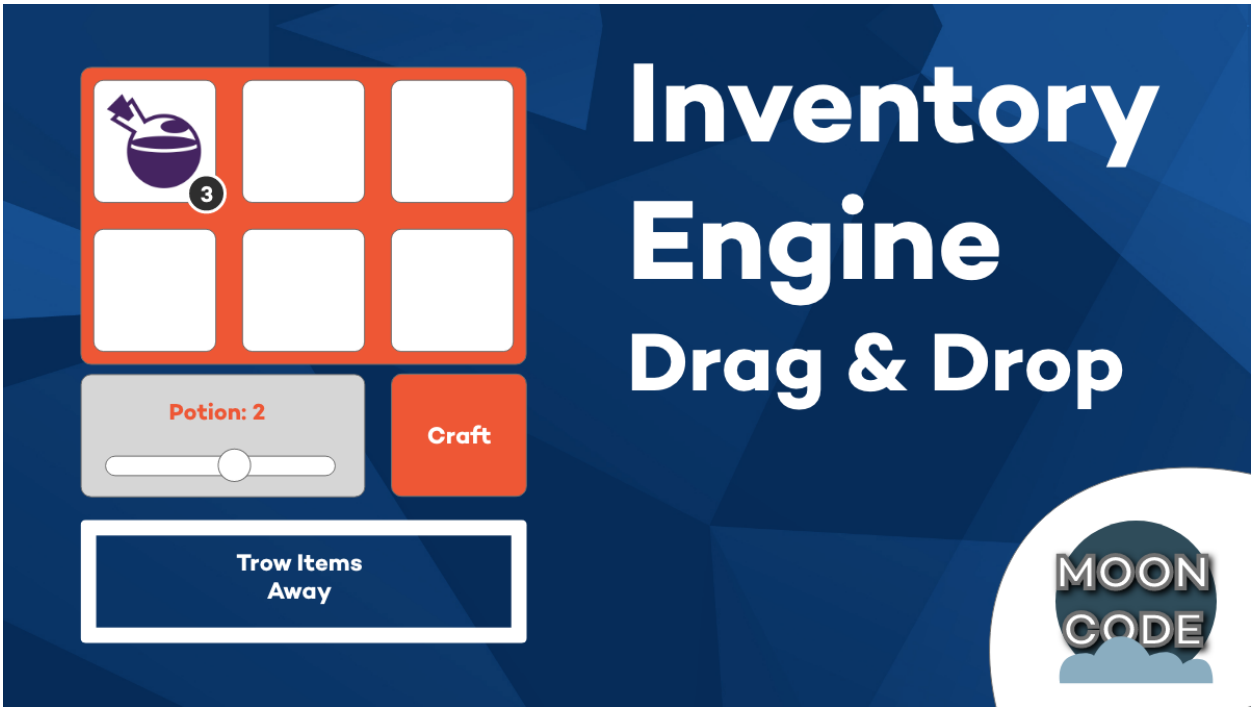
Inventory Engine

Documentation



Introduction:

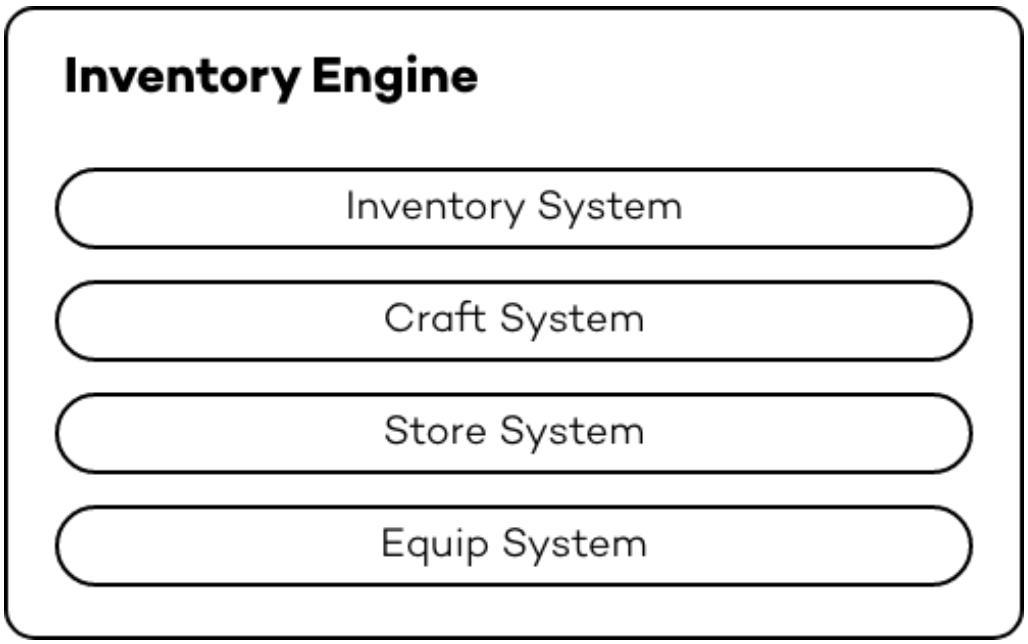
Thanks for purchasing Inventory Engine!

The Inventory Engine is very simple in use and fast in learning. It uses JSON files as a database, so you can easily add, change and remove any parameters. It is divided on 4 parts (Inventory System, Craft System, Storage System, Equip System), so you can easily add/remove inventory features you want. All scripts are written on C# and have comments.

Feature list:

- Drag and drop items
- Collect items
- Throw items away
- Stack and split items
- Swap items
- Craft items
- Equip items
- Store items in chests
- See item tips
- Create items
- Create recipes for crafting
- Minimal demo scene
- Open/Close the inventory interface

About:



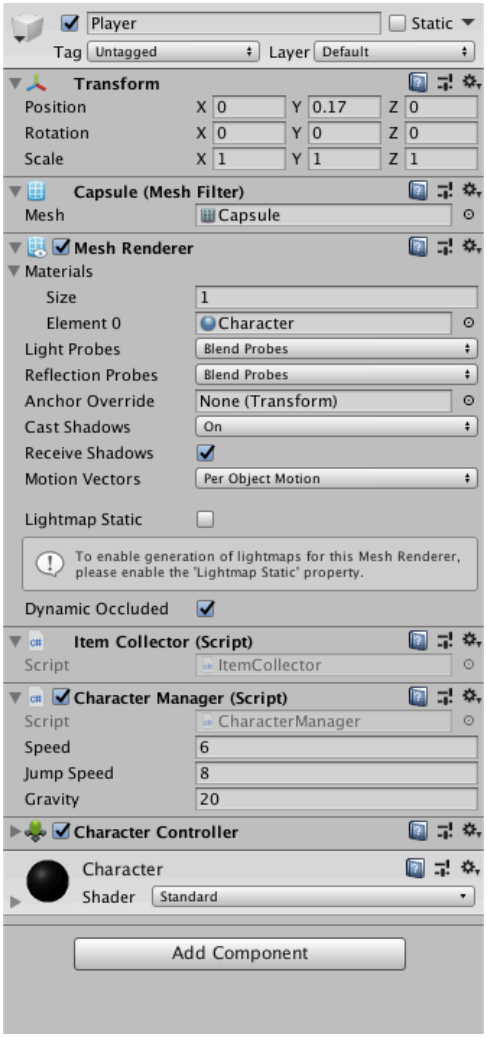
Inventory Engine is divided on 4 parts:

1) Inventory System – lets player drag and drop items, throw them away, stack, swap, collect, split and etc.
2) Craft System – let player craft items from another items.
3) Store System – lets player store items in other GameObjects like chest.
4) Equip System – let player read items' parameters and set player stats.

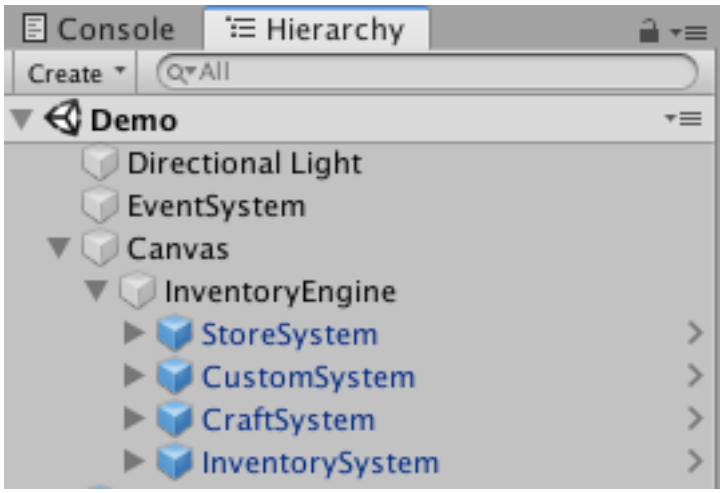
Inventory System is the main part of the Inventory Engine, as it is responsible for drag and drop, throw away, stack, swap, collect and split items. Other systems use methods from Inventory System to interact with it.

How to setup Inventory Engine:

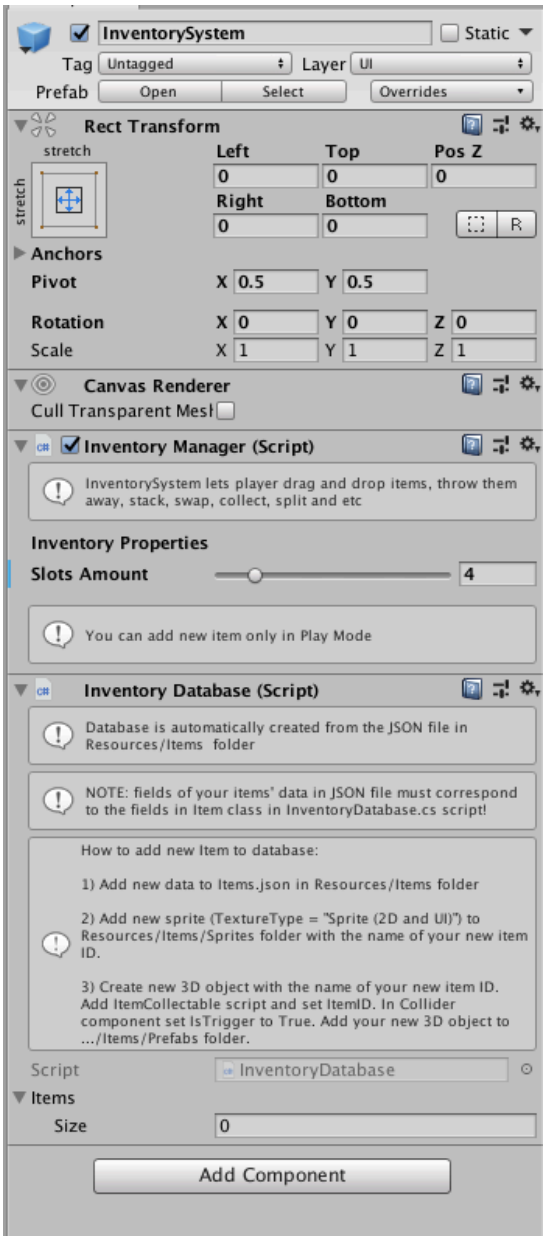
1) Create player
    a. Create new 3D GameObject and attach Main Camera to it
    b. Add “ItemCollector” script.
    c. *Additionally, add ”CharacterManager” script and “Character Controller” component to control your character. You can find the Character prefab in folder: Resources/Prefabs.



2) Add “InventorySystem” to Canvas in Hierarchy from Resources/Prefabs folder. Add other systems, such as CraftSystem, if you need them.



3) Set Inventory slots amount



4) Add (see “How to add new Item to database”) and place items from Resources/Items/Prefabs folder in your game.
5) Add (see “How to add new Storage”) and place storage (chest) in your game. (You can find Storage prefab in Resources/Prefabs folder). Don’t forget to add “StoreSystem” to Canvas in Hierarchy from Resources/Prefabs folder.

How to add new Item to database:

1) Add new data to Items.json in Resources/Items folder

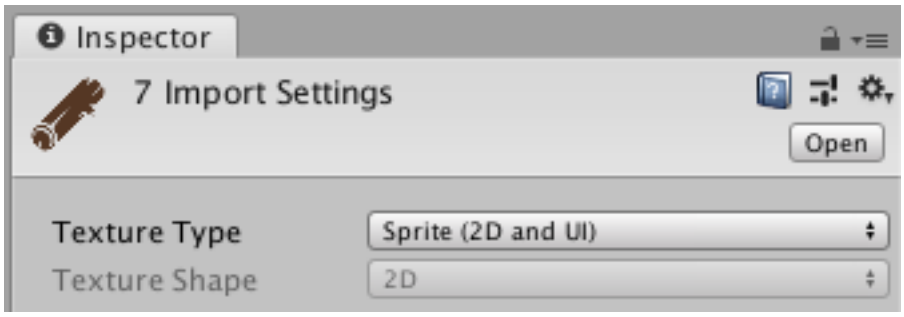
```
51    {
52      "id": 7,
53      "name": "Wood",
54      "price": 1,
55      "power": 0,
56      "stackable": true
57    }
58  ]
```

Note that fields of your items' data in JSON file must correspond to the fields in “Item” class in InventoryDatabase.cs script:

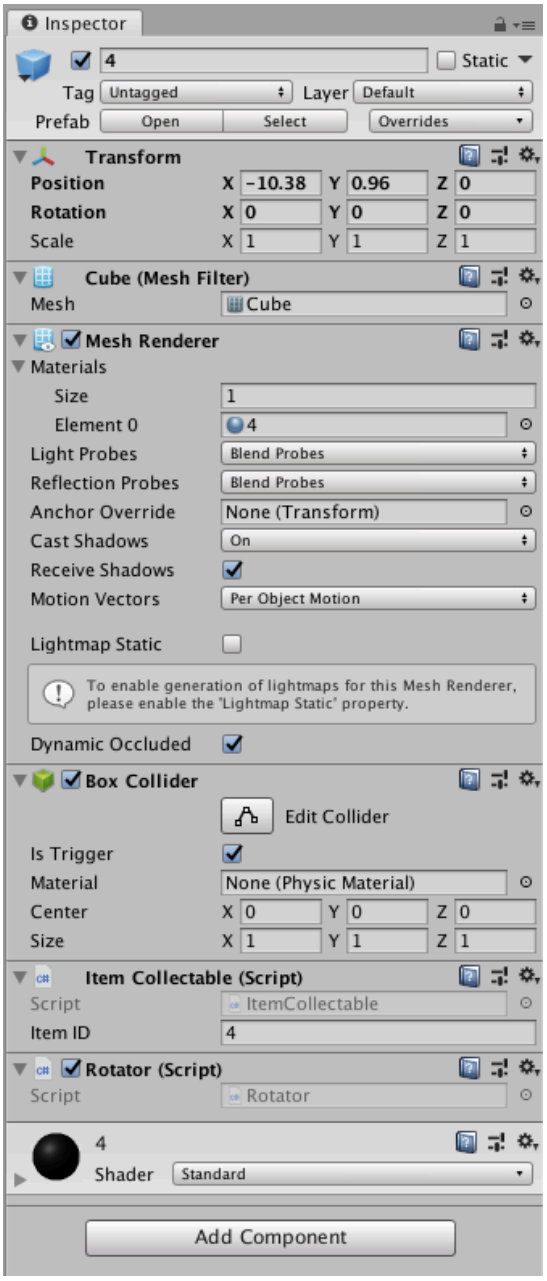
```
public class Item
{
    public int ID;
    public string Name;
    public int Price;
    public int Power;
    public bool Stackable;
```

2) Add new sprite to Resources/Items/Sprites folder with the name of your new item ID.

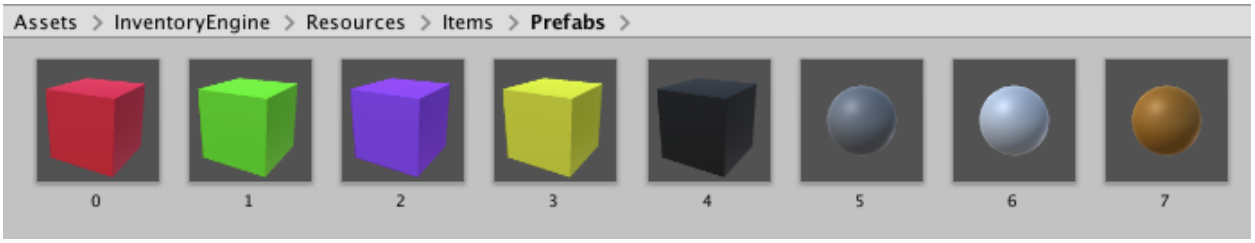
Don’t forget to set TextureType = "Sprite (2D and UI)":



3) Create new 3D GameObject with the name of your new item ID. Add “ItemCollectable“ script and set ItemID. In Collider component set IsTrigger to True.



4) Add your new 3D GameObject to Resources/Items/Prefabs folder.



How to add new Recipe to database:

1) Add new data to Recipes.json in Resources/Recipes folder

```
30    {
31      "id": 2,
32      "itemID": 1,
33      "items": [
34        {
35          "id": 0,
36          "amount": 1
37        },
38        {
39          "id": 6,
40          "amount": 2
41        }
42      ]
43    }
44  ]
```

Note that “itemID” and “id” in “items” from Recipes.json must correspond to items “id” from Items.json.

Note that fields of your recipes' data in JSON file must correspond to the fields in “Recipe” class in CraftDatabase.cs script.

How to add new Storage (chest):

1) Create new 3D GameObject
2) Add “StoreData“ script to this GameObject and set IsTrigger to True in Collider component"
3) Set your new storage initial properties such as title, description and amount of slots.