RENDERING & NAVIGATION OF A VIRTUAL ENVIRONMENT AND PROFILING ON THE MOBILE DEVICES

By

ASHWIN KULKARNI
184101049

Under the Guidence of

Dr. Samit Bhattacharya

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF TECHNOLOGY

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
Department of Computer Science Engineering

JUNE 2020

CERTIFICATE

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

This is to certify that the work contained in this thesis titled as **Rendering and Navigation of Virtual Environment and profiling on the mobile devices** is a bonafide work of Ashwin Kulkarni (Roll No. 184101049), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.

Dr. Samit Bhattacharya, Advisor

JUNE, 2020
Guwahati

ACKNOWLEDGMENT

I would first like to thank my thesis advisor Dr. Samit Bhattacharya, Associate Professor at Indian Institute of Technology, Guwahati. Prof. Bhattacharya was always there to support me whenever I stuck into a trouble spot or had a question about my work. He consistently allowed this thesis to be my own work, but steered me in the right the direction whenever he thought I needed it. His immense knowledge and broad thinking have helped a lot. I would also like to thank Dr. T Vyankatesh Sir for his guidance at a crucial phase. Without their passionate participation and input, some part of the thesis might gone into another direction. My sincere thanks to the all members of UCCN Lab at Dept. of CSE, IIT Guwahati for their immense support, special mention for Nilotpal Biswas, Shakeel and Subrata Tikadar.

Finally, I must express my very profound gratitude to my late father, who has just passed away, and to my mother for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without blessings of them. My sincere thanks to all friends, specially my lab partner and batchmate Abhishek Chawda and every person who is directly or indirectly involved in this. Thank you.

ABSTRACT

by Ashwin Kulkarni, Master of Technology
Indian Institute of Technology Guwahati

Virtual reality has given us a new perspective on how we perceive technology; it has given us a new way to improve user experience. There is a tremendous amount of immersiveness we can achieve by developing useful VR applications. Still, in commercial applications, there is much scope for research. There is much need to preserve heritage sites, unique landmarks nowadays. VR is an effective way of doing this because, with the use of 360-degree VR, we can achieve more realism in virtual environments. This well-formed thesis gives you an idea about different commercial tours and game engines, which can help to create Virtual Environments. The need to use 360-degree panoramic images and an effective way to capture and handle those images can be learned from this document. Our developed module helps the user to create a hassle-free virtual environment without knowing the internals of Unity. Then we are giving insights about the performance of this module on different desktops and a workstation while developing the VE, which are helpful to acknowledge the limitations of a system while developing a tour using our module. Then in the second part of this thesis, we are analyzing performance parameters on mobile devices. There is an in-depth analysis of how different CPUs or GPUs reacts to the developed VR application. This overall study helps to recommend the minimum device requirement to run the application, these results show how we can transform mobile VR experience with the help of gestures in a minimal amount of resources.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter One

Introduction

1.1 About Virtual Reality

Nowadays, people want their stories, gaming experiences, and prototypes should be more handful to the audience rather than giving just a presentation or demonstration. Developers or content creators want to immerse their audience into his/her world. The term immersive ness means diving into the world, which is not there, and there comes the term Virtual reality. We, humans, analyze every information through senses and our perceptive system. So in this VR world, we cannot stand there in reality. However, our perception feels that we are there, This is a very vague explanation about VR, so in technical terms, Virtual reality is an environment in three dimensions and generated by computer graphics in which users can interact through different gestures, manipulate objects and scenes in the environment. VR can replicate the environment in a real or virtual means to have an interaction using sound or touch or movements of the user. We can do real-time simulation and interaction using many different sensorial mediums such as vision, hearing, smell, and taste. There are many different definitions of VR given by many authors and researchers, but the ground truth of all is an immersion into the newly created 3D world using computer graphics. The current hype of VR began in 2012, where Oculus Rift [24] was introduced to the public. But the actual virtual reality concept was started almost a century before. In 1935 the visual

technologies in Virtual reality had introduced in a medium of science fiction, in a short story named Pygmalion's Spectacles, by Stanley G. Weinbaum [50]. His team proposed a set of glasses by wearing then we can get immersed in the world and speak to the shadows, and they reply. After in the late 1960s, Ivan Sutherland invented a sketchpad [33], and by its inspiration, one MIT student created a headset called Sword of Damocles. Through which we can see the floating cube in the real world, this was a more towards AR than VR, but it was a great inspiration for these technologies. While working, Sutherland also worked on computer-generated graphics instead of analog images. In 1973 Sutherland and Evans developed graphics generators that can show 20 frames per second that were the motivation behind rendering different scenes. [6] Meanwhile, in 1970-1980 much work was done in this field, but most of it remained unpublished. Later in 1981 National Aeronautics and Space Agency (NASA) created LCD based HMD, which they called VIVED, which was used back then to create a virtual prototype of a space atmosphere. Later in 1985, the project was joined by Scott Fisher, who then integrated Glove in it. Until then, none of this technology had a particular name; hence Jaron Lanier, who was a college dropout, named this technology "Virtual Reality" in late 1980. In real-world VR, helped gamers and developers a lot, whether, in the arcades or the gaming zone of every mall, VR then got popularity everywhere. Then comes high potential smartphones, which have gyroscope and accelerometers in it. Then it becomes a boom for developers to use these features and create something which can sense the motions of mobile phones. And hence all hardware related issues that were acted as obstacles suddenly got disappear. After that, in 2012, Luckey, who was 19 yr old VR enthusiast, created a prototype of a VR Headset, and it got famous in a short period. His company Oculus raised more than $2 million to kickstart the production. Later in 2014, Facebook bought Oculus, and after that, it became more popular. In 2016, the HMDs which can connect to the High-end PC's and take the power supply from them to give users a fully immersive experience of VR came, out of the HTC Vive and Oculus Rift was stood out popular.

1.1.1 Three I's of Virtual Reality

As per the above discussion, we can easily conclude that VR should be Interactive and Immersive as well. Also these things depends on the development, but these are the two fundamentals of VR. There is one more feature which is also equally crucial for VR applications. VR is not just meant to take leverage of high-end computing and graphical power, but it should be used in real life to make people's life easier, whether in the Teaching or the medical industry or Military VR can be beneficial. But the quality of these applications depends on developers' imagination and how much they can simulate the real world in the VR application. Hence the third fundamental thing is Imagination in VR applications. Hence Virtual reality is made of a triplet called Interaction-Imagination-immersiveness. [37]

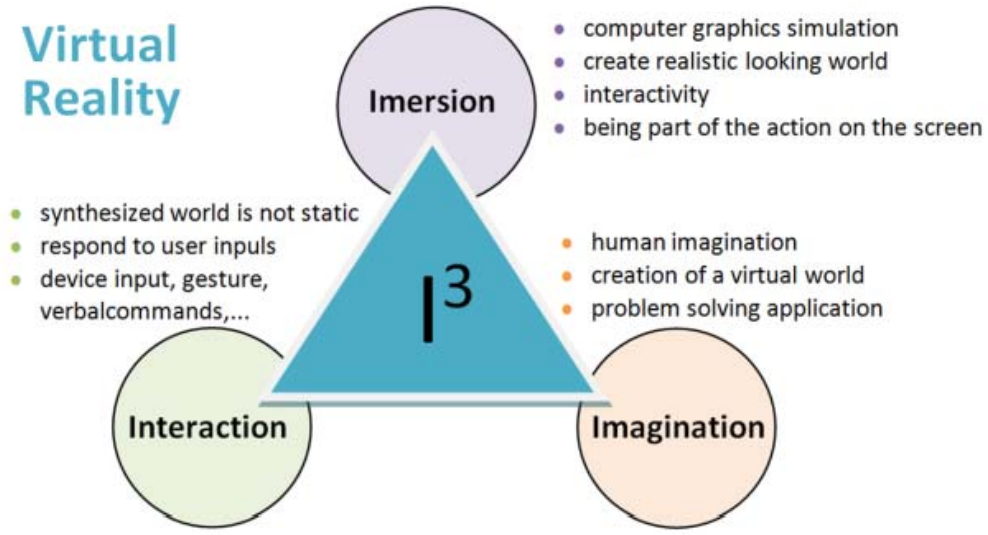

Figure 1.1 3 I's of Virtual Reality

1.1.2 Types of Virtual Reality

Based on the interaction, User experience, Imagination overall VR implementation divided into 5 broad types. [36]

1. **Non-Immersive Virtual Reality:** This VR concept refers to the VR experience through a computer where users can interact with the computer directly but not immersed in the Virtual world. The user experience of this VR type depends on the software through which we are interacting on the computer. For example, playing a video game on the computer or Xbox.

2. **Fully Immersive Virtual Reality:** In contrast to the above type, this fully immersive VR ensures that the user can get fully enter into the virtual world as if he is physically present in the virtual world. This is an expensive form of VR which requires HMDs or gloves or some sense detecting devices. The classic example of this is, a user is standing in a room and wearing a helmet for virtual shooter game with all gears in the hand where on the screen of helmet user can see the actual shooting plot any by running/walking/jumping in that room user can play the game.

3. **Semi-immersive Virtual Reality :** This is a mixture of Non-immersive and fully-immersive virtual Reality. All the activities concentrated towards the user in this type where a user can't make actual body movement except screen or mouse or some touch gestures or simple HMD movement. This type does proper use of a gyroscope. This is a cost-effective and more common type of VR.

4. **Augmented Reality :** This is not actual user based VR; instead, some virtual object is placed into the real world using some device. These virtual objects are in the 3D or 2D form, and we can create them using any applicable software. We can see some characters or object through the mobile screen where the surrounding world is the same as the real world.

5. **Collaborative VR:** In this form of VR, many people across different locations can come together to see the virtual created object or character. Virtual meeting rooms or business rooms can make most out this type of VR.

1.1.3 Applications of Virtual Reality

As explained Previously, VR is mostly used in the gaming industry, but now, after knowing the importance of using VR in many different areas, some of them are listed below:

- **Military:** VR reality has been accepted by the military in three areas - army, air-force, and navy. This can be used in Flight and vehicle simulation in difficult situations, War simulation, virtual medical training in the field, etc. [48]

- **Healthcare:** The Healthcare sector is a significant area where the VR adoption rate is high. The use of VR can be software for Human simulation, Virtual robotic surgery, virtual diagnosis, and virtual simulation of disease in front of patients. [48]

- **Fashion:** Virtual fashion stores and virtual models to check how dress look after wearing are trending things nowadays possible through the use of VR. Also, to build a 3D fashion portfolio as well, VR can be used. [47]

- **Education:** In this field, there are many applications of VR as well. The teacher can deliver the concept using virtual reality, where they can present a 3D model of an object to explain better. [46]

- **Heritage preservation:** VR can give a better demonstration of museum and historical places, old towns, and villages by navigating through them virtually using HMDs or interacting with the environment.

There are many more applications of VR are there as well here I have listed out popular one's. Now by considering available resources there is much hype about 360-degree image or video based VR tours. Since these tours give users total feeling about roaming into the VR world. Let's have a look on what is actually 360-degree VR?

1.1.4 360-degree VR

360-degree virtual reality concept became famous after releasing of the low-cost head-mounted displays or VR headsets. The main advantage of this type of VR is, users can choose his point of interest while viewing in the virtual world. It is an augmented or altered environment where the system allows the user to view in any direction. This 360-degree VR is usable in many aspects other than just gaming or site viewing. We can create 360-degree, VR based applications using 360-degree videos or images. In the later parts, we have explained our method to work with 360-degree VR [51]. For this virtual tour, instead of creating a similar-looking graphical model of the place, we can take 360-degree photos or videos, which are helpful to preserve details about the site as giving users natural experience; This is much important because having already created a graphical environment limits user interactivity and naturalness of the environment. We can create a virtual environment similar to the natural environment using computer-generated graphics, but this requires much more time, creativity, and processing power. Instead, using 360-degree cameras, this can be done with a single click. There are many cameras available in the market to capture these videos or images, and we are going to discuss them in later sections.

1.1.5 Virtual Reality tours

After gaining popularity, the usage of VR has increased in many ways like attract the users through VR commercials, spread the knowledge, or preserve the cultural heritage. Developers can create VR tours by creating 3D models of an environment using graphical-based CGI, or they can use 360-degree multiple images or videos. We are focusing more on the cultural heritage preservation using 360-degree VR throughout this thesis. VR based tours are essential in many ways, such as spreading awareness about the site with the people who are sitting 1000 miles away, explaining minor details of a site using these tours, which while roaming people might have missed. The 360-degree based VR tours offer great exposure

while engaging with the audience. Creating these 360-degree based VR tours requires 360-degree panoramas or videos, some software to connect all these captured videos to make the overall environment and a proper navigation model. We are going to focus on these aspects of this thesis. There are many software, commercial applications available to create virtual reality tours based on 360-degree photos or videos. The types of input devices to navigate in an environment is subjective according to the need, and These can be a simple mouse, keyboard, trackpad, gloves, HMDs, or easiest is head movements. The scope of expanding the application, targetted audience, hardware availability, these things matters while considering input format for VR. Also, the body movements can be captured using micro or nano-sensors attached to the different parts of the body. There can be sound-based inputs as well to navigate inside the environment, which involves usage of a microphone attached to the device and some commands to move inside an environment. Now people are using Natural Language processing to recognize complex commands given by users and then do some action accordingly inside the VE. There can be a great use of mixed reality in these VR tours because developers can augment the 3D model of some valuable object of a tour so that users can observe them or interact with them.

Figure 1.2 CGI based VE

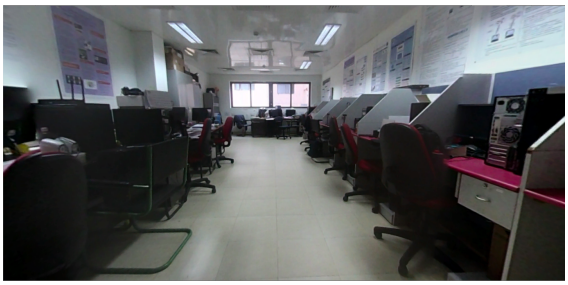

Figure 1.3 360-degree image based VE

Based on the user's input, the virtual environment system show an appropriate visual scene on the screen, and This is an essential part of these VR tours. Because showing appropriate visuals on each user's input is a primary task and lots of research going on in this field as well. If there is a need to generate some 3D models or Computer Generated

Imagery (CGI) [10] based graphics inside these tours, then these should be rendered with the lowest latency and with appropriate use of processing power. If there is a need to display 360-degree videos or panoramas, they should also be displayed with lower latency and without collapsing the naturalness of the environment. Also the quality of these images or videos matters a lot, but again there is a trade-off between quality and latency. These things need to be considered while developing VR tours. We are going to discuss all these aspects to create a virtual tour in the upcoming sections.

There are many more applications of VR are there as well here I have listed out popular one's. Now while developing a VR application, one should take care of it's deployable hardware limitations, like one cannot make a very high GPU consumable application for mobile phones because every will not have that much power packed smartphone, hence likewise there are many cases where not just creativity but performance of an application matters a lot in the next section I would like to put focus on this area.

1.2 Performance Analysis of Mobile Devices

1.2.1 Importance of Performance checking

The Virtual Reality applications has great hardware requirement, that too without spoiling user's experience. After creating a VR application according to the target user base it should run on the maximum number of devices. There can be many aspect of performance checking, whether it's in UI/UX or it's in hardware usage or it's in software section. All of these aspect should meet to run an application. There can be many stages of performance checking right from the beginning of the development stage to the deployment and even after installing to the targeted device, alpha/beta/gamma levels of testings are required. Both of the evaluations whether it is on the developer's side or it's on the user's side requires different set of parameters. Here we will discuss about mobile based performance checking

and I will explain more about it's parameters and general terminologies. More in depth mobile application testing provides developer to scale the application on a large levels and multiple platforms. The main advantage of performance testing is to increase functionality, durability and usability. The performance checking can be done using automated methods or Manual testing. In this thesis we have done manual testing upto some extent with the help of some external applications. The core motive of the mobile application testing is to check it's performance across many devices. Many testing teams thinks that testing on the actual device is more beneficial rather than some emulators [38]. Because while testing on the actual device, device performs same as real life, it has it's own load while testing. On the other hand the emulators have their own limitations, they are dependent on the hardware and software on which they are running.

1.2.2 Some Important factors of mobile testing

- **Selection of Mobile devices:** The selection of most suitable as well as diverse devices helps to get more insights.
- **Simulators or Real devices:** This is also important decision to take while testing for mobiles. Generally developers use both of the type of devices because all high end to low end real devices might not be available at each developer.
- **Performance:** This can be done by measuring load/stress of an application to measure it's behaviour.
- **Manual/Automated Testing:** There are different pros and cons of each testing method, automated testing has a fixed set of parameters and we need to go according to the application we are using while testing while in manual testing there is a scope for human error.

1.2.3 Quality assurance testing for Virtual Reality:

QA testing for virtual reality is an important thing while developing the application, because it provides developers insight about how application performs on the large scale, is there any health related issues causing because of the application, these queries can be resolved through QA testing.

- **Compatibility testing:** Compatibility testing helps developers when application gets spread across different configuration devices. It helps to check performance on the extreme lower end configuration to extreme high end configuration and then developer can do optimisations accordingly if application cannot run on the low end device.

- **Real device testing:** Instead of any simulation, real device testing is a meaure part of QA based testing. Because it checks hardware requirement is matching or not, it includes testing on the HMDs like Oculus rift or HTC Vive, or in case of mobile devices then testing on Google daydream like devices is sufficient.

- **Accessibility testing:** Using of VR applications can cause severs problems to human body like haziness, headaches, motion sickness, eye pain or more severely it can affect heart as well. Hence to keep the application according to the motion sickness guidelines is an important aspect. [15]

1.2.4 Parameters required for performance checking on mobile devices

- **Understanding CPU and GPU workload:** How application manages to use the available CPU and GPU is an important measure while performance checking we will go in depth of this type in the later part

- **Understanding Draw cells:** This parameter helps to understand how each point can

be drawn on the screen using internal draw calls. This analysis includes Mesh Data, Materials and properties associated which can be seen on the screen, Shader details as well, we will go through each of these terms in the later sections

- **Frame rates:** Frame rates on which application is running according to the real device
- **Display resolution:** The compatible resolution of each scene on which application runs comfortably on any device, this analysis helps to improve the space which application takes on the mobile device as well as speed of the application.
- **Physical and Virtual Memory usage:** RAM is an essential component for running applications smoothly, hence having min and maximum values of it is always beneficial.

Knowing all or some of the above parameters we can decide minimum requirement to run an application, before going to the actual analysis of the whole system, let's have some insights about motivation and objectives of this thesis.

1.3 Motivation

Virtual reality opened doors to the world for many research and development ideas. The virtual environment depends on the 3I's as explained above, and the developer's creativity. Besides that, the scalability of the project is also essential. There are many applications, software or systems throughout the world which does the same job we have done here but the main aim is to give easiness by creating a module which helps the user to create a virtual environment without writing a single word in a script as well as with just basic knowledge of Unity interface. The effort required to create a virtual environment from scratch with the use of the Unreal Engine or Unity game engine is a big task for a naive person; this thing motivated us to pursue this project further. Users can walk into such environments using HMD's or mobile devices without knowing many technical details as well. Many sites

across the world acted as underrated sites, which is having much potential to attract tourists. However, unfortunately, this didn't happen; hence to conserve and increase the popularity of these sites and spread the importance of it worldwide is our job to develop such virtual environments; this is another reason which motivated us to do such site conservation VR application. Of course, anyone can put any set of 360-degree images and create a Virtual Environment, but our primary focus is on the conservation of heritage sites. Hence in this project, we have focused on creating applications for Majuli Islands (Assam). This application provides an exciting way of interacting with the place to see the greatness of its mythological buildings, areas, and temples. There is a limitation on replicating the exact size in the virtual environment by generating graphics; hence we have used 360-degree images with the same details as the actual site. We wanted to develop an algorithm to navigate in the virtual environment, which needs to give close to the realistic experience. The first site viewing using VR is still in it's growing phase; hence we wanted to set a small example in its path of research, which can be fruitful to anyone In upcoming years for sure. Also, this project can help to spread more awareness among developers, content creators that the Virtual reality concept is not just about games, but it can be used in many other ways such as conservation of heritage, creation of the environment using direct 360-degree images. As after creating such hardware consumable virtual reality application, our motive is to deploy it on the mobile phones because everyone doesn't have HMD's like Oculus Rift or HTC Vive, but everyone does have a smartphone hence our motive is to optimize the application in such a way that it can be scaled onto mobile devices. Even after deploying to mobile phones, the main aim is to get performance statistics of the application on different kinds of configurations of mobile phones. We might conclude the minimum requirement to run these kinds of apps on mobile phones, which might be helpful for future mobile developers. These insights about performance on mobile phones are the second part of this thesis work.

1.4 Objectives

As explained in the motivation, we expect to get close to the real rendered virtual environment of Majuli and navigation in the VE. For this purpose, we have developed a whole module based on the Unity engine as almost 52% VR developers use this application, not only this but to convince the audience we have proposed a detailed study about different VR development platforms/engines and why we have chosen Unity only. Then comes the major part about we have created such a large environment, why we have chosen 360-degree images is the main thing to consider that we wanted to explain in the later part. How to capture and export 360-degree images and what will be the impact of exported resolutions all these things will be described in the following section. There will be two parts of this thesis, where in the first part, we will discuss methods about creating an environment and rendering workflow of Unity with its necessary terminologies as well. What is the significance of using C# script and how it is beneficial, we will focus on these topics as well. Navigation can be done in the created environment, and what convention we have used to create a grid for the environment is also an important topic to discuss. We want to give a spotlight on the human walking model as well. The hardware requirements to render such an environment and how different configurations affect the performance is also an important topic to discuss. In the second part of the thesis, we will look at performance measures for mobile devices. How different configurations react to different resolutions, frame rate, as well as the number of images. What are the limitations of mobile devices? What are the pros and cons while rendering on mobile devices? It is also an essential insight to give. As discuss above, different parameters affect the performance of the VR application; we will focus on in-depth insights about those parameters in the later sections.

1.5 Layout of the Thesis

In the introduction part, we have discussed the introduction of VR, different types of VR, and use cases of VR. As this thesis revolves around 360-degree VR and Virtual tours, we have given the necessary information about them. Then we have explained the importance of performance checking, dependent factors of performance checking, and some aspects about it. After that, brief information about Motivation and Objectives which need to accomplish at the end of this project. In the second chapter, related work, we have given information about the history of VR and its evolution in recent times. What are the available options of Game engines to create VR tours or applications has explained after that. Then there is an explanation about some commercial applications and systems; some of them are cloud-based, and some are desktop based. We have mentioned their pros and cons and insights about probable ways to tackle them. Then there is information about different benchmark testing applications for mobile performance measurement. A brief explanation of available profiling methods to capture mobile device performance explained in the remaining part of this chapter. The third chapter is all about the proposed methods of our whole work. Overall work has been divided into two parts, one is about developing a module to create a VR tour, and the second one is about the performance analysis of mobile devices. Initially, we have explained the required terminologies that are used anywhere in this chapter. Then an explanation about grid-based virtual environment creation, usage, and handling of 360-degree images and how Unity handles the VR tour creation. There are three sections (3.5 to 3.7), which explains our developed module, the scripts which help to run the module, and methods to navigate inside the VE created using the module. Later, there is information about different mobile devices used to carry out the complete performance analysis, the cube resolutions and image specifications used in this experiment and how we have gathered the data to analyze the performance from a profiling method. In the Observations chapter, there are observations about rendering times of a VR tour using our module by varying

resolution, number of images, etc. Then listings of observations about each mobile device and explanation about them. In Chapter 5, there are some conclusive and comparative statements and, finally, the future work which can be done from this much progress. We hope the audience will get maximum insights about our work after going through each chapter of our thesis.

Chapter Two

Related Work

As explained in the introduction, the Virtual Reality concept was first initiated in 1935. Then in 1965, Ivan Sutherland introduced a device that users can sense and feel the virtual world realistically. In the past few years after the introduction of Oculus in 2012, people have developed many different VR environments and games. There is much more research going on in this field, as well. Developing a VR application is not very simple. It consists, software engineering, architecture, AI and lightning effect, sound management, etc. Hence to combine it with many development platforms, engines have been developed in that way these past 8 years. For performance analysis, there is also much work going on from the beginning itself. Researchers and developers were trying to push the limits of their application by optimizing limited resources. Hence performance analysis is crucial to do while development; there is much work going on in this field as well. There will be two parts of this segment, one will be focusing on Virtual Reality related work, and another will be focusing on Performance-related analysis work.

2.1 Virtual Reality

There are many development platforms and development engines are available for VR development, we are using Unity Engine [41]. In this paper, Trenholme and Smith [42] presented

Engine	Scripting Language	Pros	Cons
Unity	C#	Flexible, great community support Robust asset store, The workflows supports 2D/3D env. Out of the box VR support. Supports more than 15 platforms (Mobile, PC, etc)	A closed source with expensive source code license Decentralized, need to right program shaders by own or purchase them. We need to attach colliders manually. Old looking IDE
Unreal Engine	C++	As C++ is gaining more popularity, UE has become choice for many developers. It has fancy custom workstation.	It requires more space and computational power. Not much convenient for 2D development. Costlier license
GoDot Engine	C# or C++	Well designed IDE. It has MIT license and it is open Source. Very lightweight (size around 30 MB)	Less tools in the Asset store, unofficial support of VR and AR platforms. Not mature as Unity.
CryENGIN	C++	Very convenient UI. Efficient for rendering outdoor environment	Poor documentation and less community support. Bad architectural pipeline compare to others

Table 2.1 Information about Virtual environment rendering tools

the different options available for rendering a virtual environment. Some of them comparable to Unity are listed as follows:

1. **Unreal Engine:** Unreal Engine's architecture allows everyone to create stunning visuals and that can be cast to lower end systems as well. It has a strong community support and the scripting language use by them is C++. [18]

2. **Godot Engine:** It is well known wide used MIT licensed game development and 3d rendering platform. Because of it's friendly content creation interface for artists, designers, animators, it has gained lot of popularity.

3. **CryENGIN:** This tool is mostly similar to Unity but by providing support for MacOS and Windows both, unity has some edge over it.

We have studied and described the pros and cons of the above systems to get more insights into all of them in Table 2.1

Our goal is to render 2D 360 degree images; for this, there is an excellent provision of tools by Unity as well as it gives flexibility while doing the same compared to others [2] [4]. The Asset store, which has a collection of third-party plugins and APIs, is arguably one of the best, very robust, and has excellent community support throughout the globe. For example, Google Maps API, Google Speech API.

There is a considerable amount of research and development that happened in navigation and rendering of the virtual environment. A project known as Virtual Museum [21] is the same concept as ours where user can move in all four directions and can view in a 360-degree angle, this project consists of 45 views of 360-degree images and videos. But as we are going to explain more details in the later part, how videos are space inefficient, and hence this is the disadvantage of this project that it requires a large amount of space required to store the data and a considerable amount of time to render the scenes. There are different types of environmental maps that were used to create virtual environments. The process of how to process anamorphic images optically or electronically to create a complete 360-degree view was submitted way back by a group of MIT students [22]. There is a classic example of using the grid-based network to roam in an environment called "Navigation" created by MIT students [18].

Figure 2.1 History of VR

Cave automatic virtual environments are one of the popular tools which support im-

mersive VR. Where in a single room, users can roam on the floor and the walls and floor, there are projected screens. The user needs to wear 3D glasses and can move around. In Heritage conservation of cultural places, this technology can be used precisely [28]. There is a method provided by Greene and Heckbert, which can be used to create a VR application by composting multiple image streams into a single one while rendering, which reduces load while rendering. Currently, virtual reality tools are more popular because they can't just be used in the gaming industry, but it can be used in the medical industry by neuroscientists and research, tourism industry [9]. There is a considerable amount of work has been done in clinical studies with the help of Virtual Reality.

The user's VR experience can be measured on the combined points of realism and reality levels. Heeter explains the study about the experience of realistic presence in an environment in his studies [5]. In recent times there is a hardcore review of Slater, which explains about VR application, including advantages and disadvantages in different research areas such as travel, education, behaviors, entertainment, and news. All of the mentioned techniques have been frequently used to create a realistic experience in some commercial applications and systems in VR environments. There is a definite amount of research happened in the navigation field of VR and which is used by many systems to create their commercial applications.

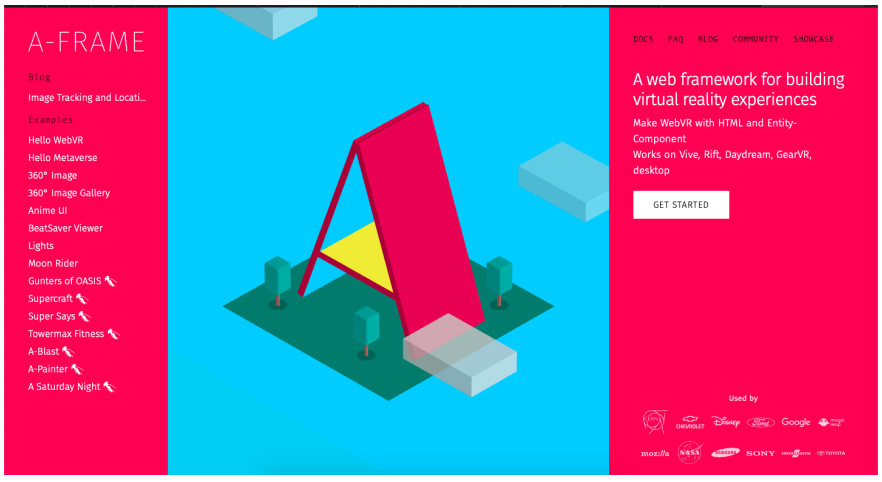

Figure 2.2 A-Frame Website

A-Frame is a web-based framework that provides facility to create cloud-based webVR applications that work on Oculus Rift, HTC Vive, or mobile devices. As it's web VR, there are many issues about ES6 and HTML rendering as well as if a user disables javascript, then it's impossible to run these applications [16].

2.1.1 Cloud-based VR tour creation tools

There are some cloud-based popular applications to create virtual tours. The main advantages of them are listed below:

1. Work from anywhere on the virtual tour in any supported browser and get rid of hardware dependency
2. If any issues found we can update them live from anywhere and frequently as well
3. Simpler workflow and no need to focus the background activities

- **Roundme [49]**

  An online cloud-based platform called "Roundme" where users can provide their set of 360-degree images, and by analyzing them, the application creates a virtual environment. But while creating such an environment, there is a minimal scope of customizing it. Also, Navigation in that system is not seems natural; this application only provides a virtual tour; somehow, there is nothing about giving realistic experience. Below is the workflow to create a virtual tour on Roundme application:

  1. We need to upload a set of images after clicking on the create tour tab.
  2. It analyzes whether uploaded images are panoramas or not
  3. It creates a VR tour by generating its 3D model of an environment
  4. It gives us the ability to define hotspots, maps, the field of importance, and many more features.

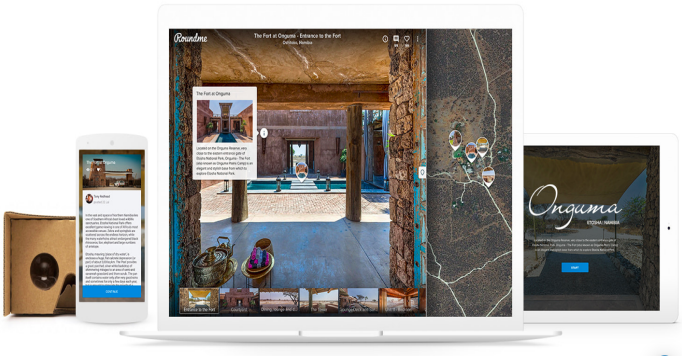

Figure 2.3 Roundme portability

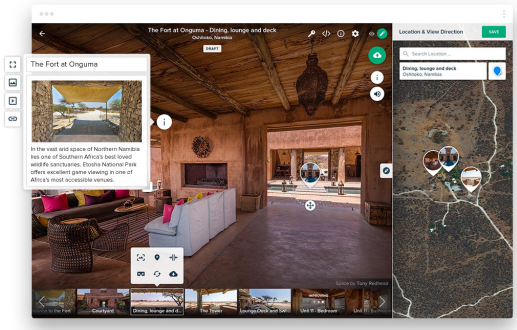

Figure 2.4 Roundme Dashboard

The main advantage of roundme application is, it's web-based and not system based. We can publish our applications online, and we can share them just by sharing the link of the application. But this comes with a drawback of compatibility with the browsers. The browser should have enabled JavaScript support; also, it should be compatible to run webVR. There is a fixed set of parameters that we can change in the tour created by roundme. But for simplified applications and to get publicity, this cloud base application is at the top for generating Virtual reality tours.

- **Ocurus [26]**

  Another similar application is Ocurus; this is almost a similar application as Roundme. The pricing of Ocurus is more comfortable than Roundme. The teleportation on Ocurus is not as good as Roundme because when moving forward, it directly drastically changes from one direction to another. The process to create a tour is as shown below:

  The features of Ocurus application are as follows:

  1. We can create a fully responsive web application which can work on mobiles, computer and tablets as well
  2. Feature to create introductory information screen for an application

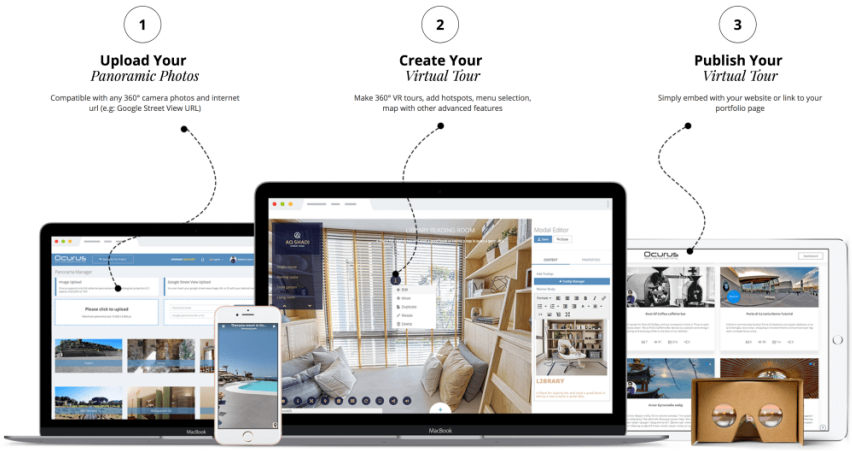

Figure 2.5 Ocurus: VR tour build process

3. Online editor with real-time facility
4. Creations of custom maps, hotspots
5. Analytics about the tour, how people are interacting.

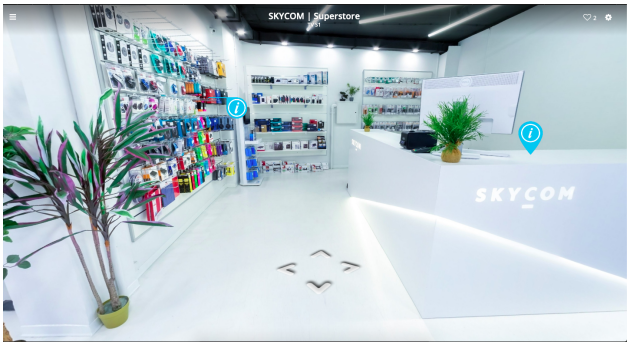

Figure 2.6 Ocurus Virtual Environment

The latency and quality depend on internet connectivity, speed. Also, speech-based navigation or gesture-based navigation is not available.

- **GoThru [17]**

  This is a totally Google StreetView based web app to create virtual tours. In this application, we can upload stitched images that are captured using Google StreetView

application and publish it directly on the same as well. Google StreetView is a technology that helps to create equirectangular panoramas using mobile devices. Initially, it was released with Google maps and Google Earth, but now there is a separate application of StreetView for Android or IOS

The GoThru dashboard is shown below:

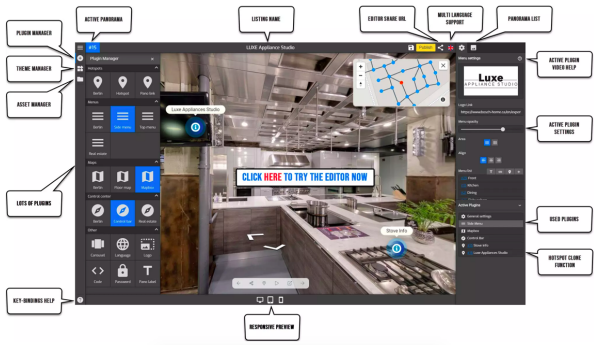

Figure 2.7 GoThru dashboard

Features of GoThru:

1. Predefined Themes to quickly work on
2. There are plugins available to generate hotspots, maps, and many more.
3. It has a feature called Progressive web application, which helps to generate IOS or Android-based applications for the tour
4. It also has a statistical dashboard to get analytics about the tour

GoThru application is more comfortable to work because it supports directly captured images from Google StreetView, but these images are hard to capture using our mobile devices. Also, there is no availability of gesture-based navigation.

There are many more cloud-based available tools to create virtual tours such as Lepentor, WalkInto, Spinattic, VRMaker. More or less every tool does the same job, according to

need, suitability users can choose among them. But there are problems with this cloud-based VR applications. The main concern is security. Some tours might have a national level importance site and might contain sensitive information underneath, which is risky. Another problem is working on all these cloud-based applications depends on network connectivity. While working on all these cloud-based apps, developers need to follow a strict set of guidelines given by the application, which sometimes makes it difficult for creativity. Hence there are some offline applications available that help creates virtual tours offline and export them for different platforms.

2.1.2 Desktop based VR tour creation tools

- **3D Vista [1]**

  3D vista is one of the famous Virtual reality-based tour makers on windows or mac. It is dedicated only to create virtual tours. It helps to create 360-degree interactive virtual tours. With the use of 360-degree videos or images(panoramas), a user can create pleasant virtual tours in which we can embed different sounds, visual effects. We can create hotspots, make them interactive, and add buttons, but we cannot add scripts on any element. Here is the main dashboard of 3D Vista Shown below:

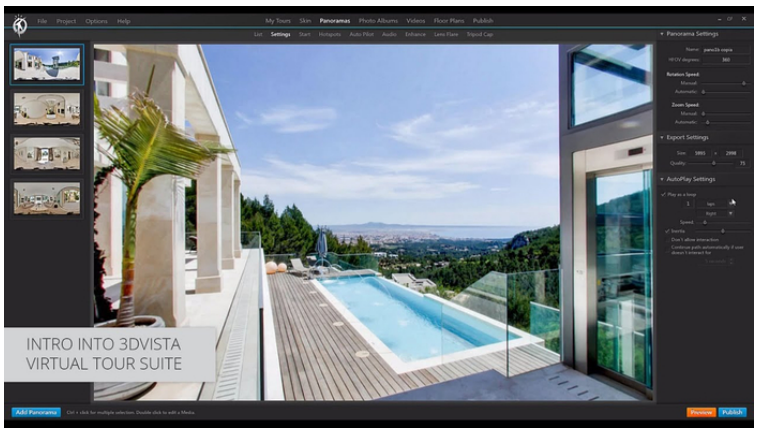

Figure 2.8 3D Vista: Development Window

3D vista is the overall package, which provides support for photography sessions as well. This application provides predefined floor plan templates as well as a mobile hosting service. It provides up to some extent, coding facility as well. With the help of coding in C/C++, we can generate maps, visual effects.

Features:

1. Synchronization of all tours across all accounts
2. Online and Offline working facility
3. Hosting services provided by 3D Vista
4. Underwater photography and videography
5. Provision of Interactive VR tours
6. Package for Aerial photography and Video services

3D vista has excellent community support, which helps to solve any underlying difficulties while creating a tour. It doesn't have gesture-based controls. We can generate .apk for android and .app for IOS devices for the generated virtual tour, which can be automatically scalable for mobile devices according to their screen sizes.

- **OpenSpace 3D [27]**

  Open-source platform to create VR and AR applications. It has excellent community support and doesn't require any coding knowledge. This tool mostly used for business purposes. It has excellent options available to create Augmented Reality or Virtual Reality applications. The leap motion feature inside the tool provides the facility to create Virtual Reality applications. It has easy documentation and very easy to use the tool.

  Features of OpenSpace 3D:

  - Augmented Reality support

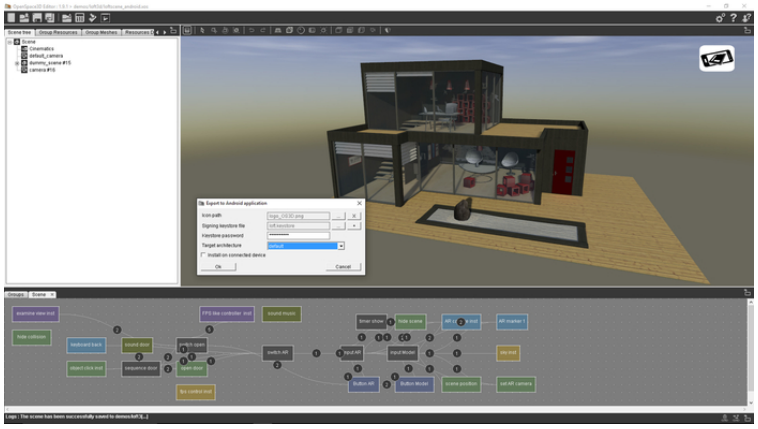

Figure 2.9 OpenSpace

- Visual connection based programming
- Library of 3D models
- Feature of Leap motion

- **Pano2VR [29]**

  Helps to create 360-degree virtual tours from panoramas or videos. It helps users to export tours for web browsers with built-in support of HTML5. Facility to export projects of Android, IOS, or desktop app. Different transitions which help to navigate inside the tour from one scene to another. Predefined map structure or facility to integrate the whole tour with Google Maps.

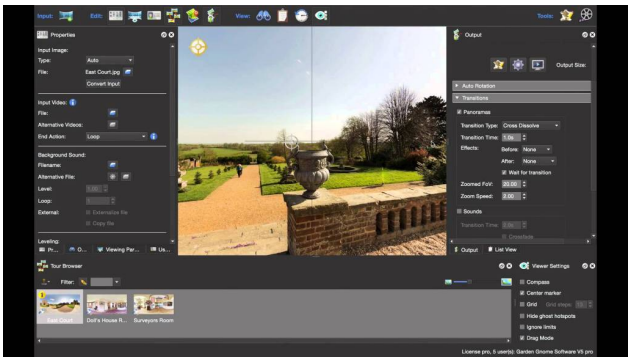

Figure 2.10 Pano2VR Development Window

Features of Pano2VR:

- Facility to improve 360-degree panoramas (Removal of tripods)
- Publish tours on Google StreetView
- Directional sound effects
- HTML5 based Integrated webserver
- To publish a tour on any Wordpress site, it has inbuilt support.

Pano2VR provides animation based scene navigation where user can create animated paths which automatically navigate to the significant scene. Users can export this as walk through video file.

Comparison based insights about one Cloud based and one Desktop Based VR development tools has shown in the Table 2.2 below:

Editor	Platform Supported	Pros	Cons
Roundme	IOS, Android	Cheaper compare to 3D vista, more interactive for beginners or who don't have knowledge about this field. Good documentation Lightweight because of mobile support.	No platform support for Windows or Mac, hence customizations are limited. No speech-based navigation, No AR support.
3D Vista	Mac, Windows	Very much popular on big platforms such as Facebook Amazon, etc. Crisp and clear images, Hotspot creation Good documentation	Not natural looking navigation experience, Jumping from one hotspot to another is slow. No speech-based navigation

Table 2.2 Information about Cloud based and Desktop based VE creation tools

While summarising these cloud-based and desktop-based VR tour creating applications, all these have their pros and cons. The advantage of these cloud-based applications is, no requirement of top-notch hardware to create a VR tour and to render it. The cloud servers are capable enough to handle the load. The main disadvantages of cloud-based applications are Internet connectivity and security. If the rendered scene quality is high, instead, we can say 720p or 1080p or 4K, then it takes much time to load the tour, and it consumes too much internet. Every application is different in some aspects; otherwise, they all have

almost the same workflow and architecture. In the case of desktop based applications, there is a minimum requirement of hardware and software instructed by the application; this may or may not be an advantage for some users. Those who have higher-end machines with them, they can adequately utilize the capability of desktop based applications. Most of these desktop based VR tour creators don't provide gesture-based action support. If we want to embed some 3D models in the VE, then maximum applications don't support it.

There is a big problem in VR applications of motion sickness, hence there were many efforts taken have been taken to reduce it's amount while running VR systems [7] [25]. Many experiments were conducted on people to capture Human's linear walking speed, accelerations while walking, running as well as head movement speed. The user's discomfort and environmental presence are greatly linked together in the body [8]. And this discomfort can be used using different models of navigation, proper frame rate, etc. The causes because VR can sustain up to 10 mins to 3 hours, and hence this is the developer's responsibility to control the bad things that can happen during a live session of VR applications.

2.2 Performance checking in mobile devices

After developing a VR application on the computer with the help of Unity or any engine, to scale it on a broader base, either we need a computer-executable file or mobile device executable file. After generating that free executable file, many parameters affect the profiling information before and after installing it. We have thoroughly explained it in the latter part of the thesis. Before that, let's discuss some related software for profiling or performance testing of an Android device. The terms profiling and performance testing depict the same meaning here hence no need for any confusion.

The first mobile phone was launched in 1973. Back then, it was much more substantial and bulky compared to nowadays slim, but bigger screen sized mobiles. Since its discovery and after the introduction of smartphones, it has become an essential part of our lives. We

can do almost everything on a mobile phone, consider it is shopping, messaging, calling or video calling, playing games. The Physical memory of these smartphones has increased from 100MB to almost 8 or max up to 12 GB. Even most of the computers don't have these many RAM sizes. Still, smartphone application developers find it challenging to manage the workload of CPU and GPU, memory management, etc. compared to computers. The limitations of mobile phones are lack of Virtual memory and Limited power.

User experience is the topmost priority for any application in mobile devices, but profiling an application doesn't consider how it feels outside. However, profiling gives us information about internal factors inside the mobile devices, which affects the application's user experience. The essential things that need to consider while profiling are: [23]

- CPU Usage
- GPU Usage
- Memory Usage
- Battery consumption
- Network Usage

These are the parameters to be considered while mobile device profiling depending on the requirements. After developing and rendering the application on the desktop or workstation, we need to create a free executable file and that we need to deploy and install on the device. To check how well an application supports mobile hardware and software, there is a need for profiling and information gathering; accordingly, developers can optimize their codes. Also, measuring the performance of the device and comparing it with the other device's benchmark scores is essential. Benchmark score is a total measure of each vital component of a device such as UI, CPU efficiency, Battery optimization screen frame rate, and all of these scores add up to make a single benchmark score. These are not application-specific scores, but these are a measure for a mobile phone condition, and based on that, we can provide the

minimum required benchmark score to run our application. There are many online or offline applications available on the android play store to check benchmark scores; out of all, some popular are as follows: [32]

1. **Quadrant Standard Edition (QSE):** This tool can test CPU performance along with 2D or 3D graphics testing. It's an online application, where after running the tests, scores will be uploaded on its servers and we can see the comprehensive graphs and scores.

2. **Linpack:** This is a well-known benchmark testing application that gives CPU performance scores in terms of MFLOPS (mega-floating point operations per second). It is the easiest and most trusted parameter for performance checking [51]. This application is popular because of its benchmark testing application for some of the prods fastest computers.

3. **Neocore:** This tool first used in Qualcomm's Adreno GPU to check the performance of CPU and GPU but because of it's more accurate results it made public later on.

4. **AnTuTu:** This is also a well-known application for benchmark testing in the whole industry. This application has a major fanbase. This application tests performance of CPU, GPU, Memory, etc. Its server has benchmark scores of almost all of the devices in the world.

To get profiling information of the particular application, above listed devices may not be useful for that specific purpose we need specific profiling applications for android profiling. We can do trusted android profiling with the use of Android Studio profiler [20] or Dalvik Debug Monitor Server (DDMS) [3] or a native profiler. All android versions have it's own DDMS irrespective of themes or profile on it, and it is IDE independent. We can directly launch it from Command Line or terminal. DDMS can give information about General System info such as usage of FPS, CPU, and GPU, etc. as well as threading information,

Logcat. On the other hand, the Android studio profiler can provide more visualized and detailed information. It can show real-time usage of CPU, GPU as well as packet transferring in networking in graphical form. Native profiler is not the best way to do profiling but it's good when there is no availability of computers.

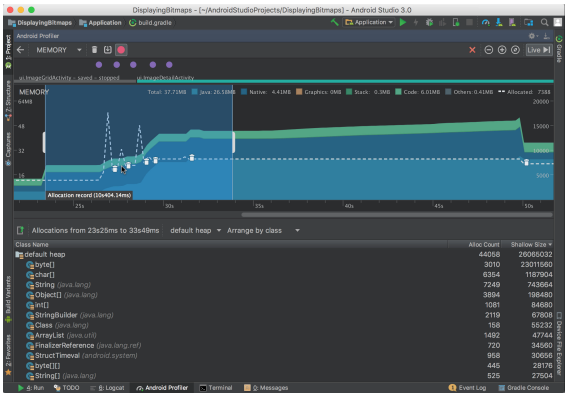

Figure 2.11 Android Studio Profiler

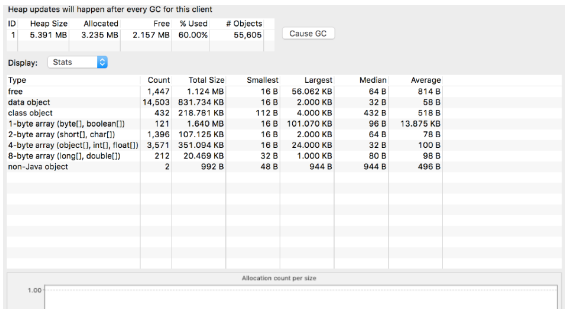

Figure 2.12 Dalvik Debug Monitor

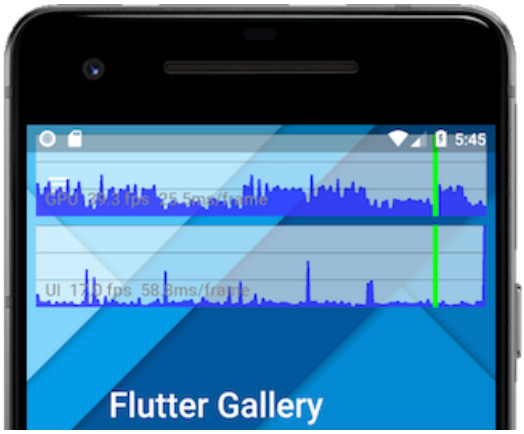

Figure 2.13 Native Android Profiler

Among these, there are more tools available that can find bugs in the application which can affect the performance of the application like Little Eye and Test Fairy. For our application, we are restricted to check CPU usage, GPU usage, and memory usage. Hence any listed above profiling tool is sufficient. Proper methodology and observations about profiling will be explained in the later sections of the thesis.

Chapter Three

Proposed Work

The total work has been divided into two parts as all the above sections notifies it. The first part is all related to the rendering of a virtual reality environment and the second part is all about rendering on the mobile devices to check performance analysis and profiling. We will proceed step by step in this section by covering all the required information to reach towards observations. This section gives the audience an overall idea about all our modules, device setups, all the difficulties, etc. before proceeding in any portion we are going to introduce some terminologies which will be helpful throughout the further thesis.

3.1 Terminologies

The following listed words will be commonly used throughout the thesis, as some of them are related to specifically Unity Engine and some are general terms used in VR development.

- **Shaders:** A script that contains mathematical formulation and mapping information each pixel as well as some algorithm to calculate the color of each pixel rendered from Material.
- **Materials:** It has properties like color tints and includes textures in it to define how the surface should be rendered.

- **Textures:** These are bitmap images of an original image. A material can contain a reference to one or many textures so that shaders can apply its mathematical formulation to them.
- **Bitmap Images:** These images are a series of tiny dots called pixels, each pixel is a small square that has some dimensions and color properties.
- **Rendering:** It is the process in which a 2D or 3D model is generated from an image and shown on the display
- **2D texture:** It uses 2D space to map the position of each pixel into a value with respect to the center point
- **CubeMap:** It maps the direction of each pixel to value unlike the position in 2D texture. Cubemap tries to consider the direction of each pixel from the center. It has 6 sets of 1 texture which can be splitter on 6 sides of a cube.
- **Skybox:** It's the wrapping around the entire screen beyond the geometry

3.2 Idea about the Grid

Before proceeding to the actual generation of Cubemap and applying 360-degree images, there is a need for discussion about how to create the structure of this virtual environment. Grid is a simple N x M matrix that has length and width according to the area which we want to cover in the environment. Now here comes the question about what will be the size of each box in the grid? considering all optimal situations about space to save the images and rendering optimally we have taken one survey about human step size. The idea behind this survey is, we should change each image as human proceeds step by step. Hence the survey gives information about actual human step size which is shown in the table 3.1. As shown

in the table 3.1, the average indoor step size of humans is 18 inches whereas the average outdoor step size of humans is 27 inches.

Heights (in cms)	Indoor step-size (in inches)	Outdoor step-size (in inches)
157.48	17.0	24
159	18.1	24
165.10	18.6	26.9
167.64	18.5	27.1
172.72	18.9	27.4
175.26	19.1	27.3
177.8	18.3	27.8
180.34	18.2	27.5
185.42	19.2	28.3
187.92	19.0	28.1

Table 3.1 User information about indoor and outdoor step-size

Considering the average indoor step size is 18 inches, there can be two cases in any environment, the whole environment is empty or there are obstacles in some parts of the environment. While considering the clean scenario where there aren't any obstacles total number of minimum steps required to cover the whole area can be formulated as:

$\frac{(X-18)}{18} * \frac{(Y-18)}{18}$

where x and y is the length and width of the environment. Now here is an important question why to reduce that 18 inches from length and width, so if the area is covered by boundaries or walls from all of the directions then we need not take pictures in the first and last row as well as columns.

If we are going to take in an outside environment with indoor step size then no need to reduce that extra 18 inches. then the formula will be:

$\frac{(X)}{18} * \frac{(Y)}{18}$

Now consider a room with obstacles as shown in the Figure 3.1, where part A represents a room with obstacles which sticks to the boundaries, then as shown there total length of all three obstacles is (A+B+C) and the total width of the obstacles is (P+Q+R), hence the minimum number of images required are l x m, where

$$
\begin{aligned}
l &= \frac{[(X-18)-(A+B+C+18*3)]}{18} \\
m &= \frac{[(Y-18)-(P+Q+R+18*3)]}{18}
\end{aligned}
\tag{3.1}
$$

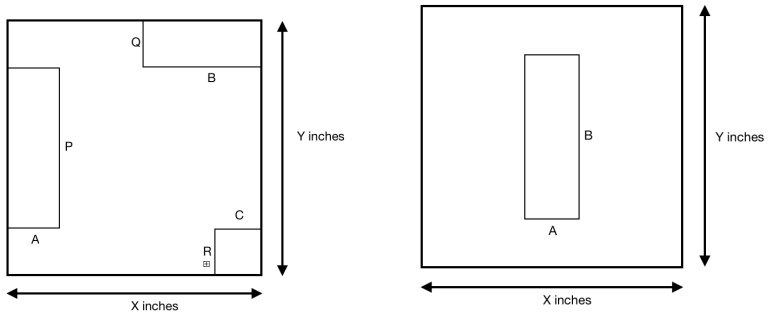

Figure 3.1 Images of room with obstacles sticking to the boundaries and not sticking to the boundaries

Where the extra 18*3, is because of the deduction of 18 inches from each obstacle boundary because the nearest image from each obstacle is 18 inches away from its boundary. Now image B represents an obstacle that is not sticking with the boundary then while calculating length we need to calculate it two times. because form both the sides of an obstacle there we need to keep space of 18 inches hence formulation of the minimum number of images will be = l x m

$$
\begin{aligned}
l &= \frac{[(X-18)-(A+36)]}{18} \\
m &= \frac{[(Y-18)-(P+36)]}{18}
\end{aligned}
\tag{3.2}
$$

Now while generalizing the formula for the number of images required with obstacles stick to the boundary and obstacles won't stick to the boundary is:

$$
\begin{aligned}
l &= \frac{[(\text{Length}-18)-(\text{Total length of obstacles}+18*(\text{Number of obstacles}))]}{18} \\
m &= \frac{[(\text{Width}-18)-(\text{Total width of obstacles}+18*(\text{Number of obstacles}))]}{18}
\end{aligned}
\tag{3.3}
$$

Maximum number of images = l X m

After getting maximum number of images required for an environment, all the images will form a grid structure as shown in the below Table 3.2:

img_0_0	img_0_1	img_0_2	-
-	img_1_1	-	img_1_3

Table 3.2 A grid for a room with 3 obstacles

Here the grid has a total of 8 places to capture the images but maybe because of obstacles or irrelevancy, there are 3 spaces left blank. Now if the user starts at (0,0) then if it moves right or to be precise by viewing right and moving forward, he will moves to the position (0,1) and the same as in all the 4 directions only if the image is present there. Now, this is an array, but what if there will be too many blank spaces means obstacles, then anyway those spaces will remain empty and it won't take space in memory.

in the next part how we take images and what's need for 360-degree images will be explained.

3.3 Significance of 360-degree 2D images

Rather than creating virtual models, graphical environments, or 360-degree videos, 360-degree images are the most promising way to create a virtual environment in this case. But what's the significance of using these images?

3.3.1 Why we need 360-degree images?

These are images captured of real scene hence this the way of getting the most natural-looking view of the real environment. Generation of 3D graphics or augmentation of 3D models to the scene which will then create a view near about real environment is not a

good way here because basically, it's computationally hard and time-consuming as well because of its complex simulation and rendering. The rendering quality of this generated environment depends on two factors, one is the developer's consistency and the second one is hardware and real-time constraints of the consumer [45]. Whereas the gears and technology required to capture 360-degree images are developing exponentially. Many different quality devices, cheaper to expensive are available in the market to capture 360-degree images. The commonly known cameras are LG 360, GoPro, VUZE 360, Kodak SP360, Samsun Gear 360, Insta 360, etc. [34] We have user VUZE 360 camera which has the following configuration:

1. **Cameras:** 8 16MP fish-eye lenses
2. **Processors:** Two Umbrella A9 Video Processors
3. **IP Rating:** IP65 Dust and water jet proof
4. **Wifi:** IEEE 802.11
5. **Lenses:** 8xF/2.4 Fisheye lenses (100-1600 ISO)
6. **Sensors:** 8 Sony FHD image sensors

These 360 images are having a total unbroken view of the whole region as well as the rendering of these images is computationally less expensive also the main advantage is these are actual images of the environment hence it offers great realism to VE. Some of these cameras can provide 360-degree images as well as videos. But why to choose only images? To capture 360-degree images there should be some stable gear or robotic arm available which can hold the camera and roam around, also these cameras don't have much stabilization. Now the main drawback of 360-degree videos is, it has much more size required compared to images. For example, one 360-degree image with resolution (3840x1920) requires 2MB of space while only 2.5 seconds of HD 360-degree video required 60-70 MB of space, hence this can affect rendering time too much while rendering for larger areas.

Many developers are now trying to augment 3D models while keeping the 360-degree background, where users can even interact with these 3D models [13] [14]. If proper augmentation has been done then this augmentation can provide natural experience.

3.3.2 How to capture 360-degree images?

Capturing 360-degree images is not a rocket science but we should follow some methodology while capturing and exporting these images. There are many different camera available to do this as explain earlier, mostly these cameras has 8 lenses or one big lens which combinely having 8 lenses like GoPro Camera. Good practise while taking images is to use monopod or tripod. These cameras can capture a whole 360-degree image in one shot otherwise there is a difficult option of combining 6 (forward, backward, left, right, up, down) images captured by DSLR to make a Cube. All these 360-degree cameras can provide cubical as well as equirectangular type of images. Cubical format has 6 undistorted and perspective images from the directions: forward, backward, left, right, up, down. Where as equirectangular image has 360-degree view horizontally and 180-degree view vertically. In our case we have used equirectangular images.

Figure 3.2 Cubical 360-degree image

Figure 3.3 Equirectangluar 360-degree image

There is a software for VUZE 360 camera, where we can export our captured images

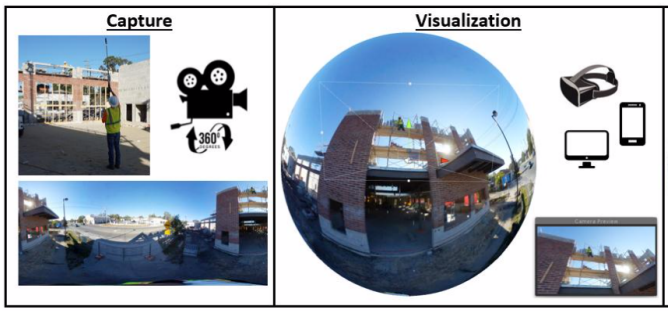

Figure 3.4 Capturing and Visualising 360-degree images

named as "VUZE Studio" which is both available on MacOs and Windows. We can export captured images into required specifications and resolutions. For our experimental purpose we have exported images in 2:1 ratio as our project will be in the landscaper form. The exported resolutions are 3840 x 1920 and 1920 x 960. All images are in the JPEG format with color specifications a RGB 24 [11]. These images can be transformed into the sphere into the applications like Unity.

Another simpler way to capture these images is to use Google StreetView application. Where according to app's instructions we need to move our mobile phone into 360 angles. But there is too much possibility of human errors hence this option can be only used where there is no gears available.

3.4 Cubemap creation workflow in Unity

Form this section onwards we are exploring how unity works while creating a cube map. The motive of this is to create a cube map from a simple equirectangular image to render on the HMDs. As explained in the above section, these exported images from the VUZE studio can be directly used in Unity.

We need a Skybox to represent the created Cubemap of an image while rendering. We can represent these cube maps with the sphere or skybox. But we have used skybox here

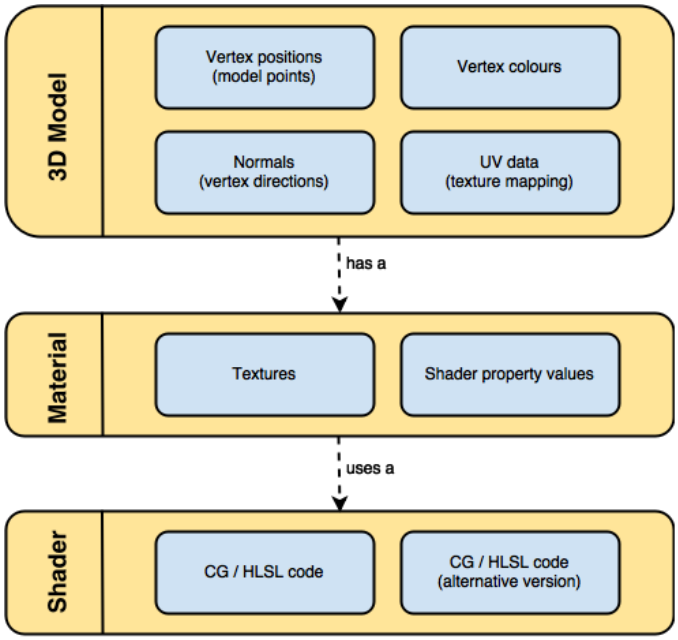

Figure 3.5 Unity Workflow of Creating Cubemaps

because it is easier to integrate textures into the skybox and also skybox is infinite hence it never collides with the geometry. To integrate these cubic textures of the image we need some material, here we have chosen skybox material. These textures with the help of some shaders will get integrated with the skybox material and we can render required skybox at each grid point in the virtual environment. In the Figure 3.5 the general model description can ve explained. A final 3D model which we can see after rendering on the screen on HMD has all vertex information about the material which encapsulated in it [31]. The material has all the mapping information about the texture with the help of the shader included in it. This newly created skybox material can be applied to the world using a lightning setting in unity manually. Hence by moving In any four directions, we can render an appropriate skybox, more on this will be explained in the navigation section.

3.5 Developed Module

In this section, we will go through an in-depth working of our module and how it is beneficial to the user. Before we move onto procedure let's have a look on the minimum requirement to run Unity on any system running Windows OS or MacOS

Minimum requirements	**Windows**	**MacOS**
Operating system version	Windows 7 (SP1+) and Windows 10, 64-bit versions only	Sierra 10.12.6+
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-capable GPUs	Metal-capable Intel and AMD GPUs
Additional requirements	Hardware vendor officially supported drivers	Apple officially supported drivers

Table 3.3 Minimum Requirements for Unity [3]

3.5.1 Development Process

Creating a virtual environment with a module which requires least user interaction with Unity:

we have developed a module on top of the Unity layer which consists of different methods in C-sharp to generate textures, cube maps, and shaders from a giver set of images that directly outputs a rendered working Virtual Environment based on Unity. The upper-level working of our module can be explained as: User provides a folder path of the set of images using which he wants to create a virtual environment using the Grid structure. Each image must follow a particular naming convention, which is readable to the developed code. Then our module accepts the given folder path, the grid size in the form of length and width, and the starting point of the virtual environment in (x,y)coordinate basis. All of it then proceeded by our module using its methods and functions by calling some system calls, Unity calls and outputs the rendered environment. The initial window of our application is as shown in the Figure 3.6.

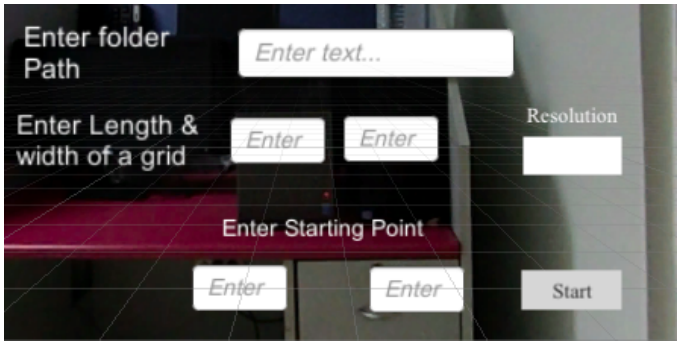

Figure 3.6 Initial screen to create a VR tour

The in-depth workflow of the module can be explained as follows: To develop this module we have to use C-sharp as a scripting language, as it has already explained that Unity only supports C-sharp language. The base class from which unity starts to execute the program is MonoBehavior class [43]. It is the same as the main class in C or C++. GameObject in unity is a container for every UI element which can be displayed on the screen, it helps to create a connection between UI element and script. These UI elements consist of Text boxes, Buttons, Input Boxes, etc. GameObject represents the character, object, and scenery. We can typecast the game object into any required component [41].

for eg. Button x = (Button) (GameObject name which holds Button UI)

this can typecast the void GamObject into Button. The first function is to create a 2D texture from the provided equirectangular image. These 2D textures hold information of each pixel of the provided picture in the RGB24 format. After giving the folder path, we are iterating over all the JPG formatted images in the folder and the method of creating 2D texture is called on every image thus creating the texture of each equirectangular image. then we store these set of the image in the form of the grid which has the same size as the original grid. The constructor to call default texture 2D converter class, is

Texture2D(width, height, texture format) [44]

There is by default some functions are provided in the C-sharp script which can be

readable by Unity. Update() function is one of them, which updated makes changes to the screen after updation of every frame. This can be used to change the position, movement, and state of the game objects [39]. Users can navigate as well as users can do head movement using this function, which will be explained in the navigation section. The generated 2D texture form the image will be provided to the function which can create a Cubemap from a texture that can be used as a material to apply on the skybox. To create the cube we need to define a 3D vector that consists of 6 cube faces in the form of 1's and 0's. we need to define the 4 edge points for each face concerning the center of the cube in the form of vector. The definition of a 3D vector is

Vector3(int x, int y, int z) [40]

where x, y, and z are the points in a 3-dimensional plane which defines the position of a point. Before the creation of Cubemap, we need to create a Cubemap object by defining cubemap function.

cubemap(resolution, textureformat)

The resolution parameter helps to define the PPI density of the created cube map, we have selected resolutions as 256, 720, 1080 pixel sizes. The color of each pixel is then defined by averaging the surrounding pixel. While creating a cube map from a 2D texture first task is to calculate polar coordinates corresponding to each pixel of a texture. the second step is to map those polar coordinates to the corresponding face of the cube. The process is as follows if coordinates of the spherical image on the 2D texture are (i, j) and width and height of the image is "w" and "h" respectively then normalized coordinates given by:

$$x = \frac{2i}{w-1} \qquad y = \frac{2j}{h-1}$$

Then the polar coordinates θ and ϕ are then derived from the normalized vector points x and y. The rage of θ is $(0, 2\pi)$ and range of the ϕ is given as $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. The equations for them is as follows:

$$\theta = x * \pi \qquad \phi = y * \frac{\pi}{2}$$

From these polar coordinates, we then move to the second step which defines x, y, and z coordinates of the cube. where x is the right direction, y is upwards direction z is front direction.

$$x = \cos(\phi) * \cos(\theta)$$
$$\mathrm{y} = \sin(\phi)$$
$$\mathrm{z} = \cos(\phi) * \sin(\theta)$$

These points specify position on the particular face of the cube. Using this fomula for each pixel of the image the overall cube map has been created. Our generated environment from an equirectangular image can be shown using Figure 3.7 and Figure 3.8

Figure 3.7 Equirectangular Image

Figure 3.8 Developed Environment

The advantage of this method is simplicity and less hesitation while creating a Virtual environment. The user just needs to provide a folder path that consists set of images and the module will then give ready to use rendered VE. Here is the trade-off between rendering quality and time required to generate the environment. Because when resolution increases then the time required to generate the environment increases and vice versa. The time and resolution based observations will be explained in the observation section.

3.6 Navigation in the Virtual environment

After rendering the Virtual environment in the form of a grid, the next thing is to navigate in the environment in the best possible natural way. Here we are going to learn about the navigation methodology that we have used in this project. The model which we have proposed for navigation, surveys, and also speech-based navigation approach. Let's discuss Some terms about the head movement before we head towards actual model:

1. **Yaw:** Vertical axis based head rotation
2. **Pitch:** Lateral axis based head rotation
3. **Roll:** Forward and Backward axis based head rotation
4. **Teleportation:** It is a method of moving a user from one point to other in the virtual world

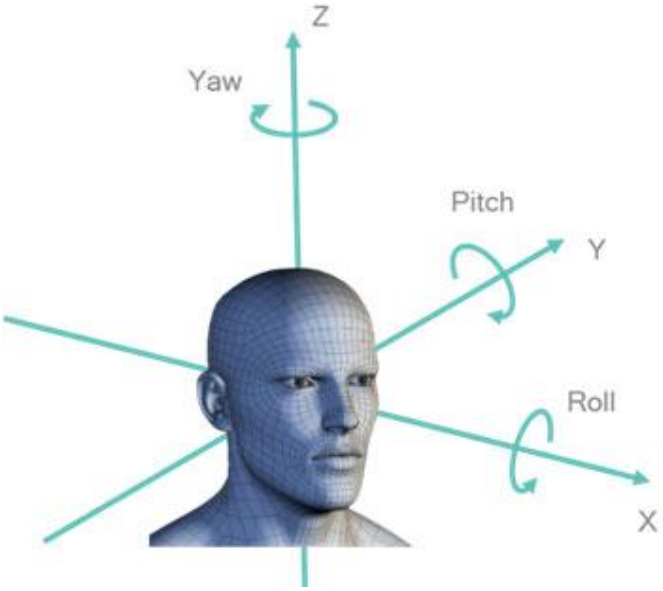

Figure 3.9 Head movement terms

3.6.1 Grid based navigation approach:

A grid structure works as a total virtual map for the environment. In the overall grid, some points will have Cubemaps and some will be empty because they considered obstacles. The advantage of the Grid-based model over the typical path-based model is, we have freeness to move anywhere inside the environment unless some points considered as obstacles. From a point, the user can move in any direction by moving in the left, right, forward, or backward direction. Now one point to be considered while movement is if there are two rows which are parallel in a grid and there is a wall in between them in the actual environment then in the virtual world as well we should restrict the user while direct crossing the wall when moving from this row to other. Hence we have found a method for that is, while taking input of the image folder path, we are taking one file input as well which has obstacle information in it. We then read that information in the script and accordingly restrict the user from jumping over walls or similar kinds of conditions. These small things make more impact on the user's experience while moving in the environment.

There are some issues when we move into the virtual environment because our mind thinks that we are moving somewhere because of visuals on the screen, but our legs and hands are mostly still hence the user can feel motion sickness while navigating into the environment. This issue has been in discussion for many years now by developers or scientists. Many types of efforts have taken by developers to generate natural feel or close to natural feel while moving in the environment [25]. To create the most natural experience while navigation, forward linear speed should not be more than 3 m/s and backward speed should be less than 1 m/s. The forward and backward acceleration should be not more than 0.1 m/s^2. In terms of head rotation, the angular speed should be lesser than 5 Degrees/s. [7] [35]

This grid-based model can be applied to any kind of Virtual environment just by taking appropriate length and width of the environment such that not a single point gets missed. Also, the grid model is beneficial because we can easily visualize the map of the whole

environment using grid points and hence we can directly point out whenever we are going wrong while development.

The next question arises is how we are going to start to navigate in this environment? That means how we are going to move forward or backward or left or right? Typical answer on the computer is keyboard arrow keys! But here's a catch we cannot click keys when we are wearing HMD. Hence we have found two ways, speech recognition based navigation and head movement based navigation.

3.6.2 Speech recognition based Navigation

Here we can simply put 4 keywords and computers can detect them and move accordingly, this is the basic idea. On windows, we have used Windows speech recognizer which is free of cost But which cannot be used on Mac or Mobile phones because it's made for Windows operating system only. Hence we have moved on and used head movement based navigation.

3.6.3 Head Movement based Navigation

The idea behind this type of navigation comes from a natural sense of walking. When we walk, we check on the downside of the front and move forward, similarly, if we want to turn right then we turn our face to right check for the path by looking slightly downwards and move on. This concept we have used in this type of navigation. Here we are using different angle based detection of a head position or simply mobile position and then move accordingly.

Here in the Figure 3.10, we can see the horizontal and vertical axes which control head movements. Here there are ranges about 4 directions in the environment. Around 360 degrees, ranges for **Forward movement:** >325 to <=25, **Rightward movement:** >25 to <=140, **Backward movement:** >140 to <=225, **Leftward movement:** >225 to <=325. And to move forward user needs to look downwards in the x-axis ranging from (25-degrees to

Figure 3.10 Possible angles of head movements

80-degrees). We have dynamically shown the arrows according to the possibility of moving in a particular direction. From Figure 3.11 and Figure 3.12 we can see the program dynamically changes the arrows according to the possibility of movement.

Figure 3.11 Movement possibility 1 (L,F,B)

Figure 3.12 Movement possibility 2 (R,F,B)

3.7 Module Implementation

In the previous section, we have seen the module's development concepts, where all the required processes to create a module have been explained. In this section, we have explained about the actual implementation of the module. Where we have given a set of images and out of it, a tour has been created by the module. For this initial implementation, we have

used panoramic images of Majuli island taken using VUZE 360-degree camera.

3.7.1 VE creation of Majuli Island

Majuli [19] is a river island on the Bramhaputra river, which is the first Indian island to be made as a district. It had a total area of 880 sq. Km at the beginning, but due to erosion, it has drastically decreased up to 352 sq. Km. It is the world's largest river island listed in the gunnies world records. Because of its decreasing area, it is an excellent choice to make a VR tour on it for its preservation and spread awareness about this site. This island is accessible by the ferry from the city Jorhat in Assam. It is around 350 km from the city of Guwahati. The island is a primary center of the Assamese Neo-Vaishnavite culture, which has many "Satras" or Monasteries. In this initial implementation, we have covered one of its Satras, North Kamalambari Satra. A team from the UCCN lab has taken images of this Satra.

Figure 3.13 Image naming convention for overall grid

The whole internal part of the Satra has one main praying area, and around there are rooms for devotees. Our team has taken 88 images of the Satra in the 23 x 12 sized grid structure. Some of the parts of the grid are blank because those are outside of the relevance. All these 88 images have spread across the whole of Kamlambari Satra. As we have previously

explained about maintaining the naming convention is the most important part of the whole module because, based on the naming convention of each image, the module extracts the position of each image and creates cubemap and place accordingly inside the grid. In Figure 3.13, we can see the naming convention about these images. Each image has named as "img_posX_posY".jpg. The module code extracts posX and posY from the name.

All these 88 images are in the resolution 1920x1120 as explained earlier. Figure XXX shows the initial screen before rendering the tour, which has the path of the folder where all these images are stored, then there are dimensions of the grid and starting position of the tour. Then by providing a resolution of the cubemap, we can start to create a tour.

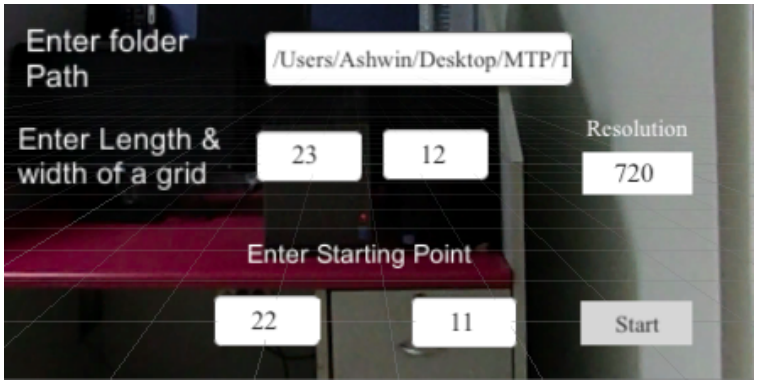

Figure 3.14 Intial window with the details

Figure 3.15 Equirectangular image

Figure 3.16 Created Virtual tour

We have implemented this on the Alienware Aurora R7 (IntelCore i7 8th gen, 2666 MHz DDR4 memory 16 GB, HDD 2TB, NVIDIA GeForce GTX 1080 8GB GDDR5X, Windows

10 OS). The overall time required to create the tour is **5 mins and 14 seconds** for creating a tour with 88 images and 720p cube dimension. The panaromic equirectangular image of the starting point and its equivalent scene in the image is shown in the Figure 3.15 and Figure 3.16.

After creation of the tour, to demonstrate, we can use either HMD or directly interact with the mouse from the screen. Now for this demonstration, on Windows OS, we have used WindowsSpeechRecognizer to navigate inside the environment with the commands as Left, Right, Forward or Backwards. If the Speech recogniser fails to recognise the command at particular moment, there there will always have an option to look downwards in the 25-80 degree of angles and after 2 seconds it will move forward in the direction an user is looking at. This is how navigation works inside the VE using our developed module.

3.7.2 Basic Anatomy of the script C# script used in the module

A C# script makes its internal connection to the unity's external rendered using build in class called as **MonoBehaviour**. This class is a gateway to attach any component created in the script to the GameObject. The name of the class can be anything, but it is convinent to use camel case string. Name of the class and name of the file should be matched.

```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Figure 3.17 Vanilla C# Script for Unity

In a vanilla C# script, there are two predefined functions available, one is **Start** and

another one is **Update**. For dynamic contents, Update function is used. Update function runs the code inside it on updating each new frame, hence movements, camera actions, response to the user often put in this function. It can work with all the GameObjects attached inside the function. Before any update done, there is a starting point of a code where initialisation of every component can be done, that function is Start.

3.7.3 C# script for developed module

The overall script has many functions, and methods embedded in the main Monobehaviour class to run the actual module, below is a brief explanation about important code snippets of the script.

- **variable Initialisation** In the Monobehaviour class itself, there are almost every variable initialisation has done. the objective here is to make all these variables global so that, they can be accessible within any method.

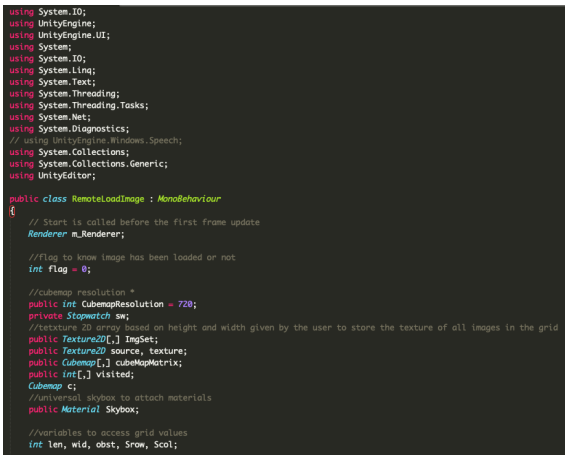

Figure 3.18 Variable Initialisation 1

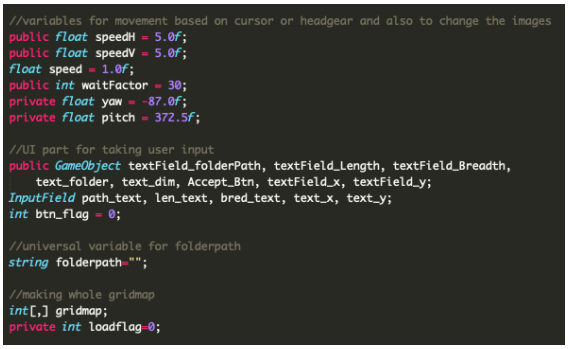

Figure 3.19 Variable Initialisation 2

In the above images, initially there is need to import required Libraries of Unity and C#. Then in the Monobehaviour class, there are 2D arrays to store textures, Cubemaps of the whole grid. There are variables to save grid dimensions, starting point and

information about obstacles. In the second images, there are variables such as speed, yaw, pitch to help for camera movements. The GameObject components are helpful to typecast into required UI components to take user's input.

- **Start Function and other relevant code snippets** The start function is just a starting part of the phase. After taking relevant inputs, Start function keeps waiting for a button click to proceed towards the creation of VR tour.

```
void Start()
{


    Screen.SetResolution(900, 506, true);
    Button btn = Accept_Btn.GetComponent<Button>();
    btn.onClick.AddListener(TaskOnClick);

}
```

Figure 3.20 Start Function

After getting instructions from the Start function, the onclick function starts its working according to the Figure 3.21. First it extracts all the information given by user into the input box on the initial screen. It extracts path of the images, grid size, starting point, resolution and converts it into string or Int form to assign to the global variables declared initially. Also from the naming convention given in the folder, it tries to resolves which images are present and which are not inside the overall grid. accordingly it creates gridmap matrix with 0 and 1.

- **Update Function**

Update function has a crucial role in this script. As written in the Figure 3.23, it continuously tries to detect HMD movement or the inputs from the gyroscope and tries to change arrows and direction of the user dynamically as explained in the above subsection. It takes y axis based angles and x axis based angles and changes user's movement accordingly. In depth explanation about rages of these angles can be found in the previous section.

```
public void TaskOnClick()
{
    //input taking for folder path and grid len and wid
    path_text = textField_folderPath.GetComponent<InputField>();
    folderpath = path_text.text;
    len_text = textField_Length.GetComponent<InputField>();
    len = int.Parse(len_text.text);
    bred_text = textField_Breadth.GetComponent<InputField>();
    wid = int.Parse(bred_text.text);

    //input taking for obstacles
    text_x = textField_x.GetComponent<InputField>();
    row = int.Parse(text_x.text);

    text_y = textField_y.GetComponent<InputField>();
    col = int.Parse(text_y.text);


    //print(len+ " "+wid);

    gridmap = new int[len, wid];

    for (int i = 0; i < len; i++)
    {
        for (int j = 0; j < wid; j++)
        {
            gridmap[i,j] = 0;
        }
    }

    DirectoryInfo d = new DirectoryInfo(folderpath);//Assuming Test is your Folder
    FileInfo[] Files = d.GetFiles("*.jpg"); //Getting Text files
    FileInfo[] TextFile = d.GetFiles("*.txt");
    //string str = "";
    string[] sep = { "_", "_", ".jpg" };
    int count = 4;
    foreach (FileInfo file in Files)
    {
        string name = file.Name;
        //print(name[4] + " " + name[6]);
        String[] strlist = name.Split(sep, count,
            StringSplitOptions.RemoveEmptyEntries);
```

Figure 3.21 On-Click Function

```
void Update()
{

    //mouse control code in
    yaw += speedH * Input.GetAxis("Mouse X");
    pitch -= speedV * Input.GetAxis("Mouse Y");
    transform.eulerAngles = new Vector3(pitch, yaw, 0.0f);

    //getting angle of view
    float angle = Camera.main.transform.eulerAngles.y;

    if(angle >= 325 || angle <25)
    {
        displayArrowFront(row, col);
    }
    else if(angle >= 225 && angle < 325)
    {
        displayArrowLeft(row, col);
    }
    else if(angle >25 && angle <=140)
    {
        displayArrowRight(row, col);
    }
    else
        displayArrowDown(row, col);

    print(vrCam.eulerAngles.x);
```

Figure 3.22 Update Function

```
if ( Input.GetKeyDown(KeyCode.UpArrow) || (vrCam.eulerAngles.x >= 25.0f && vrCam.eulerAngles.x < 79.0f))
{


    //front
    if(angle >=325 || angle <25)
    {
```

Figure 3.23 Condition check for movement

As per the Fig 3.19, initially the code detects if, an user tries to move forward or not based on the x axis angles. While watching downwards we have fixed the angle range as (25-80 degrees), and after triggering this condition, it checks whether to go Left, Right, Forward or Backward according to the y axis angles.

- **Functions to create Cubemap**

  This part runs before the start of the tour. All these function creates Cubemap for each available image in the matrix. Figure 3.24 has a function to create 2D texture of each equirectangular image from the folder. Then Figure 3.25 and 3.26 generates polar coordinates for each pixel and applies on the appropritae side of the pixel respectively.

```
//function to extract the texture from filepath of an image
public static Texture2D LoadPNG(string filePath)
{

    Texture2D tex = null;
    byte[] fileData;

    if (File.Exists(filePath))
    {
        fileData = File.ReadAllBytes(filePath);
        tex = new Texture2D(10, 10, TextureFormat.RGB24, false);
        tex.LoadImage(fileData);
    }
    return tex;
}
```

Figure 3.24 Generate texture form an image

```
Color Project(Vector3 direction)
{
    float theta = Mathf.Atan2(direction.z, direction.x) + Mathf.PI / 180.0f;
    float phi = Mathf.Acos(direction.y);

    int texelX = (int)(((theta / Mathf.PI) * 0.5f + 0.5f) * source.width);
    if (texelX < 0) texelX = 0;
    if (texelX >= source.width) texelX = source.width - 1;
    int texelY = (int)((phi / Mathf.PI) * source.height);
    if (texelY < 0) texelY = 0;
    if (texelY >= source.height) texelY = source.height - 1;

    return source.GetPixel(texelX, source.height - texelY - 1);
}
```

Figure 3.25 Cubemap Creation 1

```
Color[] CreateCubemapTexture(int resolution, CubemapFace face)
{
    texture = new Texture2D(resolution, resolution, TextureFormat.RGB24, false);

    Vector3 texelX_Step = (faces[(int)face][1] - faces[(int)face][0]) / resolution;
    Vector3 texelY_Step = (faces[(int)face][3] - faces[(int)face][2]) / resolution;
    //Vector3 texel2_Step = (faces[(int)face][5] - faces[(int)face][4]) / resolution;

    float texelSize = 1.0f / resolution;
    float texelIndex = 0.0f;

    // Create textured face
    Color[] cols = new Color[CubemapResolution];
    for (int y = 0; y < resolution; y++)
    {
        Vector3 texelX = faces[(int)face][0];
        Vector3 texelY = faces[(int)face][2];
        //Vector3 texel2 = faces[(int)face][4];
        for (int x = 0; x < resolution; x++)
        {
            cols[x] = Project(Vector3.Lerp(texelX, texelY, texelIndex).normalized);
            texelX += texelX_Step;
            texelY += texelY_Step;
        }
        texture.SetPixels(0, y, resolution, 1, cols);
        texelIndex += texelSize;
    }
    // texture.wrapMode = TextureWrapMode.Clamp;
    texture.Apply(false);

    Color[] colors = texture.GetPixels();
    return colors;
}
```

Figure 3.26 Cubemap Creation 2

Working of these functions already explained in the section 3.5

3.7.4 Module Workflow

Figure 3.27 and Figure 3.28 gives idea about overall workflow of the module with the help of Sequence diagram and Class diagram respectively.

Initially user needs to open the application generated from Unity. Then by giving HMD access, the welcome screen asks user to provide details such as folder path, grid size, resolution, etc. After giving all the required credentials, as explained above our module creates the Cubemaps for all the available grid positions. And display the starting point Cubemap into the skybox. Afterwards based on navigation input module displays required Cubemap in the skybox.

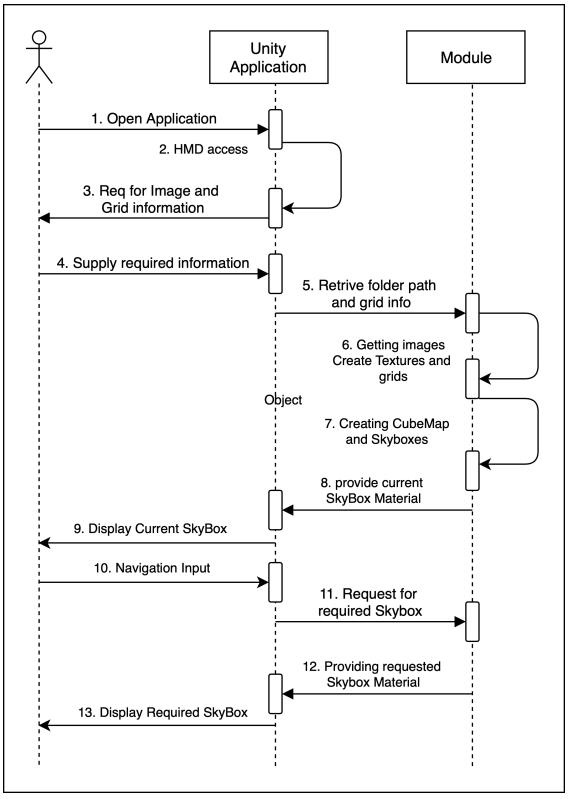

Figure 3.27 Sequence Diagram

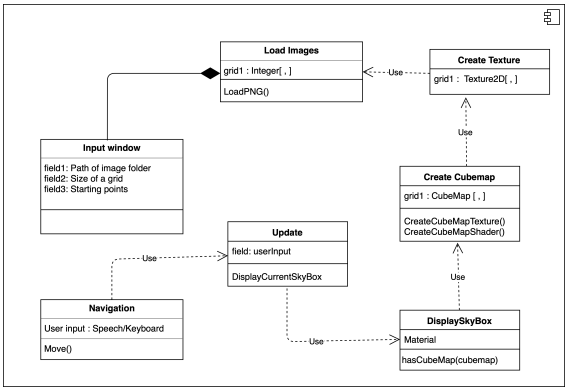

Figure 3.28 Class Diagram

Chapter Four

Results and Observations

This section is a crucial part of the thesis. Our observations shed light on the performance analysis of many devices on which we have tested our application. We have tried to maintain a fair environment while taking these readings, whether on desktops or mobile devices. Our work consists of two parts; one is the Development of a module, which creates a Virtual tour for us, and the second one is Running the application on the mobile devices to get the performance insights. Before directly going to the results and graphs in each part, first, let's go through the summary of our work done, and then we will discuss the observations.

4.1 Observations on Computer based developed module to create a VE

Initially, in our developed module, users need to give the folder path of panoramic images exported from VUZE or any 360-degree camera, which is in the form of equirectangular. Then the user needs to put grid size, and in the folder itself, the user needs to create a file that contains information about obstacles if requires. Then after putting the starting point, the user has to press the start button. Now hereafter, the process of converting equirectangular images into a cubemap and putting them into the appropriate 2D array, so that after reaching to any position VE can show a particular cubemap by applying it onto the skybox. The

process of converting an equirectangular image to a cubemap is explained in the Proposed Work section. In this process, based on the number of images and resolution of cube time require to generate all cubemaps varies. We need to put a light on these observations on different machines so that we can say a word about the importance of high-end hardware in the process.

For this experiment, we have decided to go with two different machines with different OS and architecture.

1. **System 1:** Alienware Aurora R7 (IntelCore i7 8th gen, 2666 MHz DDR4 memory 16 GB, HDD 2TB, NVIDIA GeForce GTX 1080 8GB GDDR5X,Windows 10 OS)

2. **System 2:** MacBook Air Early 2015 (1.6GHz Intel Core i5, 1600 MHz DDR3 memory 4 GB, SSD 128 GB, Intel HD Graphics 6000 1536 MB, macOSCatalina)

VUZE 360-degree camera clicks images by default with dimensions as 3840x2160; this is a 4K image that is very sharp by default. It might not require to us in every case while developing the module. Still, we have observed the results on this resolution. We have scaled down the image dimensions as 1920x1080, which is more suitable in terms of clarity and size. Each image with dimensions as 3840x2160 has size around 2.5-3 MBs, and each image with dimensions as 1920x1080 has a size in the range 1.3-1.8 MB. Now on the above mentioned two systems, we have taken observations to create a VE after providing details by varying the number of images and resolution of the cube. We have taken discrete observations on the number of images as 100, 50, and 10. With each of these 3 number of images, we are generating a 256p, 512p, 720p, 1080p cube dimensions. Here 256p cube dimension means a cube size is 256x256x256 (length, width, height), and the texture imposing on it has dimension as 256x256, because it is just a wrapping from outside like a wrapping with a paper on a gift box. When such a large 4K image gets imposed on the cube with size as 256p, it automatically gets scaled-down, with our algorithm, and automatically the quality of the generated cubemap gets decreased.

Number of images	Resolution	time
100	256 px	69 secs
100	512 px	195 secs
100	720 px	371 secs
100	1080 px	730 secs
50	256 px	41 secs
50	512 px	101 secs
50	720 px	220 secs
50	1080 px	405 secs
10	256 px	7 secs
10	512 px	20 secs
10	720 px	39 secs
10	1080 px	61 secs

Table 4.1 Observations on System 1 with image size 3840 x 2160

Number of images	Resolution	time
100	256 px	204 secs
100	512 px	531 secs
100	720 px	-
100	1080 px	-
50	256 px	113 sec
50	512 px	288 sec
50	720 px	552 sec
50	1080 px	-
10	256 px	12 secs
10	512 px	28 secs
10	720 px	50 secs
10	1080 px	104 secs

Table 4.2 Observations on System 2 with image size 3840 x 2160

Number of images	Resolution	time
100	256 px	40 secs
100	512 px	162 secs
100	720 px	344 secs
100	1080 px	698 secs
50	256 px	21 secs
50	512 px	71 secs
50	720 px	178 secs
50	1080 px	374 secs
10	256 px	5 secs
10	512 px	17 secs
10	720 px	33 secs
10	1080 px	52 secs

Table 4.3 Observations on System 1 with image size 1920 x 1080

Number of images	Resolution	time
100	256 px	176 secs
100	512 px	418 secs
100	720 px	845 secs
100	1080 px	-
50	256 px	88 sec
50	512 px	201 sec
50	720 px	467 sec
50	1080 px	-
10	256 px	6 secs
10	512 px	23 secs
10	720 px	52 secs
10	1080 px	95 secs

Table 4.4 Observations on System 2 with image size 1920 x 1080

(**Note:** The hyphens in some columns suggests that system was non-responsive)

From the values captured in the Table 4.1 - 4.4, one main thing we need to consider is, all of these observations are discrete hence, if we have observations for 50 images, then we cannot conclude on 49 images or 51 images and so on. The basic thing all of these tables suggests that System 2 suffers a lot due to it's lower configurations. There is a vast rendering time difference between both the systems. The main stage while creating cubempas is, mapping of each 2D pixel from an equirectangular image to a cube having 3D polar co-ordinates. For an image having resolution as 3840x2260 has 86,78,400 pixels. Now to map them on a cube which has 256x256x256 dimension and a texture has 256x256 dimension. Hence, the overall pixel count need to scale down to 65,536. Now this can be done by aggregating the neighbouring pixels into one. Since this scale down count is very less, converting in a 256p requires lot less time and computation power, because instead of converting each pixel it need to first aggregate them and then convert only 65,536 pixels. Similarly for 512p dimension we need to convert 86,78,400 pixels to 2,62,144 pixels and for 720p and 1080p we need to map 5,18,400 and 11,66,400 respectively. Similarly with the dimensions as 1920x1080, system need to map 20,73,600 pixels to 65,536 for 256p dimension, 2,62,144 for 512p, 5,18,400 for 720p and 11,66,400 for 1080p. Since the System 1 has Intel core i7 generation CPU and has normal clock speed as 1.8GHz means it can perform 1.8 billion operations per second compared to 1.6 billion of i5 for System 2. This doesn't make much difference right? Here comes Hyperthreading factor, i5 processors doens't have Hyperthreading means they cannot share threads among there cores, while i7 has hyperthreading option enables hence it can perform operations with same number of cores but by sharing among them. Hence Load among all got shared and work done gets quicker. The RAM factor also comes here, Since System 1 has higher RAM, it can hold more number of active processes inside the memory and in case of non-responsiveness of System 2, it cannot handle that much workload in such a short memory. Although MacBooks has a concepts of Virtual memory where it can share memory load with the Hard disk present on the computer but the SDD hard disk is not that

much fast as RAM hence Macbook faced a problems when converting large number of pixels and also it took much more time compared to System 1.

4.2 Mobile based rendering and performance checking

This section will discuss parameters of mobile-based rendering, how image size, number of images, and Cubemap resolution affects performance and how to measure mobile devices. In the Observations section, we have produced graphical data and an in-depth analysis of mobile performance.

This section includes all processes after generating an Android deployable file from a computer. How different image resolutions, number of images, and Cubemap resolutions affect the size of the application, and ultimately, the performance of the application on a particular device is a fundamental aspect of this section. The default 360-degree panoramic image resolution taken by Oculus is 3840x2160and default Cubemap resolution in the Unity application is 2048 pixels. Image resolution comprises the height and width of the 2D equirectangular image in terms of pixels. Cubemap resolution means the length, height, and width of a cube while considering it in 3D space. The initial thing we have done while exporting an image from VUZE studio is decreasing image resolution from 3840x2160 to 1920x1080. As we don't need an unnecessary 4K image based tour because that only takes more space and time while rendering, and also it takes much memory while generating Cubemap. Figure 4.1 gives an idea about resolutions.

Unity, by default, creates a cube for a Ceubemap with resolution as 2048p; this means the size of the cube is 2048x2048x2048, and on top of it, the texture of the panoramic image is applied to make a Cubemap, the whole process is explained earlier. The good thing about this higher sized cube is the very sharpened and crisp quality of a scene, but the bad thing is it takes too much size, and it requires a higher amount of time to render. And for a large virtual environment, we need too many cubes to hold the 360-degree panoramas, in that

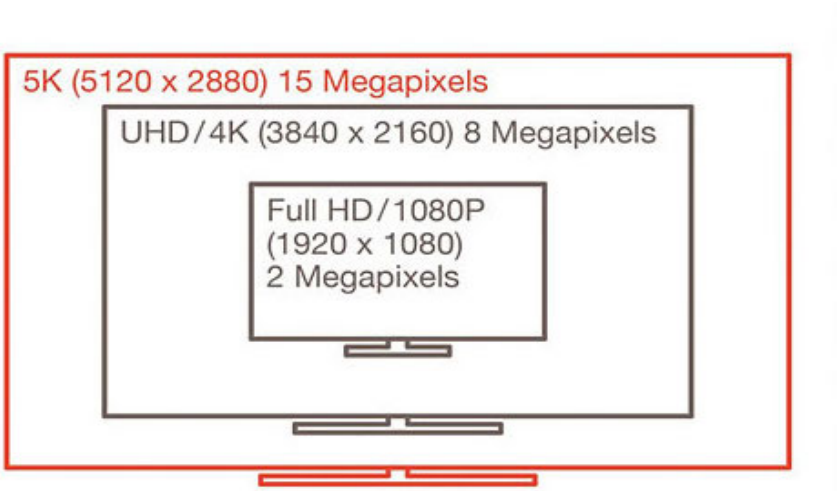

Figure 4.1 Resolutions

case, the overall application size increases drastically. Table 4.5 gives an idea about how varying Cube resolutions and the number of images creates an impact on application size. All the sizes mentioned here are in MBs.

Above table has listings of application sizes while varying resolution of a cube and number of images. The justification about 88 images is, our application consists sub-tours of many different parts of Majuli islands, among all the bigger sub-tour has highest 88 images, hence we have checked for a one sub-tour so that similar readings can be replicated while adding more images and application size will get increase accordingly.

4.2.1 Necessity of choosing cube dimensions as 512p

The exported image from VUZE 360-degree camera has resolutions like 3840 x 2260, which is in the 4K format. Which has higher pixel density as well, but for mobile or even for our computer screens this is a very higher sized image and to create Cubemap from this image while generating Virtual environment requires much more time and space as well; hence we have decided to scale down the resolution of Cubemap so that we can save space in the final app size. In Table 4.5, we can see that while choosing resolution as 2048 and the number of

		Cubemap resolution			
		256	512	1024	2048
Number of Images	10	15 MB	27 MB	56 MB	176 MB
	20	18 MB	34 MB	72 MB	202 MB
	30	26 MB	48 MB	90 MB	282 MB
	40	31 MB	54 MB	111 MB	326 MB
	50	35 MB	68 MB	136 MB	409 MB
	60	41 MB	78 MB	157 MB	492MB
	70	49 MB	86 MB	189 MB	601 MB
	80	54 MB	93 MB	203 MB	708 MB
	88	58 MB	97 MB	232 MB	790 MB

Table 4.5 App sizes with varying resolution and number of images

images 88, the application size has increased to 790 - 800 MB, which is too much for a naive user. Also, in our application, there are many different areas of Majuli, which consist of many such sub-tours, and you can imagine by adding all those, application size get increase up to 1 or 1.3 GB, which is not a good thing while scaling down application across many users. Hence we have checked performance by scaling down to 1024, 512, and 256 resolutions.

PPI: The term PPI stands for pixels per inch. As all our devices have screen sizes measured in inches, PPI is essential while having insight about image quality or video quality on a particular device. By reducing the distance between pixels means increasing the size of a cube without changing screen size increase PPI. More straightforwardly, by keeping screen size constant, 2048p has higher PPI than 1024p and 512p. for example, if a particular application or screen has 100 PPI, there are 100 pixels per 2.54 centimeters, and for 400

PPI, there are 400 pixels placed within the same width. [12]

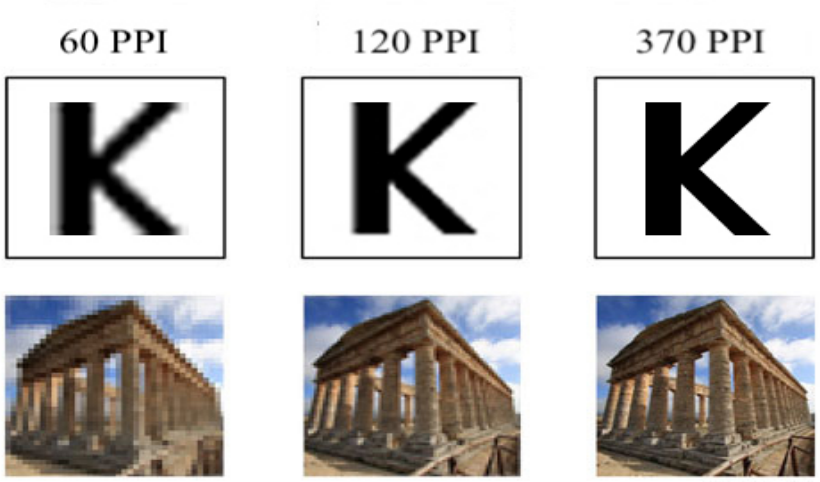

Figure 4.2 PPI Comparison

PPI is an essential aspect of creating a difference in appearances. In the Figure 4.2, first image has 60 pixels per inch; hence, it is a blurred one compared to others, while in the second image with 120 PPI, the image is convincible, whereas the 370 PPI image is super crisp. Nowadays, the mobile size has on an average 5.5 to 5.8 inches diagonally [30]. In our case with 2048 pixels of Cube dimensions, and 5.5 inches average mobile screen size, we can get the crisp 372 PPI virtual tour. But there is a drawback of space and time as well. In the observation section, we have provided more studies on this about how a device can cope up with these applications. Even with the 1024p of cube dimensions, we can get PPI as 190, which is the perfect midway, but some devices are still unable to run the application. The safer PPI range for running an application on mobile devices is 100-300 PPi, so if we are going with the least 100 PPI by fixing cube dimensions as 512p, we can get the satisfiable amount of VR tour as you can see below.

Application with cube dimension as 512p runs on almost every device, and by adding images up to 200, we can still get application size around 230-260 MB, which is more convenient for scalability. While developing on the MacBook Air, we have observed that it

Figure 4.3 Running VR tour in the application

was difficult for a system to develop an application with a higher number of images and 1024/2048p Cube dimensions. In the later part, we have all the observations traced to 512p based applications running on different mobile devices.

4.2.2 Information about mobile devices used in this experiment

As described in the Introduction section, there are two types of availability of devices, while mobile-based testing. One is testing on actual devices, and the second option is testing on emulators. Emulator based testing requires SDK download and many other processes. Generally, developers choose based on requirements. We have decided to go with 4 actual devices, which vary in terms of CPU, GPU chips, RAM availability, and many different architectural terms.

1. Xiaomi Redmi Note 5 Pro
2. Samsung Galaxy S8+
3. Asus Zenfone max M2
4. Xiaomi Redmi Note 3

These devices has following configuration:

Device Name	Processor	RAM	GPU
Xiaomi Redmi Note 5 pro	1.8GHz Qualcomm Snapdragon 636 octa-core	4 GB	Adreno 509 GPU
Samsung S8 plus	2.3GHz + 1.7GHz Exynos 8895 octa-core processor	4 GB	Mali-G71 MP20 GPU
Asus Zenfone Max M2	1.8 GHz Qualcomm Snapdragon 632 Octa-Core Processor	3 GB	Mali-G71 MP20 GPU
Xiomi Redmi Note 3	1.4 and 1.8 GHz Qualcomm Snapdragon 650 Hexa-coreProcessor	2 GB	Adreno 510/509GPU

Table 4.6 Mobile devices used for performance checking

As we can see in the table 4.6, all these devices has different configurations and that helps to get more diverse information about application performance. We have 4 different CPU architectures available. Out of 4 different processors, 3 are Qualcomm Snapdragon based and 1 is from Samsung Exynos.

	Qualcomm Snapdragon 636	Exynos 8895	Qualcomm Snapdragon 632	Qualcomm Snapdragon 650
Devices	Redmi Note 5 Pro	Samsung Galaxy S8 Plus	Asus ZenfoneMax M2	Redmi Note 3
Semiconductor size	14nm	10nm	14nm	28nm
Direct X version	12	12	12	11
CPU speed	4 x 1.8GHz & 4 x 1.6GHz	4 x 2.3GHz & 4 x 1.7GHz	4 x 1.8GHz & 4 x 1.6GHz	4x1.4 GHz &2x1.8 GHz
Cores	8	8	8	6
RAM Speed	1333MHz	1866MHz	933MHz	933MHz
Single-core Benchmark score	1330	2015	1236	1458
Multi-core Benchmark score	4943	6711	4515	2869

Table 4.7 Comparison among mobile processors - CPU

Explanation about some concepts used in Table 4.7 [23]

1. **Semiconductors:** Semiconductors which has small size, are better performers. Because smaller chipsets requires lesser power and allow to fit more transistors on a chip. Smaller semiconductors allow to operate more tasks at lesser temperatures.

2. **DirectX:** DirectX is an API to handle multimedia tasks, gaming performance and many graphical based tasks. The latest version has better graphical supporting facility.

3. **CPU Speed:** CPU speed indicates how many processing cycles can be executed per second by the processor. Generally this can measured in Giga Heartz, which is thousand million cycles per second. Simply these number shows how quickly CPU can

process the data. All these CPUs are having ARM architecture which can correctly distribute load among all cores. Based on how demanding a task is, this CPU can assign it accordingly on a particular core.

4. **RAM Speed:** Faster RAM speed has more advantage while swapping in and out processes, cache maintenance. Higher RAM speed increases overall system performance.

5. **Benchmark Scores:** benchmark scores are calculated based on overall performance of CPU, GPU, network, etc. Single core score reflects performance when there is no sharing of processes between two or more cores. While multi-core performance is based on sharing of a process among more than one core. Here Samsung Exynos outperformed in single as well as multicore testing. Snapdragon 650 doesn't support threading in multi-core hence its score is very less in it.

Based on the above parameters and global Benchmark ranking Samsung Exynos 8895 processor is a better performer among all the 4 processors, and that should reflect in the observations as well. We will go in depth observations on CPU, GPU and RAM usage and management in the next section.

Among all 4 available devices, we are having 2 different kinds of GPUs available:

1. **Qualcomm Adreno 509** (Xiaomi Redmi Note 5 Pro, Xiaomi Redmi Note 3)

2. **ARM Mali-G71** (Samsung S8+, Asus Zenfone Max M2)

Above table shows, how both GPU stands in front of each other. Qualcomm Adreno 509 is used for mid-range moble devices, while ARM Mali G-71 GPU used in may higher end devices as well. One reason behind this is support of FMA4. It is used to change contrast of an image or a video according to the need of an application, which saves pixel based processing load of a GPU, and this technology increases score of the ARM Mali G-71 GPU. There is very slight different in the versions of DirectX, that won't matter much. OpenGL and OpenCL libraries helps while rendering any graphical content.

	Qualcomm Adreno 509	ARM Mali G-71
Devices	Redmi Note 5 Pro, Redmi Note 3	Samsung Galaxy S8+, Asus ZenfoneMax M2
Semiconductor size	14nm	16nm
DirectX version	11.2	12.1
OPENCL version	2	2
OPENGL version	3.2	3.2
FMA4 Support	No	Yes

Table 4.8 Comparison among mobile processors - GPU

4.3 Methodology to observe profiling information on Mobile devices

This section introduces one of the available methods to observe profiling information. There are many available profiling tools such as Android Studio Profile, Dalvik Debugger, or a Native Profiler. The Android Studio profiler works within the Android Studio application, which has its benefits and limitations. The main benefit is, it gives us graphically rich information so that we can quickly get average value or comparison based insights. The main limitation is that it works in a stipulated manner; there is not much room to get new insight other than their fixed parameters. Dalvik Debug Monitor is a Java application that works irrespective of any IDE. DDM is mainly used to get LogCat information of any application which can display Log messages or any warnings. It also gives an idea about Heap usage, Stack but not exact Memory, CPU, or GPU usage breakdown. It is widely used to simulate networks to check speed and Latency. The Native debugger on any android phone is not heavily loaded with information. Instead, it is used to get upper-level insights about memory usage or CPU usage in terms of cores. It shows only graphs on the mobile screen without giving exact values. It is useful only when there is no availability of desktops. These three services have their advantages and disadvantages, but we need to work in their

prescribed way. Hence, we've decided to use the Android Debug Bridge.

4.3.1 Android Debug Bridge

Android Debug Bridge (ADB) is a utility which works with the command line and let user communicate with the connected device. ADB utility allows the user to install or uninstall the application, check device logs, access the Unix shell inside the device, get application status or usage, and many more such facilities, this is why ADB is unique than the above-discussed tools. The communication between a desktop and a mobile device is working as a client-server application. There are 3 parts in this client-server program: [3]

1. **Client:** This is the desktop end of a whole process which sends commands and wait for a response. The client can be invoked on the command line using "adb" command
2. **Daemon:** A daemon is responsible for executing commands sent by a desktop on the connected mobile device. It works in the background on a mobile device.
3. **Server:** This part manages communication between Client desktop and daemon. The server runs on a client-side.

Working of ABD:

ADB comes by default with every SDK version of Android. By locating the SDK folder of any compatible version, we can configure ADB globally on the machine. Initial Steps follow by ADB:

- After starting ADB, it checks whether any server is running already; if not, then it starts a new ADB server by binding TCP port 5037 to communicate with the daemons.
- There can be two ways of connecting devices to ADB clients. One is direct via USB, and another is connecting over the common Wi-Fi between desktop and a mobile device.

- In any case, the USB debugging option should be enabled on mobile devices.
- After setting up a server, ADB tries to locate devices by scanning ports in the range of 5555 to 5585. at max 16 devices can be connected at a time
- The connected devices can be seen on a command-line interface as: Emulator 1, console: 5556.
- There are many commands we can put upon a console to do different activities on the device.

Some sample set of commands on ADB:

1. **adb devices:** lists all the connected devices
2. **adb start-server:** starts adb server
3. **adb kill-server:** kills adb server
4. **adb shell:** enabling background terminal on daemon (there are multiple commands in adb shell as well)
5. **adb shell getprop ro.build.version.release:** Android device version
6. **adb logcat:** logcat of the connected daemon
7. **adb -e install path/to/app.apk:** Installing any application from a desktop
8. **adb shell list packages:** all installed packages on a mobile device

There are too many other adb shell commands available, here we have shown a glimpse of them, in the next part let's understand how we have gathered observations:

4.3.2 Getting observations using ADB

For this experiments we are focusing only on three parameters:

1. CPU Usage
2. GPU Usage
3. Memory Usage

To get these readings using ADB we have used following commands and results are attached while connected to one device:

- To get list of all connected devices we have used **adb devices -l** command. The result we have got as follows while connected to one device is shown in the Figure 4.4 Here

```
Ashwin@Ashwins-Mac ~ % adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
3628fa06        device
```

Figure 4.4 Connected Devices to ADB

  we can see one connected device which is Redmi Note 5 Pro, and ADb shows device id as: 3628fa06.

- To get live process status running inside device, we have used command **adb shell top**, with use of its options we can get live running process status with many parameters as shown in the Figure 4.5 Here we have used adb shell top command with 2 parameters as "adb shell top -m 5 -d 1" which defines m [n] = n number of processes which are using maximum CPU and d[n] = Requires updates at each n seconds. Hence here the results are showing top 5 processes which consuming CPU atmost and it is updating at each 1 second. The parameters shown in front of each process is as follows:

  1. **PID:** process Id

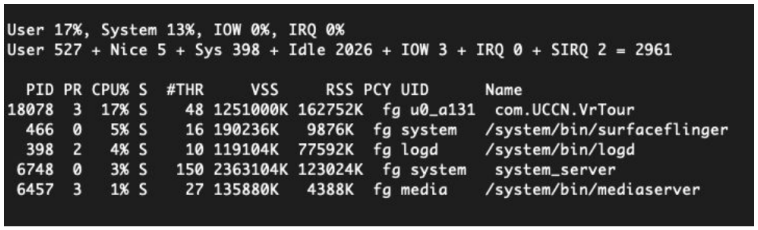

Figure 4.5 Running processes

2. **PR:** priority for the process, given by the mobile system
3. **CPU%:** percentage use of CPU by that process
4. #**THR:** Number of threads involved in that process
5. **VSS:** Total accessible address space (Memory) for the process, It's given according to available space, priority and type of the process
6. **RSS:** Actual memory held in the RAM by the process.RSS is giving actual memory usage by the process.
7. **Name:** Package name of the process/Application

- To get GPU usage, we have to put command as **adb shell dumpsys gfxinfo** on the server side terminal and we can get Frame usage, how frames get rendered on the scree, and all UI performance data. It updates information every second.

Using the above commands and methodology, we have got our required observations. While gathering GPU performance data, the Native android profiler helped us get more insights about GPU usage. To capture the performance measures, we had run the application for 1 minute on all of the four devices and tried to replicate all the movements, gestures on every device to make fair competition among them and noticed the values in every 6 seconds of interval up to 60 seconds.

In this overall section, we have tried to mention every minute detail to give the audience the right amount of idea about our Proposed work. The overall work done has many minor

tests involved; many times, we have gone on a different blocked path, and finally, we have put everything in a nutshell. Now in the next section, you can get to know about our Observations and Results.

4.4 Observations on Mobile devices:

These observations are independent of the above task. In this experiment, we have created cubemaps with the help of Unity and not with our developed module. The reason behind that is, we wanted to focus on the performance analysis of mobile devices, and it is easier to create an Android deployable file (.apk) when we create the tour by using all the resources of Unity. But it requires time and effort to generate cubemap for each provided image; our previously developed module is beneficial to get rid of these efforts of creating cubemaps.

Now coming to the main topic, as discussed in the proposed work, we have captured all the observations on 512p cube dimensions. In the proposed work section, we have discussed how 512p resolution is considerable in terms of PPI and the size of the application. To give more confidence to this argument, we have installed applications with varying sizes and cube dimensions on 3 mobile devices and observe how the application behaved. The 3 mobile devices used in this purpose are:

1. Xiaomi Redmi Note 5 Pro
2. Samsung S8+
3. Asus Zenfone Max M2

In the proposed work section, we have already discussed configurations of these devices, and with those on paper configurations, Samsung S8+ was a better performer. How a varying number of images and cube dimensions has an impact on the app size, we have already seen in the proposed work. Again to have that data handy, we have put here in the Table 4.9 below.

		Cubemap resolution			
		256	512	1024	2048
Number of Images	10	15 MB	27 MB	56 MB	176 MB
	20	18 MB	34 MB	72 MB	202 MB
	30	26 MB	48 MB	90 MB	282 MB
	40	31 MB	54 MB	111 MB	326 MB
	50	35 MB	68 MB	136 MB	409 MB
	60	41 MB	78 MB	157 MB	492MB
	70	49 MB	86 MB	189 MB	601 MB
	80	54 MB	93 MB	203 MB	708 MB
	88	58 MB	97 MB	232 MB	790 MB

Table 4.9 App sizes with varying resolution and number of images

There are 36 combinations of the number of images and cubemap resolution or cube dimensions. We have tried to install all 36 applications on every device, and the observed status is below:

1. **Xiaomi Redmi Note 5 Pro:** Till 40 images with all possible max cube sizes as 256, 512, 1024, 2048, all the applications are running correctly. The maximum size with 40 images and 2048 resolution is 326 MB. By default, android gives a warning for not-verified applications with size more than 250 MB. Now, for 50 images, with resolution as 256, 512, 1024 application is running perfectly fine, but for 2048 it wasn't responding. The application was installed but didn't open; the possible reason is the lack of available space in the RAM. From 60 to 88 images, the application wasn't responding at 1024p and 2048p resolution. This also gives confidence about the usability of 512p resolution.

2. **Samsung Galaxy S8+:** This device seems more potent than the other two in this test. According to paper readings and global benchmark testings, Samsung Galaxy S8+ has a more powerful processor in it. Up to 60 images, Samsung Galaxy S8+ runs perfectly fine, which means it can bear more images and resolution than Xiaomi Redmi Note 5 Pro. From 70 to 88, this device struggles to run applications with cubemap resolutions as 1024 and 2048. Hence for more images, it is clear that we should go with 512p resolution. Unity provides very few choices in terms of these cube dimensions: 256, 512, 1024, and 2048. In our developed module, we have provided the facility to put any required cubemap dimension or resolution that was another advantage.

3. **Asus Zenfone Max M2:** This device has the lowest configuration among all 3, and that reflects in the observations as well. This device performed almost similar to the Xiaomi Redmi Note 5 Pro with some less performance. Till 40 images with all resolutions, this device performed well. But when the number of images reaches to 50 images, it struggles with resolutions 1024 and 2048 and application unable to open. After that, with all combinations up to 88 images, this device was unable to run the applications with resolution 1024 and 2048.

In these studies, you may think that only app size matters, but that is not the case; phones can run even applications with sizes over a GB. But to render the VR application, there is a different pipeline, and there is much computational power required for higher quality. The above study helps to give a concrete base for using 512p as a default configuration for all further observations.

After finalizing to proceed with 512p cube dimensions, we are moving towards an important task, which is performance checking. As mentioned in the proposed work, we have captured performance information for 3 matrices:

1. CPU Usage

2. GPU Usage

3. Memory Usage

There are many more things that also affect the mobile device performance, but for a naive user, from the upper level, these 3 things are more than enough to analyze. We have got much more insights in just these 3 parameters.

The devices we are going to use for this experiment are:

1. Xiaomi Redmi Note 5 Pro

2. Samsung Galaxy S8+

3. Asus Zenfone Max M2

4. Xiaomi Redmi Note 3

Configurations and comparisons of these devices are mentioned in the proposed work, and now we will go through each device one by one and observe the readings taken by us for all 3 parameters. The method we have followed and the commands we have used in ADB are mentioned in the previous section.

Before procceding towards the observations lets understand some important things while rendering on the mobile devices.

4.4.1 VR rendering pipeline:

The game engine present in the mobile devices has two responsibilities:

1. Do physics simulation, from the inputs taken by the gyroscope or touch based inputs

2. Maintaining app state and sending final contents towards GPU for rendering

Game engine performs both of these tasks on the CPU, hence first CPU decides the work for GPU and then GPU takes the responsibility.

Draw calls:

The update function from C# script which gets converted into android runnable file gets updated at each frame and at that moment, game engine captures final image of the app and sends series of draw calls from it to the GPU which contains:

- Mesh Information
- Materials and associated properties
- Shader information required for materials

Game engine tries to send minimum required information to the GPU because for higher frame rates, it might get tougher for GPU to compute everything. Still some hidden objects from the screen gets rendered as well.

Actions performed by GPU on each draw calls:

GPU has to do calculations for each vertex based on the script attached to the material. Then pass these values to the shader to calculate color and interpolated position of each pixels. There are many more things GPU need to do to avoid access load on it such as Z-culling for transperent objects, understanding depth buffers, but in our application some of these features are unncessary.

The idea behind understanding this VR rendering pipeline is to get more insights from the CPU, GPU observations. By understanding the pipeline we might locate some limitations of CPU, GPU or memory usage.

4.4.2 CPU, GPU and Memory Usage readings for mobile devices:

We have taken readings for 1 minute with 6 seconds of intervals. We have tried to maintain the competition as fair as possible by running only system applications, quitting all the services and apps which use extra resources. While navigating in an environment, we have tried to move similarly and did head movement more similarly. In this Virtual tour, there are 3 sub-tours of Majuli islands named as North Kamalambari, Auinati, Jangraimukh. Each of these three has 88, 61, 51 images; hence we have a total of 200 images in a tour with app size as 183 MB.

- **Observations on Redmi Note 5 Pro:**

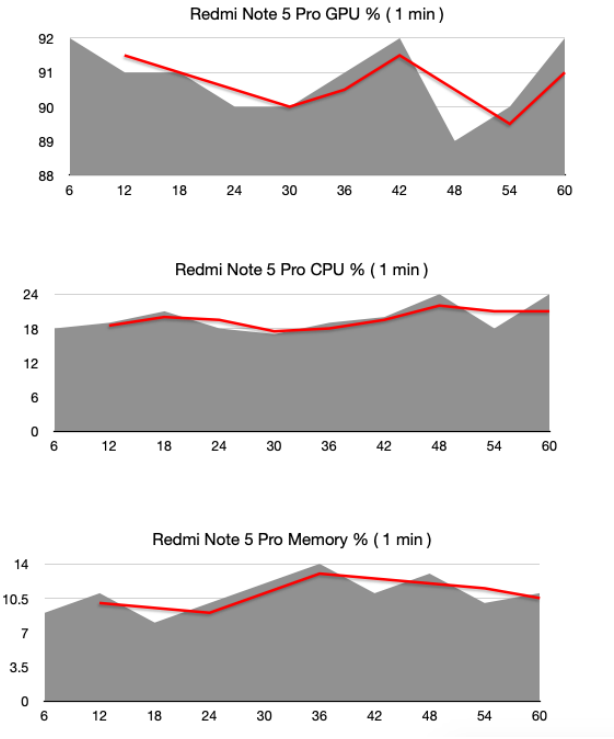

Figure 4.6 Redmi Note 5 Pro Observations

1. **CPU usage**: The CPU usage was not much because as explained above, CPU aligns tasks for GPU and all the load then transferred to the GPU and hence just

to run a VR application minimal of around 20% CPU usage was there, highest is 24%. Every time game engine on the CPU tries to work with minimal amount of resources and sahre the work with GPU.

2. **GPU usage**: On the Redmi Note 5 Pro device, we have seen Higher GPU usage which is a maximum of up to 92%. The Spikes in the GPU usage graphs show sudden movement in the camera. Otherwise while changing each scene it took near about 90% GPU. GPU requires to do many tasks for each new frames, and when we increase head movement on the phone, new pixels need to be generated at high speed, to increase this speed GPU puts more efforts.
3. **Memory usage**: The memory Usage of Redmi Note 5 Pro is around 520340 Kb which is around 12-13%, the minimum is 9% and the maximum is 14% out of 4 GB memory. The memory managemnt is good because game engines doesn't keep much information pending in the memory while sending to the GPU, also GPU uses its own dedicated memory in the RAM to put its cache and its processes. Hence the overall RAM usage was not much on this device.

- **Observations on Samsung Galaxy S8+:** This device is having a more powerful processor compared to others hence it has managed to keep the load as minimum as possible.

  1. **CPU usage**: The Qualcomm series got its power after 700 series but below that Exynos has outperformed over it. The application ran fluently on the device and also CPU wisely transferred its load on the GPU hence its usage was not much. It was somewhat similar to the Redmi Note 5 Pro device. We can easily recommend this processor while running our application. The device was cooler than other devices while running this application. It also managed the CPU load accordingly with maximum usage of only 14% and mostly around 11%. Exynos CPU is much more efficient than others, also it has more clock speed to put the commands in

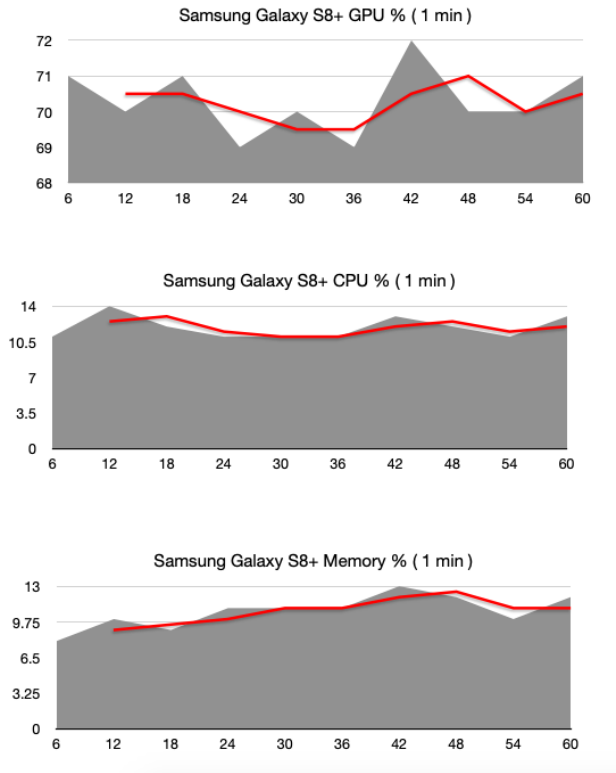

Figure 4.7 Samsung Galaxy S8+ Observations

line for GPU, and the shareable load across the cores helps this device to keep the CPU usage was quite low comapred to others.

2. **GPU Usage:** Samsung has ARM Mali-G71 MP20 GPU which is good enough for performance compared to Adreno 509 according to Benchmark scores. Also the support of the Exynos processor makes it a good combo, hence it clearly reflects in the results. The GPU usage was upto 72% only and maximum times is around 70%. The spikes are generated because of heavy camera movement like constant head shaking while viewing through cardboard. The draw call activity requires requires GPU but ARM Mali uses FMA4 support which helps to understand pixel depth more significaltly to the GPU and which helps to gice colors to pixel and it keeps load low. The major head movement refelects at 42nd seconds onwards which gives higher spikes in the graph.

3. **Memory:** Out of 4 GB RAM, The device has able to give much more space to run the application (VSS), It has provided almost a Gigabyte space to run the application, but still it has taken only up to 13% maximum and stayed near to 10% only and the minimum was 8%, which is very efficient. Out of availble space based on the priority and type of the application, this CPU and GPU has succeded to keep the overall memory load low, the usage of dedicated memory assigned for GPU might go high but that measurement we haven't checked.

- **Observations on Asus Xenfone Max M2:** This device has 1 GB lesser RAM than above two still because of Mali GPU it has performed well. It has also Octa-Core processor but less powerful than Exynos and very slightly different than Redmi Note 5 Pro.

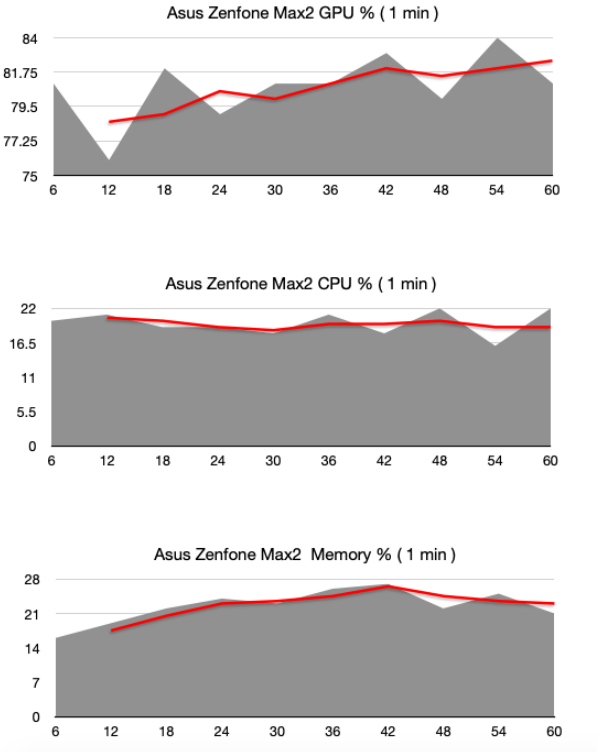

Figure 4.8 Asus Xenfone Max M2 Observations

1. **CPU usage**: CPU usage on this device seems normal as this is a comparatively

older processor Still it has managed its load quite nicely with lesser RAM. The maximum usage was 22% with an average is around 21%. Octa-core CPU helps to distributes overall load, although this device doesn't support hyperthreading, still by assigning higher proirity based task to that core which has higher clock speed is a good thing about this processor. CPU manages to keep the usage low by distributing its laod to the GPU accordingly. Throught the test CPU usage is quite closer to the average because it always try to keep app state updated and pass the required information to the GPU.

2. **GPU usage**: This phone has ARM Mali-G71 MP20 GPU which is good than Adreno 509 and that we can see in the graph as well. Although it has a less powerful processor than Redmi Note 5 Pro, still the GPU manages to utilize itself better. The maximum use of GPU was 84% which is good than Adreno 509. The overall performance was near about 80-82%. This GPU is more capable as discussed in pervious section, still because of the lesser CPU support compared to Samsung Galaxy S8+ this device has slight higher GPU Usage. The reason behind that is, because of less powered CPU, it lags while giving instructions, the lag is not visible to us, but we can see it in the performance, hence after getting instructions at lesser speed from the game engine, GPU workload increases.
3. **Memory usage**: The 3 GB RAM is not less for running our application. The percentage-wise load was quite higher because of overall lesser capacity but the device managed it fluently. While running only essential system applications and our application, the maximum load was 27% while keeping low at 16% at the start.

- **Observations on Redmi Note 3:** This is the oldest device among all still it has got upgraded processor compared to Asus Zenfone and Redmi Note 5 Pro. This device has the least RAM among all the devices.

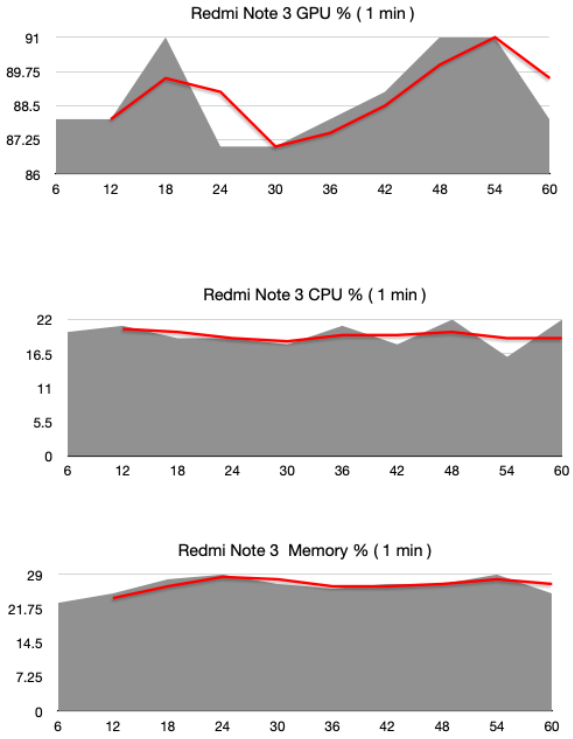

Figure 4.9 Redmi Note 3 Observations

1. **CPU usage:** Unexpectedly this phone has released earlier than Redmi Note 5 Pro, still it has a better processor but with fewer cores. Since hyperthreading is not allowed anyway the CPU load was not shared on different cores. Hence this device also performs similar to the Redmi Note 5 Pro. The Maximum CPU usage was 22% with a minimum of 16%.

2. **GPU usage:** This phone has Adreno 510 GPU which clearly less effective than the Mali chipset. Hence it signifies in the graph as well. The maximum load on the GPU was 91% but almost all the time it stays low around 87%. The spikes in the graph show sudden camera movement captured by the gyroscope. When we increase the movements, the draw calls need to be done much more faster per frame hence GPU load has increased and that effect might kept in the dedicated block for GPU in the cache memory for longer time hence from 43 sec, the spike

was constantly at the top.

3. **Memory usage:** This device has the least memory among all, still our application runs on it smoothly it's really good. Still, because of the better processor, the load was balanced on the memory and it took a maximum of 29% and around 26% on average.

After analyzing individual observations for each device, in cumulative observation, we can say that GPU usage was very high in the overall rendering process on any device. The main reason behind this is we can see in the rendering pipeline that CPU schedules the job for GPU, and almost everything is done by GPU. The Snapdragon processors, based on the available RAM, perform well enough in all these devices. All 3 Snapdragon-based devices have CPU usage is around 22%, which is still higher in terms of the tasks they are doing. There is nothing exhaustive for CPUs in this application. The Exynos processor on the Samsung Galaxy S8+ performs way better than all of the remaining three devices. It has a more capable CPU as well as supports hyper-threading, and a smaller 10nm chip helped to perform better. GPU has to do a tremendous amount of work compared to the CPU in this application. In all the above observations, we have seen a higher GPU being consumed throughout the application. The sudden spikes depict that sudden movement inside the environment. Because of the reliable, supportive processor in the Samsung S8+, the ARM Mali GPU works well compared to others. The same GPU has been used in the Asus Zenfone Max M2; still, it is way better in the S8+. Hence, the main thing to notice is that the CPU and GPU work together, and each other's performance is heavily dependent. In Snapdragon Adreno based devices, GPU usage was up to 91-92%, wherein ARM Mali-based devices, GPU usage was up to 82% in Asus Zenfone Max and up to 76% in Samsung S8+. The memory usage was lesser in the 4 GB RAMs, where it goes high in 3 and 2 GB RAMs, which is obvious. Still, this application uses an almost similar RAM amount in all devices, but the percentage goes higher because of the total RAM capacity. The maximum load has been

shared with the dedicated memory of GPU.

Hence, all these 3 parameters affected while using our application in all 4 devices. And the trend is almost constant throughout all these devices. Based on these observations, we might give some recommendation or some minimum limits to any user. The next section will help to understand what we can infer from all this data.

Chapter Five

Conclusion & Future Work

5.1 Conclusion

In this thesis, we have proposed a VR tour rendering technique using 360-degree images with the help of the Unity game engine. On top of that, this thesis explains about one of the possible navigation techniques and its parameters. Later this thesis put a focus on what are the performance statistics using different parameters after deploying the application on different mobile devices. The virtual reality is a vast and exponentially growing field. This thesis gives an idea about different spectra of VR as well as all types of VR, how VR can be useful in different fields in our day-to-day life, and how it can be improved. The 3I's of VR is a very crucial aspect while developing a VR application from the perspective of an end-user. The usage of 360-degree 2D images can provide real and interactive experience while doing a virtual tour. This thesis summarises different platforms available to create and render a virtual environment and what we have used. Different types of virtual reality and how that can be useful in different domains is well explained above. Some commercial systems, working applications in virtual reality as well as pros and cons of current systems have been mentioned in the previous part of this report. The commercial systems has a fixed way to create a VR tour, if we want to include obstacles or we want to create VR tours as per our required resolution, then it is not possible with them. This report gives information about

capturing a 360-degree image and the different technologies and gears available to capture and visualize these images. Our thesis gives an upper-level idea about some terminologies and concepts required to create VE in the Unity. Then we have explained the methodology behind our developed module. All the terms functions to create a cubemap based VR tour is explained there. The C# script based explanation is more helpful to get more insights. Below is the comparison about how our developed module is essential while creating a VR tour, and helps to decrease the efforts required to create a VR tour.

Steps required to create a VR tour unsing Unity:

1. Import Panaromic 360-degree images
2. Creation of new materials equal to the number of imported images
3. Changing each material shader to the Cubemap
4. Changing each image texture to the Cube
5. Set Cube dimension for each image
6. Conversion of each image to a layer of Cube
7. Apply each converted cube texture of the image to the respective Material.

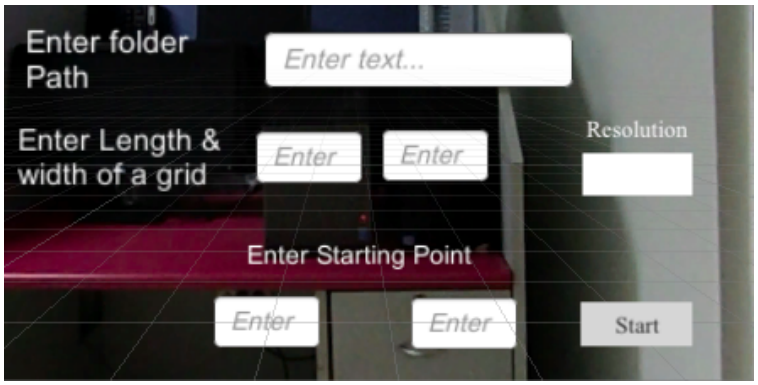

Figure 5.1 Initial window of the module

All of these above mentioned steps can be compromised using our module by just entering the path of the images, grid size, starting point, and required cube resolution as shown in the above Figure 5.1; this can be a more straightforward and minimalist approach to creating a VR tour. While concluding about this, our module can provide ease while developing a VR tour.

The critical study tells about the minimum system requirement for the rendering of the minimum number of images under a particular resolution. From that, we can conclude that either we can compromise the time or quality up to some extent. While explaining the Navigation part, this report gives an idea of grid-based navigation. Some preliminary survey tells about minimum step-size in the indoor or outdoor environment, which can help while capturing images by keeping the correct distance between them without failing to give subjective experience in the virtual tour. A further explanation tells the importance of Speech-based Navigation and its limitations as well.

In the later part of the thesis, we have explained different types of mobile device testing methods, performance parameters, and bench-marking tools. What was the need to choose Cube max dimension as 512p, and what's the effect on application size while using higher resolution and higher images were explained in an excellent way.We have used real device-based testing rather than simulation-based testing for performance checks. We have given in-depth information about devices that we have used and their conclusive study. Based on that, in conclusion, Exynos 8895 processor on the Samsung Galaxy S8+ turned out to be the best performer among all 4 devices. Also, ARM Mali G-71 MP20 GPU supports amazingly to its CPU processor, and hence it helps to improve the overall performance. The base for this argument is, the Redmi Note 5 Pro and Asus Zenfone Max M2 have almost similar processors but because of higher strength from ARM Mali G-71 GPU compared to Adreno 509 helped Asus Zenfone Max device to increase its performance. The CPU performance is best in Samsung Galaxy S8+ and worst in Redmi Note 5 Pro, which shows that a higher performance game engine on the CPU helps to send app state and to-do tasks to GPU

quicker. There was no lag on any device while running the application. GPU has done the main task for this application, and according to the results, we can easily conclude that ARM Mali G-71 chipset has a better performer than Adreno 509. The memory usage was throughout the same in all the devices but depended on the total capacity overall percentage varies. Based on the overall analysis, Snapdragon 650 and Adreno 509 were the lowest performers. Snapdragon 650 is almost 4.5 yr old CPU, still performed well while running the application. Hence, according to the global benchmark tests, CPU with 1458 as a single-core benchmark score can entirely run this application as Snapdragon 650 has the same score. The complimentary GPU is required, and with the Snapdragon 650, there is always Adreno 509 GPU in combination; hence this is also compatible with our application

In conclusion, we also want to put a thought about how this work is going to be helpful for a developer or a naive person while using this application. In a device which supports multi-application opening, this information is essential. Consider a scenario where the user opens 2 VR applications simultaneously, then by having CPU, GPU scores in hand can help if any of the application crashes because the user can easily debug if the cause is exhaustive usage of CPU, GPU or Memory. The novel thing about this work is to provide the 3 metric based measurements along with the application. Compared to any VR or AR application on Google Play store, they only provide minimum android versions and required permissions. In a compatible device, If the application still crashes then the list of compatible devices might be given on the developer's website, if still, the application doesn't run then there are no ways of debugging but in our case user still have parameters with him to debug with one of the reasons of crashing is depends on these 3 parameters.

In the end, we want to say our developed module will help to create a VR tour with minimum efforts, and the performance statistics are undoubtedly helpful while scaling applications for many devices and debugging the causes of crashing, if any. There is still much more scope available to expand this work, in the developed module we can include generalized speech-based navigation support or in the performance stats, we can include more

parameters and more simulated devices as well. We will discuss these things in the next section.

5.2 Future Work

Till now, we have seen too many terms, methodologies about our work. There are many benefits of the developed module or performance analysis. At the same time, while following the quote, there is nothing perfect, we can also say that there are some things about the whole work which can be better with the future research, some more studies and implementations. There is a significant scope to make our module more interactive and functional from our basic module. The teleportation method should be improved. While moving from one step to another, there is a sudden change in the image; instead, we can use a more advanced teleportation method to get a smoother and natural experience while changing the scene. There can be a provision of inclusion of 3D AR models directly by providing its local path and coordinates. One another problem is about image color scheme maintenance. In the VE, if we go inside a room, there is a sudden decrease in light, and everything becomes darker, and outside there is too bright. Of course, this can be easily achieved by capturing more suitable images, but this might not be possible each time, hence to maintain constant color tone throughout, there is a scope to implement it. One way to do this is by using some external software like Adobe Photoshop. Still, it is tedious work to do, hence if there is a provision of automatic color adjustments among images, that will be beneficial. The generalized speech-based navigation support is more crucial. When wearing an HMD or daydream, its impossible to give touch controls, we can only rely on gesture-based powers. But constant moving or head movement might cause motion sickness or pain. Hence there is a need for generalized API or a NN based model for speech-based navigation, which can run both on the desktop and mobile phones regardless of any OS.

While developing an application for mobile devices, first thing in the future we can do is,

develop a similar app for IOS devices as well. While measuring performance statistics, for mobile devices, we can increase the parameters like VRAM, display frame rate, battery usage, and many more. There can be more in-depth study like the number of threads required, heap memory used. To increase more User Interface options, there can be an inclusion of a map of the whole area showing main hotspots on it. Also, in this thesis we have decided to work with 512p cube resolution because it was maximally comfortable with most of the devices, but in future we can get statistics about many parameters while running an application with 720p, 1024p resolution as well. And after having all these insights and observations we can make a recommendation system, where by providing mobile device configuration we can possible tell which resolution can run smoothly on that device and upto how many images.

This is how we can conclude our work on the Virtual Reality-based tour creation and performance study on the mobile device.

REFERENCES

[1] *3DVista - Virtual Tours, 360º video and VR software*. URL: https://www.3dvista.com/.

[2] *8 Reasons Why Unity Game Engine is the Best*. en-US. Nov. 2014. URL: http://click-labs.com/8-reasons-unity-game-engine-best/.

[3] *Android DDMS: A Guide to the Ultimate Android Console*. en. URL: https://www.toptal.com/android/android-ddms-ultimate-power-console.

[4] Anurag. *13 Pros & Cons to Know Before Choosing Unity 3D - NewGenApps*. en-US. Mar. 2018. URL: https://www.newgenapps.com/blog/unity-3d-pros-cons-analysis-choose-unity/.

[5] *Being There: The Subjective Experience of Presence*. URL: http://commtechlab.msu.edu/randd/research/beingthere.html.

[6] Wayne E. Carlson. "13.3 Evans and Sutherland". en. In: *Computer Graphics and Computer Animation: A Retrospective Overview*. The Ohio State University, June 2017. URL: https://ohiostate.pressbooks.pub/graphicshistory/chapter/13-3-evans-and-sutherland/ (visited on 06/03/2020).

[7] Digital Catapult. *Keeping simulator sickness down*. en. Nov. 2018. URL: https://medium.com/digital-catapult/keeping-simulator-sickness-down-ca0935db95f4.

[8] *Choosing the Perfect Tech Stack: Part 1*. en-US. Sept. 2019. URL: https://visartech.com/blog/visartech-tech-stack-frontend/.

[9] Bohil Cj, Alicea B, and Biocca Fa. *Virtual Reality in Neuroscience Research and Therapy*. en. Nov. 2011. DOI: 10.1038/nrn3122. URL: https://pubmed.ncbi.nlm.nih.gov/22048061/.

[10] *Computer-generated imagery*. en. Page Version ID: 958072201. May 2020. URL: https://en.wikipedia.org/w/index.php?title=Computer-generated_imagery&oldid=958072201.

[11] *Computer Graphics | The RGB color model*. en-US. Mar. 2019. URL: https://www.geeksforgeeks.org/computer-graphics-the-rgb-color-model/.

[12] *Confused about HiDPI and Retina display? — Understanding pixel density in the age of 4K | EIZO*. URL: https://www.eizo.com/library/basics/pixel_density_4k/.

[13] Stéphane Côté et al. "Live Mobile Panoramic High Accuracy Augmented Reality for Engineering and Construction". In: Oct. 2013.

[14] Ricardo Eiris, Masoud Gheisari, and Behzad Esmaeili. "PARS: Using Augmented 360-Degree Panoramas of Reality for Construction Safety Training". In: *International Journal of Environmental Research and Public Health* 15 (Nov. 2018), pp. 1–21. DOI: 10.3390/ijerph15112452.

[15] Brandon Getty. *Virtual Reality QA Testing: What to Test and Why*. en-us. URL: https://blog.qasource.com/virtual-reality-qa-testing-what-to-test-and-why.

[16] *Glitch: The friendly community where everyone builds the web*. en. URL: https://glitch.com/aframe..

[17] *GoThru Share Link to Login*. en. URL: https://gothru.co.

[18] Mathew E. Hodges and Russell M. Sasnett. *Multimedia Computing: Case Studies from Mit Project Athena*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1993. ISBN: 9780201520293.

[19] *Majuli*. en. Page Version ID: 961624193. June 2020. URL: https://en.wikipedia.org/w/index.php?title=Majuli&oldid=961624193.

[20] *Measure app performance with Android Profiler*. en. URL: https://developer.android.com/studio/profile/android-profiler.

[21] Gavin Miller et al. "The virtual museum: Interactive 3D navigation of a multimedia database". en. In: *The Journal of Visualization and Computer Animation* 3.3 (1992), pp. 183–197. ISSN: 1099-1778. DOI: 10.1002/vis.4340030305. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/vis.4340030305.

[22] Gavin Miller et al. "The virtual museum: Interactive 3D navigation of a multimedia database". en. In: *The Journal of Visualization and Computer Animation* 3.3 (1992), pp. 183–197. ISSN: 1099-1778. DOI: 10.1002/vis.4340030305. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/vis.4340030305.

[23] *Mobile performance profiling | Mindtree*. en. URL: https://www.mindtree.com/blog/mobile-performance-profiling.

[24] *Oculus Rift | Oculus*. en. URL: https://www.oculus.com/rift/.

[25] *Oculus Rift Best Practices*. en. 2015. URL: https://www.semanticscholar.org/paper/Oculus-Rift-Best-Practices/a0e5cc13f1fb1a3ef17b4c2d4faa57e45a03eed1.

[26] *Ocurus | Build a Slick, Modern Virtual Tour*. URL: https://ocurus.com/features.

[27] *OpenSpace3D - Open Source Platform For 3D Environments*. en. URL: https://www.openspace3d.com/lang/en.

[28] Michela Ott and Francesca Pozzi. "ICT and Cultural Heritage Education: Which Added Value?" en. In: *Emerging Technologies and Information Systems for the Knowledge Society*. Ed. by Miltiadis D. Lytras et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 131–138. ISBN: 9783540877813. DOI: 10.1007/978-3-540-87781-3_15.

[29] *Pano2VR - Virtual Tour Software*. en-US. URL: https://ggnome.com/pano2vr/.

[30] Daniel Petrov. *Average phones screen size, resolution and storage are growing indeed*. en_US. URL: https://www.phonearena.com/news/Average-phone-screen-size-resolution-storage-RAM-report-AnTuTu_id106725.

[31] *Rendering and Shading*. URL: https://learn.unity.com/tutorial/rendering-and-shading.

[32] Ed Rhee. *5 apps for benchmarking your Android device*. en. URL: https://www.cnet.com/how-to/5-apps-for-benchmarking-your-android-device/.

[33] *Sketchpad*. en. Page Version ID: 958573533. May 2020. URL: https://en.wikipedia.org/w/index.php?title=Sketchpad&oldid=958573533.

[34] *Slant - 20 best alternatives to Vuze Camera as of 2020*. en. URL: https://www.slant.co/options/7717/alternatives/~vuze-camera-alternatives.

[35] Richard H. Y. So, W. T. Lo, and Andy T. K. Ho. "Effects of Navigation Speed on Motion Sickness Caused by an Immersive Virtual Environment:" en. In: *Human Factors* (Sept. 2016). DOI: 10.1518/001872001775898223. URL: https://journals.sagepub.com/doi/10.1518/001872001775898223.

[36] <img Alt=" Src='https://Secure.gravatar.com/Avatar/68fb51278d54f847dd2fa4e8aa06c978?s=80 et al. *5 Types Of Virtual Reality – Creating A Better Future*. en-US. Sept. 2019. URL: https://rextheme.com/types-of-virtual-reality/.

[37] L. Stuchlíková et al. *Virtual reality vs. reality in engineering education*. en. 2017. URL: https://www.semanticscholar.org/paper/Virtual-reality-vs.-reality-in-engineering-Stuchl%C3%ADkov%C3%A1-Kosa/af8968cda2d5042d9ea253b746324348eeebfb8b/figure/0.

[38] Cigniti Technologies. *Why Mobile App Testing is Important for Application Development?* en-US. Aug. 2017. URL: https://www.cigniti.com/blog/mobile-app-testing-important-application-development/.

[39] Unity Technologies. *Unity - Manual: Event Functions*. en. URL: https://docs.unity3d.com/Manual/EventFunctions.html.

[40] Unity Technologies. *Unity - Scripting API: Vector3*. en. URL: https://docs.unity3d.com/ScriptReference/Vector3.html.

[41] Unity Technologies. *Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations*. en. URL: https://unity.com/.

[42] David Trenholme and Shamus P. Smith. *Computer game engines for developing first-person virtual environments*. Sept. 2008.

[43] *Unity - Scripting API: MonoBehaviour*. URL: https://docs.unity3d.com/530/Documentation/ScriptReference/MonoBehaviour.html.

[44] *Unity - Scripting API: Texture2D.Texture2D*. URL: https://docs.unity3d.com/560/Documentation/ScriptReference/Texture2D-ctor.html.

[45] V Vanijja and S Horiguchi. *360 Interactive Video Scenes with Multi- Directional Moving Capability. Multidirectional Moving Capability.*

[46] *Virtual Reality and Education*. en-GB. May 2017. URL: https://www.vrs.org.uk/virtual-reality-education/.

[47] *Virtual Reality in Fashion*. en-GB. May 2017. URL: https://www.vrs.org.uk/virtual-reality-applications/fashion.html.

[48] *Virtual Reality in the Military*. en-GB. May 2017. URL: https://www.vrs.org.uk/virtual-reality-military/.

[49] *Virtual Tours Made Simple*. URL: http://roundme.com.

[50] Stanley G. (Stanley Grauman) Weinbaum. *Pygmalion's Spectacles*. English. Oct. 2007. URL: http://www.gutenberg.org/ebooks/22893?msg=welcome_stranger.

[51] *What is 360-degree VR (360-degree virtual reality)? - Definition from WhatIs.com*. en. URL: https://whatis.techtarget.com/definition/360-degree-VR-360-degree-virtual-reality.

APPENDIX

Appendix A

Mixed Reality

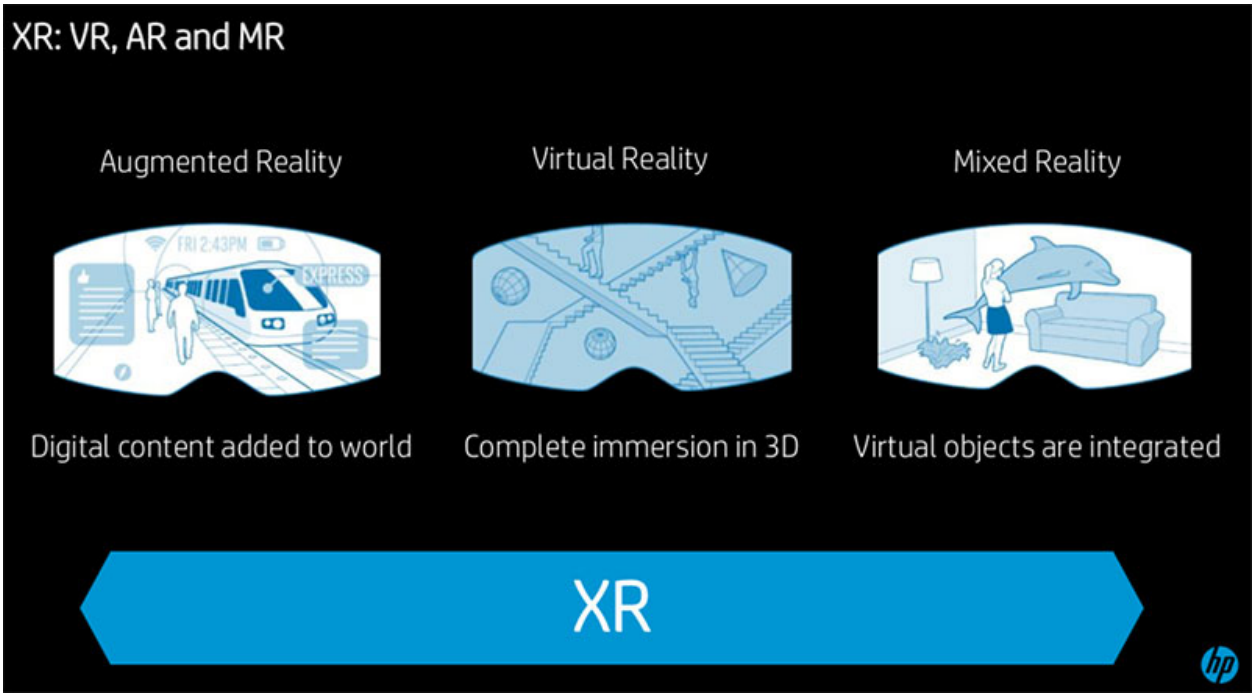

Figure A.1 Mixed reality continuum shows about merging of virtual objects into the real world.

Appendix B

Benchmark Scoring

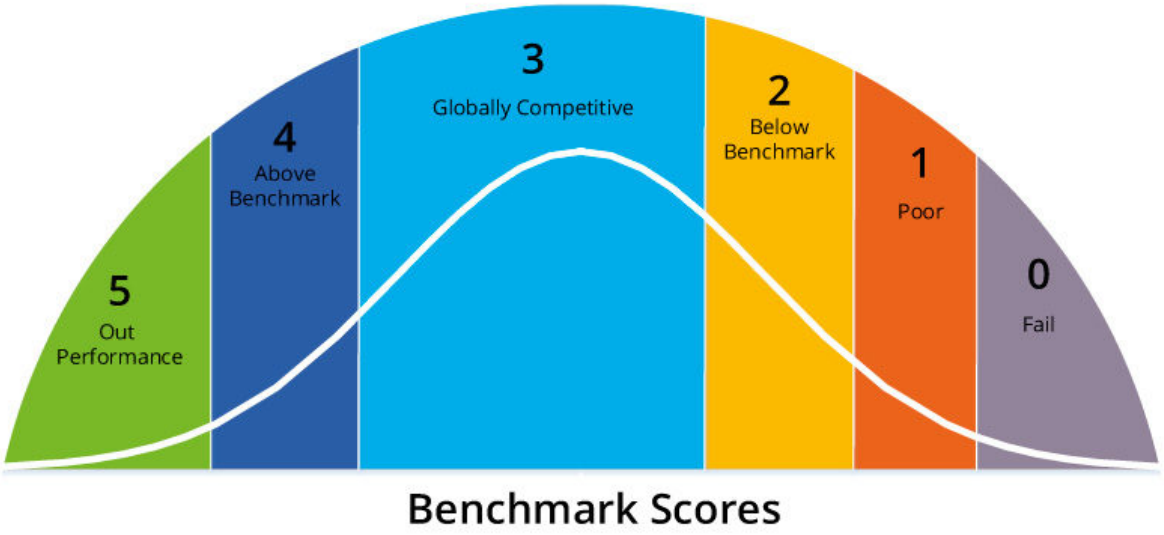

Figure B.1 Rating functionality out of 5 in benchmark scoring.

Appendix C

Core-based Benchmark Scores

Pos	Model	Cores	Architecture	64 Bit	Geekbench 4.4 64 Bit Single-Core Score	Geekbench 4.4 64 Bit Multi-Core Score
288*	Apple A12Z Bionic	8	ARM	✔		
289*	Apple A12X Bionic	8	ARM	✔	4993 n2	17866 n2
378*	Apple A13 Bionic	6	ARM	✔		
379*	Qualcomm Snapdragon 865	8	ARM	✔	4276 n3	13279 n3
417*	Apple A12 Bionic	6	ARM	✔	4774 n5	11480 n5
418*	HiSilicon Kirin 990 5G	8	ARM	✔	3925 n2	12549 n2
419*	HiSilicon Kirin 990	8	ARM	✔	3898	12280
420*	Samsung Exynos 990	8	ARM	✔	4791	12557
421*	Qualcomm Snapdragon 855+ / 855 Plus	8	ARM	✔	3654 n2	10546.5 n2
422*	Qualcomm Snapdragon 855	8	ARM	✔	3491 n14	10987.5 n14
423*	Apple A11 Bionic	6	ARM	✔	4263 n3	10380 n3
424*	Apple A10X Fusion	6	ARM	✔	3928.5 n2	9325.5 n2
562*	Samsung Exynos 9825	8	ARM	✔	4550 n2	10425.5 n2
563*	Samsung Exynos 9820	8	ARM	✔	4505.5 n4	10374 n4
564	HiSilicon Kirin 810	8	ARM	✔	2836	7891
565*	HiSilicon Kirin 980	8	ARM	✔	3314 n9	9854 n9
566*	Samsung Exynos 9810	8	ARM	✔	3688 n3	8874 n3
568*	Qualcomm Snapdragon 845	8	ARM	✔	2429 n27	8849 n27
570*	Qualcomm Snapdragon 765G	8	ARM	✔	2689	6213

Figure C.1 Global top processors with highest single core and multi core benchmark scores usinf Geekbench.