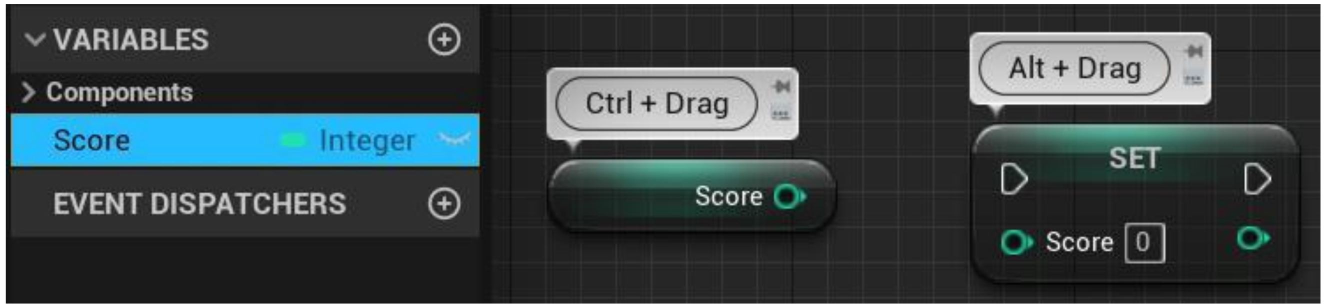
Blueprints Tips

9th Week, 2022

Blueprint Editor shortcuts (1)

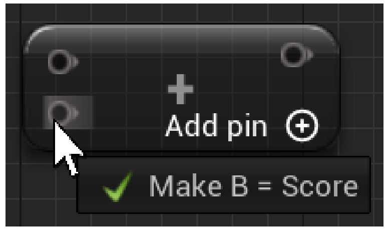
› Shortcuts to create **GET** and **SET** nodes

- Ctrl + Drag: to create **GET** node
- Alt + Drag: to create **SET** node



< Shortcuts to create GET and SET nodes >

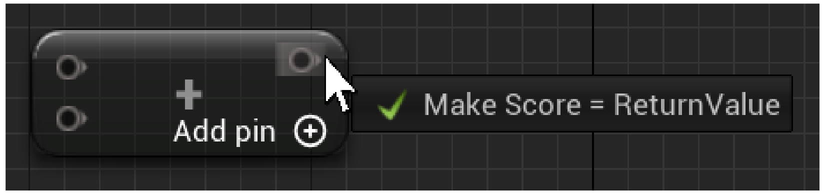
- Dropping a variable on an input parameter pin: to create a **GET** node



< Dragging a variable and dropping it on an input pin to create a GET node >

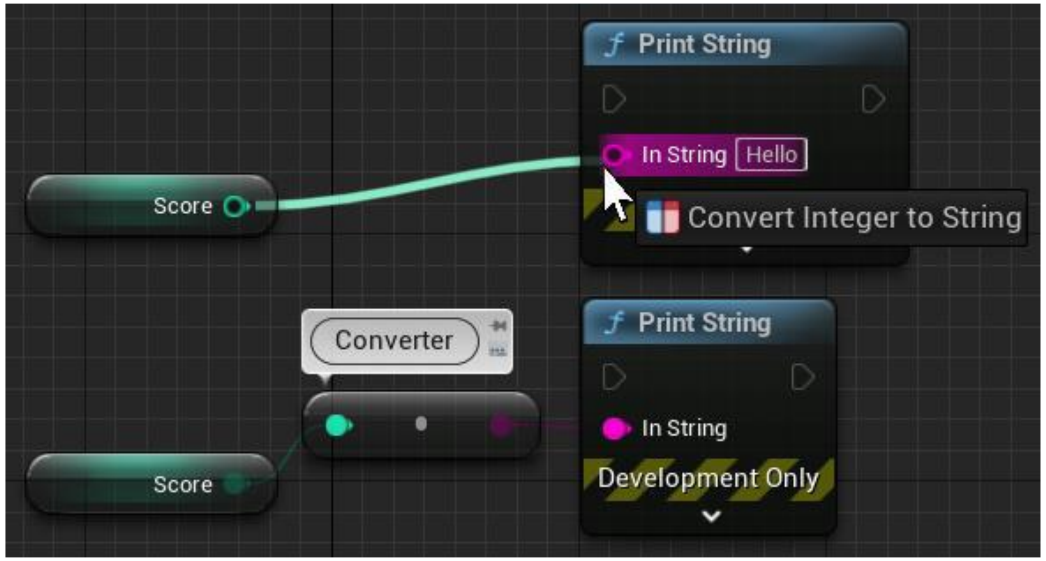
Blueprint Editor shortcuts (2)

- Dropping a variable on an output parameter pin: to create a **SET** node



< Dragging a variable and dropping it on an output pin to create a SET node >

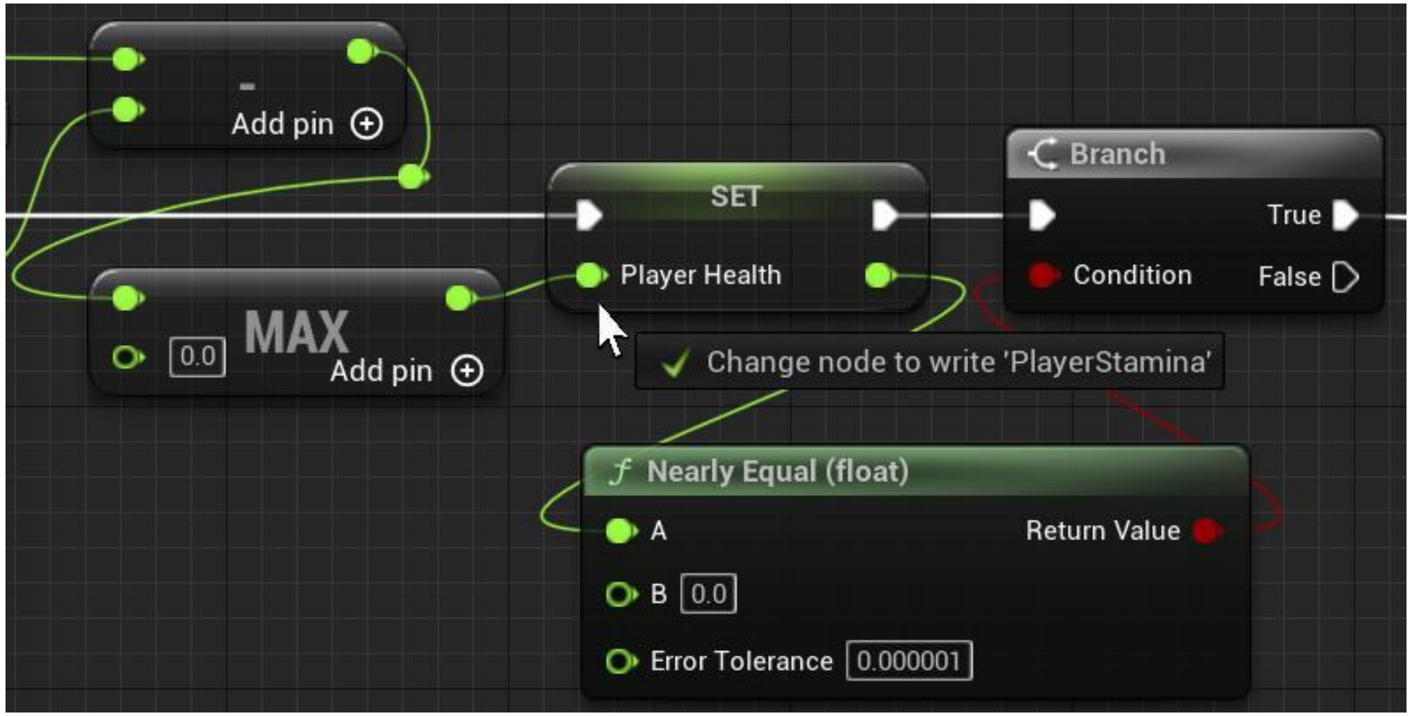
› The Blueprint Editor has an automatic type conversion system.



< Creating a converter node >

Blueprint Editor shortcuts (3)

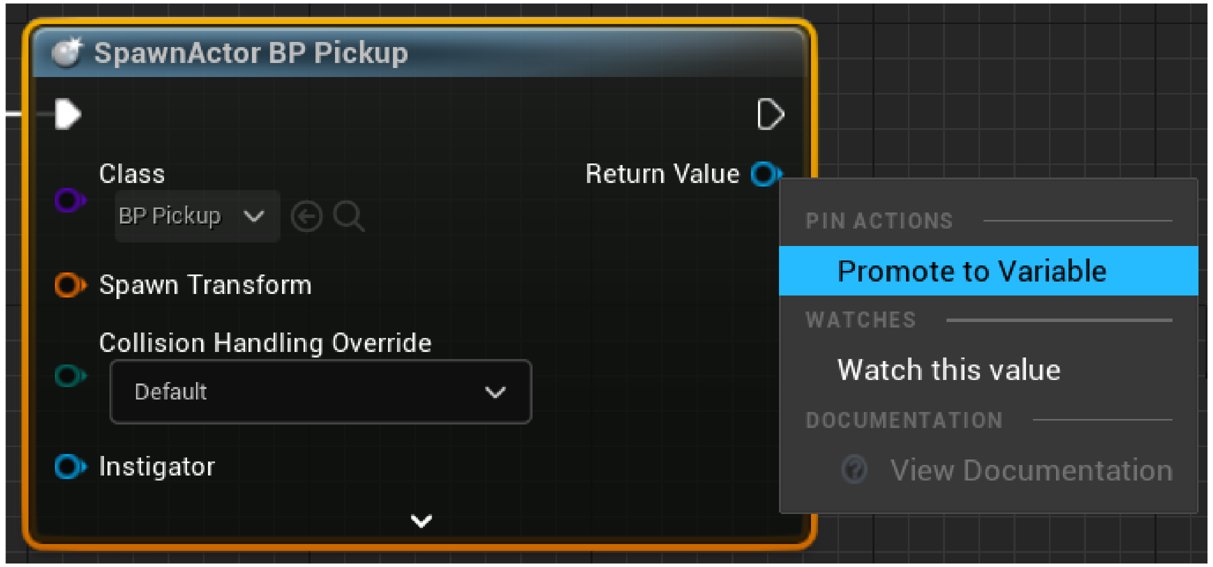
› In the Blueprint Editor, it is possible to change an existing node for another node that uses the same variable type without breaking the connections.



< Changing a node and keeping all connections >

Blueprint Editor shortcuts (4)

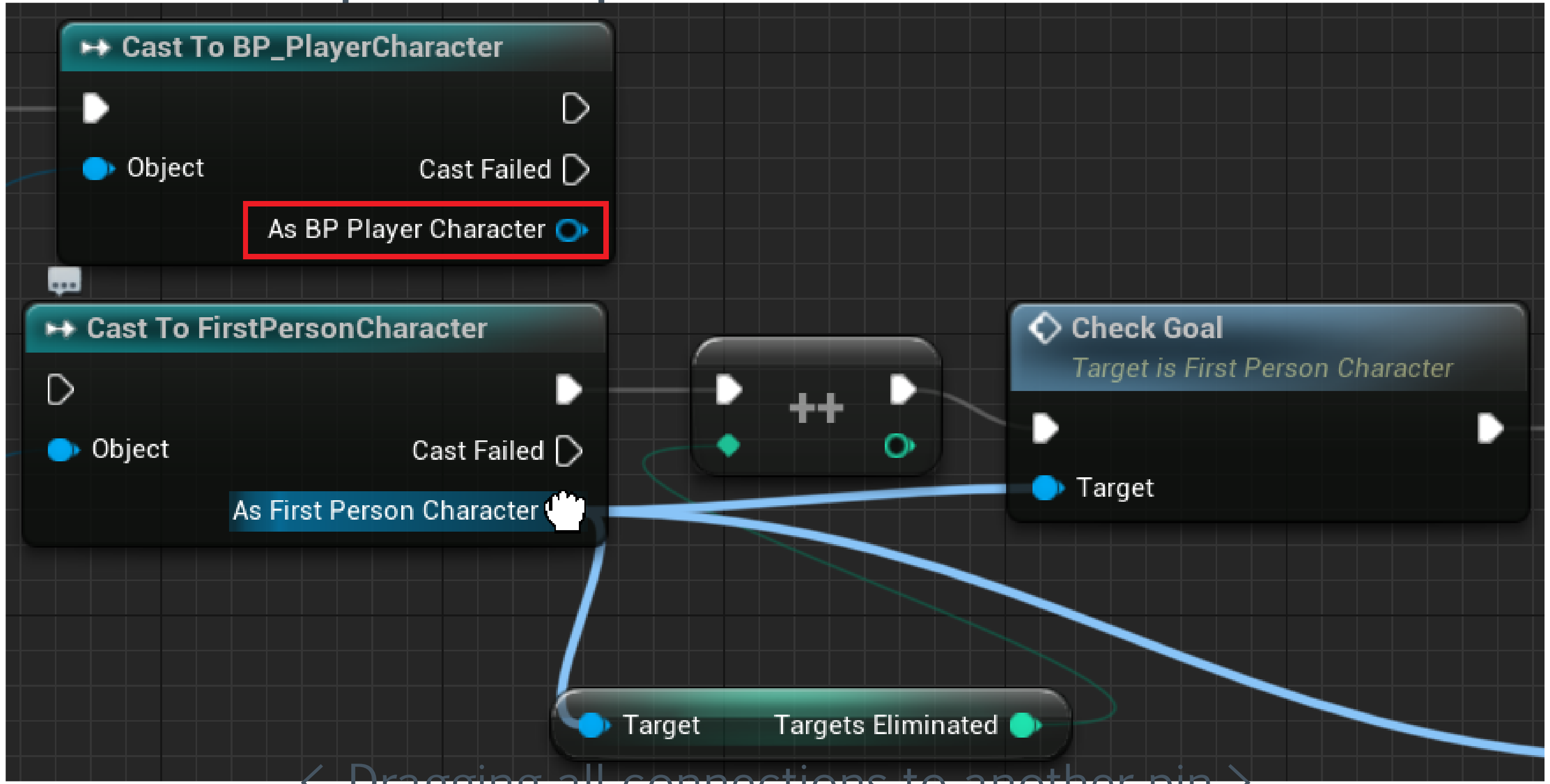
› The **Promote to Variable** option: A shortcut to create variable based on the type of an input or output pin of a node.



< Promoting the return value to a variable >

Blueprint Editor shortcuts (5)

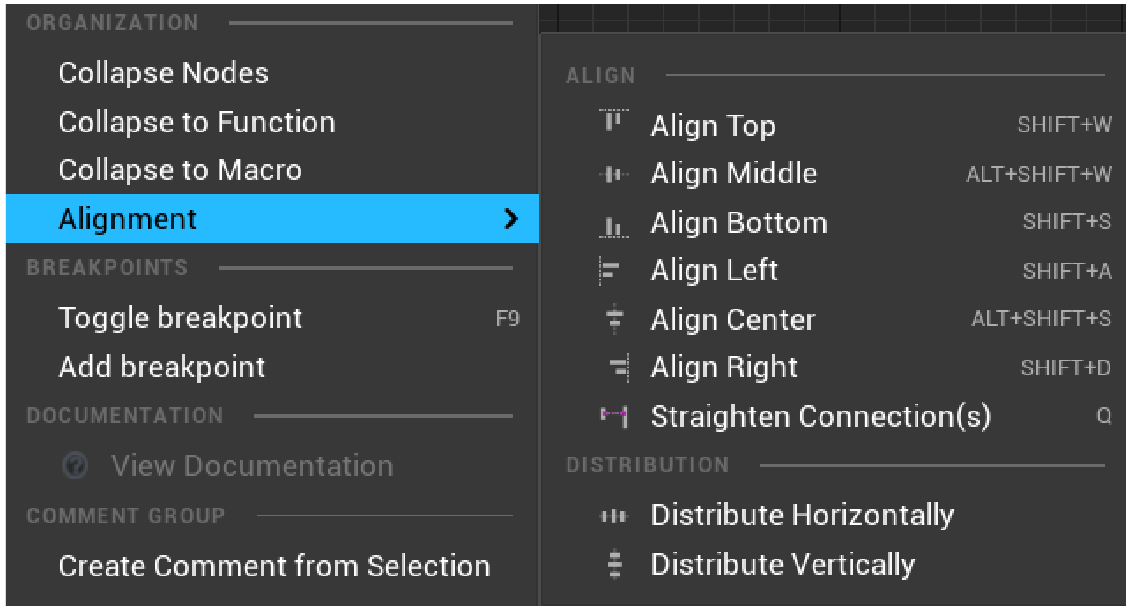
- Alt + Click: to break all the connections of a pin
- Ctrl + Drag: to move all the connections of a pin to another compatible pin


Cast To BP_PlayerCharacter
Object
Cast Failed
As BP Player Character
Cast To FirstPersonCharacter
Object
Cast Failed
As First Person Character
++
Check Goal
Target is First Person Character
Target
Target
Targets Eliminated

< Dragging all connections to another pin >

Blueprint Editor shortcuts (6)

› The Blueprint Editor offers several options for node alignment.



ORGANIZATION

Collapse Nodes

Collapse to Function

Collapse to Macro

Alignment

BREAKPOINTS

Toggle breakpoint	F9
Add breakpoint	

DOCUMENTATION

View Documentation

COMMENT GROUP

Create Comment from Selection

ALIGN

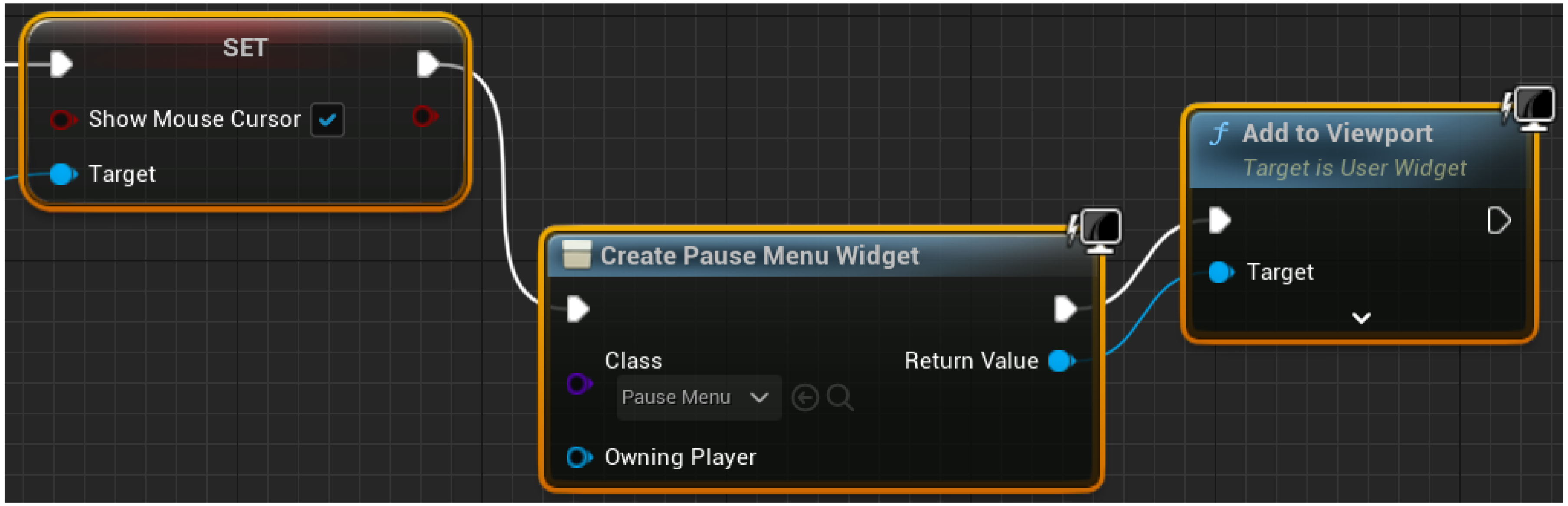
Align Top	SHIFT+W
Align Middle	ALT+SHIFT+W
Align Bottom	SHIFT+S
Align Left	SHIFT+A
Align Center	ALT+SHIFT+S
Align Right	SHIFT+D
Straighten Connection(s)	Q

DISTRIBUTION

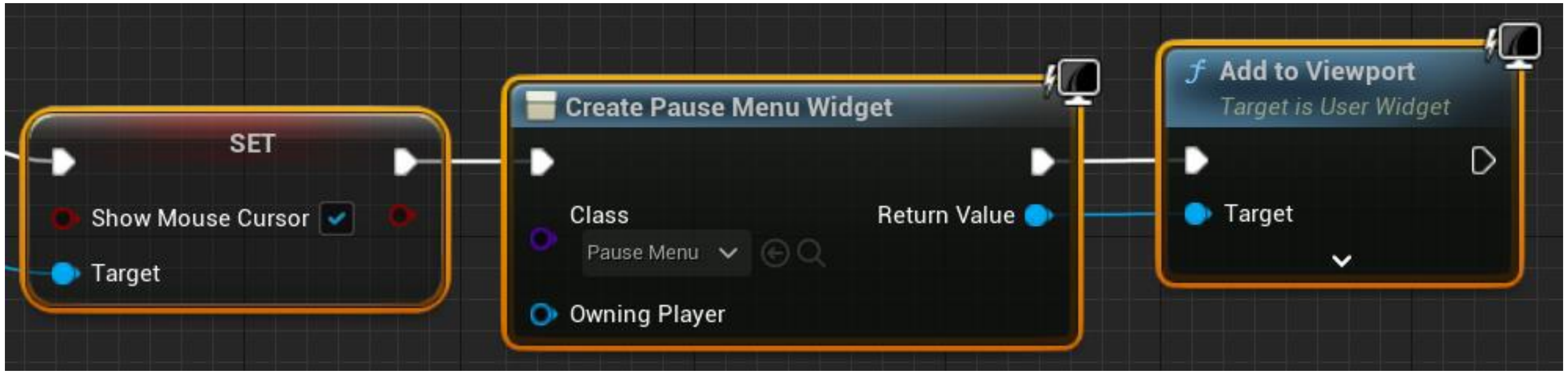
Distribute Horizontally

Distribute Vertically

< The Alignment options >



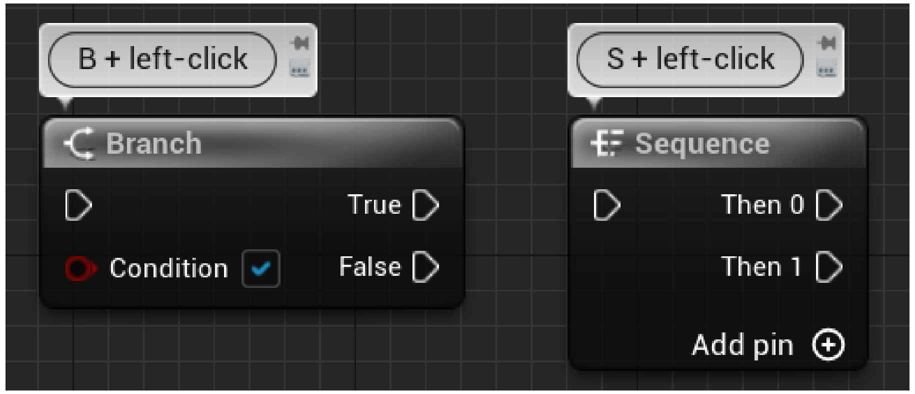
< These nodes will be aligned >



< The nodes after applying Straighten Connection(s) >

Blueprint Editor shortcuts (7)

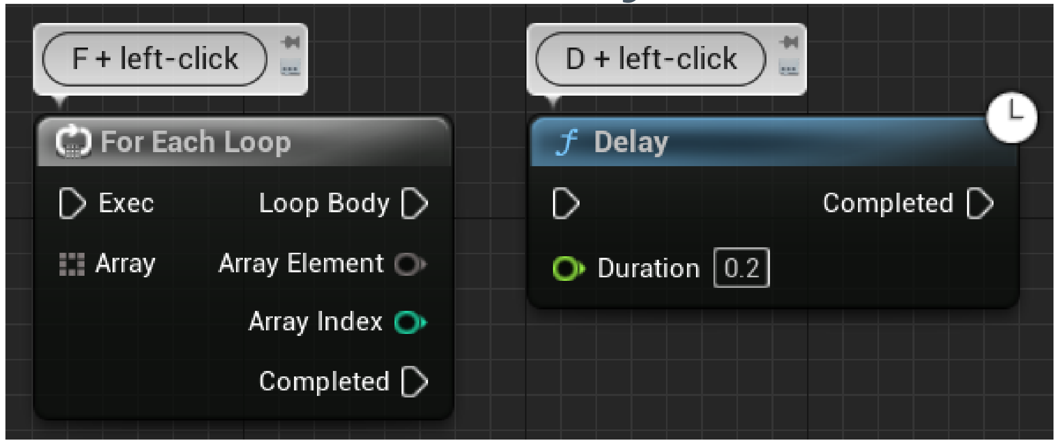
- Shortcut keys to create some common nodes in Blueprints
  - B + left-click: to create a **Branch** node
  - S + left-click: to create a **Sequence** node



< Shortcuts for Branch and Sequence nodes >

Blueprint Editor shortcuts (8)

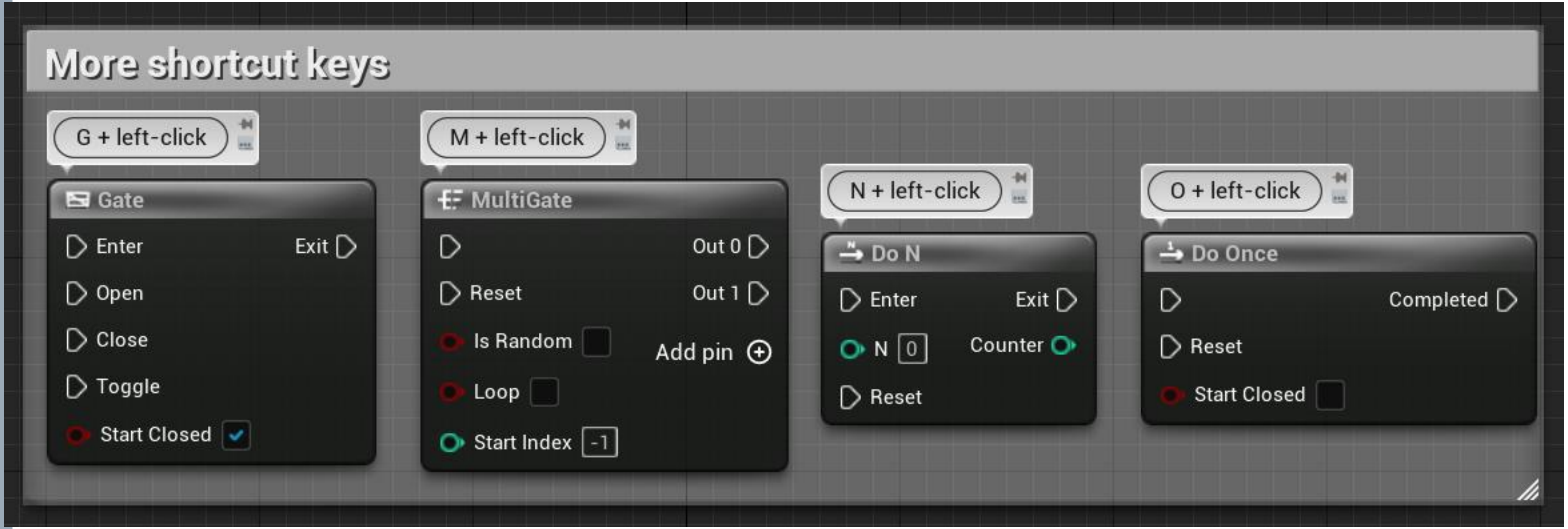
- F + left-click: to create a **For Each Loop** node
- D + left-click: to create a **Delay** node



< Shortcuts for the For Each Loop and Delay nodes >

Blueprint Editor shortcuts (9)

› To create comment box around some nodes, first select the nodes, then right-click on one of the selected nodes and select the **Create Comment** option from **Selection**, or you can just press the C key.

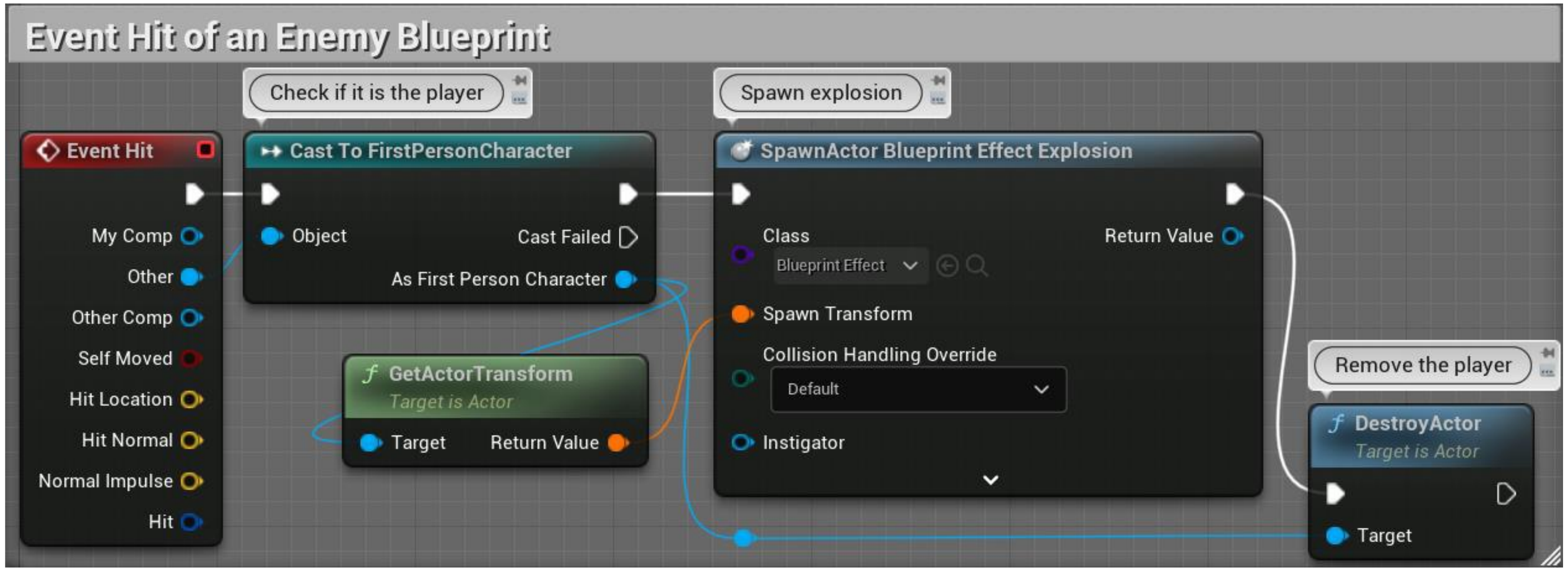


< Shortcuts for some flow control nodes >

Blueprint best practices (1)

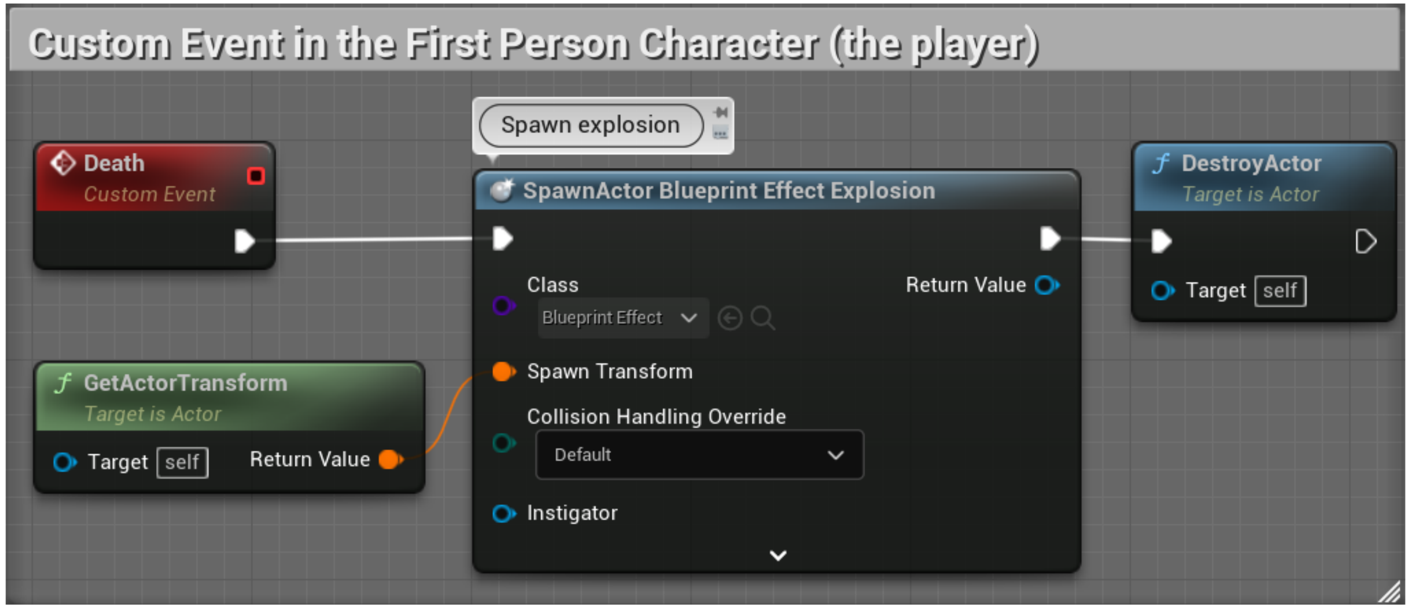
› Blueprint responsibilities
  - When creating a Blueprint, you need to decide what its responsibilities will be.
  - This refers to what it will do and what it will not do.
  - You need to make the Blueprint as independent as possible.
  - A Blueprint must be responsible for its internal state.

Blueprint best practices (2)

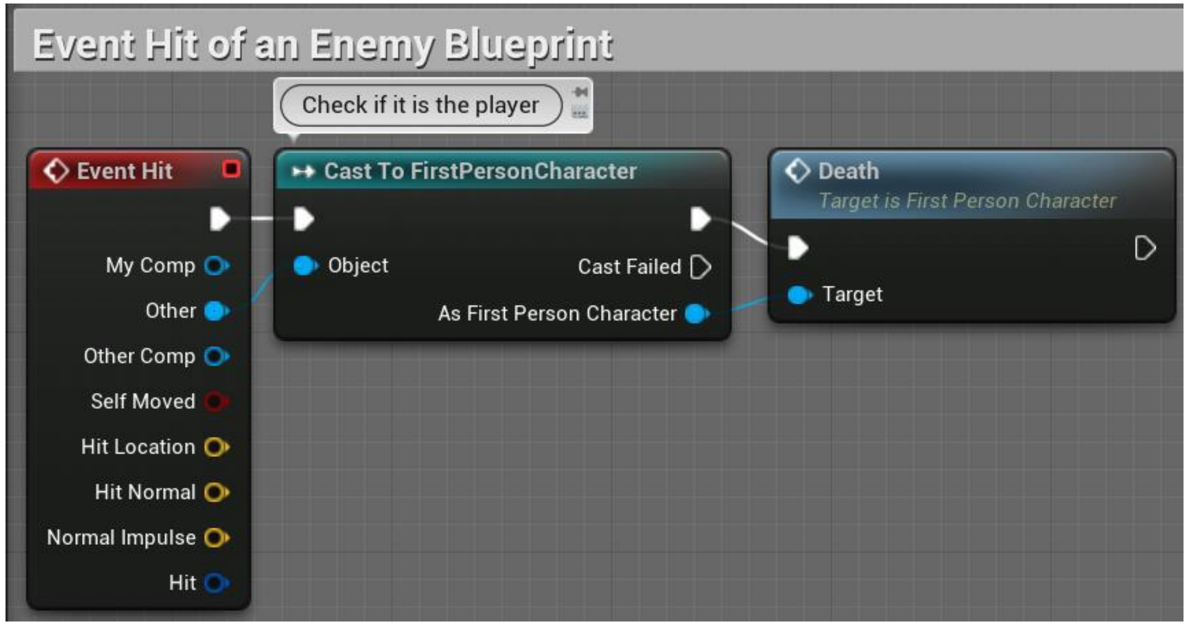


< Event hit o an enemy Blueprint >

– But you decide to change the way the player dies.



< Creating the Death event in the FirstPersonCharacter Blueprint >


Event Hit of an Enemy Blueprint
Check if it is the player
Event Hit
My Comp
Other
Other Comp
Self Moved
Hit Location
Hit Normal
Normal Impulse
Hit
Cast To FirstPersonCharacter
Object
Cast Failed
As First Person Character
Death
Target is First Person Character
Target

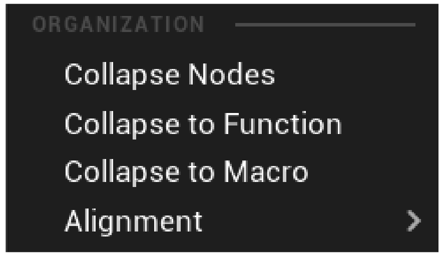
< New Version of Event Hit of Enemy Blueprint >

Blueprint best practices (3)

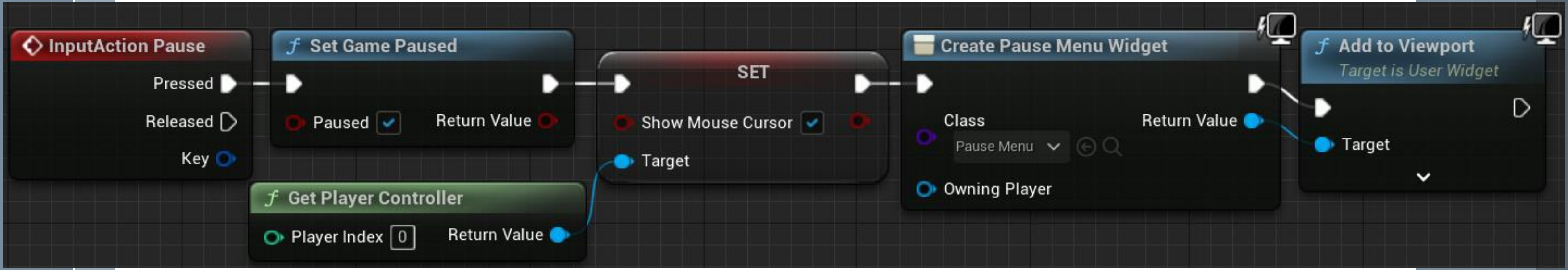
- A Level Blueprint must be used only for logic and situations specific to one Level.
  - If your game rules logic changes, then you will need to all the Level Blueprint of the new Level.
  - A better place to implement game rules logic is in a `GameMode` Blueprint class.
  - The logic for other actors should be implemented in Blueprint class rather than being implemented in the Level Blueprint.

Blueprint best practices (4)

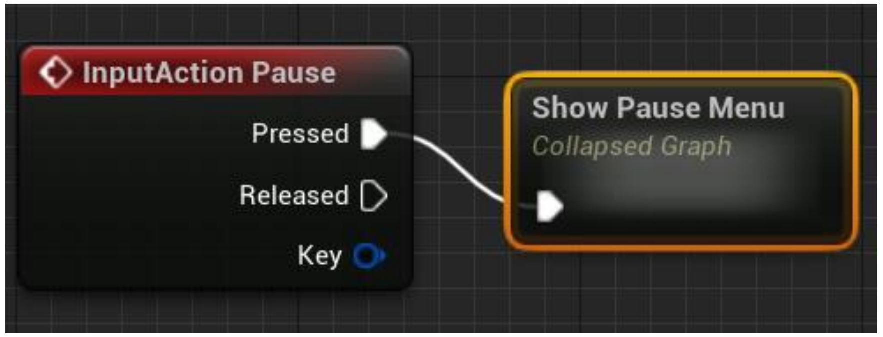
- Managing Blueprint complexities
  - A Blueprint **EventGraph** can become very complex and scary.
  - **Abstraction** is used to handle complexities by hiding low-level details, allowing the developer to focus on a problem at a high abstraction level.
  - A simple way to apply abstraction: to select a group of nodes and convert them into a collapsed graph, Function, or Macro.
    - To convert the nodes, right-click on the selected nodes.



< Collapse options >



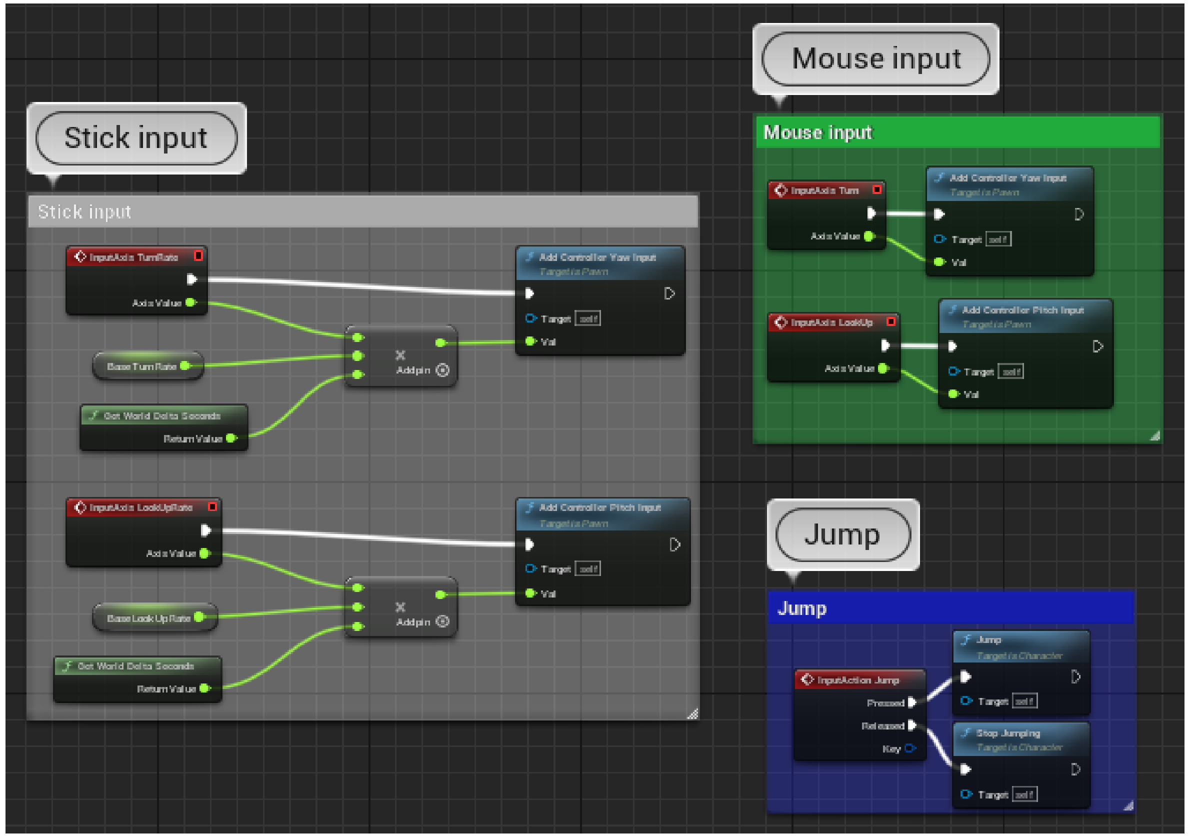
< Node used to show the Pause Menu >



< The nodes were converted into a collapsed graph >

Blueprint best practices (5)

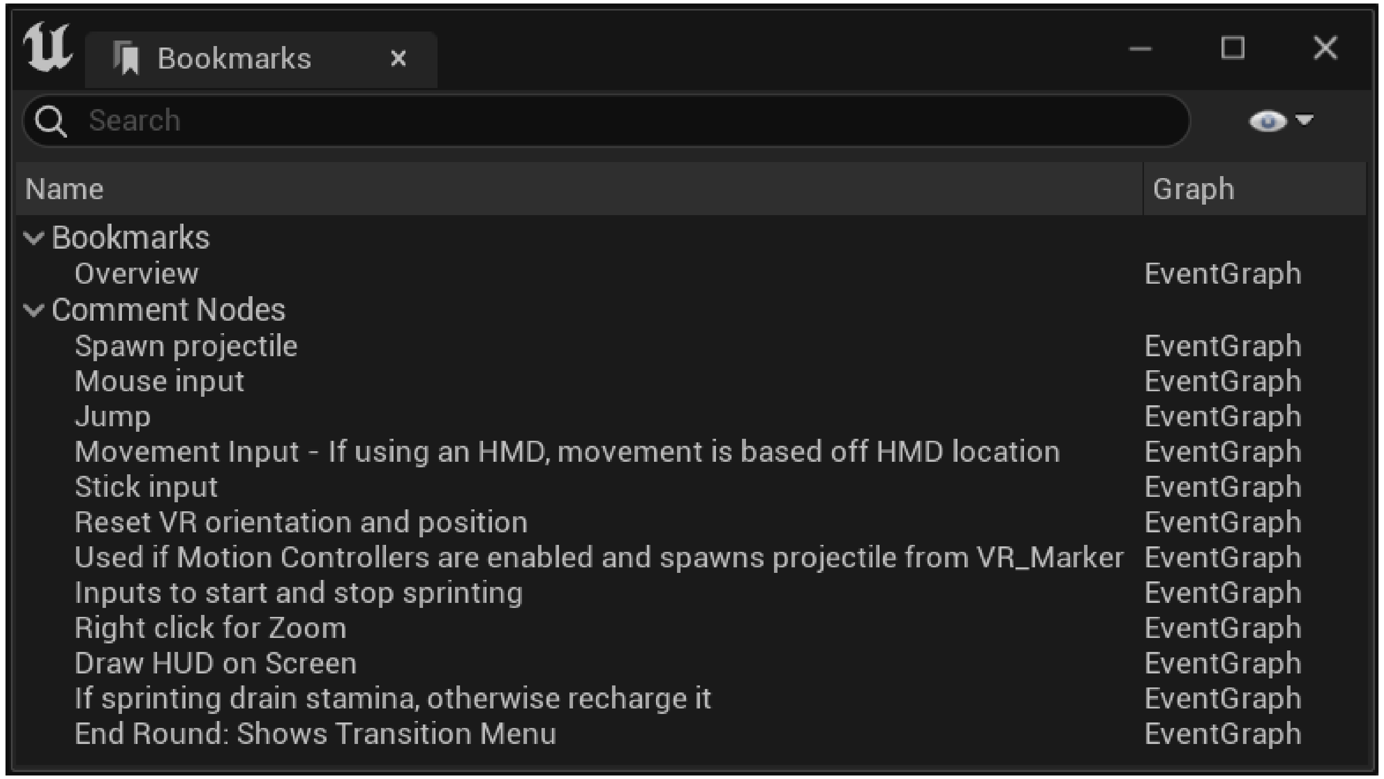
- Another handy tool that can increase the readability of a complex **EventGraph** is comment box.



< The comments are visible when the EventGraph is zoomed out >

Blueprint best practices (6)

– You can see a list of the comment boxes of a graph in the **Bookmarks** window, which can be accessed from the top menu by going to **Window** > **Bookmarks**.

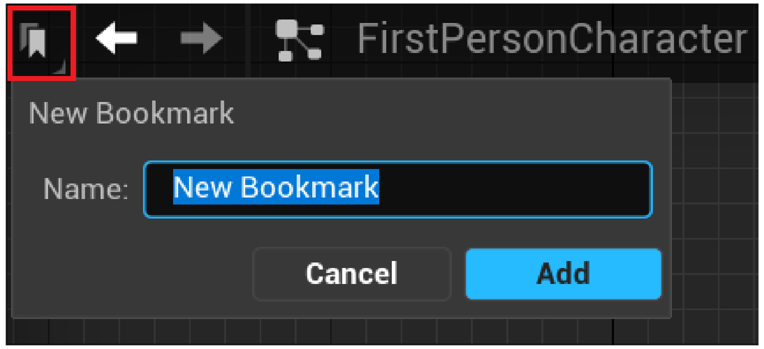


Name	Graph
Bookmarks	
Overview	EventGraph
Comment Nodes	
Spawn projectile	EventGraph
Mouse input	EventGraph
Jump	EventGraph
Movement Input - If using an HMD, movement is based off HMD location	EventGraph
Stick input	EventGraph
Reset VR orientation and position	EventGraph
Used if Motion Controllers are enabled and spawns projectile from VR_Marker	EventGraph
Inputs to start and stop sprinting	EventGraph
Right click for Zoom	EventGraph
Draw HUD on Screen	EventGraph
If sprinting drain stamina, otherwise recharge it	EventGraph
End Round: Shows Transition Menu	EventGraph

< Bookmarks window >

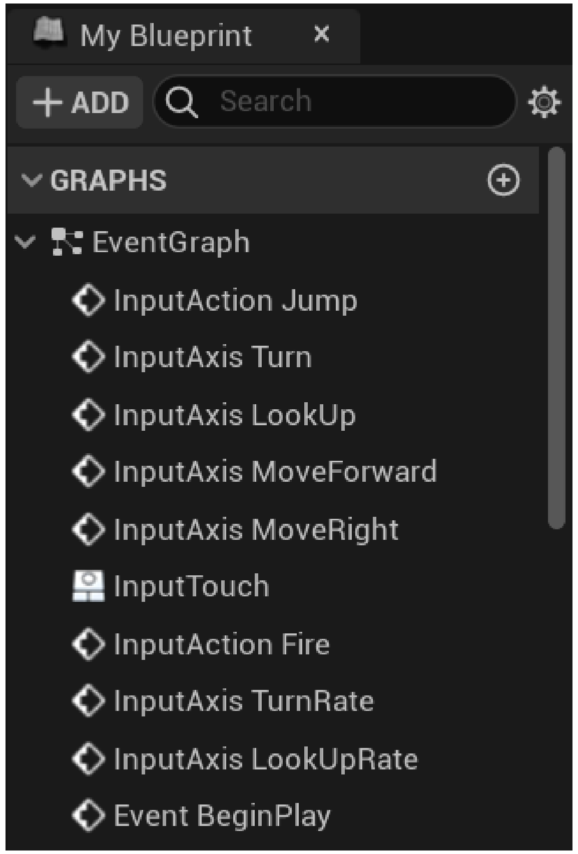
Blueprint best practices (7)

- You can create bookmarks to reference a location of the **EventGraph** by clicking on the icon located in the top left of the **EventGraph**.



< Creating a bookmark >

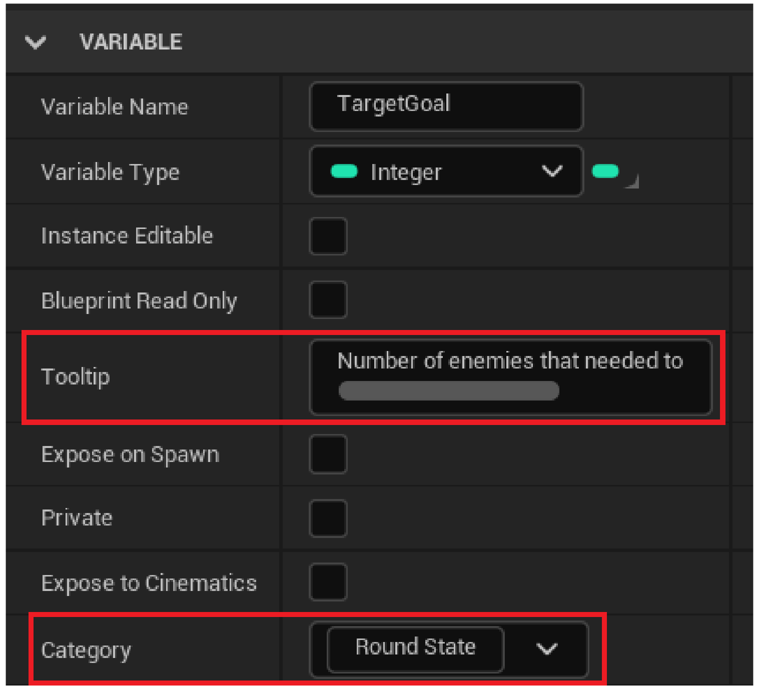
- Double-click on an event name to move the **EventGraph** to the position of the event:



< List of events in the EventGraph >

Blueprint best practices (8)

– **Tooltip** and **Category** which help you identify and organize variables.



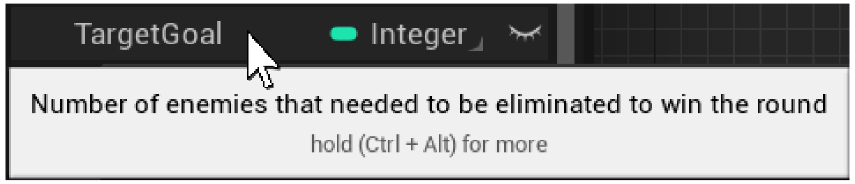
VARIABLE

Variable Name	TargetGoal
Variable Type	Integer
Instance Editable	☐
Blueprint Read Only	☐
Tooltip	Number of enemies that needed to
Expose on Spawn	☐
Private	☐
Expose to Cinematics	☐
Category	Round State

< Tooltip and Category properties >

Blueprint best practices (9)

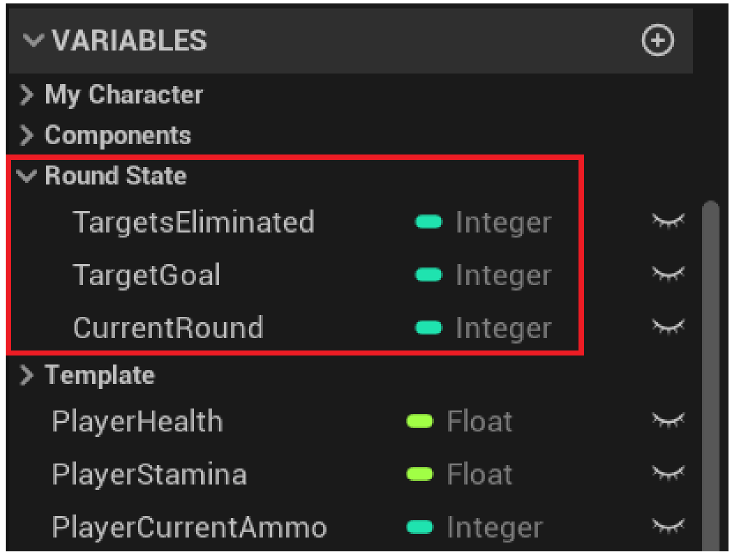
› The tooltip is shown when the mouse cursor is over the variable.



Number of enemies that needed to be eliminated to win the round
hold (Ctrl + Alt) for more

< The tooltip appears when hovering over a variable >

– You can create categories or select an existing category in the drop-down menu.

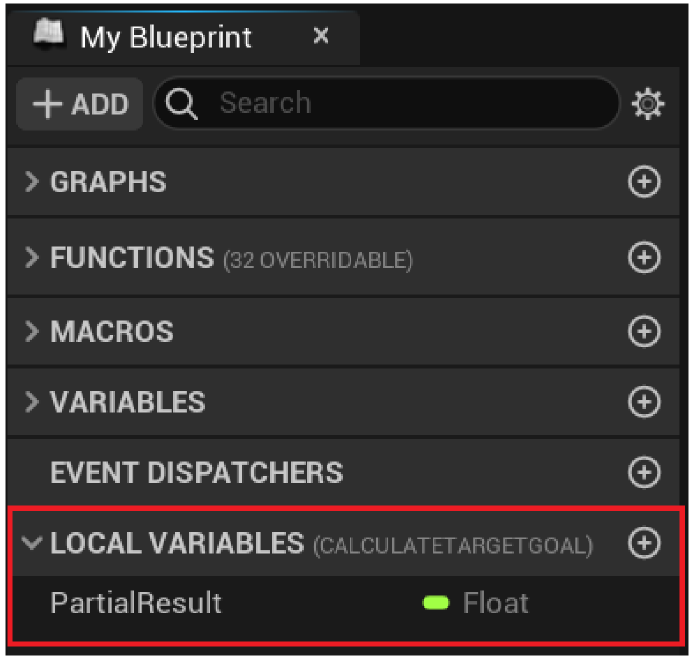


< Variables are grouped by categories >

Blueprint best practices (10)

지역변수 선언가능

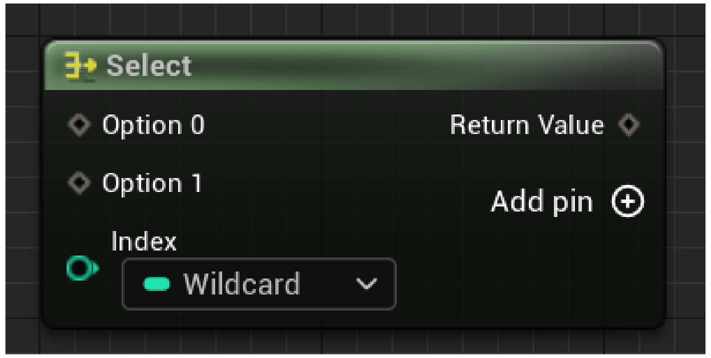
- A function allows the create of local variables (to hold temporary values), which are only visible within the function.



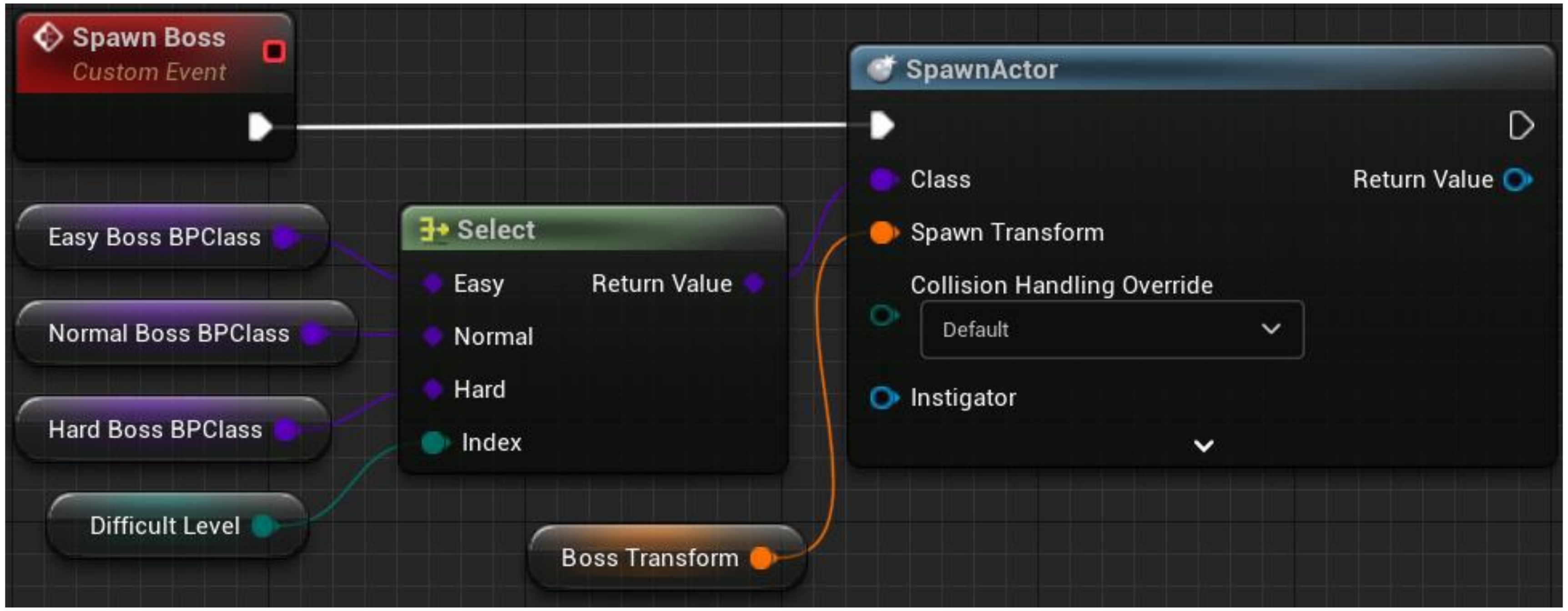
< Creating a local variable >

Using miscellaneous Blueprint nodes (1)

› **Select**
  - The node returns a value associated with the option that corresponds to the index that is passed as input.
  - **Option0** and **Option1** can be of any type, but the **Index** type must be **Integer**, **Enum**, **Boolean**, or **Byte**.



< The Select Node >

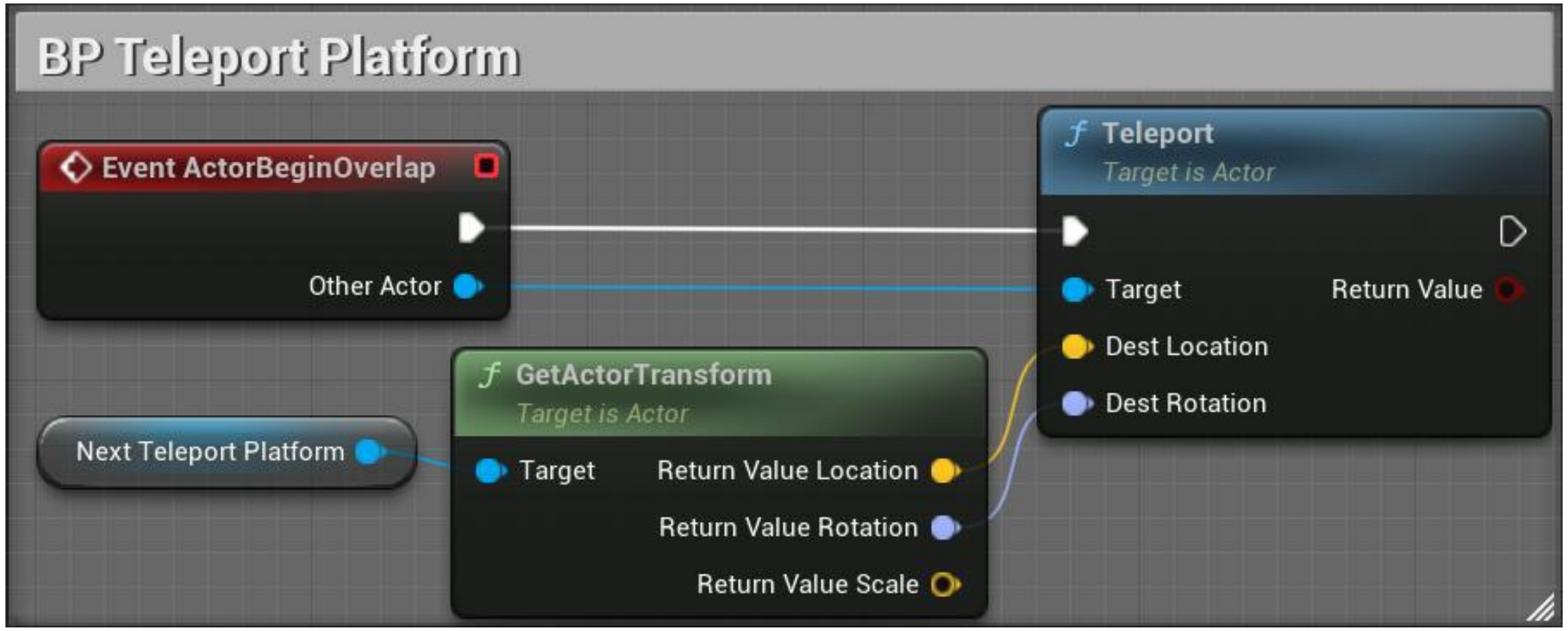


< Example of a Select node >

There is an enumeration named **Difficult Level** that has the values of **Easy**, **Normal**, and **Hard**.

Using miscellaneous Blueprint nodes (2)

- **Teleport**
  - The node moves an actor to the specified location.

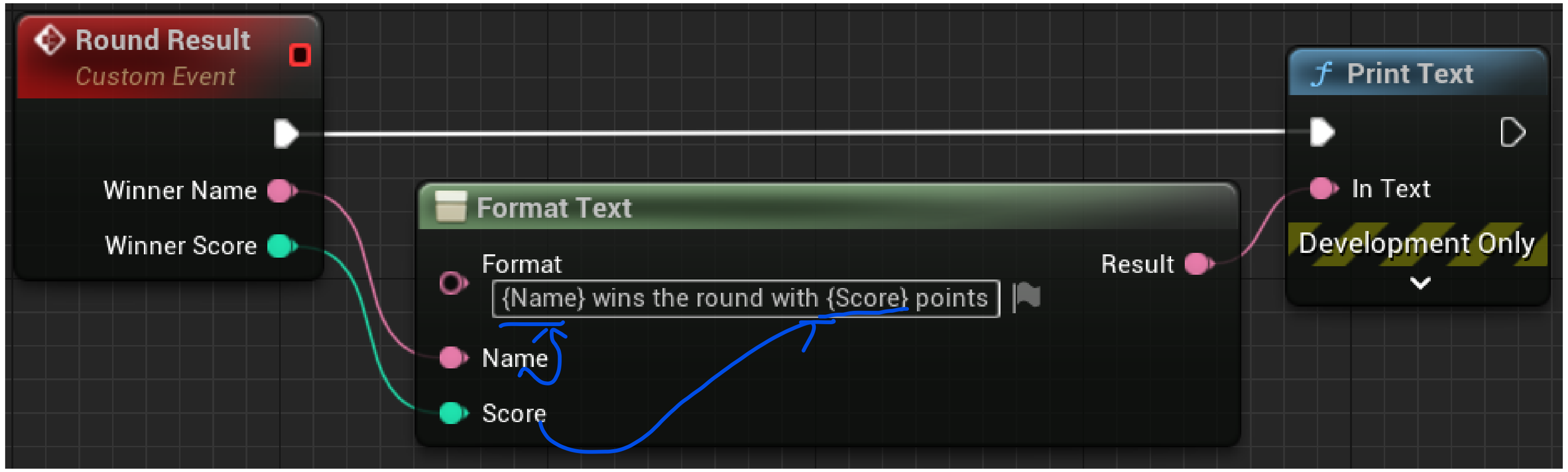


< Example of a Teleport node >

Using miscellaneous Blueprint nodes (3)

› **Format Text**

– The node builds text based on a template text and parameters specified the Format input parameter.

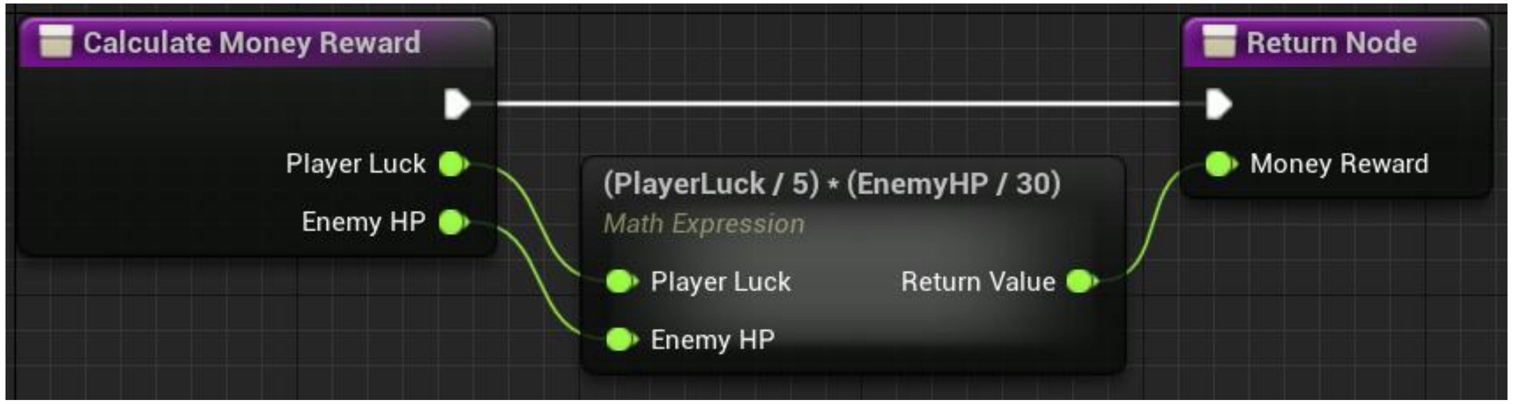


< Example of a Format Text node >

"Sarena wins the round with 17 points"

Using miscellaneous Blueprint nodes (4)

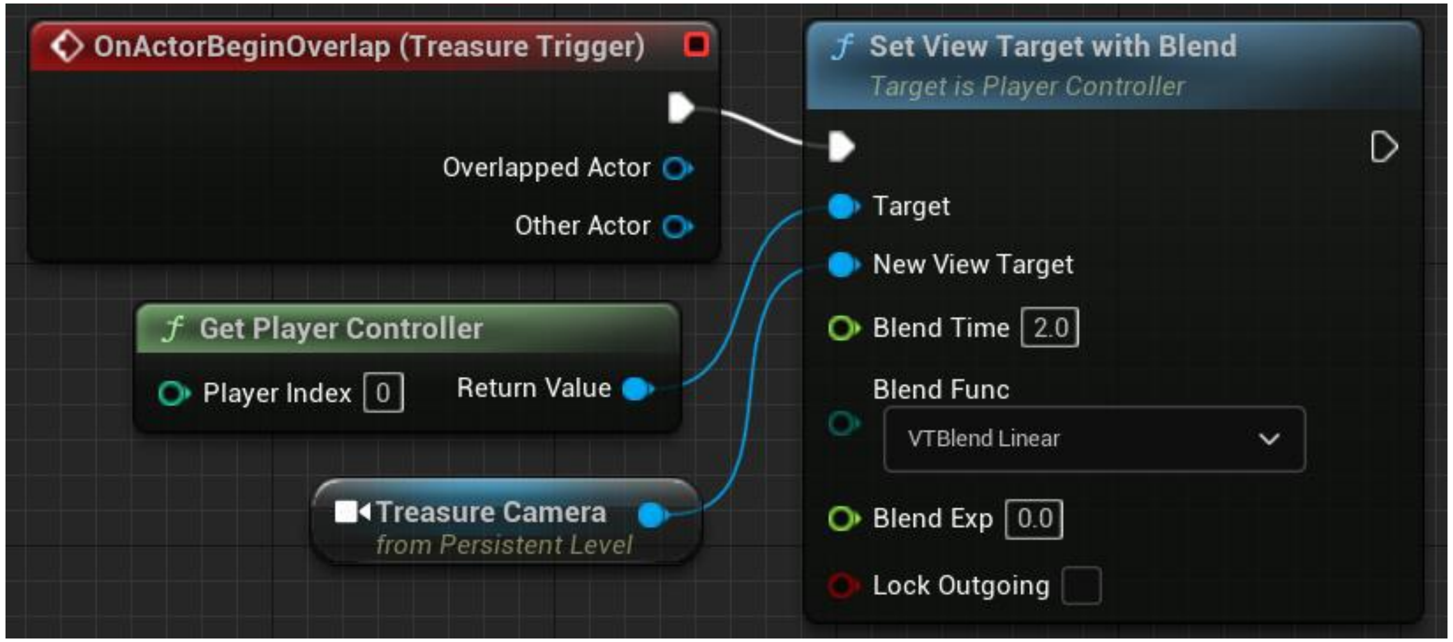
- **Math Expression** 자동으로 해석하여 input 달라질 자동으로 생성
  - The node is a collapsed graph created by the editor and is based on the expression typed in the name of the node.
  - An input parameter pin is created for each variable name found in the expression.



< Example of a Math Expression node >

Using miscellaneous Blueprint nodes (5)

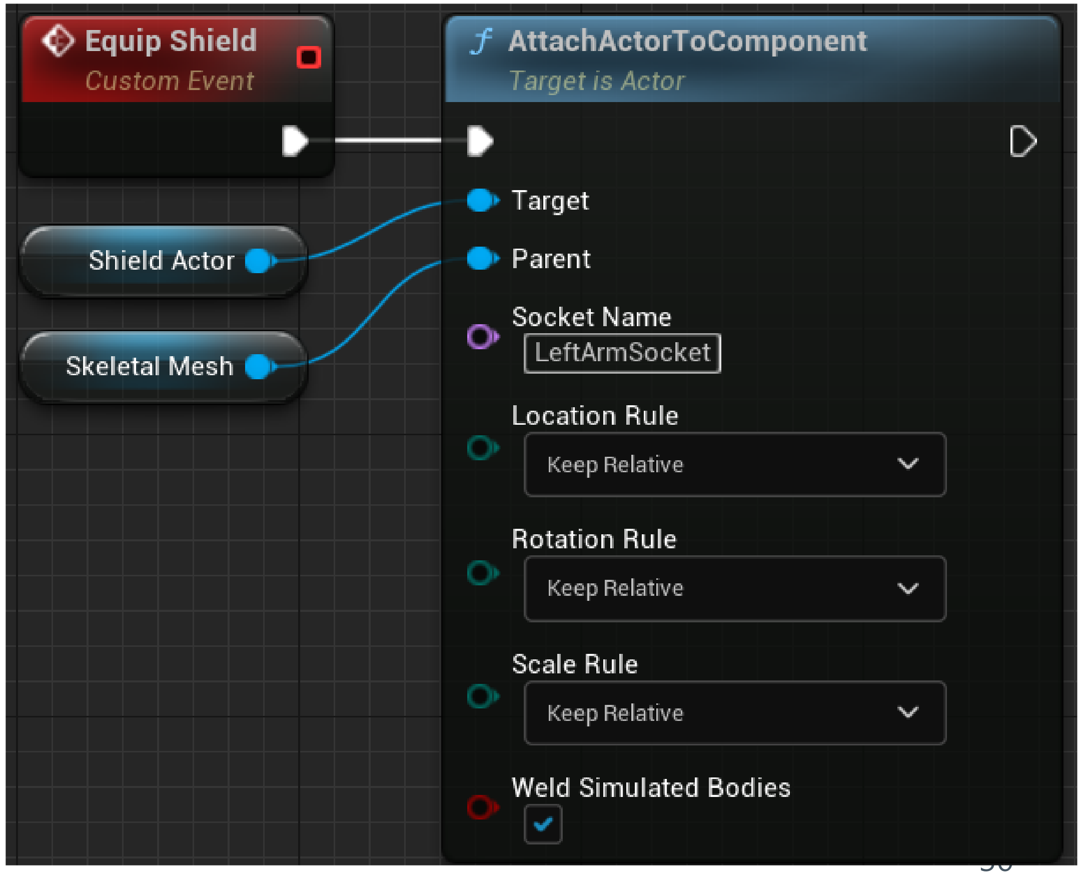
- **Set View Target with Blend**
  - The node is a function from the **Player Controller** class.
  - It is used to switching the game view between different cameras.



< Example of a Set View Target with Blend node >

Using miscellaneous Blueprint nodes (6)

› **AttachActorToComponent**
  - The node attaches an actor to the component referenced in the **Parent** input parameter.
  - Optionally, **Socket Name** can be used to identify the place where the actor will be attached.



Equip Shield
Custom Event

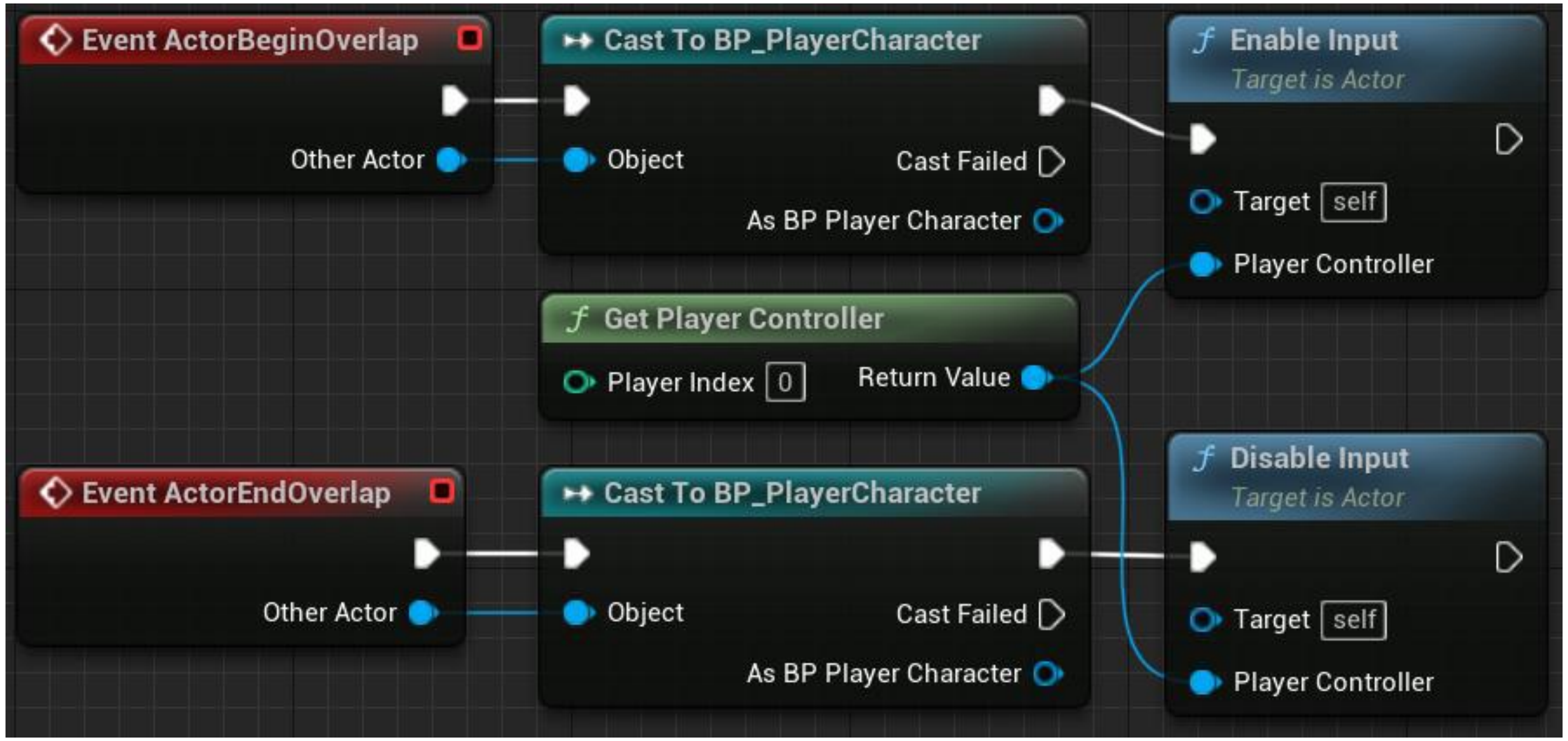
AttachActorToComponent
Target is Actor

Shield Actor → Target
Skeletal Mesh → Parent
Socket Name: LeftArmSocket
Location Rule: Keep Relative
Rotation Rule: Keep Relative
Scale Rule: Keep Relative
Weld Simulated Bodies: ✓

< Example of an AttachActorToComponent node >

Using miscellaneous Blueprint nodes (7)

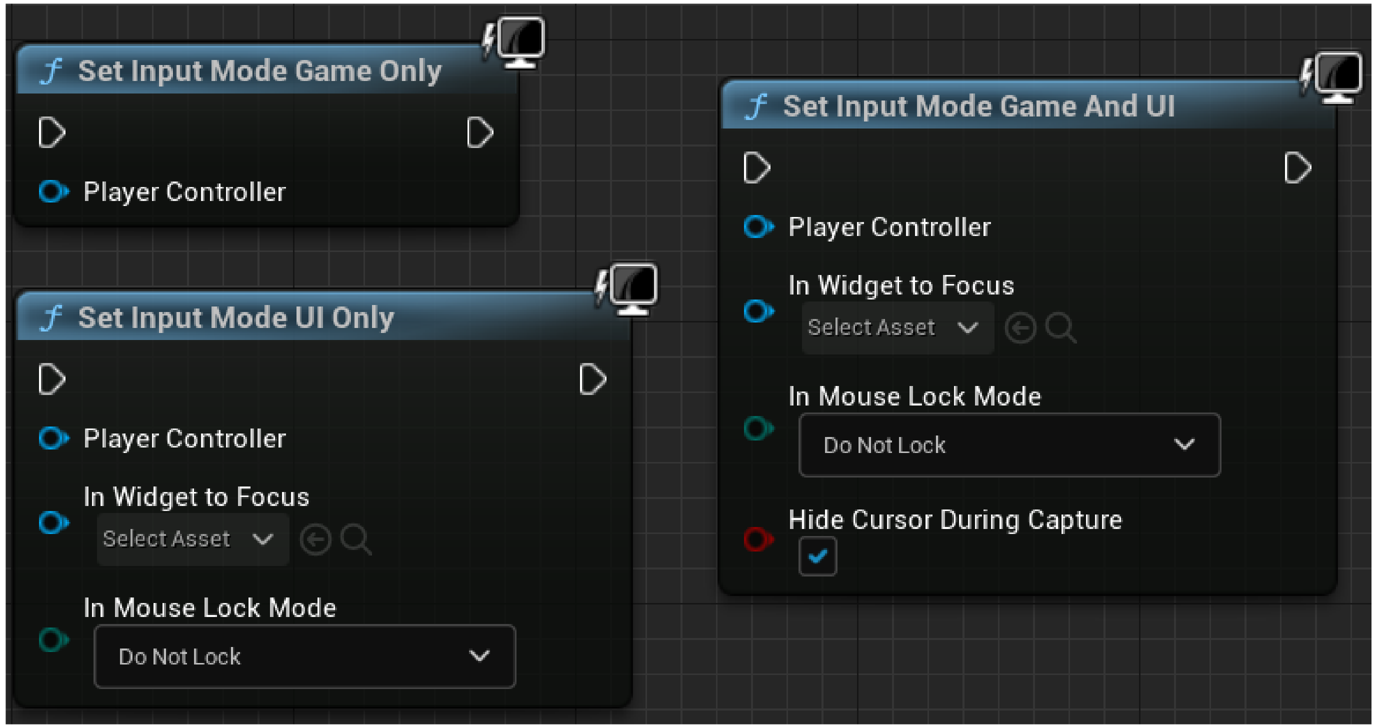
- **Enable Input** and **Disable Input**
  - The nodes are functions used to define whether an actor should respond to inputs events such as from a keyboard, mouse, or gamepad.
  - The nodes need a reference to the **Player Controller** class in use.



< Example of Enable Input and Disable Input nodes >

Using miscellaneous Blueprint nodes (8)

- The **Set Input Mode** nodes
  - There are three nodes that are used to define whether the priority in handling user input events is with the UI or with the player input.
    - **Set Input Mode Game Only**: Only Player Controller receives input events.
    - **Set Input Mode UI Only**: Only the UI receives input events.
    - **Set Input Mode Game and UI**: The UI has priority in handling an input event, but if the UI does not handle it, then **Player Controller** receives the input event.


Set Input Mode Game Only

Player Controller

Set Input Mode UI Only

Player Controller

In Widget to Focus: Select Asset

In Mouse Lock Mode: Do Not Lock

Set Input Mode Game And UI

Player Controller

In Widget to Focus: Select Asset

In Mouse Lock Mode: Do Not Lock

Hide Cursor During Capture: ✓

< The Set Input Mode nodes >