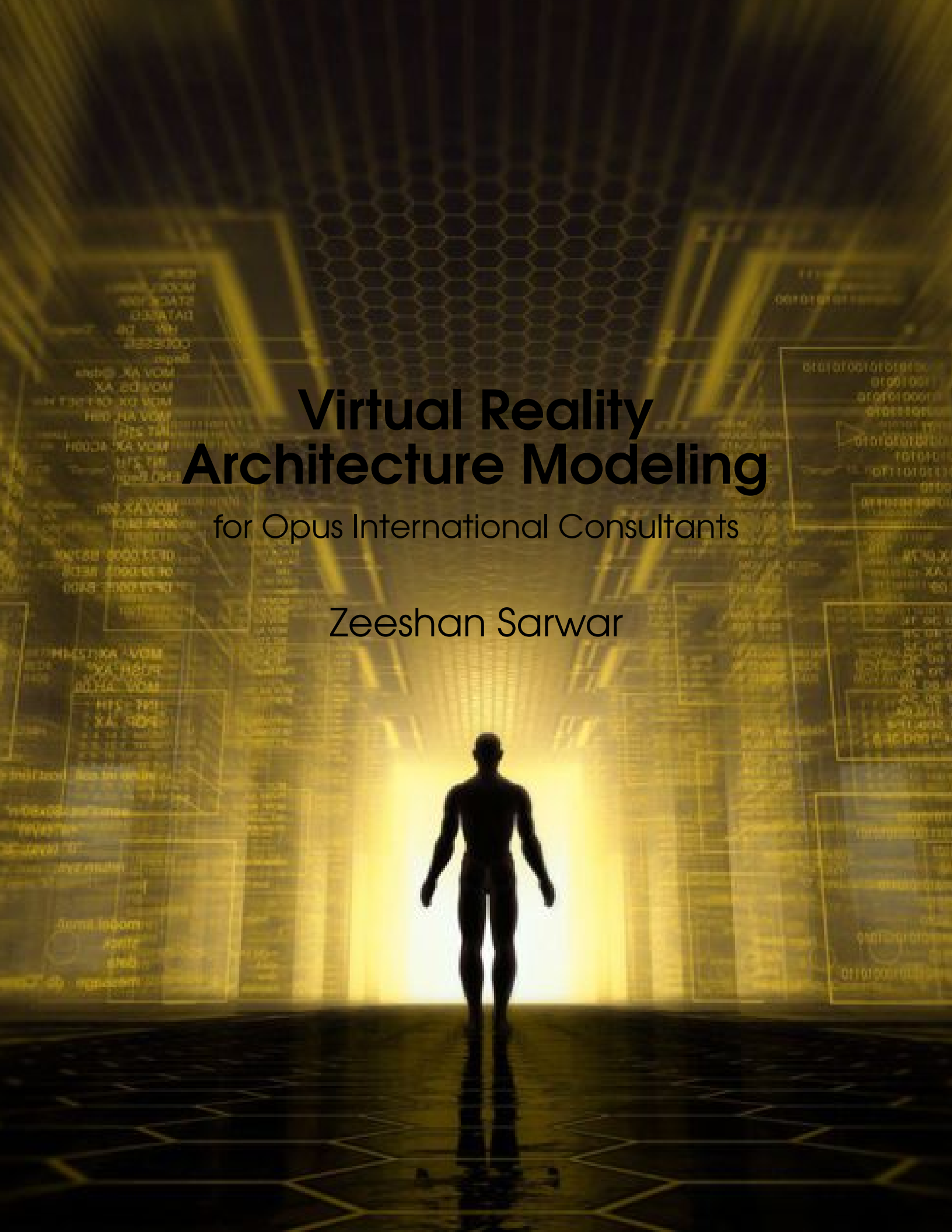


# Virtual Reality Architecture Modeling

for Opus International Consultants

Zeeshan Sarwar

## Abstract

VR is a emerging technology which has great potential, Opus wants to use utilize the VR technology for their clients. Opus will host content which will enable their clients to view 3D architecture models on a cheap VR system such as Google Cardboard. The goal of this project will be to help Opus implement such a solution in the form of WebVR.

The following report is based on a project which i am undertaking with Opus International Consultants. The project is part of my full year BTech451 degree in the University of Auckland. The project was initialized in March 2016 and is currently on going with the final report due late 2016. This is a year long project with the initial part (this report) focused primarily on the research of Virtual Reality (VR) - this includes: VR systems, VR applications, VR use cases and WebVR. In addition to the research, the secondary goal will be to prototype a WebVR solution for Opus.

I will expand upon the prototype to produce a WebVR solution which will allow the Opus VR team to convert 3D model(s) to a WebVR compatible form which they can host onto the Web. The solution can be in the form of a utilization of established software or a script/program (coded from scratch by me). Development of the WebVR solution will require research on VR itself, thus it is vital to research general VR before diving into implementation; this is why the first semester will be focused primarily on research.

There is no exact research question for the project, however the end-of-year goal is to develop an online WebVR solution which will Opus's clients to view content hosted by Opus on their in website in VR via a smartphone. The solution implemented must be user-friendly, meaning that the Opus VR team must be able to utilize the program/script by applying it on their templates and produce a file which they can host on the web with any difficulty and pre-requisite knowledge.

BTECH 451, UNIVERSITY OF AUCKLAND
This project was undertaken under the supervision of Dr Aniket Mahanti from the University of Auckland and Sam White from Opus International Consultants betwen March and November of 2016 . *First release, June 2016*

# Contents

# 1. Introduction

3D virtual reality walkthroughs allow designers and technicians the ability to gain valuable new perspectives of their design that may not be otherwise apparent. Via the use of VR, company staff and clients can explore and experience their design in full scale by being fully immersed in the virtual 3D environment.

This offers a number of opportunities including:

- Ability to visualize the completed construction of infrastructure including complex details which are much more difficult to visualize on a 2D screen
- Better understanding of the design by users, this will allow them to improve their current design by making much more informed decisions due to being able to visualize the final product (in 3D) before construction

The aim of this project is to develop a solution that can assist Opus in hosting 3D models on the Web in VR format which can be viewed using a VR headset such as Google Cardboard. This will allow staff and clients to be able to view the VR content from anywhere in the world (assuming they have a VR headset – Google cardboard, Samsung VR, Oculus rift) and/or a compatible smartphone.

## 1.1 The Company

Opus International Consultants is a leading multi-disciplinary infrastructure with local reach and global connections. Opus operates in 5 major markets - Australia, Canada, New Zealand, the United Kingdom and the United States of America.[1] Opus has over 3,000 engineers, designers, planners, researchers, advisers and work with more than 12,000 clients.[1]

Opus offer fully integrated asset development and management services at all phases of the lifecycle including concept development, planning, detailed design, procurement, construction, commissioning, operation, maintenance, rehabilitation and upgrading. In New Zealand, Opus operates from a network of 40 offices and employs over 1,700 staff which

provide services on leading infrastructure projects for both the public and private sectors.[1]

### 1.1.1 The VR Team

Opus is a very large company so it assigns teams/groups that perform niche roles to cater for clients. A specific VR team was assigned the task to produce VR solution(s) to for clients interested in modeling 3D architecture and design. I was very privileged to work with the Opus VR team since VR is an emerging technology which has great potential. The VR team consists of Sam, Andrew, James, Kodie and me (Zeeshan). Kodie has no direct involvement in VR but is rather leading the website development team that will be working alongside the VR team to host the VR solutions for their clients.

### 1.1.2 Opus's interests in VR

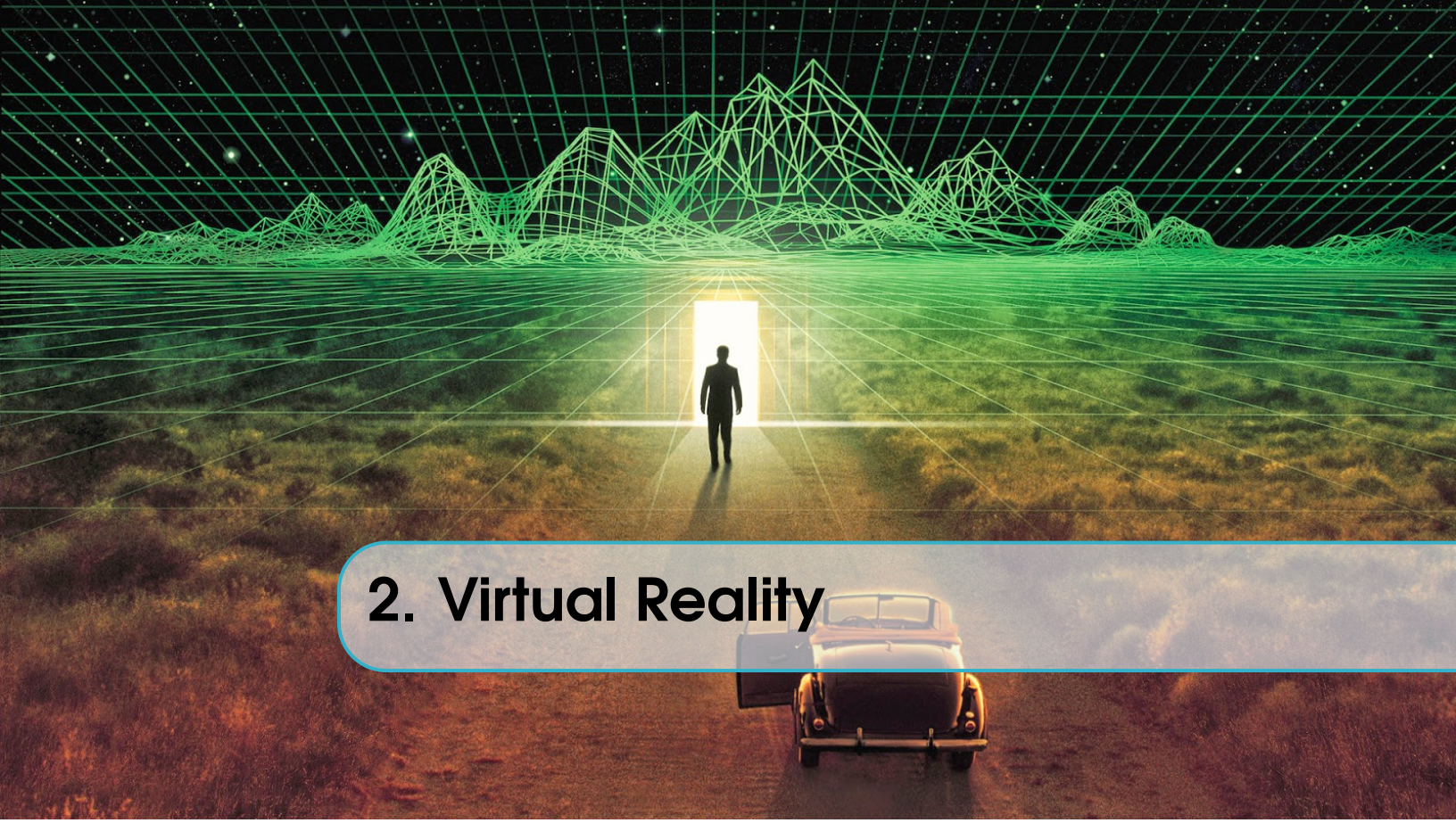
Opus International consultants are interested in VR because they believe it is important to utilise and leverage new technologies to gain and maintain competitive advantages in the industries in which the company operates. In addition to that , Opus-VR is a low cost, low risk project to explore the use of virtual reality technologies throughout all departments. The Opus VR team aims to achieve:

- Improved design quality and efficiency
- Better client experience and reputation
- Promotion of Opus

To understand Opus's interest in VR technologies we must first understand what is Virtual Reality.

The first part of this project (this report) will have general information on Virtual Reality, this is because Opus instructed me to conduct research on VR before delving into producing code/solution. Opus wants me to identify the best possible VR solution to cater for clients. To find such a '*perfect*' solution it is essential that i perform research on VR (as a general topic) because it is possible that the solution may not lie in WebVR at all (due to technological limitations) and we may have to resort to some other form of VR solution (Android/IOS app) . My research and findings on VR will be presented in the following report; in addition to that i will also prototype a WebVR solution on which i can build upon in the next semester

As part of my research, it is my duty to to first introduce what Virtual Reality is before getting into the details of Opus and VR.



# 2. Virtual Reality

Virtual reality is the term used to describe three-dimensional, computer generated environment which can be explored and interacted by a user (person). It is a form of immersive multimedia in which physical presence is simulated in virtual environments. This is generally achieved through the use of a head mounted display on which a stereoscopic image is presented, giving the illusion of depth. Motion tracking enhances the simulation, allowing the user to look or move around the virtual environment by turning their head or moving their body.

Virtual reality is generally associated with video gaming, but is currently being used for a wide range of applications including engineering and architecture. Significant developments have taken place over the past few years into virtual reality technologies; developments in a number of fields including display technology, computing power and motion tracking have significantly enhanced user experience. The technological improvements in VR has significantly reduced the price of virtual reality devices while also increasing their performance.

When using a VR headset, a user becomes part of this virtual world or is immersed within this environment and whilst there, they are able to manipulate objects or perform a series of actions. Actions such as head movements trigger a response in the headset which causes the scene to move relative to your head - using gyroscopes. Virtual Reality is primarily experienced through two of the five senses - sight and sound.

## 2.1 Brief History of Virtual Reality

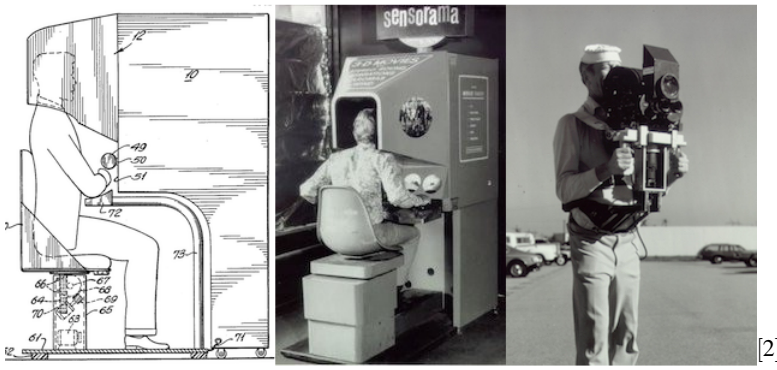
- 1938 - Stereoscopic photos and viewers: Initially VR originated from stereoscopic images. A pair of stereoscopic images are images which are angled in such a way to provide an illusion of depth - more on this later.

[2]

- 1920: - Flight simulator: The first flight simulator produced by Edwin Link was designed to train novice pilots.

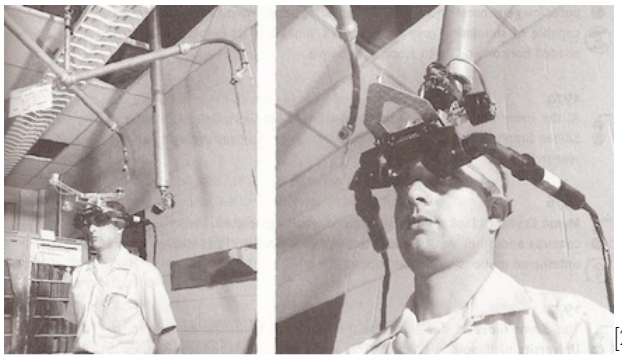

*Left: Edward Link, Right: The Link Trainer* [2]

- 1957 - Sensorama: The Sensorama produced by Morton Heilig was the first interactive theater experience with stereoscopic images, oscillating fans, audio output (speakers) and also devices which emitted smell.



[2]

- 1968 - Head Mounted Displays: In 1968 Ivan Sutherland introduced one of the first head mounted displays which attached to a computer. They enabled the user to see virtual world - in wireframe format. However had many problems such as being too heavy thus required suspension.

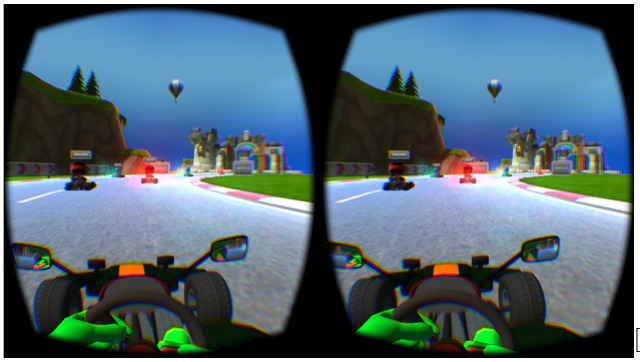

[2]

- 1970 - Interactice Map: The first interactive map was created by researchers at MIT. This enabled people to virtualy walk through the town of Aspen.
- 1990's - Introduction of VR gear: This is when the term *Virtual Reality* was adopted and picked up by the media. VR was overhyped by Jaron Lanier and Tim Zimmerman to market the devices; however VR did not live up to expectations and people started losing interest. This is mainly because the technology was not ready yet - unable to provide full immersion.[2]
- 2012 to Present - A Kickstarter project introduced the Oculus Rift headset to the masses in 2012; this headset offered to fulfill previous promises which the previous headsets could not; and thus virtual reality was resurrected.[3] The term Virtual Reality is often repackaged as '*Virtual Environments*' since VR was associated with dissatisfaction. Due to substantial improvements in hardware and computational power Virtual Reality has gained consumer confidence and seems like the next frontier.

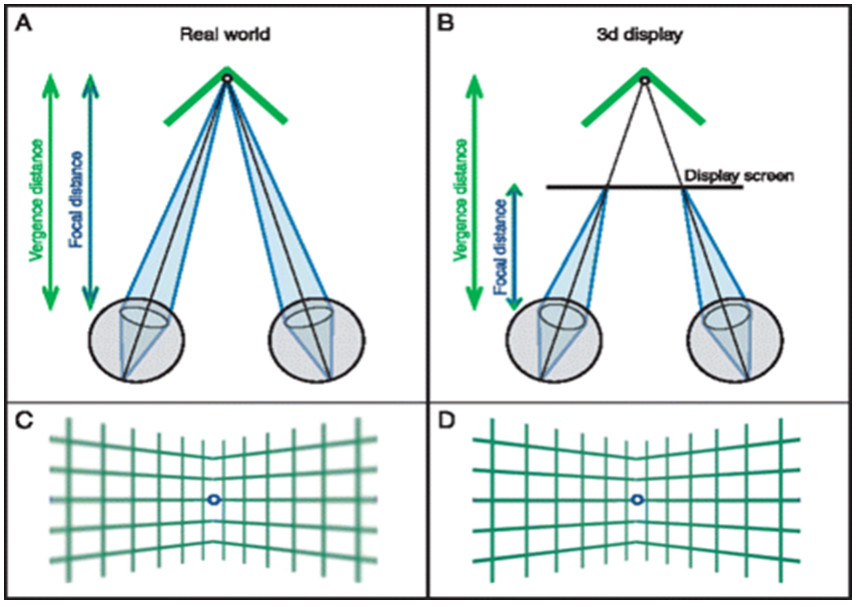
## 2.2 How Does Virtual Reality work

From the list above, we have seen how VR has changed over the years, with each iteration generally providing more immersion than the previous. But how does Virtual Reality actually work?

- A Split screen video feed is sent from the computer via HDMI into the headset. The video feed can also be sent through mobile display - for Google Cardboard and/or Gear VR. An example of the video feed is shown in the image below.


[4]

- The two slightly different views of the video feed are independently rendered . In a video feed, each frame acts as a stereoscopic image delivered to each eye.
- The lens in the headset enlarges each of the video feeds sent to each eye
- The two video feds reshape the overall frame delivered to each eye - by angling the two images. The stereo image pairs (frame-by-frame) are different to ordinary images since a stereo image (rectified) contains an element of depth. This effect mimics how we view the world which causes the immersion. An example of this is shown in the diagram below:



As shown above, in ideal conditions Box-C should appear as Box-D; the eyes should not be able to distinguish between a virtual image and the world.
- The headset/phone renders the environment based on the desired field of view. It must be noted that a greater field of view requires more computational performance since a greater number of pixels being rendered; 180 degree FOV is recommended
- Movement of the head sends signals to the gyroscopes of the headet/phone which causes the scene to rotate in the opposite direction (of the user head rotation) - this mimics how we see the world
- To note - For the virtual environment to be convincing, a minimum of 60 fps is required to avoid stuttering, otherwise users may start to feel sick

Now we know how VR actually works, so what devices offer the best VR experience? The amount of immersion in VR is dependent on the VR device one uses. Generally the more expensive options offer better immersion with rich and detailed virtual environments while the other cheaper ones are usually used to introduce communities to VR (such as Google cardboard).

## 2.3 VR devices

There are many Virtual Reality devices in the market. Each VR device may offer a unique VR immersion experience, some more than others. There are four major players in the market: Google Cardboard, Samsung Gear VR, Oculus Rift and the HTC Vive.

### Google Cardboard

Cardboard offers a *beginners virtual reality* experience. It has it's own pros and cons; with the main pro being that its **cheap**, probably the cheapest VR option there can be. At a price tag of $5, It's basically a cardboard dock with two lenses that holds a smartphone (for the display).[5] The docked smartphone must possess a gyroscope to function (synchronize with head movements). This makes the Cardboard extremely simple to use - since Cardboard is just a phone holder, thus potentially every modern cellphone is compatible with it. For VR content, the user requires access to the Google Playstore (or Appstore). But now there are websites which are developed specifically for VR; thus a user can test out VR regardless of the phone they possess - via a WebVR website. This is also what Opus wishes to utilize; developing a web solution to showcase models in VR to their clients regardless of what smartphone they possess.

[8]

However Cardboard does come with its fairshare of problems. Since each pixel is enlarged by the lenses of Cardboard, the experience is highly dependent on the smartphone display. The higher the smartphone resolution, the better the VR experience with cardboard. Also since Cardboard is just a dock, it is not specialized for any smartphone, but rather tries to cater to all smartphones. Due to this factor, there is no optimal positioning for any smartphone in the dock. You also have to hold the cardboard to your head with your hands throughout viewing VR content; regardless of this factor, cardboard dock also lets in small amounts of light which reduce VR immersion. There is only one button for controls, this severely limits interactivity.[5] The phone must be removed from the cardboard each time you wish to launch a new VR application.[5] Due to all these factors Cardboard fails to provide the full fledged feeling of VR immersion but rather only acts as a product which introduces VR to the masses.

### Samsung Gear VR

The Gear VR on the other hand is Samsung's version of the Google Cardboard; it is specifically designed for Samsung's galaxy smartphones. At a price tag of $150, it is a it is much more expensive than the Google Cardboard; it is for those who have invested in Samsung's ecosystem (devices).[5] The VR content is supplied primarily from Oculus store (which is now owned by Facebook), it combines software from Facebook's Oculus and downloaded apps created by a range of developers.[5] The screen resolution is the resolution of the inserted high end Galaxy smartphone - which is typically 1440x2560 pixels.


[9]

It has adjustable straps (unlike Google Cardbooard) so the user does not have to hold the device while also having a scrolling wheel which allows a user to manually adjust the distance between the phone display and the lens. This allows users to view VR content for longer periods of time (compared to Google Cardboard). In addition to that, Gear VR also provides basic input such as directional pad, volume control and a dedicated back button. Users can also attach compatible game-pads (via Bluetooth) to further enhance their VR experience. This not only allows one to just view VR content, but also interact with the Virtual environment. The user does not need to take off the headset since the navigation such as switching between apps is done in VR mode. According to the VR team, they will be experimenting with this very soon and most possibly use it for prototyping on Samsung galaxy devices.

#### Oculus Rift

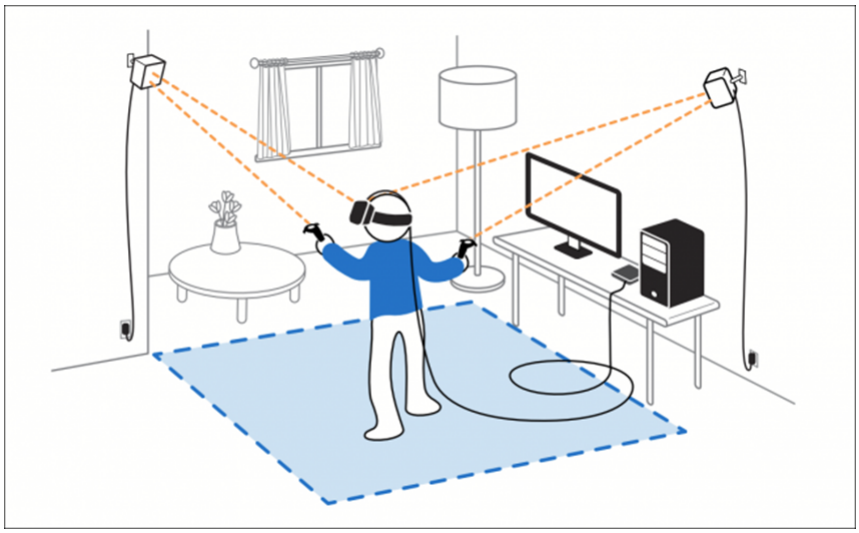
The Oculus Rift is the leading example of mainstream virtual reality technology. It features a low latency display and head tracking with support for a very wide range of applications which are run through a desktop or laptop computer. The Oculus rift is the original product which resurrected, re-introduced and to some extent re-hyped Virtual Reality. The Rift is a dedicated head mounted display (HMM) which connects to a PC via HDMI cable(s). It offers a high-end VR experience, though at a hefty price tag of $600.[6] The headset itself contains a gyroscope, accelerometer and magnetonometer. It also offers head-tracking via an external sensor; this allows a user to lean in and out of virtual in the virtual environment - thus offering a greater field of depth than the Gear VR (and Cardboard). However due to the fact that it connects to a PC, the performance is directly proportional to the PC specs (mainly GPU); a reasonably high-end PC is required to for a good VR experience.[6] This means that user has to also pay for extra to up their PC specs. Also it is not as easy to setup as Gear VR and Cardboard since the user is required to download software and connect many cables.

[10]

This product is quite pricey for most users and this doesn't even account for the PC requirements. The Oculus is for those who will pay for no compromise on the VR experience. This is the primary VR headset used by the VR team to prototype client models on the unity and RevIt Software.

### HTC Vive

The final high end VR display on this list is the HTC Vive. This is a relatively new high-end HMD which comes at a price tag of $800.[6] This product is manufactured by HTC and Valve.[7] It has the most VR content then the other displays since VR content is supplied by Valve (Steam) themselves. It offers mostly everything what the Oculus offers plus more - such as two external proprietary controllers and *Laser-powered Light House Tracking technology*.[7] Multiple camera's are setup in a defined perimeter, this allows the user to interact with virtual objects within the perimeter and thus provides maximum immersion compared to the other products. An example of this is shown in the image below:



It must be noted that to use this device, the product requires a specific perimeter for the setup (atleast 2mx2m); users may not have this much space.[7]

The Oculus rift and HTC Vive are very similar in terms of hardware, there is however a subtle price difference and that is most likely due to the extra feature (Light house tacking) offered by the Vive.

## 2.4 Opus VR device(s)

The primary focus of Opus is distribution of VR content in the form of architectural modeling for clients. This means that the cheaper the device, the more likely their client will get/use it. Distribution of 3D models/content is the main focus of Opus, this is why they plan to use Google Cardboard as the primary platform for content distribution for clients. Google Cardboard is the cheapest VR device (not exactly a device but a dock) which is easy to buy in mass quantities and distribute to clients. If clients want a better quality of VR immersion then they are free to buy their own VR device which may offer a more realistic (e.g. higher resolution) view of their model.

When i went to Opus, they initially introduced me to VR through the Google cardboard. As stated before it is the cheapest way to get a client invested into their VR. After the

introduction phase, Opus then revealed the *real* VR product - the Oculus Rift. Opus primarily uses Oculus rift for development of VR content (3D VR models); to present and distribute the 3D models (in VR), they use Google Cardboard. In addition to that, Opus also plans to use Samsung Gear VR for testing on Samsung Galaxy devices; this will be a new addition to their arsenal of VR headsets which i can use for testing.

## 2.5 VR Applications

Virtual Reality is a rapidly emerging field which holds many applications in any field. Some examples of the industries where Virtual Reality can be used extensively in are:

- **Military**: can be used and used to some extent in flight simulation, medic training and vehicle simulation.
- **Healthcare**: Virtual Reality surgery - reduces invasive procedures from doctors, allows doctors to practice surgical procedures before practically applying them on patients
- **Entertainment**: Main hype around VR is its uses in entertainment; this opens up a new world to gaming and movies (netflix)

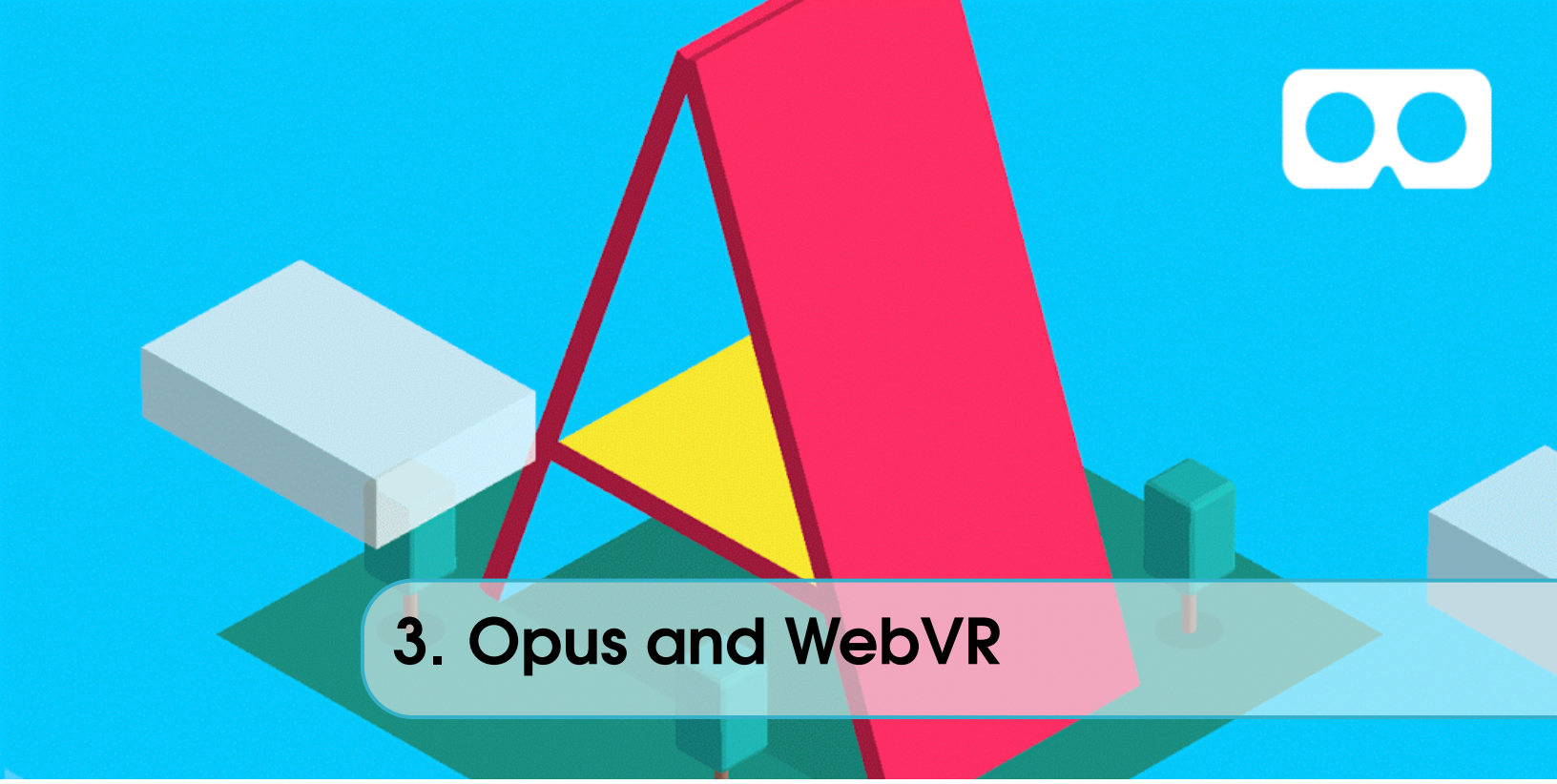

[11]

The above image is an example of full VR immersion via the use of a multi-directional treadmill and a proprietary controller (gun). The same concept can be used for military training.

- **Fashion**: In the Fashion industry VR can be used for 3D Modeling of wearable clothing
- **Engineering**: 3D modeling of general infrastructure and machinery such as bridges and vehicles etc.
- **Sport**: Can be used to provide a first person view into sports (via use of 360 cameras). This can be used extensively for analysis and identification of good technique. In addition to that, can be used as a vector for coaching new sportsman

- **Scientific Visualization**: A means of visually conveying complex 3D information. For example in biology it can be used to map the internals of the human body. In astrology can be used to produce a 3D map of space and the position of earth relative to other objects
- **Search & Rescue** - Virtual reality devices paired with remote controlled drones could be used to provide accessibility to hazardous environments while also improving safety by reducing exposure.
- **Construction** - Architecture modeling and visualization; this is the **main use case for Opus**

VR technology holds potential in every industry, these are just some of the industries where VR technology can be utilized. This is also the main reason why major companies are investing into VR technology (Google is one example).



# 3. Opus and WebVR

We have briefly defined what is VR and also seen some of the applications which VR holds for different industries. Opus on the other hand wants to go one step further and that is utilizing WebVR.

## 3.1 WebVR

WebVR is an experimental javascript API which can be used to produce content for Virtual reality devices such as the Oculus Rift and Google Cardboard. WebVR makes heavy use of WebGL library which is also a recent web technology that is used to display graphically intensive content without the use of plugins. In simpler terms, WebVR is: making use of a stereoscopic camera in a scene rendered by WebGL. So basically we are working with WebGL but using a stereoscopic (split view) camera rather than the default camera; this would allow a user to view the content (WebGL rendered scene) in first person through a VR headset. One could deduce an even more simpler definition of WebVR; it is VR on top of the WebGL framework in a web page.

## 3.2 Opus Objectives

Virtual Reality, as previously stated, has potential in any field. For Opus, they would like to utilize VR to provide clients with 3D virtual walkthroughs of virtual environment(s). By '*Virtual Environments*', we mean an architectural model proposed by the client themselves. A 3D virtual walkthrough would provide clients with an insight on how their design would look like (from first person perspective) before construction. The clients will then be better informed and thus could make improvements of their current model to obtain the *perfect solution* before proceeding to implementing it (construction). Opus commonly uses '*Computer Aided Design*' (CAD) software applications such as Revit, Sketchup, and 3DS

Max to rapidly convert the CAD files to a format which are viewable in 3D through the Oculus Rift. CAD files are the format of models provided by their clients.

There are many benefits to using VR as a primary means of sharing 3D content. Two major factors are:

#### Client experience and communication of ideas

VR allows for clear and effective communication of ideas to clients. This is especially useful for less technical clients where conventional methods of presentation may not be as effective. A virtual walkthrough of a new road alignment (for example) could showcase the roading design, bridge structures and landscape design, while giving clients a perspective of how the road will be experienced by its customers. The ability to deliver such a VR walkthroughs will also enhance the '*wow factor*' of Opus's work and has the potential to boost their reputation for delivering innovative and high-tech solutions.

#### Promotion

VR technology can also be used to promote Opus at schools, trade shows, conferences and other public events. Attendees could be given a tour of current projects that Opus is working on or perhaps experience a virtual earthquake which demonstrates innovative new structure designs. The VR team is working on such a project (in previous statement) and has managed to '*surprise*' attendees via presenting such a demonstration.

The VR team utilizes many resources for production of VR content to cater for their clients. These resources can be broken down to hardware and software requirements:

#### Hardware

- Oculus Rift and Google Cardboard
- CAD-spec computer - for building models and presenting media
- Game controller to navigate virtual environments

#### Software

- 3D modeling software - Revit, SketchUp, Unity and 3DSMax
- Model conversion software - primarily IrisVR

### 3.2.1 The Current Procedure

At the moment, clients must physically go to the Opus headquarters to test their architecture models. They give the model to the VR team which uses software to convert it to a format that is compatible with the Oculus rift (and/or Google Cardboard). Once the conversion process is complete, the client is able to view the 3D content through the VR headset.

This procedure requires the client to be physically present at Opus headquarters, thus the client is completely reliant on Opus to be able to view their own models (in VR). This is a time consuming procedure for both the client and Opus. What Opus wants to do is produce a WebVR solution; this means converting the model (provided by the client) into WebVR format and pushing it onto the web for the client to view through a VR headset (Google Cardboard). This will allows the client(s) to perform virtual walkthrough's of their models from home (rather than having to visit Opus).

### 3.2.2 The Desired Procedure

What Opus wishes to achieve is to allow clients to view VR content from the freedom of their homes. The procedure which they wish to implement for clients is as follows:

- A client sends their 3D model(s) to the Opus VR team
- The VR team converts their file(probably CAD) into VR format. This format can be an APK (for Android), IOS (for Apple) or WebVR (WebGL with stereoscopic camera)
- Once the model is converted, Opus will post the new content onto the server into the client(s) account - cannot be public for privacy reasons
- The client will login to the Opus VR website, they will be able to view the VR content under their respective account provided they have a VR headset and/or smartphone
- Note: Opus will provide Google Cardboard headset for free to their clients

Ideally Opus wishes to post WebVR content (compared to IOS/Android app) as it is compatible with potentially any smartphone. This is because for APK/IOS VR app, the user would have to download and install the program; WebVR would allow the user to visit the URL and view the content instantly upon entering the web page. Opus hasnt implemented any of these solutions yet; that is where i come in.

## 3.3 My Objectives

My role is essentially to convert a 3D model produced by the VR team to web-compatible VR format which can be hosted on a webpage. I have to produce a script (or program) which can convert these models to the desired VR format - which is javascript. The models which i must convert can be of any form; 3D infrastructure, vehicles or even simple objects. The script must be generic and form independent; i must be able to extract the main contents from the model(s) and use the objects/data to create a javascript file which can utilized by an html file to display VR content on a webpage.

In my research, if i come across any software/plugins that do this, i can use the established software/plugins to my advantage for the conversion procedure. This means that if an established software can convert models to WebVR (webgl) format, then my job is essentially done. The VR team stated why reinvent the wheel; they would use the already established software - if it reliable and doesn't have any flaws/bugs. I have tried looking for such software, but there is none available - yet; so i will most likely have to develop the script/program from scratch. The difficulty of this development depends on what other developers in this field have already tried; since WebVR is a very recent and emerging field, there are not many resources available for me to utilize. The implementation of a conversion script may be extremely difficult and the team would have to go with an easier solution such as conversion to APK or IOS format - if i am unable to produce a viable WebVR solution.

### 3.3.1 Motivation

At the start of 2016 our course supervisor Dr. Manoharan Sathiamoorthy gave us a list of options for the possible BTech projects we could do this year. The first option (out of

the 8 options) was the Virtual Reality showcase for Opus. I was interested in VR and VR modeling (game/content production) before the project options were even shown to me since VR appeared to be the next promising emerging technology. In my spare time, i would watch videos on Youtube on how VR worked, the devices, the entertainment and also how to produce basic scenes in VR (for game development). So naturally, i was inclined to choose the VR option for my project. A major factor which helped me select VR as my project option was the potential this technology had, it appeared to be '*the next big thing*'.

### The Next Big Thing

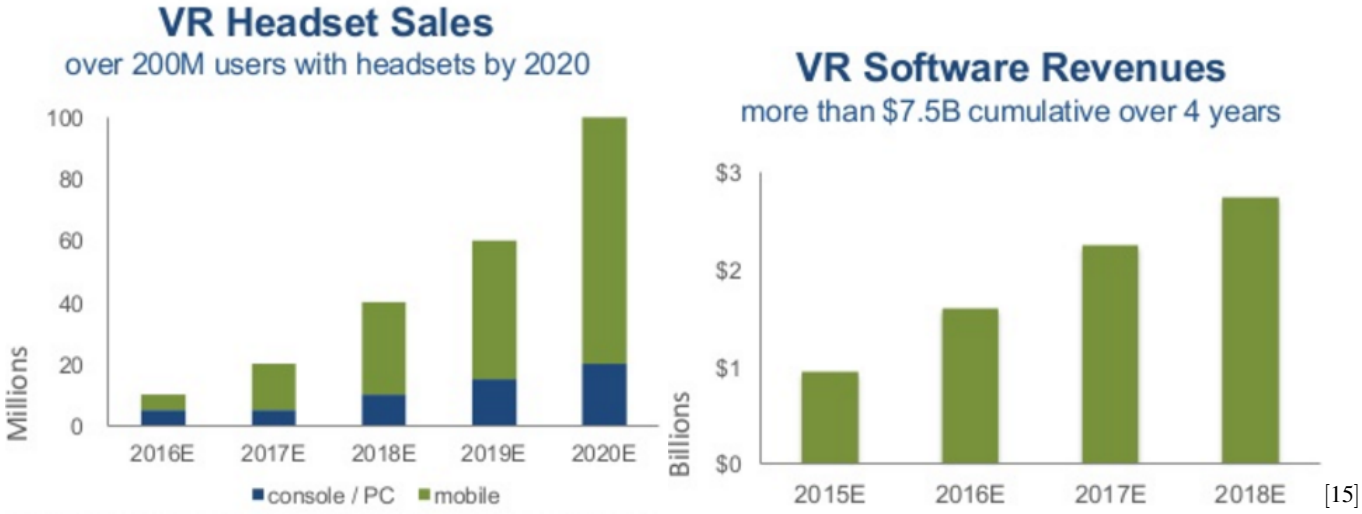
It is not only me who believes that VR (and WebVR) may be the, many big companies such as Google, Facebook, HTC and even Sony believe that VR has tremendous potential. In fact many companies have also heavily invested in VR; Mark Zuckerberg, the owner of Facebook has bought Oculus for $2.2 billion in 2014.[12] In addition to that, eight of the top ten tech companies are invested in VR. These tech companies include:

- **Apple**: They do not reveal specific details on product developments, according to the report by Tim Bradshaw at *The Financial Times*, Apple "has been building prototypes of possible headset configurations for several months".[13]
- **Samsung**: Samsung struck a strategic partnership with Oculus, this gave Samsung early access to Oculus software platform (Oculus store) for content. This partnership was also helped Samsung co-develop the Gear VR. It is poised to become one of the most owned headsets in the market due to its portability and quality of VR content delivery.[13]
- **Microsoft**: Microsoft has arguably the best research teams in Computer Vision, this is what enabled Microsoft to be the first to develop an Augmented Reality headset - the Microsoft Hololens. Augmented reality delivers VR-like content on top of reality (somewhat analogous to Google Glass). Additionally the company offers XBox One controller support for the Oculus rift.[13]
- **Google**: Google is one of the major players which has released the cheapest form of VR - the Google Cardboard. VR involvement has transformed from a "*20% project*" to a fill-on department in less than two years.[13] It has shipped more than 5 million headsets since inception; in addition to that the company is working on a VR version of the OS - Daydream, which will be released later this year.[13]
- **IBM**: IBM's cloud based *Watson platform* is one of the most technically advanced Artificial Intelligence system in the world.[13] This system is also beginning to power VR experience; this is important since this AI system could be used to bring unparalleled levels of depth and interaction of virtual characters (e.g. voice communication) in a VR environment.
- Intel: Intel is looking to use *RealSense* technology for mobile VR (like Samsung).[13] The company recently announced it will be releasing developer kit phones powered by this technology.[13]
- **HP**: This company has recently created as $999 "*VR Ready PC*".[13] This is a huge step for PC HDM's such as HTC Vive and the Oculus rift because they require high end PCs with specific requirement; with a PC like this, one would not need to buy all

the required PC components since it would all come in 1 package.

- **Foxconn**: This company is the worlds largest electronics manufacturer; it has also worked previously to help develop zSpace which is an educational "*real world VR*" system.[13]

These 8 companies only represent a fraction of the whole VR ecosystem. Others such as Sony, LG, HTC and Facebook (who are not be in the top 10) are also making huge impacts on VR with their respective headset/VR-software launches.



[15]

Analysing the stats above, we can clearly see the that there is alot of potential for VR for the future. According to the estimations, VR is a big industry; by the end of this year, there will be atleast 10 million VR users around the world. We can also see that overtime, the mobile market will be dominant. Mobile VR is powered by smartphone, these offer portability which is always better than using VR in a static location. In addition to that smartphones get more technologically advanced (updated) in terms of specs every year; thus they will be able to support the demanding requirements of VR. An example of this is the new upcoming Android N, which is an android OS specifically built with a focus on VR.[14] Technological advances such as these really support mobile VR emerge as a successful technology which will make it more dominant than console/PC VR.

From these statistics and actions made by companies, we can clearly see the potential VR holds for the future. VR is proving to be a promising emerging technology which will have many use cases in the future; this is a major factor which highly motivates me in working with VR and WebVR technology (BTech project). My development practices can be useful in content production for the VR industry.

# 4. Methodology

The main focus of the first semester was to research on Virtual Reality and WebVR with minimal prototyping. Major prototyping of a proposed solution will occur in the second semester. The main research on VR and WebVR has already been covered in this report; now we will move onto prototyping. By prototyping i imply, the approach taken to implement a possible solution which could convert models to WebVR format. The model(s) provided by the VR team can be processed in two common applications, Unity and Google Sketchup. Both these softwares enable me to display and modify the 3D models retrieved from the model file.

## 4.1 Web Technologies

Firstly to implement a WebVR solution for Opus, i have to become with familiar with certain web technologies such as basic HTML, javascript and CSS (in some cases - for code readability). These technologies are the basic building blocks required to make a basic WebVR scene which can be hosted on the web.

### 4.1.1 HTML

Hyper Text Markup Language (HTML) is the primary language used to display webpages. A set of markup tags describe the layout of the web document. Each tag emphasizes specific document content. With the introduction of HTML5, we are no longer limited to just drawing 2D rectangles on the screen. This is because of HTML5's **Canvas** API.

### 4.1.2 canvas

Initially developed by *Apple*, **canvas** is an HTML tag that can be used to draw complex graphics using javascript. [16] What canvas offers (that other tags don't offer) is that it allows

us to address each pixel individually. In addition to that canvas also enables us to edit images and video.[16] This significantly improves website performance since one does not need to download images from the network and can implement one within the document using canvas. However not all browsers (and versions) support the canvas tag, this is an important factor because we (Opus) want to cater for a maximum number of clients an thus browser compatibility is vital.

The supported browsers (and versions) are:

- Safari 2.0+
- Chrome 3.0+
- Firefox 3.0+
- Internet Explorer 9.0+
- Opera 10.0+
- IOS (Mobile safari) 1.0+
- Android 1.0+

[16] Since the WebVR solution will primarily be run on a smartphone, compatibility with IOS and Android browsers is essential.

#### Canvas - technical details

What happens when using canvas (for drawing) is that javascript is executed on the CPU, which interacts with the GPU to display the pixel on the screen. For a simple scene (2D), this is fast procedure; but for a complex scene it is not fast enough, this is due to JS being single threaded, which means you only access one pixel at a time. Major smartphone screen sizes in most cases have atleast 1280 x 720 pixels = 921,600 pixels; sequential pixel access is a major problem for such a large number of pixels. We can optimize this however through the use of WebGL - more on this later.

Implementation of the HTML itself is of least importance (to me) because Kodie (and his IT team) will be producing the frontend and backend of the website. To render graphics on a specific webpage i have to be able to append the canvas tag into the html file. Javascript will be used for appending the canvas tag and rendering the WebGL (WebVR) scene on the webpage.

### 4.1.3 CSS

Cascading Style Sheets (CSS) on the other hand is a language/script which is used for describing the presentation of a document written in HTML. It can be said that CSS is an extension to HTML since its primary purpose is to define the visual presentation of the HTML document. CSS implies a concept analogous to "*Seperation of concerns*"; we seperate the visualization/presentation of data (using CSS) from the raw data (in HTML). For WebVR implementation, the least used technology/language will be CSS; we are not concerned on the layout/presentation of the webpage, we are rather only interested in the content within the *canvas*tag of the webpage.

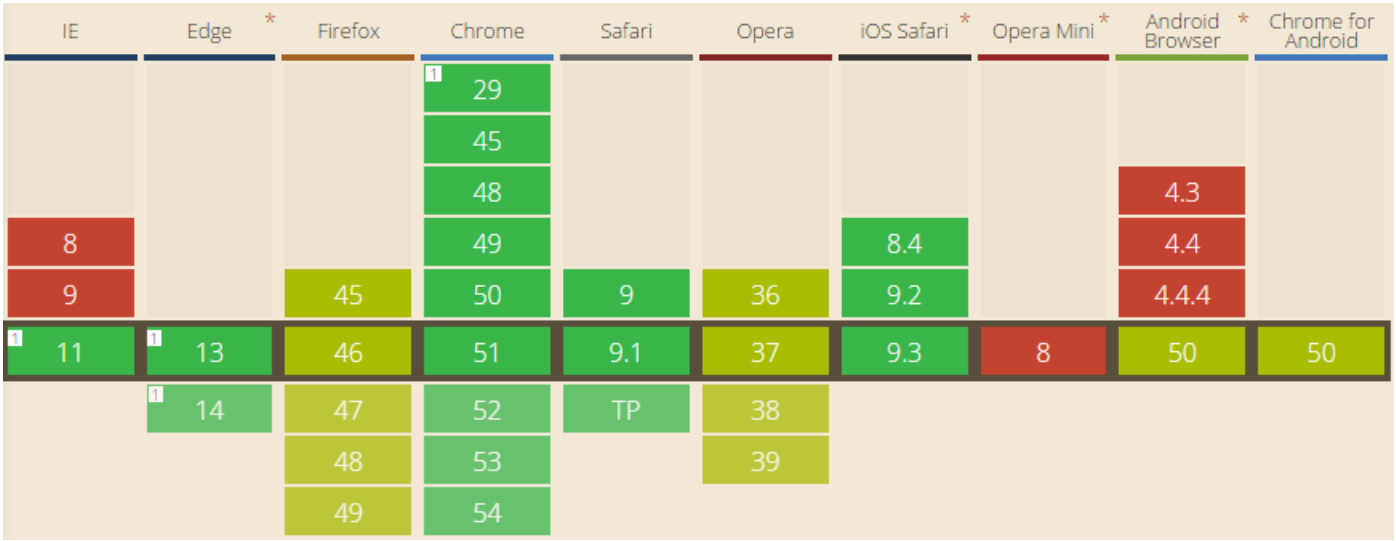
## 4.2 javascript

Javascript (js) is a high-level interpreted programming language. It's a fairly *easy* language to learn (compared to others such as C/C++) due to its dynamic and untyped nature. In addition to that it is also a very powerful multi-paradigm programming language which has support for object-oriented, imperative and functional programming styles. JS is also supported by all major browsers without the use of any plugins. HTML, CSS and javascript are the core programming languages of the web; since i will be implementing a web based solution, it is essential to know how to code in javascript. If we wish to use javascript for our webpages (which we surely will), then it must be referenced by the HTML webpage - usually done in the '*head*' tag of the html document.

To implement a WebVR solution we have to be able to display graphics onto the screen. A javascript API, WebGL allows us to do exactly this.

### 4.2.1 WebGL

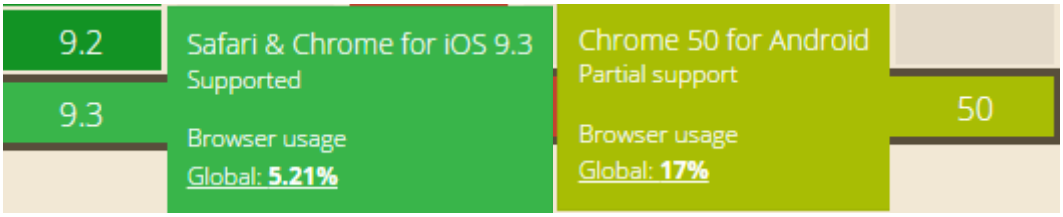
Web Graphics Library (WebGL) is a javascript API which is used for rendering 2D and/or 3D computer graphics. WebGL is derived from OpenGL ES 2.0, it utilizes the HTML **canvas** tag for displaying graphics.[17] The most prominent feature of WebGL is that it can render graphics **without the use of any plugins** - on compatible browsers.[17] Since the implementation of a WebVR solution requires WebGL, it is important that the Opus's clients have a compatible browser that can display the WebGL rendered content. These browsers (and their respective versions) are shown n the diagram below:



[18]

- Red: Not supported
- Light Green: Partial support
- Solid Green: Fully supported

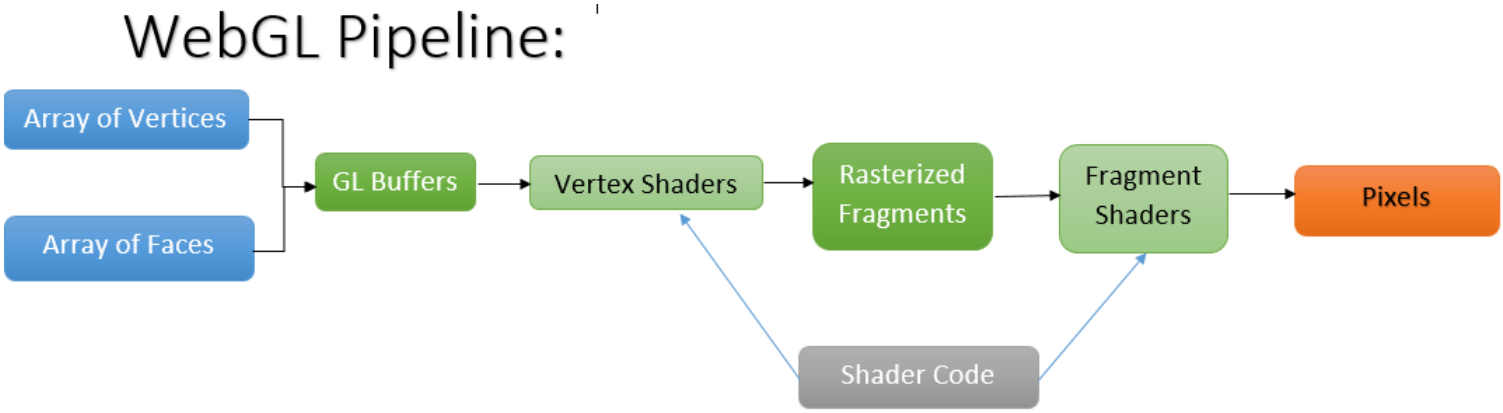
The browsers of most interest are iOS Safari, Android browser and Chrome for Android. This is because these are the mobile browsers (used in smartphones) which clients will use to view the WebVR solution (3D model in VR). IOS is fully supported so thats not a problem, but however there is only partial support for android (chrome and stock browser).




When hovering over Safari/Chrome for IOS and Chrome for android, we see that these browsers have a reasonably high amount of WebGL global usage. Android has a whooping 17% WebGL browser usage, thus meaning that a significant number of clients will be most likely be using the Android browser for viewing WebGL content (and later our proposed WebVR solution). The partial support of WebGL for android could become a constraint on how we implement the WebVR solution since the android platform has a significantly large user base.

#### Technical details

When using simple javascript for rendering graphics in the canvas tag, the JS only has sequential pixel access (as stated previously). This is a major problem but, the canvas tag can be optimized using WebGL. This is because WebGL uses the GPU in parallel, it doesn't go through the CPU sequentially like normal JS would. This allows WebGL to access billions of pixels in parallel for rendering graphics in the canvas tag and thus significantly increases graphical performance of the rendered scene.

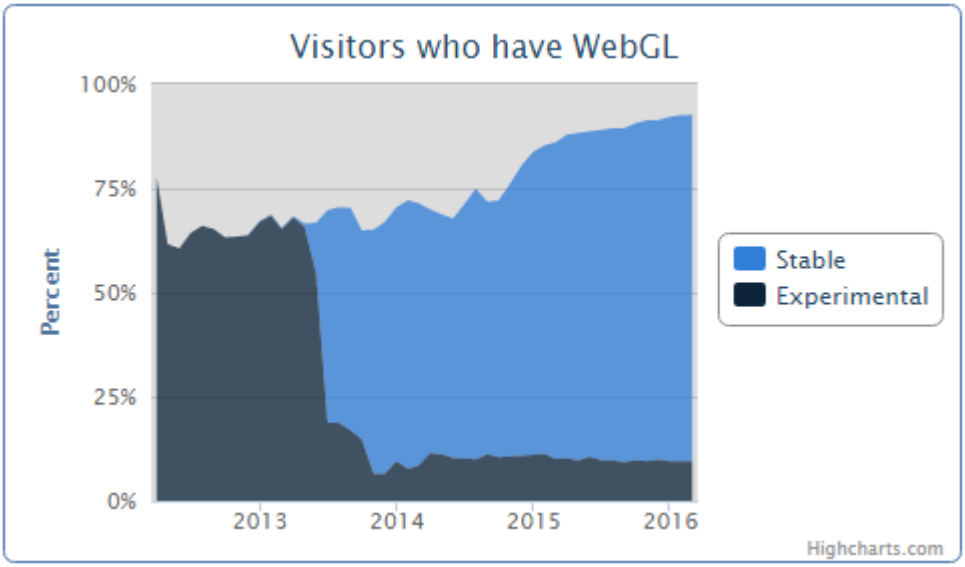



The above diagram illustrates the a basic pipeline of how WebGL efficiently utilizes the GPU. Initially two arrays (Array of Vertices and Array of Faces) are uploaded from WebGL to GL Buffers. A GL Buffer is a section of memory on the GPU (embedded in the GPU) which stores the vertices and faces. By uploading these arrays directly into the GL buffers is an example of how WebGL gets direct access to the GPU. The buffers then pass on data to the vertex shaders which return rasterized fragments. These fragments are filled with colour via Fragment shaders and output as a pixel on the screen. The Vertex Shaders and Fragment Shaders are coded in GL shader language (shader code) which is run on the GPU. We can also write in GLSL in WebGL, this is another way we can use to get WebGL direct access to the GPU. This whole process runs in **parallel** on the GPU for all pixels which results in optimal performance, this allows users to render reasonably complex scenes at high frame

rates - 60 fps+.

### Why WebGL

WebGL enhances user experience and in some cases allows viewers to get a better understanding of the concept (illustrated by the graphics). Before implementing a WebGL/WebVR solution we should analyze how prominent WebGL is as a web technology.



[19]

The following graph illustrates the percentage of users which have WebGL browser compatibility to view WebGL web pages in the time period of 2012-2016 (present). This site was last updated in February 4th 2016 (recent statistics), and shows us that a large majority of users ( 90%) have a WebGL compatible browser. These stats reinforce the fact that we should use WebGL to produce a WebVR solution as it is highly likely that clients will have a compatible WebVR browser which they can use to view their 3D VR model (WebVR solution).

## 4.3 Prototyping

To implement a viable solution i have to *test out* or in other words prototype code via making use of the web technologies. The development phase consists of having a simple idea, applying coding knowledge to implement the idea. This can be for example as basic as rendering a cube in WebGL on the screen. Once you have basic understanding of how the code works, you can build upon it to produce complex objects/structures . The initial prototyping consists of thinking, testing, and thinking again; creating your own hypothesis and conclusions on how the code works, in the process learning what works and what doesn't. This is exactly how i prototyped, testing my hypothesis and seeing what works and what doesnt. Once i have a solid understanding of the web technologies, i can then use my code to implement a solution (by the end of the year) for Opus.

### 4.3.1 Workflow

Before actually getting into prototyping i have to plan out a path to follow; without any direction it would be very difficult to produce a solution. The steps for producing a WebVR solution are:

- Retrieve a 3D model from Opus which i can use to convert into a WebVR format
- Export the model to a json object which can be read by javascript for displaying
- Render the imported model in javascript using WebGL (within the Canvas tag of a webpage)
- Once we have a successful model which we can render, then we need to convert the camera into a stereoscopic camera which will split up the perspective view into two views (one for each eye)
- Test out the solution using VR headset on a mobile device (Google Cardboard); the viewer should be able to view the scene in VR
- Once we can view the scene in VR, we need a way to interact with the environment, a way to move within the model. Initially for prototyping we will use button/keys. Remember buttons/keys are only functional if we view the scene on a PC browser, we will not have any buttons/keys when the scene is viewed on a mobile browser
- Once we have an established solution for movement via keys, then we can expand on this movement system via using just our headset. For example since we are only interested in moving forward within a model, i can implement a script which moves the camera forward when the user looks down. This will provide with the viewer with a method to traverse the scene without using any keys
- **Note**: Once we can move within the scene, we have officially converted the model to WebVR format; however we have only done so **manually** meaning that the conversion procedure may be specific to this model. I have to be able to convert **any** 3D model to WebVR automatically (without any user input - code).
- The final *ideal* solution should essentially allow the VR team to parse a 3D model into a script/program which will convert the script/program into WebVR format and output a single JS file which they can place in the directory of an html file.

From the workflow above, this means that throughout my prototyping procedure it is very important that i keep my code compatible to any input model, meaning that the javascript cannot be specifically designed for one 3D input model.

We can breakdown the workflow to 6 simple steps as a reference point to see my progress in implementing the VR solution.

#### Prototyping procedure

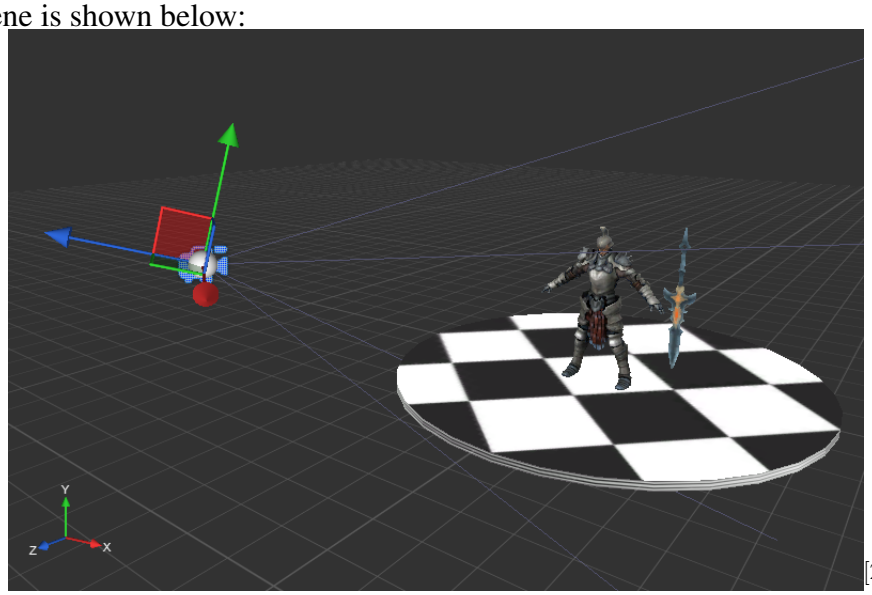
1. Setup a basic scene to render in Javascript and be able to render a simple object
2. Convert 3D model provided by Opus into json format, then load the model into javascript for rendering
3. Change the camera parameters so that i can walk through the model (rather than just view from afar)
4. Replace the normal camera with a stereo camera to make it VR compatible (Google Cardboard)

5. Allow camera movement in the scene without the use of keys
6. Generalize javascript so it is compatible with any/all 3D models provided by Opus

We must remember that these steps can only be followed in ideal conditions, by this we mean that the procedure can only be followed is each step is possible; e.g. we cannot render a scene if we cannot export the scene to json object. As long as there are no technical constraints/problems, i should be able to implement a successful VR solution by the deadline.

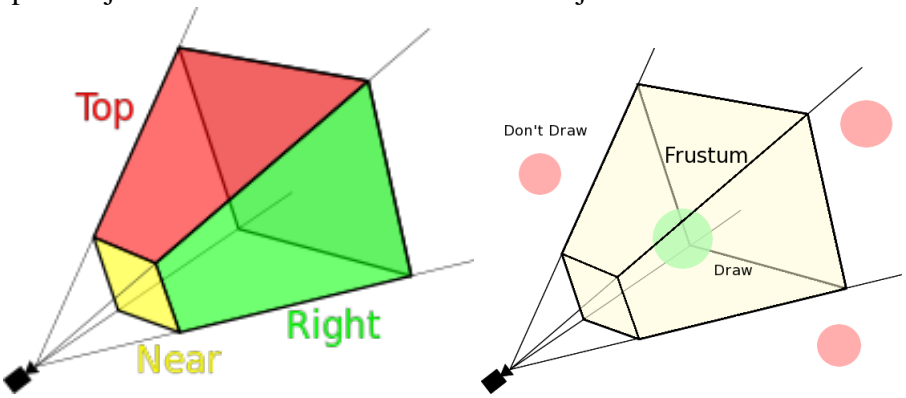
### 4.3.2 3D graphics - basics

To perform any type of rendering of 3D content, we have to first understand how 3D graphics work. To display 3D graphics on a webpage (or any platform) we require 3 essential ingredients; a **camera** a **scene** and a **mesh**. The setup of such a model in a basic scene is shown below:



[22]

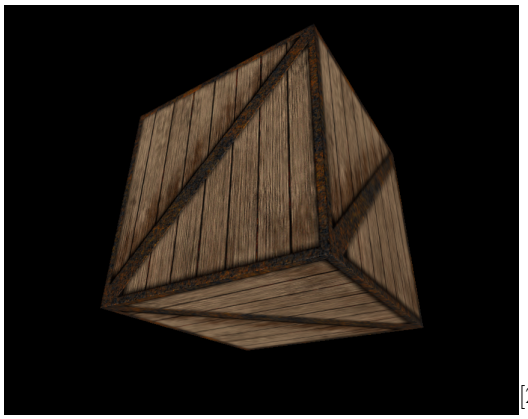
In a scene we have a mesh; a mesh consists of a bunch of triangles. Shaders are used (remember GLSL shaders) which fill in the triangles with a *material* to give us a colourful complex objects. However we cannot see this object without a camera.




A scene can only be viewed through a camera; the camera casts our rays into the scene as frustum. A frustum is basically the camera view, everything in this frustum is exactly

what the camera can see. Both the camera and the mesh have a position and a rotation in the scene. Static models have no rotation, but they do have a position. To be able to view the model, the mesh must be within the parameters of the frustum (camera view), this can be done by translating the 3D coordinates of the mesh to be within the frustum, or, a better approach would be to change the view (rotate the camera) so that the frustum contains the mesh. Once the frustum contains the mesh we need to render the scene; the renderer is used which projects all the 3D stuff within the scene onto the 2D webpage. Complex objects have alot of meshes, and thus have alot of triangles which need to be shaded with a material (via GLSL) before rendering; this is the reason why highly complex objects/models can be graphically intensive to render - this can affect the performance of the scene.

An example of such a simple 3D rendered object in a scene is shown below:


[23]

Now that we have an understanding of how 3D graphics work in general (and specifically WebGL) and also have a path to follow, we can begin prototyping. Remember that the initial step is to be able to convert a 3D model provided by Opus to a json format.

Since Opus uses two technologies for 3D modeling, Google Sketchup and Unity, it would be logical if we start experimenting with these technologies to see if we can export the model to **json** format from within these applications.

## 4.4 Google SketchUp

Google SketchUp is a free 3D modeling computer software. It is used for a wide range of applications such as modeling architecture, interior design, engineering, film and also video game design. The application allows a user to load up and view CAD (Computer Aided Design) files, these are models of structures/architecture. Initially i experimented on Google SketchUp because the first model which Opus supplied to me was in CAD format which could be viewed in SketchUp. The VR team told me to experiment with Google SketchUp because that is one of the software's they use to produce/modify models for their clients.

This software offers many tools to build 3D models; for a base to build on, users can download basic models from the SketchUp website and modify those to produce more complex models - such as the one provided to me by Opus. Model design is not my job however, my role is to convert the 3D model supplied by Opus to a WebVR format.

After tinkering around with the model, i started looking for online solutions to export the model to a json format. Exporting the model to a json format (webgl) is essential in the implementation of a WebVR solution because it would be infeasable to manually convert the models to json - takes too much time. After some searching i came across a plugin produced by **TAK2HATA** (online name).[20] This plugin allowed me to export the model as a scene to a WebGL; essentiall allowing me to skip the json conversion. The plugin converted the model to json and produced the javascript and html files which could load the model and display it on a webpage. In addition to that the exported WebGL webpage offered extensive control on how the model was viewed.

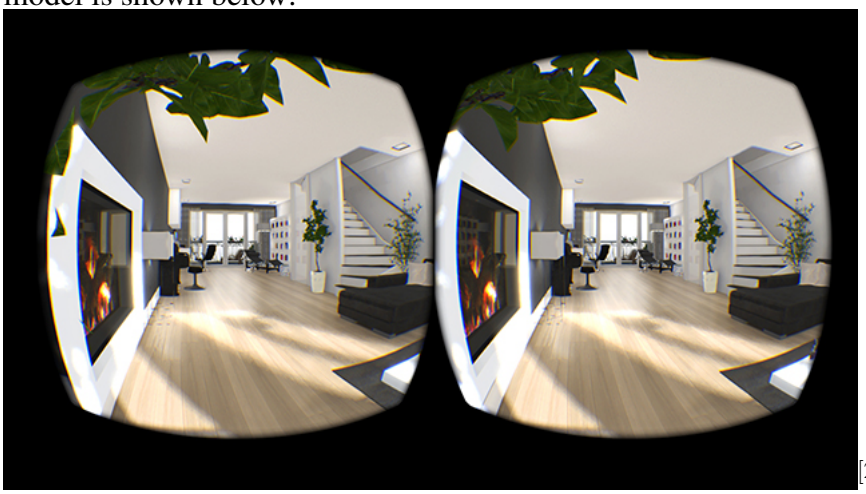
I have uploaded this model to: **http://zsar419.github.io/webgl_1/**

Usage:

- Use the arrow keys to move the camera in the scene (translation)
- The mouse changes the view of the camera (rotation)

This is a scene in WebGL, but not in a WebVR format; meaning that it is not a stereoscopic scene which allows a user to walk in the model. This export only allows a camera to view the model, it doesnt deliver immersion. For immersion we require a stereoscopic camera which can walk within the model.

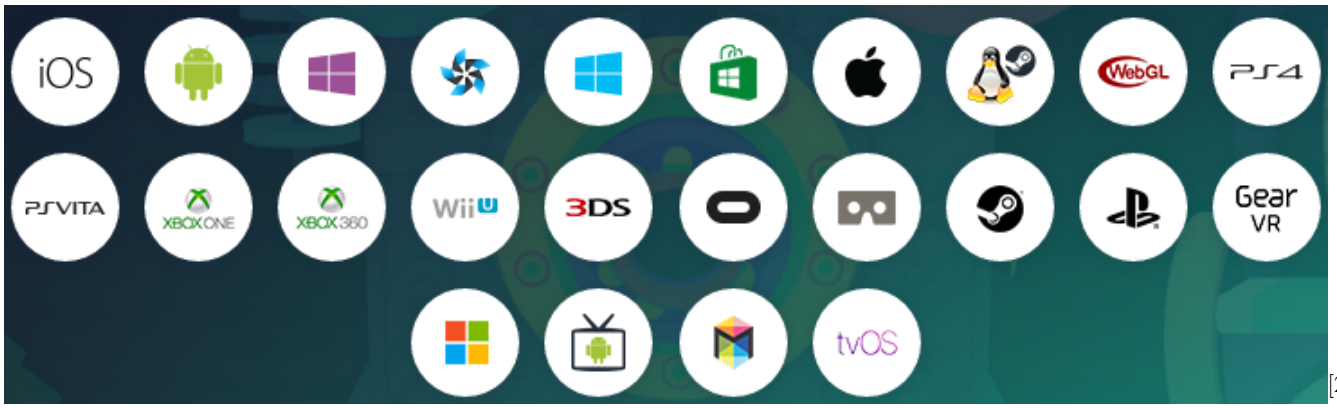
Such an example of a scene in which the camera (person) can walk within the complex 3D model is shown below:


[24]

This is what we wish to achieve with the input model provided by Opus; immersion within the infrastructure. Since the exported code is very difficult to analyze (not produced by me), thus i moved onto the other application which Opus uses.

## 4.5 Unity

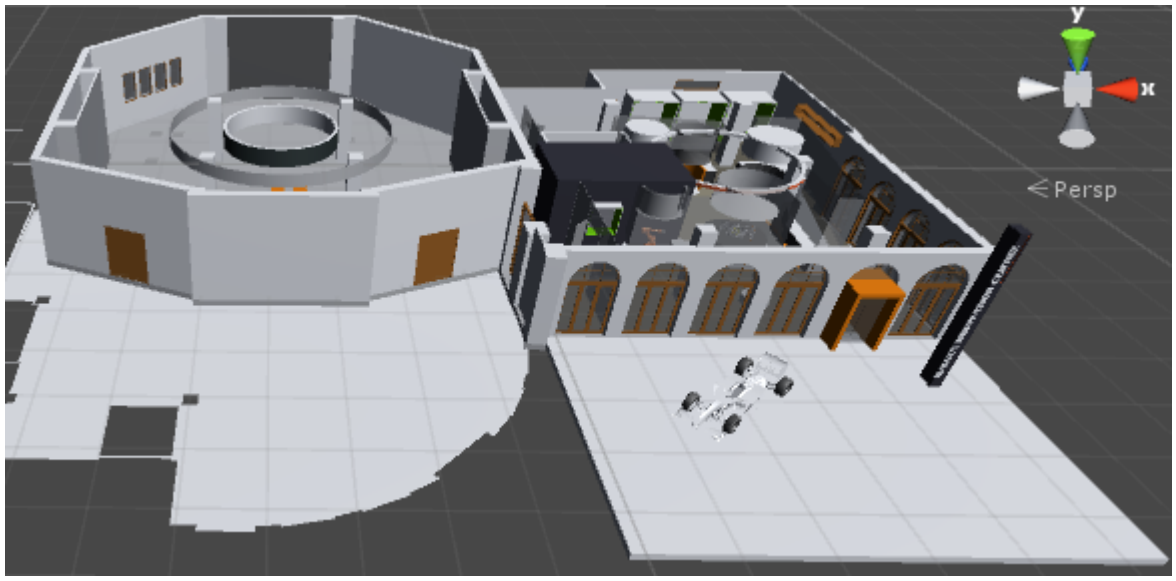
Unity is a cross-platform game engine which is primarily used to develop video games for PC, consoles, mobile devices and websites. A major upside of using unity compared to any other engine is that it follows the "*Build Once Deploy Anywhere*" model; this allows a developer to produce content only once and effectively distribute it to all platforms with a click of a button. The supported platforms are shown in the diagram below:



[21]

From the image above we can see that unty provides support for a vast majority of the platforms thus Opus can cater to potentially all clients using Unity - provided that the client downloads the 3D model. The platforms of interest are IOS, Android and WebGL. The benefit of WebGL is that it can be run on the browser without any other software, thus is the preferred solution. In the worst case scenario however, the VR team can accept an iOS/APK app (of the model) if a WebVR solution is unfeasible.

To work with unity i exported the model from Google Sketchup as .dae (digital asset exchange) format. The exporting process output two files, sketchup-demo.dae and a textures file which contained all the textures which the sketchup-demo implemented. Then i imported the sketchup-demo DAE with the textures (material) file, this gave me the following result.

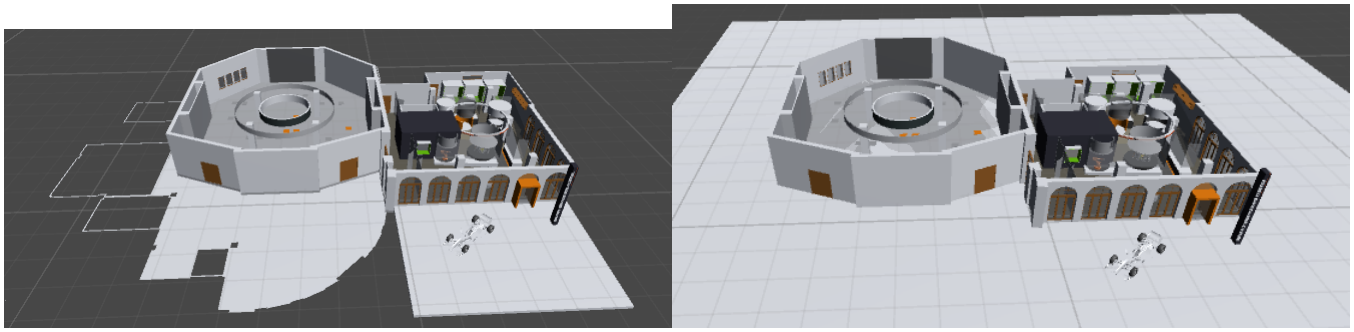



Now that the model has been succesfully imported in unity, i have to be able to walk through it in first person (like how a real person would). To be able to do walk in a scene we need to have a plane which accounts for collision - prevents camera from falling through the floor. To add a plane in unity we go to:

GameObject -> 3D object -> plane

Once the plane is added to the scene we have to scale the plane so that it represents the size of the model, we can do this by selecting the plane and looking at the "*Inspector* tab,

then scale the X and Z axis; we dont scale Y because that will change the height of the plane (we want height to be at 0).



The images above shows the the 3D model without the plane on the left and the model with the plane on the right; as you can see now we have a floor which we can use to walk around in the scene.

Now that we have a plane (scaled to model size) we can add a camera - which will act as a person walking in the scene. Now that we have a camera, we can traverse through the model. The image below illustrates the first person view of the camera within the model.


### Exporting scene

Now that we have a functional scene in unity, we need to be able to test it. You can manually test the scene within the unity engine by pressing the play button, however this is only a testing method within unity. Clients will most likely not have the unity application and thus we need to export to a platform which the Opus clients can have ready access to. Such platforms include Android, IOS and WebGL. WebGL is the most preferred platform since potentially anyone can access the webpage provided that they have a browser (which everyone has).

### Android and IOS

Since Unity is a multi-platform 3d development engine, we can also export to IOS/Android. Since exporting a project as an app is supposed to be a simple straightforward procedure, i thought i'll just export the model as an the app to my Android smartphone and see if it works. If i am able to do this, then we have a way of exporting models as apps for clients. To export the model to app you have to go to:

File -> Build Settings -> Android

Then choose the texture compression (ETC by default) and build the model as an **.apk**. After producing the apk file i transferred it to my android smartphone and tried to run the app; it failed to run. This is because for producing an app, it must be signed.

After some searching, to sign the app (for testing) you must register with oculus and sign it at **https://developer.oculus.com/osig/**. To get your device ID you can download an app from the playstore, i installed **SideLoad VR**, an app which displays your device id which you can input into the oculus website to retrieve a **.osig** key file. You must copy this file into the applications Assets/Plugins/Android/assets folder and then build the app. After performing this procedure i was successfully able to install the app, to run the app however you need to insert your phone into GearVR. This is a problem since clients will most likely not have a GearVR display, Opus is providing clients only with Google Cardboard. Nevertheless i still tested the app to see if i can view the model in VR mode. When i ran the app, to my surprise the view was automatically converted to stereoscopic, thus without even implementing a split view stereoscopic camera i was able to view the scene successfully in VR. This was a major advantage since there is no need to manually implement a stereoscopic camera as my phone detected the GearVR display and automatically changed the view to split-screen VR format.

I tested this procedure on an android smartphone, an IOS app development procedure in unity would be very similar to this. Going through this procedure i realized it is not a user friendly procedure, for example if Opus do plan to convert their models to apps through Unity, they require the clients device ID for for the app, in addition to that the clients will have to download SideloadVR (or similar) just to get their device ID. Even if the client does cooperate to give their device ID, they will need to have a GearVR headset - thus this is not a viable VR solution for clients.

Using an app to display the the model in VR is possible but we have seen that it is an undesirable solution (using the approach above) since it is too much work on the clients part and they also require GearVR device (for a samsung smartphone). We must look at the other desired approach, which is WebGL. Since we do get a working VR solution (although not using the Web), we can use this partial solution as a last resort if we are unable to implement a WebVR solution by the end of the year.

### WebGL

Unity can also export to WebGL. To do this we must first download the WebGL unity package - which can be done using the unity installer. Now that we have installed the webGL package, to export the scene we go to:

File -> Build Settings -> WebGL

Since this is a development build and we are just testing, optimization (build option) is not of great concern and thus we will just select the option which builds the scene as WebGL the fastest (unoptimized).

**Unity conversion - technical details:** The way that unity 3D works is that it converts the model/scene into C# code (changes are reflected in C#), it does so by compiling the code into IL code (intermediate language).[25] Then it uses the IL2CPP converter to create a C++ version of the code which is translated to javascript/WebGL.[25] This procedure also optimizes the output javascript/WebGL code and thus is an extremely convenient way to develop graphic heavy content for the Web.

But with these pros, there are also some significant cons which must be taken into account. Some of these are:

- Extremely difficult to debug, you can only debug through the console (very inefficient)
- Highly limited code accessibility and readibility, it is very hard to understand the code produced by the conversion
- In addition to that even if one tries to understand the code, the output file size is huge - in most cases 300MB+ because 300MB is just the size of the unity library
- Due to its humongous file size, difficult readability and hard to understand, the developer is very limited in code modification and in most cases cannot perform his/their own modifications

[25] The ease of development still (in most cases) outweigh the downsides of Unity. In addition to that it is an emerging technology which most people are working hard on; there is alot of investment on the Unity3D platform and thus this technology will continue to improve.

After exporting the project to WebGL file(s) we click the index.html to open and view the scene. Due to the fact that the project is produced in unity, we are met with a unity loading screen. We can enlarge the scene to fullscreen to get a better full-screen view.

I have uploaded the WebGL model at:

**http://zsar419.github.io/webgl_model/**

Since we have been able to successfully implement the scene in WebGL, the next logical step is to convert the camera to WebVR. By this we mean convert the camera into a stereo camera so that the user can use google Cardboard to look around in the scene. But first i must check if the hosted website works on a smartphone.

#### Unity WebGL compatibility problem

When i tried to access the hosted WebGL webpage from my android smartphone, this is when i encountered my first major problem. The unity based WebGL webpage failed to load on my smartphone, when searching for a remedy, i found out that unity exported WebGL webpages are **not compatible with smartphones**. There are major compatibility issues with smartphones, you can see this if you try to access the model from your smartphone (you may be lucky to even load the screen). This was a major problem and thus i realized that implementation of a WebVR solution is not as straightforward, currently unity does not support mobile WebGL and thus i cannot use unity to implement a WebVR solution.

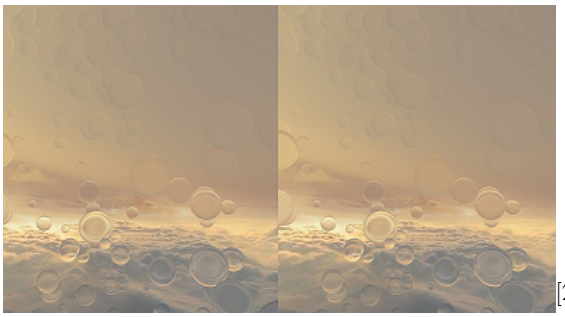
After using both these technologies (SketchUp and Unity) and failing to retrieve a viable WebVR solution, i realized i had to implement a solution from scratch - i could not use the applications to export directly to WebVR solution. Since WebVR is an emerging technology (very recent), there aren't a many resources available, thus implementation from scratch is a difficult task; nonetheless i must try. To produce a WebVR solution, this requires understanding of what is required, then planning towards it; testing out and getting results. Since there is no major support other than some examples, i realized this would be an incremental procedure in which i would have to read documentation (API's), acquire knowledge, then test out the knowledge in small incremental steps.

### 4.5.1 ThreeJS

Three.js (3JS) is a cross-browser open source and lightweight javascript library/API which is used to create and display computer graphics on a web browser. Its an API on top of WebGL which simplifies much of the WebGL code so the developer could focus more on the implementation rather than the repeated '*dirty work*'. For example shaders in WebGL are expressed via complex GLSL code, in Threejs to represent a shading, a mesh is only 1-2 line(s) of code which runs the same underlying GLSL code. This effectively means that we can render a scene with WebGL performance but using much simpler code. The source code for this graphics library/API is hosted on:

**https://github.com/mrdoob/three.js/** and **threejs.org**

After some basic researching on implementations i was able to come up with a way to use threejs for possibly implementing a WebVR solution. This is due to examples on stereoscopic threejs site which are similar to what i want. The picture below shows a WebVR scene implemented in threeJs:


[26]

The picture above shows a split screen**WebVR** scene of bubbles; using Google Cardboard a user can turn their head to view moving bubbles. You can view this scene at: **http://threejs.org/examples/#webgl_effects_stereo**

ThreeJS will remove some of the complex WebGL code for 3D graphics and allow for a much simpler way to render the same scene. To start off with 3JS i have to be able to manually render a scene. The process is the same for all 3D graphics, thus we will be following the 6 step prototyping procedure (described before) so we can start producing our own WebVR solution.

### 1 - Setting up the a basic WebGL scene

The first step is to setup the required variables in javascript to render a basic scene. To do this lets first make an html file which will be used to host the webpage. This can be a basic webpage with no content, but it must have the **canvas** tag.

To setup a basic webpage on which we can render 3D graphical content we will make a basic html file:

```
<html>
    <head>
        <script src="http://threejs.org/build/three.min.js"></script>
        <style>canvas { width: 100%; height: 100% }</style>
    </head>
    <body>
        <script>
            // Javascript will go here.
        </script>
    </body>
</html>
```

For prototyping a solution, we do not need to make our webpage visually appealing and thus we wont be using CSS. Now we have the webpage and the canvas setup we can use the canvas tag to run WebGL content. To run WebGL content we need to use javascript; in addition to that we will be using the ThreeJS graphics API instead of coding in WebGL (which will run underlying webGL). The script tag will contain all the javascript code (ThreeJS) - although this can be put in a seperate JS file (will do later).

To setup a basic 3D graphics scene we need to first create a scene, we do so by:

```
var scene = new THREE.Scene();
```

After we have setup a scene, we need to create a camera.

```
var camera = new THREE.PerspectiveCamera(75,window.innerWidth/window.inne
rHeight, 1,10000);
```

Creating a camera is not as simple, this is because ThreeJS has more than 1 camera, in here be are using a basic perspective camera. The camera requires four parameters; the vertical field of view (degrees), the aspect ratio, the clipping plane and the far clipping plane. The close and far clipping plane define the length of the camera view frustum, anything outside this range will not be rendered. After we have setup the camera we need to setup up the WebGL Renderer.

```
var renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);
```

The code above creates an instance of a renderer with a set size to render on (inner parameters of the window); then we add this renderer to the canvas tag in which rendering can successfully occur.

Now that we have a scene, camera and renderer setup we need to create an object/mesh to render, in this case we will be rendering a basic cube.

```
var geometry = new THREE.BoxGeometry(500, 500, 500, 0, 0, 0);
```

This creates a cube with height width and depth of 500. The box geometry defines the vertices of the cube - in this case a box structure. We need to colour the geometric faces with a material:

```
var material = new THREE.MeshBasicMaterial({color: 0xffffff, wireframe: t
rue});
var cube = new THREE.Mesh(geometry, material);
scene.add(cube);
```

We colour the cube with a blue coloured material (defined by 0xffffff), in addition to that we make it a wireframe model which will help us view the animations. To actually create the cube we need to use the geometry and its mesh, the next statement combines the geometry and the mesh to produce the cube. The last statement adds the cube to the scene at world coordinates 0,0,0 (by default).

Now that we have a scene setup, we must rememebr that the position of the camera is at the origin (0,0,0) while the position of the cube is also at the origin, meaning that our camera is probably within the cube, we must move it back to be able to see the cube. We do so by:

```
camera.position.z = 1000;
```

Now the camera is at a reasonable distance from the cube, this will allow the cube to be within the camera's frustum (view).

We cannot see the cube yet, this is ebcause we havent rendered the scene.

```
function render() {
    requestAnimationFrame(render);
    renderer.render(scene, camera);
}
render();
```

This is an automatic render loop since the script continuously calls the render method by default (doesnt end) after reaching end of file. The *requestAnimationFrame* allows the user to pause the WebGL when it is not in the foreground - for example when the user switches to another tab the WebGL scene pauses automatically.

To animate the cube we need to rotate/translate it:

```
cube.rotation.x += 0.01;
cube.rotation.y += 0.01;
```

In this case we add the above code to the render function so that with each frame the cube rotates along the X and Y axis.

Now we also have an animation in our scene; the basic setup of a WebGL scene is complete. You can view this basic scene at:

**http://zsar419.github.io/webgl_cube/**

It is very important that we understand what is going on in the scene (hence my explanation) because we will be working on this to produce the complex models.

Now that we are able to render a basic scene we can move onto step two, which is converting the 3D model (provided by Opus) into json format so that our javascript can load the model into the scene and render it.

### 2 - Converting 3D model to json and loading into javascript

The second step is to convert the 3D model provided by Opus (seen in SketchUp, Unity and hosted on github) into json format so that we can render it in our scene. The model provided by Opus was in .dae format, the first logical step is to look for a .dae to json converter.

After failing to find such a converter which can convert the model, i had an idea; why not just directly read in the .dae file into our javascript. After some searching, to my surprise, the ThreeJS library had such a dae loader. Instead of converting to json, this loader could directly load up the dae file into the JS scene. This is a very convenient solution since i would not have to manualy convert scenes to json, i can essentualy use the JS code with potentially any 3D .dae model provided by Opus.

To my excitement i tried out the loader, first i added the ThreeJS loader script into the head of the html file:

```
<script src="http://threejs.org/examples/js/loaders/ColladaLoader.js">
</script>
```

The script above (online) would allow me to use the dae loader code which would allow me to load up the *sketchup-demo* model (exported from SketchUp) into my javascript file. After adding the loader source, i added the following code in the script to load the .dae model:

```
loader.options.convertUpAxis = true;
loader.load('sketchup-demo.dae', function ( collada ) {
    var dae = collada.scene;
    setMaterial(dae, new THREE.MeshBasicMaterial({color: 0xff0000}));
    dae.position.x = -500;
    dae.scale.set(0.5,0.5,0.5);
    dae.updateMatrix();
    scene.add(dae);
  },
// Function called when download progresses
function ( xhr ) {console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );}
);
```

Now this code looks abit complcated, so lets break it down. Sicne the loader loads the file in the incorrect orientation ,we must fix this, the first line would correct the orientation of a model which is to be loaded in. The next line loads up the **sketchup-demo.dae** model as an anonymous function. We assign the model with a name, assign it with a mesh (red colour), give it a position in the scene, then we scale up the model so we can view it (because it is too small). To reflect the chanegs we have to update the matrix of the loaded model,

then add it to the scene so it can rendered and viewed by the camera/user.The last line is just an indicator which tells us (in the console) the the model has been loaded. Now i am finally able to render the 3D model in WebGL; but without the original materials of the model. I assigned it the colour red, but however we want realistic models, not red models. This is a major problem on which i am currently working on, currently im not able to load the original textures of the 3D model. I hope to come up with a solution soon and be able to fully render the original model in the scene. I am currently only upto this stage for prototyping, the progression on this prototype will be detailed in the end of year report.

### 4.5.2 Other frameworks

There are other high-level API's such as ThreeJS which also allow us to render 3D complex graphics on a web browser. These also simplify much of the underlying WebGL code and make it easier for us to code and render 3D graphics. These API's can potentially make it easier for me to produce a WebVR solution compared to using the ThreeJS API and thus it is essential that i also look into the following libraries/API's.

**NOTE: The following two graphics API's will be covered and compared to ThreeJS in the final report.**

#### Babylon.js

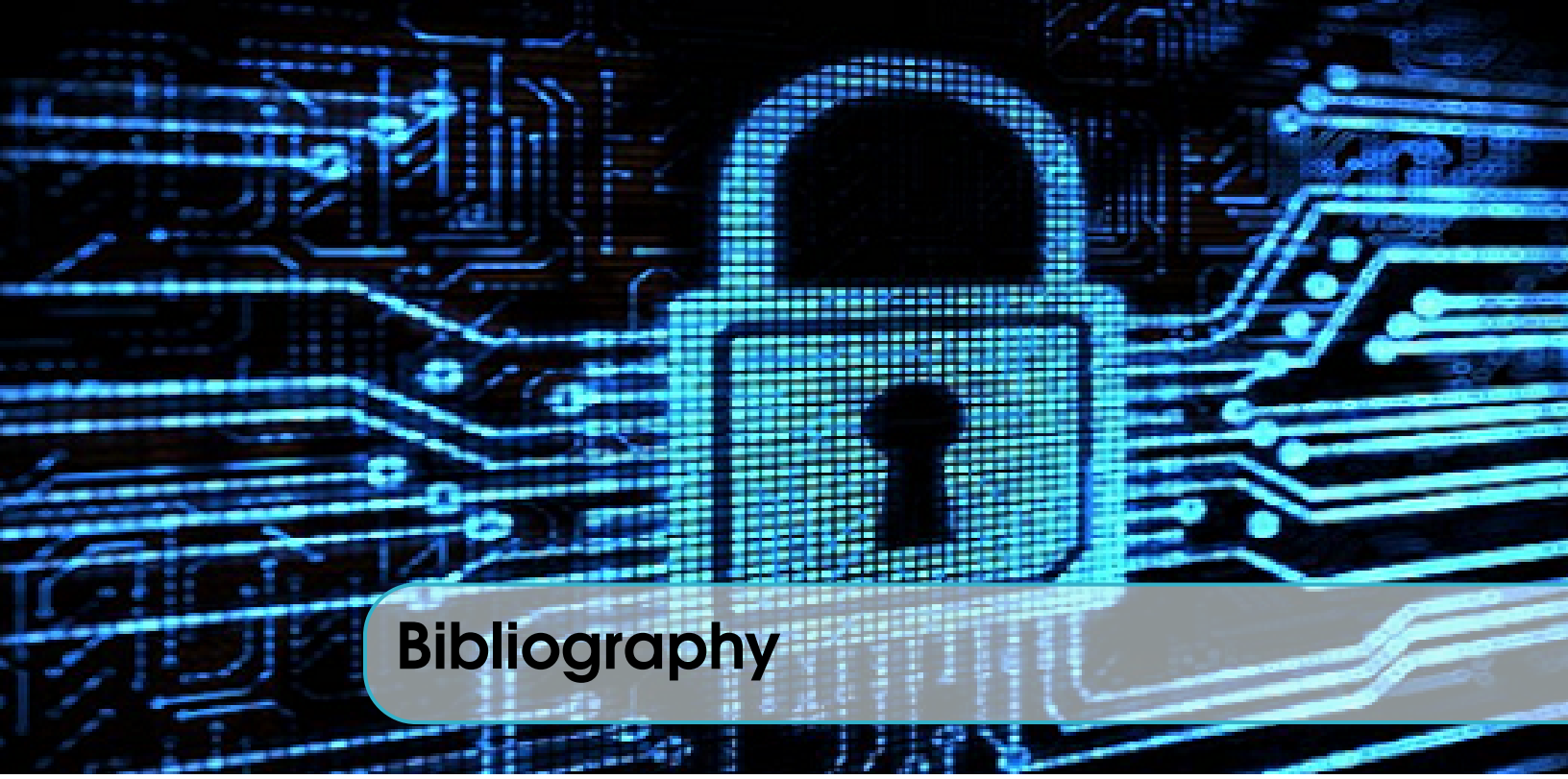
#### Aframe

# 5. Conclusion

This report covers a small set of potential benefits presented by virtual reality technologies. Many other opportunities exist, but these will only become apparent through experimentation and familiarization with the technology. Opus-VR allows this experimentation to occur with very little cost or risk and has the potential to significantly change Opus for the better. From the research provided, technological advances in VR has made it a promising technology which can be used for a vast majority of applications. In our case - for 3D scene modeling on a website for Opus's clients. For general purpose VR research, have covered the history, the current devices and applications of VR. I have clearly explained how the technology works and now currently working on implementing a WebVR solution for Opus. Out of the 6 stages on the implementation of a prototype (which will become my VR solution), i am currently on stage 2. Since the first semester of my project is complete, now the focus shift towards on prototyping of the WebVR solution for Opus. Which will be provided in the final report.

# 6. Future Work

Recently Google has also developed their own VR platform which they will be releasing in November 2016.[27] Officially announced at Google I/O 2016, they call it **Google Daydream**.[27] The platform will includes both software and hardware specifications, this means that with this API, they are planning to release their own VR headset.[27] For the software, they will be adding **Android VR** mode in the latest upcoming version of android - Android N.[28] What this mode does is that it gives VR apps exclusive access to processor cores while they are running in the foreground thus reducing latency and delivering significant performance. Sadly, due to its late rumored release date, i may not be able to experiment with any of its API since my project period with Opus will be complete by then. But this is something which Opus can look into as a platform to develop VR solutions for clients.



# Bibliography


[1] Home » Opus NZ. (n.d.). Retrieved June 01, 2016, from http://www.opus.co.nz/

[2] History Of Virtual Reality - Virtual Reality. (2015, December 25). Retrieved June 01, 2016, from http://www.vrs.org.uk/virtual-reality/history.html

[3] Thank you from Oculus · Oculus Rift: Step Into the Game. (n.d.). Retrieved June 01, 2016, from https://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game/posts/1458224

[4] Retrieved June 01, 2016, from http://vrfocus.com/wp-content/uploads/2014/10/VRKarts_1.png

[5] Mobile VR pros and cons - UploadVR. (2015, November 09). Retrieved June 01, 2016, from http://uploadvr.com/pros-cons-mobile-vr

[6] Spec Comparison: The Rift is less expensive than the Vive, but is it a better value? (2016, April 05). Retrieved June 01, 2016, from http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/#:w8h_c272yfhtma

[7] HTC Vive Review: A Mesmerising VR Experience, if You Have the Space - Road to VR. (2016, April 05). Retrieved June 01, 2016, from http://www.roadtovr.com/htc-vive-review-room-scale-vr-mesmerising-vr-especially-if-you-have-the-space-steamvr/

[8] Retrieved June 01, 2016, from http://blogs-images.forbes.com/erikkain/files/2014/06/Cardboard-2.jpg

[9] Retrieved June 01, 2016, from http://www.samsung.com/us/explore/gear-vr/assets/images/desktop/GearVR_Hero_Gold.png

[10] Retrieved June 01, 2016, from https://cdn2.vox-cdn.com/thumbor/eIQgeQJ0845uTw0cSiN-Hf0qV5s=/0x0:1920x1080/1280x720/cdn0.vox-cdn.com/uploads/chorus_image/image/48508449/oculus-rift-image_1920.0.0.jpg

[11] 5 Virtual Reality Accessories That Could Change Gaming Forever - Games Under Pressure. (2014, January 13). Retrieved June 01, 2016, from http://gamesunderpressure.com/features201417best-vr-accessories/

[12] Will Virtual Reality Be Next Big Thing Across Technology? (n.d.). Retrieved June 01, 2016, from http://finance.yahoo.com/news/virtual-reality-next-big-thing-224200034.html

[13] 8 of the Top 10 Tech Companies are Invested in VR and AR. (2016, March 08). Retrieved June 01, 2016, from http://uploadvr.com/8-of-the-top-10-tech-companies-invested-in-vr-ar/

[14] Daydream is Google's Android-powered VR platform. (2016, May 18). Retrieved June 01, 2016, from http://www.theverge.com/2016/5/18/11683536/google-daydream-virtual-reality-announced-android-n-io-2016

[15] The Emerging Virtual Reality Landscape: A Primer. (n.d.). Retrieved June 01, 2016, from http://www.slideshare.net/BDMIFund/the-emerging-virtual-reality-landscape-a-primer?next_slideshow=1

[16] Roberts, A. (2014, February 20). Introduction to the HTML5 Canvas Element. Retrieved June 01, 2016, from https://www.sitepoint.com/web-foundations/introduction-html5-canvas-element/

[17] WebGL Specification. (n.d.). Retrieved June 01, 2016, from https://www.khronos.org/registry/webgl/specs/1.0/#1

[18] Can I use... Support tables for HTML5, CSS3, etc. (n.d.). Retrieved June 01, 2016, from http://caniuse.com/#feat=webgl

[19] WebGL Stats. (n.d.). Retrieved June 01, 2016, from http://webglstats.com/

[20] T2H EXPORT WEBGL | SketchUp Extension Warehouse. (n.d.). Retrieved June 01, 2016, from https://extensions.sketchup.com/en/content/t2h-export-webgl

[21] Build once deploy anywhere. (n.d.). Retrieved June 01, 2016, from http://unity3d.com/unity/multiplatform/

[22] HOWTO: Setup a basic scene. (n.d.). Retrieved June 01, 2016, from http://doc.xenko.com/1.6/manual/getting-started/howto-setup-a-basic-scene.html

[23] Three.js / examples. (n.d.). Retrieved June 01, 2016, from http://threejs.org/examples/#webgl_geometry_cube

[24] ArchiVision Architecture (DK2) - NEW Kitchen demo - SDK0.4.0. (n.d.). Retrieved June 01, 2016, from https://forums.oculus.com/vip/discussion/11272/archivision-architecture-dk2-new-kitchen-demo-sdk0-4-0

[25] Unity3d is not ready to build WebGL/HTML5 web games with. (n.d.). Retrieved June 01, 2016, from http://blog.doublecoconut.com/unity3d-webgl-is-it-viable/

[26] Three.js / examples. (n.d.). Retrieved June 01, 2016, from http://threejs.org/examples/#webgl_effects_stereo

[27] Google Daydream Launch Date Confirmed | VRFocus. (n.d.). Retrieved June 01, 2016, from https://www.vrfocus.com/2016/05/google-daydream-launch-date-confirmed/

[28] Amadeo, R. (2016, May 18). Gear VRs for everyone! Google turns Android into a VR-ready OS: Daydream. Retrieved June 01, 2016, from http://arstechnica.com/gadgets/2016/05/android-vr-os-gets-a-virtual-reality-mode-and-vr-ready-smartphones/

*To be continued..., Zeeshan Sarwar*