INVENTORY SYSTEM GUIDE VERSION 1.1.1

BY ASBJØRN THIRSLUND - BRACKEYS

For exclusive video tutorials on the Inventory System and more, head over to

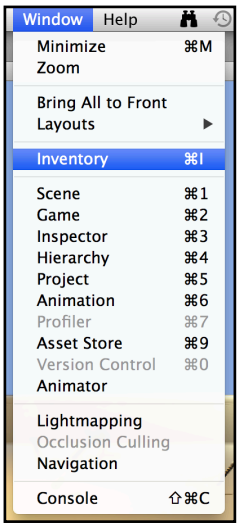
www.brackeys.com/inventory/

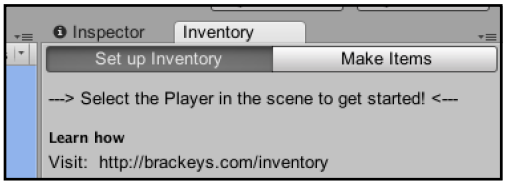
Table of Contents

Set up Inventory

To learn this using a video instead, please visit www.brackeys.com/inventory/ or www.youtube.com/brackeys/.

Setting up the Inventory is easily done using the Inventory window. Find it under "Window" and then "Inventory" or simply press "Ctrl/Cmd + i".



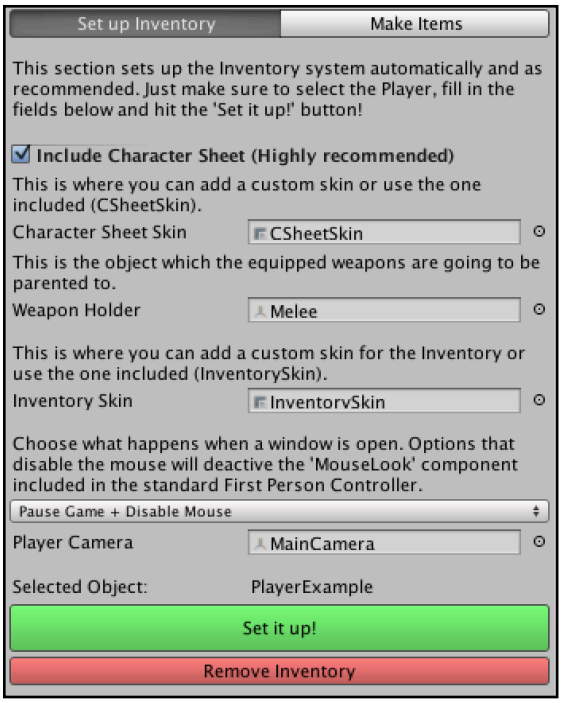


Once you have the Inventory window open it's a good idea to dock it somewhere so it isn't floating on top of everything else. Simply drag on the "Inventory" title and release where you want it to go. I prefer docking it next to the inspector like shown.

You will notice two tabs. "Set up Inventory" and "Make Items". For now we are going to focus on the "Set up Inventory" part. *Items can be made without using the wizard like shown in the "Make Items" chapter.*

Select the Player you want to add the Inventory to. Make sure to select the Player in the scene and not in the project pane. When you select the Player you will notice a bunch of settings appear. You have to fill out all of them before pressing the "Set it up!" button. *Remember that all of these settings can be changed later on and that you can always undo a set up or press the "Remove Inventory" button.*

With the Player selected please follow these steps:



1. Choose if you want to include the Character Window (CSheet) where you can equip weapons and armor to. It is recommend (especially for beginners) to make sure this is marked as true. If not you can skip to bullet point 2.

    1.1. Assign the Character Sheet Skin. This is where you can add a custom skin or just use the one included (CSheetSkin). Simply drag and drop it from the "Project" panel to assign it.

    1.2. Assign the Weapon Holder. This is the object weapons are going to be equipped to. In most cases that would be an empty object inside a hand of a character. Simply drag and drop it from the "Hierarchy" to assign it.

2. Assign the Inventory Skin. This is where you can add a custom skin or just use the one included (InventorySkin). Simply drag and drop it from the "Project" panel to assign it.

3. Choose what happens when a window from the Inventory System is open. We can choose to pause the game which will set the Time.timeScale to 0 and/or we can choose to disable the MouseLook component included in the standard First Person Controller which will make sure that the camera won't rotate. Unless you choose "Keep Playing" the Inventory System will also automatically lock/unlock the mouse cursor. All of this can of course be changed later.

    3.1. Skip this step if you chose "Keep Playing" or "Pause Game" above. This is where we assign the Camera so we can disable the MouseLook component attached to it. Simply drag it in from the "Hierarchy".

4. Press "Set it up!". Check the console for error messages telling you what do to (if any).

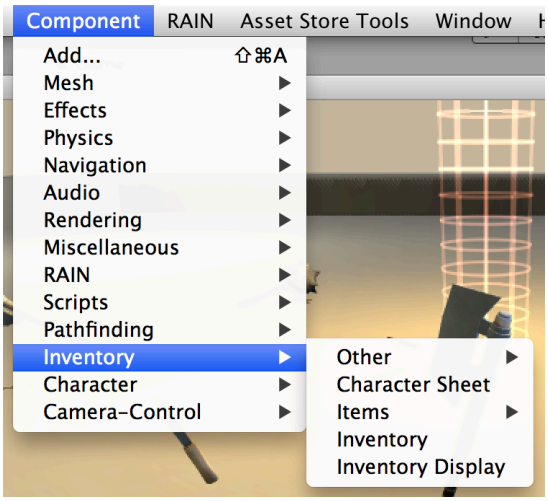
Congratulations! You have just set up your Inventory System! Now you should be able to press play without getting any error messages (make sure to check the console window) and press "i" to open up the Inventory and Character windows.

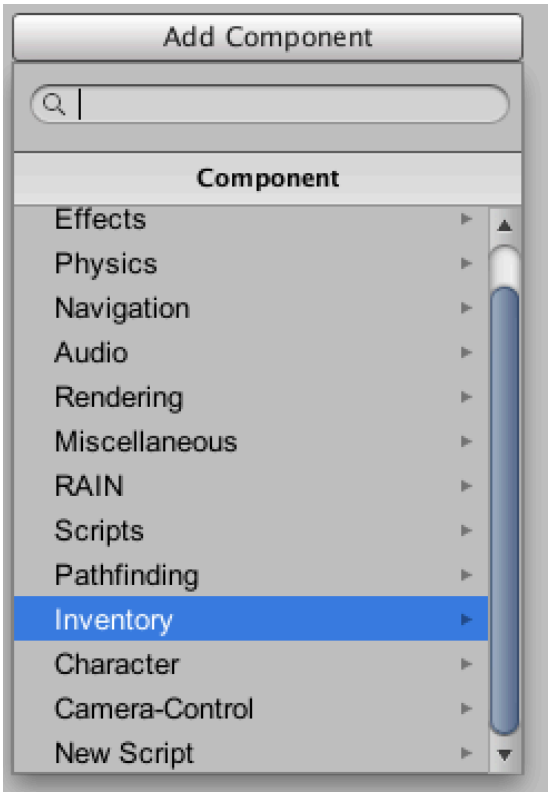
The Character Sheet is set up with three "Weapon" slots which allows the Player to assign weapons to them and then quickly switch between them "in game". This could be achieved using the WeaponSwitch script.

The window texture leaves room on the right hand side for displaying stats or other useful information to the Player.

Now it is time to tweak settings, start creating Items and so on. **Remember that all Inventory Components can be found under "Component" and then "Inventory".**

Side note: Place the "Gizmos" folder under Assets so you can quickly find things in your scene.

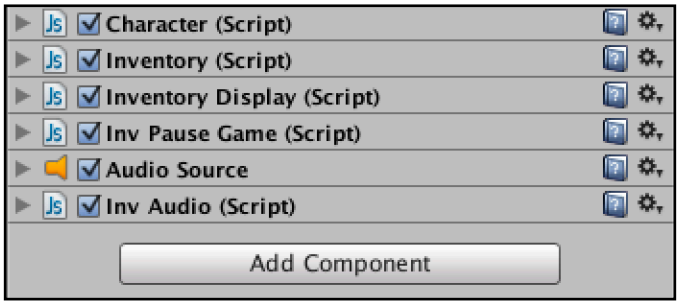




Configuring the system

Understanding the basic components

The very base of the Inventory is split up into three scripts that all work closely together (*found under Inventory/Scripts*). They are all placed on an object called "Inventory" which is automatically created and parented to the Player when setting up the Inventory (see the chapter above).



- **Inventory** - This is the component that stores the items in the Inventory.

- **InventoryDisplay** - This is the component that show and configures the GUI (graphical user interface) for the Inventory.

- **Character** - This is the component that sets up and show the Character Window where items can be equipped to.

For a list of functions scroll down to "Programming for the system".

The rest of the components are something you can choose to include or not (*found under Inventory/Scripts/Other*).

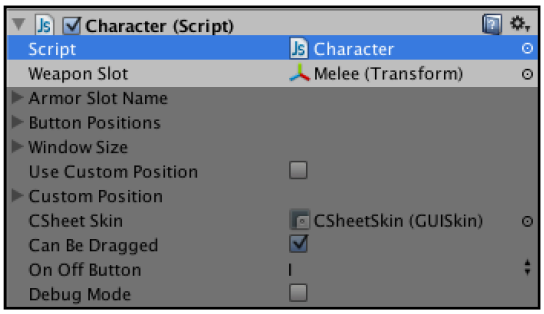
- **InvPauseGame** - This is the script that handles what happens when a window is open. It is included unless "Keep Playing" is chosen when setting up the Inventory.

- **InvAudio** - This is the component that plays sounds on events. The script itself doesn't register anything. Instead the 'play-sound-functions' are called using gameObject.SendMessage from the three base components. This component automatically adds an Audio Source when attached.

Tweaking settings

Now that the base of Inventory System is set up it's a good idea to tweak a few things to make it match your game. I recommend you wait with making big changes, like messing with the GUI, until you have set up some items to test it with. *To learn how to set up items head over to the "Make Items" chapter.* To configure a part of the Inventory you simply find the correct component (using the guide above) and adjust the different variables to your liking. If you are unsure about what a variable does you can simply open up the script, find it at the top and read the comment next to it. All scripts are fully commented out.

Example:




```
var WeaponSlot : Transform; //This is where the Weapons are going to go (be parented to
private var ArmorSlot : Item[]; //This is the built in Array that stores the Items equi
var ArmorSlotName : String[]; //This determines how many slots the character has (Head,
var buttonPositions : Rect[]; //This list will contain where all buttons, equipped or n
var windowSize : Vector2 = Vector2(375,300); //The size of the character window.
var useCustomPosition = false; //Do we want to use the customPosition variable to defin
var customPosition : Vector2 = Vector2 (70, 70); //The custom position of the Character
var cSheetSkin : GUISkin; //This is where you can add a custom GUI skin or use the one
var canBeDragged = true; //Can the Character window be dragged?
var onOffButton : KeyCode = KeyCode.I; //The key to toggle the Character window on and
var DebugMode = false; //If this is enabled, debug.logs will print out information when
```

To learn more head over to www.brackeys.com/inventory/.

Make Items

When creating Items it is important that instances of the same Items are set up in the same way. If for example we create a health potion and duplicate it 4 times it is very important that they all share the same properties such as "Max Stack" or they won't work properly. Therefore it is recommended to make Items into Prefabs so changes can quickly be applied across all instances in the scene.

Understanding the Item components

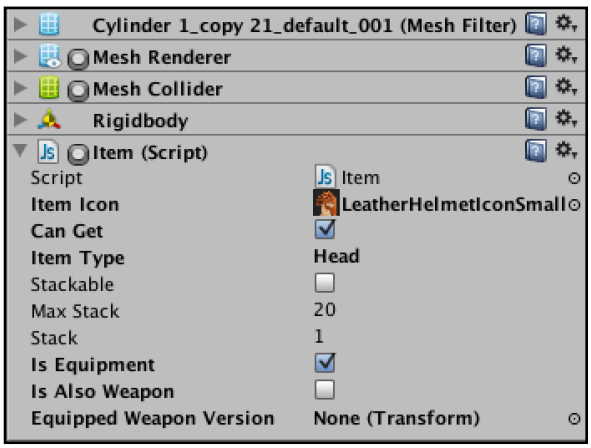
These should all be attached to the Item if you choose to include them. They can be found under *Inventory/Scripts/Items*.

- **Item** - This is the core component that stores information about the Item and communicates with the Inventory. This is only component you need to have attached in order for the Item to work.
- **ItemEffect** - This is the component that should be applied to non-equipment/non-weapon Items. It handles the deletion of the Item on use and allows you to insert custom code (such as a health potion that should add HP to the Player).
- **EquipmentEffect** - This is the component that allows you to add custom code to equipment and weapon Items. This could be useful for implementing stats or magical effects or changing a characters texture to show that the Item is equipped. It can be used for OnEquip events and WhileEquipped effects.
- **FirstPersonPickUp** - This is the component that you attach if you want to pick up the Item by standing close to it and then pressing a button. If this component is not attached the Item will be picked up by clicking on it.

Creating equipment Items

Equipment Items are the easiest to set up.

1. Drag your model into the scene.
2. (Optional if using FirstPersonPickUp) - Attach a collider (remember to check "Convex" if you want a mesh collider to interact with other mesh colliders).
3. (Optional) - Attach a Rigidbody to make the Item follow the laws of physics.
4. Attach the Item component.
    4.1. Fill out the variables. Remember that the Item Type should match the name of the slot the Item should be equipped to. For example if you are creating a Helmet you would make the Item Type "Head".



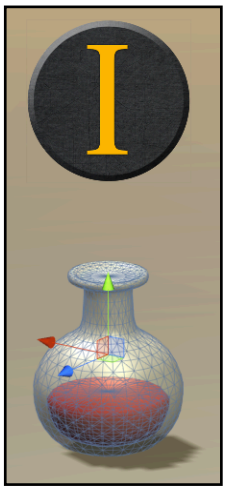
4.2. Make sure to check “IsEquipment” and uncheck “IsAlsoWeapon”.

4.3. The “EquippedWeaponVersion” variable can be left empty.

5. (Optional) - Attach an EquipmentEffect and a FirstPersonPickUp component.

6. (Optional) - Drag the Item down to the project pane to make a Prefab.

Creating consumable Items (non-equipment)



Consumables are different because they have nothing to do with the Character window. They just get used when pressed.

1. Drag your model into the scene.

2. (Optional if using FirstPersonPickUp) - Attach a collider (remember to check “Convex” if you want a mesh collider to interact with other mesh colliders).

3. (Optional) - Attach a Rigidbody to make the Item follow the laws of physics.

4. Attach the Item component.

   4.1. Fill out the variable information and make sure that both “IsEquipment” and “IsAlsoWeapon” are set to false. The “EquippedWeaponVersion” variable can be left empty.

5. Attach the ItemEffect component.

   5.1. Open up the ItemEffect script and insert “effect-code”. “Effect-code” could be calling a “Buff” function or an “AddHealth” function etc. It’s completely up to you.

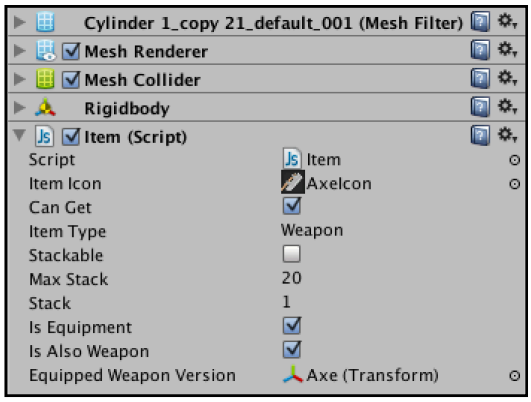
```
24  //This is called when the object should be used.
25  function UseEffect ()
26  {
27      Debug.LogWarning("<INSERT CUSTOM ACTION HERE>"); //INSERT CUSTOM CODE HERE!
28
```

6. (Optional) - Drag the Item down to the project pane to make a Prefab.

Creating weapon Items

Weapon Items are the most difficult to create. This is because you need to create two versions.

- **The Item version** - This is much like the Items above. We use this until we pick up the Item.



   1. Drag your model into the scene.

   2. (Optional if using “FirstPersonPickUp) - Attach a collider (remember to check “Convex” if you want a mesh collider to interact with other mesh colliders).

   3. (Optional) - Attach a Rigidbody to make the weapon follow the laws of physics.

   4. Attach the Item component and fill out the variables.

5. Make sure that “IsEquipment” and “IsAlsoWeapon” are set to true.

6. Now it’s time to assign the “EquippedWeaponVersion” variable by dragging the Weapon version from the project panel but first we have to make the equipped weapon version.

- **The Weapon version** - This is a very scraped version of the Item without many of the components. We use this when equipping the Weapon because then we don’t need components like Rigidbody.

1. Duplicate the Item version.

2. Rename the duplicate to the name of the Weapon + “Weapon”.

3. Rename the Item version to the name of the Weapon . For example the Weapon version could be “AxeWeapon” and the Item version could be “Axe”.

4. Delete the Collider, the Rigidbody and the Item component on the Weapon version. There should only be a Mesh Renderer and a Mesh Filter left.

5. Drag the Weapon version down to the project pane to make a Prefab.

6. Drag the Item version down to the project panel to make a Prefab.

Now we should have two versions of the Item as prefabs. One called “AxeWeapon” and one called “Axe”. “Axe” has all of the components and “AxeWeapon” has only a few.

Now we can select “Axe” in the Hierarchy and drag “AxeWeapon” from the project panel into the “EquippedWeaponVersion” variable. Delete “AxeWeapon” from the scene.