CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING

Department of Cybernetics

BACHELOR PROJECT

Subjective Speech Intelligibility Testing Methodology Design Inspired by ITU-T P.807 Deploying Parallel Load and Virtual Reality

Author: Michaela Urbanovská

Supervisor: Prof. Ing. Jan Holub, Ph.D.

June 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Michaela Urbanovská

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Subjective Speech Intelligibility Testing Methodology Design Inspired by ITU-T P.807 Deploying Parallel Load and Virtual Reality

Guidelines:

Elaborate on overview and state-of-the-art analysis of intelligibility and parallel task (sometimes referred as "parallel load" or "dual task") in subjective tests. Based on this overview propose and demonstrate experimentally a new methodology for implementation of subjective intelligibility tests with a parallel load using virtual reality (VR). The proposed methodology should reasonably (means in aspects where it is theoretically possible) respect existing standards for intelligibility testing., HTC Vive platform and Unity environment should be used as a VR environment.

Bibliography/Sources:

[1] ITU-T Rec. P.800 "Methods for Subjective Determination of Transmission Quality", Series P: Telephone Transmission Quality, ITU, Geneva, 1996, am. 1998
[2] ITU-T Rec. P.807 "Subjective Test Methodology for Assessing Speech Intelligibility", Series P: Terminals and Subjective and Objective Assessment Methods, Geneva, 2016
[3] ANSI/ASA S3.2
[4] ISO 9921 standard on the "Assessment of Speech Communication"
[5] Online at http://www.dynastat.com/Speech%20Intelligibility.htm
[6] Objective and Subjective Measurements, Kai Roemer, kai.roemer@gmx.de

Bachelor Project Supervisor: prof. Ing. Jan Holub, Ph.D.

Valid until: the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 6, 2017

Acknowledgement

I would like to thank everyone who helped me and supported me during working on this project. In the first place to my project supervisor professor Holub for his guidance and help when it was needed. Then, of course, to my family and friends who supported me through the years of my bachelor studies.

Author statement

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instruction for observing the ethical principles in the preparation of university theses.

Prague, date ....

signature

Abstract

Intelligibility of speech is an important factor and subjective intelligibility tests held in standard laboratory conditions as recommended in ITU-T P.807 [2] don't present the same results as tests held with the additional parallel task [8] that simulates the field conditions. Intelligibility is extremely important especially in emergency situations, that's why the field conditions shouldn't be ignored. I decided to show, whether is virtual reality a good generator of the parallel task or not. That means, that an application which involves both parallel task and the testing process has to be implemented and that a demonstrative test has to be held to compare results with different measurement [8].

Keywords: intelligibility, subjective testing, parallel task, dual task, virtual reality, Unity, VR development, MRT, Modified Rhyme Test, ITU-T P.807, HTC Vive

Abstrakt

Srozumitelnost řeči je důležitým faktorem a subjektivní testy, které se provádějí ve standardizovaném laboratorním prostředí jako je doporučeno v ITU-T P.807 [2], nevykazují stejné výsledky jako testy, které jsou prováděny s paralelní zátěží [8], která simuluje skutečné prostředí. Srozumitelnost je extrémně důležitá především v krizových situacích a proto by tyto podmínky neměly být ignorovány. Rozhodla jsem se ověřit, zda je virtuální realita dobrým zdrojem paralelní zátěže nebo ne. To znamená, že je zapotřebí implementovat aplikaci, která bude obsahovat jak paralelní zátěž, tak proces testování. Také je potřeba provést demonstrativní test, aby mohly být výsledky porovnány s výsledky jiného měření srozumitelnosti [8].

Klíčová slova: srozumitelnost, subjektivní testování, paralelní zátěž, dual task, virtuální realita, Unity, VR development, MRT, Modified Rhyme Test, ITU-T P.807, HTC Vive

Contents

List of figures

List of tables

1. Introduction

This thesis is focused on designing speech intelligibility test involving dual-task but otherwise compliant with MRT [7] or ITU-T P.807 recommendations [2]. The dual-task, in this case, will be created using the virtual reality environment in which the testing will take place. The goal is to implement a sustainable application which can be used to perform such testing and to perform a test run with several subjects and compare results with different intelligibility measurement results to see whether the virtual reality is an appropriate tool for the tests.

1.1. Project workflow

To implement such application, it is necessary to become familiar with common subjective testing procedures as specified e.g. in ITU-T P.807 standard [2], which is the most up-to-date recommendation for the intelligibility testing. It is also important to search for as many information sources as possible to get some general knowledge of the problematics and to be able to fulfill all needs of the tests and to understand them.

The next step is getting familiar with HTC Vive and Unity and starting the development. A certain amount of time has to be dedicated to designing the application logic and its functionalities. The goal is to implement an application which can be used again for similar or a different test, so the test mechanics should be easy to modify.

After completing the application, demonstrative test run has to be held with several subjects to acquire data. This data can be later analyzed and compared with already existing data from different measurement [8] to find out if the virtual reality provides a similar parallel load for the subject.

2. Intelligibility testing

Intelligibility of speech is an important predictor of user acceptability. Modified Rhyme Test method (MRT) which will be used in this experiment is one of American National Standards Institute's approved procedures for intelligibility testing according to ANSI S3.2 [7]. The purpose of this test is to figure listener's ability to understand sets of single syllable words in specific conditions with certain sound modifications.

Together with Diagnostic Rhyme Test (DRT), MRT can be assigned to a category of methods with a closed response as the answer is picked from several options and the samples and there's no rating included. MRT and DRT use single syllable stimuli in a multiple-choice task. MRT will be performed in this experiment, so some choices will be in total 6. The intelligibility score is adjusted for guessing. Samples used in this testing are as we said single syllable words which are more difficult to identify and are likely to be used in emergency situations [1].

Usually, there are 50 samples from which 25 have choices which differ in an initial consonant and 25 differ in the last consonant. That means that single consonants are tested. Vocabulary used in this test is fixed and available to the public. The last consonants can differ in six different features, voicing, nasality, sustention, sibilation, graveness and compactness [2].

During the demonstration test, I will be using two last samples from the list as training samples in the tutorial section of the application. The rest of the samples, which is 48, will be used in the test itself. The same approach was used in the testing which will be later analyzed in comparison with my demonstrative experiment. The MRT word list I used for my demonstrative test is available in Appendix A.

MRT test developed from the DRT test which was presented earlier in the year 1958 by Fairbanks at 'Test of phonemic differentiation: The rhyme test.' Ninety-six pairs of words were used for the testing and tested subject always chose from two words which differed only in the initial consonant. The most significant disadvantage was the fact that trained expert listeners had to be used for testing which made the performance more complicated and very time-consuming.

MRT was presented in 1965 by House in 'Psychoacoustic speech tests: A modified rhyme test' [16]. The test consists of 50 sets of 6 one syllable words, 300 words in total. The word is identified from a choice of six words. Also, there's usually a carrier sentence or a carrier sound used when playing a sample. The carrier sentence makes the subject "ready" for the sample word and it also gives space for dynamic sound modifiers to activate and stabilize. The advantage of rhyme tests is their relatively short time to perform because the samples are short sentences focused on one word instead of multiple sentences, also there's no complicated training needed for naive listeners as the principle is very simple. On the other hand, in the case of participating multiple times, the subjects may remember the correct answers due to the fact, that the combinations are fixed [3].

In my demonstrative test, I decided to use carrier sound like the announcer. The participants haven't attended any intelligibility subjective tests for at least half

a year which is a condition that has to be met according to [12] in the 'Eligibility of subjects' section. Also, all of the subjects were non-native English speakers. A new document which recommends methods for intelligibility testing is ITU-T P.807 recommendation [2] which deals with subjective speech intelligibility testing. Methods presented in ITU-T P.807 recommendation are designed for word testing, not a sentence or phrase testing, and apply to North American English. It can also be adapted to other languages. However, in this experiment, we will be using North American English, and non-native speakers as listeners as similar test data using regular laboratory environment are already available to us for comparison purposes.

Samples for each listener need to be randomized for each session separately. For this purpose, I implemented an algorithm into the application, which will randomize samples for each subject. The algorithm used for this task is Fisher-Yates shuffle which was originally intended to be performed by a person with pencil and paper. It was first introduced in 1983. The algorithm was modified with respect to modern technologies and has a linear complexity [6].

2.1. Dual-task in intelligibility testing

Testing with the parallel task is useful for various reasons. Firstly, technologies which require intelligibility testing are usually used in different than standard lab conditions. Talking on the phone is performed while walking, reading and during different mental or physical tasks. Due to this fact, results from standard testing may not meet with real world requirements. These differences could have fatal consequences in certain situations, for example for the military and public safety applications, airport approach control dispatchers and in other emergency or dangerous situations.

There are several different types of parallel task. The first one is a purely mental task which requires mental activity from the subject during testing. It can consist of mathematical calculations, memory games etc. Another type is a purely physical task which requires physical activity like push-ups, running on a treadmill or similar device. The type of task I am going to work with in this experiment is 'dual-task' which requires both, mental and physical task.

Dual-task requires combining both physical and mental task at the same time. During multitasking, the smaller processing capacity of the brain can be focused on the intelligibility testing task as it's being loaded with multiple tasks at a time. Dual-task consists of a primary and secondary task. In this case, the primary task is the process of intelligibility testing which includes focusing on a playing sample, understanding the sample and picking the correct option. The secondary task is supposed to distract the listener and load the brain with a different activity.

Dual-task such as WART (The Walking and Remembering Test) was used to measure how impairments in attention may affect performance in balance and walking of people with brain injuries [5].

In this case, I want to work with virtual reality environment to develop both mental and physical load. To do so I should develop an application that will allow the listener perform the testing part and the dual-task part at the same time.

The mental task should consist mostly of the pressure of being in the virtual environment and trying to adapt to the virtual world. This includes understanding the moving in such world and figuring out how to work with controllers. The physical task should consist mostly of looking around the environment and mapping the world. Also, locating objects in the scene and the testing part itself requires enough movement.

3. The usage of virtual reality

I decided to use virtual reality for this testing because it could provide different scenarios for the testing and could also be a good source of the mental load. Since not as many people are familiar with virtual reality, in general, the listeners may already be distracted enough, so the computation capacity of their brain is lowered just like when performing different activities that have been used to develop parallel task. Processing the surroundings and figuring out controls may be a difficult enough task to deal with for an inexperienced user. Also, the game which is used as the parallel task is simple and quite immersive. Its goal is to be distractive but also fun, so that the listener is involved in the gameplay.

Another great advantage of the virtual reality development is the community which has developed around this topic online. Since VR is becoming more and more popular, it is easier to access tools suitable for its development. There are many open source solutions which can make the developing process easier.

3.1. Advantages of VR

In general, the tests could take place on multiple devices at the same time, which would lower the testing time since the number of listeners is possibly unlimited and depends more on the space. Another advantage is the variability of the application. The environment can change as wanted and there are many possible testing scenarios.

There are also many available platforms which can be used for testing which can make the cost of the test very low. For example, when using mobile phones.

3.2. Disadvantages of VR

The biggest disadvantage is possible cybersickness, motion sickness or nausea which may be caused by experiencing virtual reality. Some people may take the distraction which the VR provides negatively, and that could affect the results. It may also have a different impact on people who already have experiences with some form of VR.

Another disadvantage is the current cost of the high-end virtual reality devices. If the risk of cybersickness was eliminated, cheaper alternatives could be used. However, in this test, I decided to prefer better quality hardware to minimalize any possible negative effects on the results.

4. Possible problems in VR

The main discomfort the users may experience in virtual reality is caused by the low frame rate. As the world starts to move unrealistically and begins to lag, users can experience all kinds of problems which can be similar to motion sickness. They're usually called cybersickness or virtual reality sickness. Symptoms of such condition can be a headache, stomach ache, nausea, blurred vision and many others.

To prevent cybersickness, the application should run smoothly and keep high FPS (frames per second), so the virtual world looks natural. Another prevention is to make sure that movement is performed in the application only when it's performed in real life. When the world moves without the user actually engaging in some kind of movement, the brain receives information that doesn't correspond with the state of the body and it can also cause cybersickness. This problem, in particular, is related to the Sensory Conflict Theory, which is a theory related to cybersickness. It is based on the fact, that senses that provide information about orientation and motion are in conflict because of the VR and the body doesn't know how to handle the situation [15].

To prevent cybersickness, the movement in the application will be performed in specified area which corresponds with the one in the application and further movement will be made by teleportation which seems to be rather natural for the user as it uses fade-in and fade-out effect, which makes the place transition less difficult to proceed. The whole application will be taking place in a quite small area, so the teleportation probably won't have to be used frequently enough to cause some sickness.

5. VR platform

The VR platform I will be using is HTC Vive which is currently one of the most suitable solutions which could be used for intelligibility testing. The device was used at VR Lab at Karlovo namesti.

The original plan was to use Google Cardboard for the testing. It has several advantages, one of them being the low price of the needed hardware. This would be important criteria in case of testing which would have to be performed with more users at one time, but since my demonstrative test took place with one listener at a time, I decided to use the HTC Vive.

Another disadvantage is the controls. Since at the time I started my development, there was no alternative for HTC Vive controllers, the mobile platform could be controlled just by one click. In my application, it would be a problem to implement such controls, but the listeners would have to keep their right hand at the head-mounted display all the time, and not only this is not a very natural way of controlling virtual reality, but it could also be uncomfortable or even painful after a longer period. Thanks to controllers which are part of the HTC Vive platform, the controls seem more natural and understandable.

The main reason, however, was the display quality. Mobile phones tend to have low frame rate and low resolution, which also depends on the specific phone, but still, the phones aren't really comparable with the HTC Vive when it comes to quality of the visuals. From my experience, I know that virtual reality can cause issues mainly to inexperienced users. They can experience a headache, nausea, etc. Since the test was planned to take about fifteen to twenty minutes, it could be hard for some people to perform the whole test. This would also affect the data significantly. This also depends on every individual user, but I assumed that my subjects would be rather inexperienced.

5.1. Technical specifications of used hardware [4]

HTC Vive:

Resolution: 2160 x 1200

Refresh rate: 90Hz

Field of view: 110 degrees

Tracking area: 15x15 feet

The tracking area is limited, and the user can see its boundaries in the application when getting too close to them. There's enough room for listeners to make a step or two forward and to turn around which is very important for my application. Looking around at the surroundings is the key action, and enough space is important because people don't even realize how much space they need in real life when being in virtual reality.

6. Used software

For developing the application, I decided to use Unity for its great support which is provided for HTC Vive development. I wrote all the scripts in C# and all packages I used are available for free.

One of the most important packages is SteamVR[1] which is a package of basic tools to work with during development. They provide such things as scripts for tracking the controllers in the scene, attaching ray to the controller, placing tracked area to the scene and many other basic functions.

Another open-source framework I will be using is the HTC Vive teleportation framework[2] which is available on GitHub. It's an alternative to an already implemented teleport which is available in the SteamVR package. This one has already implemented fading effect and a parabolic pointer which makes the teleportation easier to visualize to users.

[1] SteamVR Plugin available at: https://www.assetstore.unity3d.com/en/#!/content/32647
[2] HTC Vive Teleportation System available at: https://github.com/Flafla2/Vive-Teleporter

7. Used assets

The asset is a representation of any item which can be used in a project. There are assets which can be imported from external sources like sounds of models, but there are also assets which are provided in Unity such as Animator Controller or Audio Mixer. [11]

In this application, I used mostly open-source assets from the Unity Asset Store[3]. List of these assets is presented below. All other assets were made by me or were a part of the Unity Standard Asset Package. Last assets I used were the testing samples which were provided by my thesis supervisor for the purpose of testing.

7.1. List of assets from the asset store

SteamVR Plugin - https://www.assetstore.unity3d.com/en/#!/content/32647

Free SpeedTrees Package - https://www.assetstore.unity3d.com/en/#!/content/29170

Classic Skybox - https://www.assetstore.unity3d.com/en/#!/content/24923

Wooden Floor Pack - https://www.assetstore.unity3d.com/en/#!/content/31492

7.2. List of remaining assets

Carrier beep sound – https://www.soundjay.com/button/sounds/beep-02.mp3

Controller images in the tutorial room – http://media.bestofmicro.com/G/D/578605/original/HTC-Vive-retail-controllers.jpg

Bubble pop sound – made by me

Texture of the bubbles – made by me

7.3. List of prefabs

Prefab is an asset [11] type in Unity which can be useful when several objects with the same properties and components are needed. Prefab makes it possible to store a GameObject complete with all its components and properties, and it acts as a template. This saved prefab can be then instantiated as desired. [14]

TextBubble – prefab used for instantiating options in voting. This prefab has a child GameObject which is a 3D text used to display text.

TargetBubble – prefab used for instantiating targets in the scene.

[3] Unity Asset Store available at: https://www.assetstore.unity3d.com/

8. Scripts

In this section, I will describe all scripts used in the application. They were all written by me in C#.

In Unity, every object in the scene is controlled by its Components. A script is also a component, and it allows us to trigger events, modify other Components and respond to user input in any desired way [10].

8.1. GameControllerScript

The main script which controls the whole application. Its main function is to keep the game loop in motion and to communicate with all other important scripts. The script is in the Main scene because it controls the testing process. Its main function is to communicate with other key scripts.

Among others, this script is responsible for the game loop. To adjust the game loop, we can set the delay between samples and number of samples after which there will be a break.

GameControllerScript takes care of initializing the testing when the Main scene is entered. It calls the FileController and gets some samples and the first sample which is going to be played in the test session according to a randomization. It also retrieves how many samples there are in a session so the application can be stopped after finishing the last sample. Then is starts spawning routine by calling the TargetSpawner and starting the countdown until next sample.

When the countdown ends, a sample is played, and bubbles with choices are spawned around the listener. GameControllerScript announces voting to all the scripts which need the information and waits until the listener chooses an option.

When the listener votes, PlayerScript announces the choice number to GameControllerScript, and it forwards it to FileController that saves all the votes. Information about ending the voting period is announced to all scripts that behave according to it and the countdown to the next sample is started again.

This simplified sequence of actions is repeated until there's time for a break in between the samples or until the test is over.

8.2. PlayerScript

The script which is a component of one of the controllers. The script controls all shooting in the scene by handling the trigger of the controller. When the trigger is pressed, the script tries to recognize the object the player is trying to shoot and if an object is recognized, the desired action happens. In the case of bubble targets, a bubble is popped, and listener's score is incremented, in the case of the vote bubble, a choice is logged into a file, and all bubbles are popped. In the case of shooting during the break, testing is resumed, and in the case of finishing the testing, the application quits.

8.3. Highlighter

The script which is attached to the same controller as the PlayerScript. It highlights all highlightable objects in all scenes by assigning them a different color.

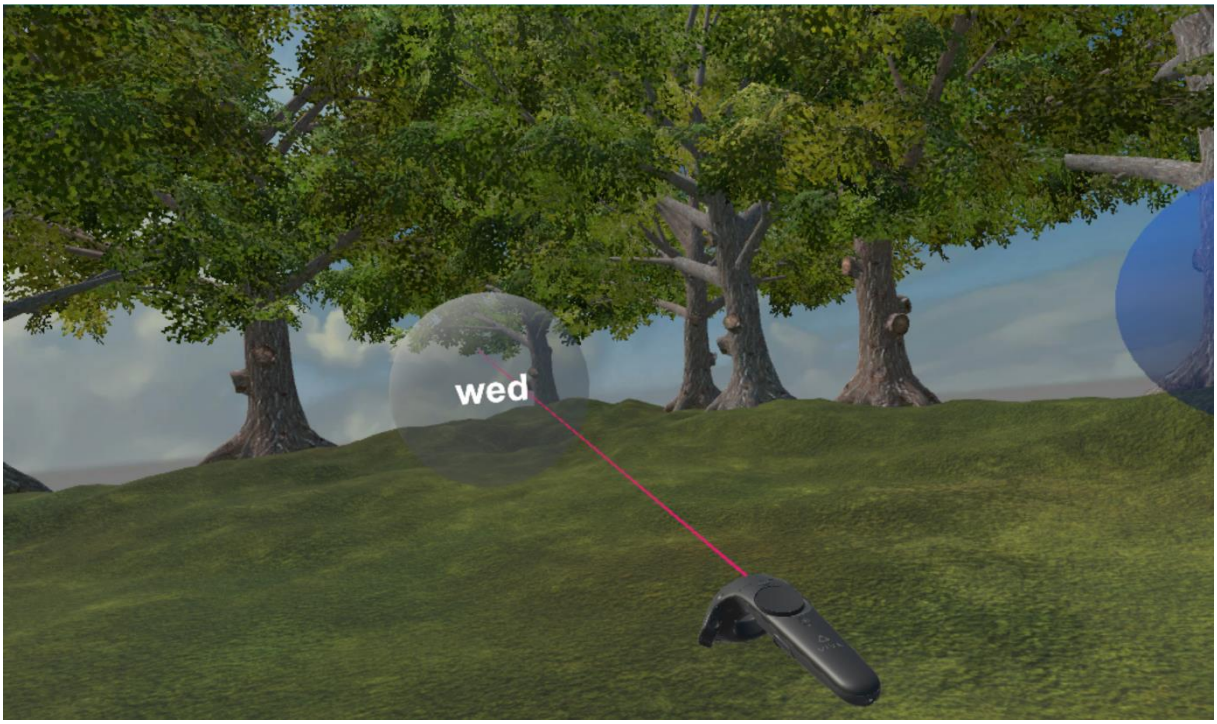

Figure 1: Example of a highlighted target (application screenshot)

8.4. SpawnBubbles

The script which takes care of loading all the words used in the test and spawning choices for every played sample. According to the randomization made at the start of the application, it takes proper choices for current sample and places one choice into every TextBubble. It also assigns every bubble a choice number which is later saved into the log file. Choice numbers correspond to the order in which the words are loaded.

Figure 2: Example of spawned choices (application screenshot)

8.5. SpawnTargets

The script which controls spawning target bubbles. It computes a random interval in which the bubbles should spawn and the location of the spawning and also the target's color is picked randomly. When the voting process takes place, it clears the area around the player so no targets are in the way and until the voting ends, it takes care of not spawning any new bubbles in the current voting area.

Figure 3: Example of randomly spawned targets (application screenshot)

8.6. VoteBubbleScript

The script which is a component of every TextBubble prefab. It controls the movement of the bubble and has methods to set its text and its number which is later logged. It also controls the destroying process which consists of sound effect, animation and a particle effect.

8.7. BubbleScript

The script which is a component of every TargetBubble prefab. It controls movement and after reaching a certain height it pops the bubble, so there aren't millions of targets in the scene. It also controls the destroying process which consists of sound effect, animation and a particle effect.

Figure 4: Particle effect played when destroying the target (application screenshot)

8.8. ControlSounds

The script which is a component of an AudioSource GameObject. This script can be called for several reasons; the main one is playing the samples. It can also be called to play a sound effect when popping a bubble or to play an announcing beep sound before the sample.

8.9. PauseSession

The script which is responsible for displaying the pause screen when the listener should take a break. There's a countdown during which the controllers are disabled so there's no accidental resuming the test.

8.10. EndSession

The script which is responsible for displaying the end screen of the test.

8.11. FileController

The script which controls loading or creating a log file at the start of the test. During every break and after the test ends, it writes all gained votes into the log file. It also provides a total number of samples.

8.12. TutorialPlayerScript

Script similar to PlayerScript but in the tutorial scene where it controls the shooting.

8.13. TutorialStarter

The script attached to the interactive board in the tutorial scene. It controls the whole tutorial section which consists of two voting actions and controls spawning targets. It also starts the main test.

8.14. FlyAway

The script which controls the animation of the roof in the tutorial scene. It plays the animation when the tutorial round is started to "open" the roof and prevents the room from overcrowding with targets.

9. Scenes

The scene in Unity contains all objects of the game. One scene can be used to create a menu, one level or anything else. [13] In my application I have two scenes in total. One is the "Tutorial room" which is the first thing to appear when starting the application.

The second scene is called "Main" and as the name suggests, this scene is the place of testing session. The player is transferred there after completing all the tasks in the tutorial scene.

Both scenes have several objects which are the same. The most important one is the CameraRig from SteamVR package[4]. It contains both controllers including their models, head-mounted-display, which means the camera and takes care of displaying the tracking area. That means that in case the listener is too close to stepping out of it, a blue grid will appear to indicate, that he shouldn't move any further in that direction.

Tracking the controllers is also extremely helpful because the listener can see the same controller in the virtual reality as he holds. It can even be visible that a button is pushed on the model controller.

9.1. Tutorial scene

The most important GameObject in the Tutorial scene is the Room, which contains parts of the room where the listener is. The floor of the room is Terrain GameObject which had to be used because of the Vive Teleport[5], which works only on terrains.

The rest of the objects are the boards with instructions that appear on the walls. The last board has a TutorialStarter script attached to it. Last GameObject is a Teleporter which contains all required assets from the Vive Teleport. The last object is the SoundController which is an AudioSource used to play sound effects and samples.

GameObjects which aren't at the scene from the beginning are the target bubbles and the bubbles used for voting. Those are instantiated during the tutorial section from prefabs.

9.2. Main scene

In the Main scene, there are GameObjects for all the most important scripts, so the scripts that are well-arranged in the scene can be easily found when editing is needed. I tried to make the tests as customizable in the editor as possible. Changing the delay between votes can be changed right in the editor, as well as some samples which have to be completed before a break. Those properties can be changed by GameController script on the GameController GameObject.

[4] SteamVR Plugin available at: https://www.assetstore.unity3d.com/en/#!/content/32647
[5] HTC Vive Teleportation System available at: https://github.com/Flafla2/Vive-Teleporter

Figure 5: The Main scene (application screenshot)

Another customization can be done by the SpawnBubbles script on the GameObject ChoiceSpawner. There we can set the desired prefab of choice, set the number of choices which are going to be available and even set the radius of the spawned bubbles. The last thing that can be set is the rotation target which makes all the choices rotate towards a certain GameObject. In this case, it is set to CameraRig, so the choices always face the listener.

Figure 6: Voting process in the Main scene (application screenshot)

There's also the same Teleport GameObject as in the Tutorial scene with all necessary assets from Vive Teleporter[6]. And the last GameObject is the Terrain which affects the appearance of the scene. The terrain was built in Unity by using its default tools for shaping the terrain and placing the trees in the scene.

[6] HTC Vive Teleportation System available at: https://github.com/Flafla2/Vive-Teleporter

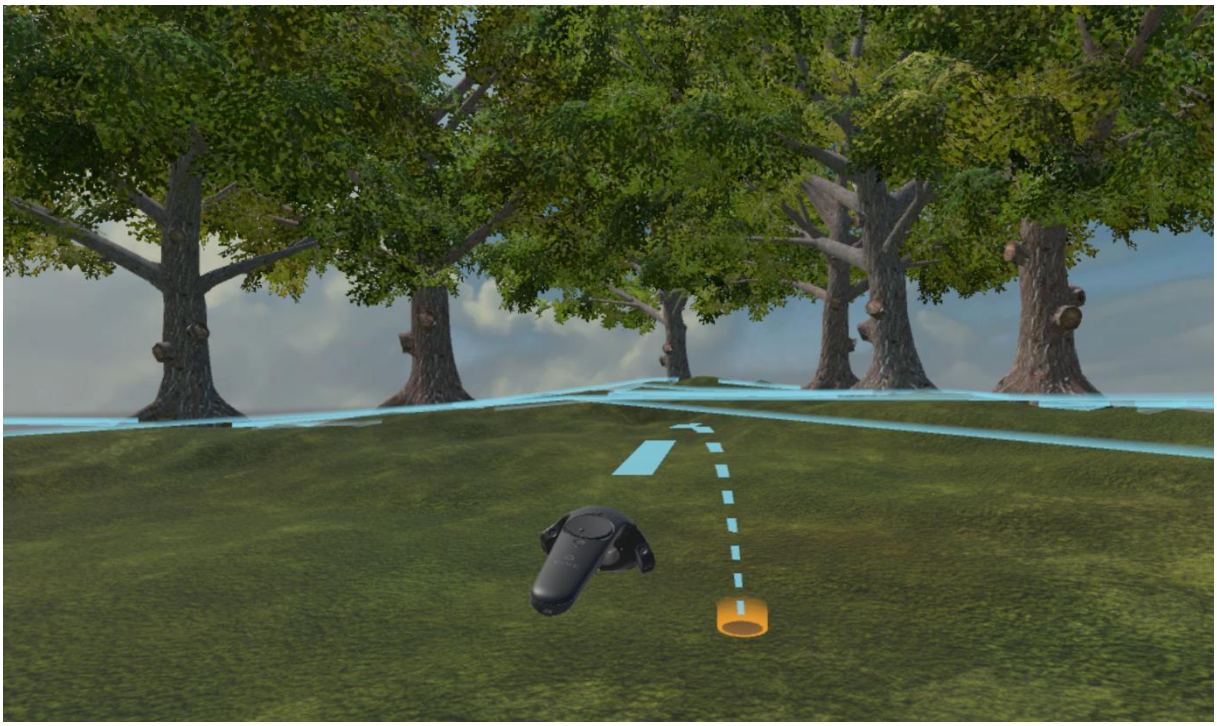

Figure 7: Using teleport (application screenshot)

10. Test implementation in VR

To make intelligibility testing possible in the VR, the logic of the game had to be implemented to fulfill the basic needs of the testing.

1. Perform secondary task
2. Listen to sample and pick a correct word from visible choices
3. Resume secondary task.
4. Repeat until all samples from current session are played

There are several ways of timing this loop. The first option is to perform a secondary task for a certain amount of time or to perform it until a certain condition is met, for example until ten targets are hit. In this case, I decided to implement the first option. It could be possibly used as a motivation because a competitive element can be added to a secondary task when there's a time limit.

Another problem in the loop was timing the voting process. The first option was to start a timer after the sample is played and after a few seconds, the choices would disappear, and the secondary task would resume. This would be efficient timewise, but there could be empty votes in the data, which is a problem in further data analysis. To prevent this situation, I preferred to keep the voting process unlimited and to let every listener take all the time they needed. Since the sample can be played only once, the votes took just a few seconds in my demonstrative tests. The listener is suddenly interrupted from the game and still can see the targets floating and spawning but can't shoot them. This produces stress towards the listener and time pressure. The only timer that was used in the voting process was for blocking the shooting controller to minimize accidental voting.

In general, the secondary task consists of locating, highlighting and destroying moving objects in the scene. The objects are represented by floating bubbles which randomly spawn around the whole area and float up until 15 meters where they pop by themselves to prevent spawning millions of instances.

The bubbles spawn at random places in random intervals, so the localization requires a certain amount of movement. The controller used to shoot targets has a ray coming out of it so it can be identified from the other controller which is used for teleportation. The ray can have a limited range which would require usage of the teleport to get closer to target and shoot it. The problem is that controls seem to be quite complicated as they are so the ray length is set in a way so that every spawned bubble can be popped no matter where it appears.

The testing procedure starts after every ten seconds of "free shooting." In this period the listener tries to get as high score as possible. The sample is then introduced by a beep sound which is a replacement for carrier sentence that can be used in such tests. After the sample is played, bubbles with word choices written in them spawn in a circle in listener's current location. All the words are facing the listener and to see all the options it is necessary to look around. The desired word can be picked by popping its bubble just like all regular target bubbles. After that, the countdown is on again for the secondary task performing.

During audio tests, listeners take breaks after every several minutes. In a standard laboratory environment, the parallel task usually doesn't have such impact on the health of the subject, so it is important to take a break during testing in VR. A long stay within the virtual reality may cause trouble to the subject and could have an impact on the results. I decided to announce a break to the listener after every 16 samples which in this case means after every third of the test.

10.1. Application preparation

For the proper run of the application, there are several files that need to be prepared before the start. In a folder, which contains build of the application, there should be a folder named TestData. Before the start, this folder should contain two files. The first one is MRT_words.txt, which is a file that contains all choices for all samples that are played in the test session. Each line in this file belongs to one sample. The second file is mrt_res.csv, which is a file that contains two lines and is in the save format as the log file which is generated later during the testing. The first line of this file contains all the sample numbers in the test and the second line contains correct answers of the samples. This file is necessary to create a proper log file.

Every time the application starts to run a new testing session, the first thing is checking the existence of a log file. To make processing the results easier, all sessions are saved into one logfile which is in .csv format. CSV, comma-separated values is a file format which consists of lines that include values separated by commas. It can be opened by MS Excel, which is important for further data processing and analysis. Every session the existence of a log file is checked. If the file exists, it will be opened for appending more data, if not a new one will be created. In the file, each line has its number which represents a session, and it's followed by all the votes.

Another important action which takes place at the start of a session is randomization. As I already mentioned, it is done by Fisher-Yates shuffle algorithm. The samples are then played in randomized order, and the answers are saved into an array. After a break is announced, all current data is saved into the log file in case of some unfortunate event or problems with the application. After the testing all the votes are saved again.

10.2. Instructions

Speech intelligibility testing always includes instructions of some kind. In this case, listeners are going to be told about the system and the controls before they put on the head-mounted display, but the application itself is going to have a "tutorial room" for the listeners to get used to the controls and to test the aiming and teleportation. The "tutorial room" has instructions on the walls in a very brief form. There are in total six panels with instructions.

Figure 8: Tutorial room instruction panels (application screenshot)

The first panel has general information about the upcoming intelligibility testing. It tells the number of samples that will be played and describes the main task which is listening to all the samples and picking the right option.

The second panel introduces the secondary task, which is shooting the targets. It encourages the listener to shoot as many targets as possible.

The third panel describes the workflow of the testing in 4 simple steps, so it's easy to understand and to remember.

The fourth panel describes the controller which is used for shooting and has a picture of it with highlighted trigger to express which button has to be used.

The fifth panel shows the highlighted touchpad on the picture of a controller and describes the teleportation.

The sixth panel, which is the last one, is highlightable and has an instruction written on it. The instruction says that to start the tutorial, the user has to shoot the panel. After shooting it, the tutorial section starts. It consists of playing two samples and is important because it introduces the whole process to the subject. Between the two samples there are five seconds for practicing the target shooting. After completing the tutorial section, it should be obvious to the listener, how will the test will continue and how to complete it.

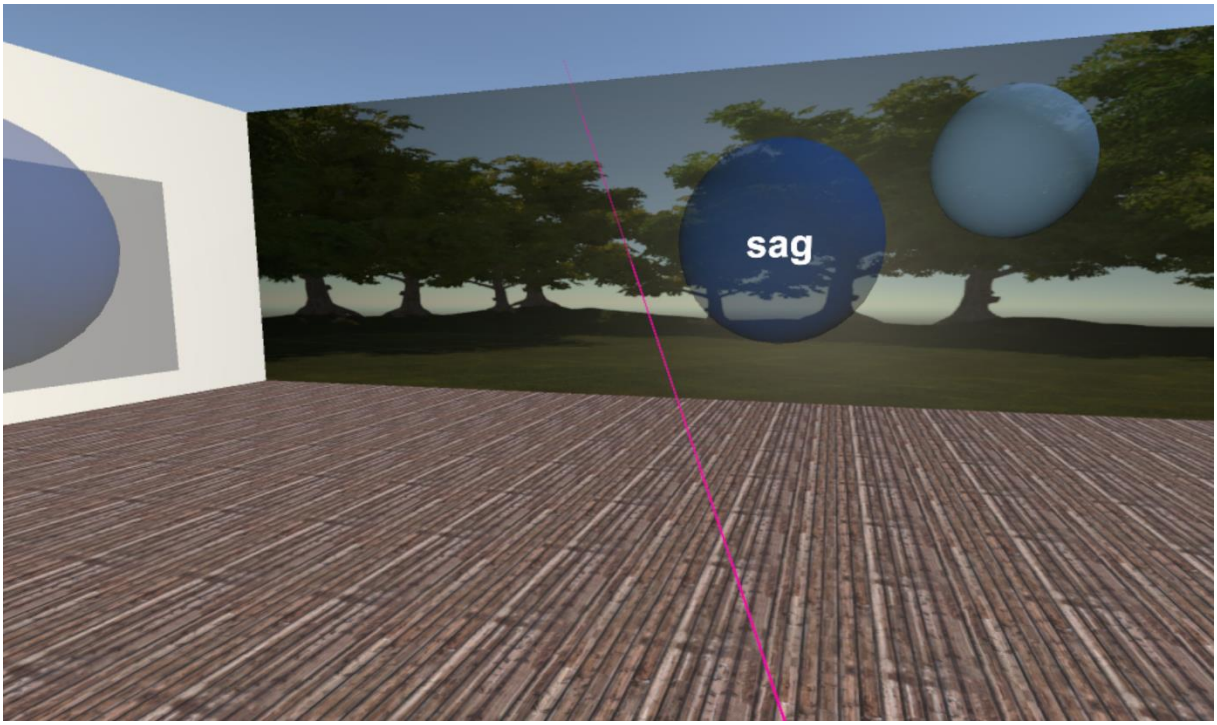

Figure 9: One of the practice votes in Tutorial scene (application screenshot)

After the second vote is completed, the interactive board changes its text and by shooting it the testing session starts in the Main scene, which is different from the Tutorial one.

10.3. Break in the testing

After every 16 samples, the listener should take a short break to regenerate a little to continue with the testing. That means there are two breaks in the whole testing session. A break will be announced by displaying a panel in front of the listener with instructions. The listener will be instructed to take off the equipment and take a short break. Resuming the test is done by shooting with the controller with ray. To prevent accidental shooting right after the panel is displayed, the shooting is disabled for 10 seconds so the listener has enough time to realize what should be done. After 10 seconds the test can be resumed.

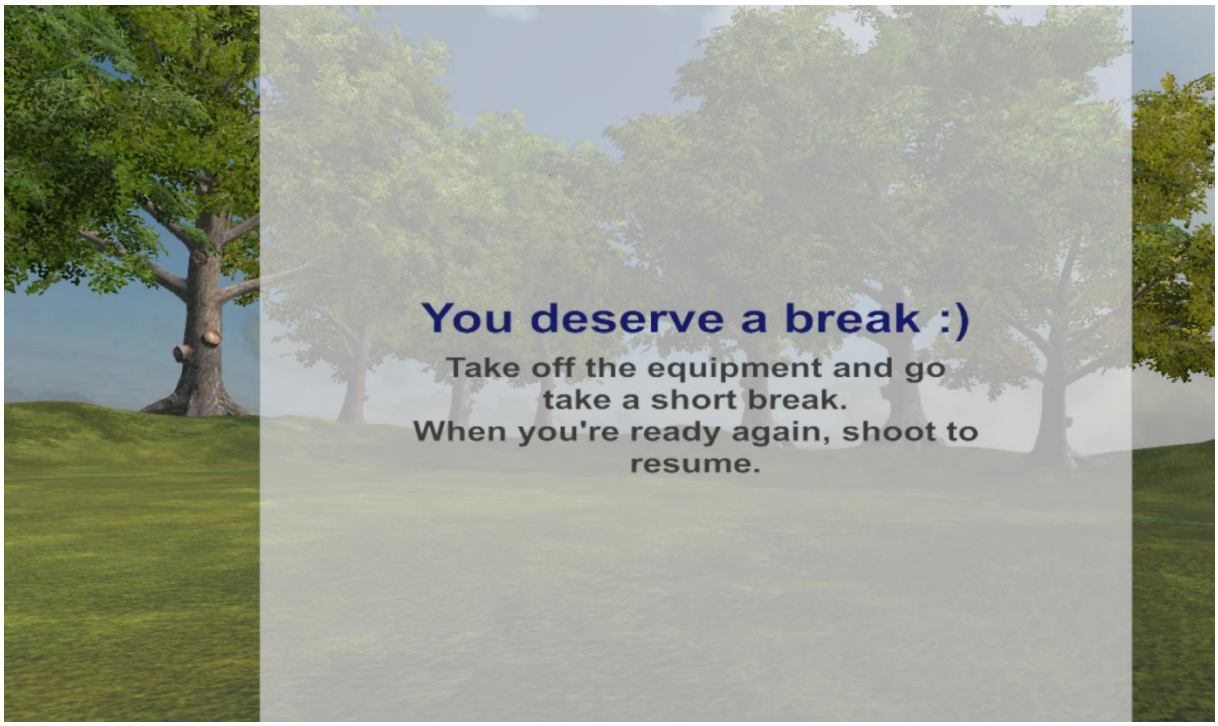

Figure 10: Panel displayed when pause is scheduler (application screenshot)

10.4. End of the testing

After all samples are played and all votes are logged, all targets are destroyed, and a panel with a thank you note is displayed. The panel also has a score written on it to tell the listener how well he did in the secondary task. Similar to the panel with pause instructions, this panel can also be closed by shooting with the controller which is meant for shooting, and this will close not only the panel but also the whole application.

10.5. Application modification

In the introduction, I said that the goal of this project is to build a sustainable application that can be used for audio testing purposes. I will be using it for the MRT testing, but it is possible to change the application to perform different types of tests with parallel load or just to modify the current test.

Modification of the wordlist or the test results is very simple. All that has to be done is replacing or editing proper files in the TestData folder which is located in the same folder as the built application.

In the case of changing the wordlist, so there is a different number of options for each sample, it is possible to set some choices in the Unity Editor when we open the project and its Main scene. It is also possible to change the radius of the appearing samples or the delay between samples. After changing any of these settings, the application has to be built again to use the newly set properties.

The most complicated is changing of all the samples. Samples are located in "Assets/Resources/samples" folder. Samples are the only test-related data that

have to be loaded from the Resources folder. I wasn't able to implement a suitable alternative for loading sound clips from the external location, so the application has to be built again when making changes in the "samples" folder.

By combining all the changes above, we could easily modify the application for the DRT testing. We would replace our list of words by a file that has only two words in a line; then we would put new samples in the "samples" folder and build the application. These changes could also be used for the different voting system. For example, it would be possible to perform quality testing on the parallel task by using the MOS, Mean Opinion Score, which has five options to select. However, this is just an example of possible quick modification; the quality testing would have to be implemented on the ITU-T P.800 standard [12].

11. Demonstrative test

To prove functionality of my application, demonstrative tests were held to acquire data which can be used for comparison. Subjective tests took place at VR Lab at Karlovo Namesti. Date of the testing was 28th April 2017, and the tests were performed throughout the entire day. There were ten listeners in total from which six had never used a device which provides virtual reality or had used it just very briefly, and four of them could be called more experienced users.

At the beginning of the test, every subject was instructed about controls and the testing process. All instructions were repeated in the application itself, the listener had enough time to ask any questions about the testing before starting the tutorial section. Then the whole test took place. I didn't measure the duration of every test separately, but every listener took about 15 – 20 minutes to complete the whole test.

11.1. Motivating of the subjects

It is important for every subjective test to have a proper motivation for the participants. In some cases, people get paid to participate in subjective tests like this one. However, I decided to count on the curiosity of several of those who had never experienced virtual reality, and I also offered snacks which turned out to be a satisfying reward for the most of them.

During the implementation, I decided to add competition to the testing by setting the free-shooting period to ten seconds, and this also turned out to be a motivation for at least a half of my participants because they wanted to achieve a good score.

11.2. Reactions of listeners

During the testing, I made several observations. The first one was that the tutorial room is not designed in the most user-friendly way. In the future, it may be more appropriate to reimplement the user interface so that it's clear what is the next action that should be done. It happened a few times, that the order of instruction boards wasn't clear or that someone didn't notice changing the text on the interactive board, so they didn't know how to start the main test. It even happened that the order of the boards wasn't clear.

On the other hand, there was no problem in understanding the testing process. It seemed that everyone had understood the process right at the tutorial section and they didn't need any more explanation in the testing process later. I didn't notice any struggle in the test section with the controls, but the teleport was not used by many users. This may be caused by the fact that it wasn't necessary to use it to get closer to the targets. From a total of 10 users, 2 used it more than once to get closer to a stuck target or in a similar situation, and 3 used it once to try it out and then were probably too distracted to use it again.

Another observation is that majority of participants didn't need the break between every 16 samples. The flow of the testing was quite quick, and almost everyone just waited for the compulsory 10-second break and then resumed the test right away. None of the listeners had experienced headache or any other form

of cybersickness. This could be because of the short time which was spent in virtual reality environment. I asked the listeners about their medical condition after and during the tests, and there was no report of any medical complication. Due to this fact, I believe that the final data shouldn't be affected by the cybersickness at all.

The shooting score was counted for every listener, so I constructed a table to see if the score has anything to do with the VR experience of the listener.

Subject number	Score	VR experience level
1	150	Low
2	159	Low
3	129	Low
4	207	Low
5	134	Low
6	182	High
7	145	High
8	157	High
9	196	Low
10	221	High

Table 1: Subjects' scores

We can see that two of the three highest scores were scores of people who hadn't had any experience with virtual reality. It may have been caused by their experience with video games, but I did not verify this possibility.

The average score of a less experienced user is 162.5, the average score of an experienced user is 176.25. The difference between these two values is 13.75 which is a value that may be partially caused by the randomization in spawning process.

Figure 11: One of the participants (photo from testing)

12. Data analysis

Next step is to analyze acquired data and compare it with results from a laboratory experiment. Since my experiment was only demonstrative, the proper statistical analysis was not constructed as it is not fully possible with such little data. Further on I will describe performed analysis.

12.1. Data preparation

As I mentioned before, the data is logged into a CSV file which can be further modified. During data analysis, I found out that the CSV file is a reasonable choice for logging the data but not for its further analysis because graphs and functions don't save in this format. To process data without any loss or complications, the file should be opened and saved as an XLSX file for using formulas, creating graphs and so on.

12.2. Data comparison

The MRT test is a set of repeated trials in which we can label each one as a success or a failure. Since every sample has six choices, the expected limit for the probability of success is one-sixth. Also, [8] specifies a transformation that maps the success probability to intelligibility and is denoted as R.

$$R = \frac{6}{5}\left(\hat{p} - \frac{1}{6}\right)$$

Where $\hat{p} = \frac{1}{6}$(the success rate for guessing) to R = 0. It also maps $\hat{p} = 1$ to R = 1.

In the case of the proper experiment with some subjects stated in [2], there would be possible to perform further analysis of the data and to consider the question of statistical significance. In that case, we would posit a null hypothesis and perform a statistical test to confirm or reject it. It is possible to perform for example chi-squared test on the data as stated in [9]. I did not perform this statistical analysis because with this small set of data the results wouldn't have much corresponding value.

To compare my results, I took the graph from [8] which shows a comparison of laboratory data and parallel task data. I then reconstructed the graph with the field data replaced by data from my experiment. Because of the size of the graph and its information value, the graph can be found in Appendix B.

In the original graph, we can see that intelligibility at some samples was way higher when testing with the parallel task. The samples 8, 14, 17, 27, 32, 34, 36 and 42 are the example of this fact.

In my graph, samples 14 and 34 were also the example of better intelligibility with the parallel task, but it was also the case of samples 35 and 41. Another anomaly was in the sample 36 which didn't have a single correct vote in my demonstrative test. For better comparison of these results, I would need a higher number of votes. This comparison is rather just informative, but I would say that the usage of

virtual reality is suitable for this testing since in several cases, the results seem similar.

13. Conclusion

The main goal of this thesis was to implement an application that can be used for intelligibility testing on MRT or ITU-T P.807 methodology and to perform the demonstrative test. Another step was to compare acquired data to already existing data from a different measurement.

I believe I fulfilled all given goals. The demonstrative test was performed in my application which is in the final working state for the demonstrative experiment and can be further modified and expanded to become suitable for different testing scenarios. The results were compared to actual test results acquired from a different measurement which was performed in the laboratory. The analysis may be affected by the amount of gained votes, but there are cases in which the already performed parallel task data matches similar conclusions as for the data from virtual reality testing.

14. Future steps

I consider my application a proof of concept in this field and believe that virtual reality environment is an opportunity for intelligibility testing. In the future, I would like to perform a proper test with a similar number of subjects as in [8] to create a better comparison of the results of both measurements.

Another possibility is to expand the application and provide multiple environments which could be changing as the test proceeds. The same change could be done with the secondary task logic, where we could have multiple game mechanics that could switch during the test.

There are possibly many more ways to improve and modify this application so it can fit the intelligibility testing needs even better. I would like to proceed to research these possibilities further in my diploma thesis in the future.

Resources

[1] – Speech Intelligibility: Summary of Speech Intelligibility Testing Methods [online]. [cit. 2017-05-11]. Available at: http://www.dynastat.com/Speech%20Intelligibility.htm

[2] - ITU-T Rec. P.807 - Subjective Test Methodology for Assessing Speech Intelligibility, Series P: Terminals and Subjective and Objective Assessment Methods. Geneva, 2016.

[3] - MCDONALD, Pierce. Modified Rhyme Test DETERMINATION OF COMMUNICATION PERFORMANCE TEST FOR SPEECH CONVEYANCE AND INTELLIGIBILITY OF CHEMICAL, BIOLOGICAL, RADIOLOGICAL, AND NUCLEAR FULL-FACEPIECE AIR-PURIFYING RESPIRATOR STANDARD TEST PROCEDURE [online]. In: . 2005 [cit. 2017-05-11]. Available at: http://slideplayer.com/slide/4516681/

[4] – SPEC COMPARISON: Does the Rift's Touch update make it a true Vive competitor? Digital Trends [online]. 2016 [cit. 2017-05-11]. Available at: http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/

[5] – WEIGHTMAN, Margaret M., Karen MCCULLOCH. Mild Traumatic Brain Injury Rehabilitation Toolkit: Dual-task assessment and intervention. 2014.

[6] – Fisher-Yates shuffle. Programming-Algorithms.net [online]. 2012 [cit. 2017-05-11]. Available at: http://www.programming-algorithms.net/article/43676/Fisher-Yates-shuffle

[7] – ANSI/ASA S3.2-2009 (R2014): Method for Measuring the Intelligibility of Speech over Communication Systems. 2009.

[8] - Avetisyan, Hakob: Subjective Speech Quality Measurement: Comparison of Laboratory Test Results and Results of Test with Parallel Task, accepted for ETSI Workshop on Multimedia Quality in Virtual, Augmented or other Realities. 2017.

[9] – VORAN, Stephen D., Andrew A. CATELLIER. Speech Codec Intelligibility Testing in Support of Mission-Critical Voice Applications for LTE: NTIA Technical Report TR-15-520. 2015.

[10] - Creating and Using Scripts. Unity - Manual [online]. 2013 [cit. 2017-05-11]. Available at: https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

[11] - Asset Workflow. Unity - Manual [online]. 2015 [cit. 2017-05-11]. Available at: https://docs.unity3d.com/Manual/AssetWorkflow.html

[12] – ITU-T Rec. P.800 – Methods for subjective determination of transmission quality, Series P: Telephone Transmission Quality. Geneva, 1996.

[13] - Scenes. Unity - Manual [online]. 2015 [cit. 2017-05-11]. Available at: https://docs.unity3d.com/Manual/CreatingScenes.html

[14] - Prefabs. Unity - Manual [online]. 2014 [cit. 2017-05-11]. Available at: https://docs.unity3d.com/Manual/Prefabs.html

[15] - LAVIOLA, Joseph J. A discussion of cybersickness in virtual environments. ACM SIGCHI Bulletin [online]. 2000, 32(1), 47-56 [cit. 2017-05-13]. DOI: 10.1145/333329.333344. ISSN 07366906. Available at: http://portal.acm.org/citation.cfm?doid=333329.333344

[16] - HOUSE, Arthur S., Carl WILLIAMS, Michael H. L. HECKER a Karl D. KRYTER. Psychoacoustic Speech Tests: A Modified Rhyme Test. The Journal of the Acoustical Society of America [online]. 1963, 35(11), 1899-1899 [cit. 2017-05-15]. DOI: 10.1121/1.2142744. ISSN 0001-4966. Available at: http://asa.scitation.org/doi/10.1121/1.2142744

[17] – The 300 Stimulus Words of the MRT. Speech Intelligibility Papers - MRT List [online]. [cit. 2017-05-17]. Available at: http://meyersound.de/support/papers/speech/mrtlist.htm

Appendix A – MRT word list

went	sent	bent	dent	tent	rent
hold	cold	told	fold	sold	gold
pat	pad	pan	path	pack	pass
lane	lay	late	lake	lace	lame
kit	bit	fit	hit	wit	sit
must	bust	gust	rust	dust	just
teak	team	teal	teach	tear	tease
din	dill	dim	dig	dip	did
bed	led	fed	red	wed	shed
pin	sin	tin	fin	din	win
dug	dung	duck	dud	dub	dun
sum	sun	sung	sup	sub	sud
seep	seen	seethe	seek	seem	seed
not	tot	got	pot	hot	lot
vest	test	rest	best	west	nest
pig	pill	pin	pip	pit	pick
back	bath	bad	bass	bat	ban
way	may	say	pay	day	gay
pig	big	dig	wig	rig	fig
pale	pace	page	pane	pay	pave
cane	case	cape	cake	came	cave
shop	mop	cop	top	hop	pop
coil	oil	soil	toil	boil	foil
tan	tang	tap	tack	tam	tab
fit	fib	fizz	fill	fig	fin
same	name	game	tame	came	fame
peel	reel	feel	eel	keel	heel
hark	dark	mark	bark	park	lark
heave	hear	heat	heal	heap	heath
cup	cut	cud	cuff	cuss	cud

thaw	law	raw	paw	jaw	saw
pen	hen	men	then	den	ten
puff	puck	pub	pus	pup	pun
bean	beach	beat	beak	bead	beam
heat	neat	feat	seat	meat	beat
dip	sip	hip	tip	lip	rip
kill	kin	kit	kick	king	kid
hang	sang	bang	rang	fang	gang
took	cook	look	hook	shook	book
mass	math	map	mat	man	mad
ray	raze	rate	rave	rake	race
save	same	sale	sane	sake	safe
fill	kill	will	hill	till	bill
sill	sick	sip	sing	sit	sin
bale	gale	sale	tale	pale	male
wick	sick	kick	lick	pick	tick
peace	peas	peak	peach	peat	peal
bun	bus	but	bug	buck	buff
sag	sat	sass	sack	sad	sap
fun	sun	bun	gun	run	nun

Table 2: Used MRT word list ([17])

Appendix B – Data Comparison Graph

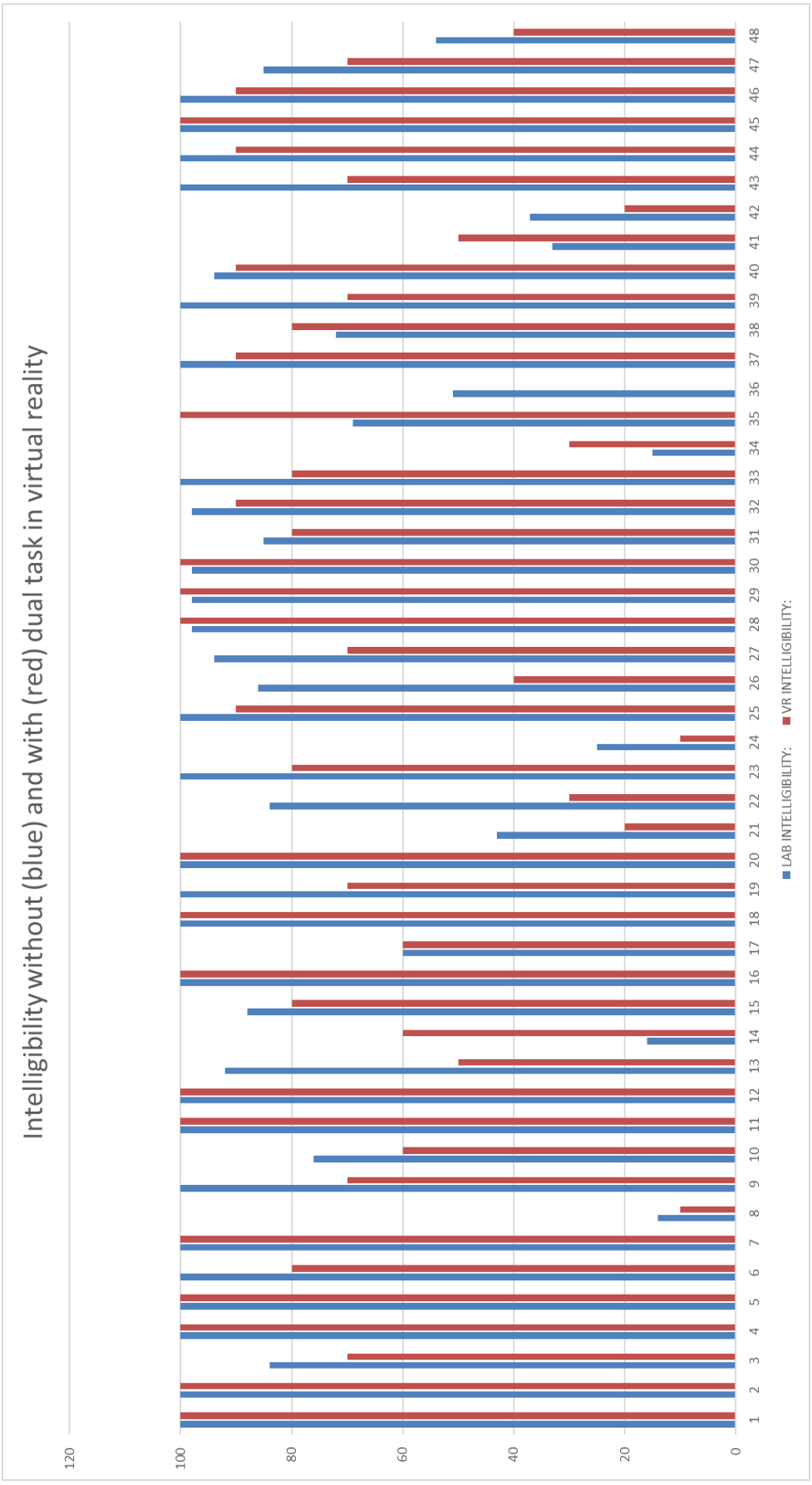

Figure 12: Graph of intelligibility comparison (data from my experiment and [8])