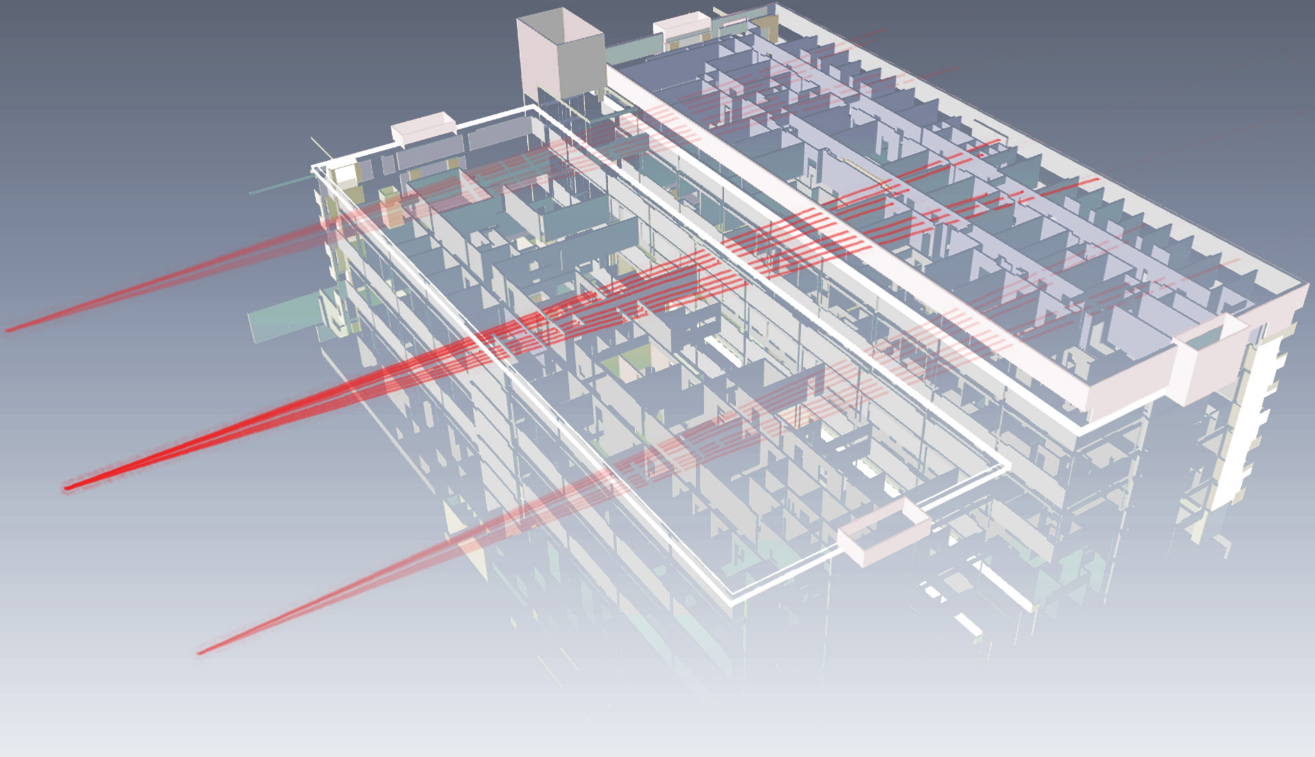
MSc Thesis in Geomatics for the Built Environment

TU Delft

Simplification & Visualization of BIM models through Hololens



Panagiotis Karydakis

October 2018

SIMPLIFICATION AND VISUALIZATION OF BIM MODELS THROUGH HOLOLENS

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Panagiotis Karydakis

October 2018

ISBN 999–99–9999–999–9

The work in this thesis was made in the:

3D geoinformation group
Department of Urbanism
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors:	ir. Stelios Vitalis
	Dr.ir. G.A.K. (Ken) Arroyo Ohori
Co-reader:	ir. Rusne Sileryte
Delegate from the Board of Examiners:	ir. W. Willers

ABSTRACT

Augmented Reality (AR) is the technology that superimposes digital generated objects on the physical world. It has the potential to create new products and services, like visualizing future buildings or objects that will decorate a place, which creates new opportunities for applications in both the public and private sector. The architecture and construction fields are particularly interested in investigating this innovative technology to engage stakeholders, such as designers and engineers, in every stage of decision making and to minimize discrepancies between the original design of a building and the final outcome. Portability is one of the greatest advantages of Head-Mounted Display (HMD) AR technologies, but there are limitations regarding the amount of data which can be visualized using the computational power of the current generation of devices. Also, automated methods and approaches that can cope with the intricacy of the models have to be produced and tested to make the usage of augmented reality feasible.

Within this thesis, a methodology is developed to isolate each storey of a Building Information Modeling (BIM) model and its exterior envelope. Firstly, the model was converted, from the `Revit` file format, to the open standard Industry Foundation Classes (IFC), which made the file human readable. Only semantic information was used to isolate each storey of the building, while for the extraction of the outer shell geometrical calculations were performed. This extraction took place by sending rays from one side of the model to the other and checking the intersection of the rays with the model.

Afterwards, the storeys and the exterior were visualized through an AR device, the `Hololens`. The `Unity` platform, which provides many tools for holographic development, was used for the configuration of the scene where the final user will interact with the created models. Scripts in the `C#` programming language were developed to allow interaction between the user and the holograms. I created a simple and intuitive menu, consisting of `3D` buttons to allow the user to visualize only the desired parts of the model. After having visualized the model, the user has the ability to scale and rotate the model using the corresponding buttons. In the same way, when the user stares at an element of the building, this element is highlighted and by making the tap gesture, he/she can visualize metadata information about this element as text above the model. Finally, spatial perception functionality is provided by virtually placing the model on horizontal planes identified by the device.

The proposed methodology was tested in a use case on a sample BIM model, and specifically of the Amsterdam Medical Center. The large size of the file and the high complexity of its geometry made the model a challenging test that made it possible to highlight the limitations and efficiency of the developed approach. Despite the positive results of the process, the accuracy is affected by the computational power of the current generation of hardware. Nevertheless, the clear perception of a construction coupled with the interactions capabilities provide an immersive experience which can actively involve the user with the visualization process.

ACKNOWLEDGEMENTS

The accomplishment of this thesis project was not an individual work but a team effort and a few people assisted in one way or another to the final result.

I would like to express my gratitude to my supervisors both from TU Delft and SWECO. Particularly, I would like to thank *Stelios Vitalis*, my main mentor, for his invaluable recommendations during the execution of the project and for his patience and willingness to help with the complicated problems that popped up during this project. Furthermore, I would like to express my appreciation to *Ken Arroyo Ohori* for his guidance and support, especially during the phase of the isolation of the exterior shell considering his inspirational ideas. SWECO's people gave me the great opportunity to work in an excellent environment and to gain crucial working experience. *Henri Veldhuis*, as conceptual supervisor, and *Marco Grimaudo*, as technical supervisor, critically contributed to the specification of the topic and to the configuration of the visualization through the Hololens. I would also like to thank Stefanos Koufopoulos, a great architect and a close friend of mine, for the editing of the report's covers.

Finally, I would like to thank my family and notably my mother for supporting me all the way through the project and for always being there for me. My last thanks goes to my friends for helping me relax when I was too stressed. . . .

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

GIS	Geographical Information System	12
GUI	Graphical User Interface	48
BIM	Building Information Modeling	v
CADs	Computer-Aided Designs	8
AEC	Architecture, Engineering and Construction	1
AR	Augmented Reality	v
VR	Virtual Reality	1
HMD	Head-Mounted Display	v
MR	Mixed Reality	12
IFC	Industry Foundation Classes	v
CSG	Constructive Solid Geometry	xi
B-rep	Boundary Representation	xi
GPS	Global Positioning System	11
AV	Augmented Virtuality	11
MRLab	Mixed Reality Systems Laboratory	12
ISMAR	International Symposium on Mixed and Augmented Reality	12
IMU	Inertial Measurement Unit	13
UV	Ultraviolet	13
WLAN	Wireless Local Area Network	14
CMOS	Complementary Metal–Oxide Semiconductor	14
HPU	Holographic Processing Unit	14
GPU	Graphics Processing Unit	14
HD	High Definition	14
LoD	Level of Detail	17
CityGML	City Geography Markup Language	17
RGB	Red-Green-Blue	14
AMC	Academisch Medisch Centrum	xii
CGAL	Computational Geometry Algorithms Library	44
API	Application Programming Interface	45
FTP	File Transfer Protocol	46
SFTP	Secure File Transfer Protocol	46
IDE	Integrated Development Environment	46
IHMD	Immersive Headsets	47
UX	User Experience	47
FPS	frames per second	49
CPU	Central Processing Unit	66

1 INTRODUCTION

More and more companies are willing to move great parts of their design and construction procedures into computers. Their main target is to replace traditional tools with digital prototypes in order, through automation, to achieve cost and time minimization. During the last decade, serious advances have been accomplished with regard to the processes of conversion of traditional means into new formats in construction field.

The participation of people of multidisciplinary backgrounds in the planning process demands new tools which have the capability to host information derived from many fields of expertise. The identified model that facilitates with that is the Building Information Modeling (BIM), since it is a 3D digital structure that can host information by diverse sources. BIM models not only intend to the visualization of a construction but they can also host information from multiple disciplines, like mechanical & electrical engineering, providing a holistic digital representation of a building. The usage of BIM models in the Architecture, Engineering and Construction (AEC) industry will support enhanced construction industry performance through expanded cooperation among project parties, less inefficiencies and rework on alterations and improvements of the final result.

Figure 1.1: Digital generated objects are overlaid on the real environment[1]

Despite the advantages that BIM models can offer in the construction industry, the rapidness with which this technology is expanding and its potentials to gather large amount of data, triggers worries of how people in the field can handle the quantities of information reaching them [Chu et al., 2018]. Augmented Reality (AR) technology has been identified as the mean to facilitate the visualization of essential information for stakeholders, in an intuitive and simple way.

Augmented Reality (AR) is the technological advance that allows digital objects and images to be overlaid on the real environment in real time (Figure 1.1)[1]. It offers a realistic experience since the user can interact with the holograms using physical objects in a seamless way, in contrast to Virtual Reality (VR), with which the user is perfectly immersed in a virtual world. Among other applications, AR can be utilized

1 http://greenbuzzagency.com/attract-audiences-with-augmented-reality/

Figure 1.2: Visualization of holograms though Microsoft Hololens[2]

in the field of architecture to visualize building models in order to give to the designers a clear perception of the future construction. In this project, the `Microsoft HoloLens` is used as a Head-Mounted Display (HMD) AR device (Figure 1.2)[2]. It has the capability to reconstruct the surrounding space by creating a spatial model of meshes and to identify a number of gestures permitting interaction with the users.

1.1 PROBLEM STATEMENT

As every other 3D model, BIM models can be highly beneficial throughout the AEC life-cycle, since they allow designers to obtain a clear perception of their designs. This awareness allows them to amend or to adapt the models before their creation.

Despite the capability of the designers to correct their projects, construction industry is characterized by a high rate of rework. Rework is the cause of the 52% of the whole increase of incurred costs and 22% of time excess of the predefined tasks [Love Peter E. D., 2002]. Furthermore, costs due to rework have been calculated to account for 5% - 20% of the contract value in building projects and design discrepancies could be the reason of 50% of the rework that occurs [Barber et al., 2000; Love and Edwards, 2004]. In addition to that, it has been noted by Love and Edwards [2013] that design errors and/or omissions have a slightly greater effect on cost growth than other determinants.

The facts mentioned above highlight the demand for visualization of the final outcome before of the start of the construction project. Even today the most common means used to visualize architectural designs are either posters (Figure 1.3a)[3] or screens (Figure 1.3b)[4]. None of them gives a clear and intuitive perception of the designed building to people who are not familiar with this kind of projection means. Another quite common method of visualization of the future construction are maquettes (Figure 1.4a)[5] which are physical 3D models of the building which is about to be created(Figure 1.4b)[6]. Maquettes give a clear perception of the future construction and can assist in evaluation of the final outcome which improves collaboration among the stakeholders and minimizes the rework. They are combined though with a number of limitations like, the fixed scale, portability, time & money needed for their creation and demand for experienced personnel for their construction.

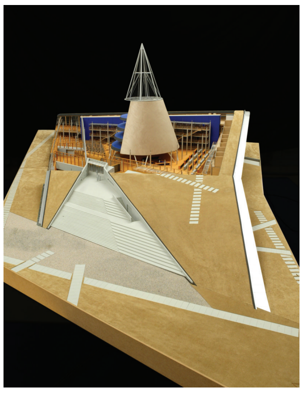
Augmented Reality (AR) devices, could be the mean to visualize only the desired parts of building models in a more efficient way considering time and cost. How-

2 https://www.intellectsoft.net/blog/microsoft-hololens-usage-in-construction/
3 https://www.pinterest.co.uk/mr_m8/sheet-composition/
4 https://www.gelement.com/category/trade-shows/page/3/
5 https://www.pinterest.co.uk/pin/488359771908253 22/
6 http://www.appliedlogictudelft.nl/event/categories-logic-linguistics-sociology/

(a)

(b)

Figure 1.3: Traditional means of visualization. (a) Architecture poster[3]. (b) Screens for 3(2)D visualization[4].


(a)

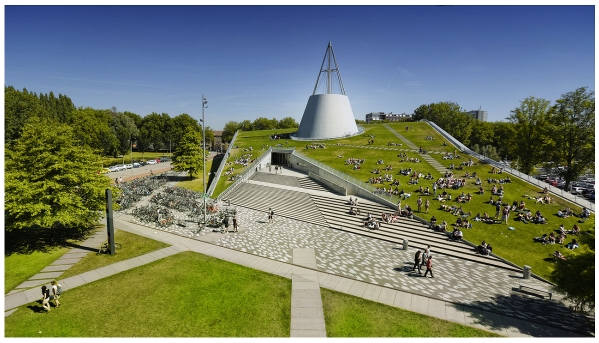

(b)

Figure 1.4: TU Delft library. (a) Maquette[5]. (b) Original building[6].

ever, `Hololens`, which is the best commercial available AR device up to now, comes with limited computational power. That makes the visualization of complex and verbose building models, like BIM models, impossible. BIM models contain large amount of both geometric and semantic information which might not be always relevant with the purpose of the visualization. So, prerequisite for their visualization through `Hololens` is the extraction of the wanted parts. One of these parts usually is the exterior envelope. The extraction of a geometrically complicated exterior shell, which will serve as a digital maquette of the construction, is not a straight forward process with regard to the steps that have to be followed to achieve it. In the same way, a feasibility investigation considering either a semantic or a geometric isolation of each floor has to be performed in order to provide a clear depiction of the interior of a BIM model. Finally, given the performance limitations of `Hololens`, it should be checked if the created holograms will provide interaction capabilities and will be possible to be visualized through it.

1.2 USE CASE

This thesis project aims to simplify a BIM model, in terms of extracting its exterior envelope & storeys and visualize them through an AR device. A sequence of steps

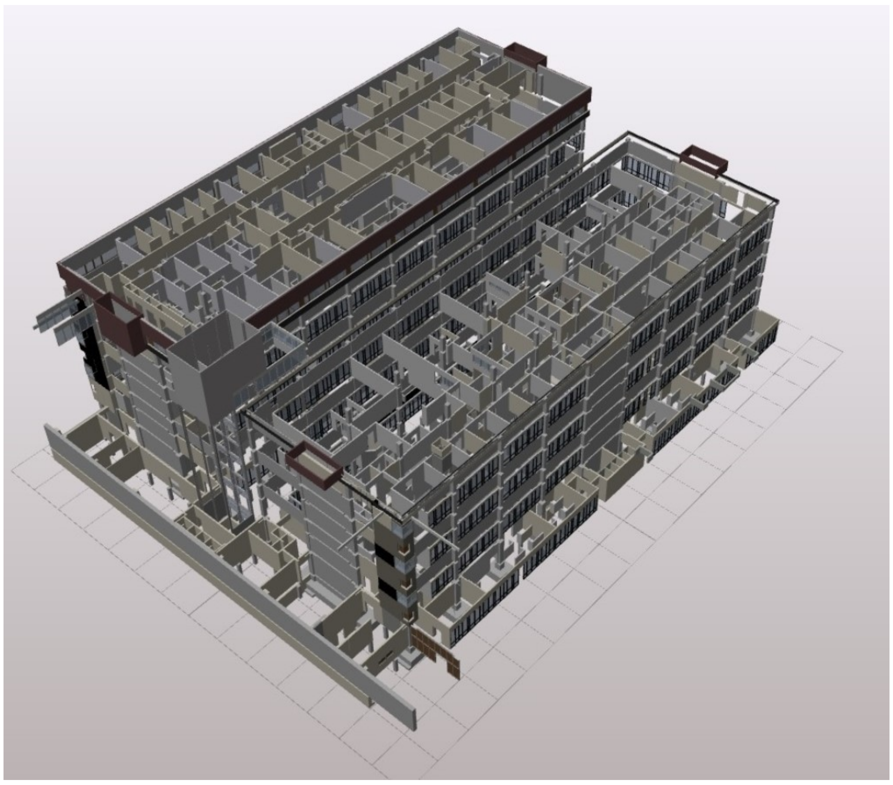


Figure 1.5: Original BIM to be projected through Hololens

was specified in order to achieve that and a complicated model was used to verify the validity of the methodology. The sample dataset, which was used during this project, was given by `SWECO`, one of the largest engineering companies in Europe. It wants to give a holographic illustration of a BIM model combined with interaction capabilities to its customers. The provided model is part of Academisch Medisch Centrum (AMC) containing many and complex geometries (Figure 1.5). Particularly, its original size was almost 128 `MB` while it contained approximately 778.000 faces. In addition to that, in the extensive cavity of its `U-shape` hosted many geometries of the exterior envelope that had to be extracted for the creation of the digital maquette.

Since the BIM model in its original form, it was impossible to be visualized through `Hololens`, the extraction of the outer shell was a necessity. The initial file format was proprietary and a conversion process was needed to transform it into Industry Foundation Classes (IFC). The intricate structure and size of the building will give a clear indication of the defects and imperfections of the suggested methodology.

1.3 RESEARCH OBJECTIVES

This thesis project aims to develop a methodology which provides the essential steps to simplify a complex BIM model and visualize it through an AR device. The proposed methodology has to be functional regardless of the size and the complexity of the building model. The generated hologram has to provide an intuitive perception of the model and to incorporate interaction with the user. The research question to be answered is:

How to simplify a rather complex BIM model and visualize it via an augmented reality device such as `HoloLens`?

Sub questions have been defined to facilitate answer the main research question:

1. Is semantic information enough to isolate building storeys from a BIM model?
2. What are the determinants and how do they affect the extraction of the exterior envelope?
3. How the user will interact with the hologram?
4. What are the limitations of `Hololens` with regard to the visualization of a building model?

1.4 SCOPE

1. During this project only open file formats are used for the isolation and extraction of desired building elements in order the developed approach to be tested and assessed by the academic community.
2. The developed proof of concept was created for and in combination with `SWECO`.
3. This prototype aims at visualizing buildings and buildings elements only through HMD devices as opposed to other approaches which intend to tablets and mobile devices.
4. Multiprocessing is a significant factor for the extraction of the exterior envelope. Many cores are needed for the production of an accurate model in an acceptable time period.
5. The dynamic connection of the AR device with other devices or the web is considered out of the scope of this project.
6. Although visualization of building objects of scale 1:1 is recognized as a challenging and interested topic, it is not examined by this thesis.

1.5 THESIS OUTLINE

The rest of the project is structured as below:

- Chapter 2 gives the needed theoretical background to understand the concepts behind the BIM & IFC structure and the AR technology. Furthermore the related work, considering the simplification of building models and the visualization of them through AR devices, is described.
- Chapter 3 illustrates all the steps followed to achieve the semantic isolation of each storey of the BIM model, the extraction of the exterior shell, the configuration of the holographic scene and the manipulation of the holograms.
- Chapter 4 gives information about the given sample dataset, the programming languages combined with the modules that were used and the software which was utilized during the project.
- Chapter 5 illustrates how the holograms were adjusted in the real environment, indicating how immersive the application is and functions as a proof-of-concept.
- Chapter 6 provides the conclusions as they were formed during the execution of the project, indicates some future work which was considered relevant and discusses some recommendations with regard to the IFC schema and `Hololens`.

2 THEORETICAL BACKGROUND & RELATED WORK

2.1 THEORETICAL BACKGROUND

This section provides all the information connected with the major concepts mentioned during this thesis project. This information is needed in order someone to be able to follow the next chapters.

Particularly, Section 2.1.1 refers to BIM's chronology, principles and concepts. Section 2.1.2 presents the IFC's architecture & entities and the geometric models describing its structure. Finally, Section 2.1.3 illustrates AR chronology & fields of applications and head-worn AR devices released up to now.

2.1.1 Building Information Modeling (BIM)

BIM models are geometrically accurate 3D representations of constructions. They provide complete governance of buildings' assets by hosting information from multidisciplinary fields about every phase of its life cycle.

The National Institute of building sciences and `buildingSMARTalliance` define BIM as a digital depiction of tangible and functional assets of a construction. So, it is considered as common knowledge storage for information about a building, shaping a robust foundation for decisions along its life cycle [NBIMS, 2015].

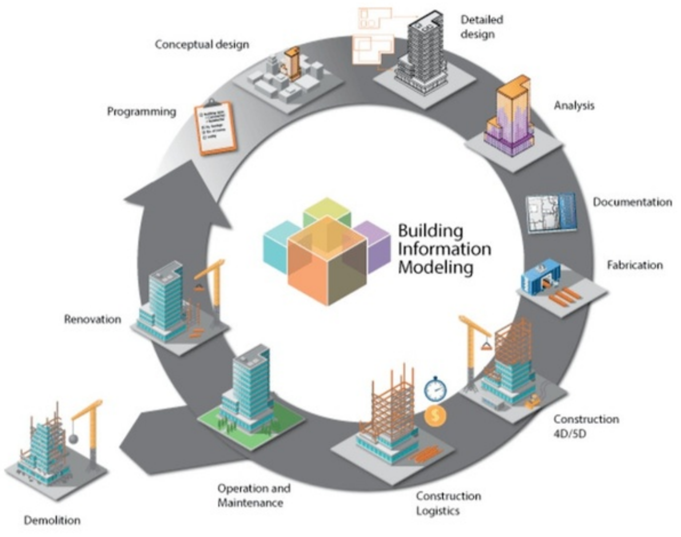


Figure 2.1: BIM: A multidisciplinary project of every phase of the life cycle of a construction[1]

Azhar et al. [2015] mention that although the roots of BIM models could be followed back to the parametric modeling studies operated in USA and Europe in late 1970s and early 1980s, the AEC industry essentially started to take advantage of it from the mid-2000s. Its adoption provided the ability to enhance the efficiency of the construction processes (Figure 2.1)[1] regarding time, cost & rework and collaboration among different parties, such as designers, engineers & workers who considered themselves as adversaries. This boosted cooperation inspires fusion of the

1 https://www.reuters.com/brandfeatures/venture-capital/article?id=34655

tasks of the stakeholders on a building project. So, BIM models affect the work-flow and project delivery processes and not only serve as three-dimensional intelligent models. The plethora of information which is included in these 3D intelligent models has guided to the suggestion of terms like 4D (referring to the addition of the time aspect to the model) and 5D (adding quantities and cost of materials) [AGC, 2006].

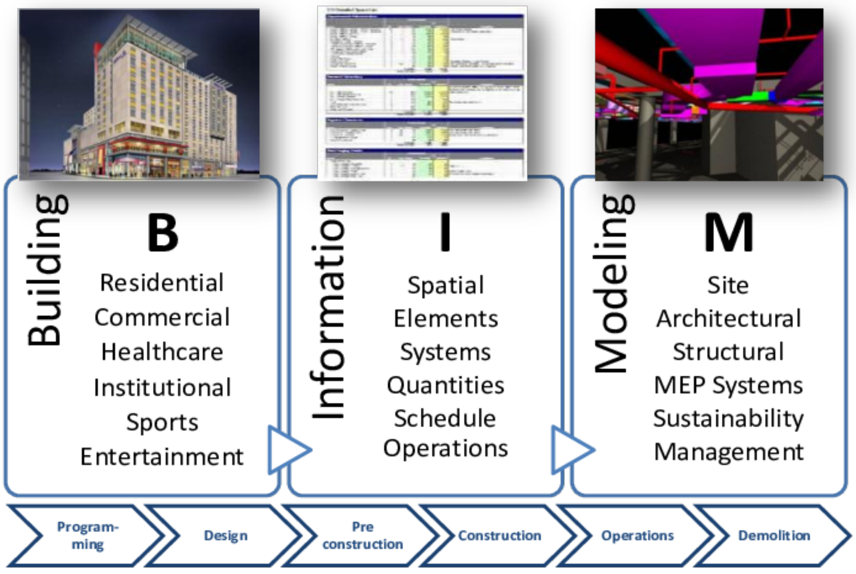


Figure 2.2: A description of BIM concept [Azhar et al., 2015]

BIM models' entities are described as "smart object" due to the load of information that they host. They are a source of intelligent contextual semantic information in contrast to traditional 3D Computer-Aided Designs (CADs) which only provide different 3D views, such as plans or sections. They do not carry any conceptual knowledge and someone has to treat them independently in case that any modification is needed, which is an error-prone procedure.

Created to facilitate every stage of building's life cycle, BIM involves in programming, designing, preconstruction, construction and post-construction (operations and maintenance) phases. During the creation of the model, each and every person, who is involved with the construction of the building, is steadily improving and adjusting his/her part according to the project specifications. The ongoing alterations assure that the model is correct enough in order the construction processes to start. Since, BIM aims at providing increased efficiency through enhanced communication and collaboration, its adoption demands early engagement of all people with a vested interest. Subsequently, the traditional way of proceeding though the required tasks has evolved in a more efficient and comprehensive approach with less inefficiencies and errors (Figure 2.3).

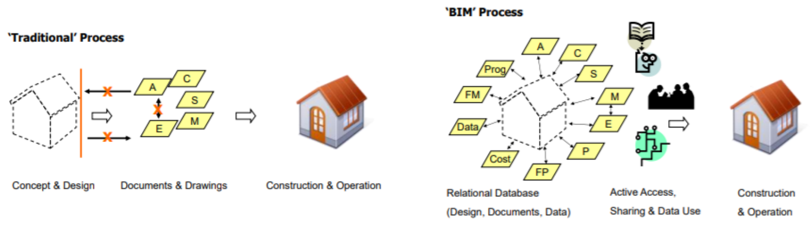
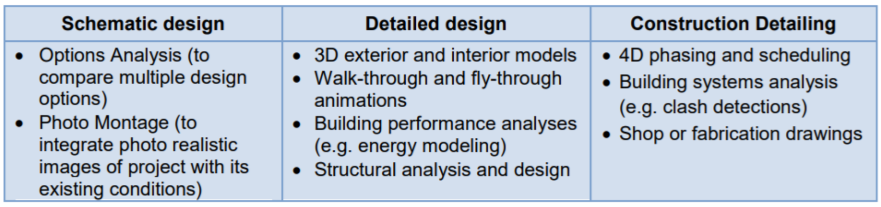


Figure 2.3: "Traditional" and "BIM" flow of tasks [Azhar et al., 2015]

Considering the design phase, 3 stages have been identified by Azhar et al. [2015] that can enhance the efforts of architects and engineers. These are Schematic design, Detailed design and Construction Detailing (Figure 2.4).



Schematic design	Detailed design	Construction Detailing
• Options Analysis (to compare multiple design options)	• 3D exterior and interior models	• 4D phasing and scheduling
• Photo Montage (to integrate photo realistic images of project with its existing conditions)	• Walk-through and fly-through animations	• Building systems analysis (e.g. clash detections)
	• Building performance analyses (e.g. energy modeling)	• Shop or fabrication drawings
	• Structural analysis and design	

Figure 2.4: BIM in design phase [Azhar et al., 2015]

2.1.2 Industry Foundation Classes (IFC)

IFC standard was produced by `buildingSMART` and it is a well-known data model intended for storing multidisciplinary information for construction life-cycle and exchanging it among people in the AEC industry. IFC is the only BIM open standard and it is related with two other `buildingSMART` standards which are International Framework for Dictionaries (IFD) and Information Delivery Manual (IDM) (Figure 2.5)[2].

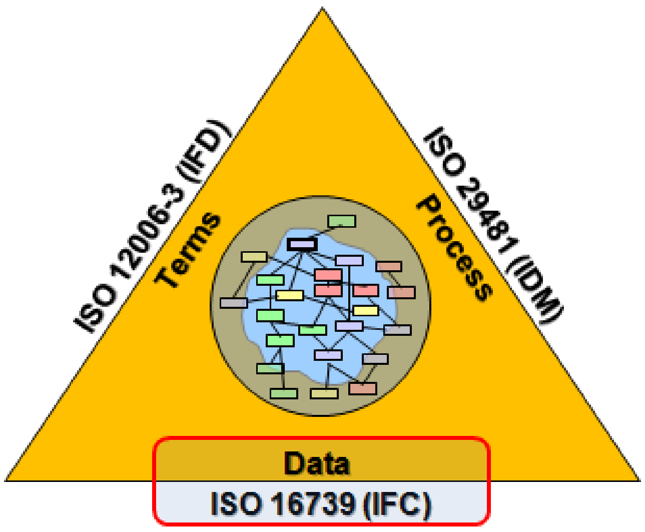


Figure 2.5: `buildingSMART`'s standards [2]

`EXPRESS` data definition language was used for the IFC specification[2] since it contains data validation rules. It is an international standard, `ISO10303-11`, used in multiple standards for the definition and exchange of product data.

According to Donkers et al. [2016] the most common IFC classes for building entities, and their relationships are shown in Figure 2.6. IFC schema, often called 'Product (Data) Model', is consisted of 1) entities, 2) attributes, 3) relationships between entities, defining how the elements of the model have to be structured in order to produce robust building representations. An `IfcObject`, one of the most fundamental entities of an `.ifc` file, and its sub-classes can be recursively decomposed into other `IfcObjects`.

IFC standard utilizes three main representation techniques to depict solid models. These are Constructive Solid Geometry (CSG), Boundary Representation (B-rep) and Sweep Representation.

2 `http://www.buildingsmart-tech.org/`

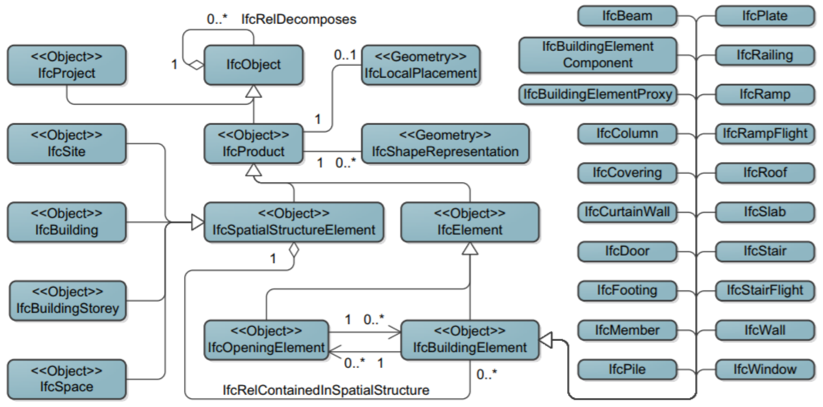
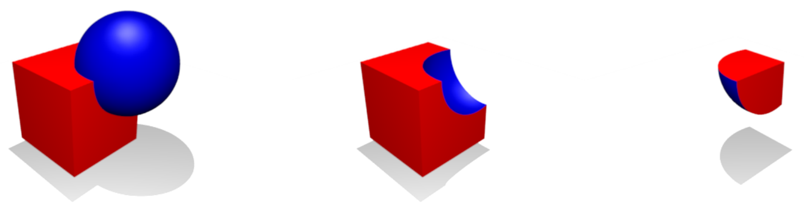


Figure 2.6: Most common IFC classes for buildings and their relationships [Donkers et al., 2016]



(a) (b) (c)

Figure 2.7: CSG Boolean operations. (a) Union. (b) Difference. (c) Intersection.[3]

- CSG: Constructive solid geometry is the technique that uses a limited number of primitive objects to produce more complex ones by performing a series of Boolean operations on the primitive solids (Figure 2.8a)[3]. Union, difference and intersection are the most common operations used for the formation of the final geometry (Figure 2.7)[3]. This series can be illustrated in a tree structure, having the leafs of the tree as the base solids (Figure 2.8a)[3].

- A swept volume is the space captured by a geometric shape as it moves along a trajectory (Figure 2.8b). This shape could be either a surface or a collection of surfaces. The trajectory is defined by an axis if it is a linear extrusion, by an axis and an angle or by using higher order sweep trajectories.

- B-rep structure represents a 3D object as set of vertices, edges and faces (Figure 2.9)[4]. It includes two types of information: topological and geometric. Topological indicates the relationship among vertices, edges and faces and the orientation of edges and faces, while geometric information is usually equations of the edges and faces. According Xu et al. [2007] the B-rep representation is more flexible than CSG and has a more inclusive operation set such as extrusion, chamfering, blending, drafting, shelling and tweaking

CSG and Sweep Representation are implicit types of modelling which means that only the parameters are stored in the IFC file to rebuild the geometry. In order the software to visualize the 3D objects needs to perform the appropriate computations using the given parameters. Figure Figure 2.10 depicts the logic that an IFC viewer follows in order to reconstruct the implicit geometry of CSG model stored in an IFC file. According to El-Mekawy and Östman [2010], the majority of IFC models use

3 https://en.wikipedia.org/wiki/Constructive_solid_geometry
4 https://artxinc.com/boundary-representation-in-computer-graphics.html

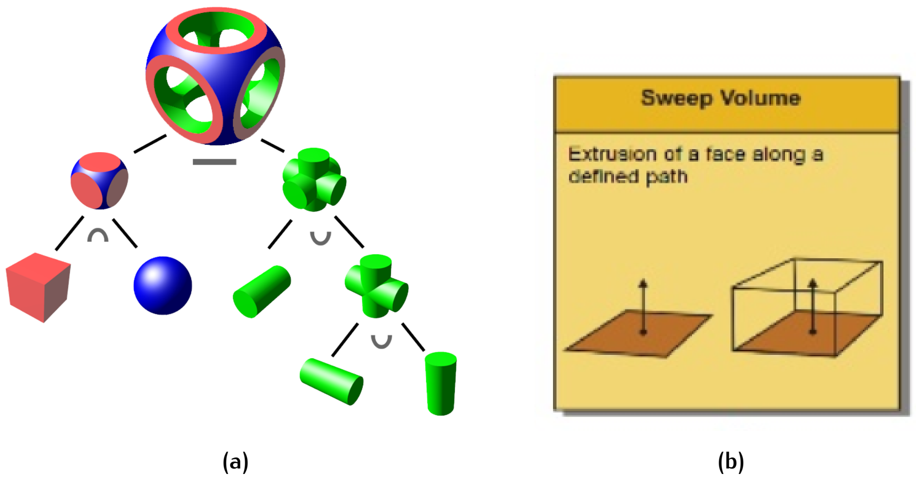


Figure 2.8: (a) CSG tree structure[3]. (b) Swept volume [Diakité, 2017]

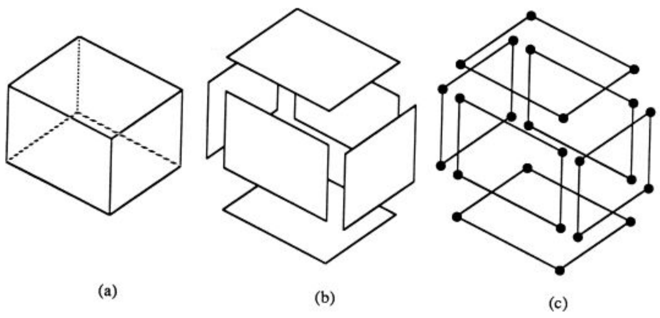


Figure 2.9: B-rep structure. (a) 3D Object. (b) Faces compose the solid. (c) Vertices and edges compose the faces.[4]

CSG and sweep representation to store the geometry of 3D objects. B-rep explicit geometry is not in great use and usually it does not lead to a unique solution; the number of created faces depends on the converter.

2.1.3 Augmented Reality (AR)

Augmented Reality (AR) is the technology that superimposes digital generated entities on the physical environment. Agarwal [2016] defines AR as a real time, immediate or indirect view of the real-world space whose reality is augmented by artificially-generated input such as sound, video, graphics, Global Positioning System (GPS) data etc. AR is just a part of the reality-virtuality continuum (Figure 2.11). The other two parts are VR and Augmented Virtuality (AV), in which physical objects are aggregated to virtual ones, which replace the real environment by an artificial one.

The graphics expert Ivan Sutherland was the first one who designed and constructed AR models, in collaboration with his students at Harvard University and the University of Utah in the 1960s, adopting a see-through technology to present the virtually designed objects. During the 1970s and 1980s quite a few organizations and institutes like `U.S. Air Force's Armstrong Laboratory`, the `NASA Ames Research Center`, the Massachusetts Institute of Technology and the University of North Carolina at Chapel Hill continued with the investigation of the AR field. The next decade was crucial for AR technology since, the reduction of the size of personal computers was achieved allowing platforms like tablet PCs and mobile phones to utilize this new feature. During the same decade, 1990s, the term

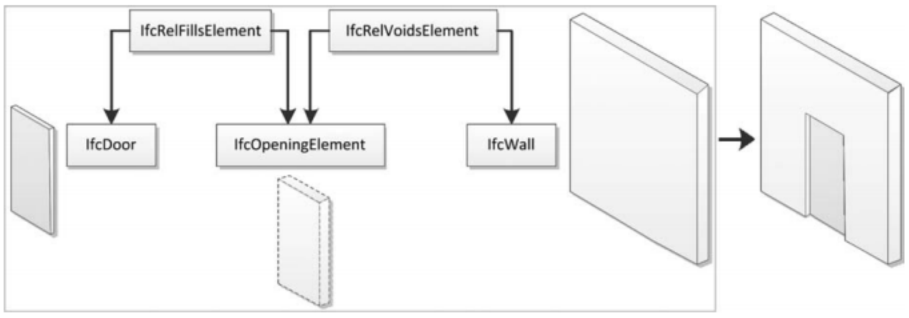


Figure 2.10: CSG structure for reconstructing IFC wall-door objects [Beetz, 2012]

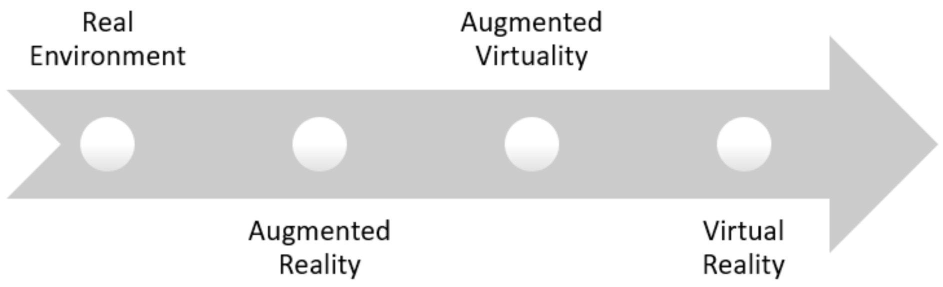


Figure 2.11: All 4 stages of Reality-Virtuality continuum

"Augmented Reality" specified by Caudell and Mizell [1992], researchers at `Boeing Corporation` working on achieving cost reduction and efficiency improvements in many of the human-involved operations in aircraft manufacturing, by eliminating templates, formboard diagrams and other masking devices. In 1993 Loomis et al. [1994] constructed the first AR navigation system for the visually impaired people. The application determined the user's position & orientation in the environment using GPS data and by obtaining information of the surrounding space from a Geographical Information System (GIS) database, was providing oral directions to the user (Figure 2.12).

After computers having became small-scaled and robust enough to project graphical, artificial objects in mobile devices, Feiner et al. [1997] constructed a mobile AR platform that superimposed information about Columbia's University campus, using a head-tracked, see-through, head-worn, 3D display, and an untracked, opaque, hand-held, 2D display. Conferences, like `International Workshop and Symposium on Augmented Reality`, the `International Symposium on Mixed Reality and the Designing Augmented Reality Environments` workshop, were initiated during 1990s and organizations, such as Mixed Reality Systems Laboratory (MRLab) in Nottingham and the `Arvika consortium` in Germany, were created. In 2002, the first International Symposium on Mixed and Augmented Reality (ISMAR) conference was held in Darmstadt of Germany, which is the leading international academic conference in the fields of AR and Mixed Reality (MR) to discuss limitations and solutions. The first AR applications focused on military, industrial and medical fields, but now more and more areas adopting this new technology to offer new experiences and improved performances.

As mentioned above, AR devices may be divided into three classes based on their position in relevance to the user and the real environment: head-worn, hand-held and spatial. Since during this master thesis a head-worn AR device used, a demon-

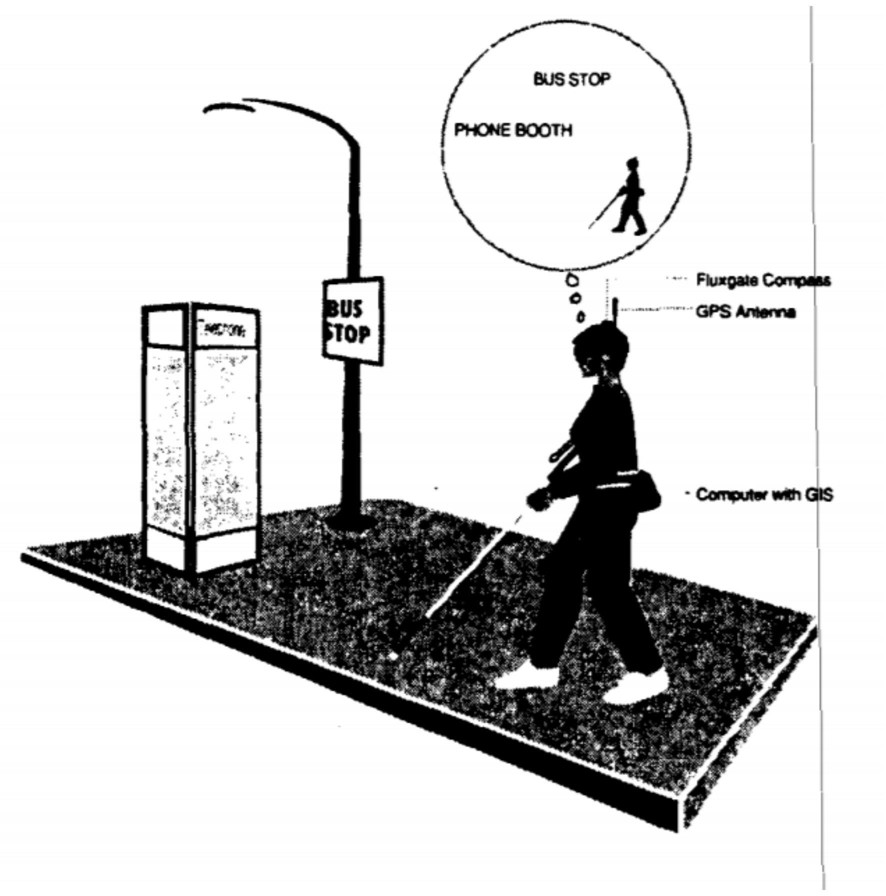


Figure 2.12: 1st AR application: a navigation system for the visually impaired people by Loomis et al. [1994]

stration of successful head-worn augmented reality devices, have been released up to now, considered relevant. As they reviewed by Loomis et al. [1994] are:

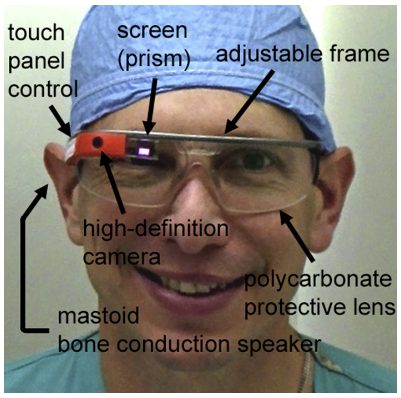
- `Google Glass` (Figure 2.13a) includes four sensors that could be utilized for activity identification: a camera, a microphone, an Inertial Measurement Unit (IMU) and an infrared proximity sensor looking at the users' eye, in order to discover the blinks of their eyes. Some of the functionalities that `Google Glass` provides are pictures taking, video calling and web searching in combination with the feature of visualization of a virtual screen when the user stares at top right.

- `Vuzix M100` (Figure 2.13b)[5] is an AR device in the form of eyeglass. It is a device with web access, speaker, camera and voice & touch controls. Its functionalities are web connection, capturing & projecting pictures, recording & listening sound, taping and playing videos via the display mounted in front of the right eye of the user.

- `Epson Moverio BT-200` (Figure 2.13c)[6] seems like an improved pair of eyeglasses. The device contains two screens and they are placed in the field of view of the user. Two removable Ultraviolet (UV) protectors are included to the device. It incorporates a movement tracking infrastructure, camera, sensors, touch-pad, Wi-Fi and Bluetooth connectivity.

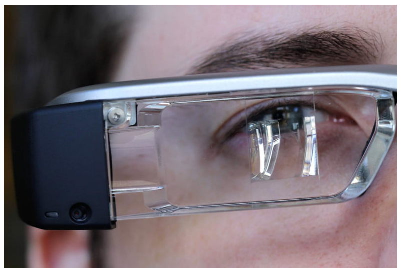
- `Sony SmartEyeglass` (Figure 2.13d)[7] superimposes text, symbols and images on the real space. It is a lightweight unit containing battery, speaker, mi-

5 https://androidcommunity.com/vuzix-m100-smart-glass-non-developer-pre-orders-begin-20131203/
6 https://www.digitaltrends.com/smartglasses-reviews/epson-moverio-bt-200-review/#/2
7 https://www.extremetech.com/electronics/199419-sony-is-releasing-its-own-ar-glasses-in-spite-of-google-glass-failure



(a) Google Glass [Muensterer et al., 2014].

(b) Vuzix M100 [5]



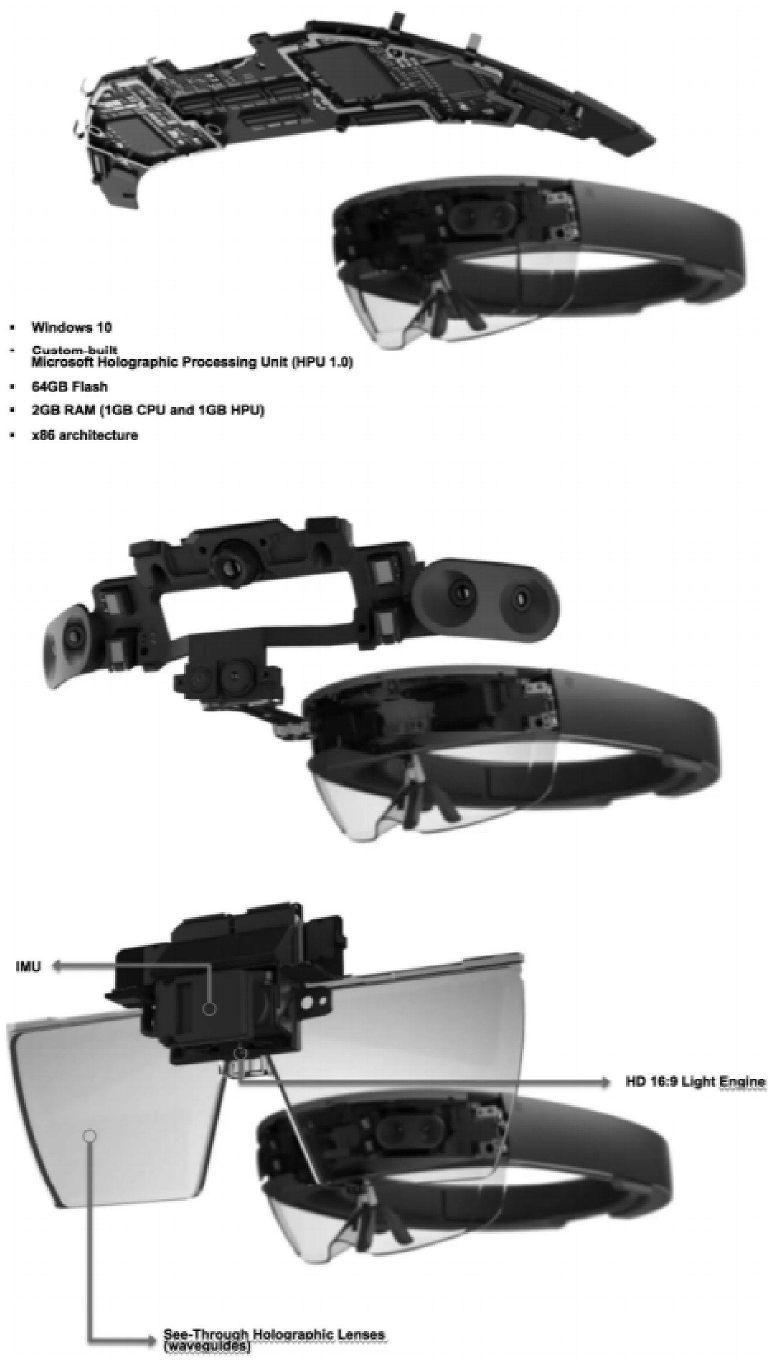
(c) Epson Moverio BT-200[6]

(d) Sony SmartEyeglass[7]

Figure 2.13: Commercially available HMD AR devices

crophone and touch buttons which facilitates the navigation. Furthermore, it contains a camera of 3 MP Complementary Metal–Oxide Semiconductor (CMOS) image sensor, accelerometer, gyro, electronic compass and brightness sensor. It can be connected with a smartphone over Bluetooth and Wireless Local Area Network (WLAN).

- Microsoft Hololens was first released in the market in 2016 supporting development functionalities. It is a holistic device which means that there is no need for another computer to be tethered with it to support its computational demands. HoloLens includes four Intel Atom x5-Z8100 1.04 GHz Intel Airmont Logical Processors, a Holographic Processing Unit (HPU)/Graphics Processing Unit (GPU) , 64 GB Flash, 2 GB RAM and provides 2-3 hours of battery life. These processors are used to run 2 High Definition (HD) 16:9 light engines that project light through holographic lenses achieving a total resolution of 2.3 million light points. HoloLens also contains an IMU, 4 environment-processing cameras, a Red-Green-Blue (RGB) camera and 1 depth camera (Figure 2.14). Other features incorporate 4 microphones, gaze tracking, gesture input, spatial sound and voice support. This hardware is used to produce high resolution 3D virtual objects, map and reconstruct its surroundings, plant the generated holograms accurately in the space and allow interaction with the user. Since HoloLens is a Windows 10 unit, it can support 32-bit-processor-compatible Windows Universal Platform apps, currently available at the Windows Store.



Windows 10
Custom-built Microsoft Holographic Processing Unit (HPU 1.0)
64GB Flash
2GB RAM (1GB CPU and 1GB HPU)
x86 architecture
IMU
HD 16:9 Light Engine
See-Through Holographic Lenses (waveguides)

Figure 2.14: `Microsoft Hololens`' Architecture [Kress and Cummings, 2017]

2.2 RELATED WORK

Given the defined processing capacity of the energy-efficient `Atom` processor, limitation to the amount of detail that can be visualized through AR devices, like `Hololens`, are set. The necessity for simplification of BIM models, which by definition are complex, is obvious. This chapter aims to illustrate studies have been conducted up to now, which are related to manipulation of BIM models and usage of AR technology to superimpose digital objects on the real scene. Consequently, it is structured as follows: Section 2.2.1 presents the developed approaches to isolate the exterior shell of a building or a construction. Section 2.2.2 discusses the visualization of digital objects through AR devices to provide a simple and intuitive perception to the user. Furthermore, it illustrates the related work considering the creation of a consistent and clarified user interface.

2.2.1 Simplification of BIM/IFC models

There are complicated and too "rich" datasets, as initially designed, which are relevant and necessary in their authentic form for certain applications, but convoluted and irrelevant to other uses. Considering the AEC industry, data transfer often requires simplification in combination with translation and/or interpretation when certain applications are involved.

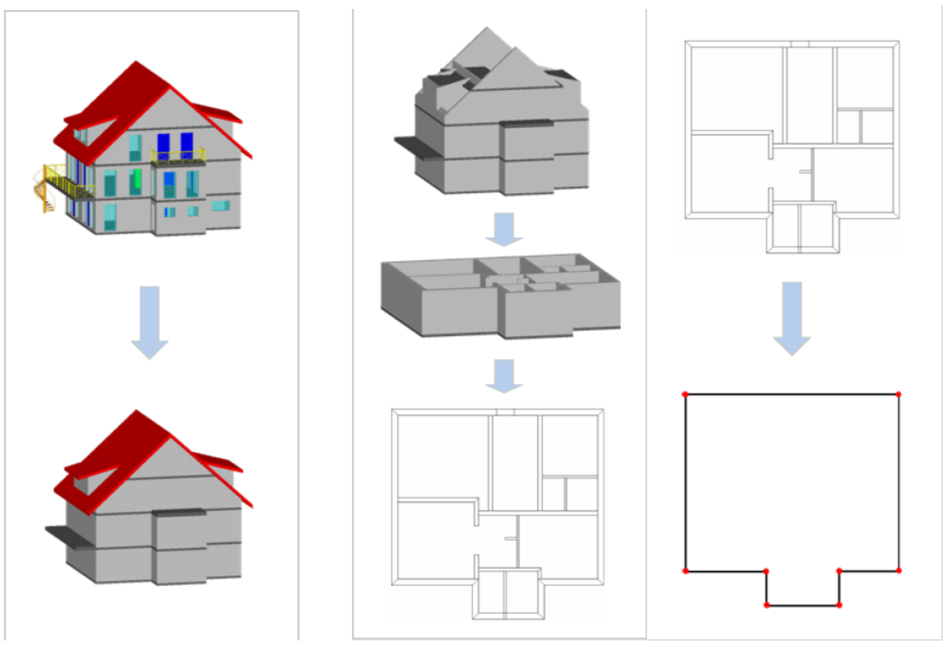


Figure 2.15: Exterior isolation [Nagel et al., 2007]. (a) Semantic level. (b) Footprint. (c) Dissolve

- Benner et al. [2005] aim to extract data from an IFC model and assign it to the corresponding classes of a `QUASY` object model. `QUASY` is a parametric 3D model which can store both semantic and geometric information providing flexibility with regard to the modelling phase. In order to achieve the isolation of the outer shell, the first step is the declaration of the semantic building structure and the matching of the IFC entities to the `QUASY`'s ones, in order to isolate the building elements of each storey. By vertically projecting the elements of each floor, one 2D geometric structure, called footprint, is generated for the elements representing the vertical objects, such as walls and columns

(`IfcWall`, `IfcWallStandardCase`, `IfcColumn`, etc.) and one for the horizontal elements, like floor slabs (`IfcSlab`). After having merged and dissolved the projection of the elements of each storey, only outer contours are retained to specify the exterior of the building. Geometric analysis is performed to determine the building elements touching the footprint. The surfaces having common points or edges with the footprint are stored keeping information about the building element to which they belong. Further geometric manipulation takes place to import the elements (`IfcWindow` / `IfcDoor`) that fill the `IfcOpeningElement` entities.

The suggested method produces promising results for simple IFC models in an acceptable time period, but it does not manage to create an accurate depiction of the exterior envelope for complicated models.

- Nagel et al. [2007] propose a method to derive a valid representation for the City Geography Markup Language (CityGML) Level of Detail (LoD)1 out of an IFC model by extracting a footprint for each floor and extruding it along the vertical axis. They simplify the IFC model on a semantic level retaining only elements which have a geometrical presence. Such horizontal element is `Ifcslab` and vertical elements like `IfcWall`, `IfcColumn`, `IfcBeam` and roof elements. Since, CityGML LoD1 is represented by exactly one prismatic extrusion solid there is no need to include geometries like windows or doors (Figure 2.15a). Each floor is isolated and treated separately. Next step is the production of an element-based contour polygon by depicting horizontal and vertical building elements onto the `x-y` surface (Figure 2.15b). They merge the element-based contour polygon using 2D Boolean operation and they dissolve it to obtain only the contour line of the exterior part of each floor (Figure 2.15c). Finally, the produced footprint is used in combination with a vector, pointing to the `z-axis` direction and having as value the height of the storey, to extrude the footprint along to the `z-axis` and produce the corresponding solid form of the floor.

  Since, this approach intends to generate CityGML LoD1 compliant files, which are not precise considering the creation of the outer shell, it works satisfactorily. However, these models can not be used as digital maquettes due to the limited amount of details that they include.

- To make a distinction between exterior and interior building parts El-Mekawy and Östman [2010], propose the generation of two footprints, one vertical and one horizontal. The vertical footprint derives from the projection of vertical elements, like `IfcWalls` and `IfcColumn`, while the horizontal one is formed by the IFC entities such as `IfcSlab` and `IfcBeam`. A geometrical investigation with regard to footprints that intersect with building elements of the model takes place to identify these elements. Only the objects which intersect with the footprints will be retained as exterior building elements. Elements which include an `IfcOpeningElement` are handled differently to contain this information after the geometric comparison of the footprint with the IFC entities.

  As with the methodologies described above, this methodology creates simplified models that can not be used to provide a clear perception of the exterior of a model which incorporates many and complex geometries.

- Diakité et al. [2014] suggest a new method to retrieve the inner and outer parts of a building model by utilizing its topological structure. A 3D building model can be perceived as a group of volumes like walls, columns or floors, connected to shape rooms. These objects and their relationships are described by 3D combinatorial maps. A combinatorial map is an edge-centered data structure consisted by sets of darts and the connection of them. A dart can be seen as a part of an oriented edge and a part of incident vertex, face and

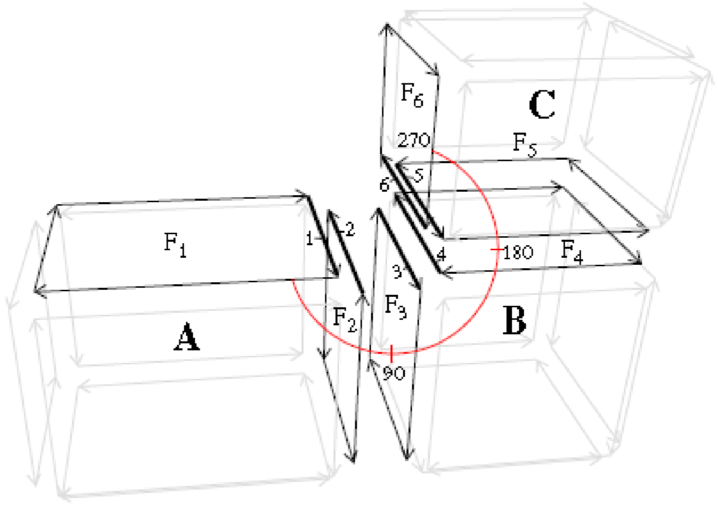


Figure 2.16: Representation of combinatorial map [Diakité et al., 2014]

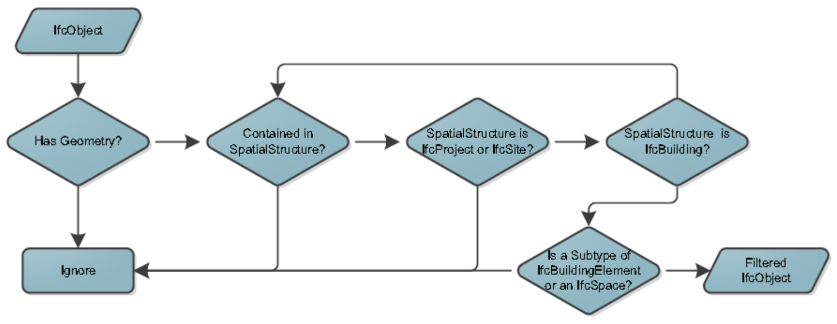
volume. By linking the darts someone is able to topologically reconstruct the faces and the volumes of the building (Figure 2.16). According to them, the outer part of a 3D model can be considered as a generalization in LoD3. The assumption of a closed building model, by means of windows and doors at every opening, is made. They duplicated all the faces of the reconstructed objects and checked the number of adjacent objects of each dart in order to obtain the indoor and outdoor volumes.

The first limitation of this approach is that it makes an assumption of a building model without any gaps or discontinuities. This is almost impossible for BIM models due to either design mistakes or round-off errors. Furthermore, the suggested methodology might trigger mistakes with objects of the exterior part that their geometry is complicated like railings. The creation of a combinatorial map for these objects could be a problematic process.

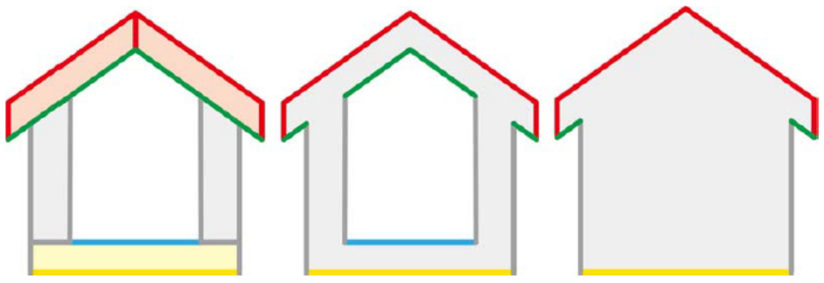
- In order to obtain the exterior envelope of a building, Donkers et al. [2016] perform a semantic filtering and a geometrical analysis. Since IFC schema contains approximately 900 classes, many of them representing movable objects or entities without geometry at all, a process to remove entities irrelevant to a building is essential. Every `IfcObject`, included in the IFC file, is certified whether it has a geometrical presence and if it is part of the building, using recursively the `IfcRelContainedInSpatialStructure` relation (Figure 2.17a).

  The extraction of the exterior envelope takes place using as input the output of the previous step. To achieve the extraction, they divide the space into partitions, which is the breaking down of the building objects to non-overlapping volumes. Then depending on the adjacency of solids' faces they execute a Boolean union process and the geometries contained in the exterior envelope are discarded. The latter happens either by retaining only the 2-manifold of the greatest volume or by performing a topological analysis of the building elements (Figure 2.17b).

  It is a reliable approach that perform well with topologically accurate models. However, many IFC models are not consistent with regard to the connection among the building elements. This could lead to the creation of an exterior envelope that would miss essential parts of it like railings which usually present topological inconsistencies.


IfcObject
Has Geometry?
Contained in SpatialStructure?
SpatialStructure is IfcProject or IfcSite?
SpatialStructure is IfcBuilding?
Ignore
Is a Subtype of IfcBuildingElement or an IfcSpace?
Filtered IfcObject

(a)



(b)

Figure 2.17: (a) Filtering of `IfcObjects` to isolate the ones with geometrical presence. (b) Isolation of the outer shell by merging the 3D building objects and retaining the 2-manifold of the greatest volume [Donkers et al., 2016].

2.2.2 Visualization through AR devices

Since, the combination of Augmented Reality (AR) with 3D building models is a rather new field of research and the visualization of BIM models, which are rather complex 3D building models, through AR devices is an intricate process, not many studies have been published discussing the issue.

- Kopsida and Brilakis [2016] use the `Kinect` camera and the corresponding `Kinect Fusion` algorithm to reconstruct the inner volumes of a building. New camera technologies, such as `Kinect`, can support both colour and depth images. These systems introduced new potentials for marker-free AR applications by providing a 3D understanding of the space and facilitating identification of the place and orientation of the user. After having loaded the BIM file of the construction, they manually displaced the model to correspond with the reconstructed model by the `Kinect` camera. Then, they connect the posture "calculated" by the `Kinect Fusion` algorithm with the BIM viewer in order the virtual projection to take place. The `Kinect` camera is tracked and chased by the BIM viewer's camera. To not lose essential detail during the tracking process, the motion of the `Kinect` sensors must be slow. Iterative Closest Point (`ICP`) algorithm is utilized to fix the adjustment and to reduce the discrepancies of the BIM model with the model produced by the scanning process.

  This approach demands the matching of the model, which has been produced by the scanning process, with the existent BIM model. Even that the `ICP` algorithm highly facilitates the matching process, the alignment of the two models can be a time consuming procedure that demands labour work for different building models. Furthermore, it does not provide the option of replacing the digital generated objects.

- Fonnet et al. [2017] aim to visualize parts of a historic/heritage BIM (`hbim`) to facilitate the building inspectors with the conservation of historical and cultural buildings. They perform a filtering approach to semantic level in order to retain only entities like walls, floors, ceilings etc. Inspectors have to scan the building using the spatial mapping functionality of `Hololens` to reconstruct the inner part of it and to approximately match the digital reconstruction with the `hBIM`, without the necessity of this matching to be extremely accurate. Using the `ICP` algorithm they perform the final rectifications, comparing the planes extracted from the mesh produced by the `HoloLens` with the simplified `hBIM` (Figure 2.18).

Their goal is to visualize 1:1 scale digital objects in the interior of a building. The replacement of these objects is out of the scope of their project. Furthermore, in order to place the computerized objects in the real environment, the interior of the building has to be scanned and aligned with the BIM model of the building.

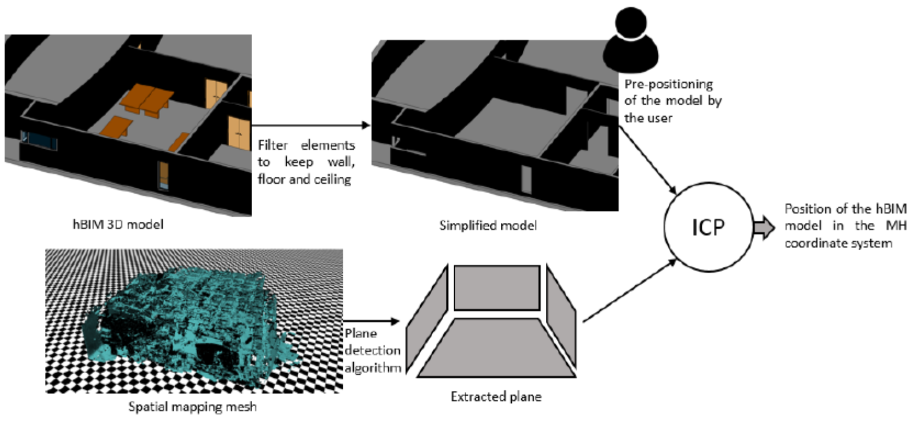

Filter elements to keep wall, floor and ceiling
hBIM 3D model
Simplified model
Pre-positioning of the model by the user
ICP
Position of the hBIM model in the MH coordinate system
Plane detection algorithm
Spatial mapping mesh
Extracted plane

Figure 2.18: Steps which are followed to align a BIM model with the derived model from a scanning process [Fonnet et al., 2017]

- Cardoso et al. [2017] suggest a method to project a BIM model into real world, in order to give to the users the perception of the future construction. To achieve this they manipulate the BIM model through `AUTODESK 3DS MAX`, which is a 3D modeling and rendering software engaging virtual and augmented reality functionalities. Then, the output of the previous step is passed as input to `Unity 3D` in order the configuration of the 3D scene to take place. Finally, the hologram is loaded to the `Microsoft HoloLens` to be projected into the real space (Figure 2.19).

  The suggested methodology produces an accurate depiction of an electrical substation. The use of `Unity 3D` game engine facilitates the development of holographic applications since it provides plenty of holographic tools and options. Furthermore, they incorporate some interaction capabilities for providing information about certain elements. However, the users do not have the option to replace the model, which would produce a more immersive experience.

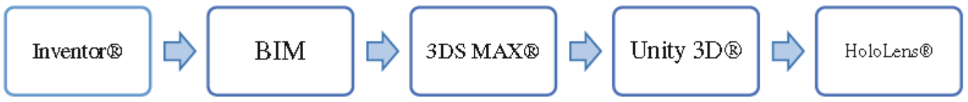


Figure 2.19: Methodology to visualize a BIM model through Hololens [Cardoso et al., 2017]

One of the most important aspects for AR applications is the creation of appropriate interaction means that will allow end users to communicate with the virtual content in an intuitive way. Considering AR interfaces Zhou et al. [2008] have identified 3 broad categories:

- Tangible user interfaces, a conception originally formed by Ishii and Ullmer [1997]. Real objects are used as a mean in order the users to have the ability to manipulate digital generated information. The main power of this idea stems from the fact that physical objects have characteristics with which the users are used to, like tangible shapes and intuitive properties, making their use easy and simple. Usually physical elements are associated with gestures, gazes and sound interactions which guides to a multiple way of interaction with the users.

- Collaborative user interface. AR can assist both distant and co-located tasks. For co-located cooperation, Augmented Reality (AR) can be adopted to boost a common real work environment and provide an interface for three dimensional Computer Supported Collaborative Work (`CSCW`). After Billinghurst and Kato [2002] having conducted tests with the `Shared Space` application,it was identified that this kind of interface is straightforward and simple for extensive cooperation. In contrast to other interfaces, collaborative user interface remains uncomplicated and depends on social protocols activities.

- Hybrid AR interface. It naturally incorporates a dynamic number of input and output units and the interconnection of them. The provided options will remain simple and clear to provide the users the opportunity to expand their effectiveness even more. `Microsoft HoloLens` is a HMD which does not demand physical touch by the user's hands to interact with the hologram that it is projected. It means that the interaction will be completely different with the one that the users are used to (keyboards, mice and touchpad). In contrast to the classical means of interaction, communication with the projected holograms takes place using the gaze, gestures and voice while reaction comes in the form of sound or graphics. Since, the great majority of the users are not familiar with this way of interaction, although more instinctive, at first it will be confusing and disturbing for them. So, it is of a high importance to supply detailed instructions on how to use the developed applications.

3 METHODOLOGY

This chapter aims to illustrate the sequence of steps (Figure 3.1) followed to achieve the simplification of a BIM model and its projection through an AR device. Section 3.1 discusses the conversion of a proprietary file format to an open standard, Section 3.2 describes how each storey of a BIM model is isolated, the process of creation of a different file for each building element is mentioned in Section 3.3, a detailed description of exterior extraction takes place in Section 3.4, the configuration of the holographic scene is illustrated in Section 3.5, the steps which are followed for the holograms' manipulation are presented in Section 3.6 and finally Section 3.7 refers to the way that the application is deployed to the AR device.

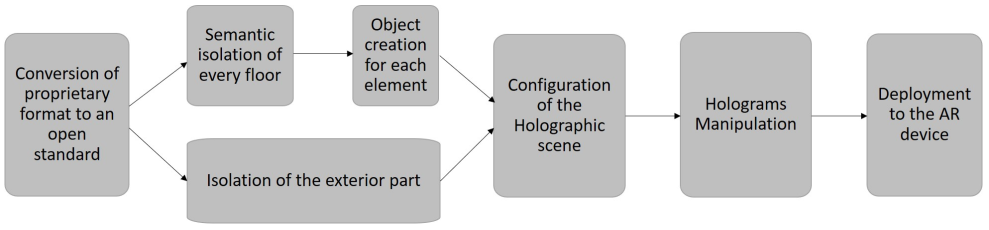

Conversion of proprietary format to an open standard
Semantic isolation of every floor
Object creation for each element
Isolation of the exterior part
Configuration of the Holographic scene
Holograms Manipulation
Deployment to the AR device

Figure 3.1: The followed methodology to simplify a BIM model and project it through an AR device

3.1 CONVERSION OF THE PROPRIETARY FILE FORMAT TO AN OPEN STANDARD

An open standard is a standard that can be used by everyone free of charge, there is no necessity of proprietary software to open or edit it and there is not a patent that restricts its usage. The decision to use an open standard derives from the fact that there is no chance for the information to get locked in by a specific enterprise or institute. Furthermore, open standards highly support interoperability, communication among different parties and improve information transfer & usage. Particularly for this thesis project, the schema of the standard that is to be adopted is available online. In the same way, there are freely available sample datasets on the web which can be used to test the functionality and identify the imperfections of the proposed methodology. Finally, this adoption complies with the academic scope of this project. The developed approach could be tested and assessed by the academic community.

3.2 SEMANTIC ISOLATION OF THE STOREYS

Each storey of a building could be extremely different than the others or have small dissimilarities. Therefore, each of them should be isolated and projected separately to give a clear perception of the interior of the building.

To achieve the extraction of each floor, an instance of the building should be obtained at first. Out of this model, a list of names of all available storeys in the building, as these specified by the BIM model's designers, is created. BIM models contain some entities, such as the site, the project and the building, that are connected with every other entity that is included in the model. Instances of these entities are isolated since they have to be written to every new IFC file corresponding to each floor. The next step is the generation of a list of spatial structures which contain elements of the same level of spatial hierarchy. Each element of this list includes a "tuple" of entities' ids that are contained in the corresponding storey. An iteration on the list of storeys takes place in order to create a different file for each storey. The first lines of every file are composed of the schema and some metadata information concerning the edition and the ownership of the file, having as their source the original file. The project, site and building instances obtained earlier, are used to populate the files with entities and elements which are connected with them.

All the entities that contained in the BIM model are checked in order to specify to which floor do they belong. A function is created to receive as input parameter each entity of the spatial structures. These spatial structures contain the ids of the elements that compose a storey. The model is recursively traversed in order to identify the elements that are connected with these ids. A list, accommodating these elements, is introduced that gradually gets empty to finalize the identification process. In addition to that, a set is created enclosing all the elements that have been identified connecting with a floor in order to avoid the writing of entities that are used by more than one elements. An extensive modification is performed on the names of the entities which are written in every file, in order to be feasible for the BIM viewers to project the files. Since some of the elements are structures which are connected with more than one other elements, a thorough check is performed to identify and extract them,in order to avoid the loss of any essential information.

3.3 CREATION OF A DIFFERENT FILE FOR EACH BUILDING OBJECT

To visualize metadata of every object in the BIM model, there should be a clear distinction of the objects that compose each storey. Every object will be hosted by a different file having as its name the metadata information that is considered relevant to be projected.

Every storey is hosted by a different `.ifc` file. To achieve an automated extraction of every object included in every of these files, the directory, which contains the `.ifc` files, is recursively traversed to identify them. The loop generates a new folder for every IFC file that it recognizes. The new folders have the names of the storeys from which they derive from and their purpose is to host every file that corresponds to a different building object.

The next step is the filtering of the `.ifc` files in order to clear them from the entities without geometry or entities that represent movable objects. Each and every entity that has geometry is checked individually and a file, compliant with the IFC specification, is created for them. The initialization of a set, which hosts the ids of the objects that have already been manipulated, assures the avoidance of duplicates during the file creation process. Furthermore, a check is performed to identify objects that have the same name but different geometries and objects that might contain collections of other objects.

Since, the visualization of holograms through an AR device takes place only using certain file formats and `.ifc` is not one of them, a conversion to a file format, which is compliant with the ones that the AR device support, is essential. A temporary file is created to host all the entities that take part in the construction of each building object. In order this to happen a recursion is performed to isolate the entities that compose this object. These entities are written in the temporary file combined with the lines describing the edition and the ownership of the file in order to be `.ifc` compliant.

After having produced the temporary file, the conversion software is called to produce a file format that is acceptable by the AR device. After its execution, the temporary file is closed and deleted. The newly created file is stored in the folder of the corresponding floor. The recently generated file has been stored having as its name the metadata information that considered relevant to be visualized to the user. For this particular project the type, the id and the name of each element is the most useful information contained in the BIM model. By storing every object of the BIM file in the same folder, there is the opportunity to load them all together in the holographic scene without the need of further distinction among them. Furthermore, the creation of a different file for every building object takes place after having performed the isolation of every floor. The acquisition of a different `.ifc` model for every storey provides a clear picture of the elements contained in them and assures that there will not be the necessity to re-perform the execution of the converter software in case that it will not run properly.

3.4 EXTERIOR EXTRACTION

One of the main goals of this project is the isolation of the exterior part of a building. Having isolated the outer shell of a construction and visualizing it through an AR device, it is feasible for someone to have a clear perception of the future building before its creation. The visualization of the exterior envelope could serve as a digital maquette and an AR device might host more than one version of the future building in order the decision makers to decide which one fits the most with their desires.

During the last years many researchers tried to isolate the outer shell of a BIM model to adjust it to other models like the CityGML. The majority of them, as described in the Section 2.1.1 section, produce a vertical and a horizontal projection of the objects of each floor and dissolve the inner geometries of the generated footprint. After that they extrude the dissolved line to a certain extent and "stick" the extruded parts to reconstruct the building. Its an efficient but simplified method that does not give an accurate indication of the details of the building.

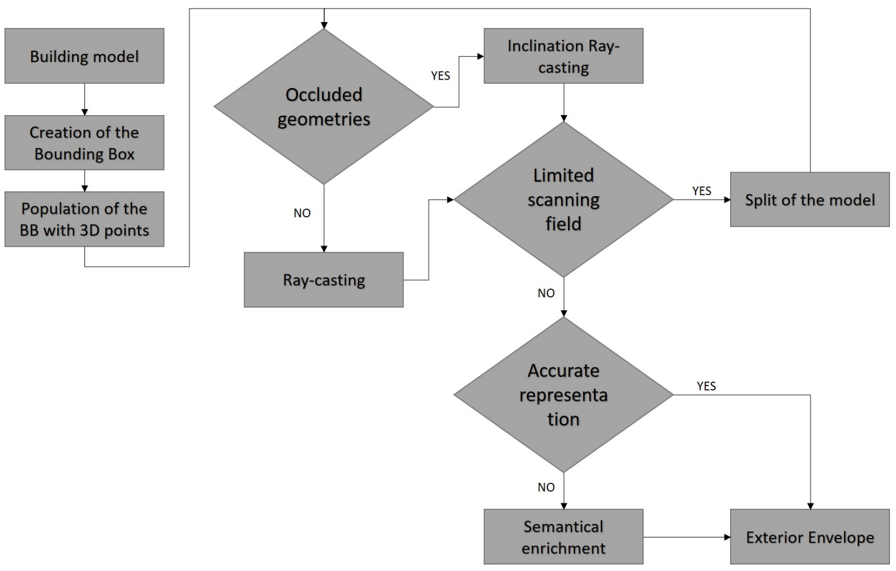


Figure 3.2: The followed steps to obtain the exterior envelope of the BIM model

In order to obtain a precise representation of the outer shell, we have developed a ray-casting process. Ray-casting is a technique, widely used in the computer graphics field, according to which rays are sent in order to identify the first surfaces that intersect with them. The points from where the rays are sent, could simulate points from where a person stares at an object. Just as the human's gaze stops at the first object that it will meet, so too the ray stops at the first surface with which it will intersect. The removal of the non-observable surfaces of any model highly simplifies the model and makes its rendering through software and devices either feasible or more efficient.

The sequence of steps followed for the extraction of the exterior envelope is described as below: the creation of a bounding box is described in Section 3.4.1, Section 3.4.2 explains the ray-casting process, Section 3.4.3 analyzes how and why the model is split and Section 3.4.4 describes the enrichment of the outcome of the ray-casting process with semantic isolated objects. Figure 3.2 shows the sequence of the followed steps in order to obtain the most accurate representation of the exterior part of the building model.

3.4.1 Bounding Box

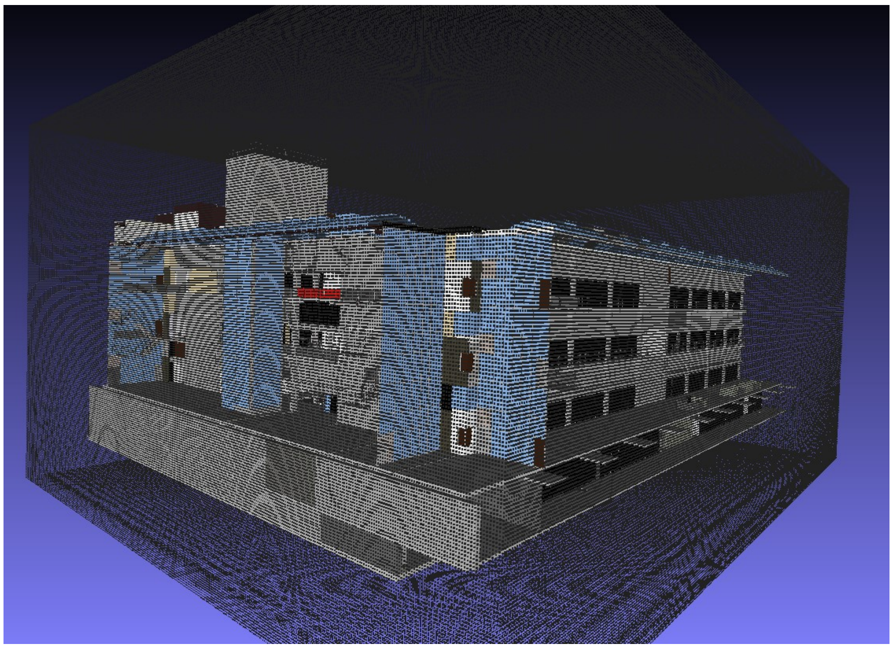
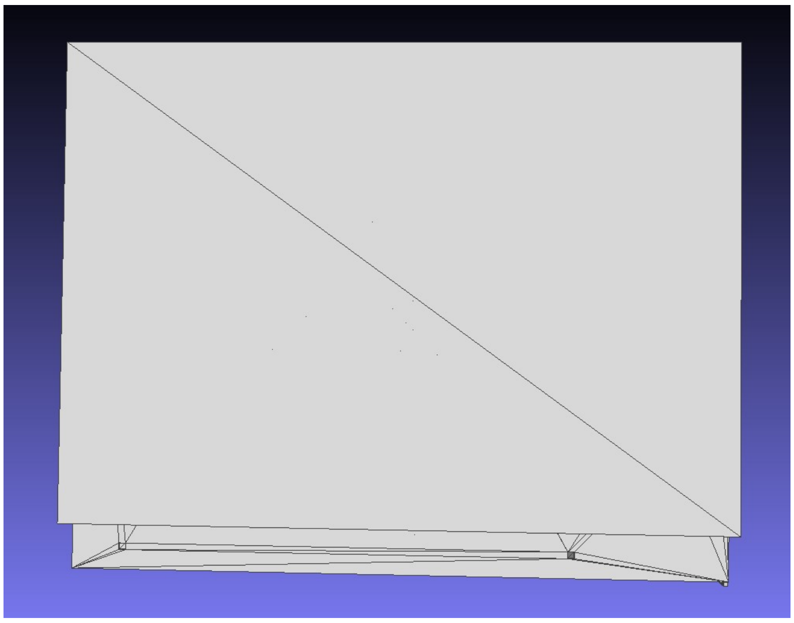


Figure 3.3: Creation of a bounding box and population of it with 3D points

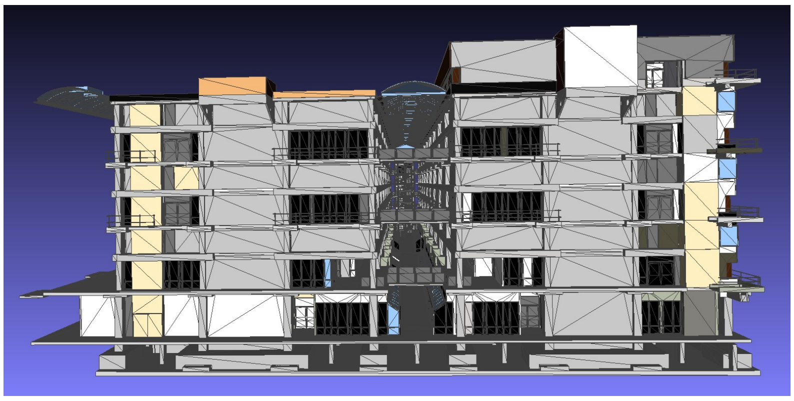
The ray-casting approach demands starting points, or points from where the rays would be sent. These starting points serve as possible view-points from where a spectator could look at the building model. In order to extract as many triangles as possible, the distribution of the points should present a symmetry to guarantee the efficiency of the casting. It would be pointless to create a number of points in random positions that would not cover with rays all the parts of the building. So, a bounding box is created around the building and its sides are populated with points having the same distance among them. Depending on the number of points, every part of each side will be bombed with rays equally, without leaving any areas without being scanned by the rays.

In order to create the bounding box around the building, firstly the BIM model must be reconstructed in terms of being identifiable by ray-casting algorithm. Every vertex, face and entity connected with them is read and stored in a way that is acceptable for further manipulation by the script. Having imported the coordinates of every vertex contained in the model, a geometrical comparison among all of them is performed to identify the minimum and the maximum value of every coordinate among all vertices. The maximum and the minimum values define the extent of the building. Having obtained them, it is possible to define the dimensions of the bounding box by adding a certain margin, which is considered reasonable, to the values derived from the building. This happened in order to avoid the collision of the building objects, having extreme coordinate values, with the generated points and to make the ray-casting process functional.

In addition to that, the maximum and the minimum values will be used to define a step according which the density of the points will be specified. To populate each face of the bounding box with points, one of the extreme values of one axis is kept steady while the maximum and the minimum values of the other axis are used to define a step according to which the population with points takes place. The step is defined by finding the distance between the extreme values for each axis and dividing it with a number which specifies the number of point. This number might be defined after experimentation and observation of the number of points that each step produces. This process is repeated 6 times. One time for each face of the box. The bounding box looks like the Figure 3.3



(a)



(b)

Figure 3.4: Sides of different complexity of the same building (a) Bottom simple side. (b) Random complex side

Nor all the sides of the building model are of the same dimensions neither of the same complexity (Figure 3.4). To optimize the use of the number of points, the distribution of them took place in a way that the faces of the bounding box looking at the complex sides hosted more points than the others. By doing this, it is possible to achieve better accuracy considering the extraction of triangles. The available computational power will be dedicated to the most complex sides to achieve better visual outcome. The determination of the complex sides was achieved by visualizing the model through software that has the ability to show the triangles forming each face of a model. The unequal distribution was achieved by specifying different steps during the population of the sides of the bounding box with 3D points.

3.4.2 Ray-casting

The 3D points, which were created on the top of the bounding box, will be used as starting and end points for the rays that will pass through the BIM model. A ray will start from each point of each face of the bounding box having as its direction the corresponding point of the opposite face. While it will be penetrating the building, it will intersect with plenty of faces of the building. Some of these faces or triangles (the depiction of the construction takes place using triangles) belong to the exterior shell of the building while others are part of the interior. All the triangles of the model are checked whether they intersect with the ray or not. The computational demand of this process exponentially grows in accordance with the complexity of the model or in other words with the number of triangles that the building contains. To make the distinction among them, the references of all the ids of the triangles that intersect with the ray are stored in a list.

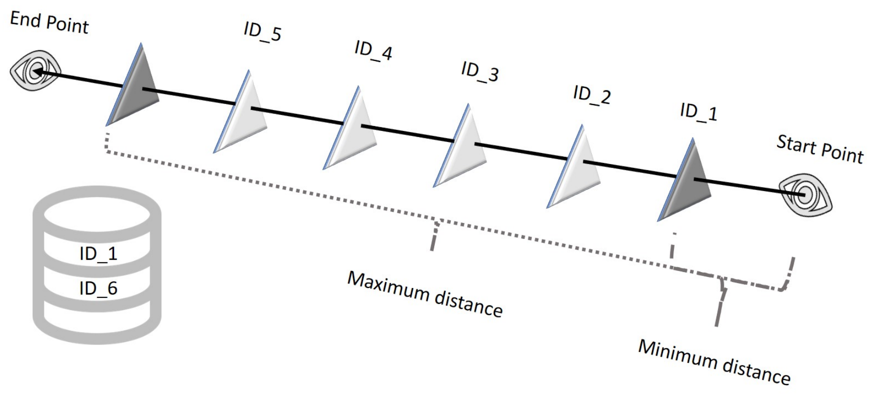


Figure 3.5: Ray-casting process

This list is traversed and a constant calculation of the distance of each of the triangles with the start point of the ray is performed. The minimum and the maximum distances are stored alongside with the ids of the corresponding triangles. The ids of the triangles of the maximum and the minimum distance are stored in order later on to reconstruct the envelope of the building (Figure 3.5). For computational power issues, rays are not sent from each face of the bounding box but only from the "top", "front" and "right" face. Knowing the minimum and the maximum distance we can extract the first and the last triangle that the ray meets. The maximum distance of a triangle from the start point is equal with the minimum distance from the end point. Sending a ray from the corresponding point of the opposite side would return exactly the same values upside down.

The number of needed points was specified experimentally. Initially it was calculated how much time was needed to perform the intersection process for just one ray. After that defined what would be a reasonable time to run the algorithm. Having these as inputs, it was easy to specify the number of points needed to extract the exterior shell.

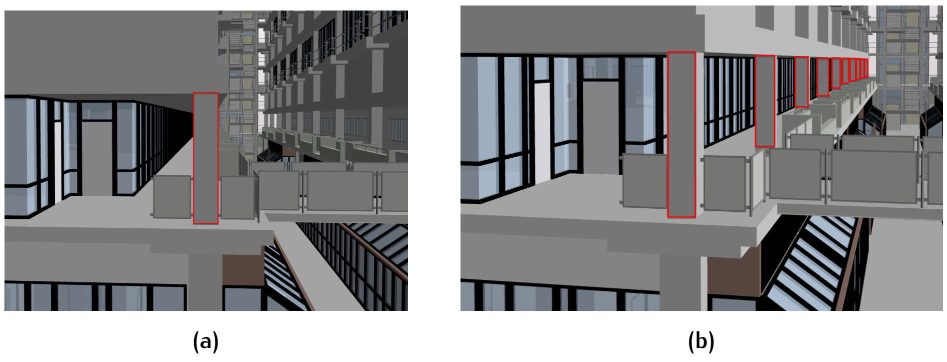


Figure 3.6: The same part of the building from different points of view (a) Obstructing element. (b) Obstructed elements.

Despite the number of 3D points that populate each side of the bounding box, there are objects of the building that belong to the exterior part but the vertical rays can not extract them. This happens because some parts of the construction are occluded by others, not allowing the algorithm to perform efficiently (Figure 3.6).

To avoid this shortcoming 9 rays, instead of one, are emitted having the same starting point but different end points. Except from the one which goes vertically to the corresponding opposite point, the others move diagonally. To achieve the inclination, the rays are sent to the points that surround the opposite 3D point (Figure 3.7). This inclination assures that the rays will intersect with triangles of the building which are part of the exterior shell but it was impossible to reach them with the original approach. These faces are either obstructed by other objects or included in recesses of the buildings or they are oblique sides of building objects. Each starting point is checked if it lies on one of the edges of the bounding box in order to adjust the number of rays that are sent from it, since there are not 8 points that surround every opposite point.

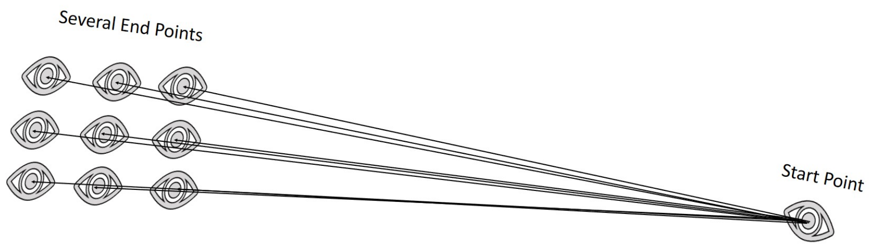


Figure 3.7: Inclination ray-casting process

For each ray that is sent and passes through the building, a distance check is performed between the start point of the ray and all the triangles that it meets. Its common for rays starting from the same point to have the same triangle as the one with the minimum distance. The id of this, is stored only once to avoid duplicates. The collection, containing the ids of the identified triangles, will be used to produce the envelope of the building.

3.4.3 Split of the model

There are buildings which are so complex and convoluted, that the (inclination) ray-casting algorithm demands a large number of 3D points in order to produce an accurate representation of the exterior envelope. To reduce the necessity of high computational power and to obtain a decent depiction of such complex buildings, a split of the model can take place. The split of the model is not a generic part of this methodology, but its adoption depends on the complexity of each building. In any case, the split will separate the model into two or even more parts. By doing so, it is possible to diminish both the number of needed points for the ray-casting process and subsequently the required time for the execution of the algorithm.

The first step is to identify the axis according to which the model will be split. This axis passes through the most complicated part of the building. An object that lies on this part has to be recognized in order to isolate its coordinates. They will be used as reference points for the creation of the splitting axis. Then, the model is deconstructed to its faces and vertices. An iterative process is initiated to check the coordinates of every vertex contained in the building model. Within the framework of this project and specifically due to the complexity of the sample dataset, we decided to split the model into two parts. So, three lists were created to host the objects according to their position in the model. If the whole geometric presence of an object does not intersect with the created axis, it is assigned to one of the two lists which host objects that do not form the complicated part. Otherwise, the object is assigned to the list that host the objects that will be part of both models.

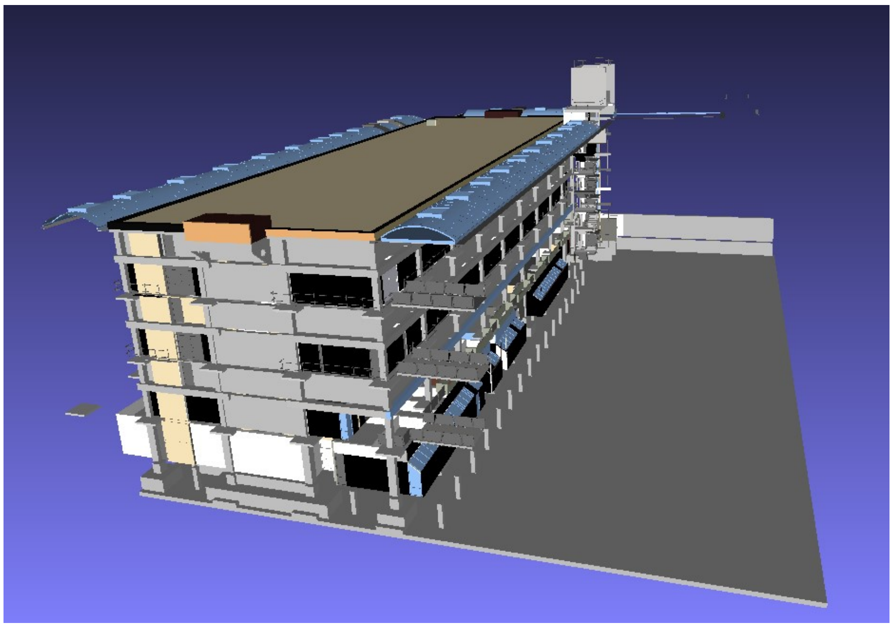


Figure 3.8: Outcome of the splitting process

Two building models were created in order to host the objects of each list. The objects which lied on the complicated part are written in both models. These new models can be handled separately by the ray-casting algorithm in order to make the process more efficient. The outcomes of the algorithm, one for each model, were merged to give the final envelope of the building. During the merge procedure, a check takes place for duplicates since some of the building's objects may be included in both models. The original building could be split into even more than two parts. That depends on the complex sections of the model and the computational power that someone possesses.

3.4.4 Enrichment with semantically isolated objects

The facade of a building presents great complexity in comparison with the rest of it. The extraction of a complex facade, having defined a large number of points, is a rather "expensive" process. Since there were constraints regrading the computational power, the outcome of the algorithm was enriched with objects that the majority of them belong to the exterior part. To achieve that a semantic isolation of these objects was performed according to their type.

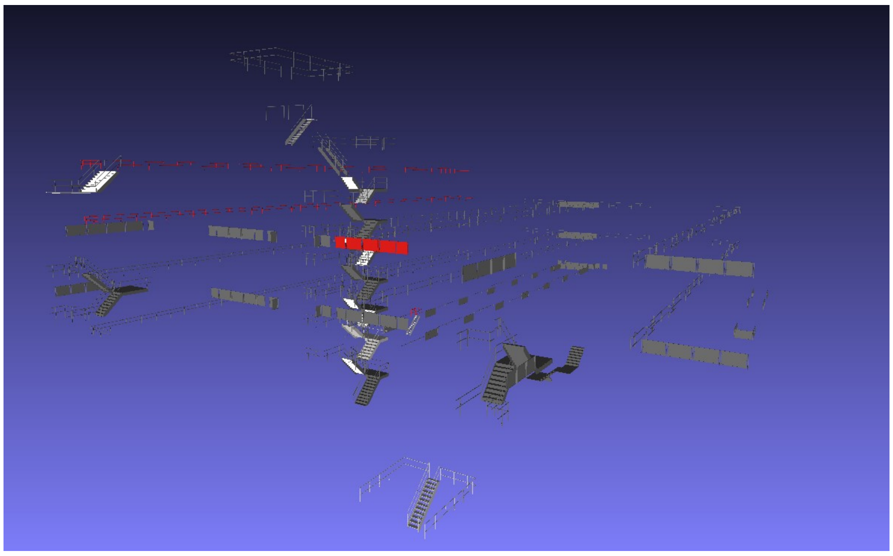


Figure 3.9: Semantically isolated objects that will enrich the outcome of the ray-casting process.

Initially, the conversion of the `.ifc` file format should be adjusted in a way that metadata information about objects' types will be retained. Subsequently, the building objects that are mainly part only of the exterior envelope, such as railings and skylights, have to be identified and isolated semantically. After having decided which elements will enrich the product of the ray-casting algorithm, a list of them is created and traversed recursively in order to create a new file out of them (Figure 3.9). Objects of the outer shell having complex geometries, are excluded in order not to create a model that the AR device will not be able to visualize.

The semantically isolated objects will be combined with the outcome of the ray-casting process to produce the final model. Many of the triangles that the scanning process extracted will be part of the semantically isolated objects as well. The removal of the duplicates is essential to produce an acceptable version of the envelope in terms of storage size.

3.5 HOLOGRAPHIC SCENE CONFIGURATION

The hologram which will augment the reality is digitally designed in an artificial environment called scene. The developed scene accommodates all the assets, obstacles and decorations that are used to form the general frame of the produced application. It is the level in which all functionalities and projections settings of the project are defined.

Firstly, the camera and the lights are modified in relevance with the user. Lights define the color and mood of the 3D environment. The color, mode, intensity, shadow type and resolution of the lights are adjusted according to the suggested values by the manufacturer of the AR device. A camera is defined as the point through which the user views the world. For the configuration of the camera a simulation environment is used through which it is possible to place the camera in a location that will facilitates the finding of the hologram. The placement of the camera is of high importance since misplacement can lead to a negative experience or even to inability to distinguish the hologram.

The holograms are placed in a distance that the user has the ability to interact with it. In addition to that, they are rotated in relevance with the users, in order

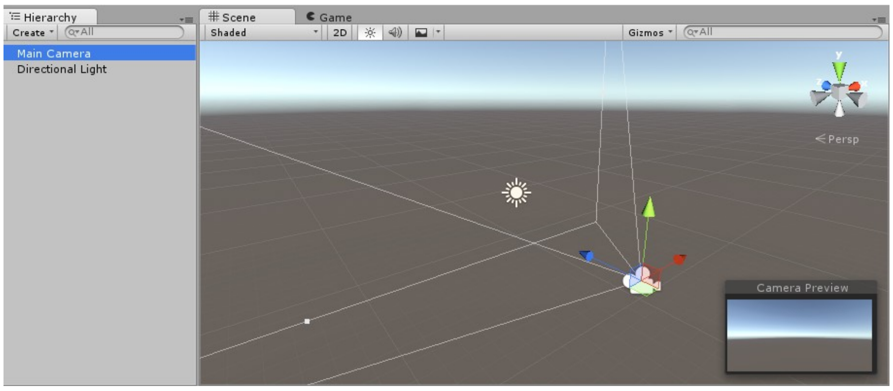


Figure 3.10: Placement of the camera and lights assets to configure the holographic scene.

to be convenient for them to perceive and interact with the model. The suggested distance to place a hologram of a moderate size, is 2 meters away from the user's gaze.[1]

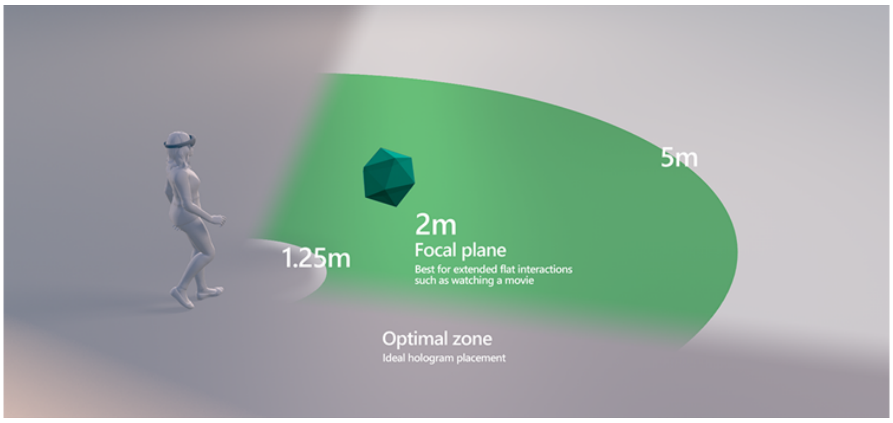

5m
2m
Focal plane
Best for extended flat interactions such as watching a movie
1.25m
Optimal zone
Ideal hologram placement

Figure 3.11: Optimal distance for placing holograms from the user [1]

3.6 HOLOGRAMS MANIPULATION

A properly configured scene will host the holograms that will be projected to the final user having as purpose to give an intuitive perception of the building before its creation. Except of the exterior envelope, each storey will be visualized to provide a clear depiction of the interior of the construction. The visualization of them should be efficient and robust which means that it is essential to produce holograms proper considering their complexity to be deployed in the device. The visualization of holograms that flying in the room would be pointless and annoying for the user. While the user is staring at a model he/she should have the sense of a real object and not of something that interrupts his/her field of view. Furthermore, it is crucial to implement some interaction with the user in order to engage him with the process of visualization.

The users are granted with the opportunity to project some metadata of the entity of their preference, to position the models to whatever place whey want, to grow and shrink the model in order the model to fit on every horizontal plane and to rotate the model. In order not to hinder the performance of the device and to facilitate the users with the use of the application a menu was designed and implemented to have the capability to visualize and hide models according to their

1 https://docs.microsoft.com/en-us/windows/mixed-reality/hologram-stability

willing. The sequence of steps followed in order to provide the holograms with interaction capabilities are illustrated in Figure 3.12

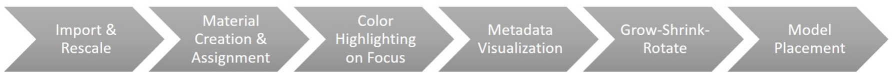


Figure 3.12: The sequence of steps followed in order to provide the holograms with interaction capabilities

3.6.1 Import & Rescale of the model

The process of importing a model in the holographic scene is not that complicated. Special attention has been paid on the classification of the object during it. Since many storeys of the same building and the exterior envelope of it will be imported to the scene, the management of all these assets was be an intricate action. To avoid that a hierarchical structure defined storing the objects of each spatial structure in another system of folders and sub-folders to facilitate the assignment of the scripts and materials later during the project. Additionally, separate structures are defined to host scripts, materials, textures and other assets that could potentially be added to every model.

The importing process introduced the objects in dimensions that is impossible to accurately be projected by the AR device. To diminish the size of the objects, in order to be feasible by the application to visualize the objects, a rescaling operation implemented. A loop created to take an instance of each object in the directories created to host building objects. Dimensions of them read and a rescaling factor multiplied with every instance. The outcome replaced the original files to minimize the storage necessity. Finally, the newly, shrunk developed objects imported in the holographic scene.

3.6.2 Material Creation & Assignment

Materials, shaders and textures that will be used during the rendering process will highly affect the performance of the AR device. Each material is correlated with a shader and a texture which, by applying mathematical functions, calculate a color for each projected pixel based on parameters such as the lights and shadows.

The materials which are used during this project are the ones that support the best holographic applications, since they load the model with the minimum extra amount of information and the holograms can easily be adjusted to the environment's conditions. Dynamic lighting, which incorporates reflections, hover lighting & image based lighting, and dynamic shadowing is not supported in order not to hinder the performance of the holographic application. In the same way, it is assigned 0% metallic texture and the surface defined to be 50% smooth to achieve natural perception of the material.

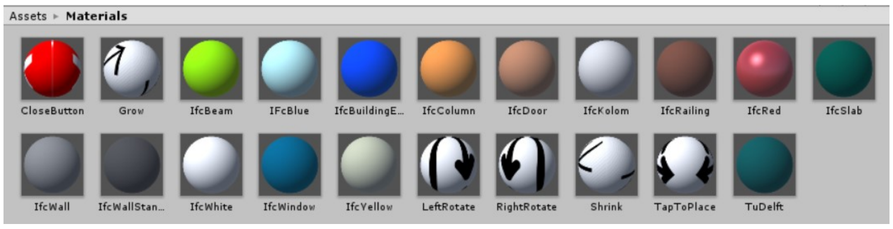


Figure 3.13: Multiple materials creation in order to be assigned to every group of objects that are hosted in the building model

Plenty of different materials are created to support the different entities that are hosted in the building model (Figure 3.13). The process of their creation is the same for each and every of them. A standard shader is assigned, by default, to every material and is replaced by the one which best supports holographic applications. Several functionalities of the this shader are disabled like reflections, rim lights, hover light, round corners environment coloring to make the execution of the application smoother. For some of the materials special textures are used in order to be assigned to the buttons created for the holograms interaction. These textures are designed to give to the user an intuitive perception of the functionality of the button and they are loaded to the material as bitmap image. The system of folders and sub-folders, described above, is used to assign massively the material which corresponds to each building object. An iterative process constructed to identify the type of each object in order to be matched with the corresponding material.

3.6.3 Color Highlighting on Focus

The interaction of the users with the AR device is performed by allowing them to point at holograms and afterwards operating some input action. Gaze is the mean which indicates both the application and the users for the place at which they stare at each moment. It is essential for the intuitive purpose of the application to allow the users to recognize where they stare at and how this is perceived by the application.

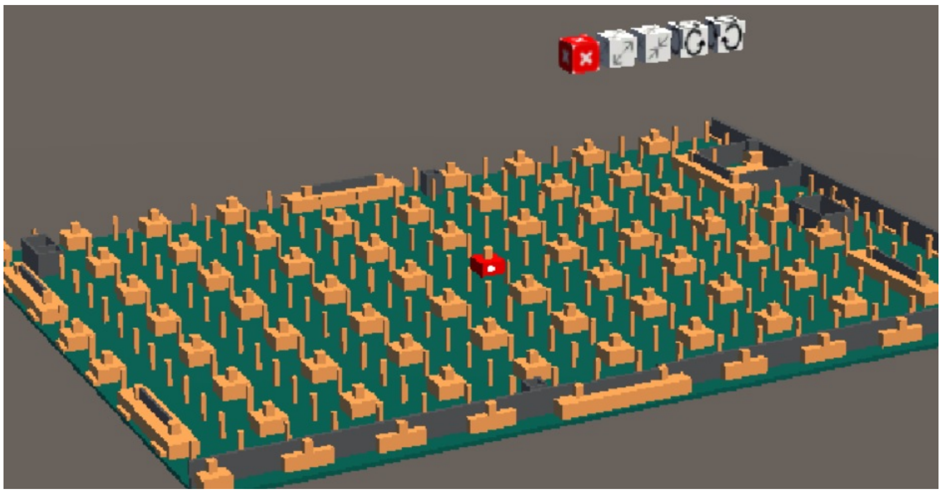


Figure 3.14: Highlighting the observed object by changing its color

To visualize the gaze of the users a cursor is initiated to follow the their eyes. This cursor is modified to support holographic applications by minimizing the highlights, special effects and the graphics power that is demanded for its projection. It seems like a white disk which when it lies on a surface of a building object, it changes the object's color from whatever color to red (Figure 3.14). By doing this

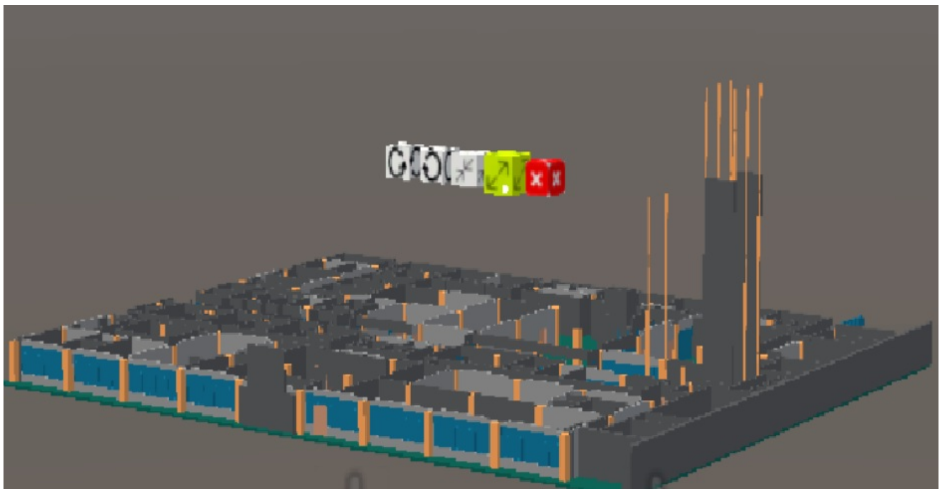


Figure 3.15: Highlighting the observed button by changing its color and its size

it is obvious to the users at which object they are looking at any given moment. Furthermore, as soon as the AR device recognizes the hand of users it transforms the withe disk into a white torus to notify them that they can interact with the hologram using gestures. If none of the objects of a storey has an intense red color means that either the users do not stare at one of them or they are too far away from the hologram in order it to identify its gaze.

The indication that a user stares at any of the created buttons takes place using another visualization way. Instead of turning the button's color into red, its color becomes highlighted yellow and it gets bigger (Figure 3.15). It is even more clear to the users that they stare at a button and this button is ready provide interaction with the hologram. As soon as the collider of an object identifies the cursor on one of them, sends a reference to the object's material and mesh constructor. The reference which is sent to material is used for the change of the color while the reference to the mesh constructor is used for the resize process.

3.6.4 Metadata Visualization

For this project it is regarded relevant to visualize information which is related with every object contained in the building model. The users must have the opportunity, by using gestures, to visualize information about a building element that they stare at the given moment. The first step of this procedure is the isolation of each building object and the extraction of the information which is acknowledged important. After that a file, of a format compatible with the AR device, is created for each object and their names include the metadata information extracted earlier, described in the Section 3.3.

After having imported each object with the modified name in the corresponding directory, as described in the Section 3.6.1, the objects are imported in the holographic scene in order their geometries to be reconstructed. The configuration of the scene allows the identification of gestures that intend to visualize information about the stared object.

A script, that recognizes the gaze of the user and the making of a gesture, is integrated with every object of the building. As soon as the script identifies them, it connects a newly created instance of the corresponding object with an empty text object that lies on the scene. The name and the position of the object are retrieved in order to be applied to the text object. When a user performs the tap gesture, the script modifies the position and the content of the text object in order the application to visualize a yellow text above the model showing the type, the id and the name of the corresponding building object (Figure 3.16).

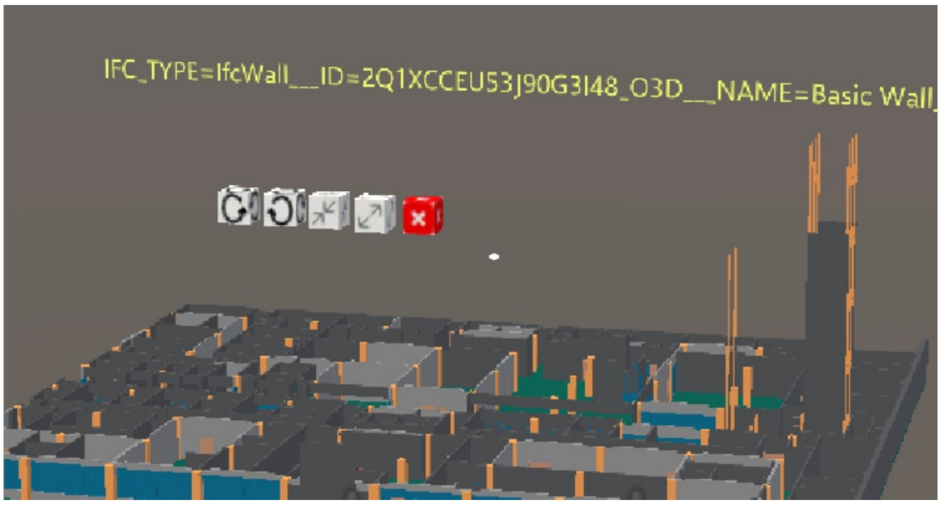
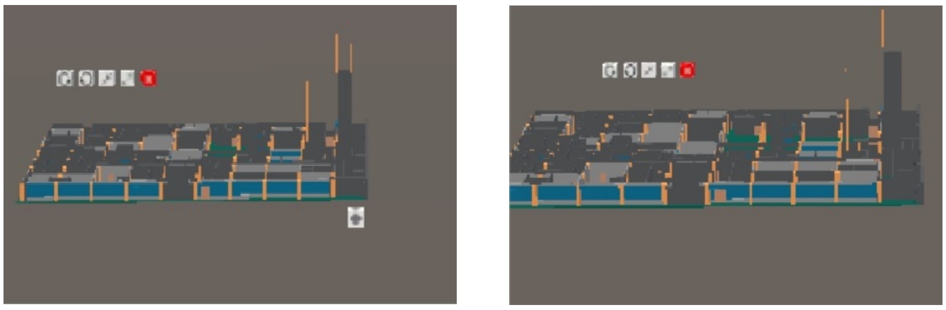

IFC_TYPE=IfcWall__ID=2Q1XCCEU53J90G3I48_O3D__NAME=Basic Wall

Figure 3.16: Metadata visualization above the model

3.6.5 Scale

Except for the metadata visualization it is considered useful for the users to have the opportunity to grow or shrink a model according to their willing. Having the ability to rescale a model, the users observe better each floor without the necessity to bend over the model which might trigger performance issues. Furthermore, the rescaling option offers them the opportunity to fit the model on any horizontal face that might exist in a room which provides a more immersive experience to the users. Resizing, in accordance with the real objects that decorate a place, gives to the users a greater sense of realism and enable greater involvement with the application.



(a) (b)

Figure 3.17: Grow functionality (a) Pre-grow size of a storey (b) After having tapped the grow button

Both the grow (Figure 3.17) and shrink (Figure 3.18) functionalities combined with cubic buttons which are placed above the model in order them to be visible by the users.To achieve the massive grow or shrink of a specified storey, all the building objects, that belong to it, are assigned to a greater entity which does not have geometry but it accommodates the geometries of them . Each time that a user taps on one of the buttons an instance of this greater entity is obtained. It is used to modify the dimensions of the storey by multiplying the entity with a constant value in order to grow or shrink the model.

3.6.6 Rotate

The developed application provides the ability to rotate (Figure 3.19) the visualized models. Instead of the users to move around the model, which sometimes can be

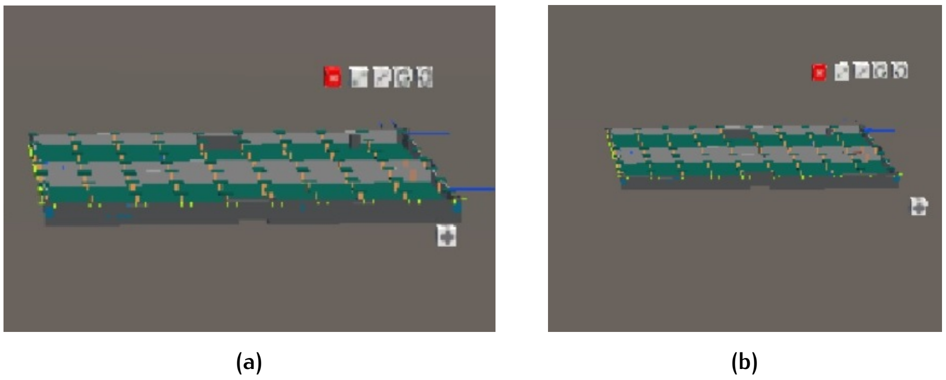


Figure 3.18: Shrink functionality (a) Pre-shrink size of a storey (b) After having tapped the shrink button

proved difficult due to space limitations, they have the option to rotate the model. The rotation functionality allows the users to observe parts of the model in greater detail giving a clearer perception of the building.

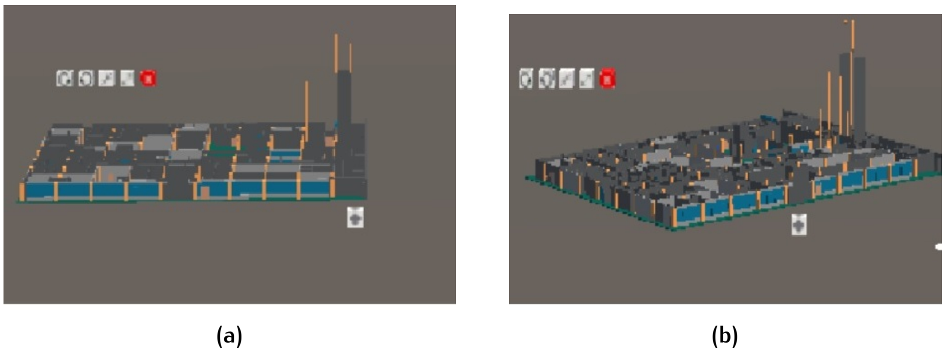


Figure 3.19: Rotate functionality (a) Before applying rotate (b) After having tapped the rotate button

The greater entities that host every object of each storeys, as described in Section 3.6.5, are used for the execution of the rotation. As soon as a user taps at one of the two buttons, that rotate the model either left or right, an instance of the hosting entity is created and passed for modification. This instance is used to alter the class which describes the position of the model by multiplying its original position with a fixed value indicating degrees of transformation. After having tapped at one of the buttons, the model retains the modified position even if the user closes and reopens the model. The application assumes that this position is the most convenient for the user.

3.6.7 Model Placement

One of the most crucial features of a holographic application is the positioning of the holograms in a way that seem real. The blending of the physical environment with the virtually designed objects is a part that can highly affect the success of the application and enhance the experience of the user. Furthermore, the placement of the digital objects will deeply influence the real-world nature and interaction of them in accordance with the users. Given this importance, the holograms are put on real surfaces instead of flying. To achieve that, the AR device has to be able to scan the surroundings and to reconstruct the identified faces.

Spatial mapping is the functionality which allows the recognition and reconstruction of the real environment in order to be used for the holographic application. By identifying the surrounding surfaces, the creation of applications which support placement, occlusion, physics and navigation is possible. All these functionalities are essential for the visualization of building models that will be perceived as real objects placed in the surrounding space.

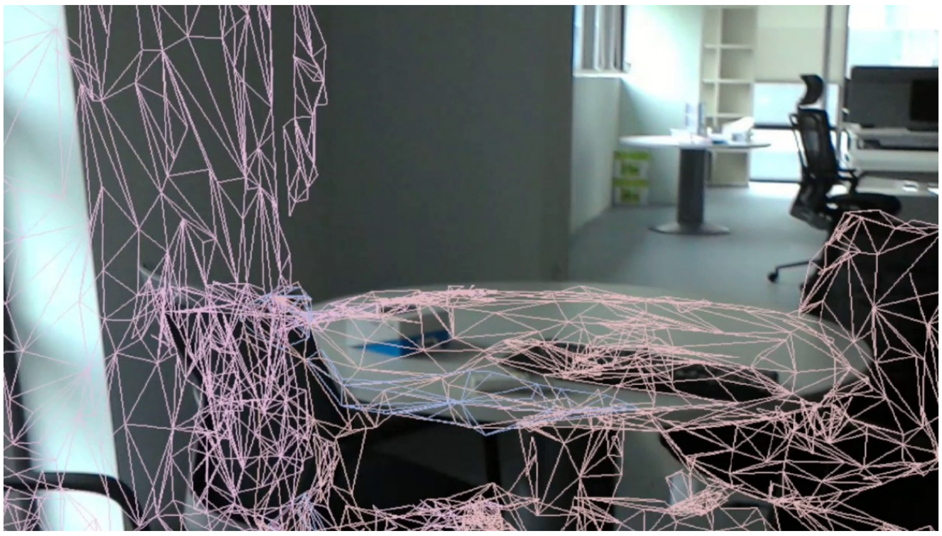


Figure 3.20: Triangulated mesh produced by the spatial mapping functionality depicting the real surfaces [2]

The spatial mapping is highly connected with the device which should be capable of reconstructing the space using hardware like environment-processing cameras, IMU and depth cameras. Once it posses them, the next step is the definition of the extent of the surrounding space that it will reform. It would be impossible for the device to reconstruct the faces of the whole surrounding space especially for outdoor applications. A primitive shape, a sphere or a cube, is used to define the region of interest that will be scanned and reconstructed by the device. A thick triangle mesh is the mean to depict the spatial surfaces after them have been scanned (Figure 3.20)[2]. Its storage, render and processing usually is complicated procedure and demands great computational power. So, once the region of interest has been represented by the mesh it is stored and anchored to be used in the future without the necessity to scan the space again. Nevertheless, each space has to be frequently scanned to identify any changes that might happen meanwhile, which means that a time interval between updates has to be specified. To minimize the usage of the computational resources a number of triangles per cubic meter is defined so as to have an accurate representation of the space without the maximum computational usage. Finally, the chosen material for the representation of the mesh is the one that performs the best with holographic applications.

3.6.8 Menu Creation

Since not only one model is loaded to the AR device, the users have to have the option to visualize only the desired model among a number of options. In order not to develop a separate application for each storey, which would be time consuming and inefficient process, a menu (Figure 3.22) is created from which the user will have the opportunity to visualize or hide (Figure 3.21) the desired model. The other option would be the visualization of all the models at random places, which would highly hinder the performance of the application and would produce a convoluted, chaotic and intricate experience for the users. In essence, the created menu is a list of all available models that someone can visualize.

2 https://www.youtube.com/watch?v=rqzapoTn970

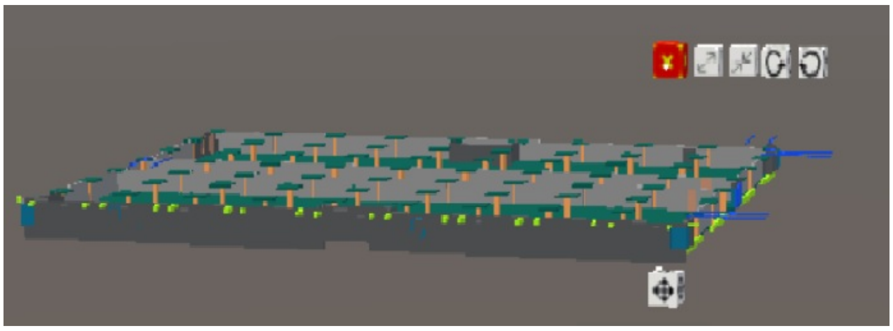
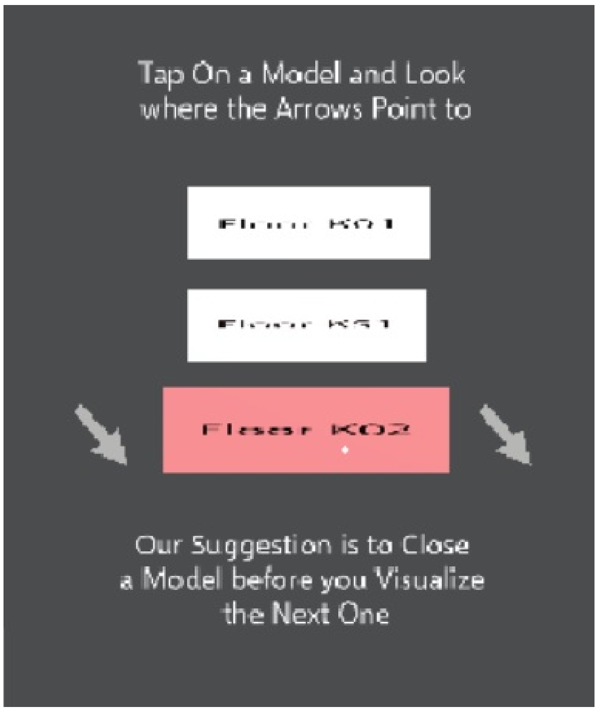


Figure 3.21: A close button is introduced to hide a model



Tap On a Model and Look where the Arrows Point to

Our Suggestion is to Close a Model before you Visualize the Next One

Figure 3.22: A menu is created having a button for each model that the application hosts. The arrows pointing to the direction where the hologram lies

3.7 DEPLOYMENT TO THE AR DEVICE

The final stage of creating an holographic application is the deployment of it in the selected AR device. The first step of the deployment is to build the application which will be deployed later. For this, the scene that will be imported to the application has to be chosen. In combination with it the platform and the Software Development Kit that will be used for this application are specified. In addition to that the programming language which will be used for providing interactions capabilities to the holograms is defined. After that the folder where the building process will take place is defined and the deployment starts.

4 IMPLEMENTATION

4.1 AMC DATASET

As mentioned in the Section 1.2, the sample dataset (Figure 4.1) which is used during this project, was given by *SWECO* and it is in a proprietary file format. Particularly, it was designed using *Revit* which is a software produced by *Autodesk* corporation to facilitate the creation of BIM models. Its in original form the file is not human readable and subsequently it is impossible to develop code in order to manipulate its entities and structure. Furthermore, the proprietary file formats do not comply with the academic scope of this research since they limit the exchange and use of data among people who do not use software by the same vendors and do not support interoperability.

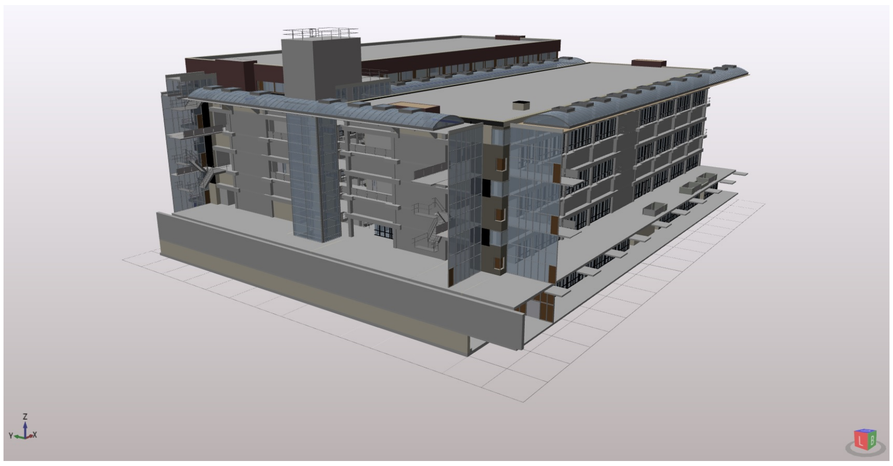


Figure 4.1: The sample dataset which was used during the project is the BIM model of Academisch Medisch Centrum (AMC)

A conversion took place in order to transform it into the corresponding open source scheme which is the IFC and the adopted edition of it is the *2x3*. The conversion is a simple export process (Figure 4.2) and it was performed using the *Autodesk Revit* which was provided by *SWECO*.

4.2 PYTHON & IFCOPENSHELL FOR SEMANTICAL ISOLATION

One of the most crucial parts of this thesis is the extraction of each storey that was hosted in the `.ifc` file. To achieve that, *python*, which is an object-oriented programming language, was used because of its high-level characteristics that allow a fast rate of code development. Furthermore, it is a completely free for use and distribution programming language, even for professional purposes. As it is specified by its official page, *python* developed under an *OSI-approved* open source license and is administered by the *Python Software Foundation*. Finally, it is a programming language

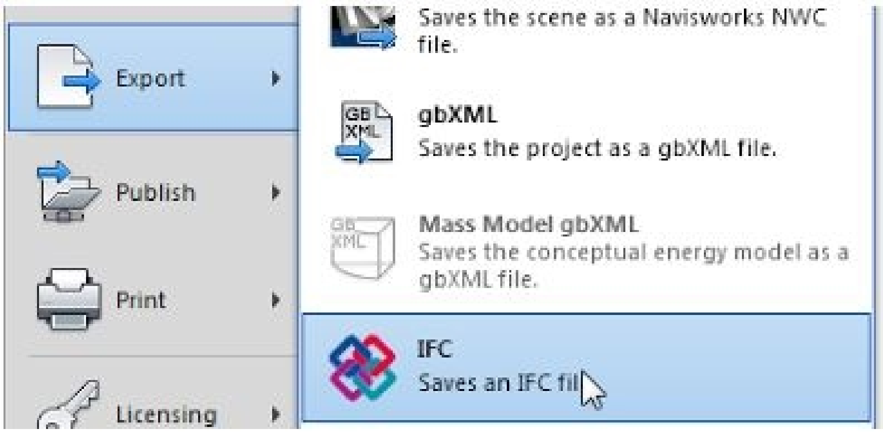

Export
Publish
Print
Licensing
Saves the scene as a Navisworks NWC file.
gbXML
Saves the project as a gbXML file.
Mass Model gbXML
Saves the conceptual energy model as a gbXML file.
IFC
Saves an IFC fil

Figure 4.2: Conversion of the proprietary file format to IFC using *Autodesk Revit*

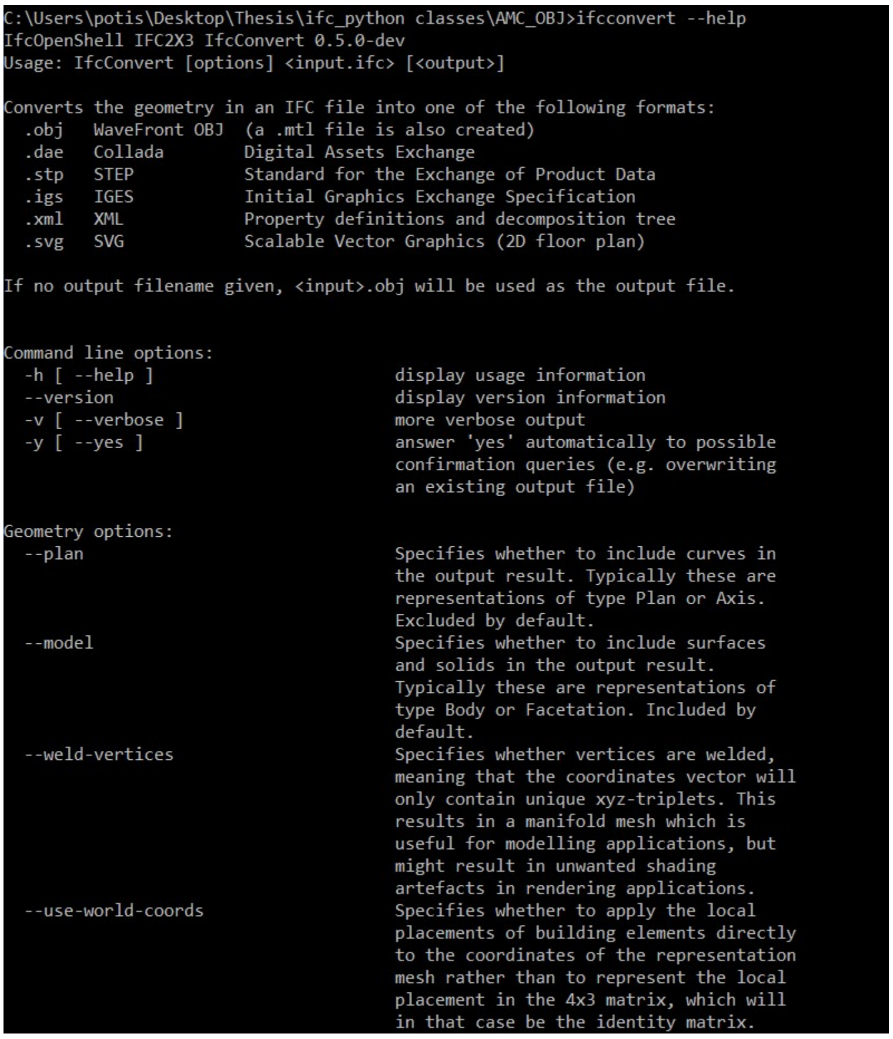
which hosts thousands of of third-party modules and a community which endlessly support with new contribution facilitating the development of new products.

One of the above mentioned modules is the *IfcOpenShell* which is an open source library providing great opportunities considering the manipulation of `.ifc` files by people who is familiar with software development. Building and construction data, which was contained in BIM files, were extracted and formulated in a way that considered entities, which allowed the creation of many functionalities by parametrizing objects and spaces of the model. In order the *IfcOpenShell* to transform the implicit geometry of the `.ifc` file into explicit geometry, it uses the *Open CASCADE*. The creators of it mention that, by performing the geometrical transformation any *CAD* software or modelling package can understand the implicit geometry. A great advantage of *IfcOpenShell* is that there are freely available tutorials and examples that provided with sample code which was used during the completion of the project. The graphical display window, which is supported by *IfcOpenshell*, offered an illustrative depiction of the changes that applied to the model using the classes of the module and helped with correcting inaccuracies and errors that were produced during the development.

Several functions of the *IfcOpenShell* module were used to extract the objects of each storey. Initially, the open function was used to obtain an instance of the original file. After that, the *.by_type* function was used to list all entities that correspond to the classes which were used as parameters and particularly *IFCBUILDINGSTOREY*, *IFCPROJECT*, *IFCSITE* and *IFCBUILDING* were extracted. *IFCRELCONTAINEDINSPATIALSTRUCTURE* entity was used to spot elements that belong to the same level of the spatial project structure. Each element can only be assigned once and it might include a tuple of entities that are contained in the corresponding storey. Each of these entities are recursively traversed to identify elements that were belonged to this using the *.id()* attribute. After having classified them in the corresponding storey, the file functionality, which is supported by *python*, was used to write an `.ifc` compliant file for every floor.

4.3 IFCCONVERT FOR THE CREATION OF THE .OBJ FILES

The `.ifc` file format is not supported by the AR device so its conversion to a supported file format was a prerequisite for its visualization. Except for that, the structure of the `.ifc` file is rather complicated to perform geometrical analysis in order to isolate the exterior shell of the building. *IfcConvert* was used to convert the models, both the original one and the ones that represent each storey, into a format that is supported by `Hololens` and its scheme will be simple enough to apply geometrical calculations on it. *IfcConvert* is a command line app for transforming IFC files into different file formats (Figure 4.3).



```
C:\Users\potis\Desktop\Thesis\ifc_python classes\AMC_OBJ>ifcconvert --help
IfcOpenShell IFC2X3 IfcConvert 0.5.0-dev
Usage: IfcConvert [options] <input.ifc> [<output>]

Converts the geometry in an IFC file into one of the following formats:
  .obj   WaveFront OBJ  (a .mtl file is also created)
  .dae   Collada        Digital Assets Exchange
  .stp   STEP           Standard for the Exchange of Product Data
  .igs   IGES           Initial Graphics Exchange Specification
  .xml   XML            Property definitions and decomposition tree
  .svg   SVG            Scalable Vector Graphics (2D floor plan)

If no output filename given, <input>.obj will be used as the output file.


Command line options:
  -h [ --help ]                      display usage information
  --version                          display version information
  -v [ --verbose ]                   more verbose output
  -y [ --yes ]                       answer 'yes' automatically to possible
                                     confirmation queries (e.g. overwriting
                                     an existing output file)

Geometry options:
  --plan                             Specifies whether to include curves in
                                     the output result. Typically these are
                                     representations of type Plan or Axis.
                                     Excluded by default.
  --model                            Specifies whether to include surfaces
                                     and solids in the output result.
                                     Typically these are representations of
                                     type Body or Facetation. Included by
                                     default.
  --weld-vertices                    Specifies whether vertices are welded,
                                     meaning that the coordinates vector will
                                     only contain unique xyz-triplets. This
                                     results in a manifold mesh which is
                                     useful for modelling applications, but
                                     might result in unwanted shading
                                     artefacts in rendering applications.
  --use-world-coords                 Specifies whether to apply the local
                                     placements of building elements directly
                                     to the coordinates of the representation
                                     mesh rather than to represent the local
                                     placement in the 4x3 matrix, which will
                                     in that case be the identity matrix.
```

Figure 4.3: Functionalities of IfcConvert

It has the capability to receive the directory of an `.ifc` file and transform it to another type of geometrical representation. The user has to specify some parameters to define the format of the output file and to perform any adjustment that he/she might want to apply to it. For this project, the `.obj` file format was used. The `.obj` file format supports geometry in form of vertices/edges/faces and parametric surfaces, vertex normals, textures, material properties, and groups [McHenry and Bajcsy, 2018]. This file format includes a number of lines, which depends on the complexity of the model. Each of the lines starts with a key indicating the context of the line and a value describing the entity of the line (Table 4.1).

Furthermore, *IfcConvert* except for data conversion provides more functionalities depending on the parameters that have been specified. Such functionalities are: inclusion of curves in the output result, applying the local placement of building objects directly to the coordinates of the representation mesh, conversion of the geometrical output to the measurement units in which it is originally defined, sewing the faces of *IfcConnectedFaceSets* together, merging all *IfcOpeningElement* operands into a single operand before applying the subtraction operation, disabling the `Boolean` subtraction of *IfcOpeningElement* representations from their *RelatingElements* and etc.

Key	Description
#	Comment
v	Vertex
l	Line
f	Face
vt	Texture Coordinate
vn	Normal
g	Group
...	...

Table 4.1: Common keys in an `.obj` file

4.4 PYTHON & CGAL FOR EXTRACTION OF THE EXTERIOR ENVELOPE

As mentioned above, *python* is an easy-to-use programming language providing many functionalities in many different developments fields. Among other functionalities it provides modules and libraries to manipulate models with geometrical presence. Many geometrical calculations can be performed on both *2D* and *3D* models to redefine, clean, simplify and enhance existing designs.

For this project, the Computational Geometry Algorithms Library (CGAL) was used to apply mathematical algorithms to manipulate the geometry of the given BIM model. It is a software package that grants access to effective and robust geometric algorithms [1]. The official page mentions that CGAL is used in various areas needing geometric computation, such as geographic information systems, computer aided design, molecular biology, medical imaging, computer graphics and robotics. Furthermore, this computational geometry library offers data structures and algorithms like triangulation techniques, *Voronoi* diagrams, `Boolean` operations on polygons and polyhedra, point set processing, arrangements of curves, surface & volume mesh generation, geometry processing, alpha shapes, convex hull algorithms, shape analysis and *AABB* & *KD* trees [1]. It is developed in the C++ programming language but it supports *python* compatibility using bindings.

For the creation of the bounding box, described in Bounding Box, the *.obj* file was traversed in order to specify all the lines that correspond to the vertices of the model. For each identified vertex, an entity in a list was created having as first element a unique id which was followed by the coordinates of the corresponding vertex. This list was passed as an argument to a function which traversed the list and, using the `Point_3` class, constructed objects which were points in the three-dimensional Euclidean space. Every newly created object was assigned to a list corresponding to each side of the bounding box.

This list of `Point_3` objects was used to create the rays that passed through the model in order to make the distinction between triangles that belong to the exterior part and those that are part of the interior. So, the lists of points of the opposite sides were passed as arguments to a function that used them as start and end points of the rays. The list corresponding to the one side was used as start point and it was chosen depending on the complexity of this side of the model. The side of the bounding box which was closer to the most complex side was used as starting side to achieve greater efficiency of the raycasting algorithm.

The rays were constructed using `Segment_3` class of the CGAL library. `Segment_3` is a directed straight line segment in the three-dimensional Euclidean space, that is a straight topologically closed line segment `[p,q]` connecting two points `p` (source) and `q` (target), having as length the Euclidean distance between `p` and `q`. In order to achieve the extraction of the triangles that belong to the exterior shell but were

1 https://www.cgal.org/

obstructed by others, (Section 3.4.2), not only one end point was specified but 9 of them. A different segment was created for each of them. As end points for these segments, the points that surrounded the opposite of the start point were picked. To define the end points, the step, which was used for creating the distance among points, was utilized to find out the surrounding points. By either adding or subtracting the step or any multiple of the step to the coordinates of the central point, it is possible to achieve the creation of `Segments_3` with the desired inclination. Any created segment was appended to the list of rays to be used for the exterior extraction process.

For the raycasting process it was essential to reconstruct the faces or triangles of the model using entities that are identifiable by CGAL. To achieve that, the original file was traversed and lines starting with `'f'`, indicating faces were isolated and populated a list. Each of these lines had three numbers after `'f'`, which corresponded to the ids of the vertices that form this face. These ids were used to define the lines of the vertices in order to use their coordinates to reform the geometry of them in `Point_3` class objects. These point objects were passed as arguments, to reconstruct the face as `Triangle_3` object, a class that belongs to CGAL and represents a triangle in the three-dimensional Euclidean space.

The extraction process took place by identifying the triangles that intersect with each segment. A calculation of the distance between the triangles and the start point was performed. The list which hosted the segments was traversed and checked with regard to the triangles that each of the segment intersected. The intersection point of these two objects was defined as the point `p` that is part of both `obj1` and `obj2`. This process was performed using the intersection function of CGAL which can check for intersection of `2D` objects, `3D` objects and a combination of them depending on the arguments that are passed to the function. To identify with which triangles a ray intersected, an iteration over all triangles of the model was essential. Two variables, `min` and `max` were initiated to keep the distance between the start point and the point on which the segment intersected with the triangle. To the `max` variable an extremely small value, compared to the dimensions of the model, was assigned while to the `min` variable an extremely large one. This happened in order the first time that the iteration will identify a triangle with which the segment intersects, to assign to both to the `min` and `max` values the calculated distance between the start and the intersection point. This comparison was repeated each time a triangle intersected with a segment. After having traversed all the triangles of the building, a reference of the faces that had the `max` and the `min` distances (triangles of the exterior envelope) from the start point were kept and populated a set, in order later to reconstruct the model.

To obtain a decent representation of the exterior part of the building, many `3D` points have to be introduced on the faces of the bounding box. Considering that each triangle, almost `800.000` of them, had to be checked whether it intersects with any of the created segments, the necessity of great computational power arises with regard to the complexity of the model and the number of points that will be introduced. This shortcoming was easily identifiable during the execution of the raycasting algorithm. To overcome with this problem, the `python's multiprocessing` module was used to spread the workload to all the cores of the machine. `Multiprocessing` module is a package that supports spawning processes using an `Application Programming Interface (API)` similar to the threading module and offers both local and remote concurrency, effectively side-stepping the `Global Interpreter Lock` by using sub-processes instead of threads [2]. The module allows the programmer to fully leverage multiple processors on a given machine and it has the capability to run on both `Unix` and `Windows`[2].

Willing to produce a respectable outcome, many `3D` points were created on top of the bounding box's faces. The number of points in combination with the triangles of the model, created a large number of iterations that was impossible to be handled

2 https://docs.python.org/2/library/multiprocessing.html

properly by a personal computer of 8 cores. Access to the 40-cores-machine of TU Delft's Architecture faculty was provided to be feasible to run the raycasting algorithm with the given data. 30 out of 40 cores of the server were used to parallelize the exterior obtaining process by distributing the creation of points and segments, the raycasting, the intersection and the distance calculation and comparison.

PuTTy was used to accomplish the remote connection with the university's server. Through it, the installation of Python 3 programming language, which supports the usage of CGAL using bindings, was achieved. Furthermore, the execution of the code was performed using a terminal which is provided by PuTTy. The transfer of the files from the local machine to the server and vice versa was achieved by using FileZilla. FileZilla is an File Transfer Protocol (FTP)/Secure File Transfer Protocol (SFTP) client for Windows and can act as a site manager with the potential to perform multiple simultaneous transfers. FileZilla was used to upload the BIM model to the server in order the geometrical calculations to be performed on that and to download the generated simplified file.

4.5 UNITY

Unity (commonly known as Unity3D) is a game engine and Integrated Development Environment (IDE) for creating interactive media, typically video games. The Unity game engine is developed by Unity Technologies in Denmark. It mainly comprises eight products such as the Unity, Unity Pro, Asset Server, iOS, iOS Pro, Android and Android Pro and allows programmers to quickly develop a scene by using the visual integrated development environment of Unity3D, without the necessity to write complex codes [Xie, 2012]. Some keys benefits of Unity3d game engine are:

- **Documentation:** The Unity engine comes with complete documentation with examples for its entire API. This is the biggest benefit of Unity since it leads to increased productivity.
- **Developer Community:** There is an active on-line developer community which provides assistance to users. The Unity Technologies' developers also are very willing to add features to the engine at a users request.
- **Drag-n-Drop:** Content is listed in a tree and is added to an environment in a drag-n-drop manner. Objects in the environment are listed in a separate tree, each of which can be assigned multiple scripts written in C#, JavaScript or Boo as well as physics and rendering properties.
- **Physics & Rendering:** By using physics properties, objects can be given mass, drag, springiness, bounciness and collision detection as well as be assembled using a variety of joints. The physics properties are simulated by nVidia's PhysX engine. The rendering properties include shader and texture assignment which affect the appearance of visible objects.
- **Multiplatform Distribution:** The Unity engine's editor runs on OSX, however applications created using Unity can be compiled for OSX, Windows or as a Web-Player (which runs in a web browser via a plugin, similar to Adobe Flash).
- **Low Cost:** It has a relatively low cost for a complete game engine (although more expensive than the free open source engines).

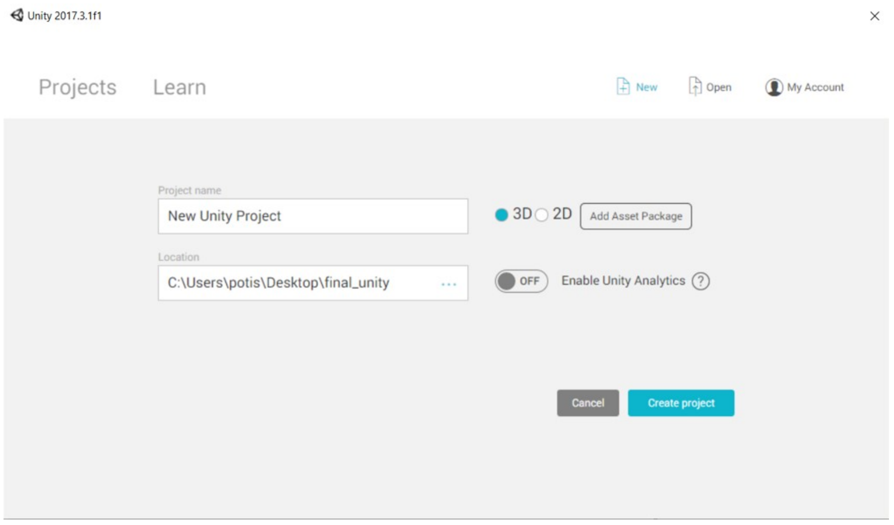
In June 27, 2013 Unity Technologies have entered into an agreement with Microsoft to collaborate on development tools to make it even easier for developers to bring popular games, entertainment and apps to Microsoft platforms [3].

3 https://unity3d.com/company/public-relations/news/unity-announces-strategic-collaboration-microsoft

One of the platforms to which Unity supports with functionality is Microsoft Hololens. In particular they have developed the HoloToolkit-Unity which is a collection of scripts and components that are intended to accelerate the development of applications targeting to Microsoft HoloLens and Windows Mixed Reality headsets [4]. It is aimed at reducing barriers to entry to create mixed reality applications and contribute back to the community[4].

4.5.1 Scene Configuration & Tools Installation

Since Unity was decided to be the environment in which the models will be loaded and manipulated to produce holograms, settings and tools had to be installed to facilitate this process. Firstly, a new project was created specifying the directory where the Unity's project and holographic application will be hosted (Figure 4.4).



Project name: New Unity Project
[✓ 3D] 2D
Add Asset Package
Location: C:\Users\potis\Desktop\final_unity
OFF Enable Unity Analytics

Figure 4.4: Creation of a new directory in order to host the Unity project and the holographic application

The HoloToolKit for Unity is an essential part of the development of holographic applications since it highly facilitates the configuration of the holographic scene and provides functionalities which are used during the run of the apps. It is a free of charge, a .unitypackage file that someone can download and import into the Unity project (Figure 4.5).

The HoloToolkit for Unity includes many APIs to accelerate the development of AR projects for HoloLens and other Immersive Headsets (IHMD) [4](Figure 4.6a). Some of the main functionalities that it supports are:

- **Input:** Scripts that leverage inputs such as gaze, gesture, voice and motion controllers. Furthermore, Mixed Reality camera prefabs are included in it.
- **Sharing:** Sharing library enables collaboration across multiple devices.
- **Spatial Mapping:** Scripts that allow applications to bring the real world into the digital using HoloLens.
- **Spatial Sound:** Scripts to help plug spatial audio into your application.
- **UX Controls:** Building blocks for creating good User Experience (UX) in your application like common controls.
- **Utilities:** Common helpers and tools that you can leverage in your application.

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

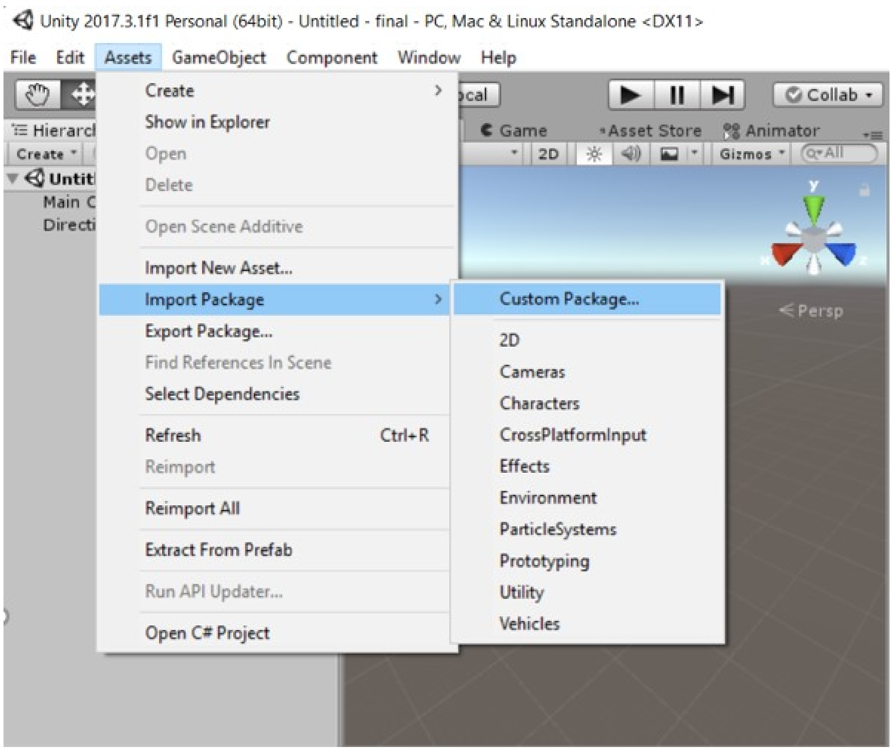


Figure 4.5: Import of HoloToolkit to Unity

- **Spatial Understanding:** Tailor experiences based on room semantics like couch, wall etc.
- **Build:** Build and deploy automation window for `Unity Editor`.

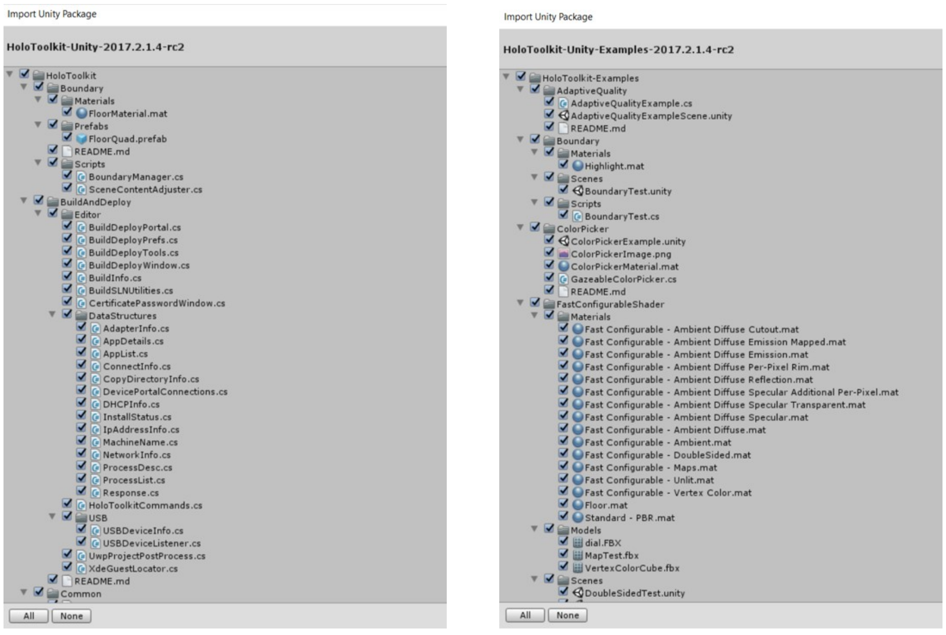
Apart from `HoloToolkit-Unity`,the `HoloToolkit-Examples .unitypackage` file (Figure 4.6b) was imported to provide with test scenes and assets that verify functionality of `HoloToolkit-Unity` components and host samples/examples that exemplify the `HoloToolkit` to help solve interesting scenarios[4].

4.5.2 Import & Rescale

The import of the models to `Unity` game engine is a simple and straight forward process. A different folder created for every floor of the building. All the objects that were belonged to a certain storey, were imported to the corresponding folder by drag n drop. They retained their original dimensions as these have been specified by the designers of the model.

The dimensions of the models were too large for the purpose of this application and so a rescaling factor had to be applied to them. While trying to achieve that using the `Transform` tab of `Unity's` Graphical User Interface (GUI), it was realized that the faces of the objects became darker and it was impossible to change them using the material's settings. Most probably the `Shader` of the objects could not handle properly the tiny scale because of floating point precision. To overcome this problem a script was applied to change the global scale of the whole project which means that every object was multiplied by `0.00005f`.

Another issue that was observed during the import was that the models were introduced vertically (Figure 4.7) in the holographic scene due to the inconsistency of `Unity's` axis with the ones of the software in which the design of the BIM model took place. To position them properly in the holographic scene the rotation field in the `Transform` tab was used. A rotation of `270` degrees along the `X` axis was performed and that leaded to the correct placement of the hologram in the scene.



Import Unity Package

HoloToolkit-Unity-2017.2.1.4-rc2

```
HoloToolkit
  Boundary
    Materials
      FloorMaterial.mat
    Prefabs
      FloorQuad.prefab
    README.md
    Scripts
      BoundaryManager.cs
      SceneContentAdjuster.cs
  BuildAndDeploy
    Editor
      BuildDeployPortal.cs
      BuildDeployPrefs.cs
      BuildDeployTools.cs
      BuildDeployWindow.cs
      BuildInfo.cs
      BuildSLNUtilities.cs
      CertificatePasswordWindow.cs
      DataStructures
        AdapterInfo.cs
        AppDetails.cs
        AppList.cs
        ConnectInfo.cs
        CopyDirectoryInfo.cs
        DevicePortalConnections.cs
        DHCPInfo.cs
        InstallStatus.cs
        IpAddressInfo.cs
        MachineName.cs
        NetworkInfo.cs
        ProcessDesc.cs
        ProcessList.cs
        Response.cs
      HoloToolkitCommands.cs
      USB
        USBDeviceInfo.cs
        USBDeviceListener.cs
      UwpProjectPostProcess.cs
      XdeGuestLocator.cs
    README.md
  Common
```

(a)

Import Unity Package

HoloToolkit-Unity-Examples-2017.2.1.4-rc2

```
HoloToolkit-Examples
  AdaptiveQuality
    AdaptiveQualityExample.cs
    AdaptiveQualityExampleScene.unity
    README.md
  Boundary
    Materials
      Highlight.mat
    Scenes
      BoundaryTest.unity
    Scripts
      BoundaryTest.cs
  ColorPicker
    ColorPickerExample.unity
    ColorPickerImage.png
    ColorPickerMaterial.mat
    GazeableColorPicker.cs
    README.md
  FastConfigurableShader
    Materials
      Fast Configurable - Ambient Diffuse Cutout.mat
      Fast Configurable - Ambient Diffuse Emission Mapped.mat
      Fast Configurable - Ambient Diffuse Emission.mat
      Fast Configurable - Ambient Diffuse Per-Pixel Rim.mat
      Fast Configurable - Ambient Diffuse Reflection.mat
      Fast Configurable - Ambient Diffuse Specular Additional Per-Pixel.mat
      Fast Configurable - Ambient Diffuse Specular Transparent.mat
      Fast Configurable - Ambient Diffuse Specular.mat
      Fast Configurable - Ambient Diffuse.mat
      Fast Configurable - Ambient.mat
      Fast Configurable - DoubleSided.mat
      Fast Configurable - Maps.mat
      Fast Configurable - Unlit.mat
      Fast Configurable - Vertex Color.mat
      Floor.mat
      Standard - PBR.mat
    Models
      dial.FBX
      MapTest.fbx
      VertexColorCube.fbx
    Scenes
      DoubleSidedTest.unity
```

(b)

Figure 4.6: (a) HoloToolkit for Unity. (b) HoloToolkit-Examples

4.5.3 MixedReality Materials & meshCollider

One of the most important aspects of the visualization, is the use of colors and patterns that give an intuitive perception of what a user stares at by making the distinction among the different objects of the model. To achieve that materials of different colors were created for every group of objects that were hosted in the building model (Figure 3.13). The shader which was used for every material provided by the `HoloToolkit`. It is a shader which intents to holographic applications and it is called `MixedReality Shader`. It is created to provide the best feasible performance with the HPU and to maximize the frames per second (FPS) that can be achieved during the run of the application. Options like shadows and reflections were disabled in order not to hinder the visualization of the models.

Each game object with a geometrical presence in `Unity` has a component which called `Mesh Renderer`. This component is responsible to handle the geometry of an object and render it at the place specified by the `Transform` component. This component hosts a field called `Element` (Figure 4.8) in which the material of the corresponding object is assigned.

Finally, in order the user's gaze to be detectable by the object that is being looked at a certain time, a `Mesh Collider` (Figure 4.8) was assigned to each object, as it is the component which is responsible for the collision detection. It automatically receives the mesh that forms an object and assigns it to the `Mesh` field in order to specify its shape.

4.5.4 onFocus Highlighting & Metadata Visualization

The interaction of the users with the holograms is highly depended on where they look at each moment. The `InputManager` object, was introduced when the `HoloToolkit .unitypackage` was imported to the project, containing scripts which handle the gaze of the users.

Particularly, it contains the `GazeManager` script which handles the maximum collision distance, the stability of the gaze, stabilize the user's gaze to account for head jitter and makes the connection with the selected cursor. In addition to this, a `C#` script was assigned to every object of the models in order to change its color.

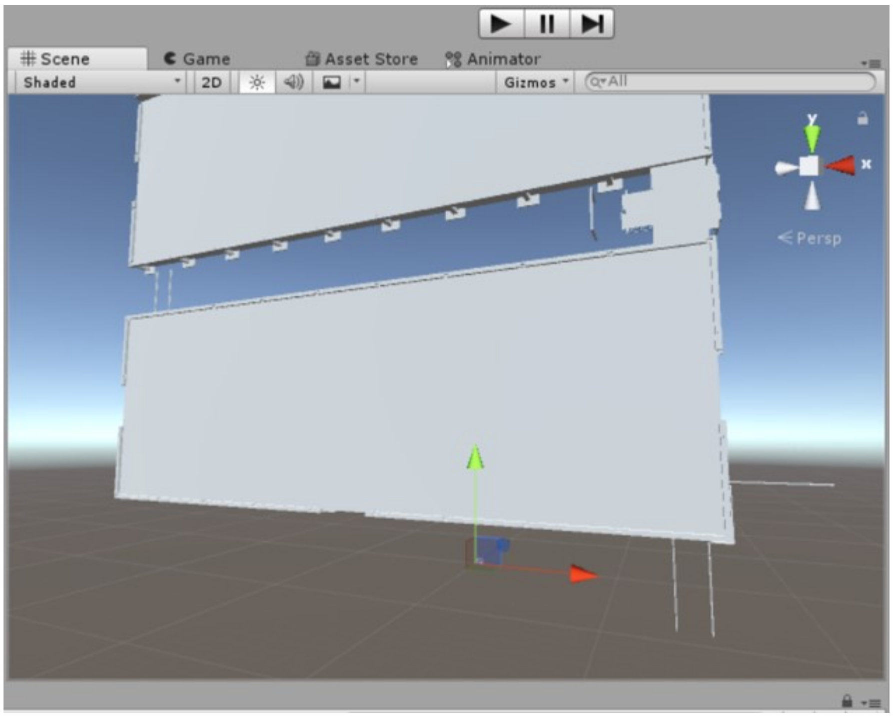
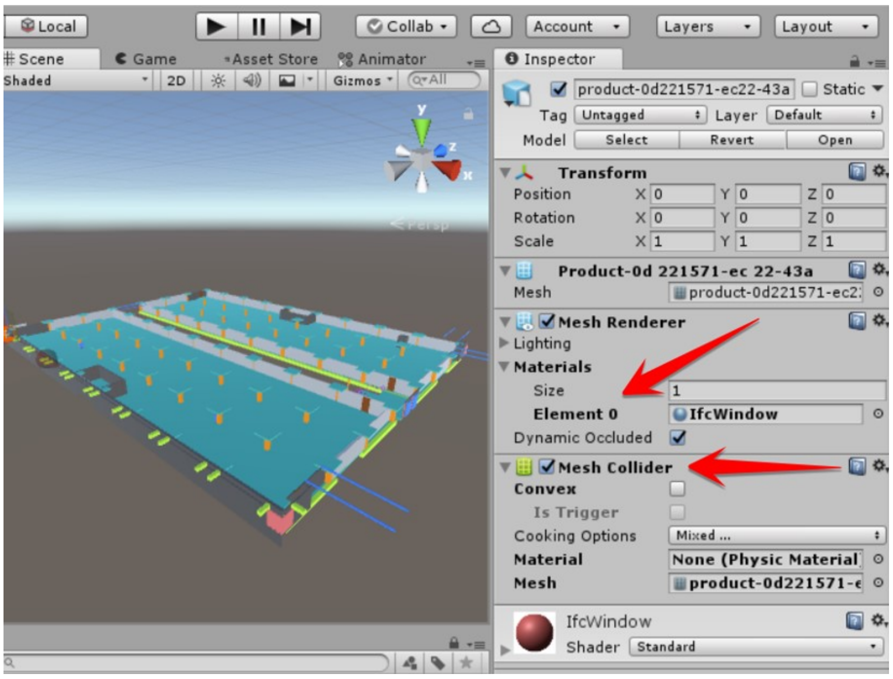


Figure 4.7: Vertical positioning of the imported models

So, once the `Mesh Collider` component identified the gaze of a user on the corresponding object, it created an instance of the object. Through this instance it was possible the material's color property, which is part of the `MeshRender` class, to be accessed of the gazed object. To this property the variable `focusedColor`, having the value red part of the `Color` class, was assigned. This results in changing of the color of the observed object (Figure 3.14). Similarly, when the gaze of the user moved away from the previously stared object the `defaultColor` variable was assigned to the `MeshRenderer` of the object.

Once the users have stared at the object of their interest, they have to be able to visualize information about it. To project this information a user has to perform the tap gesture. The `InputManager` object contains a child object, which is called `GesturesInput`, that is responsible to handle the gestures of the users. Once, `Hololens` identified the hand of the user in its field of view the cursor was transformed from disk to torus to notify the user that his/her hand has been recognized by the device. When the user performs the tap gesture, an instance of the observed object is created.

This instance is used to connect the observed object with the `TextPrefab`,which has earlier been introduced in the scene. `TextPrefab` is a game object that was imported by the `HoloToolkit` to visualize text in holographic applications. Its color and texture were adjusted to perform the best using Hololens. The instance of the observed object was used to retrieve its name and its position. The name of the object had earlier been modified to include the metadata, which was considered relevant to be visualized. Once, this information was retrieved the name variable was used to reform the text by adjusting its properties. Furthermore, not to obstruct the user's attention from the model, its position was extracted and assigned to a `Vector3` variable, slightly changed, in order later to be used as the position of the `TextPrefab`. So, the metadata was always visualized above the observed object, since its position was dynamically being retrieved even if the model had been placed on a different place using the transport capability.



product-0d221571-ec22-43a ☐ Static
Tag Untagged Layer Default
Model Select Revert Open

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Product-0d 221571-ec 22-43a
Mesh product-0d221571-ec2:

✓ Mesh Renderer
Lighting
Materials
Size 1
Element 0 IfcWindow
Dynamic Occluded ✓

✓ Mesh Collider
Convex ☐
Is Trigger ☐
Cooking Options Mixed ...
Material None (Physic Material)
Mesh product-0d221571-e

IfcWindow
Shader Standard

Figure 4.8: Assignment of meshMaterial and meshCollider to one of the objects of the building

4.5.5 Grow–Shrink-Rotate

Grow, shrink and rotate functionalities were combined with cubic buttons which were placed above the models to be visible to the users. On these buttons a material was put, as this described in Material Creation & Assignment, having as a symbol one that offers an intuitive perception of its functionality. When a user stares at one of these buttons, it changes its color into yellow and it gets bigger to allow the user to know that he/she is gazing at it (Figure 3.15).

To achieve the massive grow, shrink or rotate of every object of a model, all the objects were assigned to a greater object which adopted the combined geometry of objects. So, each time a user tapped on one of these buttons an instance of this greater object was obtained and passed to the corresponding `C#` script. Regarding the grow and shrink functions, it is possible to modify the dimensions of the storey by multiplying the instance of the model with a constant value in order to adjust the size of the model depending on the tapped button.

The same pattern follows the rotate capability. As soon as a user tapped on one of the two rotate buttons, an instance of the object was created and passed for modification. This instance was used to alter the class which described the position of the object. It changed the direction at which the model was pointing, by multiplying its original position with a fixed value indicating degrees of transformation (Figure 3.19). After having tapped at the button, the model retains the modified position even if the user closes and reopens the model. The application assumes that this position is the most convenient for the user.

4.5.6 Positioning & Spatial Mapping Prefab

The placement process of the hologram was achieved using with the spatial mapping functionality, which was provided as prefab by the `HoloToolkit.unitypackage`. A button was created underneath each model to support its transportation. A material assigned to it having as its texture a `bitmap` image that provides the user with an perceptive impression of the functionality of the button. When the user stared at it, it changed its color and got bigger. As soon as they tap on it, the scanning process starts and a mesh is visualized indicating the digital reconstruction of the space and the ability to move the model to the desired position is provided.

By making the tapping gesture on the transportation button, a signal is sent to the spatial mapping functionality to start scan the place in order to identify horizontal surfaces that could potentially host the models. During the scan procedure the model sticks in front of the field of view of the user and the surrounding environment is covered by a triangulated mesh indicating the progress of the scanning. Even during this process the users can perform the tap gesture in order to place the model on one of the identified surfaces. This action will leave the model on the place where the user was gazing at this point in time and the spatial mapping process will stop to operate. The model will stay there as long as the application will be on and can even be anchored in order to be visualized in the same place for future use of the application.

4.6 VISUAL STUDIO DEPLOYMENT

The deployment procedure is performed to install the created application to the device. Certain settings, such as the software's edition and the way of connection of the deployment software with the device (`usb` or `WiFi`) have to be set. If the building process has been performed accurately, there is no need for debugging action, otherwise, errors will be raised indicating the problematic parts. Finally, only for the first time of pairing the AR device with the pc, in which the applications was developed, a coupling process has to be followed. To achieve that the device is set into developers mode and a `PIN` code is specified by the device to be typed in the developers portal.

4.7 HOLOLENS

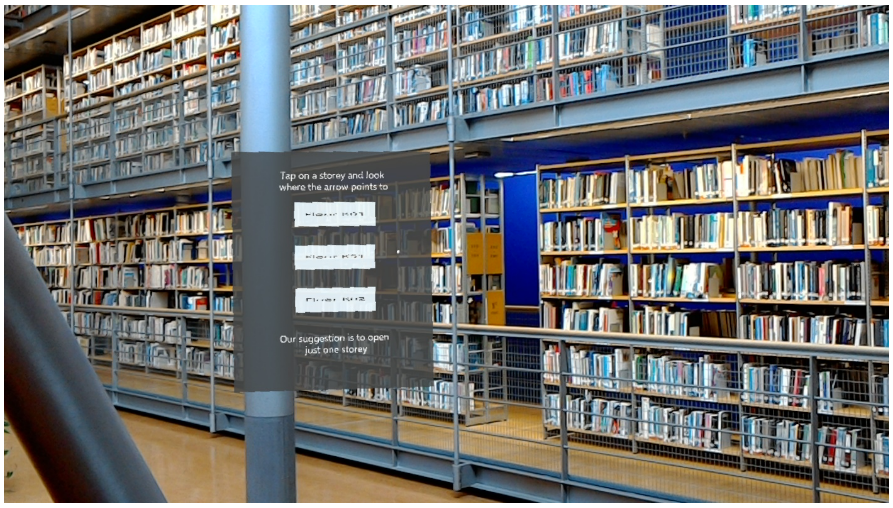
As soon as the application has been loaded to `Hololens`, an icon of it should be visualized in the list of applications with the name of the application which earlier had been specified by the developer. A single tap gesture is enough in order to start the app. A plane surface will pop up, saying `‘‘made with Unity’’`, while the application is being loaded. When it finally has been loaded, the menu will be visualized in front of the user's field of view.

5 RESULTS

This chapter aims to provide a visual depiction of the final holographic application. Different shots were taken through `Hololens` in order to offer an indication of how the models and the menu look like. Since it is an application which is developed for an AR device, the surrounding space is obvious to the user. So, TU Delft's library was used as the background for the acquired pictures.

5.1 MENU

A menu (Figure 5.1) was created in order the users to have the option to visualize only the desired model. The created menu hosts a different button for each model that a user can visualize. A dark color is used as background color for the menu, not black since it is impossible for the device to remove the light from the environment, in order to make contrast with the created buttons and the text that lie on it. The messages, which appear on it, intend to provide some guidance to the users on how to use the application. Their font is white in order to be easily readable by the users. In the same way, the color of the created buttons is white in order them to be quickly identifiable on the dark background. They host a text which corresponds to the name of the storey that they visualize by tapping on them.


Tap on a storey and look where the arrow points to

Our suggestion is to open just one storey

Figure 5.1: Menu of the holographic application

Figure 5.2 demonstrates how a button of the menu changes color and gets bigger when the users stare at it. The red color that was picked as "on focus" color intends to give a clear indication to the users that the corresponding button is ready to interact with them.

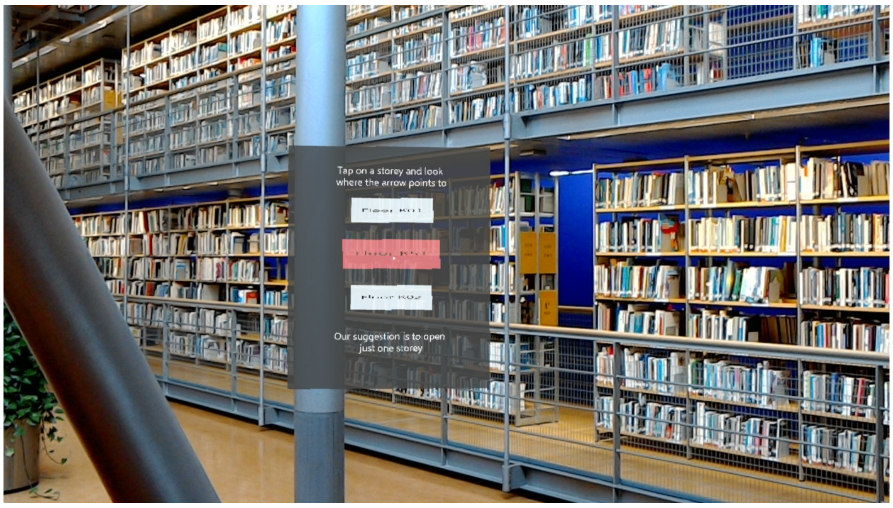

Tap on a storey and look where the arrow points to
Our suggestion is to open just one storey

Figure 5.2: One of the Menu's buttons while being focused

After the user has tapped on it, the corresponding hologram pops up. Due to the limited field of view of `Hololens`, it might not be obvious to the user where the model is. This is the reason why two arrows were placed to the left and to the right of the tapped button (Figure 5.3). They point to the direction that the model has been placed when the button is pressed for the first time. If the users use the replacement capability, the arrows no longer will point to the direction of the hologram, even if they close and re-open the model.

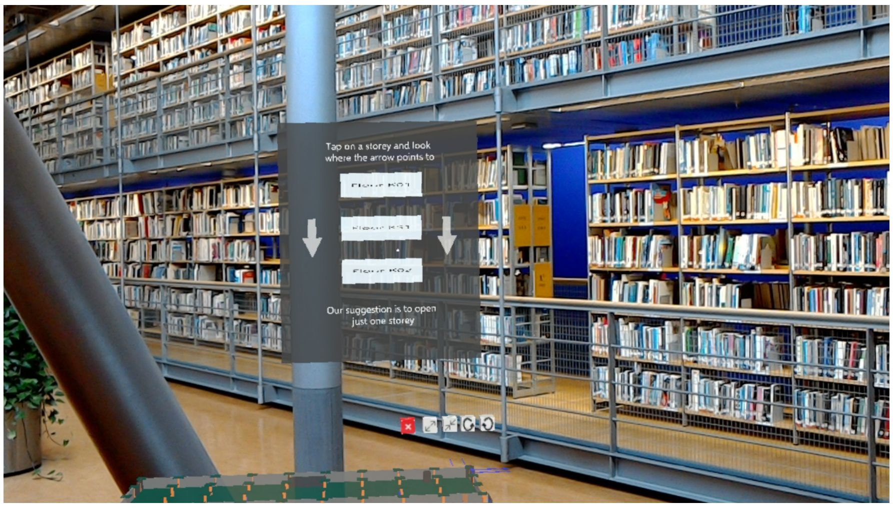

Tap on a storey and look where the arrow points to
Our suggestion is to open just one storey

Figure 5.3: Menu's arrows indicating the initial position of the Hologram

5.2 onfocus highlighting & metadata

Except for the button of the menu that the users stare at, they have to be aware of the building object that they look at after they have visualized a hologram. The same concept for the building objects as well. As soon as the users stare at a certain objects, this objects turns its color into red (Figure 5.4). That gives a clear perception to the users of which objects they look at any given time.

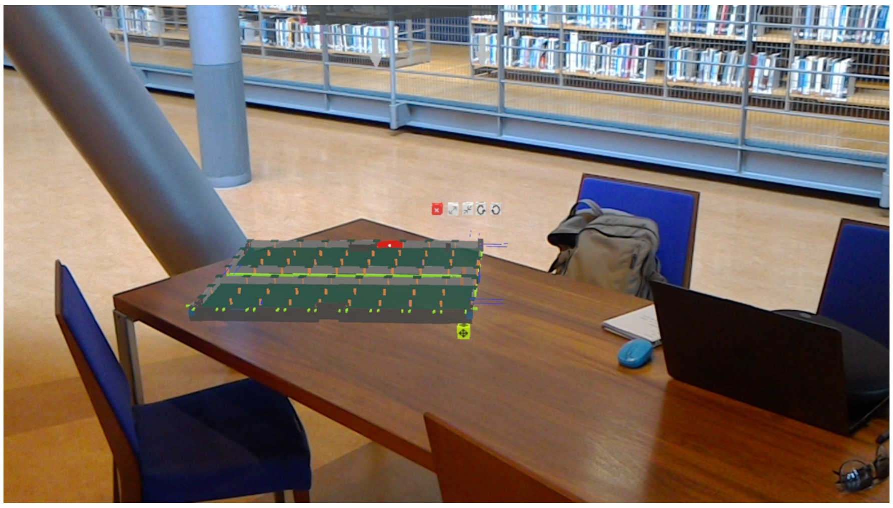


Figure 5.4: Highlighting of the object which is being gazed

After they have focused on the desired building object, they perform the tap gesture in order to visualize metadata information about the stared object. A yellow text is visualized above corresponding building element to facilitate the users to read it. It provides information about the type, the id and the name of the object (Figure 5.5)

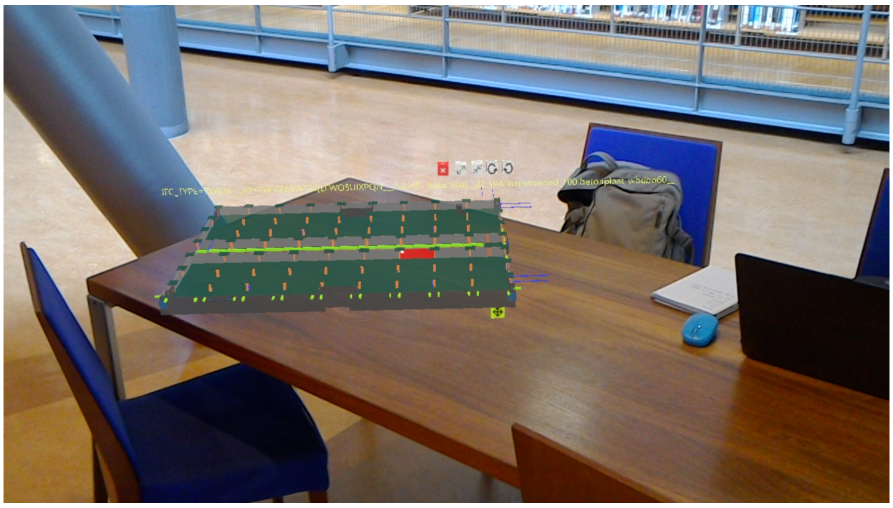


Figure 5.5: Metadata visualization above the tapped object

5.3 GROW-SHRINK-ROTATE

Other capabilities that the application presents in order the users to interact with the holograms are the rescaling and reorientation of them. A series of button have been placed above the model, having intuitive icons indicating their functionality, which are visualized together with the model. Figure 5.6 shows how the model gets bigger after the user has tapped the second button form left to right.

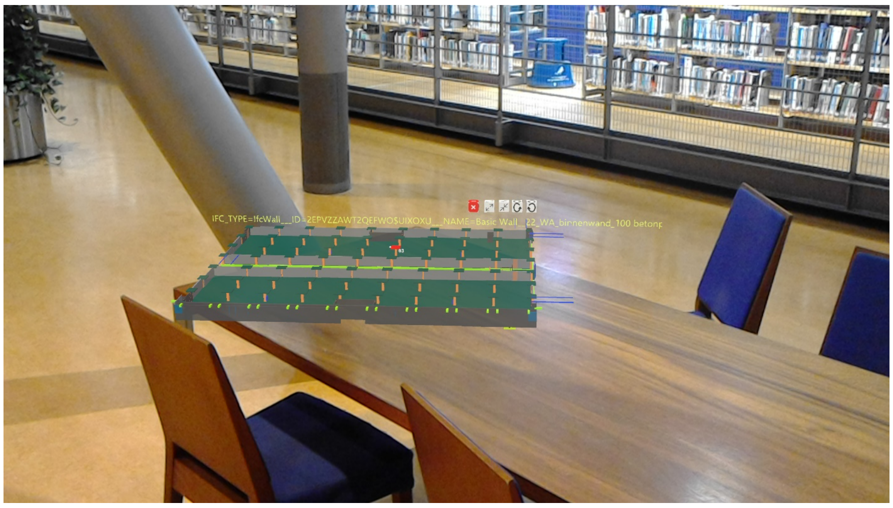

IFC_TYPE=IfcWall___ID=2EPVZZAWT2QEFWO$UIXOXU___NAME=Basic Wall__22_WA_binnenwand_100 betonp
03

Figure 5.6: The hologram has been grown after the user has tapped on the corresponding button which is above the model

As with the grow functionality, the Figure 5.7 shows how the hologram has been placed after the rotate button has been tapped. The hologram can be observed by different point of views in order to provide a clear perception of the interior of the building.

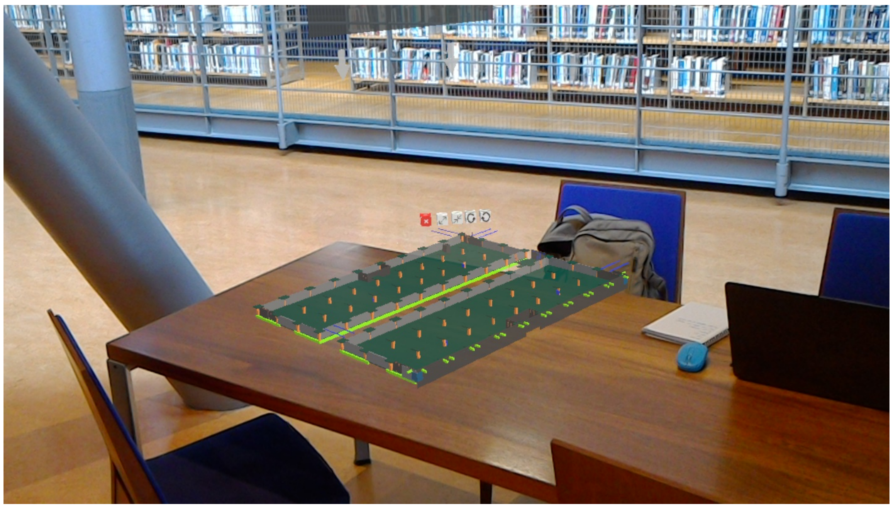


Figure 5.7: The hologram has been rotated after the user has tapped on the corresponding button which is above the model

5.4 SPATIAL MAPPING

Last interaction capability which is offered by the holographic application is the replacement of the model. This was achieved by using the spatial mapping functionality that `Hololens` supports. A button was placed underneath each model which by tapping on it allows the users to place the model on the surface of their desire. When the users stare at this button, it changes its color into yellow and gets bigger to signify that they look at it. As soon as they perform the tap gesture the spatial mapping starts to scan the surrounding environment. Gradually, a triangulated mesh is constructed which represents the surfaces have been identified (Figure 5.8).

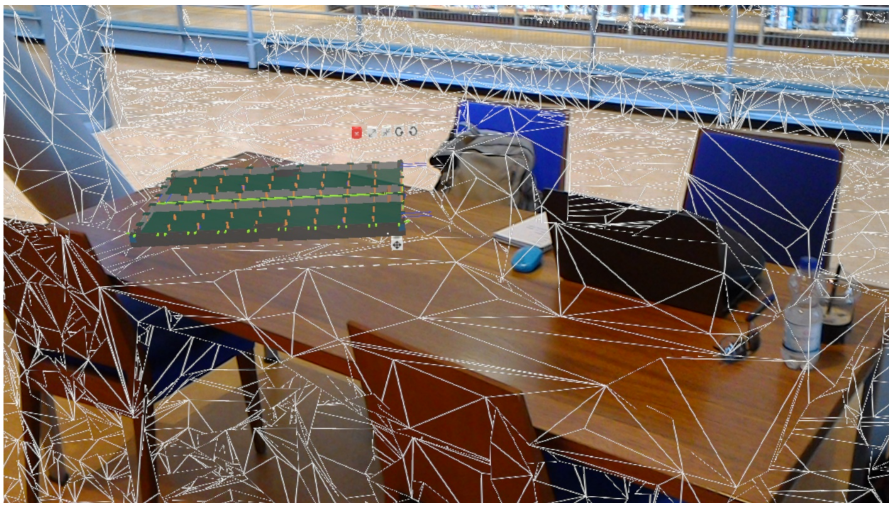


Figure 5.8: Spatial mapping capability constructs a triangulated mesh which represents the identified surfaces

In the same time, the model starts to follow the users' gaze. The created mesh gives a clear indication of where the users can place the model. By placing the model on one of the identified surfaces, the application provides a way more immersive experience compared to the original placement of the model. To leave the model on the new position, the users only have to perform the tap gesture. Since, the users are not always able to freely move in the space, the replacement functionality allows them to put the model on a place that will be convenient for them to observe the model.

5.5 EXTERIOR ENVELOPE

The main goal of this application is to visualize the exterior envelope of the BIM model in order to serve as a digital maquette. The original model was almost 128MB and it was impossible for Hololens to visualize it. So, the extraction of the outer took place in order the visualization through the AR device to be feasible. The size of the produced file is 8.5MB.

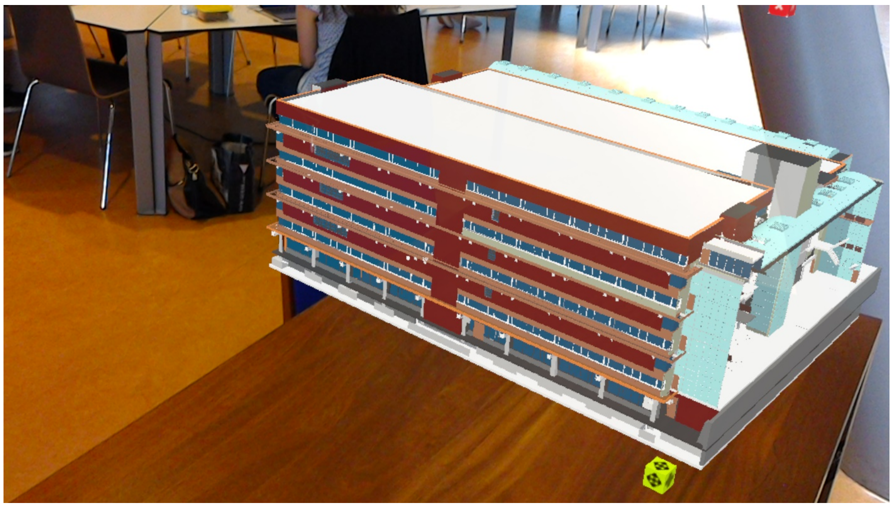


Figure 5.9: Exterior envelope of the AMC(1)

Despite that the size of the model is way smaller, Hololens crashed many times while it was visualizing it. Furthermore, during the visualization of the model, phenomena like trembling or shaking were observed.

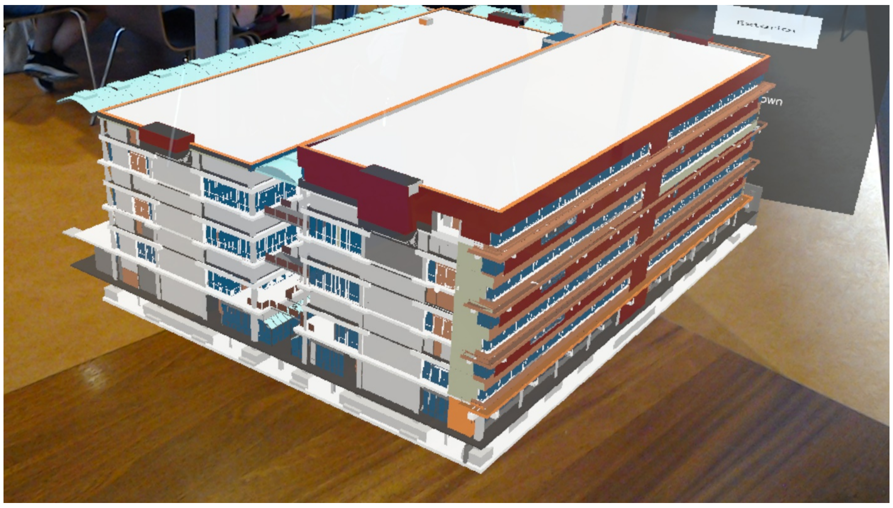


Figure 5.10: Exterior envelope of the AMC(2)

Generally, Hololens managed to visualize the complex model accurately. The users can walk around the model and have a clear perception of the exterior envelope of the building (Figure 5.10, Figure 5.13). The developed application provides an immersive experience since the model placed on a real surface, specifically on a table of TU Delft's library, and the users can watch the surrounding environment while they are observing the model (Figure 5.9, Figure 5.12). Finally, the hologram hosts even small and complex details of the building stating its visualization a decent substitute of a physical 3D model (Figure 5.11, Figure 5.12).

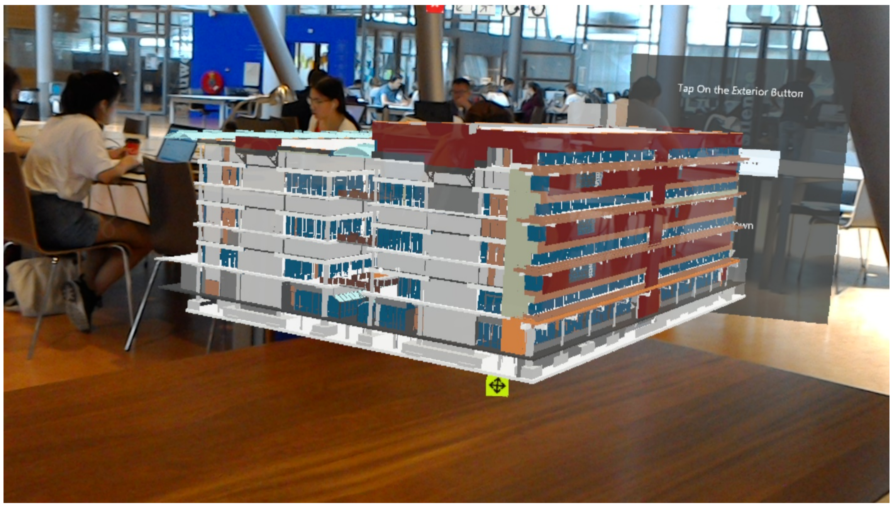


Figure 5.11: Exterior envelope of the AMC(3)

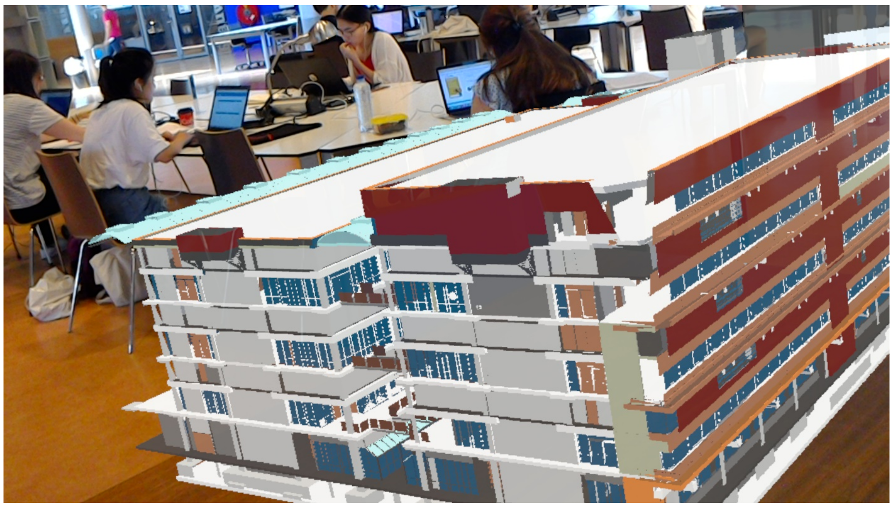


Figure 5.12: Exterior envelope of the AMC(4)

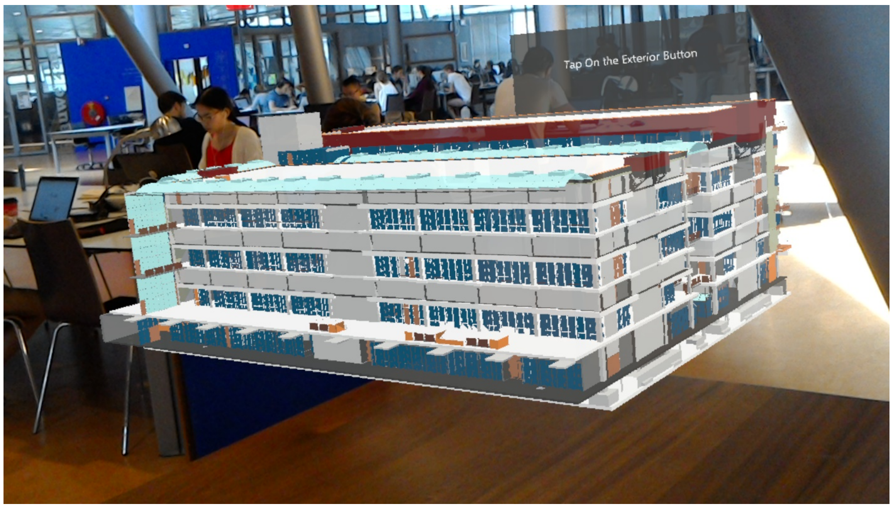


Figure 5.13: Exterior envelope of the AMC(5)

6 CONCLUSIONS

Chapter 6 starts with the formation of the conclusions in Section 6.1, possibilities for future directions are given in Section 6.2 and recommendations are suggested in Section 6.3.

6.1 CONCLUSIONS

6.1.1 General conclusions

IFCOPENSHELL `IfcOpenShell` is a robust library which highly enhances the creation and manipulation of IFC files. It offers many tools and options for various of possible cases. Familiarity with the `IfcOpenShell` library was demanded for the isolation of the storeys. There is an official website providing documentation about the classes, entities and structure of the library. The formation of the site is intricate and problems were faced with the understanding of the content. However, there are available tutorials which are a valuable aid since they provide guidance and sample code. There is also a community providing help to reasonable questions but it is small and not that active.

It generally returns accurate results and `.ifc` compliant outcomes but some deficiencies were identified. While trying to isolate some of the entities, they were returned written with lowercase letters instead of uppercase which caused compliance issues. Furthermore, the library failed to assign some of the values to certain entities, leading to files that could not be read by IFC viewers.

EXTERIOR EXTRACTION The extraction of the exterior shell was the most challenging task of this project. The suggested methods, that have been developed up to now, either produced models which can not serve as digital maquettes due to the lack of details or they made the assumption of topologically accurate building models which is not common among BIM models. The adoption of the ray-casting process was inspired by the computer graphics field and has not been implemented or tested up to now to extract the exterior part of a building.

The extracted model was accurate and descriptive enough to serve as a digital maquette. The inclination ray-casting highly enhanced the extraction of the triangles that were occluded by other geometries. The final outcome depends on the computational power that someone can devote on the ray-casting algorithm. The time which is needed for the execution of the process is affected by the number of cores that are used for the parallelization.

Some of the triangles that were "captured" during the ray-casting process, belonged to the interior part. This happened due to gaps on the original dataset. Nevertheless they were not many enough to hinder the performance of `Hololens`. While the original file was almost 128`MB`, the outcome of the exterior extraction was only 8.5`MB` which significantly facilitated the visualization of the model through the AR device.

UNITY `Unity` game engine was decided to be the environment in which the holographic application was build. This happened because of the cooperation of `Unity`'s enterprise with `Microsoft`, which provided many tools, functionalities and sample scripts that assisted the progress of the holographic development.

However some limitations were identified. One of them was the re-sizing of a model. Despite that the import of a model in the digital scene was an easy drag n drop process, its size transformation using the `Unity`'s rescaling boxes was not that straight forward. After having implemented a certain extent of rescaling the faces of the objects became darker and was not possible to change their colors. Apparently, `Unity`'s interface contains several "bugs". This shortcoming delayed and hindered the development process since the finding of the solution was part of the community's forum which means not easily identifiable.

Once the models had been imported in the scene, they had to be placed in a position that would be easily identifiable by the user. The placement of the models to their final position wasn't simple, since minor changes to their coordinates leaded to huge miss-placement. In the same way, positioning axis, which are an extra mean to change the position of a model, need configuration with regard to the extent that they transport the model on the corresponding axis. It wasn't always clear if one model was aligned with the other ones. Many times after having transported a model there was the necessity to change the view point of the camera in order to realize if the model was actually in the desired place. That was a distracting and time consuming process which demands improvement by `Unity`. Alignment axis or snapping tools would highly facilitate the process of placing the models accurately in the holographic scene.

6.1.2 Is semantic information enough to isolate building storeys from a BIM model?

The implementation of the suggested methodology was proved to be enough to isolate each storey of a BIM model only by using semantic information. The `python` programming language was combined with the open source `IfcOpehShell` library in order to aqcuire all the essential tools to achieve the semantic extraction. Among other functionalities, the library was used to load the IFC file and parse it into its structural entities, which highly accelerated the isolation process. The extraction of each storey was performed using the `IFCRELCONTAINEDINSPATIALSTRUCTURE` entities and was proved to be a simple and straight-forward process since only these entities had to be traversed. Each of these entities corresponding to a floor of the building. The ids of the objects that were part of a storey were stored there. By only obtaining an instance of each object, it was managed to isolate and write them in a new `.ifc` file. `IFCRELCONTAINEDINSPATIALSTRUCTURE` entity is connected with efficient functions which were provided accurate outcomes.

The needed time to complete the execution of the algorithm was considered acceptable. The process depends on the number of objects that a storey contains. The greater the number of objects is, the more the lines in the original file that have to be traversed, stored and written in a new IFC file are. Furthermore, the complexity of the geometry of the objects affects the demanded time for the execution of the script. Convoluted geometries demand more vertices and faces to describe their shape and this makes the isolation of them slower.

There is no demand of geometrical calculations for the extraction of the storeys. The proposed approach is affected by the way that the designers had constructed the original file. Since the identification of the objects that belong to a storey was performed using the `IFCRELCONTAINEDINSPATIALSTRUCTURE` entity, the assignment of an object to a wrong spatial structure will lead to inaccurate creation of building floors.

6.1.3 The determinants which affect the extraction of the exterior envelope?

For the completion of this master thesis, the raycasting concept was used. It can produce decent results depending on the factors stated below:

- **Points:** As points were considered the source from which the rays were sent and the target where they ended up. The number of points and their distribution are some of the key factors that affect the efficiency of the process. The number of points was determined empirically. A small number of points would capture only a few triangles leading to the creation of an inadequate model that would contain too many gaps and holes. On the other hand, a bounding box with too many points would implement many computational power constraints that might guide to an endless run of the algorithm.

  Except for the number of points, the distribution of them affects the final outcome. The placement of points in clusters, which means many points in one place leaving areas of having just few of them, and not equally distributed triggers problems as well. The raycasting from the points belonging in the same cluster will capture the same triangles, while many triangles of the outer shell will be left without being scanned.

  Not all the models are of the same extent, complexity or number of triangles in every side (Figure 3.4). The different number of points that were introduced on each face of the bounding box, enhanced the visual outcome of the raycasting process. The results of the unequal distribution was better and more promising compared to those derived from having the same amount of points for each side.

- **Rays:** Except for the number of points, the number and the inclination of the segments that will be introduced as rays influenced the final outcome. The more the rays are the greater the number of exterior triangles, that will be captured, is. However, that might set performance issues to the execution of the algorithm. As it was mentioned in Section 3.4.2, from one point many rays can be sent. The intersection of them with the triangles of the building is a time consuming and computationally expensive procedure which affects the needed time for the execution of the algorithm. As with the points, the specification of the number of the introduced segments is an experimental process.

  In the same way the specified inclination can contribute a huge difference to the final result. The majority of models presents some form of complexity in terms of having geometries that are obstructed by other objects. The inclination raycasting was proved to be an efficient way to overcome this problem. However, the percentage of inclination of a ray affects the number of triangles with which it will intersect and subsequently the needed time for the algorithm. The greatest the inclination of a ray is, the greatest the number of triangles with which it intersects will be.

- **Number of cores:** The computational power that someone possesses, significantly affects the final visual outcome and the demanded time to complete the execution of the algorithm. It is the factor which sets limits to the number of points and segments that will be introduced for the ray-casting process. This means that the greater the number of cores is, the greater the parallelization of the exterior extraction process could be. The developed algorithm designed in a way that does not demand the result of the intersection of a ray in order to proceed with the next ones, enhancing the parallelization across the cores. A smooth distribution of points across the cores of the machine essentially decreases the needed time to execute the process. The multiprocessing module, which was used during this project, is easy to be implemented and can highly improve the extraction of the exterior envelope.

- **Complexity of the model:** The shape of the building itself affects the accuracy of the model which will derive from the raycasting algorithm. The provided dataset is a good example as its extensive `U-shape` hindered the isolation of the outer shell. The recess, that lies on the middle of the building, hosts many objects of complicated geometry which are obstructed by other objects making the scanning of them a challenging action. The inclination raycasting provided way better results than the standard one, since it managed to capture many of the obstructed geometries, but not good enough to serve as a digital maquette. A process that split the model into half was performed. When as input to the algorithm was given only the half of the model, the final outcome was way more representative of the exterior part of the building.

6.1.4 How the user will interact with the hologram?

The communication of the user with the model is accomplished through buttons, which is a way to provide more than one possible interaction with a model. Gestures are the mean to interact with the holograms. They are the most common way to send "messages" to `Hololens`. Users easily adapted themselves to this mean of interaction, despite they are only used to traditional ways, mice and keyboards.

They have the ability to grow, shrink, rotate, visualize metadata, replace and hide or visualize holograms. These functions were considered relevant for the creation of an meaningful involvement with the hologram. Spaces, like buildings under construction, in which the holograms are supposed to be visualized, might set constraints with regard to the freely motion of people.

Grow, shrink and rotate help with the better observation of the model without the necessity for the users to come closer or walk around the holograms. Visualization of metadata provides a deeper understanding of the objects that consist a model. Metadata could be even used during the construction phase to inform workers and supervisors about the dimensions and types of objects that have to be used in certain parts of the building. The replacement of hologram's original position provides the opportunity to put the model on horizontal surfaces that have been identified during the scanning phase providing a more immersive experience. Finally, the hiding and visualization of the desired model enhances the performance of the device. Except for, it is not efficient for `Hololens` to visualize more than one model, the simultaneous projection of many models obstructs the user's attention from the important model.

6.1.5 What are the limitations of `Hololens` with regard to the visualization of a building model?

`Hololens` is considered as the best commercial available AR device. However it is just the first model of its generation, so it brings some software and hardware limitations with it.

- **Field of view:** Field of view is the field of real space in which it can visualize digital generated information. It is quite narrow, introducing problems to the users when they do not look straight forward to the created holograms. Many times the holograms or part of them were disappeared because of a slight motion of the user's head, despite their place was still observable (Figure 6.1). This limitation derives from the restricted screen technology that `Hololens` supports. A user has to directly look at the hologram in order to watch it be overlaid on the physical environment.

- **Battery life:** `Hololens` has issues with the time that a holographic application can run on it. Draining of the battery is intense while the application performs. Furthermore, the more complex the geometry of the model is, the greater the consumption of energy is. That sets a big constraint to one of the greatest advantages of Hololens which is its portability. The visualization of a future building to various stakeholders in the construction field, where there might be no electricity during the initial stages, would be highly restrained due to the limited autonomy.

- **Comfort:** The way that the Hololens fits in someone's head could be bothersome for some people. It is quite heavy which might annoy the users while they are gazing at holograms. Furthermore, there were issues with the users wearing glasses, since it was more difficult for them to adapt the helmet to their skull. It has to be stated that if someone adjusts it properly, it is well adapted to his/her head and that significantly enhances the user's experience.

- **Performance:** Considering the optical performance of the AR device it provides an immersive and realistic experience to the user. The rendering performance is affected more by the complexity of the model than by its size. When Hololens visualizes intricate geometries, there might be some phenomena like high frequency shaking and move or "jump" of the model away from where it was originally placed. Furthermore, while it was trying to visualize the the exterior envelope, loaded with the onFocus and metadata scripts, presented a high decrease on the number of FPS that could project. In the same way, the materials that were assigned to the created holograms influenced the efficiency of the visualization. The projection of shades and reflections, connected with the assigned materials, is a computationally expensive process for it. I recommend that AR developers use as textures of their materials the ones that are included in HoloToolkit package, since they were created to perform the best with holographic applications. Finally, when more than one models are visualized in parallel then the performance of the application significantly decreases since the HPU has to process large volume of data simultaneously.

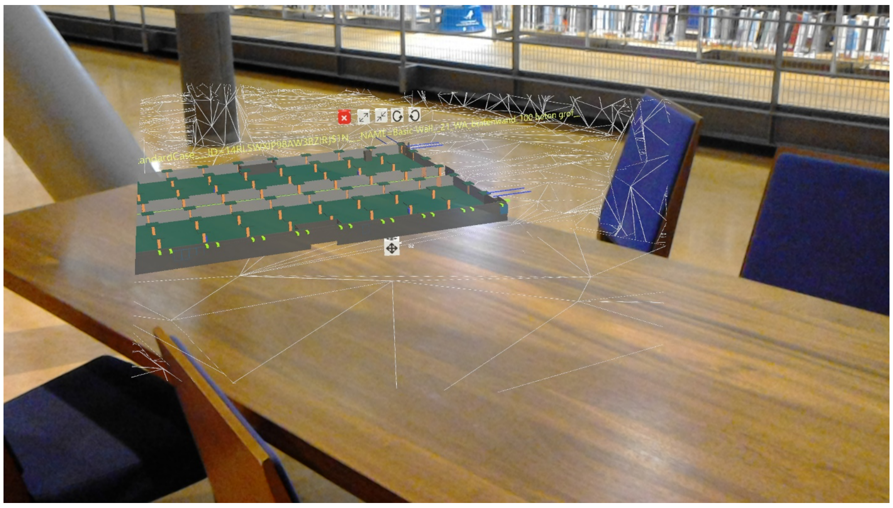


Figure 6.1: Limited field of view of Hololens

6.2 FUTURE WORK

This section provides some future directions considering the interaction with the user and the optimization of the multiprocessing procedure. The followed suggestions were not investigated in order to invest the limited time resource to the development of a comprehensive report.

6.2.1 Voice Commands

As earlier mentioned in Holograms Manipulation, the only way with which a user can interact with a model is by performing gestures either to one of the created buttons or to the gazed building object. Another possible way of interaction would be voice commands. Instead of making a gesture, a user could command the hologram to perform an action. A developer can specify some "key" words that will be recognizable by the `Hololens` and will apply some actions to the hologram which is being gazed at this moment. Once it identifies a "key" word, it sends the corresponding message to the hologram. A user could say `''grow''` to grow a model, and `''left''` or `''right''` to rotate a hologram to the left or right. The use of voice orders it may lead to an even more intuitive interaction with the hologram. A problem that has been identified is that `Hololens` does not perform well with "strong" accents.

6.2.2 PyCUDA

Section 4.4 described the extraction of the exterior envelope by implementing a raycasting algorithm. It mentioned that in order to accelerate the extraction of the exterior envelope, more than one cores of the Central Processing Unit (CPU) can be utilized. However, the number of iterations could be thousands, hundreds of thousands or even millions, depending on the complexity of the model and the number of introduced points and segments. A potential future change that could decrease the needed time for the execution of the algorithm, would be the usage of GPU's cores instead of the ones of CPU.

`PyCUDA` is the module which supports the usage of the cores of the GPU and it can be utilized by writing scripts in `python` programming language. GPU possesses way more cores than CPU but they are not easily programmable. So, the possibility to use them has to be checked. The potential usage of them could highly decrease the amount of time which is needed to execute the raycasting process even for complicated models of many thousands triangles and points.

6.3 RECOMMENDATIONS

This chapter includes some suggestions considering the improvement of `Hololens` and the adoption of a property that would facilitate the extraction of the exterior envelope. All of them derived from the experience obtained during the graduation project.

6.3.1 Hololens

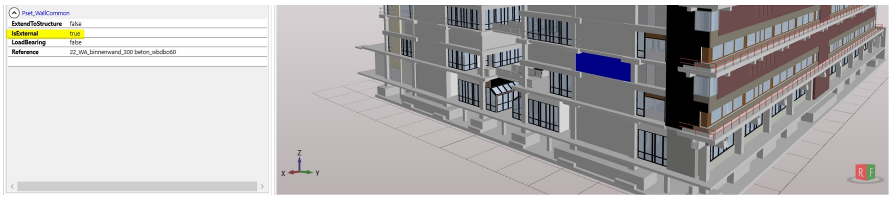
One of the greatest limitations of the `Hololens` is its limited field of view. That highly constrains the user's experience and decreases the amount of information which is projected to him/her. Since `Microsoft` intents to release a new version of `Hololens`, the supported holographic displays have to be improved. That will enhance the success of the applications targeting to `Hololens` and will provide a more immersive and realistic sense to the users.

In the same way, in order for the device to be more useful especially for AEC applications, it is highly recommended to enhance the autonomy of the battery. Usually many people are involved in the creation and maintenance of a building and all of them have to observe the models. This demands a reliable device considering the time that it can perform. The next generations of `Hololens` have to be coupled with a more advanced battery and more efficient hardware. It will contribute to the commercial adoption of `Hololens` by many companies doing outdoor projects.

Another shortcoming of `Hololens` is its weight and the way that it fits to the user's head. Depending on the application, a user might have to wear the helmet for a long period of time. Due to its weight, users were forced to remove it from their head in order to feel more comfortable and put it back again. Many of them after wearing it for a couple of minutes they got a red spot on their nose, because of the pressure that was applied to it. In addition to this, its adjustment to the user's head wasn't always successful which leaded to misplacement and forced them to re-adjusted it. Concluding, a lighter and more comfortable `Hololens` would enhance the user's experience.

6.3.2 IFC

The `IFC2x3` schema supports for two of its entities, `IfcWall` and `IfcWallStandardCase`, to set a reference to their properties in order to make the distinction between internal and external geometries. The name of the reference is `IsExternal`, its property type is `IfcPropertySingleValue` and its data type is `IfcBoolean`. This `Boolean` variable can take the value `TRUE` if this entity is an external object and faces towards the outside of the building and the value `FALSE` if it is part of the interior (Figure 6.2).

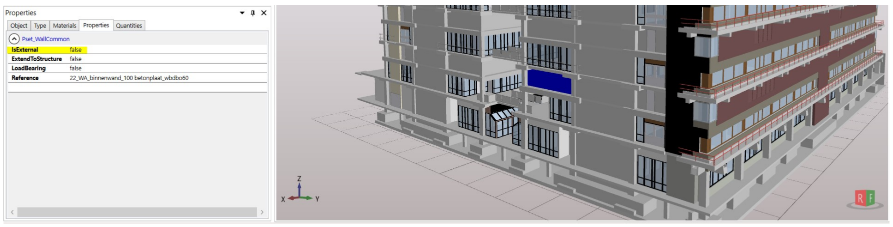


Pset_WallCommon

ExtendToStructure	false
IsExternal	true
LoadBearing	false
Reference	22_WA_binnenwand_300 beton_wbdbo60

Figure 6.2: Correct assignment of isExternal property

It is recommendable to the creators of the next versions of IFC, to set this reference to every geometrical entity that the schema supports, such as `IfcWindow` or `IfcDoor`. By doing that, the extraction of the exterior shell will be a straight-forward procedure based on semantics. Since the semantical isolation is a faster process than the geometrical extraction, the whole procedure will be implemented way quicker. The `TRUE` value has to be assigned to objects that have at least one face as part of the exterior envelope. So the designers of the model have to be aware of the importance that this attribute would cater with it (Figure 6.3).



Pset_WallCommon

IsExternal	false
ExtendToStructure	false
LoadBearing	false
Reference	22_WA_binnenwand_100 betonplaat_wbdbo60

Figure 6.3: Incorrect assignment of isExternal property

BIBLIOGRAPHY

Agarwal, S. (2016). Review on Application of Augmented Reality in Civil Engineering. page 4.

AGC (2006). Associated General Contractors.

Azhar, S., Khalfan, M., and Maqsood, T. (2015). Building information modelling (BIM): now and beyond | Azhar | Construction Economics and Building.

Barber, P., Graves, A., Hall, M., Sheath, D., and Tomkins, C. (2000). Quality failure costs in civil engineering projects. *International Journal of Quality & Reliability Management*, 17:479–492.

Beetz, J. (2012). Gebruik van 3d gebouw-modellen in 3d gis: Building information modelling (bim/ifc) voor beginners (in dutch).

Benner, J., Geiger, A., and Leinemann, K. (2005). Flexible generation of semantic 3d building models. In *Proceedings of the 1st international workshop on next generation 3D city models, Bonn*, pages 17–22.

Billinghurst, M. and Kato, H. (2002). Collaborative Augmented Reality. *Commun. ACM*, 45(7):64–70.

Cardoso, A., Peres, I. C. d. S., Lamounier, E., Lima, G., Miranda, M., and Moraes, I. (2017). Associating Holography Techniques with BIM Practices for Electrical Substation Design. In *Advances in Human Factors in Energy: Oil, Gas, Nuclear and Electric Power Industries*, Advances in Intelligent Systems and Computing, pages 37–47. Springer, Cham.

Caudell, T. P. and Mizell, D. W. (1992). Augmented reality: an application of heads-up display technology to manual manufacturing processes. In *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, volume ii, pages 659–669 vol.2.

Chu, M., Matthews, J., and Love, P. E. D. (2018). Integrating mobile Building Information Modelling and Augmented Reality systems: An experimental study. *Automation in Construction*, 85(Supplement C):305–316.

Diakité, A. A. (2017). 201703330 Booosting 3d IMRO - Abdoulaye Diakite TU Delft.

Diakité, A. A., Damiand, G., and Maercke, D. V. (2014). Topological Reconstruction of Complex 3d Buildings and Automatic Extraction of Levels of Detail. pages 25–30. Eurographics Association.

Donkers, S., Ledoux, H., Zhao, J., and Stoter, J. (2016). Automatic conversion of IFC datasets to geometrically and semantically correct CityGML LOD3 buildings: Automatic conversion of IFC datasets to CityGML LOD3 buildings. *Transactions in GIS*, 20(4):547–569.

El-Mekawy, M. and Östman, A. (2010). SEMANTIC MAPPING: AN ONTOLOGY ENGINEERING METHOD FOR INTEGRATING BUILDING MODELS IN IFC AND CITYGML. page 11.

Feiner, S., MacIntyre, B., Hollerer, T., and Webster, A. (1997). A touring machine: prototyping 3d mobile augmented reality systems for exploring the urban environment. In *Digest of Papers. First International Symposium on Wearable Computers*, pages 74–81.

Fonnet, A., Alves, N., Sousa, N., Guevara, M., and Magalhães, L. (2017). Heritage BIM integration with mixed reality for building preventive maintenance. In *2017 24 #186; Encontro Portugu #234;s de Computa #231; #227;o Gr #225;fica e Intera #231; #227;o (EPCGI)*, pages 1–7.

Ishii, H. and Ullmer, B. (1997). Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, pages 234–241, New York, NY, USA. ACM.

Kopsida, M. and Brilakis, I. (2016). *Markerless BIM Registration for Mobile Augmented Reality Based Inspection*.

Kress, B. and Cummings, W. (2017). 11-1: Invited Paper: Towards the Ultimate Mixed Reality Experience: HoloLens Display Architecture Choices - Kress - 2017 - SID Symposium Digest of Technical Papers - Wiley Online Library.

Loomis, J. M., Golledge, R. G., Klatzky, R. L., Speigle, J. M., and Tietz, J. (1994). Personal guidance system for the visually impaired. pages 85–91. ACM Press.

Love, P. and Edwards, D. (2004). Determinants of rework in building construction projects. *Engineering, Construction and Architectural Management*, 11:259–274.

Love, P. E. D. and Edwards, D. J. (2013). Curbing rework in offshore projects: systemic classification of risks with dialogue and narratives. *Structure and Infrastructure Engineering*, 9(11):1118–1135.

Love Peter E. D. (2002). Influence of Project Type and Procurement Method on Rework Costs in Building Construction Projects. *Journal of Construction Engineering and Management*, 128(1):18–29.

McHenry, K. and Bajcsy, P. (2018). An Overview of 3d Data Content, File Formats and Viewers. *Technical Report*, page 21.

Muensterer, O. J., Lacher, M., Zoeller, C., Bronstein, M., and Kübler, J. (2014). Google Glass in pediatric surgery: An exploratory study. *International Journal of Surgery*, 12(4):281–289.

Nagel, C., Stadler, A., and Kolbe, T. (2007). Conversion of IFC to CityGML. In *Meeting of the OGC 3DIM Working Group at OGC TC/PC Meeting, Paris (Frankreich)*.

NBIMS (2015). Welcome to the National BIM Standard - United States | National BIM Standard - United States.

Xie, J. (2012). Research on key technologies base Unity3d game engine. In *2012 7th International Conference on Computer Science Education (ICCSE)*, pages 695–699.

Xu, X. G., Taranenko, V., Zhang, J., and Shi, C. (2007). A boundary-representation method for designing whole-body radiation dosimetry models: pregnant females at the ends of three gestational periods—RPI-P3, -P6 and -P9. *Physics in Medicine & Biology*, 52(23):7023.

Zhou, F., Duh, H. B.-L., and Billinghurst, M. (2008). Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 193–202, Washington, DC, USA. IEEE Computer Society.

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.