2018

Using Google Cardboard to perform a visual field screening test

Dhanraj Selvaraj
Iowa State University

Using Google Cardboard to perform a visual field screening test

by

Dhanraj Selvaraj

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Greg Luecke, Major Professor
Judy Vance
Stephen Gilbert

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

DEDICATION

I dedicate this thesis to my parents, Dr. Eunice Selvaraj and Dr. Selvaraj Rajendran, and my sister, Suganthi Selvaraj for their constant support, encouragement, devotion to my dreams and ambition to become an engineer.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

NOMENCLATURE

VR	Virtual Reality
HFA	Humphrey Field Analyzer
FOV	Field Of View

ACKNOWLEDGMENTS

I would like to thank my Father, God Almighty for blessing me with this opportunity, and helping me with the process, by giving me strength, perseverance, wisdom and everything I needed to accomplish to be where I am today.

To Dr. Greg Luecke, my major professor, I am ever grateful to him for giving me the opportunity to work alongside him. I am thankful that he saw the potential in me and helped me identify my strengths and build on it. I have drawn so much inspiration through his patient, personable, and tolerable nature. I am also thankful for his guidance through all my undergraduate and graduate education. It has been a privilege to work under his mentorship.

To the members of my Program of Study (PoS) committee, Dr. Judy Vance and Dr. Stephen Gilbert, I express my gratitude for helping me successfully complete this work.

To my colleagues, who have been a source of inspiration and became good friends with me, Don Kieu, Dao Lim, Avinash Radhakrishnan, Shikhar Vats, Rishikumar Suresh Kumar, Ramanathan Ramu, Baskar Gopalakrishnan, Piranava Tamilselvan, Sayani Malty, Saptarshi Basu, and Vignesh, I thank you for the educational discourses, invaluable help, and encouragement throughout my graduate program. In addition, I would also like to thank my friends, the department faculty and staff for making my time at Iowa State University a wonderful experience.

ABSTRACT

The visual field test is used to detect areas on the retina where there is a loss of vision. The equipment used to conduct the test is bulky and can cost a significant amount to patients to take the test. Google Cardboard is an inexpensive headset which is paired with a mobile phone to run virtual reality applications. In this work, a visual field screening test is developed to enable people to do an eye exam with a low-cost and portable device such as a Google Cardboard and a smart phone. The Google Cardboard application helps reduce the cost of performing a visual field test by enabling a patient to do a self-administered visual field test before going into a clinic or hospital to do a more detailed eye exam. The patient can perform the test at home and with greater frequency, indications of advancing vision loss can be identified and treated earlier to prevent any irreversible damage to the eye caused by diseases.

There is an increase in demand for virtual reality products due to its affordability, portability, and accessibility. It is making a significant impact in many industries, including the healthcare industry. Virtual reality can assist with patient rehab, simulate surgeries to train doctors, treating PTSD [1]. In this project, a Google Cardboard application is developed to perform a visual field screening test. A set of lights arranged in a 7 x 7 grid is switched on and off randomly. The user presses the button on the Google Cardboard headset and data is recorded in the application as to which light the user sees or not. This data is plotted and is helpful in evaluating any defect in the visual field of the patient. This application of low-cost and portable VR headsets can significantly reduce time and cost for taking visual field tests because such a device is accessible even in remote areas of the world where bulky medical equipment is limited.

CHAPTER 1. INTRODUCTION

Virtual Reality Applications

Virtual reality (VR) is a computer-generated scenario that simulates a realistic user-experience. The scenario is displayed to the user by means of a VR headset and a screen to display the computer-generated scenario. A VR headset is a head-mounted device that a user wears on his/her head. The VR headset immerses the user into a virtual environment which will make the user go through an experience that will be similar to a real-world experience. VR headsets are an emerging technology as there is an increase in use of virtual reality applications in many different industries. It is prominently used in three main industries, namely gaming, sports, and healthcare industries. The second biggest market that VR has is in the healthcare sector. VR adds value to the healthcare sector because it creates inexpensive and portable solutions in training doctors and treating disorders like PTSD, Amblyopia, etc. As currently eye exams devices are expensive and usually not portable, a VR application can help solve the issue. I will briefly discuss about VR headsets and how VR is used in the healthcare industry.

Virtual Reality Headsets

The headset comprises of two main parts: a frame/body made of plastic or cardboard, a display, and a pair of wide-angled lenses. The lenses are the key components in creating the immersive experience. The lenses help create a 3-dimensional image of the content displayed on the screen. There are two types of VR headsets, one that comes with a built-in display and one that uses a smartphone as a display. The technology used in virtual reality headsets is developing rapidly. There is an increase in interest by large companies such as Microsoft,

Google, Facebook, and HTC who are engaging in development of VR headsets [2]. Some of the VR headsets that are most commonly used today are:

1. Oculus Rift
2. HTC Vive
3. Google Cardboard

Oculus Rift & HTC Vive

The Oculus Rift and the HTC Vive are VR headsets that have a built-in display. These headsets are used to develop advanced applications as they have higher processing capabilities and are required to be connected to an external desktop or laptop computer. The Oculus Rift or HTC Vive cost anywhere around $500 - $1000. Images of the Oculus Rift and HTC Vive are shown in Figure 1-1 & 1-2.

Figure 1-1 *Oculus Rift* [3]

Figure 1-2 *HTC Vive* [4]

Google Cardboard

The Google Cardboard shown in Figure 1-3 is a headset viewer that uses a cell phone screen as the display and the application is therefore dependent on the processing capabilities of the cell phone. Google Cardboard is fast, accessible, and costs only $15 [5]. Users can either build their own stereo viewer hardware from simple, low-cost components using specifications published by Google, or purchase a pre-manufactured viewer. To use the platform (i.e., viewer and cell phone), users run cardboard-compatible applications on their phone, place the phone into the back of the viewer, and view content through the lenses. The software divides the phone screen into 2 halves, and processes the images into the left-eye and right-eye perspectives to provide stereo vision.

Figure 1-3 *Google Cardboard* [6]

The key differences between the two types of headsets are the costs and portability. The Google Cardboard is paired with a smart phone while the headsets with built-in displays relies on the desktop or laptop computer to display the immersive virtual environment. The Google Cardboard and phone can be transported easily while the additional weight of the headset with a built-in display and computer can cause inconvenience when being transported. The cost of the Google Cardboard and a smart phone is significantly lower when compared to the cost of a VR headset with a built-in display and a computer put together. Because of the affordability, portability, and ability for rapid application development of these VR headsets, they are widely used in the healthcare industry in order to assist with patient rehab, simulate surgeries to train doctors, treating PTSD, etc. [1]. VR headsets are also used in treating amblyopia (i.e. lazy eye) as it helps in saving vision by controlling the images shown to each eye [7]. The technology requirements for my application is well met by the Google Cardboard, so I will use it to replicate a simple visual field screening test.

Visual Field

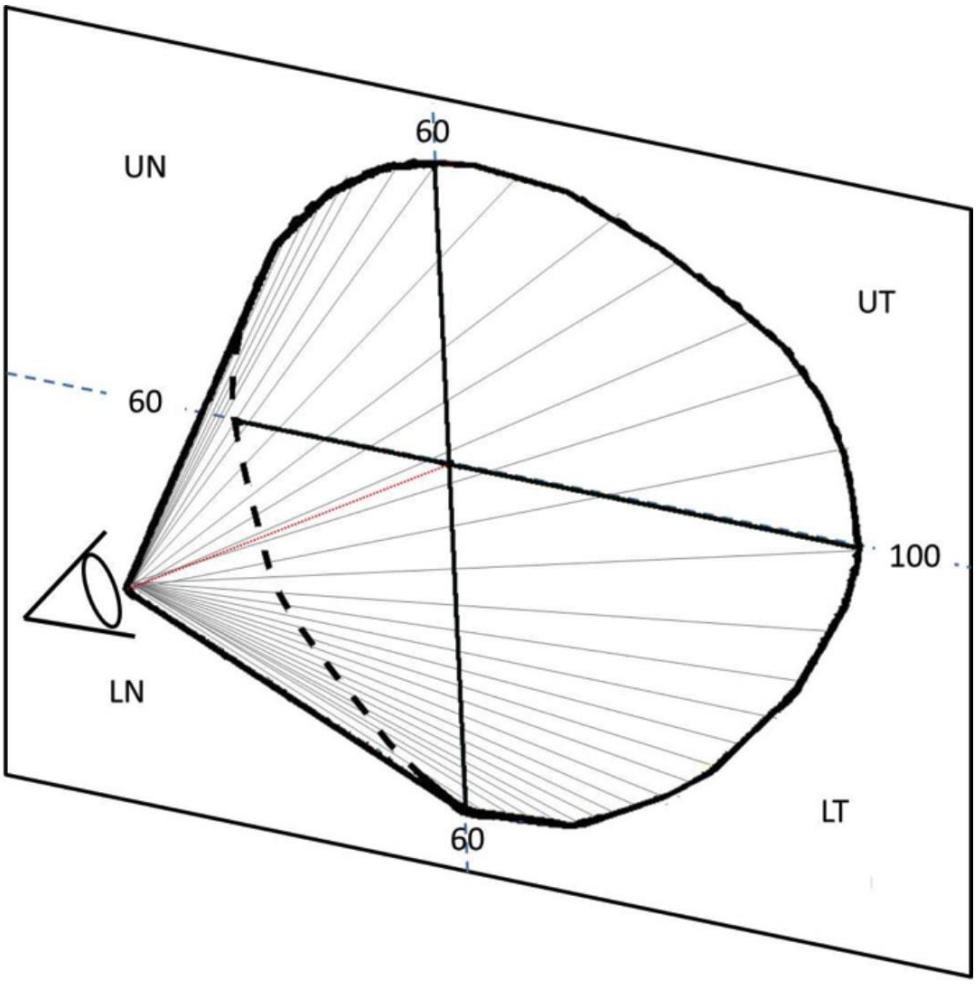

Figure 1-4 *Visual Field of Right Eye* [8]

The visual field refers to the entire area that can be seen when the eye is focused on a central point without moving, including what can be seen in the peripheral vision [9]. The condition of the visual field can be altered negatively due to many diseases. A visual field test assesses the condition of the visual field. The visual field test uses a center fixation light and switches on and off light stimuli in your peripheral and central vision [10]. The test evaluates vision loss due to glaucoma, damage to the visual pathways of the brain, and other optic nerve diseases. When glaucoma is diagnosed, the visual field test data is used to determine the severity of the disease. [11]

Glaucoma

Glaucoma is a disease that damages the optic nerve of your eye due to the fluids that builds up in the front part of your eye. That extra fluid increases the pressure in your eye, damaging the optic nerve. Glaucoma is a leading cause of blindness for people over 60 years old. But blindness from glaucoma can often be prevented with early treatment. The loss of vision can be detected using the visual field test.

Humphrey Field Analyzer

The visual field test is conducted using a device called the Humphrey Field Analyzer. The patient is first asked to insert his or her head into a large bowl-like instrument. The head of the patient is held still by a forehead and chin rest inside the large bowl-like instrument. The patient then stares at a source of light straight ahead and light spots of different intensities are flashed from random points in the visual field. Each time the patient sees one of these lights, he or she immediately presses a button or use some other means to indicate his or her response [12]. The test can take 10 minutes to an hour depending on the patients' ability to concentrate. The data from the Humphrey Field Analyzer is collected and plotted to map the blind spots of each eye for the patient. The ophthalmologist uses the report generated by the HFA to diagnose the patient eye condition.

Figure 1-5 *Humphrey Field Analyzer*

The visual field test can often be time consuming as the patient might lose concentration during the test which will require the test to be retaken. This incurs additional cost to the patient as he or she will be spending extra time at the clinic or hospital. A simple visual field screening test can be taken before taking the test using the HFA. Taking a visual field test using the HFA can be time consuming as it can take anywhere from 10 minutes to an hour depending on the ability of the patient to maintain concentration. If the test using the HFA has to be retaken, then that incurs additional cost to the patient. The visual field test can take lesser time if the technician has an idea which area of the visual field of the patient has a defect. The objective of this project is to create a simple, personal visual field screening application using a virtual reality headset such as a Google Cardboard and an Android device such as a Nexus 6P. Such a visual field test device is also cost-effective, portable, accessible, and that which enables self-administered eye screening tests.

Visual Field Easy

The Visual Field Easy (VFE) application is a screening tool to identify visual field defects in patients. It is an iPad app that was developed by George Kong software at the University of Iowa. The app flashes light spots at different areas of the visual field of the patient. This application enabled visual field screenings in remote areas of the world where access to bulky medical equipment is limited. The developers did a study to evaluate the accuracy and efficiency of this application to perform a visual field screening test. The number of missed fixation points and missed test locations of the VFE app correlated well with results from a standard visual field test procedure. The study authors said that their findings indicated that rapid and accurate screening for eye disease can be achieved using the VFE app [13]. This application can also be replicated with a Google Cardboard. A slight advantage that VR headset has is that it can provide more accuracy as the space between the eye and the screen is more consistent due to frame of the headset. If an application like VFE can be replicated with Google Cardboard headset then it can possibly provide a more cost-effective and functional visual field screening device.

CHAPTER 2. TECHNICAL DEVELOPMENT

In this chapter, we will discuss the technical details of the visual field test and the Google Cardboard with which the test will be replicated.

Visual Field Test

As discussed in Chapter 1, the visual field test is evaluated by a device called the Humphrey Field Analyzer (HFA). The patient is asked to place their head inside a bowl like instrument called the test bowl (Fig 2.1) and a lens is placed in front of their eye. The patient is asked to focus on a central fixation light. The patient focuses on the central fixation light throughout the entire test for each eye. Light spots, also known as “targets”, of different intensities are displayed at random points in the visual field and a buzzer is pressed by the patients when they see a target. It is important that the patient is focused on the central fixation light as some targets will be displayed in the peripheral vision and in order to obtain reliable results. Visual field tests can be difficult as it depends on the patients’ ability to respond quickly and requires a high level of concentration. The patient has to press a buzzer when they see a target, and this requires quick reflexed. As it is possible to miss targets and make mistakes, the test can take longer than usual for some patients [14]. The test is administered by a technician who monitors the patient and determines if the patient maintained appropriate concentration throughout the test in order to obtain reliable results [15].

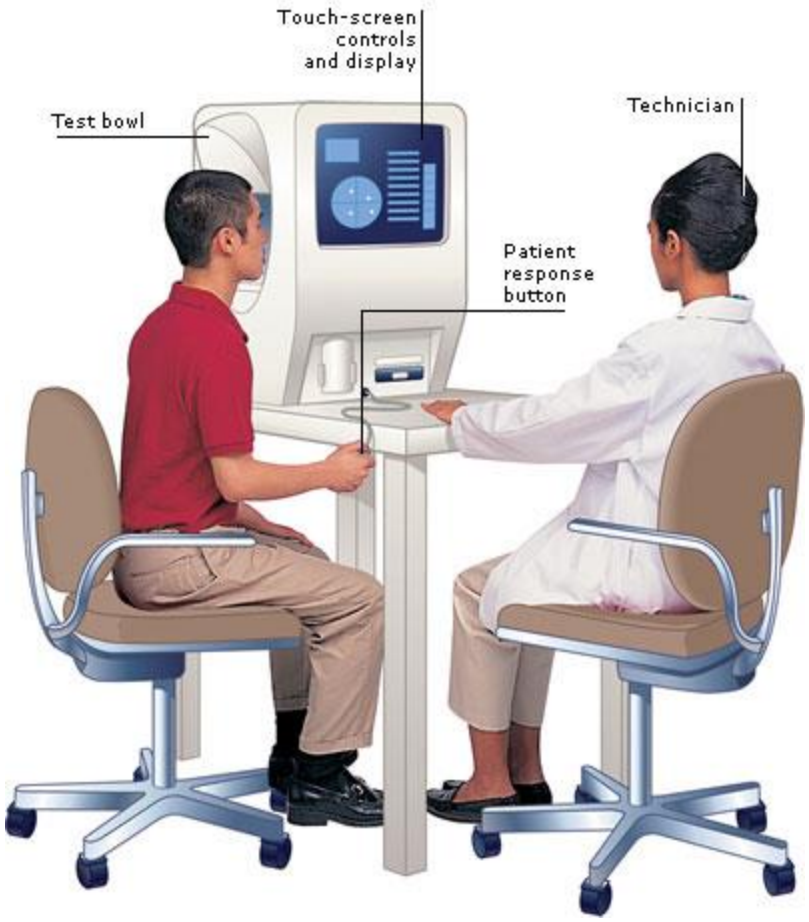

Figure 2-1 *Visual Field Test using the Humphrey Field Analyzer (HFA)* [16]

Most patients with visual field loss have some detectable field loss within the central 24-30 degree area of the visual field [17]. The visual field is commonly tested at a 30 degree field of view for the patient. A diagram of the visual field of the right eye is shown in Figure 2-2.

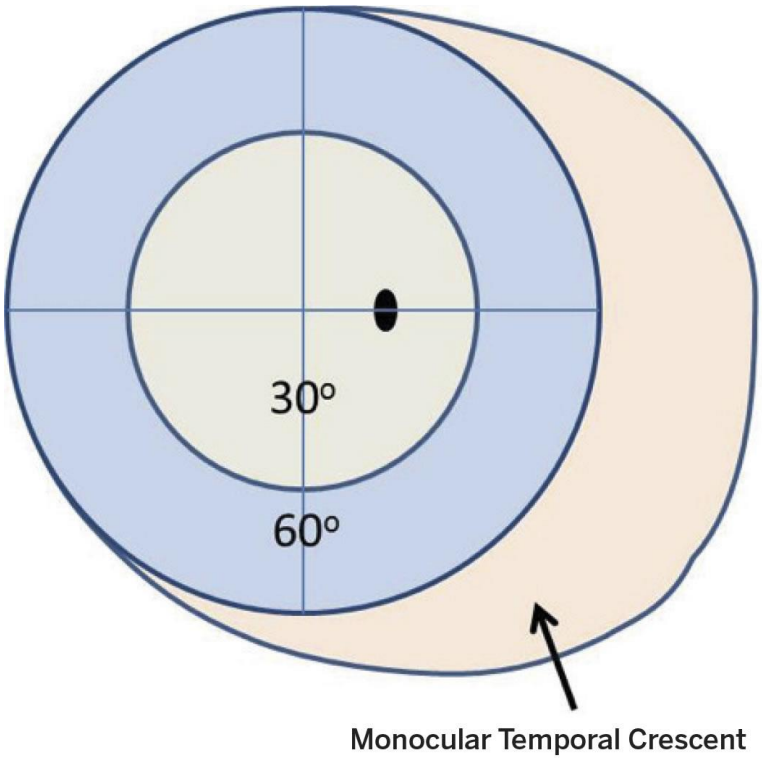

Figure 2-2 *Diagram of visual field of the right eye. The three regions of the field are the central field (0 to 30 degrees), the peripheral field (30 to 60 degrees), and the temporal crescent.* [18]

The visual field extends for approximately 60 degrees superior, inferior, and nasal and 100 degrees temporally. The visual field is divided into three major parts: the central 30 degrees, the peripheral field (from 30 to 60 degrees), and the temporal crescent [19]. The black spot in the central 30 degrees is known as the blind spot. The blind spot is the area where a patient cannot see any targets.

In the visual field test, the target light spots are displayed in a fixed grid pattern. The spacing between the locations of these target light spots varies according to the examination area targeted. When testing with the HFA, the test patterns are named 30-2 or 24-2 which means the horizontal spacing between the target light spots within a field of view range of 30 or 24 degrees respectively is 3 degrees [19]. An example of the 30-2 pattern is shown in Figure 2-3:

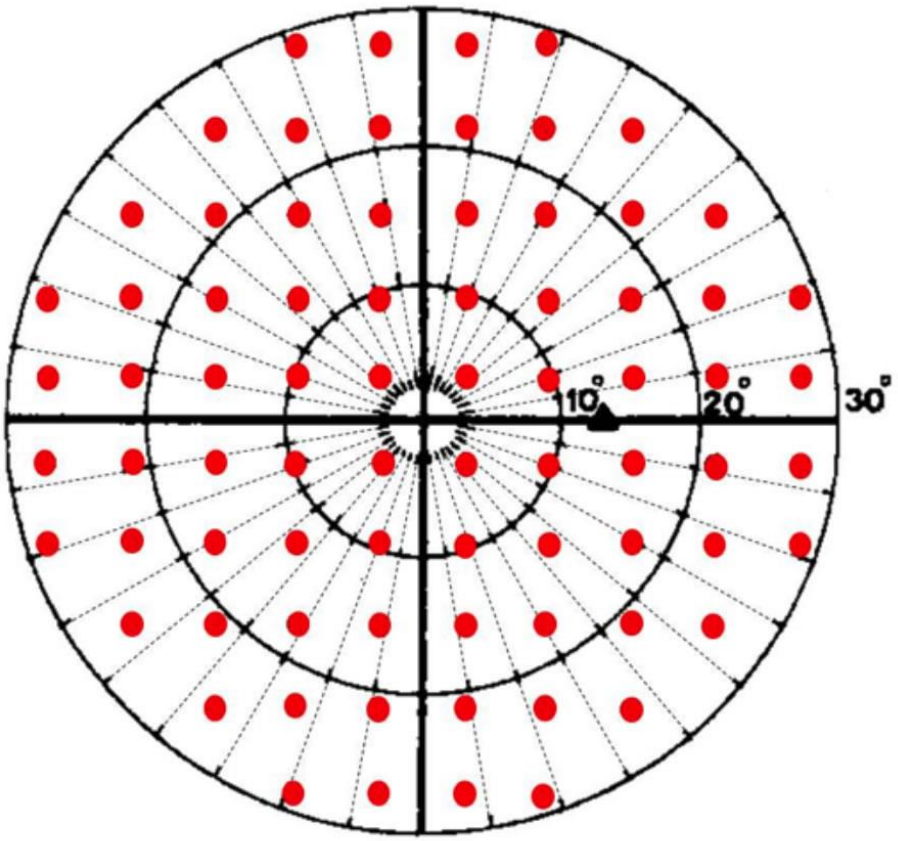

Figure 2-3 *30-2 pattern: The distribution of the target light spots are at equal distances from the horizontal axis at 3 degrees.* [8]

This is the pattern that will be aimed to replicate with the Google Cardboard application. The target light spots will be displayed randomly for each eye separately and the data will be recorded as to which targets the patient does not see with the help of a buzzer.

Once the test is completed by the patient, a report is generated by the machine in order to assist the doctor to evaluate the condition of the eye. The condition of the field of view for each eye is generated. An example set of typical results is shown in Figure 2-4 and Figure 2-5:

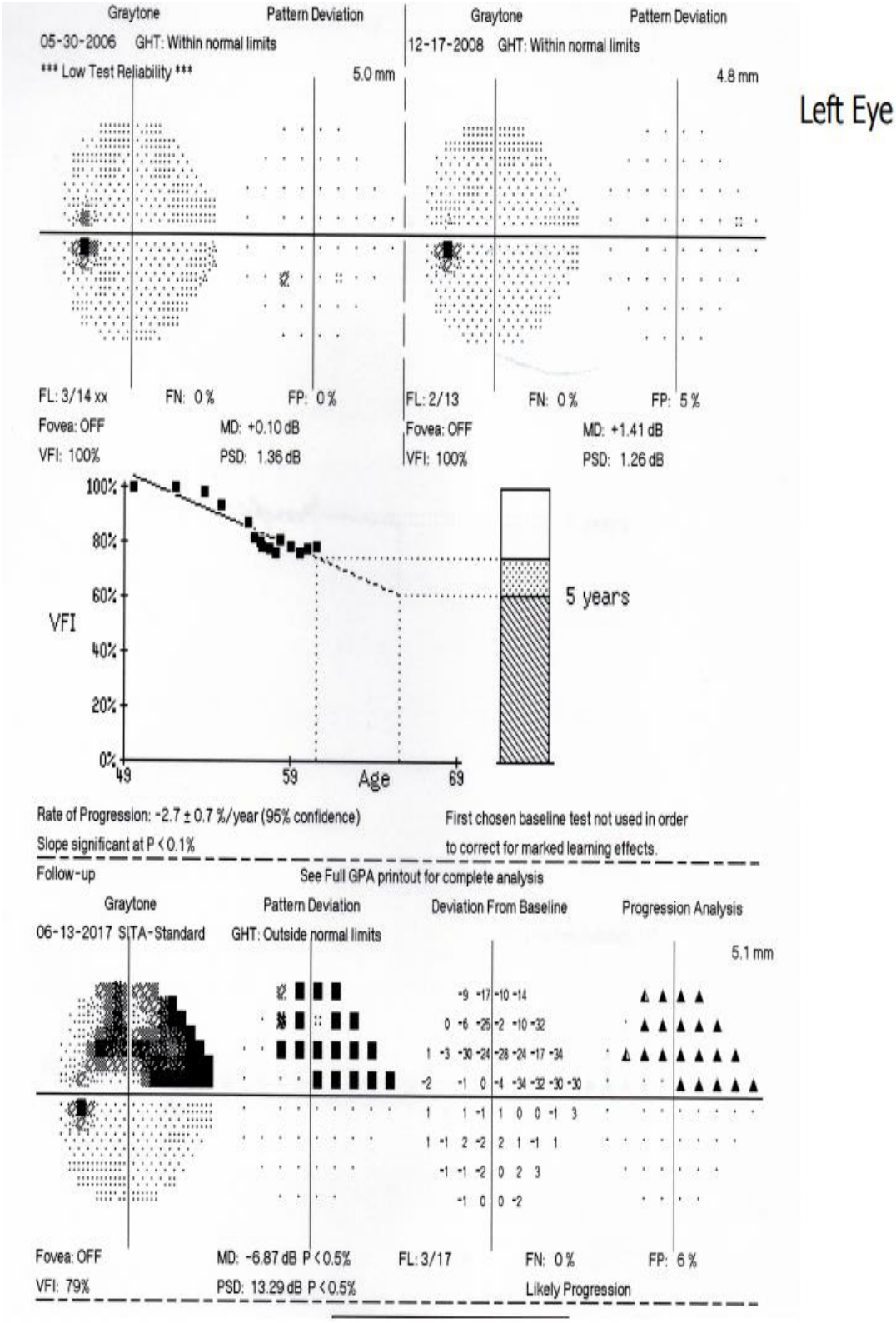

Figure 2-4 *HFA Visual Field Test Results (Left Eye)*

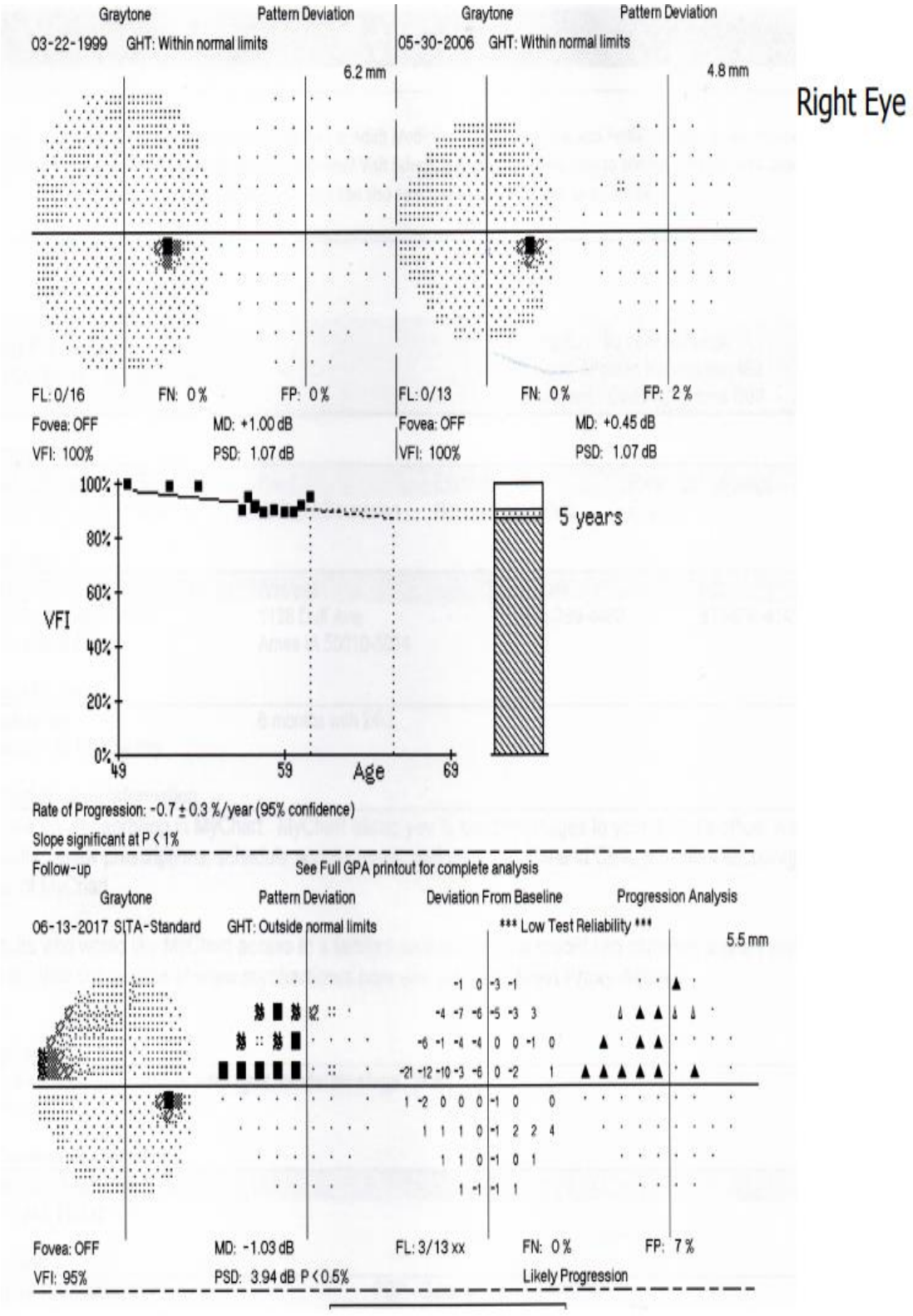

Figure 2-5 *HFA Visual Field Test Results (Right Eye)*

Google Cardboard

Figure 2-6 *Google Cardboard*

Google Cardboard is a low-cost virtual reality headset developed by Google and works with most smart phones. It was launched in 2014 by Google to introduce the concept of "Cardboard VR" [20]. While the Google Cardboard has its technical limitations when compared to the Oculus Rift [21] or the HTC Vive [4], it can still be used to develop a wide variety of complex applications. Applications for Google Cardboard can be developed fairly easily with a basic programming background. Game-engines such as Unity3D [3] or Unreal [22] assists the developer in creating VR applications with a user-friendly interface and pre-loaded libraries. The headset specifications and assembly instructions are freely available on Google's website, which allows people to make their own VR headset from readily available parts. The basic parts that make up a cardboard viewer are a piece of cardboard cut into a precise shape, 45 mm focal length lenses, magnets or capacitive tape, a hook and loop

fastener (such as Velcro), and a rubber band. Pre-assembled cardboard VR headsets are also available for less than $5 from multiple vendors [23].

Field Of View (FOV)

One of the most important factors to consider in this project was the Field Of View of the headset. The maximum FOV that Google Cardboard is capable of displaying is 80 degrees [6]. The maximum FOV of the human eye that is tested in the visual field test is 60 degrees. Hence, the FOV of the Google Cardboard is sufficient for this project.

Unity3D

Unity3D is a software framework designed for the creation and development of video games, mobile applications, and virtual reality applications. The framework has a user-friendly graphical user-interface in order to aid developers in creating applications. The programming language that is used is C# which is a relatively easy-to-learn programming language when compared to the C++ programming language. This game engine also has the capability to build games for multiple platforms such as Android devices, iOS devices, gaming consoles (PlayStation 4), Linux operating systems, personal computers, etc. Online forums and resources are available specifically for Unity3D in order to help with application development. This game engine is also freely available to use for development of personal applications.

CHAPTER 3. EXPERIMENTAL SETUP AND RESULTS

Experimental Setup

First, Unity3D was set up before developing the application for the cell phone. Then, the application to replicate the visual field test was developed in Unity3D and exported to a cell phone. The application was then tested with the Google Cardboard viewer. Finally, the output from the application was plotted and validated.

Display Device

The device that is used in this work is an Android device manufactured by Huwaei known as Nexus 6P. This device runs on an Android 'Oreo' (version 8.0.0) mobile operating system. It is capable of running Google VR applications and is compatible with Google Cardboard. The phone has to be configured to 'Debug Mode' before installing any new applications to it. Debug mode enables developers to install new applications into the phone and debug any errors if necessary. The basic hardware specifications are listed below in Table 1:

Table 1 Nexus 6P Specifications [24]

Display	5.7-inch 2560x1440 AMOLED
Processor	Qualcomm Snapdragon 810 Processor
Memory & Storage	RAM: 3GB Internal storage: 32GB
Dimensions	159.3 x 77.8 x 7.3 mm
Weight	178g
Battery	3450 mAh

The cell phone device is inserted into the Google Cardboard viewer as shown in Figure 3-2:

Figure 3-1 *Nexus 6P*

Google Cardboard Viewer

The Google Cardboard viewer is designed to work with Android or iOS phones with screen sizes from 4 to 6 inches. It has an interactive button that works with all compatible phones. Technical specifications of the Google Cardboard are shown in Table 2:

Table 2 Google Cardboard Specifications [25]

Lens Diameter	34 mm
Field Of View (FOV)	80 degrees
Magnets	0.75 inches diameter 0.12 inches thickness
Dimensions	5.9 x 3.5 x 2.2 inches
Weight	96g

In order to create a cardboard viewer, only 6 parts are required:

1. Cardboard
2. Lenses
3. Magnets
4. Two strips of Velcro
5. Rubber band
6. NFC tag (optional)

These parts are easily accessible almost anywhere in the world and can be easily purchased to create a cardboard viewer with simple easy-to-follow instructions from the Google website. An image of the required parts is shown below:

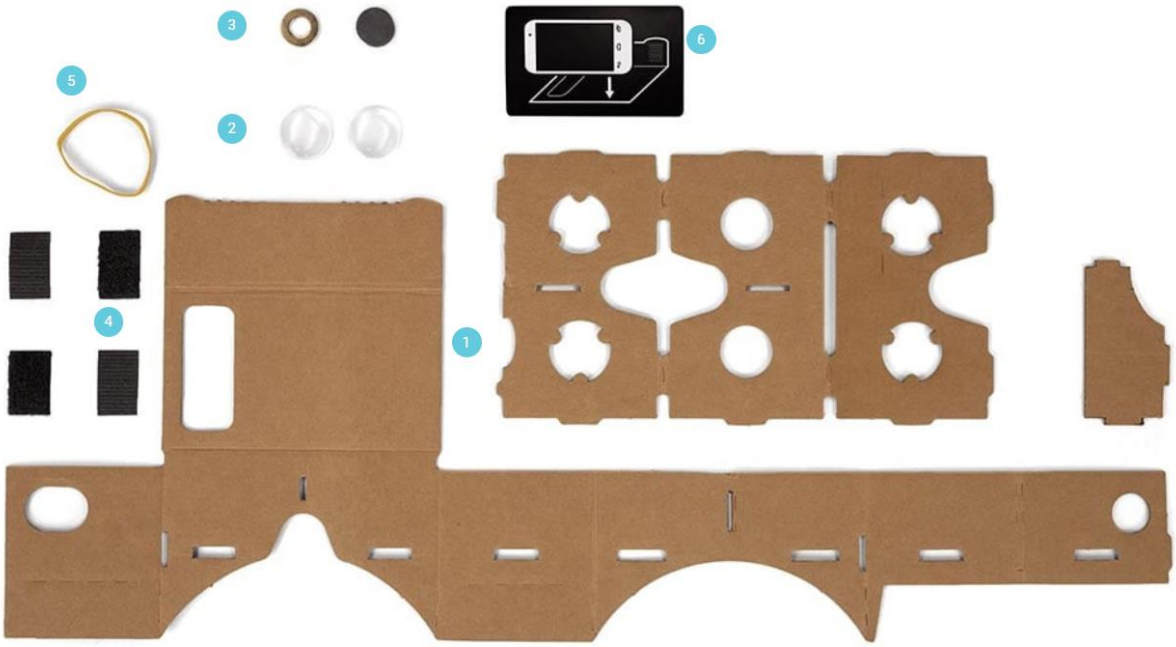

Figure 3-2 *Google Cardboard Viewer Parts* [25]

Setting Up Unity3D

The application was developed using the game engine, Unity3D. At first, the software development kit (SDK) for developing Google VR applications was integrated into the Unity3D development platform. This first step is required to be done in order to use virtual reality features with the phone application. The instructions to install the Google VR SDK package for Unity3D is given in the Google developer's website [26]. It is available freely online and the instructions are fairly simple to follow. In order to use the Google VR SDK, there are two basic hardware and software requirements.

- The cell phone has to be an android device with at least the API SDK version 21 (also known as "Lollipop") installed. Gyroscope sensors must be included in the device.
- To develop Google VR applications, you must have a viewing device, a Google Cardboard or Google Daydream (VR headset).

After integrating the SDK, the ‘Track Rotation’ and ‘Track Position’ features were disabled by unchecking them in the ‘Inspector’ area of Unity3D development interface. The ‘Track Rotation’ and ‘Track Position’ features uses the phones’ gyroscope and accelerometer in order to track the orientation and position of the head. These two features are not required for this application and hence, they are disabled as shown in Figure 3-3:

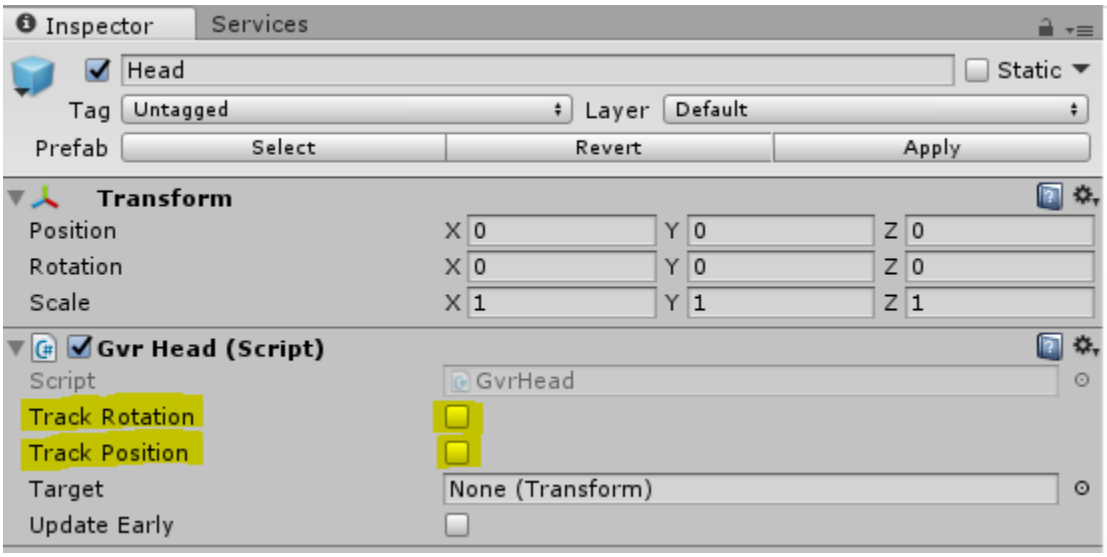

Figure 3-3 *Track Rotation and Track Position disabled in the Inspector (Screenshot)*

The maximum Field Of View (FOV) that is required for this application is 30 degrees so the FOV of the main camera is also set to 30 degrees. Setting the FOV in the beginning of development will help with placement of target light spots within the FOV. Setting the FOV in the Unity3D editor is shown in Figure 3-4:

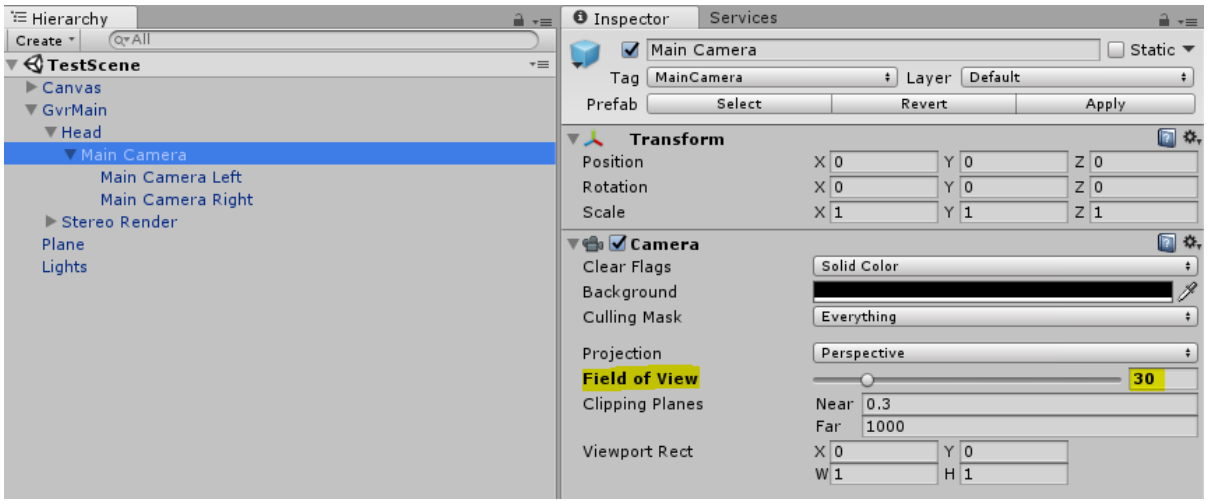

Figure 3-4 *Field Of View set to 30 degrees (Screenshot)*

Developing The Application

Once Unity3D and the cardboard viewer were configured and set up, the software was developed by using interactive features in Unity3D and by writing code in C# (Appendix A). First, the visual field test requires a central fixation point so the user can focus on a central spot on the screen while taking the test. A red dot was added to the application with Unity3D. The central fixation point is visible at all times when running the application. Then, a flat black surface was created so it will act as the reflective surface as to where a light source will shine on and the user will be able to see a spot of light on the display. The visual field test blinks lights at random positions within a field of view of 30 degrees. In order to replicate this pattern of lights blinking at random positions within a FOV of 30, a pre-fabricated model of a ‘spot light’ is created. This light is instantiated into a 7x7 rectangular pattern of lights. The initialized light pattern is shown in Figure 3-5:

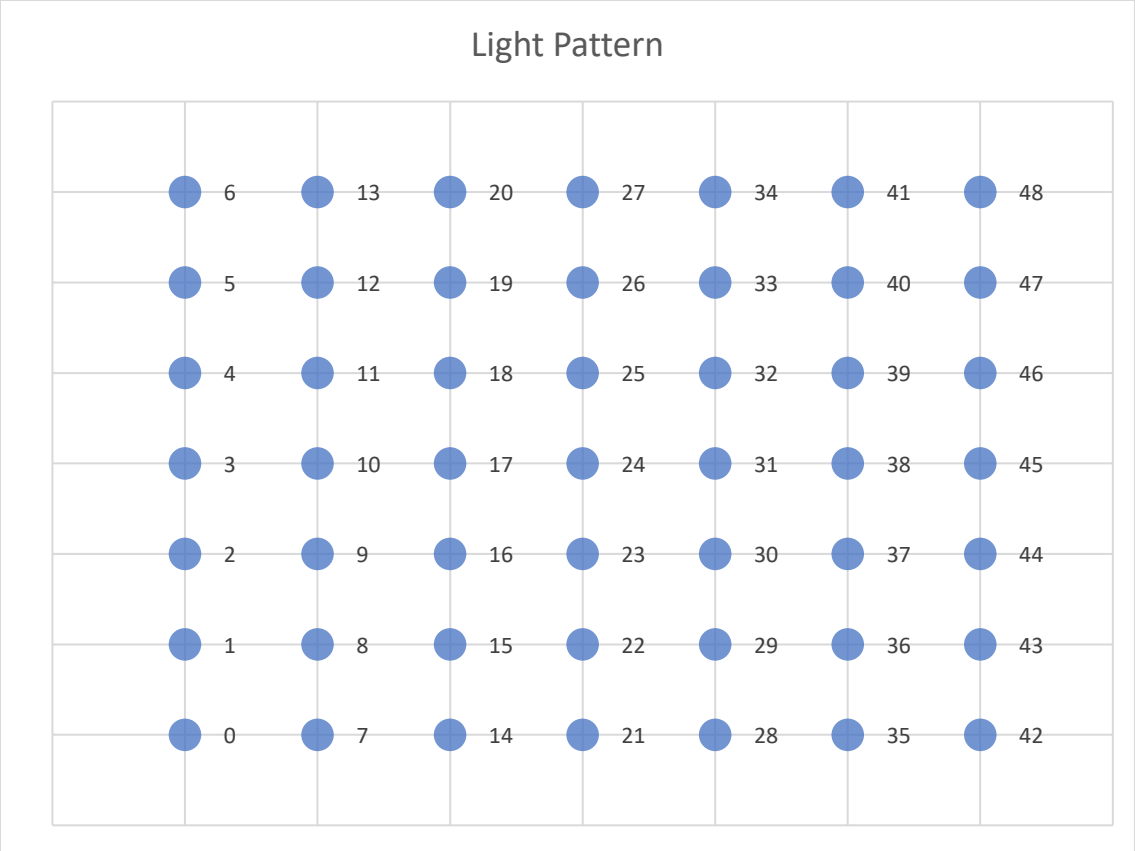

Figure 3-5 Initialized 7 x 7 Light Pattern

These lights are initially turned off so only a black screen can be seen on the screen in the beginning of the application. The location of each of the lights are assigned a unique identifier and stored in a list. For example, if the location number is 1, then the unique identifier is set to 1. This is done to help keep track of the lights when the application is running. Along with the unique identifier, a trigger value is associated with each of the lights. The trigger value is used to track whether the light was seen or not seen. Initially, before all the lights are switched on or off, they all have a trigger value of 0. If the light was seen by the user then its trigger value is set to 1. If the light was not seen after it has been displayed, then the trigger value is set to -1.

A C# (C-sharp) function named ‘System.Random’ was used to generate random numbers between 0 and the number of lights to be displayed. For example, if there were 9

lights, then the function would return a random value between 0 and 9 each time it is called. A for-loop was programmed so that every time a random number is generated, the program will find the light associated with the random number and turn it on for 0.2 seconds, then turn it off.

If the user sees this light after it blinks, then the user can press down on the magnetic button on the viewer. When the button is pressed, the location of the light that was seen is recorded in a file within the application. If the light is seen, then it is not displayed again and the trigger value for that specific light is set to 1. If the user does not see that light that was turned on and the next light is turned on without the user pressing the button for the previous light, then the trigger value for the previous light is set to -1. The light that was missed is also recorded in a separate file within the application. All the lights that were missed are programmed to be displayed randomly again with a higher intensity. Like earlier, the lights that are seen and not seen are recorded.

When all the lights for one eye are finished displaying, then the lights are displayed again for the other eye. The data is recorded for each eye, as to which lights the user sees or not. The data is parsed from the mobile device as text files and plotted in excel. Examples of the test results are shown in the next section.

Results

The application was tested by running it through a basic test process. The basic test process involved blinking all the lights for each eye, where the user pressed the button as they saw a blinking light, and the data acquired from the application was plotted to show lights seen in the visual field of each eye. First, a 3 x 3 grid of light pattern was tested in order to simplify results for validation. The light number assigned to each light is placed on the right in the following plots. The 3 x 3 pattern is shown in Figure 3-6:

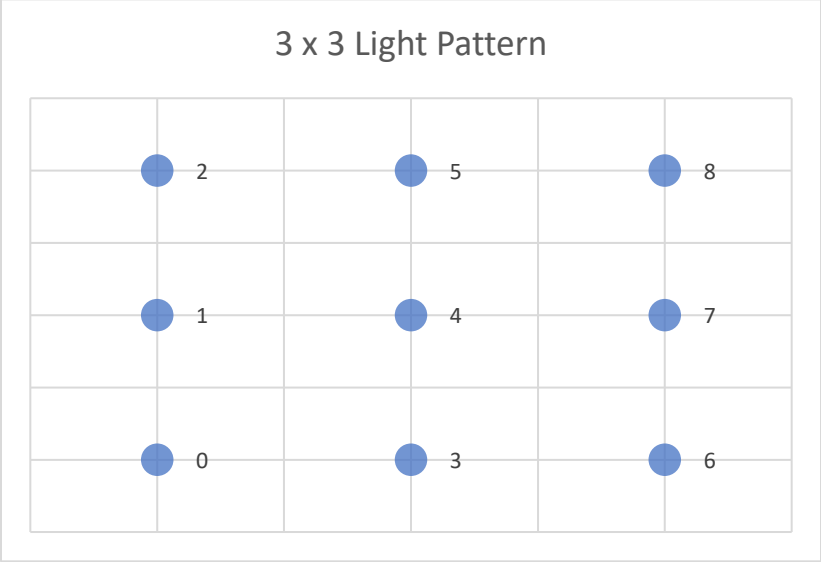

Figure 3-6 *Light Pattern (3 X 3)*

We have performed limited testing to check for functionality of the application, including response times, proper sequencing of stimulus presentation, and error detection logging. The user focuses on the central fixation point and watches for the lights as they take the test. The test is repeated and we purposely did not press the button for some lights in order to check results. These results are shown in Table 3 and Table 4 and were revised to validate that the application was recording data correctly. Figure 3-6 & Figure 3-7 shows an example of the visual representation of the results.

Table 3 Right Eye Results (3 x 3 Pattern)

Light No	0	TRUE
Light No	1	TRUE
Light No	2	FALSE
Light No	3	TRUE
Light No	4	TRUE
Light No	5	TRUE
Light No	6	FALSE
Light No	7	TRUE
Light No	8	TRUE

Table 4 Left Eye Results (3 x 3 Pattern)

Light No	0	TRUE
Light No	1	TRUE
Light No	2	TRUE
Light No	3	FALSE
Light No	4	TRUE
Light No	5	TRUE
Light No	6	TRUE
Light No	7	TRUE
Light No	8	TRUE

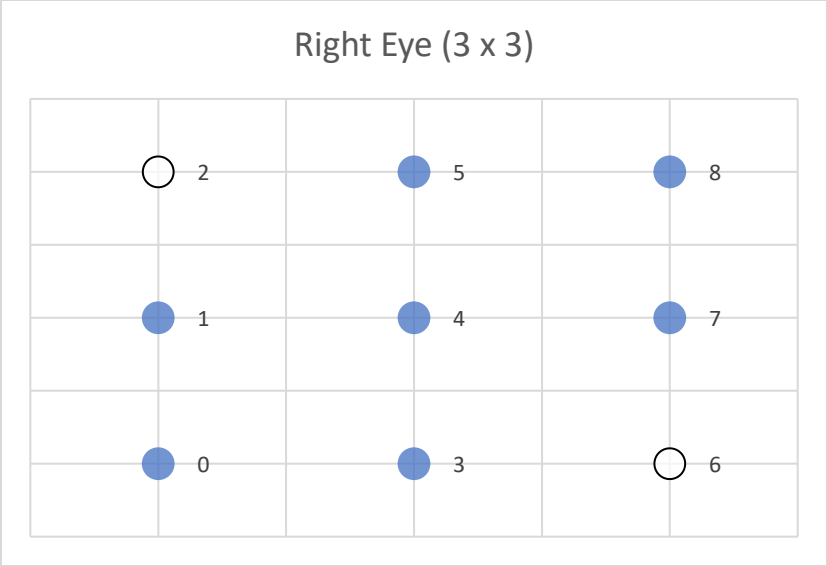

Figure 3-7 *Right Eye Light Pattern Results (3 X 3)*

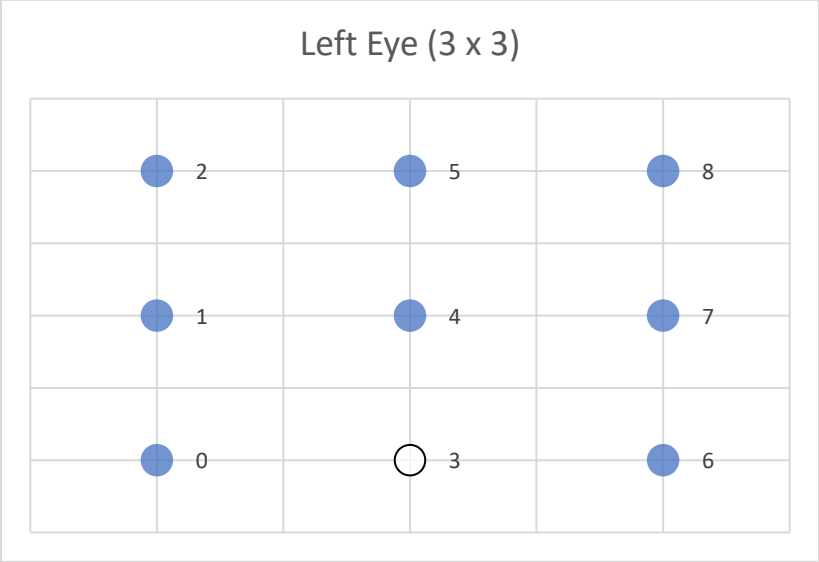

Figure 3-8 *Left Eye Light Pattern Results (3 X 3)*

The application we developed uses a 7 x 7 light pattern for a total of 49 light spots presented to each eye. During testing, we missed some lights intentionally; and there were some lights that could not be seen naturally because they were outside the peripheral vision, and because there is a natural blind spot at the nerve location. We kept track of the missed lights and verified the match with the results. The test was repeated multiple times to ensure

correct acquisition of results. The button on the Google Cardboard was also tested by pressing the button multiple times whenever a light was turned on and off. The data recorded showed only one instance of the light number that was seen. We were able to verify that the light that was seen was recorded only once though the button was pressed multiple times. Whenever a light was seen, the button was pressed. All the light numbers were correctly recorded, so we were able to verify the reliability of the button. This helped verify that the application was working correctly. The results for each eye with the 7x7 pattern are shown in Figure 3-9 and Figure 3-10.

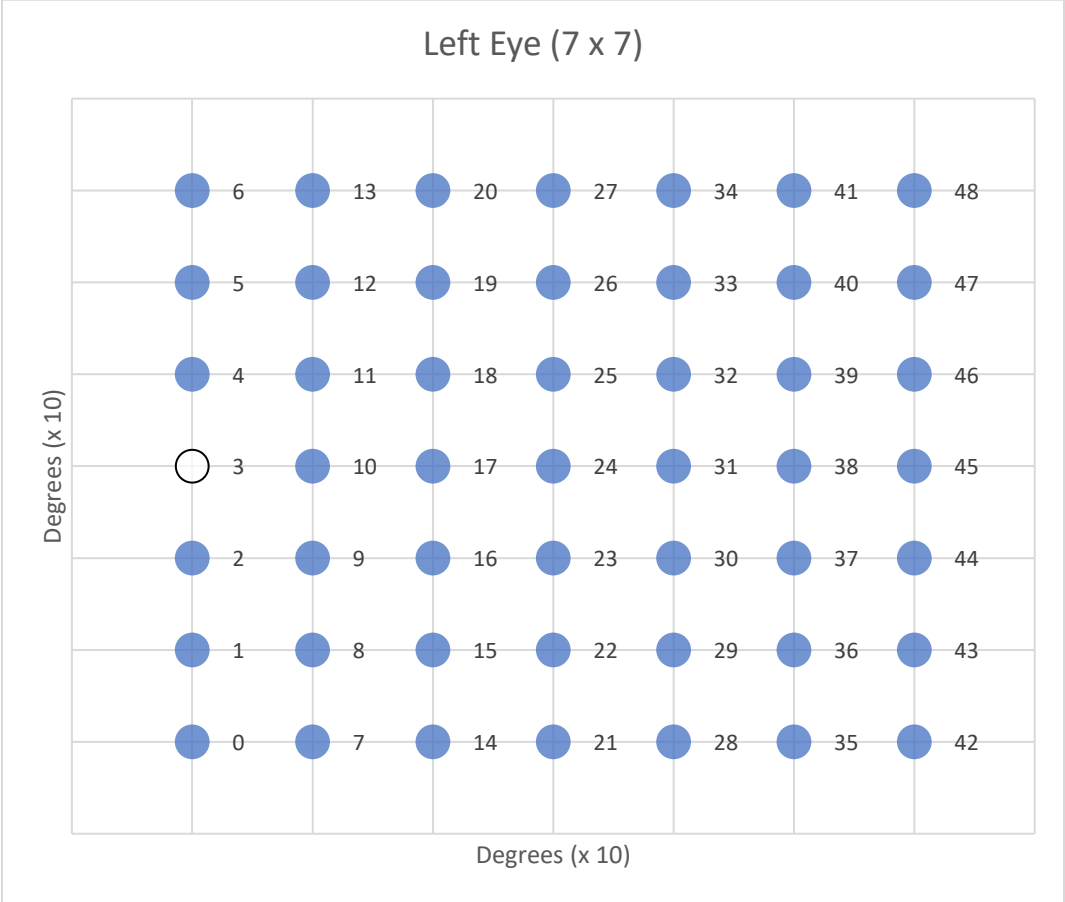

Figure 3-9 *Left Eye Light Pattern Results (7 X 7)*

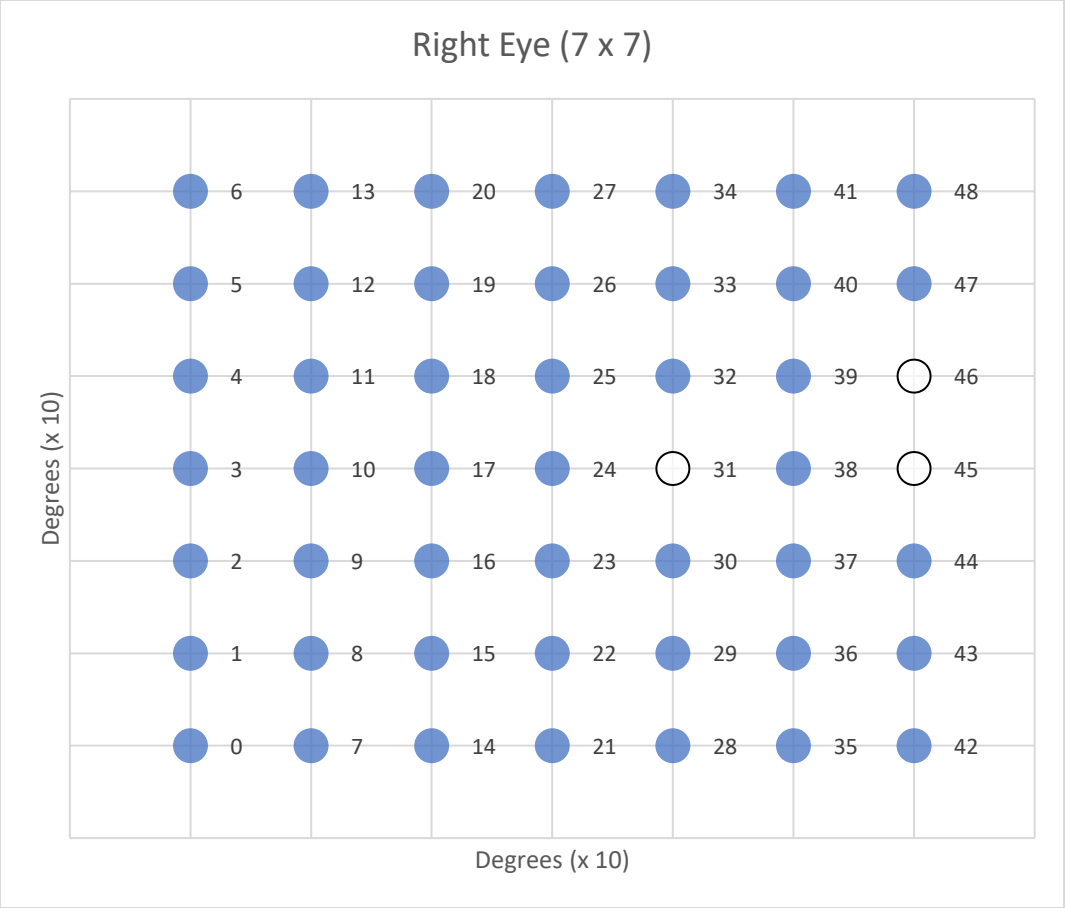

Figure 3-10 *Right Eye Light Pattern Results (7 X 7)*

The blue dots represent all the lights that were seen and the white dots represent all the lights that were not seen. This plot helps evaluate the condition of the visual field of a patient.

CHAPTER 4. DISCUSSION AND CONCLUSION

Discussion

The visual field graphs shown in the results give an approximate visual representation of the visual field condition of each eye. An approximate representation of the visual field condition is sufficient to assist in conducting the VFT using the Humphrey Field Analyzer. Hence, it is proved that it is possible to conduct a visual field screening test with a low-cost apparatus by combining a Google Cardboard headset and a smart phone. The application used to run the test on the smart phone can be made freely available, therefore, making it easily accessible to areas with limited access to eye clinics or hospitals.

Other research includes using Google VR headsets to recreate the visual field test but these headsets have built-in displays which significantly increases costs and portability [27]. The built-in displays have advantages such as increased field of view (FoV), higher processing capability, and better screen resolution. But along with this powerful VR headset comes the additional cost of the technology. The Google Cardboard headset requires a piece of cardboard and wide-angled lenses which is accessible in urban and rural areas. Being easily accessible is one of the main advantages of using the cardboard headset. The cost of the cardboard also makes it affordable to families or individuals with low-income. Nowadays, virtual reality capable smart phones are also available anywhere in the world at a low cost. With its low cost applications and easy accessibility, there is a great potential for virtual reality applications in the healthcare industry.

With the visual field test, signs of diseases such as glaucoma can be detected. The earlier the disease is detected, the lesser the rate of loss of vision in the eye. The machine used to conduct the visual field test, known as the HFA, is available only in certain clinics

and hospitals. The test is conducted under the supervision of a trained technician. The HFA costs about $8000. The machine is also bulky and weighs about 63 pounds [28] which makes it hard to transport. Virtual reality can help save time and money to patients who want to take the visual field test. The visual field screening application is loaded on a smart phone and a low-cost VR headset such as a Google Cardboard can be used by the patient to take the test. The patient can conveniently take the test at home before going to the eye clinic. Populations who are economically disadvantaged can have access to such a device. Doctors who go on mission's trips or medical camps, can easily carry this device even to remote areas. The visual field test can cost on an average of $150 [29] but with such a VR application, it can help the patient save time in the clinic by getting an idea of the condition of their own visual field. The data can also be sent to the clinic directly to be evaluated by the doctor. The doctor can assess the condition of the visual field of that patient and decide if the patient has to come in for further testing or not. From the experiments, one result from my experiment shows similarity to the results of an actual visual field test conducted by the HFA.

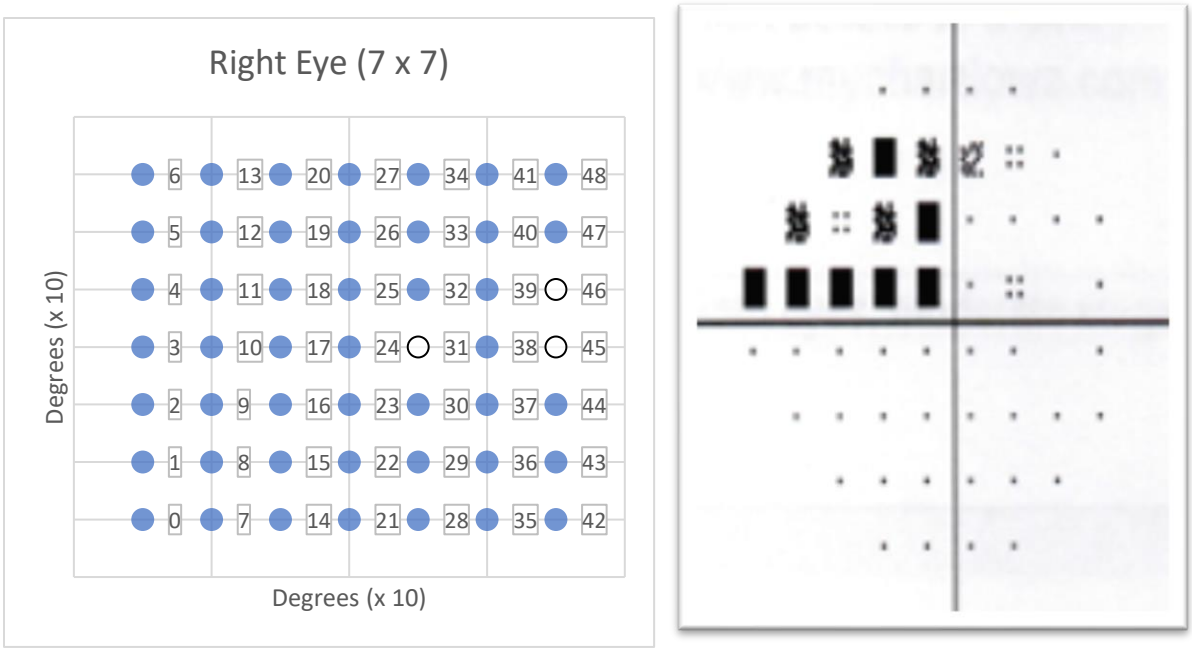

Figure 4-1 *Right Eye Visual Field Test Results using Google Cardboard (Left Image) and Right Eye Humphrey Field Analyzer Results (Right Image)*

The reliability of the results of this application depends on the patients' judgement of how he/she maintained concentration throughout the test process. The patient has to focus on the central fixation point in order for the test results to be reliable. Maintaining focus on the central fixation point is hard to do even on a Humphrey Field Analyzer (HFA). The HFA tracks the movement of the eye to check if the patient is focused on the central fixation patient and is able to conclude which results are reliable or not. VR headsets do not have eye tracking but there has been development of eye tracking used in VR headsets [30]. Using VR headsets with eye tracking capability can help solve reliability issues.

Conclusion

My application displays light spots of a certain intensity to a patient within a 30 degree field of view and record data as to which lights the patient can see or not. The data results are plotted to show the condition of the visual field of the patient and indicate any sign of vision loss for each eye. This application is a self-administered test and does not need

the assistance of technician. The VR headset and cell phone are easily portable and accessible even in areas with limited access to technology. From the application I developed, I can conclude that it is possible to conduct a visual field screening test using low-cost VR components.

There is certainly some future work that can be done to improve the functionality of this application. Different levels of light intensity can be displayed according to the patients' ability to see the light. The HFA uses a certain algorithm to run the test so an algorithm that optimizes the efficiency of the test can be programmed into the application. There is also a possibility for eye tracking to be integrated in VR headsets in the future so it will help in achieving reliable test results. In the eye clinic, the visual field test might have to be retaken due to common mistakes by the patient such as losing focus of the central fixation point during the test. This leads to the patient retaking the test which in turn leads to spending additional time at the clinic retaking the test. With the VR application, it is possible to retake the visual field test at the comfort of the home and spend less time waiting at the clinic to retake the test. As VR technology keeps developing rapidly, there is a possibility that VR devices will replace eye testing devices like the Humphrey Field Analyzer in the future.

REFERENCES

[1] S. DiGuilio, "NBC News," 22 August 2017. [Online]. Available: https://www.nbcnews.com/mach/science/3-ways-virtual-reality-transforming-medical-care-ncna794871. [Accessed 9 February 2018].

[2] Martin Němec, Radoslav Fasuga, Jan Trubač, Jan Kratochvíl, "Using Virtual Reality in Education," Stary Smokovec, Slovakia, 2017.

[3] "Unity," [Online]. Available: https://unity3d.com/. [Accessed 8 February 2018].

[4] "Vive," [Online]. Available: https://www.vive.com/us/. [Accessed 8 February 2018].

[5] P. Sinclair, "All Home Robotics," 15 December 2017. [Online]. Available: https://www.allhomerobotics.com/oculus-rift-vs-google-cardboard/. [Accessed 6 February 2018].

[6] "Google VR," [Online]. Available: https://static.googleusercontent.com/media/vr.google.com/en//cardboard/downloads/manufacturing-guidelines.pdf. [Accessed 8 February 2018].

[7] N. Fliesler, "Boston Children's Hospital," Vector, 18 April 2017. [Online]. Available: https://vector.childrenshospital.org/2017/04/virtual-reality-headsets-could-treat-amblyopia/. [Accessed 9 February 2018].

[8] [Online]. Available: https://www.cehjournal.org/article/visual-fields-interpretation-in-glaucoma-a-focus-on-static-automated-perimetry/.

[9] "Medline Plus," [Online]. Available: https://medlineplus.gov/ency/article/003879.htm. [Accessed 6 February 2018].

[10] "Galucoma Research Foundation," [Online]. Available: https://www.glaucoma.org/treatment/what-is-a-visual-field-test.php. [Accessed 6 February 2018].

[11] "Glaucoma Research Foundation," [Online]. Available: https://www.glaucoma.org/treatment/why-do-i-need-a-visual-field-test.php. [Accessed 6 February 2018].

[12] M. Haddrill, "All About Vision," 2017. [Online]. Available: http://www.allaboutvision.com/eye-exam/visual-field.htm. [Accessed 6 February 2018].

[13] Kahl, Kristie L, "Healio," Association for Research and Opthalmology, 6 May 2015. [Online]. Available: https://www.healio.com/ophthalmology/retina-vitreous/news/online/%7Bdf66396a-13cf-4e7f-8223-54eebd855926%7D/visualfields-easy-app-may-screen-for-visual-field-deficits. [Accessed 13 March 2018].

[14] Broadway, David C, "Visual field testing for glaucoma," vol. 25, no. 79 & 80, 2012.

[15] Ravi Thomas, Ronnie George, "Interpreting Automated Perimetry," *Ophthalmology Practice,* vol. 49, no. 2, pp. 125-140, 2001.

[16] [Online]. Available: https://www.aviva.co.uk/library/images/med_encyclopedia/cfhg542visfie_001.jpg.

[17] Pradeep Ramulu, Sarwat Salim, "American Academy of Ophthalmology," 20 October 2017. [Online]. Available: http://eyewiki.aao.org/Standard_Automated_Perimetry#cite_note-Ballon1992-7. [Accessed 7 February 2018].

[18] [Online]. Available: https://s160131.gridserver.com/wp-content/uploads/Fig2.jpg.

[19] M. Yaqub, "Visual fields interpretation in glaucoma: a focus on static automated perimetry," *Community Eye Health Journal,* vol. 25, no. 79 & 80, 2012.

[20] Wendy Powell, Vaughan Powell, Phillip Brown, Marc Cook, Jahangir Uddin, "Getting Around in Google Cardboard – Exploring Navigation Preferences With Low-Cost mobile VR," in *2016 IEEE 2nd Workshop on Everyday Virtual Reality (WEVR),* Greenville, South Carolina, 2016.

[21] "Oculus," [Online]. Available: https://www.oculus.com/rift/#oui-csl-rift-games=mages-tale. [Accessed 8 February 2018].

[22] "Unreal Engine," [Online]. Available: https://www.unrealengine.com/en-US/what-is-unreal-engine-4. [Accessed 8 February 2018].

[23] C. Dougherty, "Google Intensifies Focus on its Cardboard Virtual Reality Device," New York Times, 2015.

[24] A. Dobie, "Android Central," 29 September 2015. [Online]. Available: Table 1.1 Example – Use this page if you are using a table in landscape orientation. The page number must be to the right (what would be the top when your paper is all orientated as portrait.. [Accessed 12 February 2018].

[25] "Cardboard," Google, [Online]. Available: https://vr.google.com/cardboard/get-cardboard/. [Accessed 12 February 2018].

[26] February 2018. [Online]. Available: https://developers.google.com/vr/develop/unity/get-started. [Accessed 12 February 2018].

[27] Dariusz Wroblewski, Brian A. Francis, Alfredo Sadun, Ghazal Vakili, Vikas Chopra, "Testing of visual field with virtual reality goggles in manual and visual grasp modes," *BioMed Research International,* vol. 2014, p. 10, 2014.

[28] "Humphrey Field Analyzer – HFA3 Visual Field Series," Zeiss, [Online]. Available: https://www.zeiss.com/meditec/us/products/ophthalmology-optometry/glaucoma/diagnostics/perimetry/humphrey-field-analyzer-hfa3.html#technical-data.

[29] "Glaucoma Service Foundation to Prevent Blindness," [Online]. Available: http://willsglaucoma.org/visual-fields-2. [Accessed 16 February 2018].

[30] P. Lamkin, "Wearable," 28 July 2016. [Online]. Available: https://www.wareable.com/vr/fove-eye-tracking-vr-headset-price-specs-release-date-1157. [Accessed 15 February 2018].

APPENDIX A. CODE FOR APPLICATION

C# code for the application developed in Unity3D is given below:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;
using System.IO;
using System;
using UnityEngine.UI;
public class BlinkingLights : MonoBehaviour {

    public int lightNo = 0;
    public string trueLightNo = "";
    public GameObject LightTest1;
    public GameObject LightTest2;
    public GameObject leftCamera;
    public GameObject rightCamera;
    public Light lightComp;
    public Light lightComp1;
    public Material LightON;
    public Material LightOFF;
    public float offsettime = 100;

    public Image crosshair;
    public List<Transform> theLights;
    public List<Transform> theLights2;
    public List<Transform> theLightsL;
    public List<Transform> theLightsR;
    public List<Transform> lightList;


    public List<Transform> backUp;
    Transform temp;
    public Transform currentLight;
    public Transform oldLight;
    int tempTrigger;
    bool completeState = false;
    bool startState = false;

    public bool leftEyeState = false;
    public bool rightEyeState = false;

    public List<int> lmissedLightsList = new List<int>();
    public List<int> rmissedLightsList = new List<int>();
    public List<int> lsecondmissedLightsList = new List<int>();
    public List<int> rsecondmissedLightsList = new List<int>();
    public List<int> thirdmissedLightsList = new List<int>();

    public int[] trigger;


    public int[] intensity;
    // Use this for initialization
    private int[] level;
```

```
    private bool[] completed;
    private float[] completionTime;
    private string[] scores;
    string fileName;
    string fileName4;
    string fileName6;
    // int i = 0;

    void Update()
      {



        if (GvrViewer.Instance.Triggered == true && startState == false)
        {
            StartCoroutine(Blink());
            startState = true;

        }

        if (completeState == true)
        {
            Application.Quit();
        }

          if (GvrViewer.Instance.Triggered == true)
          {
              Debug.Log("The trigger was pulled!");

              //Code to check and update the trigger Matrix & Intensity Matrix.

              if (currentLight != oldLight)
              {
                  int triggerIndex = Int32.Parse(trueLightNo);
                trigger[triggerIndex] = 1;

                if (!rightEyeState)
                {
                    string appendText = "Light Number: " + currentLight.name + "    "
+ trigger[triggerIndex] + "\n";

                    File.AppendAllText(fileName, appendText);
                }
                else
                {
                    string appendText = "Light Number: " + currentLight.name + "    "
+ trigger[triggerIndex] + "\n";
                    File.AppendAllText(fileName4, appendText);
                }


                  oldLight = currentLight;


              }
          }

      }
```

```
    void Start () {
        fileName = Application.persistentDataPath + "/RightEye.txt";
        fileName4 = Application.persistentDataPath + "/LeftEye.txt";


        // Center Light
        int lightNo = 0;
        theLights = new List<Transform>();
        String fileName1 = Application.persistentDataPath + "/lightNumber.txt";

        for (int x = -1; x <= 1; x++)
        {
            for(int y = -1; y<= 1; y++)
            {
                LightTest1 = new GameObject(lightNo.ToString());

                lightComp = LightTest1.AddComponent<Light>();
                lightComp.color = Color.white;
                lightComp.enabled = false;
                lightComp.transform.position = Camera.main.ScreenToWorldPoint(new
Vector3((Screen.width / 2) + x * 1000, Screen.height / 2 + y * 1000,
Camera.main.nearClipPlane));

                lightComp.range = 10;

                string value = "x =  " + x + "    y=  " + y + "    LightNo=  " +
lightNo + "\n";
                File.AppendAllText(fileName1, value);
                theLights.Add(lightComp.transform);

                backUp.Add(lightComp.transform);
                lightNo++;

            }
        }
        trigger = new int[theLights.Count];
        for (int i = 0; i < theLights.Count; i++)
        {
            trigger[i] = 0;
        }

        Vector3 vector = new Vector3(Screen.width / 2, Screen.height / 2,
Camera.main.nearClipPlane);
        //string appendText = "This is base vector " + vector.x + vector.y +
Environment.NewLine;
       // File.AppendAllText(fileName, appendText);
        theLights.OrderBy(wp=>wp.name);
        int height = Screen.height;
        print(vector.x);
        print(vector.y);
        int width = Screen.width;
        print(width);

        leftCamera = GameObject.FindGameObjectWithTag("LeftCamera");
        rightCamera = GameObject.FindGameObjectWithTag("RightCamera");
    }
```

```
public void startButton()
{

}


public void EyeControl(string camera, bool state)
{


    if(camera == "left")
    {

        leftCamera.SetActive(state);

    }

    if (camera == "right")
    {

        rightCamera.SetActive(state);
    }

}
IEnumerator Blink()
{

    if (leftEyeState == false)
    {
        EyeControl("left", false);
     }

    if (leftEyeState == true)
    {
        EyeControl("left", true);
        EyeControl("right", false);

        trigger = new int[theLights.Count];
        for (int i = 0; i < theLights.Count; i++)
        {
            trigger[i] = 0;
        }
    }

    yield return new WaitForSeconds(3);


    int n = theLights.Count;
    System.Random random = new System.Random();

    for (int i=n-1; i>=0; i--)
        {

        // Get Random number from (0,i)
        int rand = random.Next(0, i+1);
```

```
            theLights[rand].GetComponent<Light>().enabled = true;
            theLights[rand].GetComponent<Light>().intensity = 4;
            theLights[rand].GetComponent<Light>().bounceIntensity = 8;
            theLights[rand].GetComponent<Light>().range = 15;
            theLights[rand].GetComponent<Light>().type = LightType.Spot;
            theLights[rand].GetComponent<Light>().spotAngle = 1;
            currentLight = theLights[rand];
            trueLightNo = theLights[rand].name;
            int triggerIndex = Int32.Parse(trueLightNo);


            lightNo = triggerIndex;

            yield return new WaitForSeconds(offsettime);

            theLights[rand].GetComponent<Light>().enabled = false;
                theLights[rand].GetComponent<Light>().intensity = 0;

                lightNo = triggerIndex;

            yield return new WaitForSeconds(1);


            if (GvrViewer.Instance.Triggered == true)
            {

                trigger[triggerIndex] = 1;

            }

            yield return new WaitForSeconds(2);

            if (trigger[triggerIndex] == 0)
            {
                if (!rightEyeState)
                {
                    theLights2.Add(theLights[rand].transform);
                    trigger[triggerIndex] = -1;
                }
                else
                {

                    theLights2.Add(theLights[rand].transform);
                    trigger[triggerIndex] = -1;
                }


                if (!rightEyeState) {
                    String lmissedLights = Application.persistentDataPath +
"/lmissedLights.txt";
                    File.AppendAllText(lmissedLights, "lightNo = " + triggerIndex +
"trigger value = " + trigger[triggerIndex] + "\n");
                }
                else
                {
                    String rmissedLights = Application.persistentDataPath +
"/rmissedLights.txt";
```

```
                    File.AppendAllText(rmissedLights, "lightNo = " + triggerIndex +
"trigger value = " + trigger[triggerIndex] + "\n");
                }


                temp = theLights[rand];
                theLights[rand] = theLights[i];
                 theLights[i] = temp;

                if (!rightEyeState)
                {
                    lmissedLightsList.Add(triggerIndex);
                }
                else
                {
                    rmissedLightsList.Add(triggerIndex);
                }

            }
            else if(trigger[triggerIndex] == 1)
            {
                temp = theLights[rand];
                theLights[rand] = theLights[i];

                theLights[i] = temp;


            }


            String lightsRandFile = Application.persistentDataPath +
"/randlightNumber.txt";
            File.AppendAllText(lightsRandFile, "lightNo = " + Int32.Parse(trueLightNo)
+ "\n");
            yield return new WaitForSeconds(offsettime);
        }


        int m = theLights2.Count;

        for (int j = m - 1; j >= 0; j--)
        {


            // Get Random number from (0,i)
            int randIndex = random.Next(0, j+1);

            theLights2[randIndex].GetComponent<Light>().enabled = true;
            theLights2[randIndex].GetComponent<Light>().intensity = 2;
            theLights2[randIndex].GetComponent<Light>().bounceIntensity = 5;
            theLights2[randIndex].GetComponent<Light>().range = 15;
            theLights2[randIndex].GetComponent<Light>().type = LightType.Spot;
            theLights2[randIndex].GetComponent<Light>().spotAngle = 1;

            currentLight = theLights2[randIndex];
            trueLightNo = theLights2[randIndex].name;

            int triggerIndex = Int32.Parse(trueLightNo);
```

```
            yield return new WaitForSeconds(offsettime);
            theLights2[randIndex].GetComponent<Light>().enabled = false;
            theLights2[randIndex].GetComponent<Light>().intensity = 0;

            yield return new WaitForSeconds(1);


            if (GvrViewer.Instance.Triggered == true)
            {

                trigger[triggerIndex] = 1;

            }

            yield return new WaitForSeconds(2);

            //if trigger matrix is still 0 then change to -1

            lightNo = triggerIndex;

            if (trigger[triggerIndex] == -1)
            {
                if (!rightEyeState)
                {
                    String secondmissedLightsL = Application.persistentDataPath +
"/lsecondmissedLights.txt";
                    File.AppendAllText(secondmissedLightsL, "lightNo = " +
triggerIndex + "trigger value = " + trigger[triggerIndex] + "\n");
                    lsecondmissedLightsList.Add(triggerIndex);
                }

                else
                {
                    String secondmissedLightsR = Application.persistentDataPath +
"/rsecondmissedLights.txt";
                    File.AppendAllText(secondmissedLightsR, "lightNo = " +
triggerIndex + "trigger value = " + trigger[triggerIndex] + "\n");
                    rsecondmissedLightsList.Add(triggerIndex);
                }

            }

            temp = theLights2[randIndex];

            theLights2[randIndex] = theLights2[j];

            theLights2[j] = temp;

            String lightsRandFile = Application.persistentDataPath +
"/secondrandlightNumber.txt";
            File.AppendAllText(lightsRandFile, "lightNo = " + Int32.Parse(trueLightNo)
+ "\n");
            yield return new WaitForSeconds(offsettime);
        }

        leftEyeState = true;
```

```
        if (rightEyeState)
        {
            completeState = true;
        }

        if (!rightEyeState)
        {
            StartCoroutine(Blink());
            rightEyeState = true;
            theLights2.Clear();
        }

        /*  for (int i = secondmissedLightsList.Count - 1; i >= 0; i--)
          {
              //theLights[i].GetComponent<Renderer>().material = LightON;
              // print(theLights.Count);
              // Get Random number from (0,i)
              int randIndex = random.Next(0, secondmissedLightsList.Count);

theLights[secondmissedLightsList[randIndex]].GetComponent<Light>().enabled = true;

theLights[secondmissedLightsList[randIndex]].GetComponent<Light>().intensity = 3;

theLights[secondmissedLightsList[randIndex]].GetComponent<Light>().bounceIntensity =
3;
              theLights[secondmissedLightsList[randIndex]].GetComponent<Light>().range
= 15;
              // print(theLights[i].GetComponent<Light>().transform.position);
              theLights[secondmissedLightsList[randIndex]].GetComponent<Light>().type
= LightType.Spot;

theLights[secondmissedLightsList[randIndex]].GetComponent<Light>().spotAngle = 5;
              trueLightNo = theLights[secondmissedLightsList[randIndex]].ToString();

              int triggerIndex = (int)Char.GetNumericValue(trueLightNo.ElementAt(0));

              if (GvrViewer.Instance.Triggered == true)
              {

                  trigger[triggerIndex] = 1;
                  //       string appendText = "This is light no " +
trueLightNo.ElementAt(0) + "Trigger value = " + triggerIndex + "\n";
                  //       File.AppendAllText(fileName4, appendText);

              }
              //initialize trigger
              // trigger[rand] = 0;

              //check light intensity matrix

              // print("Hello");
              yield return new WaitForSeconds(offsettime);
              //theLights[i].GetComponent<Renderer>().material = LightOFF;
              //print("Hello1");

theLights[secondmissedLightsList[randIndex]].GetComponent<Light>().enabled = false;
```

```
theLights[secondmissedLightsList[randIndex]].GetComponent<Light>().intensity = 0;

              //if trigger matrix is still 0 then change to -1

              //lightNo = triggerIndex;
              Light temp =
theLights[secondmissedLightsList[randIndex]].GetComponent<Light>();
              //   Debug.Log("Light check = " + theLights[rand]);


              // assign trigger value?


              theLights[secondmissedLightsList[randIndex]] = theLights[i];
              //  trigger[rand] = trigger[i];

              //  Debug.Log(trueLightNo.ElementAt(0));
              //   Debug.Log(theLights[rand] + " | " + theLights[i]);
              theLights[i] = theLights[secondmissedLightsList[randIndex]];
              //   trigger[i] = trigger[rand];
              //    Debug.Log("2: " + theLights[rand] + " | " + theLights[i]);

              String lightsRandFile = Application.persistentDataPath +
"/thirdrandlightNumber.txt";
              File.AppendAllText(lightsRandFile, "lightNo = " +
trueLightNo.ElementAt(0) + "\n");
              //Debug.Log(trigger[triggerIndex]);
              yield return new WaitForSeconds(offsettime);
              if (trigger[triggerIndex] == -1)
              {
                  trigger[triggerIndex] = -1;
                  String thirdmissedLights = Application.persistentDataPath +
"/thirdmissedLights.txt";
                  File.AppendAllText(thirdmissedLights, "lightNo = " + triggerIndex +
"trigger value = " + trigger[triggerIndex] + "\n");


                  thirdmissedLightsList.Add(triggerIndex);


              }

          }*/



    }


}
```

APPENDIX B. EXCEL DATA RESULTS

Data results for 3 x 3 light pattern

x	=	-1	y=	-1	LightNo=	0
x	=	-1	y=	0	LightNo=	1
x	=	-1	y=	1	LightNo=	2
x	=	0	y=	-1	LightNo=	3
x	=	0	y=	0	LightNo=	4
x	=	0	y=	1	LightNo=	5
x	=	1	y=	-1	LightNo=	6
x	=	1	y=	0	LightNo=	7
x	=	1	y=	1	LightNo=	8

Light	Number:	7	1	TRUE	1
Light	Number:	5	1	TRUE	1
Light	Number:	8	1	FALSE	-1
Light	Number:	4	1	TRUE	1
Light	Number:	3	1	TRUE	1
Light	Number:	1	1	TRUE	1
Light	Number:	0	1	FALSE	-1
				TRUE	1
				TRUE	1
Light	Number:	6	1	TRUE	1
Light	Number:	5	1	TRUE	1
Light	Number:	8	1	TRUE	1
Light	Number:	4	1	FALSE	-1
Light	Number:	0	1	TRUE	1
Light	Number:	2	1	TRUE	1
Light	Number:	7	1	TRUE	1
Light	Number:	1	1	TRUE	1
				TRUE	1

Data results for 7 x 7 light pattern

x	=	-3	y=	-3	LightNo=	0
x	=	-3	y=	-2	LightNo=	1
x	=	-3	y=	-1	LightNo=	2
x	=	-3	y=	0	LightNo=	3
x	=	-3	y=	1	LightNo=	4
x	=	-3	y=	2	LightNo=	5
x	=	-3	y=	3	LightNo=	6
x	=	-2	y=	-3	LightNo=	7
x	=	-2	y=	-2	LightNo=	8
x	=	-2	y=	-1	LightNo=	9
x	=	-2	y=	0	LightNo=	10
x	=	-2	y=	1	LightNo=	11
x	=	-2	y=	2	LightNo=	12
x	=	-2	y=	3	LightNo=	13
x	=	-1	y=	-3	LightNo=	14
x	=	-1	y=	-2	LightNo=	15
x	=	-1	y=	-1	LightNo=	16
x	=	-1	y=	0	LightNo=	17
x	=	-1	y=	1	LightNo=	18
x	=	-1	y=	2	LightNo=	19
x	=	-1	y=	3	LightNo=	20
x	=	0	y=	-3	LightNo=	21
x	=	0	y=	-2	LightNo=	22
x	=	0	y=	-1	LightNo=	23

x	=	0	y=	0	LightNo=	24
x	=	0	y=	1	LightNo=	25
x	=	0	y=	2	LightNo=	26
x	=	0	y=	3	LightNo=	27
x	=	1	y=	-3	LightNo=	28
x	=	1	y=	-2	LightNo=	29
x	=	1	y=	-1	LightNo=	30
x	=	1	y=	0	LightNo=	31
x	=	1	y=	1	LightNo=	32
x	=	1	y=	2	LightNo=	33
x	=	1	y=	3	LightNo=	34
x	=	2	y=	-3	LightNo=	35
x	=	2	y=	-2	LightNo=	36
x	=	2	y=	-1	LightNo=	37
x	=	2	y=	0	LightNo=	38
x	=	2	y=	1	LightNo=	39
x	=	2	y=	2	LightNo=	40
x	=	2	y=	3	LightNo=	41
x	=	3	y=	-3	LightNo=	42
x	=	3	y=	-2	LightNo=	43
x	=	3	y=	-1	LightNo=	44
x	=	3	y=	0	LightNo=	45
x	=	3	y=	1	LightNo=	46
x	=	3	y=	2	LightNo=	47
x	=	3	y=	3	LightNo=	48

Light	Number:	20	1	TRUE	1
Light	Number:	27	1	TRUE	1
Light	Number:	21	1	TRUE	1
Light	Number:	15	1	TRUE	1
Light	Number:	32	1	TRUE	1
Light	Number:	48	1	TRUE	1
Light	Number:	13	1	TRUE	1
Light	Number:	3	1	TRUE	1
Light	Number:	18	1	TRUE	1
Light	Number:	11	1	TRUE	1
Light	Number:	9	1	TRUE	1
Light	Number:	30	1	TRUE	1
Light	Number:	36	1	TRUE	1
Light	Number:	8	1	TRUE	1
Light	Number:	42	1	TRUE	1
Light	Number:	41	1	TRUE	1

Light	Number:	23	1	TRUE	1
Light	Number:	12	1	TRUE	1
Light	Number:	22	1	TRUE	1
Light	Number:	37	1	TRUE	1
Light	Number:	39	1	TRUE	1
Light	Number:	47	1	TRUE	1
Light	Number:	0	1	TRUE	1
Light	Number:	5	1	TRUE	1
Light	Number:	16	1	TRUE	1
Light	Number:	38	1	TRUE	1
Light	Number:	43	1	TRUE	1
Light	Number:	26	1	TRUE	1
Light	Number:	24	1	TRUE	1
Light	Number:	7	1	TRUE	1
Light	Number:	40	1	TRUE	1
Light	Number:	25	1	FALSE	-1
Light	Number:	2	1	TRUE	1
Light	Number:	35	1	TRUE	1
Light	Number:	28	1	TRUE	1
Light	Number:	44	1	TRUE	1
Light	Number:	1	1	TRUE	1
Light	Number:	14	1	TRUE	1
Light	Number:	4	1	TRUE	1
Light	Number:	17	1	TRUE	1
Light	Number:	6	1	TRUE	1
Light	Number:	29	1	TRUE	1
Light	Number:	33	1	TRUE	1
Light	Number:	19	1	TRUE	1
Light	Number:	34	1	TRUE	1
Light	Number:	10	1	FALSE	-1
				FALSE	-1
				TRUE	1
				TRUE	1
Light	Number:	2	1	TRUE	1
Light	Number:	7	1	TRUE	1
Light	Number:	23	1	TRUE	1
Light	Number:	6	1	FALSE	-1
Light	Number:	31	1	TRUE	1
Light	Number:	44	1	TRUE	1
Light	Number:	40	1	TRUE	1
Light	Number:	25	1	TRUE	1

Light	Number:	38	1	TRUE	1
Light	Number:	24	1	TRUE	1
Light	Number:	45	1	TRUE	1
Light	Number:	12	1	TRUE	1
Light	Number:	15	1	TRUE	1
Light	Number:	4	1	TRUE	1
Light	Number:	9	1	TRUE	1
Light	Number:	37	1	TRUE	1
Light	Number:	5	1	TRUE	1
Light	Number:	13	1	TRUE	1
Light	Number:	48	1	TRUE	1
Light	Number:	19	1	TRUE	1
Light	Number:	1	1	TRUE	1
Light	Number:	17	1	TRUE	1
Light	Number:	39	1	TRUE	1
Light	Number:	16	1	TRUE	1
Light	Number:	30	1	TRUE	1
Light	Number:	28	1	TRUE	1
Light	Number:	10	1	TRUE	1
Light	Number:	34	1	TRUE	1
Light	Number:	47	1	TRUE	1
Light	Number:	41	1	TRUE	1
Light	Number:	26	1	TRUE	1
Light	Number:	11	1	TRUE	1
Light	Number:	42	1	TRUE	1
Light	Number:	22	1	TRUE	1
Light	Number:	33	1	TRUE	1
Light	Number:	29	1	TRUE	1
Light	Number:	8	1	TRUE	1
Light	Number:	0	1	TRUE	1
Light	Number:	35	1	TRUE	1
Light	Number:	18	1	TRUE	1
Light	Number:	27	1	TRUE	1
Light	Number:	20	1	TRUE	1
Light	Number:	43	1	TRUE	1
Light	Number:	14	1	TRUE	1
Light	Number:	36	1	TRUE	1
Light	Number:	21	1	TRUE	1
Light	Number:	46	1	TRUE	1
Light	Number:	32	1	TRUE	1
				TRUE	1