Variable Access Library

by Pixeltale Motion

Description

Variable Access Library will improve your blueprinting workflow by allowing you to create a widely modular coding structure, reducing your code to a minimum, making it look cleaner and more organized.
All hard reference calls via "Cast To" can be replaced with this library using soft references which result in a way more performant code and lower RAM usage.

With an evergrowing library of functions, you can GET/SET variables of the supported types of any object simply by name (soft reference), but also Get All Variable Names of a Target, Execute Functions by name, and much more.

To help to debug, the plugin allows you to print error messages to the log and/or screen (can be toggled in the project settings), with all needed information so you can find missing references faster.

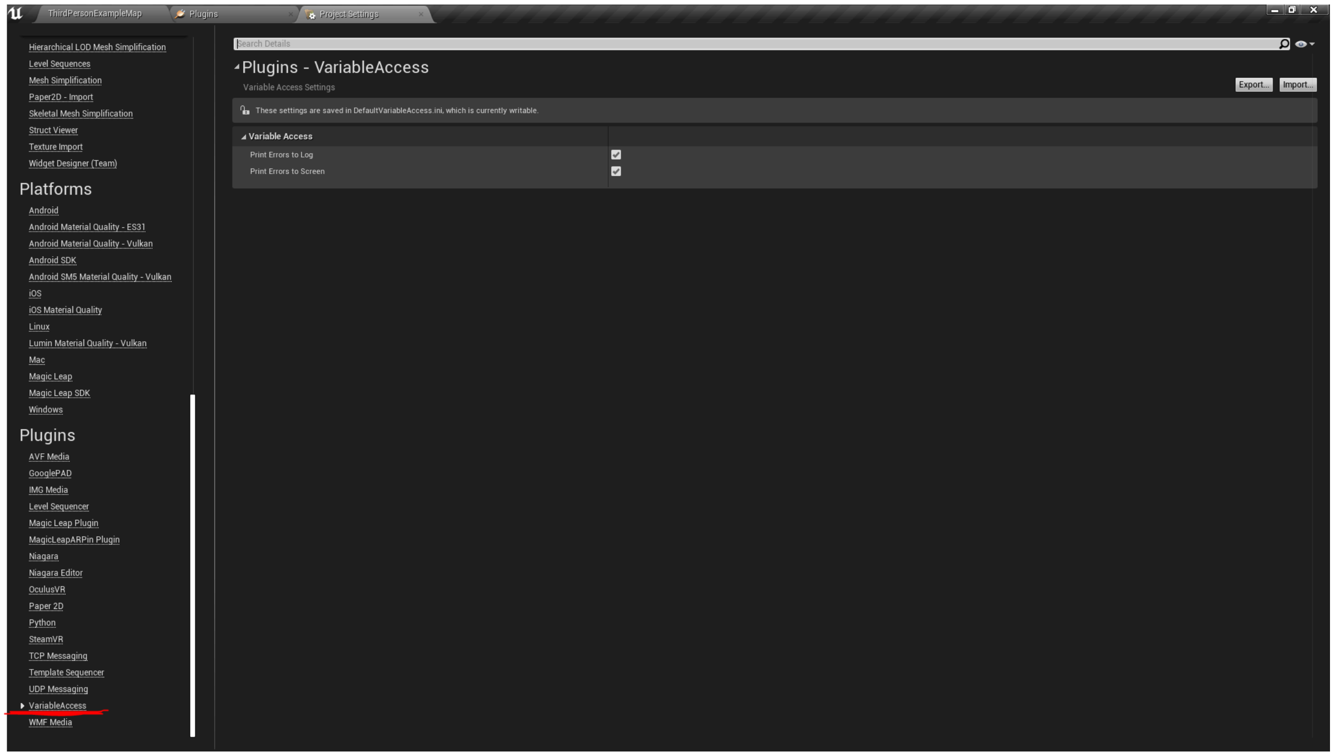
All functions are widely accessible (in C++ and blueprint)and support also replicated variables and RPC events.

Why is this better than a cast?

These functions all utilize soft references!
Because of these soft references, actor classes can be easily deleted without needing to replace the cast nodes or to worry that these cast nodes break, supporting a modular blueprint structure for complex modular code. This also is very useful to merge multiple project files together and won't require you to replace all cast nodes (This plugin is comparable with Unreals Interface System)
The most important reason is, that with less hard references load times will be way shorter and your RAM will not fill with references.
Very useful cases are overlap or line trace detection with multiple different actor classes. Instead of casting, one node of this library can set Variables of any actor, and as many actors as needed, anywhere. **The only requirement is the existence of that**

variable of the required type and name in the referenced target. Correctly setting up and using this plugin will save you a ton of time and keep problems far away.

Plugin Settings (Supported in V1.3) *If requested I can also add successful messages!*



Plugins - VariableAccess

Variable Access Settings

These settings are saved in DefaultVariableAccess.ini, which is currently writable.

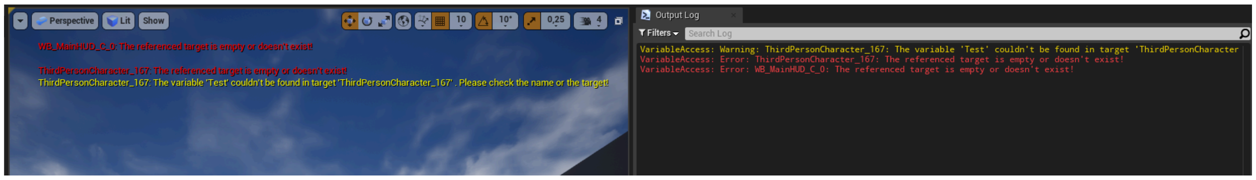
Variable Access

Print Errors to Log ✓

Print Errors to Screen ✓

To ensure easier debugging for all functions, I added 2 global booleans via Project Settings.

If set to true each function will print a warning to the log and on the screen if the searched for variable couldn't be found

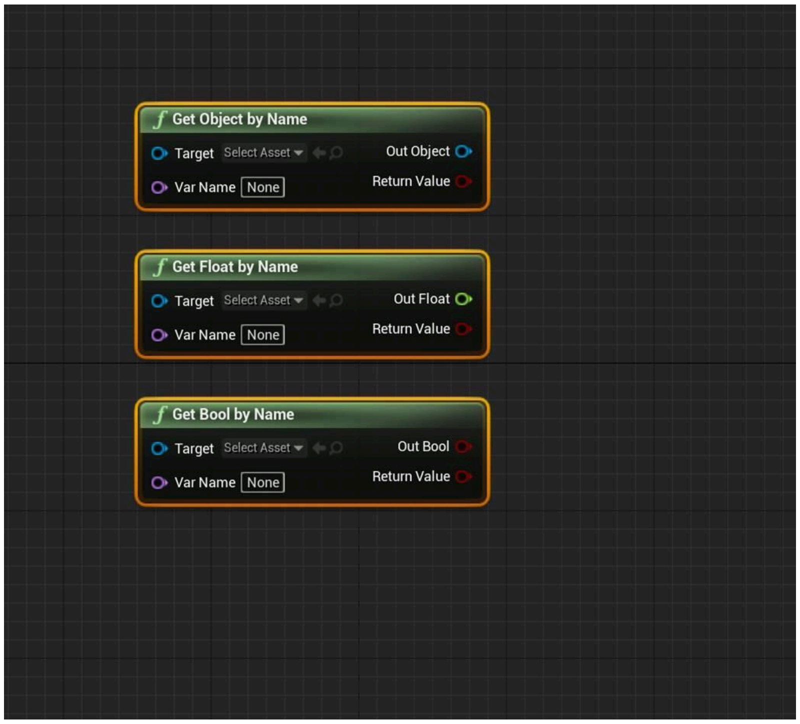


```
WB_MainHUD_C_0: The referenced target is empty or doesn't exist!

ThirdPersonCharacter_167: The referenced target is empty or doesn't exist!
ThirdPersonCharacter_167: The variable 'Test' couldn't be found in target 'ThirdPersonCharacter_167' . Please check the name or the target!
```

```
VariableAccess: Warning: ThirdPersonCharacter_167: The variable 'Test' couldn't be found in target 'ThirdPersonCharacter
VariableAccess: Error: ThirdPersonCharacter_167: The referenced target is empty or doesn't exist!
VariableAccess: Error: WB_MainHUD_C_0: The referenced target is empty or doesn't exist!
```

How to connect



Target -> Connect the object (like actor or player character) which contains the required variable

Var Name -> here you add the variable name exactly as named in the Target.

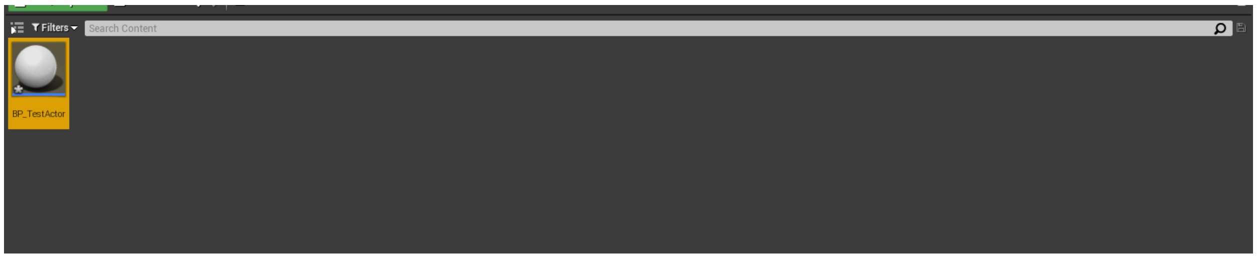
Return Value -> true if variable exists in Target

Out Value -> returns the variable value

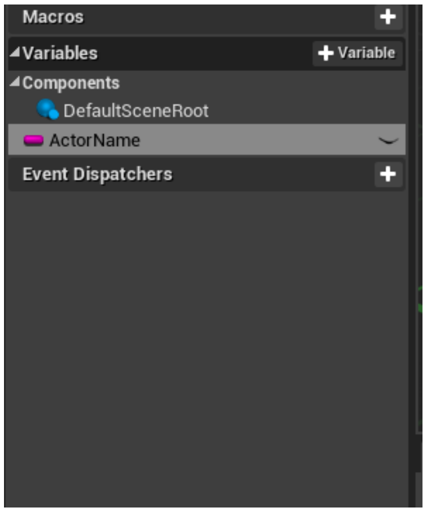
How to set up - General Example

This is a general explanation on how to utilize the functions correctly. If you need a more specific example, check out the example project!

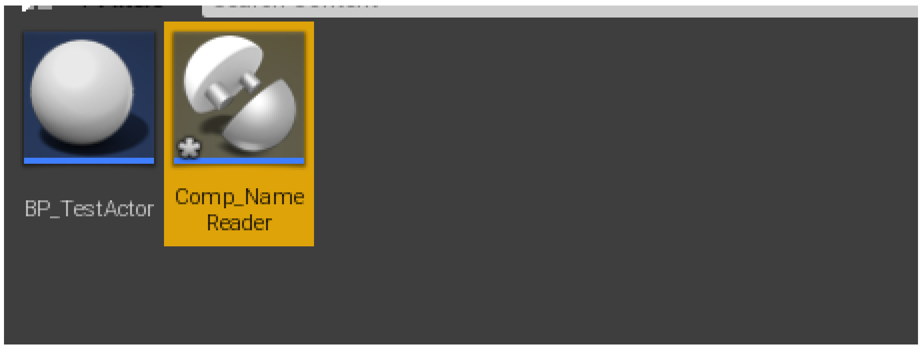
1. Create any object type



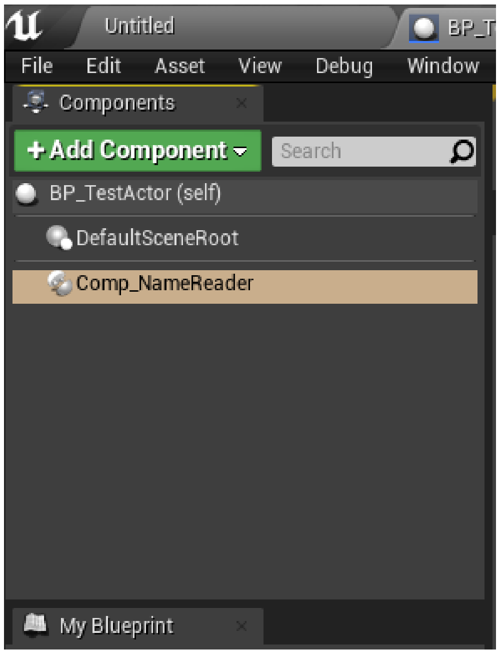
2. Create Variables of supported type



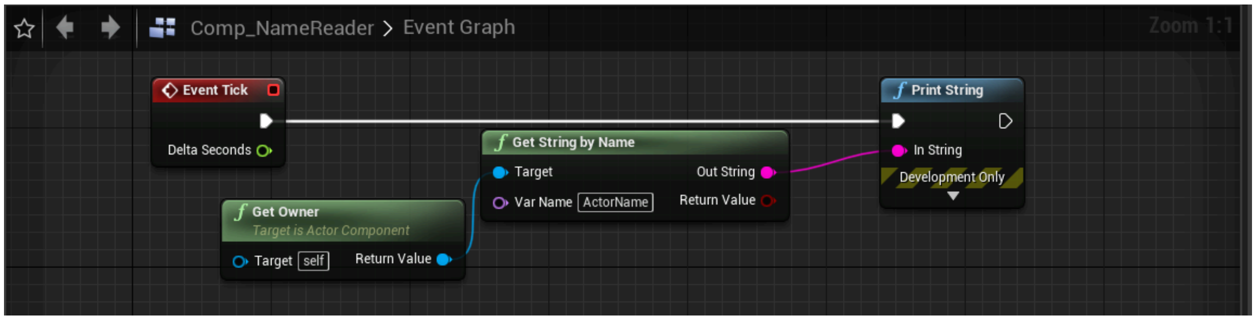
3. Setup other objects which may need to refer from the first object (an actor component in this example)



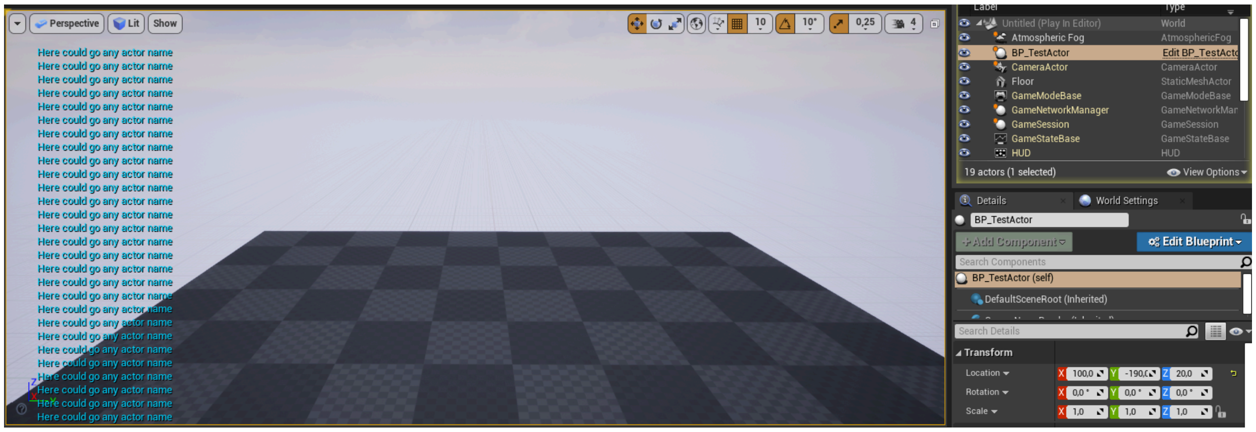
4. Build up a reference to the actor holding the variable (in this example we handle that through ownership)



5. Build Code to receive setted variable by name from referred object (the owner in this case)



6. Place in world and test

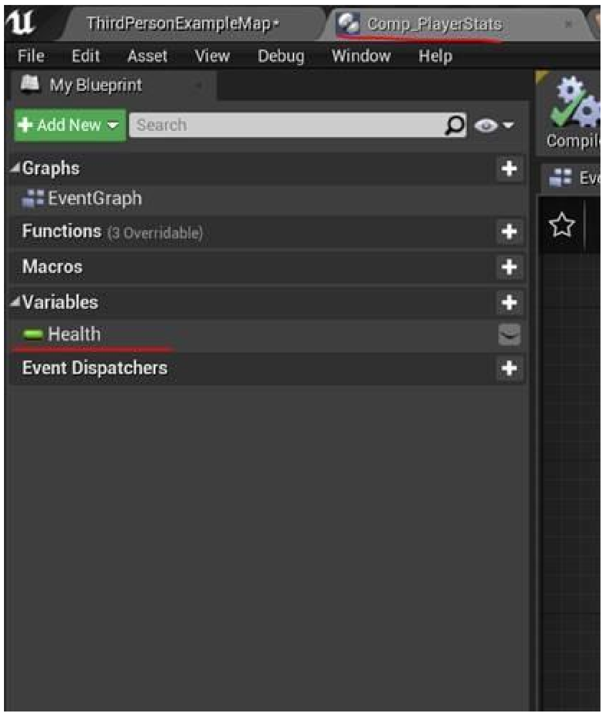


```
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
Here could go any actor name
```

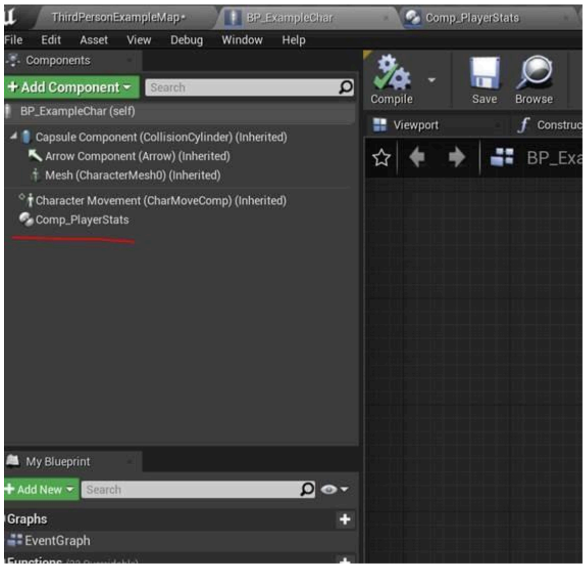
How to set up - Component Example

This is an explanation on how to utilize the functions when requesting a variable of a component.

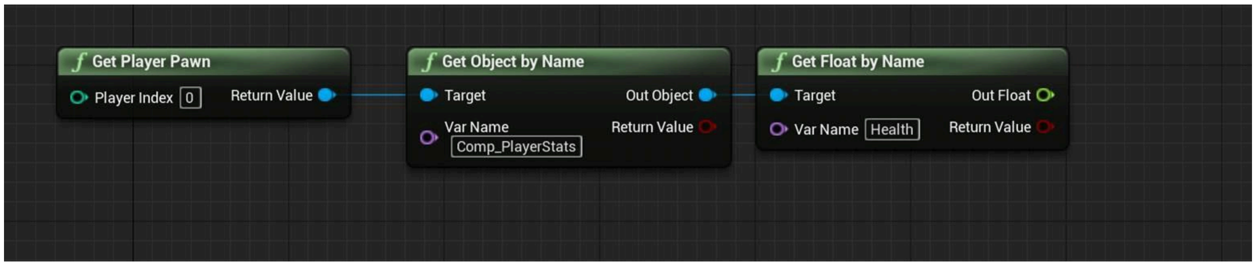
1. Create an actor component and add a float with the Name “Health”



2. Add Component to Player Character



3. Request Health Variable



“Get Object by Name to receive our component.
Var Name should be set with the exact name as in your player character.

“Get Float by Name” we get the float “Health”