U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Virtual reality in the architecture, engineering and construction industry

proposal of an interactive collaboration application

THIAGO VILELA CRUZ

Dissertação submetida para satisfação parcial dos requisitos do grau de

MESTRE EM ENGENHARIA CIVIL — ESPECIALIZAÇÃO EM CONSTRUÇÕES CIVIS

Orientador: Professor Doutor João Pedro da Silva Poças Martins

Coorientador: Engenheiro Aymeric Del Nibbio

Janeiro 2018

MESTRADO INTEGRADO EM ENGENHARIA CIVIL 2017/2018

DEPARTAMENTO DE ENGENHARIA CIVIL

Tel. +351-22-508 1901

Fax +351-22-508 1446

✉ miec@fe.up.pt

Editado por

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rua Dr. Roberto Frias

4200-465 PORTO

Portugal

Tel. +351-22-508 1400

Fax +351-22-508 1440

✉ feup@fe.up.pt

🌍 http://www.fe.up.pt

Aos meus pais

Eu não tenho idolos. Tenho admiração por trabalho, dedicação e competência

Ayrton Senna

ACKNOWLEDGEMENTS

Aos meus pais pelo suporte incondicional em todos os momentos da minha vida, sempre me aconselhando e estando sempre presente, que sem vocês eu nunca teria a oportunidade de realizar os meus sonhos que já conquistei e os desafios futuros que ainda estão por vir.

A meu irmão e irmã, Luiz Fernando e Ana Flavia, e também aos meus primos, Gabriel e Igor, no qual para mim são como irmãos, por todos os momentos que vivemos juntos e maravilhosas recordações que guardo em minha memória da nossa infância.

A todos os meus amigos de infância de minha cidade natal de Juiz de Fora, em especial ao grupo de amigos *Galerê*, por todo o apoio dado a mim ao longo dos anos e por mesmo apesar da distância, ainda estarmos sempre unidos.

A todos meus amigos de curso da Universidade Federal de Juiz de Fora, local no qual comecei a trilhar o meu percurso acadêmico, pelas memorias felizes dos churrascos e festas, as conversas ao lado da beira do trilho do trem na cantina e muitos outros momentos inesquecíveis.

Aos meus amigos e professores da Universidade do Porto, pela excelente hospitalidade e que me permitiram aprender muito e crescer tanto academicamente quanto pessoalmente, especialmente por terem feito eu me sentir em casa mesmo estando longe do meu país de origem.

Ao meu orientador João Poças Martins pelo voto de confiança e pelas palavras de incentivo para a realização de um bom trabalho.

A Carlos Meira e Ruben Monteiro pela confiança depositada no meu trabalho e por ter me permitido conhecer o mundo Bouygues. A André Meira, Clement Duc-Dodon, Jodie Decoopman, Sarah Dutaiza, Aymeric Del Nibbio e todos outros colegas da Bouygues Bâtiment Île-de-France Habitat Social, que transformaram o desafio de estagiar em uma empresa no estrangeiro extremamente agradável.

RESUMO

O processo de *Building Information Modeling* (BIM) representa uma evolução a métodos tradicionais de gerenciamento e desenvolvimento para projetos de construção. No entanto esta tecnologia mesmo tendo tido um recente crescimento considerável em sua adoção na indústria de arquitetura, engenharia e construção(AEC), ela ainda apresenta barreiras e limitações na sua utilização de forma generalizada, devido a sua complexidade e outros fatores envolvidos.

A realidade virtual é uma tecnologia que apresentou, nos últimos anos, um considerável desenvolvimento relativamente ao *hardware* e que tem tornado os óculos de realidade virtual por preços mais acessíveis e consequentemente tornando-a mais difundida. A capacidade de imersão associada a interatividade das aplicações de realidade virtual apresenta diversas potencialidades para a indústria da construção devido ao usuário ter a capacidade de visualização em escala real, desta forma compreendendo melhor os modelos digitais de edifícios, antes mesmo de serem construídos.

Com isso recorreu-se a um motor de jogo para o desenvolvimento de uma aplicação de colaboração interativa, no qual permite os diferentes participantes do projeto de construção interagirem e avaliarem o design de um projeto utilizando realidade virtual, no qual não é necessário ter nenhum tipo de formação técnica (a interface do usuário é intuitiva) e em que o fluxo de informação é feito de forma automática do modelo de realidade virtual para o modelo digital BIM.

PALAVRAS-CHAVE: Realidade Virtual, indústria AEC, BIM, motor de jogo, revisão de projeto

ABSTRACT

The process of Building Information Modeling (BIM) represents an evolution compared to traditional methods of management and development to construction projects. However, this technology even though it had a recent significant increase in its adoption at the architecture, engineering, and construction (AEC) industry, it still presents barriers and limitations in its extensive usage, due to its complexity and other factors involved.

Virtual Reality(VR) is a technology that presented, in recent years, a considerable development towards its hardware that allowed the head-mounted-displays(HMD) with more affordable prices and consequently making it more widespread. The capacity of immersion associated with the interactivity of the VR applications presents various potentials towards the AEC industry due to the capability of the user to visualize in one-to-one scale. Thus, better understanding the digital models before they are even built.

As such, it was used a game engine to develop an interactive collaboration application, in which the various projects stakeholders can interact and evaluate the project design using virtual reality, in which it is not necessary any type of technical formation (the user interface it is intuitive) and that the flux of information it is done automatically from the VR model to the BIM model.

KEYWORDS: Virtual Reality, AEC industry, BIM, game engine, design review

GENERAL INDEX

FIGURES INDEX

TABLE INDEX

SYMBOLS AND ACRONYMS

3D – Three Dimensional

AEC – Architecture, Engineering and Construction

API – Application Programming Interface

AR – Augmented Reality

BIM – Building Information Modeling

CAD – Computer Aided Design

CAVE – Cave Automatic Virtual Environment

FBX -- Filmbox

FPS – Frame Rate per Second

GUI – Graphical User Interface

HMD – Head Mounted Display

LOD – Level of Development

OBJ – Object Files

SDK – Software Development Toolkit

VPL – Visual Programming Language

VR – Virtual Reality

1

INTRODUCTION

1.1. FRAMEWORK

The architecture, engineering, and construction (AEC) industry have always involved the necessity of exchanging information between different project stakeholders. Traditionally, this flux of information was either made through a piece of paper or over oral communication, which can be insufficient considering the complexity of contemporary construction projects. Weak communication between the project stakeholders can compromise the productivity of a construction project [1].

The building information modeling (BIM) is a process of generating, creation and management of information through digital models that are intended to support the decision-making at all phases of the construction process. This technology is currently having an increasingly broad adoption in the industry, however, the BIM software in the current market offers a limited alternative to visualization on a one-to-one scale.

Virtual Reality (VR) this medium has seen a considerable development concerning its hardware in recent years that it is currently putting this technology in a ‘spotlight' and raising a considerable amount of investments. By putting the users on a human scale, the users have a natural perspective as being ‘present' at the virtual model, and this allows the detection of potential design defects concerning ergonomics, constructability, and aesthetics [2].

1.2. MOTIVATION AND OBJECTIVES

One crucial element of the project productivity it is that everyone involved in the project such as client, architects, consultants, and engineers are able to communicate effectively, VR can be used as a potential tool to supply this demand as it is going to be discussed further during this thesis.

Currently, BIM software such as Revit are not optimized enough to be able to handle real-time image processing with sufficient frame rate per seconds (FPS) required for most VR devices for an acceptable experience. Mainly due to the high complex geometry of the elements and its high complicated interactions between themselves.

It is intended in the present thesis to discuss the applicability and its potentials of VR towards the AEC industry, and develop an interactive collaboration VR tool using a game engine, that allows different project stakeholders to give feedback on a construction project in earlier phases, detecting potential design defects and potentially avoiding the cost of reconstruction.

1.3. STRUCTURE

This thesis was divided into 6 different chapters which contain the following content

- Chapter 1 – Introductory, it establishes the framework and motivation, defines the thesis proposal and presents its structure.
- Chapter 2 – Contemplates the literature review regarding the topic of virtual reality, introducing its essential elements, its history and discussing the hardware and software requirements for a successful experience.
- Chapter 3 – Similar to chapter 2, this chapter corresponds of a literature review focused at the VR applications in the AEC industry, discussing its applicability and its potentiality while presenting examples of their current stage at the industry.
- Chapter 4 – The proposed interactive collaboration application it is formulated with discussions reflecting the particularities of the VR medium in the matter of user interface and interactivity.
- Chapter 5 – This chapter consists of a case study of one residential apartment. A brief explanation of the construction project it is given and the workflow of achieving the final application it is deliberated.
- Chapter 6 – In this final chapter the main conclusions of this work are displayed together with the main difficulties found in this thesis, and it is made indications towards future research.

2

VIRTUAL REALITY – THE IMMERSIVE MEDIUM

2.1. DEFINITION OF VIRTUAL REALITY

While applicability of VR evolved throughout the years, its concept has been constantly changing. Using a simple definition from the *Merriam-Webster's* [3] it is possible to find the definition of the word *reality* as *"something that is neither derivative nor dependent but exists"*. In a similar fashion, if we use the same dictionary for the word *virtual* [4], one of its meaning is *"being on or simulated on a computer or computer network"*. However, by combining both words the meaning doesn't have a narrow definition, leaving it the possibility of misleading uses of its term.

Virtual reality is a medium that, just like music or any other forms of media, can be used for many purposes. The primary purpose for any medium is the communication of ideas, and these ones in VR can range from purely abstract (what is it like to live inside an animated cartoon?) to the purely practical (will this design work correctly after we build it).

A VR experience has four key elements which are going to be discussed on the following sub-chapters: the virtual world, sensory feedback, immersion, and interactivity. Combining the etymology with these elements, it is possible to have a more precise definition.

"A medium composed of interactive computer simulations that sense the participant's position and actions and replace or augment the feedback to one or more senses, giving the feeling of being mentally immersed or present in the simulation (a virtual world) [5]*."*

2.1.1. THE VIRTUAL WORLD

The virtual world corresponds to all the content of a computer-generated simulation. These virtual environments can diversify from being extremely simple to complex worlds according to the application objective. The virtual world contains laws of nature(rules) that can simulate reality, or it can be a fantasy world. These rules are how the content behave according to the developers. A few examples of these are gravity, locomotion, topography, communication and real-time actions.

In the AEC industry context, it is desirable to have the virtual worlds displaying the geometry according to the reality, or a minimized representation of it. One common practice that it is used with virtual worlds in the architectural visualization it is to create ultra-realistic scenery as it can be seen in the figure 2.1. However, as we are going to discuss further in section 2.3 about the

performance of a VR hardware, it is important to not overwhelm the hardware with lots of complex geometries otherwise the frame-rate might be downgraded and can compromise the VR experience.

Fig 2.1 - Image of a detailed realistic architectural visualization, one complex virtual world [6].

2.1.2. SENSORY FEEDBACK

The sensory feedback is a key element of making the VR experience more compelling to the participant. Unlike other types of media, the VR systems give the capability to the user to affect events in the virtual world in a unique way.

These experiences provide synthetic stimuli to one or more of the participants senses. Typically, at least the visual sense is stimulated, with the aural sense also frequently stimulated. Less common are the systems that are able to give skin-sensation and force feedback, which are referred to as the haptic sense. Even more unusual are the stimuli to other senses such as the vestibular(balance), olfaction(smell) and gustation(flavor)[2].

2.1.3. IMMERSION

In most of the types of media, the interpretation of the state of "being immersed "usually is related to the emotional state of being involved in the experience. When it is referring to the medium of VR, however, it also integrates the capability of the VR system to augment or replace the stimulus of the user's senses. Due to that, it is possible to point out two main types of immersion: mental immersion and physical immersion.

Achieving mental immersion, it is the goal of most media creators. For example, a novel that has a very interesting storyline can pull the reader into the story, creating a suspension of disbelief. Furthermore, in most of the typical media, the storyline is only presented to the viewer from a third person point of view, making it more dependable on the writer's intelligence to achieve this type of immersion.

The experience with VR on the other hand, while having the participant having some or even all of his senses stimulated, it facilitates the creation of a partial or a complete suspension of disbelief. The degree of the virtual world faithfully reproducing reality consistently determines the degree of suspension of disbelief[5].

Immersion can be related as well to the subjective sensation of being 'present' in the scene reproduced by a medium. This is a sensation if compared to other types of medium, in which VR is the closest one to achieve. Participants regularly feel as they have been 'transported' to a different place. Since it is a subjective perception, it depends on personal human factors to be able to believe in this state of 'presence'[7].

2.1.4. INTERACTIVITY

Considering that immersion is a response to a possible static form of representation, the interactivity element requires a dynamic environment. It is not simply about the capacity of the participant to navigate through the environment, but most importantly the virtual environment reacting to the user's actions. Using the sensors of the VR system to capture the users positioning and giving them, the freedom of movement could be a satisfactory experience, but when those actions do not result in any reactions in the virtual world, it limits the degree of the user immersion.

Using a metaphor to describe interactivity is to compare to a theatrical performance. The VR users are like the theatrical audiences, not only are able to watch the play but can also join the stage to participate, become various characters, changing the action by what they say and do in their roles[8].

The degree of interactivity is affected by various factors. Steuer [9] suggested that interaction depends mainly on three factors: speed, range, and mapping.

Speed is defined by the rate that a user's actions are incorporated into the virtual world. Faster response relates to more actions and changes in the virtual world, this velocity depends directly on the capacity of the VR system. Range is referred as the number of possible outcomes for any user action. This factor can be compared to a set of tools, in which the more tools you get, the more workable the environment is. Finally, mapping refers to the capacity of the system to map from user actions to changes in the environment in a predictable manner.

It is relevant to the user that he can foresee the results of his actions to some extent, otherwise his actions are only pure movements and not intent-driven actions. The interactivity in the majority of VR applications is desirable to be as most intuitive and practical as possible, adapted to the objective of the application itself.

2.2. HISTORY OF VIRTUAL REALITY

Assuming simplistically the concept of VR as a simulation of an environment that allows the participant to experience a place or event differently where he physically is, then it is reasonable to consider flight simulator as early examples of VR systems. In the early 1970s, interactive computer displays were developed to be able to train pilots, encouraged by the high cost of training. These computer displays were mechanically linked to the pilot's controls such as the yoke, rudder pedals, throttle controls, etc. The objective of these simulators was to develop familiarity of the pilot to the aircraft controls.

Attempts to try to immerse a person in a different place has been made since the middle of the 19th century with Charles Wheatstone's stereoscope. However, it was in the 20th century that the concept and technology of VR were created and developed, and until the year of this thesis, it is developing in a rapid progression, due to the increased market interest.

During the advance of VR, it is noticeable that often the content was driven by technology advancements, and at other times, it was the technology that drove the content advancements. The technology improvements were encouraged often throughout its history within the objective to be used in the entertainment industry such as in video-games.

Even though the technology has the potential to be applied in many industries with different types of applications, the entertainment industry was the main one that driven technological advancements. Since its focus it is to the mass market, this particular competitive industry concerns a lot about the prices of the hardware, being pressured to always lower its prices, benefiting other industries such as the AEC. Following are some of the most relevant milestones of conceptual and technological development.

1838 – The first type of stereoscope, Charles Wheatstone's work had two mirrors paired in a 45-degree angle to the user's eyes, with each mirror reflecting a picture located offsite. This experiment showed that when two pictures of the same location, simulating left and right eye are presented in a way that each eye sees only the representation designed for it, the brain subconsciously fuses the images and accepts it as a 3D object. This was the first attempt to give a sensation of depth and immersion.

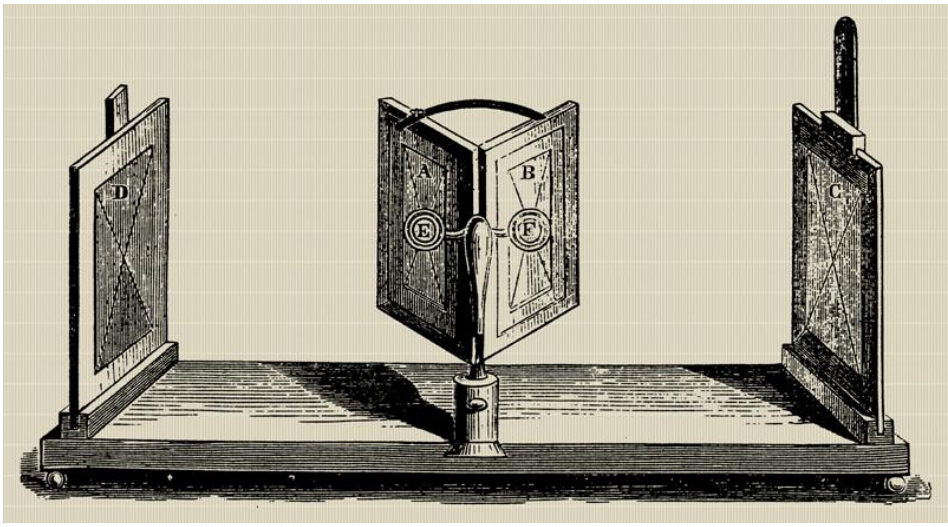


Fig 2.2 – Image of the Charles Wheatstone stereoscope [10].

1950s - Sensorama – One of the firsts sensory displays, the Sensorama created by Morton Heilig, was a scripted arcade-like experience. The participant was seated in front of a display screen which contained a variety of sensory stimulators. The stimulator displays included wind, smell, sound, and vibration. The intention of the experiences was to fully immerse the participant in the movie. One of these scenarios was a motorcycle ride with contained stimulators that were triggered at the appropriate time. For example, when the motorcycle was near to a bus, the rider was exposed to a whiff of exhaust. This system, however, was lacking one major key element of the VR systems, which was the response to the user's actions.

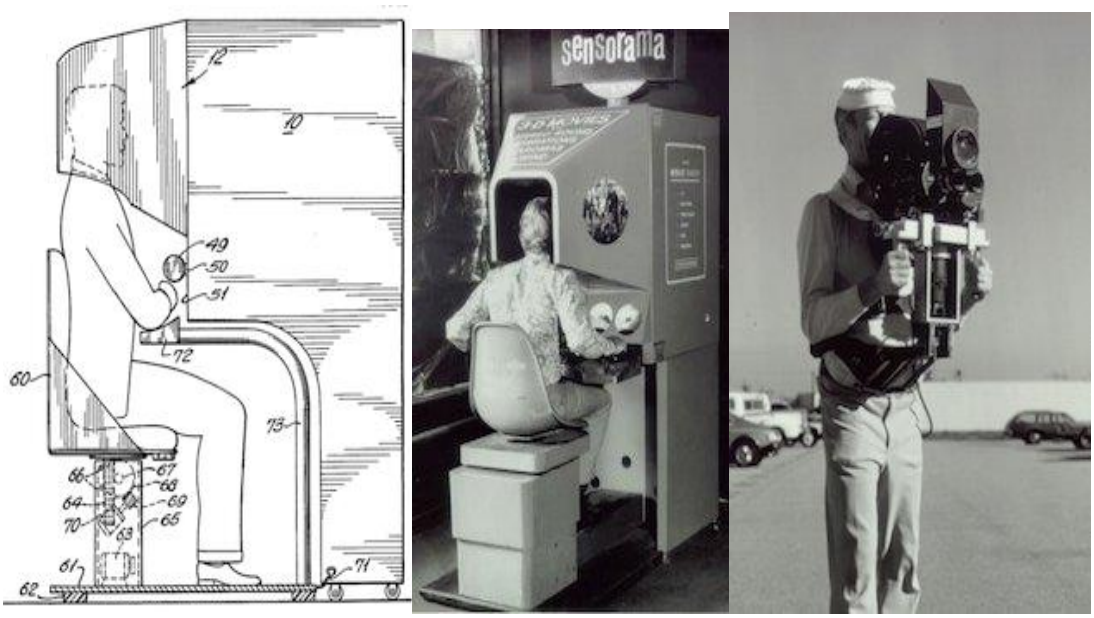


Fig 2.3 – Images of the arcade device Sensorama[11].

1960 – Telesphere Mask - It was also Morton Heilig that created the first prototype of an HMD which provided a stereoscopic image with stereo sound, the Telesphere Mask was patented in 1960. The device ,however, didn't have any type of motion tracking or interactive response to the participants' actions.

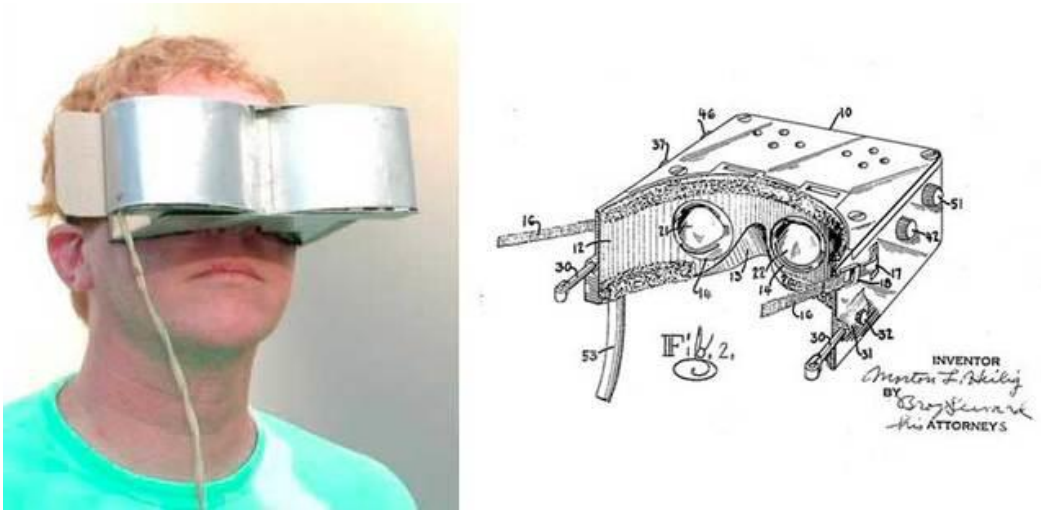


Fig 2.4 – Telesphere Mask, the first head-mounted stereoscopic display [11].

1965 - The "ultimate display" by Ivan Sutherland – The concept created by Sutherland and presented in the International Federation for Information Processing (IFIP), was to simulate reality to a point in which the user cannot tell the difference from the virtual and the physical reality. It included a virtual world viewed through an HMD and a realistic augmented 3D environment with tactile feedback, where the users could interact with the objects in the virtual world in a realistic way. That was the first time when the concept of how VR is understood today was presented. A few years later, in 1968, Sutherland demonstrated for the first time his prototype

called "Sword of Damocles" which was able to accomplish his vision concerning the visual component. It was a heavy contraption hanged from the ceiling connected to a computer, which allowed the user to move his head and navigate with very primitive computer-generated images.

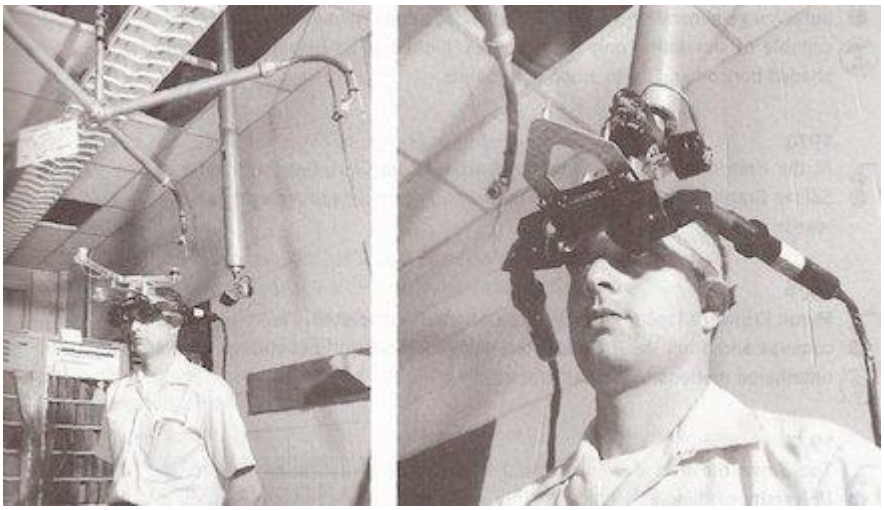


Fig 2.5 – "Sword of Damocles" the first VR headset [12].

1977 – Sayre Gloves – It was developed at the Electronic Visualization Lab at the University of Illinois at Chicago. It consisted of a glove with light-conductive tubes transmitting various amounts of light proportional to the finger bending. The information is interpreted by a computer to estimate the configuration of the user's hand.

1984 – VPL Research Inc was founded with the objective of creating a visual programming language. However, soon after its creation, it changed the initial objective of the company to participate under grants of NASA VIEW lab to create the DataGlove in 1985 and EyePhones in 1989. The first one was an instrumented glove that similar to the Sayre gloves, reported the wearer's hand position to the computer. The Eyephones consisted of an HMD that had a pair of LCD displays in conjunction with LEEP optics. These two products in conjunction with a computer system responsible for real-time rendering were released in 1989 as a Virtual Reality System – RB2. It was the first time that the term 'virtual reality' was introduced to the market and it popularized.

Fig 2.6 – Image of the Virtual Reality System – RB2 developed for NASA[13].

1986 – "Super Cockpit" – With the intention of military training, Thomas Furness developed an incredibly ahead of its time flight-simulator project. The training cockpit was able to project computer-generated 3D maps, infrared and radar images, including as well real-time 3D information of the simulated airplane. The project allowed the trainee to control the aircraft using gestures, speech, and eye movements.

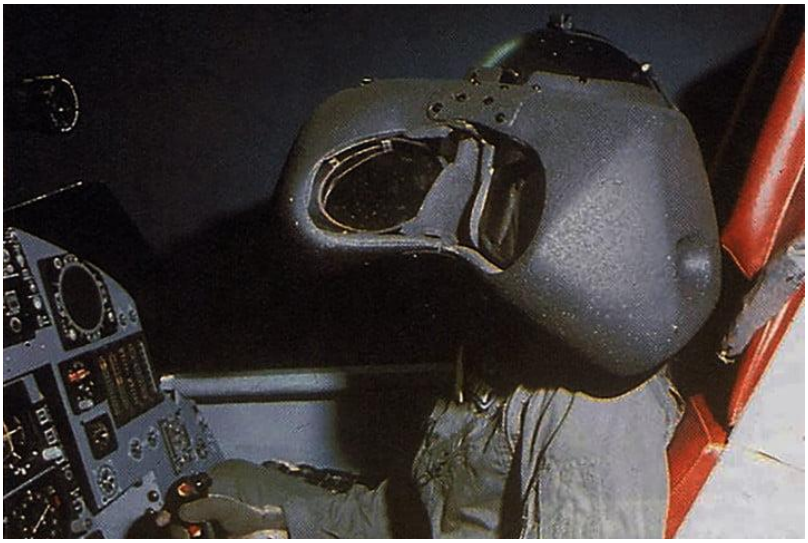


Fig 2.7 – Image of the Super Cockpit flight simulator [14].

1992 – The main attraction of the SIGGRAPH convention of that year was the CAVE, it was developed by the Electronic Visualization Lab at the University of Illinois, it consisted of a projected VR environment allowing the users to immerse into interactive scientific and artistic applications. It was an alternative to experience VR since most of the experiences were being developed on HMDs. The CAVE consisted of a walk-in virtual reality theater, configured with 3 or more surfaces containing stereoscopic, head-tracked, computer graphics.

Fig 2.8 - Photo of a user using the CAVE system [15].

1994 - 2012 – This period of time was marked by a decrease of commercial interest about the VR technology. Even though the industry had already established academic conventions (such as the first convention VRAIS'93) and it was continuously been developed in university laboratories. Limited computing performance, and a series of commercial failures in the entertainment industry such as SEGA's VR glasses and Nintendo's Virtual Boy due to problems such as high price and technical limitation, set the slow commercial pace during this period.

Fig 2.9 – Virtual boy, commercial failure due to technical issues, created by the company Nintendo[16].

2012 – Oculus Kickstarter campaign – The entrepreneur presented his prototype to key participants of the gaming industry before launching in 2012, a campaign to raise funds at the website Kickstarter with the objective of developing an affordable VR HMD called Oculus Rift, focused primarily on the gaming industry. This innovative product received huge support, raising ten times more than the initial goal of $250.000 and was responsible for gaining once again the public interest in VR technologies.

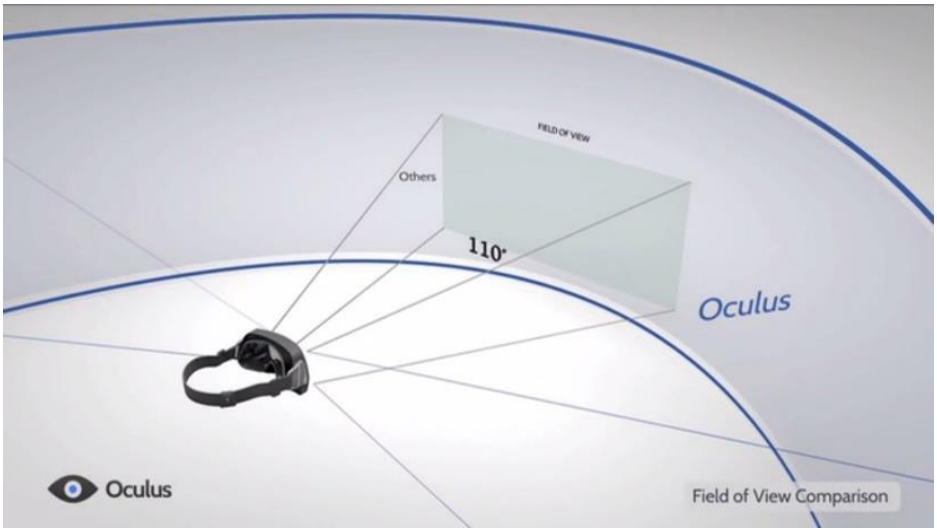


Fig 2.10 - Image retrieved from the Kickstarter video of Oculus[17]

2014 - Facebooks buys Oculus company – Even though the company had not yet released publicly its main product the Oculus Rift, the giant tech company Facebook saw the huge market potential and purchased the Oculus company for an impressive amount of 2 billion dollars [18].

2016 - VR Commercial Competition starts – This year was marked with the commercial launch of many HMD's to the consumers market. Companies such as Google, Samsung, HTC, Sony all deployed their VR HMD's version, setting up a more competitive market, increasing the number of options and driving prices down.

The concept of VR existed for more than 50 years when in 1965 Ivan Sutherland's described "the ultimate display", before being able to establish as it is nowadays a faster-growing industry. At the beginning of the 1990s the VR systems started to popularize due to a few of its commercial releases and created a lot of expectation from the public due to its potential and curiosity, however it failed as a commercial product due to either technical limitations that could potentially make people feel sick or due to its incredibly high costs.

This variation of popularity throughout the years can be explained with one branded representation of emerging technologies created by the firm Gartner called the "hype cycle". This representation has five sequential key phases:[19]

1. **Technology trigger** - A potential technology breakthrough kicks things off. Early proof-of-concept stories and media interest trigger significant publicity. Often no usable products exist, and commercial viability is unproven.
2. **Peak of inflated expectations** - Early publicity produces a number of success stories — often accompanied by scores of failures. Some companies take action; many do not.
3. **Through of Disillusionment** - Interest wanes as experiments and implementations fail to deliver. Producers of the technology shake out or fail. Investments continue only if the surviving providers improve their products to the satisfaction of early adopters.
4. **Slope of Enlightenment** - More instances of how the technology can benefit the enterprise start to crystallize and become more widely understood. Second- and third-generation products appear from technology providers. More enterprises fund pilots; conservative companies remain cautious.
5. **Plateau of Productivity** - Mainstream adoption starts to take off. Criteria for assessing provider viability are more clearly defined. The technology's broad market applicability and relevance are clearly paying off.

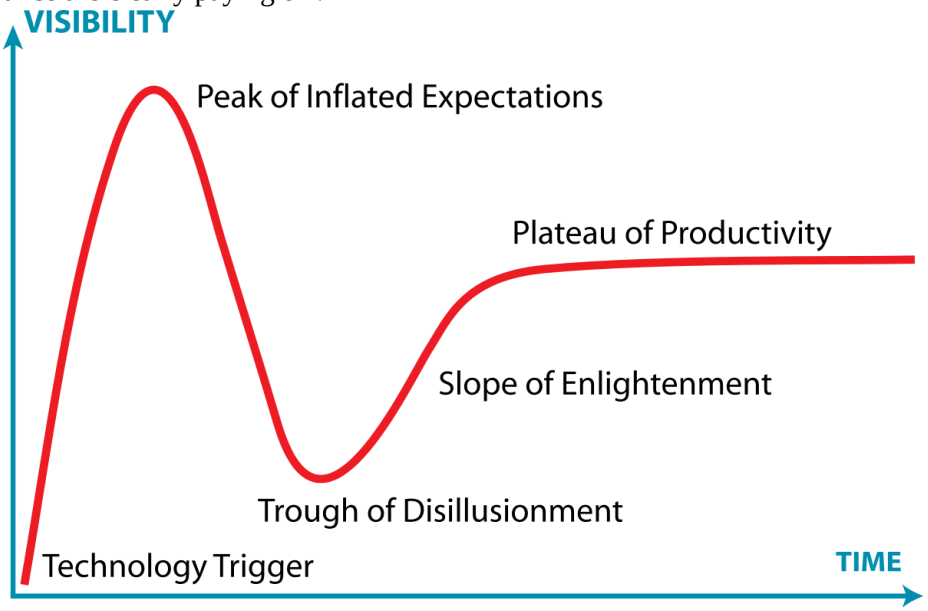


Fig 2.11 – Graph representing the Hype Cycle [19].

Taking this representation into account, it is possible to say that the VR technology in the current state, considering the recent entrance of multiple technology companies into to market, is at the *Slope of Enlightenment* phase. As more and more industries start to use and develop applications using the VR, it is reasonable to say that the VR can potentially reach the next phase of this cycle, the *Plateau of Productivity* in the following years. Gartner's report of July 2017 for instance, predicts that VR will reach this phase in between 2 to 5 years[20].

In the AEC industry, as it is going be discussed in the chapter 3 of this work, recent years has shown a significant increase in direct applications at the professional level. As the benefit results from the applications continue to start appearing, it is expected that the adoption of this technology continues to escalate.

2.3. HARDWARE

A virtual reality system is composed of multiple components that must be well integrated. These components are the system hardware, together with its support software responsible for linking the displays with the input of the user's actions, a user interface design that allows a convenient mean of interactivity, finally and equally important is the virtual world containing the content that the user will interact with.

The hardware used in these systems can be roughly categorized as display devices, with input devices that a user consciously triggers, the other input devices that are responsible for tracking the user position, accompanied by the computer that supports the modeling and rendering of the virtual world [2].

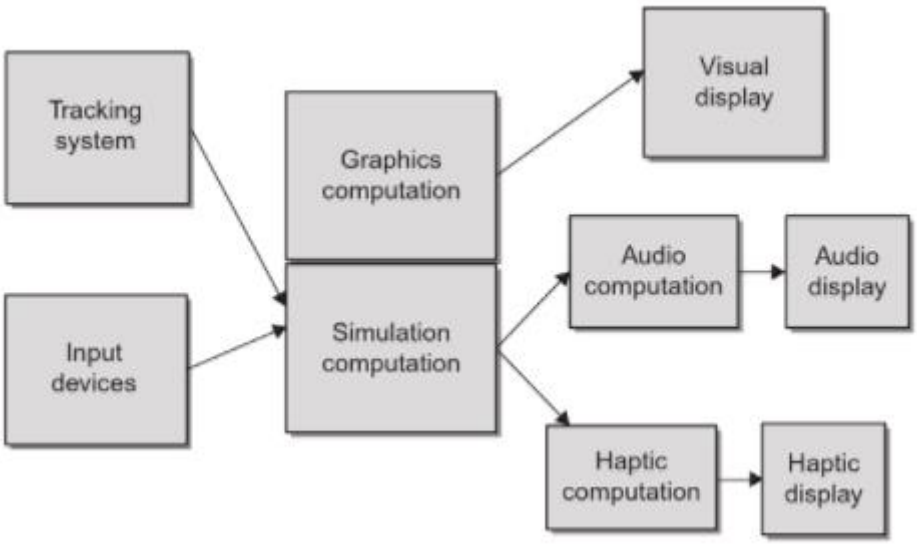


Fig 2.12 – Example of a typical VR System [5].

The computing component of the VR hardware is responsible for calculating all the behavior of the virtual world and rendering it to be represented for the user through mainly visual or audio displays. To be able to have an effective immersive VR experience, it requires real-time interactions and rendering.

The visual display of the hardware has the most influence on the overall design of the virtual reality system. This influence is a result of the visual system being the predominant sense of most

of the people. There are primarily two different types of visual display: stationary and head-based. The stationary refers to systems such as the CAVE and the head-based refers to head-mounted-displays(HMD).

2.3.1. VR SICKNESS

The experience of feeling sick while being in VR can ruin the experience to the user, and possibly make them be cautious and hesitate to use an HMD again [21]. This type of sickness causes very similar symptoms as the motion sickness, including nausea, general discomfort, headache, vomiting, sweating, and apathy. Motion sickness it is caused by the disagreement between the visually perceived movement and the vestibular system's sense of movement. However, VR sickness differs from motion sickness because the cause can be the visually-induced perception of self-motion, while the real self-motion is not necessarily needed[22].

Currently, there isn't a common agreement of the primary cause of VR sickness. One theory is that similarly to the motion sickness, the VR sickness is caused through the sensory conflict of the perception of self-motion and the vestibular system, particularly when these inputs of motion are different from with the user's expectation based on prior experience. A common VR experience example is the virtual experience that simulates the participant to be at a roller-coaster. Even though the participant it is seated or in a standing posture, the simulation goes on a fast-paced speed, providing the user a perception of motion. As a result of this conflicting perception, not so rarely the users experience some of the VR sickness.

The second theory about the cause of the symptoms is related to the postural instability. This explanation says that the sickness is caused by poor postural adaptations in response to unusual visual stimuli. The idea is that if your eyes convince your brain that you're in the virtual world, the body it is going to respond to it, instead of the real physical world, creating an instability at the persons' vestibular sense. For example, a user that is experiencing VR of a simulation of driving a car, while the car approaches a turn, the participant unconsciously will try to lean into it to balance himself, generating this instability[23].

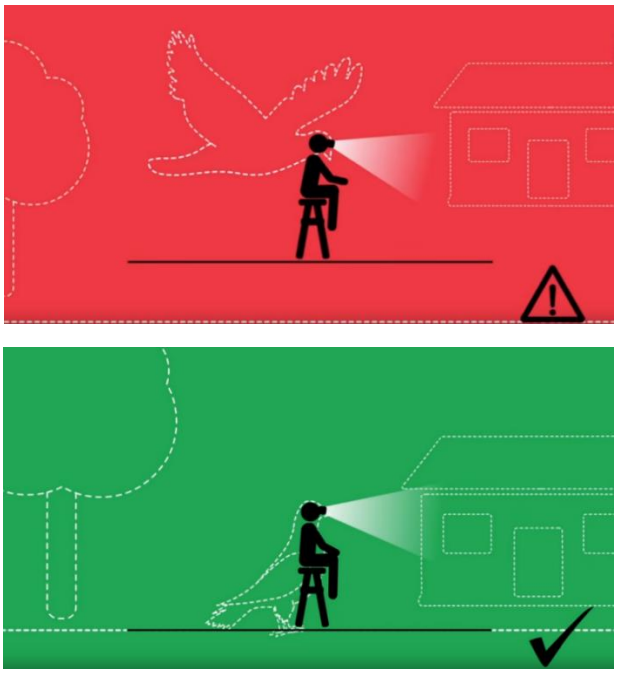


Fig. 2.13 - Design considerations to avoid sickness, let the user control the pace of the movement [24]

2.3.2. CAVE AUTOMATIC VIRTUAL ENVIRONMENT

This technology was created by the EVL of the University of Illinois in 1992. It consists of projectors directed to between 3 and six walls on a room-sized space. It is a video theater situated in large rooms that integrates the user to a space giving the possibility to interact with the virtual world through motion sensors. The user uses a stereoscopic glass and sees through the projections at the walls on a scale 1 to 1.

Since it is necessary to have large rooms to be installed and considering that the technology consists of lots of different components such as the walls that have projectors or big LED panels. It contemplates a huge cost to implement this technology, limiting its usage and development to almost exclusively Universities laboratories or big companies.

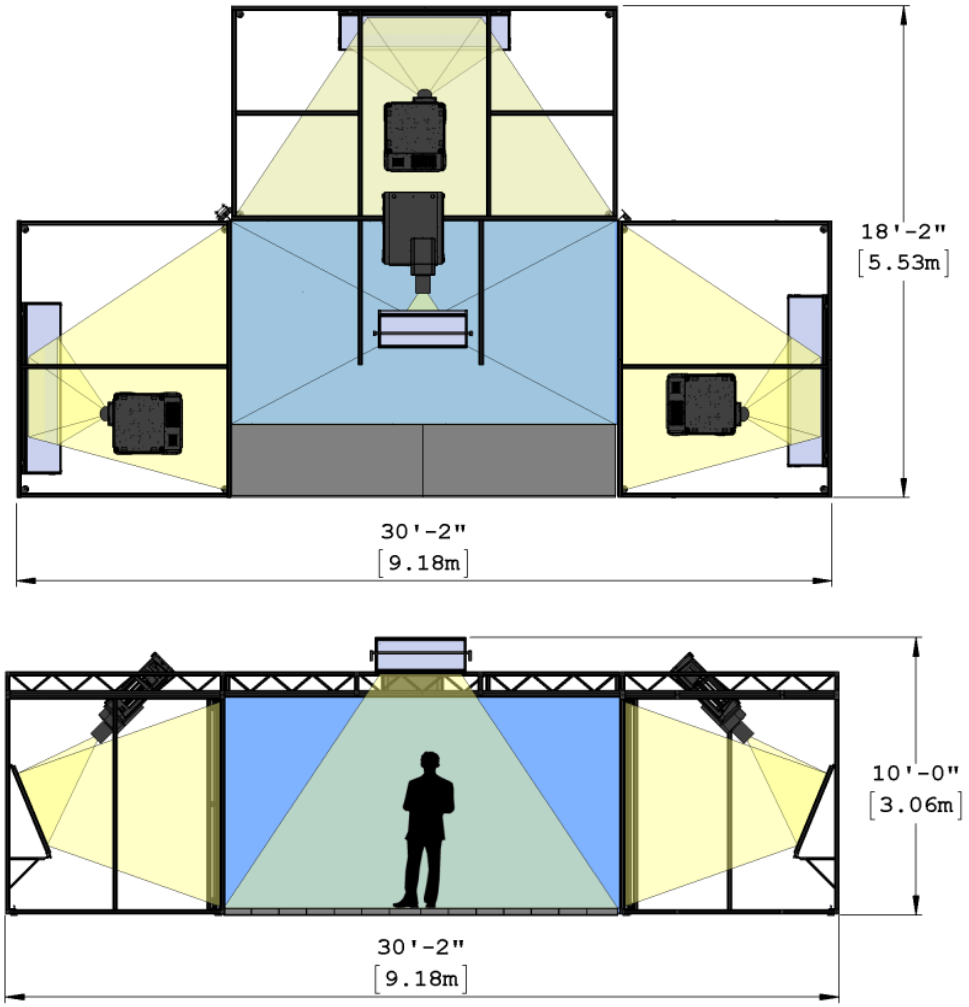


Fig. 2.14 – Image of the product VisCube C4-4K from Visbox[15].

2.3.3. HEAD-MOUNTED DISPLAYS

Even though the concept of the HMD's has been present for around 50 years with the Ivan Sutherland Sword of Damocles, this type of technology only recently started to regain a lot of investment and attention. The recent development of support VR elements such as accelerometers (detects motion), gyroscope (detects the orientation) and displays (better resolution) that it is

accessible to most of the population through the smartphones, allowed the technology to be reachable with lower prices and allowed the users to have better experiences. It is also relevant to highlight, the increase of computation capacity together with tracking sensors technology that also allowed HMD's to thrive[25].

There are a lot of requirements of the HMD's to give the user an enjoyable experience due to a few reasons. First of all, the participant has a device with visual displays very close to their eyes, it is important that these displays contain enough refresh rate(frame rate per second), otherwise, it can be visually uncomfortable. Secondly, the hardware device needs to correctly track the user position, if not one of the most important elements of VR, the sense of immersion of the user can be broken. The immersion as it was discussed in the section 2.1, can be related to the subjective feeling of being present in another place. Because of this importance, one of the main companies of VR hardware, the Oculus has done a research and within it detailed the five key elements hardware requirements for *presence*[26].

Table 2.1 – 5 key elements for presence

Tracking
Six degrees-of-freedom (ability to track not only rotation but movement through 3D space too), 360-degree tracking (ability to track 6DOF no matter which direction the user is facing)
Sub-mm accuracy, no jitter noise (no shaking while the tracked object is completely still)
Comfortable tracking volume (the space within which the headset can be tracked in 6DOF)
Latency
Less than 20ms from when motion happens to when the last photon is emitted from the display
Minimized loop: tracker → CPU → GPU → display → photons
Low persistence
Currently, less than 3ms to turn pixels on then off (to reduce smearing)
90Hz+ refresh rate to avoid visible flicker
Resolution
Correct stereoscopic (3D) rendering
At least 1,000 x 1,000 pixels per eye
No visible pixel structure
Optics
Wide field of view: greater than 90 degrees horizontal
Comfortable eyebox (the minimum and maximum eye-lens distance wherein a comfortable image can be viewed through the lenses)
High-quality calibration and correction (correction for distortion and chromatic aberration that exactly matches the lens characteristics)

One complication of the HMDs, however, is the need to render each eye separately, since the HMD have two different displays (one for each eye) that are on a slightly different position to one another, the hardware must render twice as much as compared to a regular computer screen. Because of this, the VR experiences requires to be optimized, otherwise, it can compromise the experience overall with lowered frame rate per second[27].

There are currently lots of different options in the market with different functionalities. It is important to note the difference between computing hardware, the visual displays, tracking system, price range.

Modern-day VR HMD's can be categorized roughly into two categories: mobile or tethered. Mobile headsets are the ones that don't require an exterior computer hardware to be used. They are a shell with lenses into which it is placed a capable smartphone. The lenses separate the screen into two images for the eyes, turning the smartphone into a VR device. This category of device offers some advantages. First of all, they do not require any wires to be connected to the headset, all the processing is done by the smartphone, allowing more freedom to the participant. Another advantage is the relatively cheap price of it, devices such as the Google Cardboard can transform a smartphone to a VR device for less than 15,00€[28]. Because it is processed through a mobile phone, however, it understandably has some limitations. Since the phones aren't designed specifically for VR, they can't offer the best picture even with special lenses, most of the mobile headsets have only the capability of 3 degrees of freedom (the participant can only rotate his vision around) and they are also underpowered compared to tethered PC VR[29].

Fig 2.15- Image of the low-cost VR system, the Google Cardboard[28]

Tethered headsets are physically connected to a computer. Most of the options in the market offer headsets with a dedicated display, built-in motion sensors, and an external camera tracker. Since they are a dedicated hardware for VR, understandably they offer better image fidelity and head tracking. One disadvantage compared to mobile VR it is the cable connections that can be unwieldy.

2.4. SOFTWARE

A variety of software components are needed to enable an effective VR experience. The software being used to allow VR contains elements such as the libraries of content(objects) for simulating events, the rendering display imagery capability, the interface with input and output devices and the laws of nature of the virtual world.

The computer software used in the AEC industry evolved through many years from being able to create simple 2D blueprints until the recent ever-growing adoption of the more complex BIM software. As it was discussed previously in this work, to have an enjoyable experience in VR it is important for the software to be able to handle real-time simulation and rendering. Even though the BIM software's used are extremely useful for many of the industry needs, most of these software currently lacks the capability aspect of real-time manipulation with the sufficient refresh rate requirement for VR.

Differently, from the AEC industry, game-engines used by the entertainment industry constantly had to worry about frame-rate per second(FPS), which allowed them to continuously develop to be able to create video games with an optimized FPS. A game engine is a type of software that contains several components used for the process of game development. Since each video games have a number of functionalities in common with each other, the process of development is often reduced by using these types of engines in order that the developers focus on the creation of content itself, instead of being required to program all the components of the game from ground zero[30].

Taking into consideration the capability of the game engines to real-time rendering, a developed physics engine, scripting and many other components already prepared for the creation of content, the game engine software offer many advantages for creating VR reality applications for various intentions.

2.4.1. PHYSICS ENGINE OR COLLISION DETECTION.

The VR experiences have programmed laws of nature that govern the behaviors and interactions carried out by the objects in the world. A physics engine it is required to be able to simulate physical behavior in the virtual environment. The physics engine often includes features such as rigid body dynamics, fluid dynamics, soft body dynamics and collision detection. The physics engine is essential for VR because it is this feature that calculates the physic interaction between the elements. For example, the physics engine can be responsible for not allowing the avatar (the representation of the human in the virtual environment) to run through walls and many other important features for the experience.

2.4.2. SCRIPTING

Scripts are codes written for the run-time environment that automate the execution of tasks that could alternatively be executed step by step by a human operator. This is a type of coding language that is often interpreted rather than compiled. While the application it is running, many of the behaviors of the virtual environment can be controlled by a script. The scripts are essential for the applications, they are used for various tasks, such as controlling the user inputs, determine actions to certain events, changing scenes, etc.

2.5. AUGMENTED REALITY – THE MAIN DIFFERENCES COMPARED TO VR

While virtual reality the user it is fully integrated or isolated on a virtual environment, there are similar types of technology that can be potentially useful for the AEC industry, which is the case of augmented reality(AR). Augmented reality is defined by Azuma as any system fundamentally having three characteristics [31]:

- Combines real world with virtual elements
- Is interactive in real time
- Is registered in three dimensions.

The concept of AR generally refers to the overlap of virtual objects created by a device with a real object within the real environment, creating a mixed world. The most noticeable difference between AR and VR is that the participant in AR it is not fully immersed.

Augmented reality in the AEC industry has the potential of bringing information to the construction sites. Although it lacks the immersion factor of VR, the capacity of showing information on the site and the mobility of its devices makes it a potential wearable equipment for the construction workers.

AR is currently used in most of times, through applications on smartphones or tablets. It can be used to bring the BIM digital models to the site, with the intention of visualizing and interacting with it while the worker is at the real location of the digital model. The applications can also allow the worker to a series of useful tools such as:

- Seeing through walls (visualizing piping system, electrical, etc.).
- Measure distances.
- Visualize thermal data real-time.
- Send warning information (for safety purposes).
- Etc.

Although the ability of showing information in real time in the site create benefits for the worker, it is necessary that the user interfaces and the way the worker interacts with the information be as easy as possible, because the objective it is the worker to be focused at the activity itself, and not be distracted with other types of information that could be even potentially dangerous to the worker.

Some companies already are creating special products designed for the AEC industry, such as the DAQRI Smart Helmet(Figure 2.16) which the design is intended to replace the hardhats used in the construction sites. This device is capable of displaying information to the user in real time while the worker is still hand's free. Because the user has its hand's available, it enhances the productivity of the worker due to multitasking [32].

In summary, the AR devices comparatively to VR devices have more potential to be used at the construction sites for workers at the production level. While the VR devices, as it is going to be discussed in the next chapter, are more effectively used in the conception phase, at the planning level.

FIG 2.16 - DAQRI smart helmet, an AR equipment.[32]

3

VIRTUAL REALITY APPLICATIONS IN THE AEC INDUSTRY

Since VR is a medium, similarly like others type of media, it can have various purposes of usage. This technology it is being applied and studied, not only in the AEC industry but in other fields such as the military, automotive, entertainment, educational, medical, artistic, etc. [2].

The professionals must consider whether their objective will gain a sufficient advantage via this medium. Initially, people are attracted to VR simply due to the novelty of the technology. That is a valid reason to make use of it, if the objective is to investigate new technologies or to attract people to see a project, for advertising or marketing purposes. However, for using it with professional intentions at the project development, the efficiency of the application needs to be justified and present benefits, otherwise, the usage of the technology can be minimized as a gimmick (*"a trick or device used to attract business or attention"*[33]).

In this chapter, it is going to be discussed the different types of applications in some areas related to the AEC industry, with the advantages and disadvantages being presented.

3.1. VR AS AN INNOVATIVE WAY TO LEARN

As it was discussed in the previous chapter, VR has some promising features that make the experience more appealing to the participant. The immersion, interactivity and sensory feedback all together create a good set of tools to stimulate the comprehension about a specific subject or situation[34].

In the VR context, the experimental nature of the experiences supports a constructivist learning path. Constructivism is a theory that the person constructs knowledge by learning from their previous experiences. People acquire the information from these experiences through their sensory system. Thanks to the replacement of the sensory input from the real world to the computer-generated sensory input, VR allows the user to actively be present a new situation, instead of just imagining it [35].

In summary, humans that actively engage with new information are more likely to retain this information and be able to remember it in future moments. As a result of that, the VR interactivity element presents advantages to be used in the learning path.

3.1.1. EDUCATION

The applications in VR with educational purposes can be designed to be used during all phases of the learning process of AEC students. Didactic models can be prepared by the teachers to present interactive experiments in physics or chemistry, structural detailing, architectural concepts, etc.

Having a 3D immersive environment can be an important tool to enhance the ability of students to develop their space perception[36]. The interactivity of a virtual environment stimulates the participant on critical thinking, rather than being just passive learners[37].

Humans throughout their entire lives are accustomed to gain information knowledge by seeing objects on a human scale. The comprehension of a representation by viewing through a computer monitor or a drawing can be harder to learn, especially if the person does not yet have the technical knowledge to understand it. Often, structural elements can be difficult to represent on a 2D single plan, and requires many drawings to be able to represent it such as plans, detailing, sections, isometric view, etc. Having a 3D model that the participant can view from all angles and interact with helps the student more intuitively to comprehend the element[38].

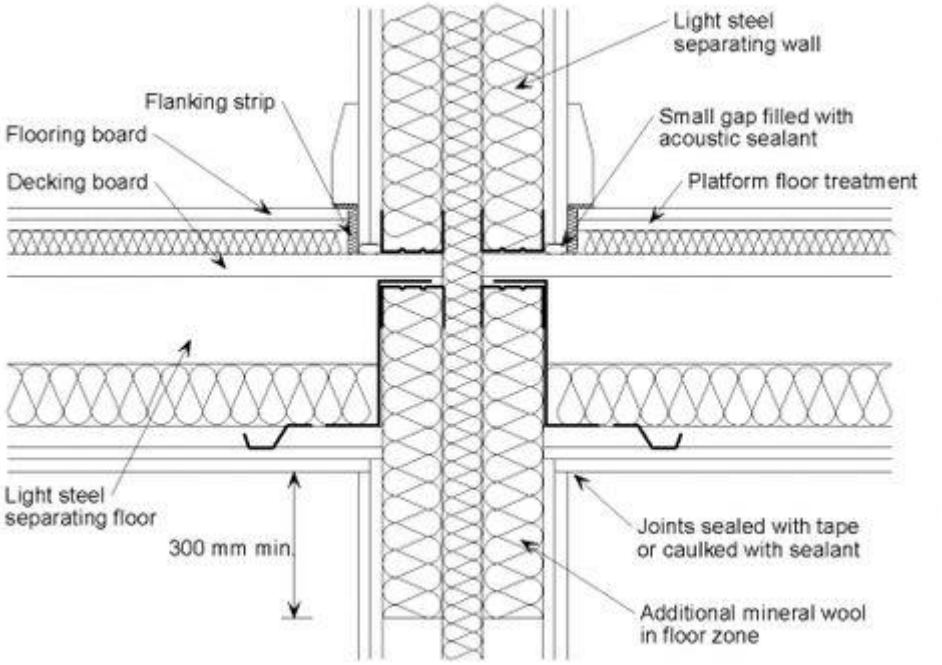


Fig 3.1 – Structural detailing of an element [39]

3.1.2. TRAINING

As we have seen in the history of VR, one of the oldest examples of this technology, flight simulations, were used for training purposes. One main reason for this is that mimicking the three-dimensional world has a clear metaphor; the behavior in the virtual world should simulate the behavior in the real world.

The applications created for training, however, allows one to do more than just simulating reality. For example, in construction sites often the workers face risks in which minor mistakes could potentially create a harmful situation for the worker itself and others. The virtual world simulation can add the ability to practice uncommon, dangerous, and expensive tasks, without having to face 'do-or-die' consequences.

In VR it is possible to put these workers in potentially dangerous situations and be able to make them train until they are ready for it. Their mistakes committed in the application doesn't generate any real consequences, and they can benefit with it by rewinding the experience over and over until they gained enough knowledge to understand the procedure for achieving success in the task. By not doing everything correctly as well, the trainee learns the real consequences of their actions.

By practicing tasks in this computed-mediated system, other benefits are noticeable. The creator of the content has the control over what scenarios are going to be presented to the trainee, and possibly change the scenario in response to the performance. If the trainee is having difficulties in realizing one specified task, easier scenarios can be presented with increasing difficulty in order that the worker can understand better the process of accomplishing the task and learn from their previous mistakes. Because the scenarios are made in virtual environments, the performance can be easily recorded and analyzed.

One safety intended application that was created by Bouygues Construction concerns the placement of formwork elements at the construction site[40]. The trainee in the application it is required to do security tasks to begin the training such as putting on personal protective equipment like the hard hat and hearing protection. When the training starts, the apprentice it is demanded to do all the tasks required to a safe placement of the framework, such as helping to guide the placement of the framework and placing all the safety screws of the framework.

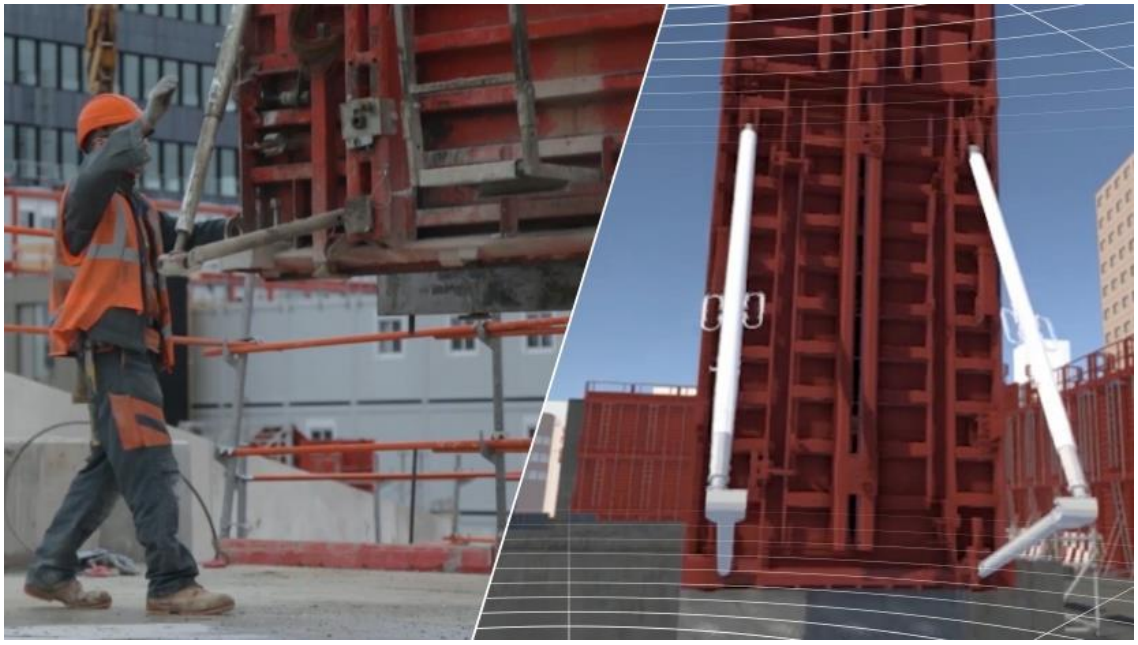


Fig 3.2 - Application of placing frameworks [40]

One common practice used for training labor with an experimental aspect is related to On-the-job-training(OJT), in which the worker performs the activities while being at the job itself, being granted a certain level of independence. However, this type of training is often criticized for being limited, expensive and occasionally lack the actual training context itself [41].

Another example of this type of training related to the AEC industry are the simulations created by the company Industrial Training International. The simulator is used on a crane course, and it contains a VR HMD integrated with pedals and throttles that simulates commercially available cranes. The trainee has to complete a serial of tests that allows him to increase their familiarity with the equipment at the same time as developing their skills[42].

Fig 3.3 – Crane simulator [42].

3.2. PROJECT DEVELOPMENT

The activity of developing a construction project regularly involves multidisciplinary teams, different firms and it is necessary to take into consideration many different factors that makes each project, one of a kind. As a result of this, new problems appear and needs to be solved and evaluated throughout the whole project cycle. One fundamental aspect of the success of a construction project it is communication. The final users of the building need to express their requirements and expectations towards the project in a manner that the designers understand, and the designers need to be able to express their ideas and solutions in a way that the users understand. The usage of VR applications towards project development is beneficial due to its intuitive understanding for all people involved in the project.

Adding up to these benefits is the virtual prototyping feature in VR. This can be used to analyze a product from a variety of perspectives such as ergonomics, constructability, and aesthetics.

The ergonomics of a building can be difficult to express through traditional plans. It often requires the person to use their imagination to put themselves in the specific situation. By allowing people to 'test' the building in VR, it helps them to discover little annoyances that could potentially appear only when the building it is already being used, or even discover problems that can make the planned area unusable. Some examples of verification in construction are to check if the area planned it is comfortable or if there isn't anything that blocks the passage that would eventually cause tripping and many others verification.

Using VR to analyze aesthetics can presents benefits, especially due to the scale perception of the space. The immersion factor provides a different perspective to analyze some aesthetics qualities such as proportion, symmetry, unity, and rhythm. Since it is a computer-generated environment, changing aesthetics elements such as materials, colors and geometry forms doesn't imply on the necessity of rebuilding, resulting in cost savings.

FIG 3.4 – Ikea Kitchen – It simulates a kitchen environment allowing ergonomics to be tested[43]

Due to the benefits that VR can present to a construction project such as immersion and interactivity, it can be used in different phases as different tools such as project modeling or to support the decision making of the project.

3.2.1. VR AS A DESIGN TOOL

Whenever people build things in real life with tools or even with their own hands, it is an intuitive process. Humans are used to understanding the scale of objects and spaces through their everyday experiences. The constructability of the elements is easily understandable. The designers traditionally need to express their space ideas on representations in 2D or 3D using paper or a computer, losing in this process, the natural engagement from the persons perspective. VR can be an alternative for bringing intuitiveness by allowing the designer to engage more actively with the space.

Using intuitive user inputs such as physical controllers or technologies such as the Leap Motion that tracks the user hands, combining with the tracking of the creator position, allow the architects or engineers to use their entire body to create their designs. Because of that, the designer can have a much more intuitive tool to express their ideas of models and potentially be even entertaining. Placing elements in a virtual environment has a defect awareness as similar as building things in real life because the designer intuitively sees the elements in scale. Using VR to analyze aesthetics can presents benefits, especially due to the scale perception of the space. The immersion factor provides a different perspective to analyze some aesthetics qualities such as proportion, symmetry, unity, and rhythm. Since it is a computer-generated environment, changing aesthetics elements such as materials, colors and geometry forms doesn't imply on the necessity of rebuilding, resulting in cost savings.

One example application used for design it is shown at the figure 3.5, the designer uses their VR controller to place, change materials, change scale, the orientation of the elements while being on a one to one scale environment, supporting the creativity of the designer.

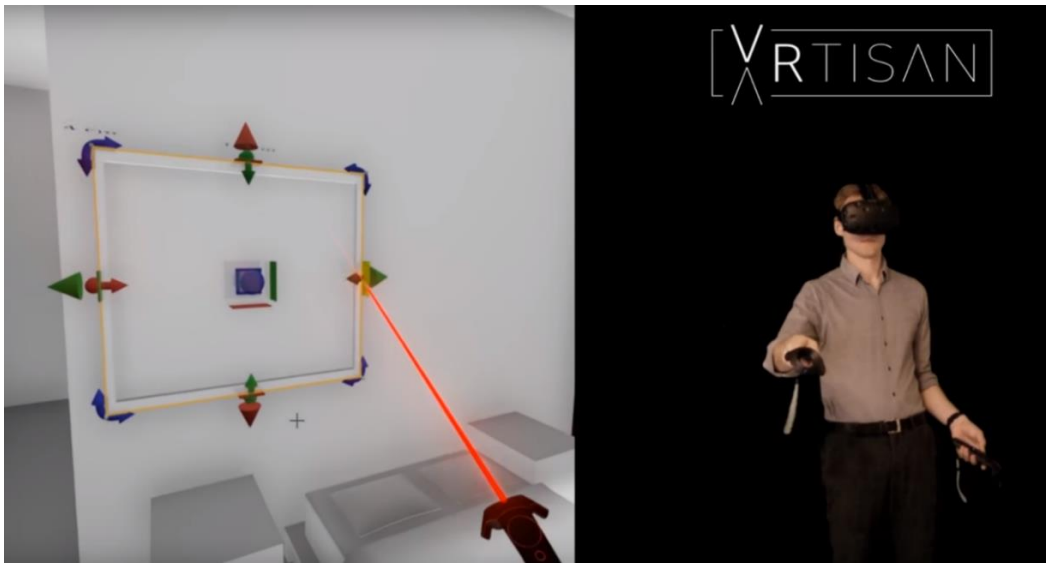


Fig 3.5 – Architectural Design in Virtual Reality - VRtisan [44]

One potential use for design in VR is using the empathic design approach. This type of design is user-centered and takes into consideration the user's feeling toward a product. In the construction activity, it often takes into consideration exclusively the legislative design approach, supporting a design through various regulations previously stated. For example, when it concerns the accessibility of the building, there are a number of regulations that must be implemented. However, these regulations often can be complicated and involve different factors. Using a VR application with a wheelchair as a controller, for example, the designer can evaluate if the design not only complies to the regulation but most importantly if it is functional, by putting him or her on the others person perspective [24].

Fig 3.6 - Wheelchair VR newton's apple[45].

3.2.2. VR AS A COLLABORATIVE AND COMMUNICATION TOOL

Since the projects related to the construction business is a one of a kind type, the project managers very often must apply their knowledge to improvise solutions for the potential defects and

problems. The solution presented can be successful and often solves the problem entirely, however, these types of improvisation are also that can lead to even more problems. Being able to anticipate and solve problems in the planning phase presents advantages towards cost and time-saving at the construction sites.

A common practice to anticipate problems of a construction it is by integrating models with the different layers of plans (architectural, structural, electrical, plumbing). However, by seeing plans or a 3D model in conventional equipment such as computer screens or projectors, some problems possibly can pass as unnoticed. By using VR to immerse an observer into the model the perception of the space, it is as similar as being present at the construction site. The point of view of the observer transforms from being external to being inside ('present') at the scene. This allows VR to be a useful tool to discover problems associated with the constructability, ergonomics, and aesthetics of the space.

Often people that can give valuable feedback for the construction project does not have the technical knowledge to visualize plans and elevations of a project. One practical case where VR can be useful is on the decision making towards complex facilities such as the one Mortenson Construction used at the Sanford Fargo Medical Center. To be able to integrate more the final users into the project, VR was used to show surgeons how the final project it was going to look like because most of them don't have the technical knowledge to be able to read construction plans effectively. The efficiency of space and movement in the operation room can be potentially the difference between life and death. Allowing the surgeons to use their experience and give their feedback on the design it is essential for a successful plan. By inserting them on an immersive space, it gives them intuitively the position awareness of the outlets, medical equipment, resulting in a better overall understanding of how the room it is going to be and function. This usage permitted them to give more efficient feedbacks. The construction company claims that it avoided in total the cost of potential $675.000,00 by change-order rework and medical device relocation. [46]

Fig 3.7 – Demonstration of a virtual operating room[46]

The concept of BIM 4D it is referred as the intelligent linking of the 3D CAD components with time or schedule related elements [47]. The main objective of 4D is to visualize the entire series

of events and be able to see the progress of the construction activities. The visualization aspect of VR can be potentially linked towards 4D and presents benefits towards the decision making of the project. One example of 4D connected with VR was used in the project of the second tallest skyscraper in London called the '22 Bishopsgate'. In this project, the design information was attached to tasks within the planned schedule. Activities such as steel sections being placed, the supply materials being reconfigured, etc. are all attached to graphical information in order that the observer can understand the sequence of activities. It was an effective tool to communicate and give the work personnel an informed view. Another advantage of the VR usage was the evaluation towards the crane positioning, since the top of the building it is only nine meters below London aviation's ceiling, the project team was able to test virtually the fitting of the tower cranes before doing it in the real construction [48].

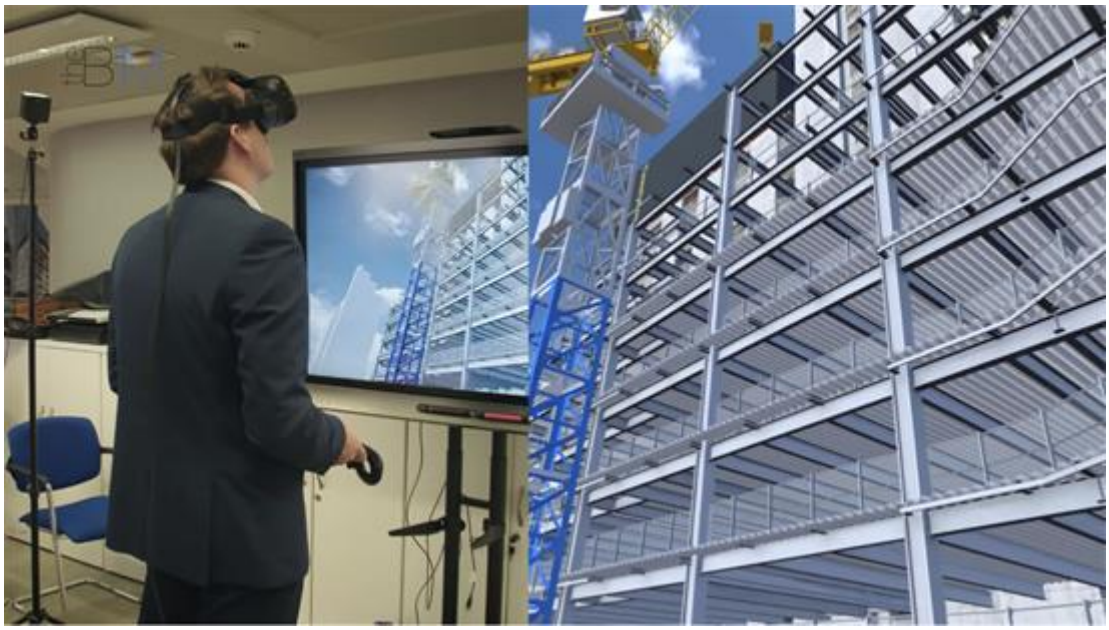


Fig 3.8 - Immersive 4D VR – intelligent linking of construction elements with time[48].

Fig 3.9- Simulation procedures with a crane inside the virtual model[48].

Being able to communicate and collaborate with others involved in the project is advantageous and supports the decision making. One conceptual situation that can exemplify a usage of VR in communication it is a project manager on site needs to have an urgent feedback for a problem from the structural engineer. Currently, one alternative to be able to create a solution it is by sending multiple emails to each other, however, consistently this is time-consuming because it is necessary to write down many lines describing the problem while annexing many photos of the problem, and it is often a non-efficient way of communication. Another alternative to achieve a solution it is the engineer transporting himself to the construction site. Losing his time while transporting himself to the construction site and going back to the office afterwards. This sometimes it is not possible because, for example, the structural engineer works for a company situated in France, and the construction site is in Brazil. In VR it is possible that while the project manager is at the construction site, he can use a tablet or a smartphone to make notes and annotations on describing the problem in real time, while the structural engineer is seated at the project office visualizing the designed 3D BIM modeled structural elements in the virtual environment at the same time he is seeing the project manager through a panel interface with a telecommunication window. By visualizing elements in detail through a virtual environment, it enhances the capability of decision-making. One similar concept was already shown by the Microsoft HoloLens that is presented at the Fig 3.10.

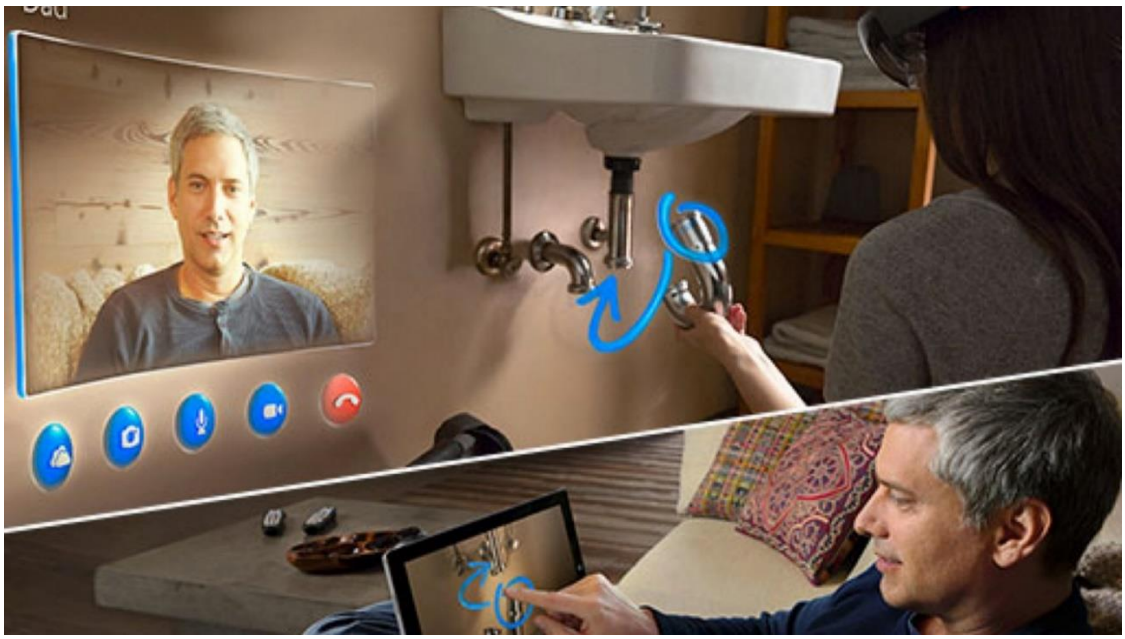


Fig 3.10 – Concept communication through devices designed for HoloLens[49].

3.3. REAL ESTATE INDUSTRY

The activity of selling real property it is known to commonly use new technologies to improves sales. The application of virtual reality technology in this activity it is already a common practice in companies throughout some markets[50]. The VR technology it is advantageous for the usage by selling agents to help to eliminate two common difficulties.

- The selling agents have to manage the time it takes to go from one visit to the other, very often dealing with traffic and losing productivity.
- Showing the clients an immersive and a reliable representation of the real estate helps to reduce surprises possibly caused by other media. For example, taking into consideration images that give a mistakenly perspective of the space.

VR by being immersive and interactive, it can be emotionally arousing and potentially quicken up the sales[51]. When the construction project it has not yet finished, one VR model can offer a reliable source of information to the buyer of how the building it is going to be like [52].

The virtual models can also be used for properties that are already built. The designer has to either model the property from ground zero using a software using the plans of the space or use technologies such as laser scanning, in which the house it is scanned and digitally created.

The VR applications are similar to virtual "tours," allowing the buyer to visit the property virtually. The three main options common practiced are:[52]

- Guided virtual visit: A promotional video at 360 degrees. It allows the user to visualize the property by changing his field of view.
- Interactive virtual visits: The buyer can click and go to a specific hotspot and see information about the property, be able to interact with some hot spots. interacting with some elements such as turning on lights or changing materials of the house. One example can be seen in figure 3.11.
- Virtual commerce: The property it is customizable just as the buyer requires, he virtually chooses from a range of furniture elements, changes materials of the surroundings, change colors of walls, etc.

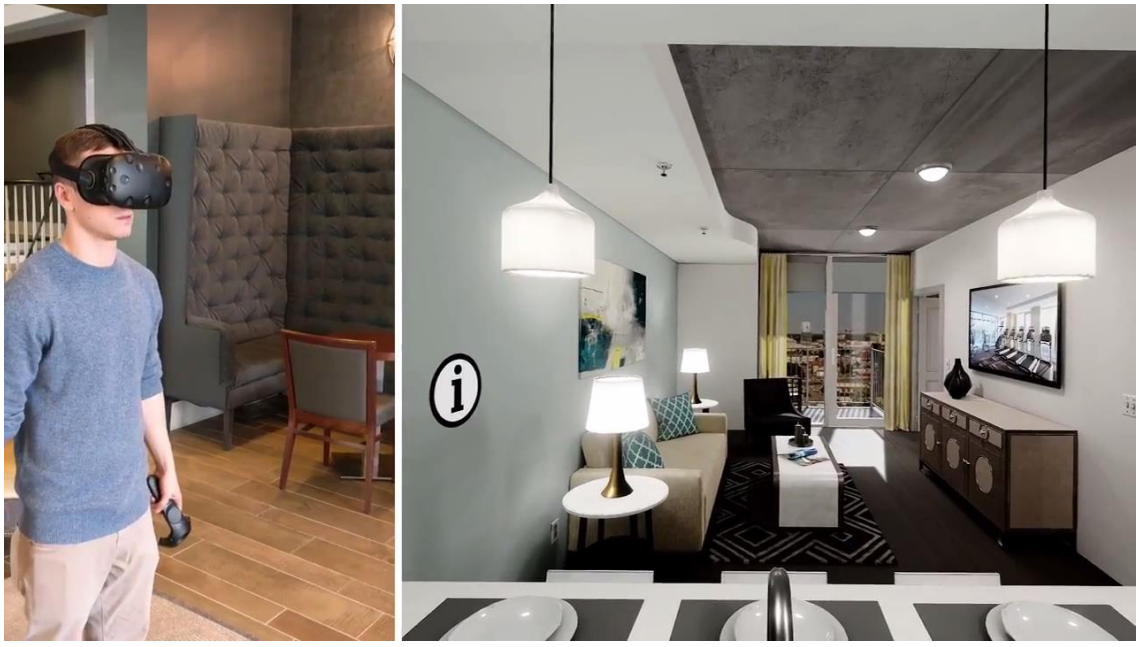


Fig 3.11 – Interactive Virtual visit in Real estate[53]

4

DEVELOPMENT OF A VR INTERACTIVE COLLABORATION APPLICATION

4.1. THE PROPOSED APPLICATION

As it was mentioned before in chapter 3 of this thesis, VR has the potential to be used as a collaborative tool due to its effective and intuitive communication. Taking this into consideration, for this thesis it was developed an application that allows the project stakeholders (client, architects, engineers, contractors, etc.) of the project to visualize, validate decisions and interact with the project model while one observer is immersed in the virtual model with a VR headset. The main concept it is shown in figure 4.1, it shows that the feedback it is generated in the VR application, and then it is inserted back in the Revit model so changes can be made in the BIM model. The creation of the virtual model acts as a virtual prototype potentially substituting the need for physical mock-ups. One definition of the application it is a virtual reality design review application which has the main goals of it as:

- Validate choices of materials and suppliers of the selected equipment.
- Detect design defects.
- Have a new perspective on features such as aesthetics and ergonomics.
- Automatically transfer the information generated at the VR model to the BIM model.

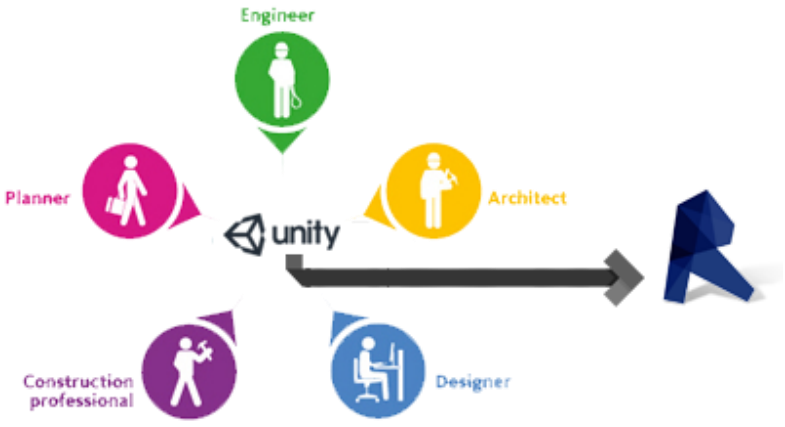


Fig 4.1 – Collaboration tool concept, the feedback it is generated at Unity and later inserted in Revit

Since the application it is used for evaluating and confirming the design decisions towards the project, the usage of it is intended before the construction phase of the project, earlier than any of the reviewed area is already built in order that changes of the project can be made inexpensively. The design review takes place so that the team involved can evaluate if the project requirements are satisfied and decide if it is necessary to change elements due to either technical or personal reasons.

The application can be used as a support for an Integrated Project Delivery (IPD) approach due to its collaboration aspect. This method promotes the collaboration of the project stakeholders in the earlier phases of the project to maximize efficiency towards the project [54]. As the figure 4.2 shows, the capacity to impact on the cost of the project are significantly higher in the earlier phases due to the ability to change the design without having to rebuild anything [55].

Furthermore, the VR application it is an executable file that can be run without the need of installing any software, this facilitates the distribution because it doesn't require a license to run it. The utilization of it can be either individual or made in a group. It generates better results while it is being used in a reunion with the stakeholders because the design or more aspects can be discussed and resolved immediately, filtering possible doubts that can appear individually. Another advantage of the application it is that the user interface it is intuitive, so it does not require any previous technical knowledge to use it, facilitating its integration with all the project members.

One relevant aspect to be considered, it is the project area that it is going to be analyzed. If the primary intention it is to evaluate the material and supplier's equipment choices, depending on the type of the construction it is not desirable to analyze the whole building. One area can be selected and represent the rest of the building, considerably reducing the repetitiveness of the process. For example, if the construction project concerns a student housing, usually most of the student rooms have the same materials and equipment. By using one room or apartment as a model, the design choices can be evaluated much faster rather than evaluating the whole building.

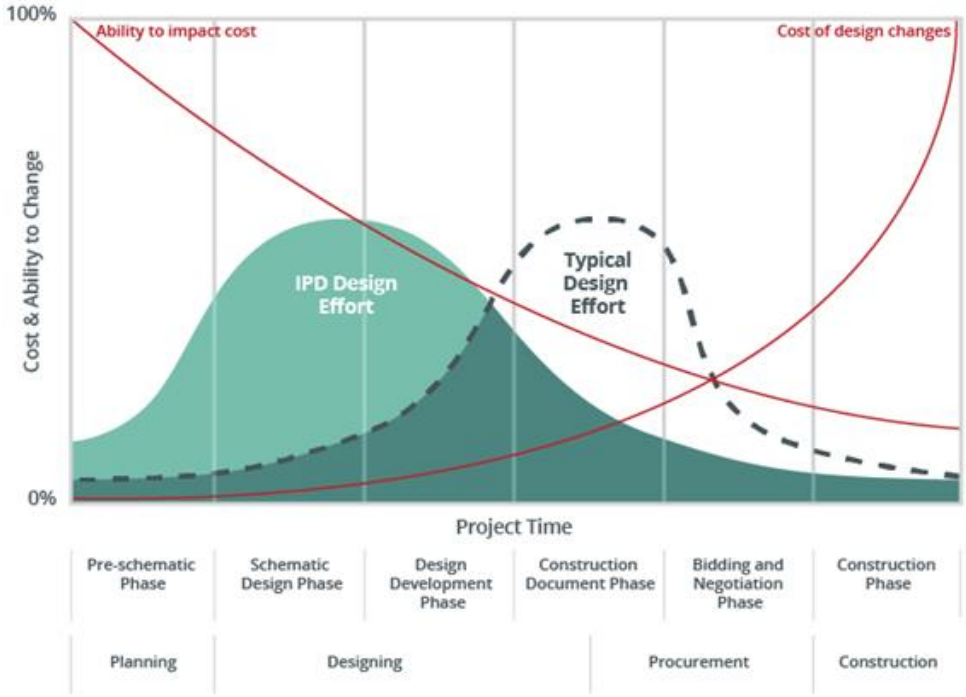


Fig 4.2 – IPD approach with all the project phases.[55]

4.2. THE APPARATUS UTILIZED

In this section, it is going to be briefly introduced the software and hardware used to achieve the final application. In total, it was used three software: Revit, Unity, and Dynamo. The first one it is responsible for the BIM modelling containing all the information of the building. The second one it is responsible for creating the user interface, the interactivity elements and for allowing the usage of VR. Finally, the third one it is used to import the information of the report generated at the Unity application to the Revit model.

4.2.1 AUTODESK REVIT

The Autodesk Revit it is a software designed for BIM that supports professional projects of buildings and infrastructure including features for architectural design, structural engineering, mechanical electricity and plumbing(MEP) and construction. The software carries many tools that help the user to plan, design, construct and manage on an intelligent model-process[56]. One capacity of the software it is the work-sharing. This allows more than one user to access at the same time a shared central model, in which each user can later synchronize changes, changing all models from the users.

4.2.2. AUTODESK DYNAMO

Autodesk Dynamo is an open source software graphical algorithm editor that extends the building information modeling with the data and logic environment[57]. The software is a visual programming tool that it is intended to be accessible either if the user has programming knowledge expertise or not. It allows the users the ability to define customs pieces of logic, visually script behavior, and script using textual programming languages[58]. This application can either be run as a stand-alone “Sandbox” mode or as a plug-in for other Autodesk software such as Revit. While it is being run as a plug-in for Revit, Dynamo can be used for various purposes such as automatically managing information, create graphical representations, modify the BIM model, etc.

By being a graphical algorithm editor, it presents the user an intuitive way to design parametrically and manage the building information without having to know a regular programming language. The coding aspect is developed through ‘nodes', instead of the typical code lines. Each node performs an operation that varies from a simple mathematical operation to complex geometry querying. These nodes are placed through a drag and drop interface, and they are connected to each other by wires, creating a sequential logic workflow. The wires transport information through input and output ports connected to the nodes. The ones that are at left side of the node are the input ports (receives information), and the port that is on the right side are the output ports (transfer information).

Another functionality of the nodes is the capacity of writing code either at DynamoScript (DS) or Python programming languages through the nodes called ‘code blocks’. These blocks allow the user to program functionalities that are not available to the nodes library, allowing the user to develop more exploratory workflows[58]. It is possible to see in the Figure 4.3 the dynamic of a Dynamo code. First, the node ‘Sequence' receives three number inputs resulting on a list of sequence numbers. Thereafter the ‘code block' that contains a simple operation script that multiplies the inputs with each other, receive this sequence of numbers and the number two.

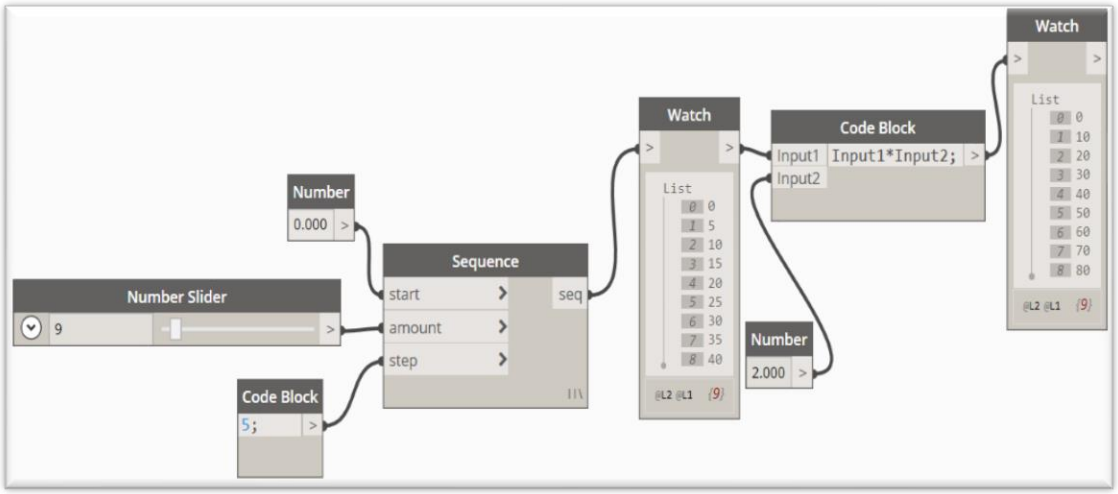


Fig 4.3 – Simple example of how Dynamo nodes work

4.2.3. GAME ENGINE - UNITY

The Unity software it is a multipurpose game engine which is used to develop both three-dimensional and two-dimensional simulations and video games for computers, mobile phones, and video-games consoles. This program supports drag-and-drop functionality, 2D or 3D graphics and scripting through the programming language C#.

This software currently is available at four different licenses; personal, plus, pro and enterprise. Each license contains their differences of level of support from the developer company. The personal license it is the most basic and currently it is free if the company or person has a revenue capacity under of $100,000. All the other license options have a monthly fee that starts from 35$. The program as shown in figure 4.4, it is currently the most used game engine on the market[59]. The main reasons for this are the capacity of creating content for multiple platforms, associated with a user-friendly interface, one free license option that incorporates most of the market share and an extensive manual to help the new users to be familiar with the software.

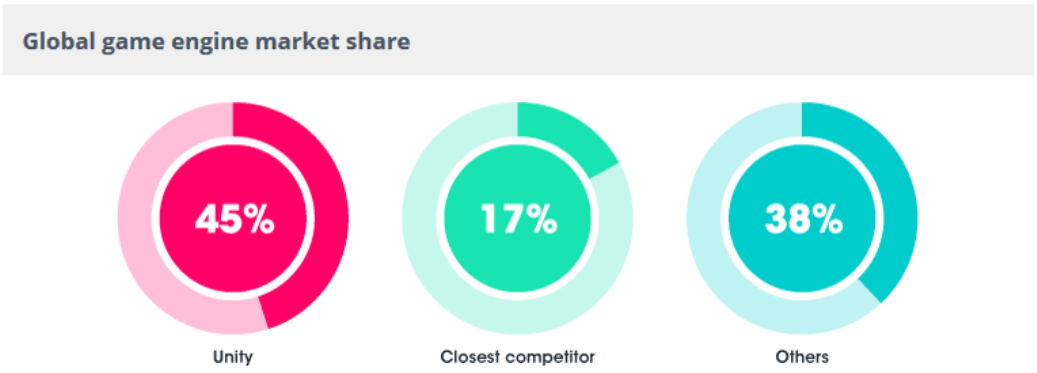


Fig 4.4 – Market Share Unity[59]

Even though Unity it isn't a software intended primarily to be used in the AEC industry, it presents a number of features that were taken into consideration to be chosen as the VR design software such as:

- Most importantly, it supports VR development;
- It includes an extensive manual and a large-scale collaborative community;
- Validated as a game engine that preserves well the Revit geometry through software interoperability [60, 61];
- The author of this work had previous knowledge with the programming language C# used by the software for creating codes;

4.2.4. USED VR HARDWARE - HTC VIVE

The HTC Vive it is a virtual reality headset that it was launched in April 2016 in partnership between HTC and Valve, an American video game developer and digital distribution company. The main components of the system it is the Vive headset, the Vive controllers, and the Vive base stations. They all can be seen together in the figure 4.5.

One essential element of virtual reality it is the tracking system that is responsible for giving the sense of sensory feedback to the user. At the Vive system, the base stations are responsible for tracking the headset and the controllers. The stations emit 60 infrared pulses per seconds that are detected by both headset and controller with sub-millimeter precision.

To interact with the virtual world, the Vive controllers are wireless to give a sense of more immersion. The controller includes multiple input methods such as grip buttons, a touchpad, and a trigger. In order to be detected by the base station, the ring of the controller has in total 24 infrared sensors.

The headset device contains two screens (one per eye), each with a display resolution of 1080x1200 pixels, a field of view of 110 degrees and a refresh rate of 90 Hz. For safety reasons it includes a front-facing camera that allow the user to observe their surroundings without having to remove their headset (by clicking twice the menu button). Another feature is the capability of identifying any static or moving objects in a room, alerting the user by displaying a wall in the virtual world so that the user won't collide with any real-world obstacles.

For the development of the VR application of this thesis, the HTC Vive was chosen due to being one of the most developed VR systems commercially available and due to the availability at the Faculty of Engineering of University of Porto.

Fig 4.5 –The main components of the HTC Vive: headset, physical controllers, and base stations [62]

4.3. PROJECT MODELING – CONSIDERATIONS TOWARDS BIM VS GAME ENGINE

The modelling of the project it is something that requires a considerable time and effort, so it is desirable to utilize the most efficient way in order to increase productivity. BIM software such as Revit allows the user to model parametrized through many features that aides the user to achieve the desired design. For example, to design a simple wall, it is necessary to click on the wall button and select the start point and the end point that the wall will be automatically created. Since much of the element information is parameterized, it is easy to change features such as height, floor, and other parameters after its creation. Revit as well has integrated many predefined aspects that help to model a building within some constructability rules. For example, a ceiling it is attached primarily to the walls below or the insertion of a window can only be made if it is inserted in a wall, and many others.

There are a few different types of game engines. Some game engines are more code dependable allowing the developer to design their project more freely but at the cost of requiring programming more things. The Unity software used in this work, however, has a whole set of tools that allows the developer to organize their project through a point-and-click style interface, that presents a practical and customizable user interface. The unity hierarchy works with *GameObjects* that are diverse, and it can vary from being a 3D object from lightning effects. To be able to create a wall with an empty space in the middle intended for the window, as it is possible to see in figure 4.6 for example, the simplest way to do that it is to create four 3D game objects cubes (two for the sides, one to be at the top and one to be at the bottom), resize and reposition each one.

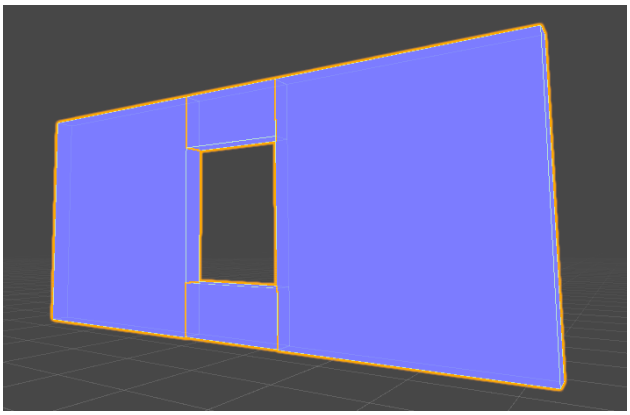


Fig 4.6 – Example of a wall in Unity

Because of the difference of efforts in modelling, it is noticeable that building elements such as walls, windows, ceilings, railings, stairs, etc. are created more efficiently by using Revit due to its predefined CAD components optimized for construction.

Many of the AEC companies have already adopted BIM software, as their software to develop their projects. Most of these BIM projects are designed in 3D models. Through Revit, for example, it is possible to export 3D files such as FBX(FilmBoX) and OBJ (Object File) that are supported to be imported by Unity as game objects. These format files allow interoperability between the two of them while preserving the geometry of the model[60]. By importing an FBX file to Unity, it helps to save a lot of modelling time, especially if the desirable VR project it has many objects to be modelled.

Analyzing the process of insert new elements inside the Unity Scene it is possible to conclude that there are basically three options:

- Download objects from the asset store - There is an extensive options range of objects. From ultra-detailed objects to very simple objects as well. There is a considerable amount of good-looking free assets to be downloaded.
- Import FBX or other 3D types of files to the project - There is an extensive offer of 3D objects on the internet, however, it needs to be considered if it is optimized for VR.
- Create the object with the Unity tools - If the project designer has no experience, this option remains only to create simple geometry objects.

Even though the importation of BIM objects with the FBX files represents in time saving, it should be considered that often the Revit files are not optimized to be used on a VR project. The Revit elements as it can be seen in figure 4.7 contain a complex hierarchy with categories, families, types and instances. Since inside Revit, these objects are commonly parametrized through various parameters, this results on a lot of geometrical information. For example, one instance of a window. Frequently at BIM projects, the family of windows are designed to be as much customizable as possible, with the objective of changing a few parameters (such as height, width, frame material, hinge, casing, shutter, and many others) to create various types of windows without having the need to create a new family for each new project. Even though this is a useful feature for Revit, considering in the VR context that the main goal it is simply of representation, it results in a series of points and polygons that are exported to Unity, that are not being used, it is either inside the geometry(mesh) or simply coinciding.

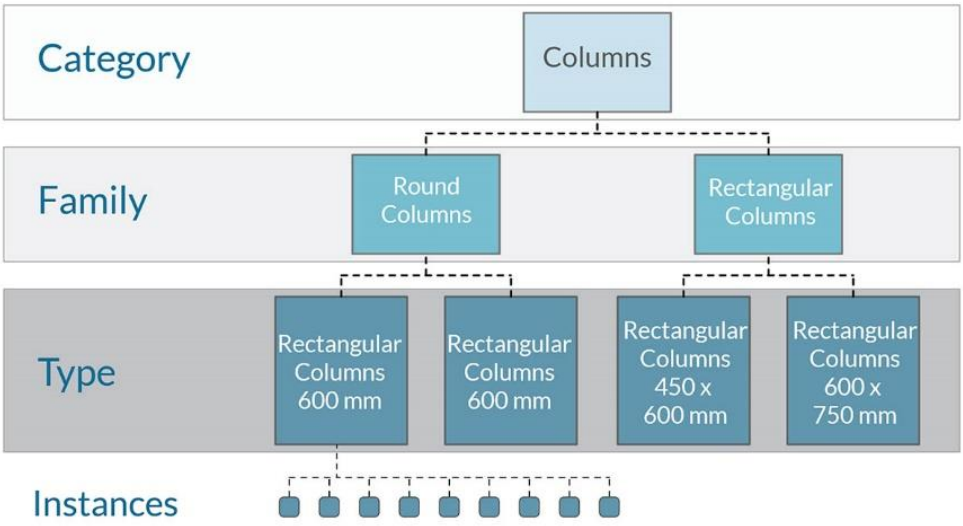


Fig 4.7 - Revit Hierarchy[58]

Another problem of importing the FBX files is the loss of object information. The FBX file created through Revit contains only the information about the geometry (*meshes*). Information like texture, material, and colors are lost through the process [60].

As it was discussed before in this work, the VR experience requires an excellent graphical performance to be user acceptable. The overall performance of the software application it is directly affected by the number of polygons that it is necessary to render (image processing), especially if the computing element of the hardware it is limited. Therefore, if the BIM project contains a considerable number of objects with a high quantity of complex elements, when these

elements are imported to Unity it can potentially cause a downgrade at the performance with lower frame-rate per second, enough to the point that it results in a not enjoyable VR experience.

Due to this problem towards the high polygon count of elements that are exported directly through Revit and the loss of material information, one solution it is to use an intermediary software such as the Autodesk Maya or Autodesk 3DS Max that can be used to optimize this process. However, both of these software also have a high cost associated with it, for example the Autodesk 3DS Max, currently has a subscription of 1 year costing approximately 2000€ to a single computer [63], and by using it as an intermediary software it is necessary to add one additional step before inserting it into game engines, representing more time consuming at the process workflow.

By analyzing the project modelling situation, in summary, there are three main aspects that need to be considered for the project modeling that it changes significantly depending on the project specification: the complexity of elements, the specification requirement for the aesthetics, and the area planned to be designed. In conclusion, currently there is not an ideal solution that it is the most productive with every single project, each project has its own specifications and context that can highly influence on the final result, it can be either modeling everything on the Unity Editor or simply importing an entire building from Revit.

The higher are the specifications, it is more recommendable to use a specific software design such as the 3Ds Max. However, as it is going to be presented in the chapter 5 of this work. For VR projects that only incorporates one apartment, the workflow presented in the case study it is sufficient to present an effective final application.

4.4. USER INTERFACE AND INTERACTIVITY

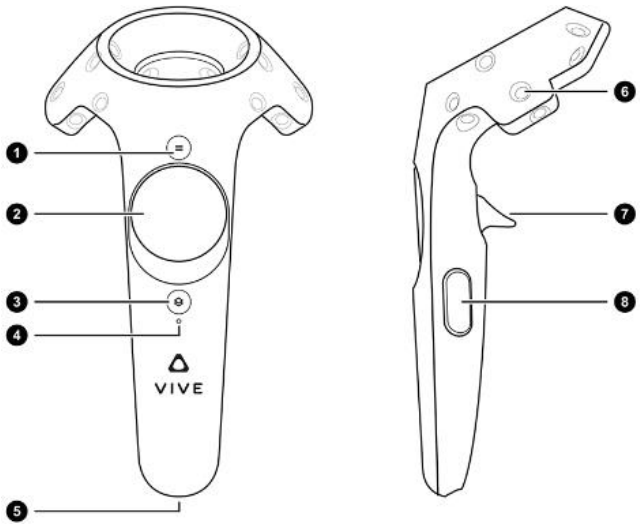
The user interface it is the space where humans interact with the machines. The goal of the user interface design it is to be as efficient and comfortable (user-friendly) as possible to be able to operate a machine to achieve the desired outputs. An uncomfortable user interface has adverse effects on user acceptance of the application[64].

In projects that do not involve VR, the information to the user it is commonly overlaid on top of the screen. However, in VR displaying information by overlaying it at the visual display, it is not a valid option because humans are unable to focus on objects that are too close to their eyes. As a result of it, designing a user interface in VR requires a different approach [65].

4.4.1. USER INPUT DESIGN CONSIDERATIONS

To develop an interactive GUI, with the hardware available for this, there were primarily three different options of input to be considered: physical controllers, the reticle and speech recognition.

The first one, the controllers offer functionality to the user inputs, the HTC Vive controller has a series of different input options such as trackpad, menu button, system button, trigger and a grip button as it is shown in figure 4.8. This provides the developer a good number of functionalities that can be integrated to the user to interact with the virtual world. For the proposed application the physical controllers are used for most of the interactions. The physical controller allows the user to simply point the controller and interact with the objects or GUI elements by pressing either the trigger or the grip button.



Number	Description
1	Menu Button
2	Trackpad
3	System button
4	Status Light
5	Micro-USB port
6	Tracking sensor
7	Trigger
8	Grip button

Fig 4.8 – Input mapping for the HTC Vive controller [66].

Following the controllers, the reticle, incorporates a small circle at the center of a user's field of vision. Allowing someone to use the gaze to interact with the surroundings, it is a natural and intuitive way to interact with the environment. To be able to 'activate' the interaction there a few alternatives, such as clicking on a physical button at the HMD or controller, or a 'look and lock' interaction (Center the vision in the object and look and 'lock' at it with a time delay to confirm the user intention). This type of interaction it is mostly used by low-cost VR devices[67]. At the proposed collaborative application, it could be potentially used, for example, to maximize or open a panel with information by simply looking at it. However, since the HTC Vive it has two physical controllers that has already a series of different options, this type of input it was ignored.

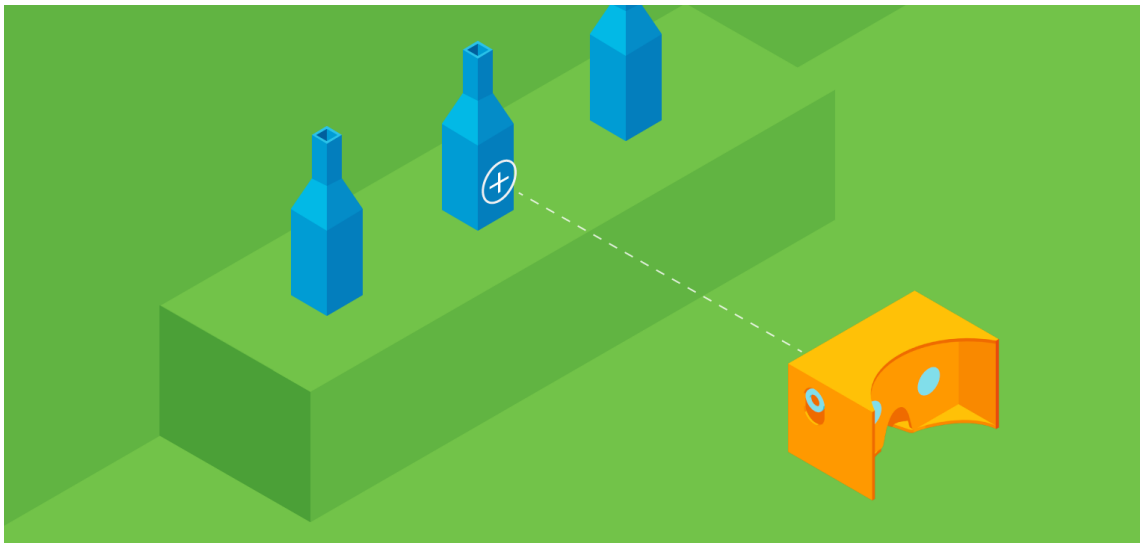


Fig 4.9 The reticle input [68]

The third one, speech recognition, is a natural and practical input that allows the user to say something and the computer respond to it. The technology to allow this input has evolved a lot recently through the development of virtual assistants such as Microsoft Cortana and Google Assistant[69], for the developed application, the speech recognition it is used to allow the user to easily insert commentaries to the scene.

4.4.2. INTERACTIVITY WITH THE VIRTUAL WORLD

To be able to interact with the virtual world, it was created a radial menu shown in figure 4.10 that it is activated by touching the touchpad of the Vive controller, this allows the user an easy way to access to the tools developed for the application such as commenting, ruler, interaction and opening the report panel. The user needs to only slide his finger towards the option desired.

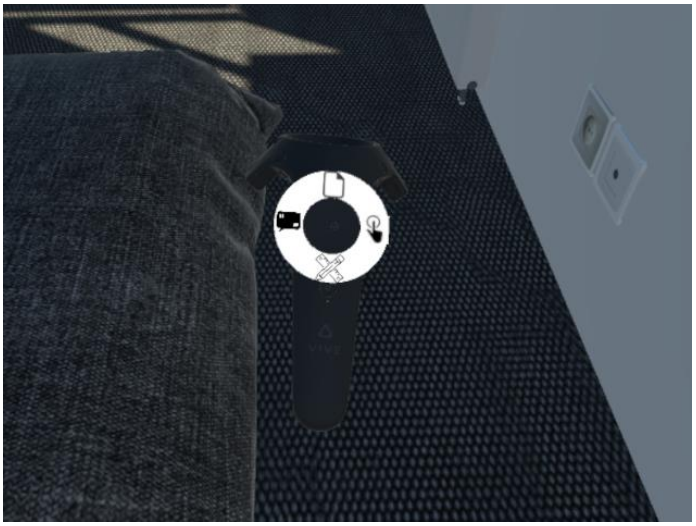


Fig 4.10 – Radial menu

Notably, one thing that it is crucial concerning the user interface of VR application it is that the information cannot be overlaid on the user display, due to the lack of human eyes capacity to focus on close objects. For example, if someone has something written at their hand and try to read by putting the hand right on front of the eye, it is noticeable that the human cannot focus on really close objects(the vision gets blurry). As a result of this, the GUI panels with the information to be interacted with at the VR are displayed through space within a comfortable distance, as the figure 4.11 shows, it is preferable to have the information at least of 0,5 meters away from the user.

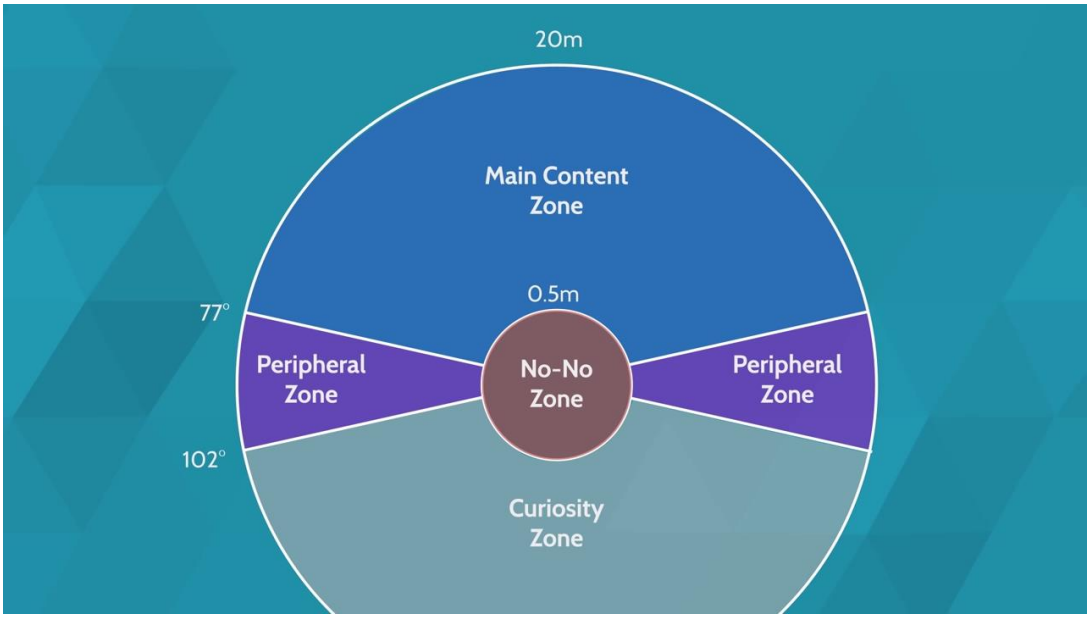


Fig 4.11 – Desirable interface position of information[70].

4.4.2.1 Locomotion

A critical element of VR it is that the participant does not feel VR sickness. As discussed in chapter 2, the VR sickness it is caused mainly due to the sensory conflict between the perception of self-motion and the vestibular system along with achieving a comfortable frame rate [71].

In order to prevent the user from feeling sick, it is important that the virtual world responds correctly to the user motion. For example, if the participant moves their head to the right, the avatar from the virtual world also moves his head to the right with the least delay as possible.

The tracking sensors from the HTC Vive hardware allow a reliable response to the user's movements because it can track the user movement and replicate it in the virtual world. However, most of the times the real-world environment it is smaller than the virtual environment. Due to that if the user continuously walks in real life, eventually he will face an obstacle such as a wall or a table and potentially harm himself or damage an object.

In the application, it is necessary that the user can move through the different rooms in order that he can evaluate the space. To be able to locomote in the virtual world, one option it is the user using the trackpad from the Vive controller to use it as a joystick, to click on a direction and move towards it, however this type of movement cause VR sickness due to the fact that the participant it is steady while his avatar is moving[61]. Another solution would be not allowing the user to move his avatar into the virtual world. There would be hotspots in which the user can select and can only visualize by rotating his vision. However, this is consequently limited because it does not give the user the capacity to inspect details.

The option developed to the proposed application it is through teleportation. The user points the controller to the desired location, and by pressing the trigger button, he teleports to the desired location as it can be seen in the Figure 4.12. The teleport gives the freedom to the participant to check minor details due to the immersion, while at the same time it significantly reduces the side effects the VR sickness because the user does not have a sensory conflict between what he sees and what his body is feeling. This type of mobility also has proven as effective if the virtual world it is immense. For example, on occasions that the user it is required to move through a significant area that is far from each other, by simply teleporting the participant it is transported immediately to the desired place, saving time.

Another aspect developed was the locomotion between rooms. The user is not allowed to pass through walls or doors in order that the immersive factor it is preserved because the intention it is the application to be a simulation of the real-world physics. For opening a door, the user has to grab the door handle with the grip button as similar he would do in real life, and mimic the movement of the door opening. This is an intuitive way of opening doors that do not require any form of explanation.

Fig 4.12- Image of the teleportation - before and after.

4.4.2.2 Interactive button

The interactive buttons refer to some relevant elements that must be taken into account and the information associated with it is relevant such as the supplier, material, dimensions, and localization. Fixed equipment such as the flooring, tiling, sanitary, lighting, kitchen countertop and other elements are all decisions that influence the usage of the space in general, in addition to it to make changes towards them require a considerable amount of expenses and effort after it is already built. Therefore, it is crucial that every element choice it is validated and in case that there is a problem, create feedback of why the object it is not validated. These buttons also help the users by guiding the design review with settled points of interest that must be considered.

Since the digital model works as a virtual prototype, the elements also can be analyzed aesthetically. For example, the ceramic tiles designed in the kitchen can evaluate if the dimensions and material have the intended look, otherwise, a commentary saying that they need to be changed can be created.

The creation of the interactive buttons since it is a task that it is required to be done repetitively, it was developed to be created through a database, that holds all the information towards the equipment, this is beneficial because it enhances the organization of the information and saves time on the insertion of information by being more efficient. As it is possible to see at the figure 4.13, through the script *Custom List*, it is possible to add and manage the information towards the equipment such as the name, localization, supplier, material, and dimensions. The associated GO it is responsible for managing which object it is going to be seen through the panel camera. In this example shown in the figure, it is related to the kitchen countertop.

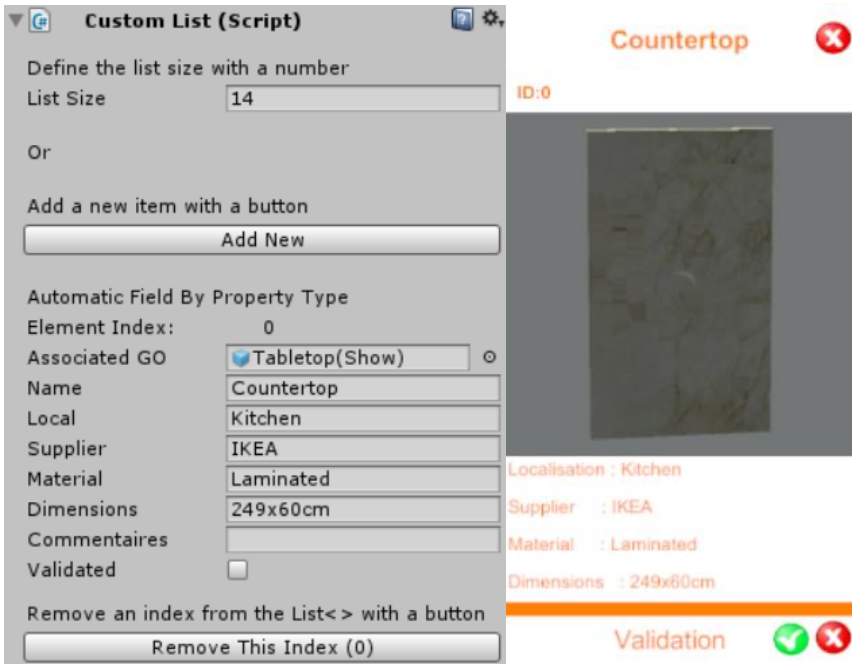


Fig 4.13 – Image of the database at the Unity Inspector in correspondence with the element in the scene

Whenever the user clicks on the interactive button, a panel with the information appears and the content of this panel it is managed by one *game controller* associated with the clicked button canvas. As it can be seen at the figure 4.14, the *game controller* has the script *Interactive Panel Script* that contains one correspondent identification, by clicking at one interactive button it changes the active identification of the information panel and updates its information and camera.

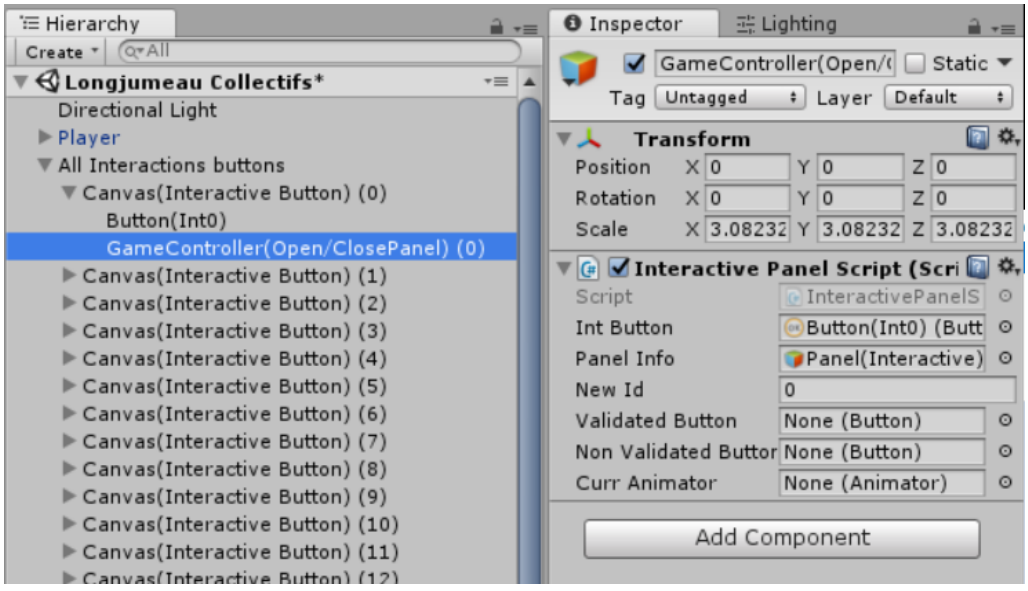


Fig 4.14 – Image of the game controller attached to the canvas of the button

By confirming or not the validation of the equipment, it activates one animation that changes the color of the button. If the validation it is confirmed the button turns into green. Similarly, if the validation it is not confirmed the button turns into red as it can be seen in figure 4.15. This helps the participant to track down which elements he already interacted with it or not.

Fig 4.15 – All three states of the button – neutral, non validated, validated.

If the equipment it is not validated, it is relevant to register the reason why in order that changes can be made to solve this problem. However, one problem of being immersed in VR it is that the physical controllers are not optimized to be able to type efficiently (usually it is done by a virtual keyboard, but with physical controllers it is unwieldy). One solution developed for this application was using one of the most intuitive ways of communicating, by simply talking. The application uses the efficient Windows Speech Recognition services, in which it is currently optimized for cloud computing that can accurately recognize what the user is saying[69]. At the commentary panel, as it is shown it at the figure 4.16, to start the dictation the user needs to click on the record button and starts speaking, the system listens and write down what it understood from it. If the user it is satisfied with the provided text, he then can proceed and save the commentary, otherwise, he can reinitiate the procedure by clicking another time on the record button.

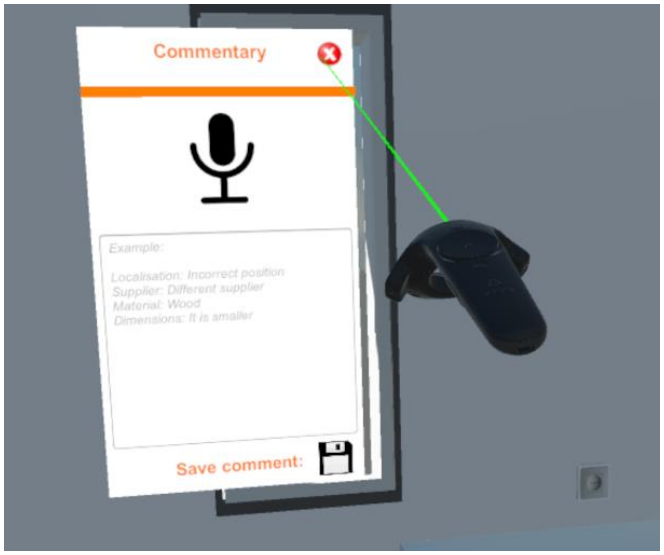


Fig 4.16 – Image of the commentary panel.

4.4.2.3 Ruler tool

The ability to measure distances in the virtual environment it is useful to verify different things inside the virtual world. It can be used to verify for example if the scale of the objects is correct, the light switches are positioned at the correct elevation, the height of the countertop it is appropriate, the minimum distance towards a hazardous object, check accessibility issues such as the minimum width of corridors, etc.

This is achieved by the participant selecting through the radial menu the ruler icon and afterwards pointing the Vive controller and pressing the trigger button two times, resulting in two ray casts

that are shoot and store the position of it to where the pointer hits. The software draws a line between these two points and calculates the distance between each other, allowing the user to verify the distances as it can be seen in figure 4.17.

Fig 4.17 – Example of the measurer of the positioning of a light switch.

4.4.2.4 Commentary Tool

This feature is a tool that allows the user to create markups into the project. This is useful to point out eventually problems related to the project, such as defects, personal preferences, etc. The user is required to select from the radial menu the commentary tool and point to the position of the desired markup to be placed and click the trigger button of the Vive controller. The user it is presented a panel similar when fixed equipment it is not validated as shown in figure 4.19 and one canvas button (shown in figure 4.18) it is created inside the application that can be used afterwards to be able to access and change the information of the markup. To be able to insert the commentary it is also used the voice recognition from the Windows Speech recognition services. The commentary is saved into a database that afterwards it can be later exported to the report panel through the extraction button.

Fig 4.18 – Example of one commentary created

4.4.2.5 Report panel

One important aspect to be considered it is the capability of the user to access the information of the validation of all elements in order to recapitulate and confirm the decision made while being immersed in the application. To be able to access this panel the participant needs to choose from the radial menu the icon of the folder (placed in the north direction, as seen in figure 4.10). This panel displays the information of all the fixed elements prepared for the design review as it is shown in figure 4.19 and automatically updates according to the changes made during the VR experience by accessing the database of the elements.

The report panel also contains the extraction button, which is responsible for exporting the database of all elements and commentaries created to be imported back into the BIM model project.

Fig 4.19 – Report panel

4.5. THE CONNECTION BETWEEN THE VR PROJECT AND THE BIM MODEL.

Since the application project uses mainly two different software, the VR application Unity, and the BIM software Revit, it is necessary to create an intermediary logfile (.txt based) that it is going to be used by both of them to establish a flux of information as it is shown at figure 4.20. The log file it is created by Unity, and then it is imported into Revit by using the plugin Dynamo. By clicking on the extraction button at the report panel, the VR application writes a log file that has a simple comma-separated-value(CSV) structure. The creation of a log file it is beneficial because it facilitates the flux of information and it is created automatically, the participants of the design review it is not required to write anything on paper or fill in an excel sheet.

Into the log file, there are two different groups of information as it is shown in figure 4.20 and 4.21. the fixed equipment and the new commentaries. The first concerns the fixed equipment prepared for the design review containing data such as the ID, the associated information and the

coordinates of it. The second group it is formed by the new commentaries(markups), and it has the data of the ID, the commentary associated with it and the coordinates of it.

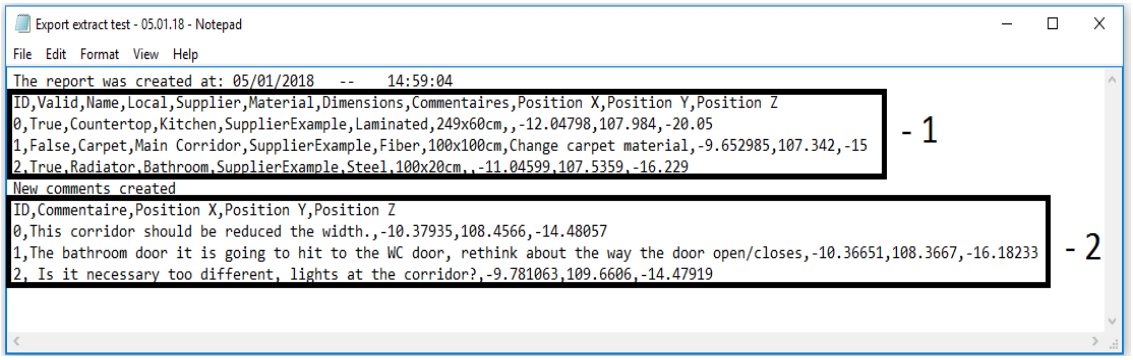


Fig 4.20- Example of one log file with the two groups highlighted.

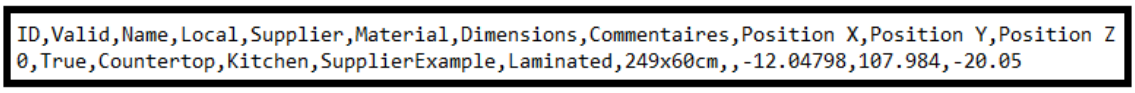


Fig. 4.21 – Zoomed information of the example

To be able to represent the information in Revit, it was created two new Revit families that are placed at the coordinates received from the log file, in addition with eight Revit parameters that are utilized as the holders of the information received from the VR application. These elements have similar appearances as the ones used to indicate the buttons in the VR application in order to create a graphical identity of the elements.

Fig 4.22 –Image of the Revit families created, and the parameters associated with it.

Similar to the logic on the color changing at the VR application. It was created a graphic override(filter) in Revit that changes the color of the families according to the validation or not of the equipment. This it is intended to work as a problem tracker for the project stakeholders working in the Revit file. The color red it is a commonly used as the color to indicate a problem, because of that in the filter the changes the color of the element to red when the parameter *UNT_Valid* it is equal as false (as shown in figure 4.23), likewise if the element it is validated the color changes to green. Therefore, by using different colors, the users can track if the problems were already solved or not inside the BIM model.

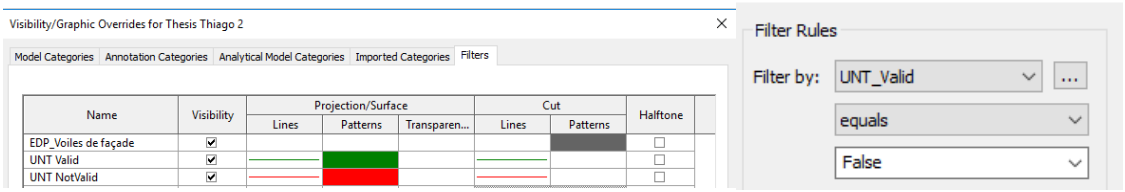


Fig 4.23- Visibility filter to change the colors.

The comma separated value has a straightforward structure that allows the Dynamo code to be able to read it without the need of programming much code, to read it is required only three nodes as it is shown in the figure 4.24. Each line of the notepad file corresponds to one list in Dynamo, and this facilitates the management of information inside the plugin.

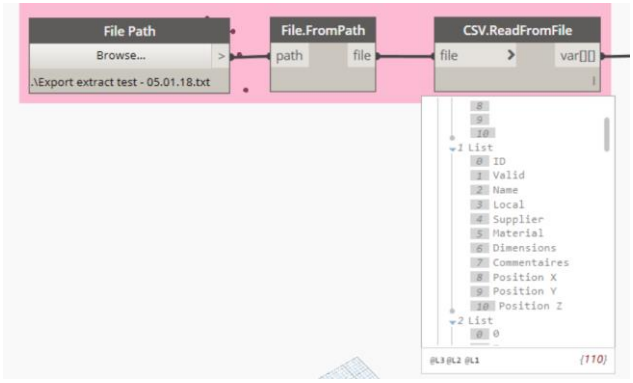


Fig 4.24 – Nodes responsible for reading the .txt file into the Dynamo structure

The logic created for importing information with the Dynamo code it is simple, and it is going to be briefly explained. The code has four primary group of nodes that are separated by different colors as it is possible to see the figure 4.25. The first one (colored pink) it is responsible for reading the CSV file created by Unity and transforming the values into lists to be used in Dynamo. Following that, there are 3 group of nodes colored with orange that manages the values received from the CSV file, by removing null values and separating the lists into two (one for the fixed elements and one for the new commentaries). The third group it is responsible for creating the Revit elements into the BIM model and place it accordingly to the coordinates received. Finally, the fourth group it is responsible for setting up all the parameters with the correspondent values in the BIM model.

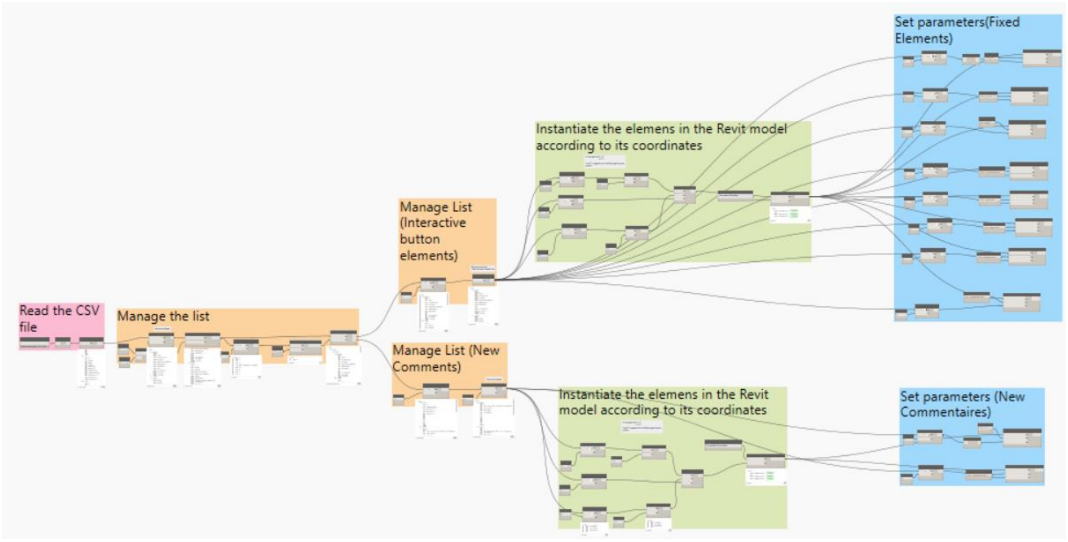


Fig 4.25 - General view of the Dynamo code

5

CASE STUDY

5.1. CASE DESCRIPTION

The study case presented in this thesis concerns a construction project that was under construction in October 2017, of a social housing building at the municipality of Longjumeau located in the south-east of Paris.

This construction work has a total business volume of around 16 million euros and refers to two buildings with a total of 5 floors(R+4), one designated for student housing with a total of 130 student accommodations and another one designated for social housing.

The contractor is a property development company situated in the Parisian region called LinkCity Ile-de-France that belongs to the Bouygues Construction group. This company promotes different types of real estate development throughout the Parisian region. The responsible firm for the architectural project it is Domela Architectes.

Table 5.1 – Characteristics of the project Longjumeau Rue du Verdun lot Yb.

Designation	Characteristics
Main contractor	Linkcity Ile-de-France
Architectonical project	Domela Architectes
Business volume	16M€
Upper floors	R/C + 4
Social Housing	35
Student Housing	130

5.2. MODEL OF THE PROJECT

In the occasions that the digital model it isn't created by the client or architect firm, the task of modeling usually is taken care by either the engineering firm or by outsourcing to BIM modelers. It is necessary to model with rigorous standards in order that all necessary studies can be made and that it can be used as a good source of documentation for the project. In the case presented in this thesis, the BIM model(visible in figure 5.1) was created by engineers of the department of engineering of the construction company Bouygues Bâtiment Ile-de-France Habitat Social.

Fig 5.1 - Image of the building designated for social housing of the Longjumeau project.

The modeling of this project was based on CAD plans from the preliminary technical studies (structure and thermal) and detailed documents provided by the architect company responsible for the project.

It is important to emphasize that the model used was developed in the procurement phase. For that reason, the LOD of the project it is not at the highest standards. Most of the elements of the digital model have the primary objective as a source of documentation in which it is not required to have the elements with a precise geometry(realistic).

Since at the case study this model it is intended to be used for the validation of the application, it was selected just one of the apartments at the social housing building to be representative for the whole building.

5.3. METHODOLOGY

In order to implement the interactive application, it is necessary to consider a number of variables that influence on the workflow of it, such as availability of BIM model, size of the project, the level of detail required for it, and other particularities that can differentiate from each project. For example, if the construction project does not have a BIM model, the whole part of the Revit it can be ignored, and the VR application can be developed by using solely the Unity software at the setback of needing to model more.

The workflow sequence developed for this case study is shown at figure 5.2, in summary, the geometry of the BIM model elements it is imported, afterwards adapted to be used in virtual reality, subsequently, it is implemented the interactivity elements and the collaborative file it is created. With the file ready, the design review occurs and later the feedback it is inserted back into the BIM model. Taking into consideration that often the construction companies already

develop their projects with a BIM, the workflow presented in this section it is a practical workflow that can be potentially used for a wide range of projects.

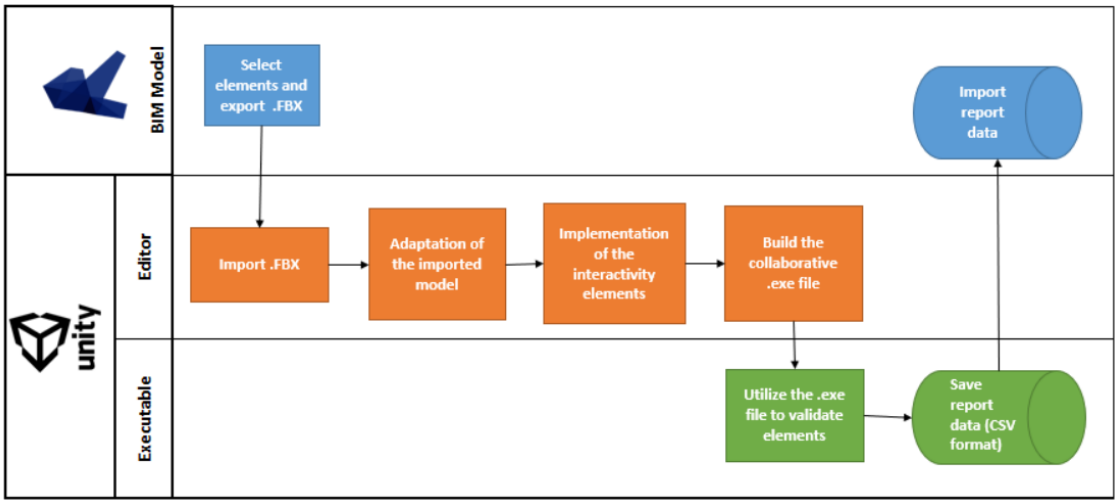


Fig 5.2 Flowchart of the methodology using the VR application

5.3.1. INTEROPERABILITY WITH THE FBX FILE

The first action to start to prepare the application it is to select the area of the desirable elements. Commonly in most AEC companies, a considerable number of the components used in the BIM model are not optimized to be used in VR. As it was discussed previously in section 4.3 of this thesis, the main reasons for this are either due to the high complex geometry of the elements such as furniture with a complex geometry, that can potentially compromise the refresh rate or due to the loss of material through the export that makes it unusable at the VR project.

Revit only takes into consideration at the FBX file export, the elements that are shown at the selected 3D view. Therefore, one way to optimize the export it is to select the elements that are going to be transferred by hiding all the unwanted elements in Revit as it is possible to see in figure 5.3.

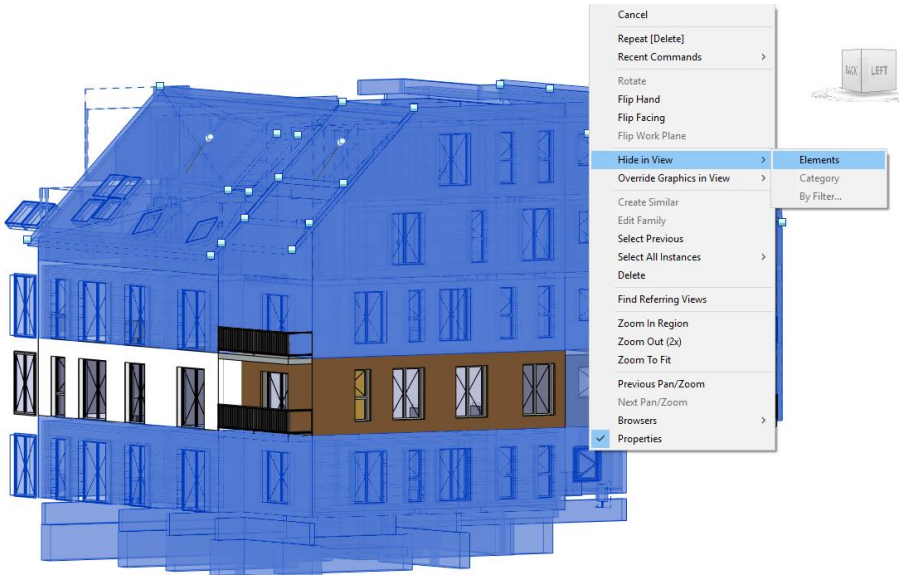


Fig 5.3 – Hiding elements in Revit

At this BIM model in particular, after testing the FBX export with the project elements it was concluded that the most productive path, considering these project specifications, besides from hiding all the other levels and apartments was also to hide all furniture, doors, and windows due to the loss of material that made them unusable in Unity. Resulting in a model with basically consisting of walls, railings, and floors as it is shown in figure 5.4. In summary, the process of exporting at Revit it is made by following the steps: *Select 3D View → Hide all unwanted elements → Revit Menu → Export FBX.*

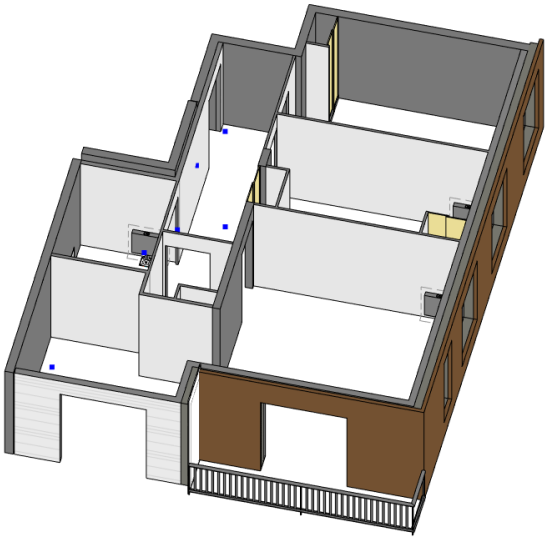


Fig 5.4 – Image of the featured apartment with all unwanted elements hidden in Revit

With the FBX file created, the following step it is to import all the geometry into the Unity editor. This is done by simply dragging the FBX file into the editor screen. It is crucial that the imported geometry is not repositioned because the coordinates of the elements from the Revit file is the same in the Unity and by repositioning the geometry this information it is lost.

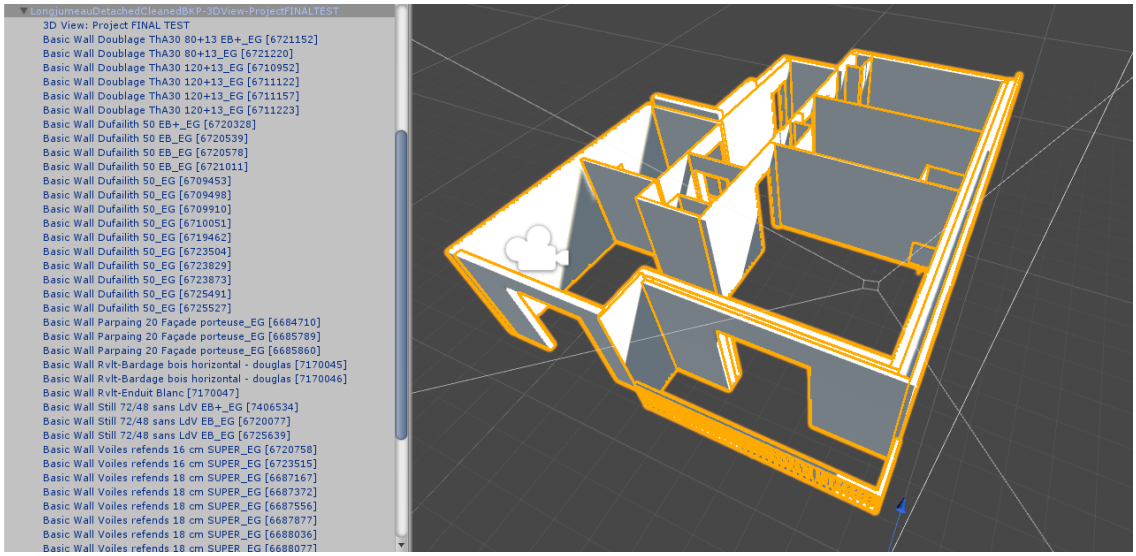


Fig 5.5 – Imported FBX to the Unity editor.

5.3.2. ADAPTATION OF THE IMPORTED MODEL

Next step it is to adapt the model by adding all the desirable details. Since the project contains basically the geometry of the walls, it is relevant to add all the other objects of an apartment in order that the project model has an acceptable appearance.

There are basically 3 options, as discussed in section 4.3, to add elements to the VR scenery: implementing objects from the Unity asset store, import FBX or another 3D format of files to the project, and create the object with Unity tools. The majority of objects used in this case study was downloaded from the Unity asset store, with a few exceptions such as the doors and windows. To organize the objects, one good practice it is to create a library of objects (one scene with all the elements organized) as the one created for this case study as shown in figure 5.6 in order to facilitate the reusability of the objects in future projects.

It is noticeable that by each project the VR designer develops, it increases their library of objects. By having a developed library, the reusability of the objects enhances the productivity significantly. The library can contain a range of elements such as furniture, doors, windows, sanitary objects, lighting, materials, texture, colors, MEP systems, etc.

Fig 5.6 – Example of the library of objects used in the project

Unity has a set of tools (Figure 5.7) that allow the user to easily arrange the elements at the correct position by easily moving the object, rotating, changing the scale and changing coordinates of the element. With a good variety of elements integrated on a library, taking this workflow as reference, most of the work required in the VR scenery it is done by placing the elements of the library and eventually adapting it to the scenery by changing scale, rotation or position.

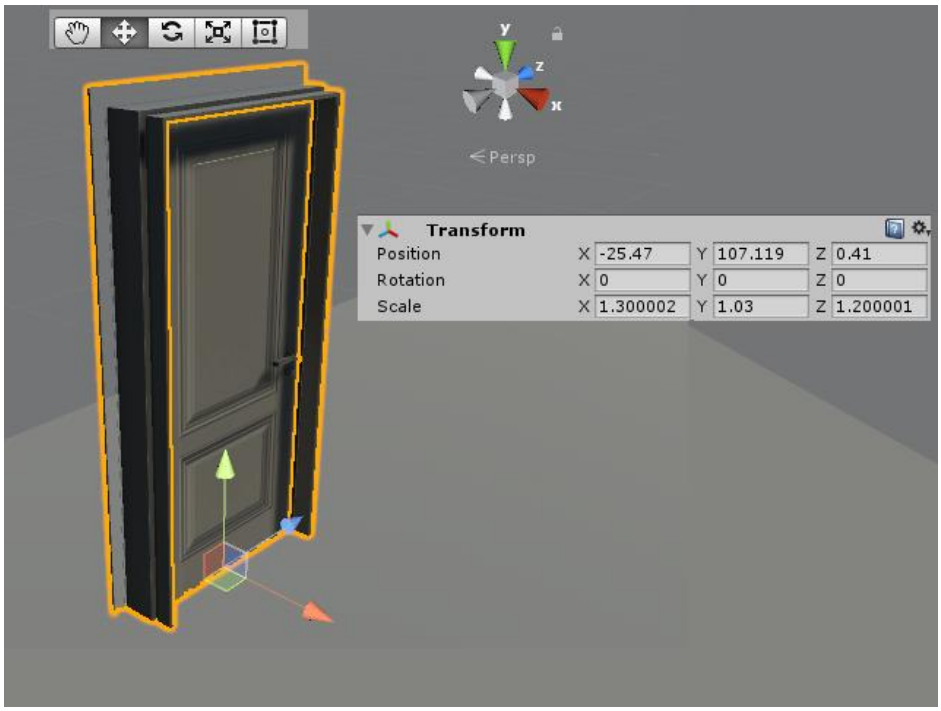


Fig 5.7 - Positioning tools options

The BIM model imported in this case study didn't have electrical fixtures such as the lightning spots, light switches, electrical plugs, Ethernet output, phone, and cable TV. Since this is an important element that can potentially raise problems with the positioning(ergonomics). It was used an electrical CAD plan from the preliminary phases of the study of the building as a base plan to position the electrical fixtures in the VR scene.

5.3.3. CREATION OF THE INTERACTIVITY ELEMENTS

As soon as the VR model it is ready. It is necessary to place the interactive buttons with the intended fixed equipment. This can vary significantly according to the VR project requirements and objectives. In this case study, it was decided that the fixed elements most relevant to contain an interactive button are: tiling, flooring materials, heating elements(radiators), sanitary elements (sinks, bathtubs, toilets, etc.) and kitchen countertops. This intentionally left elements such as furniture as being representative objects only. Even though there are elements which can be relevant to the project that does not have an interactive button (such as the electrical fixtures), it is still possible to create markups while running the application indicating a problem or a design preference.

To increase the productivity, it was created a number of pre-fabricated elements that allow the designer to simply drag and drop these elements to include them at the scene. One of these pre-fabricated elements are the interactive buttons, after placing the interactive buttons at all desired elements, it is necessary to fill in the information of the equipment as briefly explained in section 4.4.2.2.

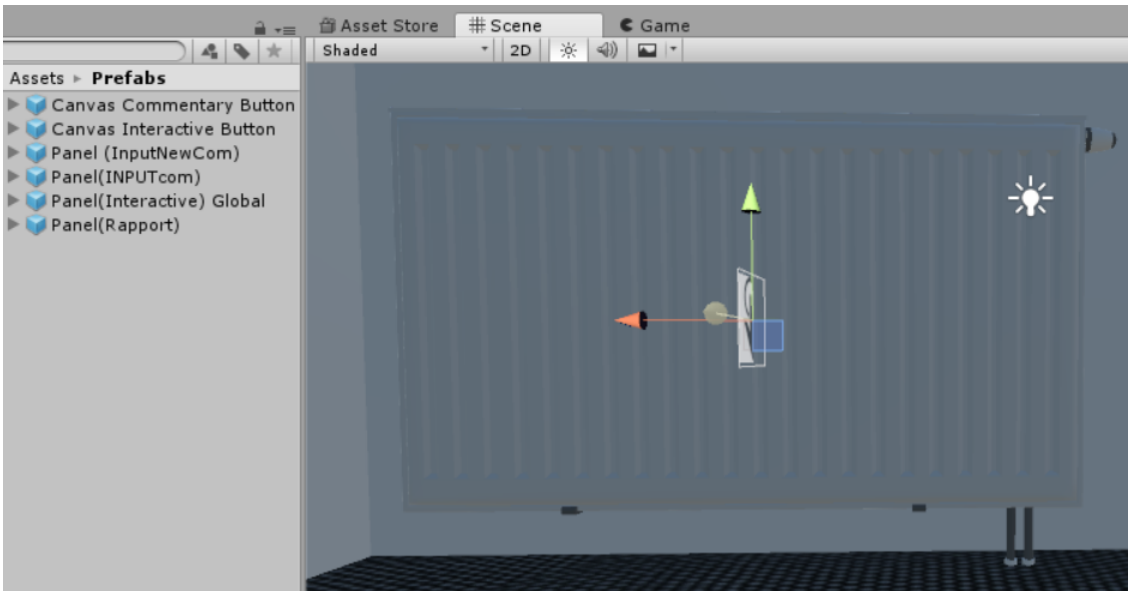


Fig 5.8 – Placing prefabs

5.3.4. TESTING AND CREATING THE SHAREABLE APPLICATION

With all the elements of the VR scene ready, it is important that all the buttons of the scenes and inputs are tested in order that the final application works as it is intended to work. The Unity software has a console that allows the designer to test(debug) if some features are working properly. For most of the scripts that were developed for this case study (presented in the attachments of this thesis), it was implemented debug messages that allow the developer to have better control and understanding of how the VR scene it is functioning in real-time. As it is possible to see in figure 5.9, for example, whenever the user at the application clicks on an interactive button and changes the values, it is displayed a message of the values changing in order that the developer can track down possible errors.

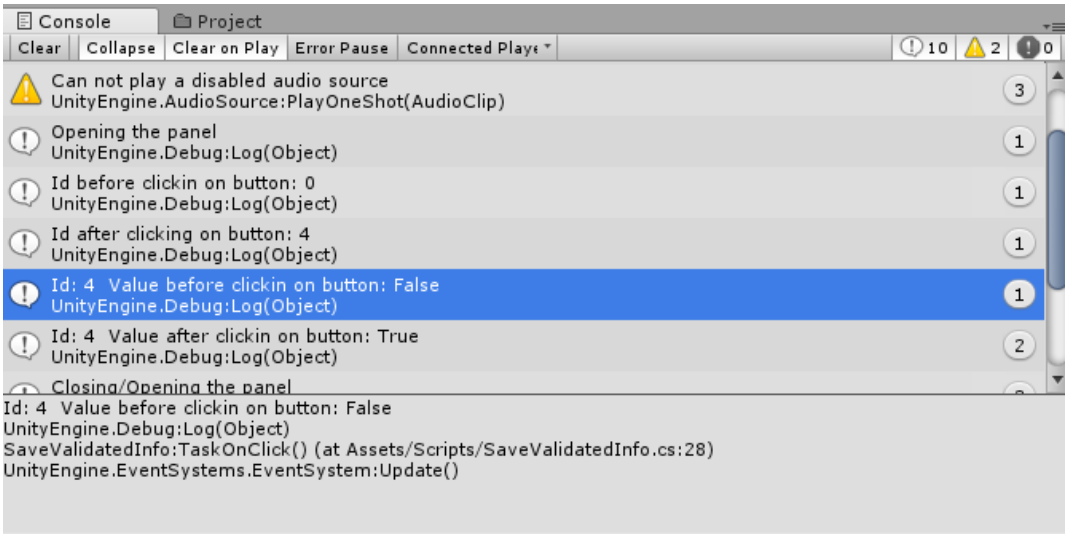


Fig 5.9 - Console image of the debugging process

Before creating the final executable file, for the proposed application it is crucial to test a number of conditions such as: Test if whenever the equipment is validated, the color of the button changes correctly.

- Check if the VR radial menus are working properly.
- Check if the tools such as the ruler and creating commentaries are working perfectly
- Test if the speech recognition from the commentary is working accurately.
- Check if the report panel it is updating and functioning accordingly while the application it is running.
- Test if the extraction button (creation of the log file) it is working correctly.

With the application running as designed, it is required at the following step to create the executable file. To create the file, it is necessary to choose the scenes that are going to be in the application and click on the build icon as shown in figure 5.10. Click on every single interactive button to see if they open a panel with the correct corresponding information.

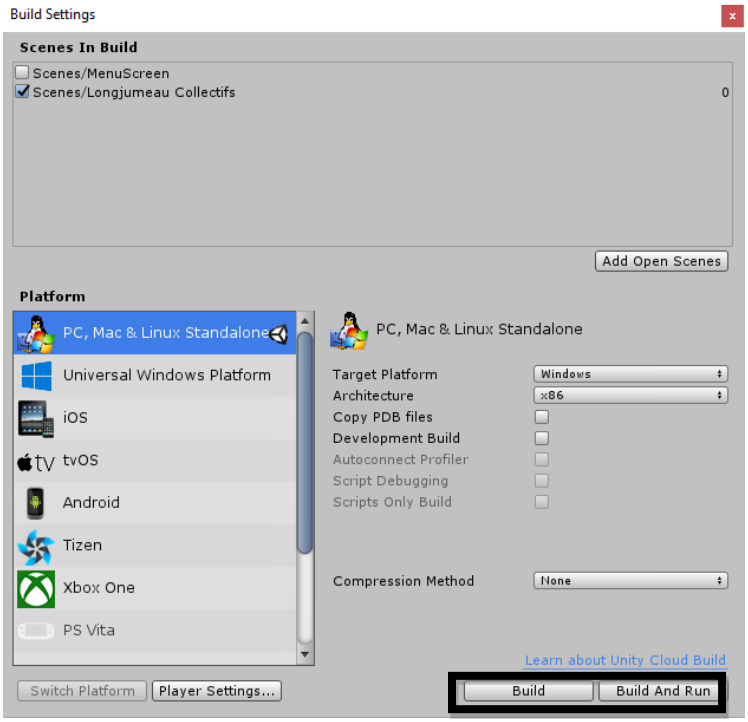


Fig 5.10 – Building up the game

5.3.5. IMPORTING INFORMATION REVIT

Now that the collaboration application file it is ready to be used, the application must be used either individually or by a group of the project stakeholders to analyze the VR model and validate the decisions of the fixed equipment and create commentaries towards the project.

In order to prove the methodology for this thesis, the application was hypothetically used, and all the interactive buttons were validated or not, with the addition of a few commentaries. The log file was created and can be seen in the figure 5.11.

Export extract - 07.01 - Notepad

File Edit Format View Help

```
The report was created at: 07/01/2018   --    17:59:04
ID,Valid,Name,Local,Supplier,Material,Dimensions,Commentaires,Position X,Position Y,Position Z
0,True,Countertop,Kitchen,IKEA,Laminated,249x60cm,,-12.04798,107.984,-20.05
1,False,Carpet,Main Corridor,Example,Fiber,100x100cm,The carpet color it is too ugly, evaluate other materials,-9.652985,107.342,-15
2,True,Radiator,Bathroom,Finimetal,Steel,100x20cm,,-11.04599,107.5359,-16.229
3,True,Tile Floor,Bathroom,IKEA,Ceramic tiles,20x20cm,,-11.21896,107.348,-17.241
4,True,Radiator,Master Bedroom,Finimetal,Steel,100x20cm,,-5.244965,107.633,-13.422
5,True,Tile Floor,WC,IKEA,Ceramic tiles,20x20cm,,-9.648987,107.3387,-17.72099
6,True,Toilet Seat,WC,Allia Prima,Porcelain,120x40cm,,-9.457001,107.885,-18.297
7,True,Sink,Bathroom,Allia Prima,Porcelain,110x70cm,,-11.80997,108.0685,-17.005
8,True,Bathtub,Bathroom,Allia Prima,Porcelain,White tiles 20x20cm,,-11.621,107.708,-17.946
9,True,Carpet,Single Bedroom,Example,Fiber,100x100cm,,-6.562988,107.3452,-14.657
10,True,Radiator,Living Room,Finimetal,Steel,110x20cm,,-5.378983,107.772,-16.614
11,True,Ceramic Tiles,Kitchen,Supplier tiles 1,Ceramic,20x20cm,,-11.24692,108.354,-18.66
12,True,Radiator,Single Bedroom,Finimetal,Steel,100x20cm,,-5.185982,107.484,-13.995
13,True,Radiator,Living room,Finimetal,Steel,100x20cm,,-4.915981,107.484,-20.495
New comments created
ID,Commentaire,Position X,Position Y,Position Z
0,This corridor should be reduced the width.,-10.37935,108.4566,-14.48057
1,The bathroom door it is going to hit to the WC door rethink about the way the door open/closes,-10.36651,108.3667,-16.18233
2, Is it necessary too different lights at the corridor?,-9.781063,109.6606,-14.47919
```

Fig 5.11 – Image of the final extract report

With the log file created, it was used the dynamo code explained in section 4.5 to import all the information from the CSV file into the BIM model. There were in total 17 generic models inserted in the Revit file through the dynamo code, in which 14 were fixed equipment and 3 were commentaries created as it is shown in the figure 5.12. The information is imported and can be used within the BIM model as a problem tracker. For example, in the figure 5.13 it is shown, one of the elements inserted into the Revit file pointing out that the carpet material should be changed. While the carpet material it is still not decided, the material will remain red due to the indication of the validation at the parameter *UNT_Valid* as false.

Fig 5.12 – Image of the before and after of the import through Dynamo code.

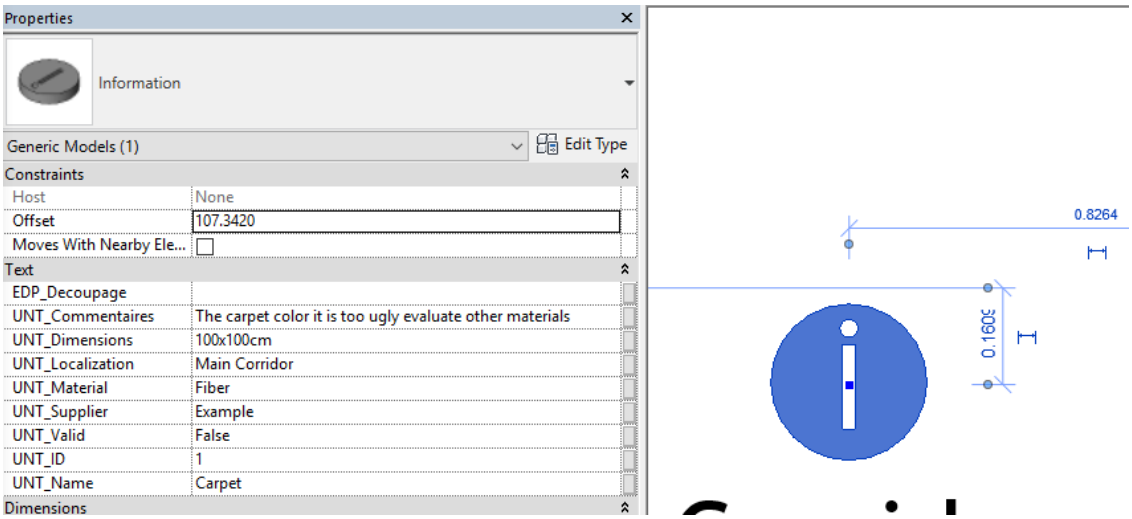


Fig 5.13 – Image of the element imported with the parameters

6

CONCLUSION

6.1. FINAL CONSIDERATIONS

At the end of this thesis, it is possible to conclude that the main proposal of this thesis to create an interactive, collaborative tool using VR was achieved.

The case study focused on a residential apartment helped conclude that the implementation of the application can be easily applied with little effort to any type of building. The VR model provided a reliable virtual prototype of the area of study that can be used as a collaboration tool due to its interactivity elements and shared with the project stakeholders due to its intuitive usability that does not require any previous formation of the software.

The main difficulties encountered throughout the thesis are featured in the following list:

- Create a database in Unity with the minimum effort for the developer;
- The lack of extensive academic research concerning the VR due to this relative novelty factor.
- Working on a still young medium for the AEC, that contains many specific requirements;
- Developing an effective user interface in VR due to its still in flux development concerning the hardware and the user experience overall;
- Achieving an efficient workflow to implement VR in a project building with a good-looking aspect with optimal refresh rate;

In summary, the medium of VR even though it still has a slow expansion into the professional level in the AEC industry, the tendency of its applicability will continuously increase within a significant number of AEC companies due to its easy adaptation with BIM models on a not-so-distance future, being used as either as a supportive medium or the medium itself that some aspects of the construction projects are developed.

6.2. FUTURE WORK

Being this work as an initial proposal for an immersive collaborative tool, there are naturally two clear paths to be considered as future research: The creation of a new VR application through a different software or the refinement of the application developed in this thesis. As such, the following list represents proposals that could be developed in future research:

- Optimize the programming codes and the workflow to be even more efficient;
- Develop the project modeling option to integrate with construction projects that are already completed by using technologies such as laser scanning;
- Upgrade the application to have new forms of interaction while immersed in VR such as: making 3D annotations, moving elements with the physical controller, create sections (to analyze constructability details); change the scale of the whole project(for example, the

participant can either analyze the project as a giant to have a general overview or being an ant to analyze minor details), integrate a fly mode, changing the position of the sun(to analyze natural lightning), 'virtual camera' tool to be able to take screenshots and send it to stakeholders;

- Integration of the time factor in the VR model, to be able to see and analyze the project throughout a timeline;
- Introduction of multi-user capabilities allowing more than one user to interact with the same project model at the same time;
- Explore the application capability towards facility management – allowing the user to intuitively manage and interact with building information while being immersed in a virtual model;
- Development of an application that integrates AR and VR at the same virtual model, with the AR tools to be used especially in the construction site, and VR tools to be used primarily by designers and consultants;
- Adapt the same workflow but integrate with different BIM software such as Bentley Architecture, ArchiCAD, Vectorworks, etc.

BIBLIOGRAPHY

[1] Olanrewaju, A., S.Y. Tan, and L.F. Kwan, *Roles of Communication on Performance of the Construction Sector.* Procedia Engineering, 2017. 196: p. 763-770.

[2] Craig, A.B., W.R. Sherman, and J.D. Will. Developing Virtual Reality Applications : Foundations of Effective Design. 2009, Boston: Morgan Kaufmann.

[3] *"Reality"*, in *Merriam-Webster.com*. https://www.merriam-webster.com/dictionary/gimmick, Accessed : Oct 2017.

[4] *"Virtual"*, in *Merriam-Webster.com*. https://www.merriam-webster.com/dictionary/virtual, Accessed : Oct 2017.

[5] Sherman, W.R. and A.B. Craig. Understanding Virtual Reality: Interface, Application, and Design 2003, San Francisco: Morgan Kaufmann.

[6] Reis, R., *Brazilian Old Kitchen.* 2015, https://ue4arch.com/brazilian-old-kitchen/. Acessed : Dec 2017.

[7] Thornson, C.A., B.F. Goldiez, and H. Le, *Predicting presence: Constructing the Tendency toward Presence Inventory.* International Journal of Human-Computer Studies, 2009. 67(1): p. 62-78.

[8] Laurel, B., *Computers as Theatre*. 1991, Menlo Park, Ca: Addison Wesley.

[9] Steuer, J., *Defining Virtual Reality: Dimensions Determining Telepresence*. Journal of Communication, 1992. 42(4): p. 73-93.

[10] Pereira, I., *Realidade Virtual.* 2013, http://web.ist.utl.pt/ist170613/. Acessed : Oct 2017.

[11] Heilig, M., *The father of virtual reality.* http://www.mortonheilig.com/. Accessed : Dec 2017.

[12] Virtual Reality Society, *History of Virtual Reality.* 2017, https://www.vrs.org.uk/virtual-reality/history.html. Accessed : Dec 2017.

[13] Nasa, *The virtual Interface Environment Workstation (VIEW).* 2014, https://www.nasa.gov/ames/spinoff/new_continent_of_ideas/. Accessed : Nov 2017.

[14] Human Interface Technology Laboratory, *Super Cockpit Program.* http://www.hitl.washington.edu/people/tfurness/supercockpit.html. Accessed : Nov 2017.

[15] Visbox, *VisCube C4-4K Datasheet.* 2017, http://www.visbox.com/products/cave/viscube-c4-4k/. Accessed : Dec 2017.

[16] Fast Company, *Unraveling the enigma of Nintendo's Virtual Boy, 20 years later.* 2015, https://www.fastcompany.com/3050016/unraveling-the-enigma-of-nintendos-virtual-boy-20-years-later. Accessed : Dec 2017.

[17] Oculus, *Oculus Rift: Step Into the Game.* 2012, https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game. Accessed : Oct 2017.

[18] The Verge, *Facebook buying Oculus VR for $2 billion.* 2014, https://www.theverge.com/2014/3/25/5547456/facebook-buying-oculus-for-2-billion/in/3631187. Accessed : Nov 2017.

[19] Gartner, *Hype Cycle.* 2017,https://www.gartner.com/technology/research/methodologies/hype-cycle.jsp. Accessed : Dec 2017.

[20] Gartner, *Top Trends in the Gartner Hype Cycle for Emerging technologies.* 2017, https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/. Accessed : Nov 2017.

[21] Brooks, J.O., et al., *Simulator sickness during driving simulation studies.* Accident Analysis & Prevention, 2010. 42(3): p. 788-796.

[22] Joseph J. LaViola, J., *A discussion of cybersickness in virtual environments.* SIGCHI Bull., 2000. 32(1): p. 47-56.

[23] Stoffregen, T.A., Riccio, Gary E., *An ecological theory of orientation and the vestibular system.* Psychological Review, 1988. Vol 95(1): p. 3-14.

[24] Voshart, D., *Architecture and VR.* Masters Thesis, University of Toronto, 2014.

[25] The Verge, *The rise and fall and rise of Virtual Reality, The Verge.* 2017, https://www.theverge.com/a/virtual-reality. Accessed : Dec 2017.

[26] Oculus, *Oculus Connect Keynote: Brendan Iribe and Nate Mitchell.* 2014, https://www.youtube.com/watch?v=1xeK8zUXAvQ. Accessed : Dec 2017.

[27] Unity, *Optimisation for VR in Unity.* https://unity3d.com/pt/learn/tutorials/topics/virtual-reality/optimisation-vr-unity. Accessed : Dec 2017.

[28] Google, *Tenha seu Google Cardboard.* 2017, https://vr.google.com/cardboard/get-cardboard/. Accessed : Dec 2017.

[29] PCMag, *The best Virtual Reality Headsets of 2018.* 2017, https://www.pcmag.com/article/342537/the-best-virtual-reality-vr-headsets. Accessed : Dec 2017.

[30] Ward, J., *What is a Game Enigne?* 2008, https://www.gamecareerguide.com/features/529/what_is_a_game_.php. Accessed ; Dec 2017.

[31] Azuma, R.T., *A Survey of Augmented Reality.* In Presence: Teleoperators and Virtual Environments, 1997: p. 355-385.

[32] Autodesk, *Augmented Reality in Construction Lets You See Through Walls.* 2017, https://www.autodesk.com/redshift/augmented-reality-in-construction/. Accessed : Dec 2017.

[33] *"Gimmick"* in *Merriam-Webster.com.* https://www.merriam-webster.com/dictionary/gimmick, Accessed : Dec 2017.

[34] Grajewski, D., et al., *Improving the Skills and Knowledge of Future Designers in the Field of Ecodesign Using Virtual Reality Technologies.* Procedia Computer Science, 2015. 75: p. 348-358.

[35] Winn, W., *A Conceptual Basis for Educational Applications of Virtual Reality.* University of Washigton, 1993.

[36] Paes, D., E. Arantes, and J. Irizarry, *Immersive environment for improving the understanding of architectural 3D models: Comparing user spatial perception between immersive and traditional virtual reality systems.* Automation in Construction, 2017. 84(Supplement C): p. 292-303.

[37] Sampaio, A.Z., et al., *3D and VR models in Civil Engineering education: Construction, rehabilitation and maintenance.* Automation in Construction, 2010. 19(7): p. 819-828.

[38] Abdelhameed, W.A., *Virtual Reality Use in Architectural Design Studios: A Case of Studying Structure and Construction.* Procedia Computer Science, 2013. 25: p. 220-230.

[39] *Detailing of a precast concrete.* https://i.pinimg.com/474x/96/9c/71/969c7162236d897f28529b828100dcac--stair-handrail-precast-concrete.jpg. Accessed : Nov 2017.

[40] Bouygues Construction, *Bouygues Construction and HTC Vive to collaborate on developing virtual reality training for accident prevention on construction sites.* http://www.bouygues-construction.com/en/press/news/bouygues-construction-and-htc-vive-collaborate-developing-virtual-reality-training-accident-prevention-construction-sites, 2017. Accessed : Jul 2017.

[41] Goulding, J., et al., *Construction industry offsite production: A virtual reality interactive training environment prototype*. Advanced Engineering Informatics, 2012. 26(1): p. 103-116.

[42] Industrial Training International, *ITI Crane VR simulators.* 2017, https://www.iti.com/vr. Accessed : Dec 2017.

[43] IKEA, *Virtual Reality - Into the Magic.* 2017, http://www.ikea.com/ms/en_US/this-is-ikea/ikea-highlights/Virtual-reality/index.html. Accessed : Dec 2017.

[44] VRtisan, *Architectural Design in Virtual Reality - VRtusan - Unreal Engine VR Editor*. 2016, https://www.youtube.com/watch?v=dXI8Z-tu1PY&t=109s. Accessed : Oct 2017.

[45] Newtons apple, *Virtual Reality on Newtons apple.* 1993, https://www.youtube.com/watch?v=u1A3SmtLW_07. Accessed : Dec 2017.

[46] Mortenson Group, *Virtual Reality Game Changer.* 2017, http://www.mortenson.com/company/news-and-insights/insights/vr-in-mmg. Accessed : Dec 2017.

[47] Kathleen Mckilinley, J.K., Martin FIscher, Craig Howard, *Interactive 4D-CAD*. Computing in Civil Engineering, 1996.

[48] Theb1m, *Building a skyscraper with 4D BIM VR.* 2017, http://www.theb1m.com/video/building-a-skyscraper-with-4d-vr. Accessed : Dec 2017.

[49] Microsoft, *Microsoft HoloLens: Skype.* 2016, https://www.youtube.com/watch?v=4QiGYtd3qNI. Accessed : Dec 2017.

[50] South China Morning Post, *How virtual reality will revolutionise sales and marketing in the real estate industry, South China Morning Post.* 2017, http://www.scmp.com/property/hong-kong-china/article/2105922/how-virtual-reality-will-revolutionise-sales-and-marketing. Accessed : Dec 2017.

[51] Felnhofer, A., et al., *Is virtual reality emotionally arousing? Investigating five emotion inducing virtual park scenarios.* International Journal of Human-Computer Studies, 2015. 82: p. 48-56.

[52] Forbes, *How virtual reality could revolutionize the real estate industry.* 2017, https://www.forbes.com/sites/forbesagencycouncil/2017/03/28/how-virtual-reality-could-revolutionize-the-real-estate-industry/#f1497bdd9b34. Accessed : Dec 2017.

[53] AtlasBayVR, *High-end virtual reality for real estate.* 2017, https://www.youtube.com/watch?v=8l9qRDMVNRY. Accessed : Dec 2017.

[54] Ma, Z., D. Zhang, and J. Li, *A dedicated collaboration platform for Integrated Project Delivery.* Automation in Construction, 2018. 86(Supplement C): p. 199-209.

[55] Scozzari, *Collaborating to increase Value and Efficiency, while reducing waste.* http://www.vjscozzariandsons.com/what-we-do/integrated-project-delivery/. Accessed : Dec 2017.

[56] Autodesk, *Built for Bim.* 2017, https://www.autodesk.pt/products/revit-family/overview. Accessed : Dec 2017.

[57] Dynamo, *Dynamo BIM.* 2017, http://dynamobim.org/. Accessed : Jan 2018.

[58] Dynamo Primer, *What's a Code Block?* 2017, http://dynamoprimer.com/en/07_Code-Block/7-1_what-is-a-code-block.html. Accessed : Dec 2017.

[59] The Next Web, *This engine is dominating the gaming industry right now.* 2016, https://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now/. Accessed : Dec 2017.

[60] Monteiro, M., *Desenvolvimento de interfaces tridimensionais para aplicações móveis a partir da tecnologia BIM.* Masters Thesis, Faculty of Engineering of University of Porto, 2013.

[61] Dinis, F.A., *Desenvolvimento de processos de interação entre tecnologias BIM e equipamentos de Realidade Virtual e sua aplicabilidade.* Masters Thesis, Faculty of Engineering of University of Porto, 2016.

[62] Scan, *Vive contents.* 2017, https://www.scan.co.uk/images/infopages/ln70472/vive-contents.png. Accessed : Dec 2017.

[63] Autodesk, *Subscribe to 3DS Max.* 2018, https://www.autodesk.pt/products/3ds-max/subscribe/. Accessed : Jan 2018.

[64] Shin, D.-H., *The role of affordance in the experience of virtual reality learning: Technological and affective affordances in virtual reality.* Telematics and Informatics, 2017. 34(8): p. 1826-1836.

[65] Unity, *User interfaces for VR.* 2013, https://unity3d.com/pt/learn/tutorials/topics/virtual-reality/user-interfaces-vr. Accessed : Dec 2017.

[66] Unity, *Input for OpenVR controllers.* 2017, https://docs.unity3d.com/Manual/OpenVRControllers.html. Accessed : Dec 2017.

[67] Alex Hagenah, *Reticles, controllers and buttons: Commn interactions in VR.* 2017, http://ixd.prattsi.org/2016/11/interactions-in-vr/. Accessed : Dec 2017.

[68] Google, *Display Reticle.* 2017, https://designguidelines.withgoogle.com/cardboard/interactive-patterns/display-reticle.html. Accessed : Dec 2017.

[69] Microsoft, *Voice Input in Unity.* 2017, https://developer.microsoft.com/en-us/windows/mixed-reality/voice_input_in_unity. Accessed : Jan 2018.

[70] Alger, M., *Visual Design Methods for Virtual Reality.* Master thesis, Rabensbourne University, 2015.

[71] Oculus, *Locomotion.* 2017, https://developer.oculus.com/design/latest/concepts/bp-locomotion/. Accesed : Dec 2017.

ATTACHMENT A

All the scripts created for the Unity application

In this section it is going to be presented all the codes created for the interactive collaboration application. Each code it is going to be presented the name (that already suggests one of the primary objective of the code), the primary objectives are going to be briefly introduced, the programming language code used and in which GameObject the code was attached, if it was one GameController (empty game object cointaining only the code) or if it was attached to an specific object of the VR scene.

A.1. ALWAYS FACE CAMERA

Main objective: Redirect a GameObject to always face a certain camera, no matter which direction the camera is

Programming code: C#

Attached to: GameObject

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class AlwaysFaceCamera : MonoBehaviour {

    public Camera cameraToLookAt;

    void Start()
    {
        //transform.Rotate( 180,0,0 );
    }

    void Update()
    {
        if (cameraToLookAt == null)
        {
            // Create a temporary reference to the current scene.
            Scene currentScene = SceneManager.GetActiveScene();

            // Retrieve the name of this scene.
```

```
            string sceneName = currentScene.name;


            if (sceneName == "Scene VR")
            {
                cameraToLookAt = GameObject.Find("Camera
(eye)").GetComponent<Camera>();
            }


            if (sceneName != "Scene VR")
            {
                cameraToLookAt =
GameObject.Find("FirstPersonCharacter").GetComponent<Camera>();
            }
        }


        Vector3 v = cameraToLookAt.transform.position - transform.position;
        v.x = v.z = 0.0f;
        transform.LookAt(cameraToLookAt.transform.position - v);
        transform.Rotate(0, 180, 0);
    }
}
```

A.2. SCALE TO CAMERA

Main objective: Rescale a GameObject according to the main cameras position, in order that the object has always the same size on the screen.

Programming code: C#

Attached to: GameObject

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class ScaleToCamera : MonoBehaviour
{
    public Camera cam;
    public float objectScale = 1.0f;
    private Vector3 initialScale;

    // set the initial scale, and setup reference camera
    void Start()
    {
        // record initial scale, use this as a basis
        initialScale = transform.localScale;
    }

    // scale object relative to distance from camera plane
    void Update()
    {
        if(cam == null)
        {

                // Create the active scene reference
                Scene currentScene = SceneManager.GetActiveScene();

                // Retrieve the name of this scene.
                string sceneName = currentScene.name;
```

```
                if (sceneName == "Scene VR")
                {
                    cam = GameObject.Find("Camera (eye)").GetComponent<Camera>();
                }

                if (sceneName != "Scene VR")
                {
                    cam =
GameObject.Find("FirstPersonCharacter").GetComponent<Camera>();
                }
        }

        Plane plane = new Plane(cam.transform.forward, cam.transform.position);
        float dist = plane.GetDistanceToPoint(transform.position);
        transform.localScale = initialScale * dist * objectScale;
    }
}
```

A.3. CUSTOM LIST

Main objective: Create the list of database to be shown on the unity Inspector

Programming code: C#

Attached to: GameController (ListObjects)

```
using UnityEngine;
using System;
using System.Collections.Generic; // Import the System.Collections.Generic class
to give us access to List<>


public class CustomList : MonoBehaviour
{
    //This is our custom class with our variables
    [System.Serializable]
    public class MyClass
    {
        public GameObject AssociatedGO;
        public string Name;
        public string Local;
        public string Supplier;
        public string Material;
        public string Dimensions;
        public string Commentaires;
        public bool Validated = false;
        public Vector3 position;
        //public int[] AnIntArray = new int[0];
    }




    //This is our list we want to use to represent our class as an array.
    public List<MyClass> MyList = new List<MyClass>(1);

    void AddNew()
    {
```

```
        //Add a new index position to the end of our list
        MyList.Add(new MyClass());

    }

    void Remove(int index)
    {
        //Remove an index position from our list at a point in our list array
        MyList.RemoveAt(index);
    }
}
```

A.4. INTERACTIVE PANEL SCRIPT

Main objectives:

- This script manages the animation of the interactive button canvas. If the Yes Valid button it is pressed, the animation turns the button to green. Similarly, if the No Valid Button it is pressed the animation turns the button to red.
- It gets the position of the button on Unity and store it to be placed in Revit
- It allows the button to open or close the information panel of the objective.

Programming code: C#

Attached to: GameController that is linked to a Canvas

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;



public class InteractivePanelScript : MonoBehaviour {

    public Button intButton;

    public GameObject panelInfo;

    public int newId;


    //Part dedicated to control the animations

    public Button validatedButton;

    public Button nonValidatedButton;

    public Animator currAnimator;



    void Start()

    {

        Button btn = intButton.GetComponent<Button>();

        btn.onClick.AddListener(OpenClosePanel);


        // Activate the panel to get the reference of the buttons (it initiates
as deactivated)

        panelInfo.SetActive(true);


        validatedButton = GameObject.Find("Button (Yes)").GetComponent<Button>();
```

```
        nonValidatedButton = GameObject.Find("Button
(No)").GetComponent<Button>();


        validatedButton.onClick.AddListener(ValidatedPressed);
        nonValidatedButton.onClick.AddListener(NonValidatedPressed);


        currAnimator = intButton.GetComponent<Animator>();


        // Save the position of the Canvas Object to be created in the Revit
        DataBaseInfo.instance.listRef[newId].position = this.transform.position;


        // Deactive the panel
        panelInfo.SetActive(false);
    }



    public void OpenClosePanel()
    {
        // The variable -1 is selected to be the default ID
        if (DataBaseInfo.instance.actId == newId)
        {
            Debug.Log("Closing/Opening the panel");
            panelInfo.SetActive(!panelInfo.activeSelf);
        }
        else
        {
            Debug.Log("Opening the panel");
            panelInfo.SetActive(true);
            Debug.Log("Id before clickin on button: " +
DataBaseInfo.instance.actId);
            DataBaseInfo.instance.actId = newId;
            Debug.Log("Id after clicking on button: " +
DataBaseInfo.instance.actId);
        }
    }
```

```
    public void ValidatedPressed()
    {
        if (newId == DataBaseInfo.instance.actId)
        {
            currAnimator.SetBool("Validated",
!currAnimator.GetBool("Validated"));
        }

    }
    public void NonValidatedPressed()
    {
        if (newId == DataBaseInfo.instance.actId)
        {
            currAnimator.SetBool("NonValidated",
!currAnimator.GetBool("NonValidated"));
        }
    }
}
```

A.5. DATABASE INFO

Main objectives:

- Copy the information that is retrieved from the CustomList script and store this information on a dynamic variable reference
- Update the information that it is going to be shown at the information panel.
- Update the game object that it is going to be shown at the showcase camera.

Programming code: C#

Attached to: GameController(SetText)

```
using UnityEngine;

using System.Collections;

using System.Collections.Generic;

using UnityEngine.UI;


public class DataBaseInfo : MonoBehaviour{


    public Text namePanel;

    public Text informationPanel;


    //Necessary to create an instance to be able to change the value of actId

    public static DataBaseInfo instance;



    // ActId is the ID that it is currently active according to the button
pressing, that changes the text on the panel.

    public int actId = -1;


    //Create the list using the same Class in the CustomList Script, values to be
showed on the panel and exported

    public List<CustomList.MyClass> listRef = new List<CustomList.MyClass>(1);


    public string _Name;

    public string _Localisation;

    public string _Supplier;

    public string _Material;

    public string _Dimensions;

    public int sizeOfList;
```

```
    public void Awake()
    {
        instance = this;
        //Get the non-static value from the inspector and create a static
reference in Unity
        GameObject myGO = GameObject.Find("_ListObjects");
        CustomList test = myGO.GetComponent<CustomList>();

        listRef = test.MyList;
        sizeOfList = listRef.Count;
    }
    public void Update()
    {
        if (actId + 1 > sizeOfList)
        {
            Debug.Log("Invalid ID, please verify the values at the game
controllers to be able to update the panel information");
        }

        else
        {

            //Important part for the text

            //Pass the reference to the strings to be shown on the panel
according to the ActiveID
            _Name = listRef[actId].Name;
            _Localisation = listRef[actId].Local;
            _Supplier = listRef[actId].Supplier;
            _Material = listRef[actId].Material;
            _Dimensions = listRef[actId].Dimensions;

            //Change the text name of the Panel according to the name of object
            namePanel.text = _Name;
```

```
            //Change the information of the object according to the info inserted
in the inspector
            informationPanel.text = "Localisation : " + _Localisation + "\r\n" +
                                    "Supplier     : " + _Supplier + "\r\n" +
                                    "Material     : " + _Material + "\r\n" +
                                    "Dimensions   : " + _Dimensions;




            // Activate Object to show on the panel camera
            if (listRef[actId].AssociatedGO.activeSelf == false)
            {
                //It will disable all game objects and let only the actId object
active
                for (int i = 0; i < sizeOfList; i++)
                {
                    listRef[i].AssociatedGO.SetActive(false);
                }

                listRef[actId].AssociatedGO.SetActive(true);
            }



        }
    }
}
```

A.6.SAVE COMMENTARIES

Main objective: Save the commentaries added according to the information panel

Programming code: C#

Attached to: GameController(SaveCommentaires)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class SaveCommentaires : MonoBehaviour
{

    public Button saveDatafile;
    public Button quitDatafile;
    public InputField newCommentaire;
    public GameObject panelGlobal;
    public GameObject panelCommentaire;

    private int newId;
    void Start()
    {
        Button btn = saveDatafile.GetComponent<Button>();
        Button btn2 = quitDatafile.GetComponent<Button>();

        btn.onClick.AddListener(SaveComm);
        btn2.onClick.AddListener(QuitComm);

    }
    public void SaveComm()
    {
        newCommentaire.Select();
        // Set the Id to the active panel ID
        newId = DataBaseInfo.instance.actId;
        DataBaseInfo.instance.listRef[newId].Commentaires = newCommentaire.text;
```

```
        Debug.Log("Id: " + newId + "  Commentaire saved: " +
DataBaseInfo.instance.listRef[newId].Commentaires);
        Debug.Log("Saved file!");



        //Reset the input field
        newCommentaire.text = " ";



        //Close the panels
        panelGlobal.SetActive(false);
        panelCommentaire.SetActive(false);
    }
    public void QuitComm()
    {
        //Close the panels
        panelGlobal.SetActive(false);
        panelCommentaire.SetActive(false);
    }
}
```

A.7.SAVE VALIDATED INFO

Main objective: Save if the object selected it is validated or not.

Programming code: C#

Attached to: GameController(SaveValidatedInfo)

```
sing System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class SaveValidatedInfo : MonoBehaviour {

    public Button validatedButton;
    public Button nonValidatedButton;
    public int newId;

    void Start()
    {
        //Validated button
        Button btn = validatedButton.GetComponent<Button>();
        btn.onClick.AddListener(TaskOnClick);

        //Non validated button
        Button btn2 = nonValidatedButton.GetComponent<Button>();
        btn2.onClick.AddListener(TaskOnClick2);
    }

    public void TaskOnClick()
    {
        // Set the Id to the active panel ID
        newId = DataBaseInfo.instance.actId;

        Debug.Log("Id: " + newId + "  Value before clickin on button: " +
DataBaseInfo.instance.listRef[newId].Validated);
        DataBaseInfo.instance.listRef[newId].Validated = true;
```

```
        Debug.Log("Id: " + newId + "  Value after clickin on button: " +
DataBaseInfo.instance.listRef[newId].Validated);

    }

    public void TaskOnClick2()

    {

        // Set the Id to the active panel ID

        newId = DataBaseInfo.instance.actId;


        Debug.Log("Id: " + newId + "  Value before clickin on button: " +
DataBaseInfo.instance.listRef[newId].Validated);

        DataBaseInfo.instance.listRef[newId].Validated = false;

        Debug.Log("Id: " + newId + "  Value after clickin on button: " +
DataBaseInfo.instance.listRef[newId].Validated);

    }

}
```

A.8.DROP DOWN LIST

Main objective: Change the editor in Unity related to the CustomList script (adding buttons and giving an intuitive interface)

Programming code: C#

```
using UnityEngine;

using System.Collections;

using System.Collections.Generic;

using UnityEditor;


[CustomEditor(typeof(CustomList))]


public class CustomListEditor : Editor

{

    CustomList t;

    SerializedObject GetTarget;

    SerializedProperty ThisList;

    int ListSize;


    void OnEnable()

    {

        t = (CustomList)target;

        GetTarget = new SerializedObject(t);

        ThisList = GetTarget.FindProperty("MyList"); // Find the List in our
script and create a reference of it

    }


    public override void OnInspectorGUI()

    {

        //Update our list


        GetTarget.Update();


        //Resize our list

        EditorGUILayout.Space();

        EditorGUILayout.Space();
```

```
EditorGUILayout.LabelField("Define the list size with a number");
ListSize = ThisList.arraySize;
ListSize = EditorGUILayout.IntField("List Size", ListSize);

if (ListSize != ThisList.arraySize)
{
    while (ListSize > ThisList.arraySize)
    {
        ThisList.InsertArrayElementAtIndex(ThisList.arraySize);
    }
    while (ListSize < ThisList.arraySize)
    {
        ThisList.DeleteArrayElementAtIndex(ThisList.arraySize - 1);
    }
}

EditorGUILayout.Space();
EditorGUILayout.Space();
EditorGUILayout.LabelField("Or");
EditorGUILayout.Space();
EditorGUILayout.Space();

//Or add a new item to the List<> with a button
EditorGUILayout.LabelField("Add a new item with a button");

if (GUILayout.Button("Add New"))
{
    t.MyList.Add(new CustomList.MyClass());
}

EditorGUILayout.Space();
EditorGUILayout.Space();

//Display our list to the inspector window
```

```
        for (int i = 0; i < ThisList.arraySize; i++)
        {
            //GetValues from the list, creating a reference
            SerializedProperty MyListRef = ThisList.GetArrayElementAtIndex(i);
            SerializedProperty MyName = MyListRef.FindPropertyRelative("Name");
            SerializedProperty MyLocal = MyListRef.FindPropertyRelative("Local");
            SerializedProperty MySupplier =
MyListRef.FindPropertyRelative("Supplier");
            SerializedProperty MyMaterial =
MyListRef.FindPropertyRelative("Material");
            SerializedProperty MyDimensions =
MyListRef.FindPropertyRelative("Dimensions");
            SerializedProperty MyCommentaires =
MyListRef.FindPropertyRelative("Commentaires");
            SerializedProperty MyValidated =
MyListRef.FindPropertyRelative("Validated");

            //If necessary
            //SerializedProperty MyArray =
MyListRef.FindPropertyRelative("AnIntArray");
            SerializedProperty MyGO =
MyListRef.FindPropertyRelative("AssociatedGO");

            // Display Information in the inspector window
                EditorGUILayout.LabelField("Automatic Field By Property Type");
                EditorGUILayout.LabelField("Element Index:             " + i);
                EditorGUILayout.PropertyField(MyGO);
                EditorGUILayout.PropertyField(MyName);
                EditorGUILayout.PropertyField(MyLocal);
                EditorGUILayout.PropertyField(MySupplier);
                EditorGUILayout.PropertyField(MyMaterial);
                EditorGUILayout.PropertyField(MyDimensions);
                EditorGUILayout.PropertyField(MyCommentaires);
                EditorGUILayout.PropertyField(MyValidated);

            EditorGUILayout.Space();
```

```
            //Remove this index from the List
            EditorGUILayout.LabelField("Remove an index from the List<> with a
button");
            if (GUILayout.Button("Remove This Index (" + i.ToString() + ")"))
            {
                ThisList.DeleteArrayElementAtIndex(i);
            }
            EditorGUILayout.Space();
            EditorGUILayout.Space();
            EditorGUILayout.Space();
            EditorGUILayout.Space();
        }

        //Apply the changes to our list
        GetTarget.ApplyModifiedProperties();
    }
}
```

A.9.RAYCAST HIT COM

Main objectives:

- Allow the creation of the NewCommentary with the click of a button (creation of the Canvas with the button with it)
- Add one object to the InfoNewCommentaryList
- Save the position of the NewCommentary

Programming code: C#

Attached to: GameController(Raycast comm)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


public class RaycastHitCom : MonoBehaviour {


    public GameObject panelPrefab;
    public GameObject textDisplay;
    public GameObject panelNewCommentaire;
    public Text HeaderPanelId;
    public GameObject Player;


    public Color debugColor;


    [HideInInspector] public Vector3 posInstance;
    // Avoid having problems while creating clones
    [HideInInspector] public GameObject prefabInstance;
    public int numberNewComments = 0;


    // Use this for initialization
    void Start ()
    {


       }


       // Update is called once per frame
```

```
       void Update ()
    {
        // It will only create the rays if the display text is activated
        if(textDisplay.activeSelf == true && panelNewCommentaire.activeSelf ==
false)
        {

            // It will cast a ray when the mouse button is pressed down
            if (Input.GetMouseButtonDown(0))
            {
                // Get the origin of the ray according to mouse position
                Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

                // Variable reading information about the collider hit
                RaycastHit hit;

                if (Physics.Raycast(ray, out hit, 100))
                {

                    //Debug the raycast in the scene
                    Debug.DrawRay(ray.origin, hit.point, debugColor);

                    // Get if the collider has hit something
                    if (hit.collider != null)
                    {

                        // Slightly move the canvas so it can be better
positioned
                        posInstance = Vector3.MoveTowards(hit.point,
Player.transform.position, 0.18f);

                        // Create a new canvas with the prefab in the hit point
                        prefabInstance = Instantiate(panelPrefab, posInstance,
transform.rotation);

                        //Enable the deactivated scripts to the new instantiated
object
```

```
                        prefabInstance.GetComponent<AlwaysFaceCamera>().enabled =
true;
                        prefabInstance.GetComponent<ScaleToCamera>().enabled =
true;

                        // Change the ID according to the number of comments
created
prefabInstance.GetComponent<IdNewCommentaire>().newCommentId = numberNewComments;

                        // Change the name of the instance according to the
commentaire
                        prefabInstance.name = "Canvas New Commentaire (" +
numberNewComments + ")";
                        Debug.Log("Commentaire ID:" + numberNewComments + "  was
created!");

                        SaveNewCommentaires.instance.currentId =
numberNewComments;
                        HeaderPanelId.text = "ID:" + numberNewComments;



                        // Add one string to the InfoNewCommentaireList
                        InfoNewCommentaire.instance.MyNewList.Add(new
InfoNewCommentaire.NewComments());
InfoNewCommentaire.instance.MyNewList[numberNewComments].position = hit.point;

                        // Increment the value
                        numberNewComments++;

                        // Open panel commentaire and close the display text

                        panelNewCommentaire.SetActive(true);
                        textDisplay.SetActive(false);
                    }
                    else
                    {
                        Debug.Log("The raycast didn't find a collider!");
```

```
                        }
                    }
                }
            }
        }
}
```

A.10.SAVE NEW COMMENTAIRES

Main objectives:

- Allow the creation of the NewCommentary with the click of a button (creation of the Canvas with the button with it)
- Add one object to the InfoNewCommentaryList
- Save the position of the NewCommentary

Programming code: C#

Attached to: GameController(SaveNewComm)

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;


public class SaveNewCommentaires : MonoBehaviour

{


    public Button saveNew;

    public Button quitNew;

    public InputField inputNewCommentaire;

    public GameObject panelNewCommentaire;


    public static SaveNewCommentaires instance;


    public int currentId = 0;


    public void Awake()

    {

        instance = this;

    }


    void Start()

    {

        // Make sure to instance this script to be able to be access from another
scripts
```

```
        Button btn = saveNew.GetComponent<Button>();
        Button btn2 = quitNew.GetComponent<Button>();

        btn.onClick.AddListener(SaveNewComm);
        btn2.onClick.AddListener(QuitNewComm);
    }

    public void SaveNewComm()
    {

        // Make sure to select the input field
        inputNewCommentaire.Select();

        // Get the input from the inputfield and save it to the variable at the
list
        InfoNewCommentaire.instance.MyNewList[currentId].newString =
inputNewCommentaire.text;

        // Debugger
        Debug.Log("Succesfully saved new comment!!\r\n ID: "+ currentId + "\r\n"
+ "Comment: " + InfoNewCommentaire.instance.MyNewList[currentId].newString);

        // Reset the input field
        inputNewCommentaire.text = " ";

        // Close the panel
        panelNewCommentaire.SetActive(false);



    }

    public void QuitNewComm()
    {
        InfoNewCommentaire.instance.MyNewList[currentId].newString = "Deleted
Commentaire";
```

```
        Destroy(GameObject.Find("Canvas New Commentaire (" + currentId + ")") );


        Debug.Log("Deleted commentaire ! " + "Commentaire ID: " +
InfoNewCommentaire.instance.newActId);


        // Reset the input field
        inputNewCommentaire.text = " ";


        // Close the panel
        panelNewCommentaire.SetActive(false);
    }
}
```

A.11. ID NEW COMMENTAIRE

Main objectives:

- Changes the ID of the new commentary that will be shown in Unity
- Manages the information that it is going to be shown at the NewCommentaryPanel

Programming code: C#

Attached to: CanvasPrefab of new commentaries

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;


public class IdNewCommentaire : MonoBehaviour

{

    public Button commentButton;

    public InputField inputNewCommentaire;

    public Text HeaderID;

    public int newCommentId;


    public void Start()

    {

        commentButton =
this.transform.Find("Button(Comm)").GetComponent<Button>();

        if (commentButton != null)

        {

            Button btn = commentButton.GetComponent<Button>();

            btn.onClick.AddListener(ChangeId);

        }

        else

        {

            Debug.Log("Failed to assign the comment button.");

        }

    }


    public void ChangeId()

    {
```

```
        InfoNewCommentaire.instance.newActId = newCommentId;

        SaveNewCommentaires.instance.currentId = newCommentId;

        inputNewCommentaire.text =
InfoNewCommentaire.instance.MyNewList[newCommentId].newString;

        HeaderID.text = "ID:" + newCommentId;

    }

}
```

A.12. INFO NEW COMMENTAIRE

Main objective: Creates the list that it is going to receive the new commentaries information while the application it is being executed

Programming code: C#

Attached to: GameController(ListNewCommentary)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class InfoNewCommentaire : MonoBehaviour {



    public static InfoNewCommentaire instance;

    // Used on the SaveNewCommentaires script to deleted the canvas with this ID
    public int newActId = -1;

    public List<NewComments> MyNewList = new List<NewComments>(1);

    void Awake()
    {
        instance = this;
    }

    public class NewComments
    {
        public string newString;
        public Vector3 position;
    }
}
```

A.13. OBJECT BUILDER SCRIPT

Main objective: Allow the creation in the inspector of a object builder, that create a clone of a new object according to the desired coordinates.

Programming code: C#

Attached to: GameController(Builder)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class ObjectBuilderScript : MonoBehaviour {
    public GameObject obj;
    public int numberOfCopies;
    public Vector3 spawnPoint;


    public void BuildObject()
    {
        for(int i = 0; i<numberOfCopies; i++)
        {
            Instantiate(obj, spawnPoint, Quaternion.identity);
        }
    }
```

A.14. OBJECT BUILDER EDITOR

Main objective: Editor that changes the inspector to allow the ObjectBuilderScript to run.

Programming code: C#

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEditor;

[CustomEditor(typeof(ObjectBuilderScript))]

public class ObjectBuilderEditor : Editor
{
    public override void OnInspectorGUI()
    {
        DrawDefaultInspector();

        ObjectBuilderScript myScript = (ObjectBuilderScript)target;
        if(GUILayout.Button("Build Object"))
        {
            myScript.BuildObject();
        }
    }

}
```

A.15. MEASURE DISTANCES

Main objective:

- Allow the creation of the measurement tool
- Creates 2 spheres(start point and end point), one line renderer connecting these two points and the text with the distance between these two points.

Programming code: C#

Attached to: GameController(Raycast hit comm)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class MeasureDistances : MonoBehaviour
{
    public GameObject textDisplay;
    public GameObject text2Display;
    public GameObject Player;
    public Material LineRendererMaterial;

    public bool Ray1Casted = false;
    public bool Ray2Casted = false;

    public float duration = 15.0f;
    public float distance;

    [HideInInspector] public GameObject instanceText;
    [HideInInspector] public int numberLines = 0;
    [HideInInspector] public Vector3 pos1;
    [HideInInspector] public Vector3 pos2;
    [HideInInspector] public Color c1 = Color.green;

    // Update is called once per frame
    void Update()
    {
        // It will only create the rays if the display text is activated
        if (textDisplay.activeSelf == true)
```

```
        {
            // It will cast a ray when the mouse button is pressed down and Ray1
wasnt casted
            if (Input.GetMouseButtonDown(0) && Ray1Casted == false)
            {
                // Get the origin of the ray according to mouse position
                Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

                // Variable reading information about the collider hit
                RaycastHit hit;

                if (Physics.Raycast(ray, out hit, 100))
                {
                    // Get if the collider has hit something
                    if (hit.collider != null)
                    {
                        Debug.Log("Ray1 Casted!");
                        Ray1Casted = true;
                        pos1 = hit.point;
                        text2Display.SetActive(true);
                    }

                    // If it doesn't hit any collider
                    else
                    {
                        Debug.Log("The raycast didn't find a collider!");
                    }
                }
            }
            // It will cast a ray when the mouse button is pressed down and ray1
was already casted
            else if (Input.GetMouseButtonDown(0) && Ray2Casted == false)
            {
                // Get the origin of the ray according to mouse position
                Ray ray2 = Camera.main.ScreenPointToRay(Input.mousePosition);
```

```
    // Variable reading information about the collider hit
    RaycastHit hit2;

    if (Physics.Raycast(ray2, out hit2, 100))
    {
        // Get if the collider has hit something
        if (hit2.collider != null)

            Debug.Log("Ray2 Casted!");
        Ray2Casted = true;
        pos2 = hit2.point;
    }
    // If it doesn't hit any collider
    else
    {
        Debug.Log("The raycast didn't find a collider!");
    }
}

// If both rays were casted, draw the line and show the distance
if (Ray1Casted == true && Ray2Casted == true)
{

    distance = Vector3.Distance(pos1, pos2);

    // Draw a temporary line
    Debug.Log("Line render:  " + numberLines + "  Casted!");
    GameObject myLine = new GameObject();
    myLine.name = "Measure  " + numberLines.ToString();
    myLine.transform.position = pos1;
    myLine.AddComponent<LineRenderer>();
    LineRenderer lr = myLine.GetComponent<LineRenderer>();
    lr.material = LineRendererMaterial;
    lr.widthMultiplier = 0.01f;
```

```
                float alpha = 1.0f;
                Gradient gradient = new Gradient();
                gradient.SetKeys(
                    new GradientColorKey[] { new GradientColorKey(c1, 0.0f), new
GradientColorKey(c1, 1.0f) },
                    new GradientAlphaKey[] { new GradientAlphaKey(alpha, 1.0f),
new GradientAlphaKey(alpha, 1.0f) }
                    );

                pos1 = Vector3.MoveTowards(pos1, Player.transform.position,
0.10f);
                pos2 = Vector3.MoveTowards(pos2, Player.transform.position,
0.10f);

                lr.colorGradient = gradient;
                lr.SetPosition(0, pos1);
                lr.SetPosition(1, pos2);

                // Draw the points(spheres of the line)
                GameObject mySphere1 =
GameObject.CreatePrimitive(PrimitiveType.Sphere);
                mySphere1.name = "Sphere Start";
                mySphere1.transform.localScale = new Vector3(0.02f, 0.02f,
0.02f);
                mySphere1.transform.position = pos1;

                GameObject mySphere2 =
GameObject.CreatePrimitive(PrimitiveType.Sphere);
                mySphere2.name = "Sphere End";
                mySphere2.transform.localScale = new Vector3(0.02f, 0.02f,
0.02f);
                mySphere2.transform.position = pos2;

                // Draw the text of the distance

                GameObject myText = new GameObject();
```

```
        myText.name = "Text3D";

        // Set the text to be centralized of the line
        myText.transform.position = (pos1 + pos2) / 2;

        // Rotate the text to be in the same directio as the points
        Vector3 targetDir = pos1 - pos2;
        myText.transform.Rotate(-targetDir.x, targetDir.y, targetDir.z);
        // Set the scale of the text mesh
        myText.transform.localScale = new Vector3(0.01f, 0.01f, 0.01f);
        var textMesh = myText.AddComponent<TextMesh>();
        myText.AddComponent<AlwaysFaceCamera>();
        textMesh.fontSize = 60;
        textMesh.fontStyle = FontStyle.Bold;
        textMesh.text = distance.ToString("F2") + "m";

        // Set all game objects created to be child of the myLine

        mySphere1.transform.parent = myLine.transform;
        mySphere2.transform.parent = myLine.transform;
        myText.transform.parent = myLine.transform;

        // Destroy my line with all other components after the duration
        GameObject.Destroy(myLine, duration);



        //Set the ray casts to default value to false
        Ray1Casted = false;
        Ray2Casted = false;
        numberLines++;

        textDisplay.SetActive(false);
        text2Display.SetActive(false);
    }
}
```

A.16. SPEECH RECOGNITION

Main objective:

- Uses the windows 10 speech recognition to insert a text
- Push to talk logic.

Programming code: C#

Attached to: GameController(Speech Recognition)

```
using UnityEngine;
using System.Collections;
using UnityEngine.Windows.Speech;
using UnityEngine.UI;
public class SpeechRecognition : MonoBehaviour
{
    DictationRecognizer dictationRecognizer;
    public InputField inputNewCommentaire;

    // Use this for initialization
    void Start()
    {

        dictationRecognizer = new DictationRecognizer();
        dictationRecognizer.DictationResult += onDictationResult;
        dictationRecognizer.DictationHypothesis += onDictationHypothesis;
        dictationRecognizer.DictationComplete += onDictationComplete;
        dictationRecognizer.DictationError += onDictationError;
        dictationRecognizer.Start();
        Debug.Log("Speech recognition script started with no errors.");
    }
    void onDictationResult(string text, ConfidenceLevel confidence)
    {
        // Run if the dictation resulted in success
        Debug.LogFormat("Dictation result: " + text);
        // Pass the result(succesfull) to the inputfield text
        inputNewCommentaire.text = text;
    }
```

```
    void onDictationHypothesis(string text)
    {
        // Write the current hypothesis of the dictation
        Debug.LogFormat("Dictation hypothesis: {0}", text);
    }

    void onDictationComplete(DictationCompletionCause cause)
    {
        // If the dictation failed, debug the cause
        if (cause != DictationCompletionCause.Complete)
            Debug.LogErrorFormat("Dictation completed unsuccessfully: {0}.",
cause);
    }

    void onDictationError(string error, int hresult)
    {
        // If an error occurred, do something
        Debug.LogErrorFormat("Dictation error: {0}; HResult = [48].", error,
hresult);
    }
```

A.17. SCROLLABLE LIST

Main objective: Allow the fast creation of an generic scrollable list (for the application used it is used for the creation of the extract report list).

Programming code: C#

Attached to: GameController(ScrollableList)

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;

public class ScrollableList : MonoBehaviour
{
```

```
    public GameObject itemPrefab;

    public GameObject basePanel;

    public int itemCount = 10, columnCount = 1;


    public void CreateScrollableList()

    {

        RectTransform rowRectTransform =
itemPrefab.GetComponent<RectTransform>();

        RectTransform containerRectTransform =
basePanel.GetComponent<RectTransform>();


        //calculate the width and height of each child item.

        float width = containerRectTransform.rect.width / columnCount;

        float ratio = width / rowRectTransform.rect.width;

        float height = rowRectTransform.rect.height * ratio;

        int rowCount = itemCount / columnCount;

        if (itemCount % rowCount > 0)

            rowCount++;


        //adjust the height of the container so that it will just barely fit all
its children

        float scrollHeight = height * rowCount;

        containerRectTransform.offsetMin = new
Vector2(containerRectTransform.offsetMin.x, -scrollHeight / 2);

        containerRectTransform.offsetMax = new
Vector2(containerRectTransform.offsetMax.x, scrollHeight / 2);


        int j = 0;

        for (int i = 0; i < itemCount; i++)

        {

            //this is used instead of a double for loop because itemCount may not
fit perfectly into the rows/columns

            if (i % columnCount == 0)

                j++;


            //create a new item, name it, and set the parent

            GameObject newItem = Instantiate(itemPrefab) as GameObject;
```

```
            newItem.name = basePanel.name + " item at (" + i + "," + j + ")";
            newItem.transform.parent = basePanel.transform;

            //move and size the new item
            RectTransform rectTransform = newItem.GetComponent<RectTransform>();

            float x = -containerRectTransform.rect.width / 2 + width * (i %
columnCount);
            float y = containerRectTransform.rect.height / 2 - height * j;
            rectTransform.offsetMin = new Vector2(x, y);

            x = rectTransform.offsetMin.x + width;
            y = rectTransform.offsetMin.y + height;
            rectTransform.offsetMax = new Vector2(x, y);
        }
    }
}
```

A.18. UPDATE SCROLL LIST

Main objective: Update the generic scrollable list with the information provided in the DataBaseInfo.

Programming code: C#

Attached to: GameController(ScrollableList)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class UpdateScrollList : MonoBehaviour {



    public Button openReport;
    public GameObject panelReport;

    public GameObject scrollList;



    public string tempID;

    public Text _Name;
    public Text _ID;
    public Text _Commentaires;

    void Start()
    {
        // Set the function of the button to update the list while clicked
        Button btn = openReport.GetComponent<Button>();
        btn.onClick.AddListener(UpdateList);

        // Activate the panel to be able to get the references
        panelReport.SetActive(true);

        for (int i = 0; i < DataBaseInfo.instance.sizeOfList; i++)
```

```
        {
            int j = i + 1;
            scrollList = GameObject.Find("Scrollable item at (" + i + "," + j +
")");




                if (DataBaseInfo.instance.listRef[i].Validated == true)
                {
                    getChildGameObject(scrollList, "Text (Yes)").SetActive(true);
                    getChildGameObject(scrollList, "Text (No)").SetActive(false);
                }
                else
                {
                    getChildGameObject(scrollList, "Text
(Yes)").SetActive(false);
                    getChildGameObject(scrollList, "Text (No)").SetActive(true);
                }


            _Name = getChildGameObject(scrollList, "Text
(NameObject)").GetComponent<Text>();
            _ID = getChildGameObject(scrollList, "Text
(ID)").GetComponent<Text>();
            _Commentaires = getChildGameObject(scrollList, "Text
(Info)").GetComponent<Text>();


            //Fill in the IDs on the report menu
            tempID = i.ToString();
            _ID.text = tempID;


            //Change the names and the commentaires
            _Name.text = DataBaseInfo.instance.listRef[i].Name;
            _Commentaires.text = DataBaseInfo.instance.listRef[i].Commentaires;
        }


        // Deactive the panel
        panelReport.SetActive(false);
```

```
    }


    static public GameObject getChildGameObject(GameObject fromGameObject, string
withName)
    {
        //Author: Isaac Dart, June-13.
        Transform[] ts =
fromGameObject.transform.GetComponentsInChildren<Transform>(true);
        foreach (Transform t in ts) if (t.gameObject.name == withName) return
t.gameObject;
        return null;
    }




    public void UpdateList()
    {
        // Activate the panel to be able to get the references
        panelReport.SetActive(true);
        for (int i = 0; i < DataBaseInfo.instance.sizeOfList; i++)
        {
            int tempI = i + 1;
            scrollList = GameObject.Find("Scrollable item at (" + i + "," + tempI
+ ")");



            if (DataBaseInfo.instance.listRef[i].Validated == true)
            {
                getChildGameObject(scrollList, "Text (Yes)").SetActive(true);
                getChildGameObject(scrollList, "Text (No)").SetActive(false);
            }
            else
            {
                getChildGameObject(scrollList, "Text (Yes)").SetActive(false);
                getChildGameObject(scrollList, "Text (No)").SetActive(true);
            }
```

```
            _Commentaires = getChildGameObject(scrollList, "Text
(Info)").GetComponent<Text>();


            _Name.text = DataBaseInfo.instance.listRef[i].Name;
           _Commentaires.text = DataBaseInfo.instance.listRef[i].Commentaires;


        }
    }
}
```

A.19. EXTRACT REPORT

Main objective:

- Create a comma-separated-value(CSV) file with all the database information, concerning the interactive buttons database and the NewCommentary database.

Programming code: C#

Attached to: GameController(ExtractData)

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.IO;
using System;
using System.Linq;
using System.Collections.Generic;


public class ExtractReport : MonoBehaviour
{
    public Button ExtractData;


    // Export a CSV File to Excel to be read in Revit
    public static string fileName =
(@"C:\Users\cruzt\OneDrive\Documentos\T4Longjumeau\Export Extract.txt");


    void Start()
    {
        // Assign the function of the button to the class SaveToFile
        Button btn = ExtractData.GetComponent<Button>();
        btn.onClick.AddListener(SaveToFile);
    }
    public static void SaveToFile()
    {
        Debug.Log("Starting saving file!");
        // Initiate the writer
        StreamWriter writer = new StreamWriter(ExtractReport.fileName, true);
```

```
        // Get the system data and hour

        string headerTime ="The report was created at: " +
DateTime.Now.ToString("dd/MM/yyyy   --    HH:mm:ss");

        writer.WriteLine(headerTime);

        // Set up and write the header of the CSV file

        string header =
"ID,Valid,Name,Local,Supplier,Material,Dimensions,Commentaires,Position
X,Position Y,Position Z";

        writer.WriteLine(header);


        // For each object in the list, write all the informations throughout one
line.

        for (int i = 0; i < DataBaseInfo.instance.listRef.Count; i++)

        {


            string infoObject =
string.Format("{0},[48],{2},{3},{4},{5},{6},{7},{8},{9},{10}", i,
DataBaseInfo.instance.listRef[i].Validated,
DataBaseInfo.instance.listRef[i].Name, DataBaseInfo.instance.listRef[i].Local,
DataBaseInfo.instance.listRef[i].Supplier,
DataBaseInfo.instance.listRef[i].Material,
DataBaseInfo.instance.listRef[i].Dimensions,
DataBaseInfo.instance.listRef[i].Commentaires,
DataBaseInfo.instance.listRef[i].position.x.ToString(),
DataBaseInfo.instance.listRef[i].position.y.ToString(),
DataBaseInfo.instance.listRef[i].position.z.ToString());

            writer.WriteLine(infoObject);

        }


        string separate = "New comments created";

        writer.WriteLine(separate);

        string headerNewComments = "ID,Commentaire,Position X,Position Y,Position
Z";

        writer.WriteLine(headerNewComments);


        for (int l = 0; l < InfoNewCommentaire.instance.MyNewList.Count; l++)

        {

            Debug.LogFormat("{0},[48]", l,
InfoNewCommentaire.instance.MyNewList[l].newString);

            string infoComments = string.Format("{0},[48],{2},{3},{4}", l,
InfoNewCommentaire.instance.MyNewList[l].newString,
InfoNewCommentaire.instance.MyNewList[l].position.x.ToString(),
```

```
InfoNewCommentaire.instance.MyNewList[l].position.y.ToString(),
InfoNewCommentaire.instance.MyNewList[l].position.z.ToString());

            writer.WriteLine(infoComments);

        }


        writer.Close();

        Debug.Log("Finished exporting the file!!!");


    }

}
```