InventorySystem

An advanced Drag ‘n Drop inventory system

Documentation

We’re working on making InventorySystem even easier!

A HowToCompute Publication

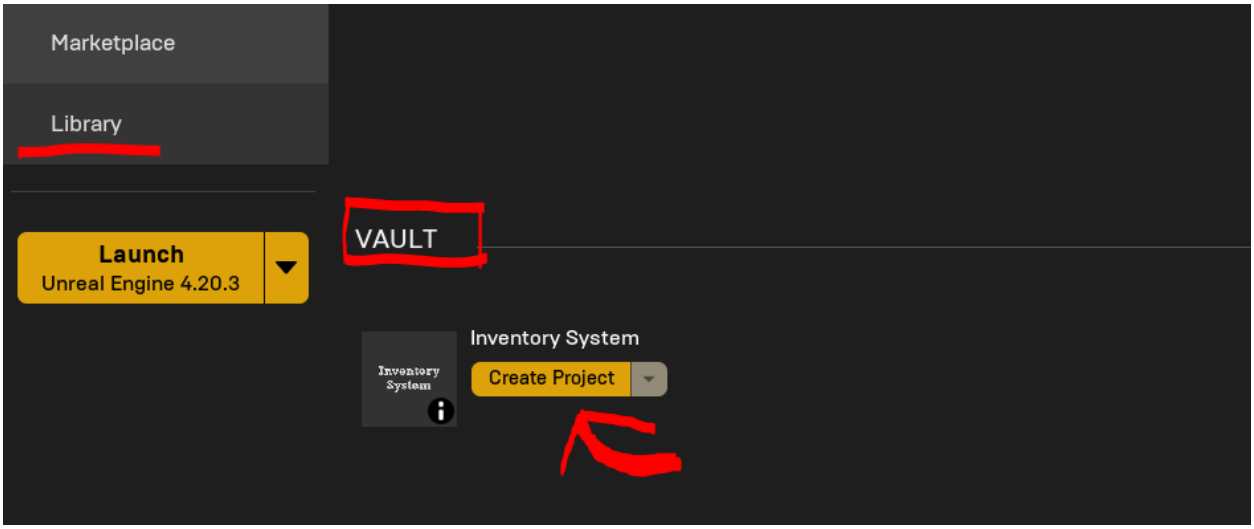
Table of Contents

Getting Up And Running

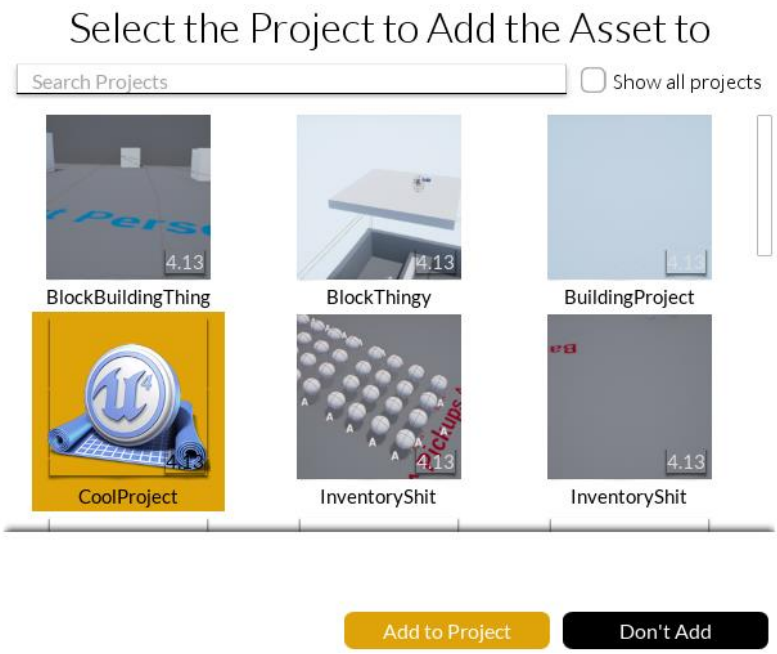
A quick note before we get started - we're currently working on improving the documentation and making it even easier for you to get started with InventorySystem. Our apologies for the inconvenience, and we hope you will enjoy using InventorySystem.

Please also note that this (pdf version) of the documentation may be outdated. We tend to package new versions of the documentation with updates pushed to Inventorysystem, however, we highly recommend taking a look at the google docs version, that can be found here: https://docs.google.com/document/d/1y8cLoL3ss643BEw2uLcuNoHbWmXbkMinwVGaAWJ2lT4/edit?usp=sharing

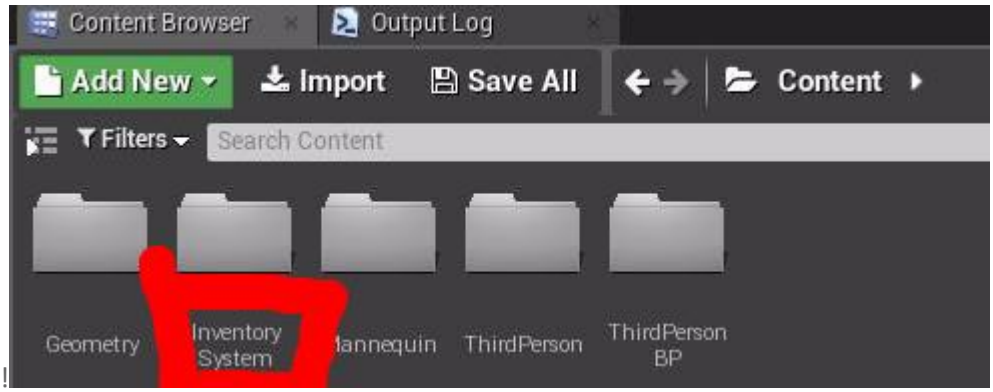
Start by adding InventorySystem to your project. After you have purchased InventorySystem from the unreal engine marketplace, it will end up in your vault. So open up the epic games launcher, go to the unreal engine tab, Library and if you scroll to the bottom, there is a tab named vault. Find Inventory System and click on Add To Project.



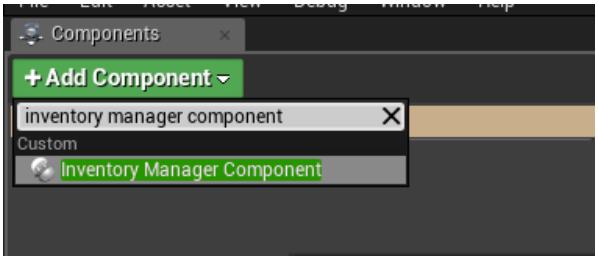
Now select the project you want to add it to, in my case this will be CoolProject. For reference, this is just the 3rd person template with a custom playerstate.



Now in your content browser, you should have the InventorySystem file, if not be sure you are not in a subfolder!



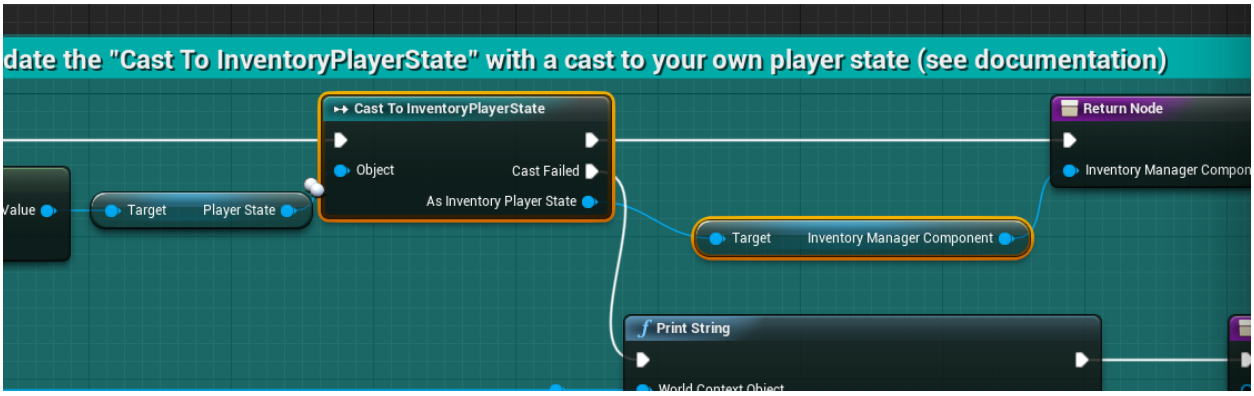
So the first thing that you will want to do is add an InventoryManager component to your PlayerState. For this open up your playerstate and add an “Inventory Manager Component”. (AddComponent, and go through the list)



Once you have added that hit Compile and Save.

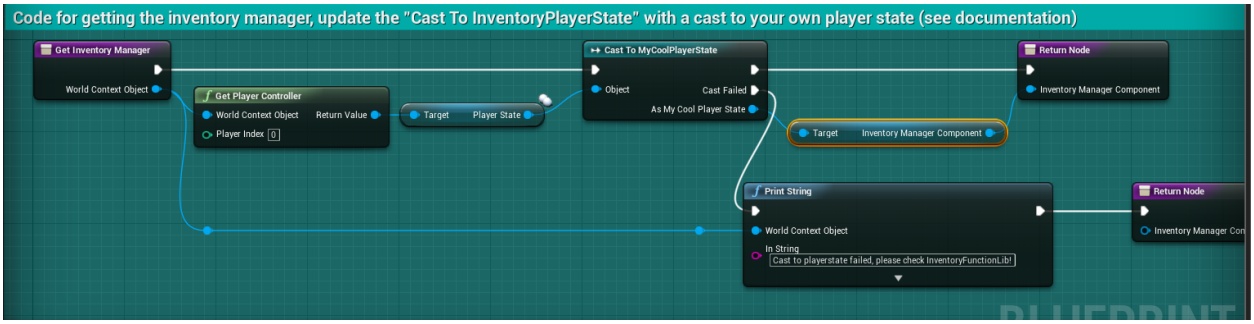
Now head over to InventorySystem > Blueprints > InventoryFunctionLib (blueprint file). And find the cast

node.



Delete the highlighted, and drag out of the playerstate. Type in cast to <your playerstate>. Then drag out of that and type in “get Inventory Manager Component” and press the one under variables > default. Now connect the output of that to the return node, and take the execution pin from “Get Inventory Manager” to the cast node. Now connect the execution pin without text to the return node, and the one with text to the Print String.

In the end you should end up with something like the following (MyCoolPlayerstate being the name of your playerstate):

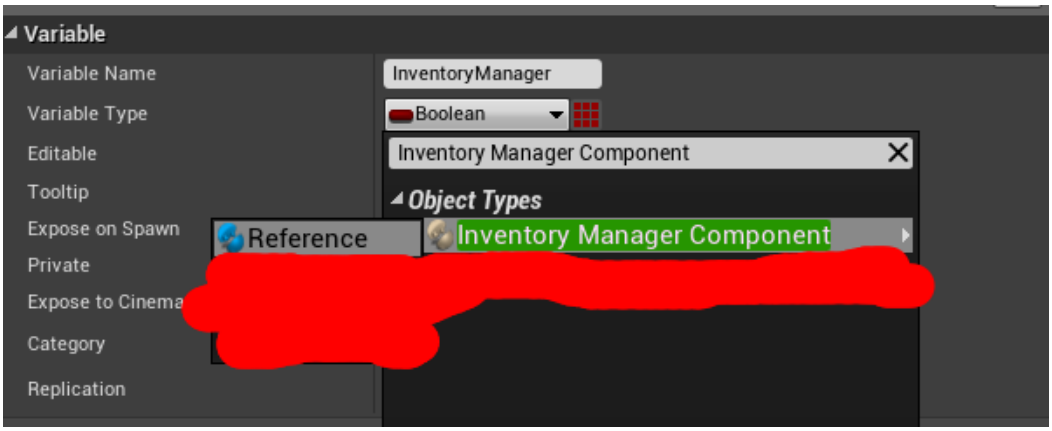


Now head over to InventorySystem > Example Project > InventorySystemCharacter. Head over to the “Inventory System code” part. Now head back to your content browser (don’t close this tab! Just minimize it) and open up your character (in my case the 3rd person character). If your character doesn’t yet have a begin play event, add an “Event BeginPlay” node. Now find the Variables tab on your left and press the +.

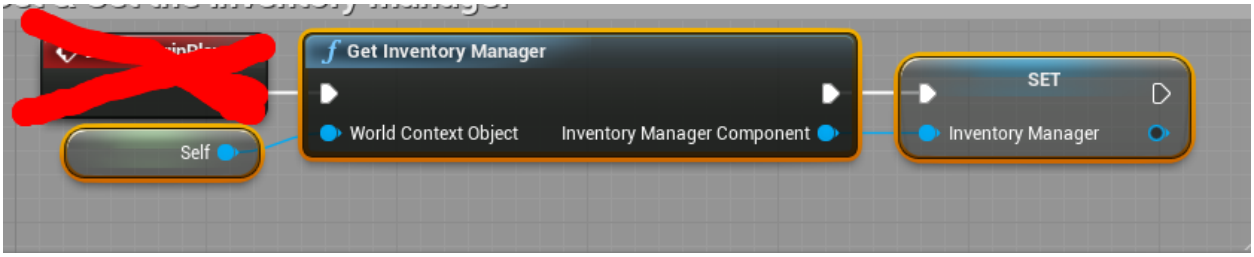
Name it “InventoryManager”. Be sure to spell this name correctly, or stuff will go wrong!

Give it the type “Inventory Manager Component” by Pressing the existing Variable Type (probably

Boolean), typing in “Inventory Manager Component”, clicking it and pressing reference. Be sure to quickly compile and save.

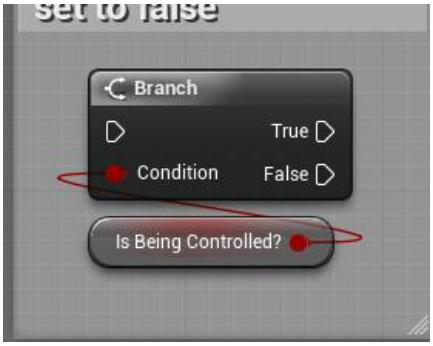


Head back to the InventorySystemCharacter and copy the below selected code.

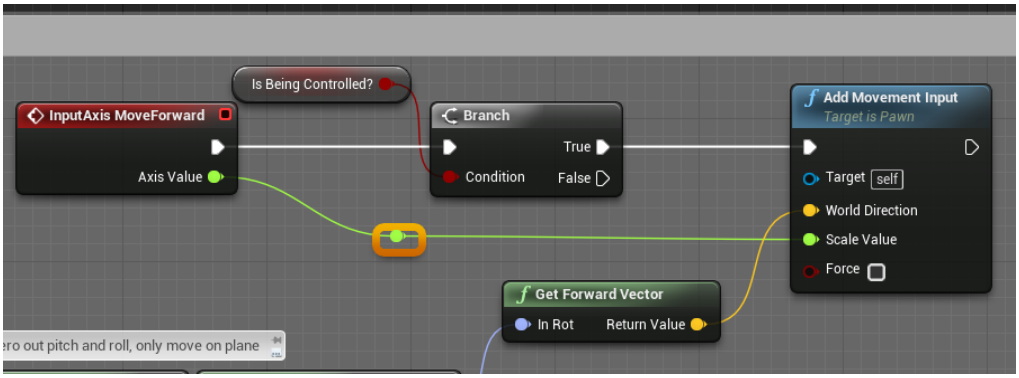


Now go back into your character, and paste this right after begin play (to avoid the case in which it would not be called!). Be sure to connect the input from begin play to the Get Inventory Manager node. Don’t forget to connect the output from the SET node to the rest of your own BeginPlay code.

Now create a new variable of type Boolean and the name “IsBeingControlled?”. Copy the following piece of code over to all of your input functions (unless you already have a system for losing control in place, in which case you will need to modify later code). In the case that the checkbox is not check, please check it or you won’t have any control over your character!

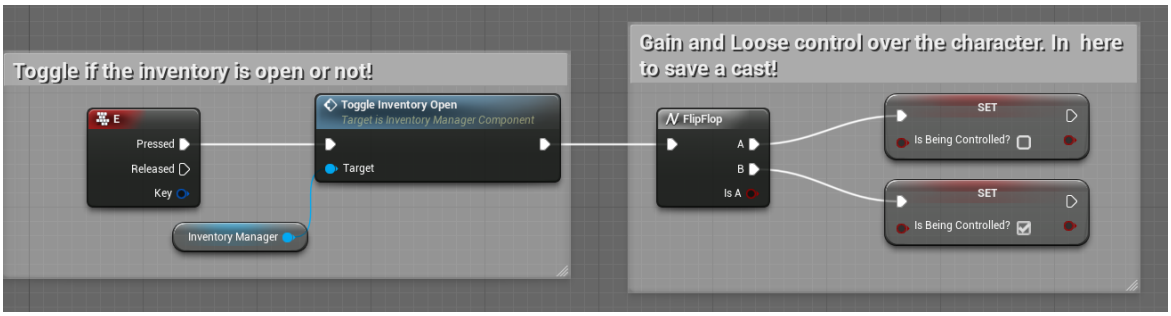


You should end up with code like this:



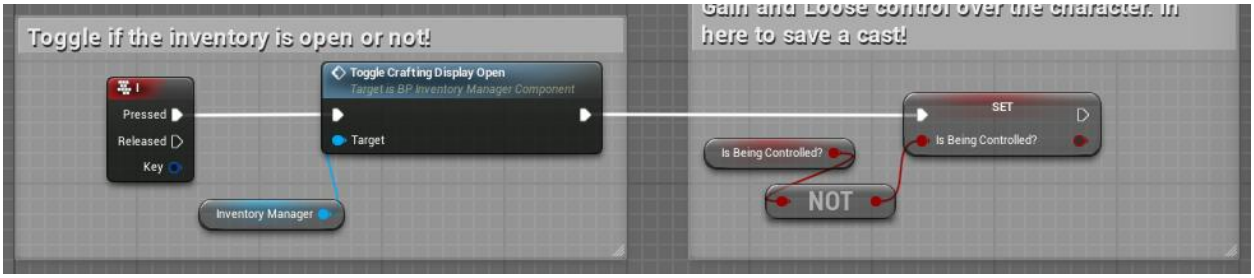
[explanation] Before your character can move or do anything it will check if it is being controlled, this allows the system to not let the user look around etc when they have their inventory open.

And now for the easiest part! Find the piece of code that toggles the inventory in the InventorySystemCharacter.

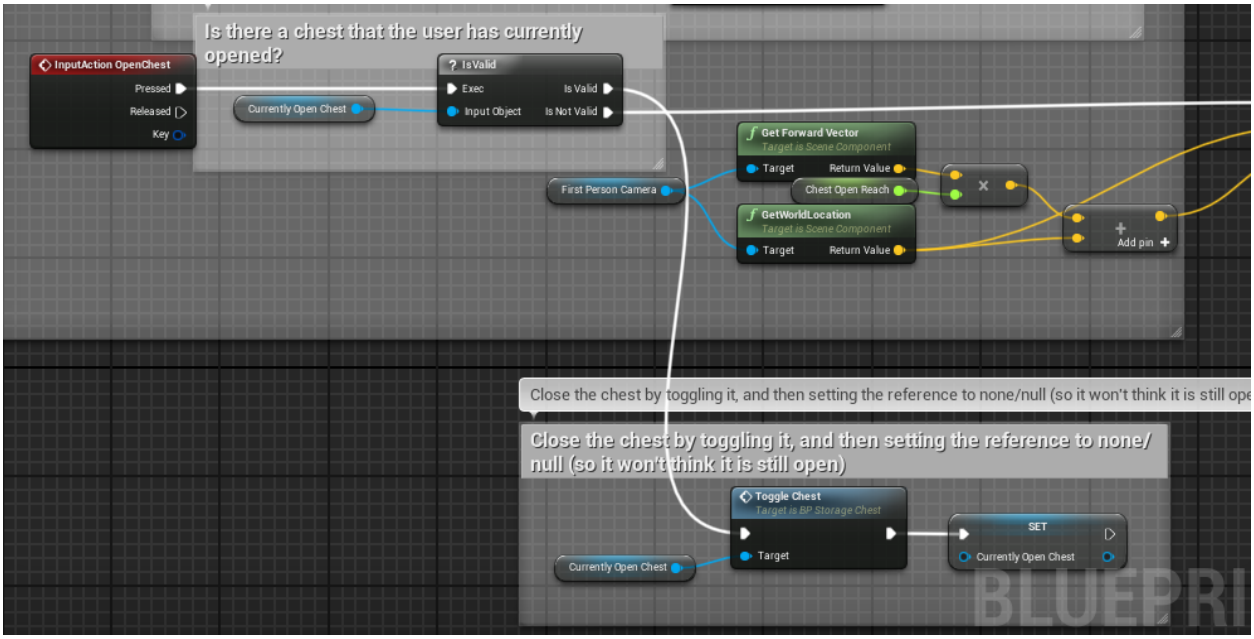


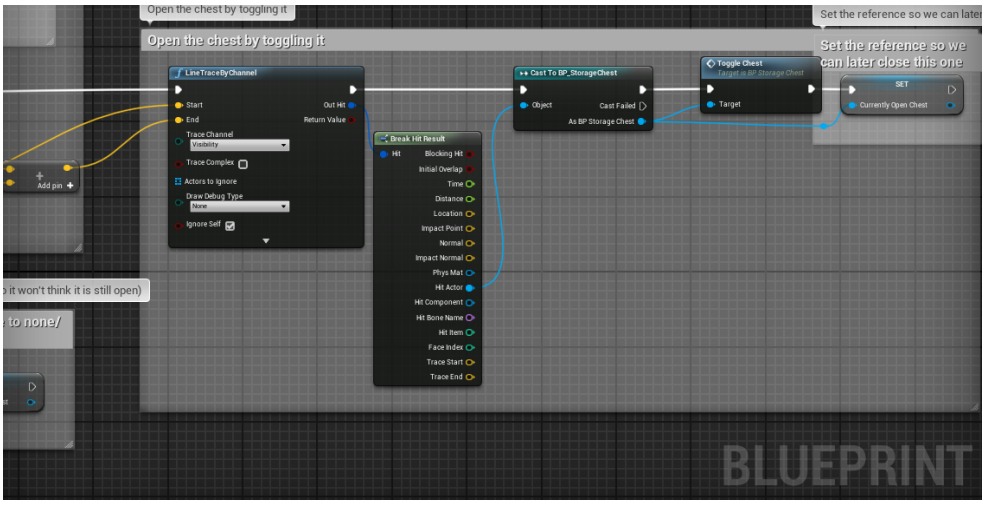
Copy that and paste it in a free spot of your own character.

If you'd like to enable crafting, also make sure you copy the following snippet of code that toggles the crafting display.



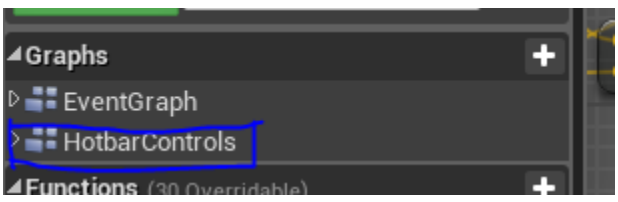
For chests, there's some ray-tracing involved. This means the code will be a little more complicated, thus we have included two screenshots to help you out. Please ensure you copy the "InputAction OpenChest" event, as can be seen below

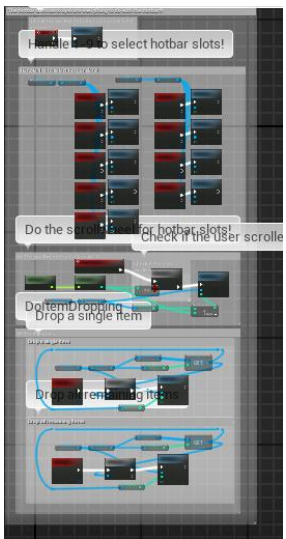




You may need to replace the “InputAction OpenChest” node with A. a custom input action, or B. a hard-coded key, like “C”.

For the HotBar to function, we’ll need head over to the “HotbarControls” graph. Depending on your editor, the “Graphs” selection will likely be on the left, right under the “Add New” button. Double-click the “HotbarControls” item here.





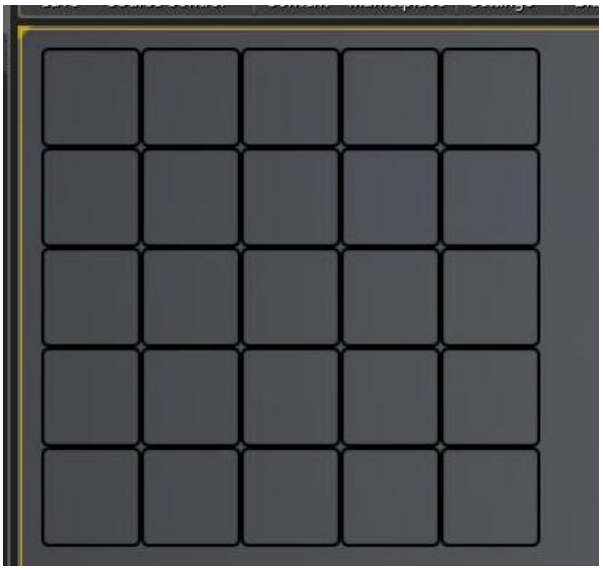
Now make sure you new character has a secondary event graph for the hotbar controls too (press the "+" sign on your graph), and then copy all of the nodes in the example character's graph over. This will enable your character to switch between the slots in your hotbar using they scrollwheel and/or number keys.

Compile and Save your character and we are ready to test!

Testing!

Now fire up one of your maps with your just edited character in it. When you press play you should see something like the following:

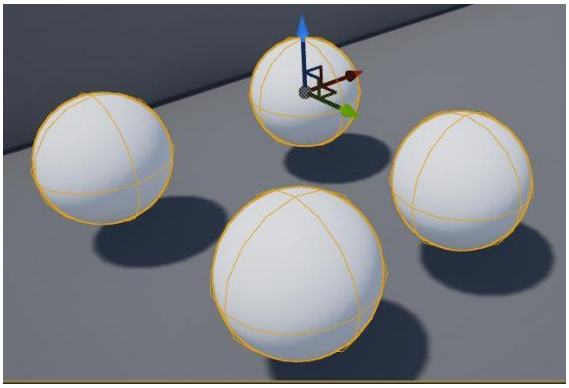
Now press E and you should get something like:



This means the basics work! Now let's try and add/remove some items!

Let's place a few pickups by going to InventorySystem > Mysc and dragging a few "Pickup"s into the

scene.



Now you can adjust the pickup's settings, change the item and the amount. I'd personally recommend making 2 of them apples, and 2 of them burgers. I'd also make at least one less than 0 to test if removing items works. Now pick up items and drag them around in your inventory and move them to your hotbar. Try dropping a few and picking them up. (see the below article on controls).

If everything worked as expected, congratulations, you have successfully ported InventorySystem to your project!

Controls

Use E to open/close your inventory.

Use I to open/close the crafting display.

Use O to open/close a chest.

Use keys 1-9 or your scrollwheel to select hotbar slots.

Use Q to drop an item, use CTRL+Q to drop all the items that are remaining in the selected slot.

Common Errors:

I get the error: "Cast to playerstate failed, please check InventoryFunctionLib"!

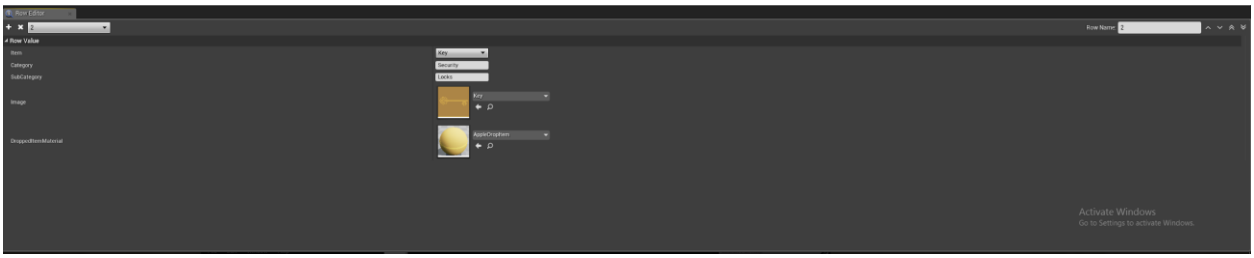
This means that it could not cast the current PlayerState to whichever cast node you specified. This is most commonly caused by setting the incorrect playerstate. Head over to your gamemode, and set the correct one. Did that not work? Check if you are overriding the playerstate in your world settings!

Adding your own items!

Adding items is easy!

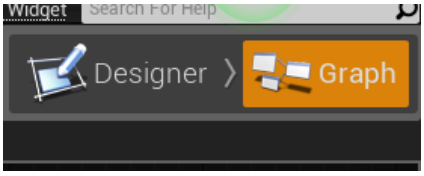
First open up the EItems enum (in InventorySystem > Enums). Now Press the new button on the top right. Add as many as you want, giving them sensible names.

Now open up the Items datatable (in InventorySystem > Datatables). For each item press the + in the row editor. The current version of the system counts on the fact that you have ordered your enums and your datatable the same. So first of all give your row the name: previous row+1 (so if the last item had the rowname 2, this one will be 3). Now populate it with the wanted data. Category, SubCategory and DroppedItemMaterial are optional, tho setting a material is strongly recommended! Y**ou MUST also set the "Item" to your newly created Enum for this to work.** Your row should look something like the following:



Now place a pickup in the your scene of choice, and set it's item to your newly created object. Now pick it up and see if it works as intended! Have you already downloaded the bonus assets? Click here.

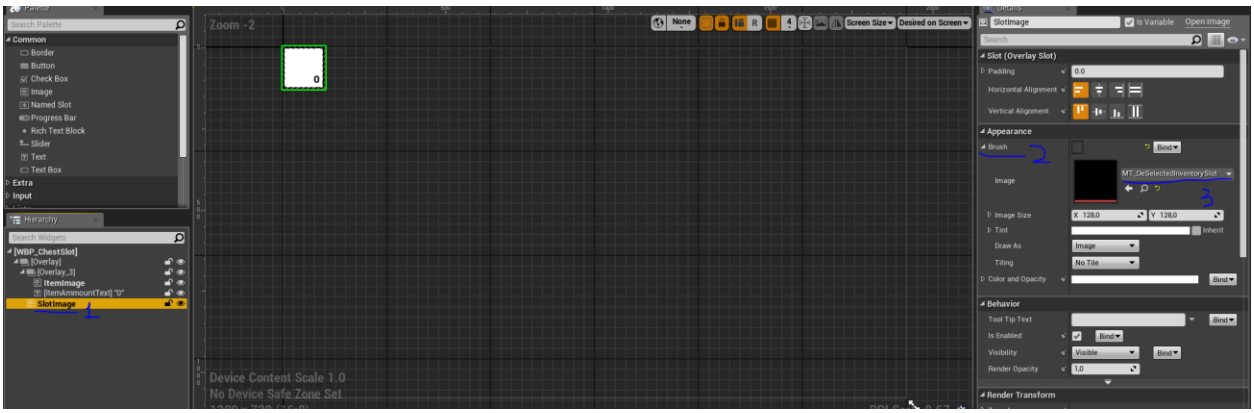
Customizing the UI



If you head over to the UMG folder (Inventory System > UMG). For most of the widgets (atleast the ones that contain amounts) if you head to the graph, you can set the amount's color in the ItemAmmountText_Color variable!

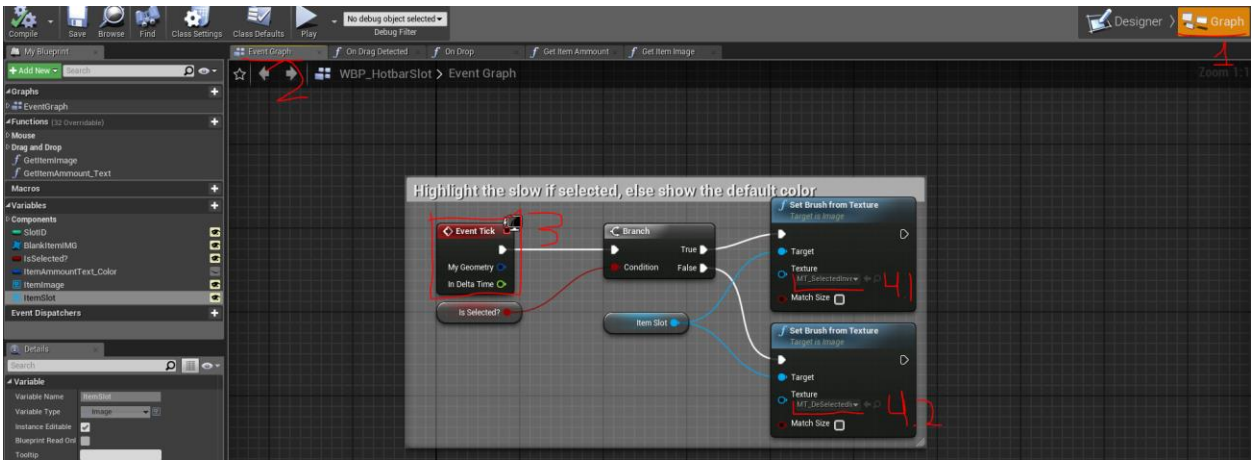
Another easy way to customize the look of InventorySystem, is changing the way your slots look. You can do this by heading into InventorySystem > UMG, and opening up any of the WBP_XXXSlot blueprints. Once in here, make sure you're on the designer tab, and update the "SlotImage" brush, as

can be seen below:



There's one exception to the above method though - and that's the WBP_HotbarSlot widget! This widget can have two states, a selected state, and a deselected state. We recommend that you use two different images for these slots, since that will make it a lot easier for the user to distinguish between a selected slot and the other slots.

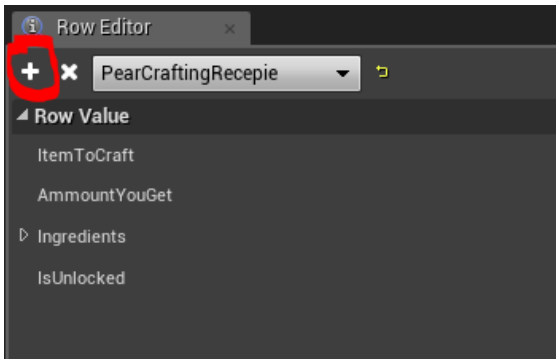
To update these images, open the widget, then press the Graph tab, and ensure you are on the "Event Graph" tab. Now find the "Event Tick" node, and update the textures for the True and False pins. The "True" pin will contain the selected slot texture (4.1), and the "False" pin the deselected slot (4.2). See the image below:



Even more UI customization is possible, but not officially supported. Please stay tuned for future updates to feature more exposed variables to change the color of the slots etc!

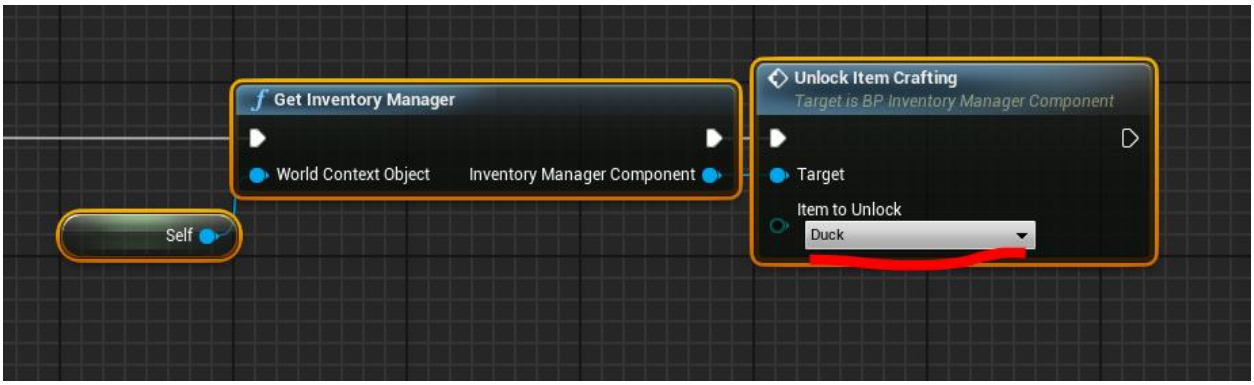
Using the crafting functionality

InventorySystem has a crafting system built into it! Crafting is enabled by default, and can be disabled by unchecking the checkbox “Is Crafting Enabled?” checkbox in the InventoryManager component.

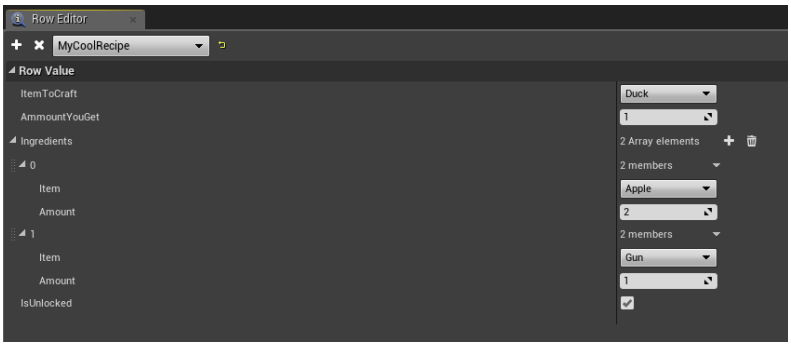


The first step to using the crafting functionality is adding your recipes. This is done by opening up the DT_Crafting datatable (located in InventorySystem/Datatables/DT_Crafting). To add a recipe, press the + button in the row editor, and give your row a name (don’t worry about what you call it, however, I would recommend using a descriptive name). Next, use the dropdown to select the ItemToCraft (what the user should get if they use the recipe). Now also change the AmmountYouGet to the number of items the user should receive when crafting. Next press the plus for Ingredients, and expand the newly created item by pressing the >. Change the item that is required, and the amount of it that is required. NOTE: You should not have the same item in your recipe multiple times. Repeat this procedure for any further ingredients you may want to add.

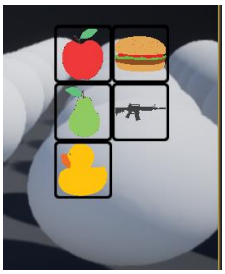
Once you’ve added the ingredients, you need to decide if the user can craft this right away, or you want to wait with unlocking the recipe until the user has progressed in your game. If you want the user to be able to craft it right away, then check the “Is Unlocked” checkbox, otherwise, you can use the code below (updating the ItemToCraft parameter) to unlock the recipe during runtime:



A final crafting recipe that is unlocked by default will look a lot like this:



In order to start using this recipe, please remove your save file (see the troubleshooting section for more information about this), and reopen the editor. You should now be able to see your new recipe when pressing i in-game. You can see the duck added to my crafting display below to the side.



Using the Chest functionality

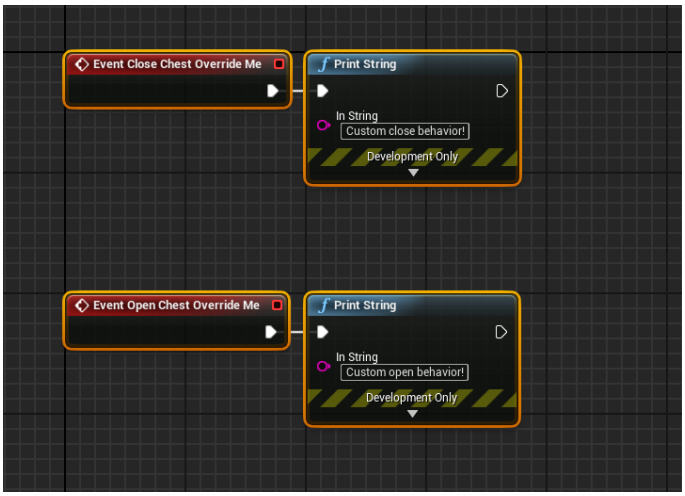
InventorySystem's latest update contains chest functionality! Please note that you will be required to use Unreal Engine 4.19 or later, and the latest version of InventorySystem for this to work (at the time of writing).

To get started with adding chests to your game, create a new blueprint class.

Now Use the All Classes part to select BP_StorageChest as your parent class.

Now that you've created your chest blueprint, open it up. You easily change parameters like the default number or rows/columns, and the mesh now. To edit the number of rows/columns, edit the Rows and Row Size parameters under default (as shown next to here).

To change the mesh, open the Default Root > StaticMesh component, and update it's static mesh value. You can now drag it into your level and start using it. To open a chest, move close to where you placed it, and press O. You need to be looking at the chest for this to work, since it raycasts from the player's character. You can now drag anything from your hotbar, or if you open your inventory (using E), your inventory into the chest, and it will handle saving/etc. automatically. If you'd like to add custom behavior (like an opening/closing animation) to your chest, then please override/use the "Event Close Chest Override Me" / "Event Open Chest Override Me" events in your event graph. You can see an example that prints a message out when the user opens/closes the chest below:



Troubleshooting Tips

I'm sorry to hear you're experiencing an issue using InventorySystem. I've compiled a list of common issues, fixes, and things to try when experiencing an issue, with which I hope to enable you to solve the issue as quickly as possible. Please note that the list of common issues/fixes will grow over time.

I added an item/recipe, but it isn't showing up!

This is likely caused by there still being an old(er) version of your unlocked recipes or items in your save file. In order to fix this issue, please remove your save file (See the below section).

How do I remove my save file / reload InventorySystem?

When experiencing issues with InventorySystem, the most likely cause is your save file still containing old data after you've updated your datatables/settings. In order to fix this issue, open up your operating system's file browser, and navigate to your Unreal project. Now head into Saved/SaveGames, and remove the "InventorySave.sav" file. You may want to close your game/editor before doing this, and reopen it after.

Still having issues?

Now that InventorySystem has gone free, please use the below contact information:

For Further Support: Please report any issues to marketplace_support@unrealengine.com

Multiplayer Footnote

If you're having issues using Multiplayer, then please note that it is currently not supported, and your mileage may vary. InventorySystem was designed with Singleplayer in mind, and we did our best to make it as easy-to-use as possible for that scenario. Our apologies for any inconvenience this may cause.

Conclusion

This will conclude the documentation of the InventorySystem, we hope you were able to get started using InventorySystem, and can't wait to see what you'll create next!