Quantum Inventory System Documentation

Thank you for acquiring the advanced Quantum Inventory System. In this document, you can find all system's functionalities. If you have any doubts regarding the product, feel free to contact me through email or Patreon to get support. Visit my YouTube Channel for more info.

Content

This package contains: **three scripts** (inventory, item and container), **four prefabs** (inventory, hotbar, default item and slot), **seven materials** (ground, item, document, key, consumable, slot and custom), **custom player** and a **test scene**.

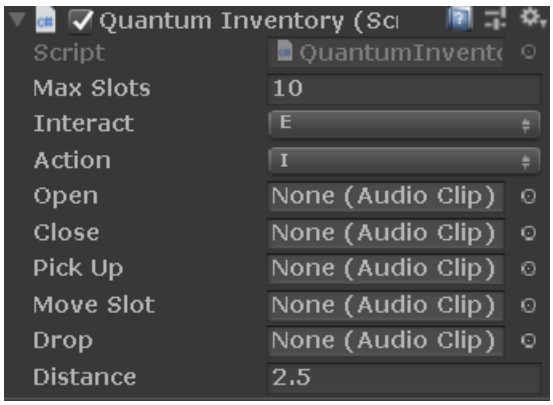
Set Up

In order to integrate the asset into your project, create a new folder under your root directory (Assets) named "Resources". Be careful as the name is really important.

After that, drag the "Core" folder from within your QIS folder into your new Resources folder. Done! The package has been successfully integrated.

Configure

To configure the system, all you need to do is drag and drop the QuantumInventory script to your character. Please be reminded that the asset is optimized for the custom player as I do not know which player you use (either a custom one or the standard asset's characters). Either way, an integration script for Standard Asset's Character will be created soon for you to use.



As you can see, you can set the maximum slots available when the game starts. You can set the interaction (picking up objects, opening containers) and the action (open/close inventory) keys. Also you can set the maximum interaction distance and you can assign the sound effects played when a specific event occurs.

Updates

Some extra few steps are required to perform before updating the package (if any new update is available). Drag the "Core" folder (found in Resources) and drop it into the QIS directory. Then update the product and import the new changes. After that, put the "Core" folder once again into the "Resources" directory. This way you will make sure nothing from your project with the Inventory System integrated is compromised. Please be reminded that if you do not follow these simple steps, compatibility issues might occur. Do it at your own risk and consult informative videos for more information.

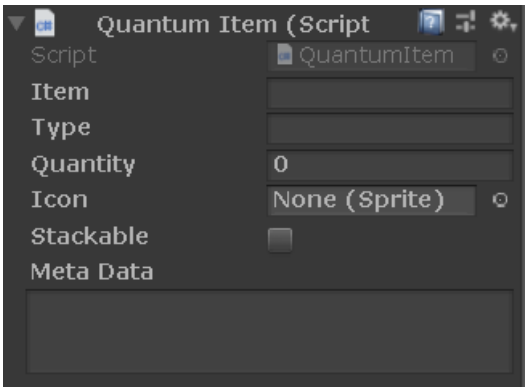
Items

These are game objects that contain information that is carried over to the inventory. Items can serve different purposes depending on their type. These objects are added into your inventory and their information is used to identify them.

Creating a Custom Item

In order to create a custom item, add the QuantumItemm component in any game object that has a collider and a rigidbody. Then, you should add the information required:

- **Item** (name): this is the name your item will have. Matching names, types and conditions will stack items. Also the name is used to find a specific slot/item within the inventory. Remember the item's name as you will use it in order to create integration scripts for your player.
- **Type**: it can be item, document, key, consumable, slot and custom. The type is just a way to determine how the inventory will behave when interacting with the object. Items, keys, additional slots (e.g. bags) and custom items will be moved from the inventory to the hotbar (or to a container if active) when clicked. Documents will display a text instead.
- **Quantity**: this is the amount of items are stacked in the gameobject.
- **Icon**: the image shown in the inventory slot.
- **Stackable**: is the item stackable or not? Should the item occupy a single inventory slot?
- **Metadata**: additional hidden information the object carries. This field is commonly used in items that accomplish a specific function when used/clicked. Documents read what is written in this field. Slots uses any integer value written in this area (adding and removing that specific number).



Documents

These objects show a specific text when pressed. These cannot be moved to the hotbar but they can be stored in containers.

Creating a Custom Document

In order to create a custom document, write "Document" in the "Type" field. Then write anything you want in the "Meta Data" field.

Other Types

These are not the only two items that exist. Not all of these types have a specific use, but they can be combined with other systems.

- **Key**: these are marked with a gold layout. Used to open containers. In order for a key to be usable, write something in the "Metadata" field. See the *Containers* category below for additional information.
- **Consumable**: these are marked with a red layout. This is meant to be combined with a status/survival system.
- **Slot**: these are marked with a cyan layout. When used, they add additional space to your inventory
- **Custom**: these are marked with a green layout. These are custom carriers. Metadata is important. Manipulate these items with a custom script.

Containers

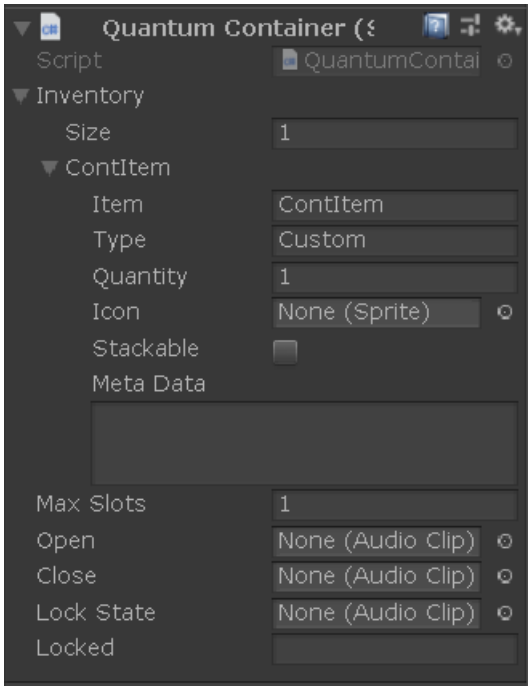
These objects can hold items of any type. Containers have a 20 item slots limit. You only need to manually input the data of the item.

- **Inventory – Size:** this is the amount of items that the container has.
- **Inventory – Slot**: this is the item that the inventory holds.
- **Max Slots**: set the maximum slots (20 is the limit).
- **Open/Close**: the sound effects when the container is opened/closed.
- **Lock State**: the sound effect when the container is locked.
- **Locked**: should the container be locked?

Creating a Locked State

If you want to lock your container, write something in the "Locked" field. Then create an object (Key type) and add what you've written in the "Locked" field to the "Metadata" field.

Done! When you try to open the container, the script will try to locate any item within the inventory/hotbar with matching metadata-locked values.



Functions

Many functions have been added to the Quantum Inventory System. In fact, the functions were split in two categories: **User Functions** and **DEV Functions**. This will indicate the function's difficulty.

You can use these functions in any script you create to freely interact with the Quantum Inventory System.

User Functions

Freeze(): freezes/unfreezes the player.

ActionInventory(): opens/closes the inventory.

DEV Functions

Gather(Item): using a Quantum Item component, adds a new item to the inventory.

Gather(Slot): using Slots (custom class: string, string, int, Sprite, bool, string), adds a new item to the inventory.

GatherHotbar(Slot): using Slots, adds a new item to the hotbar.

FindSlot(string): looks for a slot with the given name and returns the slot if found.

FindHotbarSlot(string): looks for a slot with the given name and returns the slot if found.

FindMetaData(string, string): searches an item with the given type and metadata value and returns true or false.

FindItem(string): searches for an item with the given name and returns true or false.

FindItem(string, int): searches for an item with the given name and quantity and returns true or false.

FindItemAndRemove(string): searches for an item with the given name and deletes it from the inventory.

FindItemAndRemove(string, int): searches for an item with the given name and quantity and removes the desired quantity. If slot's quantity reaches 0, it is deleted from the inventory.

GetHotbarID(int): returns the hotbar slot located in the given index.

I personally thank you for purchasing this asset and I truly hope it fits your needs. If you need anything, please make sure you contact me through my email: **quantumdev.zzn@gmail.com**

If you enjoyed the asset, please make sure to visit my Patreon site: https://www.patreon.com/tk742
