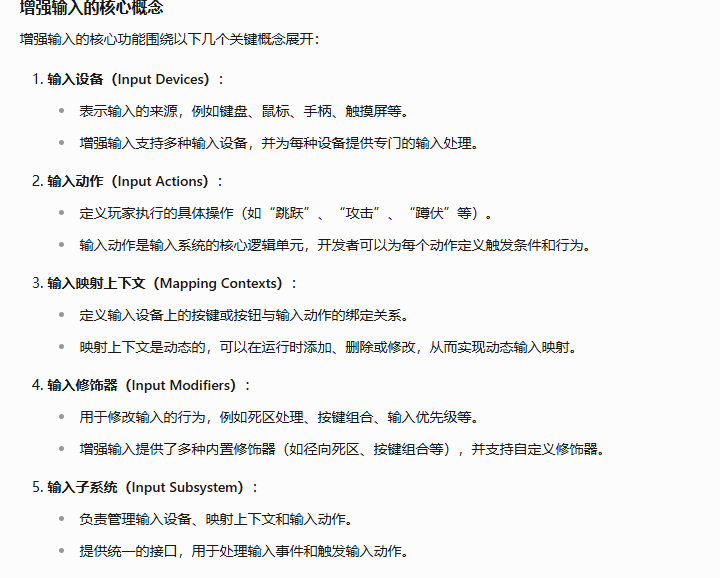
**虚幻引擎5**（UE5）项目有时会需要更多高级的输入功能，例如复杂的输入处理，或在运行时重新映射输入按键。**增强输入（Enhanced Input）** 为开发人员提供了这类功能，并能向上兼容 **虚幻引擎4**（UE4）的默认输入系统。

此插件实现了多种功能，例如径向死区、同时按键、上下文输入和优先级安排，并且能够在基于 **资产** 的环境中，拓展对于原始输入数据的筛选和处理功能。

**动态和上下文输入映射**

使用增强输入时，你可以在运行时为玩家添加和删除 **映射上下文（Mapping Contexts）** 。这样可更轻松地管理大量 **操作（Actions）** 。你可以根据玩家的当前状态更改特定输入的行为。

例如，如果你有一个可以行走、冲刺、俯卧的玩家角色。对于其中每种角色移动类型，你可以交换映射上下文，让CTRL键执行不同的操作。行走时按下CTRL键，角色应该蹲伏。冲刺时按下CTRL键，角色应该滑行。俯卧时按下CTRL键，角色应该重新站起来。







**输入动作（Input Actions）** 是增强输入系统和项目代码之间的通信链接。输入动作在概念上相当于 **操作（Action）** 和 **轴（Axis）** 映射名称，但它们是数据资产。每个输入动作应该表示用户可以执行的某件事，例如"蹲伏"或"发射武器"。你可以在蓝图或C++中添加 **输入侦听器（Input Listeners）** ，侦听输入动作的状态何时发生变化。

输入动作可以是多种不同的类型，这些类型将确定行为。你可以创建简单的布尔动作或更复杂的3D轴。动作类型将确定值。布尔动作采用简单的 **布尔** 值， **Axis1D** 为 **浮点** 值， **Axis2D** 为 **FVector2D** ， **Axis3D** 为整个 **Fvector** 。

**触发状态**

**触发状态（Trigger State）** 表示动作的当前状态，例如 **已开始（Started）** 、 **进行中（Ongoing）** 、 **已触发（Triggered）** 、 **已完成（Completed）** 和 **已取消（Canceled）** 。通常，你将使用"已触发"状态。你可以绑定到C++和蓝图中的特定状态。

* **已触发（Triggered）：** 动作已触发。这意味着它完成了所有触发器要求的求值。例如，"按下并松开"触发器会在用户松开按键时发送。
* **已开始（Started）：** 发生了开始触发器求值的某个事件。例如，"双击"触发器的第一次按键将调用一次"已开始"状态。
* **进行中（Ongoing）：** 触发器仍在进行处理。例如，当用户按下按钮时，在达到指定持续时间之前，"按住"动作处于进行中状态。根据触发器，此事件将在收到输入值之后在对动作求值时，每次更新触发一次。
* **已完成（Completed）：** 触发器求值过程已完成。
* **已取消（Canceled）：** 触发已取消。例如，在"按住"动作还没触发之前，用户就松开了按钮。

**输入映射上下文**

**输入映射上下文（Input Mapping Contexts）** 是输入动作的集合，表示玩家可以处于的特定上下文。它们描述了给定输入动作的触发规则。映射上下文可以动态地为每个用户添加、移除或安排优先次序。

要创建输入映射上下文，请右键点击 **上下文浏览器（Context Browser）** ，展开 **输入（Input）** 选项，然后选择 **输入映射上下文（Input Mapping Context）** 。

输入映射上下文的基本结构是一个层级结构，最上层包含一组输入动作。在输入动作层下面，是可以触发各个输入动作的用户输入，例如键、按钮和动作轴。

底层包含各个用户输入的输入触发器和输入修饰器列表，可用于确定如何筛选或处理输入的原始值，以及它必须满足哪些限制才能驱动顶层的输入动作。

你可以通过本地玩家的增强输入本地玩家子系统（Enhanced Input Local Player Subsystem）将一个或多个上下文应用到本地玩家，并安排它们的优先次序，避免多个操作由于尝试使用同一输入而发生冲突。

你可以在这里将实际的键与输入动作绑定，并为每个动作指定额外触发器或修饰器。将输入映射上下文添加到增强输入子系统时，你还可以指定其优先级。如果你有多个上下文映射到同一个输入动作，那么在触发输入动作时，将考虑优先级最高的上下文，而忽略其他上下文。

例如，你可以为一个可以游泳、行走、驾驶载具的角色提供多个输入映射上下文。一个用于通用动作（始终可用且始终映射到相同用户输入），其他分别用于各类移动模式。

开发人员随后可以将与载具相关的输入动作放入单独的输入映射上下文中，这些操作将在本地玩家进入载具时添加到玩家，并在退出载具时从本地玩家中移除。

这样做有助于确保不合适的输入动作无法运行，从而优化并预防漏洞。此外，使用互斥的输入映射上下文还有助于避免输入冲突，因此当某个用户输入被用于不同的输入动作时，该输入绝不会意外触发错误的操作。

**输入修饰器**

**输入修饰器（Input Modifiers）** 是一种预处理器，能够修改UE接收的原始输入值，然后再将其发送给输入触发器（Input Trigger）。增强输入插件随附多种输入修饰器，可以执行各种任务，例如更改轴顺序、实现"死区"、将轴输入转换为世界空间。

输入修饰器很适合用于应用灵敏度设置，在多个帧上平滑输入，或基于玩家状态更改输入的行为。由于你在创建自己的修饰器时可以访问 UPlayerInput 类，你可以访问所属玩家控制器，并获取所需的任意游戏状态。

**输入触发器**

输入触发器用于确定用户输入在经过一系列可选输入修饰器的处理后，是否会激活输入映射上下文中的相应输入动作。大部分输入触发器都会分析输入本身，检查最小动作值并验证各种模式，例如短暂点击、长时间按住或典型的"按下"或"释放"事件。此规则的一个例外是"同时按键"输入触发器，该触发器仅通过另一个输入动作触发。默认情况下，任意用户输入活动都会在每次更新时触发一次。

输入触发器有三种类型：

* **显式（Explicit）** 类型将使输入在输入触发器成功时成功。
* **隐式（Implicit）** 类型将使输入仅在输入触发器和所有其他隐式类型输入触发器都成功时成功。
* **阻碍（Blocker）** 类型将使输入在输入触发器成功时失败。

下面是一个逻辑示例，说明每种触发器类型在某种情况下如何与其他触发器类型交互：

|  |  |
| --- | --- |
|  | 隐式 == 0，显式 == 0 - 始终触发，除非值为0。 |
|  |  |
|  | 隐式 == 0，显式 > 0 - 至少一个显式已触发。 |
|  |  |
|  | 隐式 > 0，显式 == 0 - 所有隐式已触发。 |
|  |  |
|  | 隐式 > 0，显式 > 0 - 所有隐式和至少一个显式已触发。 |
|  |  |
|  | 阻碍 - 重载其他所有触发器，强制触发器失败。 |

复制完整片段

处理用户输入后，输入触发器可能返回以下三种状态之一：

* **无（None）** 表明未满足输入触发器的条件，因此输入触发器失败。
* **持续（Ongoing）** 表明部分满足了输入触发器的条件，并且输入触发器正在处理，但尚未成功。
* **已触发（Triggered）** 表明已满足输入触发器的所有条件，因此输入触发器成功。

你可以通过扩展输入触发器基类，或 **Input Trigger Timed Base** 来创建自己的输入触发器。