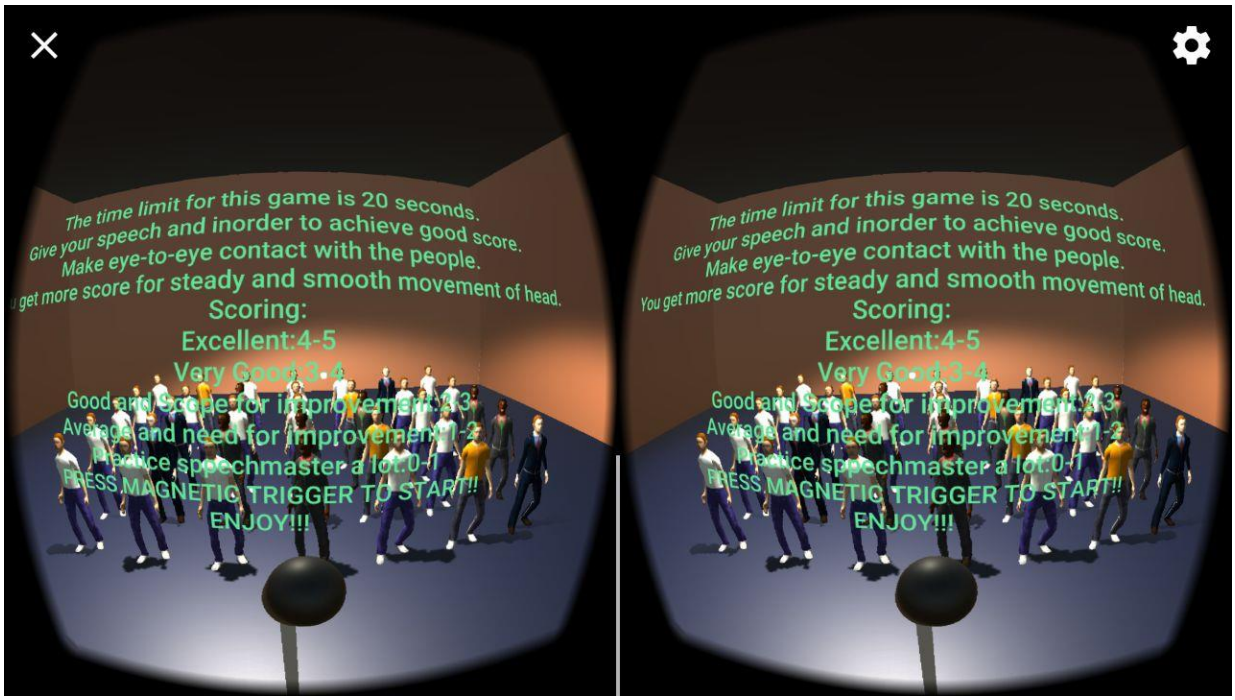
SPEECH MASTER

(Where Virtual Reality and Public speaking skills meet)



Game Design Document
Written by:-

Akshay Pandita

akshaypandita20000@gmail.com

Nimish Mangal

nimish.mangal910@gmail.com

Himanshu Nagdive

nagdive.himanshu@gmail.com

1 Approach:

With every new project, there needs to be a purpose, a roadmap, a cohesive plan as to why, and how one can and will achieve one's intended ideals. With this detailed document, we intend to emphasise why we are making Speech master, as well as how we are going to make it and what benefit will it have .

1.1 Purpose:

With Speech master, we aim to utilise VR technology to create a game which projects a state of utter immersion for the player to:

- See the world through the eyes of the protagonist (speaker in our case trying to improve his public speaking skills).
- Experience his eye contact with the audience.
- Judge and Improve his public speaking skills on the basis of his score
- Learn and try to improve the score by practicing till you reach your level best…

1.2 Design Philosophy:

In order to achieve a harmonious gaming experience, we will focus on the following two design fundamentals:

- Simplicity/Accessibility through gameplay and control design
- Virtual reality through character and narrative design

1.3 Our Target Market:

For a player who is interested to improve his speaking skill and out-perform his stage fear and experience immersive VR-gaming.

1.4 Our Ideals:

As a team of student developers, we want to create a product which we all can be incredible proud of, a product we can showcase to others, and naturally, a product which others can wholeheartedly enjoy and cherish.

2 General Information:

2.1 Genre:

- Cinematic
- Virtual Reality

The chosen genre, enables us as developers to craft a convincing VR experience that has the potential to influence the hearts and minds of the participating gamer.

With Cinematic Virtual Reality, we have the ability to build an immersive world for the player to delve into and experience, as if that inhabited space were real and tangible.IT gives a real life public speaking experience.

2.2 Single Player Campaign:

- Speech Master is designed from the outset, to function as single player, gaming experience.
- Why single player?

The main motivation of the game is to do self-analysis. With each score the players get, he/she can analyse their speaking skill ability and try to improve upon it. Which is possible only through single player mode. Thus, the game is set only on single player mode.

2.3 Marking scheme:

- Excellent – Given when the player scores between 4 and 5 out of 5.
- Very Good – Given when the player scores between 3 and 4 out of 5.
- Good- Given when the player scores between 2 and 3 out of 5.
- Average- Given when the player scores between 1 and 2 out of 5.
- Urgent Need for Improvement- given when the player score between 0 and 1 out of 5.

2.4 Platforms:

- Android, Windows smartphone supporting gyroscope. To play it on the smartphones, the phones must also have google cardboard as it provides better experience to the player of the VR gaming.

2.5 Game Objective:

Speech master is a simple yet useful game, as it deals with our own public speaking experience.

The end-game goal for Speech Master, can be described as the following:

- As the player, you need to stand in front of a mic, the player becomes the speaker.
- In front of the speaker will be audience which are all set to hear the the speech, try not to disappoint them.
- There is a 20s timer in which the player has to complete his speech.
- The player will be given score on the basis of how he interacts with the audience.
- Try to improve the score. With the score getting better the player learns to master his speaking skills in the real life

2.6 Characters:

Speech Master will wholeheartedly appear empty and hollow without its characters, the 3d audience which pave the way for real-life experience.

Given below is the description of the characters.

Speaker: The player standing on the stage who gives the speech in front of the audience.

Audience: Audience are the characters standing in front of the stage of the speaker ready to hear the speech of the speaker.

2.7 Player Perspective:

Speech Master will be experience through the:

• 1st person camera angle/perspective

The 1st person perspective is the chosen perspective as it possesses a greater gaming experience player when working with VR technology, than that of a 3rd person perspective. One descends easier into the body of the speaker and the world around you, by staging the game to be experienced through the eyes of the speaker. We want to establish a sense of connectedness between player and audience.

So, without any further ado, take an experience of the virtual reality through the speech master.

.

2.12 Technology:

- Unity 3D.
- Blender
- C#

3 Player and Game Interaction:

To successfully enjoy the entirety of immersive journey, the player needs to use the following input devices:

Magnetic Button Trigger of google cardboard

VR google cardboard

Any supporting platform

4 Source code and explanation:

```
using System.Collections;          // importing header files

using System.Collections.Generic;

using UnityEngine;                 // importing properties and features from unity3D

using System;

using UnityEngine.SceneManagement;

public class player : MonoBehaviour {     // script written in C# for the object player

public TextMesh infotext;         // textmesh is the datatype used to variable to print things on the
                                     screen in unity.
    public float timer=20f;        // time for which the player can play the VR game

    public int flag=0;             // variable used as a counter

    public int[,] arrayaud = new int[6,8]; // to get the coordinates of the component the reticle hits

    public int countAud=0;         //counting the audience

    public int flag2 = 0;          //counting the total number of audience the player sees

    public int flag3=0;            //counter

    public float score1=0;         // score calculated for seeing the maximum audience

    public float score2 = 0;       // score calculated for the steady velocity of the head

    public int count=0;

    public float threshold_time=1.0f;

    public float ins_velocity=0;     // instantaneous velocity of change from one component to other

    public float avg_velocity=0;

    public float dist=0;           // distance between the two components

    public int t = 0;

    public int flag4=0;

    public float totalscore=0; // final score for the game i.e. score1 + score2

    public float averagevelocity = 0;

    //public float no_of_frames=0;

    //public int totalframes=int(timer/Time.del)

    public string[] aud = new string[6300];    // string containing all the components frame wise

    public string current="";               // containing current component in a particular frame and
                                               storing it in current
```

```
    public float diff;

    public float restarttimer = 5f;                 // time to restart the game


    // Use this for initialization

    void Start () {
                                                   //runs only once at the start of the game
        for (int i = 1; i <= 5; i++)
            for (int j = 1; j <= 7; j++) {          //initializing the array of components of the
                arrayaud [i, j] = 0;                 audience to 0
            }

        infotext.text="The time limit for this game is 20 seconds.\n";

        infotext.text += "Give your speech and inorder to achieve good score.\n" +

            "Make eye-to-eye contact with the people.\n"+

            "You get more score for steady and smooth movement of head.\n"

            +"Scoring:\n"

            +"Excellent:4-5\n"

            +"Very Good:3-4\n"+

            "Good and Scope for improvement:2-3\n"

            +"Average and need for improvement:1-2\n"

            +"Practice sppechmaster a lot:0-1\n"

            +"PRESS MAGNETIC TRIGGER TO START!!\n"

            +"ENJOY!!!";

    }

    // Update is called once per frame
    void Update () {            //update function runs in every frame of the game .approximately this
                                   will run 6300 times in the time period of this game

        RaycastHit hit;             //this is basically the eye tracker and gets the component which
                                      the eye sees

        if (timer>0.0f && flag==1){

            if (Physics.Raycast (transform.position, transform.forward, out hit)) {
                                    //the eye tracks a component and the eye tracker hits any component

                //Debug.Log (hit.transform.name);
                aud [flag2] = hit.transform.name;  //this basically stores the component any gets hit
                                                     by the eye tracker

                if (hit.transform.name [1] == ',') {    //since the coordinates of the audience are of
the form (x,y) and if (,) is there in the second index it means that that component is an audience
```

```
                    countAud += 1;
                    //Debug.Log ((int)hit.transform.name[0]-48);

                    arrayaud[(int)hit.transform.name[0]-48,(int)hit.transform.name[2]-48]+=1;
                //now changing the values of the arrayaud as soon as the components are being hit by
the reticle
                }

                flag2++;
            }
            if (flag2 >=1 && flag4==0 && hit.transform.name[1]==',') {

                current = hit.transform.name;  //storing the coordinate of the component being tracked
by the reticle at that particular instance of time and that frame

                flag4 = 1;

                t = 1;

            } else if(flag4==1){

                if (String.Compare (current, hit.transform.name) == 0) { //comparing the components of
this frame and the previous frame and if its same then the velocity is 0

                    ins_velocity += 0;

                    current = hit.transform.name;

                    t += 1;
                } else {
//if both the strings are not same then the distance is calculated in the dist variable using the
distance formula between two coordinates.

                    if (current [1] == ',' && hit.transform.name [1] == ',') {

                        t += 1;

                        dist = (float)Math.Pow (((current [0] - hit.transform.name [0]) *(current [0]
- hit.transform.name [0]) + (current [2] - hit.transform.name [2]) * (current [2] - hit.transform.name
 [2])), 2);
//distance calculated using the distance formula

                        ins_velocity += dist/(Time.deltaTime*t);
 //instantaneous velocity is incremented every time and at last it is divided by the total number of
frames to get the average velocity.

                        current = hit.transform.name;

                        t = 0;

                    }
                }
            }
        }
        if (Input.GetButtonDown ("Fire1") && timer>0.0f) {
 //as soon as the player triggers the magnetic trigger the game starts showing the timer at every frame
on the unity screen.
```

```
            infotext.text = "Timer:" + Math.Round (timer, 1);

            flag = 1;

        } else if (flag == 1 && timer>0.0f) {

            infotext.text = "Timer:" + Math.Round (timer,1);
        }

        if (timer > 0.0f && flag==1) {
        //the timer has been continuously decremented by a certain value i.e. time.deltatime which is
the time used in one frame.
            timer -= Time.deltaTime;
        }

        else if(flag==1 && flag3==0){

            timer = 0.0f;

            flag3 = 1;
            //Application.Quit ();
            for (int i = 1; i < 6; i++) {
                for (int j = 1; j < 8; j++)
                    if (arrayaud [i,j] != 0) {
                        count += 1;
//count is the variable used to count the number of audiences tracked by the reticle.
                    }
            }

            score1 = (float)((count * 2.5) / 35);
//score1 is calculated here

            avg_velocity = ins_velocity / flag2;
//average velocity is calculated here by dividing ins_velocity with the number of frames (flag2).
            if (avg_velocity == 0.0f) {
                score2 = 0.0f;
            }
            else if (avg_velocity > 0.0f && avg_velocity < 2.0f) {

                score2 = 2.5f;

            } else {

                diff = 400.0f - avg_velocity;

                score2 = (float)diff * (2.5f) / 398f;
            }

            totalscore =(float) Math.Round(score1 + score2,2);

            infotext.text = "Your final score is=" + totalscore+"/5"+"\n";
//printing the final score on the unity screen after the game ends after the time 20sec have passed.



            if (totalscore > 4 && totalscore <= 5) {    //scoring according to the achieved score

                infotext.text += "Excellent Performance\n";
```

```
            } else if (totalscore > 3 && totalscore <= 4) {

                infotext.text += "Very good interaction!!\n";

            } else if (totalscore > 2 && totalscore <= 3) {

                infotext.text += "Good and scope of improved\n";

            } else if (totalscore > 1 && totalscore <= 2) {

                infotext.text += "Average and can be improved\n";

            } else {

                infotext.text += "Practice a lot with SPEECH MASTER\n";
            }

            infotext.text+="Game will automatically restart in 5sec\n";
        }

        if (flag3 == 1) {

            restarttimer -= Time.deltaTime;     //for restarting the game after 5 sec

            if (restarttimer <= 0f) {

                SceneManager.LoadScene (SceneManager.GetActiveScene().name); //loading the scene once
again
            }
        }
    }
}
```