GNG1103

Design Project User and Product Manual

FIRE INSPECTOR CLIMATE SIMULATION

Submitted by:

V.ORB – GROUP 3

JAMES BUGEAUD, 300158603

MOHAMMAD ODEH, 300189071

MALCOLM ROSS, 300438516

KRITHIVAAS VYAS, 300457627

4 April, 2025

University of Ottawa

Table of Contents

List of Figures

List of Tables

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
HMD	Head-Mounted Display
LMB	Left Mouse Button
RMB	Right Mouse Button
SFX	Sound Effects
VFX	Visual Effects
VR	Virtual Reality
XR	Extended Reality

Table 2. Glossary

Term	Acronym	Definition
C#	[N/A]	Programming language used in Unity.
Plugin	[N/A]	Program added to Unity to simplify or provide additional functions.
Prefab	[N/A]	Conjoined objects in Unity forming a single effective mechanism.
Script	[N/A]	File of code written to perform a specific task in Unity.
Unity / Unity Engine	[N/A]	Game development software in which the product was built.
Universal Render Pipeline	URP	One of the configurations for Unity to prepare and display graphics.
Visual Studio	VS	Software environment for writing code and running backend programs.

1 Introduction

This User and Product Manual (UPM) provides the information necessary for mature audiences and interested developers to effectively use or iterate on the Fire Inspector Climate Simulation (FICS) and for prototype documentation. This document begins by explaining the purpose of the product, then flows into how to start up, interact with, and close down the simulation. Extensive instructions are then provided on how the various are assembled and details on the various tests and prototypes are tabulated.

General users and developers should refer to this document for instructions on how to use the FICS and recreate or iterate upon it, respectively. This product is intended to be used in VR, which may induce motion sickness in some users. This simulation includes depictions of wildfires and house fires. Users are advised to exercise discretion when using the FICS.

2 Overview

Climate change is a matter of global concern that has been accelerating in severity over the last nearly three centuries due in no small part to human technological advances that have as a byproduct wrought havoc on the environment. In recent decades, new understandings of climate science have continuously demonstrated that the current pace of climate change has surpassed a critical point and now requires human intervention to curb it as soon as possible. The scale on which this phenomenon occurs however, both geographically and chronologically, makes it something difficult to fully understand and be concerned about for those not already familiar with and invested in it. Academics and climate advocacy groups have worked to use the immersive potential of XR and VR technology to develop a greater appreciation of the issue in target audiences [2-6] but often struggle to engage and educate users much more than traditional media.

To meet the user's needs, the product must measurably increase their understanding of and concern for the consequences of climate change. To achieve this, the client seeks a simulation that makes use of XR technology to bring the user closer to said consequences and allow them to be experienced.

To achieve this, the simulation aims to tie both the immediately observable and the more distant, abstract effects together in a way the user feels personally engaged. Unique among those that focus on wildfires, this simulation aims to meet the objective through the interplay of a scanner tool and companion dialogue system. With the scanner, the user explores and interacts with their local environment to find the visible signs of climate change, then tags those objects to get specific data and trends on a diegetic in-game display to provide education on the issues right before them. The scanner tool is designed to make the data more immersive, but as an analog for a scientific device, it would not make sense in the canon of this simulation for the tool to communicate how climate change will personally affect people. As such, it is complemented with a companion character who follows the user and will play contextual dialogue when they tag an object with their scanner. The companion has a personal stake in the destruction caused by the fire and wants to know how it came about and so will engage in short conversations with the user that steer toward connecting what was just scanned to what it entails for the nearby townspeople. This is intended to make the world feel more lived in to make it clear that the climate effects being observed affect people and not just an abstract environment.

First Scene - Forest Outskirts

Second Scene - Evacuating Town

Third Scene - Future Burning Town

2.1 Conventions

The document will commonly refer to ‘attaching’ components to objects. When it is stated to ‘attach’ an object to a script, that script has some set of global parameters in the Inspector window. Instructions will state which parameter to assign the object to. If the component is not being attached to a script, navigate to the Inspector window, click ‘Add Component’, and type in the name of the desired component. Click the component to add it and continue with the instructions.

2.2 Cautions & Warnings

This simulation features depictions of wildfires, house fires, and mass evacuation. Users sensitive to such content are advised to exercise discretion. This simulation is intended to be engaged with through VR, which may induce motion sickness in some users. If such symptoms occur, immediately remove the HMD to exit the simulation.

3 Getting started

The simulation itself is a linear experience intended to be played through the application file (.apk) using a VR headset. The hardware setup is expected to take the bulk of the effort.

1. Connect the USB-C cable from the Oculus headset to the computer used for the simulation. If the headset only displays a grainy black-and-white image of what is in front of you, try to reconnect the cable to a different USB-C on the computer.
2. If the headset directs you to set up the physical environment, follow the instructions it provides until it states that the setup is complete.
3. Remove the headset to be able to navigate the computer's desktop. Open the simulation's folder and double click the "VR_Prototype" file to begin.

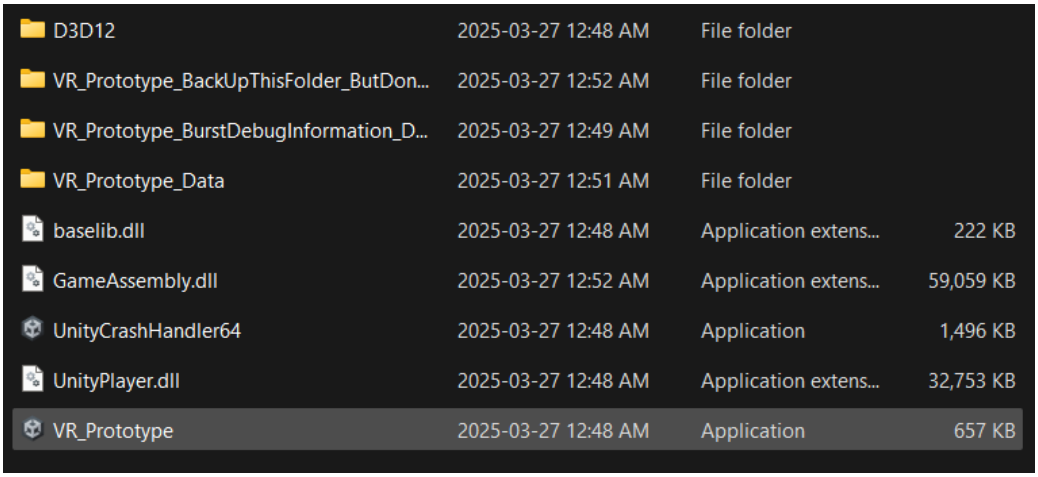

Figure 1. Application file directory

4. Put the headset back on to begin once the language selection menu appears on your monitor. Use the controller to direct the laser over menu and world elements, then press the trigger under your index finger to interact.

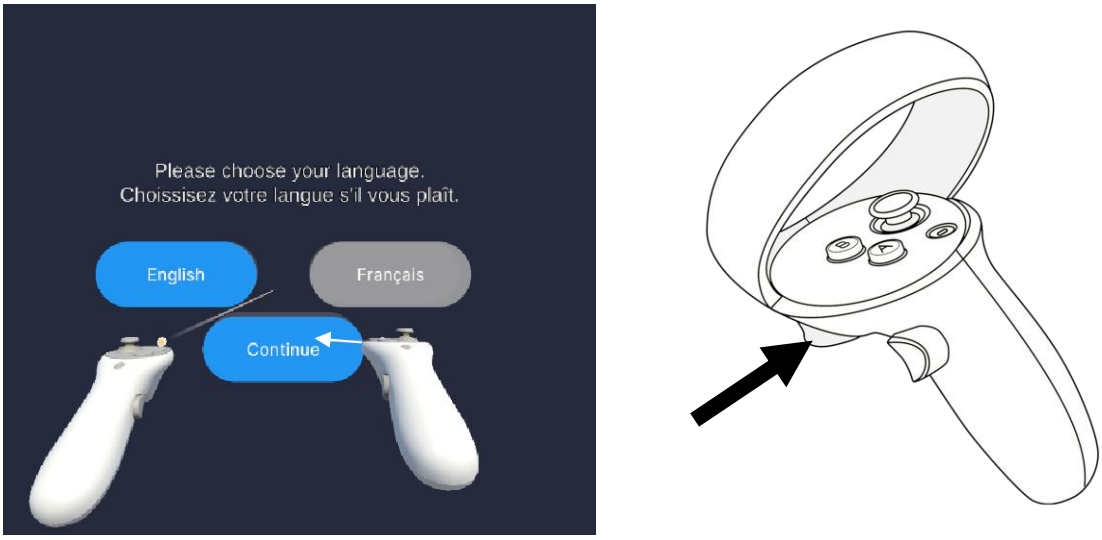

Figure 2. Menu and interaction button

5. The simulation is progressed by pressing the "Continue" button on menus and by clicking through on the companion character's dialogue box when in a scene. This will play through the start menu, three scenes, and end at the end menu.
6. When the end menu is reached and the "Continue" button no longer does anything, remove the headset and close the program by pressing ALT+F4.

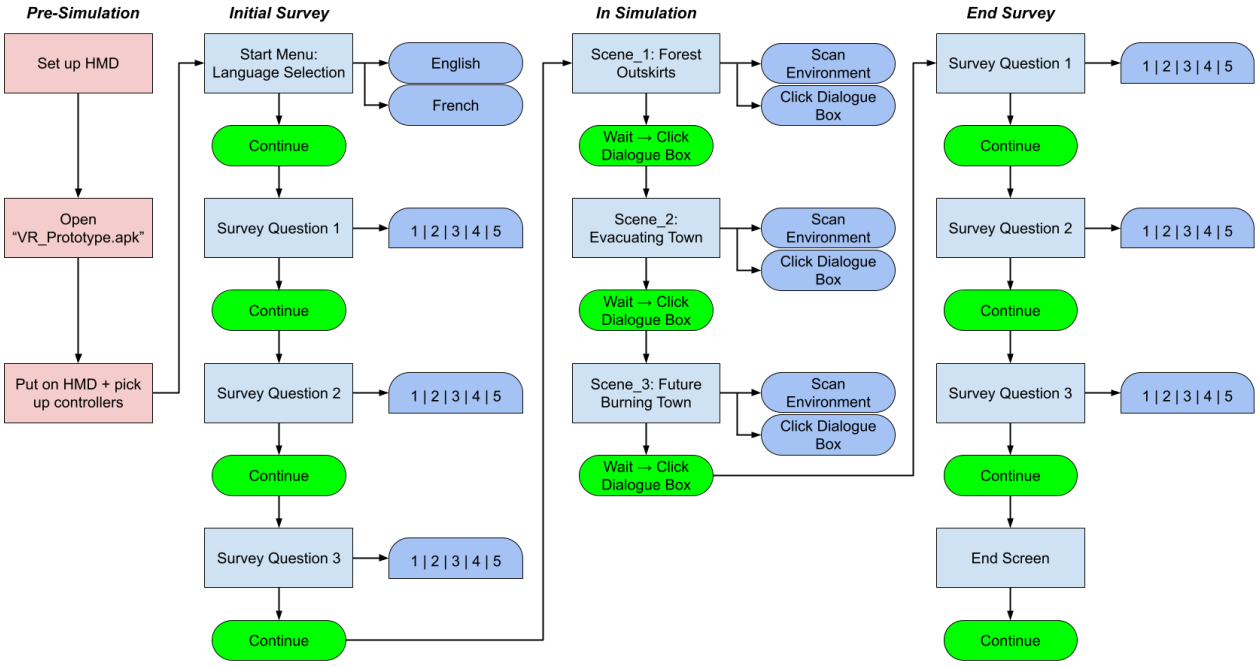

Figure 3. Simulation progression flowchart

3.1 Configuration Considerations

Refer to the guides linked in section 3.3 to configure the Oculus Quest 2 headset as well as set up the Unity editor to iterate on the program. To run the Unity editor, it is highly advised to use a computer with an integrated graphics card for suitable performance.

3.2 User Access Considerations

The simulation is intended to be played using a VR headset and controllers, which require a moderate range of arm and neck motion along with appropriate empty space to exercise such motion to fully engage with the product. The VR hardware may also be prohibitively expensive for some users, with prices upwards of $300. Using the MockHMD XR plugin in Unity allows users to instead play the simulation on a desktop or laptop computer with the mouse and keyboard for controls, listed as follows:

- W, A, S, D keys: Move the character forward, back, left, and right, respectively.
- Mouse movement: Rotate the camera and controllers in the direction moved.
- LMB click: Press button (Menus), fire scanner tool (In-Game).
- R key press: Change function of RMB hold between free rotation and free translation of the camera.
- RMB Hold + Mouse movement: Rotate or translate the camera in the mouse's direction independently from the controllers, depending on setting (see R key press).
- Shift key hold: Alters the controls such that Mouse movement rotates the left controller instead of the camera.
- Space key hold: Alters the controls such that Mouse movement rotates the right controller instead of the camera.
- Space key hold + W key hold: Alters the right controller to direct a teleport marker at the end of a parabola. Release the W key to teleport to the marked circle.
- Space key hold + S key press: Snap the camera to face backwards from the camera's current facing.
- Space key hold + A or D key press: Snap the camera left or right, respectively, by 45^{o} increments.

Though less immersive or convenient to use than the actual VR hardware, this enables the simulation to be experienced by those who cannot access such hardware. When playing the simulation with a headset, users will not be able to wear glasses, which may impede the legibility of menu elements. When in doubt, the 'continue' button is always placed at the bottom center of the screen. The simulation does not feature specific support for colour blind users, which may impede the ability to recognize the visual queues and feedback provided for the scanner tool.

3.3 Accessing/setting up the System

Setting up the VR hardware

If your VR headset is already set up or you do not intend to play the simulation using a headset, you can skip to the following section on starting up the simulation. Otherwise, the user should refer to guides provided by Meta for installing the necessary software and setting up the Oculus app or turn to similar 3rd party tutorials.

Setting up the simulation in and out of the Unity editor

For standard use, the simulation is run by double clicking on the "VR_Prototype" application file. If the simulation is run from within the Unity editor, a Unity ID is required for access. Procedures for account creation, sign-in, and recovery can be found through the Unity sign-in page [1]. From there, return to the Unity homepage and navigate to the Unity Engine download under the Products tab to download version 3.11.1 or higher of the Unity Hub. Once the hub is installed, install version 6000.0.39f1 or higher of the Unity Editor. Through the Unity hub, open the project file "VR_Prototype" and press the play button at the center top of the editor window.

3.4 System Organization & Navigation

The simulation provides a linear experience, with each section only allowing progression to the singular next section. This is divided into the starting and ending survey menus, which are identical in how the user can engage with them, and the separate scenes that compose the story. Refer to Figure 3 for a visual representation of how each section is navigated between.

3.4.1 Start and End Menus

The start menu features a language selection screen and three survey questions, each as a separate tab, while the end screen features the same three survey questions and a final screen thanking the user for the time and informing them that the simulation has ended. Only one tab is displayed at a time, and each has a button labeled "Continue" at the bottom to allow the user to progress to the next tab. This can be clicked by directing the controller's laser pointer over it and pressing the trigger. At the third survey question on the starting screen, the "Continue" button will navigate the user from the starting menu into the simulation proper.

3.4.2 Story Scenes

After a set period of time in each scene, the companion character will automatically play special dialogue which, upon the user interacting with the dialogue box as usual, will navigate them to the next scene: if they are on scene 1, they will move to scene 2; if they are in scene 2, they will move to scene 3, if they are in scene 3, they will be taken to the end menu screen. If the user is unsure how to progress, the companion will always follow the player and only progress dialogue

when the user clicks on the text box, with the special dialogue added to the queue after 40-50 seconds.

3.5 Exiting the System

1. Remove the HMD if it was used to play the simulation and return both it and the controllers to their housings.
2. If playing through the application file, press ALT+F4 to terminate the program and skip the remaining steps. Otherwise, press the escape key on the keyboard to unlock the cursor from the game window.
3. Navigate to and click on the “Stop” button at the top center of screen, represented by a grey square as per media control convention.
4. If the simulation will not be played again, navigate to the close window button in the top right of the screen and click to close the Unity editor.

4 Using the System

The following subsections provide detailed, step-by-step instructions on how to use the various functions or features of the FICS.

4.1 Menus

Menus feature text, a continue button, and either a slider or a pair of selection buttons. To operate, move the controller to direct the laser pointer over the element, indicated by the element turning a shade darker. Press the trigger on the controller to interact with the element, either to switch the language option or set the value on a slider. Only pressing the “Continue” button will change the menu tab.

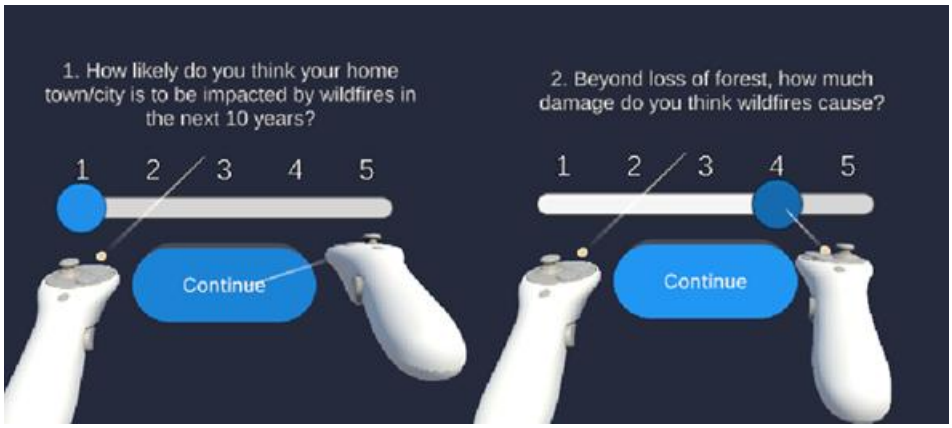

Figure 4. Navigating a menu with the VR controllers

4.2 Scanner Tool

The scanner tool is locked to the user’s right hand during story scenes. Point the tool at an object and read the text in the top right of the device’s display to see if the currently targeted object is scannable. Press the right trigger to fire the scan, which will update the display based on the object and whether or not it was scannable.

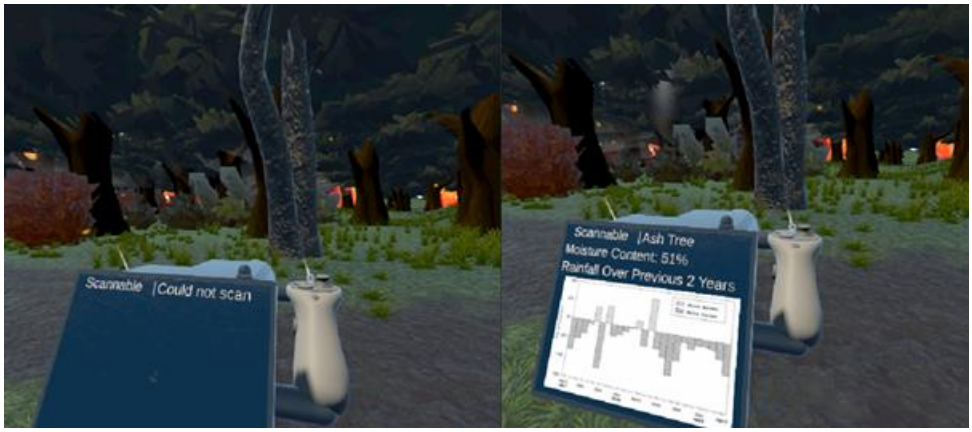

Figure 5. Before and after tagging a scannable object

4.3 Dialogue

Dialogue is the mechanism for story progression. Move the controller to direct the laser pointer over the text box hovering in front of the companion character and press the trigger to play the next dialogue segment. The companion's dialogue displays in a blue box with a typewriter effect, while the player character's dialogue displays in a solid white box instantly.

Figure 6. Directing the laser pointer at the dialogue box

5 Troubleshooting & Support

5.1 Error Messages or Behaviors

Headset does not leave the VR hub on playing

This issue is typical of the headset's client software, such as the Meta app for the Oculus Quest or SteamVR for the Valve Index. Ensure you are signed into the client and running the latest version of the software. In the client, navigate to Settings → Video and turn off "Pause VR when headset is idle".

Headset/controller motion not registering

When testing from within the Unity editor, it may occur that the scene is displayed on the headset, but the controllers do not appear and/or head motion does not rotate the camera. To fix this, go to the top of the screen and navigate to Edit → Project Settings → XR Interaction Toolkit and disable the checkbox for "Use XR Device Simulator in scenes".

Visual effects only playing for the left eye

While playing the simulation with the headset, particle effects may look unnatural, and by closing one eye or the other they may appear or disappear. This occurs as a conflict that some packages and assets have in broadcasting to the VR headset. If playing through the editor, this can be remedied by navigating to Edit → Project Settings → OpenXR → Render Mode and selecting "Multi-pass" from the drop-down menu.

Namespace could not be found or doesn't exist when building the project

This compiler error may occur when building the application file (Section 6.1), causing the process to crash. The first way this can occur is if the directions of Section 6.1 have not been completed. The second likely way is through accidentally importing faulty libraries while writing scripts. Libraries that are added but not used will be shown in a faded colour when using VS Code and will not be imported when playing through the editor, but will throw an error in the build program. To fix this, review allscripts you have made that are not included in the project template and remove any faded import tags or tags that you did not consciously add.

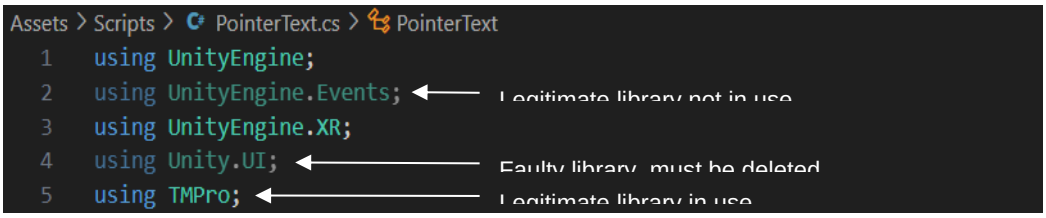

Figure 7. Valid and invalid libraries used while writing scripts

Imported asset is completely pink

This issue occurs when the shader of an imported material does not load properly. Refer to Section 6.4.3 and begin at step 4 to fix the broken asset.

5.2 Maintenance

This simulation was built in Unity 6000.0.39f1 and is not guaranteed to function in higher or lower versions. Backup any progress made to a directory of the project fixed in the above version of Unity.

5.3 Support

The developers can be contacted for questions by sending an email to James Bugeaud (jbuge008@uottawa.ca), Krithivaas Vyas (kvyas032@uottaw.ca), or Malcolm Ross (mross022@uottawa.ca). When contacting for support, include a complete description of what was displayed on the computer monitor and VR headset at the time of the error and if possible what occurred leading up to the issue.

6 Product Documentation

6.1 Building the Application File

6.1.1 BOM (Bill of Materials)

1. *Desktop development with C++ workload* ($0.00). This package is installed through the Visual Studio Installer application.
2. *Game development with Unity workload* ($0.00). This package is installed through the Visual Studio Installer application.

6.1.2 Equipment list

1. *Visual Studio 2022* ($0.00). Download Visual Studio Tools - Install Free for Windows, Mac, Linux
2. *Unity Editor v6000.0.39f1* ($0.00). Unity Hub Installation

6.1.3 Instructions

1. Launch the Visual Studio Installer by searching for “Visual Studio Installer” in your desktop’s search bar.

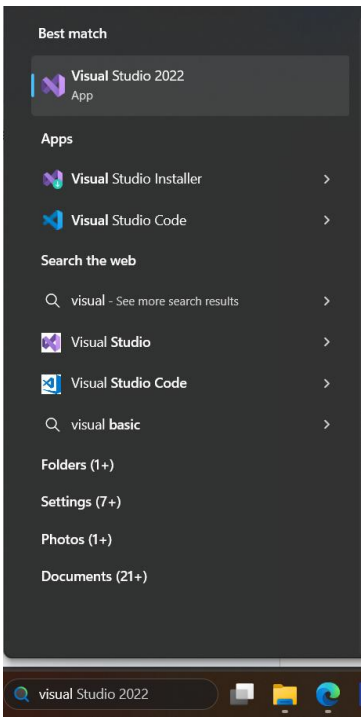

Figure 8. Accessing Visual Studio through the Windows search bar

2. Click the “Modify”, then select the two packages listed in the BOM (Section 6.1.1). Click the “Modify” button in the bottom right and allow the application to make changes to your computer.

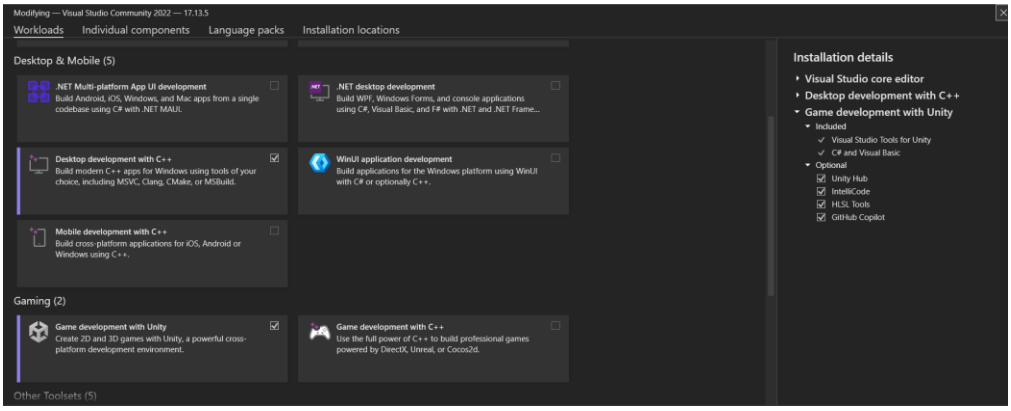

Figure 9. Selecting the necessary packages in the Visual Studio installer

3. Once the installation is complete, return to the Unity editor. If you have not already configured the project’s build profile, go to File → Build Profiles. Click “Switch Platform” and then check on “Override Global Scene List”.

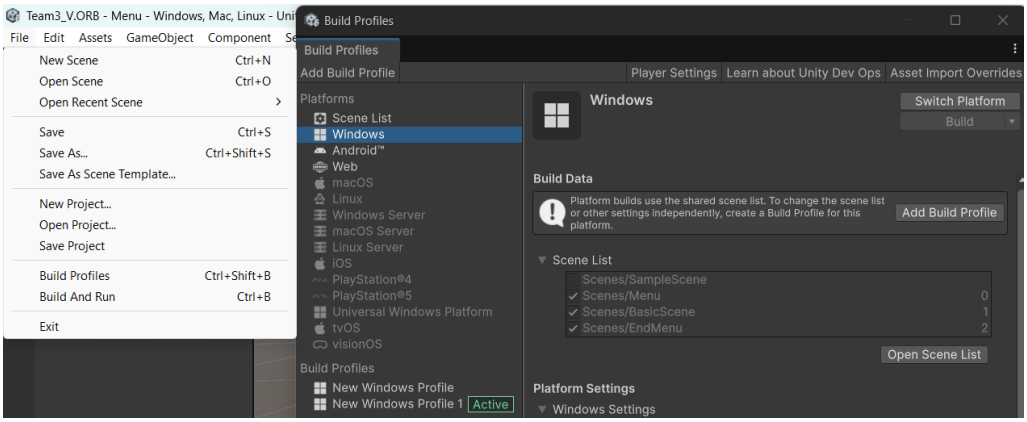

Figure 10. Accessing the "Build Profiles" menu

4. Open each scene you want to be playable and click “Add Open Scenes” to add it to the list. Repeat for each, then drag the scenes into the order you want them to be played. Keep in mind the program will open scene 0 when it launches.

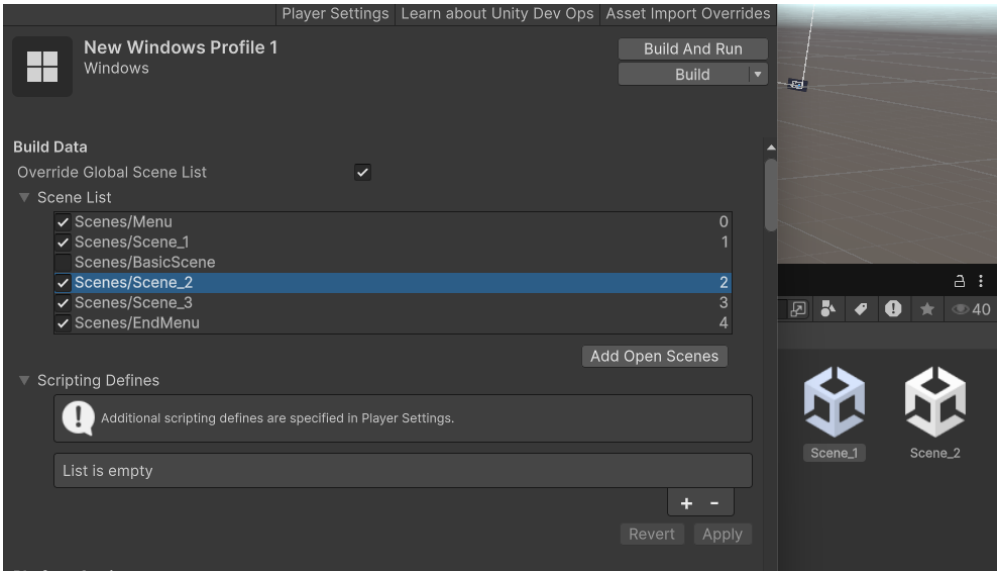

Figure 11. Selecting and ordering the scenes in the "Build Profile" menu

5. Exit the Build Profiles menu to confirm the setup. When the project is complete and ready to be exported as an application file (.apk), click the “Build” button and wait for the process to complete. Be advised this may take upwards of a few hours to complete, even using mid- to high-end hardware, and so should only be undertaken after sufficient testing in the editor.

6.2 XR Rig

6.2.1 BOM (Bill of Materials)

1. *Input System* ($0.00). This package is installed through the Unity package manager. Input System | Input System | 1.13.1
2. *MockHMD Plugin* ($0.00). This package is installed through the Unity package manager. About the Mock HMD XR Plugin | MockHMD XR Plugin | 1.4.0-preview.2
3. *OpenXR Plugin* ($0.00). This package is installed through the Unity package manager. OpenXR Plugin | OpenXR Plugin | 1.14.1
4. *XR Hands* ($0.00). This package is installed through the Unity package manager. XR Hands | XR Hands | 1.5.0
5. *XR Interaction Toolkit* ($0.00). This package is installed through the Unity package manager. XR Interaction Toolkit | XR Interaction Toolkit | 3.0.7
6. *XR Plugin Manager* ($0.00). This package is installed through the Unity package manager. About the XR Plug-in Management package | XR Plugin Management | 4.5.0

6.2.2 Equipment list

1. *Unity Editor v6000.0.39f1* ($0.00). Unity Hub Installation

6.2.3 Instructions

The XR rig does not need to be built by the simulation developer. Starting the Unity project using the built-in VR template in the Unity hub will automatically install all necessary plugins, import all scripts and settings, and create and assemble all of the subsystem's components. Navigate to Scenes → BasicScene in the Assets folder and copy the "XR Origin (XR Rig)" from the hierarchy into each new scene you make.

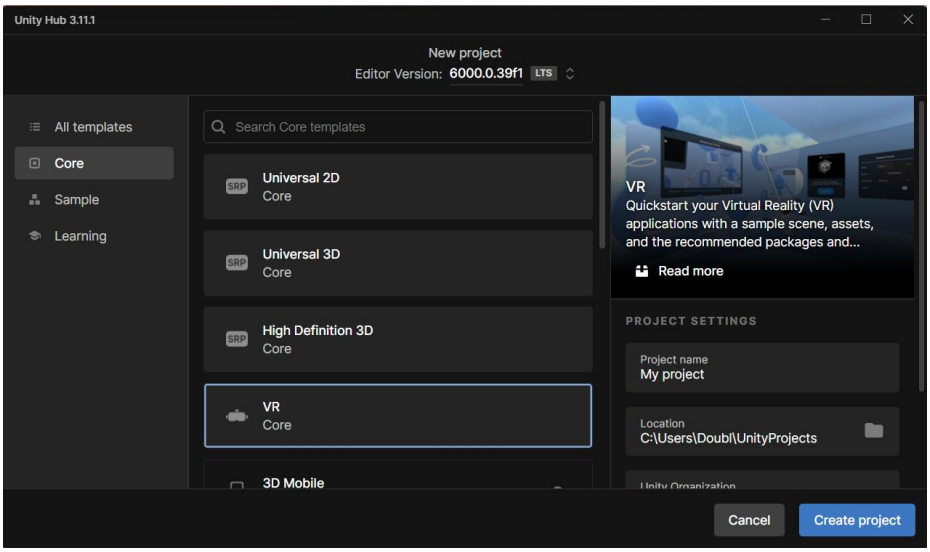

Figure 12. VR template on the Unity hub "Create Project" screen

6.3 Companion Character

6.3.1 BOM (Bill of Materials)

1. *AI Navigation* ($0.00). This package is installed through the Unity package manager. AI Navigation | AI Navigation | 2.0.6
2. *NPCFollower Script* ($0.00). https://www.sharpcoderblog.com/blog/npc-follow-player-in-unity-3d

6.3.2 Equipment list

1. *Unity Editor v6000.0.39f1* ($0.00). Unity Hub Installation

6.3.3 Instructions

Creating the companion begins with the body and follower script:

1. In the Hierarchy window, create, name, and nest the basic objects as follows:

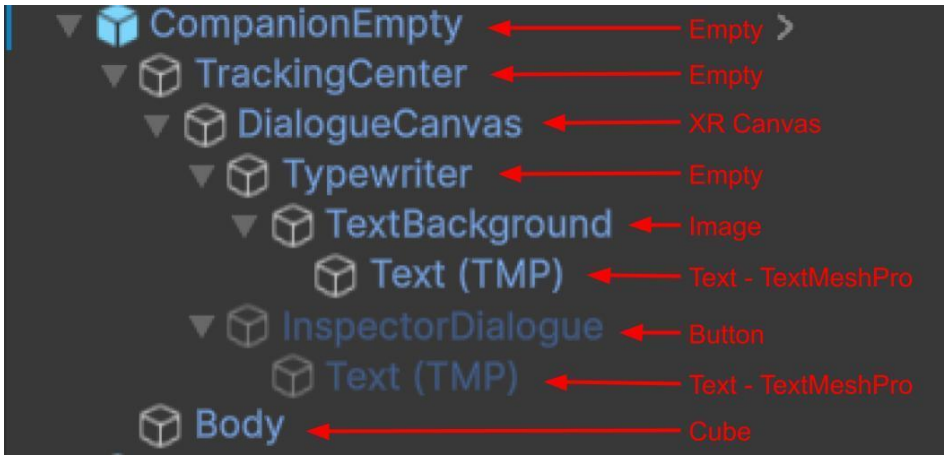

Figure 13. Companion component structure

2. Attach a Capsule Collider, Nav Mesh Agent, and 'NPCFollower' script to the Companion.
3. To configure the collider, set its Y transform to 0.5.
4. To configure the agent, set the Base Offset to 0.5 and Stopping Distance to 5.
5. To configure the script, set its Transform To Follow to the main camera.

Note that the companion will not be able to navigate properly until the navigation mesh is implemented, as done in section 6.5.3. Creating the dialogue system can be done by:

1. To the DialogueCanvas, attach the 'TypewriterAssistant' script.
2. Set the TrackingCenter's Y transform to 0.4 and attach the 'LookAtPlayer' script to it. Configure the script by setting its Player to the main camera.
3. To the Typewriter, attach the 'Typewriter' script and a Button component.
4. Configure the script by setting its Text to the Text (TMP) under TextBackground and set its Inspector Dialogue Text to the Text (TMP) under InspectorDialogue. Attach the Typewriter to the Button's On Click function, and assign the remaining fields as shown:

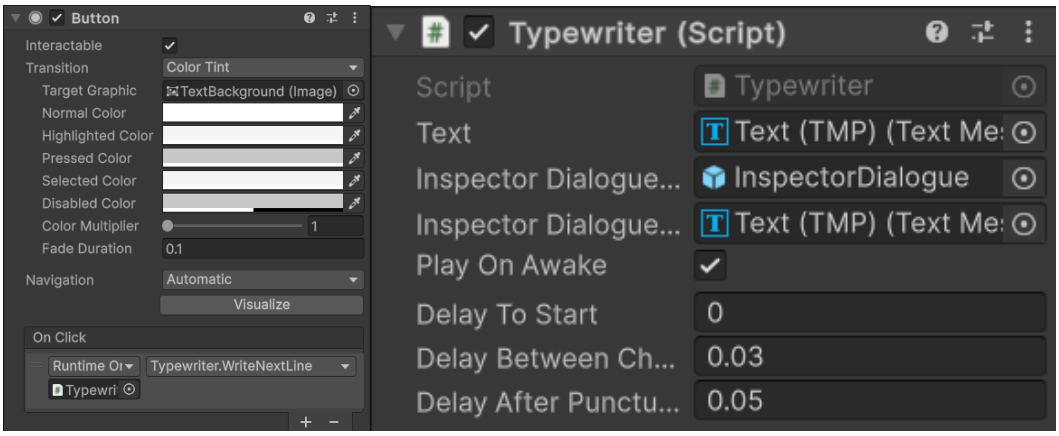

Figure 14. Typewriter interaction component settings

Configure the Button component of the InspectorDialogue as follows:

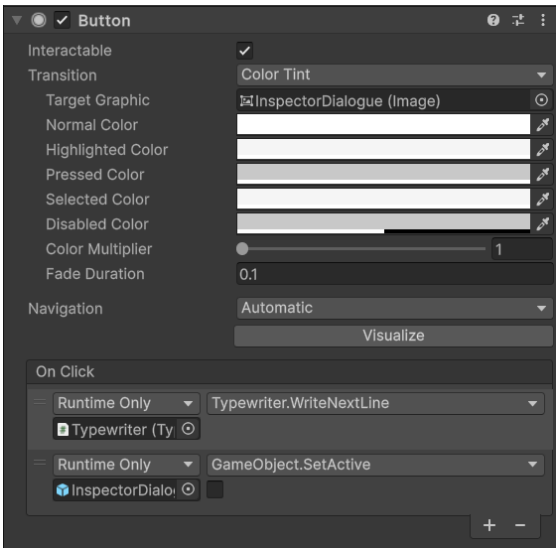

Figure 15. InspectorDialogue button configuration

5. To make the dialogue box properly resize itself as text is added, several other components must be added and configured in the exact order listed. First, in the top left corner of the Typewriter's Transform component, click the square in the upper left hand corner, then hold ALT and click the 'stretch' option in the bottom right corner of the menu.
6. To the Typewriter, add and configure a Vertical Layout Group component, then repeat for a Content Size Fitter, with the specific settings shown below:

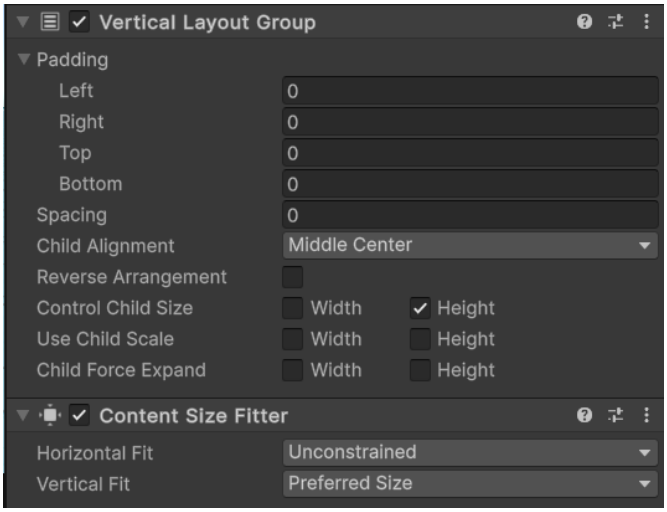

Figure 16. Typewriter layout configuration

7. To the TextBackground, add a Vertical Layout Group component, copying the settings above but with the Width boxes ticked on for Control Child Size and Child Force Expand.

This completes the assembly of the companion character. For further aid on creating the dialogue system, a video tutorial is available at Dynamic Textbox | Auto resize text background | Unity3d | UI.

6.4 Character Movement and Animation

6.4.1 BOM (Bill of Materials)

1. *Firefighter* ($15.00). https://assetstore.unity.com/packages/3d/characters/humanoids/humans/firefighter-171169
2. *Starter Asset: Character Controllers URP* ($0.00). https://assetstore.unity.com/packages/essentials/starter-assets-character-controllers-urp-267961
3. *Fantasy Horde-Villagers* ($50.00). https://assetstore.unity.com/packages/3d/characters/humanoids/fantasy/fantasy-horde-villagers-3793
4. *'NPCRun' script* ($0.00). V.ORB (Team 3) - Wildfire Inspector | MakerRepo

6.4.2 Equipment list

1. *Unity Editor v6000.0.39f1* ($0.00). Unity Hub Installation

6.4.3 Instructions

1. Import the 'Firefighter', 'Starter controller' and 'Fantasy Horde-Villagers' assets into the project file.
2. Select the prefab file in the 'Firefighter' asset and drag the prefab into the Hierarchy Window.
3. In some cases, prefabs in the scene will be pink partially or fully. If this is not the case, skip steps 4-6.
4. Select 'Search by Type' in the Project window and select the 'Materials' option.
5. Select the material that is showing to be pink. Information will be displayed in the Inspector window, at the very top of the Inspector window, select Shaders, then replace the shaders with 'Standard'.

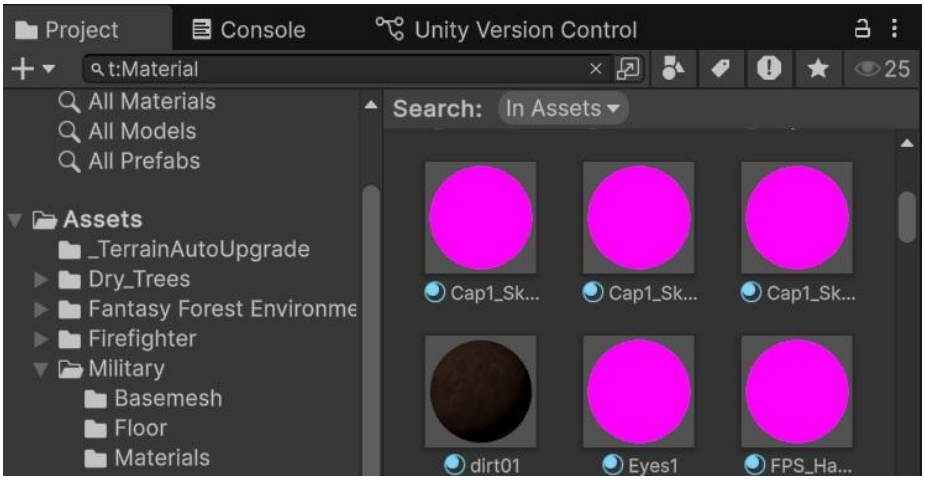

Figure 17. Search by Type tab

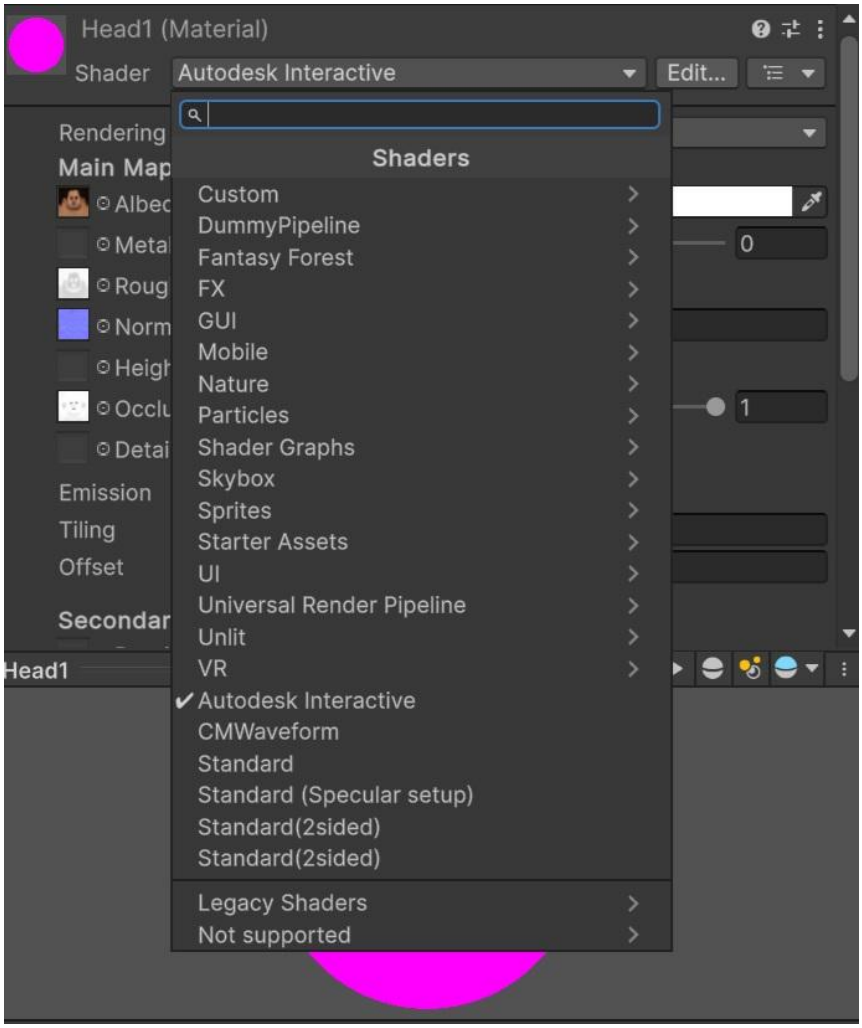

Figure 18. Shader selection dropdown menu

6. Select ‘Edit’ in the file window then select ‘Rendering’→ ‘Materials’ → ‘Convert Selected Built-in Materials to URP’.

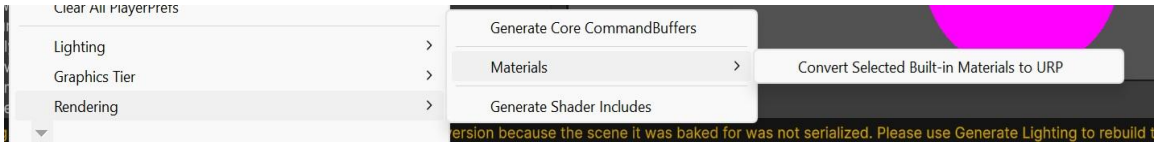

Figure 19. Navigating to the material conversion tab

7. Import Starter Asset: Character Controllers URP asset into the project file, select prefab ‘PlayerArmaturefile and drag prefab into the Hierarchy window.
8. Both character models are now inside of your scene and Hierarchy, in the Hierarchy select Firefighter and move the xyz positions and rotations to completely match the xyz position of the starter asset prefab.
9. Inside the Hierarchy window, drag ‘Firefighter’ into the ‘PlayerArmature’. Under ‘PlayerArmature’, remove components ‘Geometry’ and ‘Skeleton’.
10. In your Project window, select ‘Base Mesh’ in ‘Firefighter’ asset, press the small arrow next to the ‘Firefighter’ as shown in the image below.

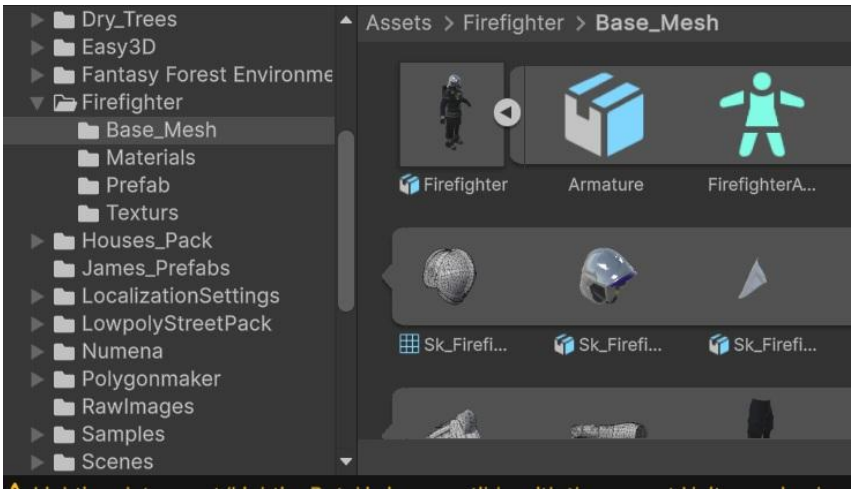

Figure 20. Expanded prefab for the avatar asset

11. In the Hierarchy window, select ‘PlayerArmature’, head over to the inspector window and drag the ‘Firefighter’ from the project window into the inspector window under Animator.

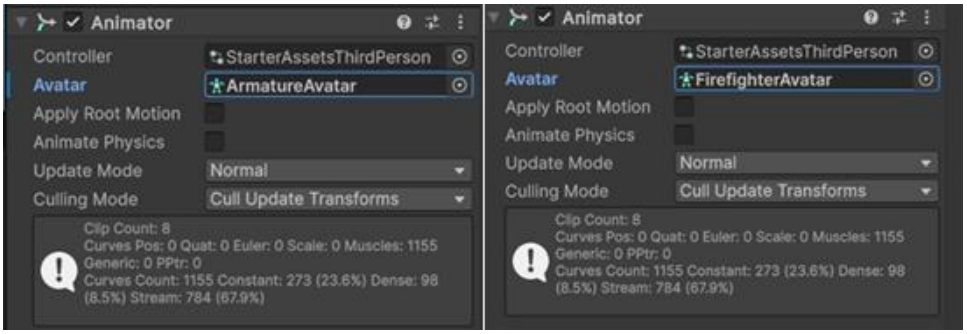

Figure 21. Before and after setting the armature's Avatar parameter

12. In the file window, select windows → Animation → Animator.

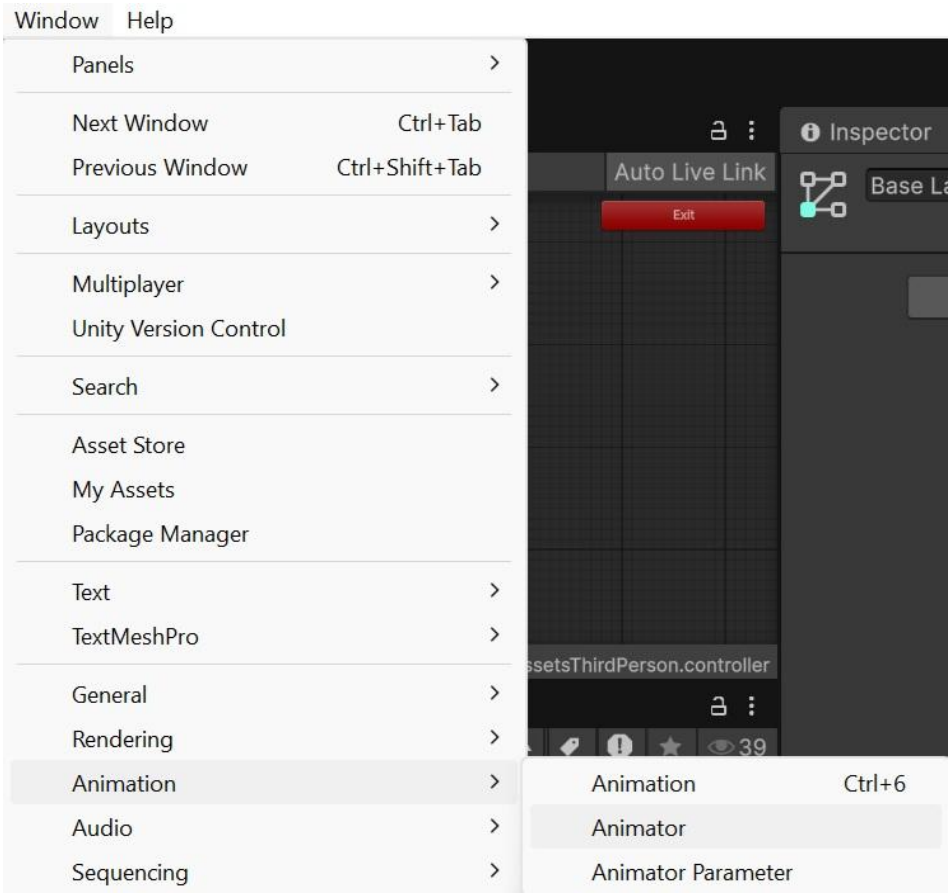

Figure 22. Navigating to the Animator tab

13. In the Animator tab, make sure that the movement animation is set up like the figure below.

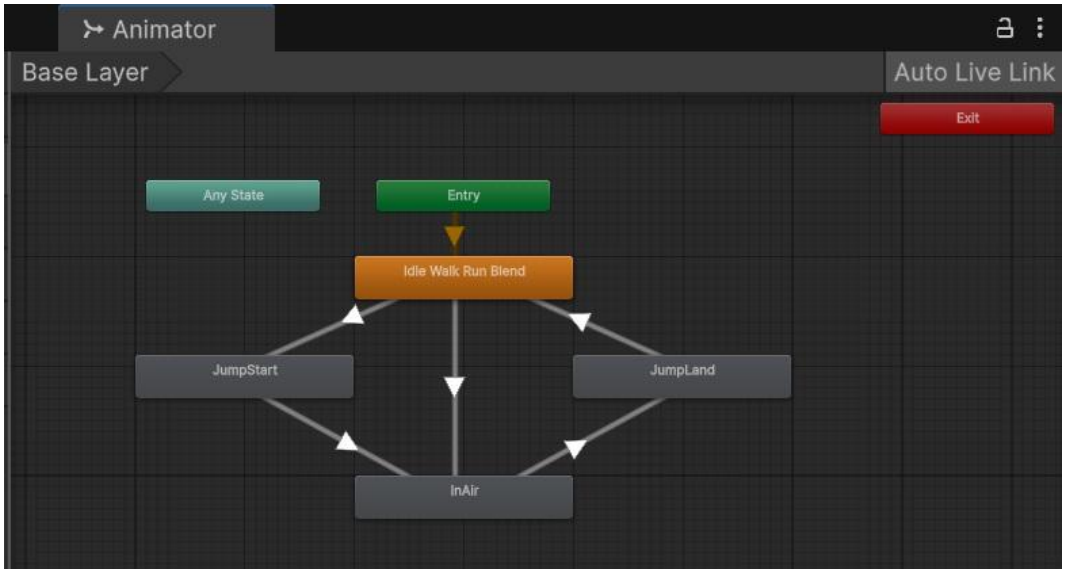

Figure 23. Inside the Animator tab

14. If this animation is set for the companion character, refer to section 6.3.3 for all further instructions. If this animation is for a townsperson, go to the Inspector window and attach the 'NPCRun' and 'SplineAnimate' scripts under the Animator component.
15. Using the "Draw Spline" tool on the left side of the Project window, click on the terrain to add nodes you want the character to travel to, and press the Enter key when you are done drawing the spline.
16. Select the townsperson again and set your new spline as the Spline parameter in their SplineAnimator script, set Loop Mode to Once, and change the Duration to the number of seconds you wish for them to spend moving. Testing will be necessary to find the best speed, but for this application, it was found that between 26 and 32 seconds is ideal.

6.5 Environment

6.5.1 BOM (Bill of Materials)

1. *Dry Trees* ($0.00) https://assetstore.unity.com/packages/3d/vegetation/trees/dry-trees-86967
2. *Dead forest* ($10.00). https://assetstore.unity.com/packages/3d/environments/dead-forest-190976
3. *Yughues bushes* ($0.00). https://assetstore.unity.com/packages/3d/vegetation/plants/yughues-free-bushes-13168
4. *Town house pack* ($0.00). https://assetstore.unity.com/packages/3d/environments/urban/town-houses-pack-42717
5. *UK terraced house pack* ($0.00). https://assetstore.unity.com/packages/3d/environments/urban/uk-terraced-houses-pack-free-63481
6. *Free Fire VFX - URP* ($0.00).

https://assetstore.unity.com/packages/vfx/particles/fire-explosions/free-fire-vfx-urp-266226

7. *Village house pack* ($0.00). https://assetstore.unity.com/packages/3d/characters/village-houses-pack-63695
8. *Fantasy forest environment* ($0.00). https://assetstore.unity.com/packages/3d/environments/fantasy/fantasy-forest-environment-free-demo-35361
9. *Low poly street pack* ($0.00). https://assetstore.unity.com/packages/3d/environments/urban/low-poly-street-pack-67475
10. *Rock Package* ($0.00). Rock package | 3D Exterior | Unity Asset Store
11. *AllSky Free - 10 Sky / Skybox Set* ($0.00). AllSky Free - 10 Sky / Skybox Set | 2D Sky | Unity Asset Store
12. *Stylized Water Texture* ($0.00). Stylize Water Texture | 2D Water | Unity Asset Store

6.5.2 Equipment list

1. *Unity Editor v6000.0.39f1* ($0.00). Unity Hub Installation

6.5.3 Instructions

1. In the Project window, select 'Scenes'. Right click inside of the folder where the menu in Figure 24 will appear. Select Create → Scene → Scene.

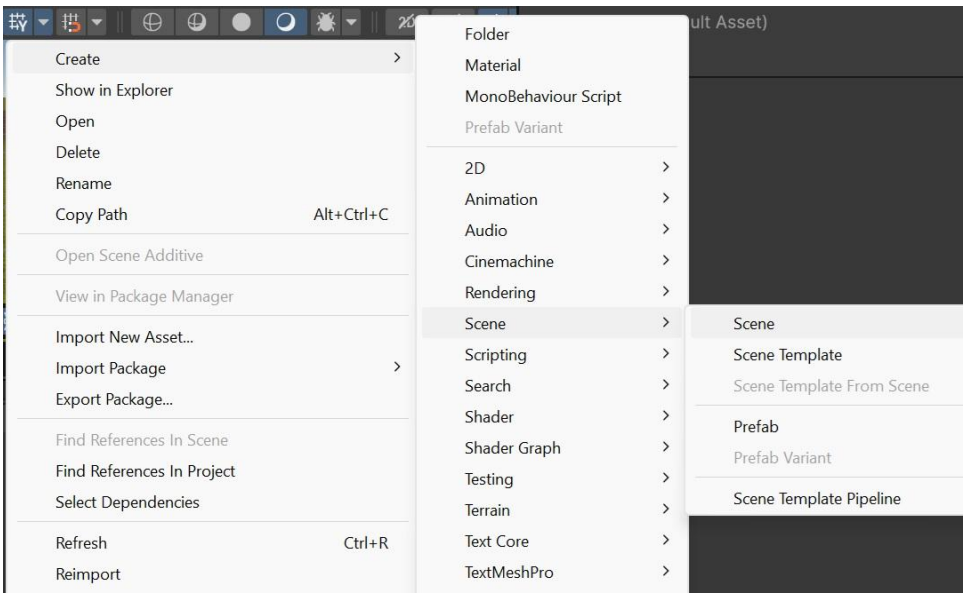

Figure 24. Navigating to create a new Scene object

2. Create five duplicates of the "BasicScene" from the VR template. To keep track, name them "StartMenu", "Scene_1", "Scene_2", "Scene_3", and "EndMenu".
3. Import all assets from the bill of materials into the project file through the package manager and open Scene_2.

4. Right click in the hierarchy, then select ‘Create empty’→ ‘3D object’→ ‘Terrain’ and name it “Terrain_1”.
5. In the project window, select Fantasy Forest Environment asset→ Scenes folder→drag ‘assets’ into Hierarchy.
6. Rename Fantasy Forest Environment to “Terrain_2” and select it. In the Inspector window select ‘Terrain’ component→ ‘Paint Terrain’→ ‘Paint Texture’→ ‘Edit Terrain Layers’ → ‘Create Layer’→ Select ‘grass01’ as the diffuse material.

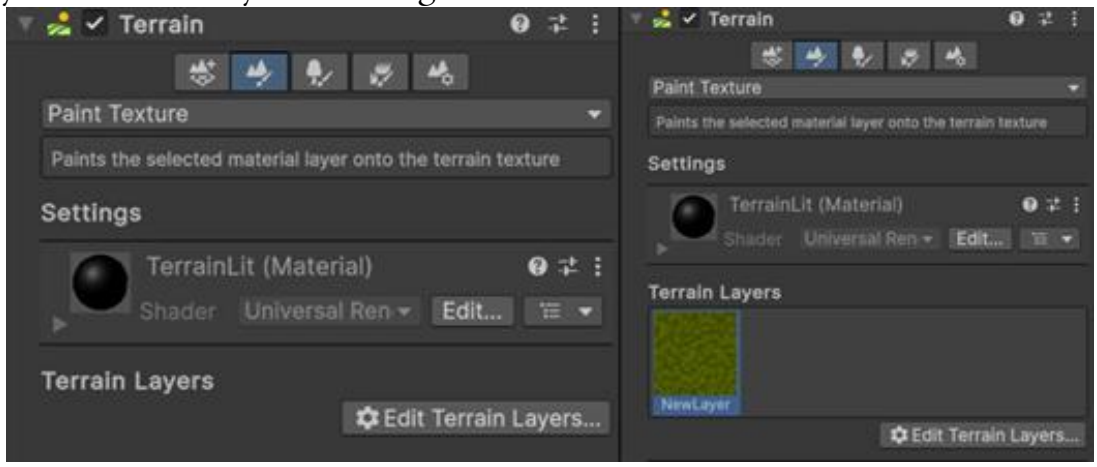

Figure 25. Grass terrain addition before and after

7. In Terrain_1, select ‘Terrain Settings’ → ‘Mesh Resolution’ → set Terrain Width to 400 → set Terrain Length 400 → set Terrain Height to 50.
8. In Terrain_2, select ‘Terrain Settings’ → ‘Mesh Resolution’ → set Terrain Width to 300→ set Terrain Length 300→ set Terrain Height to 20.

Figure 26. Mesh resolution settings for terrain sizing

9. In Terrain 1, select ‘Paint Terrain’ → ‘Paint Texture’→ change it to ‘Lower and Raise Terrain’.
10. Change brush size between 25-50 and opaqueness between 1-5 (Test to see which is most suitable), in scene view drag and slide brush sideways in order to raise terrain near the edges of the terrain.

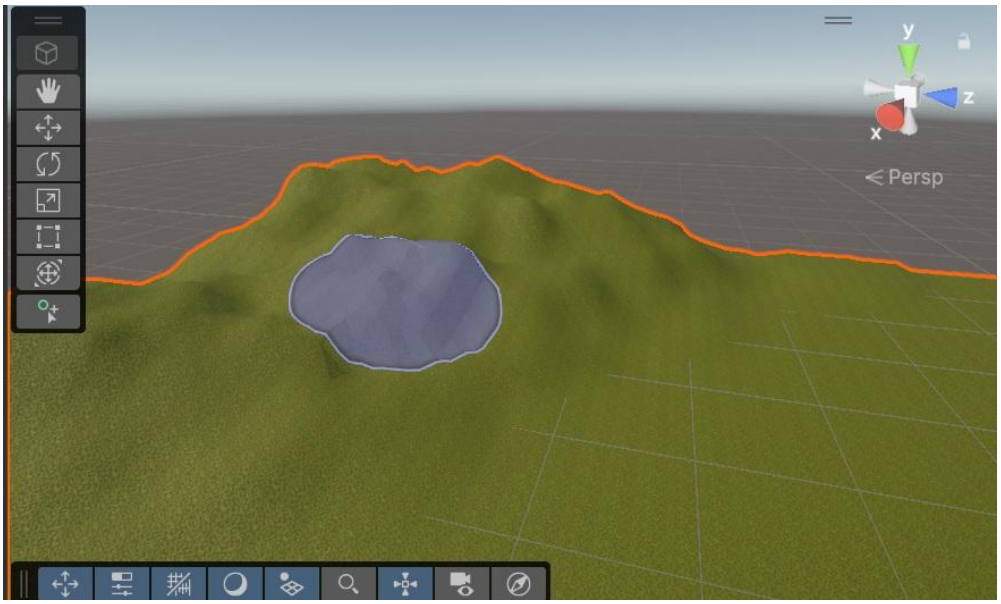

Figure 27. Raising and lowering terrain using the brush tool

11. In Terrain 2, select 'Paint Terrain' → 'Paint Texture' → 'Edit Terrain Layers' → 'Create Layer' → select 'dirt01'.

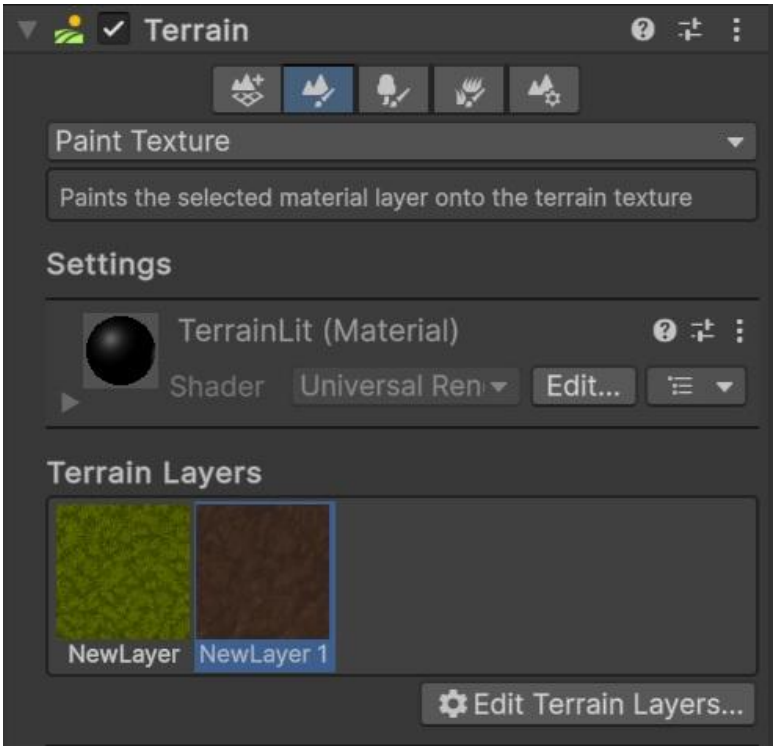

Figure 28. Dirt layer addition to terrain

12. Select ‘dirt01’ layer as shown in the figure above, then in the scene view drag the brush to create a dirt layer over the grass layer to create a path in Terrain 2.

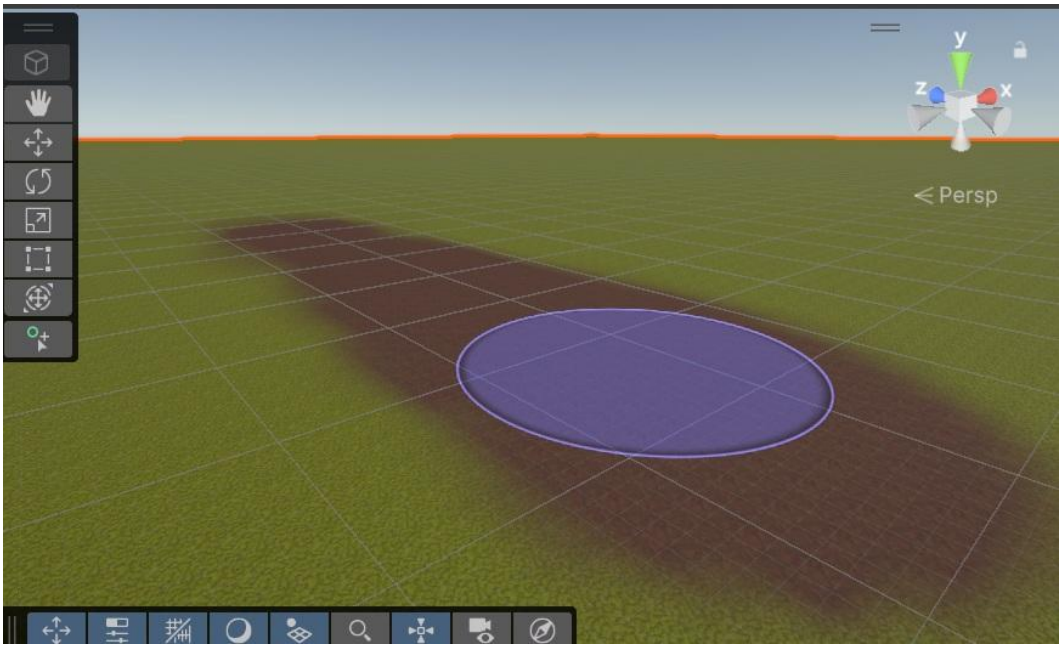

Figure 29. Layer addition in scene using the brush

13. In the project window, select ‘Fantasy Forest Environment' asset and select ‘Meshes’ file → prefabs → select ‘tree_1’.

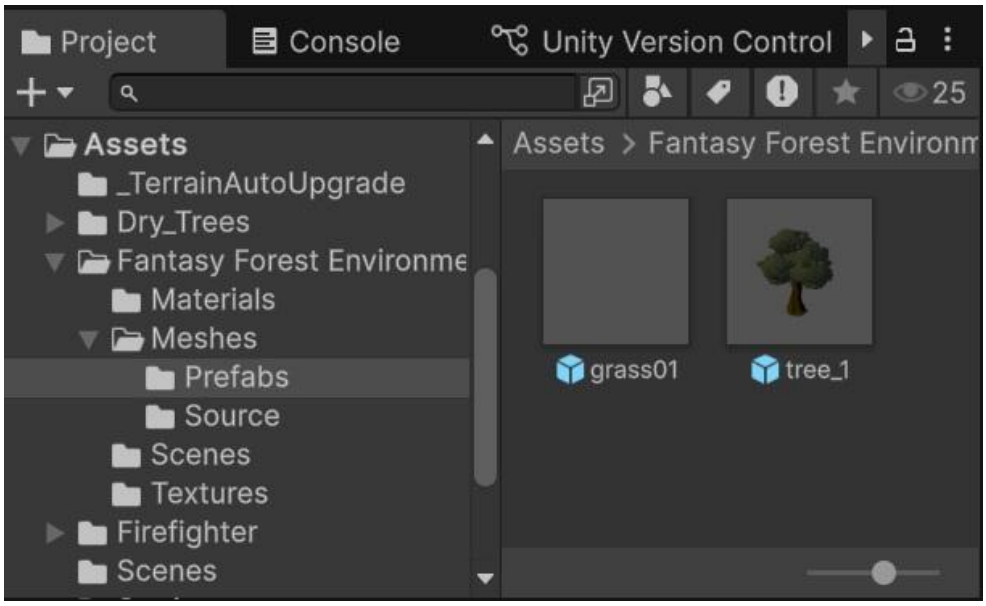

Figure 30. Tree prefab placement

14. Drag and place ‘tree_1’ into terrain. You will need to place around 800-1000 trees in order to fully populate the forest and surrounding hill area in both Terrain 1 and Terrain 2.
15. Drag and place rocks, bushes and dead trees into the project and customize the area you would like the scene set up in.
16. After setting up Scene 2, open Scene 3 and drag Scene 2 icon into the Hierarchy, then copy and paste all elements into Scene and then remove Scene 2.
17. Head to the Project window, select ‘UK Terraced House’ and ‘Town Creator Kit’ assets and drag the prefabs into the scene window.
18. If the size of the houses is too large, navigate to the Inspector window → Transform→ Scale and decrease the values in that row.

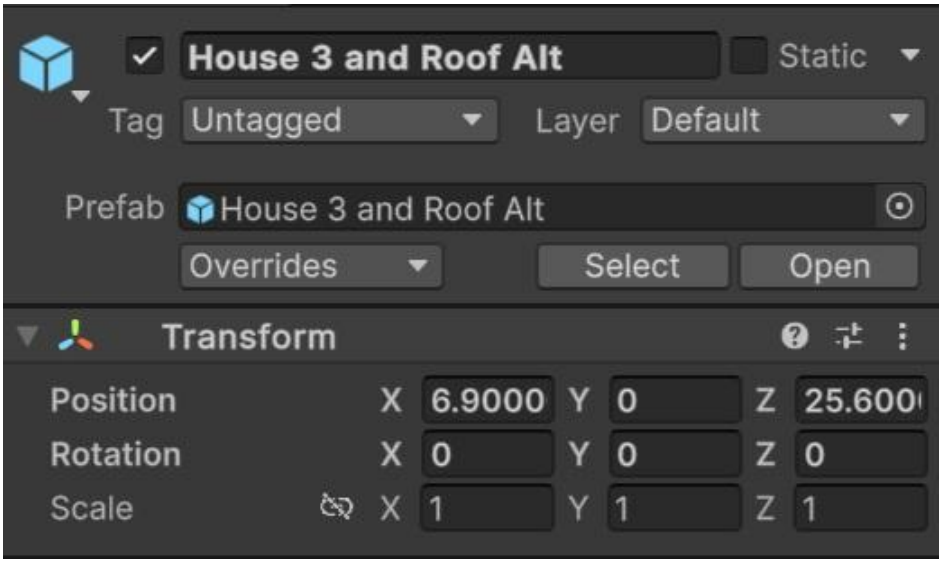

Figure 31. Inspector window 3D object transform

19. After setting up the order of the houses, create a layer of dirt in the middle of the area, referring back to step 11.
20. Go to scenes and select Scene 4(last scene) and copy paste Scene 2 into Scene 4 similar to step 16.
21. Import and use sketchup model houses as shown in section 6.8 Sketchup House Models.
22. Use the ‘Low poly street pack’ asset and place prefabs for the road, street signs and accessories.
23. After customizing the town and forests, setup scene switcher. Open the StartMenu scene, and from the Prefabs folder, drag an XRSceneChanger into the Hierarchy window.
24. Still in the Hierarchy window, navigate to XR Origin (XR Rig) → Camera Offset → Main Camera → Survey Menu, then drag the XRSceneChanger into the Scene Changer parameter of the MenuCycler script.
25. Create a new Empty object outside this tree and attach the LocaleSelector script to it.
26. Open each of Scene_1, Scene_2, and Scene_3 and add a SceneController prefab to each. Configure each as follows:

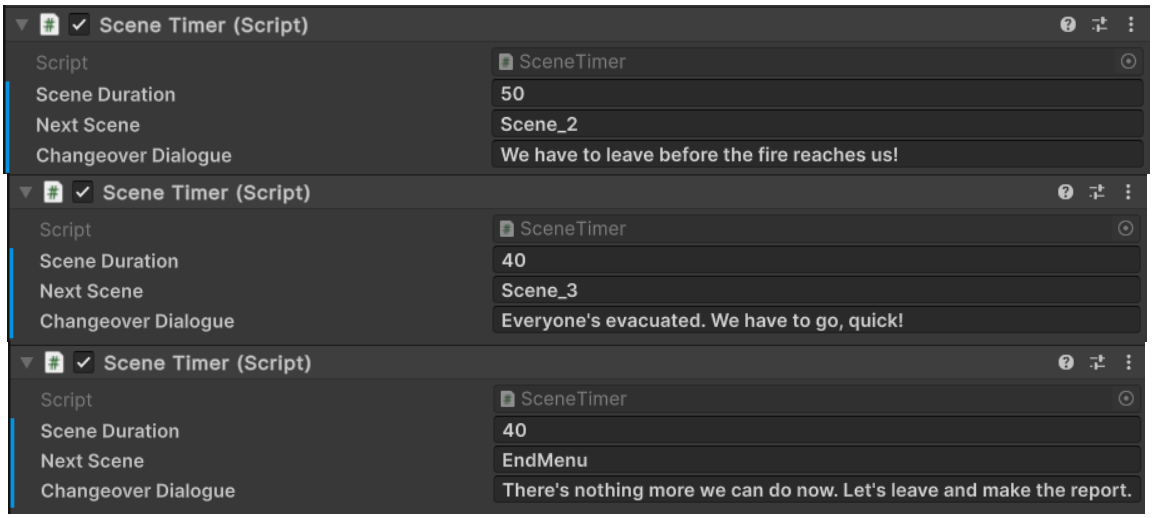

Figure 32. Scene timer configuration settings

27. Finally, to ensure the companion character is able to navigate, check on Static in the top right of the Inspector window for all objects the companion should go around.
28. Select one of the Terrain objects and add a NavMesh Surface component to it in the Inspector window. Press Bake to allow it to calculate the pathing surface for the companion.

6.6 Fire

6.6.1 BOM (Bill of Materials)

1. *Free Fire VFX - URP* ($0.00). https://assetstore.unity.com/packages/vfx/particles/fire-explosions/free-fire-vfx-urp-266226
2. *Starter Assets: Character Controllers | URP* ($0.00). https://assetstore.unity.com/packages/essentials/starter-assets-character-controllers-urp-267961

6.6.2 Equipment list

1. *Unity Editor v6000.0.39f1* ($0.00). Unity Hub Installation

6.6.3 Instructions

1. Open a scene you wish to add fire to; either Scence_1, Scene_2, or Scene_3.
2. In the environment section of the Hierarchy, create two cubes and name one “SpawnTrigger” and the other “DestroyerTrigger”.

Figure 33. Hierarchy window with two cube triggers

3. Create an Empty object under the SpawnTrigger cube and name it “spawnpoint1”.
4. Attach the “Spawn Trigger” script to the cube with the same name, and the “Destroy Trigger” script to its box.
    a. The script was a modified spawn and destroy script found in this video
5. Just underneath the cube’s name in the Inspector window, create a SpawnTrigger and DestroyTrigger tag and assign them to their respective cubes.
6. In the script section for the SpawnTrigger, apply the empty spawnpoint1 created to the “Spawnpoint” section of the script, and the particle that’s going to be spawned in (the fire in this case) under the “Prefab” section. Refer to Figure 32 for configuration.

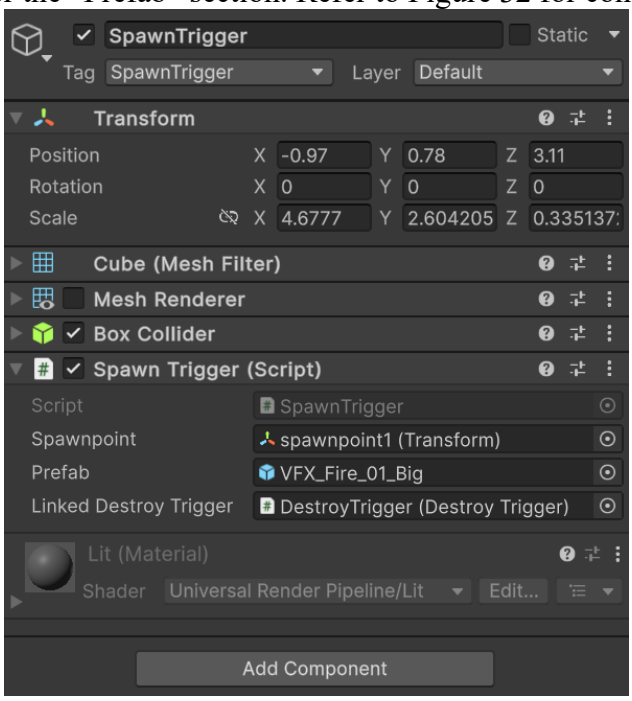

Figure 34. Inspector window view of the fully configured SpawnTrigger script

7. Repeat this process for any other triggers you wish to create, adding a number or symbol after the name to keep them in pairs.

6.7 Sound

6.7.1 BOM (Bill of Materials)

1. *Voiceover files* ($0.00). V.ORB (Team 3) - Wildfire Inspector | MakerRepo

6.7.2 Equipment list

1. *Unity Editor v6000.0.39f1* ($0.00). Unity Hub Installation
2. *Eleven Labs TTS AI voice* ($0.00). AI Voice Generator

6.7.3 Instructions

*Note: This process should begin only after finalizing and appropriately separating all written dialogue lines. All voice lines necessary to recreate the simulation as presented have been included in the project files as linked in the bill of materials.

1. Go to the Eleven Labs website linked in the equipment list and create an account following the instructions provided on the site.
2. Under the "Voice" category, search and select the "Adam" voice.
3. Type in the voice lines in the text box and adjust the sliders until you are satisfied with the output voice.
4. Download all of the voice lines and store them in a folder for ease of access and transfer.

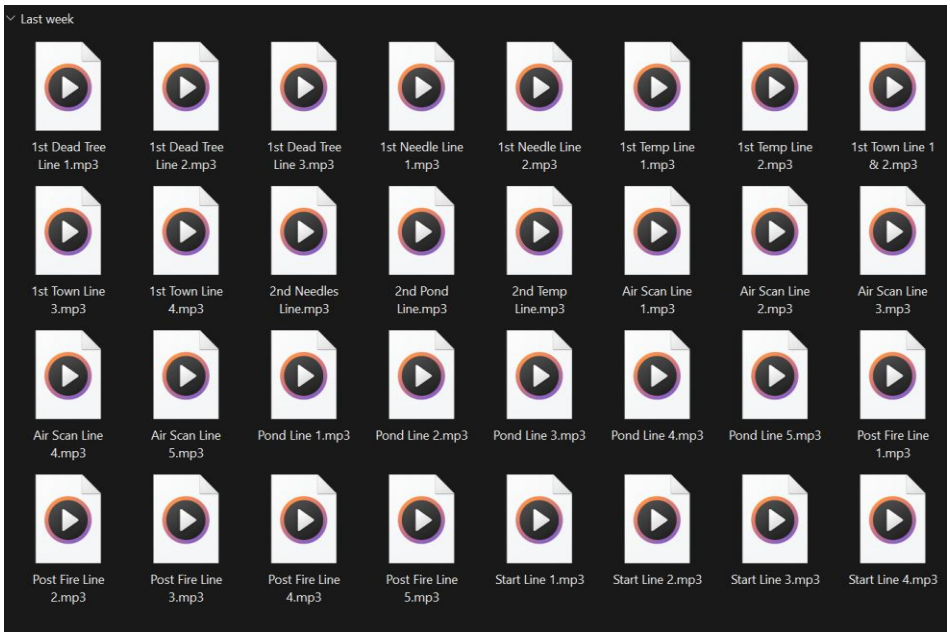

Figure 35. All of the voice lines needed for the simulation

5. Move the folder into the project's Asset folder or, from within the Unity editor, right click in the Assets window and select 'Import New Asset'. Navigate to and select the Voicelines folder to import all voice files to the project.
6. If the XRSceneChanger prefab does not have all of the lines in its Voiceover List script filled out, it is simplest to navigate to the prefab from the Assets window, the right click it and select 'Reimport' to fill in the fields with the newly added voice lines.

6.8 Sketchup House Models

6.8.1 BOM (Bill of Materials)

1. *Single family house model* ($0.00). house - Model - 3D Warehouse
2. *Small house model* ($0.00). house - Model - 3D Warehouse
3. *Modest house model* ($0.00). House - Model - 3D Warehouse
4. *Simple house model* ($0.00). Simple House - Model - 3D Warehouse

6.8.2 Equipment list

1. *Unity Editor v6000.0.39f1* ($0.00). Unity Hub Installation

6.8.3 Instructions

Choose any one of the house models listed in the bill of materials to follow the instructions with. The procedures are identical for each.

1. In the Assets folder, create a new folder to contain all the Sketchup house models. Within that, create a folder for your chosen model and navigate into it.
2. Download the model file (.skp) from Sketchup, then return to Unity and, in the house's asset folder, click RMB → Import New Asset… and choose the house's file.

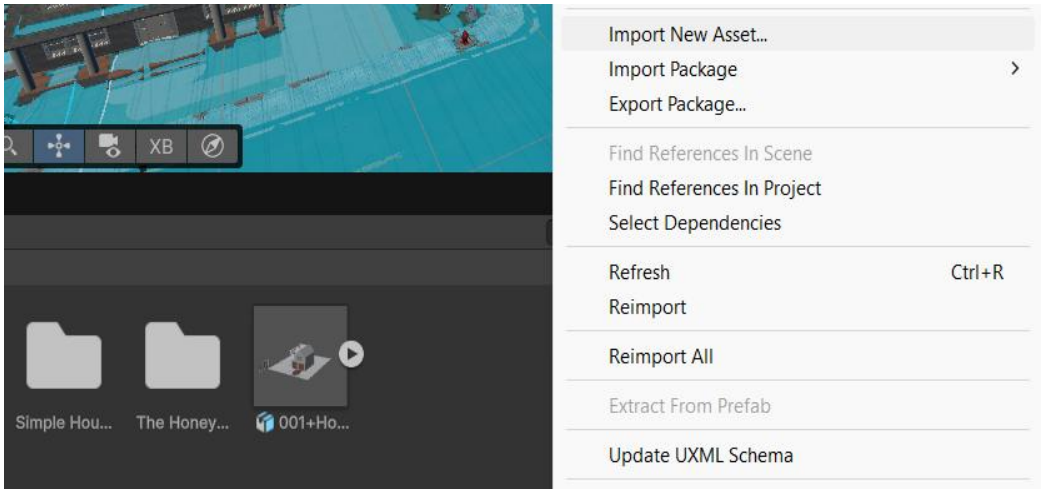

Figure 36. Asset import button on right click

3. Click on the model in the Assets folder, then click on “Extract Textures” then “Extract Materials” in the Inspector window. Both times, click “Select Folder” on the pop-up window.

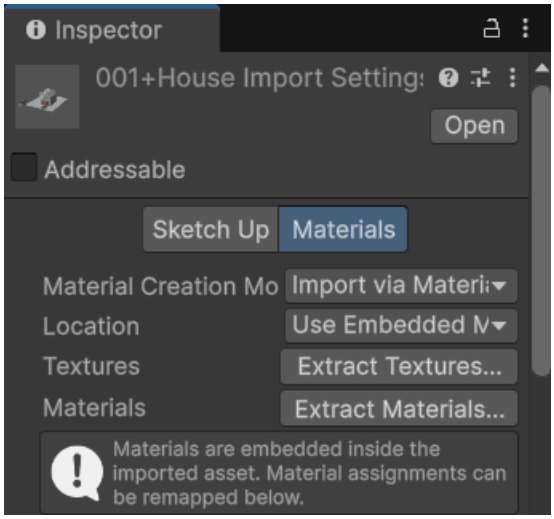

Figure 37. Import buttons for a Sketchup house in the Inspector window

4. For each texture extracted, go to the corresponding material and assign the given texture as its base map.
5. Select the model again, then drag and drop each material from the Assets folder to its corresponding parameter on the house in the Inspector window.

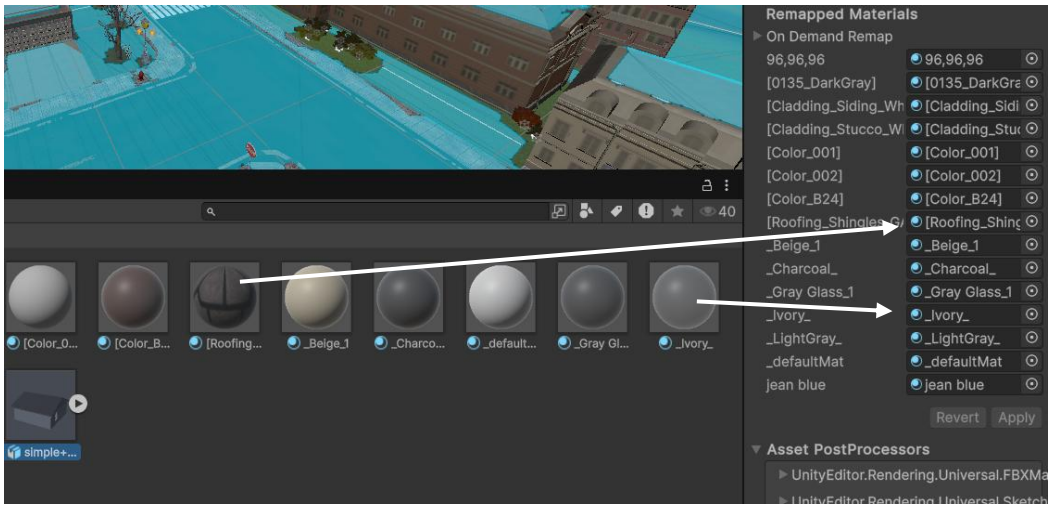

Figure 38. Directions to assign materials to a Sketchup model

The house model prefab can now be dragged and dropped into the scene as a regular model.

6.9 Testing & Validation

The initial planning and budgeting is detailed in (PD-E), while the first, second and third prototype iterations are documented in (PD-F), (PD-G), and (PD-H), respectively. See the appendix in section 9 links to the referenced documents.

6.9.1 Fire

Fire Prototype 1

The first fire tests focused on the overall visual appeal of the particle effects along with how the light would interact with its surroundings. This was done by using the Fantasy Forest environment (see Section 6.5.1, item 8) with two different fire effects placed from two different packages. The first, an asset titled Particle Pack, had two notable issues, the first being that it did not fit the visual style that aligned with the intended tone, and the other being that the particles themselves kept disappearing without an identifiable cause. The other pack was Free Fire VFX - URP (see Section 6.6.1, item 1), which better the intended realistic style, caused no apparent bugs, and came with built in sound and smoke effects.

The other half of the prototype was the lighting. This was done by using the built in "Point Light" tool in Unity and attaching a flicker script to the object, which would periodically alter the light intensity to simulate how a real fire would act. The light would then be layered directly over the fire to act like the fire is simulating the light itself. Both of these tests were successful in showing how immersive the fires would eventually be in the final simulation.

Figure 39. VFX particles testing the fantasy environment

Fire Prototype 2

The second fire tests revolved around creating a very basic Spawn/Despawn trigger to control how the fire works around the simulation. The way it was implemented was by having an invisible cube with its collision set to ‘Is Trigger’ to activate the ‘SpawnDestroy’ script in Figure 40 (see below). This script uses an empty object as the spawnpoint location, defining where the fire will be in the scene. The prefab section is for what the particle will be, i.e. a small fire, a big fire, etc. Walking through the cube spawns the fire, and walking through it a second time gets rid of it. This was initially a success as it could spawn and despawn particles using the cube trigger. On starting the third prototype, though, two critical flaws were found with the spawning script. The first issue was that, as coded, only one instance of the script could exist in the simulation, meaning there could only be one trigger for spawning and destroying fire objects. The other issue is caused by the fact that the one cube both spawns and despawns the particles, meaning that the user has to walk through the same area twice for the full effect to occur. For these reasons, this prototype would be considered a failure in retrospect.

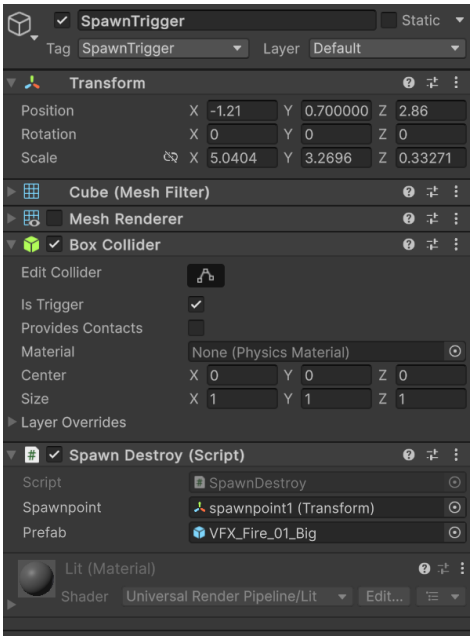

Figure 40. The first implemented spawning trigger, "SpawnDestroy"

Fire Prototype 3

With the issues present in the second prototype, the third aimed to create a proper spawning and despawning trigger. Video tutorials were used to learn how to program in C# and rebuild the activation script. This resulted in the spawning destroying scripts being made as separate objects, which worked for pairs of spawning and destroying trigger cubes using a setup otherwise identical to that of Fire Prototype 2. The spawn trigger works almost the exact same way in prototype two, but the destroying part instead has to be assigned to a new section in the inspect element. This new element is what makes it so you can have multiple different pairs of triggers in one simulation, by linking another cube with the destroy script attached to its

respective spawn trigger. This final implementation of these fire triggers is what was used in the final scenes of the simulation.

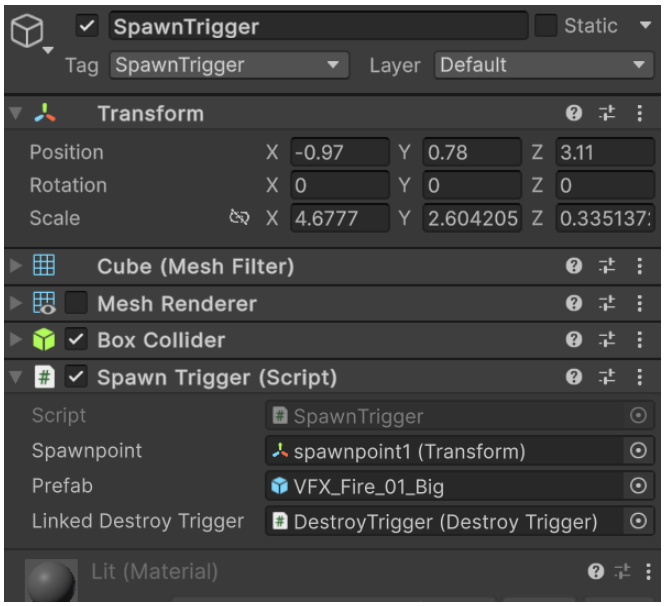

Figure 41. The final version of the spawning and despawning trigger implemented

6.9.2 Character movement and animation

Character Model and Movement Prototype 1

The first prototype was focused on choosing a character model that accurately depicts a fire inspector, which was a simple matter of browsing the Unity asset store and procuring a fitting model and proved to be a quick success (see Section 6.4.1, item 1).

Following this was the first stage of configuring the character's movement and animations. The first test, I used a very simple code for movement. Initial tests successfully implemented walking, running, and jumping motions using keyboard controls. Some issues with the character sinking through the ground arose but were able to be solved by attaching colliders to both the character and terrain. Testing arm and leg animations caused further issues as the character's animations were not playing properly based on its movement state while the character also drifted horizontally with no apparent cause. An attempt was made to fix this by importing a new character movement asset, which fixed the animation issues but did nothing for the troubles with randomly floating and sinking. As movement could not successfully be implemented with the firefighter model, a default placeholder model was used to test the animations.

Figure 42. Fire inspector character model (left) and placeholder character (right)

Character Animation Prototype 2

The second prototype was successful in implementing character movement for the fire inspector model. This was done by attaching the fire inspector model over the placeholder character's animation rig. The movements, using the character controller asset, were reliable when tested on sloped and elevated terrain as well as between scenes. The next development was to merge this with the first person perspective movement and equip the character with the scanner tool. Following this was the companion character, as shown in Figure 43 (left). The character comes dressed in safety equipment and can also wear a helmet and a mask, making him appear well-suited for the role.

Figure 43. Character model of companion character (left) and player view (right)

Following selection of the user and companion models, the next test was to implement the civilian evacuation, such that they run away when the user arrives in the scene. The characters' animations are intended to be similar to those of the user and companion, but instead take them along a predefined path. The first implementation of this faced issues with the character speed being too slow and their animations not activating. The second test added a "speed" parameter and set it to 1, but this failed to alter the previous behaviour. For the third test, the controller was changed from freehand to starter controllers, though this still did not fix the issue.

Figure 44. Townspeople character models

Character Animation Prototype 3

In the third prototype, after testing prototype 2 in VR, it was decided to discard the orange suited companion character and instead keep the original fire inspector model to be used as the companion character. This was decided because the XR rig replaces the need for a body as we only need the user to be using the scanner tool which would be equipped on the VR controller which gives him the ability to scan assigned objects. The companion character was set up with a third person controller that allows him to move alongside the user.

After further visual tests on prototype 2, we decided to change the evacuating townspeople from the old town setting. The new characters and buildings are intended to make the time difference between the two immediately apparent. These characters are equipped with an automatic movement script that allows for them to appear to run in different directions.

Figure 45. Updated model of townspeople

6.9.3 Sound

Sound Prototype 1

The first prototype for sound focused on testing the sound quality of two different asset packs that were chosen to be a part of the simulation. The first was Minimal UI sounds, which only had approximately 8 sounds, none of which were of adequate length or quality for what was needed. The other pack, Casual Game Sounds, had a lot more to offer with roughly 40 sounds, most of which were of good quality and suitable style for the intended setting. This prototype was partially successful as only the latter sound pack was useful.

Sound Prototype 2

The second audio prototype was layering the audio from Casual Game Sounds over the main interaction element, the scanner tool. This was done by taking some of the sounds that would most fit the look and feel of the scanner tool and putting the sounds on top of the different actions of the tool in an editing software, Shotcut. {The work done for this was not kept for the final prototype as it was expected that the sounds may be changed later in development, but it was successful in finding how well the sounds would work with the scanner tool.

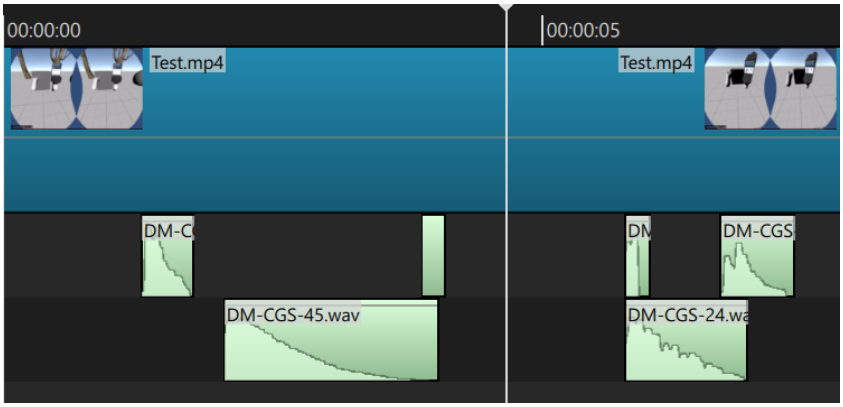

Figure 46. Sound being layered over the scanner tool

Sound Prototype 3

The third and final sound prototype was creating the dialogue for the simulation. This was done by using the Eleven Labs Adam text to speech voice and tuning the settings to get the right tone, volume, and pace of speech. The voice lines were taken from a script that was prepared in advance of the second prototypes, greatly simplifying the process. The difficulty faced with this was that, after the dialogue lines had to be re-cut for their text to fit in the dialogue box, the lines all had to be re-recorded. Even with that, the prototype was a success, and all lines were transferred for use in the final simulation (see Section 6.7).

6.9.4 Environment

Environment Prototype 1

The first tests were focused on importing and integrating environmental assets, including terrain, trees, ground textures, fire, and lighting. The lighting was adjusted to enhance depth and realism, ensuring fire sources interacted naturally with the scene. The ground textures were evaluated for consistency with the dry forest setting, with minor refinements made where necessary. Fog and rain assets were also added to the environment to have a more immersive feeling. Testing the scale of the character in the environment was not possible at this stage as the team was yet unable to share Unity files.

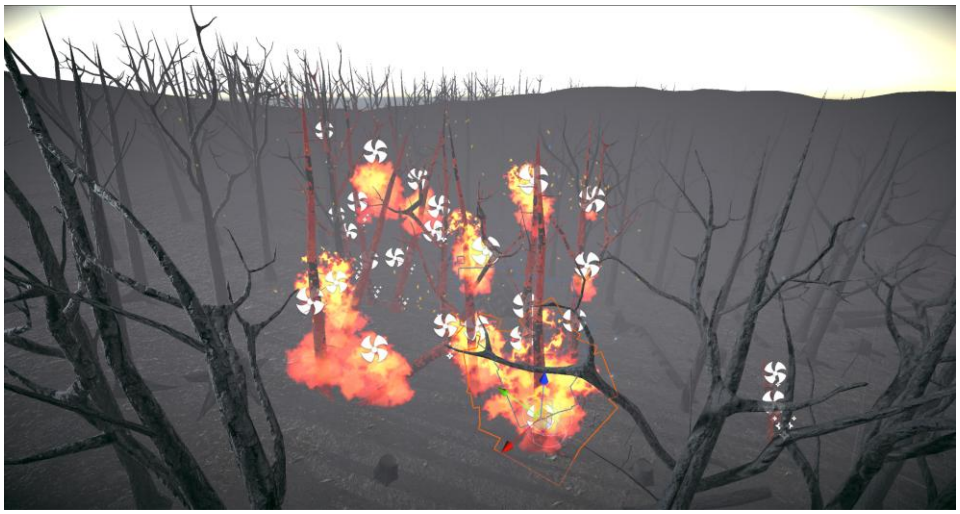

Figure 47. Bird's-eye view of the burnt forest scene, showing fire effects and particles sources

Figure 48. Ground level view of the forest scene, closely modelling the intended environment

Environment Prototype 2

The second prototype prioritized completion of the town setting to be used for the second scene. Several assets were imported such as street, house and accessory assets in order to increase the realistic look of the town. The first test for this focused on fixing the sizes of the items and making them proportional to the character while also adding colliders and setting up their positions. The second test was focused on lighting as shadows appeared disproportionately large compared to the objects.

Figure 49. Further development of the town scene

Figure 50. Early model for the forest scene

Environment Prototype 3

With the issue of file sharing resolved, the character and scanner tool were successfully able to be integrated into the environment. This was tested with success in the burnt forest environment using XR controls, with colliders added to prevent the user from unintentionally clipping through the trees and modifications made to the scale of environmental objects to keep the setting as realistic as possible.

Figure 51. Dead forest environment with the scanner tool in the simulated VR view

The old town model was developed further in this prototype by adding hills and increasing tree density in the forest and surrounding area. This environment was completed in the third prototype with the town being made ready for character interaction by the addition of colliders on the buildings. The dead trees in the pre-fire environment were successfully configured to be intractable with the scanner tool.

Figure 52. Old town editor view

Figure 53. Forest trail environment for the pre-fire scene

7 Conclusions and Recommendations for Future Work

This project was an exercise not only in learning to develop in the Unity engine, but to manage time and work as a group towards a long-term objective. Through the months, the group became better able to communicate and schedule time to collaborate on documentation. The workload demanded by this course provided a hard lesson on the value of thinking and working ahead on tasks to avoid having to crunch everything in the last couple days before they're due. Some of the features we had wished to implement had to be culled from the pipeline for lack of time and for other features taking priority in criticality.

One such feature was the localization option, which intended to expand the audience beyond just those fluent in English, but the method implemented in the early menu prototypes was not compatible with the way the textual dialogue was implemented, nor would it have done anything for the audio voice overs. Future work would have to build the system from the ground up, heavily revamping how the Localization package is used and writing new scripts to assign the use of French or English audio based on the starting menu language selection. Another element that was sorely felt in user testing at the end was the lack of tutorial or explicit direction. Confusion arose regarding how to progress the story, how to scan items, where to go, and even how to navigate the menus, which may have been alleviated had there been more time and more tests on how users engaged with the early prototypes. Finally, the broad scope of the project, with regards to what can be scanned, how much data is available to the user, how compelling the companion dialogue is, and how much of the world is observable and immersive were things we would have hoped to expand, but did not have the resources to do. This does not enhance the accessibility of the simulation as the other two would, but would be important to better communicate the message of how seemingly disparate climate factors converge to drive worsening wildfires and how that's going to harm people.

The in-engine development time of the simulation was only around a month and half, which largely focused on learning Unity from scratch. With another few months and the experience already gained, the simulation could be vastly improved, though the precise avenues chosen for this would depend on whether the original time constraint was lifted. The given 1-3 minutes was appropriate for the development time available and the speed of character movement and dialogue fit this appropriately, so continuing to work within this limit, the highest priority elements would likely be reworking scenes to improve system performance and display a more realistic setting as well as reworking the visual effects to better communicate the tone. Without the same time constraint, it would be beneficial to develop a more cohesive and compelling narrative with the inspector and deputy characters to allow the user to feel more attached and empathetic to the situation.

8 Bibliography

1. Unity sign-in page. Unity ID Account Login | Sign in. Accessed 12 March, 2025.
2. UBC Forestry, "Exploring the Arctic's Climate Crisis Through Virtual Reality", 22 June 2023, Available at https://forestry.ubc.ca/news/arctic-climate-crisis-through-virtual reality/
3. Z. Xu, Y. Liang, A. Campbell, and S. Dev. "An Explore of Virtual Reality for Awareness of the Climate Change Crisis: A Simulation of Sea Level Rise", IEEE 2022 8th International Conference of the Immersive Learning Research Network, 11 July 2022, Available at https://ieeexplore.ieee.org/document/9815983/
4. E. Erisen, F. Yildirim, E. Duran, B. Șar, and I. Kalkan. "Exploring the effectiveness of virtual reality in combating misinformation on climate change", Political Psychology, 00, October 2024, pp. 1-29, Available at https://doi.org/10.1111/pops.13057
5. D. Markowitz and J. Bailensen. "Virtual reality and the psychology of climate change", Current Opinion in Psychology, V.42. December 2021, pp. 60-65. Available at https://doi.org/10.1016/j.copsyc.2021.03.009
6. S. Thoma, M. Hartmann, J. Christen, B. Mayer, F. Mast, and D. Weibel. "Increasing awareness of climate change with immersive virtual reality", Frontiers in Virtual Reality, V.4. 2023. Available at https://doi.org/10.3389/frvir.2023.897034

APPENDICES

9 APPENDIX I: Design Files

The documents listed below record the complete engineering design process of this simulation, from first empathizing with the client's needs to working out the final technical elements. All documents as well as the ZIP file containing all referenced custom scripts can be found at the project's MakerRepo page, linked below for each document.

Table 3. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
GNG1103 – VR Climate Change Simulation (PD-B)	Under the Project Files tab at V.ORB (Team 3) - Wildfire Inspector | MakerRepo	26 Jan. 2025
GNG1103 – Deliverable C: Design Criteria and Specifications (PD-C)	Under the Project Files tab at V.ORB (Team 3) - Wildfire Inspector | MakerRepo	2 Feb. 2025
GNG1103 – Conceptual Design for a VR Climate Change Simulation (PD-D)	Under the Project Files tab at V.ORB (Team 3) - Wildfire Inspector | MakerRepo	9 Feb. 2025
GNG1103 – Project Schedule and Development Cost for a VR Climate Change Simulation (PD-E)	Under the Project Files tab at V.ORB (Team 3) - Wildfire Inspector | MakerRepo	16 Feb. 2025
GNG1103 – Client Feedback and First Prototype for a VR Climate Change Simulation (PD-F)	Under the Project Files tab at V.ORB (Team 3) - Wildfire Inspector | MakerRepo	2 Mar. 2025
GNG1103 – Client Feedback and Second Prototype for a VR Climate Change Simulation (PD-G)	Under the Project Files tab at V.ORB (Team 3) - Wildfire Inspector | MakerRepo	9 Mar. 2025
GNG1103 – Client Feedback and Third Prototype for a VR Climate Change Simulation (PD-H)	Under the Project Files tab at V.ORB (Team 3) - Wildfire Inspector | MakerRepo	16 Mar. 2025