University of California,
Santa Barbara

CS189 Fall'17 Capstone

VR Telemedicine

Product Requirement Documentation

Jinfa Zhu
Kenneth Chan
Shouzhi Wan
Xiaohe He
Yuanqi Li

Supervised by
Ole Eichhorn
Helen Hawkins
Nate Pincus
Marco Pinter
Jazarie Thatch

Contents

1 Introduction

1.1 Problem

Doctors are limited by the way they access patient data. Current interfaces are a step up from excel spreadsheets and file folders with attached scan images, but can be improved.

1.2 Vision

As virtual reality (VR) becomes more mainstream, we recognize the potential for using VR to create a more powerful and efficient user interface, aimed towards enhancing physician experience and making their jobs easier. A VR interface, coupled with hand gesture detection, can make the beautiful cinematic interfaces from science fiction into reality.

1.3 Challenges

Despite the popularity VR has gained, it is still a technology that is rather immature compared to other fully developed technologies. Intense research with relatively few resources available is needed to get familiar with VR-related software development. Current challenges in this field, such as low resolution and video quality, are factors that should be taken into consideration. Human-computer interaction, in particular, is a problem many researchers are working on, and is closely related to this project. Another downside of VR in general is its high expense, which underlines the need to make full use of VR-specific features to provide new and useful services that cannot be done in traditional platforms, in order to make this platform worth its price.

1.4 Innovations

Compared to traditional interfaces, an interface utilizing VR provides many more future remote access possibilities for doctors. The first phase of this project implements a dashboard displaying and providing easy access to real-time patient information. The advantage this dashboard has is the significant increase of the amount of information a doctor can see in his/her field of vision, and the convenience of navigation through this system. Although this phase itself is only improving the performance of current features, this VR platform we are building will open up further possibilities for brand new features, such as monitoring conditions of patients by doctor-controlled movable cameras, remotely performing surgery, or collaborating in the same virtual space. Although we will not be able to explore all these possibilities, this project will be the first step in transforming imagination into reality.

2 Project Specifics

2.1 Objectives

The main goal of our project is to give doctors more accessibility to real-time patient information and records via an immersive user interface. The VR interface will use Cculus controllers in allowing doctors to retrieve vital signs and accessing simultaneous data and imagery windows.

2.2 Background

Modern telemedicine solutions allow doctors to see patients remotely over video conferencing, and interact with them on a limited level. However, this user experience is restricted by traditional human-computer interaction methods, such as with a keyboard and a mouse. Touch screen innovations expand the capabilities of the user interface by permitting swipe, pinch, and rotate gestures to navigate a 2D menu, but are not as powerful as a 3D virtual reality interface.

2.3 Assumptions

- Data (vitals data, live video, brain MRI scans, lab results) will be readily available.
- Internet connection between dashboard client and data servers will not be dropped.

3 System Architecture Overview

3.1 High level diagram

Illustrated below is our preliminary system architecture. There are four parts of this architecture: Physician's client, Patient's client, Server, and Cloud database. Arrows illustrate communication between respective entities.

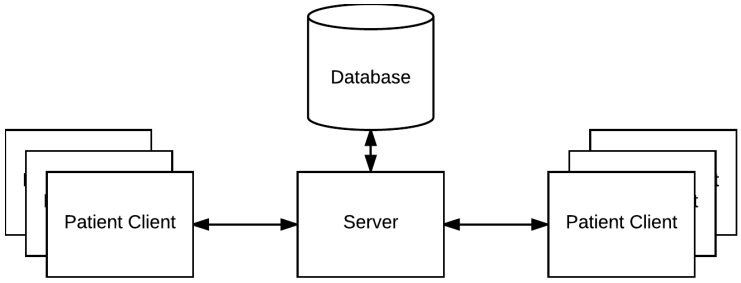

Figure 1: High level architecture

3.1.1 Physician's client

The physician's client consists of a desktop computer and a VR device. The physician can connect to the server by logging in via the webapp. This will retrieve the physician's account data and patient data associated with the physician. Usually, the physician's client will only communicate with the server, which handles the transmission of all data and messages.
In an emergency, a video/radio call will be initiated directly between a physician's client and a patient's client instead of through the server to guarantee performance.

3.1.2 Patient's client

The patient's client is an application that connects to the server to retrieve and display patient exam data. In addition, the patient can update personal information and send messages to their physician. This is similar to the physician's client. It will only communicate with the server, except in the case of an emergency, in which the patient's client will initiate a connection with the physician's client.

3.1.3 Server

The server is responsible for responding to all user connections. It connects to the database to retrieve and update the data.

3.1.4 Cloud database

The database stores all user data, including users' account data and messages. It connects to only the server to guarantee security.

3.1.5 VR application architecture

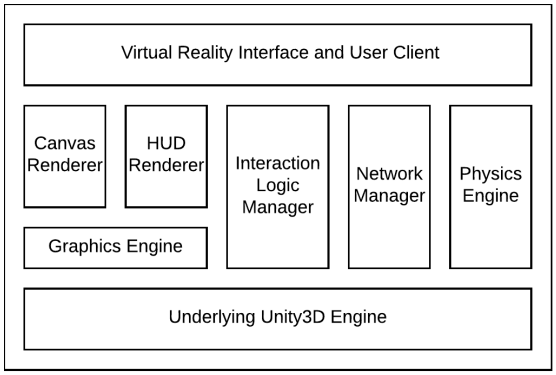

Figure 2: VR application architecture

The VR application is used by a physician to access all his/her data. The underlying architecture is shown as above. Several modules are responsible for different functionalities: graphics, interaction logic, networking, and physics. All of these modules are built upon the Unity3D engine.

3.2 User interaction and design

3.2.1 Web interface

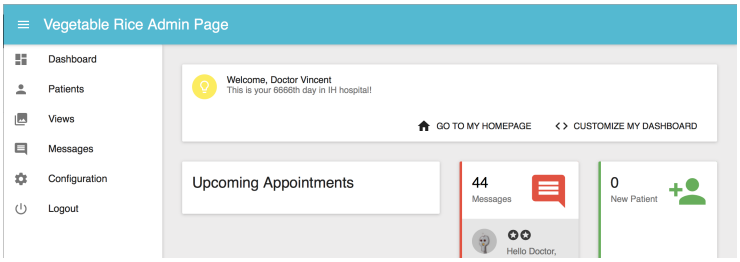

Figure 3: Web administration Interface

After signing in, the doctor can see a dashboard which he can look for his appointment, check messages and add new patient.

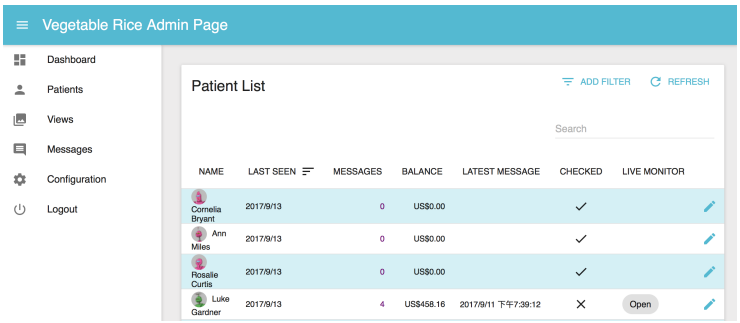

Figure 4: Web administration patient list

After clicking on Patient button on the left panel, the doctor can see a patient list with their detail information, and after clicking on the View button, the doctor can see the patient through the patient's camera.

3.2.2 VR application interface

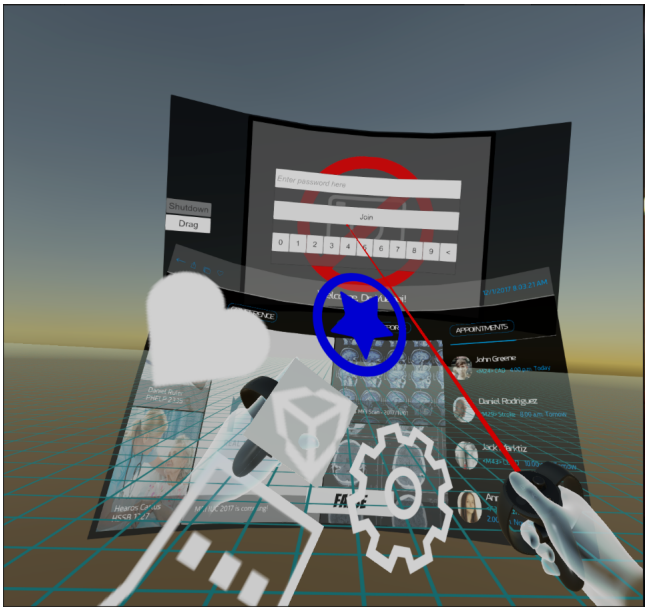

Figure 5: VR interface

In this common VR interface, user's left hand holds a menu which has four buttons that each can open/close a dashboard; user's right hand holds a ray shooter which allows object selection. In front the user are two dashboards (the top one is WebRTC interface, while the bottom one is main dashboard).

4 System Models

4.1 Structural UML diagrams

4.2 Sequencing diagrams

Here are two important sequences in this project. Figure 6 illustrates the communication between a web client and a central server, which dispatches data queries and updates to the backend database. The web client doesn't need to send a specific SQL query, which is generated by the server. Figure 7 illustrates the communication between a VR application and a central server. It works in the same manner as the previous one. Upon message arrivals, the VR application will render data contained in responses and display them in a 3D world.

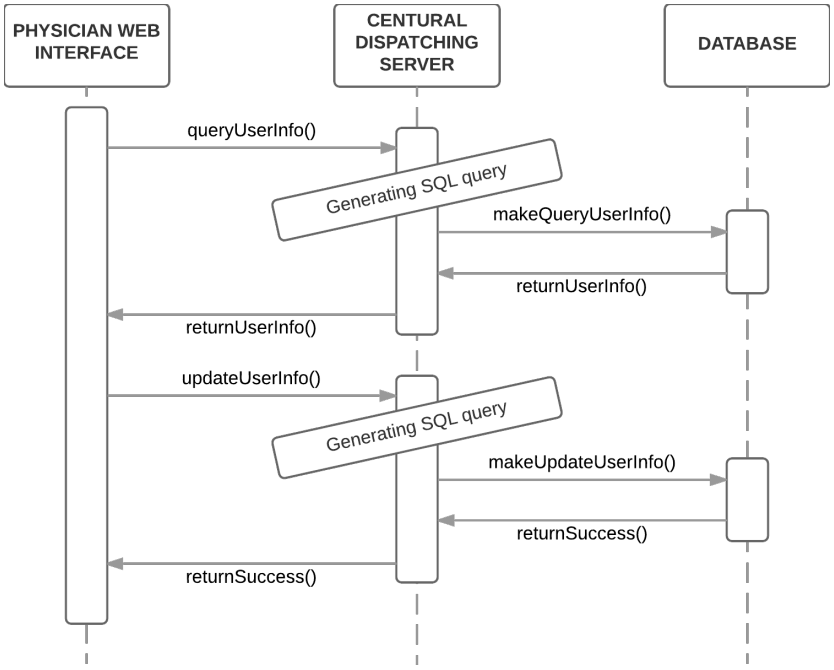

Figure 6: Web administration sequencing diagram

4.3 Use cases

The use case diagram illustrates a use case in which a patient sends a request to the doctor to have a online diagnose. The doctor pulls out the patient's documentation and sends a message to another doctor. Both doctors join the chat room with the patient to study the case and finally, the main doctor gives

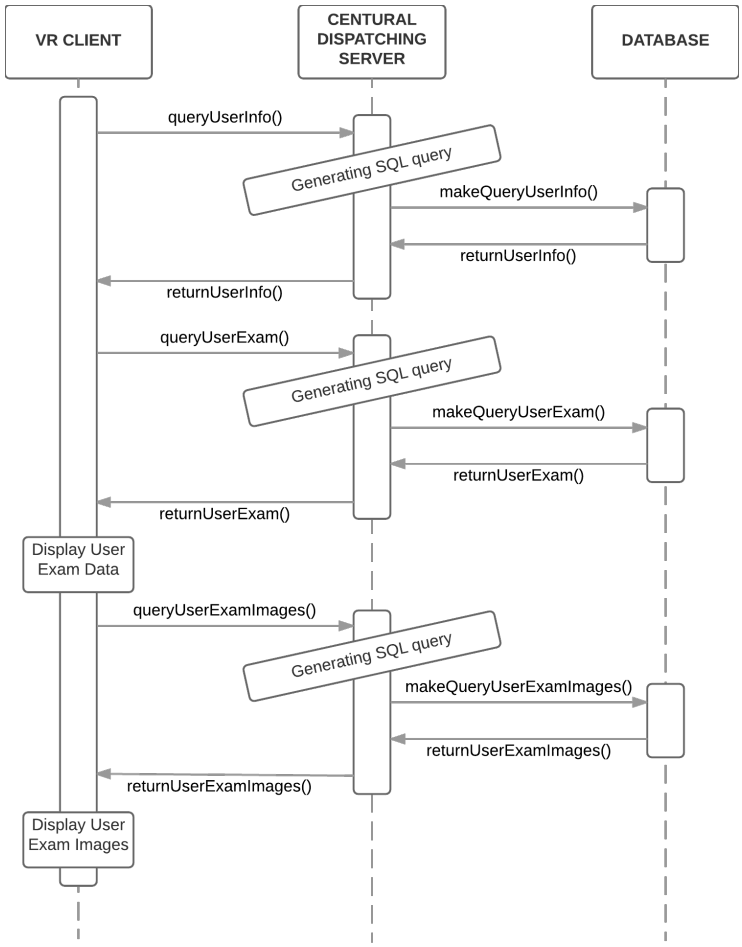

Figure 7: VR application query sequencing diagram

a prescription to the patient and ask a nurse to update the documentation of that patient.

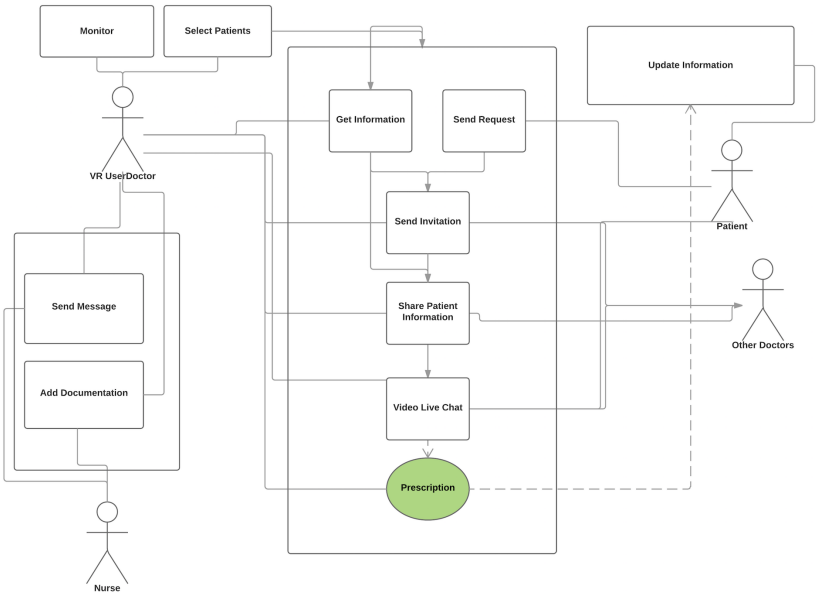

Figure 8: Real-time video streaming

5 User Stories

5.1 Physician identity and device registration

As a doctor, I want to access the website, so that I can register an account for myself and my VR device and update my information.

5.1.1 Acceptance criteria

- A physician cannot be registered without a device serial number
- All updates made by a physician should be synchronized on the database immediately

5.1.2 Non-functional requirements

- Front-end is implemented using Bootstrap
- Back-end is implemented using Ruby on Rails
- All information is stored on a database managed by a server

5.1.3 Prototype and tests

`https://github.com/yuanqili/CS189-F17/tree/master/web/Server`

5.2 Physician Login

As a doctor, I want to log in to the server so I can view my VR dashboard.

5.2.1 Acceptance criteria

- Login fails if incorrect user credentials are submitted

5.2.2 Non-functional requirements

- Use Unity to build a canvas for log in page
- Use Unity to build login buttons and functions of the buttons
- Use C# to access the database
- Design ER diagram and relational Schema
- Implement the Database using MySQL
- Create sample data for login
- Compile and test the login system

5.2.3 Prototype and tests

`https://github.com/yuanqili/CS189-F17/tree/master/src/unity-ui/Assests/_Scene`

5.3 Physician main dashboard navigation

As a doctor, I want to look around in the VR dashboard so that I can have different kinds of data available at my fingertips.

5.3.1 Acceptance criteria

- Appointments should be present as a preview in the dashboard
- Patient records should be displayed in a curved UI
- View should change when looking around

5.3.2 Non-functional requirements

- Put in the canvas and the 3D background
- Connect the VR and test the dashboard
- Re-modify the Dashboard based on VR testing

5.3.3 Prototype and tests

`https://github.com/yuanqili/CS189-F17/tree/master/src/unity-ui/Assests/_Scene`

5.4 Physician Window Selection

As a doctor, I want to select a window on my dashboard so that I can view the information in another screen in my VR device.

5.4.1 Acceptance criteria

- Look at the window and the window button should respond
- Use controller to select the window on the VR

5.4.2 Non-functional requirements

- Implement button functions and keyboard interaction
- Implement windows and canvas after click on the button
- Implement controller functions and interaction
- Implement camera selection and movement.

5.4.3 Prototype and tests

https://drive.google.com/drive/folders/19TFNBxUA2S0h1OXHT8zCnCOPZHpdv_Di?usp=sharing

5.5 Patient specific dashboard

As a doctor, I want to select a square in my dashboard containing patient data so that I can see all the patient's exam data, scan images, and patient information in a separate window.

5.5.1 Acceptance criteria

- On click, the scene should change to patient's electronic record
- Patient's record scene should list all information that has already stored in the database

5.5.2 Non-functional requirements

- Design a patient info dash board
- connect different windows to different database contents
- Put in the fake information about the patients
- profile shows a graph of recent change if necessary
- pop up a small window show detail of certain info (eg. brain scan) if clicked

5.5.3 Prototype and tests

https://github.com/yuanqili/CS189-F17/tree/master/src/unity-ui/Assests/_Scene

5.6 Body Scan Information

As a doctor, I want to compare multiple body scan images from the same patient so I can see the change in the patient over time.

5.6.1 Acceptance criteria

- Upon selecting a scan image, the scene should display the scan image.
- Scan image should be labeled and relative metadata displayed.
- The doctor should be able to interact with the image with basic move, resize, and rotate actions, as well as image editing options such as adjusting transparency, color, and brightness/contrast.
- The doctor should be able to open up multiple scan images and display them side to side, or layer them on top of each other.

5.6.2 Non-functional requirements

- Patient profile shows a graph of recent change
- Scan images highlight differences over time

5.7 Doctor Controlled Windows

As a doctor, I want to reposition the dashboard so I can more comfortably view the data displayed to me.

5.7.1 Acceptance criteria

- Ability to move, zoom, and select windows with the controller

5.7.2 Non-functional requirements

- use C# to implement, select, open and close
- finger gestures zoom in and zoom out windows
- hand gestures move and throw windows
- Test the controller on VR and modify the control system

5.7.3 Prototype and tests

https://drive.google.com/drive/folders/19TFNBxUA2S0h1OXHT8zCnCOPZHpdv_Di?usp=sharing

5.8 Notification Sending

As a patient, I want to send a message to my doctor so that I may notify the doctor of any updates or concerns.

5.8.1 Acceptance criteria

- Able to enter in text data via the website and send SMS messages to the doctor

5.8.2 Non-functional requirements

- Use Python Flask to build the server webapp to handle patient messages
- Text messages should be sent to the dashboard application as well as inserted into the database

5.8.3 Prototype and tests

`https://drive.google.com/drive/folders/19TFNBxUA2S0h1OXHT8zCnCOPZHpdv_Di?usp=sharing`

5.9 Doctor Popup Message notification

As a doctor, I want to receive notifications while using the dashboard, so that I can address any concerns from my patients.

5.9.1 Acceptance criteria

- Pop-up icon present on the toolbar notifying the doctor about new message

5.9.2 Non-functional requirements

- Selecting the message notification should bring up the message page
- Multiple unread messages supported

5.9.3 Prototype and tests

`https://drive.google.com/drive/folders/19TFNBxUA2S0h1OXHT8zCnCOPZHpdv_Di?usp=sharing`

5.10 Record and Documentation Update

As a doctor, I want to record my prescription and notes for each patient so that I can have them for future reference.

5.10.1 Acceptance criteria

- Voice to text should transcribe the doctor's notes
- Records should be inserted into the database
- The patient should able to obtain the relevant prescription and notes from the server.

5.10.2 Non-functional requirements

- System need to generate a audio record of what doctor says
- store the prescriptions into the database
- add documentation functions to VR system

5.11 Camera Movements

As a doctor, I want to move around the camera so that I can monitor the patient

5.11.1 Acceptance criteria

- Move the head should see different scenes

5.11.2 Non-functional requirements

- Use controller to move around the camera

5.11.3 Prototype and tests

`https://drive.google.com/drive/folders/19TFNBxUA2SOh1OXHT8zCnCOPZHpdv_Di?usp=sharing`

5.12 Doctor On-Hand Notepad

As a doctor, I want to hold a Notepad when visiting different pages.

5.12.1 Acceptance criteria

- Able to hide on the bottom of the screen
- Able to move with the camera
- Able to contain text notes, images, and videos.

5.12.2 Non-functional requirements

- Build up a new canvas
- hand gestures to hide and hold the note pad
- Test the controller on VR and modify the control system

5.12.3 Prototype and tests

https://drive.google.com/drive/folders/19TFNBxUA2S0h1OXHT8zCnCOPZHpdv_Di?usp=sharing

5.13 Real-time Doctor-Patient Communication

As a patient, I want to talk with doctors in real-time so that I can follow their instruction.

5.13.1 Acceptance criteria

- With good network condition, patient should transmit video data and doctor should transmit voice data.
- videos and voice data should be stored if necessary

5.13.2 Non-functional requirements

- implement video chat function in patient app
- add function to doctor.

5.13.3 Prototype and tests

https://drive.google.com/drive/folders/19TFNBxUA2S0h1OXHT8zCnCOPZHpdv_Di?usp=sharing

5.14 Doctor Tool Box

As a doctor, I want to have some flouting tool box on my sceen whatever page I visit.

5.14.1 Acceptance criteria

- Able to select and use the tools on anypage
- Tool box should not disappear when change windows.

5.14.2 Non-functional requirements

- create other canvas arround the camera
- put and implement tool bars on the canvas

5.14.3 Prototype and tests

https://drive.google.com/drive/folders/19TFNBxUA2S0h1OXHT8zCnCOPZHpdv_Di?usp=sharing

5.15 Patient Website

As a patient, I want to login to a website and see my records and send messages to the doctor.

5.15.1 Acceptance criteria

- Able to register and login
- Able to view all the documentations
- Able to send text messages to the doctor

5.15.2 Non-functional requirements

- Use python flask to build a website.
- connect the website to the server
- have access to the database

5.15.3 Prototype and tests

`https://github.com/yuanqili/CS189-F17/tree/master/web/server/app/main`

5.16 Database Manager

As a service provider, I want to create doctor and patient profiles so that they can get access to the server.

5.16.1 Acceptance criteria

- Other people except service provider should not have access to create these profiles

5.16.2 Non-functional requirements

- implement management application to change user info

5.17 Emergency System

As a patient, I can send an alert to the Doctor so that they can instruct me when there is a emergence.

5.17.1 Acceptance criteria

- Send alert from phone and VR pop up a window

5.17.2 Non-functional requirements

- Implement emergency function in patient application

5.18 Doctor-doctor Communication

As a doctor, I can contact with other doctors in VR system so that we can share the opinions about some patient cases.

5.18.1 Acceptance criteria

- Login and select communication, Should able to see online doctors
- Send request to join the cases
- Two or more doctors in the same scene
- Have access to the same patient case

5.18.2 Non-functional requirements

- Real time chat system for doctor application

5.18.3 Prototype and tests

`https://drive.google.com/drive/folders/19TFNBxUA2S0h1OXHT8zCnCOPZHpdv_Di?usp=sharing`

5.19 Settings and preferences

As a doctor, I want to receive notifications while using the dashboard, so that I can address any concerns from my patients.

5.19.1 Acceptance criteria

- Can see a little pop up window informing the doctor about the coming message

5.19.2 Non-functional requirements

- tap on the message notification should bring the doctor to the massage page
- Automatic close the notification if there is not action
- should allow multiple messages to come in

5.20 3D Modeling of Slice Images

As a doctor, I want to see 3D modeling of slice images (e.g., DICOM files), so that I can have a clearer understanding of a patient's status.

5.20.1 Acceptance criteria

- A 3D model built upon DICOM files should be properly rendered in Unity, and the doctor can touch and manipulate it.

5.20.2 Non-functional requirements

- A volumetric renderer is needed to render 3D model from DICOM slice images.
- The model rendered from images should be still clear enough so it doesn't lose image details.
- It should be rendered in real-time, or at least after a short wait of pre-rendering, and then there is no delay. Otherwise it will be useless and hard to manipulate.

5.21 Selecting Objects Using Shooting Rays

As a doctor, I want to select objects and canvas in the 3D space using a shooting ray, so that I don't have to walk closer to them to grab and move.

5.21.1 Acceptance criteria

- A shooting ray originates from right hand controller collides with the first object it touches, and that object should be marked as movable.

6 Appendices

6.1 Technologies employed

6.1.1 Server

- **Bootstrap**. Simple yet pretty front-end web pages can be written using Bootstrap, which saves our a lot of energy.
- **Apache2**. Apache2 is our web server to publish our website powered by flask.
- **Python Flask**. Python Flask is our main framework to build website backend. It provides stable and powerful building tools to setup the back-end and database connection and also, it provide some simple structures to connect database.

6.1.2 Database

- **MySQL**. The main advantage of MySQL is that our database administrator is very familiar with it and has years of experience using it.

6.1.3 Virtual reality application

- **Unity**. The Unity game engine provides a full set of developing tools to implement VR applications. It also has an very active community which provides us solutions and assets that boost our development.
- **Oculus Rift**. Oculus Rift is one of the most popular VR devices. It also has an active community and works very well with Unity.
- **WebRTC**. WebRTC is a collection of communications protocols and application programming interfaces that enable real-time communication over peer-to-peer connections. This allows different clients to not only request resources from backend servers, but also real-time information from clients of other users.
- **DICOM**. Digital Imaging and Communications in Medicine (DICOM) is a standard for storing and transmitting medical images enabling the integration of medical imaging devices such as scanners, servers, workstations, printers, network hardware, and picture archiving and communication systems (PACS) from multiple manufacturers. It has been widely adopted by hospitals, and is making inroads into smaller applications like dentists' and doctors' offices.