INVENTORY SYSTEM
QUICK START GUIDE

Owner: Mehmet Hayri Çakır

v.0.1.1

Istanbul, January 2020

Table of Contents

1. Change Log

- Fixed a visual bug that shows items icon in wrong place.
- Fixed a bug that adds infinite number of items to player, when you try to take an item from chest.

2. Introduction

Goal of this project is to create a modular, scalable, and ready-to-use inventory system.

3. Requirements

I used 2 assets from the Unity Asset Store, they are in project, so you do not have to download them:

3.1. ProceduralUIImage

I used it to create UI elements of this inventory system.
https://assetstore.unity.com/packages/tools/gui/procedural-ui-image-52200

3.2. SerializableDictionary

I used it to create serializable dictionary for item properties in ItemData scriptable object.
https://assetstore.unity.com/packages/tools/integration/serializabledictionary-90477

4. Setup

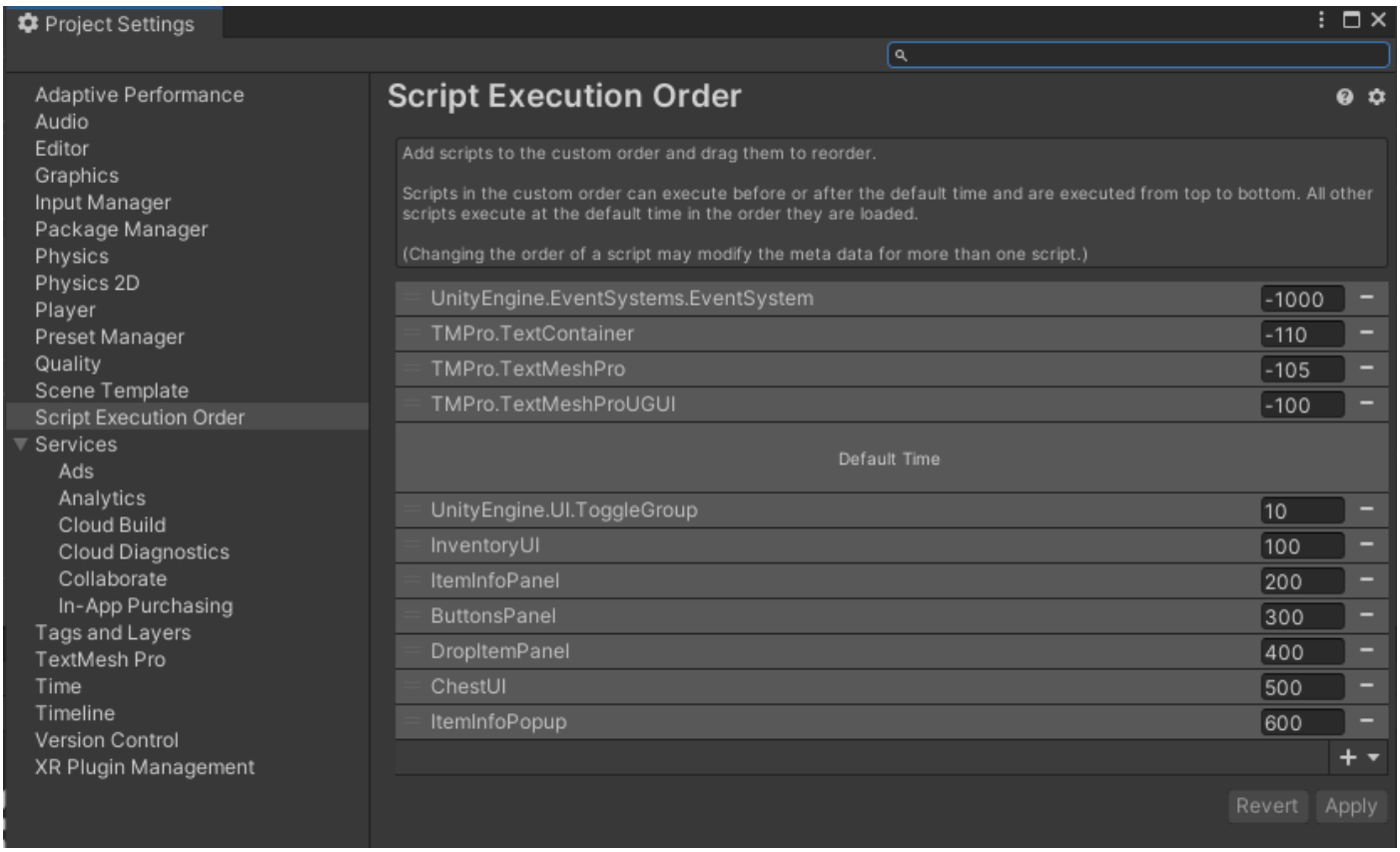
You need to follow these steps to setup the inventory system:

4.1. Script execution orders

You need to setup the script execution orders. It is necessary for Singleton patterns to work correctly.

1) Edit->Project Settings->Script Execution Order
2) Order must be: InventoryUI, ItemInfoPanel, ButtonsPanel, DropItemPanel, ChestUI, ItemInfoPopup.

You can see the screenshot below.



4.2. Create a new scene

Create a new empty scene or use an existing one.

4.3. Create a floor

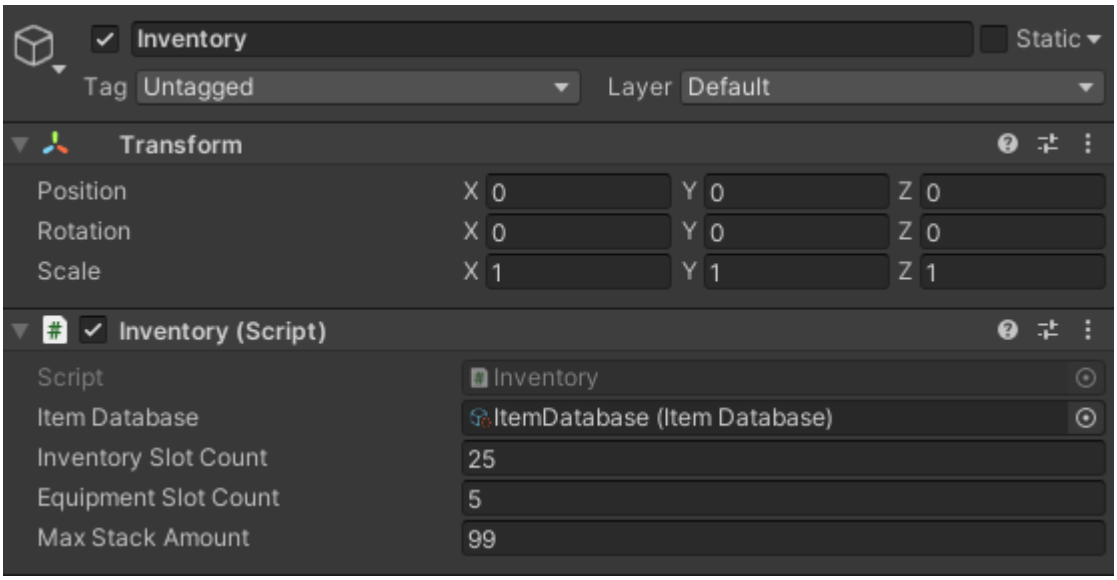
Create a plane to act as floor/ground.

4.4. Creating the inventory

You only need one inventory since there will be one player. You can follow these steps to easily setup the inventory:

1) Create an empty game object.
2) Add “Inventory” script to it.
3) Attach “ItemDatabase” to the inspector field named “ItemDatabase”.
   (Default location for ItemDatabase: /ScriptableObjects/)
4) Set the other fields. They are pretty much self- explanatory. (InventorySlotCount, EquipmentSlotCount, MaxStackAmount)

You can see example values in the screenshot below.

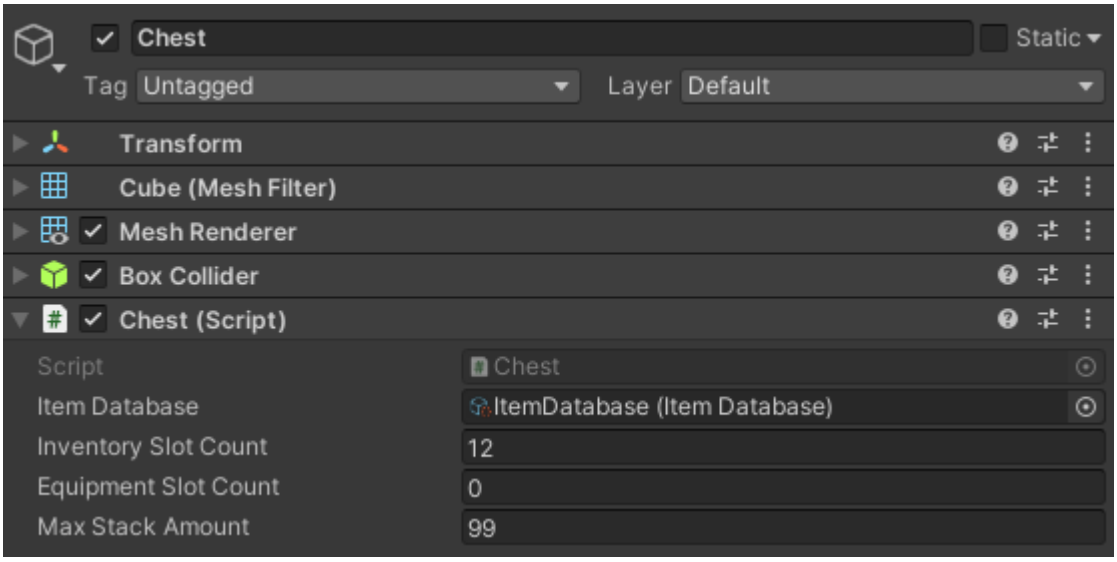


4.5. Creating a chest

You can create as many chests as you wish. But you can only open one chest window at any time. There is just one chest window for UI and all chests use it. You can follow these steps to easily create a chest:

1) Create a cube or any other type of game object that has a collider component. (Chest object requires a collider component.)
2) Add “Chest” script to it.
3) Attach “ItemDatabase” to the inspector field named “ItemDatabase”.
   (Default location for ItemDatabase: /ScriptableObjects/)
4) Set the “InventorySlotCount” and “MaxStackAmount” fields. Leave “EquipmentSlotCount” field as 0.

You can see example values in the screenshot below.

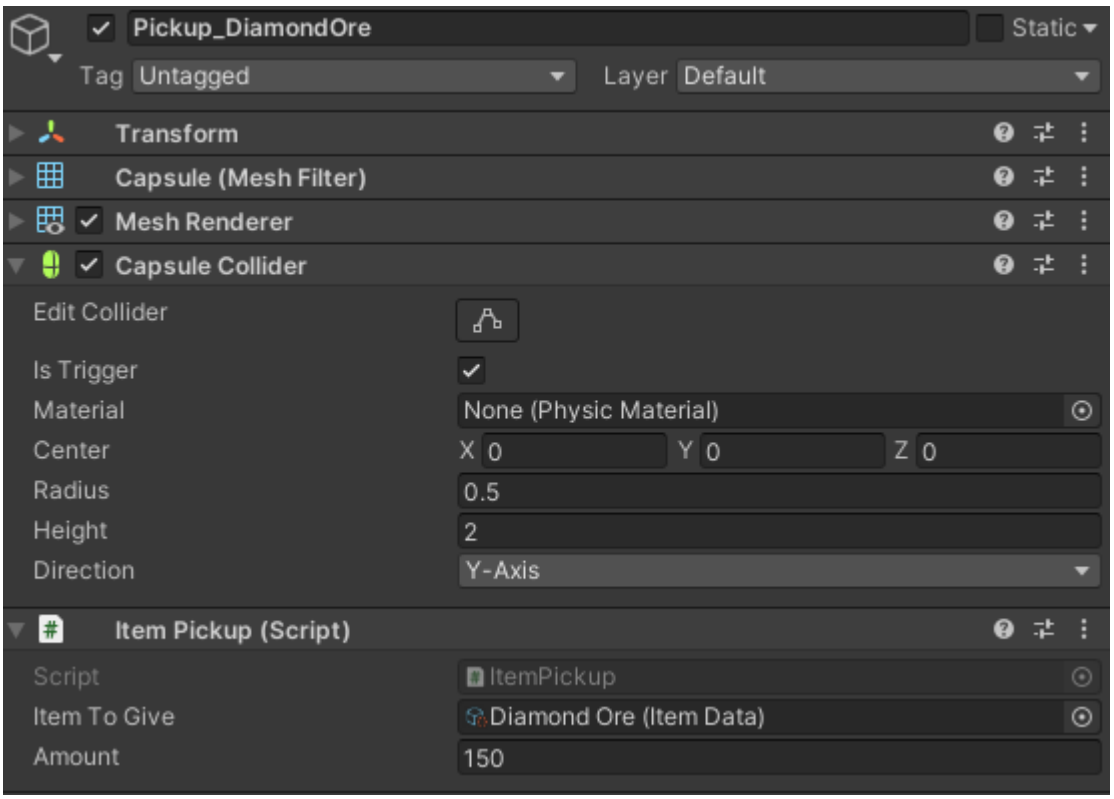


4.6. Creating a pickup item

You can easily create a pickup item by following these steps:

1) Create a capsule or any other type of game object that has a collider component.
2) On collider component, check the “Is Trigger” box.
3) Add “ItemPickup” script to it.
4) Attach an “ItemData” scriptable object or any other scriptable object that inherits from ItemData to “ItemPickup” script. You can find some examples in directory /ScriptableObjects/Items/
5) Set the amount.

You can see example values in the screenshot below.

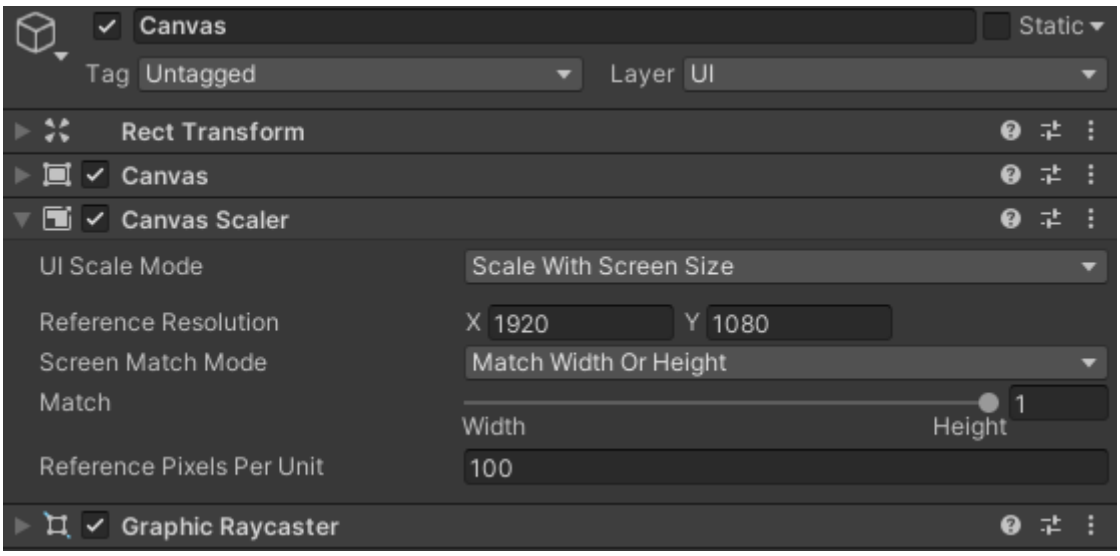


4.7. Creating the canvas

It is a bit harder from the previous ones but if you follow the steps, you will be fine. I hope :)

1) Right Click to the Hierarchy panel->UI->Canvas. It will automatically create the EventSystem, do not delete, we will need it.

2) Change the fields of Canvas Scaler component as shown below: (if you use other settings, UI elements may seem strange, uneven, or badly aligned)



4.7.1. Setting up “InventoryWindowRoot” prefab

“InventoryWindowRoot” is the upmost object of user inventory UI. It is in the following directory: /Resources/Prefabs/Inventory/

1) Add "InventoryWindowRoot” prefab to Canvas as child object. (Drag&Drop it on Canvas in hierarchy)

It has various fields you must populate but they are filled already in the prefab. You can see what fields they are in the 3. Chapter.

4.7.2. Setting up “ChestWindowRoot” prefab

“ChestWindowRoot” is the upmost object of chest inventory UI. It is in the following directory: /Resources/Prefabs/Chest/

1) Add "ChestWindowRoot” prefab to Canvas as child object. (Drag&Drop it on Canvas in hierarchy)

It has various fields you must populate but they are filled already in the prefab. You can see what fields they are in the 3. Chapter.

4.7.3. Setting up “Cursor” prefab

You must aim at the chest to interact with it, so I created a very simple cursor prefab. You can find it in the following directory: /Resources/Prefabs/

1) Add “Cursor” prefab to Canvas as child object. (Drag&Drop it on Canvas in hierarchy)

You can also use custom cursor of your own.

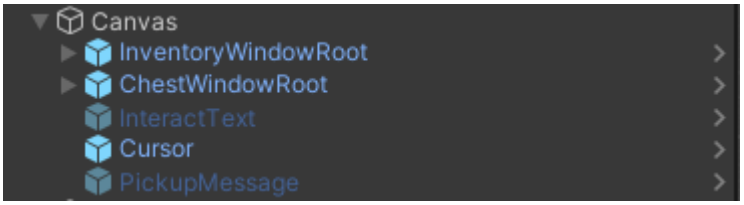
4.7.4. Setting up “PickupMessage” prefab

This is the text that will appear on upper right side of the screen when you pick up an item. You can find it in the following directory: /Resources/Prefabs/

1) Add “PickupMessage” prefab to Canvas as child object. (Drag&Drop it on Canvas in hierarchy)

It is disabled by default. When you pick up an item, it will become visible so leave it disabled. You can create your own Text element too.

Now Canvas is ready too. It should look like the picture below if you are following so far.

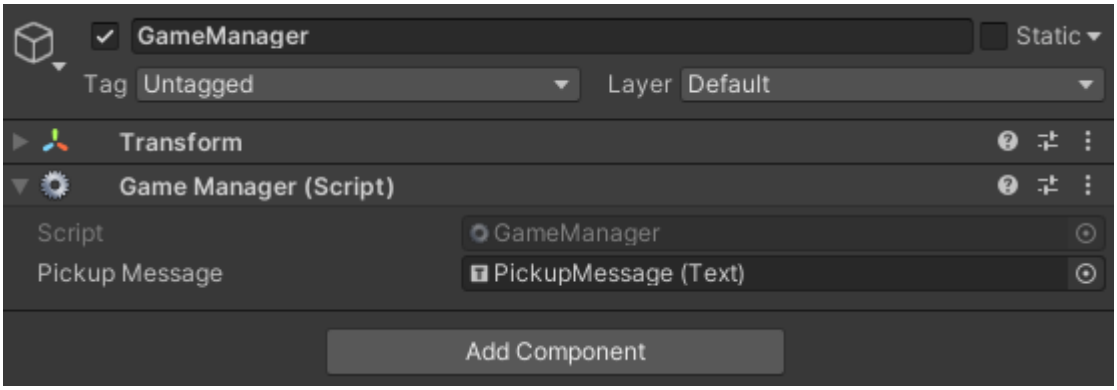


4.8. Creating the game manager

I implemented a game manager to give items to the player. Also, item pickups give items to the player via game manager's Singleton instance. It is very straight-forward to setup:

1) Create an empty game object.
2) Attach the "GameManager" script to it.
3) If you want to see messages when you pick up items, attach the child object of Canvas named "PickupMessage" to the Pickup Message inspector field of GameManager script. Otherwise, you can leave it empty.

See an example screenshot below.



5. How to test the system

There are two ways that you can test the system:

5.1. Demo scene

You can use the demo scene.

5.2. Second method

After you setup everything above, you can set up the player and camera too.

5.2.1. Setting up "InteractText" prefab

This is the text that will appear on lower right side of the screen when you can interact with a chest. You can find it in the following directory: /Resources/Prefabs/

1) Add " InteractText" prefab to Canvas as child object. (Drag&Drop it on Canvas in hierarchy)

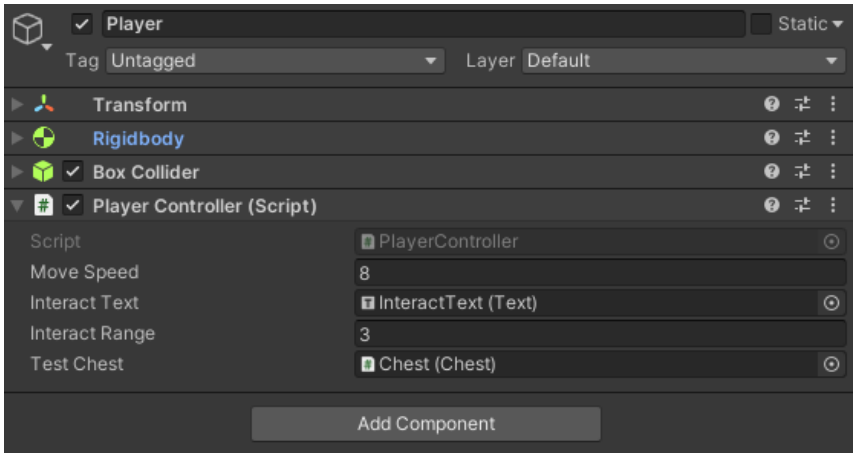
It is disabled by default. When you can interact with a chest, it will become visible so leave it disabled. You can create your own Text element too. Also you should manually update its text when you change the interact key.

5.2.2. Setting up the player

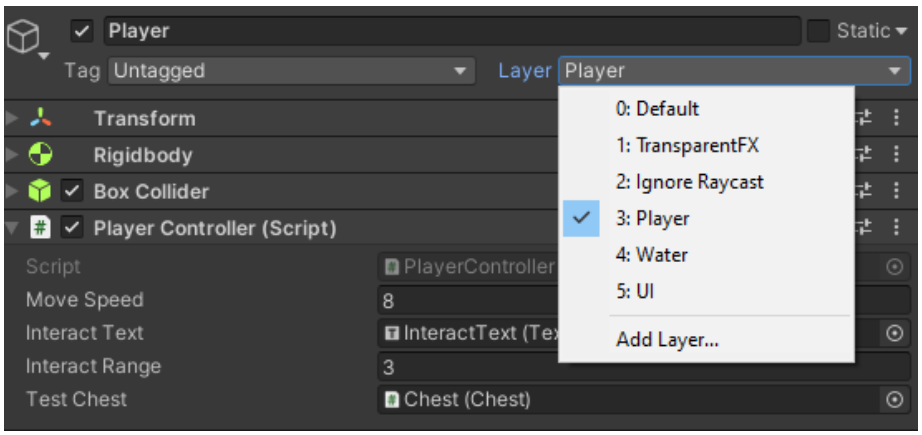
I created a very simple player controller for testing purposes. I do not recommend it for normal use, but you can use it to quickly test the inventory system. See the demo scene.

1) Create an empty game object.
2) Attach rigidbody component to it.
3) Attach "PlayerController" script to it.
4) Drag and drop the "InteractText" object from hierarchy to the "Interact Text" inspector field.

5) Drag and drop the “Chest” object you created from hierarchy to the “Test Chest” inspector field.
6) Set the “Move Speed” and “Interact Range” inspector fields to your liking.



7) Change Player’s layer from Default to “Player”



5.2.3. Setting up the camera

For testing purposes, I used first person camera.

1) Attach the Main Camera to the player.
2) Reset the position to (0,0,0).
3) Attach “CameraController” script to the Main Camera.

5.2.4. Setting up the “TestButtons” prefab

TestButtons is just a Text object that shows which button does what. You can find it inside following directory: /Resources/Prefabs

1) Drag and drop “TestButtons” prefab on Canvas.

You should manually update it when you change these keys.

Everything is ready to go!

6. Operations

6.1. Adding Item

6.1.1. Adding Item to the Player

There are 2 ways to add item to player:

1) Use game manager to add item to player. This function returns the remaining amount. See the screenshots below.

```
public int GiveItemToPlayer(ItemData itemToGive, int amount)
```

```
GameManager.Instance.GiveItemToPlayer(Inventory.Instance.ItemDatabase.FindItem("Diamond Ore"), 1);
```

2) Use Inventory script to add item to player. This function returns the remaining amount too. See the screenshots below.

```
public virtual int AddItemDull(ItemData itemToAdd, int amountToAdd)
```

```
Inventory.Instance.AddItemDull(Inventory.Instance.ItemDatabase.FindItem("Diamond Ore"), 150);
```

6.1.2. Adding Item to a Chest

Use the "Chest" script of the Chest object that you want to add item. testChest is a Chest script. See the screenshot below.

```
public virtual int AddItemDull(ItemData itemToAdd, int amountToAdd)
```

```
testChest.AddItemDull(testChest.ItemDatabase.FindItem("Silver Pickaxe"), 1);
```

6.2. Opening Inventory

Call the following function with 'true' parameter to open inventory.

```
InventoryWindowRoot.Instance.ShowInventory(true);
```

6.3. Closing Inventory

There are 2 ways to close the inventory:

1) Click on the X button at upper right corner of the inventory window.
2) Call the following function with 'false' parameter to close inventory.

```
InventoryWindowRoot.Instance.ShowInventory(false);
```

6.4. Opening Chest

You can call the following function to interact with a chest but you will need a code piece for it to work correctly. chest is the target chest's "Chest" script.

```
chest.Interact();
```

You need to obtain Chest function using Raycasts. See the following piece of code.

```
float interactRange;
Chest chest;
Unity Message | 0 references
void Update()
{
    Ray ray = new Ray(Camera.main.transform.position, Camera.main.transform.forward);
    RaycastHit hit;
    Physics.Raycast(ray, out hit);
    if ((hit.collider != null) && hit.collider.GetComponent<Chest>() && (hit.distance < interactRange))
    {
        interactText.gameObject.SetActive(true);
        if (Input.GetButtonDown("InteractWithChest"))
        {
            chest = hit.collider.GetComponent<Chest>();
            chest.Interact();
        }
    }
    else
    {
        interactText.gameObject.SetActive(false);
    }
}
```

6.5. Closing Chest

There are 2 ways to close a chest:

1) Click on the X button at the upper right corner of chest window.
2) Call the following function with 'false' parameter to close chest.

```
ChestWindowRoot.Instance.ShowChest(false);
```

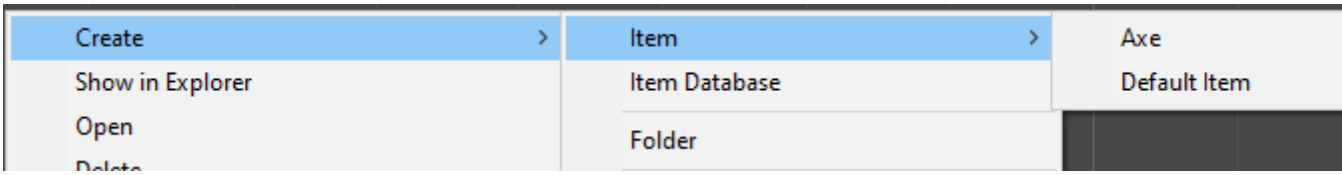
6.6. Use/Equip an Item

When inventory is open you can use or equip an item -depending on its category (consumable, can be equipped, etc.)- by 4 ways:

1) Double click to the item.
2) Right click to the item.
3) Drag and drop the item to the equipment slot.
4) Select the item by clicking it and click to the Use/Equip button.

6.7. Creating a new item

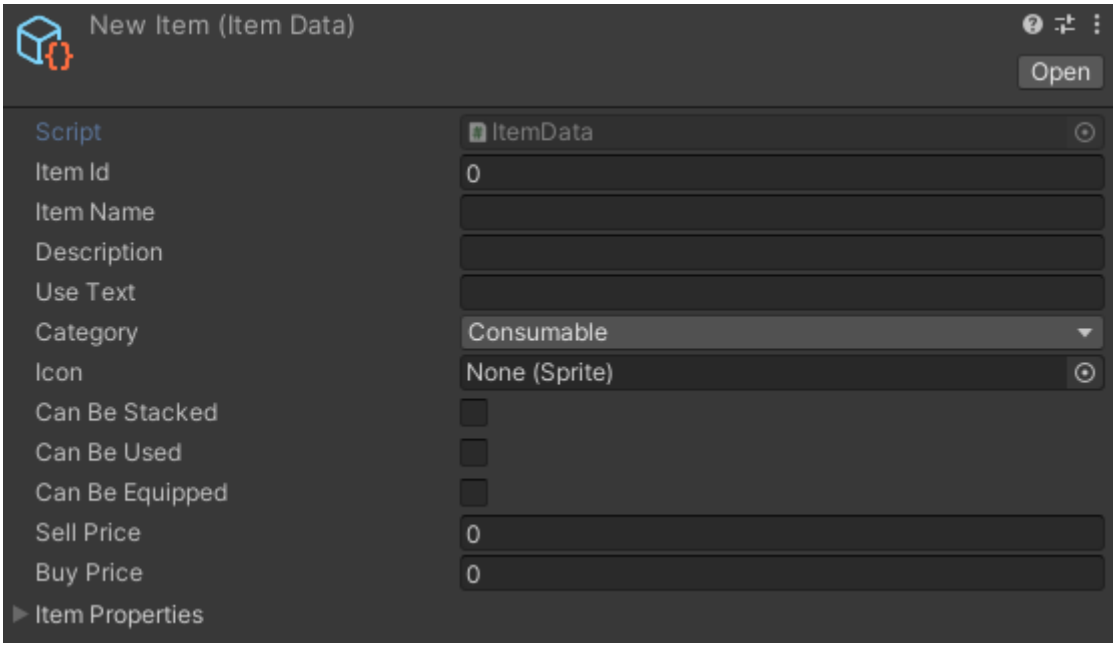
1) Right click to the project panel->Create->Item->Default Item



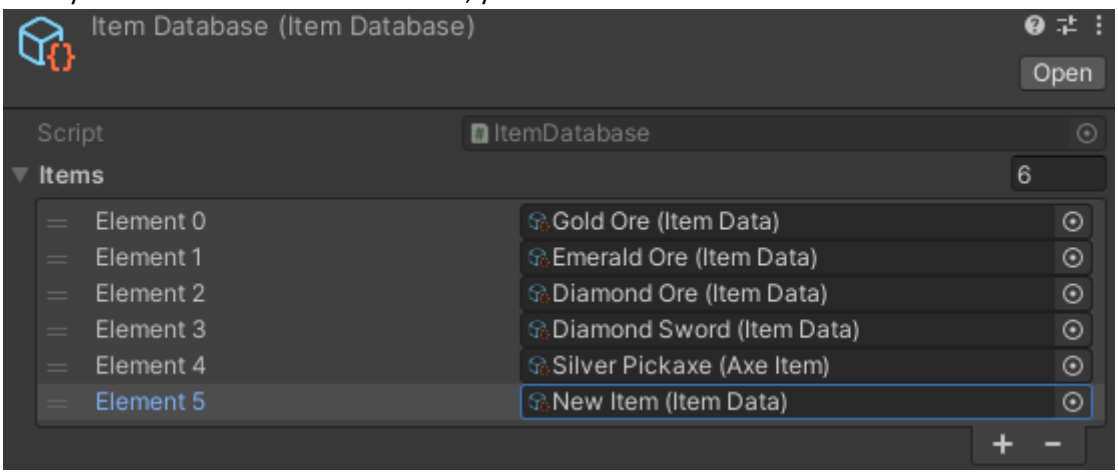
Also, you can create your own custom items by inheriting from ItemData class. See the AxeItem class below for example.

```
[CreateAssetMenu(fileName = "New Axe", menuName = "Item/Axe")]
Unity Script | 0 references
public class AxeItem : ItemData
{
    2 references
    public override void OnUse()
    {
        Debug.Log("Axe Equipped!");
    }
}
```

2) After you create the item, you must fill its information from inspector.



3) You must keep track of item ids manually; I plan to automatize it in the future releases.
4) Item Name: When you search an item in ItemDatabase, you search it by either id or name, try to name it meaningfully so you will not forget it when you try to search.
5) Description: Description text to be showed in inventory.
6) Use Text: When you select an item in inventory, if it is usable, this text will appear as use button. If it is not usable, you can leave it empty.
7) Category: Category of the item. You can not drop Quest items.
8) Icon: Item icon to be showed in inventory.
9) Can Be Stacked: Self-explanatory.
10) Can Be Used: Can we consume or equip this item. This is a bit confusing with Can Be Equipped, I will try to simplify it in the future releases.
11) Can Be Equipped: Can we equip this item.
12) Sell Price, Buy Price: When I implement the market subsystem in the future, these two will make sense. Right now, they are not used.
13) Item Properties: You can add properties like Damage or Armor to the items. Not fully implemented yet.
14) After you create an item and fill its fields, you must add it to the ItemDatabase asset.



7. Roadmap

7.1. Two-way Transaction Between Inventory and Chest

Currently, you can get items from chest, but you cannot give an item from inventory to chest using UI.

7.2. Equipment Subsystem and User Stats

There are equipment slots but they are not functioning and do not modify any user stats.

7.3. Market Subsystem

I want to create a market interface and add currency for trading.

7.4. Sounds, Animations

I will add sounds and animations to make it look nice.

8. Conclusion

It is a hobby project so I will improve it when I find time. Feel free to use it, modify it. I hope you find it useful.