MULTIPLAYER READY INVENTORY & BUILD SYSTEM

System is mainly designed for for 3D games

- I don’t recommend changing scripts that are in the package, because they will be overwritten on updating to new version / reimporting
- After reloading scripts you have to reenter play mode for package to work correctly
- Right after package was successfully imported you have to reopen unity so script templates works correctly

Table of contents

Exceptions
Scene setup
Inventory (player) setup
    Hotbar
    Items
    Inventory Menu
        Creating Pages
    Item Equipping
Interactions
Collectible items
Harvestable objects
Effects
Inventory Prefabs
Crafting
Shop
Skills
Build system
    Creating buildings
    Interactable buildings
Save and load system
    Custom data
        PlayersData

Exceptions

- Stacking items with durability (maxDurability > 0) is not supported
- You can’t have multiple equipable slots with same equip position on one page
- You can't have more than one same inventory displayers of one type onto one page (example: you can't have two inventory slots displayers on one page) (you can't display one slot multiple times on one page)

Scene setup

- Your scene needs InventoryGameManager assigned on any of your gameobject (that won't be destroyed or disabled in your scene)

Inventory (player) setup

- Your player has to have InventoryCore assigned on himself
- Add Inventory.cs and set up

Hotbar

- Create slot prefab or use already created one (don't forget to assign Slot.cs onto root of the prefab)
- Create new object and use it for hotbar slots and selected background only
- Put Slot prefab under this object as many times as you want and assign them to hotBarSlots in Inventory.cs

Items

- Create new item (Create >> InventorySystem >> Item)
- Set up your new item as you like (if you hover over variable in inspector it will show you tooltip)
- Item categories
    - Create new category (Create >> …) and set it up
    - Assign it into CategoriesHandler (All handlers are located at "Assets/Essentials") or update it (Click get categories button)
    - Now you can select it in any category dropdown
- Item Rarities
    - Basically same as categories, but you are creating ItemRarity and assigning it into ItemRaritiesHandler
- Equip Positions
    - Basically same as categories, more onto this here
- Currencies
    - Again … same as categories …
- Picking up items
    - Add Pickupableitem.cs onto gameobject with colliders
    - More on interactions here

Inventory Menu

- Add InventoryMenu.cs onto your player object and set it up
- Setup button spawning

Creating Pages

- Create new object (child of assigned inventory) and assign InventoryPage.cs
- Now you can add any InventoryPageContent under this object
- You can have only one of each SlotsDisplayer type on one page !

Item Equipping

- Create EquipPosition scriptable object and assign it into handler
- Add new item into EquipTransforms array in Inventory.cs and setup
- Now set up SlotsDisplayer for EquipPositions

Interactions

- Add InteractionsHandler.cs onto your player and setup
- Interactable objects
    - Just assign any class onto GameObject with colliders that has implemented IInteractable interface
    - Or you can create new interactable object through create asset menu (C# templates >> NewInteractableObject)

Collectible items

- Creating collectible item
    - Assign CollectibleItemsManager onto InventoryGameManager object and set up
    - Assign CollectibleItemsHolder onto your player and setup
    - Create CollectibleItem scriptable object and set up
    - Assign CollectibleItemHolder onto collectible object and set up

Harvestable objects

- You have to have InteractionsHandler on your Player
- Assign HarvestableObject onto any object

Effects

- Add EffectsHandler onto your player
- Create new script that inherits from Effect.cs and override voids that you need
- Assign new effect into EffectsDatabase

Inventory Prefabs

- Editing native PrefabSpawned
    - Create custom prefabSpawner (C# Templates >> CustomPrefabsSpawner)
    - For updating prefabs that you have created call them directly like this:

```
InventoryPrefabsSpawner.(spawner as CustomSpawner).SpawnCustomPrefab();
```

    - Or create static void and call it like this:

```
CustomSpawner.SpawnCustomPrefab();
```

- Editing PrefabUpdator
    - Same as PrefabSpawner, but instead you work with PrefabUpdator and InventoryPrefabsUpdator (C# Templates >> CustomPrefabsUpdator)

Crafting

- Assign Crafting onto your player
- Create new CraftingRecipe scriptable object and set up
- Now you can use it in CraftingMenu for example

Shop

- Assign Shop onto your player
- Currencies
    - Create new Currency scriptable object and assign it into CurrenciesHandler
    - Tip: use TMP_Sprite to display coin image in text (documentation)
- Now you can use it in ShopMenu for example

Skills

- Assign Skills onto your player
- Create new Skill scriptable object
- Now you can use things like SkillsMenu

Build system

- Assign BuilderManager onto your game manager object
- Assign Builder onto your player

Creating buildings

- Create new prefab of your building
- Assign Building.cs onto your prefab and set it up

Interactable buildings

- Assign ActionBuilding.cs (including classes that inherits from this class) and set it up
- ActionBuildings does not inherits from Building.cs, therefore can’t be used for building placement by itself (you have to add Building.cs onto them as well)

Save and load system

- Assign SaveAndLoadSystem.cs onto your game manager object

Custom data

- Create new class that will hold your data
- Create new class that inherits from DefaultDataLoader or create new via (C# Templates >> CustomDataLoader)

```
namespace InventorySystem.SaveAndLoadSystem_
    public class CustomDataLoader: DefaultDataLoader
    {
        [InitializeOnLoadMethod]
        private static void M()
        {
            DataLoader.loader = new CustomDataLoader();
        }
    }
}
```

- Now you can override GetCustomData and LoadCustomData

PlayersData

- Create C# script that inherits from PlayerData
    - Don’t forget to set class as Serializable
    - Don’t forget to specify base constructor (Check demo for more info)
- Override load data and save whatever serializable variable you want
- Override SavePlayer in your custom data loader
- Override LoadPlayer in your custom data loader