Cyber Invaders

Group Seven's User Manual

IBM Cybersecurity Virtual Reality Game

Prepared By:
Grace West (lxvl71) - Meenal Prajapati (sdth32) - Mithara De Alwis (brjv82)
Fatima Jabeen (fgcv57) - Vilde Ringstad (hpxn64)

Contents

General Information

Versions of unity and C#

Unity Version	C# Version	C# Compiler	Documentation links	Date
Unity 2020.3.25	C# 8.0	Roslyn	https://docs.unity3d.com/Manual/index.html https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-8	13/03/2022

User Manual Version

Version	Date	Content/Changes
1.0.0	13/03/2022	• **General information** • **User journey** • **Backend design and implementation** • **Technical section**

Introduction

This user manual will explain and support our IBM virtual reality game, based on the IBM 'Cybersecurity fundamentals' badge and is implemented using IBM's Watson speech-to-text. The purpose of the game is to provide IBM's customers with an enjoyable and interactive way to learn about the contents of the badge. In the game the user is spawned in a spaceship where they will be faced with various threats in the form of alien attacks, which they will have to destroy using the correct gun. Points can be gained by destroying these threats which can be used to purchase team members from the in-game shop. The alien attacks represent real life cybersecurity threats, the guns represent various cybersecurity defences and the team mates represent real world professions available in the cybersecurity field. All of the information for the team members, threats and guns was extracted from the badge learning resource, as requested by IBM.

Our game will be implemented using Unity and MySQL for the back end and played using the Oculus Quest 2 headset and Oculus Touch controllers.

This manual will cover the user playthrough in the game, how the backend is designed and implemented, what each of the C# scripts in the game do and how they are structured, information about the various programs and languages used to create our game, how to use the equipment required to run the game, future developments and current limitations.

Health and Safety

Our project is heavily based on Virtual Reality hence why the Oculus headset is at the front end of our system.Therefore there are some key safety notices to be aware of.

Boundaries:

Whenever the headset is in use there should be a Guardian boundary set up that marks the edges of the play area. The game should be played in a clean and free zone where there should not be any objects around or on the floor of the play area which could potentially harm the player. There should always be another person standing by to ensure that the person wearing the headset is safe since they are blindfolded to the real world and unaware of their surroundings.

Motion sickness:

If a player is prone to motion sickness it is advised that they do not play the game since the headset contains moving around and turning from front to back sometimes. This constant movement may cause nausea and in serious cases vomiting. It is advised that the player takes regular breaks from the headset, around 15 minutes. Constant play of the game may lead to nausea.

Lenses Should be Away from Direct Sunlight:

The headset should not be used outside and should be used away from any windows since sunlight can cause lasting damage to the lenses.

COVID-19 and Contagious Conditions:

Due to the coronavirus pandemic it is essential to clean the headset and the controllers thoroughly with non-alcoholic antibacterial wipes and a dry microfiber cloth for the lenses, to prevent any spreading. Users with the headset should ensure they have santised their hands and have no symptoms of Covid-19 when using the headset. Furthermore, to avoid the

transmission of contagious conditions do not share the headset with people with contagious conditions, infections or diseases.

Use of controllers:

The VR game involves using the controllers to detect and shoot the threat. The player using the controllers should be aware they are in a ‘game’ and be sensible with the controllers to prevent dropping or accidentally hitting the person standing by.

Use of headset:

When a player is using the headset they should not be handling sharp or dangerous objects. Furthermore, the headset should never be worn in situations that require attention such as driving, cycling or running or if they are intoxicated. The headset should be level and sit comfortably on the user’s head so that a single, clear image is seen. The user should read the warning screen before the game starts if they are trying the game for the first time.

Sound:

Sound is used in the VR game therefore the sound in the headset should not be at high volume so the user can maintain awareness of their surroundings and also reduce the risk of hearing damage. It is important to ensure that the user is in a quiet area since background noise can make sounds seem quieter than they actually are.

Electrical:

To reduce the risk of the user suffering an electric shock ensure the product is not used if any cable is damaged or any wires are exposed. Ensure the device is not used if any part is broken or damaged.

If the user has any of these following symptoms: seizures, loss of awareness, eye strain, blurred , double vision, dizziness, impaired hand-eye coordination, excessive sweating; increased salivation, nausea or any symptoms similar to motion sickness then immediately discontinue the player from using the headset. We also advise young children, pregnant women and elderly persons to refrain from playing the game to prevent any harmful situations.

Game Design

The look of the game is inspired by the ‘Star Trek Bridge Commander’ game. This is at the request of our contact at IBM. Visuals for this game can be seen here: https://www.gog.com/en/game/star_trek_bridge_commander.

Features which we implemented that were inspired by the Star Trek game:

- ➔ The spaceship and threat themes in the game. See figures 1 and 2 for comparisons.
- ➔ The large window at the front of the ship which is where threats can be seen and destroyed from. See figures 1 and 2 for comparisons.

Figure 1: Star Trek bridge command scene taken from gameplay. The user can be seen attacking a asteroid in space from the ship's game window.

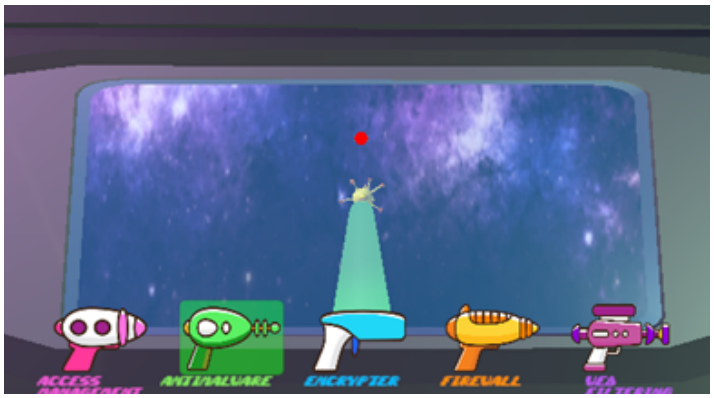


Figure 2: Cyber Invaders ship scene. The user can be seen here shooting at a threat from the ship's game window which looks out onto space.

- ➔ The idea of having multiple team members present in the game. See figures 3 and 4 for comparison.

Figure 3: Star Trek bridge command scene taken from gameplay. It is shown that there are multiple team members.

Figure 4: Cyber Invaders ship scene. In this scene the player can be seen looking at a fellow team member.

It is a first person perspective shooter game, see figures 5, 6 and 7 for more details. Figure 7 shows what the first person player actually looks like in the game. The user will never be able to see his full body as they will be playing as him but if they look down or to the sides they will be able to see his legs, feet and arms. Ideally we would have a better animation for our player character, however this was not possible in this version of the implementation as better character bodymeshes requires a payment being made and we did not have a budget for that.

Figure 5: This image shows the first person view of the Star Trek Bridge Commander game. The player is seen looking at a fellow team mate.

Figure 6: This image shows the first person view in Cyber Invaders of part of the ship.

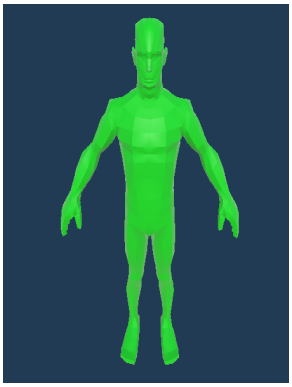


Figure 7: This image shows exactly what the player actually looks like when looked at third person perspective.

User journey

This section will explain the step by step walk through of the game as the user experiences the game and explains how the user plays the game.

Game Warnings

On entering the game the user is met with full screen game warnings, which states any safety warnings the user should be aware of.
The user can click on the NEXT button to enter the main game.

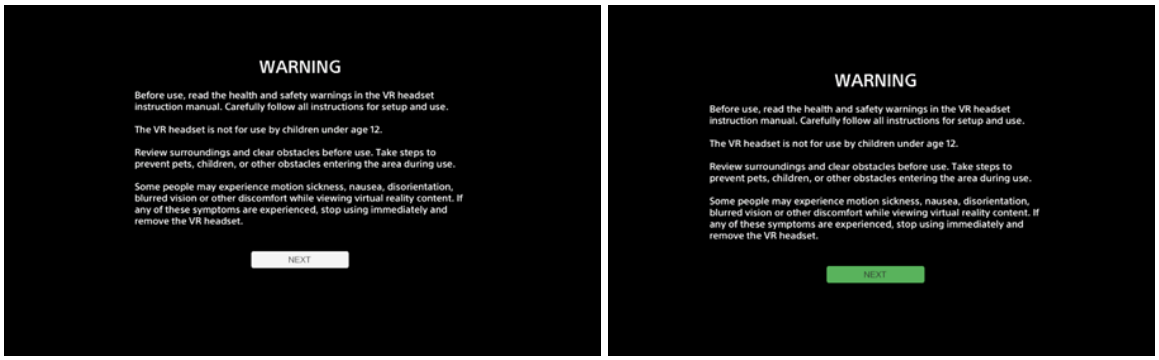


Figure 8: Shows the warning message, when the user hovers over the next button it turns green.

Home page

After the Warnings page, the user is taken to the game homepage which has the start button. The homepage shows the game's logo and a start button (the homepage design is further discussed in future developments). The user can click on the START button to enter the main game.

Figure 9: Shows the homepage with the game logo, when the user hovers over start it turns green.

Main Game

Once entering the game the user finds themselves in the control room of a spaceship. The user can walk around the room of the spaceship; this is reflected by walking in the real

space within the limits set by the user itself. The user starts the game with their gun set to access management (highlighted in green).

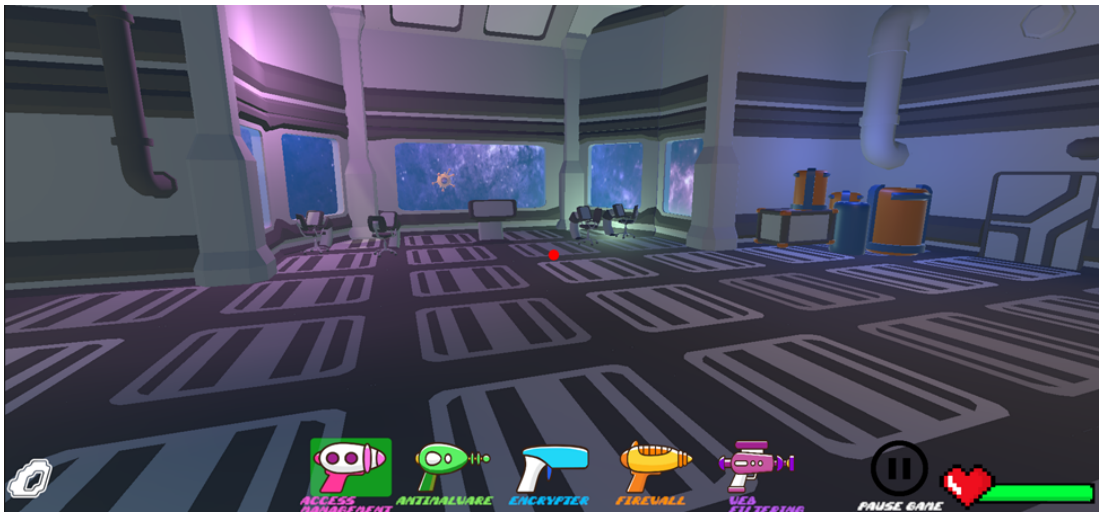
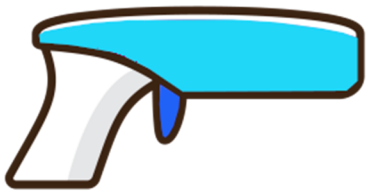
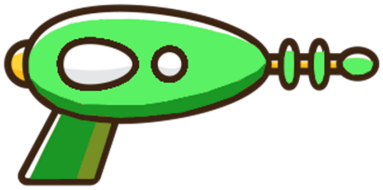
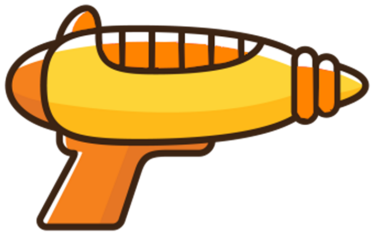
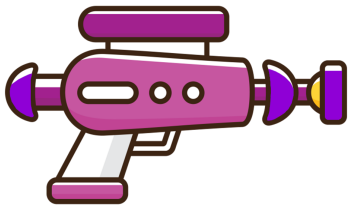


Figure 10: Entering the game
Going across the bottom of the screen: The number in the left hand corner keeps track of points the user gains in the game. The (green) highlight shows what option of each gun or the pause game icon the user has currently chosen. The user can pause the game (stopping incoming threats) with the pause button. The health bar to the right keeps track of the ship's health.
The red dot keeps place of the middle of the screen

The aim of the game is to defeat incoming cybersecurity threats with the various cybersecurity defences (represented by guns) without destroying the ship in the process.
The ship loses health when the threats come into contact with the ship, this is indicated by the ship's heath bar being reduced and the threat disappearing. If the ship's health reaches zero the ship has been destroyed by the cybersecurity threats, the game ends and the user has lost the game.

Figure 11: the end screen that the user sees when they lose the game, after 10 seconds the game closes.

The user has access to five guns which can be swapped by stating the name of the desired gun to the VR headset. Each gun represents a cyber security defence. The five defences (guns) can be seen at the bottom of the screen and are as followed; firewall, access management, web filtering, anti-malware and encrypter. The user can shoot by pressing the shoot button on the Oculus Touch hand held device.

Gun Name	Image
Encrypter	
Anti-Malware	
Firewall	
Access Management	
Web Filtering	

Threats appear in the space outside the spaceship and approach towards the spaceship. Each threat can be defeated with the correct corresponding gun being selected, and shooting the threat successfully three times with that gun. If the user manages to defeat it, the threat will explode and an explosion sound effect will be heard.

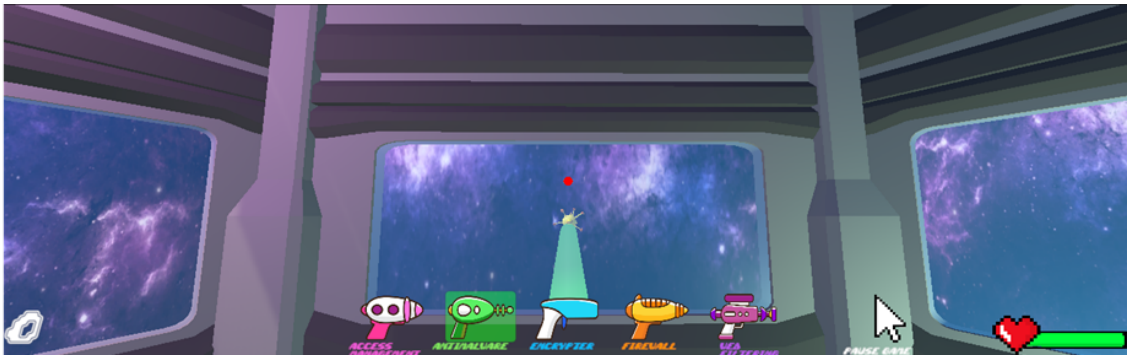


Figure 12: shows the gun being shot

Figure 13: shows the explosion, and the score has been increased

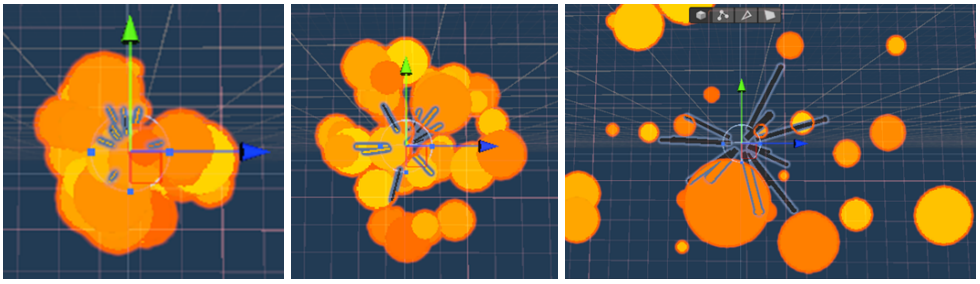
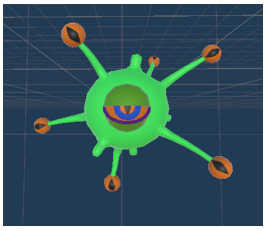
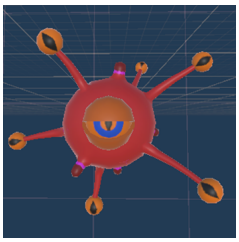
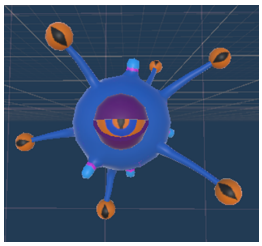
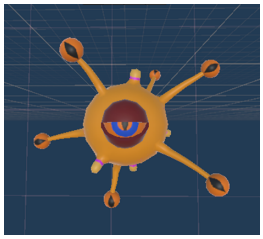
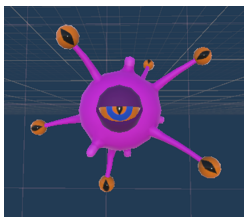
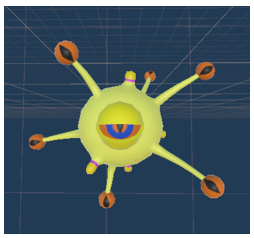


Figure 14: shows the explosion at various stages

Threat	Appearance	Which gun will destroy this threat?

Denial of Service Attack		**Firewall**
Hacker		**Access Management**
Malware attack		**AntiMalware**
Man in the middle attack		**Encrypter**
Phishing attack		**WebFiltering**

SQL injection attack		**Firewall**

The user earns points by defeating the various threats, these points are then updated in the bottom left hand corner (as seen in between figure 12 and 13).

The user can use points to buy cybersecurity professionals, team members, which will help them in the game. Each team member gives the user different power ups, this will be further explained in the shop and team member sections.

The user can pause the game by saying the command “pause”, which in turn will highlight the pause icon on the bottom of the screen and stop the threats from moving, to unpause the user can say the command “play” to get back into play mode where the threats resume their movements. This gives the user enough time to use their points to buy team members in the on board shop.

The game will have 6 waves of threats, each wave incorporating one more threat than the last. Each wave of threats will initially announce the new threat and how the user can defeat it.

Back Wall including Shop

If the player turns 180 degrees from the ship's front window is the back wall of the ship. The back wall is an interactive on-board display which contains a home page, shop page, manual page and warnings page.
The user can swap between these pages with the tabs on the left hand side while the right hand side will display the relevant information which will help them throughout the game.

Home page: The title screen
Shop page: This is where the user can buy team members with points they gain in the game.
Manual page: This has information on the different threats, team members and guns which will be loaded from the SQL database connected to the system.
Warnings page: has information on the game warnings the user should be aware of whilst playing the game.

Home Page

Figure 15: Screenshot of the Home Page of the Back Wall with labelled features

1. Home Tab - current tab
2. Manual Tab
3. Shop Tab
4. Warnings Tab
5. The Home Page with the Logo
6. The Pause Button, which is recommended to have been pressed before navigating to the Back Wall so that the user doesn't lose progress.

Manual Page

Figure 16: Screenshot of the Manual Page of the Back Wall with labelled features

1. Manual Tab - current tab
2. The Manual Page with the instructions on how to play the game
3. Scroll down to be able to read all the instructions and the information on threats and game features

Shop Page

Figure 17: Screenshot of the Shop Page of the Back Wall with labelled features

1. Button to buy the relevant team member with points
2. Shop Tab - current tab
3. Other team members will be added here
4. The Shop Page

Warnings Page

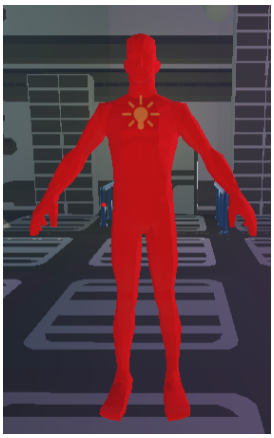
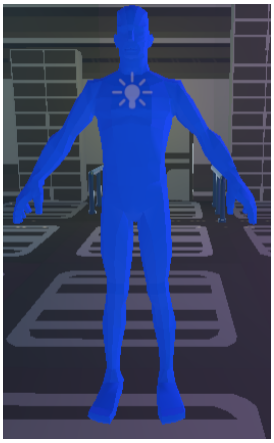
Figure 18: Screenshot of the Warnings Page of the Back Wall with labelled features

1. Warnings Tab - current tab
2. The Warnings Page which displays some warnings for the user to be aware of during the use of the game
3. The warnings which can be read at any time after starting the game

Team members

Team members are a central part of the game as they are how the users are educated on what professions are available in the cybersecurity field which are discussed in the IBM badge. Team members can be bought using points gained by killing threats from the in-game shop. Each team member has a different job and aids the user in different ways. The team members each appear on the ship after they are bought and follow the users player around. The name, appearance and description of each available team member can be seen in the table below.

Team Member	Appearance	Description

Ship Engineer		Represents a security administrator in real life. In the game, the ship engineer will come on and repair the ship which will make it harder for threats to destroy the ship.
Assistant shooter		Represents an incident responder in real life. In the game, the assistant shooter will come on and be a second shooter along with the player.
Threat Detector		Represents a SOC-analyst in real life. In the game, the threat detector will come on and identify and report incoming threats to the player before they can be seen by the player. This will be done using game voice.

Space Expert		Represents a security consultant in real life. In the game, the space expert will come on and tell the player what kind of threat they are dealing with and what type of gun is best suited to kill the threat without the user needing to pause the game to read the onboard manual. This will be done using in-game voice.
Ship enhancer		Represents a Pen tester in real life. In the game, the ship enhancer will come on and add extra strength to the ship which will mean that the ship will have more health that it did not have to begin with. It will also repair any current damage to the ship's health. Without the shield engineer, this role in the game is somewhat redundant.
Cryptographer		Represents a Cryptographer in real-life. In the game the cryptographer will make it harder for threats to identify your exact position so shooting threats will have less accuracy in their shots so fewer hit you.

Win End Screen

Once the user has completed all six waves the game ends and the user has won.

The game shows an end page with a congratulatory message.

Figure 19: the end screen that the user sees when they win the game, after 10 seconds the game closes.

Backend design and implementation

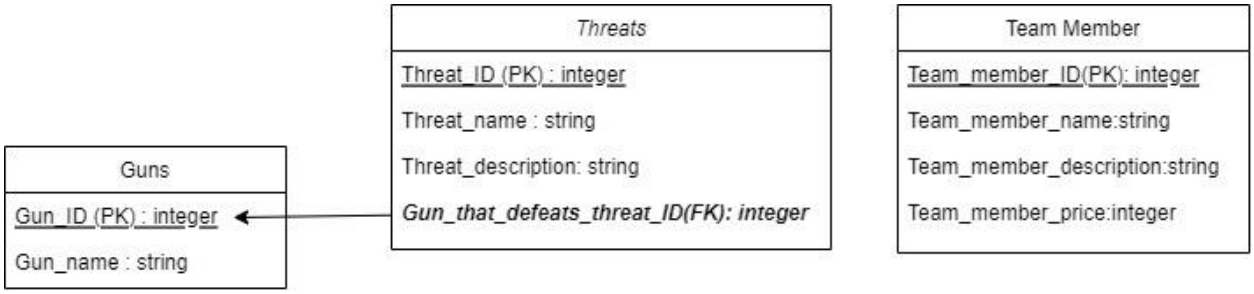
Our database system has been created and implemented using SQLite version 3.38.0 and will be connected to our game using C#. The version of C# used for connecting the database is the same as the version used for the other scripts in the game (C# 8.0). The documentation for the SQLite version is linked here: https://www.sqlite.org/releaselog/3_38_0.html .

Our database is very simple because it only stores information that is to be used in the back wall. This includes information about the team members, about the threats and about the guns which can be found in the manual section. Further, it will be used to display the correct costs for each of the team members. Although it may seem redundant it means that all the information needed for the back wall is stored in one place as opposed to multiple files which will cause repeated data which will lead to inefficient use of storage. Moreover, in the future if the game is extended to contain more team members, guns or threats it will be easy to update the database to contain this new information.

Our database consists of four different tables: Team members, weapons, threats and guns. The team members table stores information about each of the different team members such as their unique ID, name, description and price. This data will be extracted from the database and displayed in the in-game user manual, so that the user knows the name and functionality of the team member and how many points they will cost the user when bought from the in-game shop. The threats table contains the unique ID, name, description and the ID of the gun which is used to destroy that specific threat. This gun ID is a foreign key in the threats table and is the primary key in the ‘guns’ table. The gun ID and description and name is also used in the in-game user manual to tell the user the purpose and name of each gun and which threat it should be used for. Finally, the guns table contains the gun ID and gun name. The gun name will be displayed with the threat that it is used to destroy in the in-game user manual.

Database Structure

This class diagram shows the relationships between the various tables in our database. The arrows represent a link between a primary and foreign key in two different tables.



Technical section

Comparison to requirement specification

In order to check that the system meets the intended requirements, we have created tables evaluating each requirement set out in the initial requirement specification below and how each was met and/or changed.

Functional requirements

	Requirement	Does the solution meet the initial requirement?	How and why was it changed?
2.1.1	Introductory stage	Yes as there is a home page and warning screen before the game starts as discussed in the original requirement.	The requirement has slightly changed as originally we planned to do the game home screen before the warning screen. In our implementation we have put the warning screen before the game home screen which is better because the user should be aware of the warnings as

			soon as they start to use the VR headset as there are warnings associated with use of the actual headset itself.
2.1.2	Game Instructions	Yes as the game will have step-by-step instructions available for the user to read on the back wall at any point in the game.	The requirement has slightly changed because the tutorial stage will be step-by-step instructions that the user can read in their own time available on the back wall instead of a video as originally planned. This is better because it means the user can access the instructions at any point in the game and they can only read the parts they need to read instead of having to watch a whole video to get help.
2.1.3	Speech-to-text	Yes, the system will be able to recognise names of guns in order to swap them. It will also recognise pause and play so the user can pause and play the game.	N/A
2.1.4	Attacking threats	Yes, as the user is able to shoot the threats which will cause the threats health to decrease and will eventually destroy the threat.	The requirement has been slightly altered because in the original requirement it required that speech to text should be used to name the correct gun which will be used to destroy the threat. If the wrong gun was named then the user won't be able to shoot at the threat, until the correct one is named. In our current implementation you can shoot at the threat using any of the guns but only the correct gun will do damage to the threat. This is better as it's better for the user to learn from their mistakes about

			which gun links to which threat.
2.1.5	Ship Health	Yes, the ship's health decreases when attacked by a threat and increases when the ship engineer or ship enhancer team members are purchased.	N/A
2.1.6	Sound	No, when the user purchases the ‘space expert’ team member, the system audibly tells the user which threats are heading towards them. This will be done in the next stages of development.	Sound effects have also been added to the game ie. laser sound effect and explosion sound effect.
2.1.7	Controls during the game	Yes, the system incorporates the Oculus headsets and the Oculus Touch Controllers in the game. The game includes pause/play functionality.	Quit and restart not developed, however have been included in the future development section.
2.1.8	Gaining points	Yes, as the points increase when a threat is destroyed.	N/A
2.1.9	Detecting threats have been defeated	The threats disappear when they have run out of health. The functionality of the user winning when all the waves have been completed so the game ends will be developed in the next stages and hasn't been completed yet.	N/A
2.1.10	Detecting ship has been destroyed	Yes, when the ship runs out of health the user is taken to an end screen (the user has lost)	N/A
2.1.11	Buying power ups and team members in between waves in the shop (shop).	Yes, the shop has been implemented in the system. The user can buy team members by pressing the shop button and then each team member (given they have enough points to buy them).	Power Ups have been incorporated into team members’ functionality so they are redundant. Team members are available to buy and appear in the room once purchased. Each team member has a

			powerup functionally (see team member table above) Points go down when a team member is purchased.
2.1.12	Onboard user manual	Yes, a game manual can also be accessed on the back wall and contains the information on how to play the game, information on the threats and team members.	N/A
2.1.13	Threats coming at the user	Yes, the threats approach the ship when they have been generated.	N/A

Non-Functional requirements

	Requirement	Does the solution meet the initial requirement?	How and why was it changed?
2.2.1	Maintainability	Yes, we have taken maintainability into consideration at all stages of our game development. The maintainability will be discussed later in the user manual in the systems maintenance section.	N/A
2.2.2	Portability	Yes, we have implemented portability in our game by using github to code it which means that it is accessible from any device that has unity downloaded on it. The headset is portable as it is not very large or heavy. We are able to create executable versions of our game and distribute them easily.	We couldn't completely meet the portability requirement because our game file overall is quite large. This means that you can only download and play it on devices that have enough spare memory. However, this was out of our control as this is the nature of most VR games coded using Unity.
2.2.3	Useability	Yes, we have considered and implemented high usability at each stage of our game development. For example, our game is easy to follow as it is	N/A

		quite straight forward by nature, it provides instruction to the user in the game which they can read at any point during the game.	
2.2.4	Security	Yes, we have considered the security of the system. We have also used IBMS software which is secure in itself.	At project handover, we will give the client an executable file of the code (along with the code itself) which will make any distribution of the system more robust as the code itself cannot be changed.
2.2.5	Safety	Yes, we have considered safety by including appropriate safety warnings at the start of the game.	We also added warnings to users currently playing the game in the gameplay itself. This is added to the Back Wall under the 'Warnings' tab and can be accessed at any time.
2.2.6	Performance	Yes, having tested performance with the headset, the system runs smoothly with no crashes.	N/A

Unity and C#

Unity is an object oriented game engine and cross platform IDE that can be used for creating virtual reality games. Unity integrates well with the oculus headsets and provides lots of built in features which makes creating games like 'cyber invaders' less complicated which is why it was chosen for this project. Along with unity, C# has been used to code functions for providing functionality to objects in the unity scenes.

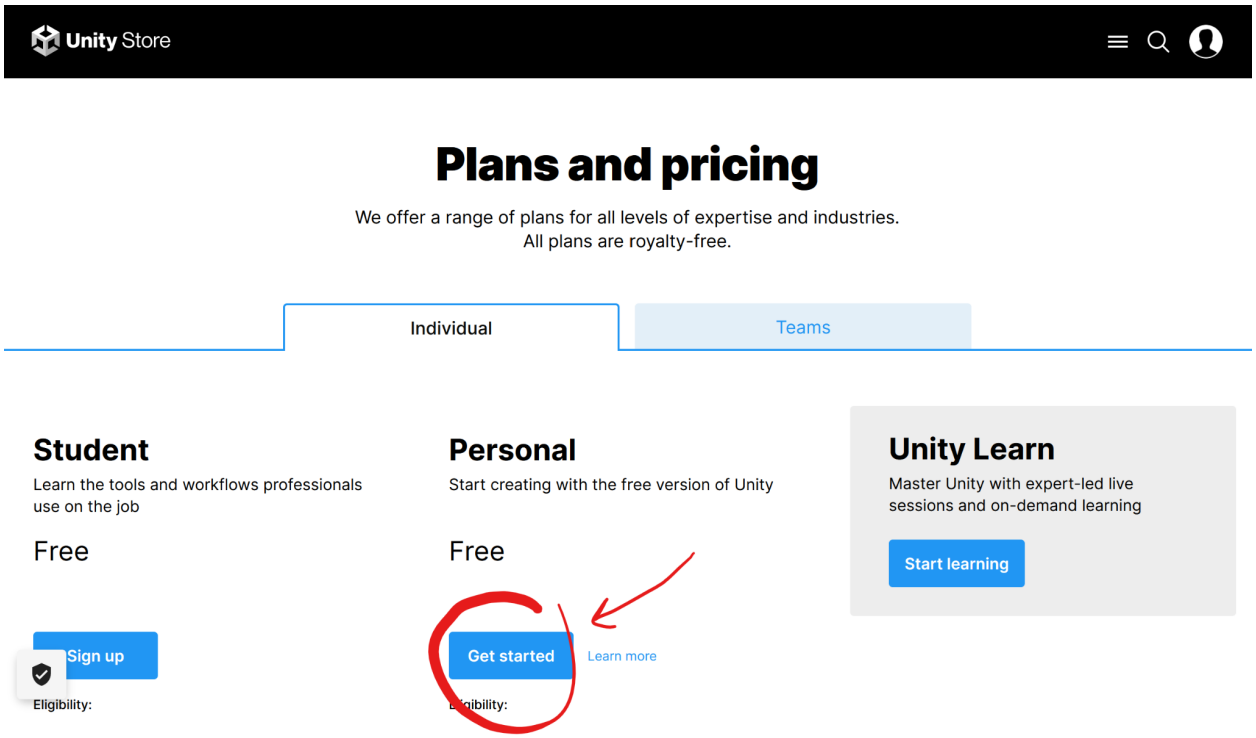
Unity and C# documentation can be found on page two of this user manual.

To download unity, follow the steps from:

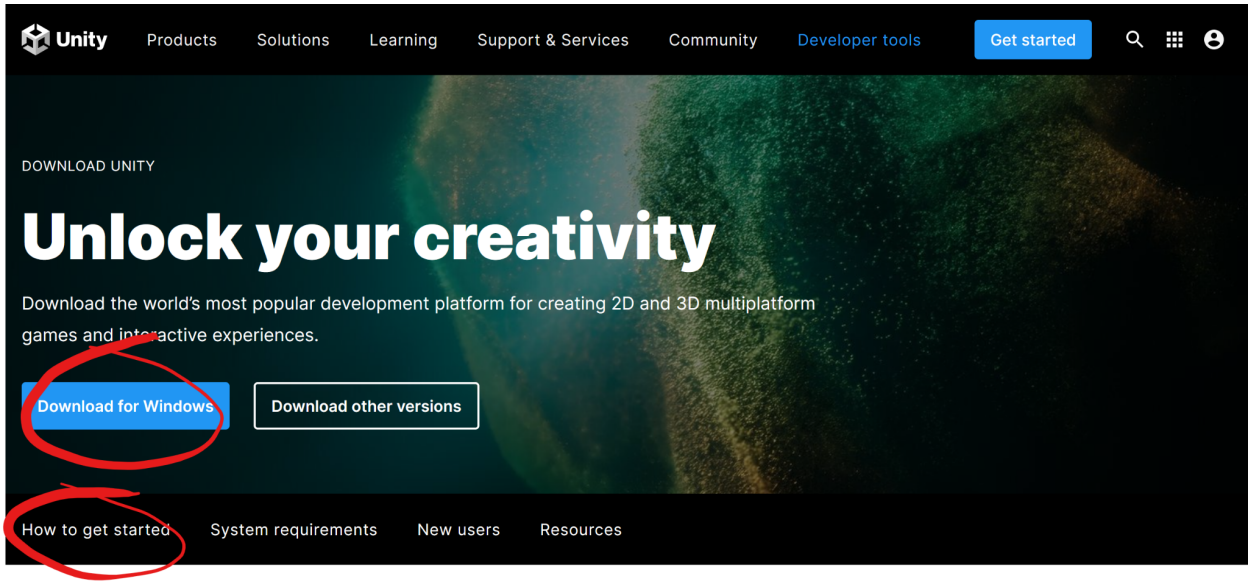
Unity Store

https://store.unity.com/?_gl=1*89ga40*_gcl_aw*R0NMLjE2NDY4MzQ4NzcuQ2p3S0NBaUF2YUdSQmhCbEVpd0FpWS15TUpGSERHNzNMeklybkNlQ3VvaE1SSnhDazFlOWZtdGdjWTNpb0pCS1VYRDhBSHV6anFfd3FSb0NyWGdRQXZEX0J3RQ..*_gcl_dc*R0NMLjE2NDY4MzQ4Nzcu

Q2p3S0NBaUF2YUdSQmhCbEVpd0FpWS15TUpGSERHNzNMeklybkNlQ3VvaE1SSnhDazFlOWZtdGdjWTNpb0pCS1VYRDhBSHV6anFfd3FSb0NyWGdRQXZEX0J3RQ..#plans-individual



Step 1: Go to the above link and click the ‘Get Started’ button



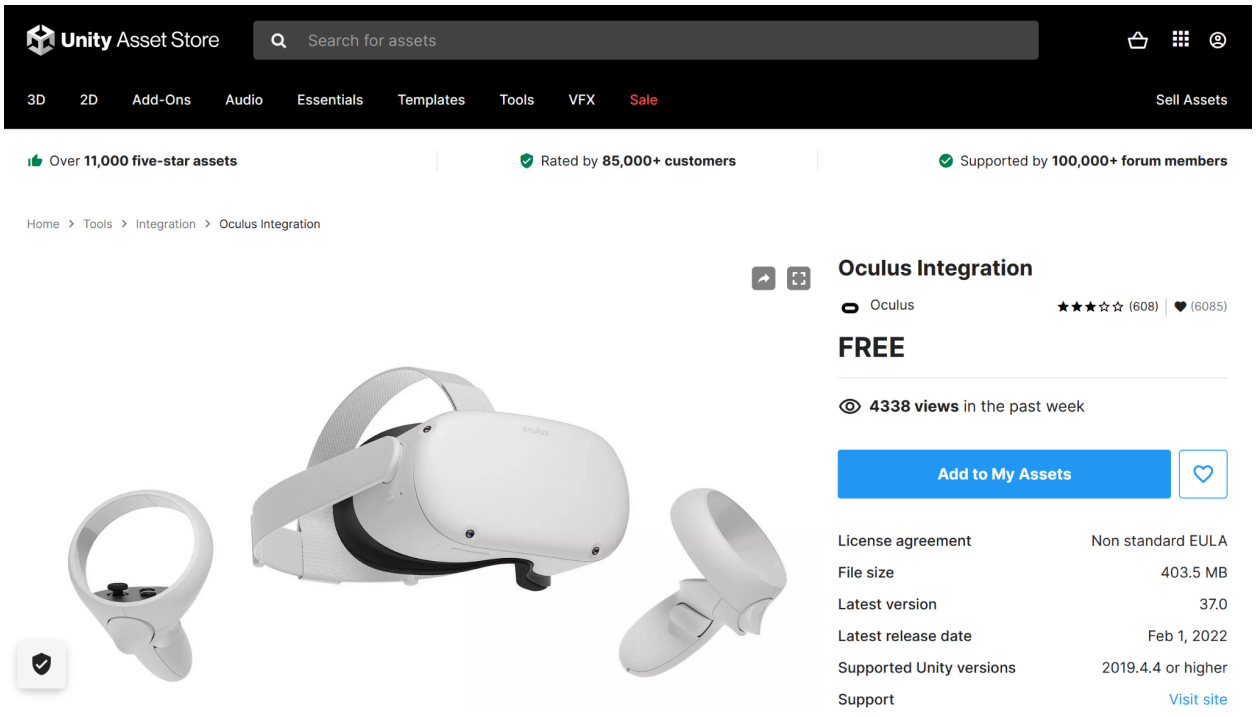
Step 2: Read this page, from top to bottom - make sure you have the correct system requirements, and read the ‘How to get started’ section

Step 3: Download Unity for your operating system; if you don't use Windows, then click the button next to it then click the 'Download other versions'.

Step 4: Follow instructions and dialogue boxes to complete downloading all the required packages needed to run Unity for development.

Assets from the unity store have been used in this project. The asset store provides prefabs of objects that can be used in Unity projects:

[Asset Store](https://assetstore.unity.com/?gclid=CjwKCAiA4KaRBhBdEiwAZi1zzl8nYpWQJ7gxE6Rqj9W5dc8mzcFjkEGyJ57xittFx2D9po7ZvhKY4RoCZwEQAvD_BwE&gclsrc=aw.ds)
https://assetstore.unity.com/?gclid=CjwKCAiA4KaRBhBdEiwAZi1zzl8nYpWQJ7gxE6Rqj9W5dc8mzcFjkEGyJ57xittFx2D9po7ZvhKY4RoCZwEQAvD_BwE&gclsrc=aw.ds



Step 5: Take a look at the Asset Store, this particular Asset is something that should be downloaded given that you are using an Oculus Headset to maintain the game from source code.

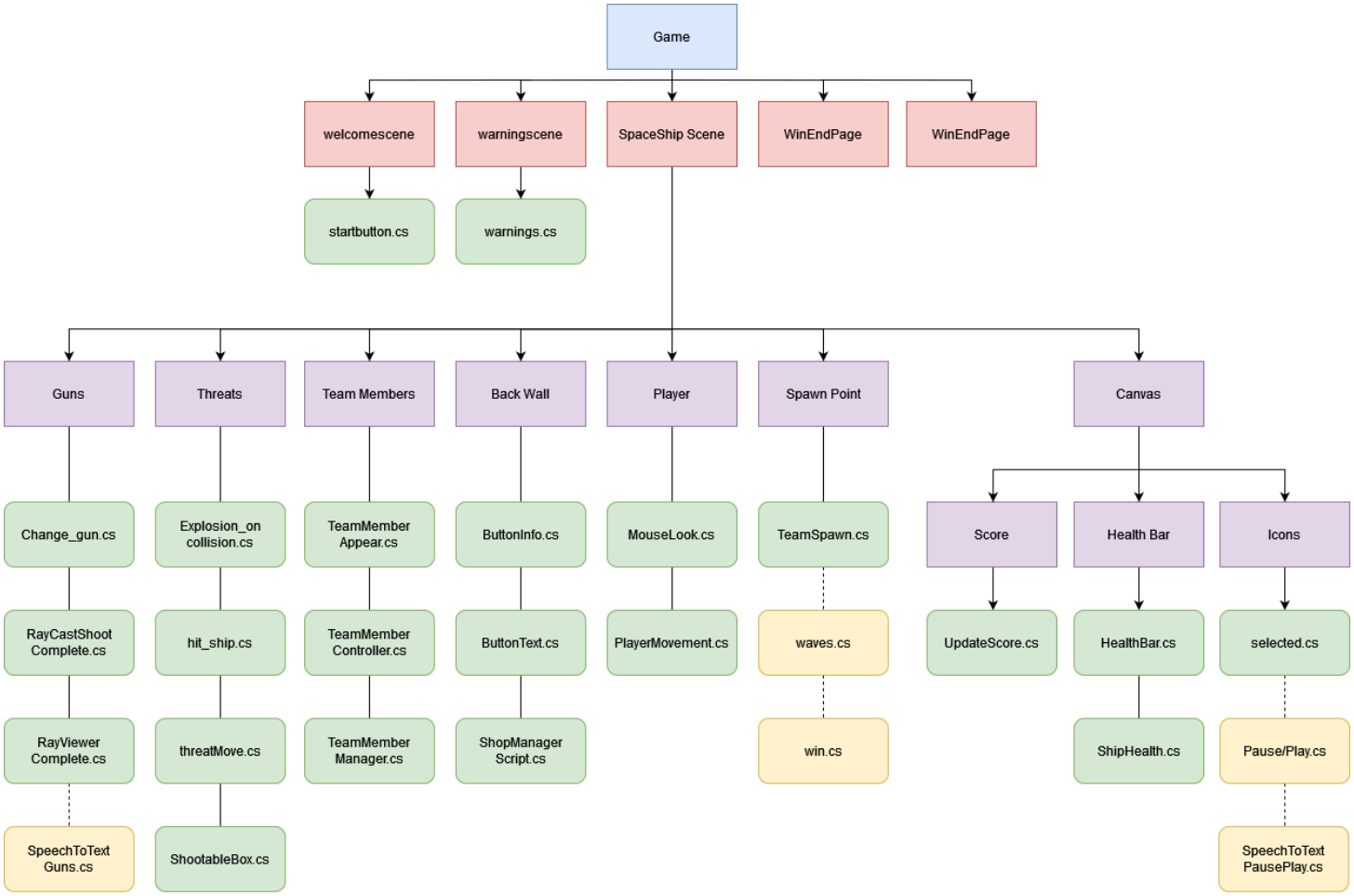
Step 6: You will have created an account during the download of Unity, use that to download Assets

Main component scripts

Introduction

This section will describe in detail each of the various scripts that our program uses. All scripts are programmed in C#. Figure.5 shows how each of the scripts link to the different scenes and game objects. The different colour boxes in figure.5 represent:

- ➔ Blue box- entire game
- ➔ Red box- different scenes in the game
- ➔ Purple box- game objects within a scene
- ➔ Green box- scripts attached to the game objects
- ➔ Yellow box- scripts that will be coded before the final hand-in but have not been completed yet.



change _gun.cs

Function name	description	input	output	Links to other script
Update()	Changes the gun which has been selected	N/A	N/A	Speech to text

explosion _on_collision.cs

Function name	description	input	output	Links to other script	Links to Object
OnCollisionEnter()	Destroys the threat that has been defeated (when the threat collides with the laser) and causes an explosion to occur.	N/A	N/A	UpdateScore	Threats

hit _ship.cs

Function name	description	input	output	Links to other script	Links to Object
Start()	Finds the GameObject with the tag of “health bar” and saves the component “ShipHealth” to a variable.	N/A	N/A	ShipHealth	Threats
OnCollisionEnter()	When the threat collides with any part of the ship it destroys that threat and causes damage to the ship	N/A	N/A		

RayCastShooterComplete.cs

Function name	Description	Input	Output	Links to other script	Links to Object
Start()	Stores the reference to the lineRenderer	N/A	N/A	N/A	Gun Main player threats
Update()	When the user shoots their gun, it causes the raycast _(the laser line) to be activated to make it seem like a laser has been fired by the user.	N/A	N/A		
shootEffect()	Makes laser sounds when the user fires their gun (called in update)	N/A	N/A		

RayVeiwerComplete.cs

Function name	description	input	output	Links to other script	Links to Object
Update()	When called it allows the raycast to be activated	N/A	N/A	RayCastShooterComplete	Gun
Start()	Finds the vector where the ray should start.	N/A	N/A		

selected.cs

Function name	description	input	output	Links to other script	Links to Object

Update()	Moves the highlight box to show which gun (or pause button) has been selected by the user	N/A	N/A	N/A	Green Highlight box in main game Icons

Threat move.cs

Function name	description	input	output	Links to other script	Links to Object
Update()	Moves the threat towards the ship	N/A	N/A	N/A	Threat

Timed spawn.cs

Function name	description	input	output	Links to other script	Links to Object
Start()	Starts spawning threats from the spawn point	N/A	N/A	Threats	Spawn Point
SpawnObject()	Spawns threats	N/A	N/A		

Startbutton.cs

Function name	description	input	output	Links to other scenes	Links to Object
PlayGame()	Calls the game scene	N/A	N/A	Game Scene	N/A

warnings.cs

Function name	description	input	output	Links to other	Links to

				scenes	Object
PlayGame()	Calls the welcome scene	N/A	N/A	Welcome scene	N/A

ShipHealth.cs

Function name	description	input	output	Links to other script	Links to Object
Start()	Sets the current health of the ship to the maximum health at the start of the game.	Integer maxHealth =100 Integer CurrentHealth	N/A	HealthBar.cs- uses setMaxHealth and setHealth functions.	healthBar
Update()	Sets the health back to maximum health when the ship engineer is bought.	Boolean ReapiredShip Integer maxHealth =100 Integer CurrentHealth		HealthBar.cs- uses setHealth function.	healthBar TeamMember
TakeDamage()	Updates ships health after the ship has taken damage.	Integer damage Integer currentHealth		HealthBar.cs- uses setHealth function.	healthBar

HealthBar.cs

Function name	description	input	output	Links to other	Links to

				script	Object
SetMaxHealth()	Sets the slider in the health bars value to the maximum health. Then fills the slider in green.	Integer health	N/A	N/A	slider fill gradient
setHealth()	Sets the slider in the health bars value to a updated value for health.Then fills the slider in with the correct colour of the gradient depending on the value the health is.	Integer health	N/A	N/A	slider fill gradient

MouseLook.cs

Function name	description	input	output	Links to other script	Links to Object
Start()	Locks the cursor to the centre of the view and the cursor is made invisible.	N/A	N/A	N/A	N/A
Update()	Rotates the player to face the way that the mouse is facing.	Float mouseSensitivity=100 Float xRotation=0f	N/A	N/A	playerBody

PlayerMovement.cs

Function name	description	input	output	Links to other script	Links to Object
Update()	This method moves the player when the user moves.	Float speed=7 Float gravity=-9.8 Vector3 velocity Bool isGrounded Float groundDistance	N/A	N/A	Controller groundCheck groundMask

TeamMemberAppear.cs

Function name	description	input	output	Links to other script	Links to Object
Update()	Makes a team member appear when it is bought from the shop.	Boolean appeared=false	N/A	N/A	TeamMember

TeamMemberController.cs

Function name	description	input	output	Links to other script	Links to Object
Start()	Sets the target for the team member to follow as the players	Target agent	N/A	Uses instance and player from TeamMemberManager.cs	Target agent

	character in the game.				
Update()	If the users player is in the set radius of the team member then the team member will follow them and face them.	Float lookRadius=10	N/A	N/A	Target agent
FaceTarget()	Sets the team members rotation to face the player.	Float lookRadius=10	N/A	N/A	target

TeamMemberManager.cs

Function name	description	input	output	Links to other script	Links to Object
Awake()	Creates an instance of team member.	N/A	N/A	N/A	Player instance

UpdateScore.cs

Function name	description	input	output	Links to other script	Links to Object
Start()	Sets the score value to a string	Text scoreText Integer score	N/A	N/A	N/A
IncreaseScore()	Increases the score by a certain change.	Integer change Text	N/A	N/A	N/A

		scoreText Integer score			
DecreaseScore()	Decreases the score by a certain change.	Integer change Text scoreText Integer score	N/A	N/A	N/A

ButtonInfo.cs

Function name	description	input	output	Links to other script	Links to Object
Update()	Initialises the price and the quantity of the items of the store to 0 when the game starts.	Int itemID Text PriceText Text QuantityText GameObject ShopManager	N/A	N/A	Item1 Panel

ButtonText.cs

Function name	description	input	output	Links to other script	Links to Object

Start()	Adds active on click listeners to all of the buttons to act upon when they have been clicked.	Button Warnings Button Shop Button Manual Button Tutorial	N/A	N/A	Warnings Button Shop Button Manual Button Tutorial Button
ActivateWarnings()	Changes the active state of warnings to True (it makes it visible) and toggles the other panels to inactive (making them invisible to the user) so that it is interactable.	Button Warnings Button Shop Button Manual Button Tutorial Canvas WarningsCanvas Canvas ShopCanvas Canvas Manual Canvas Canvas Tutorial Canvas	N/A	N/A	Warnings Button Shop Button Manual Button Tutorial Button WarningsCanvas TutorialCanvas ShopCanvas ManualCanvas
ActivateShop()	Changes the active state of the Shop to	Button Warnings	N/A	N/A	Warnings Button

	True (it makes it visible) and toggles the other panels to inactive (making them invisible to the user) so that it is interactable.	Button Shop Button Manual Button Tutorial Canvas WarningsCanvas Canvas ShopCanvas Canvas Manual Canvas Canvas Tutorial Canvas			Shop Button Manual Button Tutorial Button WarningsCanvas TutorialCanvas ShopCanvas ManualCanvas
ActivateManual()	Changes the active state of the OnBoard Manual to True (it makes it visible) and toggles the other panels to inactive (making them invisible to the user) so that it is interactable.	Button Warnings Button Shop Button Manual Button Tutorial Canvas WarningsCanvas Canvas	N/A	N/A	Warnings Button Shop Button Manual Button Tutorial Button WarningsCanvas TutorialCanvas ShopCanvas

		ShopCanvas Canvas ManualCanvas Canvas TutorialCanvas			ManualCanvas
ActivateTutorial()	Changes the active state of the Home page to True (it makes it visible) and toggles the other panels to inactive (making them invisible to the user) so that it is interactable.	Button Warnings Button Shop Button Manual Button Tutorial Canvas WarningsCanvas Canvas ShopCanvas Canvas ManualCanvas Canvas TutorialCanvas	N/A	N/A	Warnings Button Shop Button Manual Button Tutorial Button WarningsCanvas TutorialCanvas ShopCanvas ManualCanvas

ShopManagerScript.cs

Function name	description	input	output	Links to other	Links to

				script	Object
Start()	Initialises the array for the shop with ItemIDs, Prices and Quantity (currently 0).	Int[,] shopItems	N/A	Links the items from ButtonInfo.cs	Item1
Buy()	Used to buy items in the shop whilst decreasing the currency and increasing the quantity of item bought	Int coins Text CoinsTXT Int[,] shopItems	N/A	Links the items from ButtonInfo.cs	EventSystem with tag “Event” Item1 ShopManager score

Development has not been completed, as project handover to the client is April 29th. Scripts that have yet to be written are as followed:

Script	Description
WAVES	As there are 6 threats there will be 6 waves of the game. Each wave will incorporate a new type of threat Between each wave the user will see a message telling them they have got onto the next wave and which threat has been added
PAUSE/ PLAY	When the user says pause the threats will stop spawning and threats that have been spawned will stop approaching the ship. The user won't be able to shoot at the threats. This will give the user enough time to use the back wall (shop, user manual etc…)
WIN END SCREEN	Once the user defeats all the waves of threats they win the game, at this point they will be taken to the win scene (which has already been created)

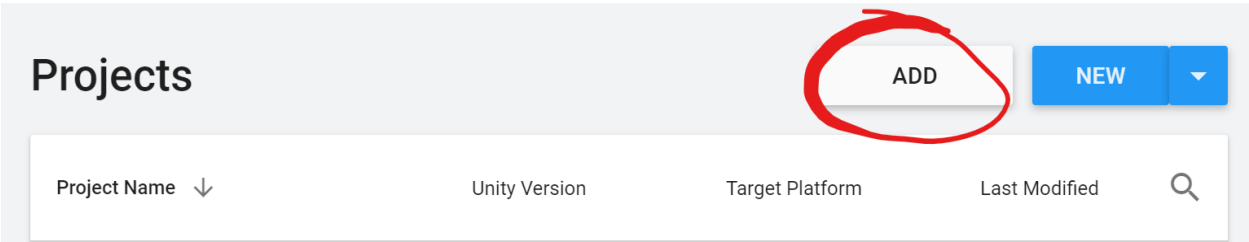
REMAINING TEAM MEMBERS	The Cryptographer, Ship enhancer, Space Expert, Threat Detector and Assistant shooter have yet to be implemented in the game. Scripts for each of the functionality of these team members is therefore needed to be written. Ship Engineer has been fully implemented into the system, so the development team already has a template on how team members can be implemented
SPEECH TO TEXT	To create the full VR experience the implementation of Watson Speech To Text will be completed. It will be used to swap guns during the game by saying the name of the gun that the user would like to use. Speech To Text will also be used for pausing and resuming the waves if threats.

How to run the program from source code

https://github.com/gracewest333/Software-Engineering-VR-and-AI-Serious-Security-Game

The people who wish to run from source code will need to download the GitHub repository onto their own device, making sure to have enough storage in the drive they are using. Also, IBM Watson requires an IBM Account which the user can find more information on through the repository in a relevant Markdown file.

Open Unity Hub, and click the 'Add' button; navigate to the game file downloaded, and then confirm. It should show up in the Unity Hub.

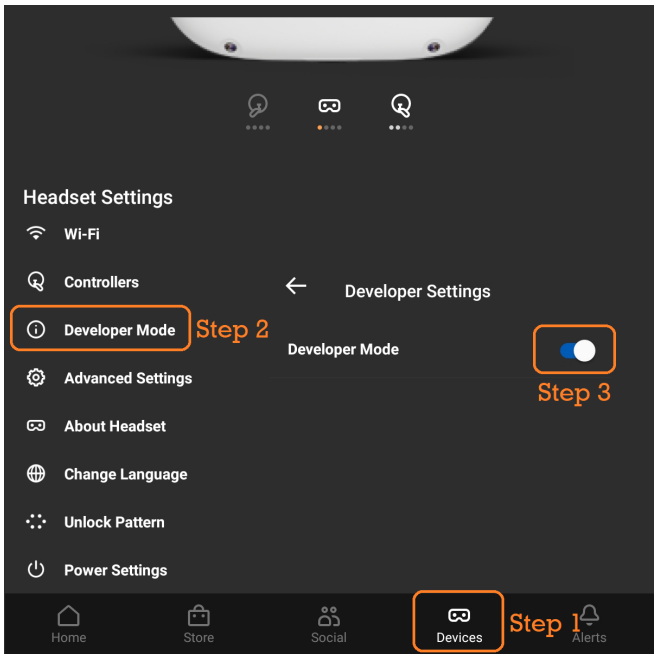


Open the world and you should see the game in 'Scene' mode. Click the tab that says 'Game' and press the play button which should run the game.

Note : If the headset functions have been fully developed (this will be clear in the repository) - please refer to Oculus Quest 2 and its Limitations. Then continue and follow the next few steps to connect and run the game on the VR

Device

Make sure the Headset is connected via the USB cable into the machine in use, and then make sure it's recognised.



Download the Oculus Application on a mobile device or alternatively, a PC application is available.

Follow instructions to connect the headset to the Oculus App - then follow instructions to enable developer mode so that if additions are to be made for maintenance then the backend of the headset is accessible. Keep an eye on the charge levels on both the headset and the controllers.

To access Watson Speech To Text the user needs to create an account in IBM Cloud where the Speech To Text service can be downloaded. Following the download the user will be able to access an api key and service url that enables the service. When the game is run in Unity there will be an object that is linked to the Speech To Text service. When clicking on this object whilst in Unity, an input field will appear on the right hand side of the screen. Here the personal api key and service url can be inputted as well as preferred language.

Limitations

As the first release of the system it inevitably has a few limitations.

Firstly, even with mitigations such as good code practices, e.g. appropriate variable names, use of white space, and commenting the code. It is not straightforward for a new developer to add new threats, guns and team members to the system. These additions would require major changes in the code. For example to add a new threat into the system a developer would have to execute the following steps.

- ➔ Duplicate a pre-existing prefab of a threat, giving it a new name, a new tag (and colour)
- ➔ Change what type of gun its defeated by (potentially at this step the developer may have to create a new gun to represent the defence it can be defeated with)
- ➔ Create a new wave of threats which incorporates this threat
- ➔ Add the threat to the database
- ➔ Add the threat to the user manual on the back wall

Secondly, with the given time frame and budget, the graphics for the game are fairly simple. The game had been made with free assets available from the unity asset store and objects we have made ourselves. Better graphics, buying assets/making assets ourselves. Given more time and/or money the game could have visually looked alot more exciting and even more tailored to the star trek aethstetic brief given by the client.

Finally, Given more time, people working on the project and experience working with unity and C# the system could have been more sophisticated in design and functionality.

Oculus Quest 2 and its Limitations

The headset was key for development since this is a VR game, and was part of the initial specification and needs to be implemented. Development was started between the end of 2021 and beginning of 2022, without a headset because one was unavailable to use at the time. Development without a VR headset continued throughout the new year until around a week before the final presentation which was on the 15th March 2022. Therefore, efforts were made to incorporate the VR element into the game at this point in development, but lacked in areas such as mapping controls to the Oculus Touch Controllers. The end of the project will hopefully involve the Oculus Touch Controllers, and a full working game in Virtual Reality. The lack of a headset did not slow down development as a fully working game was created, VR ‘style’. However, since this was not part of the initial specification, efforts are being made to meet the requirements for the client, despite these conditions not being in our control.

What was accomplished using the VR headset:

- ➔ Headset was set up successfully and connected to a developer’s device
- ➔ Headset was able to enable developer mode to be able to run an external application
- ➔ Unity connected to the headset and recognised it as a device
- ➔ Unity built to the headset (APK file which is the application, was run on the headset)

- ➔ Headset was able to render the game
- ➔ Oculus Touch controllers connected

What wasn't accomplished before the publication of this User Manual:

- ➔ Mapping all the keyboard and mouse inputs to the controllers, therefore there are vague references to these controls in this manual as this hasn't been fully implemented and tested.

Future development

This section will discuss future developments that can be made to the game to improve it and the user experience. The first development that the game would benefit from is a quit and restart function that is available at any point during the gameplay. This would likely be an extension of the game pause function, so that after the user has paused the game they are able to quit if they need to or go back to the beginning of the game if they need to. The quit and restart functions would also be used on the end pages when the user has lost or won. We felt that this isn't a development we need for our initial implementation of the game because the user can automatically quit the game just by turning the headset off and they can restart the game by restarting the program on the headset, however a formal version to quit and restart would aid game flow and user experience.

Another development which the game would benefit from is having a further options button on the home screen below the start button. If the user clicks on this button they should be taken to another screen where they can change: the volume of the game, the language that the game is in and the size of the text in the game. This will aid accessibility to the game as certain learning disabilities may require larger text for example. We felt that its development is important for user experience however, it is difficult to implement and would require a lot of time. As we were limited on time we had to focus on implementing the necessary parts of the gameplay which is why this is considered a future development.

More threats and team members would also benefit the game as it would add more levels and learning opportunities to the game making it more educational and enjoyable. Moreover, if more threats are added to the game more guns should be added to defeat them which would further the educational capacity of our game. In our current implementation we have implemented all the cybersecurity threats and professions discussed in the IBM we were asked to base our game off, which is why we didn't need to add anymore to our current implementation as it meets our client's requirements.

The final example of a development that the game would benefit from is having a multiplayer version of the game. This would be a good development because it would make our game

more interactive and exciting for users meaning it would be more popular which is good for our client. We didn't implement this in our initial implementation because it was not required by the client. Moreover, it would be very time consuming which would mean more important features to the games core functionality may suffer.

Systems maintenance

To maintain the system, stakeholders must make sure that their Oculus headset is regularly updated with any updates available from Oculus themselves and ensure that the program runs correctly on the new updates. Similarly, using current version of unity and C# is paramount. Regular checks on the packages and assets and their compatibility with the versions of Unity and Android versions for the headset need to be updated and downloaded from Unity itself.

Preparation for Final Handover (29th April)

This section covers what still needs to be implemented into the game for it to be considered to have completed and ready to be handed over to the client.

As we are far into the development process the team has a good idea of what still needs to be done and how it will be implemented into the current system. The team is confident that these final functionalities will be completed within the given time frame.

Finial implementations

- ➔ Create the six waves of threats, each wave incorporating one more threat than the last waves, giving indications to the user when each wave starts.
    - ◆ Link win pages to the game. Once the user completes every wave the win page is displayed
- ➔ Finish team members (implienting Cryptographer, Ship enhancer, Space Expert, Threat Detector and Assistant shooter) and adding them to the shop.
- ➔ Pause and play functionality, when the user says pause the threats will stop moving and no more threats will spawn into the game, the user will also not be able to shoot the threats (this gives them time to use the shop ect.) when they say play the threats will continue and the user will be able to shoot once more.
- ➔ Linking the sql database. The SQL database needs to be linked to the system so that the information in the database is shown on the back wall manual page and the shop.
- ➔ Connecting the controllers properly to the game by mapping all the inputs