The Hong Kong Polytechnic University

Department of Computing

Capstone Project(2016/2017)

Final Report

Using Computers to aid Vision Screening in Innovative Manners

---

Programme Title:	BSc (HONS) Information Technology
Programme Code:	61431-FIT
Student Name:	Lui Ho Lam
Student ID:	13068831D
Supervisor Name:	Dr. NGAI Grace
Co-examiner Name:	Dr. CHAN Keith
2nd Assessor Name:	Dr. CHAN Henry / Dr. CHUNG Korris

1. Abstract

Traditional vision screening tests are inefficient in terms of time and cost. In order to provide an easily accessible and easy to perform vision screening tests, this project have studied the related literature of using computer in vision screening and how virtual reality technology can be used in vision screening tests.

The purpose of this project is to create a virtual reality game with vision screening feature, so as to raise children's interest in the vision screening test and to make vision screening tests more effective and efficient.

A virtual reality game Valley Runner which runs on a smartphone is developed with Unity Game Engine and Google Cardboard SDK. The game includes binocular vision test and colorblind test.

This report documented the existing measures, the scope of the project, the project design and implementation, and finally an evaluation.

2. Acknowledgement

First, I would like to thank my supervisor professor Grace Ngai for the support and useful feedback on my project.

I would also like to thank Dr. Korris Chung and Dr. Henry Chan for giving useful suggestions on my project planning and report writing.

Table of Contents

Table of Figures

3. Introduction

3.1 Background

Virtual reality(VR) technology refers to the technology that uses computer generated images, audio to recreate a real environment or create an imaginary space [1]. Because of the hardware improvement and cost reduction, it has become more and more popular and easily accessible in the public. The cheapest VR equipment is the Google Cardboard, which also require a smartphone as the display screen.

Vision screening is a relatively short test that can indicate the presence of a vision problem or a potential vision problem [2]. According to a recent research, there is a significant increase number of Hong Kong preschoolers having myopia(nearsightedness) in past ten years [3]. The research also suggests that vision screening programmes may need to be tailored to correspond to the local population and be adjusted accordingly from time to time. Nowadays, the department of Health of Hong Kong provide preschool vision screening to children at 4 years old, the tests include visual acuity and binocular function[4]. Moreover, the Student Health Service Centers also provide yearly health body check which include vision screening to primary and secondary school students [5].

However, these tests take places in official health care centers, which is inconvenient to parents and students as they need to register or book the service. They may take professional vision examination in eye clinics, but the cost is expensive. Computer technologies should be applied to solve the above problems.

Although there is no vision screening VR game at this point, similar ideas have already been implemented and tested using computer games and similar approaches, which are useful to my project as they provide information of the advantages and what have to be improved. The next part introduces some of the vision screening software with their performances and features.

3.2 Existing Measures

3.2.1 EyeSpy 20/20

EyeSpy 20/20™
Vision Screening for the 21st Century
Better Vision for a Better Future

Figure 1 *EyeSpy 20/20*

EyeSpy 20/20 is an automated computer program that assesses vision while a child plays a video game. In addition to visual acuity, it incorporates an analysis of binocular function [6]. The idea of this program is to apply traditional visual acuity test using charts with symbols or English characters into an adventure game. The research concludes that the sensitivity, specificity, and conventional positive likelihood ratio were 88%, 87%, and 6.8 when EyeSpy was used with a patch compared to gold-standard professional eye examination results, the results were not significantly different.

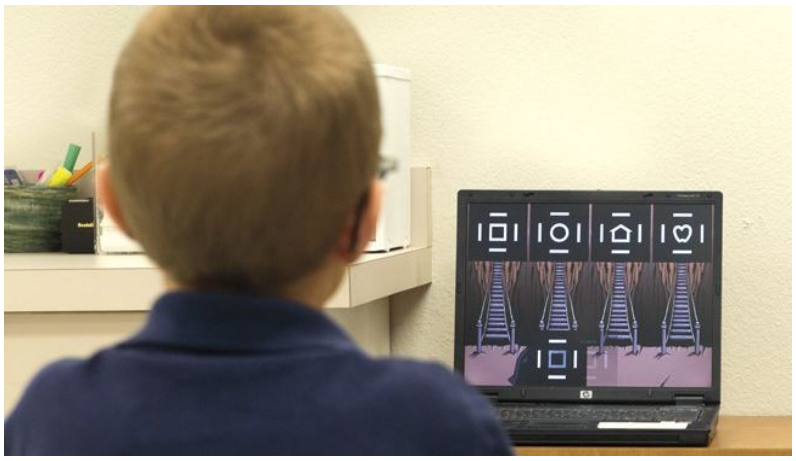


Figure 2 *A boy performing visual acuity test with EyeSpy 20/20*

However, EyeSpy vision screening system runs on a computer and it is performed by a lay screener, which means an examiner is required to give instructions to children. It also need a proper setup for an accurate distance between the computer and the children in order to perform the test which is inconvenient.

3.2.2 DoDo's Catching Adventure

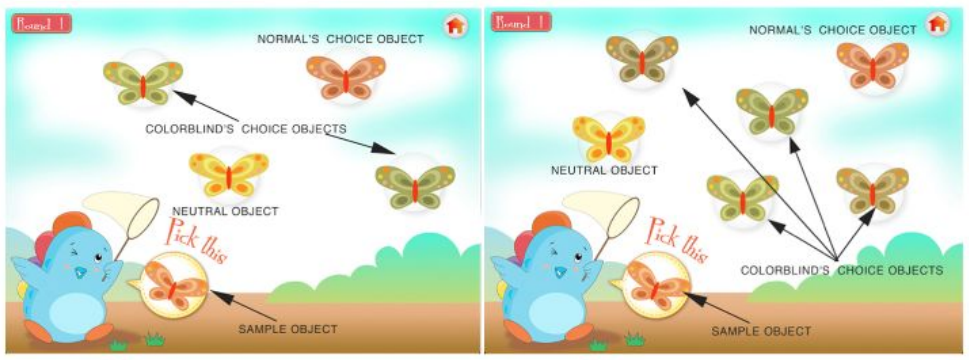


Figure 3 *Game interface of DoDo's Catching Adventure*

DoDo's Catching Adventure is a color vision deficient screening tablet game for young children. A user study conducted at Singapore National Eye Centre showed that DoDo was adequately effective in identifying Red-Green color vision deficiency and comparable to two current gold standard colorblind tests, Ishihara and Farnsworth D15[7][8], as it successfully detected ten significant Red-Green deficient subjects who are detected by both Ishihara and the D15.

The result shows that using computer display screen would not affect the accuracy of color blind test compared to traditional methods using paper.

3.2.3 Automated Computerized vision screening test

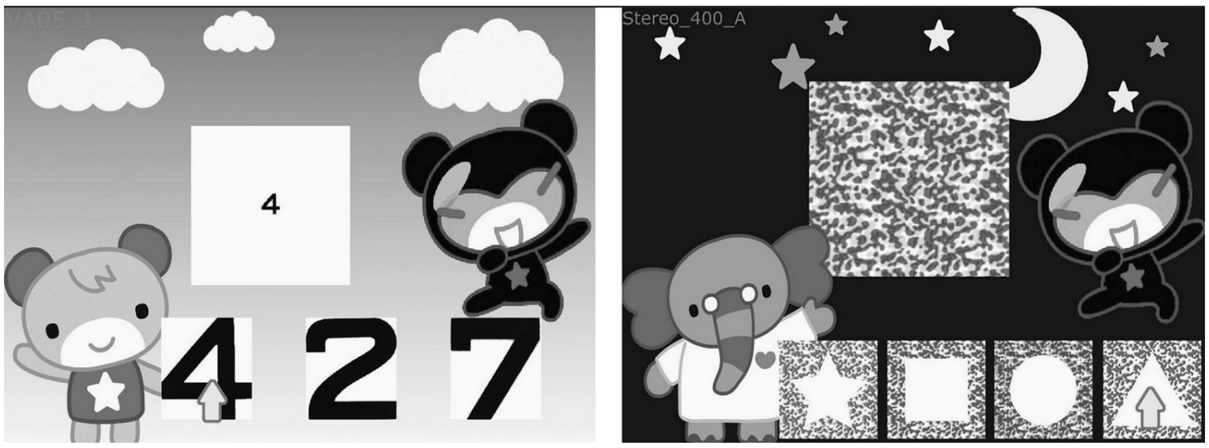


Figure 4 *Automated computerized distance visual acuity test and stereoacuity test*

A research in automated computerized distance visual acuity and stereo acuity test using an interactive video game also claimed that the test has good reliability and acceptable validity compared with the Snellen visual acuity chart and the Distance Randot Stereo test [9]. The game will display the results on a report screen after the tests are completed. A report file will also be generated on the desktop database, containing the subject's identification, elapsed time, and the measurement of visual acuity and stereoacuity. The game developed in the research is similar to the EyeSpy 20/20 software which runs on a computer, and also require an examiner and environment setup.

3.2.3 EyeNetra

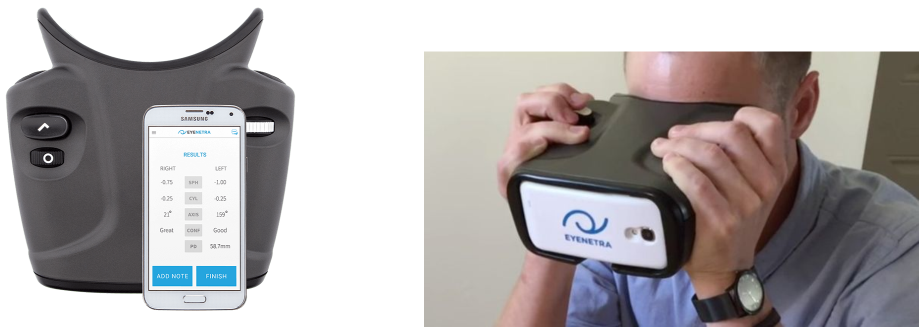


Figure 5 *EyeNetra auto-refractor*

EyeNetra portable autorefractor is an equipment that measures sphere, cylinder, axis and pupillary distance through a series of game-like interactions in a virtual-reality environment [10]. Its accuracy is similar to traditional auto-refractors (~0.35D). The equipment is designed for patients aged from 13 to 65 because the instructions are relatively complicated for young children.

The following table concludes the features of different computer softwares discussed above:

	EyeSpy 20/20	DoDo's catching adventure	Computerized visual acuity game	EyeNetra	My Proposed Virtual Reality game
Visual acuity	✓	✗	✓	✓	✗
Color vision	✗	✓	✗	✗	✓
Binocular vision	✓	✗	✗	✓	✓
Examiner required	✓	✗	✓	✗	✗
Environment setup required	✓	✗	✓	✗	✗
Entertaining features	✓	✓	✓	✗	✓
Cost	USD $1500-$2500 Annual License	Not Public	Not Public	USD $1099	Free
Accuracy (Compared to standard vision screening tests)	Not significantly different	Comparable	Acceptable validity	Very Accurate	Need further tests

Figure 6 *Comparative Table of the various vision screening software*

3.2.4 Vivid Vision for Amblyopia (Diplopia)

Figure 7 *Vivid Vision for Amblyopia*

Vivid Vision uses virtual reality headsets like the Oculus Rift, HTC Vive, Samsung Gear VR, and Google Daydream to treat amblyopia, strabismus, and convergence disorders[11]. It is designed for all ages with interactive and exciting exercises. The gameplay preview of Vivid Vision shows that the game has applied the advantage of virtual reality which it outputs different contents to left and right eye.

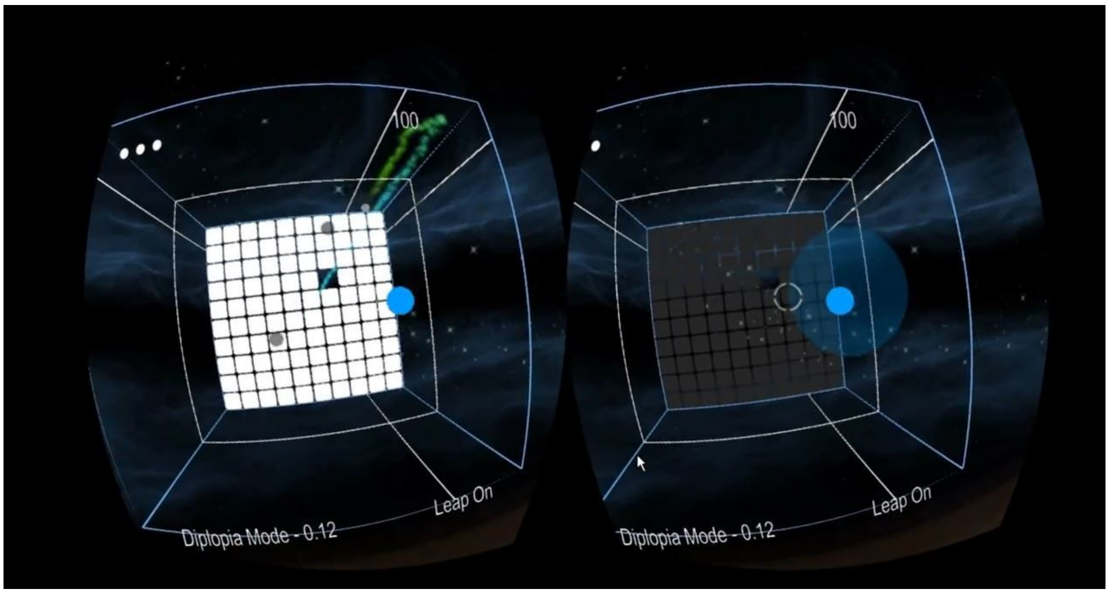


Figure 8 *Game Screen Capture of "Diplopia"*

Vivid Vision was also known as "Diplopia" before the company changed its name in 2015. The game delivers a different image to each eye, forcing the two eyes to work together in order to win the game. The game shows the fast moving ball only to the weak eye of the players so they must exercise the muscle in control of that eye more than normal. Showing brighter and dimmer paddles to the weak and strong eye respectively can also help the players practice integrating two sets of visual information into one coherent picture [12].

Although the software is only provided by eye clinics which limit the accessibility, its idea of using virtual reality to display different image to different eyes and the game design are very useful for my project as references.

3.3 Problem Statement

Although the above computer games or software used for vision screening have high accuracy, their functions are separated and the level of automation is not high enough to reduce the time and human resources in the tests compared to professional eye examination. It is important as the aim of using the technology in vision screening is not only to entertain children, but also to reduce the cost so as to increase the efficiency of traditional vision screening tests.

For published software, their prices are expensive as they are designed for organizations or schools, not individuals. They are not easily accessible by the public.

3.4 Objectives and Outcome

This project aims to create a virtual reality game used as an alternative vision screening tests which does not require an examiner. The game includes color blindness and binocular vision tests in the game. It can be run on a smartphone with a gyroscope sensor using Google cardboard.

The target users are children aged above four years old as researches have conducted that net benefit obtained is the highest when using photoscreening on children aged from 3 to 4 years old [13], and earlier detection may allow for treatment to be more cost effective by reducing the number of medical visits required for resolution [14].

The objective of this project is to increase the efficiency of vision screening by lowering the cost of time, resources and manpower, as well as to remove children's trepidation about getting their vision screened [15] using a virtual reality game.

It is expected that the children are able to play the game by themselves without a vision screening examiner giving instructions, the result should be output after the game is completed so teachers or parents can understand the vision condition of the children.

If the overall result of the product is accurate and acceptable compared to traditional vision screening provided by the government and other organizations, it may hopefully replace traditional eye charts and equipments in order to increase the mobility of the vision screening services provided by these parties.

4. Project Design

4.1 System Requirments

As one of the objectives of this project is to increase the accessibility of vision screening tools with lower cost. The game was designed to be play on the cheapest virtual reality device, which is Google Cardboard with a smartphone.

A research on interaction methods for google cardboard has concluded that controller-less interaction methods such as gaze input and cardboard button are more popular among users[16]. In the early planning stage of the project, in order to reduce the physical size of the equipment and increase the convenience of performing the vision screening test, external bluetooth controller was not take into consideration although it may bring more different interactions throughout the game.

4.1.2 Google Cardboard

Google published the first generation of Google Cardboard (v1.0) in 2014, and the second generation of Google Cardboard (v2.0) in 2015. A virtual reality equipment store BrizTechVR has reviewed and compared two versions of Google Cardboard. To summarize, the second generation of Google Cardboard has a number of significant improvements over the first generation[17].

The first improvement is the simple construction, the v2.0 only takes 3 fold for the setup, while the v1.0 comes as a cardboard net and requires putting the components together. A simple setup process is important as one of the objective of this project is to reduce the time cost.

The second improvement is that the v2.0 support up to 6" screen phones, while the v1.0 only support up to 5.2” screen phones. By increasing the support screen size, more devices can be supported by the newer Google Cardboard and hence it can increase the accessibility.

The third improvement is the physical screen-touch button. The first generation of Google Cardboard uses a magnet based switch to activate a “click” within the virtual reality app, it was a good idea but one large problem is that it requires the smartphone to have a magnetic sensor on the side in order to receive the magnetic signal from the cardboard button. Since not every smartphone has a magnetic sensor, using the v1.0 would be a huge disadvantage in the accessibility. On the other side, the v2.0 uses a physical screen touch button with a foil tip, which means that when the user presses the button on the cardboard, it moves a lever to simulate a touch to the screen.

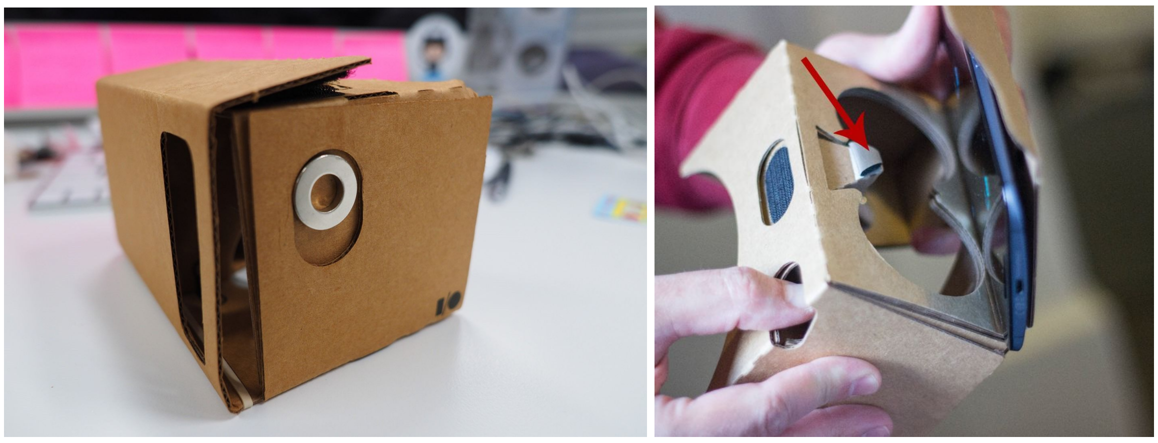


Figure 9 *Button Trigger from Google CardBoard v1 and v2*

To summarise, the second generation of Google Cardboard has a number of significant improvement to support more device on the market compared to the first generation. Although the v2.0 has a higher cost per unit and a larger packed size, its simple setup and larger amount of supporting device have overcome the disadvantage above, considering the cost of Google Cardboard v2.0 is still a lot lower than other virtual reality device such as Google Daydream and some VR headset.

4.1.3 Display Device

This project was first designed to support as more smartphone as possible while not affecting the overall quality of the product. Due to the resources limitation, the only device to be tested is my Samsung Galaxy S3 which runs a custom Android 4.4.4 version. This smartphone can run the official Google Cardboard app and some virtual reality apps from the Google Play Store without problem, so it is considered as a reference device for this virtual reality project. The game should be able to run on any Android device with version 4.4.4 or above.

As mentioned in the previous part, the interaction methods include the cardboard button and gaze input with tilt from the device, which means there must be a gyroscope sensor in the device in order to control the tilt and rotation interaction within the virtual reality game.

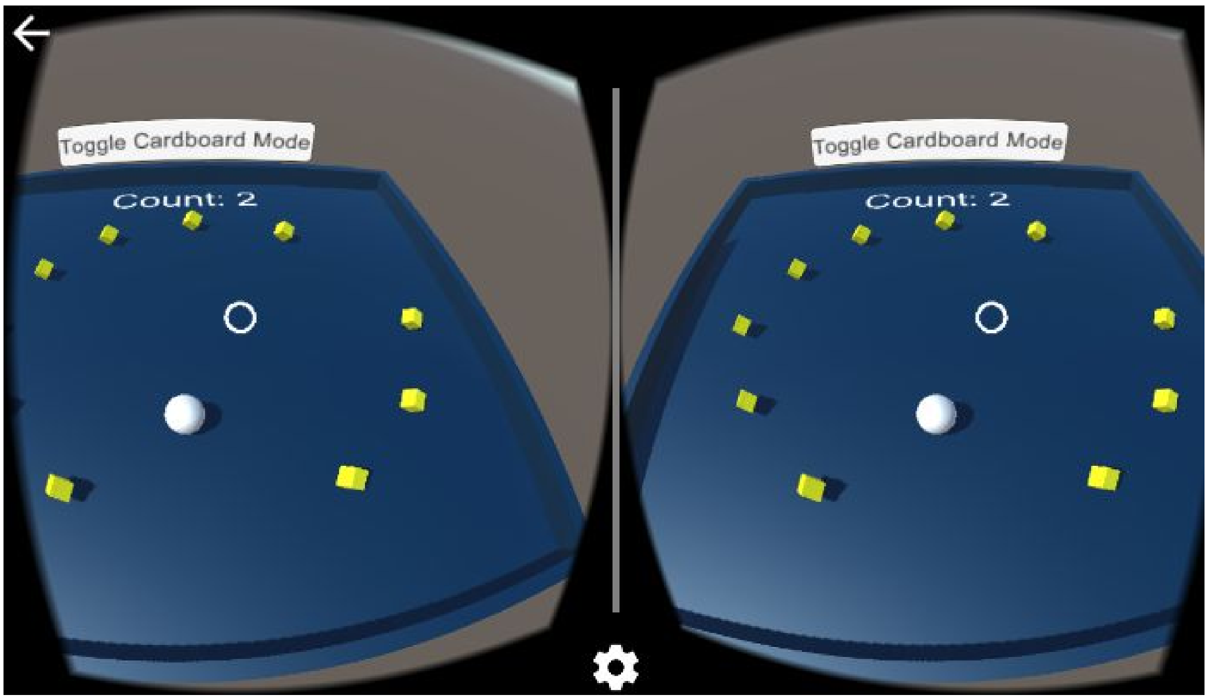


Figure 10 *A virtual reality game with gaze input (reticle)*

4.2 Binocular Vision Test

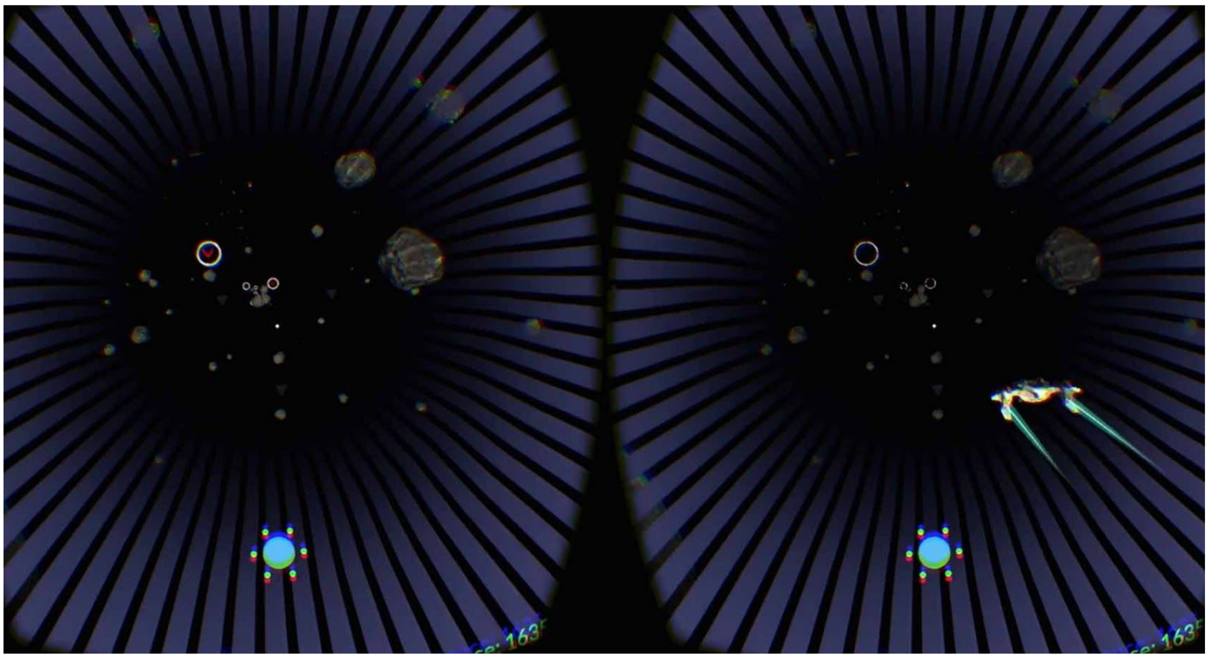
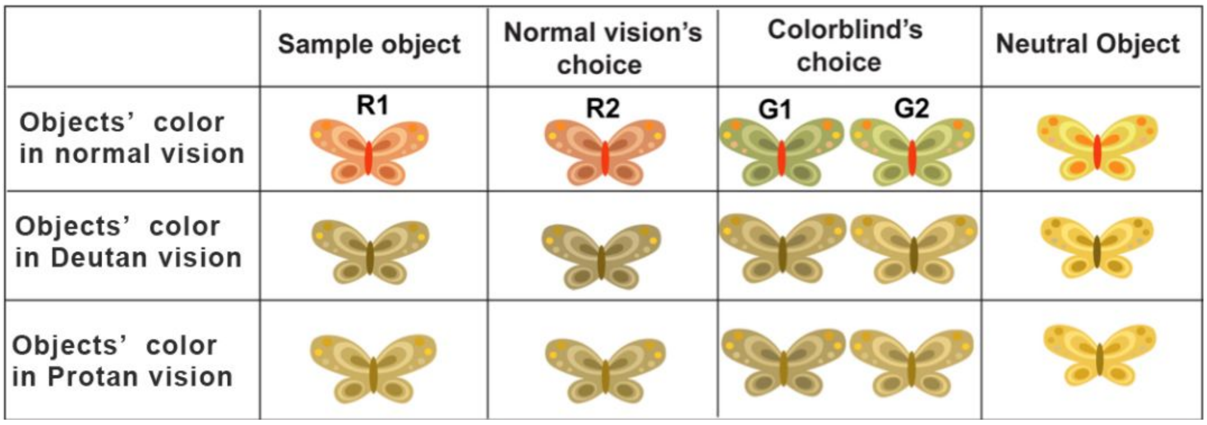


Figure 11 *A game screenshot from Vivid Vision*

The biggest advantage of using virtual reality in this project is that it can output two different contents to different eyes. In order to diagnose binocular vision related problems such as diplopia, amblyopia and strabismus, different contents should be output to different eyes, it can check whether or not the user can focus on the target object based on the limited information received from different eyes.

The idea is similar to Vivid Vision, a virtual reality game as treatment of amblyopia and strabismus[11]. The main difference is that instead of targeting on a moving object like the spaceship in Figure 11, this project uses a "Avoid and Collect" approach on the target object. Players have to use limited information on different eyes to predict the position of different objects, the objective of the game is to avoid some obstacles and to collect items, it does not only make the game more interesting, but it also increases the accuracy of the test by making the player "Move To" and "Move Away From" the targeted object.

4.3 Color Blind Test



	Sample object	Normal vision's choice	Colorblind's choice		Neutral Object
	R1	R2	G1	G2	
Objects' color in normal vision					
Objects' color in Deutan vision					
Objects' color in Protan vision					

Figure 12 *Color of game objects of DoDo's Catching Adventure in different vision[7]*

Apart from the binocular test, this project also provides color blind test on Deutran (Green deficiency) and Protan (Red deficiency) vision. Since it is expected that children are able to play the game by themselves without an examiner providing instruction, traditional color blind test like Ishihara and Farnsworth does not work as they use a "question and answer" approach which requires an examiner to perform the test.

This project uses the approach similar to the DoDo's Catching Adventure, by applying different color setting on an object, people with colorblind would see objects of different colors and this helps to diagnose the color deficiency of the player.

To make the gameplay natural, the color blind test objects are designed in a way that by applying different colors to the object, it limits the information if the player is colorblind. In other words, if the player is colorblind, they are not able to see the information (hints) on the object that the game requires. Different colors are also applied on the object so as to cover both Deutran and Protan vision.

4.4 Game design

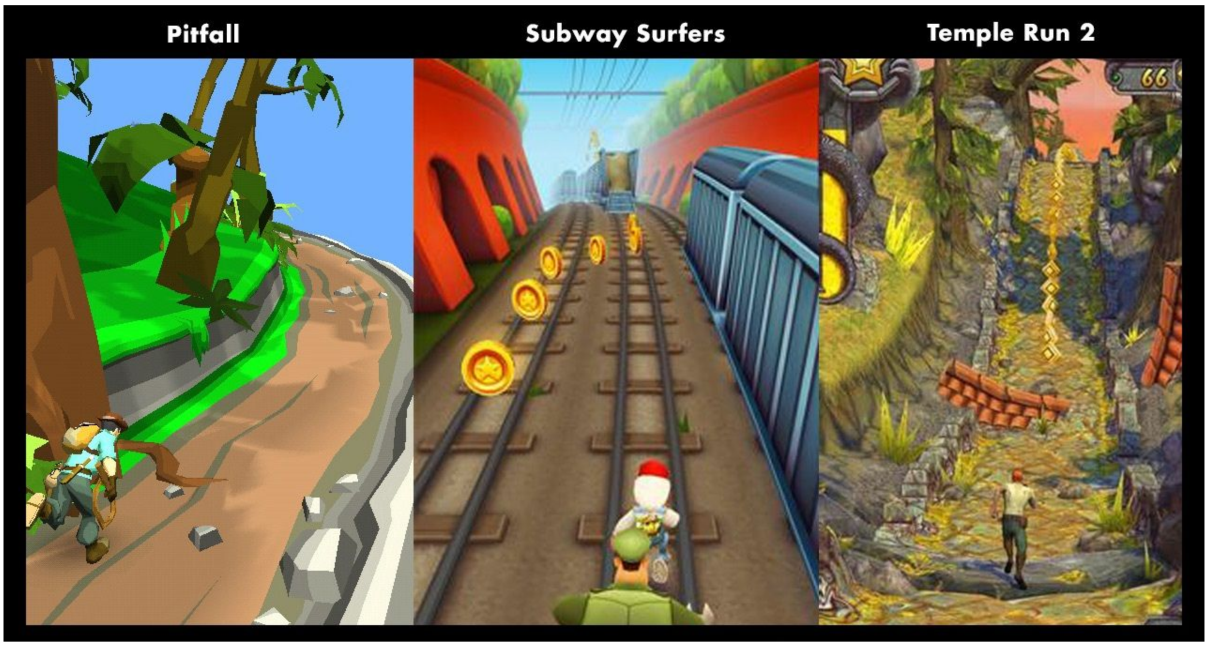


Figure 13 *Three famous endless runner games*

The main idea of the gameplay comes from a popular genre of game called "Endless Runner". There are many famous endless runner game such as Temple Run and Subway Surfers. Their characteristics match the objectives of this project so the game design of this project is based on these endless runner games.

The first characteristic is that the control is very simple, there are usually three paths in the game: left, middle and right. The player are required to swipe on the screen to change the side they are running on. The control methods are straight forward even for children. In the virtual reality environment, this control methods can be applied using the tilt from the phone's gyroscope sensor.

The second characteristic is that it is easy for the players to understand how to play the game. In a traditional endless runner game, there are usually two main objectives, which are collecting items like coins, gems and avoiding obstacle like rocks and trees to run for the longest distance, these objectives match the "Avoid and Collect" design of the binocular test of the project.

To summarize, the gameplay of this project is based on endless runner games. The game output items and obstacles to different eyes, the player needs to focus on the object with two eyes. The objective is to collect items and avoid obstacles by moving their head as this controls the tilt of the VR device. Color blind test items are also included in the game, where players need to look at the hints from the colorblind test object to chose the correct path in order to avoid points penalty. Since it is a vision screening test game, it can not be "endless". The game ends after showing a certain number of test items.

For patients who have binocular vision related problems, since the images are overlapped inappropriately, they may miss some items and hit obstacles in a higher rate. The game should identify if the player suffers from binocular vision related problems based on the number of obstacles they pass through and the items they collect during the game.

For patients who are colorblind, since they are not able to see the hidden information in the test object, they may not know the only path to avoid points deduction. The game should identify if the player are colorblind based on the number of incorrect paths they pass and it should also identify the type of color blindness based on the path selection.

The game should record the number of test objects provided and the number of test object the player passed after the game is finished. Detail information such as the layer (which side of the eye does the object output) and the tag for color blind test should also be recorded for further identification of the players' binocular vision and color deficiency. Results of different tests will be generated by the game separately and further explanations will be shown as a recommendation to parents or teachers for the visual condition of the children.

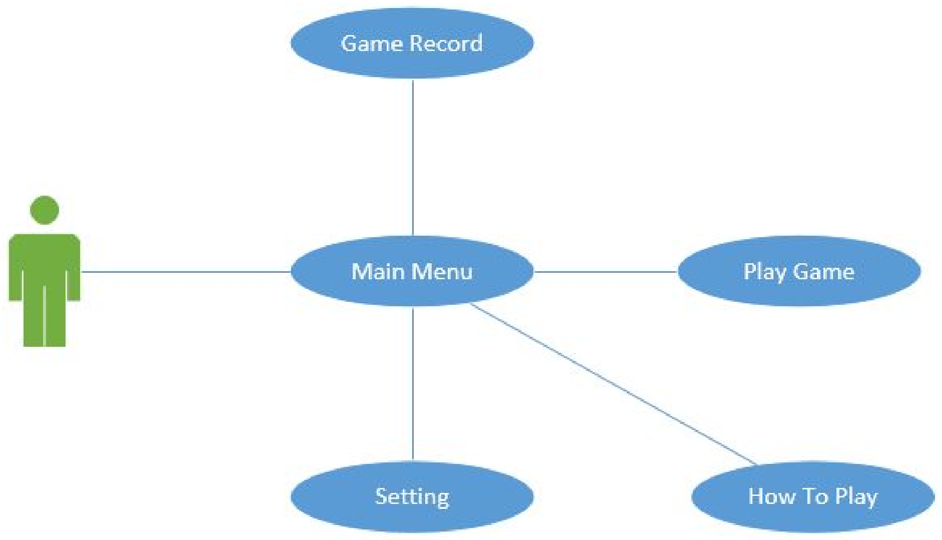


Figure 14 *A simple use case diagram of the game scenes*

Four game scenes are designed for different purposes. When players first open the game from the smartphone, it show the main menu of the game which they can go to other scenes from here.

Setting is the scene where players can adjust different game settings such as the length of the game and the spawn rate of different items. This can help players find the most suitable test setting according to their age.

How To Play is an extra scene where players can find simple instruction of the game. Since there is a chance that the player may not understand the objective and control methods of the game, simple instructions should be provided to avoid unnecessary misunderstanding about the game.

Game Record is the scene where players can check for the game record for themselves. It includes an overall result like a normal game with the time and score, players can also view a more detailed record with a brief report on their binocular vision and color blind test result.

Main Game is the scene where players perform a series of tests while playing the game. This scene is displayed in the “Virtual Reality” Mode, where players can only interact with the game by tilting the device with their head and by pressing the cardboard button.

5. Game Implementation

5.1 System Setting

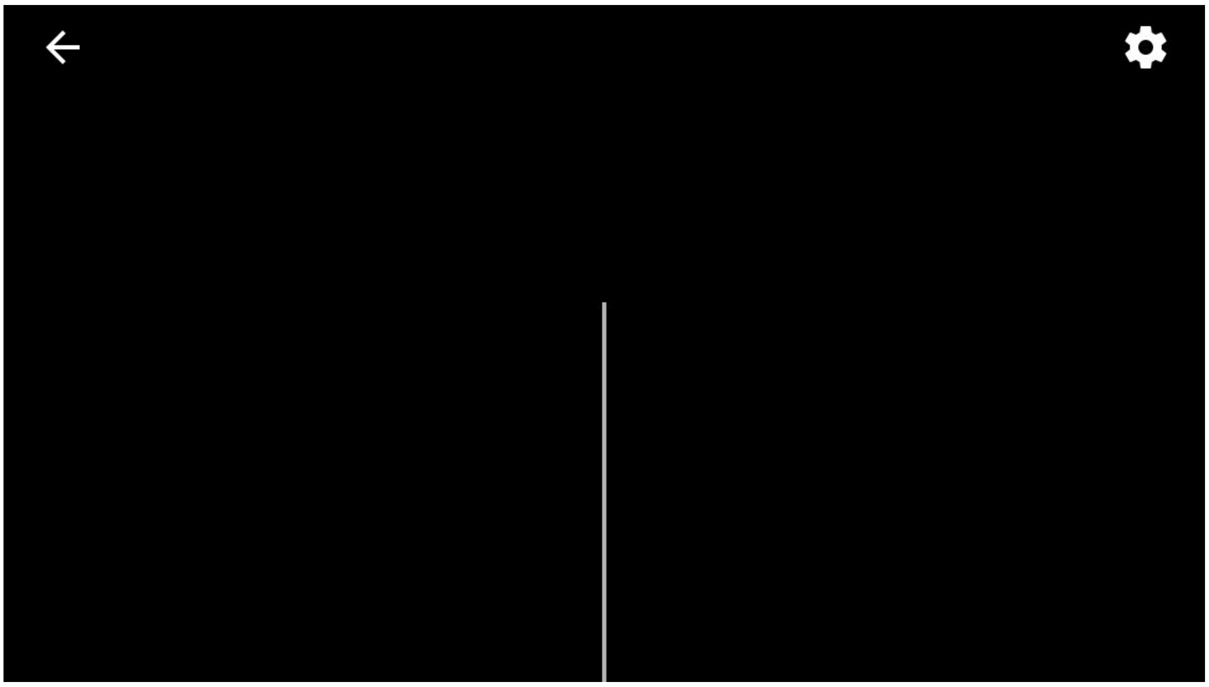


Figure 15 *Rendering problem of the virtual reality app created with Unity*

In the early development stage, the newest version of Unity Game Engine and Google Cardboard SDK were used in the testing. However, a rendering problem occurs on the testing device runs on the Android 4.4.4 platform, the app worked fine in the Unity Game Engine but when it was output to the device, it showed an empty screen in Figure 15.

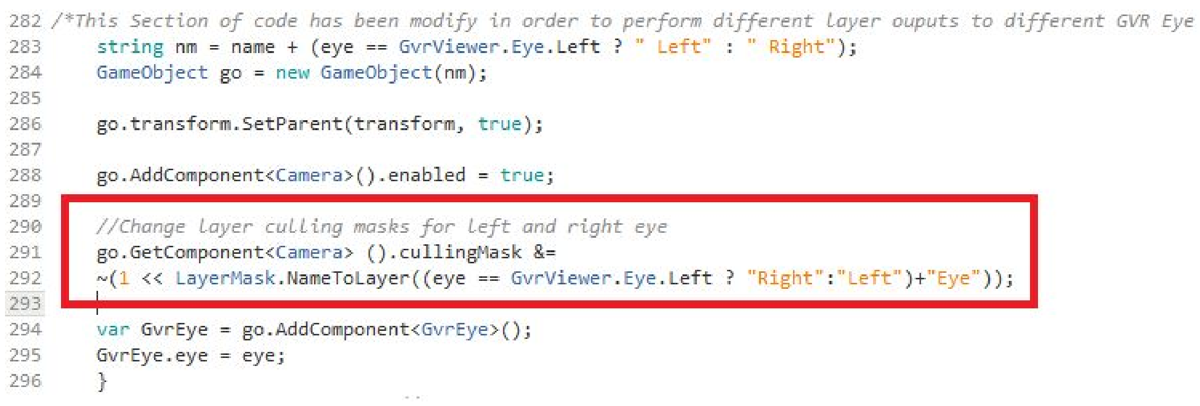
After trying different combinations of versions of Unity Game Engine and Google Cardboard SDK, the prototype app was finally able to run on the test device without any problem using Unity 5.3.3 f1 with Google Cardboard SDK 1.0.0. Although virtual reality is a new technology and the related tools keep updating, the same stable versions are used throughout the development process of this project to avoid unnecessary errors. It can also support more devices by covering those running on an older Android platform.

5.1.1 Google Cardboard SDK changes

The original idea of displaying different contents to different eyes is to use layer in Unity. However the Google Cardboard SDK creates the left and right eye camera from a parent camera in runtime, which means the behaviour of the child cameras would inherit the behaviour of the parent camera, including the layer culling mask.

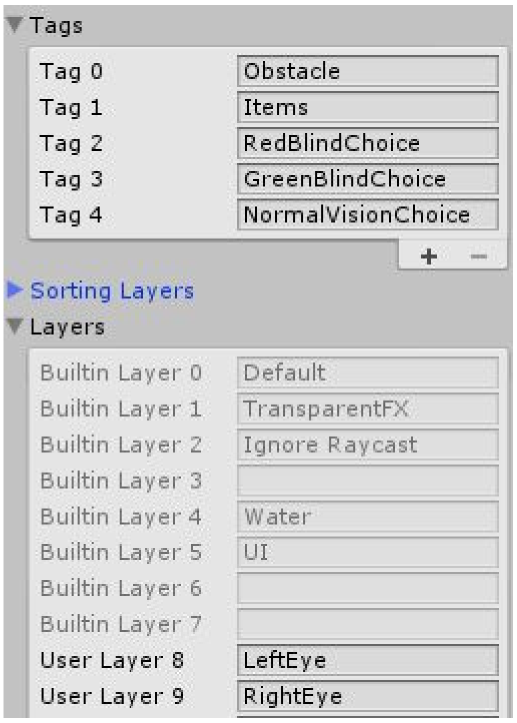
In order to display different content to different eyes, a change in the Google Cardboard SDK was made by not showing the object to another eye in the culling mask layer. A line of code to manually change the culling mask of the VR camera was added in the SteroController script of the Google Cardboard SDK.

When the game starts, the GVR Main component calls this controller function to generate two camera for each of the eye. Since the default camera displays all layers, by disabling the Right layer on the left eye and the Left layer on the right eye, objects in different layers can be output to the desired camera only.



```
/*This Section of code has been modify in order to perform different layer ouputs to different GVR Eye
    string nm = name + (eye == GvrViewer.Eye.Left ? " Left" : " Right");
    GameObject go = new GameObject(nm);

    go.transform.SetParent(transform, true);

    go.AddComponent<Camera>().enabled = true;

    //Change layer culling masks for left and right eye
    go.GetComponent<Camera> ().cullingMask &=
    ~(1 << LayerMask.NameToLayer((eye == GvrViewer.Eye.Left ? "Right":"Left")+"Eye"));

    var GvrEye = go.AddComponent<GvrEye>();
    GvrEye.eye = eye;
    }
```

Figure 16 A part of SteroController script in Goole Cardboard SDK



Tags

Tag 0	Obstacle
Tag 1	Items
Tag 2	RedBlindChoice
Tag 3	GreenBlindChoice
Tag 4	NormalVisionChoice

Sorting Layers

Layers

Builtin Layer 0	Default
Builtin Layer 1	TransparentFX
Builtin Layer 2	Ignore Raycast
Builtin Layer 3	
Builtin Layer 4	Water
Builtin Layer 5	UI
Builtin Layer 6	
Builtin Layer 7	
User Layer 8	LeftEye
User Layer 9	RightEye

Figure 17 *List of Tags and Layers in the game*

A list of layers has been added so the left and right camera can display the corresponding content according to the layer of the objects. A prototype game was created to test the layer culling mask with different layer setting. In Figure 18 below, three different layers (Left, Right, Default) have been applied on different objects.

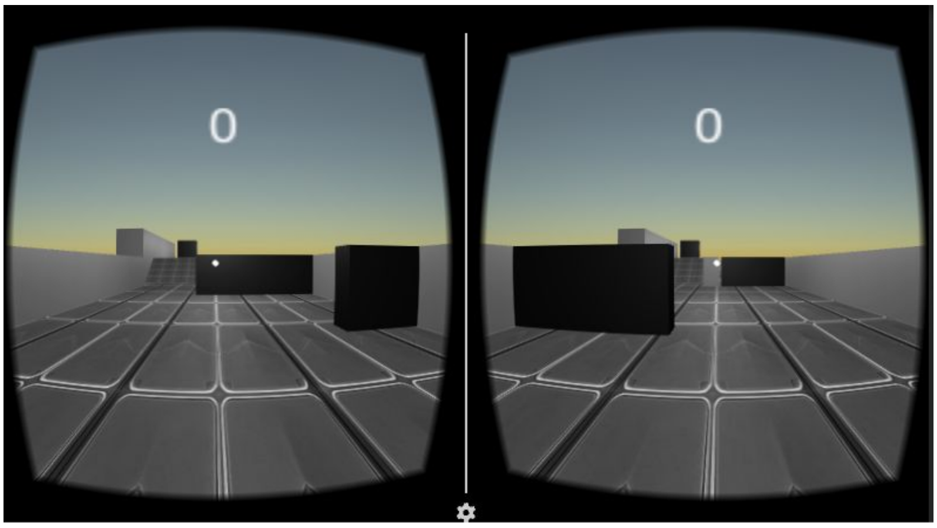


Figure 18 *The prototype game screen capture*

5.2 Main Menu

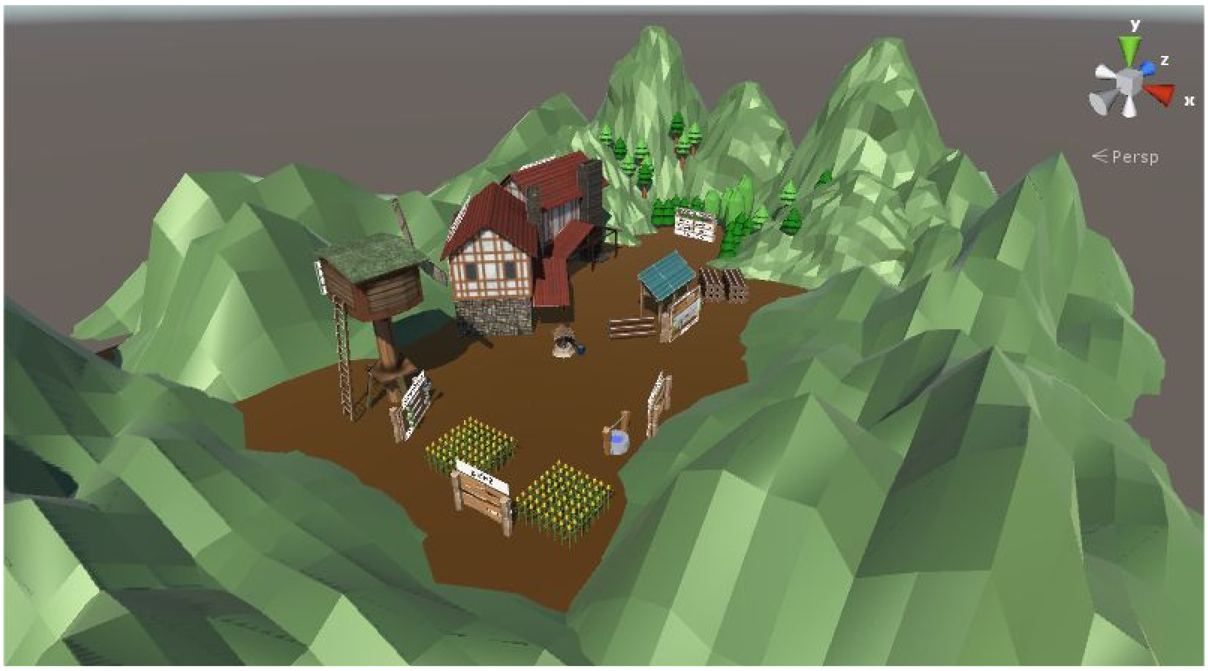
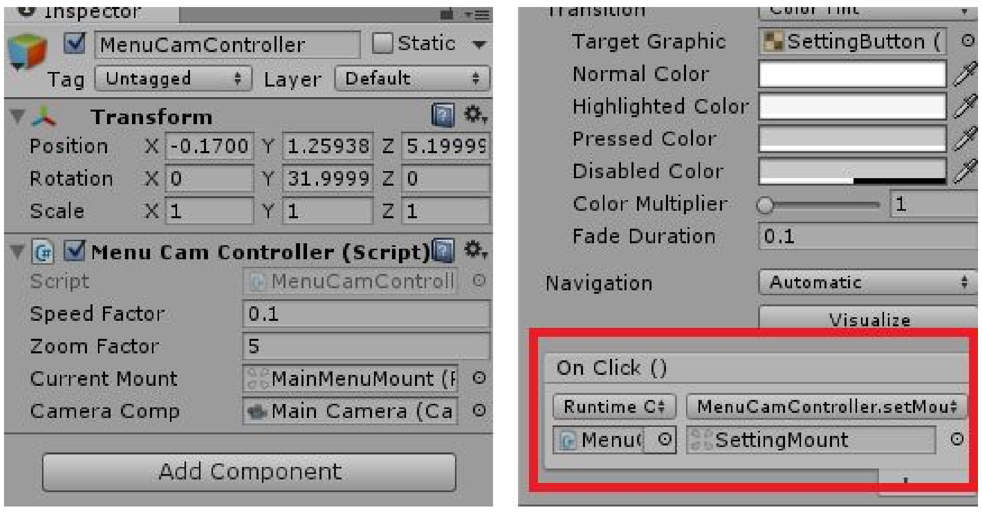


Figure 19 *Game scene of the main menu*

To reduce the loading time between each menu, they are all put in the same scene with a world space UI Canvas.


Inspector
MenuCamController   Static
Tag Untagged   Layer Default
Transform
Position X -0.1700 Y 1.25938 Z 5.19999
Rotation X 0 Y 31.9999 Z 0
Scale X 1 Y 1 Z 1
Menu Cam Controller (Script)
Script MenuCamControll
Speed Factor 0.1
Zoom Factor 5
Current Mount MainMenuMount (I
Camera Comp Main Camera (Ca
Add Component

Transition Color Tint
Target Graphic SettingButton (
Normal Color
Highlighted Color
Pressed Color
Disabled Color
Color Multiplier 1
Fade Duration 0.1
Navigation Automatic
Visualize
On Click ()
Runtime C   MenuCamController.setMou
Menu(SettingMount

Figure 20 *Menu Cam Controller Script*

A script is created to control the position of the camera when a button is clicked. This can move the main camera to the corresponding position according to which button the player clicks.

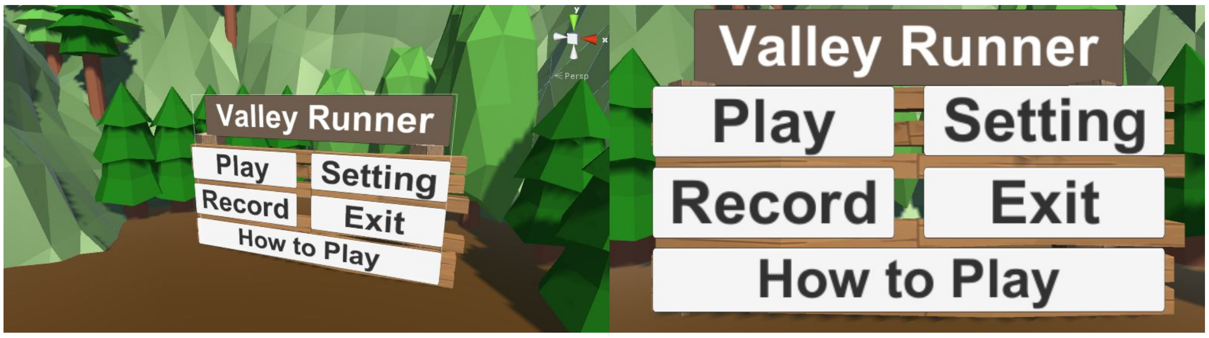


Figure 21 Main menu in the game scene(Left) and in the game(Right)

5.2.1 How To Play Menu

The How To Play Menu uses simple images so children should be able to understand the instructions. Moreover, the instruction of the color blind test is designed in a way that players who are colorblind should also see the same thing as non-colorblind players.

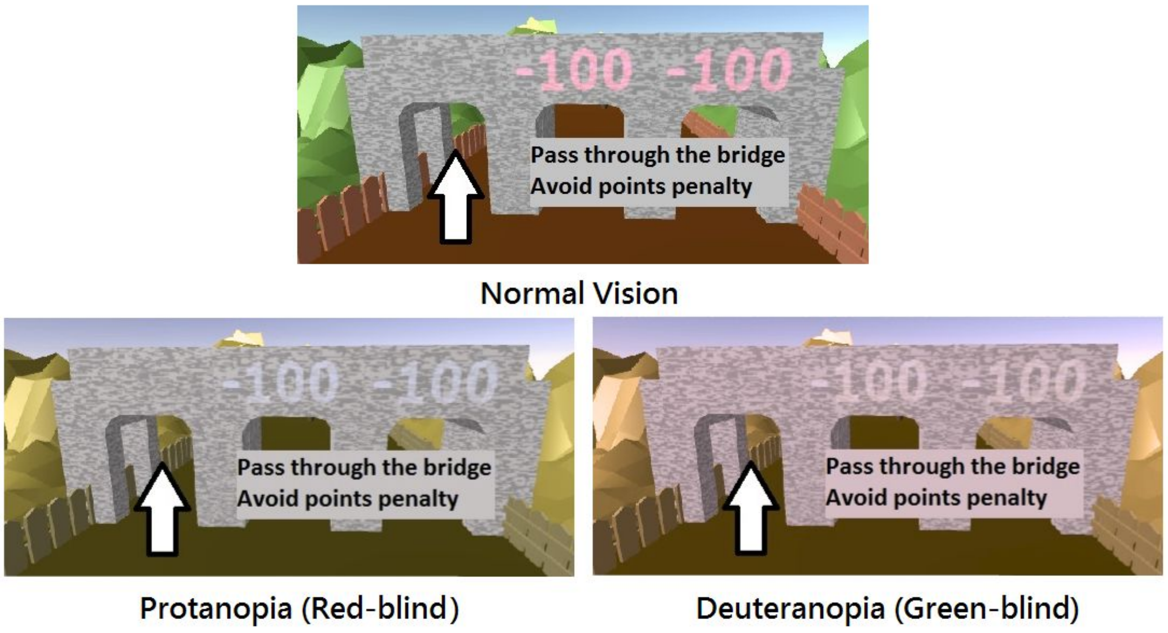


Figure 22 *How To Play menu in different vision*

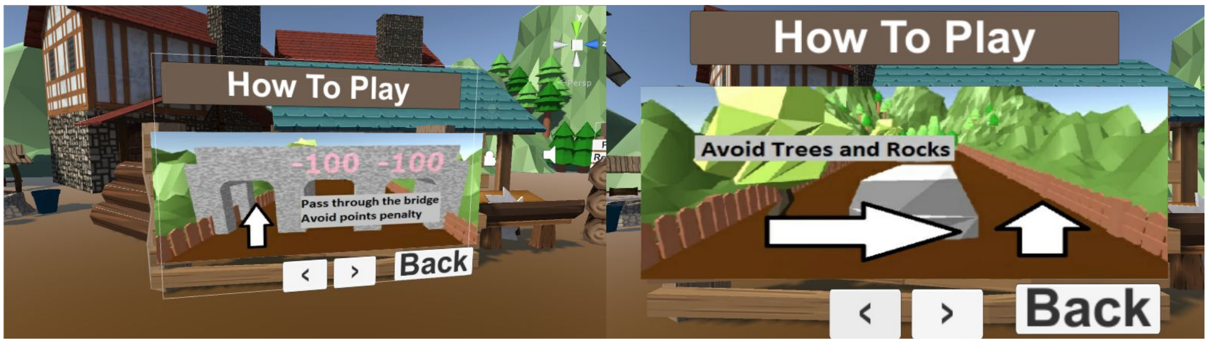
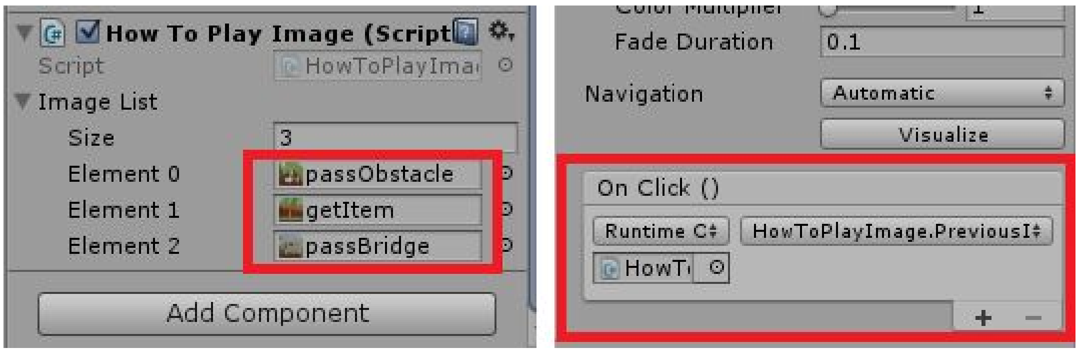


Figure 23 *How To Play Menu in the game scene(Left) and the game(Right)*

A simple script is created so the player can change the image by clicking the left and right button. When a button is clicked, the texture of the image changes according to the list of images.


How To Play Image (Script)
Script HowToPlayIma
Image List
Size 3
Element 0 passObstacle
Element 1 getItem
Element 2 passBridge
Add Component

Fade Duration 0.1
Navigation Automatic
Visualize
On Click ()
Runtime C HowToPlayImage.PreviousI
HowT

Figure 24 How To Play Image Controller Script

5.2.2 Setting Menu

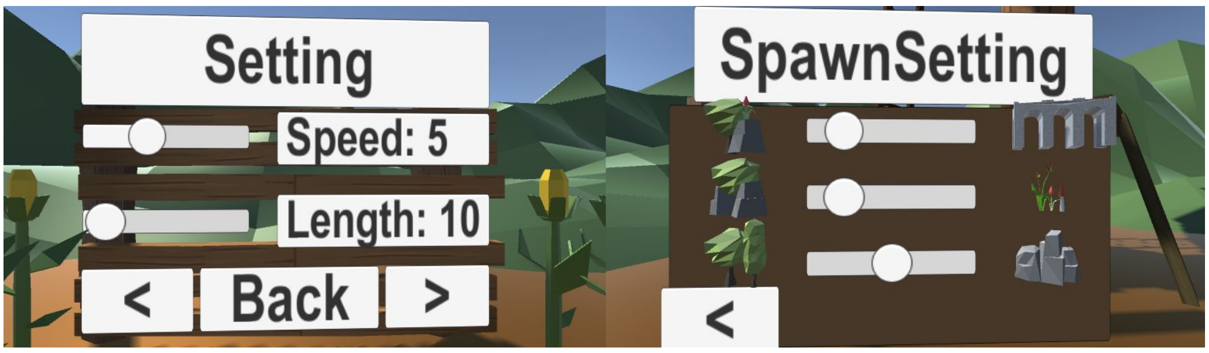
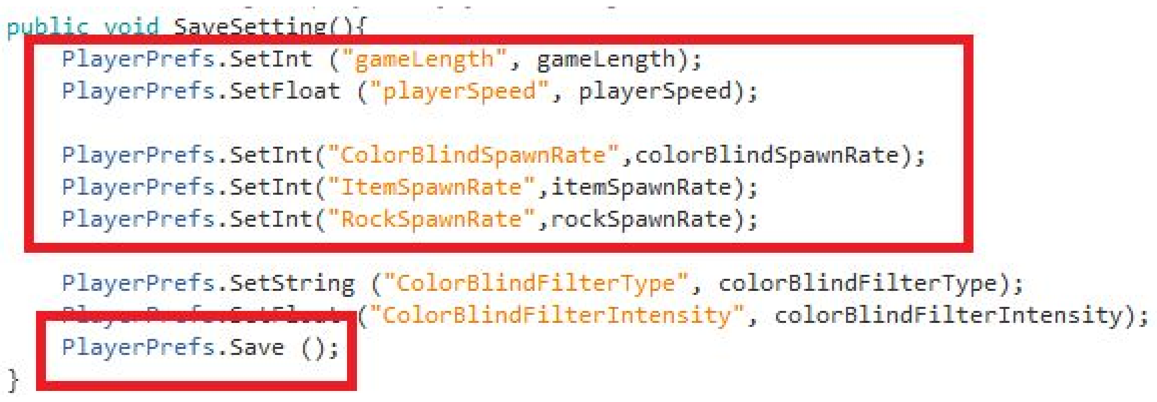


Figure 25 *Game Setting Menu and Spawn Setting Menu*

The original idea of the game setting and spawn setting menu is to find out the best setting for a vision screening test by adjusting different parameters. The game settings parameters are saved in the game player prefabs, so even if the player close the game and reopen it, the game setting saved in previous session can be loaded from the Unity PlayerPrefs class. The game setting can also be loaded from the main game scene.



```
public void SaveSetting(){
    PlayerPrefs.SetInt ("gameLength", gameLength);
    PlayerPrefs.SetFloat ("playerSpeed", playerSpeed);

    PlayerPrefs.SetInt("ColorBlindSpawnRate",colorBlindSpawnRate);
    PlayerPrefs.SetInt("ItemSpawnRate",itemSpawnRate);
    PlayerPrefs.SetInt("RockSpawnRate",rockSpawnRate);

    PlayerPrefs.SetString ("ColorBlindFilterType", colorBlindFilterType);
    PlayerPrefs.SetFloat ("ColorBlindFilterIntensity", colorBlindFilterIntensity);
    PlayerPrefs.Save ();
}
```

Figure 26 *Save Setting Function in Game Setting Script*

5.2.3 Record and Report

The game record menu includes a brief record and a full record. The record menu shows only most recent record.

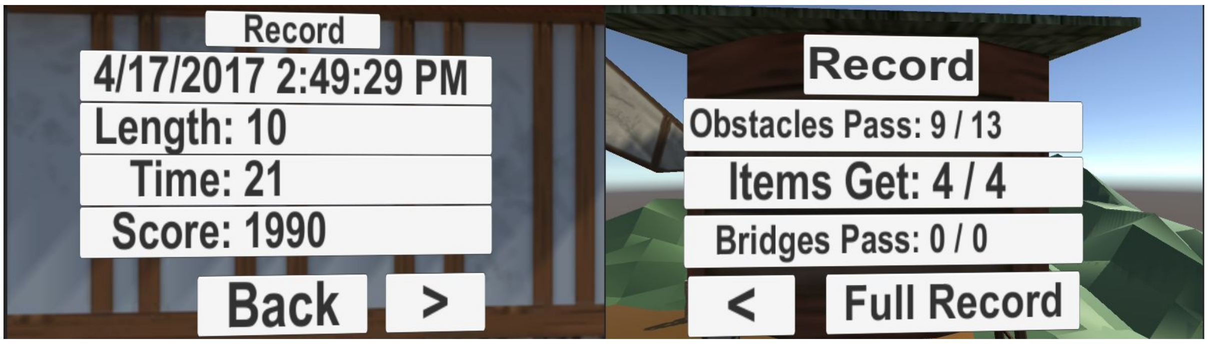
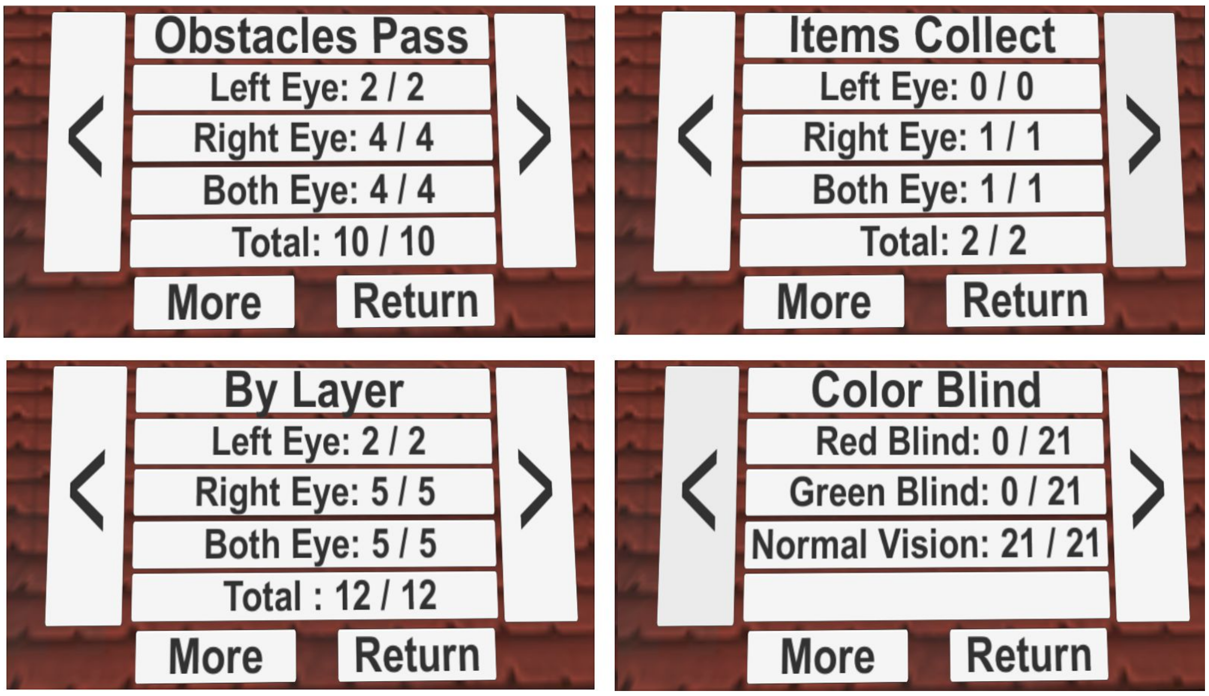

Record
4/17/2017 2:49:29 PM
Length: 10
Time: 21
Score: 1990
Record
Obstacles Pass: 9 / 13
Items Get: 4 / 4
Bridges Pass: 0 / 0

Figure 27 *Brief Record Menu*

The brief record includes the record time, the length of the game, the finish time and score. The score is calculated based on the number of items collected, the number of obstacles passed and the finish time of the game with the game length setting. The score is not a reference of the test result, as it can easily be affected by the number of objects spawned and the game length.

The second part of the brief report includes the total number of obstacles passed and the number of items collected for binocular vision test, while the bridges passed is the color blind test result. As the spawn setting can be changed in the game, the game can be adjusted for color blind test and binocular vision test separately.


Obstacles Pass
Left Eye: 2 / 2
Right Eye: 4 / 4
Both Eye: 4 / 4
Total: 10 / 10

Items Collect
Left Eye: 0 / 0
Right Eye: 1 / 1
Both Eye: 1 / 1
Total: 2 / 2

By Layer
Left Eye: 2 / 2
Right Eye: 5 / 5
Both Eye: 5 / 5
Total : 12 / 12

Color Blind
Red Blind: 0 / 21
Green Blind: 0 / 21
Normal Vision: 21 / 21

Figure 28 *Detail Record Menu*

Besides of the brief report, the game also provides a detailed report including the layers of the test objects and the color blind selection. This helps to give more information about the test that the player has performed. A test result is also provided about the player's binocular vision and color deficiency.

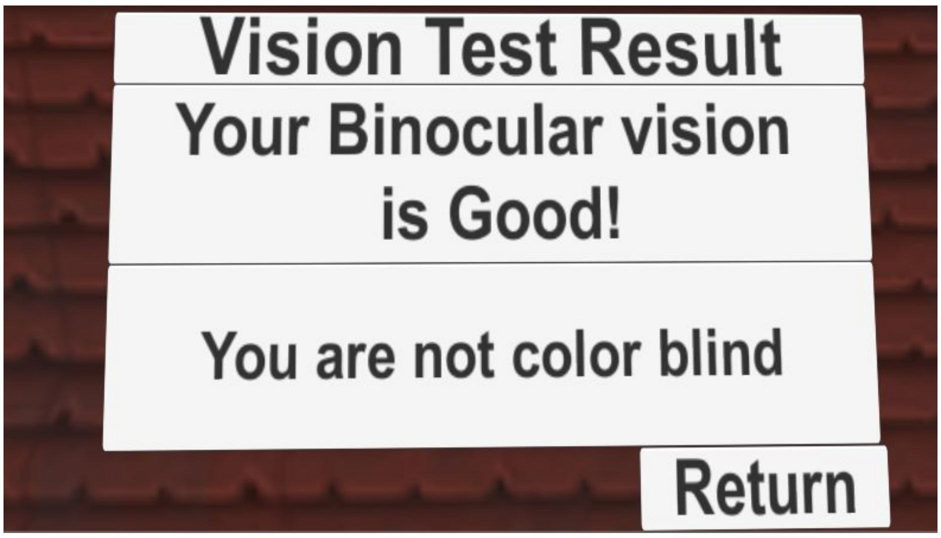

Vision Test Result
Your Binocular vision is Good!
You are not color blind

Figure 29 *Vision Test Result Menu*

5.3 Main Game

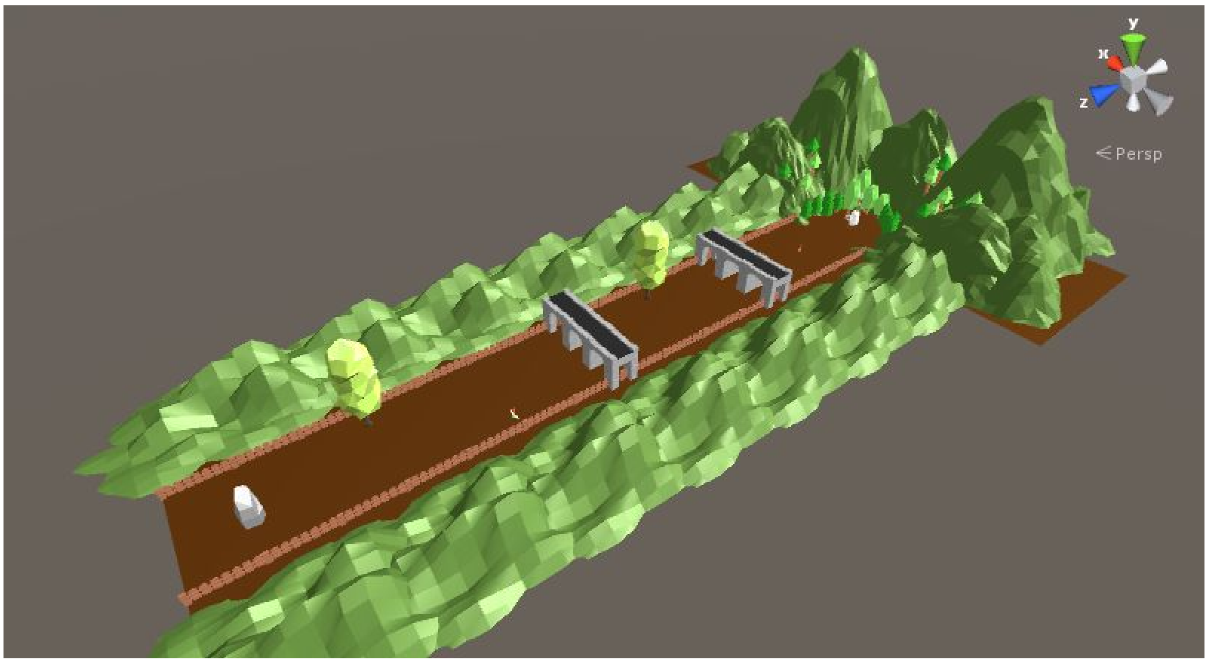


Figure 30 *Game scene of the main game*

Since the game is a 3D virtual reality game, it may cause large amount of CPU usage of the mobile devices for the rendering. In order to reduce the memory cost and CPU usage, the game use limited amount of objects and the 3D objects are all in low-poly style. The game should be able to loaded quickly and run smoothly on mobile devices.

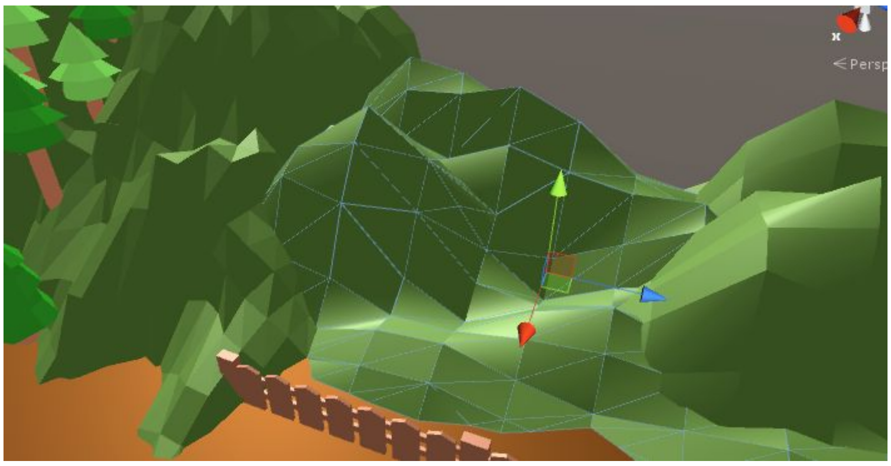


Figure 31 *Mesh outline of a low-poly mountain object*

5.3.1 Player Movement

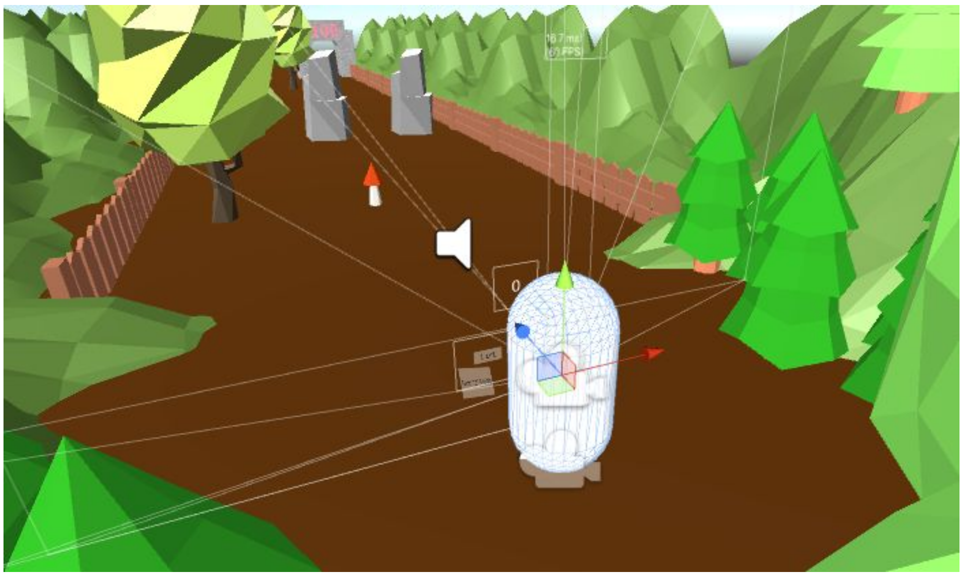
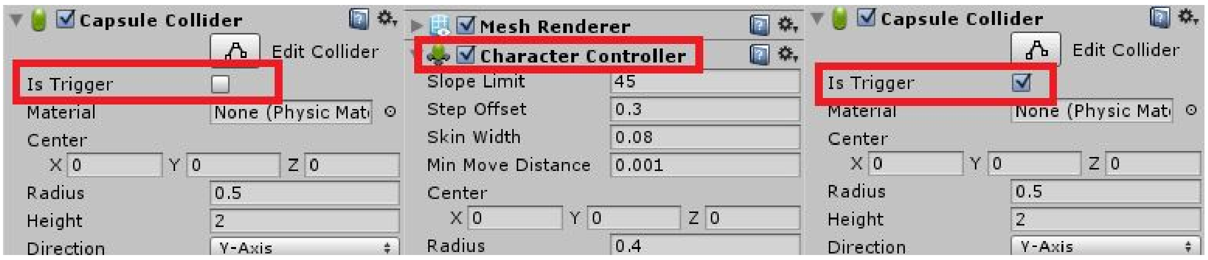


Figure 32 *Player character in the game*

This is a first person virtual reality game, so players are not able to see their character in the game. To keep the game design simple, a capsule 3D object is created as the main character.

A character controller and two capsule colliders are attached to the player object. The character controller is the core component to make the character move with a script. The first capsule collider which is non-Trigger is a collider that handles collisions between the character and non-test objects such as the environment in the starting position and the fences on the two sides, these collision are not counted as hitting obstacles. The second capsule collider which is a trigger handles collisions between the character and all test objects such as the items and obstacles, the collisions are handled in the script attached to the test object which is explained in the next part of the report in the section of game collision.

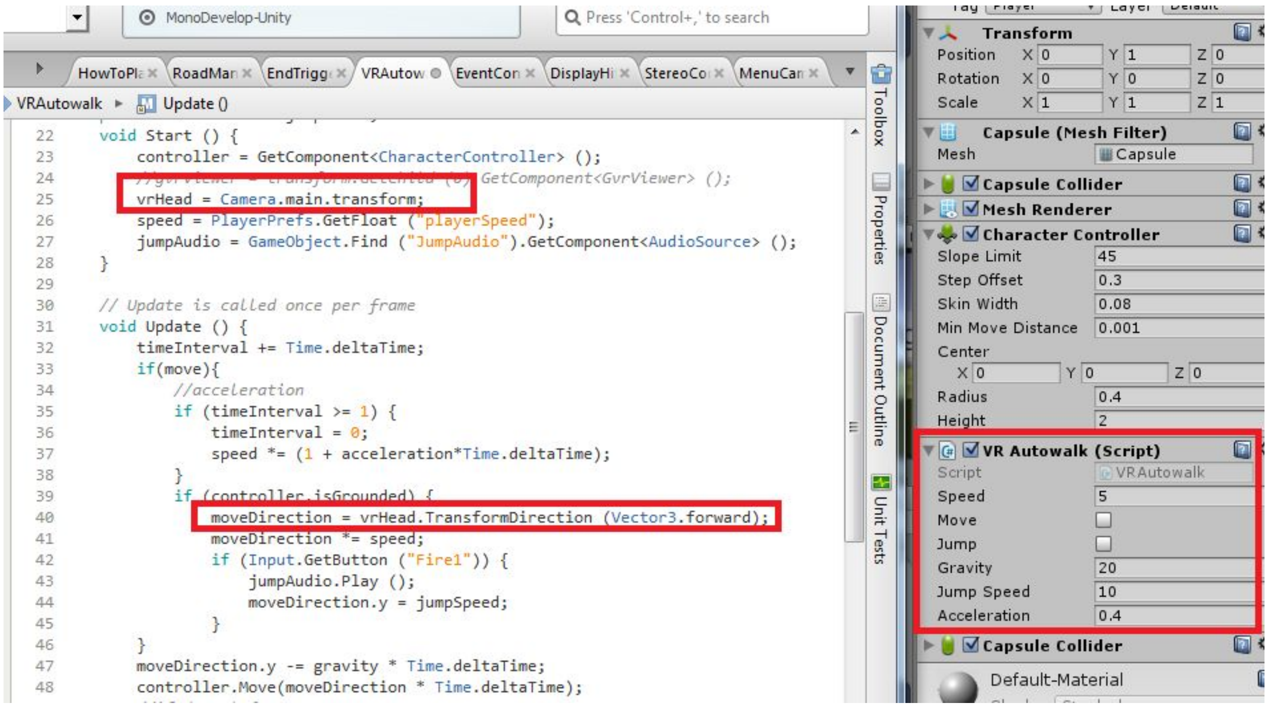


```
Capsule Collider                      Edit Collider
Is Trigger       ☐
Material         None (Physic Mat
Center           X 0    Y 0    Z 0
Radius           0.5
Height           2
Direction        Y-Axis
```

```
Mesh Renderer
Character Controller
Slope Limit          45
Step Offset          0.3
Skin Width           0.08
Min Move Distance    0.001
Center               X 0    Y 0    Z 0
Radius               0.4
```

```
Capsule Collider                      Edit Collider
Is Trigger       ☑
Material         None (Physic Mat
Center           X 0    Y 0    Z 0
Radius           0.5
Height           2
Direction        Y-Axis
```

Figure 33 *Collider and Character Controller components in player object*



```
    void Start () {
        controller = GetComponent<CharacterController> ();
        //gvrViewer = transform.GetChild (0).GetComponent<GvrViewer> ();
        vrHead = Camera.main.transform;
        speed = PlayerPrefs.GetFloat ("playerSpeed");
        jumpAudio = GameObject.Find ("JumpAudio").GetComponent<AudioSource> ();
    }

    // Update is called once per frame
    void Update () {
        timeInterval += Time.deltaTime;
        if(move){
            //acceleration
            if (timeInterval >= 1) {
                timeInterval = 0;
                speed *= (1 + acceleration*Time.deltaTime);
            }
            if (controller.isGrounded) {
                moveDirection = vrHead.TransformDirection (Vector3.forward);
                moveDirection *= speed;
                if (Input.GetButton ("Fire1")) {
                    jumpAudio.Play ();
                    moveDirection.y = jumpSpeed;
                }
        }
        moveDirection.y -= gravity * Time.deltaTime;
        controller.Move(moveDirection * Time.deltaTime);
```

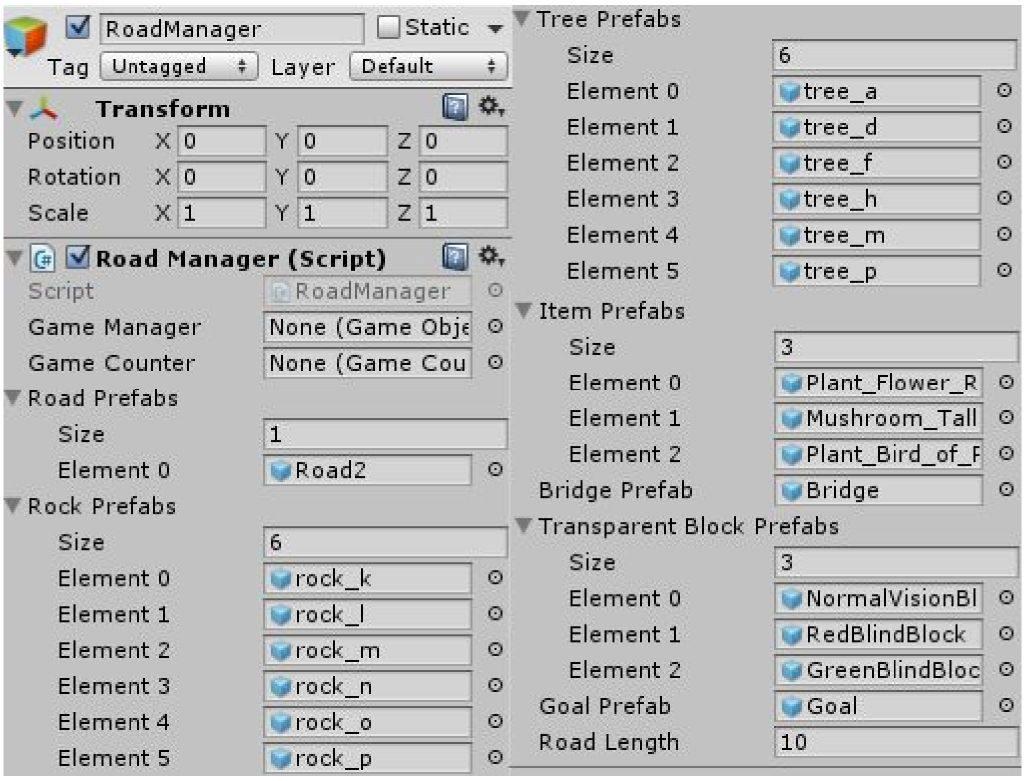
```
Transform
Position    X 0   Y 1   Z 0
Rotation    X 0   Y 0   Z 0
Scale       X 1   Y 1   Z 1
Capsule (Mesh Filter)
Mesh                 Capsule
☑ Capsule Collider
☑ Mesh Renderer
☑ Character Controller
Slope Limit          45
Step Offset          0.3
Skin Width           0.08
Min Move Distance    0.001
Center
   X 0        Y 0        Z 0
Radius               0.4
Height               2
☑ VR Autowalk (Script)
Script               VRAutowalk
Speed                5
Move                 ☐
Jump                 ☐
Gravity              20
Jump Speed           10
Acceleration         0.4
☑ Capsule Collider
Default-Material
```

Figure 34 *Player movement in VR Autowalk Script*

As player control the rotation of the character using the gyroscope sensor by rotating their head, the player movement script need to get the vrHead component which keeps track on the rotation of the device. This script makes the character moves towards the position that the player is facing continuous until it reaches the goal.

There are some important public variables in the script. Speed is the initial speed of the player, the speed can increase during the game with acceleration. The Move and jump trigger the moving state and jump function of the player.

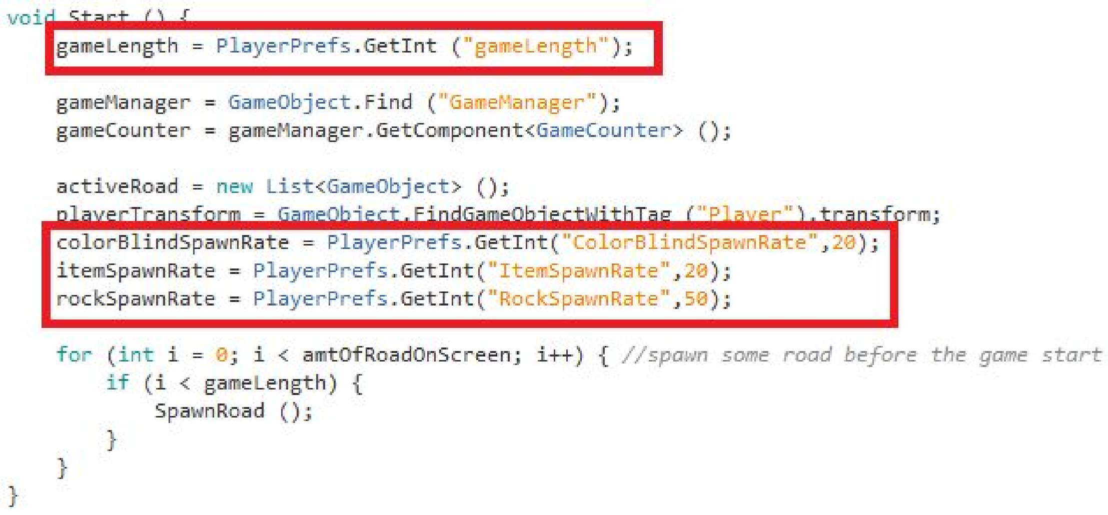
5.3.2 Object Spawning



```
RoadManager   Static
Tag Untagged   Layer Default
Transform
Position   X 0   Y 0   Z 0
Rotation   X 0   Y 0   Z 0
Scale      X 1   Y 1   Z 1
Road Manager (Script)
Script          RoadManager
Game Manager    None (Game Obje
Game Counter    None (Game Cou
Road Prefabs
  Size          1
  Element 0     Road2
Rock Prefabs
  Size          6
  Element 0     rock_k
  Element 1     rock_l
  Element 2     rock_m
  Element 3     rock_n
  Element 4     rock_o
  Element 5     rock_p
Tree Prefabs
  Size          6
  Element 0     tree_a
  Element 1     tree_d
  Element 2     tree_f
  Element 3     tree_h
  Element 4     tree_m
  Element 5     tree_p
Item Prefabs
  Size          3
  Element 0     Plant_Flower_R
  Element 1     Mushroom_Tall
  Element 2     Plant_Bird_of_F
Bridge Prefab   Bridge
Transparent Block Prefabs
  Size          3
  Element 0     NormalVisionBl
  Element 1     RedBlindBlock
  Element 2     GreenBlindBloc
Goal Prefab     Goal
Road Length     10
```

Figure 35 *RoadManager with a list of prefabs to be spawned*

As a endless runner game, a road manager script is required to spawn new objects and delete old objects throughout the game. The road manage in this project contain a public list of different type of items to be spawn.



```
void Start () {
    gameLength = PlayerPrefs.GetInt ("gameLength");

    gameManager = GameObject.Find ("GameManager");
    gameCounter = gameManager.GetComponent<GameCounter> ();

    activeRoad = new List<GameObject> ();
    playerTransform = GameObject.FindGameObjectWithTag ("Player").transform;
    colorBlindSpawnRate = PlayerPrefs.GetInt("ColorBlindSpawnRate",20);
    itemSpawnRate = PlayerPrefs.GetInt("ItemSpawnRate",20);
    rockSpawnRate = PlayerPrefs.GetInt("RockSpawnRate",50);

    for (int i = 0; i < amtOfRoadOnScreen; i++) { //spawn some road before the game start
        if (i < gameLength) {
            SpawnRoad ();
        }
    }
}
```

Figure 36 *Intialization of RoadManager script*

Before the game starts, the road manager loads the spawn setting from the PlayerPrefs so it can spawn the corresponding objects during the game. After the spawn setting is loaded, the road manager spawns some roads and objects before the player click the play button.

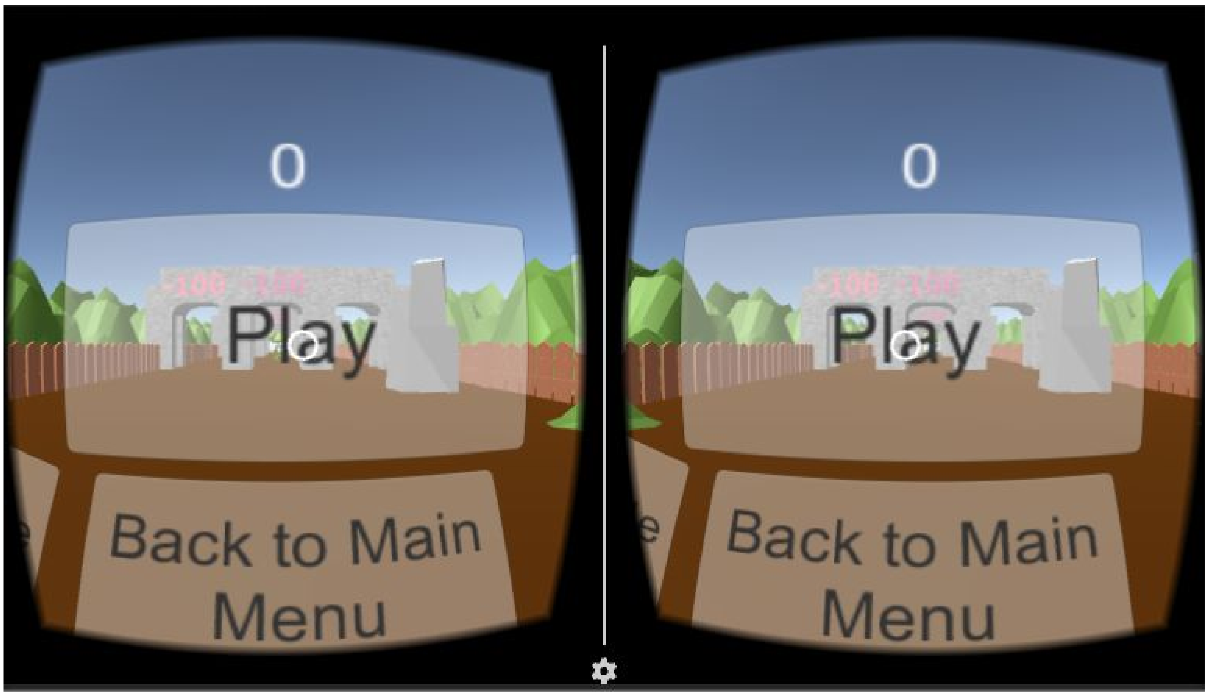
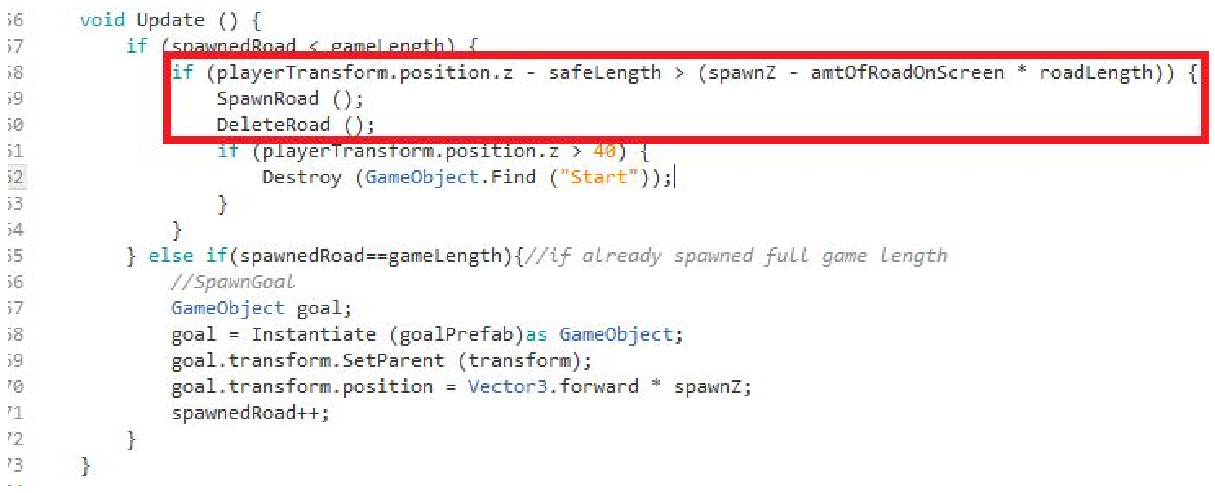


Figure 37 *Main Game menu before the game starts*



```
void Update () {
    if (spawnedRoad < gameLength) {
        if (playerTransform.position.z - safeLength > (spawnZ - amtOfRoadOnScreen * roadLength)) {
            SpawnRoad ();
            DeleteRoad ();
            if (playerTransform.position.z > 40) {
                Destroy (GameObject.Find ("Start"));
            }
        }
    } else if(spawnedRoad==gameLength){//if already spawned full game length
        //SpawnGoal
        GameObject goal;
        goal = Instantiate (goalPrefab)as GameObject;
        goal.transform.SetParent (transform);
        goal.transform.position = Vector3.forward * spawnZ;
        spawnedRoad++;
    }
}
```

Figure 38 *Road Manager main update function*

The road manager spawn new objects and delete old objects according to the z position of the player. The safe Length is a value to make sure that there is number of roads behind the player when older roads are going to be deleted, while the amount of road on screen defines the number of road displayed in the world, a proper setting can reduce the rendering cost while keeping the game looks natural.

The starting platform is also deleted after the player has walked for a long distance, while the goal platform should be spawn after the number of game length of roads are spawned.

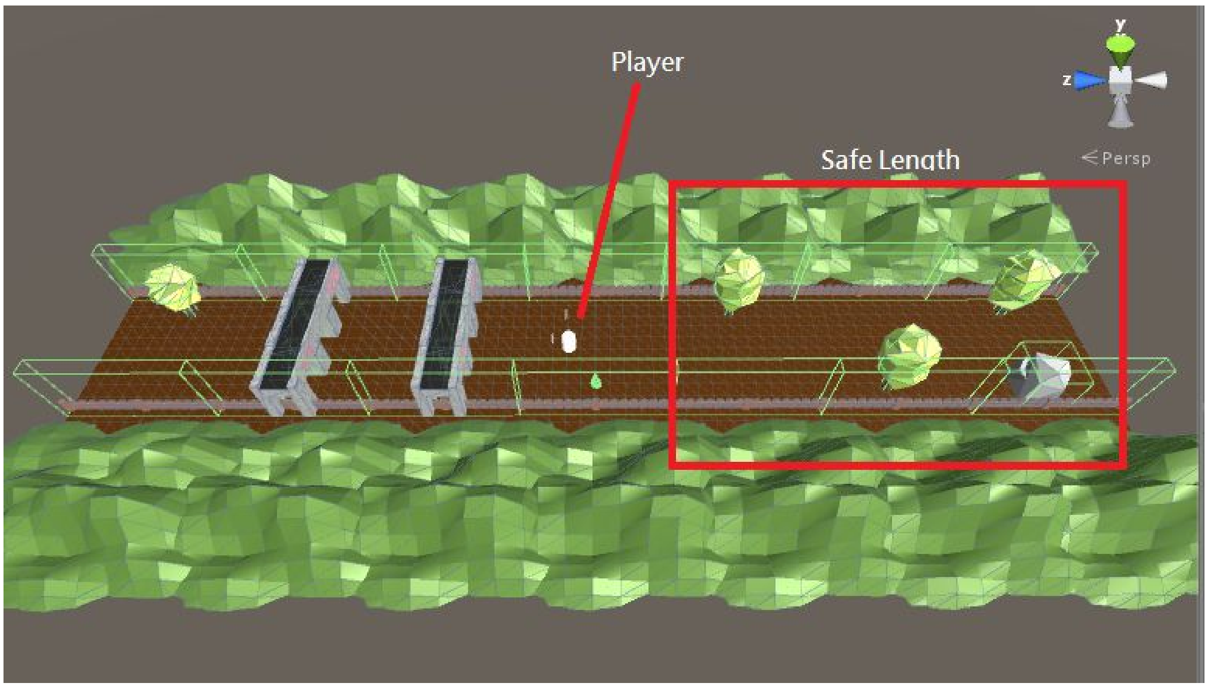
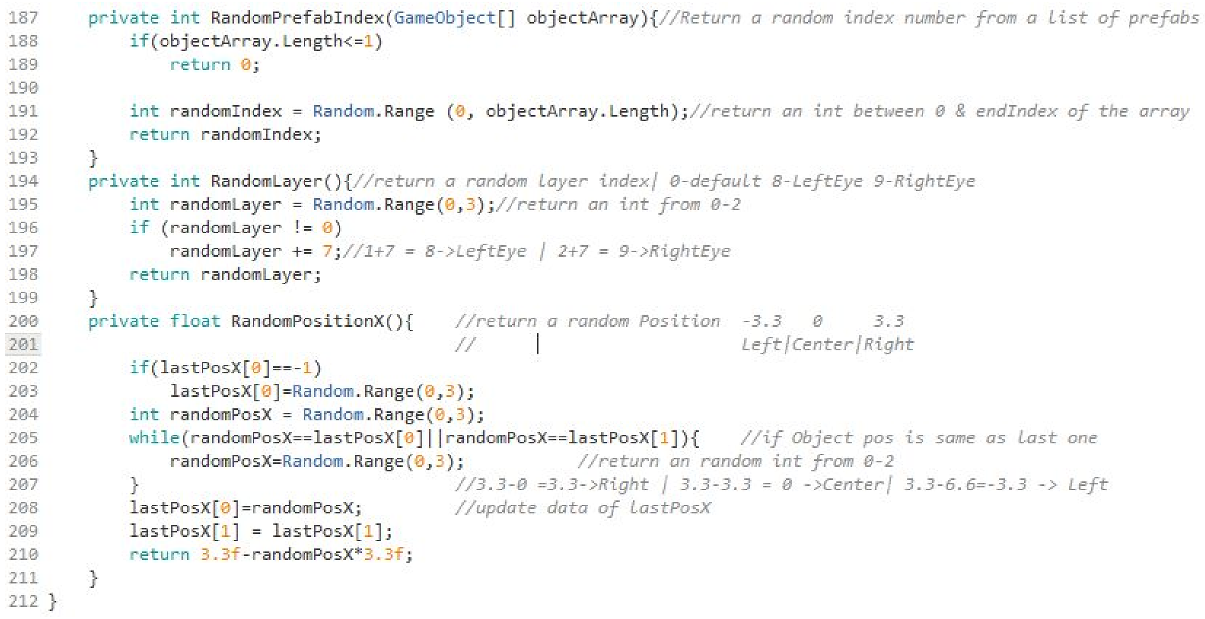


Figure 39 *Safe Length setting in the main game*



```
    private int RandomPrefabIndex(GameObject[] objectArray){//Return a random index number from a list of prefabs
        if(objectArray.Length<=1)
            return 0;

        int randomIndex = Random.Range (0, objectArray.Length);//return an int between 0 & endIndex of the array
        return randomIndex;
    }
    private int RandomLayer(){//return a random layer index| 0-default 8-LeftEye 9-RightEye
        int randomLayer = Random.Range(0,3);//return an int from 0-2
        if (randomLayer != 0)
            randomLayer += 7;//1+7 = 8->LeftEye | 2+7 = 9->RightEye
        return randomLayer;
    }
    private float RandomPositionX(){    //return a random Position  -3.3   0     3.3
                                        //                          Left|Center|Right
        if(lastPosX[0]==-1)
            lastPosX[0]=Random.Range(0,3);
        int randomPosX = Random.Range(0,3);
        while(randomPosX==lastPosX[0]||randomPosX==lastPosX[1]){    //if Object pos is same as last one
            randomPosX=Random.Range(0,3);           //return an random int from 0-2
        }                               //3.3-0 =3.3->Right | 3.3-3.3 = 0 ->Center| 3.3-6.6=-3.3 -> Left
        lastPosX[0]=randomPosX;         //update data of lastPosX
        lastPosX[1] = lastPosX[1];
        return 3.3f-randomPosX*3.3f;
    }
}
```

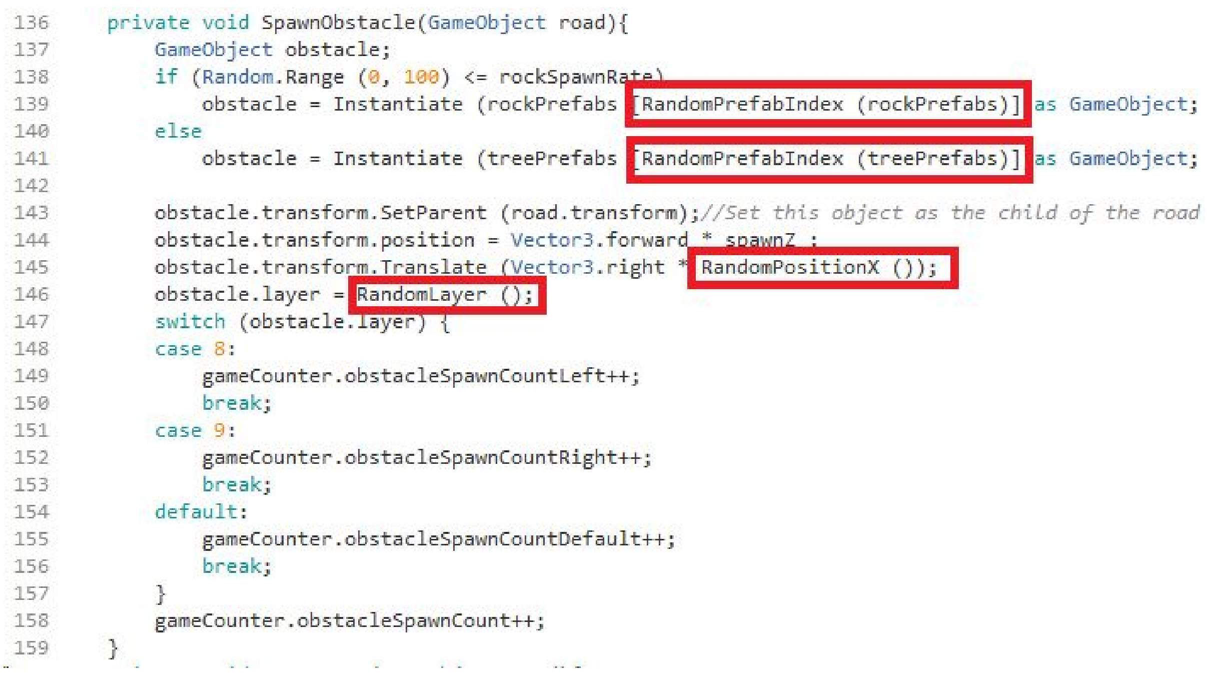
Figure 40 *Some Random functions used to spawn random objects*

Three functions are used to make the spawning system of the game totally random.

RandomPrefabIndex is a function to pick a random game prefabs from the list, it affect which type of trees or rocks or flowers to be spawn, it only affect the appearance of the game object as it has no conflict with the game spawn setting.

RandomLayer is a function to pick and assign a random layer to the game object, only items and obstacles are generated with this function so they are randomly displayed on different eyes or both eyes.

RandomPositionX is a function to generate a random x position of the object to be spawned. Similar to other endless runner games, this function returns the x coordinates of three different X position (Left, Center, Right), so objects are assigned to a random position on the road.



```
private void SpawnObstacle(GameObject road){
    GameObject obstacle;
    if (Random.Range (0, 100) <= rockSpawnRate)
        obstacle = Instantiate (rockPrefabs [RandomPrefabIndex (rockPrefabs)]) as GameObject;
    else
        obstacle = Instantiate (treePrefabs [RandomPrefabIndex (treePrefabs)]) as GameObject;

    obstacle.transform.SetParent (road.transform);//Set this object as the child of the road
    obstacle.transform.position = Vector3.forward * spawnZ ;
    obstacle.transform.Translate (Vector3.right * RandomPositionX ());
    obstacle.layer = RandomLayer ();
    switch (obstacle.layer) {
    case 8:
        gameCounter.obstacleSpawnCountLeft++;
        break;
    case 9:
        gameCounter.obstacleSpawnCountRight++;
        break;
    default:
        gameCounter.obstacleSpawnCountDefault++;
        break;
    }
    gameCounter.obstacleSpawnCount++;
}
```

Figure 41 *SpawnObstacle function in RoadManager*

5.3.3 Game Collision

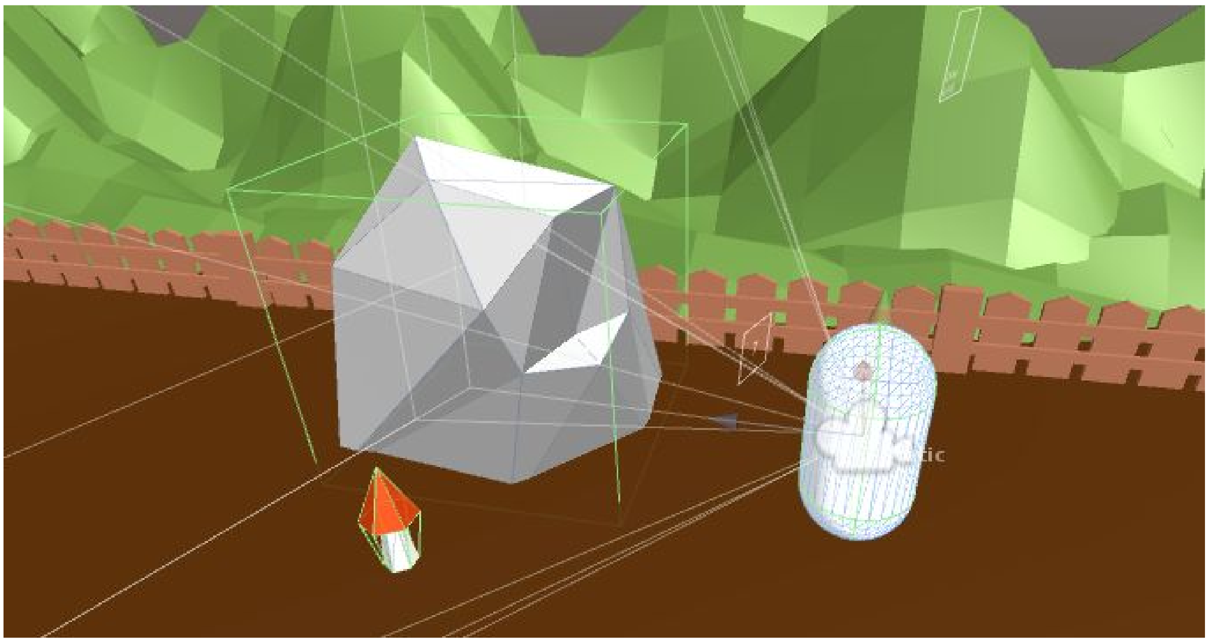
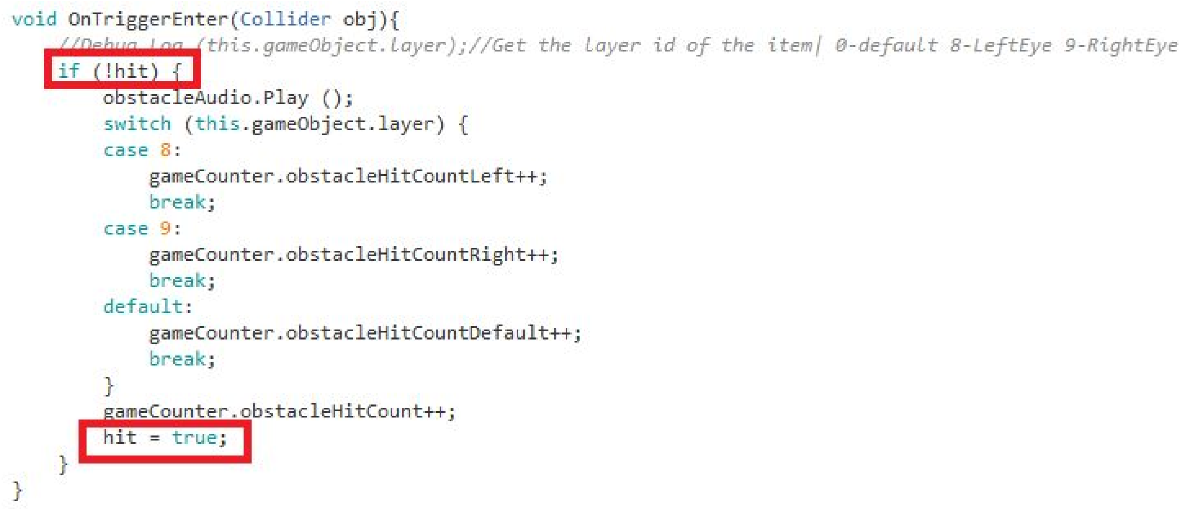


Figure 42 *Collider of game objects*

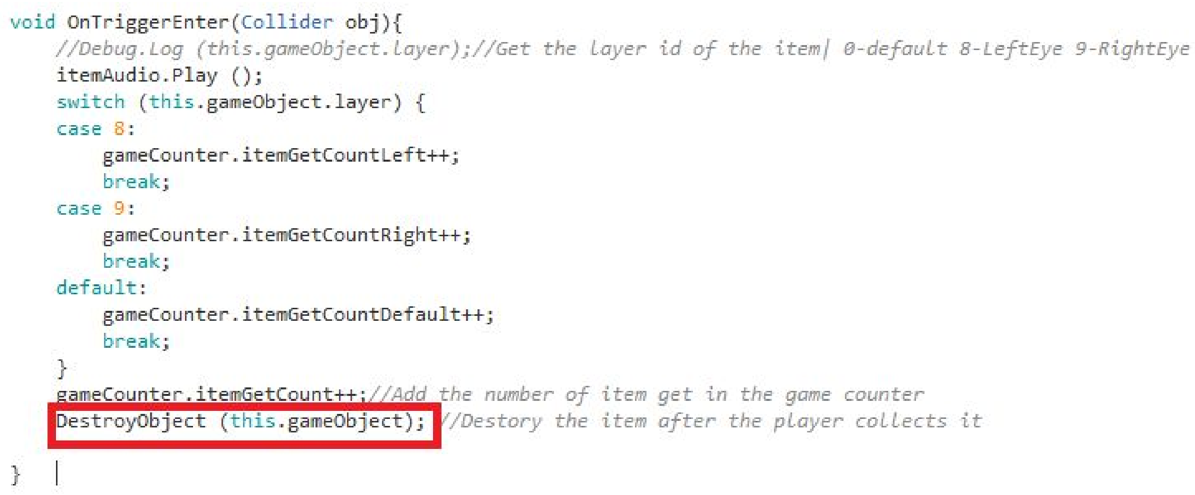
There are three types of test objects in the game: Obstacles, items and color blind triggers. Because of different game objective, the game objects are attached with different scripts to handle the collision events.

In the obstacle collision script, the variable Hit is used to record whether or not this object was already hit. It can prevent multiple count on the collision trigger.



```
void OnTriggerEnter(Collider obj){
    //Debug.Log (this.gameObject.layer);//Get the layer id of the item| 0-default 8-LeftEye 9-RightEye
    if (!hit) {
        obstacleAudio.Play ();
        switch (this.gameObject.layer) {
        case 8:
            gameCounter.obstacleHitCountLeft++;
            break;
        case 9:
            gameCounter.obstacleHitCountRight++;
            break;
        default:
            gameCounter.obstacleHitCountDefault++;
            break;
        }
        gameCounter.obstacleHitCount++;
        hit = true;
    }
}
```

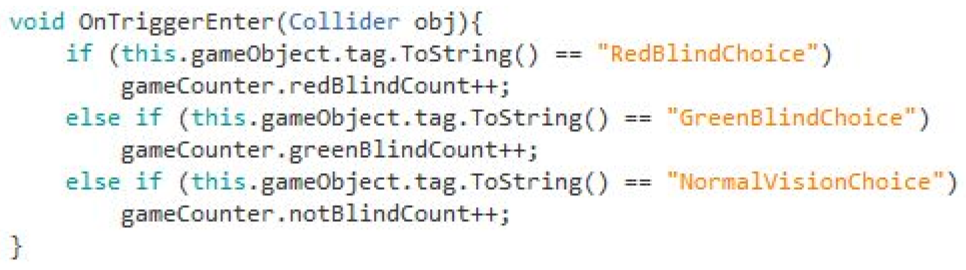
Figure 43 *Collider trigger function for obstacles*



```
void OnTriggerEnter(Collider obj){
    //Debug.Log (this.gameObject.layer);//Get the layer id of the item| 0-default 8-LeftEye 9-RightEye
    itemAudio.Play ();
    switch (this.gameObject.layer) {
    case 8:
        gameCounter.itemGetCountLeft++;
        break;
    case 9:
        gameCounter.itemGetCountRight++;
        break;
    default:
        gameCounter.itemGetCountDefault++;
        break;
    }
    gameCounter.itemGetCount++;//Add the number of item get in the game counter
    DestroyObject (this.gameObject); //Destory the item after the player collects it

}
```

Figure 44 *Collider trigger function for items*

For collectable items, they should be destroyed when the player collect (hit) them. A DestoryObject function is called at the end of the collision trigger function for items.

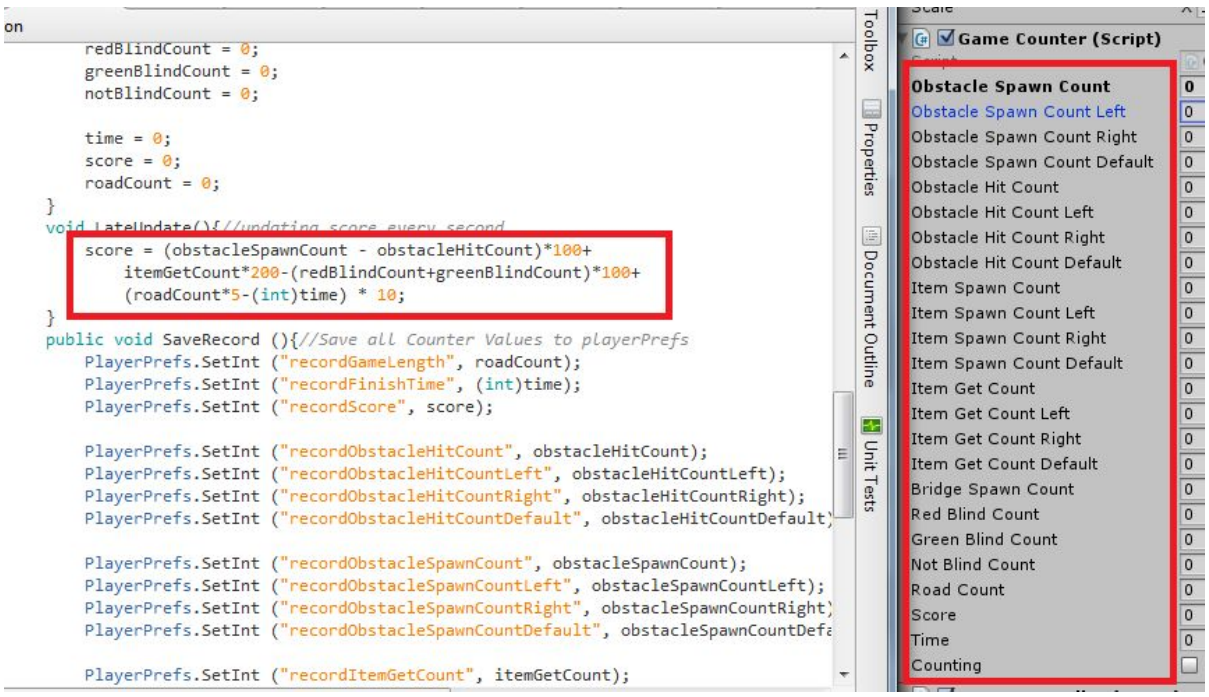


```
void OnTriggerEnter(Collider obj){
    if (this.gameObject.tag.ToString() == "RedBlindChoice")
        gameCounter.redBlindCount++;
    else if (this.gameObject.tag.ToString() == "GreenBlindChoice")
        gameCounter.greenBlindCount++;
    else if (this.gameObject.tag.ToString() == "NormalVisionChoice")
        gameCounter.notBlindCount++;
}
```

Figure 45 *Collider trigger function for color blind triggers*

For colorblind triggers, since they are transparent and not visible to the player, no special function are required.

5.3.4 Game Counter



```
on
        redBlindCount = 0;
        greenBlindCount = 0;
        notBlindCount = 0;

        time = 0;
        score = 0;
        roadCount = 0;
    }
    void LateUpdate(){//updating score every second
        score = (obstacleSpawnCount - obstacleHitCount)*100+
            itemGetCount*200-(redBlindCount+greenBlindCount)*100+
            (roadCount*5-(int)time) * 10;
    }
    public void SaveRecord (){//Save all Counter Values to playerPrefs
        PlayerPrefs.SetInt ("recordGameLength", roadCount);
        PlayerPrefs.SetInt ("recordFinishTime", (int)time);
        PlayerPrefs.SetInt ("recordScore", score);

        PlayerPrefs.SetInt ("recordObstacleHitCount", obstacleHitCount);
        PlayerPrefs.SetInt ("recordObstacleHitCountLeft", obstacleHitCountLeft);
        PlayerPrefs.SetInt ("recordObstacleHitCountRight", obstacleHitCountRight);
        PlayerPrefs.SetInt ("recordObstacleHitCountDefault", obstacleHitCountDefault)

        PlayerPrefs.SetInt ("recordObstacleSpawnCount", obstacleSpawnCount);
        PlayerPrefs.SetInt ("recordObstacleSpawnCountLeft", obstacleSpawnCountLeft);
        PlayerPrefs.SetInt ("recordObstacleSpawnCountRight", obstacleSpawnCountRight)
        PlayerPrefs.SetInt ("recordObstacleSpawnCountDefault", obstacleSpawnCountDefa

        PlayerPrefs.SetInt ("recordItemGetCount", itemGetCount);
```

Game Counter (Script)

Obstacle Spawn Count	0
Obstacle Spawn Count Left	0
Obstacle Spawn Count Right	0
Obstacle Spawn Count Default	0
Obstacle Hit Count	0
Obstacle Hit Count Left	0
Obstacle Hit Count Right	0
Obstacle Hit Count Default	0
Item Spawn Count	0
Item Spawn Count Left	0
Item Spawn Count Right	0
Item Spawn Count Default	0
Item Get Count	0
Item Get Count Left	0
Item Get Count Right	0
Item Get Count Default	0
Bridge Spawn Count	0
Red Blind Count	0
Green Blind Count	0
Not Blind Count	0
Road Count	0
Score	0
Time	0
Counting	☐

Figure 46 *Game Counter script*

The game counter stores a list of local variables in the game. They are the detail information of the test including the layer information. The collision trigger functions mentioned in the previous part of this report has shown how the objects collision increment the value of variables in the counter script. After the game is finished, all the values in this counter is saved in the PlayerPrefs for the record and report generation.

5.3.5 Goal / Endgame

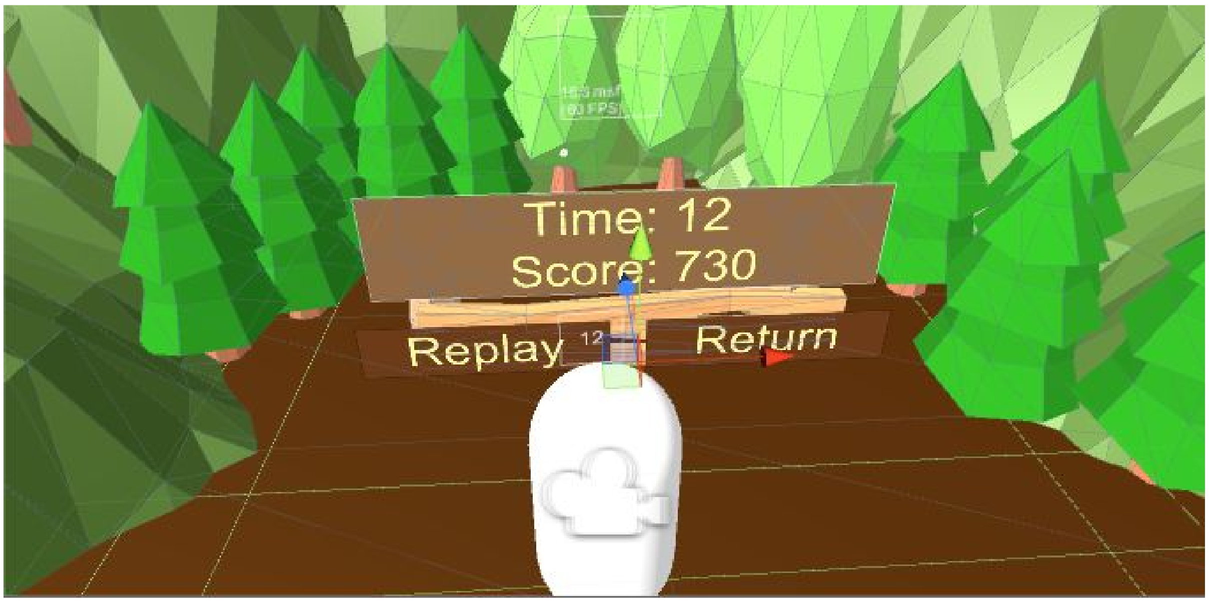
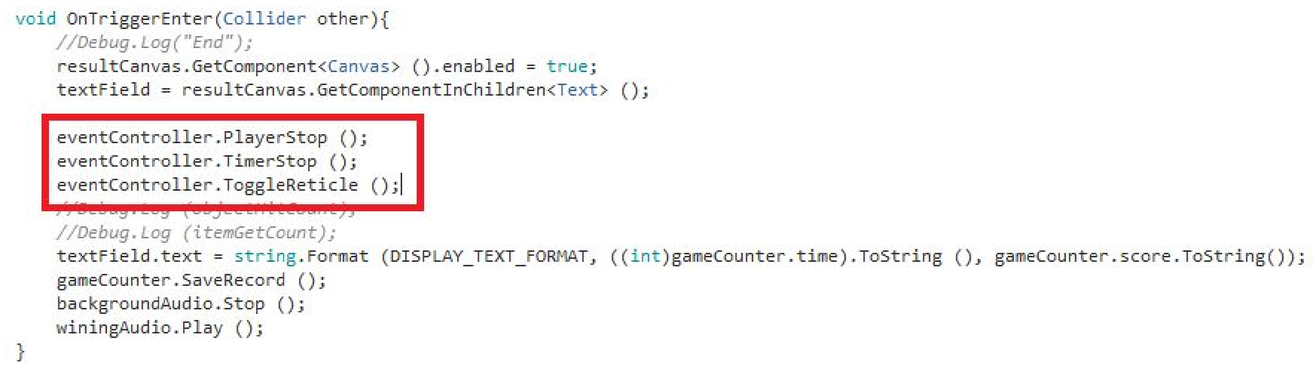


Figure 47 *Goal display menu*

When the player arrive the goal, an invisible box collider will trigger the collision function to perform a number of end game tasks.

The script call the functions in the event controller script. First, it stop the player from moving by setting the Move variable in the VR Autowalk script to false. It stop the timer which count the game time and it reactivate the reticle gaze input for interacting with the end game menu. Player can chose to replay the game or return to the main menu after the game is finished.



```
void OnTriggerEnter(Collider other){
    //Debug.Log("End");
    resultCanvas.GetComponent<Canvas> ().enabled = true;
    textField = resultCanvas.GetComponentInChildren<Text> ();

    eventController.PlayerStop ();
    eventController.TimerStop ();
    eventController.ToggleReticle ();
    //Debug.Log (objectHitCount);
    //Debug.Log (itemGetCount);
    textField.text = string.Format (DISPLAY_TEXT_FORMAT, ((int)gameCounter.time).ToString (), gameCounter.score.ToString());
    gameCounter.SaveRecord ();
    backgroundAudio.Stop ();
    winingAudio.Play ();
}
```

Figure 48 *Collider trigger function for goal trigger*

6. Testing and Findings

Due to time time limitation and a tight schedule, the testing of the product was conducted with a few number of my schoolmates. The tests are performed in different setting for the same person and useful comments were given for the future improvement on the project.

6.1 Normal Vision

For people who are not color blind and do not have binocular vision problem, they should be able to identify the position of the items and obstacles. A number of tests are conducted with different people, a detailed list of test results is included in the appendix.

To summarize the tests result, it is quite positive as the performances of people with normal vision perform are good.

In the first time they play the game without any instruction by looking at the How-To-Play menu, all players are able to pass over 80 percent of the colorblind test object, while most of the players (3 out of 4) are able to achieve over percent in binocular test the first. The only player who can not achieve the 80 percent standard only passed 13 out of 16 test objects.

Based on the above findings, a second testing are conducted on same players. The result shows that the performances of some players have improved, while some remains a similar percentage above 80 percent.

On the other hand, the playing style of the player has also analyzed. The playing habits of players are important as this indicates whether the game is designed in a natural way.

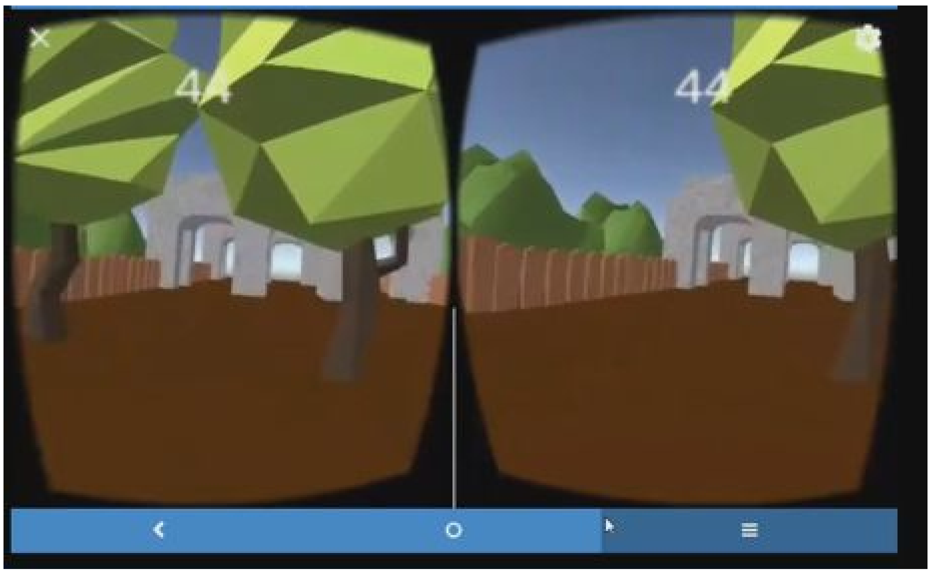


Figure 49 *Screen capture of the game performed by Player A*

One interesting finding is that one player (Player A) tries to pass through two trees in both tests, however, it is possible that the character touches the tree and triggers the obstacle hit collision in the game which affect the result.

The tests are conducted in a default game setting of player speed equals to 3 and game length equals to 10, the spawn setting is also set to the default with 50 percent of color blind test, 25 percent of items and 25 percent of obstacles. The comments from the player about the setting is that the player speed is good, not too fast or too slow, while the game length is also suitable for a vision screening game. The only drawback is that since the objects are generated randomly, it is possible that two consecutive items are spawned at two ends of the road, the player may not be able to move the head quick enough to control the direction. The same problem also applied to the color blind tests when the correct paths of two consecutive bridge are at two ends.

6.2 ColorBlind Simulation

Although no color blind people participate in the test, an alternative test method is used to simulate people with colorblind.

Because of the virtual reality and phone rendering problem, the color blind simulation only works on the computer in non-VR mode.

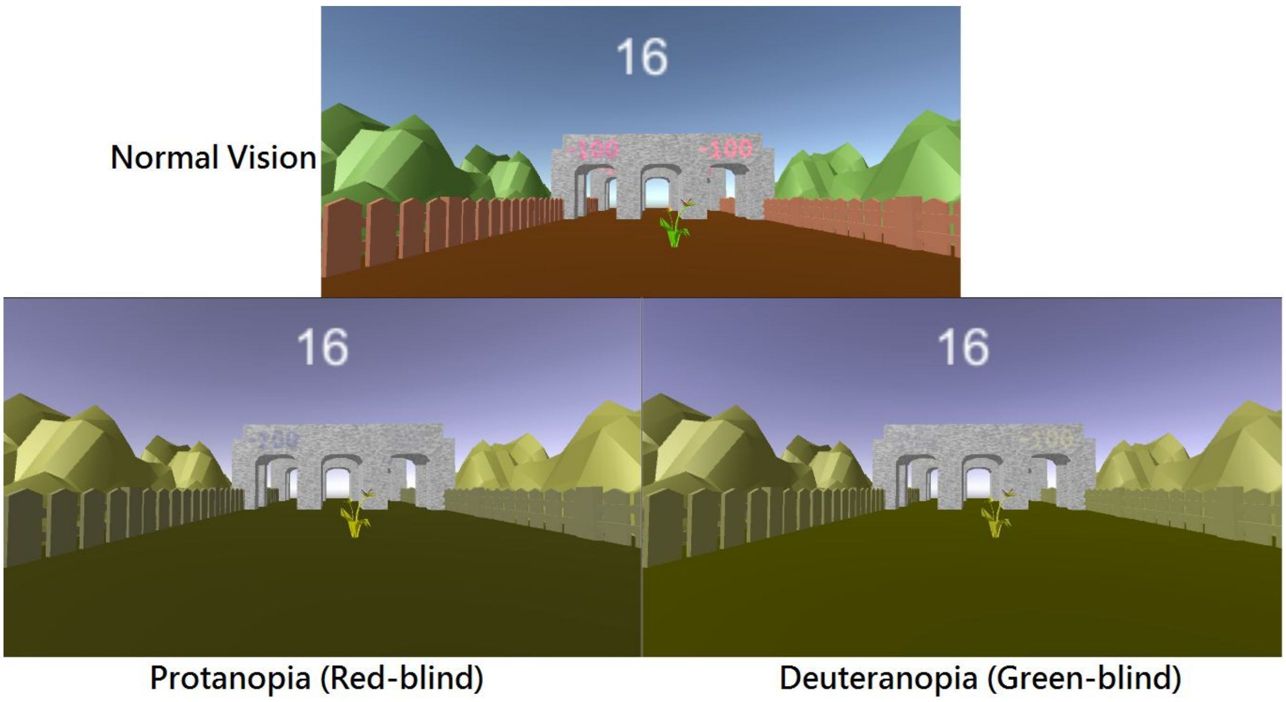


Figure 50 *Screen capture of the game in different vision*

The players reviewed that they are not able to distinguish and normal vision path and the color blind path (according to the color blind mode they played). However, some were still able to identify the color blind object in a very close distance. Since the character keeps moving forward, the player are not able to react quick enough to avoid the incorrect path in such close distance, they have to guess the correct path with 50 percent of chance.

The main drawback from the comments is that since there is a 50 percent of chance that color blind player can get the correct answer, it is possible that they can pass the test with a percentage higher than 80 percent. This can be adjusted by increasing the game length and changing the spawn setting to color-blind test only, multiple number of tests can also be conducted in order to carefully identify whether or not the player are colorblind or just not good at playing the game.

6.3 Diplopia Simulation

Although people who suffers from binocular vision problem participate in the test, an alternative test method is used to simulate people with the problem.

By adjusting the position and rotation of the camera of one eye, a diplopia vision is simulated in the virtual reality environment.

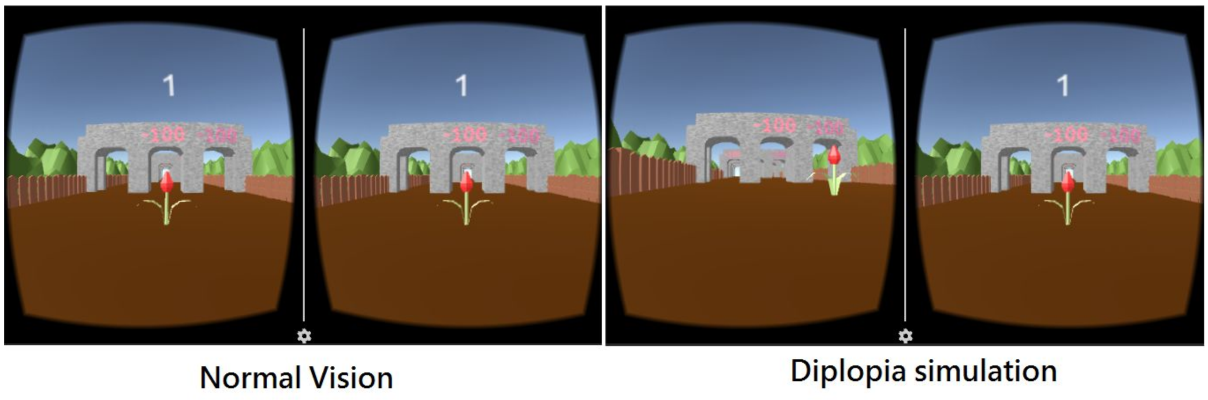
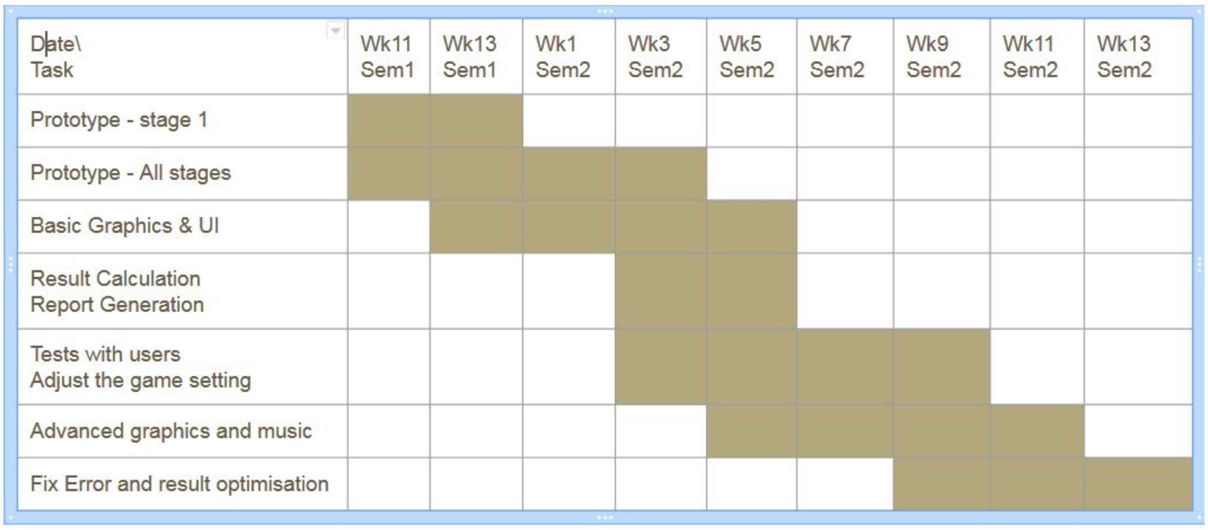


Figure 51 *VR Screen capture of the game in different vision*

As the center focus of two eyes are different, it is difficult to combine contents from different eyes to one single image. The results and comments revealed that it is totally not possible to avoid obstacles and collect items and even pass the color blind test as the players lost focus and see two objects instead of one.

7. Project Schedule

The project is divided in smaller parts for a better time management and schedule planning.



Date\ Task	Wk11 Sem1	Wk13 Sem1	Wk1 Sem2	Wk3 Sem2	Wk5 Sem2	Wk7 Sem2	Wk9 Sem2	Wk11 Sem2	Wk13 Sem2
Prototype - stage 1	▨	▨							
Prototype - All stages	▨	▨	▨	▨					
Basic Graphics & UI		▨	▨	▨	▨				
Result Calculation Report Generation				▨	▨				
Tests with users Adjust the game setting				▨	▨	▨	▨		
Advanced graphics and music					▨	▨	▨	▨	
Fix Error and result optimisation							▨	▨	▨

Figure 52 *Schedule planning of the project*

The above graph show the work order of different tasks, with important parts put in the front and the extra functions in the back, the project schedule is more flexible when encountering obstacles that are difficult and take time to solve.

Prototype for the main game was finished first for interaction testing as this is the most vital part of the project. Further improvement in graphics and game components are made after the result calculation and report generation part is completed. Tests with users are required to adjust the game setting and algorithm in order to optimize the result in terms of sensitivity and accuracy.

8. Future Work

8.1 Limitation

Due to the time and resources limitation, this project has a number of constraint.

The number of test from players conducted is very small, and it only cover people with normal vision only. No actual color blind and diplopia/strabismus patients are tested in this project. The result can not fully support the functionality and accuracy of this vision screening game.

The game user interface in the menu use the basic default spite which is very simple, the in-game objects are all low-poly and very simple, it may not look attractive to the children.

The game record part only record the most recent gameplay, it can not store multiple records for further comparison. Users have to record the result manually by hand if they want to perform a new test while keeping the old record.

8.2 Improvement

There are some improvements on the project that can be made in the future.

First, more test results should be collected from different users including those who are colorblind and have binocular vision problems. The data is important for optimzing the accuracy of the test. Minor adjustments could be made based on the comments and test results from people in different ages. The game should be able to provide a recommended setting for different users with different needs.

Second, the game record and report generation should be improved so it is more clear about different records. For example, the report should be able to output to a database with all the detail game counter values, so the record can be used as a reference for the future comparison or for the doctor.

Lastly, the game theme should be able to change to make the game more attractive to children, for example, a space or pirate theme for boys and a fantasy theme for girls as this game was designed for children.

9. Conclusion

This project has provided a great opportunity for me to work on a large project on my own. It has strengthen my programming skill with Unity and my project management skill.

As the objective of this project is not only creating a game, but more importantly, a game to aid vision screening. It reminds me of the service learning subject form the computing department I studied, which we use our knowledge to help others. I feel like I am doing something important as this project aims to help others.

Although the time and resources are limited, I am satisfy with the final product as I put a lot of time and effort into the project. The outcome of this project may be limited by my ability, but I hope that in the future a similar idea of a vision screening virtuality reality game will be developed as one of the vision screening standard with accurate results.

10. References

[1] "Virtual reality," in *Wikipedia*, Wikimedia Foundation, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Virtual_reality. Accessed: Nov. 1, 2016.

[2] "The difference between a vision screening and a comprehensive eye examination," 2016. [Online]. Available: http://www.visionaware.org/info/your-eye-condition/eye-health/eye-examination/125. Accessed: Nov. 3, 2016.

[3] D. S. Fan, C. Lai, H. H. Lau, E. Y. Cheung, and D. S. Lam, "Change in vision disorders among Hong Kong preschoolers in 10 years," *Clinical & Experimental Ophthalmology*, vol. 39, no. 5, pp. 398–403, Feb. 2011.

[4] "Pre-School vision screening," in *Family Health Services - Department of Health - The Government of the Hong Kong Special Administrative Region*, 2016. [Online]. Available: http://www.fhs.gov.hk/english/health_info/child/14812.html. Accessed: Dec. 2, 2016.

[5] "About student health service centre / special assessment centre," in *Student health service - Department of Health - The Government of the Hong Kong Special Administrative Region*, 2006. [Online]. Available:http://www.studenthealth.gov.hk/english/aboutus/aboutus_shscsac/aboutus_shscsac.html. Accessed: Dec. 14, 2016.

[6] R. H. Trivedi, M. E. Wilson, M. M. Peterseim, K. B. Cole, and R. G. W. Teed, "A pilot study evaluating the use of EyeSpy video game software to perform vision screening in school-aged children," *Journal of American Association for Pediatric Ophthalmology and Strabismus*, vol. 14, no. 4, pp. 311–316, Aug. 2010.

[7] LC. Nguyen, E. Y.-L. Do, A. Chia, Y. Wang, and H. B.-L. Duh, "DoDo game, a color vision deficiency screening test for young children," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2289–2292, Apr. 2014.

[8] LC. Nguyen, W. Lu, E. Y.-L. Do, A. Chia, and Y. Wang, "Using digital game as clinical screening test to detect color deficiency in young children," *Proceeding IDC '14 Proceedings of the 2014 conference on Interaction design and children*, pp. 337–340, Jun. 2014.

[9] D. J. Ma, H. K. Yang, and J.-M. Hwang, "Reliability and validity of an automated computerized visual Acuity and Stereoacuity test in children using an interactive video game," *American Journal of Ophthalmology*, vol. 156, no. 1, pp. 195–201.e1, Jul. 2013.

[10] "Smartphone Autorefractor," in *EyeNetra*, EyeNetra Store, 2016. [Online]. Available: https://store.eyenetra.com/products/netra-autorefractor-smartphone. Accessed: 2016.

[11] "Move Beyond the Eye Patch,"in *Vivid Vision*. [Online]. Available: https://www.seevividly.com/. [Accessed: 16-M-2017].

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

[13]P. Lempert, "A cost-benefit analysis of vision-screening methods for preschoolers and school-age children," *Journal of American Association for Pediatric Ophthalmology and Strabismus*, vol. 8, no. 1, p. 74, Feb. 2004.

[14] S. P. Donahue, B. Arthur, D. E. Neely, R. W. Arnold, D. Silbert, and J. B. Ruben, "Guidelines for automated preschool vision screening: A 10-year, evidence-based update," *Journal of American Association for Pediatric Ophthalmology and Strabismus*, vol. 17, no. 1, pp. 4–8, Feb. 2013.

[15] R. Tirendi, "A Video Game that Detects Vision Disorders," in *Schoolhealth*, 2013. [Online]. Available: https://www.schoolhealth.com/blog/a-video-game-that-detects-vision-disorders/. [Accessed: 3-Nov-2016]

[16] S. Yoo and C. Parker, "Controller-less interaction methods for Google cardboard," *SUI '15 Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, p. 127, Aug. 2015.

[17] B. T. Ltd., "Google Cardboard V1 and V2 Comparison," in *vr.brizTech.co.uk*. [Online]. Available: https://vr.briztech.co.uk/cardboard-comparison.html. [Accessed: 17-Nov-2016].

Appendix

Tools and Sofware used

Unity 5.3.3 f1
Google Cardboard SDK 1.0.0
CVD Test Generator : http://www.notquite.me/CBTest

Unity Assets used

Free Low Poly Pack :
https://www.assetstore.unity3d.com/en/#%21/content/65375
Low Poly: Free Pack :
https://www.assetstore.unity3d.com/en/#%21/content/58821
Low Poly Nature Pack (Lite) :
https://www.assetstore.unity3d.com/en/#%21/content/40444
Low poly styled trees:
https://www.assetstore.unity3d.com/en/#%21/content/43103
Low poly styled rocks:
https://www.assetstore.unity3d.com/en/#%21/content/43486
Medieval Lowpoly Pack:
https://www.assetstore.unity3d.com/en/#%21/content/24857
Color Blindness Simulator for Unity:
https://www.assetstore.unity3d.com/en/#%21/content/19039

Sample Test Results

Test Player A (No Color Blind, No stramibus)

04/06/2017 16:24:44 Play For the first time
Setting: Default (50%ColorBlind Test, 25% Items, 25% Obstacles)
Speed:3 Length: 30 Time: 73
Score: 3470

	Obstacle Pass	Item collect
Left Eye	4/6	2/2
Right Eye	4/4	1/1
Both Eye	5/5	4/5
Total	13/15	7/8

Red Blind	0/15
Green Blind	0/15
Normal Vision	15/15

04/06/2017 17:42:31 Play For the second time
Setting: Default (50%ColorBlind Test, 25% Items, 25% Obstacles)
Speed:3 Length: 30 Time: 70
Score: 3200

	Obstacle Pass	Item collect
Left Eye	4/5	1/2
Right Eye	8/8	2/2
Both Eye	2/2	2/2
Total	14/15	5/6

Red Blind	0/15
Green Blind	0/15
Normal Vision	15/15

Test Player B (No Color Blind, No stramibus)

04/06/2017 18:15:30 Play For the first time
Setting: Default (50% ColorBlind Test, 25% Items, 25% Obstacles)
Speed:3   Length: 30 Time: 70
Score: 2600

	Obstacle Pass	Item collect
Left Eye	3/3	2/3
Right Eye	1/2	4/4
Both Eye	4/4	0/1
Total	8/9	6/8

Red Blind	2/20
Green Blind	0/20
Normal Vision	18/20

Test Player C (No Color Blind, No stramibus)

04/06/2017 19:00:27 Play For the first time
Setting: Default (50% ColorBlind Test, 25% Items, 25% Obstacles)
Speed:3 Length: 30 Time: 71
Score: 4090

	Obstacle Pass	Item collect
Left Eye	5/5	2/3
Right Eye	5/5	4/4
Both Eye	5/5	3/3
Total	15/15	9/10

Red Blind	0/10
Green Blind	0/10
Normal Vision	10/10

04/06/2017 19:44:27 Play For the second time
Setting: Default (50% ColorBlind Test, 25% Items, 25% Obstacles)
Speed:3 Length: 30 Time: 71
Score: 4090

	Obstacle Pass	Item collect
Left Eye	2/3	5/5
Right Eye	4/4	3/3
Both Eye	7/7	4/5
Total	13/14	12/13

Red Blind	0/10
Green Blind	0/10
Normal Vision	9/9

Test Player D (No Color Blind, No stramibus)

04/06/2017 19:53:54 Play For the first time
Setting: Default (50% ColorBlind Test, 25% Items, 25% Obstacles)
Speed:3   Length: 30 Time: 69
Score: 5890

	Obstacle Pass	Item collect
Left Eye	5/6	1/1
Right Eye	1/1	2/4
Both Eye	2/2	3/4
Total	8/9	6/9

Red Blind	0/10
Green Blind	3/16
Normal Vision	13/16

04/06/2017 20:03:23 Play For the Second time
Setting: Default (50% ColorBlind Test, 25% Items, 25% Obstacles)
Speed:3   Length: 30 Time: 72
Score: 4150

	Obstacle Pass	Item collect
Left Eye	3/3	6/6
Right Eye	3/3	4/4
Both Eye	5/5	3/3
Total	11/11	13/13

Red Blind	1/13
Green Blind	2/13
Normal Vision	10/13