Project Deliverable K - Group 5

User and Product Manual

GNG1103-C01

December 10th, 2023

Noor Trigui 300350022

Ava Butts 300350306

Luca Chayer 300375838

Dev de Haan-Sharma 300339994

Table of Contents

List of Acronyms and Glossary

Table 1: Acronyms

Acronym	Definition
BBUPS	Blueprint Brigade Unity Prototype Simulation
UPM	User and Product Manual
VR	Virtual Reality

Table 2: Glossary

Term	Acronym	Definition
Latency	N/A	The gap in time between the users input and the input being reflected on the screen.

1 Introduction

This User and Product Manual (UPM) provides the information necessary for (users) to effectively use the Blueprint Brigade Unity Prototype Simulation (BBUPS) and for prototype documentation. The BBUPS is a virtual reality simulation made in the game development software Unity. It was created for the client Mines Action Canada and it intends to convince policy makers and the public that autonomous killer robots are a danger to our society and should be banned by governments around the world. Our intended audience for this document is anyone who is using or intending to use the BBUPS. We believe that this document will enhance the experience of the user by making the BBUPS more accessible and understandable to the user which will make it more convincing which was the goal of the BBUPS. We assume that users of the BBUPS have a basic grasp of modern technology. This document is divided into six major sections, overview, setup, using the system, troubleshooting and support, product documentation and conclusion. The manual is structured in this fashion to guide the user logically and sequentially through the process of using the BBUPS. The overview section gives the user context about the simulation. The setup, using the system and troubleshooting section all guide the user on how to use the system once they understand it. And finally the product documentation section shows the user how the product was made.

2 Overview

Mines Action Canada is one of the leading humanitarian organizations in Canada focussed on advocating, researching, and engaging youth in social justice. This project aimed to raise awareness about the devastating effects of autonomous killer robots on societies throughout the world through creating a Virtual Reality (VR) aimed at policy makers, who would have the power to preemptively ban autonomous killer robots. Based on our research from the campaign Stop Killer Robots (partnered with Mines Action Canada), we found that these weapons exacerbate systems of systematic injustice because robots work through a facial recognition system and often the data is not representative of the diversity present within worldwide communities[2]. Thus, some of the most important concerns addressed were dehumanization, systematic injustice, and racial profiling.

Based on the raw data gathered from the client in the first meeting, we divided our client needs into three sections: design specifications, content specifications, and overall impact.

Design Specifications	Content Specifications	Overall Impact
• Create VR curated experience • 30-60 secs • Final product runs smooth enough to be convincing	• Address 2-3 specific concerns as described by the client (Digital dehumanization; Inability to abide by humanitarian law; Technological failures and hacking; Bias & morality; Arms race and diversion of resources)	• Convince policy makers to preemptively ban autonomous killer robots

To specify fundamental needs, we quantified the original client needs and separated them into target specifications like functional requirements, non-functional requirements, and constraints.

	Design Criteria	Relation	Value	Units	Verification Method
FUNCTIONAL REQUIREMENTS					
1	Include relevant and current policy issues / interactive elements	≈	two	Interactive elements	Inspection
2	Implement locomotion so that users can move around a scene	=	yes	N/A	Testing
3	Aim for maximum of 30 secs	<	30	seconds	Demonstration

CONSTRAINTS					
1	Choose only 1-2 concerns to streamline VR focus	≈	two	Killer Robot Concerns	Demonstration
2	Specific technical criteria to measure VR performance 1. Minimal Loading Time 2. High-Quality Resolution	1: < 2: =	1: five 2: 1080p	1: seconds 2: resolution	Analysis
NON-FUNCTIONAL REQUIREMENTS					
1	VR experience should be relatable to a wide variety of users, each with different perspectives	=	yes	N/A	Testing
2	Use interactive elements to keep the user engaged 1. Include human faces 2. Use background music	=	yes	N/A	Testing

Problem Statement: Mines Actions Canada wants to convince policy decision makers to preemptively ban autonomous robots by showcasing, through a curated VR experience, the concerning and devastating effects of these weapons on society.

Our product has four key differentiators.

1) Concise Simulation: Based on initial technical benchmarking from Facebook, we found that most advertisements have a time limit capped to 30 seconds. Our project aims to convince policy makers in a similar way that marketing agents may try to convince a user so we decided to implement the same 30 second time limit.

Demo Video for VR Simulation:
https://drive.google.com/file/d/1ny7M79m-H0tjNH6_rGzGB5IRYV5cOqkW/view?usp=sharing

2) Research Based: Throughout the design process, we focussed on implementing research to decide on the alterations to include in the simulation. For instance, by watching the documentary Immoral Code and by reading an article about race and robots from Stop Killer Robots, we decided to include masks to hide the individual's identity as they leave the house setting[7]. Additionally, we included the idea of dehumanization, which was continuously brought up by the client, by adding different textures to the house setting creating a more deteriorated look.

Textures Added to House Setting

Boarded Windows

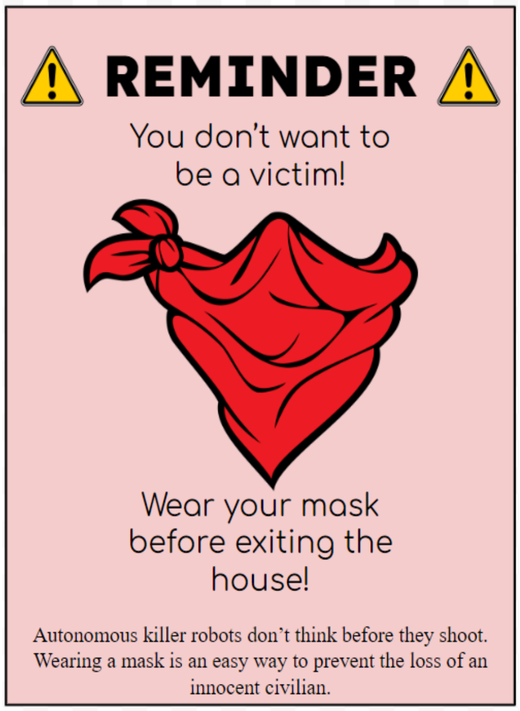
Smashed TV & Toppled Shelves

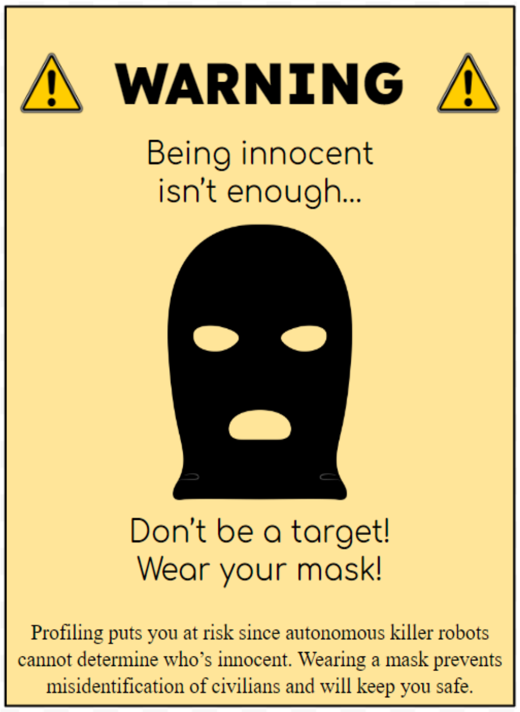
3) Functional & Interactive: To make this VR simulation easy to use and convincing, one of our main priorities was interactivity. Our project includes two interactive elements, which was one of our previously defined project goals. We prioritized the door opening mechanism which would correspond to the end of our simulation. Once the door is opened by a simple click, the simulation ends and the user sees the Mines Action Canada slogan for this project, #Join Team Human. Another interactive element is the bandana which is lifted off the hanger and moves towards the object once the user is at a close proximity. This signifies the user putting on a mask before exiting the house.

Door Opening

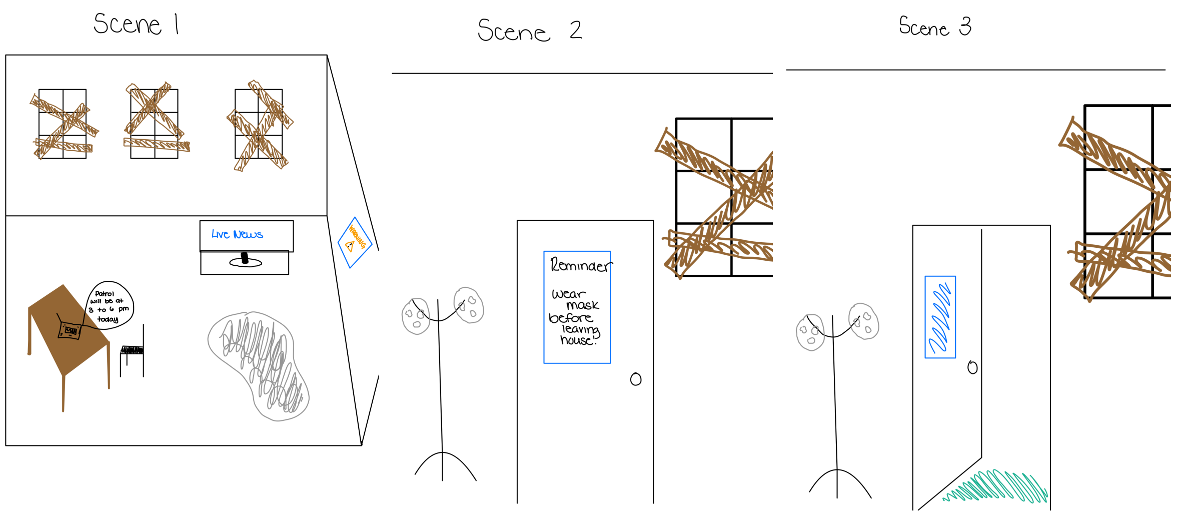
4) Convincing & Unique: Posters have been added to the house environment to signal to the user the effects of autonomous killer robots and the actions that innocent civilians have to take to protect themselves. Furthermore, we decided to plan out our VR as a story, including creating a story board to make the experience more memorable. Another insight from our technical benchmarking was the importance of including sound/audio into a video to make the message more interesting for the viewer, based on a study conducted by the University of Indiana[3]. So, in the VR, we include a radio element that plays a recording of a broadcaster detailing the patrol times.

Posters


REMINDER
You don't want to be a victim!
Wear your mask before exiting the house!
Autonomous killer robots don't think before they shoot. Wearing a mask is an easy way to prevent the loss of an innocent civilian.

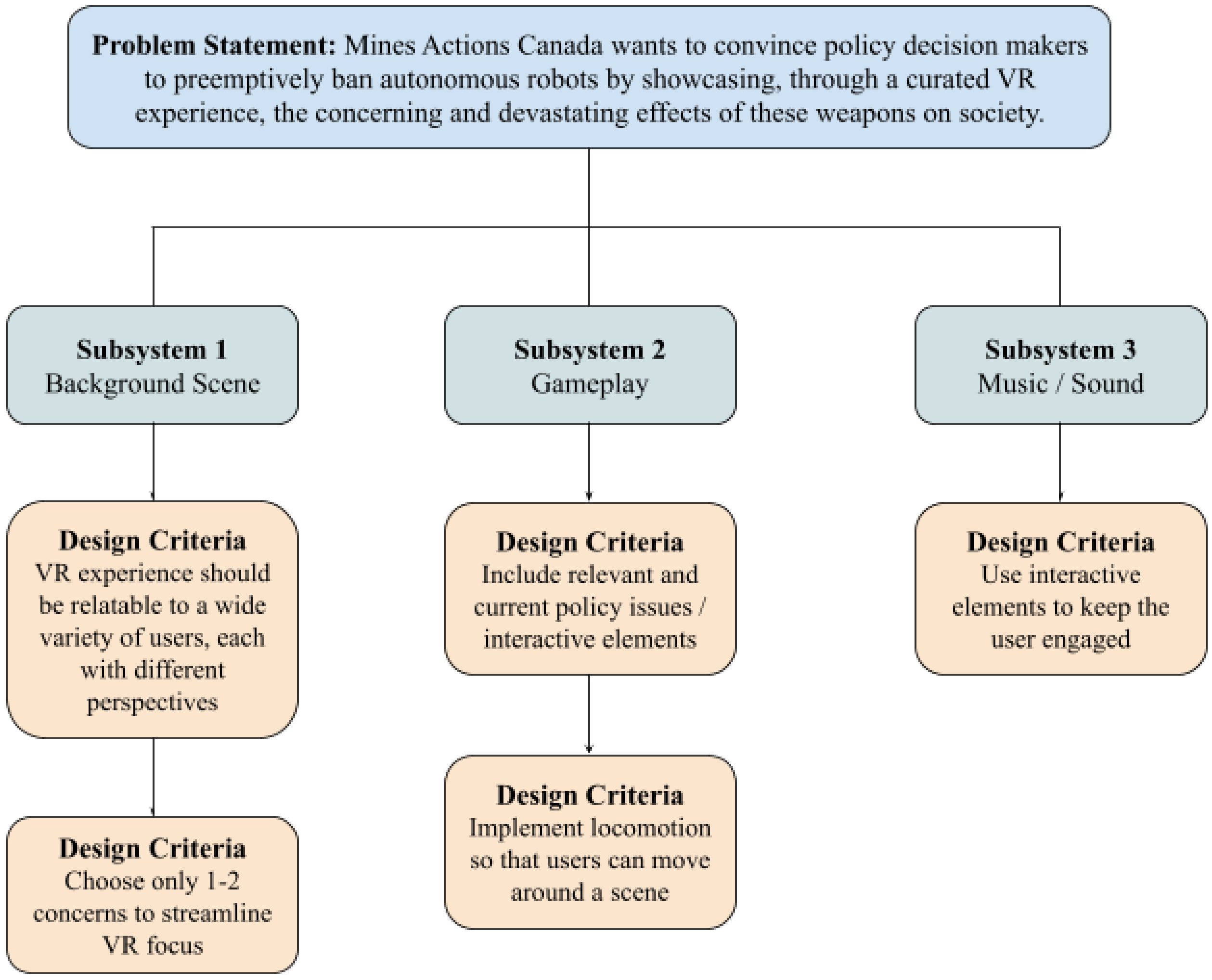

WARNING
Being innocent isn't enough...
Don't be a target! Wear your mask!
Profiling puts you at risk since autonomous killer robots cannot determine who's innocent. Wearing a mask prevents misidentification of civilians and will keep you safe.

Storyboard


Scene 1
Scene 2
Scene 3
Live News
WARNING
Patrol will be at 3 to 6 pm today
Reminder
Wear mask before leaving house.

The main functions of the product are its interactive elements which are the door opening to the outside and the mask moving toward the user to signify someone putting a mask on. Some other key features include the various textures used to show a deteriorated look within the house as well as other design elements like posters, boarded windows, and audio from the radio.

The system was built on unity and the product development was split into three main project goals including interactivity, movement, and realism. These goals also fit within the three subsystems that we initially used to come up with ideas.


Problem Statement: Mines Actions Canada wants to convince policy decision makers to preemptively ban autonomous robots by showcasing, through a curated VR experience, the concerning and devastating effects of these weapons on society.
Subsystem 1 Background Scene
Subsystem 2 Gameplay
Subsystem 3 Music / Sound
Design Criteria VR experience should be relatable to a wide variety of users, each with different perspectives
Design Criteria Include relevant and current policy issues / interactive elements
Design Criteria Use interactive elements to keep the user engaged
Design Criteria Choose only 1-2 concerns to streamline VR focus
Design Criteria Implement locomotion so that users can move around a scene

2.1 Conventions

There are no particular conventions associated with this product report.

2.2 Cautions & Warnings

Our prototype is fairly safe as it is a software application which cannot pose any physical risk to the user. The potential risks of our prototype are associated with using the VR technology that our prototype runs on. The largest and most common risk of using our VR prototype is the risk of developing motion sickness. This is believed to be caused by high latency in VR technology. If the user starts to experience motion sickness we advise that they take off the VR headset, sit down and drink water. We warn against using our VR prototype if the user is known to be susceptible to seizures as the VR screens can be triggering to people with a history of seizures.

3 Setup

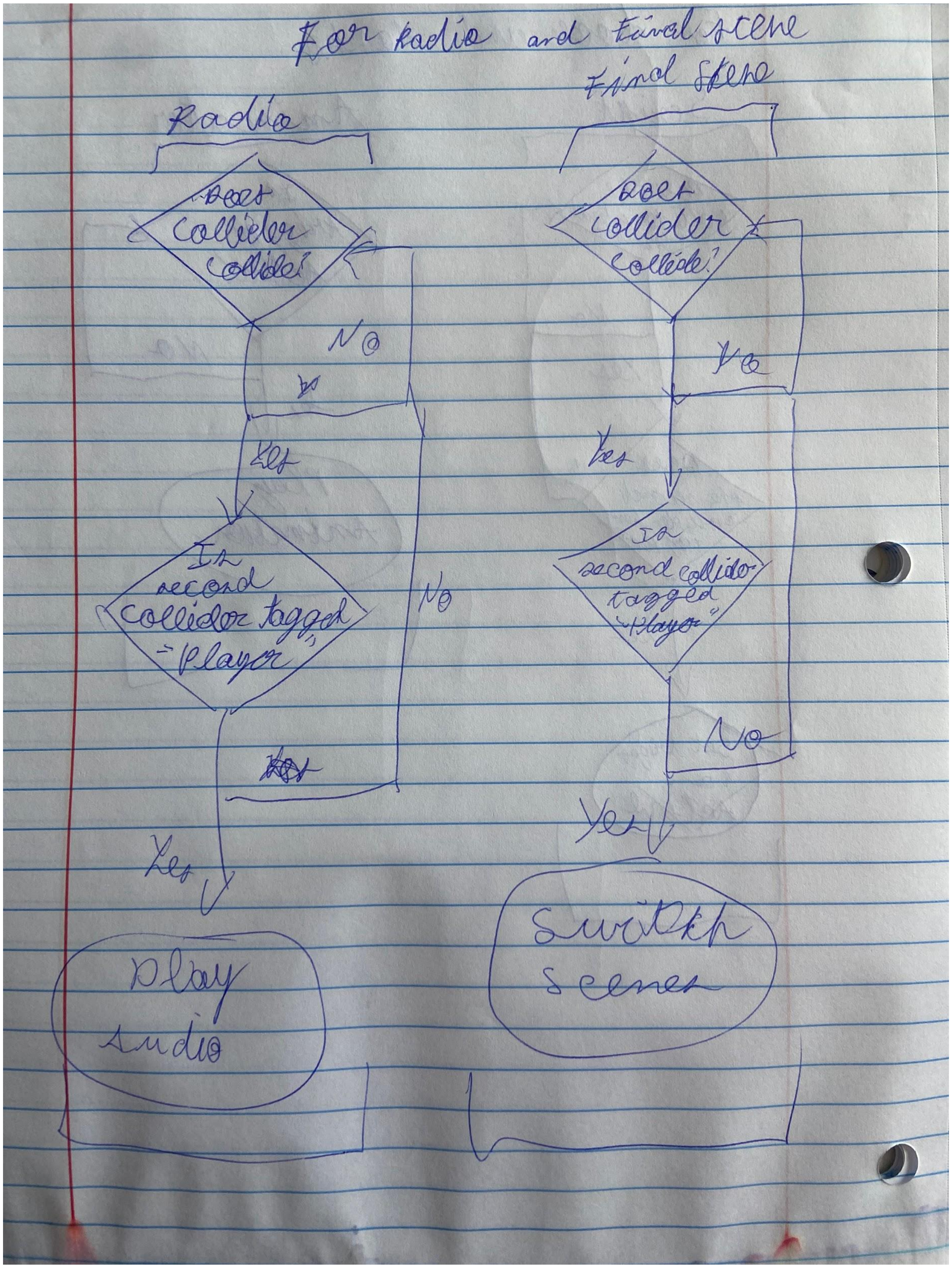
This section will describe the process that we advise users to follow when they set up and put away our system. This section will describe the best practice for a user entering, setting up and exiting the system.

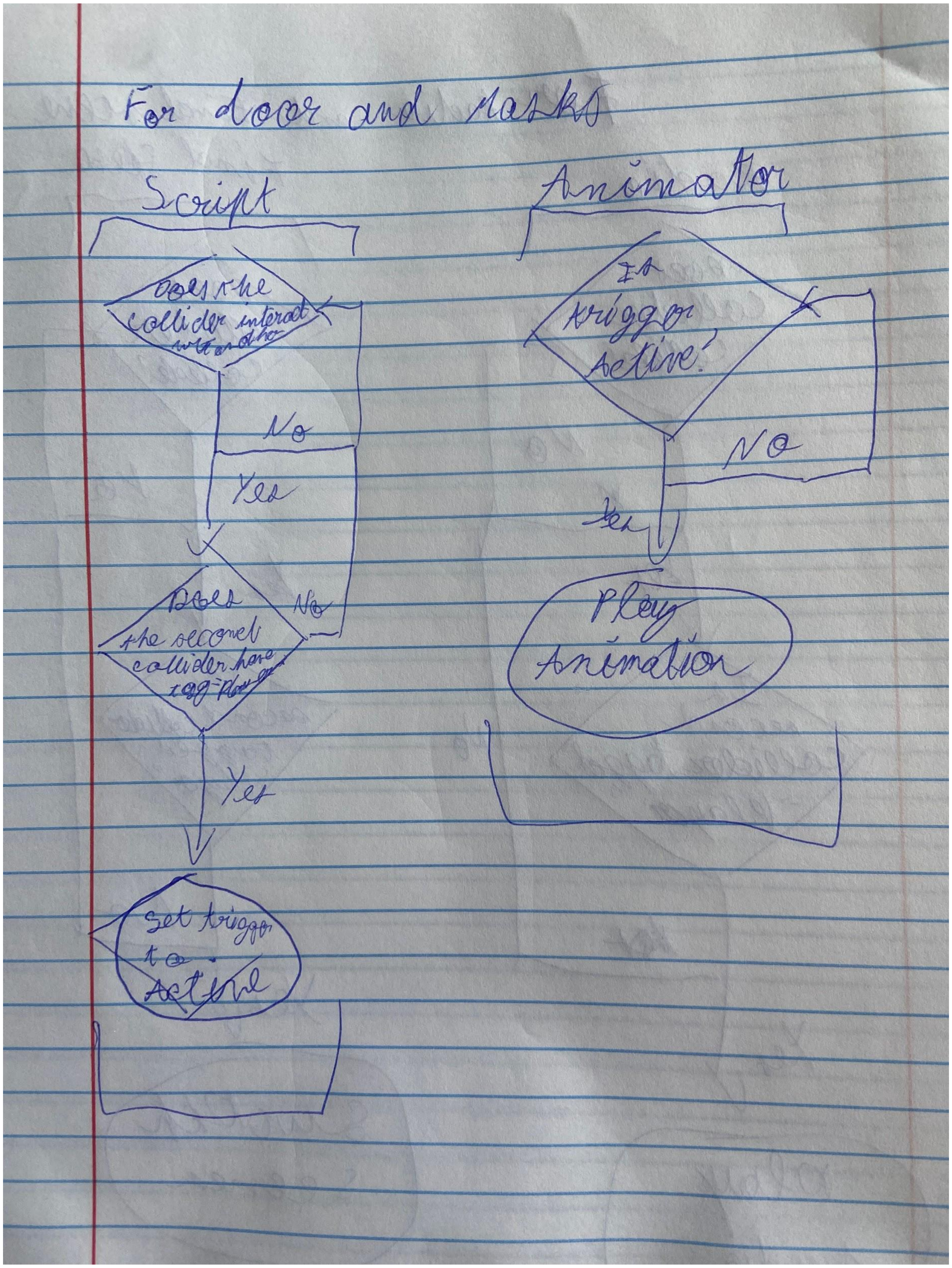
3.1 Configuration Considerations

Input: Within the simulation there are two main points that may require input, the camera movement and the interactive elements. For the movement, all the inputs are handled by the VR headset, which uses a complex system of motion tracking and handheld remotes as inputs. The interactive elements are far more simple however. The scripts receive an input when the player's collider intersects with that of the relevant object. In the case of the bandanas and door, there is a second layer in the form of the animation controller, which receives an input of the trigger being set to active.

Output: For the movement, the outputs essentially mimic the user's inputs. The camera's rotation directly follows the user's movement, while the actual displacement responds to the joysticks movement on the handheld remote. For the interactive elements, the outputs are each different. The radio plays an audio clip, the bandana and door play an animation, and the exit switches the scene to show the designed User Interface.

Diagrams


For Radio and Final scene
Radio
Does collider collide?
No
Yes
Is second collider tagged "Player"
No
~~Yes~~
Yes
Play audio
Final scene
Does collider collide?
No
Yes
Is second collider tagged "Player"
No
Yes
Switch scenes


For door and tasks
Script
Animator
Does the collider interact with another
No
Yes
Does the second collider have tag = Player
No
Yes
Set trigger to Active
Is trigger Active?
No
Yes
Play Animation

3.2 User Access Considerations

The users for our prototype are not restricted, anyone who is interested in learning about the effects of Autonomous Killer Robots on society could use our prototype. The VR simulation is accessible to anyone who has the Unity build file downloaded to their desktop. This acts as a restriction as if you don't have the file on your computer you will not be able to access the prototype. In addition, the file that must be downloaded is only compatible with certain computers. As such, if the user has a MacBook they would not be able to access the file. Another restriction is if the user does not have access to a VR headset. Without a headset the user would not be able to fully experience the VR simulation. It is also important to note that the game file will not play without a headset.

3.3 Accessing/setting up the System

Once the game is downloaded onto the user's laptop they have to open up the file on their desktop that is named "build". Once the folder is opened the user will have to access the game through the file named "Team 5".

In order to connect the game to the headset the user will have to have the Oculus app downloaded to their laptop, this is not compatible with MacBooks. The app will require the user to create an account and they will simply follow the steps that show up on the screen. Once the app is set up the user will have to connect the app to an Oculus Quest 2 headset and follow the instructions displayed in the headset. When the VR headset is connected to the laptop the game will automatically start and the user will be able to freely move around the simulation.

3.4 System Organization & Navigation

Initial View of Simulation: Upon putting on the VR headset, the user will see boarded windows within the house looking out into a city landscape.

Moving Around with Oculus Headset: The Oculus Headset is connected to two joysticks. These joysticks allow the user to move around the simulation. The user is able to move in any direction as long as it is between the boundaries of the simulation.

Towards the Radio: To move towards the radio the user must face in the direction of the radio and use the joystick to move forwards.

Towards the Masks: In order to move towards the masks the user must turn towards the television, walk forward, turn towards the mask and walk to the tile on the floor.

Looking Around: The headset is also able to turn in the direction of the user, if the user turns right the simulation will be turned right as well.

Exiting the House: In order to exit the house the user must move towards the tile on the floor. This will then cause the door to open and the user can walk into the door frame. This causes the user to exit the house and the words "Join #TeamHuman" will appear.

3.5 Exiting the System

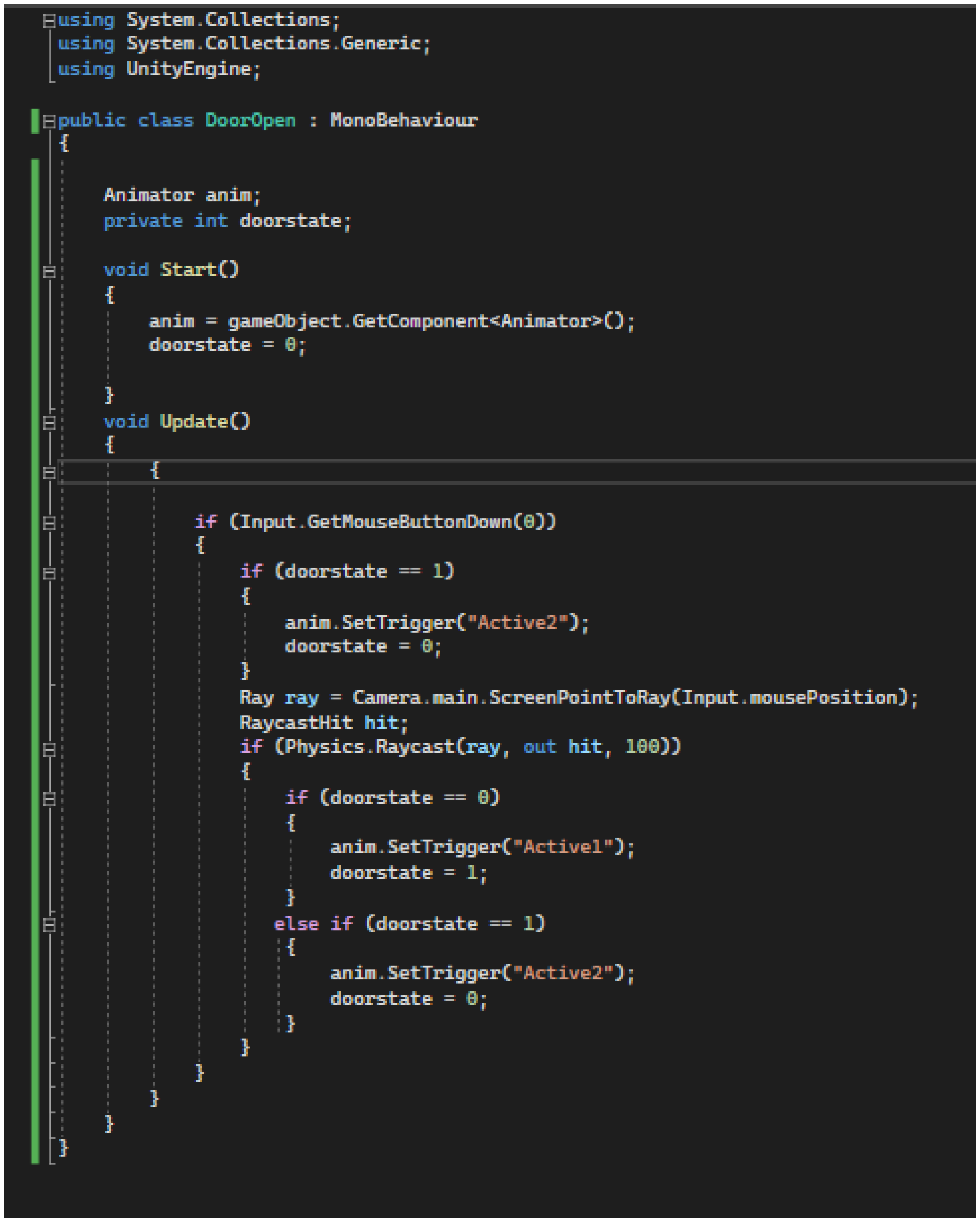
In order to properly exit the simulation when the VR headset is in operation the user must click the exit simulation button on the headset. This would then close the simulation on the laptop as well. Another method of exiting the system is simply by removing the headset. In order to then close the simulation on the laptop the user would have to press the windows key on their laptop and then close the tab.

4 Using the System

This section describes the process that we advise users to follow once they have set up our system and before they turn off our system. This section will describe the interactive elements of our system. There are two main interactive elements present in our simulation consisting of the opening of the door within the house and the user "picking" up the mask which is represented by the mask moving towards the user. Another user interaction with the system is indirect, but the proximity of the user to the radio triggers the audio to begin playing. For this part of the technical report, we will detail only the door opening mechanism and the movement code. The mask and radio interactions can be found in 6.1.3 Instructions.

4.1 Door Opening

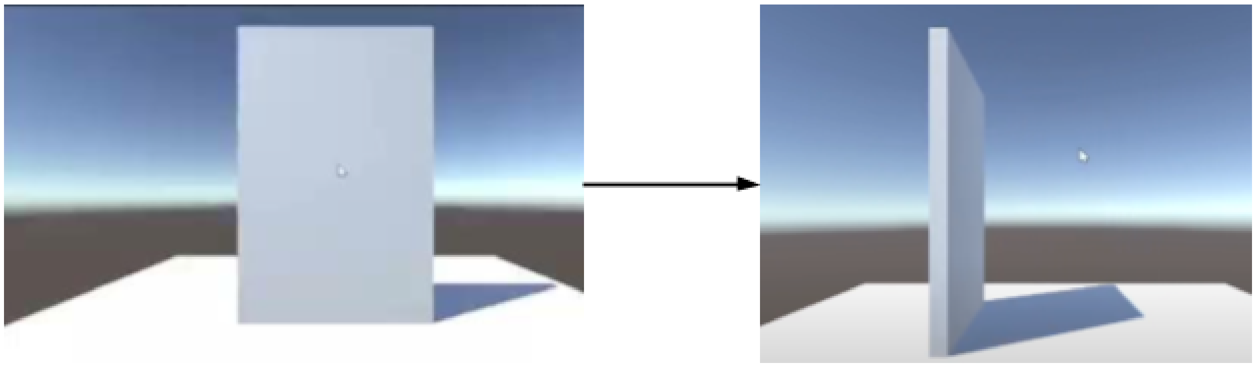
The code we used for the door opening is as follows:



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

public class DoorOpen : MonoBehaviour
{

    Animator anim;
    private int doorstate;

    void Start()
    {
        anim = gameObject.GetComponent<Animator>();
        doorstate = 0;

    }
    void Update()
    {
        {

            if (Input.GetMouseButtonDown(0))
            {
                if (doorstate == 1)
                {
                    anim.SetTrigger("Active2");
                    doorstate = 0;
                }
                Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
                RaycastHit hit;
                if (Physics.Raycast(ray, out hit, 100))
                {
                    if (doorstate == 0)
                    {
                        anim.SetTrigger("Active1");
                        doorstate = 1;
                    }
                   else if (doorstate == 1)
                    {
                        anim.SetTrigger("Active2");
                        doorstate = 0;
                    }
                }
            }
        }
    }
}
```

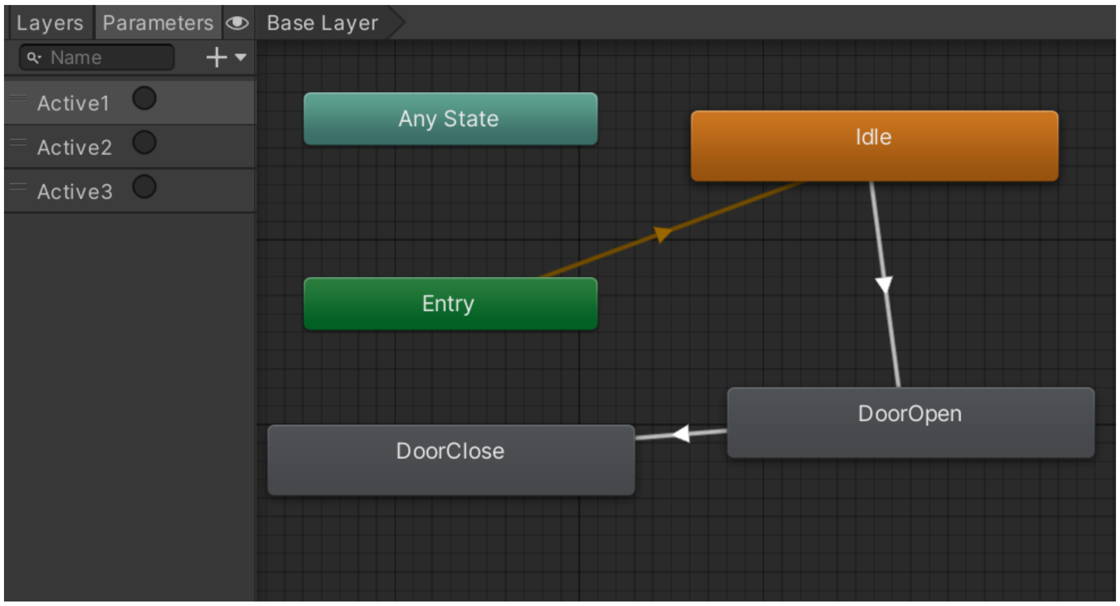
The “anim” syntax grabs the animations that we’ve made in unity (see subsection 4.1.1) and the “get mouse down” gathers inputs from the mouse. Once the mouse is clicked, a loop starts and the Ray ray line gathers the position of the mouse whilst the raycast function finds out if the click was on the object. From there the if-else loop determines the doorstate (1 being open 0 being closed). Once the doorstate is determined the correct animation is triggered and then the doorstate is altered.

Image of Door Opening (Aligns with Prototype 1)



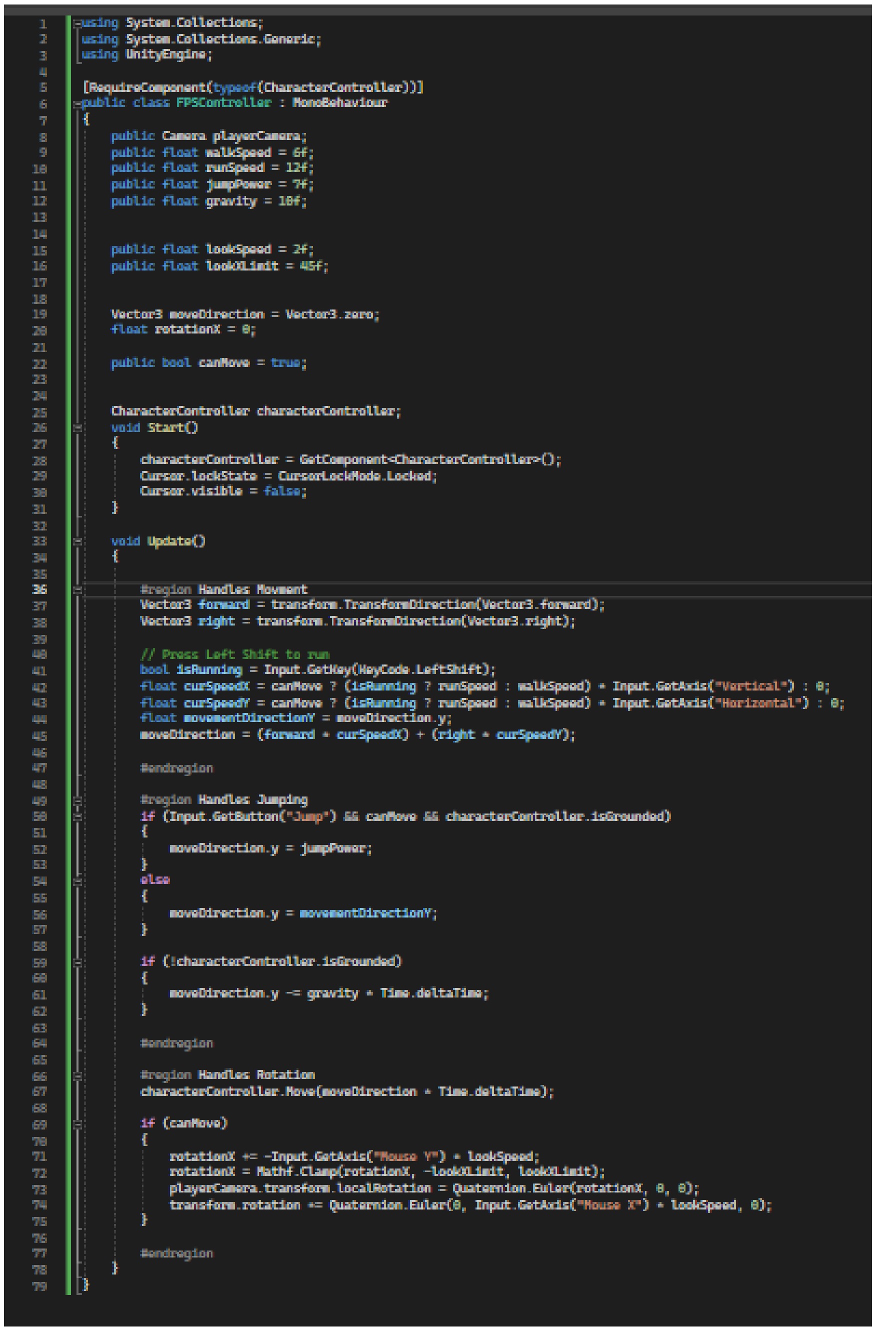
4.1.1 Animations Made for Door Opening Interaction

Each of the boxes below represents an animation which we have made in Unity. The vectors in between the boxes represent transitions so the animations can flow one into another. The transitions are dependent on triggers from the user and they are shown in the top left “Active1…” The triggers act as a loop. At first the door is closed but once the Active1 trigger is activated the animation starts and the door opens. Now that the door is open the system is waiting on another input from the user to trigger Active2. Once Active2 is triggered the door closes.



4.2 Movement

The code we used for movement is as follows:



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

[RequireComponent(typeof(CharacterController))]
public class FPSController : MonoBehaviour
{
    public Camera playerCamera;
    public float walkSpeed = 6f;
    public float runSpeed = 12f;
    public float jumpPower = 7f;
    public float gravity = 10f;


    public float lookSpeed = 2f;
    public float lookXLimit = 45f;


    Vector3 moveDirection = Vector3.zero;
    float rotationX = 0;

    public bool canMove = true;


    CharacterController characterController;
    void Start()
    {
        characterController = GetComponent<CharacterController>();
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }

    void Update()
    {

        #region Handles Movment
        Vector3 forward = transform.TransformDirection(Vector3.forward);
        Vector3 right = transform.TransformDirection(Vector3.right);

        // Press Left Shift to run
        bool isRunning = Input.GetKey(KeyCode.LeftShift);
        float curSpeedX = canMove ? (isRunning ? runSpeed : walkSpeed) * Input.GetAxis("Vertical") : 0;
        float curSpeedY = canMove ? (isRunning ? runSpeed : walkSpeed) * Input.GetAxis("Horizontal") : 0;
        float movementDirectionY = moveDirection.y;
        moveDirection = (forward * curSpeedX) + (right * curSpeedY);

        #endregion

        #region Handles Jumping
        if (Input.GetButton("Jump") && canMove && characterController.isGrounded)
        {
            moveDirection.y = jumpPower;
        }
        else
        {
            moveDirection.y = movementDirectionY;
        }

        if (!characterController.isGrounded)
        {
            moveDirection.y -= gravity * Time.deltaTime;
        }

        #endregion

        #region Handles Rotation
        characterController.Move(moveDirection * Time.deltaTime);

        if (canMove)
        {
            rotationX += -Input.GetAxis("Mouse Y") * lookSpeed;
            rotationX = Mathf.Clamp(rotationX, -lookXLimit, lookXLimit);
            playerCamera.transform.localRotation = Quaternion.Euler(rotationX, 0, 0);
            transform.rotation *= Quaternion.Euler(0, Input.GetAxis("Mouse X") * lookSpeed, 0);
        }

        #endregion
    }
}
```

The movement feature was mainly used for testing when we weren't able to access the VR headset for our product. A public class was used to facilitate creating a code that would allow the character to rotate and move based on cursor movement and certain keys. The user is represented by an empty game object that can move in the virtual space. Once the code is written, it is added to the player as a script which can be updated after each test. The code starts by initializing the user speed, making sure that the speeds are such that there are smooth transitions between different directions and movement. Changes in character position in the x or y direction are controlled, through if-else loops by the state of one of the components accessible in Unity (characterController) or the state of a boolean (canMove). The actual change in movement/direction is controlled by various variables, taking into account direction (by WASD input handling), cursor speed, axis of movement, and position in vector space.

5 Troubleshooting & Support

In general our system is fairly streamlined and does not experience many errors. The errors that users have run into during testing are usually caused by factors outside of our control like poor Oculus setup and Oculus software bugs. The following subsections will address more specific details of our troubleshooting and support process.

5.1 Error Messages or Behaviors

There are no error messages associated with our VR prototype when it is in use. The only error messages associated with our system are on the back end and are related to the Unity compiler that our system was coded on. The error messages in the compiler are traditional C compiler error messages and the user should reference the Unity user manual to find out more about their error messages. If the Oculus hardware displays any error messages we encourage the user to reference the Oculus user manual. If our VR system is not working the way that the user intends or expects and none of the previously listed options have helped we encourage them to reach out to us with their question by following the process outlined in 5.4 or reset our system and start again.

5.2 Special Considerations

There are no special considerations for troubleshooting.

5.3 Maintenance

There is no maintenance associated with our prototype specifically as it is a fully virtual product. Reference the Oculus user manual to find more information on maintaining the Oculus hardware.

5.4 Support

If the user experiences any troubles with our system we encourage them to email Ava Butts at abutt011@uottawa.ca. Please include the phrase “Blueprint Brigade VR simulation” in the subject line and expect a response in 24 to 72 hours.

6 Product Documentation

This section describes how the prototype was built and developed into a VR simulation, with detailed insights provided into the asset selection, costs, and construction in Unity. The equipment list covers the various assets used through the Unity database whilst the instructions section delves into building the simulation environment from altering and combining assets to coding interactive components. Throughout the process, rigorous testing was done to collect user feedback on the effectiveness of intermediate prototypes and the results of qualitative and quantitative tests are outlined.

6.1 Final Prototype

Here is a Demo Video for our final VR Simulation:

https://drive.google.com/file/d/1ny7M79m-H0tjNH6_rGzGB5IRYV5cOqkW/view?usp=sharing

The following subtopics will go into what went into the prototype and how it was built on Unity.

6.1.1 BOM (Bill of Materials)

Assets	**Realistic Interior Volume 1**	**Bandana Face Masks**	**Radio**	**Plank Textures**	**One-Click Wall Materials**
Description	Realistic House Interior	Different Styled Bandana Masks	Realistic Radio	Variety of wood textures	Variety of wall textures
Units of Measure	Each	Each	Each	Each	Each
Quantity	1	1	1	1	1
Unit Cost	$35.50	$6.74	$0.00	$0.00	$0.00
Tax	$3.25	$0.88	$0.00	$0.00	$0.00
Extended Cost	$37.75	$7.62	$0.00	$0.00	$0.00
Link	https://assetstore.unity.com/packages/3d/props/interior/realistic-interiors-vol1-148120	https://assetstore.unity.com/packages/3d/props/clothing/accessories/bandana-face-masks-162265	https://assetstore.unity.com/packages/3d/props/radio-230712	https://assetstore.unity.com/packages/2d/textures-materials/wood/plank-textures-pbr-72318	https://assetstore.unity.com/packages/2d/textures-materials/one-click-wall-materials-117123
Total Cost (without taxes)	$41.24				
Total Cost (with taxes)	**$45.37**				

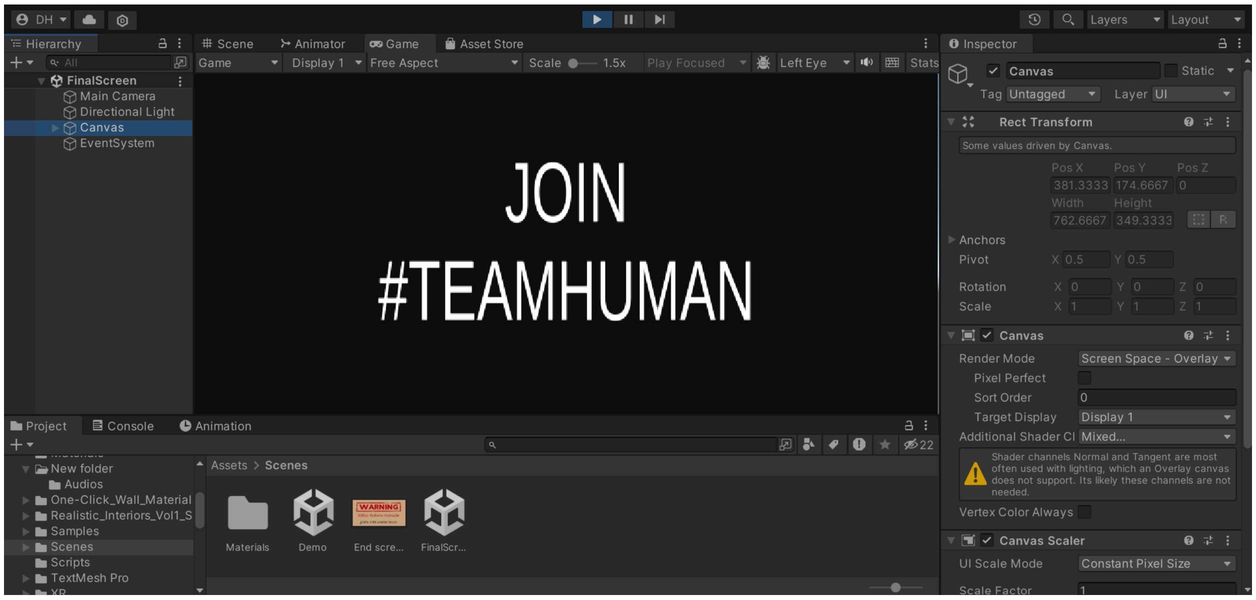
6.1.2 Equipment list

Equipment	Type
Unity	Software
Oculus Meta Quest 2	Hardware
Oculus app	Software
Microsoft Visual Studio	Software

6.1.3 Instructions

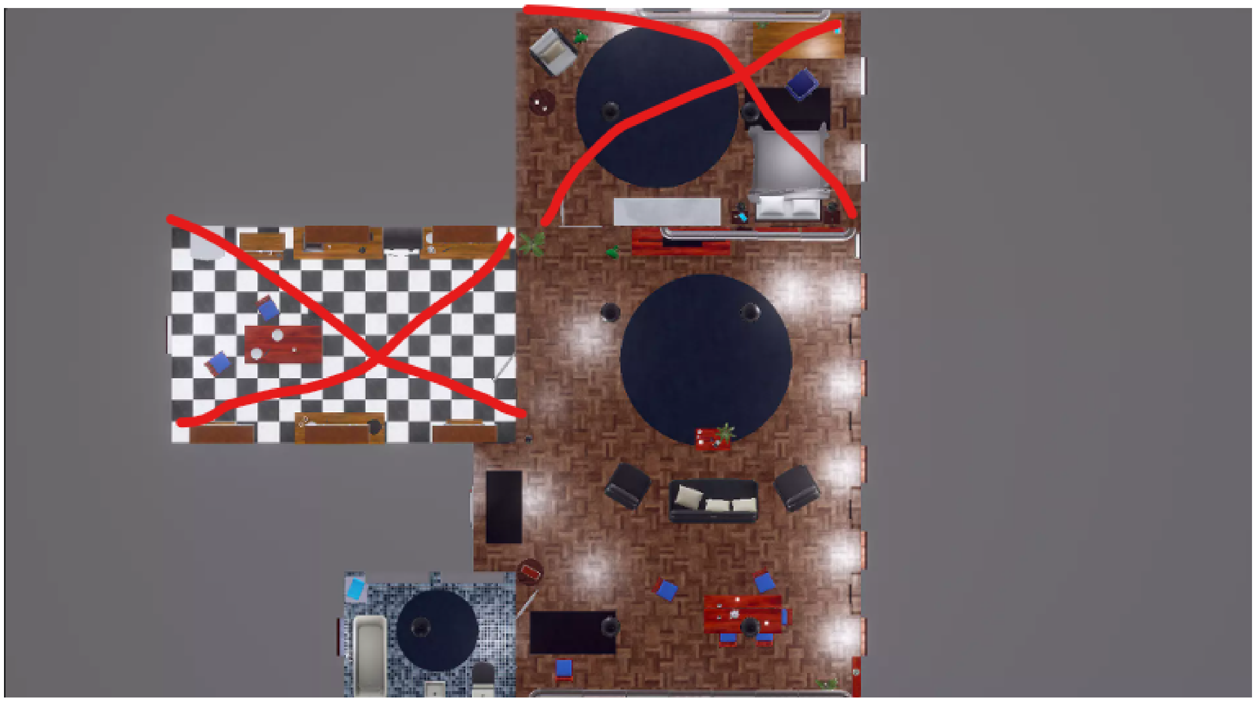
Assets and environment

Before even developing the main apartment, the end screen of the game can be developed. This is done by making a second scene, which only contains a UI with a black background and white text.

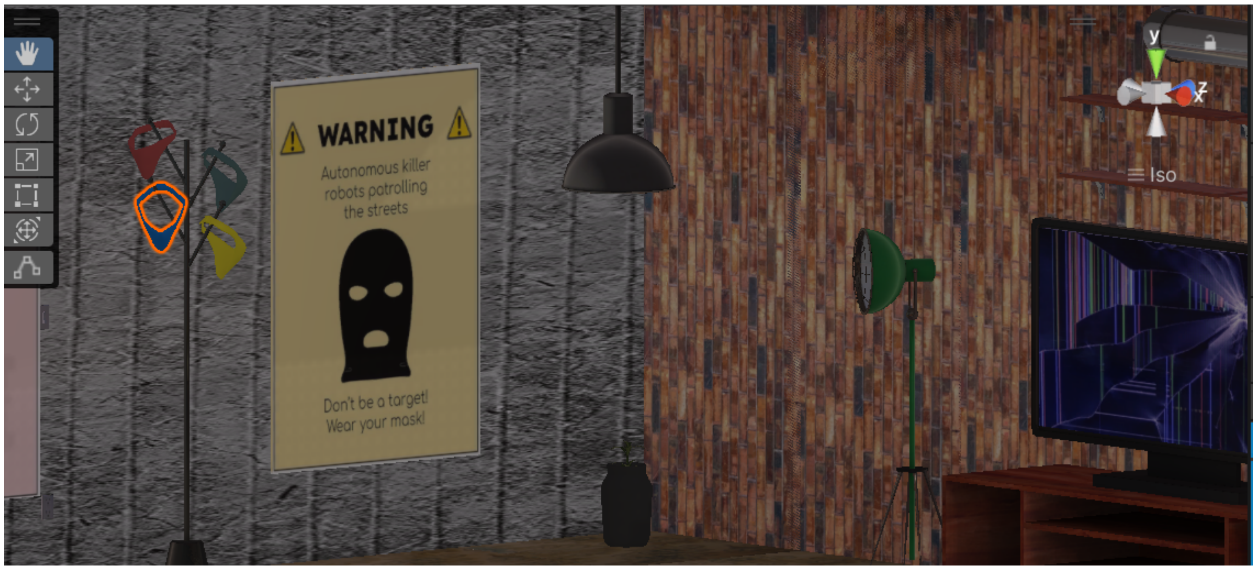


Canvas
Tag Untagged Layer UI
Rect Transform
Some values driven by Canvas.
Pos X 381.3333 Pos Y 174.6667 Pos Z 0
Width 762.6667 Height 349.3333
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas
Render Mode Screen Space - Overlay
Sort Order 0
Target Display Display 1
Additional Shader Cl Mixed...
Shader channels Normal and Tangent are most often used with lighting, which an Overlay canvas does not support. Its likely these channels are not needed.
Canvas Scaler
UI Scale Mode Constant Pixel Size
Scale Factor 1

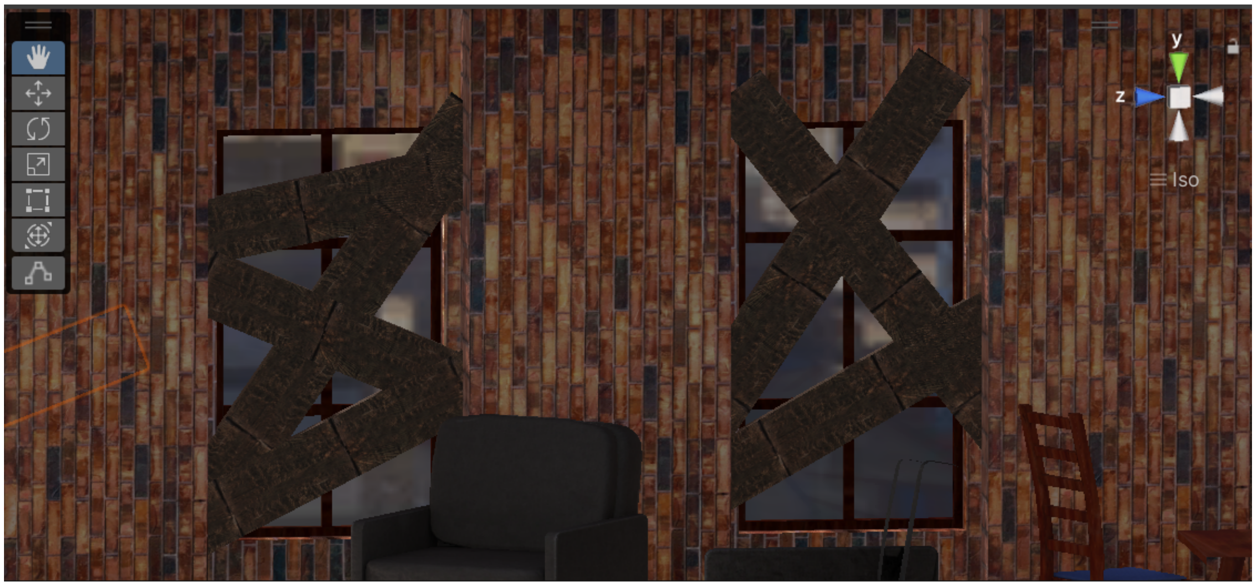
To build the environment of the simulation, we began by searching for relevant assets that best resemble our sketches and storyboards. We divided the components we were looking for into three categories, the general setting of the apartment, textures to modify the atmosphere, and the specific elements to be used in interactions such as the masks and radio. After searching the unity store, we found the 5 assets presented on the BOM, which are then used to develop the setting of the simulation. As the main foundation of the simulation, we began by downloading the apartment asset. From here, we deleted a large part of the apartment to just have the main living room and exit.



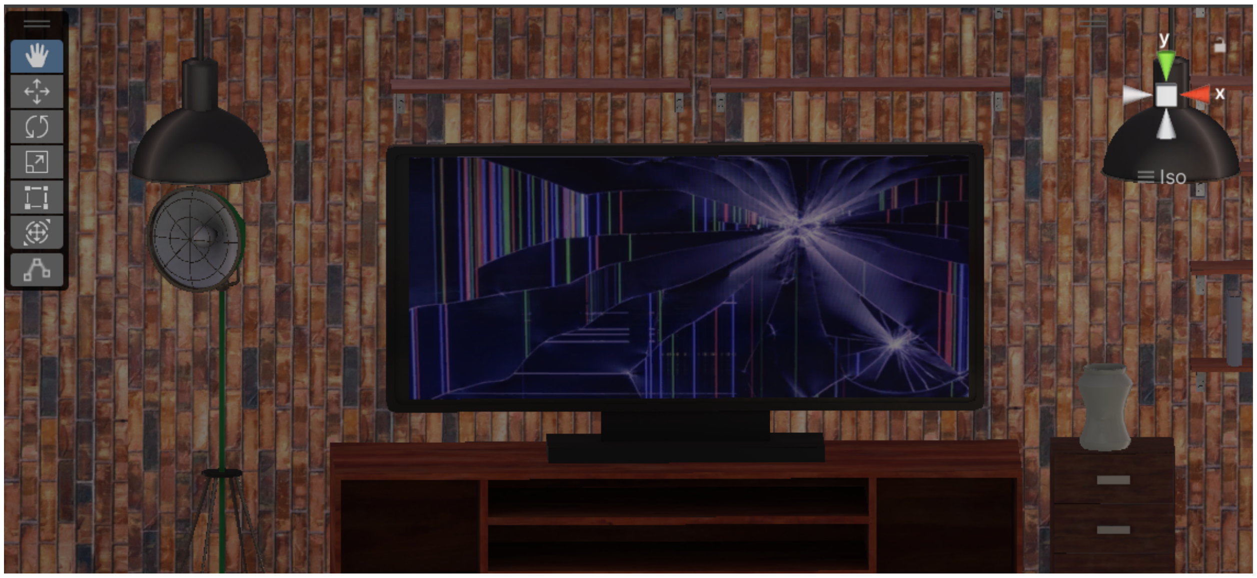
Next, certain wall assets are duplicated then shifted to cover some gaps in the simulation. Finally, 3D colliders are added to each of the elements, so that the player does not phase through the objects as they walk into them. With the base assets in place, we then went to adjust the atmosphere of the environment. This was done by simply applying the assets from the "One click wall materials" to the floors, ceilings and walls.


WARNING
Autonomous killer robots potrolling the streets
Don't be a target! Wear your mask!
y
z
x
Iso

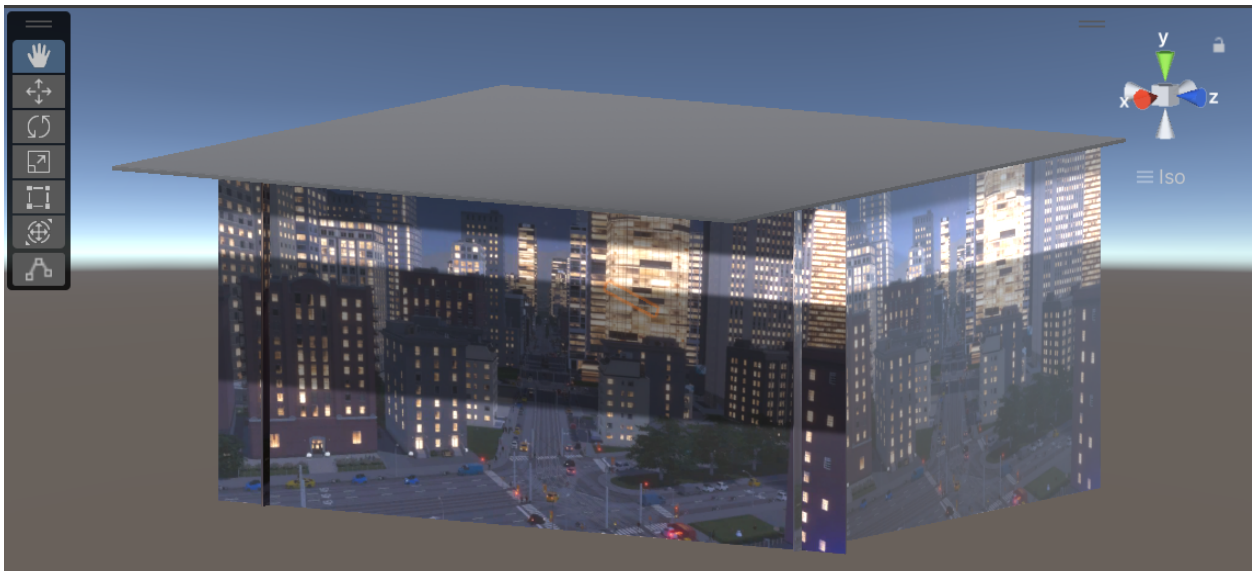
From here, we went to board up the windows, which was done by simply making cubes on which we applied a texture from the “Plank Textures” asset pack. These planks are then rotated on the X axis and displaced arbitrarily to add to the apocalyptic atmosphere already established by the texture pack.



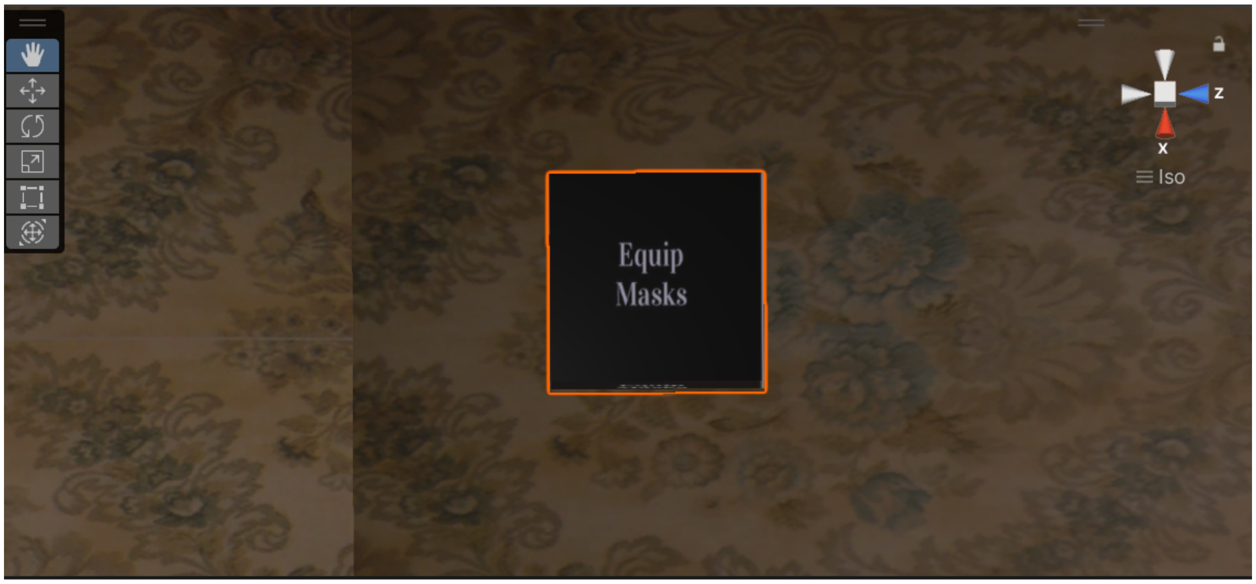
A few elements are then added without the use of asset packs. The first is the broken television screen, which was accomplished by simply adding an image of a broken screen as a material to a very thin cube, then placing this cube over the screen of the television



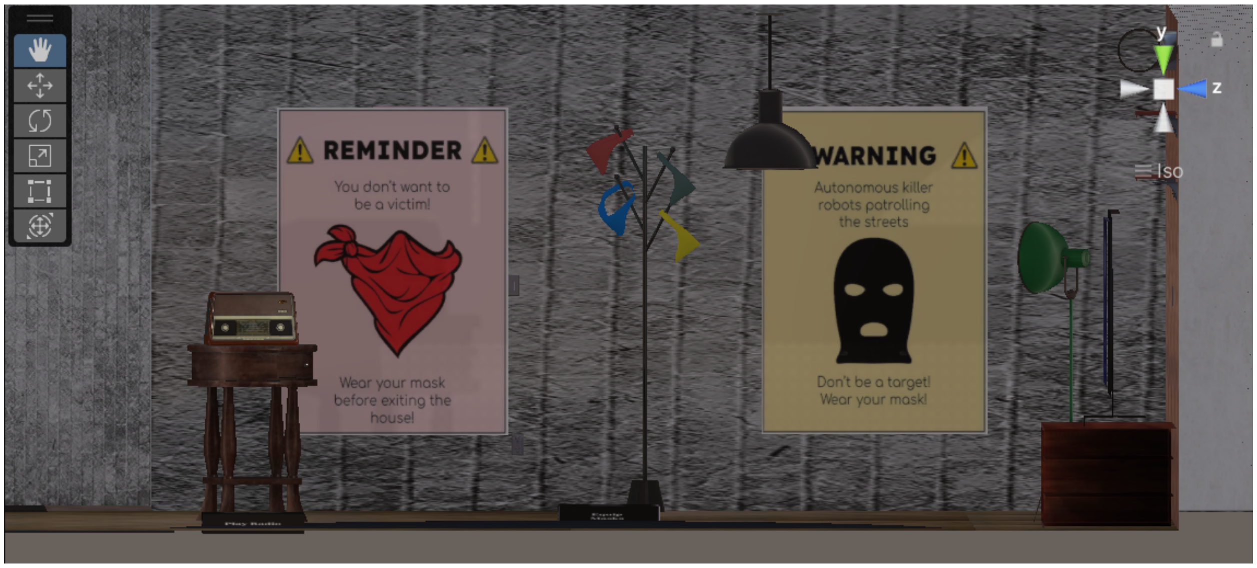
Using a similar technique, the exterior of the apartment is set up by surrounding the room with large thin cubes with an image of a cityscape used for the material. Then, the sky is made by adding another cube with a black material applied to it.



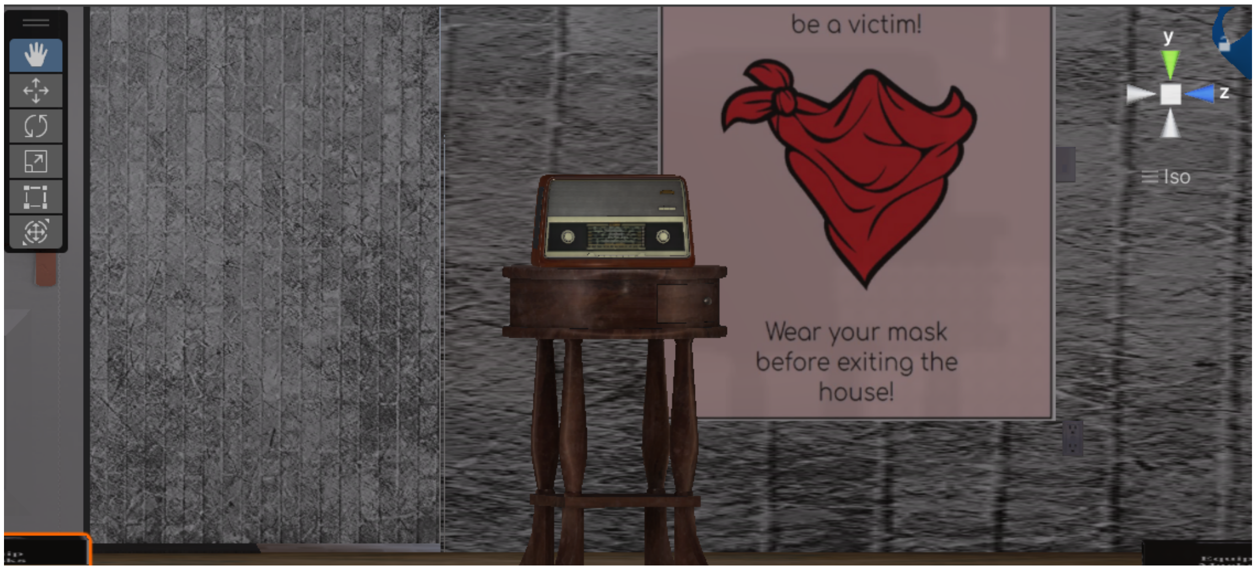
The "Interactive tiles" also use images as materials, these simply being text over a black backdrop prepared in any graphic design application.

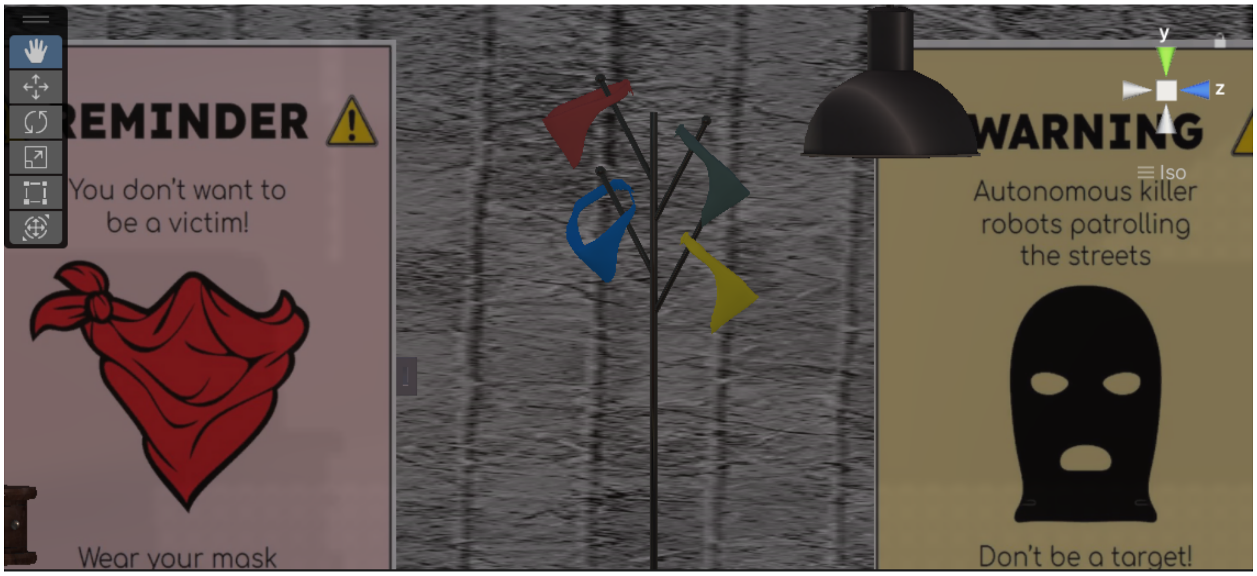


The final use of this technique helped integrate our posters into the simulation, by simply adding the PNGs of the posters to thin cubes as a material.


REMINDER
You don't want to be a victim!
Wear your mask before exiting the house!
WARNING
Autonomous killer robots patrolling the streets
Don't be a target! Wear your mask!
Iso
y
z

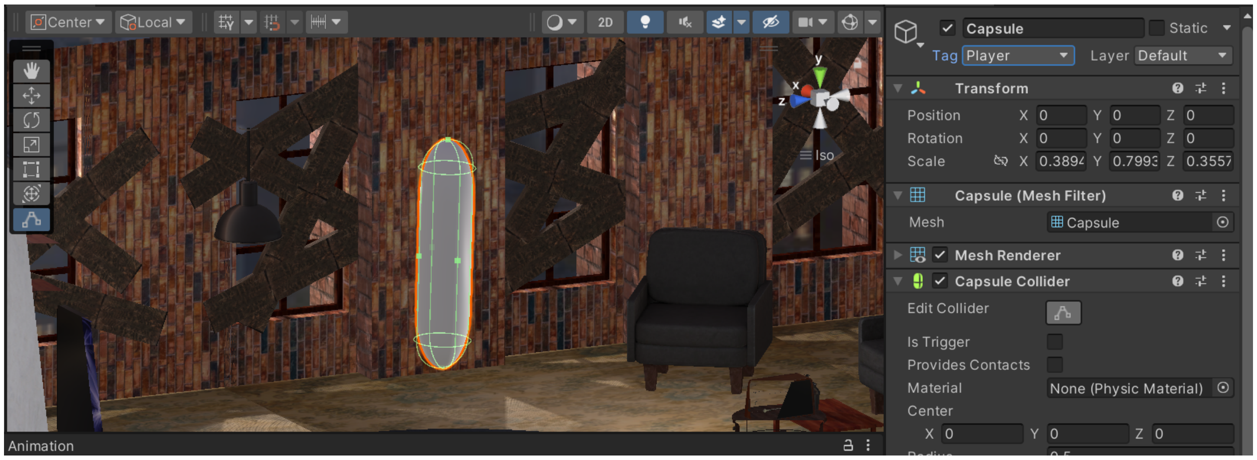
Finally, the interactive objects can be brought into the simulation. For the radio, we found that the apartment asset already contained a radio element, so we decided to simply use that one rather than importing a new asset. The radio and the table it sits on were displaced to better direct the flow of the simulation. Similarly the coat hanger in the apartment asset was shifted to between the posters as to eventually hold the masks. These were then brought in and placed on the coat hanger, and a flat color material was placed on the objects to make them more simplistic.




REMINDER
You don't want to be a victim!
Wear your mask
WARNING
Autonomous killer robots patrolling the streets
Don't be a target!
Iso

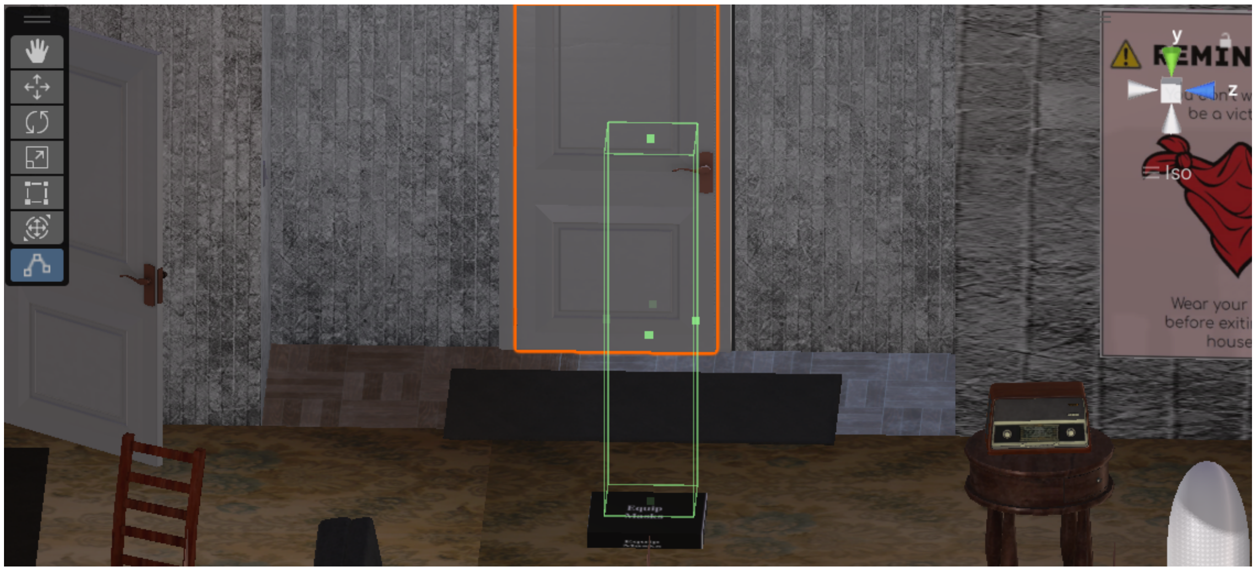
Programming and Interaction

For a product as simple as ours aims to be, the programming work could be divided into two main parts, the movement of the player and the interactions with objects in game. The first part of this is mostly resolved by the Oculus headset and its pre-programmed movement script, saving a lot of work for us. To program the interactive elements, the process begins by editing the player. First a collider must be added to the player so they can come in contact with the objects, then a tag called "Player" is assigned to the camera.

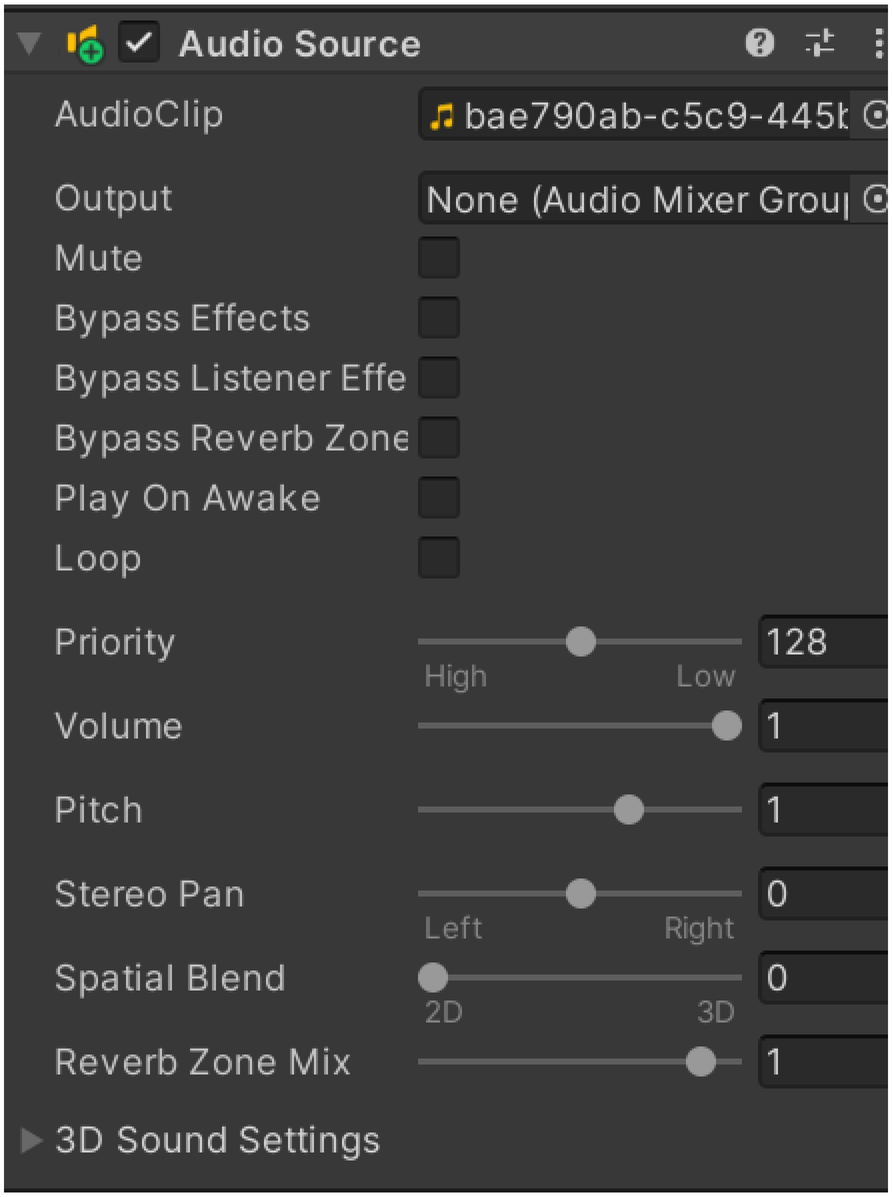


Capsule Static
Tag Player Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 0.3894 Y 0.7993 Z 0.3557
Capsule (Mesh Filter)
Mesh Capsule
Mesh Renderer
Capsule Collider
Edit Collider
Is Trigger
Provides Contacts
Material None (Physic Material)
Center
X 0 Y 0 Z 0

Next, the colliders of each of the interactive objects are manipulated to sit above the interactive tiles, in such a way that if the player steps on the tile, their collider will intersect with that of the object.

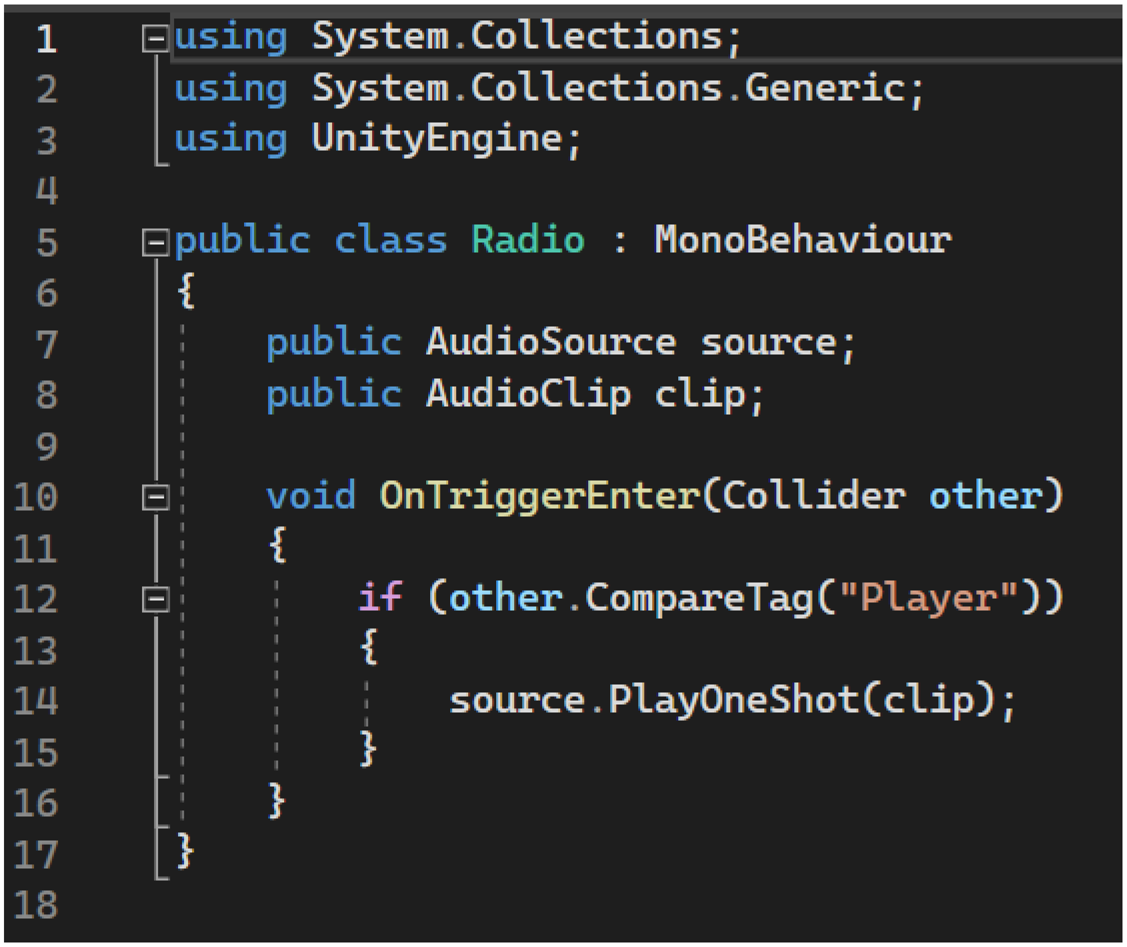


From here, the paths diverge for each of the different objects. For the radio, this can be programmed rather easily. Since we intend for it to play an audio, we first have to assign an audio source to the radio, and an mp3 file of the intended audio is assigned to this source.



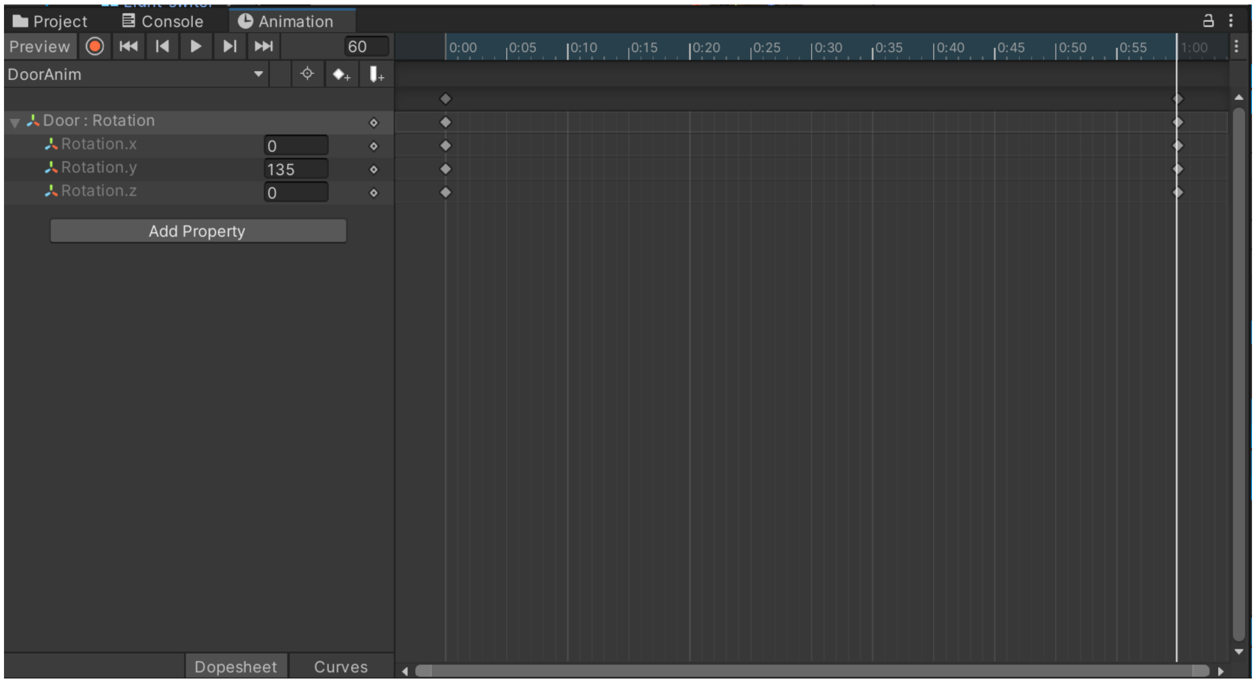
AudioClip	bae790ab-c5c9-445t
Output	None (Audio Mixer Grou
Mute	☐
Bypass Effects	☐
Bypass Listener Effe	☐
Bypass Reverb Zone	☐
Play On Awake	☐
Loop	☐
Priority (High – Low)	128
Volume	1
Pitch	1
Stereo Pan (Left – Right)	0
Spatial Blend (2D – 3D)	0
Reverb Zone Mix	1

Next, we wrote a script that essentially states that if the collider with tag "Player" intersects with that of the object, it will play the audio assigned to the audio source.



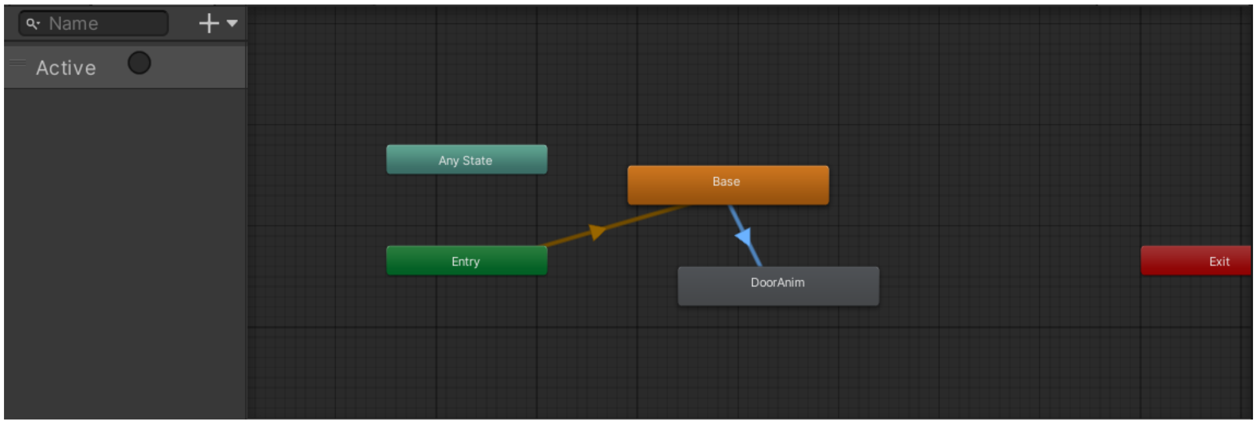
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Radio : MonoBehaviour
{
    public AudioSource source;
    public AudioClip clip;

    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            source.PlayOneShot(clip);
        }
    }
}
```

For both the mask and the door, the process is very similar. Both animations have to be recorded using the Unity animator, by simply beginning a recording and changing the parameters of the location and rotation of the objects to the intended final point. The animator itself then fills in the points in between, and develops a seamless transition between the two points. It must be emphasized to remove a loop time from the animation so it only plays once upon interaction.

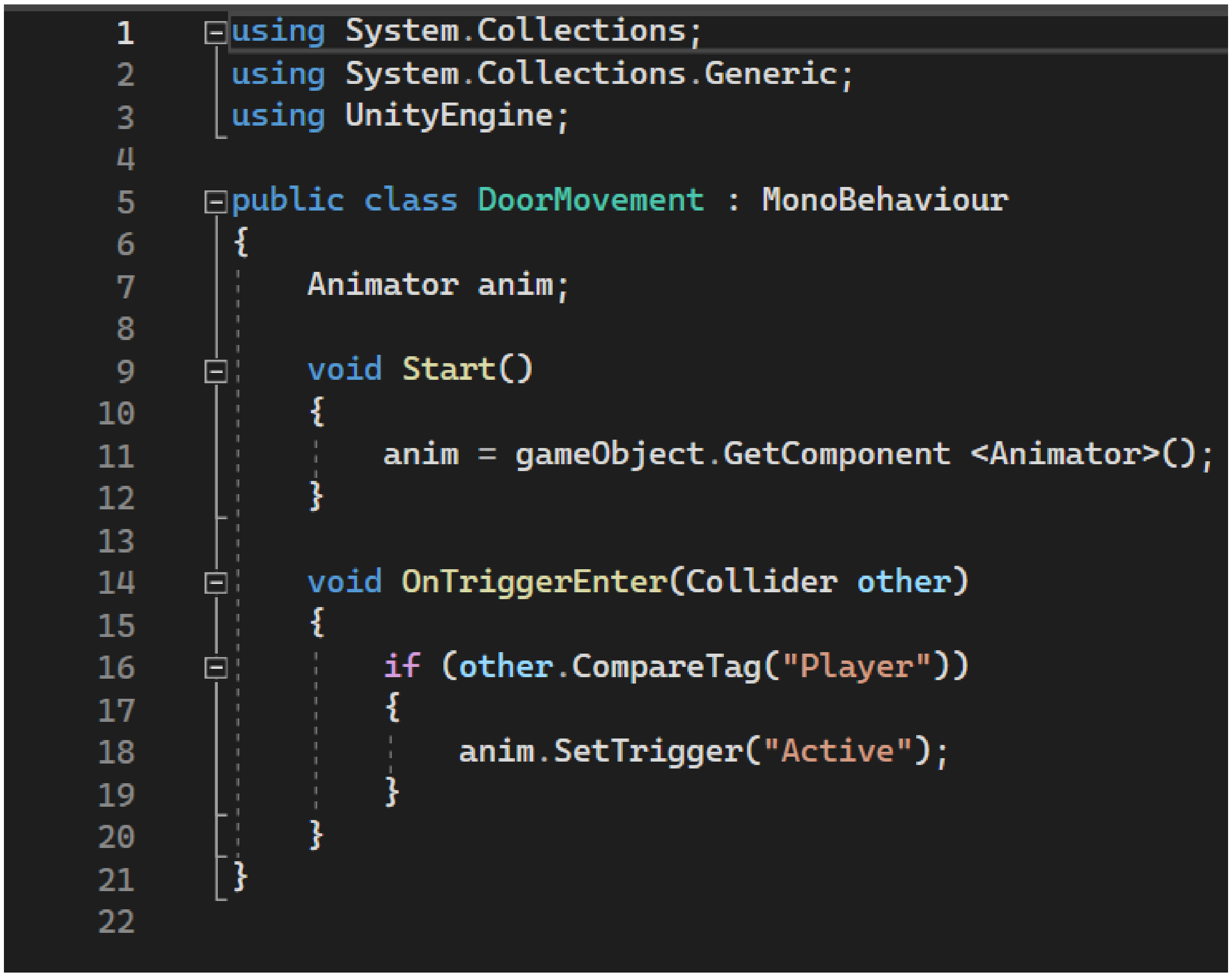


Rotation.x 0, Rotation.y 135, Rotation.z 0

From here, we go to the animation controller and create a new base state so the program does not automatically play the animation when the game is opened. A transition is then formed from the base state to the animation, with a dependency on a trigger.

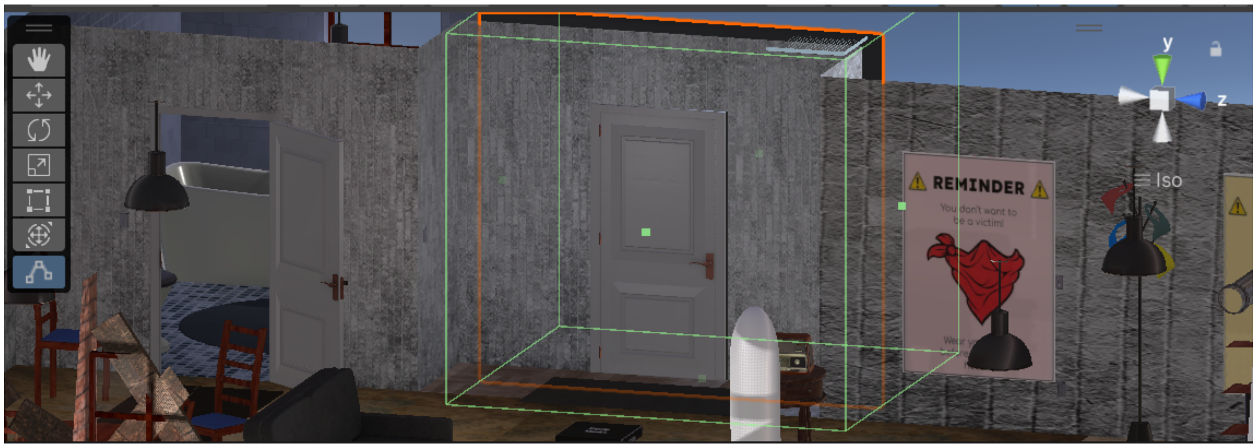


Then the following script is written, essentially stating that once the colliders intersect, the trigger will be set to active, thus prompting the animation.

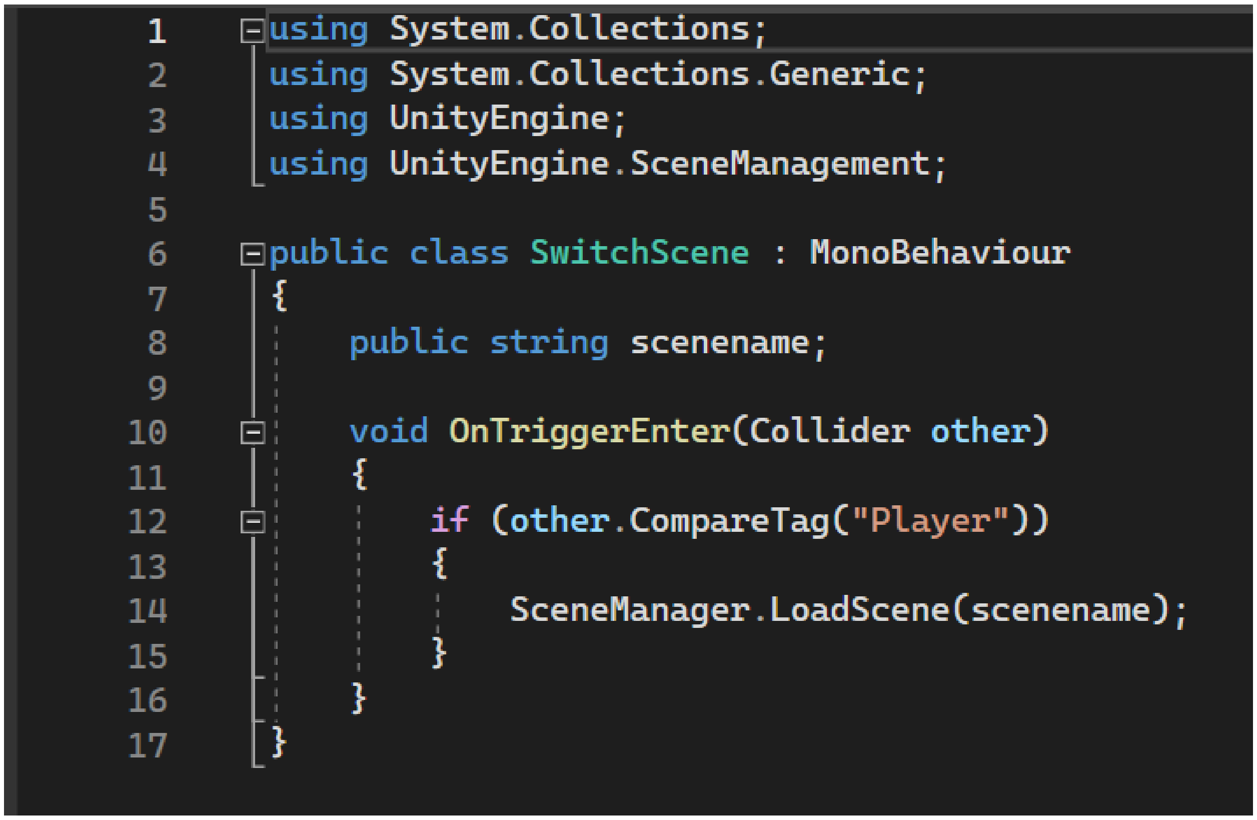


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DoorMovement : MonoBehaviour
{
    Animator anim;

    void Start()
    {
        anim = gameObject.GetComponent <Animator>();
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            anim.SetTrigger("Active");
        }
    }
}

```

Finally, for the final screen a collider is placed near the exit of the apartment.



Next a very similar script to the others is written, with the same dependency on colliders intersecting, but here switching scenes to the scene with the final screen UI.



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

public class SwitchScene : MonoBehaviour
{
    public string scenename;

    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            SceneManager.LoadScene(scenename);
        }
    }
}
```

By this point all the interactive elements are about complete, with the only possible edits being to the colliders of the objects.

6.2 Testing & Validation

We tested every prototype we made during this process. The prototypes we made were, the door interactive prototype, the house asset prototype, the movement prototype, the alterations to the house prototype and our final interactivity prototype. We tested our prototypes based on our prototype test plan which is shown below. Most tests addressed multiple Test ID's as we found that many of our Test ID's were somewhat redundant. For most tests we did not use the VR headset itself as it was often unrealistic to use a VR headset as it was difficult to get time with one. We instead ran our tests by using a movement script and playing our VR simulation as a traditional videogame. The following details from our tests are taken from Deliverables F, G and H which are linked in section 9.

Overview of Prototypes and Tests

Test ID	Objective	Prototype & Test Description	Results to be Collected	Estimated Time
1	Ensure that the storyboard communicates our main message	Make a basic storyboard which has screenshots of the assets we are planning to use and communicates the intended game play and scene structure	Qualitative observations based on other students, and hopefully the client, of how well the story board communicates risks of autonomous killer robots	About one hour to make the storyboard and a week to gather sufficient feedback / perspectives
2	Ensure the player can move properly	Set a list of paths meant to correspond to certain inputs, and run these using a “movement” script in an undeveloped environment.	Mostly qualitative observations of how well the program produces the desired movements.	Given we already have an idea of such a program, I assume it would take one hour to develop the script and test it.
3	Ensure that the scenery is realistic based on the assets used	Import the assets into the scene and make sure that the movement still works. Update the scene to make it more realistic and fit our original storyboard	Qualitative observations to check that the scene is realistic and assets are successfully imported	It will take 20 mins to import the needed assets but we may take an hour or more to play around with it and adjust the scene based on our research and user feedback
4	Ensure objects can be interacted with	Design a simple code that produces a response when the user interacts with an object. Run this code through an undeveloped environment with a designated object to be interacted with.	Qualitative observations on if the object responds to interaction as intended. Designate one clear change in behavior, such as change in color.	Given the simplicity of the test, this should only take an hour to develop and test.
5	Ensure that movement works in closed environments (the house	If the program from test 1 runs successfully, it can be tested in a closed environment, most	Same observations as the first movement test (Test ID = 2), with keen attention on how the player interacts with	Given the code would be written, this should take only 30 minutes.

	where most of the VR will take place)	likely whatever model for the house we end up using. The test would be similar using paths and inputs, not also causing collisions between the player and walls to see how they interact.	other objects.	
6	Adjust the house asset to show the effects of autonomous killer robots (see project plan for these elements)	Examine the window and radio (scene 1) and the window, masks, and signs (scene 2) making sure that they are realistic and visible no matter how you move around the scene	Qualitative observations to check that the scenes are realistic	Could easily take a couple hours to make adjustments to the surroundings and possibly even to the code of the VR
7	Ensure that the transition from the indoor to the outdoor environment is seamless and doesn't cause any lagging or discomfort	There must be a virtual door that users can interact with to exit the house as well as an animation of the door opening to reveal the outside environment (similar elements in every house - curtains over windows, locked doors, etc.). Some lighting changes would make the outdoor setting more realistic	Qualitative observations to check that the scenes are realistic and that the transition does not introduce any performance issues, such as lagging	May take about one hour to code and test the transition but some adjustments may need to be made to the outside world which would take a longer period of time (about an hour or two more)
8	Integrate sound effects into the VR based on certain interactions (ex. creaking noise when the door to outside opens and radio which would	Ensure that sound effects are in synch with VR interactions and that the sounds don't produce any lagging or affect user experience	Qualitative observations to check that the scenes are realistic and that the sound effects do not introduce any performance issues, such as lagging	Given the majority of the code would be written and most of the VR should be designed, this should take only 30 minutes to an hour.

	play throughout the VR)			

6.2.1 The Interactive Door Test

Test plan from Deliverable E:

Test ID	Objective	Prototype & Test Description	Results to be Collected	Estimated Time
4	Ensure objects can be interacted with	Design a simple code that produces a response when the user interacts with an object. Run this code through an undeveloped environment with a designated object to be interacted with.	Qualitative observations on if the object responds to interaction as intended. Designate one clear change in behavior, such as change in color.	Given the simplicity of the test, this should only take an hour to develop and test.

Because of the inherent consistency that code provides and the simplicity of our prototype we did not need to run many tests. We tested the door interactivity cycle 3 times and had the same results each time.

Qualitative observations

Aspects of prototype that work as intended	Prototype shortcomings	Change in behavior for next iteration
● Door opens with a mouse click ● Door closes with a mouse click ● Door stays open as long as the user does not click ● Animation is smooth and lifelike	● The opening and closing door cycle cannot loop ● Door can only close when the width of the door is clicked	● Door can open and close as many times as the user wants ● Door can be closed when any portion of the door is clicked

User feedback

While it would've been fun to go up to parliament and ask the parliamentarians to try out our door code as they are our ideal users of our final system we realized that this might not be the best use of our time. We instead decided to ask our friends and roommates to try the code as they probably have about as much experience in door unity codes as the people on parliament hill. We selected three users and gave them the laptop with the simulation running with no input or guidance from us. We monitored their actions and what they were saying out loud.

User	Experience	Quote
1	● User 1 immediately clicked on the door and it opened immediately ● User 1 then tried to close the door by clicking the same place as they did to open it and was unsuccessful. ● User 1 clicked on various parts of the door for roughly 15 seconds before clicking on the width of the door to close it again.	"Very intuitive to open the door. Closing it, not so much"
2	● User 2 opened the door immediately like user 1 ● User 2 was also confused as to how to close the door but figured it out by spam clicking until they clicked the right spot ● Once the door was closed user 2 tried to open it again but was unable to ● User 2 tried to open the door for roughly 10 seconds before giving up	"I wish I could open and close it again"
3	● When User 3 was given the laptop they didn't know what to click on or do ● User 3 sat dormant looking around the unity interface unsure of what to do. ● After about 45 seconds of inactivity user 3 clicked on the door and exclaimed "I didn't know that was a door" ● Once user 3 opened the door they like User 1 and 2 couldn't figure out how to close it for roughly 10 seconds until they figured it out	"Overwhelming"

6.2.2 The House Asset and Movement Prototype Test

We ran the House Asset and Movement Prototype tests simultaneously as they were both reliant on the house asset.

Model Used For House Asset and Movement Prototype Test

We used an experimental model for this test. An experimental model was the best option for this test because it was the most effective and simple model for our case. It would have been ineffective to make a numerical or analytical model of the house and movement code when we could instead use the Unity software to make it happen and test it experimentally. Our model is fairly high fidelity because we made it in the same software as our final product will be. This makes the results of our test more valuable because the movement and house model closely resembles our vision for our final product.

Specifics of Tests

1. The first test we ran was a simple observation based test to determine if the asset we purchased is realistic and functional.
2. The second test we ran was a movement based test to determine if the user could move in all 6 directions (right, left, forward, back, up and down) all without leaving the confines of the home. This test was run 3 times.

Test 1 Qualitative Observations

Areas where the prototype worked as intended	Areas that need improvement
● House looks like a traditional house ● Windows are functional IE can be looked through ● User could still move in the house with the assets imported ● House looked realistic and not overly animated	● Couch cushions seemed to be floating over the couch ● Template items that need to be customized

Test 2 Qualitative Observations

Areas where the prototype worked as intended	Areas that need improvement
● User was easily able to move around the scene with the WASD keys in all 6 directions ● Movement was smooth and intuitive ● User was able to control their speed of movement as to stay in control	● User was free to fly out of the house as the house does not confine the player ● User was free to move too far in the upwards direction which takes away from our intended level of realism ● User could move through household objects like chairs and tables

User Feedback

We decided to have our roommates and friends test our prototype as they likely have about as much experience with unity movement codes as the politicians on parliament hill. We selected three users and gave them the laptop with the simulation running with no input or guidance from us. We monitored their actions and what they were saying out loud.

User	Experience	Quote
1	● User 1 seemed unfazed by the layout of the house and did not make any remarks about the house which suggests that our simple and traditional background is working as intended ● User 1 quickly figured out how to move around the scene and explored the layout of the house ● User 1 initially seemed shocked when they were able run straight through elements of the house	"That chair is only for show"
2	● User 2 took roughly 10 seconds to look around the house with the mouse before asking "What do I do?" ● User 2 has no experience in gaming so they did not understand that they needed to press WASD to move. ● User 2 continued to look around but never figured out how to move. User 2 remarked "Very pretty house"	"How do I move"
3	● User 3 looked around the house for around 5 seconds before moving no remarks about the house layout or style ● User 3 originally attempted to move with the arrow keys before trying WASD and succeeding ● User 3 moved around the house on the ground level for around a minute, exploring the house before they realized that they could fly and were not confined by the house and flew through the ceiling.	"What happens if it rains?"

6.2.3 Alterations and Final Interactive Prototype Test

We again decided to run the alteration and final interactivity prototype tests simultaneously as they did not contradict one another so they could be run simultaneously.

Specifics of the Tests

1. Our first test will involve qualitative observations and poll data as to how much our adaptations send our intended message of humans adapting to life with killer robots. We showed 5 of our roommates and

friends our scene and then asked them a series of questions to gauge their feelings about our scene. Four of the questions were asked on a scale of 1 to 5. We deemed that if the aggregate scores for the numerical questions were over a score of 17 then that portion of the test was deemed successful.

The questions were:

1. On a scale of one to five how much does our scene resemble how you imagine life to be in a world with killer robots?
2. On a scale of one to five how "grungy" is our scene?
3. On a scale of one to five how much does our scene resemble a familiar environment?
4. On a scale of one to five how unappealing does this house appear?
5. How does this scene make you feel?
6. Any other comments?

2. Our second test will be a mix of qualitative observations and data. This test will be run by five different users and they will each do three trials. In each trial the user will try to exit the house via the door. Their success or failure will be noted and they will be asked how to describe how seamless and easy the transition was. There will also be qualitative observations as to the ease with which they proceeded through the scene.

Test 1 Results and Observations

The successful aggregate test results are highlighted in green and the unsuccessful result is highlighted in red. These results show that we can make our scene more realistic to user expectations.

Question Number	Aggregate Score
1	**15**
2	**20**
3	**23**
4	**18**

Comments From Users in Response to Questions 5 and 6

Question 5	Question 6
● "This scene makes me feel uncomfortable and gross" ● "This scene is very unappealing and does not seem like a good place to live" ● "Seems like a trap house"	● "I don't really know what a world with killer robots would look like but this seems plausible." ● "I want more context as to why the house looks like this"

● "If this is what life is like with killer robots it is not a world I want to live in. ● "Ick"	● "I like how the message of the simulation is subtle" ● "I don't really understand what the point is." ● "No further comments"

Test 2 Results and Observations

Percent of trials where the door was successfully opened - 73%

	User Descriptions of Ease to Open Door	Our Commentary & Observations on Tests
User 1	"Was intuitive, the door opened as I would expect it to in real life. The outside world is undeveloped though"	3/3 on door openings and never had any difficulty opening the door on every trial.
User 2	"There were gaps between the door frame and the door but otherwise it worked well."	2/3 on opening the door but paused every time to look at the door gaps.
User 3	"At first it didn't click that you could click anywhere on the door to open it but once I understood that it was easy"	1/3 on door openings. Kept running into the door expecting it to open via proximity but on the third trial realized that you had to click.
User 4	"I got caught on an invisible asset and couldn't leave the door for one of the trials but otherwise it was good."	On trial 2 user 4 went to open the door and succeeded but was unable to exit the house as they were caught on an invisible asset. Otherwise they had no issue.
User 5	"It's a door and it opens, what more is there to say"	3/3 on door openings, it was quick and easy for them.

7 Conclusions and Recommendations for Future Work

We learned many things during this project. Our main takeaways are the importance of soft skills like teamwork and communication. This project was a massive undertaking so we were all reliant on each other to make our project as good as possible and get the job done. We also learned the importance of communication as it prevented tension from building up in our group because we could talk out our problems and stay on the same page. We also learned a lot about coding in Unity which is a transferable skill to C coding. We would advise future groups to use the VR technology as a foundation for their coding as a lot of useful codes are already included in the VR technology. If we had more time we would've removed the tiles on the floor which cued our interactive elements and would've made it more elegant. We abandoned the outside section of our simulation and if we had had more time we would've extended our simulation to outside and would've added more elements to make our anti robot message even clearer.

8 Bibliography

[1] M. Wareham, "Country Views on Killer Robots," Human Rights Watch, https://www.hrw.org/report/2020/08/10/stopping-killer-robots/country-positions-banning-fully-autonomous-weapons-and (accessed Nov. 28, 2023).

[2] "Race and killer robots," Stop Killer Robots, https://www.stopkillerrobots.org/race-and-killer-robots/ (accessed Oct. 24, 2023).

[3] J. J. Kellaris, A. D. Cox, and D. Cox, "The effect of background music on Ad Processing: A contingency explanation," Journal of Marketing, vol. 57, no. 4, p. 114, 1993. doi:10.2307/1252223

[4] International Committee of the Red Cross, "'risks from the unconstrained use of autonomous weapons in armed conflict are Stark,'" International Committee of the Red Cross, https://www.icrc.org/en/document/risks-unconstrained-use-autonomous-weapons-armed-conflict-are-stark (accessed Oct. 24, 2023).

[5] International Committee of the Red Cross, "ICRC position on Autonomous Weapon Systems," International Committee of the Red Cross, https://www.icrc.org/en/document/icrc-position-autonomous-weapon-systems (accessed Oct. 24, 2023).

[6] "33 states call for urgent negotiation of New International Law to limit autonomous weapons: UNA-UK," UNA, https://una.org.uk/news/33-states-call-for-urgent-negotiation-of-new-international-law-to-limit-autonomy-in-weapons-systems (accessed Oct. 24, 2023).

[7] Immoral Code - A Film by Stop Killer Robots. Campaign to Stop Killer Robots, 2022.

9 APPENDIX I: Design Files

MakerRepo project link: https://makerepo.com/avabtts/1843.blueprint-brigade

Table 3. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
Immoral Code	https://immoralcode.io/index.html	October 24, 2023
Campaign to Stop Killer Robots	https://www.stopkillerrobots.org/	October 24, 2023
The International Campaign to Abolish Nuclear Weapons (ICANW)	https://www.icanw.org/	October 24, 2023
International Committee of the Red Cross	https://www.icrc.org/en	October 24, 2023
Deliverable H	Deliverable H	November 26th, 2023
Deliverable B	Deliverable B	October 1st, 2023
Deliverable C	Deliverable C	October 8th, 2023
Deliverable F	Deliverable F	November 4th, 2023
Deliverable G	Deliverable G	November 12th, 2023

Wrike Link:

https://www.wrike.com/frontend/ganttchart/index.html?snapshotId=ycPMSOZWas1m6usc8wtSVm0Xz3i7PVqM%7CIE2DSNZVHA2DELSTGIYA