gesamten Raum scannen. Das bedeutet, wir erstellen einen Observer mit einem riesigen Volumen, damit wir auch alle Updates erhalten.

Beim Backen gibt es mehrere Faktoren um zu entscheiden, welche Oberfläche als Nächstes prozessiert werden soll.

### 5.3.3 Baking Priorisierung

Zum einen wollen wir vor allem neue Oberflächen priorisieren. Updates bestehender Flächen sind zweitrangig, weil diese meist nur kleine Änderungen mit sich bringen. Um eine Wand oder Decke zu erkennen, sind solche Details nicht von Belang.

Weiter sollen die Stellen bevorzugt werden, die der Benutzer gerade betrachtet. Dies vermittelt die Aktivität der Applikation und gestaltet den Scan Prozess interaktiver. Dazu wird der Blick des Benutzers getrackt und der Zeitpunkt der letzten Berührung des Blicks mit der AABB der Oberfläche wird vermerkt. Es muss die AABB verwendet werden, da oft das Oberflächen-Mesh ja noch gar nicht zur Verfügung steht.

### 5.3.4 PriorityQueue

Um die nächste Oberfläche zum Backen auszuwählen wird eine PriorityQueue verwendet. Als erste Priorität wird der BakedState verwendet. Es werden also immer zuerst die Oberflächen genommen, die noch nie gebacken wurden. Unter den Verbleibenden wird die zuletzt betrachtete Oberfläche ausgewählt.

Die Reihenfolge in der PriorityQueue müsste aufgrund des Blicks des Benutzers kontinuierlich in jedem Update-Cycle aktualisiert werden. Um dies zu verhindern, wurde eine eigene LazyPriorityQueue implementiert. Diese evaluiert die Reihenfolge erst, wenn das nächste Element der Queue zurückgegeben werden soll.

### 5.3.5 Aktualisierung visualisieren

Um zu visualisieren, welche Oberflächen gerade hinzugefügt oder aktualisiert wurden, wird nach dem backen das Wireframe der Oberfläche in Rot gezeichnet. Die Farbe wird dann über den Zeitraum von einer Sekunde wieder zu Weiss gewechselt. Diese Farben und der Zeitraum können im Editor angepasst werden.

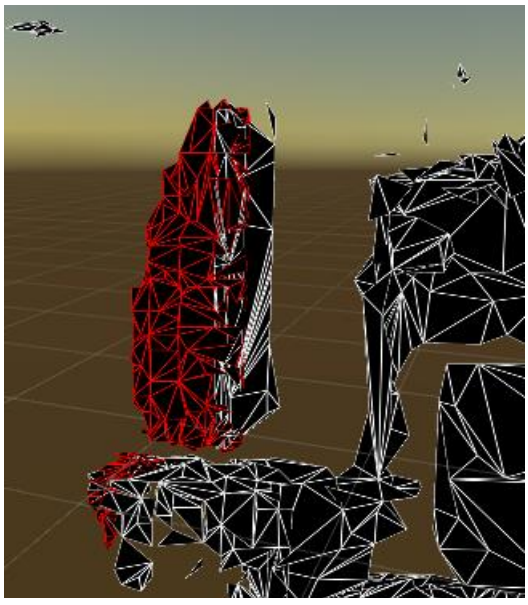


Abbildung 11: Beispiel einer gerade hinzugefügten Oberfläche (rot)

### 5.4 ScanProgress

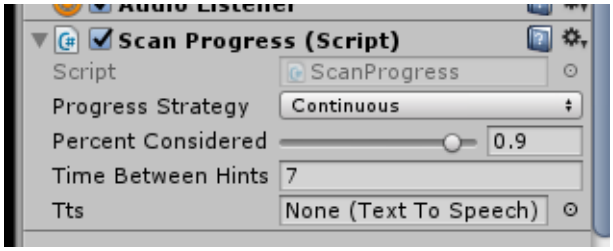



Abbildung 12: Scan Progress in Unity

Der ScanProgress wird während dem Scan-Prozess ausgeführt und entscheidet, wann genügend Oberflächen gesammelt wurden, um den Raum zu identifizieren.

Die Berechnungen, um die Wände und Decke zu identifizieren, sind zeitaufwendig und können nicht kontinuierlich aufgerufen werden. Es muss also eine simplere Methode verwendet werden.

#### 5.4.1 Sensoren

Im Modul werden Sensoren in alle Richtungen definiert. Die Sensoren sind in Längen- und Breitengrade unterteilt, um die Auswertung zu vereinfachen. Sie gehen immer vom Benutzer aus. Sie bewegen sich also mit der Brille mit, aber behalten ihre Orientierung bei. Ein Sensor zeigt z.B. also immer nach Norden.

Bei jedem Update-Cycle von Unity wird von jedem Sensor aus ein Raycast gemacht, der nur auf das Umgebungs-Mesh treffen kann. Trifft er auf eine Oberfläche, setzt er seinen Status auf “detektiert” und wechselt seine Farbe in der Unity Entwicklungsumgebung auf Grün.

Diese Sensoren sieht man auf dem ScreenShot oben.

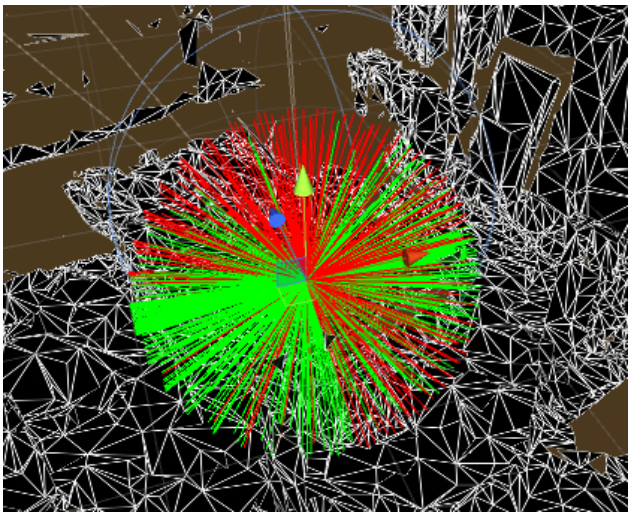


Abbildung 13: Sichtbare Sensoren innerhalb der Unity-Szene

#### 5.4.2 Wann wurde genug gescannt?

Der nächste Schritt ist das Detektieren der Wände, des Boden und der Decke. Also benötigen wir Oberflächen gegen unten, oben und horizontal in alle Richtungen.

Um diese Checks auszuführen, werden die horizontalen Sensoren ( ±30 Grad Lat.) in acht Sektoren unterteilt. Zusätzlich gibt es je einen Sektor gegen oben und unten.

Der Scan gilt nun als komplett, sobald 90% der Sensoren in jedem Sektor detektiert haben.

#### 5.4.3 Strategie

Es wurden zwei Strategien zur Sensorauswertung implementiert: SingleFrame und Continuous.

Bei SingleFrame gilt ein Sensor nur als detektiert, wenn er im aktuellen Update-Cycle eine Oberfläche detektiert hat. Bei Continuous gilt ein Sensor als detektiert, solange er zu irgendeinem Zeitpunkt eine Oberfläche berührt hat.

Es hat sich schnell herausgestellt, dass für unsere Anwendung nur Continuous in Frage kommt. SingleFrame wird nie fertig, da immer wieder Sensoren auf Fenster zeigen und daher nicht detektieren. Man könnte die Prozentzahl verringern, um dem entgegenzuwirken. Das hat aber zur Folge, dass die Komplettierung des Scans sehr willkürlich wird.

### 5.4.4 Benutzerführung

Um den Benutzer zu den Stellen des Raumes zu führen, die noch nicht gescannt wurden, wurde eine Benutzerführung mit Sprachausgabe implementiert.

Durch die Aufteilung in Sektoren kann das Modul ermitteln, welchen Sektor der Benutzer gerade betrachtet und welche noch gescannt werden müssen. Basierend darauf wird der Benutzer aufgefordert den Bereich zu seiner rechten/linken, hinter ihm, die Decke, etc. zu scannen.

### 5.5 MeshAnalyzer

Dieses Modul nimmt das Umgebungs-Mesh und extrahiert daraus Boden, Decke und alle Wände oder Wand-ähnlichen Objekte. In einem zweiten Schritt werden die korrekten Wände ermittelt und die Raumdimensionen und die Ecken des Raumes berechnet.

#### 5.5.1 Kategorisierung der Normalen

Vektoren haben nach Definition nur eine Richtung und eine Länge. Der Begriff passt also nicht, um unsere Oberflächennormalen zu beschreiben, da diese einen Ort und eine Richtung haben. Wir haben unsere Normalen daher simpel “Line” getauft. Dieser Begriff wird von nun an verwendet.

Diese Line-Objekte sind simple Konstrukte, welche zwei Vektoren beinhalten. Einer für den Ursprung und einer für die Richtung. Jede Normale, welche wir vom Umgebungs-Mesh erhalten, wird, zusammen mit ihrem Vertex, zur Erstellung eines solches Objektes benutzt.

Die Line-Objekte müssen anschliessend noch kategorisiert werden. Dies geschieht, in dem wir die Line-Objekte in drei Kategorien klassieren: Nach oben, unten oder horizontal ausgerichtet. Dazu verwenden wir jeweils ein Referenz-Vektor und erlauben eine gewisse Abweichung.

#### 5.5.2 Decke und Boden

Die Decke und der Boden können sehr einfach ermittelt werden. Wir nehmen alle Decken- und Boden-Lines und prüfen, welche am höchsten, beziehungsweise am tiefsten gelegen ist.

Damit können viele Störfaktoren umgangen werden. Beispielsweise haben Tische eine grosse Oberfläche und sind alle auf derselben Höhe. Würde man nach der grössten Oberfläche mit der Normalen nach oben suchen, besteht die Gefahr, dass der Boden auf der Höhe der Tische gesetzt wird. In den Schulzimmern des Gründerzentrums befinden sich auch fast flächendeckend Gitter an den Decken. Unsere Variante verhindert auch in diesem Fall eine Fehlplatzierung.

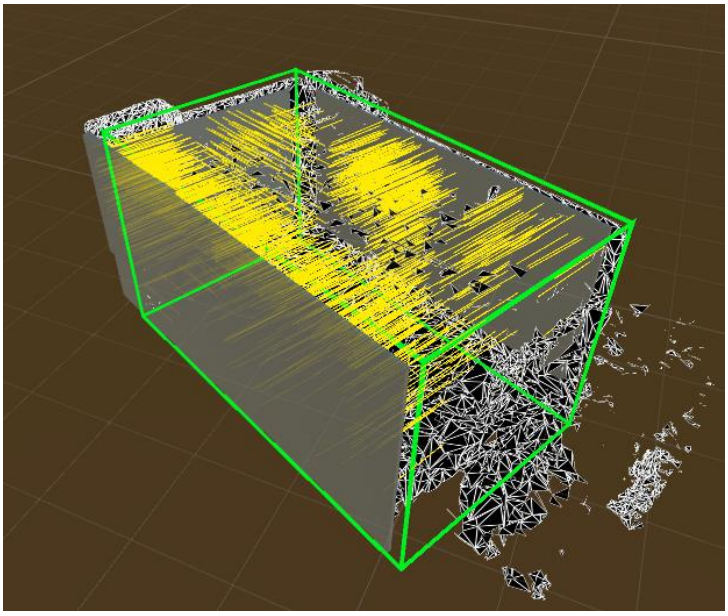


Abbildung 14: Erkannter Raum

#### 5.5.3 Wände

Dies war eines der schwierigsten Probleme dieser Arbeit. Aus einer Menge an horizontalen Lines die Wände des Raumes bestimmen. Es ist nicht nur schwierig, ein gutes Resultat zu bekommen. Die Performance ist auch eine Herausforderung. Unsere erste Version rechnete auf einem Desktop fünf Minuten.

##### 5.5.3.1 Lines zu Planes zusammenfassen

Der erste Schritt ist, die Lines, die sich auf derselben Ebene befinden, zusammenzufassen. Das bedeutet, es werden die Oberflächen-Stücke, die zur selben Wand gehören, gruppiert.

Dazu wird eine Line zufällig ausgewählt. Diese besitzt einen Ort und eine Normale, was wir benutzen, um eine Ebene zu definieren. Dann werden die verbleibenden Lines überprüft. Befinden sie sich auf derselben Ebene und haben dieselbe Normale, so werden sie zusammengefasst. Stimmen sie mit keiner bestehenden Ebene überein, wird eine neue Ebene mit dieser Line erstellt.

#### 5.5.3.2 Variable Planes
Dieses Vorgehen hat einen Nachteil. Die Ebene, die zu Beginn von der Line erstellt wurde, stimmt oft nicht genau mit der Wand überein. Die Normale der Linie stimmt kaum einmal perfekt mit der realen Wand überein. Als Folge davon werden nicht alle Lines einer realen Wand in die selbe Ebene zusammengefasst und es gibt oft mehrere ähnliche Ebenen für eine Wand.

Um dem entgegenzuwirken wurden Ebenen eigeführt, die sich verändern. Diese berechnen für jede hinzugefügte Line das arithmetische Mittel der Normalen aller Lines dieser Ebene. Das hat den Effekt, dass sich die Ebene, währenddem immer mehr Lines hinzugefügt werden, kontinuierlich an die reale Wand anschmiegt.

##### 5.5.3.2.1 Ebenen zusammenfassen
Damit kann es jedoch noch immer zu mehreren Ebenen pro Wand kommen. Um diese zu eliminieren, werden die Ebenen auch noch zusammengefasst. Dabei werden dieselben Regeln angewandt, wie wenn eine Line zu einer Ebene hinzugefügt wird. Stimmen die Normalen überein und befinden sich die zwei Ebenen am selben Ort, so werden sie zusammengefasst.

#### 5.5.3.3 Die vier Wände
Nun müssen aus der Menge der gefundenen Ebenen die Wände identifiziert werden. Die Schwierigkeit dabei ist, dass abhängig vom Scan, die Ebenen sehr unterschiedlich gross sind. So umfasst die grösste Wand mehrere hundert Lines. Dagegen kann die Wand, die vom Scan nicht so gut erfasst wurde, nur gerade ein paar Dutzend Lines besitzen.

Beispielsweise kann die Wandtafel im Schulzimmer eine grössere Fläche besitzen als die Fensterfront. Eine generische Lösung ist also schwierig umzusetzen. Deshalb stützen wir uns auf Vorwissen über die Räume. Die Schulräume des Gründerzentrums sind alle rechteckig und besitzen damit genau vier Wände. Immer zwei Wände stehen sich parallel gegenüber.

Der Algorithmus startet mit der grössten Ebene. Diese ist mit grösster Wahrscheinlichkeit eine reale Wand. Dann wird die grösste Ebene gegenüber dieser ersten Wand gesucht. Das wird gemacht, indem die Normale der ersten Wand invertiert wird und nach der grössten Ebene mit dieser Normalen gesucht wird.

Dasselbe wird mit den zwei verbleibenden Wänden gemacht. Diese stehen in einem 90 Grad Winkel zur ersten Wand und können so identifiziert werden.

### 5.5.4 Dimensionen
Um später den Raum zu identifizieren, muss die Grösse des Raumes bestimmt werden.

Um die Distanz zwischen zwei Wänden zu bestimmen, wird ein Punkt auf der ersten Wand genommen und damit der nächste Punkt auf der gegenüberliegenden Wand berechnet. Die Distanz zwischen diesen beiden Punkten ist eine Dimension des Raumes.

Dasselbe wird mit dem anderen Wände-Paar gemacht und mit der Decke und dem Boden.

### 5.5.5 Raumecken
Für die weitere Verarbeitung werden auch die Ecken des Raumes benötigt. Diese werden aus den Schnittpunkten von je zwei Wänden und der Decke oder dem Boden berechnet. Insgesamt erhält man damit 8 Eckpunkte, mit denen der Raum definiert werden kann.

### 5.6 RoomIdentifier

Dieses Modul ist dafür zuständig, den Raum, in dem der Benutzer steht zu identifizieren.

#### 5.6.1 Virtuelle Räume

Hier kommen die 3D Modelle von Blender ins Spiel. Diese werden in Unity importiert. In Unity müssen sie dann mit Metadaten angereichert werden. Dies sind zum einen die Dimensionen des modellierten Raumes und die Nachricht, falls der Raum identifiziert wurde. Auf dem Modell muss markiert werden, welche Objekte die Wände darstellen und welche die Fenster. Dies wird mit Unity Layers gemacht.

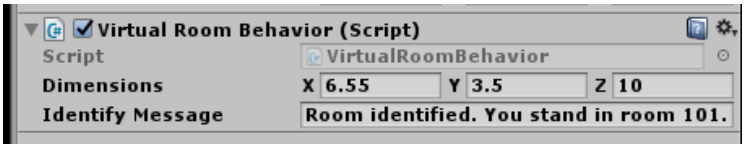



Abbildung 15: Virtual Room Behavior in Unity

#### 5.6.2 Identifizierung

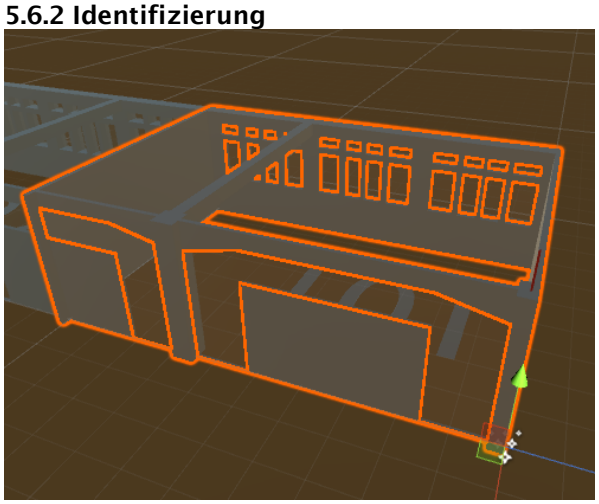


Abbildung 16: Bekanntes Modell innerhalb der Unity Szene

Um den Raum zu identifizieren, stehen uns sehr limitierte Optionen offen. Die HoloLens besitzt kein GPS und wir haben auch keinen Zugriff auf den Kompass.

Uns steht mehr oder weniger nur das Umgebungs-Mesh zur Verfügung.

Um die Identifizierung per Umgebungs-Mesh durchzuführen, benutzen wir die Tatsache, dass die Tiefensensoren der HoloLens durch die Scheiben des Raumes hindurchgehen. Das bedeutet, im Umgebungs-Mesh sind die Scheiben Löcher in der Wand.

#### 5.6.2.1 Footprint

Um nun den realen Raum mit den Modellen zu vergleichen, wird ein sogenannter Footprint erstellt. Dazu werden auf halber Höhe des Raumes den Wänden entlang Raycasts gemacht. Dabei wird gespeichert, ob sie auf eine Oberfläche treffen oder nicht. Im ScreenShot oben sind die Raycasts, die auf etwas getroffen sind, grün und die anderen rot.

Wir sehen im Screenshot, dass die Fenster gut sichtbar werden. Es wird aber auch offensichtlich, dass wir Löcher im Umgebungs-Mesh haben.

Im Gegensatz dazu ist der Modell Footprint perfekt.

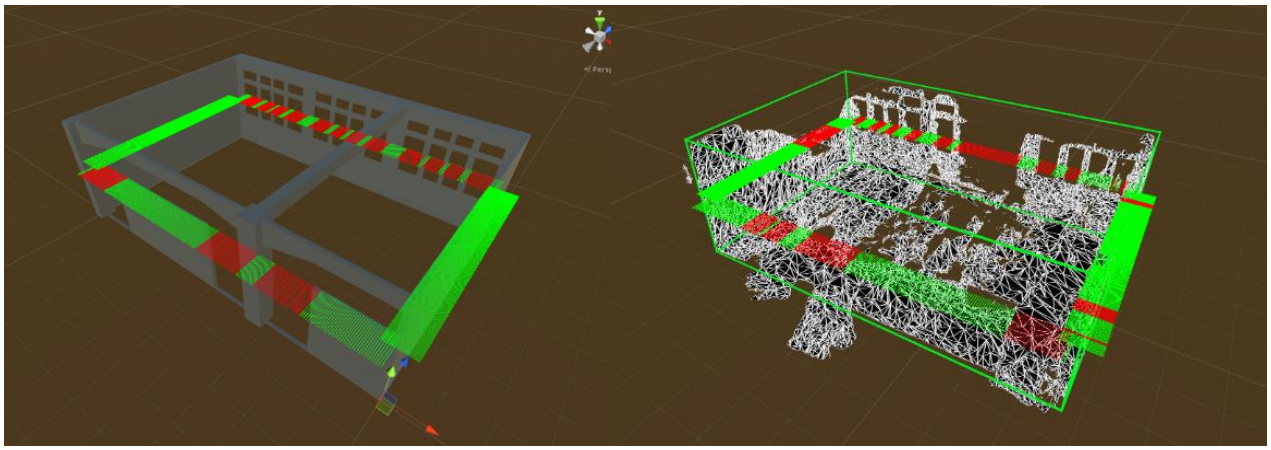


Abbildung 17: Beispiel eines Footprints im modellierten Raum

Abbildung 18: Beispiel eines Footprints im gescannten Raum

#### 5.6.2.2 Vergleich
Um den Vergleich erst möglich zu machen, muss noch sichergestellt werden, dass die korrekten Wände miteinander verglichen werden. Beispielsweise soll die Fensterfront im realen Raum mit der Fensterfront des Modells verglichen werden.

Leider haben wir keine Möglichkeit zu wissen, welche Wand welche ist. Wir haben vier Möglichkeiten die Wände zu vergleichen.

Diese Zahl kann noch halbiert werden. Der Vergleich ist so definiert, dass er immer in einer Ecke startet, die vom Inneren des Raumes aus gesehen die längere Wand zur rechten und die kürzere Wand zur linken hat.

Damit haben wir noch zwei Möglichkeiten. Dies können wir nicht verbessern, da uns der Zugriff auf einen Kompass fehlt.

Während des Vergleichs wird jedes Modell zweimal geprüft. Einmal normal und einmal um 180° gedreht. Der bessere Wert wird als Wert für das Modell genommen und es wird abgespeichert, ob das Modell dazu gedreht werden muss. Damit kann beim Identifizieren auch gerade die Orientierung des Raumes festgestellt werden.

#### 5.6.2.3 Vergleichsfunktion
Die Vergleichsfunktion prüft zuerst, ob die Dimensionen des Raums und des Modells übereinstimmen. Ist der Dimensionsunterschied mehr als einen halben Meter, so wird das Modell aussortiert.

Im zweiten Schritt wird der Unterschied zwischen dem Footprint des realen Raums und dem Footprint des Modells berechnet. Es gibt zwei Möglichkeiten für unterschiedliche Messungen.

Erstens, wenn im Modell eine Wand detektiert wurde und im realen Raum keine. Dieser Unterschied wird nicht stark bewertet. Wir sehen im realen Raum viele Löcher im Umgebungsmesh. Das sind Fehler, die häufig auftreten.

Die zweite Möglichkeit ist, wenn im realen Raum eine Wand detektiert wurde, aber im Modell keine vorhanden ist. Dies ist ein starkes Indiz, dass das Modell nicht auf den realen Raum passt. Dieser Fehler wird zehnmal so stark bewertet, wie der erste Fehler.

#### 5.6.2.4 Ergebnis
Aus den Modellen wird dasjenige ausgewählt, das den kleinsten Unterschied aufzeigt.

Solange die Dimensionen übereinstimmen, wird also ein Ergebnis gefunden. Das bedeutet aber auch, dass die Applikation nur in Räumen genutzt werden soll, die eingespeichert sind. Sonst könnte es zur falschen Lokalisierung führen. Es wäre natürlich möglich, den Unterschied des Footprints zu analysieren und bei einem zu grossen Unterschied das Modell als nicht erkannt einzustufen. Dazu fehlen jedoch die Erfahrungswerte. Wir haben zwei Räume modelliert, was jeweils zu vier Werten führt, von welchen ein Wert der korrekte Identifikationswert ist.

### 5.6.3 Modell positionieren
Nachdem das korrekte Modell und seine Orientierung bestimmt wurden, muss das Modell an den korrekten Platz im realen Raum gestellt werden. Das ist mit den Unity Werkzeugen keine allzu grosse Sache. Die Rotation zwischen dem realen Raum und dem Modell wird bestimmt und korrigiert. Dann wird das Modell an die korrekte Stelle gerückt.

Hier ist noch anzumerken, dass die Rotation und Translation nicht nur auf das Modell des identifizierten Raumes angewandt wird, sondern auf alle Räume. Sind sie nämlich in Unity korrekt angeordnet, stimmen sie dann auch mit der Realität überein.

### 5.7 MiniMap

Das MiniMap Modul ermöglicht es, das Modell per Sprachkommando zu einer MiniMap zusammenschrumpfen zu lassen. Das kleine Modell kann dann für eine bessere Betrachtung auf einer Oberfläche platziert werden. Im Modell wird die Position des Benutzers im Raum mit einer kleinen Sonne markiert.

Das Modul stellt auch eine reibungslose Transition zwischen dem MiniMap Modell und dem lebensgrossen Modell, der MaxiMap, sicher.

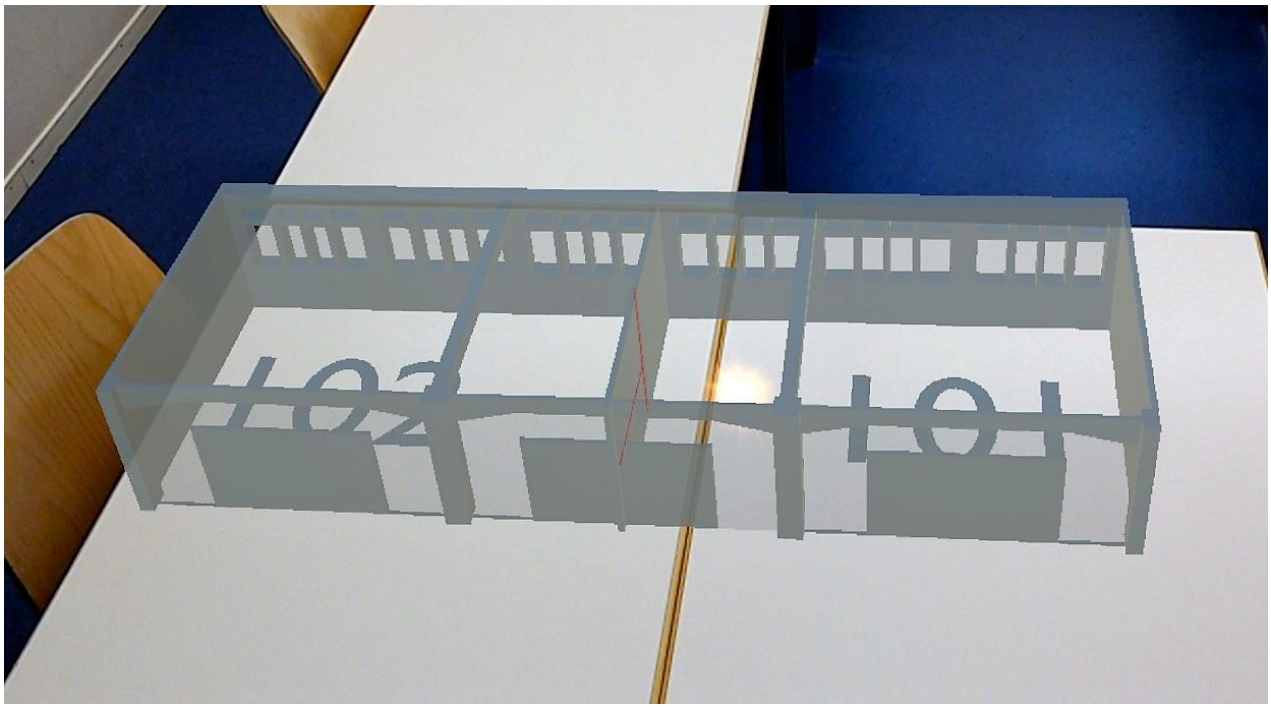


Abbildung 19: Die MiniMap

# 6 Projektmanagement

### 6.1 Planung

Die Planung der Tasks wurde mithilfe der Projects-Ansicht von GitHub verwaltet. In dieser wurden vier Spalten erstellt: Backlog, Weekly Spring, In Progress und Done. Die Tasks wurden laufend erfasst, um wichtige Aufgaben nicht zu vergessen. Die genaue Planung und der Zwischenstand wurden jeweils in Person besprochen. Dies führte teilweise dazu, dass die Planungsansicht nicht den genauen Stand der Dinge repräsentierte.

### 6.2 Arbeitsjournal

Während der Arbeit wurden die Tätigkeiten der Woche regelmässig in kurzen Sätzen festgehalten.

Die Arbeit wurde jeweils pro Woche rapportiert. Die Arbeitstage variierten und wurden unter den zwei Tagen in der Woche und dem Wochenende aufgeteilt. Freitags wurde im Gründerzentrum gearbeitet und das wöchentliche Meeting mit dem Betreuer abgehalten.

#### 6.2.1 Woche 1, 22. September 2017

- Die Gebäudepläne des Gründerzentrums wurden beschafft.
- Die benötigten Entwicklungs-Tools wurden installiert oder auf den neuesten Stand gebracht.
- Das Gründungszentrum wurde mit der HoloLens gescannt und das Resultat gesichtet.

#### 6.2.2 Woche 2, 29. September 2017

- Beide haben das SpatialMapping/Understanding Tutorial durchgearbeitet.
- Bestehende CAD und 3D Modellierungs-Software wurde analysiert. Für unsere Zwecke haben wir uns für Blender entschieden.
- Das Gründerzentrum wurde mit einem Laser-Messgerät ausgemessen.

#### 6.2.3 Woche 3, 6. Oktober 2017

- Die APIs für Spatial Mapping und Spatial Understanding wurden recherchiert.
- Ein Scanner-App wurde erstellt, die das Spatial Mapping API benutzt und visualisiert.
- Der Raum 107 des Gründerzentrums wurde in Blender modelliert.

#### 6.2.4 Woche 4, 13. Oktober 2017

- Der Raum 107 des Gründerzentrums wurde mit Fenstern erweitert. Das Modell hat nun für unsere Zwecke genügend Details.
- PCL Arbeitsumgebung wurde eingerichtet.
- Das Spatial Understanding API wurde von Grund auf selbst angesprochen. Leider ohne grossen Erfolg.

#### 6.2.5 Woche 5, 20. Oktober 2017

- Die PCL Dokumentation wurde gesichtet und Tutorials durchgeführt. Folgende Themen wurden behandelt: surface normals, plane model segmentation, extracting indices, clustering.
- Das Spatial Understanding API wurde anhand des Example Projects getestet. Die Möglichkeiten und Limitationen wurden ausgelotet.

#### 6.2.6 Woche 6, 27. Oktober 2017

- Das PlaneFinding API wurde analysiert. Es wurde ein Programm erstellt, dass den Boden und die Decke erkennen kann.
- Eine Plane detection wurde in PCL umgesetzt.

#### 6.2.7 Woche 7, 3. November 2017

- Ein Hilfsprogramm wurde erstellt, welches entscheidet, ob genügend Umgebungsdaten gesammelt wurden, um mit der Auswertung zu starten.
- Die Normalen der Umgebungsdaten wurden für die weitere Auswertung kategorisiert.

#### 6.2.8 Woche 8, 10. November 2017

- Die Visualisierung des gefundenen Raumes wurde vorbereitet.
- Die ersten parametrisierten Wände wurden anhand der Umgebungsdaten platziert.

#### 6.2.9 Woche 9, 17. November 2017

- Die bisherigen Komponenten für Scan Fortschritt, Mesh Generierung und Mesh Analyse wurden aufbereitet und untereinander verbunden.
- Verschiedene Ansätze und Verbesserungen, um Wände aus den Umgebungsdaten zu extrahieren wurden besprochen.
- Ein Ansatz, der die Wand progressiv präzisiert und stabilisiert, wurde umgesetzt.

#### 6.2.10 Woche 10, 24. November 2017

- Die vier Wände des Raumes wurden erkannt und die Raumdimensionen konnten ausgelesen werden.

#### 6.2.11 Woche 11, 1. Dezember 2017

- Verschiedene Ansätze zur Identifikation eines Raumes wurden analysiert.

#### 6.2.12 Woche 12, 8. Dezember 2017

- Der Raum wird anhand von Scheiben identifiziert. Die Scheiben sind für die HoloLens unsichtbar und sind daher als "Löcher" in der Wand erkennbar.

#### 6.2.13 Woche 13, 15. Dezember 2017

- Das Modell (Minimap) wird nun an der korrekten Stelle im physikalischen Raum platziert.

#### 6.2.14 Woche 14, 22. Dezember 2017

- Um den Scan Prozess mehr responsive zu gestalten, wurde eine Priority Queue implementiert, die die zuletzt fokussierten Oberflächen zuerst verarbeitet.

#### 6.2.15 Woche 15, 29. Dezember 2017

- Eine visuelle Hilfe wurde implementiert, um den User beim Scannen des Raumes zu unterstützen.

#### 6.2.16 Woche 16, 5. Januar 2018

- Alle Module wurden nochmals überprüft und diverse Bugfixes wurden implementiert.

#### 6.2.17 Woche 17, 12. Januar 2018

- Dokumentation
- Video
- Präsentation

# 7 Entwicklungsumgebung

In diesem Kapitel soll festgehalten werden, welche Software genutzt wurde und wie diese installiert wurde. Dies soll einem Leser dieser Dokumentation ermöglichen, die nötige Entwicklungsumgebung einzurichten, um die Entwicklung der Applikation weiterzuführen.

## 7.1 Unity

Die Hauptapplikation der Entwicklungsumgebung ist Unity. Unity ist eine plattformübergreifende Game-Engine, welche hauptsächlich zur Entwicklung von 2D- & 3D-Videospielen und Simulationen genutzt wird.

Bei der HoloLens wird Unity genutzt, um die Applikation zu erstellen und auf die Brille zu laden.

### 7.1.1 Genutzte Version

In unserer Entwicklungsumgebung wurde folgende Version genutzt:

**Unity 2017.1.1**

## 7.2 MixedRealityToolkit-Unity

Um die Arbeit mit der HoloLens und Unity zu erleichtern, bietet Microsoft das **MixedRealityToolkit-Unity** an. Dieses Toolkit ist eine Sammlung von Skripten und Komponenten, welche den Einstieg in die Entwicklung von HoloLens oder Mixed Reality Applikationen erleichtert.

Das gesamte Toolkit wird auf GitHub zur Verfügung gestellt und wird stetig weiterentwickelt, um kompatibel mit den aktuellsten Unity Versionen zu bleiben.[14]

### 7.2.1 Genutzte Version

In unserer Entwicklungsumgebung wurde folgende Version genutzt:

**HoloToolkit-Unity-v1.2017.1.1.unitypackage**

## 7.3 Visual Studio

Im Zusammenspiel mit Unity wird Visual Studio benötigt um die C#-Skripte zu erstellen und zu bearbeiten.

Visual Studio ist die von Microsoft zur Verfügung gestellte Entwicklungsumgebung. Sie unterstützt viele verschiedene Hochsprachen und ermöglicht es, je nach Bedarf verschiedene Module zu installieren.

Unser Visual Studio wurde konfiguriert, um mit C# und C++ umzugehen.

### 7.3.1 Genutzte Version

In unserer Entwicklungsumgebung wurde folgende Version installiert:

**Visual Studio 2017 Version 15.3.5**

[14] https://github.com/Microsoft/MixedRealityToolkit-Unity

### 7.4 Installationsanleitung

Nach dem die nötigen Installationsdateien zur Verfügung stehen, müssen sie folgenderweise installiert und konfiguriert werden:

1. Entwicklermodus aktivieren *(Windows 10)*
    a. *Action Center → All Settings → Update & Security → For Developers*
    b. *Enable Developer mode* aktivieren.

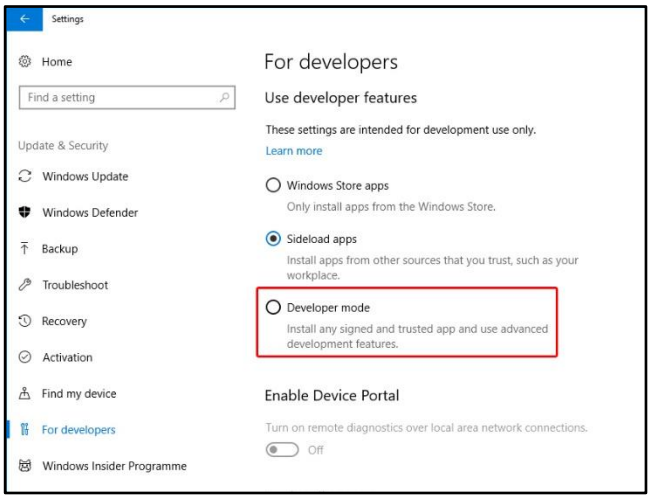



Abbildung 20: Developer mode in Windows 10 aktivieren

2. Visual Studio Installation durchführen
    a. Bei der Auswahl der Komponenten folgende Auswahl treffen:

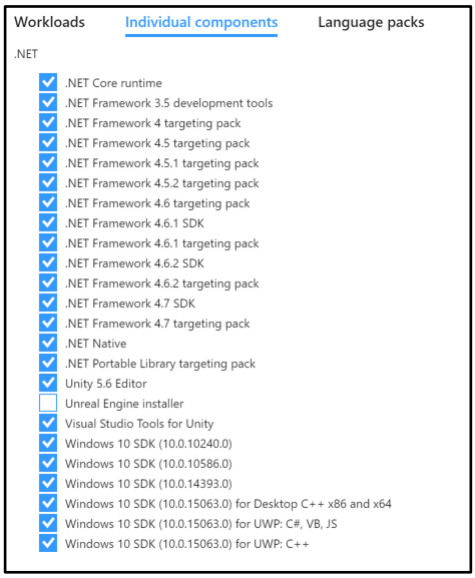



Abbildung 21: In Visual Studio genutzte Komponenten

3. Unity Installation durchführen
    a. Bei der Auswahl der Komponenten folgende Auswahl treffen:

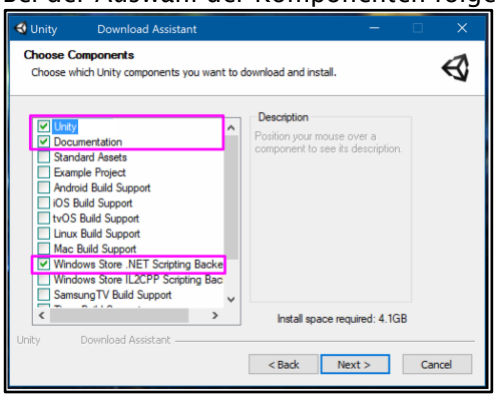



Abbildung 22: Installierte Unity Komponente

4. Via Github die aktuellste Version der Applikation klonen
    a. Zu finden auf https://github.com/shylux/HoloLens-BIM
    b. Sicherstellen, dass der ***master*** Branch ausgecheckt ist.

5. Das Unity Projekt öffnen
    a. ***Room-Box*** Ordner auswählen

6. MixedRealityToolkit-Unity package hinzufügen
    a. In Unity navigieren durch:
    *Assets → Import Package → Custom Package…*
    b. Das heruntergeladene unitypackage auswählen und importieren.

7. Das neu vorhandene HoloToolkit Menu verwenden
    a. *Configure → Apply HoloLens Scene Settings → Apply*


Abbildung 23: HoloLens Scene Settings

    b. *Configure → Apply HoloLens Project Settings → Apply*

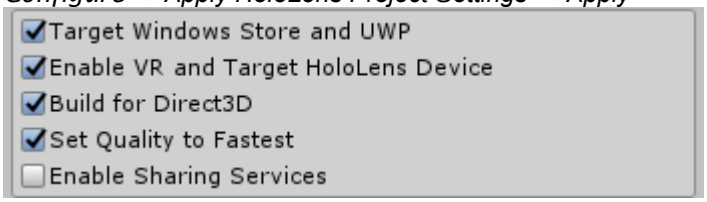



Abbildung 24: HoloLens Project Settings

c. *Configure → Apply HoloLens Capability Settings → Apply*

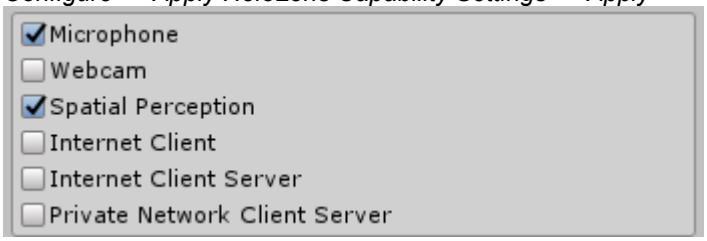



Abbildung 25: HoloLens Capability Settings

8. Project in Unity *builden*
   a. *File → Build Settings...*

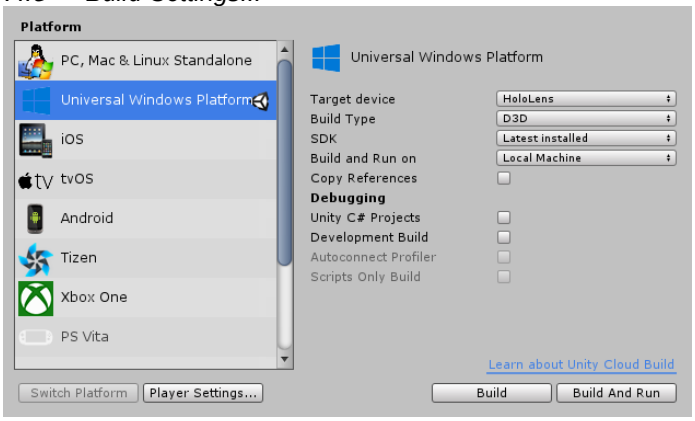



Abbildung 26: Build Settings in Unity

   b. *Build*

9. Visual Studio Project starten
   a. Im unter 8. angegebenen Ordner HoloLens-BIM.sln ausführen.

10. App auf die HoloLens deployen
    a. Via Dropdowns folgende Einstellungen vornehmen
       i. Debug
       ii. X86
       iii. Device

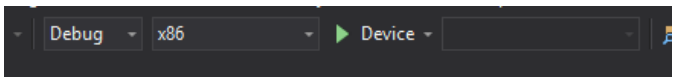



Abbildung 27: Deployment-Einstellungen in Visual Studio

   *b. Debug → Start Without Debugging…*

11. Die App startet automatisch auf der HoloLens

# 8 Schlussfolgerungen/Fazit

Unsere Arbeit hat gezeigt, dass die Technologie der HoloLens in der Lage ist, im BIM-Umfeld genutzt zu werden. Das gescannte Mesh stellt genügend Informationen zur Verfügung, um die Wände eines Raumes zu identifizieren. Dies hat uns erlaubt, unsere Modelle mit einem gescannten Raum zur Übereinstimmung zu bringen.

Eine der Hauptschwierigkeit lag vor allem darin, diese Daten korrekt zu interpretieren. Wie es das Kapitel Mesh Analyse zeigt, wurde ein grösserer Teil unserer Zeit darin investiert, einen nutzbaren Ansatz dazu zu finden. Die Wahl und das Experimentieren mit den verschiedenen Möglichkeiten erwiesen sich dabei als zeitintensiver als geplant.
Der daraus resultierende Ansatz, es mit einer eigenen Implementation anzugehen bot uns die Möglichkeit, näher am Mesh zu experimentieren und eine bessere Kontrolle über die Resultate zu haben.

Gewisse Probleme, welche wir angetroffen haben, standen im Zusammenhang mit der Ausgabe der Resultate. Die HoloLens ist in der jetzigen Iteration nicht ideal um Hologramme auszugeben, welche grösser als das Sichtfeld sind. In solchen Situationen geht die Übersicht schnell verloren und die Position der Hologramme beginnt an Genauigkeit zu verlieren.

Weiter ist die Scandistanz von drei Metern momentan noch zu gering, um grössere Räume auf effiziente Art und Weise zu scannen. Der Vorgang dauert länger als ursprünglich erwartet und zwingt einem dazu, sich direkt vor jeder Wand zu positionieren. Somit wird das Scannen eines Raumes umständlicher, je grösser der Raum ist.

Als Abschluss können wir deshalb sagen, dass die HoloLens durchaus in der Lage ist, sich in einem Gebäude zu positionieren. Es wird jedoch ein besseres Sichtfeld und eine grössere Reichweite der Tiefensensoren benötigt, um eine solche Applikation möglichst genau und effektiv nutzen zu können.

Natürlich kann auch die Applikation noch weiterhin verbessert werden. Es gibt gewisse Ansätze und Verbesserungen, welche wir aus Zeitgründen nicht mehr angehen konnten. Wir haben zum Beispiel die Möglichkeit besprochen, die Wände anhand der Kreuzung mit der Decke oder dem Boden zu definieren.

Wir können uns aber sehr gut vorstellen, dass diese Technologie sich in den kommenden Jahren durchsetzen wird. Es kann gut sein, dass in nur wenigen Jahren diese Technologie wie eine gewöhnliche Brille aussieht und im alltäglichen Gebrauch unsere Umgebung erweitert.

# Abbildungsverzeichnis


Abbildung 1: Die HoloLens von Microsoft 6
Abbildung 2: Frontansicht der HoloLens 7
Abbildung 3: Übersicht der HoloLens-Sensoren 9
Abbildung 4: Beispiel einer Fotoaufnahme mit der HoloLens 10
Abbildung 5: Plan der BFH-Räume an der Wankdorffeldstrasse 14
Abbildung 6: Model eines Raumes, erstellt in Blender 15
Abbildung 7: Erreichtes Resultat mit der PCL. In Rot sind die Punkte, welche den Boden ausmachen. 17
Abbildung 8: Das Mesh nach der Verwendung der Spatial Understanding API. 18
Abbildung 9: Scan Manager in Unity 22
Abbildung 10: Beispiel einer Surface (Oberfläche) 22
Abbildung 11: Beispiel einer gerade hinzugefügten Oberfläche (rot) 23
Abbildung 12: Scan Progress in Unity 24
Abbildung 13: Sichtbare Sensoren innerhalb der Unity-Szene 24
Abbildung 14: Erkannter Raum 26
Abbildung 15: Virtual Room Behavior in Unity 28
Abbildung 16: Bekanntes Modell innerhalb der Unity Szene 28
Abbildung 17: Beispiel eines Footprints im modellierten Raum 29
Abbildung 18: Beispiel eines Footprints im gescannten Raum 29
Abbildung 19: Die MiniMap 31
Abbildung 20: Developer mode in Windows 10 aktivieren 35
Abbildung 21: In Visual Studio genutzte Komponenten 35
Abbildung 22: Installierte Unity Komponente 36
Abbildung 23: HoloLens Scene Settings 36
Abbildung 24: HoloLens Project Settings 36
Abbildung 25: HoloLens Capability Settings 37
Abbildung 26: Build Settings in Unity 37
Abbildung 27: Deployment-Einstellungen in Visual Studio 37


# 9 Tabellenverzeichnis


Tabelle 1: Hardware Spezifikationen der HoloLens ..... 12

# 10Literaturverzeichnis


**Windows Dev Center - Holograms 230**
https://developer.microsoft.com/en-us/windows/mixed-reality/holograms_230

**GitHub - MixedRealityToolkit-Unity - Saving Spatial Meshes**
https://github.com/Microsoft/MixedRealityToolkit-Unity/tree/master/Assets/HoloToolkit-Examples/SavingSpatialMeshes

**Capture A Real Room For The HoloLens Emulator**
http://hololenshelpwebsite.com/Blog/EntryId/1009/Capture-A-Real-Room-For-The-HoloLens-Emulator

**Case study - Expanding the spatial mapping capabilities of HoloLens**
https://developer.microsoft.com/en-us/windows/mixed-reality/case_study_-_expanding_the_spatial_mapping_capabilities_of_hololens

**Windows Dev Center - Spatial mapping**
https://developer.microsoft.com/en-us/windows/mixed-reality/spatial_mapping

**Using a HoloLens scanned room inside your HoloLens app**
http://dotnetbyexample.blogspot.ch/2017/02/using-hololens-scanned-room-inside-your.html

**Blender For Noobs - Modelling for beginners - Part 10 of 10**
https://www.youtube.com/watch?v=ZhAyeJaiiVQ

**Unity Spatial Mapping Manual von Windows Holographic**
https://docs.unity3d.com/Manual/windowsholographic-spatialmapping.html

**Windows Mixed Reality Forum - Depth Camera**
https://forums.hololens.com/discussion/comment/3204#Comment_3204

**Spatial Understanding in Unity**
https://developer.microsoft.com/en-us/windows/mixed-reality/spatial_mapping_in_unity#higher-level_mesh_analysis:_spatialunderstanding

**Spatial Mapping case study**
https://developer.microsoft.com/en-us/windows/mixed-reality/case_study_-_expanding_the_spatial_mapping_capabilities_of_hololens

**HoloLens - Introduction to the HoloLens, Part 2: Spatial Mapping**
https://msdn.microsoft.com/en-us/magazine/mt745096.aspx

**PCL - Estimating Surface Normals in a PointCloud**
http://pointclouds.org/documentation/tutorials/normal_estimation.php

**PCL - Plane model segmentation**
http://pointclouds.org/documentation/tutorials/planar_segmentation.php

**PCL - Extracting indices from a PointCloud**
http://www.pointclouds.org/documentation/tutorials/extract_indices.php

**PCL - Conditional Euclidean Clustering**
http://www.pointclouds.org/documentation/tutorials/conditional_euclidean_clustering.php

**PCL - VoxelGrid**
http://pointclouds.org/documentation/tutorials/voxel_grid.php

# Erklärung der Diplomandinnen und Diplomanden
*Déclaration des diplômant-e-s*

## Selbständige Arbeit / *Travail autonome*

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

*Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.*

| | |
|---|---|
| Name/*Nom*, Vorname/*Prénom* | Lukas Knöpfel..... |
| | Vincent Genecand..... |
| Datum/*Date* | ..... |
| Unterschrift/*Signature* | ..... |
| | ..... |

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
*Ce formulaire doit être joint au rapport de la thèse de bachelor.*