G00327095 Gesture Project
Virtual Reality Shooting Range

GITHUB Link: https://github.com/JamesP1996/Project-for-Gesture-Based-UI-Development

Purpose of the Application

The purpose of this application was to create a virtual reality game to display the movement and immersion of modern VR devices along with gestures such as grabbing, point trigger movement, trigger pulling to shoot along with being able to pick up objects such as glass bottles or a two handed weapon which utilizes both hands in a realistic way.

I created a firing range using Low Poly Assets and a Terrain such that the game could run well on most devices because VR is often hard to run on older hardware and there is now standalone headsets such as the Oculus Quest 2 which offer standalone headset experiences on a mobile based chipset such as a Snapdragon Variant.

The initial considerations for Gesture Based Games and Hardware were to use Voice Controls using Grammar Recognizers, A Leap Motion or a AR based game but I opted to create a virtual reality game/experience because I felt it was an interesting topic and has endless applications to improve productivity and create entertainment experiences for users and these Virtual Reality Devices are only becoming cheaper to acquire for the common consumer over time. I felt that picking up some skills and knowledge of how their locomotive systems work along with their touch-based controllers and head tracking would be an invaluable experience to have as the world moves more into the Extended Reality space.

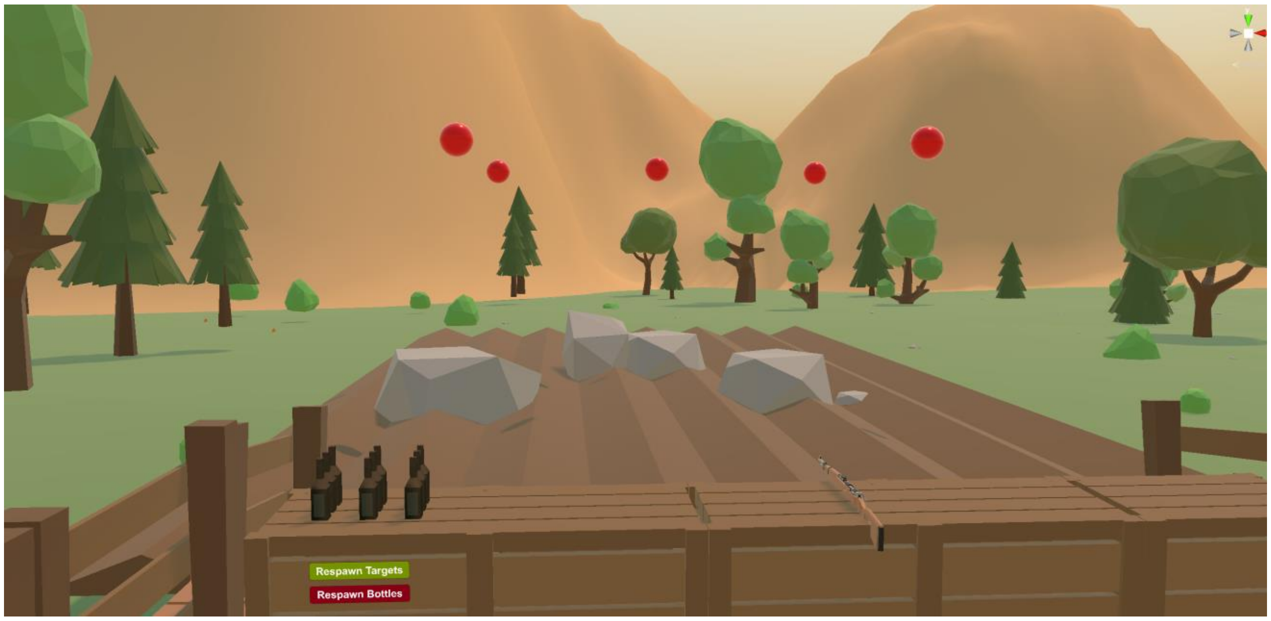
In the next section images of the application will be displayed along with a short intro to their purposes or how they work in layman's terms and then for the sections after such, greater detail of their workings and explanations will be given.

Testing of the Application

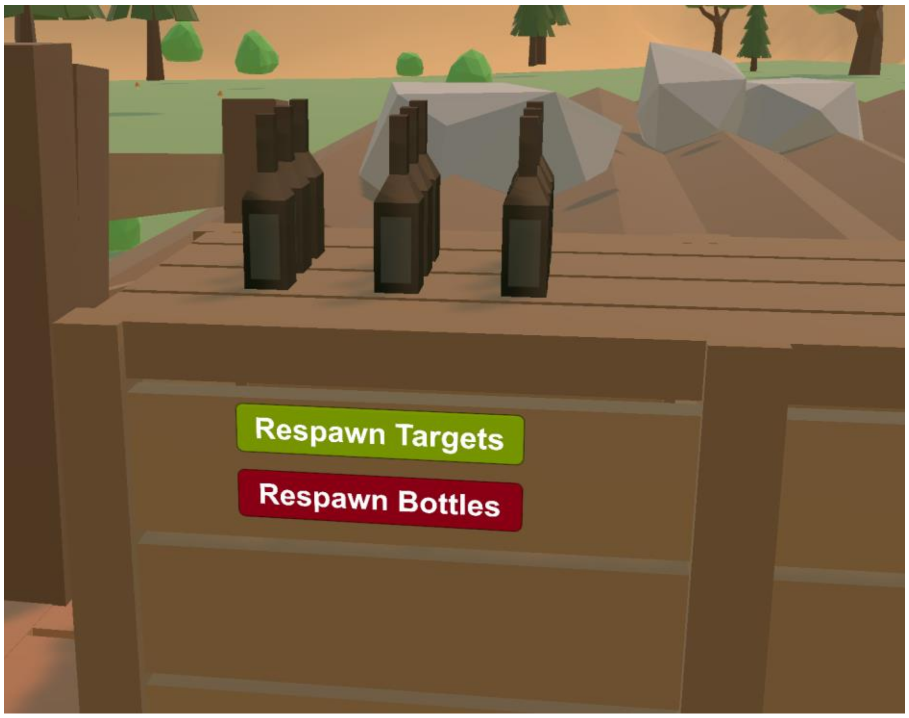
As I did not have experience within Unity or C# testing, my testing was limited to that black box user story testing in which I tried to come at the game from a user based perspective and test the different elements and how natural they felt to someone who may not be well versed in video games or computing. I play tested the game multiple times looking for major game breaking bugs and the only two issues I have found but cannot seem to repair is that the second hand grip is not always functional during the build versions of the game but overall it still possible to balance the gun on your secondary hand to shoot it and then in terms of the ray teleport interactor, left handed users may have issues with it showing if they hold the grip side of the gun with their dominant hand but the game is still functional despite this. Based off my own personal development experience with the XR Interaction Toolkit, black box testing seems to be a viable way to test the functionality of the different components as even if the code seems to work in certain circumstances, the game can still have bugs on different build scenarios or hardware such that even outside of this development for the game, I had seen through posts and videos made by other creators that they were having issues with their own code due to the XR toolkit consistent major updates and the general issues with the virtual reality space and its implementations within the unity engine but overall its still very useable and none of the bugs encountered should ruin the experience of being in a Virtual Reality Outdoors shooting range.

Images of User Interface

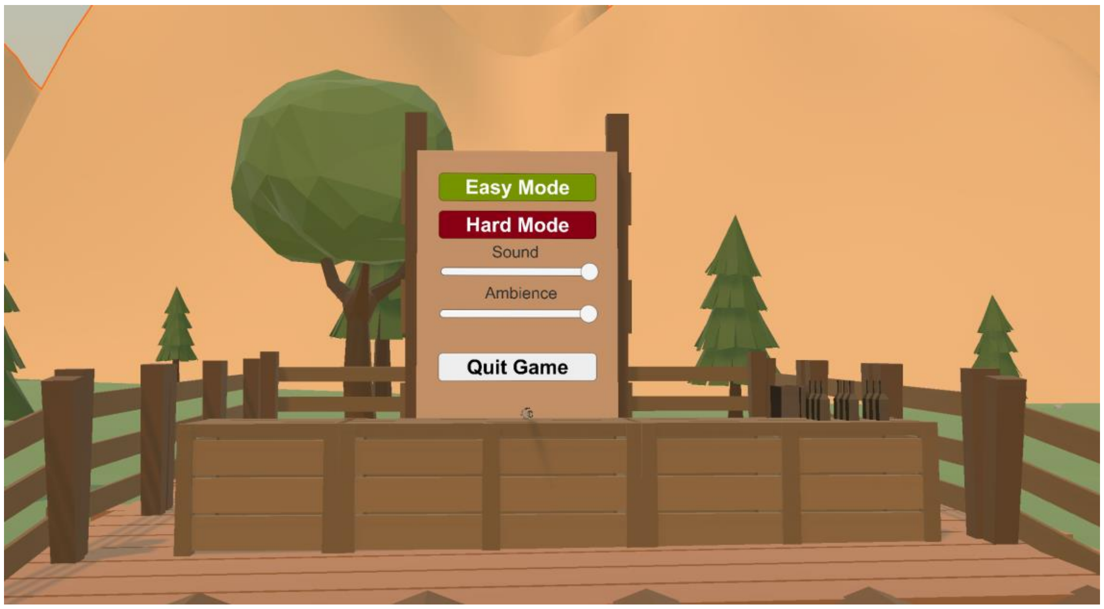
Shooting Range Forward (Targets and Bottles)



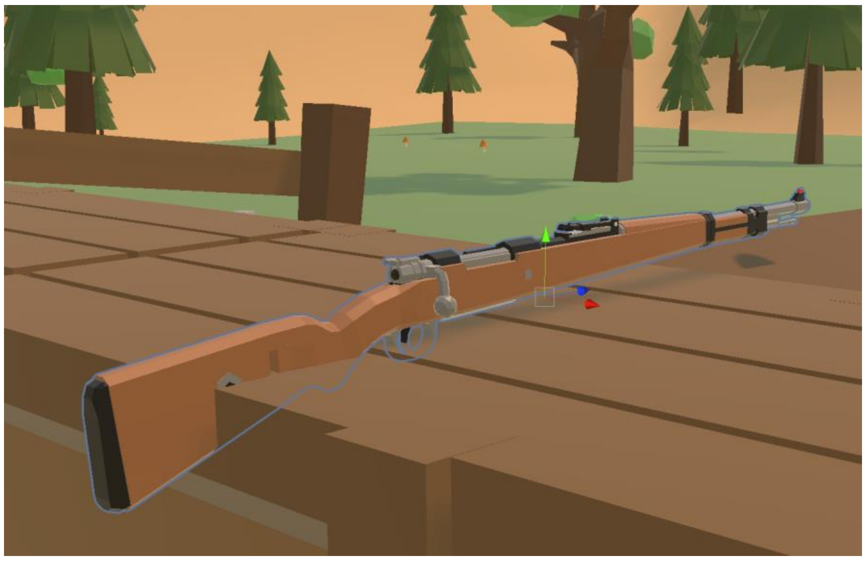
Bottles with Respawn Buttons



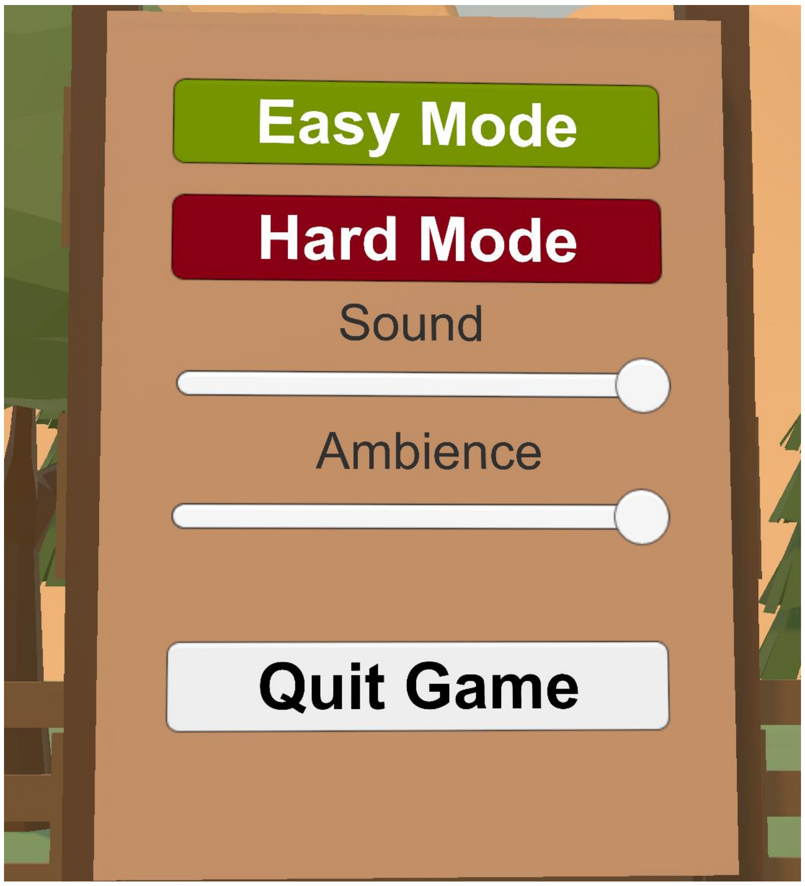
Behind Player – Back of the Firing Range(Contains Interactable Menu)



Two Handed Weapon (Can be fired in single hand which may be harder for the player)



Interactable UI Menu



How each screen shot works

1. Shooting Range

The Shooting Range **holds multiple spawn points** where targets will spawn in front of the player.

These targets have a **Boolean value** for whether they are **on their harder mode or not**. This will trigger whether their animation will play and **if they will move rather than stay still**.

On a targets destruction a particle system will play along with a sound to give feedback to the player that they successfully destroyed one of the targets.

2. Bottles with Respawn Buttons

In this screenshot you can see a **6 pack of bottles and 2 buttons**, one to **Respawn Targets** and another to **Respawn Bottles**. In this image the bottles are interactable with and the player can pick them up using their virtual hands linked to their virtual reality devices controllers. The user can decide to setup these bottles for shooting or just throw them around if they wish to.

The buttons are interactable with **the left-hand controller's trigger button** such that the user is **gripping their index finger downwards**. Depending on the UI button the player interacts with, the game will either respawn the **Bottles (Put them back into their original position on the table)** or **Respawn the Targets(Respawn Targets at the current difficulty level)**.

3. Back of the Firing Range

This is what the player will see if they turn around towards their back. It is somewhat empty, but the player will see a **large interactable billboard** and a **trash can to their right**. This **billboard works as the menu** as the player *can change the difficulty which will respawn the targets upon the difficulty selected* or change the ambience / sound effect volumes by interacting with the sliders. The **Ambience being the background bird and forest sounds** and the **sound effects being the gun shots, casing falling and dings from targets being destroyed**.

The user can also quit the game in the built version of the game by pressing the bottom button. This works by using Application.Exit(); which does not work on the Unity Editor, but I added a print statement to inform the Editor users what is happening when the button is pressed.

4. Two Handed Weapon (Kar 98k)

This is the players weapon they can pick it up with one hand or two hands and even drop it if they wish or throw it around although not recommended as there is currently no system to retrieve the weapon and the player is bounded inside the wooden fences. This weapon is complicated in that it tries to emulate reality with recoil, grip locations and firing animations along with actual projectile force and ricochets. The weapon will fire depending on the grip hand trigger being pressed, nothing will happen if the user uses the trigger on the guard hand apart from the normal locomotive teleportation movement.

5. Interactable UI Menu

This is the main menu of the game as explained in "(3) Back of the Firing Range" , this screen shot merely works as a zoomed in image of the menu values for smaller screens reading this documentation.

Gestures Identified as appropriate for this application

The current gestures in the application can be listed as follows.

- Hand Grip, Pinch and Open Hand
- One Handed pick up using Grip Axis on VR controller (Bottles, Gun)
- Two Handed Pickup (Holding Gun Grip and Guard Simultaneously
- Left Hand Trigger (Index Finger Press) Teleportation
- Left Hand Interactive UI Trigger Press (Index Finger Press)

Considered Voice Gestures but due to time constraints had to leave this feature out. No other gestures were considered that were within the realm of knowledge I gained, as it was hard to figure out certain components of the XR Development toolkit as the documentation and lack of forum resources on it made it difficult to figure out certain aspect. The only other VR option I would have liked to include is the ability to pull back the spring system in the rifle and let it go to reload the gun but I did not have the knowledge or resources to figure that feature's workings out along with how the mathematics behind it would work was beyond my skill level currently.

Hand, Grip , Pinch

I decided to give the user realistic hands from the Oculus provided hand models that you can access on their website. I then used animations with based on grip and trigger thresholds to show the users hand gripping into a fist or opening. I decided to use such features to give the user an immersive experience and make them as if it was there hands within the game, in the future it would be interesting to try create per finger animations based on the hands bones using the [Oculus Quests Hand Tracking feature](), and thus removing the need for controllers and making it fully hand gesture based.

The grip and pinching was just a natural animation I felt was appropriate to show the user that the users hands can pinch their fingers together which could simulate a trigger fire in real life if you were to that on a gun and grip which you would need to grip an object.

One Handed pick up using Grip Axis on VR controller (Bottles, Gun)

Using the grips on either touch controller the user can grip the bottles or the grip handle/ guard grip of the gun. This was originally tested using a sphere and I felt it added some dynamics to the game, I decided upon having the user grip objects rather then starting with them in their hand as it would be more immersive and show further the potential dynamics you could have within a VR application as in most modern fps like Call of Duty, you spawn with a gun in your hand and can only cycle through those options, but in virtual reality it feels a lot more effective to have the user physically pick up what they want and with the 6 Way Degrees of Freedom the Oculus Quest 2 offers the user needs to physically bend over or stretch out their body to grab the object like they would in real life.

Two Handed Pickup (Holding Gun Grip and Guard Simultaneously)

The player can use both controller grips to grip the gun's grip handle and guard simultaneously, although as the developer I am right handed, there is issues with ray casting on the left hand for left handed users, I was unsure how to fix this issue but the functionality remains the same.

I chose to allow the gun to be gripped by two hands as it's a rifle and would require such in real life, thus adding to the experience, it also allows the user to point the gun using their guard grip hand and have a steady shot as the recoil will be controlled by the grip of both of their hands whilst with one hand it would be harder to aim as the recoil may force your single hand out of position causing further re-adjustment then you would need to with two hands. It is also possible to hold two bottles at the same time or hold the grip of the gun while holding a bottle and throw the bottle in the air and quickly shoot it, for more skilled players.

Left Hand Trigger (Index Finger Press) Teleportation

By actuating the left controllers trigger button with your index finger, it will bring up a ray casted ray to teleport the player to the cylinder reticles position. This option was decided upon for ease of use as not many players will want to move or have the play space to move only using the XR Rigs natural walking play space that is within the 6 Degrees of Freedom however it is possible with a big enough play space to do so and never use the teleportation option. This Ray Cast does will appear red on un-teleportable places and a blue color for where the user can teleport to. By offering this features its very easy to navigate the players XR rig through the shooting area as needed and even hold a gun or bottle while doing so.

Left Hand Interactive UI Trigger Press

By actuating the same index button on the left controller upon the trigger, when your hand is pointed at the direction of a UI element such as the UI billboard or the Spawner buttons near the Six Pack Bottle prefab, it is possible to interact with the UI buttons and perform their actions. I decided upon using the same trigger for both actions as during play testing it felt natural for the teleporter locomotion movement system and the UI interaction to be upon the same index actuation. There is scripts setup to prevent two rays being cast when this happens as it will determine whether you are pointing at a UI element or a teleport area. During my time with the oculus quest it was also quite normal to use the trigger buttons for most UI based or movement actions in games like Half Life ALYX released by Valve so I wanted to keep this familiar system for accustomed VR players.

Hardware used in creating the application

For this application I decided upon using the Oculus Quest 2 which was released in October 13, 2020 and has been receiving consistent updates since then to improve its specifications and speed. Such as the 75hz to 90hz update along guardian system improvements. *The guardian system allows the user to see their real world through the oculus's inbuilt cameras that are used for the 6 Degree of Freedom Tracking.*

I decided upon purchasing this device as it retailed for cheaper then the average virtual reality headset at a price of 300 to 350 euro for the 64 GB variant but offers excellent visuals and tracking and software compatibility such that it can be developed for easily on the unity engine along with another other SDK's that are popular among VR development for oculus based products.

The other options were not available to me but comparable hardware that could serve the same purpose and even work with this game are as follows:

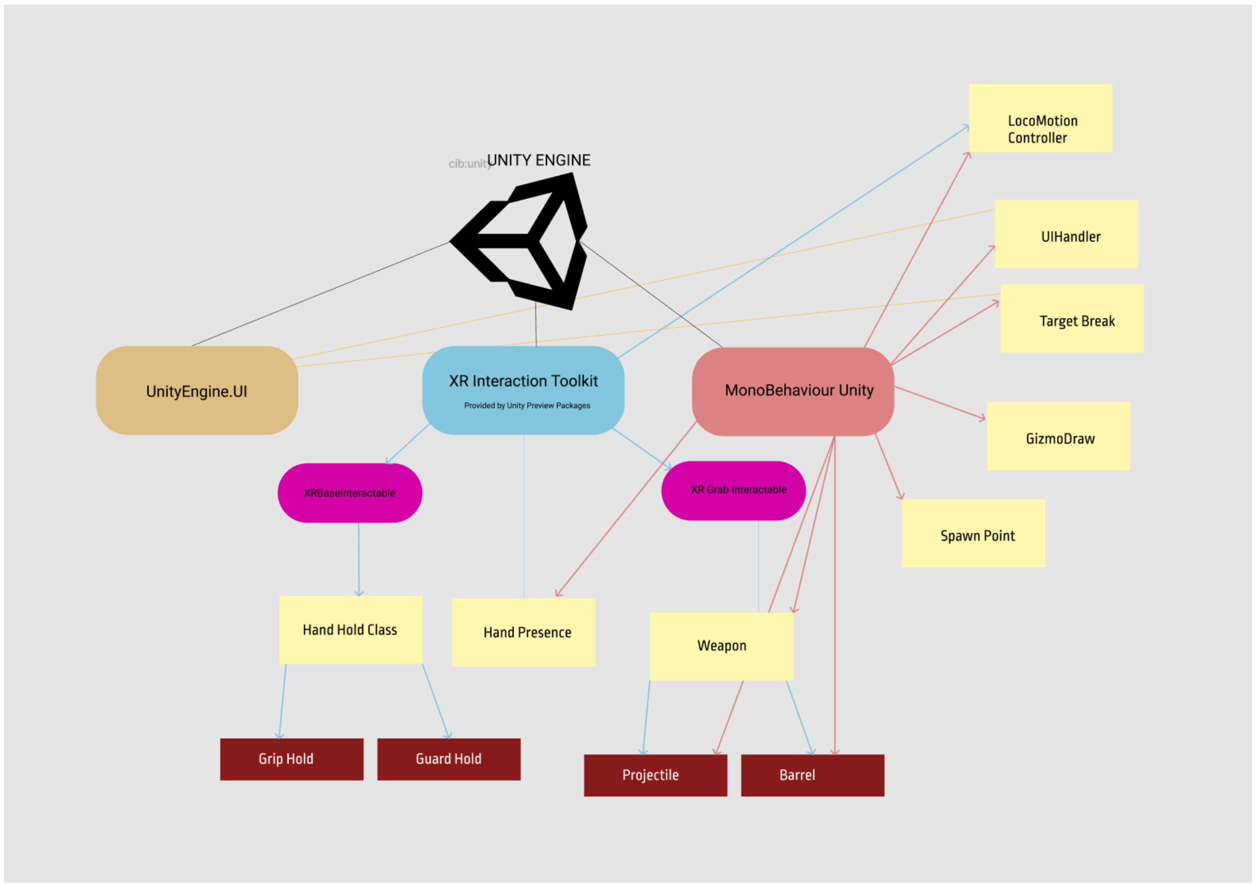
- Oculus Rift S – 400 Euro
- Oculus Quest 1 – 400 Euro
- HP Reverb G2 – 600-800 Euro
- Valve Index – 850-1000 Euro

All of the above headsets are supported by the XR Interaction Toolkit and will achieve the same effects within the game. The XR Interaction toolkit is Unity's own solution to the issue of having multiple Virtual Reality Hardware products and trying to unify their control systems such that a developer can create a game that will work on all headsets and not on a per VR producer brand name such as Valve , HPF or Facebook(Oculus) based hardware. This toolkit was released in 2020 and receives consistent updates such that as of right now its fall back is that it is hard to keep up with the current standards. An example of this within the development of the Shooting Range VR Game provided at the GitHub link within the document, was that for the XRBaseInteractor Extensions in my classes, I had to change the code from the tutorials I read or watched from OnSelectExit() / OnSelectEnter() to OnSelectExited();/ OnSelectEntering(); as the older versions are being deprecated. For this reason development within the XR Toolkit was complicated in that the solutions to certain issues are no longer supported by Unity openly and at some point will be break your game in later updates of the Unity Engine if you do not update them, along with the lack of good documentation or examples available for the newer version of the XR Interaction Toolkit preview versions.

A future consideration would be to develop a AR version of this application using a HoloLens or Google Glasses but as of right now, those options are not available to me but it would be a interesting project to partake upon as a proof of conception of the diversity these Extended Reality Devices have.

Architecture of the Solution

Class Diagram



Explanation of Architecture and Assets/Models Used

For this project, **Unity Version 2019.4.24f1** was used although I am certain the XR Interaction Toolkit will work with later iterations of the Unity Program. The key Unity developed libraries/packages used are the Mono Behaviour Library, XR Interaction Toolkit and UnityEngine.UI. The only one that requires a separate package to be downloaded through the Unity Package Manager under Preview Packages is the XR Interaction Toolkit, for Oculus Development it is also recommend to download their Oculus Integration Package such that you can get scripts and models that can help in Oculus based device development.

I created the game level first before setting up targets or files or UI options, for these I acquired some low polygon assets from the Unity Asset Store.

Assets Used Listed as Follows:

- Trash Low Poly Cartoon Pack *by BlankfaceStanislav*
- Toon Muzzleflash Pack *by David Stenfors*
- Low Poly Farm Pack Lite *by JustCreate*
- Low-Poly Simple Nature Pack *by JustCreate*
- GameObject Brush *by Kellojo* ***(Used to Paint trees and rocks onto World)***

The purpose in using Toon or LowPolyGun Asset packs is to ensure that the game runs well on both Mobile and Desktop based devices or lower end hardware's.

In terms of the Gun Model I got a **Kar98K model from Sketchfab** and then created some small animations for it personally.

XR Rig

Once the basic world had been created to look like a shooting range. I began to add the **XR Rig Functionality from the XR Interaction Toolkit** and set it up to use the **Floor Height for Tracking** such that the height of the player in the world will follow their real-world height. This is to prevent **motion sickness** that is often caused by VR.

Player Hands and Grip Types

I then needed to make the VR hands for the player to use, I followed a tutorial on YouTube for this , the animations belong to the youtuber, but the original hand prefabs belong to oculus themselves.

The **Hand Presence** class handles the presence of VR hands in the world under the Right- and Left-Hand Game Object. This class uses elements from the XR Interaction Toolkit to work such that I use the Input Characteristics of the controllers and then use their features values to control the hand animations.

I then needed to create the **Hand Hold** script which extends the XRBaseInteractable library, this class handles the holding actions and plays listeners depending on the hold state. The code did not require work as a lot of the functionality for holding objects was built into the XR Base Interactable Library and these settings were adjusted within the Unity Editor, for example On Hover Haptic Feedback or Which Buttons/Hands are assigned to.

Then I needed to build classes for the **Grip Hold** and **Guard Hold** such that they are separated, and the user can pick up the gun with two hands and use it in a realistic way. These classes extended my Hand Hold Class and over written behavior based on the XR Base Interactable library such that when a user begins a action , i.e. pulls the action button the gun will shoot or if I release the trigger the gun will stop its shooting coroutine, this class also includes logic for the Grab and Drop Interactions.

Within **Guard hold** the script is simpler than grip hold as we are just assigning if which hand is the guard hand and whether to drop that interaction.

There is also a Script called **Locomotion Controller** to handle player movement using XR Ray Casting such that the player can teleport to certain locations with a Teleportation Area. It has methods to check if it has been activated and **if it is allowed to activate based on if the Interaction Ray for UI .** If it6 Ds allowed to teleport to a location it will show a cylinder based reticle and a blue ray cast line otherwise it will show up red with no reticle.

Weapon Scripts

Next, I needed to create the weapon scripts, such as **Weapon, Projectile, Barrel** and assign them to their objects accordingly. Such that the gun script is on the Kar98K model, the Projectile is instantiated from the **Barrel Script** and the **Barrel Script** Is attached to an empty game object at the end of the barrel of the gun. This part was awkward to do as a mis alignment of the empty game object to the barrel of the gun would lead to shooting inaccuracies where the user is otherwise being accurate and then throw off the play experience and immersion for the user.

Within the **Weapon** Script I made multiple **variables based on the hand interactors, animators, grip rotations , recoil, hand holds, the barrel, the muzzle flash, and the guns rigid body**. All these variables made it hard to setup the gun as intended but I made multiple methods intended for this functionality to try cleaning out each method with its assigned purpose only being left.

On the weapon scripts **awake, I setup the holds which assigns which hand is touching which grip** on the weapon along with the extras such as the rigidbody of the gun, the muzzle flash within its children which is activated and de-activated based on firing along with the barrel.

There is methods to setup the **initial rotation** of the gun such that when the user lets go of the gun, the original collider will revert back to its original **position** such that if the user grabs the gun from the handle it will not be in a direction that the user may find odd and will **default back to their grip hands regular rotation**. There are also methods for setting up the **hand grips and clearing them as needed**. Then a **major method called process interactable** which will **run the SetGripRotation(); methods per update of the interactable component** which are **inheriting from the XR toolkit suite**.

Lastly within the weapon script we have the **PullTrigger()** and **ReleaseTrigger()** functions along with **ApplyRecoil();** and a **IEnumerator of MuzzleFlash();** . These are the main interactions of the gun functionality, such that it fires when the trigger is pulled, it stops when the trigger is released and then the apply recoil method will add a impact force to the gun to push it back.

Targets

The targets spawn upon the target spawners within the scene, in short it holds a **Boolean** for which mode it’s in called **hardMode** and another for **UIChange**. On **Start** of the script we check which mode we are in and then spawn targets of that type based on it **(2 Target Types, Target and Target Easy).**

Within the **Update** function we check if a UIChange has been triggered i.e. the user has clicked a UI button to change the difficulty or respawn targets and then appropriately spawn the type based off the previous hardMode Boolean values.

TargetBreak sets the metal ding sound volume within the game when a bullet interacts with a target on collision.

Projectile

The projectile script has two methods **Launch();** and a **OnCollisionEnter();** based method. The launch will send the bullet from its current point with a certain force amount and destroy it within a pre-defined lifetime from a variable. If this bullet collides with a collider of a Target, it will instantiate an **Empty Game Object with a particle system** to it thus **giving the user visual feedback** of it being destroyed and adding satisfaction to the user experience of shooting and destroying spherical targets floating in the air.

After some time has passed (less than a second), the script will continue to destroy the target and the particle systems that were spawned in the location of that target upon its collision.

UI System

The UI system is handled by using **UnityEngine.UI** and it also **uses Tracked Device Graphic Raycaster by the XR toolkit**. The UI handler keeps tabs on the bottles, spawn points and the audio sources along with what difficulty the game is in and whether it is muted or not. I setup **sliders for the ambience sounds and then the sound effect sounds** such as shooting when the user uses their **left-hand trigger and points to the sliders or buttons on the UI they can change or activate the values using the trigger button on the left hand controller**. Currently the **user can change, the sound levels, the difficulty or quit the game** along with on the **smaller menu by the six pack bottles** in the level they can **Respawn** the **Bottles** or the **Targets**. The **Sound Sources are gotten by iterating through all the sound sources** in the level **per update** but in future developments, I believe it would be best to make some sort of sound manager that could manage these systems rather then a consistent iteration through them which **could cause latency issues if the game were to become bigger in size.**

Conclusions and Recommendations

In conclusion I picked up the basic functionality of the XR toolkit provided by Unity in order to develop Virtual Reality based applications in a cross platform way, these applications tend to be gaming software's as that is what the Unity program is aimed at but I believe it's also possible to develop other types of applications if need be. Through my research I learned that although virtual reality is a growing field, the Software Development Kits and associated packages with it are constantly changing thus making it harder than normal applications to develop solutions for as the documentation and common places to find information such as YouTube and Stack Overflow can outdated. Unity's own documentation would be the best bet for learning the XR interaction toolkit and I would recommend any developers using packages such as the original Oculus OVR version or SteamVR versions to switch over to the XR toolkit for unity based development as the other kits will soon be deprecated and lack support for modern unity application post the year of 2021. By switching to a cross platform idealized library/package within Unity the developers could also lower their development time by not needing to develop for each separate Virtual Reality Platform.

If I were to partake on this type of project again, I would consult the documentation by unity first instead relying on trying to find information off of Stack overflow, You tube or Git hub as most of these sources are outdated and are very precise in their approach such that it would be better to consult the official documentation by unity such that I could add a more customizable taste to my application and try to add the features that I desire, but with this I can say that this would need more time as navigating documentation never tends to be the easiest task especially for a completely new system such as Virtual

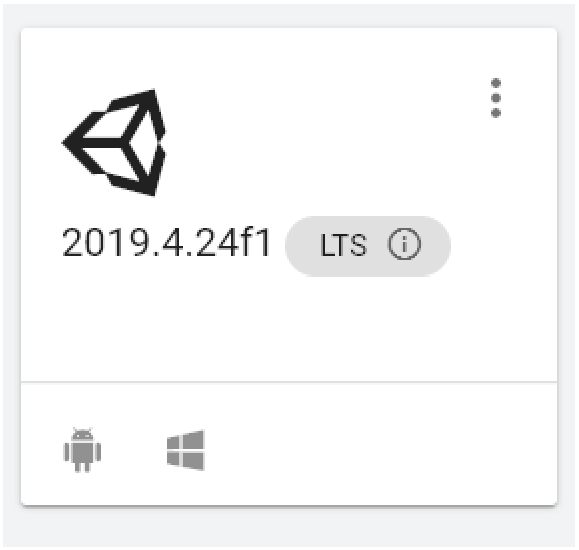
Reality and I would also need to acquire the mathematical knowledge to understand the Quaternion System properly such that the placement of in game objects or animations would be realistic and stay immersive in their approach.

I would like to one day test the limits of XR development kit and learn more about optimization for Virtual Reality games as these headsets often at resolutions way higher than the common computer screen of 1080p or 1440p and it would seem that headsets in recent times are only improving in visual clarity and this resolution will only exponential go up as different companies compete to have the best virtual reality experience on the market.

Overall I did enjoy my time creating this project as the aspect seeing my creation within a 3D environment that I could move my physical body in and have it correlate to the game was exhilarating in a way that could not be understood by those who have no done it, I will definitely try to develop future VR applications as the market for it fresh and in my own personal experience there are not many good solo developer VR games and if I were to hit this market with well developed projects, it would both be fun and profitable.

The objectives of making a Virtual Reality Shooting Range were met but critically speaking I would have liked to add more physical interactive elements to the game such as pulling the springs back in the gun or adding more diverse gun types that require different reload systems such as the PC and Oculus Quest game Pavlov has.

Unity Version


2019.4.24f1 LTS

- Toon Muzzleflash Pack *by David Stenfors*
- Low Poly Farm Pack Lite *by JustCreate*
- Low-Poly Simple Nature Pack *by JustCreate*
- GameObject Brush *by Kellojo* ***(Used to Paint trees and rocks onto World)***

Assets Used

Name and Author	Resource URL
Trash Low Poly Cartoon Pack *by BlankfaceStanislav*	https://assetstore.unity.com/packages/3d/trash-low-poly-cartoon-pack-66229
Low Poly Farm Pack Lite *by JustCreate*	https://assetstore.unity.com/packages/3d/environments/industrial/low-poly-farm-pack-lite-188100
Low-Poly Simple Nature Pack *by JustCreate*	https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153
GameObject Brush *by Kellojo*	https://assetstore.unity.com/packages/tools/utilities/gameobject-brush-118135 ***(Not Required Most Likely)***

Packages Used

Oculus XR Plugin 1.8.1
XR Interaction Toolkit 0.10.0-preview.7
XR Plugin Management

References

Name	Ref. URL
Unity XR Interaction Toolkit	https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/manual/index.html
Oculus	https://www.oculus.com/
Valve	https://www.valvesoftware.com/en/
HP Reverb G2	https://www8.hp.com/us/en/vr/reverb-g2-vr-headset.html
YouTube XR	https://www.youtube.com/watch?v=Hnoad3DM_pA
Valem Youtuber	https://www.youtube.com/channel/UCPJlesN59MzHPPCp0Lg8sLw
VR with Andrew Youtuber	https://www.youtube.com/channel/UCG8bDPqp3jykCGbx-CiL7VQ
Oculus Integration	Link

Steam VR Package	Link
Motion Sickness	https://developer.oculus.com/blog/vr-sickness-the-rift-and-how-game-developers-can-help/?locale=zh_TW
VR Performance Optimization	https://developer.oculus.com/documentation/native/pc/dg-performance-opt-guide/
VR Price Drops	https://kei-studios.com/vr-headsets-are-dropping-in-price-is-this-good-or-bad-news-for-virtual-reality/
Unity Development in VR	https://docs.unity3d.com/Manual/VROverview.html
The Future of Virtual Reality - IDERBROLA	https://www.iberdrola.com/innovation/virtual-reality
VR Depths of Field 3dof vs 6dof	https://pixovr.com/2019/10/09/difference-between-3dof-6dof/
Oculus Hand Tracking	https://support.oculus.com/2720524538265875/
Quest 2 Update Release Notes	https://support.oculus.com/release-notes/

GITHUB Link: https://github.com/JamesP1996/Project-for-Gesture-Based-UI-Development

Screen Cast Link: https://youtu.be/WSM4RZX7TrU