Documentation

Customizable Menu System for Unreal Engine 5

Made by Moonville

Menu System Pro 2 - Documentation

List of contents

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Introduction

Foreword

Thank you for choosing the Menu System Pro. This user manual will help you to get the most out of this asset. If you have any questions, feel free to contact us at marketplace@moonville.dev or join our Discord server. Our developers will respond to your request asap.

Overview

Features

- Easy to use and easy to customize
- Clean separation of code and design for better updatability
- Designs can be changed without any code changes, with config data assets
- Advanced Design changes are easy to make in UMG widgets
- Modular Settings Manager for audio, display, gameplay, graphics, controls
- Menus & Widgets for all important settings a game needs (50+)
- Modular Savegame Manager with Save & Load Menu
- Enhanced Input System key rebinding with Keyboard & Gamepad support
- Ingame Pause Menu easy to combine with other game features e.g. an Inventory Menu
- Online Multiplayer menus (Host/Join Game and Server Browser)
- Local Multiplayer (Press button to join menu, Character Selection)
- Animated Game Intro with changeable logos
- Animated Credits Roll with hold to skip functionality
- Customizable Loading Screens for each level
- Accessibility Settings (e.g. Color Deficiency)
- Menu Music and SFX included (easy to change)
- Polished Example Scene with Sample Character
- Example menus for Character Selection, Level Selection, Grid Navigation, Smooth Menu Camera Switch and Input Guide
- Multi Language Support (English and German included)
- Uses Graphics Benchmark to determine optimal performance
- All settings are stored as JSON and can be modified in a packaged build
- Easy to save and load your custom settings
- Extensive documentation & great support
- Blueprint Only!

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Included Menus

- Intro Menu
- Title Screen Menu
- Gameplay General Settings Menu
- Gameplay Multiplayer Settings Menu
- Audio General Settings Menu
- Audio Voice Chat Settings Menu
- Video Display Settings Menu
- Video Graphics Settings Menu
- Controls General Settings Menu
- Controls Key Bindings Settings Menu
- Singleplayer Menu
- Savegame Menu
- Ingame Menu
- Loading Screen
- Multiplayer Menu
- Host Online Game Menu
- Server Browser Menu
- Local Multiplayer Menu
- Design Showcase
- Character Selection Menu
- Grid Navigation Menu
- Input Guide Menu
- Level Selection Menu
- Menu Camera Switch Menu

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

- Credits

Concept

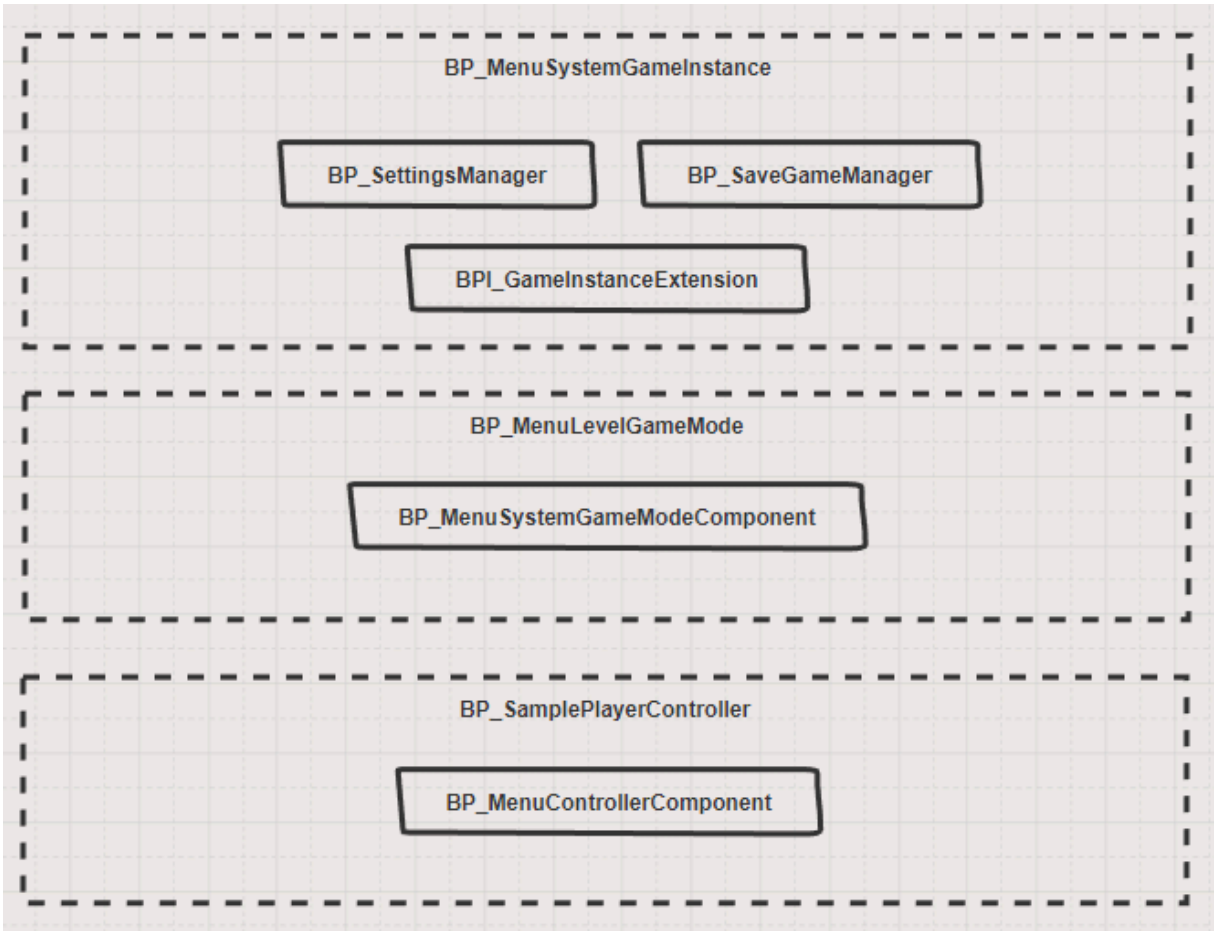
The images below show the core structure and classes of Menu System Pro.

The *Menu System Actor* instantiates the *Menu Container Widget* which serves as a container for all the UI widgets (menus and widgets). Those widgets will be added dynamically on-demand to specific layers of the container (Background, Main, SubTabs, FooterBar, Dialog, LoadingScreen and Foreground).

The UI is based on *Global Interactable User Widget*, *Interactable User Widget* and *Interactable Menu Widget*. All actual User Widgets and Menus inherit from these three classes. The *Interactable Menu Widget* acts as a container for *Interactable User Widgets* (OptionsPicker, Slider, Buttons etc.).

Furthermore there is a class called WBP_SoundManagerUI for audio management, which enables audio queueing and global handling of UI sounds.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)



The Menu System Game Instance includes a Settings Manager for UI independent management of settings and serialization using the JSON Blueprint Utilities. It also contains the Save Game Manager for Saving & Loading the game. The Game Instance Extension offers a standardized signature for function calls from other classes and therefore allows easy integration into an existing Game Instance class.

The Menu Level Game Mode (also Play Level Game Mode) is extended using the Menu Game Mode Component for local multiplayer functionality.

The Sample Player Controller is extended with the Menu Controller Component which houses essential code for interacting with the Menu System and also things like setting up the Enhanced Input System.

Getting started

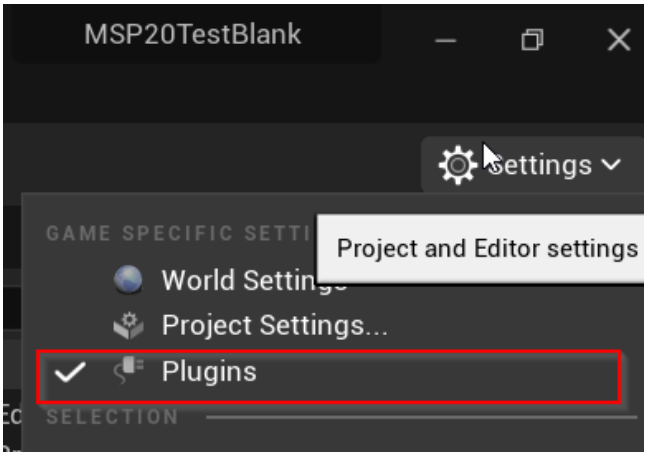
In this section, we will guide you through the process of setting up Menu System Pro v2. If you prefer video tutorials, you can access them by clicking the links below.

Menu System Pro 2.0 for Unreal Engine 5: Quick Start Guide & Customization Intro

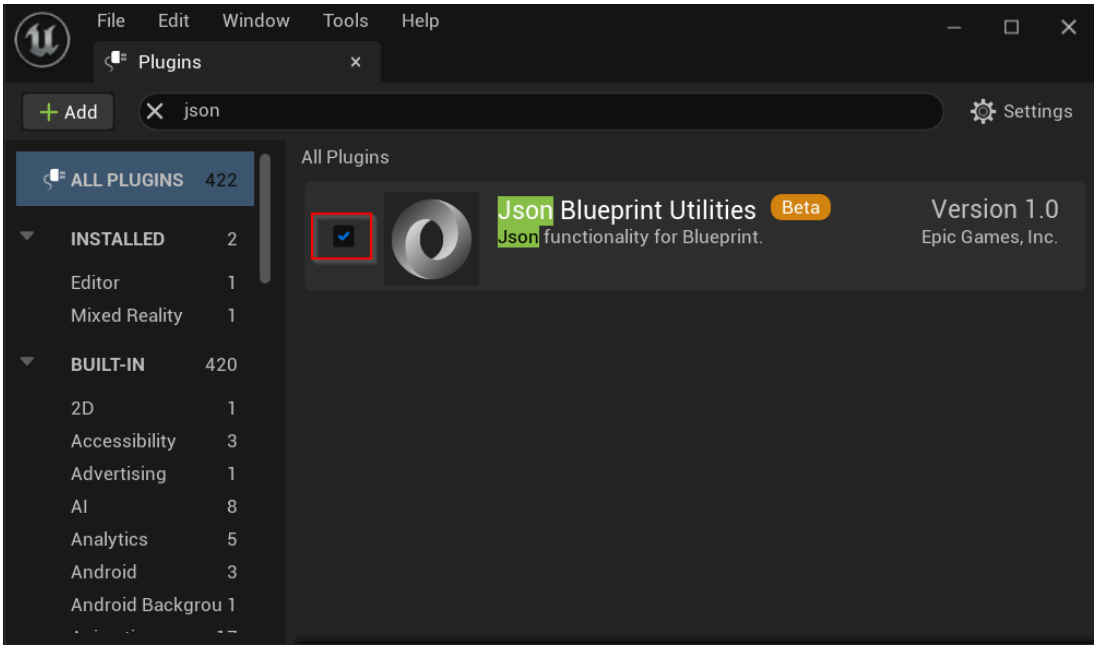
Menu System Pro 2.0 for Unreal Engine 5: Integration Guide for Game Framewo…

Installation

1. Open the Settings/Plugins Window in the top-right of the Editor

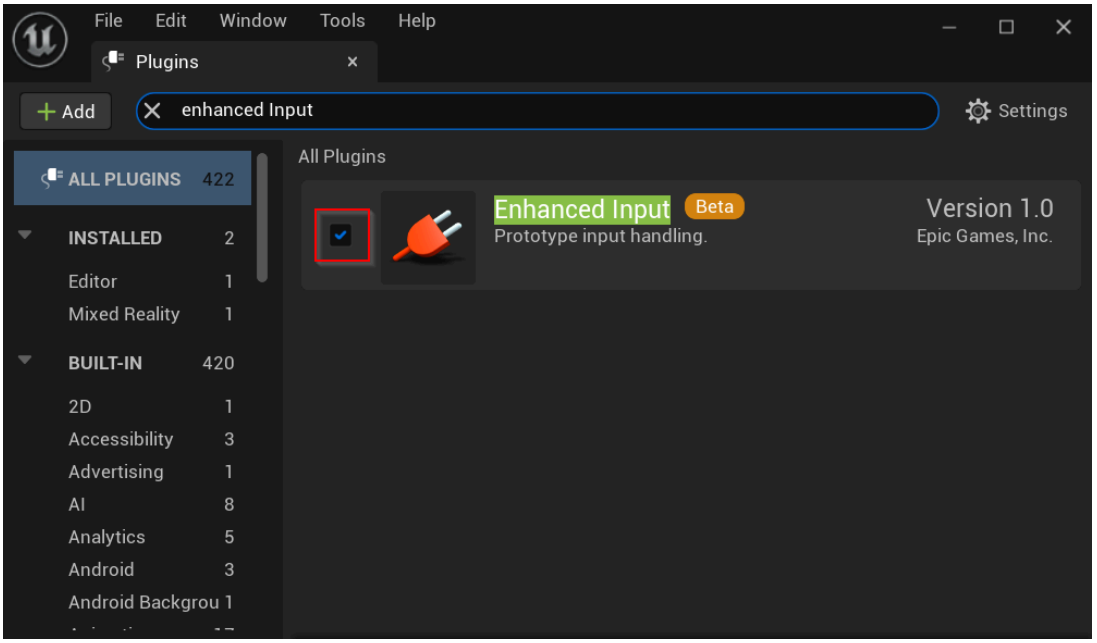


2. Enable the JSON Blueprint Utilities Plugin

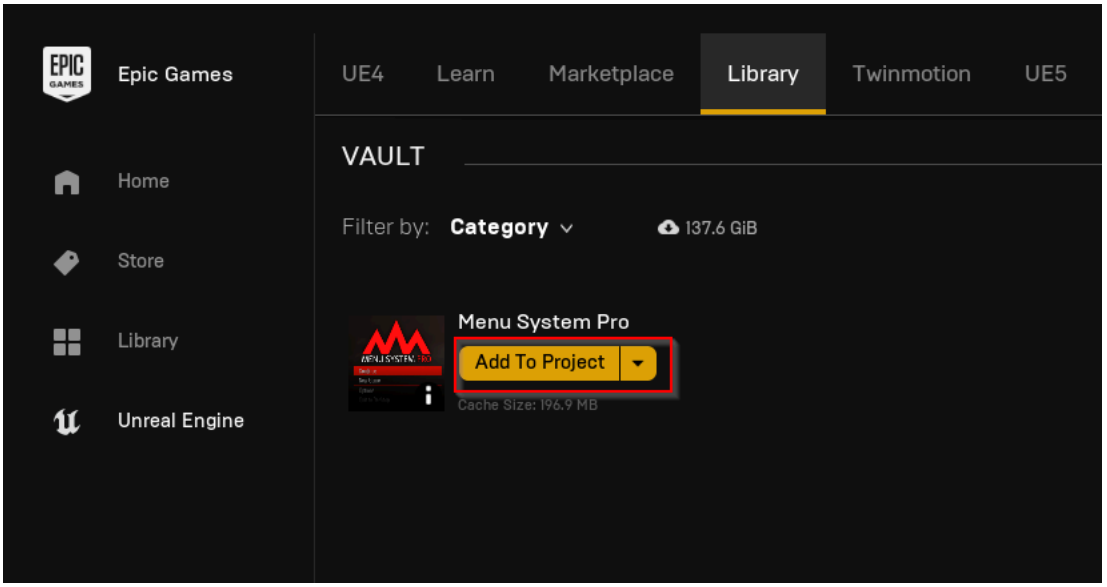


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

3. Enable the Enhanced Input Plugin (Newer UE versions include this by default)

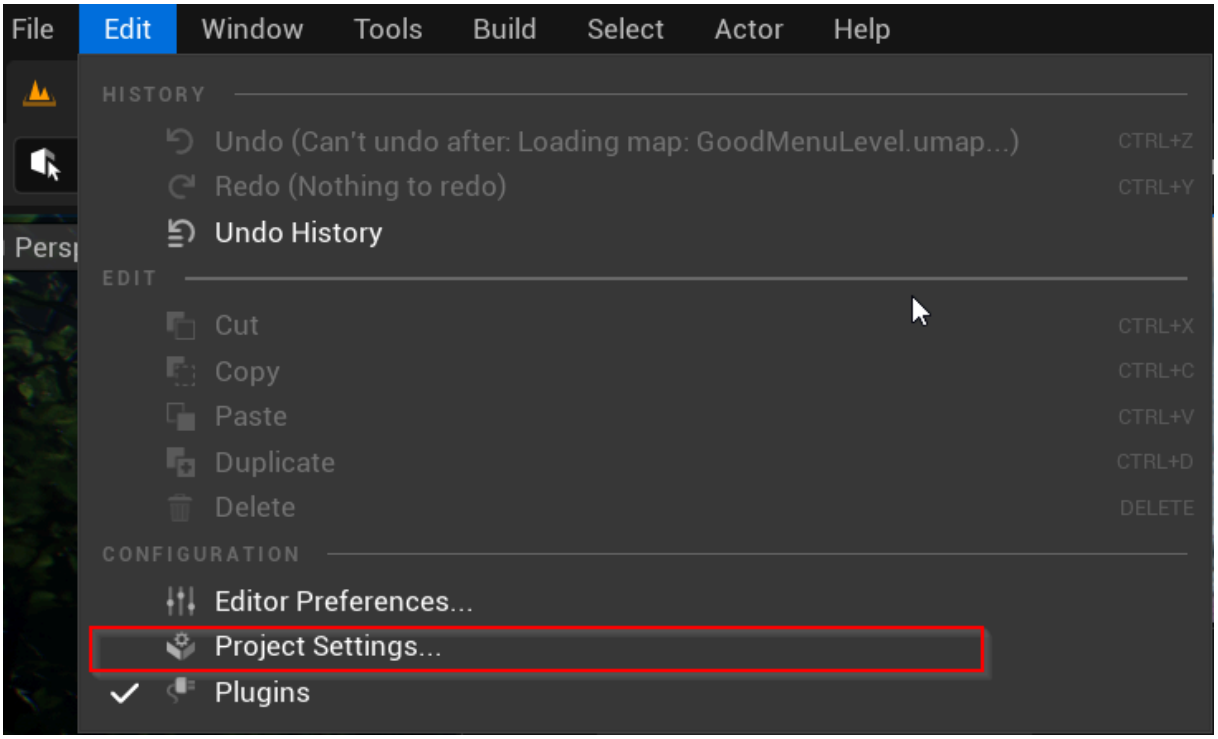


4. Add 'Menu System Pro' to your project using the Epic Games Launcher

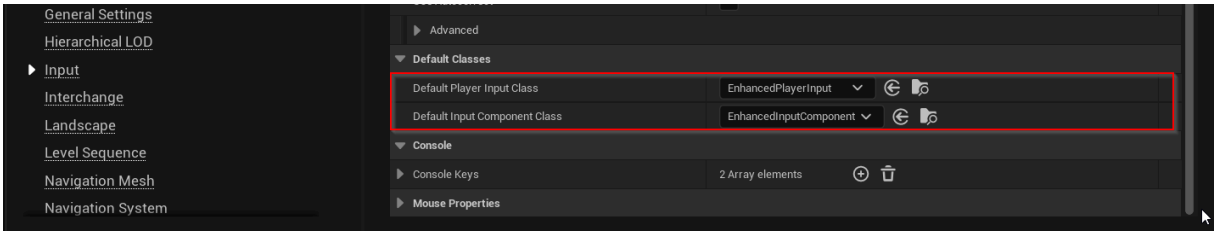


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

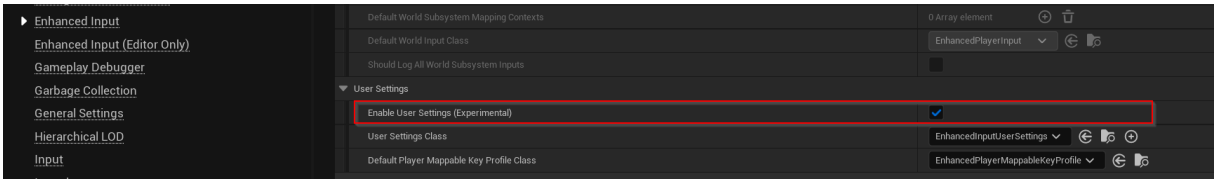
5. Open the Project Settings



6. Change Input/Default Classes to use 'EnhancedPlayerInput' and 'EnhancedInputComponent'

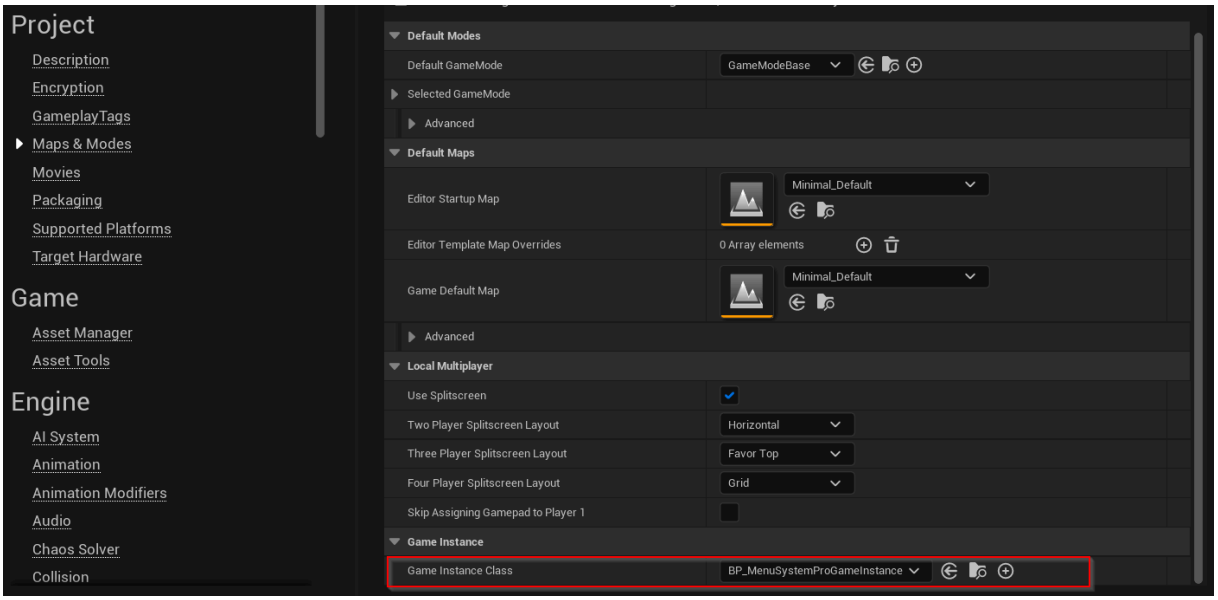


7. Enable User Settings in the Enhanced Input section (required as of UE 5.3)



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - [Click here for the new Common UI documentation (MSP 2.2.0+)]

8. Change Maps & Modes/Game Instance Class to use 'BP_MenuSystemGameInstance'. If you already have a Game Instance class in your project, jump to *'Setup your existing Game Instance'*



9. Open the menu level at '/ExampleContent/Designs/Design_%DesignName%/Levels/SilenceMenuLevel.umap'
10. Press play and enjoy

If your character can't move, you are using a source build, or have other issues with the setup we recommend that you refer to the **Common Issues & FAQ** section for further guidance. It also contains a fix for the "Missing String Table Entry" issue occurring in UE 5.4.

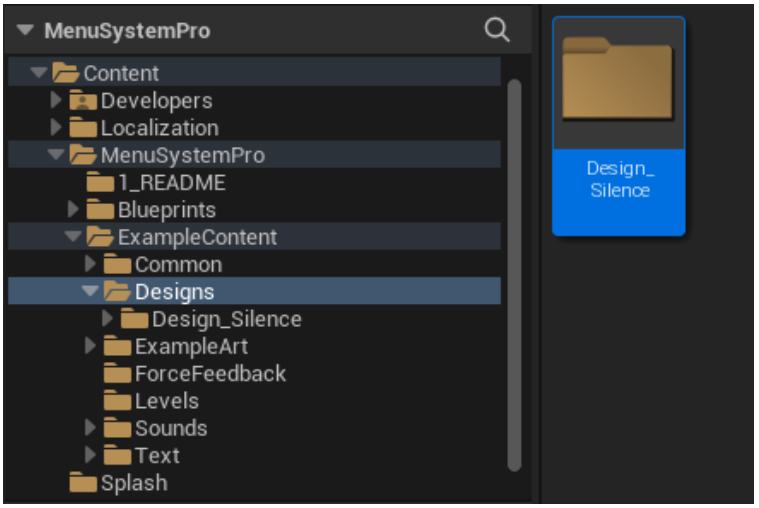
IF YOUR CHARACTER CAN NOT MOVE you very likely need to change IMC_CharacterOnFoot with the IMC your Character uses (e.g. IMC_Default) in BOTH LISTS (Registered Input Mapping Contexts & Initially Added Input Mapping Contexts) in DA_MenuSystemConfig_Silence (NOT THE PDA_ file!).

Use 'showdebug enhancedinput' in the console to debug why your character does not accept inputs!

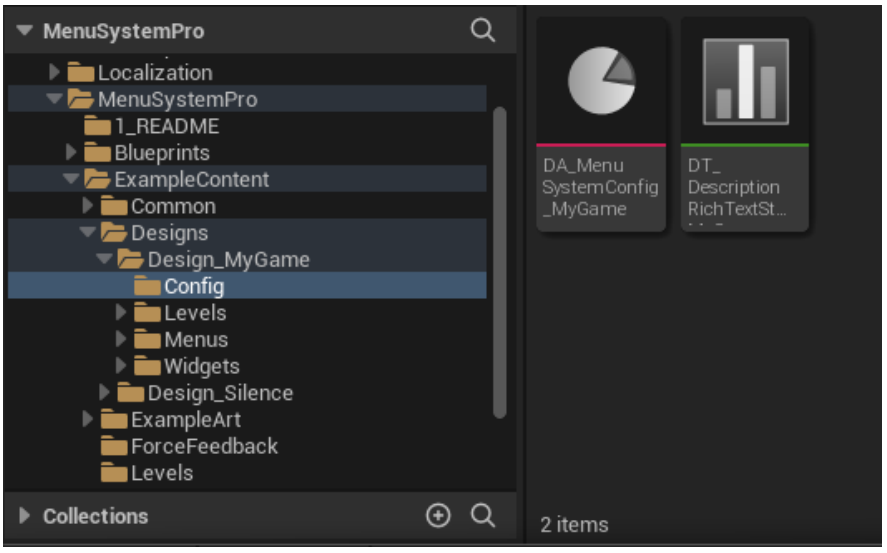
Make it your own design

This is an important step you should follow before making any design or config changes. It will save your changes from getting overwritten by future updates.

1.) Navigate to '*/ExampleContent/Designs*' and Rename from '*Design_Silence*' to 'Design_MyGame' (or something else unique).



2.) Then go ahead and rename every file in this folder that ends with _Silence to your own ending you chose before.



Tip: You can click on the Design folder and use the search with '_Silence' to find all occurrences

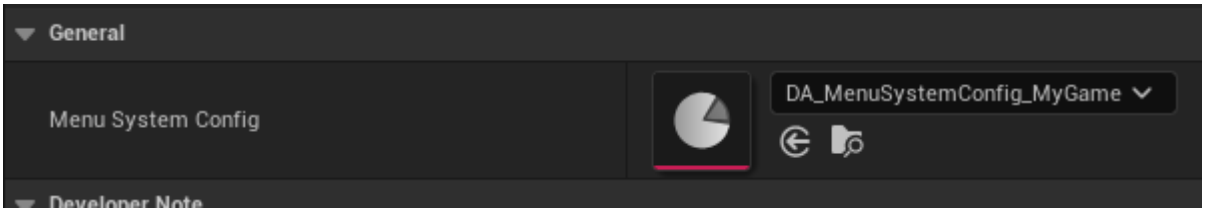
This makes sure your design is not overridden when we update the Silence

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Design.

3.) Navigate to '/Blueprints/Core' and open the DA_MenuSystemConfigRedirector

Check that your MenuSystemConfig Data Asset is selected



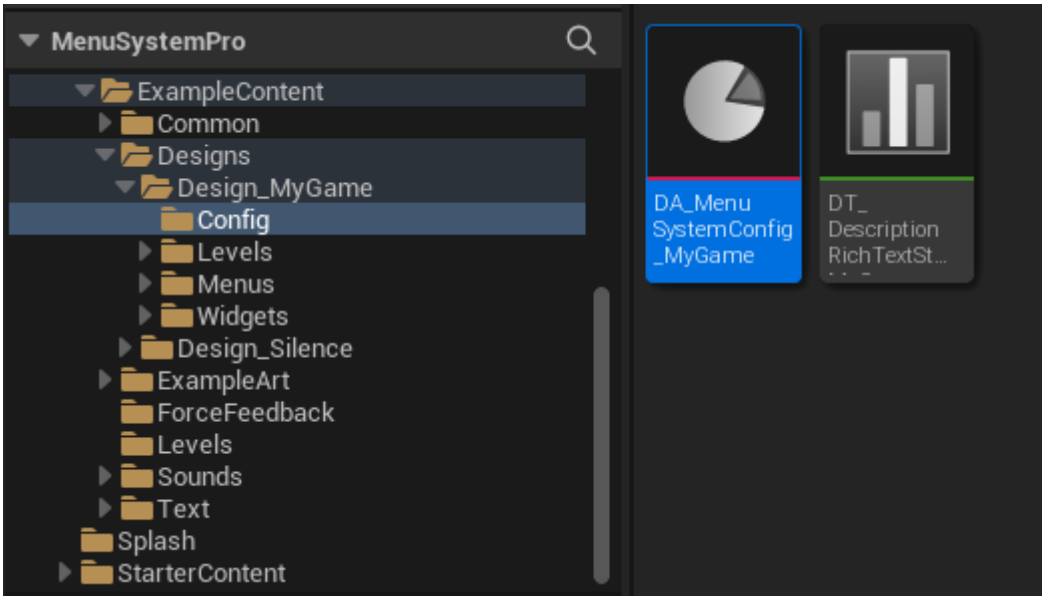
Note: This tells Menu System Pro what design config should be used

4.) Always create backups before updating (use source control like Perforce if possible).

Change Menu & Play level

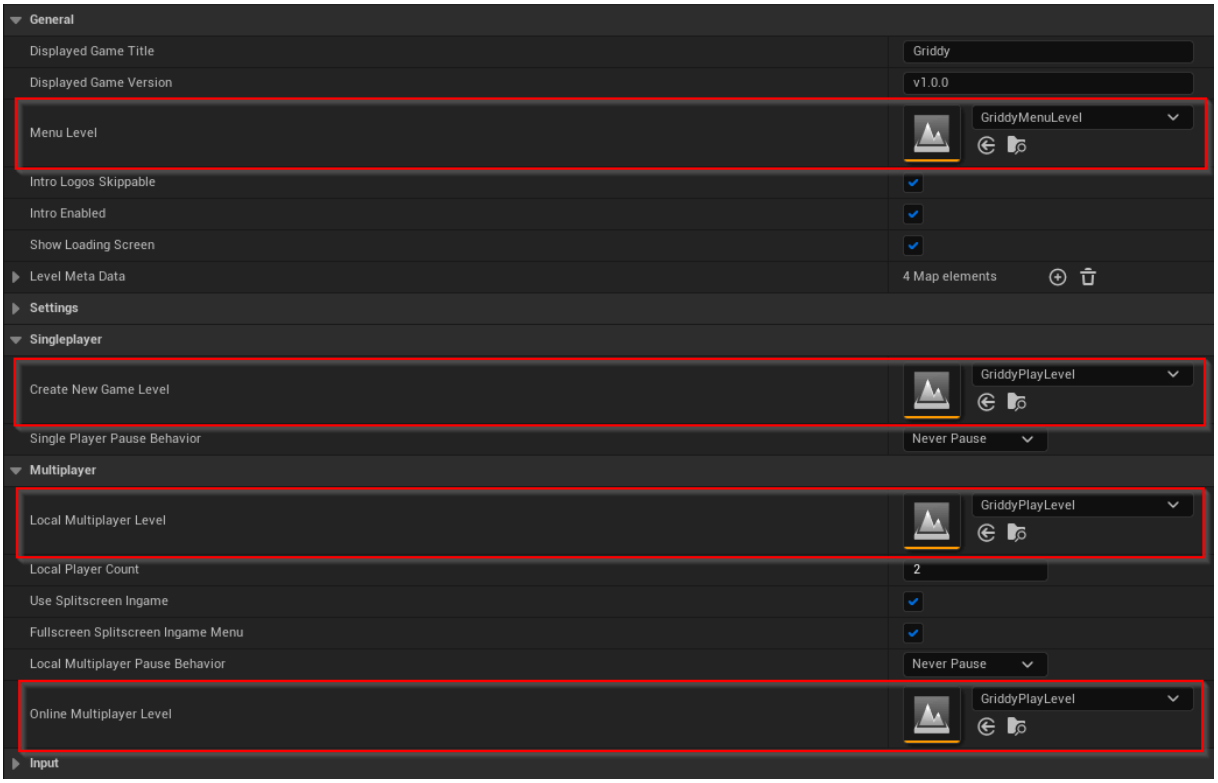
To ensure that the menu system loads the correct levels for new game, when returning to the main menu, and for multiplayer (local & online)

1.) Navigate to the Menu System Config located at '/Design_%DesignName%/Config/



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

2.) Change the levels according to your needs

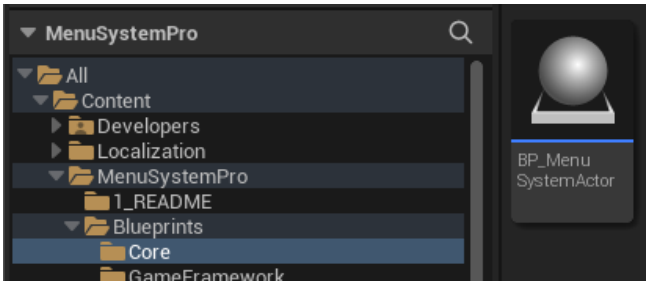


Note: If you don't have multiplayer, you don't have to change them.

Setup your menu level

To ensure that your startup map/level displays the menu immediately, you will need to follow these steps:

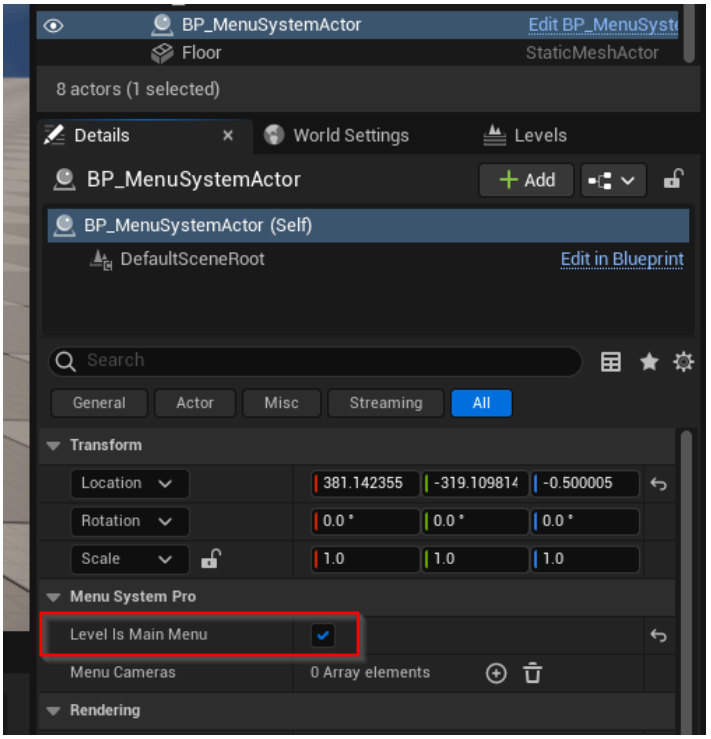
1.) Add *'BP_MenuSystemActor'* to your menu level by dragging & dropping it into your level



Location: /Blueprints/Core/

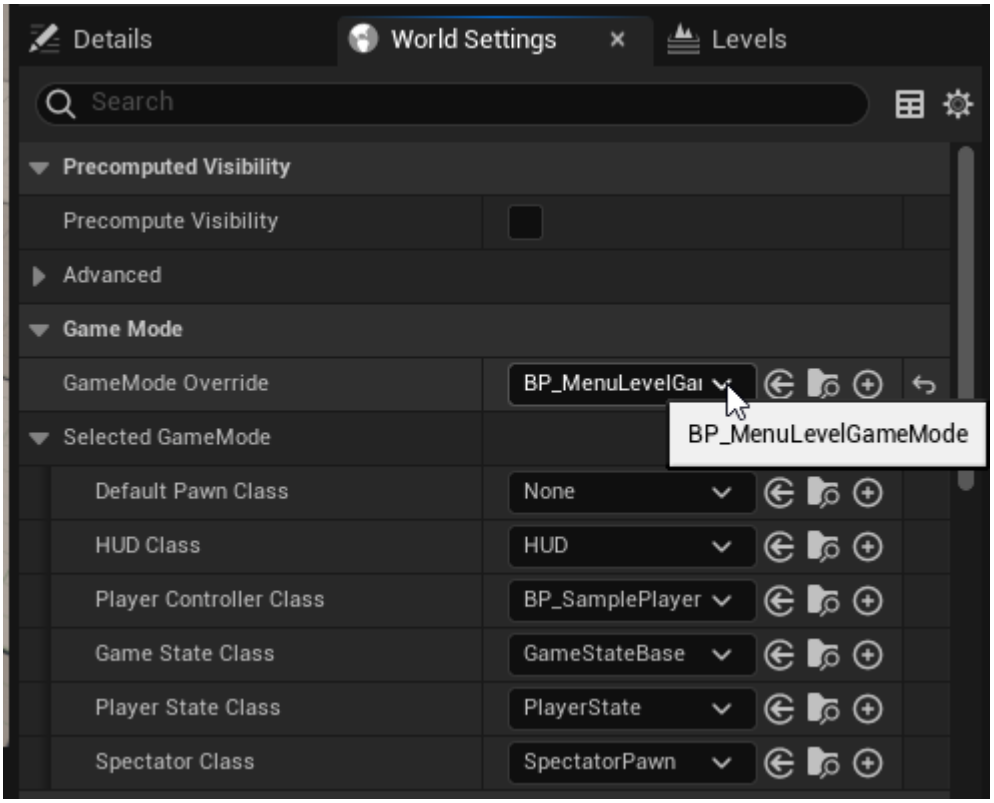
2.) Select the *'BP_MenuSystemActor'* in your level and enable *'Level Is Main Menu'*

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)



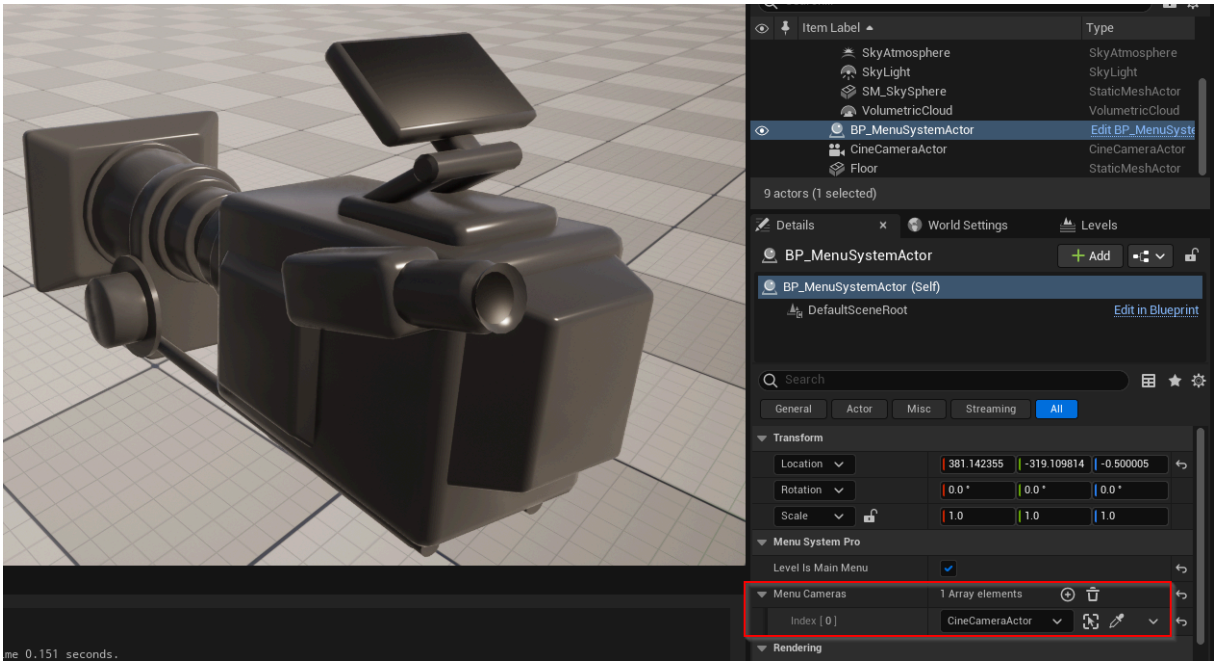
3.) Open World Settings in your level and change the Game Mode Override to '*BP_MenuLevelGameMode*'

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)



Alternative: If you already have your own game mode you can implement the *'BP_MenuSystemGameModeComponent'* on it. It's located at: *'/Blueprints/GameFramework/'*

4.) Optional Step: When a good view on the scene is needed you can place a Cine Camera Actor in the level and select it on the *'BP_MenuSystemActor'* like this



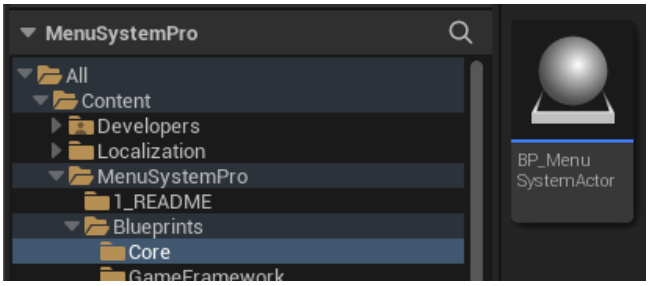
Menu System Pro will automatically set the view to the first Camera in this array. If you want to use multiple cameras have a look at the example in the Extras menu (Menu Camera Switch).

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Setup your play level

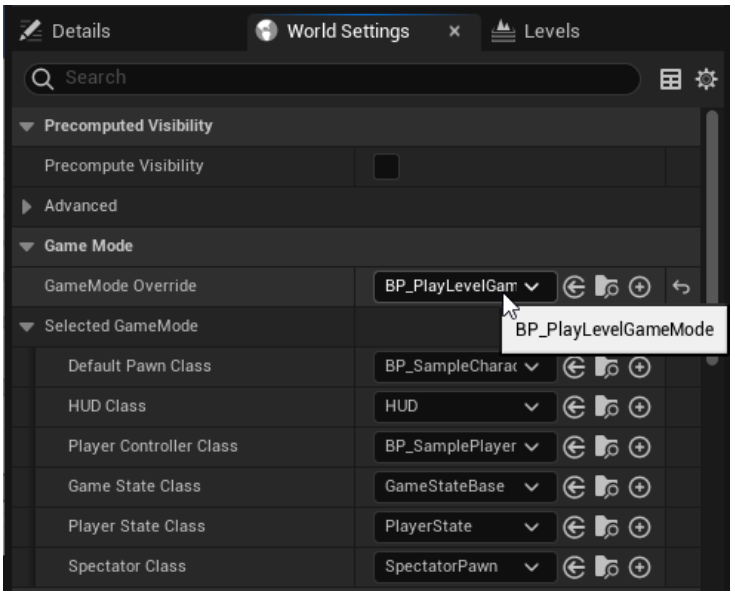
To prepare your play level for accessing the Ingame menu, please follow the steps outlined below

1.) Add 'BP_MenuSystemActor' to your play level



Location: /Blueprints/Core/

2.) Open World Settings in your level and change the Game Mode Override to '*BP_PlayLevelGameMode*'



Alternative: If you already have your own game mode you can implement the '*BP_MenuSystemGameModeComponent*' on it. It's located at: '*/Blueprints/GameFramework/*'

If you also have custom Character and Player Controller classes see the section below.

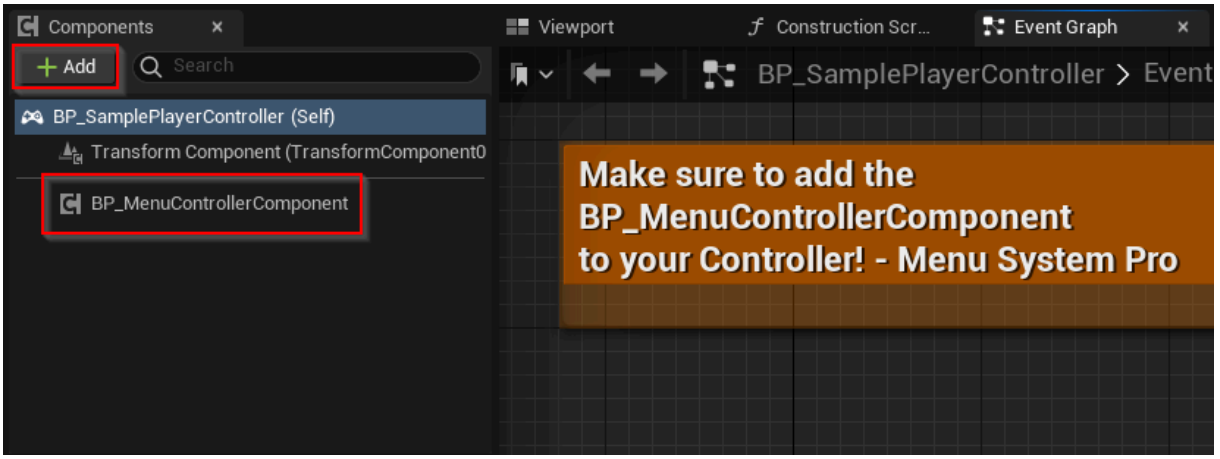
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

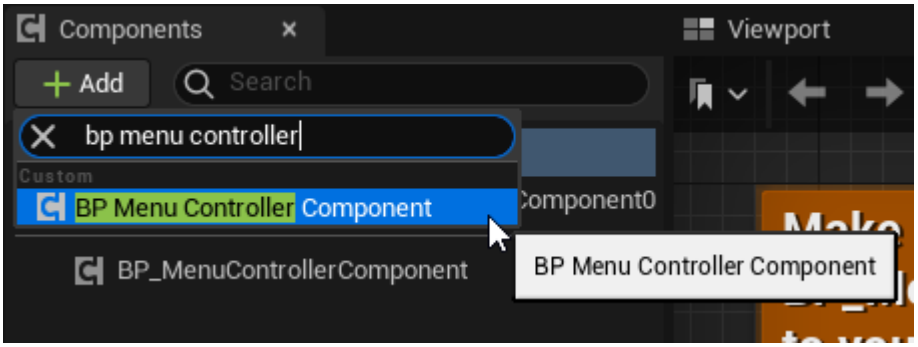
Setup your player classes

If you wish to use your own Player Controller and Character classes and make them compatible with Menu System Pro, please follow these steps

1.) Add '*BP_MenuControllerComponent*' to your Player Controller



You can locate it by using the search function on the *Add* button in the following manner:

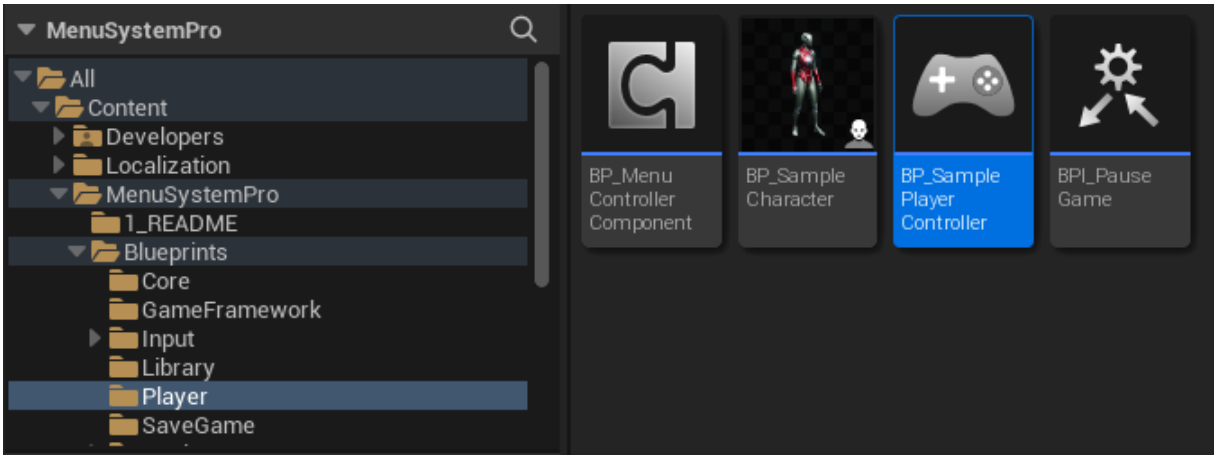


By doing so, your player controller will be ready to use Menu System Pro.

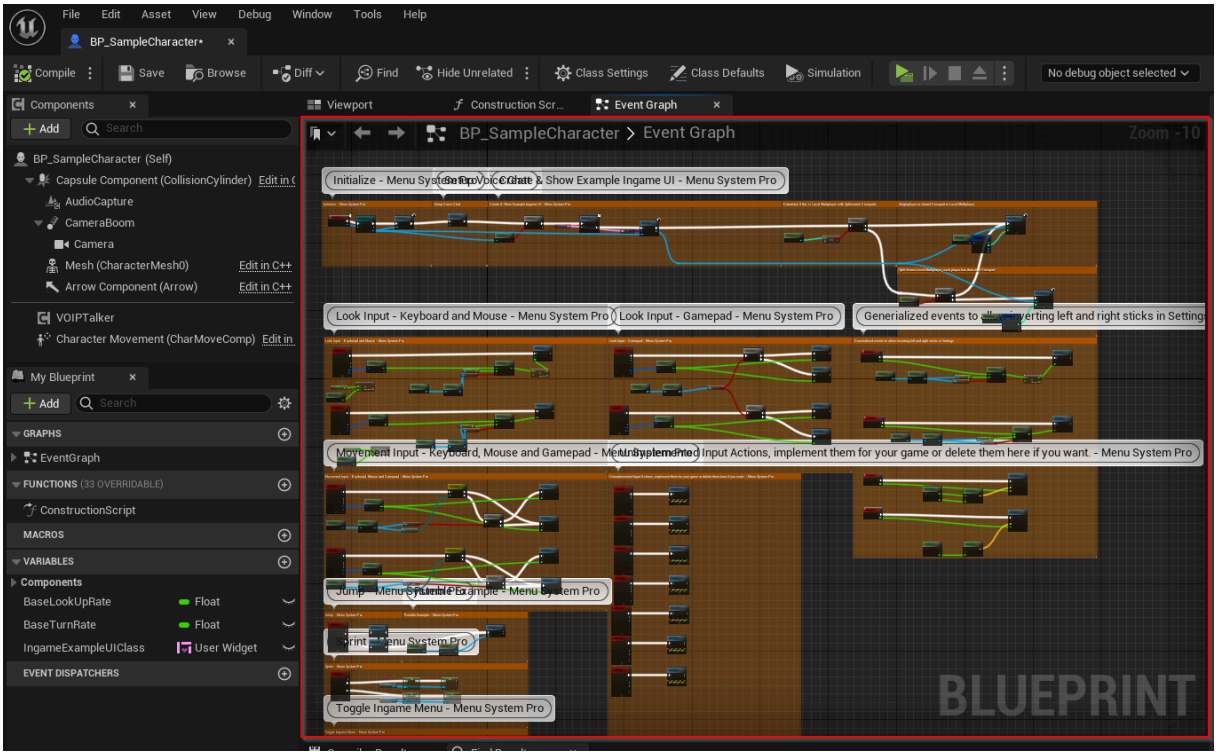
2.) Now, to set up the Character class, you will need to add some code for initialization, opening the menu, and accessing settings such as axis inversion and sensitivity. Navigate to '*/Blueprints/Player/*' and open the

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

'BP_SampleCharacter' Blueprint



3.) Copy and paste all the nodes from the event graph to your Character Class and adjust it to your needs, if necessary.



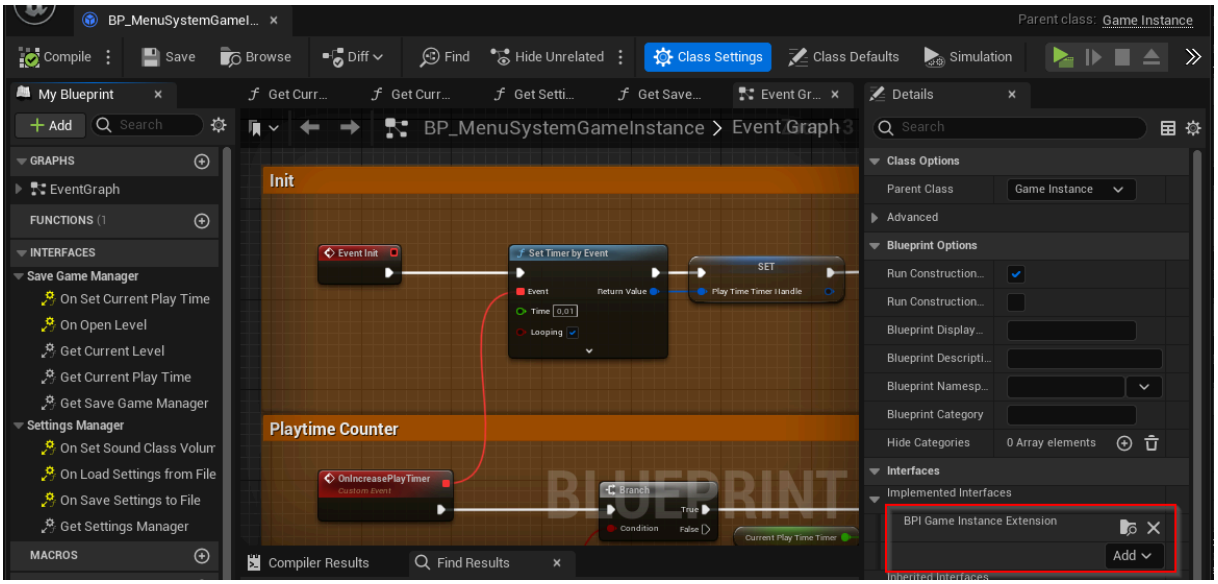
After that your character should be able to open the Ingame Menu using the specified Input Action. Make sure to use the Enhanced Input Actions for all Inputs in your Character, otherwise the Input might not work at all.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

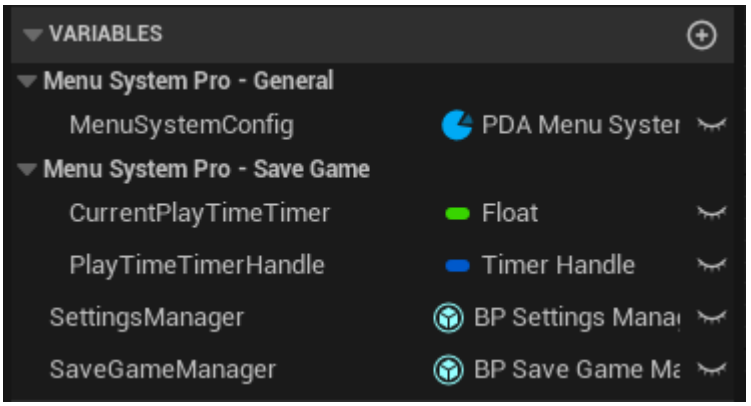
Setup your existing Game Instance

If you wish to use your own Game Instance, follow this guide. All other users can just use the existing Game Instance 'BP_MenuSystemGameInstance' (as explained above in the Installation chapter).

1. Add the 'BPI_GameInstanceExtension' interface to your custom Game Instance

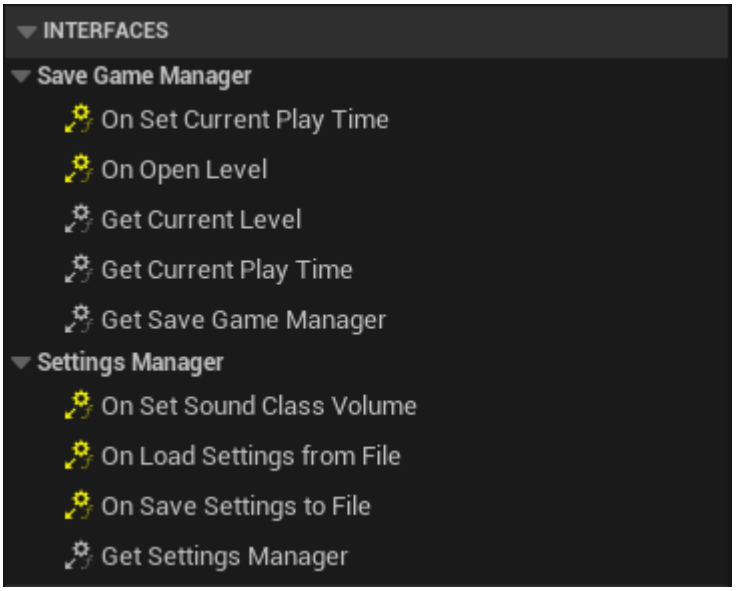


2. Copy all variables from 'BP_MenuSystemGameInstance'

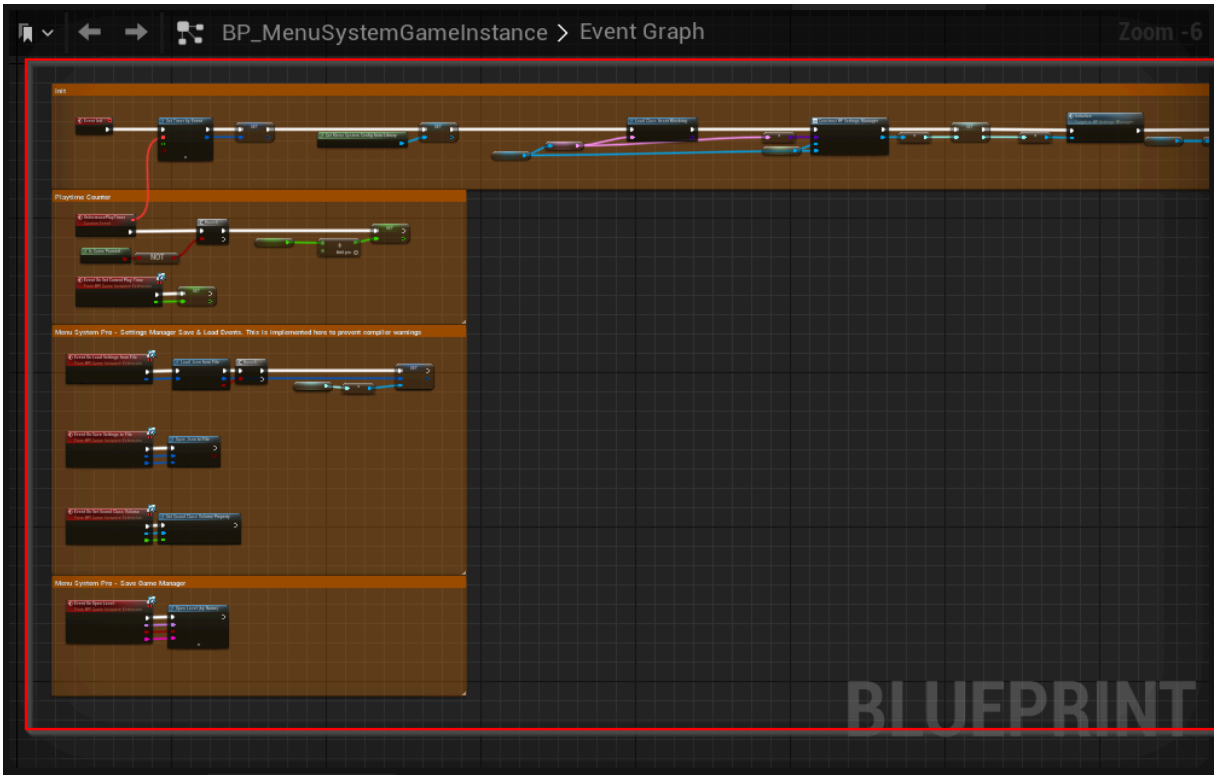


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

3. Implement all functions that 'BP_MenuSystemGameInstance' implements



4. Copy & Paste the nodes on the event graph from 'BP_MenuSystemGameInstance'



5. Compare both your game instance and our game instance side-by-side and ensure you have correctly connected all nodes

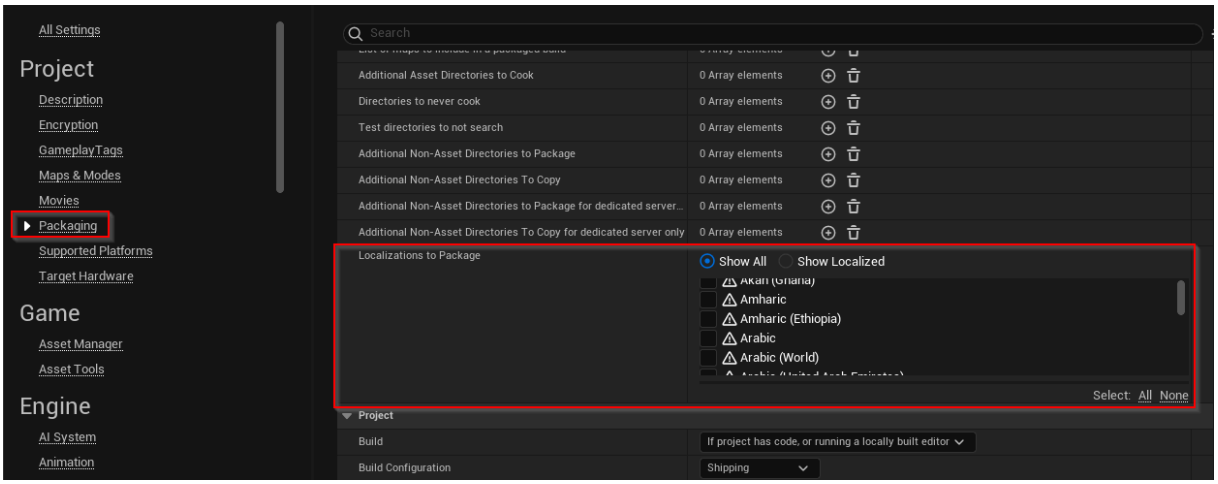
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Packaging your project

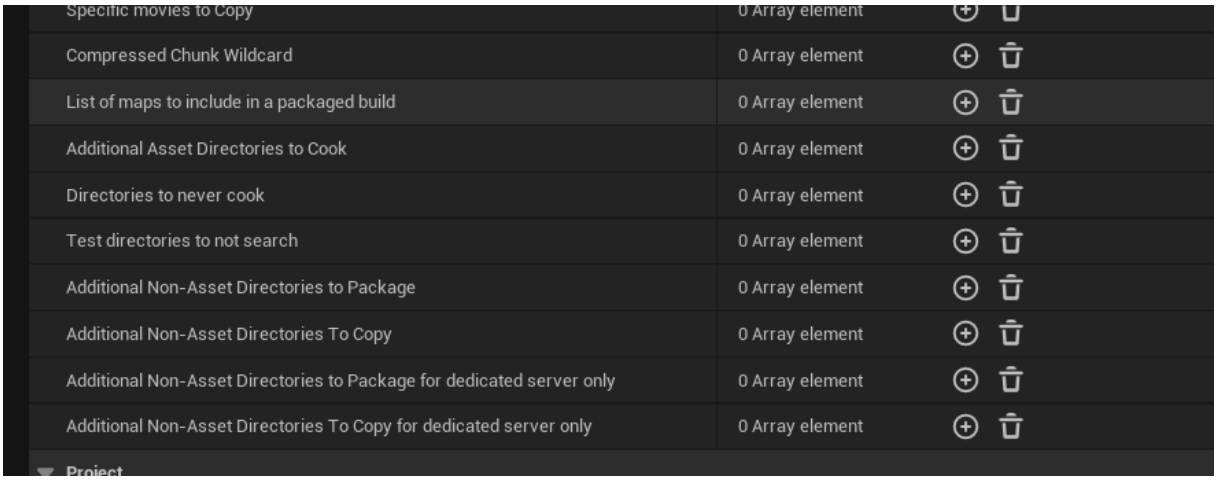
Before you can ship your game with localization support you need to change some of your Project Settings.

1. Go to Project Settings/Packaging
2. Expand the Packaging/Advanced section
3. Select 'Localizations to Package' (e.g. German and English)



4. Set 'Internationalization Support' to 'All'

5. Also add all your maps/levels you want to package



6. You can now package your project

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

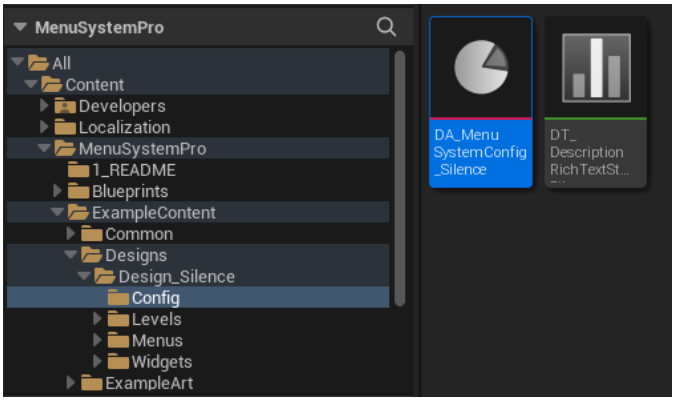
Customization

In this section you will learn how to modify the overall appearance of the UI.

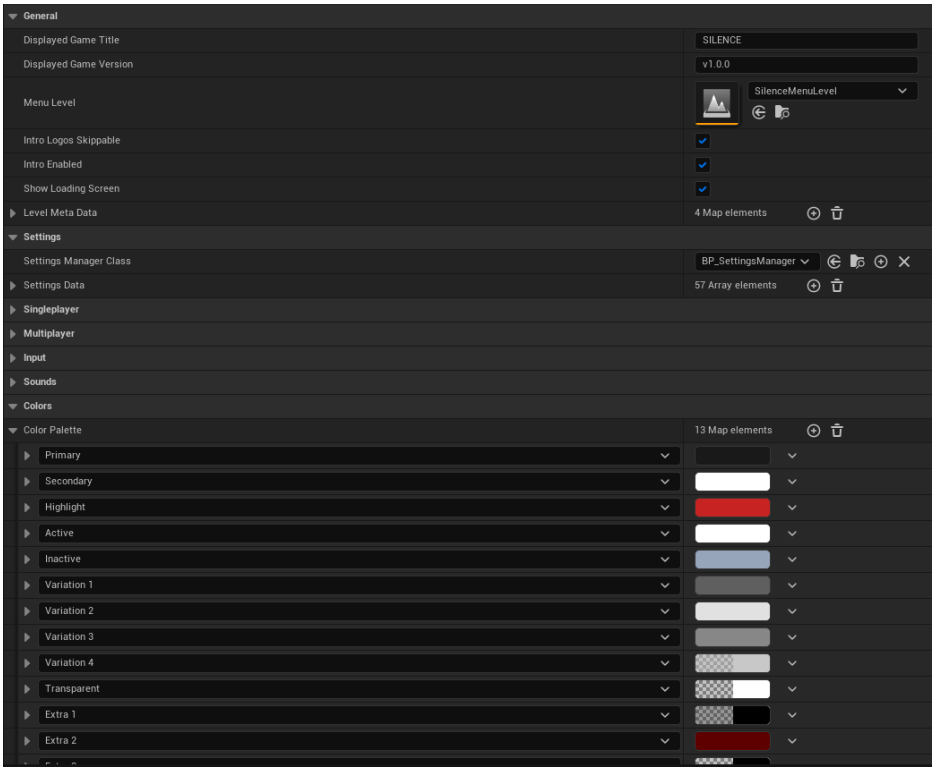
Menu System Config

The Menu System Config allows for simple adjustment of global settings, enabling you to customize the menu system according to your preferences. It includes various parameters for both functionality and styling of the menu.

You can find it at '*/ExampleContent/Designs/Design_%DesignName%/Config/*'



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

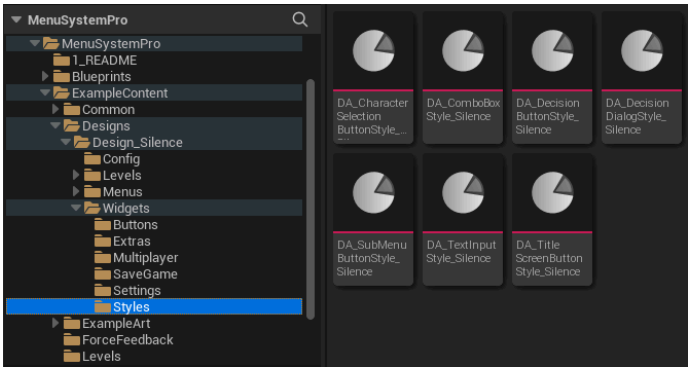


Note: Hovering over each parameter will provide you with a detailed explanation of its effect.

Widget Style Data

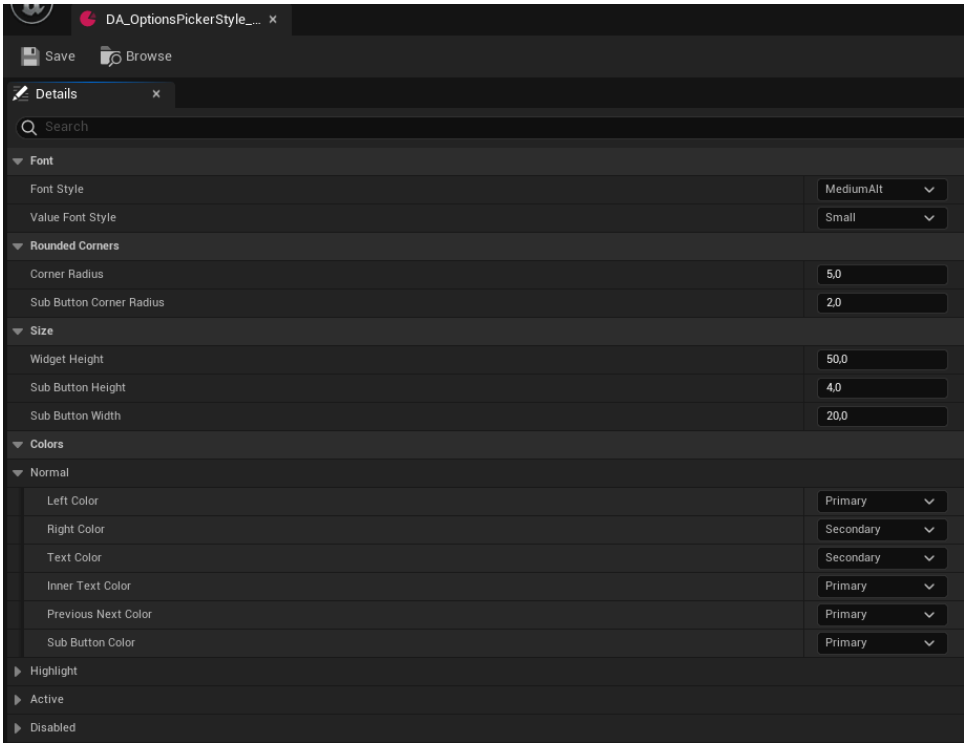
With the Widget Style Data Assets, you can easily customize the appearance of each Interactable User Widget in Menu System Pro by adjusting its style settings such as Font, Size or Colors.

The Data Assets are located at '*/ExampleContent/Designs/Widgets/Styles*'



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Example of the OptionsPicker Style Data Asset:



Fonts and Colors correspond to the ones specified in the Menu System Config.

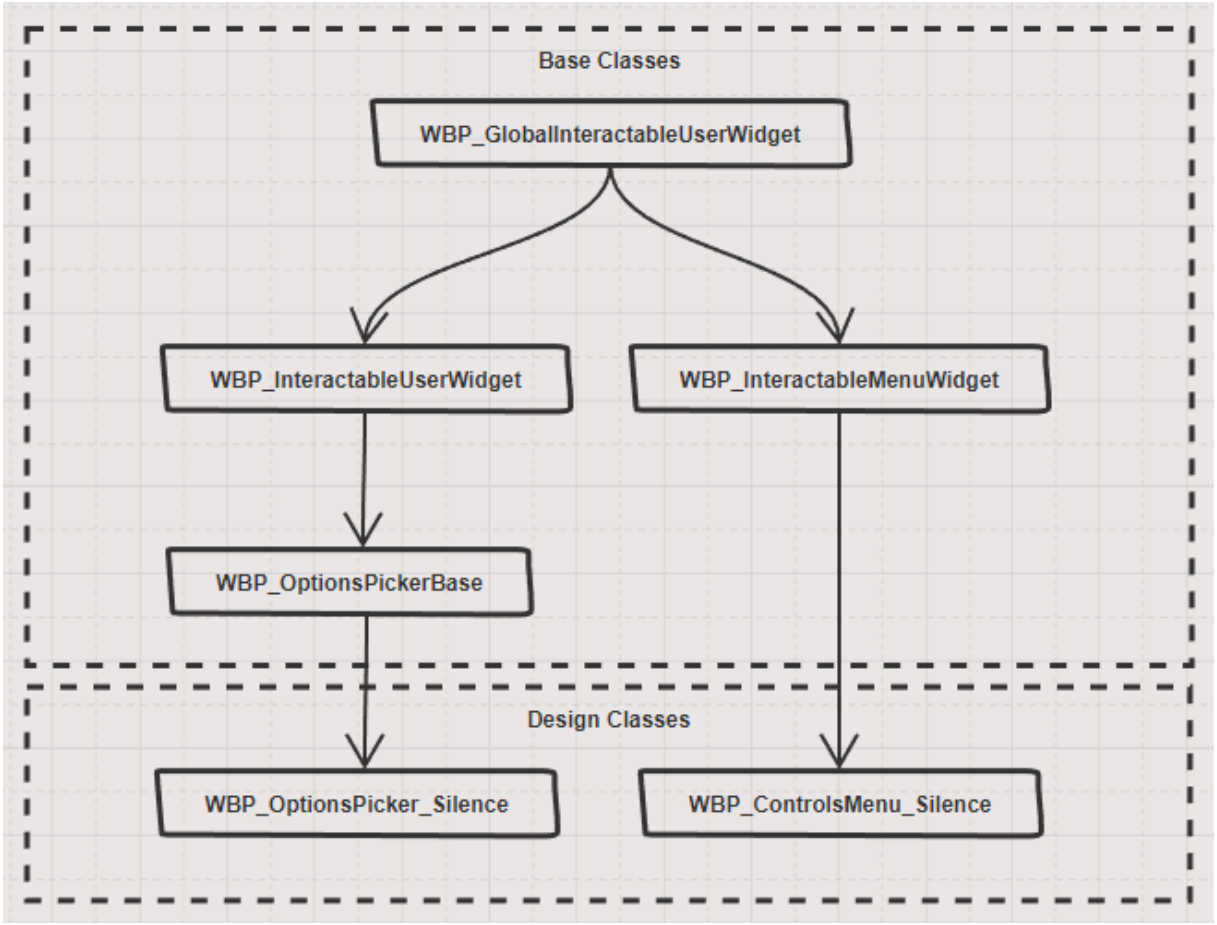
The Widget Style Data Asset is changeable on all Interactable User Widgets in every menu. Here is an example of the OptionsPicker in the Controls General Menu:

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Widget & Menu Designs

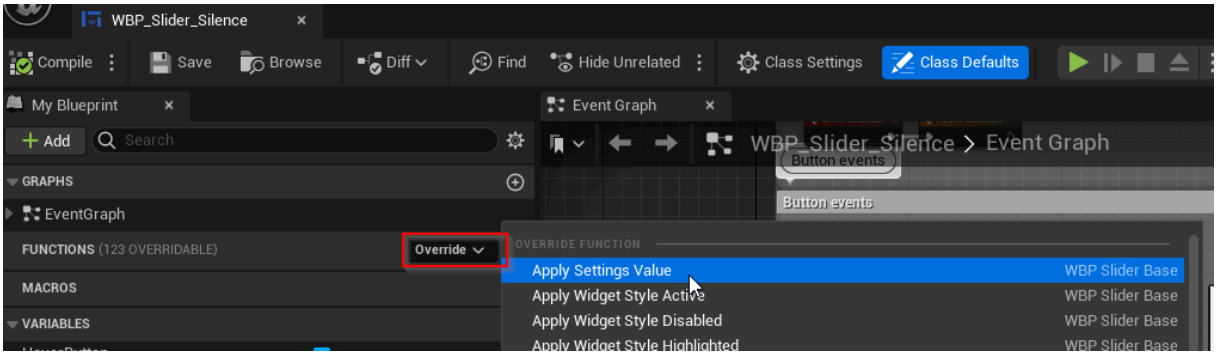
In Menu System Pro v2, we made an effort to maximize the separation between widget design and logic. Each design widget is derived from a base class that contains the majority of the code.

The following illustration provides an example of how the basic logic and design is separated:



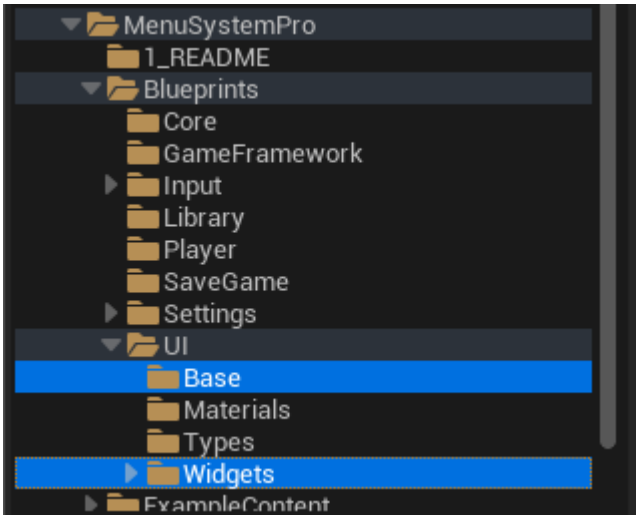
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

It is advised to only add changes and custom logic in the design classes. For example through function overrides like demonstrated in the Slider here:



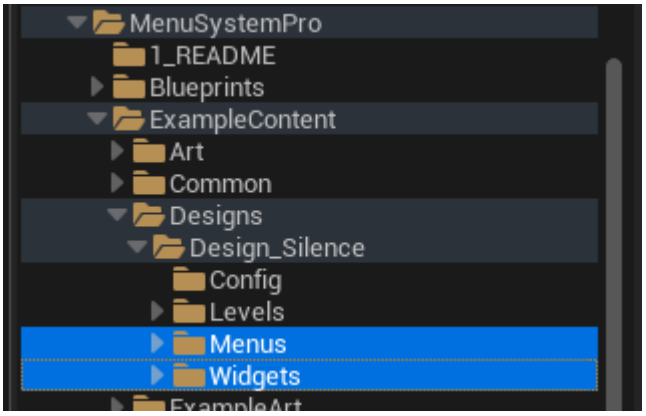
This will save your modifications from being overridden by updates.

You can find the base classes in the *'Base'* and *'Widgets'* folders located at *'/Blueprints/UI/'*.



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

The design classes can be found in the *'Menus'* and *'Widgets'* folders located at *'/ExampleContent/Designs/Design_%DesignName%/'*.



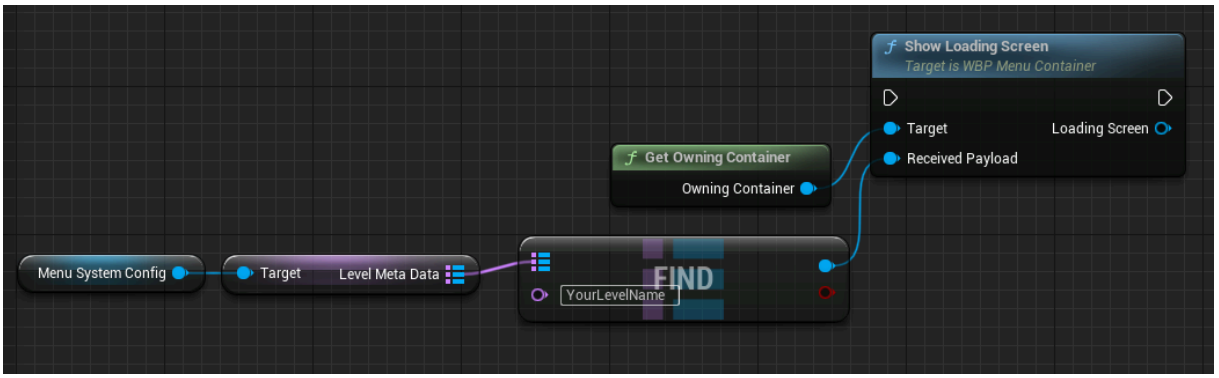
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Special Menus

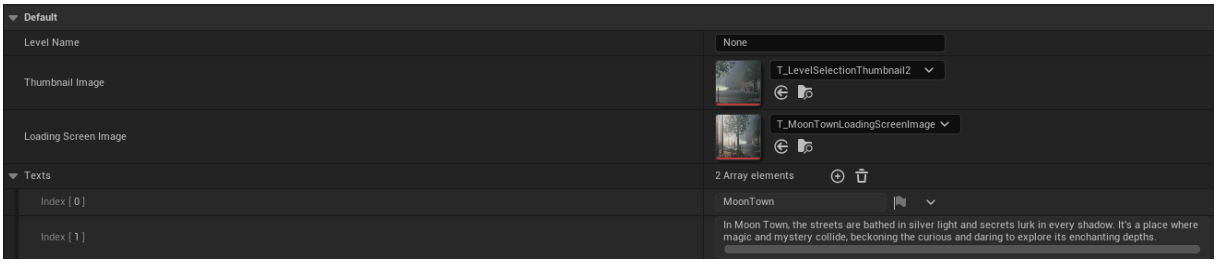
There are menus that are used in a different way throughout the entire menu system. Some of them are always visible (Background) and others are only shown in a special context (Loading Screen, Footer, Decision Dialog).

Loading Screen

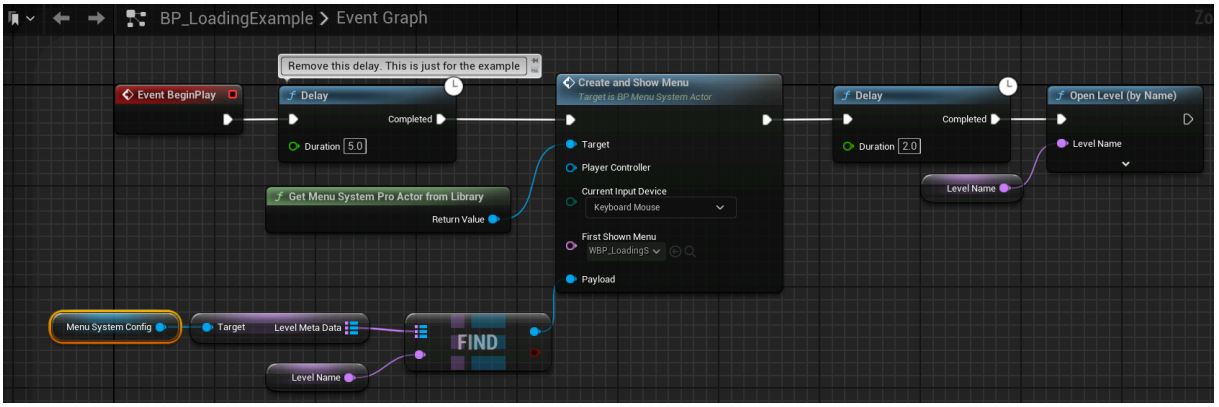
The loading screen can be triggered from any menu like this:



To make the Loading Screen show your custom text and graphics you will need to create a level metadata asset. You can find examples here: *'/ExampleContent/Designs/Design_%DesignName%/Levels/MetaData/'*



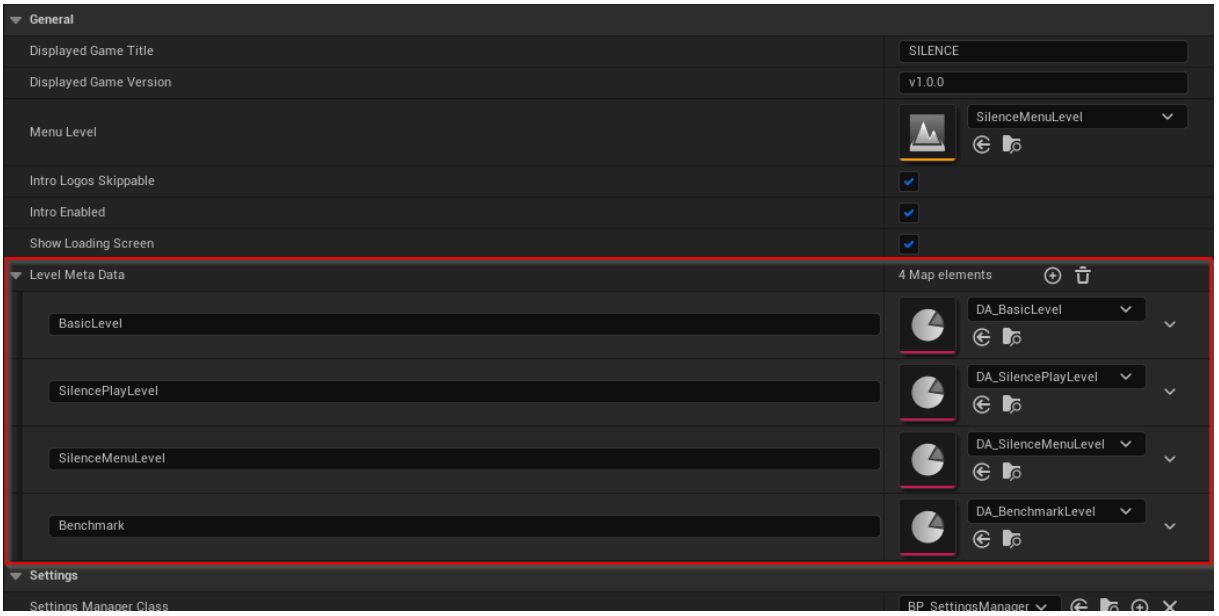
You can also call the Loading Screen from any other actor:



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

MenuSystemConfig is a variable of this actor that has the value set to your Config data asset. LevelName is a Name variable with your level name as value.

When the metadata asset is created you can add it to the Menu System Config here:



Important: The key of the map entry for the Data Asset has to contain the exact level name!

If you like to change the Loading Screen widget layout you can access it here: *'/ExampleContent/Designs/Design_%DesignName%/Menus/LoadingScreen/WBP_LoadingScreenMenu_%DesignName%'*

Background

The Background is always shown throughout the menus and can be changed to show images, material effects or just a semi-transparent black background like we did.

You can access and change the design here: '/ExampleContent/Designs/Design_%DesignName%/Menus/Special/WBP_MenuBackground_%DesignName%'

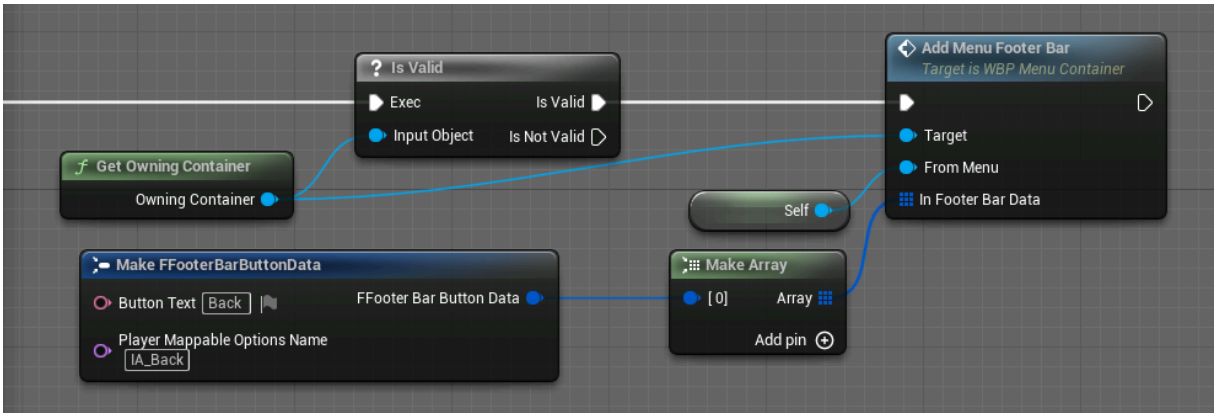
Make sure to check and modify the code on the Event Graph if needed.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

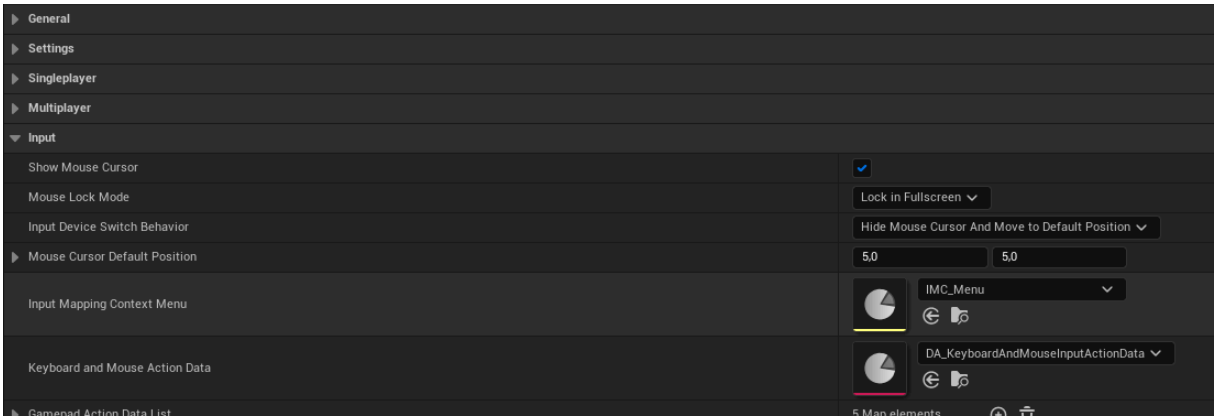
Footer

The Footer can be dynamically created in a menu

This can be done using this code:



This adds one button with the specified Menu Input Action. You can see the available Input Actions in the Menu System Config here:

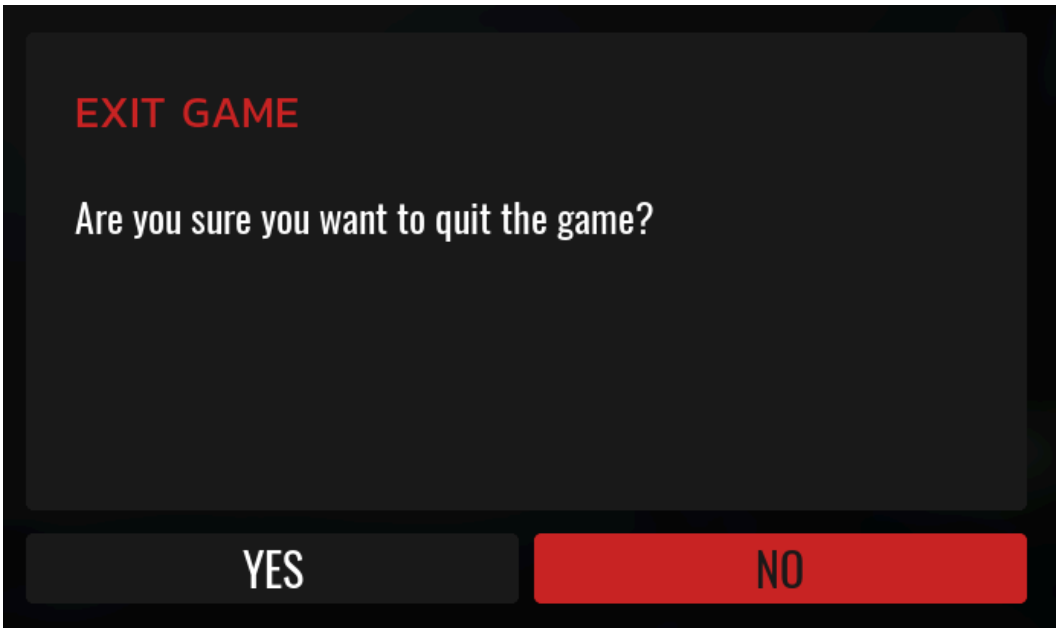


If you like to change or add Input Actions, we advise you to create an own InputMappingContext outside of our folder structure (to make it update proof).

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

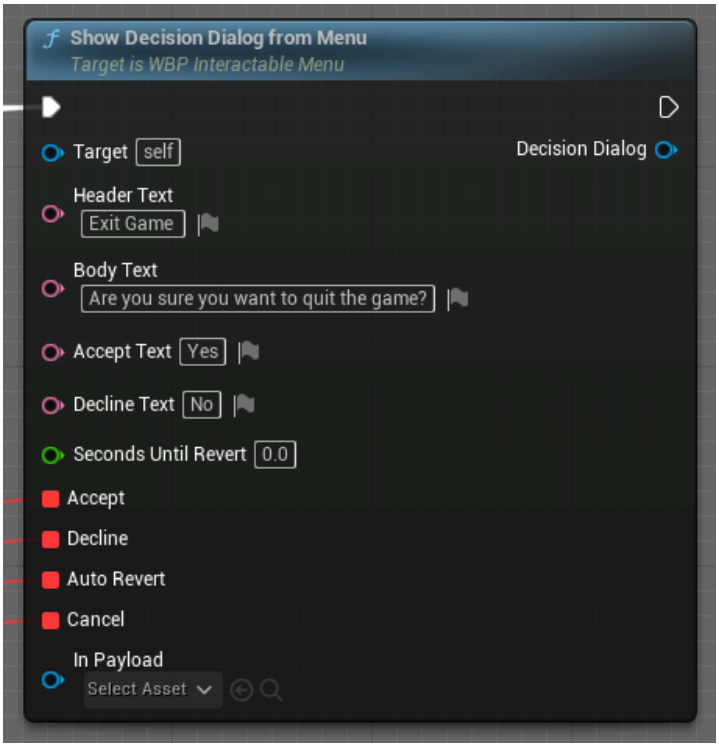
Decision Dialog

The Decision Dialog can be dynamically created when you need to process a user decision



It can also be used for acknowledgement dialogs (one button only; like 'OK').

In menus you can call it using this node:



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

A good implementation example can be found in this Blueprint at the bottom of the Event Graph:
'/ExampleContent/Designs/Design_%DesignName%/Menus/TitleScreen/WBP_TitleScreenMenu_%DesignName%'

If you want to change the design of the dialog navigate to:
'/ExampleContent/Designs/Design_%DesignName%/Menus/Special/WBP_DecisionDialog_%DesignName%'

Credits

Menu System Pro also offers an auto scrolling credits menu which can be skipped by holding any button for a specific amount of time

The content can be easily changed in String Table ST_Credits located at:
'/ExampleContent/Text/Menu/ST_Credits'

If you want to alter the layout or hook up your own string table for better updatability navigate to:
'/ExampleContent/Designs/Design_%DesignName%/Menus/Special/WBP_Credits_%DesignName%'

Customization Guides

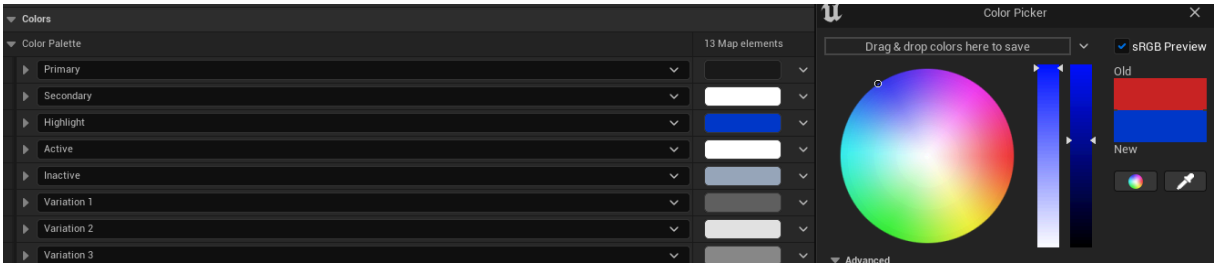
In this section you will learn how to modify the menu and make the menu fit your game.

Change Colors

1. Navigate to /ExampleContent/Designs/Design_%DesignName%/Config/
2. Open DA_MenuSystemConfig_%DesignName%
3. Expand the Colors section and the Color Palette in it

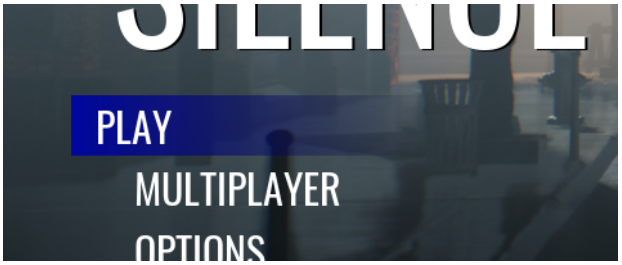
These are all the colors used throughout the menu system at one place! We have a function called *OverrideWidgetStyleFromConfig* every widget based on *Interactable User Widget* or *Interactable Menu Widget,* that reads colors, fonts etc. from the Config and sets them to the actual UI widgets (in Pre Construct).

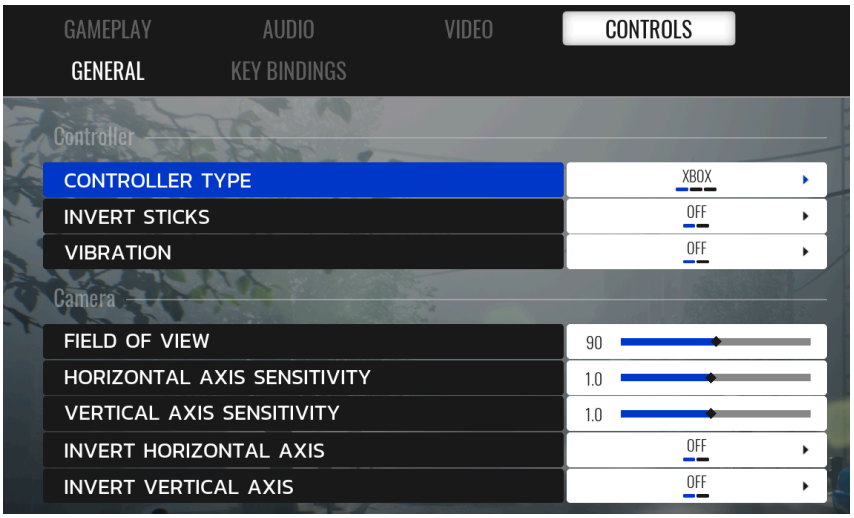
4. For this example we will change the Highlight color to blue



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

5. Hit Save, Play and Enjoy!



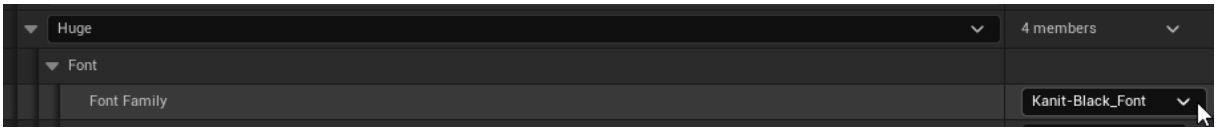


Change Fonts

1. Navigate to /ExampleContent/Designs/Design_%DesignName%/Config/
2. Open DA_MenuSystemConfig_%DesignName%
3. Expand the Fonts section and the Fonts Palette in it

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

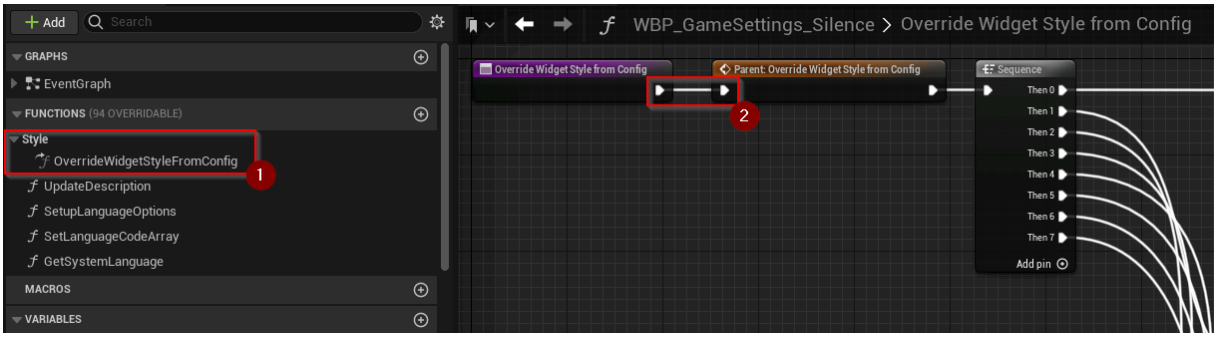
4. For this example we will change the Huge font, which is displayed on the Title Screen Menu



5. Hit Save, Play and Enjoy!

Remove Design Override

This small guide will show you how to remove the design override we have explained before. After removing it from a widget or menu, you have full control over the widgets styling.



1. Open OverrideWidgetStyleFromConfig function in any design widget
2. Break the connection here or for a specific thing in the sequence on the right-hand side

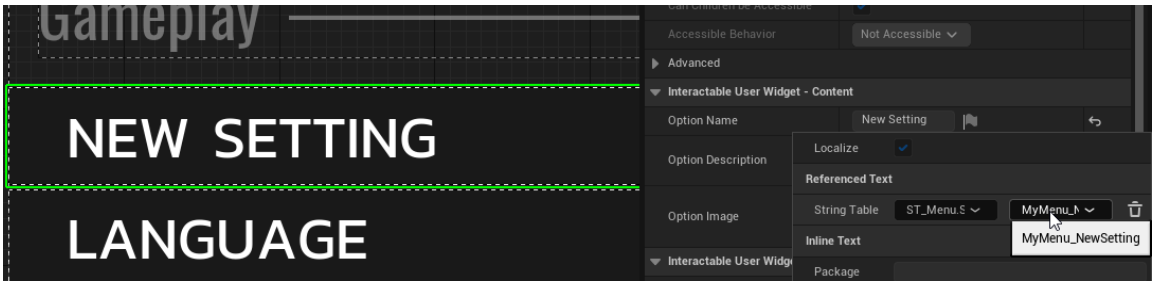
Note: We do not recommend breaking the connection, as this will disconnect the MenuSystemConfig from this widgets styling. Please only disconnect the override function if you are certain that it is necessary for your particular use case.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Change Widget Text

1. Navigate to /ExampleContent/Designs/Design_Silence/Menus/
2. Open any menu you like
3. Click on a widget in the menu and have a look at the right-hand side

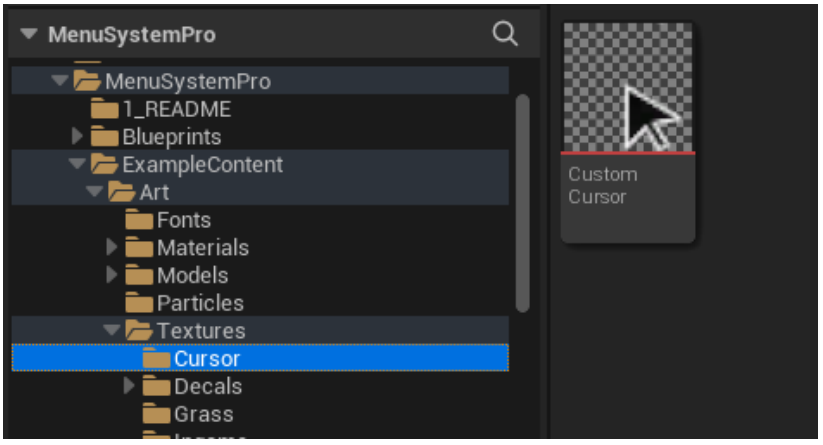
4. You can set the Option Name and Description there



   You might have to remove the String Table binding by clicking on the flag.
5. It is recommended to use your own String Table and link it there. This will be beneficial when doing localization.

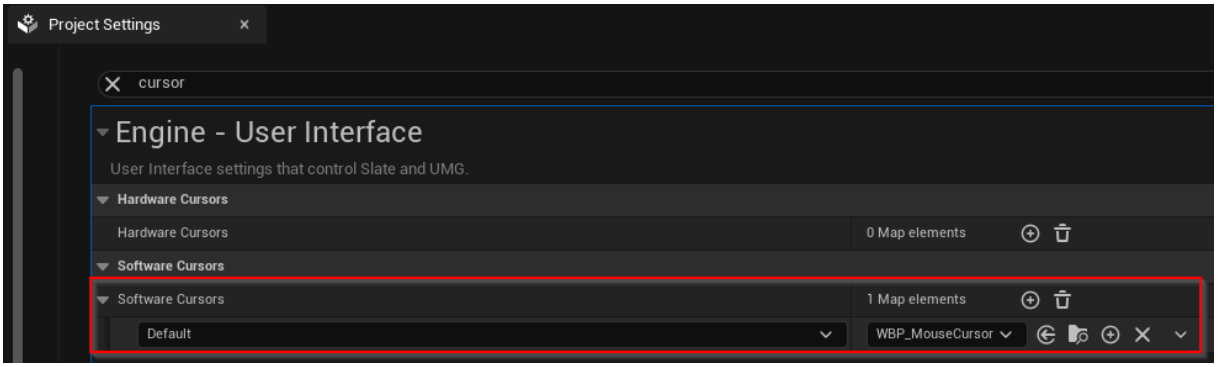
Set Custom Cursor

1. Navigate to /ExampleContent/Art/Textures/Cursor/
2. Replace the CustomCurser icon with your own cursor icon. This should be the same size as the example. The cursor tip should be in the center of the image.



   Note: Alternatively, you can open/Blueprints/UI/Widgets/Input/WBP_MouseCursor and change the cursor image to your desired image.
3. Now go to the Project Settings and type "*cursor*" into the search field
4. Set Software Cursors to *default* and add the *CursorWidget*

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

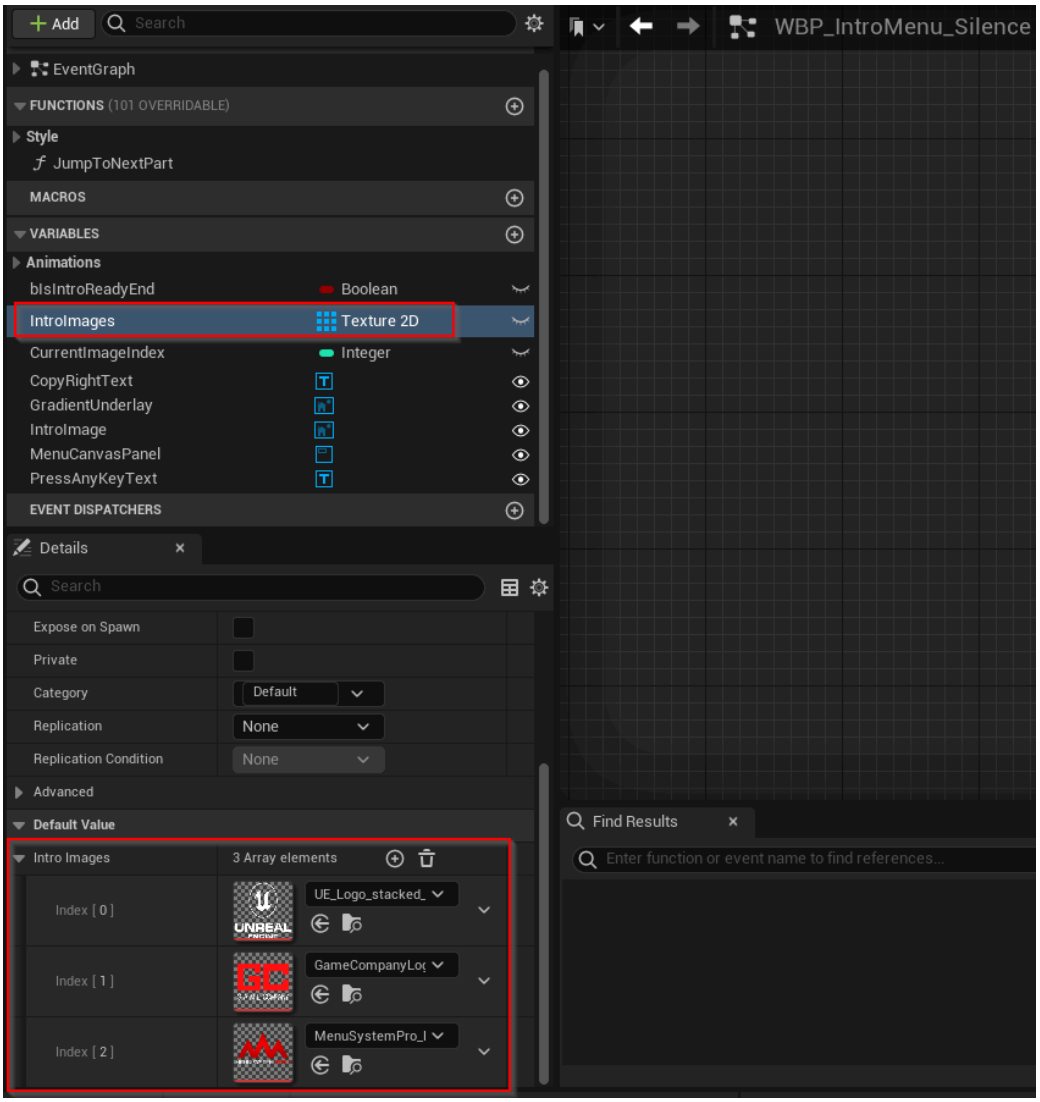


5. The cursor is now changed, and you can close the Project Settings

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Edit Intro Images

1. Navigate to /ExampleContent/Designs/Design_%DesignName%/Menus/TitleScreen/
2. Open WBP_IntroMenu_%DesignName%
3. Change the IntroImages array according to your needs:

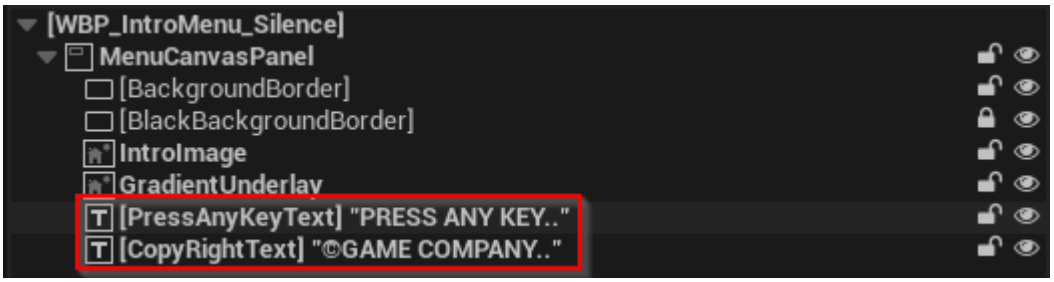


4. Hit Save & Compile.

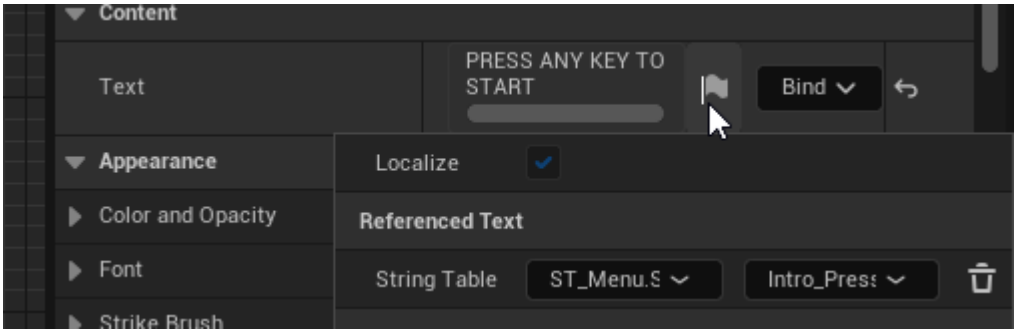
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Edit Intro Texts

1. Navigate to /ExampleContent/Designs/Design_%DesignName%/Menus/TitleScreen/
2. Open WBP_IntroMenu_%DesignName%
3. In Designer tab you can access both texts



4. Change them in the Details window on the right-hand side



You might have to remove the String Table binding by clicking on the flag

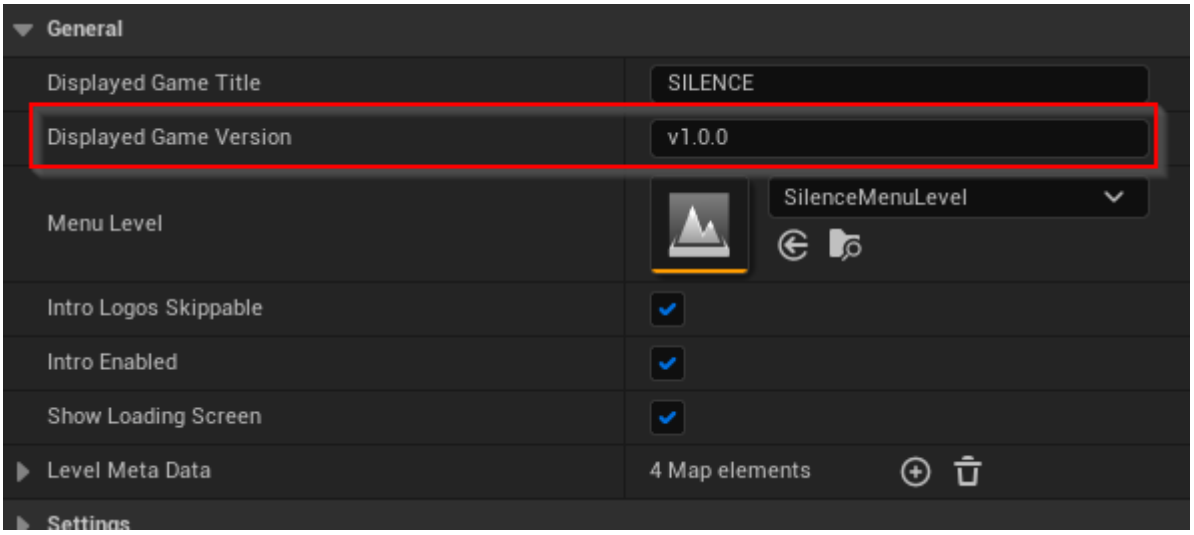
5. Hit Compile & Save

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Set Title Screen Version

Change the version display in the bottom right corner of the title screen menu.

1. Navigate to /ExampleContent/Designs/Design_%DesignName%/Config/
2. Open DA_MenuSystemConfig_%DesignName%
3. Change the Displayed Game Version here



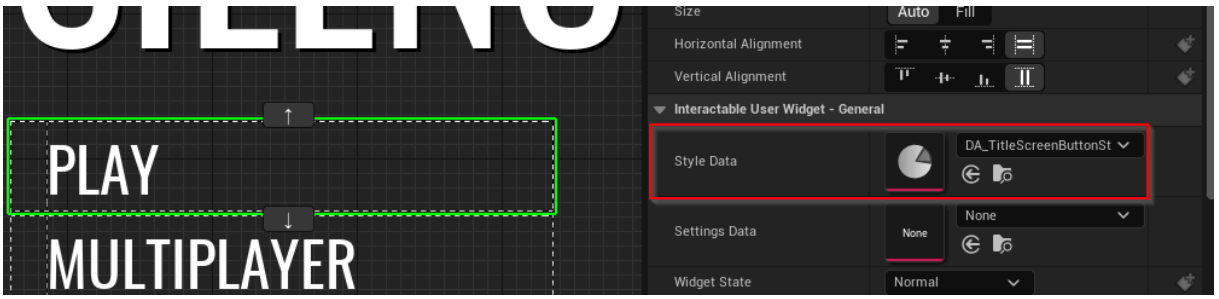
4. Hit Save

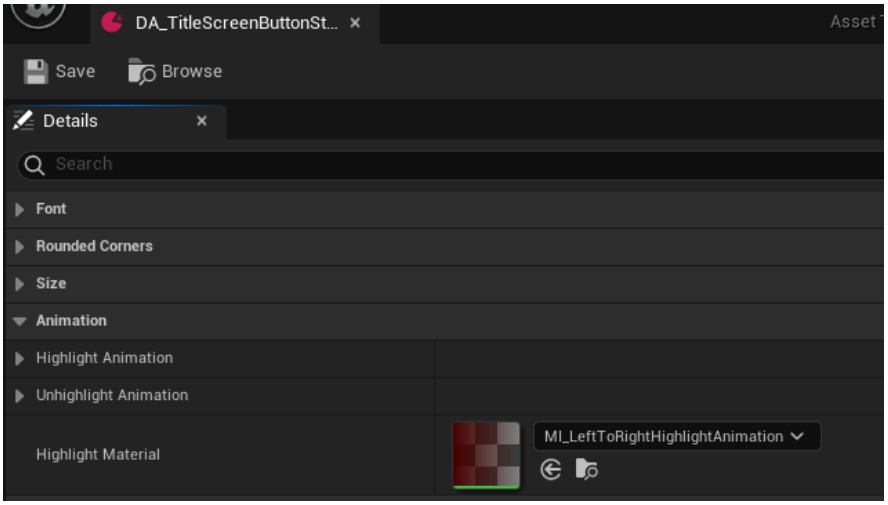
Material Effects in Widgets

We added the possibility to add a highlight material to all classes derived from the WBP_ButtonBase.

In the example above we showed the TitleScreenButton which has a pulsating material effect on it. It is set on the Style Data Asset of the widget.

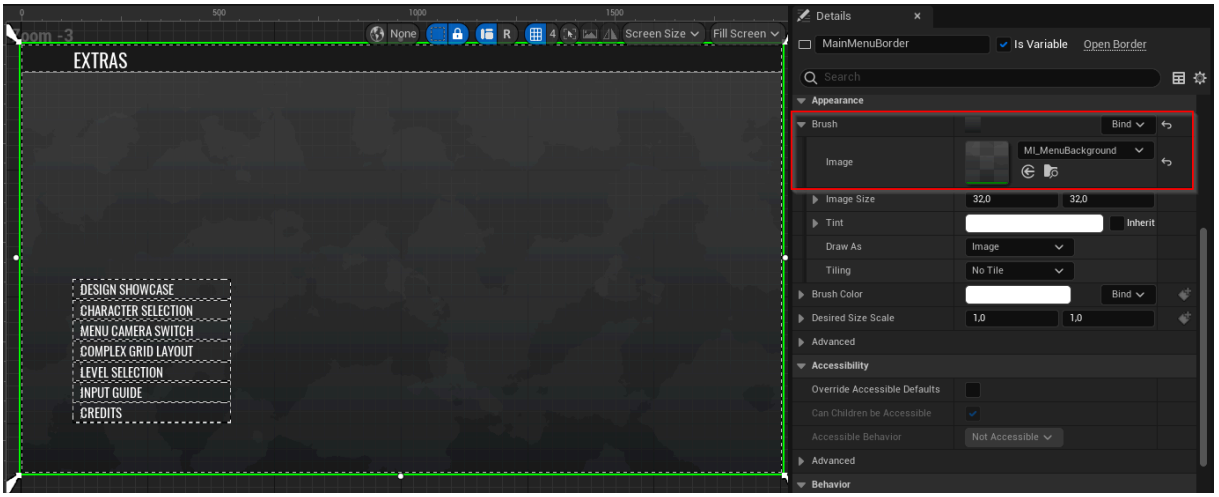
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)





If you want to implement material effects on other widgets, refer to the code we used in the WBP_ButtonBase (search for 'highlight material').

Another example for using material effects would be the Extras Menu background. It has an animated cloudy background placed on its MainMenuBorder:



Located at: */ExampleContent/Designs/Design_Silence/Menus/Extras/*

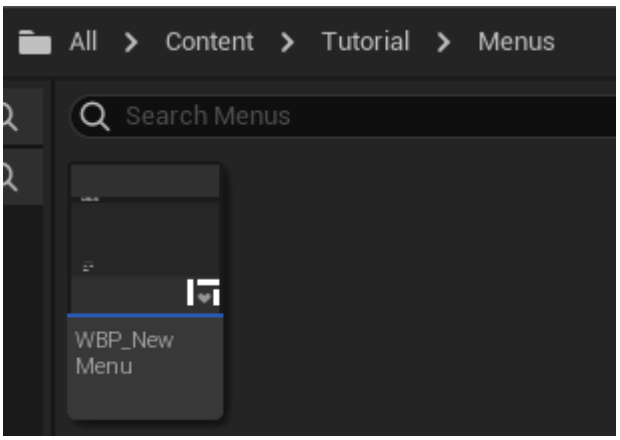
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Be mindful of using material effects as they can significantly impact performance.

Create new menu

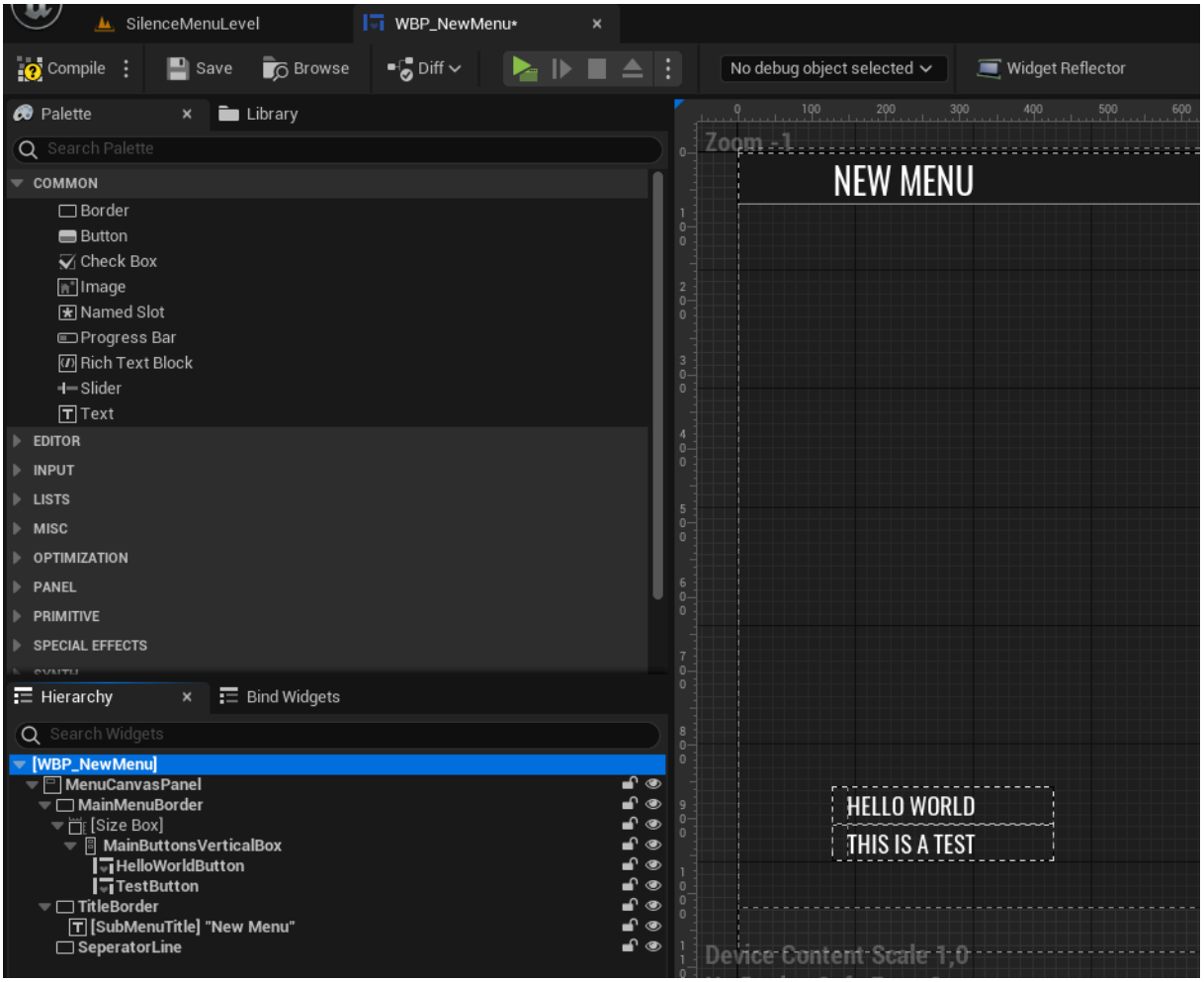
To create a new menu, we can utilize the Singleplayer menu as a foundation. However, if preferred, other existing menus can also be utilized as a template.

1. Navigate to /ExampleContent/Designs/Design_Silence/Menus/Singleplayer/WBP_SingleplayerMenu_Silence
2. Copy & paste it into your own file structure and rename as you wish.



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

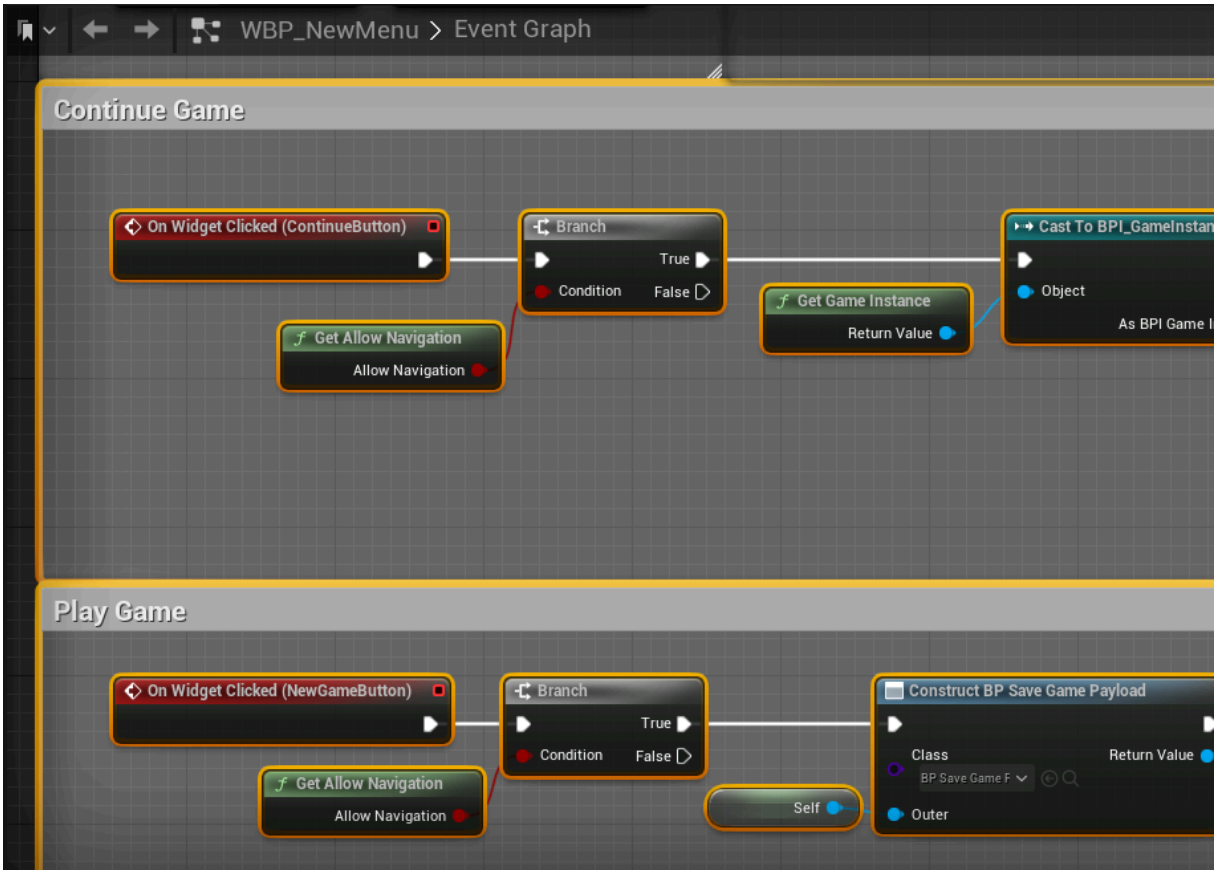
3. Adjust the design to your needs and all the widgets you might need



Make sure to only use widgets that are inherited from Interactable User Widget in your menu. Otherwise the navigation won't work!

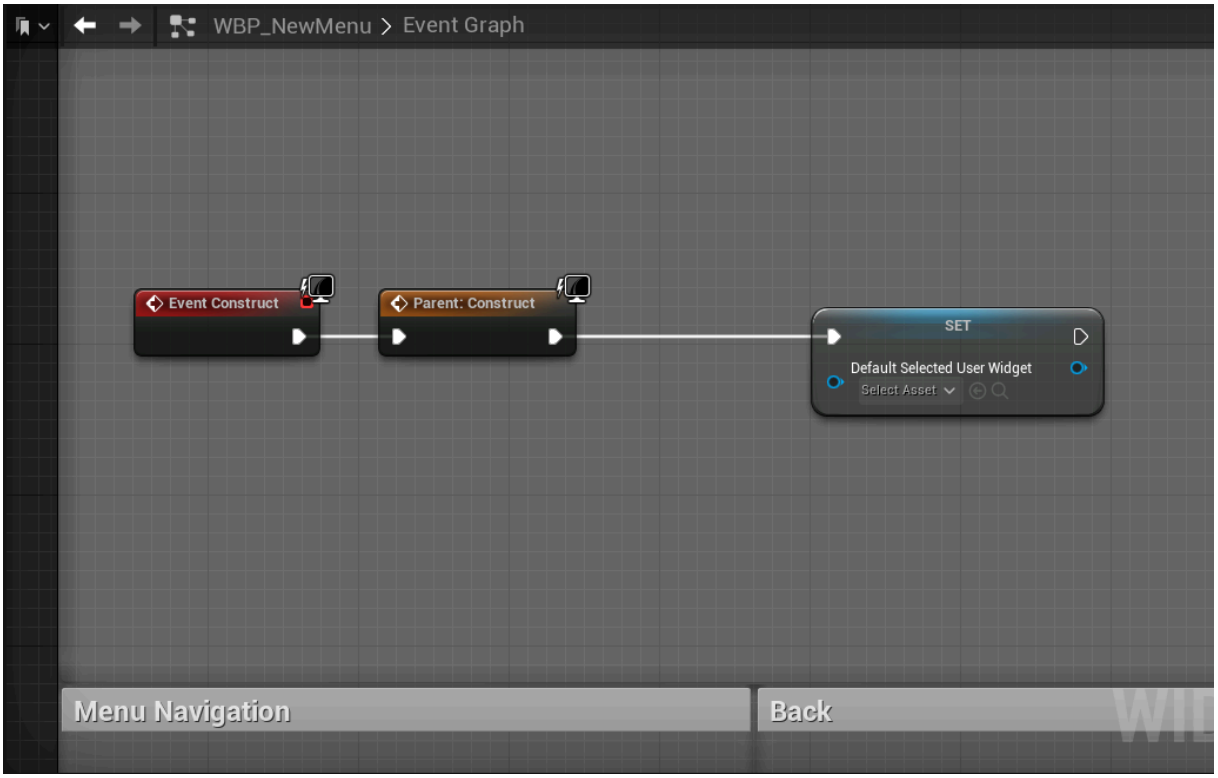
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

4. Then go to the Blueprint Graph and remove all the code from the previous widgets

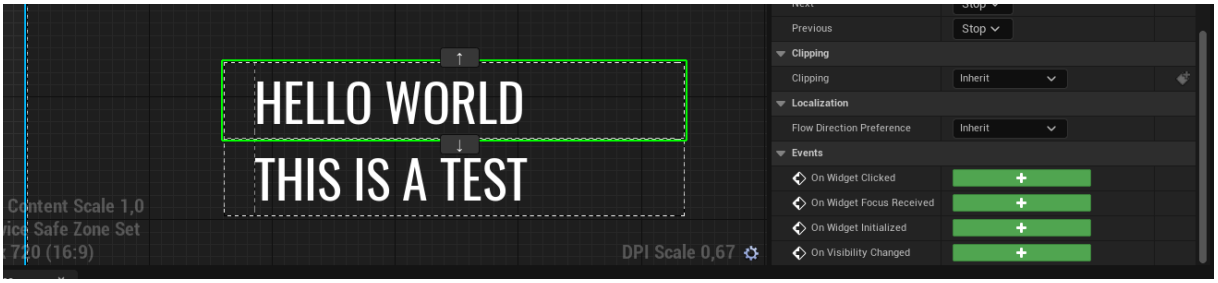


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - [Click here for the new Common UI documentation (MSP 2.2.0+)]

5. Connect the first widget of your menu to the ‘Set Default Selected User Widget’



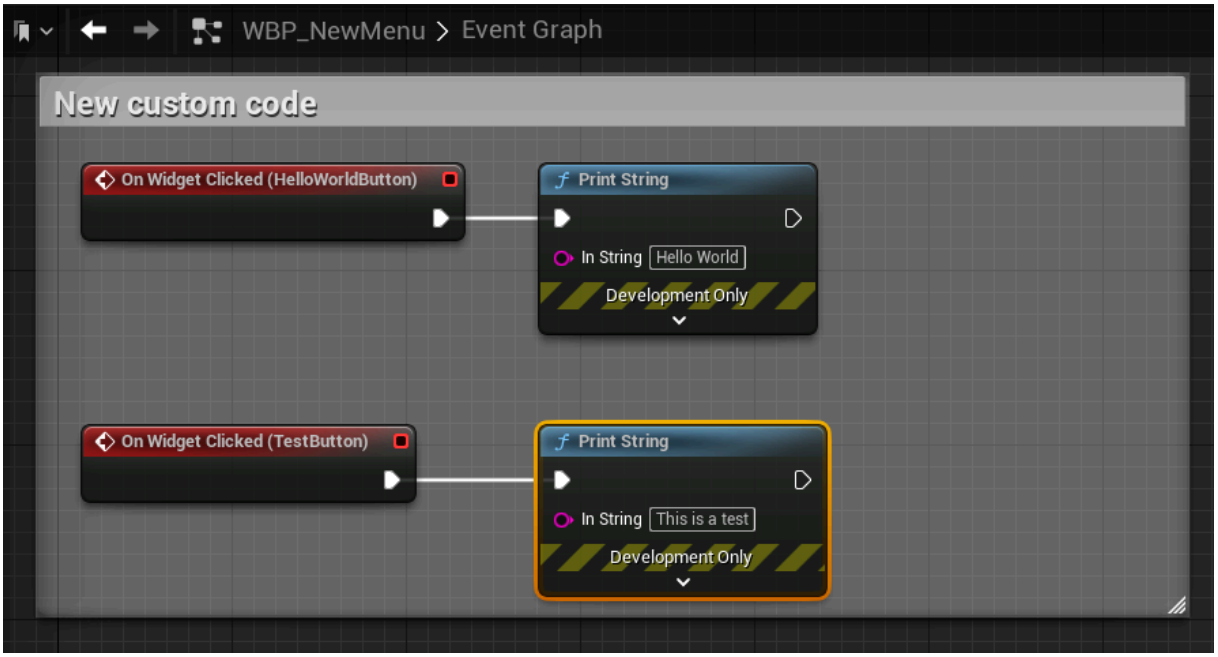
6. Hit compile and fix all the left errors
7. Go back to the Designer tab and create events for the newly added widgets by clicking on the plus next to it



In this case we add the ‘OnWidgetClicked’ event to print something

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - [Click here for the new Common UI documentation (MSP 2.2.0+)]

8. Add your custom code on the Graph



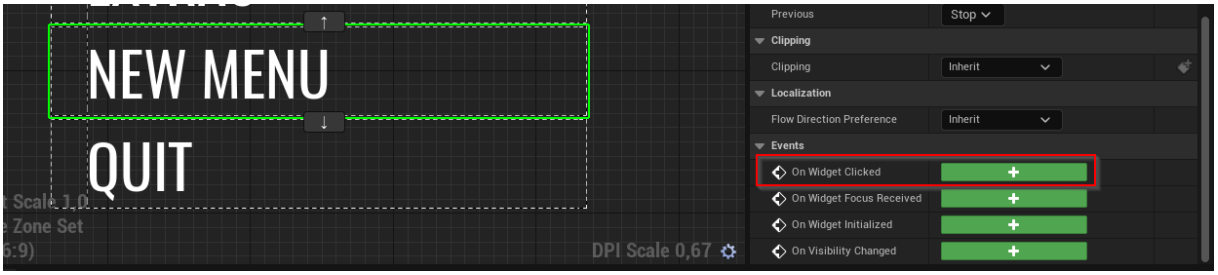
Integrate new menu

In this section you will learn how to integrate your new Menu into the existing Title Screen Menu (can also be used for other menus).

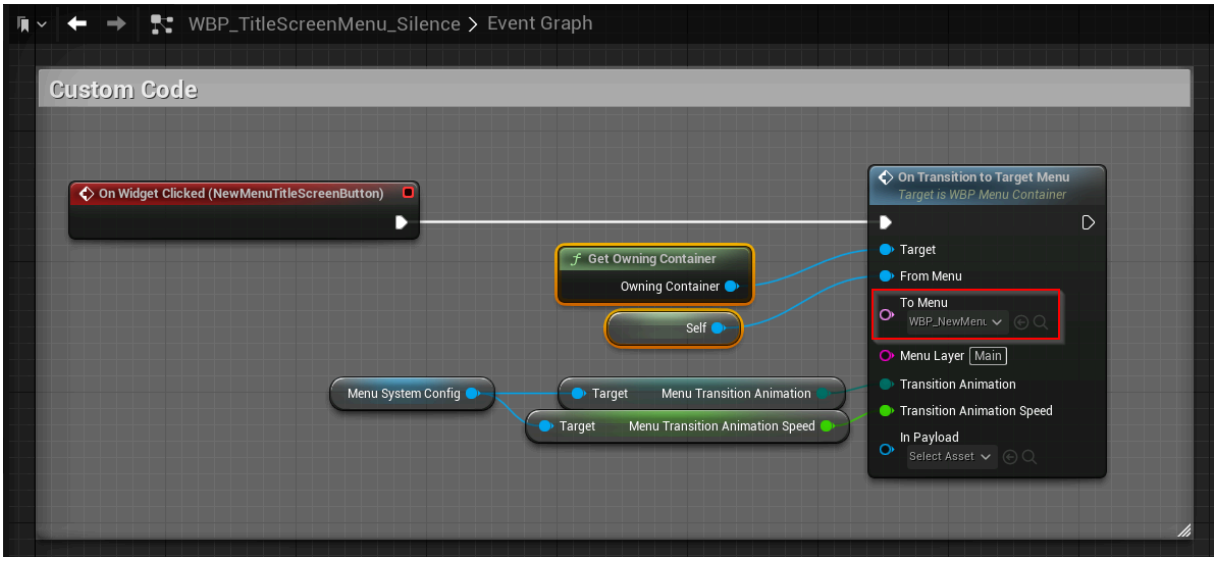
1. Navigate to */ExampleContent/Designs/Design_Silence/Menus/TitleScreen/WBP_TitleScreen Menu_Silence*
2. Add a new button for the navigation to your new menu

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - [Click here for the new Common UI documentation (MSP 2.2.0+)]

3. Add the ‘OnWidgetClicked’ event for your new button to the graph by clicking the plus next to it



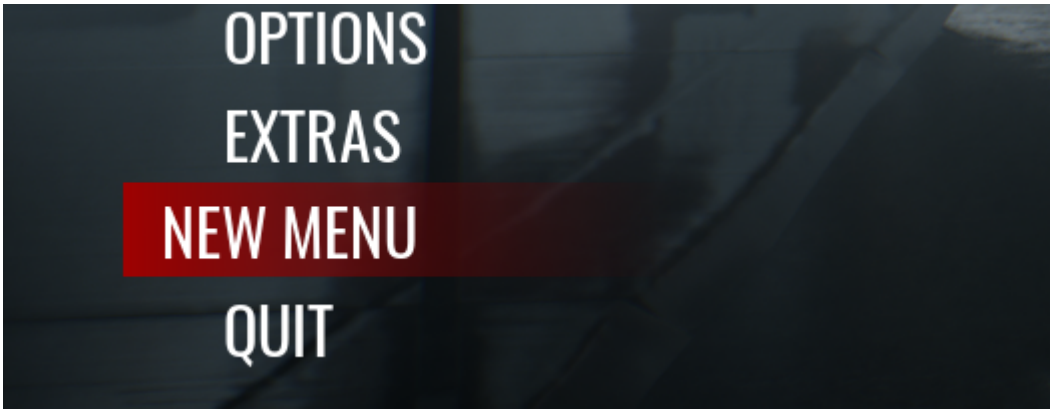
4. Add the following code to the event and select your new menu on the ‘OnTransitionToTargetMenu’ node



5. Hit Compile & Save

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

6. Your new menu should be functional and accessible now

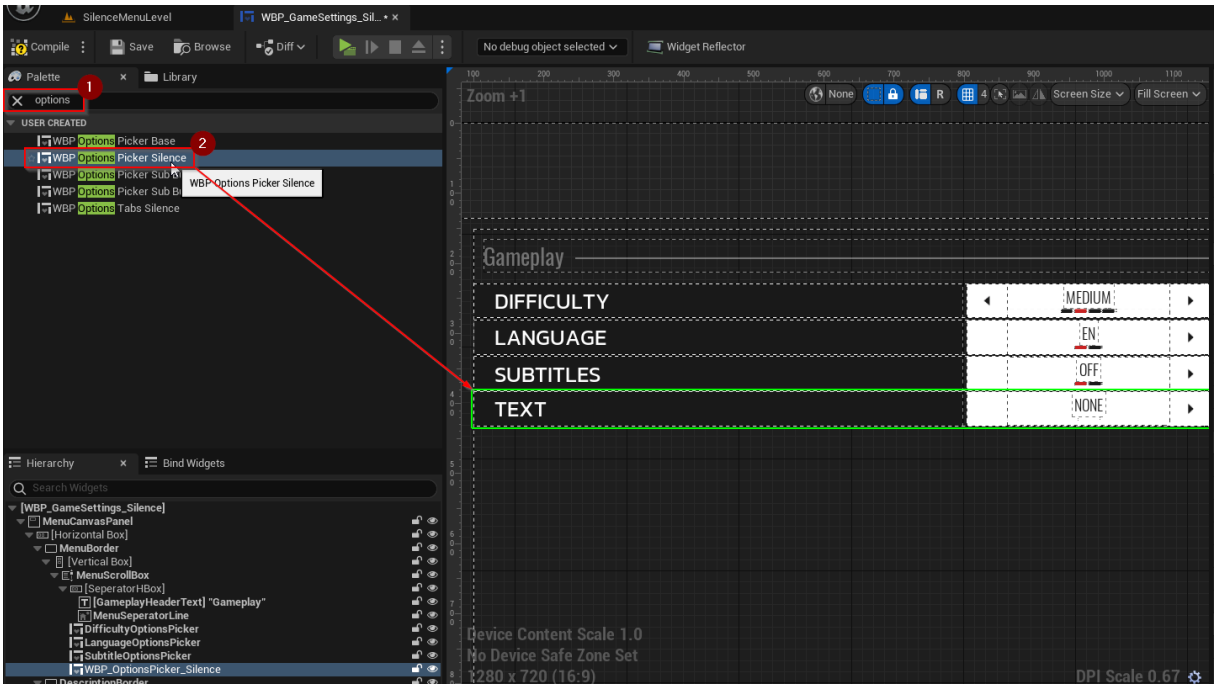


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Add widget to menu

In this section you will learn how to add a new widget to an existing settings menu, with load and save functionality. For this example we will add a new setting to the Game Settings Menu.

1. Add a new Options Picker (Silence design or others) to the menu or duplicate an existing one

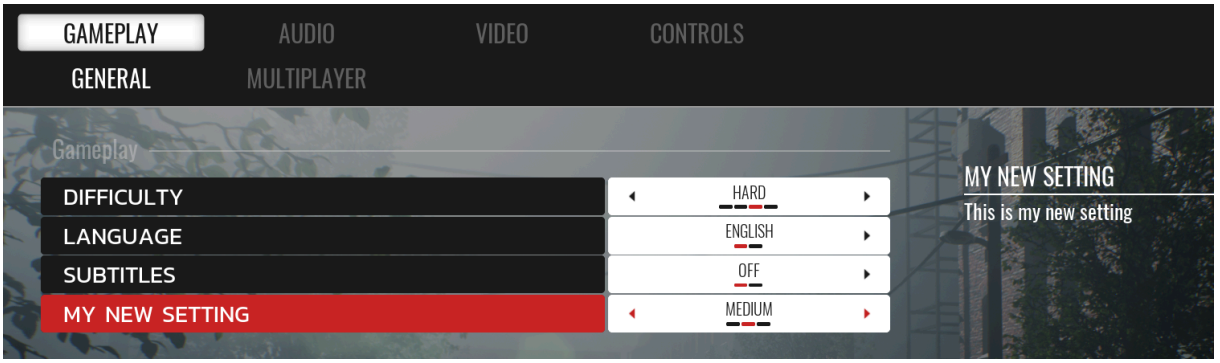


2. Edit the Options Name, Description and Image according to your needs

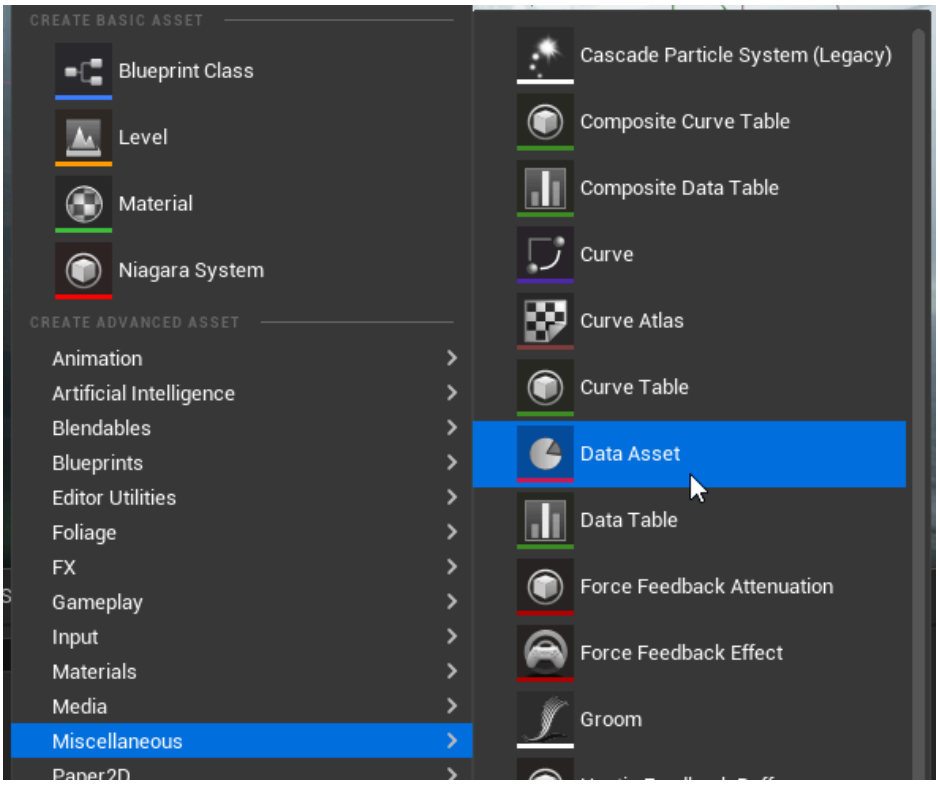
3. Add your options to the Options Array and set the Default Option Index

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - [Click here for the new Common UI documentation (MSP 2.2.0+)]

4. If you now hit save & compile, your widget should already look like this:

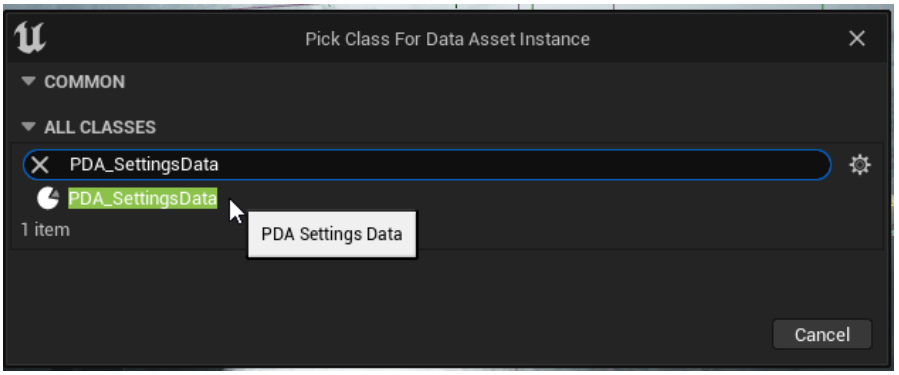


5. Now we need to add the Settings Data asset to save and load this new setting. For this open the Content Browser and create a Data Asset like this:

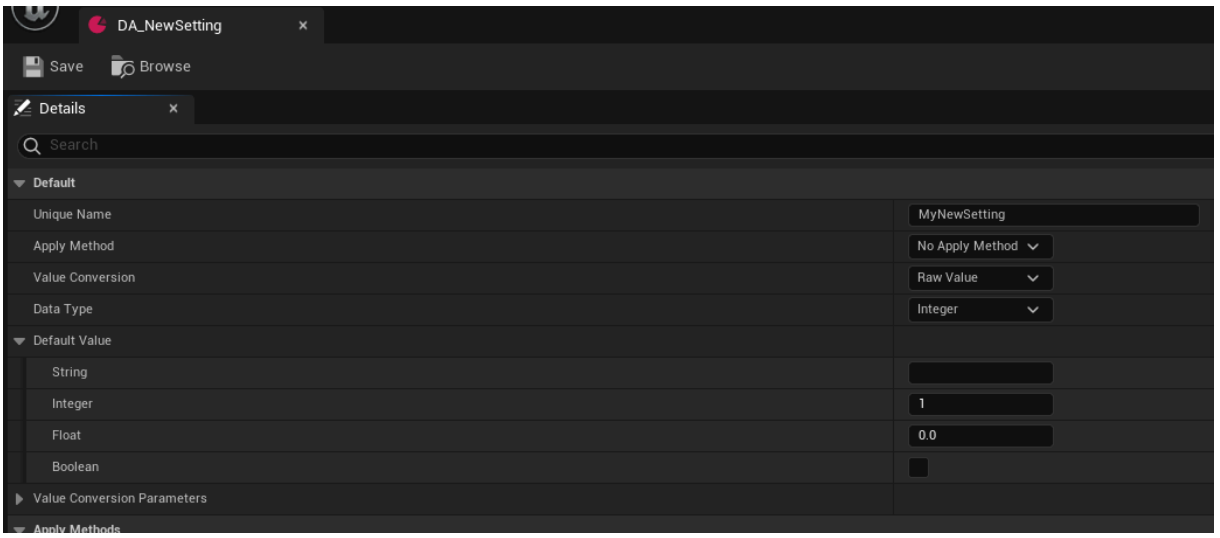


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - [Click here for the new Common UI documentation (MSP 2.2.0+)]

6. Select PDA_SettingsData in the Pop-Up window

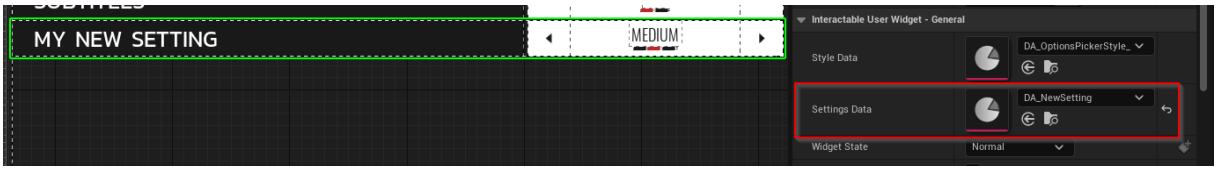


7. Name it 'DA_NewSetting' (or something else starting with 'DA_')
8. Open it up, set the properties according to your needs and save

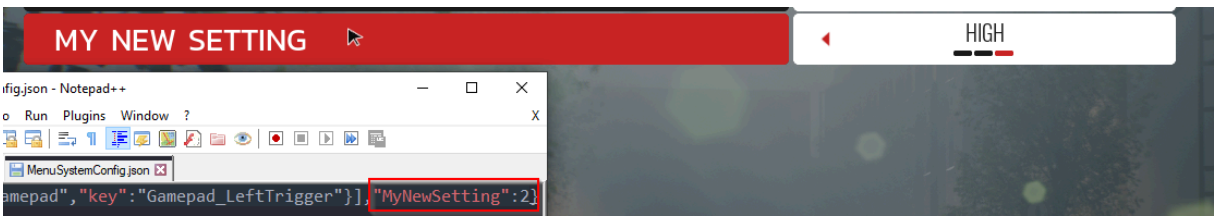


To understand the properties in detail, please refer to the section Settings Data Assets

9. Now go back to the menu and plug it into your newly placed widget



10. When you save it the change will be reflected in the Settings Json File



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

11. Finally add it to the Menu System Config's Settings Data array

Doing this will enable the Settings Manager to load the setting UI independent

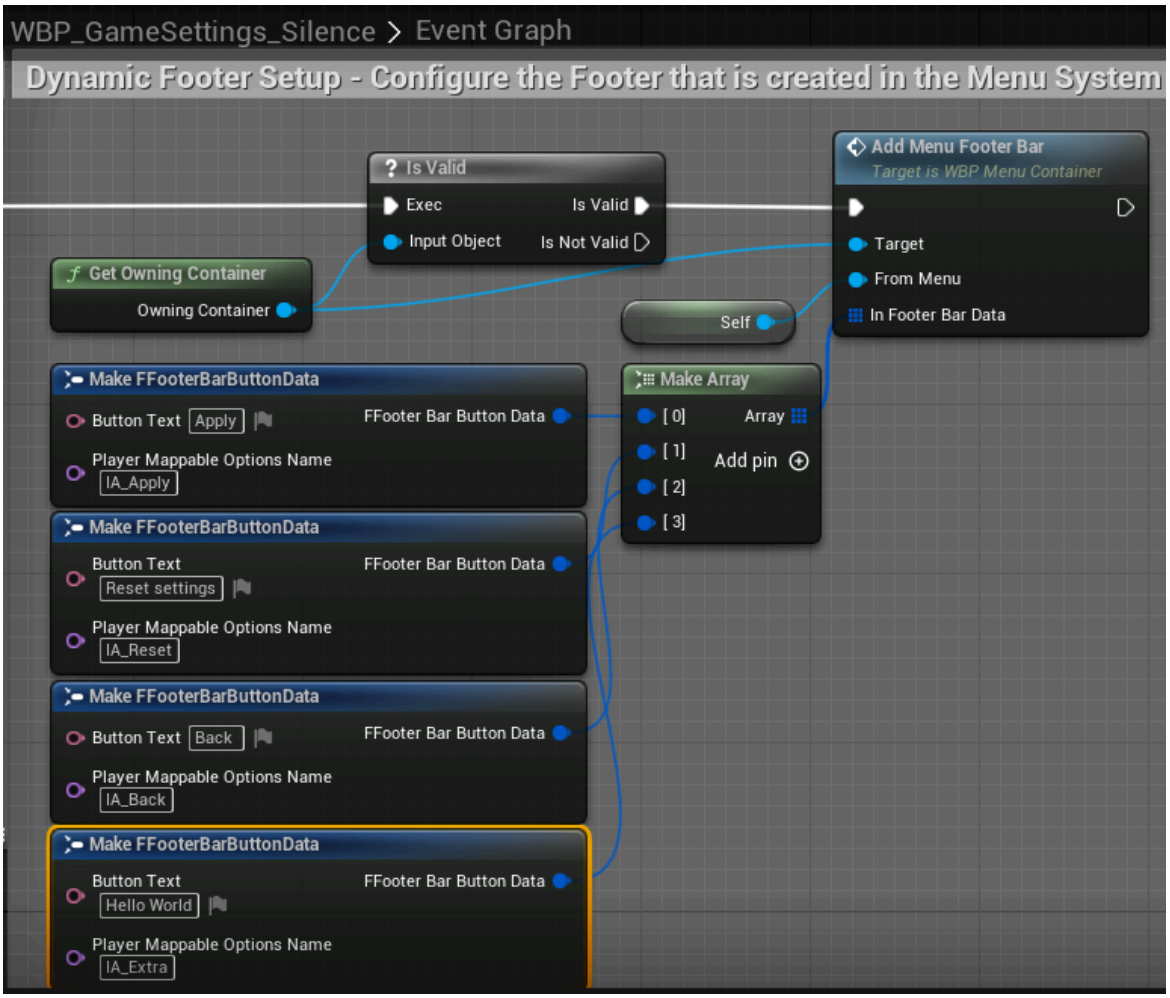
12. Done! You can easily access this data in your Blueprint code. Read the Access Settings in Blueprints section for more detailed information on that topic.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

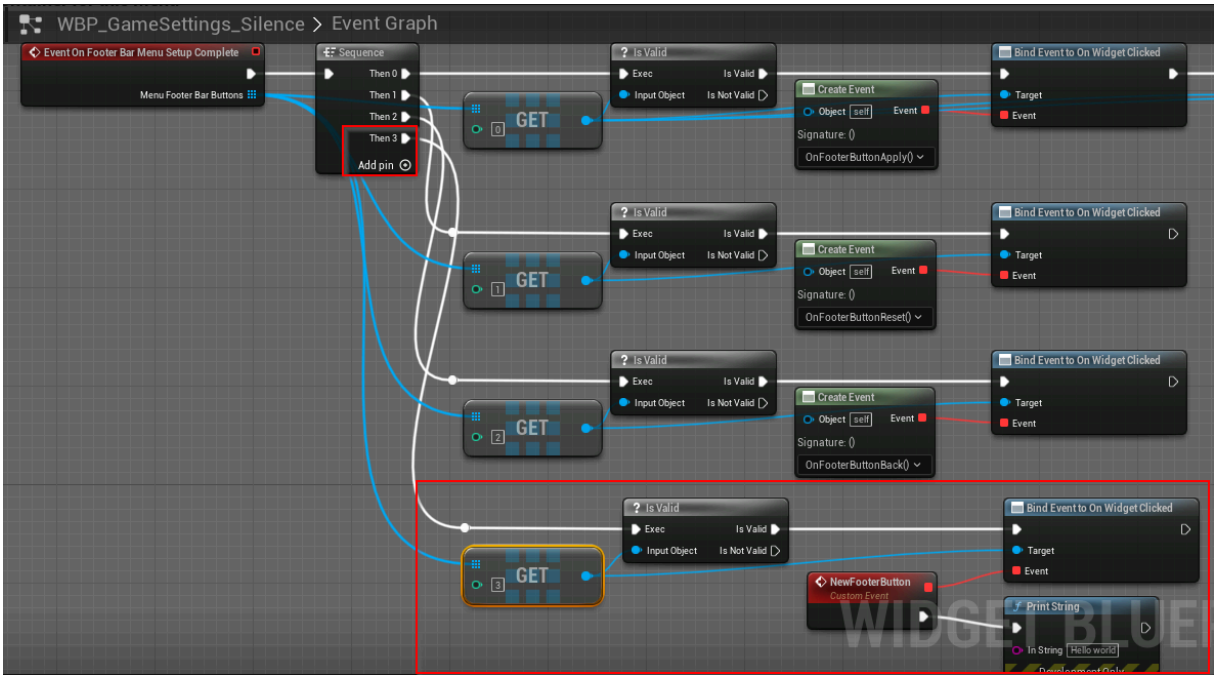
Modify existing menu

In this guide we will add a new footer button to the Game Settings menu

1. Navigate *to /ExampleContent/Designs/Design_Silence/Menus/Settings/WBP_GameSettings_Silence*
2. Locate the Dynamic Footer Setup on the Event Graph and add a new button like this:

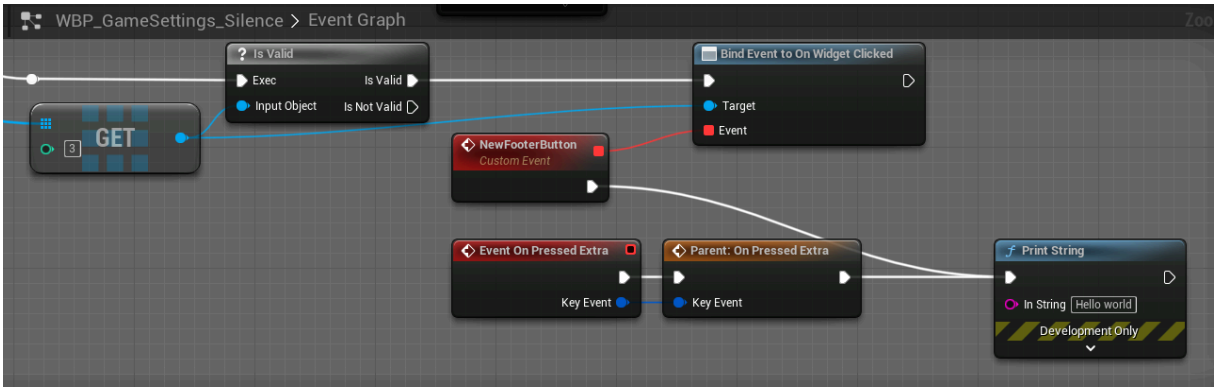


   Duplicate an existing one and connect it to the Make Array node. The Player Mappable Options Name comes from the Enhanced Input Mapping Context located at: */Blueprints/Input/Menu/IMC_Menu*
3. Move further to the right side on the graph and locate the *OnFooterBarMenuSetupComplete* event



Add a sequence pin and duplicate one of the existing paths. Then edit the index on the Get node according to the footer button you added in Step 2

4. Add your custom code to the copied logic



Note: The OnPressedExtra event was added to support button press ('X' key)

5. Hit Compile & Save

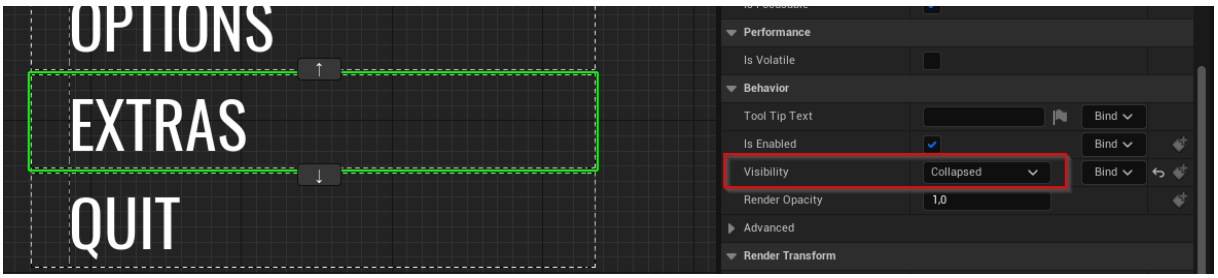
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Hide existing widget

You might want to remove one of the UI elements, such as Buttons or Options Picker from the menu. To save yourself some trouble you can just hide it, instead of deleting it.

Example #1 - Remove the Extras Menu from the Title Screen Menu

1. Navigate to /ExampleContent/Designs/Design_Silence/Menus/TitleScreen/
2. Open WBP_TitleScreenMenu_Silence
3. Select the Extras Title Screen Button and set the Visibility to 'Collapsed'



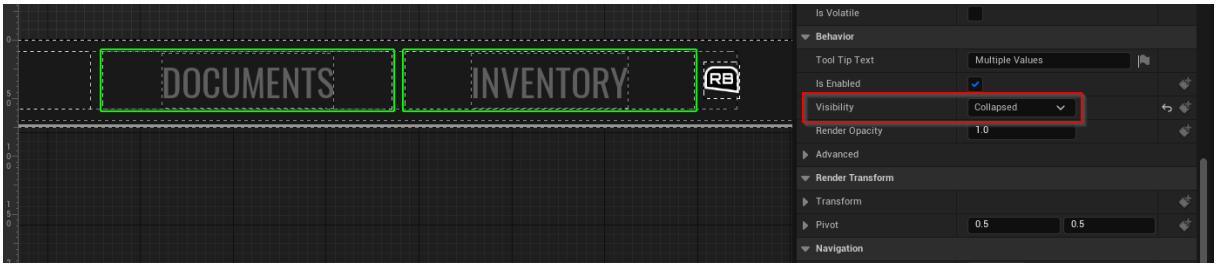
4. Hit Save & Compile
5. Enjoy!

Example #2 - Remove Documents and Inventory Tabs from Ingame Menu

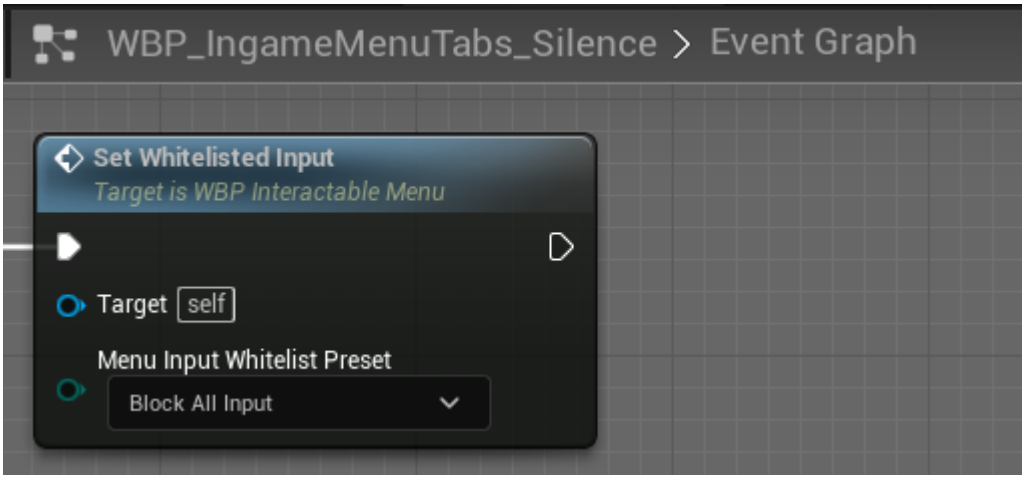
1. Navigate to /ExampleContent/Designs/Design_Silence/Menus/Ingame/
2. Open WBP_IngameMenuTabs_Silence

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

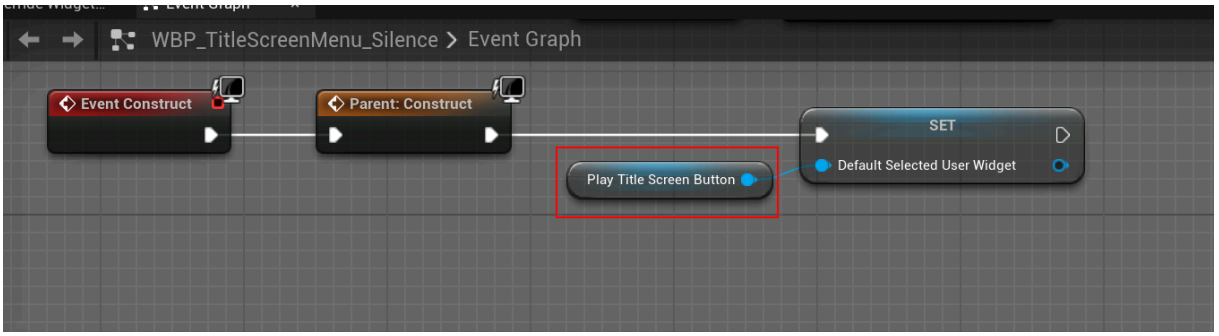
3. Select both, Documents and Inventory Large Tab Button and set the visibility to ‘Collapsed’



4. Hit Compile & Save
5. Enjoy!
6. Optional Step: If you wish to disable shoulder button interaction on the tabs you can edit the node on the Event Graph at the end of Pre Construct. Set the SetWhitelistInput node to ‘Block All Input’



Note: Remember to remove any references in code if you want to hide a widget (e.g. we often reference the first button of a menu in Event Construct of a menu)



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

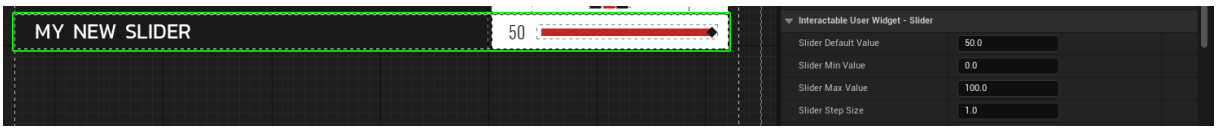
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Customize widgets

In this section you will learn how to customize Interactable User Widgets

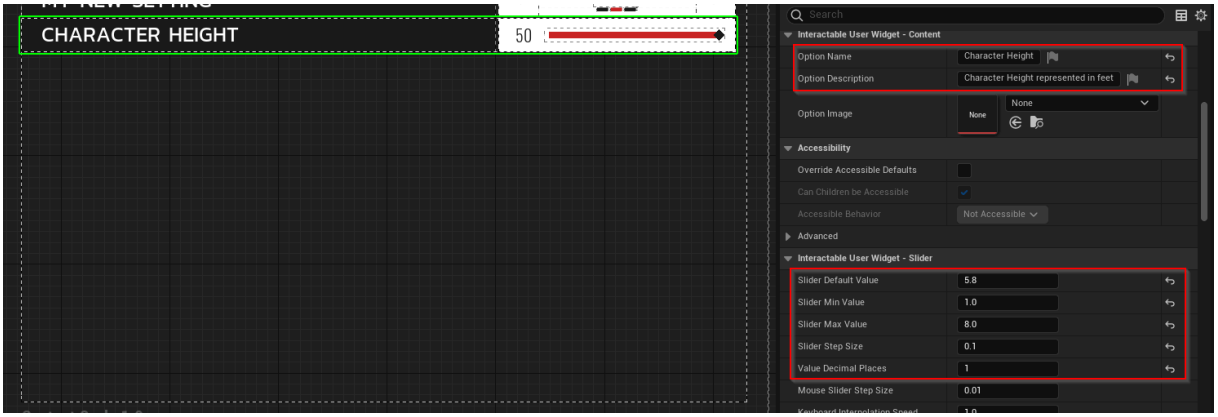
Example #1 - Change Slider Properties

1. Navigate to the menus folder located at /ExampleContent/Designs/Design_Silence/Menus/
2. Open any menu with a slider you would like to customize
3. Click on a Slider and have a look at the section 'Interactable User Widget - Slider'



4. This slider is set to display a range from 0 to 100 per default

5. We will now change it to represent character height measured in feet



Range from 1.0 to 8.0 with default set to 5.8 feet

6. This is how it looks during runtime

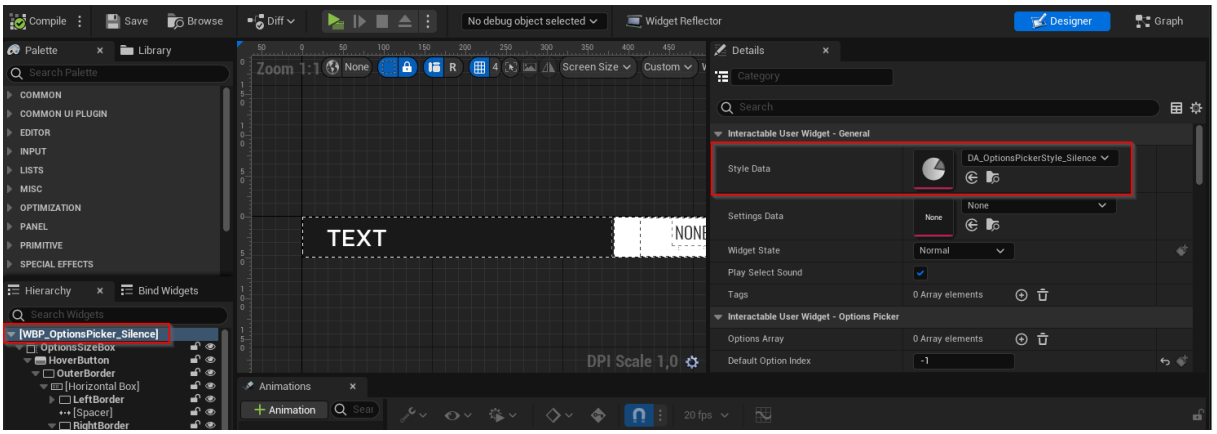
7. Enjoy!
8. Note: To make it properly load and save please follow the Guide: Add widget to menu

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

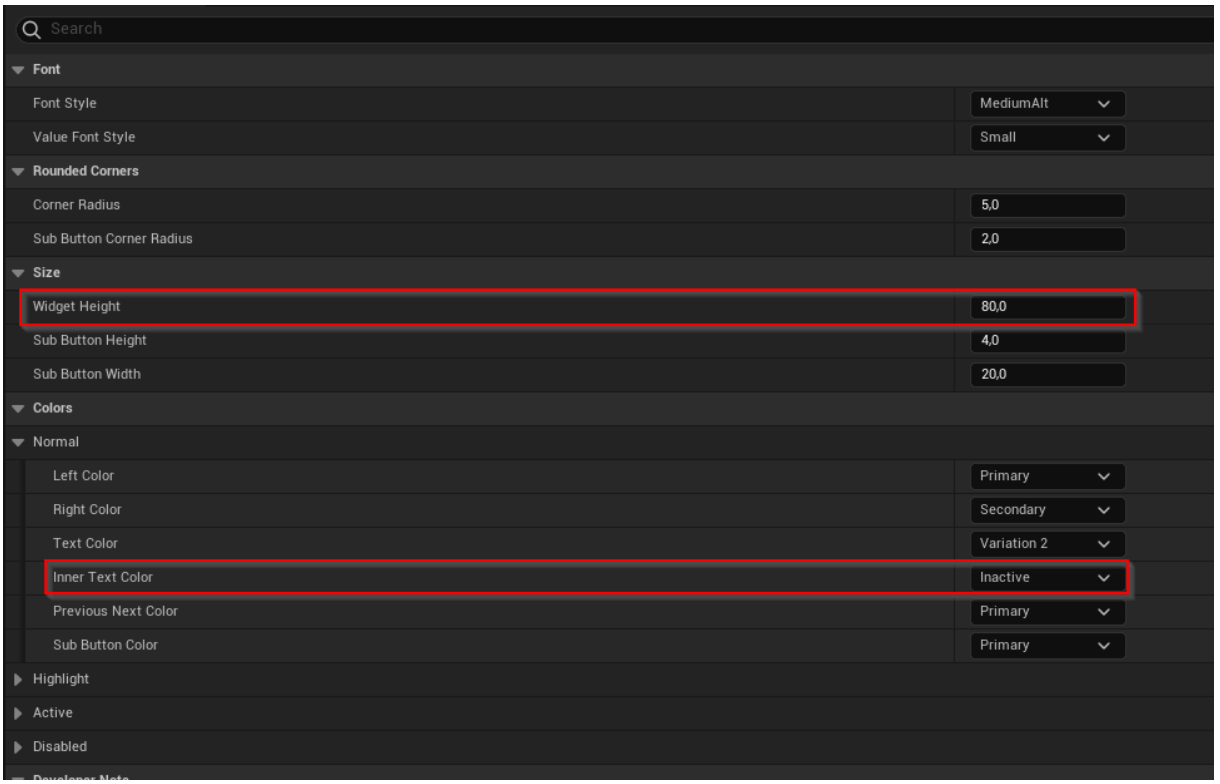
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Example #2 - Change Widget Style

1. Navigate to the widgets folder located at /ExampleContent/Designs/Design_Silence/Widgets/
2. Open any widget you would like to customize. For this example we choose the Options Picker
3. Open it's style data by clicking on the root object in the widgets hierarchy



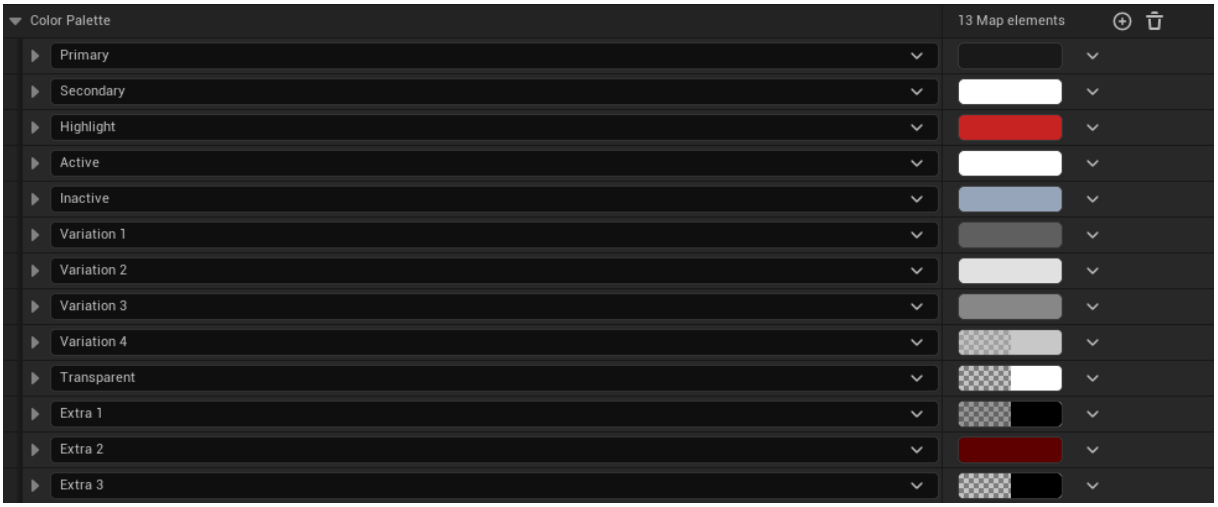
4. We changed the widget height and color for the inner text



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

5. Now the Options Picker looks like this

Note: The colors selectable in the Style Data Asset correspond to the ones set int the Menu System Config's Color Palette

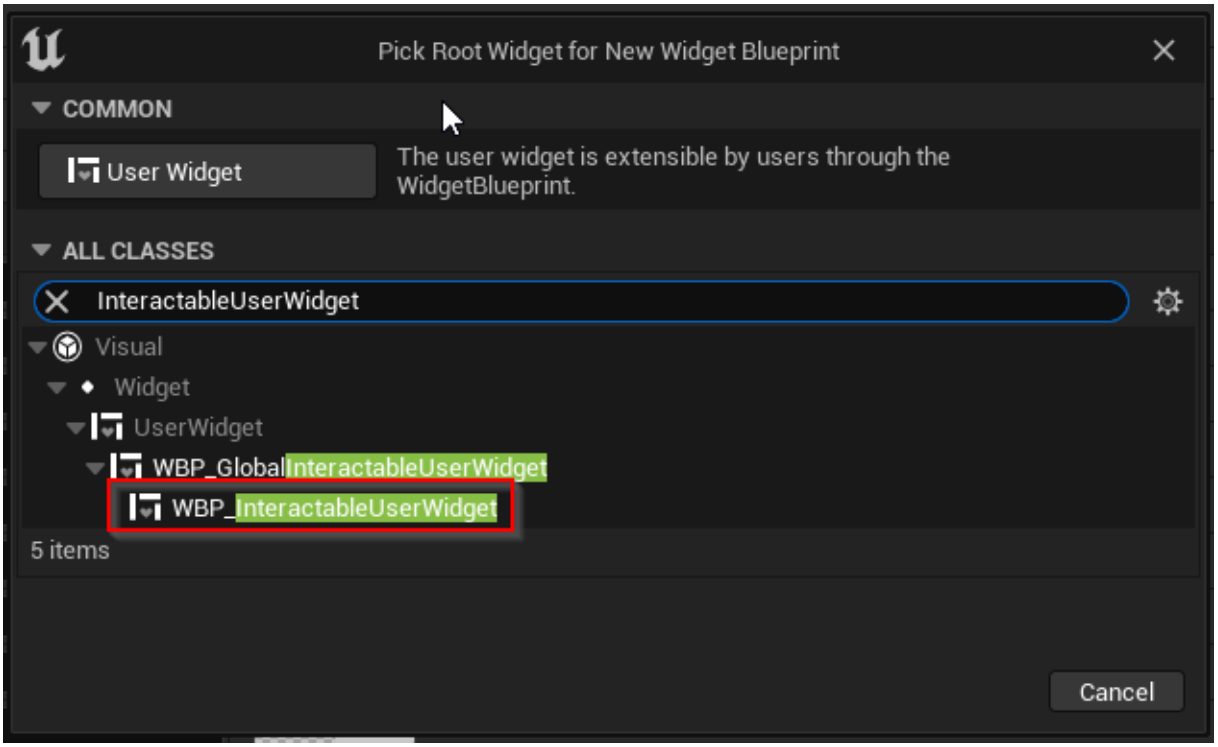


Refer to the section Widget Style Data for more detailed information.

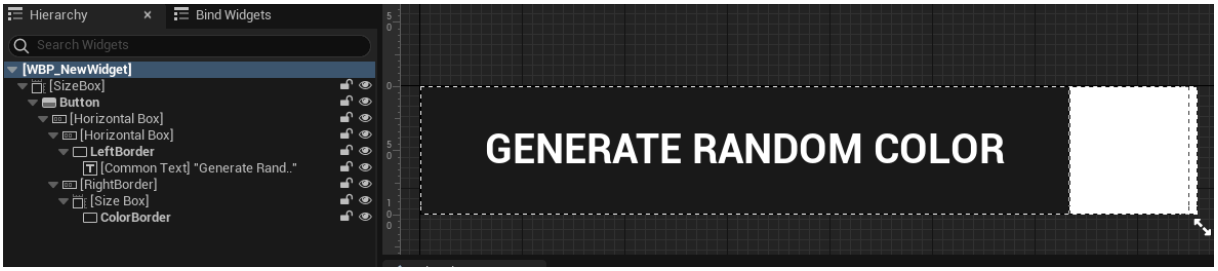
Create your own widget

If you need to create a completely new widget and integrate it into the menu system, you can refer to this guide for detailed instructions on how to do so.

1. Create a widget inherited from WBP Interactable User Widget.



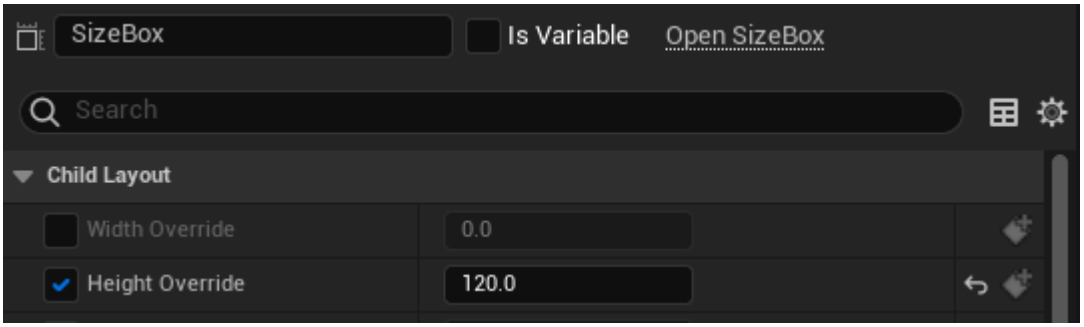
2. Add widgets to the hierarchy and wrap them with a size box.



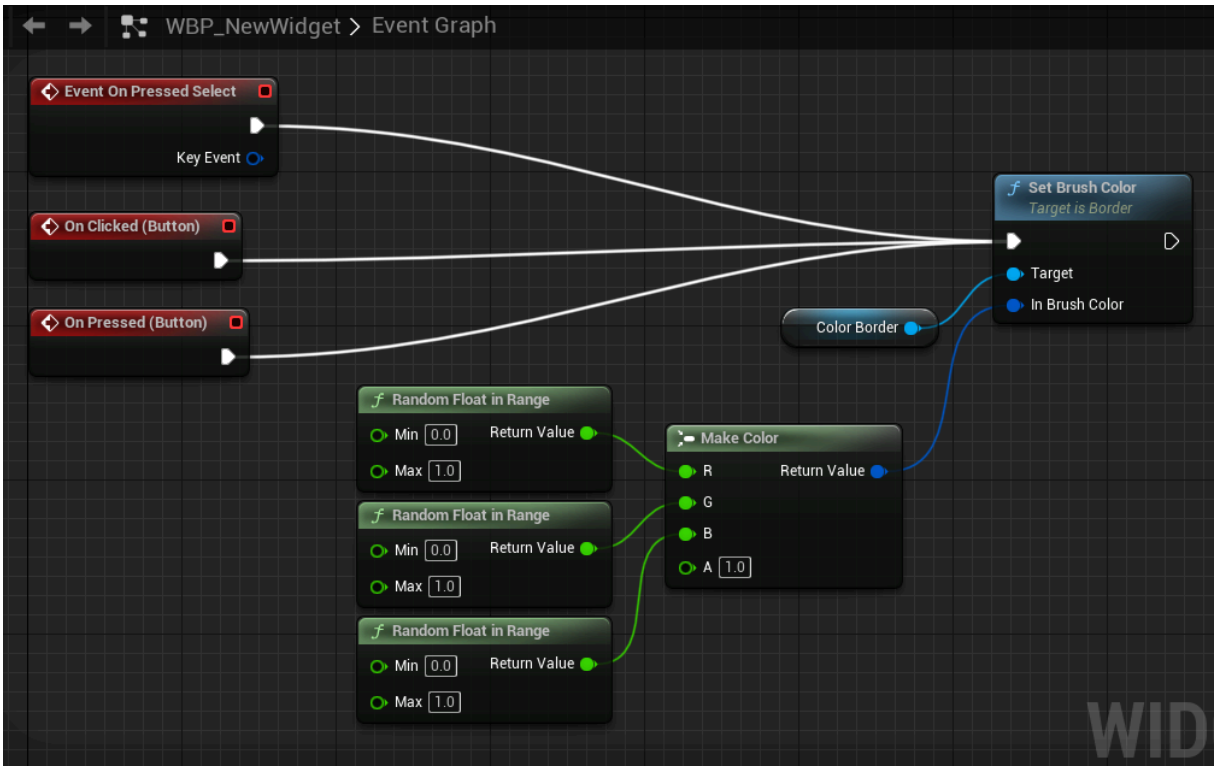
This widget represents a button that changes the color on the right side every time it is pressed.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

a. On the size box you can control the widgets height in the menu.



3. Add the buttons clicked and pressed event to the graph.



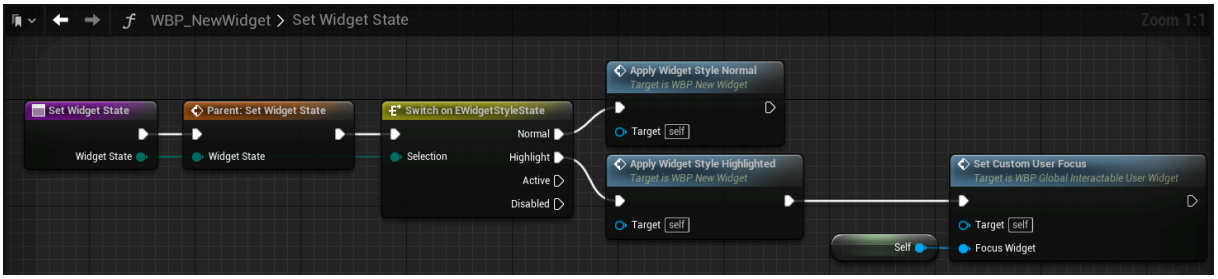
Also add the OnPressedSelect event inherited from the InteractableUserWidget class, which is called when the widget is pressed by keyboard or gamepad. The code above just generates a random color and sets it to the Border on the right side.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

4. Add those three function overrides to make the button change appearance based on the widget state (Normal and Highlighted).

   a. Those are all the available Style functions

5. Open the SetWidgetState function and add a simple logic which calls the Style Apply functions like this.

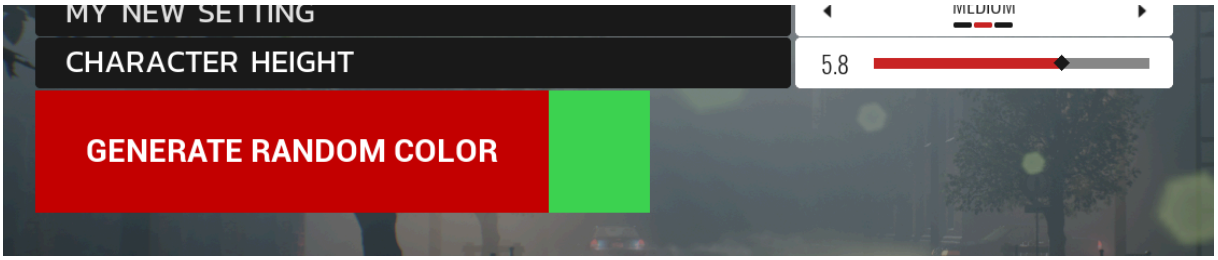


   Note: It is important to call the node SetCustomUserFocus on Highlight to set the user focus to the widget.

6. In the ApplyWidgetStyleNormal and ApplyWidgetStyleHighlighted we just set the Color of the LeftBorder to grey (normal) or red (highligted).

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

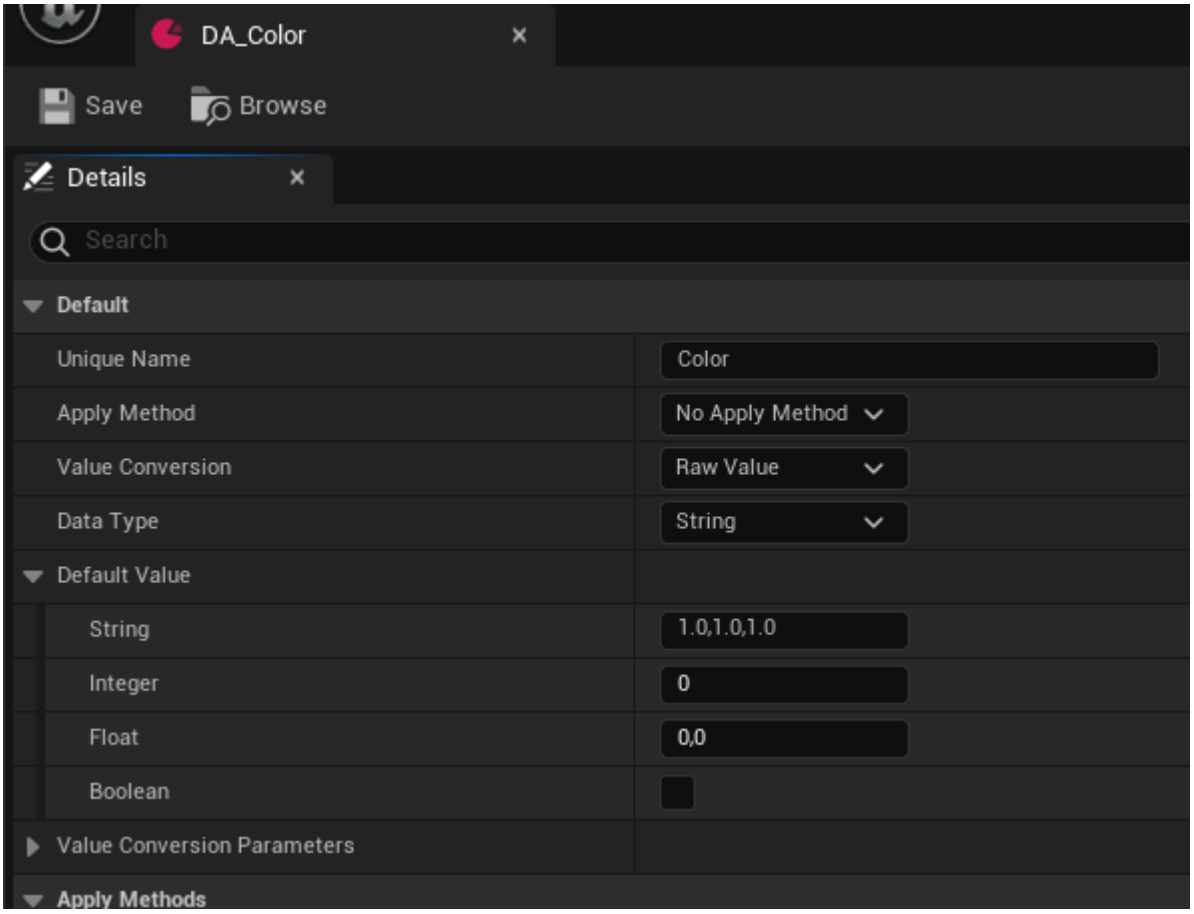
7. You should be able to interact and highlight the widget now.



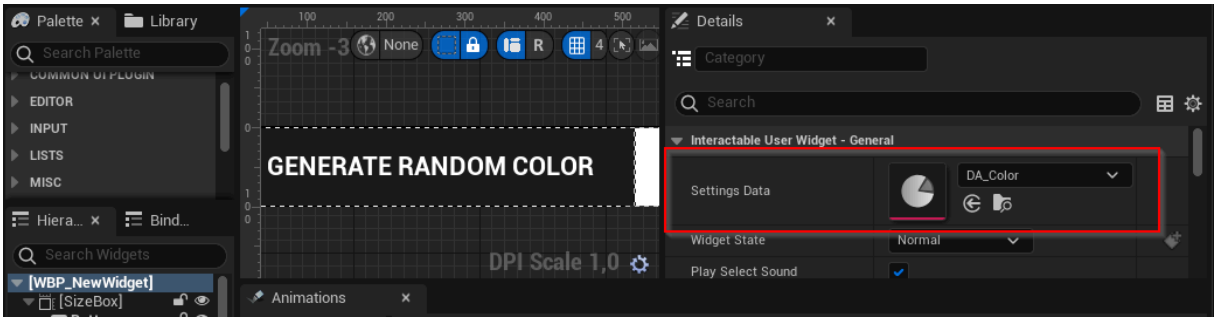
And it switches the LeftBorder Color based on its state.

8. Now we would like to save that color so it's loaded the next time we open up the menu. For that create a new Settings Data asset like described in Add

widget to menu.

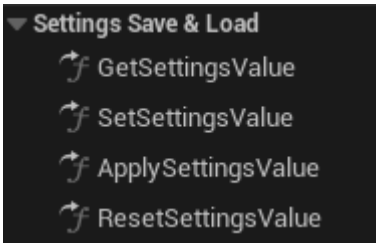


9. Go back to your widget and set it by clicking on the root on the hierarchy

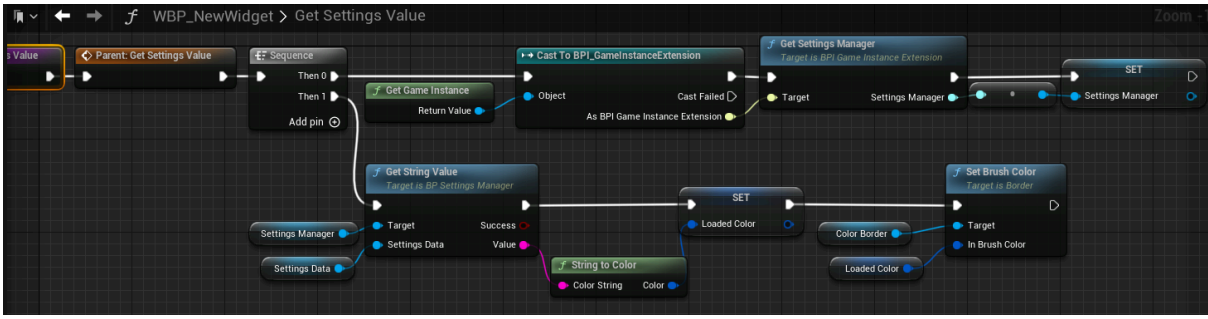


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

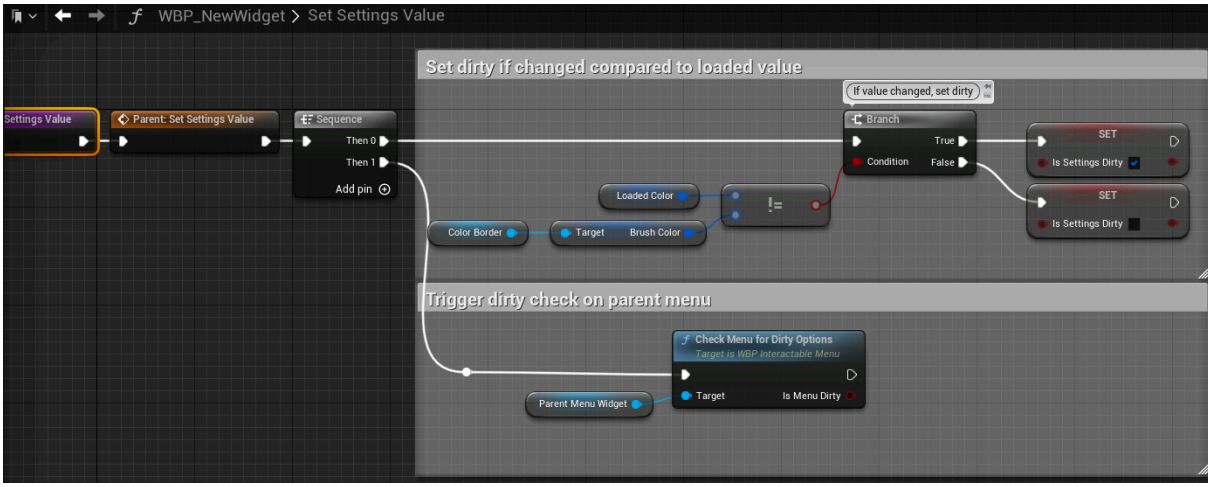
10. Implement the Save & Load functions through override like we learned before.



a. GetSettingsValue: Read Color from Settings Manager and set it to the Border

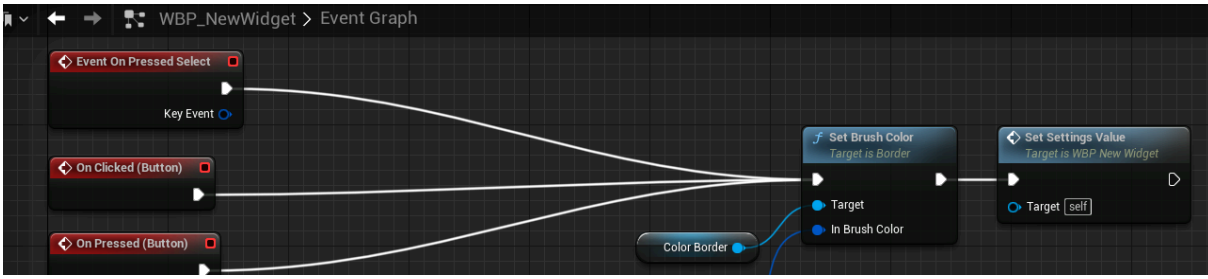


b. SetSettingsValue: Set the setting dirty / enable apply in menu

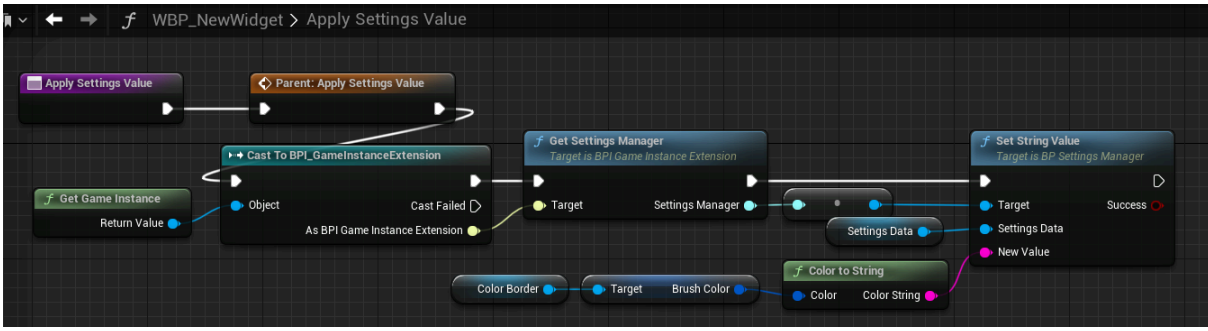


Call it after your widget has changed like this

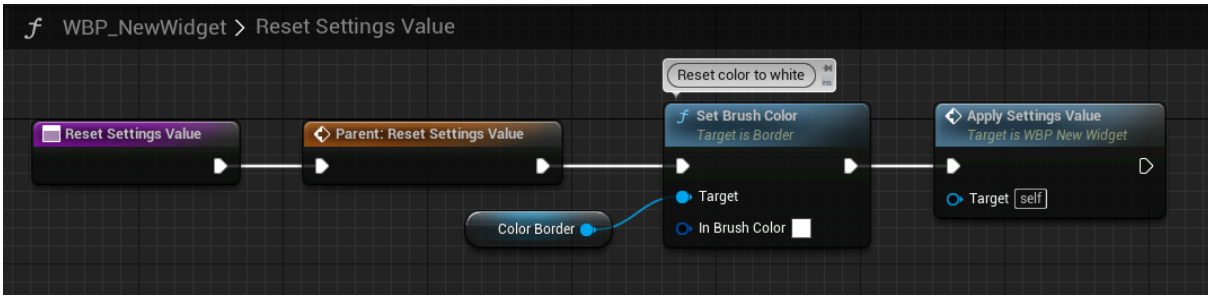
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)



c. ApplySettingsValue: Save Color to Json through Settings Manager



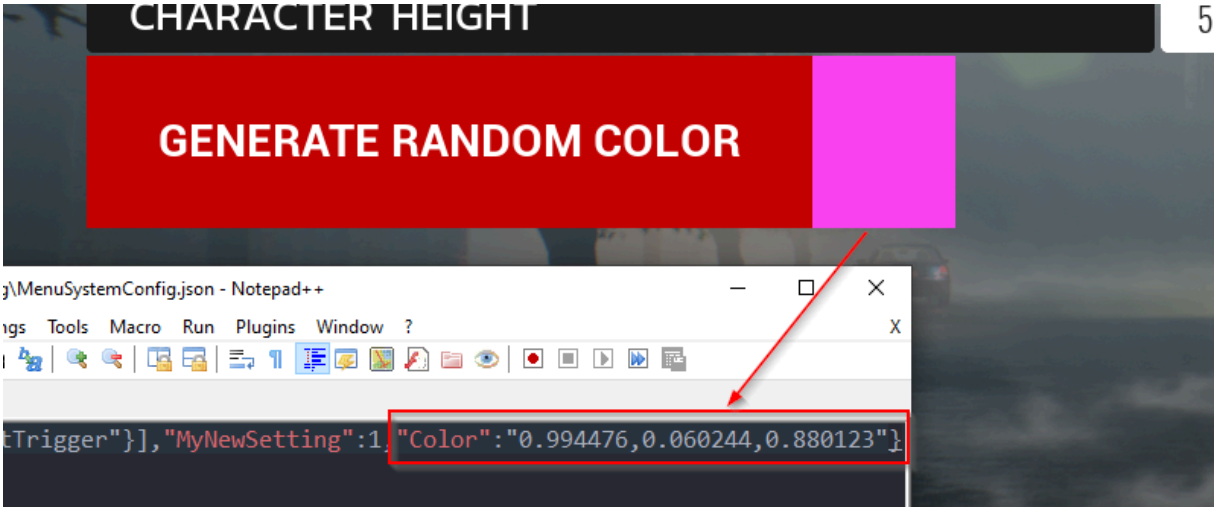
d. ResetSettingsValue: Reset to default color (white) and apply



Learn more about Save and Load Settings

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

11. Now your setting show save to the Menu System Config Json file



12. The Apply Button also turns on when the color is changed

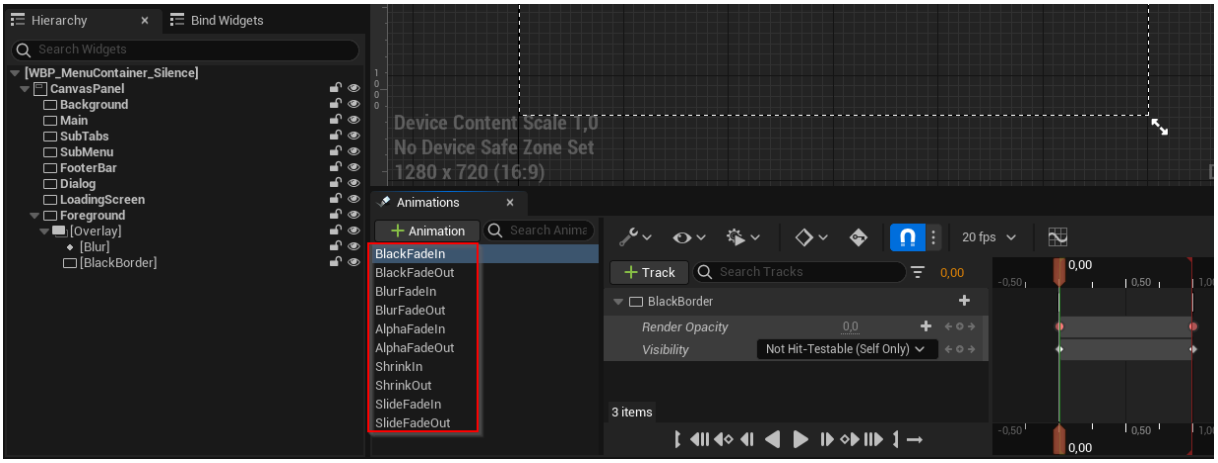
13. Congratulations! Your widget is now ready to be used.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Modify Menu Transitions

In the Menu System Config you can choose from five pre-built menu transitions. In this section you will learn how to change existing and add new menu transitions.

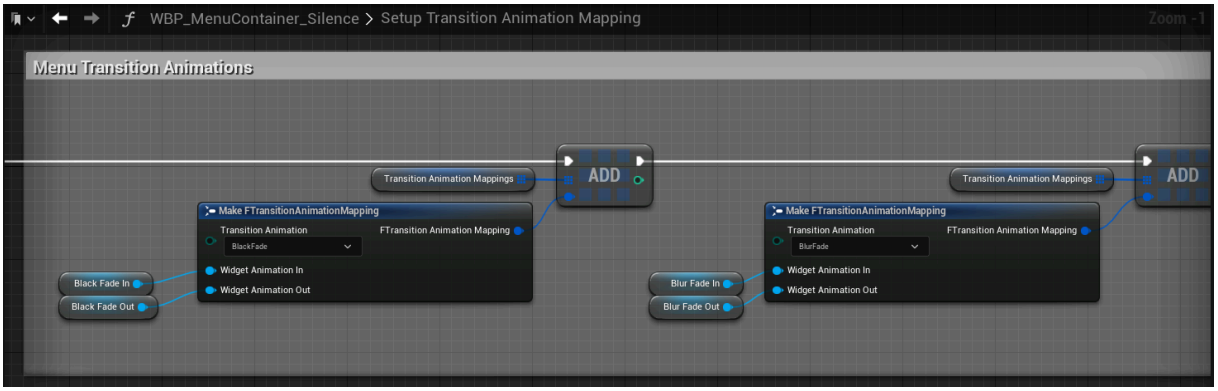
1. Navigate to /ExampleContent/Designs/Design_Silence/Menus/Base/
2. Open WBP_MenuContainer_Silence.
3. In the Designer tab within the animation window you will see the animations we provide and how they are done.



You can change them here (modify keyframes, timings or add other elements etc.)

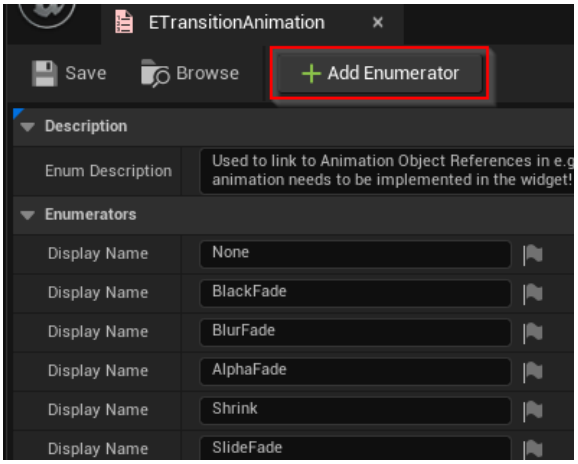
4. Then go to the Graph tab and open the Function 'SetupTransitionAnimationMapping' .

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)



In this function the animations you saw in step 3 are added to Transition Animation Mappings, which are called when transitioning menus.

5. To create a new transition you first have to extend the Enumeration asset.
    a. Navigate to */Blueprints/UI/Types/* and open *ETransitionAnimation* to add a new enum entry.

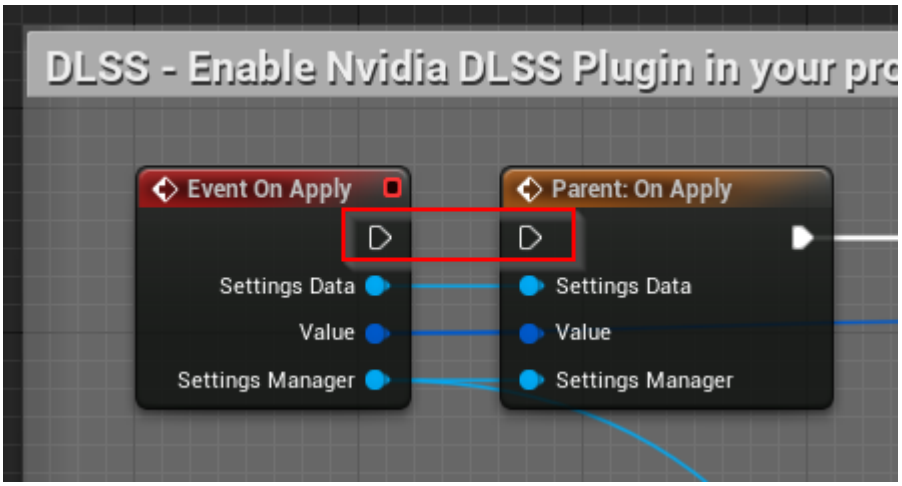


6. After that just create one in and out animations in the Designer tab and add them to the array in the function mentioned above.
7. Now you should be able to select it in the Menu System Config and it should be called when transitioning menus.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

How to use DLSS and FSR2

Download and enable both plugins (linked below). Reconnect both DLSS nodes in the Custom Apply Class:
'/Blueprints/Settings/CustomSettingsApplyClasses/BP_DLSSApply'



Please refer to the official AMD and Nvidia documentations regarding the setup and functionality.

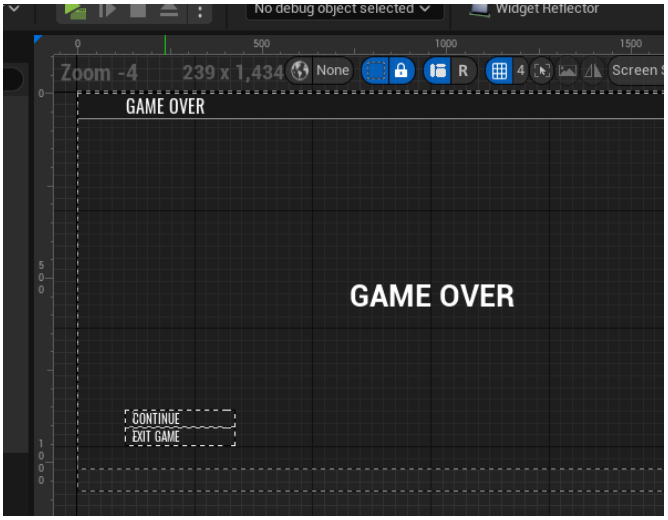
AMD Fidelity Super Resolution 2:
https://www.unrealengine.com/marketplace/en-US/product/fidelityfx-super-resolution-2

Nvidia DLSS:
https://www.unrealengine.com/marketplace/en-US/product/nvidia-dlss

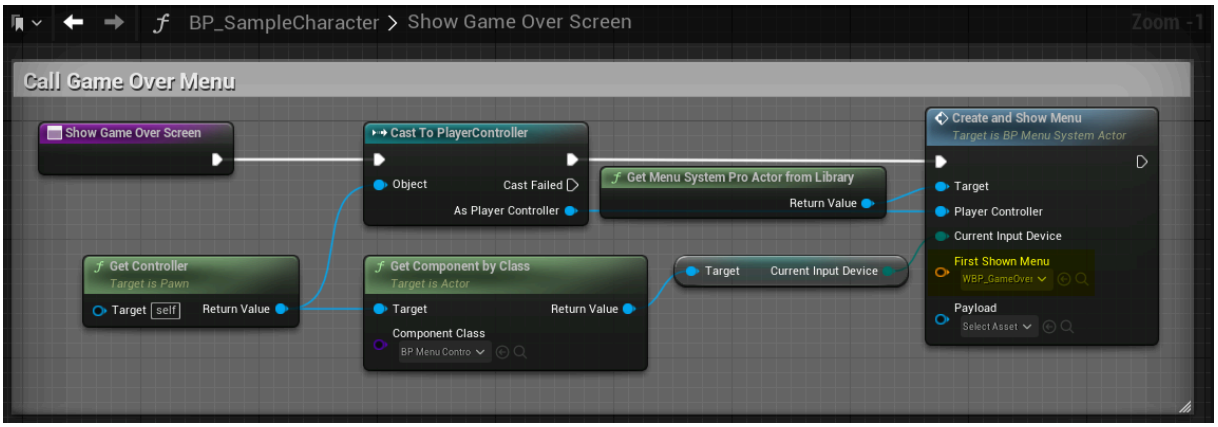
Show Game Over Screen

In this section you will learn how to call a Game Over Screen from your Character Blueprint.

1. Create a menu that will be called for the Game Over Screen. See Create new menu for a menu creation guide.



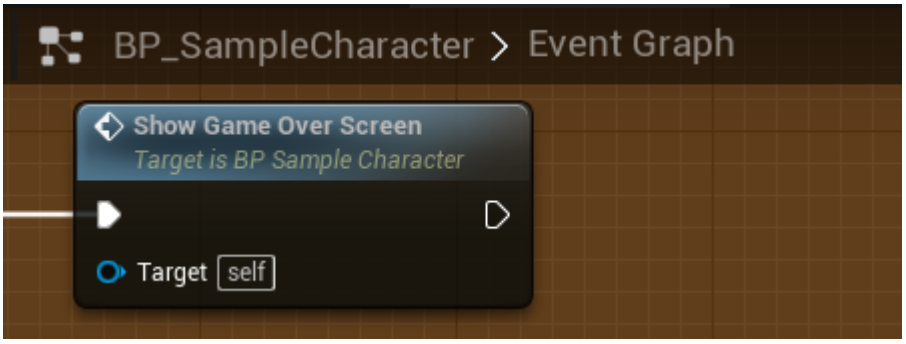
2. Open your Character blueprint.
3. Create a function with the code below to call the menu.



Select your Game Over menu at the marked pin. Select 'BP_MenuControllerCompont' on the GetComponentByClass node.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

4. Call this code anywhere in your Character blueprint based on your game over logic.



5. Done! Your menu should appear when calling the function.

Menus on 3D Widgets

You can also use Menu System Pro on 3D Widgets in your scene. We captured a quick video of an example we have made on request by one of our discord members: https://www.youtube.com/watch?v=MURuwDM7R58

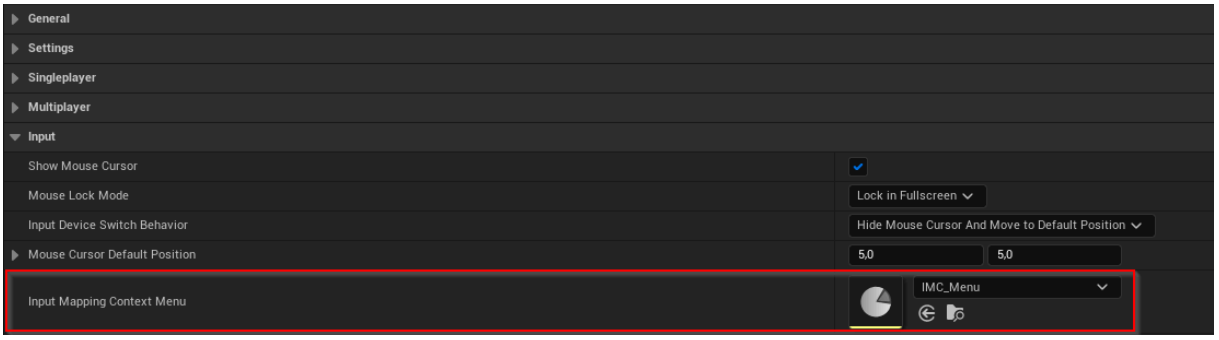
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

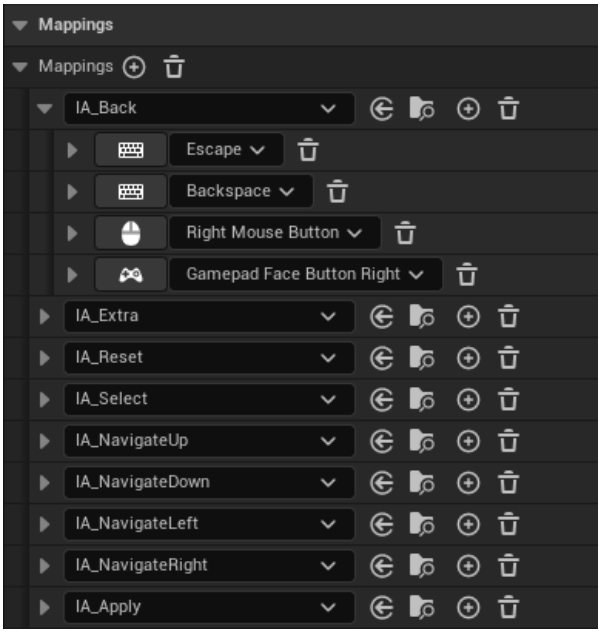
Enhanced Input Integration

We integrated the official Enhanced Input System into Menu System Pro. Please refer to the Unreal Engine Documentation for general information on it: https://docs.unrealengine.com/5.1/en-US/enhanced-input-in-unreal-engine/

Menu Input

The menu input is also handled using Enhanced Input. It has its own designated Input Mapping Context and Input Actions. They can be seen and modified in the Menu System Config:





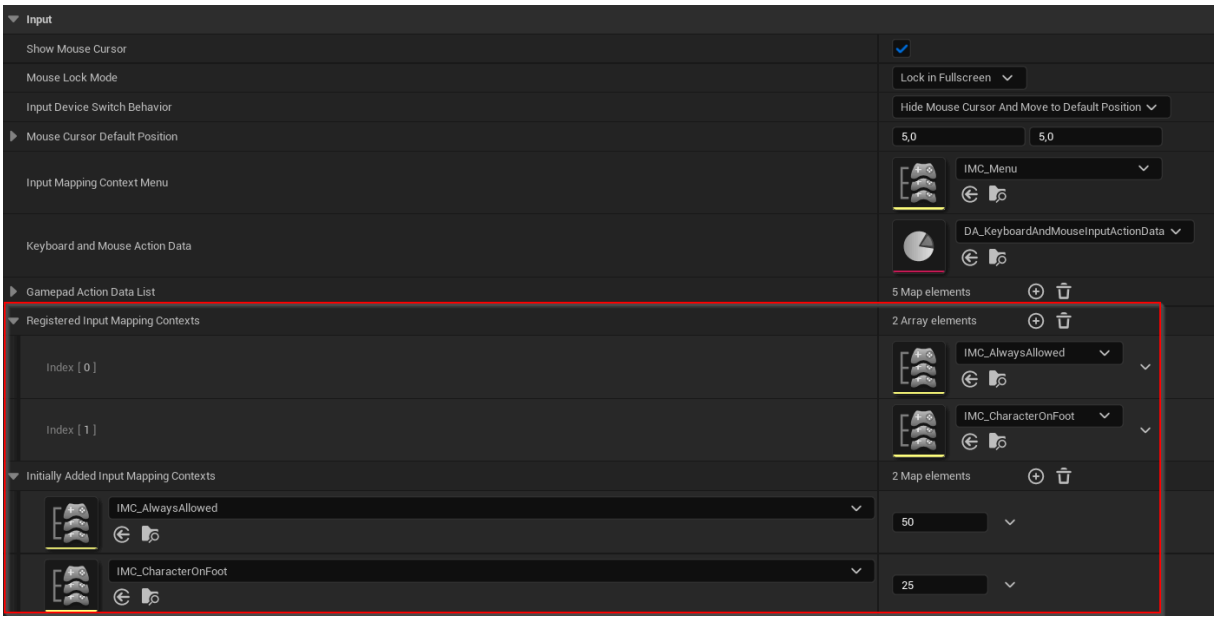
If you want to change or add actions/mappings it is advised to create an own Input Mapping Context outside of our folder structure and select it in the Menu System Config (for better updatability).

Character Input

The character input is split into two Input Mapping Contexts per default (AlwaysAllowed and CharacterOnFoot).

UE 5.3 and above

The Menu System Config contains an array for IMCs that need to be registered for user customizable bindings to make them for example compatible with the Key Binder. Generally all your IMCs should be added to this array. There is also a map for IMCs that get added to the Enhanced Input System through the Menu Controller Component on BeginPlay. This should contain all IMCs that have bindings that must work at the start of the game (e.g. IA_Pause or IA_Move).

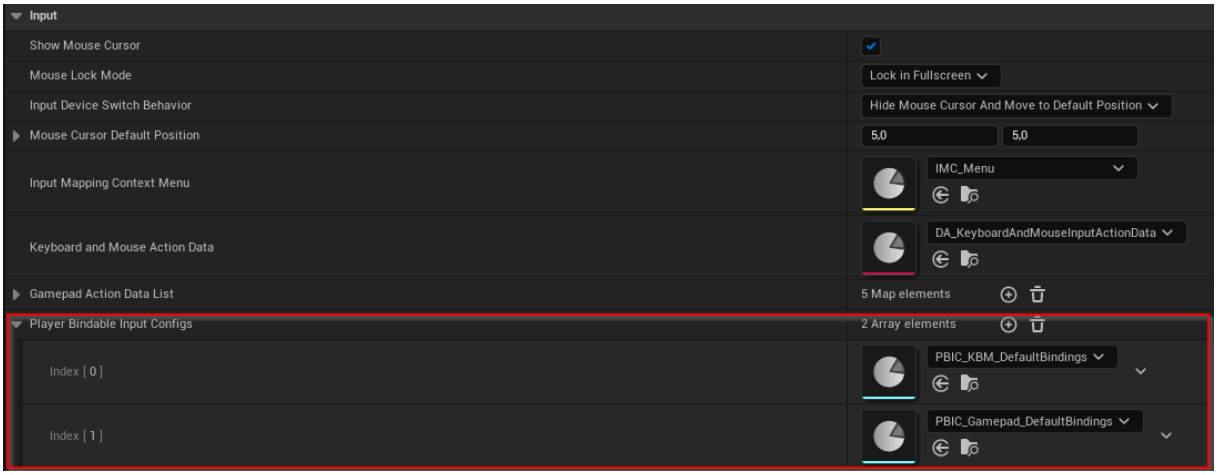


Note: The integer value in the InitiallyAddedInputMappingContexts represents the priority. Higher values have priority.

IF YOUR CHARACTER CAN NOT MOVE you very likely need to change IMC_CharacterOnFoot with the IMC your Character uses (e.g. IMC_Default) in BOTH LISTS (Registered Input Mapping Contexts & Initially Added Input Mapping Contexts) in DA_MenuSystemConfig_Silence (NOT THE PDA_ file!).
Use 'showdebug enhancedinput' in the console to debug why your character does not accept inputs!

UE 5.2 and below

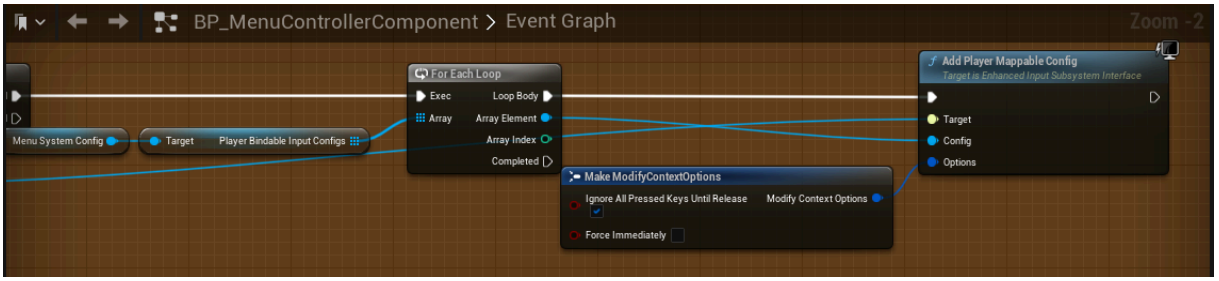
They are hooked up through a Player Bindable Input Config in the Menu System Config's Input section:



If you want to add your own Player Bindable Input Config or Input Mapping Contexts this is the right place to do so.

It contains one PBIC for KBM (Keyboard/Mouse) with its two IMCs and one PBIC for Gamepad with also two IMCs.

Those are added automatically to your Player Controller using the BP_MenuControllerComponent during runtime (Event Graph → Begin Play).



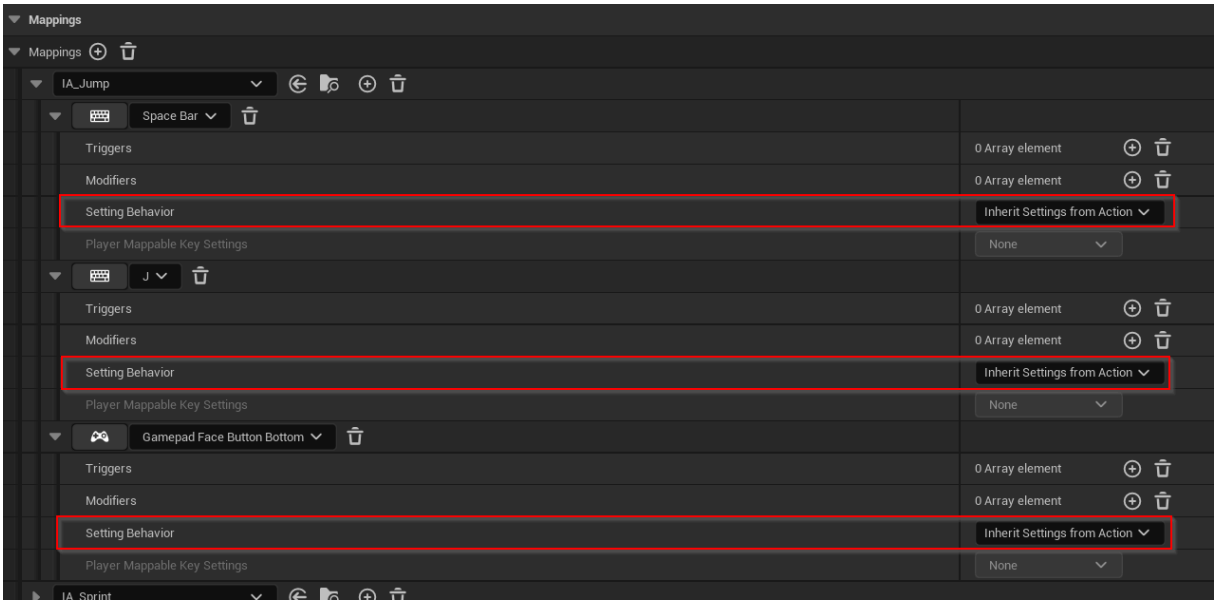
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Setup Input Mapping Context

In order to work like intended Menu System Pro needs a specific setup of bindings in the Input Mapping Context (IMC).

5.3 and above

Input Mapping Context:



Input Action:

Set the Setting Behavior on every binding to 'Inherit Settings from Action' in all of your IMCs. Then make sure to set a unique name and display name in every Input Action (e.g. IA_Jump -> Jump). This will be important for integrating the bindings in the key bindings menu later on.

Also important:

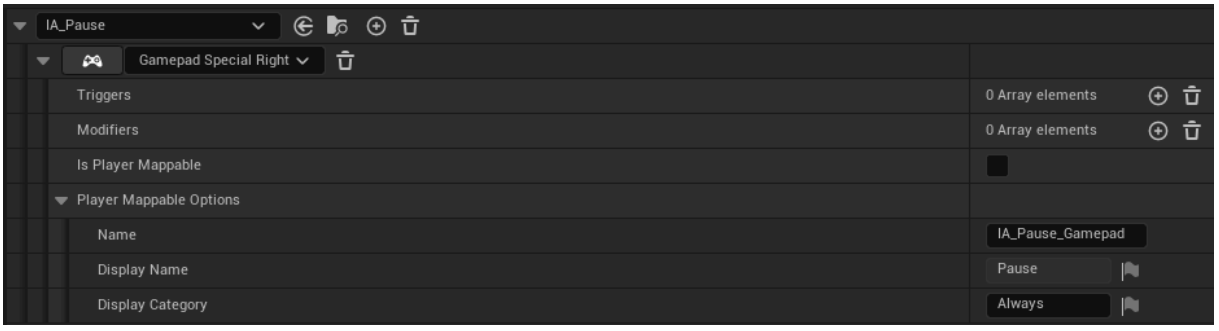
- Input Mapping Contexts should not contain multiple bindings that point to the same key/button
- Input Actions should not be re-used in multiple different Input Mapping Contexts

5.2 and below

Make sure to have this set on every Input Action binding

1. Is Player Mappable = True
2. Player Mappable Options Name = Unique Name for this binding (with suffix)
3. Player Mappable Options Display Name = Enter a user friendly name
4. You can set IsPlayerMappable to False to make the Key Binding read-only in the Key Binder widget
5. If you have more than one binding add a different suffix to tell them apart

This is the counterpart for the Gamepad:

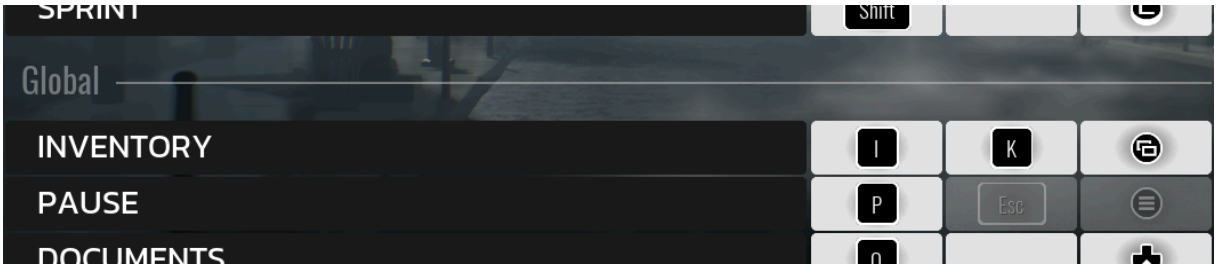


It is also set to read-only and has the unique suffix '*_Gamepad*'.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

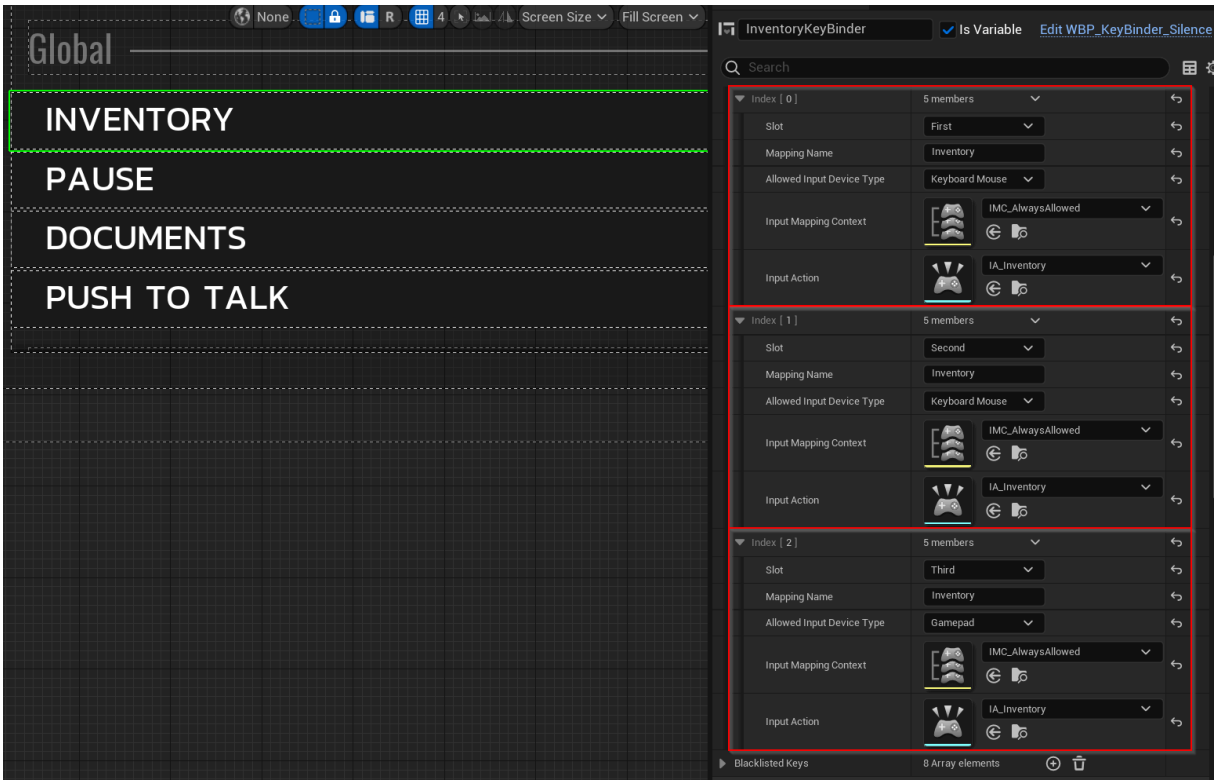
Key Bindings Menu

The Enhanced Input System is also integrated in our flexible Controls Key Bindings Menu and the Key Binder widget.



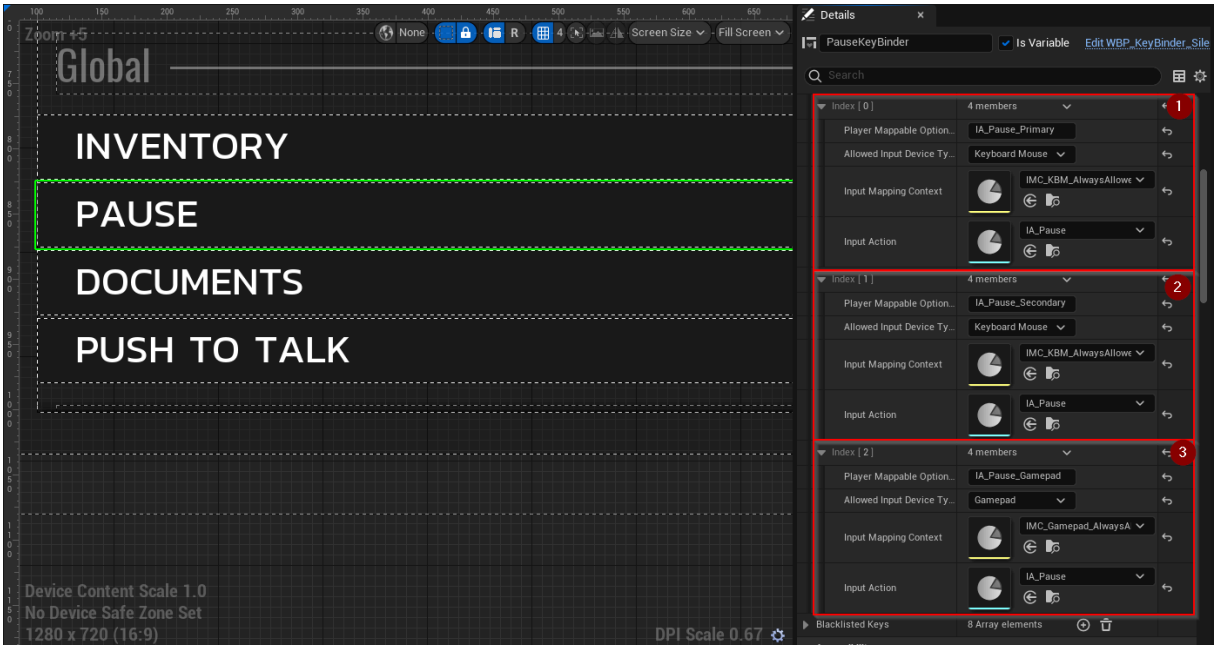
The Key Binder is modular and can be provided with the parameters you learned about before in the previous subchapter.

5.3 and above



It is fully modular so you are able to remove existing or add more elements to represent alternative bindings. In our example we used three bindings for each action (Primary, Secondary, Gamepad). Remember to use the unique names you have learned about before as the Mapping Name. The other parameters should be pretty self-explanatory.

5.2 and below

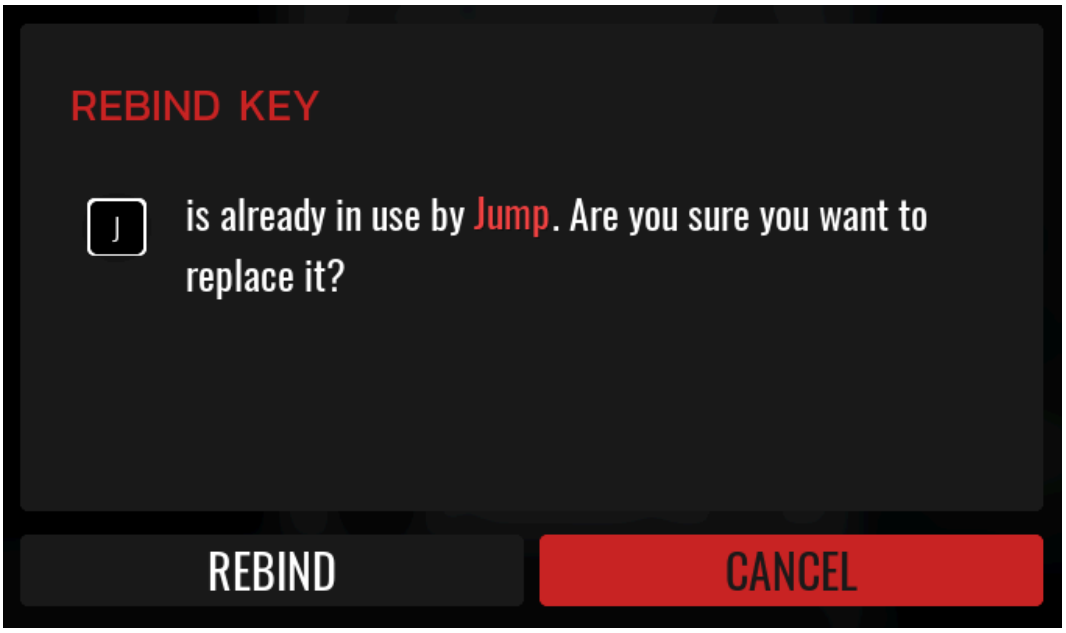


It is fully modular so you are able to remove existing or add more elements to represent alternative bindings. In our example we used three bindings for each action (Primary, Secondary, Gamepad). Remember to use the unique names you have learned about before. The other parameters should be pretty self-explanatory.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Rebinding Inputs

The rebinding process checks for duplicate bindings within the same Input Mapping Context. When a duplicate is detected it will prompt you with a Decision Dialog to override the existing binding. Here is an example:



Note; gamepad specific: Since UE5.3 it will not remove the binding on the found duplicate, but swap it with the current one instead, due to limitations introduced by EPIC.

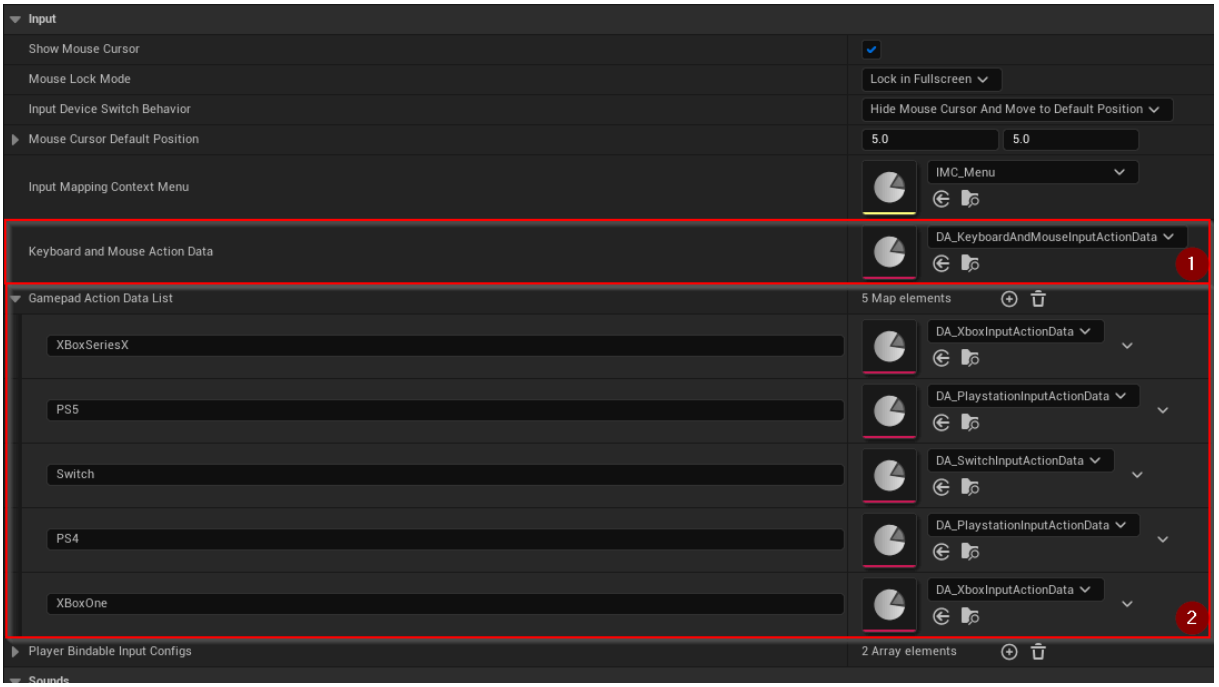
On the other hand bindings in different contexts can have the same button mapped to some action. This is useful to have the same button work differently when for example driving a car.

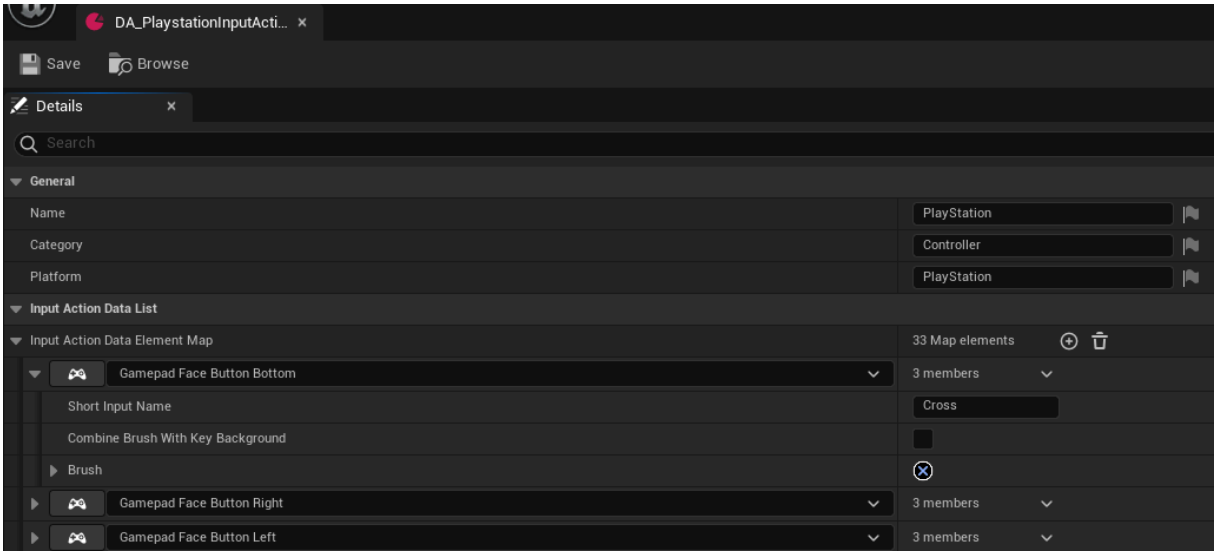
Note: If you modify bindings in the Input Mapping Context, make sure to delete the MenuSystemConfig.json so that the bindings are rebuilt correctly.

Input Icons

The Input Icon is utilized in multiple locations throughout the menu system like for example in the Footer Bar Buttons or the Key Binder widget.

It can display KBM (Keyboard/Mouse) and Gamepad keys/buttons. To map internal keys to actual button images we have set up Data Assets for KBM and common Gamepads:





THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Save and Load Settings

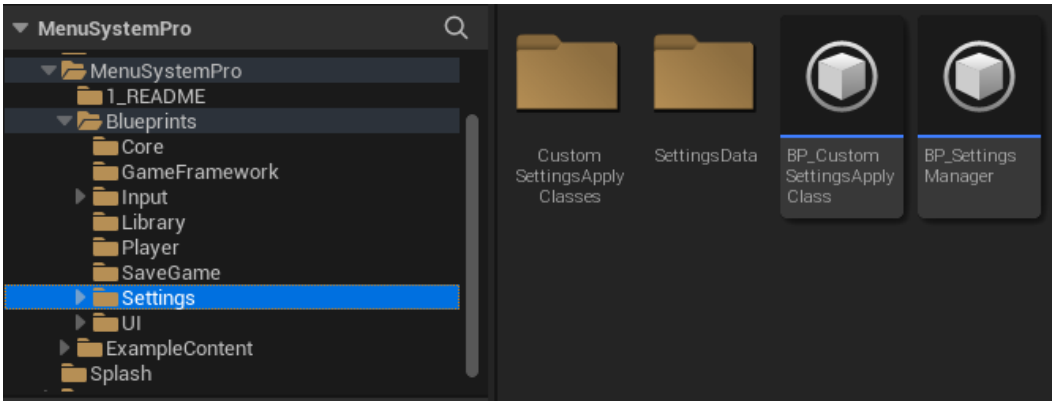
This chapter will cover the process of how Menu System Pro handles settings data and the role of the JSON Blueprint Utilities plugin in this process.

General

The menu system utilizes a Settings Manager that is linked to the Game Instance, enabling it to manage read and write operations to the JSON settings file. Read the Getting Started chapter for more information on the integration into your project.

By having the Settings Manager attached to the Game Instance, we gain the advantage of accessing settings data throughout the entire game and being independent of the user interface.

The Settings Manager Blueprint is located at: *'/Blueprints/Settings/'*



JSON Settings File

We decided to use JSON as the data format to store settings in our menu system. The following is a snippet of the JSON settings file:

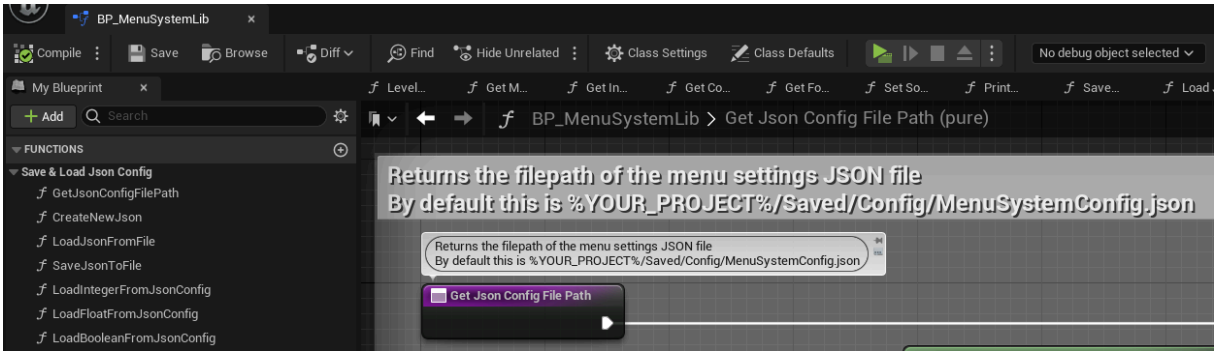
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

```
{
  "GameDifficulty": 3,
  "GameLanguage": "en",
  "EnableSubtitles": false,
  "PlayerName": "Player",
  "ShowPlayerNames": 1,
  "MasterVolume": 1,
  "MusicVolume": 1,
  "SFXVolume": 1,
  "VoiceVolume": 1,
  "EnableVoiceChat": false,
  "VoiceChatInputVolume": 1,
  "VoiceChatOutputVolume": 1,
  "WindowMode": 1,
  "DisplayResolution": "1920 x 1080",
  "DisplayBrightness": 2.1979759999999988,
  "FrameRateLimit": 30,
  "EnableVsync": false
```


It contains all regular settings as well as key bindings that differ from the default bindings within the Input Mapping Contexts. At the first start the Settings Manager will create this file with the default values specified in the Settings Data Assets.

Per default the JSON settings file is stored at this location:
Save Location in Editor: YourProject/Saved/*MenuSystemConfig.json*
Save Location Packaged Game:
C:/Users/YourUser/AppData/Local/YourProject/Saved/*MenuSystemConfig.json*

All file operations are managed by the JSON Blueprint Utilities plugin developed by EPIC. We added some functions to our BP_MenuSystemLib to handle those operations:



To change the file path you could edit the GetJSONConfigFilePath function in the mentioned Blueprint Library.

You can access data from the JSON settings file anywhere in your Blueprint code. Learn more about it in the 'Get saved settings in Blueprints' subchapter.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

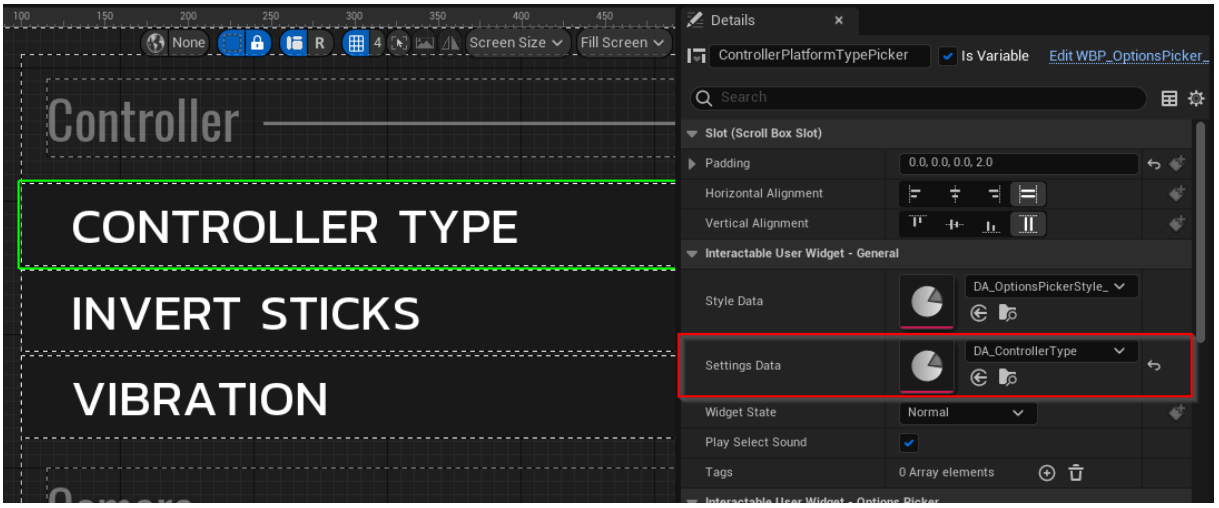
Settings Data Assets

We introduced Settings Data Assets which can be set in the Menu System Config and on Settings widgets throughout the menu system.

This is how it looks in the Menu System Config:

You will have to add all custom Settings Data Assets to make them recognizable by the Settings Manager, so it will load those settings.

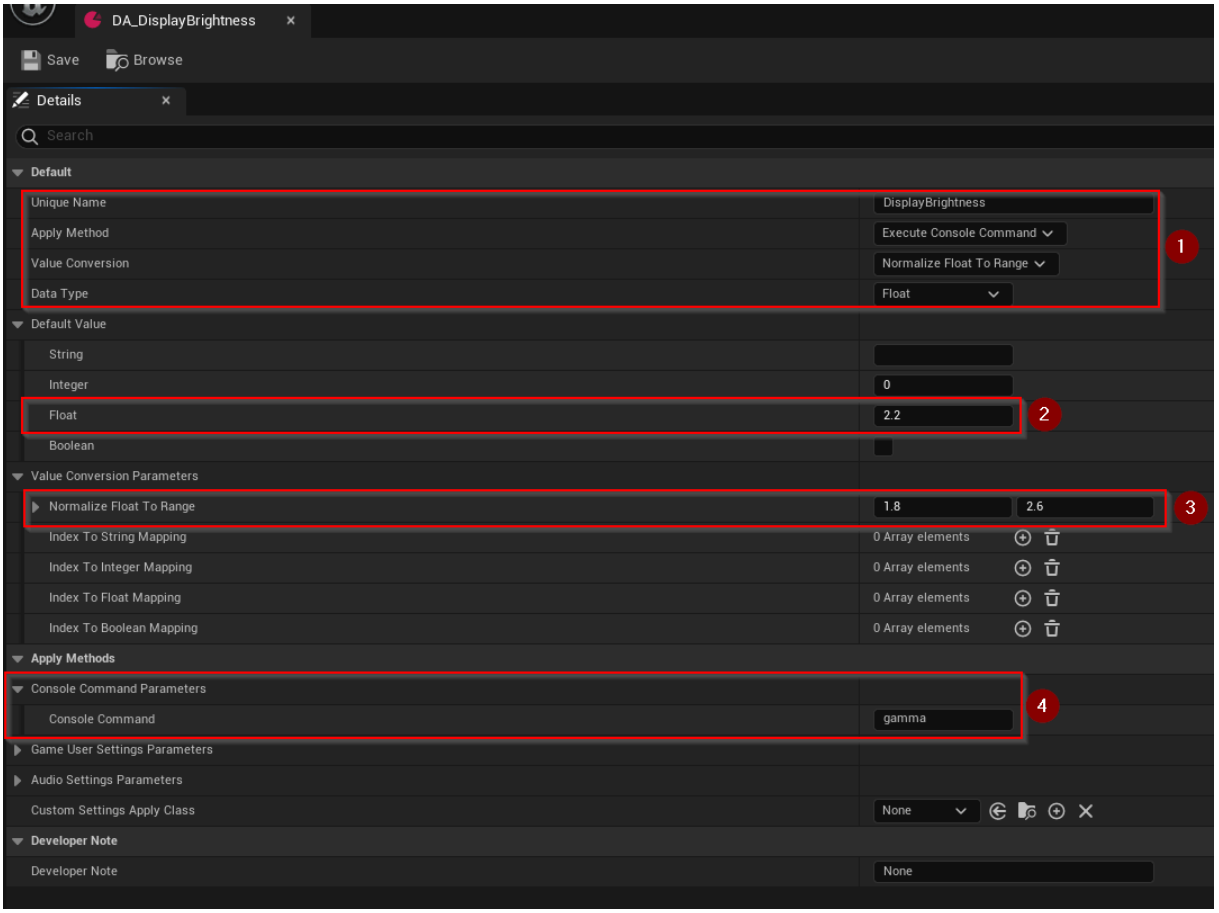
Here is an example of how it's hooked up to a settings widget:



This handles the load and save of the settings value and also the conversion/mapping from and to the widget display values.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

This is how a Settings Data Assets looks on the inside:

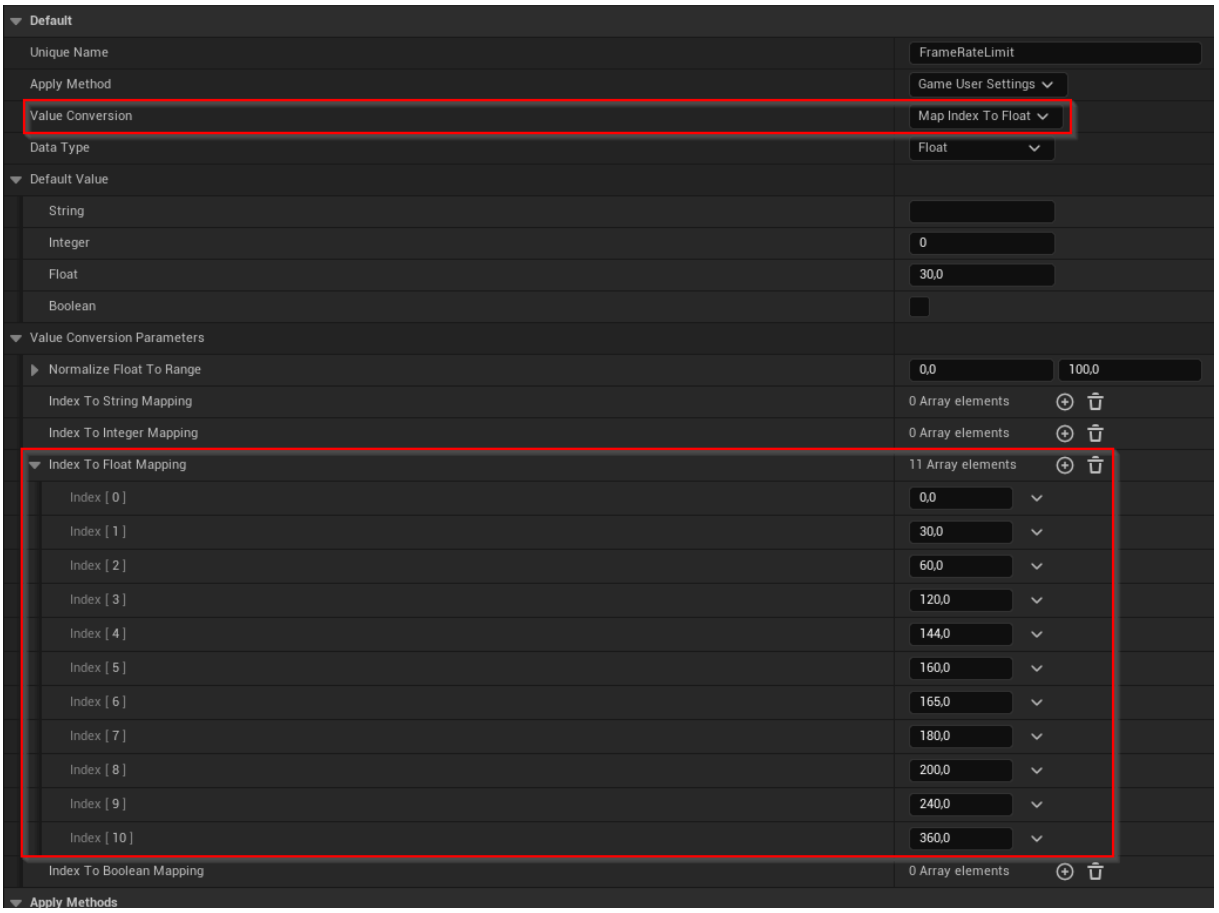


1. General Properties
    a. Unique Name = Field Name in the JSON config file
    b. Apply Method = How it's going to be applied (see Apply Methods)
    c. Value Conversion = How the value is converted (explained below)
    d. Data Type = String, Integer, Float or Boolean
2. A default value has to be specified for the chosen Data Type
    a. In this case it's 2.2 which is the default gamma value for UE
3. Since Value Conversion is set to *'Normalize Float to Range'* the slider value (50-150) will be converted to the range of 1.8 to 2.6.
4. According to the Apply Method 'Execute Console Command' we have set the console command 'gamma' which is called with the converted value. E.g. *'gamma 2.2'*

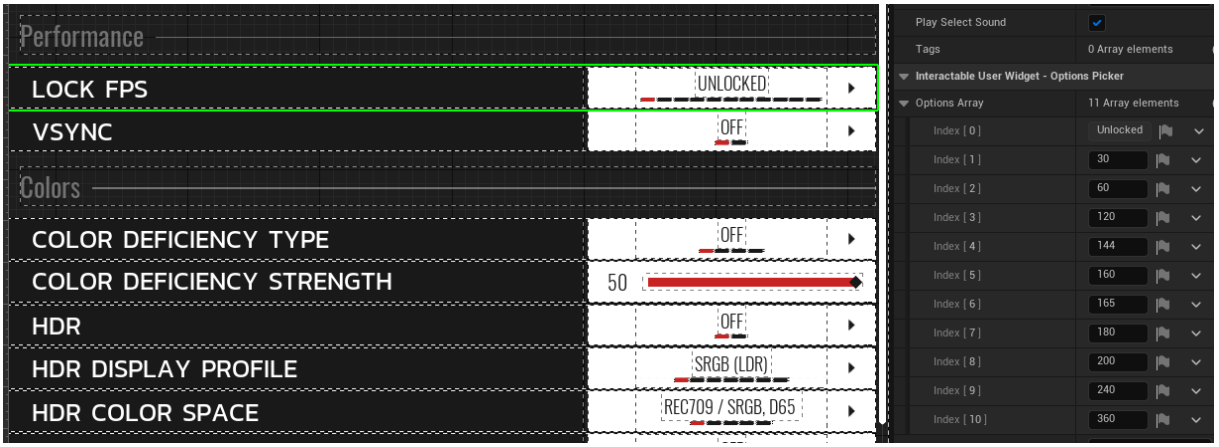
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Value Conversion Mappings

Mappings can be utilized to for example convert the index of an OptionsPicker to an actual float value. Like in this case for the FrameRateLimit Settings Data Asset:

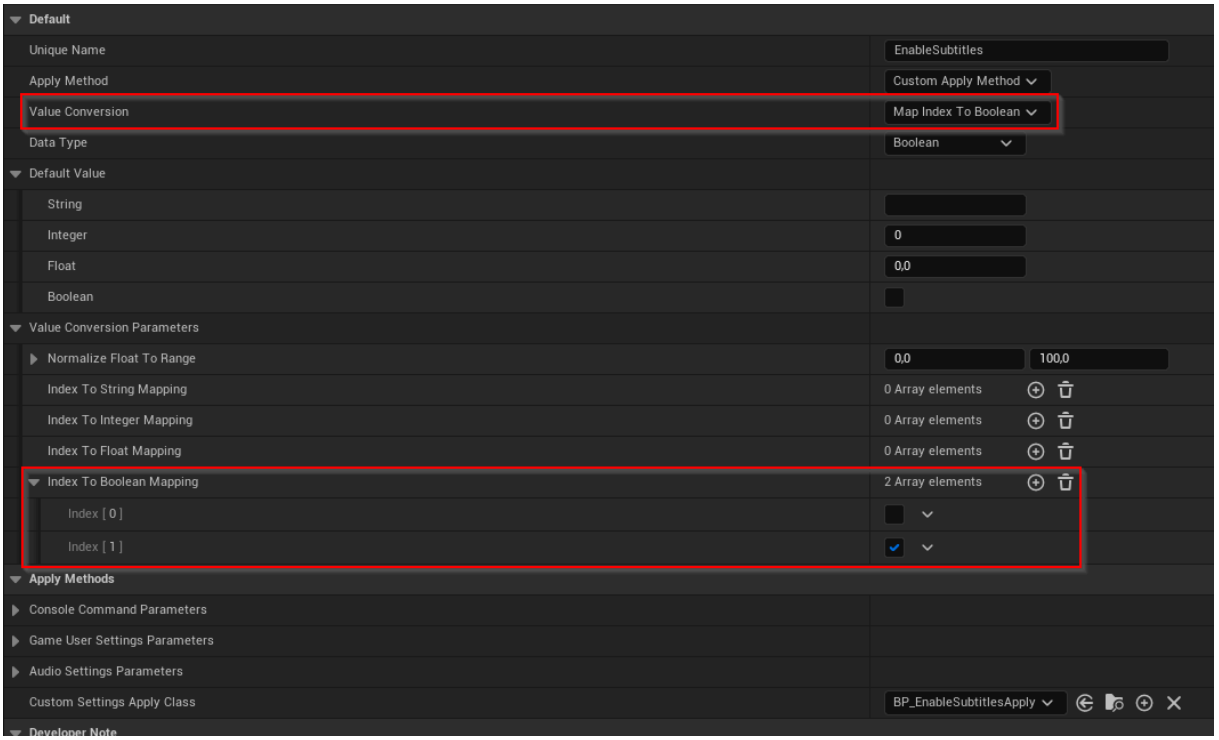


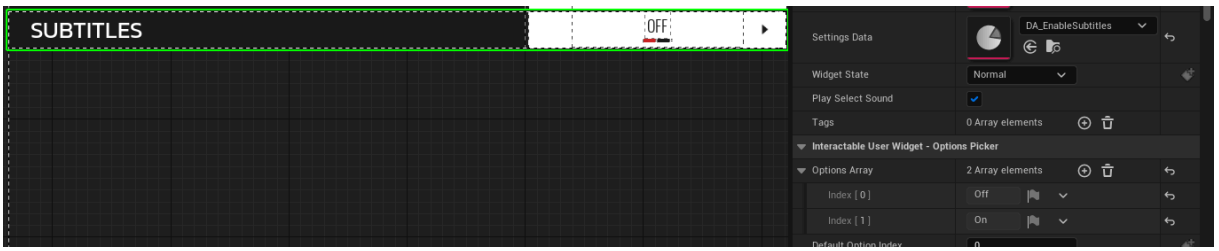
In the UI you will see the corresponding indices:



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Another good example is EnableSubtitles. In this case the OptionsPicker index is mapped to two bool values which are fed into it's Custom Apply Class:





Apply Methods

The Settings Data Asset provides several apply methods that can be selected based on specific requirements. The following is an explanation of how these methods work.

- Custom Apply Method

  - This method enables the calling of a blueprint class with custom logic
  - Learn more about it in the Custom Apply Classes subchapter

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

- Game User Settings

  - This method enables the calling of Unreals Game User Settings functions
  - It should offer all relevant functions that you might need
  - If you need to extend it though, you can find the code for it here: '/Blueprints/Settings/BP_SettingsManager' or build a Custom Apply Class which is better in terms of updatability

- Sound Class Volume Property

  - This method allows to set the volume property of a specified Sound Class asset

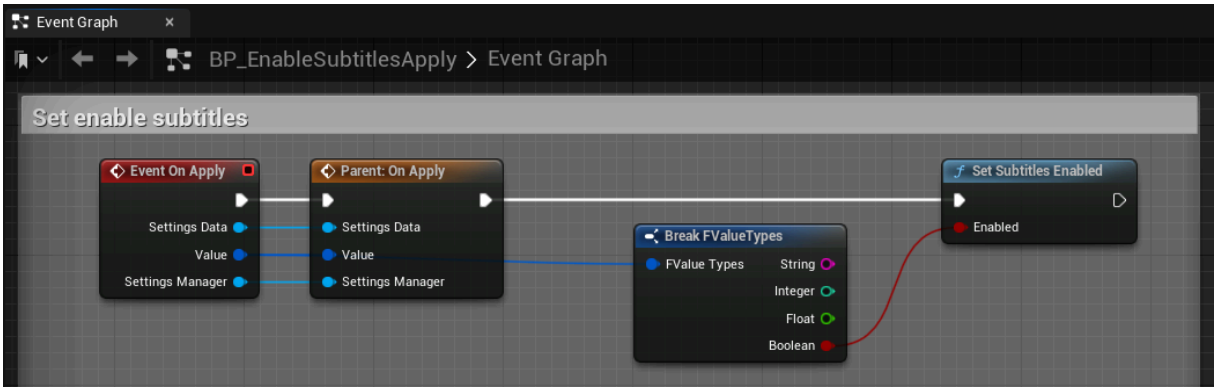
- No Apply Method
  - Nothing is done when applying. The value is just saved to the JSON config file
  - An example for this is the *'DA_PlayerName'* Settings Data Asset

Custom Apply Classes

By using the Custom Apply Class method, you can easily invoke Blueprint classes that are derived from *'BP_CustomSettingsApplyClass'*, and customize them with your own logic to achieve the desired functionality.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

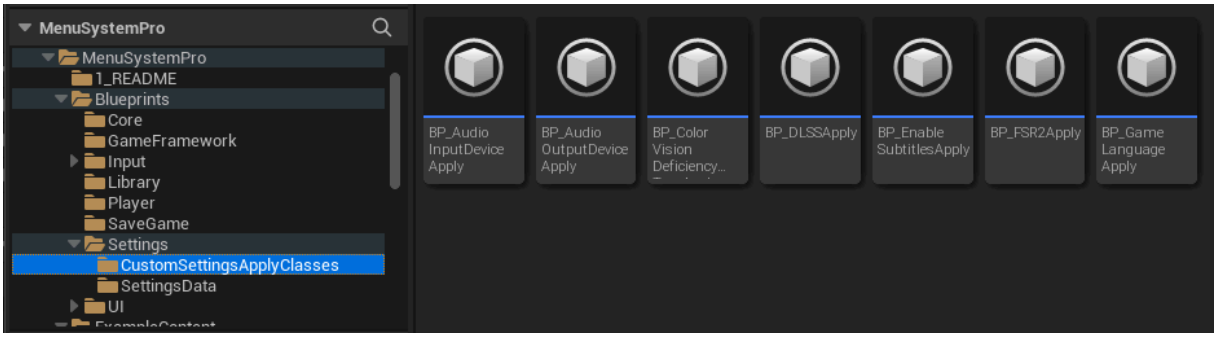
A great and simple example for an apply class is the *'BP_EnableSubtitlesApply'* class:



'/Blueprints/Settings/CustomSettingsApplyClasses/BP_EnableSubtitlesApply'

The custom event *'OnApply'* is provided by the parent class and will be called when this setting is going to be applied. So make sure to implement it when building your own Apply Class. You can then access the Settings Value by breaking the Value and take the right type as selected in the Settings Data Asset.

We have implemented a variety of Custom Apply Classes, among which DLSS and FSR2 stand out as the most complex ones:



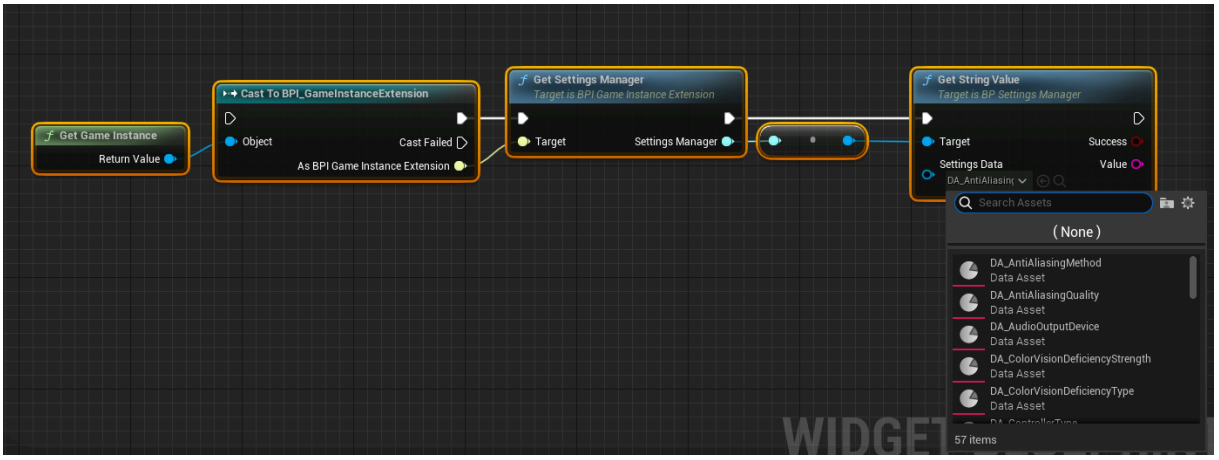
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Access Settings in Blueprints

You can easily access the saved settings by using the following nodes in your blueprint.

Access through Settings Manager

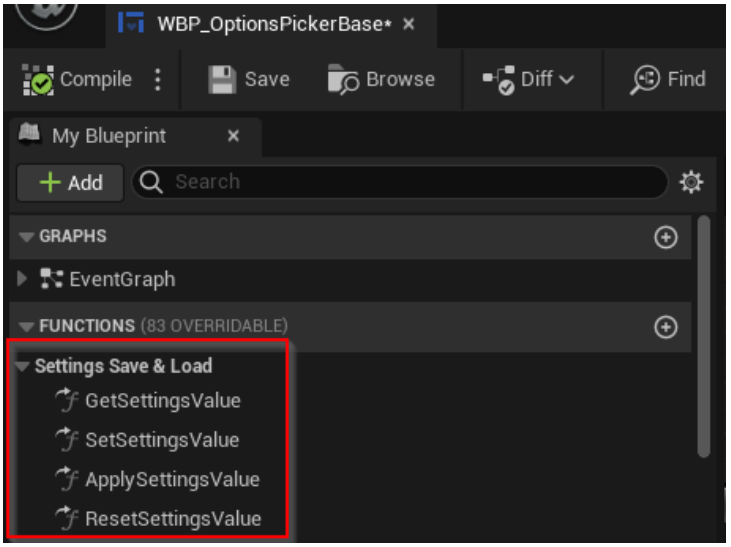
The recommended way is accessing Settings Data via our Settings Manager. It can be retrieved from the Game Instance like this:



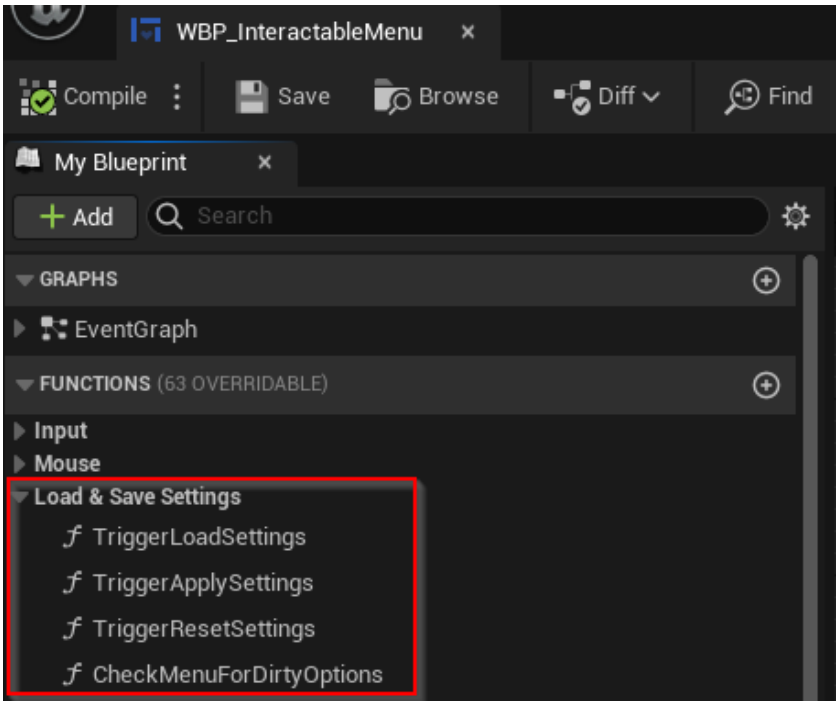
You can use the following functions: *GetStringValue, GetFloatValue, GetIntegerValue, GetFloatValue and GetGameUserSettingsValue*. These functions allow you to access data of different types such as strings, floats, integers, and more. Additionally, there are corresponding setters available to write data of these types to the settings.

Our approach to accessing settings data is designed for optimal performance. Instead of retrieving data from the actual file each time, our method retrieves data from the JSON object held inside the Settings Manager. This means that data can be accessed quickly and efficiently, without incurring any unnecessary overhead.

All of our shipped settings widgets use a similar signature to access settings data:



The Interactable Menu is also involved in triggering the settings logic in all Interactable User Widget that it contains:

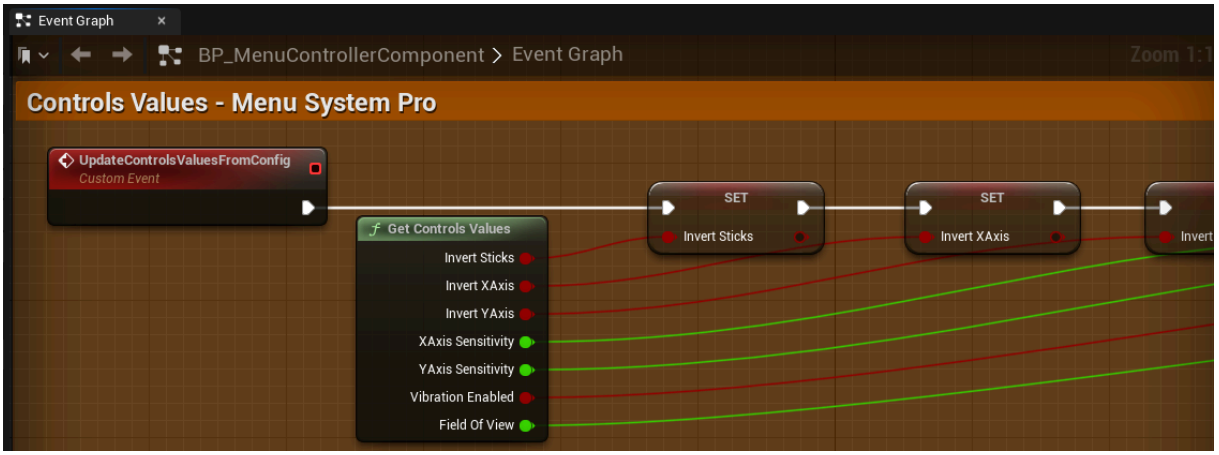


Our approach checks for "dirty" menus, indicating changes that require saving, and makes the apply button interactable.

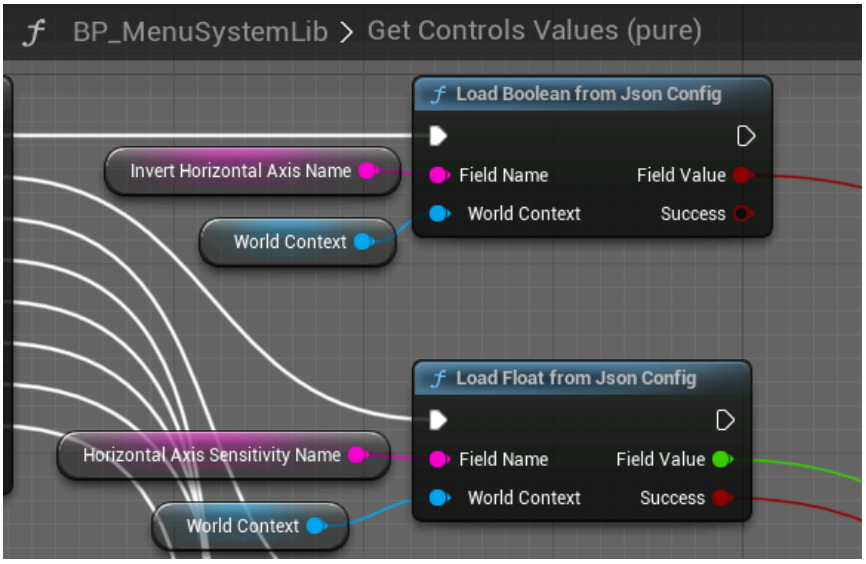
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Access through Library Functions

You can also access data from the JSON settings file anywhere in your Blueprint code using our '*BP_MenuSystemLib*'. We have a good example in our '*BP_MenuControllerComponent*' where we read controls settings for the Player Controller:



That function just reads the settings from JSON by field name:



You can also call those Load functions directly in your code. But be aware to use them in an efficient manner to not hurt your games performance.

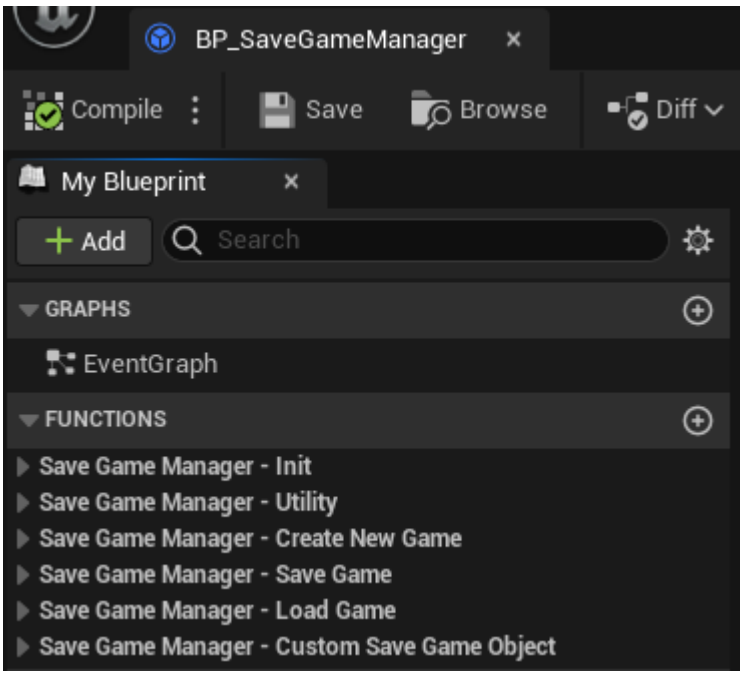
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Save Games

We integrated save & load using the official Unreal Engine Save System. Refer to the Unreal Engine Documentation for the general concept:
https://docs.unrealengine.com/en-us/Gameplay/SaveGame
Save Location in Editor: YourProject/Saved/SaveGames
Save Location Packaged Game:
C:/Users/YourUser/AppData/Local/YourProject/Saved/SaveGames

Save Game Manager

The menu system utilizes a Save Game Manager that is linked to the Game Instance, in order to handle save games (create, load & save).



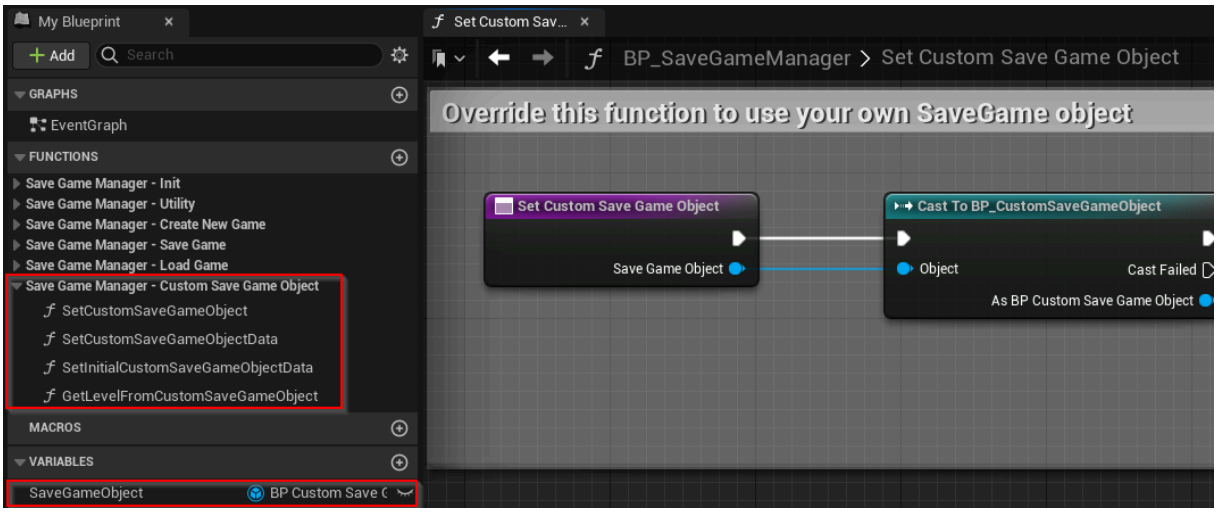
It is located at: '*/Blueprints/SaveGame/BP_SaveGameManager*'

The Save Game Manager is used in the Singleplayer Menu, Ingame Menu and Save Game Menu to handle the save slots for saving and loading the game.

We integrated our example Save Game Object in it with basic variables to store some information which is then being displayed on the Save Slot widgets. To learn more about using your own Save Game Object, please refer to the next chapter.

Save Game Object

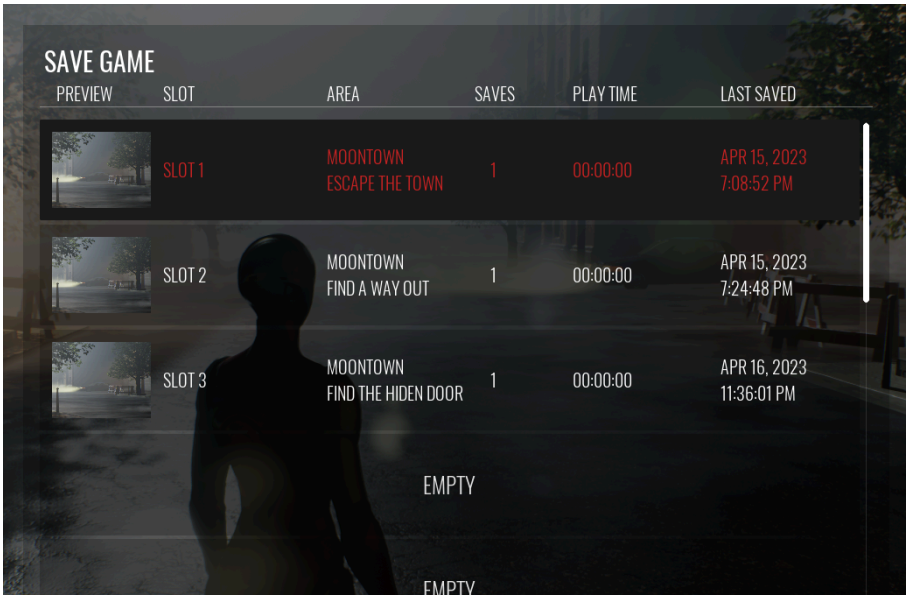
If you plan to use your own Save Game Object you will need to do some modifications in the highlighted functions of the Save Game Manager and also change the type of its SaveGameObject:



The marked functions contain casts to our BP_CustomSaveGameObject and also Getters and Setters for its specific variables. You can replace them to meet your specific needs.

Save Game Menu

Our menu system also comes with a context-based Save Game Menu which is used in the Singleplayer and Ingame Menu (Load & Save).



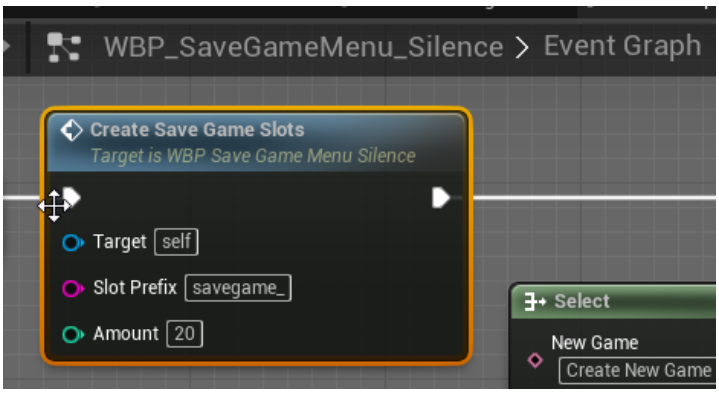
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

It shows simple information about the level, play time, timestamp of save etc.

The widget blueprint for the menu is located at:
'/ExampleContent/Designs/Design_%DesignName%/Menus/SaveGame/WBP_SaveGameMenu_%DesignName%'

Save Slot Widgets

Per default the Save Game Menu creates 20 slots for saving games with savegame_%number% as the name of it.



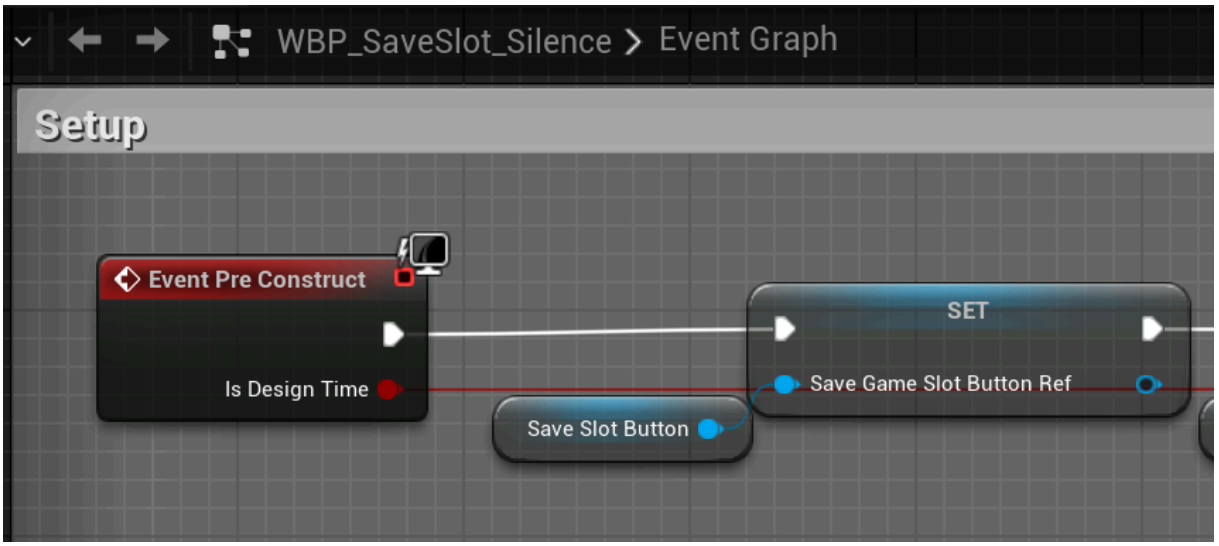
For every slot it will create a **WBP_SaveSlot** widget.

You can check the Blueprint code at:
'/ExampleContent/Designs/Design_%DesignName%/Widgets/SaveGame/WBP_SaveSlot_%DesignName%'

To modify the visuals of the **Save Slots**, you will need to open the above mentioned widget and add your custom text or visual elements. Then, create a new variable of the same type as the one you just added and link it to the corresponding variable in **WBP_SaveSlotBase**. Refer to existing variables, such as **PlayTimeTextRef**, and name your new variable with "Ref" at the end for clarity. Then modify the logic to populate the new widget by using the function **FillWidgetWithCustomSaveGameData**, which will load the custom data from the save game.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

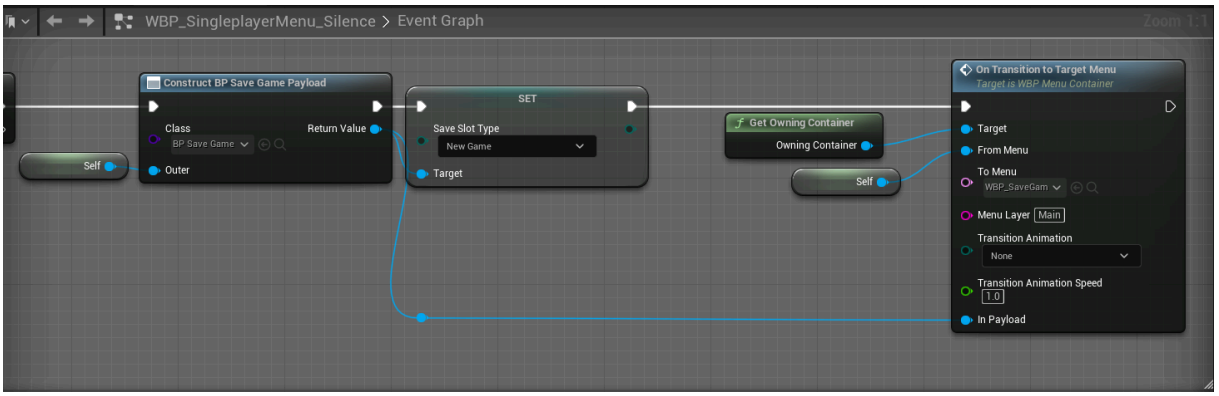
Finally, connect the text in your design's PreConstruct to the newly added **Ref** variable from the **Save Slot Base**. This step ensures the actual widget is passed to the Base class, allowing it to be populated with data using the **FillWidgetWithCustomSaveGameData** function mentioned earlier.



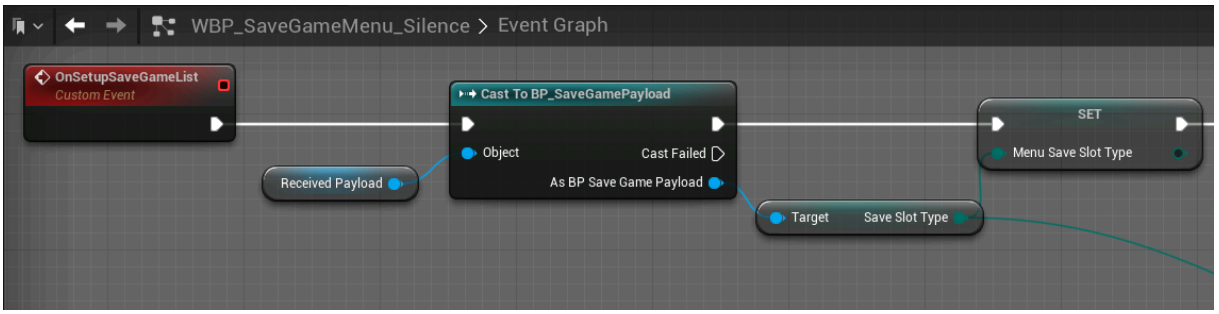
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Save Game Payload

The Save Game Menu can be called with three different Save Slot Types: New Game, Load Game and Save Game. It will be passed in a Save Game Payload object to the menu transition function. This is an example from the Singleplayer Menu:



The Save Game Menu receives the Save Game Payload object, which is then used to construct the menu as intended.



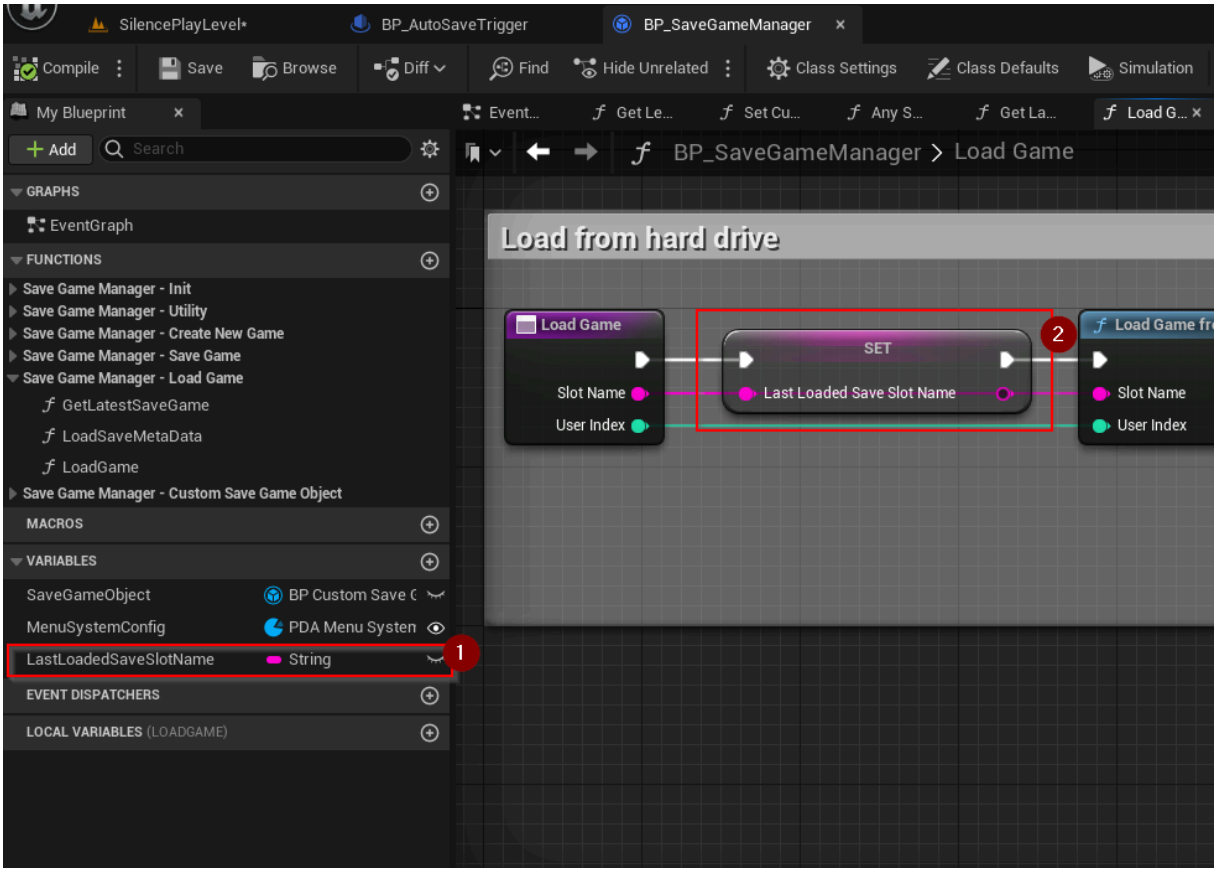
The concept of a **Payload** can be applied to all menu transitions, allowing any data to be passed as a generic object. The receiving menu can then cast the payload to a specific class and use it. This provides a flexible method of transferring data between menus. In the case of the **Save Game Payload**, we use this object to pass the **Save Slot Type** to the Save Game Menu, which can be either 'New Game', 'Load Game', or 'Save Game'. Based on the type, the menu adjusts by showing different titles and displaying or hiding specific slots. For instance, with the 'Load Game' type, empty slots are hidden, whereas for 'New Game', empty slots are visible. This approach allows us to use a single menu from different "perspectives."

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

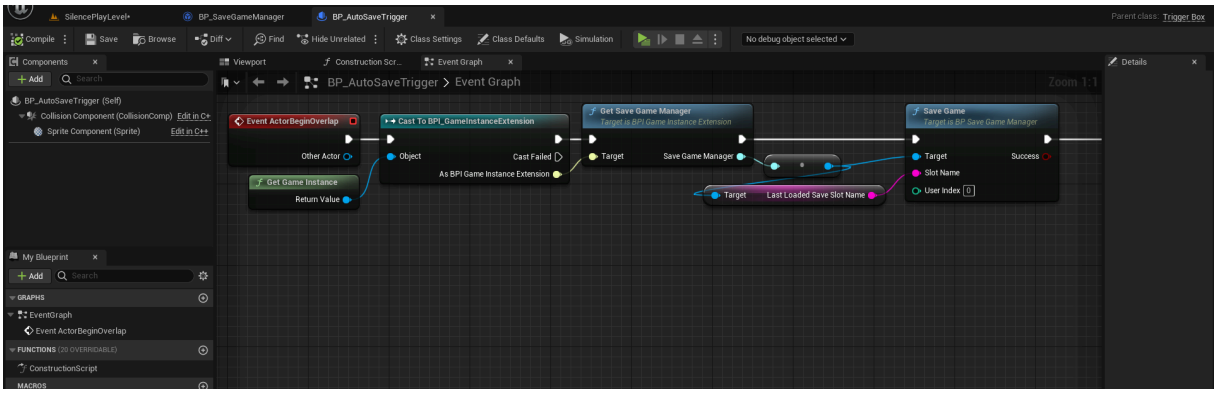
Extension Example: Auto Save

This is an example of an auto save game trigger volume which utilizes the Save Game Manager to save the game to the last loaded slot.

1. Add a new string variable to the settings manager to store the last loaded level in the LoadGame function

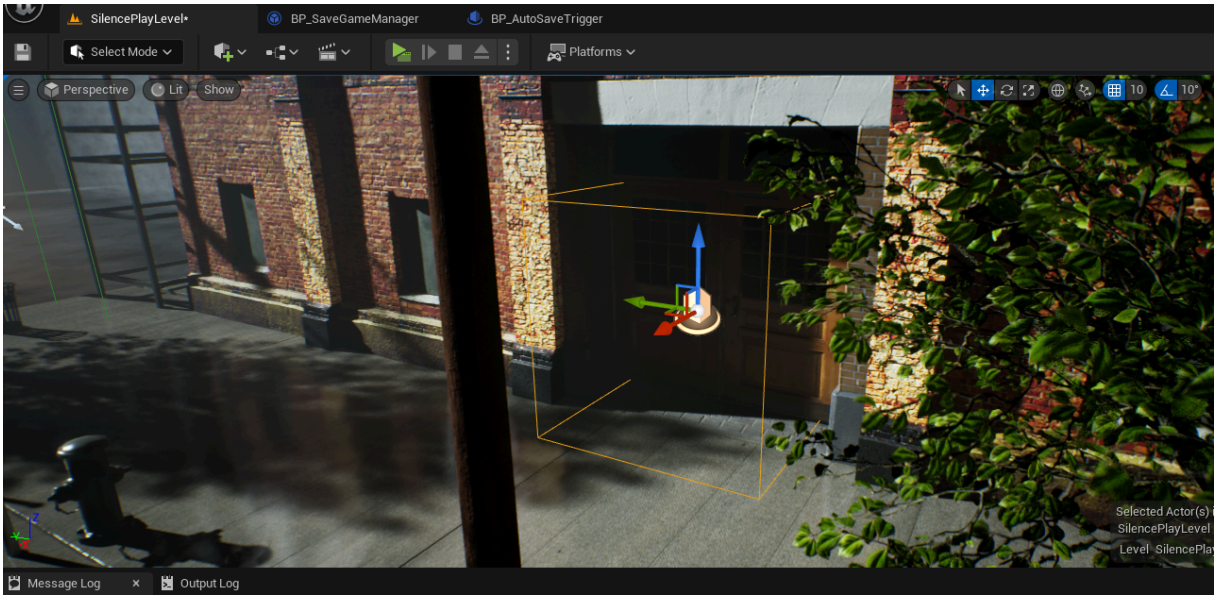


2. Create a new class inherited from the Trigger box class and add this code to ActorBeginOverlap



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

3. Place it in your play level like this



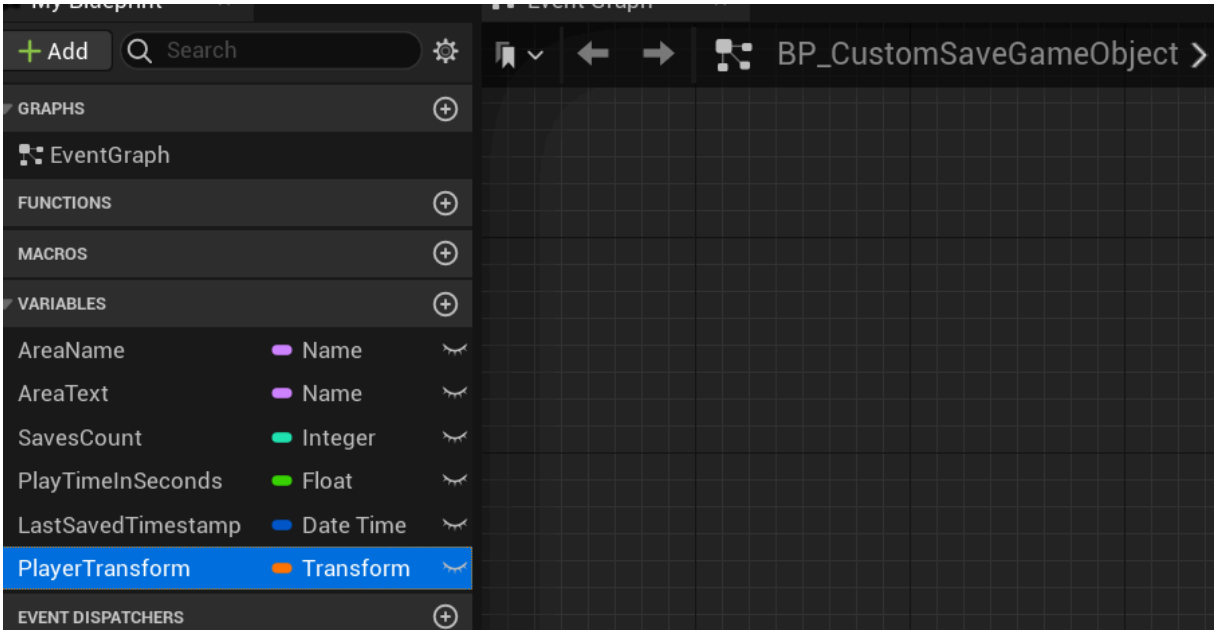
4. Done! Run into it to save to the last loaded slot.

This code can be improved by checking the last loaded save slot name variable before using it (empty, valid etc.) and check if the Actor entered the volume is actually the player character. You could also show a save game indicator UI on screen to show the user that a save has been triggered.

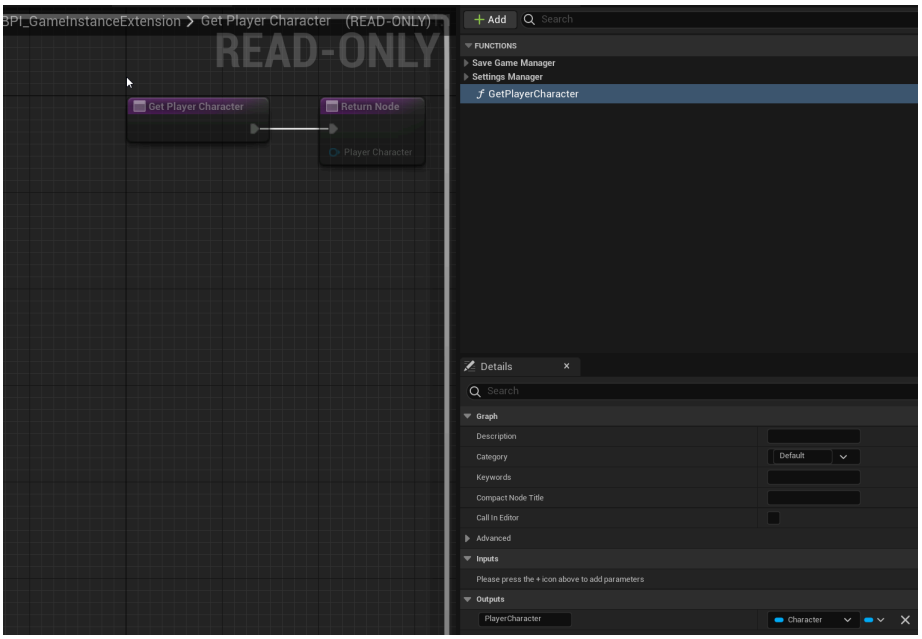
Extension Example: Save Player Transform

This is an example to show how you can save & load the player character's location and rotation.

1. Open your Custom Savegame Object class and add a new variable of type Transform

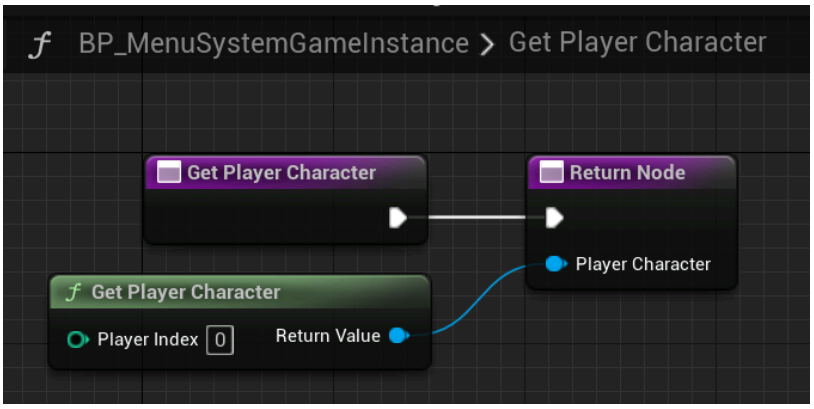


2. Open BPI_GameInstanceExtension and add a function named GetPlayerCharacter with Return Value of type Character

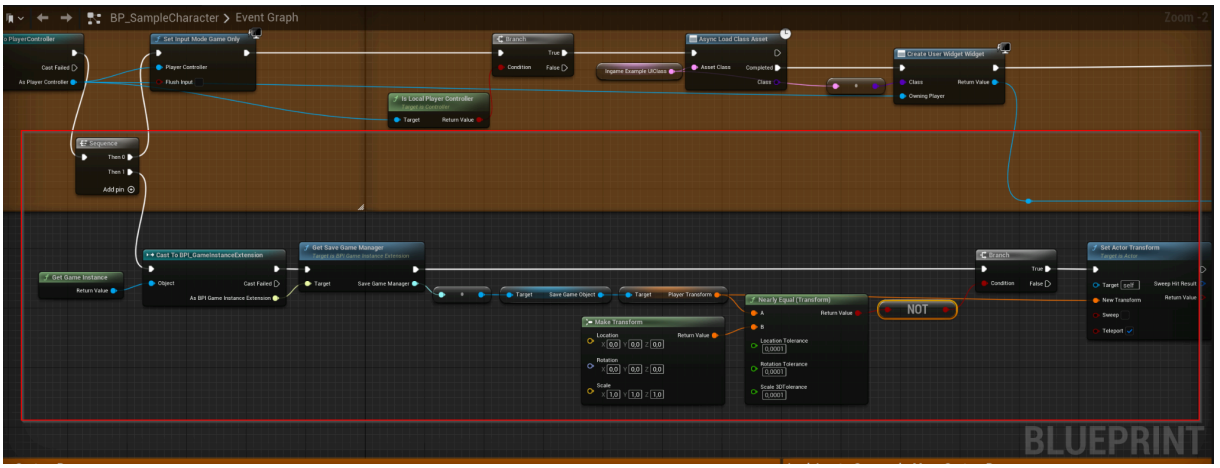


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

3. Open your Game Instance and implement the GetPlayerCharacter function

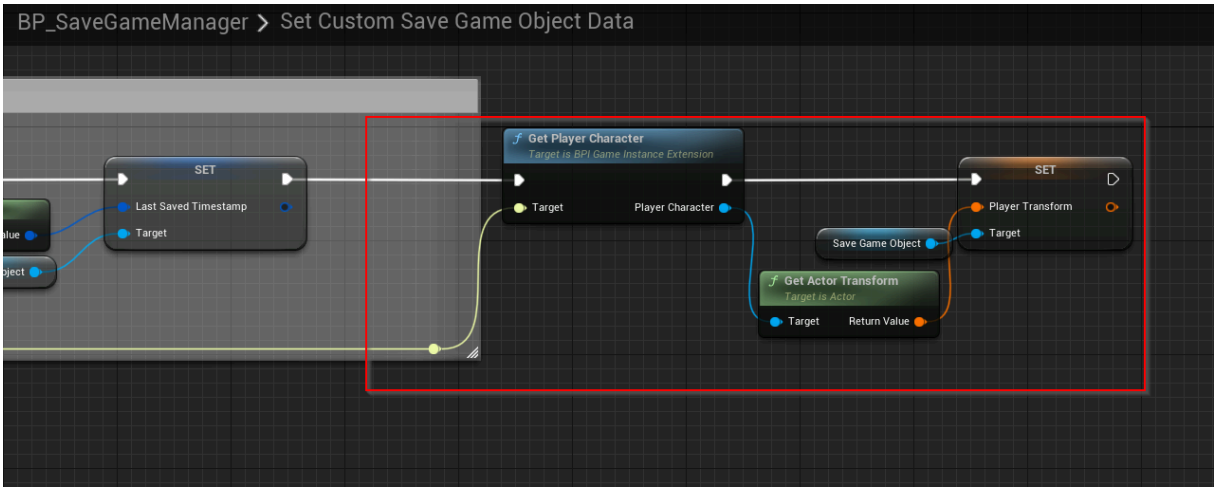


4. Open your Player Character blueprint on Event Possessed (Client) add the following code:



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

5. Open BP_SaveGameManager > SetCustomSaveGameObjectData and add the following code to the end



If you now save anywhere in your level, your character's transform is saved to the Custom Savegame Object. When your level loads and the Player Controller takes control of the Player Character it will call the Event Possessed, which in turn will call the new load transform logic. The code in the player character ensures that the Transform is only used when it is different from the Zero/Default value of a transform, so it's important not to skip that part.

Multiplayer

Menu System Pro includes pre-built functionality for both online and local multiplayer, with multiple menus available for those modes.

Online Multiplayer

We've implemented online multiplayer using the Unreal Engine's Online Subsystem, and have developed multiple menus specifically for this mode. Refer to the Unreal Engine Documentation for more information about the Online Subsystem: https://docs.unrealengine.com/5.0/en-US/online-subsystem-in-unreal-engine/

General

We use the Online Subsystem Null which is the default subsystem in Unreal Engine. If you need to change it to for example to the Steam Online Subsystem you will need to change it like explained in the official documentation.

If you use plugins like Advanced Sessions you will need to replace or modify a few nodes explained in the chapter Modification Notes.

Online Multiplayer Menus

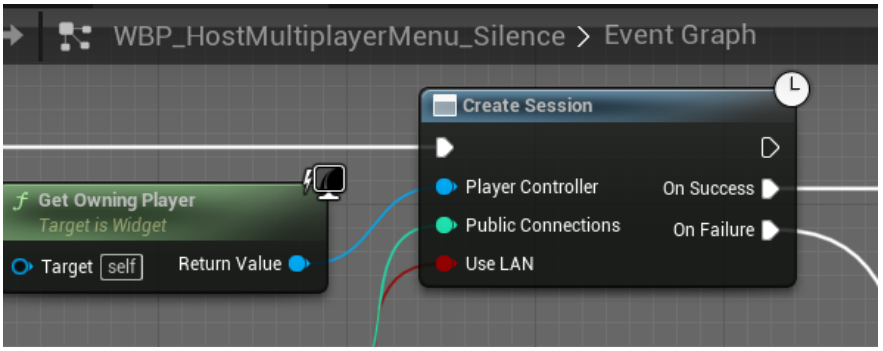
- Host Game
  - Host your own game servers for LAN and internet players.
- Join Game
  - Quickly join an open game, just like in popular games such as Fortnite or PUBG.
- Server Browser
  - Browse through a list of selected games and choose one to join

All menus are located at:
/ExampleContent/Designs/Design_Silence/Menus/Multiplayer/

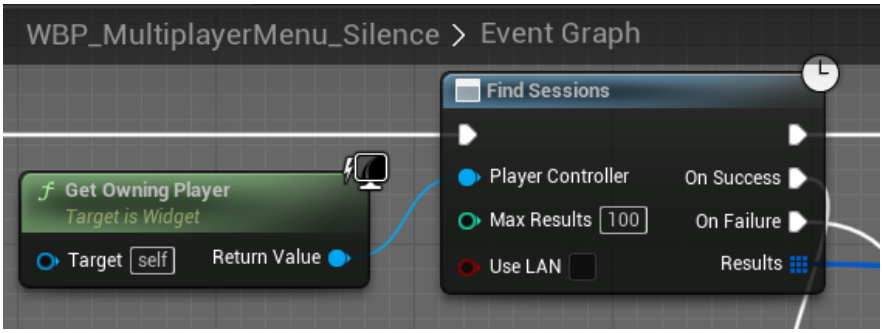
Modification Notes

We used a few nodes from the Online Subsystem that need to be replaced or modified when using a different subsystem like the one by Steam.

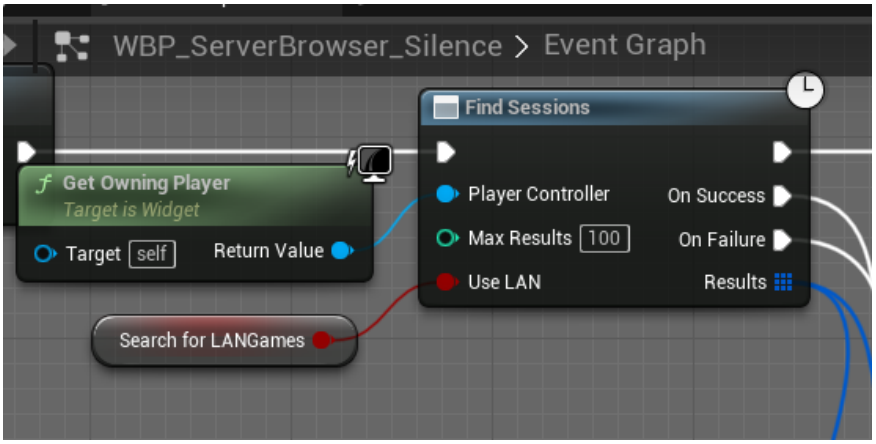
- Host Multiplayer Menu



- Multiplayer Menu (Join Game)



- Server Browser



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Local Multiplayer

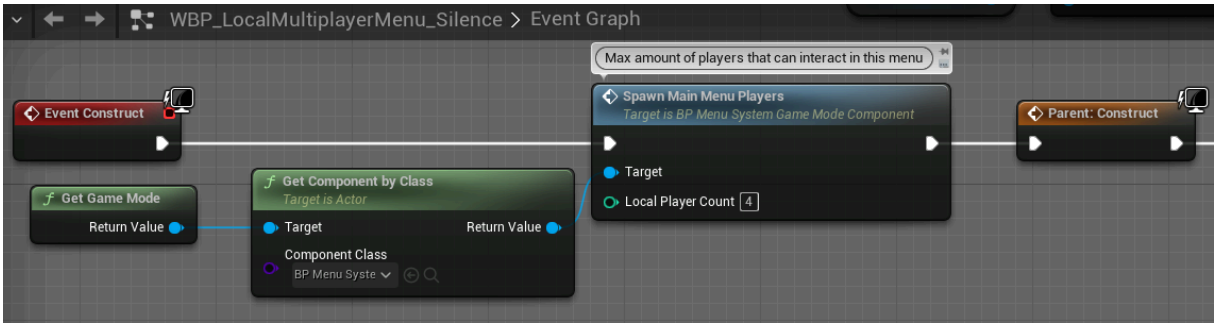
We've implemented local multiplayer functionality, including split-screen gaming, to support couch co-op and other similar scenarios.

Local Multiplayer Menu

Allows up to four players to join the local multiplayer game session.
The menu is located at:
/ExampleContent/Designs/Design_Silence/Menus/Multiplayer/

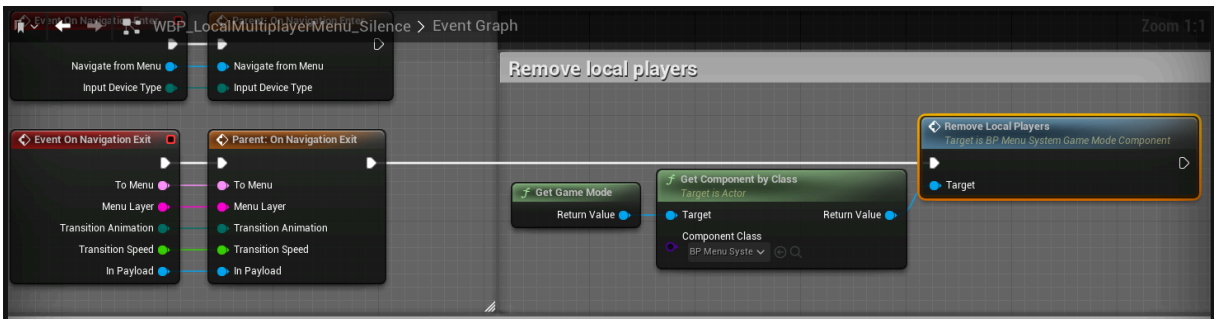
Modification Notes

- Local multiplayer menu (max. Player Count)



This node creates four local player controllers. As soon as they have been added four gamepads can be used in the menu.

When the menu is exited, the local player controllers are removed except for the first one.

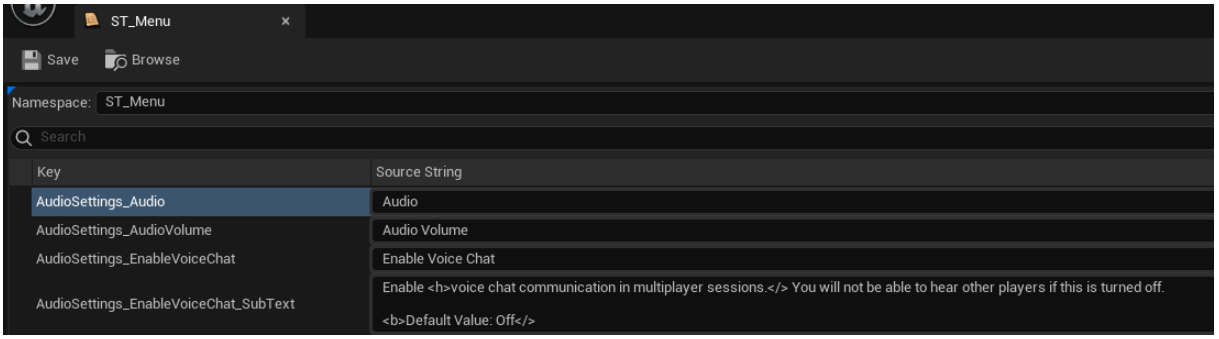


You can also use this node for your custom logic to remove local players.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

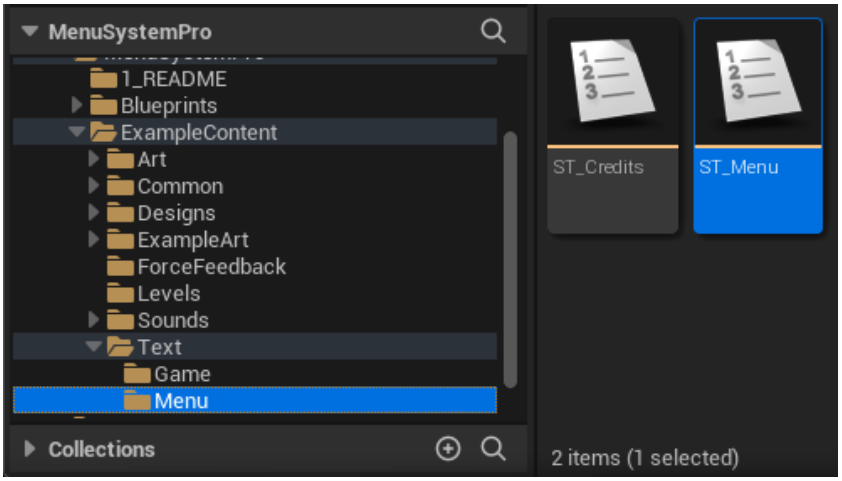
Localization

The menu integrates the Unreal Engine Localization Dashboard, see https://docs.unrealengine.com/en-us/Gameplay/Localization. The text used in this asset is stored in string tables, located in the Text folder.



Localizing your own Text Variables

1. Open the ST_Menu string table located at: '*/ExampleContent/Text/Menu/*'



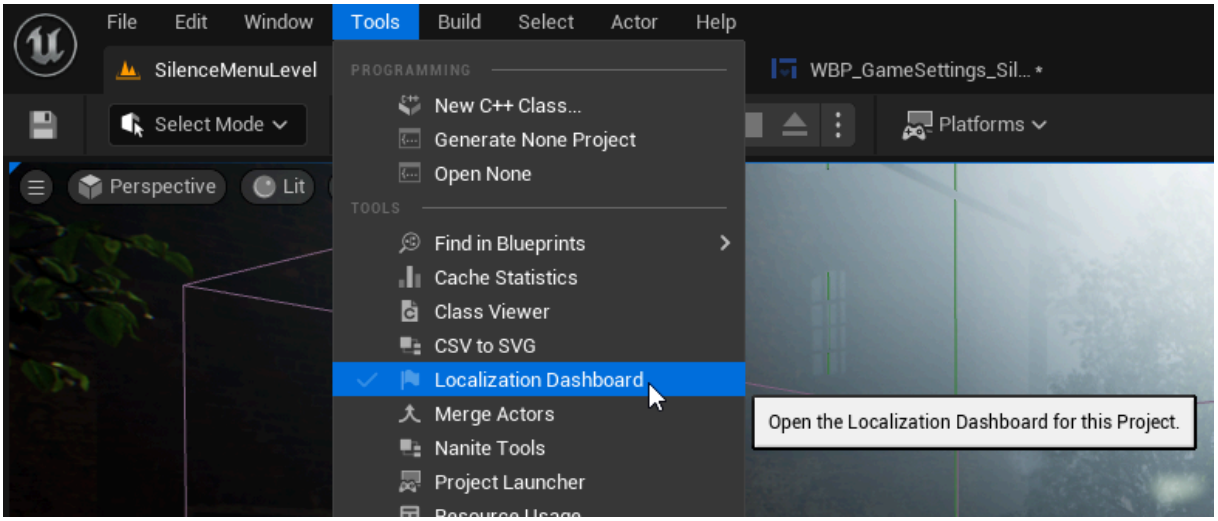
   a. Alternative: For better updatability, we suggest creating your own String Table outside of our folder structure.

2. At the bottom of the window you can add a new text entry, by entering a Key and the actual text. You can hit enter or the "Add" button on the right-hand side

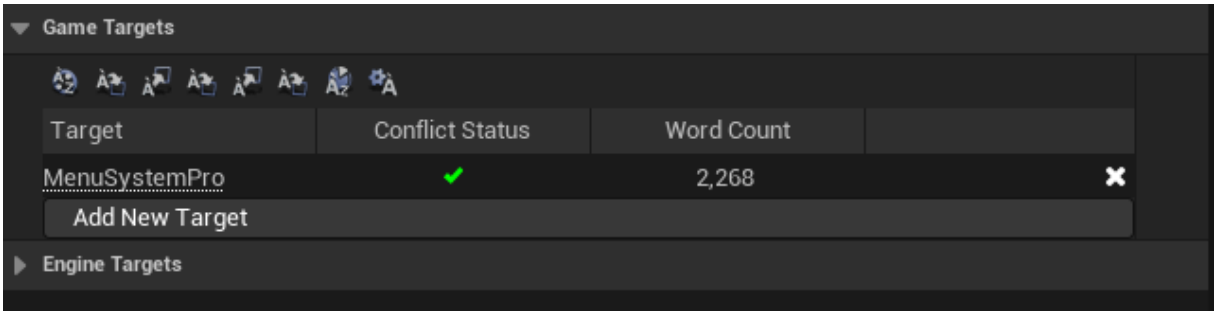
3. Save the string table

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

4. Open the Localization Dashboard

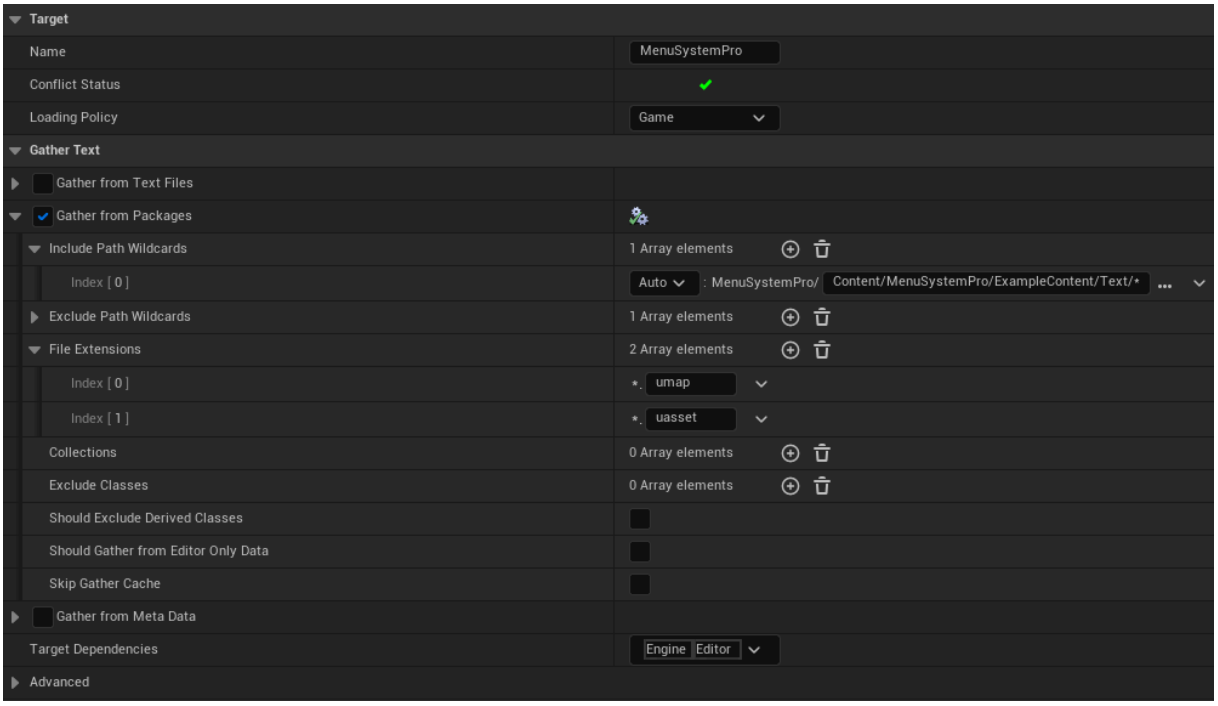


5. Make sure you have a game target named 'MenuSystemPro'. If not, create one.

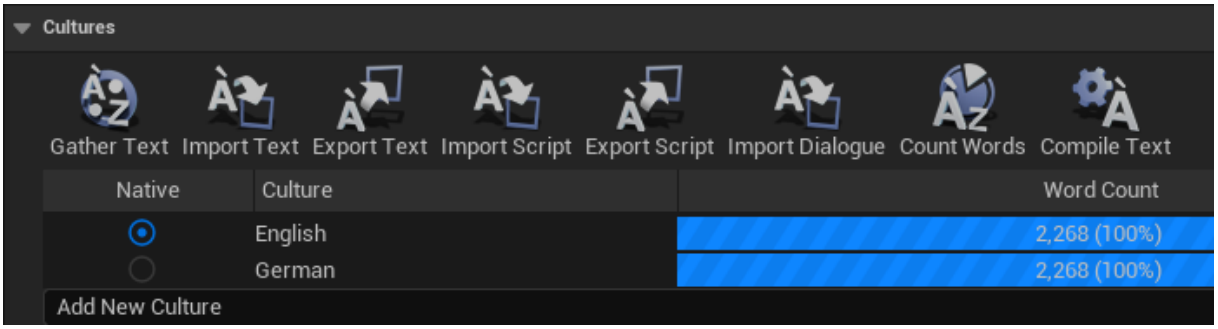


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

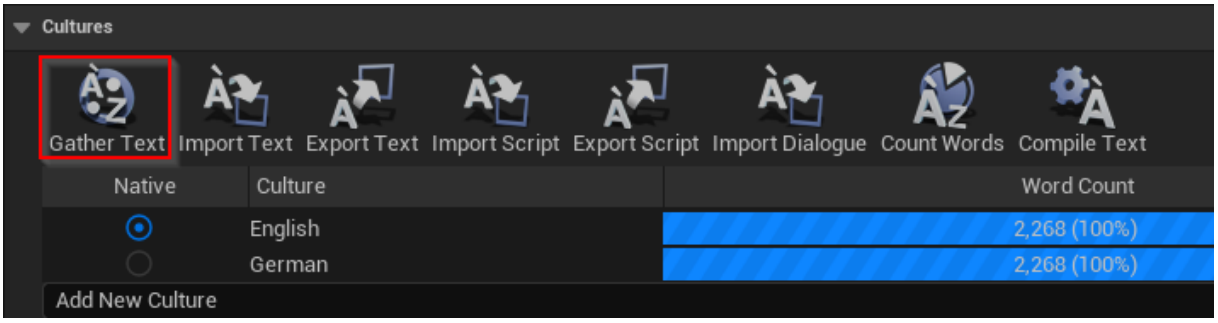
6. Setup the game target like this (add path for the string tables there!)



7. Also make sure English is set to Native before gathering texts



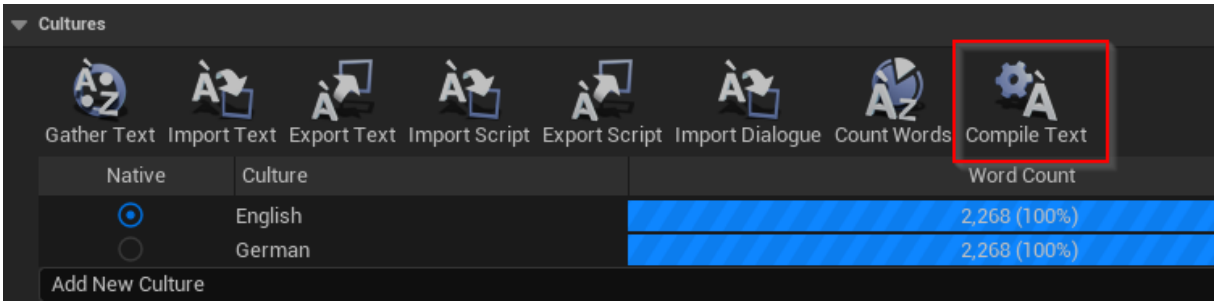
8. Now press *Gather Text*



Note: If gathering the text fails make the *'%PROJECT%/Content/Localization/MenuSystemPro'* folder writable in Windows Explorer

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

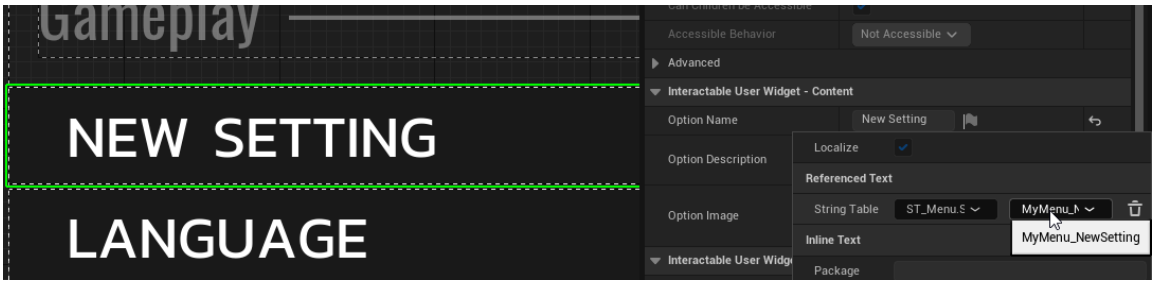
9. After Gathering the word counts should update and show a difference in other languages than English.
10. Hit *Compile Text*



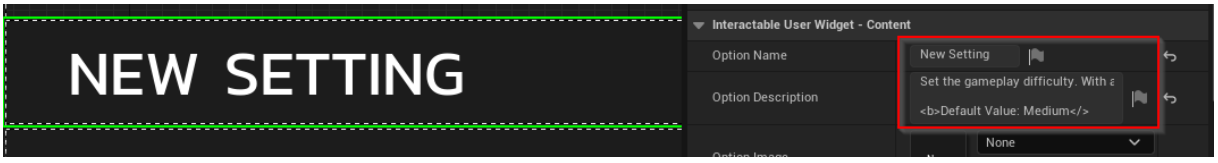
11. Refer to the Unreal Engine Documentation (linked above) for further steps regarding the translation process.

Linking texts to a string table

1. To link any text in the UMG editor to the text in the string table, simply click the dropdown next to the input field and choose the string table and text key



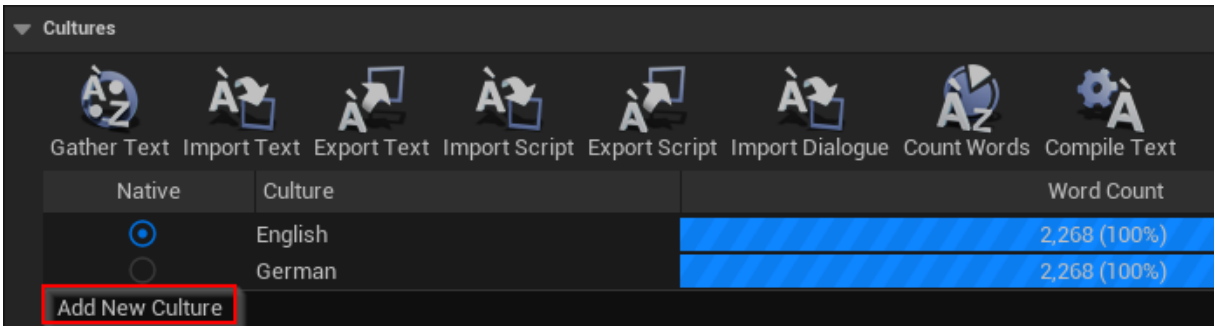
2. Your text variable will appear greyed out, when linked to a string table



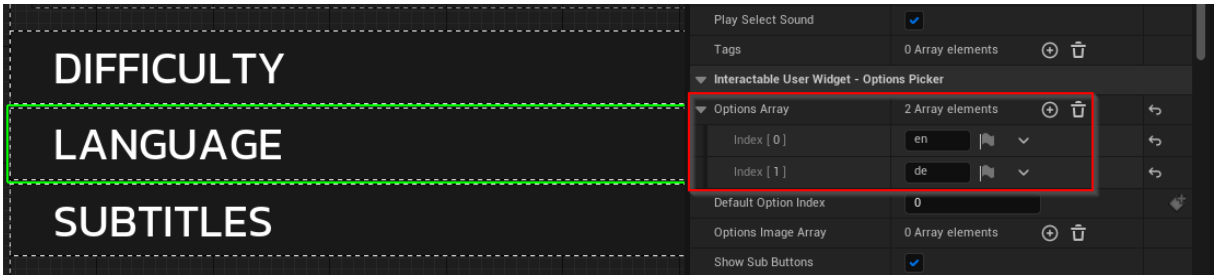
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - [Click here for the new Common UI documentation (MSP 2.2.0+)]

Adding support for new languages

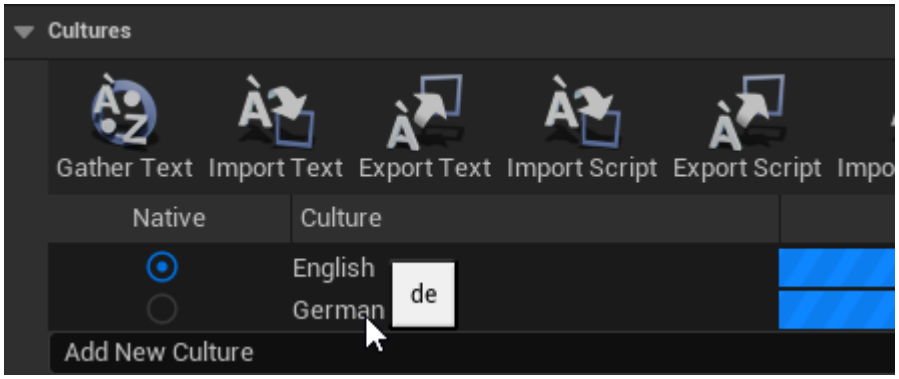
1. Follow the official Unreal Engine Documentation on the Localization Dashboard to add new cultures



2. Navigate to '/ExampleContent/Designs/Design_%DesignName%/Menus/Settings/' and open WBP_GameSettings_%DesignName%
3. Extend the the OptionsArray of the LanguageOptionsPicker

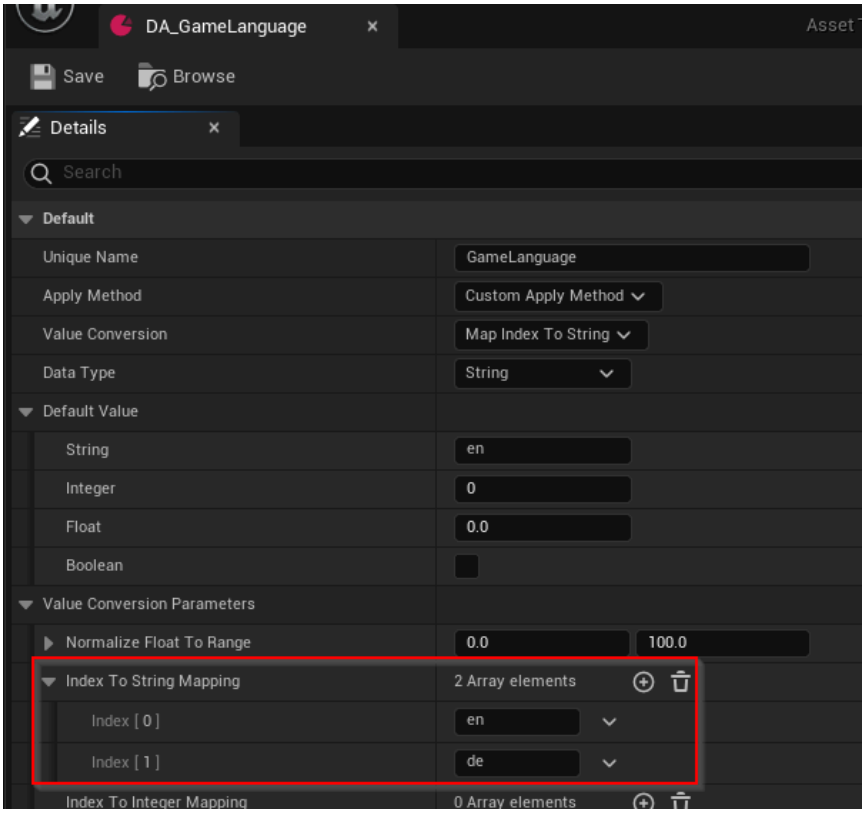


   a. You have to add the language codes like when hovering over Cultures in the Localization Dashboard



4. Also add the new language key to the Language Settings Data Asset located at: '/Blueprints/Settings/SettingsData/DA_GameLanguage'

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)



If you want to protect your changes from being overridden by updates, we advise you to duplicate this data asset into your own folder structure. Then open the Menu System Config and remove the existing data asset for the GameLanguage inside of the settings section. Add yours and you are good to go.

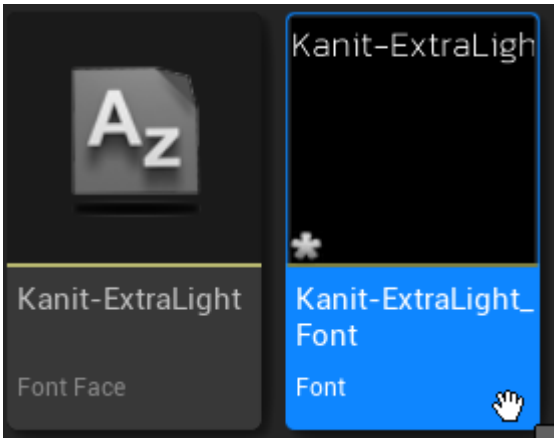
5. Language Names are dynamically loaded during runtime and in the correct language

Note: Always test localization in Standalone or packaged builds. PIE will not work properly.

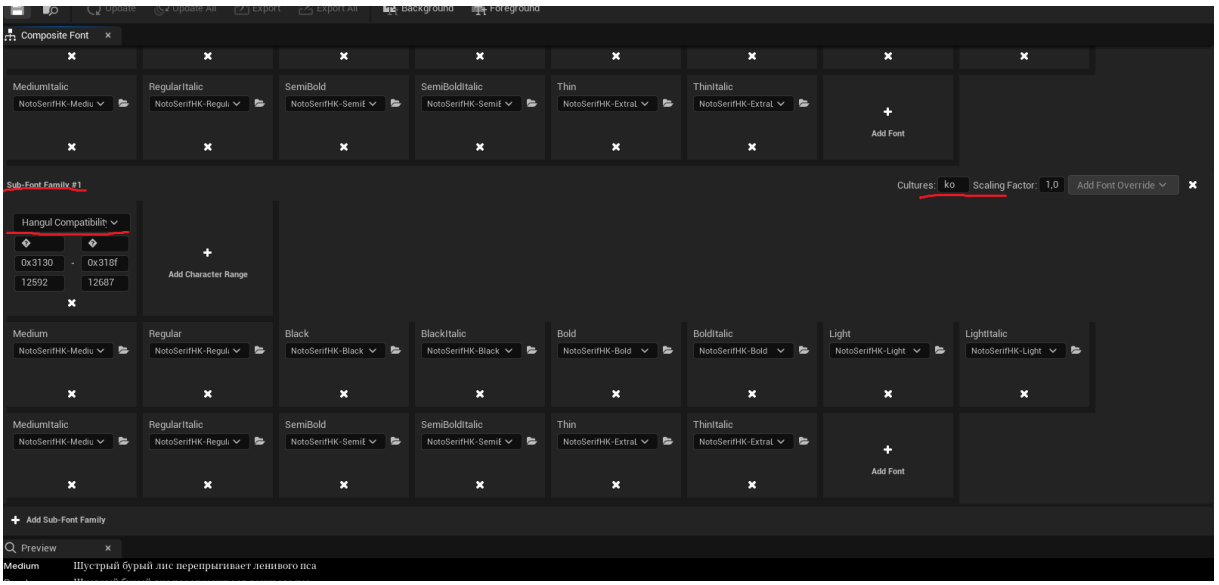
THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Adding fonts based on the language

You can add fonts in the font asset based on the language.



In order to do this add a new sub font family and state the culture code that equals to the one from the Localization Dashboard.

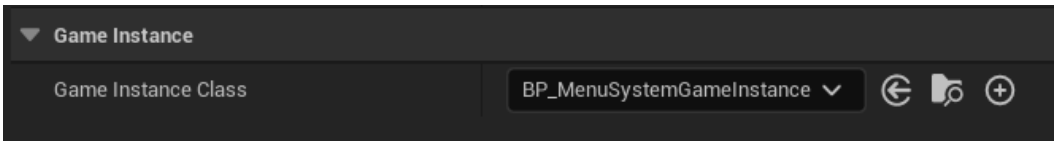


THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

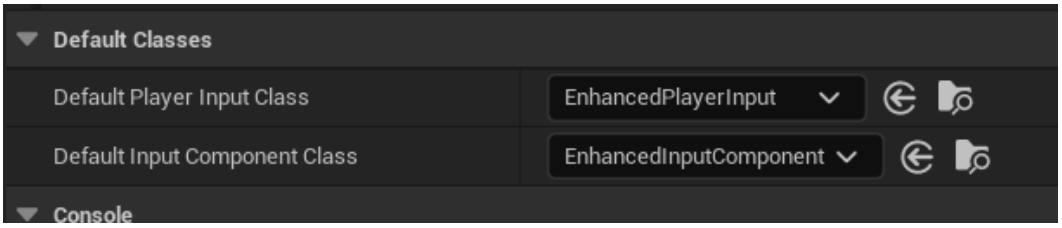
Common Issues & FAQ

Character can't move after setup

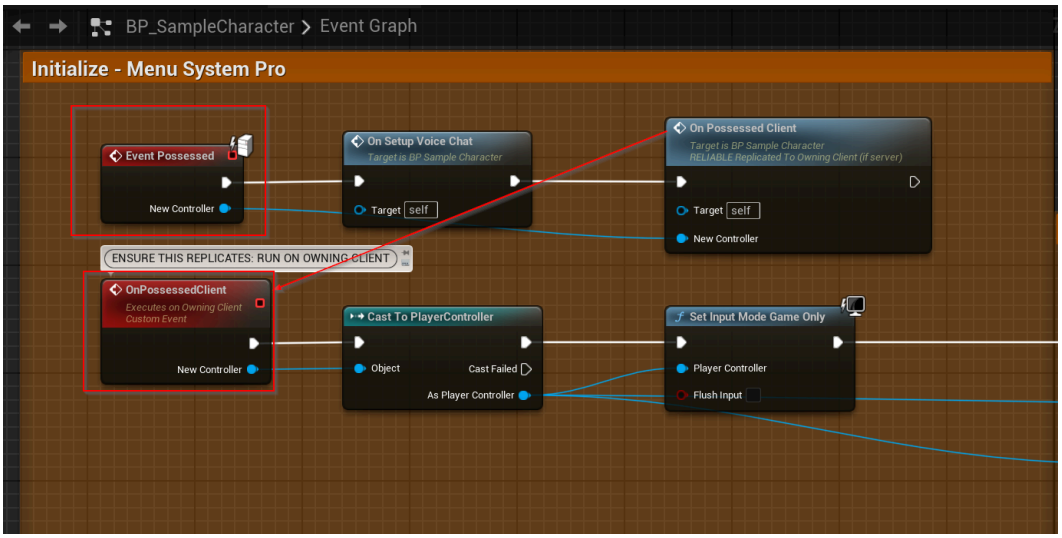
1. Project Settings > Game Instance is not set (or does not implement our events & functions)



2. Project Settings > Input > Enhanced Input classes are not set

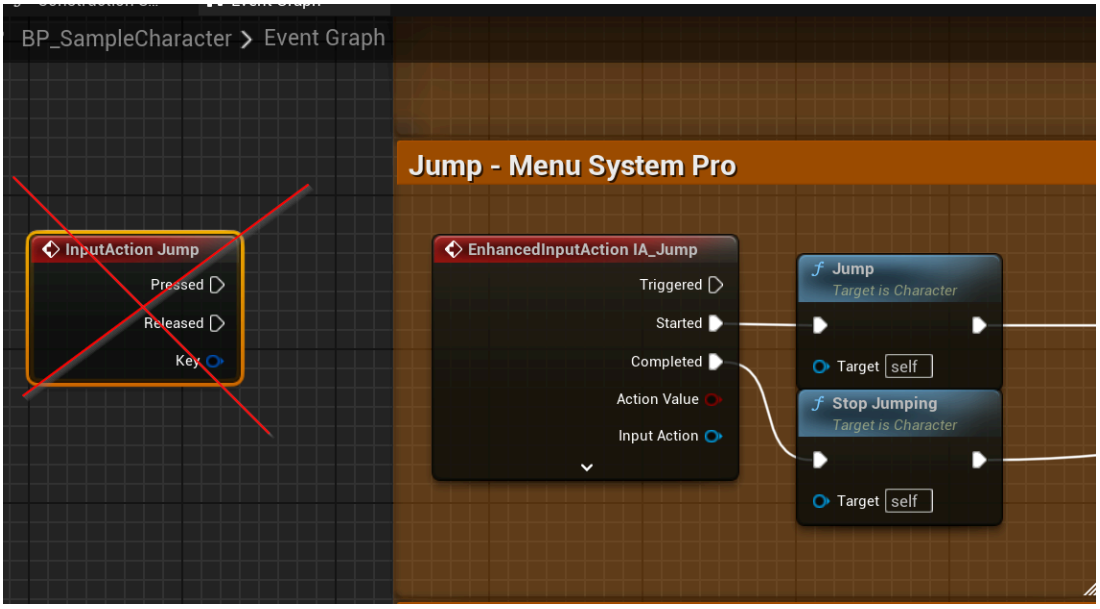


3. In the Character Event Possessed does not call the 'Set Input Mode Game Only' node (DO NOT REMOVE EVENT POSSESSED)



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

4. In the Character the Input Actions are still using the legacy input system and are not Enhanced Input events



5. In the Character the Enhanced Input events implemented have not been added to the Input Mapping Context set in the MenuSystemConfig. See Enhanced Input chapter
   IF YOUR CHARACTER CAN NOT MOVE you very likely need to change IMC_CharacterOnFoot with the IMC your Character uses (e.g. IMC_Default) in BOTH LISTS (Registered Input Mapping Contexts & Initially Added Input Mapping Contexts) in DA_MenuSystemConfig_Silence (NOT THE PDA_ file!). Use 'showdebug enhancedinput' in the console to debug why your character does not accept inputs!

Find Map error

Find Map error when opening the SaveGame menu or when loading into a level. Fix your level meta. See Loading Screen chapter

JSON Plugin in Source Builds

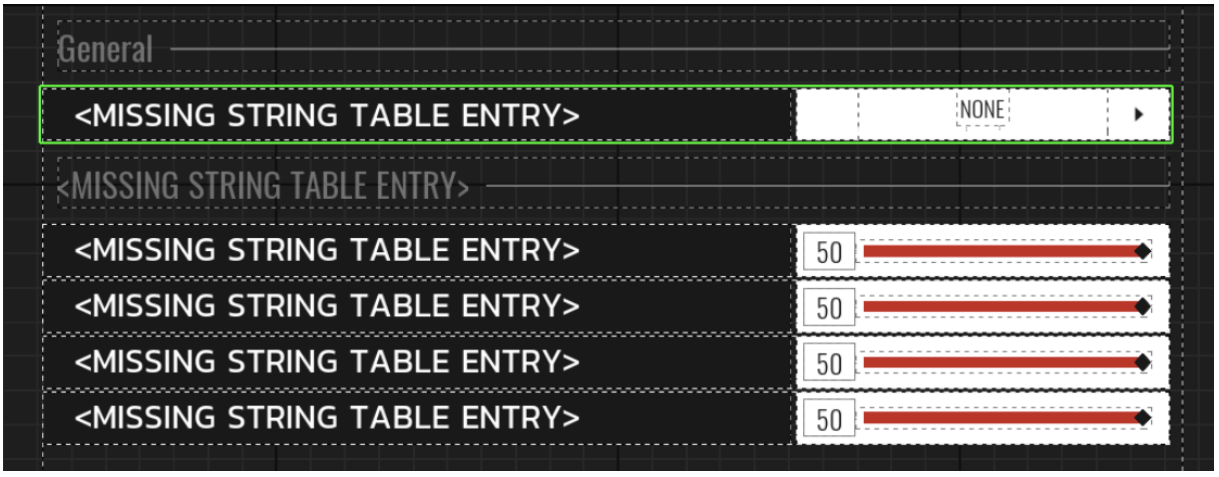
When using a source build of Unreal Engine downloaded from Github you have to setup the Json Blueprint Utilities plugin like this:

```
for source build, you should modify the JsonBlueprintUtilities plugin like that:
"Modules": [
    {
      "Name": "JsonBlueprintUtilities",
      "Type": "Runtime",
      "LoadingPhase": "PreDefault"
    },
    {
      "Name": "JsonBlueprintGraph",
      "Type": "UncookedOnly",
      "LoadingPhase": "PreDefault"
    }
  ]
```

Thanks to @Gabriel98pc from our Discord server

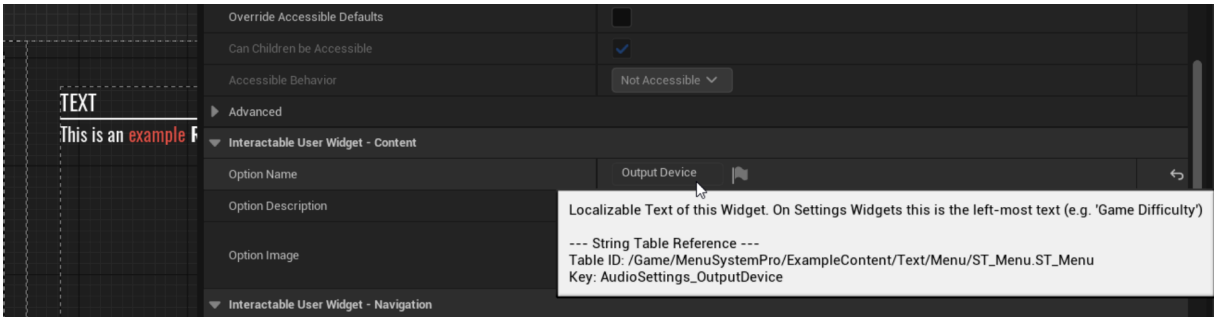
Missing string table entry

After upgrading your project to a newer version you might face the problem of missing strings everywhere, due to a problem with the Localization files.

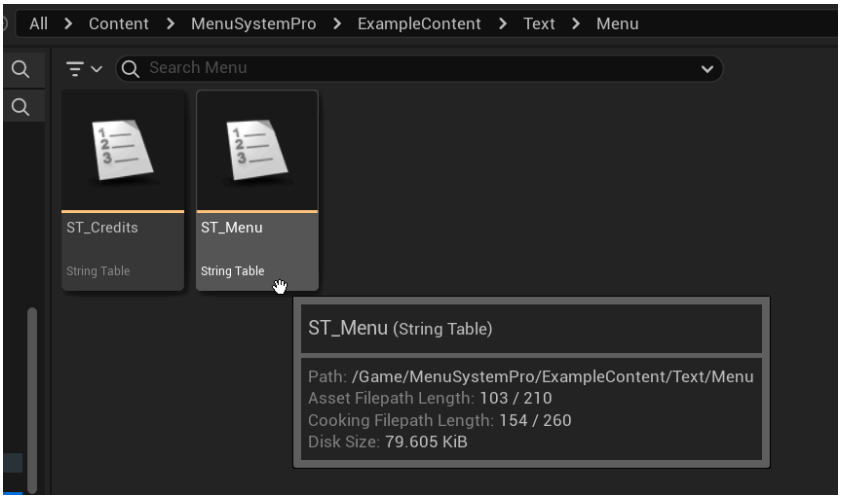


Important: The bug has been fixed by Epic in UE 5.4.2!

In Menu System Pro all texts are linked to a String Table.

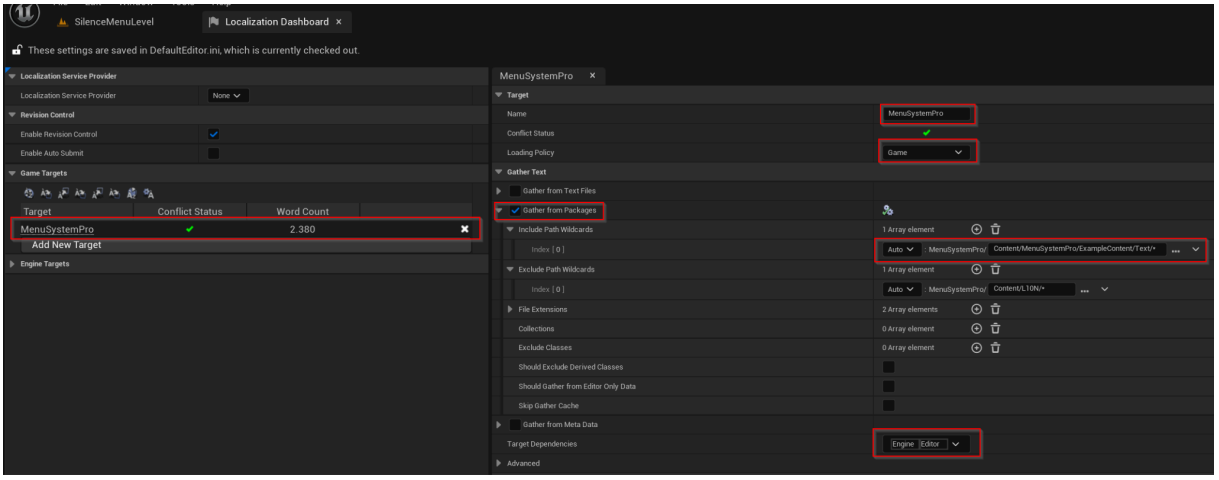


Check the ST_Menu string table here:



It should contain all the texts seen throughout the menu. If it's all good, proceed to the next step.

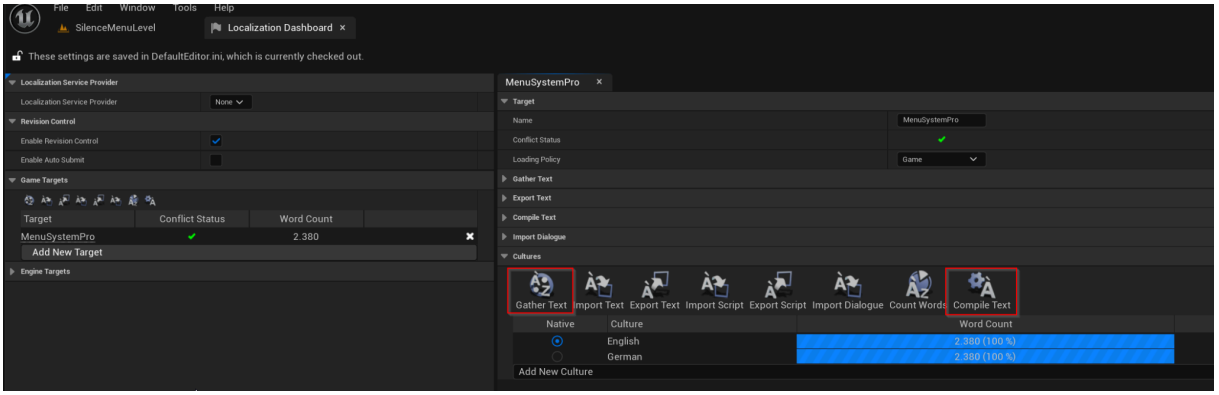
After that check the Localization Dashboard for the Menu System Pro target. This is probably missing and the cause for the missing string table entry issue!



THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

If the target is missing, create it manually with the same name and properties as shown above (double check everything).

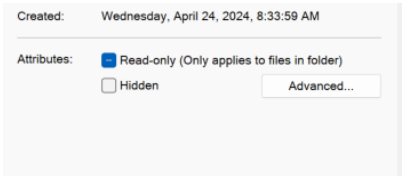
Then Gather Text and Compile Text. It should show a high word count and 100%.



If German is missing add it manually using the Add New Culture button at the bottom and Gather Text/Compile Text again.

In rare instances you might also need to redownload the localization files by adding Menu System Pro to an empty project and then copying the Content/Localization/ folder into your project. Restart the project afterwards.

When the Localization Dashboard throws errors or doesn't allow changes, remove the write protection from the whole folder and all contained files if prompted.



Note: This can be done in Windows Explorer with right-click -> properties

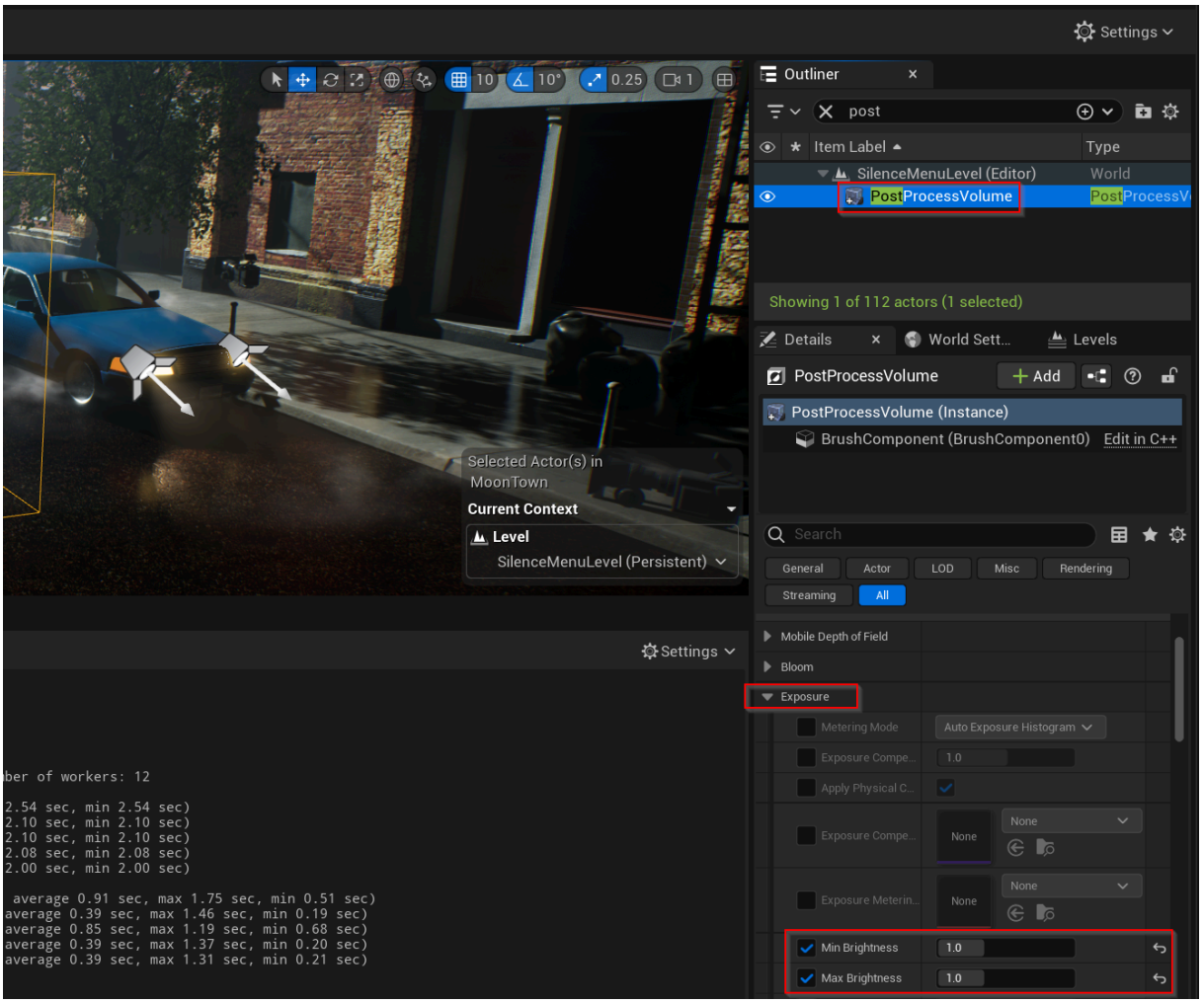
Restart the project afterwards.

Thanks to **@irondry aka CommandantRousto** from our Discord server for helping to find the issue!

It will be fixed in UE 5.4.2:
https://issues.unrealengine.com/issue/UE-211986

Scene lighting is blown out

Your scene is all white glowing and you can't see anything? This can be easily fixed by adjusting the Min/Max Brightness values on the Post Process volume:

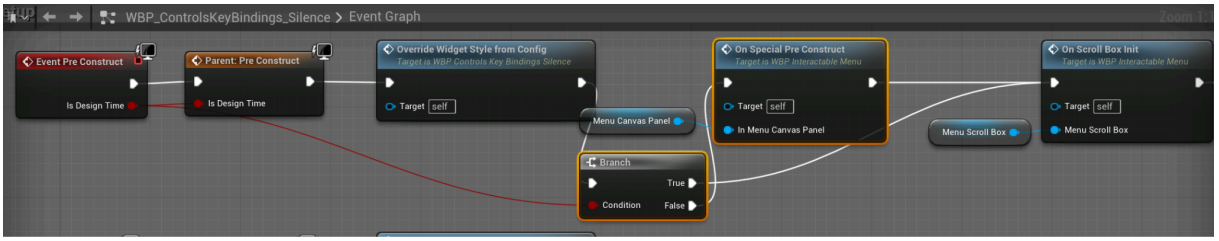


Try to set them both to 1.0 or try different values (based on your Engine version).

Selecting widgets in menu lags/freezes

You might experience lags when clicking widgets in the Designer tab of a menu. Especially in the Controls Key Bindings Menu, which can cause the editor to completely freeze. You can fix it by editing the menus' PreConstruct code. Just add this branch to prevent the OnSpecialPreConstruct from being executed during

design time:

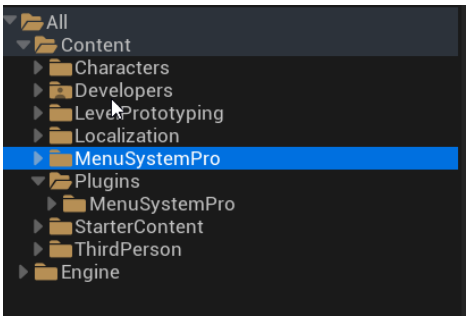


The OnSpecialPreConstruct contains the SetupNavigation function which builds the navigation for each widget in a menu. This is only really needed during runtime. We are still investigating why the performance got so much worse compared to previous versions.

Moving the Menu System Pro folder to a different location

Generally we do not recommend moving the Menu System Pro folder as it can cause various problems related to paths in Unreal Engine. If you really have to do it you may run into some problems with widgets not loading/displaying correctly. To fix this please follow these steps:

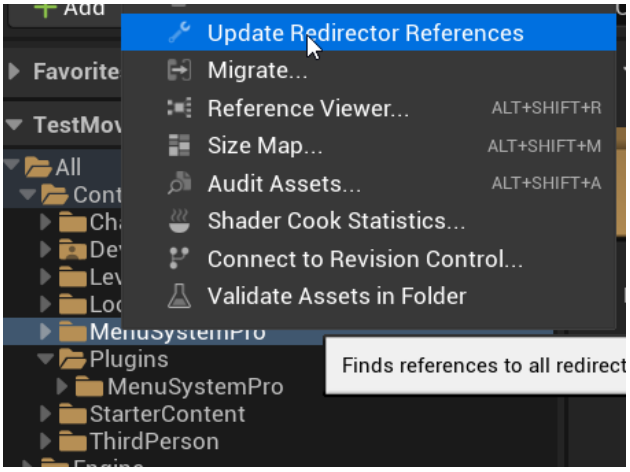
1. Move MenuSystemPro from root to your new location (e.g. a Plugins folder)



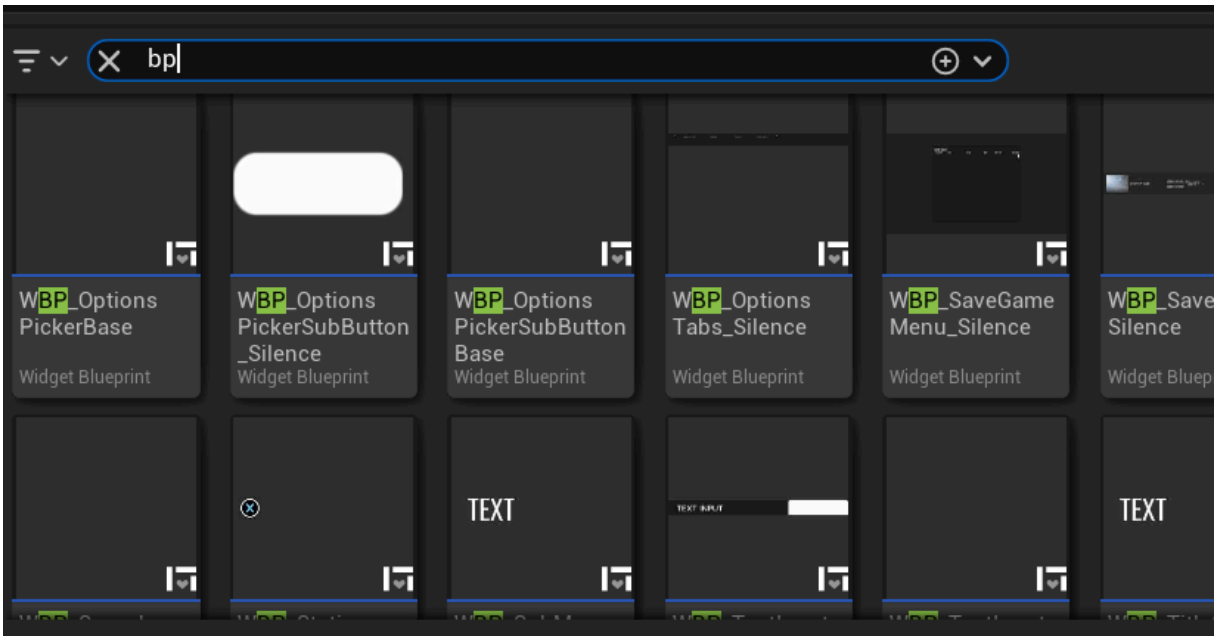
2. Save everything (Ctrl+Shift+S)

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

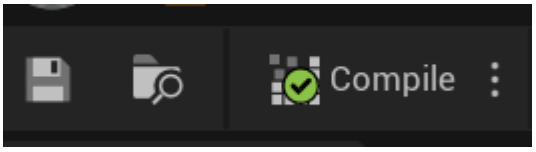
3. Right click the old MenuSystemPro folder and update/fixup file redirectors



4. Delete the old MenuSystemPro folder
5. Select the new MenuSystemPro folder and type 'BP' into the search bar, all blueprints and widgets in MenuSystemPro will show up



6. Open each blueprint/widget and one-by-one hit the compile button in each file. You can close the file after compiling it.



7. Open SilenceMenuLevel to verify that everything works

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Menu System Actor causes problems with World Partition Maps

Set the Is Spatially Loaded variable on the Actor to False. It will be present all the time after that.

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Release Notes

Upgrade notes for new versions will be added here

Version 2.0.0

- Initial Release
- Released for Unreal Version: 5.0

Version 2.0.1

- New Graphics Settings
    - Global Illumination Method
    - Reflection Method (replaces SSR on/off)
- Created new Settings Data Assets and added them to MenuSystemConfig
    - DA_GlobalIlluminationMethod
    - DA_ReflectionMethod (previously DA_EnableScreenSpaceReflections)
- Added new strings to ST_Menu for the new settings
- New Feature:
    - Auto Save Game Trigger Volume
    - Multi Line Text Input Widget
    - Nvidia DLSS 3
    - FSR2 2.2.1
- Bug fix:
    - Fixed various issues related to Always Pause in Ingame Menu (Timer and Delays) that caused problems with audio sliders etc.
    - Footer Button click fired twice
    - Loading Screen image was stretched
    - Updated unlocked frame rate command
    - Combobox now selects the right item on first selection
    - Continue now selects the latest save game
    - OnPossessed in character is now called for every client
    - Prevented cursor flashing when opening the ingame menu via gamepad
    - Menu transition can now be set to None without destroying the backwards navigation
- Adjusted some error messages and names (BP_MenuSystemActor etc.)
- Released for Unreal Version: 5.0

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Version 2.0.2

- DLSS is now controlled by Resolution Scale (50 = ultra performance, 100 = DLAA)
- Resolution Scale is now ranged to 50-100 and set via console command
- Released for Unreal Version: 5.1

Version 2.0.3

- Bug fix: Restored slider value text to visible size (Scale Box)
- Released for Unreal Version: 5.2

Version 2.0.4

- Bug fix:
    - Save Game Menu: Focus 'None' error when no save game exists
    - Back navigation cooldown issue when the same menu is opened again
    - GridButton text was not visible
    - Input bindings reset required a restart to work
- Released for Unreal Version: 5.0

Version 2.0.5

- Bug fix:
    - Save Game Menu: Focus 'None' error when no save game exists
    - Back navigation cooldown issue when the same menu is opened again
    - GridButton text was not visible
    - Input bindings reset required a restart to work
- Released for Unreal Version: 5.1

Version 2.0.6

- Bug fix:
    - Save Game Menu: Focus 'None' error when no save game exists
    - Back navigation cooldown issue when the same menu is opened again
    - GridButton text was not visible
    - Input bindings reset required a restart to work
- Released for Unreal Version: 5.2

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Version 2.0.7

- Added Menu Container Add Finished event
- Added Menu Transition Finished event
- Improved Integration for other Moonville assets
- Released for Unreal Version: 5.0
- Release Date: 21st Nov 2023
- Changed Files and Upgrade Guide

Version 2.0.8

- Added Menu Container Add Finished event
- Added Menu Transition Finished event
- Improved Integration for other Moonville assets
- Released for Unreal Version: 5.1
- Release Date: 21st Nov 2023
- Changed Files and Upgrade Guide

Version 2.0.9

- Added Menu Container Add Finished event
- Added Menu Transition Finished event
- Improved Integration for other Moonville assets
- Released for Unreal Version: 5.2
- Release Date: 21st Nov 2023
- Changed Files and Upgrade Guide

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)

Version 2.1.0

- *Warning*: This update contains breaking changes regarding Enhanced Input
    - EPIC has made severe changes on the Enhanced Input API, which forced us to change numerous classes in Menu System Pro
    - Player Bindable Input Config has been removed
    - Input User Settings have to be enabled via Project Settings
    - Input Mapping Contexts can be set in the Menu System Config directly
    - Input Actions and Input Mapping Contexts need to be set up differently
    - Gamepad bindings can not be cleared anymore and swap instead when a duplicate is detected
    - Input Mapping Contexts should not contain multiple bindings that point to the same key/button
    - Input Actions should not be re-used in multiple different Input Mapping Contexts
- Bug fix:
    - Fixed slider issues and tweaked speeds for keyboard and gamepad
    - Fixed ComboBox focus issues while navigating over it with down/up
    - Fixed a bug where the Input Icon were not referencing the correct menu container and showed wrong gamepad icons
    - Fixed ScrollBox delay issues by introducing a new function in InteractableMenu
    - Fixed tree material in the silence example level
- Feature:
    - ***New*** Delete save game on selected slot in the Save Game Menu
- Released for Unreal Version 5.3
- Changed files and upgrade guide

Version 2.1.1

- Feature:
    - Added Menu Container Add Finished event to WBP_MenuContainer
    - Added Menu Transition Finished event to WBP_MenuContainer
    - Improved Integration for other Moonville assets
- Bug Fix:
    - Fix input icons for gamepad always showing next to Settings Tabs, even when using Keyboard & Mouse
- Released for Unreal Version: 5.3, later also approved for 5.4
- Release Date: 21st Nov 2023
- Changed Files and Upgrade Guide

Version 2.1.2

- Bug Fix:
    - Scene lighting
    - Menu lag/freeze issue when selecting widgets
- Added an error message that prints when the <MISSING STRING TABLE ENTRY> issue is present. This has to be fixed manually until UE 5.4.2 arrives. See the guide below. **The bug has been fixed in 5.4.2!**
- Released for Unreal Version: 5.4
- Known Issues in 5.4. Click the links for fixes:
    - Missing string table entry
    - Scene lighting is blown out
    - Selecting widgets in menu lags/freezes

Thank you.

If you miss something in the documentation let us know on Discord or Email

THIS IS THE DOCUMENTATION FOR THE NON COMMON UI VERSION (UE 5.0 - 5.5) - Click here for the new Common UI documentation (MSP 2.2.0+)