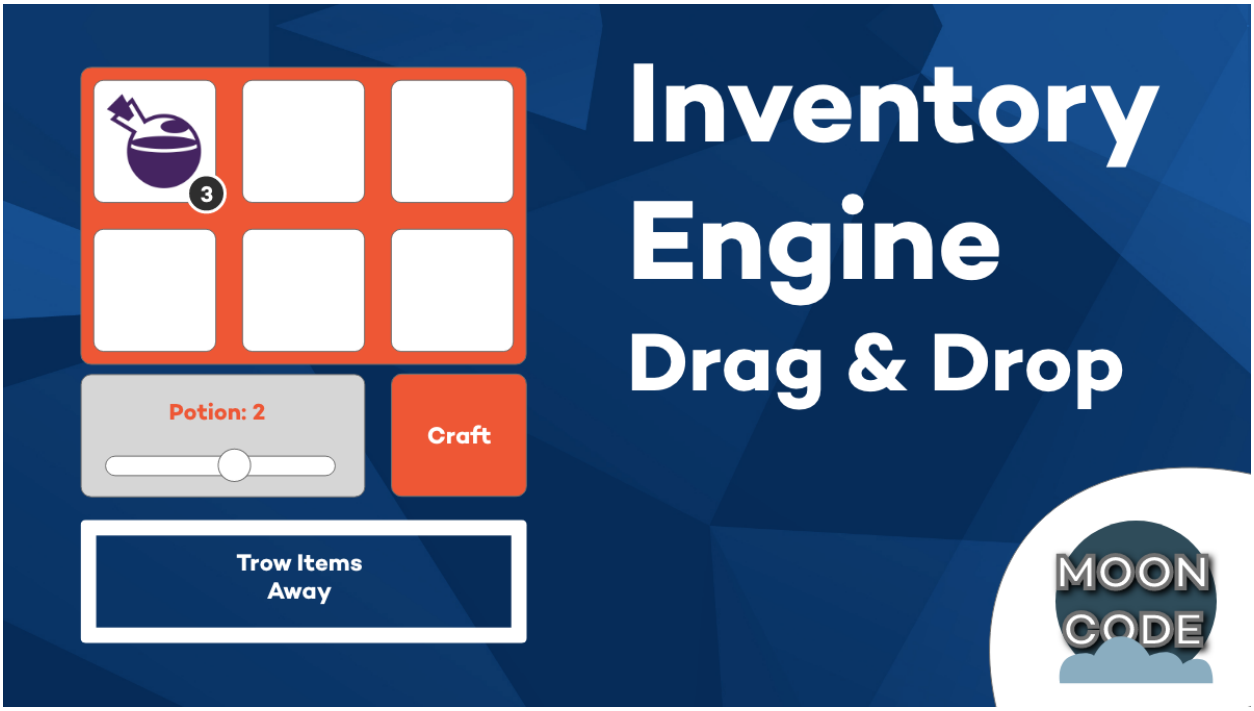
Inventory Engine

Documentation



Introduction:

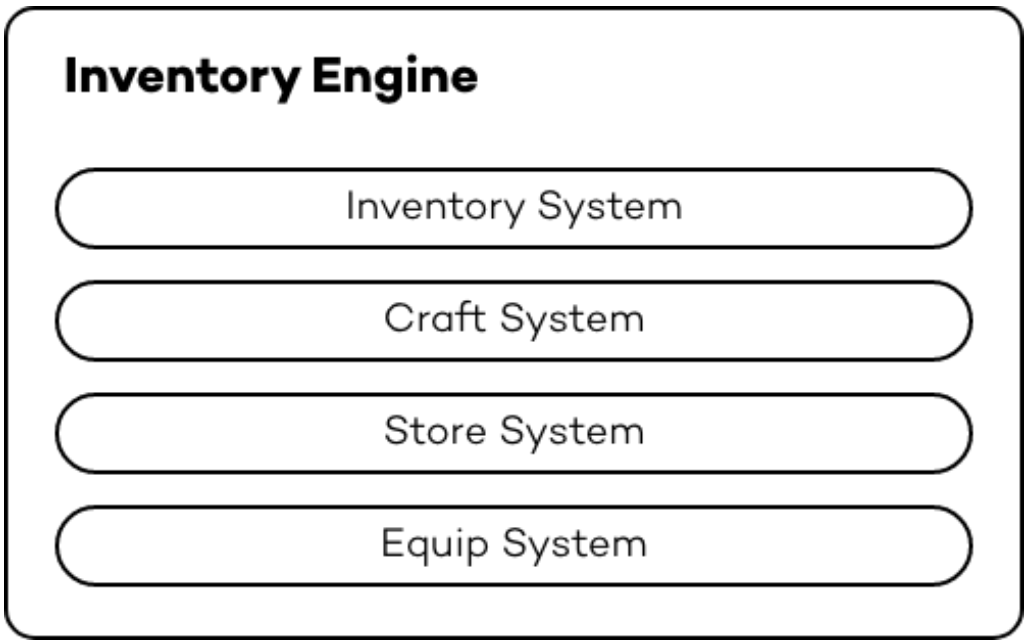
Thanks for purchasing Inventory Engine!

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:



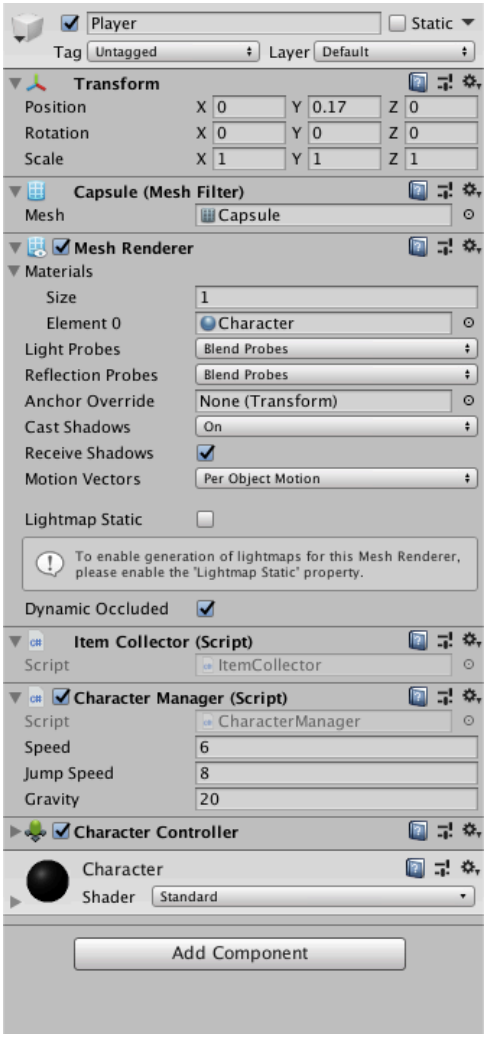
Inventory Engine is divided on 4 parts:

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.

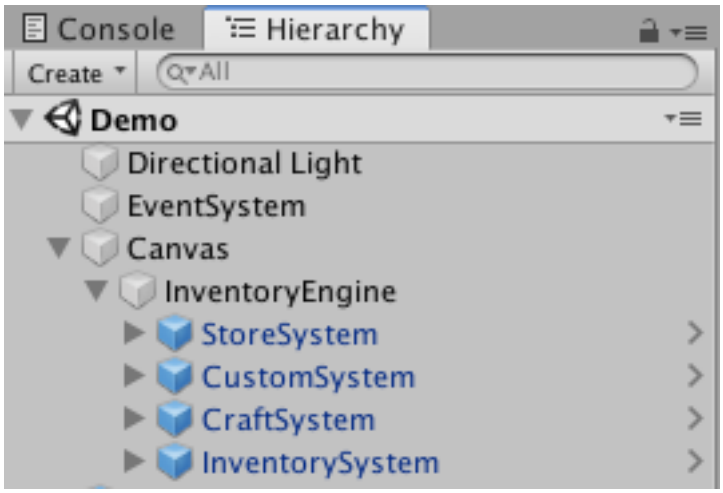
Inventory System is the main part of the Inventory Engine, as it is responsible for drag and drop, throw away, stack, swap, collect and split items. Other systems use methods from Inventory System to interact with it.

How to setup Inventory Engine:

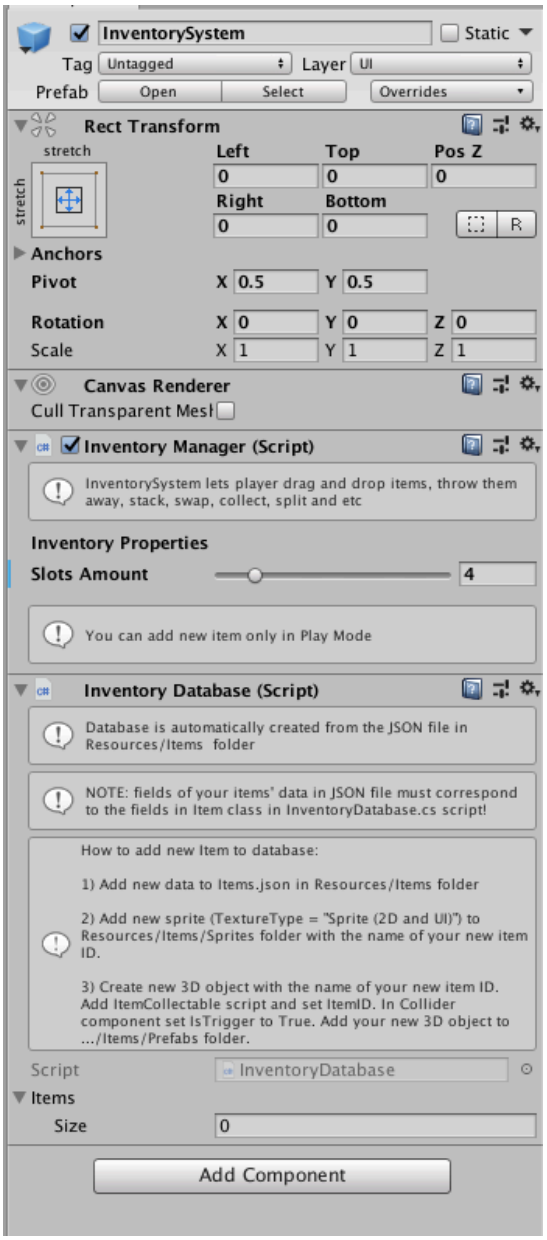
1) Create player
    a. Create new 3D GameObject and attach Main Camera to it
    b. Add “ItemCollector” script.
    c. *Additionally, add ”CharacterManager” script and “Character Controller” component to control your character. You can find the Character prefab in folder: Resources/Prefabs.



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

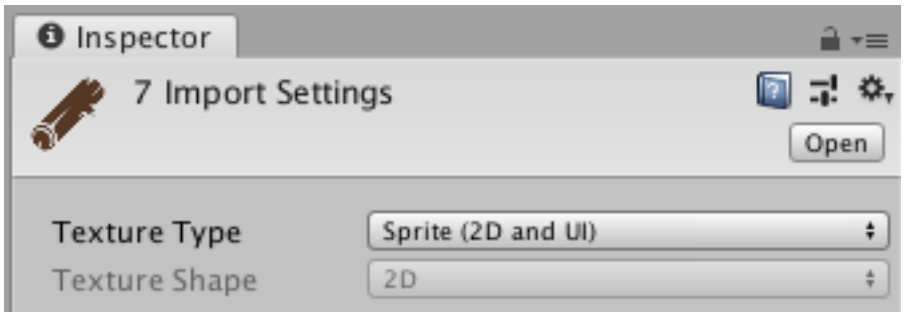
```
51    {
52      "id": 7,
53      "name": "Wood",
54      "price": 1,
55      "power": 0,
56      "stackable": true
57    }
58  ]
```

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

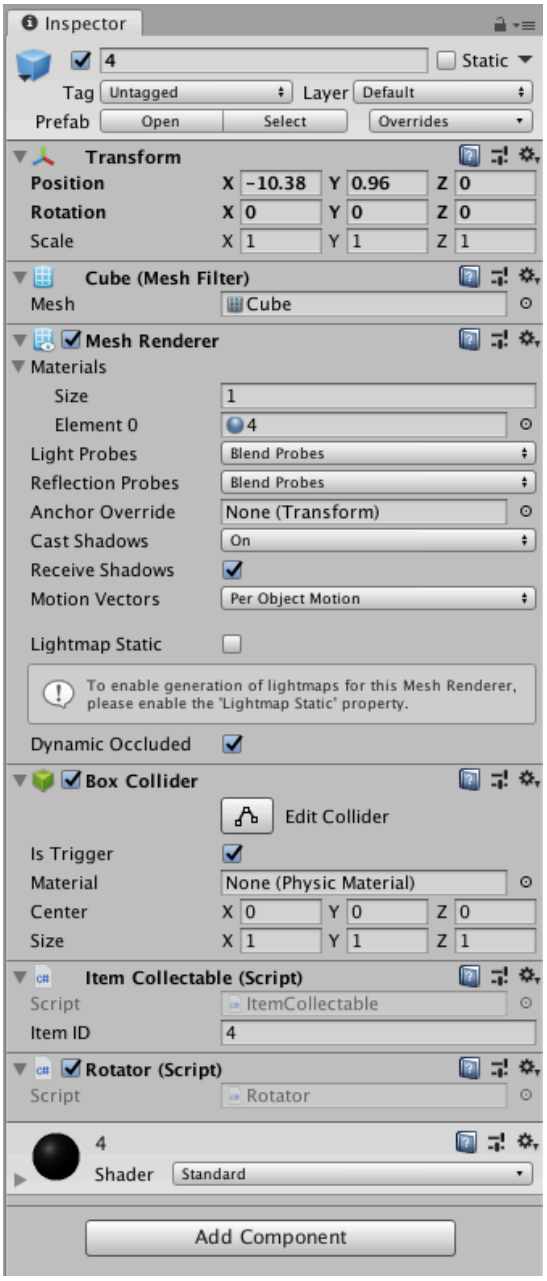
```
public class Item
{
    public int ID;
    public string Name;
    public int Price;
    public int Power;
    public bool Stackable;
```

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

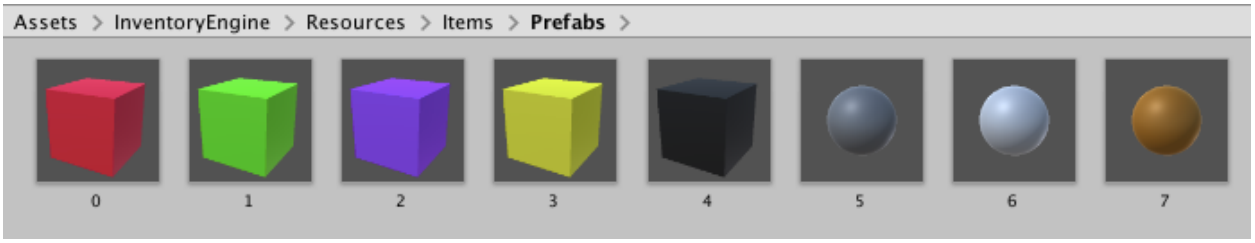
Don’t forget to set TextureType = "Sprite (2D and UI)":



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

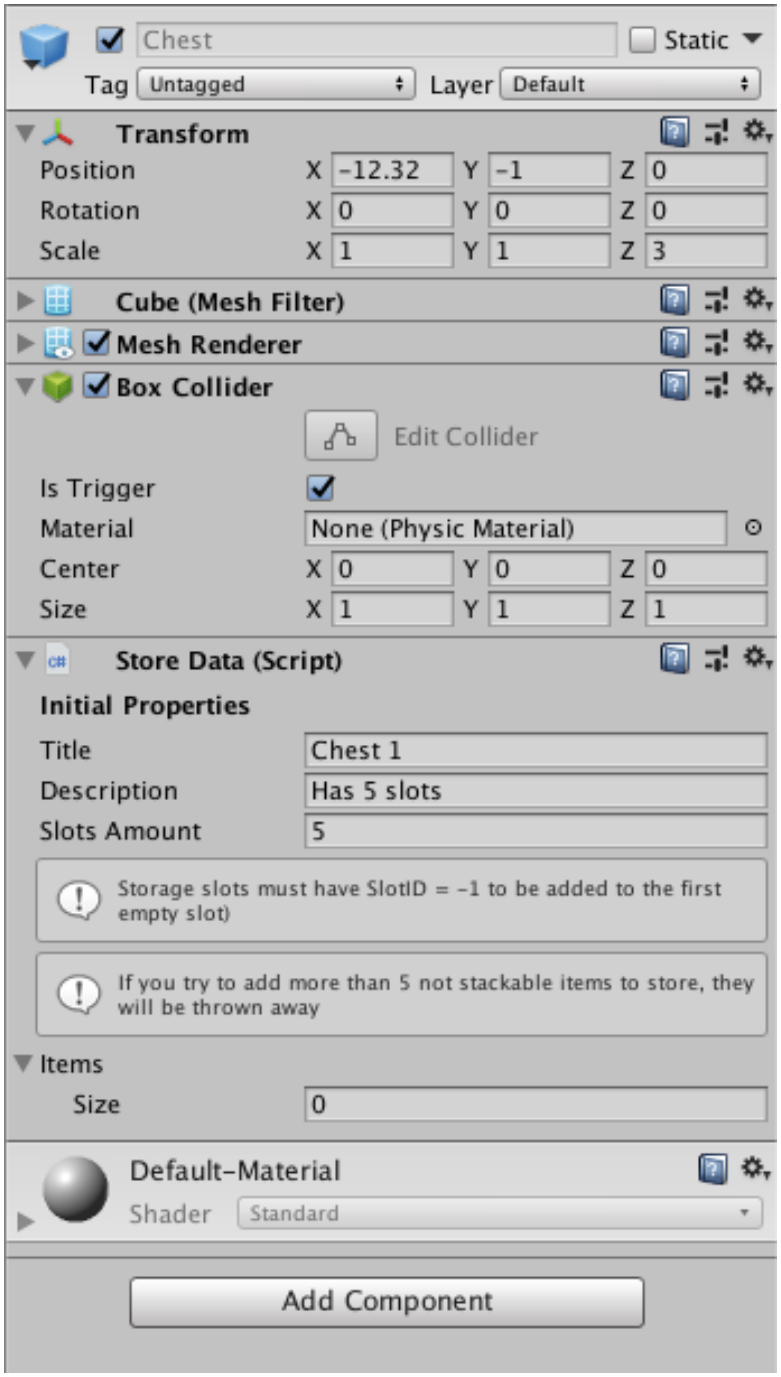
```
30    {
31      "id": 2,
32      "itemID": 1,
33      "items": [
34        {
35          "id": 0,
36          "amount": 1
37        },
38        {
39          "id": 6,
40          "amount": 2
41        }
42      ]
43    }
44  ]
```

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.



Scripts API

Craft System - lets player craft items from other items.

- CraftDatabase – loads and creates recipe database from JSON file.
  - LoadRecipeFromFile - load and parse data from JSON file.
  - FindRecipeByID - get Item by id from database.
- CraftInfo – control CraftSystem info panel.
  - SetInfoText - set text in CraftSystem info panel.
  - Activate - activate CraftSystem panel.
  - Deactivate - deactivate CraftSystem panel.
- CraftManager – create and control CraftSystem.
  - OpenCloseCraft - open or close CraftSystem panel.
  - GetSlotsAmount - return CraftSystem slots amount.
  - CraftItem - check inventory if it has all necessary items and craft chosen item.
  - CanCraft - check inventory if it has all necessary.
  - AddNewSlot - add one slot to CraftSystem panel.
  - SetCraftInfo - set text of CraftSystem info panel.
  - AddCraftSlots - add slots to CraftSystem panel.
  - ClearCraftSlots - remove all children from CraftSystem panel.
  - CreateCraftSlots - fill CraftSystem panel with the new slots.
- CraftOpenButton – set button to open/close CraftSystem panel.
- CraftSlotPlace – set place of CraftSystem slots.
- RecipeData – store recipe data.

Equip System - lets player read items' parameters and set player stats.

- EquipInfo – set equipment info panel.
- EquipManager - create and control EquipSystem.
  - SetEquipInfo - set text in EquipInfo panel.
  - CheckStats - checking EquipSystem slots and read info from items to update stats.
- EquipSlotPlace – set place of EquipSystem slots.

Inventory System - lets player drag and drop items, throw them away, stack, swap, collect, split.

- CharacterManager – additional script to control player movements.
- InventoryDatabase - loads and creates item database from JSON file.
  - LoadItemsFromFile - load and parse items from JSON file.

    - FindItemByID - get item by id from JSON database.
- InventoryItemSplit - control InventorySystem split panel.
    - ChangeSplitText - set text in split info panel.
    - Activate - activate split panel.
    - Deactivate - deactivate split panel.
    - SplitItem - Split item.
    - SplitAmountChange - change amount of items to be splited.
- InventoryItemThrow – set InventorySystem item throw panel.
    - ThrowItemAway - throw item away to the world.
- InventoryItemTip – control InventorySystem tip panel.
    - Activate – activate top panel.
    - Deactivate – deactivate tip panel.
- InventoryManager - create and control InventorySystem.
    - GetSlotByID - search and return slot by ID.
    - GetSlotsAmount - return InventorySystem slots amount.
    - HasItem - check if inventory has one item with specified id.
    - HasAmountItem - check if InventorySystem has items with specified id, amount and slot types.
    - RemoveItem - remove item with specified id from slots with specified types.
    - AddNewItem - searching empty slot in InventorySystem panel and fill it with a new item.
    - AddNewItemToSlot - add item to the chosen slot.
    - ChangeItemSlot - change slot of chosen item.
    - OpenInventory - open InventorySystem panel.
    - CloseInventory - close InventorySystem panel.
    - AddNewSlot - add one slot to InventorySystem panel with certain type.
    - AddSystemSlots - add slots of certain type to InventorySystem panel.
    - ClearSystemSlots - remove all children from InventorySystem panel.
    - CreateSystemSlots - fill InventorySystem panel with the new slots.
    - SaveInventoryInPrefs - save inventory data in player prefs.
    - LoadInventoryFromPrefs - load inventory data in player prefs.
- InventoryOpenButton - set button to open/close InventorySystem panel.
- InventorySlotPlace – set place of InventorySystem slots.

- ItemCollectable - add item to inventory or Equip panel when we collect the prefab with this script.
- ItemCollector – set GameObject which is able to collect items.
- ItemData - store item data and events (drag, drop and etc).
- SlotData - store slot data and events.
    - HasItem - check if slot has any item.
    - GetItemData - get data of slot item.
    - Clear - remove all items from slot.

Store System - lets player store items in other GameObjects like chests.

- StoreData - store storage data and events.
    - SaveStore - save items in storage.
    - LoadStore - fill opened storage with saved items.
    - SaveStoreInPrefs - save storage data in player prefs.
    - LoadStoreFromPrefs - load storage data in player prefs.
- StoreInfo – set storage info panel.
- StoreManager - create and control StoreSystem.
    - CloseStore - close storage panel.
    - OpenStore - open storage panel.
    - SetStoreInfo - set text in storage info panel.
- StoreSlotPlace - set place of StoreSystem slots.