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

STRIVE-VR: Strabismus Training Routines Inducing Vision Enhancements using Virtual Reality

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

Submitted by:

Romeo C. Valdez Jr.

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 "STRIVE-VR: Strabismus Training Routines Inducing Vision Enhancements using Virtual Reality" prepared and submitted by Romeo C. Valdez Jr. in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Perlita E. Gasmen, M.S.
Adviser

EXAMINERS:

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

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

Ma. Sheila A. Magboo, M.S.
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

Strabismus or "crossed eyes" is the misalignment of the eyes wherein each eye may go in different directions. Various treatment procedures are being offered to correct strabismus. Several traditional vision training techniques are considered to cure this kind of eye condition. However, people find these vision training techniques to be uninteresting and boring because of lack in user engagement. Because of this, serious games are used to promote user engagement, interest, and active participation among subjects towards vision therapies. A virtual reality application can be used to implement serious games for treating strabismus.

STRIVE-VR is a virtual reality application that provides vision training techniques incorporated with gamification with in-game objectives to be performed by the subject with strabismus.

Keywords: Strabismus, Vision Training Techniques, Serious Games, Virtual Reality, Gamification

Contents

Acceptance Sheet i

Abstract ii

List of Figures v

I. Introduction 1

A. Background of the Study . . . 1

B. Statement of the Problem . . . 7

C. Objectives of the Study . . . 7

D. Significance of the Project . . . 8

E. Scope and Limitations . . . 9

F. Assumptions . . . 10

II. Review of Related Literature 11

III. Theoretical Framework 18

A. Strabismus . . . 18

B. Vision Therapy in Strabismus . . . 19

C. Tondel Arrows . . . 19

D. Lazy Eight . . . 20

E. Brock String . . . 21

F. Virtual Reality . . . 22

F..1 VR Box . . . 22

F..2 VR Box Controller . . . 22

F..3 Android . . . 23

F..4 Google VR SDK . . . 23

G. Blender . . . 23

H. Unity . . . 23

IV. Design and Implementation 24

A. Use Case Diagram 24
B. Context Case Diagram 25
C. Activity Diagram 26
D. Flowchart 27
E. Technical Architecture 30

V. Results **31**
A. Main Menu 31
A..1 Parameters 33
A..2 Tutorials 36
B. Virtual Environment 40
B..1 Tondell Arrows Technique 41
B..2 Lazy Eight Technique 42
B..3 Brock String Technique 45
C. Force Stop 47
D. Post Game Screen 48

VI. Discussion **50**

VII. Conclusions **52**

VIII. Recommendations **53**

IX. Bibliography **54**

X. Appendix **59**

XI. Acknowledgement **82**

List of Figures

1 Exotropia . . . 2
2 Esotropia . . . 2
3 Hypotropia . . . 2
4 Hypertropia . . . 2
5 Diplopia brick game preview . . . 11
6 Tondel Technique . . . 12
7 Lazy Eight Technique . . . 12
8 Brock String Methodology . . . 13
9 Squashing virtual bugs preview (left) and its schematic diagram (right) . . . 13
10 Low-cost VR car racing game for amblyopia rehabilitation . . . 15
11 VR Dichoptic training for amblyopic patients . . . 16
12 Patient using VEHuNT for wayfinding task . . . 17
13 Tondel Arrows procedure . . . 20
14 Lazy eight technique . . . 21
15 Brock String methodology . . . 22
16 Overview of Use Case Diagram . . . 24
17 Overview of Context Case Diagram . . . 25
18 Overview of Activity Diagram . . . 26
19 Flowchart of patient for the Tondel Arrows procedure . . . 27
20 Flowchart of patient for the Lazy Eight technique . . . 28
21 Flowchart of patient for the Brock String methodology . . . 29
22 STRIVE-VR Main Menu Screen . . . 31
23 Choosing a vision training technique to play or to learn about . . . 32
24 Arrow indicating that the correct direction is to the left of the player 32
25 Arrow indicating that the correct direction is to the right of the player . . . 32
26 Tondell Arrows technique: Choosing no. of pairs of arrows . . . 33

27 Tondell Arrows technique: Choosing difficulty of the game 34
28 Lazy Eight technique: Choosing no. of infinity symbols to be traced 34
29 Brock String technique: Choosing no. of beads to be gazed at . . . 35
30 Brock String technique: Choosing difficulty of the game 35
31 Choosing whether or not to have a voice guide for the objectives . . 36
32 First section of the Tondell Arrows technique 36
33 Second section of the Tondell Arrows technique 37
34 Third section of the Tondell Arrows technique 37
35 Fourth section of the Tondell Arrows technique 37
36 First section of the Lazy Eight technique 38
37 Second section of the Lazy Eight technique 38
38 Third section of the Lazy Eight technique 38
39 Fourth section of the Lazy Eight technique 39
40 First section of the Brock String technique 39
41 Second section of the Brock String technique 39
42 Third section of the Brock String technique 40
43 Fourth section of the Brock String technique 40
44 First scene from the Tondell Arrows technique 41
45 Second scene from the Tondell Arrows technique 42
46 First scene from the Lazy Eight technique 42
47 Speed multiplier controls for Lazy Eight technique 43
48 Second scene from the Lazy Eight technique 43
49 Third scene from the Lazy Eight technique 44
50 First scene from the Brock String technique 45
51 Speed multiplier controls for Brock String technique 45
52 Second scene from the Brock String technique 46
53 Controls for adjusting bead 46
54 Controls for declaring adjustment of bead 46
55 Third scene from the Brock String technique 47

56 Controls for force stopping a game 48
57 Post game screen for Tondell Arrows technique 48
58 Post game screen for Lazy Eight technique 49
59 Post game screen for Brock String technique 49

I. Introduction

A. Background of the Study

Strabismus, or more commonly known as "crossed eyes", is a type of eye disorder wherein the eyes are not aligned or are in different directions. This causes an individual's both eyes to be looking at different objects at the same time [1]. Significantly, strabismus is one of the leading cases of recorded genetic disorders among humans. It is the most common observed characteristics among people which, in average, is present in up to 5% of every studied population. Despite the assumption that most cases of eye misalignments often exist together with rare complex syndromes, majority of strabismus cases are non-syndromic. However, existing genes are identified to be associated with both syndromic and non-syndromic forms of strabismus. This means that strabismus may be passed from one generation to another. Syndromic strabismus occur with other abnormalities of the body, while non-syndromic strabismus may occur on its own without being associated with other symptoms [2]. Strabismus, which adversely affects the binocular vision, may also occur when visual stimuli are not addressed immediately during the critical period of development, specifically at a very young age [3].

People with strabismus, especially the young ones, are adversely affected by their condition. It does not only cause visual problems in the development stage, but it also results to various psychosocial problems which may negatively influence the subject's self-image, interpersonal relationships, and performance in school and employment [2].

The condition of strabismus among patients differ according to the direction that the strabismic eye has gone. Distorted alignment of the eyes affects one or both of the eyes and can occur in any direction. The strabismic eye can either be labeled as esotropic, exotropic, hypertropic, or hypotropic. From these types of strabismic conditions, the strabismic eye has either turned inward, outward, upward, or downward, respectively [3].

Figure 1: Exotropia

Figure 2: Esotropia

Figure 3: Hypotropia

Figure 4: Hypertropia

Moreover, a strabismic eye can be determined through the following screening procedures: (a) the Hirschberg test, (b) the cover test [4], (c) the Maddox rod methodology [5], and (d) genetic testing [2]. For the Hirschberg test, a light is shone through the eyes of the patient wherein the area around the cornea, where the light has reflected off, is observed and will become the basis for the alignment of the eyes. On the other hand, the cover test is done by covering each eye alternately to determine which eye is strabismic. Meanwhile, in the Maddox rod method, the patient will be asked to look directly at a light source. After some time, the Maddox rod will be alternatively placed over the eyes wherein a red line and a white light can be seen by the patient. Based on the positions of the red line and the white light, the presence or absence of strabismus will be determined [5]. Lastly, the genetic testing. This methodology can detect possible risks of strabismus so that treatment could be done earlier [2].

If strabismus is actually present in a subject, various treatment procedures exist for curing strabismus eye disorder. One treatment procedure for strabismus is through surgical means. According to a study led by the Nikookari Ophthalmology Hospital of Tabriz in Iran, the cost of a single strabismus surgery was estimated to be $464 which may still vary depending on the severity of the condition and the hospital that will render the service [6]. To support this, a study

held in Canada showed that a total cost of $1632 for a strabismus surgery [2]. Additionally, in the Philippines, according to Tess Yambot of the Vision Therapy Manila, the cost of strabismus surgery can range from Php50,000 to Php150,000 depending on the case of the patient. Furthermore, strabismus surgery is mainly used to correct misalignment of the eyes, to keep their proper alignment with free ocular movement [7], and for eye muscles to compensate from the difficulty with the coordination of the eye and the brain [8]. On the other hand, strabismus surgery has its disadvantages too. It can be the main cause of ocular misalignment instead of its correction [7], and there is still no known standard protocol for the surgical style of the operation. Strabismus surgery is also subject to the recurrence of strabismus which has been reported to be the most common problem after strabismus surgeries. According to previous studies, incidence reports of undercorrection and recurrence of strabismus varied from 20% to 59% depending on the case of strabismus as well as the age of the patient [9]. Frequently, strabismus surgery is cosmetic in nature wherein despite the correct alignment of the eyes, the information coming from them are still not combined properly by the brain to produce a single image which, in turn, may lead to subsequent surgeries [8]. Moreover, genetic testing, a type of strabismus screening procedure, is relevant for strabismus surgery because this screening method allows earlier treatment of strabismus which may result to decreased probability or frequency of having a surgery [2].

Another form of treatment procedure is the botulinum toxin injection which is an alternative for strabismus surgery. Compared to strabismus surgical procedure, the botulinum toxin injection takes shorter time which means that overall anesthesia exposure of the patient is reduced, and it results to only a minimal scarring of the extraocular muscles as well as the tissues surrounding them [10]. Also, an additional advantage of the botulinum toxin injection is that its cost is lower than strabismus surgery [11]. However, the muscle response for this treatment procedure is more variable than in strabismus surgery, the toxin spreads

repeatedly to the surrounding muscles which leads to transient vertical deviation and ptosis, and is only considered to be a temporary treatment for strabismus [12]. Aside from that, high recurrence rate of strabismus has been also observed after botulinum toxin injection which means that subsequent injections are necessary to prevent the recurrence. Although this treatment procedure has similar success rates statistically with the strabismus surgery, unfortunately, there is still no existing standardized botulinum toxin dose recommendations for all of the known strabismus cases [10].

Aside from strabismus surgery and botulinum toxin injection, there are still other treatment procedures for strabismus. These are the various vision therapies, involving the eyes and the brain [3], which are supervised by doctors and are considered as non-surgical procedures [13]. Different set of tools and devices are needed for these vision therapies which are beneficial for the patient with strabismus. These are used for training, recovery, and development of the binocular vision skills [3]. Significantly, the important goal of vision therapies is to unify the two parts of the visual system, namely the motor visual system which is accountable for the aiming accuracy of both eyes, and the sensory visual system which is responsible for combining two images into a single three dimensional image in the brain [14]. Additionally, vision therapies are used for the visual system to correct itself from its misalignment and insufficient visual depth perception [13]. However, some methods are not proven to work, and the cost for these vision therapies may still amount up to $152 per visit [8]. In the Philippines, according to Tess Yambot of the Vision Therapy Manila, at least Php1,500 is needed for a single session of strabismus vision therapy. Despite its expensiveness, according to the Vision Therapy Center, the people who have undergone vision therapies for strabismus have been totally cured [15]. In other cases, performing vision therapy after strabismus surgery is necessary to maintain the alignment of the eyes and to improve depth perception [8]. Moreover, other vision therapies include vision training techniques like the Tondel Arrows technique, the Lazy Eight technique

[3], and the Brock String methodology which all account for the goal of vision therapy [16].

The Tondel technique is a type of vision training technique which induces binocular improvements, clear stimulus accommodation, and avoidance of mixed color perceptions through convergence and near point convergence training. This technique uses a pyramidal form of a Tondel board, with 6 pairs of colored Tondel arrows imprinted on it, which is placed against the nose of the patient. The goal of this technique is for the patient to have a clear perception of every pair of arrows which will only be possible through convergence and presence of proper eye alignment [3].

Another type of vision training technique is the Lazy eight technique. This technique requires tracing of infinity symbols within infinity symbols, and focuses on the development and improvement of the following skills and abilities: (a) visual perception, (b) oculomotor skills, (c) motor integration of the visual function, (d) concentration and attention, (e) hand-eye coordination, (f) tracking and laterality, and (g) the working memory.

Lastly, the Brock string technique which establishes and improves convergence and accurate fixation skills of the subject. It uses a string with 5 differently colored beads. The objectives of this technique is for the subject to alternate fixation on the beads while taking note of the visual input and sensation of convergence.

All of the mentioned treatment procedures require operations that are performed manually with the guidance and assistance of the therapist. These methods need the aid of professional skills from health providers, and some involve expensive equipment with high amount of money. Because of that, needy people could be driven out since they could not afford to pay for the necessary treatment procedures [5].

In the present, our technology advances in a fast rate, and one of those is the virtual reality technology [17]. Rehabilitation practices being incorporated in virtual reality applications can pave way to an accessible and cost-effective treatment

procedures by just using mobile devices. Recovery times of the patients can also speed up since these medical techniques and methodologies are readily available [18]. In fact, unlike in traditional procedures, virtual reality provides experiences of immersive environments that can be accessed at a low cost. Reporting strategies through VR applications help in tracking user performance too [19].

Virtual reality can also be incorporated with serious games - computerized games implemented for serious purposes like rehabilitation. It is evident that gamification - gaming elements used outside of games, can be used for rehabilitation practices. Gamification provides access to online programs to those who may not use them, improves user engagement through game-based and serious motivational dynamics, and utilizes mechanisms for change in therapeutic processes and gaming features. Both serious games and gamification have positive implications towards education and motivation. In virtual reality therapies, gamification can be used to add gaming elements such as scoring, in-game rewards, and quest engagements. In this way, user engagement and motivation can be highly developed and improved towards therapeutic procedures [20]. Using virtual reality can be more engaging than the traditional procedures because people happen to be uninterested towards these vision therapies because they find them boring [19].

To illustrate, Diplopia, a virtual reality application [21] that specializes in training an amblyopic eye [22], is a virtual reality application that incorporates rehabilitation practices and treatment procedures for amblyopia [18] - an eye disorder which may result from strabismus, and is caused by lack of eye-brain coordination for the reason that the vision of one of the eyes has been reduced [23]. Having said that, it is evident that those previously mentioned treatment procedures for strabismus can be actually reflected, specifically through the virtual reality applications [24]. However, the application is currently available only in US and Canada [25].

B. Statement of the Problem

As of now, there is a lack of gamification in the traditional vision therapies for strabismus which is important for promoting user engagement, interest, and active participation among subjects toward these vision therapies. Additionally, these existing vision therapies have no virtual reality application counterparts yet which can be used as an adjunct to them.

C. Objectives of the Study

This research introduces a virtual reality application that allows user to perform strabismus vision training which will aid in the improvement of the eye conditions of the subject. The said application will possess the following functionalities:

1. Allows the vision therapist to:
    (a) Set up the parameters needed for the following vision training techniques:
        i. For the Tondel Arrows procedure:
            A. No. of pairs of arrows
                1. Minimum of 4 pairs
                2. Maximum of 6 pairs
            B. Difficulty (the harder, the faster rate of flashing lights)
                1. Easy
                2. Medium
                3. Hard
        ii. For the Lazy Eight technique:
            A. No. of checkpoints
                1. Minimum of 5 checkpoints
                2. Maximum of 10 checkpoints
        iii. For the Brock String methodology:

A. No. of beads

1. Minimum of 3 beads
2. Maximum of 5 beads

B. Difficulty (the harder, the faster rate of obstacles to appear)

1. Easy
2. Medium
3. Hard

2. Allows the patient to:

(a) Use the Bluetooth VR controller to perform some of the in-game instructions

(b) Perform the games corresponding for each vision training technique

i. Tondel Arrows procedure (Scenario: Room)

ii. Lazy Eight technique (Scenario: Open field)

iii. Brock String methodology (Scenario: Activity Area)

(c) Stop the training manually in case he/she feels discomfort or suppression in the eyes

(d) View the tutorials for the three vision training techniques

D. Significance of the Project

This virtual reality application will serve as an adjunct to strabismus vision training techniques, namely the Tondel Arrows procedure, the Lazy eight technique, and the Brock string methodology. Each vision training technique contributes various advantages for treating strabismus. For the Tondel Arrows procedure, it results to binocular improvements, clear stimulus accommodation, and avoidance of mixed color perceptions through convergence and near point convergence training. For the Lazy eight technique, it promotes development and improvement of visual perception, oculomotor skills, motor integration of the visual function,

concentration and attention, coordination, tracking and laterality, and working memory. Lastly, through the Brock string methodology, the establishment and improvement of the convergence and accurate fixation skills of the subject can be met. Gamification of these vision training techniques also induces user engagement, interest, active participation, and user creativity towards their objectives.

E. Scope and Limitations

1. Recording of performance of the virtual reality application results are part of a separate patient record system.

2. The VR application is to be done in the eye clinic under the supervision of the vision therapist.

3. The scenario specified in the virtual reality application is selected to incorporate gamification processes.

4. The virtual reality application will only be used in the eye clinic and not outside the facility.

5. The virtual reality application is made to provide vision training for strabismus and not for other eye disorders.

6. A smartphone with an Android OS version of 5.0 (Kitkat) or higher, gyroscope, and accelerometer is needed to run the application.

7. There will be no iOS version of the virtual reality application.

8. The application will be using a VR controller for in-game actions.

9. The colors of the game objects may vary per phone.

10. The Tondel Arrows procedure will be involving 4 to 6 pairs of arrows with 6 different corresponding colors which will be randomly lighted.

11. The Lazy Eight technique only includes 5 to 10 checkpoints wherein infinity symbol/s at each checkpoint must be traced first before the gameplay can proceed.

12. The Brock String methodology will have a set of 3 to 5 beads of different colors.

F. Assumptions

1. The therapist will provide the Android smartphone, the VR headset, and the bluetooth VR controller.

2. The VR controller must be connected to the smartphone via bluetooth.

II. Review of Related Literature

Virtual reality technologies contribute to the rehabilitation processes being used for treating various disorders. People can readily access different treatment procedures at a cost-efficient way wherein using their mobile devices. Using virtual reality for performing medical methodologies also increases recovery times of the patients [18]. Additionally, because of the technological advancement, the ways of learning, educating, and working have begun to change [26].

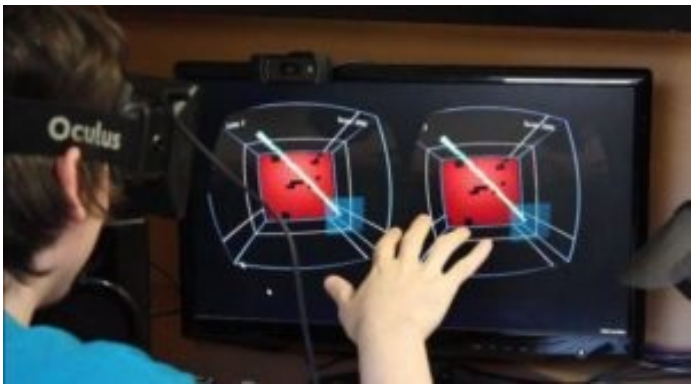

Figure 5: Diplopia brick game preview

Apollo VR company developed a virtual reality application called Diplopia which is used to assist people who have amblyopia in a way that the vision of their amblyopic eye will be restored [21] through rebalancing the visual input to both eyes [22]. The application provides different images to each eye which will force both eyes to work together to win the game. In the application, the user must clear the levels of bricks by using a gesture-controlled paddle to bounce the ball onto the wall. The paddle will only be shown to the normal eye with a highly dimmed brick wall. Meanwhile, on the weak eye, or the amblyopic eye, only the ball will be shown together with a very bright brick wall. In this way, both of the eyes will be used for the brain to understand the correct positions of the game objects. Also, the visual information coming from both eyes must be integrated by the brain too to form one coherent picture from them. This methodology ensures the user to develop depth perception in-game [21].

Figure 6: Tondel Technique

Barbu et. al. have carried out a visual training technique, called the Tondel technique, among preschool children, with and without strabismus, ages 4-6 years old. In this technique, a Tondel board, formed to a pyramidal shape with 6 pairs of colored arrows, is placed against the subject's nose. The subject should be able to clearly see every pair of arrows in order to develop proper eye alignment which will enable accurate perception of the arrows through convergence. The Tondel technique aims to provide the subjects with physiological diplopia awareness, convergence training, near point of convergence normalization, and convergence jumps exercises at different distances [3].

Figure 7: Lazy Eight Technique

Lazy Eight technique, another visual training technique used by Barbu et. al. in their experiment, requires the tracing of infinity symbol which allows patients to develop cerebral hemispheres communication and tracking skills, as well as to improve their visual perception, oculomotor skills, and hand-eye coordination [3].

Figure 8: Brock String Methodology

In baseball, since it is a vision-intensive sport, the batter must see the ball and recognize, process, and decide to swing the bat within a short period of time. Because of this, visual and cognitive systems must rapidly provide information for making a swing decision. Clark et. al. recruited 16 members of the University of Cincinnati intercollegiate baseball team for vision training programs, including Brock String to assess their positive effects towards the stereopsis and depth perception of the subjects [16] Brock string is also used for strabismus rehabilitation [27]. This methodology involves a 12-feet string with 5 differently colored, equally spaced, small wooden beads. An end of the string will be held against the tip of the nose of the subject and the other end to a fixed point. Each subject is asked to focus sight from one bead to another. This vision training will help the subjects to improve depth perception and fixation [16].

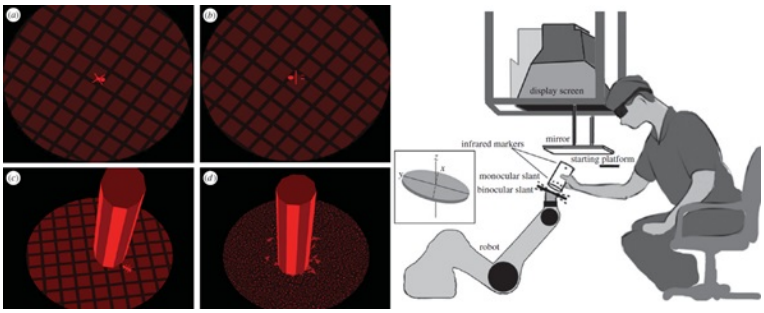

Figure 9: Squashing virtual bugs preview (left) and its schematic diagram (right)

Vedamurthy et. al. have developed a virtual reality application using a Crystal

Eyes shutter goggles for recovering stereo vision. The objective of the application is to use a Plexiglass cylinder to squash the small dichoptic virtual bugs, serving as fixation targets, which are rendered one-by-one on a disc-like slanted plane. The Plexiglass cylinder will have a corresponding virtual cylinder which will be rendered only as it moves within the workspace. The challenge in this application will be based on the slanted angles of the disc plane - the stimuli, which vary according to the suggested monocular and stereoscopic cues. These stimuli will also be rendered using regular tiled textures and randomly shaped dot textures. The regular tiled textures possess an effective monocular cues to slant in depth, while the randomly shaped dot textures do not. Significantly, in this manner, subjects will be able to improve the accuracy of their slant judgements for dot stimuli and textured stimuli [28].

A study led by Ahmed Awadein has used a computerized version of the Lancaster red-green test which is a subjective dissociated test for measuring misalignment of the eyes in different gaze positions. In the test, the patient will be needing to wear red-green goggles while seating 1-2 meters apart from the wall of rectilinear grid with drawn black dots. The objectives of this test are for the patient to determine (a) the total number of horizontal and vertical misalignment, and (b) the subjective torsion in various gaze directions being seen by the patient. In this manner, both eyes of the patient will be able to work together. This study has shown that computerized tests for vertical and torsional deviations produced significant results which are relatively good with respect to the results obtained from the traditional tests [29].

Nesaratnam et. al. have also conducted a study involving two dissociated tests for strabismus namely the virtual reality-based eye misalignment test and the conventional Lees screen test. The information gathered from these tests will be beneficial for screening and monitoring of patients with ocular misalignments. Unfortunately, traditional dissociated tests are subject to operator error which may rely solely to fixed head position. However, head-mounted displays reject

the need of head fixation which is why it is significant for this study to compare the results between the traditional tests and the virtual reality-based tests. In the study, patients have undergone both tests. For the virtual reality-based tests, the patients are subjected for horizontal and vertical measurement which were followed by torsion test. The results from both tests showed relatively the same positive impact among the patients which is why it has been concluded that virtual reality applications for screening ocular misalignment are possible alternative ways for dissociative test for strabismus [30].

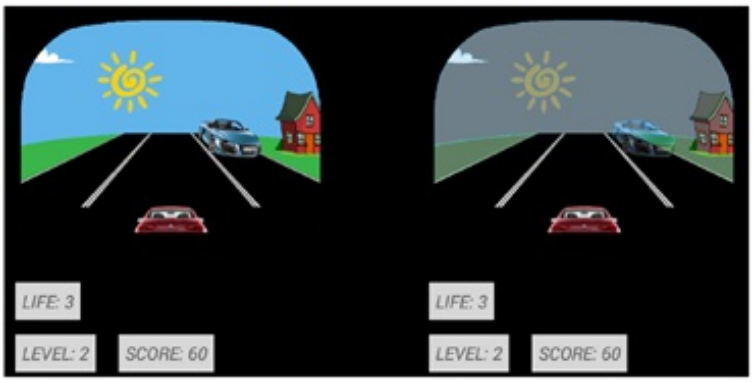

Figure 10: Low-cost VR car racing game for amblyopia rehabilitation

A low-cost virtual reality game for amblyopia rehabilitation is developed by the group of Gargantini et. al. The application uses Google Cardboard - its basis for being cost-effective. The system used a simple car racing game wherein different images are shown to the amblyopic eye and the normal eye. The amblyopic eye will be receiving all information regarding the game while the normal eye will only be getting a part of those. In this manner, the amblyopic eye will be more simulated which will encourage fusion among both eyes. As of now, the effectiveness of the application does not have clinical basis yet, however, several experiments are currently being conducted to validate the proposed approach [31].

I-BiT system is a computer-based interactive binocular treatment system that utilizes treatment procedures for amblyopia by using a 3D shutter glasses which lightens and darkens with respect to the monitor used. The system was developed through virtual reality to appeal and provide compliance to children and to

serve as an alternative for the traditional patching treatment. Herbison et. al. have conducted a study regarding I-BiT system to achieve results based on the treatment effects on visual acuity among amblyopic children. The treatment sessions held involved the use of a DVD to display video footages and a computer game called 'Nux'. Through I-BiT, both eyes of the patient are presented with images but some parts are only presented to the amblyopic eye. These images are distinct, however, visually related to each eye which promotes perception of dynamic, two-dimensional visual scene. The therapist can also adjust the settings in case the patient feels suppression or highly reduced visual acuity in their amblyopic eye. Significantly, I-BiT has been found to improve the quality of life (QoL) of amblyopic patients, and may also reduce the duration and cost of the needed treatments [32].

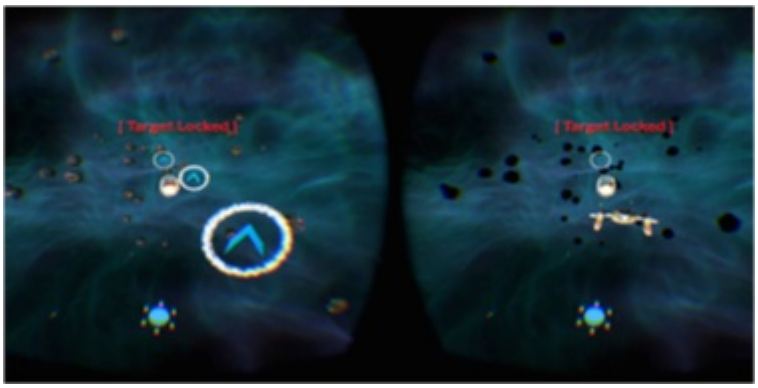

Figure 11: VR Dichoptic training for amblyopic patients

Ziak et. al. conducted a dichoptic training using virtual reality for amblyopia treatment of adults. The training used a virtual reality application that requires a head-mounted display to play the game. The training involves a space game wherein subjects have control on flying the spaceship through a system of rings. This differed from the low-cost virtual reality game mentioned a while ago in a sense that not all information is being fed to the amblyopic eye. This is because the subject can cheat in a way that they can just close their dominant eye since everything that is needed to see is already seen by the amblyopic eye. That is why the game fed different parts of the game for each eye in order for them to

work together in playing the game. The study showed significant results which provided potential usefulness of using virtual reality for dichoptic training [33].

Rectifeye, a vision-correcting system for virtual reality, was developed by Lafont et. al. for users who conventionally use corrective eyeglasses and are suffering from myopia, hyperopia, or astigmatism. The system does automatic adjusting of the virtual reality headset based on the eyeglasses prescription of the user. Because of this, users will no longer be needing to wear their eyeglasses in order to see clearly through the application. Rectifeye has the capability to correct myopia and hyperopia between -6D and 1D. In the application, images are shown to the user using ray-tracing and Gullstrand eye model which are used to simulate the human eye as well as to analyze various eye disorders like myopia, hyperopia, and astigmatism [34].

Figure 12: Patient using VEHuNT for wayfinding task

Virtual Environment Human Navigation Task, or VEHuNT, is an immersive VR application wherein patients with glaucoma can undergo a wayfinding procedure. Using VEHuNT, Daga et. al. investigated the behavior and spatial cognition of patients towards wayfinding. The application is significant because the ability of the patients with glaucoma to respond to various visual cues can be assessed. In that way, patients can get an idea regarding their surrounding environment. In this study, researchers have acquired information about the relationship of the vision loss of patients with glaucoma to their ability to perform everyday tasks associated with wayfinding, such as driving and walking [35].

III. Theoretical Framework

A. Strabismus

Strabismus, also known as cross-eyed or wall-eyed, is an ocular state wherein both eyes are abnormally aligned under normal situations [36]. It is a condition where binocularity is insufficient which results to poor integration of visual information from both eyes. Strabismus can be an outcome of lesions or seriously damaged eyes which negatively impacts the oculomotor, trochlear nerve, or higher neurological pathways. Additionally, strabismus can be caused, though rarely, by developmental or traumatic defects of the extraocular muscles [37]. Moreover, other strabismus-inducing situations involve the III, IV, and VI cranial nerves wherein they appear to be weak, paralyzed, or move involuntarily. In this case, the eye condition is referred to as paralytic strabismus which can either include third nerve palsy and superior oblique palsy [38]. Subsequently, when six different muscles around the eyes fail to work in coordination, strabismus may also transpire, and thus, making both eyes to focus on different objects. This makes one eye to look at one object while the other is turned to a different direction focusing on another object. As an outcome, the brain receives two different images from each eye which causes confusion to the brain [1]. Initially, the confusion of the brain may only produce double vision. However, as the condition lingers, the brain will decide to just ignore the image from the turned eye [39]. It should be taken note that strabismus affects to a stereoscopic depth wherein it involves the capacity for stereoscopic of 3 dimensional vision. Most importantly, patients with strabismus may not only have adverse effects on their personality traits and their employment opportunities [40], but also to their self-image, interpersonal relationships, and performance in school and work [2]. In fact, according to Tailor et. al., significant ocular misalignment can actually lead to unwanted effects towards the development, social interactions, and emotional well-being of the subject [37].

B. Vision Therapy in Strabismus

There exist different treatment procedures which can be used to cure strabismus. Some of these procedures are the strabismus surgery [7] and the botulinum toxin injection. However, because of their disadvantages, particularly the high rates of recurrence of strabismus and their lack of standard procedures [10], these two procedures have been concluded to be only effective for only a short term. However, despite those disadvantages, fortunately, alternative treatment procedures for strabismus, specifically the vision therapy, can be used which have been proven to be effective for a long term [12]. The goal of vision therapy is to unify the motor visual system which is accountable for the aiming accuracy of both eyes, and the sensory visual system which is responsible for combining two images into a single three dimensional image in the brain [14]. Some of these vision therapy include vision training techniques and eye exercises, namely the Tondel Arrows procedure, the Lazy Eight technique [3], and the Brock String methodology [16].

Usually, vision therapies are recommended by doctors when strabismus cannot be successfully treated solely with surgery or botulinum toxin injection. Vision therapies are done with the supervision of doctors, and are not considered as self-help vision improvement procedures. Even when prescribed by doctors, a home-based regimen of vision therapy cannot be counted as a complete program of vision therapy. According to a study published in Archives of Ophthalmology, home-based vision therapy was found to be ineffective, while doctor-supervised procedure was effective [13]. Vision therapies will only be concluded when significant improvements on the alignment of the eyes of the subject are observed and when the goal of the vision therapy has been met [14], [13].

C. Tondel Arrows

The Tondel technique involves Tondel boards that are specifically designed for treating binocular dysfunctions through visual therapy. These Tondel boards include Tondel arrows which are essential for conducting convergence and near point

of convergence training. These strabismus training procedures are means of getting clear stimulus for accommodation and avoidance of confusion or perception of mixed colors. The goals of this technique are: (a) for physiological diplopia or double vision awareness, (b) for training the convergence, (c) for normalizing near point of convergence, and (d) for exercising convergence jumps at different distances. The Tondel technique requires an A4 format of board containing printed 6 pairs of colored Tondel arrows are needed. The board is then folded along the central line producing a pyramidal shape making the arrow tips touch along a vertical central line. Afterwards, the resulting form of the board will be placed in front of the nose of the patient so that every pair of arrows can be seen and be clearly defined with the tips together. In order to test the effectiveness of this technique, correct perception of the arrows must be seen. If that is the case, then both eyes are in their proper alignment which is possible through convergence [3].

Figure 13: Tondel Arrows procedure

D. Lazy Eight

The Lazy eight technique is used for the development of integration and improvement of the following skills and abilities: (a) visual perception, (b) oculomotor skills, (c) motor integration of the visual function, (d) concentration and attention, (e) hand-eye coordination, (f) tracking and laterality, and (g) the working memory. This technique makes use of an infinity symbol which is based on the

behavioral optometry wherein it plays an important role in the development of visual perception, tracking skills, and communication of both cerebral hemispheres. Moreover, Bernell Company integrated numbers, letters, and other shapes in the lazy eight technique. In the technique, the patient will be tracing the outline using a marker. The technique will be starting from the center of the symbol towards the left of the smallest symbol, then to the right of the smallest symbol. Afterwards, the tracing must go back to the starting point wherein this time, the bigger symbol will be traced. The technique will just stop until the biggest symbol is completely traced [3].

Figure 14: Lazy eight technique

E. Brock String

The Brock string vision training technique is used to obtain and improve convergence skills and to disrupt suppression of the strabismic eye. Because of that, accurate fixation skills under binocular conditions can be met and further developed. This methodology makes use of a 12 feet string together with 5 small wooden beads of different colors. One end of the string will be held against the tip of the nose, and the other end will be fixated at one point. The 5 beads will be inserted and spaced up to the length of the string. The therapist will then ask the patient to alternate fixation. Afterwards, while the patient observes the visual input of each eye and the sensation of convergence, he/she must focus from one bead to the next bead, and so on. However, it will be dependent on the patient

whether or not he/she will make the procedure easier or more difficult by moving the beads closer to or further from his/her nose respectively [16].

Figure 15: Brock String methodology

F. Virtual Reality

Virtual reality refers to a three dimensional, computer generated environment. The user becomes part of this virtual world where he/she can do exploration, interaction, and manipulation of virtual elements, as well as perform series of actions [41].

F..1 VR Box

Virtual Reality Box is a virtual reality platform wherein a head mount is used together with the smartphone. Additionally, it is a Google Cardboard adoption of plastic body head-mounted display (HMD) in which the smartphone, either Android or iOS, is placed in order to play virtual reality games, applications, 3D videos, 360 immersive 3D videos, etc.

F..2 VR Box Controller

Virtual Reality Box controller is a wireless handset used together with the VR Box with the smartphone in it in order for the user to feel like using his/her own hands in the virtual world. It contains buttons that may be used to interact with virtual objects and perform actions within the virtual environment.

F..3 Android

Android is a Linux-based mobile operating system developed by Google. It is a widely-adopted open-source project. Google actively develops its platform and provides free access to some of its features to hardware manufacturers and phone carriers who want to use Android on their devices. Today, Android powers most of the phones, watches, and even car stereos.

F..4 Google VR SDK

Google VR SDK is used in order to create virtual reality applications available in Android and iOS mobile operating systems. It offers features necessary for virtual reality applications such as spatialized audio rendering, Daydream controller support, utilities, and samples.

G. Blender

Blender is a modeling software for three dimensional objects which is available online. It is also an open source tool which solely relies on affordable but efficient hardware such as the PlayStation Move controllers [41]. Having said that, Blender was able to and continues to provide a platform for the development of 2D and 3D contents, such as 3D visualizations, static images, quality videos for TV and cinema, and 3D interactive contents. These contents are produced with the use of Blender features like modeling, texturing, lighting, animation, and video post processing [21].

H. Unity

Unity 3D game engine is used for developing virtual reality applications. It also offers a multi-platform development feature which are available for various operating systems like iOS, Android, Microsoft Windows, and Linux [2].

IV. Design and Implementation

A. Use Case Diagram

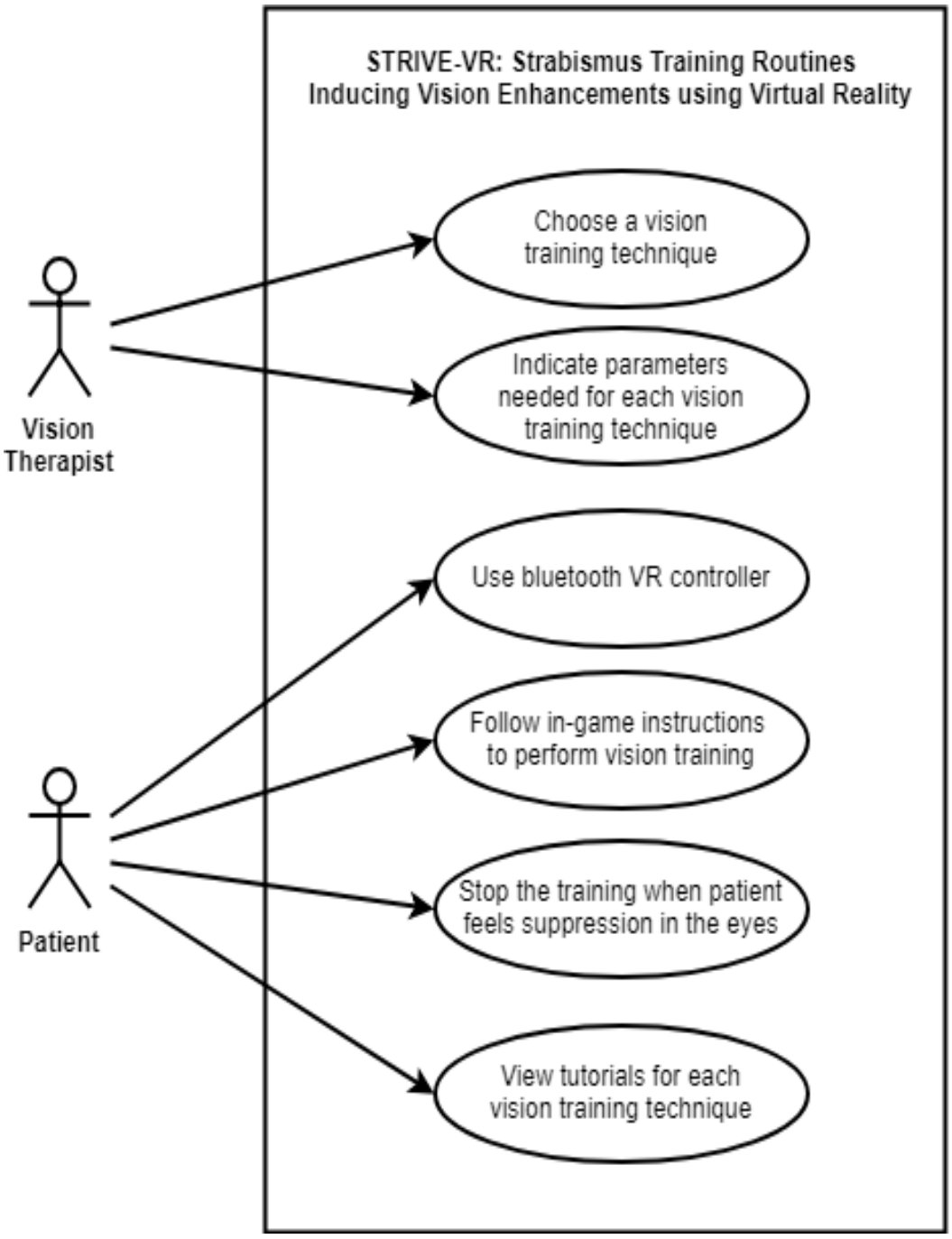

Figure 16: Overview of Use Case Diagram

The virtual reality application can be used both by the patient and the therapist. The vision therapist has the option to choose from different vision training

techniques with their corresponding parameters. The patient can use the bluetooth VR controller while performing the in-game objectives of the selected vision training technique. In case of discomfort or suppression in the eyes, the patient can force stop the training immediately by pressing a certain button on the bluetooth VR controller. Additionally, the patient can view tutorials for the three vision training techniques to know more the basics, controls, and the things to remember when performing the vision training techniques.

B. Context Case Diagram

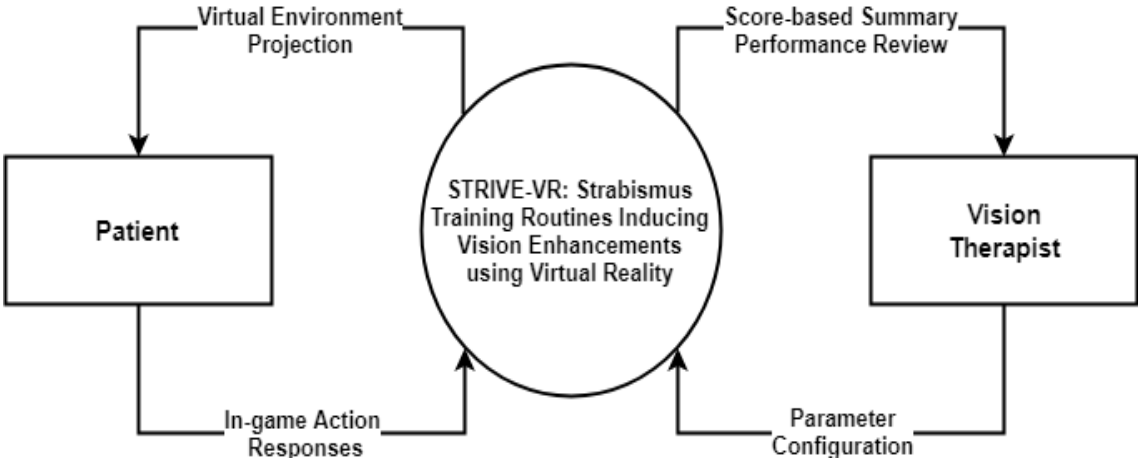

Figure 17: Overview of Context Case Diagram

The virtual reality application involves two users, namely the vision therapist and the patient. The therapist is the one that chooses the vision training technique with corresponding parameter that will be performed by the patient. Also, after the training of the patient, the therapist can view the score-based summary performance. Meanwhile, the patient is the one that uses the virtual reality application to perform the vision training. During each vision training technique, certain in-game objectives need input controls from the bluetooth VR controller which will be used by the patient.

C. Activity Diagram

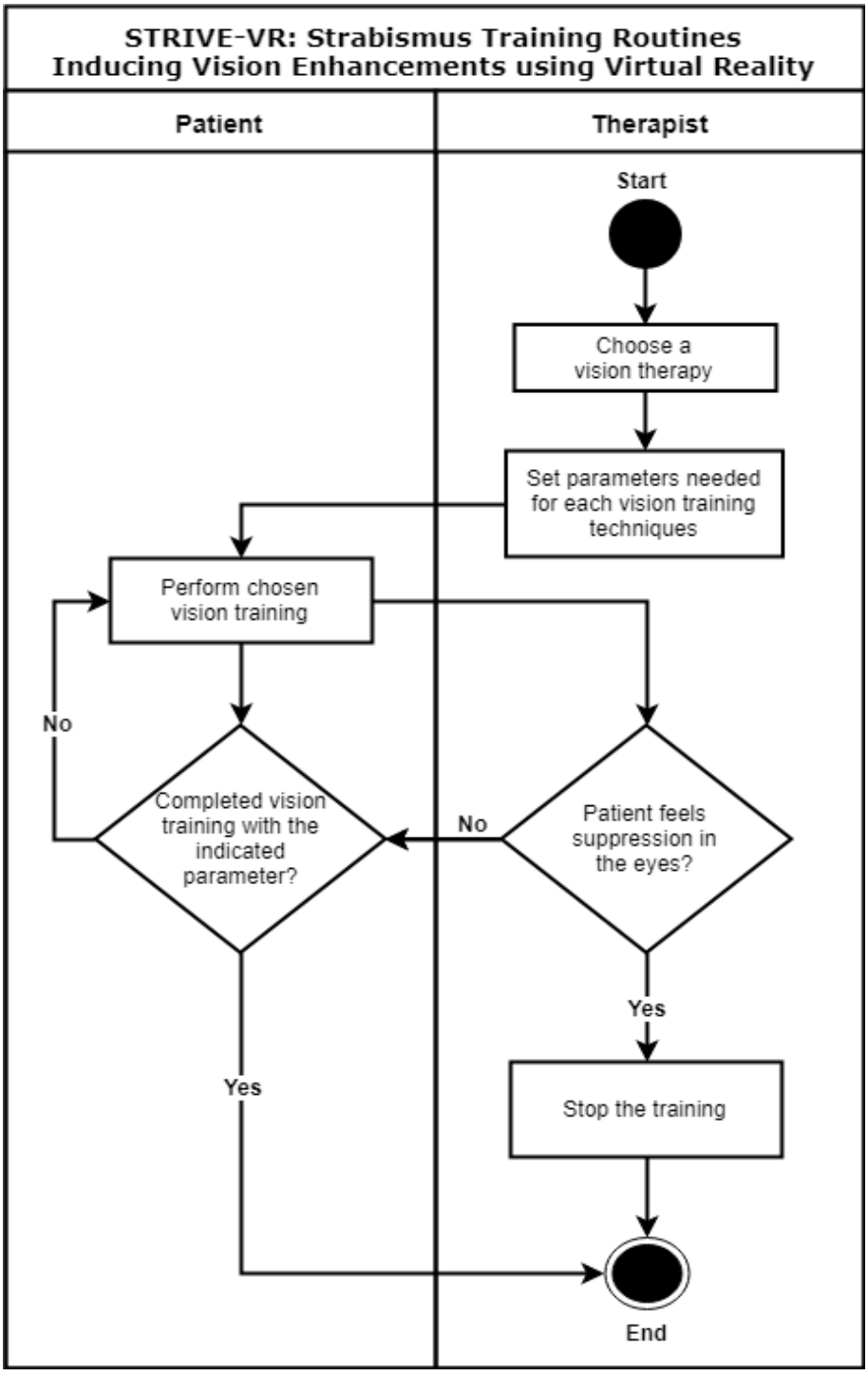

Figure 18: Overview of Activity Diagram

In the virtual reality application, the vision therapist is the one that chooses what vision training technique is to be performed by the patient. Afterwards, the therapist sets the corresponding parameters for the chosen vision training

technique. Then, the patient performs the selected vision training technique while using the bluetooth VR controller for in-game actions. The training will only end once the required parameters for the vision training technique are met, or when the patient chooses to force stop the training because of feeling of discomfort or suppression in the eyes.

D. Flowchart

The following reflects the scenarios of the patient in each virtual environment:

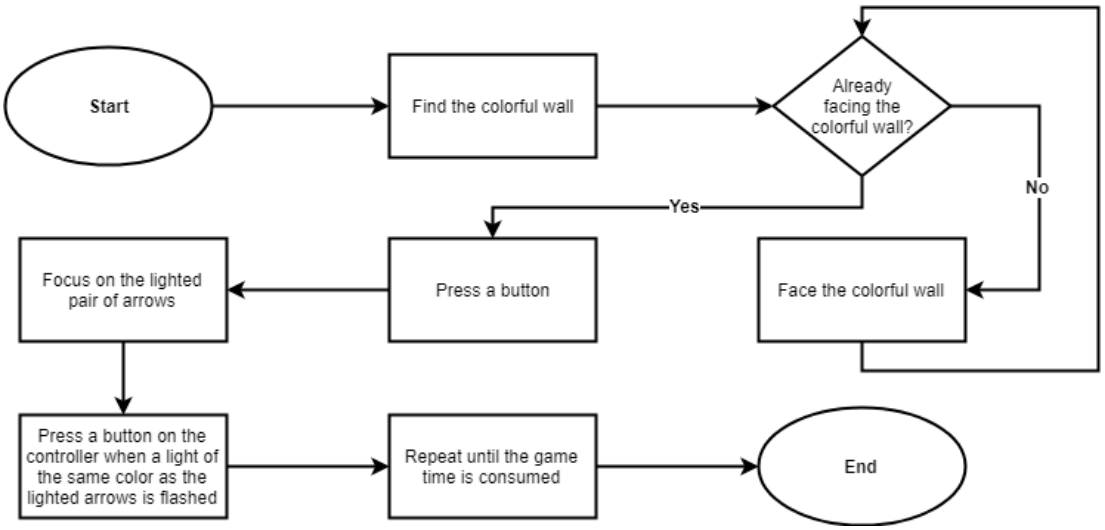

Figure 19: Flowchart of patient for the Tondel Arrows procedure

The flow of the Tondel Arrows procedure is indicated by Figure 19. This procedure is done in a room (virtual environment). Before the game starts, the patient must be able to locate the colorful wall first and press a buttpm afterwards. Then, the game starts wherein he/she must focus on the lighred pair of arrows. It should be taken note that a pair of arrows will be randomly lighted in this procedure. Afterwards, the subject must be able to detect a light that will be flashed on the screen. Once the flashed light is of the same color as that of the currently lighted pair of arrows, a button must be pressed on the VR controller for three seconds in order to gain a point. The procedure will just repeat until the 3-minute game

time is totally consumed.

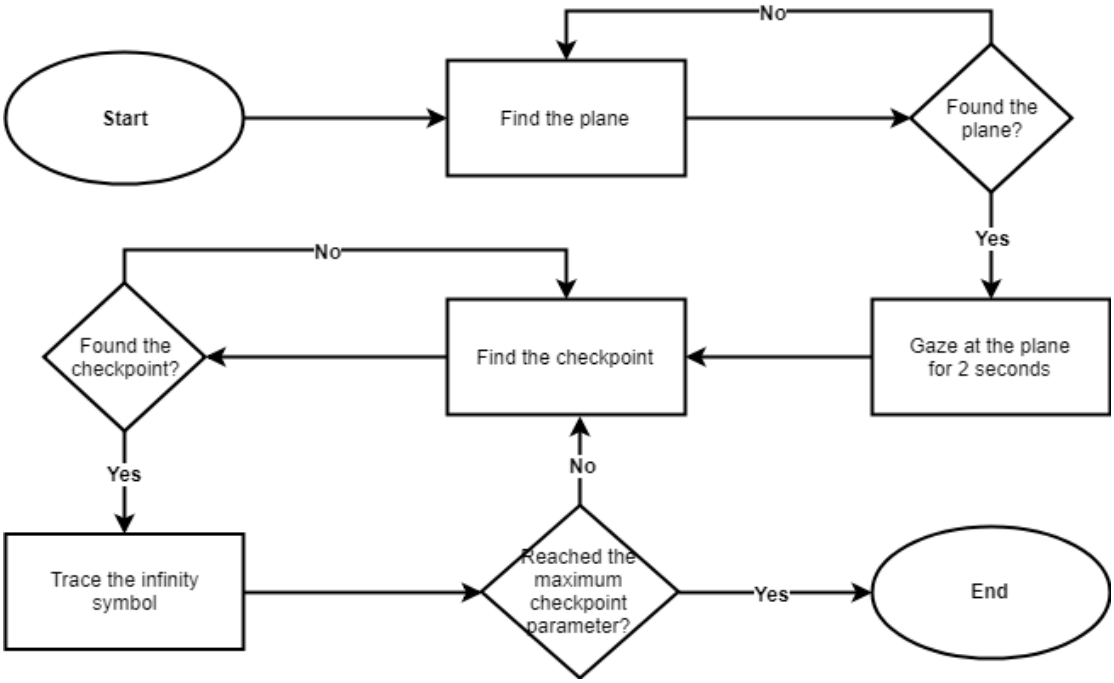

Figure 20: Flowchart of patient for the Lazy Eight technique

The Lazy Eight technique takes place in an open field (virtual environment). For this vision training technique, as shown in Figure 20, the subject must find first the plane located somewhere in the environment. Once the plane is found, the patient must gaze at it for 2 seconds to be able to ride it and be taken up in the air. Once this objective is successful, the game proper will start. The first objective is to locate the checkpoint. Upon reaching the checkpoint, the player will be halted. Then, he/she must gaze at the checkpoint for 2 seconds to trigger the infinity symbol to show up. This infinity symbol must be traced in order to continue playing. The number of infinity symbols to be traced at each checkpoint will be equivalent to the total number of passed checkpoints, e.g. if the subject is already at the 3rd checkpoint, then the number of infinity symbols to be traced will be 3. These steps will be repeated until the maximum checkpoint number is reached.

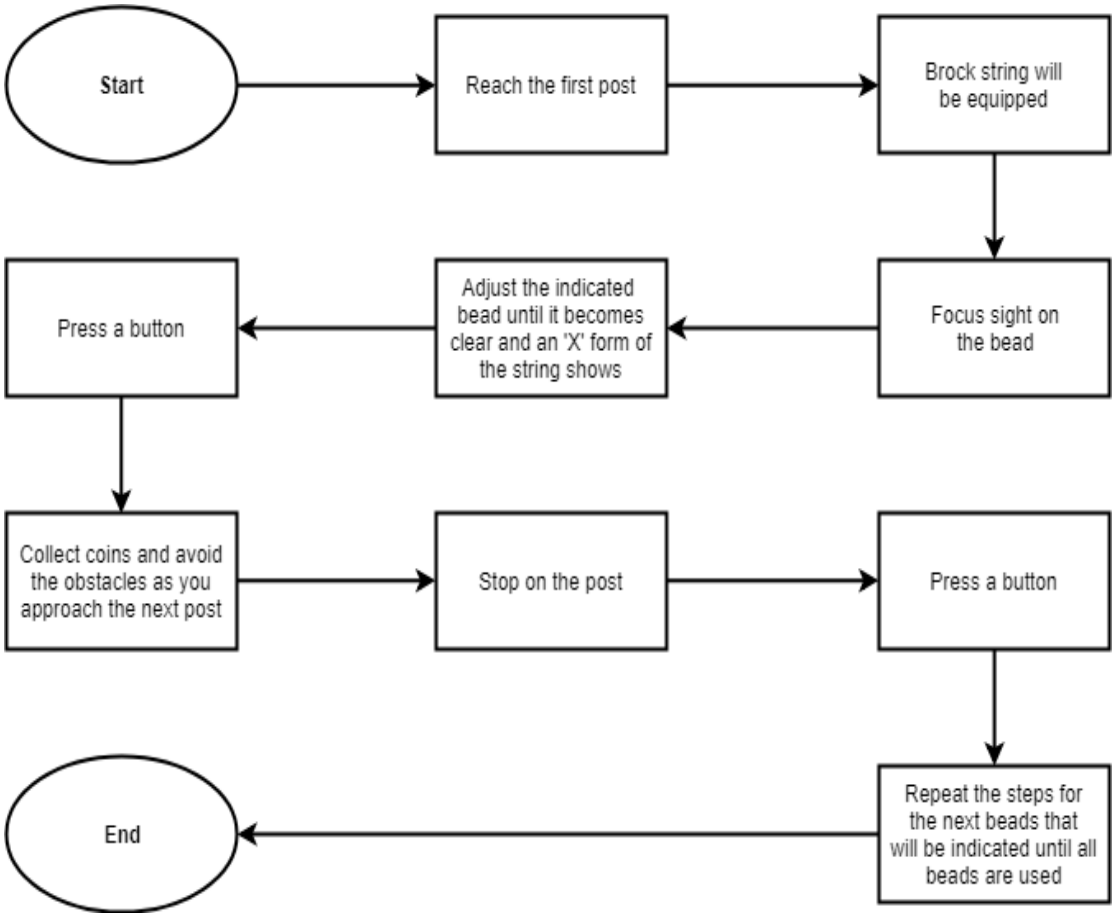

Figure 21: Flowchart of patient for the Brock String methodology

In Figure 21, the steps for the Brock String methodology are presented. This vision training technique takes place in an activity area (virtual environment). The subject must first walk towards the first post to get the brock string. Then the brock string will be displayed in front of the subject's visual field. Afterwards, the subject must gaze at the red bead. If the bead is still blurry, the user may opt to adjust the bead forward of backward using the VR controller. When the bead is finally clear to see, an 'X' figure of the string must be seen by the user. After the bead is successfully adjusted, a certain button on the VR controller must be pressed to continue the game. Then, obstacles will be approaching the user. The user must be able to avoid them until the next post is reached. A button must be pressed again to continue the game. The steps will be just repeated for the next beads that will be indicated, until all beads have been passed through.

E. Technical Architecture

The virtual reality application for strabismus treatment procedures requires a mobile device which has the following specifications:

1. Android OS version 5.0 (Kitkat) or higher
2. Gyroscope sensor
3. Accelerometer

V. Results

STRIVE-VR, or Strabismus Training Routines Inducing Vision Enhancements using Virtual Reality, is a virtual reality mobile application which allows patients with strabismus to perform the following vision training techniques: the Tondell Arrows technique, the Lazy Eight technique, and the Brock String technique which are respectively performed in a virtual room, open field, and activity area. Gamification has been applied to these vision training techniques to promote user engagement, interest, and active participation among patients. This means that gaming elements, such as scores, are involved when performing the objectives of each vision training technique.

A. Main Menu

The very first screen that appears upon starting the STRIVE-VR application is the main menu screen.

Figure 22: STRIVE-VR Main Menu Screen

At this point, the user has the option to select the “Play” button to choose and perform one of the vision training techniques, or select the “Tutorials” button to learn first the basics, objectives, and controls for each vision training technique.

Figure 23: Choosing a vision training technique to play or to learn about

If the player could not find the menu on the screen, it is probably outside the camera view. In this case, two arrows pointing to the left, as shown in Figure 24, and/or to the right, as shown in Figure 25, will appear on both sides of the screen to indicate where that gameobject is currently at. This also applies in game to guide the player to face the correct direction - the direction where where the game object, involved in the current objective, is.

Figure 24: Arrow indicating that the correct direction is to the left of the player

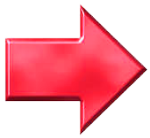

Figure 25: Arrow indicating that the correct direction is to the right of the player

A..1 Parameters

The parameters section shows up after choosing to play one vision training technique. For each vision training technique, unique set of parameters are involved.

For the Tondell Arrows technique, the patient can choose from four (4), five (5), or six (6) pairs of arrows that will be present on the Tondell board. This parameter is also the basis for the number of point light colors that will be matched to the pair of arrows. This section can be seen in Figure 26.

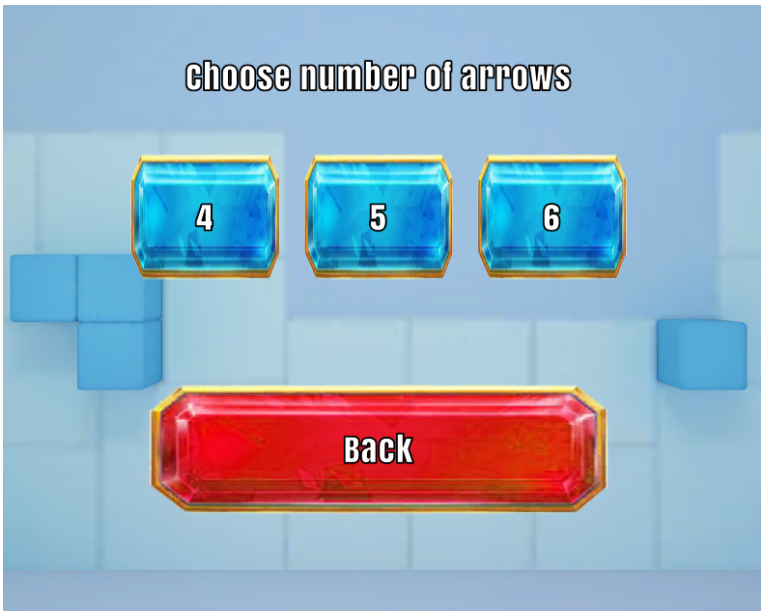

Figure 26: Tondell Arrows technique: Choosing no. of pairs of arrows

Afterwards, the patient must choose the difficulty of the game which may either be easy, medium, or hard. This section can be seen in Figure 27. The difficulty of the game affects the time in which the patient is required to gaze at the lighted pair of arrows. In this case, the higher the difficulty, the shorter the time required for the patient to gaze at the lighted pair of arrows. This implies that having a higher difficulty will make the patient do more objectives throughout the entire duration of the game.

For the Lazy Eight technique, the patient can choose from a single set of

Figure 27: Tondell Arrows technique: Choosing difficulty of the game

parameters: a minimum of five (5) to a maximum of ten (10) checkpoints to be visited in the game. This section can be seen in Figure 28. In game, the number of the checkpoint visted is also the number of infinity symbols to be traced. For example, visiting the 3rd checkpoint requires tracing of 3 infinity symbols.

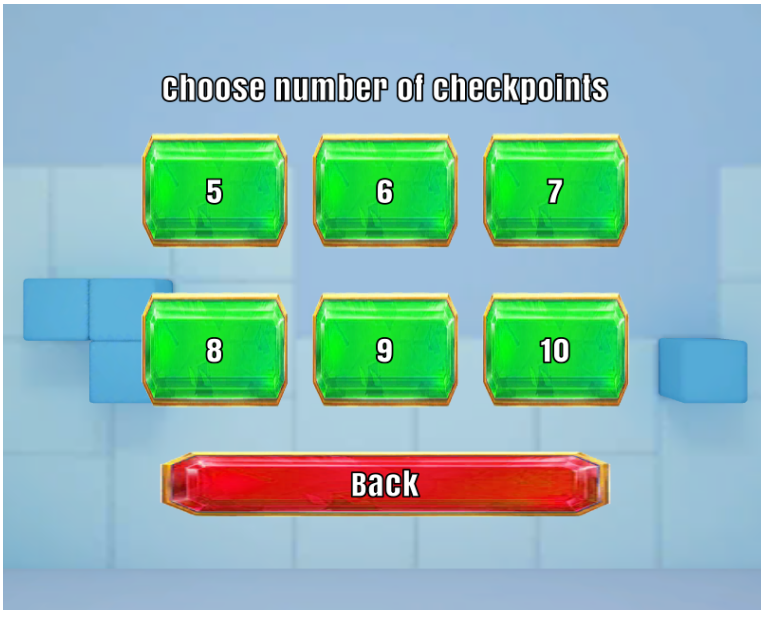

Figure 28: Lazy Eight technique: Choosing no. of infinity symbols to be traced

For the Brock String technique, the patient must choose from three (3), four

(4), or five (5) beads to be gazed at. This section can be seen in Figure 29.

Figure 29: Brock String technique: Choosing no. of beads to be gazed at

Then, the patient must choose the difficulty of the game. This section can be seen in Figure 30. The difficulty affects the spawn rate of the obstacles. Higher the difficulty means faster spawn rate of the obstacles and more obstacles to be avoided in a short period of time.

Figure 30: Brock String technique: Choosing difficulty of the game

There will be an in-game voice guide too. Turning it on gives the patient a

time to prepare and understand more clearly what the objective is all about.

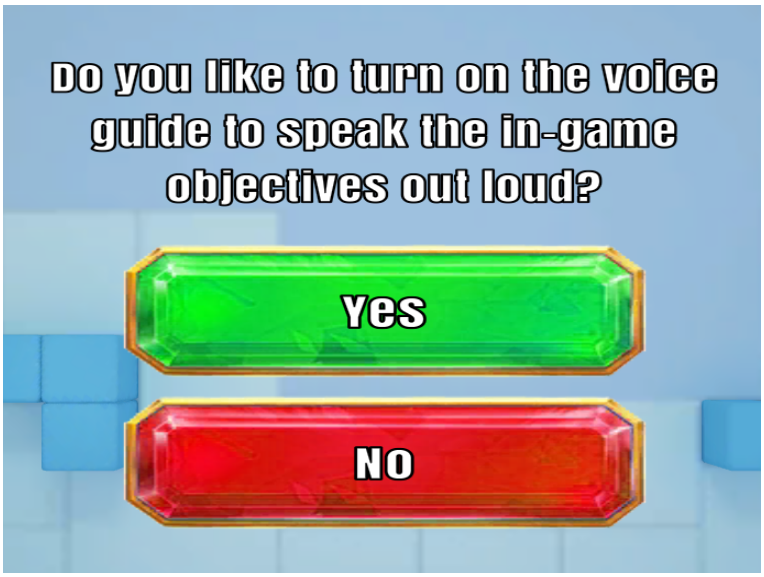

Figure 31: Choosing whether or not to have a voice guide for the objectives

A..2 Tutorials

The tutorials section only shows up when the patient selects the "Tutorials" button. This section provides basic instructions and things to remember for each vision training technique. Certain sections of the tutorials for a specific vision training technique provide in game things to do and not to do.

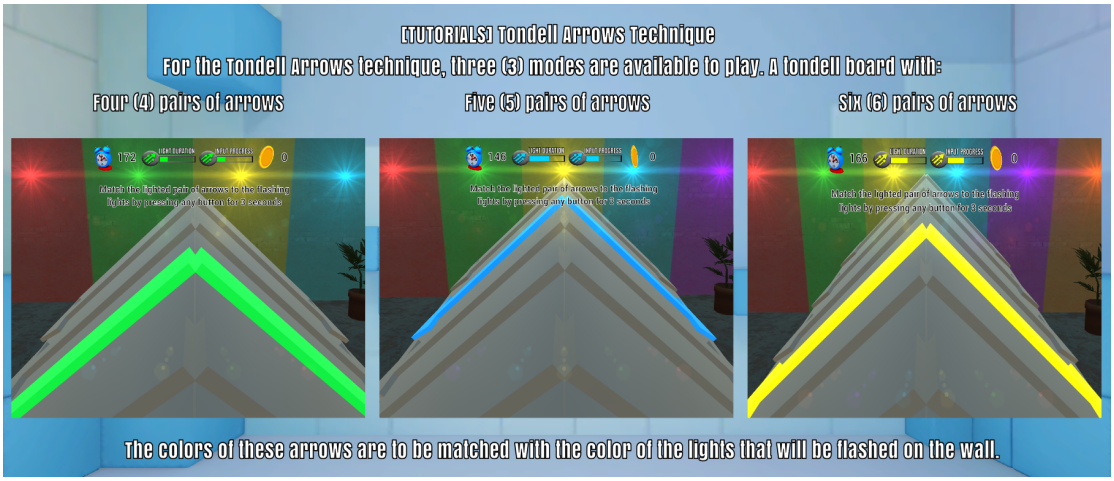

Figure 32: First section of the Tondell Arrows technique

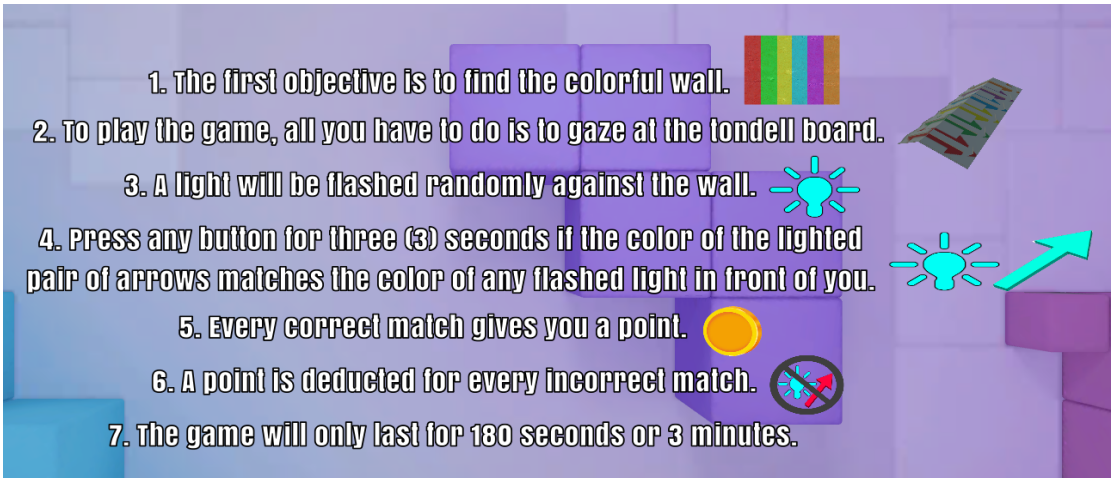

Figure 33: Second section of the Tondell Arrows technique

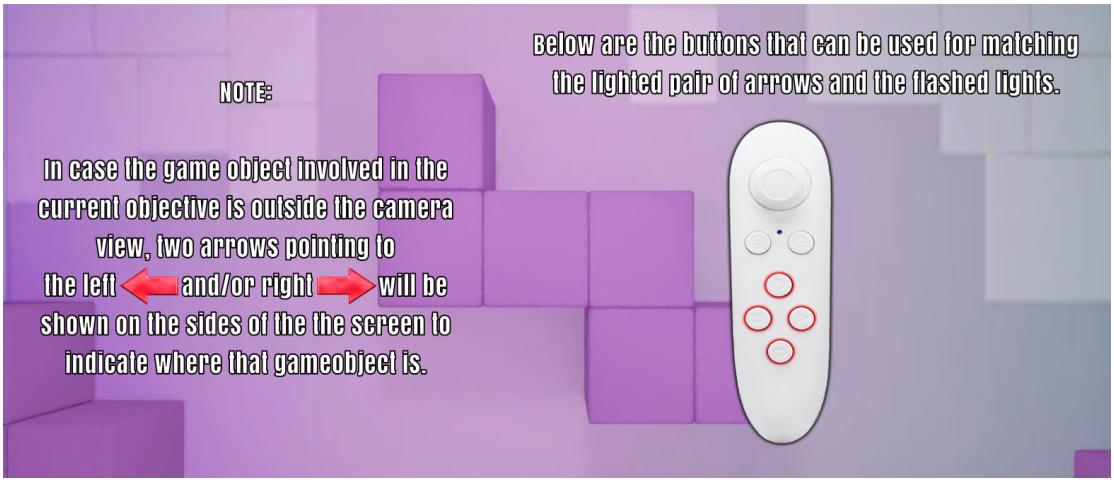

Figure 34: Third section of the Tondell Arrows technique

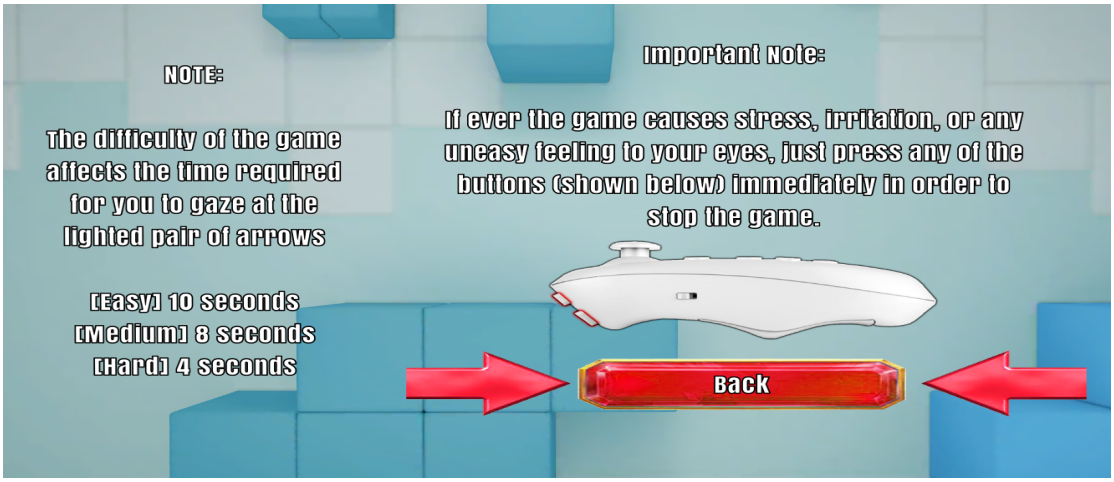

Figure 35: Fourth section of the Tondell Arrows technique

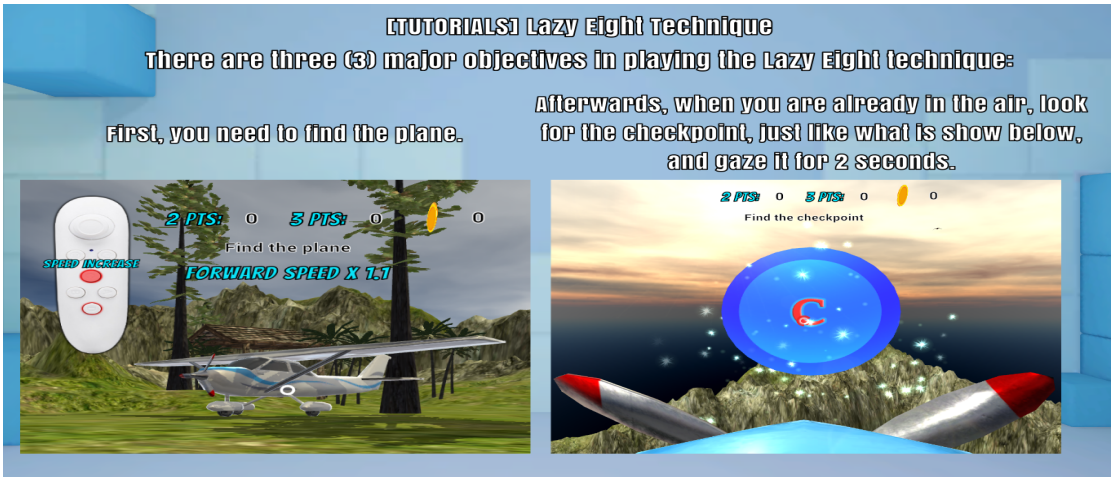

Figure 36: First section of the Lazy Eight technique

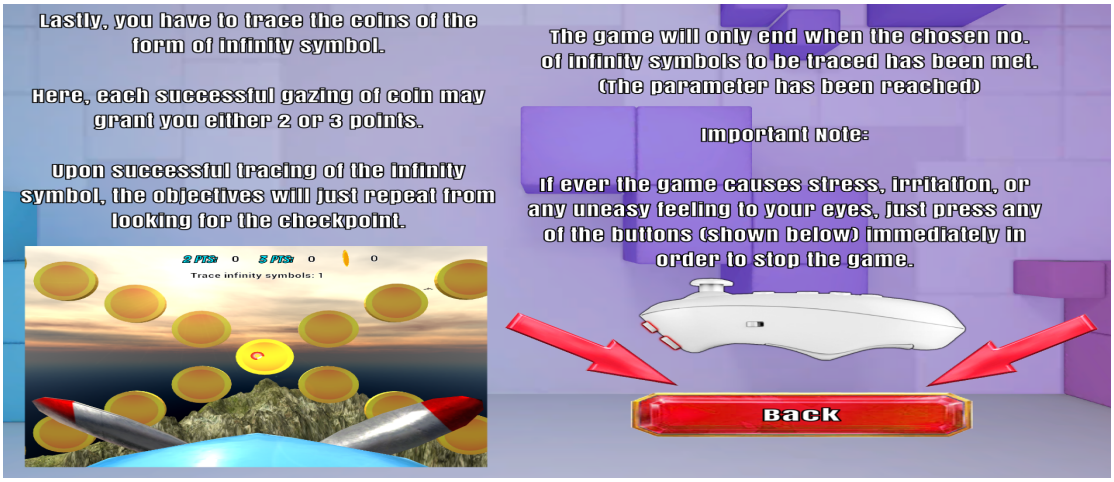

Figure 37: Second section of the Lazy Eight technique

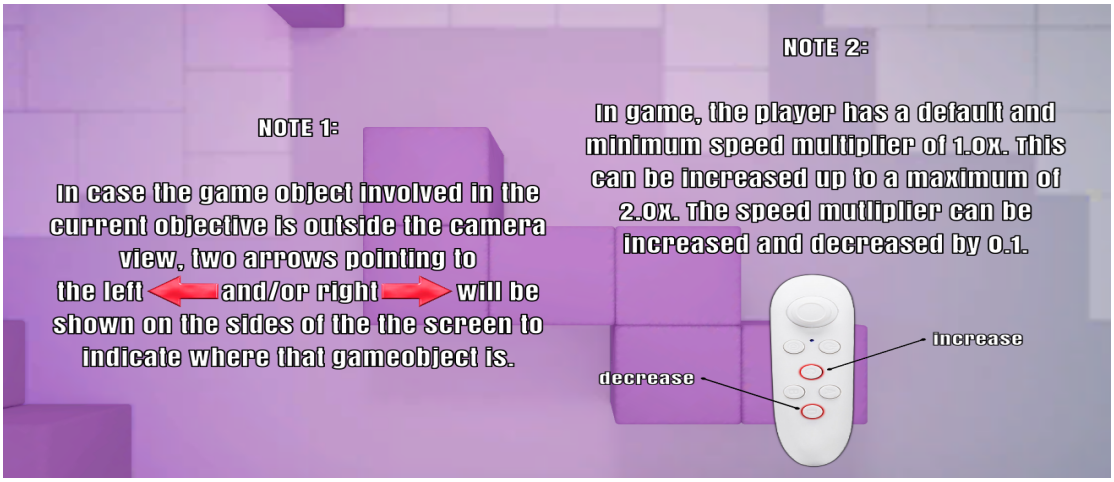

Figure 38: Third section of the Lazy Eight technique

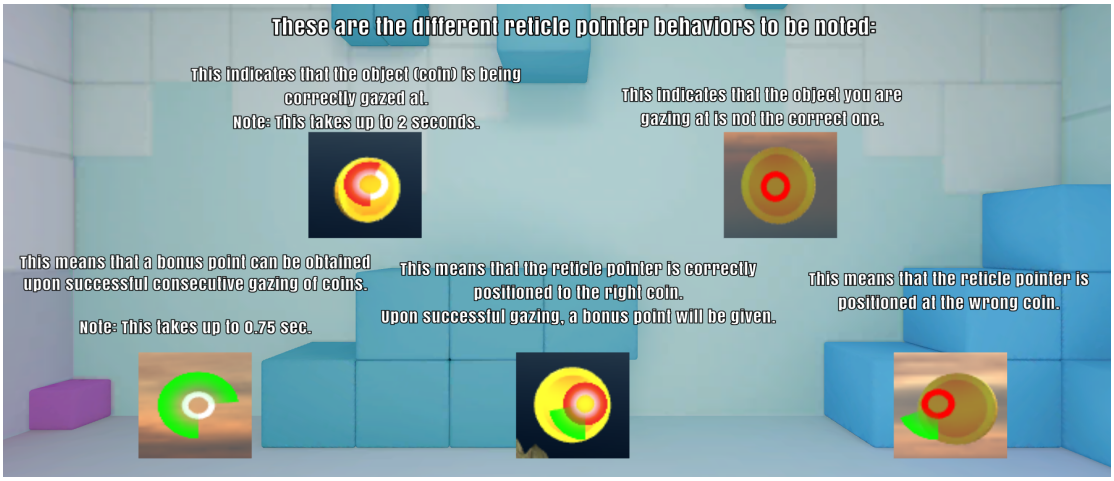

Figure 39: Fourth section of the Lazy Eight technique

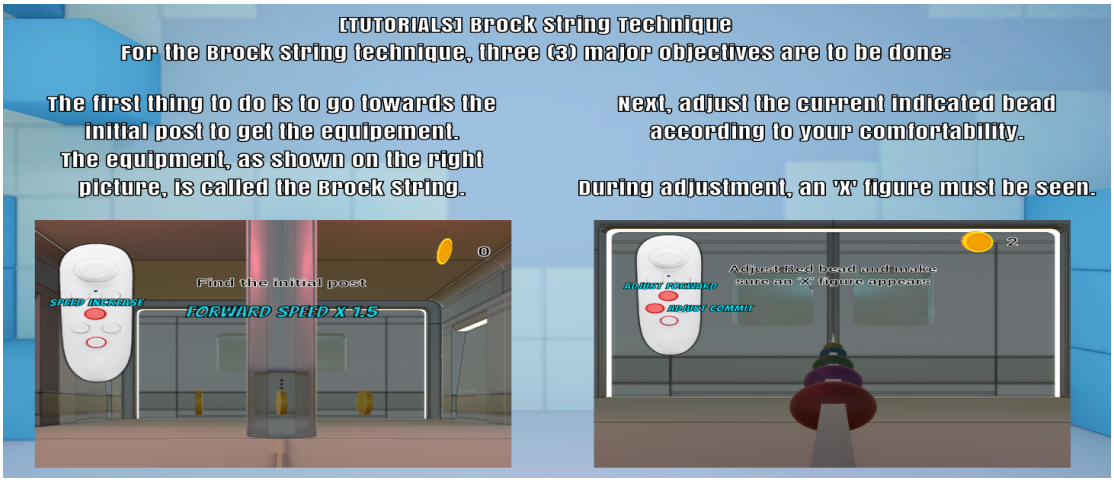

Figure 40: First section of the Brock String technique

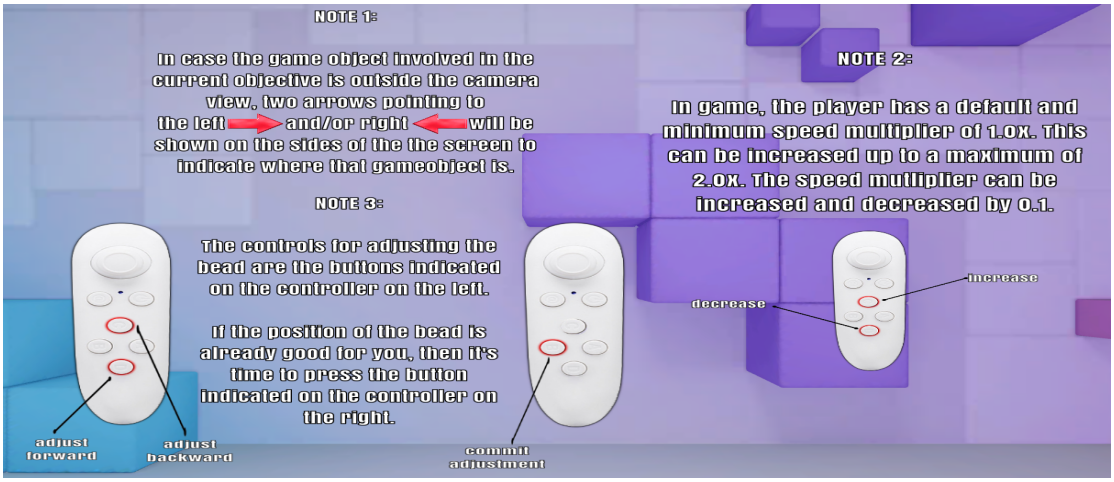

Figure 41: Second section of the Brock String technique

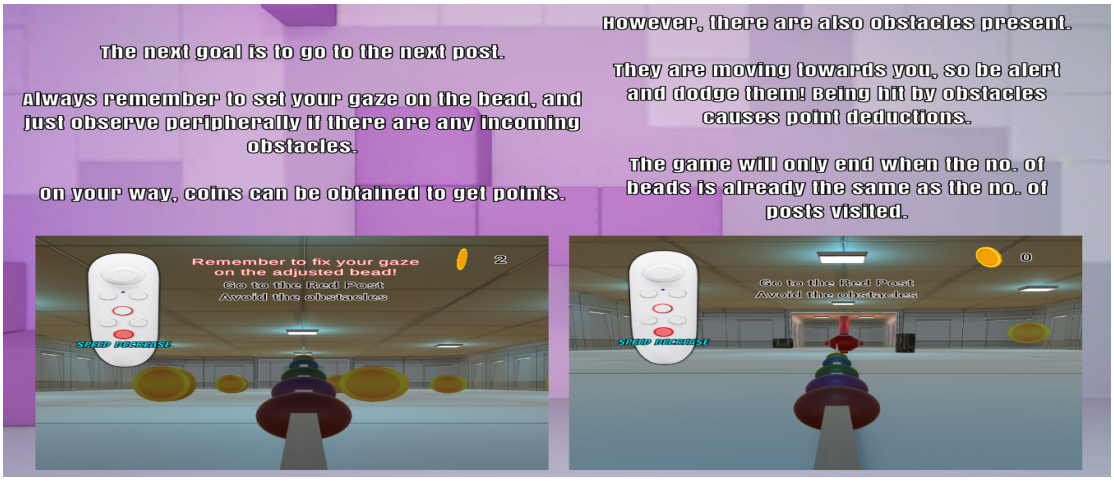

Figure 42: Third section of the Brock String technique

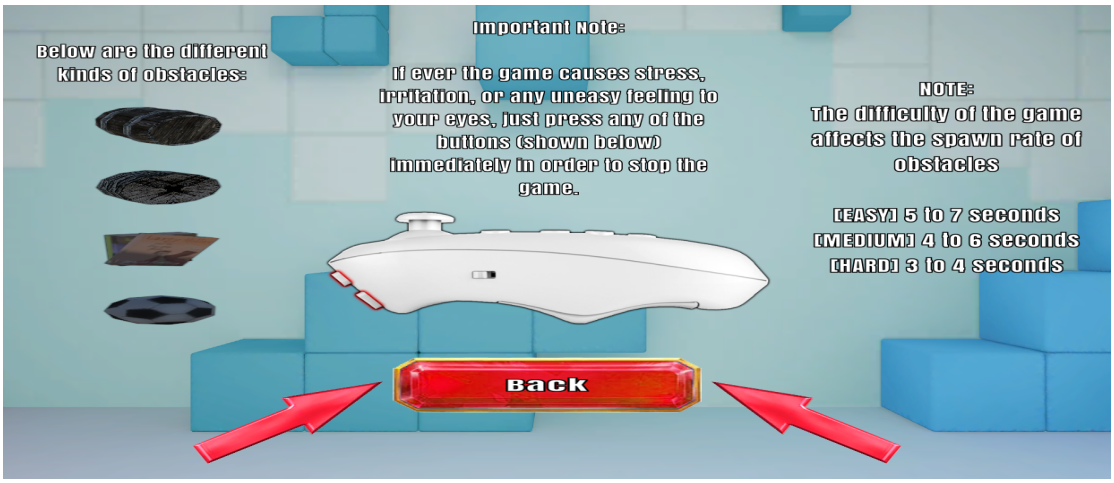

Figure 43: Fourth section of the Brock String technique

B. Virtual Environment

The virtual environment for each vision training technique will only be shown and loaded after the patient has successfully passed the parameters section. To indicate that the application is still working after the parameter section from the main menu, a loading screen is provided. The score and the game objective are displayed on the upper right corner and upper middle part of the screen, respectively. The game objective changes once the curent objective is completed.

B..1 Tondell Arrows Technique

For the Tondell Arrows technique, there are three (3) additional information that are displayed on the screen, namely: the game time which is displayed on the upper left corner of the screen, and the light up time duration of random pair of arrows which is displayed on the upper middle part of the screen just above the game objective, input progress section, which indicates the length of time for the button being pressed, located at the top of the screen.

In the game proper, the player is only standing in the middle of the room and he/she must find the colorful wall. Once found, the player needs to press any button from the controller to start the game. The indication that the game has already started is when a random pair of arrows has been lighted.

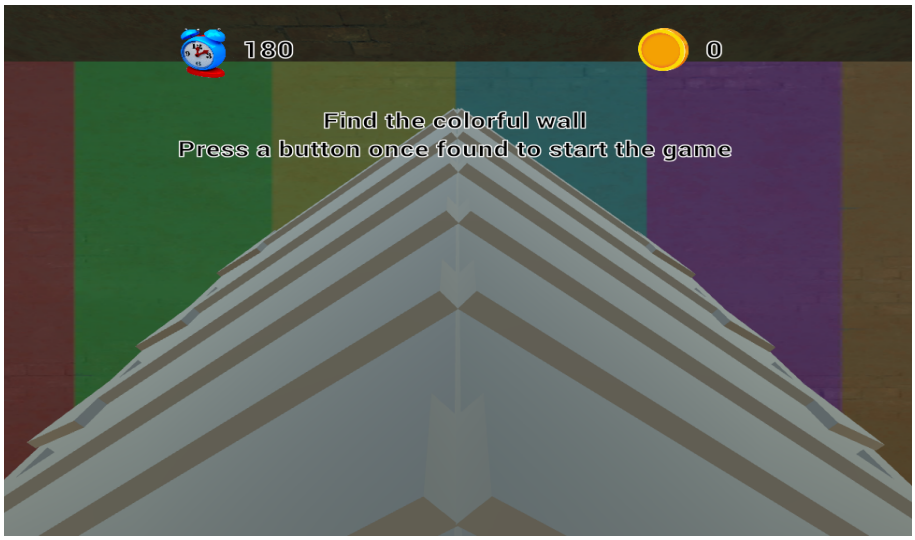

Figure 44: First scene from the Tondell Arrows technique

Then, the patient must gaze at the pair of lighted arrows and observe peripherally if there is a light that is of the same color as that of the lighted pair of arrows. If there is, the patient must press any button on the controller for three (3) seconds in order to gain a point. However, if a button is pressed for three (3) seconds, and there is no light flashed that is of the same color as the lighted pair of arrows, then a point will be deducted from the player. The player must do those objectives for 180 seconds or 3 minutes. The game ends afterwards.

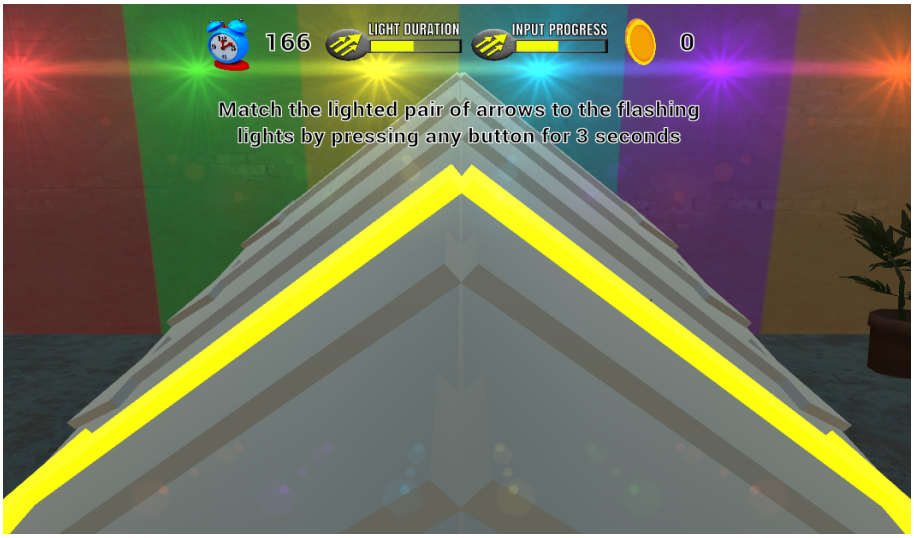

Figure 45: Second scene from the Tondell Arrows technique

B..2 Lazy Eight Technique

The first objective for the Lazy Eight technique is to find the plane within the open field. Once found, the player needs to gaze at the plane for two (2) seconds in order to ride it. Afterwards, the player will be brought up in the air in order to complete the next objective.

Figure 46: First scene from the Lazy Eight technique

In game, the player has a default and minimum speed multiplier of 1.0x. This can be increased up to a maximum of 2.0x. The speed mutliplier can be increased

and decreased by 0.1. The controls on increasing and decreasing the speed multiplier are shown in Figure 47. The speed multiplier can be manipulated while finding the plane and when riding the plane.

Figure 47: Speed multiplier controls for Lazy Eight technique

For the second objective, the player needs to locate the checkpoint and go near it until the plane halts. The same as the plane, the player needs to gaze at the checkpoint for two (2) seconds in order to complete the objective.

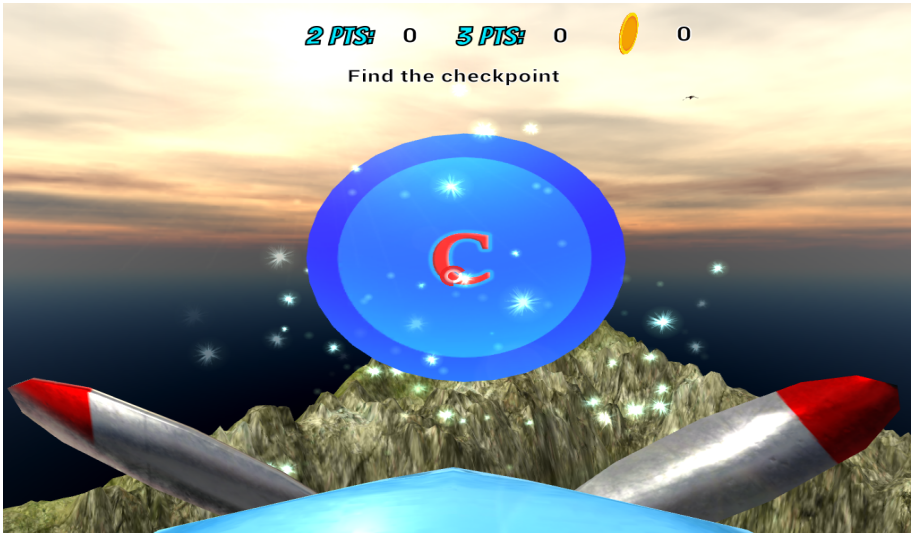

Figure 48: Second scene from the Lazy Eight technique

The third objective of the game requires the player to trace the infinity symbol which is made of coins. It should be taken note that the no. of infinity symbols to be traced is equal to the no. of checkpoints reached. For each infinity symbol, each coin must be gazed at for 0.75 second (bonus time) or 2 seconds (default time), depending on the no. of successfully gazed coins within a short period of time.

At the start of tracing, the player is required to gaze at the first coin for 2 seconds. The player will receive two (2) points for every successful gazed coin. Afterwards, the player will receive a bonus time gaze of 0.75 second. This means that the next coin that the patient will gaze at can be successfully done within 0.75 second. If successful, the player will receive three (3) points for the successfully gazed coin, and receive again a bonus time gaze of 0.75 second for the next coin. However, if the next coin is not successfully gazed at for 0.75 second, the bonus time gaze will be gone, and the player must gaze at that coin for two (2) seconds. The idea will apply throughout the entire duration of tracing, until the last coin is successfully gazed at. Lastly, the second and third objectives of the game will be repeated until the selected no. of infinity symbols (parameter) has been reached. Afterwards, the game ends.

Figure 49: Third scene from the Lazy Eight technique

B..3 Brock String Technique

The Brock String technique starts with the player being elevated to the activity grounds. Then, the first objective of the game is for the player to go towards the initial post. From there, the player can obtain the brock string equipment.

Figure 50: First scene from the Brock String technique

In game, the player has a default and minimum speed multiplier of 1.0x wich can be increased to a maximum of 2.0x. Each increase or decrease costs 0.1. The controls for manipulating the speed multiplier are shown in Figure 51.

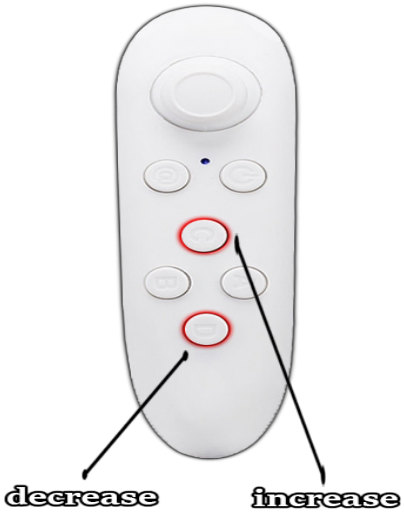

Figure 51: Speed multiplier controls for Brock String technique

Afterwards, the next objective is for the player to gaze at the first bead, and adjust it forward or backward depending on the choice of the player. Before declaring the desired adjustment of the bead, the player must be able to see an 'X' figure formed by the string. To do so, the player must actually be gazing at the current bead being adjusted. Otherwise, the core objective of the Brock String vision training technique will not be met, and the technique will not be effective.

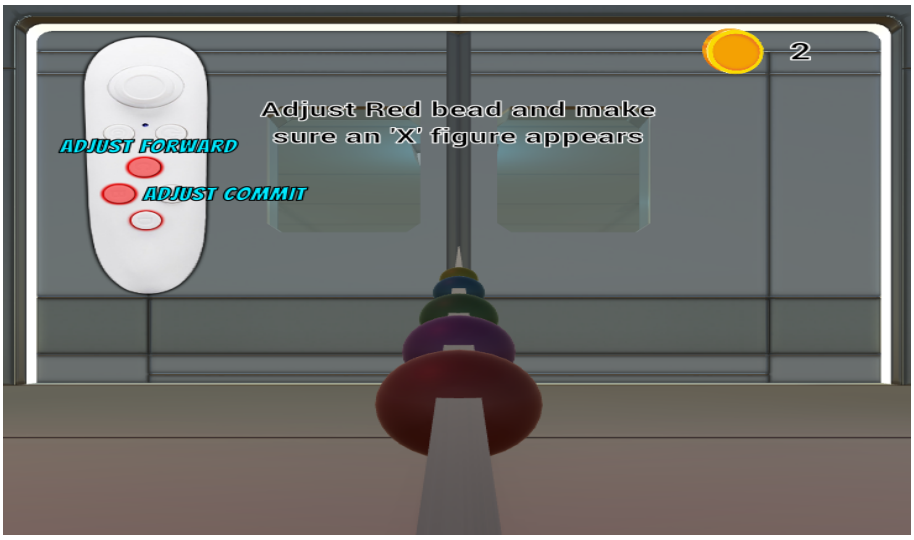

Figure 52: Second scene from the Brock String technique

Figures 53 and 54 represent the controls for the objective mentioned above.

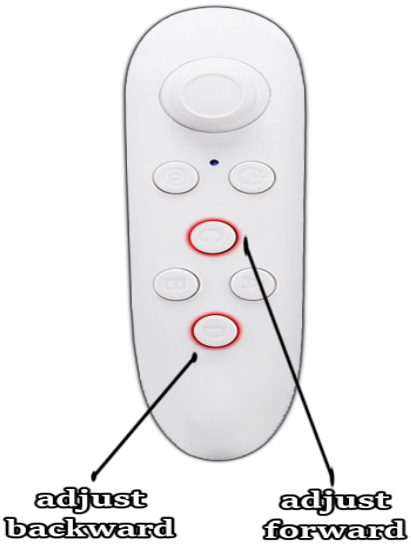

Figure 53: Controls for adjusting bead

Figure 54: Controls for declaring adjustment of bead

The next objective requires the player to go to the next post which is located behind the next door. On the way, coins can be collected by the player in order to gain a point. However, moving obstacles are also present. These obstacles must be avoided by the player, otherwise, a point will be deducted.

Figure 55: Third scene from the Brock String technique

The second and third objectives will only be repeated until the no. of posts reached is already equal to the no. of beads (parameter) present on the string.

C. Force Stop

The patient may opt to force stop the game whenever the vision training technique that he/she is performing causes discomfort, irritation, or any uneasy feeling to his/her eyes. The controls for force stopping the game is shown in Figure 56. After force stopping the game, the post game screen will be displayed.

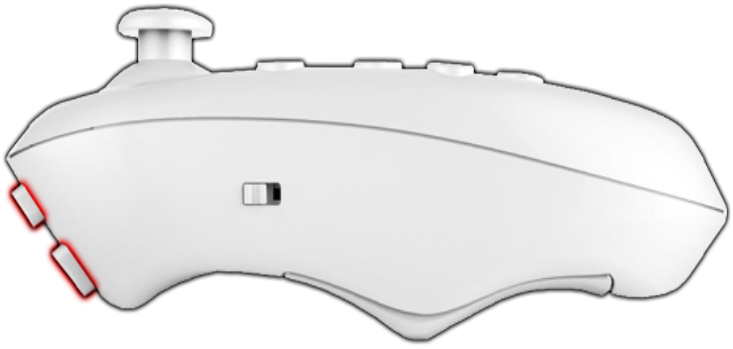

Figure 56: Controls for force stopping a game

D. Post Game Screen

The post game screen consists of the score-based summary of the performance of the patient in doing the vision training technique. It consists of the breakdown of accumulated points, deductions, and total score of the patient.

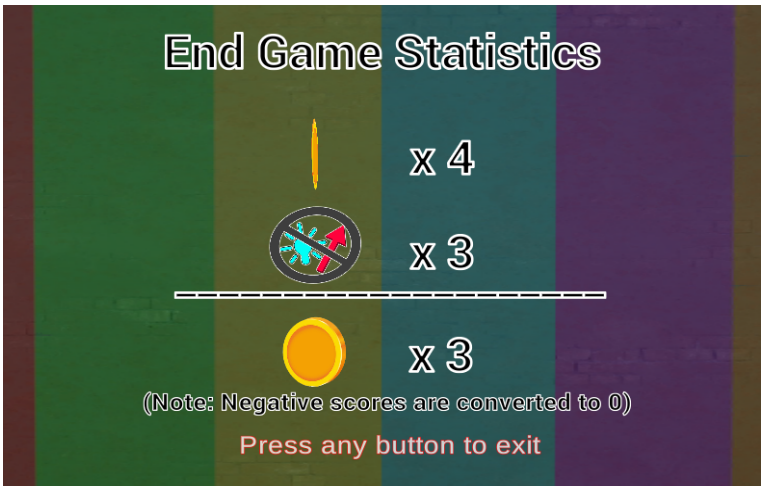

Figure 57: Post game screen for Tondell Arrows technique

Figure 58: Post game screen for Lazy Eight technique

Figure 59: Post game screen for Brock String technique

VI. Discussion

STRIVE-VR is a virtual reality application that contains three (3) vision training techniques for patients with strabismus. These vision training techniques contain different virtual environments for the patient to be immersed with. At the end of each training routine, a score-based summary performance of the patient is generated.

Software tools such as Blender and Unity were used in the development of the application. Blender allows the creation and modelling of 3D objects. In STRIVE-VR, the equipment used for the Tondell Arrows technique and the Brock String technique were made using Blender. These pieces of equipment were imported to Unity, since it supports 3D objects made from Blender. On the other hand, Unity is a game development tool that is used in the completion of the STRIVE-VR application. It includes a built-in 3D modelling section known as the ProBuilder which was used in creating most of the room components for the Tondell Arrows technique and the Brock String technique. Unity also includes downloadable assets where most of the environment components for STRIVE-VR came from. Furthermore, the imported game objects were incorporated with C# scripts to produce interactions, animations, controls, sounds, scoring system, and scene-to-scene transitions. Moreover, the sound effects used for the game application were downloaded for free from Youtube. Aside from that, several background music were also used in the application as part of the gamification. These sound tracks were obtained from NoCopyrightSounds (NCS) Youtube channel that offers free and downloadable msuic. Lastly, the in-game voice guide which speaks the game objectives out loud were made from Natural Readers, an online website which can be used for free and converts text to speech with various voices you can choose from. After development, the application was built from Unity to produce an Android Package Kit (APK) file that can be ran in an Android phone.

Moreover, the objectives of the STRIVE-VR application were all met. The vision therapist can choose one (1) among the three (3) vision training techniques

and set its corresponding parameters from the main menu screen - GoogleVR SDK provided the means of displaying the main menu in virtual reality mode. Also, an in-game voice guide can also be turned on or off. Turning it on gives the patient a time to prepare and understand more clearly what the objective is all about. Afterwards, the environment for the chosen vision training technique will be loaded. In the game proper, an automatic walking script was implemented for the Lazy Eight technique and the Brock String technique. A walking script for the Tondell Technique is not necessary since the player only stands in the middle of the room. Subsequently, the objectives of each vision training technique can be met by interacting with other game objects within the environment which were incorporated with scripts too, as stated previously. A canvas was used for displaying these objectives, and they can be viewed any time at the upper middle part of the screen. It will change as long as it is completed. Lastly, in case the patient feels irritation, discomfort, or any uneasy feeling, he/she can immediately force stop the game by pressing the front side buttons on the controller. Every after finishing a vision training technique, or even force stopping the game, a score-based summary performance of the patient is generated and be displayed on the screen.

VII. Conclusions

STRIVE-VR is a virtual reality application that allows patients with strabismus to perform vision training techniques. It includes various virtual environments for the three (3) vision training techniques: a virtual room environment for the Tondell Arrows technique, a virtual open field environment for the Lazy Eight technique, and a virtual activity area for the Brock String technique. Each vision training technique has different sets of parameters and objectives in the game that must be done by the patient. Every time a vision training technique has been finished or force stopped, a score-based summary performance of the patient will be available and will be displayed on the screen. This summary will contain the accumulated points, deductions, and total score of the patient throughout the entire duration of the game.

Unity was used mostly for the development of the STRIVE-VR application. GoogleVR SDK is used in Unity to convert all scenes into virtual reality mode. Moreover, although some of the 3D game objects imported in Unity were made from Blender, Unity has its own 3D modelling section called the ProBuilder which was used in the creation of several room components. Aside from that, Unity also has its Asset store wherein readily made game objects and environments can be imported to the application.

STRIVE-VR is designed to serve as an adjunct to the traditional vision training techniques. It incorporates these vision training techniques with gamification to promote user engagement, interest, and active participation among patients. The application also contains in-game sound effects, several background music for the patient to feel the full gaming experience while performing the vision training techniques, and an optional voice guide which speaks the current game objective out loud - turning this on gives the patient a time to prepare and understand more clearly what the objective is all about.

VIII. Recommendations

Since STRIVE-VR aims to provide vision training techniques for patients with strabismis, one major improvement that can be applied for STRIVE-VR application is by using a virtual reality headset that has eyetracker in it. Although the objectives of each vision training techniques may ensure that the patient actually trains his/her eyes, still, having a virtual reality headset that tracks the eye of the patient would be the best for the application. In that way, the vision therapist can actually verify the performance of the patient.

Another improvement for the STRIVE-VR application is the implementation of pure gaze-only in-game actions. Since a VR box controller is not always available together with the VR box headset, having pure gaze-only in-game actions would be ideal for the training of the patient.

IX. Bibliography

[1] "Strabismus." https://medlineplus.gov/ency/article/001004.htm, 2017.

[2] X. Ye, V. Pegado, M. Patel, and W. Wasserman, "Strabismus genetics across a spectrum of eye misalignment disorders," *Clinical Genetics: An International Journal of Genetics, Molecular and Personalized Medicine*, 2014.

[3] D. M. Barbu and I. M. Plesa, "Techniques and optometric tools for visual training in strabismus for preschool children," *The 5th IEEE International Conference on E-Health and Bioengineering - EHB 2015*, 2015.

[4] U. Saisara, P. Boonbrahm, and A. Chaiwiriya, "Strabismus screening by eye tracker and games," *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2017.

[5] Z. Chen, H. Fu, W.-L. Lo, and Z. Chi2, "Eye-tracking aided digital system for strabismus diagnosis," *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015.

[6] M. F. Khosravi, A. Janati, A. Imani, A. Javadzadeh, and M. M. Gharamaleki, "Cost analysis of strabismus surgery by activity based costing," *The IIOAB Journal Regular Issue*, 2016.

[7] S. B. Ozkan, "Restrictive problems related to strabismus surgery," *Taiwan Journal of Ophthalmology 6 (2016)*, 2016.

[8] D. L. Cook, "Strabismus: Non-surgical cure rates." http://cookvisiontherapy.com/strabismus--crossed-eyes-/strabismus-causes-and-curerates.html, 2017.

[9] T. Wang and L.-H. Wang, "Surgical treatment for residual or recurrent strabismus," *Int J Ophthalmol*, 2014.

[10] M. Mahan and J. M. Engel, "The resurgence of botulinum toxin injection for strabismus in children," *2017 Wolters Kluwer Health, Inc*, 2017.

[11] D. M. Sugano, C. L. Fernandez, and J. R. C. de Lima Rehder, "Botulinum toxin for strabismus correction," *Revista Brasileira de Oftalmologia*, 2013.

[12] G. R. Beauchamp and P. R. Mitchell, "A patient and parent guide to strabismus surgery." http://pediatricophthalmologypa.com/wp-content/uploads/2013/12/A-Patient-Parent-Guide-to-Strabismus-Surgery.pdf, 2013.

[13] G. Heiting, "Vision therapy for children." http://www.allaboutvision.com/parents/vision_therapy.htm, 2017.

[14] J. Cooper and R. Cooper, "What is strabismus?." http://www.strabismus.org/surgery_crossed_eyes.html, 2016.

[15] Anonymous, "Testing and treatment costs for vision therapy." http://www.thevisiontherapycenter.com/costs-the-vision-therapy-center, 2017.

[16] J. F. Clark, P. Graman, and J. K. Ellis, "Depth perception improvement in collegiate baseball players with vision training," *Optometry and Visual Performance*, 2015.

[17] N. Singh and S. Singh, "Virtual reality: A brief survey," *2017 International Conference on Information Communication and Embedded Systems (ICICES)*, 2017.

[18] C. G. Tsatsis, K. E. Rice, V. Protopopova, D. Ramos, J. Jadav, D. J. F. Coppola, M. Broderick, and D. Putrino, "Lateropulsion rehabilitation using virtual reality for stroke patients," *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 2017.

[19] Anonymous, "Training with immersive technology offers advantages over traditional methods." https://360immersive.com/360i-why-train-with-vr/, 2017.

[20] T. M. Fleming, L. Bavin, K. Stasiak, E. Hermansson-Webb1, S. N. Merry, C. Cheek, M. Lucassen, H. M. Lau, B. Pollmuller, and S. Hetrick, "Serious games and gamification for mental health: Current status and promising directions," *Front. Psychiatry 7:215*, 2017.

[21] J. Blaha and M. Gupta, "Diplopia: A virtual reality game designed to help amblyopics," *Virtual Reality (VR), 2014 iEEE*, 2014.

[22] N. Filesler, "Can virtual reality headsets save vision in people with lazy eye?." https://vector.childrenshospital.org/2017/04/virtual-reality-headsets-could-treat-amblyopia/.

[23] Anonymous, "Amblyopia." https://nei.nih.gov/healthyeyes/problems.

[24] J. Blaha, "Im james from vivid vision." https://www.reddit.com/r/IAmA/comments/77lx0t/im_james_from_vivid_vision_im_back_to_answer_your/, 2017.

[25] J. Blaha, "Virtual reality vision therapy is now available to be prescribed for home use." https://www.seevividly.com/blog/123/Virtual_Reality_vision_therapy_is_now_available_to_be_prescribed_for_home_use, 2017.

[26] G. Hristov, J. Raychev, D. Kyuchukova, and P. Zahariev, "Development of a educational three dimensional computer game by using virtual reality utilities," *2017 16th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2017.

[27] R. Evans, "3 eye exercises to help strabismus." https://www.healthline.com/health/eye-health/strabismus-exercises, 2016.

[28] I. Vedamurthy, D. C. Knill, S. J. Huang, A. Yung, J. Ding, O.-S. Kwon, D. Bavelier, and D. M. Levi, "Recovering stereo vision by squashing virtual bugs in a virtual reality environment," *Phil. Trans. R. Soc. B 371: 20150264*, 2016.

[29] A. Awadein, "A computerized version of the lancaster red-green test," *2013 American Association for Pediatric Ophthalmology and Strabismus*, 2013.

[30] N. Nesaratnam, P. Thomas, and A. Vivian, "Restrictive problems related to strabismus surgery," *Macmillan Publishers Limited, part of Springer Nature*, 2017.

[31] A. Gargantini, F. Terzi, M. Zambelli, and S. Bonfanti, "A low-cost virtual reality game for amblyopia rehabilitation," *REHAB '15 Proceedings of the 3rd 2015 Workshop on ICTs for improving Patients Rehabilitation Research Techniques*, 2015.

[32] N. Herbison, S. Cobb, R. Gregson, I. Ash, R. Eastgate, J. Purdy, T. Hepburn, D. MacKeith, and A. Foss, "Interactive binocular treatment (i-bit) for amblyopia: results of a pilot study of 3d shutter glasses system," *2013 Macmillan Publishers Limited*, 2013.

[33] P. iak, A. Holm, J. Halika, P. Moji, and D. P. Piero, "Amblyopia treatment of adults with dichoptic training using the virtual reality oculus rift head mounted display: preliminary results," *BMC Ophthalmology*, 2017.

[34] P.-Y. Laffont, T. Martin, M. Gross, W. D. Tan, C. Lim, A. Au, and R. Wong, "Rectifeye: A vision-correcting system for virtual reality," *SA '16 SIGGRAPH ASIA 2016 Posters*, 2016.

[35] F. B. Daga, E. Macagno, C. Stevenson, A. Elhosseiny, A. Diniz-Filho, E. R. Boer, J. Schulze, and F. A. Medeiros, "Wayfinding and glaucoma: A virtual reality experiment," *Invest Ophthalmol Vis Sci*, 2017.

[36] Anonymous, "What is strabismus?." Retrievedfromhttp://www.strabismus.org/.

[37] V. Tailor, S. Balduzzi, S. Hull, J. Rahi, C. Schmucker, G. Virgili, and A. Dahlmann-Noor, "Tests for detecting strabismus in children age 1 to 6 years in the community (protocol)," *2014 The Cochrane Collaboration*, 2014.

[38] Anonymous, "Strabismus." https://aapos.org/terms/conditions/100, 2014.

[39] Anonymous, "Strabismus (crossed eyes)." https://www.aoa.org/patients-and-public/eye-and-vision-problems/glossary-of-eye-and-vison-conditions/strabismus, 2017.

[40] N. Khumdat, P. Phukpattaranont, and S. Tengtrisorn, "Development of a computer system for strabismus screening," *The 2013 Biomedical Engineering International Conference (BMEiCON-2013)*, 2015.

[41] Anonymous, "What is virtual reality?." https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html, 2017.

X. Appendix

AdjustBead.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using GoogleVR.VideoDemo;
using TMPro;

public class AdjustBead : MonoBehaviour {

public GameObject instructions;
public GameObject reminder;
public GameObject player;
public GameObject front_door_checkpoint;
public GameObject adjust_controls;
public GameObject speed_controls;
public GameObject objective_good;
public Transform vrCamera;
public AudioClip next_objective;
private Material mat;
private int param_index;
private int sibIdx;
private float pressed_key;
private bool chk_done;
private bool chk_adjusted;

void Start () {
sibIdx = transform.GetSiblingIndex();
mat = GetComponent<Renderer>().material;
param_index = transform.parent.gameObject.
GetComponent<BrockParam>().get_parameter();
if(param_index == 3){
transform.parent.GetChild(1).gameObject.
transform.localPosition = new Vector3(transform.
localPosition.x, -0.333f, -1.744f);
transform.parent.GetChild(2).gameObject.
transform.localPosition = new Vector3(transform.
localPosition.x, -0.3147739f, -1.616209f);
transform.parent.GetChild(3).gameObject.
transform.localPosition = new Vector3(transform.
localPosition.x, -0.2782764f, -1.36091f);
} else {
transform.parent.GetChild(1).gameObject.
transform.localPosition = new Vector3(transform.
localPosition.x, -0.333f, -1.744f);
transform.parent.GetChild(2).gameObject.
transform.localPosition = new Vector3(transform.
localPosition.x, -0.3251727f, -1.689146f);
transform.parent.GetChild(3).gameObject.
transform.localPosition = new Vector3(transform.
localPosition.x, -0.3147739f, -1.616209f);
transform.parent.GetChild(4).gameObject.
transform.localPosition = new Vector3(transform.
localPosition.x, -0.2967456f, -1.49011f);
if(param_index == 5){
transform.parent.GetChild(5).gameObject.
transform.localPosition = new Vector3(transform.
localPosition.x, -0.2782764f, -1.36091f);
}
}
transform.parent.parent.gameObject.GetComponent
<BrockPlayerDirection>().
directions_status(false);
if(objective_good != null){
objective_good.GetComponent<TextMeshProUGUI>().
text = "Controls will only apply once the voice
is done speaking.";
GetComponent<AudioSource>().Play();
}
chk_done = false;
pressed_key = 0.0f;
chk_adjusted = false;
}

void Update () {
vrCamera.transform.eulerAngles = new
Vector3(0, 0, 0);
if(!chk_done && !GetComponent<AudioSource>().
isPlaying){
if(objective_good != null){
objective_good.GetComponent<TextMeshProUGUI>().
text = "Press any button if you understand
the objective\nto continue the game";
}
if(Input.anyKey){
if(objective_good != null){
objective_good.GetComponent<CanvasGroup>().
alpha = 0;
}
chk_done = true;
pressed_key = Time.time;
}
}

if(chk_done && !chk_adjusted && (objective_good
== null || (objective_good != null &&
pressed_key+1.0f <= Time.time))){
if(Input.GetKey(KeyCode.Joystick1Button0) ||
Input.GetKey(KeyCode.A)){
transform.position = new Vector3(transform.
position.x, transform.position.y + Time.
deltaTime * 0.0071375f * 2, transform.
position.z + Time.deltaTime * 0.05f * 2);
} else if(Input.GetKey(KeyCode.
Joystick1Button3) || Input.GetKey
(KeyCode.D)){
transform.position = new
Vector3(transform.position.x, transform.
position.y - Time.deltaTime * 0.0071375f
* 2, transform.position.z - Time.deltaTime
* 0.05f * 2);
} else if(Input.GetKey(KeyCode.
Joystick1Button2) || Input.GetKey
(KeyCode.W)){
adjust_controls.SetActive(false);
speed_controls.SetActive(true);
speed_controls.transform.GetChild(0).
gameObject.GetComponent<Controller
Script>().show_controls();
transform.parent.parent.gameObject.
GetComponent<BrockPlayerDirection>().
directions_status(true);
player.transform.localPosition = new
Vector3(-40f, -65f, -540f);
player.GetComponent<PlayerMovement>().
change_bead_ctr(sibIdx);

instructions.GetComponent<TextMesh
ProUGUI>().text = "Go to the " + (mat+"").
Substring(0, (mat+"").IndexOf("(")-1) +
" Post\nAvoid the obstacles";
chk_adjusted = true;
pressed_key = Time.time;

if(objective_good != null){
reminder.GetComponent<CanvasGroup>().
alpha = 1;
GetComponent<AudioSource>().clip =
next_objective;
GetComponent<AudioSource>().Play();

objective_good.GetComponent<TextMesh
ProUGUI>().text = "Controls will only apply
once the voice is done speaking.";
objective_good.GetComponent<Canvas
Group>().alpha = 1;
}

String mat_string = "Diffuse_01_";
if((mat+"").IndexOf("Red") >= 0){
mat_string += "red";
} else if((mat+"").IndexOf("Violet") >= 0){
mat_string += "violet";
} else if((mat+"").IndexOf("Green") >= 0){
mat_string += "green";
} else if((mat+"").IndexOf("Blue") >= 0){
mat_string += "blue";
} else {
mat_string += "yellow";
}

Material check_mat = Resources.Load
(mat_string, typeof(Material)) as Material;
for(int i = 0; i < front_door_checkpoint.
transform.childCount; i++){
Material[] mat = front_door_checkpoint.
transform.GetChild(i).gameObject.transform.
GetComponent<Renderer>().materials;
mat[0] = check_mat;
front_door_checkpoint.transform.GetChild(i).
gameObject.transform.GetComponent<Rend
erer>().materials = mat;
}
}

if(param_index == 3){
if(sibIdx == 1){
transform.localPosition = new
Vector3(transform.localPosition.x, Mathf.
Clamp(transform.localPosition.y, -0.3389896f,
-0.323067f), Mathf.Clamp(transform.
localPosition.z, -1.78613f, -1.674863f));
} else if(sibIdx == 2){
transform.localPosition = new
Vector3(transform.localPosition.x, Mathf.
Clamp(transform.localPosition.y, -0.3258441f,
-0.2990573f), Mathf.Clamp(transform.
```

```
localPosition.z, -1.694089f, -1.507505f));
} else {
transform.localPosition = new
Vector3(transform.localPosition.x, Mathf.
Clamp(transform.localPosition.y, -0.3004916f,
-0.2620853f), Mathf.Clamp(transform.
localPosition.z, -1.515928f, -1.247495f));
}
} else {
if(sibIdx == 1){
transform.localPosition = new
Vector3(transform.localPosition.x, Mathf.
Clamp(transform.localPosition.y, -0.3389896f,
-0.3283999f), Mathf.Clamp(transform.
localPosition.z, -1.78613f, -1.712285f));
} else if(sibIdx == 2){
transform.localPosition = new
Vector3(transform.localPosition.x, Mathf.
Clamp(transform.localPosition.y, -0.3254703f,
-0.3178561f), Mathf.Clamp(transform.
localPosition.z, -1.691526f, -1.637754f));
} else if(sibIdx == 3){
transform.localPosition = new
Vector3(transform.localPosition.x, Mathf.
Clamp(transform.localPosition.y, -0.314133f,
-0.3065036f), Mathf.Clamp(transform.
localPosition.z, -1.61157f, -1.55792f));
} else if(sibIdx == 4){
transform.localPosition = new
Vector3(transform.localPosition.x, Mathf.
Clamp(transform.localPosition.y,  -0.3003619f,
-0.2920688f), Mathf.Clamp(transform.
localPosition.z, -1.515501f, -1.4577f));
} else {
transform.localPosition = new
Vector3(transform.localPosition.x, Mathf.
Clamp(transform.localPosition.y, -0.274361f,
-0.2636645f), Mathf.Clamp(transform.
localPosition.z, -1.332944f, -1.257932f));
}
}
}

if(chk_adjusted && !GetComponent<Audio
Source>().isPlaying){
if(objective_good != null){
objective_good.GetComponent<TextMesh
ProUGUI>().text = "Press any button if you
understand the objective\nto continue
the game";
}
if(Input.anyKey){
gameObject.transform.parent.parent.parent.
GetComponent<PlayerMovement>().
enabled = true;
reminder.GetComponent<ReminderScript>().
start_reminder();
if(objective_good != null){
objective_good.GetComponent<Canvas
Group>().alpha = 0;
}
GetComponent<AdjustBead>().enabled
 = false;
}
}
}
}
```

ArrowLights.cs

```
 namespace GoogleVR.VideoDemo {
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
public class ArrowLights : MonoBehaviour {

public GameObject spawn;
public GameObject front_wall;
public GameObject light_bar;
public GameObject light_frame;
public GameObject input_bar;
public GameObject input_frame;
public Sprite red_bar;
public Sprite green_bar;
public Sprite yellow_bar;
public Sprite blue_bar;
public Sprite violet_bar;
public Sprite orange_bar;
public Sprite red;
public Sprite green;
public Sprite yellow;
public Sprite blue;
public Sprite violet;
public Sprite orange;
public GameObject objective_good;
public GameObject repress_button;
public GameObject light_label;
public GameObject input_label;
private int param_index;
private int difficulty_index;
private int speed;
private float currentAmount;
private float prev_time;
private float duration_factor;
private float start_time;
private bool chk;
private bool chk_once;
private bool chk_start;
private bool chk_start_duration;
private bool sound_indicator;
private string curr_color;

void Start () {
GameObject menu = GameObject.Find("Sound
Menu");
param_index = menu.GetComponent<Sound
Comm>().param_index;
difficulty_index = menu.GetComponent<Sound
Comm>().difficulty_index;
sound_indicator = menu.GetComponent<Sound
Comm>().sound_indicator;
Destroy(menu);

if(!sound_indicator){
objective_good.transform.parent.GetComponent
<AudioSource>().enabled = false;
Destroy(objective_good);
repress_button.GetComponent<AudioSource>().
enabled = false;
}

int num_arrows = param_index+4;
String mat_string = "wall_6";

if(param_index == 0){
Destroy(gameObject.transform.GetChild(0).
gameObject);
Destroy(gameObject.transform.GetChild(1).
gameObject);
Destroy(gameObject.transform.GetChild(7).
gameObject);
Destroy(gameObject.transform.GetChild(8).
gameObject);
mat_string = "wall_4";
} else if(param_index == 1){
Destroy(gameObject.transform.GetChild(0).
gameObject);
Destroy(gameObject.transform.GetChild(6).
gameObject);
Destroy(gameObject.transform.GetChild(8).
gameObject);
mat_string = "wall_5";
} else {
Destroy(gameObject.transform.GetChild(6).
gameObject);
Destroy(gameObject.transform.GetChild(7).
gameObject);
}

Material wall_mat = Resources.Load
(mat_string, typeof(Material)) as Material;
front_wall.transform.GetComponent
<Renderer>().material = wall_mat;
spawn.GetComponent<SpawnFlash>().
update_num_of_arrows(num_arrows);

duration_factor = 0.0f;
switch(difficulty_index){
case 0:
speed = 10;
break;
case 1:
speed = 8;
duration_factor = 4f;
break;
case 2:
speed = 4;
break;
default:
speed = 8;
break;
}

currentAmount = 100;
prev_time = 0;
chk = true;
chk_once = true;
chk_start = false;
chk_start_duration = false;
start_time = 0.0f;
curr_color = "";
}

void Update () {
if(transform.parent.gameObject.
GetComponent<TondellPlayer
Direction>().chk_start_game()){
```

```
if(!chk_start){
StartCoroutine("Flash");
chk_start = true;
light_bar.SetActive(true);
light_frame.SetActive(true);
input_bar.SetActive(true);
input_frame.SetActive(true);
light_label.SetActive(true);
input_label.SetActive(true);
start_time = Time.time;
chk_start_duration = true;
}
if(chk_start_duration){
if(Time.time-start_time <
speed){
currentAmount = 100-(Time.time
-start_time)*(100/speed)-
(duration_factor);
light_bar.GetComponent<Image>().
fillAmount = currentAmount/100;
} else {
currentAmount = 100;
start_time = Time.time;
}
}
}
}

IEnumerator Flash(){
string color = "";
int numSibs = transform.childCount-1;
int sibIdx = 1;
Material default_render = Resources.
Load("Default", typeof(Material)) as
Material;
Sprite bar_frame = null;
Sprite bar_color = null;
while(chk){
for(int i = 0; i < numSibs; i++){
transform.GetChild(i).GetChild(0).
GetComponent<Renderer>().material
 = default_render;
transform.GetChild(i).GetChild(1).
GetComponent<Renderer>().material
 = default_render;
}
sibIdx = (sibIdx + UnityEngine.
Random.Range(1, numSibs)) % (
numSibs);
switch(sibIdx){
case 0:
color = "Red";
bar_frame = red_bar;
bar_color = red;
break;
case 1:
color = "Green";
bar_frame = green_bar;
bar_color = green;
break;
case 2:
color = "Yellow";
bar_frame = yellow_bar;
bar_color = yellow;
break;
case 3:
color = "Blue";
bar_frame = blue_bar;
bar_color = blue;
break;
case 4:
color = "Violet";
bar_frame = violet_bar;
bar_color = violet;
break;
case 5:
color = "Orange";
bar_frame = orange_bar;
bar_color = orange;
break;
default:
break;
}

curr_color = color;
Material new_render = Resources.
Load("Emissive_"+color,
typeof(Material)) as Material;
transform.GetChild(numSibs-1
-sibIdx).
GetChild(0).GetComponent<Rend
erer>().material = new_render;
transform.GetChild(numSibs-1
-sibIdx).
GetChild(1).GetComponent<Rend
erer>().material = new_render;

light_frame.GetComponent<Image>
().sprite = bar_frame;
light_bar.GetComponent<Image>
().sprite = bar_color;
input_frame.GetComponent<Image>
().sprite = bar_frame;
input_bar.GetComponent<Image>
().sprite = bar_color;
yield return new WaitForSeconds
(speed);
}
}

public string get_curr_color(){
return curr_color;
}

public void disable_coroutine(){
chk = false;
}
}
}
```

BrockParam.cs

```
 namespace GoogleVR.VideoDemo {
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;

public class BrockParam : MonoBehaviour {

private int sibIdx;
private int param_index;
private int difficulty_index;
private int bead_ctr;
private float dist;
private bool sound_indicator;
public GameObject objective_good;

void Start () {
GameObject menu = GameObject.Find("Sound
Menu");
param_index = menu.GetComponent<Sound
Comm>().param_index;
difficulty_index = menu.GetComponent<Sound
Comm>().difficulty_index;
sound_indicator = menu.GetComponent<Sound
Comm>().sound_indicator;
Destroy(menu);

if(!sound_indicator){
gameObject.transform.GetChild(1).gameObject.
GetComponent<AudioSource>().enabled = false;
gameObject.transform.GetChild(2).gameObject.
GetComponent<AudioSource>().enabled = false;
gameObject.transform.GetChild(3).gameObject.
GetComponent<AudioSource>().enabled = false;
gameObject.transform.GetChild(4).gameObject.
GetComponent<AudioSource>().enabled = false;
gameObject.transform.GetChild(5).gameObject.
GetComponent<AudioSource>().enabled = false;
objective_good.transform.parent.GetComponent
<AudioSource>().enabled = false;
Destroy(objective_good);
}

if(param_index == 0){
Destroy(gameObject.transform.GetChild(2).
gameObject);
Destroy(gameObject.transform.GetChild(4).
gameObject);
} else if(param_index == 1){
Destroy(gameObject.transform.GetChild(5).
gameObject);
}

bead_ctr = 0;
gameObject.SetActive(false);
}

void Update () {        }

public int get_difficulty(){
return difficulty_index;
}

public int get_parameter(){
return param_index+3;
}

public void inc_bead_ctr(){
bead_ctr += 1;
}

public int get_bead_ctr(){
return bead_ctr;
}
}
}
```

BrockPlayerDirection.cs

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

public class BrockPlayerDirection :
MonoBehaviour {

public GameObject right_direction;
public GameObject left_direction;
public GameObject initial_post;
public GameObject game_post;
private Vector3 screenPoint;
private float dirNum;
private bool chk_directions_status;

void Start () {
chk_directions_status = true;
dirNum = 0.0f;
}

void Update () {
if(initial_post != null){
screenPoint = GetComponent<Camera>().
WorldToViewportPoint(initial_post.transform.
position);
} else {
screenPoint = GetComponent<Camera>().
WorldToViewportPoint(game_post.transform.
position);
}

bool onScreen = screenPoint.z > 0 &&
screenPoint.x > 0 && screenPoint.x < 1
&& screenPoint.y > 0
&& screenPoint.y < 1;

if(!onScreen){
if(chk_directions_status){
right_direction.SetActive(true);
left_direction.SetActive(true);

if(initial_post != null){
dirNum = AngleDir(transform.forward,
initial_post.transform.position -
transform.position, transform.up);
} else {
dirNum = AngleDir(transform.forward,
game_post.transform.position -
transform.position, transform.up);
}

if(dirNum == 1){
left_direction.transform.localEulerAngles
 = new Vector3(0f, 0f, left_direction.
 transform.localEulerAngles.z);
right_direction.transform.localEulerAngles
 = new Vector3(0f, 0f, right_direction.
 transform.localEulerAngles.z);
} else {
left_direction.transform.localEulerAngles
 = new Vector3(0f, 180f, left_direction.
 transform.
localEulerAngles.z);
right_direction.transform.localEulerAngles
 = new Vector3(0f, 180f, right_direction.
 transform.
localEulerAngles.z);
}

if(transform.localRotation.x > 0.3f){
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 90f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 90f);
} else if(transform.localRotation.x < -0.3f){
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, -90f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, -90f);
} else {
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 0f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 0f);
}
} else {
right_direction.SetActive(false);
left_direction.SetActive(false);
}
} else {
right_direction.SetActive(false);
left_direction.SetActive(false);
}

}

public void directions_status(bool status){
chk_directions_status = status;
}

public float AngleDir(Vector3 fwd, Vector3
targetDir, Vector3 up) {
Vector3 perp = Vector3.Cross(fwd, targetDir);
float dir = Vector3.Dot(perp, up);

if (dir > 0f) {
return 1f;
} else if (dir < 0f) {
return -1f;
} else {
return 0f;
}
}
}
```

CheckpointAudio.cs

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

public class CheckpointAudio
: MonoBehaviour {

public GameObject player;
private float dist;

void Start () {
dist = 0.0f;
}

void Update () {
dist = Vector3.Distance(transform.parent.
position, player.transform.position);
GetComponent<AudioSource>().spatialBlend
 = (dist/100);
}
}
```

CheckpointScriptBrock.cs

```
 namespace GoogleVR.VideoDemo {
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using System;

public class CheckpointScriptBrock
: MonoBehaviour {

public GameObject player;
public GameObject instructions;
public GameObject adjust_controls;
public GameObject speed_controls;
public GameObject objective_good;

void Start () { }

void Update () {
if(transform.position.z <= player.transform.
position.z){
GameObject brock = player.transform.
GetChild(0).GetChild(1).gameObject;
brock.SetActive(true);
speed_controls.SetActive(false);
adjust_controls.SetActive(true);
adjust_controls.transform.GetChild(0).
gameObject.GetComponent<Controller
Script>().show_controls();
if(player.GetComponent<Player
Movement>().get_bead_ctr() < brock.
GetComponent<BrockParam>().
get_parameter()){
GameObject bead = brock.transform.
GetChild(player.GetComponent<Player
Movement>().get_bead_ctr()+1).
gameObject;
Material mat = bead.GetComponent
<Renderer>().material;
String mat_string = "Adjust ";
if((mat+"").IndexOf("Red") >= 0){
mat_string += "Red";
} else if((mat+"").IndexOf("Violet")
        >= 0){
mat_string += "Violet";
} else if((mat+"").IndexOf("Green")
        >= 0){
mat_string += "Green";
```

```
} else if((mat+"").IndexOf("Blue")
        >= 0){
mat_string += "Blue";
} else {
mat_string += "Yellow";
}
instructions.GetComponent<TextMesh
ProUGUI>().text = mat_string + " bead
and make sure an 'X' figure appears";
if(objective_good != null){
objective_good.GetComponent<Canvas
Group>().alpha = 1;
}
bead.transform.GetComponent<Adjust
Bead>().enabled = true;
} else {
instructions.transform.parent.gameObject.
GetComponent<ForceStopScript>().
show_end_game();
}
player.transform.GetComponent<Player
Movement>().enabled = false;
if(gameObject.name == "Initial Check
point"){
Destroy(gameObject);
} else {
gameObject.SetActive(false);
}
}
}
}
}
```

CheckpointScriptLazy.cs

```
namespace GoogleVR.HelloVR {
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using GoogleVR.VideoDemo;
using TMPro;

[RequireComponent(typeof(Collider))]
public class CheckpointScriptLazy :
 MonoBehaviour, TimedInputHandler {
private Vector3 startingPosition;
private Renderer renderer;

public GameObject player;
public GameObject trace;
public GameObject infinity;
public GameObject new_infinity;
public GameObject coin;
public GameObject cube_room;
public GameObject instructions;
public GameObject propeller;
public GameObject speed_controls;
public Material inactiveMaterial;
public Material gazedAtMaterial;
public AudioClip[] next_objective;
private float dist;

void Start() {  }

public void Update(){
dist = Vector3.Distance(transform.position,
player.transform.position);
if(dist < 7.0f){
player.GetComponent<PlaneScript>().
enabled = false;
propeller.GetComponent<PropellerScript>
().enabled = false;
speed_controls.SetActive(false);
propeller.transform.localEulerAngles =
new Vector3(-30f, 90f, -90f);
}
}

public void SetGazedAt(bool gazedAt) {
if (inactiveMaterial != null &&
gazedAtMaterial != null) {
renderer.material = gazedAt ?
gazedAtMaterial : inactiveMaterial;
return;
}
}

public void Reset() {
int sibIdx = transform.GetSiblingIndex();
int numSibs = transform.parent.childCount;
for (int i=0; i<numSibs; i++) {
GameObject sib = transform.parent.
GetChild(i).gameObject;
sib.transform.localPosition =
startingPosition;
sib.SetActive(i == sibIdx);
}
}

public void Recenter() {
#if !UNITY_EDITOR
GvrCardboardHelpers.Recenter();
#else
if (GvrEditorEmulator.Instance != null) {
GvrEditorEmulator.Instance.Recenter();
}
#endif  // !UNITY_EDITOR
}

public void HandleTimedInput(){
player.transform.GetChild(1).
gameObject.GetComponent<Audio
Source>().Play();
new_infinity.SetActive(true);
new_infinity.transform.localPosition =
transform.localPosition;
new_infinity.transform.LookAt
(player.transform);
instructions.GetComponent<TextMesh
ProUGUI>().text = "Trace infinity
symbols: " + (new_infinity.
GetComponent<LazyParam>().
get_checkpoint_ctr()+1);
transform.localPosition = new
Vector3(Random.Range(25.0000f,
175.0000f), 50f, Random.Range
(25.0000f, 175.0000f));
Debug.Log(new_infinity.GetComponent
<LazyParam>().get_checkpoint_ctr());
instructions.transform.parent.
GetComponent<AudioSource>().clip =
next_objective[new_infinity.
GetComponent<LazyParam>().
get_checkpoint_ctr()];
instructions.transform.parent.
GetComponent<AudioSource>().Play();
gameObject.SetActive(false);
}
}
}
```

ClockScript.cs

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

public class ClockScript : MonoBehaviour {

private float minRotation;
private float maxRotation;
private bool chk;

void Start () {
minRotation = -20;
maxRotation = 15;
chk = false;
}

void Update () {
Vector3 currentRotation = transform.
localRotation.eulerAngles;
currentRotation.z = Mathf.Clamp
(currentRotation.z, minRotation,
maxRotation);
transform.localRotation = Quaternion.
Euler (currentRotation);
if(currentRotation.z >= 10.0f){
chk = false;
}

if(currentRotation.z <= 1.0f){
chk = true;
}

if(chk){
transform.Rotate(Vector3.forward,
10 * Time.deltaTime);
} else {
transform.Rotate(Vector3.back, 10
* Time.deltaTime);
}
}
}
```

CoinScript.cs

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

public class CoinScript : MonoBehaviour {

void Start () { }

void Update () {
transform.Rotate(Vector3.up,
200 * Time.deltaTime);
}
}
```

ControllerScript.cs

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

public class ControllerScript :
MonoBehaviour {

public GameObject upper_control;
public GameObject lower_control;
public Sprite upper_controller;
public Sprite lower_controller;
private float start_time;
private bool chk_upper;

void Start () {
chk_upper = true;
start_time = 0.0f;
if(SceneManager.GetActiveScene().name
== "brock_string"){
transform.parent.gameObject.
SetActive(false);
}
}

void Update () {
if(Time.time - start_time > 1.0f){
if(chk_upper){
GetComponent<Image>().sprite
 = upper_controller;
upper_control.SetActive(true);
lower_control.SetActive(false);
chk_upper = false;
} else {
GetComponent<Image>().sprite
 = lower_controller;
lower_control.SetActive(true);
upper_control.SetActive(false);
chk_upper = true;
}
start_time = Time.time;
}
}

public void show_controls(){
start_time = Time.time;
}
}
```

Countdown.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using GoogleVR.VideoDemo;
using TMPro;

public class Countdown : MonoBehaviour {

public GameObject main_camera;
private bool chk_start;

void Start () {
chk_start = false;
}

void Update () {
if(main_camera.GetComponent<TondellPlayer
Direction>().chk_start_game()){
if(!chk_start){
StartCoroutine("Timer");
chk_start = true;
}
}
}

IEnumerator Timer(){
int new_time;
bool chk = true;
while(chk){
new_time = Int32.Parse(GetComponent
<TextMeshProUGUI>().text+"");
new_time -= 1;
GetComponent<TextMeshProUGUI>().text
 = new_time+"";

yield return new WaitForSeconds(1);

if(new_time == 0){
chk = false;
transform.parent.parent.gameObject.
GetComponent<ForceStopScript>().
show_end_game();
}
}
}
```

DeductionScript.cs

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

public class DeductionScript :
MonoBehaviour {

private int deductions;
private int true_score;

void Start () {
deductions = 0;
true_score = 0;
}

void Update () {        }

public void inc_deductions(){
deductions += 1;
}

public void inc_true_score(){
true_score += 2;
}

public int get_deductions(){
return deductions;
}

public int get_true_score(){
return true_score;
}
}
```

DifficultyComm.cs

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

public class DifficultyComm
: MonoBehaviour {

public GameObject main_menu;
public GameObject param_menu;
public GameObject sound_menu;
public Sprite blue_sprite;
public Sprite yellow_sprite;
public GameObject option_1;
public GameObject option_2;
public GameObject option_3;
public int difficulty_index;
public int game_index;
public int param_index;
public bool chk;

void Start(){
chk = false;
}

public void Update(){
if(main_menu != null && param_menu
!= null){game_index = main_menu.
GetComponent<MainComm>().game_index;
param_index = param_menu.GetComponent
<ParamComm>().param_index;
if(game_index == 0){
option_1.GetComponent<Image>().sprite
 = blue_sprite;
option_2.GetComponent<Image>().sprite
 = blue_sprite;
option_3.GetComponent<Image>().sprite
 = blue_sprite;
} else if(game_index == 1){
if(param_index != -1){
Choose_Option(0);
}
} else {
option_1.GetComponent<Image>().sprite
 = yellow_sprite;
option_2.GetComponent<Image>().sprite
 = yellow_sprite;
option_3.GetComponent<Image>().sprite
 = yellow_sprite;
}
}
}

public void Choose_Option(int index) {
difficulty_index = index;
if(index < 3){
sound_menu.GetComponent<Canvas>().
enabled = true;
```

```
} else {
param_menu.GetComponent<ParamComm>
().Reset_Param();
param_menu.GetComponent<Canvas>().
enabled = true;
}
GetComponent<Canvas>().enabled = false;
}
}
```

DifficultyScript.cs

```
 namespace GoogleVR.VideoDemo {
using System;
using UnityEngine;
using UnityEngine.UI;

public class DifficultyScript
 : MonoBehaviour,
TimedInputHandler {
public void HandleTimedInput(){
if(transform.parent.transform.
GetSiblingIndex()        == 2){
if(transform.parent.name == "Back"){
Choose_Option(3);
} else {
Choose_Option(transform.parent.transform
GetSiblingIndex());
}
} else {
Choose_Option(transform.parent.transform.
GetSiblingIndex());
}
}

private void Choose_Option(int index) {
if(transform.parent.name == "Back"){
transform.parent.parent.GetComponent
<DifficultyComm>().Choose_Option(index);
} else {
transform.parent.parent.parent.
GetComponent<DifficultyComm>().
Choose_Option(index);
}
}
}
}
```

DirectionScript.cs

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

public class DirectionScript
 : MonoBehaviour {

private float currentAmount;
private bool chk_gradual;
private bool chk_change;

void Start () {
chk_gradual = true;
chk_change = false;
currentAmount = 0.0f;
}

void Update () {
if(GetComponent<Image>().fillAmount
<= 0f){
if(chk_change){
GetComponent<Image>().fillAmount = 0f;
chk_gradual = true;
chk_change = false;
GetComponent<Image>().fillOrigin = 0;
}
} else if (GetComponent<Image>().
fillAmount >= 1f){
GetComponent<Image>().fillAmount = 1f;
chk_gradual = false;
chk_change = true;
GetComponent<Image>().fillOrigin = 1;
}

if(chk_gradual){
GetComponent<Image>().fillAmount
 += (Time.deltaTime * 1.5f);
} else {
GetComponent<Image>().fillAmount
 -= (Time.deltaTime * 1.5f);
}
}
}
```

DoorScript.cs

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

public class DoorScript
 : MonoBehaviour {

public GameObject player;
private GameObject left_door;
private GameObject right_door;
private float x_left;
private float x_right;
private bool chk_open;
private bool chk_close;

void Start () {
chk_open = false;
chk_close = false;
left_door = transform.GetChild(0).
gameObject;
right_door = transform.GetChild(1).
gameObject;
x_left = 0.0f;
x_right = 0.0f;
}

void Update () {
if(chk_open){
if(transform.parent.parent.gameObject.
name == "Game Arena"){
x_left = left_door.transform.
localPosition.x - Time.deltaTime * 5f;
x_right = right_door.transform.
localPosition.x + Time.deltaTime * 5f;

if(x_left > -57f){
left_door.transform.localPosition =
 new Vector3(x_left, left_door.
 transform.localPosition.y, left_door.
 transform.localPosition.z);
} else {
left_door.transform.localPosition =
 new Vector3(-57f, left_door.
 transform.localPosition.y, left_door.
 transform.localPosition.z);
chk_open = false;
}

if(x_right < 57f){
right_door.transform.localPosition =
 new Vector3(x_right, right_door.
 transform.localPosition.y, right_door.
 transform.localPosition.z);
} else {
right_door.transform.localPosition =
 new Vector3(57f, right_door.
 transform.localPosition.y, right_door.
 transform.localPosition.z);
chk_open = false;
}

}
}

if(chk_close){
left_door.transform.localPosition =
new Vector3(-30f, left_door.transform.
localPosition.y, left_door.transform.
localPosition.z);
right_door.transform.localPosition =
new Vector3(30f, right_door.transform.
localPosition.y, right_door.transform.
localPosition.z);
chk_close = false;
}
}

public void do_open_door(){
chk_open = true;
chk_close = false;
}

public void do_close_door(){
chk_close = true;
chk_open = false;
}
}
```

EliminateFlash.cs

```
  using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using TMPro;
using GoogleVR.VideoDemo;

public class EliminateFlash
 : MonoBehaviour {

public GameObject tondell;
public GameObject score;
```

```
public GameObject main_camera;
public GameObject point_light;
public GameObject input_bar;
public GameObject repress_button;
private int true_score;
private float startTime = 0.0f;
private float holdTime = 3.0f;
private float waitTime = 0.0f;
private bool chk = false;
private bool chk_repress;
private bool chk_allow_press;
private bool chk_faster_play;
private string hold_curr_color;
private string curr_color;

void Start () {
true_score = 0;
point_light = GameObject.Find
("Point light");
curr_color = "";
hold_curr_color = curr_color;
chk_allow_press = true;
chk_repress = false;
chk_faster_play = false;
}

void Update () {
if(main_camera.GetComponent<TondellPlayer
Direction>().chk_start_game()){
if (Input.anyKey) {
if(chk_allow_press){
if(startTime == 0.0f){
curr_color = tondell.GetComponent<Arrow
Lights>().get_curr_color();
hold_curr_color = curr_color;
}
startTime += Time.deltaTime;
input_bar.GetComponent<Image>().
fillAmount = startTime/3;
curr_color = tondell.GetComponent<Arrow
Lights>().get_curr_color();
if(curr_color != hold_curr_color){
chk_allow_press = false;
chk_repress = true;
input_bar.GetComponent<Image>().
fillAmount = 0;
repress_button.GetComponent<Canvas
Group>().alpha = 1;
chk_faster_play = true;
}
if(startTime >= holdTime){
string score_copy = score.GetComponent
<TextMeshProUGUI>().text;
string flash_color = GetComponent
<Light>().color+"";
string arrow_color = "";
chk = false;
for(int i = 0; i < tondell.transform.
childCount-1; i++){
arrow_color = tondell.transform.
GetChild(i).GetChild(0).transform.
GetComponent<Renderer>().material+"";
if(arrow_color.IndexOf("Default") < 0){
if((flash_color.IndexOf("(1.000, 0.169")
        > 0 && arrow_color.IndexOf("Red")
> 0) || (flash_color.IndexOf("(0.122")
> 0 && arrow_color.IndexOf("Green") > 0)
 || (flash_color.IndexOf("(0.918") > 0 &&
arrow_color.IndexOf("Yellow") > 0) ||
(flash_color.IndexOf("(0.000") > 0 &&
arrow_color.IndexOf("Blue") > 0) ||
(flash_color.IndexOf("(0.490") > 0 &&
arrow_color.IndexOf("Violet") > 0) ||
(flash_color.IndexOf("(0.996") > 0 &&
arrow_color.IndexOf("Orange") > 0)){
if(gameObject.name != "Point light"){
int new_score = Int32.Parse(score.
GetComponent<TextMeshProUGUI>
().text+"");
new_score += 1;
score.GetComponent<TextMesh
ProUGUI>().text = new_score+"";
point_light.GetComponent<Eliminate
Flash>().add_true_score();
Destroy(gameObject);
}
}
break;
}
}
chk_allow_press = false;
startTime = 0.0f;
input_bar.GetComponent<Image>().
fillAmount = 0;
chk_repress = true;
repress_button.GetComponent
<CanvasGroup>().alpha = 1;
chk_faster_play = false;
}
} else {
if(chk_faster_play){
repress_button.GetComponent
<AudioSource>().Play();
chk_faster_play = false;
chk_repress = false;
}
waitTime += Time.deltaTime;
if(chk_repress && waitTime
 >= 1.0f){
repress_button.GetComponent
<AudioSource>().Play();
chk_repress = false;
chk_faster_play = false;
}
}
} else {
waitTime = 0.0f;
repress_button.GetComponent
<CanvasGroup>().alpha = 0;
chk_allow_press = true;
input_bar.GetComponent<Image>
().fillAmount = 0;
startTime = 0.0f;
}
}
}

public void add_true_score(){
true_score += 1;
}

public int get_true_score(){
return true_score;
}
}
```

EndGameScript.cs

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

public class EndGameScript :
MonoBehaviour {

private int i;
private float time_start;
private float time_end;

void Start () {
time_start = Time.time;
time_end = 1.2f;
i = 0;
}

void Update () {
if(i < transform.childCount){
if(Time.time-time_start >= time_end){
transform.GetChild(i).gameObject.
SetActive(true);
time_end += 1.2f;
i++;
}
}
}
}
```

ForceStopScript.cs

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

public class ForceStopScript :
MonoBehaviour {

public GameObject player;
public GameObject reticle_pointer;
public GameObject plane;
public GameObject walking;
public GameObject gaze;
public GameObject infinity_symbol;
public GameObject checkpoint;
public GameObject instructions;
public GameObject score;
public GameObject total_score;
public GameObject end_game;
public GameObject two_point;
public GameObject two_point_end;
public GameObject three_point;
public GameObject three_point_end;
public GameObject tondell;
public GameObject true_score;
public GameObject deductions;
public GameObject spawn;
```

```
public GameObject point_light;
public GameObject obstacle_categories;
public GameObject obstacles;
public GameObject coins;
public GameObject initial_coins;
public bool chk;
private Scene scene;

void Start () {
chk = false;
scene = SceneManager.GetActiveScene();
}

void Update () {
if(Input.GetKeyDown(KeyCode.
Joystick1Button4) || Input.
GetKeyDown(KeyCode.
Joystick1Button5) || Input.
GetKey(KeyCode.Space)){
show_end_game();
}
}

public void show_end_game(){
if(scene.name == "lazy_eight"){
if(walking != null){
walking.GetComponent
<WalkingScript>().enabled = false;
}
plane.SetActive(false);
reticle_pointer.SetActive(false);
gaze.SetActive(false);
infinity_symbol.SetActive(false);
checkpoint.SetActive(false);
player.GetComponent<PlaneScript>
().enabled = false;
two_point_end.GetComponent
<TextMeshProUGUI>().text =
"x " + two_point.GetComponent
<TextMeshProUGUI>().text;
three_point_end.GetComponent
<TextMeshProUGUI>().text = "x "
+ three_point.GetComponent
<TextMeshProUGUI>().text;
} else if(scene.name ==
"tondell_arrows"){
tondell.GetComponent<Arrow
Lights>().disable_coroutine();
for(int i = 0; i < tondell.transform.
childCount; i++){
tondell.transform.GetChild(i).
gameObject.SetActive(false);
}
end_game.GetComponent
<EndGameScript>().enabled
 = true;
true_score.GetComponent
<TextMeshProUGUI>().text
= "x " + point_light.
GetComponent<EliminateFlash>
().get_true_score()+"";
deductions.GetComponent
<TextMeshProUGUI>().text
= "x " + spawn.GetComponent
<SpawnFlash>().
get_deductions()+"";
spawn.SetActive(false);
} else {
player.transform.GetChild(0).
GetChild(1).gameObject.
SetActive(false);
player.GetComponent
<PlayerMovement>().enabled
 = false;
player.GetComponent
<SummonPlayerScript>().
enabled = false;
obstacles.SetActive(false);
coins.SetActive(false);
if(initial_coins != null){
initial_coins.SetActive(false);
}
end_game.GetComponent
<EndGameScript>().enabled
= true;

true_score.GetComponent
<TextMeshProUGUI>().text
= "x " + obstacle_categories.
GetComponent<Deduction
Script>().get_true_score()+"";
deductions.GetComponent
<TextMeshProUGUI>().text
= "x " + obstacle_categories.
GetComponent<Deduction
Script>().get_deductions()+"";
}
instructions.transform.parent.
gameObject.SetActive(false);
total_score.GetComponent
<TextMeshProUGUI>().text
= "x " + score.GetComponent
<TextMeshProUGUI>().text;
end_game.GetComponent
<EndGameScript>().enabled
= true;
}
}
```

ForwardObstacles.cs

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

public class ForwardObstacles
 : MonoBehaviour {

private float multiplier;

void Start () {
multiplier = 1.0f;
}

void Update () {
if(gameObject.name.IndexOf("Clone") >= 0){
transform.localPosition += transform.forward
 * 0.8f * multiplier;
}
}

public void update_multiplier
(float speed_multiplier){
multiplier = speed_multiplier;
}
}
```

GetCoin.cs

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

public class GetCoin : MonoBehaviour {

public GameObject score;
public GameObject obstacle_category;
public GameObject player;

void Start () { }

void Update () {        }

void OnCollisionEnter (Collision col){
if(col.gameObject.name == "Player"){
int new_score = Int32.Parse(score.
GetComponent<TextMeshProUGUI>().text+"");
new_score += 2;
obstacle_category.GetComponent<DeductionScript>
().inc_true_score();
score.GetComponent<TextMeshProUGUI>().
text = new_score+"";
player.GetComponent<PlayerMovement>().
audio_coin();

if(gameObject.name != "Coin 6"){
gameObject.SetActive(false);
}
}
}
}
```

GoToMainScript.cs

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

public class GoToMainScript : MonoBehaviour {

void Start () { }

void Update () {
if(Input.anyKeyDown){
Application.LoadLevel("loading");
}
}
}
```

GvrReticlePointer.cs

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using UnityEngine.SceneManagement;
using System;
using TMPro;
```

```
public class GvrReticlePointer
 : GvrBasePointer {
public const float RETICLE_MIN_
INNER_ANGLE = 0.0f;

public const float RETICLE_MIN_
OUTER_ANGLE = 0.3f;

public const float RETICLE_
GROWTH_ANGLE = 0.45f;

public const float RETICLE_
DISTANCE_MIN = 0.45f;

public float maxReticle
Distance = 10.0f;

public int reticleSegments
 = 20;

public float reticleGrowth
Speed = 8.0f;

public GameObject infinity;
public GameObject score;
public GameObject two_point;
public GameObject three_point;
public GameObject gaze_time;
public GameObject streak_time;
private GameObject gazedAt;
private float currentAmount;
private float currentAmountStreak;
private float gazeStartTime;
private float duration = 1.5f;
private float t_allowance = 0.0f;
private float hold_time;
private bool chk = false;
private bool chk_immed_prev = false;
private bool chk_immed_prev_allow = false;

[Range(-32767, 32767)]
public int reticleSortingOrder = 32767;

public Material MaterialComp {
private get; set; }

public float ReticleInnerAngle {
get; private set; }

public float ReticleOuterAngle {
get; private set; }

public float ReticleDistanceInMeters {
get; private set; }

public float ReticleInnerDiameter {
get; private set; }

public float ReticleOuterDiameter {
get; private set; }

public override float MaxPointerDistance {
get { return maxReticleDistance; } }

public override void OnPointerEnter
(RaycastResult raycastResultResult,
bool isInteractive) {
SetPointerTarget(raycastResultResult.
worldPosition, isInteractive);
gazedAt = raycastResultResult.gameObject;
gazeStartTime = Time.time;
}

public override void OnPointerHover
(RaycastResult raycastResultResult,
bool isInteractive) {
SetPointerTarget(raycastResultResult.
worldPosition, isInteractive);
if(gazedAt != null && isInteractive
&& gazeStartTime > 0f){
Scene scene = SceneManager.
GetActiveScene();
hold_time = Time.time - gazeStartTime;
if(scene.name != "main_menu"){
if(scene.name == "lazy_eight"){
int sibIdx = raycastResultResult.
gameObject.transform.GetSiblingIndex();
if(sibIdx >= 0){
if(infinity.active){
if(sibIdx-1 < 0){
chk = true;
} else {
GameObject parentPrev = raycastResultResult.
gameObject.transform.parent.gameObject;
GameObject prevSib =  parentPrev.transform.
GetChild(sibIdx-1).gameObject;
if(!prevSib.active){
chk = true;
} else {
chk = false;
}
}
} else {
chk = true;
}
if(chk){
if(!chk_immed_prev_allow){
currentAmount = (Time.time -
gazeStartTime)/2;
gaze_time.GetComponent<Image>().
fillAmount = currentAmount;
} else {
currentAmount = ((Time.time -
t_allowance)/2) + 0.625f;
gaze_time.GetComponent<Image>().
fillAmount = currentAmount;
}

if(((Time.time - gazeStartTime > 2.25f
&& !chk_immed_prev) || (chk_immed_
prev && Time.time - t_allowance > 1f))
&& ExecuteEvents.CanHandleEvent
<TimedInputHandler>(gazedAt)){
if(infinity.active){
int new_score = Int32.Parse(score.
GetComponent<TextMeshProUGUI>
().text+"");
if(chk_immed_prev){
new_score += 3;
three_point.GetComponent<Text
MeshProUGUI>().text = (Int32.Parse
(three_point.GetComponent<TextMesh
ProUGUI>().text+"")+1)+"";
} else {
new_score += 2;
two_point.GetComponent<TextMesh
ProUGUI>().text = (Int32.Parse
(two_point.GetComponent<TextMesh
ProUGUI>().text+"")+1)+"";
}
score.GetComponent<TextMesh
ProUGUI>().text = new_score+"";
}

gazeStartTime = -1f;
if(infinity.active){
streak_time.GetComponent
<Image>().fillAmount = 1;
t_allowance = Time.time;
chk_immed_prev_allow = true;
}
ExecuteEvents.Execute(gazedAt,
null, (TimedInputHandler handler,
BaseEventData data) => handler.
HandleTimedInput());
}
} else {
GetComponent<Renderer>().
material.color = Color.red;
}
}
}
} else {
currentAmount = (Time.time -
gazeStartTime);
gaze_time.GetComponent<Image>
().fillAmount = currentAmount;
if((Time.time - gazeStartTime >
1.25f) && ExecuteEvents.
CanHandleEvent<TimedInputHandler>
(gazedAt)){
gazeStartTime = -1f;
ExecuteEvents.Execute(gazedAt, null,
(TimedInputHandler handler,
BaseEventData data) => handler.
HandleTimedInput());
}
}
}
}

public override void OnPointerExit
(GameObject previousObject) {
if(gaze_time != null){
gaze_time.GetComponent<Image>().
fillAmount = 0;
}
GetComponent<Renderer>().material.
color = Color.white;
}

public override void
OnPointerClickDown() {}

public override void
OnPointerClickUp() {}
```

```
public override void GetPointerRadius
(out float enterRadius, out
float exitRadius) {
float min_inner_angle_radians
 = Mathf.Deg2Rad * RETICLE_MIN_
 INNER_ANGLE;
float max_inner_angle_radians
 = Mathf.Deg2Rad * (RETICLE_MIN_
 INNER_ANGLE + RETICLE_GROWTH
 _ANGLE);

enterRadius = 2.0f * Mathf.Tan(min_
inner_angle_radians);
exitRadius = 2.0f * Mathf.Tan(max_
inner_angle_radians);
}

public void UpdateDiameters() {
ReticleDistanceInMeters =
Mathf.Clamp(ReticleDistanceInMeters,
RETICLE_DISTANCE_MIN, maxReticle
Distance);

float inner_half_angle_radians =
Mathf.Deg2Rad * ReticleInnerAngle
* 0.5f;
float outer_half_angle_radians =
Mathf.Deg2Rad * ReticleOuterAngle
 * 0.5f;

float inner_diameter = 2.0f *
Mathf.Tan(inner_half_angle_radians);
float outer_diameter = 2.0f *
Mathf.Tan(outer_half_angle_radians);

ReticleInnerDiameter = inner_diameter;
ReticleOuterDiameter = outer_diameter;

if(chk_immed_prev_allow){
if(infinity.active){
currentAmountStreak = 1-(Time.time -
t_allowance);
streak_time.GetComponent<Image>
().fillAmount = currentAmountStreak;
}
if(Time.time - t_allowance > 1f){
chk_immed_prev = false;
t_allowance = 0.0f;
chk_immed_prev_allow = false;
} else {
chk_immed_prev = true;
}
}

MaterialComp.SetFloat("_Inner
Diameter", ReticleInnerDiameter
* ReticleDistanceInMeters);
MaterialComp.SetFloat("_Outer
Diameter", ReticleOuterDiameter
* ReticleDistanceInMeters);
MaterialComp.SetFloat("_Distance
InMeters", ReticleDistanceInMeters);
}

void Awake() {
ReticleInnerAngle = RETICLE_MIN_
INNER_ANGLE + RETICLE_GROWTH_
ANGLE;
ReticleOuterAngle = RETICLE_MIN_
OUTER_ANGLE + RETICLE_GROWTH_
ANGLE;

}

protected override void Start() {
base.Start();

Renderer rendererComponent
 = GetComponent<Renderer>();
rendererComponent.sortingOrder
 = reticleSortingOrder;

MaterialComp = rendererComponent
.material;
gazeStartTime = -1f;
gazedAt = null;

CreateReticleVertices();
}

void Update() {
UpdateDiameters();
}

private bool SetPointerTarget
(Vector3 target, bool interactive) {
if (base.PointerTransform == null) {
Debug.LogWarning("Cannot operate
on a null pointer transform");
return false;
}

Vector3 targetLocalPosition = base.
PointerTransform.InverseTransform
Point(target);

ReticleDistanceInMeters =
Mathf.Clamp(targetLocalPosition.z,
RETICLE_DISTANCE_MIN, maxReticle
Distance);
ReticleInnerAngle = RETICLE_MIN_
INNER_ANGLE + RETICLE_GROWTH_
ANGLE;
ReticleOuterAngle = RETICLE_MIN_
OUTER_ANGLE + RETICLE_GROWTH_
ANGLE;
return true;
}

private void CreateReticleVertices() {
Mesh mesh = new Mesh();
gameObject.AddComponent
<MeshFilter>();
GetComponent<MeshFilter>().
mesh = mesh;

int segments_count = reticleSegments;
int vertex_count = (segments_count
+1)*2;

#region Vertices

Vector3[] vertices = new Vector3
[vertex_count];

const float kTwoPi = Mathf.PI
* 2.0f;
int vi = 0;
for (int si = 0; si <= segments_
count; ++si) {
float angle = (float)si / (float)
(segments_count) * kTwoPi;

float x = Mathf.Sin(angle);
float y = Mathf.Cos(angle);

vertices[vi++] = new Vector3
(x, y, 0.0f);
vertices[vi++] = new Vector3
(x, y, 1.0f);
}
#endregion

#region Triangles
int indices_count = (segments
_count+1)*3*2;
int[] indices = new int[indices
_count];

int vert = 0;
int idx = 0;
for (int si = 0; si < segments
_count; ++si) {
indices[idx++] = vert+1;
indices[idx++] = vert;
indices[idx++] = vert+2;

indices[idx++] = vert+1;
indices[idx++] = vert+2;
indices[idx++] = vert+3;

vert += 2;
}
#endregion

mesh.vertices = vertices;
mesh.triangles = indices;
mesh.RecalculateBounds();
#if !UNITY_5_5_OR_NEWER
mesh.Optimize();
}
}
```

InfinityScript.cs

```
 namespace GoogleVR.HelloVR {
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using GoogleVR.VideoDemo;
using TMPro;

[RequireComponent(typeof(Collider))]
public class InfinityScript : MonoBehaviour,
TimedInputHandler {

public Material inactiveMaterial;
```

```
public Material gazedAtMaterial;
public GameObject trace;
public GameObject player;
public GameObject instructions;
public GameObject reticle_pointer;
public GameObject streak_time;
public GameObject checkpoint;
public GameObject propeller;
public GameObject speed_controls;
public AudioClip[] next_objective;
private Material coin_face_brighten;
private Material coin_face_darken;
private Material[] mat;
private Vector3 startingPosition;
private Renderer renderer;
private int loop_count;

void Start() {
startingPosition = transform.localPosition;
renderer = GetComponent<Renderer>();
SetGazedAt(false);

coin_face_brighten = Resources.Load
("coin_face_brighten", typeof(Material))
 as Material;
coin_face_darken = Resources.Load
("coin_face_darken", typeof(Material))
 as Material;
}

public void Update(){    }

public void SetGazedAt(bool gazedAt) {

public void Reset() {
int sibIdx = transform.GetSiblingIndex();
int numSibs = transform.parent.childCount;
for (int i=0; i<numSibs; i++) {
GameObject sib = transform.parent.
GetChild(i).gameObject;
sib.transform.localPosition = startingPosition;
sib.SetActive(i == sibIdx);
}
}

public void Recenter() {
#if !UNITY_EDITOR
GvrCardboardHelpers.Recenter();
#else
if (GvrEditorEmulator.Instance != null) {
GvrEditorEmulator.Instance.Recenter();
}
#endif  // !UNITY_EDITOR
}

public void HandleTimedInput(){
int sibIdx = transform.GetSiblingIndex();
loop_count = transform.parent.GetComponent
<LazyParam>().get_checkpoint_ctr()+1;
if(sibIdx > 0){
GameObject prevSib =  transform.parent.
GetChild(sibIdx-1).gameObject;
if(!prevSib.active){
int numSibs = transform.parent.childCount;
if(numSibs-1 == sibIdx){
transform.parent.GetComponent<LazyParam>
().update_checkpoint_ctr();
loop_count = transform.parent.GetComponent
<LazyParam>().get_checkpoint_ctr()+1;
instructions.GetComponent<TextMeshProUGUI>
().text = "Trace infinity symbols: " +
(loop_count);
if(loop_count > 0){
instructions.transform.parent.GetComponent
<AudioSource>().clip = next_objective
[loop_count-1];
instructions.transform.parent.GetComponent
<AudioSource>().Play();
}
for(int i = 0; i < numSibs; i++){
transform.parent.GetChild(i).gameObject.
SetActive(true);
if(i != 0){
mat = transform.parent.GetChild(i).
gameObject.GetComponent<Renderer>
().materials;
mat[1] = coin_face_darken;
transform.parent.GetChild(i).gameObject.
GetComponent<Renderer>().materials
= mat;
}
}
if(loop_count == 0){
transform.parent.GetComponent
<LazyParam>().inc_checkpoint_ctr();
if(transform.parent.GetComponent
<LazyParam>().chk_checkpoint_ctr()){
player.GetComponent<PlaneScript>
().enabled = false;
instructions.transform.parent.
gameObject.GetComponent
<ForceStopScript>().show_end_game();
} else {
player.GetComponent<PlaneScript>
().enabled = true;
propeller.GetComponent<Propeller
Script>().enabled = true;
speed_controls.SetActive(true);
streak_time.GetComponent<Image>
().fillAmount = 0;
}
transform.parent.gameObject.
SetActive(false);
checkpoint.SetActive(true);
}
} else {
mat = transform.parent.GetChild
(sibIdx+1).gameObject.GetComponent
<Renderer>().materials;
mat[1] = coin_face_brighten;
transform.parent.GetChild(sibIdx+1).
gameObject.GetComponent<Renderer>
().materials = mat;
gameObject.SetActive(false);
}
}
} else {
mat = transform.parent.GetChild
(sibIdx+1).gameObject.GetComponent
<Renderer>().materials;
mat[1] = coin_face_brighten;
}transform.parent.GetChild(sibIdx+1).
gameObject.GetComponent<Renderer>
().materials = mat;
gameObject.SetActive(false);
}
transform.parent.gameObject.
GetComponent<AudioSource>().Play();
}
}
}
```

InitiateFlight.cs

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

public class InitiateFlight :
MonoBehaviour, TimedInputHandler {

public GameObject player;
public GameObject instructions;
public GameObject checkpoint;
public Transform vrCamera;
public AudioClip next_objective;

public void HandleTimedInput(){
checkpoint.SetActive(true);
transform.parent = vrCamera;
transform.eulerAngles = vrCamera.eulerAngles;

vrCamera.parent = player.transform;
vrCamera.position = player.transform.position;

transform.localPosition = new Vector3(vrCamera.
localPosition.x, vrCamera.localPosition.y-2f,
vrCamera.localPosition.z-2f);

player.GetComponent<PlaneScript>().enabled = true;

Destroy(GetComponent<BoxCollider>());
Destroy(GetComponent<EventTrigger>());

instructions.GetComponent<TextMeshProUGUI>().
text = "Find the checkpoint";
instructions.transform.parent.GetComponent
<AudioSource>().clip = next_objective;
instructions.transform.parent.GetComponent
<AudioSource>().Play();
Destroy(this);
}
}
```

LastCoinTrigger.cs

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

public class LastCoinTrigger
: MonoBehaviour {

public GameObject player;
private float prev_pos;
```

```
private bool chk;

void Start () {
chk = false;
}

void Update () {
if(Math.Abs(transform.position.z - player.
transform.position.z) < 5.0f){
chk = true;
prev_pos = player.transform.position.z;
}

if(chk){
if(Math.Abs(player.transform.position.z
- prev_pos) > 2.0f){
transform.parent.GetComponent
<RegenerateCoins>().do_regenerate();
chk = false;
}
}
}
}
```

LazyParam.cs

```
 namespace GoogleVR.VideoDemo {
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class LazyParam : MonoBehaviour {

public GameObject player;
public GameObject instructions;
public AudioClip next_objective;
private int param_index;
private int checkpoint_ctr;
private int hold_checkpoint_ctr;
private bool sound_indicator;

void Start () {
GameObject menu = GameObject.Find("Sound
Menu");
param_index = menu.GetComponent<Sound
Comm>().param_index;
sound_indicator = menu.GetComponent<Sound
Comm>().sound_indicator;
Destroy(menu);

if(!sound_indicator){
instructions.transform.parent.GetComponent
<AudioSource>().enabled = false;
}

checkpoint_ctr = 0;
hold_checkpoint_ctr = checkpoint_ctr;

gameObject.SetActive(false);
}
void Update () {        }

public void inc_checkpoint_ctr(){
checkpoint_ctr += 1;
hold_checkpoint_ctr = checkpoint_ctr;
instructions.GetComponent<TextMesh
ProUGUI>().text = "Find the checkpoint";
instructions.transform.parent.
GetComponent<AudioSource>().clip
 = next_objective;
instructions.transform.parent.
GetComponent<AudioSource>().Play();
}

public bool chk_checkpoint_ctr(){
return checkpoint_ctr == param_index+5;
}

public int get_checkpoint_ctr(){
return hold_checkpoint_ctr;
}

public void update_checkpoint_ctr(){
hold_checkpoint_ctr -= 1;
}
}
}
```

LazyPlayerDirection.cs

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

public class LazyPlayerDirection
 : MonoBehaviour {

public GameObject right_direction;
public GameObject left_direction;
public GameObject plane;
public GameObject checkpoint;
public GameObject infinity;
private Vector3 screenPoint;
private float dirNum;

void Start () {
dirNum = 0.0f;
}

void Update () {
if(transform.parent.gameObject.name
== "Walking"){
screenPoint = GetComponent<Camera>
().WorldToViewportPoint(plane.transform.
position);
} else {
if(infinity.active){
screenPoint = GetComponent<Camera>().
WorldToViewportPoint(infinity.transform.
position);
} else {
screenPoint = GetComponent<Camera>().
WorldToViewportPoint(checkpoint.transform.
position);
}
}

bool onScreen = screenPoint.z > 0 &&
screenPoint.x > 0 && screenPoint.x < 1
&& screenPoint.y > 0 && screenPoint.y < 1;

if(!onScreen){
right_direction.SetActive(true);
left_direction.SetActive(true);
if(transform.parent.gameObject.name
== "Walking"){
dirNum = AngleDir(transform.forward,
plane.transform.position - transform.
position, transform.up);
} else if(checkpoint.active){
dirNum = AngleDir(transform.forward,
checkpoint.transform.position -
 transform.position, transform.up);
} else if(infinity.active){
dirNum = AngleDir(transform.forward,
infinity.transform.position - transform.
position, transform.up);
}

if(dirNum == 1){
left_direction.transform.localEulerAngles
 = new Vector3(0f, 0f, left_direction.
 transform.localEulerAngles.z);
right_direction.transform.localEulerAngles
 = new Vector3(0f, 0f, right_direction.
 transform.localEulerAngles.z);
} else {
left_direction.transform.localEulerAngles
 = new Vector3(0f, 180f, left_direction.
 transform.localEulerAngles.z);
right_direction.transform.localEulerAngles
 = new Vector3(0f, 180f, right_direction.
 transform.localEulerAngles.z);
}

if(transform.localRotation.x > 0.08f){
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 90f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, right_direction.
 transform.localEulerAngles.y, 90f);
} else if(transform.localRotation.x
< -0.08f){
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, -90f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, right_direction.
 transform.localEulerAngles.y, -90f);
} else {
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 0f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, right_direction.
 transform.localEulerAngles.y, 0f);
}
} else {
right_direction.SetActive(false);
left_direction.SetActive(false);
}
}

public float AngleDir(Vector3 fwd,
Vector3 targetDir, Vector3 up) {
Vector3 perp = Vector3.Cross(fwd,
```

```
 targetDir);
float dir = Vector3.Dot(perp, up);

if (dir > 0f) {
return 1f;
} else if (dir < 0f) {
return -1f;
} else {
return 0f;
}
}
}
```

MainComm.cs

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

public class MainComm : MonoBehaviour {

public GameObject play_game;
public GameObject param_menu;
public GameObject tutorials;
public GameObject directions;
public int game_index;

void Start(){
game_index = 0;
}

public void Choose_Scene(int index){
if(index < 3){
game_index = index;
int play_indicator = play_game.
GetComponent<PlayComm>().
play_indicator;
if(play_indicator == 1){
param_menu.GetComponent<Canvas>
().enabled = true;
} else if(play_indicator == 2){
for(int i = 0; i < tutorials.transform.
childCount; i++){
if(i == index){
tutorials.transform.GetChild(i).
gameObject.SetActive(true);
} else {
tutorials.transform.GetChild(i).
gameObject.SetActive(false);
directions.SetActive(false);
}
}
}
} else {
play_game.GetComponent
<Canvas>().enabled = true;
}
GetComponent<Canvas>().
enabled = false;
}
}
```

MainScript.cs

```
 namespace GoogleVR.VideoDemo {
using System;
using UnityEngine;
using UnityEngine.UI;

public class MainScript
: MonoBehaviour, TimedInputHandler {

public void HandleTimedInput(){
Choose_Scene(transform.parent.transform.
GetSiblingIndex()-1);
}

private void Choose_Scene(int index) {
transform.parent.parent.GetComponent
<MainComm>().Choose_Scene(index);
}
}
}
```

ObstacleAttack.cs

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

public class ObstacleAttack
 : MonoBehaviour {

public GameObject player;
public GameObject score;
public GameObject obstacle_category;
private float prev_pos;
private bool chk;
private bool chk_1;
private bool chk_2;

void Start () {
chk = false;
chk_1 = true;
chk_2 = true;
}

void Update () {
if(transform.parent.gameObject.
name.IndexOf("Clone") >= 0){
if(chk_1){
GameObject parent_spawn =
transform.parent.parent.gameObject;
transform.parent.gameObject.
transform.localPosition =
parent_spawn.transform.localPosition;

transform.localScale = new
Vector3(transform.localScale.x*
5f,transform.localScale.y*5f,
transform.localScale.z*5f);
chk_1 = false;
transform.parent.gameObject.
transform.LookAt(player.transform);
}

if(Math.Abs(transform.position.z
 - player.transform.position.z) >
 50.0f && !chk){
transform.parent.gameObject.
transform.LookAt(player.transform);
}

if(Math.Abs(transform.position.z
- player.transform.position.z)
 < 10.0f){
if(chk_2){
GetComponent<AudioSource>().
Play();
chk_2 = false;
}
chk = true;
prev_pos = transform.position.z;
}

if(chk){
if(Math.Abs(transform.position.z
 - prev_pos) > 5.0f){
if(!GetComponent<AudioSource>
().isPlaying){
Destroy(transform.parent.
gameObject);
}
}
}
}
}

void OnCollisionEnter(Collision
collision){
if(collision.gameObject.name
 == "Player"){
int new_score = Int32.Parse
(score.GetComponent<TextMesh
ProUGUI>().text+"");
new_score -= 1;
obstacle_category.GetComponent
<DeductionScript>().
inc_deductions();
if(new_score <= 0){
score.GetComponent<TextMesh
ProUGUI>().text = "0";
} else {
score.GetComponent<TextMesh
ProUGUI>().text = new_score+"";
}
player.GetComponent<Player
Movement>().audio_obstacle();
Destroy(gameObject);
}
}
}
```

ParamComm.cs

```
  using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using GoogleVR.VideoDemo;
using TMPro;

public class ParamComm : MonoBehaviour {

public GameObject main_menu;
public GameObject difficulty_menu;
public GameObject option_1;
```

```
public GameObject option_2;
public GameObject option_3;
public GameObject message;
public Sprite blue_sprite;
public Sprite yellow_sprite;
public Sprite green_sprite;
public int param_index;
public int game_index;

void Start(){
param_index = -1;
}

public void Update(){
game_index = main_menu.GetComponent
<MainComm>().game_index;
if(game_index == 0){
message.GetComponent<TextMeshProUGUI>
().text = "Choose number of arrows";
option_1.transform.GetChild(0).GetComponent
<TextMeshProUGUI>().text = "4";
option_2.transform.GetChild(0).GetComponent
<TextMeshProUGUI>().text = "5";
option_3.transform.GetChild(0).GetComponent
<TextMeshProUGUI>().text = "6";

option_1.GetComponent<Image>().sprite
 = blue_sprite;
option_2.GetComponent<Image>().sprite
 = blue_sprite;
option_3.GetComponent<Image>().sprite
 = blue_sprite;

gameObject.transform.GetChild(2).
gameObject.SetActive(false);
} else if(game_index == 1){
message.GetComponent<TextMeshPro
UGUI>().text = "Choose number of
checkpoints";
option_1.transform.GetChild(0).
GetComponent<TextMeshProUGUI>().
text = "5";
option_2.transform.GetChild(0).
GetComponent<TextMeshProUGUI>().
text = "6";
option_3.transform.GetChild(0).
GetComponent<TextMeshProUGUI>().
text = "7";

option_1.GetComponent<Image>().
sprite = green_sprite;
option_2.GetComponent<Image>().
sprite = green_sprite;
option_3.GetComponent<Image>().
sprite = green_sprite;

gameObject.transform.GetChild(2).
gameObject.SetActive(true);
} else {
message.GetComponent<TextMesh
ProUGUI>().text = "Choose number
 of beads";
option_1.transform.GetChild(0).
GetComponent<TextMeshProUGUI>().
text = "3";
option_2.transform.GetChild(0).
GetComponent<TextMeshProUGUI>().
text = "4";
option_3.transform.GetChild(0).
GetComponent<TextMeshProUGUI>().
text = "5";

option_1.GetComponent<Image>().
sprite = yellow_sprite;
option_2.GetComponent<Image>().
sprite = yellow_sprite;
option_3.GetComponent<Image>().
sprite = yellow_sprite;

gameObject.transform.GetChild(2).
gameObject.SetActive(false);
}
}

public void Choose_Option
(int index) {
param_index = index;
if(index < 6){
int game_index = main_menu.
GetComponent<MainComm>().
game_index;
if(game_index != 1){
difficulty_menu.GetComponent
<Canvas>().enabled = true;
}
} else {
param_index = -1;
main_menu.GetComponent
<Canvas>().enabled = true;
}
GetComponent<Canvas>().
enabled = false;
}

public void Reset_Param(){
param_index = -1;
}
}
```

ParamScript.cs

```
 namespace GoogleVR.VideoDemo {
using System;
using UnityEngine;

public class ParamScript
: MonoBehaviour, TimedInputHandler {

public void HandleTimedInput(){
if(transform.parent.transform.
GetSiblingIndex() == 3){
Choose_Option(6);
} else {
if(transform.parent.parent.
transform.GetSiblingIndex()
== 1){
Choose_Option(transform.
parent.transform.GetSiblingIndex());
} else {
Choose_Option(transform.
parent.transform.GetSiblingIndex()+3);
}
}
}

private void Choose_Option(int index) {
if(transform.parent.transform.
GetSiblingIndex() == 3){
transform.parent.parent.
GetComponent<ParamComm>().
Choose_Option(index);
} else {
transform.parent.parent.parent.
GetComponent<ParamComm>().
Choose_Option(index);
}
}
}
}
```

PlaneAudio.cs

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

public class PlaneAudio
 : MonoBehaviour {

public Transform camera;
private float dist;

void Start () {
dist = 0.0f;
}

void Update () {
if(transform.localPosition.z
== -2){
GetComponent<AudioSource>
().spatialBlend = 0.9f;
} else {
dist = Vector3.Distance(camera.
position, transform.position);
GetComponent<AudioSource>().
spatialBlend = (dist/100);
if(dist/100 < 0.8f){
GetComponent<AudioSource>().
spatialBlend = 0.8f;
}
}
}
}
```

PlaneScript.cs

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

public class PlaneScript
 : MonoBehaviour {

public GameObject speed;
public Transform vrCamera;
private float multiplier;

void Start () {
Destroy(GameObject.
```

```
Find("Walking"));
transform.position = new
Vector3(100, 50, 100);
multiplier = 1.0f;
}

void Update () {
if(Input.GetKeyDown(KeyCode.
Joystick1Button0) || Input.
GetKeyDown(KeyCode.A)){
if(multiplier <= 2f){
multiplier += 0.1f;
speed.GetComponent<SpeedShow>
().show_speed(multiplier);
} else {
speed.GetComponent<SpeedShow>
().show_speed(3.0f);
}
} else if(Input.GetKeyDown(KeyCode.
Joystick1Button3) || Input.
GetKeyDown(KeyCode.D)){
if(multiplier >= 1.1f){
multiplier -= 0.1f;
speed.GetComponent<SpeedShow>
().show_speed(multiplier);
} else {
speed.GetComponent<SpeedShow>
().show_speed(0.0f);
}
}

transform.localPosition += vrCamera.
transform.forward * 0.2f * multiplier;
transform.localPosition = new
Vector3(transform.position.x, 50,
transform.position.z);

if(transform.position.x >= 200.0f){
transform.localPosition = new
Vector3(200.0f, 50, transform.
position.z);
} else if(transform.position.x
<= 0.0f){
transform.localPosition = new
Vector3(0.0f, 50, transform.
position.z);
}

if(transform.position.z >=
200.0f){
transform.localPosition = new
Vector3(transform.position.x, 50,
200.0f);
} else if(transform.position.z
<= 0.0f){
transform.localPosition = new
Vector3(transform.position.x, 50,
0.0f);
}
}
}
```

PlayComm.cs

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

public class PlayComm : MonoBehaviour {

public GameObject main_menu;
public int play_indicator;

void Start () {
play_indicator = 0;
}

void Update () {        }

public void Choose_Option(int index){
play_indicator = index;
main_menu.GetComponent<Canvas>().
enabled = true;
GameObject menu = main_menu.
transform.GetChild(0).gameObject;
string option = "";
if(play_indicator == 1){
option = "[PLAY]\n";
} else {
option = "[TUTORIALS]\n";
}
menu.GetComponent<TextMesh
ProUGUI>().text = option + "Choose
a visual training technique";
GetComponent<Canvas>().
enabled = false;
}
}
```

PlayerMovement.cs

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

public class PlayerMovement
: MonoBehaviour {

public GameObject initial_door;
public GameObject game_front_door;
public GameObject game_back_door;
public GameObject brock;
public GameObject initial_arena;
public GameObject game_arena;
public GameObject front_door_checkpoint;
public GameObject coins;
public GameObject obstacles;
public GameObject speed;
public Transform vrCamera;
private int bead_ctr;
private int bead_ctr_holder;
private float z_front;
private float z_back;
private float left_chk_limit;
private float right_chk_limit;
private float x_left;
private float x_right;
private float y_fix;
private float multiplier;
private bool chk;
private bool chk_1;
private bool chk_2;
private bool chk_coin;
private bool chk_obstacle;
private bool chk_inside_checkpoint;

void Start () {
y_fix = transform.position.y;
z_front = -625f;
z_back = -740f;
left_chk_limit = -65f;
right_chk_limit = -15f;
x_left = -125f;
x_right = 45f;
chk = true;
chk_1 = true;
chk_2 = true;
bead_ctr = 0;
bead_ctr_holder = bead_ctr;
chk_coin = false;
chk_inside_checkpoint = true;
multiplier = 1.0f;
}

void Update () {
if(Input.GetKeyDown(KeyCode.
Joystick1Button0) || Input.
GetKeyDown(KeyCode.A)){
if(multiplier <= 2.0f){
multiplier += 0.1f;
speed.GetComponent<SpeedShow>
().show_speed(multiplier);
for(int i = 0; i < obstacles.
transform.childCount; i++){
for(int j = 0; j < obstacles.
transform.GetChild(i).transform.
childCount; j++){
obstacles.transform.GetChild(i).
gameObject.transform.GetChild(j).
gameObject.GetComponent
<ForwardObstacles>().
update_multiplier(multiplier);
}
}
} else {
speed.GetComponent
<SpeedShow>().show_speed(3.0f);
}
} else if(Input.GetKeyDown(
KeyCode.Joystick1Button3) || Input.
GetKeyDown(KeyCode.D)){
if(multiplier >= 1.1f){
multiplier -= 0.1f;
speed.GetComponent<SpeedShow>
().show_speed(multiplier);
} else {
speed.GetComponent<SpeedShow>
().show_speed(0.0f);
}
}

if(chk_coin){
transform.GetChild(0).gameObject.
GetComponent<AudioSource>().Play();
chk_coin = false;
}
```

```
if(chk_obstacle){
transform.GetChild(0).GetChild(1).
gameObject.GetComponent
<AudioSource>().Play();
chk_obstacle = false;
}

if(bead_ctr > 0 && (bead_ctr_
holder != bead_ctr)){
z_back = -585f;
z_front = -150f;
chk_1 = true;
chk_2 = true;
chk_inside_checkpoint = true;
coins.SetActive(true);
game_back_door.GetComponent
<DoorScript>().do_close_door();
if(bead_ctr > 1){
front_door_checkpoint.
SetActive(true);
}
if(bead_ctr == 1){
Destroy(initial_arena);
game_arena.SetActive(true);
game_back_door.GetComponent
<DoorScript>().do_open_door();
}
}

if(chk_1){
if(transform.position.z >= -560f){
chk_inside_checkpoint = false;
}
if(bead_ctr > 1){
game_back_door.GetComponent
<DoorScript>().do_open_door();
}
front_door_checkpoint.
SetActive(true);
transform.eulerAngles = new
Vector3(transform.eulerAngles.x,
0, transform.eulerAngles.z);
if(bead_ctr != 0){
if(bead_ctr != 1){
z_front = -150f;
}
if(transform.position.z >= -455f){
obstacles.SetActive(true);
if(chk_2){
obstacles.transform.GetChild(0).
gameObject.GetComponent
<SpawnObstacle>().
re_enable_coroutine();
obstacles.transform.GetChild(1).
gameObject.GetComponent
<SpawnObstacle>().
re_enable_coroutine();
obstacles.transform.GetChild(2).
gameObject.GetComponent
<SpawnObstacle>().
re_enable_coroutine();
chk_2 = false;
}
game_front_door.GetComponent
<DoorScript>().do_open_door();
game_back_door.GetComponent
<DoorScript>().do_close_door();
z_back = -455f;
x_left = -115f;
x_right = 35f;
chk = true;
}
}
}

bead_ctr_holder = bead_ctr;

transform.localPosition +=
vrCamera.transform.forward
* 0.3f * multiplier;
transform.localPosition = new
Vector3(transform.position.x,
y_fix, transform.position.z);

if(transform.position.x
<= x_left){
transform.position = new
Vector3(x_left, y_fix,
transform.position.z);
}

if(transform.position.x
>= x_right){
transform.position = new
Vector3(x_right, y_fix,
transform.position.z);
}

if(transform.position.z
<= z_back){
transform.position = new
Vector3(transform.position.x,
y_fix, z_back);
}
if(transform.position.z
>= z_front){
if(transform.position.x >
right_chk_limit || transform.
position.x < left_chk_limit){
transform.position = new
Vector3(transform.position.x,
y_fix, z_front);
} else {
if(z_front <= transform.
position.z){
obstacles.SetActive(false);
}
if(z_front - transform.position.
z <= -30f){
if(chk){
chk_inside_checkpoint
= true;
z_back = z_front + 30f;
if(bead_ctr == 0){
initial_door.GetComponent
<DoorScript>().do_close_
door();
z_front = -150f;
} else {
game_front_door.
GetComponent<DoorScript>
().do_close_door();
z_front = -27f;
chk_1 = false;
if(transform.position.z
>= -72f){
chk_inside_checkpoint
= false;
}
}
}
}
}
}

public void change_bead_
ctr(int ctr){
bead_ctr = ctr;
}

public int get_bead_ctr(){
return bead_ctr;
}

public void audio_coin(){
chk_coin = true;
}

public void audio_obstacle(){
chk_obstacle = true;
}

public bool chk_inside(){
return chk_inside_checkpoint;
}
}
```

PlayScript.cs

```
namespace GoogleVR.VideoDemo {
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(Collider))]
public class PlayScript
 : MonoBehaviour, TimedInputHandler {

public void HandleTimedInput(){
Choose_Option(transform.
parent.transform.GetSiblingIndex());
}

public void Choose_Option
(int index){

transform.parent.parent.
GetComponent<PlayComm>().
Choose_Option(index);
}
}
}
```

PropellerScript.cs

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

public class PropellerScript
 : MonoBehaviour {

void Start () { }

void Update () {
transform.Rotate(Vector3.up,
200 * Time.deltaTime);
}
}
```

RadialProgressBar.cs

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

public class RadialProgressBar
 : MonoBehaviour {
[SerializeField] private float
currentAmount;
[SerializeField] private float
 speed;

void Update () {
if(currentAmount < 100){
currentAmount += speed *
 Time.deltaTime;
}
GetComponent<Image>().
fillAmount = currentAmount/100;
}
}
```

RegenerateCoins.cs

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

public class RegenerateCoins
 : MonoBehaviour {

public GameObject player;
private bool chk;

void Start () {
chk = false;
}

void Update () {
if(chk){
for(int i=0; i < transform.
childCount; i++){
transform.GetChild(i).
gameObject.SetActive(true);
}

float rand_x = Random.
Range(-100.00f,20.00f);
if(transform.position.z+
55f < -170f){
transform.position = new
Vector3(rand_x, transform.position.
y, transform.position.z+55f);
} else {
transform.position = new
Vector3(-40f, -65f, -455f);
gameObject.SetActive(false);
}
chk = false;
}
}

public void do_regenerate(){
chk = true;
}
}
```

ReminderScript.cs

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

public class ReminderScript
 : MonoBehaviour {

private bool chk_gradual;
private bool chk_change;
private bool chk;

void Start () {
chk = false;
chk_gradual = true;
chk_change = false;
if(transform.parent.gameObject.
name == "End Game"){
start_reminder();
}
}

void Update () {
if(GetComponent<CanvasGroup>
().alpha <= 0f){
if(chk_change){
GetComponent<CanvasGroup>().
alpha = 0f;
chk_gradual = true;
chk_change = false;
chk = false;
if(transform.parent.gameObject.
name == "End Game"){
start_reminder();
}
}
} else if (GetComponent
<CanvasGroup>().alpha >= 1f){
GetComponent<CanvasGroup>
().alpha = 1f;
chk_gradual = false;
chk_change = true;
}

if(chk){
if(chk_gradual){
GetComponent<CanvasGroup>().
alpha += (Time.deltaTime * 0.5f);
} else {
GetComponent<CanvasGroup>().
alpha -= (Time.deltaTime * 0.5f);
}
}
}

public void start_reminder(){
chk = true;
}
}
```

RotateObstacle.cs

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

public class RotateObstacle
 : MonoBehaviour {

void Start () { }

void Update () {
if(transform.parent.gameObject.
name.IndexOf("Clone") >= 0){
transform.Rotate(Vector3.left,
200 * Time.deltaTime);
}
}
}
```

SoundComm.cs

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

public class SoundComm : MonoBehaviour {

public GameObject difficulty_menu;
public int game_index;
public int param_index;
public int difficulty_index;
public bool sound_indicator;

void Start () {
sound_indicator = false;
}

void Update () {        }

public void Choose_Option(int index){
if(index == 1){
sound_indicator = true;
}
game_index = difficulty_menu.
GetComponent<DifficultyComm>().
game_index;
param_index = difficulty_menu.
GetComponent<DifficultyComm>().
param_index;
difficulty_index = difficulty_menu.
```

```
GetComponent<DifficultyComm>().
difficulty_index;
Application.LoadLevel("loading");
}

public void Awake(){
DontDestroyOnLoad(this);
}
}
```

SoundScript.cs

```
 namespace GoogleVR.VideoDemo {
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(Collider))]
public class SoundScript
 : MonoBehaviour, TimedInputHandler {

public void HandleTimedInput(){
Choose_Option(transform.parent.
transform.GetSiblingIndex());
}

public void Choose_Option(int index){
transform.parent.parent.GetComponent
<SoundComm>().Choose_Option(index);
}
}
}
```

SpawnFlash.cs

```
  using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using TMPro;
using GoogleVR.VideoDemo;

public class SpawnFlash : MonoBehaviour {

public GameObject flash;
public GameObject reminder;
public GameObject score;
public GameObject main_camera;
public GameObject tondell;
public AudioClip light_correct;
public AudioClip light_wrong;
private int ctr;
private int prev_child_count;
private int new_child_count;
private int num_arrows;
private int deductions;
private float startTime = 0.0f;
private float holdTime = 3.0f;
private bool chk;
private bool chk_start;
private bool chk_allow_press;
private string hold_curr_color;
private string curr_color;

void Start () {
ctr = 0;
prev_child_count = 0;
new_child_count = 0;
chk = false;
deductions = 0;
chk_start = false;
curr_color = "";
hold_curr_color = curr_color;
chk_allow_press = true;
}

void Update () {
if(main_camera.GetComponent
<TondellPlayerDirection>().
chk_start_game()){
if(!chk_start){
StartCoroutine("Spawn");
chk_start = true;
}
if(Input.anyKey){
if(chk_allow_press){
if(startTime == 0.0f){
curr_color = tondell.
GetComponent<ArrowLights>
().get_curr_color();
hold_curr_color = curr_color;
}
startTime += Time.deltaTime;
curr_color = tondell.
GetComponent<ArrowLights>().
get_curr_color();
if(curr_color != hold_curr_color){
chk_allow_press = false;
}
if(startTime >= holdTime){
prev_child_count = transform.
childCount-new_child_count;
StartCoroutine("Deduction");
chk = false;
startTime = 0.0f;
}
}
} else {
for(int i = 0; i < transform.
childCount; i++){
transform.GetChild(i).transform.
GetComponent<EliminateFlash>
().enabled = true;
}
startTime = 0.0f;
chk_allow_press = true;
}
}
}

public void update_children(){
chk = true;
}

public void update_num_of_
arrows(int param){
num_arrows = param;
}

IEnumerator Deduction(){
yield return new
WaitForSeconds(0.01f);
if(prev_child_count ==
transform.childCount -
new_child_count){
int new_score = Int32.
Parse(score.GetComponent
<TextMeshProUGUI>().text+"");
new_score -= 1;
deductions += 1;
GetComponent<AudioSource>
().clip = light_wrong;
if(new_score < 0){
score.GetComponent<TextMesh
ProUGUI>().text = "0";
} else {
score.GetComponent<TextMesh
ProUGUI>().text = new_score+"";
}
} else {
GetComponent<AudioSource>().
clip = light_correct;
}
GetComponent<AudioSource>().
Play();
new_child_count = 0;
}

IEnumerator Spawn(){
int rand_num, rand_spawn,
rand_color;
bool chk_exist = false;
while(true){
if(num_arrows > 0){
rand_color = UnityEngine.
Random.Range(0, num_arrows);
chk_exist = false;
for(int i = 0; i < transform.
childCount; i++){
if(rand_color == 0 && transform.
GetChild(i).gameObject.name
== "Red"){
chk_exist = true;
break;
} else if(rand_color == 1 &&
transform.GetChild(i).gameObject.
name == "Green"){
chk_exist = true;
break;
} else if(rand_color == 2 &&
 transform.GetChild(i).gameObject.
 name == "Yellow"){
chk_exist = true;
break;
} else if(rand_color == 3 &&
transform.GetChild(i).gameObject.
name == "Blue"){
chk_exist = true;
break;
} else if(rand_color == 4 &&
transform.GetChild(i).gameObject.
name == "Violet"){
chk_exist = true;
break;
} else if(rand_color == 5 &&
transform.GetChild(i).gameObject.
name == "Orange"){
```

```
chk_exist = true;
break;
}
}

if(!chk_exist){
GameObject new_flash =
Instantiate<GameObject>(flash);
new_flash.transform.parent
 = transform;
float x_pos = 0f;
switch(rand_color){
case 0:
new_flash.name = "Red";
new_flash.transform.
GetComponent<Light>().color
= new Color32(255,43,43,255);

if(num_arrows == 4){
x_pos = -22f;
} else if(num_arrows == 5){
x_pos = -23f;
} else {
x_pos = -24.5f;
}
break;
case 1:
new_flash.name = "Green";
new_flash.transform.
GetComponent<Light>().color =
new Color32(31,143,22,255);

if(num_arrows == 4){
x_pos = -7f;
} else if(num_arrows == 5){
x_pos = -11f;
} else {
x_pos = -14.5f;
}
break;
case 2:
new_flash.name = "Yellow";
new_flash.transform.
GetComponent<Light>().color =
 new Color32(234,199,3,255);

if(num_arrows == 4){
x_pos = 7.5f;
} else if(num_arrows == 5){
x_pos = 1f;
} else {
x_pos = -4.5f;
}
break;
case 3:
new_flash.name = "Blue";
new_flash.transform.
GetComponent<Light>().color =
new Color32(0,177,233,255);

if(num_arrows == 4){
x_pos = 22.5f;
} else if(num_arrows == 5){
x_pos = 11f;
} else {
x_pos = 4.5f;
}
break;
case 4:
new_flash.name = "Violet";
new_flash.transform.
GetComponent<Light>().color =
new Color32(125,0,255,255);

if(num_arrows == 5){
x_pos = 24f;
} else {
x_pos = 14.5f;
}
break;
case 5:
new_flash.name = "Orange";
new_flash.transform.
GetComponent<Light>().color =
new Color32(254,58,0,255);
x_pos = 24.5f;
break;
default:
break;
}
new_flash.transform.position =
 new Vector3(x_pos, 17, 30);
if(Input.anyKey){
new_child_count = 1;
new_flash.transform.
GetComponent<EliminateFlash>
().enabled = false;
}
}
}
ctr++;
if(ctr == 3){
reminder.GetComponent<Reminder
Script>().start_reminder();
ctr = 0;
}
if(num_arrows > 0){
if(chk_exist){
yield return new
WaitForSeconds(0);
} else {
rand_num = UnityEngine.
Random.Range(7, 12);
yield return new
WaitForSeconds(rand_num);
}
} else {
yield return new
WaitForSeconds(0);
}
}
}

public int get_deductions(){
return deductions;
}
}
```

SpawnObstacle.cs

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

public class SpawnObstacle
 : MonoBehaviour {

public GameObject missile;
public GameObject barrel;
public GameObject log;
public GameObject book;
public GameObject ball;
public GameObject brock;
private int difficulty_index;
private float min_speed;
private float max_speed;
private bool chk;

void Start () {
difficulty_index = brock.
GetComponent<BrockParam>
().get_difficulty();
switch(difficulty_index){
case 0:
min_speed = 5.00f;
max_speed = 7.00f;
break;
case 1:
min_speed = 4.00f;
max_speed = 6.00f;
break;
case 2:
min_speed = 3.00f;
max_speed = 4.00f;
break;
default:
min_speed = 3.00f;
max_speed = 4.00f;
break;
}
chk = true;
}

void Update () {
if(chk){
StartCoroutine("Spawn");
chk = false;
}
}

public void re_enable
_coroutine(){
chk = true;
}

IEnumerator Spawn(){
float rand_num;
int rand_spawn;
while(true){
rand_num = Random.
Range(min_speed, max_speed);
yield return new
WaitForSeconds(rand_num);

rand_spawn = Random.
Range(0, 4);
switch(rand_spawn){
case 0:
```

```
GameObject barrel_copy =
Instantiate<GameObject>(barrel);
barrel_copy.transform.parent =
gameObject.transform;
break;
case 1:
GameObject log_copy =
Instantiate<GameObject>(log);
log_copy.transform.parent =
gameObject.transform;
break;
case 2:
GameObject book_copy =
Instantiate<GameObject>(book);
book_copy.transform.parent =
gameObject.transform;
break;
case 3:
GameObject ball_copy =
Instantiate<GameObject>(ball);
ball_copy.transform.parent =
gameObject.transform;
break;
default:
break;
}
}
}
}
```

SpeedShow.cs

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

public class SpeedShow
 : MonoBehaviour {

private bool chk_gradual;
private bool chk_change;
private bool chk;

void Start () {
chk = false;
chk_gradual = true;
chk_change = false;
}

void Update () {
if(GetComponent<Canvas
Group>().alpha <= 0f){
GetComponent<Canvas
Group>().alpha = 0f;
}

if(chk){
GetComponent<Canvas
Group>().alpha = 1;
chk = false;
} else {
GetComponent<Canvas
Group>().alpha -= (Time.
deltaTime * 0.8f);
}
}

public void show_speed(float
multiplier){
chk = true;
if(multiplier == 3.0f){
GetComponent<TextMeshPro
UGUI>().text = "Maximum
limit reached!";
} else if(multiplier == 0.0f){
GetComponent<TextMeshPro
UGUI>().text = "Minimum limit
reached!";
} else {
GetComponent<TextMeshPro
UGUI>().text = "Forward speed x "
 + multiplier;
}
}
}
```

SummonPlayerScript.cs

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

public class SummonPlayerScript
 : MonoBehaviour {

public GameObject tile_1;
public GameObject tile_2;
public GameObject speed_controls;
private float player_y;
private float tile_y;

void Start () {
player_y = 0.0f;
tile_y = 0.0f;
}

void Update () {
player_y = transform.
localPosition.y + Time.
deltaTime * 5f;
tile_y = tile_1.transform.
localPosition.y + Time.delta
Time * 0.5f;

if(player_y < -65f){
transform.localPosition = new
Vector3(transform.localPosition.x,
player_y, transform.localPosition.z);
tile_1.transform.localPosition =
new Vector3(tile_1.transform.
localPosition.x, tile_y, tile_1.
transform.localPosition.z);
tile_2.transform.localPosition =
new Vector3(tile_2.transform.
localPosition.x, tile_y, tile_2.
transform.localPosition.z);
} else {
tile_1.transform.localPosition =
new Vector3(tile_1.transform.
localPosition.x, 3f, tile_1.
transform.localPosition.z);
tile_2.transform.localPosition =
new Vector3(tile_2.transform.
localPosition.x, 3f, tile_2.
transform.localPosition.z);
GetComponent<PlayerMovement>
().enabled = true;
speed_controls.SetActive(true);
GetComponent<SummonPlayer
Script>().enabled = false;
}
}
}
```

TondellPlayerDirection.cs

```
  using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using GoogleVR.VideoDemo;

public class TondellPlayerDirection
 : MonoBehaviour {

public GameObject right_direction;
public GameObject left_direction;
public GameObject front_wall;
public GameObject instructions;
public GameObject objective_good;
public GameObject reminder;
public AudioClip objective_1;
public AudioClip objective_2;
private float dirNum;
private float pressed_key;
private bool chk_found;
private bool chk_start;
private Vector3 screenPoint;

void Start () {
chk_found = false;
chk_start = false;
dirNum = 0.0f;
pressed_key = 0.0f;
objective_good.GetComponent
<TextMeshProUGUI>().text =
"Controls will only apply once
the voice is done speaking.";
objective_good.GetComponent
<CanvasGroup>().alpha = 1;
objective_good.transform.parent.
gameObject.GetComponent
<AudioSource>().clip = objective_1;
objective_good.transform.parent.
gameObject.GetComponent
<AudioSource>().Play();
}

void Update () {
if(chk_found && !chk_start &&
(objective_good == null ||
(objective_good != null && !objective_
good.transform.parent.gameObject.
GetComponent<AudioSource>().
isPlaying))){
objective_good.GetComponent
<TextMeshProUGUI>().text = "Press any
button if you understand the objective
```

```
to start the game";
if(pressed_key+1.0f <= Time.time &&
Input.anyKey){
objective_good.GetComponent<Canvas
Group>().alpha = 0;
reminder.GetComponent<CanvasGroup>
().alpha = 0;
chk_start = true;
}
}

screenPoint = GetComponent<Camera>
().WorldToViewportPoint(front_wall.
transform.position);
bool onScreen = screenPoint.z > 0
&& screenPoint.x > 0 && screenPoint.x
< 1 && screenPoint.y > 0 &&
screenPoint.y < 1;

if(!onScreen){
right_direction.SetActive(true);
left_direction.SetActive(true);

dirNum = AngleDir(transform.forward,
front_wall.transform.position - transform.
position, transform.up);

if(dirNum == 1){
left_direction.transform.localEulerAngles
 = new Vector3(0f, 0f, left_direction.
 transform.localEulerAngles.z);
right_direction.transform.localEulerAngles
 = new Vector3(0f, 0f, right_direction.
 transform.localEulerAngles.z);
} else {
left_direction.transform.localEulerAngles
 = new Vector3(0f, 180f, left_direction.
 transform.localEulerAngles.z);
right_direction.transform.localEulerAngles
 = new Vector3(0f, 180f, right_direction.
 transform.localEulerAngles.z);
}

if(transform.localRotation.x > 0.3f){
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 90f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 90f);
} else if(transform.localRotation.x
 < -0.3f){
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, -90f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, -90f);
} else {
left_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 0f);
right_direction.transform.localEulerAngles
 = new Vector3(0f, left_direction.
 transform.localEulerAngles.y, 0f);
}
} else {
right_direction.SetActive(false);
left_direction.SetActive(false);
if(!chk_found && (objective_good == null
|| (objective_good != null && !objective_
good.transform.parent.gameObject.
GetComponent<AudioSource>().isPlaying))){
if(objective_good != null){
objective_good.GetComponent<Canvas
Group>().alpha = 0;
}
if(Input.anyKey){
if(objective_good != null){
objective_good.GetComponent<Canvas
Group>().alpha = 1;
}
instructions.GetComponent<TextMesh
ProUGUI>().text = "Match the lighted
pair of arrows to the flashing lights by
pressing any button for 3 seconds";
if(objective_good != null){
objective_good.transform.parent.
gameObject.GetComponent<AudioSource>
().clip = objective_2;
objective_good.transform.parent.
gameObject.GetComponent<Audio
Source>().Play();
reminder.GetComponent<Canvas
Group>().alpha = 1;
} else {
chk_start = true;
}
chk_found = true;
pressed_key = Time.time;
}
}
}
}

public bool chk_start_game(){
return chk_start;
}

public float AngleDir(Vector3 fwd,
Vector3 targetDir, Vector3 up) {
Vector3 perp = Vector3.Cross(fwd,
targetDir);
float dir = Vector3.Dot(perp, up);

if (dir > 0f) {
return 1f;
} else if (dir < 0f) {
return -1f;
} else {
return 0f;
}
}
}
```

ToucanScript.cs

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

public class ToucanScript
 : MonoBehaviour {

private bool chk;

void Start () {
chk = false;
}

void Update () {
transform.localPosition +=
transform.forward * 0.05f;
transform.localPosition =
new Vector3(transform.position.x,
70, transform.position.z);

if(transform.position.x >= 200.0f
|| transform.position.x <= 0.0f ||
transform.position.z >= 200.0f ||
transform.position.z <= 0.0f){
chk = true;
}

if(chk){
chk = false;
transform.eulerAngles = new
Vector3(transform.eulerAngles.x,
transform.eulerAngles.y+180f,
transform.eulerAngles.z);
}
}
}
```

TrackRightDirection.cs

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

public class TrackRightDirection
 : MonoBehaviour {

public GameObject wall;

void Start () { }

void Update () {
transform.LookAt(wall.transform);
}
}
```

TutorialsScript.cs

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

public class TutorialsScript
 : MonoBehaviour, TimedInputHandler {

public GameObject main_menu;
public GameObject directions;

public void HandleTimedInput(){
main_menu.GetComponent<Canvas>
().enabled = true;
GameObject tutotrials = transform.
parent.parent.parent.parent.gameObject;
```

```
directions.SetActive(true);

for(int i = 0; i < tutotrials.
transform.childCount; i++){
tutotrials.transform.GetChild(i).
gameObject.SetActive(false);
}
}
}
```

WalkingScript.cs

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

public class WalkingScript
 : MonoBehaviour {

public GameObject speed;
public Transform vrCamera;
public Transform plane;
private float dist;
private float multiplier;
private bool chk;

void Start () {
dist = 0.0f;
multiplier = 1.0f;
chk = true;
}

void Update () {
if(chk){
if(Input.GetKeyDown(KeyCode.
Joystick1Button0) || Input.
GetKeyDown(KeyCode.A)){
if(multiplier <= 2.0f){
multiplier += 0.1f;
speed.GetComponent<SpeedShow>
().show_speed(multiplier);
} else {
speed.GetComponent<SpeedShow>
().show_speed(3.0f);
}
} else if(Input.GetKeyDown(KeyCode.
oystick1Button3) || Input.
GetKeyDown(KeyCode.D)){
if(multiplier >= 1.1f){
multiplier -= 0.1f;
speed.GetComponent<SpeedShow>
().show_speed(multiplier);
} else {
speed.GetComponent<SpeedShow>
().show_speed(0.0f);
}
}

transform.localPosition += vrCamera.
transform.forward * 0.03f * multiplier;

if(transform.position.x >= 175f){
transform.localPosition = new
Vector3(175f, transform.position.y,
transform.position.z);
} else if(transform.position.x
<= 25f){
transform.localPosition = new
Vector3(25f, transform.position.y,
transform.position.z);
}

if(transform.position.z >= 175f){
transform.localPosition = new
Vector3(transform.position.x,
transform.position.y, 175f);
} else if(transform.position.z
<= 25f){
transform.localPosition = new
Vector3(transform.position.x,
transform.position.y, 25f);
}


dist = Vector3.Distance(plane.
position, transform.position);
if(dist <= 10.0f){
chk = false;
}
}
}
}
```

XI. Acknowledgement

Bago magsimula ang SP proposal, desidido na ako na gawing parte ng thesis ko ang paggamit ng Machine Learning. Ang sabi kasi nila, pag 'yon ang topic mo, mataas ang chance na maka-graduate ka on time. Kaya ayon, nag-download na ako ng maraming videos ni Andrew Ng mula sa Coursera para mas maintindihan ko agad kung paano gumagana 'yon, nagsagot ng mga quiz tungkol sa Machine Learning, at naghanap ng mga journals para makahanap ng topic para sa thesis. Pero bakit ganon, tila nagpapanggap lang ako na naiintindihan ang mga ito kasi wala talaga akong maintindihan. Nag-crash course naman kami tungkol dito noon kay Sir Marvin, pero bakit parang bago lahat ng concepts? Kaya sinukuan ko agad ang Machine Learning. Sayang nga 'yung mga videos na na-download ko eh. Nagtanong nalang ako sa ibang meron nang topic, nagbabaka-sakali na makaisip ng topic mula sa kanila. At sa wakas, may nahanap ako. Salamat kay Angelene Ronquillo dahil sa Glaucoma VR topic niya, na-inspire ako na maghanap din ng iba pang eye disorders na pwedeng i-translate ang treatment procedures nito sa VR.

At 'di nga nagtagal, dumating sa buhay ko ang topic na Strabismus. Sabi nila, madali ang buhay kapag Virtual Reality ang maging thesis mo, kasi kaya raw i-cram within 1 month ang buong thesis. Kaya mula Machine Learning, niyakap ko nang buong buo ang Virtual Reality. Desidido na ko. Vritual. Reality. Ang. Magiging. Topic. Ko. Kaya nagbasa na ako ng mga journals tungkol sa Strabismus at VR. Buti nalang, wala pang VR application para sa strabismus kaya pwede talagang gawing topic. Nakapagpasa ako ng Chapter 1 draft ko on time, at napunta ako sa magandang adviser ko na si Ma'am Perl na hindi ako pinabayaan hanggang dulo. Tinanggap niya ako bilang advisee kahit hindi niya forte ang VR. Kaya, ma'am, maraming salamat. Nagabayan mo ako sa maraming bagay, lalo na sa structure at strabismus part ng thesis ko. Nagpa-consult din ako sa Pro sa VR na si Sir Marvin para sa VR side ng thesis ko. Sir, maraming salamat dahil kahit hindi mo ako advisee, pumayag kang magpaistorbo. Dahil

sayo, mas naging solid 'yung VR part ng thesis ko. Sinama mo rin ako sa Team VR mo para mabantayan kami, makapag-collaborate, at gumawa nang magkakasama dahil ganon ang ginawa ng unang Team VR mo. Hindi ako nagsisi na sumama madalas sa Team VR mates ko na sina Red Quito at Angelene kahit na madalas ay daldalan lang ang nagagawa natin, kasi, sa mga panahong sama-sama tayong gumagawa, nakakapag-focus pa rin tayo sa SP natin. Kahit mahal ang mga tinda sa Coreon Vito at Tim Horton's, okay lang, kasi nagkakaroon naman tayo ng progress sa SP. Kaya, salamat peeps.

Lumipas ang panahon, at dumating na ang oras para sa SP proposal. Kabado ang lahat. Hindi pa ready ang iba kahit tapos na, kasi, sa mismong SP proposal, walang pwedeng maging handa. Hindi alam ang mga itatanong ng panel, kaya hindi mo talaga masasabi na ready ka. Isa lang ang sigurado, basta may go signal ka, pasado ka. Kaya maraming salamat uli, Ma'am Perl, dahil nabigyan mo ako ng go signal noong 2 days before ng proposal ko.

Noong turn ko na para mag-propose, kulang ako sa tulog, lutang, at sabaw. Pero nalampasan ko ang proposal kahit sobrang bigat ng semestreng iyon para sa akin. Nakaya ko pa rin kahit sobrang wasak ko na. Hindi ko iyon magagawa kung wala ang tulong ng Google, IEEE, sci-hub, CAS Student, at Cisco internet sa RH114/108. At para makagawa ng thesis nang mas komportable, sinamahan ako ng musika ng TwentyOnePilots, Kodaline, Paramore, Parachute, Coldplay, MCR, Green Day, Imagine Dragons, Callalily, Silent Sanctuary, Eraserheads, Parokya ni Edgar, Ben&Ben, IV of Spades, December Avenue, Autotelic, at nina Ed Sheeran, James Arthur, James Bay, ni idol Tyler Joseph, at ng kamukha kong si Zayn Malik. Maraming maraming salamat sa inyo.

Dumating ang huling semestre, at panahon na para sa pag-develop ng thesis. Nanatili kong kasama ang mga nabanggit kanina. Sabi ko sa sarili ko, bago mag March, dapat tapos ko na ang thesis ko. Kaya January palang, nagsipag na ko gumawa. Nagpuyat agad ng maraming gabi para hindi na maging abala ang thesis kapag malapit nang matapos ang semestre. Dumating ang March, hindi pa rin

ako tapos. May pagkakataon pang nabura ko ang project ko. Walang back-up. Walang kahit ano. Nanlumo ako sa nangyari. 'Yung pinaghirapan ko ng ilang buwan, nawala lang agad sa loob ng ilang minuto. Na-badtrip ako. Mangiyak-ngiyak dahil ang sakit, ang sakit sakit. Pero 'di ako nagsayang ng maraming oras, sinimulan ko uli sa umpisa. Pinilit kong tandaan kung paano ko ginawa 'yung mga nagawa ko dati. At sa loob ng dalawang araw, nagawa ko uli ang ilang buwan na pinaghirapan ko. Kaya simula noon, sa tuwing matatapos akong gumawa ng thesis para sa araw na 'yon, gumagawa agad ako ng back-up. Sa ngayon, meron akong 53 back-ups in total. Naging magaan din ang semestreng ito dahil hindi naging pahirap ang ibang subjects ko. Kaya para sa mga professor ko na hindi nagpahirap, maraming salamat sa inyo. Dahil sa inyo, nagkaroon ako ng mas maraming oras para sa thesis ko.

Lumipas ang panahon, at dumating na ang oras para sa SP defense. Hindi ko inakala na aabutin ang paggawa ko ng SP ng isang araw bago ang defense ko. Ang before March na sinabi ko ay naging before defense day ko. Hindi ko inakala na halos apat na buwan akong gumawa ng thesis ko. At mas hindi ko inaakala na nagawa kong lampasan ang SP defense. Salamat uli kay Angelene dahil pumayag siyang ibili ako ng Jollibee S5 (burger steak + spaghetti) para sa ibibigay ko sa panel. Salamat din sa mga panel ko dahil hindi niyo ako masyadong ginisa.

Tapos na ang thesis defense ko. Final requirement nalang, ilang exams, at matatapos na rin ang huling semestre ko sa UPM. Sobrang malalaking pagsubok ang hinarap ko, at ilang araw nalang, malalampasan ko na rin mga ito. Gusto kong magpasalamat sa mga mabubuti at malolokong kaibigan ko. Sa Mumshiez, mga marz na sina Kenneth Tabornal at Grace Mora, pati na rin si Michael Ramos, maraming salamat dahil sa kabila ng hirap ng semestreng to, nandiyan kayo para magloko. Salamat sa mga kalokohan niyo dahil mas naging madali ang semestre ko. Salamat sa pagkakaibigan na ibinigay niyo sa akin.

At para sa huling bahagi ng Acknowledgement ko, gusto kong pasalamatan ang pamilya ko na laging nandiyan para sa akin. Sa ate ko na si Princess Jayne

na kahit laging busy dahil nagtatrabaho na, maraming salamat - lalo na sa pag-goodluck sa akin bago ako sumalang sa defense dahil nabuo 'yung loob ko na makaya ang defense ko, kaya aylabyu. Sa mga pamangkin ko na sina Aedan at Gabby, labyu mga boys, kayo 'yung nagpapagaan at nagpapalala ng bawat araw na gumagawa ako ng thesis kasi ang kukulit niyo. Sa kuya ko na si Gerome, salamat, kuya dahil kung hindi dahil sa'yo, wala ako sa UP ngayon. Ikaw 'yung nagpilit sa akin na piliin ang UP kaysa UST, na alma mater mo, kahit na desidido na akong piliin ang UST noon, kasi sabi mo, UP ang para sa akin. Dahil don, labyu kuya. At sa napakaganda kong mama ko na si Gemma, maraming salamat dahil inasikaso mo ang mga requirements ko para makapasok ng UP kasi hindi ko pa kaya gawin 'yung mga 'yon nang mag-isa. Salamat sa mga luto mong masasarap, dahil nakakalimutan kong nagti-thesis pala ako. Salamat sa bawat paalala mo na wag ako magpuyat dahil masama sa kalusugan - hindi ko nagagawa dahil napupuyat pa rin ako para sa thesis ko. Sa bawat pagpunta niyo dito sa Astral ni Aedan, kahit na nakakapagod magbyahe, sinasamahan niyo pa rin ako dito para hindi ako mag-isa. Salamat sa bawat pag-asikaso sa akin, sa pag-prepare ng mga damit ko bago bumalik ng Maynila, sa pagplantsa ng polo ko para sa speech performances ko at defense, at sa pagpapaalala sa akin na wag magpalipas ng gutom - kahit minsan nalilipasan ako ng gutom dahil sa thesis. Labyu ma. Sobrang malalaking bagay para sa akin ang mga maliliit na bagay na ginagawa niyo para sa akin. Mahal na mahal ko kayong lahat. Para sa inyo ang tagumpay kong ito.