第3章

VR控制器

VR控制器是真实世界与虚拟世界交互的媒介，VR应用开发者需要了解VR控制器本身的状态以及VR使用者在VR控制器所进行的操作。本章讨论如何获取控制器的信息以及用户在控制器上的操作。

在Unity编辑器的Hierarchy视图中右击，在弹出的上下文菜单中选择XR→UI Canvas选项，在场景中建立UI画布，如图3-1所示，该UI画布将用于显示来自VR控制器的信息。

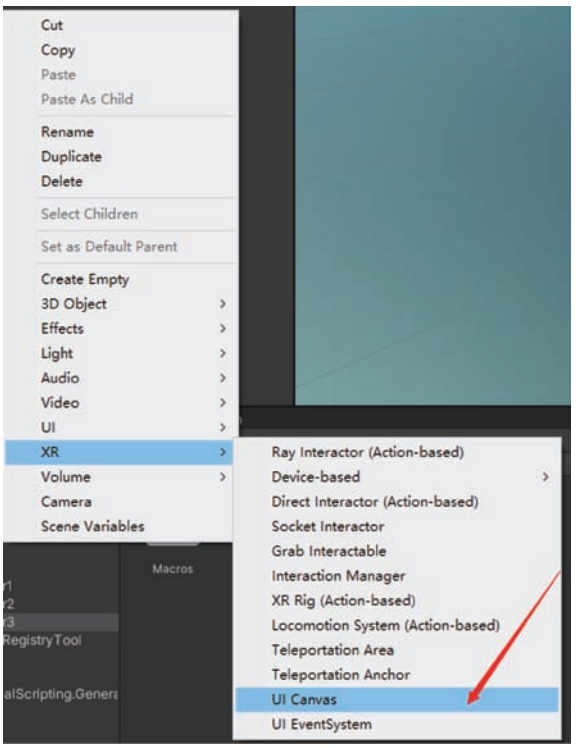

图3-1　选择XR→UI Canvas选项

调整所建立的 UI Canvas 的属性，使其能够出现在 VR 场景中头显的前方，如图 3-2 所示。

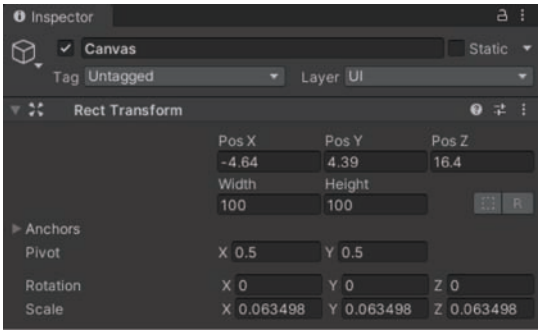

图 3-2　调整 UI Canvas 的属性

在 Hierarchy 视图中右击，在弹出的上下文菜单中选择 UI→Text 选项，如图 3-3 所示，在 UI 画布上建立文本标签。

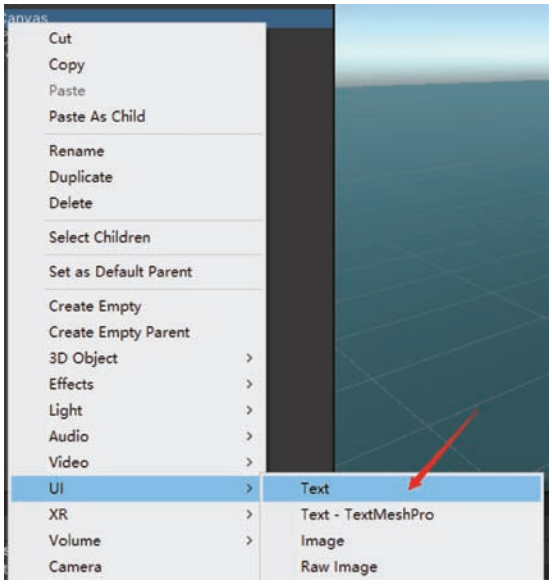

图 3-3　选择 UI→Text 选项

该游戏对象 Text 的名称为 textGetTrigger，并给其添加可视化脚本组件，指定宏名称为 UIGetControllerInfo，该宏将获取控制器上的扳机键是否被按下，如图 3-4 所示。

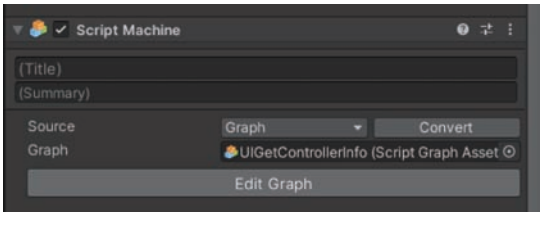

图 3-4　指定宏名称为 UIGetControllerInfo


视频讲解

3.1 获取控制器特定按键是否按下

Unity 的 XR 平台具有多种输入功能，可以在设计用户交互时加以利用。应用程序可以使用某些特定数据，这些数据引用位置、旋转、触摸、按钮、游戏杆和手指传感器。但是，在不同平台之间访问这些输入功能可能有很大差别。

Unity 提供了 InputFeatureUsage 的结构，该结构定义了一组标准的物理设备控件（例如 grip 按钮和扳机键），以访问任何平台上的用户输入。这些控件可通过名称识别输入类型。每个 InputFeatureUsage 对应一个常见的输入操作或类型，例如，Unity 将名为 trigger 的 InputFeatureUsage 定义为食指控制的单轴扳机键输入。无论使用哪种 XR 平台，都可以使用 InputFeatureUsage 通过名称来获取 trigger 状态，因此无须为常规 Unity 输入系统设置一个轴（或某些 XR 平台上的按钮）。

表 3-1 列出了标准控制器 InputFeatureUsage 的名称与常见 XR 系统的控制器按键之间的映射关系。

表 3-1 InputFeatureUsage 的名称与常见 XR 系统的控制器按键之间的映射关系

InputFeatureUsage 的名称	功能类型	控制器对应按键
primary2DAxis	2D 轴	触控板/游戏杆
trigger	单轴	扳机
grip	单轴	握把
secondary2DAxis	2D 轴	
secondary2DAxisClick	按钮	
primaryButton	按钮	主要
primaryTouch	按钮	
secondaryButton	按钮	备用
secondaryTouch	按钮	
gripButton	按钮	握把-按下
triggerButton	按钮	扳机-按下
menuButton	按钮	
primary2DAxisClick	按钮	触控板/游戏杆-按下
primary2DAxisTouch	按钮	触控板/游戏杆-触控

InputDevice 代表任何物理设备，例如控制器或头盔，它可以包含有关设备跟踪、按钮、游戏杆和其他输入控件的信息。使用 XR.InputDevices 类可访问当前连接到 XR 系统的输入设备。在 XR 系统断开与输入设备的连接之前，输入设备将跨帧保持有效。使用 InputDevice.IsValid 属性来确定 InputDevice 是否仍代表激活的控制器。可以通过特征、角色以及 XR 节点的方式访问输入设备。

设备特征描述了设备的功能或用途。InputDeviceCharacteristics 是一系列标志，可以添加到代码中，用于搜索符合特定规格的设备。可以按表 3-2 所示的特征筛选设备。

表 3-2 设备特征表

设　备	特　征
HeadMounted	设备连接到用户的头部。它具有设备跟踪和眼球中心跟踪功能。此标志常用于标识头戴式显示器（HMD）
Camera	设备具有摄像机跟踪功能
HeldInHand	用户将设备握在手中
HandTracking	设备代表物理跟踪的手。它具有设备跟踪功能，并且可能包含手和骨骼数据
EyeTracking	设备可以执行眼球跟踪并具有 EyesData 功能
TrackedDevice	可以在 3D 空间中跟踪设备，具有设备跟踪功能
Controller	设备具有按钮和轴的输入数据，并且可以用作控制器
TrackingReference	设备代表静态跟踪参考对象，具有设备跟踪功能，但该跟踪数据不应该更改
Left	将此特征与 HeldInHand 或 HandTracking 特征组合使用，可以将设备标识为与左手关联
Right	将此特征与 HeldInHand 或 HandTracking 特征组合使用，可以将设备标识为与右手关联
Simulated6DOF	设备报告 6DOF 数据，但仅具有 3DOF 传感器。Unity 负责模拟位置数据

底层 XR SDK 会报告这些特征，可以使用 InputDevice. Characteristics 查找这些特征。设备通常具有多个特征，可以使用位标志来筛选和访问这些特征。

设备角色描述输入设备的一般功能，可使用 InputDeviceRole 枚举来指定设备角色，如表 3-3 所示。

表 3-3 设备角色

角　色	描　述
GameController	游戏主机风格的游戏控制器
Generic	代表核心 XR 设备的设备，例如头戴式显示器或移动设备
HardwareTracker	跟踪设备
LeftHanded	与用户左手关联的设备
RightHanded	与用户右手关联的设备
TrackingReference	跟踪其他设备的设备，如 Oculus 跟踪摄像机

底层 XR SDK 会报告这些角色，但是不同的提供商可能会以不同的方式组织他们的设备角色。此外，用户可以换手，因此角色分配结果可能与用户握住输入设备的手不匹配。XR 节点表示 XR 系统中的物理参考点（例如，用户的头部位置、左右手之类的跟踪参考）。XRNode 枚举定义的节点如表 3-4 所示。

表 3-4 XRNode 枚举定义的节点

XR 节点	描　述
CenterEye	用户两个瞳孔之间的中点
GameController	游戏主机风格的游戏控制器。用户的应用程序可以有多个游戏控制器设备
HardwareTracker	硬件跟踪设备，通常连接到用户或物理项。可以存在多个硬件跟踪器节点
Head	由 XR 系统计算出的用户头部的中心点
LeftEye	用户的左眼

续表

XR 节点	描　　述
LeftHand	用户的左手
RightEye	用户的右眼
RightHand	用户的右手
TrackingReference	跟踪参考点，例如 Oculus 摄像机。可以存在多个跟踪参考节点

可以从特定的 InputDevice 读取输入功能，例如扳机键的状态。例如，要读取右扳机键的状态，应按照下列步骤操作。

（1）使用 InputDeviceRole. RightHanded 或 XRNode. RightHand 获取惯用右手设备的实例。

（2）有了正确的设备后，使用 InputDevice. TryGetFeatureValue 方法访问当前状态。TryGetFeatureValue()尝试访问功能的当前值，并根据情况返回不同的值。

① 如果成功获取指定的功能值，则返回 true。

② 如果当前设备不支持指定的功能，或者该设备无效（即控制器不再处于激活状态），则返回 false。

要获取特定的按钮、触摸输入或游戏杆轴值，应使用 CommonUsages 类。CommonUsages 类包括 XR 输入映射表中的每个 InputFeatureUsage，以及诸如位置和旋转之类的跟踪功能。CommonUsages 定义用于从 XR. InputDevice. TryGetFeatureValue 中获取输入功能的静态变量。使用这些静态变量可按使用情况获取 XR. InputDevice 的常用功能值，如表 3-5 所示。

表 3-5　XR. InputDevice 的常用功能值

功能值	作　　用
batteryLevel	表示设备的当前电池续航时间的值
centerEyeAcceleration	设备以眼睛为中心的加速度
centerEyeAngularAcceleration	设备以眼睛为中心的角加速度，采用欧拉角的格式
centerEyeAngularVelocity	设备以眼睛为中心的角速度，采用欧拉角的格式
centerEyePosition	设备以眼睛为中心的位置
centerEyeRotation	设备以眼睛为中心的旋转
centerEyeVelocity	设备以眼睛为中心的速度
colorCameraAcceleration	设备彩色摄像机的加速度
colorCameraAngularAcceleration	设备彩色摄像机的角加速度，采用欧拉角的格式
colorCameraAngularVelocity	设备彩色摄像机的角速度，采用欧拉角的格式
colorCameraPosition	设备彩色摄像机的位置
colorCameraRotation	设备彩色摄像机的旋转
colorCameraVelocity	设备彩色摄像机的速度
deviceAcceleration	设备的加速度
deviceAngularAcceleration	设备的角加速度，采用欧拉角的格式
deviceAngularVelocity	设备的角速度，采用欧拉角的格式
devicePosition	设备的位置
deviceRotation	设备的旋转

续表

功能值	作　　用
deviceVelocity	设备的速度
eyesData	包含从设备中收集的眼睛跟踪数据的眼睛结构
grip	表示控制器上的用户手柄
gripButton	表示设备是否被握住的二进制测量值
handData	表示设备的手柄数据的值
isTracked	告知开发人员当前是否在跟踪设备
leftEyeAcceleration	设备左眼的加速度
leftEyeAngularAcceleration	设备左眼的角加速度，采用欧拉角的格式
leftEyeAngularVelocity	设备左眼的角速度，采用欧拉角的格式
leftEyePosition	设备左眼的位置
leftEyeRotation	设备左眼的旋转
leftEyeVelocity	设备左眼的速度
menuButton	表示菜单按钮，用于暂停、返回或退出游戏
primary2DAxis	设备上的主触控板或游戏杆
primary2DAxisClick	表示被单击或按下的主 2D 轴
primary2DAxisTouch	表示被触摸的主 2D 轴
primaryButton	在设备上被按下的主要面按钮或唯一按钮（如果只有一个按钮可用）
primaryTouch	设备上被触摸的主要触控板或者摇杆
rightEyeAcceleration	设备右眼的加速度
rightEyeAngularAcceleration	设备右眼的角加速度，采用欧拉角的格式
rightEyeAngularVelocity	设备右眼的角速度，采用欧拉角的格式
rightEyePosition	设备右眼的位置
rightEyeRotation	设备右眼的旋转
rightEyeVelocity	设备右眼的速度
secondary2DAxis	设备上的辅助触控板或游戏杆
secondary2DAxisClick	表示被单击或按下的辅助 2D 轴
secondary2DAxisTouch	表示被触摸的辅助 2D 轴
secondaryButton	设备上被按下的辅助面按钮
secondaryTouch	设备上被触摸的辅助面按钮
trackingState	表示此设备跟踪的值
trigger	触发式控制，用食指按下
triggerButton	表示食指是否正在激活扳机键的二进制测量值
userPresence	使用此属性来测试用户当前是否佩戴 XR 设备和/或与之互动。该属性的确切行为因设备类型而异：有些设备有一个专门检测用户接近度的传感器，但当该属性为 UserPresenceState. Present 时，可以合理地推断出用户与设备在一起

要在 textGetTrigger 文本上显示右手控制器的扳机键是否按下，可视化脚本如图 3-5 所示。从右手控制器获取扳机键是否按下，如果按下则显示“Right Trigger Pressed!”，否则显示“Right Trigger Unpressed!”。

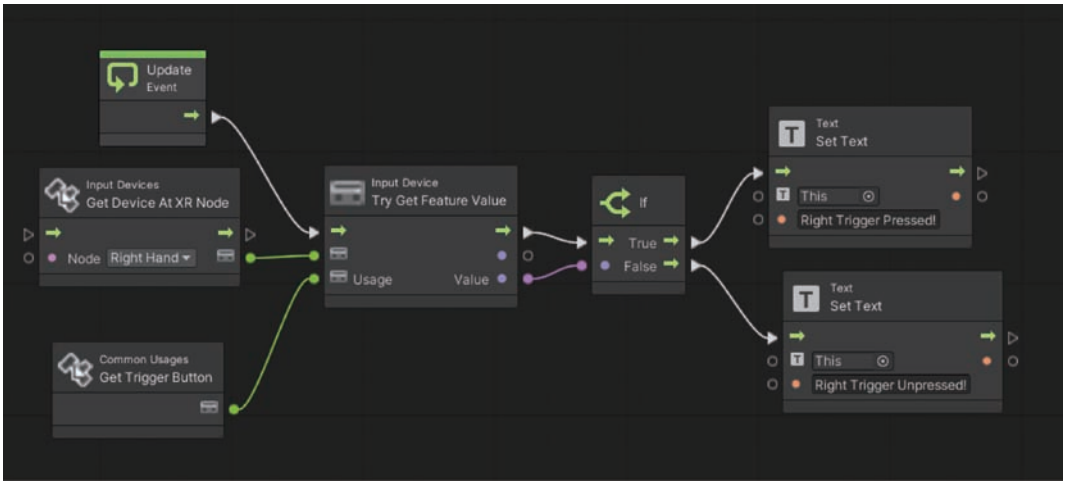

图 3-5 可视化脚本

视频讲解

3.2 获取控制器特定按键按下程度

复制 textGetTrigger 文本游戏对象，并改名为 textGetTriggerValue，如图 3-6 所示。

在场景视图中将游戏对象 textGetTriggerValue 移动至游戏对象 textGetTrigger 上方，如图 3-7 所示。

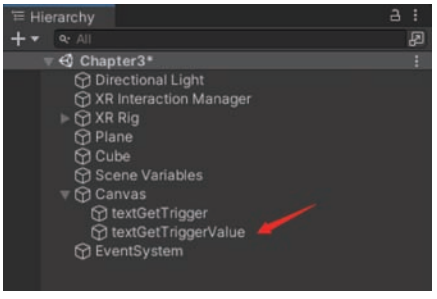

图 3-6 改名为 textGetTriggerValue

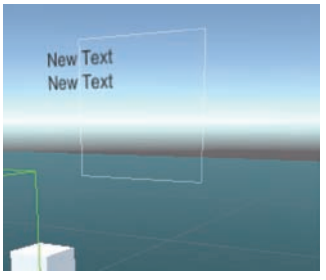

图 3-7 将 textGetTriggerValue 移动至 textGetTrigger 上方

为 textGetTriggerValue 添加可视化脚本组件，指定宏名称为 UIGetControllerTriggerValue，如图 3-8 所示。

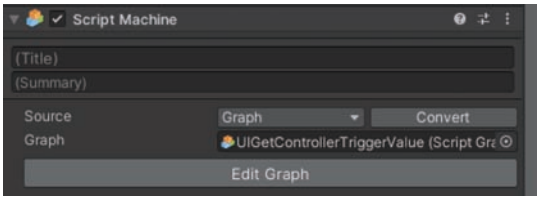

图 3-8 为 textGetTriggerValue 添加可视化脚本组件

UIGetControllerTriggerValue 宏的可视化脚本如图 3-9 所示。Unity 会根据扳机键按下的幅度返回[0,1]的浮点数，0 代表没有按下，1 代表完全按下。如果稍稍按下扳机键，在 textGetTriggerValue 上便会显示出相应的浮点数值。

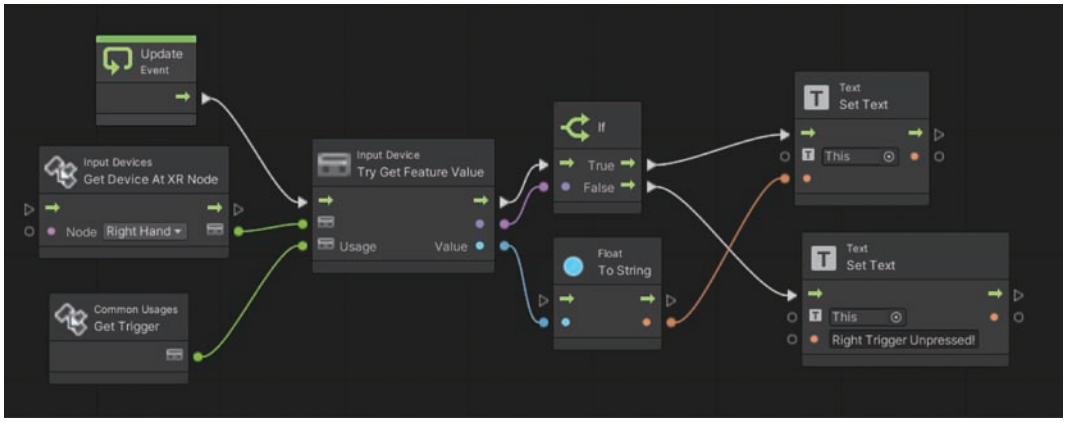

图 3-9　UIGetControllerTriggerValue 宏的可视化脚本

视频讲解

3.3　获取控制器触控板的输入

复制 textGetTriggerValue 文本游戏对象，并改名为 textGetTouchPad，如图 3-10 所示。

为 textGetTouchPad 添加可视化脚本组件，指定宏名为 UIGetControllerTouchPad，如图 3-11 所示。

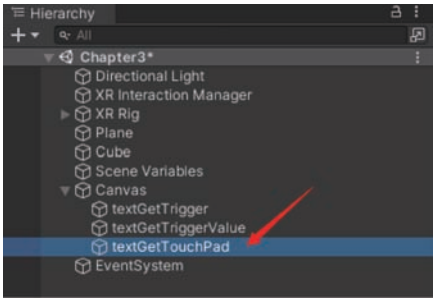

图 3-10　改名为 textGetTouchPad

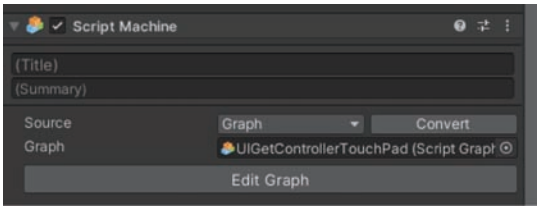

图 3-11　指定宏名为 UIGetControllerTouchPad

UIGetControllerTouchPad 宏的可视化脚本如图 3-12 所示。Unity 会根据手指在触控板按下的位置返回二维矢量< x,y >，x,y∈[－1,1]。当 x＝－1 时，手指位于触控板的最左边缘或者摇杆已经位于最左边沿；当 x＝1 时，手指位于触控板的最右边缘或者摇杆已经位于最右边沿；当 y＝－1 时，手指位于触控板的最上边缘或者摇杆已经位于最上边沿；当 y＝1 时，手指位于触控板的最下边缘或者摇杆已经位于最下边沿。

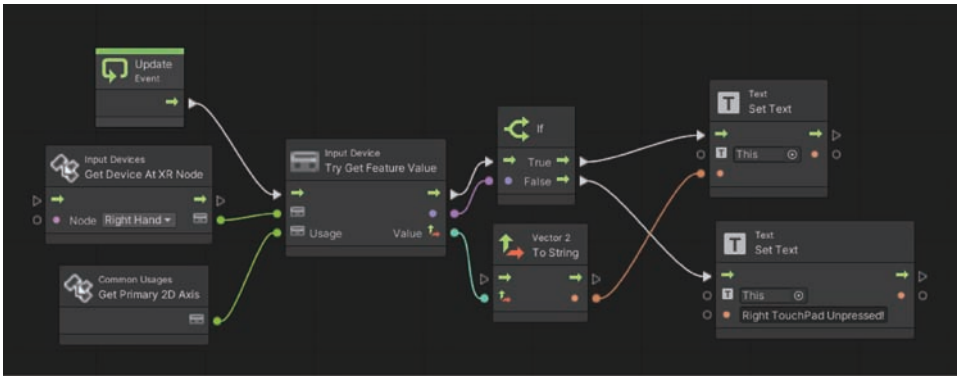

图 3-12 UIGetControllerTouchPad 宏的可视化脚本

3.4 获取控制器的位置信息

视频讲解

复制 textGetTouchPad 文本游戏对象，并改名为 textGetPosition，如图 3-13 所示。

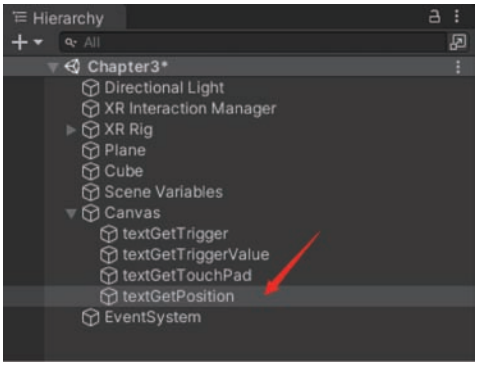

图 3-13 改名为 textGetPosition

为 textGetPosition 添加可视化脚本组件，指定宏名为 UIGetControllerPos，如图 3-14 所示。

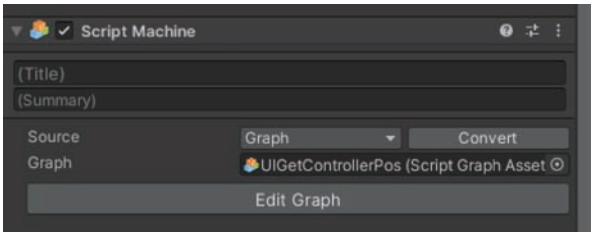

图 3-14 指定宏名为 UIGetControllerPos

UIGetControllerPos 宏的可视化脚本如图 3-15 所示。Unity 会获取右手控制器相对 XR Rig 原点的位置，并显示该位置的变化。

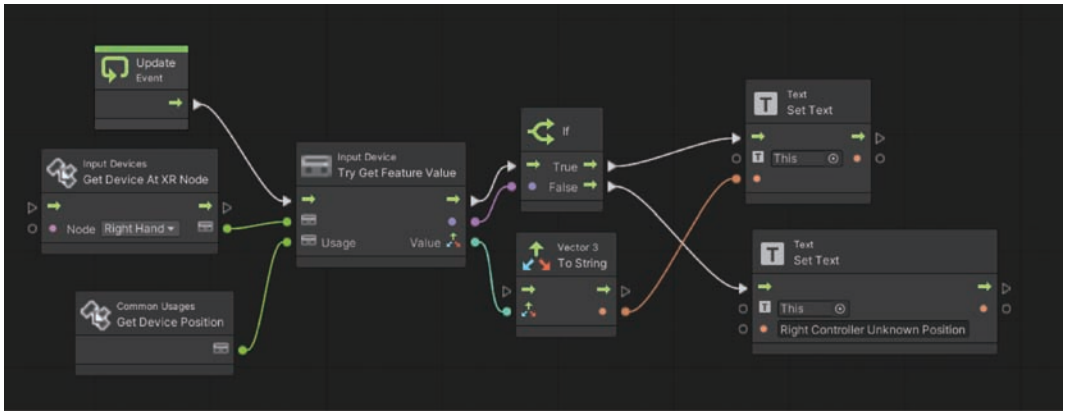

图 3-15　UIGetControllerPos 宏的可视化脚本

视频讲解

3.5　定制虚拟手

目前 VR 环境中仅用两根红线表示控制器，可以使用定制的虚拟手来代表对应的控制器，这样对用户来说更加直观一些。在 Hierarchy 视图中右击，在弹出的上下文菜单中选择 XR→Device-based→Direct Interactor 选项，如图 3-16 所示，建立名为 VRLeft 的游戏对象。

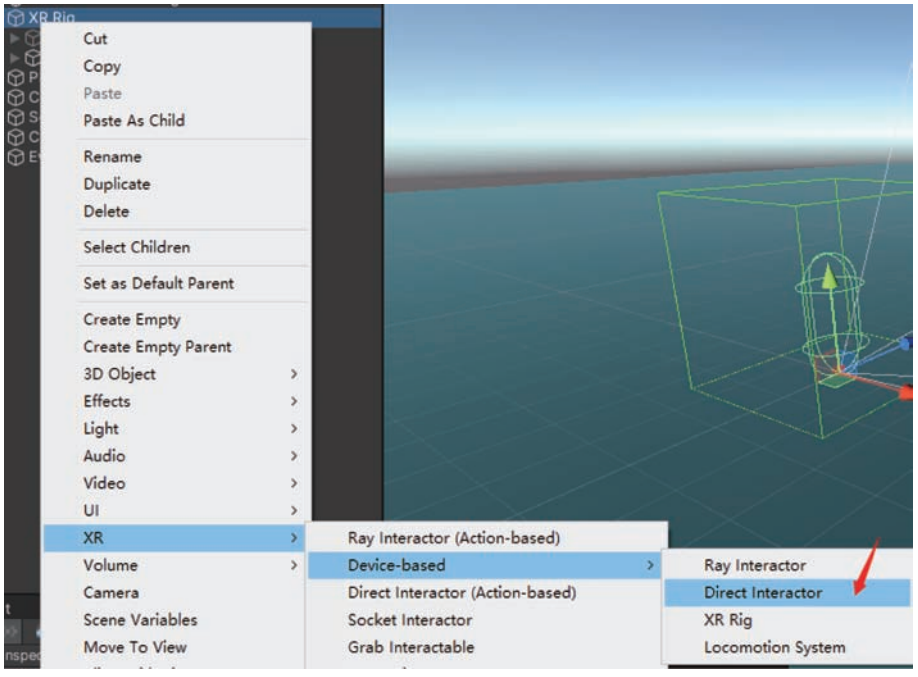

图 3-16　选择 XR→Device-based→Direct Interactor 选项

设定 VRLeft 游戏对象的 XR Controller 组件的 Controller Node 为 Left Hand，如图 3-17 所示。

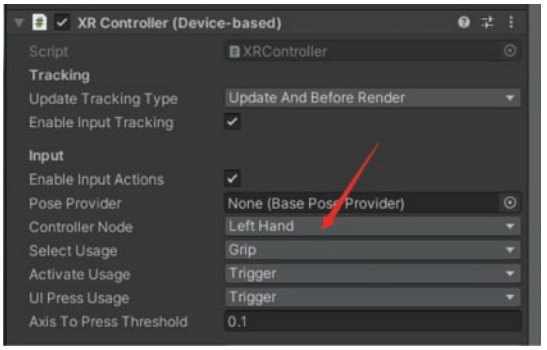

图 3-17 XR Controller 组件

设定 VRLeft 游戏对象的 XR Direct Interactor 组件的 Interaction Layer Mask 为 Nothing，如图 3-18 所示。

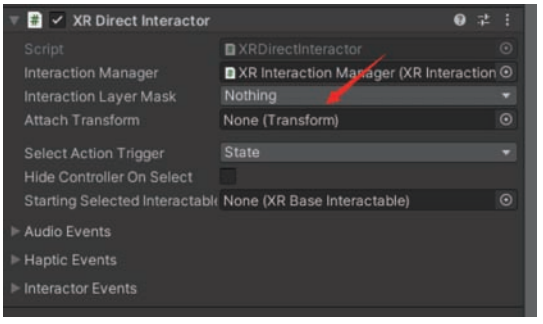

图 3-18 XR Direct Interactor 组件

在 Hierarchy 视图中选择 VRLeft 游戏对象，按 Ctrl＋D 组合键复制并重命名为 VRRight，如图 3-19 所示。

设定 VRRight 游戏对象的 XR Controller 组件的 Controller Node 为 Right Hand，如图 3-20 所示。

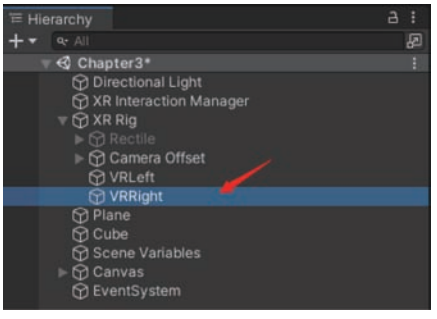

图 3-19 重命名为 VRRight

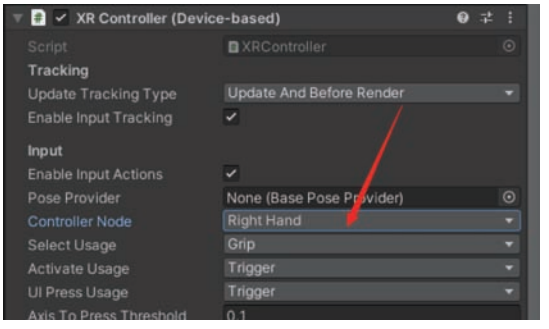

图 3-20 XR Controller 组件

在 Unity 编辑器的菜单栏中选择 Assets→Import Package→Custom Package 选项，如图 3-21 所示。

在 Import package 对话框中选择 hand. unitypackage 图标，如图 3-22 所示。hand.

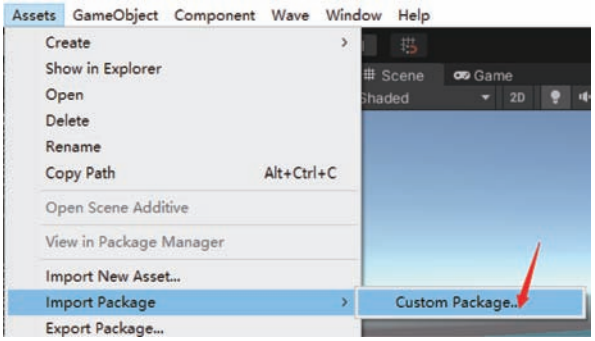

图 3-21　选择 Assets→Import Package→Custom Package 选项

unitypackage 文件中含有虚拟手的所有相关资源。

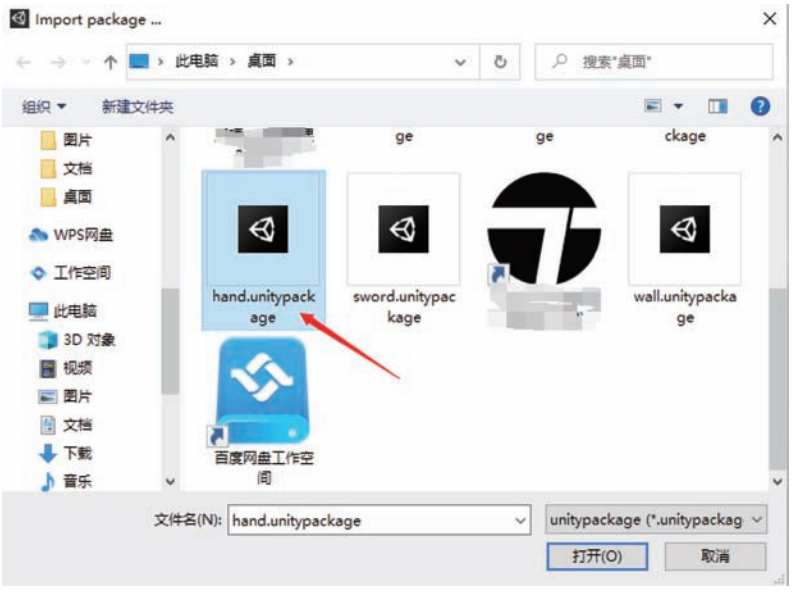

图 3-22　选择 hand. unitypackage 图标

导入 hand. unitypackage 文件后，在 Project 视图中单击 Oculus Hands 文件夹下 AnimationControler 目录下的 LefthandAC 动画控制器，如图 3-23 所示。

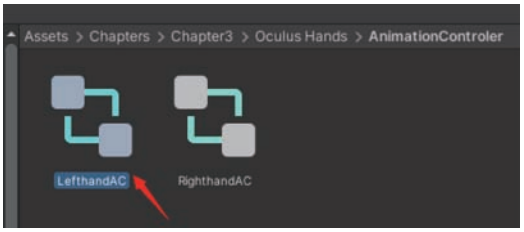

图 3-23　LefthandAC 动画控制器

LefthandAC 动画控制器含有两个浮点参数，分别为 Grip 和 Trigger，并含有一个 Blend Tree 状态，如图 3-24 所示。

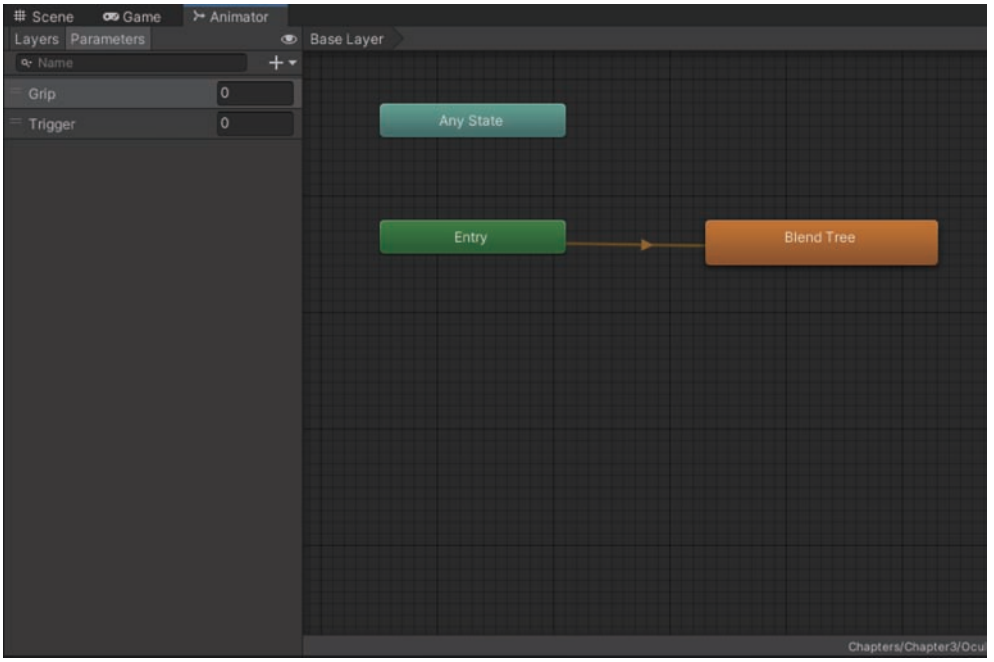

图 3-24 LefthandAC 动画控制器的参数

在 Blend Tree 中，Blend Type 为 2D Freeform Cartesian，Montion 域含有 4 个动画片段，如图 3-25 所示。动画片段 Take 001 代表手掌摊开状态，动画片段 L_hand_pinch_anim 代表拇指和食指处于捏合状态，而动画片段 l_hand_fist 代表手掌处于握拳状态，右手动画控制器 RighthandAC 也是类似的设定。

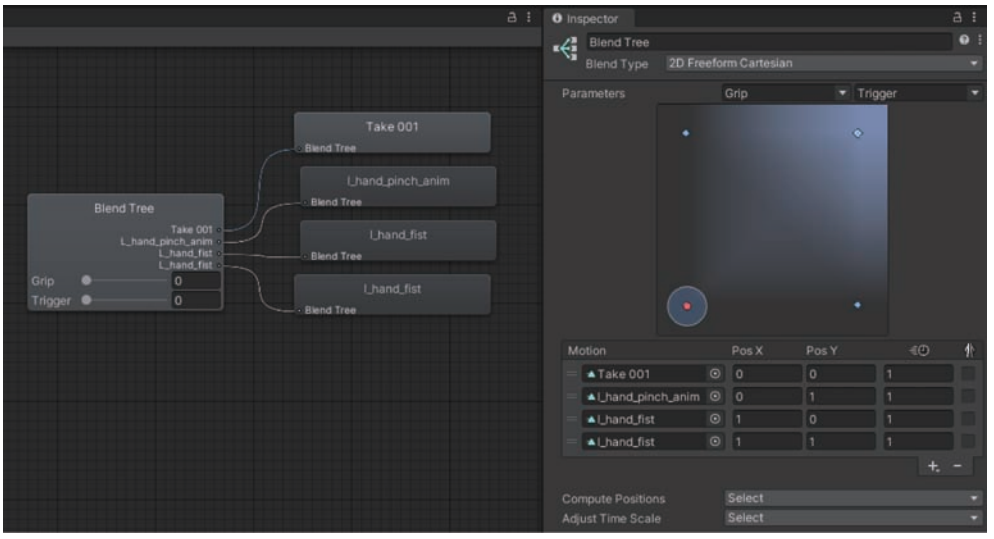

图 3-25 Blend Tree

将 Assets 中的 Oculus Hands 目录下的 Models 中的 l_hand_skeletal_lowres 放置到 VRLeft 下成为 VRLeft 的子游戏对象，将 Assets 中的 Oculus Hands 目录下的 Models 中的 r_hand_skeletal_lowres 放置到 VRRight 下成为 VRRight 的子游戏对象，放置后的结果如图 3-26 所示。

为 l_hand_skeletal_lowres 游戏对象的 Animator 组件的 Controller 指定 LefthandAC 为其动画控制器，如图 3-27 所示。

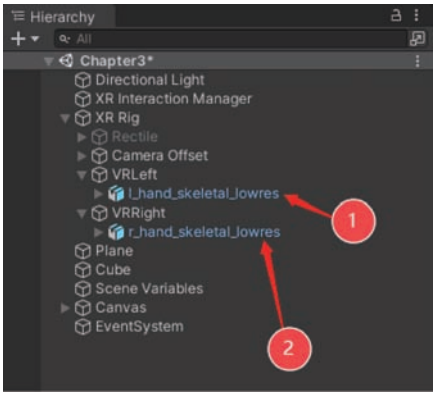

图 3-26　放置后的结果

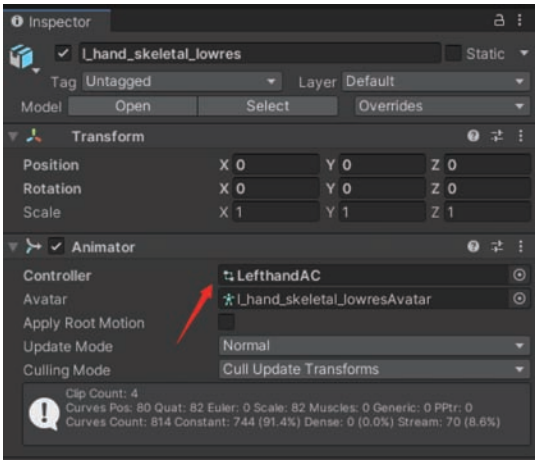

图 3-27　Animator 组件（一）

为 r_hand_skeletal_lowres 游戏对象的 Animator 组件的 Controller 指定 RighthandAC 为其动画控制器，如图 3-28 所示。

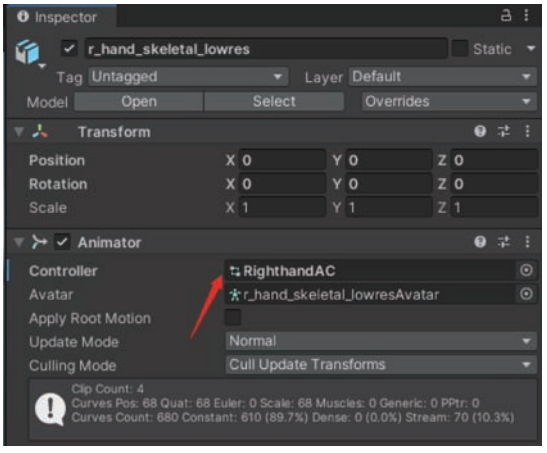

图 3-28　Animator 组件（二）

选择 Assets 目录下的 Oculus Hands 目录下的 Materials 目录下的 Hands_solid 材质，如图 3-29 所示。

把 Hands_solid 材质的 Shader 更改为 Wave/Essence/Hand/Model，如图 3-30 所示。

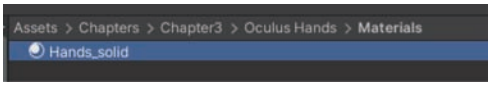

图 3-29 Hands_solid 材质

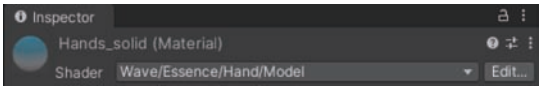

图 3-30 Hands_solid 材质的 Shader

为 r_hand_skeletal_lowres 游戏对象增加宏名为 AniRightHand 的可视化脚本组件，如图 3-31 所示。

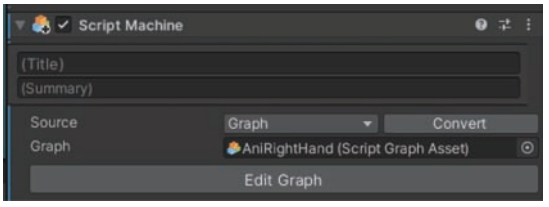

图 3-31 宏名为 AniRightHand 的可视化组件

AniRightHand 宏的可视化脚本如图 3-32 所示。在循环事件中检查右手控制器的握把键是否按下，如果按下设定动画控制器的 Grip 参数为 1，否则为 0。检查右手控制器的扳机键的按下幅度值，如果按下，则设定动画控制器的 Trigger 参数为该幅度值，否则为 0。

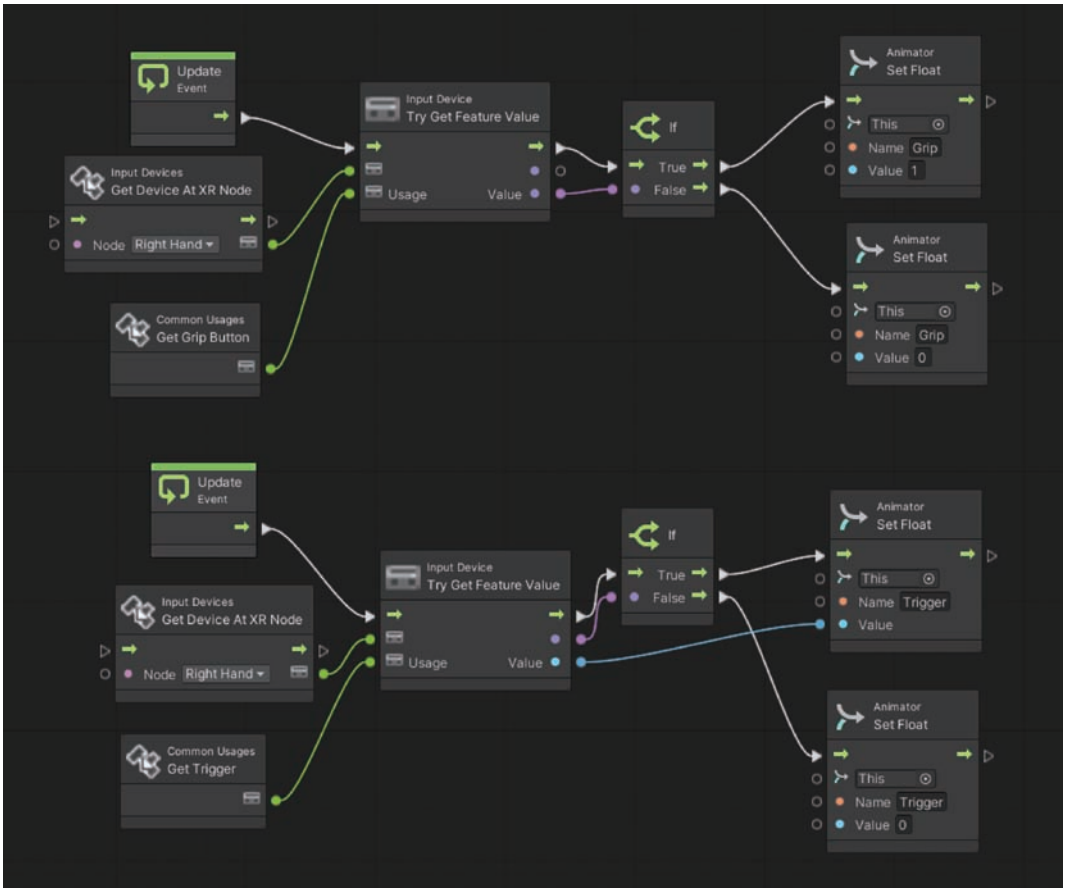

图 3-32 AniRightHand 宏的可视化脚本

为 l_hand_skeletal_lowres 游戏对象增加宏名为 AniLeftHand 的可视化脚本组件，如图 3-33 所示。

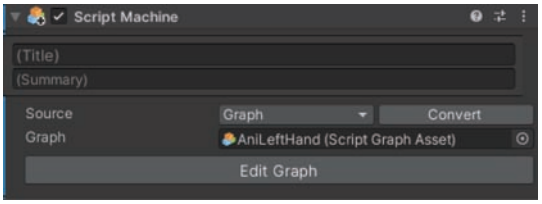

图 3-33　宏名为 AniLeftHand 的可视化组件

AniLeftHand 宏的可视化脚本如图 3-34 所示。在循环事件中检查左手控制器的握把键是否按下，如果按下设定动画控制器的 Grip 参数为 1，否则为 0。检查左手控制器的扳机键的按下幅度值，如果按下，则设定动画控制器的 Trigger 参数为该幅度值，否则为 0。

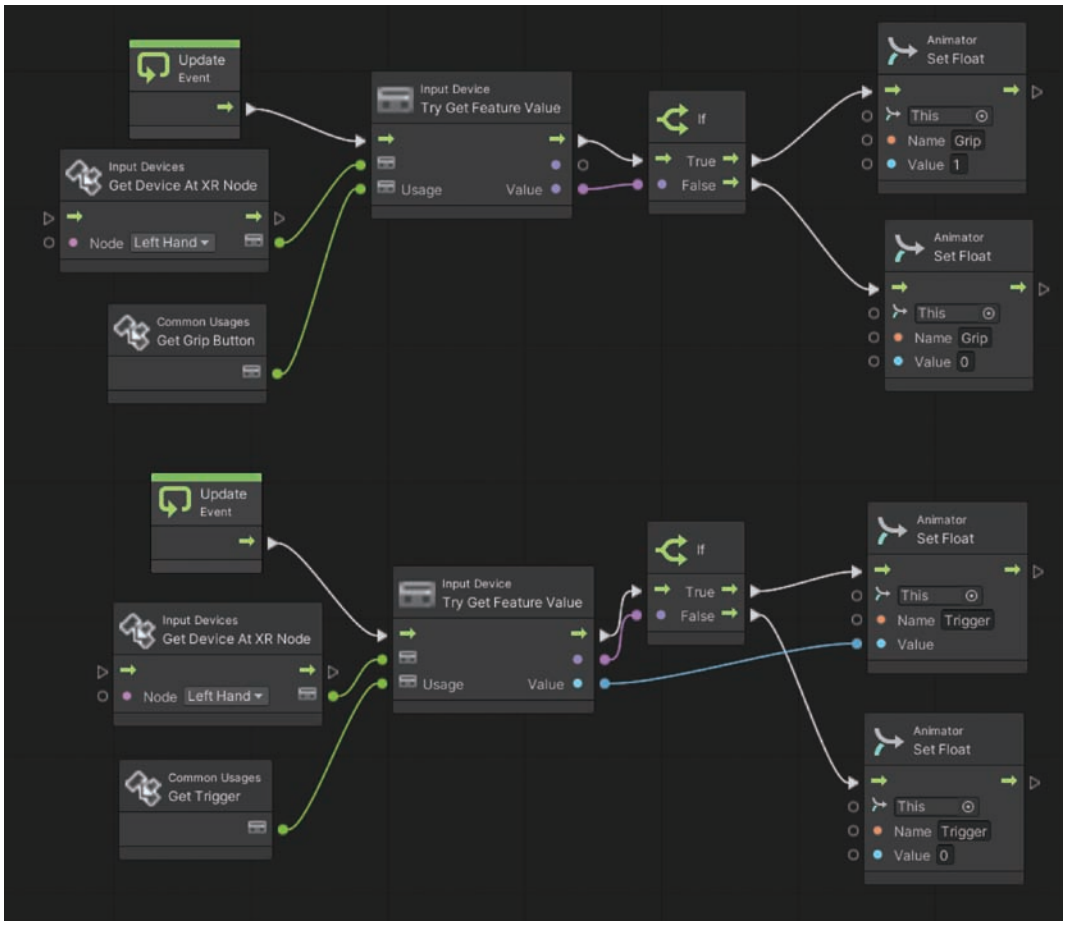

图 3-34　AniLeftHand 宏的可视化脚本