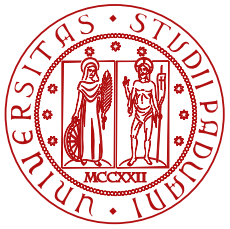

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Computer Engineering

Development of a virtual reality application for the training of cardiac surgeons

Master Candidate

Boscolo Cegion Nicola

Student ID 2074285

Supervisor

Prof. Savino Sandro

University of Padua

Academic Year
2023/2024

To my parents
and friends

Abstract

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

VR Virtual Reality

PC Personal Computer

OS Operative system

HMD Head Mounted Display

JSON JavaScript Object Notation

API Application Programming Interface

UE Unreal Engine

REST Representational state transfer

HTTP Hyper Text Transfer Protocol

IP Internet Protocol

1 Introduction

1.1 Case introduction

At the university hospital of Padua, there is a very important cardiac surgery center where various heart interventions are performed on many people, the need to teach and visualize the case of operation is very important.
Thanks to MRI, doctors can see not only pictures of the heart, but also can even make 3D models. Each 3D model can show every detail of the patient heart, this can make surgeons able to analyze and show where and how to resolve the problem.

1.1.1 Virtual reality head mounted displays

The equipment: The university of Padua has some Meta quest 2[1] [fig:1.1] a Head Mounted Display (HMD) for Virtual Reality (VR).
The Meta Quest 2 is a standalone HMD, this means that they don't need other peripherals like an external console or Personal Computer (PC) for working.
For navigating, the Meta Quest 2 uses two wireless controller, but it also has the possibility to just use your own hands to navigate the interfaces. For the context awareness it uses 4 infrared cameras and sensors like multi axes gyroscopes and accelerometers.
They use a custom version of the Android Operative system (OS), this can give

[1] All rights to meta reserved

a certain degree of liberty in creating APPs for the device.

Figure 1.1: Meta quest 2

Use cases of VR: Principally the surgeons are using VR equipment for training and showing critical health conditions of different patients hearts. They are using an APP called Shapes XR, this APP has a web interface for upload 3D models and then show them on the HMD. The app has a multiplayer functionality so that multiple people can look at the 3D models in the environment, even if the developers recommend at max 8 people, they tested with 14 users connected and there weren't any problems.

1.1.2 APP PROBLEMS

How Shapes XR works: Shapes XR lets you create rooms, accessible via a code, were multiple people can create 3D models with basic tools like 3D brushes and standard shapes like cubes, pyramids and so on. It also let you upload a 3D model file on their own website, so that in the home you can download it and start to work on it. It let you also create your own avatar. This is the main feature that the surgeons are using for show the 3D Models

The Problems: Unfortunately Shapes XR is principally used for 3D modelling, so the app has a lot of features like changing the scale of the world or brushes for modelling the objects that aren't useful for the surgeons, and quite distracting

because for a lot of people is the first time using a HMD.
The user experience is extremely important in VR because is difficult to tutoring the user while is using the HMD. Then Shapes XR position the user in an empty 3D plane with little to none point of reference so If a user accidentally uses the teleport function, they may find themselves somewhere far away from the scene they are supposed to be watching.

Feedbacks from surgeons and nurses: The 12/12/2023 I have whiteness a lesson with the integration of the Quest 2 and Shapes XR. First it was pretty chaotic, a lot of people didn't even know how to use the controllers, and they had problems even for putting the code for entering the session. Unfortunately we didn't have time to make a nice lesson for teaching how to use the HMD, the tutorial made by Meta approximates takes 15min to complete, even more if the user want to try the mini-games, so we did not have the time to show it. The main critical points were:

- Inadequate tutoring for teaching how to use the HMD
- Difficulty for accessing the multiplayer room
- Difficulty at moving in the room
- Some people avatar were blocking the visual of some people

How we can see a new app for this use case could be useful, and this is the main reason why this project exist.

2 Requirements

2.1 Functional Requirements

The main features of the project must be:

- **Show custom 3D model at runtime:** The software must be able to download 3D models in OBJ format and rendering them at runtime, with also show coloring made by a surgeon.
- **Multiplayer functionality:** The software must recreate a 3D virtual classroom where a professor can show the 3D model to the students, all students and professor must be synchronized.
- **Upload of 3D models:** There will be a web portal were a surgeon can upload and preview 3D models.
- **Tutoring functionality:** The software will have some tools for learning such as lase pointers, and the possibility to change the position and dimension of the 3D model.
- **Compatibility with Meta quest 2:** The system will need to run on the Meta quest 2.

2.2 Data requirements

The system necessitates the following data (with the following characteristics) to fulfill its objectives:

- **Required Data:**

  - 3D models in OBJ format.
  - Local Internet Protocol (IP) address for multiplayer.
  - Generating and managing codes for multiplayer sessions.

- **Data Format:** All data must be respect its standard, other communication betwin clients and server must use JavaScript Object Notation (JSON) format file. These formats facilitate data exchange with external systems.

2.3 Non-functional requirements

To ensure the development of an effective VR software, the following non-functional requirements must be considered:

- **Use of Game engine:** A game engine is a software made by another company that is capable of creating a video game, by giving the developers some tools for making the development experience quicker and easier.
- **Simple Back end:** The backend must do the least things effectively because so that the IT department does not need to update the server that will host the services.
- **Compliance with Internet standards:** The system will be compliance with Hyper Text Transfer Protocol (HTTP) for all communication betwin server and client, the HMD will use the game engine multiplayer system.

3

Preliminary project

The project is made by three main components: VR APP, backend, webui. This chapter will talk about them.

3.1 The VR APP

This section will explain the main choices behind the VR APP

The development environment: There are three main way to develop on the Meta Quest 2:

- **Native:** By using native Application Programming Interface (API) that Meta provides for the HMD.
- **Unity:** Is a famous game engine principally used for lightweight video games, It is pretty functional and easy to use, its programming language is C#.
- **Unreal Engine:** Is a famous game engine used for big games, its performance is the best in the market, it uses C++ and a graphical programming language called Blueprint.

As we said in the non-functional requirements we will use a game engine, I have opted for Unreal Engine (UE) because of its performance, the 3D models are pretty complex (~10MByte in size), so we need good performance. It has a lot of tools for multiplayer and a nice documentation other than a big community of developers.

3.2 Unreal Engine

Made by Epic Games, the project was born for the video game Unreal, now is one of the best engine that power a lot of important video games and 3D animation. For this project it will be used the 5.2.1 version for the best compatibility with the HMD, still is not an old version, the last one is 5.3.

The fundamentals: UE has a 3D preview mode that let the user set the various 3D objects in the scene. In unreal a scene is called "Level", each element in a level is called "Actor", Actors are made by multiple components.
Each element of unreal can be build with a propriety system called Blueprint, a visual programming interface, or via C++, or by combine them. Principally Blueprint can make the developing of the project faster and easier at the cost of performance, C++ perform better, and it has a bigger range of tools. So It is important to combine both languages for having the best performance and flexibility in the project.
Like a lot of game engines, UE tries to render as much frame as much as possible, each cycle of rendering is called tick.
Its important that long actions must be asynchronous respect the game ticks, and if something must be runner in the so-called "game thread", it must be as fast as possible so that the frame rate does not drop to a certain level.

Events: Events are what start the execution of code inside a Blueprint. Each blueprint have its own standard event, they depend on the object, the basic ones are:

- **Begin Play:** runs one time when the actor is spawned (this is not a constructor).
- **On tick:** runs on every game tick.

But we can also create custom event with Blueprint or C++, they can be triggered whenever we want, they can also be replicated in clients in multiplayer sessions. Like functions, events can have inputs but not outputs.

Multiplayer: UE just support a client-server configuration for manage multiplayer session, it has two main implementation: Stand-Alone server and listening server.

Stand-Alone server consist in having a server that emulates the gaming session, it requires a server powerful enough to run the basic function of the game event if it does not need to render it.
Listening Server does not need a Stand-Alone server, simply one device acts not only as a client but also as a server. Normally client-server is the best in performance and latency, but because this software is simple, the listening server is the right implementation.

3.3 Network Infrastructure

The Network Infrastructure [fig:3.1] it is simple, a server will be hosted in the IT department of the hospital that will host the Representational state transfer (REST) server for managing 3D Models and multiplayer sessions, and in the same server will host via NODEjs the Web Portal for managing the 3DModels. The multiplayer itself will be manage by the host HMD

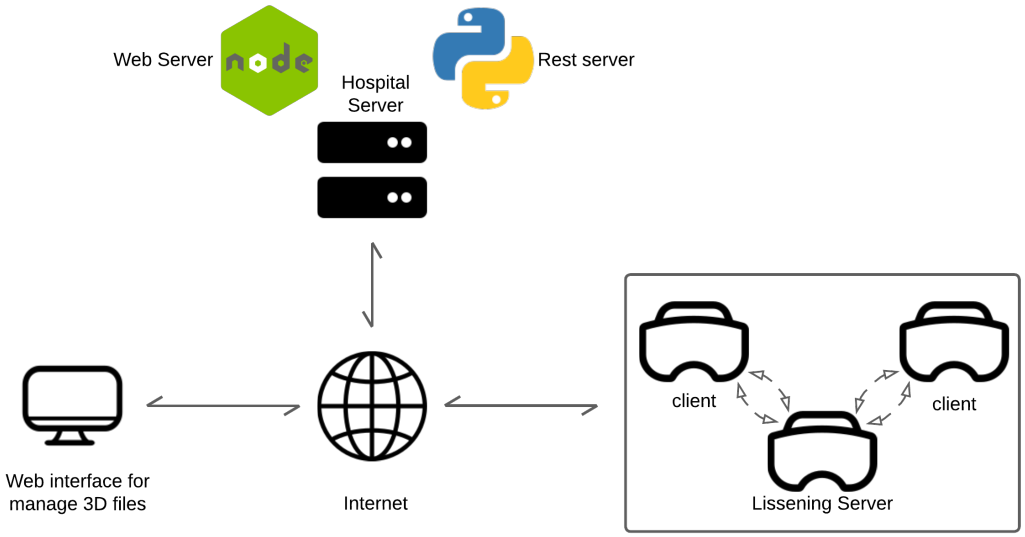

Figure 3.1: Network Schema

The backend: The backend is a simple Python program that it functions as a REST server, were each 3D model is a resource. It also manages the multiplayer session by creating the session code, and save the IP address of the listening

server. The library used is called Flask[1], it simply let you run a function respect to a HTTP response received.

The WebUI The WebUI is made with reactJS, a popular framework for website, the main reason of this choice is for faster development because its community is massive, in fact will be using a UI library called MUI[2] and a 3D library for rendering 3D models called React Three Fiber[3].
The main functionality of the WebUI will be:

- previewing 3D models with the right colors
- upload 3D models
- delete 3D models

[1] https://flask.palletsprojects.com/en/3.0.x/
[2] https://mui.com/
[3] https://r3f.docs.pmnd.rs/getting-started/introduction

4 The project

4.1 A section

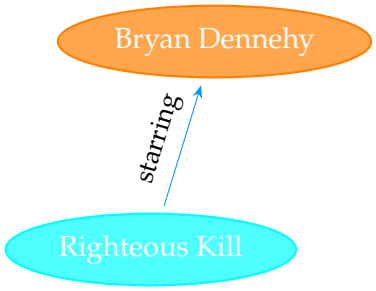

Figure 4.1: Image created with TikZ

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
```

```
    VT = np.zeros((n*m,1), int)  #dummy variable

    test = "String"

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Code 4.1: Code snippet example

5

Conclusions and Future Updates

A	**B**
C	D
E	F
G	H

Table 5.1: Table example

Acknowledgments