Mixed Reality Configurator Technical Report

CSCI 6838 Research Project and Seminar

Instructor Dr. Said Bettayeb

Mentor Dr. Ross Niswanger

Naga Vennela Kandru
kandrun0712@uhcl.edu

Alekhya Kotha
kothaA0483@uhcl.edu

Mason Lanham
lanhamm4224@uhcl.edu

Hari Charan Mukthineni
mukthinenih9340@uhcl.edu

November 25th, 2024

School of Science and Computer Engineering

University of Houston-Clear Lake

2700 Bay Area Boulevard

Houston, Texas 770

Introduction

The Mixed Reality Configurator is software made using the Unity game engine for the Microsoft HoloLens 2 platform. It includes configurators for a shoe, watch, and animal models. Each configurator allows the user to change the color of different components of the model that they are viewing. In the case of the plush toy animal configurator, the user is able to choose which animal model they want to configure as well as which textures and colors to apply to which parts of the animal.

The purpose of these configurators is to record data about the interactions that the user has with the model and with the system. The user's actions within the framework of the configurator as well as manipulation of the model that they configure are recorded along with the timing of events and other important details for later analysis and comparison against data gathered from other configurators outside of our project.

To facilitate the gathering of interaction data, users are split into 2 categories: surveyor and respondent. The surveyor is responsible for setting up the software on the HoloLens to be in a state where it is appropriate for the respondent to take control of the HoloLens. The respondent is responsible for completing the configuration process. It is their interactions with the software that are recorded. After the respondent completes configuration, the surveyor can restart the Mixed Reality Configurator to begin another data recording session, or connect it to their device to retrieve the recorded data.

What is a product configurator?

A product configurator is a tool that enables the respondent to personalize products such as a watch or shoe. The Mixed Reality configurator enhances this by integrating it into a mixed reality environment, making it possible to interact with holographic models in a more natural and interesting way.

Why compare 2D, 3D and Mixed Reality Configurators?

Comparing 2D and 3D configurators can allow for understanding respondent preferences when it comes to product customization. The 2D configurator has a good interface with static images and simple menus, making it ideal for quick customization. However, it does not have the depth of a 3D configurator, which offers an immersive experience by enabling a respondent to zoom in, rotate, and explore objects from multiple angles. The Mixed Reality Configurator extends this premise by turning the configurable objects into holograms. It is possible to evaluate key factors like respondent customisation time, and engagement levels.

Requirements Analysis

Definitions

Model: The model is the holographic 3D object that the respondent configures and interacts with.

Component/Part: A component is a part of the model that can have its color changed independently of other components. Together these components make up the Model.

Event: An event is an action that the user takes that is recorded.

Respondent: The respondent participates in the survey by performing the configuration process.

Surveyor: The surveyor sets up the configurator, and records the responses of the respondent.

❖

ID	Story Summary	Acceptance Criteria	Priority	Estimate
SC1	As a surveyor, I want to select the desired configurator (shoe, watch, or animal) so that the respondent can configure the appropriate model.	The surveyor can select a configuration option. The software is in the start menu. The respondent begins configuration.	High	2
BC1	As a respondent, I want to start the configuration process to ensure my interactions are recorded accurately.	The start button is visible and functional. The configurator begins timing when selected. Models load and menus appear.	High	1
CC1	As a respondent, I want to change the color of individual components of the model to customize its appearance.	Respondents can select colors from a menu. Components update in real time. Changes are recorded as events.	High	1

CC2	As a respondent, I want to modify the texture of animal components to personalize the model further.	Respondents can select textures. Texture updates are highlighted and recorded as events.	Medium	1
SA1	As a respondent, I want to select a specific animal model to configure it according to my preferences.	Animal models are displayed. Selected models load and configurator menus update.	Medium	1
T1	As a respondent, I want to touch the 3D model to rotate and move it for better customization visibility.	Models respond dynamically to touch. Touch interactions are recorded.	Low	1
ES1	As a respondent, I want to end the configuration session when I am done.	The submit button displays a confirmation popup. The session ends when "yes" is selected.	High	1
RD1	As a surveyor, I want to retrieve the configuration session data for analysis.	Data is accessible via the Windows Device Portal. Files include recorded events, timestamps, and configurations.	High	1

Use Cases

UC-1: Select Configurator

Summary
The surveyor selects which configurator that they want the respondent to use.

Rationale
Before the respondent can configure the model they will be working on, the surveyor should select it.

Users

The surveyor will be able to use this functionality.

Preconditions

1. The Mixed Reality Configurator should be running in the surveyor menu state with the surveyor using the HoloLens.

Basic Course of Events

1. The surveyor points to and selects the buttons of which configurator the respondent will use.
2. The surveyor points to and selects the submit button advancing to the configurator start menu.
3. The surveyor hands the HoloLens to the respondent, and they put on the HoloLens.

Alternative Paths

None

Postconditions

The software is able to begin configuration with the respondent's touch of the configurator start button.

UC-2: Begin Configuration

Summary

The respondent begins the configuration process.

Rationale

We want precise timing on how long the user spends in the configuration process, so they need a way to start the process and start timing.

Users

All users will be able to use this functionality.

Preconditions

1. The software should be running in the configurator start menu state.

Basic Course of Events

1. The respondent points to and selects the Start button.
2. The 3D model(s) load.
3. The configurator menu appears to the left of the respondent.
4. The time of the start is recorded as an event.

Alternative Paths

If the model is an animal, the animal select menu appears instead of the configurator menu upon completion of this use case.

Postconditions

The model(s) is/are viewable and configurable via the configurator menu or animal select menu.

UC-3: Change Component Color

Summary

The respondent selects a different color for a component of the model than the one that is already selected.

Rationale

Part of the configuration process is the user being able to change component colors.

Users

All users will be able to use this functionality.

Preconditions

1. The software should be running in the configurator menu state.

Basic Course of Events

1. The respondent points to and selects a color for a component from the floating menu.
2. The component changes to the selected color.
3. The color, component, and time are recorded as an event.

Alternative Paths

None

Postconditions

The Model is updated to represent the change in component color.

UC-4: Change Component Texture

Summary

The respondent selects a different texture for a component of the Model than the one that is already selected.

Rationale

Part of the configuration process for the animal is the user being able to change component textures.

Users
All users will be able to use this functionality.

Preconditions

1. The software should be running in the configurator menu state with the Model as an animal.

Basic Course of Events

1. The respondent points to and selects a texture for a component from the floating menu.
2. The component changes to the selected texture.
3. The texture is highlighted in the menu as the selected texture.
4. The texture, component, and time are recorded as an event.

Alternative Paths

None

Postconditions

The Model is updated to represent the change in component texture.

UC-5: Select Animal Model

Summary
The respondent selects a different animal model than the one that is already selected.

Rationale
Part of the configuration process for the animal is the user being able to choose which type of animal they wish to configure.

Users
All users will be able to use this functionality.

Preconditions

1. The software should be running in the animal select menu state with the model as an animal.
2. The 5 animal models are displayed for the respondent.

Basic Course of Events

1. The respondent points to and selects a model button from the floating menu.
2. The model selected is highlighted in the menu by a radio button.

3. The respondent selects the submit button when they are ready to begin configuration of their selected animal model.
4. The animal models other than the selected animal disappear.
5. The animal configurator menu appears.

Alternative Paths

None

Postconditions

The Model is updated to the one the user selected, and the animal configurator menu is ready to be used.

UC-6: Touch

Summary

The respondent may touch the model to rotate and move it.

Rationale

The respondent may want to change the orientation of the Model to better view the changes they have made.

Users

The respondent will be able to use this functionality.

Preconditions

1. The software should be running in the animal select menu or configurator menu states.

Basic Course of Events

1. The user grabs the Model.
2. The user rotates their hand.
3. The model follows the rotation and movement of the respondent's hand.
4. The time of the touch is recorded as an event.

Alternative Paths

None

Postconditions

The software will stay in the animal select menu or configurator menu state with the new model orientation.

UC-7: Change Animal Configurator Menu Mode

Summary
The surveyor changes the mode so that the animal configurator menu will either be a full size menu similar to the menus for the watch and the shoe, or be a condensed menu that is unique for the animal configurators.

Rationale
The original implementation of the animal configurator menu was a condensed menu. This offered an opportunity for comparison between this implementation and one that is more similar to the full size menus used for the watch and shoe configurators.

Users
The surveyor will be able to use this functionality.

Preconditions
1. The software should be running in the surveyor menu state with the model as an animal.

Basic Course of Events
1. The surveyor points to and selects the button to change the mode to the condensed menu mode.
2. The surveyor points to and selects the submit button advancing to the configurator start menu.
3. The surveyor hands the HoloLens to the respondent, and they put on the HoloLens.

Alternative Paths
None

Postconditions
The software is able to begin configuration with the respondent's touch of the configurator start button.

UC-8: End the Session

Summary
This use case describes how the respondent would end the session.

Rationale
The configurator will need to end when the user is done.

Users
All users would be able to use this functionality.

Preconditions

1. The software should be running in one of the configurator menu states.

Basic Course of Events

1. The user points to and selects the submit button.
2. The user is prompted via a popup if they are certain they are done.
3. If no is selected, the popup disappears, not affecting the Model. If yes is selected, the Model disappears, and a complete message is displayed.
4. The time of the completion is recorded as an event.

Alternative Paths

None

Postconditions

The software is no longer interactable, and will need to be restarted to begin a new configuration.

UC-9: Retrieving the Data

Summary

This use case describes how the surveyor would retrieve the data recorded by the configurator

Rationale

The purpose of the project is to record interaction data for analysis.

Users

The surveyor would be able to use this functionality.

Preconditions

1. The Mixed Reality Configurator has run to completion at least once on your HoloLens.
2. The HoloLens is paired with the device you are trying to retrieve the data with.
3. The HoloLens is connected via USB to the device, and Windows Device Portal is set up. Follow the instructions in the Windows Device Portal appendix to do this.

Basic Course of Events

1. Select System -> File Explorer in the Windows Device Portal page.
2. Navigate to User Folders/LocalAppData/MixedRealityConfigurator/LocalState
3. Download the configurator.csv file located in this folder by clicking the save icon next to it in the user interface.

Alternative Paths

There are multiple ways to access Windows Device Portal, USB is the most simple.

Postconditions

The configurator.csv file is saved on the surveyor's device.

Functional Requirements

User Interaction and Adaptability:

The system should allow the surveyor to select options for the respondent to customize including a watch, a shoe and an animal. The system should also allow the surveyor to select between the different menu modes for the animal models including the full size menu, and the condensed menu.

User choices of Customization:

When customizing the watch, the respondent may change the color of various components such as band color and body. When customizing the shoe, respondents may change the color of various components of the shoe including the laces, body color and stripe. Both the watch and the shoe should follow the same configuration process as shown in the web-based configurator. When customizing the plush toy, users can customize the type of animal. Respondents can adjust the color to different textures.

Movement tracking and record of data:

The system must keep track of the start time, and changes or customizations made, and the end time. After customization, the changes should be saved in a file that can be retrieved for further use by the surveyor.

Device Integration:

The device should be fully integrated to HoloLens allowing the users to customize the objects based on their choices.

Nonfunctional Requirements

Performance Requirements:

Reactiveness: To produce a flawless mixed reality experience, the system should have very little delay when processing user inputs and producing 3D models.

Real-time Processing: Real-time updates and customization should be reflected immediately in the 3D models.

Reliability Requirements:

System Availability: The system should be available for use 99.9% of the time with very less minimal downtime for maintenance or updates.

Data Integrity: Customizations done by the user should be stored securely without any corruption of the data.

Usability Requirements:

Capability to Learn: It should be easy for the users to learn how to use the system with minimum training.
User Interface: The mixed reality environment's interface should be simple to use and intuitive, making it easy for users to browse and customize items.

Compatibility Requirements:

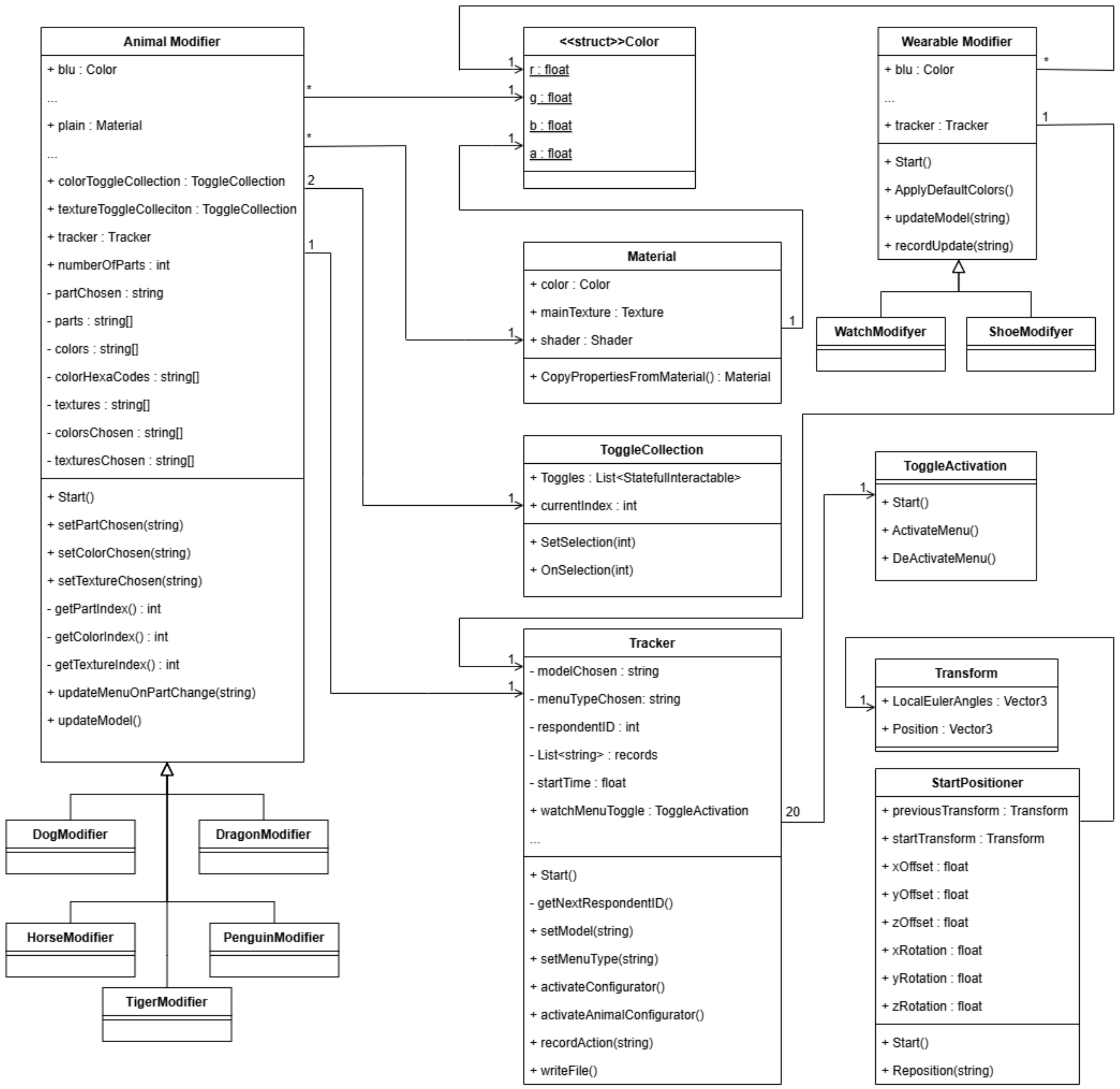
Integration: The system should have no trouble integrating with the project's file types and other current tools.

Maintainability Requirements:

Documentation: To support future development and maintenance, thorough documentation should be maintained.

System Design

Class Diagram



Modifier Classes

The modifier classes are responsible for applying any modifications or customizations to the models that the respondent applies directly by their interactions with the menus. These classes include the

WatchModifyer, ShoeModifyer, DogModifier, DragonModifier, HorseModifier, PenguinModifier, and TigerModifier.

The WatchModifyer, and ShoeModifyer classes can be generalized by the Wearable Modifier class. This is not a real class, but rather an abstraction to represent the similarities of the WatchModifyer and ShoeModifyer classes, because those 2 classes work in the exact same way. They only differ in the number of options they allow the respondent to select for their respective model and in the naming convention of their methods. The classes that implement the Wearable Modifier class include multiple Colors structs, which they apply to specific parts of the model that they are attached to. They also have an instance of the Tracker class, which they use to record the changes made to the model by applying the Colors.

The Start() method is called by Unity when the game object a script is attached to is activated. For the Wearable Modifier classes, this calls the ApplyDefaultColors() method which colors the model in its default colors, similar to how the model appeared in the 3D configurator that served as an inspiration for the Mixed Reality Configurator project.

The DogModifier, DragonModifier, HorseModifier, PenguinModifier, and TigerModifier classes can be generalized by the Animal Modifier class, which is also an abstraction representing these classes' similarities. These classes only differ in how the options the respondent selects are applied to their respective model and in the naming convention of their methods. Their behavior is significantly different enough from the Wearable Modifier classes to justify their separation from those. They have multiple Color structs, and multiple instances of the Material class which they apply to the model they are attached to. They also have 2 instances of the ToggleCollection class provided by Microsoft's Mixed Reality Toolkit (MRTK) [2], which control how the selected buttons are displayed in the menus. The instance of the Tracker class is used to record the changes made when the updateModel() method is called. The numberOfParts represents the number of regions on the model that are modifiable. The part chosen is the currently selected part of the model that will be modified. The parts, colors, and textures string arrays include strings representing all the options that the respondent can select from when modifying an animal, while the colorsChosen and texturesChosen include all the selections made by the respondent.

For the Animal Modifier classes, the Start() method populates the parts, colors, colorHexaCodes, textures, colorsChosen, and texturesChosen arrays, as well as sets the partChosen to the first element of the parts array. The getPartIndex(), getColorIndex(), and getTextureIndex() methods return the index in the corresponding array that the currently selected part, color, or texture matches. These methods are used by the updateMenuOnPartChange(string) method to set which button is toggled in the menu to match what is selected in the internal logic of the Animal Modifier classes. Finally, the updateModel() method applies the changes that the respondent made in the menu to the model, as well as communicates with the instance of the Tracker class to record those changes.

Colors, Materials, Textures, and Shaders

The Color structs [4], Material class [5], Texture class [7], and Shader class [8] are the primary ways of changing how a rendered mesh appears in the Unity game engine, outside of changing the lighting. Our scripts directly interact with the Color structs, and the Material class, which is why they are included in the above class diagram. These scripts do not directly interact with the Texture class or Shader class. Moreover, the Color struct and Material class have many more properties and methods other than the ones listed in the above class diagram, but the properties and methods listed above are the only ones relevant to our scripts. The color struct is simple enough, since the static properties it has are assigned on color instance creation, with the r, g, b, and a properties corresponding to float values between 0 and 1 representing the red, green, blue, and alpha (transparency) of the color [4]. When applying colors to a model, all that is required is setting the Material's color corresponding to the part if the model to be modified to a new color. When changing textures (as we do with the animal models), the simplest way is to use the CopyPropertiesFromMaterial() method to return a new instance of the material to apply to the part of the model that is being changed [5]. This will also apply the Textures and Shaders that material has to that part of the model [5].

Toggle Collection

The ToggleCollection class is provided by the MRTK, for managing objects that implement StatefulInteractable and need to be treated as a collection of selectable items where only specific items can be selected at once [10]. We use it to implement radio buttons on some of our menus…

Tracker and ToggleActivation Classes

StartPositioner and Transform Classes

Implementation

Development process: The project was mainly developed using the game engine software called Unity and also by using Microsoft HoloLens 2.

Our main primary goal and focus was to develop a configurator for customizing the 3D models that we have used in our project, i.e., watch, shoe, and plush animals. Here the configurator's primary goal was to customize the watch, shoe, and plush animals and also capture the user interaction data.

In our project, Unity was the primary tool, integrated with the HoloLens 2, enabling real-time 3D interactions and customization features. By allowing the respondent to interact with the 3D representation of the product. The respondent can view the product from various angles, rotate, and zoom in so that they can get a clear idea of how the customizations look in real life.

Tools and Technologies Used:

Software requirements:
Unity Hub: 3.5.1
Unity Editor Version: 2022.3.7f1
Visual Studio Community: 2022

Platform: Universal Windows Platform

<u>Hardware Requirements</u>:
Device: Microsoft Hololens 2
RAM: 16GB
CPU Architecture: ARM 64

<u>Unity and Unity Editor</u>
The game engine and editor for our project. The Unity Editor was the primary way we edited and built the Mixed Reality Configurator. It also provided a way to perform quick testing in the Unity Editor play mode.

<u>Visual Studio Community 2022</u>
This was the primary IDE we used for editing the C# code composing the scripts we wrote to control the behavior of our Mixed Reality Configurator. It was also how we deployed the application to our HoloLens.

<u>Blender</u>
This modeling software provided the tools we needed to split the meshes of our models into their components so that we could apply the different colors and textures to each individual component of the mesh. It also provided a way for us to change the format of several of the models we used so that they could be imported into the Unity Editor.

<u>Mixed Reality Toolkit and Feature Tool</u>
The Mixed Reality Toolkit provided by Microsoft and imported into our project by the Mixed Reality Feature Tool provided many of the scripts and components that made our application possible.

<u>TextMeshpro</u>
This package provided by Unity is what provides the method for rendering the text on the canvas buttons in each of our menus [11].

<u>GLTFUtility-0.7</u>
This importer created by Thor Brigsted (Siccity) on GitHub provided us the means to import some of the models we needed for our project into the Unity Editor [12].

Implementation of Features

Event Tracking:

Here it captures every interaction with timestamps and also what the respondent was changing the models that includes all types of customizations, such as the

- ➢ What colors are respondents most interested in?
- ➢ What are the textures that respondents choose most often?

➢ What are the model changes respondents make as per their requirements?

Every time the respondent selects an option, i.e., color or texture, here events will be captured. Here the selected customization is mapped to the correct component of the model (e.g., body, laces, band). All the interactions are stored in a specific CSV file, which includes the model components, selected customization, how many times they are customized, and the timestamp.

Each time the respondent performs one of these interactions that need to be recorded as an event, the recordAction() method in the Tracker class is called. The Tracker class is attached to a persistent object called the Script Object that handles all scripts that need to be running continuously for the application to work properly. This recordAction() method takes a string as input which it uses to compose a line for the csv. As long as the configurator part of the application is active, each line is stored in a private List of strings called records which is maintained by the Tracker class. Once the configurator part of the application is completed by the respondent, this List of strings is appended to the end of the configurator.csv file stored in the Local State folder of the Local App Data belonging to the configurator application on the HoloLens.

Touch Interactions:

This allows the users to rotate, pin, and reposition the models and buttons as per their requirements using the hand-tracking gestures. These are implemented by scripts and game objects provided by the Mixed Reality Toolkit (MRTK) which we included in our project via the Mixed Reality Feature Tool for Unity [2]. The most important among these are the MRTK XR Rig game object which is what allows the Unity game engine to interface with the HoloLens' various mixed reality features. The scripts attached to this game object including the Interaction Mode Manager, XR Interaction Manager, Tracked Pose Driver Lookup, and the Canvas Proxy Interactor handle the input side of mixed reality hand tracking interactions that our application relies on.

One of the primary interactions that users will perform is pushing buttons that perform some action in the world or on the holograms. The buttons used in our application are implemented as part of a canvas that provides a backdrop for multiple buttons. This makes them canvas buttons. The touch interactions having to do with canvas buttons include the Pressable Button, UGUI Input Adapter, Stateful Interactable Collider Toggle, and State Visualizer scripts. They were provided by the MRTK [2], but have been modified for use in our application. Pressable Button simply handles the events raised by the button press by providing a way to call other scripts' methods when the button is interacted with. The UGUI Input Adapter and Stateful Interactable Collider Toggle are what allow the button's box collider to trigger interactions. The State Visualizer script handles the various animations and states that the button can be in and modifies its appearance to reflect those states.

Another important interaction for our application is the manipulation of 3D models by the respondent so that they can better view their customizations. This interaction is primarily handled by the Object Manipulator script, also provided by the MRTK [2]. This script allows the respondent to grab and rotate their holograms that have associated Mesh Colliders. We modified the Object Manipulator script so that when the respondent begins manipulation of a hologram it will record the timing of their manipulation. Two additional scripts help with the manipulation process. The Constraint Manager and Min Max Scale Constraint scripts prevent the respondent from changing the size of holograms, which is a standard interaction provided by the Object Manipulator script.

Configuration of Holograms

A key part of our application is the configuration of Holograms by applying different colors and textures to the components of their 3D models. This is achieved by our menu systems using touch interactions with buttons which interface with the Modifier scripts attached to each model that we want respondents to be able to configure. To go into specific detail, this process begins when a button is pressed on one of our configurator menus. Each button makes several method calls upon being pressed.

The relevant method for the configuration of holograms is updateModel(). In the case of the wearable models (the watch and the shoe), the updateModel() method of their Modifier classes takes a string as an argument, which it uses to determine which part of the model to change and what color to apply to it by several internal switch statements. In the case of the animal models, their modifier scripts have their own data which is set by set method calls from the button press before the updateModel() method is called. They use this internal data in the form of string arrays to determine which part of the animal model to change and what texture then color to apply to it by their own multiple internal switch statements.

Activation of Game Objects

One less obvious, but important part of the user experience is the flow of menus and objects. This flow is achieved by a combination of scripts we wrote to allow us to control what objects appear when and how. The ToggleActivation script utilizes a feature of Unity called tagging to control which are active at any given time. If an object is intended to be active the ToggleActivation's ActivateMenu() method will set its tag to "GameController," and call its setActive(boolean) method (which is a native method for all objects in Unity) with the argument as true. If an object is not intended to be active, the ToggleActivation script's DeActivateMenu() method will set its tag to "Untagged," and call its setActive(boolean) method with the argument as false. One question that might arise from this description is "how does the ToggleActivation script know when objects are supposed to be active or not?" There are two ways. The first is the more simple and intuitive way: when a specific button is pressed, it calls the ActivateMenu() method for each object that is intended to appear, and the DeActivateMenu() method for each object that is intended to disappear. This method works for the majority of ToggleActivation script uses, but doesn't work when the result of that button needs to be dynamic. That is where the second method comes in.

As mentioned under the event tracking subsection above, the Tracker script's primary purpose is to track the various events as data that needs to be recorded when the respondent completes their configuration. However, some of that data can be used to determine what state the application is in, and consequently make decisions about what objects need to be active. That is why the Tracker has access to 20 instances of the ToggleActivation script. Each is attached to an object, and the internal logic of the Tracker script decides which objects need to be active. The switch cases in the activateConfigurator() method and activateAnimalConfigurator() methods provide this internal logic.

Positioning of Menus and Holograms

An additional inconspicuous part of the user experience is the orientation of menus and objects. For the Mixed Reality Configurator, we rely on the use of Solvers for the smooth movement and orientation of menus. Solvers are scripts provided by the MRTK that easily handle specific movement scenarios commonly required by Mixed Reality developers [2]. In particular, we make extensive use of the Radial View solver to allow menus to face the user and stay at a particular distance from them as they are interacting with our application. The Solver Handler script also helps orientate menus by providing additional offsets and rotation should the Solver alone not prove sufficient [2]. If a menu appears to be improperly adjusted in terms of what distance and direction it is facing relative to the user, modifying the menu's Radial View solver or Solver Handler should be the only adjustments required to remedy that issue.

The holograms that the respondent configures also need to be properly placed and face them when they begin their configuration. This is where the StartPositioner script comes in. This script takes the position of an object and applies it to a new object when that object is activated, with an additional positional and rotational offset. This allows the menus and models to appear in the same place relative to the user each time that the application is run, because without the StartPositioner, they would appear relative to the HoloLens's position at the start of the application.

Menu Layout System

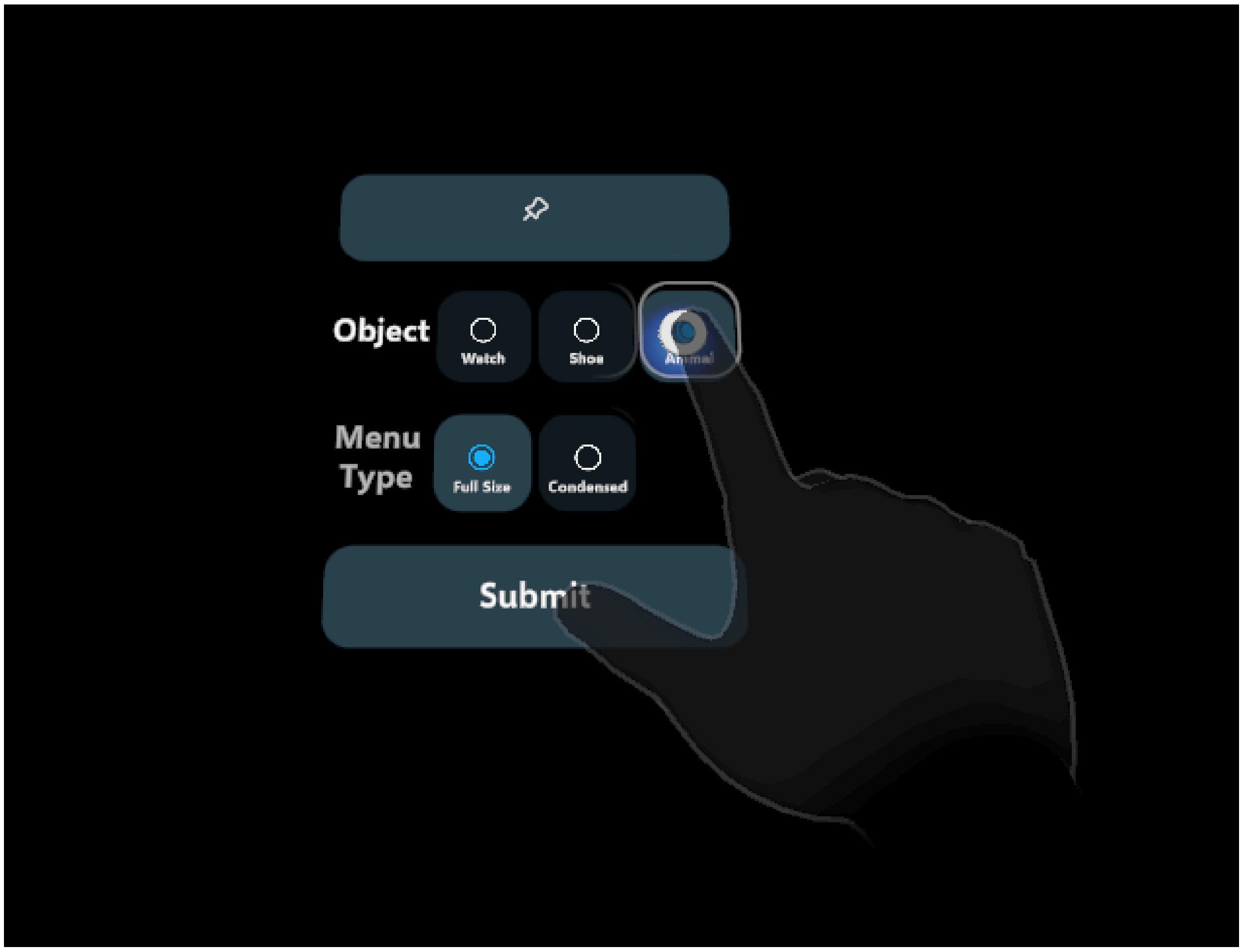
Every menu in the Mixed Reality Configurator that is more than a single button consists of the same components. Note that this framework for the menu layout system was provided by the MRTK [2]. Each menu starts with a canvas which is essentially just an invisible holder for UI elements. This canvas is divided into Vertical Layout Groups, which define the size and shape of each row of buttons. Each Vertical Layout Group is in turn divided into a Horizontal Layout Group, which defines the size and shape of each column of buttons on that row. Finally, each Horizontal Layout Group is divided into Grid Layout Group which defines the size and shape of each individual button in that group. A Toggle Collection script may also be attached to this Grid Layout Group if the buttons in it need to work as if they were Radio Buttons. Each button is placed in the Grid Layout Group and has its own Backplate, Backglow, and Frontplate components which govern how it is rendered and animated.

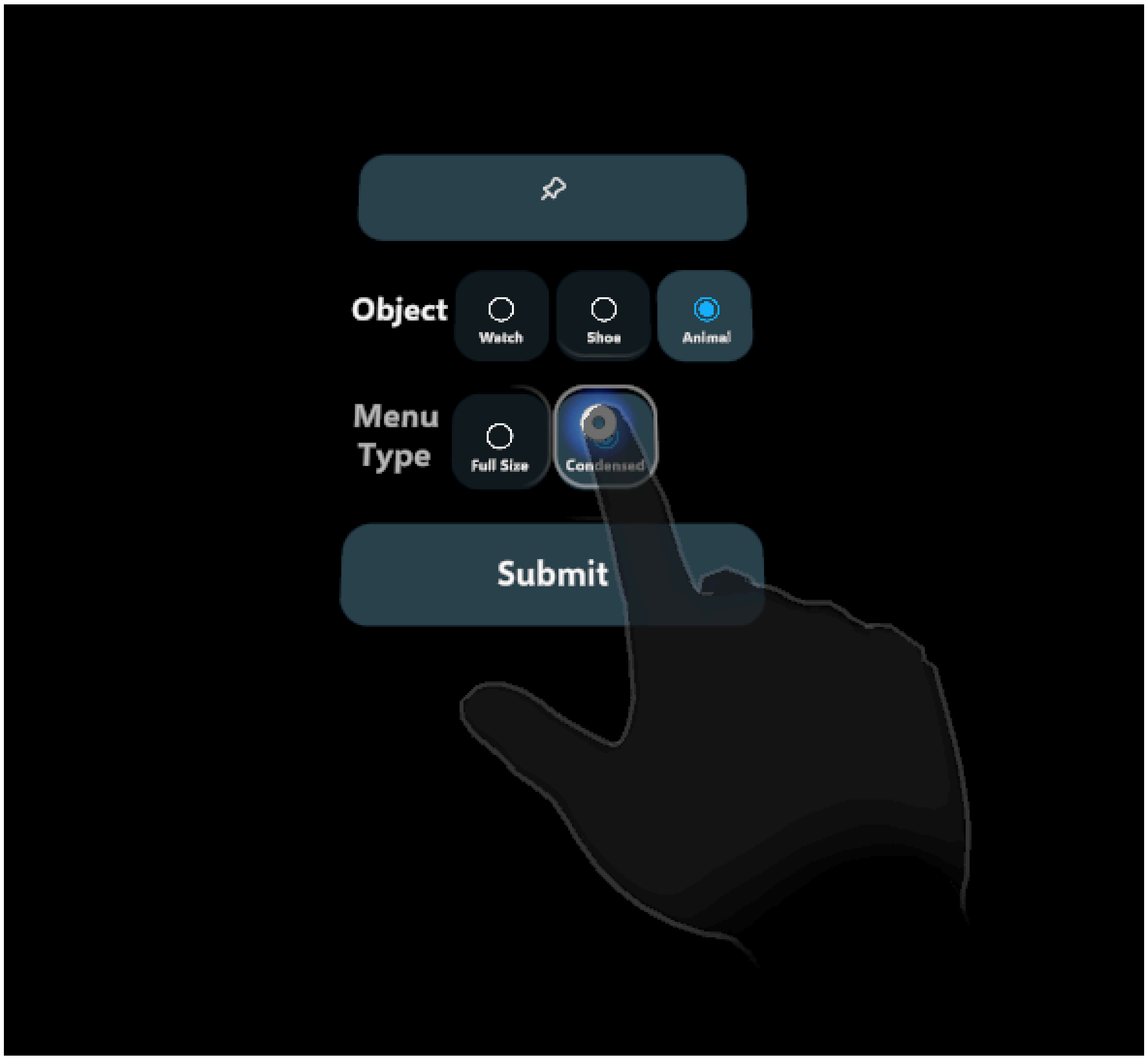
Testing

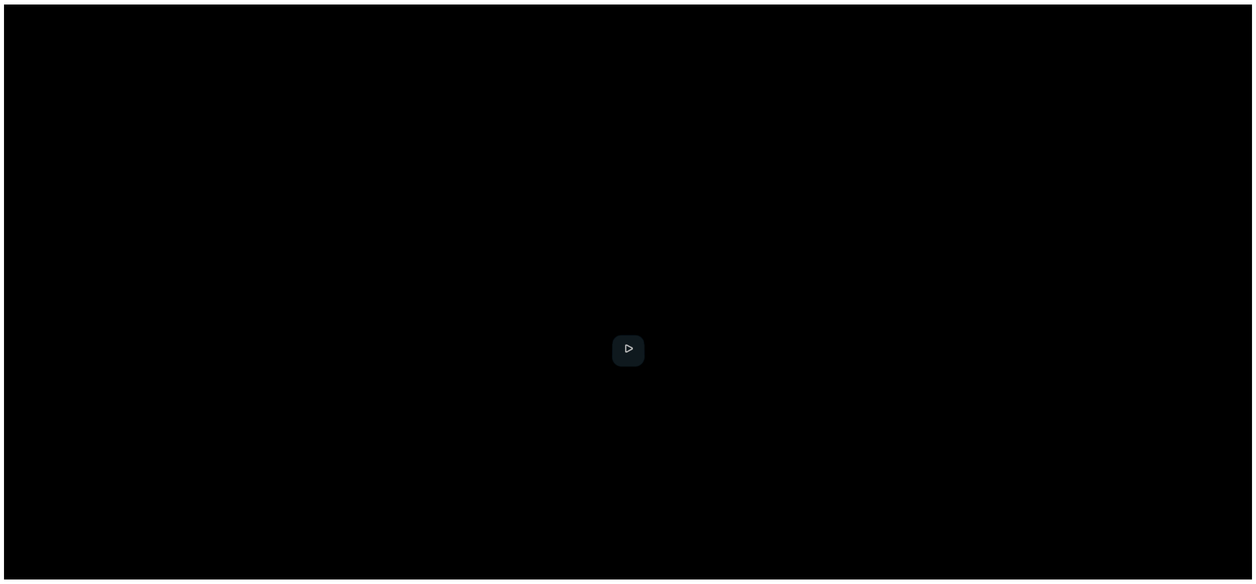
In Editor Testing

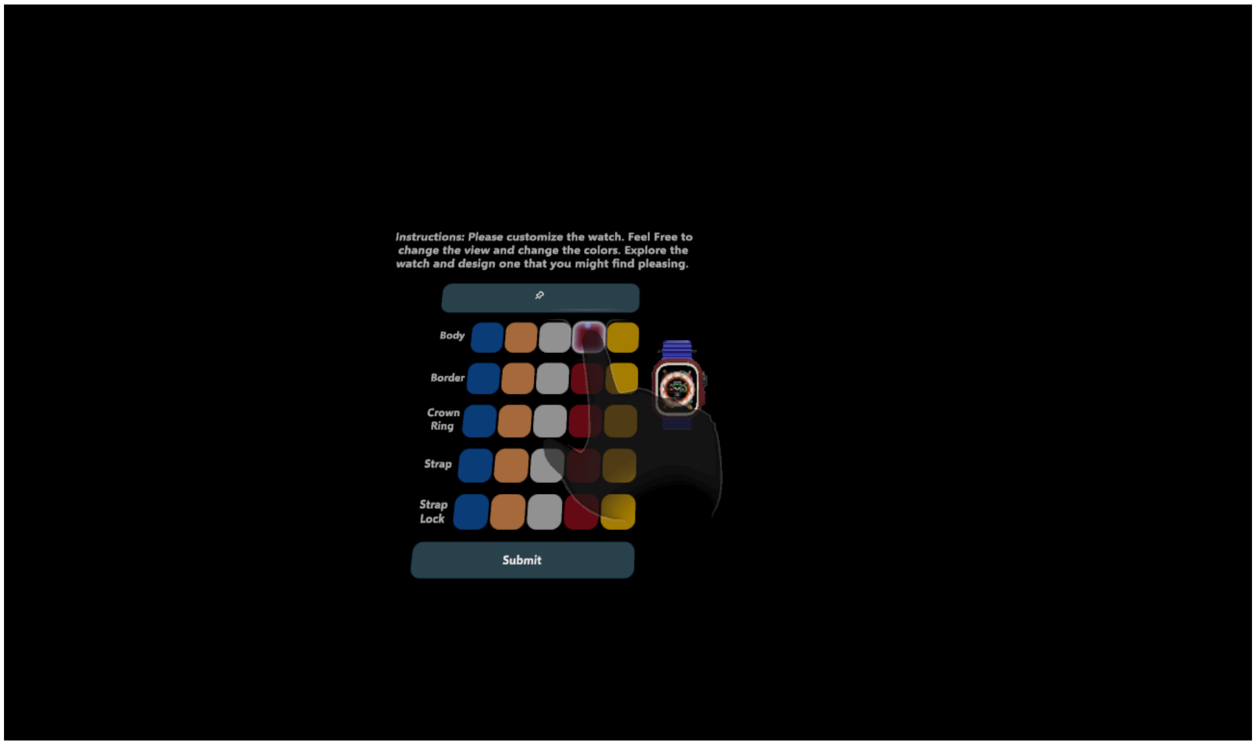
On HoloLens Testing

Results:

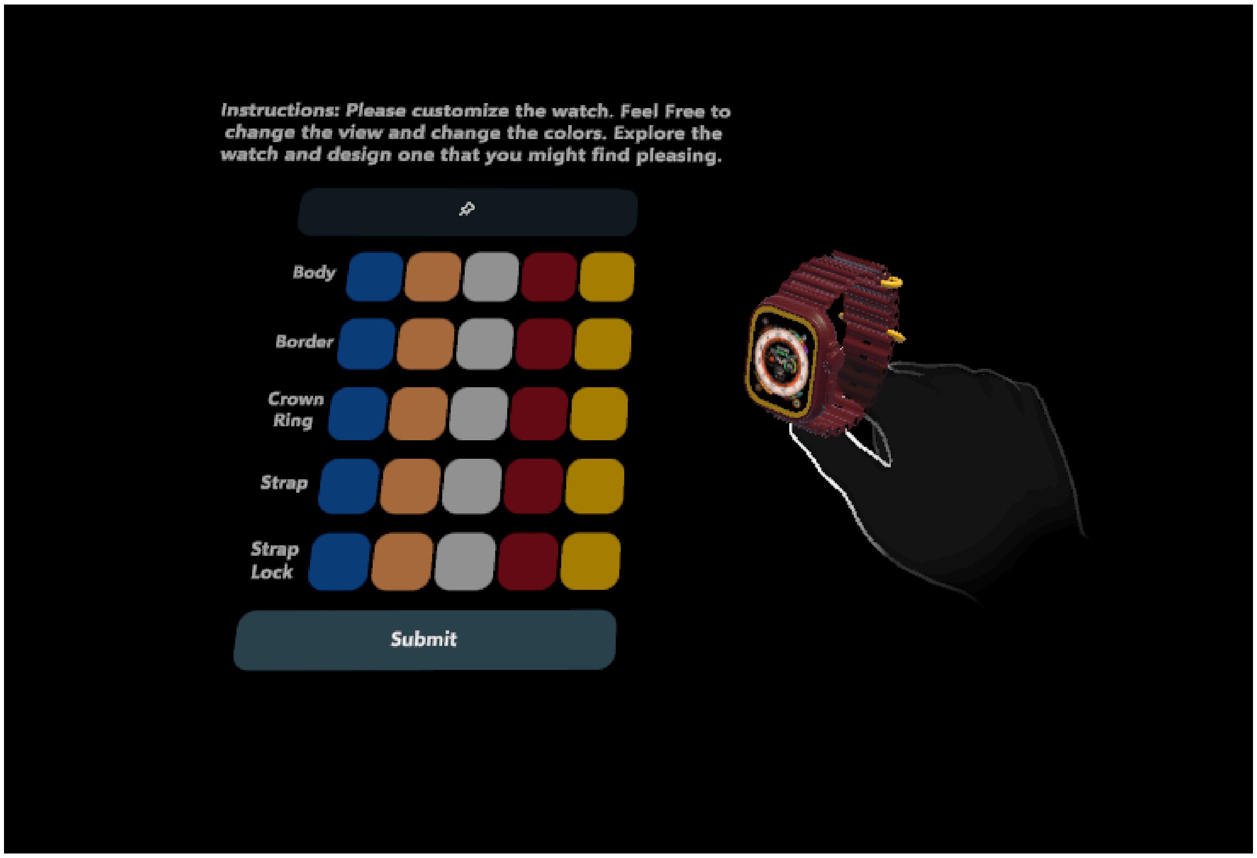

Object
Watch
Shoe
Animal
Menu Type
Full Size
Condensed
Submit


Object
Watch
Shoe
Animal
Menu Type
Full Size
Condensed
Submit

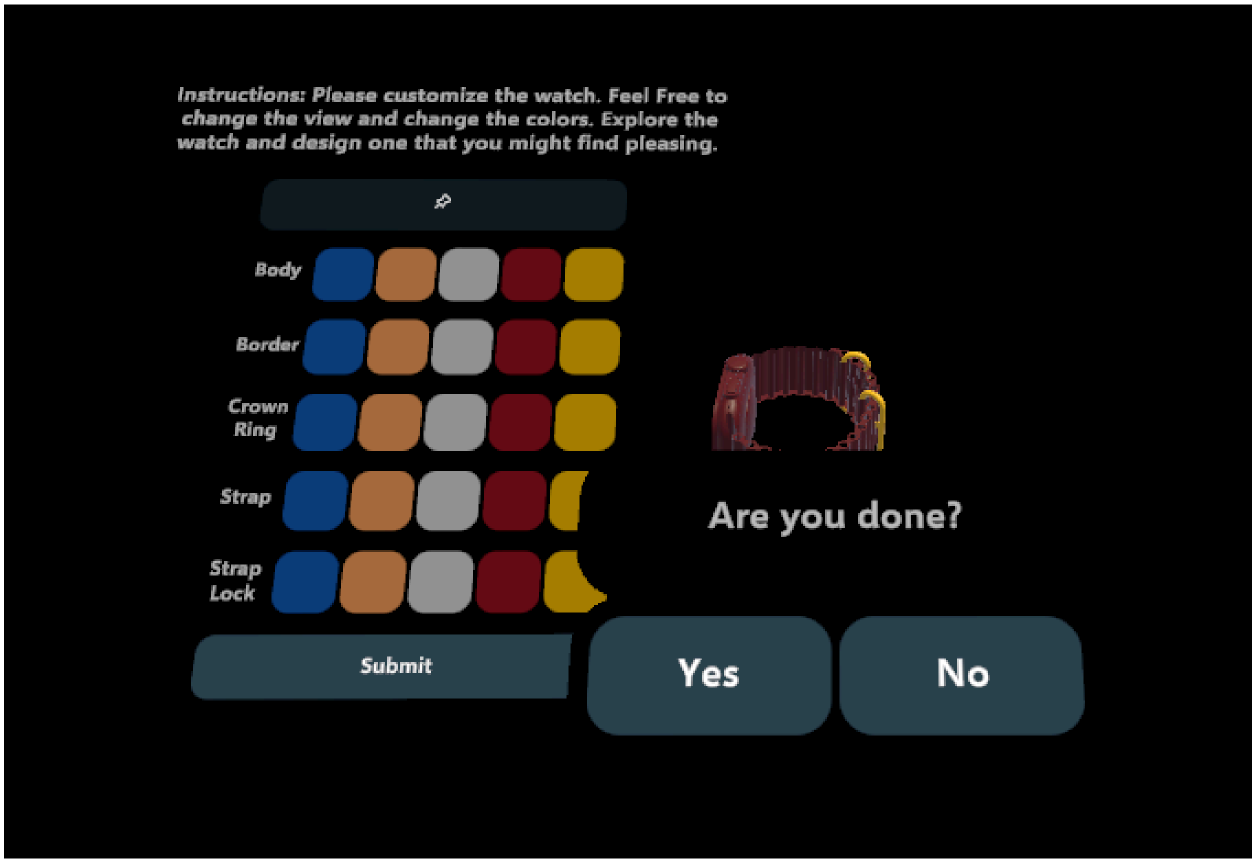


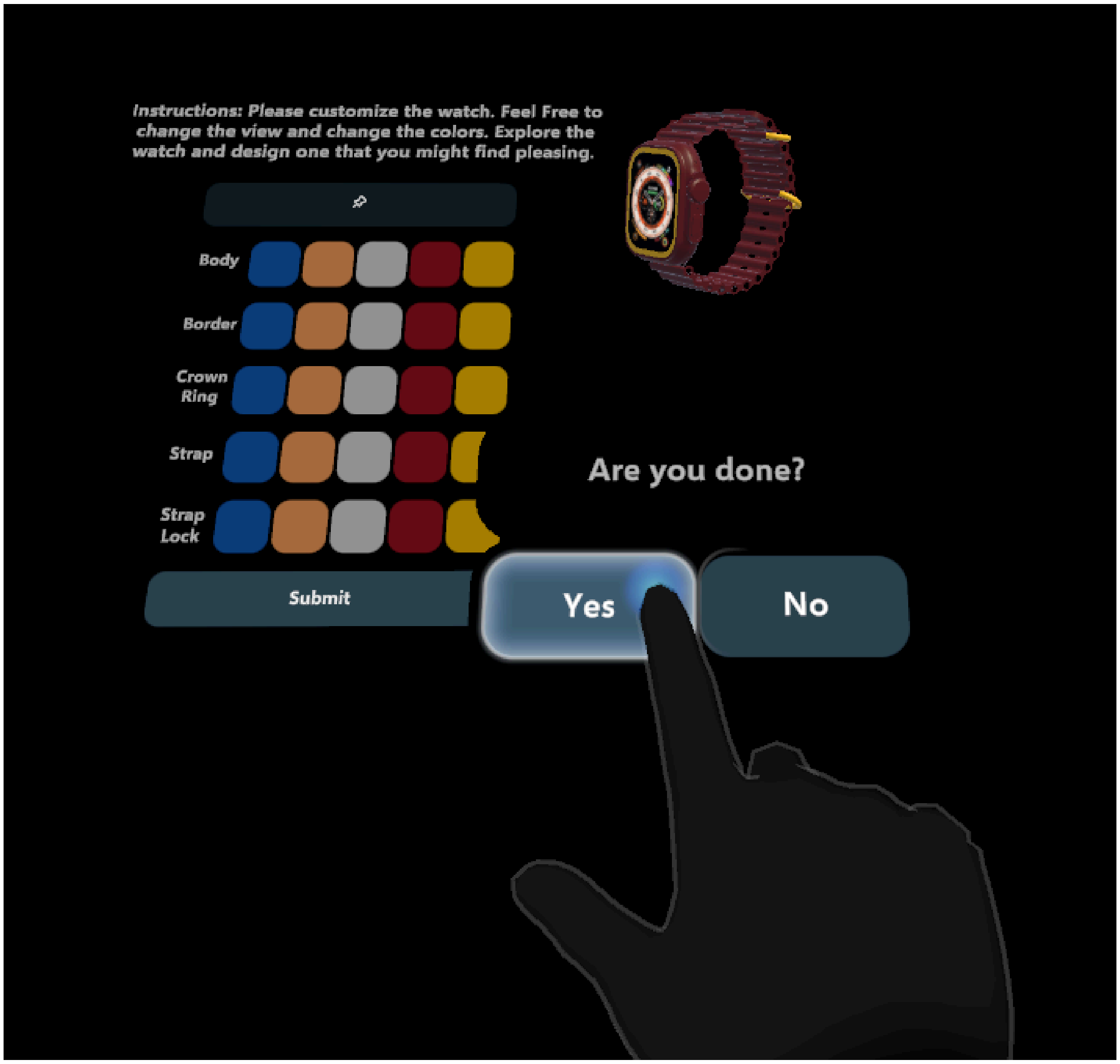


Instructions: Please customize the watch. Feel Free to change the view and change the colors. Explore the watch and design one that you might find pleasing.

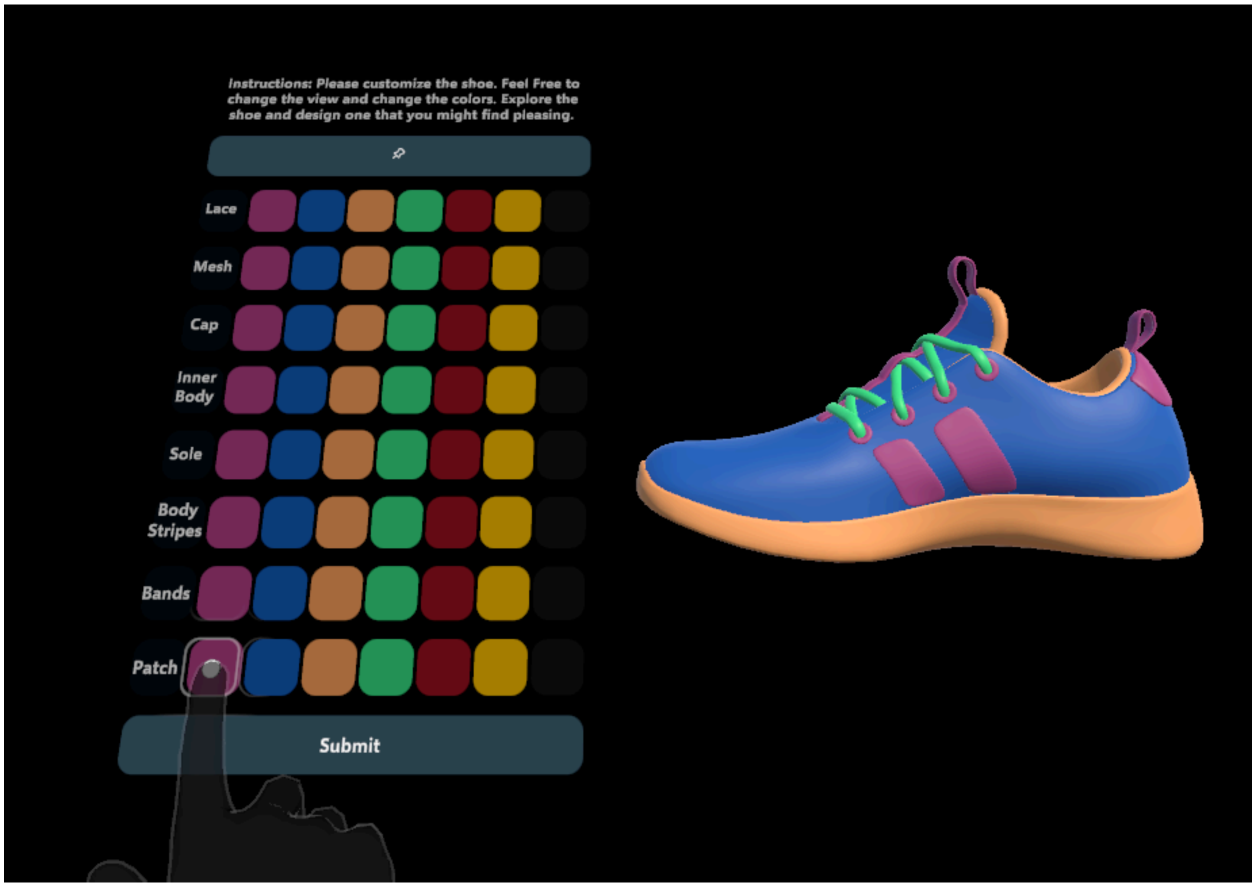


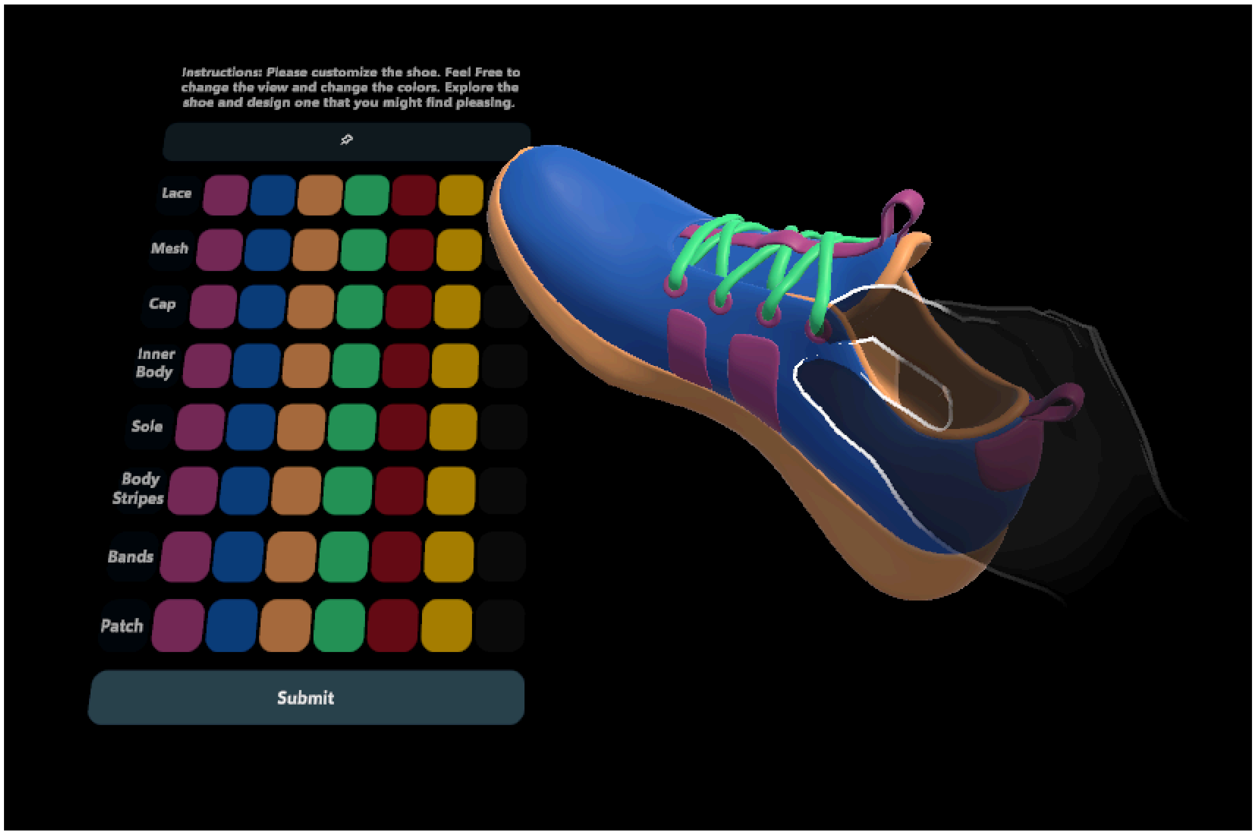
Instructions: Please customize the watch. Feel Free to change the view and change the colors. Explore the watch and design one that you might find pleasing.

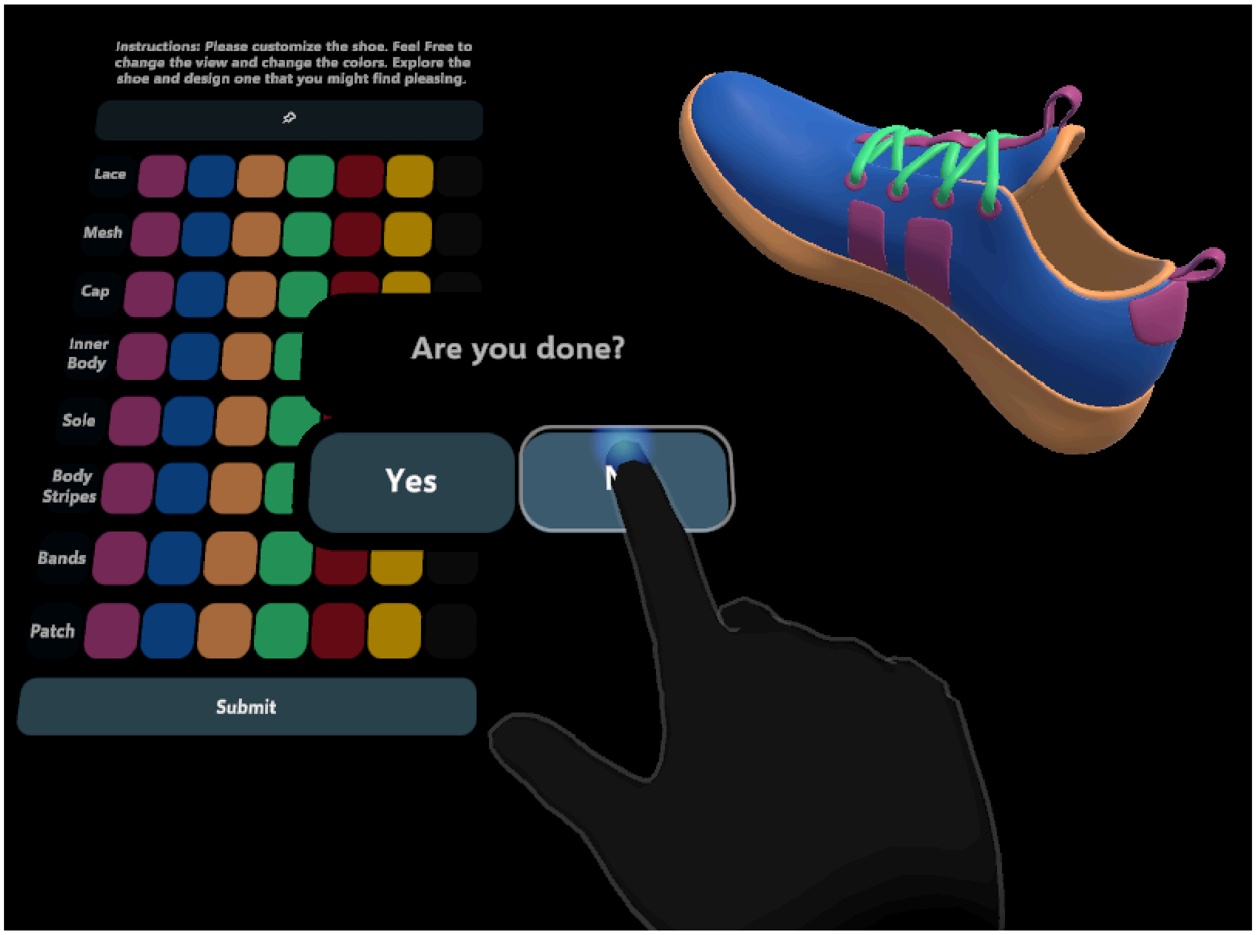

Instructions: Please customize the watch. Feel Free to change the view and change the colors. Explore the watch and design one that you might find pleasing.


Instructions: Please customize the watch. Feel Free to change the view and change the colors. Explore the watch and design one that you might find pleasing.

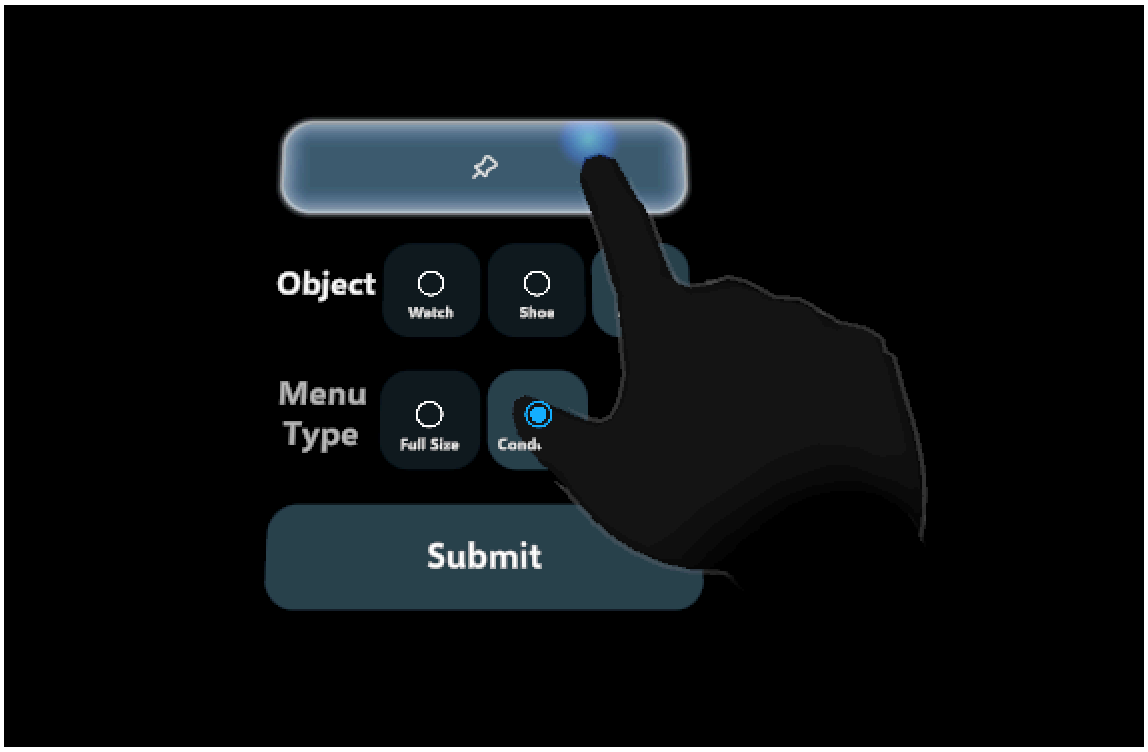
Complete!


Instructions: Please customize the shoe. Feel Free to change the view and change the colors. Explore the shoe and design one that you might find pleasing.

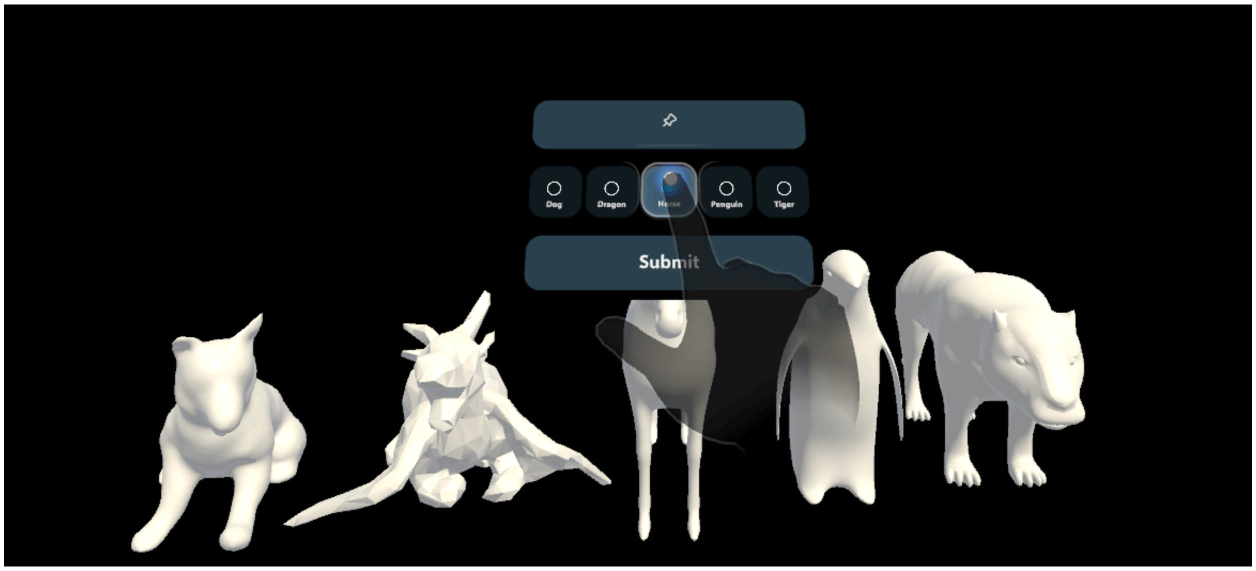

Instructions: Please customize the shoe. Feel Free to change the view and change the colors. Explore the shoe and design one that you might find pleasing.



Instructions: Please customize the shoe. Feel Free to change the view and change the colors. Explore the shoe and design one that you might find pleasing.



Menu Type: Full Size [✓ Cond]


Dog
Dragon
Horse
Penguin
Tiger
Submit

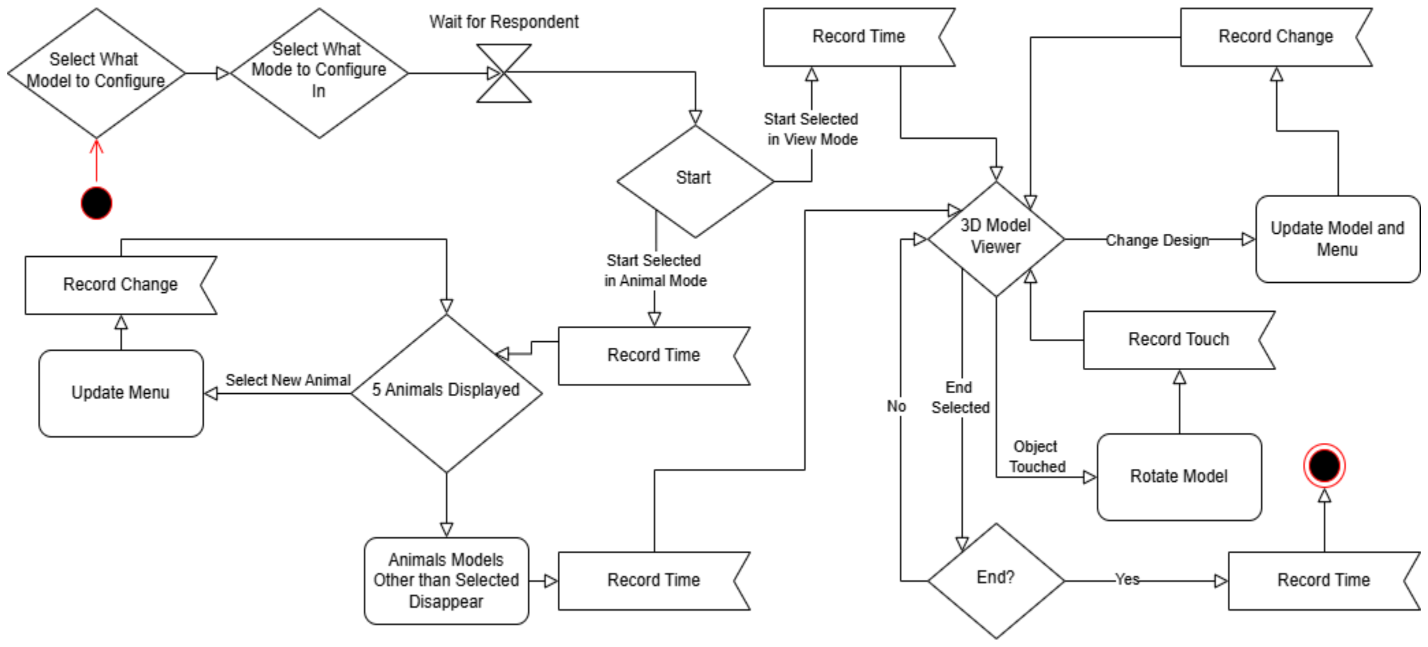
```
ID,Product,Feature,ColorCode,Texture,MenuType,TimeSinceStart(seconds)
1,HOLO-NA,Respondent set animal to Dragon,NA,NA,FullSize,2.189745
1,HOLO-NA,Respondent set animal to Dog,NA,NA,FullSize,4.958048
1,HOLO-NA,Respondent set animal to Penguin,NA,NA,FullSize,9.982691
1,HOLO-NA,Respondent started animal configuration,NA,NA,FullSize,11.15995
1,HOLO-PENGUIN,Front,#1e62c0,Plain,FullSize,14.8748
1,HOLO-PENGUIN,Back,#1e62c0,Plain,FullSize,15.26627
1,HOLO-PENGUIN,Flipper,#1e62c0,Plain,FullSize,15.68104
1,HOLO-PENGUIN,Feet,#1e62c0,Plain,FullSize,16.47906
1,HOLO-PENGUIN,Beak,#1e62c0,Plain,FullSize,19.7618
1,HOLO-PENGUIN,Eye Ring,#1e62c0,Plain,FullSize,20.1581
1,HOLO-PENGUIN,Eyes,#1e62c0,Plain,FullSize,20.51537
1,HOLO-PENGUIN,Eyes,#ffa764,Plain,FullSize,20.85902
1,HOLO-PENGUIN,Eye Ring,#ffa764,Plain,FullSize,21.34056
1,HOLO-PENGUIN,Beak,#ffa764,Plain,FullSize,21.7197
1,HOLO-PENGUIN,Feet,#ffa764,Plain,FullSize,22.21611
1,HOLO-PENGUIN,Flipper,#ffa764,Plain,FullSize,22.58976
1,HOLO-PENGUIN,Back,#ffa764,Plain,FullSize,23.06198
1,HOLO-PENGUIN,Front,#ffa764,Plain,FullSize,23.45565
1,HOLO-PENGUIN,Front,#a11f2a,Plain,FullSize,23.77242
1,HOLO-PENGUIN,Back,#a11f2a,Plain,FullSize,25.16506
1,HOLO-PENGUIN,Flipper,#a11f2a,Plain,FullSize,25.78758
1,HOLO-PENGUIN,Feet,#a11f2a,Plain,FullSize,26.27631
1,HOLO-PENGUIN,Beak,#a11f2a,Plain,FullSize,26.60981
1,HOLO-PENGUIN,Eye Ring,#a11f2a,Plain,FullSize,26.99607
1,HOLO-PENGUIN,Eyes,#a11f2a,Plain,FullSize,27.37075
1,HOLO-PENGUIN,Eyes,#ffbf00,Plain,FullSize,28.14631
1,HOLO-PENGUIN,Eye Ring,#ffbf00,Plain,FullSize,28.5457
1,HOLO-PENGUIN,Beak,#ffbf00,Plain,FullSize,28.96558
1,HOLO-PENGUIN,Feet,#ffbf00,Plain,FullSize,29.35885
1,HOLO-PENGUIN,Flipper,#ffbf00,Plain,FullSize,29.85914
1,HOLO-PENGUIN,Back,#ffbf00,Plain,FullSize,30.23997
1,HOLO-PENGUIN,Front,#ffbf00,Plain,FullSize,30.6465
1,HOLO-PENGUIN,Front,#3de68b,Plain,FullSize,31.08586
1,HOLO-PENGUIN,Back,#3de68b,Plain,FullSize,31.56306
1,HOLO-PENGUIN,Flipper,#3de68b,Plain,FullSize,32.09337
1,HOLO-PENGUIN,Feet,#3de68b,Plain,FullSize,32.60149
1,HOLO-PENGUIN,Beak,#3de68b,Plain,FullSize,33.12956
1,HOLO-PENGUIN,Eye Ring,#3de68b,Plain,FullSize,33.57201
1,HOLO-PENGUIN,Eyes,#3de68b,Plain,FullSize,33.9715
1,HOLO-PENGUIN,Front,#b3478c,Plain,FullSize,34.8375
1,HOLO-PENGUIN,Back,#b3478c,Plain,FullSize,35.19896
1,HOLO-PENGUIN,Flipper,#b3478c,Plain,FullSize,35.65356
1,HOLO-PENGUIN,Feet,#b3478c,Plain,FullSize,36.2749
1,HOLO-PENGUIN,Beak,#b3478c,Plain,FullSize,36.62674
1,HOLO-PENGUIN,Eye Ring,#b3478c,Plain,FullSize,37.10872
1,HOLO-PENGUIN,Eyes,#b3478c,Plain,FullSize,37.54039
1,HOLO-PENGUIN,Eyes,#171716,Plain,FullSize,38.05008
1,HOLO-PENGUIN,Eye Ring,#171716,Plain,FullSize,38.4388
1,HOLO-PENGUIN,Beak,#171716,Plain,FullSize,38.96204
1,HOLO-PENGUIN,Feet,#171716,Plain,FullSize,39.56528
```

```
13,HOLO-HORSE,Hooves,#171716,Checker,Condensed,219.4719
13,HOLO-HORSE,Stockings,#3de68b,Checker,Condensed,222.352
13,HOLO-HORSE,Respondent touched object,NA,NA,Condensed,230.1959
13,HOLO-HORSE,Respondent touched object,NA,NA,Condensed,248.7668
13,HOLO-HORSE,Respondent touched object,NA,NA,Condensed,252.2955
13,HOLO-HORSE,Respondent touched object,NA,NA,Condensed,307.5597
13,HOLO-HORSE,Respondent touched object,NA,NA,Condensed,312.2766
13,HOLO-HORSE,Respondent touched object,NA,NA,Condensed,332.9254
13,HOLO-HORSE,Submit button initiated,NA,NA,Condensed,341.4024
13,HOLO-HORSE,Configuration complete,NA,NA,Condensed,342.2208
14,HOLO-NA,Respondent set animal to Tiger,NA,NA,FullSize,1.011421
14,HOLO-NA,Respondent started animal configuration,NA,NA,FullSize,1.647124
14,HOLO-TIGER,Body,#1e62c0,Plain,FullSize,28.76225
14,HOLO-TIGER,Belly,#1e62c0,Plain,FullSize,29.29017
14,HOLO-TIGER,Body,#ffa764,Plain,FullSize,31.5603
14,HOLO-TIGER,Body,#e6ecf2,Plain,FullSize,32.70849
14,HOLO-TIGER,Body,#171716,Plain,FullSize,33.63322
14,HOLO-TIGER,Body,#3de68b,Plain,FullSize,36.33685
14,HOLO-TIGER,Body,#ffa764,Plain,FullSize,39.43628
14,HOLO-TIGER,Stripes,#1e62c0,Plain,FullSize,41.95274
```

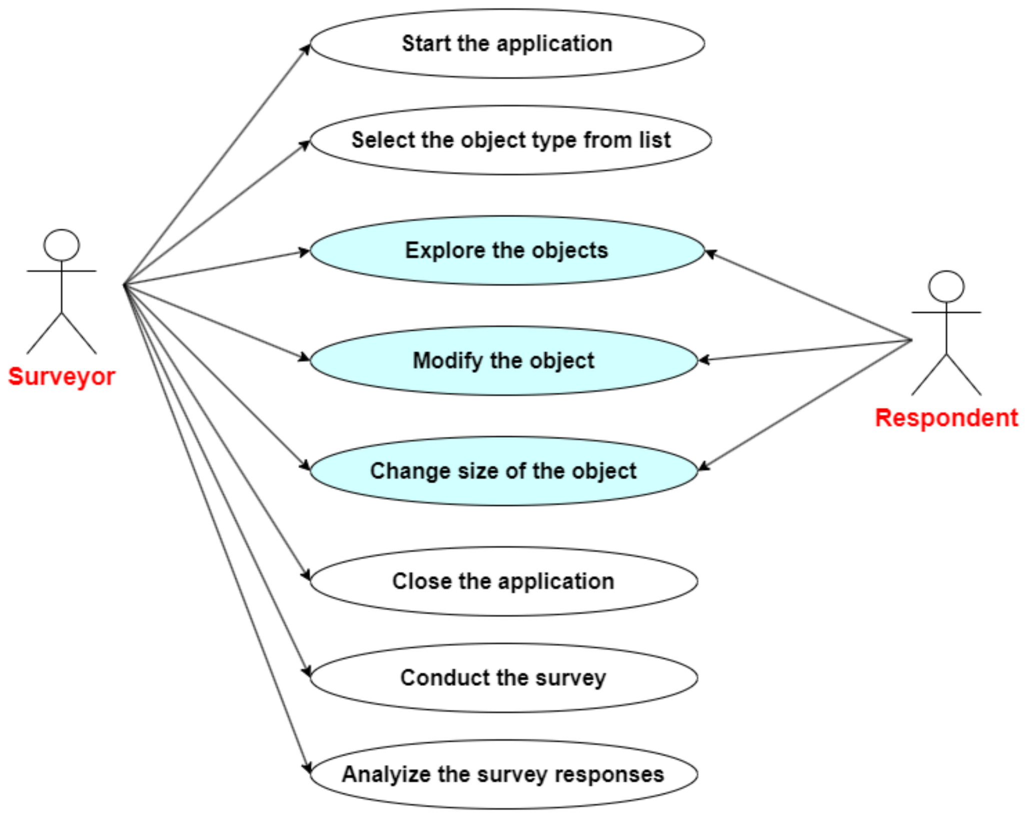
Appendices

Design Diagrams

Activity Diagram

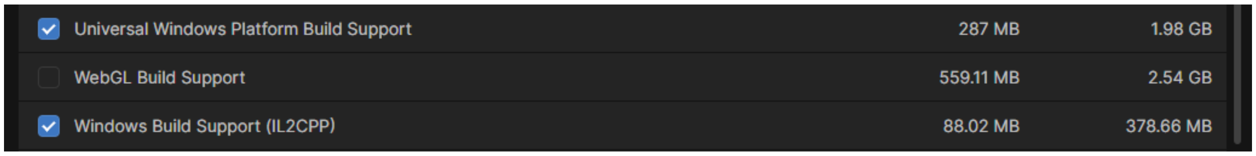


Use Case Diagram



Installing the Required Software to Develop the Mixed Reality Configurator

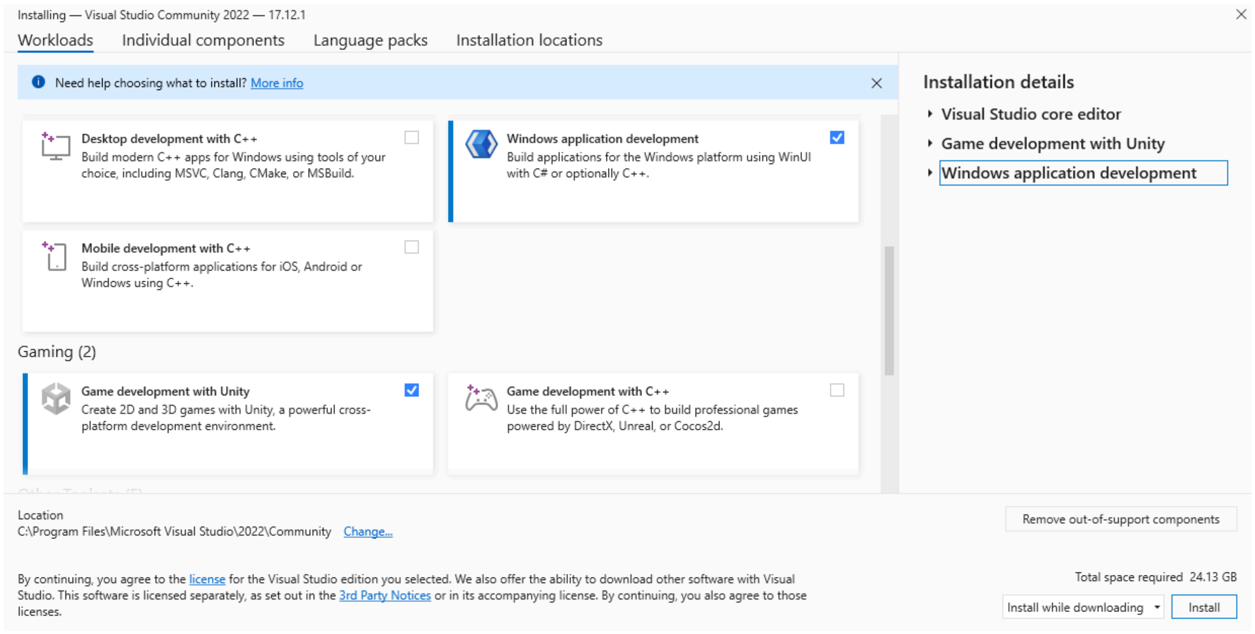
1. Download and Install Unity Hub. The version shouldn't matter, but the most recent version as of writing is Unity Hub 3.10.0.
2. Open Unity Hub. Skip installing an editor when first opening, then click the blue "Install Editor" button, select the "Archive" tab, and click the blue highlighted download archive link.
3. In your web browser scroll down and click to install Unity Editor 2022.3.7f1.
4. A window should appear showing the options for your installation of Unity Editor 2022.3.7f1. Ensure that under "Dev Tools" Microsoft Visual Studio 2022 us checked. Under platforms ensure that Universal Windows Platform Build Support and Windows Build Support (IL2CPP) are both checked.



✓	Universal Windows Platform Build Support	287 MB	1.98 GB
	WebGL Build Support	559.11 MB	2.54 GB
✓	Windows Build Support (IL2CPP)	88.02 MB	378.66 MB

5. Click continue and accept the terms and conditions. Click Install.
6. After some delay, Visual Studio Community 2022 will prompt you for some setup. You will need to select "Windows Application Development" and "Game Development with

Unity."


Installing — Visual Studio Community 2022 — 17.12.1

Workloads | Individual components | Language packs | Installation locations

Need help choosing what to install? More info

[ ] Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

[✓] Windows application development
Build applications for the Windows platform using WinUI with C# or optionally C++.

[ ] Mobile development with C++
Build cross-platform applications for iOS, Android or Windows using C++.

Gaming (2)

[✓] Game development with Unity
Create 2D and 3D games with Unity, a powerful cross-platform development environment.

[ ] Game development with C++
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

Installation details
- Visual Studio core editor
- Game development with Unity
- Windows application development

Location
C:\Program Files\Microsoft Visual Studio\2022\Community Change...

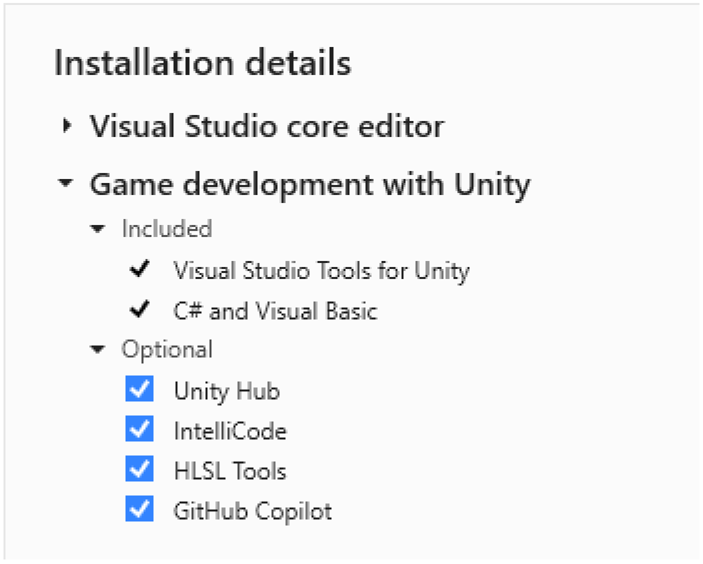
Remove out-of-support components

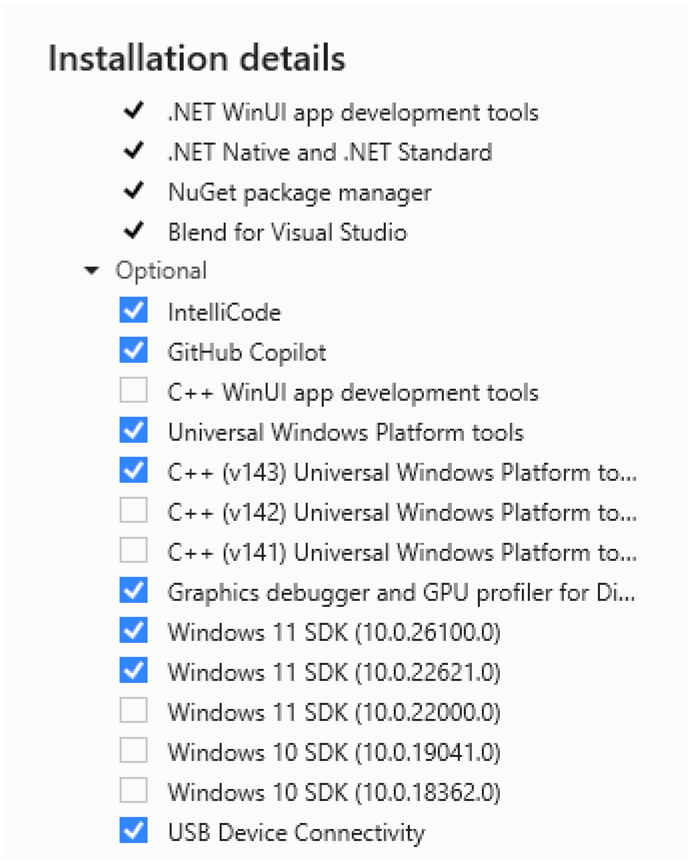
By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 24.13 GB

Install while downloading | Install

7. On the right side optional installations will appear. Ensure all the optional installations are selected for "Game Development With Unity." Be sure that IntelliCode, GitHub Copilot, Universal Windows Platform tools, C++ (v143) Universal Windows Platform tools, Graphics debugger and GPU profiler for DirectX 11, Windows 11 SDK (10.0.26100.0), Windows 11 SDK (10.0.22621.0), and USB Device Connectivity are checked for "Windows Application Development."


Installation details
- Visual Studio core editor
- Game development with Unity
  - Included
    - ✓ Visual Studio Tools for Unity
    - ✓ C# and Visual Basic
  - Optional
    - [✓] Unity Hub
    - [✓] IntelliCode
    - [✓] HLSL Tools
    - [✓] GitHub Copilot



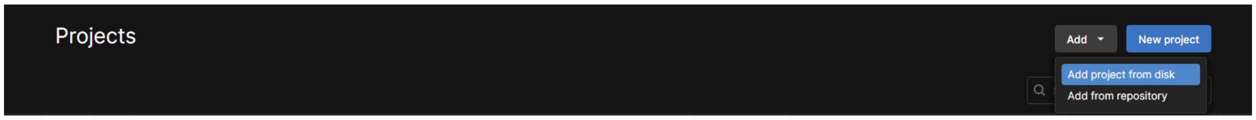
Installation details

- ✓ .NET WinUI app development tools
- ✓ .NET Native and .NET Standard
- ✓ NuGet package manager
- ✓ Blend for Visual Studio
- Optional
  - [✓] IntelliCode
  - [✓] GitHub Copilot
  - [ ] C++ WinUI app development tools
  - [✓] Universal Windows Platform tools
  - [✓] C++ (v143) Universal Windows Platform to...
  - [ ] C++ (v142) Universal Windows Platform to...
  - [ ] C++ (v141) Universal Windows Platform to...
  - [✓] Graphics debugger and GPU profiler for Di...
  - [✓] Windows 11 SDK (10.0.26100.0)
  - [✓] Windows 11 SDK (10.0.22621.0)
  - [ ] Windows 11 SDK (10.0.22000.0)
  - [ ] Windows 10 SDK (10.0.19041.0)
  - [ ] Windows 10 SDK (10.0.18362.0)
  - [✓] USB Device Connectivity

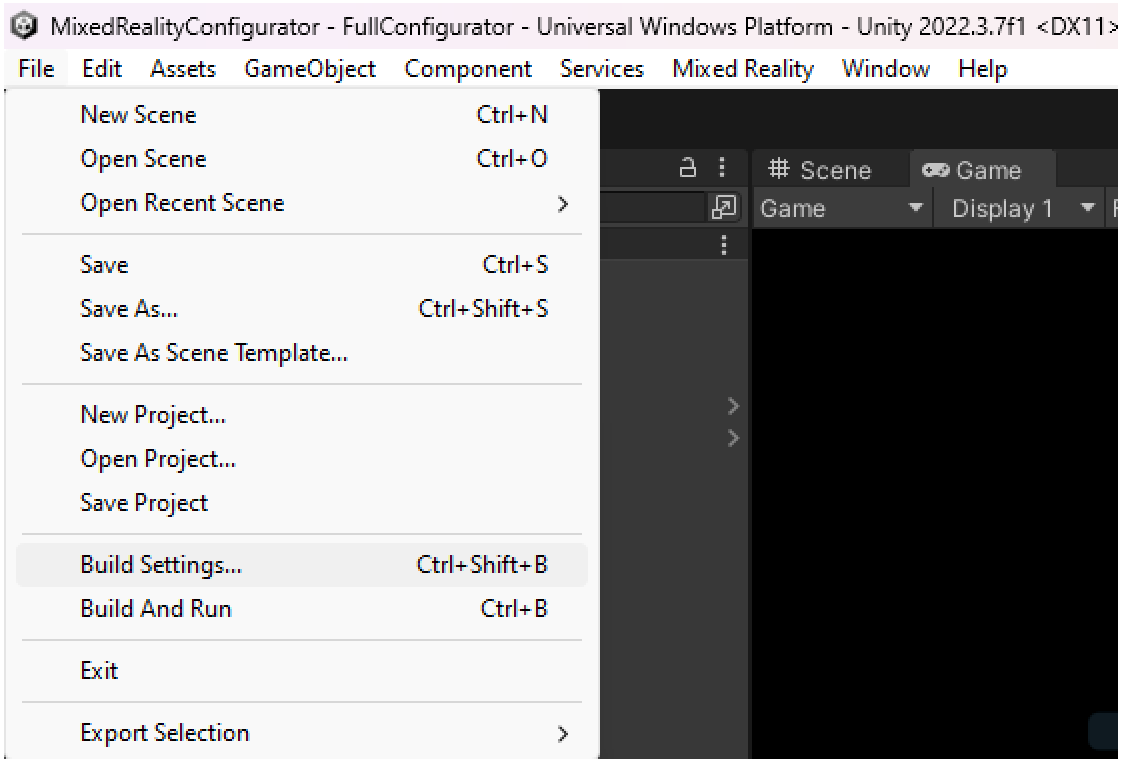
8. Click “Install while Downloading,” and close Visual Studio Community 2022 once installed.
9. That should be everything you need. You can now add Mixed Reality Configurator to your projects in the project tab of Unity Hub, and open it by clicking on that project.

<u>**Deploying the Mixed Reality Configurator to HoloLens**</u>

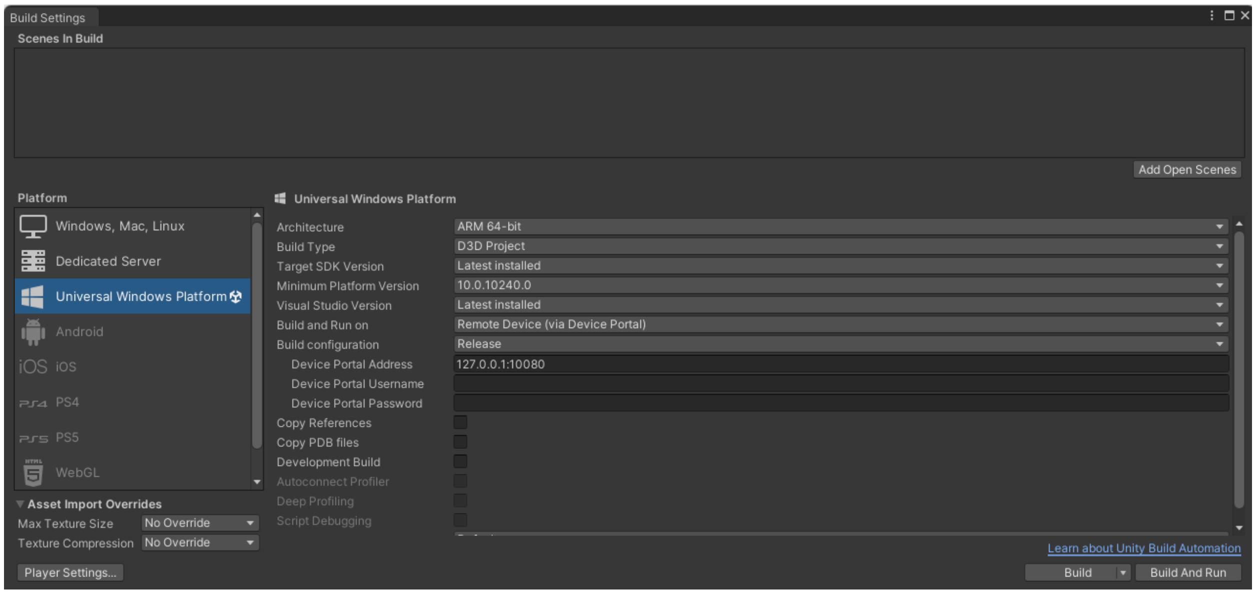
1. Open Unity Hub and make sure the Mixed Reality Configurator is available in your projects. If not you can add it from disk by clicking the Add button and selecting Add project from disk.



2. Open the Mixed Reality Configurator by clicking on it under the Projects tab of Unity Hub.



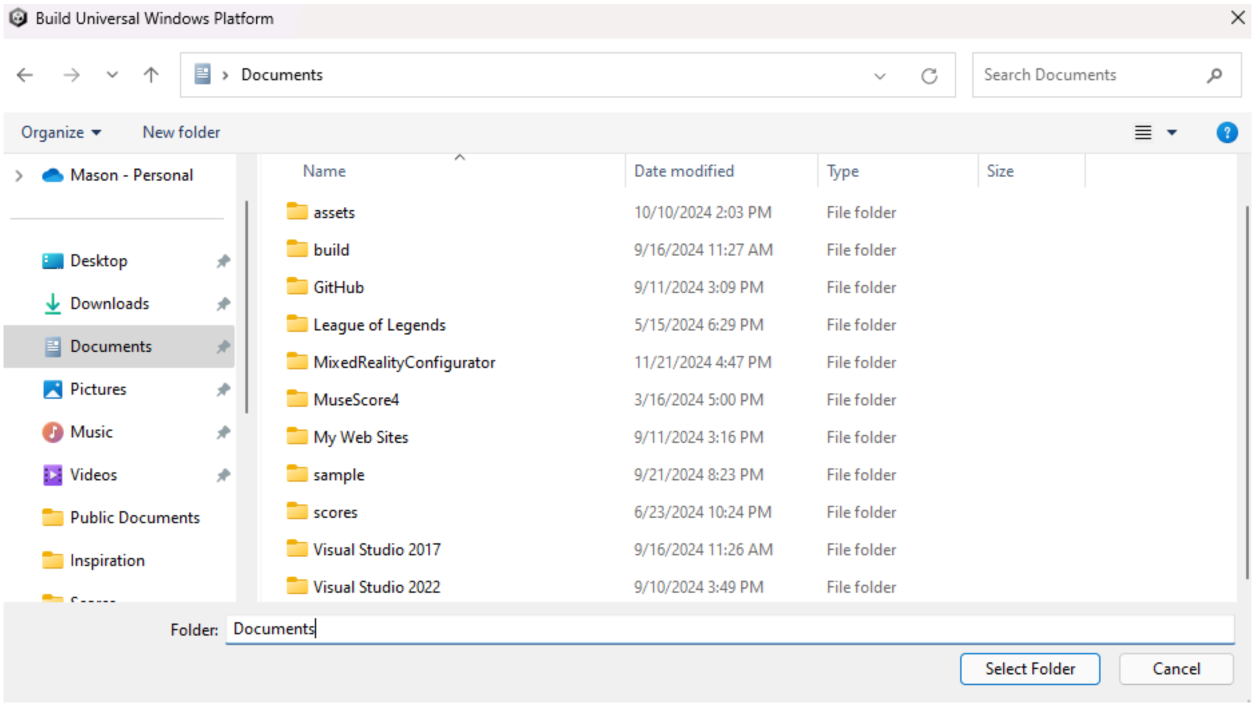
3. After opening click on File -> Build Settings. Ensure the build settings are as shown in the screenshot below. If you have deployed before and made changes, you will have to delete the previous Il2CppOutputProject folder in the directory you're building to. Then click on Build.



Architecture: ARM 64-bit
Build Type: D3D Project
Target SDK Version: Latest installed
Minimum Platform Version: 10.0.10240.0
Visual Studio Version: Latest installed
Build and Run on: Remote Device (via Device Portal)
Build configuration: Release
Device Portal Address: 127.0.0.1:10080
Device Portal Username:
Device Portal Password:
Copy References
Copy PDB files
Development Build
Autoconnect Profiler
Deep Profiling
Script Debugging

Asset Import Overrides
Max Texture Size: No Override
Texture Compression: No Override

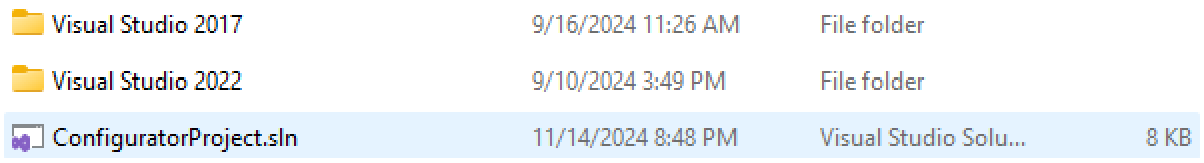
4. Select the directory you want to build the project to. After a short delay, the project should be built to your selected directory.



Name	Date modified	Type	Size
assets	10/10/2024 2:03 PM	File folder	
build	9/16/2024 11:27 AM	File folder	
GitHub	9/11/2024 3:09 PM	File folder	
League of Legends	5/15/2024 6:29 PM	File folder	
MixedRealityConfigurator	11/21/2024 4:47 PM	File folder	
MuseScore4	3/16/2024 5:00 PM	File folder	
My Web Sites	9/11/2024 3:16 PM	File folder	
sample	9/21/2024 8:23 PM	File folder	
scores	6/23/2024 10:24 PM	File folder	
Visual Studio 2017	9/16/2024 11:26 AM	File folder	
Visual Studio 2022	9/10/2024 3:49 PM	File folder	

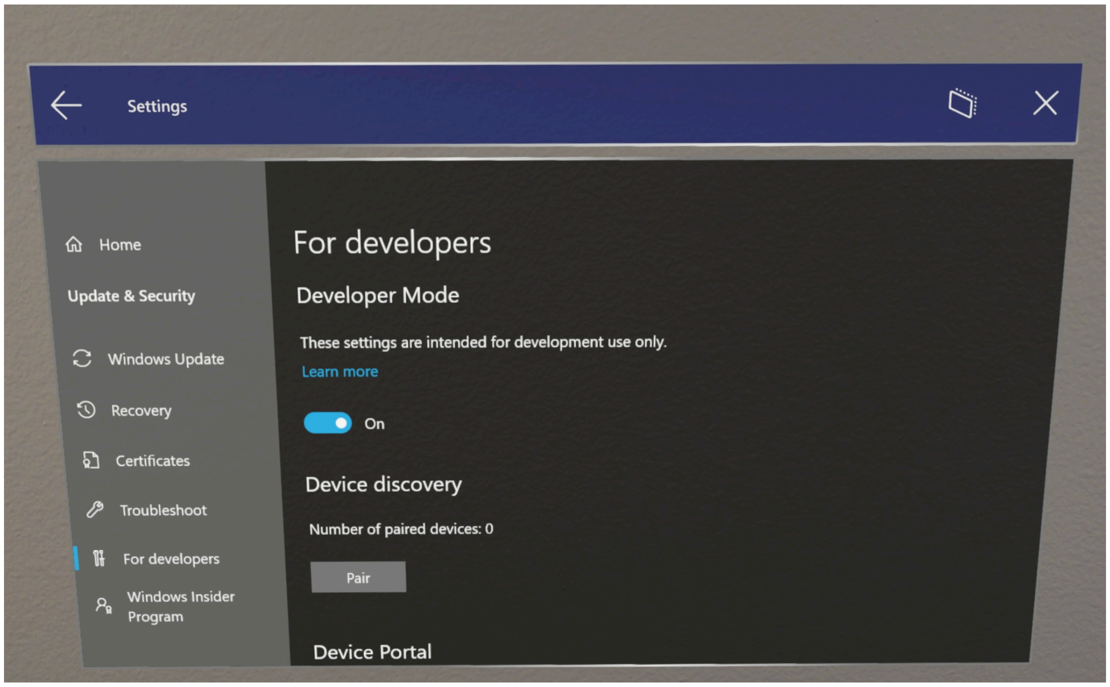
Folder: Documents

5. You can open the build by left clicking the ConfiguratorProject.sln solution file in the directory you built it to. Ensure that you open it with the Visual Studio Community 2022 installed with the correct modifications as detailed in the previous section on installing the required software to Develop the Mixed Reality Configurator.



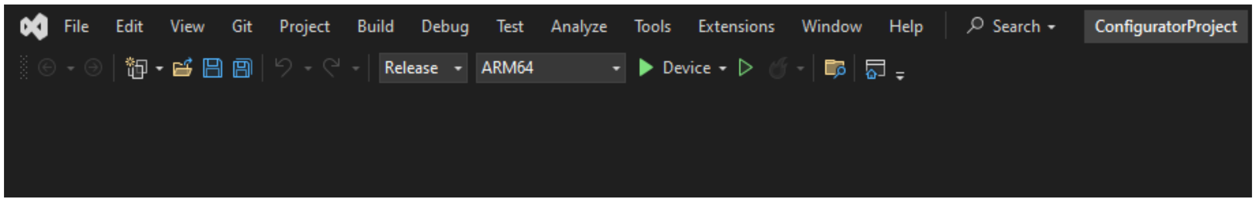
Visual Studio 2017	9/16/2024 11:26 AM	File folder	
Visual Studio 2022	9/10/2024 3:49 PM	File folder	
ConfiguratorProject.sln	11/14/2024 8:48 PM	Visual Studio Solu...	8 KB

6. Enable Developer Mode on the HoloLens. This is in the “Update & Security” section of the Settings app, under the “For developers” tab.

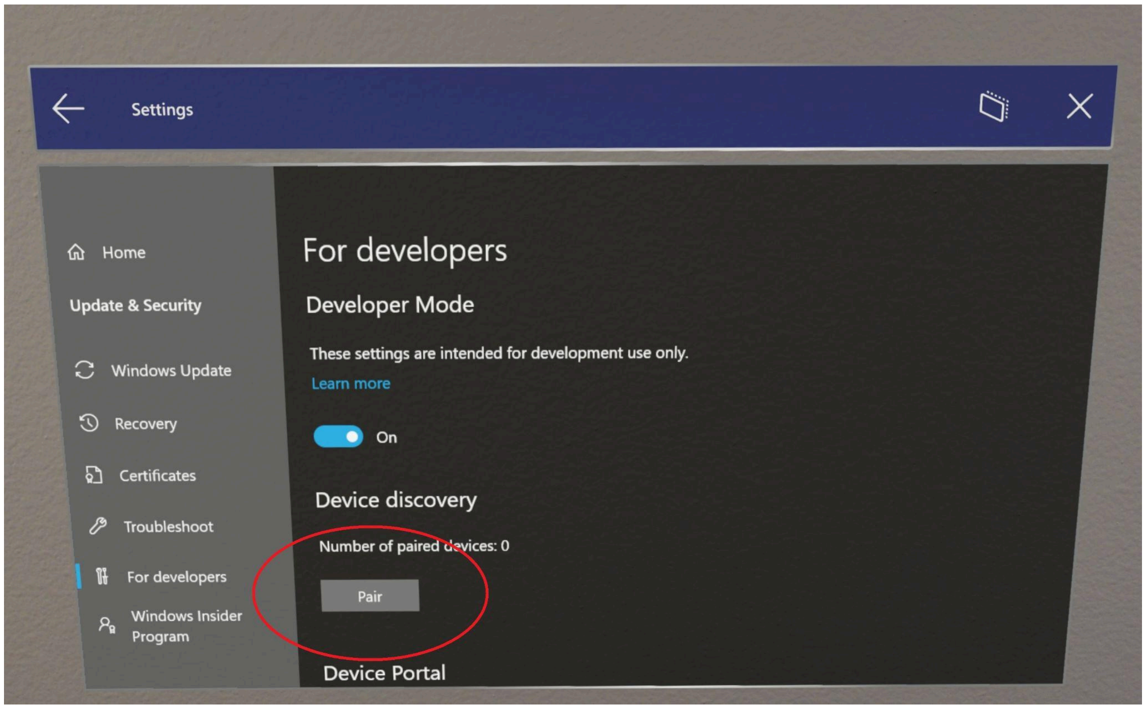


The “For developers” tab in HoloLens [1]

7. Ensure that “Device Discovery” is toggled on.
8. Connect your HoloLens to your device via USB.
9. After the ConfiguratorProject.sln solution file opens, make sure that the settings at the top match the screenshot below. Then click on the green arrow to the right of “Device.” This will deploy the app to your HoloLens as “ConfiguratorProject.”



10. If this is the first time deploying from this device, you will need to enter a PIN to allow deployment. This will appear a few minutes after you clicked the green arrow. You will need to put your HoloLens on and navigate to the “For developers” tab under “Update & Security” in the HoloLens settings app. Once there you can click on the “Pair” button under “Device discovery.”



The “For developers” tab with “Pair” button circled in HoloLens [1]

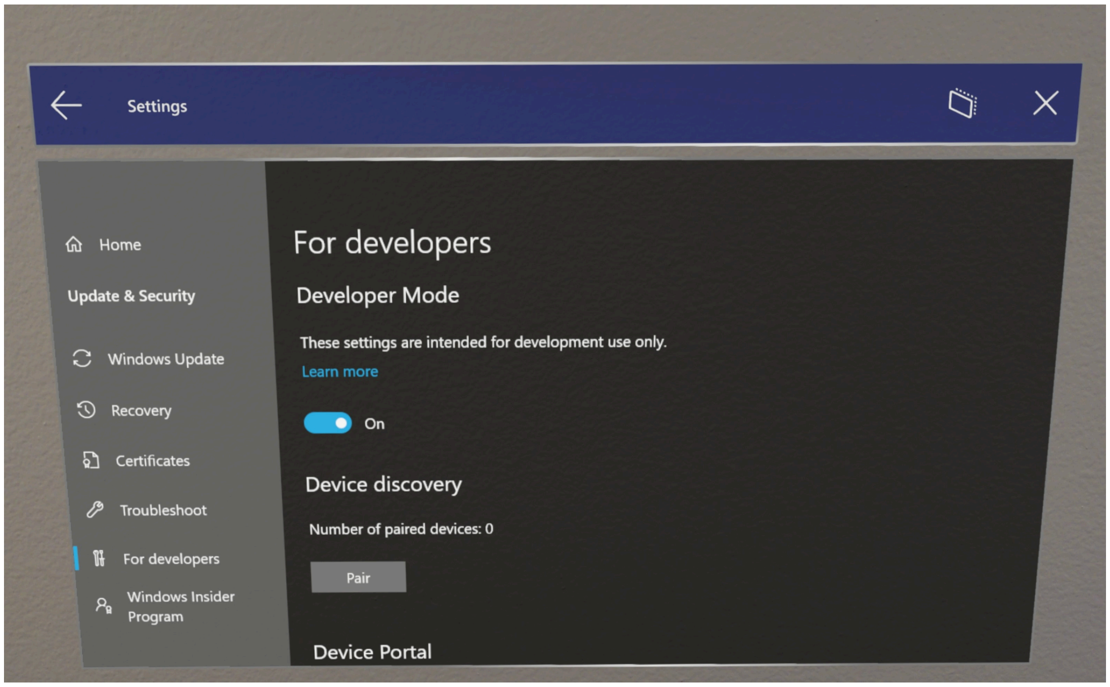
11. Enter the PIN that appears in the prompt in Visual Studio Community 2022.
12. The app should successfully deploy. As long as the device you deployed from is paired to your HoloLens, you won’t have to enter a PIN again on successive deployments.

Windows Device Portal

This helpful web server is available on all HoloLens 2’s running Microsoft’s official HoloLens operating system [1]. It’s the simplest way to access folders on a HoloLens that are hidden behind administrative access on the default file explorer available on HoloLens with the HoloLens connected to a compatible computer. This section details how to use it to access the csv files generated by Mixed Reality Configurator. For general use of the windows device portal see [1].

Accessing Files via Windows Device Portal over USB

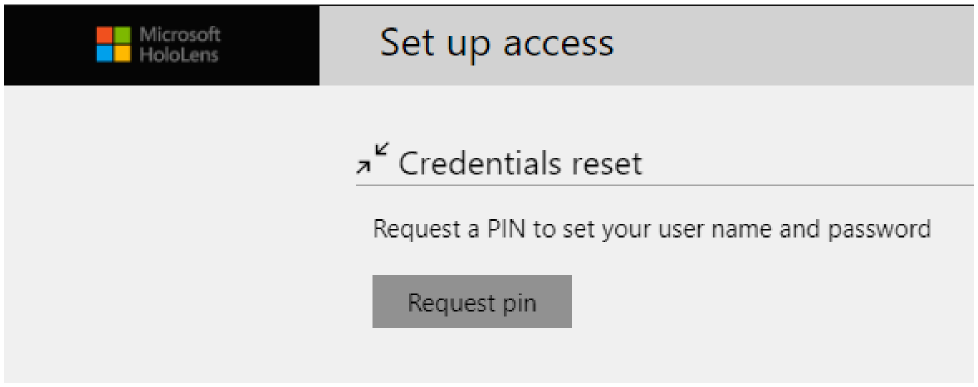
1. Enable Developer Mode on the HoloLens. This is in the “Update & Security” section of the Settings app, under the “For developers” tab.



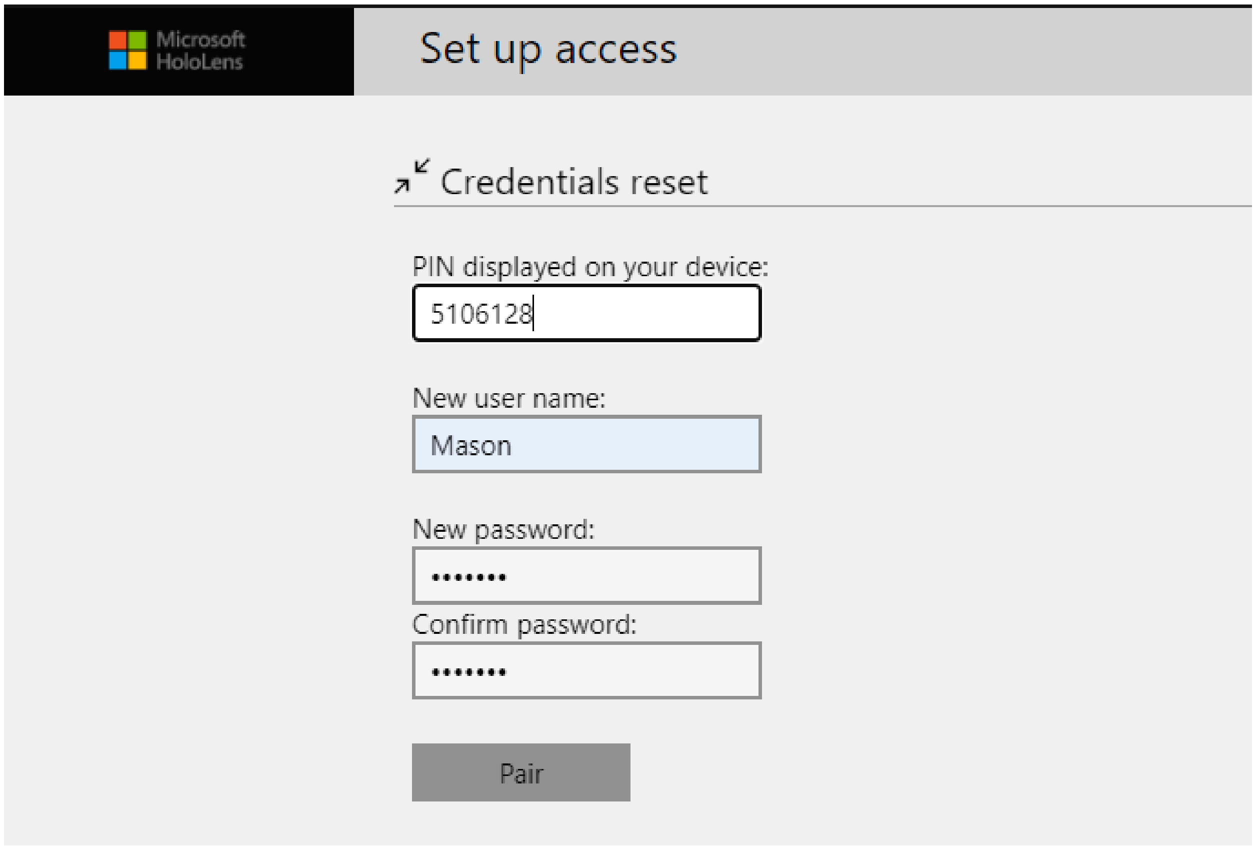
The “For developers” tab in HoloLens [1]

2. Scroll down and enable “Device Portal.”
3. Ensure that “Device Discovery” is toggled on.
4. Connect your HoloLens to the device you wish to access its filesystem from via USB.
5. Record the Ethernet IP address at the bottom of the “For Developers” tab.
6. From a browser on your computer go to “https://[the recorded Ethernet IP address]/devicePair.htm” if you need to pair your HoloLens’s Windows Device Portal, or if you’ve already done this go to “https://[the recorded Ethernet IP address]” and skip to step 9.

7. Request the PIN from your web browser and check your HoloLens for the displayed PIN.



8. Enter the new PIN as well as a new Username and Password. The password must be 7 or more characters. Press the Pair button.



Set up access

Credentials reset

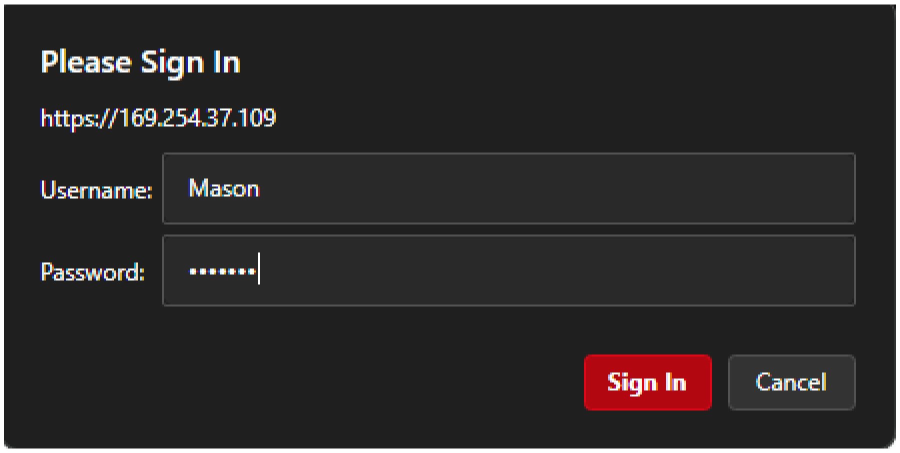
PIN displayed on your device: 5106128

New user name: Mason

New password: •••••••

Confirm password: •••••••

9. Enter your Username and Password and Sign in.



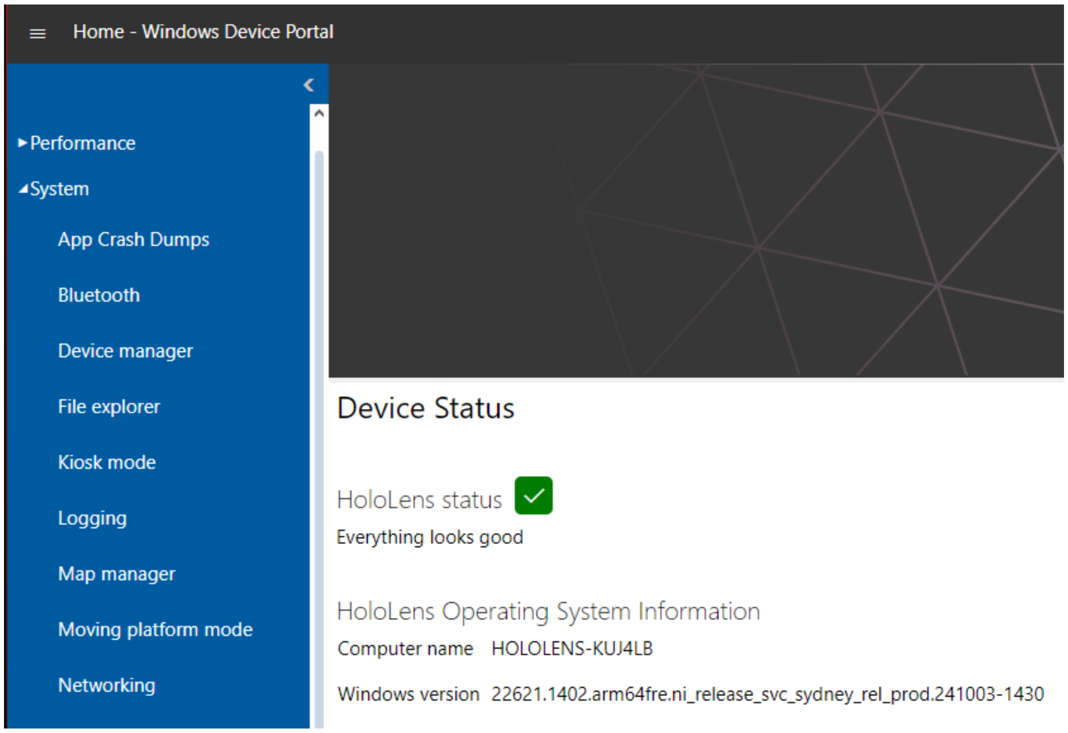
Please Sign In

https://169.254.37.109

Username: Mason

Password: •••••••

10. Navigate to System/File Explorer/User Folders/ LocalAppData/ConfiguratorProject/LocalState



Device Status

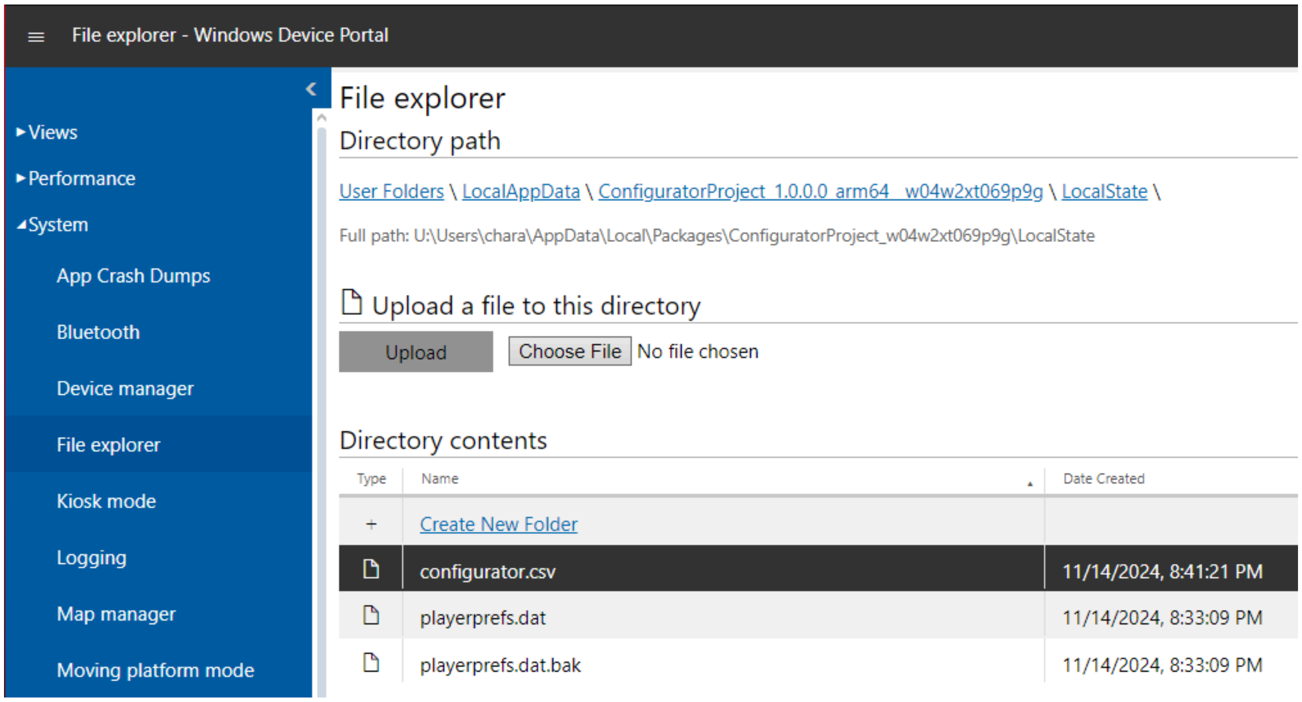
HoloLens status ✓

Everything looks good

HoloLens Operating System Information

Computer name HOLOLENS-KUJ4LB

Windows version 22621.1402.arm64fre.ni_release_svc_sydney_rel_prod.241003-1430



File explorer

Directory path

User Folders \ LocalAppData \ ConfiguratorProject_1.0.0.0_arm64__w04w2xt069p9g \ LocalState \

Full path: U:\Users\chara\AppData\Local\Packages\ConfiguratorProject_w04w2xt069p9g\LocalState

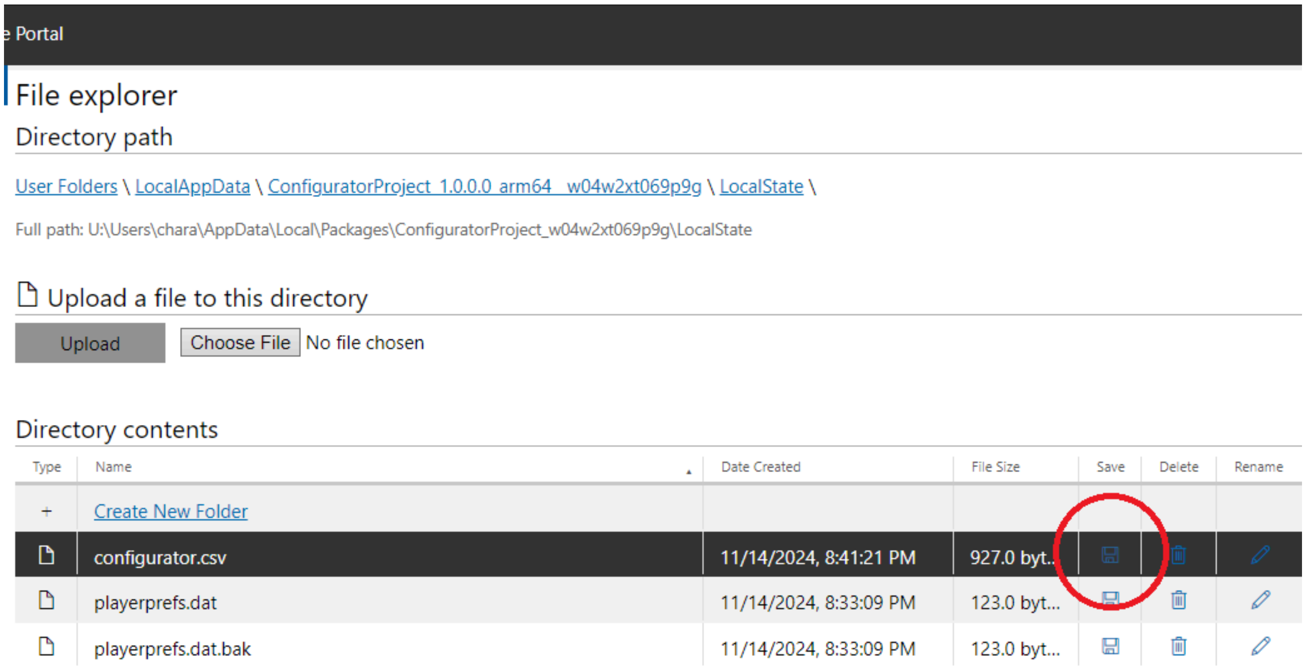
Upload a file to this directory

No file chosen

Directory contents

Type	Name	Date Created
+	Create New Folder	
	configurator.csv	11/14/2024, 8:41:21 PM
	playerprefs.dat	11/14/2024, 8:33:09 PM
	playerprefs.dat.bak	11/14/2024, 8:33:09 PM

11. Download the configurator.csv file by clicking the save (floppy disk) icon next to it.



Portal

File explorer

Directory path

User Folders \ LocalAppData \ ConfiguratorProject_1.0.0.0_arm64__w04w2xt069p9g \ LocalState \

Full path: U:\Users\chara\AppData\Local\Packages\ConfiguratorProject_w04w2xt069p9g\LocalState

Upload a file to this directory

Upload | Choose File | No file chosen

Directory contents

Type	Name	Date Created	File Size	Save	Delete	Rename
+	Create New Folder					
	configurator.csv	11/14/2024, 8:41:21 PM	927.0 byt...			
	playerprefs.dat	11/14/2024, 8:33:09 PM	123.0 byt...			
	playerprefs.dat.bak	11/14/2024, 8:33:09 PM	123.0 byt...			

References

[1] Hamalawi, Mohammed, et al. “Using the Windows Device Portal - Mixed Reality.” *Mixed Reality | Microsoft Learn*, Microsoft, 13 Apr. 2022, learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-the-windows-device-portal. Accessed 14 Nov. 2024

[2] “Hololens 2 Fundamentals: Develop Mixed Reality Applications - Training.” *Training | Microsoft Learn*, Microsoft, learn.microsoft.com/en-us/training/paths/beginner-hololens-2-tutorials/. Accessed 14 Nov. 2024.

[3] Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd ed., Addison-Wesley Professional, 2003.

[4] “Color.” *Unity - Scripting API*, Unity, 22 Nov. 2024, docs.unity3d.com/2022.3/Documentation/ScriptReference/Color.html.

[5] “Material.” *Unity - Scripting API*, Unity, 22 Nov. 2024, docs.unity3d.com/2022.3/Documentation/ScriptReference/Material.html.

[6] “Transform.” *Unity - Scripting API*, Unity, 22 Nov. 2024, docs.unity3d.com/ScriptReference/Transform.html.

[7] “Texture” *Unity - Scripting API*, Unity, 23 Nov. 2024, docs.unity3d.com/ScriptReference/Texture.html.

[8] “Shader.” *Unity - Scripting API*, Unity, 23 Nov. 2024, docs.unity3d.com/ScriptReference/Shader.html.

[9] Bean, Lola, Mark Ghanyem, et al. “Buttons - MRTK 2.” *Buttons - MRTK 2 | Microsoft Learn*, Microsoft, 23 June 2022, learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/button?view=mrtkunity-2022-05.

[10] Bean, Lola, Mark Ghanyem, et al. “Toggle Collection - MRTK3.” *Toggle Collection - MRTK3 | Microsoft Learn*, Microsoft, 21 Sept. 2022,

learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk3-uxcore/packages/uxcore/toggle-collection.

[11] “TextMeshPro.” *Unity - Manual*, Unity, 21 Nov. 2024, docs.unity3d.com/2021.3/Documentation/Manual/com.unity.textmeshpro.html.

[12] Brigsted, Thor, et al. “Siccity/Gltfutility: Simple Gltf Importer for Unity.” *GitHub*, 21 Oct. 2024, github.com/Siccity/GLTFUtility.