Getting started with Ultimate Grid Inventory

First of all, we want to thank you for buying one of our assets! This is the **Ultimate Grid Inventory** documentation!

- Initializing the Ultimate Grid Inventory

Features

We are managing to develop more and more features for UGI, but this is what we have until now:

Grid Inventory

Grid Inventory where you can build your own grid inventory and use to insert items or even create containers.

Holders

Holders where you can store **containers** or **equippable items** like *weapons*, *helmets*, *pockets items*, *granade* and etc... Now we have two type of holders

- Item Holder used to store basic items, like *weapons*, *helmets* and etc...
- Container Holder used to store container or in other words, store another inventory like a *backpack* for example.

Right Click Options

We provide a Right Click Options where you can add how many options as you need and also have **different types of options** for **different types of items**

- How to create an item option and attribute to the item you need.

On Hover Insert

When drag an item hover a container item, you can drop the item then will be *stored the container* that you hovered.

Systems

Input Handler

In order to **support** both *input systems* that Unity has, we create an abstraction from the *input system*, so you configure your inputs in the **Inventory Input Handler** and that's it!1

Draggable System

You can configure the draggable system the way you want! Now we have a few type of configuration, but we're looking forward to add more ways of *draggable system*.

Basic Highlight System

When dragging an item on top of a grid, you will be able to see a highlight if the item fits, out of bounds or missing space... You can configure the colors from the highlight.

Space Highlight System

This feature is going to show a different color when you drag a item over a container item in the grid, to have a better visualization if you can insert or not. Follow the example, when you drag a item **over** a **container item**, it will show a color that you set in the *InventorySettingsSo* meaning that you can insert the item inside the container hovering. Learn how configure the Player Space Highlight System

Scroll System

When drag a item closer to a border in a scroll area, it will auto scroll the direction that you are closer... It goes **up** and **down**

- Configuring your Scroll System Settings

Audio System

We develop an **audio system** for the *Ultimate Grid Inventory*, It have a few events that you can set right now On Being Place and On Being Picked. The audio system goes beyond the items, it **can be configured for options** either. When execute an option will play the audio you set.

Images from Ultimate Grid Inventory

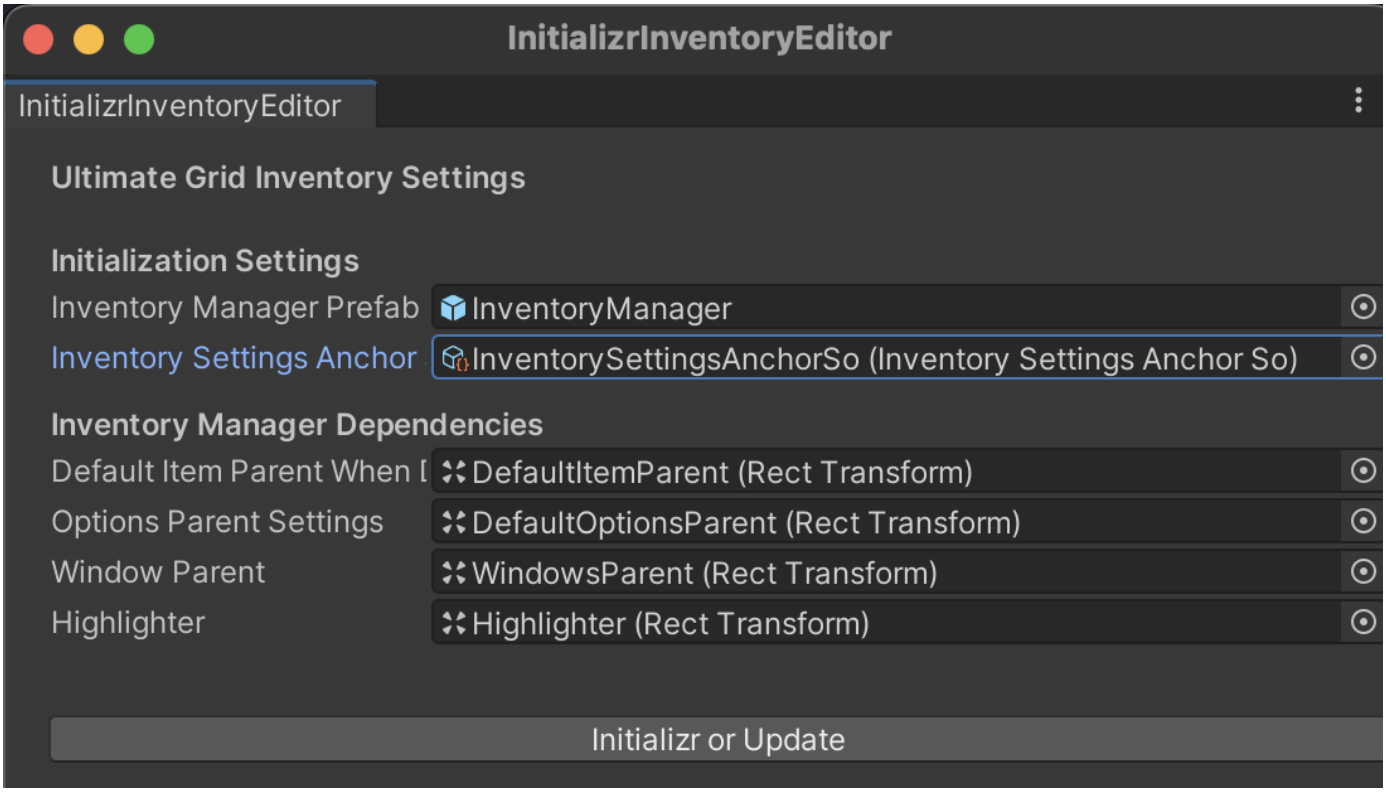

BOX
Food Box
Basic Helmet
Full Mask
Cante…
M4A1 M204
Kamorov
Explo…
Backpack Sma…
Basic Helmet
Glasses
Mask
Batte…
Cante…
AK48
Tape
MP5
Basic Helmet…
Machete
Backpack Sma…
Full Mask
Delivery Box
Smoke…
Machete
M4A1
Helmet Spec0…
AK48
Spar-16
Chest
Backpack Large
Machete
Kamorov
Explo…
Explo…
Wallet
Interact Environment Prop
Interact Environment Prop
Help (Tutorial)
CHEST
Cante…
Smoke…
M9
Smoke…
Range Finder
Mask
BACKPACK LARGE
AWM
Groza
Backpack Sma…


Glasses
Delivery Box
M4A1
Groza
Basic Helmet
Backpack Sma…
AKM
Spar-16
Backpac
Interact Envir
Interact Envir
Help (Tut

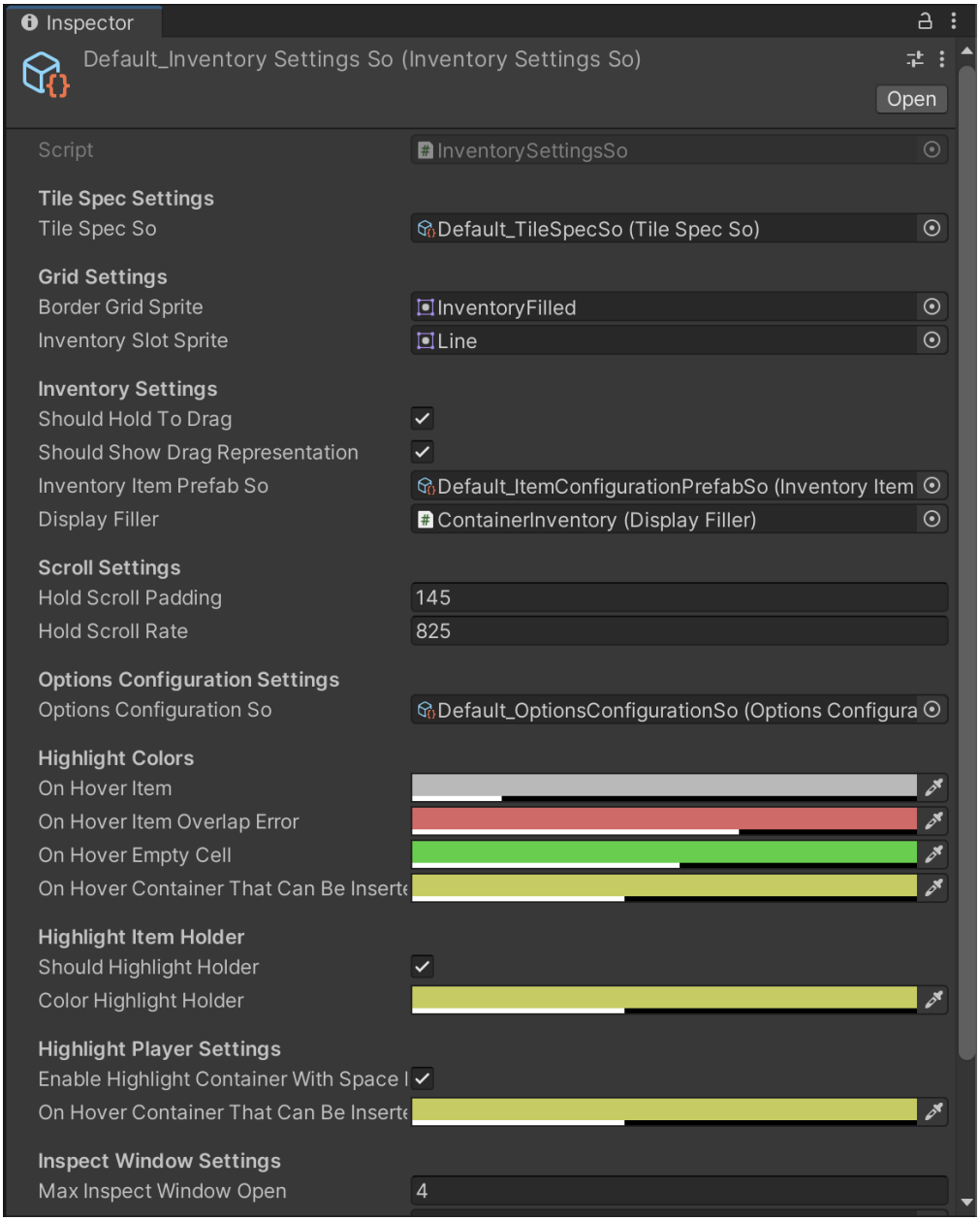
Initial steps for configuration

In order to configure the Inventory we can use a tool from the UGI, you can found in the **Menu** from Unity Editor a tab called Ultimate Grid Inventory > Initializr Inventory

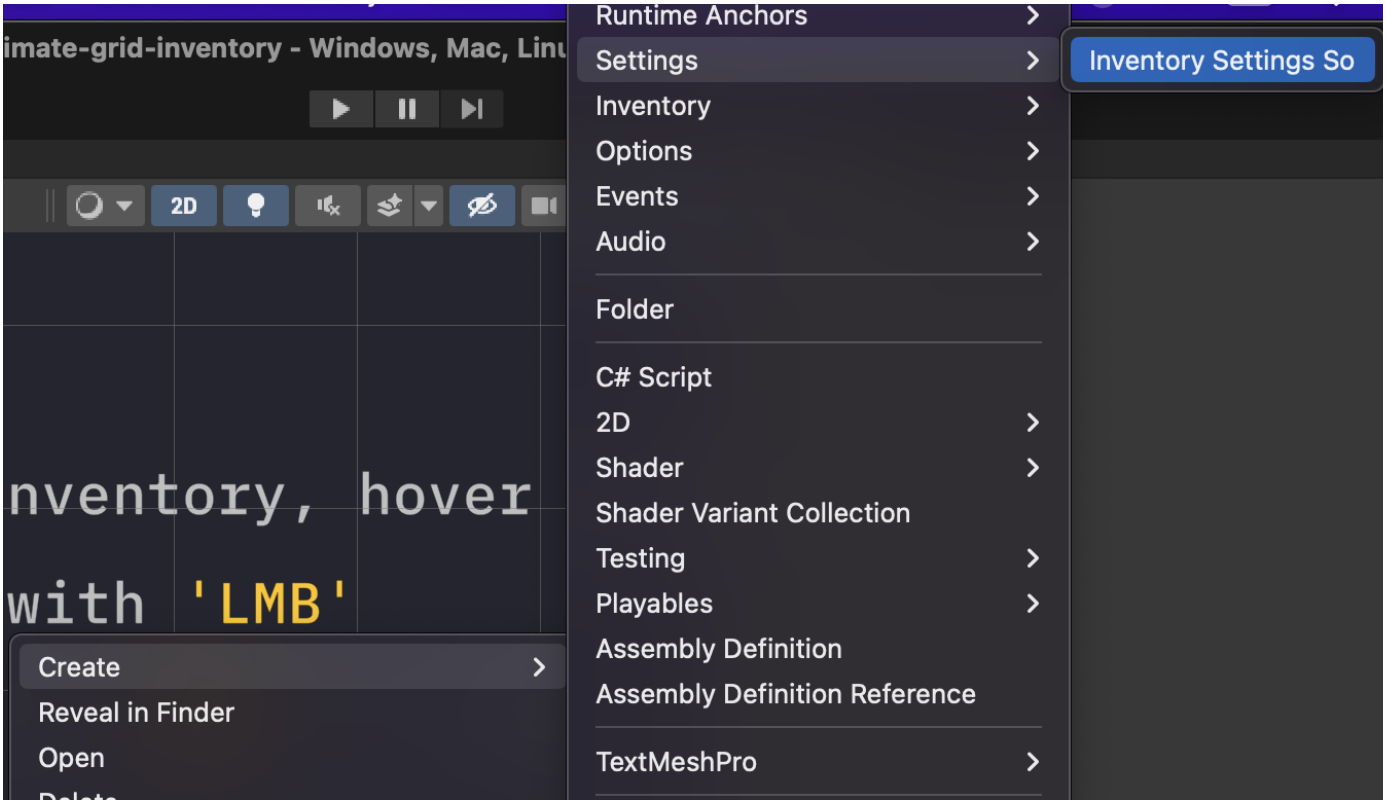
Will open the modal to configure the *initilization* from Inventory



We try to group all the important configuration in a one single Scriptable Object. Inside the **Inventory Settings Anchor So**, you will find a scriptable object called Inventory Settings So. Will be all the configuration from the Ultimate Grid Inventory:



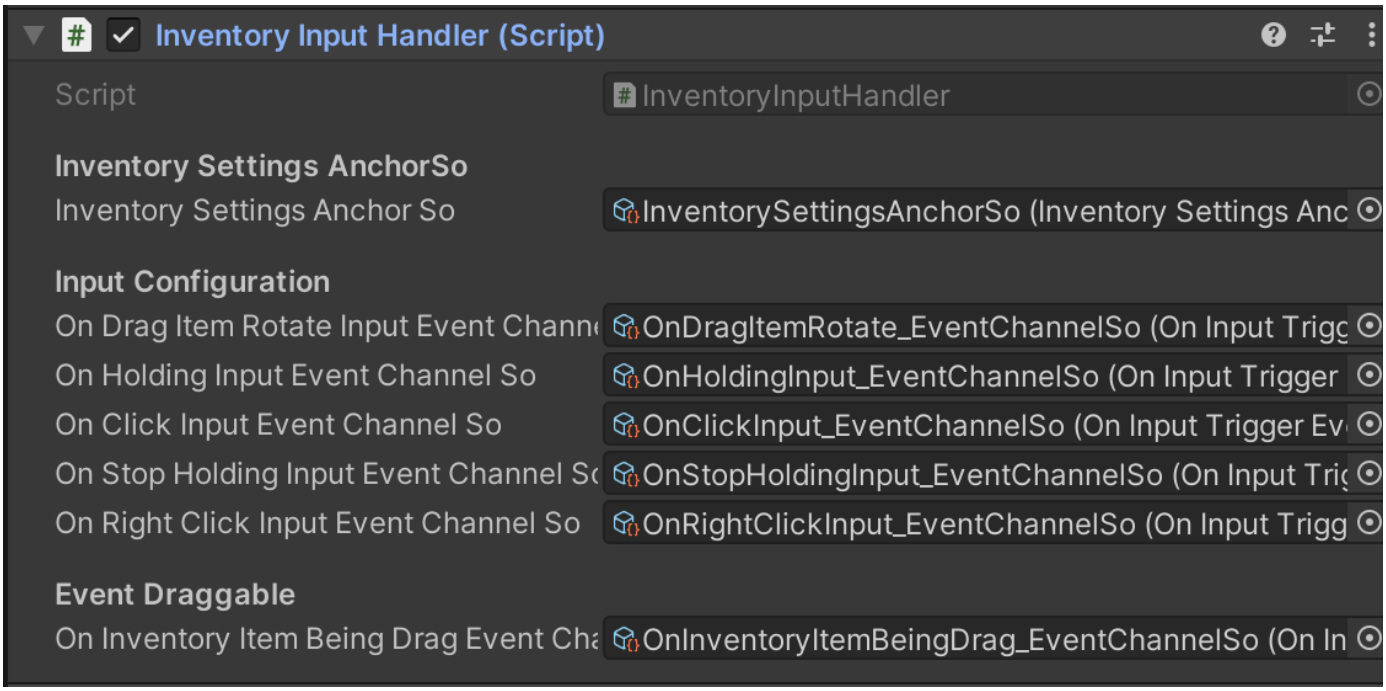
In case you want to create new configuration for Inventory Settings So, you can use the default creation file from unity:



And then configure the way you like it! Also you can change the Inventory Settings Anchor So at runtime, so if your game needs to change some colors of Inventory, will work! Only excluding properties like Tile Spec Settings and Grid Settings, this may cause some bugs in case altered in runtime

Configure the Inputs for the Ultimate Grid Inventory

In order to configure the Inputs, you need to access the Script called **Inventory Input Handler**.



Inside this class you will see where the *Ultimate Grid Inventory* handles the inputs, this will be where you're going to alter in case you want to use the ***New Input System***.

Events

We handle the inputs by triggering events, so that's why it make easy to use wherever input system you like. Also if you want to change some keys you just access one file and make the change, in case is the old input system.

Altering keys

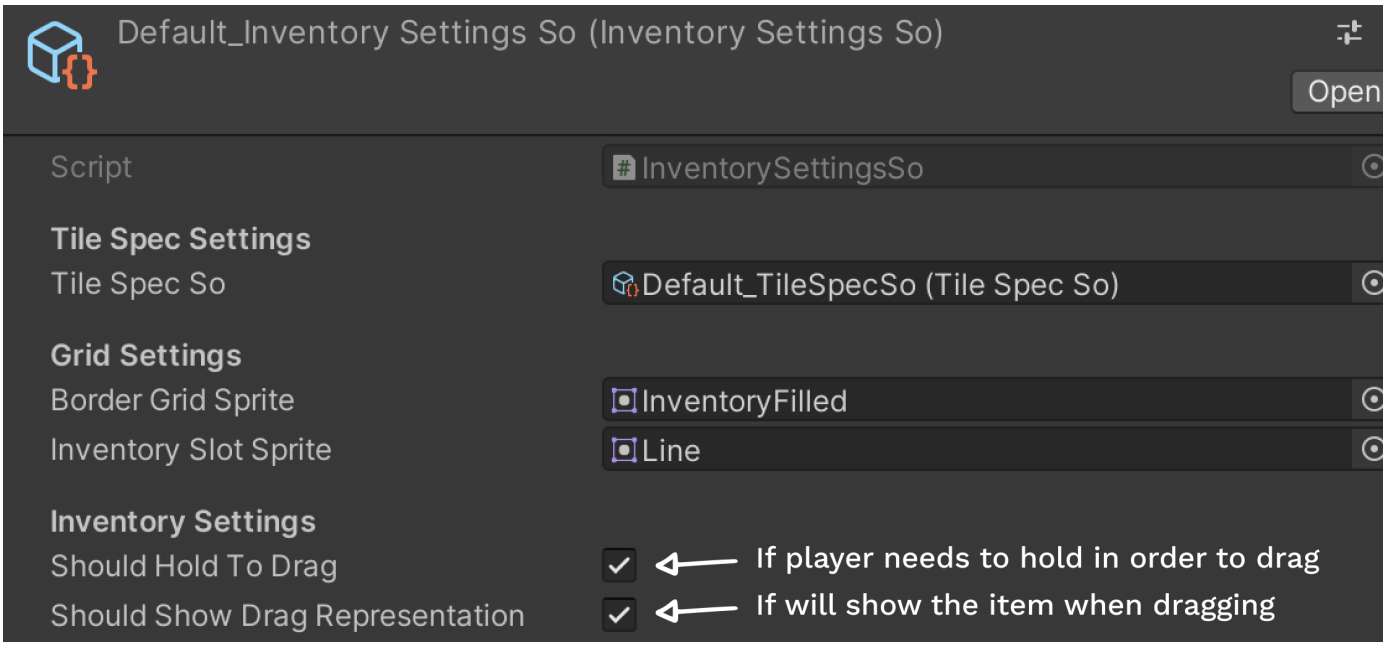
In case you're using the ***Old Input System***, you can just enter the file and change the inputs:

```
Event function  rubenskj +1
void Update()
{
    UpdateMousePosition(Input.mousePosition);

    if (Input.GetKeyDown(KeyCode.R))
    {
        onDragItemRotateInputEventChannelSo.RaiseEvent();
    }

    if (Input.GetMouseButtonDown(1))
    {
        onRightClickInputEventChannelSo.RaiseEvent();
    }

    if (GetHoldToDrag())
    {
        HandleHoldingInputs();
        return;
    }

    HandleInputs();
}
```

Also check the method **HandleHoldingInputs** and **HandleInputs** to change keys

How to change Draggable System for a better fit

In the current state, we have a few options to change the *draggable system*.

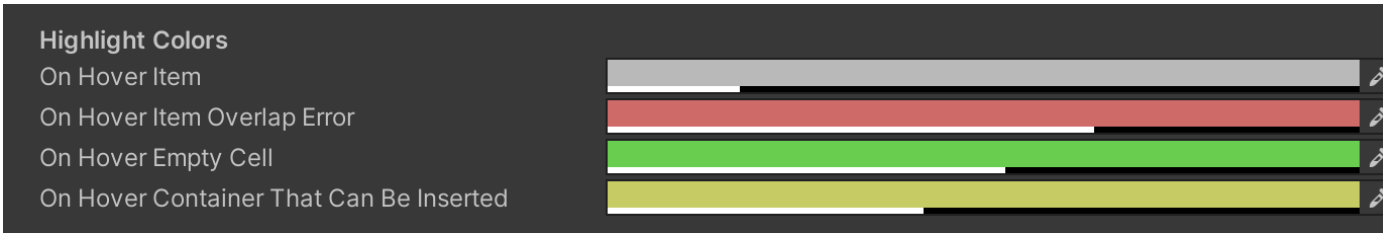


Options

- Hold Item to Drag if this is set to off, player will **not** need to **hold** the mouse button in order to move the item.
- Show Drag Representantion when the player is *dragging* an item, if this is **on**, will show a representation from this item below the mouse. Otherwise will not

Basic Highlight

The **basic highlight** is the name of the feature of when you are *dragging* a item over a grid you can see behind the [item representation] some colors showing if the item can be placed, cannot be fitted or out of bounds.



Understanding the colors

- On Hover Item when the player hover the cursor over an item in the grid.
- On Hover Item Overlap Error when the player hover a dragging item over another item or it's out of bounds from the grid.
- On Hover Empty Cell this means that the dragging item can be placed in the location is being hovered.
- On Hover Container That Can Be Inserted when hovering a container with a dragging item, if appear this color, means that the player can drop the item and then the item will be stored inside the container.

Player Space Highlight System

Player Space Highlight is *highlighted* when the player is dragging an item and you have a container in one of your Container Holder and also a container inside of this **holder**, this **container** will be **highlighted**

Changing the color

In order to change the color from Player Space Highlight, you can alter in the Inventory Settings So

Disabling this feature

In case you want to disable the feature, you can just disable set to off the Enable Highlight Container With Space In Player Inventory in Inventory Settings So, this will **prevent** to appear the *highlight* when *dragging* an item and it fits inside another container in the Player Inventory So

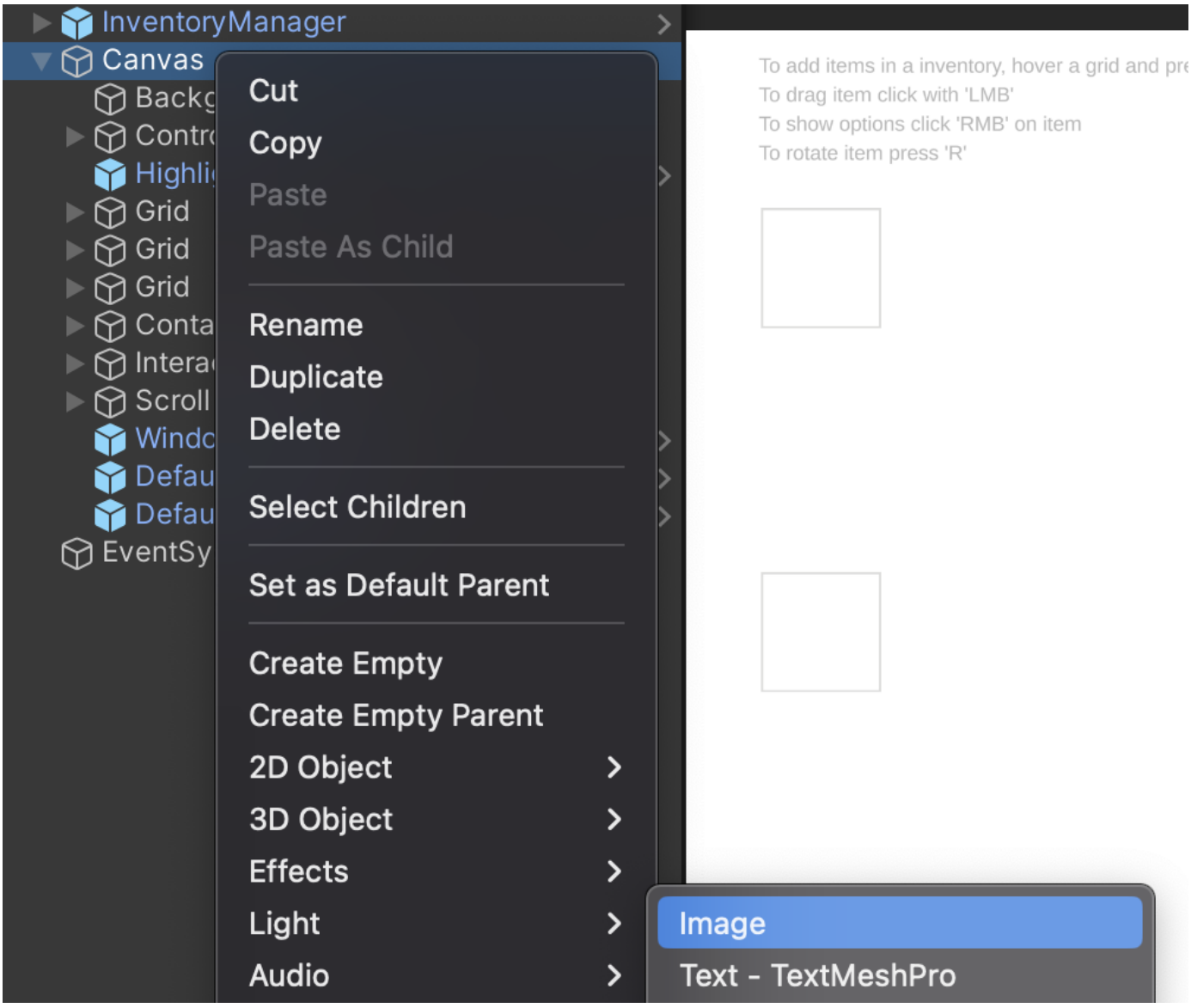
Learn how grids works and how to build

Grids are the most important thing in the **Ultimate Grid Inventory**, is where you're going to place and pick your items. The entire system is based on those grids.

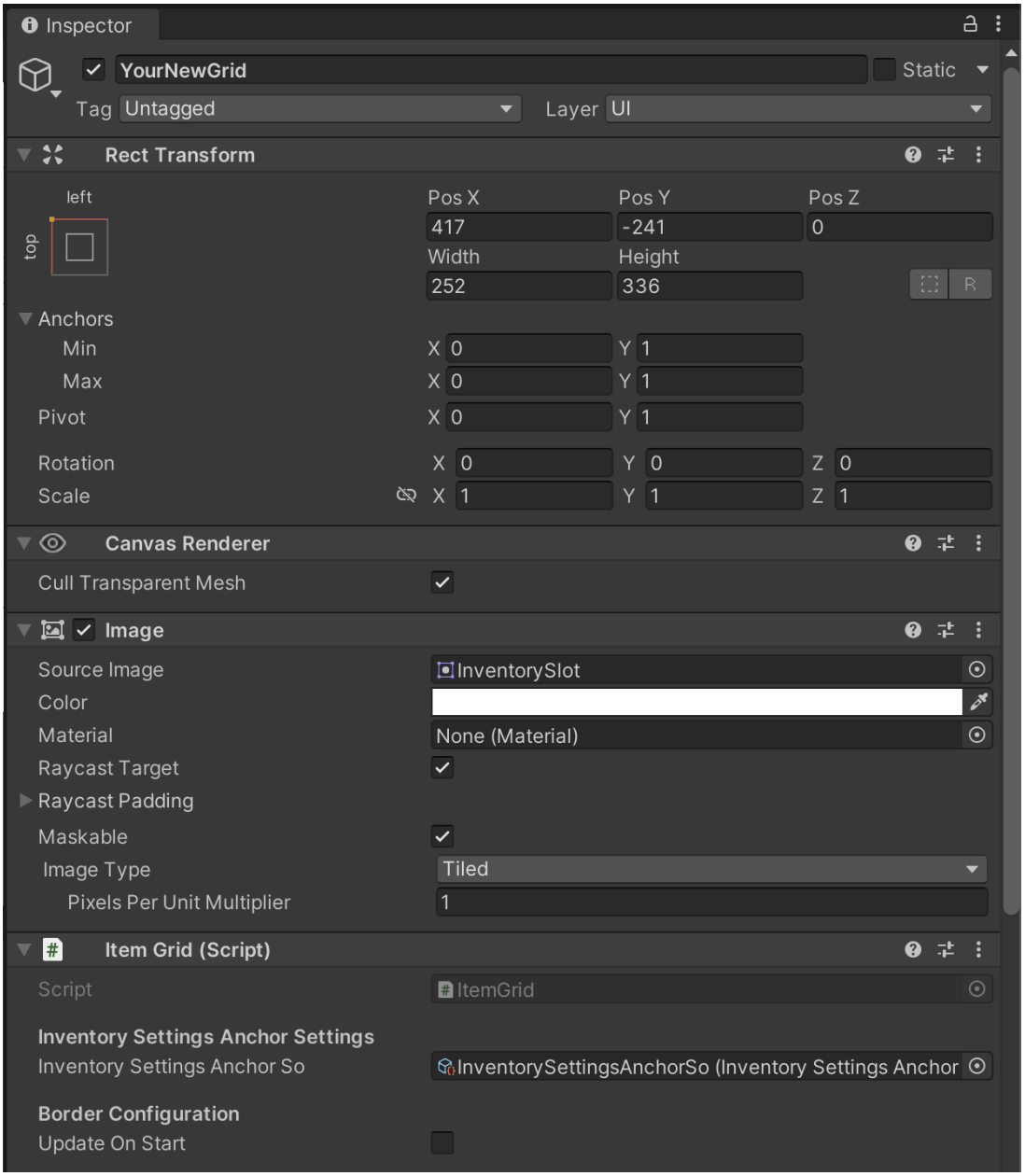
Simple grids you're going to use mostly for testing your items or in case you're create a non dynamic inventory for your game.

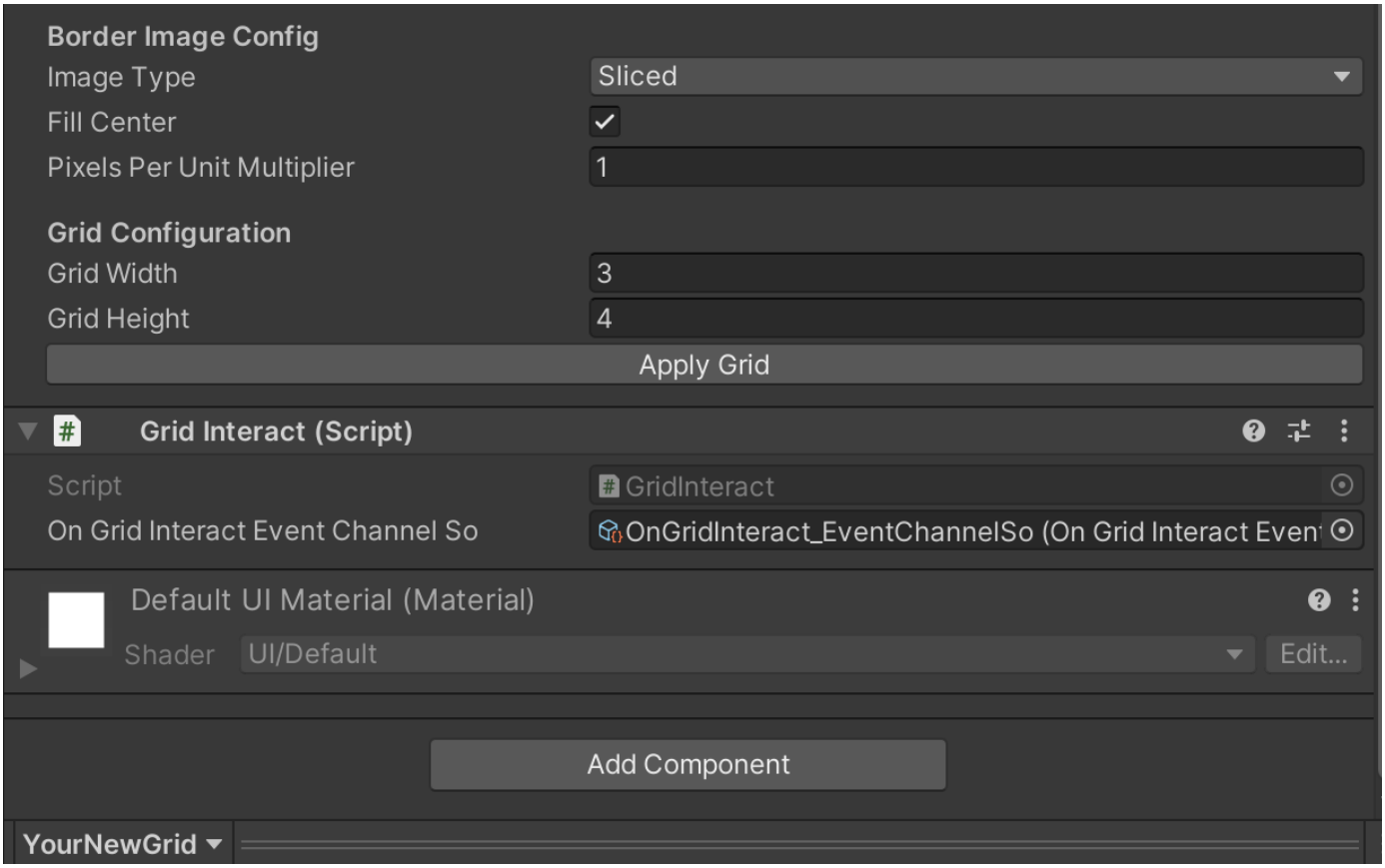
Creating a simple grid

First of all, we need to create an **Image Component** and then **attach** a few **components** to it. You can follow the images below:



After creating the **Image Component** you can attach two components, Item Grid and Grid Interact. Your Game Object should be like this:


Inspector
YourNewGrid
Static
Tag Untagged
Layer UI
Rect Transform
left
top
Pos X
Pos Y
Pos Z
417
-241
0
Width
Height
252
336
Anchors
Min X 0 Y 1
Max X 0 Y 1
Pivot X 0 Y 1
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
Cull Transparent Mesh
Image
Source Image InventorySlot
Color
Material None (Material)
Raycast Target
Raycast Padding
Maskable
Image Type Tiled
Pixels Per Unit Multiplier 1
Item Grid (Script)
Script ItemGrid
Inventory Settings Anchor Settings
Inventory Settings Anchor So InventorySettingsAnchorSo (Inventory Settings Anchor
Border Configuration
Update On Start



Ps: Those images are from an already configured **Grid**, but this to show the fields

Now you can configure some properties

Image

- Source Image the grid slots image.
- Image Type because this represents the **Slot** from the **grid**, this must be and **Tiled** type. But up to you if you dicide another is better.

Item Grid

Inventory Anchor Settings

- Inventory Settings Ancho So set the scriptable object that anchors your **Inventory Settings**.

Border Configuration

- Update On Start this will update grid configuration when the **game starts**, the same as clicking the button Apply Grid.

Border Image Config

- Image Type the type of the Image used for the grid. Recommended to be **Sliced**
- Fill Center will set the property **fill center** from image. But will whether or not to render the center of a Tiled or Sliced image.
- Pixels Per Unit Multiplier In case you want to multiplier the sliced image.

Grid Configuration

Here comes the grid configuration, where you can set the width and height from the grid.

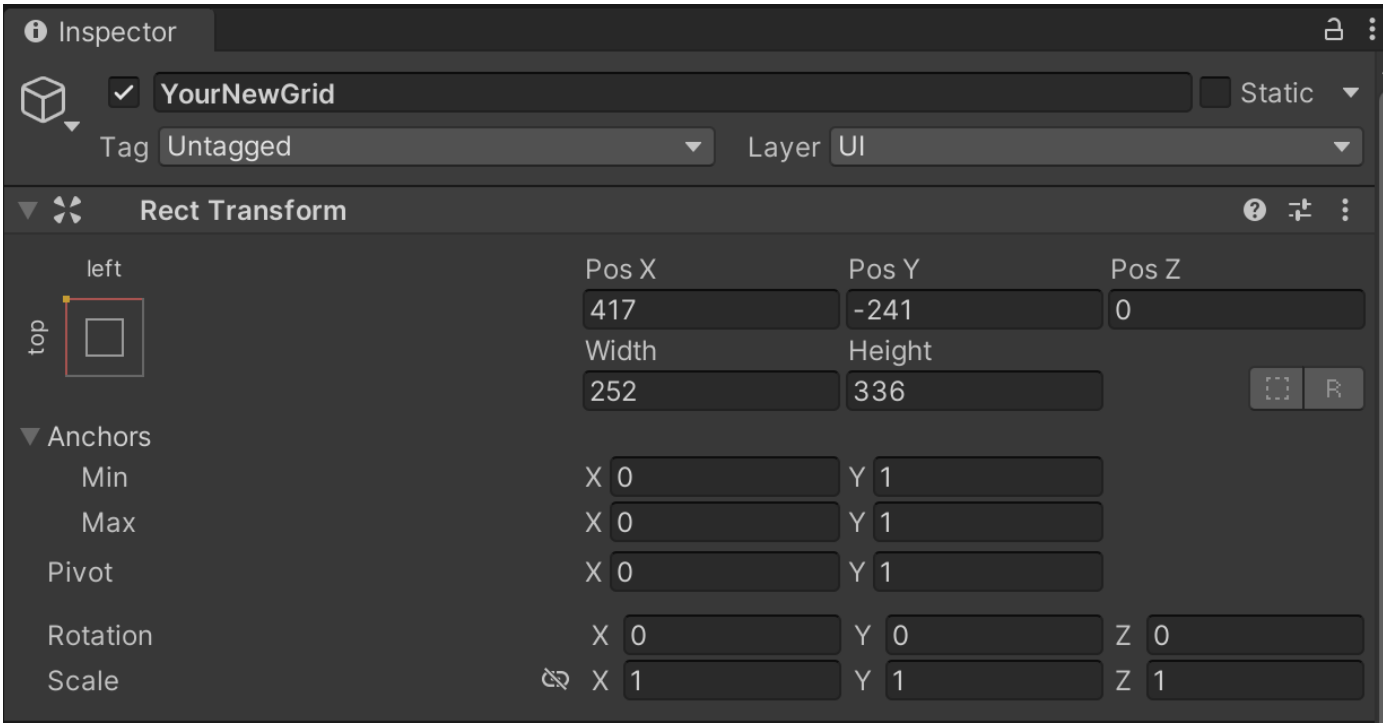
- Grid Width the width from the grid.
- Grid Height the height from the grid.

Grid Interact

Here you can set the default scriptable object that handles when the player interact with a grid.

- On Grid Interact Event Channel So the event that handles when the player interact with the current grid.

Ps: Remember to always set the **anchors** from your **Grid** to the **top left** of the container that is in.



With all this configuration you must have some grid like this:

Ps: the grid contains some items.

You always need to test the grid in runtime just to make sure everything is running good. If not, it might be a problem with the configuration.

Coming up

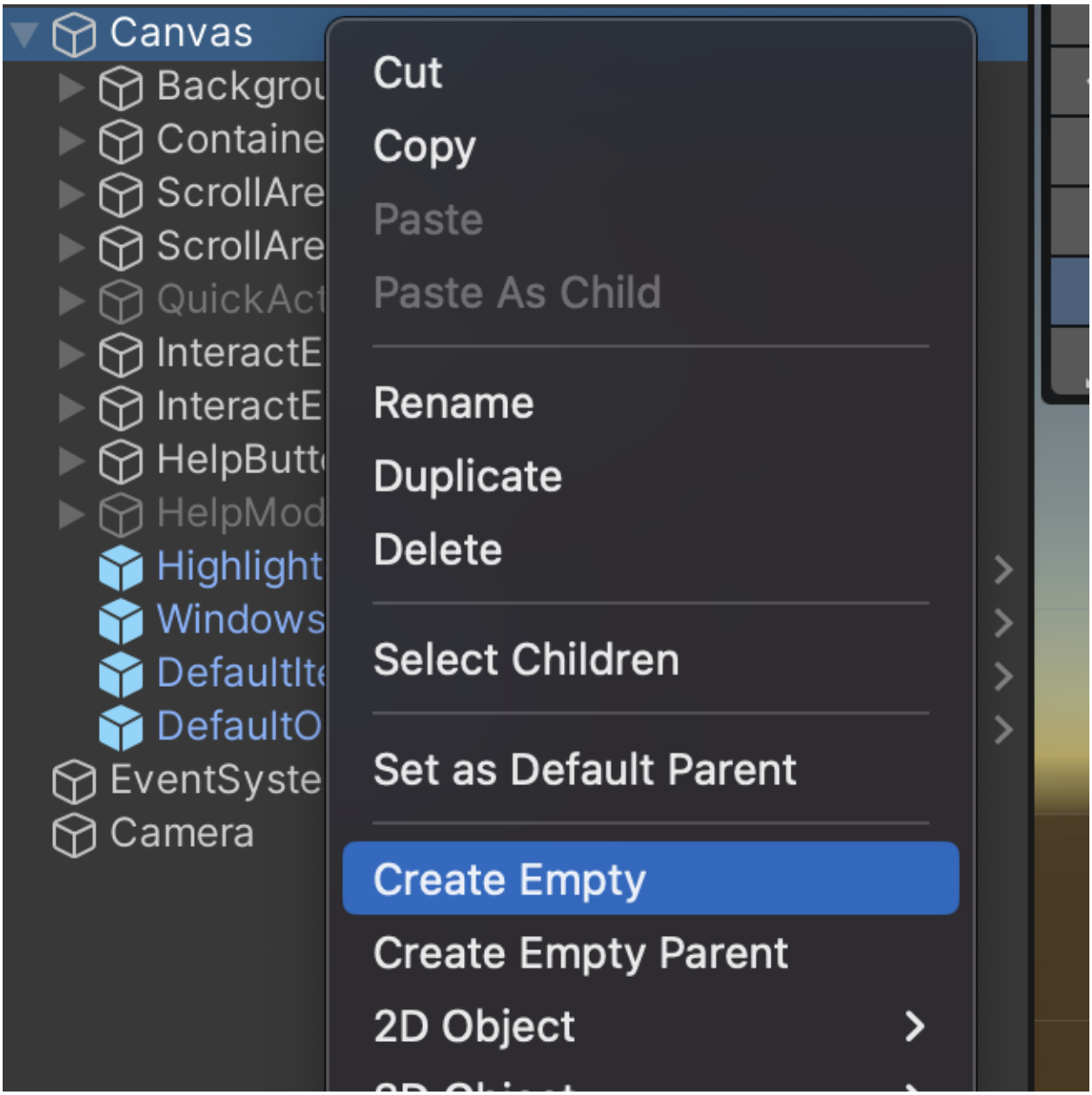
The **Item Grid** in going to be refactored in the next versions. But without prevision to a specific version

Understand how Container Grids works and how to build it

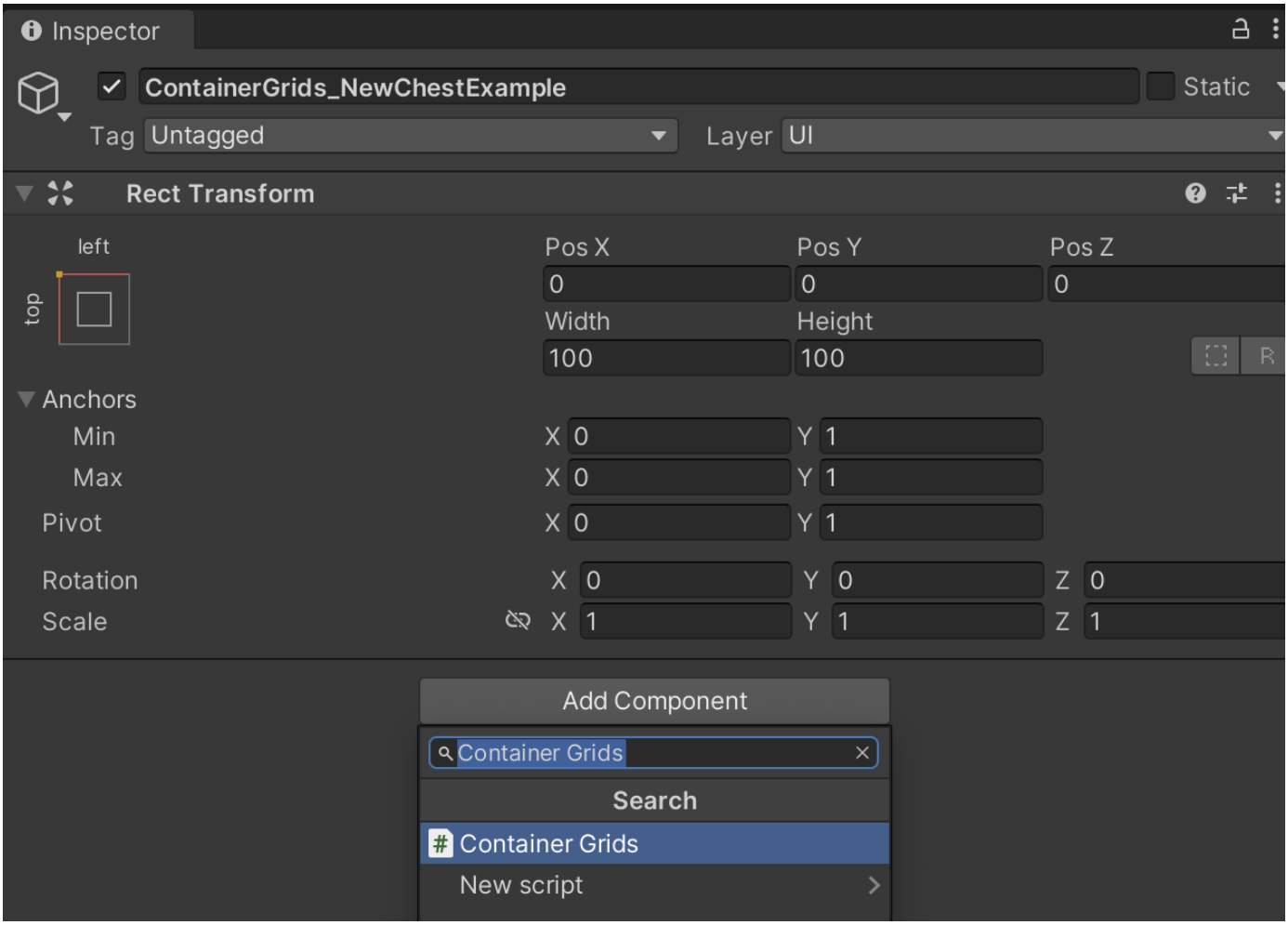
The **Container Grids** are similar to grids but they are wrapped by Container Grid script, and the are used as a prefab for the Inventory Grid for containers items. Like the Backpack, Chest...

Learn how to create an Container Grids and use as a prefab

In order to create the **Container Grids** you can create a **Empty Game Object** inside a **Canvas**

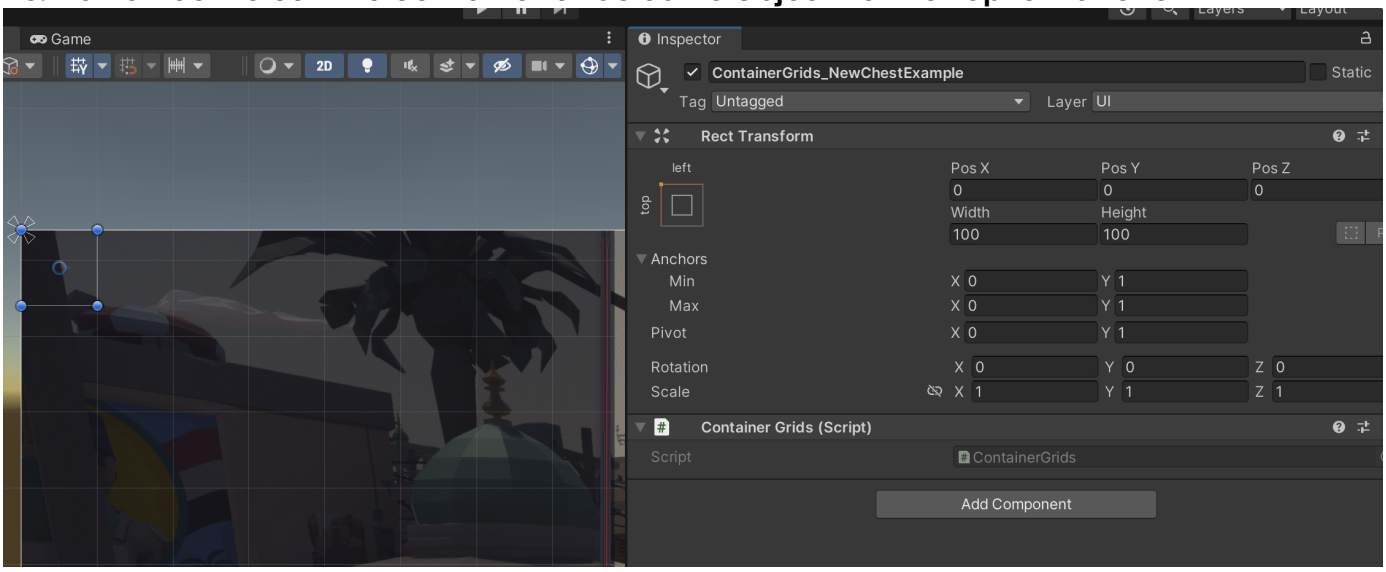


Then you can attach to this **Game Object** the script called **Container Grids**



Inside this **Container Grids Game Object** you can create your grids and decide the way is going to look.

Ps: Remember to set the **Container Grids Game Object** to the **top left anchor**

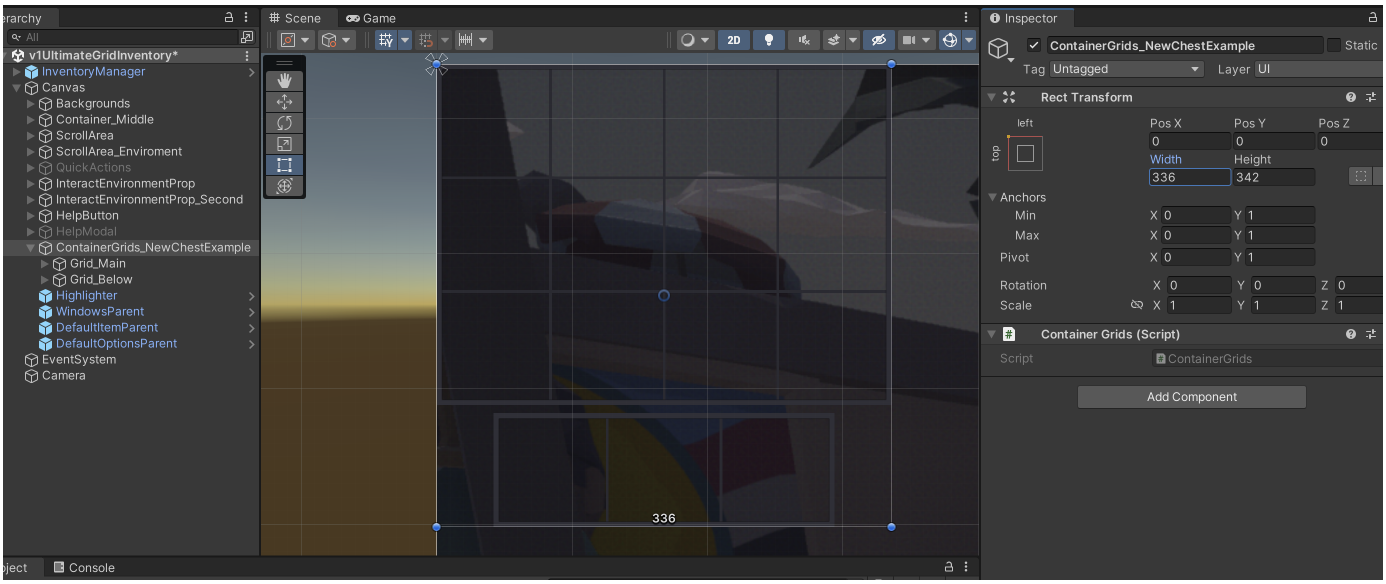


You need to do this because the Container Grids will be used to be displayed in some areas and setting this to the top left will prevent that the UI looks strange and buggy.

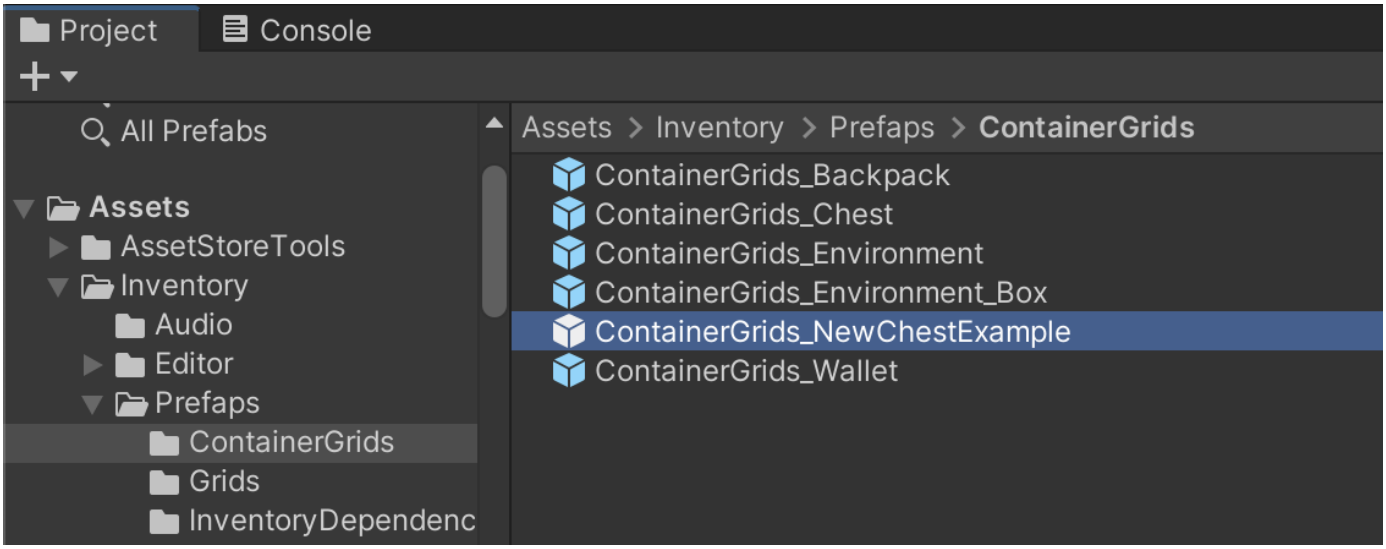
Creating grids inside the Container Grids

In order to create Grids you can follow the tutorial from [how to create a simple grid](), but you will need to create inside of the **Container Grids Game Object**

Then after creating all the grids you need, make sure to adjust the width and height from the main game object (The container grids game object)



After creating the **Container Grids** and also the grids inside of it, you can now **set** this **game object** as a **prefab**



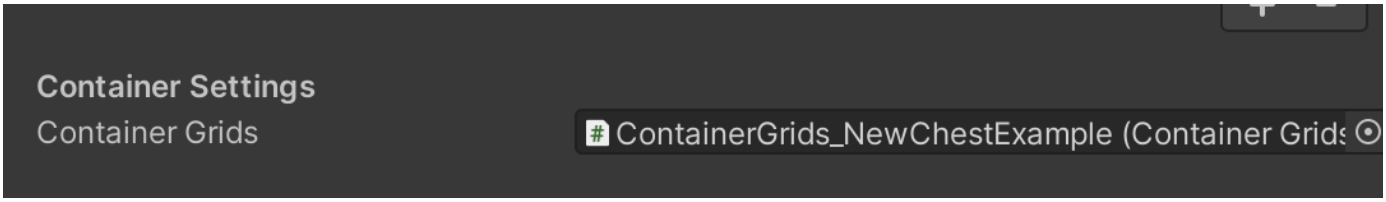
Ps: After creating the **Container Grids Game Object** as prefab, you can **delete** it from the **hierarchy**

Setting the Container Grids in a Container Item

After creating the Container Grids, you just need to set the item that you wish, to do it so, follow the steps below.

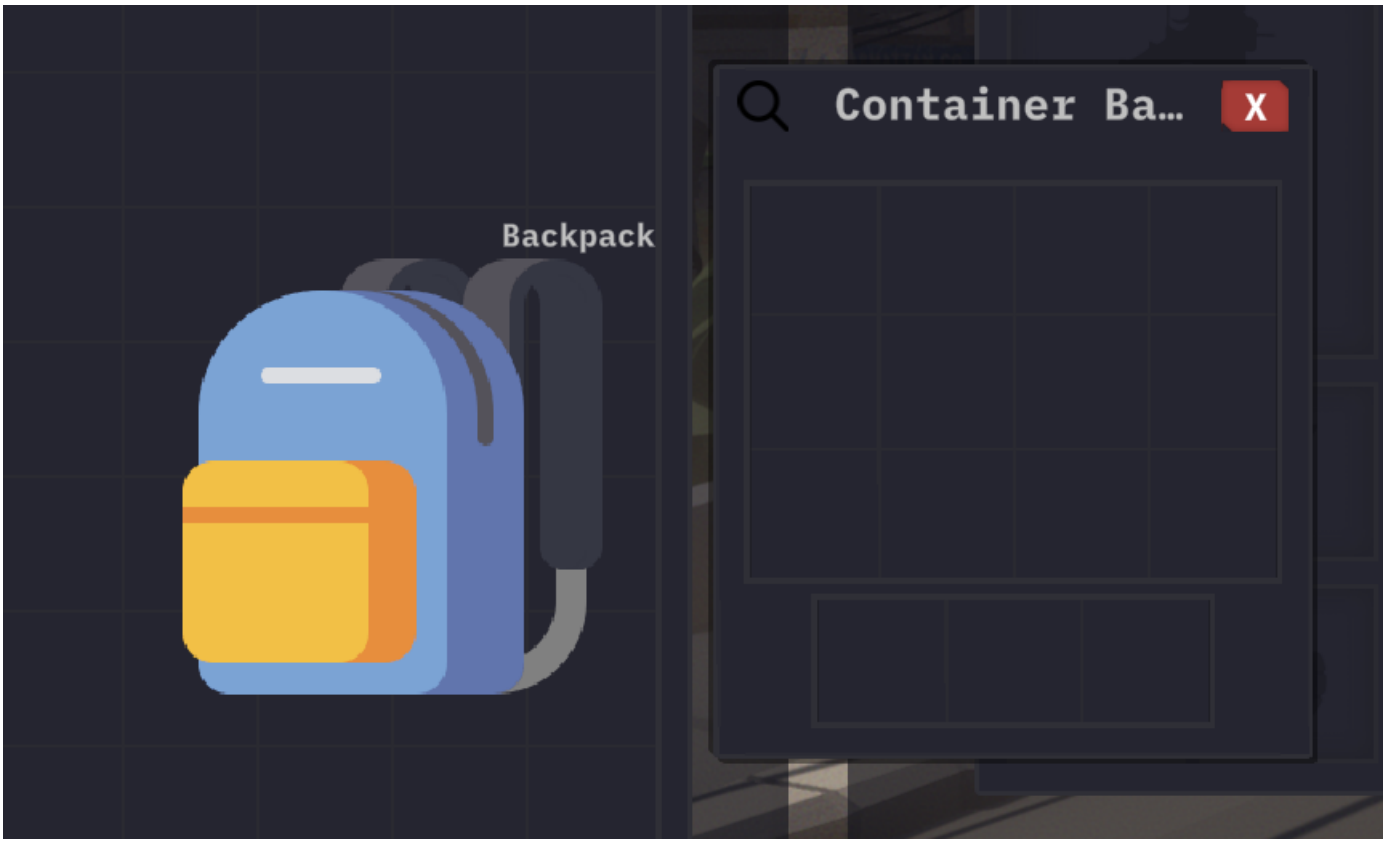
In case you don't have a **Container Item** created, you can follow the tutorial in [Creating a Container](#).

The property Container Grids in the ItemContainerDataSo you can set with the prefab that you created.



Then just hit play and you will be able to see the Container Grids you created when Open the container in the inventory.

Final result



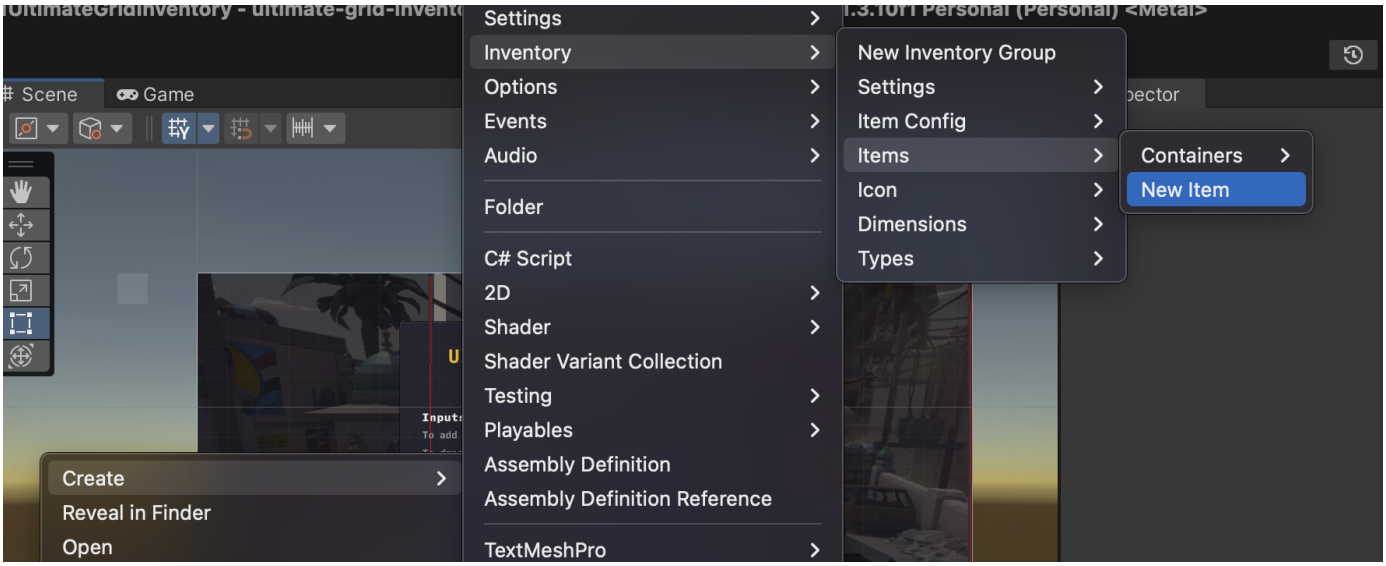
Ps: We're looking forward to **improve** this creation system in order to be easy to create new **Container Grids**. Wait for the **next versions**!

Learn how to create an Item for your Inventory

Item Data So is a scriptable object that represents the item in the inventory. You might want to check the *Inventory Item*.

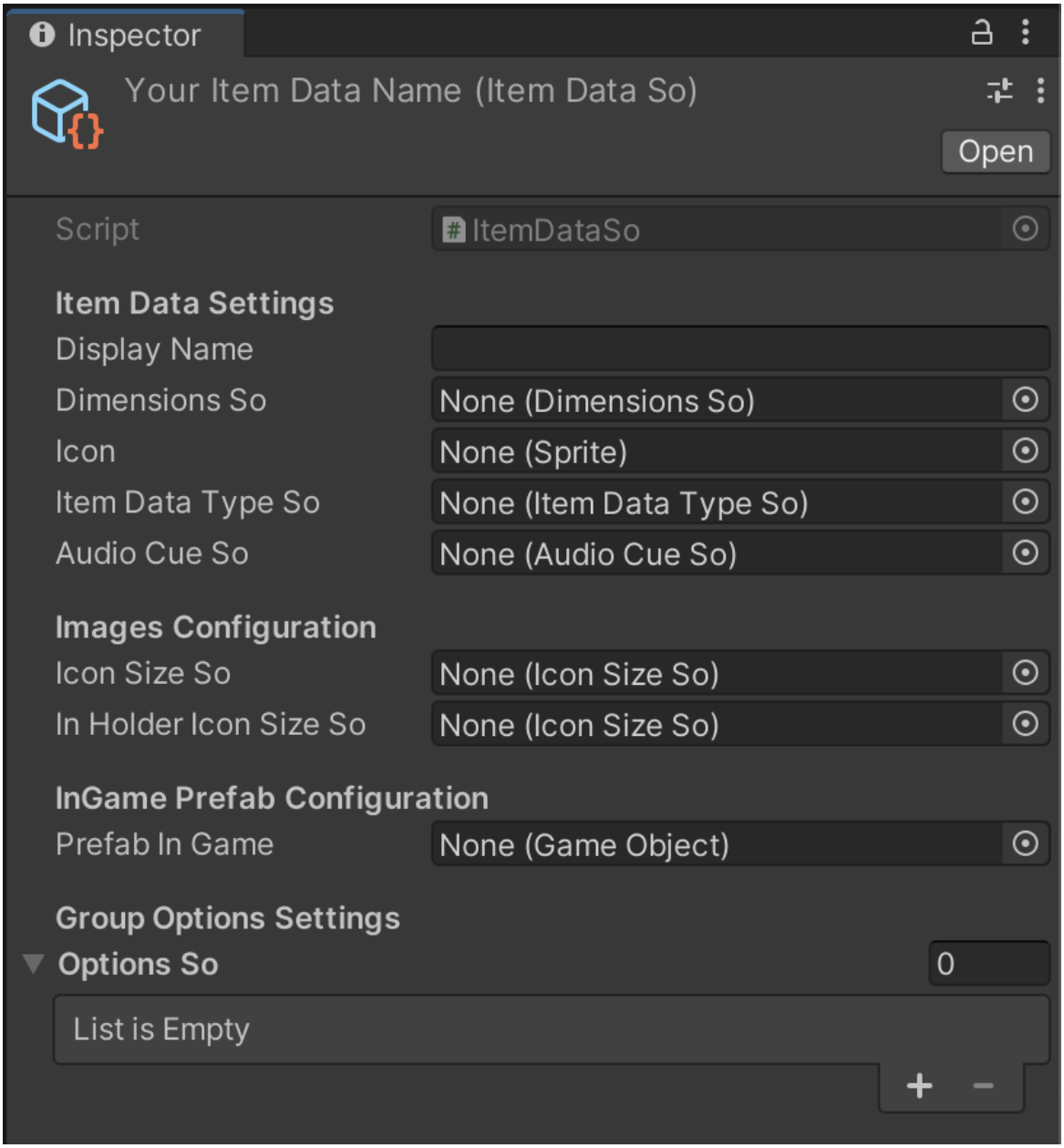
Creating Item Data So in Unity Editor

In order to create an Item Data So, you can right click in the project tab and go to **Inventory** > **Items** > **New Item**. Or if you're creating a Container you can go to the specific doc page.



Setting the properties from the new Item

Now you create the item data file, you can set your item property. Down below you can see what property do.



Item Data Settings

- Display Name is the name that will be displayed in the corner when the item is in the grid.
- Dimensions So is going to be the dimension from the Item in the grid.
- Icon is the sprite is going to be used for display the item icon.
- Item Data Type So is the type from the item, used to validate if the item fits the holder
- Audio Cue So is the audio cue will be used when the item is place or pick

Images Configuration

- Icon Size So is the icon size used when the item is in the grid.
- In Holder Icon Size So is the icon size used for when the item is in a holder

InGame Prefab Configuration

- Prefab In Game is the prefab will be displayed when item is in the “real world”. (**Feature not yet implemented**)

Group Options Settings

- Options So is the options from the item when you right click in it in a grid.

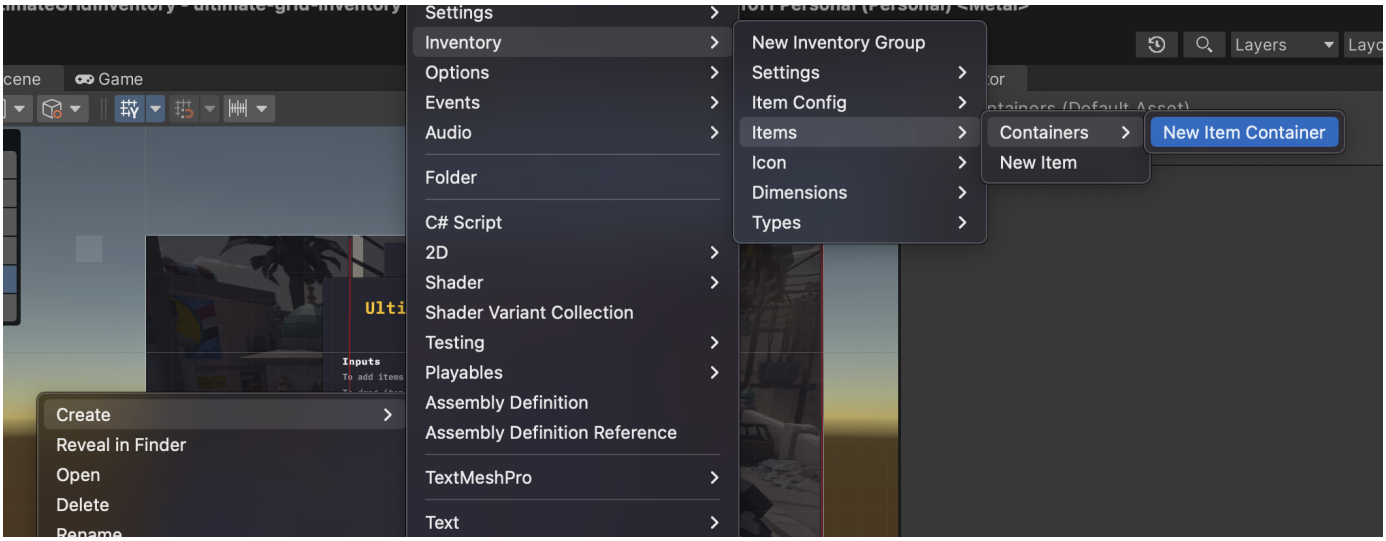
Learning how to create a Container for your Inventory

Create a **Container** is almost similar to an **Item**. The only it change is the fact that we need to pass a **grid prefab**

- Learn how to create an Container Grid

Creating Item Container Data So in Unity Editor

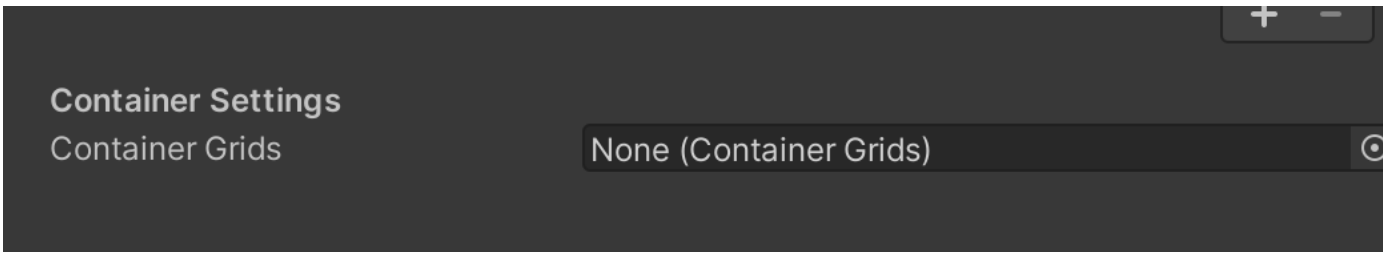
In order to create an Container Data So, you can right click in the project tab and go to **Inventory** > **Items** > **Containers** > **New Item Container**.



Setting the properties from the new Container

Now you create the container data file, you can set your container property. In case you want to understand the **other properties**, you might want to check the **item data so**

The only thing that differ from the **Item Data So** for the **Item Container Data So** is the new property **Container Grids**



Container Settings

- Container Grids is the grid prefab used to display the grid for the container that you created.

Ps: Learn how to create an Container Grid

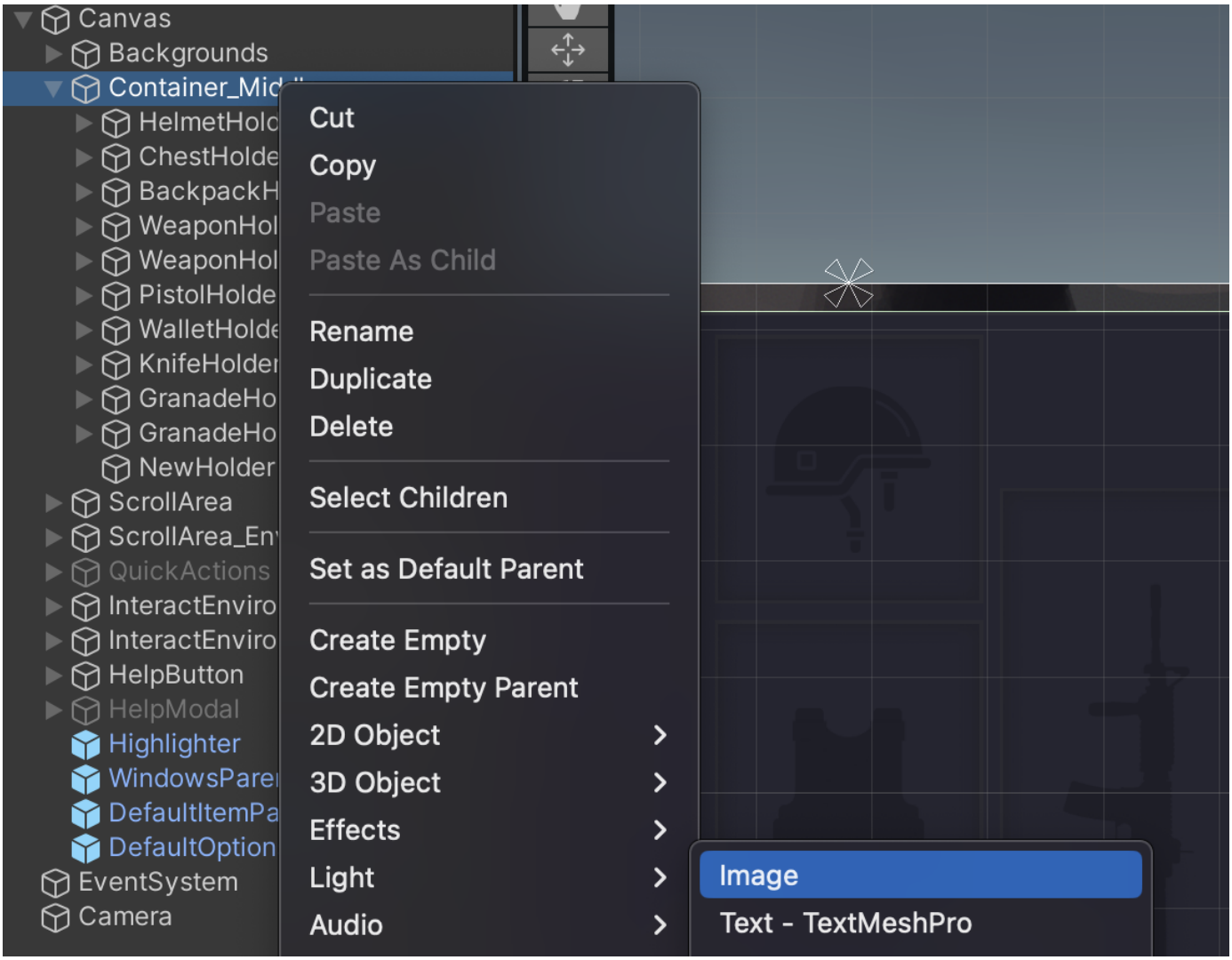
Learning how to create an Item Holder

Item Holder is the name of the script that holds a player item. It could whatever you want, you decide what your game will hold.

Ps: We're **improving** this system in the **next versions**... So things might **change**.

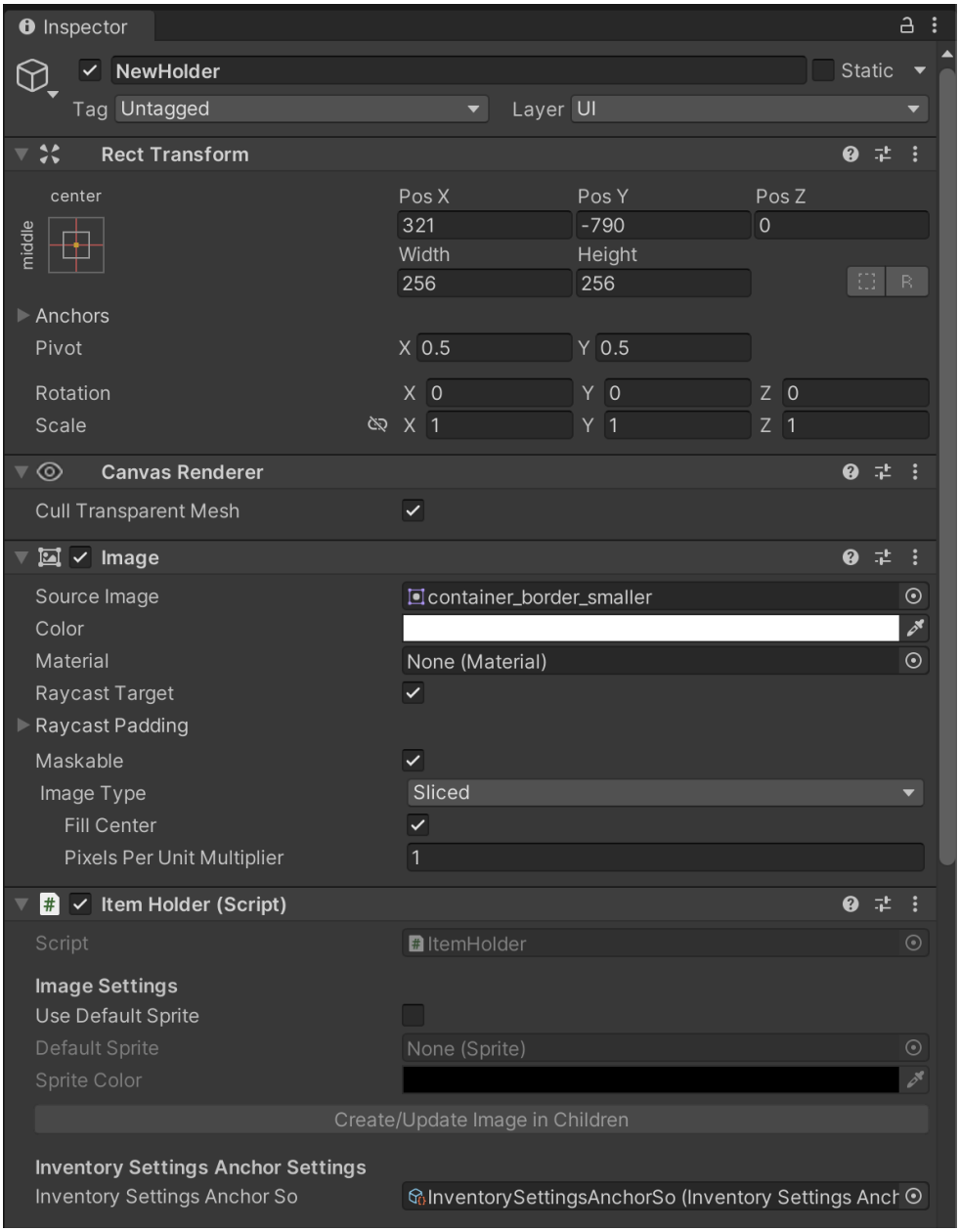
Creating the Item Holder

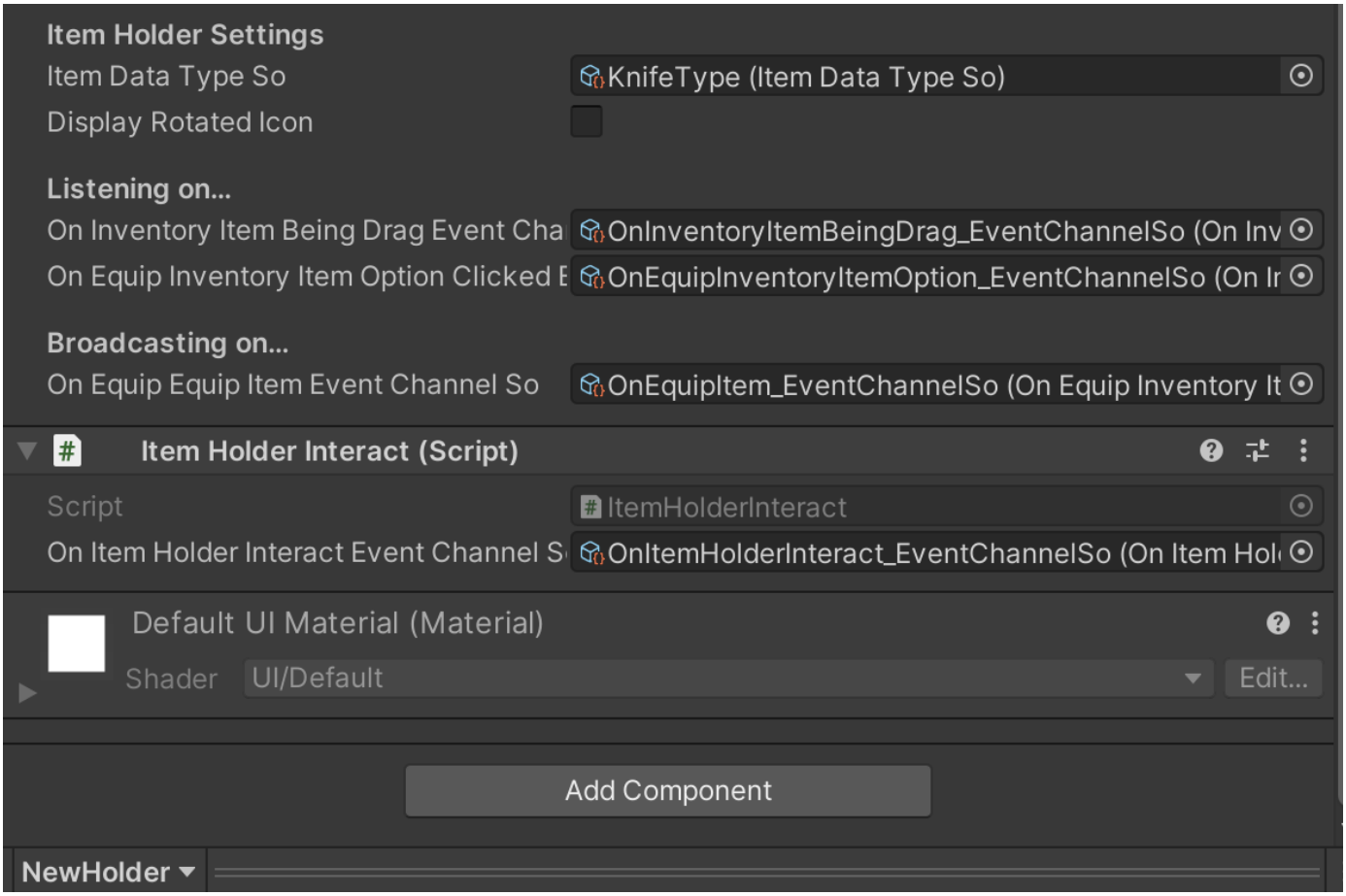
In order to create an **Item Holder**, you first need to create an UI **Image Game Object**, to make it, follow the image below:



After creating the UI **Image Game Object** you need to attach two scripts to it. **Item Holder** and the **Item Holder Interact**.

- Image is the sprite that will be the holder background.
- Item Holder is the script that holds an Item in it. If you're creating an Container Holder you can check the documentation.
- Item Holder Interact is the script that manage when the user interact with the Item Holder.

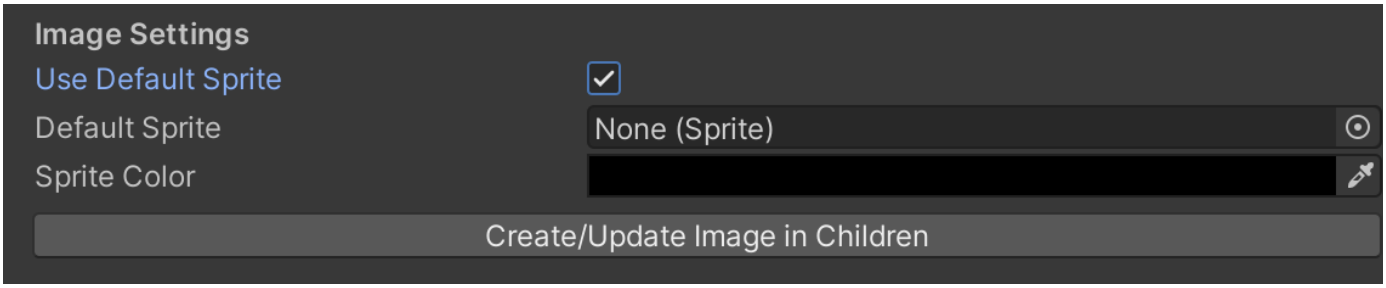

Inspector
NewHolder
Static
Tag Untagged
Layer UI
Rect Transform
center
middle
Pos X
Pos Y
Pos Z
321
-790
0
Width
Height
256
256
Anchors
Pivot
X 0.5
Y 0.5
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Canvas Renderer
Cull Transparent Mesh
Image
Source Image
container_border_smaller
Color
Material
None (Material)
Raycast Target
Raycast Padding
Maskable
Image Type
Sliced
Fill Center
Pixels Per Unit Multiplier
1
Item Holder (Script)
Script
ItemHolder
Image Settings
Use Default Sprite
Default Sprite
None (Sprite)
Sprite Color
Create/Update Image in Children
Inventory Settings Anchor Settings
Inventory Settings Anchor So
InventorySettingsAnchorSo (Inventory Settings Anch



You can set a cool Source Image in the **Image Component** in order to your holder look cooler!

Configuring Image Settings

In case you want a default image to appear when nothing is equipped, you can use **Image Settings** in the **Item Holder**.



Then after setting the **Sprite** and **Color** you can configure the **Image Component** in the children.

Final Result

After doing some adjusts, your Item Holder could look like this:

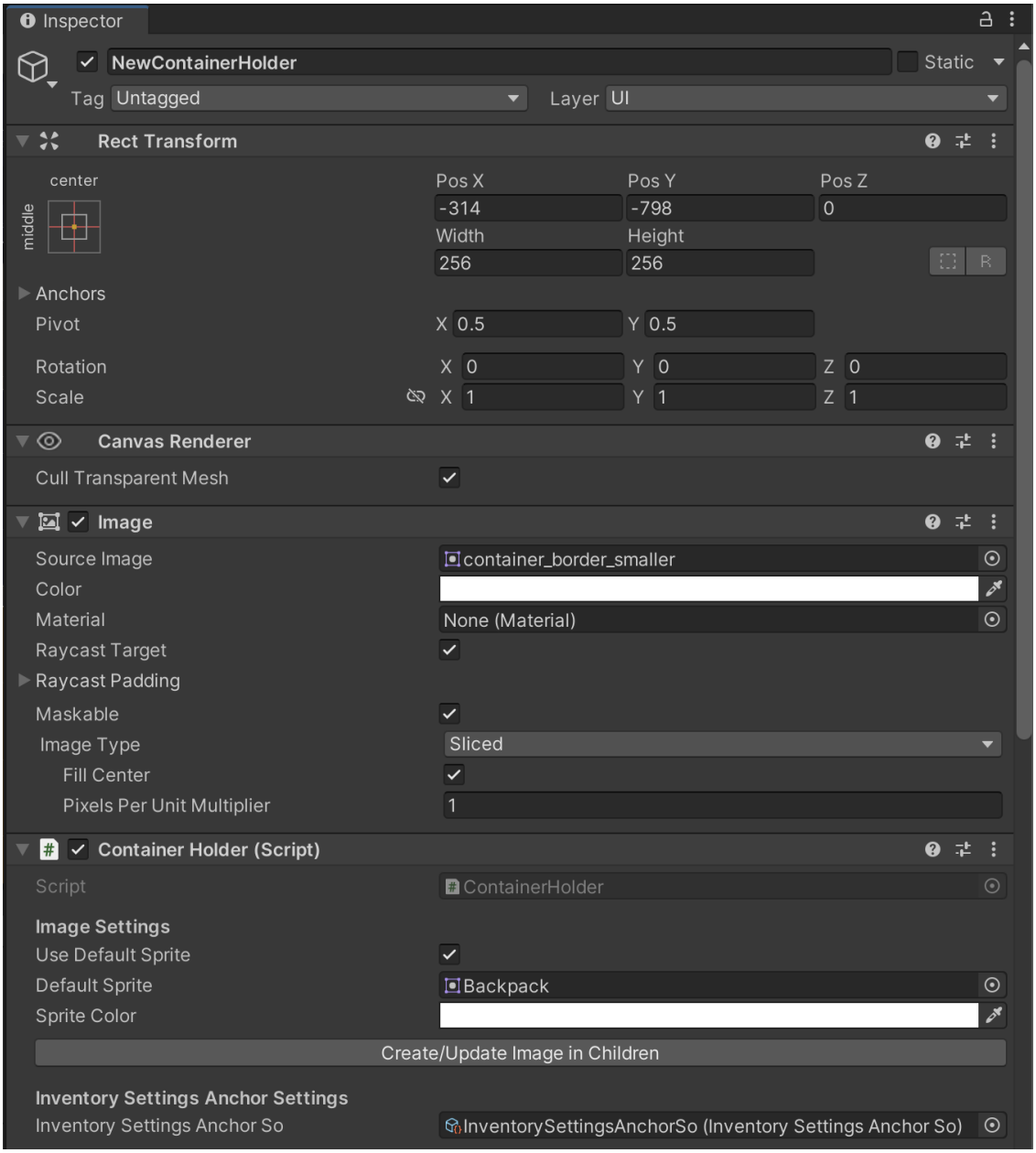
Learning how to create a Container Holder

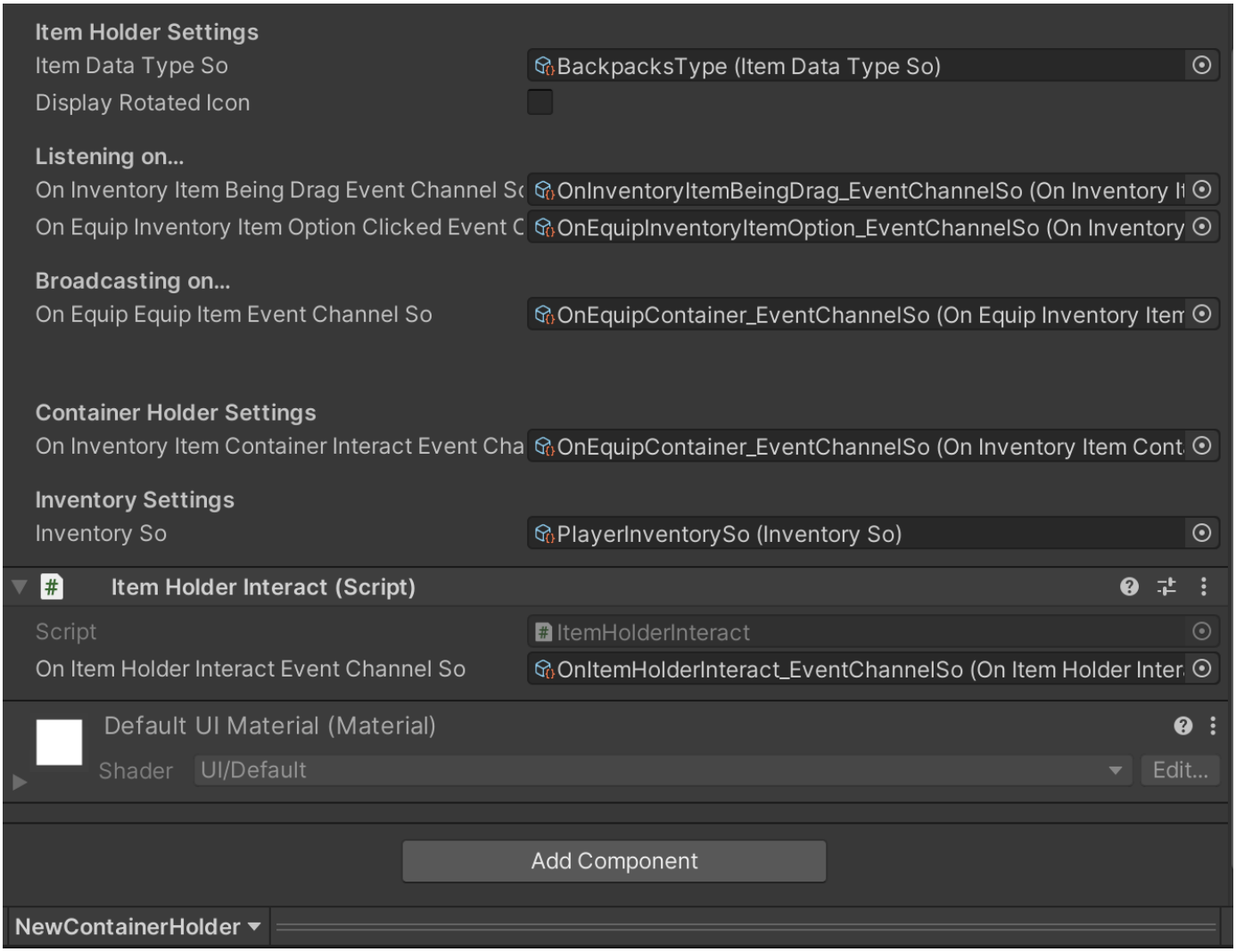
Container Holders are used for holds **Container Items**, items that contains grids. It's very similar to Item Holder, the only difference is that Container Holder contains two new properties.

- On Inventory Item Container Interact Event Channel So when the player equip the container this event will be triggered. Can be used to appear the container in a Scroll Area
- Inventory So that is a scriptable object that holds all the player inventory. This will can check validate all grids that is appearing for the player.

Creating a Container Holder

The creation is similar to the Item Holder, you can follow the same tutorial, **but instead of attaching the Item Holder**, you need to attach a **Container Holder**


Inspector
NewContainerHolder
Static
Tag Untagged
Layer UI
Rect Transform
center
middle
Pos X
Pos Y
Pos Z
-314
-798
0
Width
Height
256
256
Anchors
Pivot
X 0.5
Y 0.5
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Canvas Renderer
Cull Transparent Mesh
Image
Source Image
container_border_smaller
Color
Material
None (Material)
Raycast Target
Raycast Padding
Maskable
Image Type
Sliced
Fill Center
Pixels Per Unit Multiplier
1
Container Holder (Script)
Script
ContainerHolder
Image Settings
Use Default Sprite
Default Sprite
Backpack
Sprite Color
Create/Update Image in Children
Inventory Settings Anchor Settings
Inventory Settings Anchor So
InventorySettingsAnchorSo (Inventory Settings Anchor So)

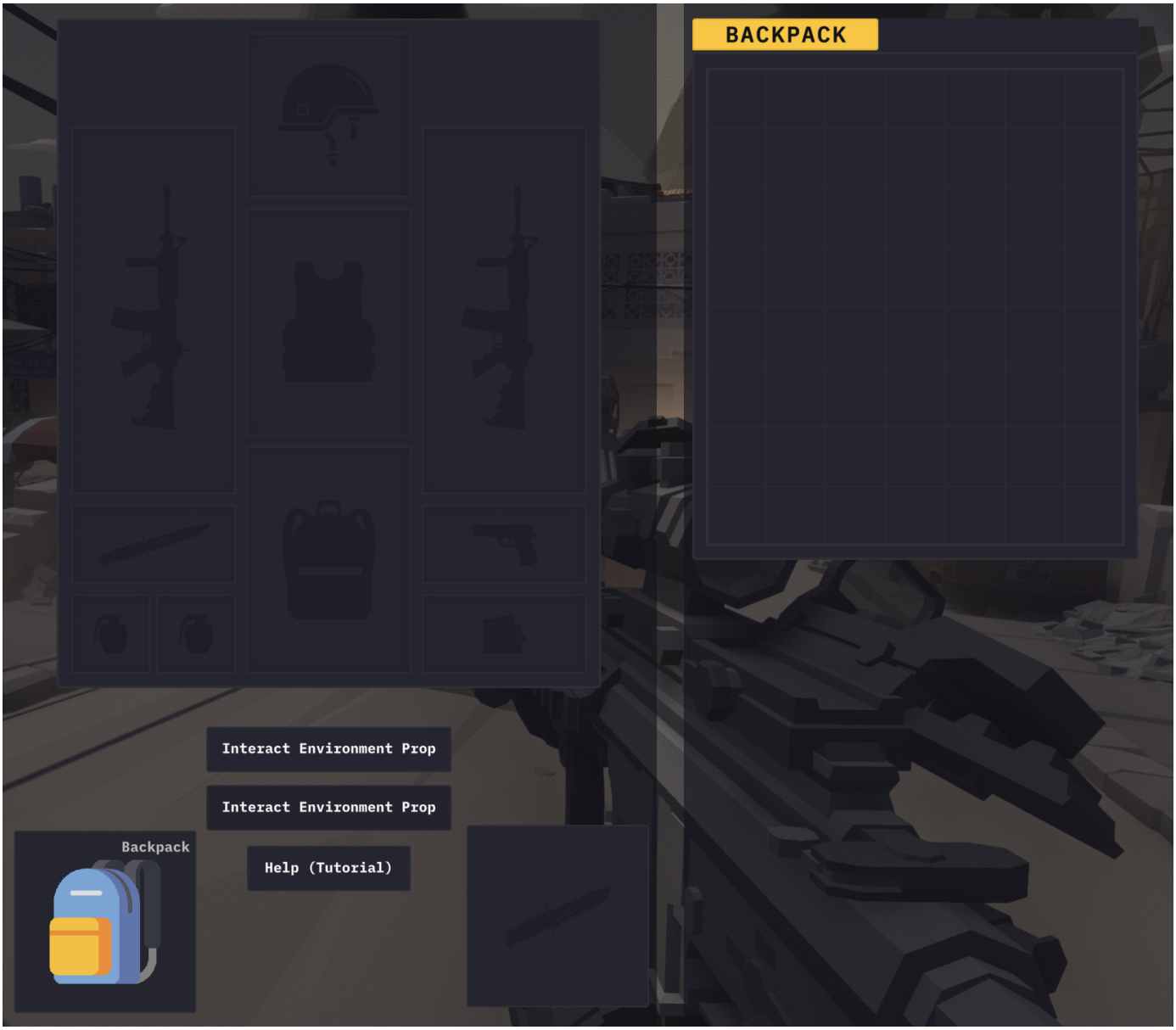


After attaching the On Inventory Item Container Interact Event Channel So and Inventory So, your container holder is configured! You can test now!

Final Results

After some adjustments, your container holder should look like this:

Then if you equip the Backpack, will be added to **Inventory So** and also because the event trigger, the UI on the right will appear the Container Grids



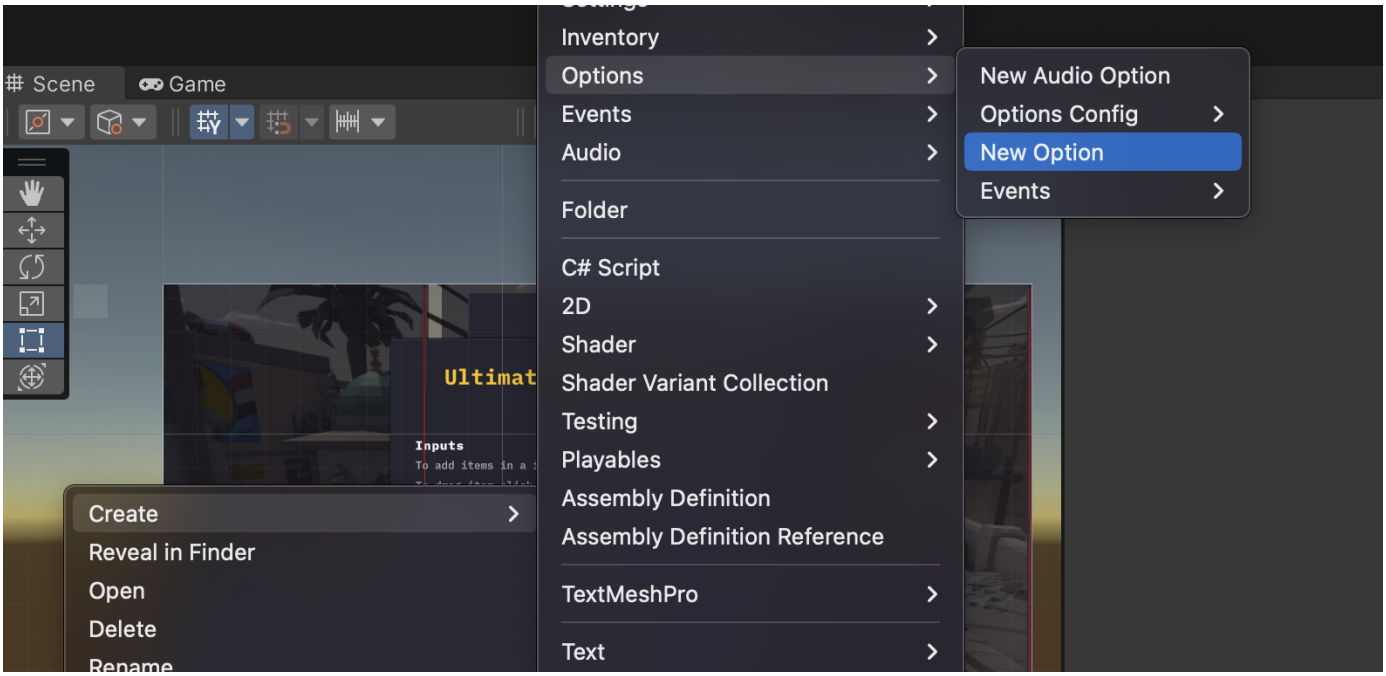
Ps: We’re looking forward to improve this Holder creation in the next versions.

Learning how to create Option with Scriptable Objects

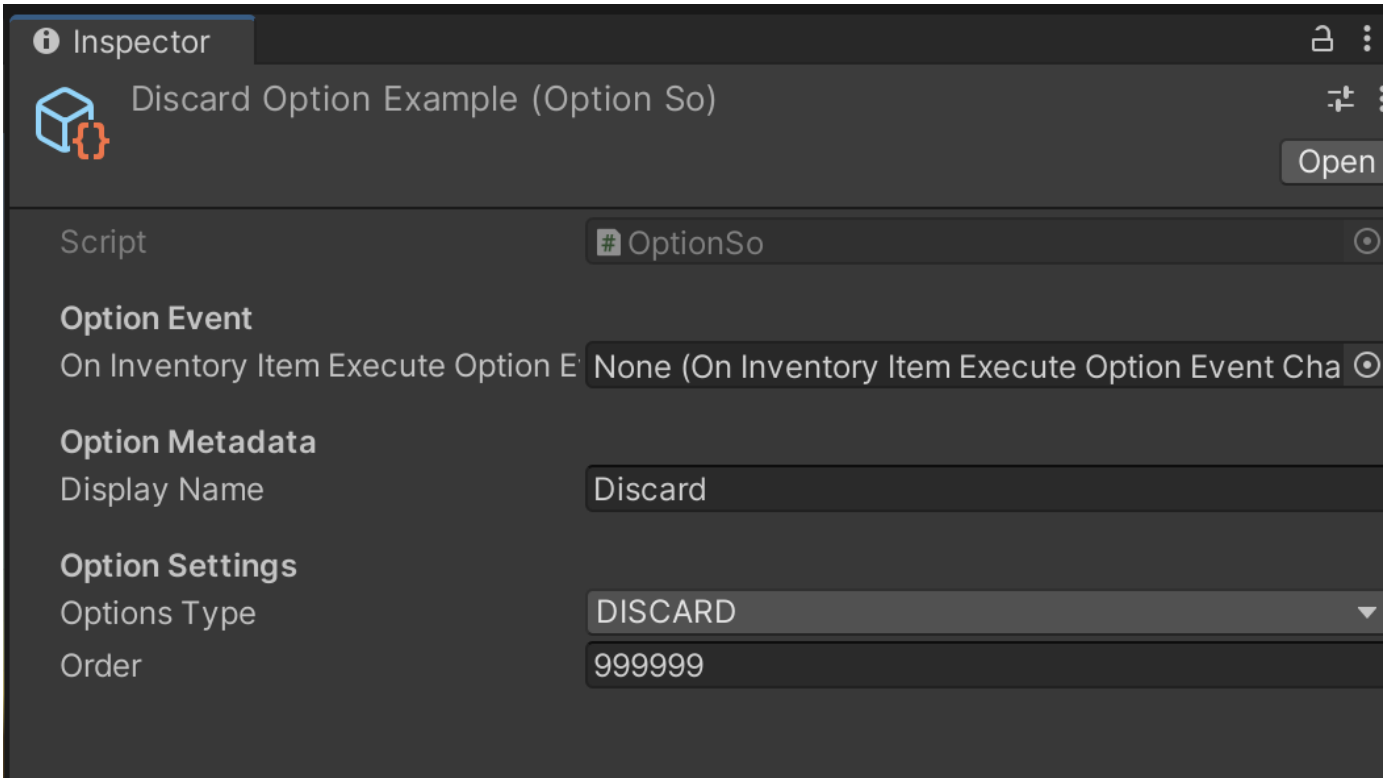
Options in the Ultimate Grid Inventory is based on **Scriptable Object**. The options system is based in **event** trigger. You create an event **attach** to this **option** and then you create a script that listen to that event and do the action you need.

In this documentation page we are going to show how to setup an Discard Option for an example.

Creating the Option So



After creating the Option So you must fill the fields



Understanding the properties

Option Event

- On Inventory Item Execute Option Event Channel So this will be filled later in the documentation.

Option Metadata

- Display Name is going to be the name that will be displayed in the game.

Option Settings

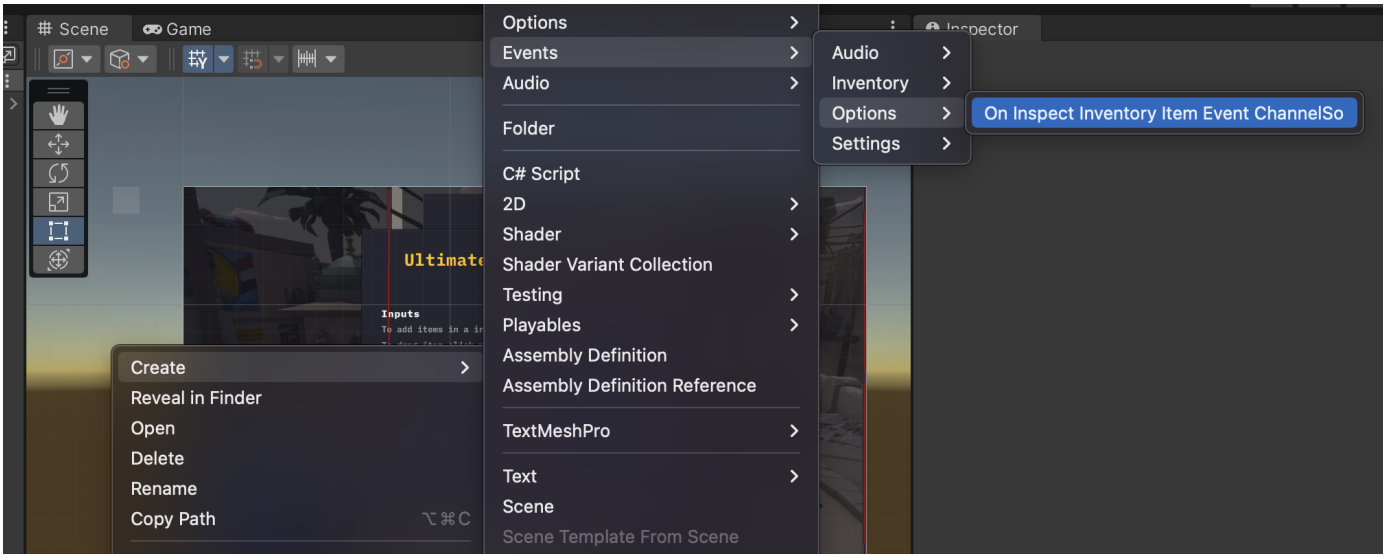
- OptionsType is the type used for add and remove from the InventoryMetadata class.
- Order the order that will be displayed in the list of options. **Higher** is going to **appear lower** in the sequence.

The **OptionsType** you can add new

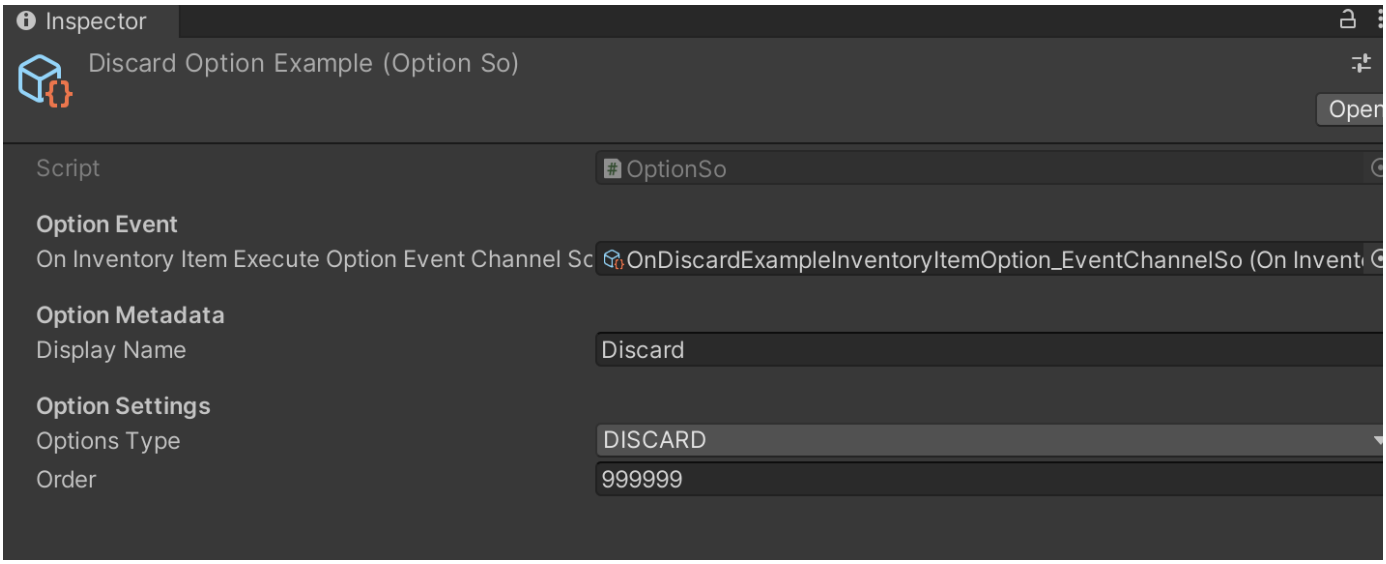
```
public enum OptionsType
{
  INSPECT,
  EQUIP,
  UNEQUIP,
  OPEN,
  DISCARD,
  // Add your new OptionType here...
}
```

Creating the event channel for Option So

It's pretty simple to create an event channel for you new option so. Here is how you do it



After creating the event channel, you can **attach** this event to the **option** that you **created before**. Your option will look something like this:



Creating the Option Listener

After you create your option, you might want to add the action for it. To do it, is pretty simple. First of all, you need to create a **MonoBehavior** with a **SerializableField** from your **Event Channel**. You need to attach this **MonoBehavior** to a GameObject in the Scene.

The MonoBehavior should look like this:

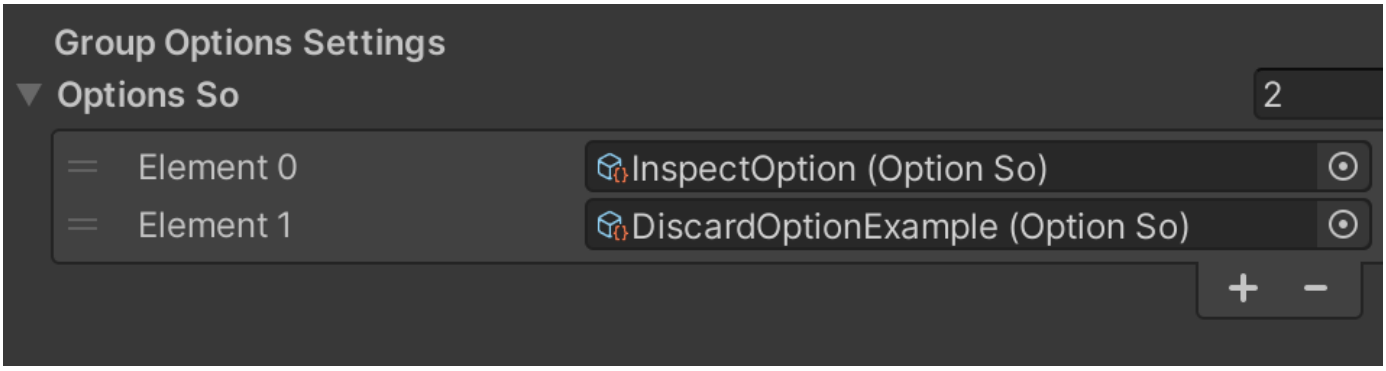
```
using Inventory.Scripts.Inventory;
using Inventory.Scripts.ScriptableObjects.Events;
using UnityEngine;

public class DiscardOptionExample : MonoBehaviour
{
    [SerializeField] private OnInventoryItemExecuteOptionEventChannelSo onInventoryItem

    private void OnEnable()
    {
        onInventoryItemExecuteOptionEventChannelSo.OnEventRaised += HandleDiscardOpt
    }

    private void OnDisable()
    {
        onInventoryItemExecuteOptionEventChannelSo.OnEventRaised -= HandleDiscardOpt
    }

    private void HandleDiscardOption(InventoryItem inventoryItem)
    {
        // Add your action in here...
        Debug.Log("Executing option...");
```

```
  }
}
```

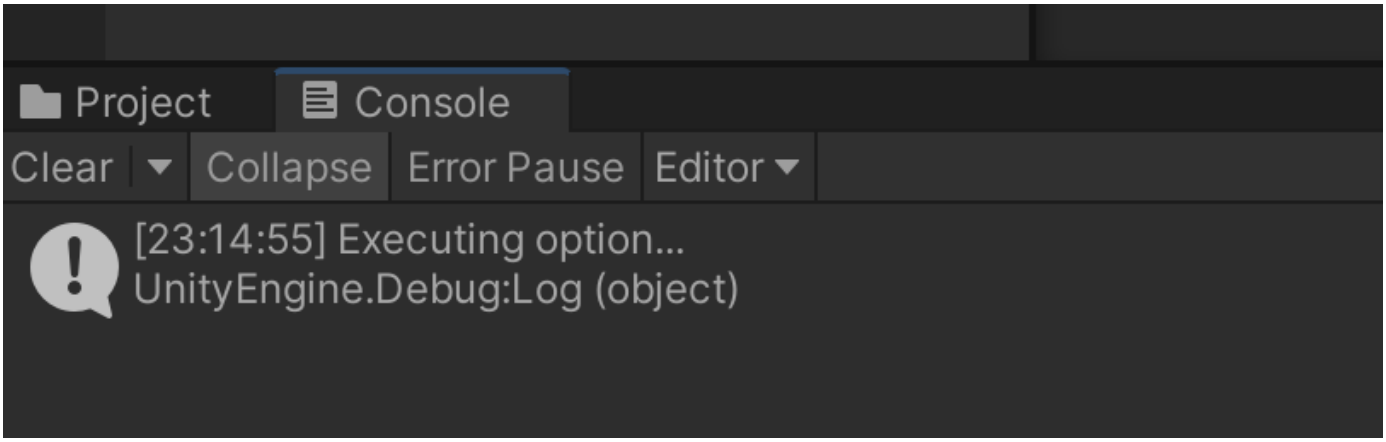
For the final step you need to add this option for an Item.

Adding option to an Item

In the Item that you desire to add the option, in field Group Options Settings you can add your option in the list. It should look like this:

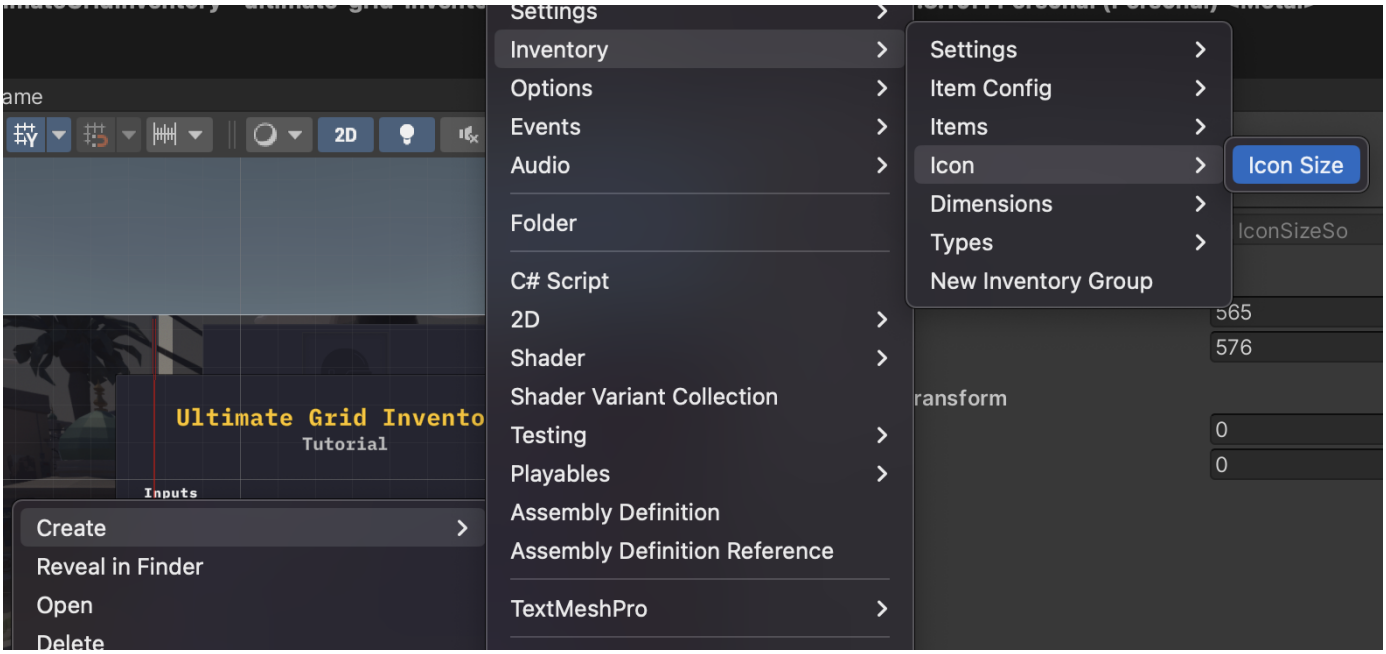


Then you can hit **play** and see your **log** from your **Option**.



Customize your item icon using Icon Size

Learn how to create an Icon Size to improve your items icons. In order to create an Icon Size, you will use the default unity creation file system.



The **filename** we used to use is something like IconSize_WIDTH_HEIGHT or IconSize_WIDTH_HEIGHT_PosX_PosY just to make easy to found the icon size with the correct width, height, pos x and pos y.

Configuring properties

Icon Settings

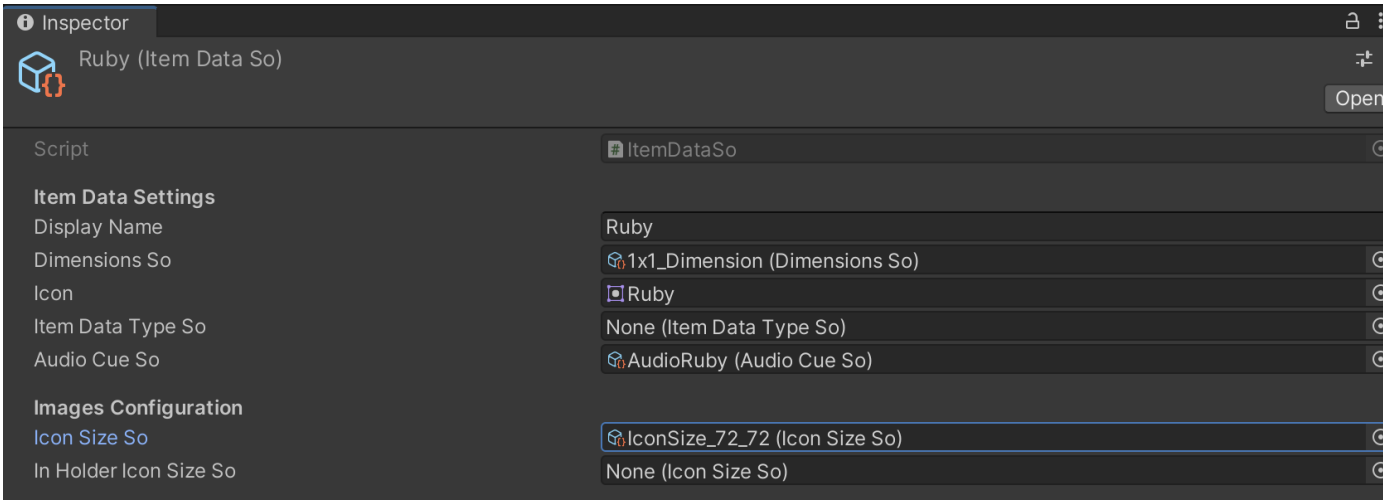
- Width from the item icon that will be shown in the item grid.
- Height from the item icon.

Position Rect Transform

- Pos X is the value that will alter the **Pos X** from the **item icon**. **Default**: 0
- Pos Y is the value that will alter the **Pos Y** from the **item icon**. **Default**: 0

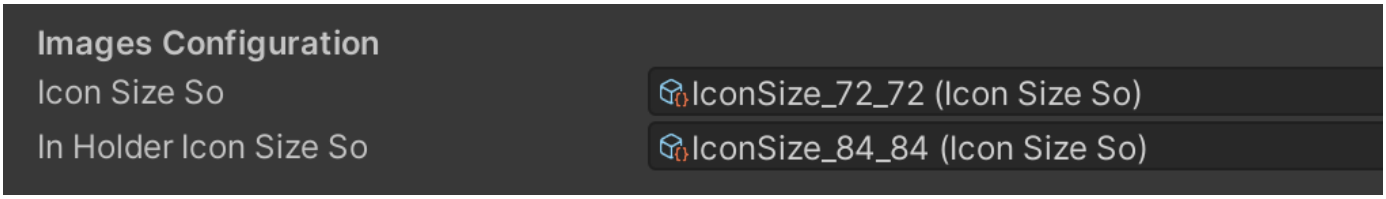
Setting Item with the Icon Size

After you create the IconSize you can set this icon size in a ItemDataSo



In the **Images Configuration** area. It is possible to configure the icon size when the item is in an *inventory slot* or in a *item holder*

- Icon Size So: The size of the item icon when is in an **inventory**
- In Holder Icon Size So: The size of the item icon when is equipped in an **holder**

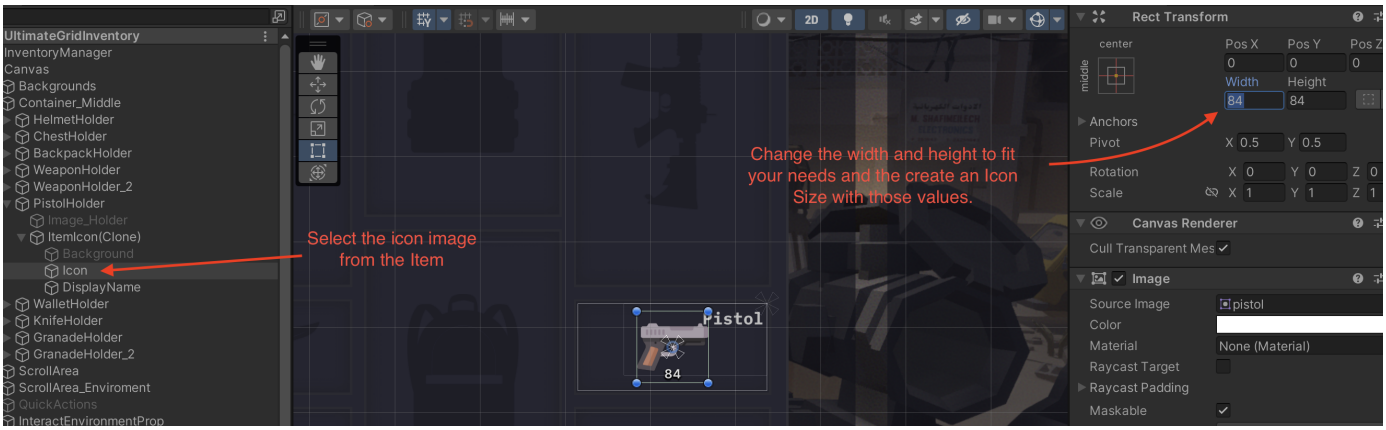


After set all configurations you can create or update a current Icon Size, then just hit play and everything should work!

Getting the right *width* and *height* for your icon

Getting sizes in the runtime

In order to get a cool looking *width* and *height* for your Icon Size, you can check in **Runtime** and alter the **width** and **height** from the Icon from the item.



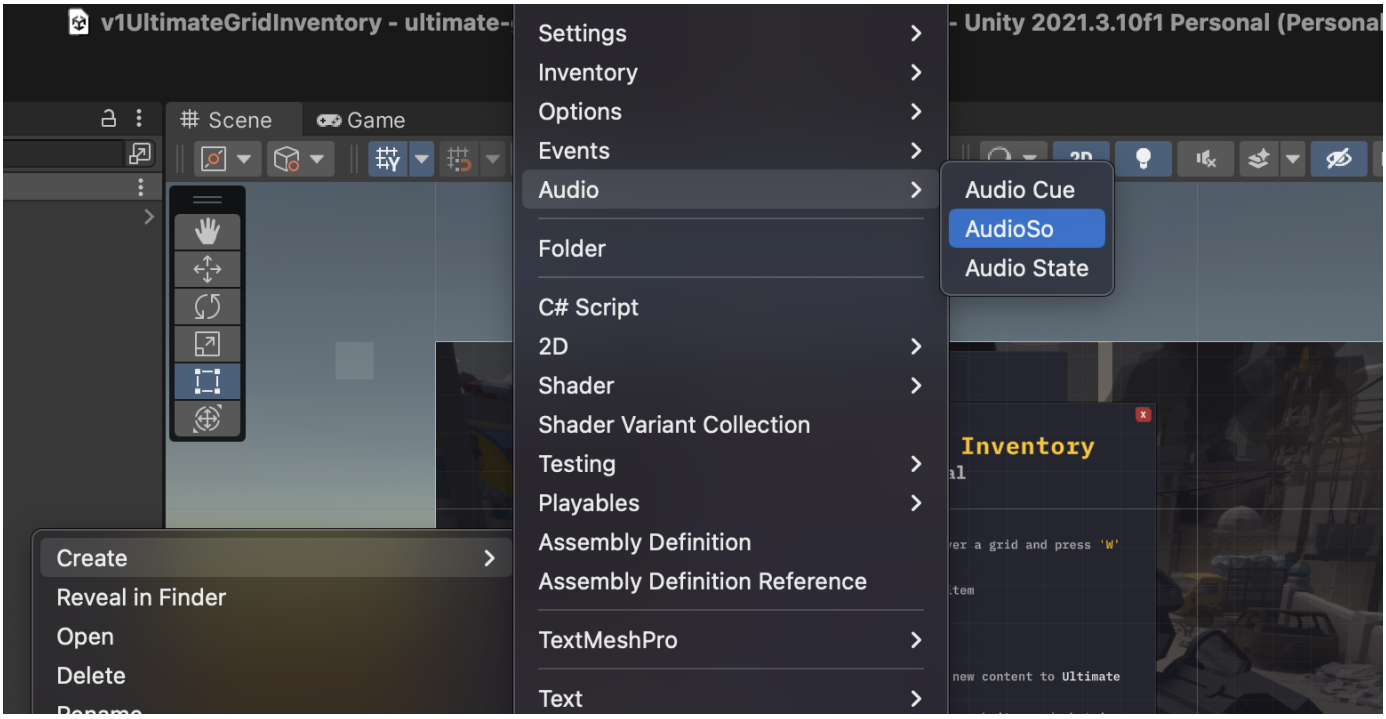
Create or Update your Icon Size

Now you need to create or update the IconSize, in case you want to learn how, [go to create an icon size doc. page]

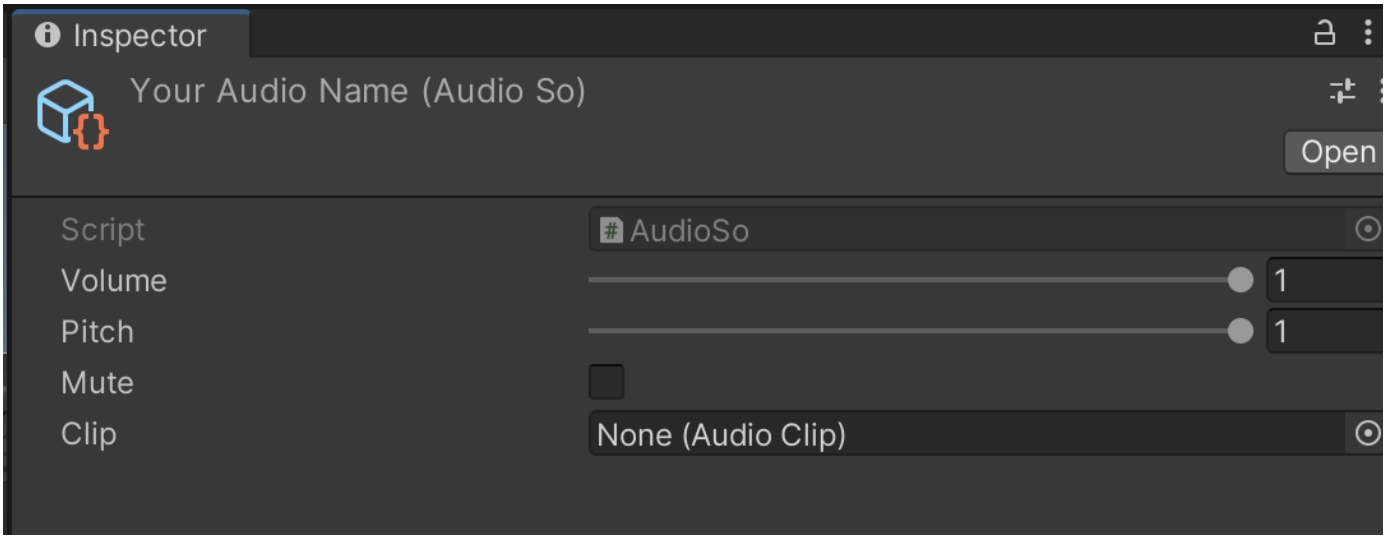
Learn how to import sounds with AudioSo

AudioSo is a scriptable object used to map all the audios used in the **Inventory**. It's used along with [Audio State So] that are the randomizer to make the audio look cooler.

First of all, you need to create an AudioSo, following the image below:



After creating the **AudioSo** you will need to configure it



Understanding properties

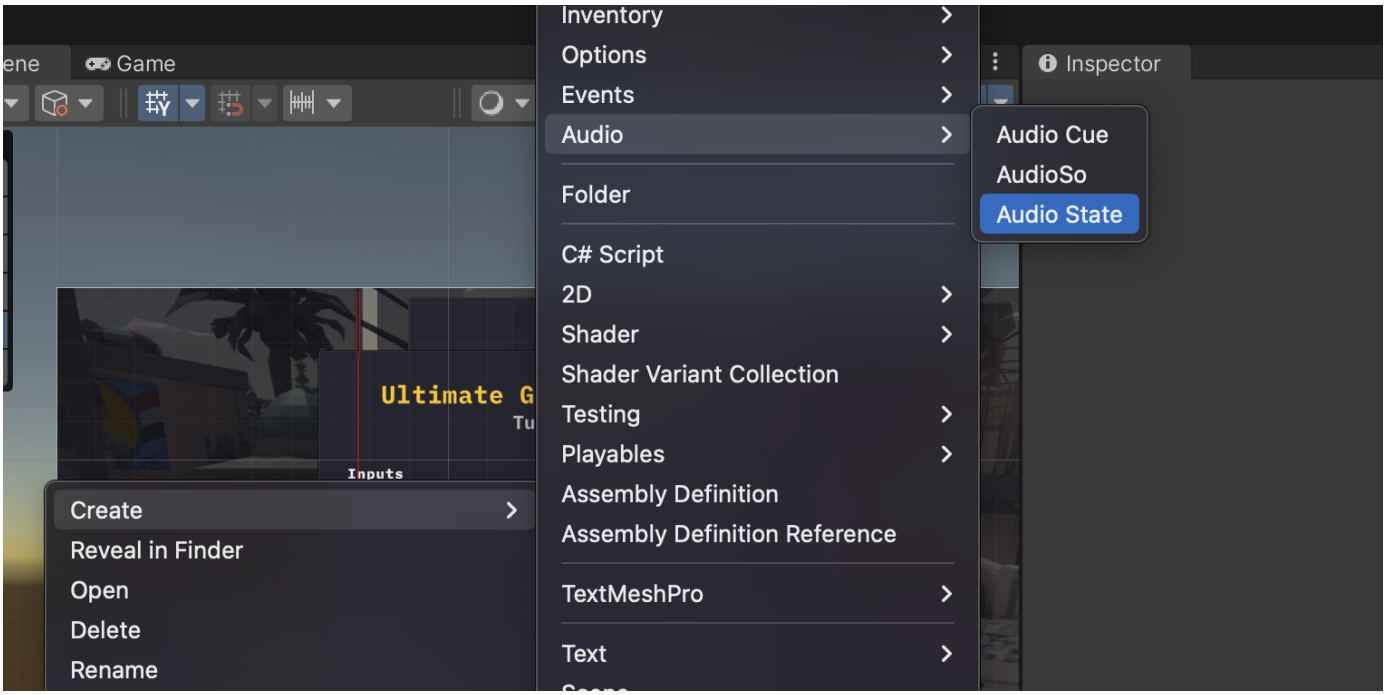
- Volume is going to be the volume from audio. It goes from **0** to **1**
- Pitch is the pitch from audio. It goes from **0** to **1**
- Mute in case you want to Mute the audio.
- Clip the audio clip that you are importing.

Configuring AudioStateSo

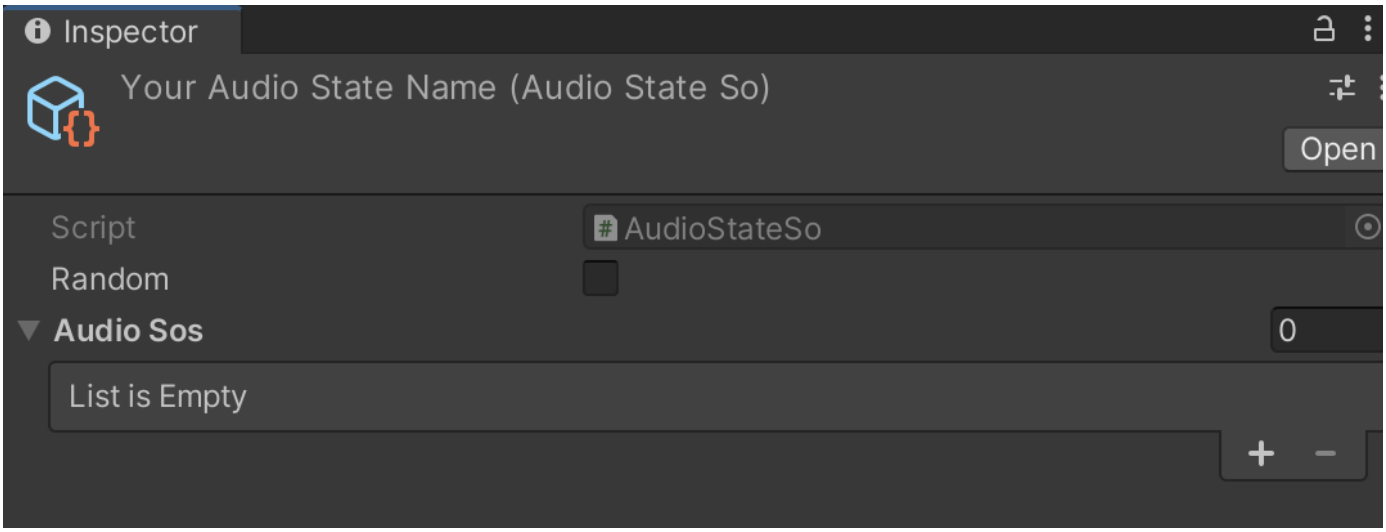
AudioStateSo is a scriptable object used to randomizer and to hold all **AudioSos** in order to play the audios when *pick*, *place* or *execute option* from an item in the inventory.

Creating Audio State So

In order to create the **Audio State So** you can right click in a folder on the project tab, then **Audio** > **Audio State**



After you created the Audio State file, you must configure. It's pretty simple to configure, you have two properties.



Understanding the properties

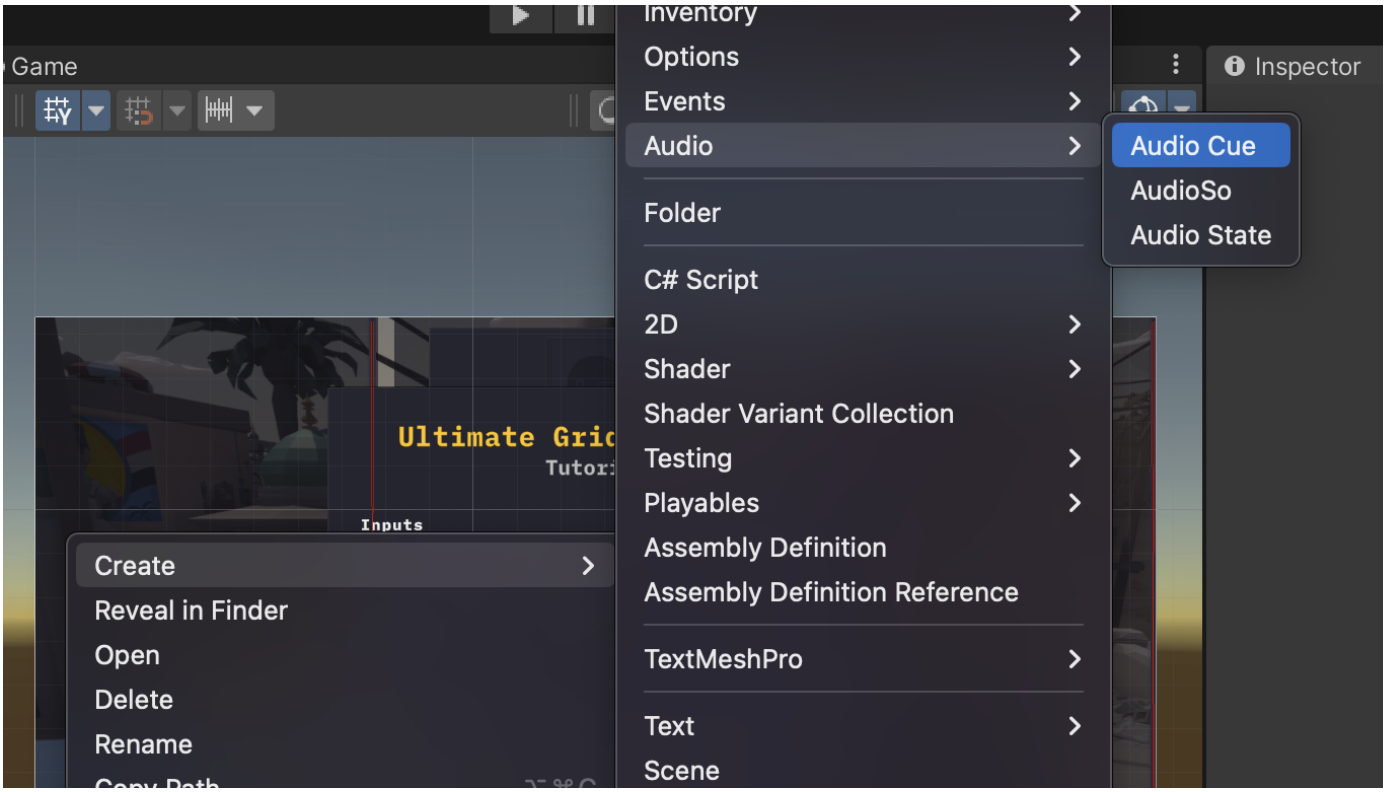
- Random is a boolean used for **randomize** the sequence of AudioSos when playing. If **off** will follow the sequence provided in the AudioSos list
- Audio Sos list of audio sos used for playing audio when *pick*, *place* or *execute option* in the inventory.

Learn how to setup your Audio Cue So

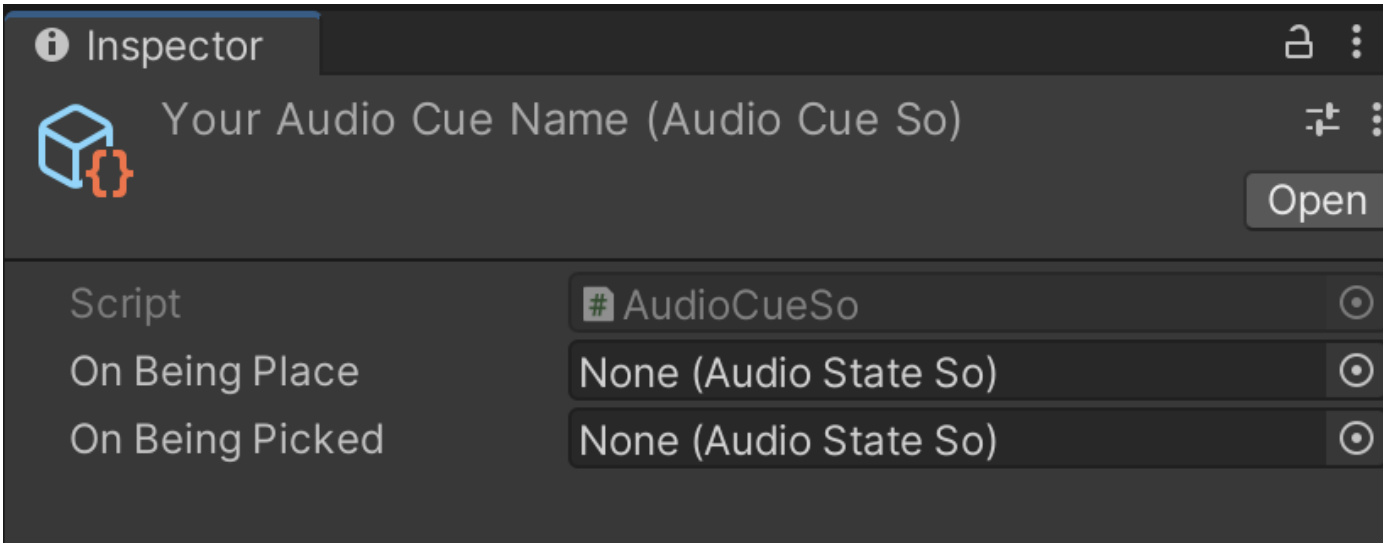
Audio Cue So are used in the **ItemDataSo** that is the scriptable object that represents an Item in the game. You can set the property **Audio Cue So** in the **Item Data So**.

The AudioCueSo contains two properties, *On Being Place* and *On Being Picked*.

Creating your Audio Cue So



After you create the file, you must configure the properties. There are pretty easy to configure, follow the section below.



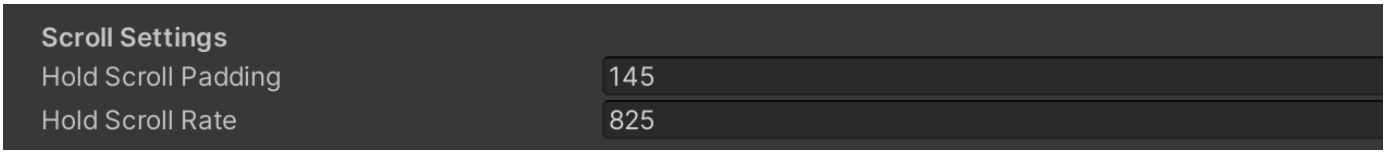
Understanding the properties

- On Being Place is the audio state so that will be executed when the Player place an item in the inventory.
- On Being Picked is the audio state so that will be executed when the Player picked an item from the inventory.

Changing the settings from Scroll Settings

The scroll settings in the **Ultimate Grid Inventory** is used to config how fast and how long is the *padding* for **auto scroll** when *dragging* an item.

You can configure those values algo in the Inventory Settings So



- Hold Scroll Padding is the *padding* from the borders in a *Scroll Area* where will auto scroll.
- Hold Scroll Rate is how *fast* will move when auto scrolling. **Higher** is fast

Remember that the Scroll Area should be from the *UGI implementation* that is called Inventory Scroll Rect and needs a Scroll Rect Interact.