nDisplay Documentation

What's in this document:

Setup

This setup guide will cover creating a nDisplay working environment with one editor node, one render node and a remote interface. One of the most compelling features of nDisplay is the ability to edit scenes in real time from multiple sources with relative ease, this gives maximum flexibility to directors, DOPs and other creative personnel on set to interact with the scenes with minimal disruption to shoots. This is added to by the ability to expose any variable to a web interface available to any device with a browser on the network.

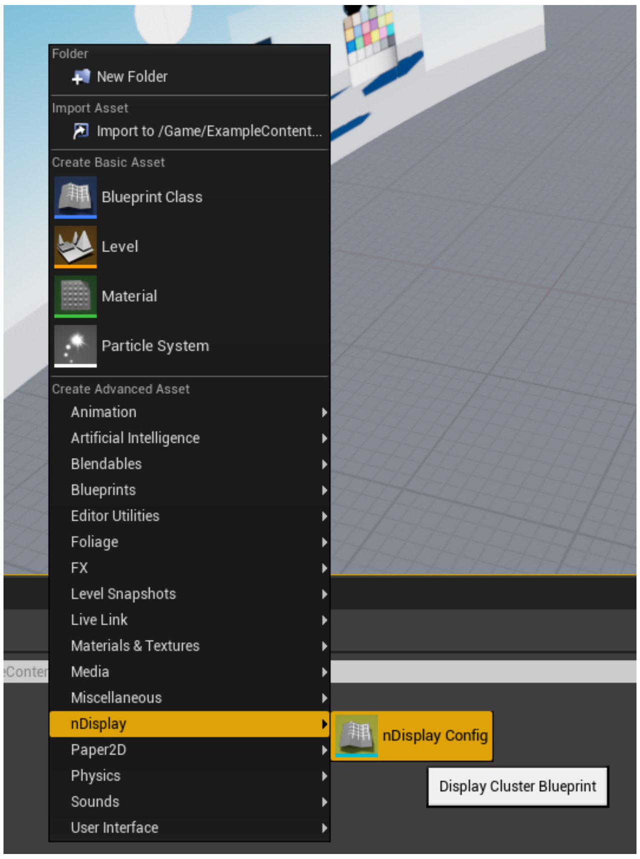
1. Setting up a scene from scratch (nDisplay configuration).
2. Configuring the switchboard.
3. Joining a multi-user session.
4. Setting up a remote interface.

1. Setting up a scene from scratch (nDisplay configuration)

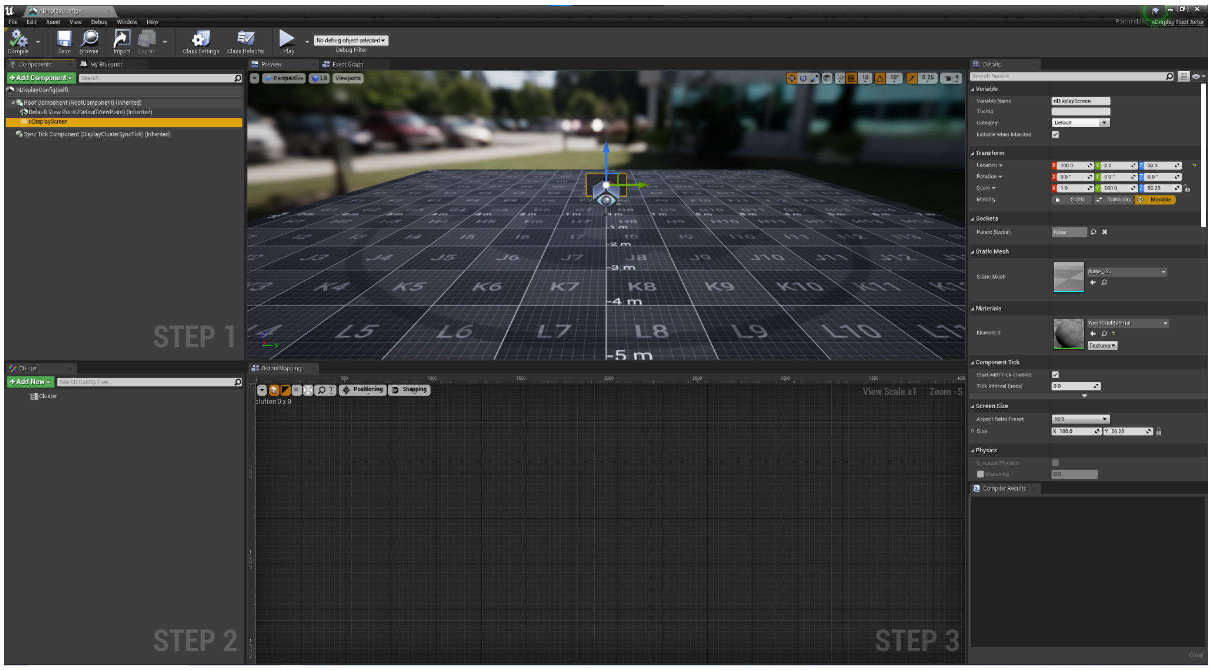
Implementing an nDisplay scene can be done relatively simply, in this example I'll discuss setup such any UE4 scene provided can be used with nDisplay. So to get started create a new games project with the in camera VFX template. Check the list of required plugins here, but the main ones to ensure are enabled are:

- nDisplay
- Live Link over nDisplay
- Remote Control API
- Remote Control Web Interface

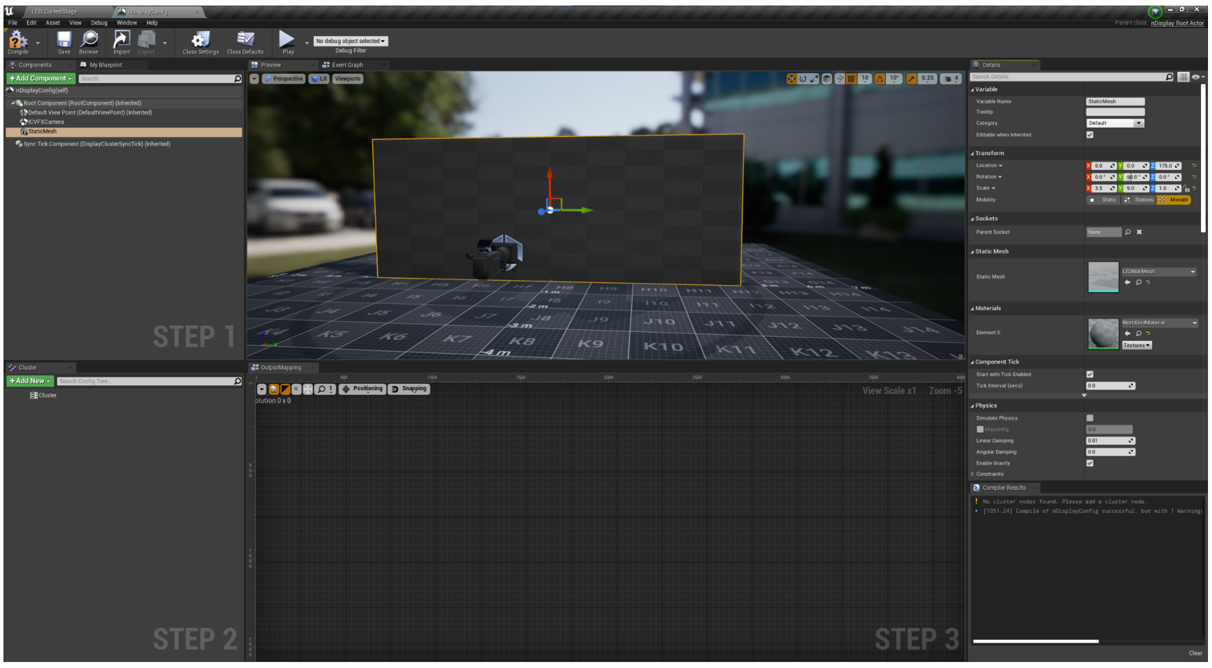
With these plugins enabled and active, create an nDisplay configuration file (RC in content browser > nDisplay > nDisplay Config).



Select the option to create a new nDisplay configuration file and press finish. Open this to find the nDisplay Configuration Editor page, new in UE4.27, edits to all things to do with the LED wall's virtual representation can be made here.



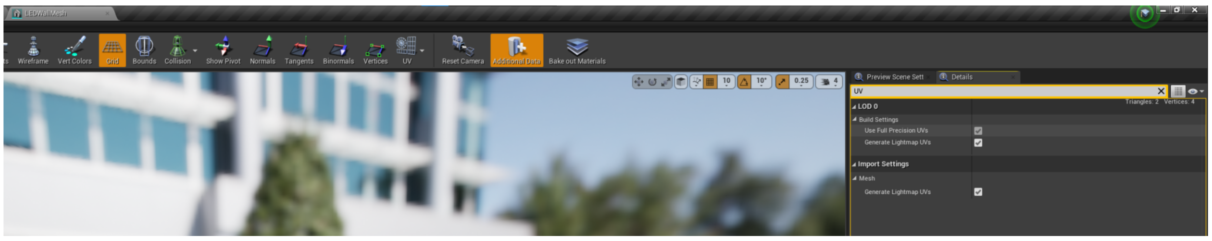
Start by removing the default wall mesh generated with the new config file and add a ICVFXCamera component. Add an nDisplayScreen or a static mesh component to the config, if using static mesh set it to either a plane, or a model representative of the wall in the studio. If using a basic unreal plane here be sure to create a copy of the static mesh (As the plane may be in use elsewhere and cannot be saved) and enable the Full Precision UV mode.



nDisplayConfig(self)
Root Component (RootComponent) (Inherited)
Default View Point (DefaultViewPoint) (Inherited)
ICVFXCamera
StaticMesh
Sync Tick Component (DisplayClusterSyncTick) (Inherited)

Variable	
Variable Name	StaticMesh
Tooltip	
Category	Default
Editable when Inherited	✓
Transform	
Location	X 0.0 Y 0.0 Z 175.0
Rotation	X 0.0° Y 90.0° Z 0.0°
Scale	X 3.5 Y 9.0 Z 1.0
Mobility	Static / Stationary / [✓ Movable]
Sockets	
Parent Socket	None
Static Mesh	
Static Mesh	LEDWallMesh
Materials	
Element 0	WorldGridMaterial
Component Tick	
Start with Tick Enabled	✓
Tick Interval (secs)	0.0
Physics	
Simulate Physics	
MassInKg	0.0
Linear Damping	0.01
Angular Damping	0.0
Enable Gravity	✓
Constraints	

```
! No cluster nodes found. Please add a cluster node.
• [1051.24] Compile of nDisplayConfig successful, but with 1 Warning(
```



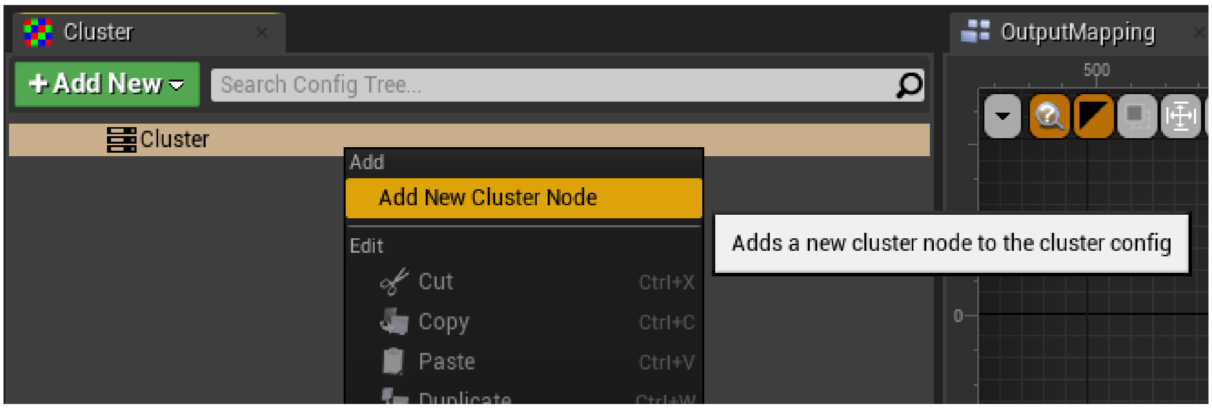
UV

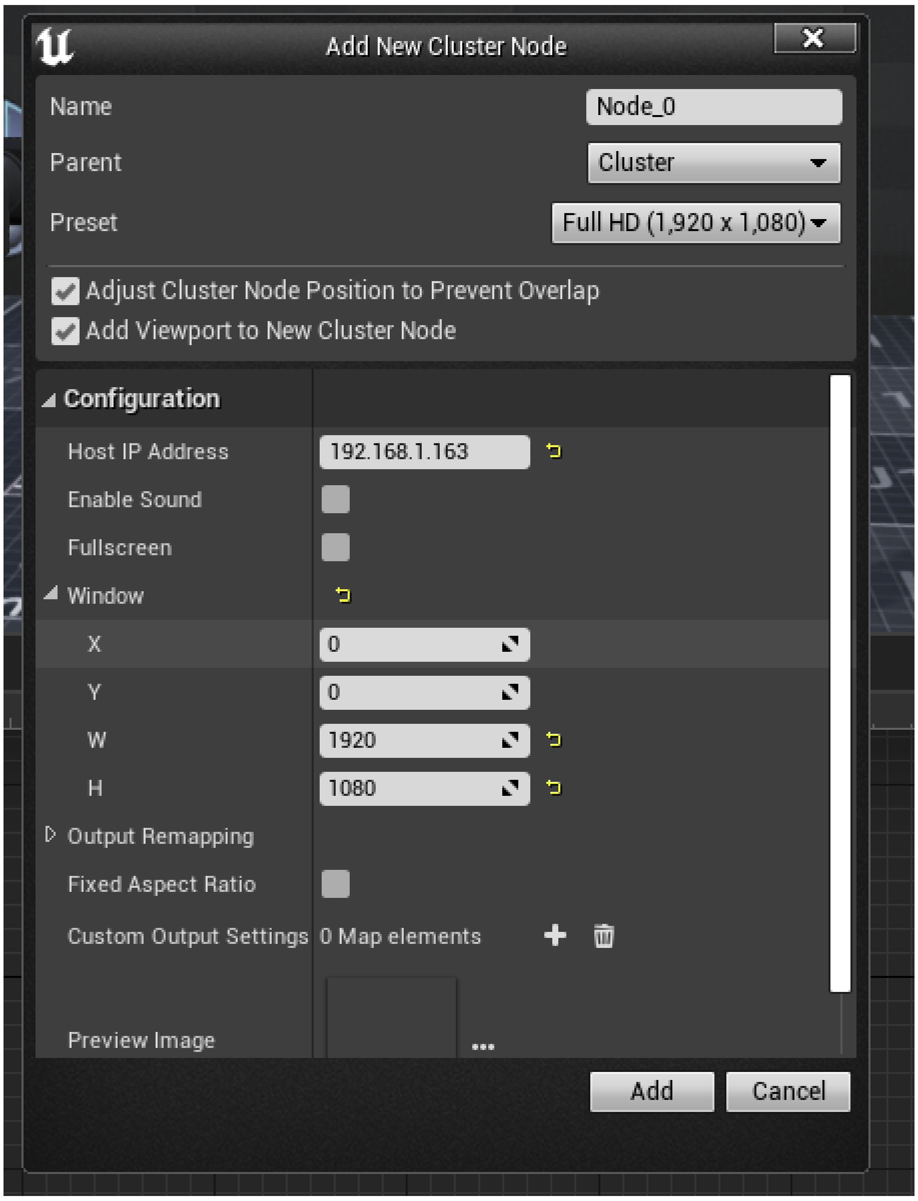
Triangles: 2 Vertices: 4

LOD 0	
Build Settings	
Use Full Precision UVs	✓
Generate Lightmap UVs	✓
Import Settings	
Mesh	
Generate Lightmap UVs	✓

Position the mesh or screen relative to where the origin camera tracking will be (0,0,0 in the viewport will be 0,0,0 for the tracking source). Set the default view point to a sensible position central to the screen.

Now in the cluster section create a new viewport.




Add New Cluster Node

Name	Node_0
Parent	Cluster
Preset	Full HD (1,920 x 1,080)

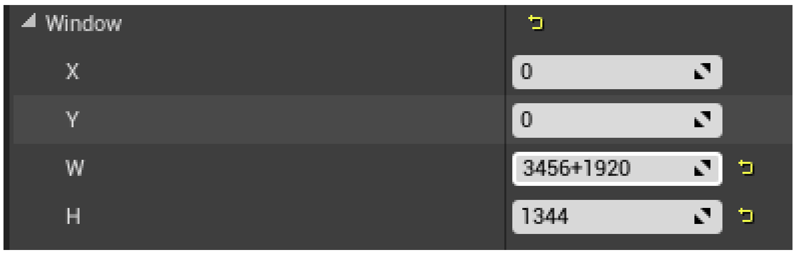
✓ Adjust Cluster Node Position to Prevent Overlap
✓ Add Viewport to New Cluster Node

Configuration

Host IP Address	192.168.1.163
Enable Sound	
Fullscreen	
Window	
X	0
Y	0
W	1920
H	1080
Output Remapping	
Fixed Aspect Ratio	
Custom Output Settings	0 Map elements
Preview Image	

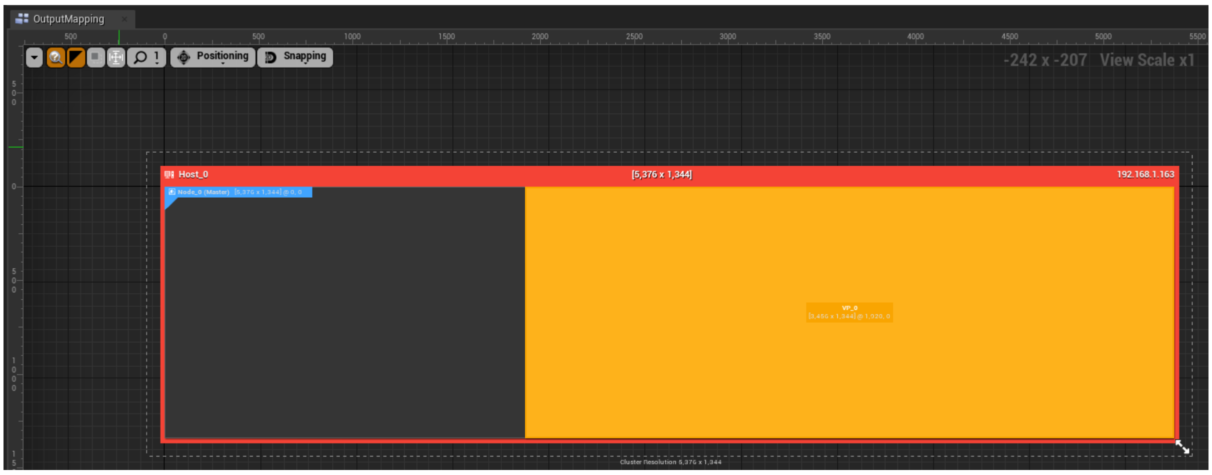
Set the parent node to the IP address of the computer which shall be rendering the image to the wall. Set the resolution preset which will be used by the backplate of the LED wall (I've set it to 1920x1080 here but the wall at the stage is 3456x1344). It's worth noting as well that if the LED wall is acting as the 2nd monitor on the node: 1) Fullscreen won't work properly 2) The window size specified here must include the dimensions of the first monitor and the relative OS position of the LED wall from the first monitor must be specified in the region, for example: If the first monitor is 1920x1080 and the wall is 3456x1344 the window and viewport settings would need to look like this:

In Node_0 (or whatever you call the render machine)



Window	
X	0
Y	0
W	3456+1920
H	1344

In VP_0 (or whatever you call the viewport)

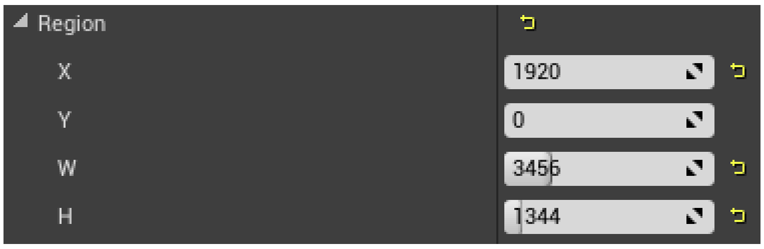


Host_0 [5,376 x 1,344] 192.168.1.163

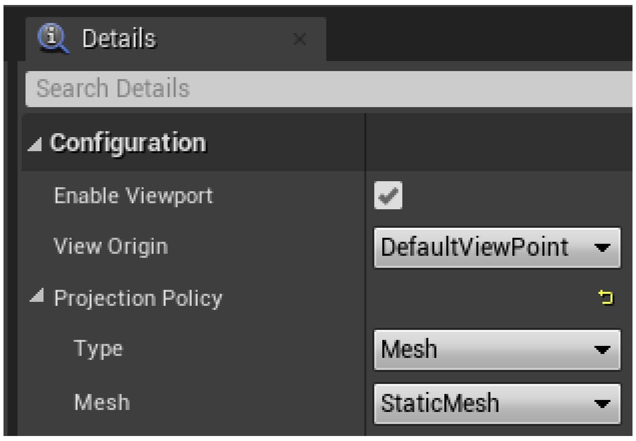
Node_0 (Master) [5,376 x 1,344] @ 0, 0

VP_0 [3,456 x 1,344] @ 1,920, 0

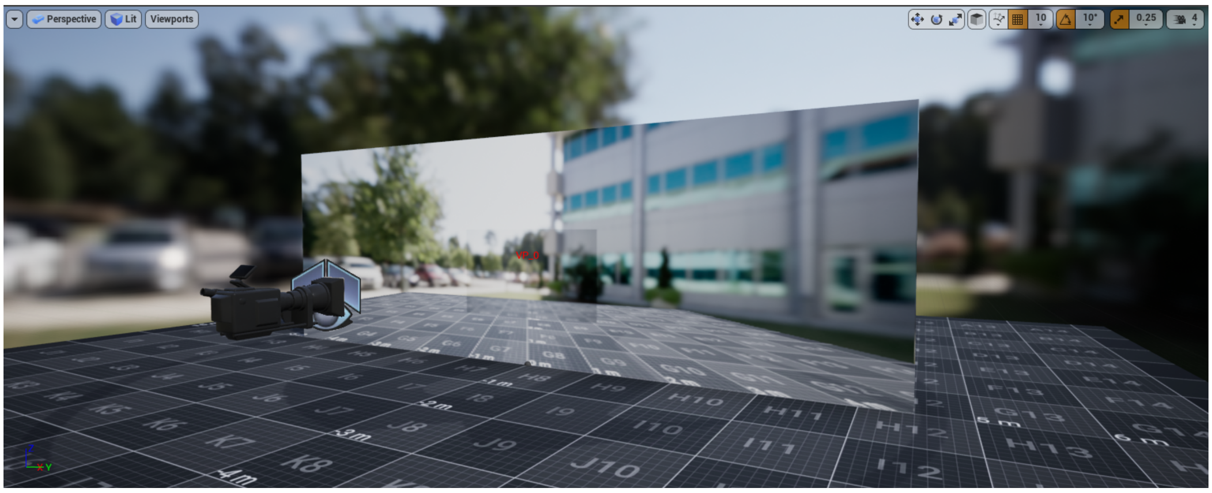
Cluster Resolution 5,376 x 1,344


Region
X 1920
Y 0
W 3456
H 1344

In the projection policy section in the viewports details set the type to mesh and then specify the static mesh.

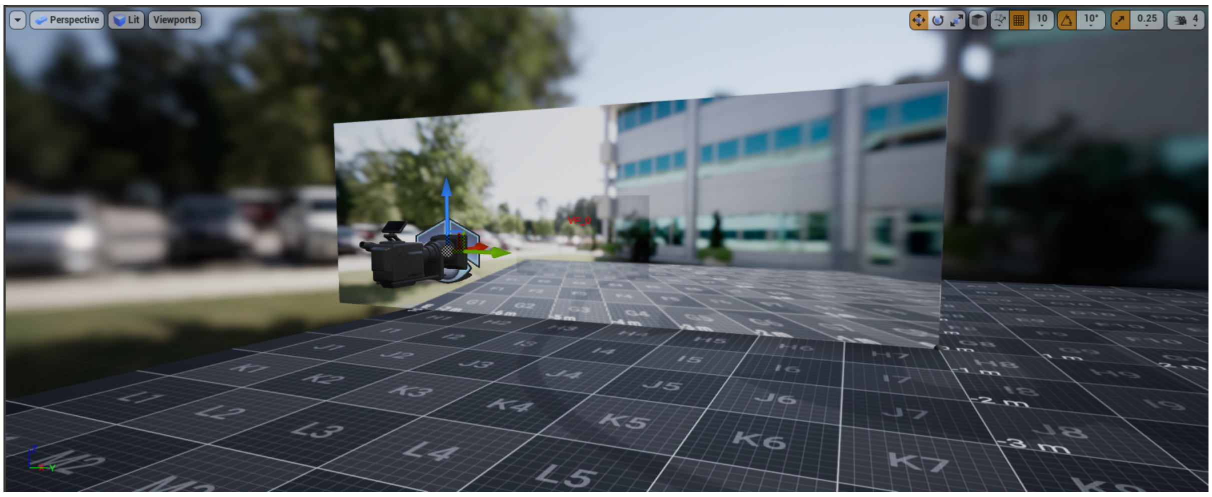

Details
Search Details
Configuration
Enable Viewport ✓
View Origin DefaultViewPoint
Projection Policy
Type Mesh
Mesh StaticMesh

Your config file should now look something like this:

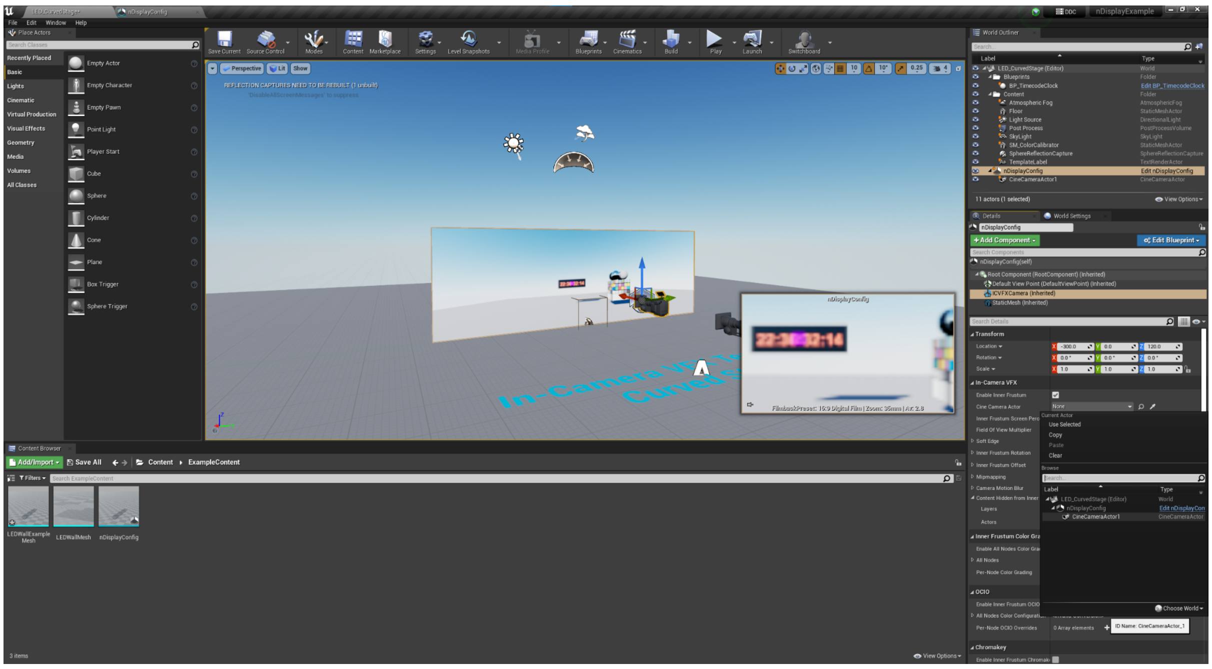


Compile and save this set up.

It’s worth noting that your life can be made easier by lining up the origin of your tracking source with the origin of the config file, for example in the Guildford studio we placed the Optitrack origin against the exact center of the bottom of the LED wall, so the set up shown above would align the worlds if the tracked camera was parented and zeroed with the config instance in the scene. Another example is when using NCam tracking we align the origin with the bottom right corner of the wall, so a setup like this would be more useful here:



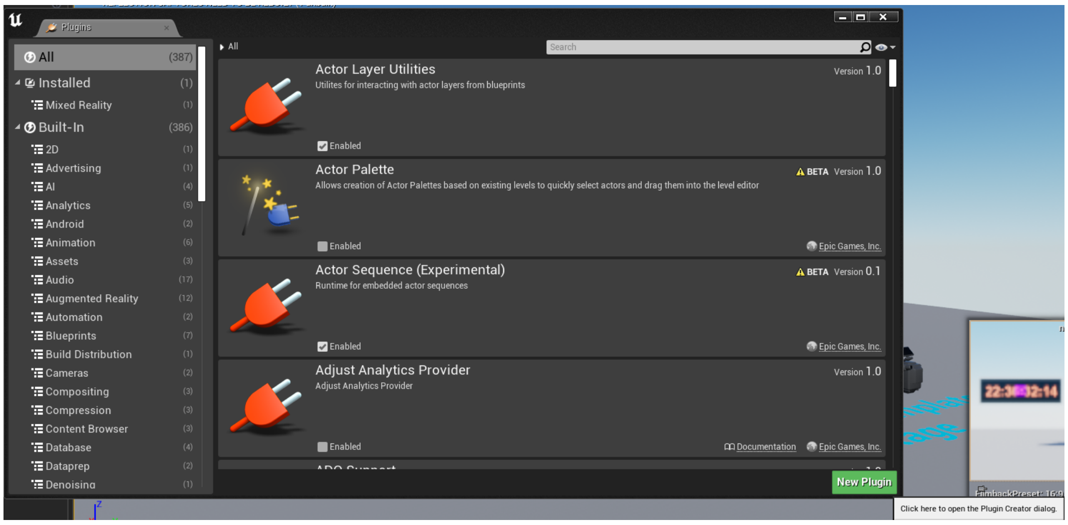
The basics of the configuration are now done and ready to deploy to a scene.



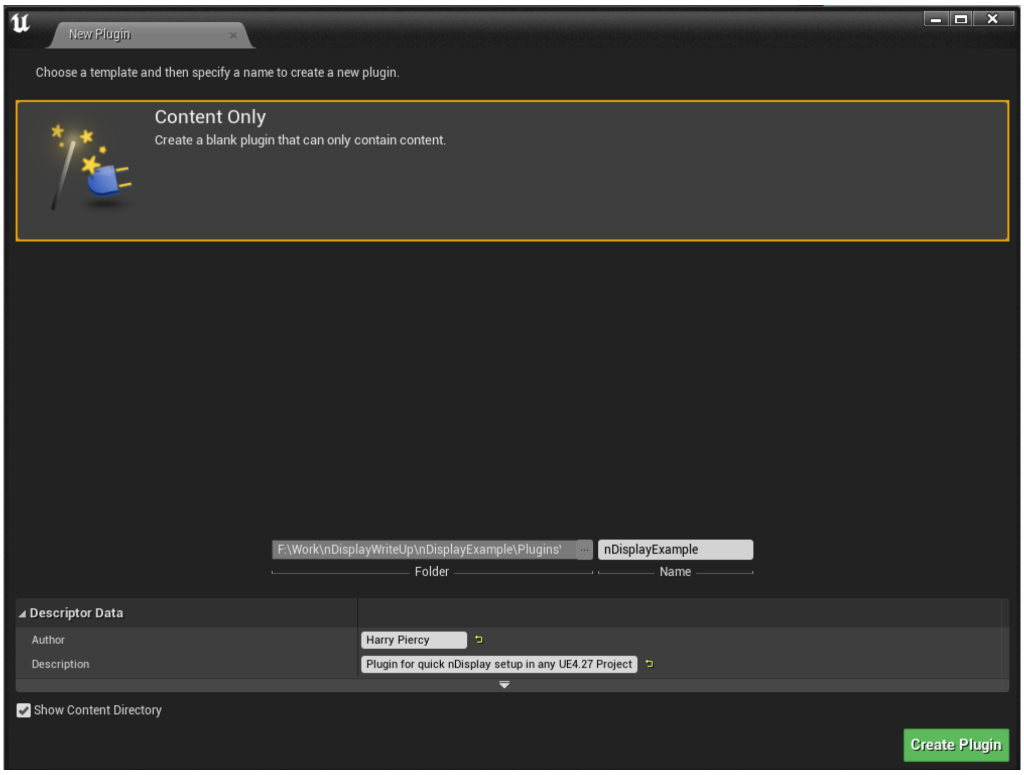
Drag the nDisplay config file into the scene and set the cine camera variable of the ICVFX camera component to a cine camera in the scene to select the camera as the chosen perspective. This will be the tracked camera but I won't go into setting up camera tracking in this write up.

Now to make this nDisplay config work for any UE4.27 project you wish to use it with, I'll do this by creating a simple plugin. The purpose of the plugin is to ensure the engine environment is suitable for nDisplay (correct plugins are enabled), and then to provide premade content which can be dropped directly into the scene.

Firstly navigate to the plugins page and select the "new plugin" button



Setup the new plugin with whatever information you want and hit create plugin.


New Plugin
Choose a template and then specify a name to create a new plugin.
Content Only
Create a blank plugin that can only contain content.

Folder: F:\Work\nDisplayWriteUp\nDisplayExample\Plugins\
Name: nDisplayExample

Descriptor Data
Author: Harry Piercy
Description: Plugin for quick nDisplay setup in any UE4.27 Project

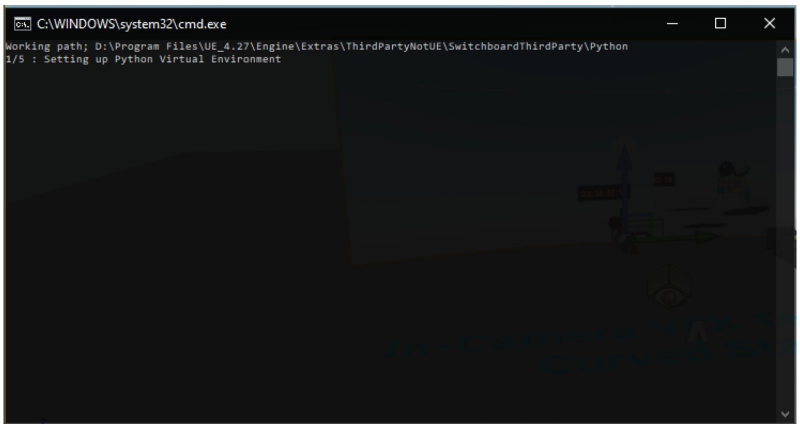
Show Content Directory ✓

This Plugin will now be available in the plugins menu. All of the content previously created be added to the plugin along with the list of dependent plugins listed at the start of this section. Once complete this plugin can be added locally or to the root engine plugins folder so it appears in any new UE4.27 project you make.

2. Configuring the switchboard

So the next step of the process is to set up a switchboard. This is Unreals external launcher and control hub for nDisplay cluster projects. From here you can launch a project across multiple machines.

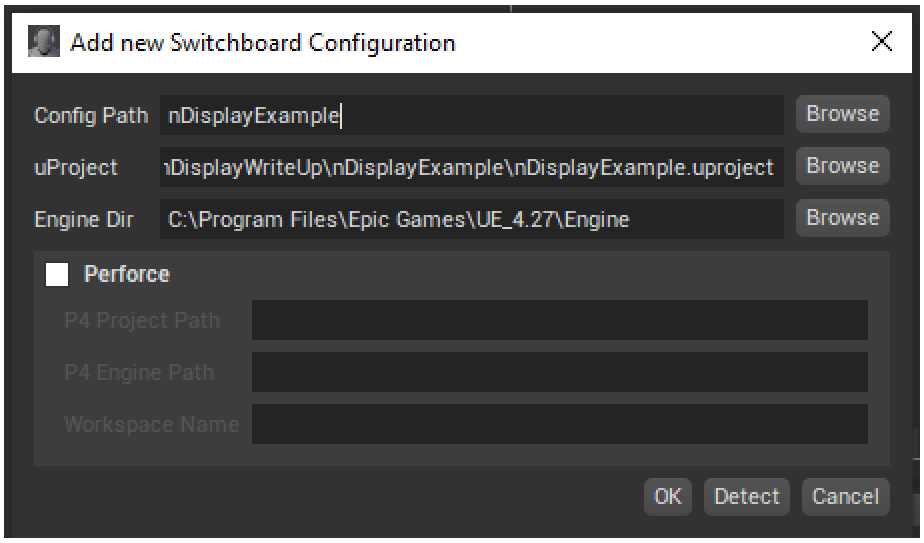
With the plugin enabled in your project you should be able to see it in the main toolbar, select this to open the switchboard (the first time a computer uses this it will need to download some elements such as python and may take a while).



```
Working path; D:\Program Files\UE_4.27\Engine\Extras\ThirdPartyNotUE\SwitchboardThirdParty\Python
1/5 : Setting up Python Virtual Environment
```

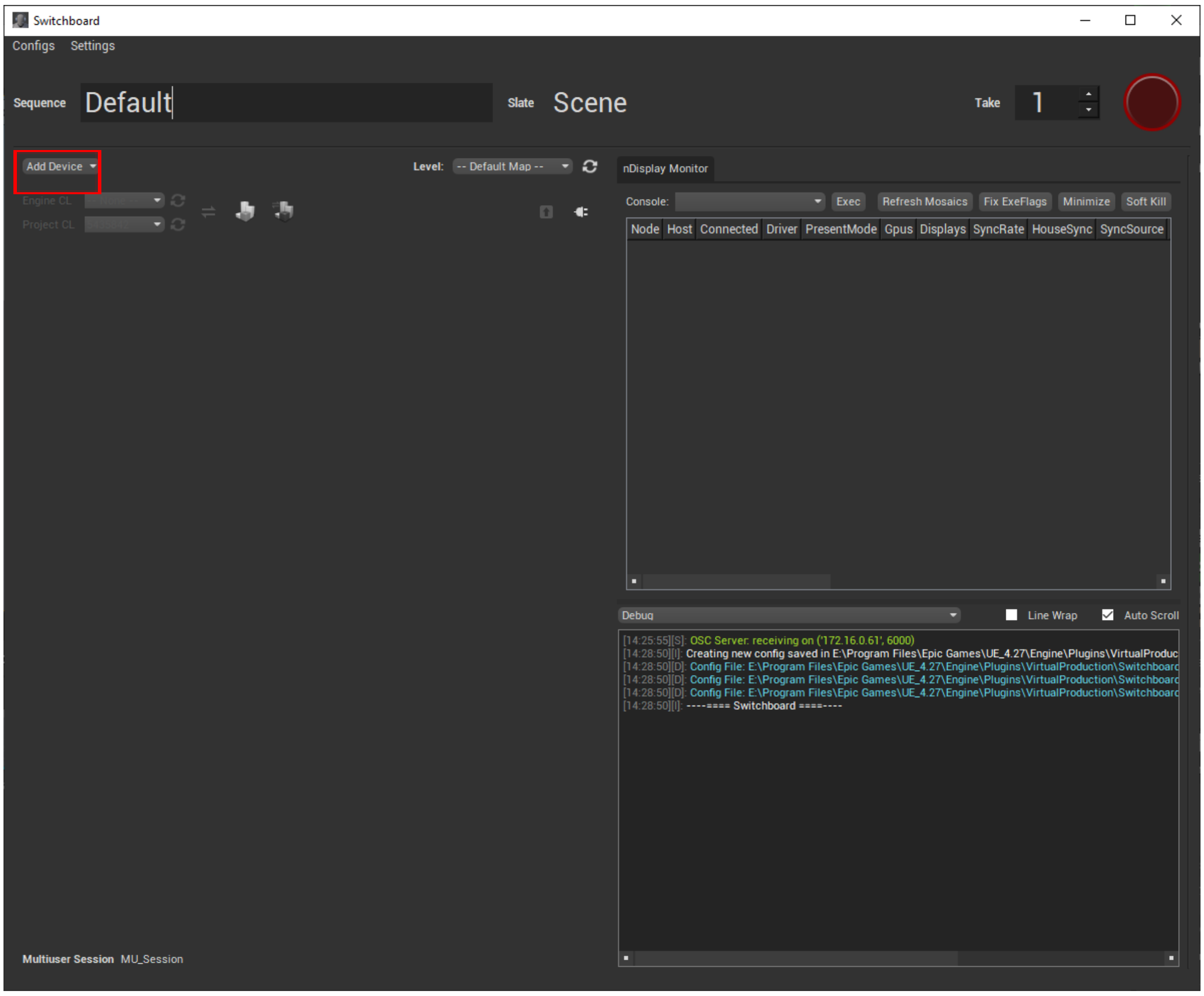
Once that's all installed a timesaving idea could be to create a desktop shortcut to the switchboard launcher so an UE4 editor doesn't need to be opened each time a workload is launched across the cluster (we've done this on our FOH editor in Guildford), the required WBF can be found at this path:
PathToUE_4.27ProgramFolder\Engine\Plugins\VirtualProduction\Switchboard\Source\Switchboard\switchboard.bat

Back in the editor (or from the new shortcut) launch the switchboard and select your current project (will be self populated if launching from the editor) then press OK.



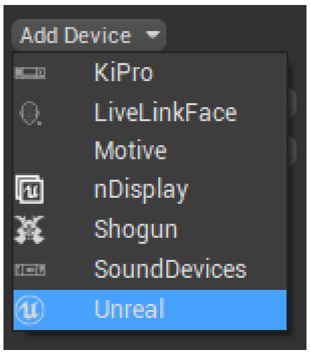
Config Path: nDisplayExample
uProject: ıDisplayWriteUp\nDisplayExample\nDisplayExample.uproject
Engine Dir: C:\Program Files\Epic Games\UE_4.27\Engine

The switchboard creates .json configuration files with all the information it needs about each node and communicated over the network using OSC messaging.

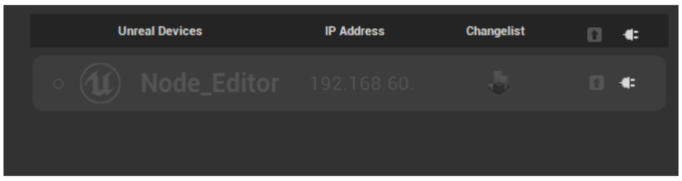


```
[14:25:55][S]: OSC Server: receiving on ('172.16.0.61', 6000)
[14:28:50][I]: Creating new config saved in E:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduc
[14:28:50][D]: Config File: E:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[14:28:50][D]: Config File: E:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[14:28:50][D]: Config File: E:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[14:28:50][I]: ----==== Switchboard ====----
```

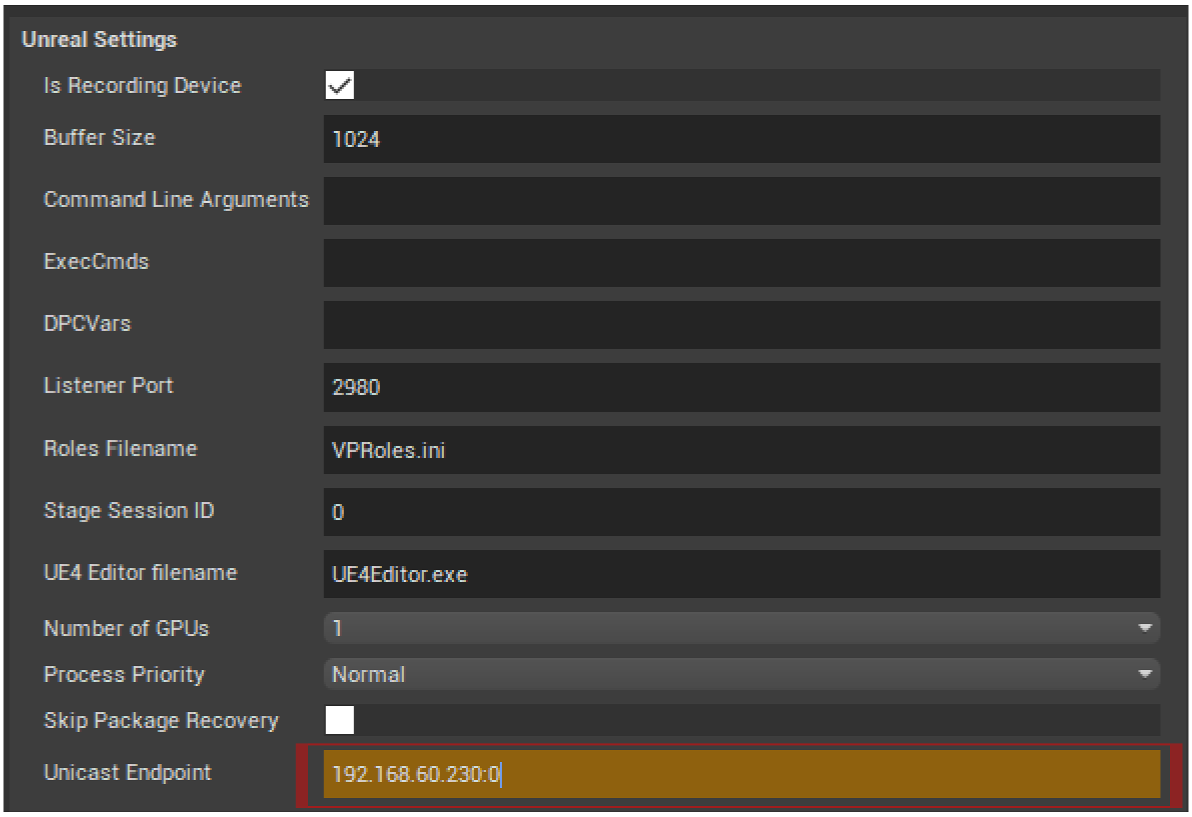
Once launched the switchboard will open to this page. First thing to get started with this is to add an editor device to the switchboard:



Enter the name and the ip address of the machine to be used as the live editor node, something like this should then be visible:



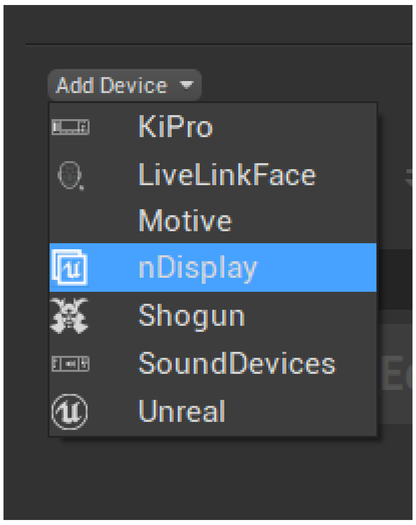
Open up the settings now and find the editor node settings, here the unicast endpoint needs to be set, this is the same as setting it in the project settings but setting it here can save time when syncing projects across multiple machines. The unicast endpoint must be set to receive both LiveLink data and enabling multi-user sessions.

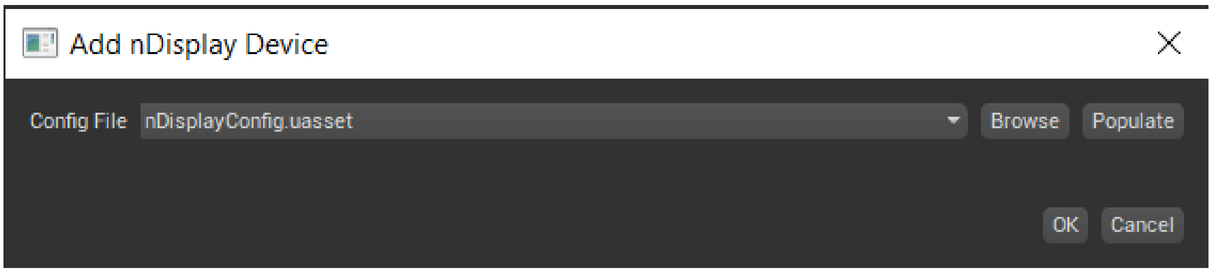


Unreal Settings

Is Recording Device	✓
Buffer Size	1024
Command Line Arguments	
ExecCmds	
DPCVars	
Listener Port	2980
Roles Filename	VPRoles.ini
Stage Session ID	0
UE4 Editor filename	UE4Editor.exe
Number of GPUs	1
Process Priority	Normal
Skip Package Recovery	☐
Unicast Endpoint	192.168.60.230:0

Now setup the nDisplay node:





Add nDisplay Device

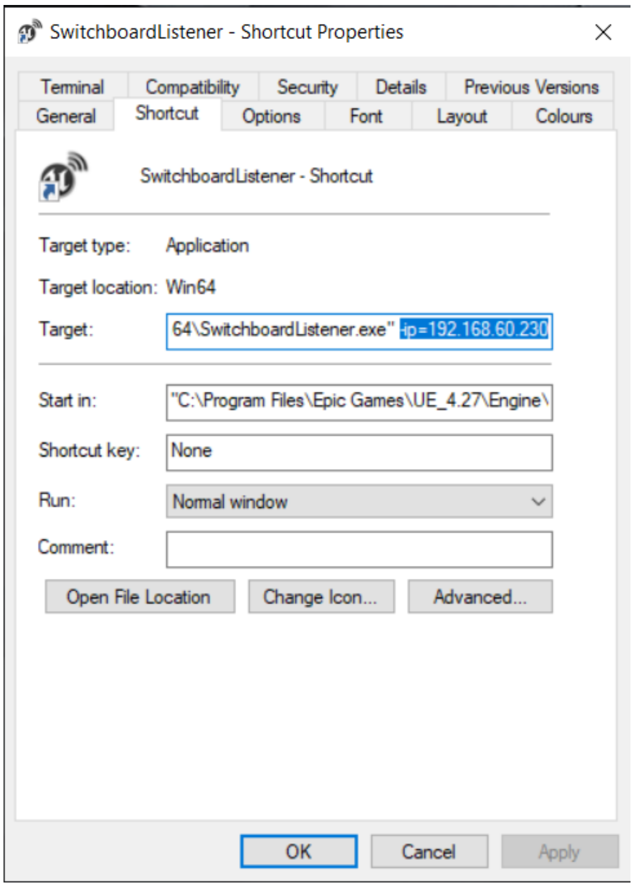
Config File: nDisplayConfig.uasset

If launching from the project, the populate button will automatically locate config files and add them to the drop down, if launching switchboard from the desktop shortcut this will need to be manually located. Once this is set ensure the IP matches that of the machine to be used as the render node and set the unicast endpoint in the same way as the editor node.

Now set up the switchboard listener, again this can either be done via the editor, or by locating the SwitchboardListener application at:

PathToUE_4.27ProgramFolder\Engine\Binaries\Win64\SwitchboardListener.exe

Then create a shortcut and add the launch parameter “-ip=[ipInterface]”, this opens the specified interface to receive messages from the machine running the switchboard.



SwitchboardListener - Shortcut Properties

SwitchboardListener - Shortcut

Target type: Application

Target location: Win64

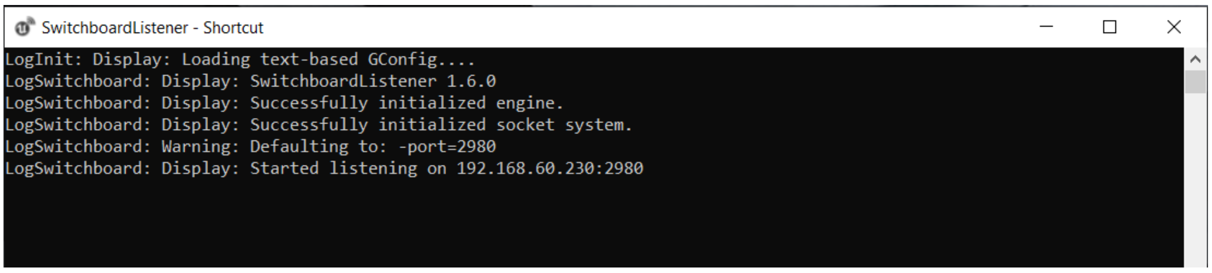
Target: 64\SwitchboardListener.exe" -ip=192.168.60.230

Start in: "C:\Program Files\Epic Games\UE_4.27\Engine\

Shortcut key: None

Run: Normal window

Comment:

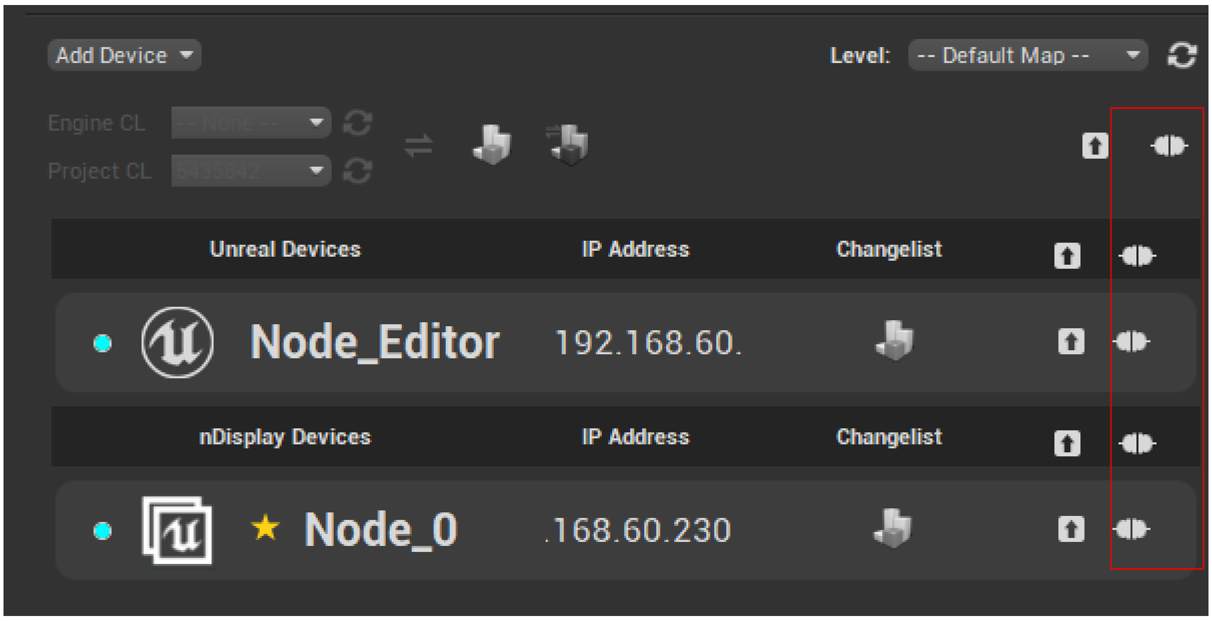


SwitchboardListener - Shortcut

```
LogInit: Display: Loading text-based GConfig....
LogSwitchboard: Display: SwitchboardListener 1.6.0
LogSwitchboard: Display: Successfully initialized engine.
LogSwitchboard: Display: Successfully initialized socket system.
LogSwitchboard: Warning: Defaulting to: -port=2980
LogSwitchboard: Display: Started listening on 192.168.60.230:2980
```

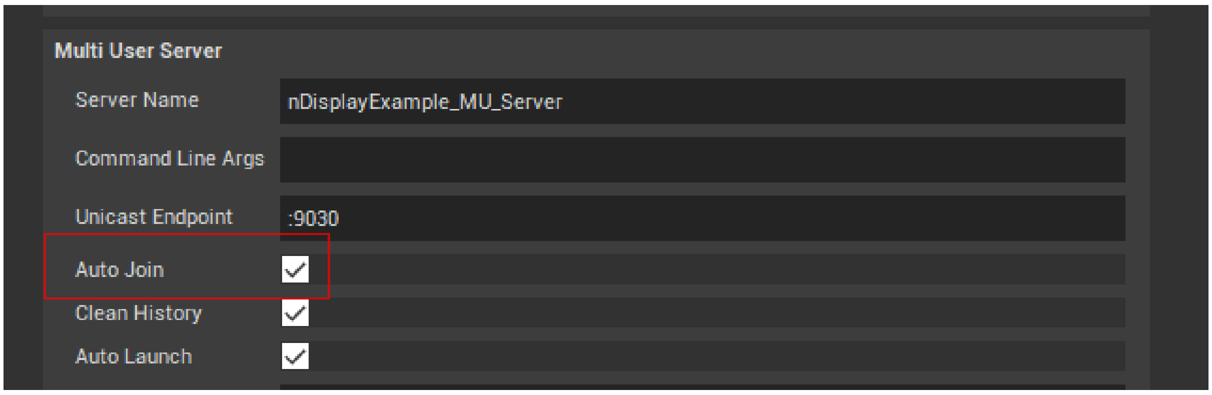
Repeat this on each machine that you wish to launch the project with via switchboard with their own ip addresses (if not already done this would be a good time to assign static ip addresses to the network adapter on the machine to avoid repeating these steps for each project).

With that done a connection between the switchboard and each machine should be establishable



Once connected the projects should be launchable, resulting in an instance of the Unreal Editor opening on the editor node and a nDisplay project running on the render node.

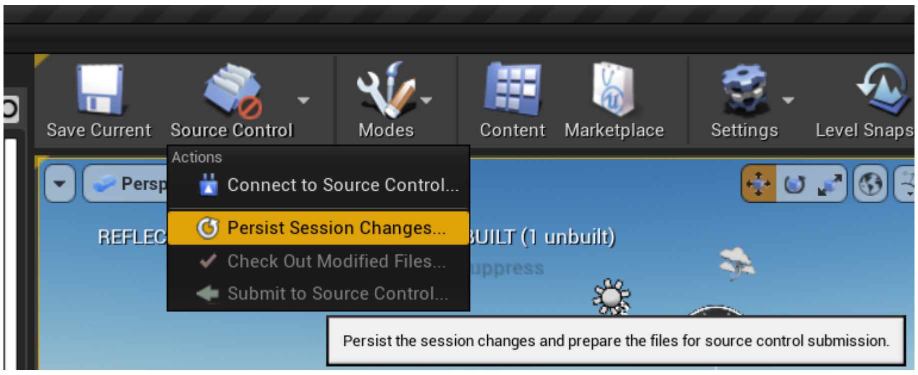
The final step is bridging these two instances of the project together via a multi-user session. Back in the switchboard open up the settings again and make sure the Auto Join option in the multi-user session section is enabled.



Multi User Server

Server Name	nDisplayExample_MU_Server
Command Line Args	
Unicast Endpoint	:9030
Auto Join	✓
Clean History	✓
Auto Launch	✓

Any changes made by the editor node should now be visible in real time on the render node, to save changes made in this state persist any changes via the source control button.


Save Current
Source Control
Modes
Content
Marketplace
Settings
Level Snaps
Actions
Connect to Source Control...
Persist Session Changes...
Check Out Modified Files...
Submit to Source Control...
Persist the session changes and prepare the files for source control submission.

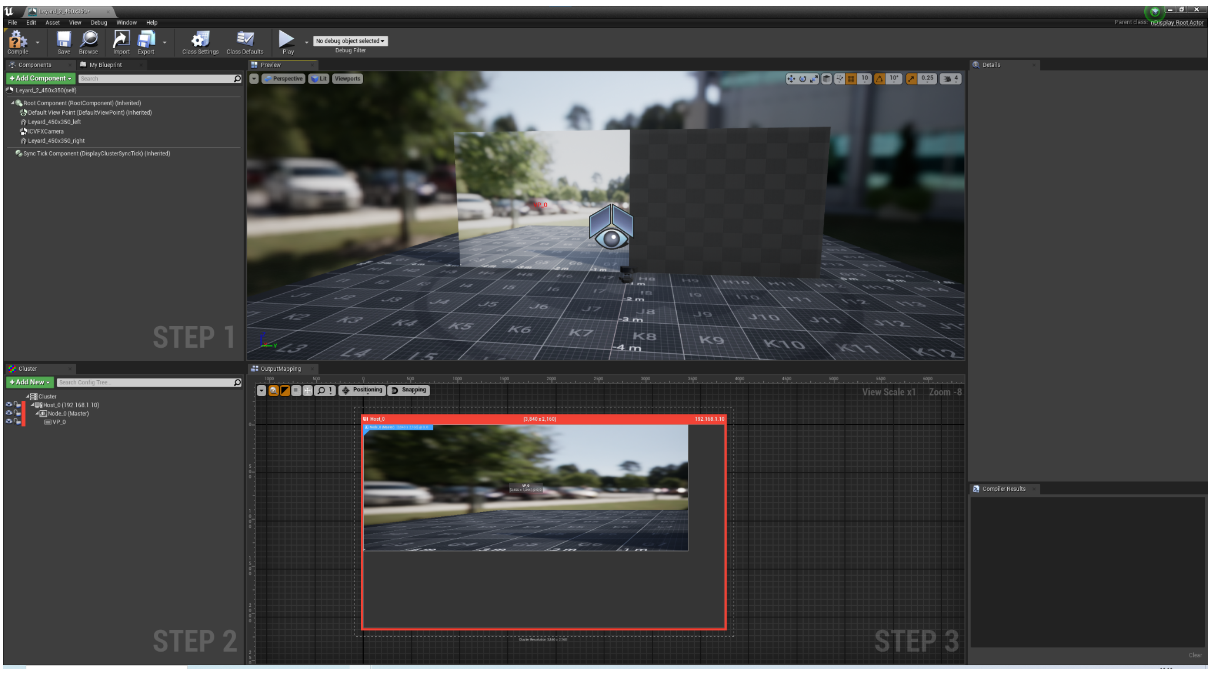
Bosh done.

Further Knowledge

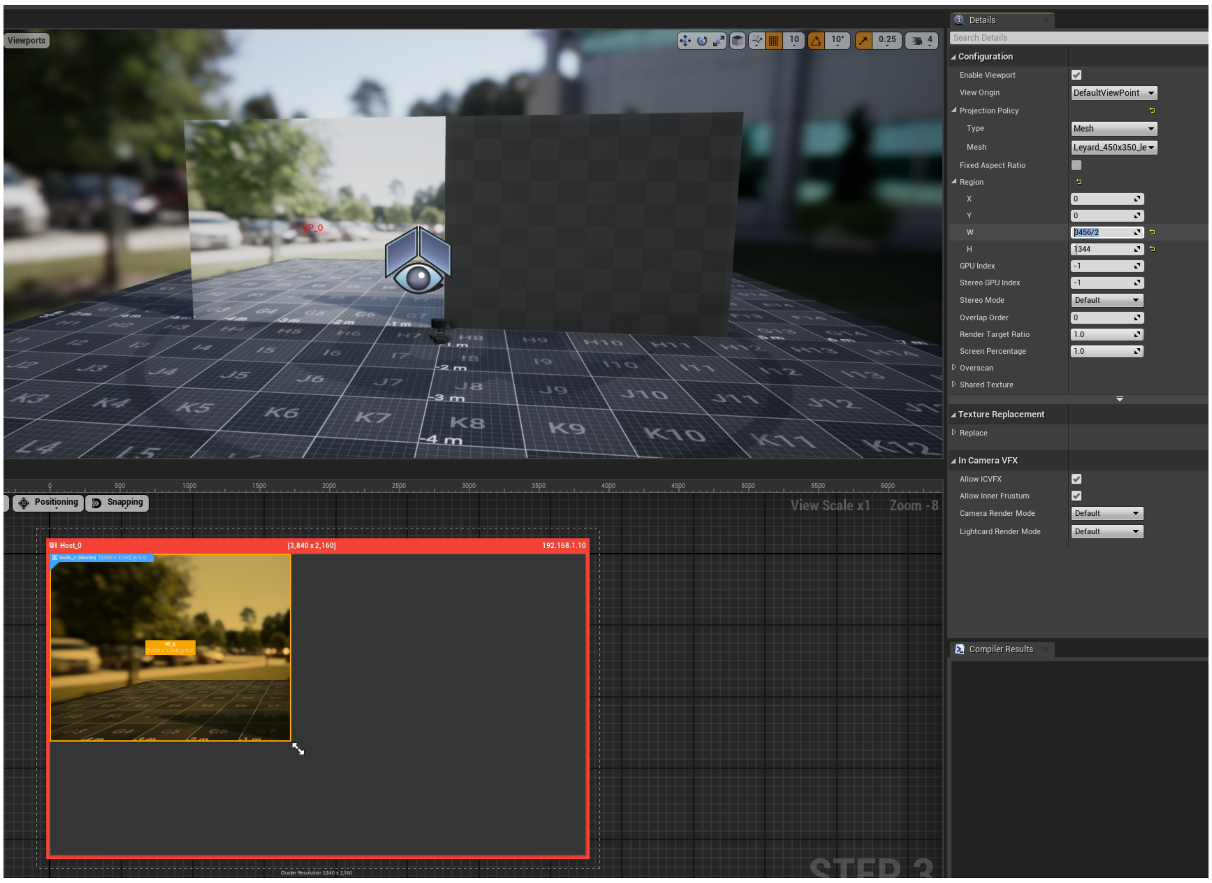
- Using multiple render nodes

If setting up multiple render nodes to drive the wall make the following changes to the nDisplay setup (Assuming the physical hardware has been set up correctly to drive one vertical half of the wall with each node via the controller input and/or video matrix):

Duplicate the nDisplay config file and open it up. The idea here is to create 2 static meshes to represent the wall and drive each half with a cluster node. So in the new config, replace the “full wall” model with a “half wall” model and duplicate, adding a “left” and “right” suffix to the relative static meshes.



Now adjust the region of the existing render node to have half its current width, do this easily by typing “/2” to the end of the width component of the region variable.



Details

Configuration
- Enable Viewport: ✓
- View Origin: DefaultViewPoint
- Projection Policy
  - Type: Mesh
  - Mesh: Leyard_450x350_le
- Fixed Aspect Ratio: ☐
- Region
  - X: 0
  - Y: 0
  - W: 3456/2
  - H: 1344
- GPU Index: -1
- Stereo GPU Index: -1
- Stereo Mode: Default
- Overlap Order: 0
- Render Target Ratio: 1.0
- Screen Percentage: 1.0
- Overscan
- Shared Texture

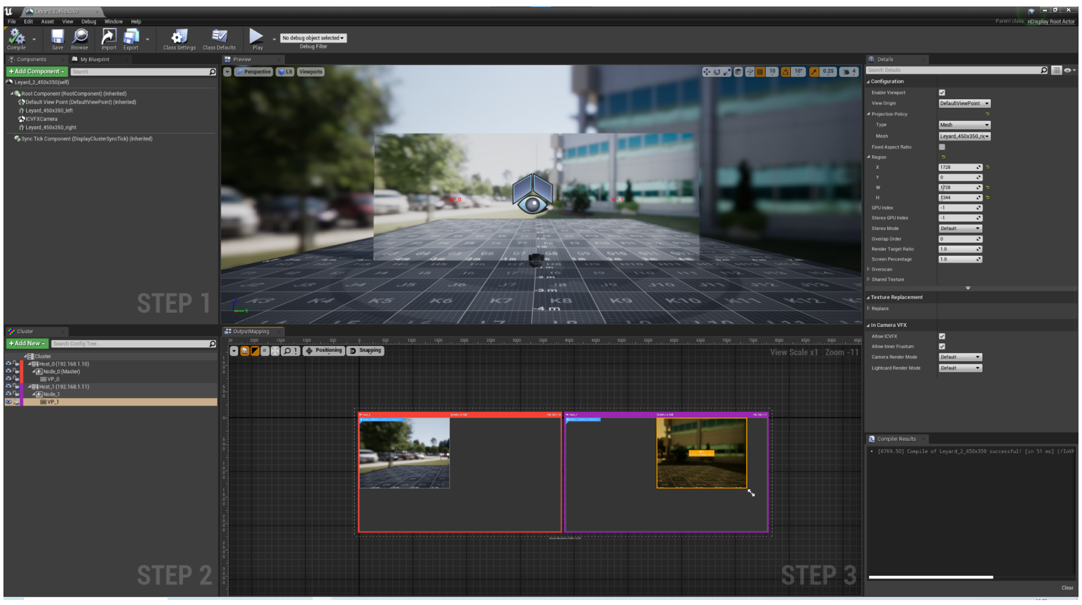
Texture Replacement
- Replace

In Camera VFX
- Allow ICVFX: ✓
- Allow Inner Frustum: ✓
- Camera Render Mode: Default
- Lightcard Render Mode: Default

The image now in the OutputMapping window shouldn't look stretched anymore.

Now create a new cluster node and set it up exactly how the first one was set up but with a couple of key differences:

- Make sure the IP address is targeting the correct render node.
- Target the other static mesh in the projection policy.
- Duplicate dimensions of the region but add in the width to the x offset.



Leyard_2_450x350(self)
- Root Component (RootComponent) (Inherited)
  - Default View Point (DefaultViewPoint) (Inherited)
  - Leyard_450x350_left
  - ICVFXCamera
  - Leyard_450x350_right
- Sync Tick Component (DisplayClusterSyncTick) (Inherited)

Cluster
- Host_0 (192.168.1.10)
  - Node_0 (Master)
    - VP_0
- Host_1 (192.168.1.11)
  - Node_1
    - VP_1

Details

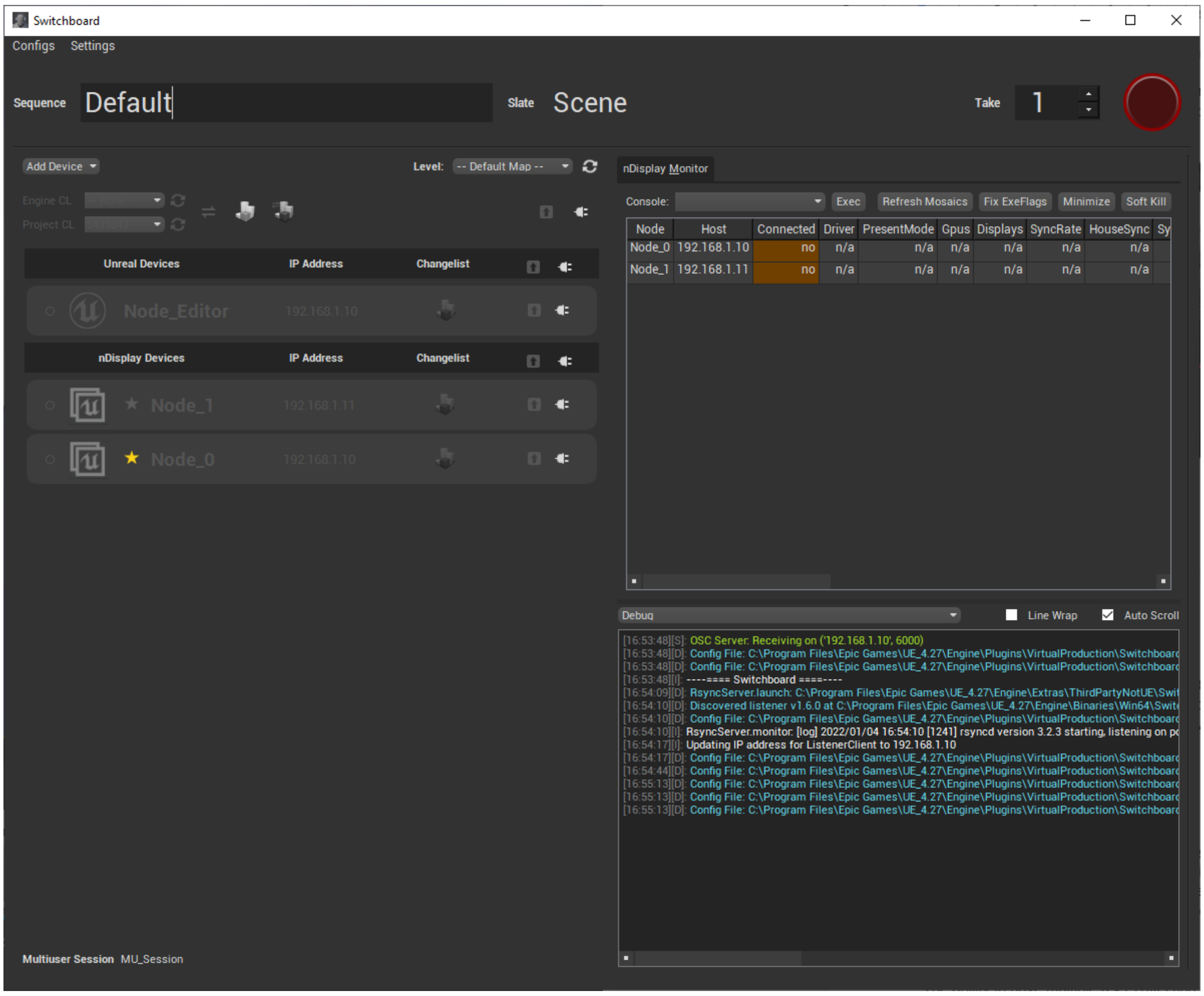
Configuration
- Enable Viewport: ✓
- View Origin: DefaultViewPoint
- Projection Policy
  - Type: Mesh
  - Mesh: Leyard_450x350_ri
- Fixed Aspect Ratio: ☐
- Region
  - X: 1728
  - Y: 0
  - W: 1728
  - H: 1344
- GPU Index: -1
- Stereo GPU Index: -1
- Stereo Mode: Default
- Overlap Order: 0
- Render Target Ratio: 1.0
- Screen Percentage: 1.0
- Overscan
- Shared Texture

Texture Replacement
- Replace

In Camera VFX
- Allow ICVFX: ✓
- Allow Inner Frustum: ✓
- Camera Render Mode: Default
- Lightcard Render Mode: Default

Compiler Results
```
[6769.50] Compile of Leyard_2_450x350 successful! [in 51 ms] (/IoVP
```

Now in the switchboard remove any old nDisplay devices, create a new one and locate the new multi node config file. This should automatically populate the nDisplay devices for all nodes in the cluster (in this case just the two).



Node	Host	Connected	Driver	PresentMode	Gpus	Displays	SyncRate	HouseSync	Sy
Node_0	192.168.1.10	no	n/a	n/a	n/a	n/a	n/a	n/a	
Node_1	192.168.1.11	no	n/a	n/a	n/a	n/a	n/a	n/a	

```
[16:53:48][S]: OSC Server: Receiving on ('192.168.1.10', 6000)
[16:53:48][D]: Config File: C:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[16:53:48][D]: Config File: C:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[16:53:48][I]: ----==== Switchboard ====----
[16:54:09][D]: RsyncServer.launch: C:\Program Files\Epic Games\UE_4.27\Engine\Extras\ThirdPartyNotUE\Swit
[16:54:10][D]: Discovered listener v1.6.0 at C:\Program Files\Epic Games\UE_4.27\Engine\Binaries\Win64\Swit
[16:54:10][D]: Config File: C:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[16:54:10][I]: RsyncServer.monitor: [log] 2022/01/04 16:54:10 [1241] rsyncd version 3.2.3 starting, listening on po
[16:54:17][I]: Updating IP address for ListenerClient to 192.168.1.10
[16:54:17][D]: Config File: C:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[16:54:44][D]: Config File: C:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[16:55:13][D]: Config File: C:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[16:55:13][D]: Config File: C:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
[16:55:13][D]: Config File: C:\Program Files\Epic Games\UE_4.27\Engine\Plugins\VirtualProduction\Switchboard
```

Once the project is fully synchronised across all cluster nodes the project can be launched.

Pitfalls

- Debugging

The only drawback singular project being edited from multiple sources is potential bottleknecking when a debugging session is required. This could be resolved by isolationing a project and workstation specifically for debugging, done most effectively by source control such as perforce which is not currently implemented in our workflow.

- Coordination

The ability to have multiple users affecting a project in real time also poses the threat of unintentional tampering with systems put in place by other users. The larger the input pool to any system the more likely mistakes and uncoordinated processing will occur, we currently don’t have a structured plan for how the Unreal operators interact with the project in the lead up to and during a shoot.