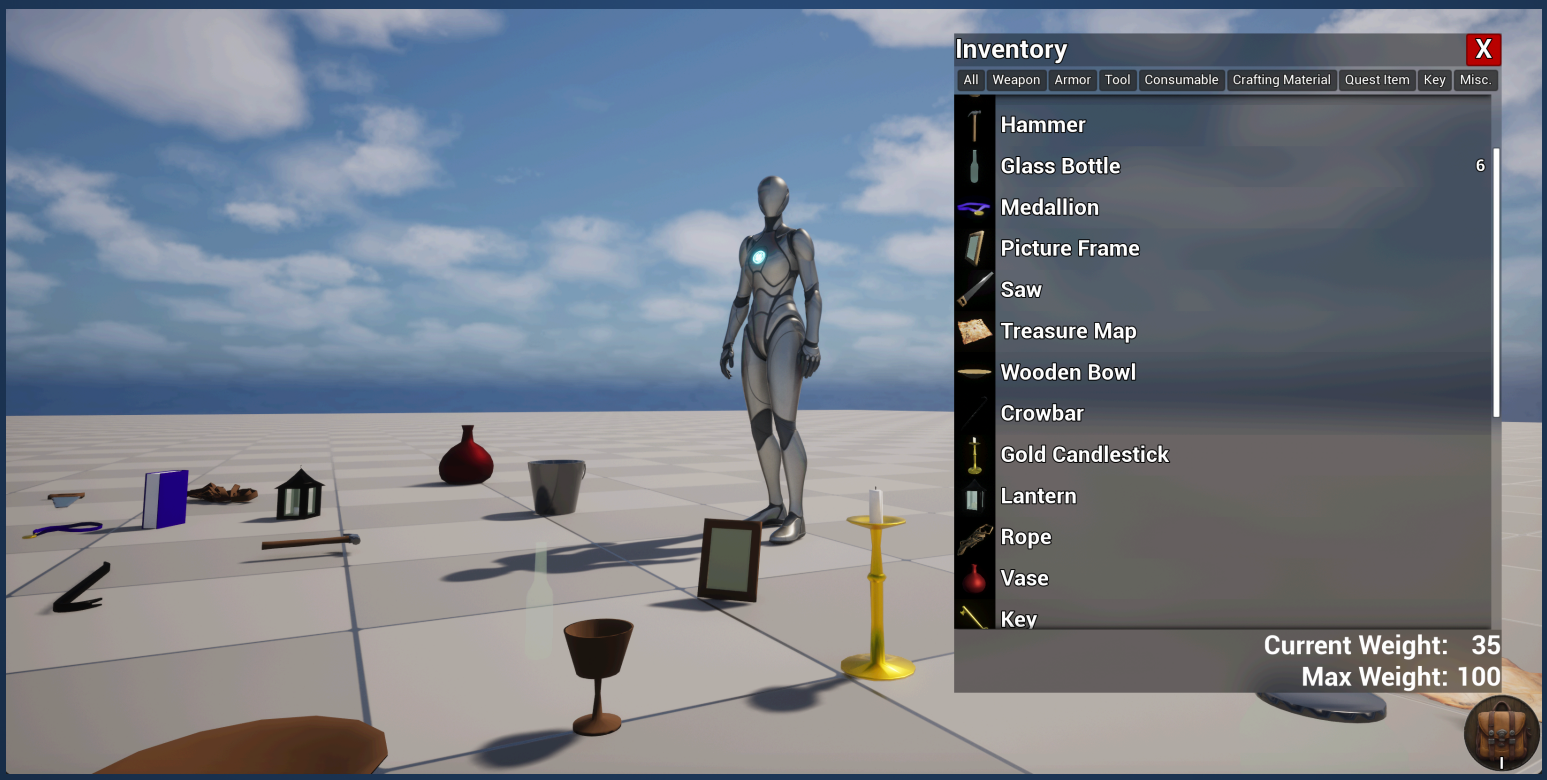
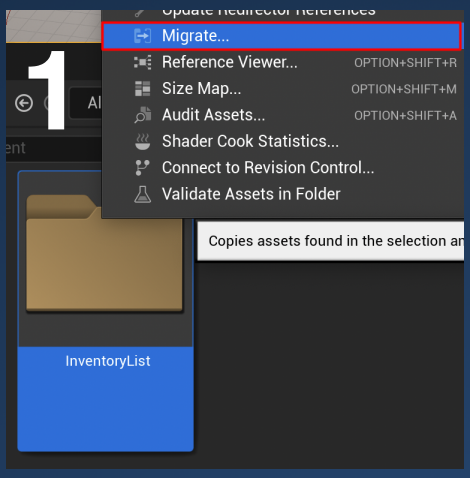
Table of Contents

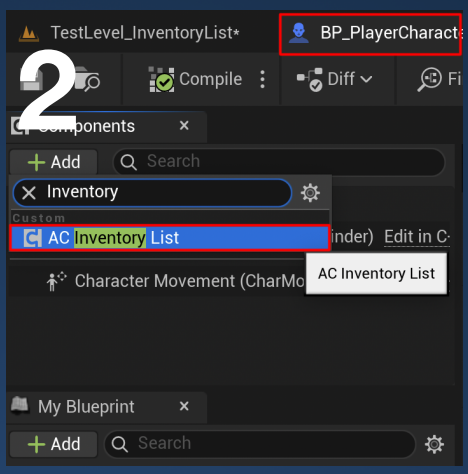
Introduction

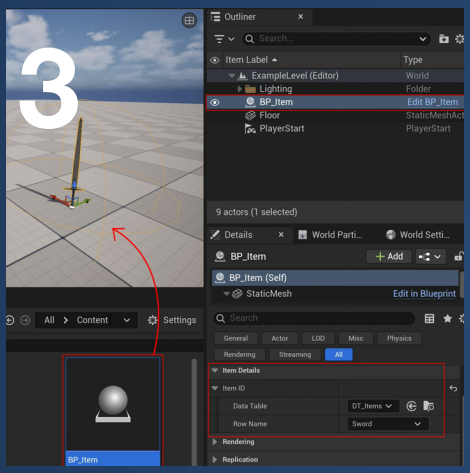
The Categorized Inventory List is a highly customizable, modular inventory system designed for Unreal Engine. It offers intuitive list-based organization with categorized tabs, weight-based inventory caps, and drag-and-drop functionality between inventory and chest systems. With extensive customization options, plug-and-play simplicity, and seamless integration with other systems and Unreal Toolkit asset packs. Whether used as a standalone system or alongside others, this inventory grid adapts to any project with ease.



Quick Start Guide







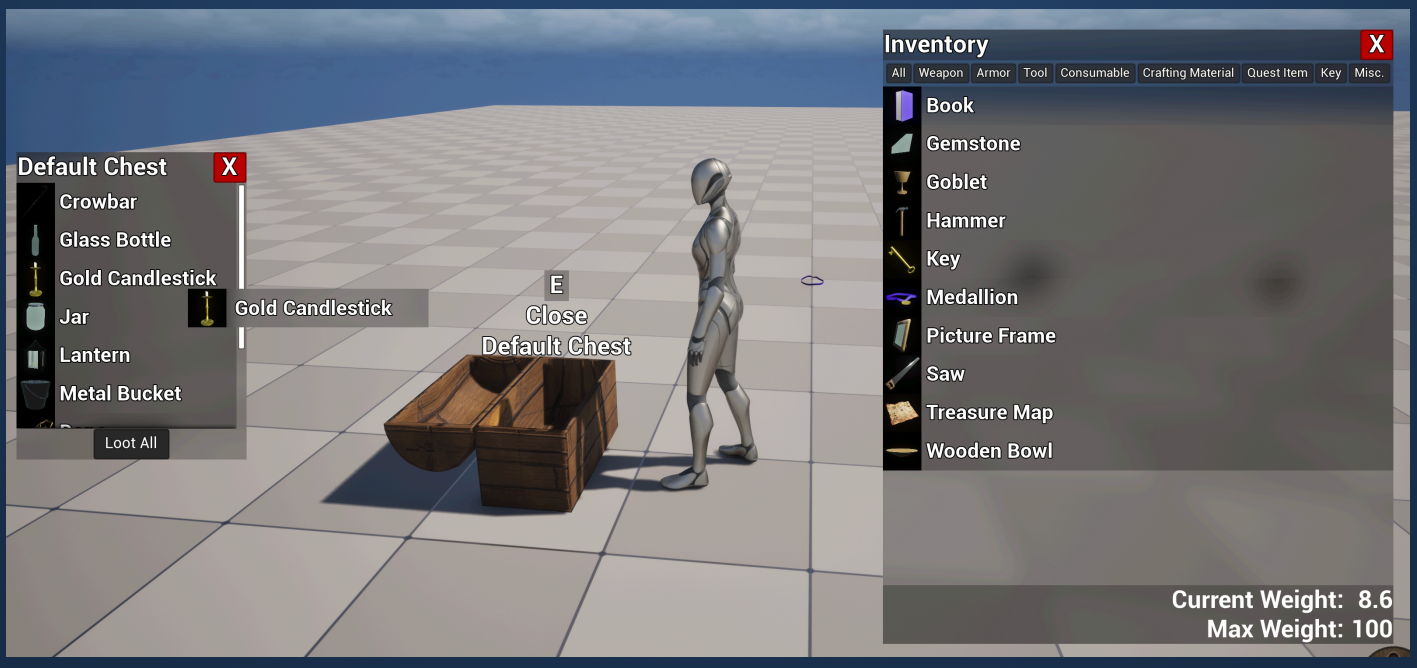
Follow these steps to get the Categorized Inventory List System running in your project:

1. Import the System:
   - Open the project file that came with this download.
   - Migrate the **InventoryList** folder to your project.
2. Add to Player Character:
   - Open your Player Character blueprint.
   - Add the AC_Inventory_List component to the player character blueprint.
3. Add Items to the World:
   - Open BP_Item
   - Ensure it has “AC_Item_List” as a component.
   - Drag BP_Item into the game world and assign an ItemID from DT_ItemData_List in the Details Panel.
4. Test Your Setup:
   - Press Play and interact with items using the default keybindings (e.g., **"I"** to open inventory, **"E"** to interact).

View full Installation Instructions.

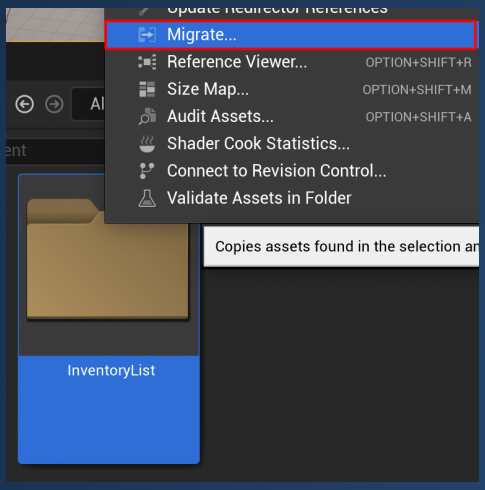
Core Features

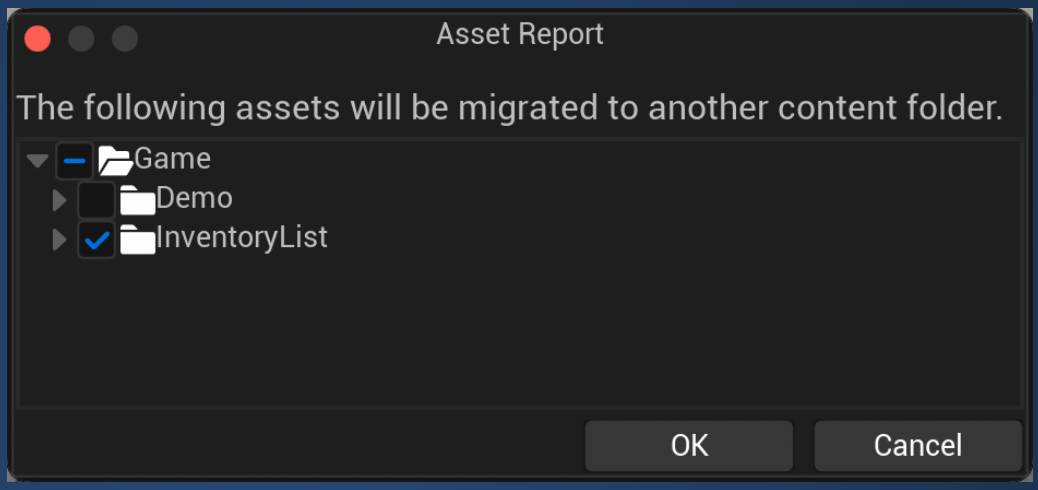
- **Categorized Inventory:**
  - Easily organize items with intuitive category tabs, allowing players to filter their inventory by item type. Perfect for games requiring structured item management.
- **Item Interaction:**
  - Pickup items, stack and split stacks, right-click menu, destroy items and drop items into the world, and more.
- **Chests:**
  - Fully integrated chest system for storing and transferring items. Includes drag-and-drop functionality, customizable chest inventories, and the ability to create custom chests via the data table.
- **Weight-Based Inventory:**
  - Optional weight cap functionality adds immersive inventory management. Dynamically displays current and maximum inventory weight, with easy customization.
- **Customization:**
  - Extensive options directly in the Details Panel, including adjustable UI elements, sound effects, animations, and notifications. All settings are accessible without needing to edit any code.
- **Seamless Integration:**
  - Easy to integrate with your own systems, and other Unreal Toolkit asset packs.
- **Example Items:**
  - Includes 20 example item models with a pre-filled data table for instant use, and easy customization.



Installation Instructions

1. Migrate Folders:
    - Open the project file that came with this download.
    - Locate the InventoryList folder in the content browser, right click it, and select **Migrate** (The Migrate option may be under **Asset Actions**.)
    - You'll get a popup with checkboxes next to all the assets in the project file.
        - **Uncheck the Demo folder.**
        - Ensure only the **InventoryList** folder is checked. (The "Game" folder will check by default.)
    - In the file browser, navigate to the Content folder of your target project and click open.





    - The InventoryList folder should now be in your target project, and the asset pack can be found inside.

View the Setup Guide to learn how to add the inventory to the player, and add items to the world.

Setup Guide

1. **Player Setup:**
   - Open your **Player Character blueprint**.
   - Add the **AC_Inventory_List** component to the blueprint.
     - Customizable settings can be found in the **details panel** when you have the component (AC_Inventory_List) selected in the components panel.

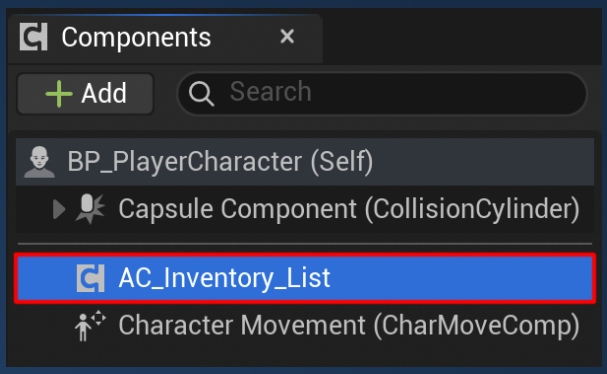
2. **Item Setup:**
   - Open **BP_Item** and ensure it has the **AC_Item_List** component (this allows it to pull data from DT_ItemData_List).
   - Drag BP_Item into the level and **select its ItemID from the Details Panel**. The item will dynamically construct based on the selected data.

3. **Chest Setup:**
   - Open **BP_Chest** and ensure it has the **AC_Chest_List** component (this allows it to pull data from DT_ChestData_List).
   - Drag BP_Chest into the level and select its ChestID from the Details Panel.

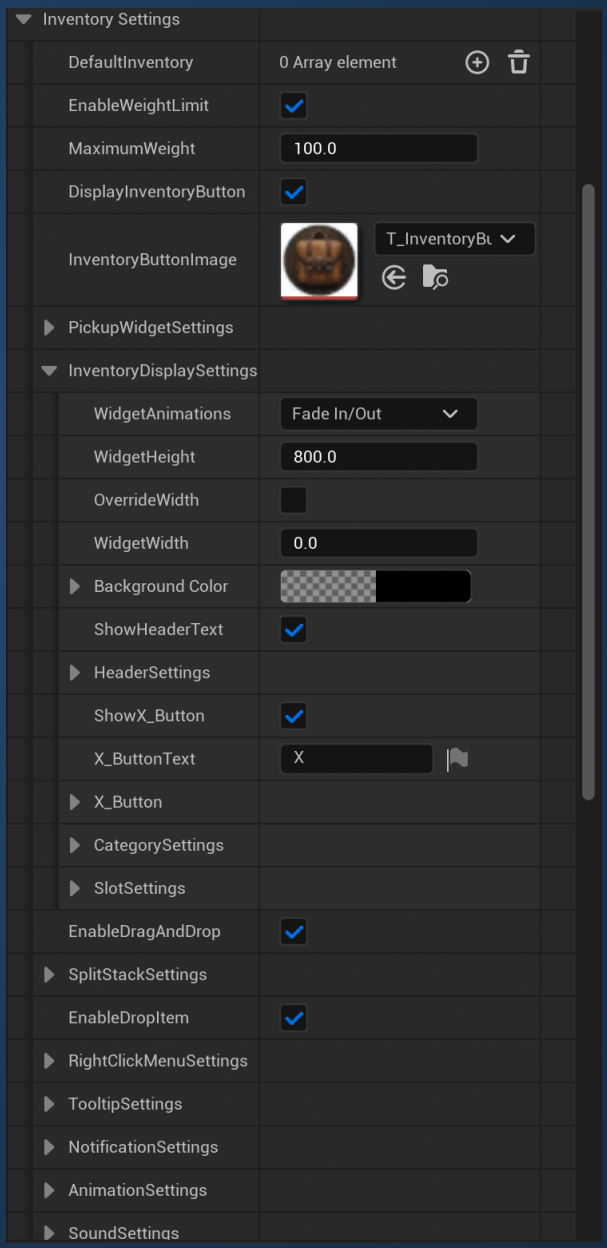
4. **Customizing the Data Tables:**
   - Expanding the system is easy—add your custom items or chests to the provided data tables
     - **DT_ItemData_List** - Data table for adding items, expand this with custom items
     - **DT_ChestData_List** - Data table for adding chests, expand to make chests with custom inventories
   - Open the data table, and select +Add to add a new row.
     - Fill out the data table row with a **unique row name**, and then fill out the item details.
     - **For ItemID and ChestID you must select the exact data table row that you're currently adding in order for the item or chest to properly be identified by the system.**

For full customization, view Customization Options.

Customization Options



The Categorized Inventory List provides extensive customization options, allowing developers to fine-tune functionality, UI design, animations, sounds, and more—all directly accessible from the Details Panel in the Player Character Blueprint. To customize, open your Player Character Blueprint, select the AC_Inventory_List component in the components panel, and locate the “Inventory Settings” section in the details panel.



Inventory List Settings

- **Category Tabs**: Toggle visibility of category tabs (e.g., Weapons, Consumables) and customize their text and style to fit your game.
- **Item Weight Display**: Enable or disable weight-based inventory caps.

UI Design

- Inventory Display Settings:
  - Toggle the visibility of headers, close buttons, and background elements.
  - Customize the appearance of buttons and fonts for a cohesive design.
- Tooltips: Enable or disable tooltips and configure the displayed item details, such as name, category, and description.

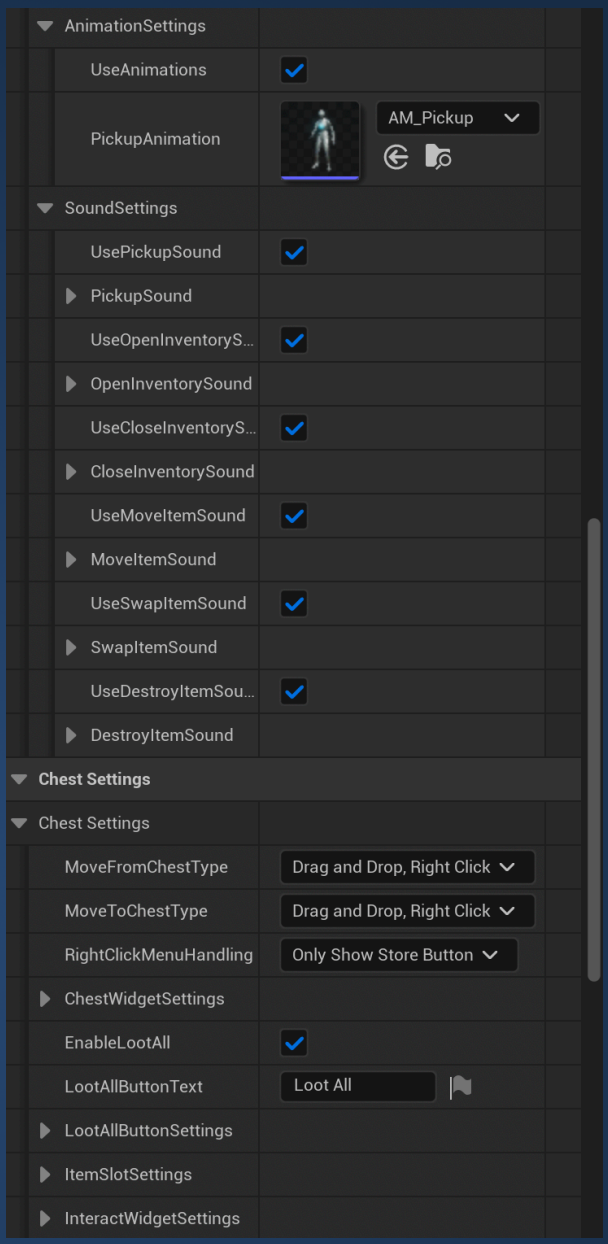
Item Interaction

- Right-Click Menu:
  - Fully customizable menu options for right-click interactions.
  - Show or hide buttons like "Cancel" or "Close" and adjust font styles and colors.
- Split Stack Options:
  - Configure the split stack menu to display item names, quantity dividers, and custom text for buttons.

Notifications

- Enable or disable inventory notifications for actions such as adding, removing, or swapping items.
- Customize notification text (e.g., "Item added to inventory" or "Inventory full") and widget styles.

Customization Options



Audio & Animations

- Sounds: Toggle individual sound effects for actions like item pickup, swapping, or destroying.
- Animations: Include custom animations for actions such as picking up an item.

Chest Settings

- Adjust chest interaction options, including drag-and-drop functionality, right-click menu behavior, and loot-all features.
- Configure the chest UI, including widget size, button text, and background colors.

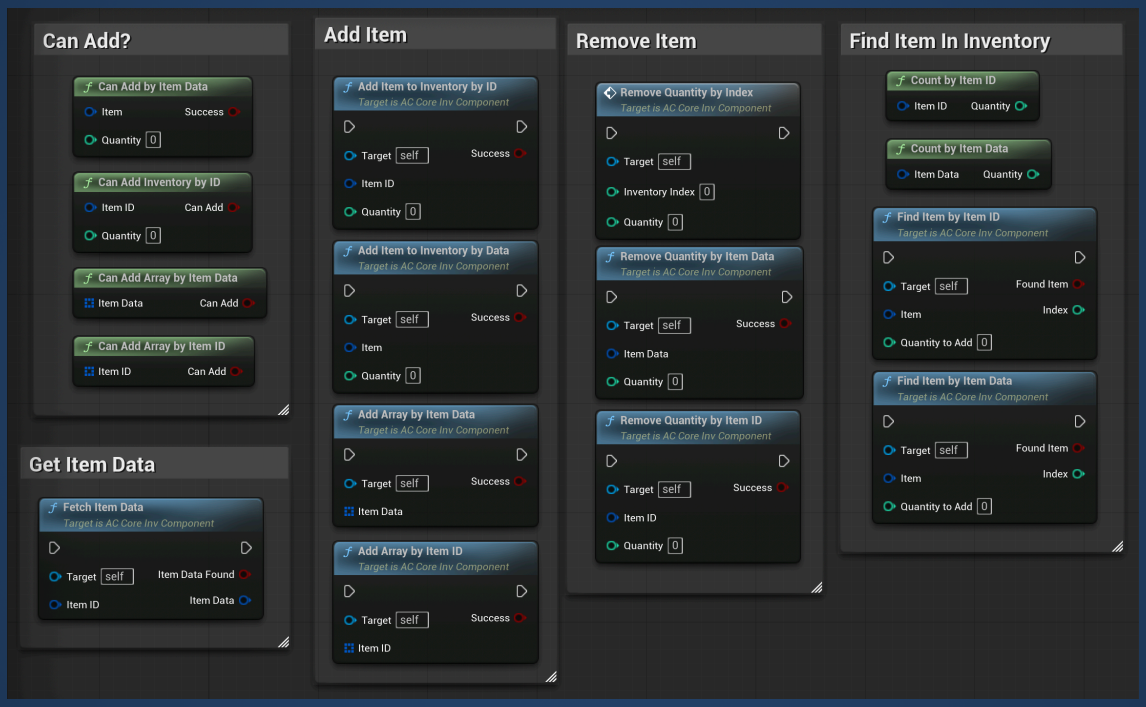
These settings offer developers the flexibility to adapt the Inventory List System to any game style or visual design. The intuitive layout of the Details Panel ensures that customization is quick and easy, requiring no additional coding. For more advanced modifications, developers can explore the blueprints and data tables for deeper integration.

Advanced Usage

Using Helper Functions:

From your other systems, use Get Component by Class to AC_Inventory_List. With the reference, you can:

- Call helper functions like:
  - AddItemToInventory
  - RemoveItemFromInventory
  - CheckIfItemsMatch
  - And more.



Item Data Handling:

- Inventory items are stored as an array of the **F_ItemBase** structure.
- Each item's **ItemID** refers to its corresponding **data table row**.
- To integrate with external systems:
  - Fetch the ItemID from the desired data table row.
  - Use the helper functions with ItemID inputs for seamless integration.
  - F_ItemBase includes a reference to the ItemID to easily identify items.
  - To get a reference to the players inventory array, use the reference to the inventory component to get the "Inventory" variable

- **ItemID:** Refers to the row in the data table.
- **ItemData:** Refers to the F_ItemBase struct containing all item variables.

For ease of use, it's recommended to work with ItemID when integrating your own systems.

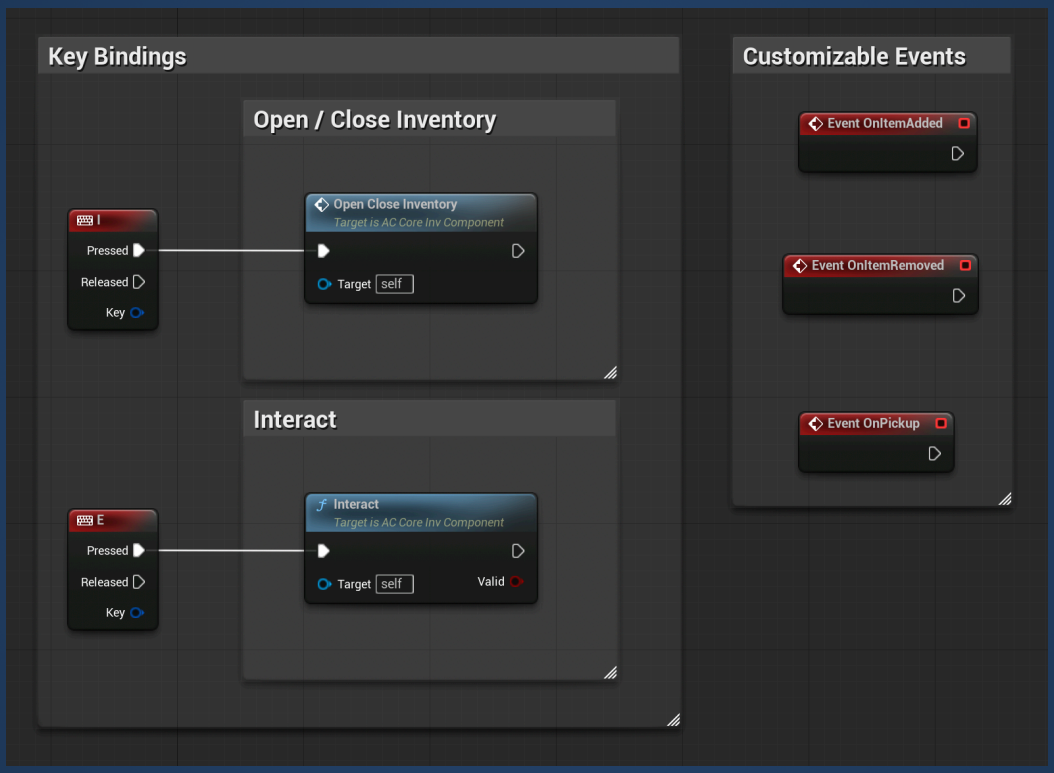
Advanced Usage

Custom Events and Extendability

The Inventory List System includes custom events for easy logic expansion:

- OnItemAdded: Triggered after an item is added. Add custom logic here.
- OnItemRemoved: Triggered after an item is removed. Add custom logic here.
- OnPickup: Triggered when the player picks up an item.

These events allow you to seamlessly extend functionality without needing to modifying core functions.



Quick Reference Guide

Main Blueprints

Asset	Purpose
BP_Item	Blueprint representing items in the world.
BP_Chest	Blueprint representing chests in the world.

Main Components

Asset	Purpose
AC_Inventory_List	Main inventory logic.
AC_Chest_List	Component that allows BP_Chest to get chest data.
AC_Item_List	Component that allows BP_Item to get item data.

Main Blueprint Interfaces

Asset	Purpose
BPI_Interact	Interface for intractable actors.
BPI_AddTooltipElements	Adds tooltip details to the tooltip UI.
BPI_DragAndDrop	Logic for drag-and-drop functionality.

Main Data

Asset	Purpose
E_ItemCategories	Enum defining item types/categories.
F_ItemBase	Struct determining Item Data.
DT_ItemData_List	Data base for adding items.

Main Widgets

Asset	Purpose
WBP_InventoryListWidget	Main UI for the Categorized List Inventory system.
WBP_ChestInventory	UI for interacting with chests.
WBP_HUD	UI that determines position of other UI elements.
WBP_RightClickMenu	Right-Click menu for item interactions.

Support & Contact

For any questions, issues, or feedback, please contact us at:

support@unreal-toolkit.com

Explore all of our assets on: Unreal-Toolkit.com