Intelligent Virtual Audience

Submitted in partial fulfilment of the requirements of the degree of
BACHELOR OF TECHNOLOGY

By

NAVROZE MISHRA (127240).

Supervisor:

Dr. B.B. Amberker

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

2015-2016

APPROVAL SHEET

This Project Work entitled by **Intelligent Virtual Audience** by **Navroze Mishra** is approved for the Degree of Bachelor of Technology, Computer Science and Engineering.

Examiners:

____

____

____

____

____

Supervisor:

Dr. B.B. Amberker

Chairman

Dr.Ch.Sudhakar

Date: ______________________________

Place: ______________________________

DECLARATION

I declare that this written submission represents our ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action of the institute of and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

____

____

____

____

(Signature)

NAVROZE MISHRA (127240)

Date: ____________________________

ABSTRACT

The ability to face a large audience and present one's ideas and skills to them has become an integral trait in today's world. The lack of proper practice in public speaking leads to instances of anxiety, nervousness and panic in many of the amateur public speakers. The initial phases of learning to speak in front of an audience involves stammering, loss of interest from the listeners and unusual body movements which can lead to loss in confidence in the speaker. This leads to fear of facing a large crowd of people, which hinders the person's ability to speak in public.

The concept of Virtual Reality (VR) is gaining popularity day by day in a wide range of fields from school education to military applications. VR gives a better experience and closer approximation of the real world as compared to traditional technologies. Moreover it is cheap and easily accessible. Hence, VR can be utilized to tackle the problem of public speaking. A virtual environment can be installed in the VR interface that emulates the behavior of a real world audience.

To create virtual environment, the application records the speaker's speech, position, body movement and gestures. The application includes an intelligence setup that makes the audience respond to the user's actions recorded. The audience's intelligence refers to the idea that they will generate adequate responses in accordance to the context of speech and body language of the user. The audience not only responds to the user's actions but also have some random actions and responses to the environment which makes them more realistic.

The setup comprise of a VR headset connected to a smartphone that detects the body motion and speech of the user. The applications computes the relevance of the spoken content to the topic in context, and based on that decides the interest levels of the virtual audience. The application also records speaker's body movements, and responds in accordance to certain irregularities. In response to the data collected, it demonstrates a virtual audience imitating a response given by an actual real life audience.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

VR	Virtual Reality
AR	Augmented Reality
VA	Virtual Artifact
HMD	Head Mounted Display
SDK	Software Development Kit
IDE	Integrated Development Environment
adb	Android Debug Bridge
TF-IDF	Term Frequency Inverse Document Frequency
TF	Term Frequency
IDF	Inverse Document Frequency
PMI	Pointwise Mutual Information

CHAPTER 1

1 Introduction

1.1 Definition of Virtual Reality

Virtual reality (VR) is the term used to describe a three-dimensional, computer generated environment which can be explored and interacted with by a person. That person becomes part of this virtual world or is immersed within this environment and whilst there, is able to manipulate objects or perform a series of actions.

On a computer, virtual reality is primarily experienced through two of the five senses: sight and sound. The simplest form of virtual reality is a 3-D image that can be explored interactively at a personal computer, usually by manipulating keys or the mouse so that the content of the image moves in some direction or zooms in or out. More sophisticated efforts involve such approaches as wrap-around display screens, actual rooms augmented with wearable computers, and haptics devices that let you feel the display images.

1.1.1 Augmented Reality v/s Virtual Reality

Augmented Reality is the blending of virtual reality and real life, as developers can create images within applications that blend in with contents in the real world. With AR, users are able to interact with virtual contents in the real world, and are able to distinguish between the two. On the other hand, virtual reality involves complete immersion of the user in the application generated scene and hence losing all contact with the actual world.

Both virtual reality and augmented reality are similar in the goal of immersing the user, though both systems do this in different ways. With AR, users continue to be in touch with the real world while interacting with virtual objects around them. With VR, the user is isolated from the real world while immersed in a world that is completely fabricated.

As it stands, AR is scenarios where projection of holographic virtual values is implemented on the real world pictures, while VR might work better for video games and social networking in a virtual environment which involve higher levels of user interaction.

1.2 How is Virtual Reality Implemented?

Most up to date virtual realities are displayed either on a computer screen or with special stereoscopic displays, and some simulations include additional sensory information and focus on real sound through speakers or headphones targeted towards VR users. Some advanced, haptic, systems now include tactile information, generally known as force feedback in medical, gaming and military applications. Furthermore, virtual reality covers remote communication environments which provide virtual presence of users with the concepts of telepresence and telexistence or a virtual artifact (VA) either through the use of standard input devices such as a keyboard and mouse, or through multimodal devices such as a wired glove or omnidirectional treadmills. The simulated environment can be similar to the real world in order to create a lifelike experience—for example, in simulations for pilot or combat training—or it differs significantly from reality, such as in VR games.

Most modern day implementations utilize complex and expensive hy6technology implementations. But cheaper and more widespread devices like the cardboard are also becoming popular.

1.3 Uses of Virtual Reality

1.3.1 Education

Strides are being made in the realm of education, although much needs to be done. The possibilities of VR and education are endless and bring many advantages to students of all ages.

Few are creating content that may be used for educational purposes, with most advances being done in the entertainment industry, but many understand and realize the future and the importance of education and VR.

1.3.2 Training

The usage of VR in a training perspective is to allow professionals to conduct training in a virtual environment where they can improve upon their skills without the consequence of failing the operation.

VR plays an important role in combat training for the military. It allows the recruits to train under a controlled environment where they are to respond to different types of combat situations. A fully immersive virtual reality that uses Head-mounted display (HMD), data suits, data glove, and VR weapon are used to train for combat. This setup allows the training's reset time to be cut down, and allows more repetition in a shorter amount of time. The fully immersive training environment allows the soldiers to train through a wide variety of terrains, situations and scenarios.

1.3.3 Other Fields

Virtual reality has multi-facet uses in a variety of other fields as well like medical science, video games, archaeological analysis etc. It is being used to train novice surgeons and also by various historians to relive and examine the historic excavations.

1.4 Background and Related Work

Virtual reality (VR) is ever evolving with each passing year, and various significant developments were made in the past to trick the human brain to believe in an alteration of the reality. In the early 80’s being able to see the scenes from a setup miles away via television was termed virtual reality, the development of video games allowing user to interact with artificial environments defined the realms of virtual reality and these days the advent of HMDs has taken over the idea of virtual reality. A few significant works in the field of VR are:

- **SENSORAMA [1960’s]:** A prerecorded film in color and stereo was augmented by binaural sound, scent, wind and vibration experiences
- **VIDEOPLACE [1970’s]:** Allowed participants to interact one with the other using the image processing techniques that determined their positions in 2D screen’s space
- **BOOM [1980’s]:** BOOM is a small box containing two CRT monitors that can be viewed through the eye holes. The user can grab the box, keep it by the eyes and move through the virtual world, as the mechanical arm measures the position and orientation of the box.
- **CAVE [1990’s]:** Instead of using a HMD it projects stereoscopic images on the walls of room. This approach assures superior quality and resolution of viewed images, and wider field of view.

In the field of utilizing VR to aid public speaking, only a few attempts are made some of which utilized the television screen to demonstrate an audience by recording the user’s input via a camera [8]. Until now no significant development has been made to develop a Virtual audience using HMDs.

1.5 Organization of the Report

Chapter 2 states the problem statement of the project. Chapter 3 explains the proposed model for the design of the application stating the design overview, basic assumptions, various terminologies (TFIDF and PMI) and techniques utilized in the report. Chapter 4 explains the implementation procedures stating the flow diagram, various steps of the procedure, the algorithms involved, the experimental setup and the code & final results. Chapter 5 details the conclusion of the project and the scope for future work.

CHAPTER 2

2 Problem Statement

The art of public speaking has been around for generations and it can only be mastered by a person after persistent practice in front of a real world audience. The initial phases of learning to speak in public involve experiences with stammering, stage fear, inconsistent body movements and irregularities in speech. The anxiety and distress of becoming a subject of mockery in front of other people often lead to the inability to learn the process of public speaking for most people.

The advent of virtual reality now provides us with an alternate path to learn public speaking without the presence of other people acting as audience. A virtual audience can be created in VR and can be utilized to trick the user's brain to presume them as a real life scenario.

In this project, we propose a working model that emulates the real world audience conduct in the VR environment. We aim to make the virtual audience proficient enough to carry out the following purposes:

- The audience unmistakably identifies the words and phrases included in the user's speech input.
- The audience efficiently defines the relevance of the dialogue produced by the user to the topic in context.
- The audience determines instances of stammering and stuttering in the user's speech.
- The audience detects irregular body movements performed by the user during his speech.
- The audience generates a dignified response to all the above specified inputs such that the response is an apt imitation of the responses generated by actual human audience.

The mounted smartphone’s microphone and gyroscope are employed to generate responses to occurrences stammering and uneven body movements. TF_IDF algorithm is utilized to extract keywords from the speech content. PMI algorithm is applied to generate the relevance between the topic in context and the speech input.

CHAPTER 3

3 Proposed Model

3.1 Overview of the Model

The VR model is based on an android smartphone mounted on the Google cardboard. The process of generating a virtual audience has been showed by the following flow diagram.

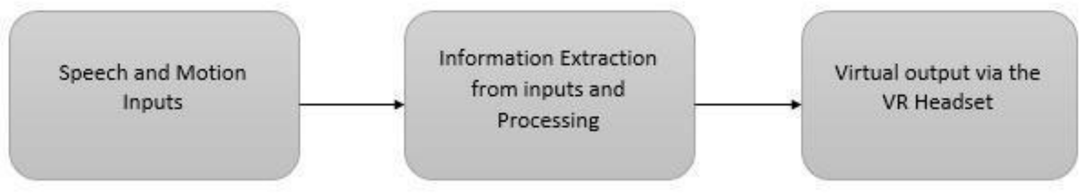

Speech and Motion Inputs
Information Extraction from inputs and Processing
Virtual output via the VR Headset

Figure 1 The flow diagram for VR audience processes

The sound and motion inputs are processed separately and both lead to the generation of responses from the audience.

3.2 Basic Assumptions

Basic assumptions with respect to the audience behavior are based on their responses to speech and movements of the user. These assumptions are as follows:

- The audience will show an increase in attention levels when the speech content by the user is in relevance to the topic of discussion.
- Long pauses, stammers or talking at a very fast pace will lead to unrest in the audience.
- A gaze or eye-contact towards audience member will lead to increase in his/her attention levels.
- Irregular movements of body like trembling & shaking of head will lead to decrease in the audience attention levels.

3.3 Terminologies

The following terminologies are of significance with respect to our application design and implementation:

3.3.1 TF-IDF

TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection [7]. The TFIDF value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

We have utilized the TF-IDF score of the terms in order to extract keywords from the speech content obtained from the user. The term TF-IDF comprises of 2 main terms Term Frequency and Inverse Document Frequency. Both the terms are described in the sections that follow along with TF-IDF.

3.3.1.1 Term frequency

We count the number of times each term occurs in each document and sum them all together; the number of times a term occurs in a document is called its term frequency.

Different variants of TF are there, which are explained in the following table.

Weighting Scheme	**TF**
Binary	0,1
Raw Frequency	$f_{t,d}$
Normalized Raw Frequency	$\frac{f_{t,d}}{d_{size}}$
Log Normalization	$1 + \log(f_{t,d})$
Double Normalization 0.5	$0.5 + 0.5.\frac{f_{t,d}}{max_{\{t' \in d\}} f_{t',d}}$
Double Normalization K	$K + K.\frac{f_{t,d}}{max_{\{t' \in d\}} f_{t',d}}$

Table 1 Variants of TF

where,

$f_{t,d}$ refers to the number of times the term t appears in the document d.

3.3.1.2 Inverse document frequency

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

Different variants of IDF are there, which are explained in the following table.

Weighting Scheme	**IDF**
Unary	1
Inverse Document Frequency	$\log\left(\frac{N}{n_t}\right)$
Inverse Document Frequency Smooth	$\log\left(1+\frac{N}{n_t}\right)$
Inverse Document Frequency Max	$\log\left(1+\frac{\max_{\{t'\in d\}} n_{t'}}{n_t}\right)$
Probabilistic Inverse Document Frequency	$\log\left(\frac{N-n_t}{n_t}\right)$

Table 2 Variants of IDF

where,

$n_t = |\{d \in D: t \in d\}|$,

N is total number of documents in the corpus.

3.3.1.3 Term frequency-Inverse document frequency

TF-IDF values are then calculated as the product of the TF and IDF values, i.e.:

$$tfidf(t, d, D) = tf(t, d). idf(t, D)$$

In our application, we have utilized Normalized Raw Frequency as TF and Inverse Document Frequency Smooth as IDF values. The same shall be applicable to all codes, algorithm and calculations.

3.4 Pointwise Mutual Information

PMI (Pointwise mutual information) is a measure that represents strength of relationship between two events [5]. PMI is calculated as:

$$PMI(x,y) = log \frac{P(x,y)}{P(x)P(y)}$$

$$= log \frac{f(x,y).K}{f(x)f(y)}$$

where,

$P(x)$ is an occurrence probability of word x,

$P(x,y)$ is co-occurrence probability of x & y,

$f(x)$ is the number of occurrences of word x,

$f(x,y)$ is the number of co-occurrences of x & y,

K is the number of frames into which the document is divided.

PMI has two problems. One is that PMI cannot measure the relationship of words which do not co-occur, and another is that PMI has an excessively large value when x & y rarely occur. In order to solve these problems, we employ the smoothed PMI. This method uses t-test to examine whether f(x) and f(y) are large enough or not. The t-score $t(x, y)$tests whether the difference between $P(x,y)$ and $P(x)P(y)$ is significant or not. The smoothed PMI is calculated as:

$$PMI(x,y) = \begin{cases} log \frac{f(x,y).K}{f(x).f(y)} & if\ t(x,y) > \theta \\ 0 & Otherwise \end{cases}$$

$$t(x,y) = \frac{\left|f(x,y) - \frac{f(x).f(y)}{K}\right|}{}$$

$$\sqrt{\hat{f}(x,y)}$$

$$f(x,y) = \begin{cases} f(x,y) & if f(x,y) > 0 \\ 0 & Otherwise \end{cases}$$

3.5 Techniques

The project revolves around the 2 basic techniques used for determination of intelligence of the audience. The first of these being the ability of keyword extraction [1] ,[2] ,[3], i.e. the audience after listening the vocal input is able to filter out unnecessary words and keep the significant ones for analysis [1]. The second technique deals with the relevance estimation of the keywords to the topic in context. This project model utilizes the fact that the words with high TF-IDF score over a set of provided documents can be filtered as keywords. The relevance of any document is considered high with respect to the topic if the PMI score of the keywords of the document is of significant value when calculated against a large dataset of topic related documents.

CHAPTER 4

4 Design and Implementation

In the project the modules which were discussed in the proposed design model are implemented. The focus of this implementation is creating a virtual experience for the user capable of imitating the real world audience responses. The flow diagram of the project and the algorithms as in the next sections will serve as the guidelines during implementation.

4.1 Flow Diagram

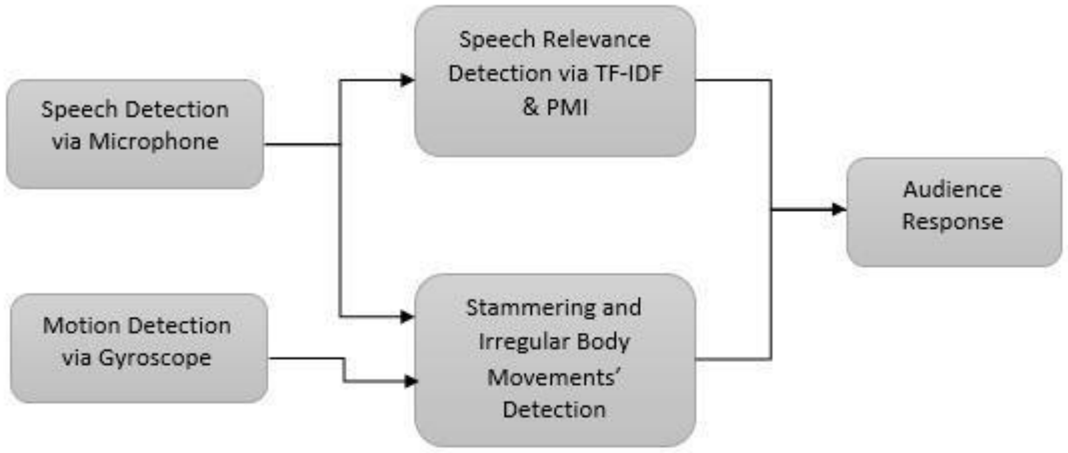

Speech Detection via Microphone
Motion Detection via Gyroscope
Speech Relevance Detection via TF-IDF & PMI
Stammering and Irregular Body Movements' Detection
Audience Response

Figure 2 Flow Diagram Explaining the Steps of the Implementation

4.2 Procedure

In the following section we are discussing the procedure followed for the processing of the vocal and motional inputs. The entire implementation can be broadly categorized into the following main subheadings.

4.2.1 Determining Relevance of the Spoken Content

Vocal inputs are recorded by the microphone of the smartphone mounted in the Google cardboard. The recorded spoken content's relevance to the topic is then determined by the steps as described in the figure below.

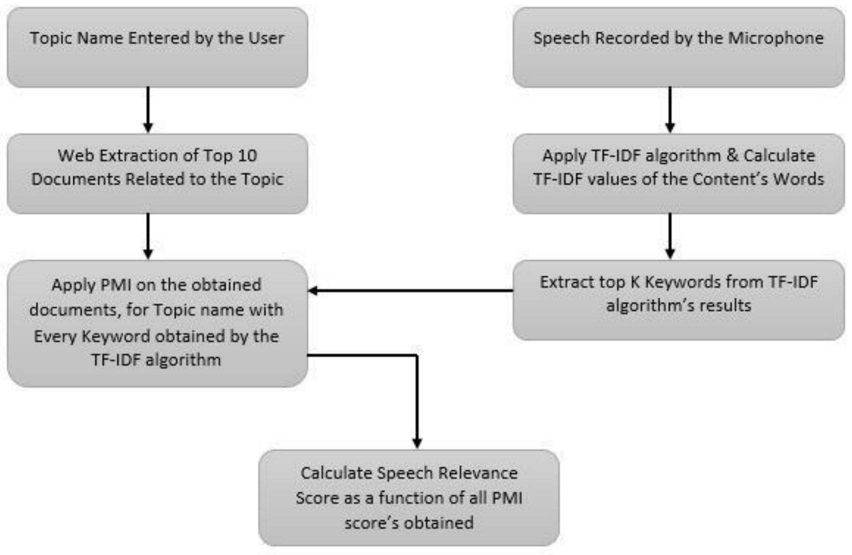

Topic Name Entered by the User
Speech Recorded by the Microphone
Web Extraction of Top 10 Documents Related to the Topic
Apply TF-IDF algorithm & Calculate TF-IDF values of the Content's Words
Apply PMI on the obtained documents, for Topic name with Every Keyword obtained by the TF-IDF algorithm
Extract top K Keywords from TF-IDF algorithm's results
Calculate Speech Relevance Score as a function of all PMI score's obtained

Figure 3 Steps determining the relevance of the spoken content to the topic

The steps of the process described in the image are elaborated in the points mentioned in the section that follows:

- As the topic name is entered by the user, the application performs web extraction and stores the content of top 10 documents related to the topic by performing a search for the topic name on the internet. These documents then act as the corpus of documents for the PMI algorithm.

- The words spoken by the user are segmented into various frames of fixed word length. These frames then undergo TF-IDF algorithm and keywords are extracted for each frame as the words with highest TF-IDF values.
- The PMI score for each keyword is then calculated with respect to the topic name over the document corpus extracted previously.
- The average of all the PMI scores is taken and then compared against a threshold value to determine the relevance of the spoken content to the topic in context.

4.2.2 Determining secondary inputs

The smartphone in the VR headset keeps on recording the user's speech and motions, which can be used as secondary inputs and generate the following outputs.

- The microphone on the smartphone continuously keeps on recording the user's voice input. Any long pause, stammering or irregularities in continuous speech are detected by the microphone and generate a response from the virtual audience.
- The phone's gyroscope is easily able to detect even the smallest of the head movements of the user. Hence any irregular movements or excessive shaking causes unrest in the virtual audience.
- If a virtual audience member is gazed upon i.e., he falls in the user's virtual line of sight for a certain duration, it causes the member to show an increase in attention level.

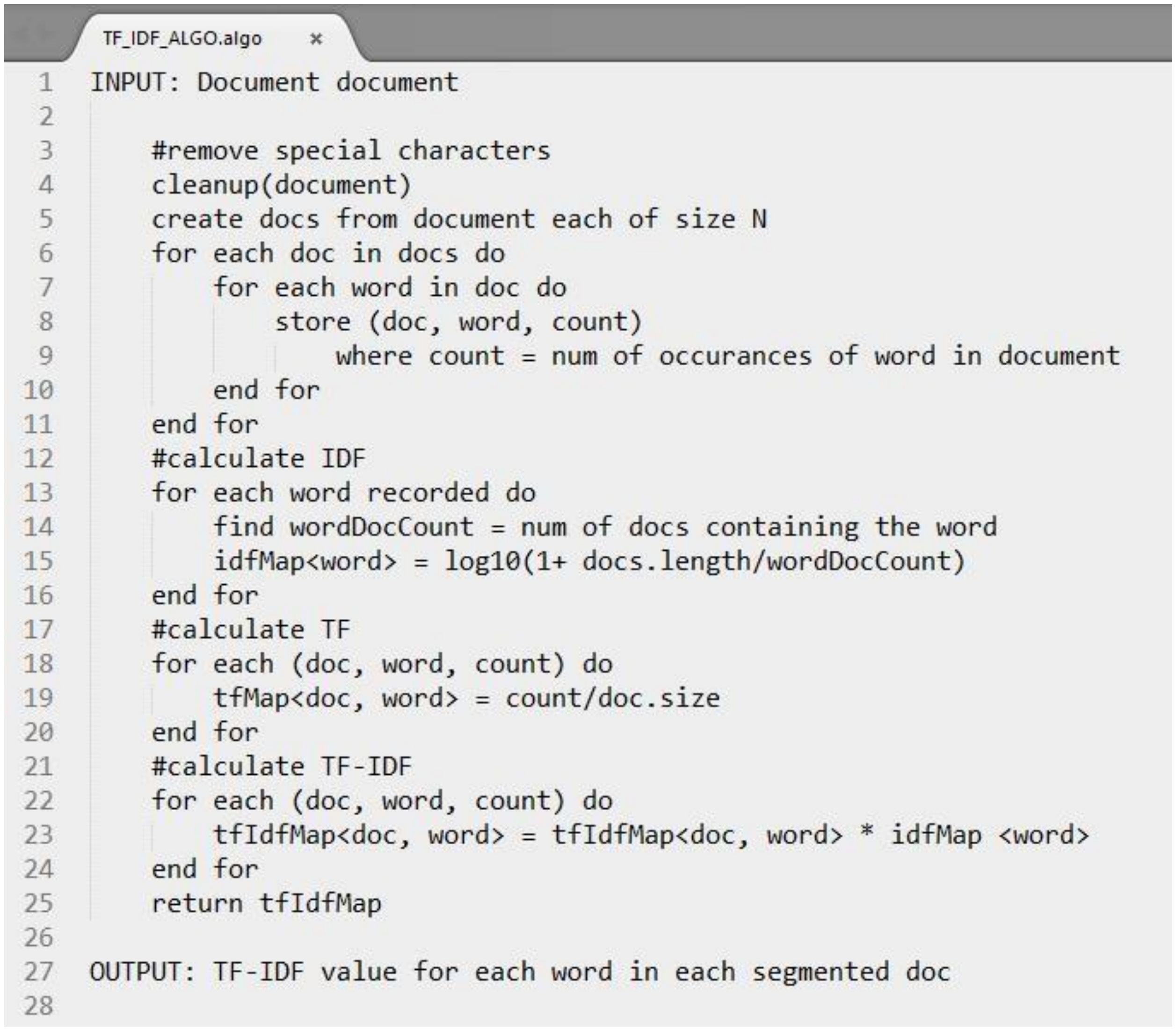
4.2.3 Creating the Animations & Virtual Environment

A virtual classroom environment was created and then integrated with the android application. In order to create the virtual audience and environment the following steps are performed:

- A classroom was created in Unity with a view from the perspective of a speaker speaking in front of the class.
- Audience animations were created in Visual Studio to emulate the movements, expressions and motions of a real world audience.
- Android application was developed to record user inputs and generate adequate responses from the virtual audience as described in the steps above.

4.3 Algorithms

4.3.1 TF-IDF Algorithm



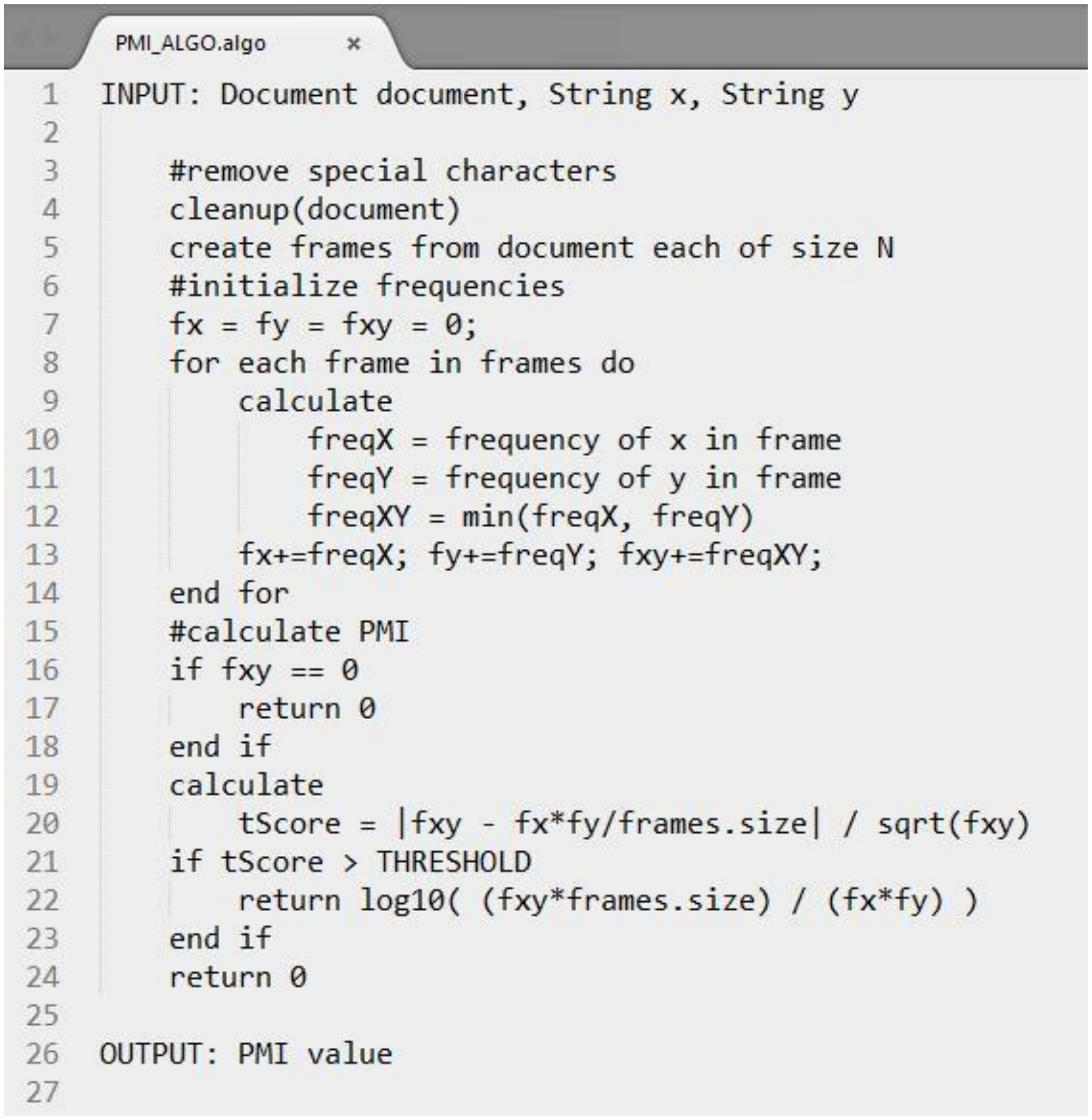
```
INPUT: Document document

    #remove special characters
    cleanup(document)
    create docs from document each of size N
    for each doc in docs do
        for each word in doc do
            store (doc, word, count)
                where count = num of occurances of word in document
        end for
    end for
    #calculate IDF
    for each word recorded do
        find wordDocCount = num of docs containing the word
        idfMap<word> = log10(1+ docs.length/wordDocCount)
    end for
    #calculate TF
    for each (doc, word, count) do
        tfMap<doc, word> = count/doc.size
    end for
    #calculate TF-IDF
    for each (doc, word, count) do
        tfIdfMap<doc, word> = tfIdfMap<doc, word> * idfMap <word>
    end for
    return tfIdfMap

OUTPUT: TF-IDF value for each word in each segmented doc
```

Algorithm 1 TF-IDF

4.3.2 PMI Algorithm



PMI_ALGO.algo

```
INPUT: Document document, String x, String y

    #remove special characters
    cleanup(document)
    create frames from document each of size N
    #initialize frequencies
    fx = fy = fxy = 0;
    for each frame in frames do
        calculate
            freqX = frequency of x in frame
            freqY = frequency of y in frame
            freqXY = min(freqX, freqY)
        fx+=freqX; fy+=freqY; fxy+=freqXY;
    end for
    #calculate PMI
    if fxy == 0
        return 0
    end if
    calculate
        tScore = |fxy - fx*fy/frames.size| / sqrt(fxy)
    if tScore > THRESHOLD
        return log10( (fxy*frames.size) / (fx*fy) )
    end if
    return 0

OUTPUT: PMI value
```

Algorithm 2 PMI

4.4 Experimental Setup

4.4.1 Operating Environment

4.4.1.1 Hardware Specifications

Operating Device: Any Android Device with Screen Size > 4.3 inches

Storage Space Required: 50 MB

HMD: Google Cardboard and Similar HMDs

4.4.1.2 Software Specifications

Operating System: Android Version 5.0 Jelly Bean & above

4.4.2 Development Software Utilized

4.4.2.1 Unity

Unity is a cross-platform game engine developed by Unity Technologies and is used to create 2D and 3D environments for video games, VR applications and other applications.

4.4.2.2 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. Amongst many functionalities of Visual Studio, it can be utilized to create and edit various animation objects used in application.

4.4.2.3 Android Studio

Android studio is the official IDE for android development. It provides an interface to create android applications on a windows software and allows the user to integrate these applications on a mobile android device.

4.4.2.4 Cardboard SDK

Cardboard SDK provides a platform for development of applications for the Google Cardboard. It integrates with unity and can be used to create virtual 3D environments from scratch. It provides toolkit for VR development tasks like lens distortion correction, 3D calibration, head tracking etc.

4.5 Code & Results

The code implementation and the screenshots from the final HMD screen are presented in Appendix A and Appendix B respectively. Appendix A shows the code implementations of TD-IDF and PMI algorithms. Appendix B demonstrates the final view that the user will get while interacting with the virtual audience.

CHAPTER 5

5 Conclusion & Future Work

5.1 Conclusion

In this project, we have developed a VR application that emulates the actions of the real world audience. The application records user's speech, derives its relevance to the topic in context using extraction from web documents and generates adequate response in the virtual environment. The application also records user's body movements, continuity in speech and line of sight to make the audience respond in a more realistic fashion.

5.2 Future Work

Although the application achieves efficient implementation of all the proposed objectives there is further scope for improvement. Following are the implementations which can extend the application even further:

Currently, the gestures and movements are only detected by the VR headset which is unable to detect the hand gestures effectively. Certain sensors can be placed on user's hands to study his gestures and create an output accordingly.

The application can be extended to other fields involving audience response like music performances, where it can detect the quality of the music performed and generate a response for that.

This implementation relies on algorithms TFIDF and PMI which scan the entire document and generates relevance. This misses out on certain single phrases, jokes and references.

APPENDIX A

A.1 TF-IDF

```
package com.finalYearProject.unityvoicelibrary.algorithms;

import android.util.Log;

import java.util.*;

public class TF_IDF {
    public static final String TAG=TF_IDF.class.getSimpleName();
     public static final int CHUNK_SIZE = 10;
public static final int CHUNK_FILTER_SIZE = 5;
public static int numOfWords;     public static
double[] idfVector;     public static double[][]
tfIdfMatrix;     public static double[][]
tfMatrix;     public static String[] wordVector;
public static int docLength[];

    public static LinkedHashMap sortHashMapByValuesD(HashMap<String,
Double> passedMap) {
        List mapKeys = new ArrayList(passedMap.keySet());
        List mapValues = new ArrayList(passedMap.values());
        Collections.sort(mapValues);
        Collections.reverse(mapValues);
        Collections.sort(mapKeys);
        Collections.reverse(mapKeys);

        LinkedHashMap sortedMap = new LinkedHashMap();

        Iterator valueIt = mapValues.iterator();
while (valueIt.hasNext()) {             Object
val = valueIt.next();
            Iterator keyIt = mapKeys.iterator();
             while (keyIt.hasNext()) {
Object key = keyIt.next();
                String comp1 = passedMap.get(key).toString();
                String comp2 = val.toString();
                 if (comp1.equals(comp2)){
passedMap.remove(key);
mapKeys.remove(key);
                    sortedMap.put((String)key, (Double)val);
break;
                }
            }

}
        return sortedMap;
    }           private  static  String  cleanup(String
document) {           return   document.replaceAll("[-
+.^:,'\"]","");
    }
    public static List<String> getTopKeywordsList(String document) {

        //STEP 0, clean TF_IDF_DOCUMENT and divide into smaller chunks of
size = chunksize
        document=cleanup(document);
        List<String> myList = new ArrayList<String>();
int ctr = 0;         String str = "\0";
        for ( String word : document.split(" ")) {
ctr++;
            str = str+word+" ";

            if(ctr == CHUNK_SIZE)
            {
                myList.add(str);
ctr = 0;                 str =
"\0";
            }
        }

        String[] docs = myList.toArray(new String[myList.size()]);


        // STEP 1, scan all words and count the number of different words
        // mapWordToIdx maps word to its vector index
        HashMap<String, Integer> mapWordToIdx = new HashMap<>();
int nextIdx = 0;         for (String doc : docs) {
            for (String word : doc.split(" ")) {
if (!mapWordToIdx.containsKey(word)) {
mapWordToIdx.put(word, nextIdx);
nextIdx++;
                }
            }
        }          numOfWords =
mapWordToIdx.size();

        // STEP 2, create word vector where wordVector[i] is the actual
word         wordVector = new String[numOfWords];         for (String
word : mapWordToIdx.keySet()) {             int wordIdx =
mapWordToIdx.get(word);
            wordVector[wordIdx] = word;
        }
        // STEP 3, create doc word vector where docCountVector[i] is
number of
        // docs containing word of index i
        // and doc length vector
        int[] docCountVector = new int[numOfWords];
docLength = new int[docs.length];
        // lastDocWordVector is auxilary vector keeping track of last doc
index
        // containing the word
        int[] lastDocWordVector = new int[numOfWords];
for (int wordIdx = 0; wordIdx < numOfWords; wordIdx++) {
lastDocWordVector[wordIdx] = -1;
        }
        for (int docIdx = 0; docIdx < docs.length; docIdx++) {
            String doc = docs[docIdx];
String[] words = doc.split(" ");             for (String
word : words) {                 docLength[docIdx] =
words.length;                 int wordIdx =
mapWordToIdx.get(word);                 if
(lastDocWordVector[wordIdx] < docIdx) {
lastDocWordVector[wordIdx] = docIdx;
docCountVector[wordIdx]++;
                }
            }
        }

        // STEP 4, compute IDF vector based on docCountVector
idfVector = new double[numOfWords];
        for (int wordIdx = 0; wordIdx < numOfWords; wordIdx++) {
idfVector[wordIdx] = Math.log10(1 + (double) docs.length /
(docCountVector[wordIdx]));
        }

        // STEP 5, compute term frequency matrix,
tfMatrix[docIdx][wordIdx]
        tfMatrix = new double[docs.length][];
        for (int docIdx = 0; docIdx < docs.length; docIdx++) {
tfMatrix[docIdx] = new double[numOfWords];
        }
        for (int docIdx = 0; docIdx < docs.length; docIdx++) {
String doc = docs[docIdx];             for (String word :
doc.split(" ")) {                 int wordIdx =
mapWordToIdx.get(word);
                tfMatrix[docIdx][wordIdx] = tfMatrix[docIdx][wordIdx] + 1;
            }
        }
        // normalize idfMatrix by deviding corresponding doc length
for (int docIdx = 0; docIdx < docs.length; docIdx++) {             for
(int wordIdx = 0; wordIdx < numOfWords; wordIdx++) {
tfMatrix[docIdx][wordIdx] = tfMatrix[docIdx][wordIdx] /
docLength[docIdx];
```

```
            }
        }

        // STEP 6, compute final TF-IDF matrix
        // tfIdfMatrix[docIdx][wordIdx] = tfMatrix[docIdx][wordIdx] *
idfVector[wordIdx]
        tfIdfMatrix = new double[docs.length][];
        for (int docIdx = 0; docIdx < docs.length; docIdx++) {
tfIdfMatrix[docIdx] = new double[numOfWords];
        }
        for (int docIdx = 0; docIdx < docs.length; docIdx++) {
for (int wordIdx = 0; wordIdx < numOfWords; wordIdx++) {
tfIdfMatrix[docIdx][wordIdx] = tfMatrix[docIdx][wordIdx] *
idfVector[wordIdx];
            }
        }

        //STEP 7, selecting the top terms with max tf_idf values

        List<String> topKeywordsList=new ArrayList<>();
         for (int docIdx = 0; docIdx < docs.length; docIdx++) {
HashMap<String, Double> keywords = new HashMap<>();             for
(int wordIdx = 0; wordIdx < numOfWords; wordIdx++) {
keywords.put(wordVector[wordIdx], tfIdfMatrix[docIdx][wordIdx]);
            }
            LinkedHashMap<String,Double>topKeywords =
sortHashMapByValuesD(keywords);             int
counter=0;
            for(Map.Entry entry : topKeywords.entrySet()) {
//Display top CHUNK_FILTER_SIZE values                 counter++;
                if(counter < CHUNK_FILTER_SIZE+1) {
                    topKeywordsList.add(entry.getKey().toString());
                }
            }
        }          return
topKeywordsList;
    }
      public static double[][] getTF_IDFMatrix()
{         return tfIdfMatrix;
    }      public static String[]
getWordVector() {          return wordVector;
    }

}
```

A.2 PMI

```
package com.finalYearProject.unityvoicelibrary.algorithms;
```

```
import android.util.Log;

import java.util.*;

public class PMI {
    public static final String TAG=PMI.class.getSimpleName();

    public static final int CHUNK_SIZE = 25;
    public static final double T_SCORE_THRESHOLD = 0.25;

    public static int numOfFrames;
    public static double fx, fy, fxy; //counts of x, y and x,y occurances
public static double tScore;     public static double pmiScore;
    //Takes arguements frames -> each of length N, total K in number
//also takes x and y as terms to get relevance.

    public static double calculateScore(String document, String x, String
y) {
        //STEP 1, Divide the document into frames of chunkSize(i.e. 100
each)
        document=cleanup(document);
        String[] words = document.split(" ");
List<List<String>> frames = new ArrayList<>();
int length = words.length;         int j = 0;
        for (int i = 0; i < length / CHUNK_SIZE + 1; i++) {
frames.add(new ArrayList<String>());
        }
        for (int i = 0; i < length; i++) {
frames.get(j).add(words[i]);             if
((i+1) % CHUNK_SIZE == 0) j++;
        }          fx = 0; fy = 0; fxy =
0;         for (String word:words){
if (word.equalsIgnoreCase(x)){
fx++;             }
            if (word.equalsIgnoreCase(y)){
fy++;
            }
        }
        for (List<String> frame:frames){
int tempCountX=0;             int
tempCountY=0;             for (String
word:frame){
                if (word.equalsIgnoreCase(x)){
tempCountX++;
                }
                if (word.equalsIgnoreCase(y)){
tempCountY++;
                }
}
            fxy+=(tempCountX>tempCountY)?tempCountY:tempCountX;
        }
        numOfFrames = frames.size();

        //STEP 2, scan each frame and note down:
        //*1) No of occurances of both x and y together
        //*2) No of occurances of x
        //*3) No of occurances of y
        // and calculate total number of occurances of x, y & (x,y)
        // as fx, fy, fxy

        //STEP 3, calculate PMI
if(fxy == 0)
pmiScore = 0;         else {
            //           |fxy - fx.fy/K|
            // tScore = ------------------
//          sqrt(fxy)             tScore =
fx*fy/numOfFrames;             tScore = fxy
- tScore;             if(tScore < 0)
tScore = -tScore;             tScore =
tScore/ Math.sqrt(fxy);
if(tScore > T_SCORE_THRESHOLD){
//                   fxy.K
                // pmiScore = log ----------
//                   fx.fy                 if(
fx!=0 && fy!=0 ) {
                    pmiScore = (fxy*numOfFrames*CHUNK_SIZE)/(fx*fy);
pmiScore = Math.log10(pmiScore);
                }
else
                    pmiScore = 0;
            }
else
                pmiScore = 0;
        }
        return pmiScore;
    }           private  static  String  cleanup(String
document) {           return   document.replaceAll("[-
+.^:,'\"]","");
    }

}
```

APPENDIX B

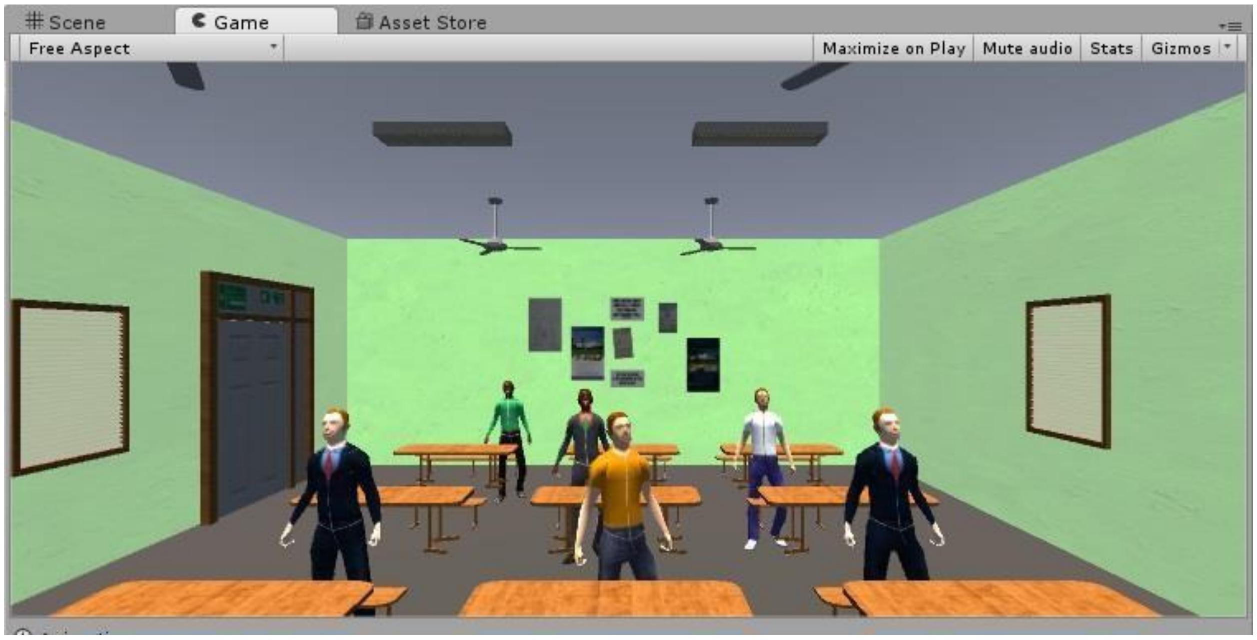


Figure 4 Audience Environment

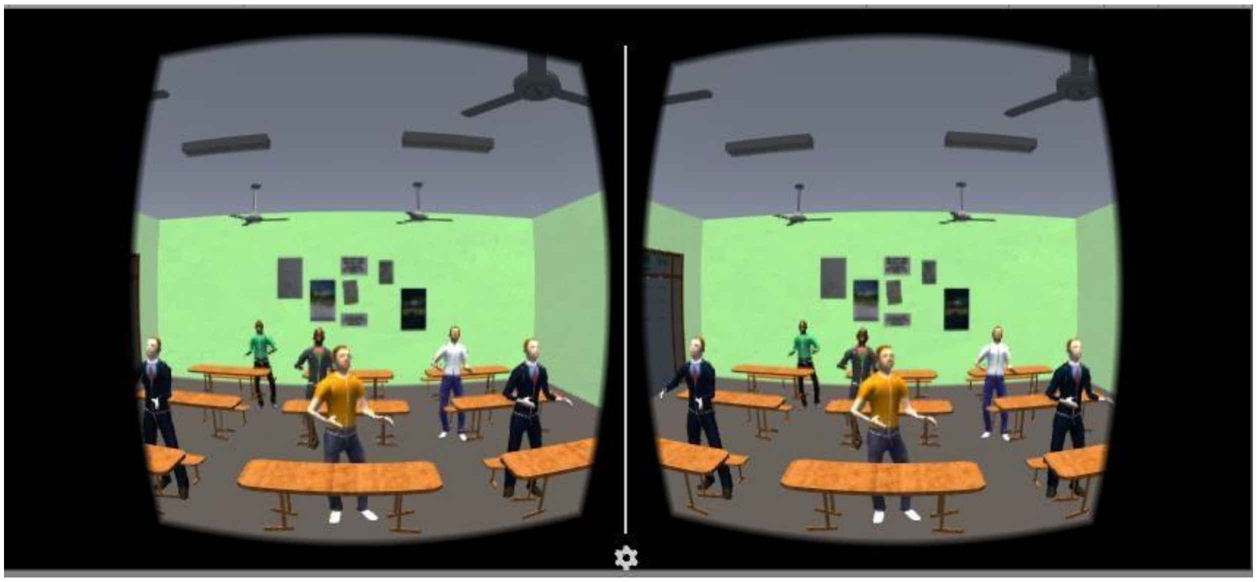


Figure 5 Audience through HMD Display

REFERENCES

[1] Chin-Hui Lee, “An Information-Extraction Approach to Speech Processing: Analysis, Detection, Verification, and Recognition”, 2013. Published in: Proceedings of the IEEE (Volume: 101, Issue: 5).

[2] Matsuo. Y., Ishizuka. M., “Keyword extraction from a single document using word cooccurrence statistical information” International Journal on Artificial Intelligence Tools, Vol.13, pp.157-169, 2004.

[3] Matsumura. N., Ohsawa. Y., Ishizuka. M., “PAI: Automatic indexing for extracting asserted keywords from a document”, New Generation Computing, Vol.21, Issue 1, pp.3747, 2003

[4] Jain, G.P., “Artificial Intelligence-Based Student Learning Evaluation: A Concept Map-Based Approach for Analysing a Student's Understanding of a Topic”, 2014. Published in: Learning Technologies, IEEE Transactions on (Volume:7 , Issue: 3)

[5] Asami. T., Nomoto. N., Kobashikawa. S., Yamagchi. Y., Masataki. H., Takahashi,”Spoken document confidence estimation using contextual coherence”, Proc. INTERSPEECH2011, pp.1961-1964, 2011.

[6] Hara Kensuke, Sekiya Hideki, Kawase Tetsuya, Tamura Satoshi, and Hayamizu Satoru, “Confidence estimation and keyword extraction from speech recognition result based on Web information”, 2013. Published in APSISPA, 2013 Asia-Pacific, IEEE Conference 2013.

[7] Leena H. Patil, Mohammed Atique “A novel approach for feature selection method TFIDF in document clustering”, Advance Computing Conference (IACC), 2013 IEEE 3rd International, 2013.

[8] M. Slater, A. Steed, “Public speaking in virtual reality: facing an audience of avatars”, IEEE Computer Graphics and Applications (Volume: 19, Issue: 2), 2002.