VR-Menu System Instructions

Please note that this documentation does assume a basic to intermediate understanding of UE and its Blueprint system, and requires UE4.26 or higher.

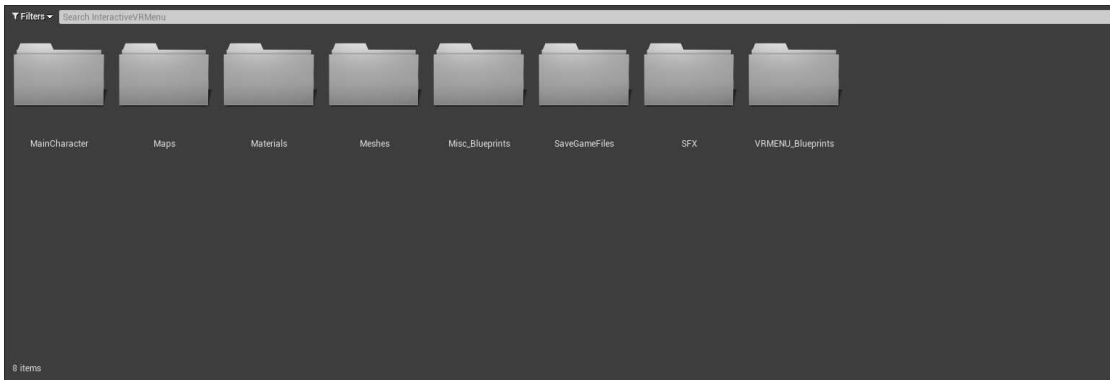
Please also note that in order to function properly this project does require the VRExpansion and DLSS plugins be installed. (VRExpansion plugin is included and can be found in the Interactive VR-Menu system projects plugin folder.)

These files are all currently free to use and can be found here:-

1. https://www.unrealengine.com/
2. https://vreue4.com
3. https://www.unrealengine.com/marketplace/en-US/product/nvidia-dlss

Directories Explained:-

On loading the Interactive VR Menu project you will be presented with the following directory system:-



Files / Folders found can be described as follows:-

1. MainCharacter:- Contains the Main Character Blueprint for the example game.

2. Maps:- Contains all of the maps as well as the individual GM_VREP's needed for each level of the example game.

3. Materials:- Contains all of the Textures and Materials used in the example game.

4. Meshes:- Contains all of the meshes needed for the example Character and game.

5. Misc_Blueprints:- Contains the Blueprints for objects that are used within the levels of the example game.

6. SaveGameFiles:- Contains all of the files needed for the Save Game system to work.

7. SFX:- Contains all of the Audio files needed for the example game to function.

8. VRMENU_Blueprints:- Contains all of the Blueprints needed for the VR Menu System to operate.

Basic VR-Menu Setup

The ‘Interactive VR-Menu System’ comes supplied with a complete demo game which you are free to use for yourself and try and build upon.

However, if you do wish to add the Menu system to your own VR Project then please use the following instructions:-

In the ‘VRMENU_Blueprints’ folder you will find the ‘BP_VRMenuSystem’ Blueprint. This will need to be added to your own VRCharacter Blueprint in order to function.

To see an example of this you can look at the supplied VRCharacter Blueprint called ‘MainCharacter_C_VREP’ which is found in the ‘MainCharacter’ folder. There you will see that the ‘BP_VRMenuSystem’ Blueprint has been added as a Child Actor.

PLEASE NOTE:- When migrating the VR_Menu System over to your own VR project please ensure that your own VRCharacter Blueprint is named ‘MainCharacter_C_VREP’, and create your own duplicate directory (Content\InteractiveVRMenu\MainCharacter) folder to place it in also.

Remember NOT TO overwrite this very important file during the actual migration.

Following these simple steps will then allow the ‘BP_VRMenuSystem’ Blueprint to be attached to your own VRCharacter Blueprint without any problems.

On first loading your game you will want to Initialize the Intro Menu. To do this simply copy the code from the ‘BP_VRMenuSystem’ Blueprint ‘START LEVEL Specific’ Graph, and then link to it from your own VRCharacter Blueprint also.

The VR Menu will store the following files when saving / loading the game settings:-

1. Audio Settings.
2. Controller Settings.
3. Graphic Settings.
4. Hand Type Settings.
5. Language Settings.

These files will then either be stored in the C:\Users\UserName\AppData\Local\AppName or the UE4 Projects Saved\SavedGames folder depending on whether the project that you are running has been built or not.

PLEASE NOTE:- When displaying the Intro Menu the VR-Menu System will automatically detect if there are any Controller Settings saved, and if not it will then show the Controller Settings page, so that the player will have to enter their actual controller settings on Start-up.

Any other files that have not already been created will automatically be created and then stored by the system using the default settings. What the default settings actually are can easily be changed by simply opening the corresponding SaveGame Class Blueprints found within the SaveGameFiles folder. (However, changing the GraphicSettings file Defaults will require you to change a few string variables as well, so please see:- 3. Graphic Settings for further details.)

There is also a ‘RESET Game Instance’ Custom Event which can be found in the BP_VRMenuSystem’s event graph which reset’s / defines the MyGameInstance start settings.

1. Audio Settings:-

SoundVolumes variable:- This float array contains 3 settings all currently set to 1 (which is full volume). Float 0 is the MUSIC setting, float 1 is the EFFECTS setting and float 2 is the AMBIENT setting.

This folder also contains the Sound Class Mix file which will need to be added to your own ‘Default Base Sound Mix’ slot found in Project Settings.

Also located here are the SFX_Ambient, SFX Effects, and SFX Music Sound Class files, which will need to be selected in your own Cue files in order to say exactly which Sound Class your sound effect files will belong to. (Examples of this can be found in the SFX folder also).

2. Controller Settings:-

LeftHanded? Variable:- Default is currently Right-Handed (un-ticked). ControlSettingsCurrent:- The EControllerTypes currently contains Knuckles, Touch, Wands, Cosmos and Mixed Reality. Default is currently Knuckles.

Other Enums found in this folder are:- ELEFTActionBTM, ELEFTActionTOP, ELEFTGrip, ELEFTThumbstick, ELEFTTrigger, ERIGHTActionBTM, ERIGHTActionTOP, ERIGHTGrip, ERIGHTThumbstick, and ERIGHTTrigger. These are used to correctly describe the current controllers inputs and are kept constantly updated along with:- Male or Female, Left or Right Handed, Language Settings Current, VR MENU Text Onscreen, START LEVEL MENU on? VR MENU on? SOLO MENU on? also.

The Blueprint code that keeps these updated can be found on the VRMENU Specific graph in the example ‘MainCharacter_C_VREP’, and the entire code found within this graph (as well as the code that is commented in green on the EventGraph) will need to be incorporated into your own project also.

Please also be sure to view the CONTROLS Specific graph from the ‘MainCharacter_C_VREP’ as this will provide you with the preferred controls / action buttons setup also.

3. Graphic Settings:-

Each of the Graphic Command string variables in the GraphicSettings file need to correspond with the selected default values from each of the relevant EGraphicsSettings also (Low, Medium, High, Epic or Cinematic.)

To find the correct string for each individual categories Graphic Enum Settings please refer to the BP_GFXSettings Blueprint's 'ApplyButton GRAPH', which can be found in the VRMENU_Blueprints\GFXMenu\GFXSettingsMenu\ folder.

DLSS Settings have no corresponding command that needs to be entered and can simply be selected from the Default Value Menu (Ultra Quality, Quality, Balanced, Performance or Ultra Performance also.)

EGraphicCategories can be found in this folder also, and is mainly used by the actual Graphics Settings Menu.

4. Hand Type Settings:-

GenderMale? variable:- Default is Male (Ticked)
SkinTypeSettingsCurrent:- The available settings here are Dark, Medium and Light.

The default is currently set to Light.

5. Language Settings:-

LanguageSettingsCurrent:- The available settings here are English, French, German, Spanish and Italian.

The default is currently set to English.

Save Game System

1. Save Game:-

ELevelNames contains all of the available level names (be sure to exactly match these to the names of your actual maps/levels or they will fail to properly load), and ECheckpointNumbers contains all of the available Check point Numbers.

The following variables are fixed variables and should not be changed.

1. Level Name Current – The Current Levels Name.
2. Checkpoint Number – The Current Checkpoint number.
3. Player Location – Needed for restoring Checkpoint position.
4. Player Rotation – Needed for restoring Checkpoint rotation.
5. Time Stamp – Created at the time the game is saved.
6. EOL Save? – If this is set the system will know it needs to load the next level.

Other changeable variables found within the Save Game file are:-

1. Number of Cubes – How many Cube pickups the player currently has.
2. Number of Spheres – How many Sphere pickups the player currently has.
3. Number of Cylinders – How many Cylinder pickups the player currently has.
4. Number of Pyramids – How many Pyramid pickups the player currently.

(These should be the entire games pickup items and can be altered / changed as you see fit.)

PLEASE NOTE:- When adding to or changing the Level names in the ELevelNames Enum, they will also need to be added to the 'Load Next Level' function found within the BP_VRMenuSystem also.

When loading a Level the correct Level name will need to be cast to the MyGameInstance file from the Level Blueprint also. (See earlier level examples.)

Also when adding or changing level names they will need to be added to each Load and Save Slots NAME SLOT Functions. These Blueprints can be found here:-

1. VRMENU_Blueprints\LoadMenu\LoadMenuSLOTS
2. VRMENU_Blueprints\SaveMenu\SaveMenuSLOTS

2. **MyGameInstance:-** This file contains all of the information needed for the levels in the example game to run and should be added to the Game Instance slot in the Project Settings.

The following variables are fixed variables and should not be changed.

1. Level Name Current – The Current Levels Name.
2. Checkpoint Number – The Current Checkpoint number.
3. Player Location – Needed for restoring Checkpoint position.
4. Player Rotation – Needed for restoring Checkpoint rotation.

Other changeable variables found within the My Game Instance file are:-

1. Number of Cubes – How many Cube pickups the player currently has.
2. Number of Spheres – How many Sphere pickups the player currently has.
3. Number of Cylinders – How many Cylinder pickups the player currently has.
4. Number of Pyramids – How many Pyramid pickups the player currently.

(These should be the entire games pickup items and can be altered / changed as you see fit.)

As you can see the MyGameInstance file mirrors the SaveGame file entries with the exeption of the time stamp and EOL save?.

PLEASE NOTE:- When changing / adding pickups to your own Save Game / Game Instance files always remember to add them to the ReturnInstanceInfo and SetInstanceInfo functions found within also, as these will also need to be added to the BP_VRMenuSystem in order for it to correctly function (See the Load / Save Game Graph for previous examples.)

Inventory System

The Inventory System (BP_Inventory) can be found in the VRMENU_Blueprints\InventoryMenu folder.

When adding objects to the inventory you will need to know the number of objects that you have in total. In our own example we have 4 objects, and the wrap around points will also need to be set in the Increase or Decrease Item Number function.

Look at the Select Correct Inv Item function to see how Inventory Items / Item Numbers should to be added in to the system. (Remember to add Static Meshes of the object and the correct names / translations also.)

You will then also need to add your items in to the USE ITEM? Function (remember to use the correct inventory item number!) This function allows the objects to be usable or unusable depending on how the relevant bool is set (Default is Unusable / Off). **This allows you to control when a player can use an object, for example you can set the bool from the level blueprint when the player has entered a trigger volume in front of a door by casting from the Level Blueprint to the Player Blueprint / VRMENU_System blueprint and then finally on to the Inventory Blueprint itself also.**

Once the item has been used you can then cast back to your own player blueprint or wherever else you may feel like via the items designated item USED function. (These are commented in Green and are as yet undefined currently).

PLEASE NOTE:- This is basically a bare bones Inventory System and we do expect that it will need to be altered slightly depending on any individual projects requirements.

And Finally…

You can change the type of voice used for the menu by changing the default value of the MenuVoiceType Variable found in the BP_VRMenuSystem and there are currently four different voice types available.

Please also note that in order for the collisions to work between the colliding objects and VR-Menu Buttons, as is shown in the example project, world static / overlap collision settings will need to be applied.

While we feel that these instructions should be comprehensive enough for most people with a basic to intermediate understanding of Blueprints, if you do require any further help at all, then it is available via our subscription based Patreon / Discord Chat room:-

https://www.patreon.com/TriggerfishStudios

Thank you!