Nestor Sabater

nsabater.com
contact@nsabater.com

Building System V3

Unreal Engine 4.24+

The project

Unreal Engine Building System is a framework that will empower you with an advanced building system like the one seen in "**The Forest**" game, allowing you to place structures around the world and carrying the resources needed to build them.

Key Features

1. **Estructures with special conditions and functionalities**
2. **Unlimited types of buildings and resources**
3. **Carriable objects**
4. **Interaction system**

Examples included

Resources

1. Wood Log
2. Wood Stick
3. Stone

Buildings

1. **Wooden Wall**: Made of *Wood Logs*, each time you add a log you can see how it appears in the structure, with collisions enabled. Each wood log adapts it´s eight to the ground height.
2. **House**: Small house made of *Wood Logs, Wood Sticks* and *Stones*. You can see how it gets constructed each time you add a material (sames as the wood wall)
3. **Campfire**: Small structure made of three *Stones* and two *Wood Sticks*. You can't see any difference when you add materials but upon completion it gets finished, with collision and with an extra interaction "Use" that lets you put it on fire. This structure also has ground alignment so it gets turned when build on an slope.
4. **Bridge**: This structure has parts that gets visible when you add materials and parts that only becomes solid when you finish adding all materials needed. Made of *Wood Logs*. It can only be built in ravines, half of it must be place in ground and the other half must be "in the air" with no ground under it.
5. **Destructible Wooden Wall:** This building is made of wooden logs and some stones. When It gets damage It's integrity decreases. Upon destroyal, It spawns the stones and logs that were provided to It as valid resources that can be carried and added to a new building. If 5 logs were provided, 5 logs will spawn, if only 2 logs were provided, only 2 will spawn...

Updates from previous version

- Refactored interaction system:

Now it works purely from HUD class, making better use of widget blueprints and removing many functionalities from Player Character.

Moreover, you can add the **NEW** interactable interface to any Actor to allow interactions on It, even if it's not a built in building.

- Abstracted functionalities by adding interfaces to "buildables":

This allows you to add building functionalities to Actors that do not extend from “BuildingBase”.

- Name and String variables replaced by “Text” ones:

This allows you to easily manage translations without modifying the Building System Core

- More options to interactions

Now you can decide if the interaction option should be shown to player when its disabled or be hidden.

Also, thanks to the Interfaces abstraction, now you can easily configure **per actor** which interactions to show based on your own criteria.

- Debug info on screen for objects (3d widget)

Now you can toggle debug for individual objects/structures to see resources needed/provided, integrity (health) and other useful info.

This is usefull as **example** on how to create your own info panels for buildings.

- Code documentation website:

Thanks to Kantan’s documentation plugin, now a website with functions and classes documentation is provided, in a visual way similar to Unreal Engines official docu site.

You can access this site here: https://nesjett.github.io/UnrealEngineBuildingSystem/

- Improved materials

Added **outline material** for objects being interacted, making the overall experience more user friendly.

- Improved examples:

Ground snapping & alignment example now works properly on all surfaces, including landscapes.

- Reworked interaction icons:

Better UI out of the box.

- Ready to carry and add more than 1 resource at a time

- Added feedback messages on user actions

Based on data tables, now is built in with the system the hability to add screen messages on different user feedback (for example when adding a material, or attemtpting to add one that failed). Ready for translations!

Also supports colors for messages

- Save game support

Now the system will save It's building states automatically with Easy Multi Save plugin, or any other that uses "save game" property in Unreal Engine

Creating new materials

1. Create a new resource BP (you can copy "Resources_Log" BP
2. Set the property "Resource Type" of the BP copy to the resource you want it to be (for example to "MetalStone")

   Note: The Resource Type is a enumeration, you can add as many types as you want to It.

Integrating into existing project

There are 2 options for this, both are easy:

1 - Inheriting from ABS_Character

This is the easiest way, as ABS_Character extends from Character and already includes all the functionality needed for the system to work.

The final class hierarchy could look like this: **Actor > Pawn > Character > ABS_Character > YourCustomCharacter**

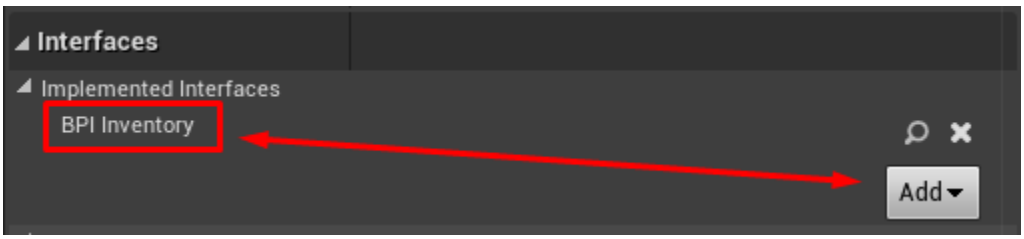
There is no additional setup required for this method.

2- Adding Advanced Building System to your custom character/pawn

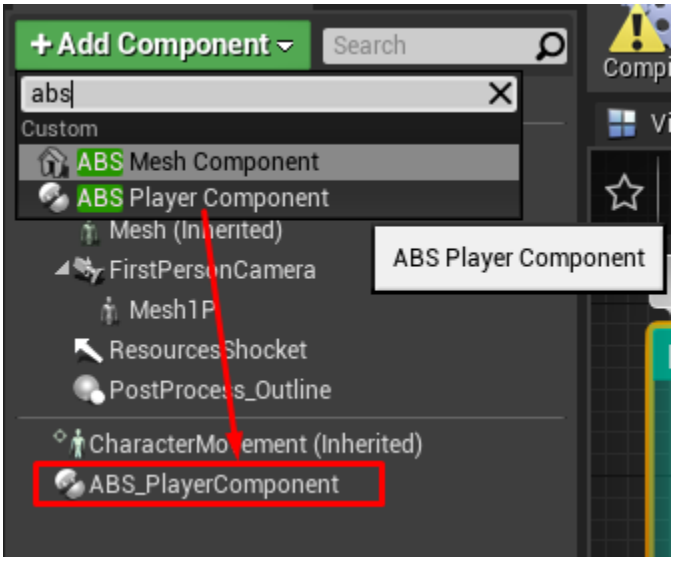
If using inheritance is not an option for some reason (For example if you are not extending your Player from character but from Pawn instead)

This are the basics you must do:

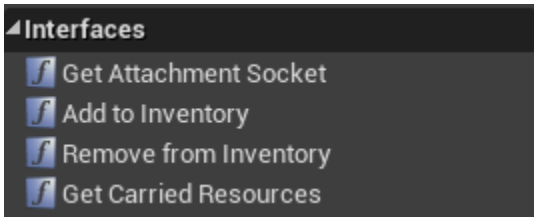
1 - Add the BPI_Inventory interface to your player class



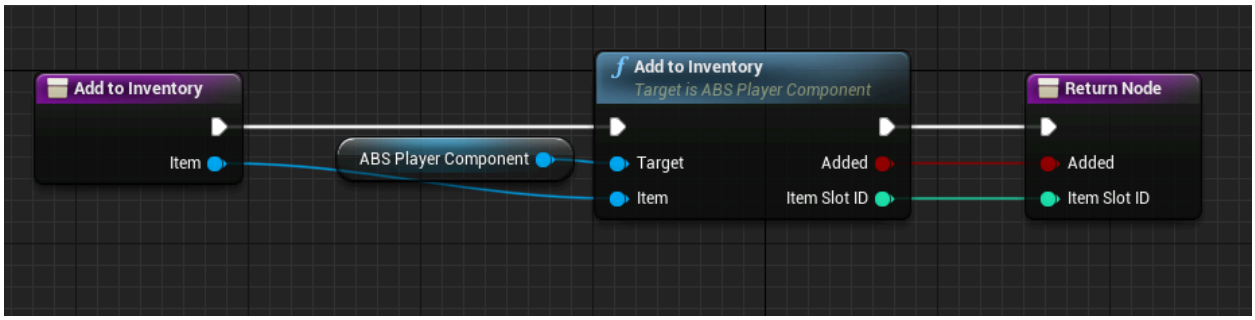
2 - Add the ABS_PlayerComponent to your Actor



3 - Implement the 4 methods this interface includes

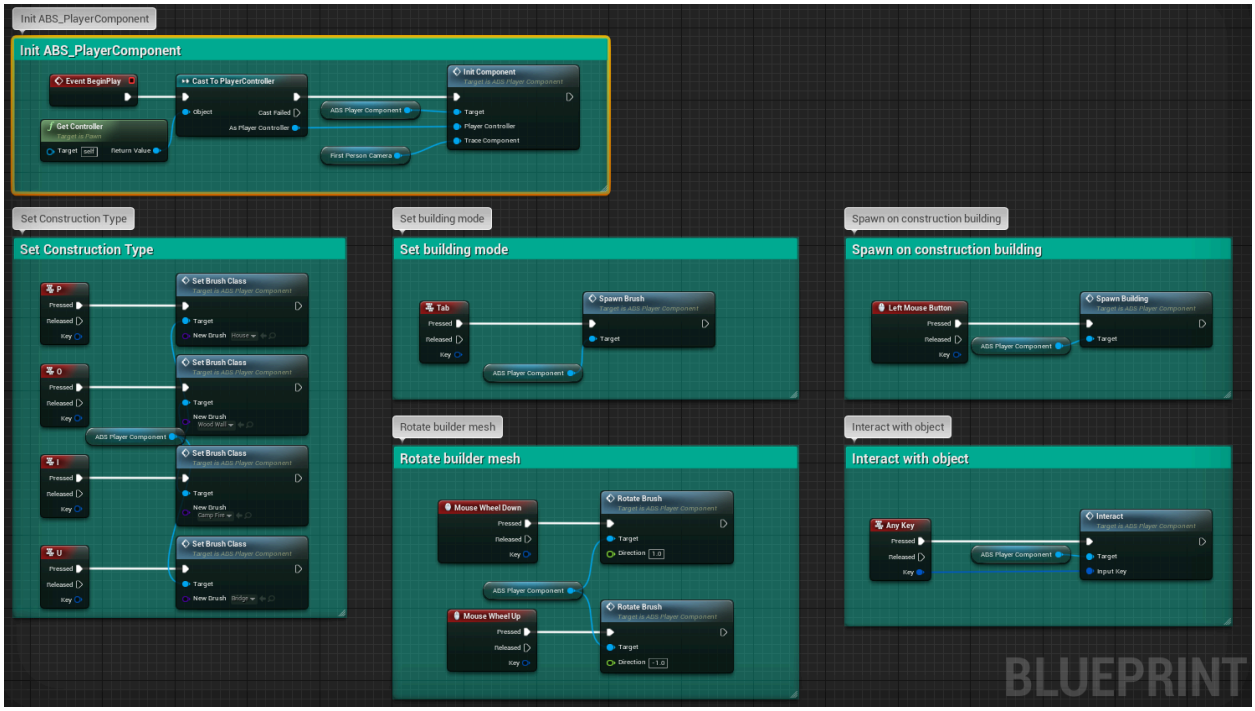


This methods only get the data from the ABS_PlayerComponent, so you just need to add a call to it. For example:



4 - Copy the inputs and initialization from the ABS_Character

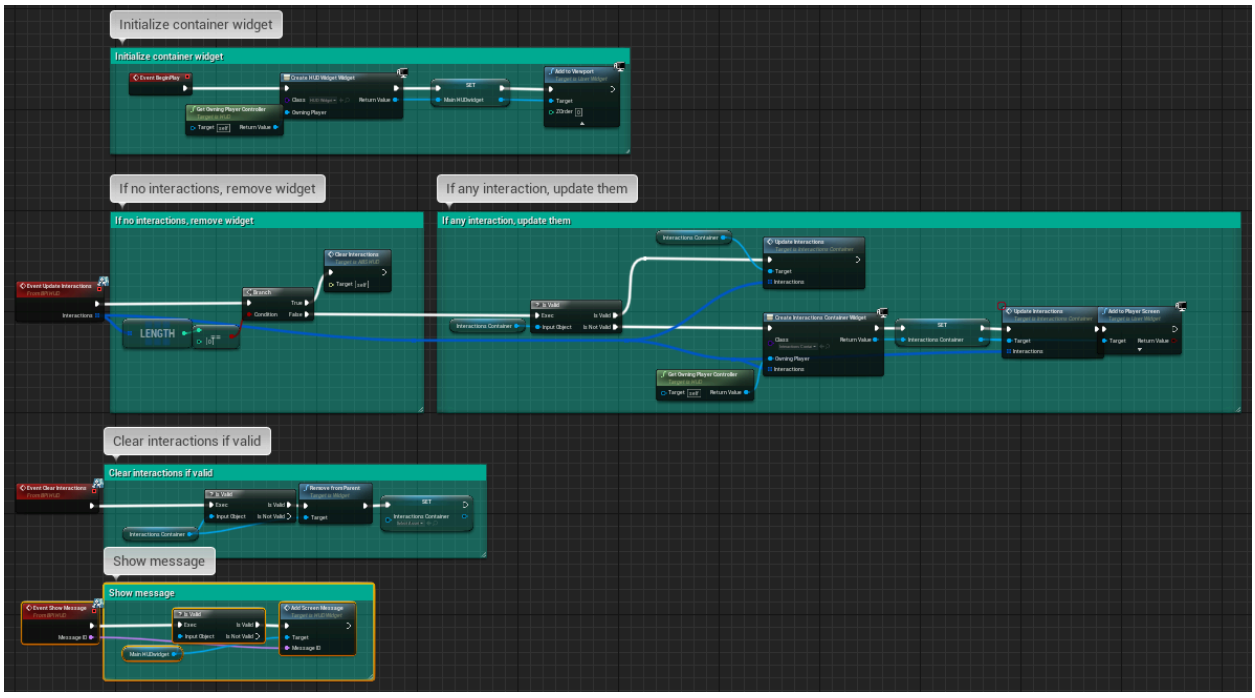
No extra functions or any code at all, just binding your inputs to the ABS_PlayerComponent as follows: **COPY AND PASTE WORKS**!



5 - (optional) Make use or extend the ABS_HUD

If you want to use the on screen messages provided by the system, you must use, extend or implement the code in the ABS_HUD.

Wondering how It looks like…?



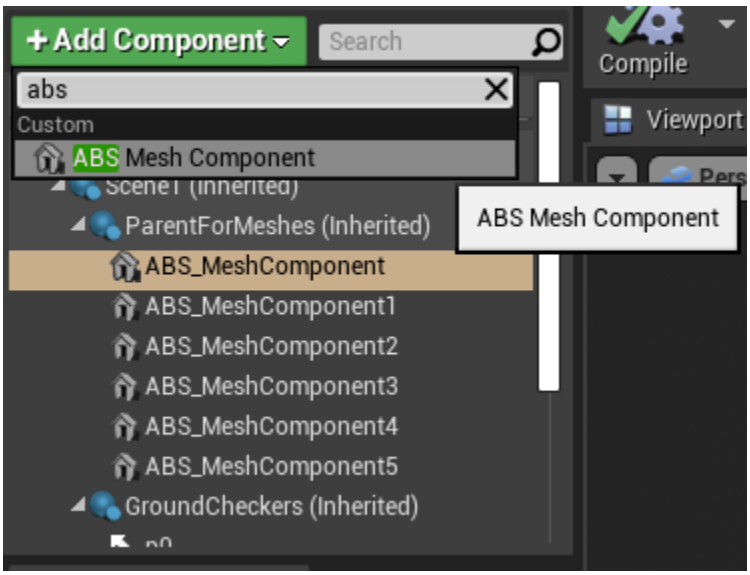
This is **all ABS_HUD includes**. Adding this to your own HUD class is as easy as copy and paste this code. NO EXTRA FUNCTIONS or anything need to be done manually.

This is ALL you need to make the system work in your existing project. Isn't it easy..?

Creating new structures

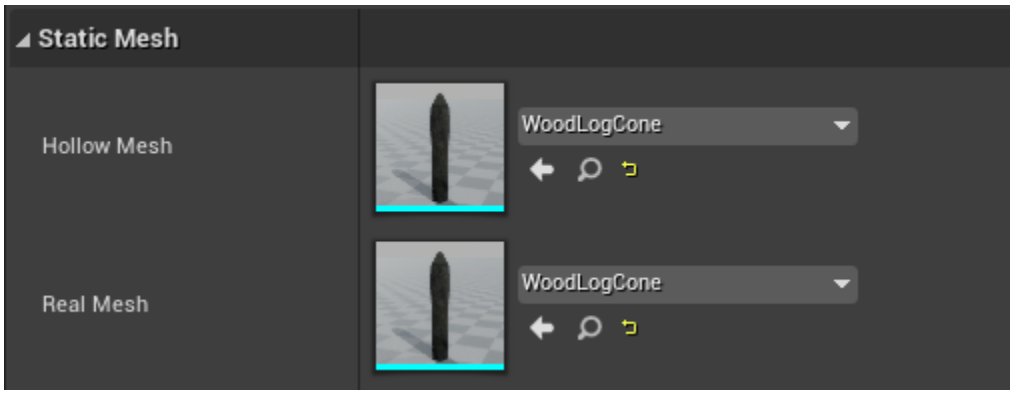
To build a structure you have to keep in mind this steps:

1. Create a new blueprint, inheriting from **BuildingBase**
2. Add as many ABS_MeshComponents as needed to build your structure

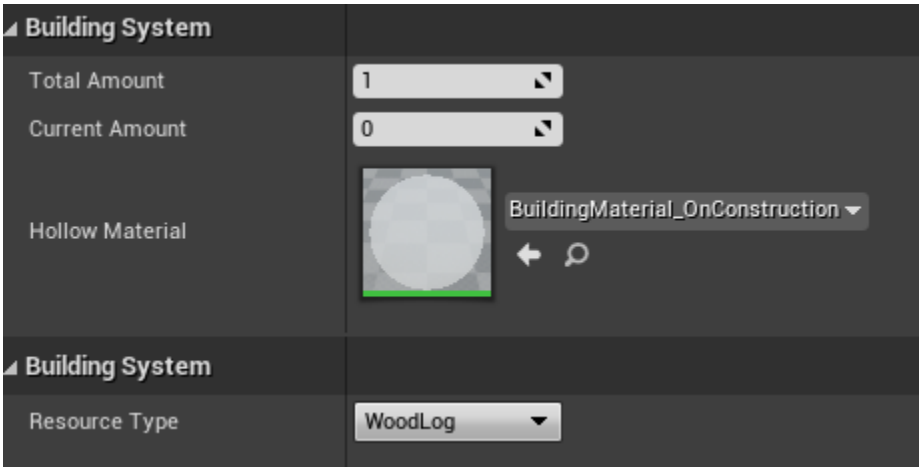


3- Configure each component

3.1 - Set the meshes:



3.2 - Set the resources data

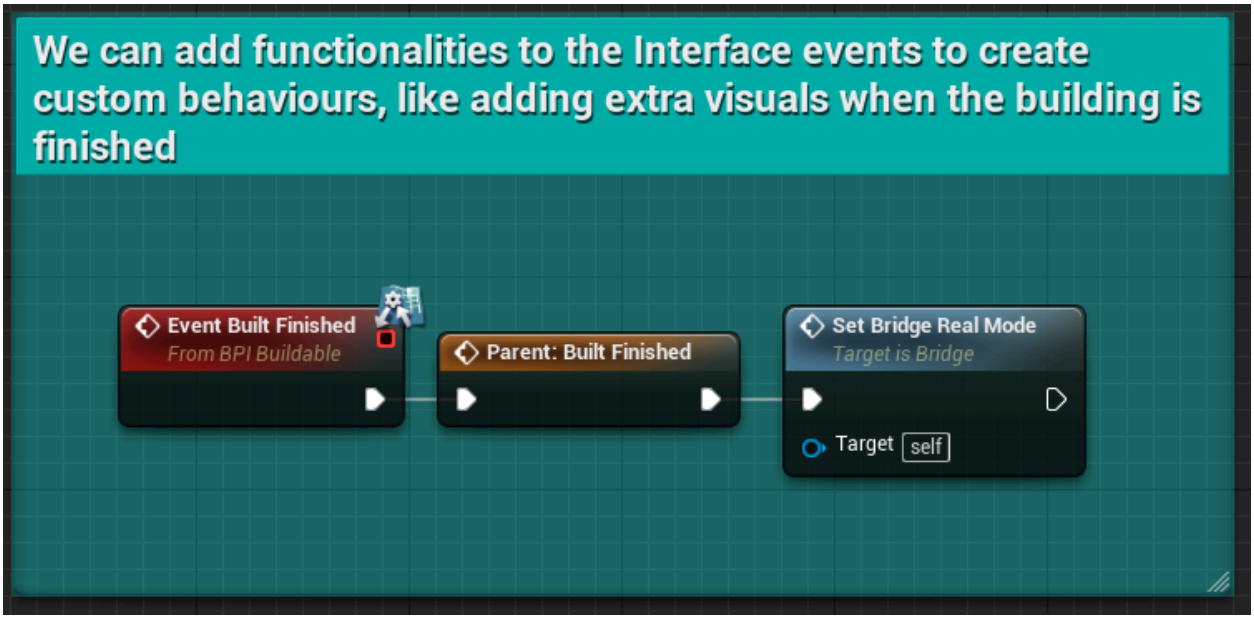


IMPORTANT: Remember that the order in which each part of the structure are going to be visibly "constructed" corresponds to the hierarchy order in which you added the mesh components in the blueprint (top children goes first). In other words, make the building chain of ABS_MeshComponents be children of each other to make them be build after the parent.

Single piece & Mixed structures

This structures has one or more body part that is not related to any material (they have components that are not ABS_MeshComponents) making them only become visible (real mesh to show up) when all the other materials have been added.

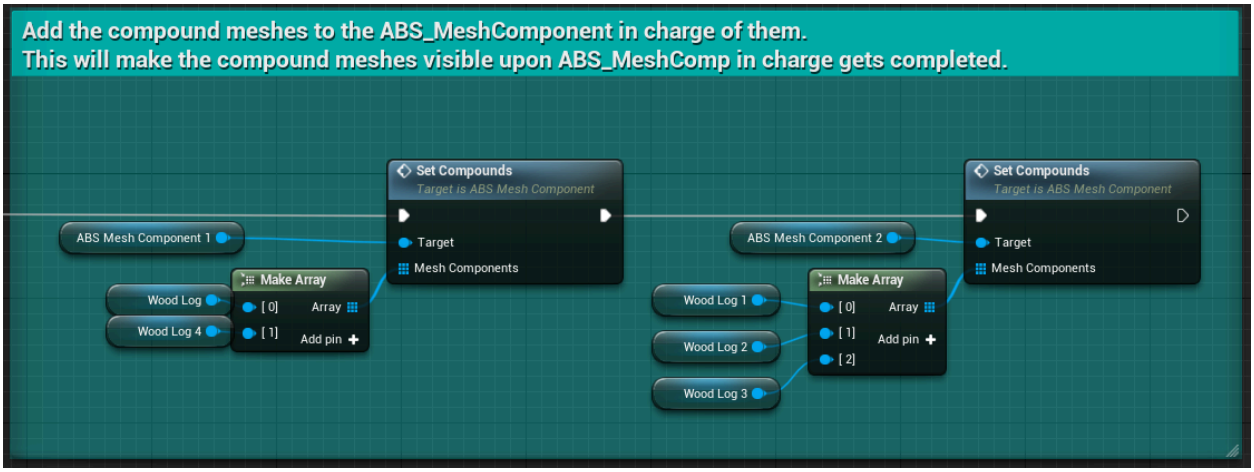
This can be done overriding the event finished:



Compound components

This are components that will become visible/hidden when the “owner” ABS_MeshComponent gets completed.

To add those compounded components to the ABS_MeshComponent you must do so in the construction script:

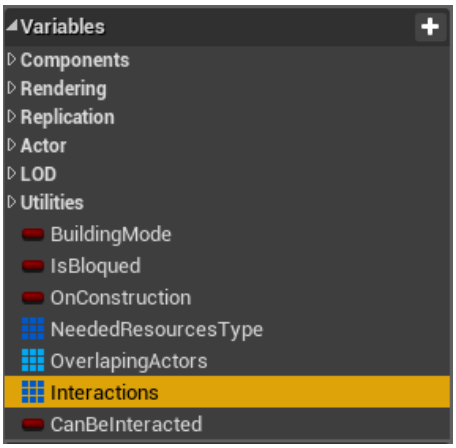


Once again, the rest will be handled automatically!

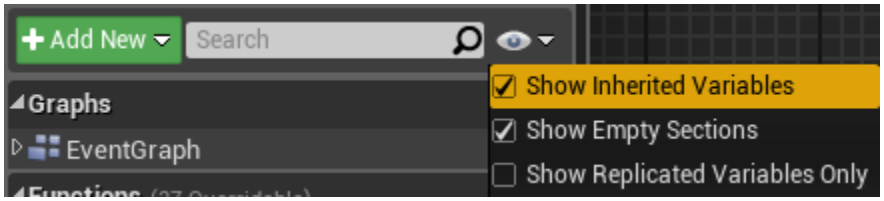
Very easy, isn't It?

Adding more interactions to a structure/material

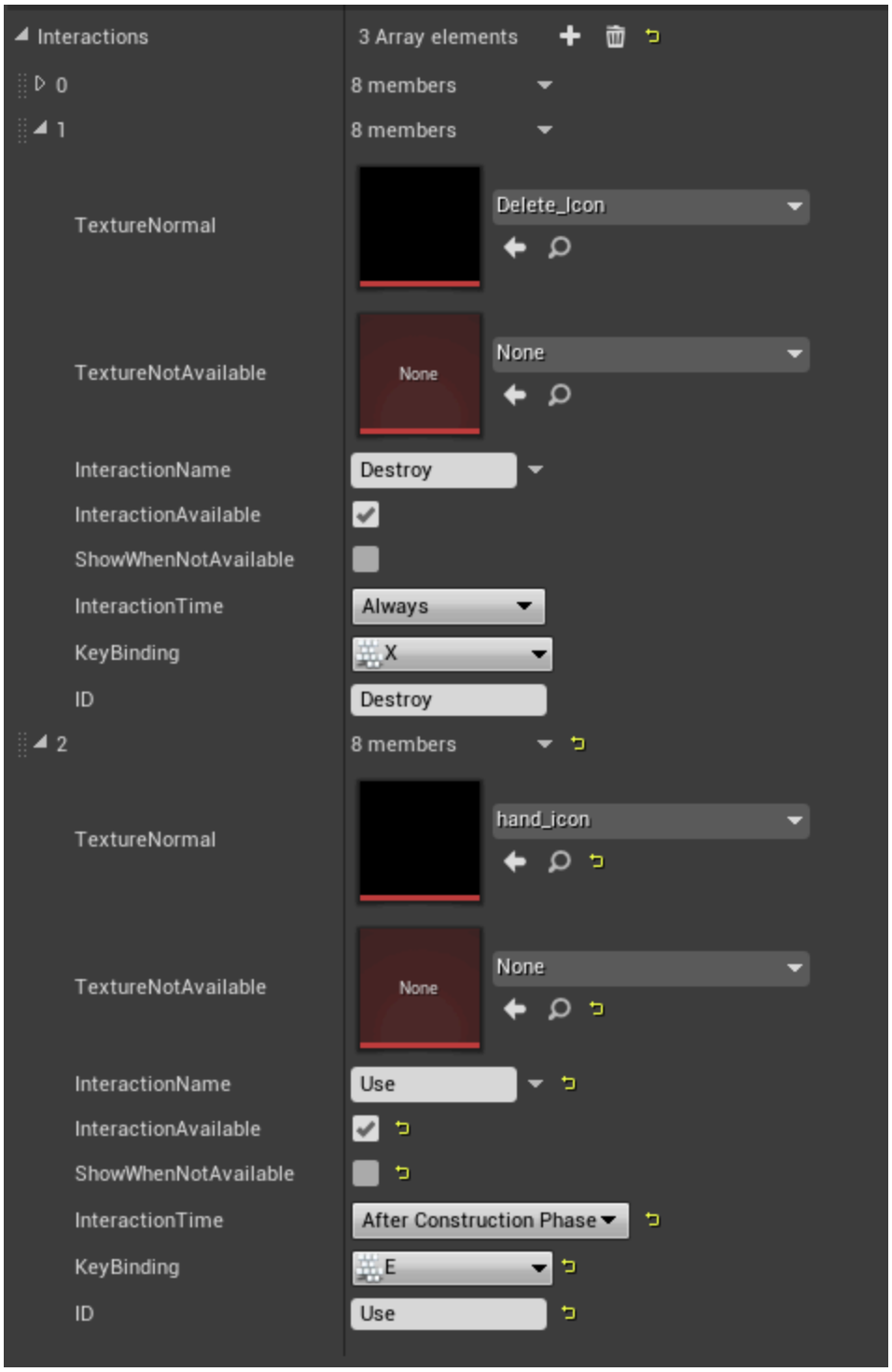
Find the variable (array) called "Interactions" and add as many values as you want:



WARNING: If you cant see this variable you will have to activate the option “Show inherited variables”



Example of Interactions that you can find in the **CAMPFIRE example**:


Interactions
3 Array elements
0
8 members
1
8 members
TextureNormal
Delete_Icon
TextureNotAvailable
None
None
InteractionName
Destroy
InteractionAvailable
ShowWhenNotAvailable
InteractionTime
Always
KeyBinding
X
ID
Destroy
2
8 members
TextureNormal
hand_icon
TextureNotAvailable
None
None
InteractionName
Use
InteractionAvailable
ShowWhenNotAvailable
InteractionTime
After Construction Phase
KeyBinding
E
ID
Use

F.A.Q.

1. Can I build bigger structures?

   Yes, there is no limitations on size but you have to keep in mind that if the structure is reaaally big, the look won't seem so realistic (because you may add a log on one corner and the it could be added on the other corner of the structure)

   For accomplish something like this, It's recommended to create a group of different buildings and control whether all of them are finished building to execute a specific functionality.

   You could also easily modify the system to add the material to the currently pointing ABS_MeshComponent, in order of following the component hierarchy order (Explained in above in the document)

2. Can I make the structures have functionalities after being build?

   Yes! There is an example of campfire where you can bring it on fire after finishing building it. You can add as many interactions as you want to each structure, also at runtime!

3. Does it works in online multiplayer?

   No, actually the system only works for single player.

4. In the video I don't see effects, is it easy to add particles and sounds to different kind of events in the gameplay?

   Yes, you have different events to override and execute feedback effects, like the event "Interact" or "Built Finished"