高级框架-核心 4.1 文档

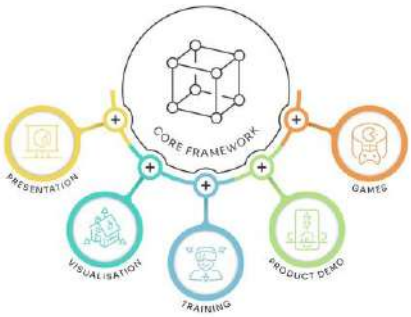
HumanCodeable

2021 年 5 月 5 日

F

内容

1 介绍



与旧版本相比，高级框架的 4.0 版本带来了一些主要的创新。为了解决用户在 3.1 版本中遇到的各种问题，我们决定将框架拆分为一个精简的核心和一些可以单独购买的扩展。核心框架可以单独使用，也可以与高级框架扩展组合使用，这些扩展为特定类型的应用程序提供额外的功能和示例。到目前为止，五个 AF 扩展计划用于以下用例:

- 演示-提供举办讲座或录制视频的工具和环境，包括完全设置的教室级别和工作室级别作为示例。
- Arch Viz -总结了所有的功能来设置，显示和修改从一个房间到整个建筑的建筑可视化。
- Showcase -专注于一个复杂的 actor，它可以配备大量用于定制和交互的功能。
- 训练-包含工具和例子，以设置训练模拟。
- 游戏-提供了一个有用的资产选择，以创建视频游戏和游戏般的体验

此外，一些多用途资产，如智能手表或键盘，以高级框架实用程序的形式分开。AF 实用程序可以与核心框架和任何 AF 扩展自由组合。到目前为止，以下 AF 公用设施已计划就绪:

- 智能手表-装备 VR 兵与手表一样的 UI，显示信息和控制显示 UI 元素。
- 键盘——提供一个可以在应用程序中用作键盘的角色
- 画笔-提供一个可以创建彩色样条的 actor
- 小地图-组件，材料和 UI 设置和显示 2D 和 3D 小地图，包括多个楼层的小地图。
- 网络浏览器- UI 和函数访问和显示网页

将框架划分为核心、扩展和实用程序，使每个人都可以购买适合自己特定用例的框架修改。此外，它还允许我们包括更多的手工制作的，准备使用的示例，以方便使用框架，即使是经验不足的用户。

1.1 核心框架

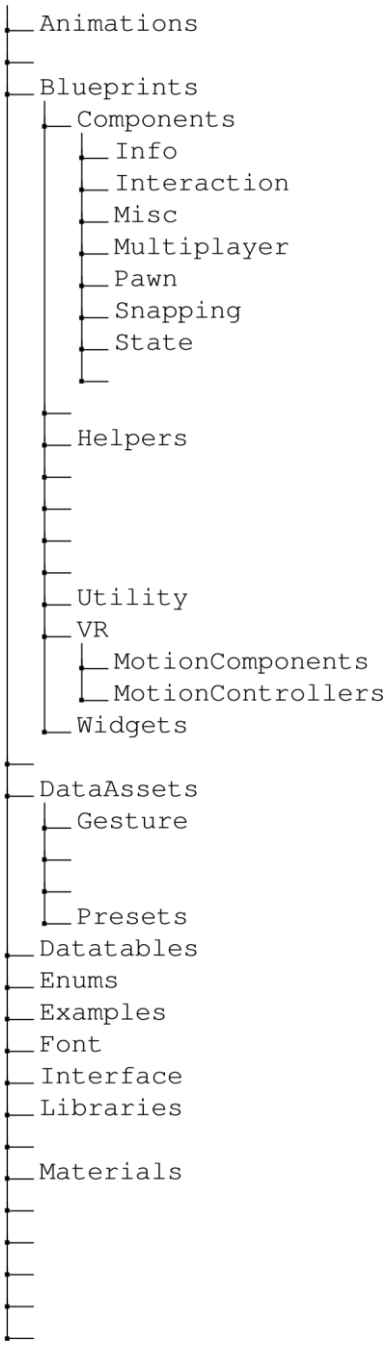
核心框架提供了所有基本功能。无论是 AF 扩展还是 AF 实用程序都不能在没有 AF 核心的情况下运行。AF 核心提供:

- 超过 20 个组件来实现各种功能。
- 兵调整到虚拟现实，桌面和移动环境。
- 各种 UI 元素，包括 HUD、菜单和托盘
- 所有基本的游戏和信息文件，助手，管理器和小部件
- 11 个示例地图，展示 AF 核心的基本工具

1.2 文件结构-核心框架

AFCore

AF 核心有一个预定的文件夹结构，可以作为基础



对于任何项目。您体验的每一个新元素都可以明确地集成到结构中，以保持即使是大型项目的整洁和易于导航。大多数文件夹都是不言自明的:命名为动画、材料、网格、粒子等的文件夹将包含各自的文件类型。接下来，您将看到本文档中最重要的文件夹列表以及它们所包含的内容。

蓝图包含了框架提供的大部分逻辑。用户界面

Components 包含 AF Core 的自定义组件。有关更多游戏信息，请参阅第 3 节。

水平

Game 包含了所有的游戏类(参见章节 2)，如游戏模式 Save 或玩家控制器。兵

helper 包含支持组件或 UI 中其他角色的角色。helper 是自动生成的，不需要任何设置。

Level 包含所有用于基本级别设置的文件，如 MapInfo Actor 或用于高亮显示的 Post Process Volume。

Curves pawn 包含了 AF Core 的 pawn 父类。

uicon 包含框架提供的所有 ui 参与者，包括

关卡托盘或信息窗口(参见第 8 节)。

面板

VR 包含 5.3.3 节和 5.3.2 节所述的运动控制器和运动组件。

小部件包含所有定制的 AF 核心小部件

数据资产包含一些用于存储数据的对象(section ??)

示例包含示例地图中显示的所有网格、材料和蓝图。

地图

接口包含所有控制信息传输的接口

在组件之间或一般情况下。网格

库提供了一组普遍可访问的函数。声音

Splash maps 包含地图模板。

结构体

纹理

2 基本知识

2.1 游戏类

游戏类实现了应用程序运行的所有基本功能。这包括关卡转换、兵导航、设置和其他基本数据。

游戏模式

声明除游戏实例之外的所有其他游戏类。

在多人游戏体验中

蓝图:BP_GameMode_Main

设置:不应触摸

游戏实例

游戏实例具有独特的能力，即使在级别更改后也能保持其状态，这使得存储持久数据非常有用。因此，它是存储语言、图形或音频等设置的最佳场所。此外，游戏实例存储当前关卡的数据资产。

蓝图:BP_GameInstance_Main

设置:所有变量在游戏实例创建时自动设置

球员控制器

玩家控制器刷出，占有和导航棋子，模式只能由服务器访问，而不是客户处理关卡转换逻辑以及暂停。

蓝图:BP_Player_Controller_Main

设置:

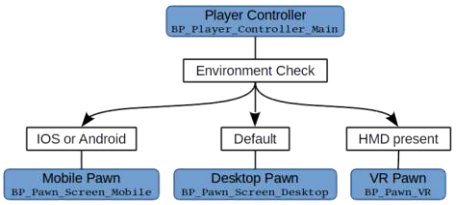
•兵选择-决定如何在开始时选择兵

应用程序 图 1 所示。启动和棋子选择。

- 动态——根据环境检查启用棋子选择，如图 1 所示
- forceevr -选择在关卡数据资产中指定的 VR 兵，不考虑环境
- ForceDesktop -选择在关卡数据资产中指定的桌面棋子，而不考虑环境
- ForceMobile -选择在关卡数据资产中指定的移动棋子，而不考虑环境
- OnlyFirstVR -定义在多人 VR 应用程序中，只有玩家索引为 0 的玩家在 VR 中

球员状态

玩家状态的所有其他设置都是



端。因此，我们从游戏模式中剥离了许多功能，并将它们分配给其他实体。玩家状态处理并提供关于每个玩家的信息，因此对于多人游戏尤其有用。

蓝图:BP_PlayerState_Main

设置:

•球员索引-球员的号码

现在外包给 playerinfo 组件，这在 3.7.1 节中介绍过

2.2 水平设置

框架的每个关卡都被封装在一个世界地图中，该地图整合了所有的地图以及关卡的加载。此外，框架依赖于每个层次上的一些基本参与者来正常工作。我们需要在每个新关卡中设置这些内容。

MapInfo 演员

指的是包含每个关卡基本信息的数据资产(DA)，如使用哪些兵或加载哪些关卡。

蓝图:BP_MapInfo 设置:

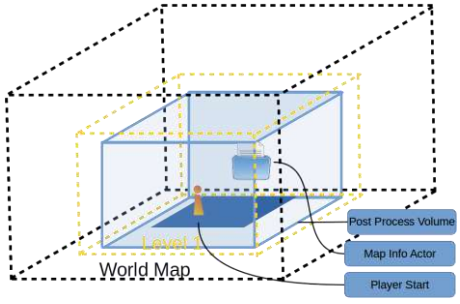


图 2。最低级别设置。

- 级别 DA——指级别数据资产(见 7.1.2 节)

球员的位置

指定玩家兵在开始关卡时产生的位置和旋转。

蓝图:BP_PlayerPosition

设置

- **玩家索引-限制在这里生成的玩家兵到一个玩家索引**
- **默认-在多人游戏体验中为所有玩家转换默认刷出点的玩家位置**

后处理卷

处理后期过程高亮。

不需要，如果没有后期处理高关卡中使用了灯光。

蓝图:BP_PostProcess

设置:不需要额外的设置

2.3 导航

框架的导航元素主要连接到对虚拟现实环境非常重要的传送系统，因为在虚拟现实中正常的运动很容易导致晕车。传送导航元素主要用于限制玩家可以传送的区域

出现。

如果你不想使用传送移动兵，你可以通过包含一个传送区或传送网

传送区域

指定玩家可以安全地传送到的更广阔的区域。传送区域是具有传送和选择组件作为可能的交互方法的平面 actor。

蓝图:BP_TeleportArea

设置:

- 网格材质-定义传送区域的外观
- 可见性-使玩家能够感知应用程序中的传送区域。
- 调整搜索距离-传送区域的痕迹搜索一个地面行动者并将兵传送到的最大距离。

图 3。传送区域

传送点

指定玩家可以传送到的有限数量的位置。

一个传送点基本上是一个包含传送组件(参见 3.1.6 节)和许多其他交互组件的参与者。

蓝图:BP_TeleportMesh

设置:

•使用旋转-如果启用，强制兵采用传送网格 actor 的旋转

传送选择 vs.普通传送

两个传送元素都可以通过传送组件或选择组件进行切换。

- 传送选择——传送是使用选择组件作为交互组件的自定义函数实现的，避免了传送运动组件和额外的输入
- 普通传送-传送是使用传送组件和传送运动组件实现的，提供了更详细的控制和设置

2.4 改变水平

框架提供了许多机制去顺利加载或改变关卡。这包括过渡和介绍屏幕，还包括在一个地图中编排流关卡的管理器。

2.4.1Intro 水平

应用程序中加载的第一个级别是介绍级别。该框架提供了一个完全设置的介绍级别，可以用作模板。它包含:

- 介绍屏幕-显示介绍小部件和处理过渡到下一个级别
- 玩家位置-指定玩家兵被衍生的位置
- 映射信息参与者-包含关于介绍级别的信息的级别数据资产
- Skysphere -为介绍关卡提供背景

除介绍屏幕外，介绍关卡中的所有其他角色都属于基本关卡设置所必需的角色。

介绍屏幕

介绍屏幕是介绍级别中处理大部分逻辑的部分。它显示介绍小部件，并切换到下一层

蓝图:BP_IntroScreen

设置:

- 从级别到加载-包含应该加载的下一级别的级别数据资产

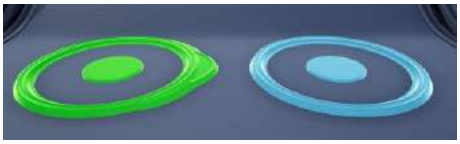


图 4。传送点

2.4.2Transition

每当加载新关卡时，就会自动创建转换对象，并填充关卡数据资产中指定的内容(参见 7.1.2 节)。

蓝图:BP_Transition

对于自定义转换，创建子转换

2.4.3 流式级别管理器

这个管理器包含所有流级别的数组，这些流级别应该可以在映射中访问，以及将一个流级别交换为另一个流级别的函数。

蓝图:BP_Manager_StreamingLevel

设置:

- 可用流级别-数组，包含每个流级别的结构，应该是可访问的
  - Name -包含键和 I18n 数据表，显示为级别名称
  - 图像-包含为关卡显示的纹理
  - 级别名称-流级别映射的全名
  - AR -正在建设中
- 当前加载级别-自动设置为所选级别，无需设置
- 当前加载索引-当前流级别数组中流级别的索引。还确定在启动关卡时加载哪个流关卡

设置简介

- 在世界地图中-将持久级别和所有流级别添加到同一个世界地图
- 在持久级别中-添加流级别管理器的实例
- 在流媒体级别管理器实例上-输入所有可访问的流媒体级别的数据和要加载的第一个流媒体级别的索引
- 访问方法——准备一个调用管理器的负载流级别函数的功能

的转换类，自定义它并将其作为转换类输入到相应级别的级别 DA 中。

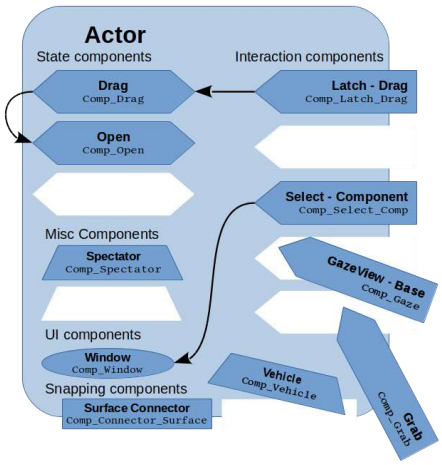
3 个组件

组件可以附加到提供所需功能的参与者类以及参与者类的实例。组件可以自由交换，并且允许自定义角色，而无需构建复杂的类层次结构。我们为 AF Core 创建的所有组件根据其主要功能分为以下类别之一:

- 交互组件——处理应用程序中玩家和参与者之间的交互，包括选择、抓取和其他。
- 状态组件——应用、启动和复制拥有参与者的状态更改。
- 捕捉组件-涵盖所有必要的需求，以允许 actor 在被播放器释放后捕捉到合适的位置。
- 多人游戏组件——整合一组处理多人游戏的有用组件。
- UI 组件——包括 UI 元素的设计、显示和内容。
- 典当组件-为典当提供类似控件的功能。
- 信息组件—保存并提供可以在运行时更改的动态信息。
- 杂项组件——由一组有用的组件组成，这些组件在后台工作，以启用高级功能。

3.1 交互组件

框架的所有棋子和运动控制器都设计为使用交互组件来实现玩家与参与者的交互。每组组件都适用于特定的交互类型，如选择或抓取。尽管所有交互组件的功能不同，但它们都有一些共同的设置。这些包括



主要是参与者和组件的识别，交互应该切换和突出显示。

组件标识

大多数交互组件将玩家输入传输到 actor 的其他组件，以实现所需的功能。这些可以在同一个 actor 上，也可以在另一个 actor 上，或者在两个 actor 上。这有助于触发器的设置。因此，必须能够明确区分单个实例上的单个组件或一组实例上的单个组件。这是由组件定义结构完成的，它包含:

- 要搜索的组件标签-标记每个组件的标签数组

图 5。组件系统

信息框-组件标签

每个组件都可以配备一组标记，其他组件可以使用这些标记引用该组件。这也包括虚幻的固有组件，如静态网格组件。组件标签在协调组件交互中起着至关重要的作用。它们可用于识别参与者上的特定组件(见 3.1 节)或从集合中选择合适的组件/参与者(见 3.3 节)。

图 6。组件定义结构

不涉及交互所切换的受影响的参与者

- 要触发的参与者——参与者的每个实例上的数组，它包含受交互组件影响的级别中每个参与者实例的元素
- “还要触发”自动在参与者本身中搜索“要搜索的组件标签”数组中收集的组件标签。

只能在参与者的实例上设置。

如果交互组件不影响其他参与者，则不需要进行其他设置。

高亮显示

大多数交互组件都支持可定制的高亮显示功能。gazeview 组件是唯一的例外。交互组件上的高亮设置决定参与者是否被高亮显示以及如何被高亮显示:

- 突出显示类型-指定参与者是否突出显示以及如何突出显示
  - 无-无高亮显示
  - 自定义-使用在参与者上实现的函数
  - 后期处理-使用彩色边缘围绕演员网格

图 7。后期流程突出显示

高亮显示，默认设置(见图??)

- 网格-复制演员的网格，并设置不同的材质来创建轮廓效果
- 材料功能-激活一个材料功能改变

材料发光参数用于高亮

不建议用于发射材料。

•高亮颜色-确定后期处理和材料功能高亮的颜色

- 高亮材质——定义材质高亮函数

目前只有一个可用。

- 要突出显示的组件标签-指定应该由标签突出显示的网格。

3.1.1Select 组件

Comp_Select

选择组件实现了应用程序中参与者的选择。然而，它们的主要功能是信息传递。因此，选择组件会不断地与其他组件协作，这取决于选择时应该发生什么。

Comp_Select_Comp
Comp_Select_SelectionMenu

选择组件- Base

作为更指定的选择组件的父类，并可能在使用选择组件的逻辑时实现自定义功能。

蓝图:Comp_Select

设置:

- SelectEnabled—暂时禁用选择组件
- 标识符——允许其他组件区分所选组件
- 组件标签选择-指网格对应的

选择组件

Tomakeallmeshesselectableenter

“没有”。

•支持选择源-为每个输入源包含一个元素，

可以与选择组件交互的

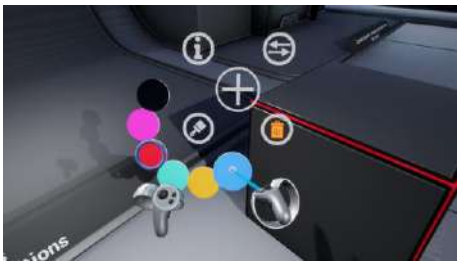
  - 无-不接受输入
  - 激光-接受激光运动控制器作为输入
  - 屏幕-接受来自所有屏幕类型兵的输入
  - 触摸-通过触摸接受来自手部运动控制器的输入

- 选择声音-当选择被切换时播放的非循环声音
- 悬停声音-非循环的声音，播放时，选择开始可用的播放器
- 悬停声音-当选择停止时播放的非循环声音，可供播放器使用。
- 按下选择-当分配给选择交互的键被按下时切换
- 释放选择-当分配给选择交互的键被释放时进行切换
- 高亮选择-当输入设备悬停在参与者上时切换

几乎不可能用
触摸屏输入设备。

选择组件-选择菜单

实现了选择菜单 UI 元素(见 8.3.1 节)，该元素由一组按钮组成，这些按钮将选择组件与一组组件(主要是状态组件)连接起来。



蓝图:Comp_Select_SelectionMenu

设置:

- 选择按钮-使用运动控制器为选择菜单的每个按钮包含一个元素。兵不断地用 gazeview 组件在兵头
  - 按钮类——决定生成哪个按钮
  - 组件定义——指定应该由按钮切换的状态组件
- 排序类型-决定选择菜单的按钮如何在空间上排列
  - 圆形——在圆形上排列按钮(参见图 8)
  - 行——按行排列按钮(参见图 76)
  - 行和列——按行和列排列按钮(参见图 77)

图 8。圆形按钮排列

自动添加一个关闭选择菜单的按钮。

可用的按钮

- BP_Selection_Delete
- BP_Selection_Bool_Info
- BP_Selection_Bool_Light
- BP_Selection_Bool_Open
- BP_Selection_Bool_Play
- BP_Selection_Spawn_Sub_Visual

选择组件-组件选择

将选择直接传输到一个组件。实现了预期的功能。

蓝图:Comp_Select_Comp

设置:

- 组件定义——指定由选择选项切换的组件

3.1.2GazeView 组件

允许玩家通过视觉与拥有的演员进行互动

的直线上检查演员，并激活组件。

Comp_Gaze
└ Comp_Gaze_Trigger
└ Comp_Gaze_SelectionButton

Gazeview 对于不熟悉 VR(或操控控制器)的玩家来说是一款很棒的工具，因为它能够让玩家通过观看与角色进行互动。它是一个特别好的匹配 VR 手运动控制器，因为 gazeview 不依赖于激光进行交互

actors out of reach of the player.

GazeView 组件-基础

作为更专门化的 gazeview 组件的父类，并作为使用 gazeview 组件逻辑激活自定义函数的基础。

蓝图:Comp_Gaze 设置:

- 标识符——允许其他组件区分远景视图

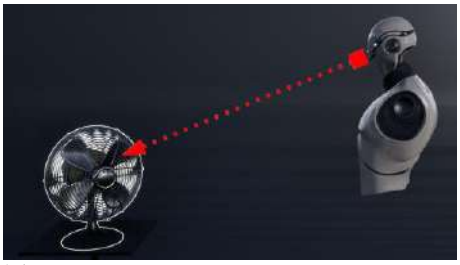


图 9。GazeView

组件

•组件标签凝视-指网格对应的

gazeview 组件

•Component Enabled -暂时禁用 gazeview 组件

声音:

- 悬停声音-当凝视开始时播放的非循环声音
- 悬停声音-凝视结束时播放的非循环声音。
- 凝视视图-当玩家凝视凝视组件指定的网格时切换

Gazeview Component -组件

将凝视信号传输到另一个组件(通常是状态组件)。Gazeview Component 组件还实现了带有进度功能的触发器，即在玩家继续观察 actor 时显示进度条，并在进度条已满时切换状态组件。

蓝图:Comp_Gaze_Trigger

设置:

- 组件定义——指定凝视切换的组件
- Toggle -指示 gazeview 组件充当一个更改开关

进度条:

- 使用进度条——为组件启用进度条功能
- 凝视超时-定义玩家停止注视行动者后多长时间凝视信号终止
- 进度持续时间-以秒为单位确定进度条完全填充所需的时间
- 进度条小部件-决定进度条看起来像什么
- 声音进度条-在进度条填充过程中循环播放的声音。
- 凝视触发-当进度条被填满时切换

- Take Root As Location -指示 gazeview 组件在根位置而不是影响位置生成按钮
- Button -声明所生成的选择按钮的按钮类

要衍生的按钮的凝视和组件定义

3.1.3Grab 组件

抓取组件可以让棋子或运动控制器抓取和移动参与者。

抓取组件-底座

Tomakeallmeshesgaze-ableenter

“没有”。

图 10。GazeView 触发器与进度

图 11。带有选择按钮的 Gazeview。

Gazeview 组件-选择按钮生成一个按钮，切换一个组件或其他组件。

蓝图:Comp_Gaze_SelectionButton

设置:

- 凝视超时-定义玩家停止注视行动者后多长时间凝视信号终止
- 位置组件的标签——指的是场景组件，它决定选择按钮在哪里生成

如果没有位置组件，则根据进一步的设置，在影响位置或根位置产生选择按钮

目前，基本的 grab 组件是唯一完整设计的 grab 组件，并实现了所有必要的功能。但是，我们希望像大多数其他交互组件一样，将其他专门的抓取组件合并到框架中。

蓝图

:Comp_Grab 行为:

- 抓取类型-决定在抓取过程中参与者的行为
  - 正常- actor 连接到运动控制器或冲击位置
  - 物理句柄- actor 通过物理句柄连接到卒或运动控制器
- 发布类型——定义参与者在发布时的行为
  - 自由放置-释放后，演员停留在任何地方，目前
  - 物理-在释放时激活角色的物理。
- 应该保持直立-自动旋转被抓取的 actor 直到它的 z 轴与世界 z 轴平行，与手运动控制器无关

听起来

- 悬停声音-当抓取开始可用时播放非循环声音
- 悬停声音-当抓取停止可用时播放的非循环声音
- 抓取声音-非循环的声音播放时，抓取互动开始
- 释放声音-当抓取互动结束时播放非循环声音

触觉

-强度-作为一个倍增器，以微调触觉反应

提前

- Snap to Controller -将 actor 附加到控制器上，并根据下面的关键部分更改控件
- 相对控制器位置-定义参与者的相对位置
  到控制器后的 snap

条件

- 可以被抓住-暂时防止演员被抓住
- 自动拾取-自动抓取演员的手掌附近的运动控制器，不相关的任何其他控制器或非 vr 环境
- 到套接字的最大距离-决定在控制器到套接字的哪个距离自动释放抓取
- 取而代之的是抓取附加的 Actor -防止在试图抓取 Actor 时意外地分离附加的 Actor
- 抓取标签——指定参与者的网格应该是可抓取的
- 允许多重抓取-允许多个参与者(兵或运动骗局

  Trollers)抓住演员
- 允许拾取(如果已经拾取)-允许另一个兵或运动控制器在物体被拿着时抓住物体
- 要求点击抓取标签-确保演员只能被抓取
  在抓取标签指定的网格上。

键

控件的自定义控件在事件图中覆盖按键或函数-阻塞按键或函数的输入

演员

图 12。应该保持正直的榜样。

捕获组件可以在发布时修改参与者的行为。参见
3.3 获取更多信息。

- 触觉抓取-从抓取开始和保持期间控制控制器的触觉响应
  - 触觉曲线-描述从抓取交互开始的触觉反应强度
  - 持续时间-决定触觉反应活跃的时间

拉伸指定的触觉曲线
并按规定的时间压缩。

只与激光控制器相关。

多重抓取和拾取如果已经选择
通常只有在一起才有意义

只与激光控制器相关。

只适用于与 snap

—AllKeys—自动清除并阻止所有键的输入 控制器功能

- 占用的键-清除和阻止所有输入的键的输入
- 已占用函数-清除和阻塞所有输入的控制器函数

为澄清

- 被占用的键用于用参与者的自定义函数替换键的控制器函数，例如枪的射击函数。
- 被占用函数用于阻塞和清除在 actor 被占用时不应该使用的控制器函数。

事件

- 抓取 Actor 拾取-当 Actor 被玩家抓取时切换
- 抓取 Actor 释放-当 Actor 被播放器释放时切换
- 抓取拇指杆轴-允许玩家在抓取时使用拇指杆进行额外的输入
- 抓取键输入-允许玩家在抓取时使用面键进行额外的输入
- 抓取高亮显示-如果参与者被抓取组件高亮显示，则切换。用于自定义高亮显示。
- 抓取触发轴-允许玩家使用百分比的触发按钮按下额外的输入，同时抓取
- 抓握轴-允许玩家使用百分比的抓握按钮按下额外的输入，而抓

3.1.4Latch ComponentsComp_Latch

抓取和锁存之间的区别在于锁存 motionComp_Latch_Drag
Comp_Latch_Physics 控制器或 pawn 附加到网格上，限制其移动

Comp_Latch_Simple

网格的移动可能性。

锁存组件-底座

作为所有锁存器组件的父组件，并用于实现自定义功能。

蓝图:Comp_Latch

设置:

- 可以锁定到-暂时禁用闩锁组件
- 最大连接距离-定义控制器与网格之间的距离，其中可以找到套接字
- 自动附加-自动锁定到演员附近的手掌

手部运动控制器 与其他控制器无关

•组件标签要附加到-指定参与者的哪个网格 non-VR 环境。

控制者或卒应该抓住

- 允许同时锁存-允许多个参与者锁存到网格上
- 最小分离距离-定义插销自动中断的插座和控制器之间的距离

听起来

- 悬停声音-当抓取开始可用时播放非循环声音
- 悬停声音-当抓取停止可用时播放的非循环声音
- 闩锁声音-当抓取交互开始时播放的非循环声音
- 解锁声音-当抓取互动结束时播放的非循环声音

触觉

- 触觉抓取-从抓取开始和保持期间控制控制器的触觉响应
  - 触觉曲线-描述从抓取交互开始的触觉反应强度
  - 持续时间-决定触觉反应活跃的时间

根据指定的持续时间拉伸和压缩指定的触觉曲线。

强度-作为一个倍增器来微调触觉反应

- Actor 已闩锁-闩锁启动时切换
- 释放锁存器-当锁存器被终止时切换
- 锁存高亮显示-当可锁存网格悬停时切换
- 锁定拇指杆轴-允许玩家在锁定时使用拇指杆进行额外的输入
- 锁存键输入-允许玩家在锁存时使用面键进行额外的输入
- 锁存触发轴-允许玩家使用百分比的触发按钮按下额外的输入，而锁存
- 锁存抓地轴-允许玩家使用百分比的抓地键按下额外的输入，而锁存

锁存组件-拖动

将锁存组件连接到拖动组件，以设置滑块、按钮、杠杆和阀门。

蓝图:Comp_Latch_Drag

设置:

- 拖拽标签——指定闩锁组件应该与哪个拖拽组件交互

闩锁组件-物理生成一个物理句柄来实现闩锁

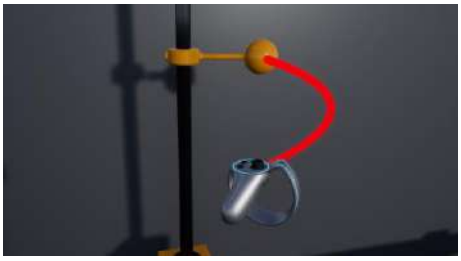


图 13。旋转拖动样条。

蓝图:Comp_Latch_Physics 设置:没有额外的设置

3.1.5Overlap 组件

重叠组件支持触发器的实现，这些触发器对不是棋子的参与者做出反应。最突出的是压力板。

重叠分量-基

作为所有重叠组件的父类，并作为实现自定义函数的一种手段。

蓝图:Comp_Overlap

设置:

- Enabled -允许暂时禁用组件
- 碰撞的组件标签——指的是切换组件的碰撞量
- 允许的参与者——包含一个应该触发重叠事件的参与者数组
- 允许的对象——包含一个应该触发重叠事件的对象数组
- 允许的组件标签——包含一个标签数组，用于指定参与者需要使用哪个标签来触发重叠事件
- 标识符——将组件标识为其他组件的触发器源

但是，重叠分量是可以的
也可以处理与卒的重叠交互。

Comp_Overlap
Comp_Overlap_Comp
Comp_Overlap_Drag
Comp_Overlap_Select

听起来

- 声音重叠-当在设置中指定的演员、对象或组件开始重叠时播放的非循环声音
- 声音重叠-当在设置中指定的演员、对象或组件停止重叠时播放的非循环声音

事件

- 参与者开始重叠-当设置中指定的参与者、对象或组件开始重叠时切换

–

- 参与者结束重叠-当在设置中指定的参与者、对象或组件停止重叠时切换

重叠分量-分量

将重叠组件的信号传输到 actor 上的一个组件(图 14)。

蓝图:Comp_Overlap_Comp

设置:

- 组件定义——定义重叠组件应该触发的参与者和组件
- 标识符——由重叠组件传输，作为起源证明

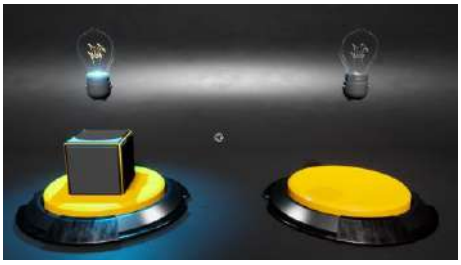


图 14。Overlap Component 组件。

- 触发切换-使重叠组件充当一个改变开关

重叠组件——拖动将重叠组件连接到一个拖动组件(图 15)。

蓝图:Comp_Overlap_Drag

设置:

- 拖动组件的标签——指定重叠组件应该与哪个拖动组件交互。

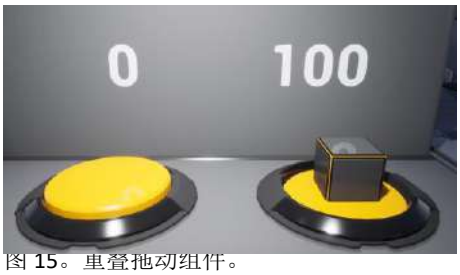


图 15。重叠拖动组件。

3.1.6 传送组件使行动者成为传送移动交互的目标。

蓝图:Comp_Teleport

设置:

启用-暂时禁用传送组件

从普通传送切换到传送

在传送点或区域上选择禁用传送组件，并启用选择组件，反之亦然。

- 传送持续时间-以秒为单位确定传送所花费的时间
- 传送淡入-定义相机如何处理传送
  - 不褪色-禁用瞬移的相机褪色
  - 淡入淡出-实现相机淡入在开始和在传送结束
  - 淡入-实现了一个相机淡入只在传送的开始
  - 淡出-实现了一个相机淡出仅在结束

传送

相机淡入与充分没有意义

- 传送运动-决定在传送过程中兵的移动有多少显示给玩家
  - None -表示没有移动

    指示-显示传送的前几秒以指示移动
  - 充分运动-显示充分的运动

运动传送运动设置。

- 传送等级-调整棋子的哪一部分被传送到所选择的位置
  - 地面-传送兵，使角色兵的脚在选定的传送位置

不适合非角色兵。

- 眼睛水平-传送兵，使相机在选择的位置。
- 旋转类型-指定在传送过程中如何处理兵的旋转。
  - 力量旋转-防止来自传送控制的任何影响，并用拥有传送组件的 actor 的旋转覆盖兵的旋转
  - 允许自定义旋转-允许从预设的数据资产(见 7.1.1 节)传送旋转，以确定传送后兵的旋转
  - 不旋转-保持兵的原始旋转

听起来

- 声音传送-执行传送时播放非循环声音
- 悬停声音——当传送轨迹击中带有传送组件的参与者时播放的非循环声音
- 悬停声音——当传送轨迹离开带有传送组件的参与者时播放的非循环声音

3.2 状态组件

状态组件实现了参与者的主要功能，如切换灯光或执行视觉更改。它们通常由交互组件或另一个状态组件激活，并实现大量的公共功能。

源信息

源信息是一个结构体，它标识了切换状态组件的交互组件。每当交互组件切换它时，它就被传输到状态组件，并且可以被事件调度程序访问，以便在自定义函数中使用。

- 交互棋子——发起交互的棋子
- 引用位置-通常是影响位置
- 交互参与者——发起交互的运动控制器或棋子
- 源对象——拥有交互组件的参与者
- 标识符——标识交互组件的标签

组件标签

组件标记充当交互组件使用的状态组件的引用。状态组件可以有多个标记以及共享标签，以平行切换。只要有一个标签符合交互组件中的规范，状态组件就会被切换。

接口
每个状态组件实现一个接口来标准化状态组件和参与者或交互组件之间的信息传输(另见第 4 节)。

图 16 - 信息来源

交互的卒、参与者和源对象只能在实例中访问。

3.2.1Active 组件

激活或禁用参与者的功能，例如在电视屏幕上播放电影

蓝图:Comp_Active

接口:Interface_StateComponent_Binary

设置:

- Value -确定 active-Boolean 的起始值

听起来

- 声音停用-当激活状态由 true 变为 FALSE 时播放非循环声音
- 声音激活-当激活状态从 FALSE 变为 true 时播放非循环声音。
- 活动状态更改-每当组件的当前状态更改时切换
- 在编辑器中，活动状态已更改-在编辑器中更改组件的当前状态时切换

3.2.2 颜色组件在给定的 actor 上启动颜色更改。

蓝图:Comp_Color

接口:Interface_StateComponent_Color

设置:

- Value -决定起始颜色
- 可用颜色——包含一个元素，对应组件应该支持的每种颜色

听起来

- 颜色变化的声音-非循环的声音播放每当一个颜色变化是启动:
- 颜色状态改变-当当前颜色改变时切换

–

- 在编辑器中颜色状态更改-在编辑器中当前颜色更改时切换

3.2.3Drag 组件

允许 actor 的一个网格相对于所有其他网格移动，以创建滑块、按钮或更复杂的触发器或谜题。

蓝图:Comp_Drag

接口:Interface_StateComponent_Trigger

设置:

- 拖动类型-定义拖动的空间曲线
  - 线性-网格沿着平行于 x, y 或 z 轴的直线拖动，就像一个滑块

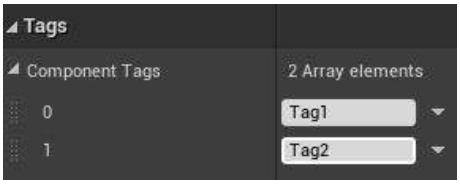


图 17。组件标签。

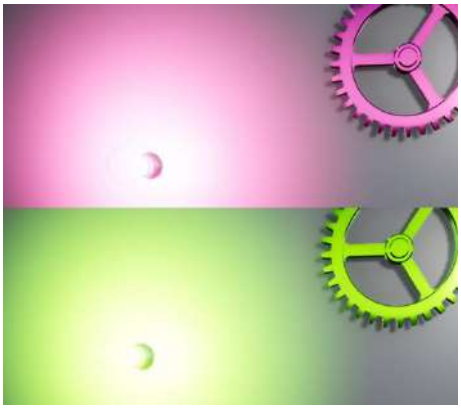


Figure 18. Color Component Example.

HowTheDeleteComponentSup-portsSpawning

The animation provided by the delete component can also be used to support spawning an actor. In this case the delete component only provides the animation which is toggled directly after spawning the actor.

角度-网格沿 x, y 或 z 轴的角度被拖动，就像一个杠杆

- 样条-网格沿着一个定制的样条被拖动，可以描述曲线和弯曲(见图)
- 旋转样条-网格沿着自定义样条拖动，并可以围绕该样条旋转(见图)
- 旋转-网格像阀门一样围绕 x, y 或 z 轴旋转

- Snap Type -定义了被拖动的网格在释放时的行为
  - 自由移动-网格保持在它所在的位置
  - 抓拍到片段-网格移动到最近的部分，并保持在那里
  - 重置-网格移动到拖动的起始位置

图 20。样条阻力。

- 起始位置-确定刷出或关卡开始时可拖动网格的位置
- 抓取速度-定义被拖动的网格抓取的速度

释放后的预期位置。

不相关的自由运动 snap 类型。

- 跟随速度-定义被拖动的网格跟随运动控制器或棋子的速度
- 选择规则——确定选择组件如何与

拖动组件

- 切换极端-选择移动可拖动的网格之间的最高和最低部分
- 步进模-选择移动可拖动的网格到下一个

如果使用锁存器组件进行交互对于拖动组件，此设置不适用。

高于电流的一段

如果达到最高部分，则选择

–

-下步模-选择移动可拖动网格到下一个低于当前的部分

重置拖动

- Toggle -使组件工作像一个改变开关
- 反向百分比-命令组件反向计算百分比

如果到达最低部分，则选择将重置拖动

- 标识符——将组件标识为其他组件的触发器源
- 组件定义——允许拖动组件切换另一个类似于交互组件的状态组件。更多信息见 3.1 节。
- Sections Set -包含一个定义相对的浮点值数组

剖面值表示相对位置。默认情况下，组件将这些转换为百分比，其中最低部分值为 0%，最高部分值为 100%

拖动部分的位置。

节集需要至少 2 个值

- 循环声音-在整个拖动活动期间播放
- Component Tag to Control -通过声明它的标签 Events 来指定应该被拖动的网格:
- 拖动状态更改—当拖动的部分发生更改时，单次切换一次
- 拖动状态改变常量-只要可拖动的网格被抓取和/或移动，就会切换每一个标记
- 抓拍完成-可拖动的网格被释放后的切换，并根据抓拍类型重新定位

开始和结束值。

3.2.4Highlight 组件

用于突出显示独立于交互组件的参与者，以指示任务目标(参见图 21)。

蓝图:Comp_Highlight

接口:Interface_StateComponent_Binary

设置:

- 类型-定义突出显示的样式
  - 无-无高亮显示
  - 自定义-使用在参与者上实现的函数
  - 后期处理-在演员网格周围使用彩色边缘来突出显示
  - 网格-复制演员的网格，并设置不同的材质来创建轮廓效果
  - 材料功能-激活一个材料功能改变材料发光参数高亮

不建议用于发射材料。

- 高亮标签-只允许高亮给定标签的网格

- 使用高亮标签-通过标签高亮选定的网格
- 高亮材质-定义复制网格上的材质

为网格高亮创建

•颜色-定义后期处理，网格和材料功能的颜色

对突出显示类型没有影响除了网格突出。

突出的声音:

- 声音开始高亮-当高亮被激活时播放非循环声音
- 声音停止高亮-播放非循环声音，当高亮被禁用时
- 声音循环-在高亮显示期间循环播放声音

状态

- Value -定义在关卡开始时高亮是激活的还是不激活的，可用于在运行时调节高亮。
- 高亮-切换自定义高亮函数

3.2.5Name 组件用于保存和复制名称变量。

蓝图:Comp_Name

接口:Interface_StateComponent_Name

设置:

- Value -名称的起始值
- 可用名称-定义一个由组件 Events 支持的名称数组:
- 名称状态更改-每当组件的名称变量更改时切换
- 在编辑器中更改名称状态-在编辑器中更改组件的名称变量时切换

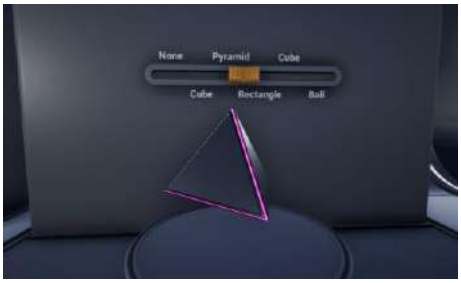


图 22。产卵器使用名称组件

3.2.6Open 组件

使角色能够根据组件的设置通过转换它的网格来打开和关闭。

蓝图:Comp_Open

接口:Interface_StateComponent_Binary

设置:

- 开放地图——用移动或缩放来描述开放动作
  - 文本字段-包含受下面描述的打开动作影响的网格的组件标签
  - 转换映射——描述开放转换的一部分

    *位置-移动网格线性平行于相应的轴

    *旋转-围绕相应的轴旋转网格

    *缩放-沿相应的轴缩放网格

    自定义——在 actor 上实现的自定义函数
  - 打开值-转换值，当参与者打开时
  - 关闭值——当参与者关闭时，转换值
  - 曲线-描述打开或关闭速度的百分比(移动)与时间
  - Duration -定义转换所需的秒数•Open -确定打开组件的起始值。
  - Value——设置为 true 使用转换映射中定义的开放值，设置为 FALSE 使用关闭值

听起来

- 声音打开-当值从 FALSE 变为 true 时播放非循环声音
- Sound Close -当值由 true 变为 FALSE 时，播放非循环声音。
- 打开百分比更改-在打开/关闭过程中，每 tick 传输移动的百分比
- 打开状态更改-当移动到达打开/关闭相对转换时切换

3.2.7%组件用于接收和保存浮点数。

蓝图:Comp_Percent

接口:Interface_StateComponent_Percent

设置:

- Value -确定浮动的起始值:
- Float Changed -当一个新的浮动值被传输到组件时进行切换
- 在编辑器中更改的浮点数——在将新的浮点值传输到编辑器中的组件时进行切换

图 23。角开放。

如果多个网格受到打开操作的影响，则为每个网格在打开映射中添加一个条目，并使用单独的标记引用它

对于更复杂的过程，如沿着多个轴的移动，包括多个转换映射，每个映射描述转换的一部分

这里定义的曲线根据下面定义的持续时间进行压缩或拉伸。

3.2.8Trigger 组件

用于接收来自另一个组件(即拖动组件)的信号，并在自定义函数中处理它。

蓝图:Comp_Trigger

接口:Comp_StateComponent_Trigger

设置:没有

听起来

- 声音触发-当接收到触发信号时，播放非循环声音。

蓝图:Comp_Values

接口:Interface_Integer

设置:

- 值映射-将每个支持的整数匹配到要在选择菜单或类似的自定义菜单中使用的名称和按钮
  - 表项的 Value - integer
  - Name - I18n 结构体，对应于当前条目使用的名称
  - 按钮-按钮结构，允许为匹配条目的选择按钮定义纹理或颜色
- Current Index -确定从值映射中定义的整数开始。
- Value Changed -每当从数组中选择一个新值时进行切换
- 在编辑器中更改的值-在编辑器中从数组中选择新值时进行切换

3.2.10Velocity 组件

接收移动的统计信息

蓝图:Comp_Velocity

接口:Interface_StateComponent_Velocity

设置:

- Velocity -确定速度结构的起始值
  - 距离——表示移动对象的当前距离
  - 百分比-表示沿角度、轴或样条预先确定的移动的当前百分比。
  - 速度-表示移动的当前速度

图 24。使用速度组件。

- Velocity Changed -当 Velocity 结构体的值被改变时切换
- 在编辑器中改变速度-在编辑器中改变速度结构体的值时切换

3.2.11Visual 组件

- 组件触发-当接收到触发信号时切换

3.2.9Values 组件

用于接收和保存整数数组。values 组件专门设计用于与选择菜单组件交互(参见 3.1.1 节)，以创建深度为 1 的菜单。

与以前的版本相反，值
组件现在配备了一个单级整数接口(请参阅第 4 节)，并且只能接收一个整数值

有关 structs 的更多信息

管理参与者的外观，包括网格、材质和 UI 元素给出的关于参与者的信息。视觉组件中的每个条目都创建了一个独特的外观，该外观由一个网格和该网格的许多材料集组成。因此，可视组件现在提供了无与伦比的灵活性。

蓝图:Comp_Visual

接口:Interface_Integer_MultiLevel

设置:

•名称-包含 HUD 标题的键和 I18n 数据表

按钮，用于切换视觉组件

•网格-包含一个元素的每个演员的外观

- 静态网格-状态的静态网格的演员在这个看。
- 骨骼网格-在此外观中显示演员的骨骼网格。
  - 材料-包含一个元素，用于该网格可以出现的每一组材料。

* Name - I18n 结构体，定义材质集的名称

* Materials -数组，包含用于控制外观的网格上的每个材料的条目。

*声音——分配给这个特定材料集的非循环声音

  - 按钮 2D -表示名称，I18n 数据表和颜色或纹理

的按钮来访问这个外观。

—数据资产—指定描述此外观的数据资产。

  - 声音-分配给这个特定外观的非循环声音
- 搜索的组件标签——指的是受视觉组件影响的网格标签。

听起来

- 声音材料变化-一般非循环的声音播放每当一个材料集交换
- 声音网格变化-一般非循环的声音播放每当外观(包括网格)被改变

事件

- 可视状态更改-每当外观或材质集更改时切换

3.3 拍摄组件

抓拍组件组包括抓拍特性所需的所有组件，抓拍特性允许抓拍对象在释放时自动将自身放置在平面上或附着在锚上。“捕捉到表面”(图 25)和“捕捉到锚定”(图 26)都需要一个仔细的提醒

视觉组件一次只能影响一个网格。如果需要改变多个网格或使用不同的材料显示，则必须为每个受影响的网格添加一个视觉组件。

不建议将通用的声音设置与地图中单独分配的声音结合使用

厄宁的外观和材料的视觉组件有一看部分 7.3.

如果没有输入数据表或键为应用程序中会显示缺少密钥本身。

您可以插入静态网格或骨架网格。永远不要在一个元素中同时使用静态网格和在另一个元素中使用骨架网格。

上面指定的网格的所有材质都必须输入以创建一组材质，即使它们没有改变。

如果没有输入数据表或键为应用程序中会显示缺少密钥本身。

设置功能正常。

框架提供了两种类型的连接器组件，使被捕获的参与者能够搜索要捕捉到的合适位置。

锚定组件通过提供用于附加的位置和旋转以及使用标记选择合适的锚定连接器组件与锚定连接器组件协作。

3.3.1 表面连接器组件

确定一个 actor 是否以及如何捕捉到平面表面。表面连接器组件的前向矢量指示参与者的哪一部分应该附加到表面上。

蓝图:Comp_Connector_Surface

设置:

- 表面旋转-使演员自动对齐到表面。
- 即使有物理也应该活跃-允许演员不顾物理而抓拍

图 25。迅速浮出水面

- 在抓拍后允许物理-在抓拍后重新激活物理
- 要求标签-允许组件通过标签识别表面

如果没有指定标记，则任何带有

- 墙壁旋转调整-补偿网格的枢轴从表面连接器期望的偏差
- 搜索距离-定义曲面连接器搜索曲面的范围。

声音:

- 找到声音位置-每当连接器找到合适的表面时，就会播放非循环声音

事件

- 连接连接器-当连接器连接到合适的表面时切换
- 连接器分离-当连接器从合适的表面分离时切换

设置:Snap To Surface

从 snap 到 surface 功能的所有逻辑都封装在 surface 连接器组件中。然而，抓拍是一个复杂的过程，需要精心设置才能正常工作。以下是在设置 snap 到 surface 功能时需要考虑的最重要的概念:

- 定位-表面连接器组件需要定位，使其前向矢量与 actor 的部分正交，actor 应该在敲击后接触表面(矢量组件可以很容易地用于可视化)并进入设置。
- 设置组合-当一个抓取被释放时，抓拍被启动，因此很大程度上受抓取组件的释放类型的影响(见章节 3.1.3)推荐以下设置组合:
  - *释放类型:FreePlacement -设置“Should Be Active Even With Physics”和“Allow Physics After Snap”为 FALSE*

冲击角在 120° 和 60° 之间的搜索轨迹构成平面表面。

- *Release Type: Physics* -设置“Should Be Active Even With Physics”为 true，“Allow Physics After Snap”为 FALSE，将演员放置在墙上或天花板上
- *Release Type: Physics* -设置“即使有物理也应该活跃”和“在 Snap 后允许物理”为 true，以将演员放在地板上

- 表面标记-在地图中的表面上的一致标记系统使表面连接器能够选择合适的表面，但不是表面连接器正常工作所必需的。
- 已附加实例-这只在加载关卡时，如果一个角色应该附加到一个表面上，这是必要的。在这种情况下:
  - *On the Instance* -输入这个 actor 实例所附加的表面 actor
  - *如果 Surface Actor 有各种网格-输入 Actor 应该附加到的网格的标签(无表示根组件)*
  - *当使用套接字时-也指定套接字*

3.3.2 锚点连接器组件

确定一个 actor 是否以及如何连接到另一个 actor 上的锚组件。连接 actor 的位置和旋转完全由锚定连接器组件和锚定组件的正向矢量对齐来定义，如图 27 所示。

蓝图:Comp_Connector_Anchor

设置:

- 搜索距离-定义锚点连接器搜索合适的锚点组件的范围。

搜索半径-调整搜索轨迹的宽度。

•连接器 ID -将连接器与它所用于的锚点匹配

声音:

- 找到声音位置-每当连接器找到合适的锚点就会播放非循环声音

事件

- 连接连接器-当连接器连接到合适的锚点时切换
- 连接器分离-当连接器从合适的锚点分离时切换

Note
In case of anchor vs surface the anchor is always preferred.

搜索半径不要太小否则，玩家将会非常乏味地将角色连接到其他角色上，因为连接器即使非常接近也找不到锚点。

3.3.3Anchor 组件

指定另一个参与者可以附加到参与者的位置以及哪些参与者是合适的。锚组件还指示附加 actor 的位置和旋转，如图 27 所示。锚组件也可以添加到角色兵中，创建身体槽，使玩家能够将演员附加到兵上。

蓝图:Comp_Anchor

设置:

- 球体半径——决定碰撞球体的大小，允许锚点连接器组件的轨迹找到锚点

- 允许的连接器 ID-为每个连接器 ID 包含一个元素(见 3.3.2 节)，应该连接

•连接器应该连接-确保演员在释放后保持连接在一起

允许使用连接器 id，因此可以将各种 actor 附加到同一个锚点。

- 允许物理-在捕捉过程后重新激活物理
- 锚失效-暂时阻止使用此锚
- 声音附加-每当连接器附加到锚点时播放的非循环声音
- 声音分离-每当连接器从锚点分离时播放的非循环声音

如果“连接器应连接”未启用，则当物理被激活时，连接的 actor 在断开后可能会掉到地上。

事件

如果附加了 actor，整个结构就会激活物理，这意味着当附加了 actor 而改变质心时，它就会倾斜。

到锚点。

锚可以有多个条目

- 连接器连接-当连接器连接到锚点时切换
- 连接器分离-当连接器从参与者分离时切换

id To Allow 数组将附加参与者。

- 附加-在 snap 之后附加一个 actor 的逻辑被封装在锚中。然而，为了跟踪附加的参与者，attach 组件是必要的。
- 附加设置-只有在加载关卡时，演员应该附加到一个表面时才有必要。

在这种情况下:

3.3.4Attach 组件

记录哪个 actor 连接到哪个 actor

蓝图:Comp_Attach

设置:无事件:

- 参与者附加-每当参与者附加时切换，将引用传递给附加的参与者
- Actor 分离-每当 Actor 被分离时切换

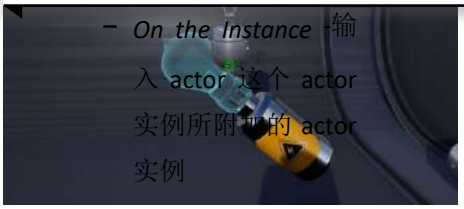
设置:抓拍锚

抓拍锚定功能实现了两个参与者(一个抓拍，一个静止)彼此之间的仔细定位，因此抓拍的参与者可以抓拍(并附加)到静止参与者的特定位置。这需要设置至少两个元素:锚点连接器组件(Anchor Connector Component)，它属于被抓取的参与者，它应该在释放时被捕获;锚点组件(Anchor Component)，它属于静止的参与者，其他参与者应该能够附加到它。

与表面连接器组件一样，锚定连接器组件和锚定组件实现了以下几个功能:

- 定位——锚点和锚点连接器都需要与它们的前向向量垂直于行动者应附着的表面，因为扣点位置是通过对齐这两个向量计算的(使用矢量组件进行可视化)。
- 连接器 ID——用于限制应该将哪个参与者附加到每个连接器分配一个 ID。仅在连接器中使用此 ID 的锚定

– *On the Instance* 输入 actor 这个 actor 实例所附加的 actor 实例



– *如果 Actor 有不同的网格-输入 Actor 应该附加到的网格的标签(无表示根组件)-当使用套接字-也指定套接字*

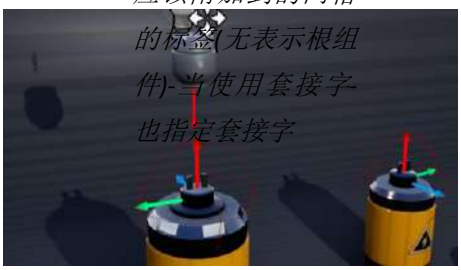


图 26。扣到锚上/连接图 27。锚和连接器。

3.4 多人组件

复制组件

多人应用程序是一个特殊的挑战，AF 核心 cannotActor Replication 为每个问题提供了一个解决方案，这与多人游戏有关。ForComponent

组件复制

现在我们集中精力提供一致的高性能复制系统，为所有组件配备复制功能，并提供一些额外的组件来处理移动复制。

组件

3.4.1Base 复制组件

作为更专门的复制组件的父组件。基本复制组件不用于应用程序中。

蓝图:Comp_Replication

设置:

- Enabled -可用于暂时禁用组件
- Interp Speed -调整插值，以显示复制参与者干净地移动到参与者复制组件转发的新位置
- 公差-确定激活复制过程必须移动参与者的最小距离

•移动期间的滴答间隔-当前移动的间隔

这不是每次都能检查的

Ing 参与者为复制传输其位置

这将超出大多数硬件的能力。

- 没有移动的滴答间隔-当前停止的间隔
  tionary actor 检查运动
- 新建转换集—每当复制组件需要更改转换时切换

3.4.2Actor 复制组件

在多人应用程序中复制参与者的位置。参与者复制组件检测参与者是否移动，并不断向服务器和其他客户端发送新的位置。

蓝图:Comp_Actor_Replication

设置:没有额外的设置相比，父

更多信息见第 6 节

3.4.3 组件复制组件

与参与者复制组件类似，组件复制组件跟踪参与者的可移动组件的相对位置，并将其发送到服务器进行复制。

更多信息见第 6 节

蓝图:Comp_Comp_Replication

设置:

•组件标签-用于识别复制过程中的组件

请注意，这两个复制组件仅用于复制非预定的运动和物理。因此，没有必要向每个移动的参与者或组件添加复制组件。有关详细信息，请参见 6.2 节。

3.5 ui 组件

AF 核心已经为各种各样的用户界面注解提供了基础。他们大多数依靠组件的功能和许多 UI 元素的 AF 核心是内容。当然，每个应用程序的 UI 元素都有很大的不同——非常简单。它们主要是作为基础阳离子或环境。因此，每个 UI 组件都有一个有限的更复杂的 UI 元素，它们应该被添加到可以由开发人员构建的个体或每个组件的描述中。购买与 AF 扩展。

3.5.1 菜单托盘组件指定菜单级别的菜单级别 DA(见章节 7.1.2)

更多信息见 8.2.3 节。

使用菜单和设置托盘加载。

蓝图:Comp_Pallet_Menu 合适的演员:vr -

pawn，董事会演员设置:

•菜单级别-指定应该加载托盘的菜单级别的级别数据资产

3.5.2 基于面板的菜单组件

面板菜单组件

为基于面板的大面板菜单的面板提供内容和基本设置
组件菜单(参见 8.2.2 节)Comp_PanelMenu 作为父类

小面板菜单

只有。这两个子类实现了两种主要的设计，可以为基于面板的菜单选择，大

组件

的垂直面板或小的瓦片状面板。

蓝图:Comp_PanelMenu 设置:

•当前居中索引-允许指定一个已经存在的面板

在创建时扩展

这并不意味着专家组是

- 开始刷出面板-当关卡加载时刷出基于面板的菜单
- 面板功能——可用于在面板上实现小部件功能，以改进复制

现在必须在菜单的中心。

如果你不希望任何面板被展开，将值设置为-1。

- 面板-大面板菜单数据资产的数组，每个面板的菜单应该有一个入口

大面板菜单组件

确定基于大型面板菜单的面板的内容和外观(参见图 28)

蓝 图:Comp_PanelMenu_Large 合 适 的 参 与 者:BP_Demo_LevelMenu_Large 内容:

图 28。大面板菜单，中间:

额外的设置:

- 居中索引-定义创建菜单时哪个托盘位于菜单中心
- 瓷砖之间的距离-定义面板之间的距离

小面板菜单组件

确定基于小面板菜单的面板的内容和外观(参见图 29)

蓝图:Comp_PanelMenu_Small 合适的参与者:BP_Demo_LevelMenu_Small 内容:

- 面板-一组小面板菜单数据资产，每个面板都有一个条目，菜单应该有

设置

图 29。小面板菜单，每行 3 个。

- 每行项-定义每行显示多少个小面板
- 自定义前进位置标签-确定扩展面板的位置和旋转。

面板

- 的行间距离-定义的行之间的距离

基于小面板的菜单不支持滚动。因此，所有的面板都必须适合给定的空间。

- 列之间的距离-定义面板列之间的距离

3.5.3 显示组件小

当激光运动控制器抓取 actor 时，使 actor 生成一个微小的显示，该显示使用名称或短文本和图像描述 actor(参见图 30)。

图 30。微小的显示 UI。

蓝图:Comp_TinyDisplay

适用演员:所有演员设置:

- 名称——包含引用名称的键和 I18n 数据表

  或者对演员的描述
- 图像——包含与角色对应的纹理
- 应该产卵-暂时禁用产卵小显示器

如果没有输入数据表或密钥丢失，则在应用程序中显示密钥本身。

3.5.4Widget 组件

允许参与者显示小部件。控件展开

虚幻引擎固有的小部件与额外的逻辑处理玩家互动，特别是与 VR 兵。

不要将小部件组件与窗口组件混淆。小部件组件只支持显示小部件。窗口组件处理拥有小部件组件(以及其他组件)的参与者的生成。

蓝图:Comp_Widget

设置:

- 递归小部件选择-为高级框架设计的小部件输入实现
- 基于 Widget 选择的交互组件-用于 Widget 输入的 UE 内置实现

AF Core 示例中使用的所有小部件

- 初始标签-定义当一个多次点击的小部件被衍生时，哪个标签是打开的

用户界面

- 小部件功能——可用于在参与者上实现小部件功能，以改进复制
- 小部件类——定义小部件显示的事件:

使用递归小部件选择来实现玩家-小部件交互。

要使用递归小部件选择实现，所有小部件都需要是 Widget_Base 类型。更多信息见 8.1 节。

3.5.5Window 组件

生成一个信息窗口参与者，显示玩家可以与之交互的小部件。

蓝图:Comp_Window

设置:

•窗口-匹配窗口演员，以生成与位置

参考标签手动放置 如需自动放置，请输入 None

- 使用根位置进行定位-使用参与者的根位置而不是窗口组件的自动定位功能的影响位置。

对一个线员一直演来说。

蓝

声音:

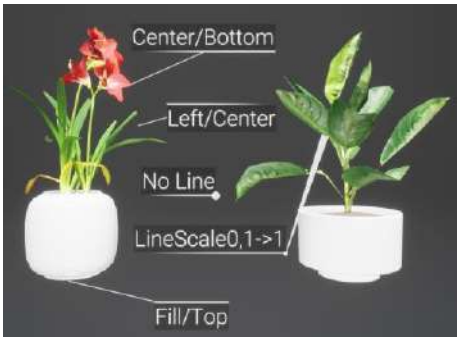
- 声音刷出-信息窗口演员被刷出时播放的非循环声音
- 声音消失-信息窗口参与者被删除时播放的非循环声音。
- 按下窗口按钮——用于在参与者本身上实现小部件功能，以改进复制

3.5.6Mini Tag 组件

生成一个显示简短描述并已连接的迷你标记 actor

图:Comp_MiniTag

设置:



- Line -定义了连接迷你标签与相应角色的线的设计
  - 行-确定是否显示一行

图 31 所示。不同设置的迷你标签。

  - 带有套接字的组件标签-指定带有线连接套接字的网格组件
  - 连接线路的套接字-确定线路连接到的套接字
  - 行位置——如果没有指定套接字，则指定行结束位置的相对位置
  - 保持连接-当 actor 移动时，使行保持连接到套接字(与行位置无关)
  - Line Start Scale -定义角色处的线条宽度
  - Line End Scale -定义了在 mini 标签处的行宽度
- 对齐-确定连接到迷你标签的线

两个对齐选项一起描述线路的明确连接位置。

  - 水平对齐——指定一行水平连接到标签的位置
  - 垂直对齐——指定直线垂直连接到标签的位置

•Text -指定迷你标签的内容

  - 文本大小-决定文本的字体大小
  - 文本——包含键和 I18n 数据表的描述

显示在 mini 标签中。 如果没有输入数据表或键为应用程序中会显示缺少密钥本身。

•边框-设计迷你标签的边框

  - 边界高度-迷你标签边缘线的宽度
  - 绘制大小—显示文本的小部件的大小

3.6 当组件

AF 核心包含许多只能由兵使用的组件。它们定义棋子控件或 UI 元素。然而，由于高级框架支持各种环境，如 VR，桌面或移动，并不是所有的棋子组件可以添加到每个棋子。棋子限制在每个组件的描述中都有说明。

3.6.1Controls 组件

控制组件包含所有运动控制器设置，包括预设，移动和传送。因此，控制组件只能在 VR 兵上发挥作用。

蓝图:Comp_Controls 典当类型:VR

控制器设置:

- 控制器集——定义由棋子衍生的左手和右手运动控制器
- 手跟踪控制器集-定义当手跟踪活动时使用哪些控制器
- 默认预设-指定在无法识别 HMD 时使用哪个预设数据资产作为备用
- 控件预设-将预设控件映射到每个支持的 HMD (见 7.1.1 节)

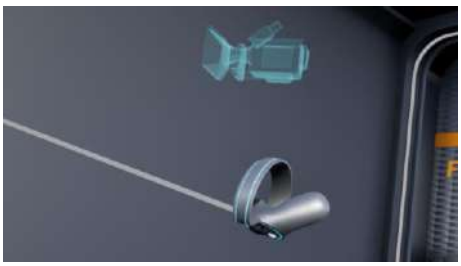
> 一般的提醒
>
> 随着控件组件的引入，控件不再是特定于级别的，而是特定于卒的。因此，如果你想在不同关卡间改变控制，你就需要一组不同的棋子。有关更多信息，请参阅第 5.3 节。

在开始手部跟踪控制预设完全覆盖手部跟踪函数映射下面。

有些手势必须分别定义为左手和右手，因为它们是

- 手跟踪功能地图-匹配手势预设(见节?? ?具有类似于普通控件的键映射的控制器功能
  - 左手手势功能映射-为控制器函数分配左手预设的手势
  - 右手手势功能映射-将右手的预设手势分配给控制器函数
- 手追踪预设-预设数据资产，定义手追踪过程中的传送模式。
- 触觉-确定控制器是否提供触觉响应
- 传送模式-确定哪些位置可用于传送
  - TeleportFree -传送位置可以在平面上自由选择，没有必要的传送区域
  - TeleportComponent——在传送区域或传送点 actor 上搜索传送组件(见 2.3 节)
  - 导航网格-使用虚幻引擎的导航网格标准为合适的传送位置
- 传送搜索类型——指定跟踪搜索合适的传送位置的工作方式
  - 直线跟踪的行为就像一条直线
  - 抛射痕迹沿轨迹向地面移动
- 传送时间-调整传送所需的时间
- 使用传送运动-决定在传送过程中兵的运动如何显示给玩家
  - 没有-没有运动显示
  - 指示-在传送过程中激活缩放效果
  - FullMotion -显示兵从传送起始位置到传送目标位置的完整移动。
- 移动样式——根据控件组件中的按钮映射指定棋子的直接移动设置
  - 流畅——兵移动不断，容易引起晕车
  - 停止运动-为了避免晕车，当兵移动时，玩家的视野会逐渐消失
  - 鬼-玩家移动一个兵的鬼代表到地图的期望区域，完成移动后，兵直接移动到鬼的位置
- 移动方向-描述兵在移动过程中如何被控制
  - 来自相机的方向-兵从相机的前进方向获得前进方向
  - 来自控制者的方向——兵根据控制者所指向的方向移动

图 33。鬼的动作:走路。



  - 相对于控制器位置的方向-移动棋子根据当前控制器位置相对于移动开始时的控制器位置
- 止动间隔-指定一个止动运动的 2 个图像之间的时间(以秒为单位)，仅与止动运动风格相关
- 移动速度-决定兵的移动速度
- 使用渐隐移动-允许移动渐隐到黑色，不建议用于流畅的移动

旋转设置:

- 旋转类型——定义如何实现旋转
  - 流畅-只要旋转功能处于激活状态，就可以流畅地旋转兵
  - Step -每次旋转函数启动时，将棋子按给定角度旋转一步
- 旋转流畅度-定义兵旋转的速度，仅与流畅的旋转类型相关
- 旋转步度-兵卒在每个旋转步中旋转的角度，只与步旋转类型相关。
- 旋转步进渐隐-使相机在步进旋转时渐隐
- 旋转步骤持续时间-以秒为单位确定每个旋转步骤所花费的时间

控制器功能说明

- 选择
- 抓住
- 平
- 暂停
- RadialMenu1
- RadialMenu2
- ToggleFly
- 传送
- 移动
- MoveForward
- MoveLeft
- MoveRight
- MoveBackward
- TurnLeft
- TurnRight
- ResetOrientation

3.6.2 径向菜单组件

在高级框架中径向菜单组件的主要目的是控制托盘 UI 元素(参见 8.2.3 节)。但是，通过设计适当的按钮，放射状菜单组件生成的按钮集也可以用于其他方法。

蓝图:Comp_RadialMenu 典当类型:VR

设置:

- 径向菜单 1 -包含一个按钮蓝图的数组，应该在打开径向菜单时产生。
- 径向菜单 2 -包含一个按钮蓝图的数组，应该
  be spawned upon opening the radial menu.

The radial menu 2 slot serves for creating different radial menus for the left hand and right hand controller for example.

Currently Implemented Button Types

- Pallet Button - spawns a pallet UI element that corresponds the button
- Spawn Button - spawns an actor like a weapon or a device.

Figure 35. Radial Menu.

3.6.3 Touch Component

registers touch input and transmits it to the mobile pawn for processing.

Blueprint: `Comp_Touch`

Pawn Type: Mobile Pawn

设置:

- 最大点击时间-允许触摸注册为点击的最大时间(以秒为单位)
- 最大双击时间-如果要注册为双击，可以在点击之间的最大时间(以秒为单位)
- 滑动灵敏度阈值-手指必须移动的最小像素量才能被识别。
- 保持时间-以秒为单位的时间，直到保持注册

3.6.4HUD 组件

定义屏幕应用程序中 HUD 的内容。有关框架中 HUD 管理的更多信息，请参见 8.2.6 节。

蓝图:Comp_UI_HUD

兵类型:移动兵，桌面兵，非对称兵

设置:

- HUD -为每个 HUD 元素包含一个条目
  - 名称-包含键和 I18n 数据表，定义 HUD 元素的标签
  - 小部件-确定哪个小部件用于 HUD 元素
  - 帧-确定小部件在 HUD 的可定制插槽中显示。要查看帧列表，请查看页边距及其在屏幕上的位置，参见图 65

3.7 信息组件

与静态的数据资产相比，信息组件用于保存和提供可在运行时更改的动态信息

3.7.1 玩家信息组件

由玩家状态用于保存除玩家索引之外的关于某个玩家的个人信息。

蓝图:Bp_PlayerInfo_Basic 玩家统计:

- 播放器颜色-分配给播放器的颜色
- 玩家名-应该保存多人游戏场景中玩家输入的名字

3.8 其他组件

杂项组件是各种功能组件的集合，由于其独特的功能，它们不能被归类到其他组中。

3.8.1Delete 组件

允许删除参与者。

蓝图:Comp_Delete

接口:Interface_Deletable

设置:

- 按接口处理——使用可删除接口重写参与者上定制函数的组件逻辑
- 过渡持续时间-调整删除过程所需的时间
- 曲线-描述删除过程中参与者的尺度变化

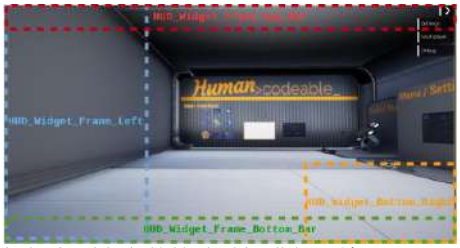
这里定义的曲线是拉伸的和

事件:

- Before Delete -允许在删除之前执行准备函数(即动画)
- 出现完成-当删除组件贡献给刷出的动画完成时切换

3.8.2Orbit 组件

轨道组件允许卒围绕一个参与者移动，就像在



根据上面定义的持续时间进行压缩。

一个轨道。但是，它不会自动移动棋子。它只是暂时改变了兵的移动和导航。

蓝图:Comp_Orbit

设置:

- 俯仰范围-限制兵位置的俯仰角度
- 缩放范围-定义兵卒位置围绕轨道组件的最大和最小半径

图 37。帧的位置和大小。

兵可以围绕轨道组件的 z 轴旋转 360°。

3.8.3Spawn Actor 组件

以多人兼容的方式生成传输的参与者。

蓝图:Comp_SpawnActor

设置:无

图 38。代码示例刷出器。

- 刷出声音-非循环的声音，在演员被刷出时播放。
- Actor 已生成-当 Actor 已生成时切换

3.8.4 刷出位置组件

在产卵器中生成传输的 actor 并管理产卵器功能。

蓝图:Comp_SpawnLocation

设置:

- 大小-缩放对象，所以它适合产卵器
- 使用刷出粒子-在刷出时显示粒子效果
- 使用旋转-在生成后旋转生成的 actor
- 自动重生-每个分离的 actor 立即被另一个相同类型的 actor 替换。
- 声音刷出-每当演员被刷出时播放非循环声音
- 新演员诞生-每当有新演员诞生时切换
- Actor 分离-当生成的 Actor 从产卵器中取出时切换

请注意

两个刷出组件独立工作，每个都能够实现和复制刷出过程。

- 刷出演员组件直接刷出一个演员
- 刷出位置组件处理刷出器的完整逻辑，在刷出时产生额外的效果，并且在 actor 被移除时重生

设置产卵器

要创建产卵器，将组件添加到蓝图中，并在事件图中调用组件的产卵 Actor 事件。(参见图 38)。

3.8.5Spectator 组件

观众组件允许演员在计算机屏幕上显示其摄像机所看到的内容。拥有的参与者可能是一个卒，例如，不对称卒(参见 5.3.5 节)，但也可能是一个安全摄像头或应用程序中的任何其他参与者。

蓝图:Comp_Spectator

设置:

- 观众屏幕尺寸-观众组件提供的实时馈送的 ptx 分辨率
- Key -为关卡中的每个旁观者组件提供标识符

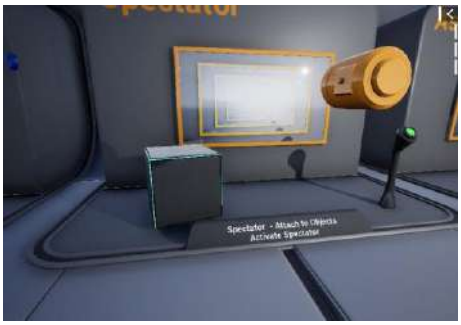


图 39。观众组件示例。

3.8.6Vehicle 组件

车辆组件被添加到车辆类型角色中，以控制棋子进入和退出车辆。此外，只要兵卒被放置在车辆内，车辆组件就会处理兵卒的移动限制。

蓝图:Comp_Vehicle 转换:

- 转移持续时间-以秒为单位决定兵从当前位置移动到车内位置所花费的时间
- 渐隐-允许在过渡过程中使用相机渐隐
  - 不褪色-禁用相机褪色
  - 渐隐-使用相机渐隐进入车辆
- 运动-定义棋子如何从当前位置移动到车辆内位置呈现给玩家
  - 无-显示卒没有移动
  - 指示-显示切割相机前的运动开始

  -充分运动-显示兵的充分运动

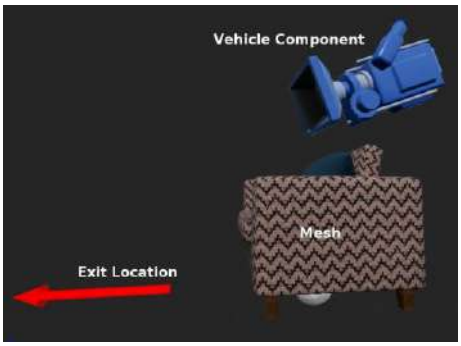


图 40。车辆组件。

退出:

•退出组件标签-指定演员的哪个场景组件作为退出车辆的目标位置(见空白处)

- 声音进入-当兵被运送到车辆部件位置时播放非循环声音
- 声音出口-当兵从车辆组件位置被运送到出口位置时播放非循环声音

信息框-车辆压缩机设置

车辆组件需要处理兵在两个位置之间的转换:

- 在车辆位置-是车辆组件相对于其拥有的参与者的网格的位置，并构成了兵的有利位置，当放置在车辆中
- 出口位置-是相对于演员网格的附加场景组件的位置。退出位置指定退出参与者时卒放置的位置。出口位置由车辆组件的单独标签指示。

  - 淡出-使用相机淡出，只退出车辆
  - 淡入淡出-使用相机淡入淡出进入和退出车辆

不能与相机相结合淡入。

车辆组件在位置之间运输卒。它不会传送兵，这意味着在兵的路上移动的角色可以被打倒或类似的情况。

3.8.7Video 组件

视频组件极大地方便了在屏幕上或应用程序中类似的屏幕上显示视频文件或流媒体视频的设置。在其默认状态下，视频组件被设置为平面静态网格。然而，静态网格可以随意改变。

蓝图:Comp_Video 来源:

- 视频源——定义组件应该使用哪种类型的视频源
  - 流媒体源-流视频从下面注册的 URL
  - 文件媒体源-在项目中使用视频文件作为源
- 源 URL -声明视频应该流媒体的 URL。源 URL 与“文件”无关

Required:媒体源”设置。

- 媒体声音组件-管理播放视频的声音
  - 要搜索的组件标签-声音组件的标签
    (虚幻固有)其中应该播放视频的声音
  - 同时触发自我-通知视频组件声音组件在同一个 actor 上
- 媒体纹理-渲染视频文件
- Material Face -在视频组件的静态网格的材料集中显示视频的材料的索引

信息框-视频设置

视频组件管理在虚幻应用程序中显示视频所需的大部分繁琐步骤。将它添加到 actor 后，只需要额外的 3 个部分:

- 网格和材质——视频组件的默认网格是一个 16:9 的平面，在激活组件时由媒体纹理取代
- 媒体纹理-通过复制现有的媒体纹理为演员创建一个单独的媒体纹理。
- 声音-添加虚幻固有媒体声音组件到相同的演员，并提供一个标签供参考

每个视频都需要自己的媒体纹理。

视频组件可以以与活动组件或类似状态组件相同的方式进行切换。

4 个接口

接口为组件或框架的其他功能之间的通信提供了标准化。每个接口都定义了可以由该接口接收和传输的单独数据类型。

4.1 状态组件接口

所有状态组件都被调整为与接口一起工作，并有一个指定的接口。所有交互组件都使用状态组件的接口与状态组件交互。状态组件接口分为两类:

- 非分层接口——只接受对应数据类型的单个值
- 多级接口——接受其对应数据类型的数组，并支持构造和传输树状数组结构

4.1.1 带有二进制接口的组件传输布尔值。

类别:Unleveled

蓝图:Interface_StateComponent_Binary

接受数据:1 布尔值

二进制接口组件

•主动组件

- 打开组件
- 突出显示组件
- 窗口组件

4.1.2Integer 接口

传输一个整数数组

类别:多层次的

蓝图:Interface_StateComponent_Integer_MultiLevel

接受数据:1 个整数数组

整数接口组件

•视觉组件

4.1.3Integer 接口

传输整数

类别:Unleveled

蓝图:Interface_StateComponent_Integer

接受数据:1 个整数

整数接口组件

•价值观组件

4.1.4Percent 接口

传输一个浮点值。

类别:Unleveled

蓝图:Interface_StateComponent_Percent

接受数据:1 个浮点数

百分比界面组件

•百分比成分

4.1.5Color 接口

传输由 3 个浮点值组成的颜色结构。

类别:Unleveled

蓝图:Interface_StateComponent_Color

接受数据:1 个线性颜色结构

彩色界面组件

•颜色组件

4.1.6Name 接口

传输名称值。

类别:Unleveled

蓝图:Interface_StateComponent_Name

接受数据:1 个名称变量

名称接口组件

•名称组件

4.1.7Trigger 接口

发送触发信号

类别:Unleveled

蓝图:Interface_StateComponent_Trigger

接收数据:无

带有触发接口的组件

- 触发组件
- 拖动组件

4.1.8Velocity 接口

对象的 3 个浮点值组成的速度结构
百分比，速度和距离。

类别:Unleveled

蓝图:Interface_StateComponent_Velocity 接受数据:1 个速度结构

具有速度接口的组件

•速度分量

4.2 交互接口

交互接口支持交互组件和参与者之间的数据传输。添加到 actor 后，它们主要用于将交互组件的设置传输给 actor。

4.2.1GazeView 接口

组件:GazeView 组件

蓝图:Interface_InteractComponent_Gazeview

4.2.2Latchable 接口

组件:锁扣组件

蓝图:Interface_InteractComponent_Latch

4.2.3Overlap 接口

分量:重叠分量

蓝图:Interface_InteractComponent_Overlap

4.2.4Grabable 接口

组件:抓取组件

蓝图:Interface_InteractComponent_Grabbable

4.2.5Selectable 接口

组件:选择组件

蓝图:Interface_InteractComponent_Selectable

4.3 其他接口

4.3.1Deletable 接口

组件:删除组件

蓝图:Interface_Deletable

4.3.2Highlightable 接口

组件:高亮组件

蓝图:Interface_Highlightable

4.3.3 抓取 Actor 界面

允许一个参与者抓取另一个参与者。每个能够抓取另一个行为的参与者都需要一个抓取参与者界面，包括兵或运动控制器。

蓝图:Interface_GrabbingActor

4.3.4I18n 接口

I18n 接口允许显示文本的参与者实现更新语言功能。在更改语言设置后，每个具有 I18n 接口的参与者(主要是小部件)都会根据其 I18n 数据表自动将文本更改为相应的翻译。有关更多信息，请参见 7.2 节。

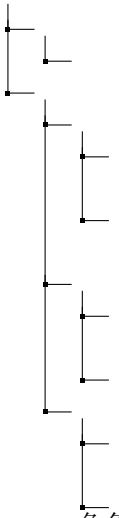
蓝图:Interface_I18n

5 环境

5.1 典当层次结构

基地典当

框架依赖于相当复杂的 pawn 类层次结构(参见 schemeVR pawn)



角色 VR Pawn 在空白处)，

它为不同的 en-提供了定制的 Pawn

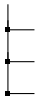
屏幕棋子的环境框架支持包括桌面，VR 和移动。最基本的卒是 BP_Pawn_Base，它作为所有卒的父类 Assymetric Camera。VR 兵是最奇异的兵类，不对称的兵，因为 VR 环境需要与其他兵有很大不同的特征

环境。

桌面典当

桌面，移动和不对称的游戏环境-角色桌面的许多功能
然而典当品却非常相似，因此被浓缩成
摄像头桌面 Pawn 屏幕 Pawn (BP_Pawn_Screen)，作为移动 Pawn 桌面，移动和非对称 Pawn 的父类。进一步的子类扩展移动角色
典当的特点:

移动棋子



- 相机兵-没有虚拟身体
- 角色兵-装备了一个虚拟的身体

基地典当

基地典当

基础兵是框架兵和屏幕兵中最基本的兵类
VR Pawn 巩固了层级中所有棋子的共同特征。基地……
Pawn 不能在任何应用程序中使用。

蓝图:BP_Pawn_Base 特点:

- 相机-放置视角到球员的位置
- Gazeview -为与 Gazeview 组件的交互建立摄像机跟踪(见 3.1.2 节)
- 瞬间移动-允许兵改变位置和旋转，并建立一些变量来调整瞬间移动的风格

屏幕典当

基地典当

屏幕棋子巩固了移动桌面屏幕棋子的相互特性
桌面兵和非对称兵不依赖运动建立控制…

控制器。

蓝图:BP_Pawn_Screen 特点:

- 移动-使卒导航的经验，而不依赖于控制组件
- 交互-建立选择，抓取和闭锁没有运动控制器
- 导航模式-为鼠标或触摸屏提供不同的导航风格

-十字准星-使用鼠标导航相机在 FPS-

游戏

- 自由鼠标-显示并允许由鼠标或触摸屏控制的光标自由移动
- 触摸-专门适应移动导航体验

设备

屏幕兵和基础兵一样，并不是为了在体验中使用，而只是作为桌面兵、移动兵和非对称兵的共同特征的总结。

准星导航模式仅为可用于桌面兵或非对称兵

触摸导航模式只能用于移动兵

5.2 启动

在启动应用程序时，玩家控制器会检查体验环境，并使用以下标准为该环境选择信息级别文件中指定的棋子(参见图 41):

- VR -当一个 HMD 出现在体验开始时，VR 兵会自动选择，并在 VR 中生成运动控制器来移动和互动(见 5.3 节)
- 桌面-桌面棋子作为默认环境和亲

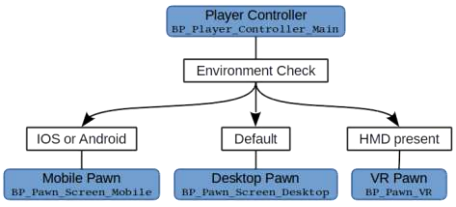


图 41。启动和棋子选择。

使用鼠标和键盘移动并与体验交互的视频功能(见章节 5.4)

- 移动——当操作系统是 IOS 或 Android 时，移动棋子会被自动选择，并提供通过触摸屏移动和交互的功能(参见 5.5 节)

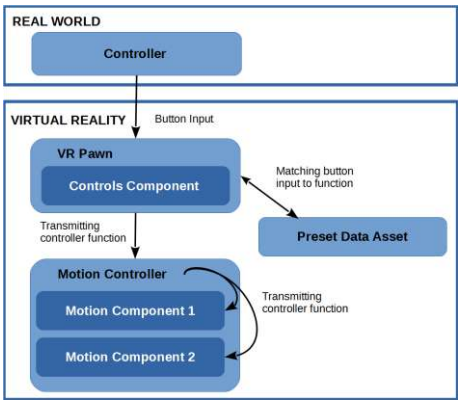
5.3 虚拟现实

VR 环境向开发人员提出了在其他环境中很少遇到的非常具体的挑战。这首先包括应用程序中真实的活动控制器的表示，但也包括移动，没有 HUD 的 UI 元素的实现和模拟疾病。

高级框架使用运动控制器来表示现实控制器和运动组件来实现它们的大部分功能。这需要一种方法将现实控制器的按钮输入与运动组件的功能连接起来。此信息流如图 42 所示。现实控制器上的每个按钮输入都通过 pawn 传输到它的控件组件(参见 3.6.1 节)，这是设计 VR pawn 控件的关键元素。控件组件引用许多预设数据资产，这些资产将现实控制器的按钮输入连接到控制器函数，控件组件将控制器函数传输回 VR 卒。随后，VR pawn 将控制器功能传递给运动控制器，而运动控制器又将其传递给所有运动组件。最后，运动

Reminder

All pawn classes described in the environment sections are supposed to be used as parent classes for the pawns of an application.



与控制器函数匹配的组件实现了所请求的功能。

图 42。信息流。

5.3.1VR 典当

基地典当

根据虚拟现实应用的需要，调整虚拟现实棋子。在 con-VR Pawn 与其他 Pawn 相比，VR Pawn 的大部分功能都外包给了角色 VR Pawn

.. .运动控制器和运动组件(参见第 5.3.2 和 5.3.3 节)。卒主要作为玩家的有利位置，并将现实控制器的输入信号传输给运动控制器。

蓝图:BP_Pawn_VR

- 运动控制器- VR 兵在体验开始时产生一对运动控制器，实现运动和与体验的交互(参见 5.3.2 节)
- 输入转发- VR 兵将玩家的所有关键输入转发到运动控制器

5.3.2Motion 控制器

运动控制器被设计来代表物理控制器

虚拟现实体验。每个控制器由运动控制器和运动控制器组成

在所有其他环境中，控制

运动组件的数量。从 3.1 版本开始，运动控制器的层次结构变得更加复杂。特别是手部运动控制器现在更加复杂，为每只手设计不同的控制器成为可能。此外，在继承手部运动控制器的两个附加类中实现了手部跟踪。

作为所有运动控制器的父类，不应该在应用程序中用作运动控制器。它主要管理控制器功能或关键输入的运动组件和传输。

基础运动控制器

蓝图:BP_MotionController

包含的运动组件:无

> 运动组件插座
>
> 由于各种 HMD 的控制器构造非常不同，每个运动控制器都配备了许多插座，以确保运动组件，如激光，传送或径向菜单连接在相对于控制器网格的正确位置。

运动控制器

为基础运动控制器配备与 HMD 匹配的控制器网格，并实现显示按钮映射的迷你标签。

蓝图:BP_MotionController_Controller

包含的运动组件:无

激光运动控制器

为运动控制器配备激光(实现为运动组件)作为与其他行动者交互的主要手段(见图 47)。

蓝图:BP_MotionController_Controller_Laser 包含的运动组件:

- 激光指针运动组件-见第 5.3.3 页
- 径向菜单运动组件-见第 5.3.3 页
- 运动运动组件-见第 5.3.3 页
- 传送运动组件-见第 5.3.3 页

暂停运动控制器

暂停兵生成自己的运动控制器，具有最小的功能。

蓝图:BP_MotionController_Controller_Pause

运动组件包括:无

由兵直接实现，运动控制器已经过时了。

基础运动控制器
运动控制器
控制器
激光运动控制器
暂停运动控制器
手部运动控制器
左手运动
控制器
右手运动
控制器
Handtracking 运动
控制器
离开 Handtracking
运动控制器
右手跟踪运动控制器

图 43。控制器迷你标签。

这意味着在暂停期间，不管关卡中控制器的外观如何，控制器将始终匹配 HMD。

手部运动控制器使运动控制器有可能出现手部的骨骼网格。然而，手部运动控制器主要作为所有手部运动控制器的父类。

蓝图:MC_Hands

包含的运动组件:无

左手和右手运动控制器

向运动控制器提供左手或右手的骨架网格和相应的动画蓝图。

蓝图:

- MC_Hands_Left
- MC_Hands_Right

包含的运动组件:

- 抓取运动组件-参见第 50 页
- 激光运动组件-见第 49 页
- Ping 运动组件-参见第 53 页
- 径向菜单运动组件-见第 51 页
- 运动运动组件-见第 51 页
- 传送运动组件-见第 51 页

手跟踪运动控制器

在开始手跟踪时，手跟踪运动控制器代替原来的运动控制器。手跟踪运动控制器可以识别由手势数据资产定义的手势(见第??节)，并将其与控件组件中定义的控制器函数进行匹配(见第 3.6.1 节)，类似于普通运动控制器的按钮映射。

蓝图:BP_MotionController_Hands_Tracking

包含的运动组件:无

左右跟踪运动控制器

为手跟踪控制器提供与人手匹配的左手或右手骨骼网格，这样他们就可以在运行时动态地跟踪玩家的手和手指位置。

蓝图:

- BP_Controller_Hands_Tracking_Left
- BP_Controller_Hands_Tracking_Right 包含的运动组件:
- 抓取运动组件-参见第 50 页
- Ping 运动组件-参见第 53 页
- 运动运动组件-见第 51 页
- 传送运动组件-见第 51 页

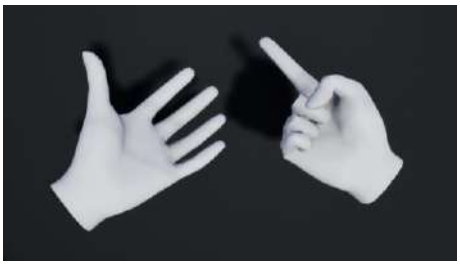


图 44。手部运动控制器

沿着 z 轴翻转右手网格，创建左手网格。

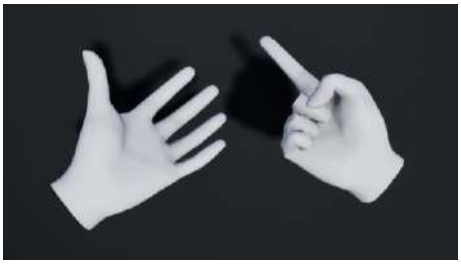


Figure 45. Meshes for Hand-tracking

HowToSetUpHand-Tracking

Hand-tracking works completely different from the normal VR controls. The controls component on the pawn has its own section controlling it. Here are the basic steps to set everything up:

- Motion Controllers - Hand-tracking has is own motion controllers. Enter them under Hand-tracking motion controllers on the controls component
- Gestures - For the Hand-tracking motion controller to recognize a gesture must be defined in a gesture data asset.
- Function Mapping - the controls component contains a map which assigns a controller function to each gesture DA for each hand
- Preset DA - Hand-tracking still needs a preset to define teleport controls

5.3.3Motion 组件

运动组件将组件系统应用于运动控制器。因此，自定义运动控制器可以以类似于新 actor 类的方式设计。运动组件经常与其他组件交互，以执行复杂的功能，如抓取或选择，就像 actor 组件之间相互合作一样，请参见图 46 中的示意图示例。AF 核心包含整体 8 个定制的运动组件。

- 激光运动组件-将激光指示器连接到控制器上
- 手指运动组件-允许用食指选择
- 抓取运动组件-管理抓取与手的运动控制器

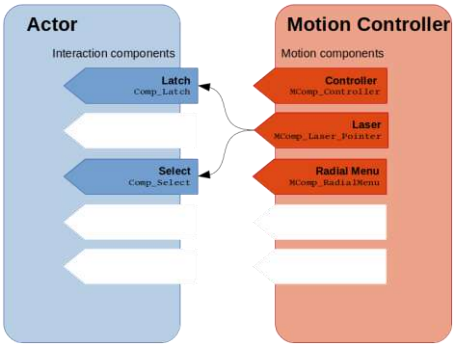


图 46。运动组件系统。

图 47。激光运动控制器。

- 手激光运动组件-更简单的激光运动组件调整为手运动控制器
- 径向菜单运动组件-生成一组径向菜单按钮
- Ping 运动组件-产生一个视觉效果来指示演员，方向等。
- 移动运动组件-允许玩家通过移动棋子来导航应用程序
- 传送运动组件-允许玩家通过传送棋子来导航应用程序

激光运动组件

激光运动组件将激光指示器连接到控制器上，发出激光轨迹，并为玩家提供远程交互的可能性。

蓝图:BP_MComp_Controller_Laser

插座首选:MComp 激光器

适合运动控制器:控制器运动控制器控制器功能:

- 选择——在参与者上切换选择交互组件(参见 3.1.1 节)
- 抓取-切换抓取组件(见 3.1.3 节)，并允许运动控制器将 actor 附加到自身
- 激光模式-切换激光模式(见下面的信息框)
- Ping -在激光撞击位置产生视觉效果
- Max。激光长度-激光轨迹的最大长度
- Max。插值距离-定义原始轨迹冲击位置和新的轨迹冲击位置之间的距离(当控制器移动时)，激光运动不再插值，而只是设置新的位置
- 指针转换交互-设置默认的激光模式
- 插补速度-决定激光轨迹跟随控制器移动的速度

控制器的功能

大多数运动组件实现一个或多个控制器功能。在为控制器选择运动组件时，必须确保每个控制器功能仅在一个运动组件上实现，以避免不兼容。

- 最大抓取距离偏移-你可以移动一个被抓取的角色远离兵的最大距离
- 首选连接插座-定义运动控制器网格上的插座，激光应该连接在那里

激光模式

由激光运动控制器抓取的 actor 不随控制器旋转。相反，玩家可以根据预设的数据资产访问角色的转换。框架提供 3 种模式:

- 距离+旋转:操纵杆的 y 轴使角色离卒更近或更远，操纵杆的 x 轴使角色绕 z 轴旋转。
- 两轴旋转:操纵杆的 x 轴围绕 z 轴旋转角色。拇指杆的 y 轴使演员绕 y 轴旋转。
- 距离+比例:操纵杆的 y 轴使角色离兵越来越近或越来越远，操纵杆的 x 轴使角色越来越近或越来越远。

抓取运动组件使运动控制器能够抓取或锁定演员。

蓝图:BP_MComp_Hand_Grab

Socket 优先级:无

适合的运动控制器:手部运动控制器

- 抓取-切换抓取组件(见 3.1.3 节)和锁存组件(见 3.1.5 节设置:
- 搜索距离-到 GrabSearch 套接字位置的手运动控制器上的距离，运动组件在其中搜索演员
- 搜索半径-确定搜索轨迹的半径
- 拉速-定义物体在抓取时向手部运动控制器移动的速度
- 首选连接插座-定义运动控制器网格上的插座，激光应该连接在那里

额外的设置抓取和闩锁

当 VR 手试图抓住或抓住一个 actor 时，手会在 actor 的网格中搜索插座，并将最合适的插座与运动控制器联系起来。通常，这将是最接近的兼容套接字。这种实现允许参与者捕捉到首选位置，而不必使用奇怪的枢轴位置制作特定的网格。为了让 VR 手在抓演员时看起来自然，每个套接字名称的前 4 个字符必须遵循以下约定:

XNN

其中 X 是字母 L 或 R，将套接字分配给左 VR

图 48。图 49.用 L_00 抓花瓶手抓位置。

手(L)或右 VR 手(R)， NN 在 00 到 13 之间分配 VR 手的姿势，如图 49 所示。

径向菜单运动组件

径向菜单运动组件生成一组圆形按钮，可以由同一只手的拇指棒或另一只手的激光进行选择。更多信息见 8.3.2 节。

蓝图:BP_MComp_RadialMenu

套接字首选项:MComp RadialMenu

适合的运动控制器:所有控制器功能:

- RadialMenu1 -根据径向菜单生成一个径向菜单

  1 棋子上的径向菜单组件的设置

- RadialMenu2 -根据径向菜单生成一个径向菜单

  2 棋子上的径向菜单组件的设置

设置:

- 首选连接插座-定义运动控制器网格上的插座，激光应该连接在那里

传送运动组件

传送运动组件允许并控制传送

蓝图:BP_MComp_Teleport

套接字优先级:MComp 传送

适合的运动控制器:所有控制器功能:

- 传送-根据控件组件上的设置执行棋子的传送(参见 3.6.1 节)
- 最大痕迹长度-确定传送搜索痕迹距离兵的最大距离
- Max。插值距离-定义原始轨迹冲击位置和新的轨迹冲击位置之间的距离(当控制器移动时)，激光运动不再插值，而只是设置新的位置
- 插补速度-决定激光轨迹跟随控制器移动的速度

- 首选连接插座-定义运动控制器网格上的插座，激光应该连接在那里

蓝图:BP_MComp_Movement

Socket 优先级:无

运动成分

移动动作组件启用并控制除传送外的所有移动选项

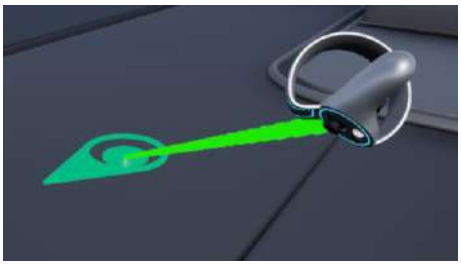


图 50。用手旋转自由传送。

适合的运动控制器:所有控制器功能:

- 行走-沿着 x 轴和 y 轴移动兵
- 飞-沿着 3 个轴自由移动兵
- TurnLeftStep -以给定角度的步骤将兵向左旋转
- TurnRightStep -以给定角度的步骤向右转动兵卒
- TurnLeftFluent -使兵向左移动
- 使兵向右移动
- MoveFluentForward -流畅地向前移动棋子
- MoveFluentBackward -流畅地向后移动棋子
- MoveFluentRight -流畅地向右移动棋子
- MoveFluentLeft -移动兵向左流畅

设置:

- 首选连接插座-定义运动控制器网格上的插座，激光应该连接在那里

Ping 运动组件

使运动控制器产生视觉效果，以指示演员，目标或方向。

Blueprint: BP_MComp_Ping Socket 优

先级:无

合适的运动控制器:全部

与激光指针运动不兼容组件。

控制器功能:

- Ping -产卵的视觉效果

设置:

- 首选连接插座-定义运动控制器网格上的插座，激光应该连接在那里

手激光运动组件

手部激光运动组件是激光的降级版本

指针调整它的使用在手上的运动控制器。

激光笔实现了一些

蓝图:BP_MComp_Hand_Laser

Socket 优先级:无

适合的运动控制器:手部运动控制器

- Select -与 actor 上的 Select 组件交互(参见 3.1.1 节)
- 最大激光距离-决定激光距离兵的最大距离
- 激光开始距离-从手运动控制器开始激光搜索冲击的距离
- 首选连接插座-定义运动控制器网格上的插座，激光应该连接在那里

手指运动组件

位于食指尖端的手指运动组件允许手部运动控制器检测伸出食指的向前运动并与行动者上的选择组件交互。

蓝图:BP_MComp_Hand_Finger

Socket 优先级:无

适合的运动控制器:手运动控制器

控制器功能:无

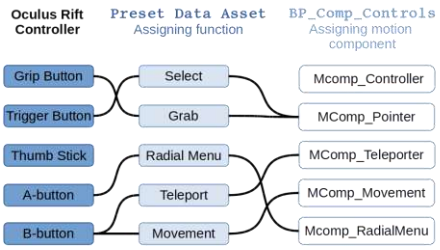
- 首选连接插座-定义运动控制器网格上的插座，激光应该连接在那里

5.3.4 预设

预设数据资产提供连接按键输入到运动控制器功能的按钮映射。该框架已经为不同 HMD 的控制器提供了许多不同的预设。

更多信息见 7.1.1 节。

内容:参见 7.1.1。



5.3.5 图 51 按钮映射示例

非对称游戏玩法将 VR 和桌面环境混合在一起，允许两名玩家在只有一个 HMD 可用的情况下体验 VR 应用程序。AF 核心提供了两种方法来实现不对称的游戏玩法:

- 观众模式-允许实现有利位置，从那里第二个玩家可以监视玩家使用 HMD 导航应用程序在桌面上。
- 不对称兵-生成一个完整的屏幕兵，可以通过鼠标和键盘独立控制移动和与关卡互动。

观察者模式

观众模式是基于观众组件，可以添加到任何演员提供额外的有利位置。观众组件的摄像机馈送既可以直接在桌面上访问，也可以传输到应用程序内屏幕。

图 52。观众模式。

- 在 actor -添加观察者组件，并根据之前的设置自动停用。到第 3.8.5 节。
- 激活 Camera Feed——旁观者组件依赖于一个事件或关键输入来激活。

控制器功能，可能会干扰为手部运动控制器设计的其他控制器功能。

在给定的时间内，每个关卡只能激活一个旁观者组件。在 ac - 在设置中，另一个观众组件旁观者组件本身包含一个摄像机，因此没

- 显示摄像机馈送-当组件被激活时，观众组件的摄像机馈送会自动显示在现实生活中的监视器上。为了在 VR 体验中显示相机馈电，创建一个 2D 渲染目标来创建一个动态材料。

不对称 Pawn base 兵

作为一个额外的棋子，由键盘和鼠标控制屏幕棋子，可以导航的经验。

不对称的兵

蓝图:Pawn_Screen_AsymmetricCharacter Asymmetric

操作方法:参见桌面 pawnPawn

设置简介

- 在 VR 兵上- VR 兵提供了“衍生非对称兵”功能，只需要调用并提供位置和兵类
- 在非对称棋子上——非对称棋子是一种屏幕类型的棋子，可以提供与屏幕环境兼容的所有组件和功能
- 不对称兵相机馈送-不对称兵有利位置自动显示在现实生活监视器。它可以通过使用上面简要描述的渲染目标在 VR 应用程序中显示。

5.4 桌面

桌面环境是框架支持的所有环境中最广泛和最多变的，包括(理论上)许多输入设备，如鼠标、键盘或游戏板。

桌面典当

基地典当

桌面典当作为所有屏幕应用程序的默认典当。屏幕典当

Desktop Pawn(或它的子

控制

- 移动-键盘的 WASD 键根据导航模式的规格移动棋子
- 移动模式-桌面兵支持移动模式

游戏控制器仍在开发中。

类)现在提供了所有的功能

个性桌面

使用鼠标和键盘导航和交互体验。兵

蓝图:BP_Pawn_Screen_DesktopCamera 桌面 Pawn

冲刺，蹲下，跳跃和飞行分配以下键

- 太空跳跃
- Shift -冲刺(保持冲刺，按住键)
- C -蹲(按下键进入/退出蹲模式)
- Y -飞行(按下键进入/退出飞行模式)

•改变导航模式- alt 键改变导航模式之间的准星和自由鼠标

- 选择——兵通过点击鼠标左键选择一个角色。
- 抓住/锁住-兵通过按住鼠标左键或点击鼠标右键抓住一个角色
- 旋转抓取的对象-按住鼠标右键移动鼠标，旋转抓取的对象。

图 53。角色桌面卒。

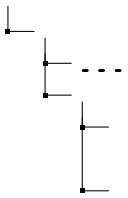
5.5 移动设备

移动和桌面环境之间最显著的区别是移动设备的容量有限，它们依赖触摸屏作为主要输入源。此外，小屏幕只能支持最少的 HUD 元素。

移动典当基地典当

移动典当是一个屏幕类型的典当，带有调整为输入屏幕典当的控件通过触摸屏。

移动棋子



移动棋子

蓝图:BP_Pawn_Screen_MobileCharacter Mobile

控制:典当

- 移动-在地板上双击将兵传送到选定的位置
- 相机移动-滑动移动兵的相机
- 在可选择的参与者上的选择-标签
- 抓取/锁扣-扣住可抓取/锁扣的动作

6 多人

创建一个多人应用程序是一项复杂的任务，包括许多超出高级框架范围的问题。因此，本章绝不是对这一主题的全面讨论。它主要介绍了我们在组件和 AF 核心的其他功能中所做的修改，以允许您设置一个多人应用程序。这首先包括复制和所有权，以及设置多人游戏的基本大厅。

6.1 Client Server Models and Ownership

The Framework supports at the moment 2 client server models:

- Dedicated Server - does not have an assigned player controller and only hosts the game (see Figure 54, top)
- Listening Server - has an assigned player controller and hosts while playing (see Figure 54, bottom)

Ownership basically constitutes which client if not the server can execute replicable events including changing states, location and more on an actor. Therefore specific interactions especially grab and latch require a change of ownership to be executed.

Generally it is beneficial if most actors are owned by the server and changes of ownership are kept to the absolute minimum.

The client generally owns

- the pawn
- actors permanently attached to the pawn like the motion controllers
- grabbed and latched actors
- all actors spawned by the client

The server generally owns

- all actors in the level (exceptions: see above)
- all actors spawned by the server

ChangingOwnership

Actor ownership changes mainly when a player grabs or latches onto an actor. In the AF Core a grabbed/latched actor is always owned by the client of the player who executed the grab or latch. Upon release the ownership is not restored to the previous owner to limit ownership changes.

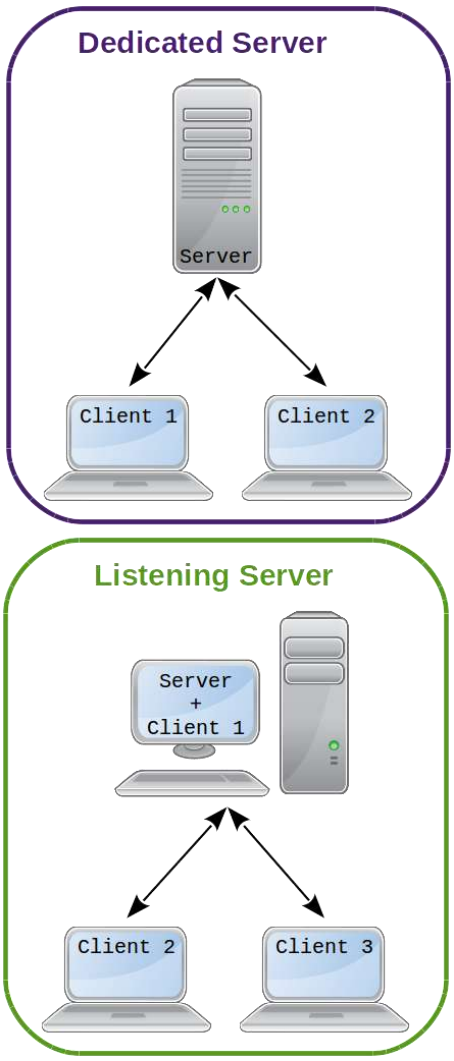


Figure 54. Client Server Models.

6.2 复制

在多人游戏中，所有状态、位置或其他属性相关的参与者都必须复制，以确保客户端之间的同步。这包括生成和销毁角色。AF Core 提供了许多解决方案来处理多人应用程序中的复制。

具体来说，广泛使用组件是为了促进复制。

不管客户端服务器模型和启动器是什么，所有的复制进程都要通过服务器，并且复制进程的数量受到网络带宽的严格限制。

6.2.1 状态组件和交互组件

状态组件和交互组件(如选择组件或活动组件)都实现了复制的能力，前提是将拥有的参与者设置为复制。组件系统已经特别调整为多人应用程序提供基础。这尤其涉及到棋子上的控件组件，该组件旨在使客户端能够访问控件设置。

6.2.2Non-predetermined 运动

参与者和组件的移动由 3.4 节中描述的复制组件复制。这些组件是专门设计来复制由兵抓住演员或物理引起的非预定运动。它们还意味着复制兵在关卡中的移动。

没有必要复制预防措施

复制组件对位置和旋转的复制是一个资源密

集型过程，例如钟摆的摆动，应该尽可能少使用。因为预定运动会自动

与所有客户端同步

服务器在同一时刻启动。

6.2.3 角色的生成和删除

具体来说，衍生过程与的问题密切相关
所有权，因为每个参与者都属于发起生成过程的实体，该实体可能是服务器或客户端。这对参与者复制的能力产生了严重的后果，因为只有由服务器发起的生成进程才会被复制。这是在衍生组件(参见 3.8.3 和 3.8.4 节)中实现的，它应该专门用于在多人应用程序中衍生演员，因为由客户端生成的演员只存在于该客户端应用程序的实例中。参与者的删除应该由 delete 组件专门处理，该组件用于复制该过程。

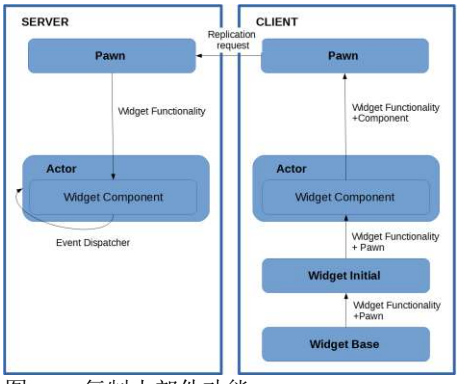


图 55。复制小部件功能。

6.2.4 用户界面

关于多人应用，用户界面呈现出许多问题，特别是在 VR 中。像托盘或菜单按钮这样的 UI 元素必须生成，并且还经常包含小部件来显示文本、按钮和图像。AF 核心提供了许多 UI 元素，但只有少数与多人游戏兼容。

•完全复制-由服务器生成，没有小部件

有关 UI 的更多细节，请参见第 8 节。

- 选择菜单——所有的按钮都是由服务器生成的，并且作为角色本身包含一个正常复制的选择组件
- 径向菜单——所有的按钮都是由服务器生成的，并且作为角色本身包含一个正常复制的选择组件

- 部分复制——通过服务器生成和交互，包含小部件
  - 托盘-由服务器在发起客户端的请求下生成。此外，与托盘的所有交互都是通过服务器重定向的，以使所有玩家都能与托盘交互，而不管它是谁生成的

  - 信息窗口-由服务器在发起客户端请求下生成，因此所有玩家都可以共享所提供的信息

- 无复制-由客户端生成，包含小部件
  - HUD -不能与其他玩家共享

小部件

widget 从未被编程为在多人环境中发挥作用。小部件只能在客户机上使用，不能在自身内部进行复制。只有触发参与者上的功能的小部件功能可以通过图 55 所示的信息流复制。这是高级框架无法更改的小部件的固有属性。

7 数据

Advanced Framework 支持许多方法来管理和提供静态数据以及有序的数据传输。并不是所有的数据实体，尤其是结构体和枚举类中的数据实体对开发人员都是必需的。有些仅以非常特定的方式使用，并且由使用它们的组件或其他实体更好地描述。其他的仅供内部使用。因此，我们将把对单个数据实体的描述限制在严格必要的范围内。

7.1 主要数据资产(PDA)

主要数据资产(PDA)基本上是一组预先安排好的变量，用于存储、安排和访问各种信息。因此，PDA 扮演着与数据表类似的角色，并且具有更直观的管理优势。该 AF 核心 4.1 提供以下类型的 PDA:

- 预设 PDA -为 VR 运动控制器提供按钮映射(参见 3.6.1 和 5.3.2 节)
- 面板 PDA——在基于面板的菜单中确定面板的内容和外观(参见 8.2.2 节)

用于手部跟踪的手势

数据资产遵循一个层次结构，顶部的数据资产类父级是主数据资产，主数据资产依次父级是 PDA 实例。PDA 和 PDA 实例之间的区别你可以想象，就像一个空表单和一个由人填写的表单之间的区别。但是，为了简单起见，我们将在文档中将术语“数据资产”粗略地用于 PDA 和 PDA 实例。

7.1.1 预置数据资产

预设 DA 提供按钮映射，将按键输入连接到运动控制器的控制器功能(见边距)。因此，预设只对虚拟现实应用重要。

主数据资产:PDA_ControllerVR

用于:控件组件(参见 3.6.1 节)

- 触发抓手-抓手由指定的按钮触发，并由指定的按钮握住。要释放指定的按钮，必须再次按下。
- 握握-要连续按住指定的按钮。在释放按钮时，所抓的参与者也会被释放。
- Key Mapping -包含一个元素用于每个控制器的按钮映射。
  - 键-确定按钮映射是用于左手运动控制器还是右手运动控制器

构建 PDA 要将 PDA 带入构建，必须将它们输入项目设置中的资产管理器。这是至关重要的一步。任何 PDA(不是 PDA 实例)，即使它是已注册 PDA 的子程序，也需要它自己的条目。任何已提交的 PDA 都无法进入构建应用程序，从而导致各种各样的问题。

控制器功能列表

- 选择
- 抓住
- LaserMode
- 平
- 暂停
- 级别 PDA -存储基本信息在每个级别像兵

(另见第 2.2 节)

- 手势 PDA -存储手指和手的位置，描述如何使用 PDA

主要数据资产基本上是一种模板，它合并了许多类型的变量。要为特定用例创建数据资产，请使用相应的 PDA 作为引用生成一个 PDA 实例，并填写所有变量。

数据资产

主数据资产 pda -实例

- RadialMenu1
- RadialMenu2
- ToggleFly
- 传送
- 移动
- MoveForward
- MoveLeft
- MoveRight
- MoveBackward
- TurnLeft
- TurnRight
- ResetOrientation
  - Value -包含一个元素，将 HMD 控制器的每个键连接到相应运动组件提供的控制器函数(通过名称)。

触觉和传送设置:

- 触觉倍增器-用于调整控制器的触觉反应强度

•传送执行-指定哪个按钮启动传送到跟踪的影响位置

- 触发-触发按钮启动传送
- NeutralThumbstick -传送被自动启动时

拇指杆返回其原始位置

- **OnTeleportButtonRelease -当分配到改变到传送模式的按钮再次被释放时，传送被自动启动**

•Teleport Deactivate -指定如何退出 Teleport 模式

- OnlyOnExecute -排除禁用传送模式的可能性
- 中性拇指杆-当拇指杆返回其原始位置时禁用传送模式。

•传送旋转类型-指定在传送过程中兵的旋转是否和如何实现

- 无-兵不会在瞬间移动时旋转
- 手动旋转-在传送时，棋子将根据运动控制器对其竖直位置的旋转而旋转，参见图 50
- 通过拇指杆旋转-兵将根据拇指杆的位置旋转。

不同的头盔显示器。使用此方法可以使每个 HMD 的响应保持一致。

不能与 NeutralThumbstick Teleport 去激活设置

这包括在搜索所需位置时必须按住激活传送模式的按钮。

不能与 OnThumbstickRelease 传送执行设置

触觉强度变化很大

目前可用的预置 DA

预置 DA 可以适应不同 HMD 的应用。AF 核心提供以下预设:

Oculus 触摸控制器的 Oculus Presetbutton 映射

Vive 预设按钮映射的 Vive 魔杖控制器

指关节的预设按钮映射指关节索引控制器

MR 预设按钮映射的窗口混合现实控制器

Vive Cosmos 控制器的 Cosmos Presetbutton 映射

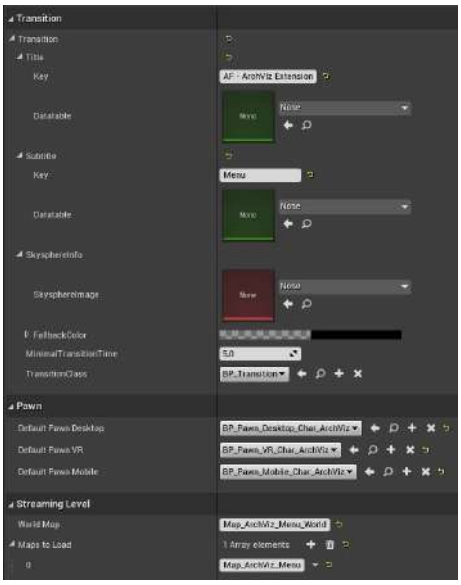


图 56。级别数据资产。

7.1.2 级别数据资产

关卡 DA 完全取代了之前包含关卡所有基本信息的信息关卡文件。每个关卡都需要自己的关卡 DA，由关卡地图中的地图信息参与者引用。

主数据资产(PDA): PDA_Level

用途:BP_MapInfo(章节 2.2)

设置:

- 过渡-决定了当关卡加载时过渡屏幕的外观

—Title—包含级别标题的键和 I18n 数据表

如果没有输入数据表或键为

– Subtitle -包含 level Subtitle 的键和 I18n 数据表
– Skysphereinfo -指定用于天空的纹理，如果没有找到图像，则指定回退颜色
– MinimalTransitionTime -以秒为单位表示转换的时间

屏幕最少显示
-过渡类-指定加载关卡时生成哪个 BP_Transition 角色(参见 2.4.2 节)。

•兵-根据环境为关卡分配兵

- Default Pawn Desktop -定义在桌面环境中使用哪个 Pawn

– 默认兵 VR -指定当 HMD 存在时使用哪个兵
– 默认典当移动-说明当操作系统是 Android 时使用哪个典当

•流水平-巩固需要加载的地图的水平

– 世界地图-指的是该关卡的世界地图
– Maps To Load -为每个 map 文件包含一个元素，必须为这个级别加载

应用程序中会显示缺少密钥本身。

尽快达到这个水平加载输入 0。

如果没有指定转换蓝图，则生成 BP_Transition_Default。

如果无法确定环境，则选择默认桌面卒

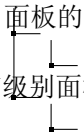
通过完整的文件名来引用每个映射

7.1.3 面板数据资产

面板 DA 整合了面板中面板类型小部件的内容大面板 PDA 大面板级 PDA 基于菜单。面板 PDA 形成一个层次结构，如小面板 PDA 的空白处所示，其中只有级别面板 PDA 应该用作 DA 的模板。小面板级 PDA

每个级别面板 DA 指定基于面板的菜单的一个面板的内容(见 8.2.2 节)。因此，每个应该从基于面板的菜单加载的级别都需要自己的面板级别 DA。AF 核心提供了两种面板设计，小水平面板和大

面板的掌上电脑



级别面板，主要区别在于其内容的搭配。

*小水平面板由一个主面板和 2 个侧面板组成，根据选择展开。*图 57 显示了一个展开的小型面板，主面板内容为紫色，侧面板内容为绿色。

设置:

- 图像 Large01 -纹理的图像在左侧面板
- Image Small01 -右侧面板上的左侧图像的纹理
- Image Small02 -右侧面板上右侧图像的纹理
- 级别信息-表示该面板可以访问的级别的级别 DA
- 口号——在主面板上指定关卡标题
- 详细信息-右侧面板上的文本字段的内容
- Logo -主面板上的小图像的纹理

推荐的决议

- 覆盖-用于主面板上的大图像的纹理
- 颜色-面板的背景颜色
- 大:1600 x975
- 小:800 x585
- 标志:250 x115

- 封面:650 x650

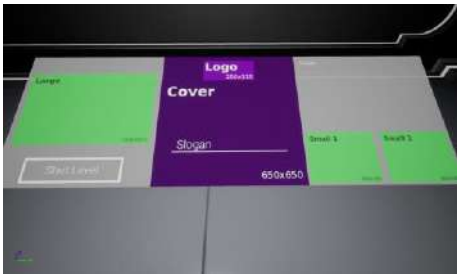


图 57。小面板示例。

- 左内容小部件-显示在左侧面板的小部件
- 右内容小部件-小部件显示在右侧面板

*大型面板由一个主面板和一个侧面板组成，在选择时展开。*展开的大型面板如图 58 所示，主面板内容为紫色，侧面板内容为绿色。

设置:

- 图片 Large01 -侧面面板顶部窗口的纹理
- 覆盖-用于主面板上的大图像的纹理
- 颜色-面板的背景颜色
- Content Widget -指定显示在侧面板中的小部件
- Image Small01 -侧面面板中左下角窗口的纹理
- 图像 Small02 -侧面面板中右下角窗口的纹理
- 级别信息-表示该面板可以访问的级别的级别 DA
- 口号——在主面板上指定关卡标题
- 详细信息-指定主面板中的左边文本字段
- 附加文本——在主面板中指定正确的文本字段
- Logo -主面板上的小图像的纹理

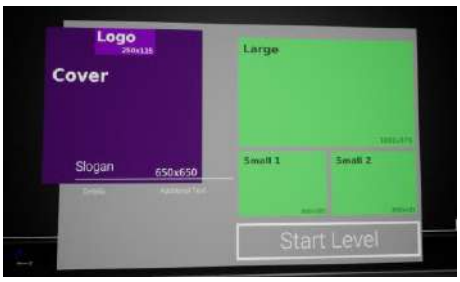


图 58。大面板示例。

7.1.4 手势数据资产

每个 Gesture PDA 实例通过它的手指和 NOTE 手指向 3 个参考点之一的位置定义一个手势。在控制控件中- 1)手势 PDA 仅限于棋子的 sinponent(见 3.6.1 节)，手势 DA 可以分配给 gle 手势，尽管你的控制器功能允许玩家执行功能，如可以定义相对位置选择，抓取和许多其他手跟踪模式。另一方面。**Primary Data Asset (PDA): PDA_Gesture 2)有些手势是不重叠的**

用于:Comp_Controls(章节 3.6.1)ping 镜像，不能

设置:描述在一个手势 PDA

- 手指需求-包含一个元素的数组，该元素描述了每个手指的两个位置。在这些情况下，我们建议创建一个单独的 ges-
  - 手指-定义手指这个要求被分配给每只手的 true DA。
  - 位置-定义手指的位置

*伸直 180——手指与手掌成 180° 角

*伸直 90° ——手指与手掌成 90° 角

向内收缩——手指收缩，触摸手掌

向外收缩——手指收缩而不接触手掌

*捏——触摸同一只手的拇指尖

-公差-与标准位置的偏差，单位为厘米

handtracking。 这涵盖了不同的手大小和手部协调能力差。

- Not -将要求转换为禁忌，意思是所有的立场

除了指定的一个被接受来创建手势

- 手的要求-数组包含一个元素描述手的位置指向一个参考点
  - 手方向-定义用于比较的手的方向
  - 其他组件-确定哪个实体作为手位置的参考点

*棋子——使用棋子的枢轴作为参考

- Not -将要求转换为禁忌，意味着除了指定的位置外，所有位置都被接受来创建手势

手部协调能力差。

7.2 数据表

AF Core 使用数据表主要是为了提供更新语言

*相机-使用兵看的方向作为参考

* OtherHand -使用另一只手的支点作为参考

  - 相对于-定义指针方向需要平行的参考点的方向
  - 角度公差-与指定位置在程度上的偏差

这涵盖了不同的手大小和

带有翻译的小部件的功能。为此提供的数据表称为 I18n 数据表。

I18n 数据表

I18n 数据表被划分为行。每一行都被分配了一个键，该键与当前语言设置一起明确指向数据表中的一个单元格，该单元格提供了小部件最终显示的文本。

基础结构:Struct_I18n_Key

7.3 结构

struct 组合了许多变量，这些变量通常被传输或

完整名称:Struct_I18n_Key

一起用于一组该框架利用这一概念来标准化可翻译文本、组件之间的通信和其他功能。AF Core 的许多结构只在内部使用，并且是自动生成的。但是，其他的则特别在组件设置中使用。所以我们将在这里提供 AF 核心结构的选择。

I18n 键结构

用于几乎所有的文本显示在 AF 核心，I18n 键结构使数据表基于更新语言功能实现

大多数小部件。

- **Key -定义 I18n 数据表中的一行，用于翻译文本**
- **数据表——确定函数在哪个数据表中搜索键**

有关 I18n 数据的更多信息

Figure 59. I18n Data Table.

> **Note**
> While it may seem unnecessary we recommend the use of multiple I18n data tables organized by usage of the texts tabulated, since huge data tables are very difficult to maintain.

用于翻译的表见 7.2 节。

图形按钮结构

用于 UI 元素，如选择菜单(参见 8.3.1 节)，以定义按钮的外观。

完整名称:Struct_Button_Graphic_2D 内容:

- **MainTexture** - specifies a texture to be shown on the button
- **Material** - specifies a material for the button

the Unreal Engine

The different parts of the struct are used in succession with the material replacing the texture if it was not found. And the color acting as ultimate fallback in case neither a texture nor a material are defined or are found.

Text Button Struct

Used in most widgets the text button struct provides a button with a name and texture.

Full Designation: `Struct_Button_Text`

Content:

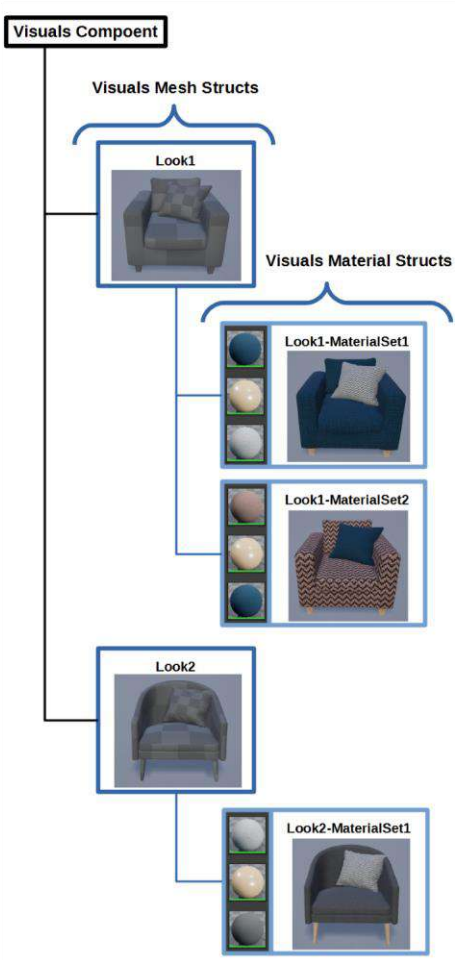
- **ButtonText** - contains a I18n key struct to display a translatable button name
- **ButtonImage** - determines the texture of the button

Visual Mesh Struct

Full Designation: `Struct_Visual_Mesh`

- 的线性颜色结构为按钮定义颜色

Not recommended for 3D buttons.



视觉组件(参见 3.2.11 节)允许为参与者提供不同的外观和材料集，这些外观和材料集是通过一组互锁结构定义的。这些结构体中最基本的是视觉网格结构体，它包含了演员一个外观上的所有必要信息。

内容:

- **Name - I18n 结构体，为外观提供可翻译的名称**
- **静态网格-静态网格对应的外观**
- **骨骼网格-骨骼网格对应的外观**
- **材质——视觉材质结构的数组，用于描述不同的材质集**
- **定义访问该外观的按钮外观的结构**
- **DataAsset—与外观相对应的数据资产**

视觉材料结构

在视觉网格结构中，视觉材料结构用于描述属于一个外观的各种材料集之一。每个材料集都由用于网格的所有材料组成，即使它们在不同的材料集之间没有变化。

- **ButtonContent -按钮结构，决定按钮访问材质集的样子**
- **材料-由材料集组成的材料数组**

7.4 枚举

完整名称:Struct_Visual_Material 内容:

- **Name - I18n 结构，为材料集提供可翻译的名称**

图 60。视觉结构方案。

每个外观必须有相应的静态或骨架网格。不可能混合或同时进入骨架网格和静态网格。

当前支持的语言

枚举提供了可修改的键列表，可用于切换功能或在应用程序的多个部分中显示。AF Core 使用的大多数枚举都在其固有环境中得到了最好的解释，即它们所使用的组件或 UI 或其他元素。但是，这里有少量的枚举需要解释。

I18n 枚举

I18n 枚举用于游戏实例和设置小部件中，以确定体验支持哪些语言

完整名称:Enum_I18n

内容:支持的语言各一项

- 英语
- 德国
- 法国
- 西班牙语
- 意大利

HMD 枚举

HMD 枚举用于在 VR 应用中调整控制器网格和功能映射到检测到的 HMD。

全名:Enum_HMD

内容:每个支持的 HMD 一个条目

目前支持的 HMD

- NoHMD
- SteamVR 万岁
- SteamVR 指数
- OculusHMD
- OculusQuest
- PSVR
- WindowsMixedRealityHMD
- 宇宙

控制器按钮 Enum

控制器按钮枚举用于运动控制器、运动组件，最重要的是用于预设数据资产的功能映射(见 7.1.1 节)，以管理按钮输入到控制器功能的分配。

全名:Enum_Controller_Buttons

内容:应用程序控件中包含的每个控制器按钮都有一个条目

控制器按钮

- 没有一个
- 触发
- 控制
- Face01
- Face02
- Face03
- Face04
- ThumbstickPress
- 肩膀
- ThumbStickUp
- ThumbStickDown
- ThumbStickLeft
- ThumbStickRight
- 菜单

手指枚举

手指枚举用于定义手势 DA(参见 7.1.4 节)中用于手跟踪的手指需求。

完整名称:Enum_HandTracking_Finger

内容:正常手每根手指一项。

手指

- 拇指•指数
- 中间
- 环
- 小指

3D 方向枚举

3D 方向枚举用于指向一个方向

完整名称:Enum_Directions_3D

- 向上-通常平行于 z+
- 向下-通常平行于 z•向前-通常平行于 x+
- 向后-通常平行于 x-
- 对，通常平行于 y+
- 左-通常平行于 y-

8 用户界面

AF 核心提供了几个用户界面选择从信息显示，以及过多的可能性，让玩家与经验互动。这包括基于小部件的 UI，如 HUD，信息窗口或托盘，以及无小部件的 UI，如选择菜单。然而，由于框架支持的环境多种多样，目前还不可能为每个环境提供所有这些选项，特别是因为 VR 环境不支持传统的 HUD 元素。下表将为您提供一个指南，说明哪些 UI 元素可以在哪些环境中使用。

请注意，这里的屏幕包含
桌面和移动环境。

	VR 只	屏幕上只	所有环境
小部件的基础	•托盘 • 微小的显示 • 小标签	•住房和城市发展部	•信息窗口 •基于面板的菜单
Widget-less	•径向菜单		•选择菜单

8.1 小部件

AF Core 提供了几个自定义的小部件，这些小部件可以用于 widget_initial

Widget_Base
使 用
Widget_Button，但也有一些父类，可以用来创建…个人的小部件。创建小部件时的主要挑战框架实现了可靠的交互，特别是在 VR 中。小部件交互的不真实的内在实现(通过小部件交互组件)并不适合框架的所有功能。因此，我们创建了自己的算法，递归小部件选择。当使用高级框架时，两个实现都可以并行使用。

- 小部件交互组件——小部件交互的虚幻的内在实现。它基于虚幻小部件交互组件。
- 递归小部件选择-高级框架固有的小部件交互实现，用于 AF 核心的所有基于小部件的 UI 元素。为了让递归小部件选择工作，必须为每个小部件提供一个函数，该函数指示小部件中包含按钮的所有子小部件。

基本小部件

作为 AF Core 中所有交互小部件的父类，并使用小部件组件实现有限数量的复制(参见 6.2.4 节)。

蓝图:Widget_Base

作为所有子小部件的框，并作为小部件组件(参见 3.5.4 节)与所有其他小部件之间的连接。

蓝图:Widget_Initial

框架的小部件组件已经针对 AF 固有的小部件进行了定制，并且只能顺利地与类的小部件一起工作

Widget_Initial 或它们的子类。它仍然支持虚幻小部件交互实现与正确的设置。有关更多信息，请参见 3.5.4 节。

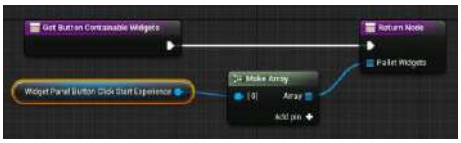


图 61。代码示例:Get Buttons 函数。

最初的小部件

8.1.1Widget 按钮

小部件按钮代表了一个专门
设计用于 widget_button_arrow 的类
框架的小部件和基于小部件的 UI 元素。尤其……调整到允许 VR 应用程序中的运动控制器与之交互
小部件。
每个按钮实现四个基本独立的状态:

- 按下-实际上是关闭状态的按钮
- 悬停-当输入设备或控制器在按钮上时
- 选中-基本上是按钮的开启状态
- 禁用-暂停按钮功能

小部件按钮-底座

小部件按钮基类作为所有小部件按钮的父类。实现小部件按钮的所有基本功能。但是，小部件按钮没有可视外观，只能作为父类使用。

蓝图:Widget_Button

8.1.2Pallet 小部件

托盘小部件是专门为显示它们的托盘 UI 设计的。其中大多数都是为其应用程序定制的，高度专门化，并且有许多子小部件。

多人游戏托盘 Widget 提供托管或加入多人游戏会话的所有信息和功能(图 62)。

蓝图:widget_initial_pallet_多人游戏内容:

- 标题框-显示小部件的标题
- 客户端字段——包含一个额外的小部件，提供可用服务器的概述
  - 标题-小部件名称和一个按钮刷新服务器列表
  - 服务器列表——包含一个搜索符号，一个可用服务器列表，如果没有找到服务器，则包含一个错误消息
- 搜索栏和键盘-允许搜索服务器的 IP
- 玩家字段-包含一个额外的小部件，提供一个回合中的玩家概述
  - 标题-小部件名称和一个按钮刷新播放器列表
  - 玩家字段-为会话中的每个玩家添加一个玩家字段小部件。这个小部件包含玩家名、ping 和静音按钮。
- 主机按钮-允许主机会话

菜单和设置托盘小部件

菜单和设置托盘包含许多小部件，这些小部件都是为它们的用途而定制设计的(图 63)。

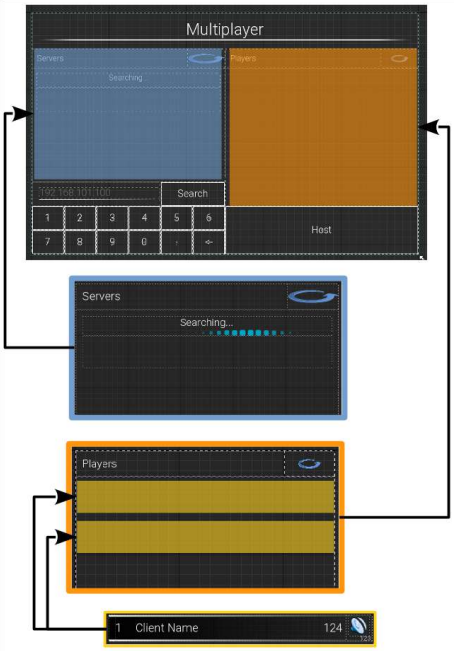


图 62。多人托盘小部件。

蓝图:Widget_Initial_Pallet_Menu

- 菜单小部件-包含一个简单的菜单，允许玩家重置级别，切换到菜单级别和退出应用程序
- 常规设置小部件-包含常规设置列表，如语言以及按钮，应用新设置和重置为默认值。
- 音频设置小部件-包含一个音频设置列表以及应用新设置和重置为默认值的按钮

- 图形设置小部件-包含图形设置列表以及应用新设置的按钮，自动检测最佳设置和重置为默认值。

调试托盘小部件作为测试应用程序的附加工具(图 64)。

蓝图:Widget_Initial_Pallet_Debug

抓住交互。

- 通用字段-显示 FPS 和自定义调试功能

图 64。调试托盘小部件。

8.1.3HUD 小部件

通常，HUD 小部件实现了与托盘小部件类似的功能。然而，由于 HUD

- 虚幻类字段-检查 AF 游戏类是否设置正确
- 兵场-在 VR 兵的情况下，说明当前兵和运动控制器
- 跟踪按钮-激活视景视图的光学跟踪，选择和

图 63。设置托盘小部件。

只提供有限的空间，我们创建了新的调整 HUD 小部件。

HUD 初始小部件

HUD 初始小部件包含一个按钮列表，这些按钮授予对其他 HUD 小部件的访问权限。

蓝图:Widget_HUD_Initial

多人 HUD 小部件

提供主机或加入多人游戏会话的所有信息和功能。

蓝图:Widget_HUD_Multiplayer

拟帧:左内容:

- 主机按钮-允许主机会话
- 客户端字段——包含一个额外的小部件，提供可用服务器的概述
  - 标题-小部件名称和一个按钮刷新服务器列表
  - 服务器列表——如果没有找到服务器，则包含搜索符号、可用服务器列表或错误消息

图 65。住房和城市发展部初始

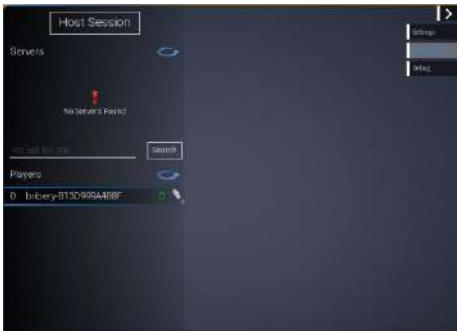


图 66。住房和城市发展部多人。

- 搜索栏-允许搜索服务器的 IP
- 玩家字段-包含一个额外的小部件，提供一个回合中的玩家概述
  - 标题-小部件名称和一个按钮刷新播放器列表
  - 玩家字段-为会话中的每个玩家添加一个玩家字段小部件。这个小部件包含玩家名、ping 和静音按钮。

HUD widget 设置

菜单和设置托盘包含许多小部件，这些小部件都是为它们的用途而定制设计的(图 67)。

蓝 图:Widget_HUD_SettingsPage 预 期 框 架:HUD_Widget_Frame_Left 内容:

- 菜单小部件-包含一个简单的菜单，允许玩家重置级别，切换到菜单级别和退出应用程序
- 常规设置小部件-包含常规设置列表，如语言以及按钮，应用新设置和重置为默认值。
- 音频设置小部件-包含一个音频设置列表以及应用新设置和重置为默认值的按钮
- 图形设置小部件-包含图形设置列表以及应用新设置的按钮，自动检测最佳设置和重置为默认值。

调试 HUD 小部件

作为测试应用程序的附加工具。HUD 小部件可以在不同调试工具的三个子小部件之间切换 68)。

蓝图:Widget_HUD_Debug

拟帧:左内容:

- 虚幻类字段-检查 AF 游戏类是否设置正确
- Pawn Field -在 VR 兵的情况下，声明当前的兵和运动控制器，并提供三个按钮来激活视野的光学轨迹，选择和抓取交互。
- 通用字段-显示 FPS 和自定义调试功能

8.2 基于 widget 的 UI

基于小部件的 UI 元素的基本方法包括生成或放置显示定制小部件的 actor 以及一些基本功能(如删除函数)。总的来说，AF 核心实现了 6 个不同的 UI 元素。

- 可托盘生成，可抓取的参与者，显示可交互的 wigdet
- 基于面板的菜单-自动生成的交互式菜单
- 信息窗口-可生成的演员，显示可交互的小部件
- 微小显示-描述抓取的演员的小显示
- 迷你标签-显示有限数量的信息的小部件

图 67。住房和城市发展部初始

蓝图:BP_InfoWindow

设置:参见 3.5.5

设置概述
信息窗口 UI 依赖于一个交互组件来生成它，一个 UI 组件来指定它的内容。 • 交互组件——根据 3.1.1 或 3.1.2 节设置的选择类型组件或 gazeview 组件 • UI 组件-窗口组件的设置，如节所述 3.5.5

8.2.2 基于面板的菜单

允许玩家通过自动生成的菜单进入应用程序的所有或选定的关卡。在基于面板的菜单中，每个级别都由一个可选择的面板表示。每个面板的内容由其面板级数据资产总结(见 7.1.3 节)。选择后，面板展开，显示一个开始按钮，允许访问该关卡。基于面板的菜单有两种设计:

- 大面板菜单-主要面板被安排在一行，使玩家可以滚动。选择后，主面板展开一侧

图 68。住房和城市发展部多人。

图 69。信息窗口

8.2.1Info 窗口

提供一个可定制的环境来显示参与者的其他信息。信息窗口是通过在参与者附近的合适位置选择组件来生成的，并显示一个或多个由处理删除和抓取逻辑的蓝图封装的小部件。(图 69)

与许多 UI 元素一样，信息窗口以其最基本的形式呈现在 AF 核心中。对于更详细的信息窗口检查 AF 扩展。

图 70。右侧的菜单面板如图 70 所示。

- 小面板菜单-主面板像瓷砖一样排列在行和列中(没有滚动)。选择后，主面板移动到其他面板的前面，展开两个侧面板，如图 71 所示。

Figure 71. Small Panel Menu

OverviewonSetup

The panel based menu relies on the collaboration of the panel menu actor which displays the menu in the application and the panel menu data assets which provide the content for each level.

- Data Assets - create a panel level DA for each level, that should appear in the panel based menu using as template *either*
  - PDA Panel Small Level

  or
  - PDA Panel Large Level

  and enter settings according to section 7.1.3.
- In the Map - add an instance of the panel menu actor corresponding

基于面板的菜单的两种设计都以相同的方式设置，但它们不能混合。

蓝图:BP_Demo_LevelMenu_Large, BP_Demo_LevelMenu_Small

到您的面板级别 DA 到菜单映射。

- 在 Panel Menu Actor 上-在菜单级别的 Panel Menu Actor 实例上访问 Panel Menu 组件，并输入所有 Panel Menu DA。

8.2.3Pallets

托盘是显示交互式小部件的参与者，可以抓取、生成和删除。托盘通常由径向菜单 UI 派生(参见 8.3.2 节)，并专用于特定的功能，如显示设置或一组函数。作为 UI 元素，托盘主要替代非 vr 应用的 HUD。托盘 UI 元素从它所显示的小部件中获得其功能。AF Core 提供了许多托盘小部件，主要作为这个 UI 元素的强大功能的示例。

设置概述
托盘是显示托盘小部件的参与者。大多数小部件已经包含了所有必要的功能。只有菜单小部件需要菜单小部件组件，该组件指定菜单级别的 info DA。要了解更多信息，请参阅关于径向菜单的第 8.3.2 节。

8.2.4Tiny 显示

可用的托盘部件

- 小部件初始托盘调试
- Widget 初始托盘多人游戏
- Widget 托盘设置
- 小部件托盘设置常规
- Widget 托盘设置图形
- 小部件托盘菜单

The tiny display is a special function of the laser motion controller. A tiny display is spawned when the laser motion controller grabs an actor giving a name and an image as well as the distance of the actor from the motion controller and rotation angle around the z-axis.

Blueprint BP_ControllerDisplay

Settings: see section 3.5.3

Figure 72. Laser controller with tiny display

OverviewOnSetup
The tiny display is a pretty simple device, that only works on the laser motion controller. All logic for spawning and deleting the display is inserted in the motion controller. Consequently, the only setup necessary is adding the tiny display component to the participating actors and entering the desired information.

8.2.5 Mini Tags

Mini tags are spawnable actors for minimalist information display. Mini tags are encapsulated in the mini tag component (see section 3.5.6). Each mini tag consist of a widget showing the text element and a line connecting the mini tag to its actor.

Blueprint BP_MiniTag

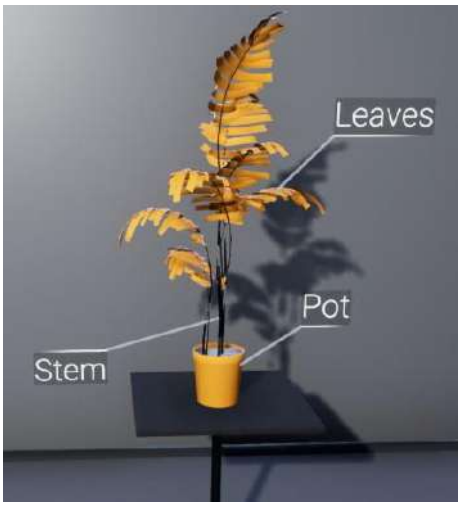


Figure 73. Actor with mini tags

设置:参见 3.5.6 节

设置概述
迷你标签的所有设置都封装在迷你标签组件中，必须将其添加到每个应该显示迷你标签的参与者中。

8.2.6HUD

Advanced Framework 的 HUD 由三种协作小部件类型组成，允许互动性和可互换的内容。

- 菜单——菜单是根据 HUD 组件的设置自动创建的(参见 3.6.4 节)，并授予访问 HUD 的可互换小部件的权限
- 框架-框架是专门的小部件，充当屏幕上的槽，并管理可互换的内容小部件。每当打开一个新的小部件时，它所分配的框架都会检查它是否被占用并移除

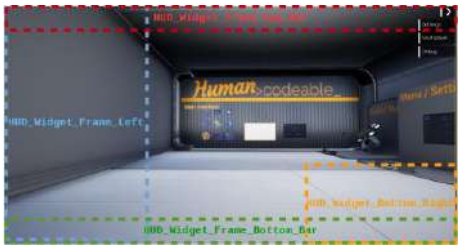


图 74。帧定位

小部件在必要时占用它。

- HUD 内容——HUD 小部件为 HUD 提供实际内容。它们由菜单访问并占据屏幕上的帧。

设置概述
HUD 是在关卡开始时自动创建的。它由菜单和许多其他小部件组成，比如十字准星导航模式中间的十字。菜单小部件的内容由 HUD 组件决定，它为每个 HUD 元素分配一个名称、一个小部件和一个框架(参见 3.6.4 节)。

8.3 widget-less UI

除了传统的基于小部件的 UI 元素，AF 核心提供了大量的 UI 元素，特别是为 VR 环境，放弃小部件。这些无小部件的用户界面对于多人应用程序特别方便，因为小部件意味着许多关于复制的问题(参见 6.2 节)。

8.3.1Selection 菜单

图 75。按钮排列

选择菜单是由 Comp_Select_SelectionMenu 实现的(参见 3.1.1 节)。它由一组预定义的按钮组成，这些按钮在方便的组装中产生，可以使用运动控制器或其他输入方法进行交互。如果需要，将自动生成额外选择的子集。因此，选择菜单提供了一种直观的方式来整合一个参与者的多个功能，这对于定制目的特别方便。

图 76。按钮排列行和列

实现在:选择菜单选择组件(章节 3.1.1)

- 更改演员的视觉外观
- 切换动画，视频或灯光

- 打开或关闭一个 actor
- 切换有关参与者的信息窗口
- 删除一个参与者

设置概述

选择菜单依赖于许多不同组件的合作，这些组件需要添加到参与者中，并进行相应的设置，以使选择菜单正常工作。以下是一个简短的总结:

- 交互组件-选择菜单选择组件与设置根据第 13 页
- 状态组件——根据 3.2 节的描述，为每个所需的按钮设置一个状态组件，并在选择组件上引用一个单独的标记。

图 77。选择菜单。

8.3.2Radial 菜单

径向菜单封装在运动径向按钮托盘上的运动组件中

Radial Button

VR 兵的径向按钮控制器(参见章节 5.3.3)。因此，radialFunctions 托盘菜单是一个仅限 vr 的 UI。它由一个圆形的按钮阵列组成，在打开径向菜单时生成径向按钮，在打开径向多人托盘时删除关闭径向按钮菜单或选择按钮。每个径向菜单按钮都是一个子菜单按钮

设置托盘

径向按钮-快速

- 8.2.3 节)
- BP_Radial_Button_Quick -作为自定义径向菜单的模板

产卵

所有按钮都实现了选择功能，可以由其他运动控制器选择，也可以使用拇指杆和触发按钮。按下按钮后，径向菜单自动关闭

选中。

目前实现的径向按钮

bp_radial_button_托盘-生成一个托盘 UI 元素(seeRadial 按钮-快速

设置概述

径向菜单是 VR 运动控制器和主要 VR UI 元素、托盘以及 UI 本身之间的主要连接。因此，它需要在兵和运动控制器上安排必要的组件。

- 在 VR Pawn 上-添加径向菜单组件，并按 3.6.2 节所述输入设置
- 在运动控制器上-根据 5.3.3 节设置径向菜单运动组件
- 在预置 DA -添加一个打开径向菜单的键到键映射，如 7.1.1 节和 3.6.1 节所述
- BP_Radial_Button_Pallet_Custom
- BP_Radial_Button_Pallet_多人
- BP_Radial_Button_Pallet_Settings
- BP_Radial_Button_Pallet_Debug
- BP_Radial_Button_Quick
- BP_Radial_Button_Quick_Spawn

使用拇指杆和触发按钮的径向菜单导航已经内置。

绝对位置:物体相对于地图原点的位置。锚:您添加到接受锚定过程的参与者的组件，以确定附加的参与者(节)的位置和方向

3.3.3)。

非对称游戏玩法-这里具体指的是一个 VR 应用程序，它结合了由玩家使用 HMD 控制的 VR 兵和由另一个玩家使用桌面和键盘控制的非对称兵。

非对称兵——VR 应用中的附加兵，由 VR 兵生成，以促进非对称游戏(章节 5.3.5)。

B

基础兵:框架(第 5 节)中所有自定义兵的父类。体槽:放置在角色兵上的锚。

按钮映射:通过预设将物理控制器的按钮分配给运动控制器的功能。

9 术语表

一个

C

角色兵:为玩家提供虚拟身体组件的兵类:实现功能的可交换功能元素(第 3 节)

D

数据表:数据表是不真实的固有表，功能类似于最简单形式的数据库。每个条目都由一个键标识，该键允许函数返回相应单元格的条目。框架使用数据表主要用于国际化(I18n，第 4 节)桌面棋子:屏幕棋子的子类，它被调整为使用键盘和鼠标导航应用程序(见 5.4 节)

一个应用程序。**游戏状态:当一个关卡被加载时，管理转换地图。**
Gazeview:使用玩家摄像机的向前轨迹来突出和选择演员的交互方法。
grab:通过将一个 actor 附加到运动控制器(VR)或 pawn(其他环境)来移动应用程序中的 actor

H

助手:主要是自动创建的角色，以支持其他类执行它们的功能。

我

影响位置:在屏幕环境接口中，参与者被抓住或锁定的位置:信息传输的标准。不要与结构体混淆。**介绍:应用程序启动时发生的第一件事。**通常它会显示一个天空和一个标志。

J

K

l

latch:通过将运动控制器(VR)或 pawn 附加到 actor 上来移动应用程序中的 actor 或网格组件。
关卡:玩家在应用过程中访问的独立设计地图或两者的组合。实际上你所分配的每个结构都带有关卡数据资产。
关卡数据资产:收集关卡中所有重要信息的数据资产，包括世界地图、过渡屏幕内容和兵。

米

菜单关卡:在你的应用程序中加载的第一关的地图，很可能包含一个菜单。
运动组件:在 VR 中实现控制器功能的组件(第 5.3.3 节)运动控制器:用于 VR 应用程序中运动组件的容器(第 5.3.2 节)

E

事件分派器:通知函数，使参与者能够对其他参与者的更改做出反应。

F

G

游戏实例:当玩家改变关卡游戏模式时，管理信息传递(最重要的是关卡键):在开始时生成玩家控制器和游戏状态

N

O

P

面板数据资产:在基于面板的菜单中整合面板的内容。基于面板的菜单:自动生成的菜单暂停映射:当玩家按下暂停键时加载的默认映射。
玩家控制器:刷出和控制兵，管理到暂停映射的过渡和返回预置:为运动控制器定义按钮映射的数据资产

问

R

相对位置:一个项目相对于一个选定点的位置，不与地图的原点重合。复制 actor:在多人游戏中复制其位置或状态以跟踪其他玩家动作的 actor

年代

screen pawn:为不在 VR(章节 5.1)结构中的应用程序调整的所有 pawn 的父类:容器中保存通常一起传输的选定变量。

T

转换映射:在关卡加载之前和加载时加载和显示的默认映射。
超时(高亮显示):从项目中删除高亮显示的时间。
触发器:可以通过操作来激活对象上的功能的参与者
接收的演员

U

UI: UI 或用户界面允许玩家访问信息
选择功能

V

视觉组件:允许更改参与者实例的外观。

W

widget:用于创建 2D 元素并在应用世界地图中显示的系统:实际上是一个关卡的空容器

X

Y

Z