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

Visual Performance Test for Patients with Glaucoma using Virtual Reality

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

Submitted by:

Angelene L. Ronquillo

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 "Visual Performance Test for Patients with Glaucoma using Virtual Reality" prepared and submitted by Angelene L. Ronquillo in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Marvin John C. Ignacio, 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. Perlita E. Gasmen, M.Sc. (*cand.*)		
5. Ma. Sheila A. Magboo, M.Sc.		
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

Glaucoma is an eye disease that damages the optic nerve. It worsens the visual field starting from the midperiphery and may develop towards the center. The most common test to detect glaucoma is the standard automated perimetry (SAP). However, glaucoma is detected only when 40% of the optic nerve is already damaged. Additionally, functional parameters that may affect visual performance of people with glaucoma are not tested in SAP.

People with glaucoma are still able to complete tasks but performs slower compared to people with normal vision. Glaucoma-VR is a mobile application that provides virtual environment with objectives to be accomplished by the user. The visual processing speed is the functional parameter used to assess the visual function of a user in the assessment tests.

Keywords: Glaucoma, Virtual Reality, Visual Function Test, Visual Performance Test, Initial Assessment Test

Contents

List of Figures

I. Introduction

A. Background of the Study

Glaucoma is an eye disease that damages the optic nerve which carries information from the eye to the brain [1]. It is caused by the impairment of retinal ganglion cells, or the neurons which is responsible for sending visual stimulations from the retina to brain via the optic nerves. Due to this, it worsens the visual field of a person which initially starts in the midperiphery and may develop towards the center [2]. According to World Health Organization, glaucoma is the second leading cause of irreversible vision loss worldwide next to cataracts [3]. The number of people,age 40-80 years old, suffering from glaucoma was estimated to 64.3 million in 2013. It is estimated to increase to 76.0 million in 2020 and 111.8 million in 2040 [4]. According to the Department of Health (DOH), in 2017, over two million Filipinos are bilaterally blind or suffering from poor vision and 14 percent was caused by glaucoma [5].

Glaucoma is also called as the "silent thief of sight" because there are no visible signs or symptoms at first to undercover glaucoma which is why most individuals are not aware that they have it [6]. According to Dr. Leo Cubillan, an ophthalmologist from the Philippine Eye Research Institute and ManilaMed, the visual field defect is detected only when 40% of the optic nerve were already affected. Consequently, diagnosis of glaucoma is often delayed. According to surveys, only 10 to 50% of people affected are aware of their condition [2].

People who have higher risks of acquiring glaucoma should immediately see eye care practitioners for glaucoma evaluation. These people are from the old population, have family history of glaucoma, from the black race, uses systemic or topical corticosteroids, have high intraocular pressure, etc. [2].

Glaucoma can be classified into two categories: (1) primary open-angle glaucoma (POAG) and (2) primary closed-angle glaucoma (PCAG). The primary open-angle glaucoma is the most common type wherein the eye cannot properly drain the aqueous humor (fluid in the eye). The pressure in the eye then builds up and starts to damage the optic nerve. The primary-closed angle glaucoma usually occurs when a person's iris is too close to the drainage angle, the angle between the iris and cornea. The iris blocks the drainage angle and when it is totally blocked, eye pressure increases quickly because excess fluid cannot flow out of the eye. This is known as an acute attack [6]. The damage caused by glaucoma is permanent and irreversible but further vision loss can be prevented early if a person regularly takes a complete eye examination.

There are various tests to detect glaucoma such as gonioscopy, opthalmoscopy, pachimetry, perimetry, tonometry, etc. Tonometry is used to measure the pressure in the eye. Comprehensive dilated eye exam is necessary to determine the level of normal eye pressure relative for each individual [7]. Pachymetry is used to measure corneal thickness that might affect the eye pressure [7]. Gonioscopy is used to check the structure of the drainage angle. If it is narrow or closed, then the aqueous humor might be blocked in the eye [8]. Opthalmoscopy, also known as funduscopy or retinal examination, is used to test the back part of the eye (fundus) which consists of the blood vessels, optic disc, and retina. This test is used to detect not only glaucoma but also other eye diseases and conditions affecting the blood vessels [9].

The most common test in assessing the visual field defects of an individual is visual field testing, also known as perimetry. It is used to measure the central and side (peripheral) vision [10]. Visual field test produces a map of a person's complete vision [8]. It uses a dark room with light control when testing [11]. Traditionally, the patient will sit in front of the machine, place chin on the ledge, look at the central target, and simultaneously press buttons when light is simulated [12]. Based on research, visual field test using standard automated perimetry (SAP) is more reliable than other visual field testing techniques [13].

Visual field test is essential for diagnosing and monitoring various ocular and neurological diseases. There are many devices widely used in standard automated perimetry (SAP) such as Humphrey Field Analyzer (HFA) and Octopus perimeter. However, visual field testing equipment are neither portable nor affordable to some people which limits some regions in accessing eye and health care according to Dr. Chris A. Johnson, Ph.D., director of the Visual Field Reading Center at the University of Iowa[14]. Aside from the cost and portability issues, the perimeters can cause discomfort and difficulty to children, elderly, and people with disabilities [12].

The use of technological advancement, such as desktop or mobile applications and virtual reality devices, would be helpful in addressing the issues of the conventional visual field testing. An example of a desktop application is EyesCream Visual Field Analyzer which is a Medical Game (MGM) intended to discover the impact of laser on the retina. It may also be used to detect glaucoma defects. However, the program was not tested yet for glaucoma subjects and it would possibly require a large monitor or connection to a large television screen [15]. The assessment of VF tests using mobile applications can be more convenient because it results to less ophthalmic visits, faster screening time, and easier follow-ups [16]. The constantly evolving interaction of medicine and technology led to the development of telemedicine. Telemedicine refers to the practice of medicine at a spatial or temporal distance by exchanging information via electric communication.

Automated perimetry is conducted through development and clinical testing of a compact, head-monted, and eye-tracking perimeter called Virtual Eye [17]. Another head-mounted perimeter is invented to remove the physical limitations of bulky machines and to measure a patient's reflex instantly when visual stimulus is presented, which is the VIP Virtual Perimetry [12]. These head-mounted perimeters aim to provide a more comfortable environment of visual field test than the standard instrumentation [17]. Virtual reality (VR) refers to a computer interface which involves real-time simulation of environment, scenario, or activity that allows user interaction via sensory channels. It provides a complex and multimodal sensory information, and an illusion of reality despite its artificial nature [18].

However, another limitation of standard automated perimetry is the relatively weak ability to measure parameters that may be linked to functional impairment of the patient due to glaucoma e.g. rate of motor vehicle collision or falls. Researchers found that virtual reality can provide more realistic testing environment than visual field testing. According to Felipe A. Medeiros, MD, "Measures from traditional static visual field tests do not mimic the visual conditions that occur day-to-day" [16]. The assessment of the functional field of view would better reflect the vision demanding tasks that occur every day, particularly when peripheral stimulus are required to monitor while concentrating on a central task simultaneously [19].

Peripheral vision is important for balance control, spatial cognition, navigation, obstacle avoidance, etc [20]. Visual field loss due to glaucoma affects the quality of life. The safety of a person in an environment is compromised such as becoming lost or disoriented when crossing streets. It also affects the construction of spatial cognitive maps needed in successful wayfinding which may correlate into significant difficulties in performing daily tasks that are navigation-dependent [2]. According to the Philippine Academy of Ophthalmology (PAO), peripheral vision is a better predictor of safe driving, compared to the central vision. Peripheral vision is the most important parameter for both static (e.g. avoidance of curbs or barricades) and dynamic situations (e.g. avoidance of collision). Though testing the peripheral visual field is important, the Land Transportation Office of the Philippines (LTO) do not require perimetry for all applicants of driver's license because it is impractical and costly [21]. Drivers with glaucoma show difficulties in observation, maintaining lane position, changing lanes, etc. Glaucoma patients are also associated with less physical activities, greater fear of falling, and increased risk of falls and injuries. Nevertheless, glaucoma patients are still able to complete tasks but they are slower than normal-visioned people. In order to compensate for their visual field defects, studies show that glaucoma patients have increase in eye patterns, head movements, and fixation durations [20].

Several applications were developed to test the visual functional difficulties experienced by older adults in performing daily activities which were not captured by the standard VF tests alone. Examples of these applications were the Useful Field of View (UFOV) Test and the PERformance-CEntered Portable Test (PERCEPT). The outcome measure used in these applications was the visual prcessing speed, rather than the visual field sensitivity.

The UFOV test evaluates the amount of visual information a person can perceive from a scenario in one glance [22]. It also measures how fast a person can cognitively process visual information within the radius of 30 degrees, also called as "visual processing speed" [23]. The visual processing speed is the minimum duration time to detect a peripheral stimulus while concentrating on a central task or stimulus, simultaneously. The UFOV test is a computer-administered and scored test via capturing an individual's ability of (1) rapid detection and localization of targets; (2) divided visual attention across central and peripheral vision; (3) and detection of relevant targets in the presence or absence of a visually cluttered array. Several studies showed that impaired performance on the UFOV is associated with mobility problems, balance, and increased risk of falling [24]. In [23], the test-retest correlation coefficient for the total performance of older adults was high (0.90). The researchers tested the psychometric properties of the UFOV. The results suggested moderate variability, greater for glaucoma patients compared to healthy controls, and a learning effect. Two consecutive tests are advised to acquire a reliable baseline measures [23].

The PERCEPT is an iPad app that challenged the visual performance of the participants through a demanding, time-constrained, dual visual task at a low contrast. The participants must perform a central task while simultaneously detecting a peripheral low contrast stimulus, at a short duration time. An important finding in this study is that the glaucoma participants perform slower at a low contrast dual visual task. The results were also significantly correlated to history of falls and motor vehicle collisions within the given population [19].

There is also a potential for improvement of visual processing speed and can be achieved through computer-based training programs. It can result to reduced car crash risk, positive health and functional well-being, and potential to increase mobility of older adults [24]. In conclusion, the visual processing speed was capable of detecting the presence of the functional damage in a significant proportion of glaucoma patients [19].

B. Statement of the Problem

As stated in background of the study, there are some problems with regard to conducting the standard visual field tests. Standard automated perimetry (SAP) has a relatively weak ability to evaluate parameters, such as visual processing speed, that may be related to functional impairment caused by glaucoma [16].

Conducting eye tests using a desktop or iPad app may incur some distractions from external sources, unlike in VR, the patient would be fully immersed in the visual field test or visual performance test. The virtual environment surrounds the viewer's entire field of view in multiple directions instead of being restricted within a monitor [25]. Virtual reality can provide more realistic environment testing and mimic everyday tasks and conditions that patients might encounter [26].

C. Objectives of the Study

This research study aims to create an Android application to test the visual performance of an individual. The application has the following functionalities:

1. Allows an eye doctor or any clinic staff to
   (a) Choose among the virtual environment.
      i. Supermarket
      ii. Forest
   (b) Choose which eye to test.
      i. Left eye

ii. Right eye

(c) View performance results of patients.

i. Supermarket

A. The number of perceived peripheral items by the user over the total number of items in the visual field.

B. The number of missed peripheral items (false negatives) by the user over the total number of items in the visual field.

C. The shortest duration time (visual processing speed) or time of initial response to detect a peripheral item.

D. The average duration time (visual processing speed) or average time of initial response to detect a peripheral item.

ii. Forest

A. The number of correctly identified objects.

B. The shortest presentation time that the user can detect the central target and locate the position of peripheral items.

C. The number of incorrectly identified objects and their location/s.

2. Allows the patient to

(a) Perform the objectives in the following virtual environment using the viewing condition selected by the eye doctor.

i. Supermarket

A. The patient must initially focus his gaze on a central point.

B. The patient should be able to locate an illuminating item from the shelves while focusing on a fixation point.

ii. Forest

A. The patient will focus his gaze on a fixation point while recognizing the items visible from the periphery.

B. The patient must correctly answer which is the center object displayed.

C. The patient must correctly answer where the peripheral object is located among the eight choices of location.

iii. View performance results.

A. Supermarket

B. Forest

D. Significance of the Project

The application acts both as screening and prevention tool since it provides an initial assessment test of an individual's visual performance. The application can also be used by the LTO for testing the peripheral vision of the applicants for driver's license. The application works like a game with a goal and scoring function using the concept of Useful Field of View (UFOV) test. It provides a challenging yet entertaining visual performance test to the patients.

Virtual reality environments can be used as an effective approach for assessing functional vision in patients and for clinical trials of emerging therapies in ophthalmology [27]. Using VR mobile application and a head-mounted device to test visual performance serve as a mobile telemedicine system which provides a precursor if a patient is suffering from visual field loss.

E. Scope and Limitations

1. The application can be used on a smartphone provided that it has a minimum version of Android 4.4 (Kitkat).

2. The application is designed to test only the visual functional abilities of patients.

3. The application does not provide a full analysis or interpretation of the patient's eye condition.

4. Immobile patients are excluded in using the application since head and body movements are required in the assessment tests.

F. Assumptions

1. There is an eye doctor or any clinic staff present while conducting the entire visual performance tests.

2. The eye doctor is the only one accountable for diagnosing patient with glaucoma and not the application itself.

II. Review of Related Literature

Today's many areas of healthcare rely in virtual reality (VR) such as surgical training, exposure therapy for treating post-traumatic stress disorder (PTSD), cognitive behavioral t7herapy, and ophthalmology. VR environment is an essential approach to assess the visual function endpoints in clinical trials of emerging ophthalmological therapies [27].

Eye-tracking. Kasneci et.al. investigated the eye movement patterns of an individual with glaucoma while doing daily tasks such as walking, driving, etc. Eye-tracking wass used as a tool to assess and analyze the visual search behavior of glaucomatous individuals under real-world conditions. Some of the recent development of video-based eye-tracking technology using head-mounted and remote technology are Dikablis Mobile, Pupil Labs, SMI Glasses and Tobii Glasses [20]. The studies cited in [20] reported that the reading performance of glaucoma patients was slower compared to the control group with normal vision for texts with small at low contrast levels. Smith et.al. reported that the reading capacity of the eye with glaucomatous field loss was worse compared to the eye with better vision.

Wayfinding Behavior. Daga et.al. assessed the wayfinding behavior and spatial cognition of glaucoma participants in an immersive virtual environment, the Virtual Environment Human Navigation Task (VEHuNT). The VEHuNT was implemented using a cave automatic virtual environment (CAVE) VR system, the 4kAVE. During the wayfinding test, patients wore polarizing glasses to recognize the 3D images and to navigate through the virtual environment by using a steering wheel and an accelerator pedal. Two virtual rooms used were geometrically identical but different in complexity of the scene. The first room (room A) contained multiple peripheral vision cues to present an allocentric reference frame and cognitive mapping for wayfinding. The second room (room B) contained a single central cue, a chair near the center of the room, to present an egocentric reference frame-based navigation. It decreases the need of a cognitive map in wayfinding tasks. Glaucoma patients perform significantly worse on allocentric-based navigation tasks in a virtual environment which suggests that visual field loss affects spatial cognition. Glaucoma patients might experience difficulty accomplishing navigation-dependent tasks [27].

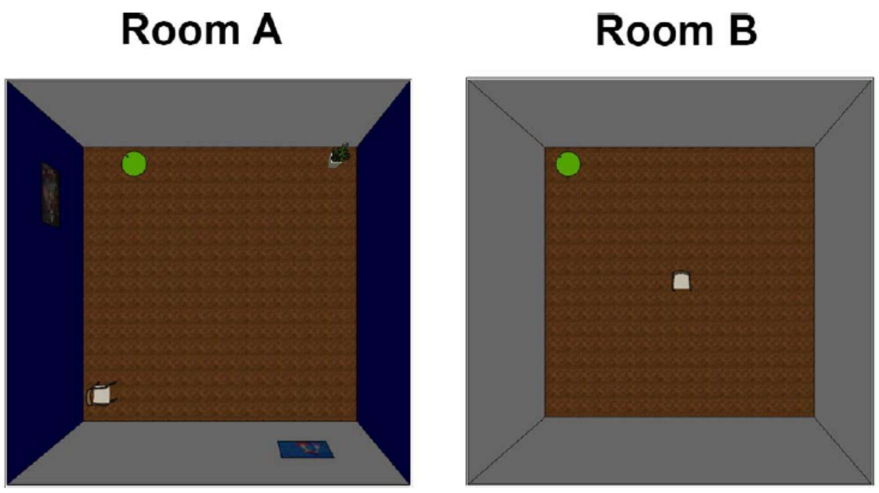


Figure 1: Two virtual rooms used in the VEHunt.

Driving Performance and Gaze Behavior. In [28], the driving performance and gaze behavior of glaucoma patients and a control group were assessed in a moving-based driving simulator for 40 minutes with nine different hazardous scenarios. The eye movements of the subjects were recorded using a mobile, head-mounted Dikablis eye-tracking system at 25 Hz while the head position and orientation were monitored via laserBIRD headtracker. Fitness to drive, lane position, time to line crossing, and speed of the subjects were analyzed. The study concluded that significant number of subjects with binocular glacoumatous visual field loss were safe drivers. Glaucoma patients who failed the driving test showed gaze bias towards the right side to maintain a stable lane position. On the other hand, glaucoma patients who passed the driving test made more extensive head movements and longer saccades to increase visual exploration and to compensate for their visual field loss. Therefore, binocular visual field loss did not directly affect safety of driving [28].

Collision Avoidance. Papageorgiou et.al. studied the effect of homonymous visual field defects (HVFDs) on collision avoidance of dynamic obstacles at an intersection in a virtual environment. HVFDs or hemianopia refers to the visual field loss in both eyes at the same relative position and directly associated with functional impairment. Patients with HVFDs might have difficulty in reading, visual exploration and navigation; collide with people and objects. Results showed that extent hemianopia was weakly associated with collision avoidance tasks under VR conditions. The performance of some hemianopia patients was similar with the subjects with normal vision due to compensatory viewing behavior through increase in eye and head movements [29].

Another study on predicting successful collision avoidance in patients with HVFDs by detecting gaze patterns was conducted by the same research group. The patients were divided into two subgroups, "adequate" ($HVFD_A$) and "inadequate" ($HVFD_1$), by the median split method. The eye and head movements of the patients were tracked. Saccades, fixations, mean number of gaze shifts, scanpath length and the mean gaze eccentricity were assessed and compared between the subgroups and controls. Both HVFDs subgroups displayed increase in gaze patterns to their blind side. Fixation number, fixation duration, scanpath length, and numer of gaze shifts were similar between the ($HVFD_A$) and normal patients [30].

VirtualEye. Wroblewski et.al. developed a compact, head-mounted and eye-tracking perimeter called VirtualEye which was the equivalent of a 24-2 visual threshold in two modes. First, the manual mode wherein the patient's response were registered via mouse click. Second, the visual grasp mode where the eye-tracker senses the change in gaze direction for target acquisition. The device had binocular eye-tracker enclosed in a head-mounted visor and had a computer program designed for data collection and analysis. It was designed to emulate the standard Humphrey field analyzer (HFA II) testing. The initial results of the clinical tests showed the similarity of visual field measurements between the HFA II testing and VirtualEye in both manual and virtual grasp modes [17].

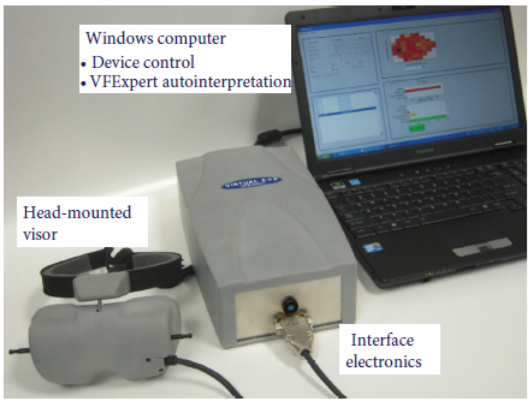


Figure 2: Components of VirtualEye perimeter.

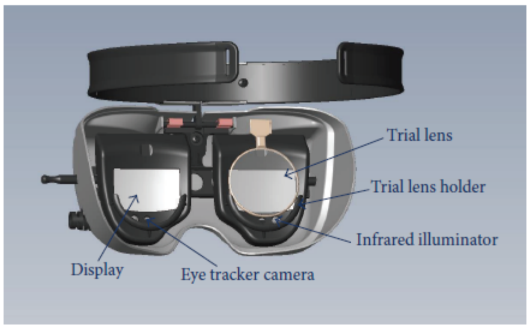


Figure 3: Back portion of VirtualEye head-mounted visor.

Vection Test. In [31], Tarita-Nistor et.al. investigated the responses in vection of patients with mild to moderate open-angle glaucoma. Glaucoma progressively affects the peripheral vision which is said to have a crucial role in inducing vection than the central vision. Vection is an illusion of self-motion in stationary observers when exposed to large moving stimuli. Fifteen patients with glaucoma together with fifteen age-matched patients were exposed to a random-dot pattern. The pattern was projected on a large screen and was rotated clockwise with an angular speed of 45° per second. Vection latency and vection duration were evaluated using a universal serial bus button-response box connected to a laptop computer. Vection latency was the time between the stimulus onset and participant's initial response; while vection duration was the total amount of time that self-motion was sensed. The objective and subjective measures of tilt were also assessed in three viewing conditions (binocular and monocular with each eye) wherein each condition lasted for two minutes. The subjective measure of tilt was the self-reported maximum tilt during vection; while the objective measure was the mean and maximum tilt angle recorded automatically via custom-made tilt sensor.The study found that glaucoma patients had longer vection latencies than, but had the same vection duration as, age-matched controls. Vections responses were not affected by the viewing conditions for each group. There was no relationship between the vection measures and the visual field sensitivity for the glaucoma patients.

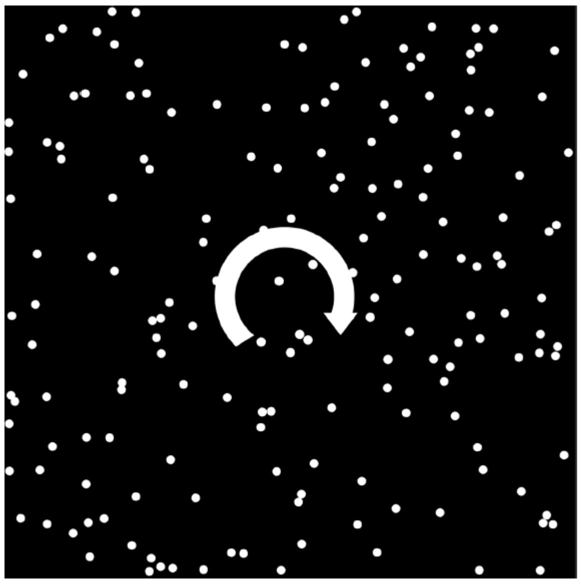


Figure 4: The pattern of the random-dot stimulus. The arrow indicates only the direction of the rotation and is not part of the stimulus.

Four-Session Driving Test. In [32], Vega et.al. examined the differences in driving performance, visual detection performance, and eye-scanning behavior of 23 patients with primary open-angle glaucoma and 12 control participants without glaucoma. The participants performed a driving test consisting of four sessions which lasted for 278 seconds each. They were instructed to maintain the driving cabin in the right lane of a two-lane highway without on- and of- ramps while speed was automatically maintained at 100 km/h. Additional tasks were also performed in the second, third, and fourth session. In the second session, the participants must detect and verbalize a 5 cm green letter appearing on the projected environment. Letters automatically disappeared after 4 seconds and the interval between the next letter varied from 3-6 seconds. Thirty-two letters were projected at 25 different preprogrammed coordinates on the screen. In the third session, the participants must avoid nine obstacles through changing lane to the left. The fourth session was a combination of the task performed in the second and third session in which the exact appearance (i.e. the time and location) of the letters and obstacles were also similar. The steering activity (measured in (s/deg) which is the average speed of the steering wheel angle), the number of missed letters, and the letter reaction time were significantly higher for glaucoma patients than the control participants. The glaucoma patients were more likely to miss a peripherally visualized stimuli. Eye-scanning recordings in this study may be inaccurate due to the system's inability to detect the pupil and the facial features of a participant.

Useful Field of View Test. The useful field of view, also known as UFOV, refers to the area of the visual space from which visual attention selectively processes information within a single fixation. Some factors affecting the UFOV are age, divided attention, task complexity, and training [33].

The UFOV Test, has been one of the most extensively researched and modified predictor tests for a range of driving outcomes measures, driving abilities, crash risks, and other everyday tasks. The UFOV test was developed to evaluate the visual difficulties that old adults experience in doing daily tasks. These difficulties are not captured by the conventional sensory visual tests such as visual acuity and visual field tests. For example, difficulty in identifying relevant information while visual clutter or distractions were present e.g. finding a specific item in a supermarket shelf. Another one was the need to divide visual attention rapidly while performing a specific task e.g. driving or walking across a busy road. The current commercially available version of UFOV test can be performed using a computer. The test run for about 15 minutes and was recommended as a screening measure in conjunction with a clinical examination of cognitive functioning or fitness to drive [24].

In the clinical visual field testing, it requires detection of threshold targets. On the other hand, UFOV test requires both the identification and localization of targets [23]. The UFOV test consisted of three subtests which measured the visual processing speed as the task demands increase in complexity. The user must detect, identify, and localize targets presented in varying time limits. In the first subtest, the user must be able to identify the the target in a centrally located fixation box. In the second subtest, the user must identify the central target and must also localize a simultaneously presented peripheral target. Lastly, the third subtest was almost similar to the second subtest but visual clutter was presented around the peripheral target [24].

For each subtest, UFOV test provided a score reported in milliseconds (ms)[24]. The test will measure the least stimulus presentation time that a user needs to identify and localize stimuli correctly. Shorter stimulus presentation time means better performance. The user must obtain atleast 75% correct response rate [22]. Aside from the visual processing speed, it also assessed higher cognitive abilities such as visual search, attention disengagement, and visual sensory function [24].

PERCEPT Visual Performance Test. Rosen et.al. proposed an iPad-enabled test called PERformance CEntered Portable Test (PERCEPT) to evaluate visual performance of glaucoma patients and to predict history of falls and motor vehicle crashes. The participants of the study were 71 patients with glaucomatous visual field defects based on standard automated perimetry (SAP) test and 59 control participants. The test was based on the concept of increasing difficulty of visual tasks to better the sensitivity for uncovering visual field defects in glaucoma. The PERCEPT test also measured visual processing speed via dual task at a low contrast (25%) Weber contrast). The central task was determining the orientation of a "tumbling letter E" (out of the 8 possible orientations) presented at the center within a limited time. The peripheral target was a vertically-oriented Gabor patch located at 7.7 degrees away from the central target and positioned in one of 8 various location meridians. The participants must initially focus their gaze at a small orange dot with a black rim in the center of the screen. Consequently, the central tumbling E and the peripheral Gabor path were presented simultaneously for the same time duration which is 1000 ms initially. Both positions of the central and peripheral target were independent of each other. After the target presentation, response screen with options of target orientations was shown. The participants must identify the orientation of the central target (tumbling letter E) and the peripheral target (Gabor patch) via touching the screen. For every two consecutive correct answers, the participants moved into the next difficulty level wherein the presentation time of the targets decrease. For every three consecutive incorrect answers, the test was terminated and the shortest presentation time a participant could correctly identify was recorded. The score used was called PERCEPT processing speed and was also measured in milliseconds.

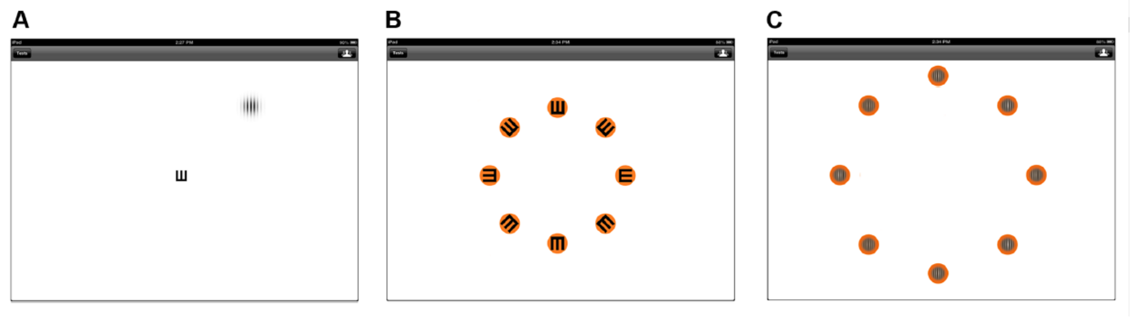


Figure 5: PERCEPT for visual assessment in glaucoma.

In this study, patients with glaucoma had higher mean PERCEPT processing speed values compared to the control group which are 493 ms and 133 ms, respectively ($P < 0.001$). It indicated that glaucoma patients perform slower at a dual visual task at a low contrast. The researchers also found out that from the 114 active drivers in 130 participants, 10 of them had motor vehicle collisions for the past few years. The performance of the patients with history of motor vehicle collisions had significantly higher PERCEPT processing speed values than others (664 ± 439 ms vs 91 ± 101 ms; $P = 0.003$).

III. Theoretical Framework

A. Glaucoma

Glaucoma is a progressive optic neuropathy caused by the deterioration of retinal ganglion cells. Retinal ganglion cells are neurons which cell bodies are contained in the inner retina and axons in the optic nerve. Loss of retinal ganglion cells worsen the visual fields which usually starts in the midperiphery and may develop towards the center until only the central or peripheral island of vision remains [2]. Glaucoma patients experience a ”tunnel vision” yet may have a normal central vision. Due to this, most patients may not recognize visual field loss until later stages of the disease when it has advanced into visual impairment [34]. The loss of retinal ganglion cells is associated with the intraocular pressure level (IOP). Intraocular pressure is determined by the balance between the secretion of aqueous humor by the ciliary body and its drainage through two main pathways (the trabecular meshwork and uveoscleral outflow pathway) [2].

Glaucoma can be classified into two categories, open-angle glaucoma and angle-closure glaucoma. Both are considered primary glaucoma and asymptomatic eye diseases wherein affected individuals are unaware of their condition until disease has advanced with significant neural damage. Secondary glaucoma can be a result of trauma; some medications such as corticosteroids, inflammation, tumor; or some conditions such as pigment dispersion or pseudo-exfoliation [2].

The primary open-angle glaucoma (POAG) is the most common type. In open-angle glaucoma, an increased resistance to aqueous outflow through the trabecular meshwork occurs. In primary closed-angle glaucoma (PCAG) is caused by the irregularities in the iris, lens, and behind the lens of the eye. The most common form of closed-angle is the pupillary block which caused resistance of the aqueous humor flow from the posterior to anterior chambers of the pupil. The fluid builds up behind the iris, increasing its convexity, which results to a closed angle. Biological factors might also result to closed-angle glaucoma such as increase in iris volume with pupil dilation and choroidal effusion. Acute primary angle closure is a very serious eye disease and requires immediate management to prevent blindness. The cornea is edematous due to a very high IOP and the pupil is often middilated and unreactive to light. Patients suffering from acute primary angle closure experience painful red eye, blurring of vision, headache, nausea, and vomiting [2].

Early diagnosis of glaucoma is challenging because there is no standard reference for establishing detection of glaucoma. It is only when 30% to 50% of retinal ganglion cells are lost, that visual field defects are identifiable by standard perimetry testing. Observation of the optic nerve head through an ophthalmoscope or obtaining optic nerve head photographs are done for longitudinal evaluation and documentation of structural damage to the optic nerve [2]. Some laser scanning imaging technologies such as confocal scanning laser ophthalmoloscopy, scanning laser polarimetry, and optical choerence tomography, have improved the identification of the eye disease; and have enhanced observation of optic nerve fiber (retinal ganglion cell axon) loss [2].

The purpose of treating glaucoma is to slow down the progression of the disease and preserve the quality of life. The only proven way to treat glaucoma is through reduction of the intraocular pressure. The first-line medical therapy of achieving the target intraocular pressure is through medication of drugs such as prostaglandin analogues. The drugs reduce the intraocular pressure through reduction of outflow resistance which increase aqueous humor flow through the uveoscleral pathway. The drugs are administered one every night and have few side effects. When all medical and laser treatment does not improve the IOP lowering and does not prevent progression of optic nerve and visual field damage, surgical treatment is recommended [2].

B. Visual Field Testing

Visual field is described as the portion of the external environment where visual information can be acquired when fixating on a particular point. The measurement of the visual field depends on the characteristics of the stimuli, the measurement conditions, and the response criteria of the patient [35].

Visual field tests are used by eye doctors and neurologists to test the health of a patient's retina, optic nerve, and visual pathway through the brain. The test is used to determine if a person is suffering from glaucoma, optic neuritis, or brain damage [12]. Several methods are available depending on the patients age, health, visual acuity, ability to concentrate, and socio-economic status [36].

B..1 Types of Perimetry

There are three types of visual field testing: (1) kinetic perimetry, (2) static perimetry, and (3) suprathreshold static perimetry [37]. Kinetic perimetry consists of detecting moving targets from the visual field of the non-seeing areas towards the seeing areas. The stimuli used have defined intensity and size [36]. Some examples of kinetic perimetry are tangent screening, Goldmann perimetry, and semi-automated Octopus 900 perimeter. Static perimetry involves detecting a stationary target that are presented in different areas of the visual field. The intensity of the stimuli varies depending on the subjects responses [35]. Static perimetry is more reliable and more consistent than kinetic perimetry in detecting glaucomatous visual field loss [36]. Some examples of static perimetry are the manual Goldmann perimeter, automated Humphrey Field Analyzer, automated Octopus 900 perimeter, and others.

Standard automated perimetry can be done using either threshold or suprathreshold analysis. In threshold analysis, the intensity is set on the level of detection or non-detection [36]. The measurement of visual field loss is performed using threshold analysis. In suprathreshold analysis, the intensities of the stimuli are set at a defined interval above the required threshold level. It is usually used for screening purposes, for checking the visual field status, and for patients who are inexperienced in using the standard visual field examination[35].

Based on several studies, visual field test using standard automated perimetry (SAP) is more reliable than other methods [36]. The first fully automated perimeter, developed by Dr. Frankhauser, was known as the Octopus perimeter. Several automated perimeter was also developed after Octopus. The currently available automated perimeters are the Fieldmaster, Humphrey Field Analyzer, Humphrey Matrix, Octopus, Easyfield (Oculus), and Medmount automated perimeters [13].

C. Virtual Reality

Virtual reality (VR) refers to a three-dimensional computer-generated environment, scenario, or activity that involves real-time simulation. It can represent various complex sensory information through different modes and can generate a substantial feeling or effect of realness despite its artificial nature [18].

There are various ways of gathering information from the participants depending on the VR systems. Some systems use joysticks, hand controls, motion tracking systems, instrument gloves, etc. The data gathered from these devices are utilized to control a computerized representation of the user moving and interacting in the virtual environment [18]. A good example is a user wearing head-mounted display with stereoscopic projection and viewing animated objects in a virtual environment [38].

Virtual reality is a technology that connects a user to an artificially generated environment. The key terms in VR are "presence" and "immersion". Presence or telepresence can be described as the "illusion of being there". Immersion is defined as the capability of the system to display a virtual environment close to a real experience [38].

Visual stimuli presented to users can be classified based on its immersion. Two-dimensional (2D) visual presentation are non-immersive. Three-dimensional (3D) presentations that utilize stereoscopic projections with fixed visual perspective are considered semi-immersive. Fully-immersive systems allows change of visual perspective with head movement [18]. The characteristics of a fully-immersive systems are real-time interaction, stereo vision, high frame rate and resolution, and multiple displays such as visual, auditory, and haptic [38].

C..1 Principles of Virtual Reality

The two main components of VR are (1) user environment and (2) virtual envionment. These two environments communicate and exchange information to and from a barrier called interface. The interface is considered the translator between the user and the VR system. Input actions (such as motion, force generation, or speech) from the user are being translated into digital signals via the interface. These digital signals can be processed and interpreted by the VR system. Consequently, the system's reactions are translated into physical quantities by the interface and perceived by the user through different display and actuator technologies (such as images or sounds). Lastly, the information is being interpreted by the user and the user will react to the system accordingly [38].

The exchange of information to and from the user and virtual environment occurs through various channels also known as modalities such as vision, sound, or touch. Multimodal interaction allows communication of the two environments simultaneously using several types of modalities. The goal of multimodal interaction is to provide a realistic scenario since people interact through multiple modalities in a real environment. It increases the quality of presence and the performance of the VR system [38].

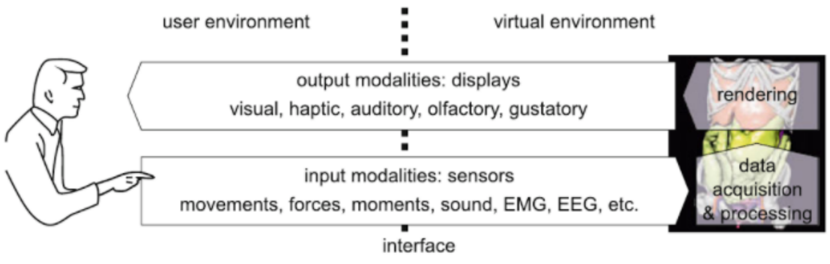

user environment
virtual environment
output modalities: displays
visual, haptic, auditory, olfactory, gustatory
input modalities: sensors
movements, forces, moments, sound, EMG, EEG, etc.
interface
rendering
data acquisition & processing

Figure 6: Multimodal, bi-directional interaction between the user environment and the virtual environment.

D. Mobile Application

Mobile applications consist of software or programs that perform certain tasks for the users. Mobile applications run on mobile devices or smartphones which usability depends on various factors such as screen resolution, hardware limitation, expensive data usage, connectivity issues and limited interaction possibilities. These applications can be used for communications, games, multimedia, productivity tools, travelling, and utilities [39].

D..1 Android

Android is an open-source mobile phone operating system (OS) released by Google with Linux-based platform. It is an essential platform to develop mobile applications using Google Android SDK [40].

D..2 Google Cardboard

Google Cardboard is described as a tool to view and interact in a virtual environment. It is the most widely used and accessible mobile VR platform [41]. It is a simple and inexpensive viewer made of foldable cardboard, 45 mm plastic lenses, and magnet or capacitive-taped lever for operating the screen. It is used by attaching a smartphone at the back of the viewer, running a Google Cardboard-compatible application, and viewing the app through the plastic lenses [42].

D..3 Google VR SDK

Google VR software development kit (SDK), together with Unity, is used to build virtual reality applications for Android and iOS. It consists of all requirements needed to develop platforms, including libraries, API documentation, developer samples, and design guideline [43]. It is integrated to Unity in order to utilize additional features such as spatialized audio, Daydream controller support, utilities, and samples. A minimum version of Unity 5.2.1 or later is required in using the native integration [44].

Figure 7: Google Cardboard. A simple VR viewer.

E. Unity

Unity is a cross-platform game development engine. It is most widely used in building 2D and 3D applications. The scripts are written using C# programming language and UnityScript and are attached to the unity objects. Games created in Unity can be deployed in desktop, consoles, mobile, augmented reality (AR), virtual reality (VR), and web. Unity has built-in support for Google Daydream, Gear VR, HTC Vive, Meta, Microsoft Hololens, Oculus Rift, Playstation VR, and any other VR headset [45].

F. Blender

Blender is a free and open-source 3D computer graphics software and runs on Linux, MacOS, and MS-Windows. It uses OpenGL as its interface to provide consistency across all supported hardware and platforms. It is used for creating 3D visualizations like images, videos, and real-time interactive video games. It supports the entirety of 3D pipeline modelling, rigging, animation, simulating, rendering, compositing, and motion tracking. It is also used in video editing and in creating game [46].

IV. Design and Implementation

A. Use Case Diagram

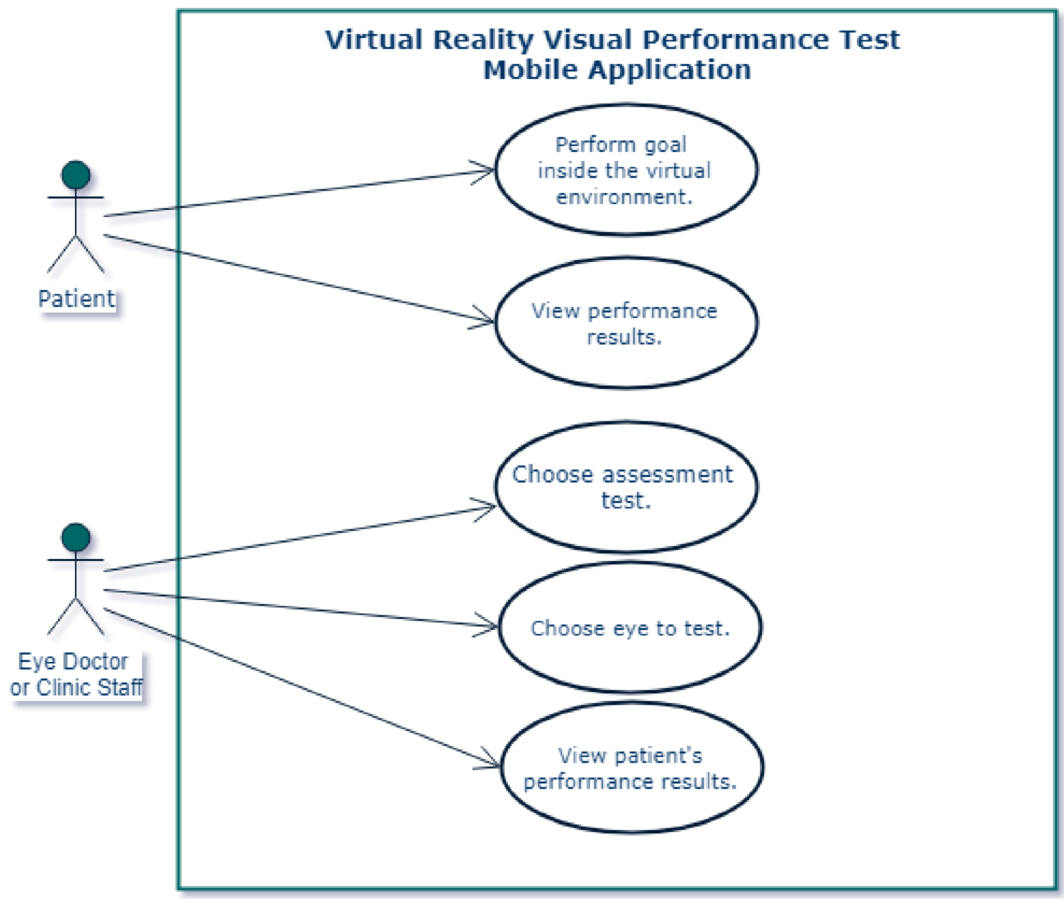

Virtual Reality Visual Performance Test
Mobile Application
Patient
Eye Doctor or Clinic Staff
Perform goal inside the virtual environment.
View performance results.
Choose assessment test.
Choose eye to test.
View patient's performance results.

Figure 8: Use Case Diagram

The mobile application has two users which are the patient and eye doctor or any clinic staff.

The patient can perform the objectives or goals set in the virtual environment. The patient can also view the performance results after each assessment test.

The eye doctor or any clinic staff can choose which assessment test should be performed and which eye should be evaluated in the application. The eye doctor or clinic staff can also view the visual performance results of each patient.

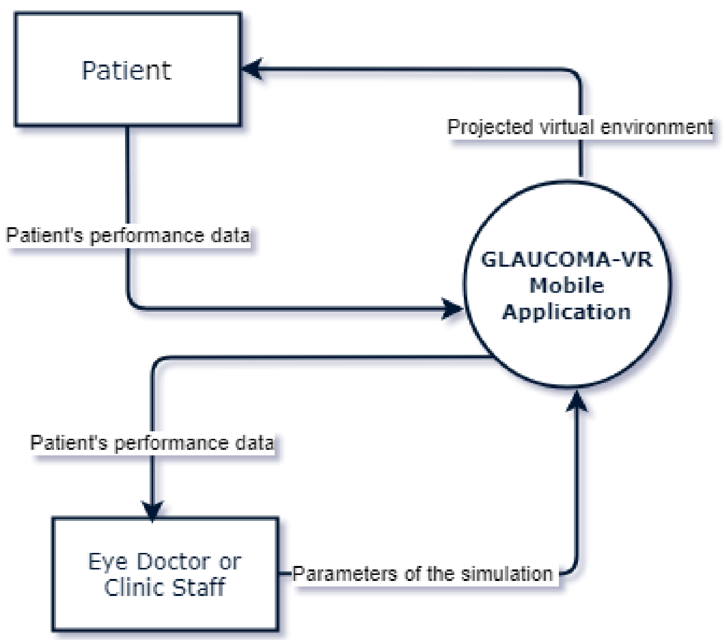

Patient
Projected virtual environment
Patient's performance data
GLAUCOMA-VR Mobile Application
Patient's performance data
Eye Doctor or Clinic Staff
Parameters of the simulation

Figure 9: Context Case Diagram

B. Context Case Diagram

In figure 9, there are two entities that can interact with the application, the patient and the eye doctor/clinic staff. The eye doctor/clinic staff chooses which assessment test to be performed and which eye to be evaluated. The application projects the virtual environment to the patient. The patient inputs performance results to the application which will be viewed by the eye doctor/clinic staff.

C. Activity Diagram

C..1 Eye Doctor or Any Clinic Staff

In figure 10, the eye doctor/clinic staff can choose the visual assessment test to be performed by the patient. The eye doctor/clinic staff can also choose which eye to test. The eye doctor/clinic staff can view the visual performance results of the patient.

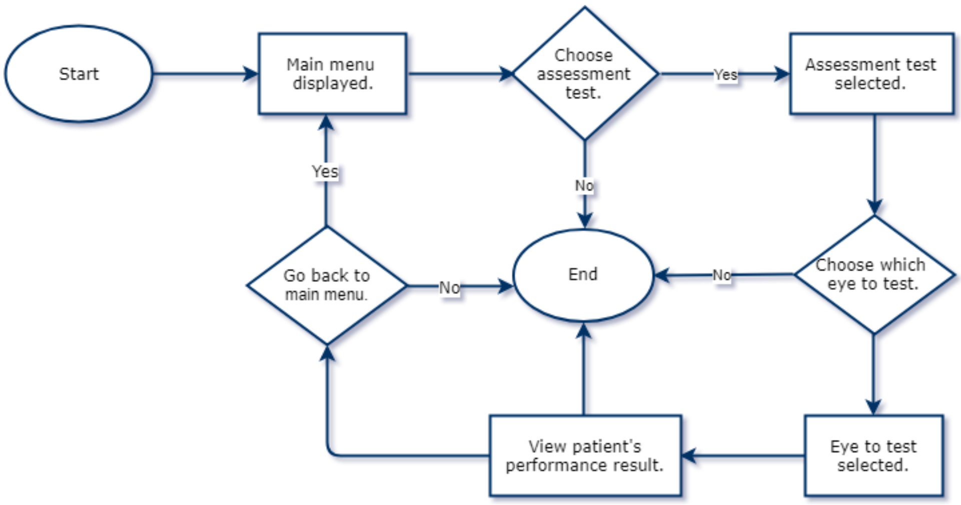


Figure 10: Activity diagram of the eye doctor or any clinic staff.

C..2 Patient

In figure 11, the patient can perform the visual performance assessment tests. After conducting the assessment tests, the patient can also view the results.

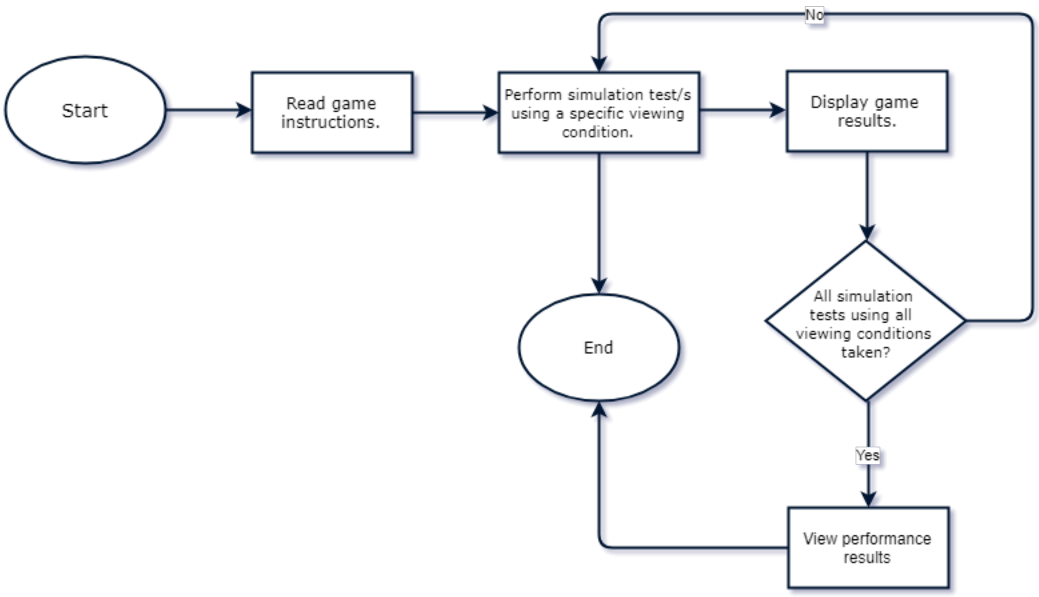

No
Start
Read game instructions.
Perform simulation test/s using a specific viewing condition.
Display game results.
End
All simulation tests using all viewing conditions taken?
Yes
View performance results

Figure 11: Activity diagram of the patient.

D. Technical Architecture

The smartphone device that will be used as head-mounted display requires the following:

1. Atleast an Android version of 4.4 (Kitkat).
2. Accelerometer
3. Gyroscope
4. Magnetometer

V. Results

The GLAUCOMA VR is a mobile application consisting of two virtual environments which are the convenience store and forest. These virtual environments produces a performance output that targets the visual processing speed of a patient.

A. Main Menu

The main menu is displayed once the application has started.

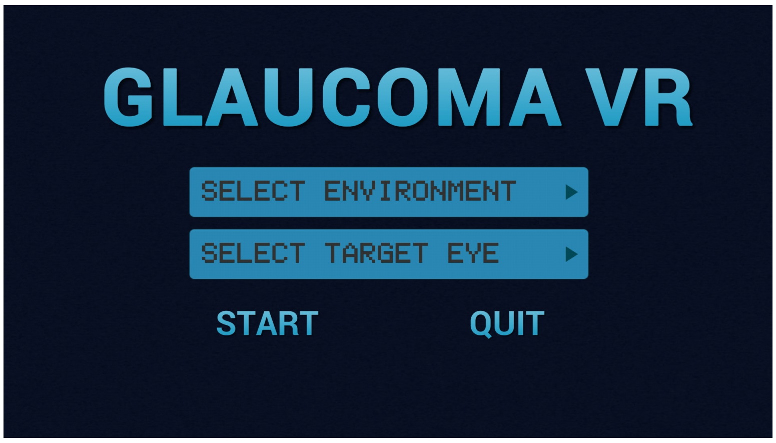


Figure 12: GLAUCOMA VR Main Menu

The clinic staff must input the required parameters, the virtual environment and the target eye. In figure 14, an error message appears when the clinic staff failed to do so.

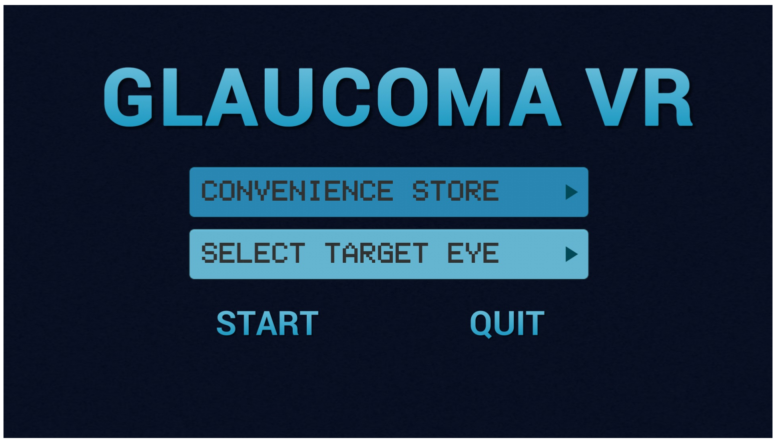
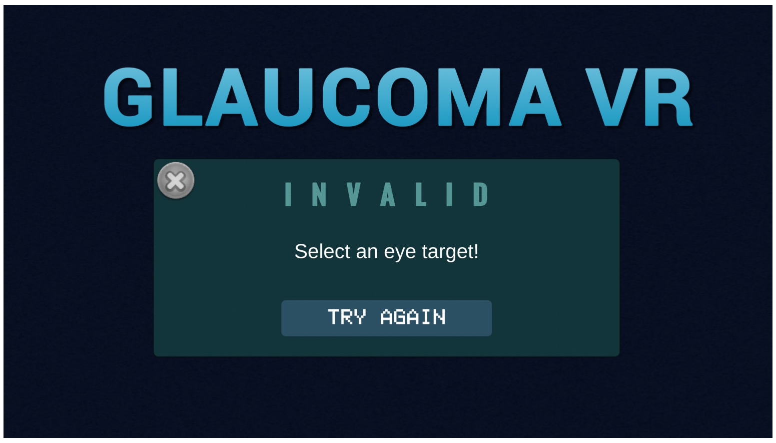


Figure 13: Invalid input. An eye target is not selected.



Select an eye target!

Figure 14: An error message displayed after an invalid input.

The clinic staff can press the start button to begin an assessment test when all necessary parameters are selected.

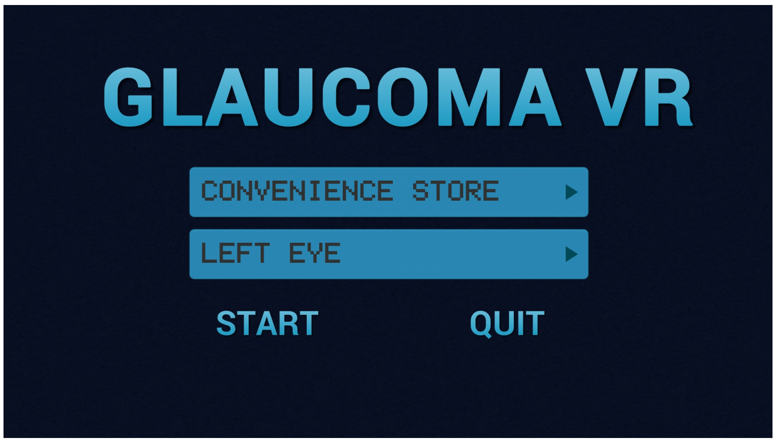


Figure 15: Valid parameters entered before starting the game.

B. Virtual Environment

When the start button is pressed, a progress bar appears which indicates that the environment is being loaded.

B..1 Convenience Store

Figure 16 represents the convenience store environment which is commonly known to most of the people as "7-Eleven". The patient should gaze at the start button for two seconds when ready to play.

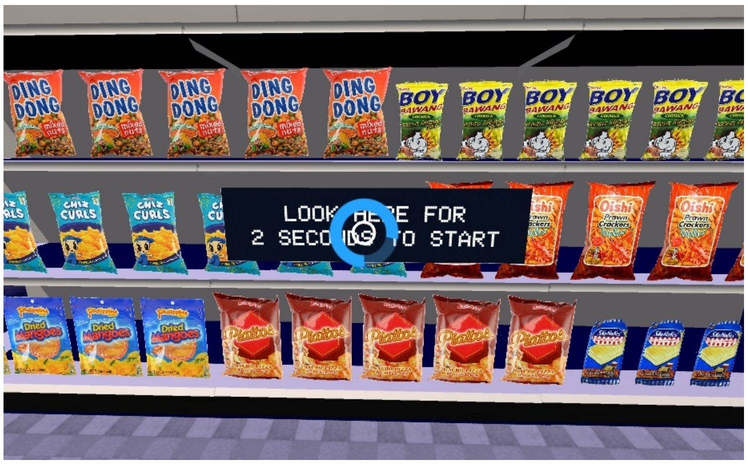


Figure 16: 7-Eleven scene before the game starts.

After selecting or gazing at the button, a light stimulus from an object appears. This indicates that the object is a target. The patient should gaze at this light stimulus for two seconds, then another light stimulus will appear from the periphery.

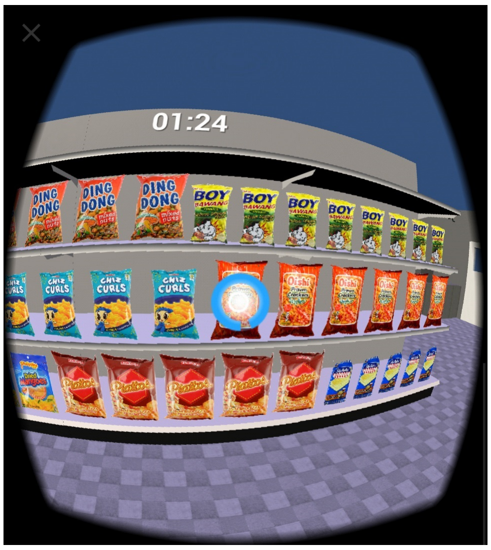


Figure 17: Target. A light component appearing from an object.

The lighting and selecting of targets occur for about three minutes. The game ends when the time is up. If the patient was able to select the target within five seconds, it would be counted as an additional point to the "perceived items". However, if the patient was not able to select the target while the countdown timer has not yet ended, it would be added to the "missed items".

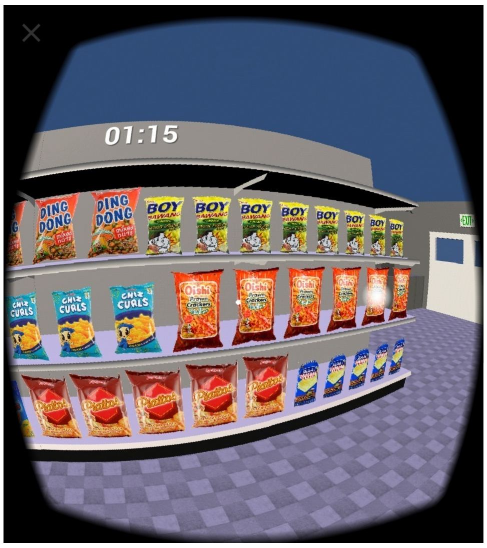


Figure 18: A light component appearing from another object after a specific object has been selected or gazed at.

B..2 Forest

This assessment test lasts for three minutes. In figure 19, the patient needs to gaze at the start button for two seconds when ready to play to begin the game.

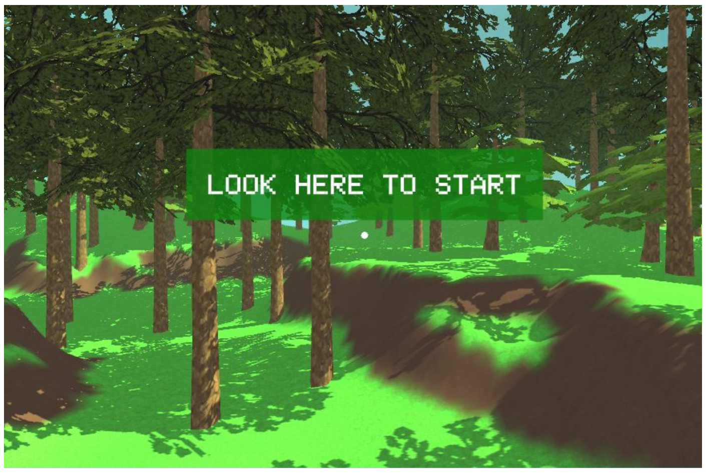


Figure 19: Forest scene before the game starts.

For about 1-2 seconds, a grid layout with a center and eight side panels will appear. A central and peripheral target appear on the screen simultaneously for three seconds, initially. In figure 20, the patient should focus on the object at the center while also recognizing the location of an object from the periphery.

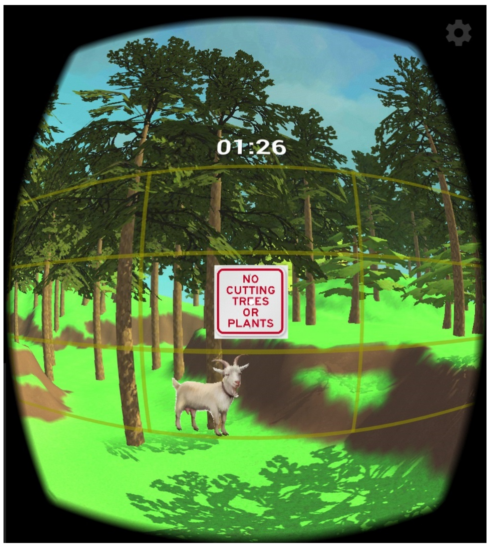


Figure 20: A central and peripheral target displayed simultaneously for a specific duration time.

Afterwards, the patient should correctly identify which central object is displayed and where the peripheral target is positioned. If both were correct, it would be counted on the "correctly identified items." If one of the answers is incorrect, it would be counted on the missed items.

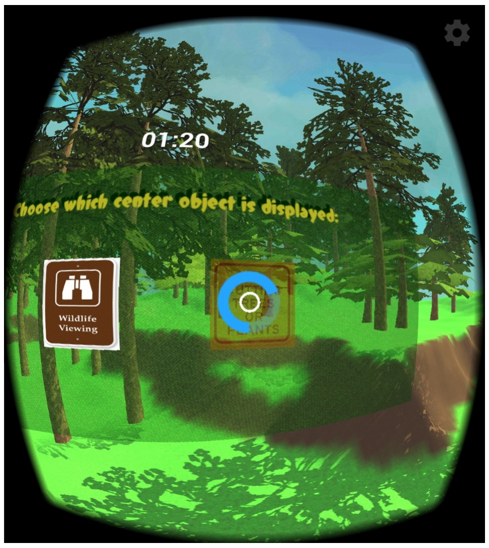


Figure 21: Identify the central object.

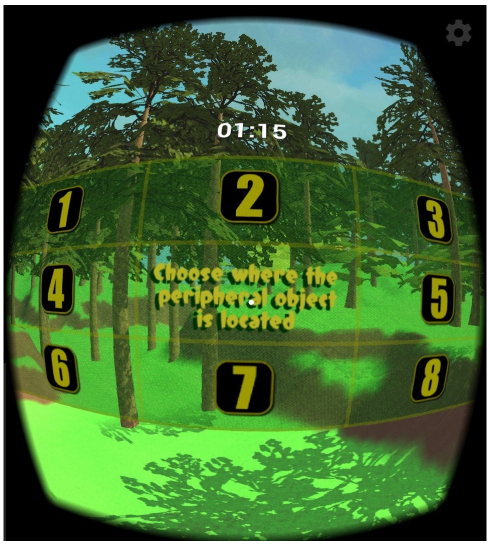


Figure 22: Localizing the peripheral target.

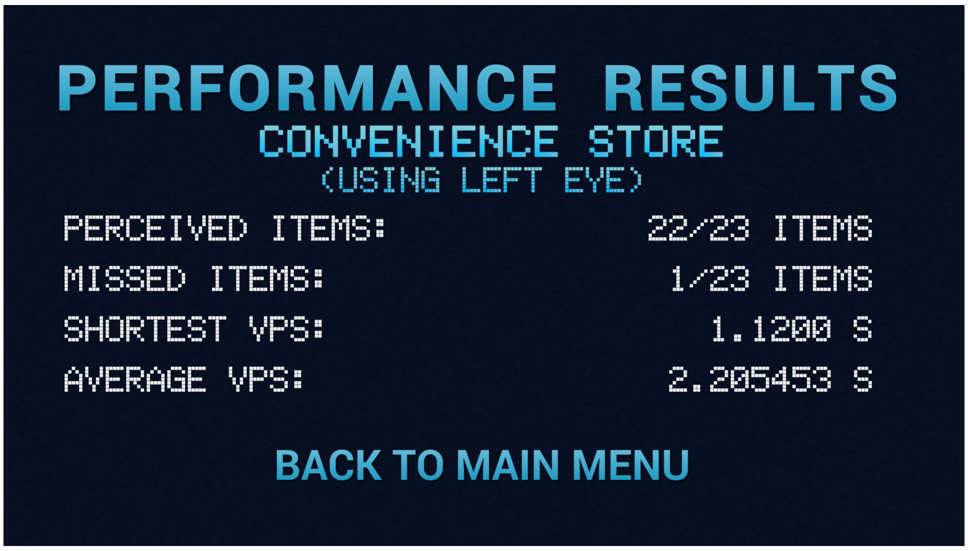
For every two consecutive correct answers, the presentation time of the central and peripheral objects decreases by 0.25s. On the contrary, the game ends when three consecutive errors is committed.

C. Performance Results

The performance results is displayed when the time is up for both scenes. Additionally in the forest scene, the results is displayed when the patient commits three consecutive incorrect answers.

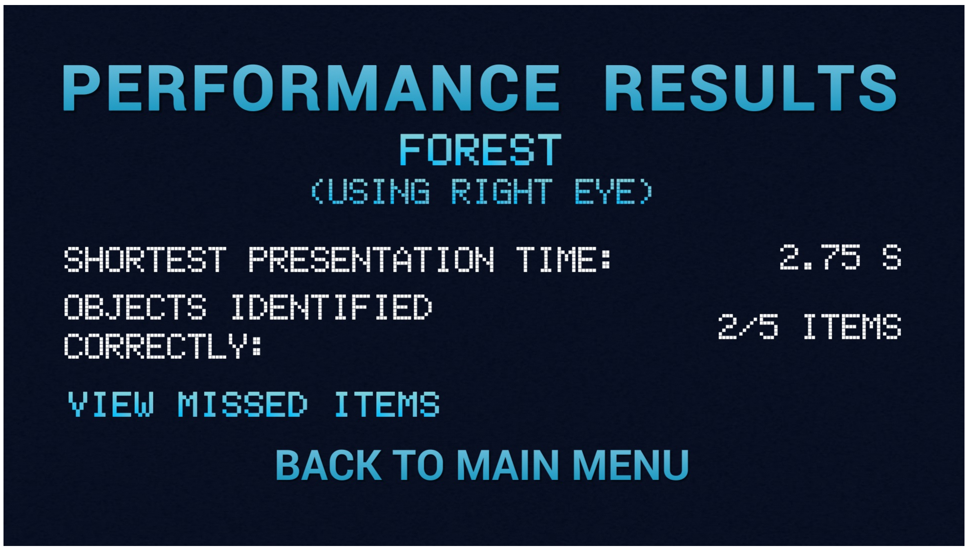
C..1 Convenience Store

In figure 23, the perceived items are the total number of items selected or gazed at by the patient. The missed items are items not selected five seconds after the light stimulus is shown. The VPS stands for "visual processing speed". In this assessment test, VPS is defined as the initial response time of the patient in seconds (s). The shortest VPS is the fastest initial response time of the patient. The average VPS is the mean of all initial response time record.


PERFORMANCE RESULTS
CONVENIENCE STORE
(USING LEFT EYE)

PERCEIVED ITEMS:	22/23 ITEMS
MISSED ITEMS:	1/23 ITEMS
SHORTEST VPS:	1.1200 S
AVERAGE VPS:	2.205453 S

Figure 23: Results for the convenience store scene.

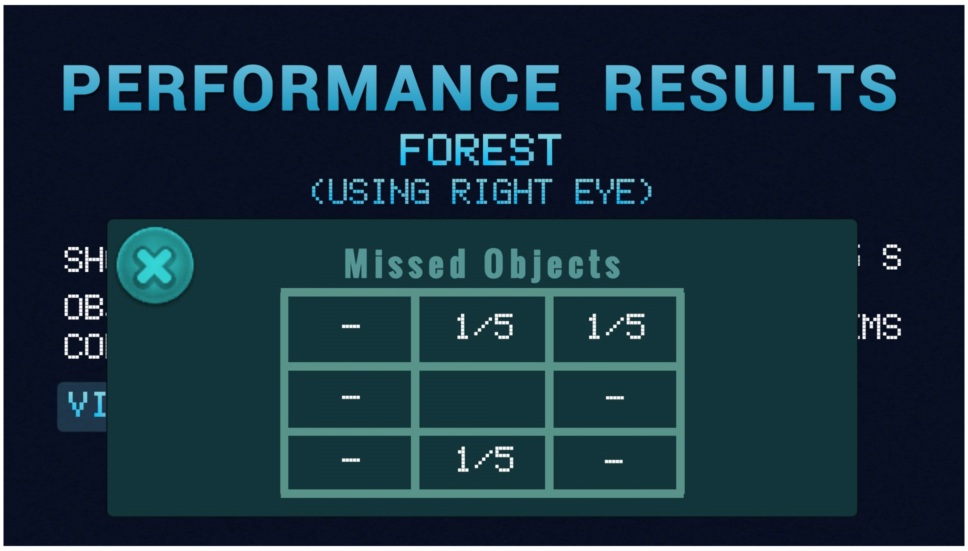

PERFORMANCE RESULTS
FOREST
(USING RIGHT EYE)

SHORTEST PRESENTATION TIME:	2.75 S
OBJECTS IDENTIFIED CORRECTLY:	2/5 ITEMS

Figure 24: Results for the forest scene.

C..2 Forest

In figure 24, the visual processing speed is defined as the minimal duration time that a patient can identify the central target and localize the peripheral target correctly. It is indicated in the "shortest presentation time." measured in seconds (s). When the patient identified the central object and located the peripheral object correctly, the point is added on the "objects identified correctly".



PERFORMANCE RESULTS

FOREST

(USING RIGHT EYE)

Missed Objects

–	1/5	1/5
–		–
–	1/5	–

Figure 25: Table showing the number of missed items per panel.

In figure 25, the number and location of missed peripheral targets, when the patient commits incorrect answers, is displayed. The patient should click the ”view missed items button” to display this.

VI. Discussions

Glaucoma VR is a mobile application designed to test the visual functional abilities of a person using virtual reality. It serves as a screening tool to indicate the possibility of a person to acquire glaucoma. The application is consisted of two virtual environments: convenience store and forest. The convenience store is a modified version of static visual field test with a different outcome measure which is the visual processing speed. The forest is based on the second subtest of the UFOV test which tests an individual's divided attention. The output of the resulting performance is generated after each assessment test.

The main program used in creating and accomplishing the virtual reality mobile application is Unity 5.6.5f1. Unity is a tool used to create 2D and 3D application using different platforms. For this application, the platform used is an Android smartphone with a minimum version of Anroid 4.4 or Kitkat. The assets and packages utilized are downloaded from the Unity asset store and other online websites that offer free 3D models such as Sketch Up 3D Warehouse[47]. The documentation and scripting API in the Unity website is an essential reference for coding scripts using C# language. After creating and building the application, an Android Package Kit (APK) file format is automatically outputted to be able to run the application on Android smartphones.

The application was able to fulfill the objectives. The eye doctor or any clinic staff can choose among the two virtual environments and which target eye to test. The main menu is displayed in a non-VR setting. It will be switched to VR setting after valid parameters are entered and start button is pressed. The scene is rendered in VR mode on either left or right target eye only. It is because the visual field defects are different for each eye. After rendering, the simulation of the assessment test to be performed by the user will be loaded. Instructions are provided in each virtual environment for the patient to follow. The GoogleVR SDK version 1.120.0 is used in the application to enable the VR mode. However, a problem was encountered during the deployment of the application on mobile.

During the switch from non-VR to VR mode, sometimes the scene will not load fully and only a black screen appears. The temporary solution to this issue is restarting the application until it functions properly.

The performance results is displayed when the time is up or when the patient is finished accomplishing the assessment test. Similar with the main menu, this is shown in a non-VR mode. The virtual scene, eye tested, parameter names, and scores are displayed. A button to go back to the main menu is present when the patient desires to conduct another assessment test.

Further clinical trials will be conducted to obtain a baseline measure for the visual processing speed of controls and patients with glaucoma using the given assessment tests. Currently, the baseline measurements of the processing speed for both subjects are still undetermined.

VII. Conclusions

Diagnosis of glaucoma is often delayed and SAP does not test the functional parameters related to vision. Although people with glaucoma can still accomplish tasks, they perform slower compared to people with normal vision. GLAUCOMA VR is designed as a screening tool to test visual processing speed and the risk of a person to acquire early glaucoma. The application does not intend to replace the conventional test for testing the visual field test or the perimetry. Both have different outcome measures. The SAP test outputs the visual field test points while this application tests the processing speed of a person's vision.

The application is developed for Android smartphones and used virtual reality to simulate two environments, the convenience store and forest. Both assessment tests should be performed by the patient for both eyes, separately. Conducting each test twice is advisable to obtain a reliable outcome measure. The application is created using Unity game engine. Virtual reality is made possible by using Google VR SDK for Unity version 1.120.0 and by using a VR headset, the Google Cardboard. The packages of various assets are downloaded from the Unity asset store. Some 3D models are also downloaded from the 3D Warehouse [47] (e.g. 7-Eleven environment).

GLAUCOMA VR is a mobile application that acts as an initial assessment to test the visual function of a person using virtual reality. Virtual reality allows the person to be immersed in any scenario. Unlike in two-dimensional screens, VR covers a person's entire field of view which limits external distractors that may affect a person's visual performance in an assessment test. Therefore, virtual reality may be a better platform for conducting assessment tests of a person's visual function.

VIII. Recommendations

The application can be improved by providing additional virtual environment or subtest (e.g. selective attention in UFOV) in testing the visual function of patients.

People suffering from glaucoma tend to increase saccades (eye movements) and head movements to compensate for their visual functional loss. However, Google Cardboard, the VR head-mounted device used in this application, does not support eye tracking yet. Using the gaze tracking technology, excessive eye movements of the user could be tracked. It could test whether the user focused on the central stimuli or the user recognized the peripheral stimulus in a single glance only. Thus, potential availability of gaze-tracking technology on VR headsets or smartphones could improve the results of each assessment test in the application. Excessive head movements should also be detected to verify if the user is focusing on a certain stimulus or not.

The results of the assessment test/s are displayed once after the user accomplished each test. However, results are not recorded in any file or database. These will eventually be removed when the "back to main menu" button is pressed or when the application is closed. It would be useful if data of the users for each test will be retrieved from a record. The performance outcome measures of each user can be compared and assessed after each retest particularly when the application will be used for training the visual function.

IX. Bibliography

[1] J. Berdahl, “Glaucoma: Types, symptoms, diagnosis and treatment. all about vision.” http://www.allaboutvision.com/conditions/glaucoma.htm, 2017. Accessed on 2017-06-19.

[2] M. Robert Weinreb, P. Tin Aung, MD, and P. Felipe Medeiros, MD, “The pathophysiology and treatment of glaucoma,” vol. 311, pp. 1901–1911, 2014.

[3] “Causes of blindness and visual impairment.” http://www.who.int/blindness/causes/en/, 2017. Accessed on 2017-06-19.

[4] B. H. Yih-Chung Tham, B. Xiang Li, P. Tien Y. Wong, FRCS, M. Harry A. Quigley, P. Tin Aung, FRCS (Ed), and P. Ching-Yu Cheng, MD, “Global prevalence of glaucoma and projections of glaucoma burden through 2040: A systematic review and meta-analysis,” *American Academy of Ophthalmology*, vol. 121, pp. 2081–2090, 11 2014.

[5] M. Jaymalin, “Over 2 million pinoys blind, sight-impaired.” https://www.philstar.com/headlines/2017/08/06/1726085/over-2-million-pinoys-blind-sight-impaired, 2017. Accessed on 2018-05-20.

[6] K. Boyd, “What is glaucoma?.” https://www.aao.org/eye-health/diseases/what-is-glaucoma, 2017. Accessed on 2017-06-19.

[7] “Facts about glaucoma.” https://nei.nih.gov/health/glaucoma/glaucoma_facts. Accessed on 2017-06-19.

[8] “Five common glaucoma test.” http://www.glaucoma.org/glaucoma/diagnostic-tests.php, 2013. Accessed on 2017-06-24.

[9] J. Martel, “Ophthalmoscopy: Purpose, procedure & risks.” http://www.healthline.com/health/ophthalmoscopy#purpose2, 2016. Accessed on 2017-06-29.

[10] "What is virtual reality." https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html, 2017. Accessed on 2017-06-19.

[11] C. Matsumoto, S. Yamao, H. Nomoto, S. Takada, S. Okuyama, S. Kimura, K. Yamanaka, M. Aihara, and Y. Shimomura, "Visual field testing with head-mounted perimeter 'imo'," vol. 11, 2016.

[12] "Virtual reality goggles create an equal opportunity eye test." https://www.sciencedaily.com/releases/2008/08/080808104929.htm, 2008. Accessed on 2017-06-29.

[13] C. A. Johnson, M. Wall, and H. S. ThompsonA, "A history of perimetry and visual field testing," vol. 88, no. 1, pp. E8–E15, 2011.

[14] "Ipad screenings effective for detecting early signs of glaucoma in underserved, high-risk populations." https://www.aao.org/newsroom/news-releases/detail/ipad-screenings-effective-detecting-early-signs-of-2, 2014. Accessed on 2017-08-21.

[15] "Free visual field analyzer eyescream eye test program as computer." http://www.eyesage.org/, 2012.

[16] S. Thakur and P. Ichhpujani, "Visual field assessment on the go," 2017.

[17] D. Wroblewski, B. A. Francis, A. Sadun, G. Vakili, and V. Chopra, "Testing of visual field with virtual reality goggles in manual and visual grasp modes," *BioMed Research International*, vol. 2014, March 2014.

[18] S. V. Adamovich, G. G. Fluet, E. Tunik, and A. S. Merians, "Sensorimotor training in virtual reality: A review," 2009.

[19] P. N. Rosen, E. R. Boer, C. P. B. Gracitelli, R. Y. Abe, A. Diniz-Filho, A. H. Marvasti, and F. A. Medeiros, "A portable platform for evaluation of visual performance in glaucoma patients," vol. 10, no. 10, 2015.

[20] E. Kasneci, A. A. Black, and J. M. Wood, “Eye-tracking as a tool to evaluate functional ability in everyday tasks in glaucoma,” 2017.

[21] “Pao policy statement vision requirements for driving.” http://www.pao.org.ph/PDF/circulars/PAO_POLICY_STATEMENT_Vision_%20Requirements_for_Driving.pdf.

[22] K. Woutersen, L. Guadron, A. V. van den Berg, F. N. Boonstra, T. Theelen, and J. Goossens, “A meta-analysis of perceptual and cognitive functions involved in useful-field-of-view test performance,” *Journal of Vision 2017*, vol. 17, p. 11, dec 2017.

[23] S. A. Bentley, R. P. LeBlanc, M. T. Nicolela, and B. C. Chauhan, “Validity, reliability, and repeatability of the useful field of view test in persons with normal vision and patients with glaucoma,” *Investigative Ophthalmology & Visual Science*, vol. 53, pp. 6763–6769, 2012.

[24] J. M. Wood and C. Owsley, “Useful field of view test,” *Gerontology*, pp. 315–318, 2014.

[25] B. Luo Luo Zheng, M. Lingmin He, MD, and M. Charles Qian Yu, “Mobile virtual reality for ophthalmic image display and diagnosis,” *Journal of Mobile Technology in Medicine*, vol. 4, pp. 35–38, 2015.

[26] R. Rajecki, “Gaming headset could offer a win for glaucoma patients.” http://ophthalmologytimes.modernmedicine.com/ophthalmologytimes/news/gaming-headset-could-offer-win-glaucoma-patients?page=0,1, 2015.

[27] F. B. Daga, E. Macagno, C. Stevenson, A. Elhosseiny, A. Diniz-Filho, E. R. Boer, J. Schulze, and F. A. Mcdeiros, “Wayfinding and glaucoma: a virtual reality experiment,” vol. 58, pp. 3343–3349, 2017.

[28] T. C. Kubler, E. Kasneci, W. Rosenstiel, M. Heister, K. Aehling, K. Nagel, U. Schiefer, and E. Papageorgious, "Driving with glaucoma: Task performance and gaze movements," *American Academy of Optometry*, vol. 92, pp. 1037–1046, 2015.

[29] E. Papageorgiou, G. Hardiess, H. Ackermann, H. Wiethoelter, K. Dietz, and H. A. Mallot, "Collision avoidance in persons with homonymous visual field defects under virtual reality conditions," vol. 52, pp. 20–30, 2011.

[30] E. Papageorgiou, G. Hardiess, H. A. Mallot, and U. Schiefer, "Gaze patterns predicting successful collision avoidance in patients with homonymous visual field defects," vol. 65, pp. 25–37, 2012.

[31] L. Tarita-Nistor, S. Hadavi, M. J. Steinbach, S. N. Markowitz, and E. G. Gonzales, "Vection in patients with glaucoma," *American Academy of Optometry*, vol. 91, no. 5, 2014.

[32] R. P. Vega, P. M. V. Leeuwen, E. R. Velez, H. G. Lemij, and J. C. F. de Winter, "Obstacle avoidance, visual detection performance, and eye-scanning behavior of glaucoma patients in a driving simulator: A preliminary study," 2013.

[33] R. V. Ringer, Z. Throneburg, A. P. Johnson, A. F. Kramer, and L. C. Loschky, "Impairing the useful field of view in natural scenes: Tunnel vision versus general interference," *Journal of Vision 2016*, vol. 16, 2016.

[34] S. Thomas, W. Hodge, and M. Malvankar-Mehta, "The cost-effectiveness of teleglaucoma screening device," vol. 10, 2015.

[35] P. A. Sample, F. Dannheim, P. H. Artes, J. Dietzsch, D. Henson, C. A. Johnson, M. Ng, U. Schiefer, and M. Wall, "Imaging and perimetry society standards and guidelines," *American Academy of Optometry*, 2011.

[36] D. Broadway, "Visual field testing for glaucoma a practical guide," *Community Eye Health*, vol. 25, pp. 66–70, 2012. Accessed on 2017-06-24.

[37] C. A. Johnson, "Principles of visual field testing and perimetry." http://www.aaopt.org/principles-visual-field-testing-and-perimetry, 2013.

[38] R. Riener and M. Harders, *Virtual Reality in Medicine.* 2012.

[39] M. R. Islam, M. R. Islam, and T. A. Mazumder, "Mobile application and its global impact," vol. 10, no. 6, pp. 72–78, 2010.

[40] S. Holla and M. M. Katti, "Android-based mobile application development and its security," vol. 3, pp. 486–490, 2012.

[41] "Google cardboard." = https://vr.google.com/cardboard/. Accessed on 2017-08-21.

[42] J. Hildenbrand, "What is google cardboard?." https://www.androidcentral.com/what-google-cardboard, 2015. Accessed on 2017-08-21.

[43] "Welcome to vr at google." https://developers.google.com/vr/. Accessed on 2017-08-21.

[44] "Google vr sdk for unity." https://developers.google.com/vr/unity/. Accessed on: 2017-08-21.

[45] "Unity for vr and ar." https://unity3d.com/unity/features/multiplatform/vr-ar. Accessed on 2017-08-21.

[46] "Blender." https://www.blender.org. Accessed on 2017-08-21.

[47] "Sketch up 3d warehouse." https://3dwarehouse.sketchup.com/.

X. Appendix

A. Source Code

MainMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.SceneManagement;
using UnityEngine.VR;


public class MainMenu : MonoBehaviour {

    public TMP_Dropdown Dropdown_Environment;
    public TMP_Dropdown Dropdown_TargetEye;

    public static int sceneIndex = 0;
    private int eyeIndex;

    [ SerializeField ] private MessageAlert alert;

    [ SerializeField ] private GameObject loadingUI;
    [ SerializeField ] private Slider loadingProgressBar;
    [ SerializeField ] private TMP_Text txt_loading;

    private const float LOADED_PERCENTAGE = 0.9f; //
          The percentage value when scene is fully loaded
    private AsyncOperation sceneAsyncLoad;

    // Function Listeners

    private void Awake()
    {
        VRSettings.enabled = false;
    }

    // =============== Triggered when Start button
           is pressed ===============
    public void StartListener()
    {
        sceneIndex = Dropdown_Environment.value;
        eyeIndex = Dropdown_TargetEye.value;

        if(sceneIndex != 0 && eyeIndex != 0)
        {
            Debug.Log("Chosen scene is " +
                   Dropdown_Environment.value);
            TargetEyeSettings.targetEye = eyeIndex;
            StartScene(sceneIndex);
        }
        else
        {
            ShowAlert();
        }


    }

    // ===============Calls the load scene functions
          ===============
    private void StartScene(int level)
    {
        loadingUI.SetActive(true);
        txt_loading.text = "LOADING...";
        //Debug.Log("Scene Level " + level + " is loading
              ...");
        //SceneManager.LoadScene(level);

        StartCoroutine(LoadAsyncScene(level));
    }

    // ===============Loads the scene
          asyncronously and loads the progress bar
          ===============
    IEnumerator LoadAsyncScene(int level)
    {
        yield return new WaitForSeconds(1.0f);
        sceneAsyncLoad = SceneManager.LoadSceneAsync(
              level);

        // Prevent auto load
        sceneAsyncLoad.allowSceneActivation = false;

        // Wait until the asynchronous scene fully loads
        while (!sceneAsyncLoad.isDone)
        {
            loadingProgressBar.value = sceneAsyncLoad.
                   progress;

            if(sceneAsyncLoad.progress >=
                   LOADED_PERCENTAGE)
            {
                loadingProgressBar.value = 1.0f;
                txt_loading.text = "READY...";

                yield return new WaitForSeconds(1.0f);

                sceneAsyncLoad.allowSceneActivation = true;
                //VRSettings.enabled = true;
            }

            Debug.Log(sceneAsyncLoad.progress);
            yield return null;
        }
    }

    // ===============Calls the pop-up window
          ===============
    private void ShowAlert()
    {
        if(sceneIndex == 0 && eyeIndex == 0)
        {
            alert.ShowMessage("Select an environment and
                   eye target!");
        }
        else if (sceneIndex == 0 && eyeIndex != 0)
        {
            alert.ShowMessage("Select an environment!");
        }
        else if (sceneIndex != 0 && eyeIndex == 0)
        {
            alert.ShowMessage("Select an eye target!");
        }

    }

    // =============== Close the application
          ===============
    public void ExitApplication()
    {
        Debug.Log("Quit button selected!");
        Application.Quit();
    }



}
```

MessageAlert.cs

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;

/**
 * Displays a warning message when the user pressed the
       start button with invalid parameters.
 **/
public class MessageAlert : MonoBehaviour {

    [ SerializeField ] private GameObject window;
    [ SerializeField ] private TMP_Text message;
    [ SerializeField ] private Button btn_Okay;

    //Shows the message alert
 public void ShowMessage(string msg)
    {
        message.text = msg;
        window.SetActive(true);
    }

    // Closes the message alert
    public void Hide()
    {
        window.SetActive(false);
    }
}
```

TargetEyeSettings.cs

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

/**
 * Sets the target eye where the scene will be rendered.
 **/
public class TargetEyeSettings : MonoBehaviour {

    public static int targetEye;

 void Awake () {

        Debug.Log("Target eye index is " + targetEye);

        VRSettings.enabled = true;
        if (targetEye == 1)
        {
            this.GetComponent<Camera>().stereoTargetEye
                 = StereoTargetEyeMask.Left;
        }
        else if (targetEye == 2)
        {
            this.GetComponent<Camera>().stereoTargetEye
                 = StereoTargetEyeMask.Right;
        }

    }

}
```

SceneTimer.cs

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

/**
 * Countdown timer for each scene that will last for 3
      minutes.
 **/
public class SceneTimer : MonoBehaviour {

    [SerializeField] private int maxTimeInSecs = 180;
    private int mins;
    private int secs;

    public TMP_Text txt_time;

    public void StartSceneTimer() {

        IEnumerator timer = GameTimer();
        StartCoroutine(timer);
    }

    private IEnumerator GameTimer()
    {
       int timeLeft = maxTimeInSecs;
       mins = Mathf.FloorToInt(timeLeft/60);
       secs = Mathf.FloorToInt(timeLeft % 60);

        txt_time.text = mins.ToString("00") + ":" + secs.
             ToString("00");

        while(timeLeft > 0)
        {
            mins = Mathf.FloorToInt(timeLeft / 60);
            secs = Mathf.FloorToInt(timeLeft % 60);

            while (secs >= 0)
            {
                txt_time.text = mins.ToString("00") + ":" +
                     secs.ToString("00");
                secs--;
                timeLeft--;
                yield return new WaitForSecondsRealtime(1.0
                     f);
            }

        }

        yield return null;

        txt_time.SetText("Time's up!");
        StopAllCoroutines();
        StartCoroutine(GameOver());
    }

    private static IEnumerator GameOver()
    {
        yield return new WaitForSeconds(2.0f);
        AsyncOperation sceneAsyncLoad = SceneManager.
             LoadSceneAsync("Performance_Output");

        // Wait until the asynchronous scene fully loads
        while (!sceneAsyncLoad.isDone)
        {
            yield return null;
        }
    }

}
```

SelectionRadial.cs

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

namespace VRStandardAssets.Utils
{
    /** This class is used to control a radial bar that fills
    * up as the user holds down the Fire1 button. When it
         has
    * finished filling it triggers an event. It also has a
    * coroutine which returns once the bar is filled.
    **/
    public class SelectionRadial : MonoBehaviour
    {
        public event Action OnSelectionComplete;

               // This event is triggered when the bar has
               filled.

        [SerializeField] private float m_SelectionDuration =
                2.0f;                             // How long
               it takes for the bar to fill.
        [SerializeField] private bool m_HideOnStart = true;
                                                  // Whether
               or not the bar should be visible at the start.
        [SerializeField] private Image m_Selection;
                                                          //
               Reference to the image who's fill amount is
               adjusted to display the bar.

        private Coroutine m_SelectionFillRoutine;

               // Used to start and stop the filling coroutine
                based on input.
        public bool m_IsSelectionRadialActive;

               // Whether or not the bar is currently useable.
        private bool m_RadialFilled;

               // Used to allow the coroutine to wait for the
               bar to fill.


        public float SelectionDuration { get { return
               m_SelectionDuration; } }

        private void Start()
        {
            // Setup the radial to have no fill at the start
                 and hide if necessary.
            m_Selection.fillAmount = 0f;

            if (m_HideOnStart)
                Hide();
        }


        public void Show()
        {
            m_Selection.gameObject.SetActive(true);
            m_IsSelectionRadialActive = true;
            HandleDown();
        }


        public void Hide()
        {
            // This effectively resets the radial for when it
                 's shown again.
            m_Selection.fillAmount = 0f;
            HandleUp();

            m_Selection.gameObject.SetActive(false);
            m_IsSelectionRadialActive = false;
        }

        private IEnumerator FillSelectionRadial()
        {
            // At the start of the coroutine, the bar is not
                 filled.
            m_RadialFilled = false;

            // Create a timer and reset the fill amount.
            float timer = 0f;
            m_Selection.fillAmount = 0f;

            // This loop is executed once per frame until the
                 timer exceeds the duration.
            while (timer < m_SelectionDuration)
            {
                // The image's fill amount requires a value
                     from 0 to 1 so we normalise the time.
                m_Selection.fillAmount = timer /
                     m_SelectionDuration;

                // Increase the timer by the time between
                     frames and wait for the next frame.
                timer += Time.deltaTime;
                yield return null;
            }

            // When the loop is finished set the fill amount
                 to be full.
            m_Selection.fillAmount = 1f;
            // Turn off the radial so it can only be used
                once.
            m_IsSelectionRadialActive = false;

            // The radial is now filled so the coroutine
                waiting for it can continue.
            m_RadialFilled = true;

            // If there is anything subscribed to
                OnSelectionComplete call it.
            if (OnSelectionComplete != null)
                OnSelectionComplete();
        }

        public IEnumerator WaitForSelectionRadialToFill()
        {
            // Set the radial to not filled in order to wait
                for it.
            m_RadialFilled = false;

            // Make sure the radial is visible and usable.
            Show();

            // Check every frame if the radial is filled.
            while (!m_RadialFilled)
            {
                yield return null;
            }

            // Once it's been used make the radial invisible.
            Hide();
        }


        private void HandleDown()
        {
            // If the radial is active start filling it.
            if (m_IsSelectionRadialActive)
            {
                m_SelectionFillRoutine = StartCoroutine(
                    FillSelectionRadial());
            }
        }


        private void HandleUp()
        {
            // If the radial is active stop filling it and
                reset it's amount.
            if (m_IsSelectionRadialActive)
            {
                if (m_SelectionFillRoutine != null)
                    StopCoroutine(m_SelectionFillRoutine);

                m_Selection.fillAmount = 0f;
            }
        }
    }
}
```

ResultsManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.VR;
using UnityEngine.SceneManagement;

/**
 * Outputs results after every scene or assessment test.
 **/
public class ResultsManager : MonoBehaviour
{
    public Canvas CSPerformanceCanvas;
    public Canvas ForestPerformanceCanvas;

    private int index;
    private TMP_Text targetEye;

    // 7-11 Scores
    public TMP_Text score_PerceivedItems;
    public TMP_Text score_MissedItems;
    public TMP_Text score_ShortestVPS;
    public TMP_Text score_AverageVPS;

    // Forest Scores
    public TMP_Text score_ShortestPT;
    public TMP_Text score_CorrectAnswers;
    public TMP_Text[] score_MissedObjects;

    public void CSPerformanceCanvasOn()
    {
        CSPerformanceCanvas.enabled = true;
        ForestPerformanceCanvas.enabled = false;
    }

    public void ForestPerformanceCanvasOn()
    {
        ForestPerformanceCanvas.enabled = true;
        CSPerformanceCanvas.enabled = false;
    }

    public void BackToMainMenu()
    {
        Debug.Log("Back to Main Menu clicked!");
        SceneManager.LoadScene("Main_Menu");
    }

    private void DisplayCS_PerformanceResults()
    {
        float itemsPerceived = ScoreManager.itemsPerceived;
        float itemsMissed = ScoreManager.itemsMissed;
        float total = ScoreManager.TotalItems();
        float shortestVPS = ScoreManager.ShortestVPS();
        float averageVPS = ScoreManager.AverageVPS();

        score_PerceivedItems.SetText(itemsPerceived.ToString
            () + "/" + total.ToString() + " ITEMS");
        score_MissedItems.SetText(itemsMissed.ToString() +
            "/" + total.ToString() + " ITEMS");
        score_ShortestVPS.SetText(shortestVPS.ToString("F4
            ") + " S");
        score_AverageVPS.SetText(averageVPS.ToString() +
            " S");
    }

    private void DisplayForest_PerformanceResults()
    {
        float shortestPT = UFOV.ShortestPresentationTime
            ();
        float correctAnswers = UFOV.GetCorrectAnswers();
        float totalObjects = UFOV.NumOfObjectsDisplayed
            ();

        score_ShortestPT.SetText(shortestPT.ToString() + "
            S");
        score_CorrectAnswers.SetText(correctAnswers.
            ToString() + "/" + totalObjects.ToString() + "
             ITEMS");

        DisplayMissedObjects_GridResults(totalObjects);
    }

    public void DisplayMissedObjects_GridResults(float total)
    {
        Debug.Log("=======Visual results Checker
            ======= ");
        foreach (VisualResults resp in UFOV.responseList)
        {
            Debug.Log("Location: " + resp.locationValue +
                "\t Missed Items: " + resp.
                numOfItemsMissed);

        }

        int missed = 0;

        for (int i = 0; i < score_MissedObjects.Length; i
            ++)
        {
            missed = UFOV.responseList[i].
                numOfItemsMissed;

            if (missed != 0)
            {
                score_MissedObjects[i].SetText(missed.
                    ToString() + "/" + total.ToString());
            }
        }
    }

    private void Awake()
    {
        VRSettings.enabled = false;
        index = MainMenu.sceneIndex;
    }

    void Start()
    {
        Debug.Log("Scene index is " + index);

        if (index == 1)
        {
            targetEye = GameObject.Find("Txt_EyeTested_1
                ").GetComponent<TMP_Text>();
            CSPerformanceCanvasOn();
            DisplayCS_PerformanceResults();
        }
        else if (index == 2)
        {
            targetEye = GameObject.Find("Txt_EyeTested_2
                ").GetComponent<TMP_Text>();
            ForestPerformanceCanvasOn();
            DisplayForest_PerformanceResults();
        }
        if (TargetEyeSettings.targetEye == 1)
        {
            targetEye.text = "(Using left eye)";
            Debug.Log("Target eye string: " + targetEye);
        }
        else if (TargetEyeSettings.targetEye == 2)
        {
            targetEye.text = "(Using right eye)";
            Debug.Log("Target eye string: " + targetEye);
        }

    }
}
```

GazeManager.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using VRStandardAssets.Utils;

public class GazeManager : MonoBehaviour,
      IPointerEnterHandler, IPointerExitHandler
{

    // This controls when the selection is complete.
    [ SerializeField ] private SelectionRadial
          m_SelectionRadial;
    [ SerializeField ] private SelectionManager
          m_SelectionManager;
    [ SerializeField ] private ScoreManager m_ScoreManager;

    private bool m_GazeOver;
    private bool m_Completed;

    //Do this when the cursor enters the rect area of this
          selectable UI object.
    public void OnPointerEnter(PointerEventData eventData)
    {
        // Debug.Log("The cursor entered the selectable UI
              element.");

        m_SelectionRadial.Show();
        m_GazeOver = true;
        m_SelectionRadial.OnSelectionComplete +=
              HandleSelectionComplete;
    }

     public void OnPointerExit(PointerEventData eventData)
     {
         m_SelectionRadial.Hide();
         m_GazeOver = false;
         m_SelectionRadial.OnSelectionComplete −=
               HandleSelectionComplete;

        if (m_Completed)
        {
            m_SelectionManager.DisableObjComponents(
                  transform);
            m_Completed = false;
        }
     }

    // Use this for initialization
    void Start()
    {
        m_GazeOver = false;
        m_Completed = false;
        SetGazedAt(false);
    }

    public void SetGazedAt(bool gazedAt)
    {
        return;
    }

    private void HandleSelectionComplete()
    {
        // If the user is looking at the rendering of the
              scene when the radial's selection finishes,
 // activate the button.
        if (m_GazeOver)
        {
            Debug.Log("HandleSelectionCompleted for " +
                  transform.name);

             if (transform != m_SelectionManager.startingObj
                  )
            {
                m_ScoreManager.AddPerceivedItems();
            }

            m_Completed = true;
            m_SelectionManager.Controller();
        }
    }
}
```

SelectionManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using VRStandardAssets.Utils;

public class SelectionManager : MonoBehaviour
{
    [ SerializeField ] private SelectionRadial
          m_SelectionRadial;
    [ SerializeField ] private ScoreManager m_ScoreManager;

    public Camera mainCamera;

    public Transform startingObj;
    public Transform container;
    private Transform[] objectsList;
    private Transform currentSelectableObj;

    private IEnumerator timer;
    private bool objectIsSeen;
    private static float responseTime = 0f;

    public static List<ResponseRecord> responseList = new
           List<ResponseRecord>();


    private void Start()
    {
        objectsList = getFirstChildren(container);

        for (int i = 0; i < objectsList.Length; i++)
        {
            DisableObjComponents(objectsList[i]);
        }
    }


    public void StartSceneGame()
    {
        currentSelectableObj = startingObj;
        AddHaloEffect(startingObj);
    }

    // =============== Stores the first children of
          each tranform into a list ===============
    private Transform[] getFirstChildren(Transform parent)
    {
        Transform[] children = parent.
              GetComponentsInChildren<Transform>();
        Transform[] firstChildren = new Transform[parent.
              childCount];
        int index = 0;
        foreach (Transform child in children)
        {
            if (child.parent == parent)
            {
                firstChildren[index] = child;
                index++;
            }
        }
        return firstChildren;
    }


    // =============== Adds or removes a halo
          component ===============
    public void AddHaloEffect(Transform obj)
    {
        Behaviour halo = (Behaviour)obj.GetComponent("
              Halo");
        halo.enabled = true;
    }

    public void RemoveHaloEffect(Transform obj)
    {
        Behaviour halo = (Behaviour)obj.GetComponent("
              Halo");
        halo.enabled = false;
    }


    // =============== Sets an object as
          interactable and uninteractable
          ===============
    public void EnableObjComponents(Transform obj)
    {
        obj.GetComponent<GazeManager>().enabled = true;
        obj.GetComponent<EventTrigger>().enabled = true;
    }

    public void DisableObjComponents(Transform obj)
    {
        obj.GetComponent<GazeManager>().enabled = false;
        obj.GetComponent<EventTrigger>().enabled = false;
    }

    // =============== Generates new random
        object to select ===============
    public Transform RandomGameObj()
    {
        int objectIndex = Random.Range(0, objectsList.
            Length);
        Transform currentObj = objectsList[objectIndex];

        while (currentObj == currentSelectableObj || !
            IsFullyVisible(currentObj.gameObject))
        {
            if (! IsFullyVisible (currentObj.gameObject))
            {
                Debug.Log("\t\t" + currentObj.ToString() +
                    " is not visible on the camera FOV!");
            }
            currentObj = objectsList[(UnityEngine.Random.
                Range(0, objectsList.Length))];
        }

        Debug.Log("NEW TARGET IS " + currentObj.
            ToString());
        return currentObj;
    }

    // =============== Sets an object new object to
        interact with ===============
    public void SetObjectToSelect ()
    {
        currentSelectableObj = RandomGameObj();
        EnableObjComponents(currentSelectableObj);
        AddHaloEffect(currentSelectableObj);
    }

    // ========= Detects if all bounds of a target is
        fully visible on camera's FOV =========
    private bool IsFullyVisible (GameObject obj)
    {
        Plane[] planes = GeometryUtility.
            CalculateFrustumPlanes(mainCamera);

        Bounds bounds = obj.GetComponent<Renderer>().
            bounds;
        Vector3 size = bounds.size;
        Vector3 min = bounds.min;

        //Calculate the 8 points on the corners of the
            bounding box
        List<Vector3> boundsCorners = new List<Vector3
            >(8) {
            min,
            min + new Vector3(0, 0, size.z),
            min + new Vector3(size.x, 0, size.z),
            min + new Vector3(size.x, 0, 0),
        };
        for (int i = 0; i < 4; i++)
            boundsCorners.Add(boundsCorners[i] + size.y *
                Vector3.up);

        //Check each plane on every one of the 8 bounds'
            corners
        for (int p = 0; p < planes.Length; p++)
        {
            for (int i = 0; i < boundsCorners.Count; i++)
            {
                if (planes[p].GetSide(boundsCorners[i]) ==
                    false)
                    return false;
            }
        }
        return true;
    }

    public void Controller()
    {
        RemoveHaloEffect(currentSelectableObj);
        SetObjectToSelect();

        responseTime = 0.0f;
        timer = CountdownTimer();
        StartCoroutine(timer);
    }

    private IEnumerator CountdownTimer()
    {
        float secondsLeft = 5.0f;

        Coroutine response = StartCoroutine(ResponseTimer
            ());

        while (secondsLeft > 0.0f && !m_SelectionRadial.
            m_IsSelectionRadialActive)
        {
            Debug.Log("Seconds Left: " + secondsLeft);
            yield return new WaitForSeconds(1.0f);
            secondsLeft--;
        }
        StopCoroutine(response);

        objectIsSeen = m_SelectionRadial.
            m_IsSelectionRadialActive;

        if (objectIsSeen)
        {
            StopCoroutine(timer);

            Debug.Log("\t TIMER STOPPED! Seconds: " +
                responseTime);

            if (currentSelectableObj != startingObj)
            {
                responseList.Add(new ResponseRecord(
                    currentSelectableObj, responseTime));
            }

        }
        else if (!objectIsSeen)
        {
            // Add score to missed objects
            m_ScoreManager.AddMissedItems();
            Debug.Log("\t TIME'S UP TO SELECT: " +
                currentSelectableObj + " missed!");

            DisableObjComponents(currentSelectableObj);
            yield return new WaitForSeconds(0.5f);

            // Select another random object
            Controller();
        }
    }

    private IEnumerator ResponseTimer()
    {
        while (true)
        {
            responseTime += Time.fixedDeltaTime;

            yield return null;
        }
    }
}
```

ResponseRecord.cs

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

// Stores the response time of the user in a list
public class ResponseRecord : IComparable <
    ResponseRecord>{

    public Transform selectedObject;
    public float responseTime;

    public ResponseRecord (Transform newSelectedObject,
        float newResponseTime)
    {
        selectedObject = newSelectedObject;
        responseTime = newResponseTime;
    }

    public int CompareTo(ResponseRecord other)
    {
        return this.responseTime.CompareTo(other.
            responseTime);
    }
}
```

ScoreManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using VRStandardAssets.Utils;

public class SelectionManager : MonoBehaviour
{
    [ SerializeField ] private SelectionRadial
        m_SelectionRadial;
    [ SerializeField ] private ScoreManager m_ScoreManager;

    public Camera mainCamera;

    public Transform startingObj;
    public Transform container;
    private Transform[] objectsList;
    private Transform currentSelectableObj;

    private IEnumerator timer;
    private bool objectIsSeen;
private static float responseTime = 0f;

public static List<ResponseRecord> responseList = new
        List<ResponseRecord>();


private void Start()
{
    objectsList = getFirstChildren(container);

    for (int i = 0; i < objectsList.Length; i++)
    {
        DisableObjComponents(objectsList[i]);
    }
}


public void StartSceneGame()
{
    currentSelectableObj = startingObj;
    AddHaloEffect(startingObj);
}

// =============== Stores the first children of
       each tranform into a list ===============
private Transform[] getFirstChildren(Transform parent)
{
    Transform[] children = parent.
           GetComponentsInChildren<Transform>();
    Transform[] firstChildren = new Transform[parent.
           childCount];
    int index = 0;
    foreach (Transform child in children)
    {
        if (child.parent == parent)
        {
            firstChildren [index] = child;
            index++;
        }
    }
    return firstChildren;
}


// =============== Adds or removes a halo
       component ===============
public void AddHaloEffect(Transform obj)
{
    Behaviour halo = (Behaviour)obj.GetComponent("
           Halo");
    halo.enabled = true;
}

public void RemoveHaloEffect(Transform obj)
{
    Behaviour halo = (Behaviour)obj.GetComponent("
           Halo");
    halo.enabled = false;
}


// =============== Sets an object as
       interactable and uninteractable
       ===============
public void EnableObjComponents(Transform obj)
{
    obj.GetComponent<GazeManager>().enabled = true;
    obj.GetComponent<EventTrigger>().enabled = true;
}

public void DisableObjComponents(Transform obj)
{
    obj.GetComponent<GazeManager>().enabled = false;
    obj.GetComponent<EventTrigger>().enabled = false;
}

// =============== Generates new random
       object to select ===============
public Transform RandomGameObj()
{
    int objectIndex = Random.Range(0, objectsList.
           Length);
    Transform currentObj = objectsList[objectIndex];

    while (currentObj == currentSelectableObj || !
           IsFullyVisible(currentObj.gameObject))
    {
        if (! IsFullyVisible (currentObj.gameObject))
        {
            Debug.Log("\t\t" + currentObj.ToString() +
                   " is not visible on the camera FOV!");
        }
        currentObj = objectsList[(UnityEngine.Random.
               Range(0, objectsList.Length))];
    }

    Debug.Log("NEW TARGET IS " + currentObj.
           ToString());
    return currentObj;
}

// =============== Sets an object new object to
       interact with ===============
public void SetObjectToSelect ()
{
    currentSelectableObj = RandomGameObj();
    EnableObjComponents(currentSelectableObj);
    AddHaloEffect(currentSelectableObj);
}

// ========= Detects if all bounds of a target is
       fully visible on camera's FOV =========
private bool IsFullyVisible (GameObject obj)
{
    Plane[] planes = GeometryUtility.
           CalculateFrustumPlanes(mainCamera);

    Bounds bounds = obj.GetComponent<Renderer>().
           bounds;
    Vector3 size = bounds.size;
    Vector3 min = bounds.min;

    //Calculate the 8 points on the corners of the
           bounding box
    List<Vector3> boundsCorners = new List<Vector3
           >(8) {
        min,
        min + new Vector3(0, 0, size.z),
        min + new Vector3(size.x, 0, size.z),
        min + new Vector3(size.x, 0, 0),
    };
    for (int i = 0; i < 4; i++)
        boundsCorners.Add(boundsCorners[i] + size.y *
               Vector3.up);

    //Check each plane on every one of the 8 bounds'
           corners
    for (int p = 0; p < planes.Length; p++)
    {
        for (int i = 0; i < boundsCorners.Count; i++)
        {
            if (planes[p].GetSide(boundsCorners[i]) ==
                   false)
                return false;
        }
    }
    return true;
}

public void Controller()
{
    RemoveHaloEffect(currentSelectableObj);
    SetObjectToSelect();

    responseTime = 0.0f;
    timer = CountdownTimer();
    StartCoroutine(timer);
}

private IEnumerator CountdownTimer()
{
    float secondsLeft = 5.0f;

    Coroutine response = StartCoroutine(ResponseTimer
           ());

    while (secondsLeft > 0.0f && !m_SelectionRadial.
           m_IsSelectionRadialActive)
    {
        Debug.Log("Seconds Left: " + secondsLeft);
        yield return new WaitForSeconds(1.0f);
        secondsLeft--;
    }
    StopCoroutine(response);

    objectIsSeen = m_SelectionRadial.
           m_IsSelectionRadialActive;

    if (objectIsSeen)
    {
        StopCoroutine(timer);

        Debug.Log("\t TIMER STOPPED! Seconds: " +
               responseTime);

        if (currentSelectableObj != startingObj)
        {
            responseList.Add(new ResponseRecord(
                   currentSelectableObj, responseTime));
        }

    }
    else if (!objectIsSeen)
    {
        // Add score to missed objects
        m_ScoreManager.AddMissedItems();
        Debug.Log("\t TIME'S UP TO SELECT: " +
               currentSelectableObj + " missed!");
            DisableObjComponents(currentSelectableObj);
            yield return new WaitForSeconds(0.5f);

            // Select another random object
            Controller();
        }
    }

    private IEnumerator ResponseTimer()
    {
        while (true)
        {
            responseTime += Time.fixedDeltaTime;

            yield return null;
        }
    }
}
```

UFOV.cs

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

/**
 * Controls the scores for the forest scene.
 **/
public class UFOV : MonoBehaviour {

    public Sprite Correct_CenterObj;
    public string Correct_PeripheralObj;
    public Sprite UserAnswer_CenterObj;
    public string UserAnswer_Location;

    public bool Menu_IsDisplayed;
    public bool Stimuli_IsDisplayed;

    // Score components
    private static int correctAnswers;
    private static int displayCounter;
    private int correctInARow;
    private int wrongInARow;
    [ SerializeField ] private GameObject messageBox;
    [ SerializeField ] private TMP_Text txt_Score;
    [ SerializeField ] private TMP_Text txt_Msg;

    // Timer components
    public static float presentationTime = 3.0f;
    private IEnumerator timer;
    private IEnumerator intervalTimer;

    [ SerializeField ] UIDisplayManager m_UIDisplayManager;

    public static List<VisualResults> responseList = new
           List<VisualResults>();


    private void Start()
    {
        Menu_IsDisplayed = false;
        Stimuli_IsDisplayed = false;

        correctAnswers = 0;
        displayCounter = 0;
        correctInARow = 0;
        wrongInARow = 0;

        presentationTime = 3.0f;

        int defaultValue = 0;

        for (int i = 1; i < 9; i++)
        {
            responseList.Add(new VisualResults(i,
                   defaultValue));
        }
    }

    public void StartSceneGame()
    {
        DisplayController();
    }

    // ========== Calls the IEnumerator for displaying
           objects using intervals ===========
    public void DisplayController()
    {
        Debug.Log("======== DISPLAY CONTROLLER
               ========");
        intervalTimer = IntervalDisplay();
        StartCoroutine(intervalTimer);
    }

    // ============ Check user's answers
           ============
    public void SelectionChecker()
    {
        Debug.Log("======== SELECTION CHECKER
               ========");

        displayCounter++;

        if ((Correct_CenterObj == UserAnswer_CenterObj)
              && (Correct_PeripheralObj ==
              UserAnswer_Location))
        {
            correctAnswers++;
            correctInARow++;
            wrongInARow = 0;

            if(correctInARow == 2)
            {
                DecreasePresentationTime();
                correctInARow = 0;

                txt_Score.SetText("Presentation Time: " +
                       presentationTime);
                txt_Msg.SetText("2 consecutive correct
                       answers!");
                messageBox.SetActive(true);

                DisplayController();
            }
            else
            {
                txt_Score.SetText("+1");
                txt_Msg.SetText("Both answers are correct!");
                messageBox.SetActive(true);

                DisplayController();
            }
        }
        else
        {
            wrongInARow++;
            correctInARow = 0;

            var resp = responseList.Find(x => x.
                   locationValue == m_UIDisplayManager.
                   location);
            int value = resp.numOfItemsMissed;
            resp.numOfItemsMissed = value + 1;

            Debug.Log("Location: " + resp.locationValue +
                   "\t Missed Items: " + resp.
                   numOfItemsMissed);

            if (wrongInARow == 3)
            {
                //Debug.Log(wrongInARow.ToString() + "
                       wrong answers in a row!");

                txt_Score.SetText("Game Over!");
                txt_Msg.SetText("3 consecutive mistakes!");
                messageBox.SetActive(true);

                StopAllCoroutines();
                StartCoroutine(GameOver());
            }
            else
            {
                txt_Score.SetText("−1");
                txt_Msg.SetText("Wrong answer!");
                messageBox.SetActive(true);

                DisplayController();
            }
        }
    }

    // ============ Random interval of object's
           display ============
    private IEnumerator IntervalDisplay()
    {
        if(messageBox.activeSelf)
        {
            yield return new WaitForSecondsRealtime(1.5f);
            messageBox.SetActive(false);
        }

        if (!Stimuli_IsDisplayed && !Menu_IsDisplayed)
        {
            float interval = Random.Range(1.0f, 2.0f);
            Debug.Log("Random time interval value is: " +
                   interval.ToString());
            yield return new WaitForSeconds(interval);
        }

        timer = CountdownPresentationTime();
        StartCoroutine(timer);

        StopCoroutine(intervalTimer);
    }

    // ============ Countdown of the presentation
        time of the objects ============
    public IEnumerator CountdownPresentationTime()
    {
        float timeLeft = presentationTime;
        m_UIDisplayManager.DisplayObjects();

        while (timeLeft > 0)
        {
            yield return new WaitForSecondsRealtime(0.25f);
            timeLeft -= 0.25f;
        }

        if (timeLeft == 0.0f || timeLeft < 1)
        {
            m_UIDisplayManager.HideObjects();
        }

        StopCoroutine(timer);
    }

    // ============ When user commits three
        consecuvtive incorrect answers ============
    private void DecreasePresentationTime()
    {
        if (presentationTime > 0f)
        {
            presentationTime -= 0.25f;
            Debug.Log("Current presentation time value: " +
                presentationTime);
        }
    }

    // ============ When user commits three
        consecuvtive incorrect answers ============
    private IEnumerator GameOver()
    {
        //StopAllCoroutines();
        yield return new WaitForSeconds(2.0f);
        AsyncOperation sceneAsyncLoad = SceneManager.
            LoadSceneAsync("Performance_Output");

        // Wait until the asynchronous scene fully loads
        while (!sceneAsyncLoad.isDone)
        {
            yield return null;
        }
    }

    public static float ShortestPresentationTime()
    {
        return presentationTime;
    }

    public static int GetCorrectAnswers()
    {
        return correctAnswers;
    }

    public static int NumOfObjectsDisplayed()
    {
        return displayCounter;
    }

}
```

UIDisplayManager.cs

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

public class UIDisplayManager: MonoBehaviour
{
    // UI Components for objects to display
    public GameObject DisplayContainer_Central;
    public GameObject DisplayContainer_Peripheral;

    public Image Img_CentralObj;
    public int location;

    // List of objects to display
    public Sprite[] Sprites_CentralObj;

    // UI Components for Selection Menu
    public Image[] Choices_CentralDisplay;
    public GameObject SelectionContainer_Menu;
    public Canvas Canvas_Selection;
    public GameObject SelectionPanel_CentralOptions;
    public GameObject SelectionPanel_PeripheralOptions;

    private IEnumerator controller;
    private int randomNum = 0;
    private int previousNum = 0;

    // Linked scripts
[ SerializeField ] private Camera UICam;
[ SerializeField ] private UFOV ufovScript;
[ SerializeField ] private SpawnPeripheralObj
    peripheralObjScript;

// ============ Selection Menu Controller
    ============
private IEnumerator MenuController()
{
    Debug.Log("======== MENU CONTROLLER
        COROUTINE ========");

    while (true) {
        if (SelectionMenuClick.CenterObjectIsSelected()
            && !SelectionMenuClick.LocationIsSelected
            ())
        {
            SelectionPanel_CentralOptions.SetActive(false
                );
            yield return new WaitForSecondsRealtime(0.5
                f);
            SelectionPanel_PeripheralOptions.SetActive(
                true);
        }

        if (SelectionMenuClick.CenterObjectIsSelected()
            && SelectionMenuClick.LocationIsSelected
            ())
        {
            SelectionContainer_Menu.SetActive(false);
            ufovScript.Menu_IsDisplayed = false;

            if (!ufovScript.Menu_IsDisplayed)
            {
                Debug.Log("Coroutine controller will stop
                    !");
                StopCoroutine(controller);

                SelectionMenuClick.ResetValues();
                ufovScript.SelectionChecker();

                yield return null;
            }
        }
        yield return null;
    }
}

//============ Display central and peripheral
    objects to detect ============
public void DisplayObjects()
{
    GenerateRandomSprites(Img_CentralObj);
    //location = Random.Range(1, 9);
    location = GenerateRandomNumber(1, 9);

    DisplayContainer_Central.SetActive(true);
    DisplayContainer_Peripheral.SetActive(true);
    peripheralObjScript.DisplayPeripheralObj(location);

    ufovScript.Stimuli_IsDisplayed = true;
    CorrectOptions();
}

// ============ Hide central and peripheral
    objects to detect ============
public void HideObjects()
{
    DisplayContainer_Central.SetActive(false);
    DisplayContainer_Peripheral.SetActive(false);
    peripheralObjScript.DestroyPeripheralObj();

    ufovScript.Stimuli_IsDisplayed = false;

    ShowSelectionMenu();
}

// ============ Select random image sprites
    ============
private void GenerateRandomSprites(Image img)
{
    int spriteRandomizer = Random.Range(0,
        Sprites_CentralObj.Length);
    //Debug.Log("Sprite randomizer value is: " +
        spriteRandomizer);

    while (Img_CentralObj.sprite == Sprites_CentralObj[
        spriteRandomizer])
    {
        Debug.Log("BOTH CHOICES ARE THE SAME
            .");
        spriteRandomizer = GenerateRandomNumber(0,
            Sprites_CentralObj.Length);
    }

    img.sprite = Sprites_CentralObj[spriteRandomizer];
}
```

```
    // ============Select random number without
          generating the previous selected
          ============
    private int GenerateRandomNumber(int start, int length)
    {
        randomNum = Random.Range(start, length);

        while (randomNum == previousNum)
        {
            randomNum = Random.Range(1, 9);
        }

        Debug.Log("Current num: " + randomNum + "\t
              Previous num: " + previousNum);

        previousNum = randomNum;

        return randomNum;
    }


    // ============ Store the correct options
          ============
    private void CorrectOptions()
    {
        ufovScript.Correct_CenterObj = Img_CentralObj.
              sprite;
        ufovScript.Correct_PeripheralObj = location.ToString
              ();
    }

    // ============Show menu that contains values to
           select ============
    private void ShowSelectionMenu()
    {
        controller = MenuController();
        StartCoroutine(controller);

        SelectionContainer_Menu.gameObject.SetActive(true);

        ufovScript.Menu_IsDisplayed = true;

        SelectionPanel_CentralOptions.SetActive(true);
        SelectionPanel_PeripheralOptions.SetActive(false);

        DisplayChoices();
    }

    private void DisplayChoices()
    {
        // Display the correct choice on this random image
              position
        int choiceRandomizer = Random.Range(0,
              Choices_CentralDisplay.Length);
        Choices_CentralDisplay[choiceRandomizer].sprite =
              Img_CentralObj.sprite;

        // Display the other choice on this random image
              position
        for (int i = 0; i < Choices_CentralDisplay.Length; i
              ++)
        {
            if (i != choiceRandomizer)
            {
                GenerateRandomSprites(
                      Choices_CentralDisplay[i]);
                break;
            }
        }
    }

}
```

SelectionMenuClick.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using VRStandardAssets.Utils;

/**
 * Controls the events for which central stimulus is
       displayed and where the peripheral stimulus is located
       .
 **/
public class SelectionMenuClick : MonoBehaviour,
       IPointerEnterHandler, IPointerExitHandler {

    private static bool centerIsSelected;
    private static bool locationIsSelected;

    private bool m_GazeOver;
    private bool m_Completed;

    [SerializeField] private UFOV ufovScript;
    [SerializeField] private SelectionRadial
          m_SelectionRadial;

    public void OnPointerEnter(PointerEventData eventData)
    {
        m_SelectionRadial.Show();
        m_GazeOver = true;
        m_SelectionRadial.OnSelectionComplete +=
              HandleSelectionComplete;
    }

    public void OnPointerExit(PointerEventData eventData)
    {
        m_SelectionRadial.Hide();
        m_GazeOver = false;
        m_SelectionRadial.OnSelectionComplete −=
              HandleSelectionComplete;
    }

    public void SetGazedAt(bool gazedAt)
    {
        return;
    }

    private void HandleSelectionComplete()
    {
        if (m_GazeOver)
        {
            Debug.Log("Transform gazed over!");
            m_Completed = true;
            SelectedValue();
        }
    }

    private void SelectedValue()
    {
        if (transform.tag == "Location")
        {
            Debug.Log("Button pressed is: " + transform.
                  name);
            Location(transform.name);
        }
        else if (transform.tag == "Center")
        {
            Debug.Log("Image pressed is: " + transform.name
                  );
            CenterObject(transform.GetComponent<Image
                  >());
        }
    }

    private void CenterObject(Image chosenImg)
    {
        centerIsSelected = true;
        ufovScript.UserAnswer_CenterObj = chosenImg.sprite
              ;
        Debug.Log("CLICKED: The chosen image is " +
              chosenImg.sprite.ToString());
        m_GazeOver = false;
        m_Completed = false;
    }

    private void Location(string locationValue)
    {
        locationIsSelected = true;
        ufovScript.UserAnswer_Location = locationValue;
        Debug.Log("CLICKED: The selected button is " +
              locationValue + ".");
        m_GazeOver = false;
        m_Completed = false;
    }

    public static bool CenterObjectIsSelected()
    {
        return centerIsSelected;
    }

    public static bool LocationIsSelected()
    {
        return locationIsSelected;
    }

    public static void ResetValues()
    {
        centerIsSelected = false;
        locationIsSelected = false;
    }

    void Start()
    {
        SetGazedAt(false);
        m_Completed = false;
    }
}
```

SpawnPeripheralObj.cs

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

```
/**
 * Selects random object to display at eight different
      locations.
 **/
public class SpawnPeripheralObj : MonoBehaviour {

    private int location;

    private GameObject peripheralObj;
    private GameObject instance_peripheralObj;
    private List<GameObject> List_Objects;

    [SerializeField] private Transform boxContainer;
    [SerializeField] private Camera peripheralObjCam;
    private Transform[] boundingBox;

    private int randomNum = 0;
    private int previousNum = 0;

    private void Start()
    {
        boundingBox = getFirstChildren(boxContainer);
    }

    // ============ Stores the first children of each
          tranform into a list ============
    private Transform[] getFirstChildren(Transform parent)
    {
        Transform[] children = parent.
              GetComponentsInChildren<Transform>();
        Transform[] firstChildren = new Transform[parent.
              childCount];
        int index = 0;
        foreach (Transform child in children)
        {
            if (child.parent == parent)
            {
                firstChildren[index] = child;
                index++;
            }
        }
        return firstChildren;
    }

    public void DisplayPeripheralObj(int locationValue)
    {
        location = locationValue;
        peripheralObj = SelectPeripheralObj();

        if (peripheralObj != null)
        {
            Debug.Log("peripheralObj is " + peripheralObj.
                  name.ToString());
            Debug.Log("Location value: " + location);

            BoxCollider b_collider = boundingBox[location -
                  1].GetComponent<BoxCollider>();
            Vector3 RandomPos = GetPointInCollider(
                  b_collider);

            instance_peripheralObj = Instantiate(
                  peripheralObj, RandomPos, Quaternion.
                  identity);
            instance_peripheralObj.transform.parent =
                  peripheralObjCam.transform;
        }
        else
        {
            Debug.Log("There's NO peripheral object!");
        }
    }

    // ============ Generates random peripheral
          game object ============
    private GameObject SelectPeripheralObj()
    {
        if (List_Objects != null)
        {
            List_Objects.Clear();
        }

        if (location == 1 || location == 2 || location == 3)
        {
            RandomObj("UpperObjects");
        }
        else if (location == 4 || location == 5)
        {
            RandomObj("MidObjects");
        }
        else if (location == 6 || location == 7 || location
              == 8)
        {
            RandomObj("LowerObjects");
        }

        return peripheralObj;
    }

    // ============ Loads random peripheral game
          object ============
    private void RandomObj(string path)
    {
        if (List_Objects != null)
        {
            List_Objects.Clear();
        }

        GameObject[] array_Objects = Resources.LoadAll<
              GameObject>(path);
        List_Objects = array_Objects.ToList();

        randomNum = Random.Range(0, List_Objects.Count);

        while (randomNum == previousNum)
        {
            randomNum = Random.Range(0, List_Objects.
                  Count);
        }

        previousNum = randomNum;

        peripheralObj = List_Objects[randomNum];
    }

    private Vector3 GetPointInCollider(BoxCollider box)
    {
        Bounds area = box.bounds;
        Vector3 center = area.center;

        Renderer p_renderer = peripheralObj.
              GetComponentInChildren<Renderer>();
        Bounds p_area = p_renderer.bounds;

        float x = Random.Range(center.x - area.extents.x +
              p_area.extents.x,
                              center.x + area.extents.x -
                                    p_area.extents.x);
        float y = center.y;
        float z = Random.Range(center.z,
            center.z + area.extents.z - p_area.extents.z);

        return new Vector3(x, y, z);
    }

    // ============ Destroys the instantiated
          peripheral game object ============
    public void DestroyPeripheralObj()
    {
        if (peripheralObj != null)
        {
            Destroy(instance_peripheralObj);
            Resources.UnloadUnusedAssets();
        }
    }
}
```

VisualResults.cs

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

/**
 * Stores the number of missed items per panel in a list.
 **/
public class VisualResults : IComparable<VisualResults> {

    public int locationValue;
    public int numOfItemsMissed { get; set; }

    public VisualResults(int location, int missedItems)
    {
        locationValue = location;
        numOfItemsMissed = missedItems;
    }

    public int CompareTo(VisualResults other)
    {
        return this.numOfItemsMissed.CompareTo(other.
              numOfItemsMissed);
    }
}
```

XI. Acknowledgement

Hindi ko alam kung paano sisimulang isulat ang seksyong ito. Higit sa lahat, hindi ko alam kung paano tatapusin ang pagsusulat ng Special Problem (SP) sa pamamagitan ng pasasalamat sa mga taong naging katuwang ko sa pagbuo nito sa anumang porma. Marahil ang sisimulan ko na lamang ito sa pagka-countdown mula sa bilang na sampu...

Sampu. Sampung semestre ang iginugol ko sa Unibersidad ng Pilipinas Maynila. Walong semestre para pag-aralan, iyakan, dasalan, at ipasa ang required na subjects para sa BS ComSci. Ang dalawang natitirang semestre ay upang mapagtuunan ng oras, luha, dugo, pawis, at atensyon ang SP o tinatawag naming "thesis".

Isang malaking pasasalamat sa Unibersidad ng Pilipinas Maynila at sa mga naging propesor ko sa limang taong pagbabahagi ng mga aral. Mga aral na hindi lamang natatago sa apat na sulok ng silid-aralan, kundi mga aral na magiging baon namin sa pagtahak ng malalaking hamon sa buhay. Salamat sa paghubog at pagmulat sa amin bilang isang mabuting estudyante at mamamayan.

Siyam...

Walo. Walong semestre kong naging propesor ang aking adviser na si Sir Marvin John Ignacio. Mula CMSC 10, CMSC 21, CMSC 126, CMSC 127, CMSC 131, CMSC 132, CMSC 191 ML, hanggang CMSC 161. Napagtanto kong propesor ko siya bawat semestre noong ako ay nasa 3rd year college na. Maraming salamat, Sir, sa pagbabahagi ng iyong kaalaman. Mula sa usapang acads, usapang hugot, hanggang sa mga pasanin sa buhay. Nakikipagkwentuhan at lumulubog sa kalagayan ng mga estudyante kung kaya't mas naiintindihan niya ang aming mga pinagdaraanan.

Napakagaling nga naman ng pagkakataon na si Sir pa ang naging lehitimo kong adviser para sa SP. Sa katunayan, "universal adviser" pa nga dahil nalalapitan ng lahat. Lagi naming uulit-ulitin na kayo ang "lodi" na adviser! Maraming maraming salamat, Sir, mula sa pagkupkop sa akin bilang advisee ninyo hanggang

sa paggabay sa mga dapat gawin sa SP. Salamat po sa pagbibigay sa amin ng iba pang oportunidad at sa paghihikayat sa amin na tapusin ang SP. Hindi ko po matatapos ito nang buo kung wala kayo. Iba ka talaga, Sir! Isa po kayo sa mga propesor ko sa UP na hinding-hindi ko malilimutan.

Pito...

Anim. Anim na semestre akong naging student assistant (SA) sa RH 108 *(na dating RH 114)* mula noong June 22, 2015 hanggang ngayon. Anim na semestre ko ring nakasama si Ma'am Eden Huelgas. Si Ma'am Eden ang nagsilbi kong pangalawang magulang sa UP Manila. Lagi akong kinakamusta at pinapaalalahanan na mag-aral nang mabuti. Minsan kahit naka-duty talaga ako, hindi niya muna ako uutusan kapag tambak ang inaaral kong exam o hinahabol na requirements sa mga oras na iyon. May pasalubong pa minsan kapag siya ay galing sa isang bakasyon o meeting. Maraming salamat po sa lahat. Ang swerte ko po na naging bahagi kayo ng UP life ko! Labyu, Ma'am!

Lima... Sa Duyog fam, FBCs, SP buddies & Team VR, blockmates at ilang mga kaibigan ko, maraming salamat!

Salamat sa Duyog fam! Kayo ang highlight ng 4th year, 2nd sem ko. Hinding-hindi ko malilimutan ang mga araw at gabing pagod tayo kaka-shoot para sa digi-pelikula habang pinagsasabay ang sandamakmak na acads. Salamat dahil hindi lang basta natapos sa DPSM Week ang samahan natin. Salamat sa mga panahong kayo ang kasama kong mag-overnight sa Coreon tuwing iyakan sa acads at walwalan sa Malate matapos ang 4th year! Special mention kay Ezekiel Romero na kasama ko madalas mag-acads sa Coreon noon at ka-chikahan sa Erra's! Salamat dahil kahit nagma-Masters na kayo ni Austin Dizon sa UP Diliman ay hindi niyo nakakalimutang mangamusta at i-motivate akong tapusin ang SP.

Salamat sa itinuturing kong pamilya sa UP, ang Freshman Block Coordinators (FBCs)! Naniniwala akong malaki ang ambag ng programa sa kung ano at sino ako ngayon. Laging nakatatak sa isip at puso ko ang mga katagang *"Always give your 100%!"* Dahil sa pamilyang ito, mas naging matatag at matibay akong tao.

Syempre, *"tatak sturdy"* eh. Ito ang pamilya ko sa UP na babalik-balikan ko at alam kong palagi akong yayakapin nang buo. Mahal ko kayo! Shoutout kanila Alee Carapas at Betina Feliciano dahil niyayaya nila akong mag-acads o thesis. Hindi ko malilimutan ang 3 days, 2 nights acads-thesis sesh namin sa Torre Lorenzo. Ang daming sabog pero masasayang ganap. Salamat din sa kanila dahil hinahayaan nila akong mag-stay sa condo nila kapag kailangan kong mag-overnight. Feel at home naman ako masyado, labyu both!

Hello nga pala sa SP buddies ko mula sa Block 12 Batch 2014! Hindi ko naramdamang wala na nga pala ang karamihan ng blockmates ko sa CAS dahil sa inyo. Sa apat na magbabarkada na sina Abegail Lopez, Faye Alano, Edward Lacanlale, at *Mareh* Naiza Asaad, salamat sa pagsama sa akin minsan sa SP sessions ninyo. Salamat kay Abe sa pangangamusta lagi sa SP ko at kay Faye sa pagpapaalala na matulog ako. Magiging memorable sa akin ang mga overnight natin sa coffee shops. Matira matibay talaga at palusugan ng eyebags ang peg! Higit na pasasalamat sa Team VR, kay Red Quito at Romeo Valdez! Salamat sa pagtulong sa akin kapag may hindi ako ma-gets na concept at kung paano gawin ito. Mami-miss ko ang mga SP sesh natin sa Tim Hortons lalo na sa Coreon Vito Cruz. Ang daming nagaganap eh? *Haha.* Mas masaya at magaan ang pakiramdam kapag kasama ko kayong gumagawa ng SP. Salamat sa pagbabahagi ng kaalaman at oras ninyo!

Salamat din sa blockmates ko dahil naging bahagi kayo ng apat (o higit pa) na taon ko dito sa UP. Unang-una, maraming salamat kay Jeffrey delos Reyes sa pag-push sa'kin na gawing SP topic ang Virtual Reality. Tinulungan niya pa ako sa pamamagitan ng pagbibigay ng references at materials para sa SP. Salamat sa tatlong babaitang kasama ko lagi sa bawat semestre at tuwing mga sem-ender, sina Micah Quisote, Jaya Pasia, at Danie Rodriguez! Salamat sa pagiging mabubuting kaibigan. Salamat kay Danie dahil bago ang araw ng defense, nagpinta pa siya ng sunflower at may mensahe na *"Goodluck! Thesis it. The last chapter."* Salamat sa laging pag-alala at paggawa ng art upang ma-motivate kami. Labyu, gurl!

Salamat kay Neil Generoso at Adrian Sabado dahil sila ang mga kasama ko the day before defense. Medyo nabawasan ang kaba ko. Nagpa-bff fries pa ang mga loko.

Apat. Apat na iba pang SP topic ang sinubukan kong buuin bago ang SP na ito. Dalawa dito ay nasimulan kong isulat mula Chapter 1 hanggang simula ng Chapter 3. Hindi naging madali ang paulit-ulit na pagbabasa ng journals at pagsusulat ng mga kaunting natutunan. Sa kabila ng pag-e-effort at pag-aaral ko ng mga ito, hindi pa rin naging sapat para matapos ang mga iyon. Ang apat na SP topics na ito ay naging parte ng 4th year college ko... na kumwestyon sa kakayahan at kapasidad ko bilang isang estudyante ng ComSci.

Nakakapanglumo. Nakakapanghina ng loob.

Gayunpaman, napagtanto kong mayroon at mayroong mararating ang hindi pagsuko. Mula noong 4th year hanggang sa huling taon ko, patuloy akong lumalaban.

Ngunit may mga pagkakataon na hindi ko kinayang lumaban nang mag-isa. Minsan kailangan natin ng mga taong hihila sa atin muli patayo...

Tatlo. Nais kong magpasalamat sa tatlong taong nariyan para sa akin sa mga panahong gusto ko nang sumuko. Ilang araw bago ang defense ay nawawalan na ako ng tiwala sa aking sarili ngunit salamat sa tatlong taong nagpaalala sa'kin na kakayanin ko pa at huwag na huwag akong bibitaw.

Kay Adrian Sabado, salamat fam! Siya ang *tatay ng block* namin. Solid 'to. Consistent. Mula first year hanggang sa huling taon ko sa UP, bahagi na siya ng support system ko. Salamat sa kanya dahil kahit *working dad* na siya simula nang maka-graduate, hindi niya pa rin nalilimutan na paalalahanan kami ni Neil na *"Rak lang fam. Tiwala lang!"* Mula sa maliit na coding exercises hanggang sa ilang malalaki kong problema sa buhay, lagi siyang nariyan para makinig at tumulong. *Tatay na tatay talaga.* Hindi ko malilimutan na day before defense, kahit na Sunday, ay nag-make time siya para matulungan ako at mag-mock defense ng SP. Salamat nang marami, fam! Solid ka!

Kay Micah Quisote, salamat teh! Alam kong ayaw mong dinadaldal ka sa social media lalo na kapag busy ka. Pero 'di ko malilimutan ang isang gabi na naglaan ka ng oras para sa isang heart-to-heart talk sa akin. Salamat sa pag-intindi at sa words of encouragement. Tatlong araw bago ang defense, kahit wala akong sinabi at nakita mo lang ako, alam mo nang may mali. Salamat din dahil sa mga oras na iyon, kahit 'di mo alam anong eksaktong nangyayari sa'kin, nariyan ka at tinabihan ko. Pinaalala muling ang linyang, "Alam kong kakayanin mo 'yan!" Salamat din dahil *one chat away* ka lang noong kailangan ko magpaprint ng docu. Salamat sa pagyakap at pagsuporta sa akin noong araw ng defense. *Alabyu alat!*

Kay Romeo Valdez, salamat din? *Hahaha joke lang 'yung question mark.* Sa kanya ko naibahagi kung bakit ako problemado tatlong araw bago mag-defense. Kahit madalas lang akong *jino-joke time* nito, maaasahan kong lagi siyang nakikinig sa mga hinaing ko. Salamat, sobra. Salamat dahil naitulak mo akong pabalik na ipagpatuloy ko lang ang nasimulan ko at 'wag mawalan ng pag-asa. Salamat din sa mga tips na ibinigay mo para ma-survive ko ang *roasting session*, aka defense. Salamat lalo sa pag-volunteer na bumili ng pagkain para sa panel, pag-photox ng docu, pagpapahiram ng laptop at kung ano-ano pa. Salamat sa iyo!

Dalawa. Dalawang beses akong sumablay.

Ang unang sablay ay napakasakit. Bumagsak. Nadapa. Nasugatan. Nag-extend ng isang taon sa UP. Ang hirap. Ang saklap. Gayunpaman, hindi dito magtatapos ang *sablay* ko.

Babagsak at madarapa pero babangon muli. Iiyak pero hindi susuko. Haharapin ang bawat hirap at pagsubok hanggang sa makamit ang ikalawa at tunay na tinatamasang *sablay*... ang sablay ng tagumpay!

Isa. Sa isang mahaba at mahirap na paglalakbay... sa wakas ay nakarating na rin sa ako rurok ng tagumpay.

Lubos ang aking pasasalamat sa aking pamilya sa pag-aalaga at pagsuporta sa akin palagi. Salamat lalo kay Ate Nene dahil nagpursigi siya upang mapag-aral ako. Salamat din kay Ate Analyn dahil kahit malayo siya ay pinaparamdam niyang nasa malapit lang siya. Salamat sa pangangamusta lagi sa thesis ko at pagpaalala na kakayanin ko itong tapusin. Mahal ko kayo, mga Ate! Salamat sa mga panlilibre ninyo sa akin ng sine o food trip para makapag-detox mula sa thesis. Salamat sa pag-iintindi palagi. Higit na pasasalamat para kay Mama. Lagi siyang nariyan mula noon hanggang ngayon para gabayan at alagaan kami. Salamat dahil tuwing magti-thesis ako sa madaling-araw, bibilhan niya pa ako ng pagkain sa McDo o Jollibee. Salamat dahil pinapayagan niya na akong mag-overnight para sa thesis. Salamat din dahil naniwala siya sa akin muli na kahit mahirap makalabas ng UP, ay magagawa ko ring makapagtapos. Hindi ko man palaging sinasabi ngunit mahal na mahal ko po kayo. Hindi lamang para sa akin ang tagumpay na ito kundi para sa inyo rin. Babawi ako sa inyo.

Last but not the least, thank you, Lord! Lagi kong tinatanong sa sarili ko kung paano ako nakakapasa sa ComSci subjects at paano ko nalalagpasan ang bawat semester. Alam kong hindi lang ito dahil sa akin o dahil sa ibang tao, kundi dahil na rin sa Inyo. Salamat dahil sa mga panahong hindi ko kayang ilabas sa kahit kanino ang mga nararamdaman ko, iiyak at idasal ko lamang ay gagaan na ang pakiramdam ko. Maraming salamat sa gabay at pagmamahal Ninyo.

Muli, maraming salamat sa inyong lahat. Higit sa pagtatapos ng SP at pagkamit ng Sablay, salamat sa mga pagmamahal, samahan, at mga aral na itinuro ninyo sa akin. Lagi't lagi ko itong dadalhin hanggang sa susunod na kabanata ng aking buhay.