LinkXR

虚拟现实沉浸式大屏交互环境

Unreal 开发者使用手册

(Version 4.0)

目录

1 软件介绍

1.1 LinkXR 简介

LinkXR（创链）是一款将 VR 内容适配到任意的虚拟现实沉浸式大屏交互环境中的 VR 内容适配软件。用户搭配任意的基于 VRPN 协议的光学追踪系统和交互设备可以对 VR 内容进行沉浸式的交互体验，且支持将头显 VR 内容无修改直接在虚拟现实沉浸式大屏交互环境中进行交互体验并保留原有立体呈现效果。此外，也可以针对头戴式显示器进行基于真实物理空间的定位统一，使每个人在虚拟场景内的相对位置关系和真实场景一致，提供与真实物理空间一致的本地多人协同内容体验。

图 1.1 LinkXR 系统结构

1.2 LinkXR 客户端

客户端主要用于针对虚拟现实沉浸式环境的硬件进行配置，提供快速配置和自定义配置两种方式。可以根据实际的沉浸式环境来生成和修改对应的配置文件，该配置文件包含渲染机、屏幕、追踪设备等参数信息。生成配置文件后，用户在内容启动界面可以选择配置文件和制作的内容，进行内容的一键分发、启动和关闭，内容会在初始化时去读取选择的配置文件中的参数，以构建整个沉浸式系统。同时，客户端也可用于基于真实物理空间下的本地多人协同空间定位和内容体验。

图 1.2 客户端主界面

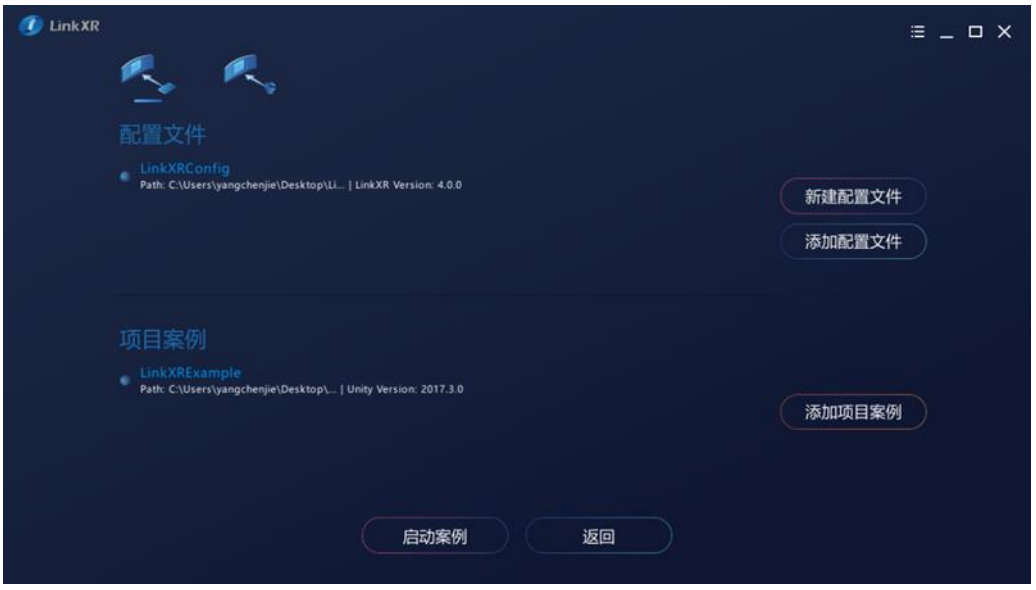

图 1.3 启动界面

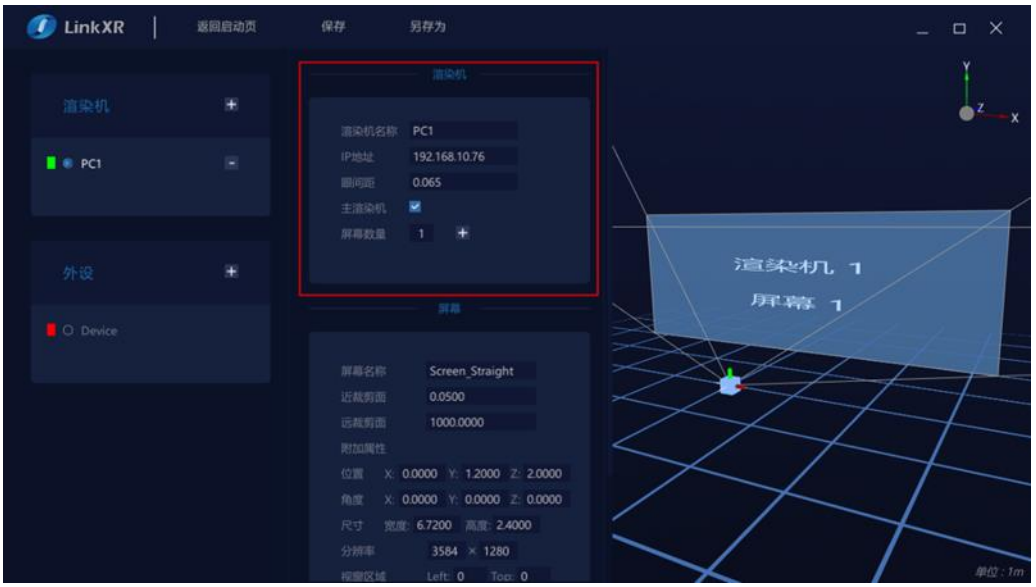

图 1.4 渲染机信息

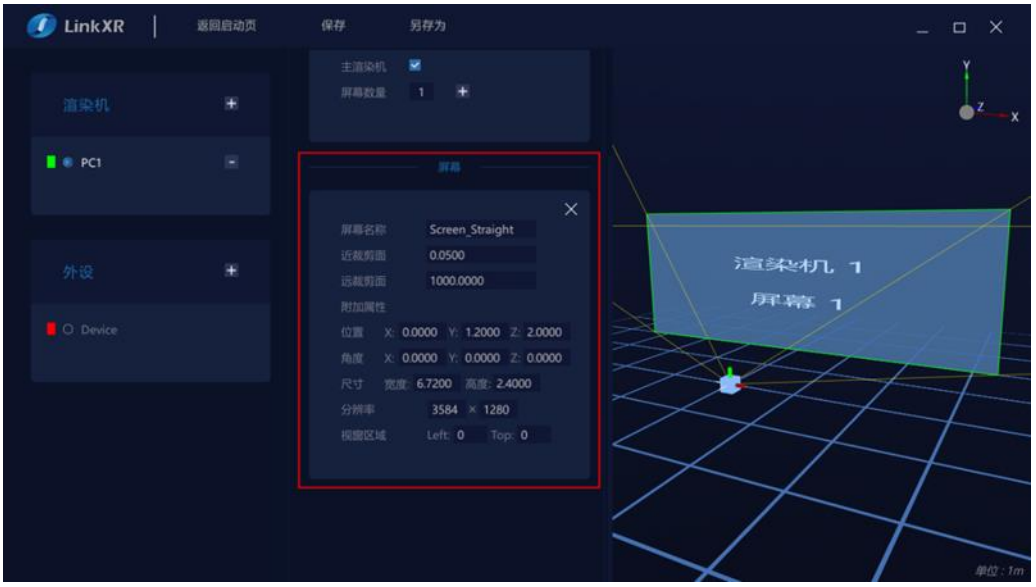

图 1.5 屏幕信息

图 1.6 追踪系统信息

图 1.7 追踪设备信息

监听端（包含在客户端内）：主要用于在单通道硬件环境中（单台渲染机）（VR 内容在单台 PC 上显示），监听和执行客户端启动或关闭 VR 内容的指令。在多通道硬件环境中（多台渲染机）（VR 内容在多台 PC 上显示，其中一台为主控 PC），接收客户端分发过来的配置文件和 VR 内容，并能监听和执行客户端启动或关闭 VR 内容的指令。

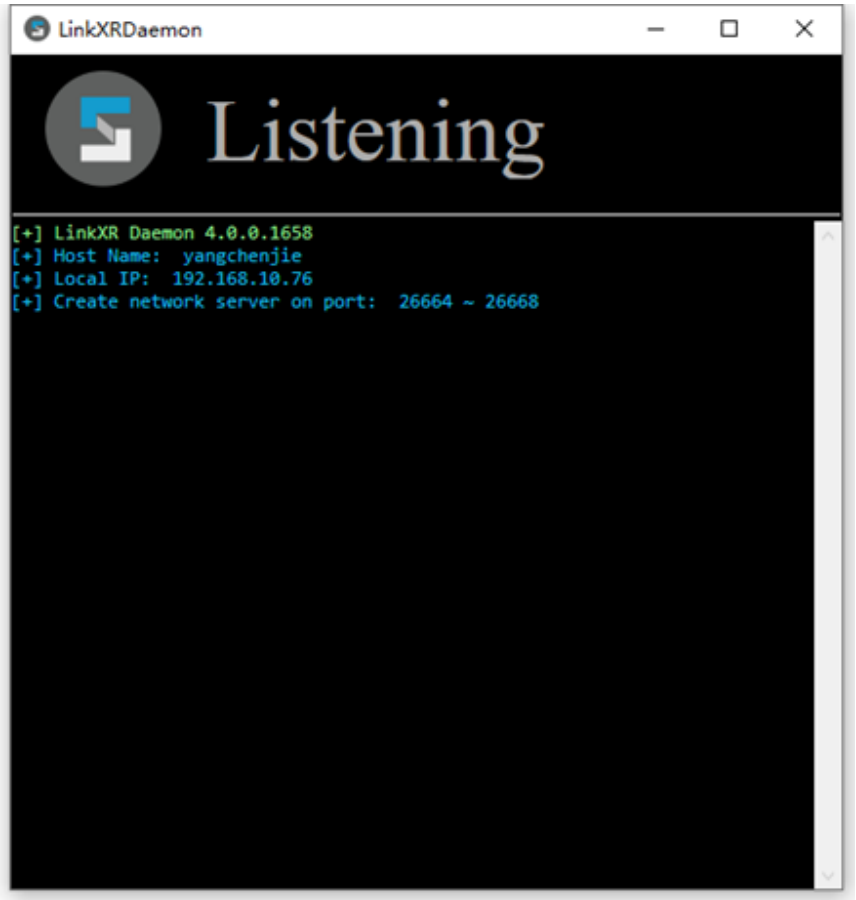

图 1.8 监听端界面

若想进一步了解 LinkXR 客户端，可以登录 http://www.gdi.com.cn/site/softwarecon/231 在页面下方申请 60 天试用。

1.3 LinkXR 配置文件

注：请务必正确配置配置文件，不要出现多个眼镜标记体或者多个左/右/单手柄标记体的情况。

➢ 外设眼镜+单手柄：

图 1.9 配置外设眼镜标记体

图 1.10 配置外设旧版单手柄标记体

➢ 外设眼镜+双手柄：

图 1.11 配置外设眼镜标记体

图 1.12 配置外设右手柄标记体

图 1.13 配置外设左手柄标记体

LinkXR 配置文件自定义编辑端与 ndisplay 配置文件的对应关系如下：

1. 渲染机

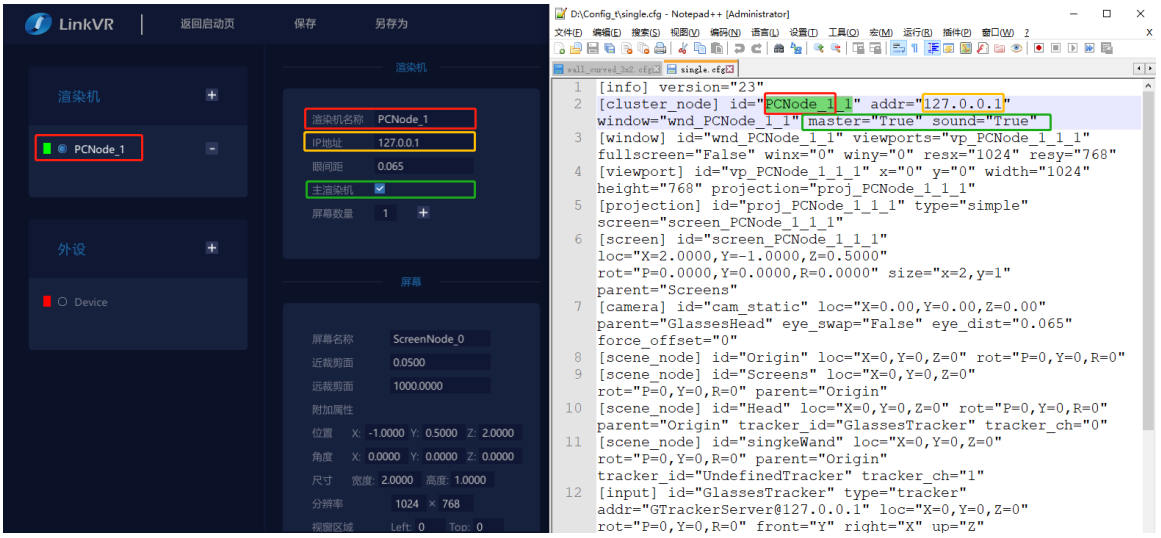

图 1.14 渲染机与 cluster_node

- Link VR 配置中，渲染机名称（可自定义输入） 对应 nDisplay 配置文件[cluster_node] 的 id。
- Link VR 配置中，渲染机 IP 地址（可自定义输入）对应 nDisplay 配置文件[cluster_node]的 addr。
- Link VR 配置中，渲染机是否是主渲染机（可自定义选择） 对应 nDisplay 配置文件[cluster_node] 的 master 和 sound。

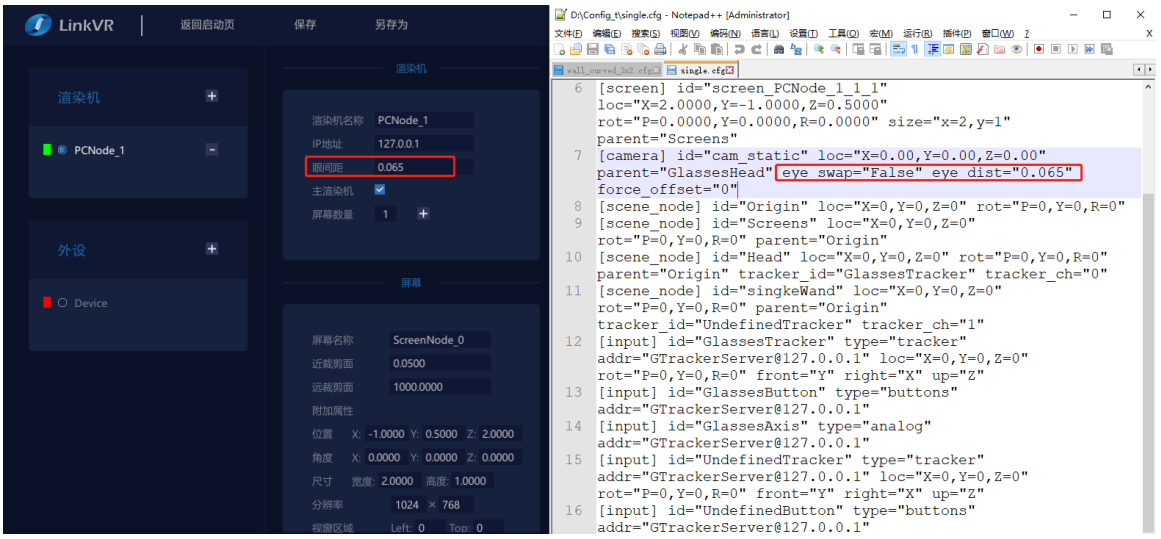

图 1.15 眼间距与 camera

- Link VR 配置中，渲染机眼间距（可自定义输入） 对应 nDisplay 配置文件[camera] 的 eye_swap 和 eye_dist。

  渲染机眼间距>0 对应 nDisplay 配置文件 [camera] eye_swap = false eye_dist = 眼间距值。

  渲染机眼间距<0 对应 nDisplay 配置文件 [camera] eye_swap = true eye_dist = 眼间距值的绝对值。

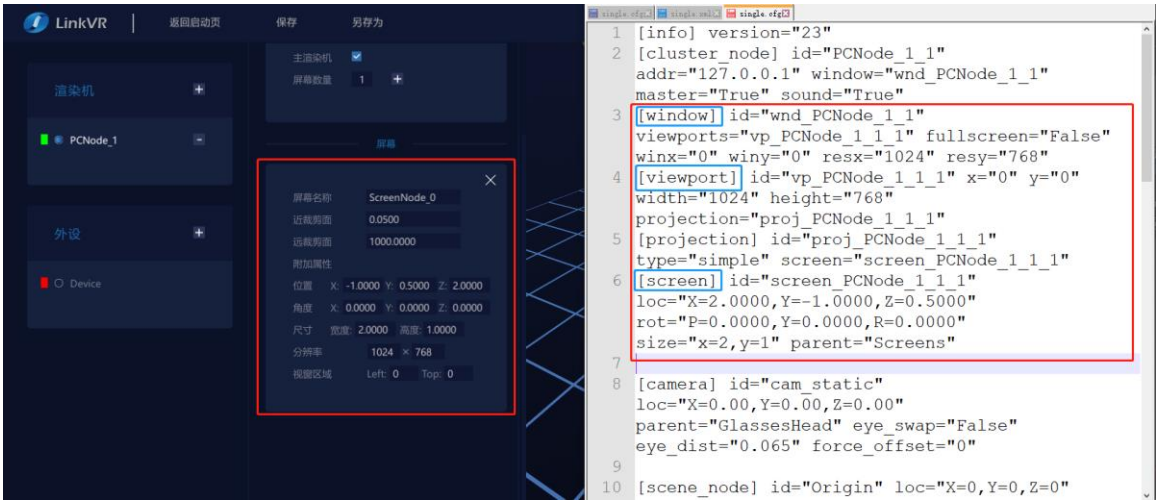

图 1.16 屏幕与 window,viewport,screen

- Link VR 配置中，屏幕位置/角度/尺寸（可自定义输入） 对应 nDisplay 配置文件[screen]的 loc/rot/size。
  - 其中，Link VR 的坐标系 X 向右，Y 向上，Z 向前；nDisplay 配置文件的坐标系为 X 向前，Y 向右，Z 向上。
- Link VR 配置中，屏幕分辨率和视窗区域（可自定义输入） 对应 nDisplay 配置文件[window] 和[viewport]。
  - [window] winx =所选渲染机对应的所有屏幕中，Left 最小的值；winy =所选渲染机对应的所有屏幕中，Top 最小的值；resx=所选渲染机对应的所有屏幕中，resolutionX +Left 的最大值；resy =所选渲染机对应的所有屏幕中，resolutiony+Top 的最大值。
  - [viewport] x=单个屏幕的 Left 值；y=单个屏幕的 Top 值；width=单个屏幕的 resolutionX；height=单个屏幕的 resolutionY。

2. 外设

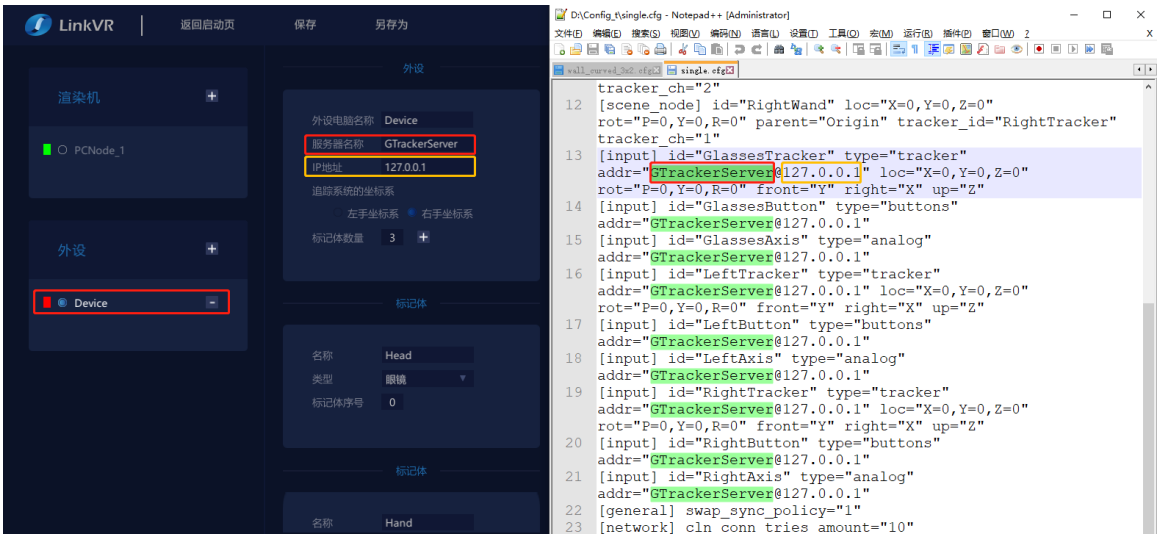

图 1.17 外设与 input

- Link VR 配置中，服务器名称与 IP 地址（可自定义输入） 对应 nDisplay 配置文件[input] 的 addr；

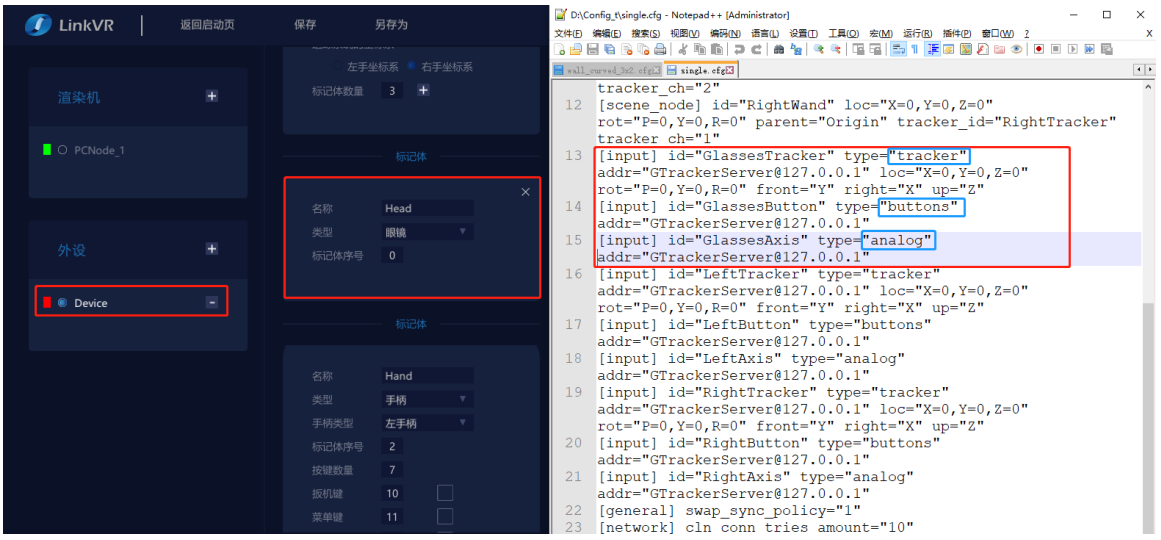

图 1.18 外设眼镜标记体与 input

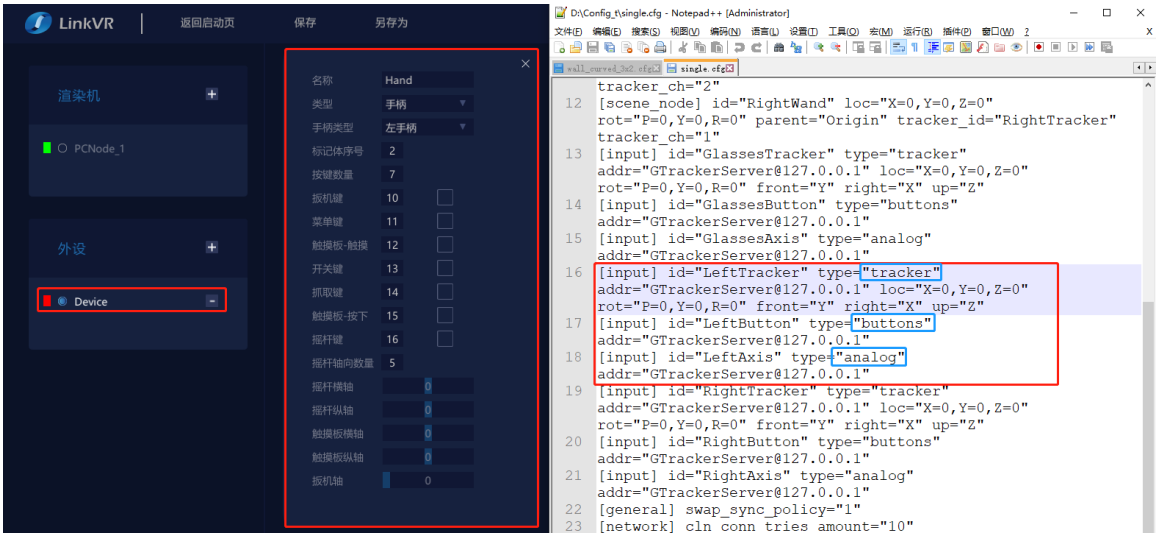

图 1.19 外设左手柄标记体与 input

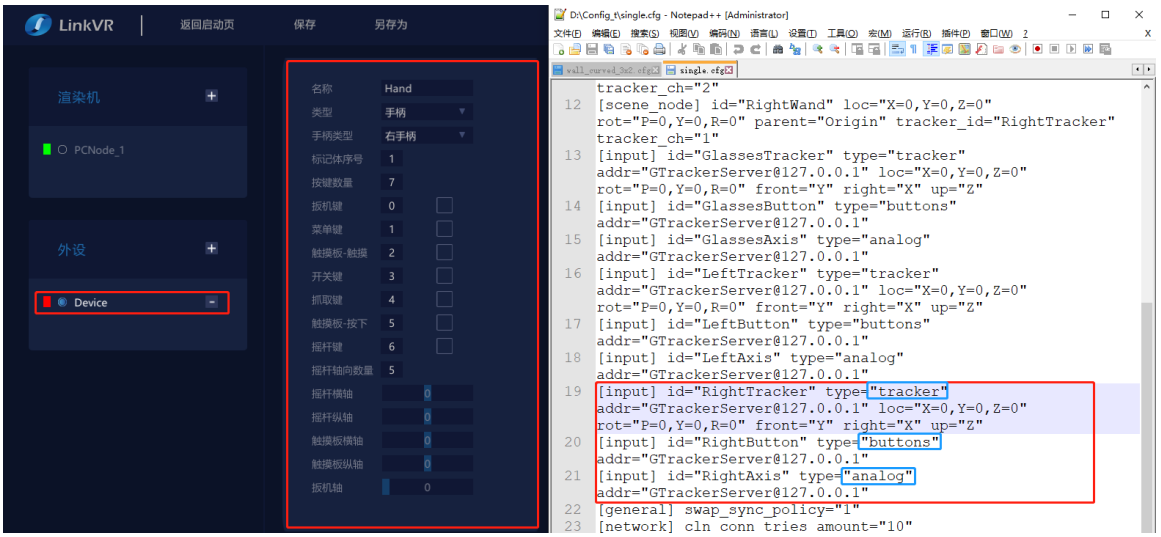

图 1.20 外设右手柄标记体与 input

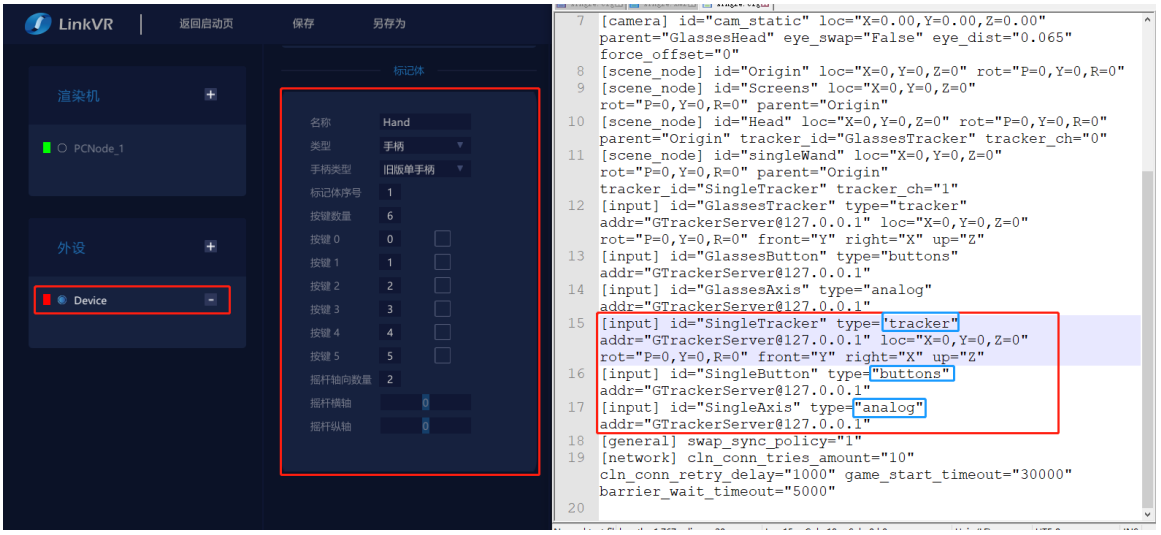

图 1.21 外设旧版单手柄标记体与 input

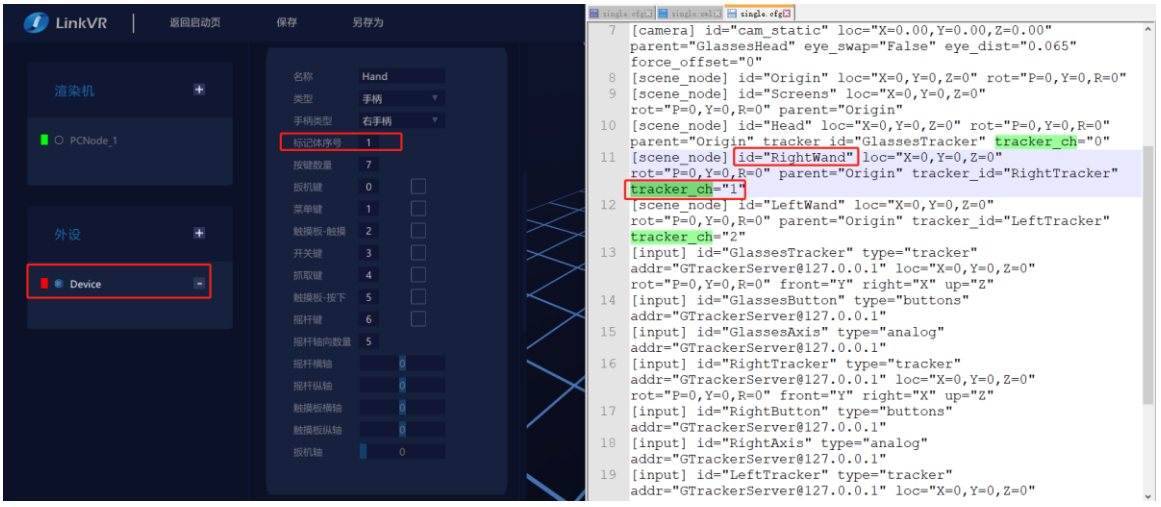

图 1.22 外设与 scene_node

- Link VR 配置中，外设标记体序号 对应 nDisplay 配置文件[scene_node] 的 tracker_ch

1.4 启动案例

图 1.23 Link VR 启动案例界面

- 添加配置文件，这里选择的是 single.xml。
- 添加项目案例，选择 UE4 案例的应用程序，如下图所示：

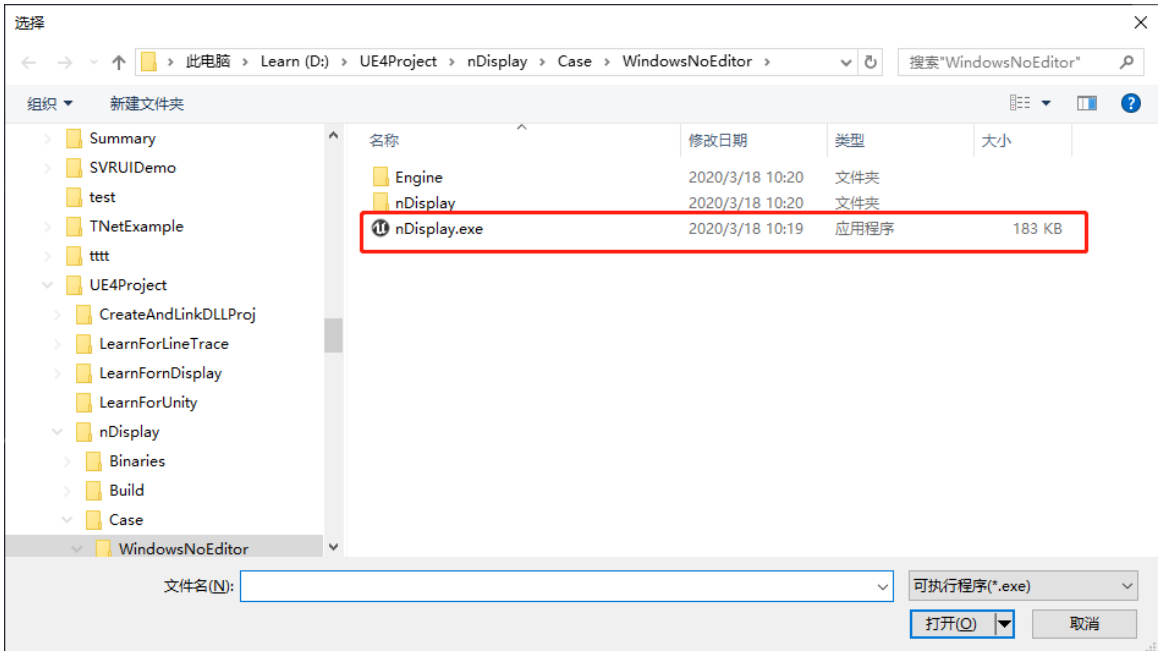

图 1.24 UE4 案例应用程序

在配置文件和项目案例选择后，点击启动案例按钮，即可通过 Link VR 客户端启动 UE4 案例。

注：点击启动 UE4 案例后，会在所选配置文件同级目录下生成相同名称的 nDisplay 配置文件，可供 UE4 编辑器下进行测试。

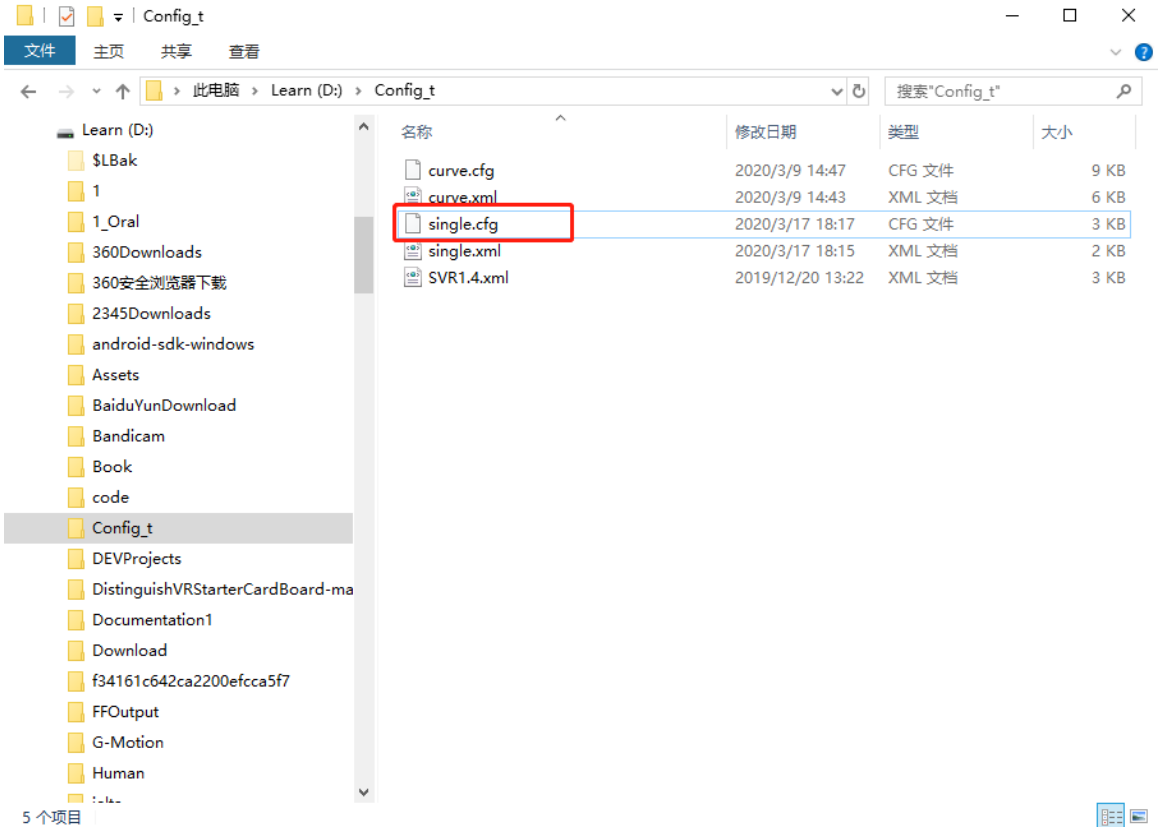

图 1.25 nDisplay 配置文件

2 开发须知

2.1 版本要求

Unreal Engine 4.23 及以上版本

2.2 VRPN 数据

1. 眼镜

   - Tracker：默认 0

2. 手柄

   - 旧版单手柄

     - Tracker：默认 1
     - Button：0,1,2,3,4,5,6
     - Anolog: 0,1,2,3,4

Button 序号所对应的按键类型：

0	手柄 2 号键
1	手柄 1 号键
2	方向键-右
3	方向键-下
4	方向键-左
5	方向键-上

Axis 序号所对应的轴类型：

0	摇杆横轴
1	摇杆纵轴

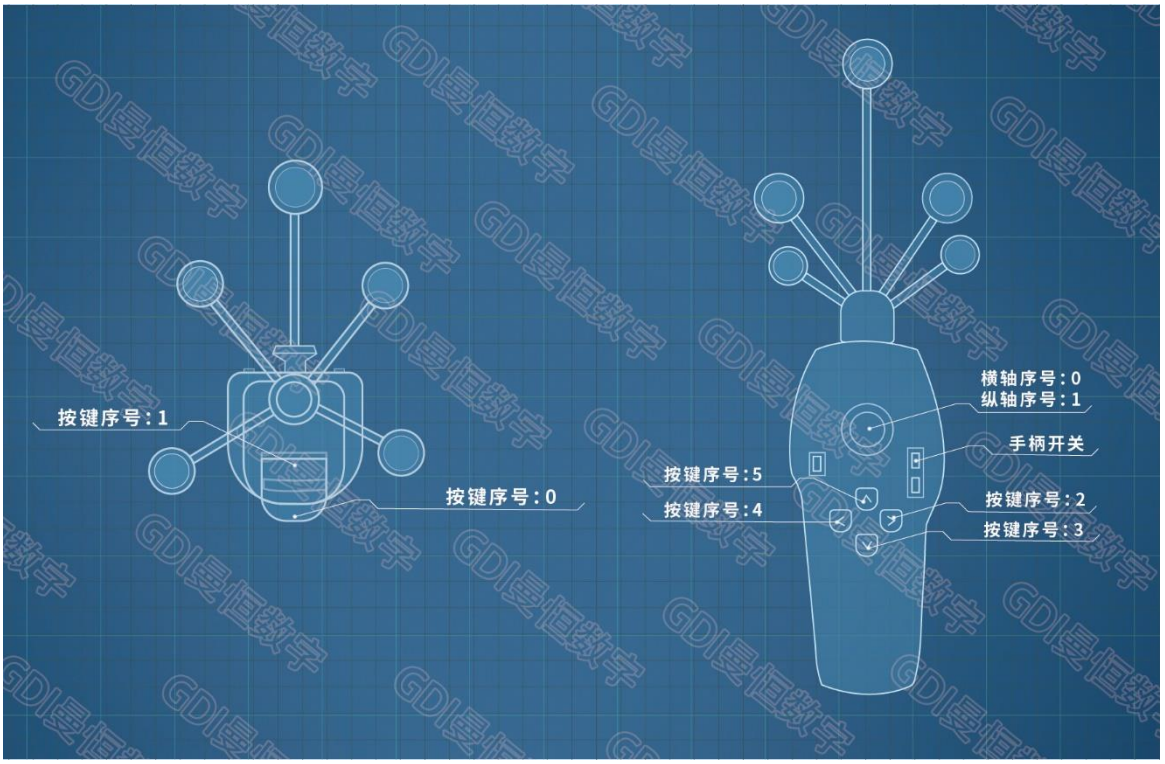

图 2.1 旧版单手柄

- 新版双手柄

  右手柄

  - Tracker：默认 1
  - Button：0,1,2,3,4,5,6
  - Anolog: 0,1,2,3,4

  左手柄

  - Tracker：默认 2
  - Button：10,11,12,13,14,15,16
  - Anolog: 6,7,8,9,10

Button 序号所对应的按键类型：

0/10	扳机键
1/11	菜单键
2/12	开关键
3/13	触摸板（触摸）
4/14	抓取键
5/15	触摸板（按下）
6/16	摇杆键

Axis 序号所对应的轴类型：

0/6	摇杆横轴

1/7	摇杆纵轴
2/8	触摸板横轴
3/9	触摸板纵轴
4/10	扳机轴

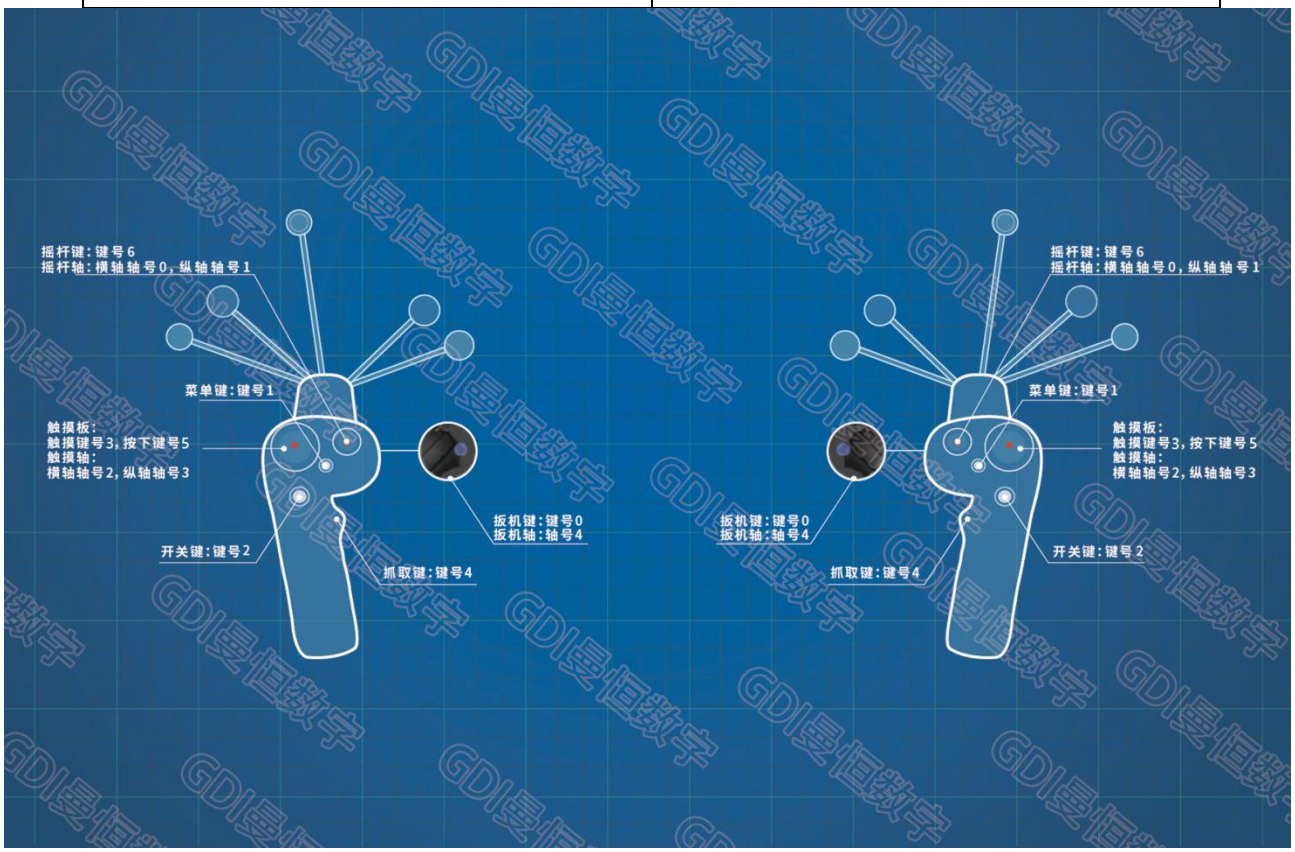

图 2.2 新版双手柄

注：以上 Tracker 需根据实际环境中所定义的追踪序号进行动态的修改。

2.3 VRPN 数据如何绑定

1. 找到新建的 nDisplay 工程默认路径下的配置文件，默认路径：案例所在文件夹\Content\ExampleConfigs**.cfg，（这里我们以 single.cfg 为例）开发者可自行查看。

2. 在配置文件中，添加[input]输入配置，如下图所示：

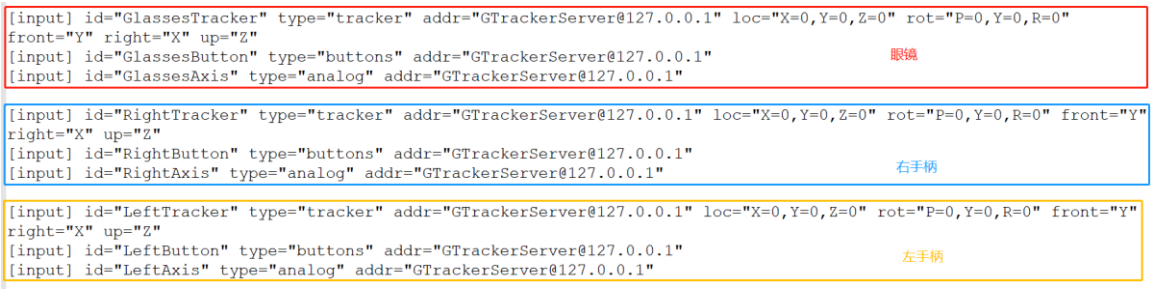

图 2.3 配置文件中的输入配置

- id：此输入设备配置的唯一命名。
- type：VRPN 输入设备的类型：
  - tracker 为跟踪设备。
  - analog 为生成轴数据的设备。
  - button 为生成布尔按钮数据的设备。
  - keyboard 为标准计算机键盘。
- addr：VRPN 设备名称@VRPN 设备所在的 IP 地址。

此时，VRPN 数据已绑定完成。

2.4 如何查看 VRPN 数据

1. 在 nDisplay 案例中，新建蓝图，右键找到 Display Cluster Module API。

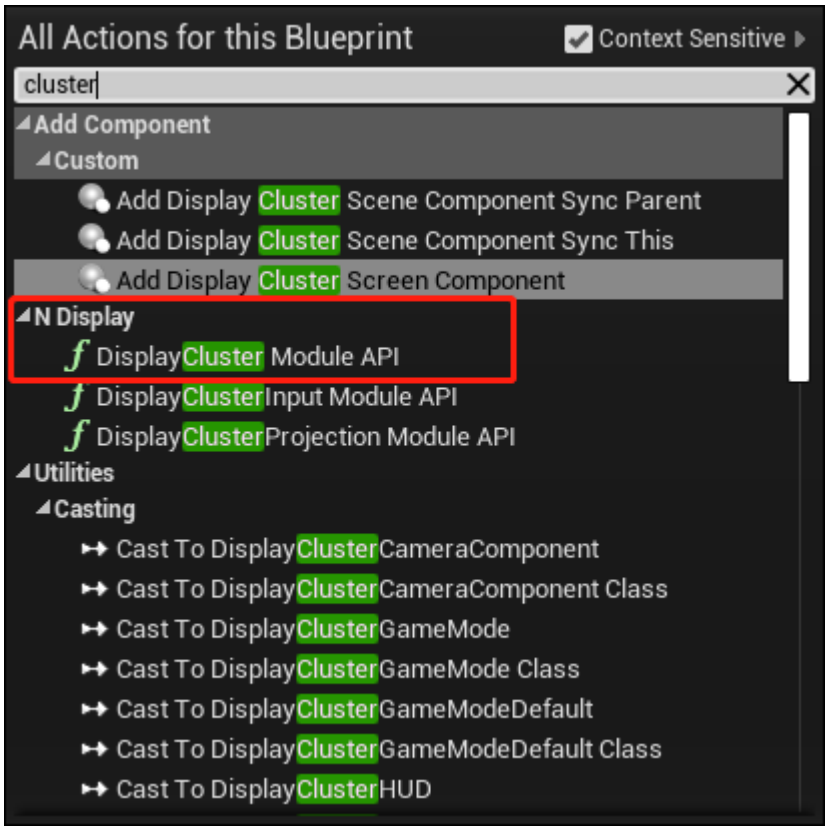

图 2.4 查找 Display Cluster Module API 组件

2. Tracker

以右手柄追踪为例，如下图所示：

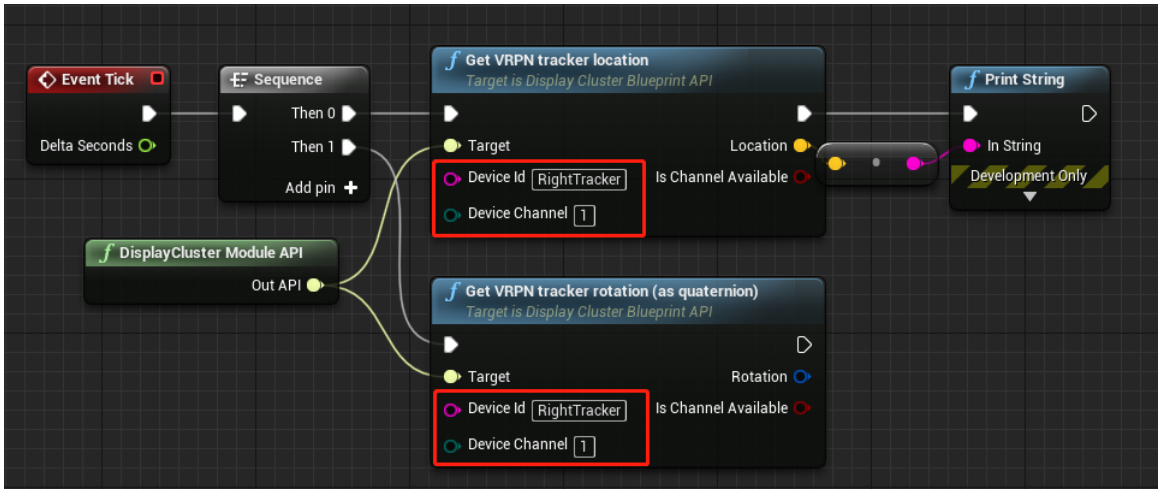

图 2.5 获取右手柄 VRPN 追踪

注：在 4.26 中函数变更为纯函数，以右手柄追踪为例，如下图所示：

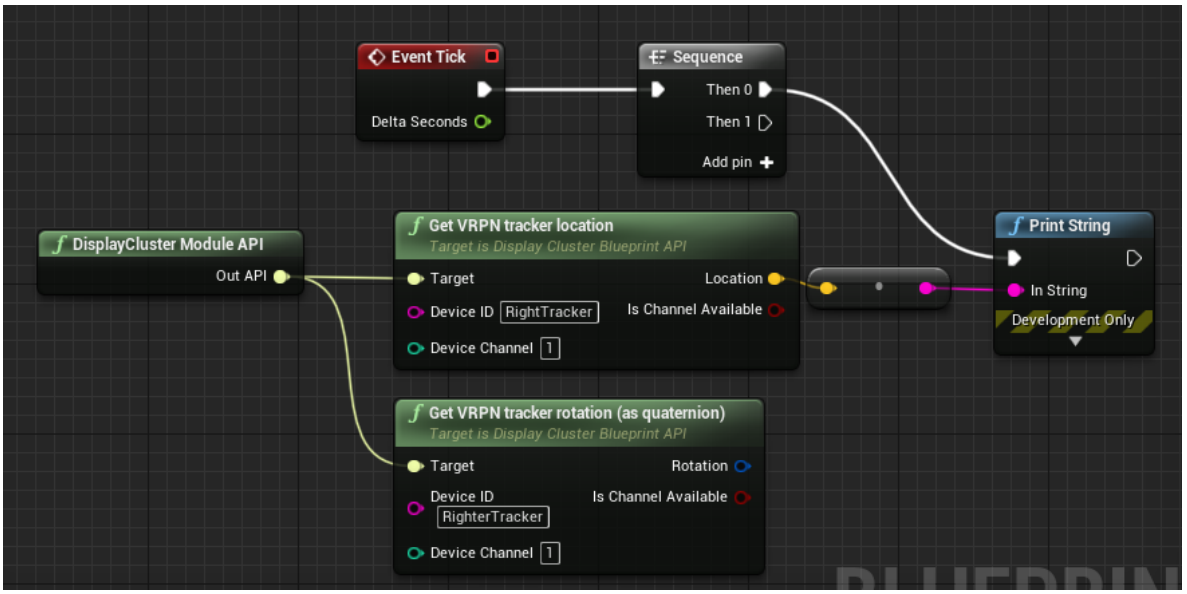

图 2.6 在 4.26 获取右手柄 VRPN 追踪

Get VRPN Tracker rotation （获取 VRPN 追踪物体的旋转）和 Get VRPN Tracker location（获取 VRPN 追踪物体的位置）组件中，主要参数：

- Device_id:配置文件中，设备 id（这里是右手柄追踪的 id）。

```
[input] id="RightTracker" type="tracker" addr="GTrackerServer@127.0.0.1" loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" front="Y"
right="X" up="Z"
```

图 2.7 配置文件中右手柄 VRPN 追踪输入配置

- Device Channel:追踪设备通道，根据上述 VRPN 数据表格可知，右手柄 trakcer 通道为 1。

3. Button

以右手柄 Trigger 按键为例，如下图所示：

图 2.8 获取右手柄 VRPN Trigger 按键

注：在 4.26 中函数变更为纯函数，以右手柄追踪为例，如下图所示：

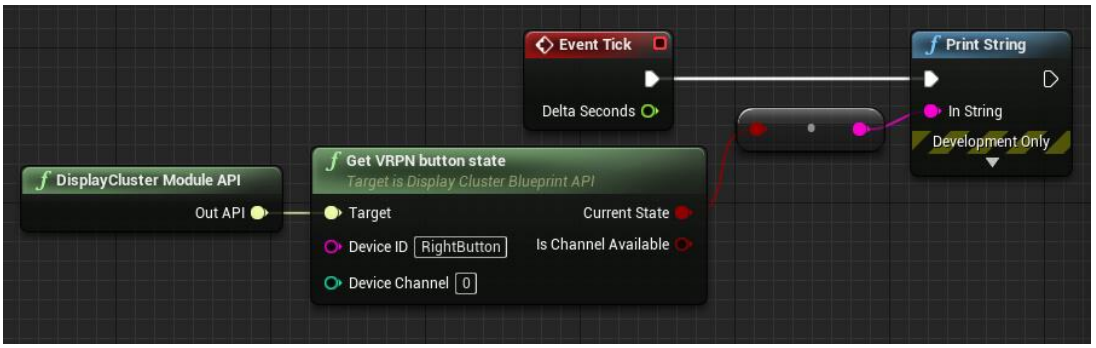

图 2.9 在 4.26 中获取右手柄 VRPN Trigger 按键

Get VRPN button state （获取 VRPN 按键状态）主要参数：

- Device_id:配置文件中，设备 id（这里是右手柄按键的 id）。

```
[input] id="RightButton" type="buttons" addr="GTrackerServer@127.0.0.1"
```

图 2.10 配置文件中右手柄 VRPN 按键输入配置

- Device Channel:设备按键通道，根据上述 VRPN 数据表格可知，右手柄 trigger 键通道为 0。

4. Anolog

以右手柄摇杆横轴为例，如下图所示：

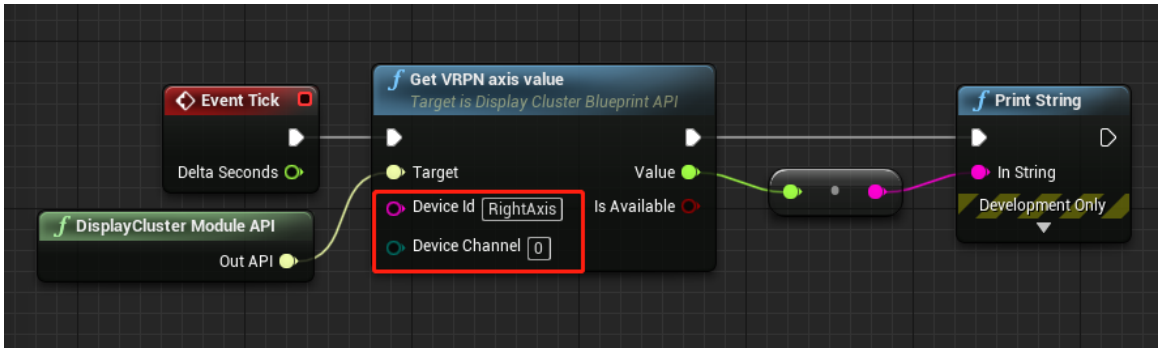

图 2.11 获取右手柄 VRPN 摇杆横轴

注：在 4.26 中函数变更为纯函数，以右手柄追踪为例，如下图所示：

图 2.12 获取右手柄 VRPN 摇杆横轴

Get VRPN axis value（获取 VRPN 轴值）主要参数：

- Device_id:配置文件中，设备 id（这里是右手柄轴的 id）。

```
[input] id="RightAxis" type="analog" addr="GTrackerServer@127.0.0.1"
```

图 2.13 配置文件中右手柄 VRPN 轴输入配置

- Device Channel:设备轴通道，根据上述 VRPN 数据表格可知，右手柄摇杆横轴通道为 0。

3 案例制作

3.1 新建 nDisplay 工程

以下以 4.23 为例进行介绍，新建一个 nDisplay 工程，如下图所示：

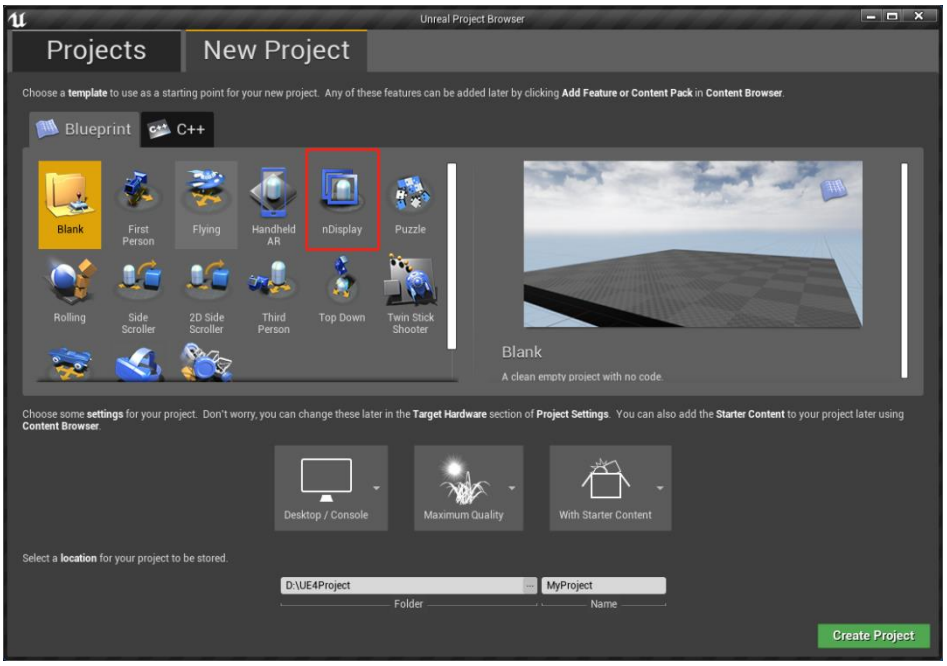

图 3.1 新建 nDisplay 工程

注：在 4.24 及后续版本中 nDisplay 项目移至 Film，Television，and Live Events，如下图所示：

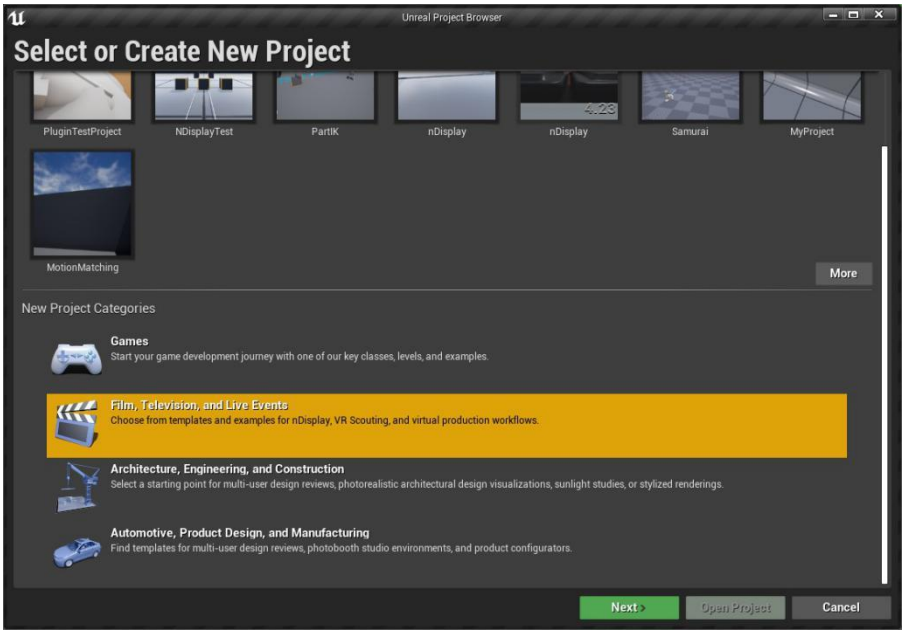

图 3.2 选择（Film，Television，and Live Events）（4.26）

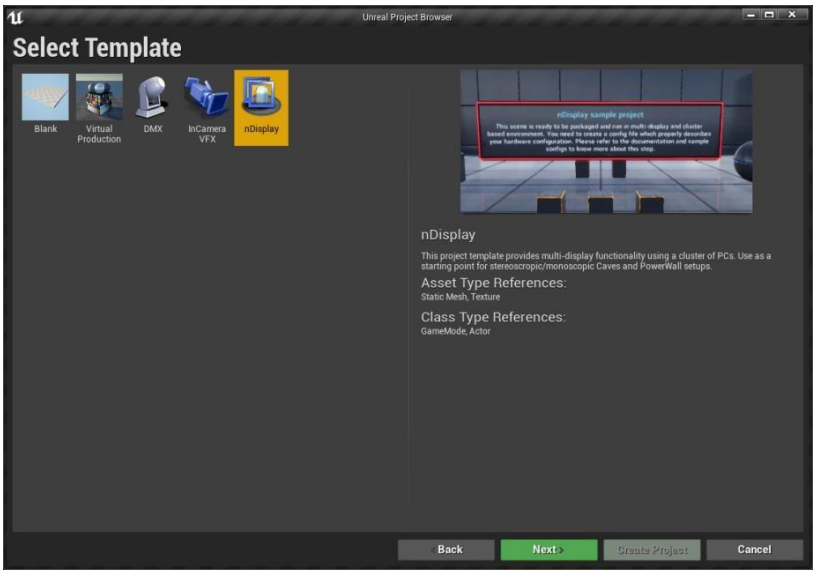

图 3.3 新建 nDisplay 工程（4.26）

nDisplay 学习教程：https://docs.unrealengine.com/zh-CN/Engine/Rendering/nDisplay/index.html

3.2 配置文件绑定

打开创建的 nDisplay 工程，在 World Outliner 中找到 DisplayClusterSettings1，修改其 details 面板中的 Display Cluster（Editor Only）的参数，如下图所示：

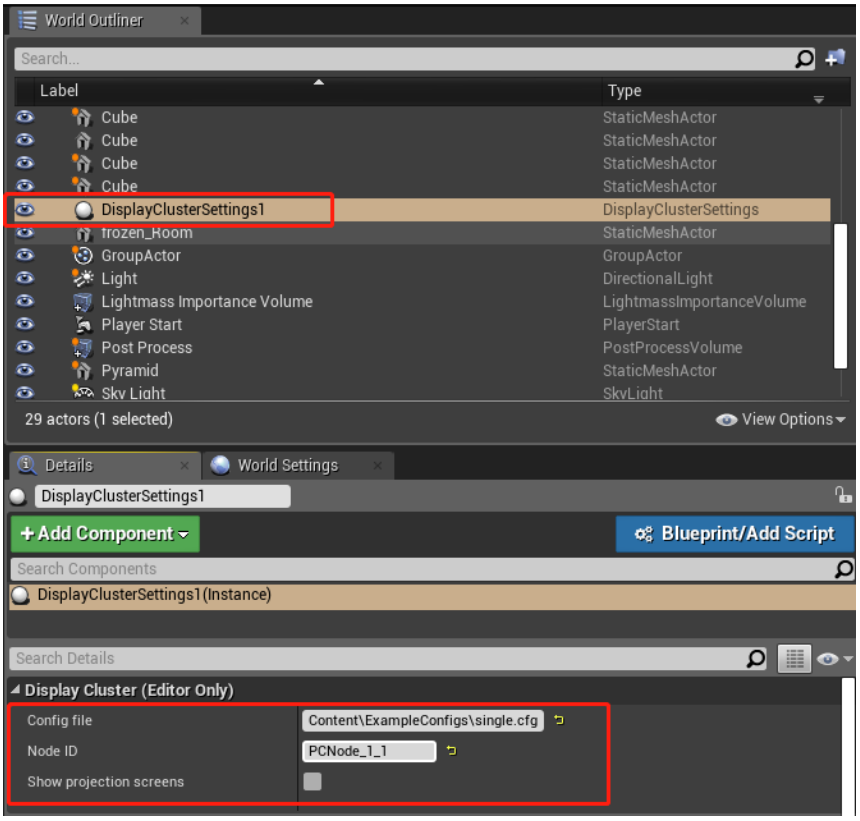

图 3.4 Display Cluster

Display Cluster 对应的主要参数：

- Config file：配置文件路径，引擎默认配置文件文件夹路径为：案例所在文件夹\Content\ExampleConfigs**.cfg（我们这里用的是 single.cfg）。
- Node ID：配置文件中，主群集节点的 ID。

```
[cluster_node] id="PCNode_1_1" addr="127.0.0.1" window="wnd_PCNode_1_1" master="True" sound="True"
```

图 3.5 配置文件中主群集节点配置

注：在 4.24 版本中 nDisplay 不再需要特殊的 Pawn 和 GameMode 类。现在 nDisplay 系统将初始自动新建一种组件类型（DisplayClusterRootComponent），其将附加一个实例到活跃的摄像。可以在场景中的 DisplayClusterRootActor 进行配置文件绑定。

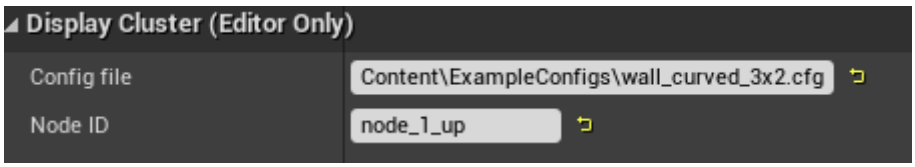

图 3.6 DisplayClusterRootActor 配置

注：在 4.26 版本中 可以使用全新的 nDisplay 配置查看器来轻松配置你的 nDisplay 设置。查看器可以显示配置层级、渲染节点、窗口、视口、输入、摄影机和投影策略的树状结构，每个都有自己的细节面板。你还可以看到视口的 2D 布局和投影策略的 3D 显示。可以在场景中的 DisplayClusterRootActor 进行配置文件绑定的同时选择主节点和相机。

图 3.7 nDisplay 配置查看器

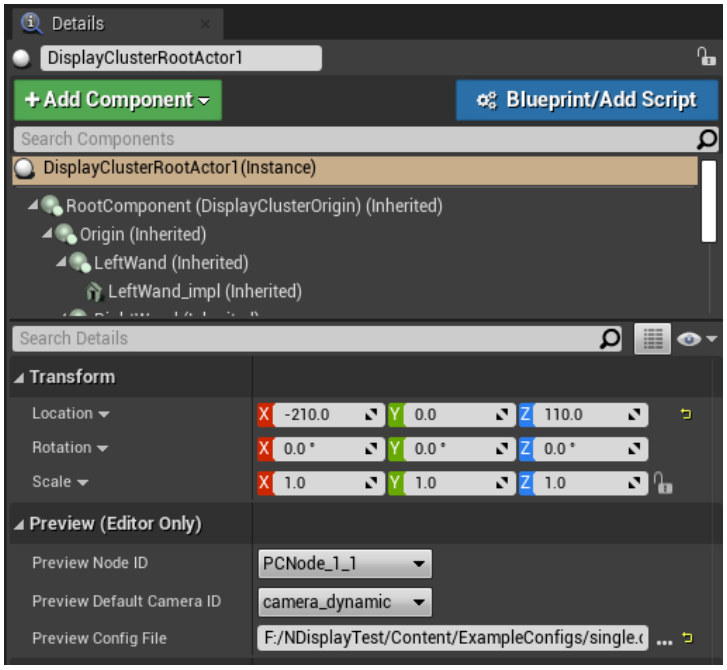

图 3.8 DisplayClusterRootActor 配置

3.3 人物漫游

新建蓝图，要实现人物漫游，首先要找到默认的 Pawn 节点，通过手柄摇杆轴的 VRPN 数据，来控制人物的旋转和前进后退（以下以右手柄摇杆横轴与摇杆纵轴来控制人物漫游为例），如下图所示：

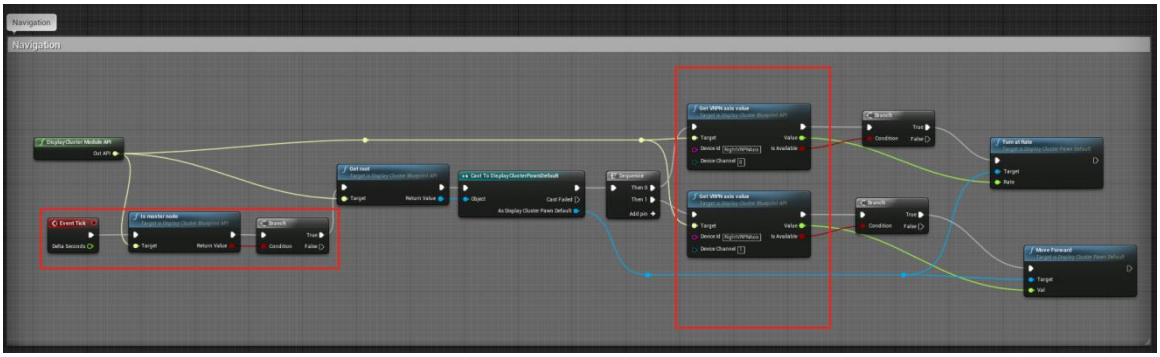

图 3.9 实现人物漫游

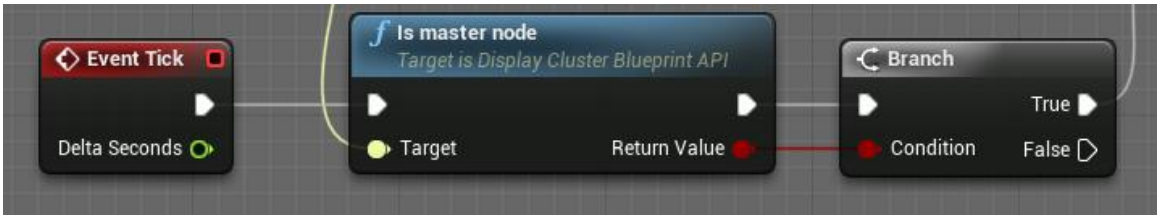

图 3.10 主节点

在实现人物漫游时，有一点很重要，那就是要获取主节点，并根据主节点的信息来控制整体的漫游。若没有这一步，则会出现人物在漫游过程中，多主机多通道的环境下，会发生由于速度不一致导致画面拼接错误的问题（即同步问题）。

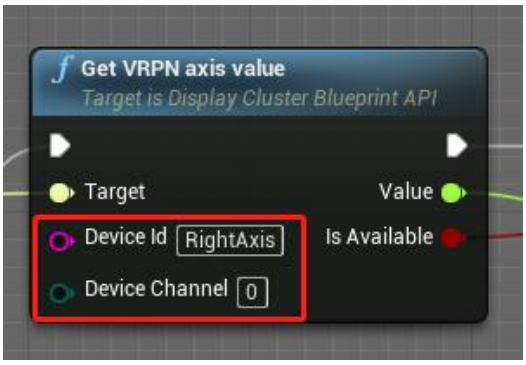

图 3.11 获取右手柄摇杆横轴

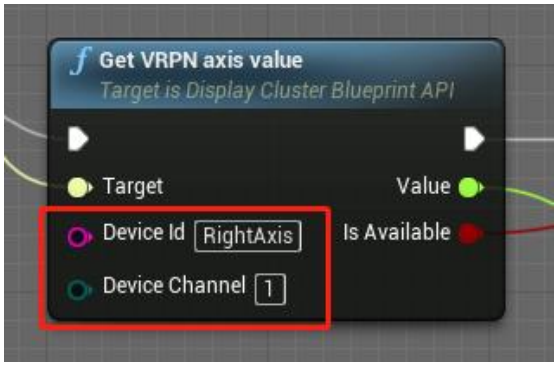

图 3.12 获取右手柄摇杆纵轴

Get VRPN axis value（获取 VRPN 轴值）主要参数：

- Device_id:配置文件中，设备 id（这里是右手柄轴的 id）。

```
[input] id="RightAxis" type="analog" addr="GTrackerServer@127.0.0.1"
```

图 3.13 配置文件中右手柄 VRPN 轴输入配置

- Device Channel:设备轴通道，根据上述 VRPN 数据表格可知，右手柄摇杆横轴通道为 0。

注：在 4.24 及后续版本中 Is master node 节点变更为纯函数

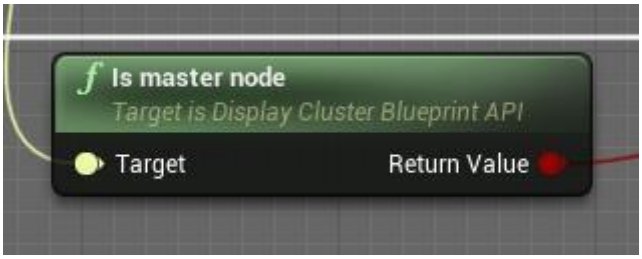

图 3.14 在 4.24 及后续版本中 Is master node 节点

注：在 4.24 及后续版本中 nDisplay 不再需要特殊的 Pawn 和 GameMode 类。原本 get root 会得到 Display Cluster Default Pawn 现在将替换成 get root actor 纯函数得到 Display Cluster Default Root Actor，这会导致用于移动的函数 Move Forword 和 Turn At Rotate 失效，因此用 Get Player Pawn 获得控制的 Pawn 来实现移动，以下以 4.26 为例。

在初始化时，查询 BP_Right 中的 Fly Mode 属性，如果为 TRUE 则开启飞行模式，关闭碰撞体。

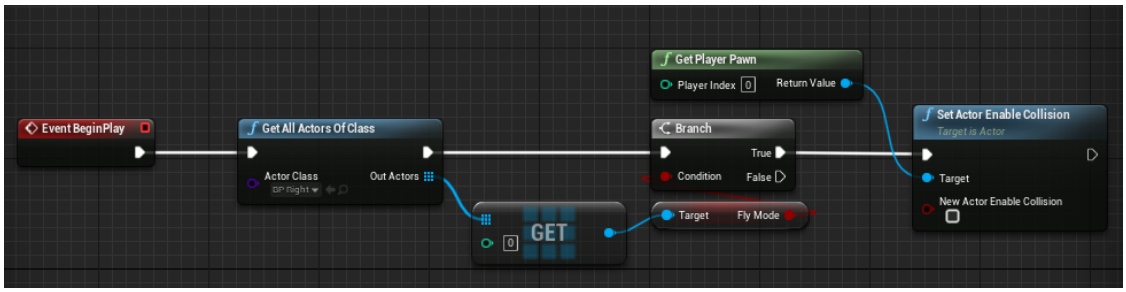

图 3.15 初始化

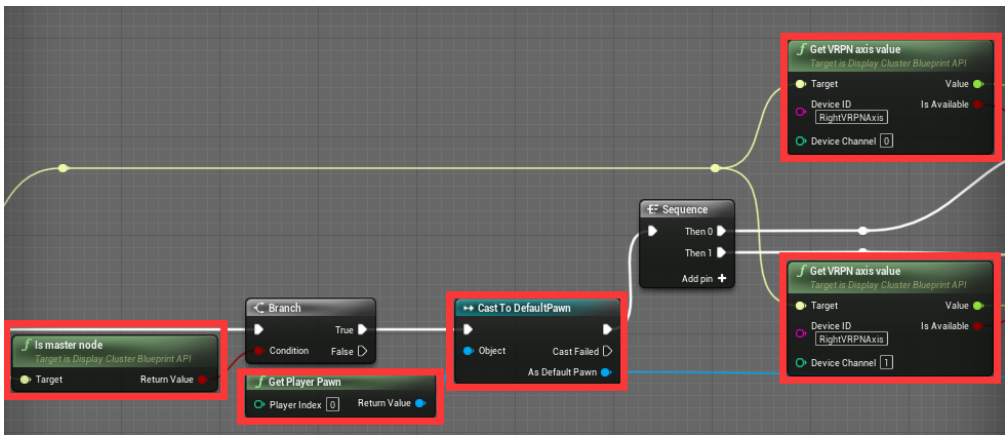

图 3.16 在 4.24 及后续版本中变更

用 Get Player Pawn 获得玩家控制的 Pawn ，调用 Default Pawn 的 Turn At Rate 实现转向。

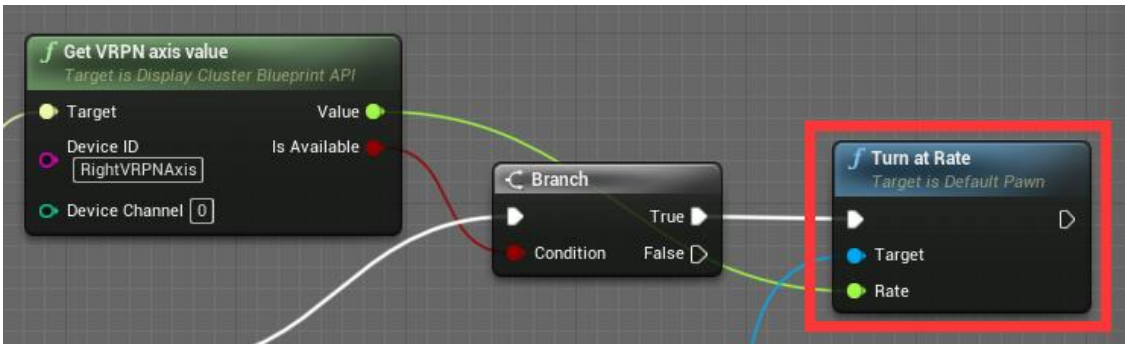

图 3.16 在 4.26 实现人物转向

获得场景中 BP_Right 实例并获得其 Fly Mode 属性及 Wand 引用，如果 Fly Mode 为 FALSE 则调用 Default Pawn 的 Move Forword 实现人物的前后移动；如果 Fly Mode 为 TRUE 则调用 Consumer Movement Input Vector 将未应用的 Movement Input Vector 置零然后利用先前的 Wand 调用 Get Forword Vector 获得手柄朝向，再将其作为输入给到 Add Movement Input 节点实现人物朝手柄方向移动的目的。

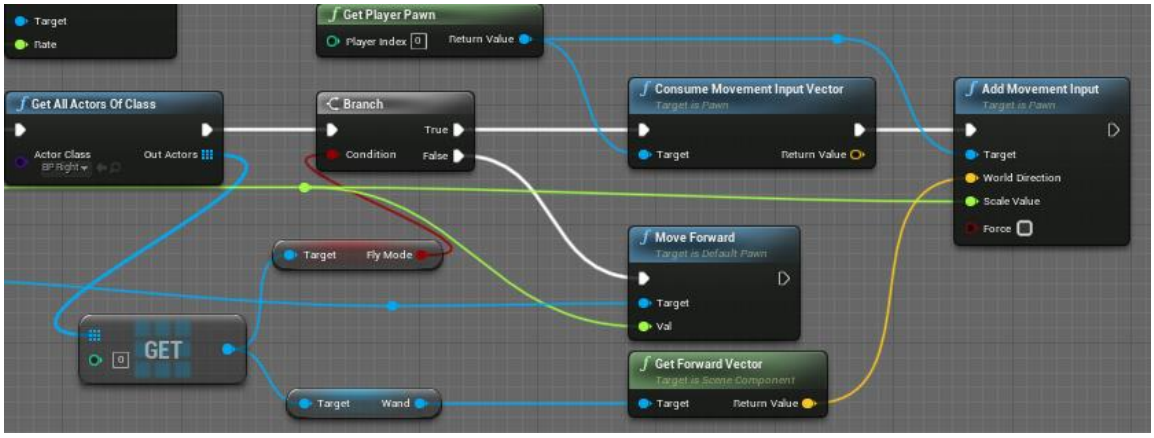

图 3.17 人物移动

3.4 手柄可视化追踪

1. 在配置文件中，我们需要新建场景节点，可定义场景节点的层级，配置文件中设置且需要 3D 空间中位置和旋转的项目（例如相机或投影屏），都可使用此类[scene_node]配置作为其父项。如下图所示：

```
[scene_node] id="Origin" loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0"
[scene_node] id="Screens" loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" parent="Origin"
[scene_node] id="Head" loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" parent="Origin" tracker_id="GlassesTracker" tracker_ch="0"
[scene_node] id="RightWand" loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" parent="Origin" tracker_id="RightTracker" tracker_ch="1"
[scene_node] id="LeftWand" loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" parent="Origin" tracker_id="LeftTracker" tracker_ch="2"
```

图 3.18 配置文件中的场景节点配置

2. 新建蓝图，命名为 BP_Right，按下图所示，添加组件，作为可视化的手柄。选择添加组件，这里我们新建一个 cube 来当作手柄，命名为 Wand；同样，新建 cylinder 来当作射线，命名 Ray。

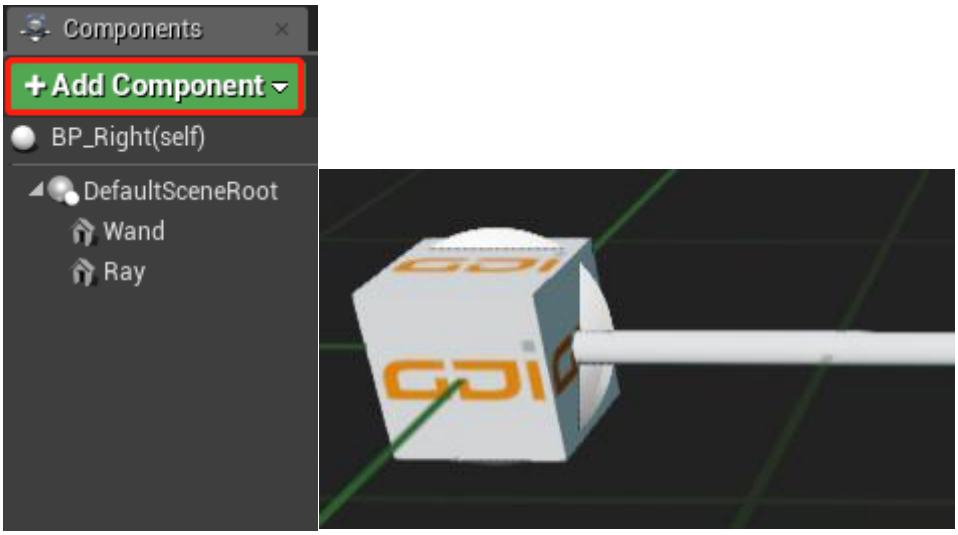

图 3.19 右手柄可视化物体

3. 我们以[scene_node] id = “RightWand”为例，在蓝图中，将右手柄通过 Get Node by ID 组件绑定 Id 为 RightWand 的节点，并将该节点设置为手柄物体的父节点，如下图所示：

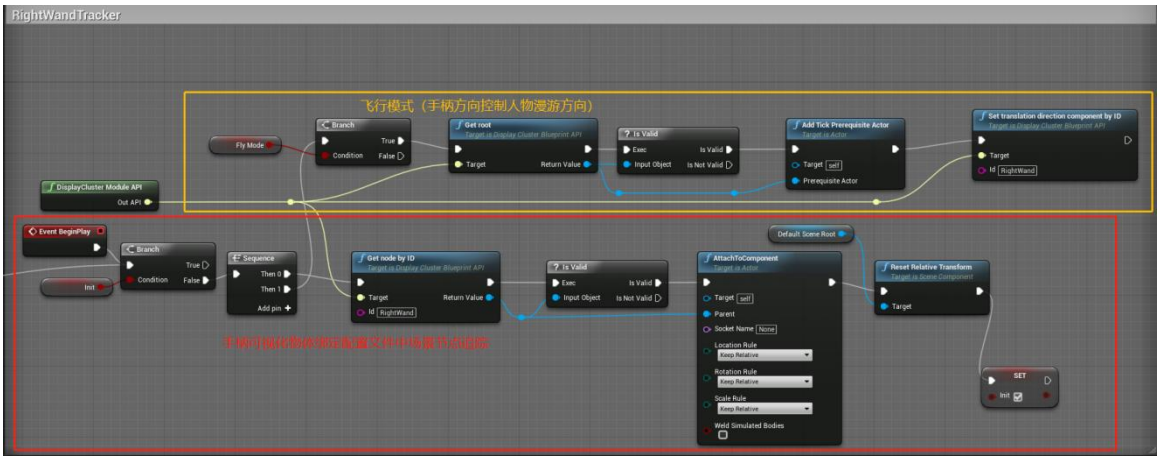

图 3.20 右手柄追踪与控制

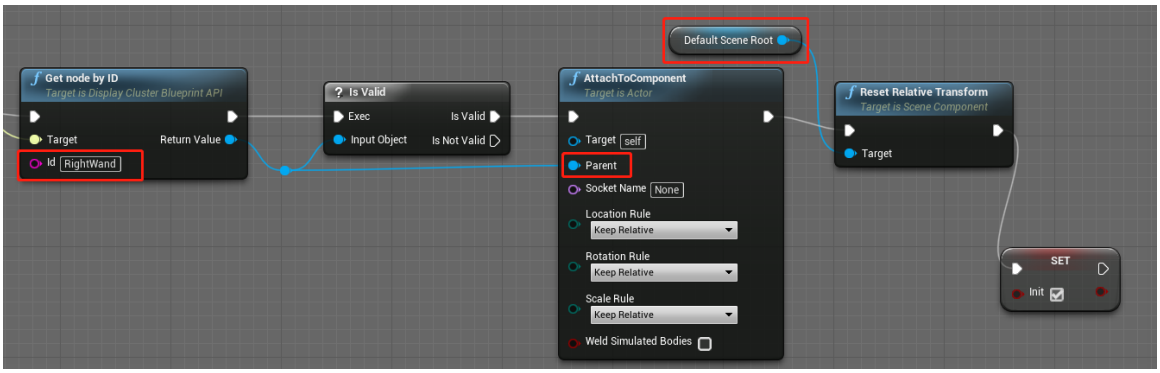

图 3.21 右手柄实现可视化追踪

注：在 4.24 及后续版本中，Get Node By ID 功能被细化，此处用 Get Component By ID 通过 ID 获得 Scene Component。

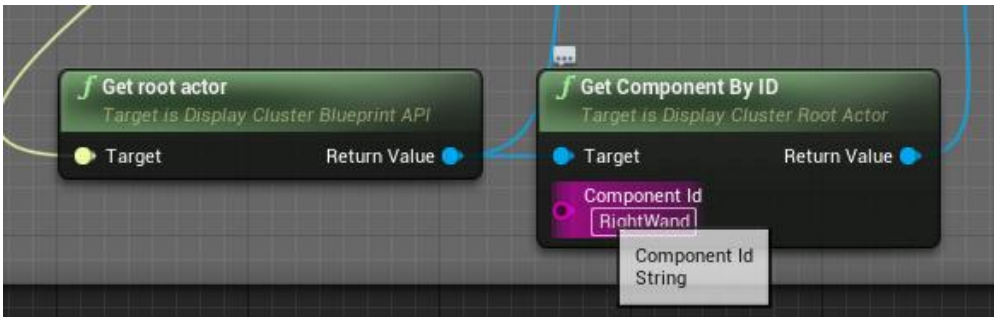

图 3.22 通过 ID 获得 Scene Component

Get Node by ID（通过 ID 获取场景节点）主要参数：

- Id：配置文件中，场景节点配置所对应的右手柄的场景节点 id。

```
[scene_node] id="RightWand" loc="X=0,Y=0,Z=0" rot="P=0,Y=0,R=0" parent="Origin" tracker_id="RightTracker" tracker_ch="1"
```

图 3.23 配置文件中右手柄场景节点配置

注：若要开启飞行模式（手柄方向控制人物漫游方向），则需将 Flymode = True（默认是 false）。

3.5 射线检测

在 BP_Right 蓝图中，上个步骤里我们新建了右手柄可视化的模型和射线，这里我们通过右手柄的位置和右手柄的方向来完成右手柄射线检测功能，如下图所示：

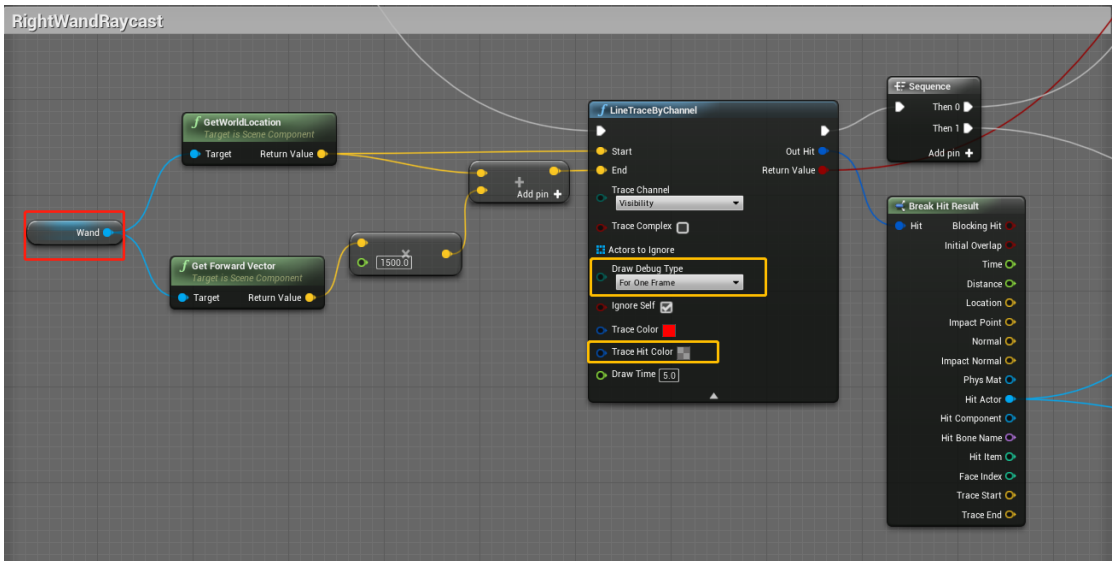

图 3.24 右手柄射线检测

LineTraceByChannel（射线检测并返回追踪命中的首个物体）主要参数：

- Draw Debug Type：选择 For One Frame 可以动态查看射线命中情况和动态控制射线长度。
- Trace Hit Color：选择透明颜色，可以射线命中物体后，延长射线出现的颜色视为不可见。

3.6 交互示例

与物体交互

1. Hover 状态，物体颜色转变。

- 射线悬浮在物体上时，颜色改变。
- 射线离开物体时，物体颜色恢复为原来默认的颜色。

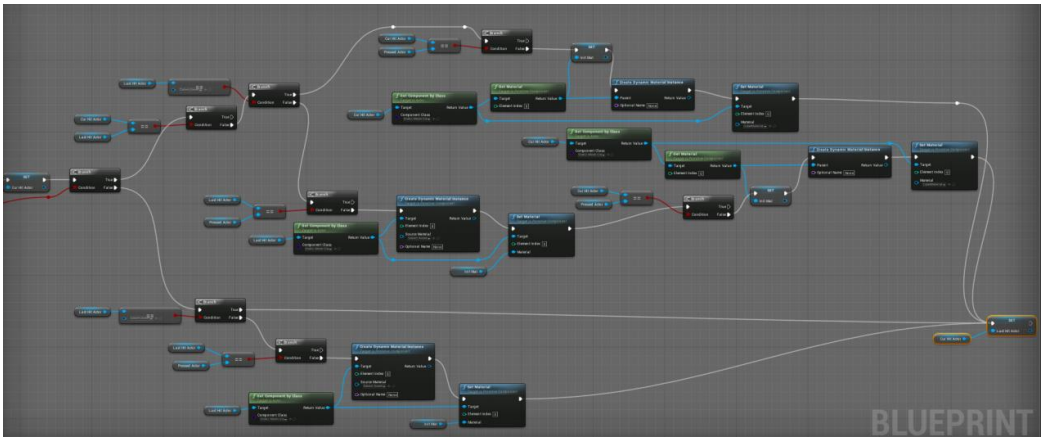

图 3.25 hover 改变物体颜色

2. Trigger 按键触发物体颜色转变并打印物体信息。

按下右手柄 trigger 键且检测到物体时，变换另一种颜色，按键抬起，恢复为 Hover 状态，且打印检测的物体信息。

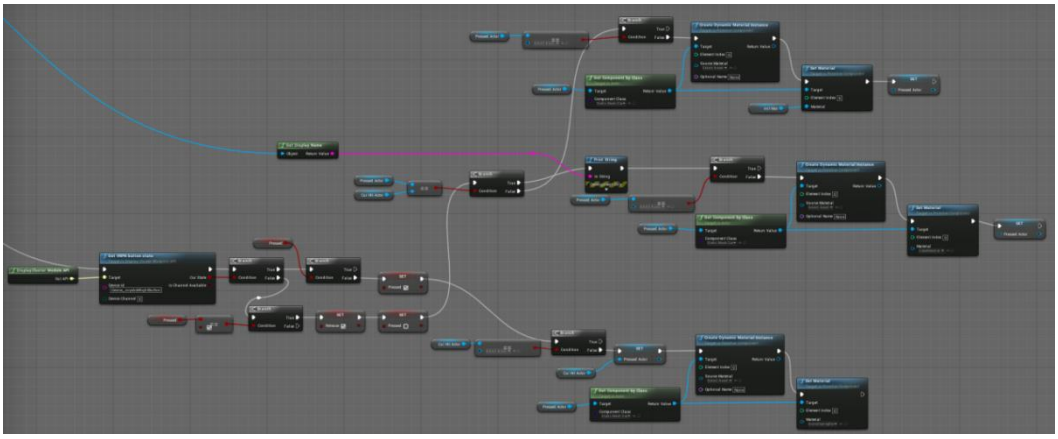

图 3.26 press 改变物体颜色

图 3.27 节点变更

4 注意事项

1. 将新建的蓝图拖到场景中。

2. 若测试 VRPN 数据时没有反应，找到蓝图所对应的 Input，将 Auto Receive Input 的值修改为 Player0。

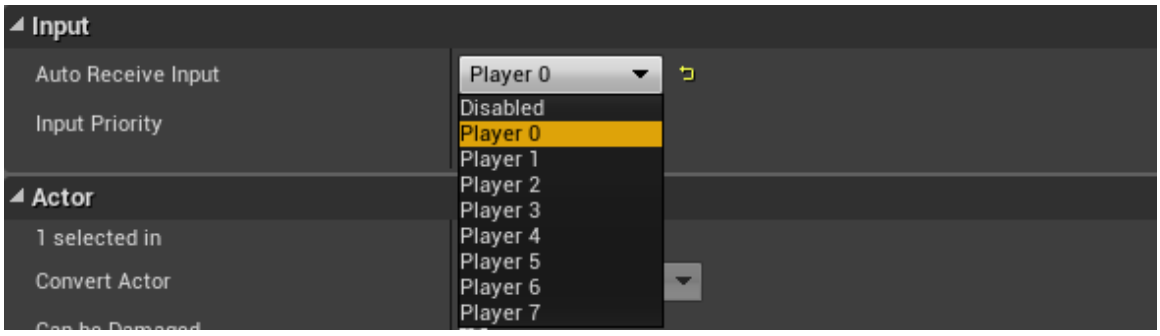

图 4.1 Input Priority

3. 若在 UE4 编辑器运行后，出现以下报错，请将上述通过 Link VR 启动 UE4 案例后，生成的 single.cfg 拷贝到：工程目录\Content\ExampleConfigs 下。

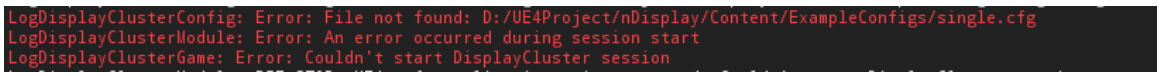

图 4.2 无相应的配置文件报错

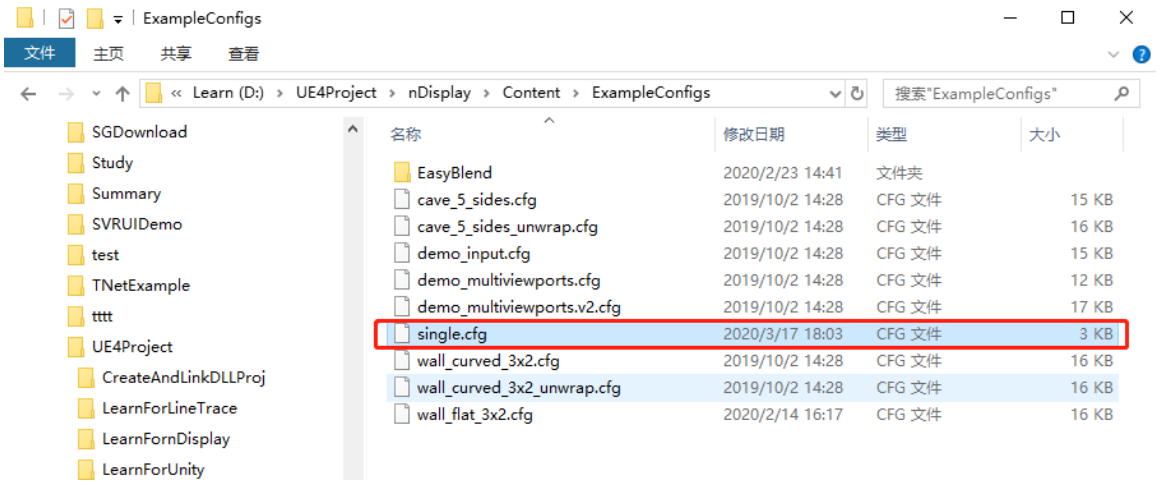

图 4.3 配置文件所在文件夹

4. 漫游方式很多种，[3.3 人物漫游]展示的是手柄摇杆横轴控制人物旋转，纵轴控制人物位移，开发者可以根据自身需求进行开发。

5. 以上所展示的开发过程仅供参考，更多的开发过程请查询 UE4 官方文档。

上海曼恒数字技术股份有限公司
Shanghai Graphic Design Information Co., Ltd
上海市松江区顺庆路500号河图公园
官方网站:www.gdi.com.cn
全国服务热线:400-233-766