# Inventory System

Since I am working alone and the documentation is still work in progress, it might need some more improvements.
**If you think that something is missing in the documentation, or something is unclear, please contact me under: gamedevibk@gmail.com**
**I am happy to answer any of your questions and also for any feedback so I can improve the documentation and make this asset much easier to use!**

## Table of Content:

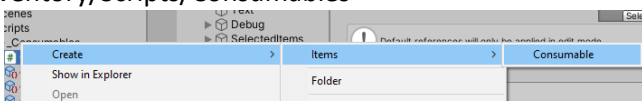


## *1.Overview*

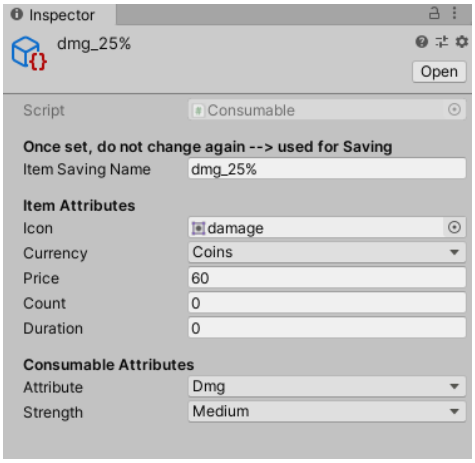
This asset includes 2 shops(auto generated and manually generated) where you can buy items and one inventory that allows the user to see all the items that he owns.
Your Inventory will automatically update with the current items that you have available and/or selected.

## *2.Items*

The used Items are based on the Consumable class which inherits from the Item class.
You can find some preInitializes items in the asset under "SF_Inventory/Scripts/Consumables"
To create new Item, right click somewhere in the project folder and go to "create->Items->Consumable"



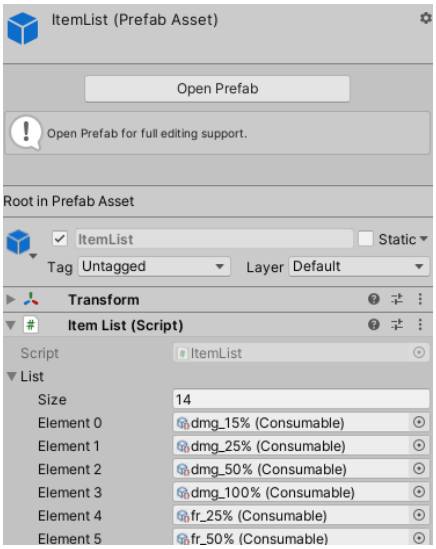

Example:
The created Item should look like the following picture:




- ItemSavingName: used as PlayerPrefs key for saving
- Icon: displayed sprite image for the item
- Currency: choose the currency in which the player have to pay for the item
- Price: define the itemPrice in the choosen currency
- Count: Initial Amount on first game start ( if items are once saved, the saved info will be used to initialize it)
- Duration: defines how long the Item is active (not implemented yet)
- Attribute: define the effected attribute on ItemActivation (Note: User can only select 1 Item of the same Attribute simultaneously. Will be explained later)
- Strength: predefined strengths of effect ( 15|25|50|100)% increasement

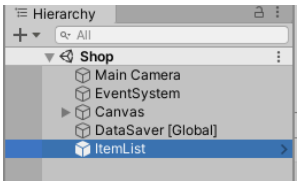
**3.Shop and ItemList**

We will focus on the Auto generated shop here, because the only difference here is that the manually generated shop needs every Item shop button to be manually added to the scrollrect, which can be better adaptable for specific cases, but in general I would use the auto generated shop! For the auto generated shop you only need to add your new created items to the ItemList Prefab, which will be explained now in detail !




The Prefab “ItemList” ( which has the ItemList script attached) has a list of all created items. To add a newly created Item to the shop, you just have to add the new Item to this Prefab.
Because the demo scene has this Prefab attached in the Hierarchy, The shop will automatically update when opened!

NOTE: make sure you add the item directly to the “original” prefab in the Projects folder, so the list will be globally updated on every prefab instance That you are using!



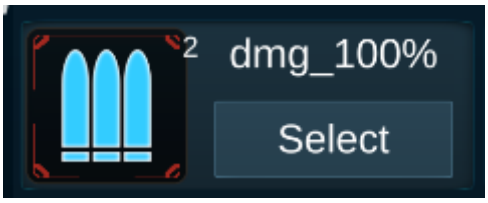

The ItemList script has a static reference to itself, so it can be accessed from every other script. But please make sure to add the Prefab to every scene you want to use the ItemList.
Another possibility is that you just add this Prefab to a GameObject hat is not destroyed on scene-switch, so you don’t need to add it in every scene you want to use it !

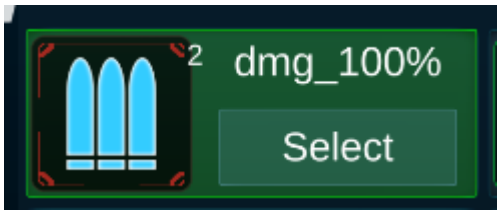
## ***4. Using the Inventory***

When you open the Inventory panel, all the items from the user will be automatically initialized, and updated every time you reopen the inventory.

### *Selecting and using Items*

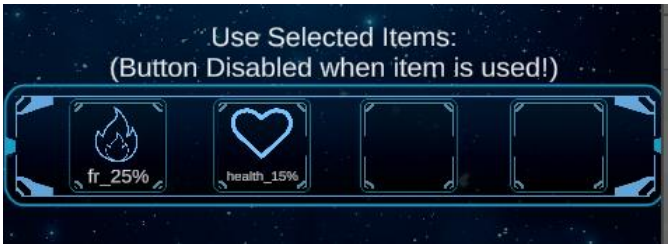
After you successfully bought the items from the shop, they will appear in your inventory. By clicking on the select button in you Inventory, you can select the items that you want to use. Selected items are marked green. The number “2” the top right corner, denotes that you have 2 dmg_100% items available. So if you select your item ( and then also use it later) the number will decrease, because you consumed one of that item. The number “1” will not be shown!




The selected items appears in the bottom panel of the demo scene. This panel can be used in your mission scene, where the selected items should be available for the player.
Simply clicking at the item will consume this item!  It will also disable the consumed item , so the user can not use it twice in one mission.
NOTE: if you consume an item from which you have more than one amount and then restart the mission, the item will still be available, until you deselect it or use the whole available amount!

To reproduce/test this behavior:
Select item from which you have min. 2 of it (e.g. health_15%), after closing Inventory it will appear at the bottom of the scene,

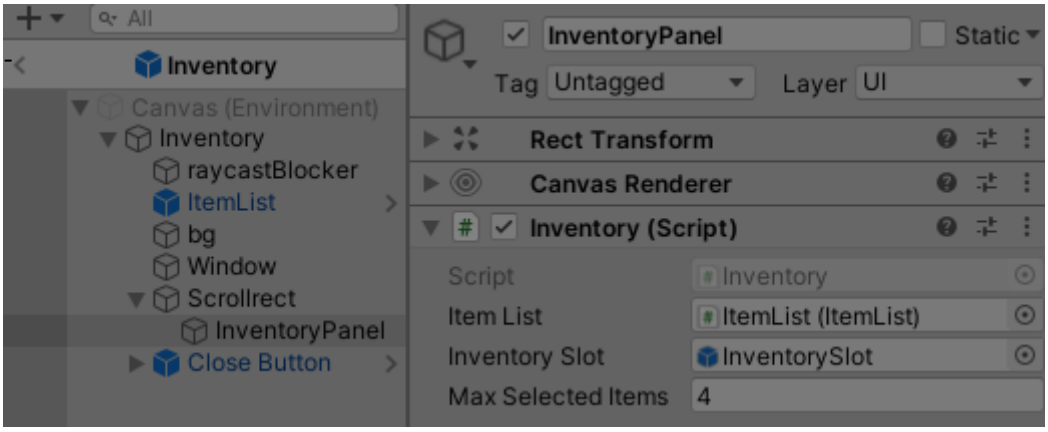



After clicking on health_15% the item is consumed and the item disappears from the panel.


When restarting the scene, and you still have one health_15% item left, it will appear again in this panel at the bottom of the scene.

Maximum selectable items:

To restrict the amount of items that the user can select simultaneously, open the “Inventory” prefab and navigate to the InventoryPanel Gameobject. This object has an Inventory script attached with a property named “Max Selected Items”. By default it is set to 4, but you can adapt it as it fits your needs. If you do not want to restrict this amount, just set it to a very high number, so that the user would never reach this limit!




Note that the inventory has the restriction that only 1 item of the same type can be selected simultaneously. That means if you already selected the Item “Dmg +15%” you can`t select “Dmg +50%” until you de-select the first one!

The reason is simply that the user should not be able to overpower one attribute, like use the dmg increasement item with 15, 25, 50 and 100% at the same time! In This case he would have a dmg increasement of 190%!

## ***5.Adapting item effects:***

Since most of the items should have some effect when used, we should go through the process of adding functionality to the items. The only method you will need to adapt is the Use() method in the Consumable Script.

Per default this script Decreases the item amount, and if it was the last one, it will deselect the item for the inventory.

```
public override void Use()
{
    DecreaseItemAmount();
    if (getItemAmount() <= 0)
    {
        setActivationValue( val: false);
    }
    // TODO: ActivateEffect ...
}
```

This method should also contain the project and item specific behavior.

Example for health item.

```
if (attribute== “health”) {
        healPlayer();
}
```