Renovation framework Documentation

Asset available on: https://u3d.as/3jub

Online documentation:

- **Last version:** https://bit.ly/RF-Doc-Last
- **Version 1.0:** https://bit.ly/RF-Doc-v1-0

For more information contact me:

- **Email:** dev.interactivelab@gmail.com

Interactive Lab

Version 1.0

Renovation Framework

Table of Contents

1. Introduction

The Renovation Framework is a package that offers developers a range of tools commonly found in simulation games. Despite its focus on "renovation", these tools are adaptable to diverse scenarios, offering flexibility to meet the specific requirements of your project. Furthermore, this package establishes a robust and expandable code foundation, allowing for the seamless integration of custom tools without concerns about compatibility.

Included within the Renovation Framework asset:

- **Trash Collection Tool:** Allows players to gather trash into a bag and dispose of it in trash bins.
- **Cleaning Tool:** Enables players to clean stains on walls, floors, or any surface within the environment.
- **Painting Tool:** Allows players to paint walls with a variety of colors and shapes.
- **Object Moving Tool:** Allows players to move objects and place new ones freely in their environment.
- **Tool Selection Menu:** Offers a convenient radial menu for players to switch between the different tools seamlessly.
- **Object Purchase Menu:** Allows players to browse and buy various objects to place in their environment.

Additionally, as a bonus feature:

- **Wall Creation Script:** Provides a script for quickly creating walls with openings, ideal for swiftly generating apartment and house layouts for renovation projects.

This documentation serves as a comprehensive guide for using the Renovation Framework package. It provides detailed explanations of each feature and tool included in the package, as well as the diverse ways to integrate them into your Unity project. If you have any further questions or need help, feel free to contact me. I am available to provide personalized support and address any inquiries to help you make the most out of the Renovation Framework.

2. Quick Start Guide

For the asset to function properly, you must follow the following initialization steps. Of course, you can skip the steps related to tools you will not be using.

2.1. Dependencies

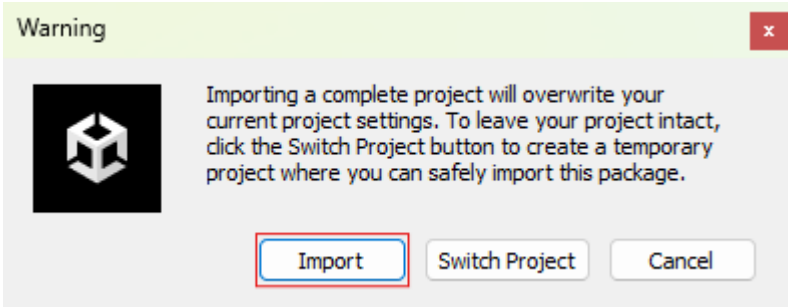
RF (Renovation Framework) relies on many Unity packages, which will be automatically installed when RF is imported into the project.

These dependencies are:

- **Input System**
- **Render Pipelines Core**
- **Render Pipelines Universal**
- **Shader Graph**
- **Cinemachine** Optional (can be removed if you already have your own player setup).
- **Text Mesh Pro** Optional (used only in the demo scene and can be removed if not needed).

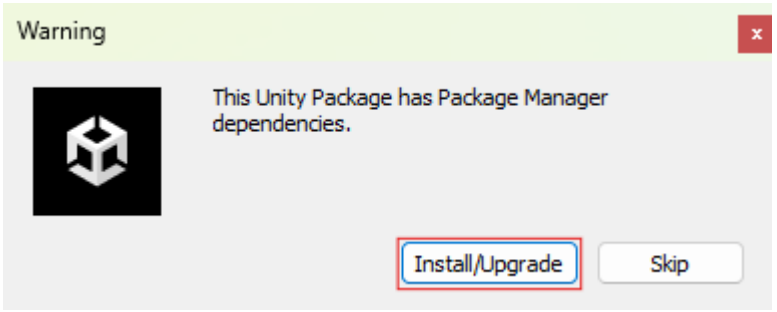
2.2. Importing the Renovation Framework

1. Use the Package Manager to download and import the Renovation Framework into your project.
2. Click "Import" to continue.

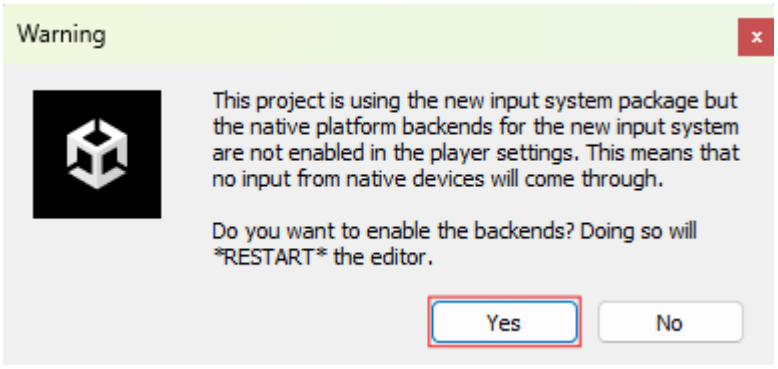


Note: Although you may choose to overwrite your project settings, only the Tag Manager is required for the demo scene to function. If you prefer not to overwrite your Tag Manager settings, all necessary information is provided in later sections for the manual setup of the layers.

3. Install and update project dependencies.

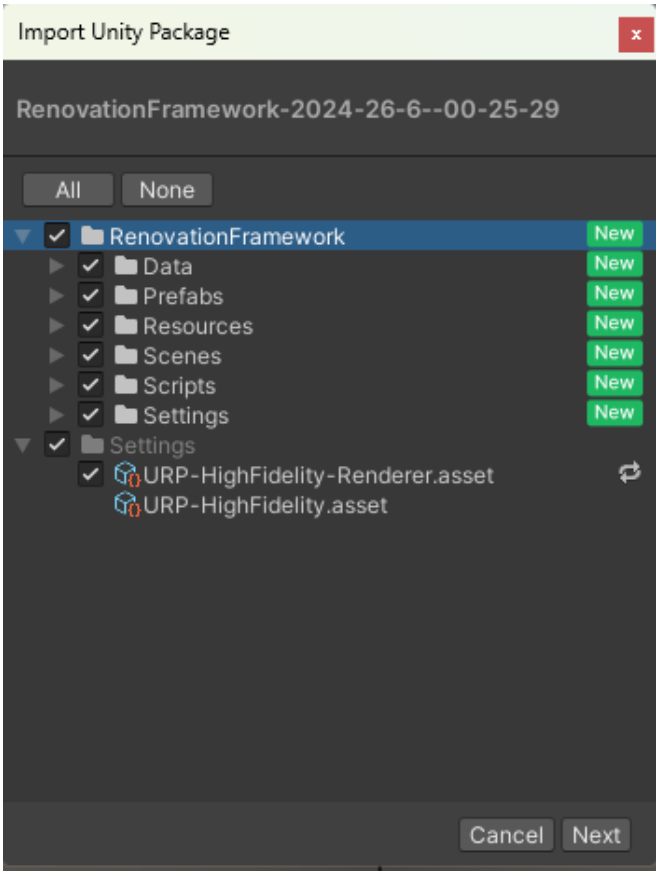


4. Click "Yes" to restart and enable the new input system for your project.

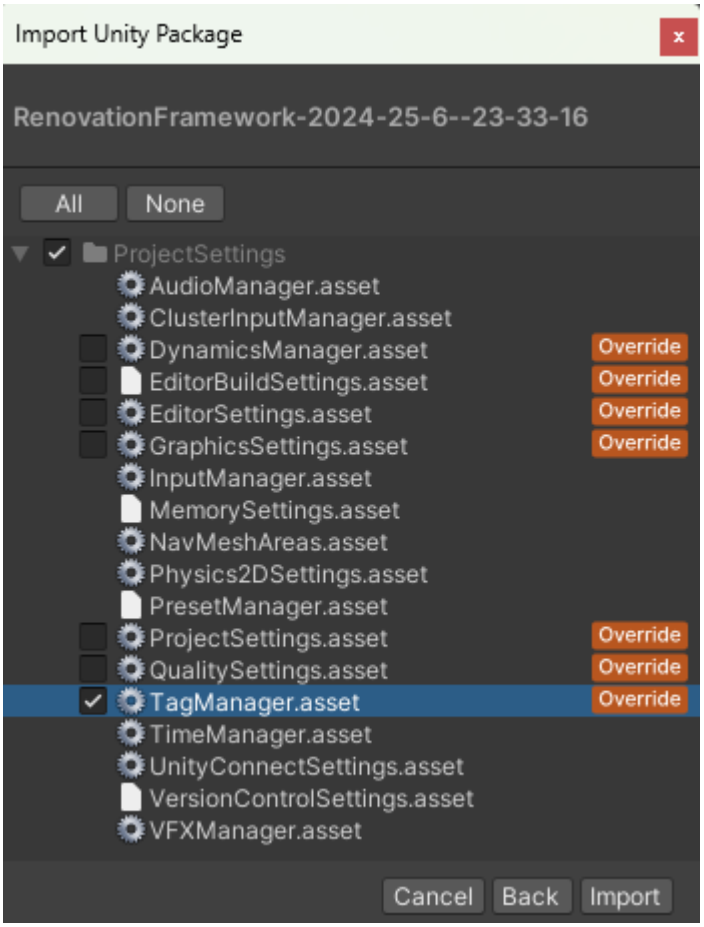


Note: After the restart, you will need to re-import the package.

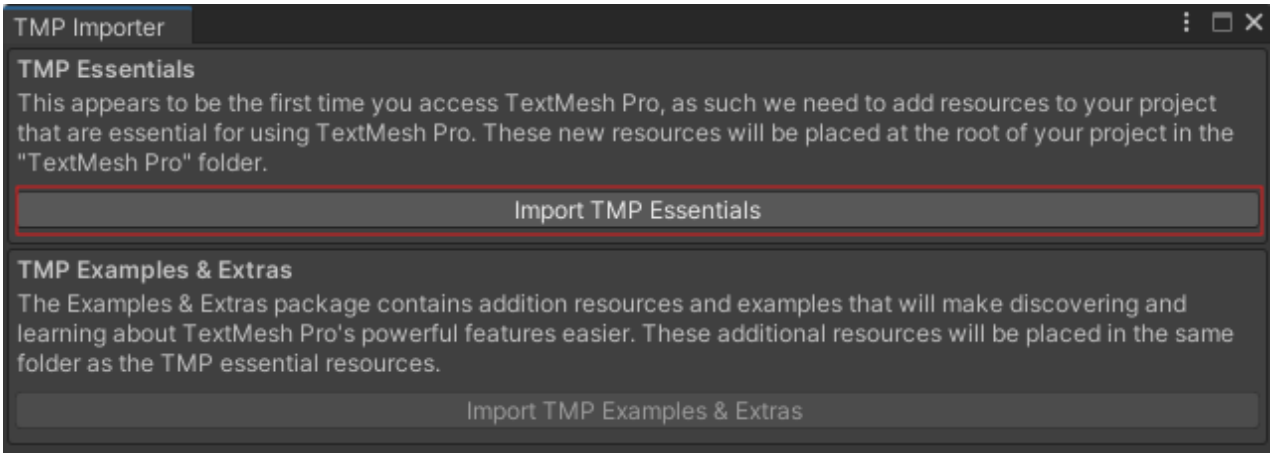
5. Select everything and click "Next"



6. Check only "TagManager.asset" and click "Import."

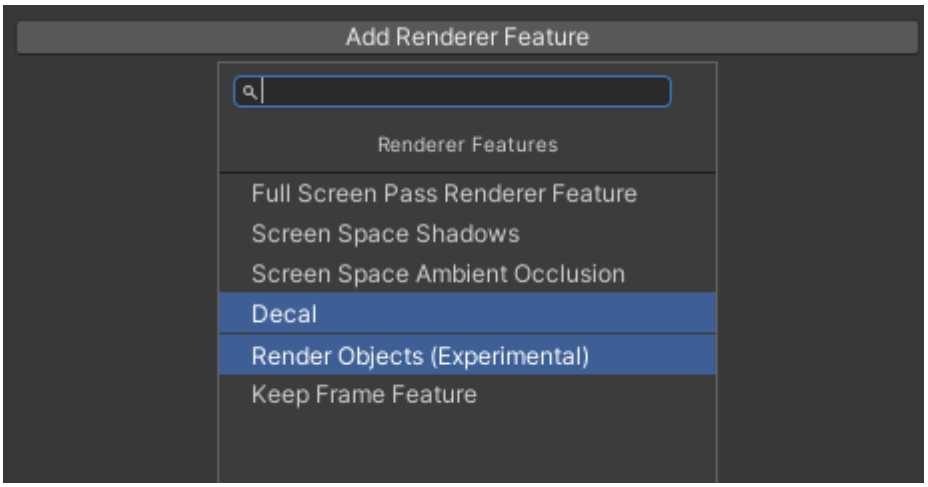


7. After importing the package and opening the demo scene for the first time, TextMesh Pro will prompt you to import TMP Essentials, which is required to display the demo scene text.

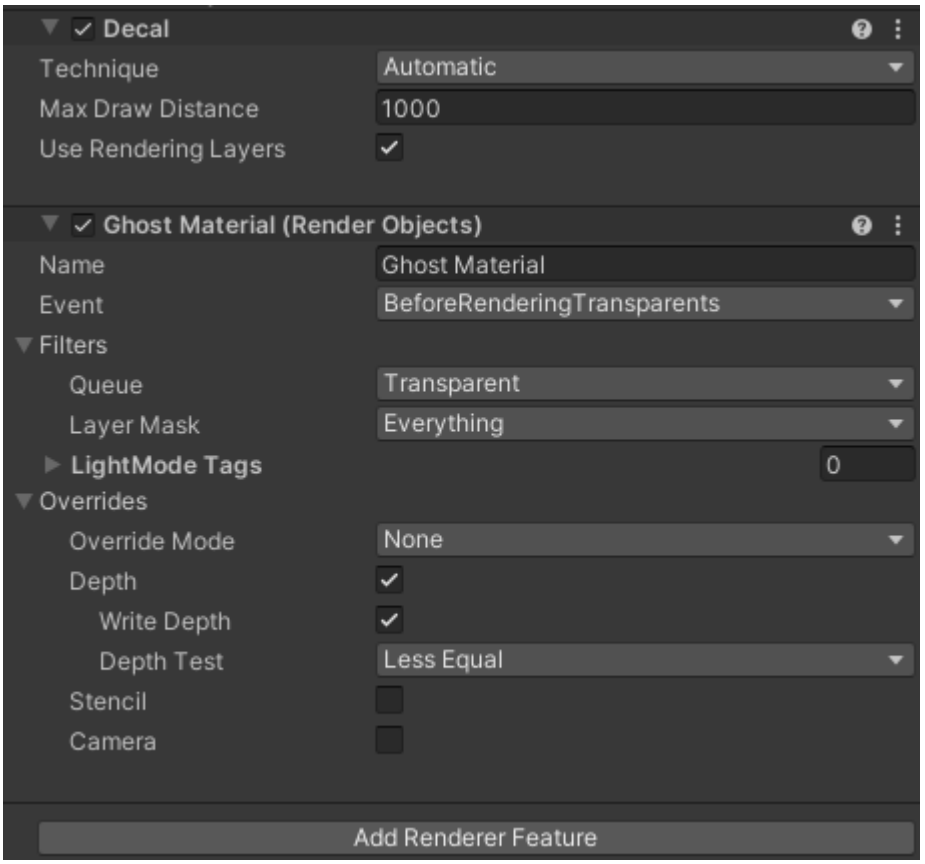


2.3. Setting up the Project

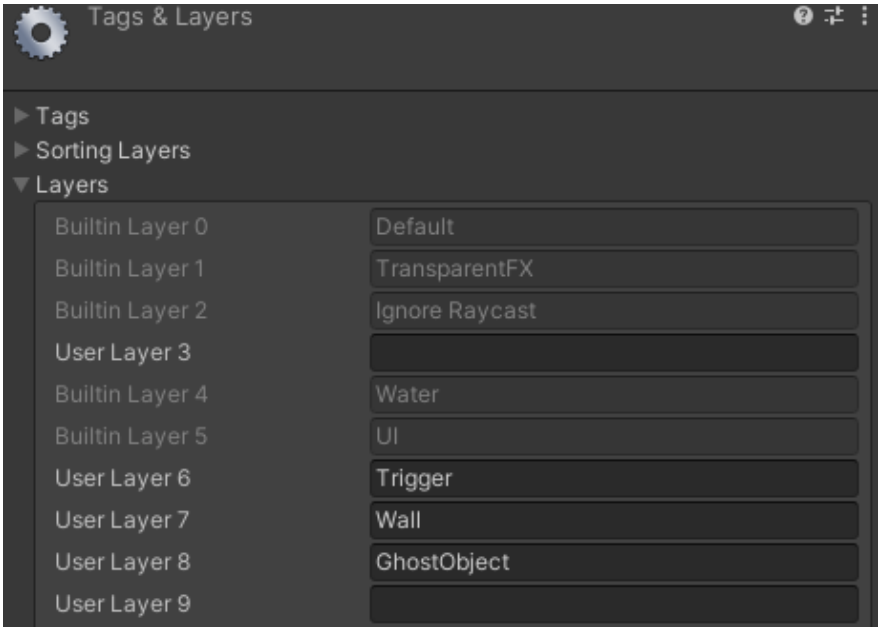
Begin the setup process by accessing your renderer data for URP or HDRP and add two new render features: **Decal** and **Render Objects**.



Then, configure them as follows.



If you have not imported the “TagManager.asset”, add the three bottom layers to your project as follows.

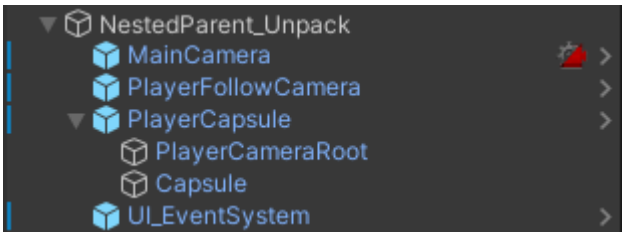


2.4. Setting up the Player

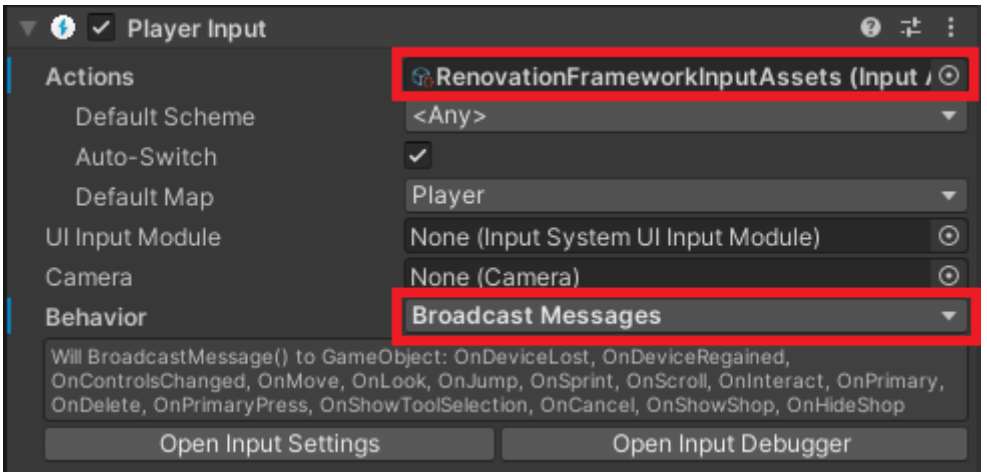
To quickly set up your player, you can import the prefab from "Prefabs/Player/Player_Spawn.prefab".

If you already have your player (for example, the First Person Controller from Unity), you will need to follow the steps below:

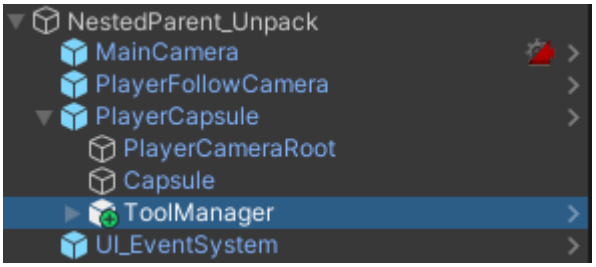
1. (Optional) Import the NestedParent_Unpack prefab into your scene from the "Assets/StarterAssets/FirstPersonController/Prefabs" folder.



2. Select the Player Capsule Game Object. In the Player Input script, set "RenovationFrameworkInputAssets" as the new Actions Asset. Then, set the Behavior variable to "Broadcast Messages".

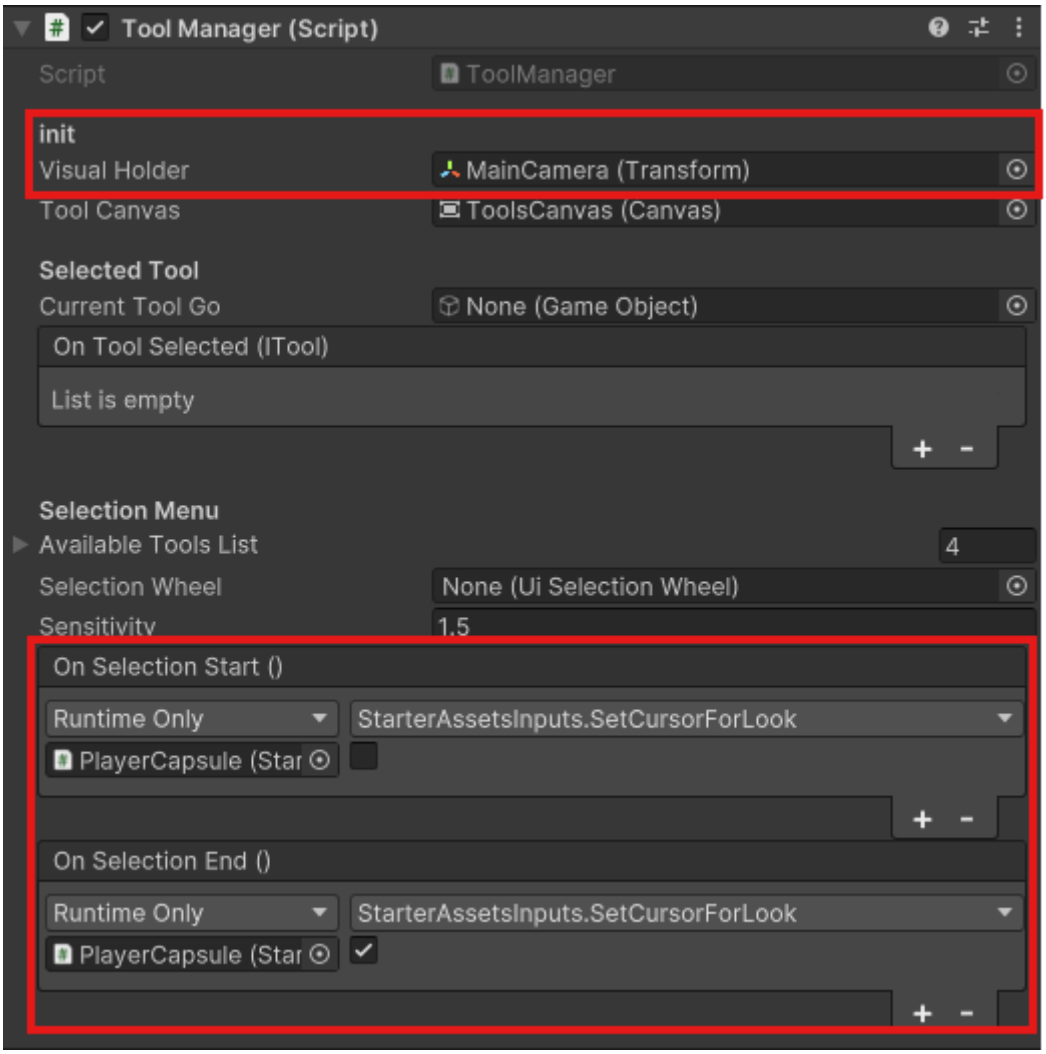


3. From the "~/Prefabs/Manager" folder, drag and drop the Tool Manager prefab into your hierarchy as a child of the Player Capsule.

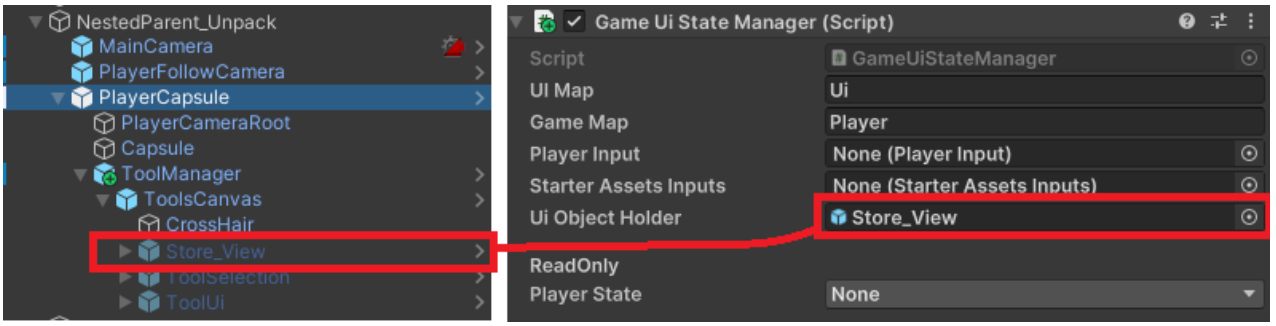


4. Select the Tool Manager Object in your scene, then initialize the Visual Holder variable with the Main Camera Game object and the UnityEvents OnSelectionStart and OnSelectionEnd with a call to the SetCursorForLook function in the Starter Assets Inputs script found on the Player Capsule.

   Note: The SetCursorForLook (bool value) function is a custom function. If you are using your own player controller script, you will need to provide similar functions which offer the feature to lock/unlock the cursor to allow the player to interact with the UI.



5. Add the “Game UI State Manager” script to the Player Capsule object and assign the UI Object Holder variable with the Store_View Game Object from your hierarchy.



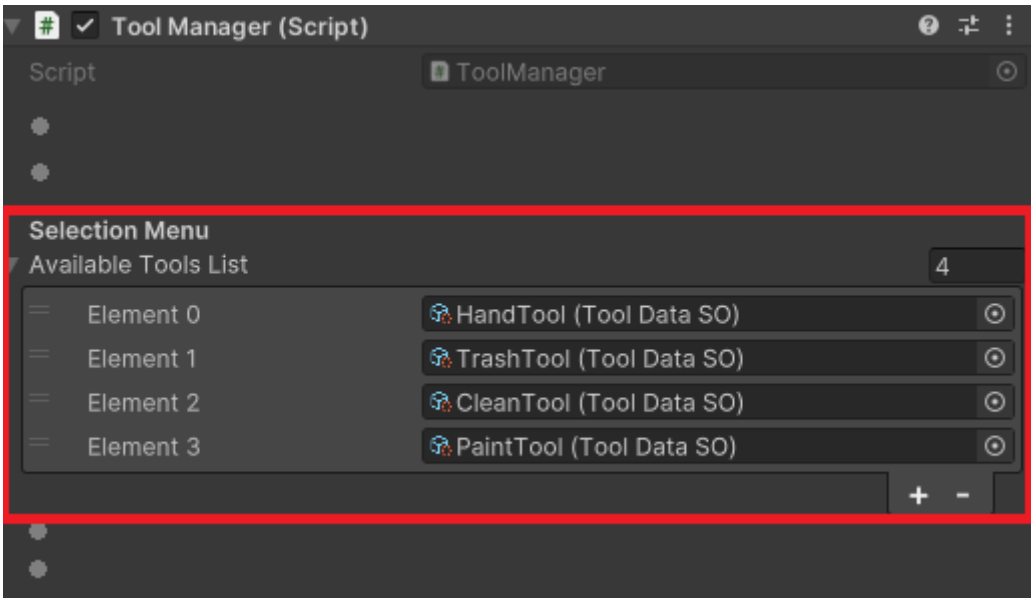
6. Enjoy.

3. Tool Manager

The Tool Manager, as its name suggests, is responsible for handling the selection and deselection of tools; it also manages the instantiation of the required prefabs when a tool is selected.

The Renovation Framework includes four tools. You can add them to the Player by dragging and dropping them from the "Assets/RenovationFramework/Data/Tools/" folder to the Available Tools List field in the Tool Manager. This will automatically display them in the Tool Selection Menu (left click while in Play Mode) and allow the Player to use them

.



When a tool is selected, its prefab will be loaded and added as a child object of the Tool Manager.

To edit the settings for a tool, you will need to open the tool prefab located in the corresponding folder at "Assets/RenovationFramework/Prefabs/Tools/". These settings can also be edited at runtime by retrieving the tool class from the Tool Manager:

```
public T GetOrAddTool<T> (ToolData toolData) where T: Tool;
```

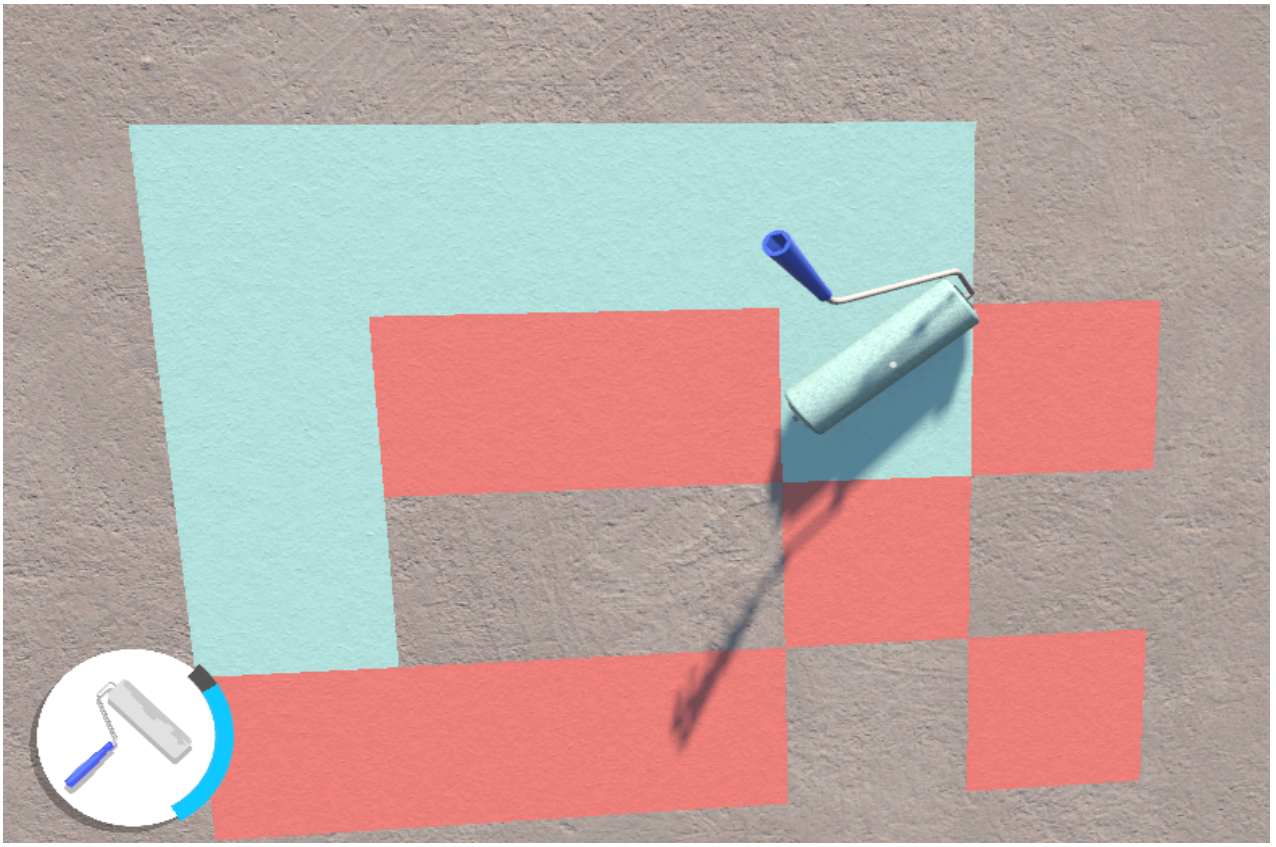
Calling this function will return the Tool if it is already loaded or load it without selecting it, then return it.

Each of the four tools implements the logic in a corresponding class that inherits from the Tool class.

All the visual feedback for these tools is managed by other classes that inherit from the Tool Visual class. All the functions in these classes are virtual, so you can extend them to add animations, particles, effects, sounds, etc., without affecting the logic of the tool.

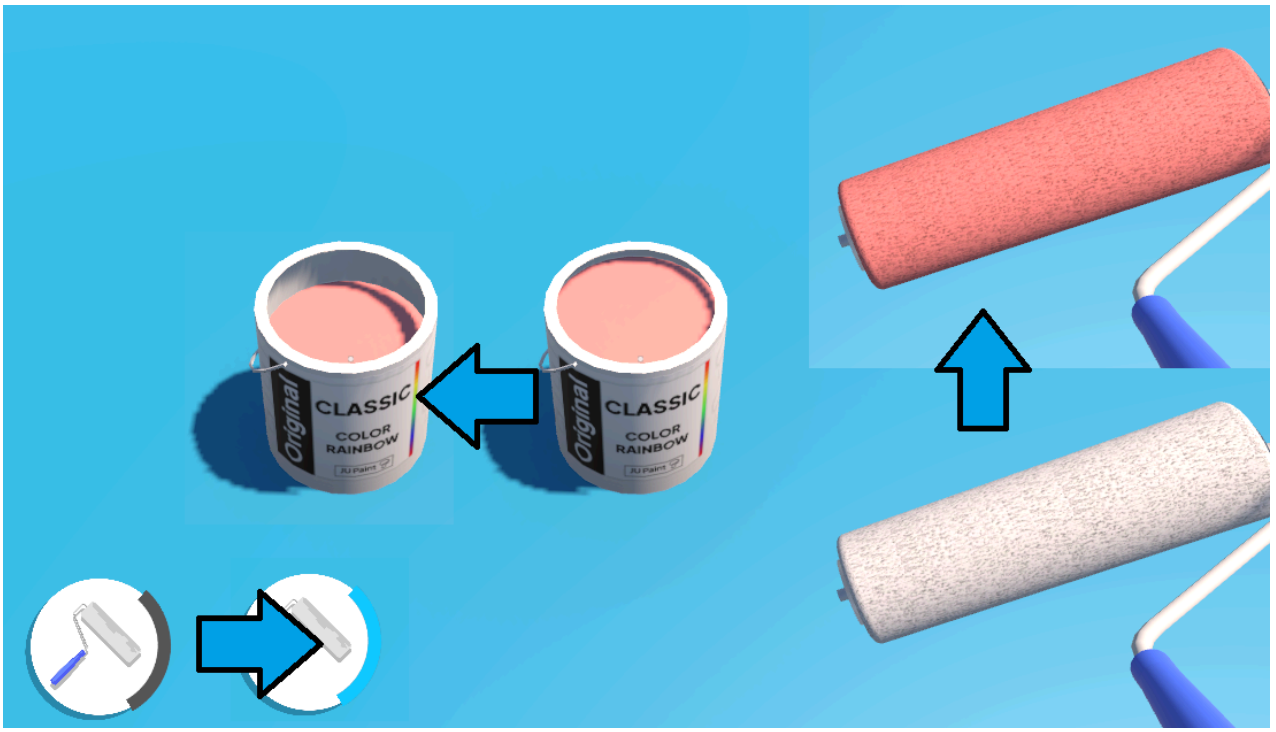
4. Painting Tool

The paint tool allows you to paint on walls using a paintbrush by holding the left click while moving the brush over the wall.



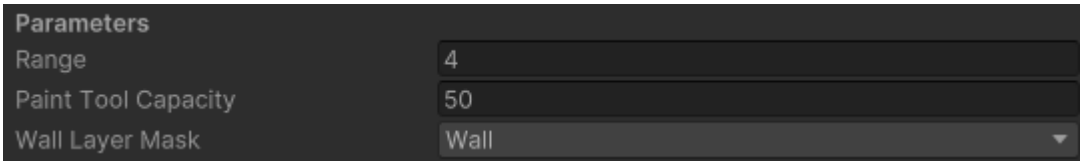
The paintbrush can hold a specified amount of paint, which can be adjusted using the “Paint Tool Capacity” variable.

To fill the paintbrush with paint, left click on a paint bucket. The brush will then reflect the color of the paint, and the UI will display the remaining amount of paint on the brush. The bucket will also show the amount of paint left inside it.



Paint buckets can be obtained from the store, which is opened with the Tab key.

4.1. Settings

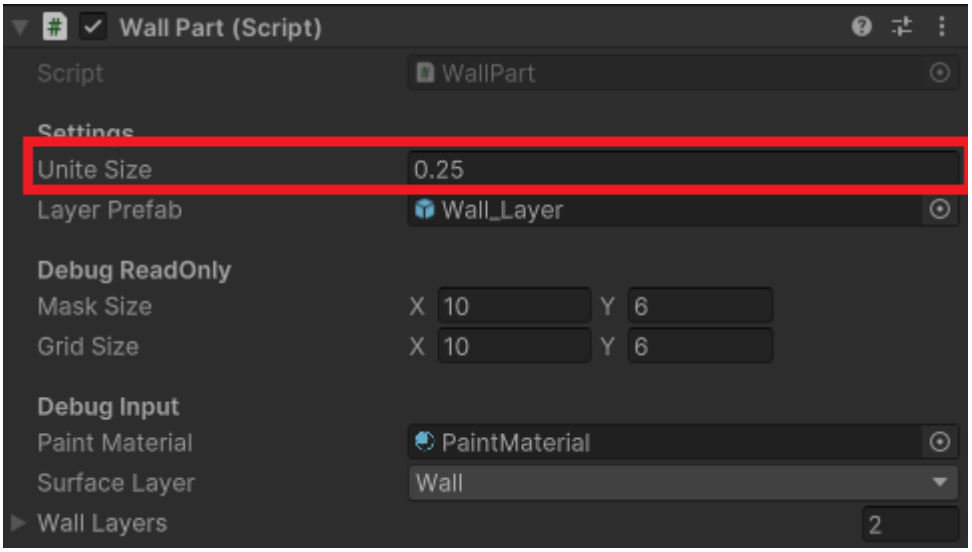


- **Range:** The maximum distance from which you can use the paintbrush. (Default: 4)
- **Paint Tool Capacity:** The maximum number of tiles that can be painted. (Default: 50)
- **Wall Layer Mask:** The layer on which painting is allowed and represents a wall.

4.2. Create a Wall

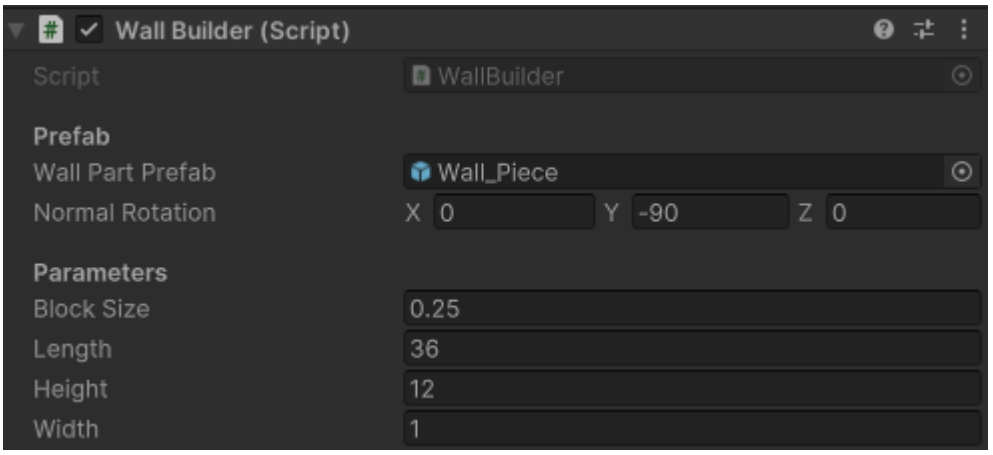
To add a paintable wall to your scene, drag and drop the Wall_Piece.prefab, then resize it using the scale parameter on the wall piece's transform.

The size of the tiles can be changed using the Unit Size variable on the Wall Part Game Object. Keep in mind that the wall needs to be scaled by values that are multiples of the Unit Size value to prevent having cut tiles.



To quickly create walls, you can also use the Wall Builder script.

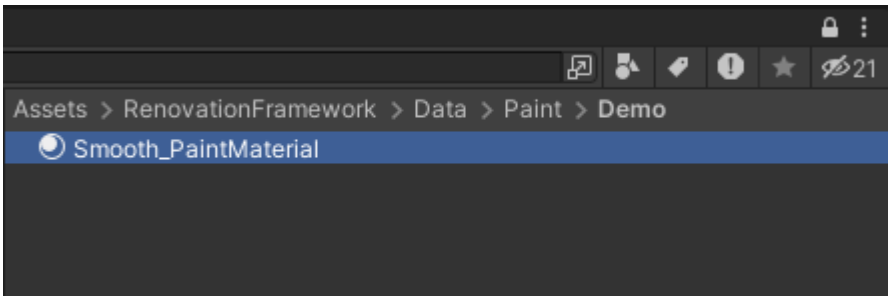
Note: Ensure that the Block Size variable in the Wall Builder matches the Unit Size variable in the Wall Part Prefab you provided.

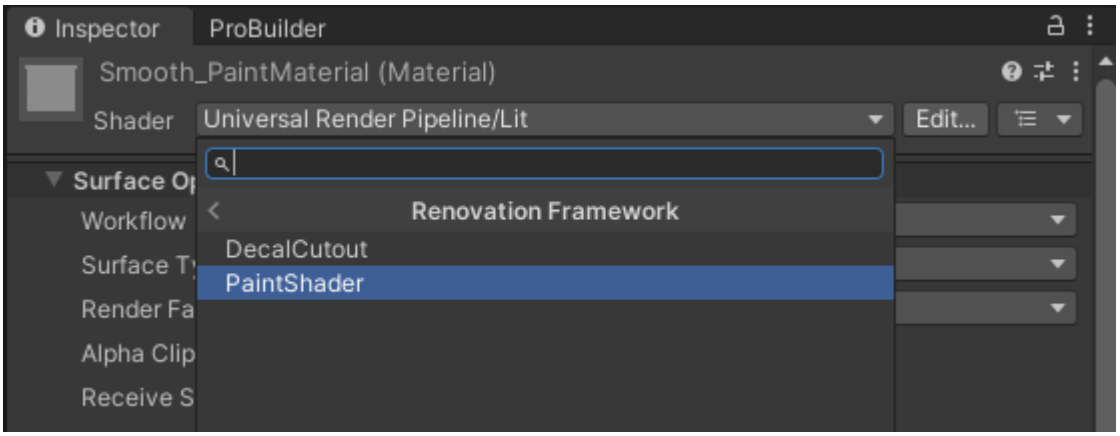


More information is available in the "painting tool" section in the Demo Scene included with the package.

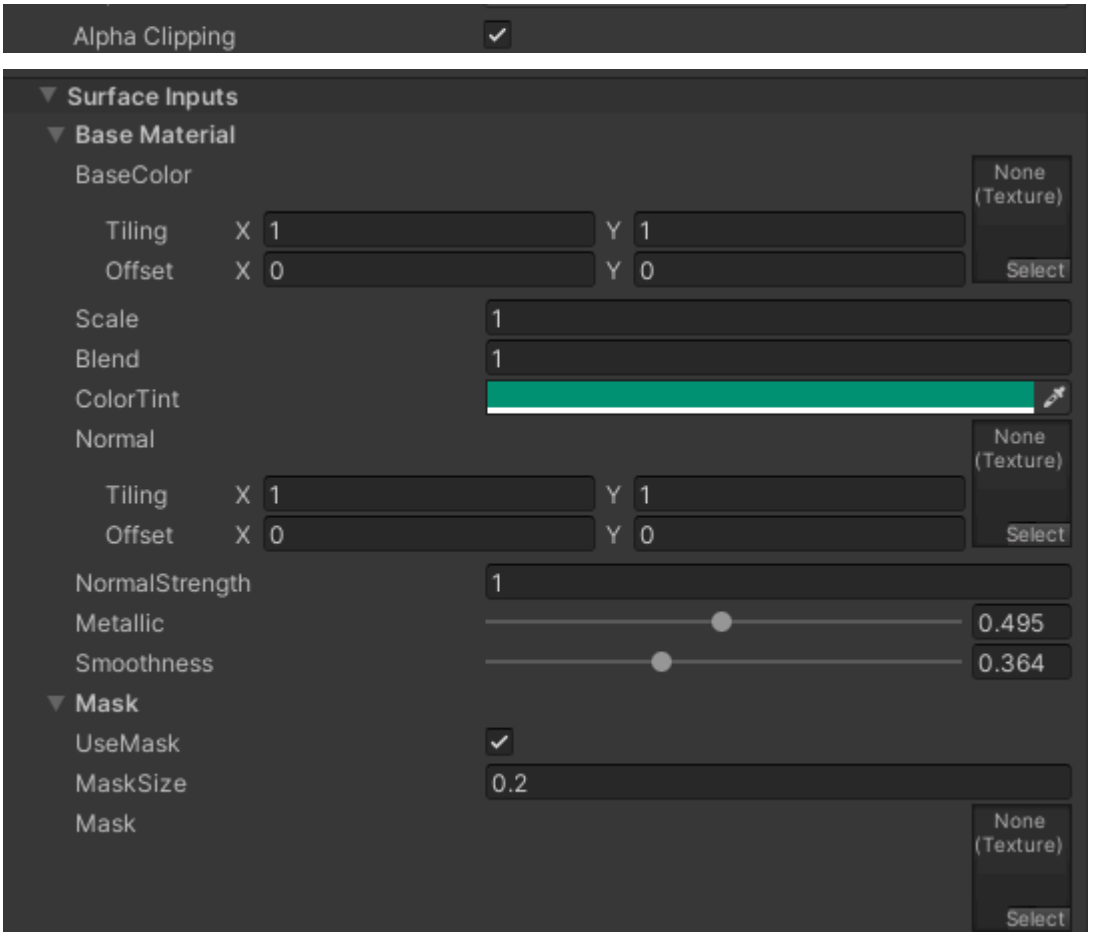
4.3. Create New Paint

Create a new Unity Material and name it as you wish, then assign "Renovation Framework/PaintShader" as the material's shader.

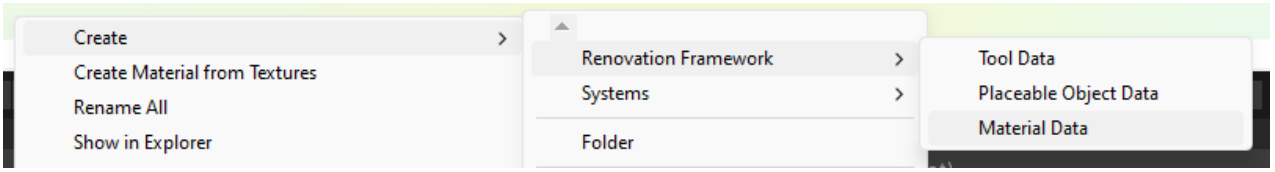




Ensure you check the boxes for "Alpha Clipping" and "Use Mask" on the material to ensure it functions correctly.

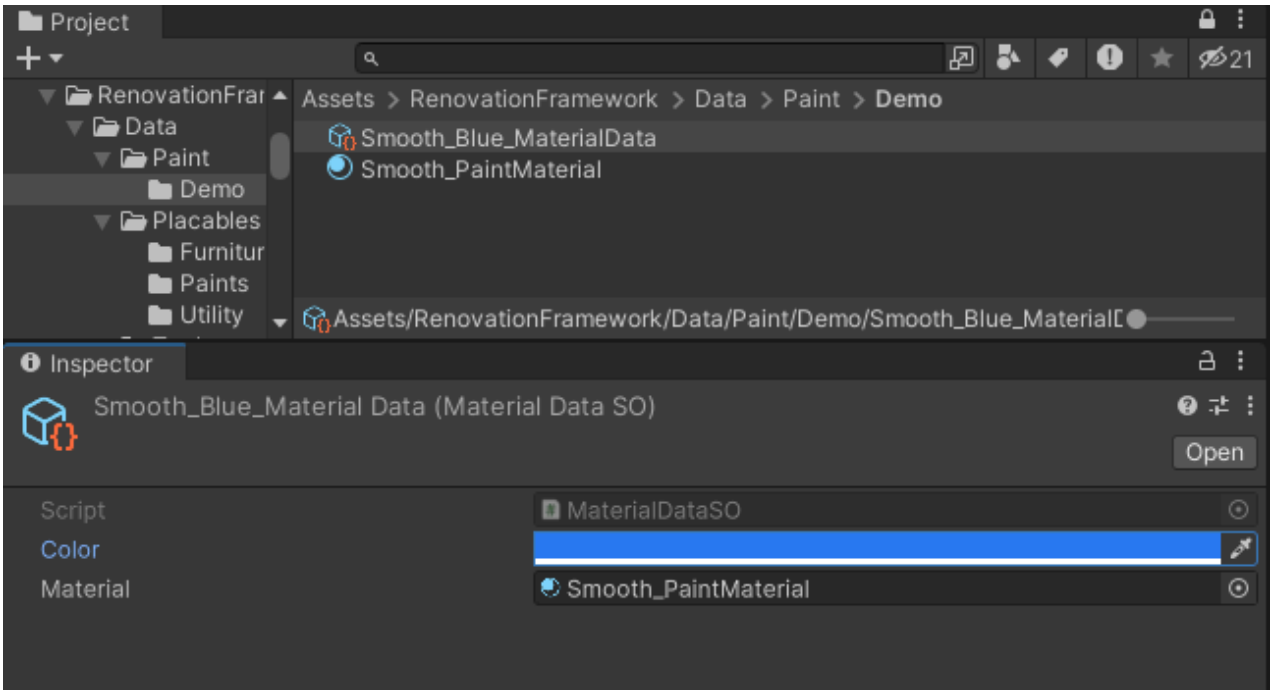


Next, create the Material Data from Create/Renovation Framework/Material Data.



Then, assign the material you previously created and choose the color you want. Note that this color will override the Color Tint you've set in the material.

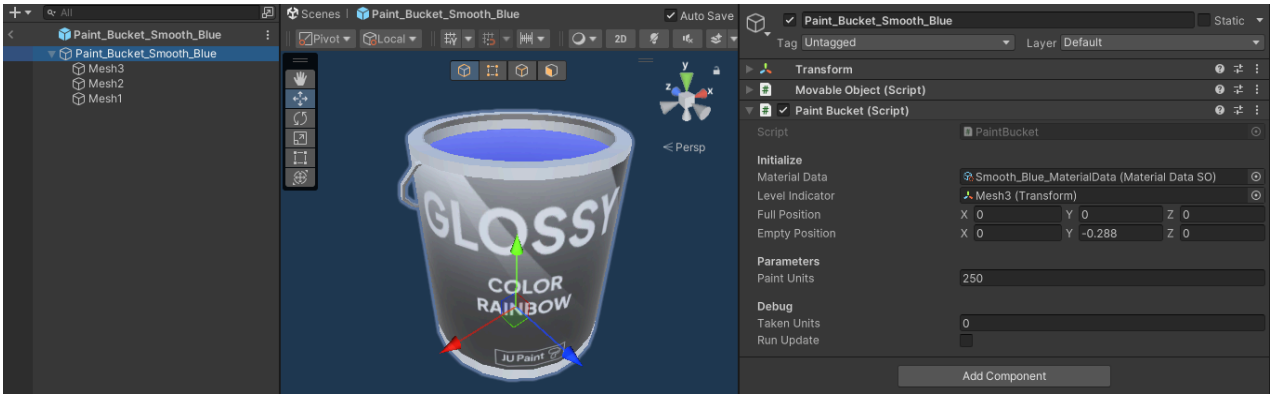
You can create many material data objects with the same paint material but different colors. However, if you need to have a different texture, normal map, smoothness, etc., you will need to create a new paint material (this may change in the future).



Finally, create a new paint bucket and add it to the store.

4.4. Create Paint Bucket

To speed up the process, duplicate an existing paint bucket from the "~/Prefabs/Paint" folder and modify the following information:



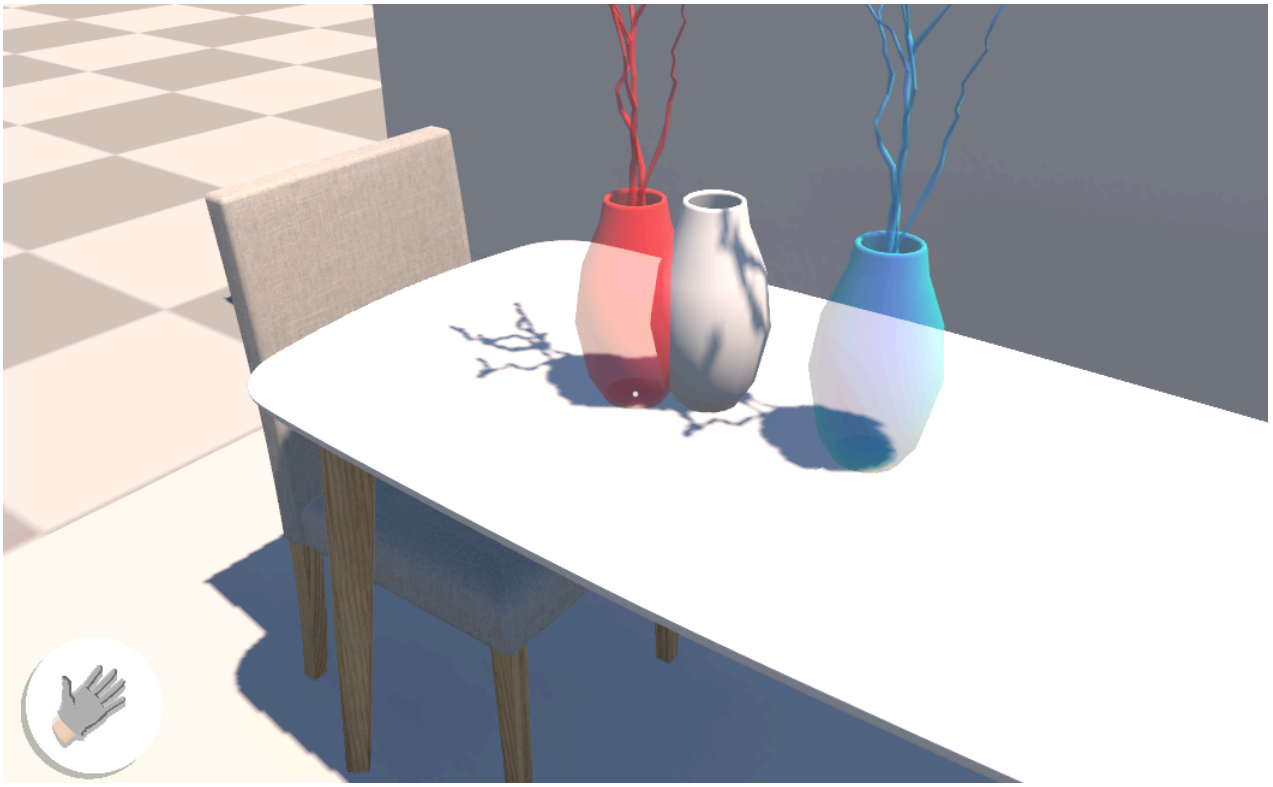
Assign the newly created Material Data to the prefab and save your changes.

You can change the Paint Units variable to adjust how much paint the bucket can hold (1 unit = 1 tile).

Now, add the bucket to the store or simply drag and drop it into your scene to allow the player to use it.

5. Moving Tool

The moving tool, when selected, provides a suite of interactions that enable you to move, rotate, and remove objects in your scene, or add new ones from the store.

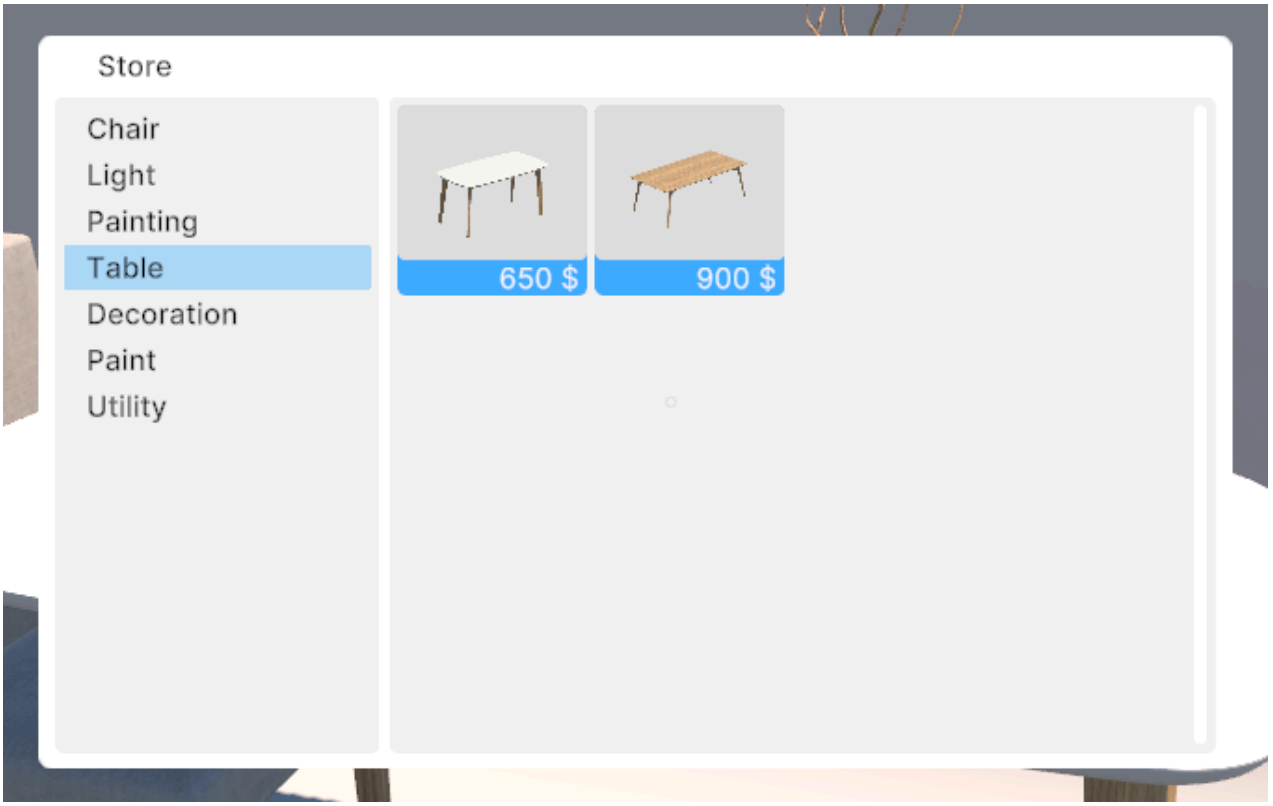


Left click on an object to initiate the moving process; this action causes the object to start moving and leaves a blue ghost object at its original position.

For a new position to be valid, the object must be correctly supported—for example, if one leg of a chair is not touching the ground, the chair is considered unsupported. Additionally, it must not overlap with any other objects in the scene. If the position is invalid, the preview object will turn red to indicate this.

While the player can still collide with the blue ghost, the moving preview object is not obstructed because the blue ghost will disappear once the new position is chosen.

While moving an object, you can rotate it using the scroll wheel. However, bear in mind that some objects cannot be rotated by the player because they auto rotate to match the orientation of the surface they are placed on.

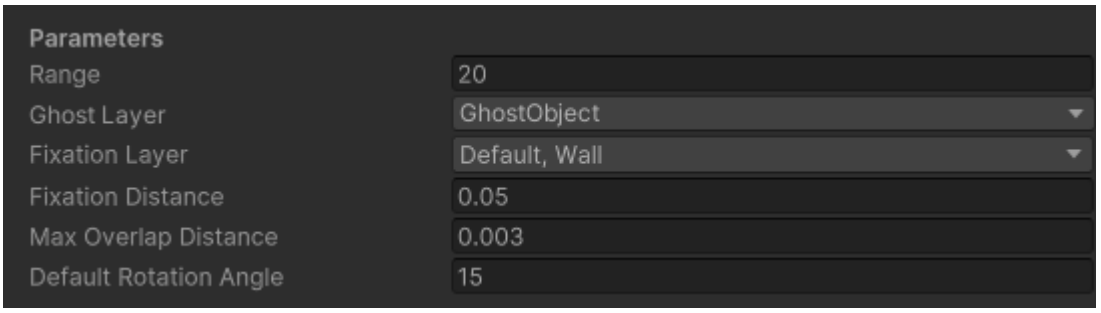


To add new objects to your scene, open the store using the Tab key and select the desired item. The left panel of the store features a simple category system to facilitate the search for specific objects.

When placing an object from the store or moving an existing one, you can cancel the action at any time by right-clicking with the mouse. If you are moving an existing object, it will return to its original position, but if you are placing a new object, the preview will simply disappear.

To remove an object, use a middle mouse click while targeting it. Upon doing so, any objects that were resting on top will fall to the ground due to lack of support.

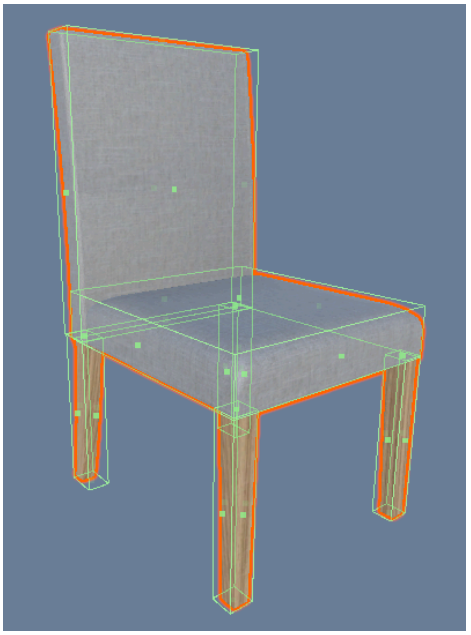
5.1. Settings



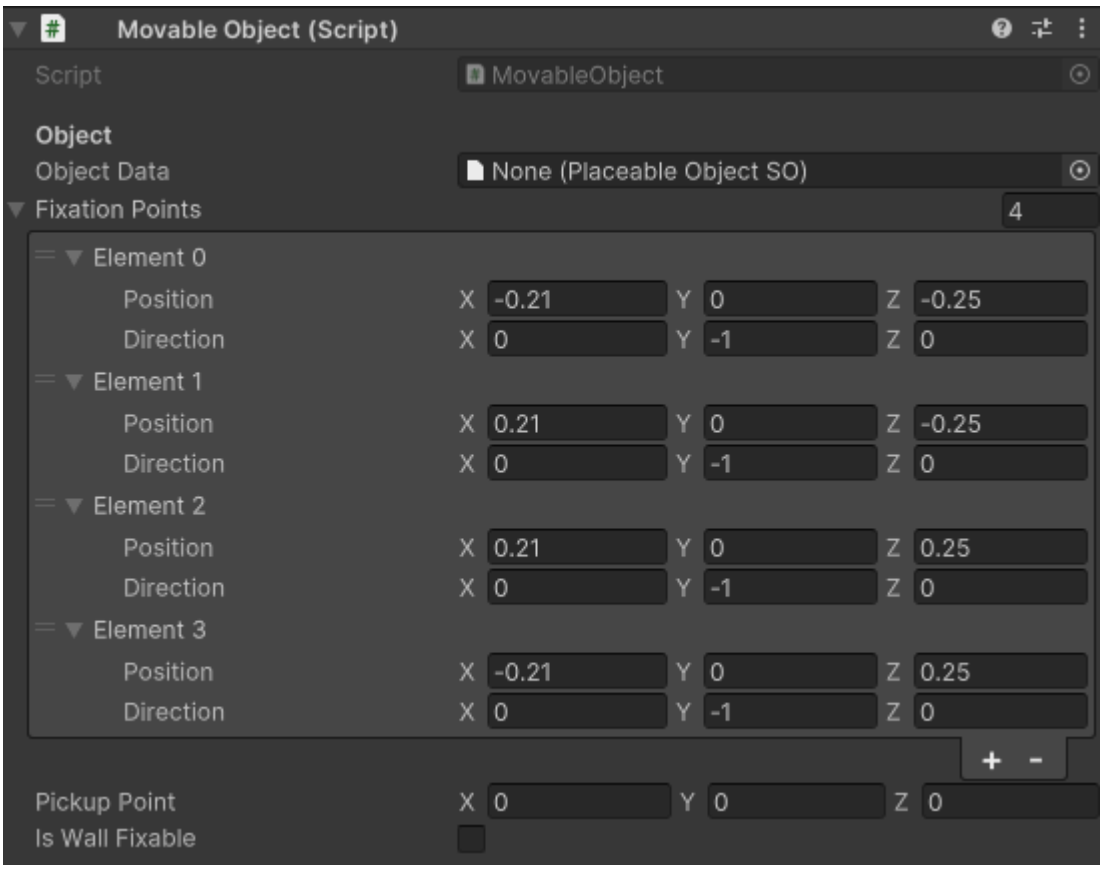
- **Range:** The maximum distance from which you can interact with objects.
- **Ghost Layer**: The layer designated for ghost objects.
- **Fixation Layer**: The layer on which objects can be placed.
- **Fixation Distance**: The length of the ray used to check for fixation or ground contact. This value should be small to prevent objects from fixating too far from the wall.
- **Max Overlap Distance**: The maximum distance an object can overlap before the position becomes invalid. Like Fixation Distance, this value should be small enough to allow objects to be placed closer to each other.
- **Default Rotation Angle**: The size of the increment for object rotation. The value should be chosen such that the object can complete a full rotation and return to 0. For optimal results, consider using values like 90, 45, 22.5, 15, 11.25, 9, 7.5, 5, 4.5, 3, and 1.

5.2. Create a new Object

Begin by importing your object model into the scene, then add either a simple collider or a mesh collider with the "Convex" option checked. You may add multiple colliders to your object, either as children or on the root object. However, keep in mind that fewer and simpler colliders will result in faster overlap checks.

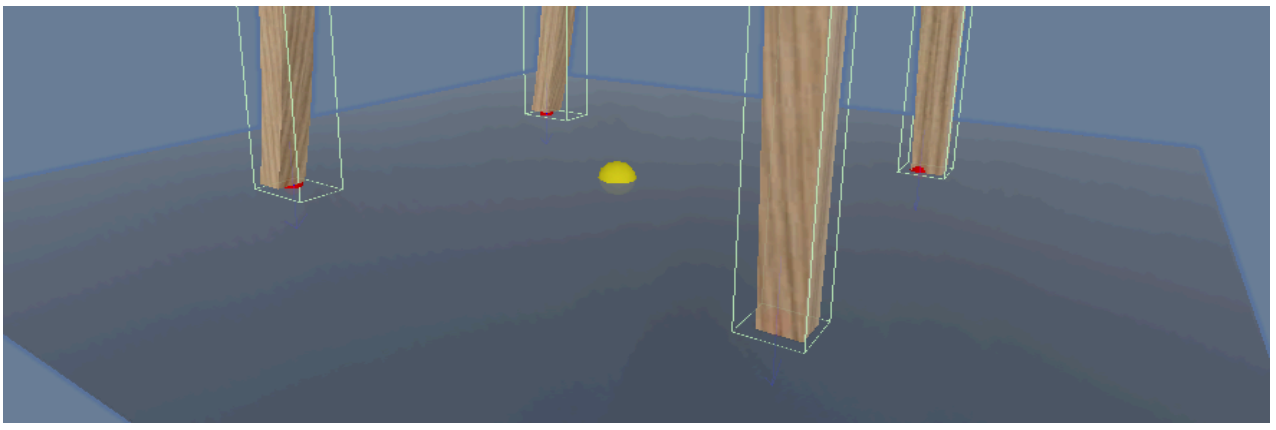


Next, attach the Movable Object script to the root transform.

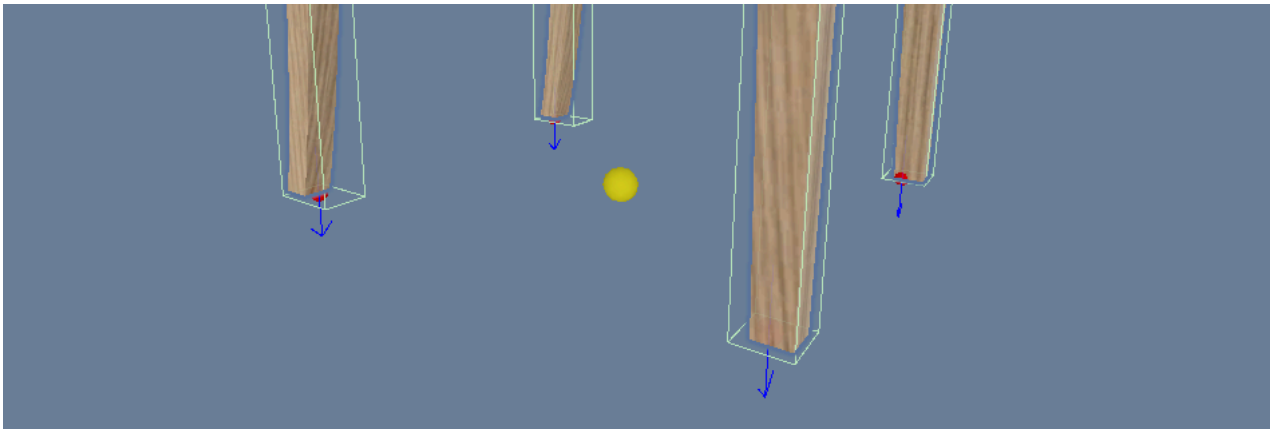


Set the "Is Wall Fixable" variable to true if your object needs to be attached to a wall, allowing it to automatically match the wall's orientation. If set to false, you can manually adjust the object's rotation as needed.

Then, identify the "Pickup Point" for the object. This point, highlighted in yellow, will act as the center for rotation and movement. Make sure it's located on the face that will attach to the ground or ceiling.



The next step is to set up the "Fixation Points". These are the points from which the system will project rays to check if the moving object is properly supported by the ground or other objects.



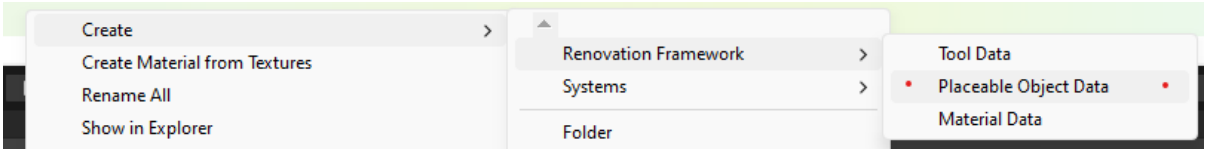
For the object to move and position itself correctly, all fixation points and the "Pickup Point" must be on the same level.

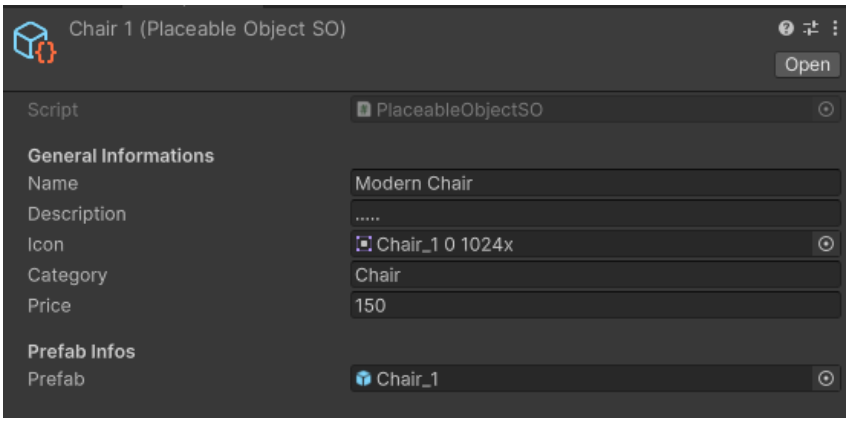
Your object is now ready to interact with the Moving tool.

5.3. Adding an object to the Store

To add an object to the store, first convert it into a prefab.

Next, create the “Placeable Object Data” scriptable object from the create menu. Set the variables such as Name, Description, Icon.

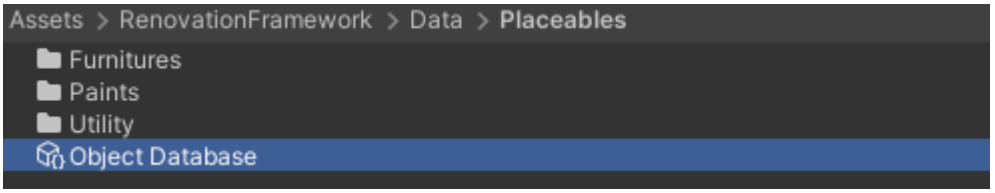




Don't forget to drag and drop your previously created prefab into the “Prefab” field to inform the system which object it will instantiate.

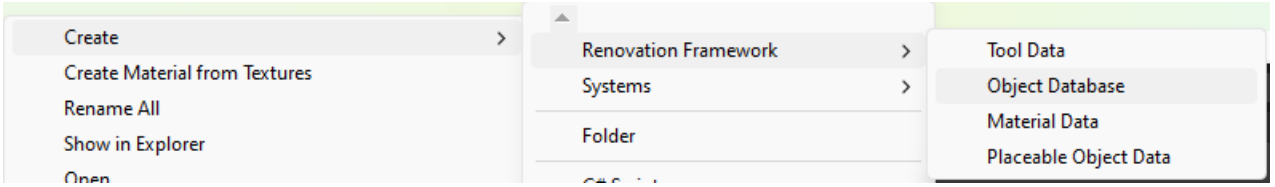
The “Category” field indicates the category under which the object will appear in the store.

Finally, add the Object Data to the list of placeables in the Object Database.

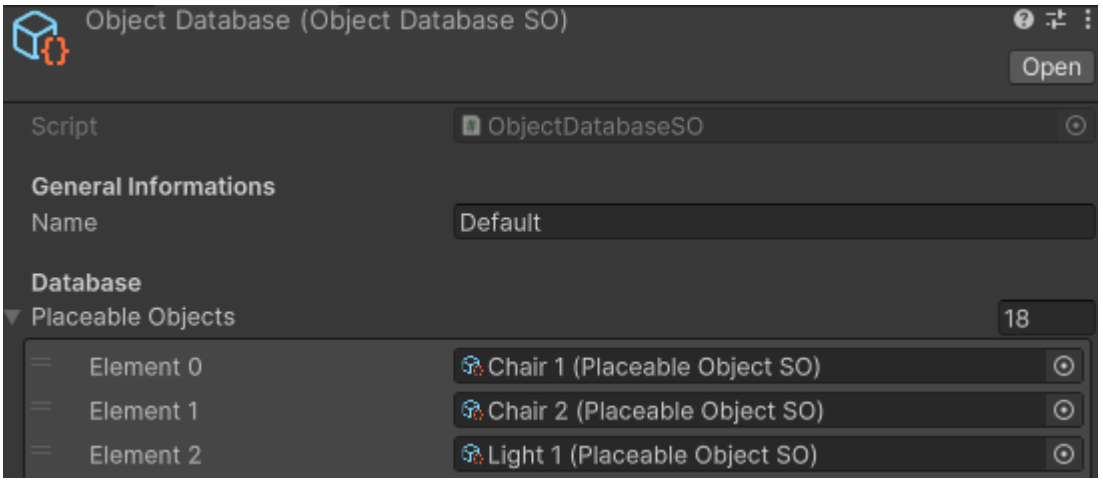


5.4. Creating a new Object Database

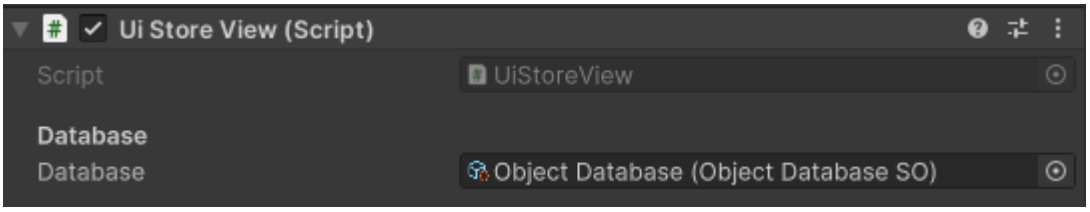
First, create the “Object Database” scriptable object from the create menu.



Next, add the placeable data items to the list.



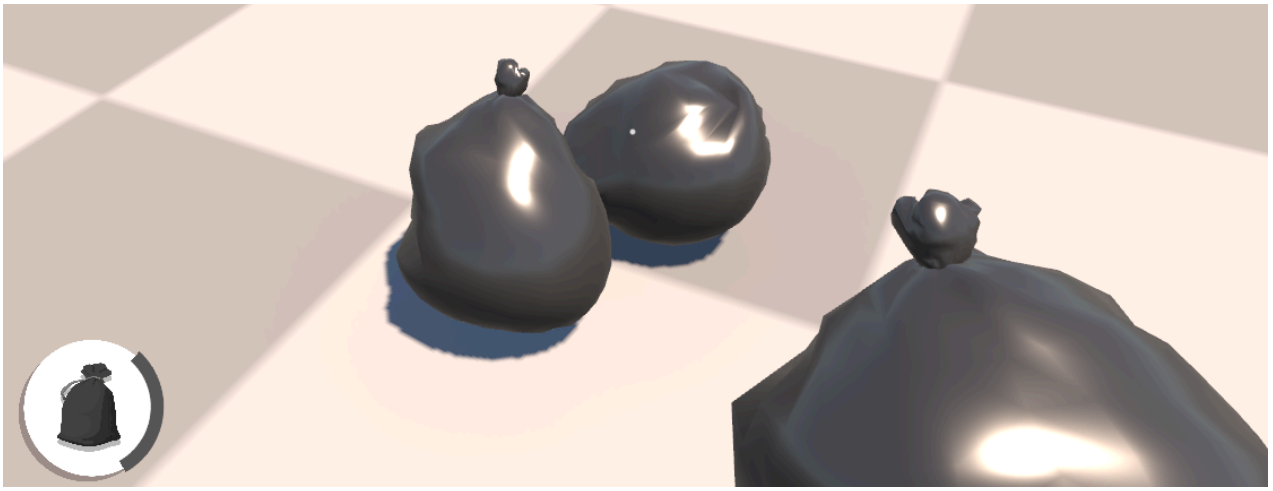
To complete the process, assign this list to the UI Store View Script on the Ui_Store_View.prefab, which is in the “Prefabs/Ui/SubComponents” folder.



6. Trash Collecting Tool

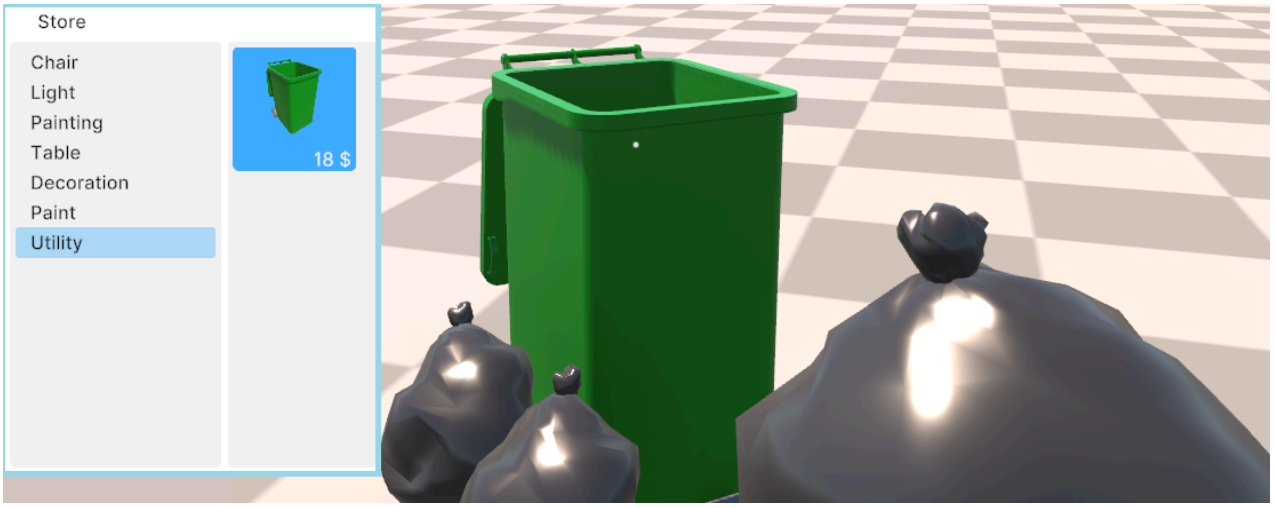
The trash collecting tool enables you to gather trash from the ground into a trash bag. To collect trash, left-click on it while the tool is selected and ensure your trash bag is not full.

The UI indicates how much trash is in the bag. Once the bag is full, it automatically closes, preventing you from adding more trash. You can also manually close the bag at any time by holding down the left click.



Once you have a full trash bag, you can drop it onto the ground with a quick left-click or throw it by holding the left click. You can also pick up a bag from the ground to move or throw it.

To dispose of the trash bags, throw them into a trash bin, which can be acquired from the Store.

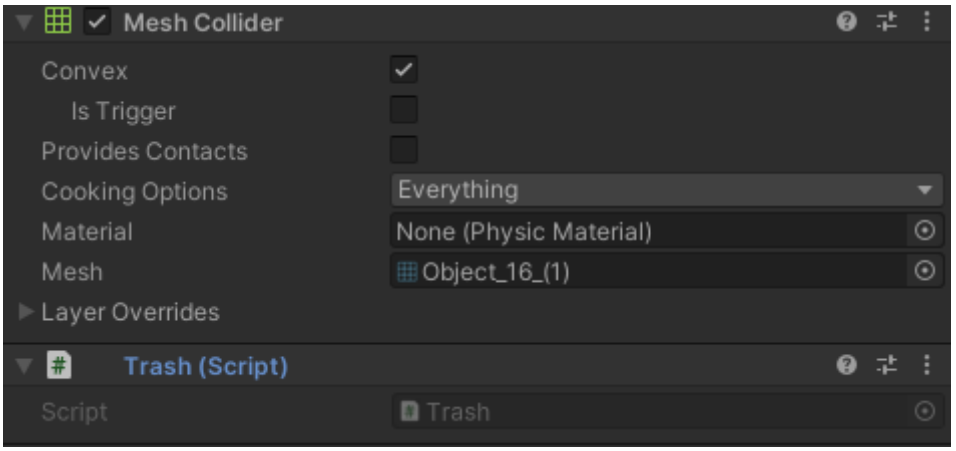


6.1. Settings

- **Trash Bag Prefab**: This is the prefab that will be spawned when you drop or throw a full trash bag.
- **Trash Collection Range**: The maximum distance from which you can interact with trash or trash bags.
- **Max Capacity**: The maximum capacity of a trash bag.
- **Throw Force**: The force value applied to the trash bag when throwing it.
- **Drop Force**: The force value applied to the trash bag when dropping it.

6.2. Creating a Trash object

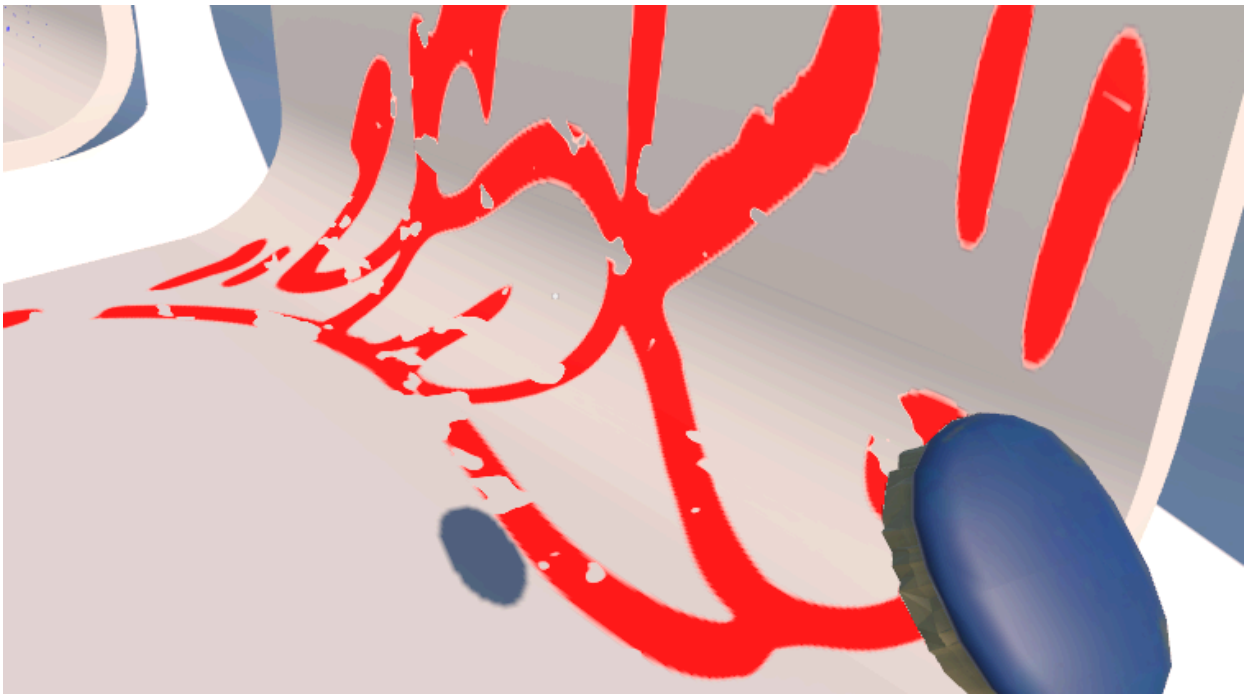
To create a new trash object, attach the Trash script to the object. Also, ensure you add a collider to the object to enable the tool to detect what you are looking at.



7. Cleaning Tool

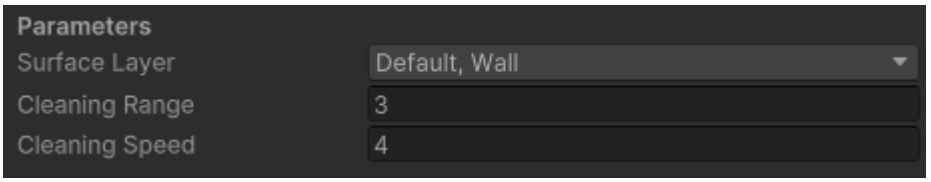
The cleaning tool enables you to remove stains using a brush. To use it, hold the left click while looking at a stain.

Stains are displayed using the Decal Renderer Feature (URP) from Unity, which projects the stain texture onto any surface and shape.



A custom shader is employed to make the stain fade organically during the cleaning process. Once the stain is completely cleaned, the game object is destroyed.

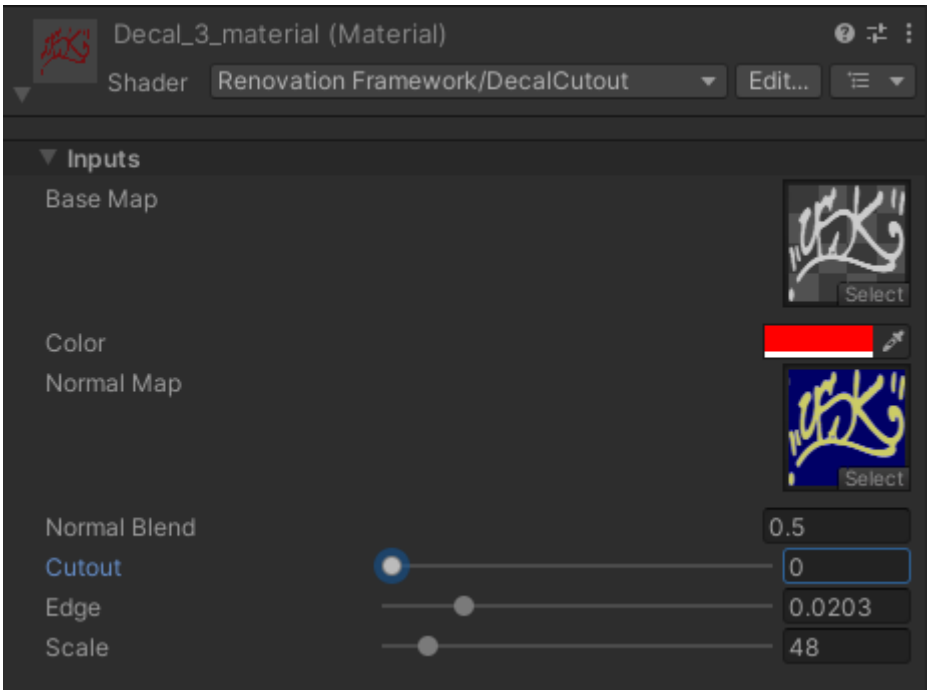
7.1. Settings



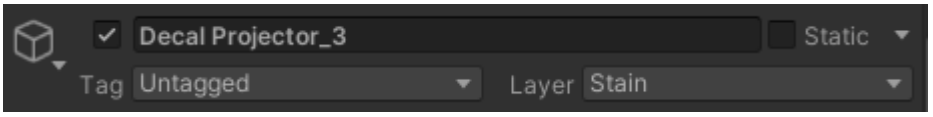
- **Surface Layer**: The layer mask with which the brush will interact, used to prevent the brush from interacting with triggers.
- **Cleaning Range**: The maximum distance from which you can interact with and clean a stain.
- **Cleaning Speed**: The speed of cleaning one unit of a stain. The speed is also determined by the stain's dirtiness in the Stain script.

7.2. Create a Stain

To create a stain, there are two steps. The first is to create a new decal material; for that, you need to create a new material in your project from the Create menu, then assign the Decal Cutout shader from the Renovation Framework and set up your new material to match your needs.

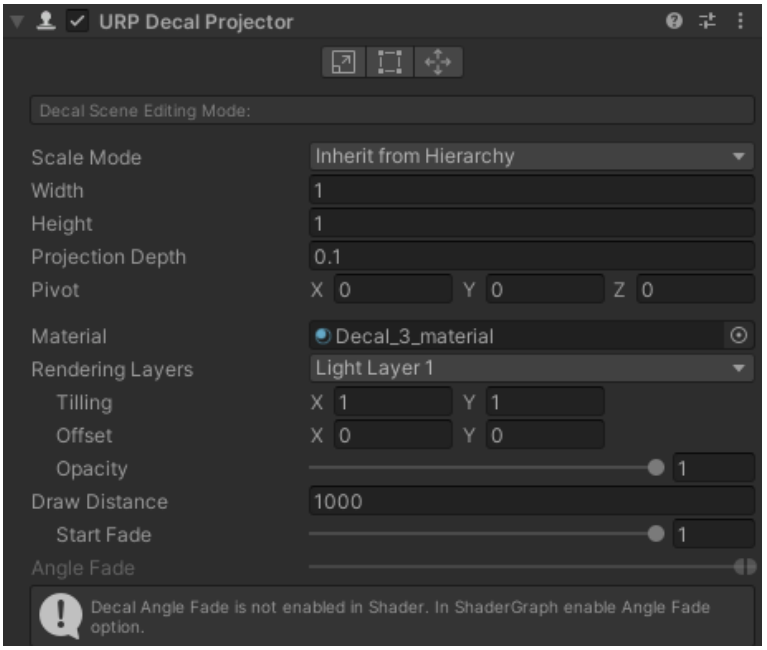


The second step is to set up the prefab of the stain. For that, you need to create an empty Game Object in your scene, then assign it to a layer called Stain.

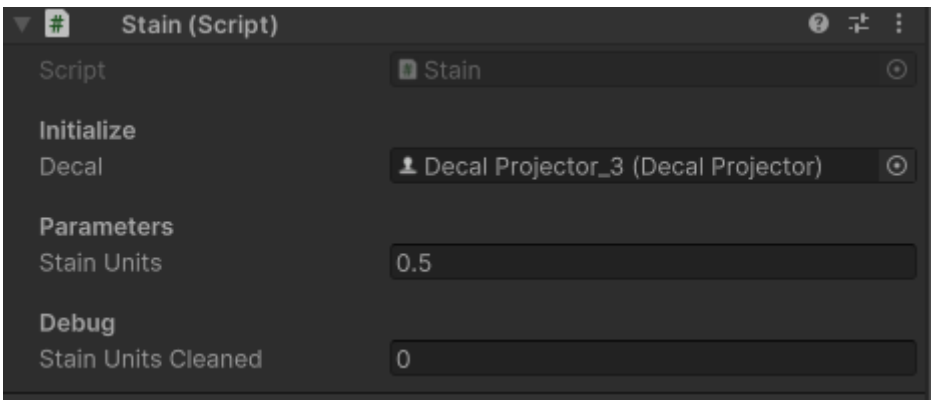


Then, add the following components to the Game Object and configure them as needed:

Add a URP Decal Projector: Assign the material you previously created to the projector and set the Rendering Layers you want to be affected. Don't forget to adjust the projector to ensure it displays correctly.



Add the Stain script: Assign the projector to the Decal variable and set the Stain Units variable, which represents how hard the stain is to clean and, by extension, the time it takes to clean.



Add a Box Collider: Set the Size.z of the collider to match the projection depth of the projector, and make sure that Is Trigger is checked.

To finish, convert the Game Object into a prefab to use it in your scene.