3D House Viewer in a Web Environment

DevScope

2017 / 2018

1140443 Diogo Azevedo

3D House Viewer in a Web Environment

DevScope

2017 / 2018

1140443 Diogo Azevedo

Graduation in Computer Engineering

June of 2018

ISEP's Tutor: **João Paulo Pereira**

External Supervisor: **João Sousa, Rui Barbosa**

“If you have built castles in the air, your work need not be lost; that is where they should be. Now put the foundations under them.” — Henry David Thoreau, Walden

Acknowledgements

I'm writing this note of thanks as part of my graduation report. It has been a period of intense learning for me, not only in the scientific arena, but also on a personal level. Writing this report and enrolled this internship was one of the best things that happened me in a professional level. I would like to reflect on the people who have supported and helped me so much throughout this path of learning and improvement.

I would first like to thank my colleagues from my internship at DevScope, for their wonderful collaboration, since they were always there to help me and share ideas related to the project and help me abstract from all the worries and stress during the breaks.

I would particularly like to single out my supervisor at DevScope, João Sousa for the guidance in putting the ideas to practice, as well as the CEO's institution, Rui Barbosa for helping me broaden my horizon, always pushing me further.

In addition, I would like to thank my tutor, Prof. João Paulo Pereira, for his valuable guidance throughout the internship, always willing to help me in anything that I needed.

I would also like to thank my parents for their wise counsel and support for me to carry this internship till the end.

Finally, there are my friends, who helped me abstract from the internship and gave me lots of fun moments to relieve the stress.

Diogo Azevedo

Abstract

Nowadays the rapid advancement of technologies and the introduction of virtual reality in our lives, asks for a step further in online home sales market when it comes to house visualization. With the same sight from the My Tours 360, the internship's institution earlier product of house visualization through panoramic pictures, the solution described in this report aims to bring the user a new way to see online real estate.

Developing a 3D house viewer for online platforms, searches to bring new freedom and interactivity between the user and the real estate, using current and emerging technologies.

The project models the house dynamically from a set of room and furniture coordinates, enabling changes on the real estate in real-time.

We can view the real estate in 3 different view modes, both for desktop and mobile devices. Incorporating virtual reality technologies, the user can walk through the real estate in a more personal and real experience. Besides the virtual reality mode, its possible to view the house from a broad perspective and in First-person experience.

The virtual reality and house modelling aspects of the solution are subject to future improvements.

Keywords (theme): Interior Modelling, 3D House Reconstruction, Simulation Visualization

Keywords (Technologies): WebGL, Virtual Reality, Photogrammetry

Index

Image's Index

Table Index

Notation and Glossary

API	Application Programming Interface
Add-ons	Extension of a computer program to add new features.
Bussiness Inteligence	Area that monitors, evaluates, collects, and shares information that supports business management.
CAGR	The compound annual growth rate (CAGR) is the mean annual growth rate of an investment over a specified period longer than one year. (Wayman, 2018)
CRM	Customer Relationship Management
Degree of Freedom	The number of different "directions" that an object can move in 3D space. 3DOF VR headsets can track your head orientation, i.e., it knows where you are looking. The 3 axis are roll, yaw and pitch. 6DOF headsets will track orientation **and** position, the headsets knows where you are looking and also where you are in space. This is sometimes referred to as *roomscale* or *positional* tracking. (Weis, 2018)
Photogrammetry	Photogrammetry is the science of making measurements from photographs, especially for recovering the exact positions of surface points.
Framework	Abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software.
Headsets	Head-mounted device that provides virtual reality for the wearer.

Machine Learning	Subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. (Arthur Samuel, 1959)
OpenGL	Cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics.
Point Cloud	Set of data points in space. Point clouds are generally produced by 3D scanners, which measure a large number of points on the external surfaces of objects around them
SaaS	(Software as a Service) is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. (Davis, 2014)
Virtual Reality	Virtual reality is the term used to describe a three-dimensional, computer generated environment which can be explored and interacted with by a person. That person becomes part of this virtual world or is immersed within this environment and whilst there, can manipulate objects or perform a series of actions. (vrs, s.d.)
WebVR	WebVR is an open specification that makes it possible to experience VR in your browser. (Webvr.info, 2018)

1. Introduction

In this chapter, the reader will be introduced to the project/internship developed in the DevScope organization, thus getting to know the scope of the project as well as the institution where it was realized.

1.1 Background

The following document was developed under the course of PESTI, taught in the 2nd Semester of the 3rd year of the degree in Computer Engineering of the Higher Institute of Engineering of Porto.

The curricular internship being presented was developed in DevScope, consisting in the development of tri-dimensional house viewer.

This project appears in the institution as the succession of its current product "Virtual Tours", which consists on viewing panoramic photographs.

This solution proves to be innovative, as it is still something underdeveloped in the current sector, as well as being an area of personal interest that will bring benefits both professionally and a great personal development in the face of all the adversities that have arisen.

1.2 Internship's Introduction

According to the Global 3D Mapping & Modeling Marker Analysis & Forecast "The global 3D mapping and modeling market to grow at a profound CAGR of around 9% during the forecast period of 2016-2020" (www.marketintelreports.com, 2016).

Due to the emergence of technologies such as Point Clouds and Photogrammetry, it is possible to recreate three-dimensional real spaces, with a level of detail very close to reality.

This solution aims to recreate a ~~property~~house, not with a huge level of detail, but rather an approximation through three-dimensional spatial coordinates.

After the modeling of the building, we intend to create different ways of ~~visualizing~~ viewing and navigating ~~the same~~, with the purpose of being incorporated into web environments for the current real estate market in Portugal.

1.3 Institution Introduction

In this section you will see a brief presentation of DevScope, and the impact of the project developed in the same organization.

1.3.1 Institution

DevScope is an IT company founded in September 2002, located at Rua Passos Manuel in Porto, acting on several fronts in the information technology sector such as:

- Business Intelligence;
- CRM;
- Collaboration (Sharepoint);
- System's Integration;
- Mobile Development;
- SaaS.

Being a Microsoft Gold Certified Partner, it also develops add-ons for Office / Excel, as well as evaluates and tests Microsoft products before being globally distributed in the market.

1.3.2 Projects role in the institution

The project comes as a succession of a company's project called "My 360 Tours", in which interlacing panoramic photographs, creates tours of immersive real estate.

To do this, DevScope intends to abandon "Virtual Tours" in support of this project so that it is a significant and highly scalable advancement towards the previous project, with the company's product team working in the future.

The subsequent development of this project can put the company in the map of the graphic applications as well as to have high profitability next to the real estate and tourism industry.

1.4 Contributions of this work

This solution is somewhat innovative, since the interior / exterior modeling is done mostly by 3D Laser Scanners, from which you get a complex and detailed Point Cloud. Being a Cloud Point heavy for working and storing, plus the scanning process is time-consuming requiring multiple scans for each division.

Modeling from panoramic photographs, is a process based on Machine Learning, much more cost and time effective, as well as the amount of information to be worked is much smaller.

The solution also has 3 view modes of the property, being a solution with strong interactivity between the user and the property. Allied to this, there's the ability to develop a customization tool, which offers the user, a chance to change to property and its furniture to their taste, while this was not possible with Point Clouds.

1.5 Organization of the report

This report is divided in 4 main chapters, being each one composed of multiple subchapters for a better organization and understanding of each topic. To better understand the report's structure, the main chapters are presented.

Chapter 2 – Context: The context chapter gives us a sight of the main problem while having a look into the business areas where the project plays a role, the state of the art and the solution of the problem portrayed.

Chapter 3 – Work Environment: This chapter portrays the planning and technologies behind the solution as well as how the project was developed in the institution.

Chapter 4 – Technical Description: On this chapter, a more in-depth description of the project's features and technical knowledge are described to better understand how each feature was accomplished.

Chapter 5 - Conclusions: In this final chapter, a reflection of the work done during the internship and the final solution is made. Also a balance between the accomplished goals and limitations and future work is done, as well as other work done during the internship that didn't play part of the solution.

2. Context

2.1 Problem

Conventional interior mapping / modeling methods use expensive equipment that is not always accessible to all users, and the scanning process is time-consuming and tedious.

To do this, with the success of mapping through panoramic photographs, such an act will become accessible to any user with a mobile device capable of taking panoramic photographs, the process being much faster and with much less information to be processed.

Regarding the viewer, this solution searches to bring property visualization to a whole new level giving the user a closer experience to the property independently of the device he is browsing on like shown in Figure 1.

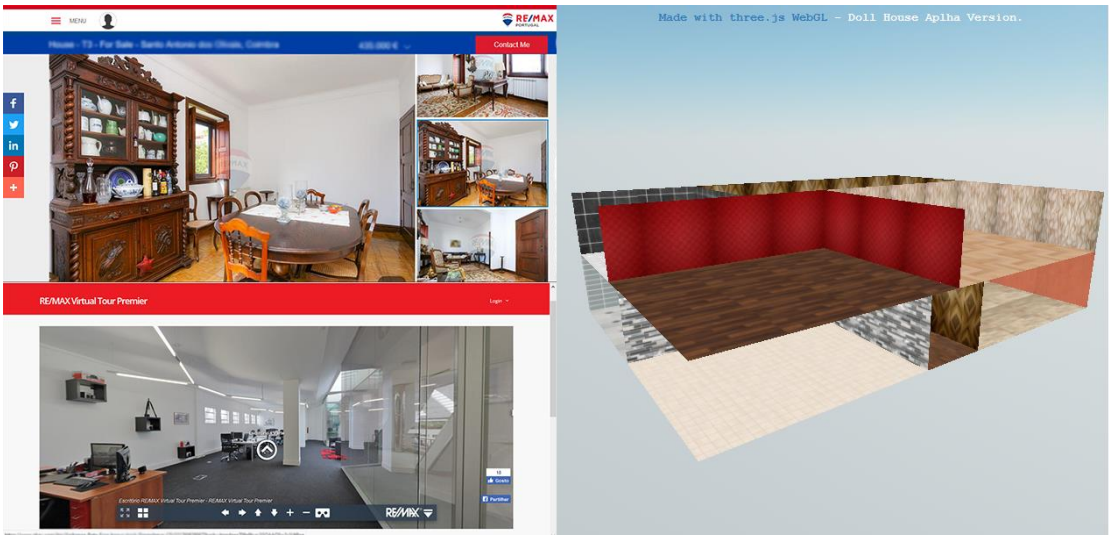


Figure 1- Current Viewers vs Doll House Viewer in Overview Mode (Alpha Version)

2.2 Solution

To achieve such experience, this solution incorporates 3 view modes of the property:

1. **Overview Mode*:*** General perspective of the property, with zoom features, as well as 360° rotation. Giving the user the ability of more overhead sight of the house, it allows the user to get a more comprehensive sense of the overall layout of the property.
2. **First-Person/Fly Mode:** Perspective in which the user is inserted inside the property with total control of movement and vision. In addition to that, the user is free to also move outside the property and surround it as if it were flying around it like the "Overview Mode" but with the freedom of movement characteristic of the First-Person Mode.
3. **Virtual Reality Mode:** Perspective adapted to current technologies of virtual reality. Compatible with the most headsets available in the market, the user navigates through the property emerged in a faithful and responsive virtual environment.

Such elaborate visualization solutions allow the user a much more interactive and close experience with the property, something that does not happen with the current solutions in the market, being also, the only one in the Portuguese real estate market to represent the property as a tri-dimensional model.

The view modes described above are all compatible with desktop and mobile devices and are organized like demonstrated in Figure 2.

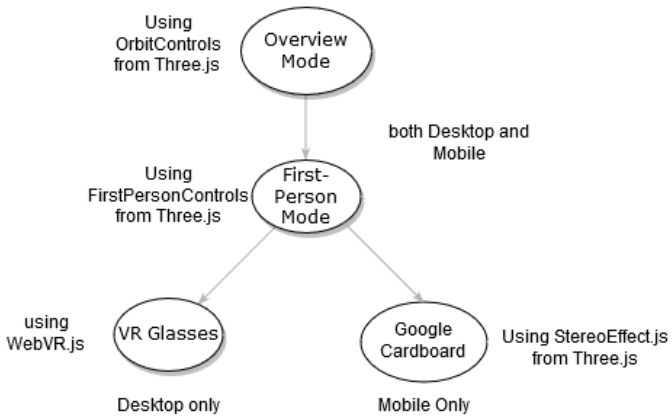


Figure 2- View Modes

2.3 Business Areas

The solution is aimed to the real estate market, as well as the 3D interior modelling market, since it combines the ability of modeling the interior of the property with the visualization of the same, it is applicable with success in both areas.

Being the next step for real estate visualization, we are going to do a brief analysis of the current & forecasted market.

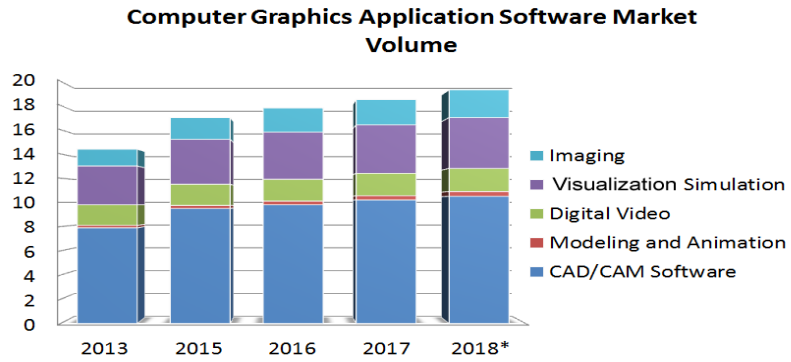


Figure 3-Computer Graphics Application Software Market Volume

(computergraphics-animation.conferenceseries, s.d.)

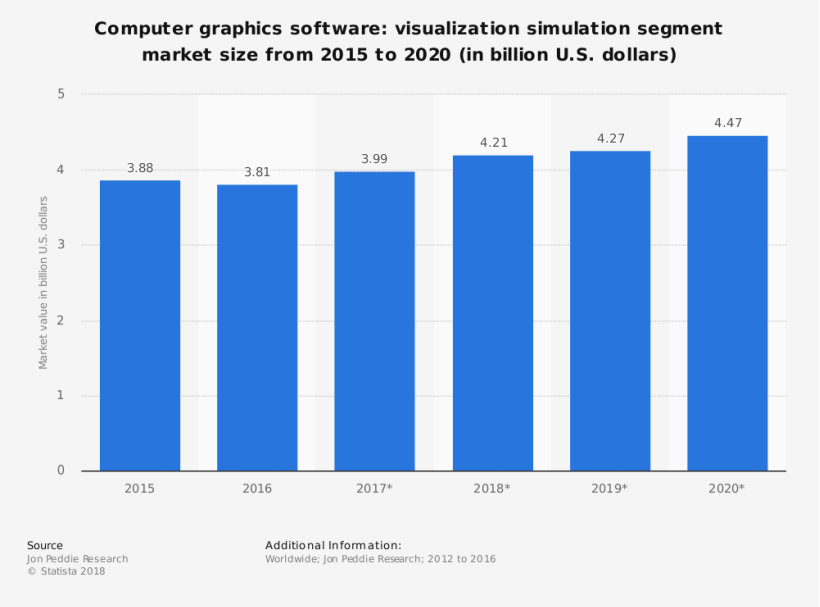


Figure 4- Visualization Simulation Market (2015-2020)

(statista.com, 2018)

The market for Computer Graphics Applications is in a slow and steady growth since 2013, as shown in the above Figure 3. The increasing market volume of Visualization Simulation software brings a good sight to the project's future profitability, as we can see in Figure 4Figure 5.

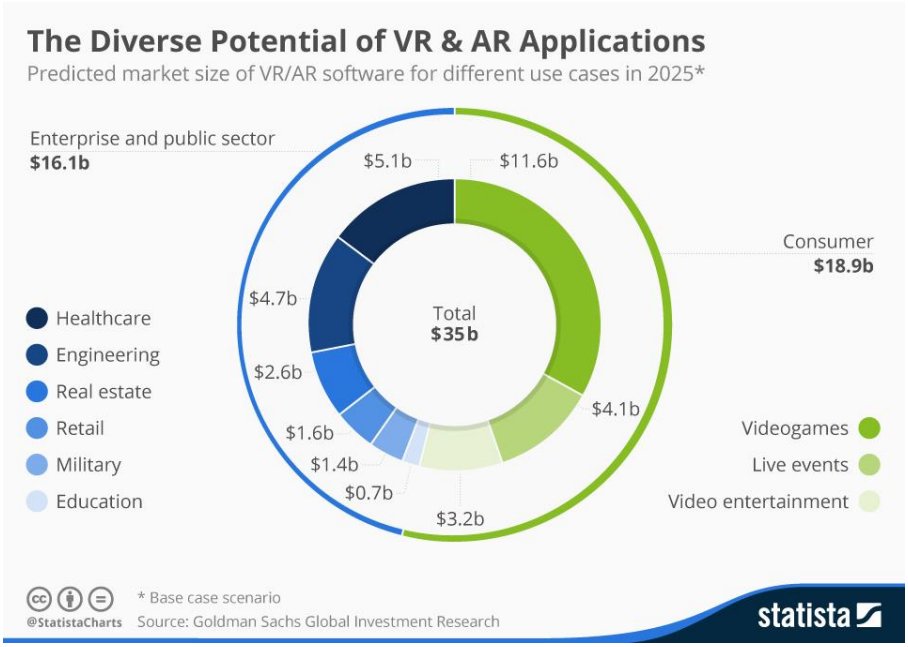


Figure 5-Diverse Potential of VR & AR Applications

(statista.com, 2018)

The Figure 5 evince that while the consumer entertainment, like the videogame industry plays a major role in the market size for VR & AR software forecasted for 2025, in the "Enterprise and Public Sector" of the market, the real estate industry reaches third place with a best-case scenario of 2.6 billion dollar, which the solution is already prepared to be part of.

2.4 State of the Art

This chapter presents all the mapping technologies proposed to the solution to model the house, and its factors that were balanced to find the best results.

In the following sub-chapters, we will analyze the positive and negative aspects of each mapping solution.

2.4.1 Photogrammetry

Photogrammetry is the art, science, and technology of obtaining reliable information about physical objects and the environment through processes of recording, measuring and interpreting photographic images and patterns of recorded radiant electromagnetic energy and other phenomena. (Sensing, 2015)

Due to the current abundance of mobile devices capable of taking panorama pictures, allied to its lightweight information, it is considered the best approach in terms of performance/quality aspect.

Working under Machine Learning technologies, the algorithms behind the process of interior mapping can be trained so that well-trained pattern recognition algorithms will result in faster and more accurate results, although the results achieved during the internship by my colleague are shown in Figure 6, they were not good enough to reliably recreate the house, so another way of model a house was used which is going to be introduced later in the report.

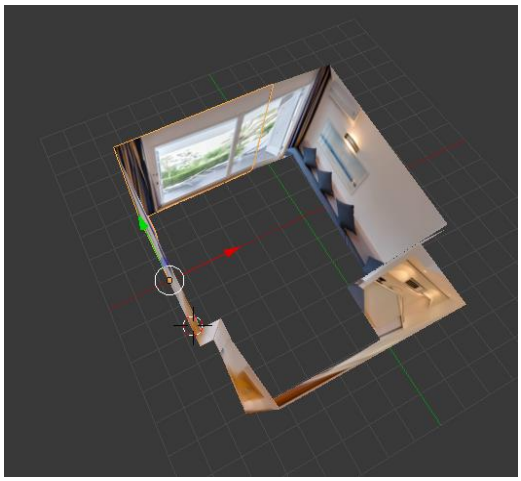


Figure 6-Indoor Mapping through Photogrammetry (alpha version)

(Maia, 2018)

2.4.2 3D Laser Scanning

"3D Laser Scanning is a non-contact, non-destructive technology that digitally captures the shape of physical objects using a line of laser light. 3D laser scanners create "point clouds" of data from the surface of an object. In other words, 3D laser scanning is a way to capture a physical object's exact size and shape into the computer world as a digital 3-dimensional representation." (laserdesign.com, s.d.)

2.4.2.1 How a Laser Scanner works?

The principle of the Laser Scanner is to shine a small light at a surface and measure the time it takes to return to its source.

"The actual calculation for measuring how far a returning light photon has travelled to and from an object is quite simple:

Distance = (Speed of Light x Time of Flight) / 2

Figure 7 – Laser Scanner Point Calculation formula

(lidar-uk.com, 2018)

The Laser Scanner fires rapid pulses of laser light at a surface, some at up to 150,000 pulses per second. A sensor on the instrument measures the amount of time it takes for each pulse to bounce back. Light moves at a constant and known speed so the LiDAR instrument can calculate the distance between itself and the target with high accuracy. By repeating this in quick succession the instrument builds up a complex map (also known as "Point Clouds") of the surface it is measuring." (lidar-uk.com, 2018)

Figure 8 – Laser Scanner Example

(shop.laserscanning-europe.com, s.d.)

2.4.2.2 What is a Point Cloud?

"A point cloud is a data base containing points in three-dimensional coordinate system. However, from the typical workflow perspective the only important thing is, that point cloud is a very accurate digital record of an object or space. It is saved in form of a very large number of points that cover surfaces of a sensed object.

Points in a point clouds are always located on the external surfaces of visible objects, because this are the spots, where ray of light from the scanner reflected from an object." (GRZEGORZ CIĘPKA, 2016).

In the Figure 9 below, we get a good example how a point cloud looks.

Figure 9 – Point Cloud of a Room

(we-get-around, 2015)

2.4.2.3 Point Cloud Optimization

"Such data sets can consist of hundreds of millions to billions of points and are therefore too large to be loaded and rendered at once.

Data sets with billions of points are not uncommon anymore. Processing and rendering these datasets are a challenging task that neither the memory nor the speed of today's hardware can handle in real time unless broken down into smaller parts." (Schütz)

Being a heavy set of data, points clouds are usually target for optimization.

Nowadays there's several ways to optimize a point cloud, below there's a few workable solutions to improve the performance issue.

2.4.2.3.1 Point Removal

"Most efficiently, you can remove a certain amount of points without losing detail" (Schütz)

To not loose detail while removing a certain amount of points, a common technique is to increase the size of each point, still giving the perception of a flat surface to the user's brain, like seen in Figure 10.

Figure 10 – Small vs Bigger points

(GRZEGORZ CIĘPKA, 2016)

"It is essential to understand that the point cloud is a set of individual, unrelated points with defined position and color. This makes point clouds quite easy to edit, display and filter.

Using individual, unrelated points is a key to point clouds usefulness, because points are objects that are easiest to handle large amount of. Computer do not have to care about scale, rotation and relations to other objects. Only position and color are things that matter for computation. This makes point clouds quite easy to edit, display and to filter data." (GRZEGORZ CIĘPKA, 2016)

2.4.2.3.2 Octree Representation

"The idea of this method is that only points inside the view frustum and up to a certain level of detail are loaded and rendered. Using adjustable point-count limits, even mobile and low-end desktop hardware can display these point clouds in real time or at least interactively." (Schütz)

"The octree is a hierarchical tree data structure from point cloud data. This enables spatial partitioning, downsampling and search operations on the point data set. Each octree node has either eight children or no children. The root node describes a cubic bounding box which encapsulates all points. At every tree level, this space becomes subdivided by a factor of 2 which results in an increased voxel resolution." (docs.pointclouds.org, 2017)

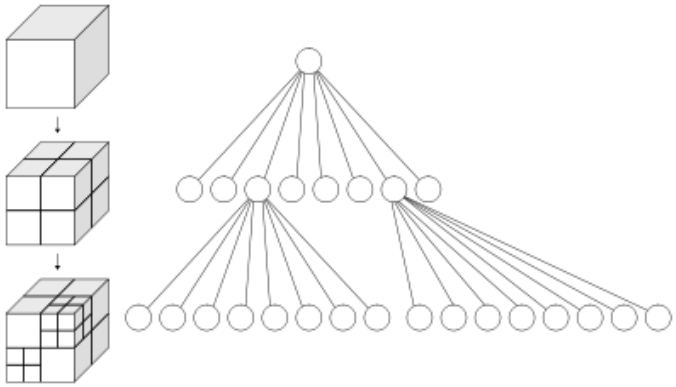


Figure 11 – Octree Bounding Box example

(Wikipedia, 2018)

Like shown in the Figure 11, the root node of the octree refers to the bounding box, which encapsulates all points, as we zoom in closer to a certain area of the point cloud, certain branches of the octree are loaded in order to save processing to that specific area inside the view frustum (area visible to the user). Figure 12, gives us a better understanding on how the point cloud is loaded and rendered according to how close the user sets the zoom.

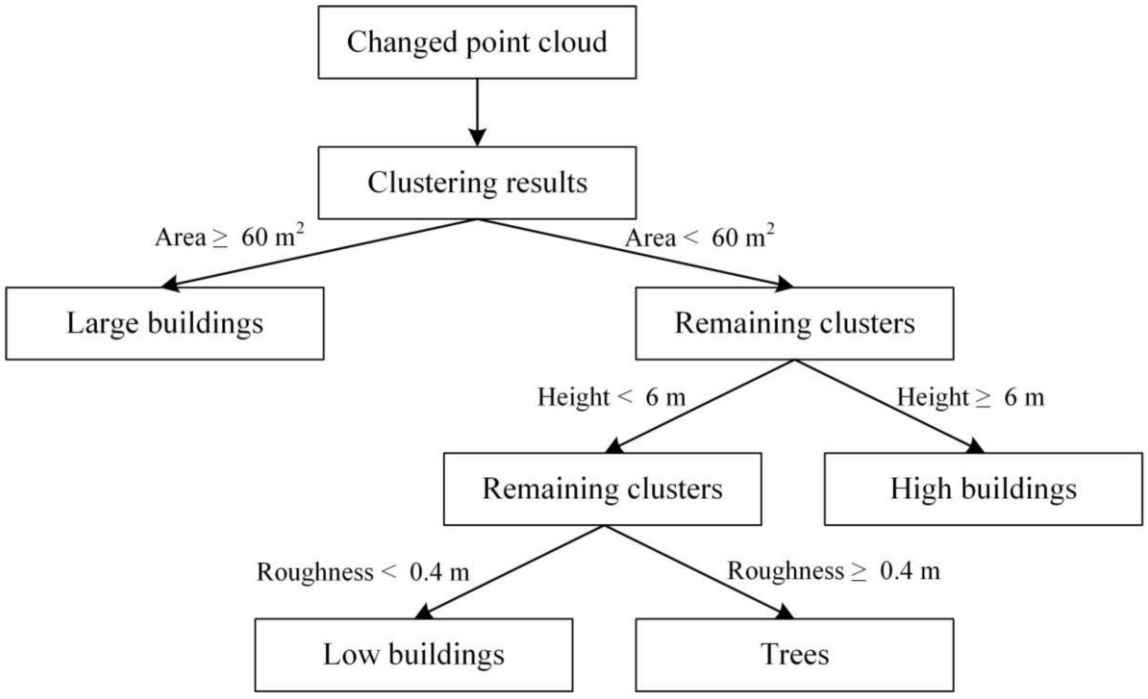


Figure 12 - Octree Working Example

2.4.2.4 SLAM Scanning

"Simultaneous Localization And Mapping (SLAM) technology was born in the robotics industry and is used by autonomous vehicles to concurrently map and navigate through an unknown environment. To do this, SLAM algorithms utilize information from sensors (often Lidar or imagery) to compute a best estimate of the device's location and a map of the environment around it." (geoslam, 2017)

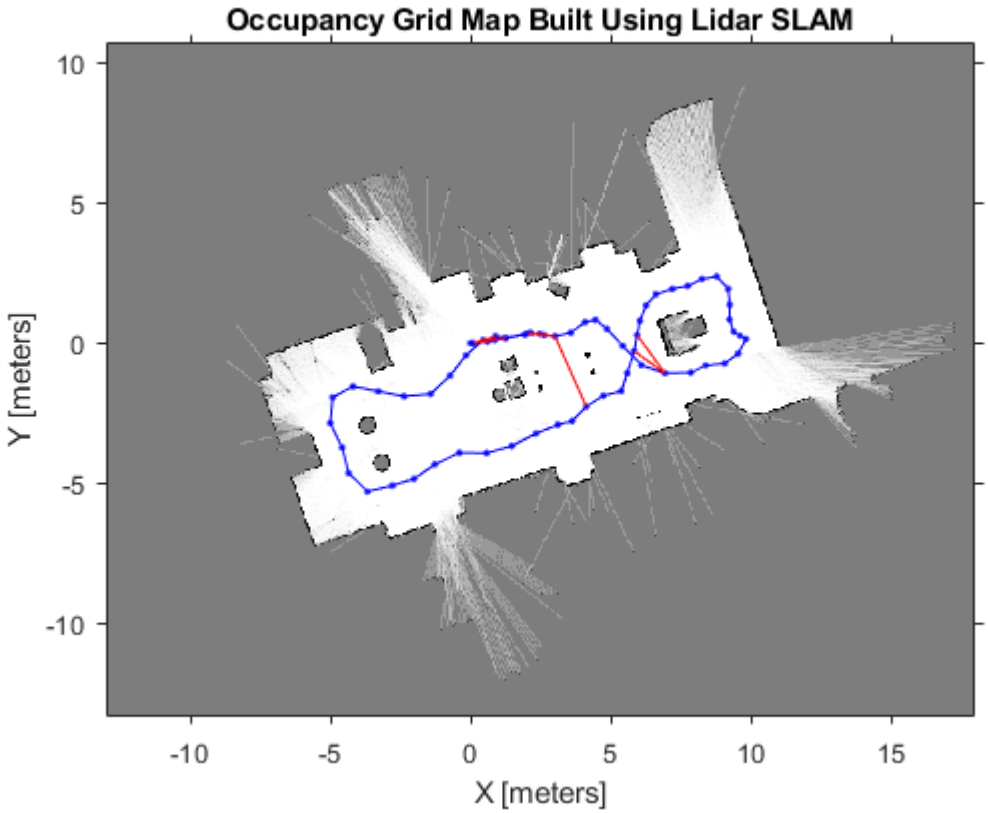


Figure 13 – Occupancy Grid Map Built Using Lidar SLAM

In the Figure 13 above, demonstrates how a SLAM system maps an entire room.

Although the SLAM technique is more practical to map the indoor of a house than mapping all the rooms one by one, the problem consists in the heaviness of the Point Clouds generated by these solutions.

Supported by the Figure 14 below, 3D Laser Scanning equipment is costly, and the scanning process can be meticulous enough to be intimidating so that not all real estate agents are qualified to perform and come up with a workable solution

3D Scanning	Infrared	Photogrammetry	LIDAR
+	Inexpensive	Balanced between cost and resolution of scans	Highest quality
-	Not as high quality	For a higher resolution, you need a more expensive camera	Most expensive option by far, also requires specialized knowledge and software and a relatively high-end workstation.

Figure 14 – 3D Scanning Pros and Cons

(Wheeler, 2017)

Besides that, each division must be scanned several times to have a usable result, has furniture always interfere with the scanning process as we can see in Figure 15, the rays of light don't scan anything beyond the furniture.

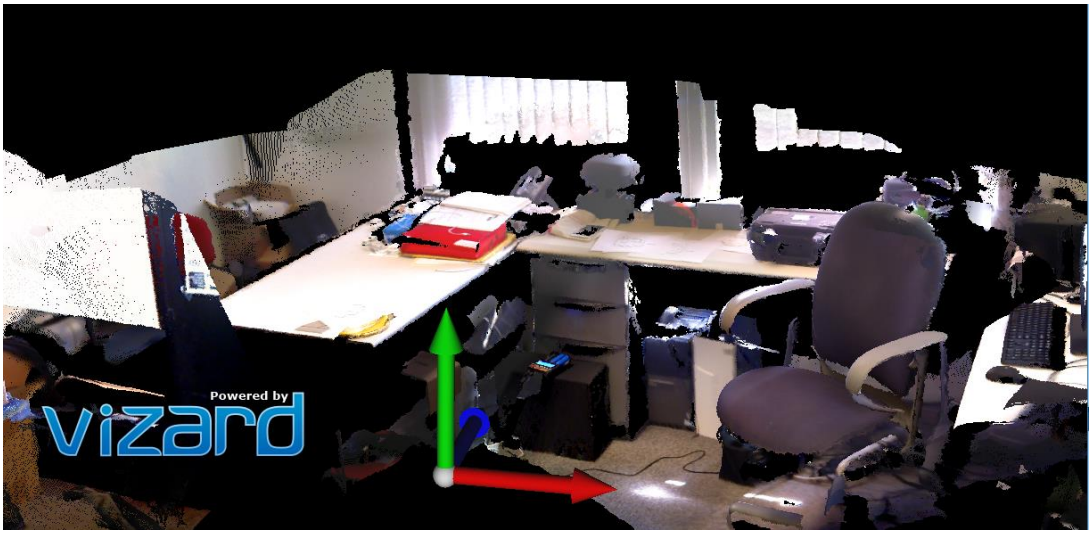


Figure 15 – Incomplete Point Cloud

(worldviz, 2017)

2.4.3 Depth Cameras

The last solution to be considered was the use of depth cameras on iOS/Android compatible mobile devices and independent depth-cameras like the XBOX Kinect's, since it allied the mobility and accessibility to the reliable 3D point cloud of the room.

But the results were not favorable due to the lack of performance of the depth cameras in generating points clouds, which are still a very embryonic technology in these type of devices like saw on Figure 16.

Figure 16 – 3D Interior Scanning using Kinect depth-camera

(Benchoff, 2012)

2.5 Solution's Scope

Subsequently, it is intended to fit a configuration tool, in which the entire property is customized to the user's taste, from the furniture to the room's walls, floor and ceiling.

Coupled with the virtual reality aspect, the configurator would offer an extremely immersive and interactive experience to the user, which in turn would have the option to keep the property faithful to the real model or customize it to taste.

With this tool, the application very easily adapts from a real estate visualizer to a virtual real estate maker with an approach to the decorative and design sector, in which the user builds and decorates in virtual environment all the real estate from scratch.

To take the house modelling step further, in a near future having non-rectangular shaped room and the possibility to have doors and windows, would be a great feature.

Depending on the Photogrammetry coordinates recognition to map the shape of the rooms as such as the doors and windows, the best solution to test this feature is to design a custom room from an array of coordinates and using a ConvexBufferGeometry in ThreeJS which, creates a convex geometry from a group of points in space.

Although creating the shape of the room, It is impossible to map the textures in the walls like it is done with BoxBufferGeometry, since it is an non predictable type of geometry, it has it is own Texture Mapping depending on the faces.

Using machine learning processes this can be achieved by identifying each face of the room and find the predominant color (usually the color on the wall/floor/ceiling) and mapping this color to the corresponding face of the convex geometry. When it comes to the furniture, machine learning processes identifies the type of furniture and its coordinates in the room. Having a collection of furniture objects, it automatically places the furniture in the ThreeJS Scene and the user later can change to a furniture object to this taste.

3. Work Environment

3.1 Work Methodology

The project was developed close to the organization's supervisor, with several informal brainstorming. The debate of ideas and projects in various disciplines is present every week until we landed on the solution presented in this document, reason why it carries out several studies in the area on the subject to approach the proposed problem.

Regarding the development process the Interactive and Incremental Development (IDD) was followed. This defends an execution of a project in several cycles (iterative) with small developments (incremental), always with the involvement of the client (in my case the supervisor), thus obtaining feedback and making the necessary adjustments along the way for a desired solution.

Keeping the client close to the development process, there is a low chance of future major changes in the final product.

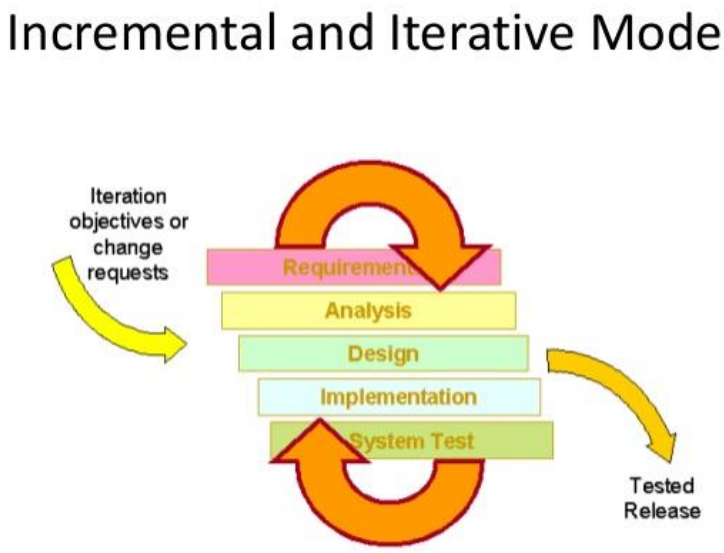


Figure 17-Incremental and Interactive model

3.2 Work Planning

The initial planning of this work was carried out according to the following steps: analysis and survey of requirements and division of the implementation phases.

The analysis and requirements analysis step consisted in evaluating the main components and functionalities of the solution to be developed and deciding which technologies to use. The decisions taken in this section were made through informal meetings with the supervisor of the organization where the functional requirements of the project were reviewed, and the use cases of the project were presented.

Developing next to the supervisor, it was easy to adapt the project to the various brainstorms made, in which most cases, the project underwent crucial changes.

Date	**Participants**	**Local**	**Topics**
25-03-2018	Eng. João Paulo Pereira, Diogo Azevedo and Luís Maia	ISEP	Completion of the formal minutes of meeting and establishment of the content of the internship
23-04-2018	Eng. João Paulo Pereira, Diogo Azevedo and Luís Maia	ISEP	Follow-up meeting
24-05-2018	Eng. João Paulo	ISEP	Follow-up meeting

Table 1- ISEP Supervisor Meetings

3.3 Technologies used

3.3.1 WebGL

The solution is based on WebGL, which benefits from all the functionality of the OpenGL API, in web environments. It is currently supported by all current browsers, and has frequent updates to its frameworks, which are constantly evolving.

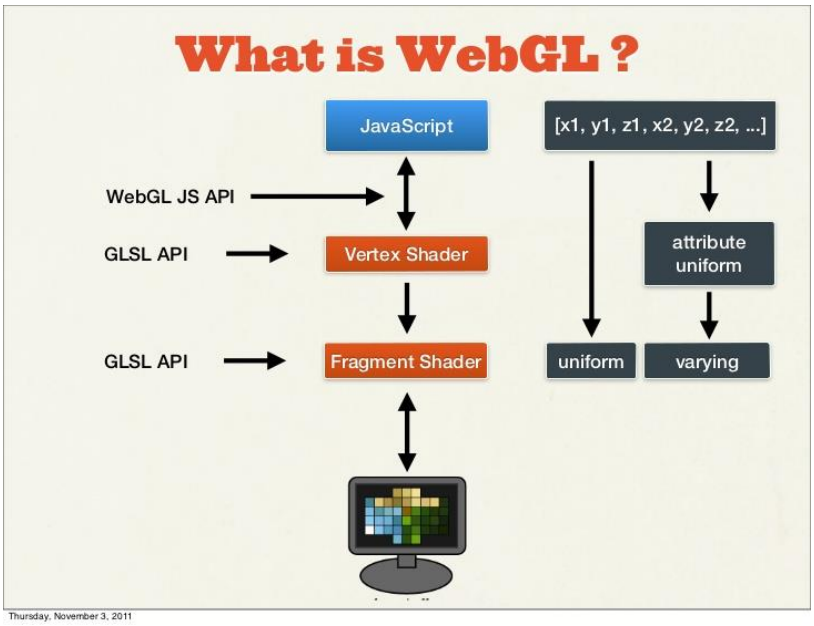


Figure 18-What is WebGL?

(Greggman, s.d.) WebGL runs on the GPU on your computer. As such you need to provide the code that runs on that GPU. You provide that code in the form of pairs of functions. Those 2 functions are called a vertex shader and a fragment shader, and they are each written in a very strictly typed C/C++ like language called GLSL. (GL Shader Language).

(Greggman, s.d.) A vertex shader's job is to compute vertex positions. Based on the positions the function outputs WebGL can then rasterize various kinds of primitives including points, lines, or triangles. When rasterizing these primitives, it calls a second user supplied function called a fragment shader. A fragment shader's job is to compute a color for each pixel of the primitive currently being drawn.

It was used the Three.js framework because it had a wide range of importers of various types of 3D objects, as well as being a strongly active framework by the community. Although Three.js does not have integrated Audio, Networking and Physics, those aspects were not relevant to the solution in hands.

3.3.2 Three.js

"Three.js allows the creation of Graphical Processing Unit (GPU)-accelerated 3D animations using the JavaScript language as part of a website without relying on proprietary browser plugins. This is possible due to the advent of WebGL." (Group, 2011)

"High-level libraries such as Three.js or a number of other libraries make it possible to author complex 3D computer animations that display in the browser without the effort required for a traditional standalone application or a plugin." (Crossley, 2010).

It supports such computer graphics features as loading Geometry and Textures, lightning and shading systems, cameras, animation and much more.

3.3.2.1 Geometry

Regarding the Geometry, Three.js has two types of 3D Geometry which are Geometry and Buffer Geometry.

"Buffer Geometry is an efficient representation of mesh, line, or point geometry. Includes vertex positions, face indices, normals, colors, UVs, and custom attributes within buffers, reducing the cost of passing all this data to the GPU." (Threejs, 2018)

"Geometry on the other side is a user-friendly alternative to Buffer Geometry. Geometries store attributes (vertex positions, faces, colors, etc.) using objects like Vector3 or Color that are easier to read and edit, but less efficient than typed arrays." (Threejs.org, 2018).

Buffers are nothing less than fast memory pools, which helps to reduce memory cost and CPU cycles in render time.

3.3.2.2 Materials

If the Geometry is the skeleton of the object, defining the shape, then the Material is the skin and how it reacts to lightning and textures.

As described in ThreeJS official website, Materials describe the appearance of objects. They are defined in a (mostly) renderer-independent way, so you don't have to rewrite materials if you decide to use a different renderer. (threejs, 2018)

There are 4 principal types of Materials available in Three.js:

1. Mesh Basic Material: The most basic material is the MeshBasicMaterial. You can pass a color in as a parameter to get a solid colored object, which has no shading. You can also adjust the opacity by passing in the opacity as a parameter with a value from 0 to 1 and setting transparent to true. (cjgammon, 2016)
2. Lambert Material: This allows our material to respond to lights MeshLambertMaterial will give our geometry shading with a dull surface. Lambert is a common material in most 3D applications. Just like before we can adjust the color. It also has an emissive property which adds a glow like color to the material. (cjgammon, 2016)
3. Phong Material: Like Lambert, Phong Material responds to lights but adds a metallic luster to the surface, reflecting light with more intensity. You can also add a specular color and adjust the shininess property of this material to adjust the intensity of the light reflection. (cjgammon, 2016)
4. Normal Material: Normal Material colors the faces of the mesh differently based on the face's normal or what direction they are facing. (cjgammon, 2016)

In the Figure 19, we can visually see the differences between each Material, and how it reacts to light and shading.

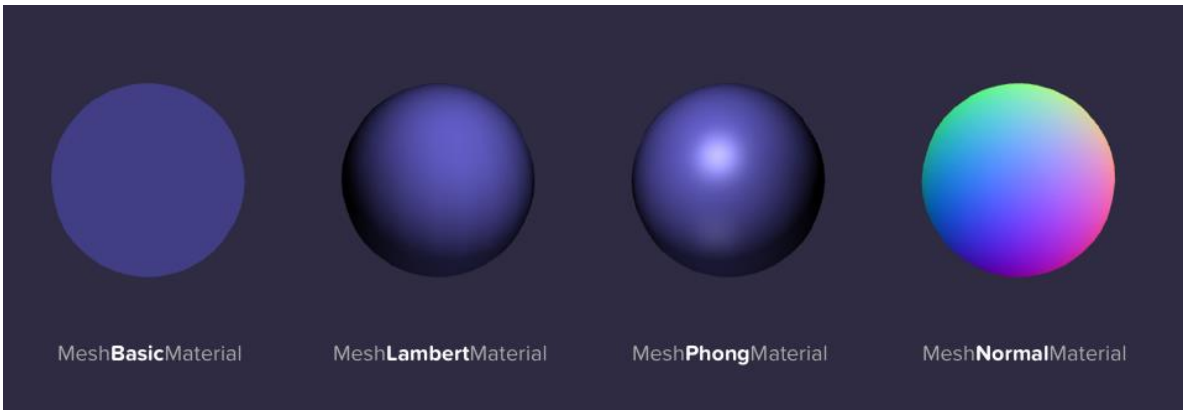


Figure 19 – ThreeJS Materials

(cjgammon, 2016)

3.3.2.3 Meshes

Like shown in Figure 20, a mesh is the combination of a geometry and a material and is placed in the scene to represent an object.

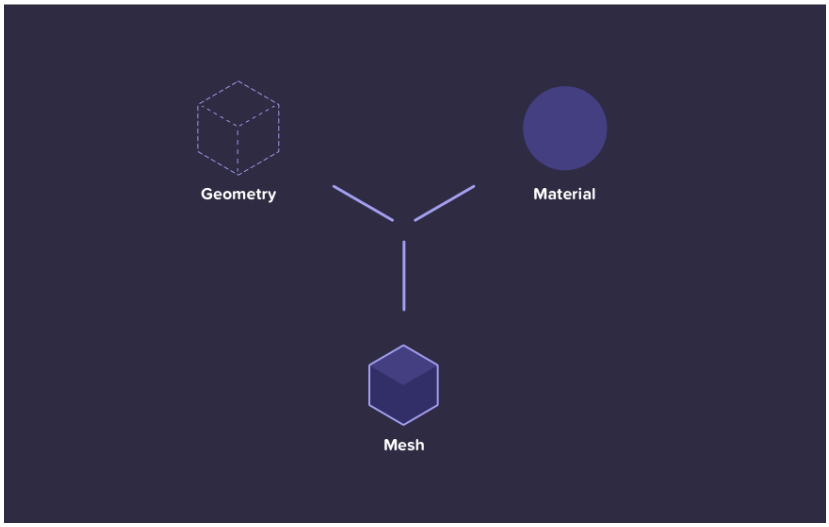


Figure 20-What is a Mesh? (Coppero, 2017)

3.3.2.4 Cameras

Three.js, along with the majority of the 2D and 3D graphical engines, supports two main camera systems, Perspective and Orthographic Cameras.

Perspective Cameras are designed to mimic the human eye sight being the most common projection mode used in 3D scenes.

“In Orthographic Cameras an object's size in the rendered image stays constant regardless of its distance from the camera.” (threejs, 2018). Orthographic cameras are mostly used for rendering 2D Scenes and user interfaces.

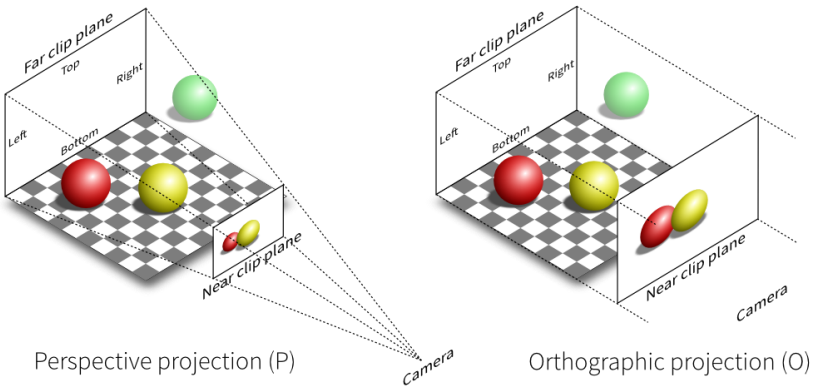


Figure 21-Perspective vs Orthographic Projection

Besides their differences both projections, like shown above in Figure 21, have two things in common, a Far Clipping Plane and a Near Clipping Plane. As the name indicates the Far Clipping Plane is the farthest point relative to the camera that it will render, and the Near Clipping Plane the closest. Anything beyond and outside the Far Clipping Plane, or between the Near Clipping Plane and the Camera, will not be rendered.

3.3.2.5 Lights

When it comes to the light system, supports 7 unique types of light, which each has its own way to represent different sources of light we see in our world.

1. Ambient Light: This light globally illuminates all objects in the scene equally. This light cannot be used to cast shadows as it does not have a direction. (Three Js, 2018)
2. Directional Light: A light that gets emitted in a specific direction. This light will behave as though it is infinitely far away, and the rays produced from it are all parallel. The common use case for this is to simulate daylight; the sun is far enough away that its position can be infinite, and all light rays coming from it are parallel. This light can cast shadows. (Threejs, 2018)
3. Hemisphere Light: A light source positioned directly above the scene, with color fading from the sky color to the ground color. This light cannot be used to cast shadows. (Threejs, 2018)
4. Point Light: A light that gets emitted from a single point in all directions. A common use case for this is to replicate the light emitted from a bare lightbulb. This light can cast shadows. (Threejs, 2018)
5. RectArea Light: RectArea Light emits light uniformly across the face a rectangular plane. This light type can be used to simulate light sources such as bright windows or strip lighting. There is no shadow support yet and to use, an include of the "RectAreaLightUniformsLib" onto the scene. (Threejs, 2018)
6. Spot Light: This light gets emitted from a single point in one direction, along a cone that increases in size the further from the light it gets. This light can cast shadows. (Threejs.org, 2018)

In the following figures (12,13,14) we can see some examples of these types of lightning.

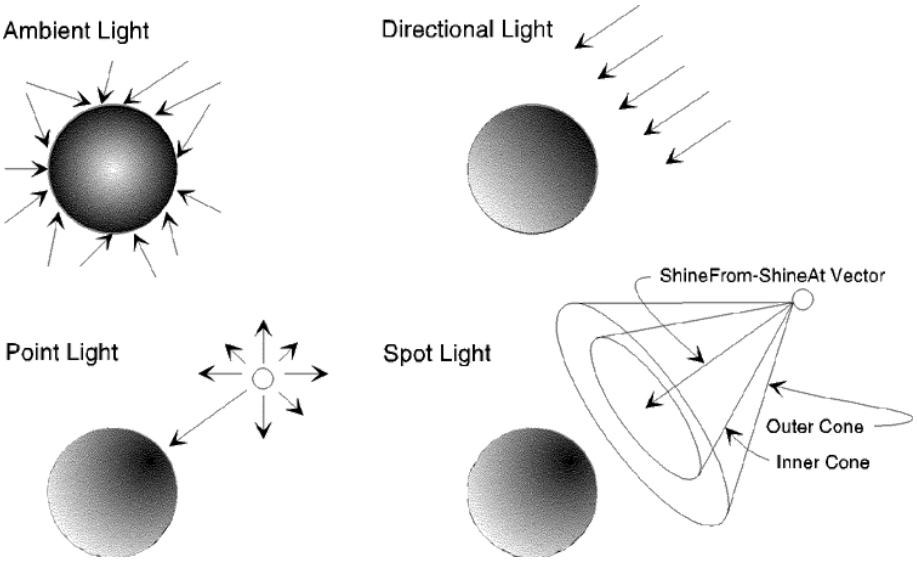


Figure 22-Ambient, Directional, Point and Spot Light

(okino.com, 2013)

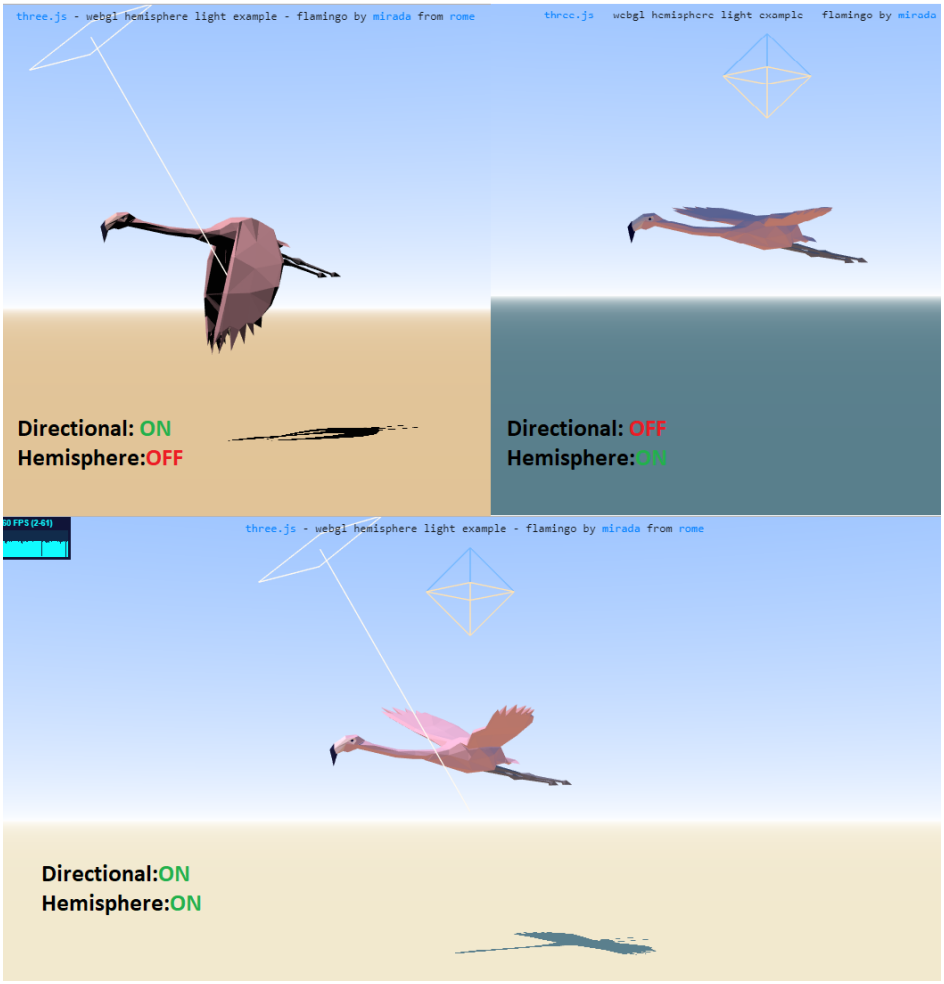


Figure 23 – Directional vs Hemisphere Light

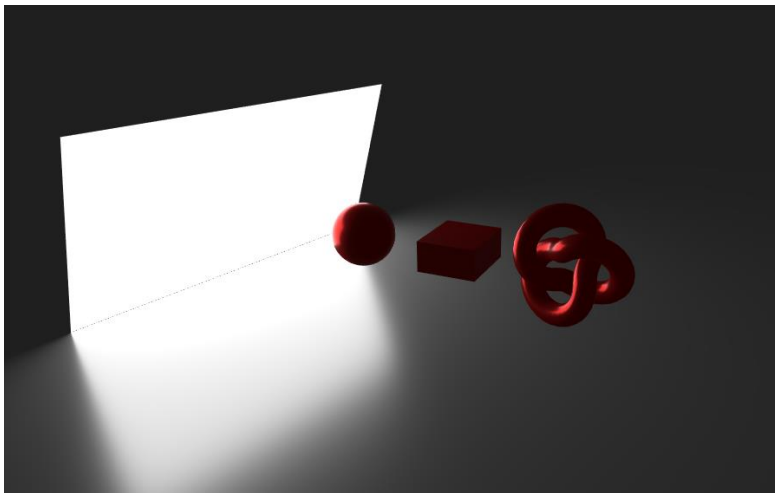


Figure 24 – RectArea Light

3.3.2.6 WebGLRenderer

The WebGLRenderer draws the scene and all its content onto a WebGL enabled canvas. This renderer should be used for browsers that support WebGL.

"The term rendering refers to the calculations performed by a 3D software package's render engine to translate the scene from a mathematical approximation to a finalized 2D image. During the process, the entire scene's spatial, textural, and lighting information are combined to determine the color value of each pixel in the flattened image." (Slick, 2018)

There are two types of rendering types, Real-Time Rendering and Pre-Rendering:

1. Real-Time Rendering: Real-time rendering is used most prominently in gaming and interactive graphics, where images must be computed from 3D information at an incredibly rapid pace. Because it is impossible to predict exactly how a player will interact with the game environment, images must be rendered in "real-time" as the action unfolds. (Slick, 2018)

2. Pre-Rendering: Pre-rendering is used in situations where speed is less of an issue, with calculations typically performed using multi-core CPUs rather than dedicated graphics hardware. Offline rendering is seen most frequently in animation and effects work where visual complexity and photorealism are held to a much higher standard. Since there is no unpredictability as to what will appear in each frame, large studios have been known to dedicate up to 90 hours render time to individual frames. (Slick, 2018)

Besides the rendering types, there are different techniques used for rendering. With advantages and disadvantages on its own, each technique is used when is better applied:

1. Scanline (or Rasterization): Scanline rendering is used when speed is a necessity, which makes it the technique of choice for real-time rendering and interactive graphics. Instead of rendering an image pixel-by-pixel, scanline renderers compute on a polygon by polygon basis. Scanline techniques used in conjunction with precomputed (baked) lighting can achieve speeds of 60 frames per second or better on a high-end graphics card. (Slick, 2018). In the Figure 25 below, we can visually see how the scanline algorithm works.

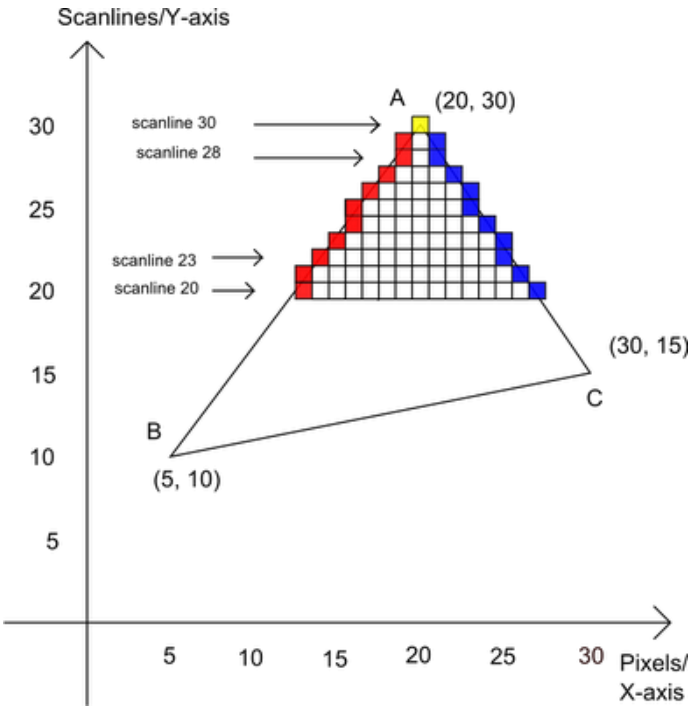


Figure 25-Scanline Algorithms

(@darkoman, 2011)

2. Raytracing: In raytracing, for every pixel in the scene, one or more rays of light are traced from the camera to the nearest 3D object. The light ray is then passed through a set number of "bounces," which can include reflection or refraction depending on the materials in the 3D scene. The color of each pixel is computed algorithmically based on the light ray's interaction with objects in its traced path. Raytracing is capable of greater photorealism than scanline but is exponentially slower. (Slick, 2018)

   Figure 26 gives us a better understanding on how RayTracing Rendering works.

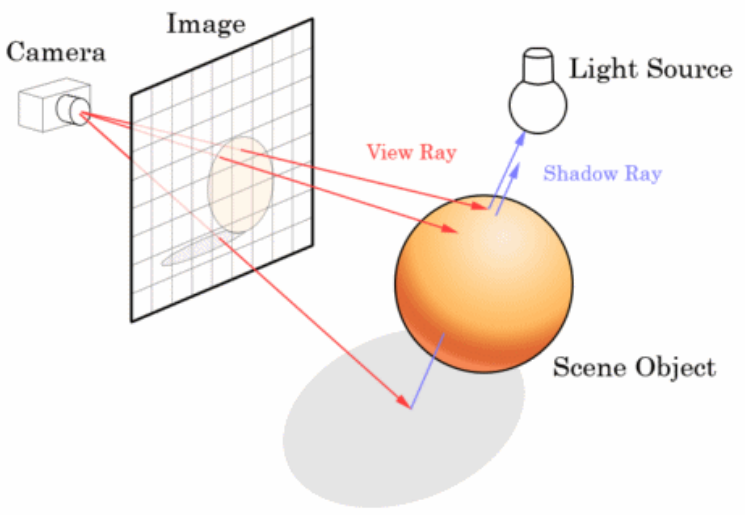


Figure 26-Raytracing Algorithm

(hardmath123, 2015)

3. Radiosity: Unlike raytracing, radiosity is calculated independent of the camera, and is surface oriented rather than pixel-by-pixel. The primary function of radiosity is to more accurately simulate surface color by accounting for indirect illumination (bounced diffuse light). Radiosity is typically characterized by soft graduated shadows and color bleeding, where light from brightly colored objects "bleeds" onto nearby surfaces. (Slick, 2018).

   Below, in Figure 27, we can see how the Radiosity Algorithms works through the multiple passes, achieving an incredible result when it comes to lightning and shading issues, giving it a Photorealistic feel to the renderer world.

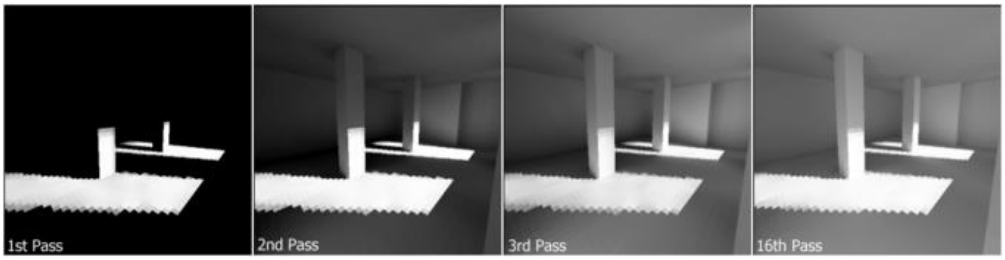


Figure 27- Radiosity Algorithms progression

(Radiosity - Wikipedia, 2018)

"In practice, radiosity and raytracing are often used in conjunction with one another, using the advantages of each system to achieve impressive levels of photorealism." (Slick, 2018)

3.3.2.7 Scene

"Scenes allow you to set up what and where is to be rendered by three.js. This is where you place objects, lights and cameras." (Threejs.org, 2018)

Figure 28, gives a more deep and clear understading on how the different roles of ThreeJS act together to make the Scene.

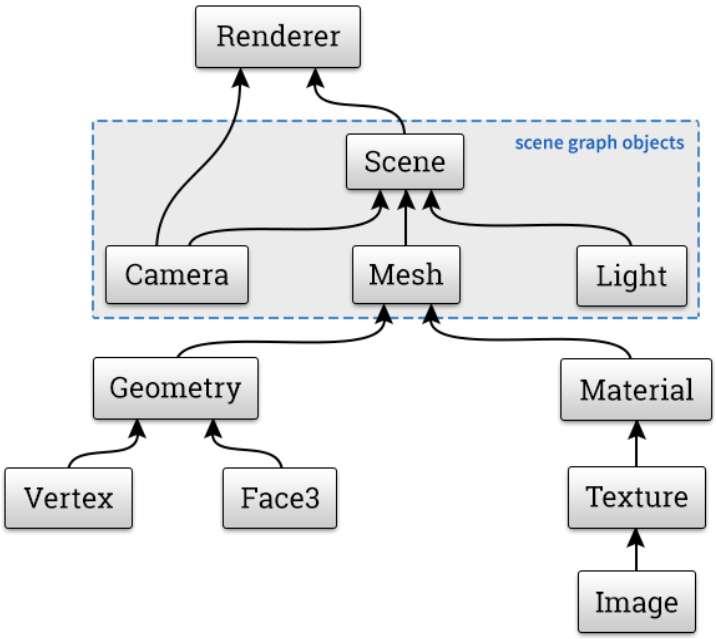


Figure 28-Scene Graph

(Lyon, s.d.)

3.3.3 Virtual Reality

Virtual Reality was used for desktop version, using WebVR.js allied to WebGL's Three.js framework. The Virtual Reality glasses that the solution was developing with was the "Lenovo Explorer" which works under Windows Mixed Reality and has 6 degrees of freedom.

Working under Windows Mixed Reality its only compatible when using the Microsoft Edge browser, like proven in.

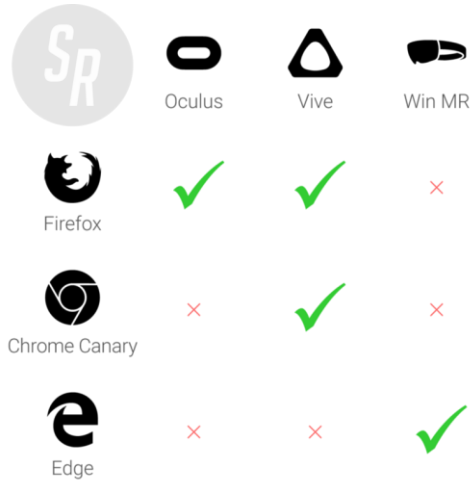


Figure 29-Headset's Browser Compatibility

(Smith, 2017)

"About the Depths of Freedom, it refers to the number of basic ways a rigid object can move through 3D space. There are six total degrees of freedom.

3DOF means we can track rotational motion but not translational. For the headset, that means we can track whether the user has turned their head left or right, tilted it up or down, or pivoted left and right.

6DOF means we can additionally track translational motion. That means we can track whether the user has moved forward, backward, laterally, or vertically." (Google, 2018)

The Figure 30 below, ilustrastes the differences between these two types of Virtual Reality headsets.

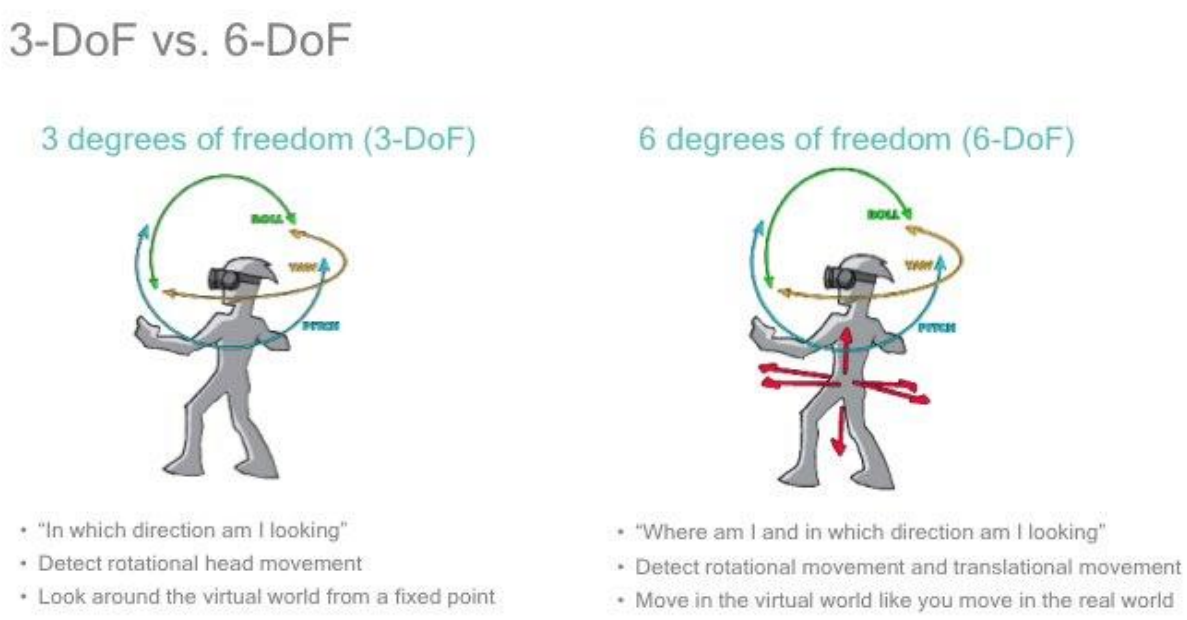


Figure 30 - 3DOF vs 6 DOF

Usually content for 3DOF systems is produced using 360º cameras and panoramic pictures, because it captures all the environment from a single point in space.

6DOF systems, on the other hand, need a 3D space environment because the user can move inside and interact with the elements, not being stuck in one point in the 3D virtual world. Because of that, such environments usually are generated in game engines and CAD software's.

3.3.3.1 WebVR API

"WebVR is a technology that allows a website author to add VR functionality to a page. The WebVR API is used by a page to display 3D content (such as 360-degree video, or a 3D model, or a 3D game) to the entirety of your headset." (Aaron & Zeller, 2017)

Using WebGL with the necessary camera settings and device interactions (such as controllers or point of view), the API works under the following path:

1. Request a list of the available VR devices.
2. Checks to see if the desired device supports the presentation modes the application needs.
3. If so, application advertises VR functionality to the user.
4. User performs an action that indicates they want to enter VR mode.

5. Request a VR session to present VR content with.
6. Begin a render loop that produces graphical frames to be displayed on the VR device.
7. Continue producing frames until the user indicates that they wish to exit VR mode.
8. End the VR session.

3.3.4 Google Carboard

Google Cardboard works under a technique called Stereoscopy, more precisely, performs a stereogram which consists in creating a 3D illusion starting from a pair of 2D images.

“The easiest way to enhance depth perception in the brain is to provide the eyes of the viewer with two different images, representing two perspectives of the same object, with a minor deviation equal or nearly equal to the perspectives that both eyes naturally receive in binocular vision.” (Wikipedia).

In the Figure 31 below, we can see an example of a Google Cardboard headset, although there are plenty of other headsets manufactured by other brands that are built in plastic.

Figure 31 - Google Cardboard headset

(store.google.com, s.d.)

4. Technical Description

In the following chapter a more detailed and technical description of the solution features is presented to have a better and deep understanding on how the result was reached.

4.1 House Modelling

Being an application resulting from the definition of interior spaces through panoramic photographs, the house modelling is made through a JSON file in which contains all the information about each room.

```
{
    "size":[3,1,5],
    "position":[2,0,0],
    "furniture":[
        {"path":"./models/furniture/livingroom/tv/","
       ,{"path":"./models/furniture/livingroom/sofa/"
       ,{"path":"./models/furniture/livingroom/center
       ,{"path":"./models/furniture/livingroom/dining

    ],
    "front": "./images/livingroom/front.jpg",
    "back": "./images/livingroom/back.jpg",
    "left": "./images/livingroom/left.jpg",
    "right": "./images/livingroom/right.jpg",
    "top": "./images/livingroom/top.jpg",
    "bottom": "./images/livingroom/bottom.jpg"
},
{
```

Code Sample 1-Json Room Structure

As we can see in Code Sample 1, a room a set of different parameters:

1. Size: working under the following format [x, y, z], the size array gives us the width, height and length of the room respectively, that will later be the on parameters for the BoxBufferGeometry function.
2. Position: Being a point in a 3D World Space, it is a 3-position array that gives us the coordinates to place the BoxBufferGeometry.
3. Furniture: The furniture array, is the collection of furniture that belongs to that room.

```
"furniture":[
    {"path":"./models/furniture/livingroom/tv/",
     "object":"tv.obj","material":"tv.mtl",
     "position":[2,-0.5,-2.4],
     "scale":0.47,
     "rotation":[0,0,0]
     }
```

Code Sample 2-Furniture Json structure

As Code Sample 2 demonstrates, a furniture is an array of objects, in which each furniture has it is own:

a. Path: folder that holds all the furniture data.

b. Object: path to the geometry of the furniture

c. Material: path to the material of the furniture

d. Position: 3D world space coordinate that tells where the furniture should be

e. Scale: Float number that scales the 3D Object to match the desired proportions according to the room.

f. Rotation: Array of float numbers that sets the furniture's desired rotation degree according to each vector [x, y, z] respectively.

4. Front/Back/Left/Right/Top/Bottom: represents the path to each image that is drawn on each face(wall) of the BoxBufferGeometry(room), being "Top" the ceiling, "Bottom" the floor, and "Left"," Right"," Front"," Back" the walls.

Being the walls with different proportions, depending on the dimensions of the room, some textures were distorted. To solve this, all images were cut to 4:4 (Square like), having them repeated according to the proportion of each wall. To do this, it was used the "THREE.RepeatWrapping" property.

```
var back = textureLoader.load( room[i].back);//back
back.wrapS = THREE.RepeatWrapping;
back.wrapT = THREE.RepeatWrapping;
back.repeat.set( room[i].size[0], room[i].size[1] );
materials.push(new THREE.MeshBasicMaterial( { map:back , side:THREE.BackSide} ) );
```

Code Sample 3 – Texture Wrapping

Like shown in the *Code Sample 3* above, it was used wrapS to wrap horizontally, and WrapT to wrap it vertically, repeating according to the room size, in the *Code Sample 3*, it was the back wall, so it depended on the height and width of the room.

To build the 3D obj of the room, it was used the BoxBufferGeometry property of Three.js in which giving the width, length and height (x, y, z in the "size" respectively) of the room, it creates a 3D box, representing the room in the WeblGL scene.

The BoxBufferGeometry supports a feature to map textures into each face, hence the name of each texture being the corresponding face being mapped.

To being able to see the interior of the house from the outside, the texture was only rendered in the backside of the face using the "THREE.Backside" property from the Material class.

To import the furniture, it was used the OBJ+MTLLoader provided by Three.js.

Being the OBJ file the vertices and faces that make the geometry of the 3D Object, the MTL file maps the textures onto the geometry rendering the furniture as shown below.

Figure 32-Kitchen Furniture Demo

4.2 Desktop vs Mobile device detection

As mentioned earlier in the essay, the solution has 3 view modes for both the desktop version and the mobile version, but before instantiating the camera and it is correspondent controllers, it is crucial to know if the user is accessing the website on a computer, or on a mobile device such as a tablet or smartphone.

```
if (/Mobi|iPhone|iPad|iPod|Android/i.test(navigator.userAgent)) {/
    mobile=true;
}
```

Figure 33-Mobile Detection Code Sample

The Navigator object contains all the information related to the browser.

“The userAgent property returns the value of the user-agent header sent by the browser to the server.

The value returned, contains information about the name, version and platform of the browser.” (w3schools, s.d.)

4.3 Camera Switching

The Camera switching process is made by double-clicking or double-tapping (in mobile devices).

In a UI/UX approach, this is the best solution to the problem of switching between view modes, because in the mobile devices, it removes the need to have extra buttons o the screen which would take screen space, and most importantly, the user attention from the house viewer.

To handle the double-tap event in the mobile devices it was used the “Hammer.js” Library.

For the Desktop version, it was used the “dblclick” event listener, which is fired when the mouse is clicked on the WebGL scene.

All 3 Camera modes for both versions are stored in an array, so when the event of camera switching is fired, they cycle through the array infinitely. Each mode when called knows through a Boolean, if the user is using a desktop or a mobile device and instantiates and renders the correct camera and it is corresponding controller.

When the user enters the application, and while the house is loaded and rendered, the user is presented with a splash-screen giving him the instructions of the controls as well as the camera cycle process.

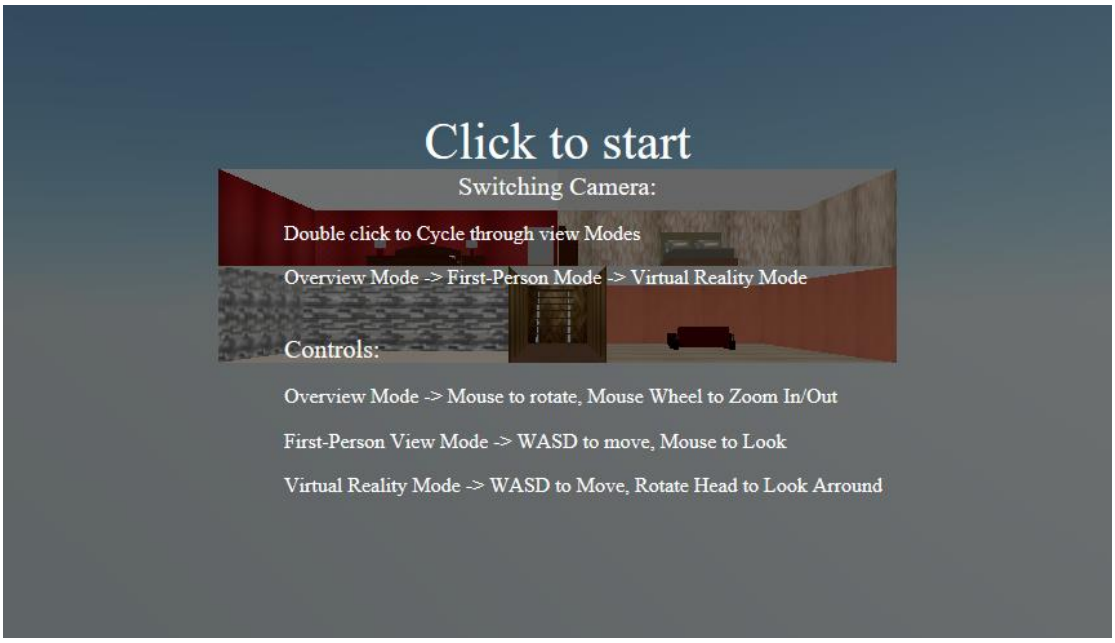


Figure 34-Splash-Screen Instructions

4.4 Overview Mode

The Overview Mode consists in a general view of the house with rotation and zoom features.

This was made possible using the OrbitControls camera controller from Three.js framework, which automatically supports both desktop and mobile controls.

Consisting in rotating the camera and zooming around a fixed point (in this case the origin point [0,0,0]), is represents one of the most effective and simplest view modes possible to show any object, since the user can see the object from afar, having a broad view which helps to better recognize the house as a first interaction.

Figure 35-Overview Mode

4.5 First-Person View Mode

The first-Person view mode consists in flying in and outside the house without wall collision (collision was not implemented to have more freedom of movement and for the impossibility of having doors on the BoxBufferGeometry object).

It was used different controllers this time for the desktop and mobile versions, because there wasn't one that would support both keyboard and touch controls.

For the desktop version, it was used the FirstPersonControls from Three.js framework, using the classic W, A, S, D for movement and the mouse for camera rotation.

For the mobile version, the movement is made with a virtual joystick and the camera rotation works on sliding on the right side of the screen. This was made possible using the touch-fps-controls plugin for three.js.

In this view mode, the user has a more personal experience with the virtual property and explore it in a more realistic way.

Figure 36 – First-Person View Mode

4.6 Google Cardboard (Mobile VR)

Using the latest VR technology for mobile devices, Google Cardboard offers a general 3D overview of the house.

Without any possibility of having control in whatsoever, the best option was to auto-rotate the house using the Orbit Controls Controller from Three.js, while having the 3-dimensional feel to it as we see in the Figure 37 below.

This was the best approach since usually Google Cardboard is a 3 degree of freedom headset, which performs better in cube/Sphere Panoramic pictures and videos where the user looks around in the virtual world from a fixed point, in this case would be in a center of a room.

While having several rooms in the house, room switching implied taking out the mobile device from the headset, touch a button to switch to the center of the desired room and then put it back on.

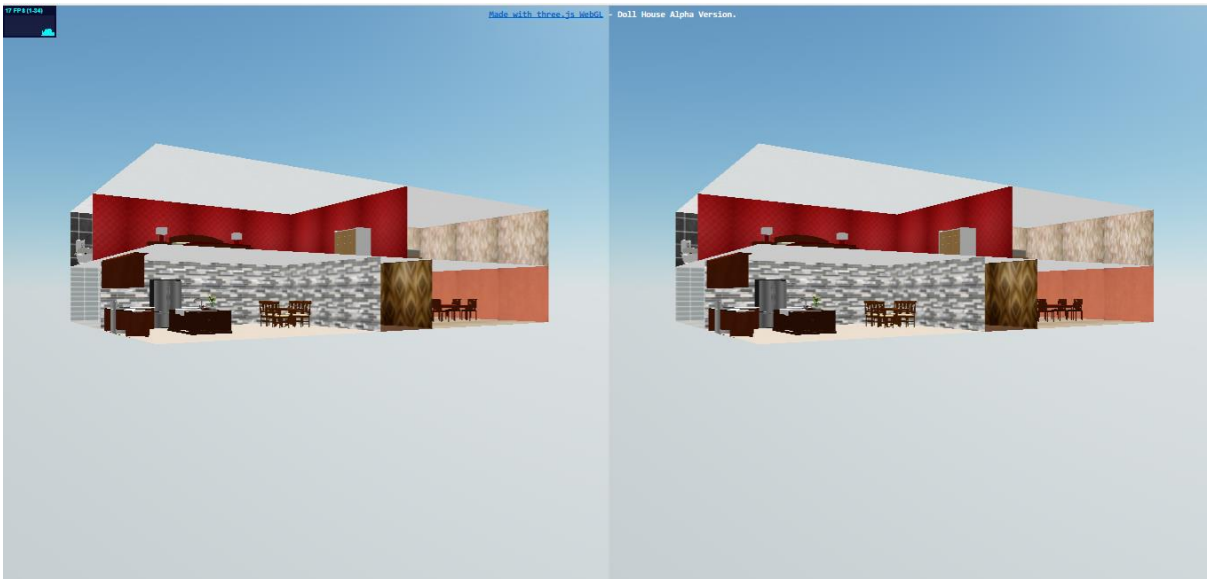


Figure 37-Google Cardboard view mode

4.7 Virtual Reality (Desktop VR)

To make the Virtual Reality come true, it was used the WEBVR JavaScript API.

During the developing process, the VR mode was implemented and tested with the 6 Degree of Freedom Lenovo Explorer headset, represented in the Figure 38 below.

This is the ultimate experience that the user can get on this solution. Being fully immersed in the virtual world, with all the freedom of movement and rotation, the user gets a more personal and realistic interaction with the house.

Figure 38 - Lenovo Explorer Headset

(bhphotovideo.com, 2018)

5. Conclusions

5.1 Objectives achieved

The work has been successfully developed apart of some limitations in the development of virtual reality in web environments.

Apart from the Virtual Reality limitation described in the sub-chapter 5.3.1 later in this report, the requirements proposed to the solution were met.

Achieving the freedom to explorer the house in a more interactive way, and having the support for desktop and mobile platforms, the solution is set to a good path to be part of the of home virtual visualization market in the future.

5.2 Other work carried out

In this section, is described side work carried out along the internship that did not had a direct impact on the project.

5.2.1 Triangulating Point Set Surfaces with Bounded Error

This study was driven with the propose of converting point cloud data set to triangulated 3D mesh to later be optimized and displayed.

“Point sets are now a popular modeling primitive in computer graphics. The initial need to model surfaces out of point sets came mostly from the emergence of affordable and accurate scanning devices able to generate dense point sets, which are initial representations of physical models. Still, creating manifold surfaces out of point sets has shown to be useful in a variety of other applications, since point sets can be used a robust technique for representing surfaces. Among the many advantages of point sets are their generality: every shape can be represented by a set of points on its boundary, where the degree of accuracy typically depends only on the number of points.” (Carlos E. Scheidegger, 2005)

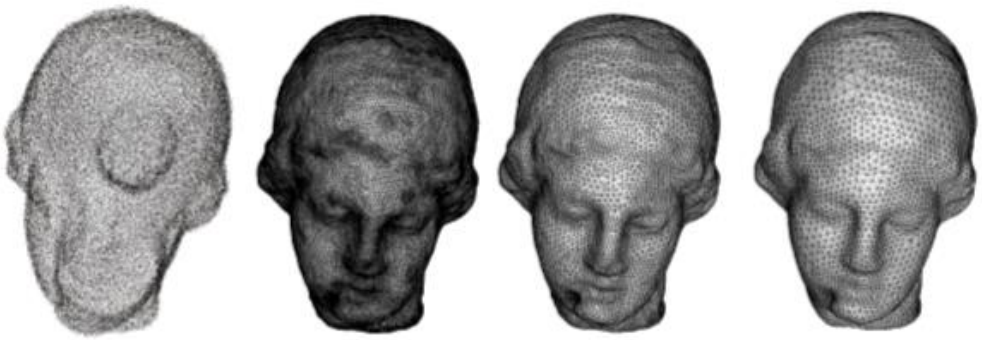


Figure 39 - Triangulations of the Igea model, perturbed with uniform noise equal to 2% of the bounding box diagonal.

(Carlos E. Scheidegger, 2005)

In Figure 39 above, we can see how the algorithm successfully reconstructs the model, even with a significant noisy point data set.

"Our algorithm takes as input a set of points without normals and produces a high-quality triangle mesh.

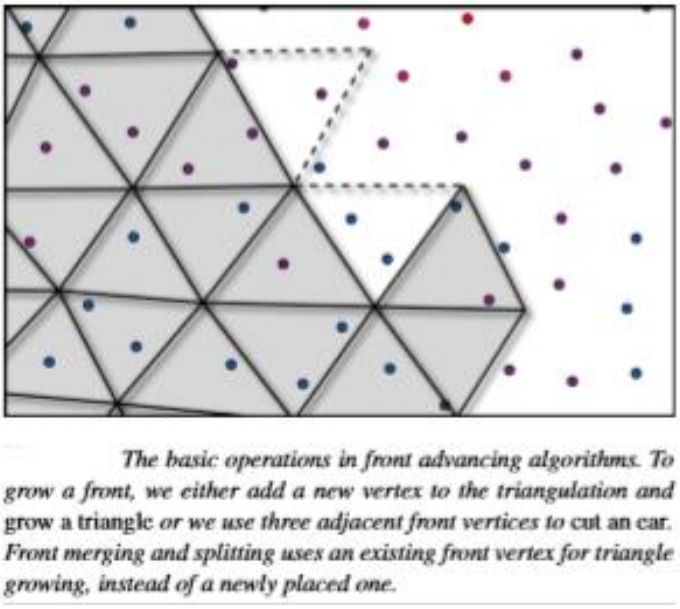
Starting with a seed triangle, its edges form the front, that is incrementally extended by two types of local operations.

A triangle growing operation connects a new triangle to one edge of the front and adds a new vertex.

A near-cut operation creates a triangle from three adjacent vertices on the front (see below in Figure 40).

Sometimes, a triangle growing step will add a new vertex that is too close to the current triangulation. This new vertex is then merged with an existing one, causing fronts to either merge or split, and changing the topology of the surface.

The algorithm terminates when there are no more fronts to be processed" (Carlos E. Scheidegger, 2005)



The basic operations in front advancing algorithms. To grow a front, we either add a new vertex to the triangulation and grow a triangle *or we use three adjacent front vertices to* cut an ear. *Front merging and splitting uses an existing front vertex for triangle growing, instead of a newly placed one.*

Figure 40 – Near-Cut Operation in Point Cloud Triangulation

(Carlos E. Scheidegger, 2005)

5.2.2 3D Mesh Optimization

During some time in the internship was studied the optimization of 3D Mesh converted from a Point Cloud generated by a 3D Laser Scanner.

In the following sub-chapters, is shown different algorithms to improve the performance and quality of a 3D Mesh Object.

5.2.2.1 Polygon Reduction (Edge Collapse)

"This technique reduces a model's complexity by repeated use of the simple edge collapse operation, shown in Figure 41. In this operation, two vertices u and v (the edge uv) are selected and one of them (u) is "moved" or "collapsed" onto the other (in this case, v)." (Melax, 1998)

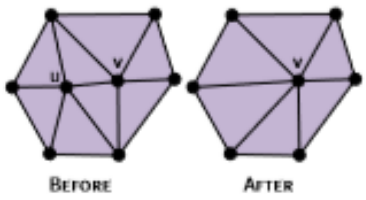


Figure 41 – Edge Collapse Before vs After

"The following steps implement this operation:

1. Remove any triangles that have both u and v as vertices (that is, remove triangles on the edge uv).
2. Update the remaining triangles that use u as a vertex to use v instead.
3. Remove vertex u.

This process is repeated until the desired polygon count is reached.

At each step, one vertex, two faces, and three edges are usually removed. "(Melax, 1998)

The Figure 42 below, gives a step by step visual example on how the edge collapse algorithm works.

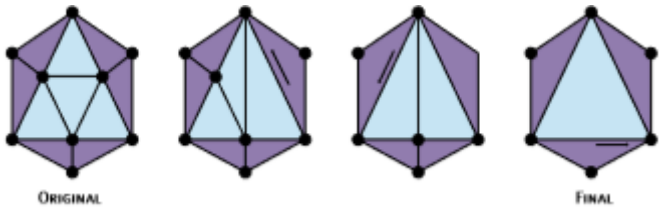


Figure 42 - Polygon reduction via a sequence of edge collapses

(Melax, 1998)

5.2.2.2 Explicit Remeshing

Like quoted by (Vitaly Surazhsky, 2003) "We present a new remeshing scheme based on the idea of improving mesh quality by a series of local modifications of the mesh geometry and connectivity. Our contribution to the family of local modification techniques is an area-based smoothing technique. Area-based smoothing allows the control of both triangle quality and vertex sampling over the mesh, as a function of some criteria, e.g. the mesh curvature."

Explicit Remeshing main goal is to improve the global quality of the 3D object, doing several local modifications. This was studied with the purpose of smoothing the mesh generated from the conversion from Point Cloud to 3D Mesh, that in many cases appears to be poorly optimized.

"Models generated by CAD software usually reflect the regular sampling of the underlying parametric domain instead of the model features. Therefore, the resulting models are usually not sampled properly, and may contain many redundant vertices" (Vitaly Surazhsky, 2003)

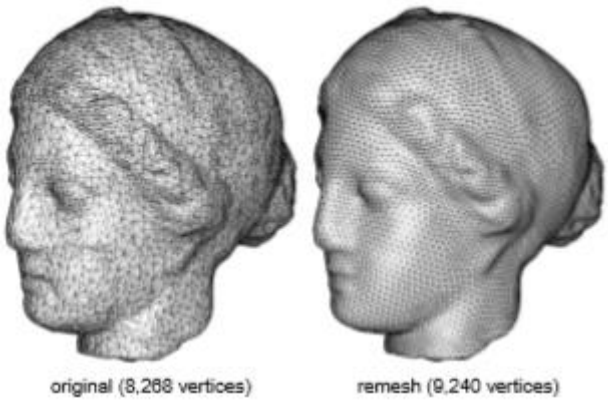


Figure 43 - A remeshing example for the Venus model

(Vitaly Surazhsky, 2003)

the Figure 43 is a good example on how the explicit remeshing technique works on a bad connected 3D mesh.

5.3 Limitations and Future Work

This chapter presents all the limitations imposed to the expected result and the aspects that need to be worked in the future, to meet and surpass these expectations.

5.3.1 Virtual Reality Controllers lack of input (Desktop)

In Virtual Reality view mode in desktop devices, there is no VR Controllers inputs whilst the user's translation must be done using the computer's keyboard. This proves to be the major limitation in this solution, for restraining the user's freedom and interactivity and being quite confusing to use the keyboard (even if it is only the W, A, S, D keys) while wearing the virtual reality glasses.

After an intense research, it was found a ThreeJS Compatible API that could handle VR Controller inputs although the API worked through JavaScript Event Handlers and was too abrasive to the work done before.

While other 3D Graphic Engines like Unity and Unreal Engine, offers more support in the Virtual Reality technologies since they were designed to the game industry (like saw in Figure 5), the solution landed since the beginning in WebGL technologies, which switching to game engines implied starting the work done all over again.

Being confident in the development and support of these new VR Technologies in the WebGL frameworks, this solution feature stand on hold until more development is done in this area.

5.3.2 Absence of doors/windows and non-rectangular shaped rooms

This limitation not only proves to be crucial in a house modelling scenario, but also when in it comes to freedom of movement, being the main reason that the mesh collision was not implemented in the solution, so that the user could move freely between the different rooms.

Doors and windows modelling, and non-rectangular room shapes could not be done while using any Box Geometry, since the Box object could not be cut to make doors and windows or have any other shape than rectangular.

It could be done using Convex Geometry which is instantiated by an array of 2D/3D coordinates in space, although the textures could not be mapped the same way as the BoxBufferGeometry.

To map textures into a convex geometry, the texture needed to be mapped according to the geometry beforehand.

Such work was part of the photogrammetry interior modeling of this project, not involved with the development of the house viewer, which this report is based.

5.4 Final appreciation

While working on such solution, I developed my capabilities with graphics technologies in a way I did not expect to dominate at the beginning of this internship. The very process of discovering and learning these technologies has proved very rewarding, while facing several barriers that have become crucial to my professional as well as personal development.

Overall, I am satisfied with the learning path and the solution achieved, looking forward to developing this solution even further, improving existing features along the way, and developing new ones.

6. Bibliografia

(s.d.). Obtido de vrs: https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html

(s.d.). Obtido de w3schools: https://www.w3schools.com/jsref/prop_nav_useragent.asp

@darkoman. (18 de March de 2011). *Code Project*. Obtido de https://www.codeproject.com/Articles/170296/3D-Software-Rendering-Engine-Part-I

(20 de January de 2016). Obtido de www.marketintelreports.com: https://www.slideshare.net/SaiTejaUppada/3d-mapping-modeling-market-growth-analysis-current-trends-shares-and-forecast

Aaron, P., & Zeller, M. (12 de 10 de 2017). *https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/webvr.* Obtido de microsoft: https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/webvr

Benchoff, B. (1 de June de 2012). *hackaday.com*. Obtido de hackaday: https://hackaday.com/2012/06/01/3d-mapping-of-huge-areas-with-a-kinect/

bhphotovideo.com. (2018). Obtido de bhphotovideo: https://www.bhphotovideo.com/c/product/1364659-REG/lenovo_g0a20002ww_explorer_mixed_reality_headset.html

Carlos E. Scheidegger, S. F. (2005). Triangulating Point Set Surfaces with Bounded Error. *Triangulating Point Set Surfaces with Bounded Error*, p. 11.

cjgammon. (21 de September de 2016). Obtido de http://blog.cjgammon.com/threejs-materials

computergraphics-animation.conferenceseries. (s.d.). Obtido de computergraphics-animation.conferenceseries: https://computergraphics-animation.conferenceseries.com/

Coppero, N. (19 de January de 2017). *Medium*. Obtido de https://medium.com/@necsoft/three-js-101-hello-world-part-1-443207b1ebe1

Crossley, R. (13 de 11 de 2010). *Web.archive.org*. Obtido de Web Archvie: https://web.archive.org/web/20100113144801/http://www.develop-online.net/news/33625/Study-Average-dev-cost-as-high-as-28m

Davis, Z. (14 de May de 2014). Definition of: SaaS. *PC Magazine Encyclopedia*.

docs.pointclouds.org. (7 de August de 2017). Obtido de docs.pointclouds.org: http://docs.pointclouds.org/1.8.1/group__octree.html

geoslam. (2017). Obtido de geoslam.com: https://geoslam.com/slam/

Google. (14 de 08 de 2018). Obtido de developers.google.com: https://developers.google.com/vr/discover/degrees-of-freedom

Greggman. (s.d.). *www.webgl2fundamentals.org*. Obtido de WebGL2Fundamentals: https://webgl2fundamentals.org/webgl/lessons/webgl-fundamentals.html

Group, K. (3 de 3 de 2011).

GRZEGORZ CIĘPKA. (3 de October de 2016). Obtido de https://www.3deling.com/whta-is-a-point-cloud/

hardmath123. (1 de June de 2015). Obtido de hardmath123: https://hardmath123.github.io/raytracing.html

laserdesign.com. (s.d.). Obtido de laserdesign: https://www.laserdesign.com/what-is-3d-scanning

Leif P. Kobbelt, M. B. (2000). An Interactive Approach to Point Cloud Triangulation. *An Interactive Approach to Point Cloud Triangulation*, p. 10.

lidar-uk.com. (2018). Obtido de lidar-uk.com: http://www.lidar-uk.com/how-lidar-works/

lwjglgamedev. (s.d.). Obtido de lwjglgamedev.gitbooks.io: https://lwjglgamedev.gitbooks.io/3d-game-development-with-lwjgl/content/chapter10/chapter10.html

Lyon, D. S. (s.d.). *davidscottlyons.com*. Obtido de davidscottlyons: http://davidscottlyons.com/threejs-intro/#slide-16

Maia, L. (2018). *3D Interior Mapping.* Porto.

mdpi.com. (2015). Obtido de mdpi.com: http://www.mdpi.com/2072-4292/7/8/9682/htm

Melax, S. (November de 1998). A Simple, Fast, and Effective Polygon Reduction Algorithm. *A Simple, Fast, and Effective Polygon Reduction Algorithm by Stan Melax*, p. 6.

okino.com. (6 de May de 2013). Obtido de okino.com: https://stackoverflow.com/questions/16406861/difference-between-positioned-and-direct-light-in-c-opengl

Radiosity - Wikipedia. (7 de April de 2018). Obtido de Wikipedia : https://en.wikipedia.org/wiki/Radiosity_(computer_graphics)

Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*.

Schütz, M. (s.d.). *Rendering Large Point Clouds in Web Browsers* . Institute of Computer Graphics Vienna University of Technology Vienna / Austria.

Sensing, A. S. (20 de 5 de 2015). *asprs.* Obtido de asprs.org: https://en.wikipedia.org/wiki/Photogrammetry#cite_note-2

shop.laserscanning-europe.com. (s.d.). Obtido de shop.laserscanning-europe.com: https://shop.laserscanning-europe.com/Rent-a-FARO-Focus3D-X-330-laser-scanner

Slick, J. (27 de August de 2018). *lifewire.com*. Obtido de life wire: https://www.lifewire.com/what-is-rendering-1954

Smith, S. (22 de December de 2017). */medium.com*. Obtido de /medium.com: https://medium.com/@stew_rtsmith/space-rocks-webvr-d4035d0ac429

statista.com. (July de 2018). Obtido de statista: https://www.statista.com/statistics/269253/computer-graphics-software-market-value-in-the-visualization-simulation-segment/

store.google.com. (s.d.). Obtido de store.google.com: https://store.google.com/gb/product/google_cardboard

techtarget.com. (Outubro de 2016). Obtido de https://whatis.techtarget.com/definition/point-cloud

Three Js. (2018). Obtido de Threejs.org.

threejs. (2018). Obtido de https://threejs.org/docs/#api/en/materials/Material

threejs. (2018). Obtido de https://threejs.org/docs/#api/en/cameras/OrthographicCamera

Threejs. (2018). Obtido de Threejs.org: https://threejs.org/docs/#api/en/core/BufferGeometry

Threejs.org. (2018). Obtido de ThreeJs: https://threejs.org/docs/#api/en/core/Geometry

Vitaly Surazhsky, C. G. (2003). Explicit Remeshing. *Explicit Remeshing*, p. 11.

Wayman, R. (26 de February de 2018). Obtido de investopedia: https://www.investopedia.com/investing/compound-annual-growth-rate-what-you-should-know/

Webvr.info. (2018). Obtido de WebVR: https://webvr.info/

we-get-around. (7 de October de 2015). Obtido de we-get-around: https://www.we-get-around.com/wegetaround-atlanta-our-blog/2015/8/photographers-how-to-convert-a-matterport-object-file-obj-to-a-point-cloud-file-pts-for-cad-3d-modeling

Weis, S. (25 de 02 de 2018). Obtido de packet39: https://packet39.com/blog/2018/02/25/3dof-6dof-roomscale-vr-360-video-and-everything-in-between/

Wheeler, A. (20 de March de 2017). *engineering.com*. Obtido de engineering.com: https://www.engineering.com/Hardware/ArticleID/14541/3D-Scanning-Understanding-the-Differences-In-LIDAR-Photogrammetry-and-Infrared-Techniques.aspx

Wikipedia. (s.d.). Obtido de Wikipedia: https://en.wikipedia.org/wiki/Stereoscopy

Wikipedia. (6 de August de 2018). Obtido de Wikipedia: https://en.wikipedia.org/wiki/Octree

worldviz. (9 de May de 2017). Obtido de worldviz: http://kb.worldviz.com/articles/2339