University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

ACE-VR: A Virtual Reality Assissted Dementia Screening Test

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Michael Thomas P. Ramos

June 2018

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "ACE-VR: A Virtual Reality Assissted Dementia Screening Test" prepared and submitted by Michael Thomas P. Ramos in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Perlita E. Gasmen, M.Sc. (*cand.*)
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*cand.*)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Richard Bryann L. Chua, M.Sc.	_____	_____
4. Ma. Sheila A. Magboo, M.Sc.	_____	_____
5. Marvin John C. Ignacio, M.Sc. (*cand.*)	_____	_____
6. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
7. Geoffrey A. Solano, Ph.D. (*cand.*)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

Dementia is a syndrome - usually of a chronic or progressive nature - in which there is deterioration in cognitive function beyond what might be expected from normal aging. It affects memory, thinking, orientation, comprehension, calculation, learning capacity, language, and judgment.Cognitive screening tests are quick useful tools used to assess the condition of patients. The basic purpose of cognitive screening tests is to indicate likelihood of genuine cognitive impairment. Cognitive screening can be enhanced in virtual reality as all aspects of the environment can be controlled and replicated. Participants can be evaluated in an environment that better simulates the real world.

Keywords: Dementia, Cognitive Screening Test, Virtual Reality

Contents

List of Figures

I. Introduction

A. Background of the Study

Dementia is a syndrome - usually of a chronic or progressive nature - in which there is deterioration in cognitive function beyond what might be expected from normal aging. It affects memory, thinking, orientation, comprehension, calculation, learning capacity, language, and judgment. [1] The impairment in cognitive function is commonly accompanied, and occasionally preceded, by deterioration in emotional control, social behavior, or motivation. Dementia affects each person differently, depending on the impact of the disease and the person's personality before becoming ill. The signs and symptoms of dementia are categorized into three stages: early, middle, late. The early stage is often overlooked as the onset is gradual; symptoms include: forgetfulness, losing track of time, and getting lost in familiar places. In the middle stage, signs and symptoms become more distinct and restricting; symptoms include: becoming forgetful of recent events, increased difficulty with communication, needing help with personal care, and behavioral changes. The late stage of dementia is the loss of independence and activity. Memory disturbances are serious and the physical signs and symptoms become more obvious; symptoms include: difficulty recognizing relatives, increased need for assisted self-care, difficulty walking, and behavior changes that may escalate and include aggression. [2]

According to the 2016 World Alzheimer Report [2], people living with dementia have poor access to health care. Around only 50 percent of people living with dementia receive a diagnosis in high income countries. This further drops to 10 percent in low and middle income countries. As the average life expectancy increases, the number of people with dementia increases as well. By 2050, an estimated 131.5 million people will be living with dementia, 68 percent of these will be from low and middle income countries. Paired with the over-specialization of dementia care, current specialist

models of dementia care are unlikely to be able to scale up to provide sufficient coverage for the growing number of people affected by dementia - especially in low and middle income countries. People with dementia are also more likely to be admitted to hospitals, have longer stays and at are increased risk of negative outcomes. In the US alone, the cost of dementia was 818 billion USD. Estimates say that the cost could rise to 2 trillion USD by 2030. Focusing on the Philippines, a 2007 study by Ogena[3] examines how the population ageing in the Philippines will shape the old-age dependency and support ratio between the year 2000 and 2040 and analyzed the consequent prevalence and incidence of dementia. While the Philippines is expected to have a young population throughout the middle of the 21st century, the population in ages 60 years and above will increase at a very rapid rate. Results show that the rapid ageing of the Philippine population will usher in a signifcant increase in the number of old-age dependents, a likely decline in the potential support ratio, and a growing proportion of dementia sufferers. Concomitant with this, is the increased burden on the working-age population of the country.

The Diagnostic and Statistical Manual of Mental Disorders (DSM) [4] provides a guide for determining if an individual has dementia, and what may be the cause. Based on DSM's latest criteria (DSV-V), dementia is classified as a neurocognitive Disorder (NCD) because it interferes with both cognitive function and performing everyday activities.The NCD category is then further subdivided into Minor NCD and Major NCD. The NCD category encompasses the group of disorders that the primary clinical deficit is in cognitive function, which is acquired rather than developmental. The impairment may arise in attention, planning, inhibition, learning, memory, language, visual perception, spatial skills, social skills, or other cognitive functions. Dementia is considered a major NCD. A major NCD is defined by the following: evidence of substantial cognitive decline, cognitive deficits are sufficient to interfere with independence, cognitive deficits do not occur exclusively in the con-

text of a delirium, cognitive deficits are not primarily attributable to another mental disorder. There are six cognitive domains that may be affected in both Minor and Major NCD. These cognitive domains include: complex attention, executive ability, learning and memory, language, perceptual - motor - visual perception, praxis, social cognition [5].

In diagnosing dementia, doctors initially assess whether a person has an underlying treatable condition such as abnormal thyroid function, normal pressure hydrocephalus, or a vitamin deficiency. Tests used to diagnose dementia include the following: cognitive and neuropsychological tests, laboratory tests, brain scans, psychiatric evaluation, and genetic tests. Focusing on cognitive and neuropsychological tests, they are tests used to assess memory, problem solving, language skills, math skills, and other abilities related to mental functioning. Cognitive screening tests are quick useful tools used to assess the condition of patients. The basic purpose of cognitive screening tests is to indicate likelihood of genuine cognitive impairment, inferred from the relationship of the patient's score to reference norms [6]. A cognitive screening test is commonly used for: screening cognitive impairment; differential diagnosis of cause; rating severity of disorder, or monitoring disease progression [7].

There is currently no standard test for the detection of dementia. Since dementia affects each person differently, many specialized screening tests have been developed to test for specific forms of impairment [8]. Tests for Dementia include the Mini-Mental State Examination (MMSE), Mini-Cog, Montreal Cognitive Assessment (MoCA), Addenbrooke's Cognitive Examination-Revised (ACE-R), etc. The MMSE is the most used cognitive screening test around the world [9]. The MMSE has a number of limitations it is less sensitive to early dementia, has limited focus on memory and the lack of an executive functioning component. Despite the known limitations of the MMSE, it is still most used by general practitioners. This may be a contributing factor to the large number of missed dementia diagnosis, particularly at the early

stages of dementia [10]. In a review by Zarit et al [11], "The MMSE has become like a somewhat embarrassing member of the family; we all know Uncle Henry's flaws, but we continue to invite him to holiday dinner, because we cannot imagine the alternative". An alternative test that improves on the shortcomings of the MMSE is the ACE-R [12]. Results generated by the two tests are highly correlated with the ACE-R improving estimates of cognitive ability by 16 percent [13]. Some of the advantage of ACE-R include: sensitivity to the early stages of Alzheimer's and linguistic deficits found in early frontotemporal dementia, ability to distinguish between patients with progressive and degenerative disorders, and detects cognitive impairment in atypical parkinsonian syndromes [12]. ACE-R has also been found to be useful for all forms of dementia [14].

Technological advancements have brought forth an increasing number of clinical benefits to using computerized assessments in clinical studies; benefits include: easier administration, automated scoring, consistent test administration, and more accurate determination of reaction time [15]. Zygouris and Tsolaki [16] performed a review on computerized cognitive tests for the elderly. Seventeen test batteries were identified and reviewed by the researchers. The mediums to conduct these tests varied from laptops, tablets, and telephones. Ruano et al[17] developed a self-administered web based test for longitudinal cognitive testing. The test could be performed at home with just a computer. A recent emerging trend in the field of health care is the use of virtual reality. It has been proven for use in different forms of therapy, the reduction of chronic pain, relaxing patients, restoring low vision, etc. [18]

Virtual reality (VR) is an artificial environment that is created with software and presented to the user in such a way that the user suspends belief and accepts it as a real environment. There are a range of tools that can be used to further the virtual reality illusion, such as headsets, omni-directional treadmills and special gloves. The ability of virtual environments to create dynamic, immersive, three-dimensional stim-

ulus environments, offers assessment options that are not available using traditional assessment methods [19]. In VR, there is far more control in interaction rather than in other computer systems. It provides a more flexible environment and is highly programmable. It allows a wide variety of controlled stimuli. The environment can also be modeled to tailor the needs of the patient. VR is also highly immersive and can cause the participant to feel "present" in the virtual rather than the real environment [20].

VR applications are also now being used as health care tools for the elderly. Rendever is a VR platform that allows residents in homes for the elderly to explore the world virtually. It also provides cognitive therapy and tracks movement data to aid in early diagnosis of dementia. Rendever's system makes of multiple virtual-reality headsets, custom software and a tablet. The headsets are synced together, so users can explore the virtual world together [21]. The use of Rendever has shown a 40 percent increase in the happiness of the resident [22]. VR applications that focus on component cognitive processes are now also being developed and proven. These processes include: attention processes, spatial abilities, learning and memory, and executive functions [23]. VR can further cognitive tests as they can immerse the patient in a controlled situation [24]. Current available technology allows opportunities to design and develop new and more effective types of tests using VR. Within a virtual environment (VE), it is possible to systematically present cognitive tasks targeting memory performance beyond what are currently available using traditional methods. Cognitive screening can be enhanced in virtual reality as all aspects of the environment can be controlled and replicated. Participants can be evaluated in an environment that better simulates the real world. [23].

A good example of a virtual reality cognitive assessment test would be Montenegro and Argyrious work [25] on the Diagnosis of Alzheimers disease based on Virtual Environments. Their tests focused on evaluating memory loss related to common objects,

recent conversations and events, the diagnosis of problems in language skills, and the ability to differentiate between virtual worlds and reality. The researchers used VR glasses to increase immersion and a depth camera to track patient movements and animate their avatars. The entire test was also performed while the participant was seated as most Alzheimers patients are the elderly. Their virtual environment of choice was doctors office. The first test they produced was a virtual objects memorization test. The test was split into three tasks. First was the recognition of different virtual objects. Second, was testing the patients ability to remember the positioning of the objects on the table. Lastly, the patient is placed into a different room in which he/she can interact with different objects. The patient is then asked to recall each objects functionality.

B. Statement of the Problem

With the Philippines rapidly aging population there will be an increase in the number of old-age dependents. The overspecialization of dementia care will cause a lack of support in the health care of the elderly. Existing virtual reality applications lack immersion and have a large learning curve for use. A VR system to screen dementia may be a helpful tool as the cases of dementia are expected to increse in the Philippines.

C. Objectives of the Study

This research provides Virtual Reality Assisted Dementia Screening Test Mobile Application with the following functionalities:

1. Allows the patient to perform a cognitive screening test, testing five domains:
    (a) Attention & Orientation:

i. Answer questions about the current time to test their understading of their current orientation.

ii. Memorize three words and then say them; this will be done three times to test their .

iii. Subtract 7 from 100, then subtract 7 from each subsequent result for a total of 5 subtractions.

(b) Memory:

i. Recall the three words he/she was asked to memorize in a previous test.

ii. Memorize a name and an address, three trials will be given to memorize the name and address.

iii. Answer questions about well established historical facts.

iv. The patient will be asked to recall the name and address mentioned at the beginning.

v. The patient may be asked to recognize which name, of three possible options, was the one he/she was asked to memorize.

(c) Fluency: Generate as many words that start with a 'P' as possible, within 60 seconds.

i.

(d) Language:

i. Repeat three phrases.

ii. Recognize twelve images.

iii. Match which image previously shown is being described.

iv. Read five words.

(e) Visuospatial:

i. Recall where the objects on the table were previously placed.

ii. Count the number of dots on an image.

iii. Identify letters that are slightly misoriented.

2. Allows the doctor to view the patient's test score.

D. Significance of the Project

This application provides a virtual reality neurological status assessment. The application has varying objects per assessment run. The application provides an adjunct test that can be used in the screening of dementia.

E. Scope and Limitations

1. This application contains only one virtual environment.
2. This application works on Android OS smart phones with a minimum OS version of Android 4.4 (KitKat).
3. This application speech input only accepts English speech.
4. The patient test scores is not placed into a database

F. Assumptions

1. The application is run on a mobile device capable of supporting the application.
2. The smart phone needed to run the mobile application has a built-in accelerometer, magnetometer, and gyroscope sensors.
3. The smart phone is attached to a VR headset.
4. The patient is seated while using the application.
5. The patient is in a silent room.

II. Review of Related Literature

The initial attempt in validating virtual reality as a cognitive screening test was performed by Parsons and Rizzo in 2008 [23]. The project, Virtual Reality Cognitive Performance Assessment Test (VRCPAT), was a virtual environment-based measure of learning and memory. The VRCPAT was run on a Pentium 4 notebook computer, with 1 GB RAM, and a 128-MB DirectX 9 compatible graphics card. The primary aim of the project was create a VR application for the standardized assessment of memory performance within virtual environment. VRCPAT was programmed to simulate a city environment. The tests were performed with the participants seated. The participants were asked to complete tasks that would test their cognitive performance. The basis for the cognitive performance would be the learning and memory rate of the participants with regard to the task that was presented to them. They were successful in validating the utility of using virtual reality as a cognitive screening test.

Yeh et al [26] developed a cognitive screening test that focused on mild cognitive impairment. The Innovative Virtual Reality System for Mild Cognitive Impairment: Diagnosis and Evaluation project was developed in Unity and uses a head mounted display (HMD) to display the virtual scenario. The HMD is embedded with posture sensors which enable the displayed scenario to be refreshed continuously according to the test-takers head movement and visual line direction; meanwhile, the test-taker uses a joystick to carry out movement and selection operations In this program, participants are placed in a virtual convenience store in order to test executive functioning, assessment, and training. The tests include memorizing a shopping list,looking for certain goods, and checking out. The study was conducted on 90 senior subjects between the ages of 50 and 90, 60 of which are diagnosed with senile dementia, the remaining 30 subjects in the control group will be healthy people. They concluded that the study was a success and that they were able to use a virtual convenience

store to assess executive functions and the memory of participants. The researchers also reported that the test subjects readily accept new technology.

Quental et al [27] evaluated the visuospatial functions of participants using the eight-test Visual Object and Space Perception (VOSP) battery. Four tests assessed object perception and the other four tests assessed space perception. The object perceptions tests included Incomplete Letters test, Silhouettes test, Object Decision test, and Progressive Silhouettes test. The Incomplete Letters test, tasked participants with identifying 20 incomplete letters. The Silhouettes test, participants were to identify what animal or object the silhouette belonged to. The Object Decision test, the participants is presented with twenty boards with four stimuli. Only one of these stimuli represents something real. The participant is tasked with identifying and naming the stimulus that represents the real shape. Progressive Silhouettes consists of two series in which boards depicting an object are presented. The first board shows the silhouette of the object, and each successive board shows the more complete image of the object. The participant is tasked with identifying the object with the smallest amount of boards. The space perception tests are the Dot Count test, Position Discrimination test, Number Location test, and Cube Analysis. The Dot Count test tasks participants with counting the number of black dots there are on a white card. The Position Discrimination test tasks the participant with identifying in which square the black spot is located exactly in the center. Number Location tasks the patient to identify which number corresponds to the black dot. Cube Analysis tasks participants to identify how many solids (cubes) there are on each board. Thirty-one patients with mild AD and forty-four healthy elders were evaluated using a neuropsychological battery and the VOSP. In all four object-perception tests there was a significant difference between the control and the AD patients, indicating that they had greater difficulty with these activities. While in the space perception tests

only two tests, Number Location and Cube Analysis, had a significant difference in the performance of the two groups

Pedroli et al [28] adapted the Multiple Errands Test (MET) to a virtual scenario of a supermarket. The MET, usually performed in shopping malls or hospital environments, is an ecological task that is used to assess the executive functions in real-life settings. The virtual MET consists of some tasks that are usually performed in a mall-like setting or shopping center and abide by certain rules. Tasks could include buying some products or acquiring some information. Rules could be to carry out all tasks in any order or not to go into the same isle more than once. Participants are showed the shop, illustrating the different sections. Then, the doctor gives different items: shopping list, a list of rules, a shops map, some information, a white paper, a pencil, and a watch. The researchers suggested that a virtual reality MET should have the following features: ease of use, high quality visual content, visualizations of the users body, strong sense of immersion, support for external devices, and collection and integration of biosensor data

Tarnanas et al [29] paper was to establish the validity and an initial construct of a Virtual Reality Day-Out Task (VR-DOT) as a screening tool for early dementia. The researchers examined the relationships among 3 groups of participants in the VR-DOT and traditional neuropsychological tests employed to assess executive functions, to compare the performance of participants with mild Alzheimer's-type dementia (AD), with amnestic single-domain mild cognitive impairment (MCI), and healthy controls. VR-DOT focuses on the errors and patterns in performing everyday activities.The researchers also assessed functional capacity by both neuropsychological tests. The results showed that the mild AD group was more impaired than the amnestic MCI group, and that both were more impaired than the healthy controls.

The researchers concluded that VR-DOT is an effective tool for discriminating predementia and mild AD from controls.

In another study by Tarnanas et al [30] attempted to investigate whether Virtual Reality Day-Out Task (VR-DOT) could predict the evolution of patients with mild cognitive impairment (MCI) to Alzheimers disease (AD), within 1 year. They compared VR-DOT with standard neuropsychological evaluation, as well as sensitive biomarkers for short-term prognosis in MCI such as AD signature cortical thickness marker and the noninvasive ERPs. 134 patients with MCI were compared to 75 healthy control subjects. Participants were initially screened in a process that included, VR-DOT, a neuropsychological evaluation, magnetic resonance imaging scan, and event-related potentials. After 12 months, participants were screened again using the same procedure as the initial screening. The results showed that VR-DOT patient performance were strongly correlated with existing predictive biomarkers.

Cassidy et al [31] proposed a mobile framework for two types of cognitive tests: Trail Making Test (TMT) and Reaction Time Task (RTT). The objectives of the mobile application were: to reduce test administration, automate data capture from completed tests, capture additional data for analysis, reduce the possibility of human error, increase timing accuracy by using an automated timer in the TMT. The TMT consists of two parts. The app could automatically govern the rules of the test, such as whether the circles selected were in sequence. Using the Deary-Liewald RTT, a computer-based program that can run simple (SRT) and choice (CRT) reaction time tasks. In the SRT, participants were tasked to click anywhere as soon as the box on the center of the screen was filled with a X. Unlike the SRT, the CRT further challenged the participant by requiring them to click the box filled with a X. Testing the application with a sample of 29 participants, aged between 19 and 59 years, results

suggested that mobile implementations have the potential to produce reliable cognitive measurements; that are significantly correlated with existing methods. They established the added ease and accuracy that a mobile application could provide to a cognitive assessment.

Lesk et al [32] developed a virtual simulation designed to assess visuospatial memory to investigate the cognitive function with a control group and a mild cognitive impairment group. Virtual reality allows daily life tasks to be simulated. The assessment required participants to traverse a virtual path to reach a destination they were require to remember. The results suggested that their simulation had the potential to be used for the early detection of early Alzheimer's

Garcia-Betances et al [24] provided a brief review and appraisal of recent virtual reality (VR) technology for Alzheimers disease applications. Applications were categorized according to their intended purpose, focus feature, methodology, immersion level, and passive or active interaction. Critical assessment indicated that majority did not take full advantage of virtual environments with high levels of immersion and interaction. Many relied on conventional 2D displays to create non-immersive or semi-immersive VR scenarios. Improvements are needed to make VR a better and more versatile assessment and training tool for AD. This could include the use of head-mounted displays and 3D smart TV technologies, together with realistic multisensorial interaction devices and neuro-physiological feedback capacity, are some of the improvements that his review suggests. It also states the it would be desirable that such VR applications be more affordable and easily transferable to more common environments such as nursing homes.

Montenegro and Argyriou [25] developed a novel e-health Alzheimer's screening test based on virtual environments. The tests focused on the immersion of the pa-

tient in a virtual room, attempting to mislead and deceive the patient [25]. Most Alzheimer's patients are elderly hence all tasks in the virtual environment are performed while seated, requiring minimum movement. The virtual environment is set to be some place that the patient can recognize, such as the doctor's office. In the virtual environment, the patient is tasked to go through two tests: Virtual Objects Memorization test (VOM) and Virtual vs Real Sounds test (VRS). 20 participants between the ages of 23 and 82, including both healthy people and Alzheimer's patients, participated in this test. The results show that those with Alzheimer's scored significantly lower than the healthy participants. The researchers concluded that with the success of their work they proved further that virtual reality is starting to become a part of diagnosis methods. They also demonstrated the flexibility of virtual reality applications to solve the ceiling effect present in most used cognitive tests.

Sandeep et al [33] developed a Cognitive Test tool (CTT) to examine the patients cognitive function for early diagnosis of AD. The tool was a PC software developed using the C# language with Micosoft Visual Studio as the development environment. The CTT was based on the Mini Mental State Examination (MMSE). MMSE is a widely used tool to thoroughly assess mental status. The CTT software scores the patients performance and then classifies them to be either normal (24-30pts), mild dementia (18-23pts), moderate dementia (10-17pts), and severe dementia (¡10pts). The CTT provides a reliable and efficient method for determining the current stage of the disease.

Siriaraya and Ang [34] performed a case study on developing virtual environments for older users and people with dementia. Results showed that even without prior experience with 3D virtual environments, older users were capable to navigating and carrying out interaction in the virtual environment. However when dealing with

those with Dementia, using complex mechanisms could result in a loss of attention from users, making it difficult for the content of the virtual environments to become meaningful. The researchers suggested that an interaction mechanism based on items which older people are already familiar with could help people with Dementia more easily understand and process their interactions with the virtual environment.

III. Theoretical Framework

A. Dementia

Dementia is a syndrome in which multiple-domain cognitive impairment, is sufficiently severe to significantly affect everyday function. Memory and one additional area of cognitive impairment, including aphasia, apraxia, agnosia, and executive dysfunction, are required to be affected according to common criteria, DSM-V [35]. Dementia is one of the leading causes death in the UK [36]. Dementia is caused by damage to brain cells. The damage obstructs the brain cells' ability to communicate with each other. Different types of dementia are associated with the particular regions of the brain. Alzheimer's disease is caused by high levels of certain proteins inside and outside brain cells making it hard for brain cells to communicate with each other. There is no one test to determine if someone has dementia. Doctor's determine the type of dementia a patient has and its cause based on medical history, a physical examination, laboratory tests, and the characteristic changes in thinking, day-to-day function, and behavior associated with each type. The treatment of dementia depends on its cause. In the case of most progressive dementias, such as Alzheimer's disease, there is no cure and no treatment that slows or stops is progression. There are drug treatments that may temporarily improve symptoms.

B. Addenbrooke's Cognitive Examination Revised (ACE-R)

ACE-R is a brief screening test that provides evaluation of five cognitive domains. It is capable of differentiating dementia and mild cognitive impairment. It can also differentiate the subtypes of dementia [37]. The five cognitive domains are: attention and orientation, memory, fluency, language, and visuospatial; with score 18, 26, 14, 26, and 16 respectively. The cut-off ¡88 gives 94 percent sensitivity and 89 percent specificity for dementia or ¡82 gives 84 percent sensitivity and 100 percent specificity

for dementia; with norm values based on 63 controls aged 52-75 and 142 dementia patients aged 46-86.

C. Virtual Reality

The definition of virtual reality comes from the definitions of 'virtual', what is near, and 'reality', what we experience as human beings. Virtual reality by the definitions of it components means "near-reality". In our context, this refers to the emulation of reality [38]. 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 image [39].

C..1 Head-mounted Displays (HMDs)

Also known as "Virtual Reality headsets" or "VR Glasses", HMDs attach straight to your head and present visuals directly to your eyes. Most HMDs use LCD or OLED displays work by presenting each eye with an almost identical, but slightly offset, version of the same image. This provides the illusion of stereoscopy [40]. On mobile phones, if you were too look at the phone's screen without the assistance of an HMD, you would see two slightly different scenes each taking up half of the phone's screen.

C..2 Google Cardboard

Google Cardboard is a virtual reality (VR) platform developed by Google for use with a head mount for a smartphone. Named for its fold-out cardboard viewer, the platform is intended as a low-cost system to encourage interest and development in VR applications [41] Users can purchase a pre-manufactured set from Google, or

opt to build their own viewer using specifications published by Google. Cardboard-compatible applications must be on the users phone in order to use this platform.

D. Android

Android is an open source mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices. It has been the best-selling OS worldwide for smartphones since 2011. As of December 2017, there were over 3.5 million apps available in the Google Play Store [42].

E. Unity

Unity is a multipurpose game engine that supports 2D and 3D graphics and scripting using C#. In 3D games, Unity allows for the specification of texture compression, mipmaps, and resolution settings for each platform that the game engine supports. On Android devices the engine utilizes the OpenGL ES graphics API.

F. Blender

Blender is a professional, open-source 3D computer graphics software used for creating animated films, visual effects, art, and etc.Some of its features include 3D modeling, UV unwrapping, texturing, raster graphics editing, rigging and skinning. Blender has a GNU General Public License v2.

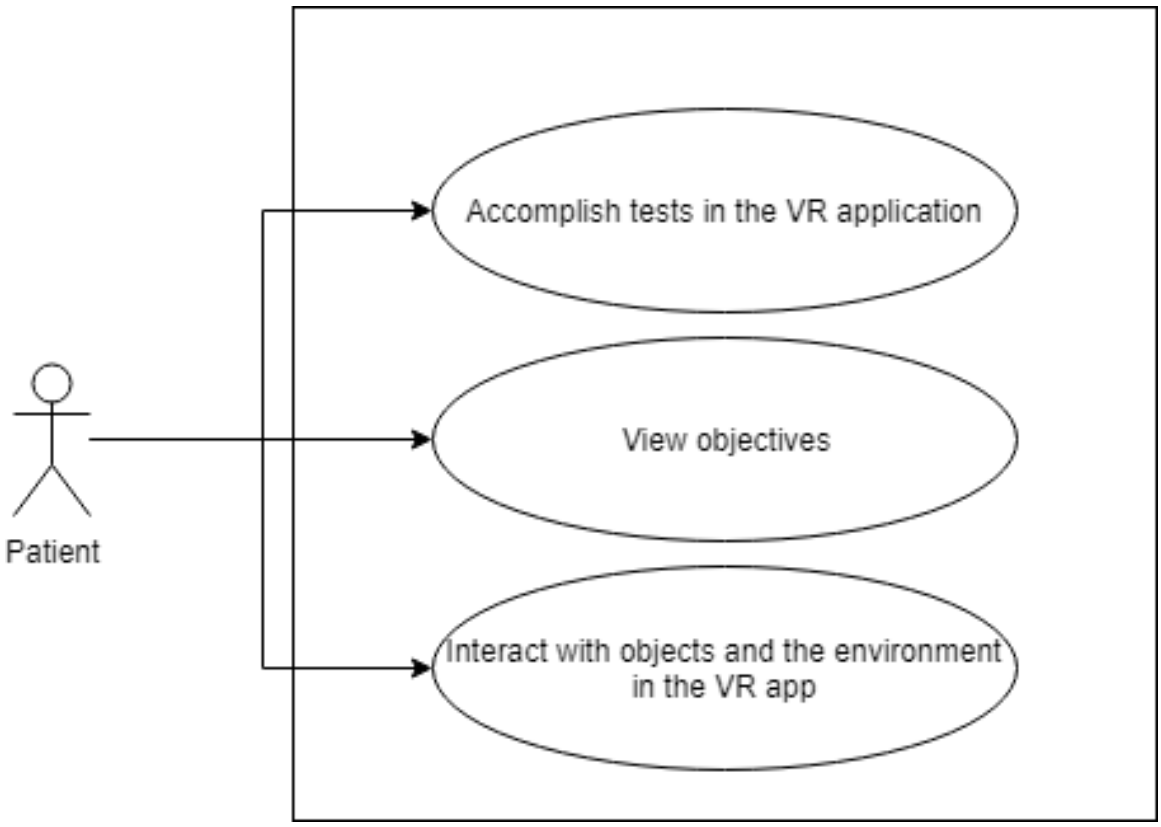

Figure 1: Patient Use Case Diagram

IV. Design and Implementation

A. Use Case Diagram

The application will have two users, the patient and the therapist.

The patient will be able to interact with the objects in the virtual environment. The patient will be lead through a series of mini-games that will assess their cognitive status. The game will constantly guide the patient in how to perform the mini-games.

The therapist will be able to view the results of the patients assessment test.

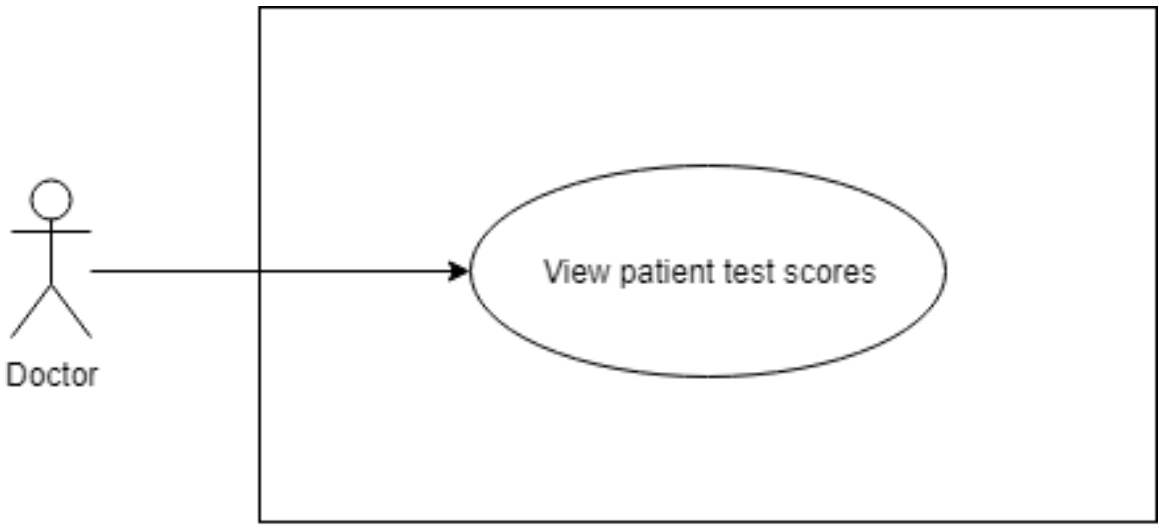

Figure 2: Doctor Use Case Diagram

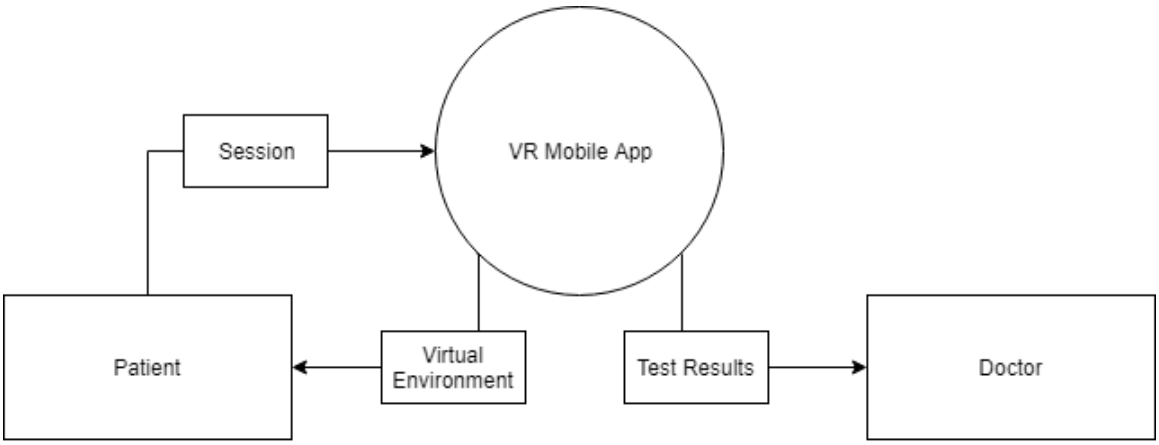

Figure 3: Context Use Case Diagram

B. Context Case Diagram

The patient and the therapist will be the two major entities that interact with the system. The patient starts the application and then performs the assessment test. The patient will then be lead to perform the multiple assessment tests within the app. The therapist will be able to view the patient's score once the test is completed.

C. Activity Diagram

C..1 Patient

The patient will be able to navigate the VR environment and perform gamified assessment tests. The game will constantly guide the patient on performing the tests. The application will end once the the assessment test are complete.

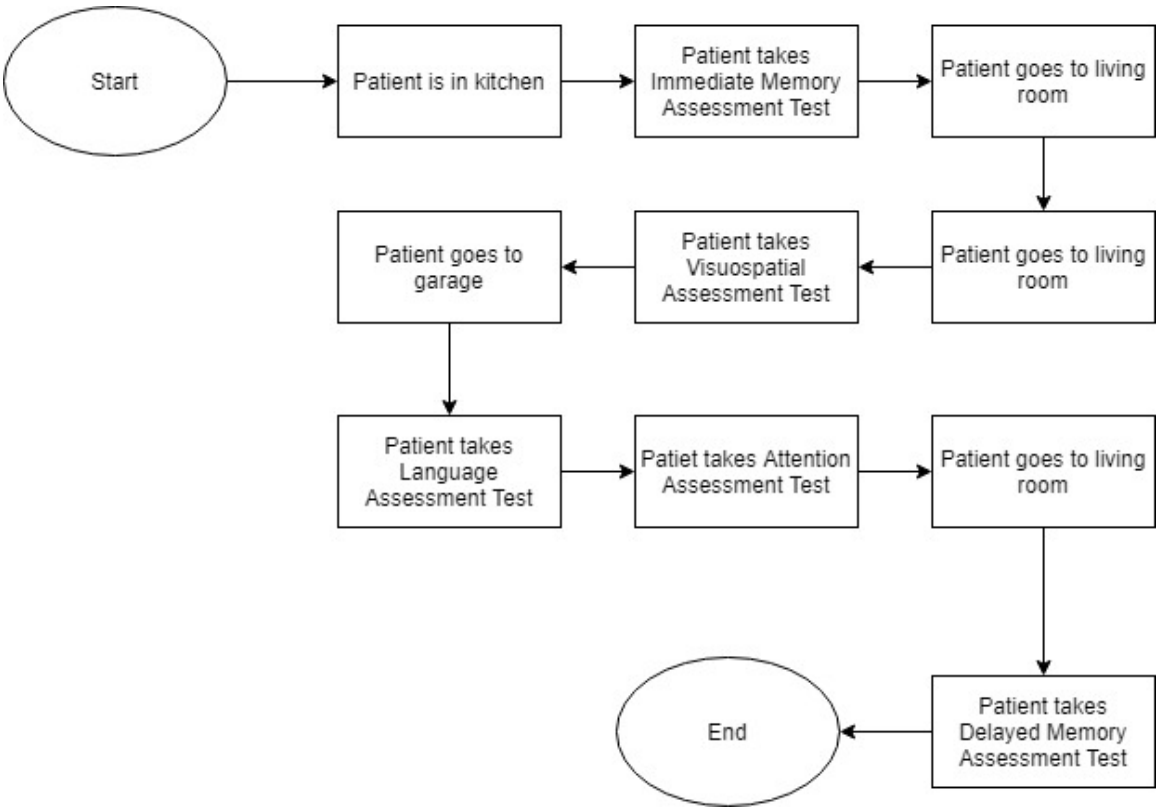

Figure 4: Patient Activity Diagram

C..2 Therapist

The therapist will be able to see the patient's score once the patient completes the assessment tests.

D. Technical Architecture

The following smartphone specifications will be required to run the mobile application:

1. Android OS Version 4.4 or later
2. 2GB of RAM or higher
3. Accelerometer
4. Gyroscope
5. Magnetometer

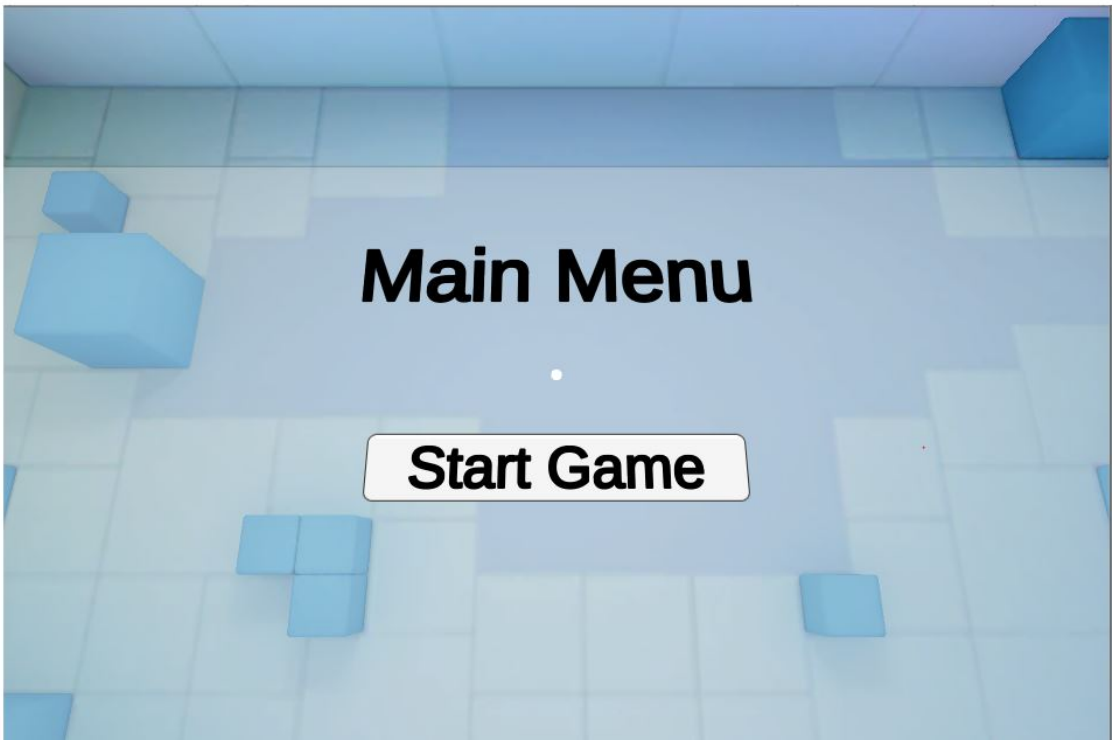

Figure 5: Screenshot of the Main Menu

V. Results

A. Main Menu

The application begins with the main menu which has a start button. On pressing the start button the next level will load.

B. Virtual Environment

The virtual environment is a generic living room with a tv, couch, and coffee table. The first test of the screening test. The "Naming" test in which the patient is asked to identify pictures. The "Dot" test in which the patient is asked to count the number of dots an image. The "Letter Recognition" test in which the patient is asked to identify a misoriented image.

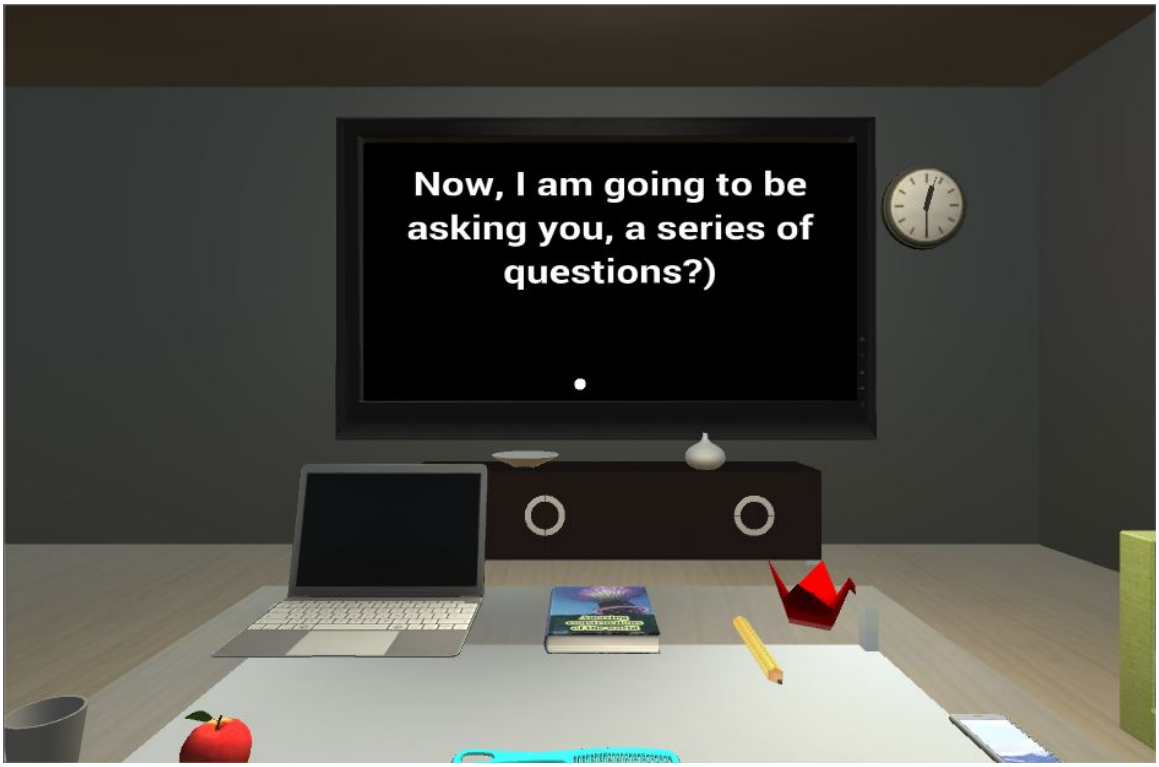

Figure 6: Screenshot of the orientation test

Figure 7: Screenshot of the virtual object memorization test

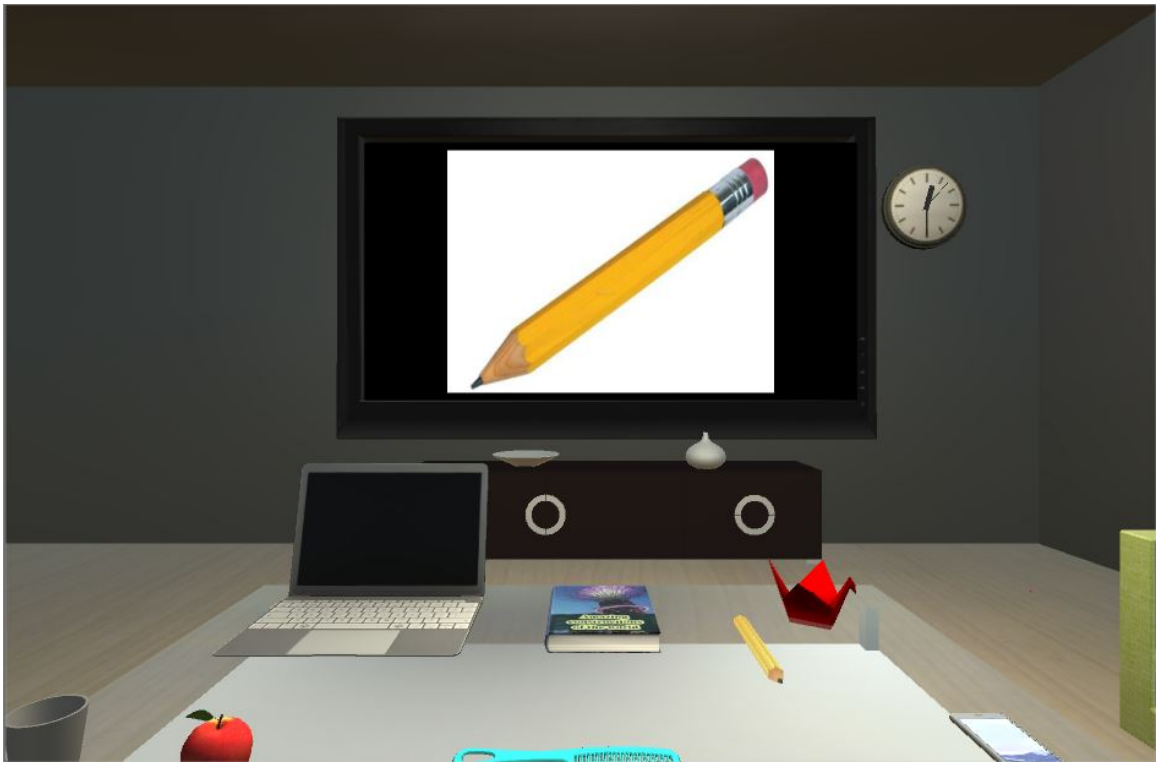

Figure 8: Screenshot of the Naming Test

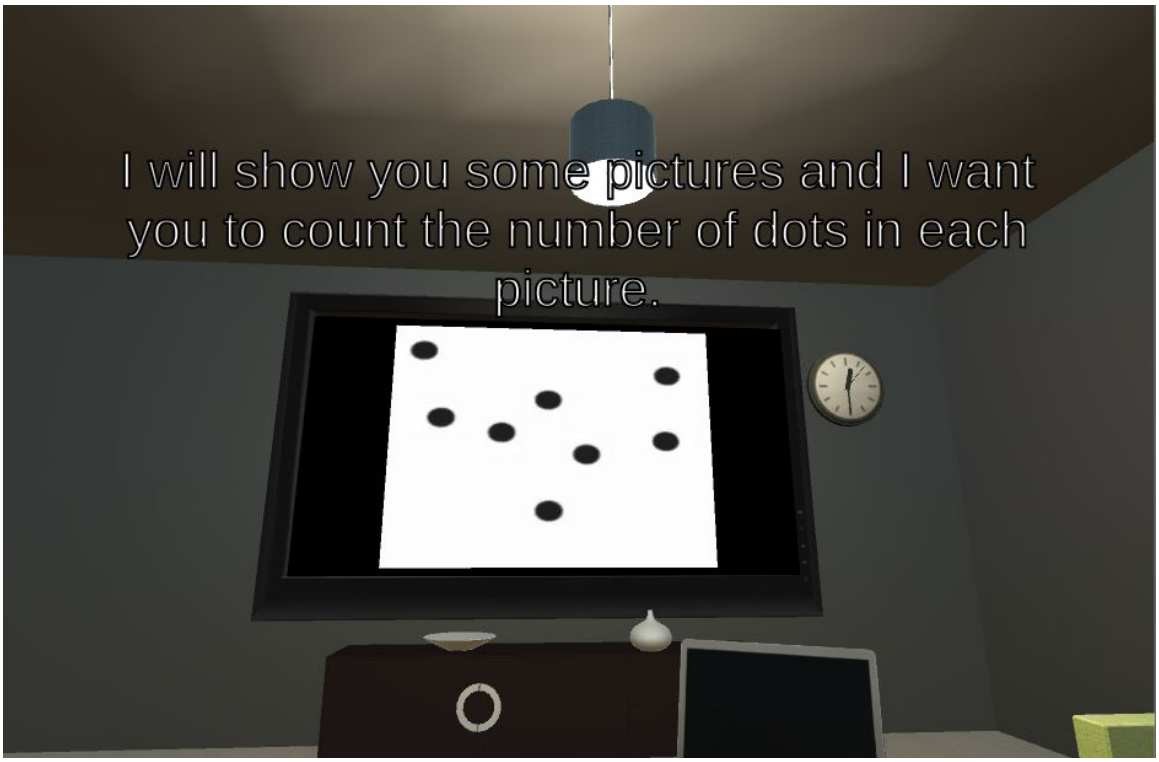

Figure 9: Screenshot of the Dot Test

Figure 10: Screenshot of the Letter Recognition test

VI. Discussions

ACE-VR is a program designed to be supplementary tool for psychologists in assessing patient's who may have dementia. It uses an Android smartphone with an accelerometer, magnetometer, and a gyroscope to run its virtual environments. The patient must be wearing a VR headset to maximize immersion.

The Unity game engine, Blender, and Google API was used in the development of this application. Unity was used to put together the virtual environments and give them life. In Unity, you can use either C# or javascript in order to make the game work. Unity provides a huge array of functionalities in order to streamline development. Blender is a open source 3D modeling software in which the models used in this application were made. Google API for Android allows developers to make use of system services for their application. In this instance, SpeechRecognizer is used to allow accurate Speech-To-Text services.

This application is built with the following functionalities: a main menu, a series of tests and the simulation of those tests, a speech-to-text recognizer for input, and a progress system that both patient and therapist can view. The first functionality simply provides a room to initiate the test. The second functionality is the tests that the patient will be answering. The third functionality allows the patient perform the necessary tasks by recognizing what he/she is saying. The fourth functionality shows the score of the patient after the test.

VII. Conclusions

ACE-VR is an application that provides virtual reality screening test to patients who may be suffering from dementia. It offers a virtual reality space in order to conduct the Addenbrooke's Cognitive Examination. It provides a virtual environment in which stimulus can be controlled and ensures that the test is always administered correctly.

ACE-VR was made using the Unity game engine, Cardboard SDK, and Google API. Unity's Asset Store provides a wide array of free models and materials that was used in the design of the environment.

ACE-VR is developed as an adjunct tool for the screening of dementia in patients,

VIII. Recommendations

ACE-VR is an application that provides virtual reality screening test to patients who may be suffering from dementia. The therapist is be able to ensure that the amount of stimulus in the testing environment is unchanging. It can be used a supplementary tool in testing for the detection of dementia. However, this system is can still be further improved.

Many of the test still draw heavily from the original test, ACE-R. Further modification in order to test memory in a more life like manner would further enhance the use of VR in creating tests that cannot be done using traditional ways. It would also be able to reflect the everyday memory skills of the patient rather than their testing skills.

Another possible improvement would be to create other means of input. Right now the application is heavily reliant on speech input. This may fall short when using complicated words or should the patient have a heavy accent that the speech-to-text model is not accustomed to. Using more convential ways such as gaze input or a controller will make the application more flexible and user-friendly.

The inclusion of more virtual environments in which to conduct the test may also help cater to the users. Having more virtual environments in which the patient may feel more relaxed will help users feel more comfortable in the virtual environments.

IX. Bibliography

[1] "About dementia." http://dementia.stir.ac.uk/information/about-dementia/. Accessed: 2018-04-30.

[2] "Dementia: Fact sheet." http://www.who.int/en/news-room/fact-sheets/detail/dementia. Accessed: 2018-03-09.

[3] N. B. Ogena, "Population aging and future dementia and alzheimer's disease sufferers in the philippines," *Philippine Journal of Geriatrics and Gerontology*, vol. 1, 2015.

[4] A. P. AssociationA, *Diagnostic and statistical manual of mental disorders (5th ed.)*. 2013.

[5] "Dignostic criteria for dementia." https://www.dementia.org.au/information/for-health-professionals/clinical-resources/diagnostic-criteria-for-dementia. Accessed: 2018-04-30.

[6] "Diagnosing dementia." https://www.nia.nih.gov/health/diagnosing-dementia. Accessed: 2018-04-30.

[7] J. G. HJ Woodford, "Cognitive assessment in the elderly: a review of clinical methods.," 2013.

[8] C. et al, "A review of screening tests for cognitive impairment," *Journal of Neurology, Neurosurgery, & Psychiatry*, pp. 790–799, 2007.

[9] A. B. DM de Melo, "Use of the mini-mental state examination in research on the elderly in brazil: a systematic review," *Cien Saude Colet*, 2015.

[10] E. Inc, *Handbook of Assessment in Clinical Gerontology*. Elsevier Inc., 2010.

[11] Z. et al, "Throwing down the gauntlet: can we do better than the mmse?," *Aging & Mental Health*, 2008.

[12] "Addenbrookes cognitive examination revised (ace-r)." https://mh4ot.com/2012/05/17/addenbrookes-cognitive-examination-revised-ace-r/. Accessed: 2018-05-24.

[13] L. et al, "Does the addenbrooke's cognitive examination-revised add to the mini-mental state examination in established alzheimer disease? results from a national dementia research register," *International Journal of Geriatric Psychiatry*, 2013.

[14] V. et al, "Review of brief cognitive tests for patients with suspected dementia," 2014.

[15] "Benefits of computerized cognitive assessment." http://www.cambridgecognition.com/blog/entry/benefits-of-computerized-cognitive-assessment. Accessed: 2018-05-24.

[16] Zygouris and Tsolaki, "Computerized cognitive testing for older adults: a review," *American Journal of Alzheimer's Disease & Other Dementias*, vol. 3, 2010.

[17] R. et al, "Development of a self-administered web-based test for longitudinal cognitive assessment," *Scientific Reports*, vol. 6, 2016.

[18] "7 ways vr is improving healthcare." https://uploadvr.com/healthcare-vr-improve/. Accessed: 2018-05-25.

[19] "What is virtual reality?." https://whatis.techtarget.com/definition/virtual-reality. Accessed: 2018-03-10.

[20] G. Riva, "Virtual reality as assessment tool in psychology," 1997.

[21] "Rendever." http://news.mit.edu/2017/virtual-reality-elderly-sloan-health-care- Accessed: 2018-05-24.

[22] "Rendever." https://rendever.com/. Accessed: 2018-05-24.

[23] A. R. T.D. Parsons, "Initial validation of a virtual environment for assessment of memory functioning: Virtual reality cognitive performance assessment test," *Cyberpsychology & Behavior*, vol. 11, 2008.

[24] G.-B. et al, "A succinct overview of virtual reality technology use in alzheimers disease," *Frontiers in Aging Neuroscience*, vol. 7, 2015.

[25] V. A. J.M.F. Montenegro, "Cognitive evaluation for the diagnosis of alzheimer's disease based on turing test and virtual environments," *Physiology & Behavior*, 2017.

[26] Yeh, "An innovative virtual reality system for mild cognitive impairment: Diagnosis and evaluation," *2012 IEEE EMBS Conference on Biomedical Engineering and Sciences*, 2012.

[27] O. B. N.B.M. Quental, S.M.D. Brucki, "Visuospatial function in early alzheimers diseasethe use of the visual object and space perception (vosp) battery," *PLoS ONE*, 2013.

[28] P. et al, "A virtual reality test for the assessment of cognitive deficits: Usability and perspectives," *Pervasive Computing Technologies for Healthcare*, 2013.

[29] T. et al, "Ecological validity of virtual reality daily living activities screening for early dementia: longitudinal study," *Journal of Medical Internet Research*, 2013.

[30] T. et al, "Can a novel computerized cognitive screening test provide additional information for early detection of alzheimer's disease?," *Alzheimer's & Dementia*, 2014.

[31] C. et al, "Mobile framework for cognitive assessment: Trail making test and reaction time test," *Pervasive Computing Technologies for Healthcare*, 2014.

[32] L. et al, "Using a virtual environment to assess cognition in the elderly," *Virtual Reality*, 2014.

[33] S. et al, "Cognitive examination for the early diagnosis of alzheimer's disease," *Computer and Information Technology*, 2017.

[34] S. . Ang, "Developing virtual environments for older users: Case studies of virtual environments iteratively developed for older users and people with dementia," *2017 2nd International Conference on Information Technology*, 2017.

[35] J. F. Quinn, *Dementia*. West Sussex, UK: John Wiley & Sons, Ltd, 2014.

[36] "Deaths due to dementia." https://www.dementiastatistics.org/statistics/deaths-due-to-dementia/. Accessed: 2018-04-30.

[37] "Addenbrookes cognitive examination-revised (ace-r)." https://www.ipa-online.org/news-and-issues/addenbrookes-cognitive-examination-revised-ace-r. Accessed: 2018-05-24.

[38] "What is virtual reality?." https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html. Accessed: 2018-03-10.

[39] "Virtual reality." http://whatis.techtarget.com/definition/virtual-reality. Accessed: 2018-03-10.

[40] "Head-mounted displays (hmds)." https://www.vrs.org.uk/virtual-reality-gear/head-mounted-displays/. Accessed: 2018-03-10.

[41] "Google cardboard is vr's gateway drug." https://www.wired.com/2015/05/try-google-cardboard/. Accessed: 2018-03-10.

[42] "Number of available applications in the google play store from december 2009 to december 2017." https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. Accessed: 2018-03-10.

X. Appendix

A. Source Code

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

public class Anterograde : Test {

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int numOfTrials;
    private int currentTrial;


    public Anterograde(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }


    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);

        numOfTrials = 3;
        currentTrial = 0;
        trialScore = 0;
        csv = LoadCSV("items");
        UI = (new GameObject("UIController")).AddComponent<UIController>();
        //UI.SetUIReferences();
        answers = new string[7];
        answers[0] = "harry";
        answers[1] = "barnes";
        answers[2] = "73";
        answers[3] = "orchard";
        answers[4] = "close";
        answers[5] = "kingsbridge";
        answers[6] = "devon";

        answer = null;
        RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }

    public override void StartTest()
    {
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }


    private void RemoveAnswers()
    {
        UI.SetGameObjectText("Answer1Text", " ");
        UI.SetGameObjectText("Answer2Text", " ");
        UI.SetGameObjectText("Answer3Text", " ");
    }

    IEnumerator StartTestCoroutine()
    {
        UI.SetGameObjectText("InstructionText", "I'm going to give you a name " +
            "and address and I'd like you to repeat after me? We'll do it thrice. I'll be asking later");
        yield return null;
        StartCoroutine(TestCoroutine());

    }

    IEnumerator TestCoroutine() {
        yield return new WaitForSeconds(6);
        UI.SetGameObjectText("InstructionText", "Harry Barnes, 73 Orchard Close, Kingsbridge, Devon");
        yield return new WaitForSeconds(6);
```

```
    UI.SetGameObjectText("InstructionText", "What name and address did I ask you to memorize? (answer after the
    RemoveAnswers();
    StartCoroutine(Listen());
}

IEnumerator Listen()
{
    yield return new WaitForSeconds(3);
    GetSpeech();
}

public override void EndTest()
{
    UI.SetGameObjectText("InstructionText", "End Test");
    speechManager.Release();
    Debug.Log("End Test");
    gameManager.AddToAssessmentScore(testScore);
    gameManager.NextTest();
}

public void GetSpeech()
{
    speechManager.StartListening(1, "en-US");
}


void OnSpeechResults(string results)
{
    UI.SetGameObjectText("InstructionText", results);
    string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit
    if (texts[0] == "No speech input.")
    {
        UI.SetGameObjectText("PlayerText", "Got em");
    }

    if (answer == null)
    {
        if (texts.Length > 0)
        {
            answer = ParseSpeech(texts);

            ConfirmAnswer(answer);
        }
        else
        {
            //TODO try again
            UI.SetGameObjectText("InstructionText", "I did't quite get that try again (answer after the beep)")
            StartCoroutine(Listen());
        }
    }
    else
    {
        ParseSpeech(texts);
        if (texts[0].Contains("yes"))
        {
            currentTrial++;
            CheckAnswer(answer);
        }
        else
        {
            answer = null;
            UI.SetGameObjectText("InstructionText", "Let's try again (answer after the beep)");
            StartCoroutine(Listen());

        }
    }

}

void OnSpeechError(string error)
{
            if (answer == null) {
                    UI.SetGameObjectText("InstructionText", "What name and address did I ask you to memorize? (
            } else {
                    UI.SetGameObjectText("InstructionText", "Please say either YES or NO (answer after the beep
            }
            StartCoroutine(Listen());
}

public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text.ToLower();
}


public string ParseSpeech(string[] detectedWords)
{
```

```
        foreach (string temp in detectedWords)
        {
            return RemovePunctuations(temp);
        }
        return null;
    }

    public void ConfirmAnswer(string answer)
    {
        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer? (Say Yes/No after th
        StartCoroutine(Listen());
    }


    public void CheckAnswer(string answer)
    {
        string[] words = answer.Split(' ');
        foreach (string word in words)
        {
            foreach (string ans in answers)
            {
                if (word == ans && currentTrial == 1)
                {
                    testScore++;
                    UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());
                }
            }
        }
                answer = null;
        if (currentTrial == numOfTrials)
        {
            EndTest();
        }
        else
        {
                        if (testScore == 3) {
                                UI.SetGameObjectText ("InstructionText", "That was great! Let's try that again.");
                        } else {
                                UI.SetGameObjectText ("InstructionText", "That wasn't quite right. Let's try that a
                        }
            StartCoroutine(TestCoroutine());
        }

    }

    public override void AddChoiceScriptToGameObject() { }

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion() { }

    public override void NextQuestion() { }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AnterogradeRecall : Test {

    public GameManager gameManager;
    public int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int numOfTrials;
    private int currentTrial;


    public AnterogradeRecall(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }


    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);

        numOfTrials = 3;
        currentTrial = 0;
        trialScore = 0;
        csv = LoadCSV("items");
        UI = (new GameObject("UIController")).AddComponent<UIController>();
        //UI.SetUIReferences();
```

```
        answers = new string[7];
        answers[0] = "harry";
        answers[1] = "barnes";
        answers[2] = "73";
        answers[3] = "orchard";
        answers[4] = "close";
        answers[5] = "kingsbridge";
        answers[6] = "devon";
        answer = null;
        RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }

    public override void StartTest()
    {
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }


    private void RemoveAnswers()
    {
        UI.SetGameObjectText("Answer1Text", " ");
        UI.SetGameObjectText("Answer2Text", " ");
        UI.SetGameObjectText("Answer3Text", " ");
    }

    IEnumerator StartTestCoroutine()
    {
        UI.SetGameObjectText("InstructionText", "Tell me what you remember of the name and address we were repeatin
        yield return null;
        StartCoroutine(TestCoroutine());

    }

    IEnumerator TestCoroutine()
    {

        yield return new WaitForSeconds(6);
        StartCoroutine(Listen());
    }

    IEnumerator Listen()
    {
        yield return new WaitForSeconds(3);
        GetSpeech();
    }

    public override void EndTest()
    {
        speechManager.Release();
        Debug.Log("End Test");
        gameManager.AddToAssessmentScore(testScore);
        gameManager.NextTest();
    }

    public void GetSpeech()
    {
        speechManager.StartListening(1, "en-US");
    }

    public void EndSpeech()
    {
        speechManager.CancelListening();
    }

    void OnSpeechResults(string results)
    {

        // Need to parse
        string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit
        UI.SetGameObjectText("Answer3Text", texts[0]);

        if (answer == null)
        {
            if (texts.Length > 0)
            {
                answer = ParseSpeech(texts);

                ConfirmAnswer(answer);
            }
            else
            {
                //TODO try again
                UI.SetGameObjectText("InstructionText", "I did't quite get that try again (answer after the beep)")
                StartCoroutine(Listen());
            }
        }
        else
        {
            UI.SetGameObjectText("Answer2Text", "Here");
```

```
            ParseSpeech(texts);
            if (texts[0].Contains("yes"))
            {
                UI.SetGameObjectText("Answer2Text", "Here na");
                currentTrial++;
                CheckAnswer(answer);
            }
            else
            {
                answer = null;
                UI.SetGameObjectText("InstructionText", "Let's try again (answer after the beep)");
                StartCoroutine(Listen());

            }
        }

    }



    public string RemovePunctuations(string text)
    {
        var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
        foreach (var c in charsToRemove)
        {
            text = text.Replace(c, string.Empty);
        }
        return text.ToLower();
    }


    public string ParseSpeech(string[] detectedWords)
    {
        foreach (string temp in detectedWords)
        {
            return RemovePunctuations(temp);
        }
        return null;
    }

    public void ConfirmAnswer(string answer)
    {
        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer? (Say Yes/No after th
        StartCoroutine(Listen());

    }


    public void CheckAnswer(string answer)
    {
        string[] words = answer.Split(' ');
        foreach (string word in words)
        {
            foreach (string ans in answers)
            {
                if (word == ans)
                {
                    testScore++;
                    UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());
                }
            }
        }

        EndTest();
    }

    public override void AddChoiceScriptToGameObject() { }

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion() { }

    public override void NextQuestion() { }
}


using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AnterogradeRecognition : Test {

    public GameManager gameManager;
        public int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
```

```
private string[] answers;
private string answer,reminder;

private int numOfTrials;
private int currentTrial;
private int questionCounter;

Question[] questions;
Question currentQuestion;

public AnterogradeRecognition(GameManager gg, Question[] questions) : base(gg)
{
    Initialize();
}

public void SetNumOfTrials(int x)
{
    numOfTrials = x;
    currentTrial = 0;
}

public override void Initialize()
{
    speechManager = new SpeechRecognizerManager(gameObject.name);

    reminder = " (speak after the beep)";

    //csv = LoadCSV("items");
    UI = (new GameObject("UIController")).AddComponent<UIController>();
    //UI.SetUIReferences();

    answer = null;
    GenerateQuestions();

    UI.SetGameObjectText("PlayerText", " ");
    StartTest();
}

private void GenerateQuestions()
{
    answers = new string[5];
    answers[0] = "harry barnes";
    answers[1] = "73";
    answers[2] = "orchard close";
    answers[3] = "kingsbridge";
    answers[4] = "devon";
    questions = new Question[5];
    questions[0] = new Question("Jerry Barnes", "Harry Barnes", "Jerry Bradford",0);
    questions[1] = new Question("37", "73", "75",0);
    questions[2] = new Question("Orchard Place", "Oak Close", "Orchard Close",0);
    questions[3] = new Question("Oakhampton", "Kingsbridge", "Dartington",0);
    questions[4] = new Question("Devon", "Dorset", "Somerset",0);
}

public override void StartTest()
{
    SetQuestion();
}

private void SetAnswers()
{
    UI.SetGameObjectText("Answer1Text", currentQuestion.choice1);
    UI.SetGameObjectText("Answer2Text", currentQuestion.choice2);
    UI.SetGameObjectText("Answer3Text", currentQuestion.choice3);
}
private void RemoveAnswers()
{
    UI.SetGameObjectText("Answer1Text", " ");
    UI.SetGameObjectText("Answer2Text", " ");
    UI.SetGameObjectText("Answer3Text", " ");
}



IEnumerator Listen()
{
    yield return new WaitForSeconds(3);
    GetSpeech();
}

public override void EndTest()
{
    Debug.Log("End Test");
    speechManager.Release();
    gameManager.AddToAssessmentScore(testScore);
    gameManager.NextTest();
}

public void GetSpeech()
{
    speechManager.StartListening(1, "en-US");
}
```

```
void OnSpeechError(string error)
{
    // Speech Recognition TimeOut or Speech Recognition Error
    UI.SetGameObjectText("InstructionText", "I did't quite get that. Please try again (answer after the beep)")
    StartCoroutine(Listen());
}


void OnSpeechResults(string results)
{

    // Need to parse
    string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit
    UI.SetGameObjectText("Answer3Text", texts[0]);

    if (answer == null)
    {
        answer = ParseSpeech(texts);
        ConfirmAnswer(answer);
    }
    else
    {
        if (texts[0].Contains("yes"))
        {
            currentTrial++;
            CheckAnswer(answer);
        }
        else
        {
            answer = null;
            UI.SetGameObjectText("InstructionText", "Let's try again (answer after the beep)");
            StartCoroutine(Listen());
        }
    }

}



public void CheckAnswer(string answer)
{
    string[] words = answer.Split(' ');

    if (answer == answers[questionCounter])
    {
        testScore++;
        UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());

    }

    if (questionCounter == 3)
    {
        EndTest();
    }
    else
    {
        NextQuestion();
    }

}

public override void AddChoiceScriptToGameObject() { }

public override void IsCorrectAnswer(int choice) { }

public override void SetQuestion() {
    currentQuestion = questions[questionCounter];
    UI.SetGameObjectText("InstructionText", "I'll give you some hints: was it" + reminder);
    SetAnswers();
    StartCoroutine(Listen());
}

public override void NextQuestion() {
    answer = null;
    questionCounter++;
    SetQuestion();
}


public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text;
}
```

```
    public string ParseSpeech(string[] detectedWords)
    {
        foreach (string temp in detectedWords)
        {
            return RemovePunctuations(temp);
        }
        return null;
    }

    public void ConfirmAnswer(string answer)
    {
        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer? (Say Yes/No after th
        StartCoroutine(Listen());

    }

}

using UnityEngine;
using UnityEngine.UI;

public class Choice : MonoBehaviour {

    // Use this for initialization
    private int choiceValue;
    private Test test;

    public Choice(Test test, int x)
    {
        this.test = test;
        this.choiceValue = x;
    }

    public void SendAnswer()
    {
        test.IsCorrectAnswer(choiceValue);

    }


    public void SendAnswerAndDisableSelf()
    {
        test.IsCorrectAnswer(choiceValue);
        Button button = this.gameObject.GetComponentInParent<Button>();
        button.interactable = false;
    }



}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Comprehension : Test {

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int numOfTrials;
    private int currentTrial;
    private int questionCounter;

    private Question[] questions;
    private Question currentQuestion;

    private string reminder;

    public Comprehension(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }

    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);
        reminder = " (answer after the beep)";
        testScore = 0;
        questionCounter = 0;
        UI = (new GameObject("UIController")).AddComponent<UIController>();

        answers = new string[5];
```

```
        GenerateQuestion();
        answer = null;
        RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        ShowImageOptions();
        StartCoroutine(StartTestCoroutine());
    }


    private void RemoveAnswers()
    {
        UI.SetGameObjectText("Answer1Text", " ");
        UI.SetGameObjectText("Answer2Text", " ");
        UI.SetGameObjectText("Answer3Text", " ");
    }

    IEnumerator StartTestCoroutine()
    {
        yield return null;

        SetQuestion();

    }

    IEnumerator Listen()
    {
        yield return new WaitForSeconds(3);
        GetSpeech();

    }

    public override void EndTest()
    {
        HideImageOptions();
        speechManager.Release();
        Debug.Log("End Test");
        gameManager.AddToAssessmentScore(testScore);
        gameManager.NextTest();
    }

    public void CheckAnswer(string answer)
    {
        if (answers[questionCounter] == answer)
        {
            testScore++;
            UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());
        }
        if (questionCounter == 3)
        {
            EndTest();
        }
        else
        {
            NextQuestion();
        }
    }

    private void GenerateQuestion()
    {

        questions = new Question[4];
        questions[0] = new Question("Which is assoiciated with the monarchy?", "Crown", "Watch", "Anchor",0);
        answers[0] = "crown";
        //Date
        questions[1] = new Question("Which one is man's best friend?", "Eagle", "Cat", "Dog",0);
        answers[1] = "dog";
        //Month
        questions[2] = new Question("Which one is found in the Antarctic?", "Penguin", "Elephant", "Eagle", 0);
        answers[2] = "penguin";
        //Year
        questions[3] = new Question("Which one has a nautical connection?", "Barrel", "Crown", "Anchor", 0);
        answers[3] = "anchor";

    }

    public override void AddChoiceScriptToGameObject() { }

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion()
    {
        currentQuestion = questions[questionCounter];
        UI.SetGameObjectText("InstructionText", currentQuestion.question + reminder);
        SetImageOptions();
        StartCoroutine(Listen());
    }

    public override void NextQuestion()
    {
        answer = null;
        questionCounter++;
        SetQuestion();
```

```
    }

    private void ShowImageOptions()
    {
        UI.ActivateRawImage("Answer1Image");
        UI.ActivateRawImage("Answer2Image");
        UI.ActivateRawImage("Answer3Image");
    }
    private void HideImageOptions()
    {
        UI.DeactivateRawImage("Answer1Image");
        UI.DeactivateRawImage("Answer2Image");
        UI.DeactivateRawImage("Answer3Image");
    }

    private void SetImageOptions()
    {
        UI.SetRawImageTexture("Answer1Image", currentQuestion.choice1);
        UI.SetRawImageTexture("Answer2Image", currentQuestion.choice2);
        UI.SetRawImageTexture("Answer3Image", currentQuestion.choice3);
    }


    public void GetSpeech()
    {
        speechManager.StartListening(1, "en-US");
    }

    void OnSpeechError(string error)
    {
                if (answer == null) {
                        UI.SetGameObjectText("InstructionText", questions[questionCounter] +reminder);
                } else {
                        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer?
\n (Say Yes/No after the beep)");
                }
                StartCoroutine(Listen());
    }

    void OnSpeechResults(string results)
    {
        // Need to parse
        string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit

        if (answer == null)
        {
            answer = ParseSpeech(texts);
            ConfirmAnswer(answer);
        }
        else
        {
            ParseSpeech(texts);
            if (texts[0].Contains("yes"))
            {
                currentTrial++;
                CheckAnswer(answer);
            }
            else
            {
                answer = null;
                            UI.SetGameObjectText("InstructionText", questions[questionCounter] +reminder);
                StartCoroutine(Listen());
            }
        }

    }

    public string RemovePunctuations(string text)
    {
        var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
        foreach (var c in charsToRemove)
        {
            text = text.Replace(c, string.Empty);
        }
        return text;
    }

    public string ParseSpeech(string[] detectedWords)
    {
        foreach (string temp in detectedWords)
        {
            return RemovePunctuations(temp);
        }
        return null;
    }

    public void ConfirmAnswer(string answer)
    {
        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer?  \n (Say Yes/No afte
        StartCoroutine(Listen());

    }
```

```
    public override void StartTest() { }
}

using System.Collections;
using UnityEngine;
using System.Linq;
public class CSVReader : MonoBehaviour {

    public TextAsset csvFile;
    private string[,] csv;
    // Use this for initialization

    public CSVReader(TextAsset file) {
        csvFile = file;
        Start();
    }



        void Start () {
        csv = SplitCsvGrid(csvFile.text);
    }

    public string[,] getCSV() {
        return csv;
    }



    // splits a CSV file into a 2D string array
    static public string[,] SplitCsvGrid(string csvText)
    {
        string[] lines = csvText.Split("\n"[0]);

        // finds the max width of row
        int width = 0;
        for (int i = 0; i < lines.Length; i++)
        {
            string[] row = SplitCsvLine(lines[i]);
            width = Mathf.Max(width, row.Length);
        }

        // creates new 2D string grid to output to
        string[,] outputGrid = new string[width + 1, lines.Length + 1];
        for (int y = 0; y < lines.Length; y++)
        {
            string[] row = SplitCsvLine(lines[y]);
            for (int x = 0; x < row.Length; x++)
            {
                outputGrid[x, y] = row[x];

                // This line was to replace "" with " in my output.
                // Include or edit it as you wish.
                outputGrid[x, y] = outputGrid[x, y].Replace("\"\"", "\"");
            }
        }

        return outputGrid;
    }

    // splits a CSV row
    static public string[] SplitCsvLine(string line)
    {
        return (from System.Text.RegularExpressions.Match m in System.Text.RegularExpressions.Regex.Matches(line,
        @"(((?<x>(?=[,\r\n]+))|""(?<x>([^""]|"""")+)""|(?<x>[^,\r\n]+)),?)",
        System.Text.RegularExpressions.RegexOptions.ExplicitCapture)
                select m.Groups[1].Value).ToArray();
    }



}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DotCount : Test {

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;
```

```
private int numOfTrials;
private int currentTrial;
private int questionCounter;

private string[] questions;
private string reminder;

public DotCount(GameManager gg, Question[] questions) : base(gg)
{
    Initialize();
}

public override void Initialize()
{
    speechManager = new SpeechRecognizerManager(gameObject.name);
    reminder = "(answer after the beep)";
    testScore = 0;
    questionCounter = 0;
    UI = (new GameObject("UIController")).AddComponent<UIController>();

    GenerateQuestion();
    answer = null;
    RemoveAnswers();
    UI.SetGameObjectText("PlayerText", " ");
    StartCoroutine(StartTestCoroutine());
}


private void RemoveAnswers()
{
    UI.SetGameObjectText("Answer1Text", " ");
    UI.SetGameObjectText("Answer2Text", " ");
    UI.SetGameObjectText("Answer3Text", " ");
}

IEnumerator StartTestCoroutine()
{
    UI.SetGameObjectText("PlayerText", "I will show you some pictures and I want you to count the number of dot

    yield return new WaitForSeconds(5);

    SetQuestion();

}

IEnumerator Listen()
{
    yield return new WaitForSeconds(5);
            UI.SetGameObjectText("PlayerText", reminder);

    GetSpeech();

}

public override void EndTest()
{
    speechManager.Release();
    UI.DeactivateRawImage("NamingImage");

    Debug.Log("End Test");
    gameManager.AddToAssessmentScore(testScore);
    gameManager.NextTest();
}

public void GetSpeech()
{
    speechManager.StartListening(1, "en-US");
}

void OnSpeechError(string error)
{
    // Speech Recognition TimeOut or Speech Recognition Error
    UI.SetGameObjectText("PlayerText", "I did't quite get that. Please try again (answer after the beep)");

    StartCoroutine(Listen());
}

void OnSpeechResults(string results)
{
    // Need to parse
    string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit

    if (answer == null)
    {
        answer = ParseSpeech(texts);
        ConfirmAnswer(answer);
    }
    else
    {
        ParseSpeech(texts);
        if (texts[0].Contains("yes"))
        {
```

```
            currentTrial++;
            CheckAnswer(answer);
        }
        else
        {
            answer = null;
            StartCoroutine(Listen());

        }
    }

}



public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text;
}


public string ParseSpeech(string[] detectedWords)
{
    foreach (string temp in detectedWords)
    {
        return RemovePunctuations(temp);
    }
    return null;
}

public void ConfirmAnswer(string answer)
{
    UI.SetGameObjectText("PlayerText", "You said: " + answer + " Is this your answer?  \n (Say Yes/No after the
    StartCoroutine(Listen());

}


public void CheckAnswer(string answer)
{
    if (answers[questionCounter].Contains(answer) || answers[questionCounter] == answer)
    {
        testScore++;
        UI.SetGameObjectText("PlayerText", "Score: " + testScore.ToString());
    }
    if (questionCounter == 3)
    {
        EndTest();
    }
    else
    {
        NextQuestion();
    }
}

private void GenerateQuestion()
{
    answers = new string[4];
    questions = new string[4];

    questions[0] = "Percept1";
    answers[0] = "8";

    questions[1] = "Percept2";
    answers[1] = "10";

    questions[2] = "Percept3";
    answers[2] = "7";

    questions[3] = "Percept4";
    answers[3] = "9";



}

public override void AddChoiceScriptToGameObject() { }

public override void IsCorrectAnswer(int choice) { }

public override void SetQuestion()
{
    UI.ActivateRawImage("NamingImage");
    UI.SetRawImageTexture("NamingImage", questions[questionCounter]);
    StartCoroutine(Listen());
}
```

```
    public override void NextQuestion()
    {
        answer = null;
        questionCounter++;
        SetQuestion();
    }

    public override void StartTest() { }
}


using UnityEngine;
using UnityEngine.SceneManagement;
using System.Collections;
public class GameManager : MonoBehaviour {

    // Use this for initialization
    private int totalScore;
    private VOM vom;
    private float timer;
    private int sceneIndex;
    public LR listRecall;
    private string[] listRecallAnswers;
    private Question[] vorRecallQuestions;
    private Recall recallTest;
    private OrientationTest orientationTest;
    private Anterograde anterograde;
    private VerbalFluency verbalFluency;
    private MathTest mathTest;
    private RetrogradeRecall retrograde;
    private Reading reading;
    private Repetition repetition;
    private AnterogradeRecall anterogradeRecall;
    private AnterogradeRecognition anterogradeRecognition;
    private Comprehension comprehension;
    private Naming naming;
    private DotCount dotCount;
    private LetterRecog letterRecog;
    private int testNumber = 0;
        private UIController uiController;

        void Start () {
        DontDestroyOnLoad(this);
        totalScore = 0;
 }


        void Awake () {
        Debug.Log(sceneIndex);
        OnLevelWasLoaded();
    }



    public void AddToAssessmentScore(int testScore) {
        this.totalScore += testScore;
    }

    public void SetListRecallAnswers(string[] lrAnswers)
    {
        listRecallAnswers = lrAnswers;
    }

    public void SetVORRecallAnswers(Question[] vorAnswers)
    {
        vorRecallQuestions = vorAnswers;
    }


    public void NextLevelCor() {
        StartCoroutine(NextLevelCoroutine());
    }

    public void NextTest() {
                testNumber++;
                if (testNumber == 1) {
                        listRecall = (new GameObject ("ListRecallController")).AddComponent<LR> ();
                        listRecall.gameManager = this;
                        listRecall.Initialize ();
                } else if (testNumber == 2) {
                        mathTest = (new GameObject ("MathTestController")).AddComponent<MathTest> ();
                        mathTest.gameManager = this;
                        mathTest.Initialize ();
                } else if (testNumber == 3) {
                        recallTest = (new GameObject ("Recall")).AddComponent<Recall> ();
                        recallTest.gameManager = this;
                        recallTest.Initialize ();
                } else if (testNumber == 4) {
                        anterograde = (new GameObject ("AnterogradeTestController")).AddComponent<Anterograde> ();
                        anterograde.gameManager = this;
                        anterograde.Initialize ();
                } else if (testNumber == 5) {
```

```
                retrograde = (new GameObject ("RetrogradeTestController")).AddComponent<RetrogradeRecall> (
                retrograde.gameManager = this;
                retrograde.Initialize ();
        } else if (testNumber == 6) {
                verbalFluency = (new GameObject ("VerbalFluencyTestController")).AddComponent<VerbalFluency
                verbalFluency.gameManager = this;
                verbalFluency.Initialize ();
        } else if (testNumber == 7) { // LANGUAGE - Repetition
                repetition = (new GameObject ("RepetionTestController")).AddComponent<Repetition> ();
                repetition.gameManager = this;
                repetition.Initialize ();
        } else if (testNumber == 8) { // LANGUAGE - Naming
                naming = (new GameObject ("NamingComprehensionTestController")).AddComponent<Naming> ();
                naming.gameManager = this;
                naming.Initialize ();
        } else if (testNumber == 9) { // LANGUAGE - Comprehension
                comprehension = (new GameObject ("ComprehensionTestController")).AddComponent<Comprehension
                comprehension.gameManager = this;
                comprehension.Initialize ();
        } else if (testNumber == 10) { // LANGUAGE - Reading
                reading = (new GameObject ("ReadingTestController")).AddComponent<Reading> ();
                reading.gameManager = this;
                reading.Initialize ();
        } else if (testNumber == 11) { // VISUOSPATIAL ABILITIES
                vom.StartTest ();
        } else if (testNumber == 12) { // PERCEPTUAL ABILITIES

                dotCount = (new GameObject ("DotTestController")).AddComponent<DotCount> ();
                dotCount.gameManager = this;
                dotCount.Initialize ();
                testNumber++;

        } else if (testNumber == 13) { // PERCEPTUAL ABILITIES
                uiController = (new GameObject ("UIController")).AddComponent<UIController> ();

                letterRecog = (new GameObject ("LetterRecogController")).AddComponent<LetterRecog> ();
                letterRecog.gameManager = this;
                letterRecog.Initialize ();

        } else if (testNumber == 14) { // RECALL - Anterograde
                anterogradeRecall = (new GameObject ("AnterogradeRecallTestController")).AddComponent<Anter
                anterogradeRecall.gameManager = this;
                anterogradeRecall.Initialize ();

        } else if (testNumber == 15) { // RECOGNITION - Anterograde
                anterogradeRecognition = (new GameObject ("AnterogradeRecognitionTestController")).AddCompo
                anterogradeRecognition.gameManager = this;
                if (anterogradeRecall.testScore < 7) {
                        anterogradeRecognition.Initialize ();
                } else {
                        anterogradeRecognition.testScore = 5;
                        AddToAssessmentScore (5);
                }
        } else if (testNumber==16){
                uiController.SetGameObjectText ("InstructionText", "SCORE: " + totalScore.ToString());
        }
    }



public void OnLevelWasLoaded()
{
    Question[] questions = new Question[3];
    questions[0] = new Question("Which word did I ask you to repeat and remember?", "Lemon", "Orange", "Apple",
    questions[1] = new Question("Which word did I ask you to repeat and remember?", "Key", "Wallet", "Phone", 1
    questions[2] = new Question("Which word did I ask you to repeat and remember?", "Doll", "Train", "Ball", 3)
    //Tried and tested
        if(sceneIndex == 1){
                uiController = (new GameObject("UIController")).AddComponent<UIController>();
                orientationTest = (new GameObject("OrientationTestController")).AddComponent<OrientationTes
                orientationTest.gameManager = this;
                orientationTest.Initialize();

                vom = (new GameObject("VOMTestController")).AddComponent<VOM>();
                vom.gameManager = this;
                vom.Initialize();

        }




}

IEnumerator NextLevelCoroutine() {
    //TODO SHOW LOADING SCREEN
    Debug.Log("Start coroutine");
    sceneIndex++;
    yield return new WaitForSeconds(3f);
```

```
        if (sceneIndex < 4)
        {
            SceneManager.LoadSceneAsync(SceneManager.GetActiveScene().buildIndex + 1, LoadSceneMode.Single);
        }
        else if (sceneIndex == 5)
        {
            SceneManager.LoadSceneAsync(2);
        }
    }

}


public class Item  {

    public string prefabName;
    public float pos_x;
    public float pos_y;
    public float pos_z;
    public float rot_x;
    public float rot_y;
    public float rot_z;
    public float scale_x;
    public float scale_y;
    public float scale_z;


    public Item(string prefabName, float pos_x, float pos_y, float pos_z, float rot_x, float rot_y, float rot_z, fl
        this.prefabName = prefabName;
        this.pos_x = pos_x;
        this.pos_y = pos_y;
        this.pos_z = pos_z;
        this.rot_x = rot_x;
        this.rot_y = rot_y;
        this.rot_z = rot_z;
        this.scale_x = scale_x;
        this.scale_y = scale_y;
        this.scale_z = scale_z;
    }




}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LetterRecog : Test {

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int numOfTrials;
    private int currentTrial;
    private int questionCounter;

    private string[] questions;
    private string reminder;

    public LetterRecog(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }

    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);
        reminder = "(answer after the beep)";
        testScore = 0;
        questionCounter = 0;
        UI = (new GameObject("UIController")).AddComponent<UIController>();

        GenerateQuestion();
        answer = null;
        RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }
```

```
private void RemoveAnswers()
{
    UI.SetGameObjectText("Answer1Text", " ");
    UI.SetGameObjectText("Answer2Text", " ");
    UI.SetGameObjectText("Answer3Text", " ");
}

IEnumerator StartTestCoroutine()
{
    UI.SetGameObjectText("InstructionText", "I will show you some pictures and I want you to identify what lett

    yield return new WaitForSeconds(5);
    UI.SetGameObjectText("InstructionText", " ");

    SetQuestion();

}

IEnumerator Listen()
{
    yield return new WaitForSeconds(5);
    GetSpeech();

}

public override void EndTest()
{
    speechManager.Release();
    UI.DeactivateRawImage("NamingImage");

    Debug.Log("End Test");
    gameManager.AddToAssessmentScore(testScore);
    gameManager.NextTest();
}

public void GetSpeech()
{
    speechManager.StartListening(3, "en-US");
}

void OnSpeechError(string error)
{
    // Speech Recognition TimeOut or Speech Recognition Error
    UI.SetGameObjectText("PlayerText", "I did't quite get that. Please try again (answer after the beep)");
    StartCoroutine(Listen());
}

void OnSpeechResults(string results)
{
    // Need to parse
    string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit

    if (answer == null)
    {
        answer = ParseSpeech(texts);
        ConfirmAnswer(answer);
    }
    else
    {
        ParseSpeech(texts);
        if (texts[0].Contains("yes"))
        {
            currentTrial++;
            CheckAnswer(answer);
        }
        else
        {
            answer = null;
            UI.SetGameObjectText("PlayerText", "Let's try again (answer after the beep)");
            StartCoroutine(Listen());

        }
    }

}



public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text;
}


public string ParseSpeech(string[] detectedWords)
{
    foreach (string temp in detectedWords)
```

```
        {
            return RemovePunctuations(temp);
        }
        return null;
    }

    public void ConfirmAnswer(string answer)
    {
        UI.SetGameObjectText("PlayerText", "You said: " + answer + " Is this your answer?  \n (Say Yes/No after the
        StartCoroutine(Listen());

    }


    public void CheckAnswer(string answer)
    {
        string[] possibilities = answer.Split(' ');
        foreach (string possibility in possibilities) {
            if (answers[questionCounter] == answer)
            {
                testScore++;
                UI.SetGameObjectText("PlayerText", "Score: " + testScore.ToString());
                break;
            }
        }
        if (questionCounter == 3)
        {
            EndTest();
        }
        else
        {
            NextQuestion();
        }
    }

    private void GenerateQuestion()
    {
        answers = new string[4];
        questions = new string[4];

        questions[0] = "K";
        answers[0] = "k";

        questions[1] = "M";
        answers[1] = "m";

        questions[2] = "A";
        answers[2] = "a";

        questions[3] = "T";
        answers[3] = "t";

    }

    public override void AddChoiceScriptToGameObject() { }

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion()
    {
        UI.ActivateRawImage("NamingImage");
                UI.SetRawImageTexture("NamingImage", questions[questionCounter]);
        StartCoroutine(Listen());
    }

    public override void NextQuestion()
    {
        answer = null;
        questionCounter++;
        SetQuestion();
    }

    public override void StartTest() { }
}


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

public class LR : Test
{

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;
```

```
private SpeechRecognizerManager speechManager;
private string[] answers;
private string answer;

private int numOfTrials;
private int currentTrial;


public LR(GameManager gg, Question[] questions) : base(gg)
{
    Initialize();
}


public override void Initialize()
{
    speechManager = new SpeechRecognizerManager(gameObject.name);

    numOfTrials = 3;
    trialScore = 0;
    //csv = LoadCSV("items");
    UI = (new GameObject("UIController")).AddComponent<UIController>();
    //UI.SetUIReferences();
    answers = new string[3];
    answers[0] = "lemon";
    answers[1] = "camera";
    answers[2] = "phone";
    answer = null;

    SetAnswers();
    UI.SetGameObjectText("PlayerText", " ");
    StartCoroutine(StartTestCoroutine());
}

public override void StartTest()
{
    UI.SetGameObjectText("PlayerText", " ");
    StartCoroutine(StartTestCoroutine());
}

private void SetAnswers()
{
    UI.SetGameObjectText("Answer1Text", answers[0]);
    UI.SetGameObjectText("Answer2Text", answers[1]);
    UI.SetGameObjectText("Answer3Text", answers[2]);
}
private void RemoveAnswers()
{
    UI.SetGameObjectText("Answer1Text", " ");
    UI.SetGameObjectText("Answer2Text", " ");
    UI.SetGameObjectText("Answer3Text", " ");
}

IEnumerator StartTestCoroutine()
{
    UI.SetGameObjectText("InstructionText", "Memorize the 3 words below");
    SetAnswers();
    yield return new WaitForSeconds(10);

    UI.SetGameObjectText("InstructionText", "What words did I ask you to memorize? (answer after the beep)");
    RemoveAnswers();
    StartCoroutine(Listen());
}

IEnumerator Listen()
{
    yield return new WaitForSeconds(3);
    GetSpeech();
}

public override void EndTest()
{
    Debug.Log("End Test");
    speechManager.Release();
    gameManager.AddToAssessmentScore(testScore);
    gameManager.NextTest();
}

public void GetSpeech()
{
    speechManager.StartListening(1, "en-US");
}

    void OnSpeechError(string error)
    {
            // Speech Recognition TimeOut or Speech Recognition Error
            if (answer == null) {
                    UI.SetGameObjectText("InstructionText", "What words did I ask you to memorize? (answer afte
            } else {
                    UI.SetGameObjectText("InstructionText", "Please say either YES or NO (answer after the beep
            }
            StartCoroutine(Listen());
```

```
    }

void OnSpeechResults(string results)
{

    // Need to parse
    string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit
    UI.SetGameObjectText("Answer3Text", texts[0]);

    if (answer == null)
    {
        answer = ParseSpeech(texts);
        ConfirmAnswer(answer);
    }
    else
    {
        ParseSpeech(texts);
        if (texts[0].Contains("yes"))
        {
            currentTrial++;
            CheckAnswer(answer);
        }
        else
        {
                            answer = null;
                            UI.SetGameObjectText("InstructionText", "Please say either YES or NO (answer after
                            StartCoroutine(Listen());
        }
    }

}



public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text;
}


public string ParseSpeech(string[] detectedWords)
{
    foreach (string temp in detectedWords)
    {
        return RemovePunctuations(temp);
    }
    return null;
}

public void ConfirmAnswer(string answer)
{
    UI.SetGameObjectText("InstructionText", "You said: " + answer + " \n Is this your answer? (Say Yes/No after
    StartCoroutine(Listen());
}


public void CheckAnswer(string answer)
{
    string[] words = answer.Split(' ');
    foreach (string word in words)
    {
        foreach (string ans in answers)
        {
            if (word == ans && currentTrial == 1)
            {
                testScore++;
                //UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());
            }
        }
    }
    if (testScore == 3 || currentTrial == numOfTrials)
    {
        EndTest();
    }
    else
    {
        answer = null;
        StartCoroutine(StartTestCoroutine());
    }

}

public override void AddChoiceScriptToGameObject() { }

public override void IsCorrectAnswer(int choice) { }
```

```
    public override void SetQuestion() { }

    public override void NextQuestion() { }
}


using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MathTest : Test {

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int numOfTrials;
    private int currentTrial;
    private int questionCounter;
    private int value;


    private string reminder;

    public MathTest(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }

    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);
        reminder = " (answer after the beep)";
        testScore = 0;
        questionCounter = 0;
        UI = (new GameObject("UIController")).AddComponent<UIController>();
        value = 100;

        answer = null;
        RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        UI.SetGameObjectText("InstructionText", "Now, could you subtract 7 from 100?)");
        StartCoroutine(Listen());
    }


    private void RemoveAnswers()
    {
        UI.SetGameObjectText("Answer1Text", " ");
        UI.SetGameObjectText("Answer2Text", " ");
        UI.SetGameObjectText("Answer3Text", " ");
    }

    IEnumerator Listen()
    {
        yield return new WaitForSeconds(3);
        GetSpeech();
    }

    public override void EndTest()
    {
        speechManager.Release();
        Debug.Log("End Test");
        gameManager.AddToAssessmentScore(testScore);
        gameManager.NextTest();
    }

    public void GetSpeech()
    {
        speechManager.StartListening(1, "en-US");
    }

    void OnSpeechError(string error)
    {
                if (answer == null) {
                        if (value == 100) {
                                UI.SetGameObjectText("InstructionText", "Now, could you subtract 7 from 100?)");
                        } else {
                                UI.SetGameObjectText("InstructionText", "Now subtract another 7 from your previous
                        }
                } else {
                        UI.SetGameObjectText("InstructionText", "Please say either YES or NO (answer after the beep
                }
                StartCoroutine(Listen());
    }
```

```
void OnSpeechResults(string results)
{
    string[] texts = null;
    texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplitOptions.

    if (answer == null)
    {
        answer = ParseSpeech(texts);
        ConfirmAnswer(answer);
    }
    else
    {
        if (texts[0].Contains("yes"))
        {
            currentTrial++;
            CheckAnswer(answer);
        }
        else
        {
            answer = null;
            UI.SetGameObjectText("InstructionText", "Answer the question again");
                        StartCoroutine(TryAgain());

        }
    }

}

    public IEnumerator TryAgain(){

            yield return new WaitForSeconds(3);
            UI.SetGameObjectText("InstructionText", "Now subtract another 7 from your previous answer");
            StartCoroutine(Listen());
    }

public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text;
}


public string ParseSpeech(string[] detectedWords)
{
    foreach (string temp in detectedWords)
    {
        return RemovePunctuations(temp);
    }
    return null;
}

public void ConfirmAnswer(string answer)
{
    UI.SetGameObjectText("InstructionText", "You said: " + answer + " \n Is this your answer? (Say Yes/No after
    StartCoroutine(Listen());
}

public void CheckAnswer(string answer)
{
    Debug.Log(answer);
    Debug.Log(value-7);
    if (answer == (value - 7).ToString())
    {
        testScore++;
        UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());
    }
    value = int.Parse(answer);
    if (questionCounter == 4)
    {
        EndTest();
    }
    else
    {
        answer = null;
        NextQuestion();
    }
}




public override void AddChoiceScriptToGameObject() { }

public override void IsCorrectAnswer(int choice) { }

public override void SetQuestion(){}
```

```
    public override void NextQuestion()
    {
        UI.SetGameObjectText("InstructionText", "Now subtract another 7 from your previous answer");
        answer = null;
        questionCounter++;
        StartCoroutine(Listen());
    }
    public override void StartTest() { }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Naming : Test {

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int numOfTrials;
    private int currentTrial;
    private int questionCounter;

    private string[] questions;
    private string reminder;

    public Naming(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }

    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);
        reminder = "(answer after the beep)";
        testScore = 0;
        questionCounter = 0;
        UI = (new GameObject("UIController")).AddComponent<UIController>();

        GenerateQuestion();
        answer = null;
        RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }


    private void RemoveAnswers()
    {
        UI.SetGameObjectText("Answer1Text", " ");
        UI.SetGameObjectText("Answer2Text", " ");
        UI.SetGameObjectText("Answer3Text", " ");
    }

    IEnumerator StartTestCoroutine()
    {
        UI.SetGameObjectText("InstructionText", "I will show you some pictures and I want you to identify what is i

        yield return new WaitForSeconds(5);
        UI.SetGameObjectText("InstructionText", " ");

        SetQuestion();

    }

    IEnumerator Listen()
    {
        yield return new WaitForSeconds(5);
        GetSpeech();

    }

    public override void EndTest()
    {
        speechManager.Release();
        UI.DeactivateRawImage("NamingImage");

        Debug.Log("End Test");
        gameManager.AddToAssessmentScore(testScore);
        gameManager.NextTest();
    }
```

```
    public void GetSpeech()
    {
        speechManager.StartListening(1, "en-US");
    }

    void OnSpeechError(string error)
    {
        // Speech Recognition TimeOut or Speech Recognition Error
                if (answer == null) {
                        UI.SetGameObjectText("InstructionText", "Identify what the picture is of " +reminder);
                } else {
                        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer?
\n (Say Yes/No after the beep)");
                }
                StartCoroutine(Listen());
    }

    void OnSpeechResults(string results)
    {
        // Need to parse
        string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit

        if (answer == null)
        {
            answer = ParseSpeech(texts);
            ConfirmAnswer(answer);
        }
        else
        {
            ParseSpeech(texts);
            if (texts[0].Contains("yes"))
            {
                currentTrial++;
                CheckAnswer(answer);
            }
            else
            {
                answer = null;
                            UI.SetGameObjectText("PlayerText", "Identify what the picture is of " +reminder);
                StartCoroutine(Listen());

            }
        }

    }



    public string RemovePunctuations(string text)
    {
        var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
        foreach (var c in charsToRemove)
        {
            text = text.Replace(c, string.Empty);
        }
        return text;
    }


    public string ParseSpeech(string[] detectedWords)
    {
        foreach (string temp in detectedWords)
        {
            return RemovePunctuations(temp);
        }
        return null;
    }

    public void ConfirmAnswer(string answer)
    {
        UI.SetGameObjectText("PlayerText", "You said: " + answer + " Is this your answer?  \n (Say Yes/No after the
        StartCoroutine(Listen());

    }


    public void CheckAnswer(string answer)
    {
        if (answers[questionCounter].Contains(answer) || answers[questionCounter] == answer)
        {
            testScore++;
            UI.SetGameObjectText("PlayerText", "Score: " + testScore.ToString());
        }
        if (questionCounter == 11)
        {
            EndTest();
        }
        else
        {
            NextQuestion();
        }
    }
```

```
    private void GenerateQuestion()
    {
        answers = new string[12];
        questions = new string[12];

        questions[0] = "Pen";
        answers[0] = "pencil";

        questions[1] = "Watch";
        answers[1] = "watch";

        questions[2] = "Dog";
        answers[2] = "dog";

        questions[3] = "Penguin";
        answers[3] = "penguin";

        questions[4] = "Anchor";
        answers[4] = "anchor";

        questions[5] = "Cat";
        answers[5] = "cat";

        questions[6] = "Guitar";
        answers[6] = "guitar";

        questions[7] = "Elephant";
        answers[7] = "elephant";

        questions[8] = "Barrel";
        answers[8] = "barrel";

        questions[9] = "Crown";
        answers[9] = "crown";

        questions[10] = "Eagle";
        answers[10] = "eagle";

        questions[11] = "Violin";
        answers[11] = "violin";

    }

    public override void AddChoiceScriptToGameObject() { }

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion()
    {
        UI.ActivateRawImage("NamingImage");
        UI.SetRawImageTexture("NamingImage", questions[questionCounter]);
        StartCoroutine(Listen());
    }

    public override void NextQuestion()
    {
        answer = null;
        questionCounter++;
        SetQuestion();
    }

    public override void StartTest() { }
}


using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;
using System;

public class OrientationTest : Test
{

    public GameManager gameManager;
    public int testScore;


    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int questionCounter;

    private string[] questions;
    private string reminder;

    public OrientationTest(GameManager gg, Question[] questions) : base(gg)
    {
```

```
    Initialize();
}

public override void Initialize()
{
    speechManager = new SpeechRecognizerManager(gameObject.name);
    reminder = " (answer after the beep)";
    testScore = 0;
    questionCounter = 0;
    UI = (new GameObject("UIController")).AddComponent<UIController>();

    answers = new string[5];
    questions = new String[5];
    GenerateQuestion();
    answer = null;
    RemoveAnswers();
    UI.SetGameObjectText("PlayerText", " ");
    StartCoroutine(StartTestCoroutine());
}


private void RemoveAnswers()
{
    UI.SetGameObjectText("Answer1Text", " ");
    UI.SetGameObjectText("Answer2Text", " ");
    UI.SetGameObjectText("Answer3Text", " ");
}

IEnumerator StartTestCoroutine()
{
    UI.SetGameObjectText("InstructionText", "Now, I am going to be asking you, a series of questions?)");
    yield return new WaitForSeconds(10);
    SetQuestion();
}

IEnumerator Listen()
{
    yield return new WaitForSeconds(3);
    GetSpeech();

}

public override void EndTest()
{
    speechManager.Release();
    Debug.Log("End Test");
    gameManager.AddToAssessmentScore(testScore);
    gameManager.NextTest();
}

public void GetSpeech()
{
    speechManager.StartListening(1, "en-US");

}

void OnSpeechError(string error)
{
            /* Speech Recognition TimeOut or Speech Recognition Error
            if (answer == null) {
                    UI.SetGameObjectText("InstructionText", questions[questionCounter] +reminder);
            } else {
                    UI.SetGameObjectText("InstructionText", "Please say either YES or NO (answer after the beep
            }
    StartCoroutine(Listen());
    */
            ConfirmAnswer ("25");
}

void OnSpeechResults(string results)
{
    string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit
    if (texts.Length == 0) {
        StartCoroutine(Listen());
    }
    if (answer == null)
    {
        answer = ParseSpeech(texts);
        ConfirmAnswer(answer);
    }
    else
    {
        ParseSpeech(texts);
        if (texts[0].Contains("yes"))
        {
            CheckAnswer(answer);
        }
        else
        {
            answer = null;
                        UI.SetGameObjectText("InstructionText", "Please say either YES or NO (answer after
            StartCoroutine(Listen());
```

```
        }
    }

}

public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text;
}


public string ParseSpeech(string[] detectedWords)
{
    foreach (string temp in detectedWords)
    {
        return RemovePunctuations(temp);
    }
    return null;
}

public void ConfirmAnswer(string answer)
{
    UI.SetGameObjectText("InstructionText", "You said: " + answer + " \n Is this your answer? (Say Yes/No after
    StartCoroutine(Listen());
}


public void CheckAnswer(string answer)
{
    string[] words = answer.Split(' ');
    foreach (string word in words)
    {
          if (word == answers[questionCounter])
            {
                testScore++;

                UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());
            }
    }
    if (questionCounter == 4)
    {
        EndTest();
    }
    else
    {
        NextQuestion();
    }
}

private void GenerateQuestion() {
    DateTime dateTime = System.DateTime.Now;

    questions[0] = "What day of the week is it? (i.e. Monday)";
    answers[0]   = dateTime.DayOfWeek.ToString().ToLower();
    //Date
    questions[1] = "What is the date today? (i.e. 28)";
    answers[1] = dateTime.Day.ToString();
    //Month
    questions[2] = "What month is it? (i.e. May)";
    answers[2]   = GetMonthString(dateTime.Month);
    //Year
    questions[3] = "What year it? (i.e. 1996)";
    answers[3] = dateTime.Year.ToString();

    //Season
    questions[4] = "What season is it? (Rainy/Dry)";
    answers[4] = GetSeason(dateTime.Month);
}

private string GetMonthString(int monthValue)
{
    switch (monthValue)
    {
        case 1:
            return "january";
        case 2:
            return "february";
        case 3:
            return "march";
        case 4:
            return "april";
        case 5:
            return "may";
        case 6:
            return "june";
        case 7:
            return "july";
```

```
            case 8:
                return "august";
            case 9:
                return "september";
            case 10:
                return "october";
            case 11:
                return "november";
            default:
                return "december";

        }
    }

    private string GetSeason(int monthValue)
    {
        if (monthValue < 12 && monthValue > 4) {
            return "Rainy";
        }
        return "Dry";
    }

    public override void AddChoiceScriptToGameObject() { }

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion() {
        UI.SetGameObjectText("InstructionText", questions[questionCounter] +reminder);
        StartCoroutine(Listen());
    }

    public override void NextQuestion() {
        answer = null;
        questionCounter++;
        SetQuestion();
    }
    public override void StartTest() { }
}

[System.Serializable]
public class Question {
    public string question;
    public string choice1;
    public string choice2;
    public string choice3;
    public int answer;
    public Question(string choice1, string choice2, string choice3, int answer)
    {
        this.choice1 = choice1;
        this.choice2 = choice2;
        this.choice3 = choice3;
        this.answer = answer;
    }
    public Question(string question,string choice1, string choice2, string choice3, int answer)
    {
        this.question = question;
        this.choice1 = choice1;
        this.choice2 = choice2;
        this.choice3 = choice3;
        this.answer = answer;
    }
}

using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;
using System;

public class Reading : Test
{

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int numOfTrials;
    private int currentTrial;
    private int questionCounter;

    private string[] questions;
    private string reminder;

    public Reading(GameManager gg, Question[] questions) : base(gg)
```

```
    {
        Initialize();
    }

    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);
        reminder = "\n(answer after the beep)";
        testScore = 0;
        questionCounter = 0;
        UI = (new GameObject("UIController")).AddComponent<UIController>();

        answers = new string[5];
        questions = new String[5];
        GenerateQuestion();
        answer = null;
        RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }


    private void RemoveAnswers()
    {
        UI.SetGameObjectText("Answer1Text", " ");
        UI.SetGameObjectText("Answer2Text", " ");
        UI.SetGameObjectText("Answer3Text", " ");
    }

    IEnumerator StartTestCoroutine()
    {
        UI.SetGameObjectText("InstructionText", "Now, I want you to read the following worrds?)");

        yield return new WaitForSeconds(10);

        SetQuestion();

    }

    IEnumerator Listen()
    {
        yield return new WaitForSeconds(3);
        GetSpeech();
    }

    public override void EndTest()
    {
        speechManager.Release();
        Debug.Log("End Test");
        if (testScore == 5) {
            gameManager.AddToAssessmentScore(1);
        } else{
            gameManager.AddToAssessmentScore(0);
        }
        gameManager.NextTest();
    }

    public void GetSpeech()
    {
        speechManager.StartListening(1, "en-US");
    }


    void OnSpeechError(string error)
    {
                if (answer == null) {
                        UI.SetGameObjectText("InstructionText", questions[questionCounter] +reminder);
                } else {
                        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer?
\n (Say Yes/No after the beep)");
                }
                StartCoroutine(Listen());
    }

    void OnSpeechResults(string results)
    {

        // Need to parse
        string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit

        if (texts.Length == 0) {
            StartCoroutine(Listen());

        }

        if (answer == null)
        {
            answer = ParseSpeech(texts);

            ConfirmAnswer(answer);
        }
        else
```

```
    {
        if (texts[0].Contains("yes"))
        {
            currentTrial++;
            CheckAnswer(answer);
        }
        else
        {
            answer = null;
                        UI.SetGameObjectText("InstructionText", questions[questionCounter] +reminder);
            StartCoroutine(Listen());

        }
    }

}

public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text;
}

public string ParseSpeech(string[] detectedWords)
{
    foreach (string temp in detectedWords)
    {
        return RemovePunctuations(temp);
    }
    return null;
}

public void ConfirmAnswer(string answer)
{
    UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer? (Say Yes/No after th
    StartCoroutine(Listen());
}

public void CheckAnswer(string answer)
{
    string[] words = answer.Split(' ');
    foreach (string word in words)
    {
          if (word == answers[questionCounter])
            {
                testScore++;
                UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());
            }
    }
    if (questionCounter == 4)
    {
        EndTest();
    }
    else
    {
        NextQuestion();
    }
}

private void GenerateQuestion() {
    //
    questions[0] = "Saw";
    answers[0]   = "saw";
    //Date
    questions[1] = "Paint";
    answers[1] = "paint";
    //Month
    questions[2] = "Root";
    answers[2]   = "root";
    //Year
    questions[3] = "Go";
    answers[3] = "go";

    //Season
    questions[4] = "Height";
    answers[4] = "height";
}


public override void AddChoiceScriptToGameObject() { }

public override void IsCorrectAnswer(int choice) { }

public override void SetQuestion() {
    UI.SetGameObjectText("InstructionText", questions[questionCounter] +reminder);
    StartCoroutine(Listen());
}
```

```
    public override void NextQuestion() {
        answer = null;
        questionCounter++;
        SetQuestion();
    }
    public override void StartTest() { }
}

using UnityEngine;
using System.Collections;

public class Recall : Test {

    public GameManager gameManager;
    private int testScore,trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;


    public Recall(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }



    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);
        trialScore = 0;
        csv = LoadCSV("items");
        UI = (new GameObject("UIController")).AddComponent<UIController>();

                answers = new string[3];
        answers[0] = "lemon";
        answers[1] = "camera";
        answers[2] = "phone";
        answer = null;
                RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }

    public override void StartTest()
    {
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }

    private void SetAnswers()
    {
        UI.SetGameObjectText("Answer1Text", answers[0]);
        UI.SetGameObjectText("Answer2Text", answers[1]);
        UI.SetGameObjectText("Answer3Text", answers[2]);
    }
    private void RemoveAnswers()
    {
        UI.SetGameObjectText("Answer1Text", " ");
        UI.SetGameObjectText("Answer2Text", " ");
        UI.SetGameObjectText("Answer3Text", " ");
    }

    IEnumerator StartTestCoroutine()
    {
        UI.SetGameObjectText("InstructionText", "What 3 words did I ask you to remember? (answer after the beep)");
        yield return new WaitForSeconds(2);
        StartCoroutine(Listen());
    }

    IEnumerator Listen() {
        yield return new WaitForSeconds(3);
        GetSpeech();

    }

    public override void EndTest()
    {
        UI.SetGameObjectText("InstructionText", "End Test");
        speechManager.Release();
        gameManager.AddToAssessmentScore(testScore);
        gameManager.NextTest();
    }

    public void GetSpeech()
    {
```

```
        speechManager.StartListening(1, "en-US");
    }

    void OnSpeechError(string error)
    {
        // Speech Recognition TimeOut or Speech Recognition Error
                if (answer == null) {
                        UI.SetGameObjectText ("InstructionText", "What 3 words did I ask you to remember? (answer a
                } else {
                        UI.SetGameObjectText("InstructionText", "Please say either YES or NO (answer after the beep
                }
        StartCoroutine(Listen());
    }


    void OnSpeechResults(string results)
    {

        // Need to parse
        string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit

        if (answer == null)
        {
            answer = ParseSpeech(texts);
            ConfirmAnswer(answer);
        }
        else
        {
            ParseSpeech(texts);
            if (texts[0].Contains("yes"))
            {
                CheckAnswer(answer);
            }
            else
            {
                answer = null;
                            UI.SetGameObjectText ("InstructionText", "What 3 words did I ask you to remember? (
                StartCoroutine(Listen());
            }
        }

    }

    public string RemovePunctuations(string text) {
        var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
        foreach (var c in charsToRemove)
        {
            text = text.Replace(c, string.Empty);
        }
        return text;
    }


    public string ParseSpeech(string[] detectedWords) {
        foreach (string temp in detectedWords) {
            return RemovePunctuations(temp);
        }
        return null;
    }

    public void ConfirmAnswer(string answer)
    {
        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer? (Say Yes/No after th
        StartCoroutine(Listen());

    }

    public void CheckAnswer(string answer) {
        string[] words = answer.Split(' ');
        foreach (string word in words) {
            foreach (string ans in answers) {
                if (word == ans) {
                    testScore++;
                }
            }
        }
        EndTest();
    }

    public override void AddChoiceScriptToGameObject(){}

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion() { }

    public override void NextQuestion() { }
}


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

```
public class Repetition : Test {

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;

    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;

    private int numOfTrials;
    private int currentTrial;
    private int questionCounter;

    private string[] questions;
    private string reminder;

    public Repetition(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }

    public override void Initialize()
    {
        speechManager = new SpeechRecognizerManager(gameObject.name);
        reminder = "(answer after the beep)";
        testScore = 0;
        questionCounter = 0;
        UI = (new GameObject("UIController")).AddComponent<UIController>();

        answers = new string[5];
        questions = new string[3];

        questions[0] = "hippopotamus, eccentricity, unintelligible, statistician";
        questions[1] = "above, beyond, and below";
        questions[2] = "no ifs, ands or buts";
        answers[0] = "hippopotamus eccentricity unintelligible statistician";
        answers[1] = "above beyond and below";
        answers[2] = "no ifs ands or buts";


        answer = null;
        RemoveAnswers();
        UI.SetGameObjectText("PlayerText", " ");
        StartCoroutine(StartTestCoroutine());
    }


    private void RemoveAnswers()
    {
        UI.SetGameObjectText("Answer1Text", " ");
        UI.SetGameObjectText("Answer2Text", " ");
        UI.SetGameObjectText("Answer3Text", " ");
    }

    IEnumerator StartTestCoroutine()
    {
        UI.SetGameObjectText("InstructionText", "Now, I need you to repeat the words/phrases that I show?)");
        yield return new WaitForSeconds(10);
        SetQuestion();
    }

    IEnumerator Listen()
    {
        yield return new WaitForSeconds(3);
        GetSpeech();
    }

    public override void EndTest()
    {
        UI.SetGameObjectText("InstructionText", "End Test");
        speechManager.Release();
        Debug.Log("End Test");
        gameManager.AddToAssessmentScore(testScore);
        gameManager.NextTest();
    }

    public void GetSpeech()
    {
        speechManager.StartListening(1, "en-US");
    }

    void OnSpeechError(string error)
    {
                if (answer == null) {
                        UI.SetGameObjectText("InstructionText", questions[questionCounter] +reminder);
                } else {
                        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer?
```

```
\n (Say Yes/No after the beep)");
                }
                StartCoroutine(Listen());
    }

    void OnSpeechResults(string results)
    {
        // Need to parse
        string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit

        if (answer == null)
        {
            answer = ParseSpeech(texts);
            ConfirmAnswer(answer);
        }
        else
        {
            ParseSpeech(texts);
            if (texts[0].Contains("yes"))
            {
                currentTrial++;
                CheckAnswer(answer);
            }
            else
            {
                answer = null;
                            UI.SetGameObjectText("InstructionText", "Please say either YES or NO (answer after
                StartCoroutine(Listen());
            }
        }

    }



    public string RemovePunctuations(string text)
    {
        var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
        foreach (var c in charsToRemove)
        {
            text = text.Replace(c, string.Empty);
        }
        return text;
    }


    public string ParseSpeech(string[] detectedWords)
    {
        foreach (string temp in detectedWords)
        {
            return RemovePunctuations(temp);
        }
        return null;
    }

    public void ConfirmAnswer(string answer)
    {
        UI.SetGameObjectText("InstructionText", "You said: " + answer + " Is this your answer? (Say Yes/No after th
        StartCoroutine(Listen());
    }


    public void CheckAnswer(string answer)
    {

        if (answer == answers[questionCounter])
        {
            testScore++;
        }

        if (questionCounter == 2)
        {
            EndTest();
        }
        else
        {
            NextQuestion();
        }
    }

    public override void AddChoiceScriptToGameObject() { }

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion()
    {
        UI.SetGameObjectText("InstructionText", questions[questionCounter] + reminder);
        StartCoroutine(Listen());
    }

    public override void NextQuestion()
    {
```

```
        answer = null;
        questionCounter++;
        SetQuestion();
    }
    public override void StartTest() { }
}

using UnityEngine;
using System.Collections.Generic;

/// <summary>
/// Unity Speech Recognizer manager.
/// </summary>
public class SpeechRecognizerManager
{
        private const string JAVA_CLASS_NAME = "com.takohi.unity.plugins.SpeechRecognizerManager";
        public const string RESULT_SEPARATOR = "/::/";
        public const int        EVENT_SPEECH_READY = -1; // On Speech ready.
        public const int        EVENT_SPEECH_BEGINNING = -2; // On beginning of Speech.
        public const int        EVENT_SPEECH_END = -3; // On end of Speech.
        public const int        ERROR_NOT_INITIALIZED = -1; // Not initialized.
        public const int        ERROR_AUDIO = 3; // Audio recording error.
        public const int        ERROR_CLIENT = 5;       // Other client side errors.
        public const int        ERROR_INSUFFICIENT_PERMISSIONS = 9; // Insufficient permissions
        public const int        ERROR_NETWORK = 2;      // Other network related errors.
        public const int        ERROR_NETWORK_TIMEOUT = 1;      // Network operation timed out.
        public const int        ERROR_NO_MATCH = 7;     // No recognition result matched.
        public const int        ERROR_RECOGNIZER_BUSY = 8;      // RecognitionService busy.
        public const int        ERROR_SERVER = 4;       // Server sends error status.
        public const int        ERROR_SPEECH_TIMEOUT = 6;       // No speech input

        private static AndroidJavaObject srManager = null;

        public SpeechRecognizerManager (string gameObjectName)
        {
                if (Application.platform != RuntimePlatform.Android)
                        return;

                srManager = new AndroidJavaObject (JAVA_CLASS_NAME, new object[] {
                        gameObjectName
                });
        }

        /// <summary>
        /// Checks whether a speech recognition service is available on the system.
        /// </summary>
        /// <returns><c>true</c> if is available; otherwise, <c>false</c>.</returns>
        public static bool IsAvailable ()
        {
                if (Application.platform != RuntimePlatform.Android)
                        return false;

                AndroidJavaClass jc = new AndroidJavaClass (JAVA_CLASS_NAME);
                return jc.CallStatic<bool> ("isAvailable");
        }

        /// <summary>
        /// Starts the listening.
        /// </summary>
        /// <param name="maxResults">Max results (optional, default=5).</param>
        /// <param name="language">Preferred language (optional, default=null for default language).</param>
        public void StartListening(int maxResults=5, string language=null)
        {
                if (Application.platform != RuntimePlatform.Android)
                        return;

                srManager.Call ("startListening", language, maxResults);
        }

        /// <summary>
        /// Stops listening for speech. Speech captured so far will be recognized as if the user had stopped speaki
        /// </summary>
        public void StopListening ()
        {
                if (Application.platform != RuntimePlatform.Android)
                        return;

                srManager.Call ("stopListening");
        }

        /// <summary>
        /// Cancels the speech recognition.
        /// </summary>
        public void CancelListening ()
        {
                if (Application.platform != RuntimePlatform.Android)
                        return;

                srManager.Call ("cancel");
        }
```

```
        /// <summary>
        /// Releases the object.
        /// </summary>
        public void Release ()
        {
                if (Application.platform != RuntimePlatform.Android)
                        return;

                srManager.Call ("release");
        }
}

using UnityEngine;

public abstract class Test : MonoBehaviour{

    private GameManager gameManager;
    private CSVReader csvReader;
    private string[,] csv;

    public Test(GameManager gg) {
        this.gameManager = gg;
    }


    public abstract void StartTest();

    public abstract void SetQuestion();

    public abstract void NextQuestion();

    public abstract void IsCorrectAnswer(int x);

    public abstract void EndTest();

    public abstract void Initialize();

    public abstract void AddChoiceScriptToGameObject();

    public string[,] LoadCSV(string resourceName) {
        Debug.Log(resourceName);
        TextAsset textAsset = Resources.Load(resourceName) as TextAsset;
        csvReader = new CSVReader(textAsset);
        return csvReader.getCSV();

    }


}

using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.EventSystems;
using System.Collections.Generic;
public class UIController : MonoBehaviour{
    // Canvas Objects
    private TextMeshProUGUI playerText;
    public TextMeshProUGUI questionText;
    public TextMeshProUGUI choice1Text, choice2Text, choice3Text;
    // GameObjects
    private GameObject[] objects;
    private Test test;
    private int objectNumber;
    private int objectCounter;

    public UIController(Test test)
    {
        this.test = test;
        objectCounter = 0;
        playerText = GameObject.Find("PlayerText").GetComponent<TextMeshProUGUI>();
    }

    public void SetPlayerText(string text)
    {
        playerText.text = text;
    }

    public void ActivateRawImage(string gameObjectName)
    {
        GameObject.Find(gameObjectName).GetComponent<RawImage>().enabled = true;

    }
    public void DeactivateRawImage(string gameObjectName)
    {
        GameObject.Find(gameObjectName).GetComponent<RawImage>().enabled = false;

    }

    public void SetRawImageTexture(string gameObjectName, string prefabName)
    {
```

```
    Texture item = Resources.Load(prefabName) as Texture;
    GameObject.Find(gameObjectName).GetComponent<RawImage>().texture = item;
}

public void ActivateGameObject(string gameObjectName)
{
    GameObject.Find(gameObjectName).GetComponent<Canvas>().enabled = true;
}

public void DeactivateGameObject(string gameObjectName)
{
    GameObject.Find(gameObjectName).GetComponent<Canvas>().enabled = false;
}

public void DeactivateButton(string gameObjectName)
{
    GameObject.Find(gameObjectName).GetComponent<Button>().enabled = false;
}


public void ActivateButton(string gameObjectName)
{
    Debug.Log(gameObjectName);
    GameObject.Find(gameObjectName).GetComponent<Button>().enabled = true;
}

public void ActivateGameObj(string gameObjectName)
{
    Debug.Log(gameObjectName);
    GameObject.Find(gameObjectName).SetActive(true);

}

public void RemoveButton(string gameObjectName)
{
    Destroy(GameObject.Find(gameObjectName));
}



public void SetObjectNumber(int x)
{
    objects = new GameObject[x];
}

public GameObject[] GetGameObjects() {
    return objects;
}

public Button[] FindButtonsInComponent(string componentName) {
    return GameObject.Find(componentName).GetComponentsInChildren<Button>();
}

public void SetUIReferences()
{
    questionText = GameObject.Find("QuestionText").GetComponent<TextMeshProUGUI>();
    choice1Text = GameObject.Find("Choice1Text").GetComponent<TextMeshProUGUI>();
    choice2Text = GameObject.Find("Choice2Text").GetComponent<TextMeshProUGUI>();
    choice3Text = GameObject.Find("Choice3Text").GetComponent<TextMeshProUGUI>();
}

public void SetButtonText(GameObject gameObject, string text) {
    gameObject.GetComponent<TextMeshProUGUI>().text = text;
}

public void UpdateCanvas(string question, string choice1, string choice2, string choice3)
{
    questionText.text = question;
    choice1Text.text = choice1;
    choice2Text.text = choice2;
    choice3Text.text = choice3;
}

public void UpdateCanvas(string choice1, string choice2, string choice3)
{
    choice1Text.text = choice1;
    choice2Text.text = choice2;
    choice3Text.text = choice3;
}

public void PlaceGameObject(string prefabName)
{
    Debug.Log(prefabName);
    GameObject item = Instantiate(Resources.Load(prefabName)) as GameObject;
    objects[objectCounter] = item;
    objectCounter++;
}

public void PlaceGameObject(string prefabName, float pos_x, float pos_y, float pos_z, float rot_x, float rot_y,
{

    Vector3 position = new Vector3(pos_x, pos_y, pos_z);
```

```
        GameObject item = Instantiate(Resources.Load(prefabName)) as GameObject;
        item.gameObject.transform.position = new Vector3(pos_x,pos_y,pos_z);

        item.gameObject.transform.rotation = Quaternion.Euler(rot_x,rot_y,rot_z);
        objects[objectCounter] = item;
        objectCounter++;
    }

    public void PlaceGameObjects(string[] prefabNames)
    {
        for (int i = 0; i < prefabNames.Length; i++)
        {
            GameObject item = Instantiate(Resources.Load(prefabNames[i])) as GameObject;
            objects[i] = item;
            objectCounter++;
        }

    }

    public void SetGameObjectLocation(string name, float x, float y, float z)
    {

        GameObject.Find(name).gameObject.transform.position = new Vector3(x, y, z);
    }

    public void SetCurrentGameObjectLocation(float x, float y, float z)
    {

        objects[objectCounter - 1].gameObject.transform.position = new Vector3(x, y, z);
    }

    public void SetGameObjectRotation(string name, float x, float y, float z)
    {
        GameObject.Find(name).gameObject.transform.Rotate(new Vector3(x, y, z));
        GameObject.Find(name).transform.rotation = Quaternion.Euler(x, y, z);
    }

    public void SetCurrentGameObjectRotation(float x, float y, float z)
    {
        objects[objectCounter - 1].gameObject.transform.Rotate(new Vector3(x, y, z));
    }

    public GameObject GetCurrentGameObject() {
        return objects[objectCounter-1];
    }



    public void DestroyGameObjects()
    {
        for (int i = 0; i < objects.Length; i++)
        {
            Destroy(objects[i]);
        }
        objectCounter = 0;

    }

    public void SetGameObjectText(string objectName, string text) {
        GameObject.Find(objectName).GetComponent<TextMeshProUGUI>().text = text;
    }


}


using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class VerbalFluency : Test {

    public GameManager gameManager;
    private int testScore, trialScore;

    private CSVReader csvReader;
    private string[,] csv;
    private UIController UI;
        private string instruction;
    private SpeechRecognizerManager speechManager;
    private string[] answers;
    private string answer;
    private float startTime,currentTime, timeElapsed=0f;
    private string reminder;
        private ArrayList arrayList;
    public VerbalFluency(GameManager gg, Question[] questions) : base(gg)
    {
        Initialize();
    }

    public override void Initialize()
```

```
{
    speechManager = new SpeechRecognizerManager(gameObject.name);
    reminder = " (answer after the beep)";
    testScore = 0;
            arrayList = new ArrayList();
    UI = (new GameObject("UIController")).AddComponent<UIController>();
    answer = null;
    RemoveAnswers();
            instruction = "Now, I need you to generate as many words as you can that begin with the letter P.";
            UI.SetGameObjectText("PlayerText", " ");
    StartCoroutine(StartTestCoroutine());
}


private void RemoveAnswers()
{
    UI.SetGameObjectText("Answer1Text", " ");
    UI.SetGameObjectText("Answer2Text", " ");
    UI.SetGameObjectText("Answer3Text", " ");
}

IEnumerator StartTestCoroutine()
{
            UI.SetGameObjectText("InstructionText", instruction + reminder);
    yield return new WaitForSeconds(2);
    StartCoroutine(Listen());
}

IEnumerator Listen()
{
    yield return new WaitForSeconds(4);
    startTime = Time.time;
    GetSpeech();

}

public void GetSpeech()
{
    speechManager.StartListening(1, "en-US");
}

void OnSpeechResults(string results)
{
    timeElapsed += Time.time - startTime;
    string[] texts = results.Split(new string[] { SpeechRecognizerManager.RESULT_SEPARATOR }, System.StringSplit
    answer = ParseSpeech(texts);
    CheckAnswer(answer);
}

void OnSpeechError(string error)
{
            StartCoroutine(Listen());
}

public string RemovePunctuations(string text)
{
    var charsToRemove = new string[] { "@", ",", ".", ";", "'" };
    foreach (var c in charsToRemove)
    {
        text = text.Replace(c, string.Empty);
    }
    return text.ToLower();
}


public string ParseSpeech(string[] detectedWords)
{
    foreach (string temp in detectedWords)
    {
        return RemovePunctuations(temp);
    }
    return null;
}


public void CheckAnswer(string answer)
{
    string[] words = answer.Split(' ');
    foreach (string word in words)
    {
                    if (word[0] == 'p' && !arrayList.Contains(word))
        {
                            arrayList.Add (word);
            testScore++;
            UI.SetGameObjectText("Answer1Text", "Score: " + testScore.ToString());
        }
    }
    if (timeElapsed > 65f)
    {
        EndTest();
    }
    else
```

```
        {
            StartCoroutine(Listen());
        }
    }


    public override void EndTest()
    {
        speechManager.Release();
        Debug.Log("End Test");
        gameManager.AddToAssessmentScore(DetermineScore());
        gameManager.NextTest();
    }

    private int DetermineScore() {
        if (testScore < 5)
        {
            return 0;
        }
        else if (testScore < 7)
        {
            return 1;
        }
        else if (testScore < 9)
        {
            return 2;
        }
        else if (testScore < 11)
        {
            return 3;
        }
        else if (testScore < 14)
        {
            return 4;
        }
        else if (testScore < 17)
        {
            return 5;
        }
        else if (testScore <22)
        {
            return 6;
        }else
        {
            return 7;
        }
    }

    public override void AddChoiceScriptToGameObject() { }

    public override void IsCorrectAnswer(int choice) { }

    public override void SetQuestion(){}

    public override void NextQuestion(){}
    public override void StartTest() { }
}


using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
using System.Collections;
/* Virtual Object Memorization (VOM) Test
 * This test is meant to gauge the visuospatial capabilities of the subject.
 * Here we test their ability to construct an objects location from memory.
 *
 *
 */
public class VOM : Test {

    private int correctAnswer;
    private string[] choices;

    private int questionSet;
    private int testScore;
    private int questionCounter;
    public GameManager gameManager;
    private UIController UI;
    private float myTime;
    private bool startTimer;
    private static int numChoices;

    private string[,] objectData;

    private Question[] questions;
    private string[,] objectSet;
    private Button button;
    private string playerCanvasName;


    public VOM(GameManager gg) : base(gg)
```

```
    {
        Debug.Log("VOM Begin");
        gameManager = gg;
        Initialize();
    }

    public override void Initialize()
    {
        numChoices = 3;
        questionCounter = 0;
        testScore = 0;
        correctAnswer = Random.Range(1, 3);
        questionSet = correctAnswer;
        string[] choices = { "Laptop", "Book", "Apple", "Comb", "Crane",  "Cup", "Phone", "Pencil"
};
        myTime = 0f;

        objectSet = new string[8, 8];
        objectData = LoadCSV("VOM");

        UI = new UIController(this);
        UI.SetObjectNumber(8);
        PlaceObjectSet();
        //StartButton();
        startTimer = false;

        playerCanvasName = "InstructionText";
        // Create intro. Show table and mention what each object is?
    }




    private void PlaceChoiceSet()
    {
        string[] choices = { "Laptop", "Book", "Apple", "Comb", "Crane", "Cup", "Phone", "Pencil" };
        UI.SetGameObjectText("InstructionText", "Where was the " + choices[questionCounter] + " placed?");
        UI.SetGameObjectText(playerCanvasName, "Where was the " + choices[questionCounter] + " placed?");

        for (int i = 0; i < 24; i++)
        {
            if ((objectData[0, i] == choices[questionCounter]))
            {
                UI.PlaceGameObject(objectData[0, i], float.Parse(objectData[2, i]), float.Parse(objectData[3, i]),

            }
        }
    }

    private void PlaceObjectSet()
    {
        /* 0 = prefabname
             * 1 = answer
             * 2 = pos_x
             * 3 = pos_y
             * 4 = pos_z
             * 5 = rot_x
             * 6 = rot_y
             * 7 = rot_z
             *
             */
        for (int i = 0; i <24; i++)
        {
            if (int.Parse(objectData[1, i]) == correctAnswer)
            {
                UI.PlaceGameObject(objectData[0, i], float.Parse(objectData[2, i]), float.Parse(objectData[3, i]),
            }
        }
    }


    public override void IsCorrectAnswer(int choice)
    {
        Debug.Log("Correct Answer:" + correctAnswer.ToString());
        Debug.Log("Correct Answer:" + choice.ToString());
        if (choice == correctAnswer) {
            testScore++;
        }

        if (questionCounter == 7)
        {
            EndTest();
        }
        else
        {
            NextQuestion();
        }
```

```
    }

    public override void NextQuestion() {
        questionCounter++;
        UI.DestroyGameObjects();
        SetQuestion();
    }

    public override void StartTest()
    {
        Debug.Log("Game");
        UI.DestroyGameObjects();
        SetQuestion();
    }

    public override void SetQuestion()
    {
        string[] choices = { "Laptop", "Book", "Apple", "Comb", "Crane", "Cup", "Phone", "Pencil" };
        PlaceChoiceSet();

        AddChoiceScriptToGameObject();
    }

    public override void EndTest()
    {
        UI.DestroyGameObjects();
        gameManager.AddToAssessmentScore(testScore);
        gameManager.NextTest();
    }

    public override void AddChoiceScriptToGameObject()
    {
        GameObject[] objects = UI.GetGameObjects();
        for (int i = 0; i < numChoices; i++)
        {

            Choice cs = objects[i].gameObject.AddComponent<Choice>();
            EventTrigger et = objects[i].gameObject.AddComponent<EventTrigger>();
            EventTrigger.Entry entry = new EventTrigger.Entry
            {
                eventID = EventTriggerType.PointerClick
            };
            entry.callback.AddListener((eventData) => { cs.SendAnswer(); });
            objects[i].gameObject.GetComponent<EventTrigger>().triggers.Add(entry);
            cs = new Choice(this, i+1);

        }
    }

}
```

XI. Acknowledgement

Many moments in my life I had made rash and split second decisions. Filling up the UPCAT form and placing Computer Science there was one of them. By the time I got my acceptance letter I didn't even remember that I had placed it as a choice. Fast forward 4 years, here I am with a defended thesis, on the edge of graduating. Of course I did not get here on my own. To my professors who guided in learning how to google for code. To my adviser for accepting me as an advisee late march and taking me all the way to my defense. To my blockmates who were there with me on this insane journey. To my friends who were always there to help me out. To my siblings who always took care of me. To my dad who drove me to school. To my mom who supported me in every way she could. And to the one person whom without any of this would not have been possible, me. This SP could not have been possible without any of you. From the bottom of my heart, I thank you all.