Data Visionaire

(Data Mining and Visualization in Virtual Reality)

DISSERTATION

Submitted in partial fulfillment of the

Requirements for the award of the degree

of

Bachelor of Technology

in

Computer Science & Engineering

Submitted By:

Aditya Arora	(014/ CSE-1/ 2014)
Manjog Singh	(038/ CSE-1/ 2014)
Amanjeet Singh	(043/ CSE-1/ 2014)
Harinder Pal Singh	(054/ CSE-1/ 2014)

Under the guidance of:

Ms. Geetika Bhatia

Department of Computer Science & Engineering
Guru Tegh Bahadur Institute of Technology

Guru Gobind Singh Indraprastha University
Dwarka, New Delhi

Year 2017-2018

DECLARATION

We hereby declare that all the work presented in the dissertation entitled **“Data Visionaire”** in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science and Engineering**, Guru Tegh Bahadur Institute of Technology, affiliated to Guru Gobind Singh Indraprastha University, Delhi is an authentic record of our work carried out under the guidance of **Ms. Geetika Bhatia**.

Date:

Aditya Arora	(014/ CSE-1/ 2014)
Manjog Singh	(038/ CSE-1/ 2014)
Amanjeet Singh	(043/ CSE-1/ 2014)
Harinder Pal Singh	(054/ CSE-1/ 2014)

CERTIFICATE

This is to certify that dissertation entitled **"Data Visionaire"**, which is submitted by **Mr. Aditya Arora**, **Mr. Manjog Singh**, **Mr. Amanjeet Singh and Mr. Harinder Pal Singh** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science & Engineering**, Guru Tegh Bahadur Institute of Technology, New Delhi is an authentic record of the candidate's own work carried out by them under our guidance. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Geetika Bhatia
(Project Guide)

Aashish Bhardwaj
(Head of Department)
Computer Science & Engineering

Date:

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to **Ms. Geetika Bhatia** to provide us the opportunity to conduct this project as the minor project which not only increased our awareness about the latest field but also taught us the importance of discipline & dedication towards our work. Without her help we could not have presented this dissertation up to the present standard. We are highly thankful to **Mr. Ashish Bhardwaj** and **Ms. Poonam Narang** for giving us the opportunity to work upon this project.

We choose this moment to acknowledge the contribution of each team member gratefully.

Date:

Aditya Arora
(014/ CSE-1/ 2014)
aditya94@gmail.com

Manjog Singh
(038/ CSE-1/ 2014)
manjogsingh@gmail.com

Amanjeet Singh
(043/ CSE-1/ 2014)
amanjeetsingh150@gmail.com

Harinder Pal Singh
(054/ CSE-1/ 2014)
harinder2612@gmail.com

ABSTRACT

Data Visionaire (Data Mining and Visualization in Virtual Reality) project is amalgamation of two profound technologies which is Data Analysis and Virtual Reality. In today's world we deal with complex and multidimensional data which can be cumbersome to represent and analyze. This multidimensional data is can be visualized in a way which is easy to interpret. Due to our limited perception towards multidimensional data, we require more intelligent environment to represent that data. Our goal is to create that environment in virtual reality and not only visualize complex data in 3D, 4D and 5D but also do OLAP operations like Roll-up, Drill-down, slice, dice, pivot and more. In this project we are using HTC Vive, a virtual reality headset, through which user will enter into the world of multidimensional data. Using techniques like clustering, classification and other data analytics operations we will form knowledge base from raw data files and this informative vital data will be visualized in Virtual Reality and also on Android Application for on the go references.

LIST OF TABLES AND FIGURES

CONTENTS

CHAPTER ONE

INTRODUCTION

1. Project Description

Data Visionaire (Data in Virtual Reality) project is amalgamation of two profound technologies which is Data Analysis and Virtual Reality. In today's world we deal with complex and multidimensional data which can be cumbersome to represent and analyze. This multidimensional data is can be visualized in a way which is easy to interpret. Due to our limited perception towards multidimensional data, we require more intelligent environment to represent that data. Our goal is to create that environment in virtual reality and not only visualize complex data in 3D, 4D and 5D but also do OLAP operations like Roll-up, Drill-down, slice, dice, pivot and more.

In this project we are using HTC Vive, a virtual reality headset, through which user will enter into the world of multidimensional data. Using techniques like clustering, classification and other data analytics operations we will form knowledge base from raw data files and this informative vital data will be visualized in Virtual Reality and also on Android Application for on the go references.

The problem statement that we have opted is the prediction of growth of crops on the basis of 4 features: Temperature, Annual Rainfall, Humidity and Soil Type.

1.1. Existing System

The conventional concept of Data Representation or Visualization is two dimensional in nature. Visualization techniques can be both elementary (line graphs, charts, bar charts) and complex (based on the mathematical apparatus). Furthermore, visualization can be performed as a combination of various methods. However, visualized representation of data is abstract and extremely limited by one’s perception capabilities and requests. Data Analytics desktop applications only lets you create a collection of queries, data connections, and reports. This limitation of data representation can be overcome with increased perception and higher interpretation with the help of virtual reality.

1.2. Proposed System

The proposed analytical tool can be considered an excellent tool to increase the quality and speed of data analysis. It offers a wide range of aesthetics and performance features compared to the classical systems and we intend to turn it into a commercial product for professional. . Our proposed learning tool provides visual feedback. With virtual reality amplifying the dimensional limits, we can achieve a much more interactive user environment for interpretation of data. The tool will consist of various data analytic operations ranging from classification and clustering to OLAP operations on multidimensional hypercubes.

1.3. Processing

In this project user will be able to see all the data in different forms with the help of HTC Vive, which is a Virtual reality headset. Basically, it solves two purposes one the Data visualization in 3d of cleaned and analyzed data. User can visualize the data mining techniques: clustering and its results, various graphs related to problem statements. In this user can also save this query for later, or bookmark this query and see it on Android App.

1.4. Problem Statement

The problem statement that we have opted is the prediction of growth of certain crops on the basis of 4 features:

- Temperature
- Annual Rainfall
- Humidity
- Soil Type

The basic assumption that we are dealing with is that the model will only predict which crop to grow only at the point of germination. Here we first clustered the data points around the favorable conditions for the growth of crops. Here we have taken 10 crops for analysis.

After clustering we have also provisions for data analysis that includes crop production analysis and cost production analysis which the farmers need to understand to maximizing their profits.

1.5. Solution and Features

This project aims to develop a Multidimensional Data Analytical Tool, which will increase the perception of user towards multidimensional data. The prominent features of this tool will be the following:

- Visualization of OLAP operations: This feature allows user to visualize data in hypercubes and also make OLAP operations on this data. E.g.: Pivot, Drill Down, Roll Up, Slice, Dice.
- Various data analytics operations: In this user will be able to see clustering and classification happening live in front of them in 3d which helps them understand the multidimensional data better and purpose of clustering and classification in Data Analysis.
- Representation of multiple dimensions of data: In this user can represent not only 2d but 3d,4d data in Virtual Reality. That means user can submit multidimensional query in which data variants are more than 4 sometimes.
- Bookmarking the previous operations: A user can bookmark any data query which they did in past. Simply by clicking on the bookmark whole data representation will be back in front of them. This feature helps easier access to more relevant data to the user.
- Easy to share screenshots of multidimensional data: All queries saved by user is also available in Android App, where user can either choose to see the screenshot of data Take in Virtual Reality, or see it live on the Android Platform.
- Inferences and Predictions: User is presented with already deduced inferences based on the analysis of data of previous year. Along with that functionality of prediction of data using regression technique is also available for user. A user can predict value of certain attribute which is dependent on multiple factors to great accuracy.

- Similar functionality in Android: User can access all this data in Android App as well. It is for on the go reference, in android app user can make predictions and represent data on Map of India.

1.6. Advantages

1. Better understanding of the patterns and data in a more interactive way.
2. Easy to visualize the multidimensional data.
3. Better understanding of the different operations, involved in data analytics in an interactive way.
4. It will help in discovering and unearthing useful information from a collection of data.
5. Discovering pattern in data that is knowledge discovery becomes easier.

CHAPTER TWO

SOFTWARE REQUIREMENTS SPECIFICATION

2. Software Requirement Specification

2.1. Introduction

2.1.1. Purpose

The purpose of this document is to give a detailed description of the requirements for the “Data Visionaire” software. It will illustrate the purpose and complete declaration for the development of system. It will also explain system constraints, interface and interactions with other external applications. This document is primarily intended to be a reference for developing the first version of the system for the development team.

2.1.2. Scope

Data Visionaire (Data Mining and Visualization in Virtual Reality) project is amalgamation of two profound technologies which is Data Analysis and Virtual Reality. In today's world we deal with complex and multidimensional data which can be cumbersome to represent and analyze. This multidimensional data is can be visualized in a way which is easy to interpret. Due to our limited perception towards multidimensional data, we require more intelligent environment to represent that data. Our goal is to create that environment in virtual reality and not only visualize complex data in 3D, 4D and 5D but also do OLAP operations like Roll-up, Drill-down, slice, dice, pivot and more.

2.1.3. Document Conventions

Table 2.1.1: Definitions, acronyms, and abbreviations

Term	Description
User	Someone who interacts with system using headset.
Observer	Someone who interacts with system using mobile application.
Headset	HTC Vive connected to PC.
Base Station	HTC Vive’s infrared base station trackers.

Controller	HTC Vive’s infrared controller wands.
PC	Any desktop or laptop capable of running HTC Vive.
System/Product	Virtual reality application, mobile application and web server.
VR	Virtual Reality environment.
Query	An operation done in Virtual Reality environment.
API/Web Server	An online application programming interface for visualization and analysis queries to fetch data.
OLAP	Online analytical processing.
Multidimensional Data	Data with more than two dimensions.
3D	Three Dimensional
Unity	The rendering engine and IDE where the VR scene is generated and visualized.
Scene	An instance of VR environment.
Play Area	The area in which the user stands.
Steam VR	A VR application for hardware and software interaction.

2.1.4. Overview

The remainder of this document includes two chapters. The second one provides an overview of the system functionality and system interaction with other systems. This chapter also introduces different types of users and their interaction with the system. Further, the chapter also mentions the system constraints and assumptions about the product.

The third chapter provides the requirements specification in detailed terms and a description of the different system interfaces. Different specification techniques are used in order to specify the requirements more precisely for different audiences.

2.2. Overall Description

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of users that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

2.2.1. Product Perspective

The product main objective is to provide a tool for Data Analysis and Visualization in Virtual Reality, to get better understanding of the data. User can also make prediction using regression technique which can help making users better decisions.

In addition to VR application, a mobile application is also developed for on the go reference. Mobile application provides the history of all the queries executed by the user with headset. Moreover, user can reference the graphs and query results in mobile application as well.

Both mobile and VR application fetches data from an online application programming interface. This API provides the developer with necessary information for visualization and analysis.

Furthermore, mobile application fetches data from Firebase an online database service that provides real time indexing in database for queries executed in Virtual Reality.

2.2.1.1. System Interfaces

NGROK will be used as web server. The user inputs data via the headset to make https queries. The actual program that will perform the operations is written in Unity Engine using C# as primary language and for web server Python as primary language.

2.2.1.2. User Interfaces

The new system shall provide a very intuitive and simple interface to the user and the observer, so that the user can easily navigate through different functions for visualization and analysis and the observer can easily queries performed for on the go reference.

2.2.1.3. Hardware Interfaces

The hardware requirements for user are as follows:

Table 2.2.1: Hardware Requirements

Component	**Recommended System Requirement**	**Minimum System Requirement**
Processor	Intel Core i5-4590/AMD FX 8350 equivalent or better	Intel Core i5-4590/AMD FX 8350 equivalent or better
GPU	NVIDIA GeForce GTX 1060, AMD Radeon RX 480 equivalent or better	NVIDIA GeForce GTX 970, AMD Radeon R9 290 equivalent or better
Memory	4 GB RAM or more	4 GB RAM or more
Video Output	HDMI 1.4, DisplayPort 1.2 or newer	HDMI 1.4, DisplayPort 1.2 or newer
USB	1x USB 2.0 or newer	1x USB 2.0 or newer
Operating System	Windows 7 SP1, Windows 8.1 or later, Windows 10	Windows 7 SP1, Windows 8.1 or later, Windows 10

The hardware requirements for observer are as follows:

- Any Android phone having android version above KitKat.

2.2.1.4. Software Interfaces

- **TECHNICAL LANGUAGES SUPPORT:** C#, Python, Java, XML, JSON
- **TOOLS/FRAMEWORKS:** Unity, NGROK service for python server, Android Studio, Firebase, Visual Studio Code, Adobe Photoshop, Blender

2.2.1.5. Communications Interfaces

The HTTP or HTTPS protocol(s) will be used to facilitate communication between the user and observer.

2.2.1.6. Operations

The product shall have operations to protect the database from being corrupted or accidentally altered during a system failure.

2.2.1.7. Operating Environment

Room-scale setup needs a minimum play area of 2 m x 1.5 m (6 ft 6 in x 5 ft). Examples of room-scale setup:

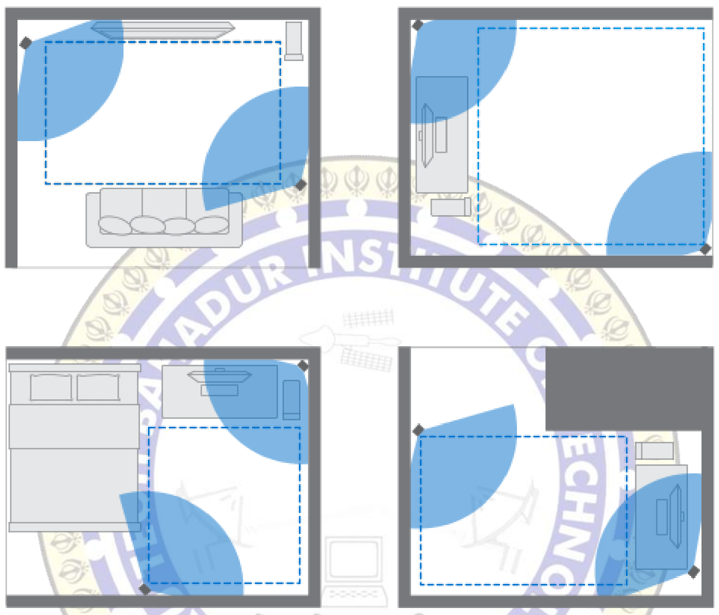


Figure 2.2.1: HTC Vive Room-scale setup [5]

Seated and standing experiences do not have space requirements. Examples of seated/standing setup:

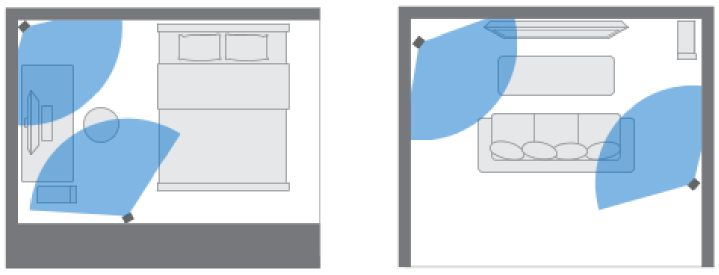


Figure 2.2.2: HTC Vive Seated/Standing setup [5]

2.2.2. Product Functions

This project aims to develop a Multidimensional Data Analytical Tool, which will increase the perception of user towards multidimensional data. The prominent features of this tool will be the following:

- Visualization of OLAP operations: This feature allows user to visualize data in hypercubes and also make OLAP operations on this data. E.g.: Drill Down, Roll Up.
- Various data analytics operations: In this user will be able to see clustering happening live in front of them in 3D which helps them understand the multidimensional data better and purpose of clustering in Data Analysis.
- Representation of multiple dimensions of data: In this user can represent not only 2D but 3D data in Virtual Reality. That means user can submit multidimensional query in which data variants are more than 3 sometimes.
- Bookmarking the previous operations: A user can bookmark any data query which they did in past. The virtual reality headset and the android application will be synced with each other due to which the user will be able to see the query history.
- Inferences and Predictions: User is presented with deduced inferences based on the analysis of data of previous years with the help of human expertise. Along with that functionality of prediction of data using clustering technique is also available for user. A user can also use our model to change his growing patterns based on the changing weather conditions.
- Similar functionality in Android: User can access all this data in Android App as well. The user will be able to visualize the analysis graphs in mobile app also.

2.2.3. User Characteristics

There are two types of users that interact with the system: users of the mobile application (Observers) and user with headset. Each of these types of users has different use of the system so each of them has their own requirements.

The mobile application users can only use the application to view the query executed by the user with headset. This means that the user has the ability to maintain the record of queries executed by user with headset and view them for on the go reference. In order for the users to get a relevant result there are multiple criteria the users can specify and can get the results relevantly.

The other user interacts with headset. The user can operate on the system to execute different visualization and analysis queries. The administrator can manage the information for each query as well as updates the mobile user's query history.

2.2.4. Constraints

2.2.4.1. User Interface Constraints

Using this system is fairly simple and intuitive. A user familiar with basic VR environment navigation skills should be able to understand all functionality provided by the system.

2.2.4.2. Hardware Constraints

The system should work on most home desktop and laptop computers which contain a VR supported GPU.

2.2.4.3. Software Constraints

The system will be intended to run on Windows 7 and above operating systems. Moreover, the system is constrained by the Internet connection. Since the application fetches data from the API over the Internet, it is crucial that there is an Internet connection for the application to function.

2.2.4.4. Operating Environment Constraints

Base Stations should be at minimum of 7ft height.

2.2.4.5. Design Standards Compliance

The system shall be implemented in Unity Engine with C# as primary language. Python is used for creating API and first phase data cleaning.

2.2.5. Assumptions and dependencies

Sole assumption about the product is that it will always be used on mobile phones and PCs that have enough performance. If the phone and PC does not have enough hardware resources available for the application, for example the users might have allocated them with other applications, there may be scenarios where the application does not work as intended or even at all.

2.2.6. Apportioning of requirements

In the case that the project is delayed, there are some requirements that could be transferred to the next version of the application. For example, integration of OLAP features might be apportioned to future versions

2.2.7. User Documentation

The following documents are to be delivered along with the software:

- A HTC Vive user manual.
- A Data Visionaire installation and maintenance manual.

2.3. Specific Requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

2.3.1. External interfaces

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

2.3.1.1. User Interfaces

Mobile application is divided into 3 modules. First is module of bookmarked queries, it is a list of all those queries which a user had bookmarked while viewing data in VR headset. Second module is of Analysis, which consist of graphs of analyzed data, user can easily compare and analyze data using different graphs in application. Third module is of live queries in which user can find list of crops and selecting any one the locations where that crop can be grown shows up.

2.3.1.2. Software Interfaces

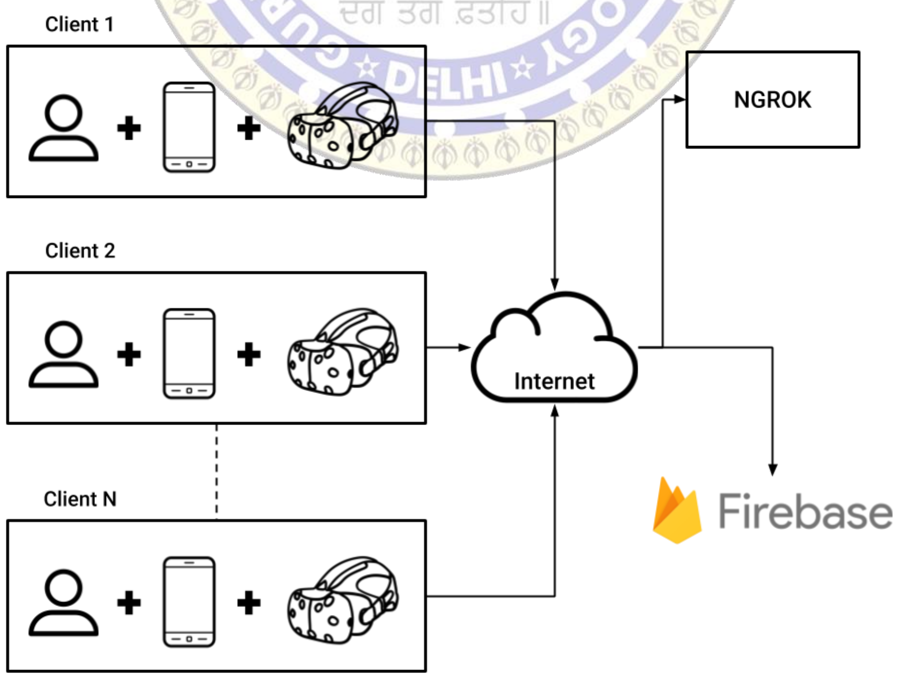


Figure 2.3.1: System Architecture of Data Visionaire

2.3.1.3. Software Data Flow Diagrams

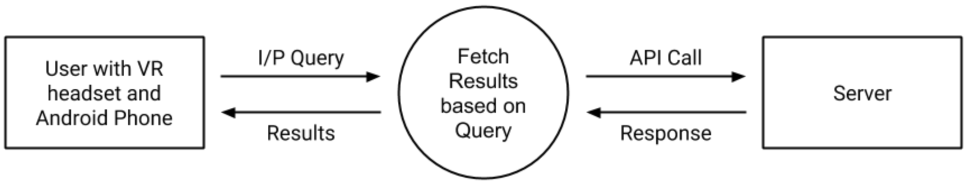

User with VR headset and Android Phone
I/P Query
Results
Fetch Results based on Query
API Call
Response
Server

Figure 2.3.2: Level 0 Data Flow Diagram of Data Visionaire

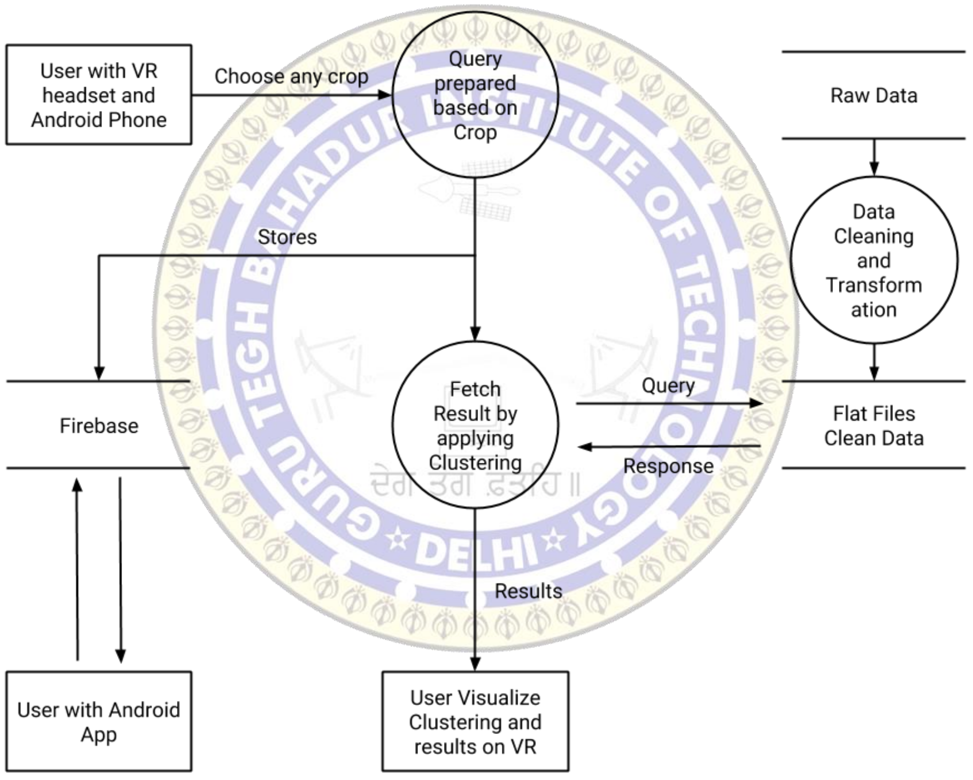

User with VR headset and Android Phone
Choose any crop
Query prepared based on Crop
Raw Data
Data Cleaning and Transform ation
Stores
Firebase
Fetch Result by applying Clustering
Query
Response
Flat Files Clean Data
Results
User with Android App
User Visualize Clustering and results on VR

Figure 2.3.3: Level 1Data Flow Diagram of Data Visionaire

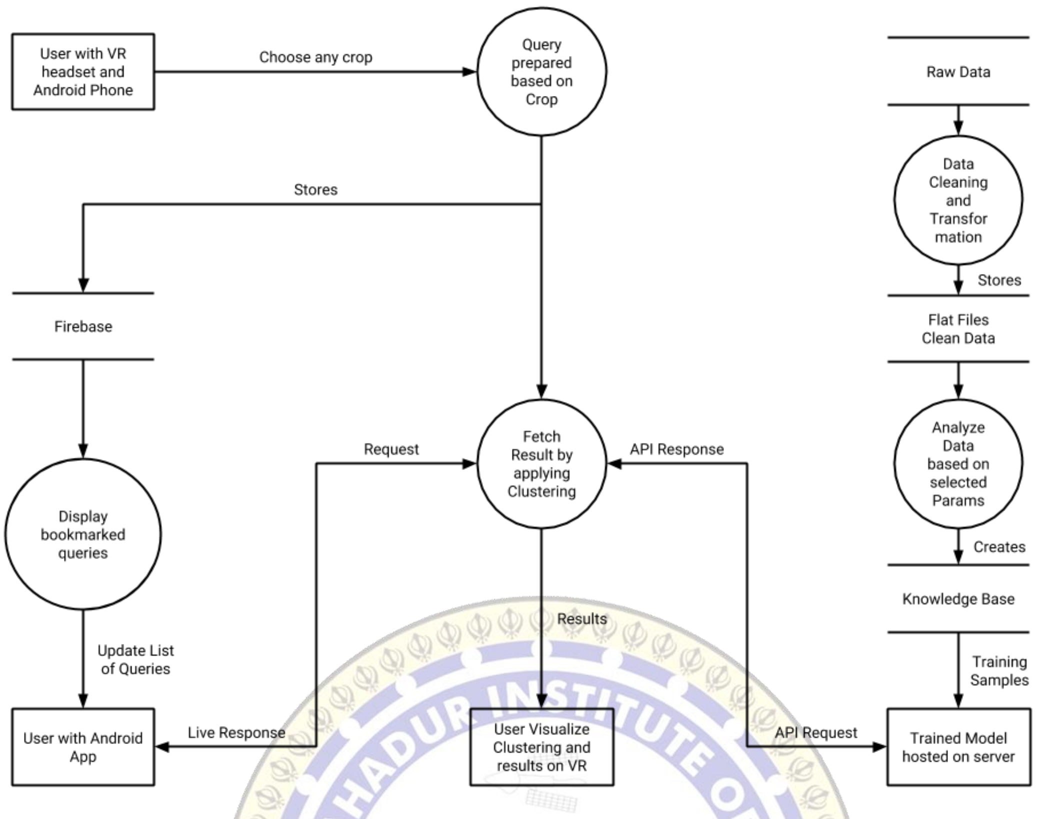

User with VR headset and Android Phone
Choose any crop
Query prepared based on Crop
Raw Data
Stores
Data Cleaning and Transfor mation
Stores
Firebase
Flat Files Clean Data
Analyze Data based on selected Params
Request
Fetch Result by applying Clustering
API Response
Display bookmarked queries
Creates
Knowledge Base
Results
Update List of Queries
Training Samples
User with Android App
Live Response
User Visualize Clustering and results on VR
API Request
Trained Model hosted on server

Figure 2.3.4: Level 2 Data Flow Diagram of Data Visionaire

2.3.1.4. Entity Relation Diagram

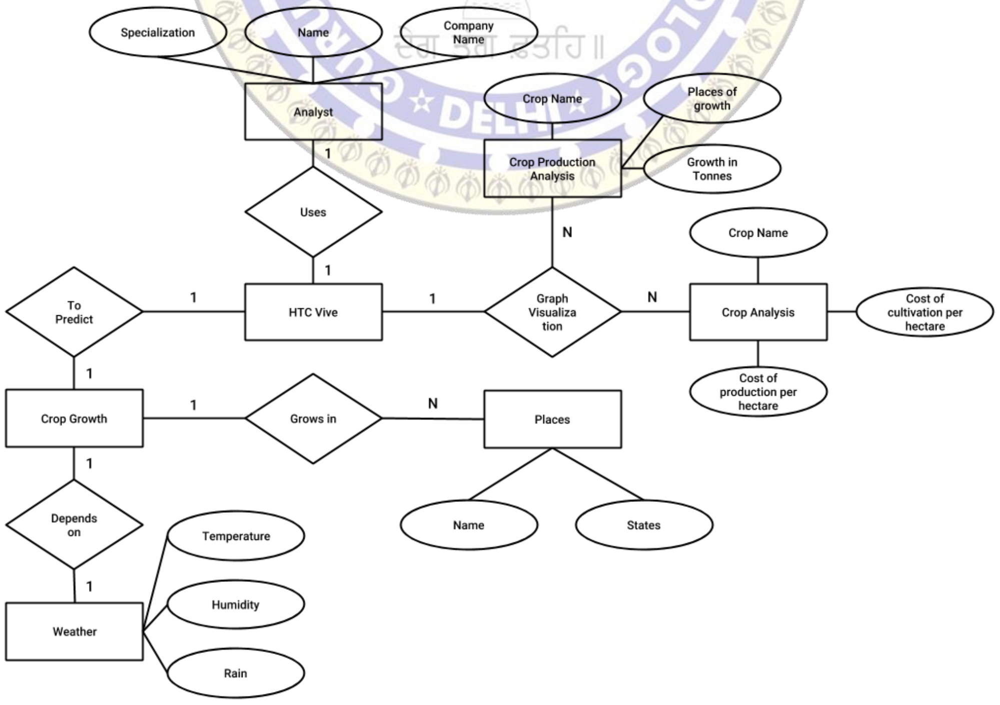


Figure 2.3.5: Entity Relation Diagram of Data Visionaire

2.3.1.5. Hardware Interfaces

The headset users interact with the system through headset and controllers.

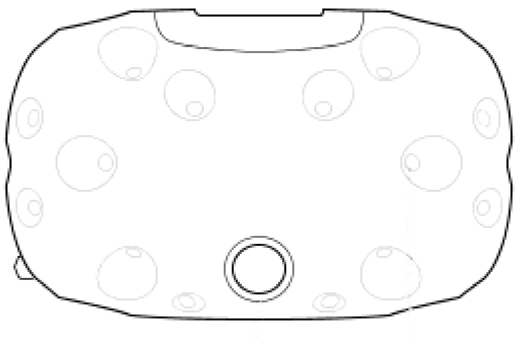


Figure 2.3.6: HTC Vive Headset Front [6]

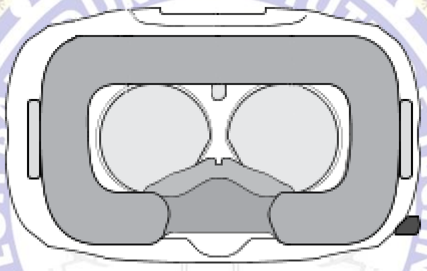


Figure 2.3.7: HTC Vive Headset Back [6]

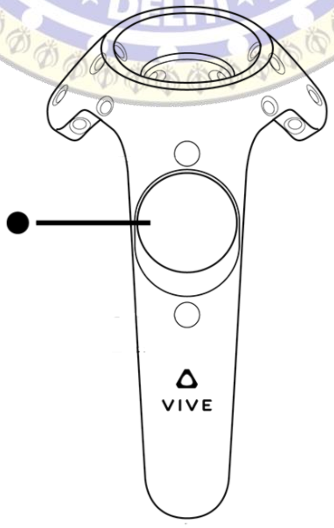


Figure 2.3.8: HTC Vive Controller Front with Touchpad denoted

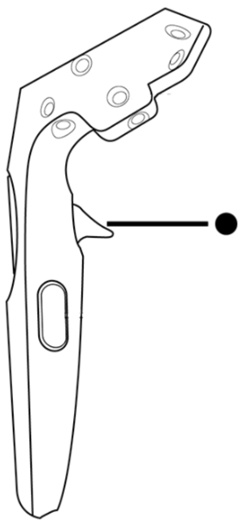


Figure 2.3.9: HTC Vive Side with Trigger denoted

2.3.1.6. Software Interfaces

The mobile and VR application communicates with the Internet in order to get query information.

The VR application in addition to Internet interacts with Steam VR application to establish connection between hardware and software. Steam VR application provides all the real time information about the user location and interactions with the system.

2.3.1.7. Communications Interfaces

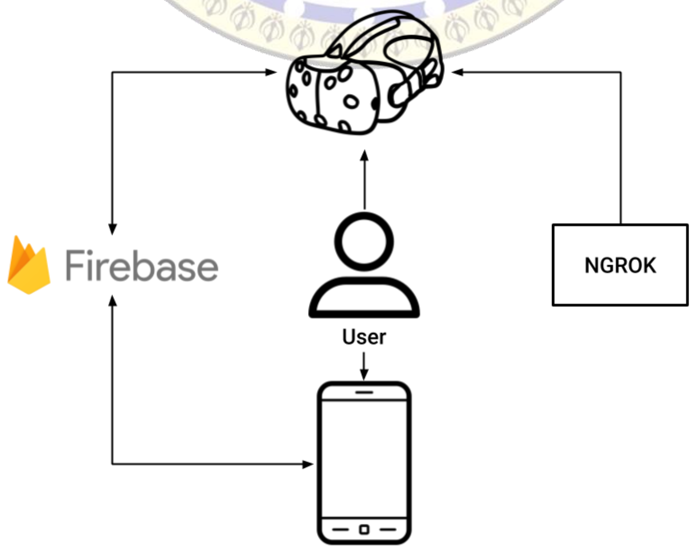


Figure 2.3.10: Communication between various modules of Data Visionaire

2.3.2. Functions

2.3.2.1. Use Case Diagram

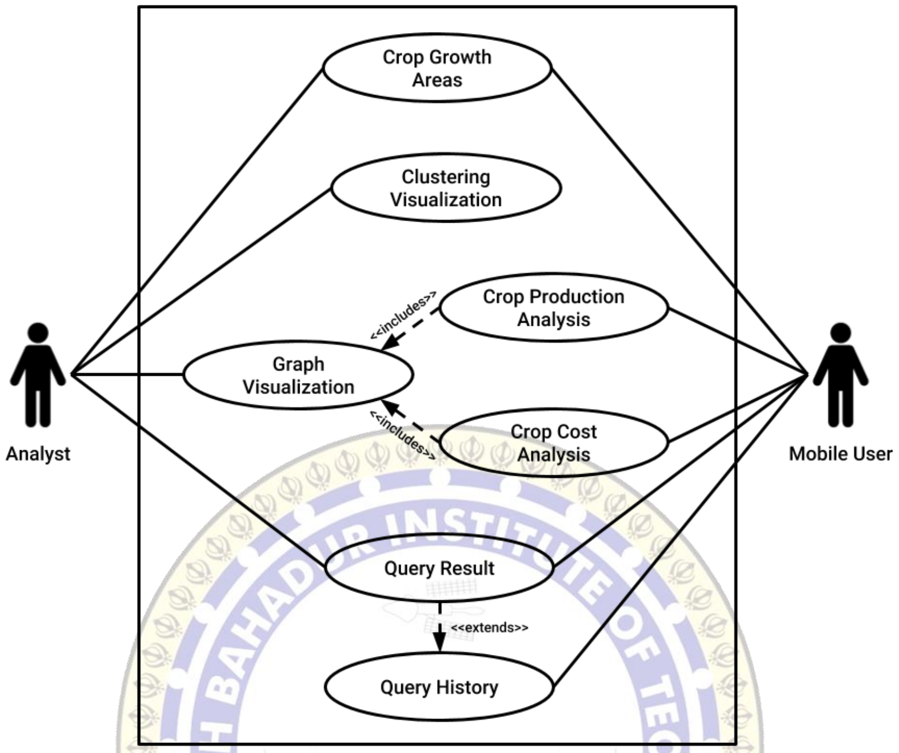


Figure 2.3.11: Use Case Diagram of Data Visionaire

2.3.3. Performance requirements

2.3.3.1. Latency

Latency is currently an issue with virtual reality devices. However, the user should experience as little latency as possible to ensure accuracy.

2.3.3.2. Graphics

The user should experience smooth gameplay, animations and interactions when using the VR application. The minimum graphics requirement for smooth gameplay is 90 FPS.

2.3.3.3. Simulator sickness

This is similar to motion sickness and may cause nausea, sweating or vomiting. The product will help mitigate this by using correct camera calibration and distortion. Also, a warning label will be printed on the product to address this requirement.

2.3.4. Software system attributes

2.3.4.1. Understandability

Applications code and comments should be written descriptively. The names of classes, methods and variables should be self-descriptive. Methods and classes will be commented to detail the purpose.

2.3.4.2. Completeness

All external libraries including their respective licenses should be documented.

2.3.4.3. Conciseness, Consistency and Efficiency

Application code will be reviewed regularly to remove redundancy, reduce complexity and increase efficiency.

2.3.4.4. Portability

Testing on multiple platforms (Windows 7, Windows 8, Windows 10) and Implementation should ensure that code and external libraries are not platform or implementation dependent.

2.3.4.5. Maintainability

Application code will be cohesive and have easily recognizable functionality Classes will be abstract enough to facilitate changes in data structures. Classes and function modularity should be implemented to avoid the need for major refactoring.

2.3.4.6. Reliability

Checked exception handling is highly recommended. Extensive testing is also required pending decision, information that is needed, conflicts awaiting resolution and the like.

CHAPTER THREE

TEST PLAN

3. Crop Wise Accuracy Percentage

Accuracy % = (correctly predicted class / total testing class)

1. Groundnut:

Correctly Predicted in: Gujarat, Haryana, Andhra Pradesh, Rajasthan, Maharashtra, Punjab, West Bengal

Accuracy = 7/10

Accuracy % = 70%

2. Gram:

Correctly Predicted in: Madhya Pradesh, Bihar, Haryana, Rajasthan, Maharashtra, Punjab, Uttar Pradesh, West Bengal

Accuracy = 8/10

Accuracy % = 80%

3. Mustard:

Correctly predicted in: Madhya Pradesh, Gujarat, Rajasthan, Punjab, Uttar Pradesh, West Bengal

Accuracy = 6/10

Accuracy % = 60%

4. Arhar:

Correctly Predicted in: Bihar, Gujarat, Haryana, Maharashtra, Punjab, Uttar Pradesh, West Bengal

Accuracy = 7/10

Accuracy % = 70%

5. Maize:

Correctly Predicted in: Gujarat, Haryana, Rajasthan, Bihar, Punjab, Uttar Pradesh, West Bengal

Accuracy = 7/10

Accuracy % = 70%

6. **Mung:**

Correctly Predicted in: Bihar, Haryana, Andhra Pradesh, Maharashtra, Punjab, West Bengal

Accuracy = 6/10

Accuracy % = 60%

7. **Cotton:**

Correctly Predicted in: Madhya Pradesh, Bihar, Gujarat, Rajasthan, Maharashtra, Uttar Pradesh, West Bengal

Accuracy = 7/10

Accuracy % = 70%

8. **Wheat:**

Correctly Predicted in: Bihar, Haryana, Andhra Pradesh, Rajasthan, Punjab, Uttar Pradesh, West Bengal

Accuracy = 7/10

Accuracy % = 70%

9. **Rice:**

Correctly Predicted in: Madhya Pradesh, Bihar, Haryana, Andhra Pradesh, Rajasthan, West Bengal

Accuracy = 6/10

Accuracy % = 60%

10. **Sugarcane:**

Correctly Predicted in: Madhya Pradesh, Bihar, Gujarat, Andhra Pradesh, Rajasthan, Maharashtra, Uttar Pradesh, West Bengal

Accuracy = 8/10

Accuracy % = 80%

CHAPTER FOUR

SOFTWARE AND HARDWARE DESCRIPTION

4. Description

4.1. Virtual Reality

Virtual Reality (VR), a technology that began in military and university laboratories more than 20 years ago, may be called Artificial Reality, Cyberspace, or Synthetic Reality. VR is a computer-created sensory experience that allows a participant to believe and barely distinguish a "virtual" experience from a real one. VR uses computer graphics, sounds, and images to reproduce electronic versions of real-life situations.

Virtual Reality is not a computer, but a technology that uses computerized clothing to synthesize reality. Most current VR systems provide only visual experiences created by computer-assisted design (CAD) or other graphics/animation systems, but researchers are working on interface devices that add sound and touch. Eventually, VR may be delivered through direct computer-to-brain connections.

Figure 4.1: User wearing HTC Vive Headset with Controllers [7]

4.1.1. How Does Virtual Reality Work?

A breakthrough in Virtual Reality came with the development of a head-mounted display with two tiny stereoscopic screens positioned just a few inches in front of the eyes. The most popular VR system is one designed by field pioneer, Jaron Lanier

(1989). The system features a head-mounted display called the EyePhone. Users also wear a DataGlove that generates movement and interaction in the virtual environment estimated system price: $205,000.

Movement in Cyberspace is simulated by shifting the optics in the field of vision in direct response to movement of certain body parts, such as the head or hand. Turn the head, and the scene shifts accordingly. The sensation is like being inside an artificial world the computer has created.

The EyePhone uses a set of wide-angle optics that cover approximately 140 degrees, almost the entire horizontal field of view. As the user moves his head to look around, the images shift to create an illusion of movement. The user moves while the virtual world is standing still. The glasses also sense the user's facial expressions through embedded sensors, and that information can control the virtual version of the user's body.

A group at NASA developed a system of helmet, glove, and a monochrome three-dimensional reality. The DataGlove, a key interface device, uses position tracking sensors and fiber optic strands running down each finger, allowing the user to manipulate objects that exist only within the computer simulated environment.

When the computer "senses" that the user's hand is touching a virtual object, the user "feels" the virtual object. The user can pick up an object and do things with it just as he would do with a real object. The DataGlove's most obvious application will be in robotics, particularly in the handling of hazardous materials, or by astronauts to control robot repairers from the safety of a spaceship, or from a space station, or even from Earth.

4.1.2. Applications of Virtual Reality

Applications for VR are many. Surgeons may soon use VR to "walk" through the brain or rehearse a surgical operation on a virtual patient. Just as flight simulators are now an integral part of pilot training, so surgical simulators will revolutionize medical training.

VR now makes possible telepresence, scientific exploration, and discovery. For example, the Jason Project for school children features both telepresence (the feeling of being in a location other than one's actual location) and teleoperation (controlling a robot submarine) (McLellan, 1995). The Jason Project, now in its sixth year, was designed to generate excitement about studying science, mathematics, and technology. NASA has a telepresence educational program that uses the Telepresence-controlled Remotely Operated underwater Vehicle (TROV) deployed in Antarctica. By means of distributed computer control architecture developed at NASA, school children in classrooms across the United States can take turns driving the TROV in Antarctica.

Someday scientists expect to explore celestial bodies and check out lakes beneath the Antarctic ice pack using VR applications. Disabled persons, through prosthetic interfaces, may one day use telerobotics to do tasks that are now only a dream; three-D sound may one day provide great applications for the blind.

Whether VR can be an effective tool for education or training depends partly on one's definition of VR and partly on one's goal for the educational experience. It may not be worth the cost if the goal of the educational experience is simply to memorize facts. However, if the goal of the educational experience is to foster excitement about a subject, or to encourage learning through exploration.

Figure 4.2: Windows Mixed Reality showing Virtual environment [8]

4.2. Unity

Unity is a cross-platform game engine developed by Unity Technologies and used to develop video games for PC, consoles, mobile devices and websites. First announced only for OS X, at Apple's Worldwide Developers Conference in 2005, it has since been extended to target 21 platforms. Nintendo provides free licenses of Unity 5 to all licensed Nintendo Developers along with their software development kits (SDKs) for the Wii U and Nintendo 3DS Family.

Five major versions of Unity have been released. At the 2006 WWDC show, Apple named Unity as the runner up for its Best Use of Mac OS X Graphics category.

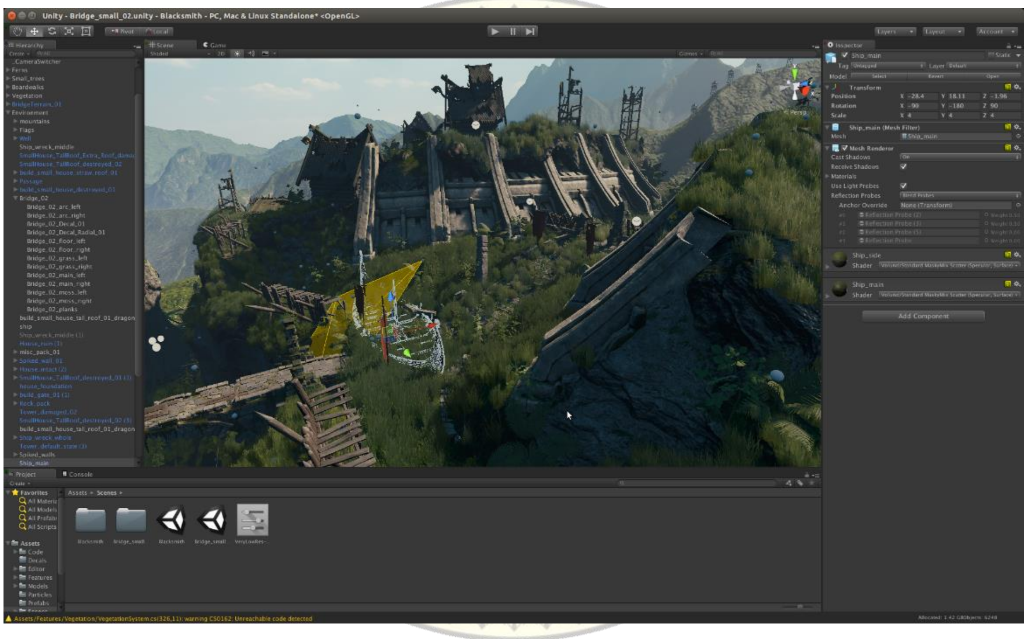

Unity - Bridge_small_02.unity - Blacksmith - PC, Mac & Linux Standalone* <OpenGL>

Figure 4.3: Unity Engine Editor window [9]

4.2.1. Overview

With an emphasis on portability, the engine targets the following APIs: Direct3D on Windows and Xbox 360; OpenGL on Mac, Linux, and Windows; OpenGL ES on Android and iOS; and proprietary APIs on video game consoles. Unity allows specification of texture compression and resolution settings for each platform that the game engine supports, and provides support for bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects. Unity's graphics engine's platform diversity can provide a shader with multiple variants and a declarative fallback specification, allowing Unity to detect the best variant for the current video hardware and, if none are compatible, to fall back to an alternative shader that may sacrifice features for performance.

4.2.2. Features

- Rich & Extensible Editor

  The Unity Editor includes multiple tools out in-on-editor of the box to support rapid editing and iteration in your development cycles with Unity’s instant play mode.

  Available on Windows and Mac, it includes a range of both artist friendly tools for designing immersive experiences and game world as strong suite of developer tools for implementing game logic and high end performing gameplay.

- Art & Design tools as well as the Unity Editor is a creative hub for artists, designers, developers, and other team members. It includes 2D and 3D scene design tools, storytelling and cinematics, lighting, audio system, Sprite management tools, particle effects and a powerful dope sheet animation system.
- Graphics Rendering

  Bring your game alive in the day with sun shafts or take your players down midnight streets glowing with neon signs or into shadowy tunnels.

  - Real-time rendering engine: Produce amazing visual fidelity with Real-Time Global Illumination and Physically Based Rendering.
  - Native Graphics APIs: Unity supports multi platforms, but still stays close to the low-level graphics API of each platform, allowing you to take advantage of the latest GPU and hardware improvements, like Vulkan, iOS Metal, DirectX 12, NVidia VR Works or AMD LiquidVR.
- Engine Performance

Optimize your interactive creations with a top performing engine that keeps on improving.

  - Advanced profiling tools offer insights, such as determining if your game is CPU or GPU-bound, and how to optimize rendering and gameplay performance for a smooth user experience.
  - Native C++ performance across platforms with Unity-developed back-end IL2CPP scripting, which is continuously improved.

- Platforms

  25+ platforms across mobile, desktop, console, TV, VR, AR and the Web.

  - More platform support than any other creation engine: With Unity, you can reach the widest audience and feel confident that your IP is future-proof, no matter how the industry evolves or where your imagination takes you.

- Virtual and Augmented Reality

  Unity is the preferred development tool for the majority of XR creators. Used by everyone from AAA game studios like Ubisoft, top creative agencies like Weiden + Kennedy, space pioneers at NASA, top Hollywood directors like Neill Blomkamp and Eric Darnell, and even our friends at Google for their Tiltbrush and Blocks experiences, Unity is your best and most valuable solution for jumping into the latest and greatest immersive technologies.

- Multiplayer
  - Quick set up: Fast to implement and highly customizable.
  - Unity-provided servers: Makes sure your players can find and play with each other.

- Team Collaboration

  Unity Teams enables creative teams to work more efficiently together with features that enable collaboration and simplify workflows.

  - Save, share, and sync your projects and use simple version control and cloud storage, all seamlessly integrated with Unity.

    - Cloud Build: Automatically create and share builds with anyone.
- Live Ops Analytics

  Unity Analytics gives you fast, easy access to important information that helps you improve your in-game economy and the player experience.

- Performance Reporting

  Unity Performance Reporting addresses issues in real-time.

    - Find and address the high priority issues that your customers are experiencing.
    - Collect and react to application errors, in real-time, across devices and platforms.

4.3. Blender

Blender is a professional free and open-source 3D computer graphics software product used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. Blender's features include 3D modeling, UV unwrapping, texturing, raster graphics editing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animating, match moving, camera tracking, rendering, video editing and compositing. It further features an integrated game engine.

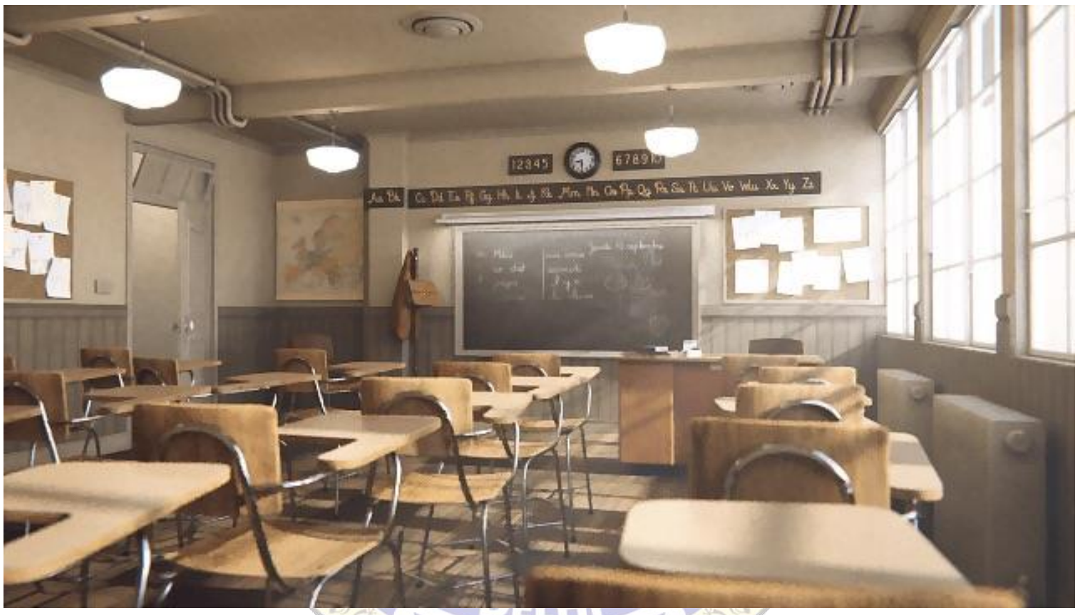


Figure 4.4: Classroom render in blender [10]

4.3.1. Overview

Blender has a wide variety of tools making it suitable for almost any sort of media production. People and studios around the world use it for hobby projects, commercials, feature films, games and other interactive applications like kiosks, games and scientific research.

Official releases of Blender for Microsoft Windows, Mac OS X and Linux, as well as a port for FreeBSD, are available in both 32-bit and 64-bit versions. Though it is often distributed without extensive example scenes found in some other programs, the software contains features that are characteristic of high-end 3D software.

4.3.2. Features

- Support for a variety of geometric primitives, including polygon meshes, fast subdivision surface modeling, Bezier curves, NURBS surfaces, metaballs, ico-spheres, multi-res digital sculpting (including dynamic topology, maps baking, remeshing, re-symmetrize, decimation), outline font, and a new n-gon modeling system called B-mesh.
- Internal render engine with scanline rendering, indirect lighting, and ambient occlusion that can export in a wide variety of formats.
- A path tracer render engine called Cycles, which can take advantage of the GPU for rendering. Cycles supports the Open Shading Language since Blender 2.65.
- Integration with a number of external render engines through plugins.
- Simulation tools for soft body dynamics including mesh collision detection, LBM fluid dynamics, smoke simulation, Bullet rigid body dynamics, ocean generator with waves.
- A particle system that includes support for particle-based hair.
- Modifiers to apply non-destructive effects.
- Python scripting for tool creation and prototyping, game logic, importing/exporting from other formats, task automation and custom tools.
- Basic non-linear video/audio editing.
- The Blender Game Engine, a sub-project, offers interactivity features such as collision detection, dynamics engine, and programmable logic. It also allows the creation of stand-alone, real-time applications ranging from architectural visualization to video games.
- Procedural and node-based textures, as well as texture painting, projective painting, vertex painting, weight painting and dynamic painting.
- Real-time control during physics simulation and rendering.
- Camera and object tracking.

4.4. Firebase

The Firebase Real Time Database is a cloud-hosted database. Data is stored as JSON and synchronized in real time to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Real Time Database instance and automatically receive updates with the newest data.

4.4.1. Key capabilities

- **Realtime:** Instead of typical HTTP requests, the Firebase Real Time Database uses data synchronization—every time data change, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.
- **Offline:** Firebase apps remain responsive even when offline because the Firebase Real Time Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any changes it missed, synchronizing it with the current server state.
- **Accessible from Client Devices:** The Firebase Real Time Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Real Time Database Security Rules, expression-based rules that are executed when data is read or written.

4.4.2. How it works

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, real time events continue to fire, giving the end user a responsive experience.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

4.5. Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on Intellij IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features.
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support

4.5.1. Project Structure

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

- manifests: Contains the AndroidManifest.xml file.
- java: Contains the Java source code files, including JUnit test code.
- res: Contains all non-code resources, such as XML layouts, UI strings.

RESULTS

Results

Based on our data set this consisted of four features: annual rainfall, humidity, soil type and temperature at any place in India we helped the farmers or government agencies to help them and predict which crops will be suitable for their agricultural land and the various geographical conditions so that they can maximize their profits.

We help the farmers by providing them with statistical outputs so that they can earn their livelihood and live in peace.

We provide them with two statistical outputs:

1. Crop production analysis: In crop production analysis we provide the farmers and agencies with historical data of crop productions over the 15 years' time span. After the clustering phase the farmers can judge depending upon their conditions at their disposal whether the growing of crop will be beneficial or not.
2. Cost Analysis: After the farmer has decided to grow one or more crops we further help the farmer in taking decisions based on various other factors like cost of growing that crop, cost of production and various other factors.

Based on clustering we found that the overall accuracy of our model is: 70%. On further drilling down the accuracy for each crop is as follows:

1. Groundnut: 70%
2. Gram: 80%
3. Mustard: 60%
4. Arhar: 70%
5. Maize: 70%
6. Mung: 60%
7. Cotton: 70%
8. Wheat: 70%
9. Paddy: 60%
10. Sugarcane: 80%

CONCLUSIONS AND SUMMARY

Conclusion

The application satisfies all provided specification

Summary

The application helps the farmers and government agencies to predict which crops to grow at their fields depending upon the various geographical conditions and suggests growing which crops will help them maximize their profits and this is further supplemented by various analysis like crop production analysis and cost analysis.

REFERENCE

[1] Ekaterina Olshannikova, Aleksandr Ometov, Yevgeni Koucheryavy, Thomas Olsson, ‘Visualizing Big Data with augmented and virtual reality: challenges and research agenda’, 2015. [Online]: https://link.springer.com/article/10.1186%2Fs40537-015-0031-2/. [Accessed: 15th February 2018].

[2] Arwa Michelle Mboya, ‘Data Visualization in Virtual Reality—A VR Demo Project’, 2017. [Online]: https://medium.com/inborn-experience/data-visualization-in-virtual-reality-a-vr-demo-project-a31c577aaefc. [Accessed: 3rd March 2018]

[3] Utkarsha P. Narkhede, K.P. Adhiya, ‘Evaluation of Modified K-Means Clustering Algorithm in Crop Prediction’ 2014. [Online]: https://pdfs.semanticscholar.org/2a16/57c09219f739abf08f6cf43520103a23dc7c.pdf [Accessed: 21st February 2018].

[4] Methodology blog, ‘Beginning Data Visualization in Unity: Scatterplot Creation’ [Online]: https://sites.psu.edu/bdssblog/2017/04/06/basic-data-visualization-in-unity-scatterplot-creation/. [Accessed: 4th March 2018].

[5] HTC Vive, ‘Planning your play area’, 2018. [Online]: https://www.vive.com/in/support/vive/category_howto/planning-your-play-area.html. [Accessed: 21st April 2018]

[6] HTC Vive, ‘About the headset’, 2018. [Online]: https://www.vive.com/in/support/vive/category_howto/about-the-headset.html. [Accessed: 21st April 2018]

[7] Adi Robertson, ‘HTC VIVE REVIEW Some of the best ideas in VR, but not the best execution’, 2016. [Online]: https://www.theverge.com/2016/4/5/11358618/htc-vive-vr-review. [Accessed: 21st April 2018]

[8] Terry Myerson, ‘Opening Windows Holographic to Partners for a New Era of Mixed Reality’, 2016. [Online]: https://blogs.windows.com/windowsexperience/2016/06/01/opening-windows-holographic-to-partners-for-a-new-era-of-mixed-reality/. [Accessed: 21st April 2018].

[9] Na'Tosha Bard, ‘The State of Unity on Linux’, 2015. [Online]: https://blogs.unity3d.com/2015/07/01/the-state-of-unity-on-linux/. [Accessed 21st April 2018]

[10] Johnson Martin, '7 mistakes Blender users make when trying to render faster.', 2017. [Online]: https://blendergrid.com/news/7-mistakes-when-trying-to-render-faster. [Accessed: 21st April 2018].

[11] Marko Teräs, Shriram Raghunathan 'Big Data Visualisation in Immersive Virtual Reality Environments: Embodied Phenomenological Perspectives to Interaction', 2015. [Online]: https://www.researchgate.net/publication/280640341_Big_Data_Visualisation_in_Immersive_Virtual_Reality_Environments_Embodied_Phenomenological_Perspectives_to_Interaction. [Accessed: 18th March 2018].

[12] Sharmistha Mandal, 'Brief Introduction of Virtual Reality & its Challenges', 2013. [Online]: https://www.ijser.org/researchpaper/Brief-Introduction-of-Virtual-Reality-its-Challenges.pdf. [Accessed: 6th March 2018].

[13] SAS 'Data Visualization Techniques', 2017 [Online]: https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/data-visualization-techniques-106006.pdf. [Accessed: 13th March 2018].

[14] 'Unity 2017', 2017. [Online]: https://unity3d.com/unity.

[15] 'Blender', 2017 [Online]: https://www.blender.org/about/

[16] 'Firebase', 2017. [Online]: https://firebase.google.com/docs/

APPENDIX A

SCREENSHOTS

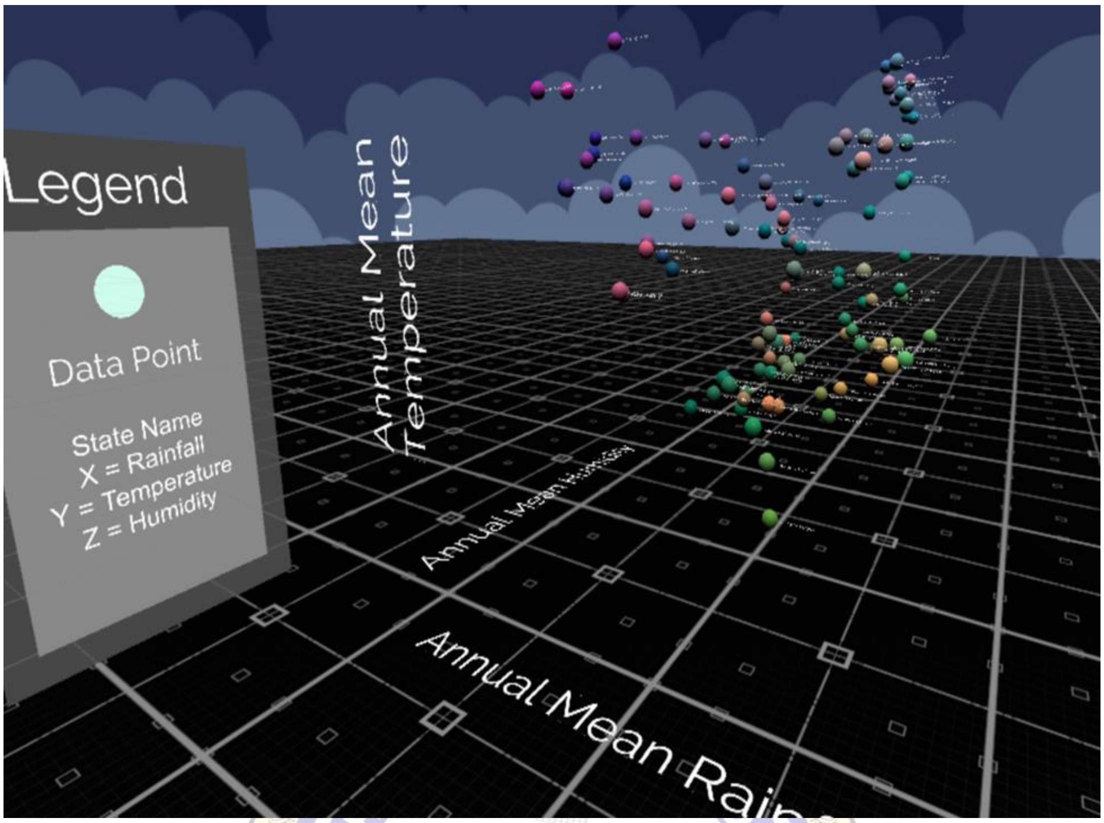


Figure A.1: Scatterplot of rain, temperature and humidity

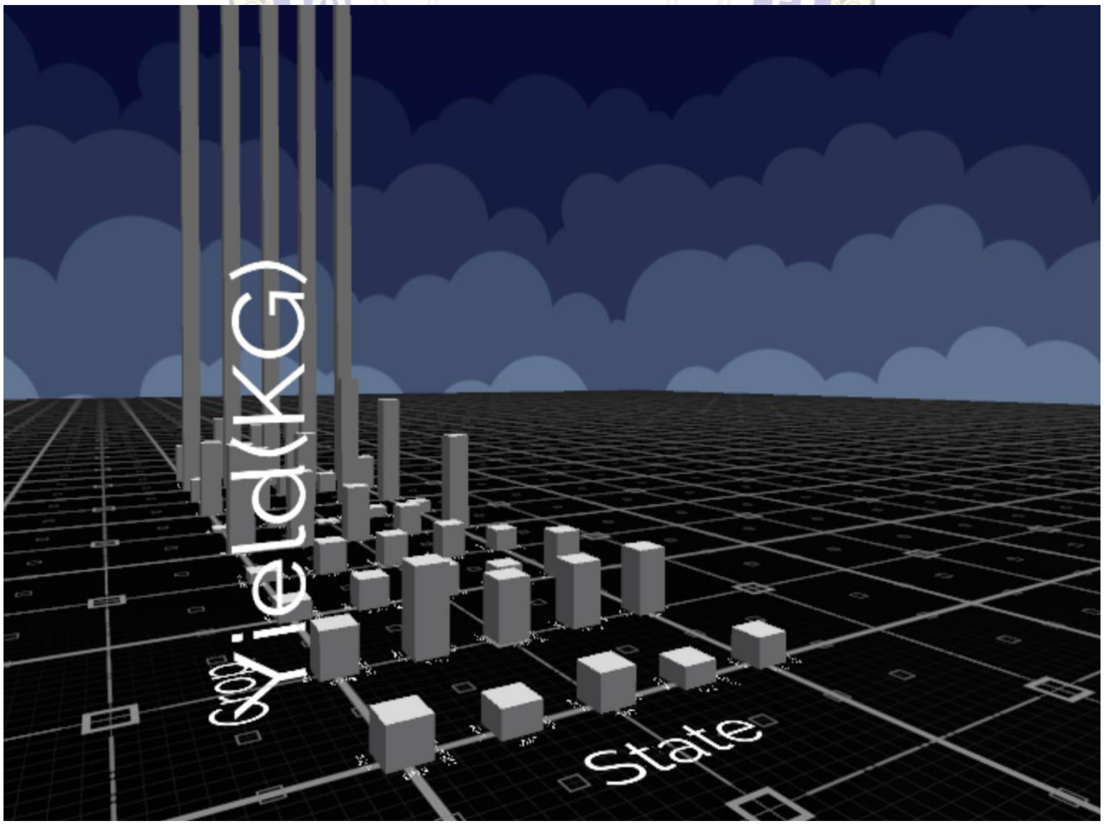


Figure A.2: Bar Graph of crop, state and yield in KG

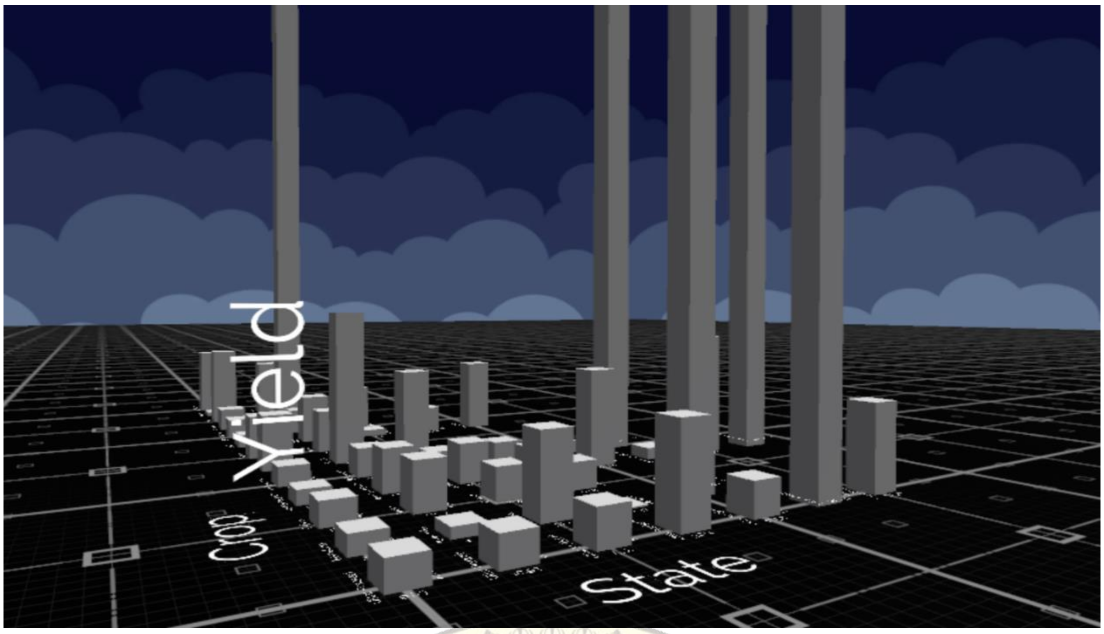


Figure A.3: Bar Graph of crop, state and yield

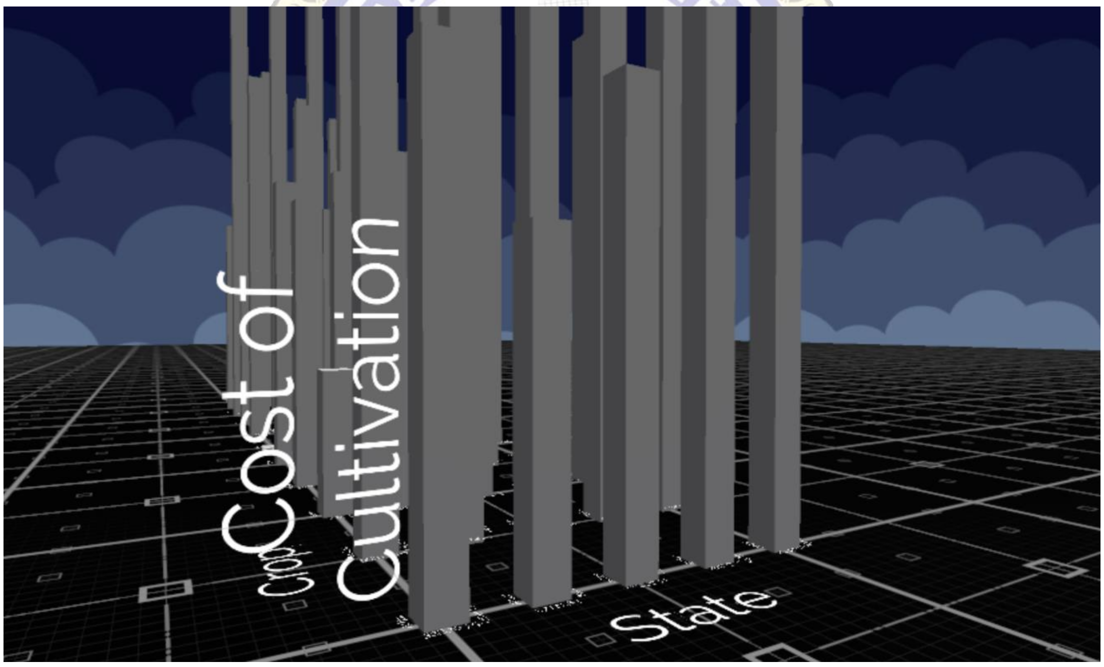


Figure A.4: Bar Graph of crop, state and cost of cultivation

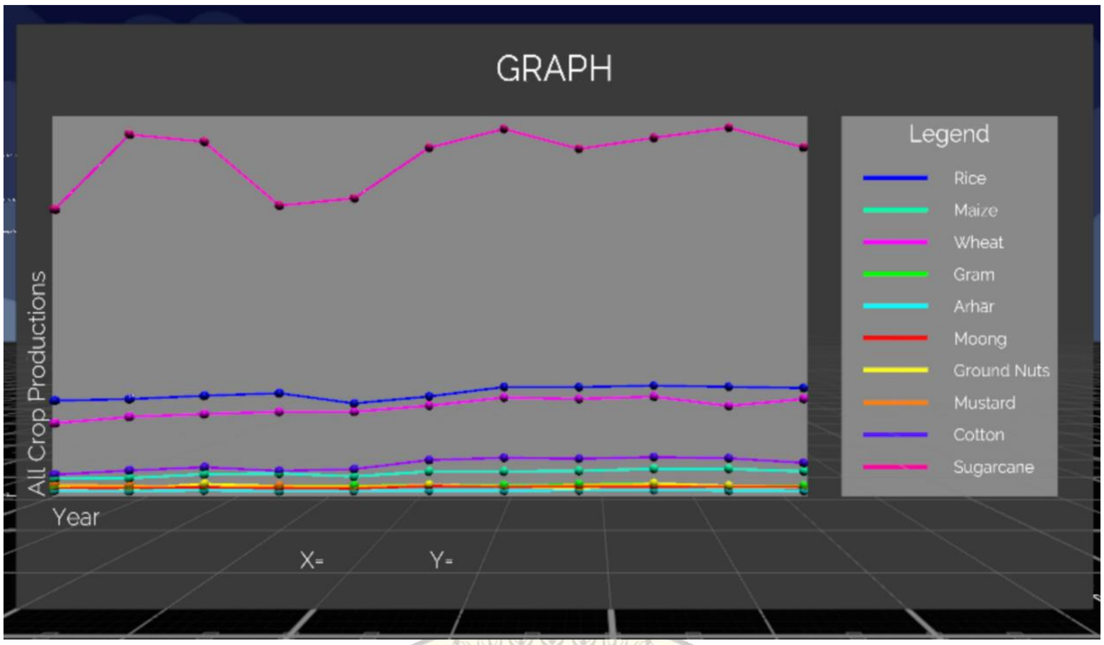


Figure A.5: Line Graph of all crop production and year

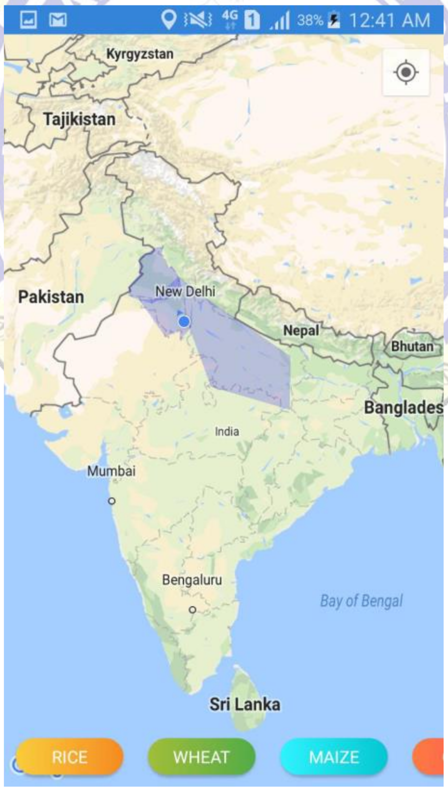


Figure A.6: Places of growth marked on map of India

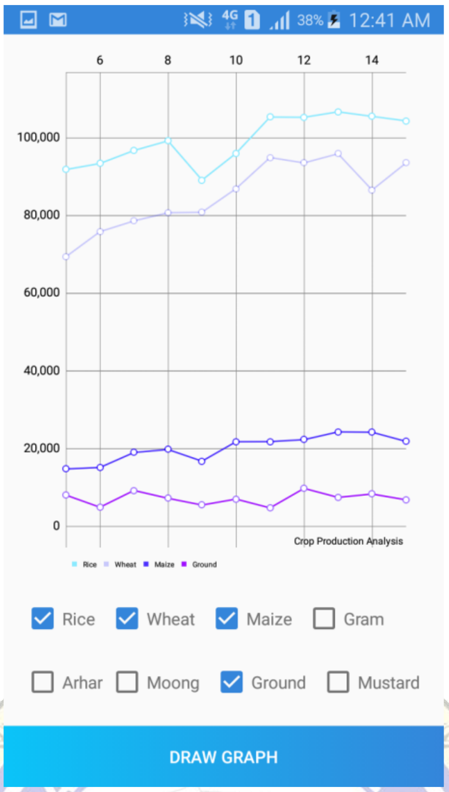


Figure A.7: Line Graph on mobile application

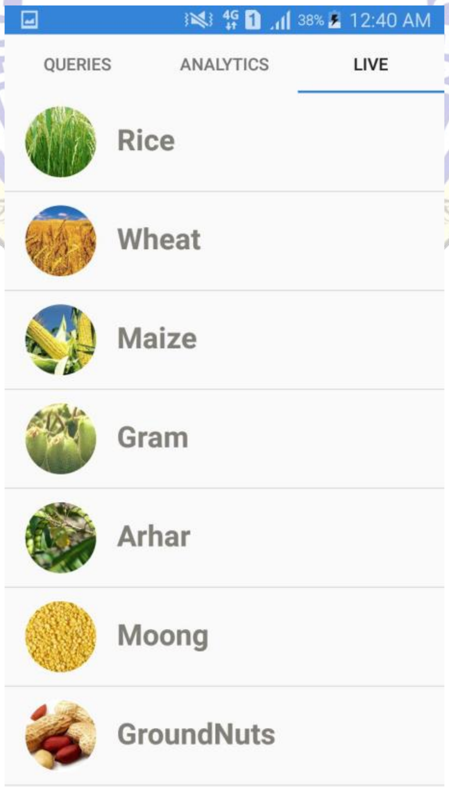


Figure A.8: List of crops on mobile application

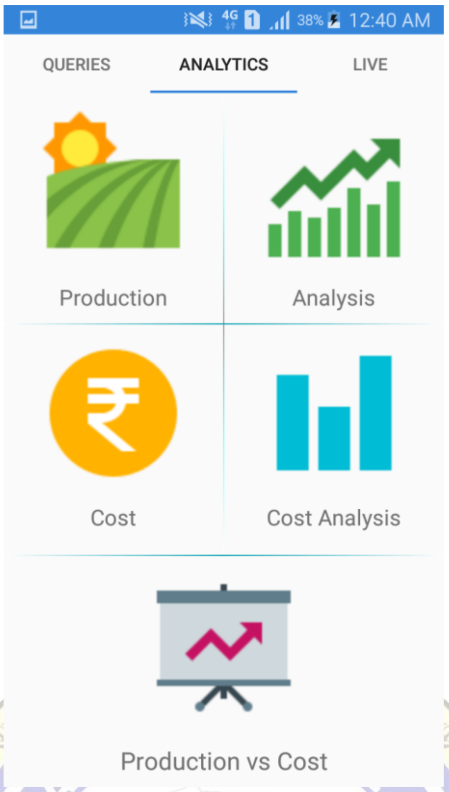
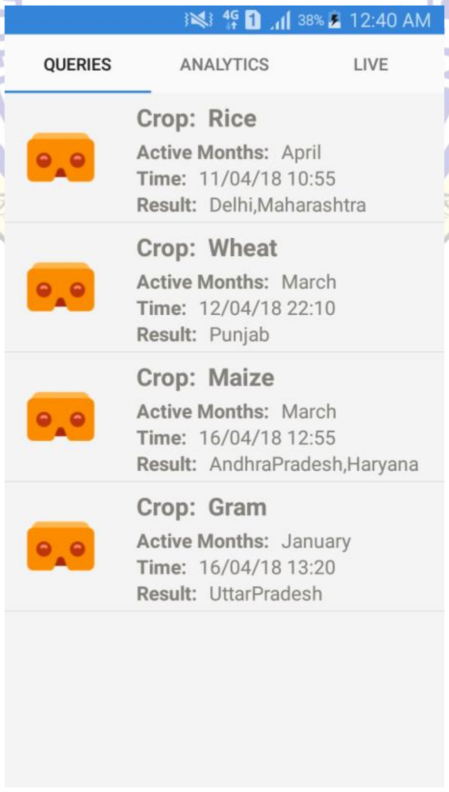


Figure A.9: List of queries available on mobile application



Crop: Rice
Active Months: April
Time: 11/04/18 10:55
Result: Delhi,Maharashtra

Crop: Wheat
Active Months: March
Time: 12/04/18 22:10
Result: Punjab

Crop: Maize
Active Months: March
Time: 16/04/18 12:55
Result: AndhraPradesh,Haryana

Crop: Gram
Active Months: January
Time: 16/04/18 13:20
Result: UttarPradesh

Figure A.10: History of queries on mobile application

APPENDIX B

SOURCE CODE

Python Files

ClusterAPI.py

```
import numpy as np
import pandas as pd
import math
import numpy
from cotton_cluster import *
from sugarcane_cluster import *
from arhar_cluster import *
from maize_cluster import *
from mung_cluster import *
from gram_cluster import *
from groundnut_cluster import *
from paddy_cluster import *
from mustard_cluster import *
from wheat_cluster import *
from flask import Flask,jsonify,request
import json

#Initialize flask app variable
app=Flask(__name__)

#imports of data
csv_file_punjab=pd.read_csv('Punjab.csv')
csv_file_bihar=pd.read_csv('Bihar.csv')
csv_file_gujarat=pd.read_csv('gujarat.csv')
csv_file_haryana=pd.read_csv('Haryana.csv')
csv_file_hyderabad=pd.read_csv('hyderabad.csv')
csv_file_jaipur=pd.read_csv('jaipur.csv')
csv_file_mumbai=pd.read_csv('Mumbai.csv')
csv_file_west_bengal=pd.read_csv('westbengal.csv')
csv_bhopal=pd.read_csv('bhopal.csv')
csv_gujarat=pd.read_csv('gujarat.csv')
csv_up=pd.read_csv('Uttar Pradesh.csv')
```

```
def np_for_state(data):
	temp=data['Annual Mean Temperature']
	hum=data['Annual Mean Humidity']
	rainfall=data['Annual Mean Rainfall']
	s1=data['Soil_1']
	s2=data['Soil_2']
	s3=data['Soil_3']
	s4=data['Soil_4']
	s5=data['Soil_5']
	result=np.column_stack((temp,hum,rainfall,s1,s2,s3,s4,s5))
	return result
def dist(x,y):
	return numpy.sqrt(numpy.sum((x-y)**2))

punjab=np_for_state(csv_file_punjab)
bihar=np_for_state(csv_file_bihar)
haryana=np_for_state(csv_file_haryana)
hyderabad=np_for_state(csv_file_hyderabad)
jaipur=np_for_state(csv_file_jaipur)
mumbai=np_for_state(csv_file_mumbai)
westbengal=np_for_state(csv_file_west_bengal)
bhopal=np_for_state(csv_bhopal)
gujarat=np_for_state(csv_gujarat)
up=np_for_state(csv_up)

#Centres and higher ranges
cotton_centre=np.array([[27.5,85,850,0,0,0,0,0]])
cotton_up=np.array([[25,80,600,0,0,0,0,0]])

sugarcane_centre=np.array([[34.5,65,1450,0,0,0,0,0]])
sugarcane_up=np.array([[32,40,700,0,0,0,0,0]])

arhar_centre=np.array([[25,65,850,0,0,0,0,0]])
arhar_up=np.array([[20,60,600,0,0,0,0,0]])

maize_centre=np.array([[22.5,65,850,0,0,0,0,0]])
maize_up=np.array([[18,60,600,0,0,0,0,0]])

mung_centre=np.array([[30,75,1000,0,0,0,0,0]])
```

```
mung_up=np.array([[25,70,850,0,0,0,0,0]])

gram_centre=np.array([[21.5,45,825,0,0,0,0,0]])
gram_up=np.array([[18,40,600,0,0,0,0,0]])

groundnut_centre=np.array([[22.5,65,950,0,0,0,0,0]])
groundnut_up=np.array([[20,60,700,0,0,0,0,0]])

paddy_centre=np.array([[23.5,65,1200,0,0,0,0,0]])
paddy_up=np.array([[20,60,1500,0,0,0,0,0]])

mustard_centre=np.array([[20,60,950,0,0,0,0,0]])
mustard_up=np.array([[22,70,600,0,0,0,0,0]])

wheat_centre=np.array([[21,60,500,0,0,0,0,0]])
wheat_up=np.array([[26,70,750,0,0,0,0,0]])

#Distances
cotton_distance=dist(cotton_centre,cotton_up)
sugarcane_distance=dist(sugarcane_centre,sugarcane_up)
arhar_distance=dist(arhar_centre,arhar_up)
maize_distance=dist(maize_centre,maize_up)
mung_distance=dist(mung_centre,mung_up)
gram_distance=dist(gram_centre,gram_up)
groundnut_distance=dist(groundnut_centre,groundnut_up)
paddy_distance=dist(paddy_centre,paddy_up)
mustard_distance=dist(mustard_centre,mustard_up)
wheat_distance=dist(wheat_centre,wheat_up)

print ('Cotton')
print
getCottonData(cotton_centre,cotton_distance,punjab,bihar,haryana,hyderabad,jaipur,mumbai,westben
gal,bhopal,gujarat,up)

print ('Sugarcane')
print
getSugarcaneData(sugarcane_centre,sugarcane_distance,punjab,bihar,haryana,hyderabad,jaipur,mumb
ai,westbengal,bhopal,gujarat,up)

print ('Arhar')
```

```
print
getArharData(arhar_centre,arhar_distance,punjab,bihar,haryana,hyderabad,jaipur,mumbai,westbengal,
bhopal,gujarat,up)

print ('Maize')
print
getMaizeData(maize_centre,maize_distance,punjab,bihar,haryana,hyderabad,jaipur,mumbai,westbeng
al,bhopal,gujarat,up)

print ('Mung')
print
getMungData(mung_centre,mung_distance,punjab,bihar,haryana,hyderabad,jaipur,mumbai,westbenga
l,bhopal,gujarat,up)

print ('Gram')
print
getGramData(gram_centre,gram_distance,punjab,bihar,haryana,hyderabad,jaipur,mumbai,westbengal,
bhopal,gujarat,up)

print ('Groundnut')
print
getGroundnutData(groundnut_centre,groundnut_distance,punjab,bihar,haryana,hyderabad,jaipur,mum
bai,westbengal,bhopal,gujarat,up)

print ('Paddy')
print
getPaddyData(paddy_centre,paddy_distance,punjab,bihar,haryana,hyderabad,jaipur,mumbai,westbeng
al,bhopal,gujarat,up)

print ('Mustard')
print
getMustardData(mustard_centre,mustard_distance,punjab,bihar,haryana,hyderabad,jaipur,mumbai,we
stbengal,bhopal,gujarat,up)

print('Wheat')
print
getWheatData(wheat_centre,wheat_distance,punjab,bihar,haryana,hyderabad,jaipur,mumbai,westbeng
al,bhopal,gujarat,up)
```

```
@app.route('/prediction')
def get_state_predictions():
        state=request.args.get('crop')
        if state=='cotton':
        result=getCottonData(cotton_centre,cotton_distance,punjab,bihar,haryana,hyderabad,jaipur,
        mumbai,westbengal,bhopal,gujarat,up)
        elif state=='sugarcane':
        result=getSugarcaneData(sugarcane_centre,sugarcane_distance,punjab,bihar,haryana,hydera
        bad,jaipur,mumbai,westbengal,bhopal,gujarat,up)
        elif state=='arhar':
        result=getArharData(arhar_centre,arhar_distance,punjab,bihar,haryana,hyderabad,jaipur,mu
        mbai,westbengal,bhopal,gujarat,up)
        elif state=='maize':
        result=getMaizeData(maize_centre,maize_distance,punjab,bihar,haryana,hyderabad,jaipur,m
        umbai,westbengal,bhopal,gujarat,up)
        elif state=='mung':
        result=getMungData(mung_centre,mung_distance,punjab,bihar,haryana,hyderabad,jaipur,mu
        mbai,westbengal,bhopal,gujarat,up)
        elif state=='gram':
        result=getGramData(gram_centre,gram_distance,punjab,bihar,haryana,hyderabad,jaipur,mu
        mbai,westbengal,bhopal,gujarat,up)
        elif state=='groundnut':
        result=getGroundnutData(groundnut_centre,groundnut_distance,punjab,bihar,haryana,hydera
        bad,jaipur,mumbai,westbengal,bhopal,gujarat,up)
        elif state=='paddy':
        result=getPaddyData(paddy_centre,paddy_distance,punjab,bihar,haryana,hyderabad,jaipur,m
        umbai,westbengal,bhopal,gujarat,up)
        elif state=='mustard':
        result=getMustardData(mustard_centre,mustard_distance,punjab,bihar,haryana,hyderabad,jai
        pur,mumbai,westbengal,bhopal,gujarat,up)
        elif state=='wheat':
        result=getWheatData(wheat_centre,wheat_distance,punjab,bihar,haryana,hyderabad,jaipur,m
        umbai,westbengal,bhopal,gujarat,up)
        else:
        result='Not a proper key'
        return jsonify(result)

if(__name__=='__main__'):
        app.run(host='127.0.0.1', port=8000)
```

C# Files

ControllerManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ControllerManager : MonoBehaviour {

        #region SteamVR variables
        public SteamVR_TrackedObject trackedObject;
        public SteamVR_Controller.Device device;
        #endregion

        #region Teleportation variables
        public LineRenderer laser;
        public GameObject teleportAimer;
        Vector3 teleportLocation;
        public GameObject Player;
        public LayerMask laserMask;
        public float yNudge = 1f;
        public bool isLeftHanded;
        #endregion

        public Material day, night;

        [Header ("Script Refrences")]
        public DataPlotter dataPlotter;
        public Clustering clustering;
        public CropSelector cropSelector;
        public DataGraphProduction menuPloter;
        public ButtonController buttonController;
        public DataGraphCostAnalysis costAnalysis;

        public Toggle Arhar, Cotton, Gram, Groundnut, Maize, Mung, Paddy, Mustard, Sugarcane,
Wheat;

        public GameObject India;
```

```
	void Start () {
		trackedObject = GetComponent<SteamVR_TrackedObject> ();
	}

	void Update () {
		device = SteamVR_Controller.Input ((int) trackedObject.index);
		if (isLeftHanded) {
			if (device.GetTouch (SteamVR_Controller.ButtonMask.Touchpad)) {

				laser.gameObject.SetActive (true);
				laser.SetPosition (0, gameObject.transform.position);
				RaycastHit hit;

				if (Physics.Raycast (transform.position, transform.forward, out hit,
10, laserMask)) {
					teleportAimer.SetActive (true);
					teleportLocation = hit.point;
					laser.SetPosition (1, teleportLocation);
					teleportAimer.transform.position	=	new	Vector3
(teleportLocation.x, teleportLocation.y + yNudge, teleportLocation.z);
				}
			}

			if (device.GetTouchUp (SteamVR_Controller.ButtonMask.Touchpad)) {
				laser.gameObject.SetActive (false);
				teleportAimer.SetActive (false);
				Player.transform.position = teleportLocation;
			}
		} else {
			if (device.GetTouch (SteamVR_Controller.ButtonMask.Touchpad)) {

				laser.gameObject.SetActive (true);
				laser.SetPosition (0, gameObject.transform.position);
				RaycastHit hit;

				if (Physics.Raycast (transform.position, transform.forward, out hit,
10, laserMask)) {
					teleportLocation = hit.point;
					laser.SetPosition (1, teleportLocation);
```

```
                                                float zSize = hit.point.z - laser.GetPosition (0).z + 0.5f;
                                                laser.GetComponent<BoxCollider> ().size = new Vector3
(0, 0, zSize);
                                                laser.GetComponent<BoxCollider>    ().center    =    new
Vector3 (0, 0, zSize / 2);
                                        }
                                }
                                if (device.GetTouchUp (SteamVR_Controller.ButtonMask.Touchpad)) {
                                        laser.gameObject.SetActive (false);
                                }
                        }
                }

                void OnTriggerStay (Collider other) {
                        if (other.gameObject.CompareTag ("State")) {
                                if (device.GetPressDown (SteamVR_Controller.ButtonMask.Trigger)) {
                                        GrabObject (other);
                                } else if (device.GetPressUp (SteamVR_Controller.ButtonMask.Trigger)) {
                                        ReleaseObject (other);
                                }
                        }
                }

                void OnTriggerEnter (Collider other) {
                        if (other.gameObject.CompareTag ("Button")) {
                                device.TriggerHapticPulse (2000);
                                buttonController.ButtonPress (other.name);
                                PressButton (other.transform);
                        }
                }

                void PressButton (Transform button) {
                        string name = button.name;
                        switch (name) {
                                case "Button1":
                                        dataPlotter.selectState ("Bihar");
                                        break;
                                case "Button2":
                                        clustering.Cluster ();
                                        break;
                                case "Button3":
                                        menuPloter.PlotFile ();
                                        break;
                                case "Button4":
                                        buttonController.ActivateSubMenu (button);
                                        break;
                                case "Button5":
                                        costAnalysis.CostOfCultivationPerHectareVsState ();
                                        break;
                                case "Button6":
                                        costAnalysis.YieldVsCrop ();
                                        break;
                                case "Button7":
                                        costAnalysis.PerHectareCostPriceVsState ();
                                        break;
                                case "Button8":
                                        costAnalysis.YieldVsState ();
                                        break;
                                case "Button9":
                                        costAnalysis.YieldKGVsState ();
                                        break;
                                case "Button10":
                                        if (RenderSettings.skybox.name == "CloudyCrown_Midnight")
                                                RenderSettings.skybox = day;
                                        else
                                                RenderSettings.skybox = night;
                                        break;
                                case "Button11":
                                        //reset
                                        break;
                                case "Button12":
                                        //restart
                                        break;
                                case "Arhar":
                                        menuPloter.DisableRest (name);
                                        menuPloter.PlotCrop (5, name, 3200, 2200);
                                        cropSelector.setCrop (name);
                                        break;
                                case "Cotton":
                                    menuPloter.DisableRest (name);
                                    menuPloter.PlotCrop (9, name, 36000, 18000);
                                    cropSelector.setCrop (name);
                                    break;
                            case "Gram":
                                    menuPloter.DisableRest (name);
                                    menuPloter.PlotCrop (4, name, 9600, 5600);
                                    cropSelector.setCrop (name);
                                    break;
                            case "Ground Nuts":
                                    menuPloter.DisableRest (name);
                                    menuPloter.PlotCrop (7, name, 9800, 4500);
                                    cropSelector.setCrop (name);
                                    break;
                            case "Maize":
                                    menuPloter.DisableRest (name);
                                    menuPloter.PlotCrop (2, name, 25000, 14000);
                                    cropSelector.setCrop (name);
                                    break;
                            case "Moong":
                                    menuPloter.DisableRest (name);
                                    menuPloter.PlotCrop (6, name, 7200, 4500);
                                    cropSelector.setCrop (name);
                                    break;
                            case "Rice":
                                    menuPloter.DisableRest (name);
                                    menuPloter.PlotCrop (1, name, 110000, 85000);
                                    cropSelector.setCrop (name);
                                    break;
                            case "Mustard":
                                    menuPloter.DisableRest (name);
                                    menuPloter.PlotCrop (8, name, 8200, 5500);
                                    cropSelector.setCrop (name);
                                    break;
                            case "Sugarcane":
                                    menuPloter.DisableRest (name);
                                    menuPloter.PlotCrop (10, name, 370000, 270000);
                                    cropSelector.setCrop (name);
                                    break;
                            case "Wheat":
```

```
                        menuPloter.DisableRest (name);
                        menuPloter.PlotCrop (3, name, 96000, 68000);
                        cropSelector.setCrop (name);
                        break;
                }
        }

        Vector3 pos;
        Vector3 defaultPosition {
                get { return pos; }
                set { pos = value; }
        }

        Quaternion rot;
        Quaternion defaultRotation {
                get { return rot; }
                set { rot = value; }
        }

        void GrabObject (Collider other) {
                defaultPosition = other.transform.localPosition;
                defaultRotation = other.transform.localRotation;
                other.transform.SetParent (gameObject.transform);
                other.GetComponent<Rigidbody> ().isKinematic = true;
                device.TriggerHapticPulse (2000);
        }

        void ReleaseObject (Collider other) {
                other.transform.SetParent (India.transform);
                other.GetComponent<Rigidbody> ().isKinematic = false;
                other.transform.localPosition = defaultPosition;
                other.transform.localRotation = defaultRotation;
        }
}
```