Cocos CreatorXR 使用手册

Cocos CreatorXR 介绍

Cocos CreatorXR 是基于 Cocos Creator 和 Cocos Engine 打造的一款 XR 内容创作工具。底层通过支持 OpenXR 标准协议来抹平不同 XR 设备之间的差异，可以一站式对创作内容进行开发并发布到不同的 XR 设备中而无需去适配不同设备的信号；基于底层我们封装了一系列不同功能的 XR 中间件来提供 XR 内容创作支持，并支持用户自定义扩展组件内容；上层基于 Cocos Creator 面板扩展出多种形式的 XR 功能菜单和组件样式，为用户提供更为便利的内容创作界面。

版本历史

v1.1.0

新增：

- 新增 AR 应用开发模块，可发布 AR 应用至 AREngine、ARCore、ARKit 和 Qualcomm Spaces 平台，提供自动化行为编辑组件来支持快速创建和体验 AR 内容，支持开启设备追踪（AR Camera）/平面追踪 /图像追踪 /锚点 /Mesh 等 AR 特性。
- 新增非缓冲式手柄控制器震动反馈，在 cc.InteractorEvent 的 Haptic Event 中选择想要开启震动的事件类型并调整震动参数。
- XRUI 一键转换功能，传统 2D UI 可以一键转化为的带有空间属性的 XR UI。
- 凝视交互器，根据头戴设备的注视中心位置进行交互行为。
- XR 视频播放器，针对 XR 设备优化了视频渲染管线并支持切换展示窗口、180 度、360 度多风格的视频。可以满足用户在 3D 场景中浏览全景视频或动态材质的需要。
- XR 内容预览新增无线串流方式，在 Web 浏览器中预览 XR 项目并同步所有来自 XR 设备

的信号，无需打包应用至设备即可快速完整地体验所有 XR 项目内容。

- 屏幕手势交互器，使用屏幕手势操作 AR 虚拟对象。
- 新增 Rokid Air 设备支持手机作为 3DOF 空鼠的功能。

修复：

- Huawei VR Glass（6Dof 套装）对 6Dof 输入的完整支持；
- Huawei VR Glass 配套的 6Dof 手柄在触摸摇杆时也产生随机信号的处理问题；

v1.0.1

- 支持发布 XR 应用至 Rokid Air、HuaweiVR、Meta Quest/Quest2、Pico Neo3/Pico4、Monado 五类设备。
- 提供设备映射、交互、虚拟移动、XR UI 模块化组件支持 XR 内容创作。
- 支持使用 Web 浏览器预览 XR 内容，可用键鼠操作模拟头显和手柄控制器设备。

架构

插件架构是指插件的系统结构。作为 Cocos CreatorXR 的用户，您可以通过了解插件架构的组成元素、它们的职责以及它们如何相互组合联系，来快速完整的构建 XR 应用。

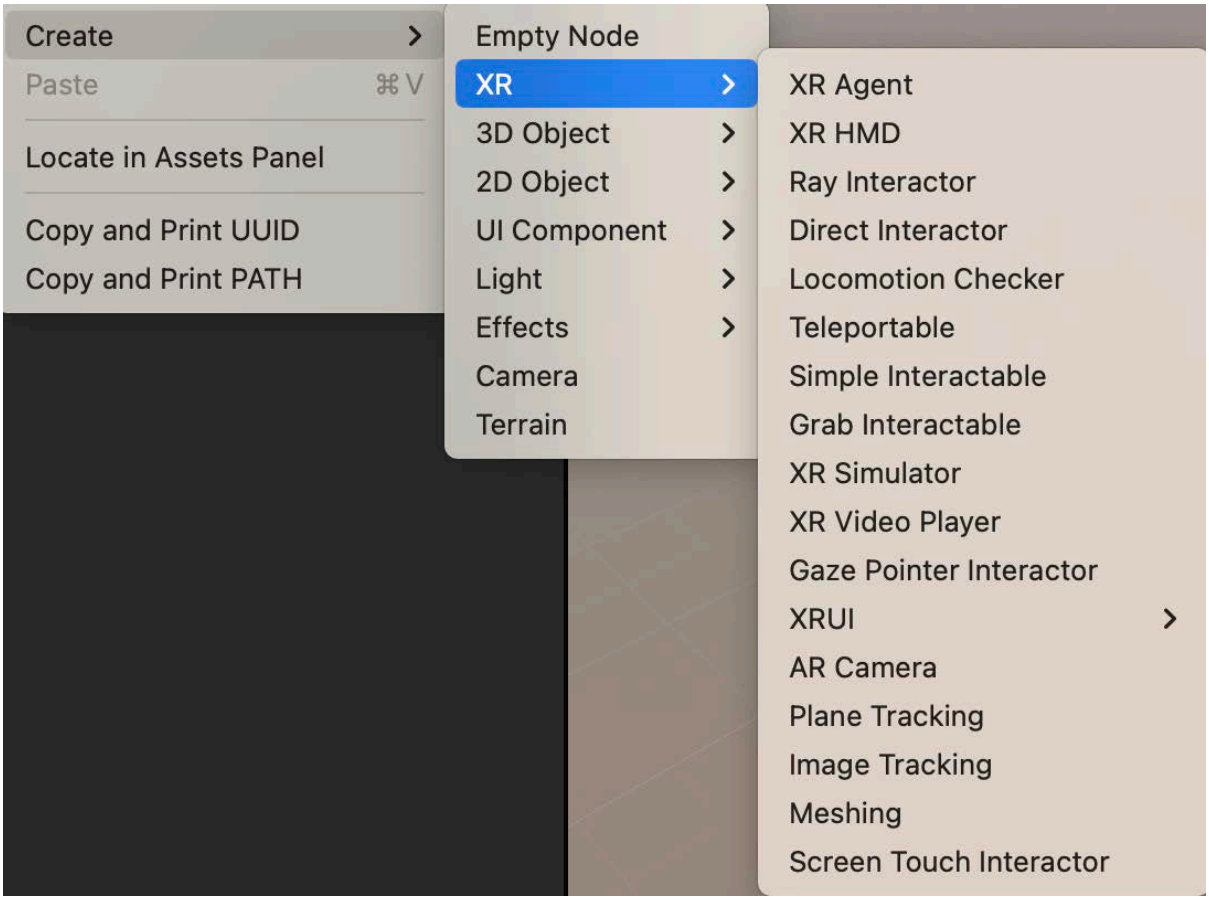
内容

内置资源与预制体

在 Cocos 扩展管理器中 开启 XR 扩展 之后就可以允许在编辑器中使用传统创建对象的方式创建 XR 对象。

在 **层级管理器** 右键选择 **创建 -> XR**，右侧会出现当前可以创建的所有 XR 预制体。

选择想要实例化生成的对象即可在场景中创建出来。



内置预制体

名称	说明	包含组件
XR Agent	现实世界主角相关的信息在虚拟场景中的代理节点，同时具有用于控制虚拟世界中 XR 主角的生命周期的功能。	TrackingOrigin
XR HMD	头戴显示器设备在虚拟世界中的抽象节点，基于 Camera 对象进行改造生成，用于同步现实世界中头戴显示器的输入信号并将引擎渲染结果输出至设备。	Camera AudioSource HMDCtrl PoseTracker TargetEye

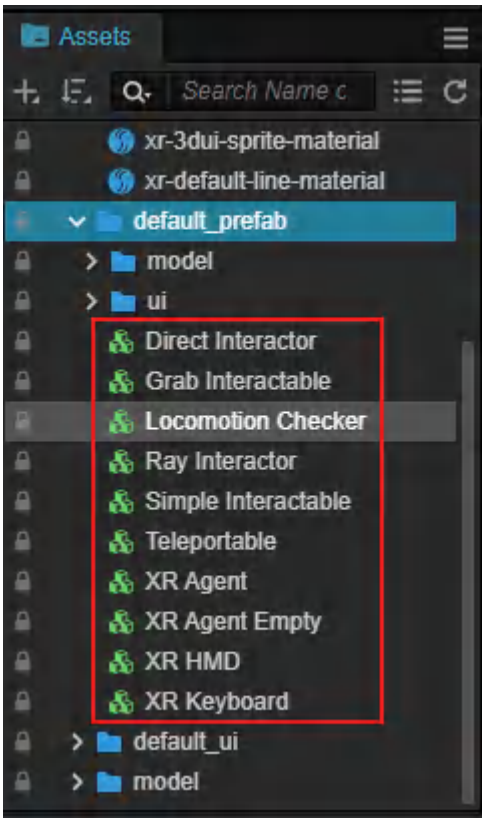
AR Camera	抽象表式移动端设备带有 AR 能力的摄像机，用于来映射物理设备的摄像头 AR 功能。	Camera PoseTracker ARCameraMgr
Ray Interactor	用于进行远距离交互的射线交互器，包含对 XR 设备手柄控制器的 I/O 映射以及射线交互功能。	PoseTracker XRController RayInteractor Line
Direct Interactor	用于进行近距离直接交互的交互器，同时也包含了对 XR 设备手柄控制器的 I/O 映射以及交互功能	PoseTracker XRController DirectInteractor
Gaze Pointer Interactor	用于进行凝视点交互的交互器，跟随头动，按凝视时间来触发交互行为	UITransform RenderRoot2D XRGazeInteractor
Screen Touch Interactor	适用于手持移动端设备的屏幕手势交互起，将屏幕手势转化为交互行为同场景中的对象进行交互。	ScreenTouchInteractor
Locomotion Checker	运动检查器，充当所有虚拟运动驱动访问 XR Agent 的仲裁者，可以保证固定时间内对唯一的运动状态的维持。	LocomotionChecker
Teleportable	支持与交互器发生传送交互行为的交互物，可以传送 XR Agent 到此对象相关的一个位置。	Teleportable InteractableEvents
Simple Interactable	简易的交互物对象，用户可以在此对象上自定义扩	InteractableEvents

	展任意的交互行为	
Grab Interactable	支持与交互器发生抓取行为的交互物。	RigidBody GrabInteractable InteractableEvents
XR Simulator	用于预览 XR 内容，提供 Web 端、无线串流两种方式。	XRInteractiveSimulator
XR Video Player	XR 视频播放器，支持在空间中播放窗口化、180 度、360 度模式的视频。	XRVideoPlayer XRVideoController XRVideoCaption
XRUI	可在空间中渲染和交互的 3D UI。	RaycastChecker RenderRoot2D BoxCollider
Plane Tracking	为应用赋能平面识别能力，在运行时使用设备 AR 能力识别出物理世界中的平面特征数据，并可以将这些平面数据可视化显示在应用程序中。	ARPlaneTracking
Image Tracking	为应用赋能图像识别能力，在运行时使用设备 AR 能力识别出 2D 图像资源。	ARImageTracking
Meshing	为应用赋能环境重构能力，根据现实环境创建 3D 网格。	ARMeshing

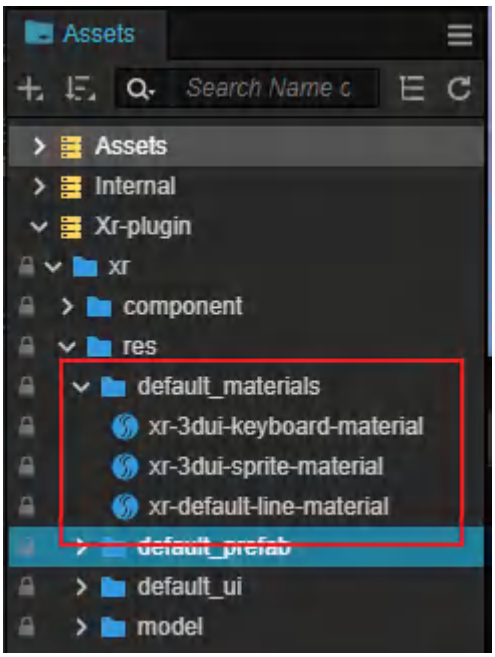
内置资源

开启 XR 的扩展后，在内置资源数据库（xr-plugin）中会新增 XR 预制体、材质和模型等资源，可供用户直接使用。具体位置如下图所示。

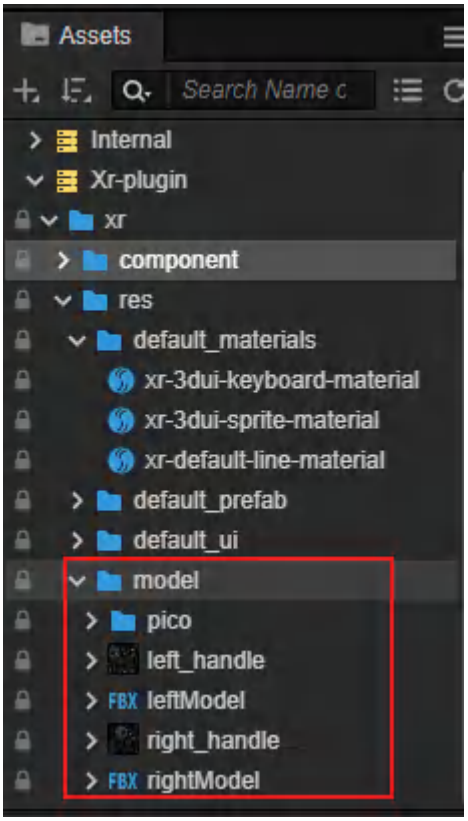
- 预制体资源



- 材质资源



- 模型资源



XR 组件

Cocos CreatorXR 通过组件的组合封装为实体赋能，实体根据其不同特性又被不同的功能系统所管理。所以编辑器中所有 XR 相关的功能底层都是由封装好的特殊 XR 组件驱动的。

Cocos CreatorXR 的功能组件主要由 5 部分构成：

- 设备映射
- 交互组件
- 交互限制组件
- 虚拟移动组件
- XR UI

开启了 xr-plugin 扩展之后，想要给场景中的对象添加 XR 相关的功能组件可以在 **属性检查器** 中点击 **添加组件** 按钮，在出现的组件列表中找到 **XR** 分类，选择 XR 分类下的想要添加的 XR 组件类别再找到类别下的对应组件即可。

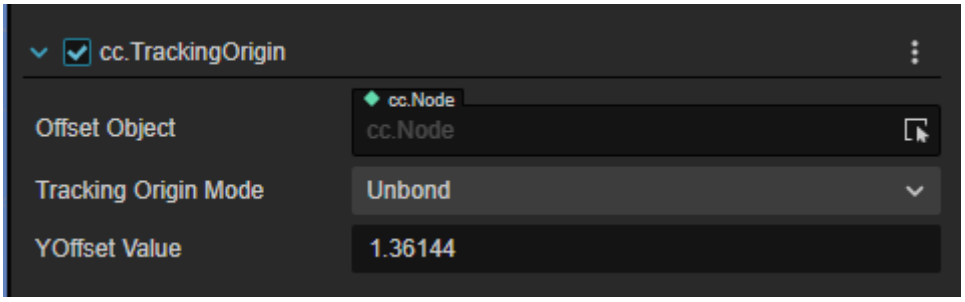
设备映射组件

此类组件主要用以同步现实世界中物理设备和虚拟世界中的代理节点之间的 I/O 信息。确保在用户在 XR 设备的使用和虚拟世界中的反馈一致。

主要包括以下组件：

TrackingOrigin

追踪原点代理组件。

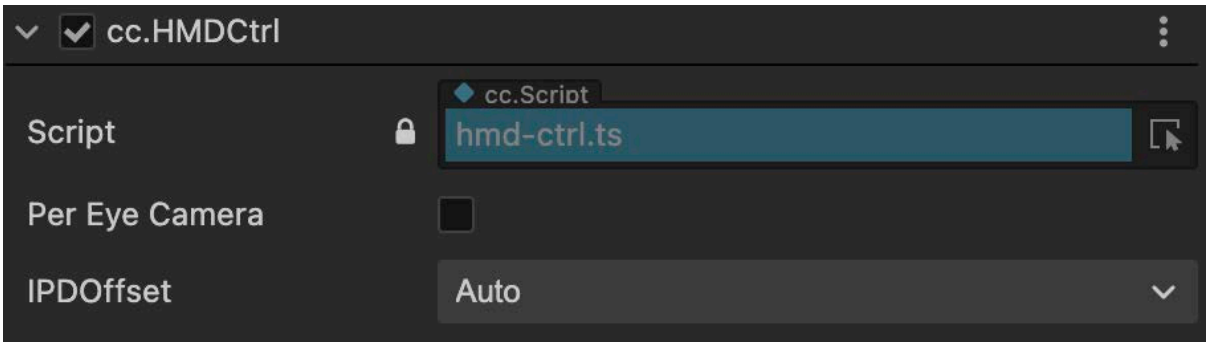


属性	说明
Offset Object	指定需要竖直偏移的对象，如果选择的对象还有子对象，则偏移的效果是选中对象及其所有子对象进行纵向偏移。
Tracking Origin Mode	追踪的偏移方式。选择 Unbond 和 Device 时，下方出现 YOffsetValue，可手动输入数据；选择 Floor 时 YoffsetValue 隐藏。如果选择为 Floor，且设备开启了安全边界，则以设备离地面高度作为当前视角高度（暂时只支持 quest2 ）；如果选择为 Device，则偏移的高度为输入的高度。
YOffset Value	设备数值偏移量。手动输入偏移的值，米为单位，默认为 1.36144m。如果为固定值，则 OffsetObject 选中

	的对象 Transform 属性的 Y 值为当前填入的值。

HMDCtrl

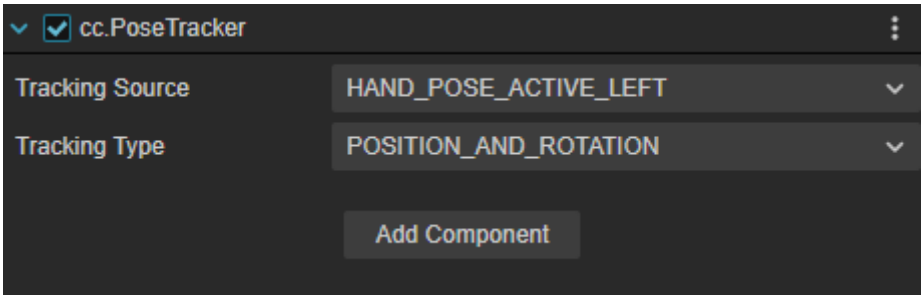
HMD（Head Mounted Display）头戴显示设备控制器，可以认为所有具备双目立体绘图能力的 XR 眼镜设备都属于 HMD 的大范畴，因此此组件定义了 XR 眼镜的图像渲染输出相关参数。



属性	说明
Per Eye Camera	开启单眼显示功能；勾选 PerEyeCamera 后下方出现 Sync With Main Camera 选项；XR HMD 下的两个子节点 LeftEye 和 RightEye 由隐藏变为显示
IPDOffset	调整瞳间距。下拉列表中选择 Manual 时下方出现 OffsetValue 输入框；在 PerEyeCamera 开启的情况下，调整 Manual 的参数可让 LeftEye 和 RightEye 的 TransForm 属性 X 值变化（变化的值为 ±IPDOffset/2）

PoseTracker

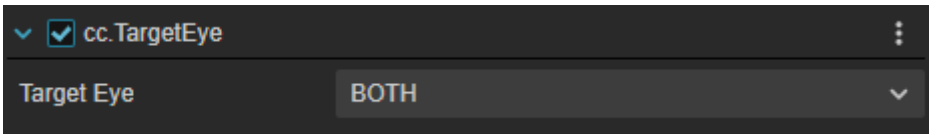
位姿追踪组件。



属性	说明
Tracking Source	选择需要追踪的设备源
Tracking Type	追踪模式。选择 POSITION_AND_ROTATION 时追踪设备的平移 + 旋转；选择 POSITION 时只追踪平移的量；选择 ROTATION 时只追踪旋转的量。

TargetEye

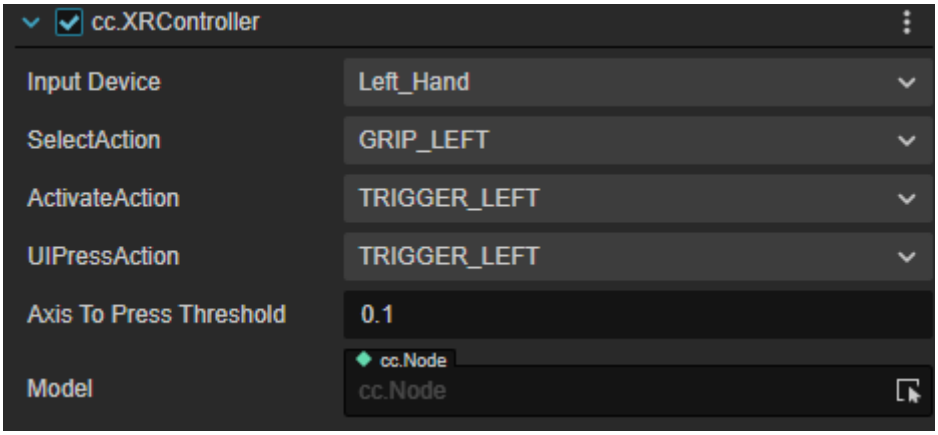
指定接受渲染相机的组件。



属性	说明
Target Eye	指定渲染的目镜。Both 为左右眼都显示，left 为左眼，right 为右眼。

XRController

控制器抽象组件。



属性	说明
InputDevice	绑定输入手柄设备。
SelectAction	将 Select 行为映射到手柄的对应实体按键
ActivateAction	将 Activate 行为映射到手柄的对应实体按键
UIPressAction	将交互 UI 行为映射到手柄的对应实体按键
AxisToPressThreshold	行为触发时的阈值
Model	用于指定手柄的模型对象。

ARCameraMgr

移动端手持设备摄像头 AR 属性

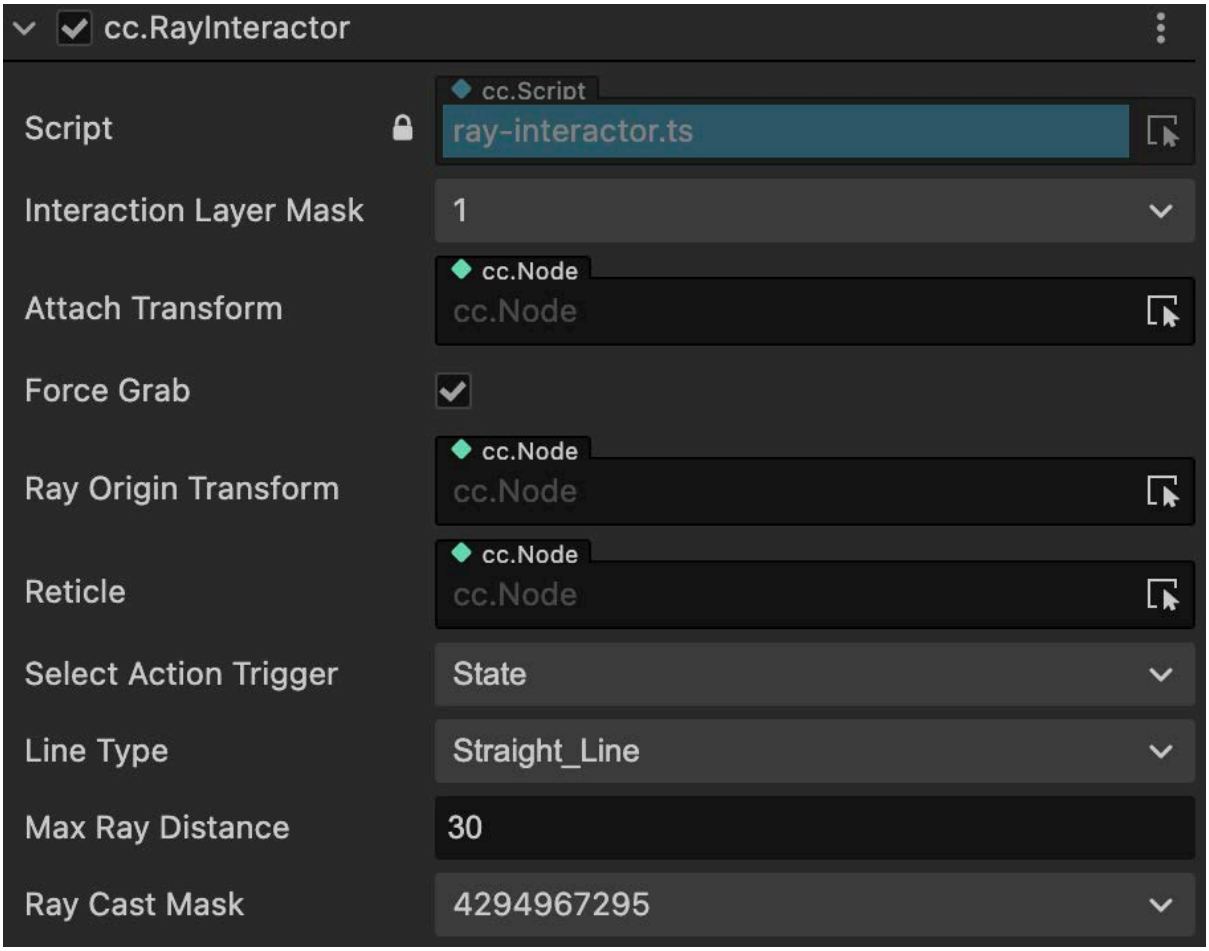
属性	说明
Auto Focus	开启或关闭相机自动对焦功能。关闭时，使用固定对焦模式。自动对焦功能是否可用取决于设备相机。
Light Estimate（实验性）	开启后，运行时估计环境光各项属性并实时调整场景光照，让虚拟物体具有与真实场景相同的光照效果（光照一致性）。

交互组件

一次交互操作需要有两种对象协调完成：交互主体和被交互物，对应的交互组件由此也分为两类：Interactor（交互器）和 Interactable（可交互对象）。

RayInteractor

射线交互器组件。



属性	说明
AttachTransform	用此 AttachTransform 的位置作为抓取的物体最终落到的位置，如果为空就用当前 Interactor 的位置
ForceGrab	远距离抓起；开启时被抓对象吸附到

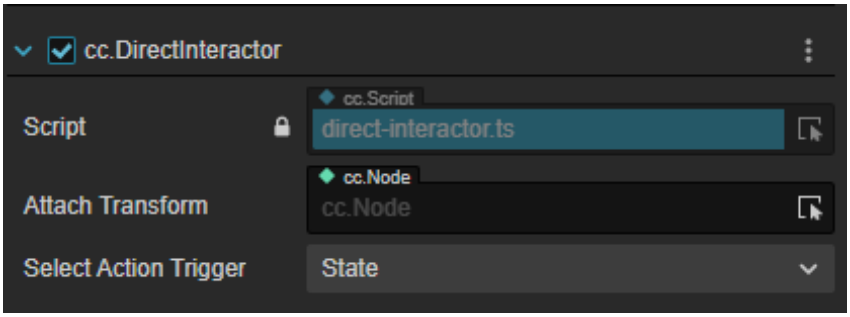
	Attach Transform、关闭后抓取挂载在交互点的位置
RayOriginTransform	可以改变发出 Ray 的起始位置，为空就默认是当前 Interactor 的位置
LineType	改变射线检测和射线样式；StraightLine 是直线；Projectile Line 是抛物线；Bezier Line 是贝塞尔曲线
MaxRayDistance	射线交互可以触发的最远距离
ReferenceNode	LineType 为 ProjectileLine 和 BezierLine 时出现此项。用曲线的参考系来定义地平面和向上向量。如果在启动时没有设置，它将尝试找到 XR Agent，如果没有引用，它将默认使用全局的上向量和原点。
Velocity	LineType 为 ProjectileLine 时出现此项。初始速度。增加这个值将使曲线延伸的更远。
Acceleration	LineType 为 ProjectileLine 时出现此项。重力加速度。
Additional Ground Height	LineType 为 ProjectileLine 时出现此项。在地平面以下的额外高度，射线超过地平线会继续向下投射。增加这个值将使终点的位置高度下降。
Additional Flight Time	LineType 为 ProjectileLine 时出现此项。在落地后的额外飞行时间，射线超过地平线会继续向下投射。增加这个值将使投射的飞行时间延长。
End Point Distance	LineType 为 BezierLine 时出现此项。增加这个值的距离会使曲线的末

	端离起始点更远。
End Point Height	LineType 为 BezierLine 时出现此项。降低这个值将使曲线的末端相对于起点下降得更低。
Control Point Distance	LineType 为 BezierLine 时出现此项。增加这个值将使曲线的峰值离起点更远。
Control Point Height	LineType 为 BezierLine 时出现此项。增加这个值将使曲线的峰值相对于起点更高。
Sample Frequency	LineType 为 ProjectileLine 和 BezierLine 时出现此项。用于近似曲线路径的采样点的数目。数目越多近似效果越好，但性能越低。
RaycastMask	只能和此 Layer 类型的交互物发生交互
SelectActionTrigger	Select 行为触发机制，详情见交互功能介绍

注：抛物线和贝塞尔曲线的功能需要**扩展版本>=v1.1.0，编辑器版本>=3.7.1**。

DirectInteractor

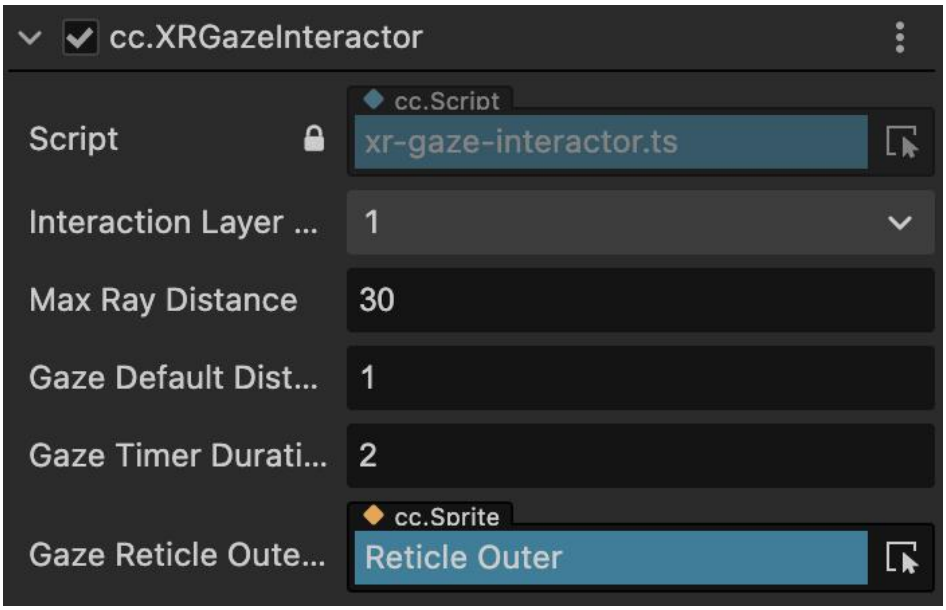
直接交互器组件。



属性	说明

AttachTransform	用此 AttachTransform 的位置作为抓取的物体最终落到的位置，如果为空就用当前 Interactor 的位置
SelectActionTrigger	Select 行为触发机制，详情见交互功能介绍

XRGazeInteractor

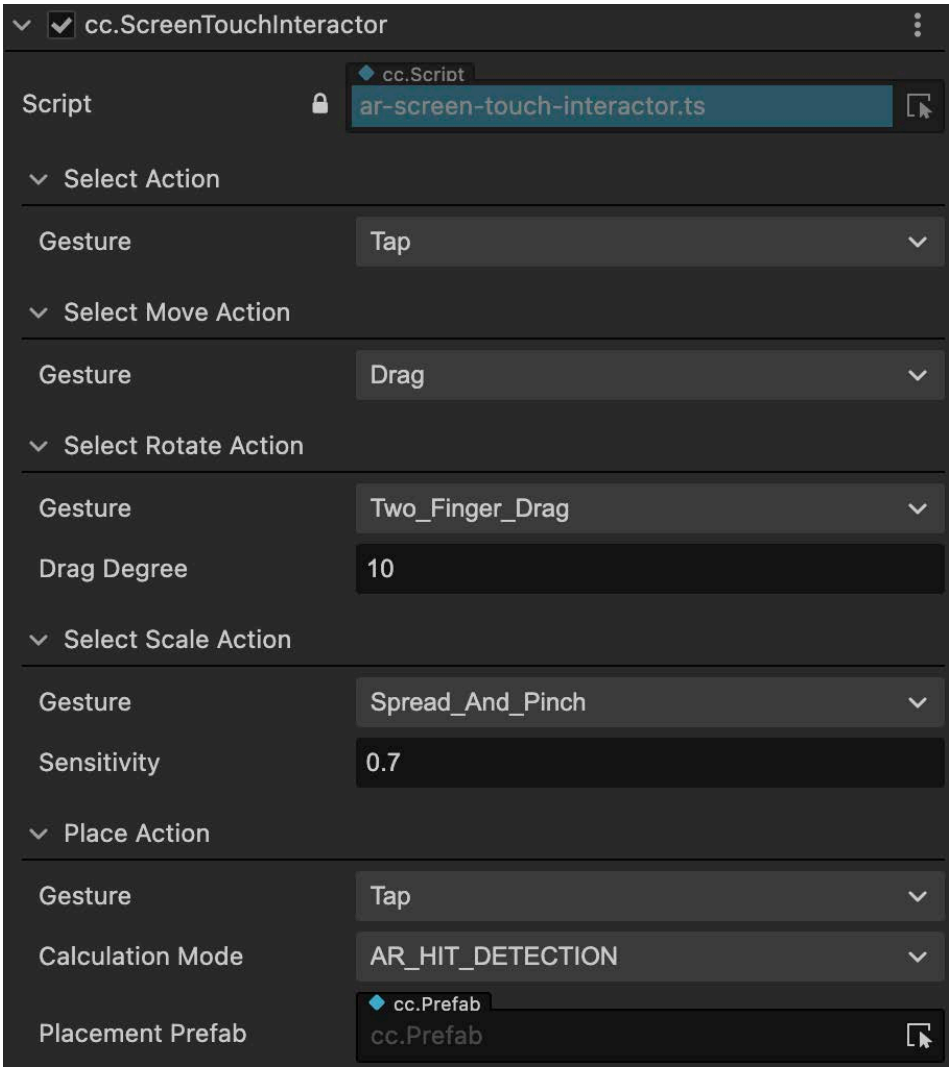


属性	说明
Gaze Pointer Offset	凝视点竖直方向偏移（单位米）
Max Ray Distance	射线最远投射距离（单位米）
Gaze Default Distance	凝视点 UI 默认距离（单位米）
Gaze Timer Duration	凝视交互触发时间（单位秒）
Gaze Reticle Outer Ring	凝视点外环 UI

注：此功能需要**扩展版本>=v1.1.0，编辑器版本>=3.7.1**。

ScreenTouchInteractor

屏幕手势交互器

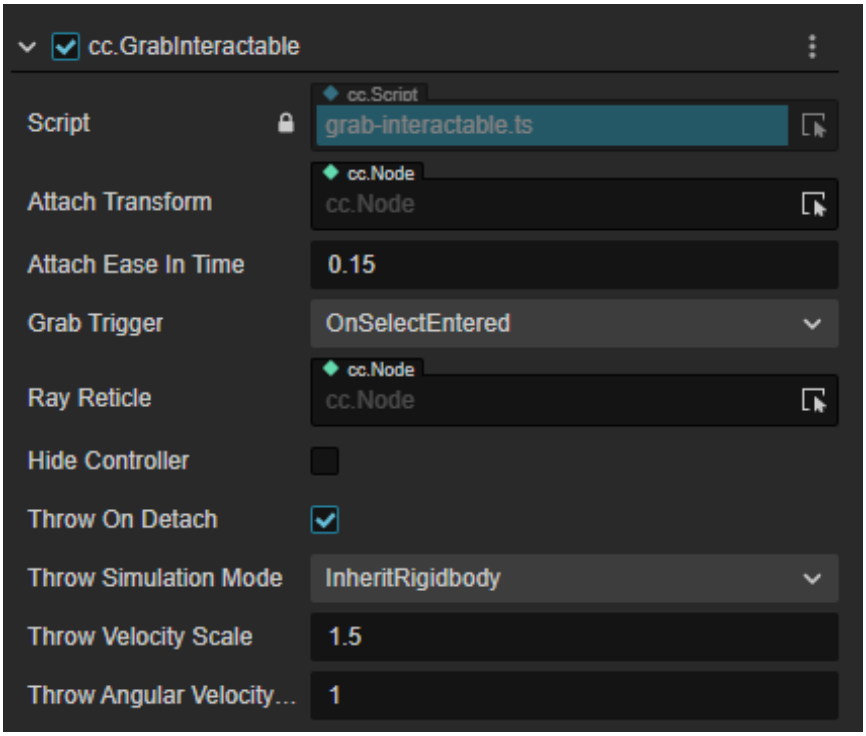


行为	属性	说明
Select Action		选择行为相关配置
	Gesture	允许用户操作虚拟物体的可选收拾类型。
	Double Tap Gap	Gesture选择为DoubleTap时出现此项，当两次点击的时间间隔小于此值时，判定为双击
	Hold Touch Duration	Gesture选择为HoldTouch时出现此项，当触碰屏幕时间大于此值时，判定为长按
Move Action		移动行为相关配置
	Gesture	绑定移动行为的手势
Rotate Action		旋转行为相关配置

	Gesture	绑定旋转行为的手势
	Drag Degree	Gesture选择为2FingersDrag时出现此项，双指拖动速率
	Twist Degree	Gesture选择为2FingersRotate时出现此项，双指旋转速率
Scale Action		放缩行为相关配置
	Gesture	绑定放缩行为的手势
	Sensitivity	放缩的灵敏度
Place Action		放置行为相关配置
	Gesture	绑定放置行为的手势
	Calculation Mode	当将物体放置在 AR 表面时，用于计算命中点位置的方法。
	Placement Prefab	引用挂载Placeable组件的预置体

GrabInteractable

可抓取交互对象组件。

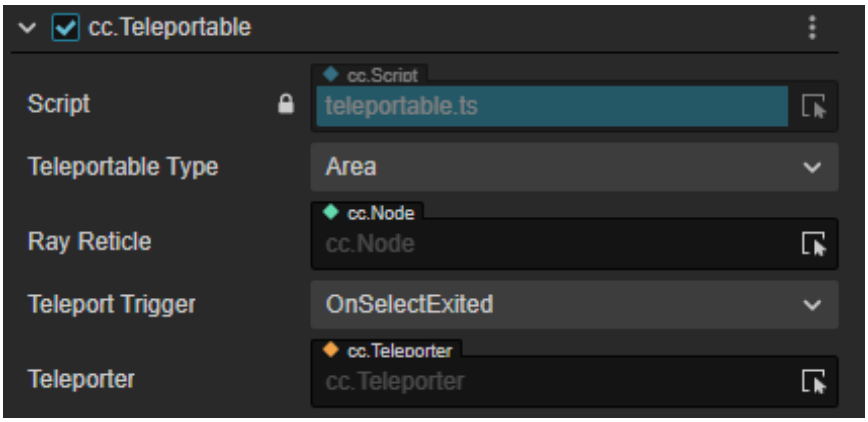


属性	说明
AttachTransform	用此 AttachTransform 的位置作为触碰点的位置，如果为空就用当前 node 的位置
AttachEaseInTime	对象被抓取到 AttachTransform 位置过程的时间（在该时间内，被抓取对象会有一个拖尾的效果，持续时间越久效果越不明显）
GrabTrigger	两种方式用于触发抓取：当 Select 的行为触发时/当 Activate 行为触发时
RayReticle	当交互器与此交互物发生交互碰撞计算时，在碰撞点会显示此属性所引用的对象
HideController	开启时，此物体被抓取后会隐藏 XR Controller 所引用的 Model
ThrowOnDetach	开启后，允许模拟抛物的动作。
ThrowSimulationMode	选择为 InheritRigidbody 时，物体抛出时继承刚体的速度；选择为 CurveComputation 时，出现下列两项 ThrowSmoothingDuration、ThrowSmoothingCurve，允许自定义抛出速度的计算
ThrowSmoothingDuration	速度计算的采样时间段。使用此值作为采样区间，用于计算物体被抛出前的速度的加权平均值，作为物体抛出时的初速度
ThrowSmoothingCurve	速度采样曲线。根据绘制的曲线进行抛出时初速度的加权平均值计算

ThrowVelocityScale	初速度的权重系数。权重越大，抛物瞬时速度所乘的系数也越大，在继承或者加权计算出的初速度基础上乘以一个系数
ThrowAngularVelocityScale	初角速度的权重系数。权重越大，抛物瞬时角速度所乘的系数也越大，在继承或者加权计算出的初始角速度基础上乘以一个系数

Teleportable

可传送对象组件。

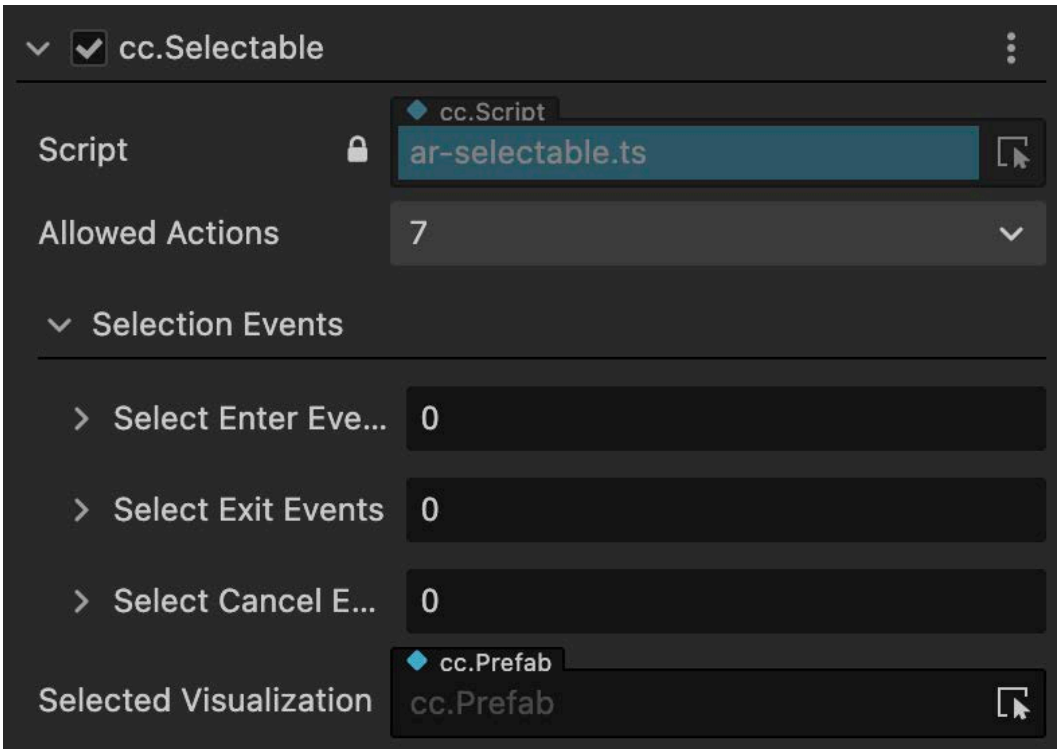


属性	说明
TeleportableType	传送点的类型。选择为 Area 时，传送到射线与传送区域交互点的位置；选择为 Anchor 时，不受射线与传送区域的交互点的限制，之间传送到区域的固定位置。
Teleport Anchor Node	TeleportableType 选择为 Anchor 时出现此项，用于标定传送的落点。此项为空时传送到传送区域的默认中心区域；此项引用了其他对象就传送到引用的对象的位置
RayReticle	当交互器与此交互物发生交互碰撞计

	算时，在碰撞点会显示此属性所引用的对象
TeleportTrigger	触发传送行为的事件：OnSelectExited 表示 Select 行为结束的时刻（按钮抬起）执行传送；OnSelectEntered 表示 Select 触发的时刻（按钮按下）
Teleporter	指定需要被传送的主体（一般为 XR Agent），主体需挂载 **Teleporter**

Selectable

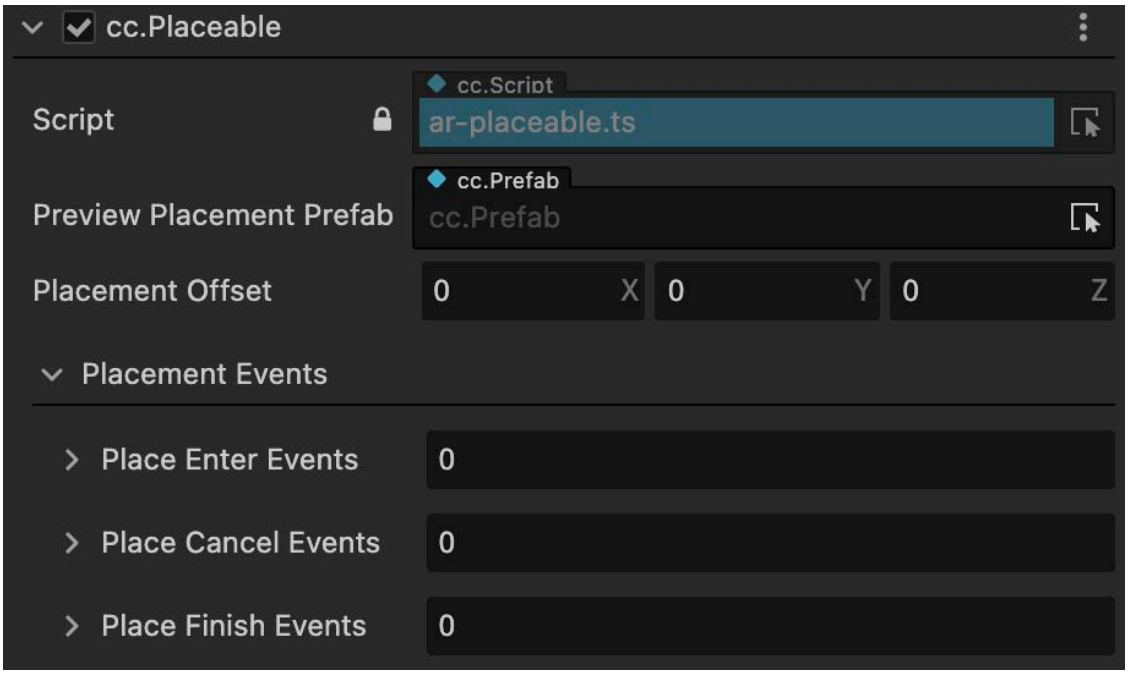
挂载此组件的对象可以被选中，且可以在选中状态下发生位移、旋转或放缩行为。



属性	说明
Allowed Actions	选择此对象后允许进行的操作。
Selection Events	选择行为的回调，它将在特定事件发生时被触发。
Selected Visualization	对象被选中时将激活的可视化效果。

Placeable

挂载此组件的对象可以使用某种方式放置于空间、AR Plane 或 AR Mesh 上。



属性	说明
Preview Placement Prefab	在放置物生成之前要放置的预放置对象。
Placement Offset	偏移由交互器放置的预制件的位置。
Placement Events	放置行为的回调，它将在特定事件发生时被触发。

事件

在项目开发过程中用户会遇到很多依赖于 XR 组件的信号来触发的逻辑。为了方便用户专注于逻辑开发而不用关心两者之间的通知关系，Cocos CreatorXR 基于传统的事件系统在一部分常被事件依赖的交互组件中封装好了一部分事件信号，用户只需要在特定的信号上绑定一系列想要触发的对象和方法即可。

事件信号

事件信号分为以下三种：

- **Hover**：指的是 XR 的输入设备的射线覆盖到物体上时触发

- **Select**：当在输入设备上按下 Select 映射的按键时触发
- **Active**：当在输入设备上按下 Active 映射的按键时触发

其交互行为描述如下：

交互行为	事件信号	说明
Hover	OnHoverEntered	Hover 行为执行的时刻
	OnHoverExited	Hover 行为退出的时刻
	OnHoverCanceled	Hover 行为取消的时刻
Select	OnSelectEntered	Select 行为执行的时刻
	OnSelectExited	Select 行为退出的时刻
	OnSelectCanceled	Select 行为取消的时刻
Active	OnActivated	激活的时刻
	OnDeactivated	取消激活的时刻

这些事件信号，可以在下列组件中选择，用于处理特定情形下的输入状态。

交互事件组件

交互组件分为：

- **InteractorEvents**：交互器事件组件。
- **InteractableEvents**：可交互对象事件组件。

其属性描述如下：

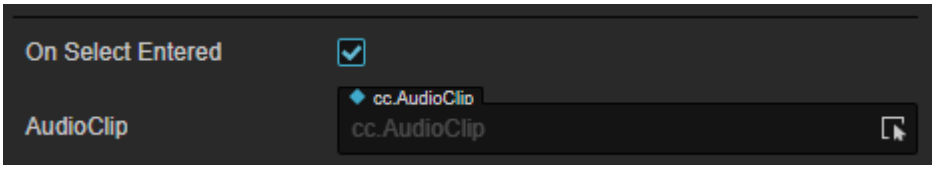
InteractorEvents

属性	说明
AudioEvents	开启后可以绑定事件触发时播放的音频
HapticEvents	开启后可以绑定事件触发时控制器的震动反馈
InteractorEvents	开启后可以绑定任意回调函数

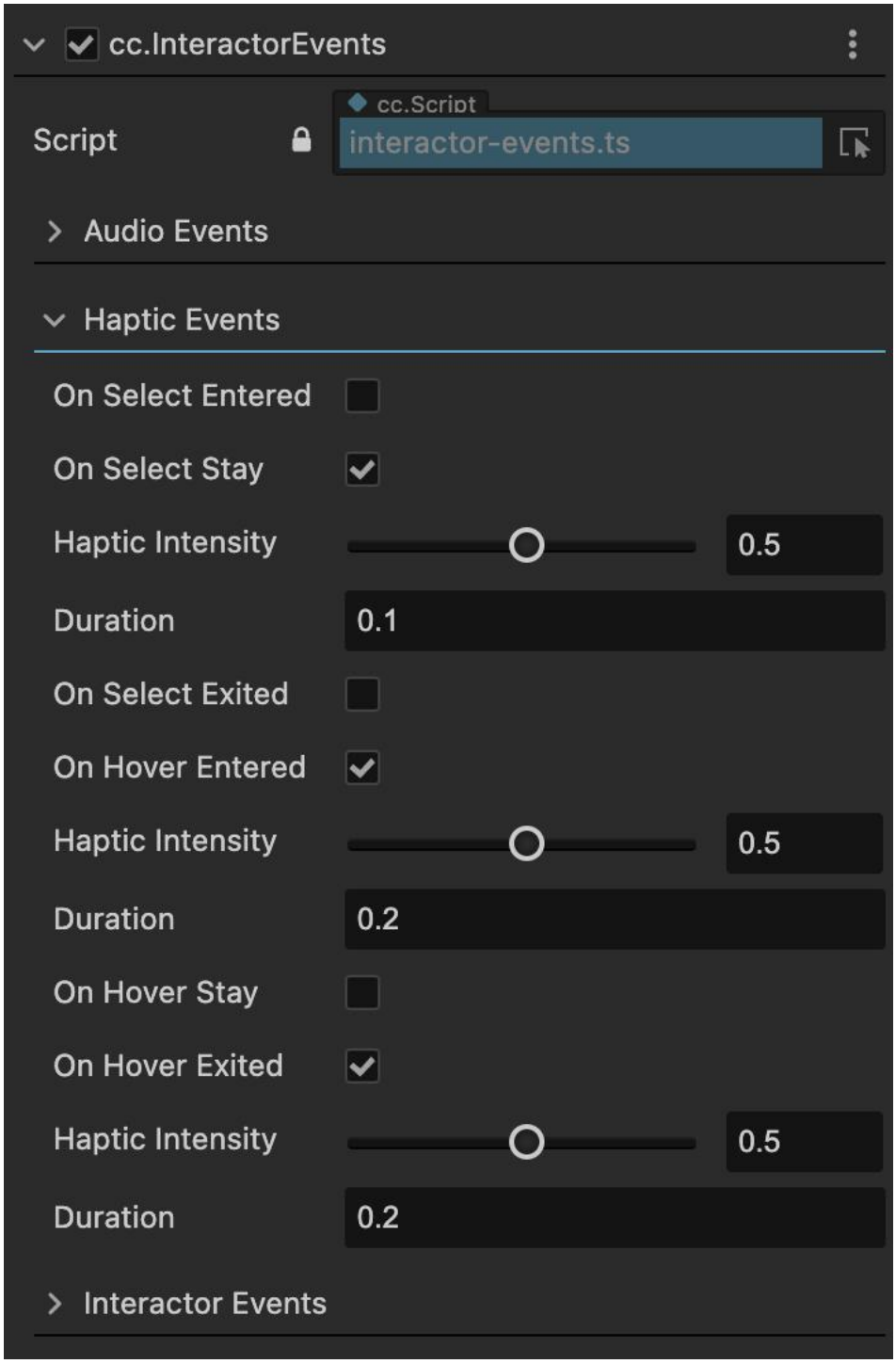
- **Audio Events：**

  Audio Events 可以根据 **事件信号** 选择不同类型的事件触发，触发后可以播放特定的音频。

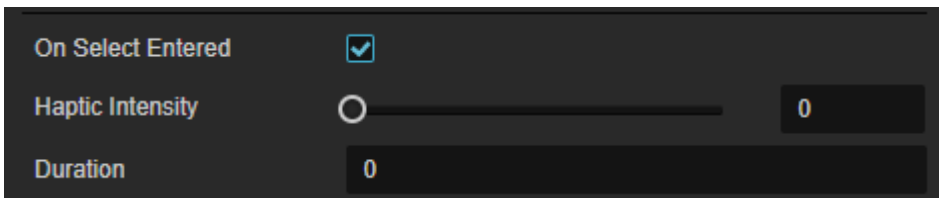
  选中 **事件信号** 右侧的单选框可启用/禁用该事件信号，启用后可以通过 **AudioClip** 属性右侧的下拉菜单选择不同的音频资源。



- **Haptic Events**（要求扩展版本>=v1.1.0，编辑器版本>=3.7.1）：



选中 **事件信号** 右侧的单选框后，可以调整控制器的震动反馈，通过震动可以给与用户更真实的触感反馈。

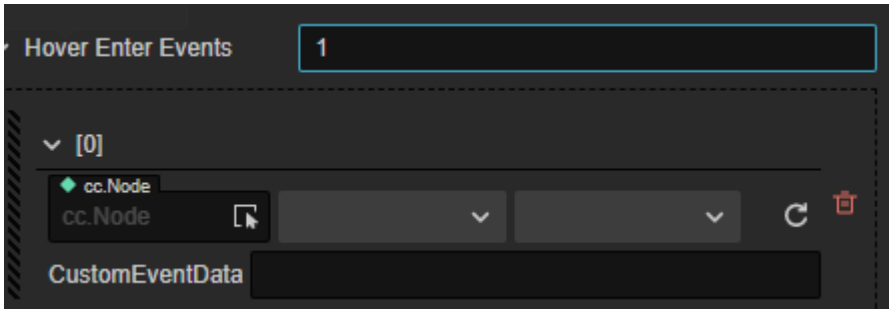


- **Haptic Intensity**：震动的灵敏度 [0,1]

- **Duration**：持续的时长

- **Interactor Events**：

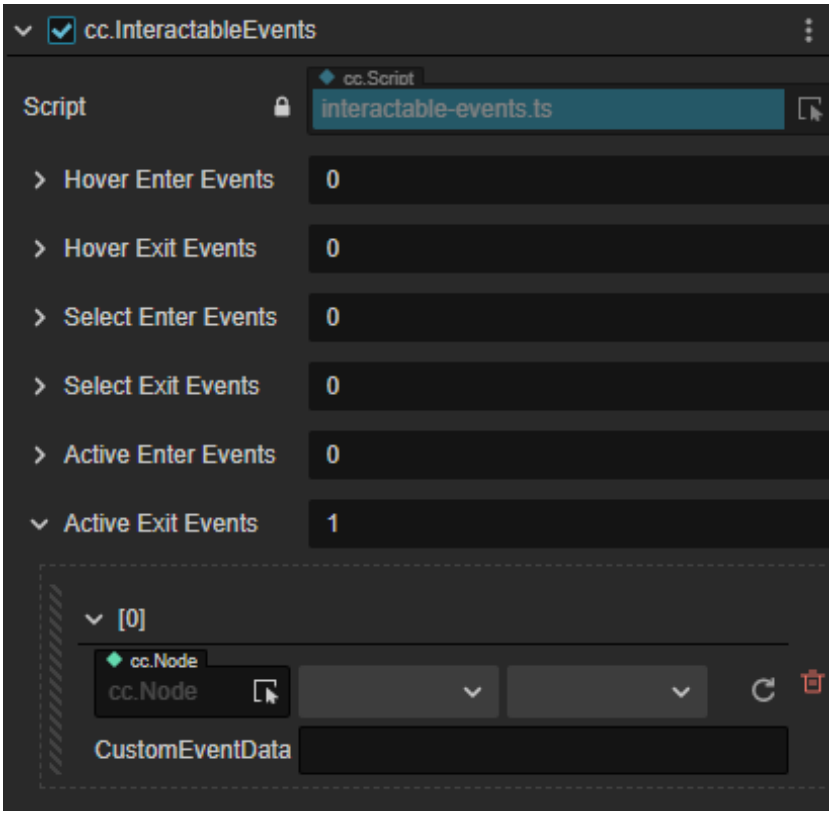
在右侧的输入框内，输入任意整数值，可在 **Interactor Events** 数组内添加元素。



- 添加完成后，可对事件回调进行配置：

- **Node**：回调接收节点
- **组件**：回调组件
- **方法**：回调方法
- **CustomEventData**：自定义事件的数据，这些数据会被当做回调方法的参数，传入到上述的回调方法中。

InteractableEvents



该组件的事件使用方式和 **InteractorEvents** 组件的 **Interactor Events** 属性类似，也可以通过在右侧输入框修改整数值的方式修改事件的数组的长度，并配置响应的回调。

交互限制组件

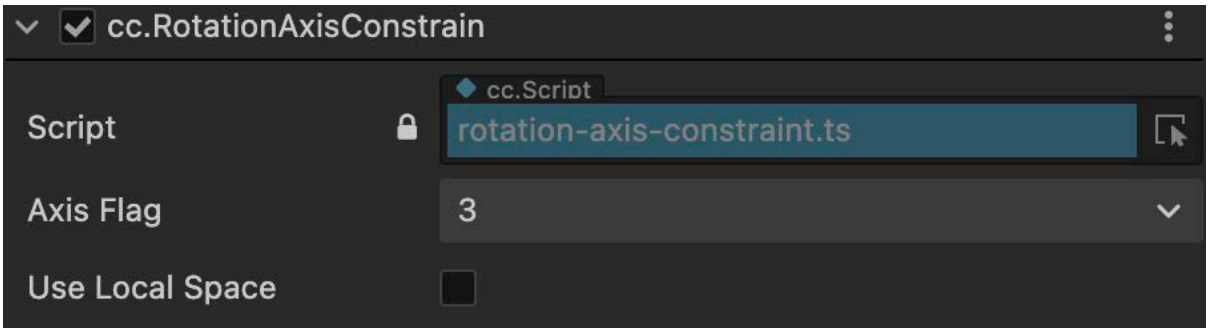
对于挂载了 Selectable 组件的交互物，可以使用交互限制组件限制他的交互效果。

MinMaxScaleConstrain

属性	说明

Min Scale	可被缩小的最小尺寸比例。
Max Scale	可被放大的最大尺寸比例。

RotationAxisConstrain



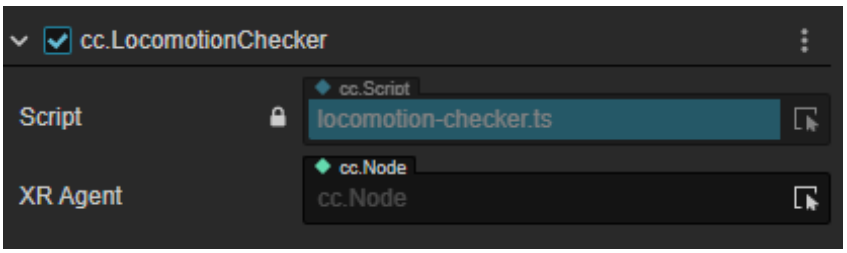
属性	说明
Axis Flag	限制交互物只能按所选旋转轴进行旋转。屏幕交互器最多只能控制双轴(x,y)旋转。
Use Local Space	是否按本地坐标轴进行旋转。默认关闭，以世界坐标轴做旋转。

虚拟移动组件

在绝大多数的 VR 项目中，用户会用第一人称的角色视角在虚拟场景中进行移动。这种移动行为一般不会依赖于用户在真实空间的移动反馈，因为位姿的追踪会受到现实世界物理空间的限制。因此我们需要一种类似传统 3D 游戏中那种利用接受控制器输入信号的方式来驱动移动行为的组件，称之为虚拟移动组件（Locomotion Component）。

LocomotionChecker

运动检查器。

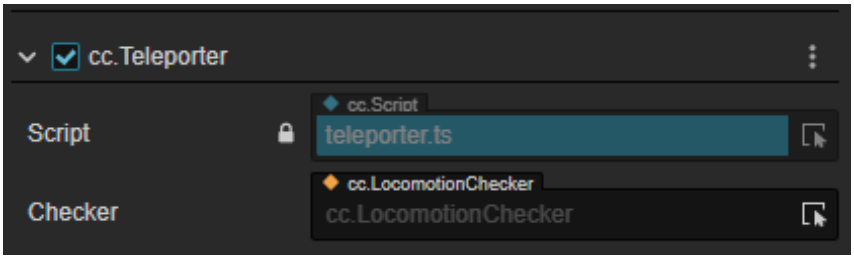


属性	说明

XR Agent	指定需要进行运动的 XR Agent（或其他对象）。添加该组件时，默认绑定遍历当前场景得到的第一个挂载了 TrackingOrigin 的节点，通常是 XR Agent；用户也可以自行指定需要进行 locomotion 操作的对象

Teleporter

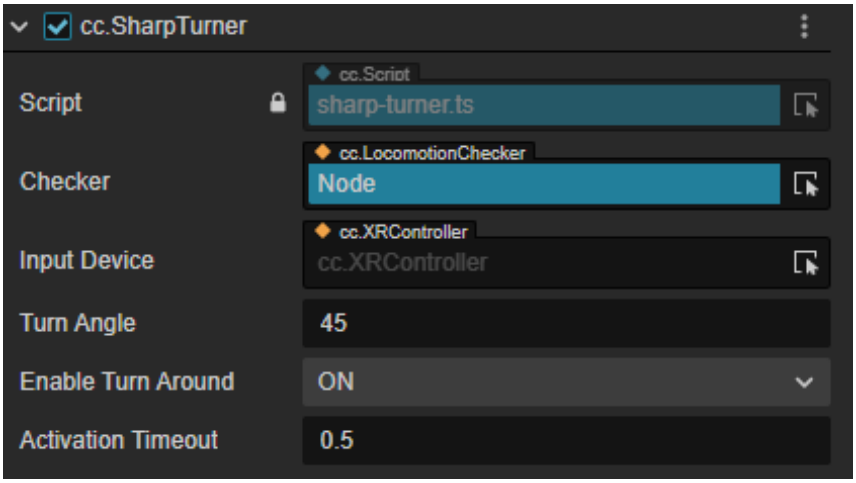
传送驱动组件。



属性	说明
Checker	添加该组件时，默认绑定遍历当前场景得到的第一个挂载 Locomotion Checker 组件的节点 用户也可以自己拖入需要指定的 Locomotion Checker

SharpTurner

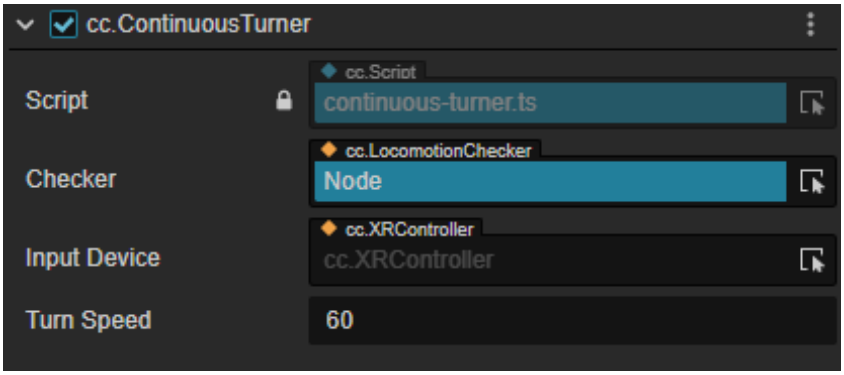
瞬间转向驱动。



属性	说明
Checker	添加该组件时，默认绑定遍历当前场景得到的第一个挂载 Locomotion Checker 组件的节点 用户也可以自己拖入需要指定的 Locomotion Checker
InputDevice	绑定挂载了 XRController 的控制器对象
TurnAngle	转动角度
EnableTurnAround	开启后按下摇杆允许旋转 180°（点击摇杆的按钮）
ActivationTimeout	执行连续的转弯时需要等待的时间

ContinuousTurner

连续转弯驱动。

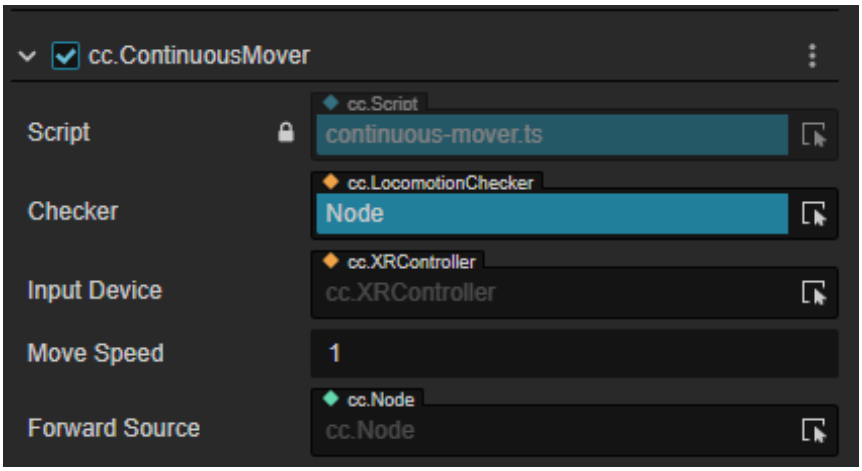


属性	说明
Checker	添加该组件时，默认绑定遍历当前场景得到的第一个挂载 Locomotion Checker 组件的节点 用户也可以自己拖入需要指定的 Locomotion Checker

InputDevice	绑定挂载了 XRController 的控制器对象
InputControl	绑定接受输入的摇杆
TurnSpeed	转动的角速度

ContinuousMover

平移运动驱动。



属性	说明
Checker	添加该组件时，默认绑定遍历当前场景得到的第一个挂载 Locomotion Checker 组件的节点 用户也可以自己拖入需要指定的 Locomotion Checker
InputDevice	绑定挂载了 XRController 的控制器对象
InputControl	绑定接受输入的摇杆
MoveSpeed	移动速度
ForwardSource	选择一个物体，用该物体节点的朝向作为移动的正方向

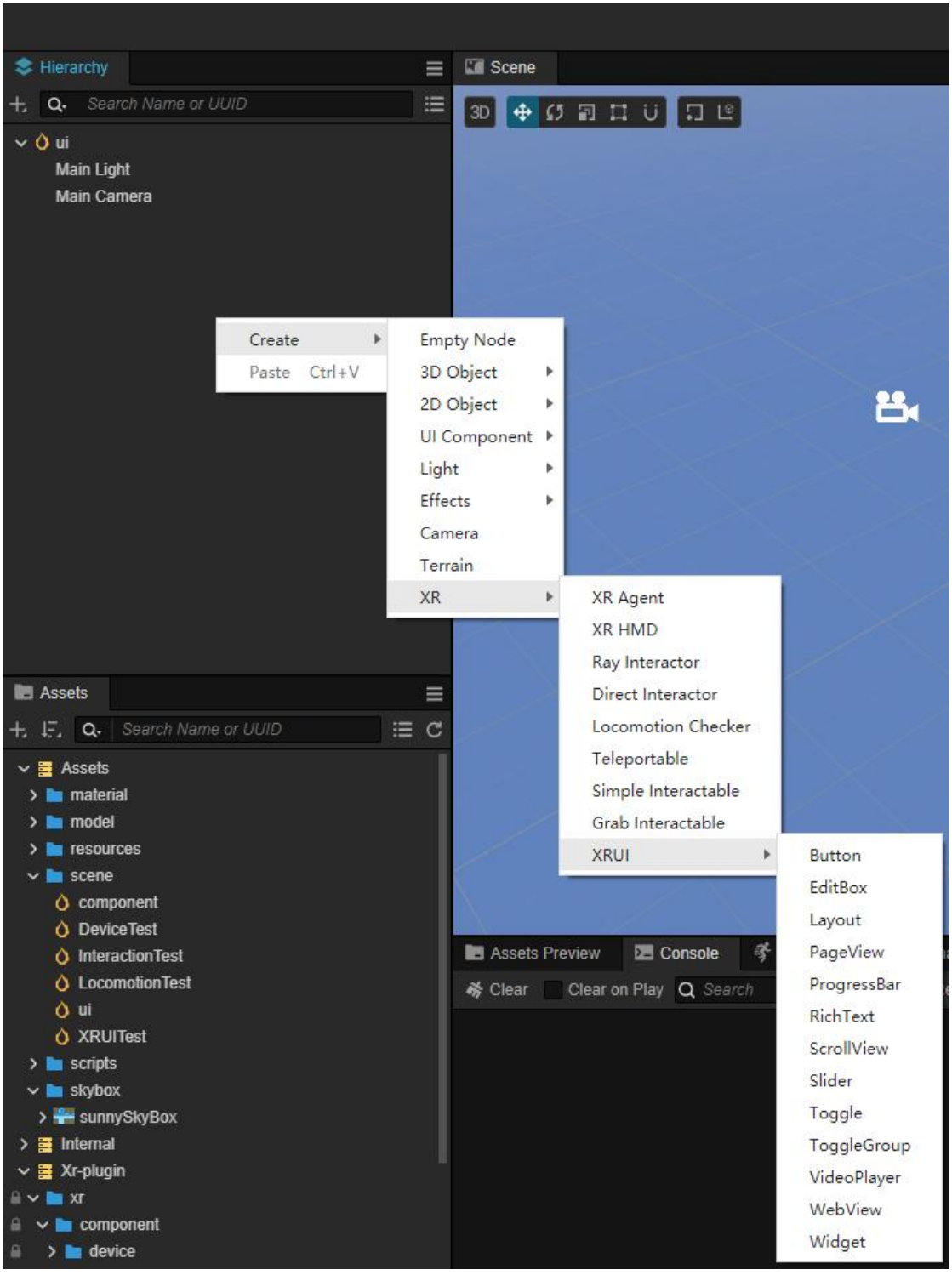
XR UI

传统的生成 UI 控件的方式都是把 UI 画在画布（Canvas）上，而画布本身不具有深度信息(位置属性不可更改)，导致了画布是贴在屏幕上的，只有与屏幕进行交互才能反馈作用于 UI 控件。由于 XR 设备的摄像头是两个目镜，不支持交互，这明显不满足于 XR 项目的需求。所以我们需要将 UI 的交互方式改为在空间中用交互器进行交互，因此需要将 UI 控件剥离出画布而能够单独存在于空间中，具有完整的位置属性并具有碰撞检测功能。

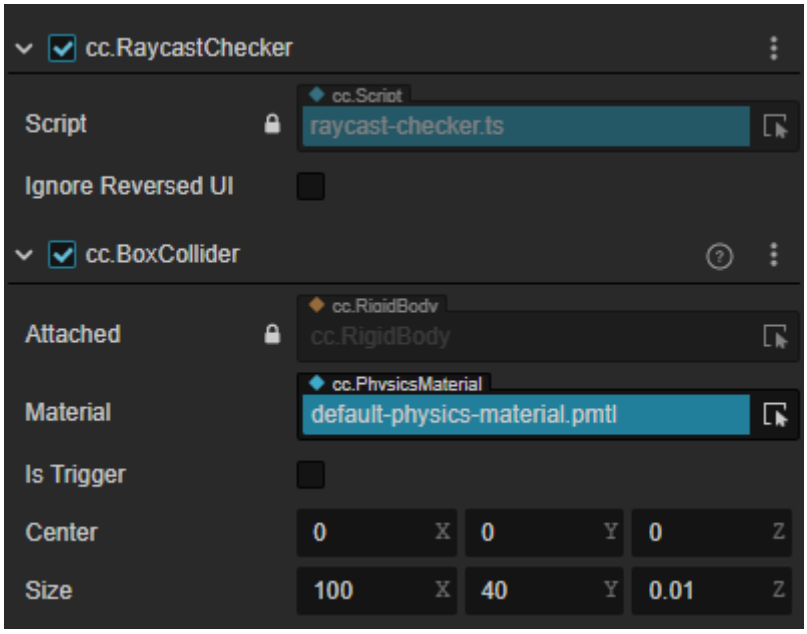
XR UI 是基于 2D UI 扩展而来，关于如何使用 2D/UI 组件可参考 2D 对象概述。

新建 UI

在 **层级管理器** -> **创建** -> **XR** -> **XR UI** 可以添加 XRUI。



相比于传统的 UI 控件，XR UI 会新增一些组件用于计算碰撞检测以触发交互，如图示，RaycastChecker 和 BoxCollider 可使其能够接收 XR 输入：



注意：对于 3D 空间上的 UI，其根节点上需要 RenderRoot2D 组件 才可以正常渲染。

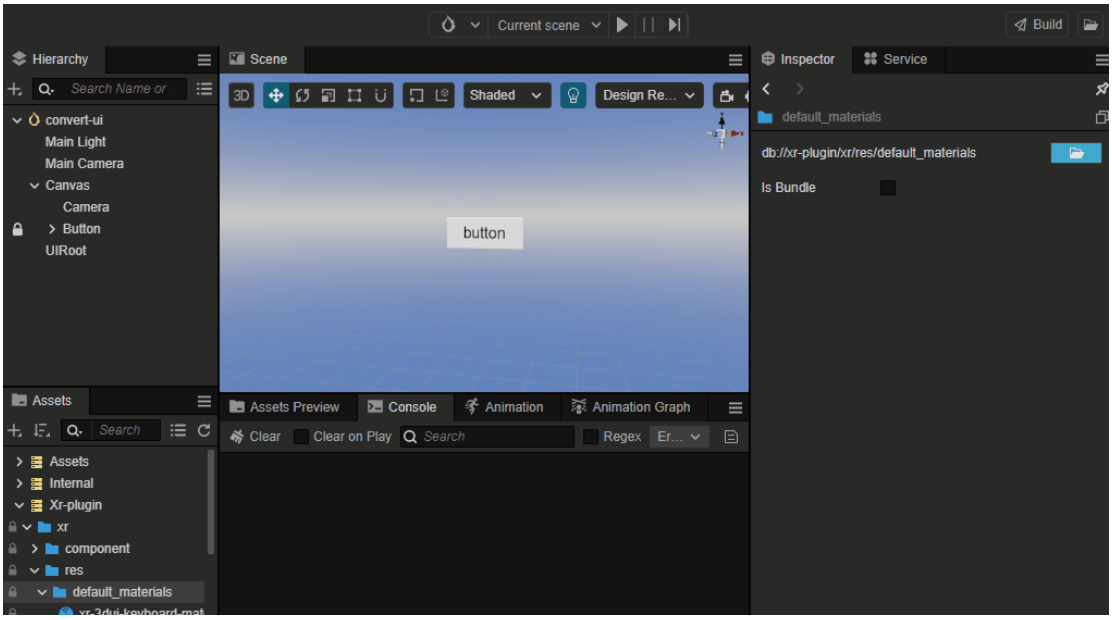
存量 UI 转换

如 UI 已制作完成，也参考下面的步骤将原本的 2D Canvas 下的 UI，转化为 XR 的 UI。

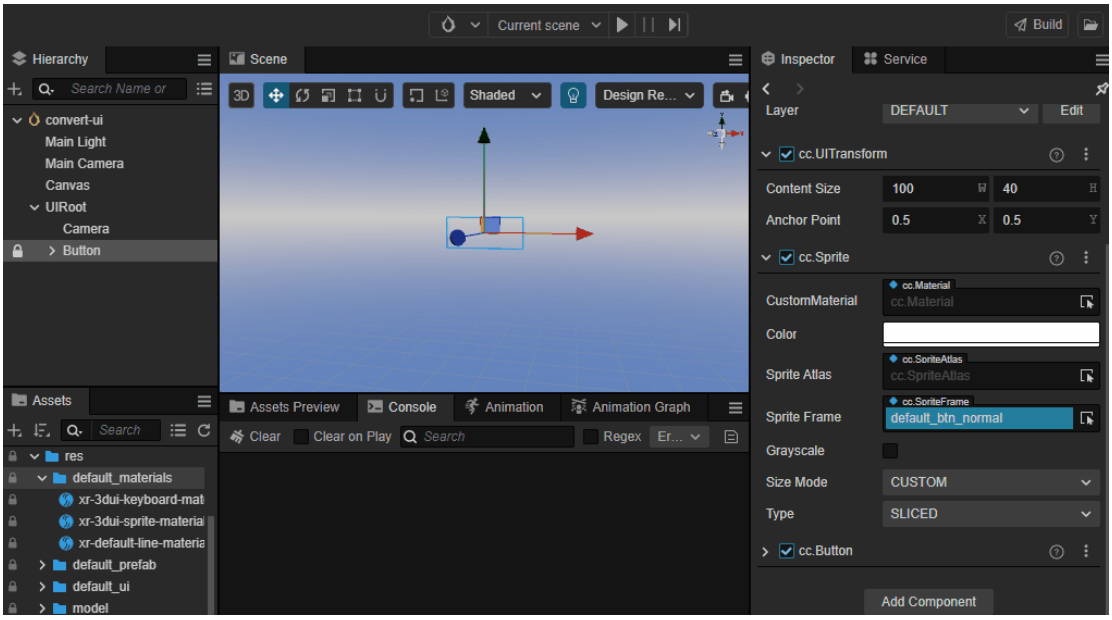
方法一：

- 在 **层级管理器** 右键创建一个空节点（如命名为 UIRoot，下文均使用 UIRoot），为节点添加组件 **RenderRoot2D** 组件，同时节点会自动添加 **UITransform** 组件：

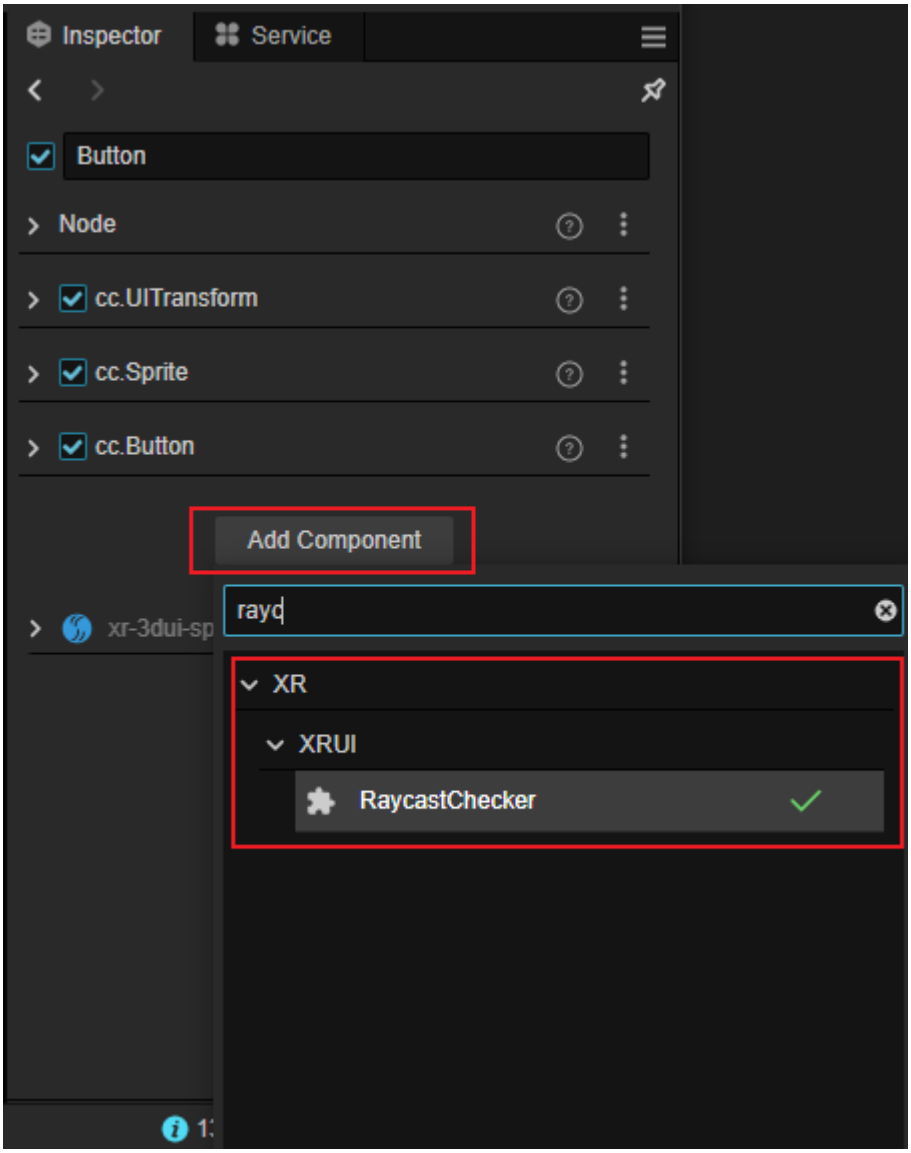
- 将原有的 2D UI 控件分离出 Canvas，移动至 UIRoot 层级下。修改 **Button** 位置和 **Layer** 属性。同时将 **Button** 及其子节点的 **Layer** 属性，都修改为和 **Camera** 的 **Layer** 属性一致（此处均为 DEFAULT）。



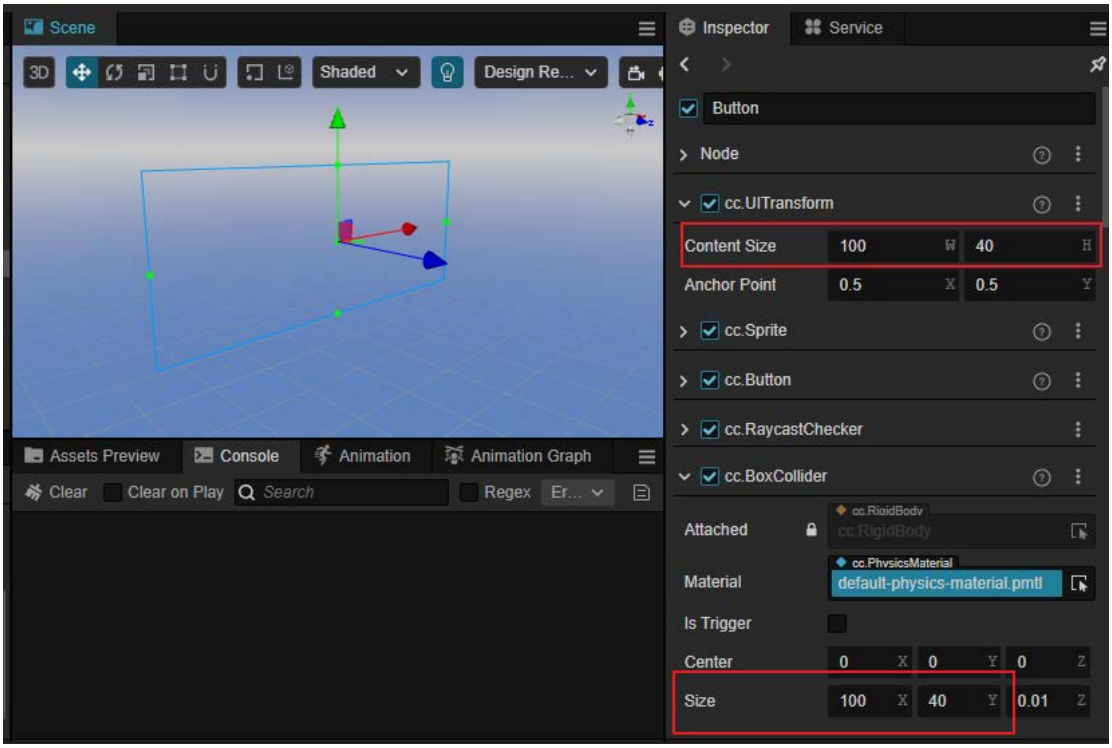
- 给 Button 及其子节点添加材质。在 **资源管理器** 中点击 **xr-plugin -> xr -> res -> default_materials**，选择 **xr-3dui-sprite-material** 拖拽至 **Sprite** 组件的 **CustomMaterial** 属性中。



- 给 Button 添加射线交互组件 RaycastChecker。点击 Button 节点，在 **属性检查器** 下方，点击 **添加组件** 按钮，选择 **XR -> UI -> RaycastChecker**。



- 在 **属性检查器** 中出现 **RaycastChecker** 组件和 **BoxCollider** 组件，且 **BoxCollider** 组件的 **Size** 属性的 **xy** 值与节点的 **UITransfrom** 的 **Content Size** 值一致，如下图：（此处可以将 **BoxCollider** 替换为其他所需 3D 碰撞体，能够贴合 UI 组件即可）。



- 移动 Button 节点至场景中指定位置，调整 Rotation 和 Scale 的值以满足设计需求。
- 添加完所有 UI 组件后，删除旧有的 Canvas 节点。

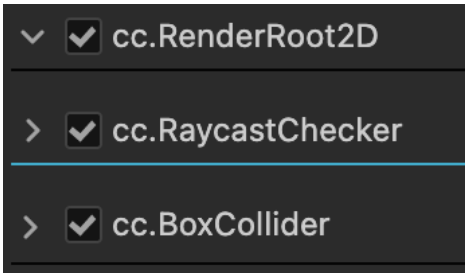
到此为止，存量 UI 到 XR UI 的转化就完成了。

方法二：

V1.1.0 版本新增一键转换为 XR UI 功能。

在场景下右键传统 2DUI，菜单中会出现 2DUI 转为 XRUI 选项

转换成功后 UI 节点会自动添加

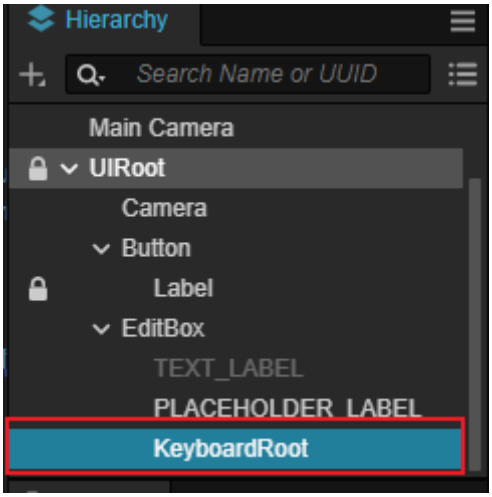


三个组件，并可以改变 UI 的深度值。

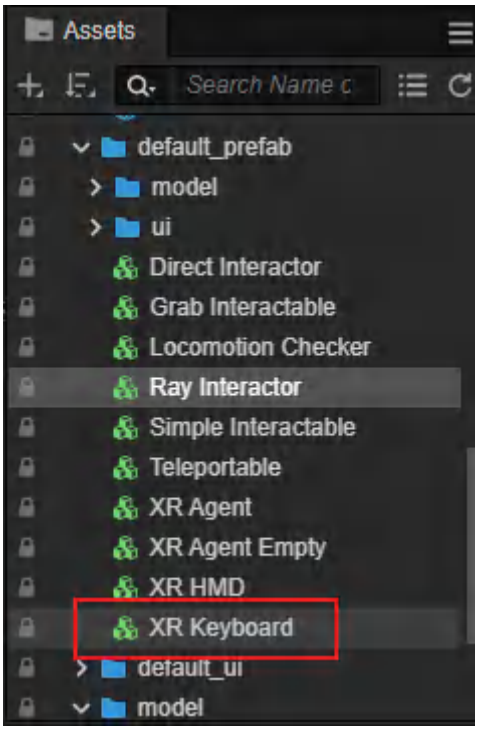
注：已经为 XRUI 的 UI 不能出现转为 XRUI

虚拟键盘（XR Keyborad）

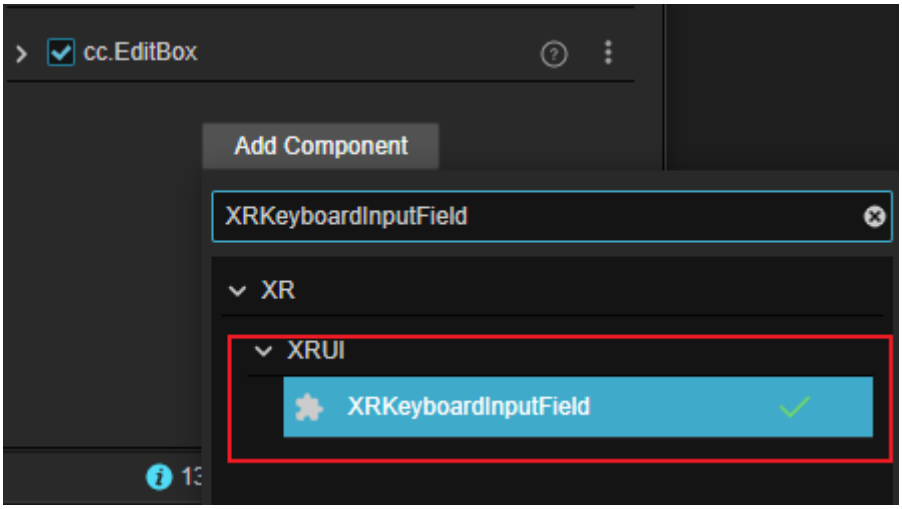
添加一个 EditBox 的 XR UI，同时给 EditBox 添加一个子节点，命名为 KeyboardRoot（命名随意），同时调整 KeyboardRoot 的位置信息（根据需求进行调整即可，可将 XR Keyboard 临时放在节点下进行调整）。

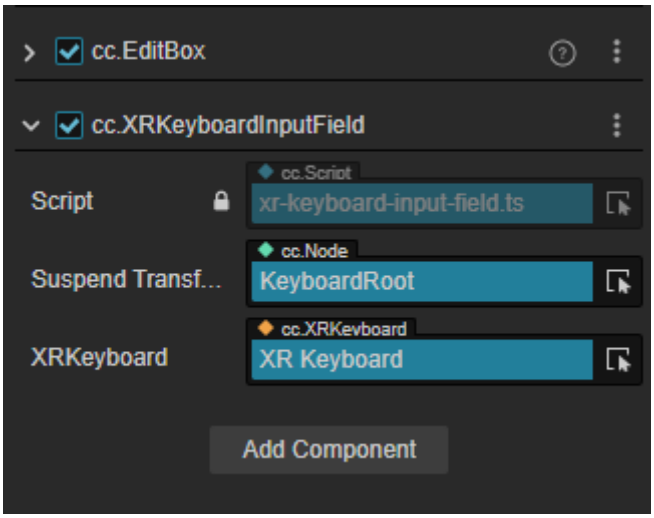


创建 XR Keyboard 对象：在资源管理器中点击 **xr-plugin-> xr -> res -> default_prefab**，选择 **XR Keyboard** 拖拽至场景中。



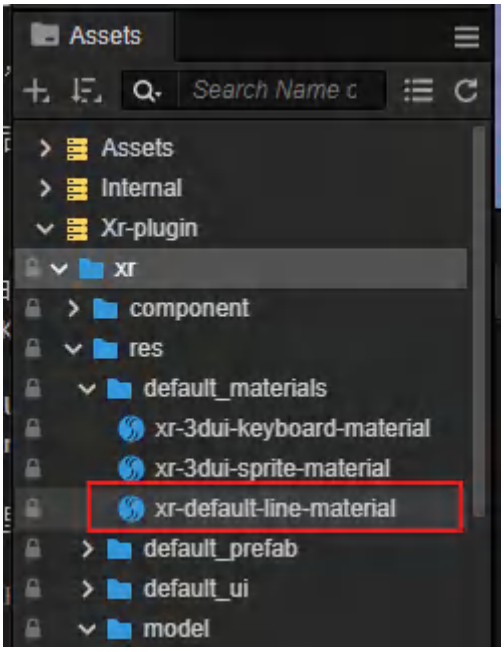
为 EditBox 节点添加 **XRKeyboardInputField** 组件，同时绑定 **SuspendTransform** 和 **XRKeyboard**，将节点拖拽进去。





射线材质

使用射线与 XR UI 进行交互时，需要给射线绑定材质 **xr-default-line-material**。位置在 **资源管理器->xr-plugin -> xr -> res -> default_materials**。

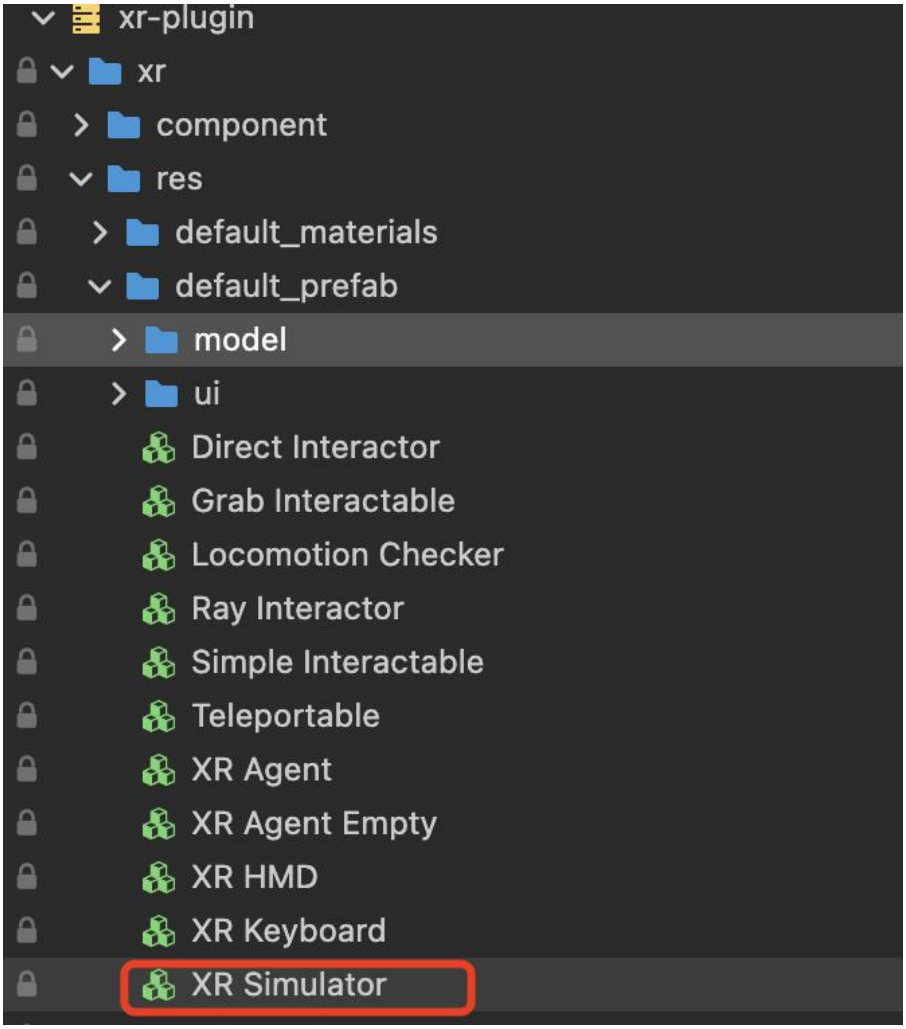


预览

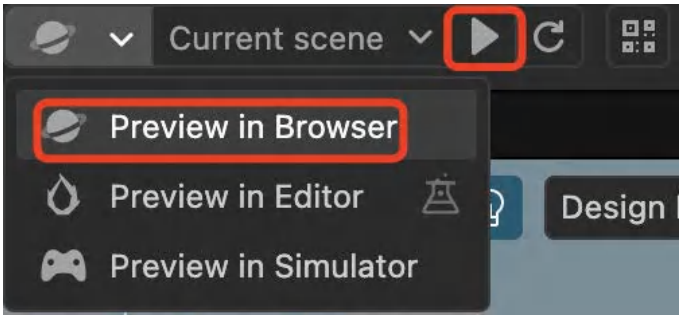
为了方便开发者在项目开发过程中实时调试，快速验证一些传统的功能逻辑来提高开发效率，Cocos CreatorXR 基于 Cocos Creator 的 Web Preview 功能开发了适用于 XR 项目的预览功能。

操作说明

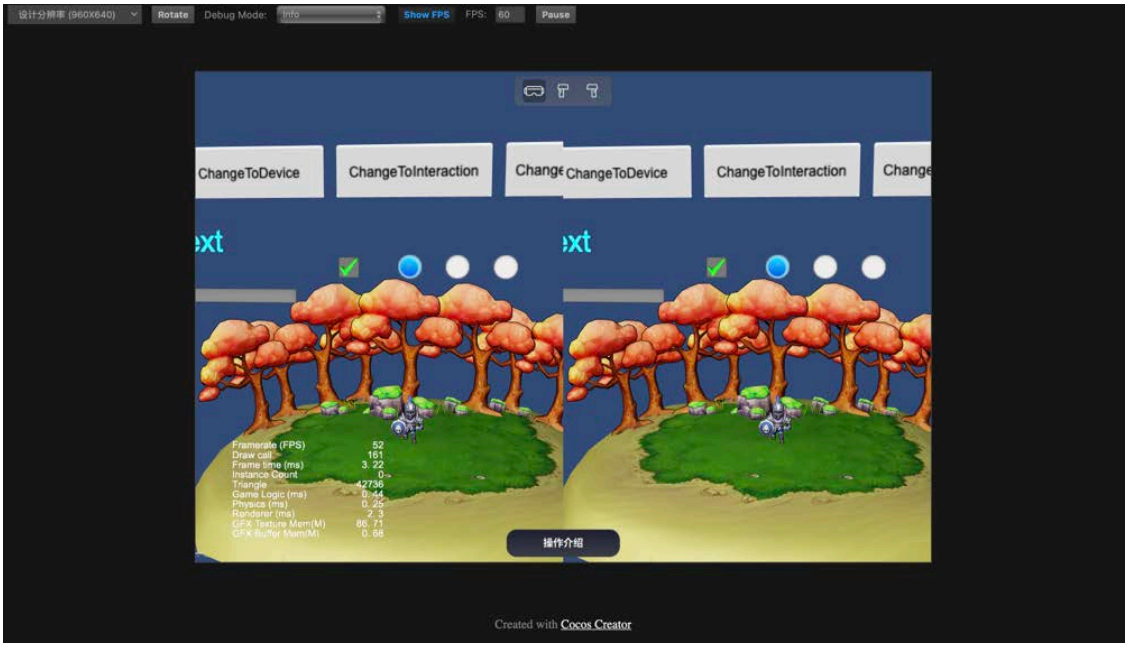
在 xr-plugin 的资源库中找到 XR Simulator，将其拖拽至场景中。



在编辑器的预览选项中选择浏览器预览，并点击运行。



运行后即可在浏览器中进行模拟预览。



键盘 WASD 来控制角色整体（HMD + 手柄）进行前左后右移动，QE 控制整体上升和下降。

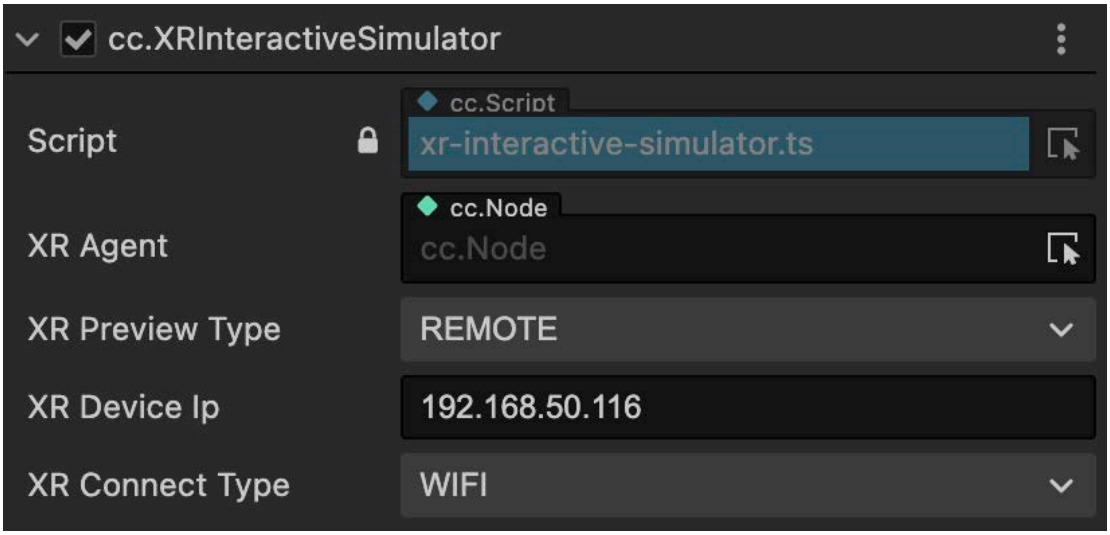
键盘 Latin 部 分的数字键 123 功能分别为：1.鼠标键盘的控制对象切换至 XR Agent（角色自身）此时前后左右上下作用于整体角色，鼠标滑动控制 HMD（Camera）转动，射线发出位置位于 HMD 中央，空格键用于触发 click（点击），按住空格拖动鼠标触发 drag（点击）；2 和 3 将鼠标和键盘的控制对象切换至左/右手柄，此时前后左右上下作用于单独的左/右手柄，射线从手柄位置发出，空格键用于触发 click（点击），按住空格拖动鼠标触发 drag（拖动）。

长按 B 键重置手柄位置。

控制手柄移动时，正前方向向量始终和 XR Agent 的前向保持一致。

XR设备无线串流调试

v1.1.0 版本的预览组件新增**无线串流模式**，内容验证是项目开发过程中极其耗时的一环，由于 XR 设备的终端独立性和串流工具的封闭性，使得在编辑器中针对 XR 设备的项目比传统移动端/PC 端项目的内容调试验证更为困难。为此，Cocos CreatorXR v1.1.0 推出了无线串流调试功能，开发者可以直接在 Web 浏览器中预览 XR 项目并同步所有来自 XR 设备的信号，正确渲染实时画面并反馈各种控制器信号触发逻辑，无需打包应用至设备即可快速完整地体验所有 XR 项目内容。



将 XR Preview Type 选择为 REMOTE 模式，XR Connect Type 选择为 WIFI 模式，将电脑和设备处于同一 WIFI 下。

填写 XR 设备网络 IP 到 XR Device IP 属性中。

在构建面板中勾选 Remote Preview，打包项目至 XR 一体机。

项目预览方式选择浏览器预览。

编辑器中点击运行进行浏览器预览，同时设备运行打包好的 apk。

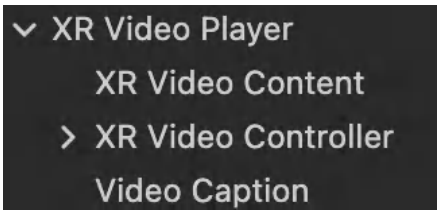
注：此功能需要**扩展版本>=v1.1.0，编辑器版本>=3.7.1**。

XR视频播放器

XR 头戴设备相较于传统的显示器拥有更为多样化的视频展示方式，结合设备自身的多轴向定位特性和双屏渲染画面，可以满足用户在 3D 场景中浏览全景视频或动态材质的需要。Cocos CreatorXR v1.1.0 提供了通用化的 XR 视频播放器，针对 XR 设备优化了视频渲染管线并支持切换展示窗口、180 度、360 度多风格的视频。同时，播放器还提供

了交互功能辅助您进行播放控制，您只需要添加或替换视频资源即可完成简易的视频播放功能的内容开发，简化创作步骤，降低开发门槛。

创建视频播放器，请在层级管理器右键 **创建 > XR > XR Video Player**，



其中包含的核心组件有：

XR Video Player：cc.XRVideoPlayer

用于调整视频的各项属性

属性	说明
Source Type	视频来源：REMOTE 表示远程视频 URL，LOCAL 表示本地视频地址
Remote URL	Source Type 为 REMOTE 时出现此项，远程视频的 URL
Clip	Source Type 为 LOCAL 时出现此项，本地视频剪辑
Play On Awake	视频加载后是否自动开始播放
Playback Rate	视频播放时的速率(0.0~2.5)
Volume	视频的音量（0.0～1.0）
Mute	是否静音。静音时音量设置为 0，取消静音时恢复原来的音量。
Loop	视频是否应在结束时再次播放
Keep Aspect Ratio	是否保持视频自身的宽高比（使用竖屏视频查看效果）
Shape	视频样式。
Content	关联带有 MeshRenderer 组件的 VideoContent 作为视频材质渲染对象。
Video Player Event	视频播放回调函数，该回调函数会在特定情况被触发，比如播放中、暂停、停止和播放完毕。

XR Video Controller：cc.XRVideoController

用于关联 UI 和视频功能。

属性	说明
Player	关联指定的 VideoPlayer，用于控制其播放功能。
HMD Control	绑定头戴显示器的控制器对象节点。
Left Hand Controller	绑定左手柄的控制器对象节点。
Right Hand Controller	绑定右手柄的控制器对象节点。
Play Pause	播放/暂停 UI。
Progress Bar	进度条 UI。
Fast Forward	快进按钮 UI。
Rewind	快退按钮 UI。
Video Shape UI	视频样式 UI。
Player Back Rate Bar	倍速 UI。
Volume UI	音量调节 UI。

Video Caption：cc.XRVideoCaption

用于解析字幕文件，目前只支持解析.srt 类型的字幕文件。

属性	说明
Caption Source Type	字幕来源：REMOTE 表示 URL 里的文件并解析字幕，LOCAL 表示本地字幕文件。
Remote URL	Source Type 为 REMOTE 时出现此项，字幕文件的 URL
Caption File	Source Type 为 LOCAL 时出现此项，本地字幕文件
Video Player	关联指定的 VideoPlayer，将字幕按时间同步于此视频。

AR 功能

AR 功能允许您使用 Cocos Creator 创建跨平台的增强现实(AR)应用程序。您可以给场景中的节点添加相应的 AR 功能组件来选择要启用哪些 AR 特性而无需关心平台的 AR SDK 之间的差异，当您在 AR 设备上构建和运行应用程序时，Cocos 引擎底层的 AR 模块(AR Module)会调用设备原生的 AR SDK 以启用 AR 功能，因此您可以只进行一次开发并发布到多种 AR 平台。

AR SDK设备支持情况

ARKit：

适用于 iOS 12.0+系统的设备，Meshing 特性需要设备具备 LiDAR；开发环境需要 Xcode 12.4+。

ARCore：

参考 Google 官方：支持 ARCore 的设备 | Google Developers 或者检查 Android 手机是否在官方系统版本里预装了“Google Play 商店”及“Google Play Services for AR”这两个应用，并且未处于“停用”状态。

AREngine：

参考华为官方：业务介绍-AR Engine | 华为开发者联盟 (huawei.com)

AR 特性支持情况

当前版本 AR 插件支持以下几种 AR 特性。

AR 特性	描述
AR Session	在目标平台上启用、禁用和配置 AR Session，以控制 AR 应用的生命周期。
Device Tracking	跟踪设备在物理空间中的位移和旋转。
AR Camera	渲染设备相机传输的视频流背景图像并可以根据环境进行光照估计。

Plane Tracking	检测和跟踪物理世界中的平面。
Image Tracking	检测和跟踪物理世界中的图像。
Hit Detection	支持使用射线（Ray cast）与跟踪实体进行命中检测。
Anchors	跟踪场景空间中的固定点。
Meshing	将物理世界网格化。

AR 特性	ARKit	ARCore	AREngine	Spaces
AR Session	✓	✓	✓	✓
Device Tracking	✓	✓	✓	✓
AR Camera	✓	✓	✓	✓
Plane Tracking	✓	✓	✓	✓
Image Tracking	✓	✓	✓	✓
Hit Detection	✓	✓	✓	
Anchors	✓	✓	✓	
Meshing	✓			

案例

AR 插件的简要案例请参考 AR Sample。

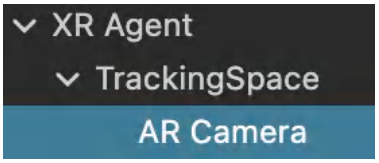
版本要求

编辑器请使用 Cocos Creator v3.7.1 或更高的版本。

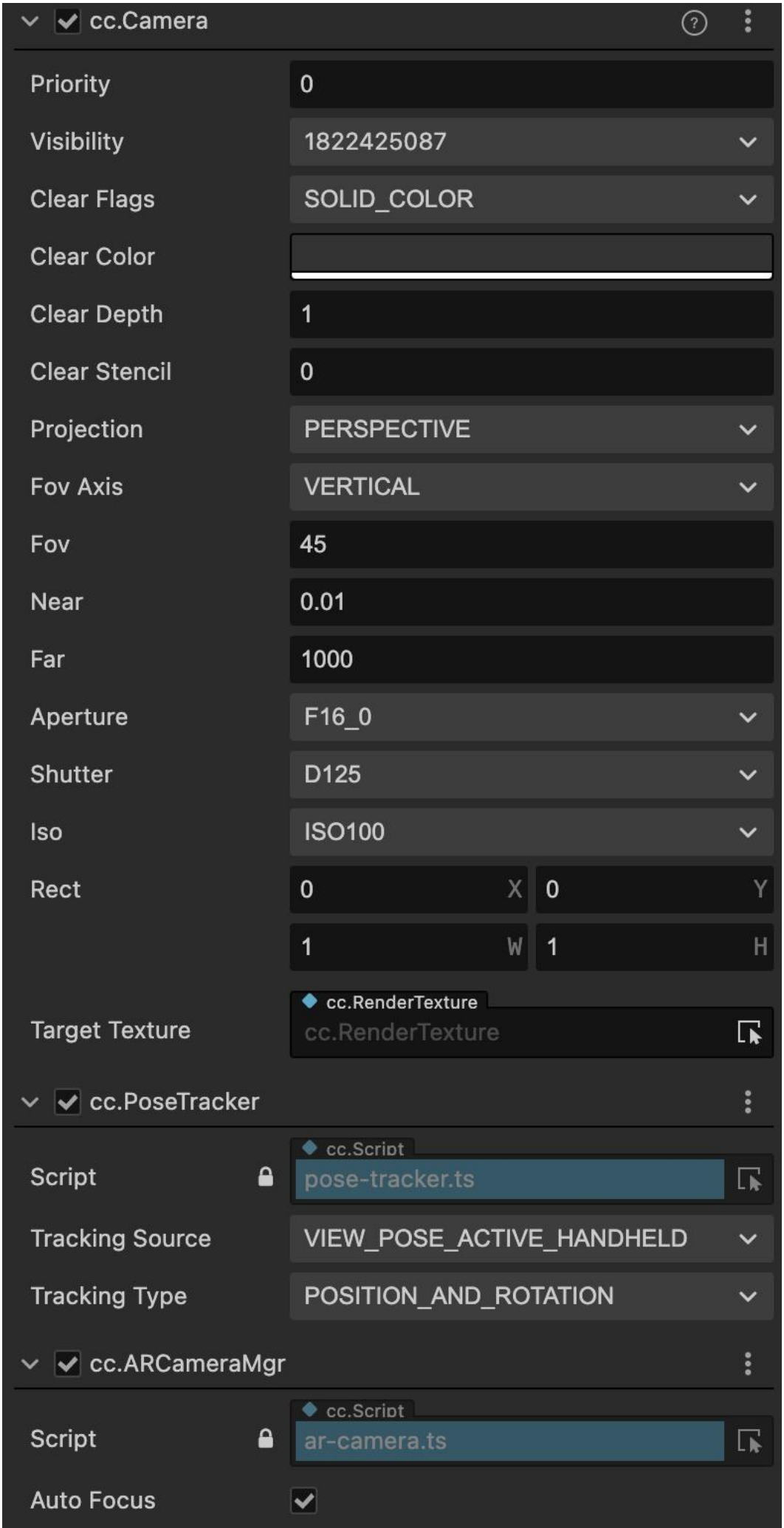
插件请使用 xr-plugin v1.1.0 或更高的版本。

AR相机

和头戴显示器一样，场景中为了能够抽象表式移动端设备带有 AR 能力的摄像机，XR 插件使用 AR Camera 组件封装一系列属性来映射物理设备的摄像头 AR 功能。



AR Camera 对象包含三种必要的组件：cc.Camera、cc.PoseTracker 和 cc.ARCameraMgr。


cc.Camera
Priority
0
Visibility
1822425087
Clear Flags
SOLID_COLOR
Clear Color
Clear Depth
1
Clear Stencil
0
Projection
PERSPECTIVE
Fov Axis
VERTICAL
Fov
45
Near
0.01
Far
1000
Aperture
F16_0
Shutter
D125
Iso
ISO100
Rect
0 X
0 Y
1 W
1 H
Target Texture
cc.RenderTexture
cc.RenderTexture
cc.PoseTracker
Script
cc.Script
pose-tracker.ts
Tracking Source
VIEW_POSE_ACTIVE_HANDHELD
Tracking Type
POSITION_AND_ROTATION
cc.ARCameraMgr
Script
cc.Script
ar-camera.ts
Auto Focus

cc.Camera 是 Cocos 引擎提供的传统的摄像机组件，为了保证良好的体验，推荐将**缓冲清除标志位（Clear Flags）**设置为 **SOLID_COLOR**，**近裁剪面（Near Plane）**设置为 **0.01**。更多相机参数介绍请查阅相机组件介绍。

cc.PoseTracker 用于将物理设备的位姿信息同步至 AR Camera，保证摄像机能够正确渲染虚拟内容并和视频流叠加，与 XR HMD 不同，适配移动端的手持设备时 **Tracking Type** 要选择 **VIEW_POSE_ACTIVE_HANDHELD**。

cc.ARCameraMgr 是用于管理 AR 摄像机功能的组件，详细属性介绍请参考**设备映射组件 > ARCameraMgr**。

注：光照估计目前只能在 ARCore、AREngine 平台上生效。

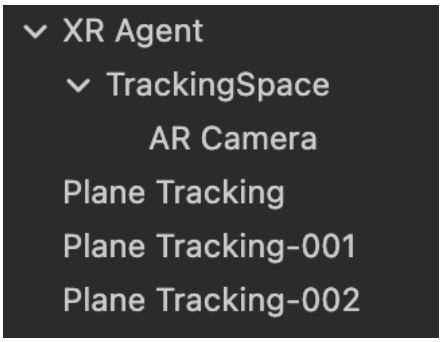
AR Manager

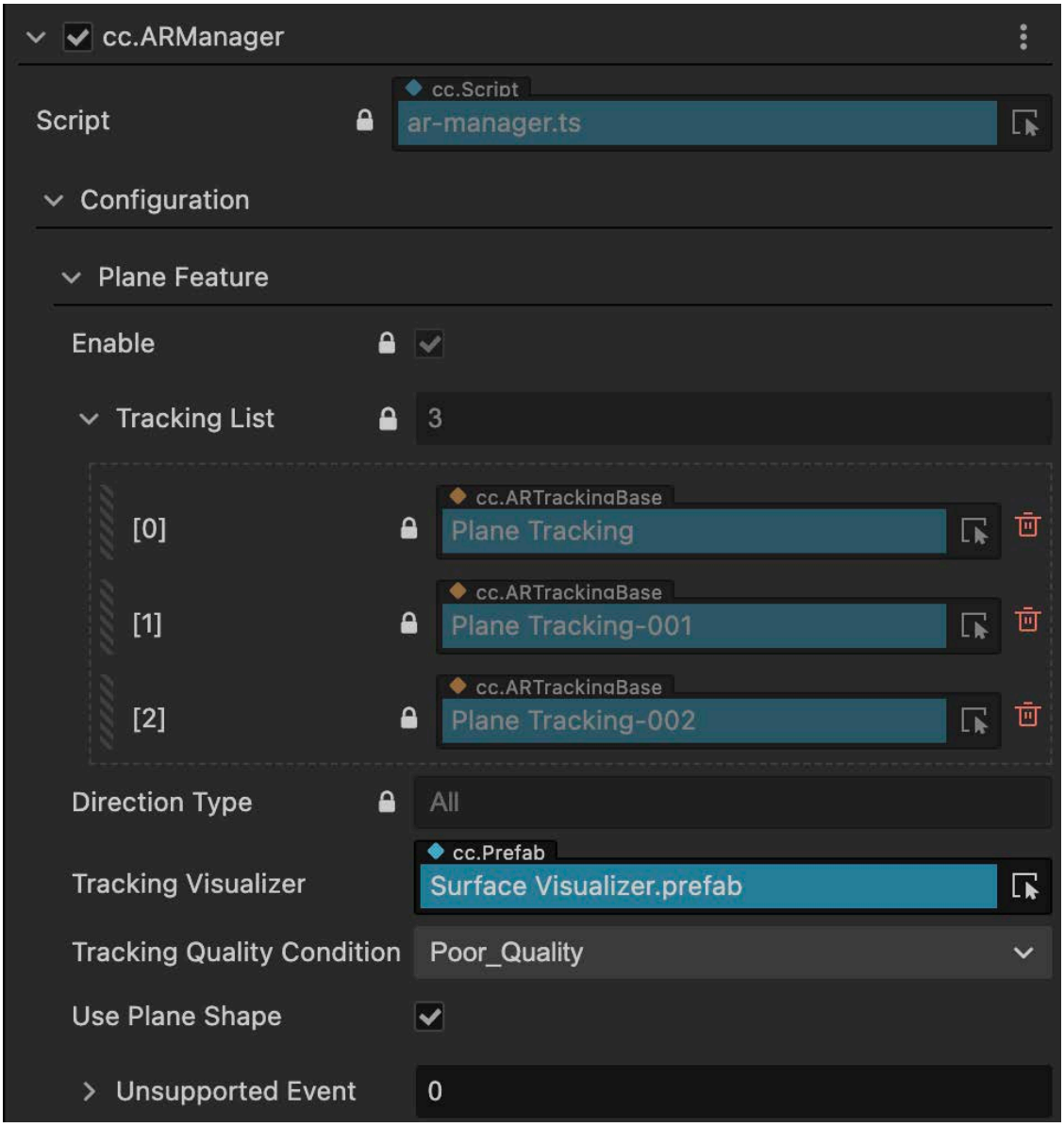
插件提供了一个全局管理器，用于收集当前项目使用到的 AR 特性并进行管理，特性管理器中每个特性的属性都是全局属性，调整参数会修改设备相关的或者项目全局的功能。cc.ARManager 默认挂载在 **XR Agent** 节点上，当您创建 AR 自动化行为节点时，ARManager 会收集此节点到其对应的特性列表中，方便后续管理和维护所有特性节点。

当前版本针对以支持的 AR 特性提供了对应的全局功能属性：

平面追踪特性

当您在场景中创建一个或多个 Plane Tracking 节点，AR Manager 中的 Configuration 会新增 Plane Feature 属性。您可以调整特性下的各项参数，或定位到对应的特性节点。





Direction Type 汇集了当前场景所有平面代理需要识别的平面朝向。

Tracking Visualizer 给所有平面代理创建默认的可视化模型。

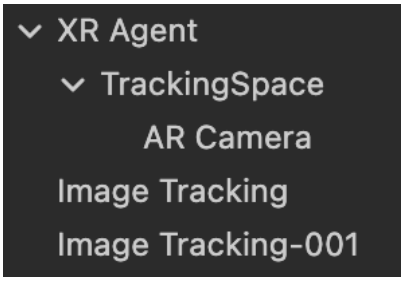
Tracking Quality Condition 表示当前可追踪的平面代理的最差质量，当稳定性质量小于此值时，平面不被可视化。

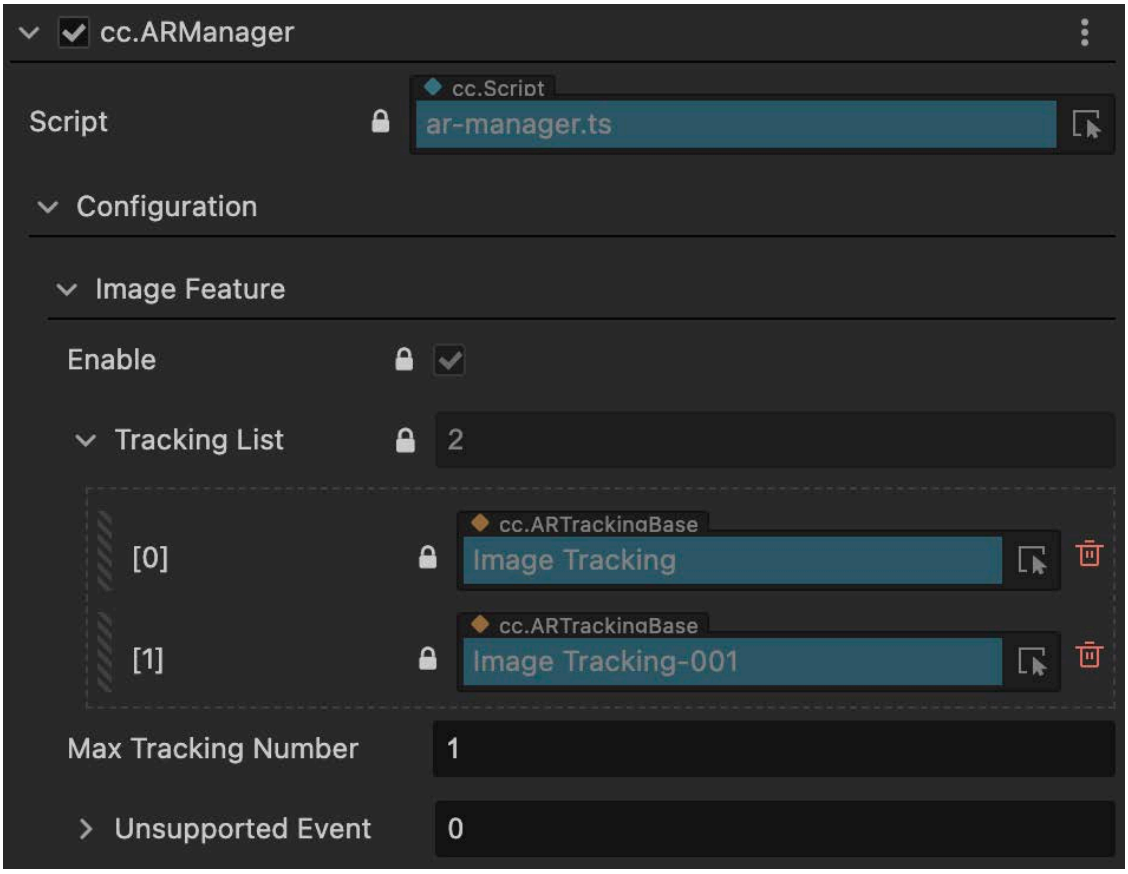
Use Plane Shape 开启后可视化效果会根据现实环境中平面的真实物理性状使用多边形绘制，关闭后则使用四边形绘制代替。

Unsupported Event 会在设备不支持平面追踪时触发，用户可以根据需求添加事件。

图像追踪特性

当您在场景中创建一个或多个 Image Tracking 节点，AR Manager 中的 Configuration 会新增 Image Feature 属性。您可以调整特性下的各项参数，或定位到对应的特性节点。





Max Tracking Number 表示当前镜头内可同时追踪图片的最大数量，可以根据需要动态修改此值。

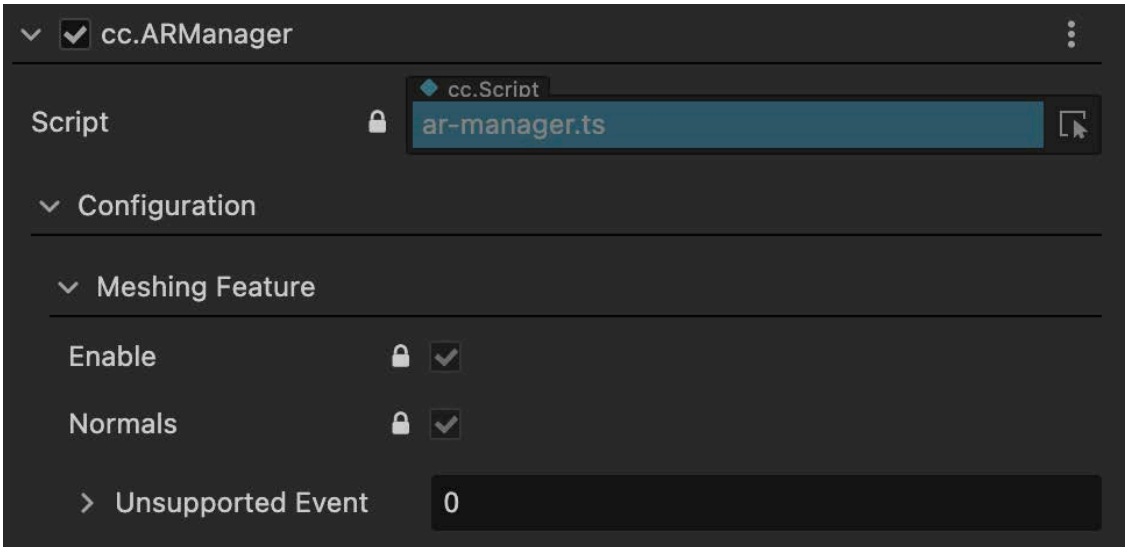
注：Max Tracking Number 的上限根据设备平台会有差异，目前已知

- ARCore 平台可同时追踪最多 20 张图，单图像库存最多 1000 张。
- ARKit 平台可同时追踪最多 4 张图，单图像库最多 100 张。
- AREngine 平台可同时追踪最多 1 张图。

Unsupported Event 会在设备不支持图像追踪时触发，用户可以根据需求添加事件。

网格化特性（实验性）

当您在场景中创建一个或多个 Meshing 节点，AR Manager 中的 Configuration 会新增 Meshing Feature 属性。由于 Meshing 功能处于实验性阶段且支持环境重构的设备硬件要求较高，暂不支持对此做特性做参数控制。



Meshing Feature

Normals 默认开启，可以根据 Mesh 信息获取法线向量。

AR Manager 还提供了控制特性开关以及特性可视化开关的接口：

```
public enableFeatureTracking (type: ARTrackingType, enable: boolean) {
    this.configuration.setFeatureEnable(type, enable);

    if (this._systemsMap.has(type)) {
        const system = this._systemsMap.get(type)!;
        system.enableFeature(enable);
    }
}
```

```
public showAllVisualizer (type: ARTrackingType) {
    if (this._systemsMap.has(type)) {
        const system = this._systemsMap.get(type);
        (system as any as IExtraDisplay)?.showAllVisualizer();
    }
}

public hideAllVisualizer (type: ARTrackingType) {
    if (this._systemsMap.has(type)) {
        const system = this._systemsMap.get(type);
        (system as any as IExtraDisplay)?.hideAllVisualizer();
    }
}
```

AR自动化行为编辑

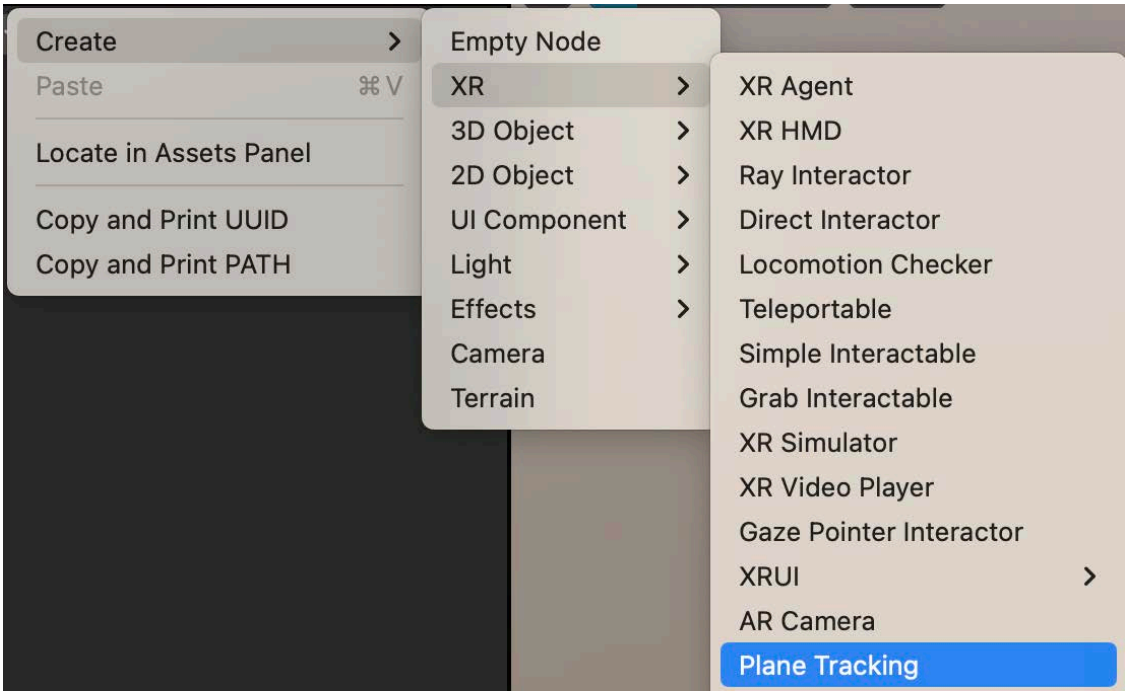
AR 场景中，虚拟物体与现实实体间总是存在未知的依赖关系，如果能够清晰方便的描述现实实体的条件特征并针对此种条件执行匹配的行为，可以很大程度上简化开发者处理复杂的 AR 功能特性而专注于编写项目核心逻辑。Cocos CreatorXR 提供了 **AR 自动化行为编辑组件**，将常用的物理特征和逻辑行为抽象成元素供开发者自由搭配，图形化的操作极大程度降低了 AR 应用的开发成本和开发门槛。

每种特性的自动化行为编辑组件都有其独特的特征库和行为库，以下介绍了当前版本支持自动化行为的所有 AR 特性：

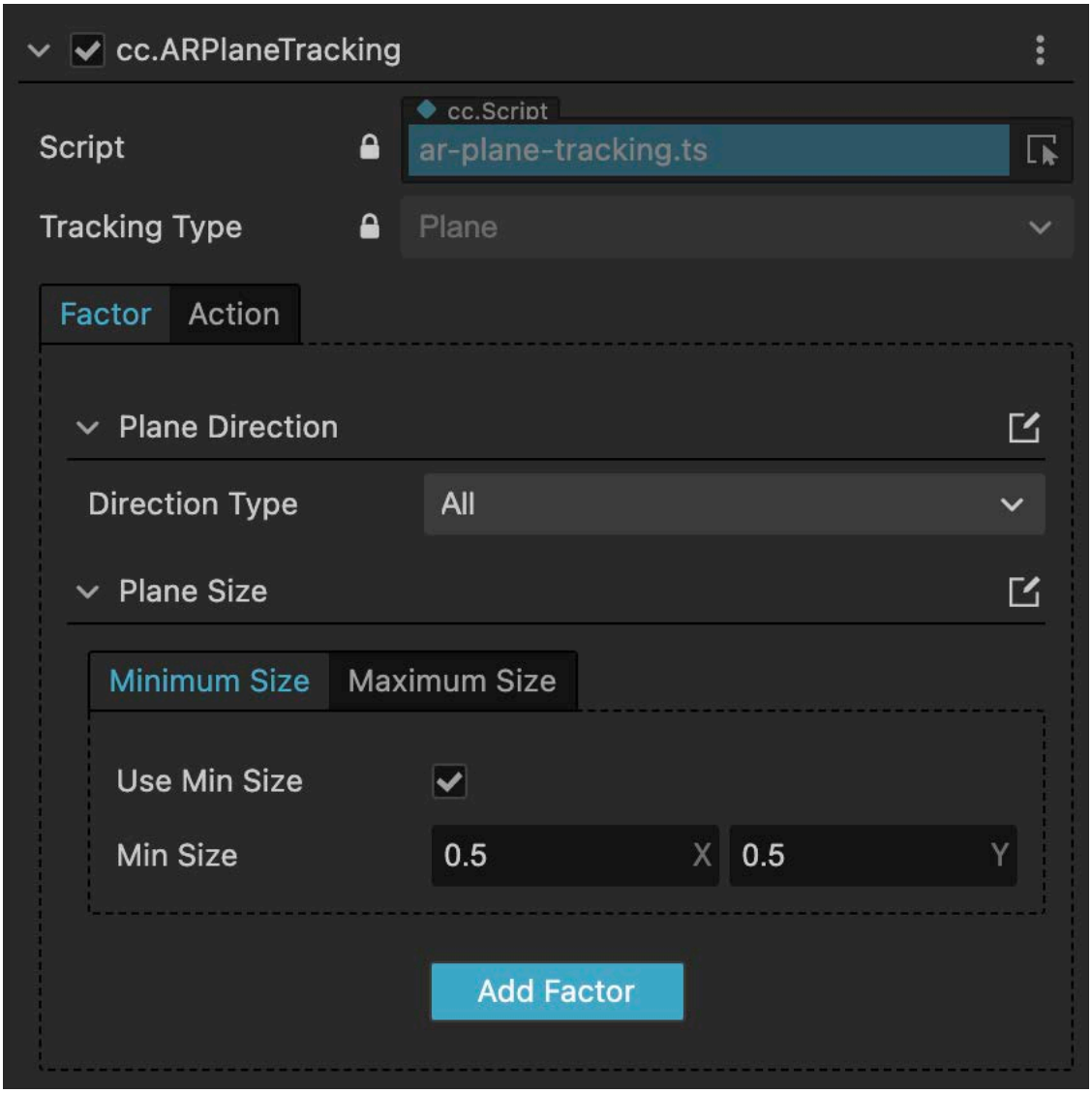
平面追踪

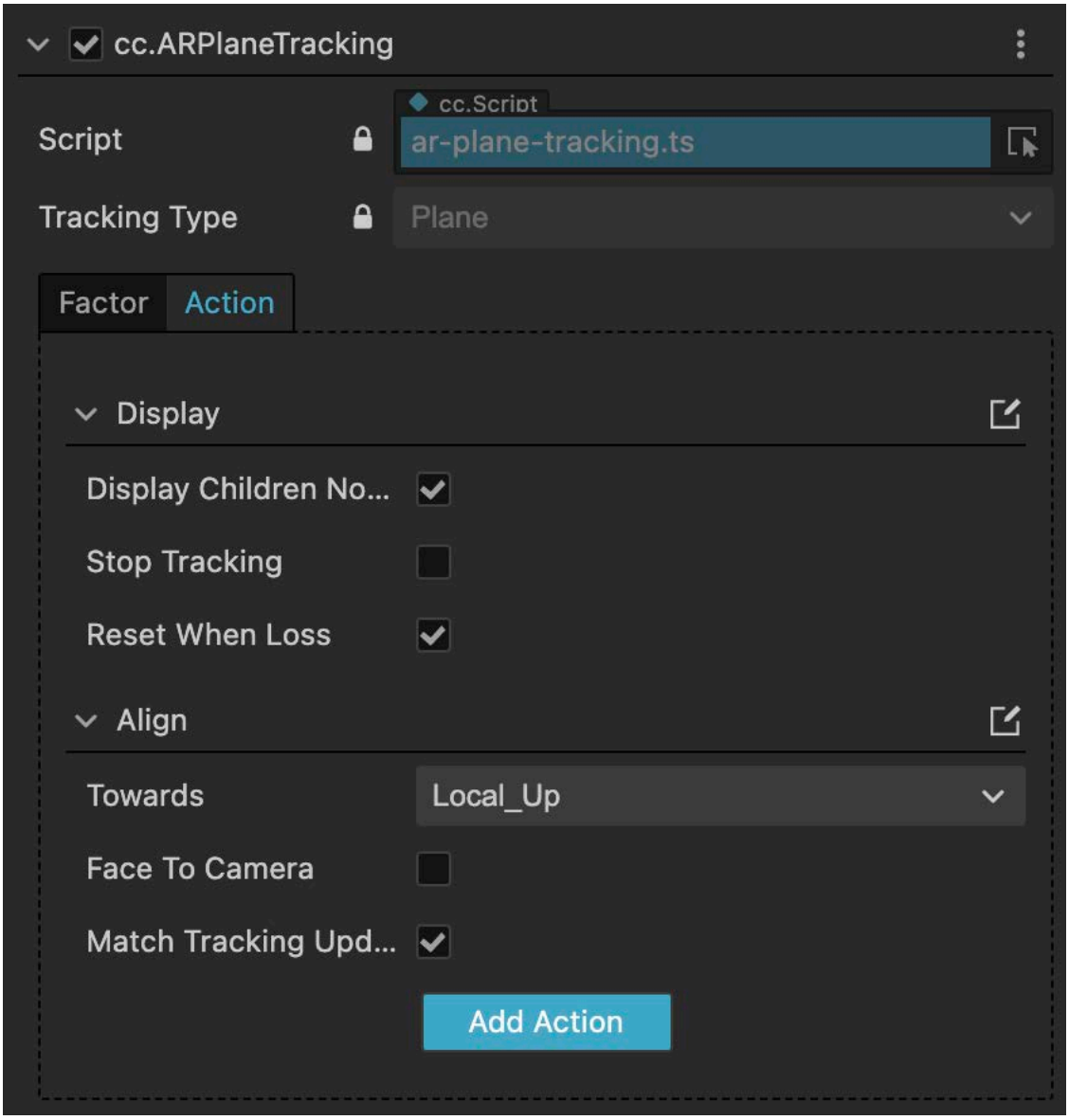
平面追踪允许您在运行时使用设备 AR 能力识别出物理世界中的平面特征数据，并可以将这些平面数据可视化显示在应用程序中。

在编辑器的层级管理器中右键 **创建 > XR > Plane Tracking**，创建平面代理节点，此节点可用于描述物理世界中的某一个平面实体。

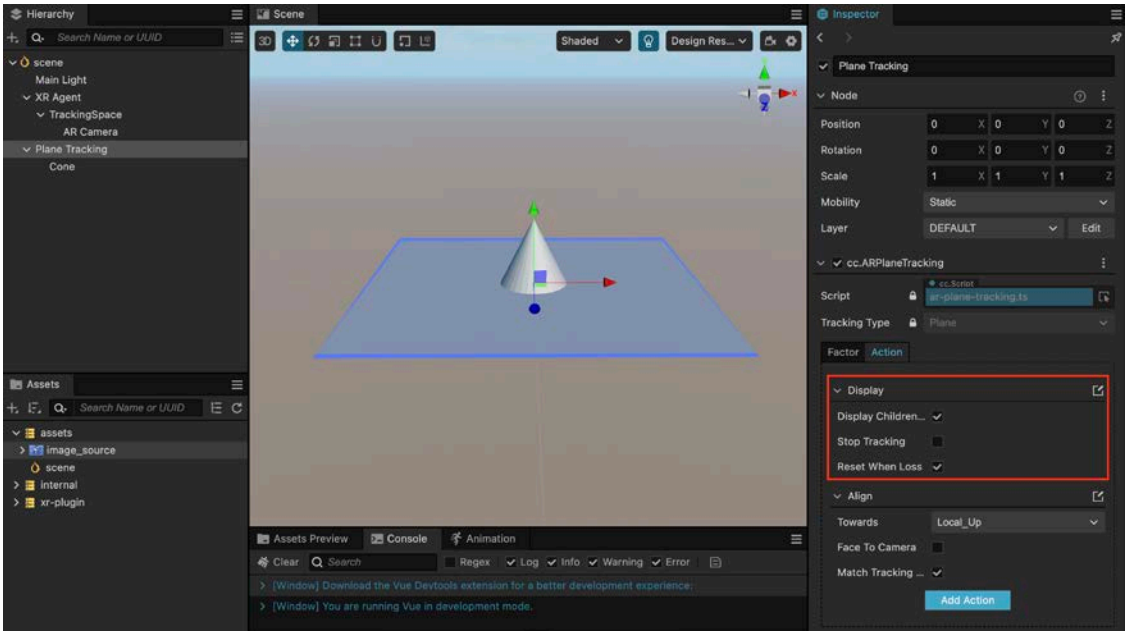


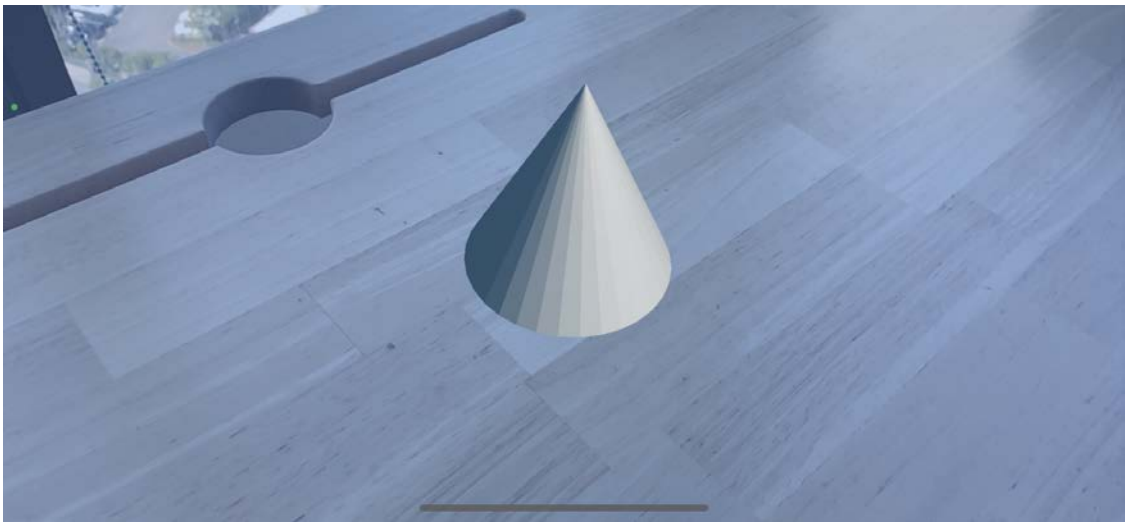
选中创建好的 Plane Tracking 节点，在属性检查器中可以看到默认添加好的 cc.ARPlaneTracking，选择 Factor 或 Action 页签可以查看当前已有的特征或行为项。点击 Add Factor 或 Add Action 可以添加特征库/行为库中其他的新项。


cc.ARPlaneTracking
Script
cc.Script
ar-plane-tracking.ts
Tracking Type
Plane
Factor
Action
Plane Direction
Direction Type
All
Plane Size
Minimum Size
Maximum Size
Use Min Size
Min Size
0.5
X
0.5
Y
Add Factor



在创建好的 Plane Tracking 节点下拖入需要展示的虚拟物体，调整为合适的大小比例。在 Action 中添加 Display 行为项（默认已经添加），即可在运行时识别到满足条件的平面后展示出虚拟物体。

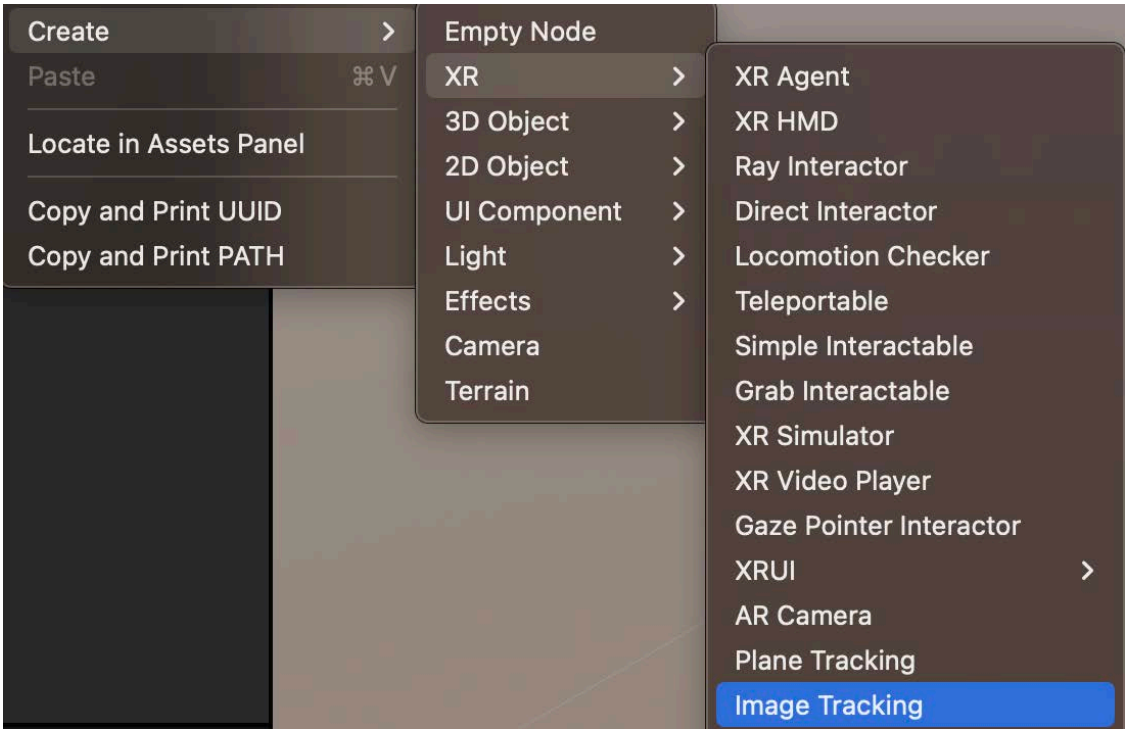




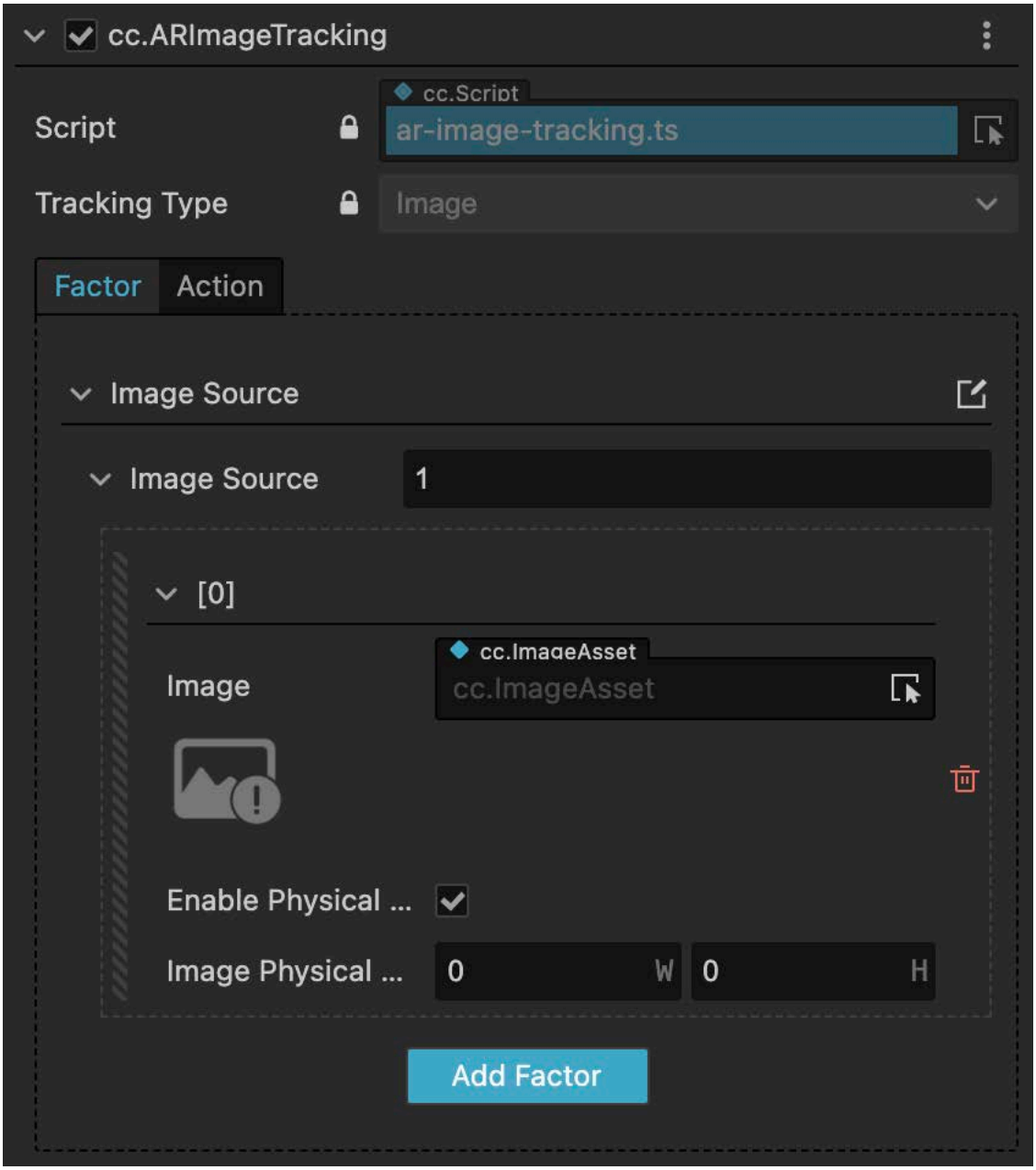
图像追踪

图像追踪允许您在运行时使用设备 AR 能力识别出 2D 图像资源。

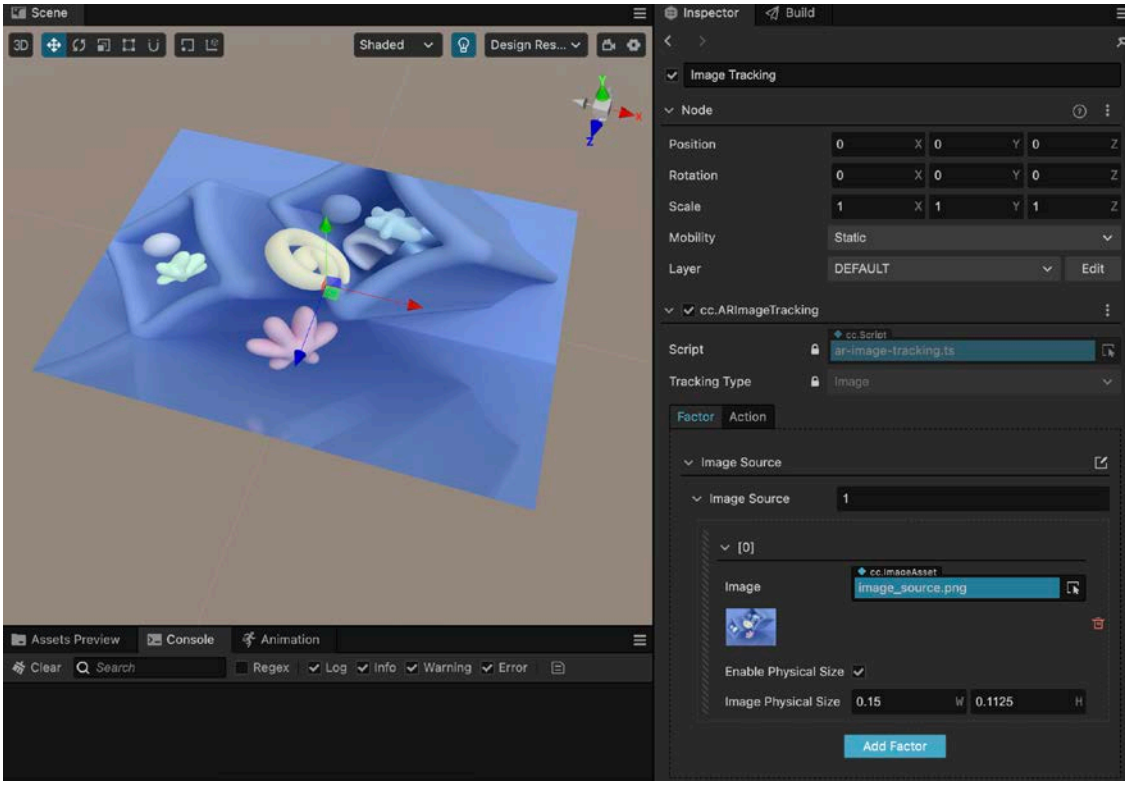
在编辑器的层级列表中右键 **创建 > XR > Image Tracking**，创建图像代理节点，此节点可用于描述物理世界中的某一个图像实体。



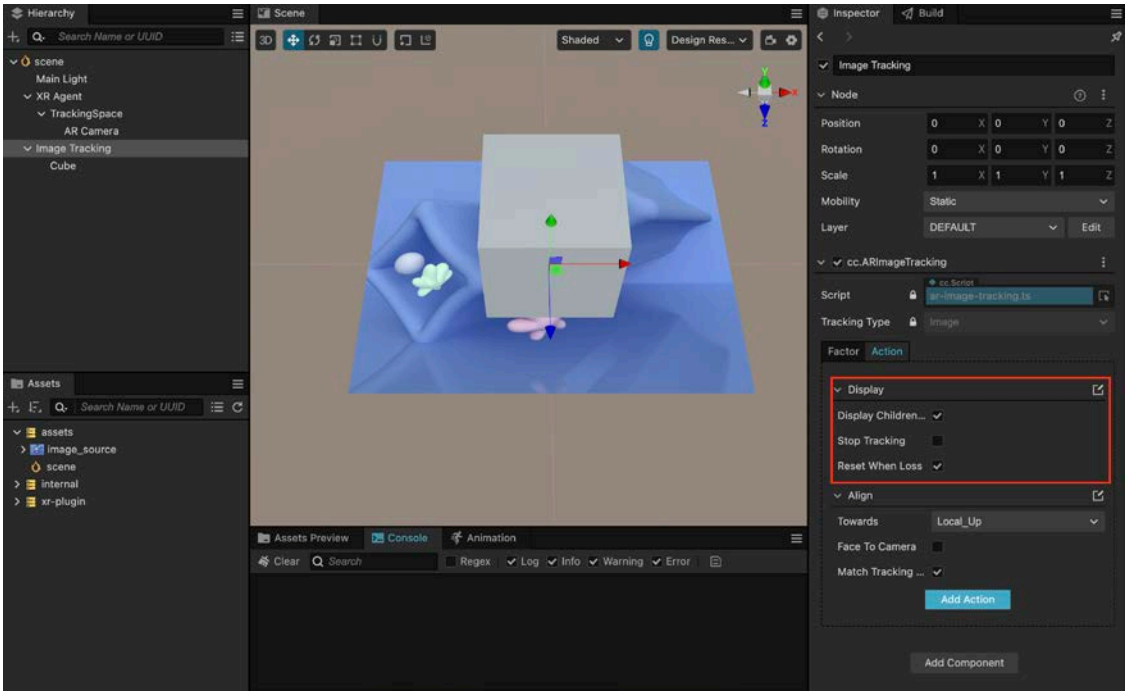
选中创建好的 Plane Tracking 节点，在属性检查器中可以看到默认添加好的 cc.ARImageTracking，在 Factor 页签的 Image Source 属性中新增一个图像资源。

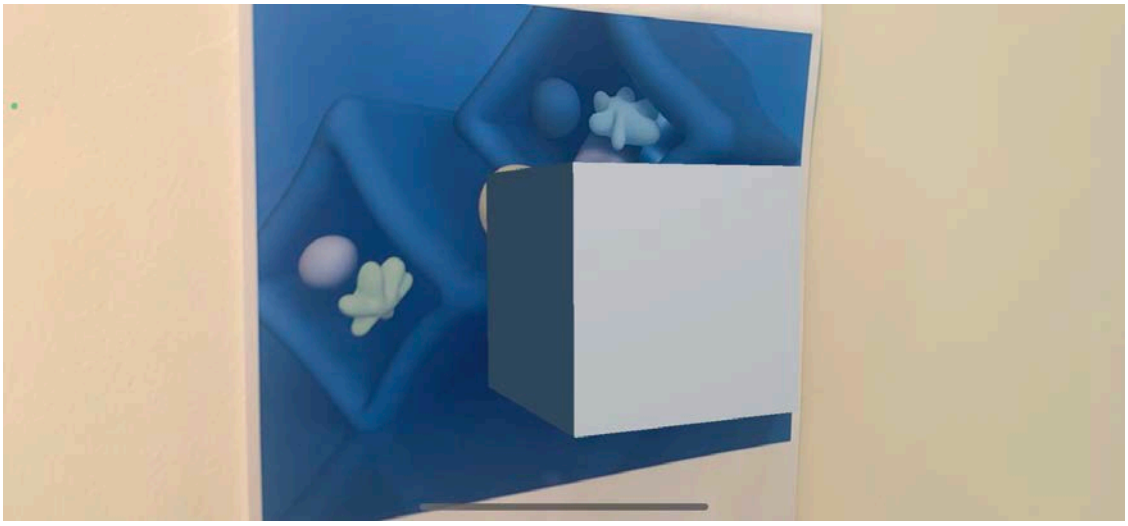


在 Image 属性中拖入或直接选择资源管理器中的图片资源，在编辑器场景中可以看到当前引用的图像，设置图像的默认的物理尺寸。



在创建好的 Image Tracking 节点下拖入需要展示的虚拟物体，调整为合适的大小比例。在 Action 中添加 Display 行为项（默认已经添加），即可在运行时识别到图像后展示出虚拟物体。

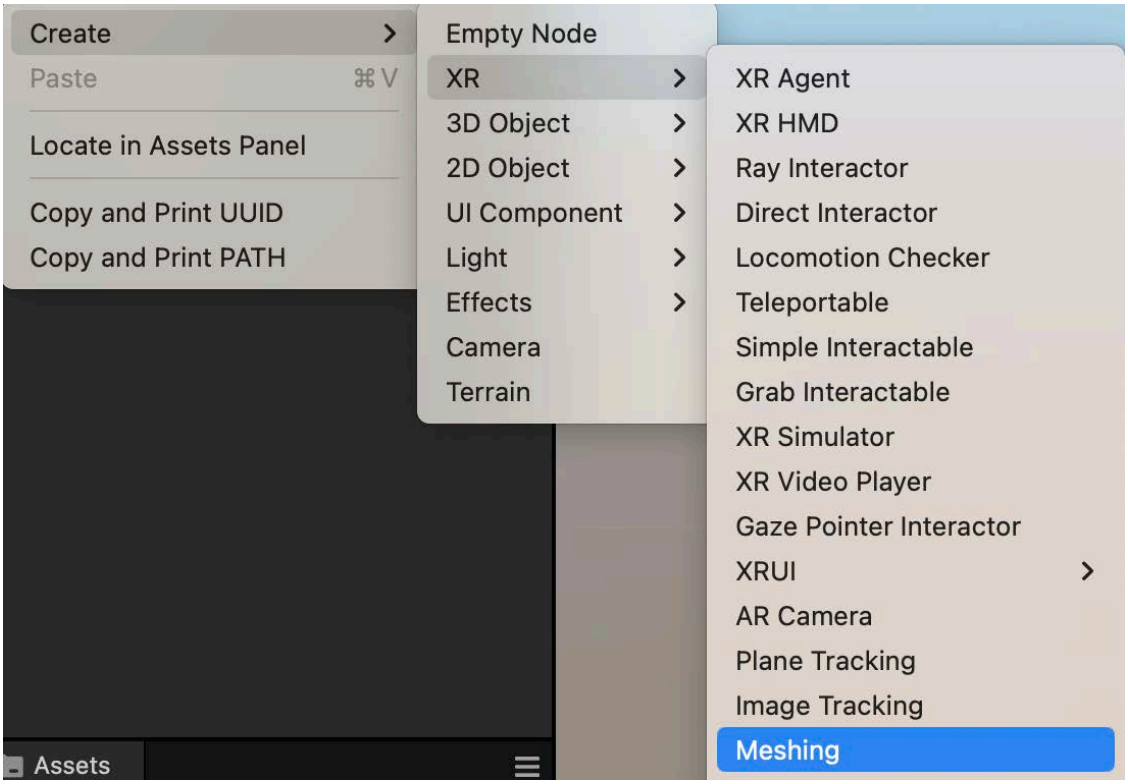




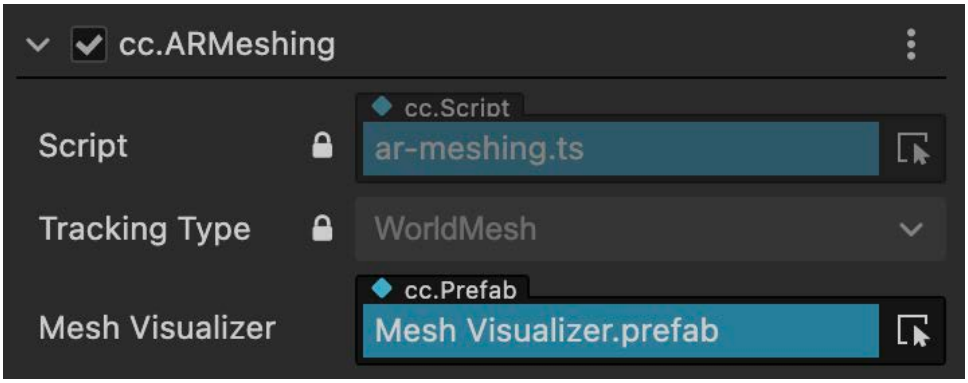
网格化（实验性）

目前网格化特性只对 iOS 平台中具有深度场景重建能力的设备上生效（带有 LiDAR 扫描仪的 iPhone/iPad Pro 系列），网格化允许您根据现实环境创建 3D 网格。

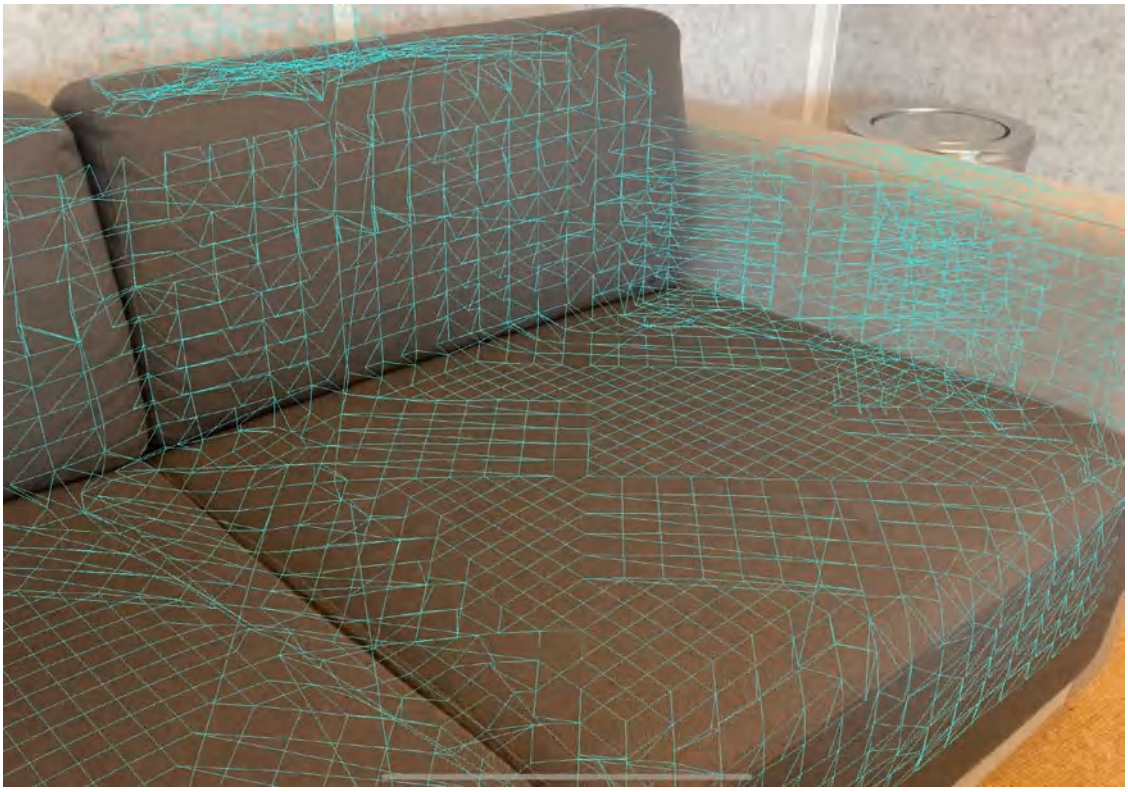
在编辑器的层级列表中右键 **创建 > XR > Meshing**，创建网格化代理节点。



在 cc.ARMeshing 的 Mesh Visualizer 属性中选择需要展示的网格模型。



即可直接在运行时将现实环境的表面网格化。



特征库和行为库

当前可以添加特征和行为如下：

特征库

特征项	特征属性	描述
Plane Direction		设置此平面需要满足给定的朝向。平面可以选

		择指定的朝向(水平，垂直或其他)。
Plane Size		设置此平面必须匹配给定的尺寸。设置时可以选择限制平面的最小/最大尺寸范围。
Image Source	Image	设置图像追踪所依赖的图像资源。
	Enable Physical Size	开启后，识别到图像资源后，将默认以此物理尺寸作为现实图像的尺寸，而无需根据深度特征计算图像大小，可以加快识别的速度。
	Image Physical Size	标定图像的物理尺寸限制(米为单位)。当改动宽或高任意一项数值时，另一项都会根据像素比自动计算数值。

行为库

行为项	行为属性	描述
Display		如果自动化行为编辑组件的所有特征与真实世界实体匹配成功，则激活子对象;否则禁用子对象。
	Display Children Node	开启后，默认展示 Tracking 节点下的子节点对象。

	Stop Tracking	满足当前节点的条件设定时，关闭此节点的 AR 追踪。
	Reset When Loss	追踪丢失时，子节点行为是否重置。
Align		该代理节点的位置与真实世界实体的位置的对齐关系。
	Towards	子物体摆放朝向。如果设置为 Local_Up，则直接使用子物体的姿态。如果设置为 World_Up，子物体 Y 轴将始终与世界坐标上方向对齐。
	Face to Camera	开启后，子物体 Z 轴朝向 AR Camera 的方向。
	Match Tracking Update	当此节点匹配的真实世界数据更新时，布局和对齐效果也跟随刷新。
Surface Overlay		满足条件后用指定的预置体覆盖原来的可视化效果。
	Surface Offset	设置覆盖在平面上的表面位置的偏移值。
	Replace Visualizer	创建后关闭并取代追踪可视化的效果。
Adaptive Scale		根据匹配的 AR 对象的边界来放缩子物体。
	Max Scale	比例调整的最大上限。

	Match Tracking Update	Scale 的行为是否跟随追踪而不断刷新。
Track Event		在跟踪特征匹配期间调用的事件集。
	on Track Success	追踪成功时调用的事件。
	on Track Refresh	追踪信息刷新时调用的事件。
	on Track Loss	追踪丢失时调用的事件。
	on Track Timeout	追踪超时时调用的事件。可以设置保持追踪检查的时间，此时间范围内没有成功匹配追踪数据会触发追踪失败的事件。

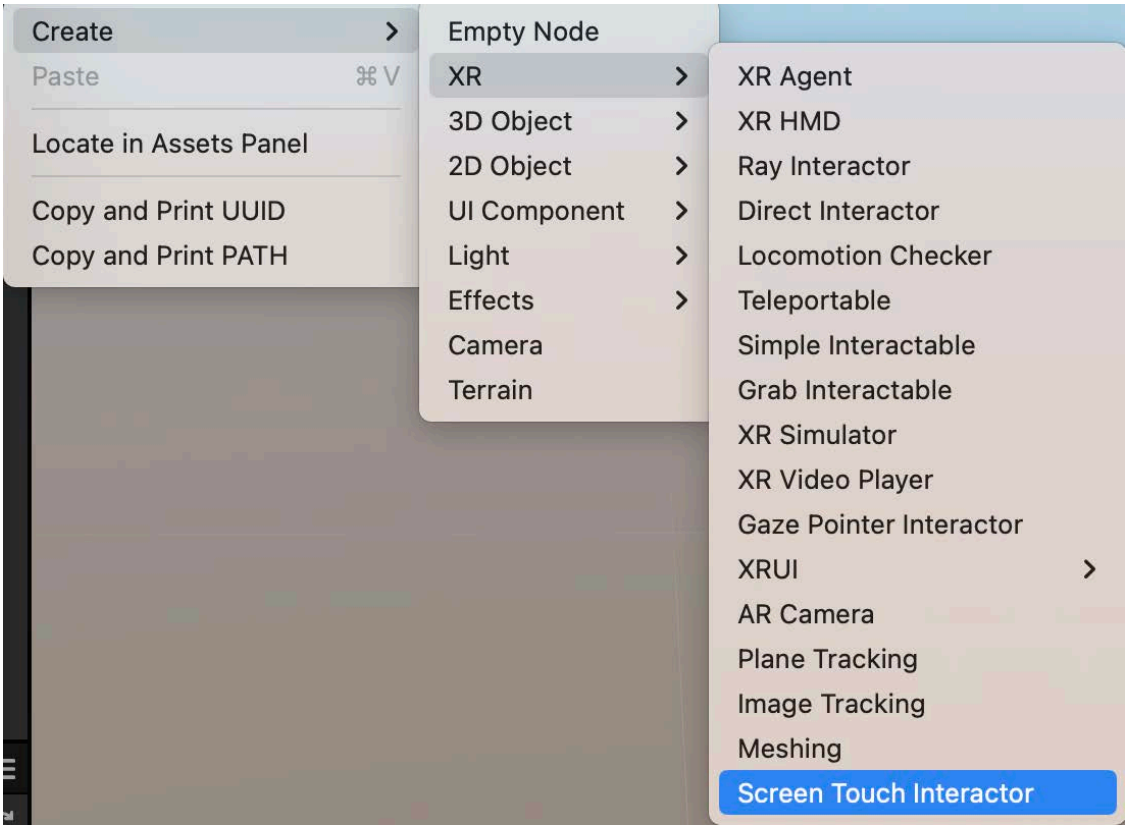
AR交互

AR 交互主要由 cc.ScreenTouchInteractor 组件驱动，该组件将触摸事件转换为点击、拖拽和捏合等手势，交互器将这些手势传递给可以交互的虚拟交互物，完成手势对应触发的行为。

手势交互

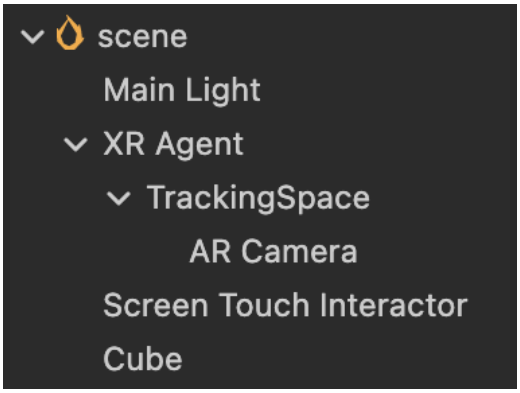
AR 手势交互器组件将屏幕触摸转换为手势。Cocos 输入系统将手势信号传递给交互物，然后交互物响应手势事件发生变换行为。交互物能发生交互行为的前提是必须绑定 cc.Selectable 组件，关于此组件的属性描述详见交互组件 Selectable。

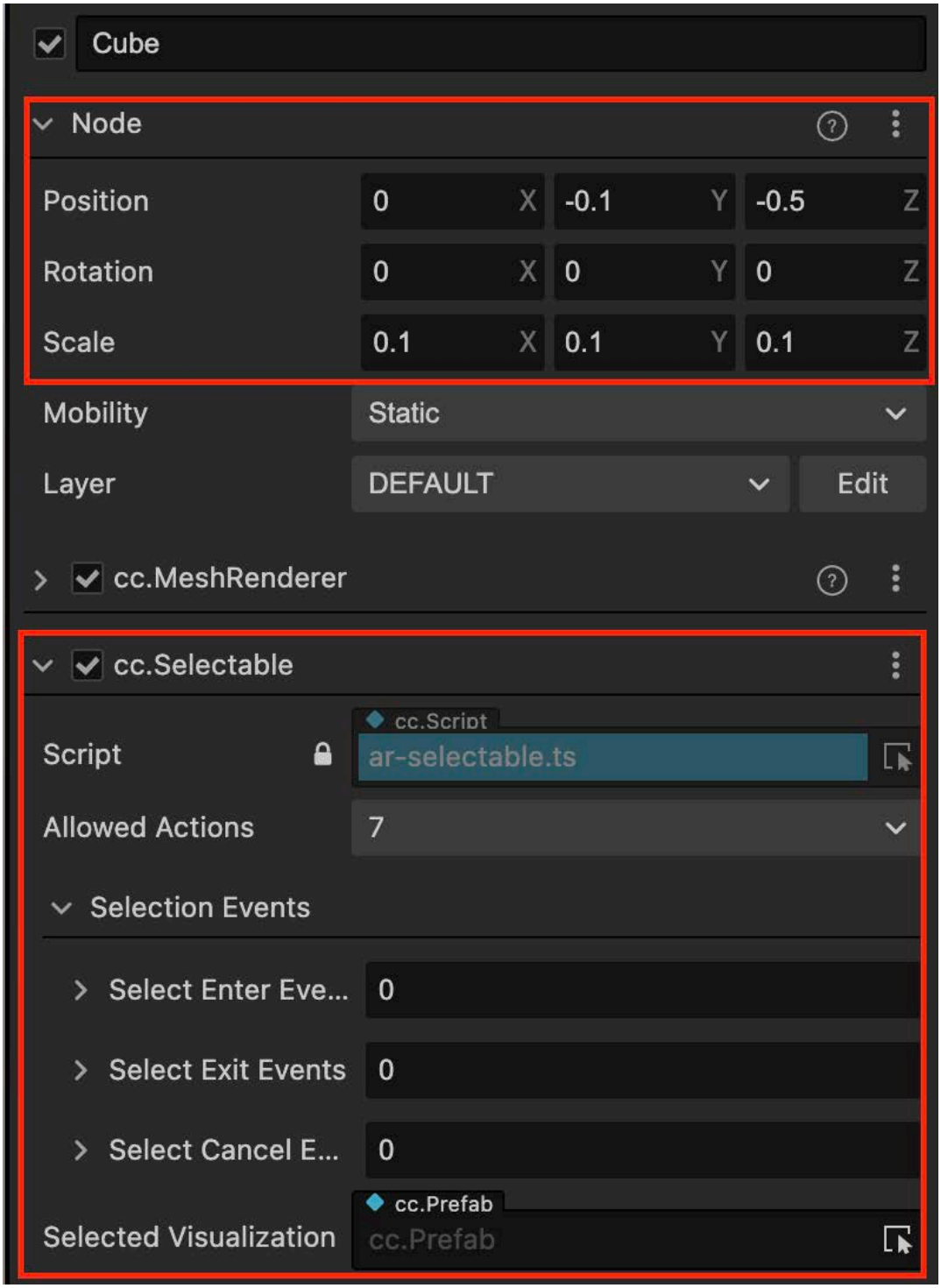
想要使用屏幕手势交互器，在层级管理器中右键创建 **XR > Screen Touch Interactor** 。



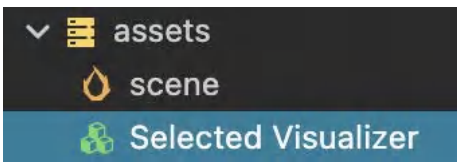
随意创建一个 3D 物体（以 Cube 为例）。

修改 Cube 的 Scale 属性为（0.1，0.1，0.1）既实际大小为 1000cm^3 ，修改 Position 属性为（0，-0.1，-0.5）即位于空间远点处 50cm 远且靠下 10cm 的位置，并添加组件 **XR > Interaction > Selectable**。

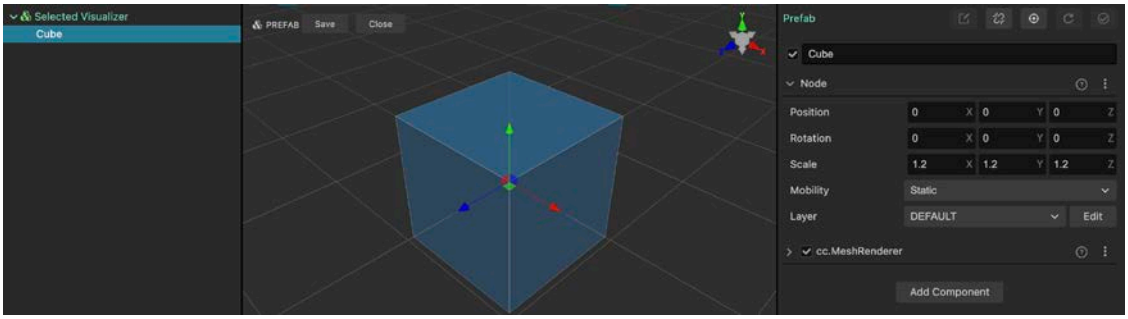




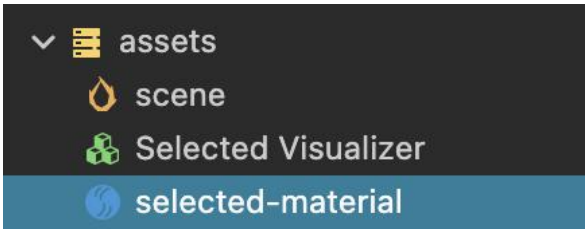
下面创建选中效果，在资源文件夹中创建一个预置体，命名为 Selected Visualizer。



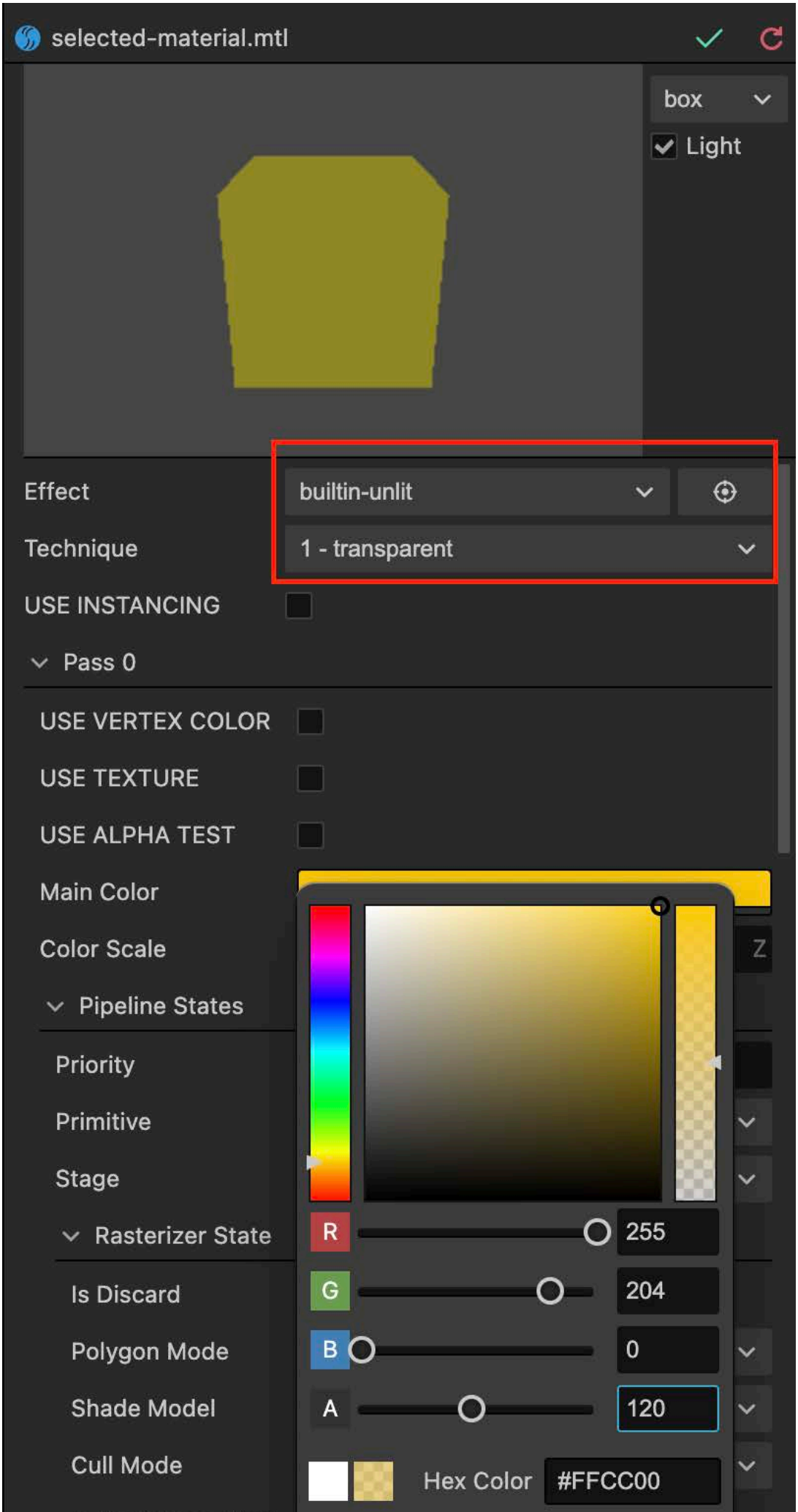
在预置体根节点下创建一个同样的 Cube 对象，Scale 大小设置为基于父节点的 1.2 倍。



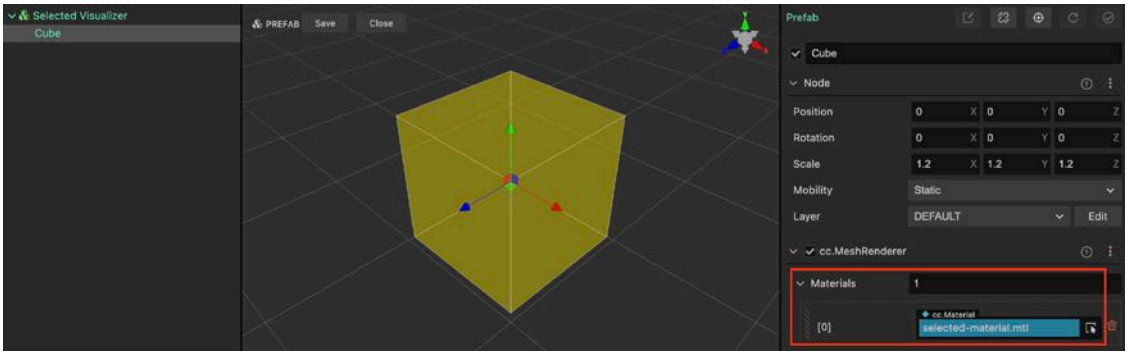
创建一个新的材质，突出表现选中态的效果。



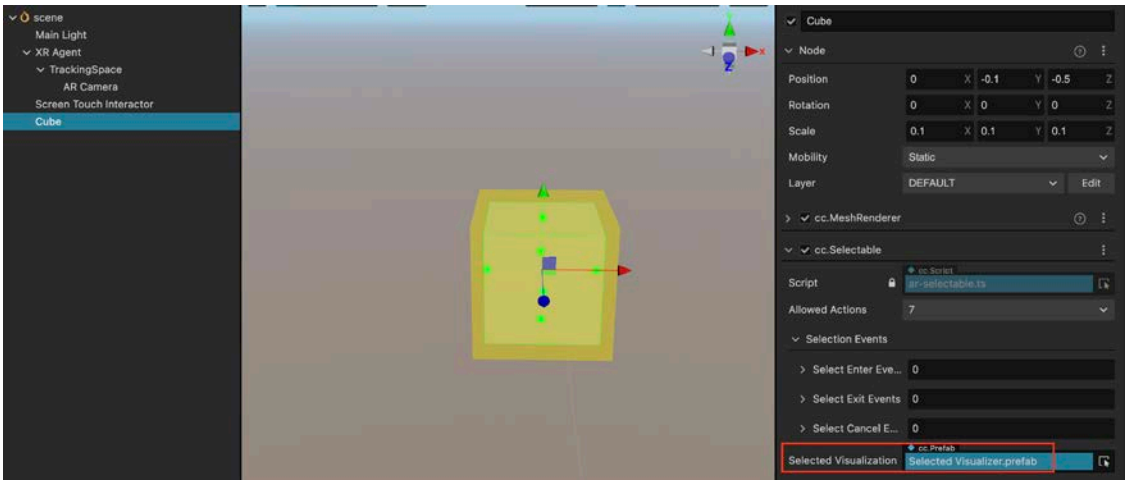
调整材质效果，建议 Effect 选择 builtin-unlit，Technique 选择 1-transparent。


selected-material.mtl
box
Light
Effect
builtin-unlit
Technique
1 - transparent
USE INSTANCING
Pass 0
USE VERTEX COLOR
USE TEXTURE
USE ALPHA TEST
Main Color
Color Scale
Pipeline States
Priority
Primitive
Stage
Rasterizer State
Is Discard
Polygon Mode
Shade Model
Cull Mode
R
255
G
204
B
0
A
120
Hex Color
#FFCC00

材质创建完毕后，应用到预置体中 Cube 的 cc.MeshRenderer 中，即可完成选中效果的创建。



最后，将预置体应用到 cc.Selectable 的 Selected Visualization 属性中。



运行时效果如下，可以结合手势来移动、旋转和放缩虚拟物体。

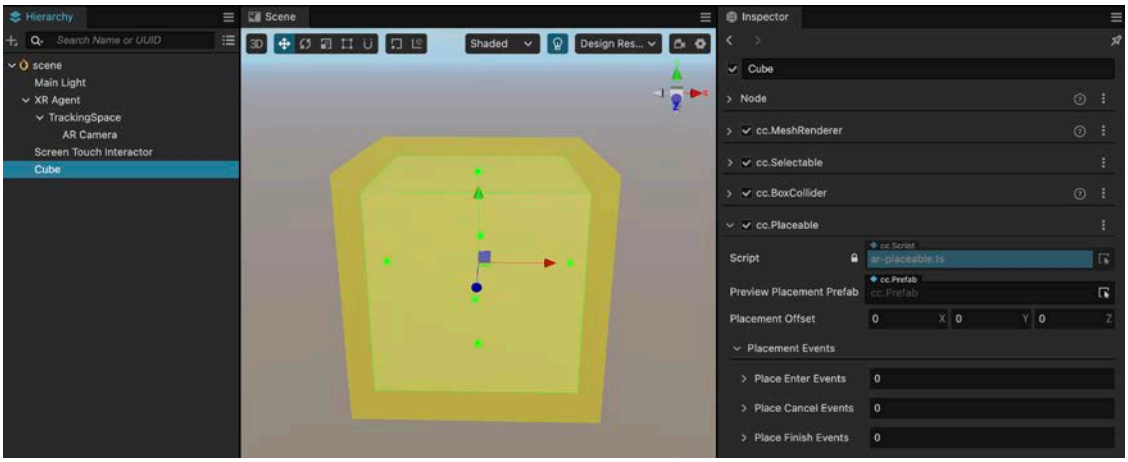
放置

使用屏幕交互器时，会启用设备 AR Hit Test 能力，根据屏幕触碰位置坐标转换到摄像机

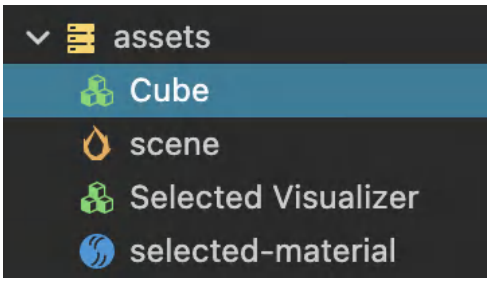
使用 Ray Cast 与 AR Plane 发生碰撞计算，来获取碰撞点的位置，最终在平面的此坐标上放置虚拟对象。能够被放置的预置体对象必须要挂载 cc.Placeable 组件。

以上述场景中制作的 Selectable 对象为例，以下对其赋予被放置交互能力。

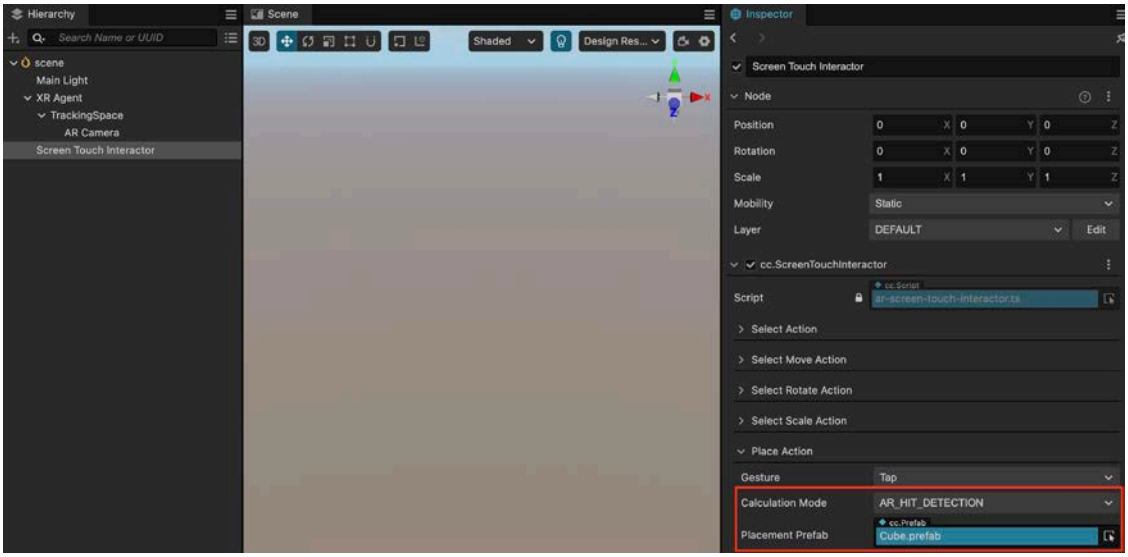
选中场景中的 Cube 对象，为其添加组件 **XR > Interaction > Placeable**。



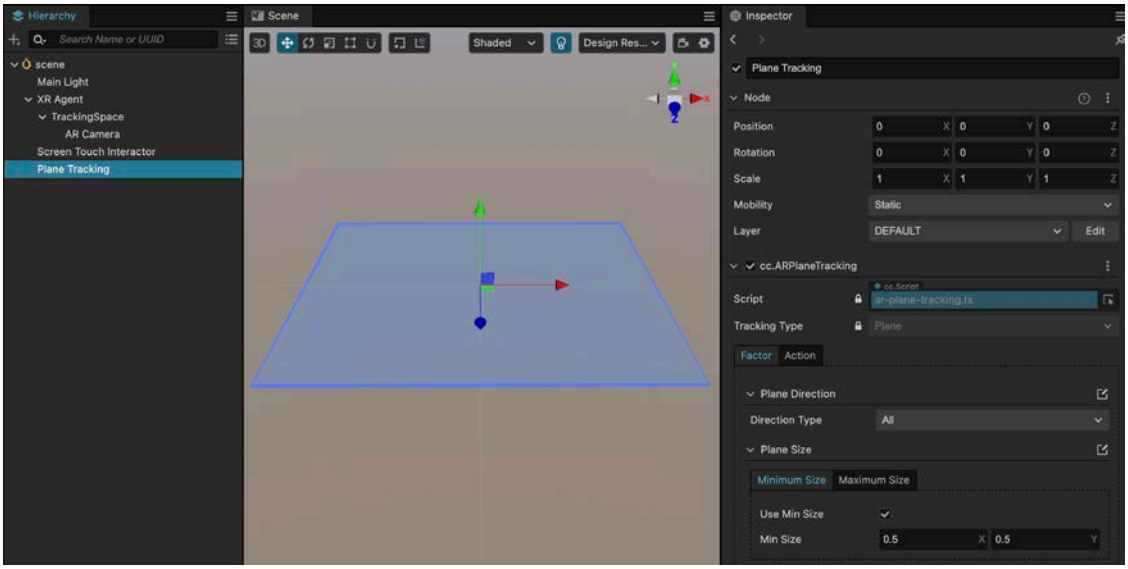
将此场景节点拖入资源管理器生成一份预置体，并删除场景中的此 Cube 对象。



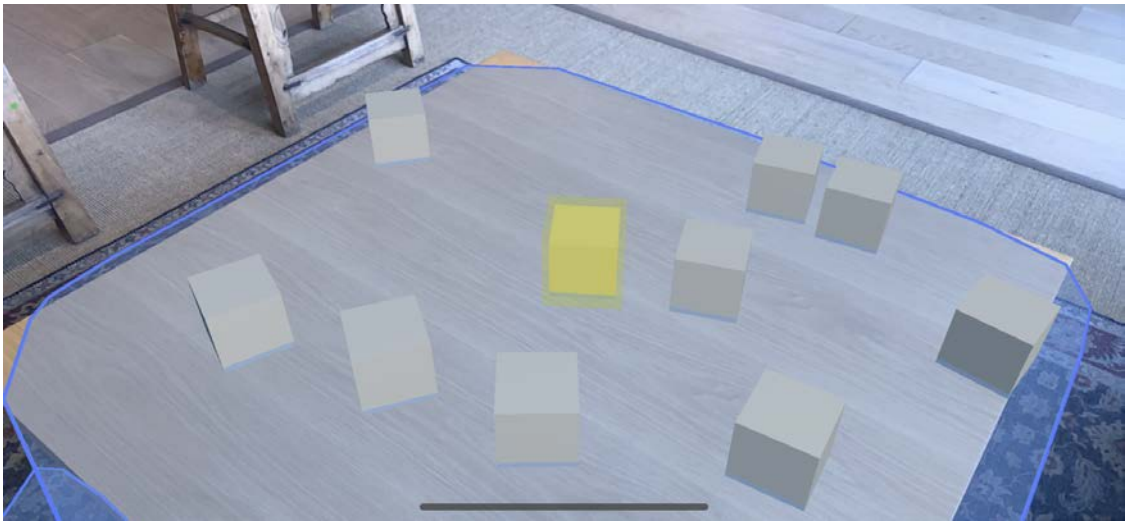
将刚生成的 Cube 预置体引用到 **Screen Touch Interactor > Place Action > Placement Prefab** 属性中，**Calculation Mode** 选择 **AR_HIT_DETECTION**。



放置对象的位置计算需要依赖于 AR Plane，所以还需创建一个 Plane Tracking 节点来请求设备激活 AR SDK 的平面识别能力。在编辑器的层级管理器中右键 **创建 > XR > Plane Tracking**，创建平面代理节点。



所有工作都完成后，即可打包发布，在运行时查看放置效果。



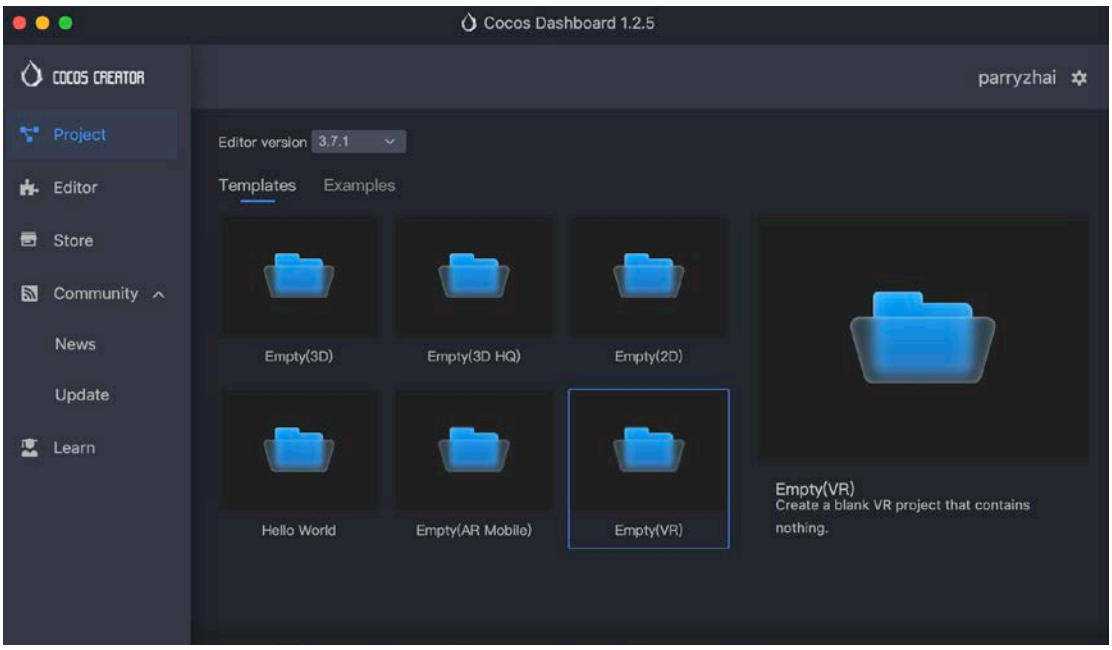
快速使用指南

VR项目创建

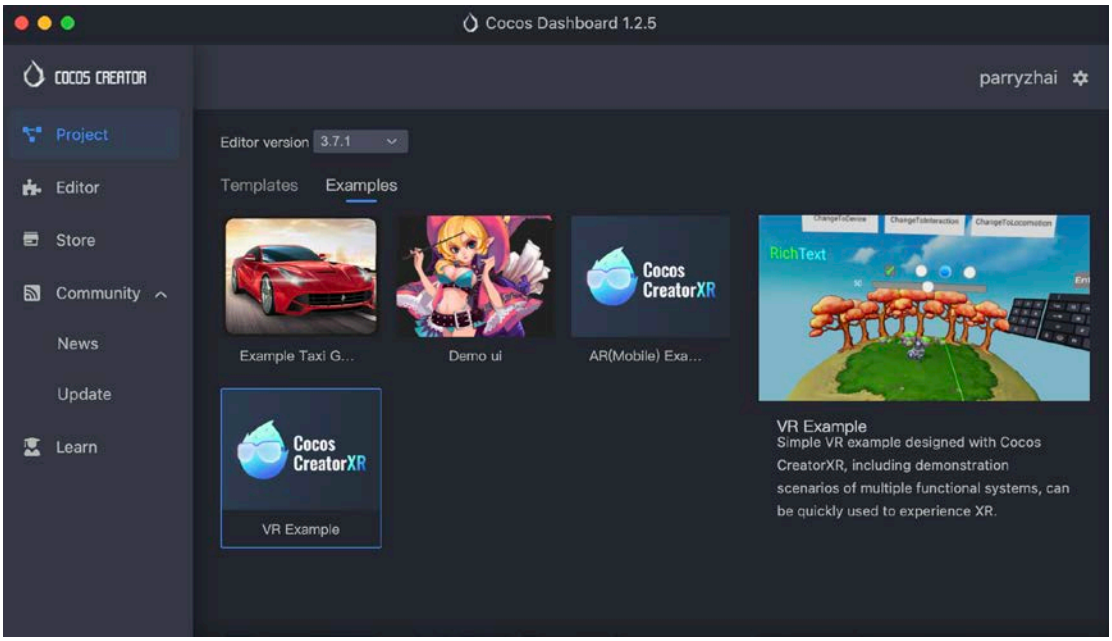
Cocos CreatorXR 支持用户使用一下几种方式快速创建 VR 项目。

注意：创建 XR 项目时务必保证编辑器版本 >= 3.6.1。

- 使用模板新建 VR 项目：在 Cocos Dashboard 中新建项目时，选择 v3.6.1 及以上的编辑器（若需要体验完整功能，引擎请选择 v3.7.1 及以上的版本），选择 Empty(VR) 模板创建。



- 使用案例体验学习创建 VR 项目：在 Cocos Dashboard 中新建项目，选择 v3.6.1 及以上的编辑器（若需要体验完整功能，引擎请选择 v3.7.1 及以上的版本），选择 VR 案例创建。

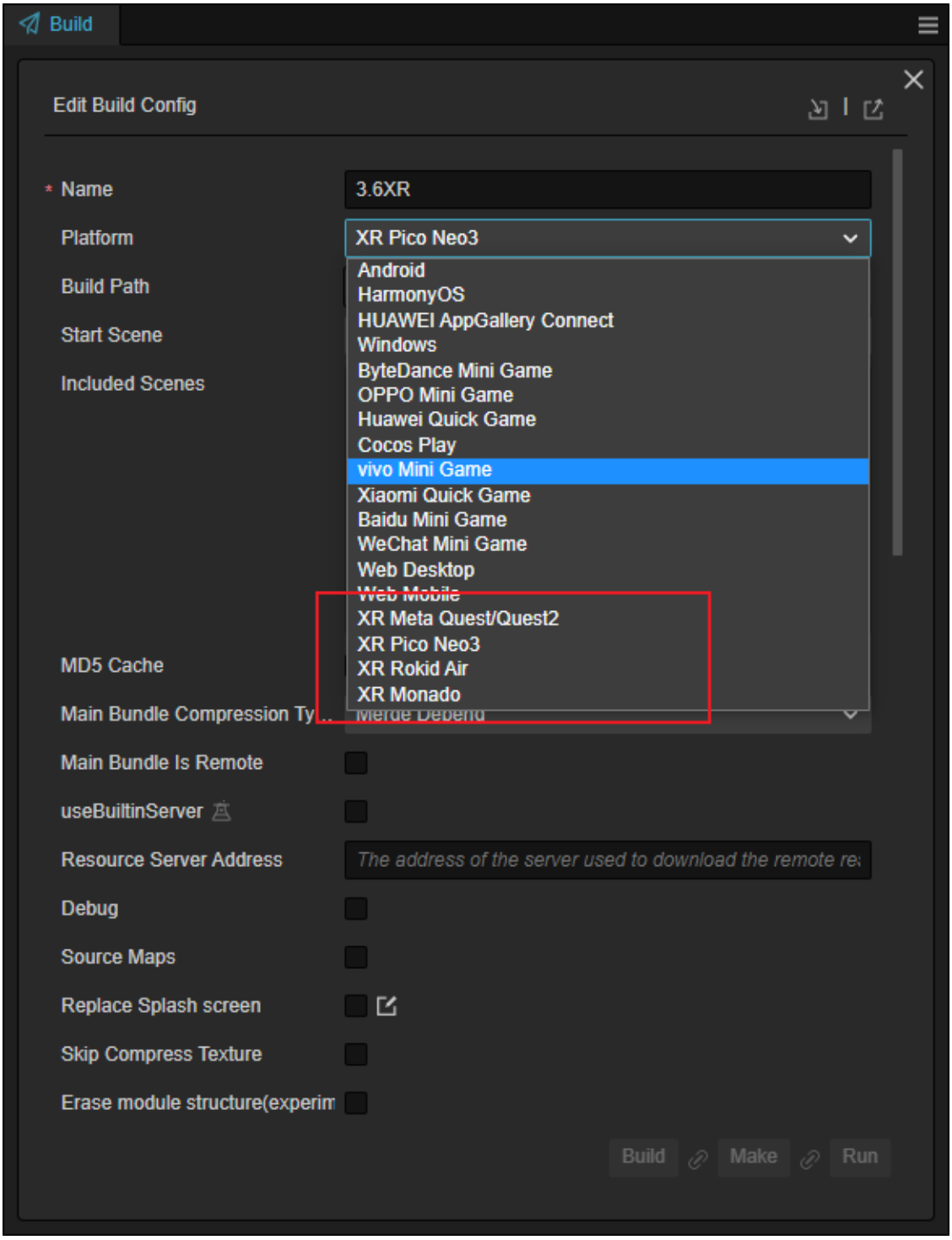


- 基于空项目或已有项目添加 XR 扩展：在 Dashboard 中的商场页面搜索 xr-plugin 下载安装并应用至项目或在 Cocos Creator 中的 **扩展->商城** 中下载安

装扩展至项目（不推荐安装到全局）。

VR项目构建与发布

当项目开发完成后，需要将项目打包到对应的平台上，在菜单栏中选中 **项目 -> 构建发布** 打开 **构建发布面板**，通过 **发布平台** 属性下的下拉框中选择目标平台：



注意：不同平台需要在 扩展安装 启用对应的扩展才可以选择。

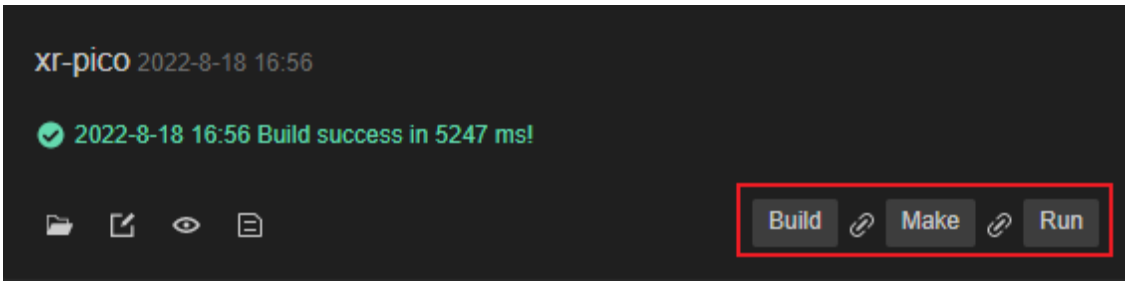
属性

构建通用属性可以参考：构建选项介绍。

目前支持的 VR 设备都采用了安卓系统，开发者需搭建对应的开发环境，详情可参考 安装配置原生开发环境。

构建

之后在构建任务中根据需要选择 **构建**、**生成** 或 **运行** 即可。



发布

生成应用程序以后，可以通过 adb 命令或设备的传输文件功能将应用程序传输到目标设备上，之后运行则可以完成整个构建和发布过程。

AR项目创建

参照以下步骤完成对项目的 AR 相关特性配置。

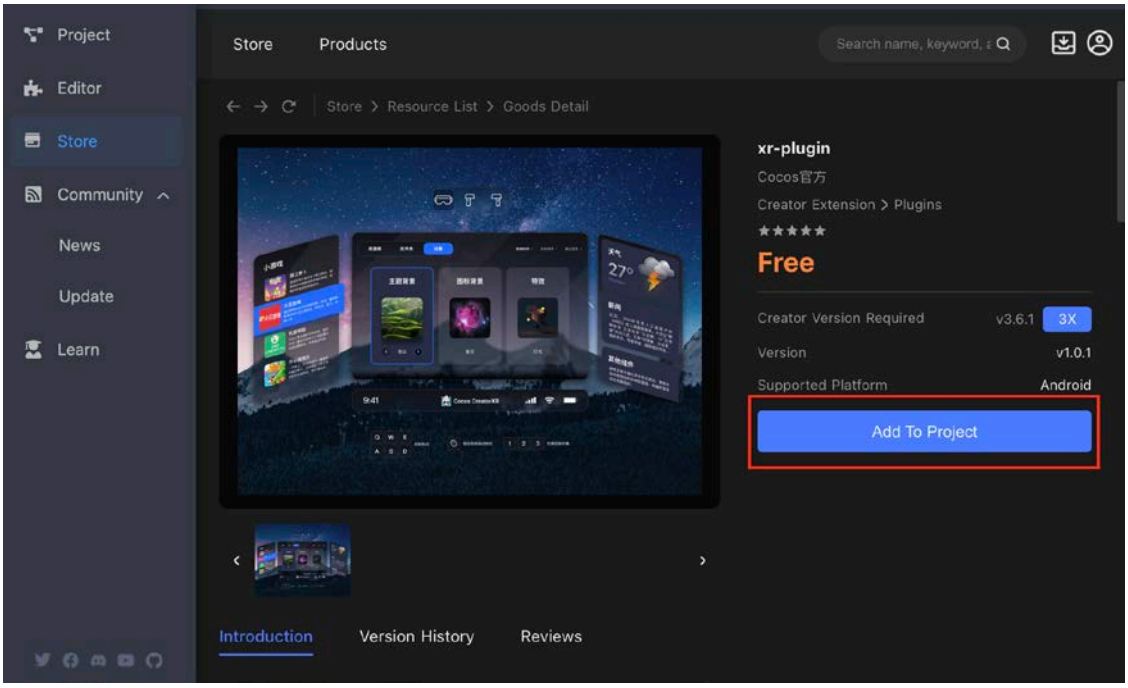
以下提供三种方法，可以**任选一种**来配置扩展或直接打开内置 AR 项目。

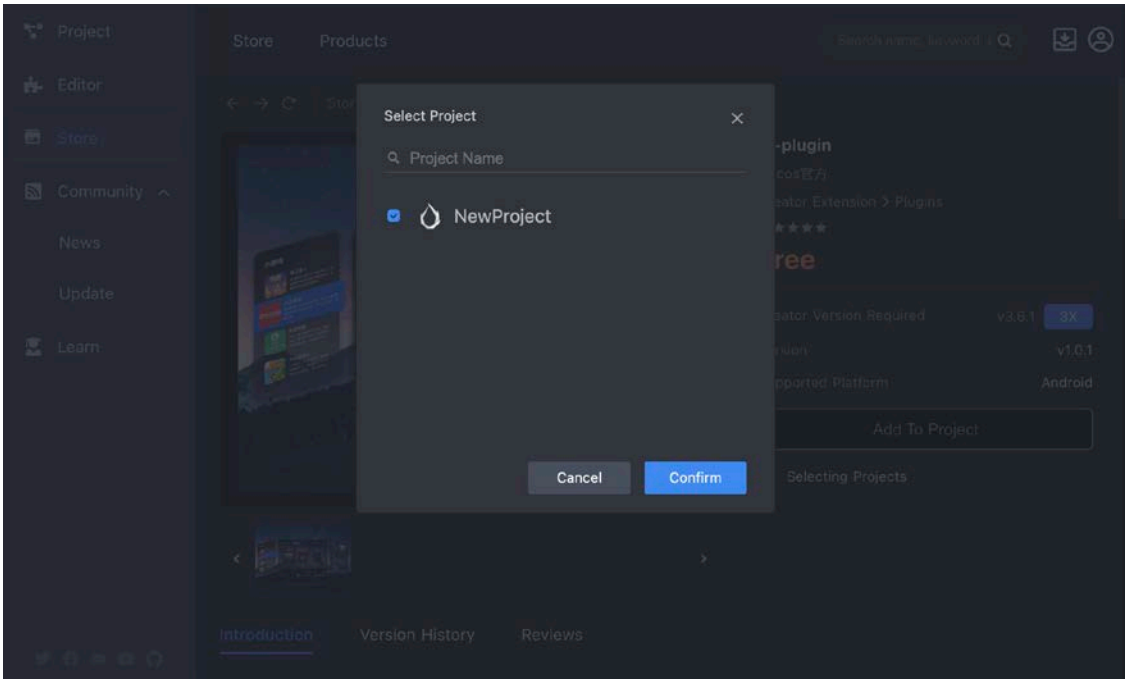
将xr-plugin应用到项目

在 cocos store 中搜索 **xr-plugin**，获取扩展并安装，具体安装说明请参考说明。

安装完毕后将扩展添加至对应工程。

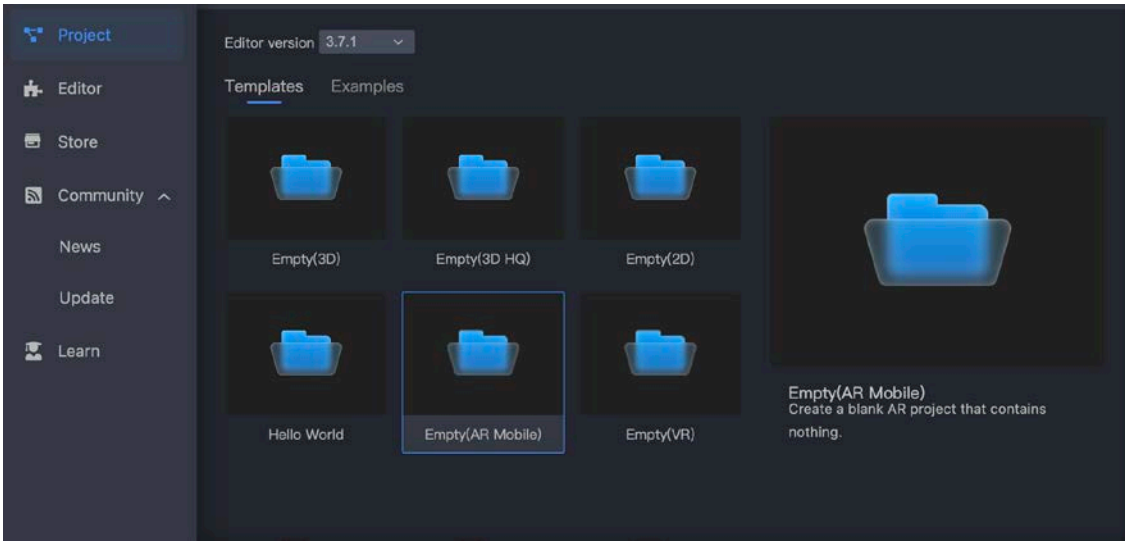
这种方式适合为存量 3D 项目做 XR 模式迁移。



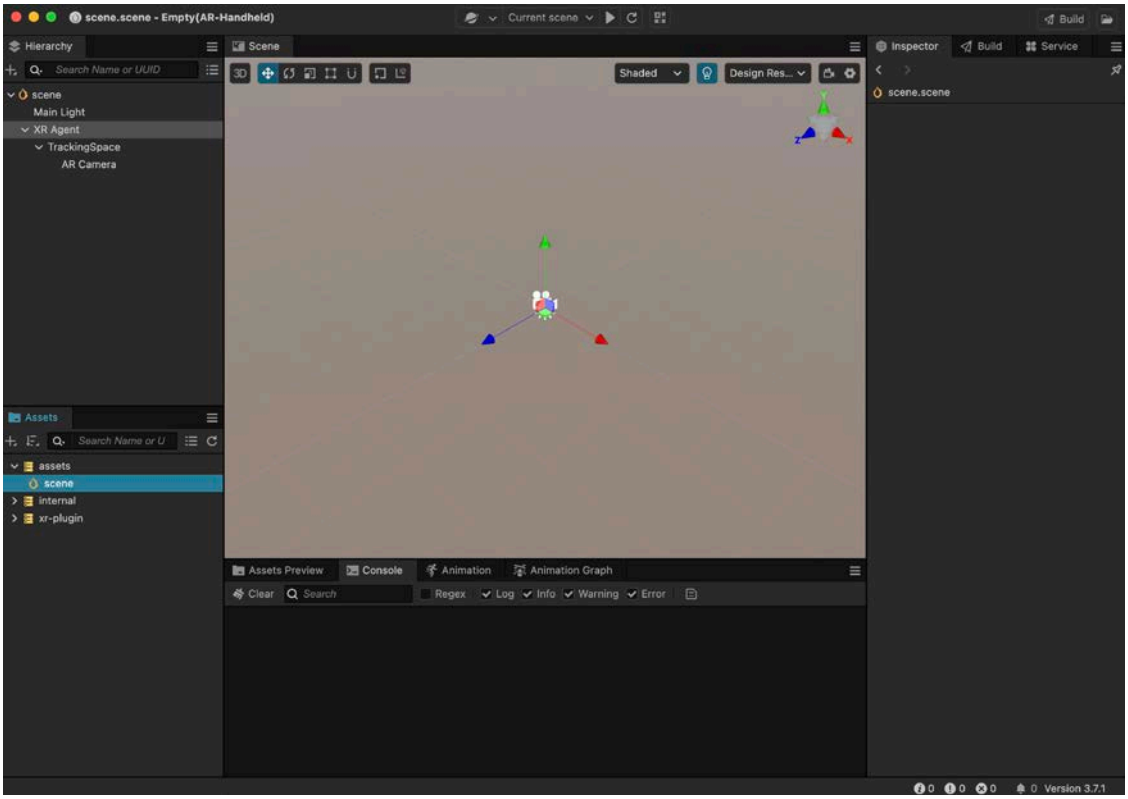


创建AR模板工程

Dashboard 中新建项目，编辑器版本选择 **3.7.1** 或更高，模板类别下选择 **Empty(AR Mobile)**进行创建。

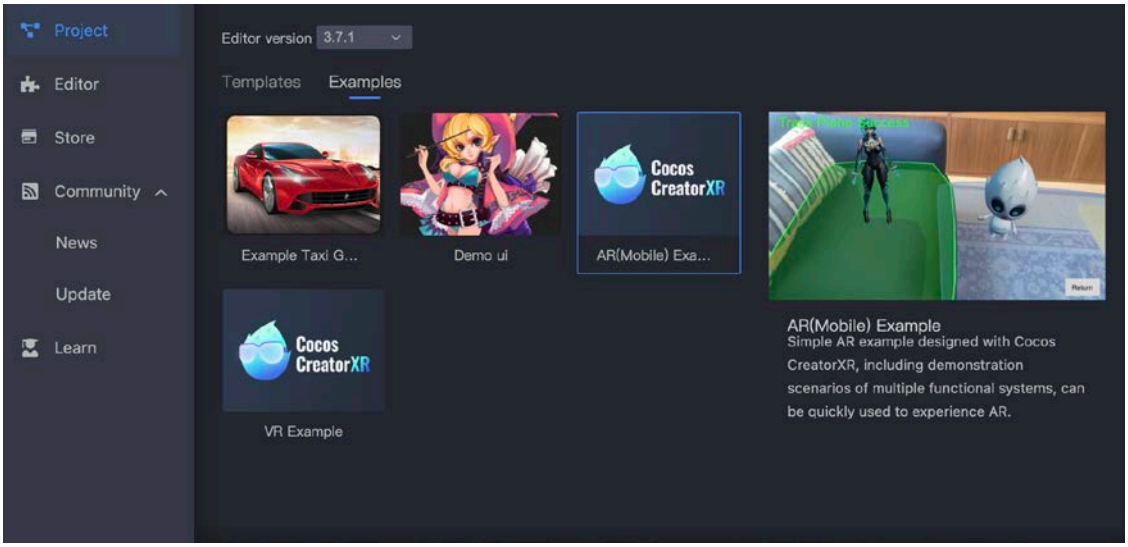


打开项目，进入 scene 场景。场景已经包含初始的 AR Camera 配置，可以直接进行功能开发。



创建AR案例

Dashboard 中新建项目，编辑器版本选择 **3.7.1** 或更高，案例类别下选择 **AR(移动端)案例**进行创建。



案例项目中包含当前版本扩展完整的有关 AR 特性功能的内容，可以直接进行构建打包体验。AR 应用的构建发布相关说明请查阅构建与发布。

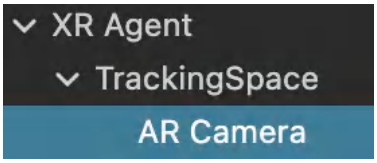
场景配置

若采用第一种**将 xr-plugin 应用到项目**的方式配置插件，还需进行以下步骤对普通的 3D 场景完成 AR 功能的基本配置。

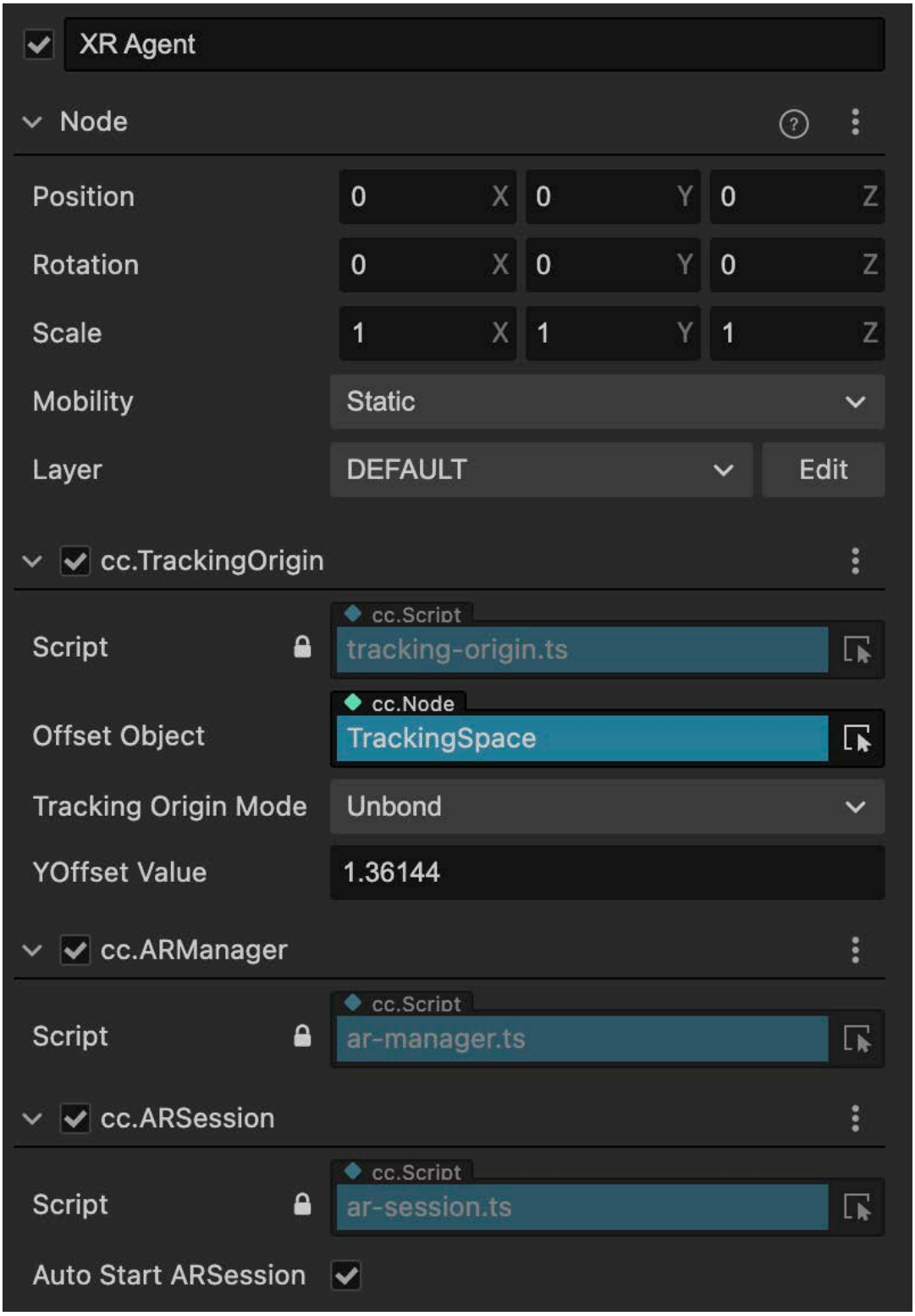
应用中的每个 AR 场景中必须要包含两个关键对象：**XR Agent** 和 **AR Camera**。

推荐以下两种方式任选一种来进行场景配置：

1. 右键单击“层次管理器”窗口，选择创建 **XR > XR Agent**。选中 XR Agent 节点，右键创建 Empty Node，并重命名为 TrackingSpace。选中 TrackingSpace 节点，右键创建 **XR > AR Camera**。



选中 XR Agent 节点，在属性检查器中点击“添加组件”，添加 **XR > AR Tracking > ARSession** 和 **XR > AR Tracking > ARManager**。



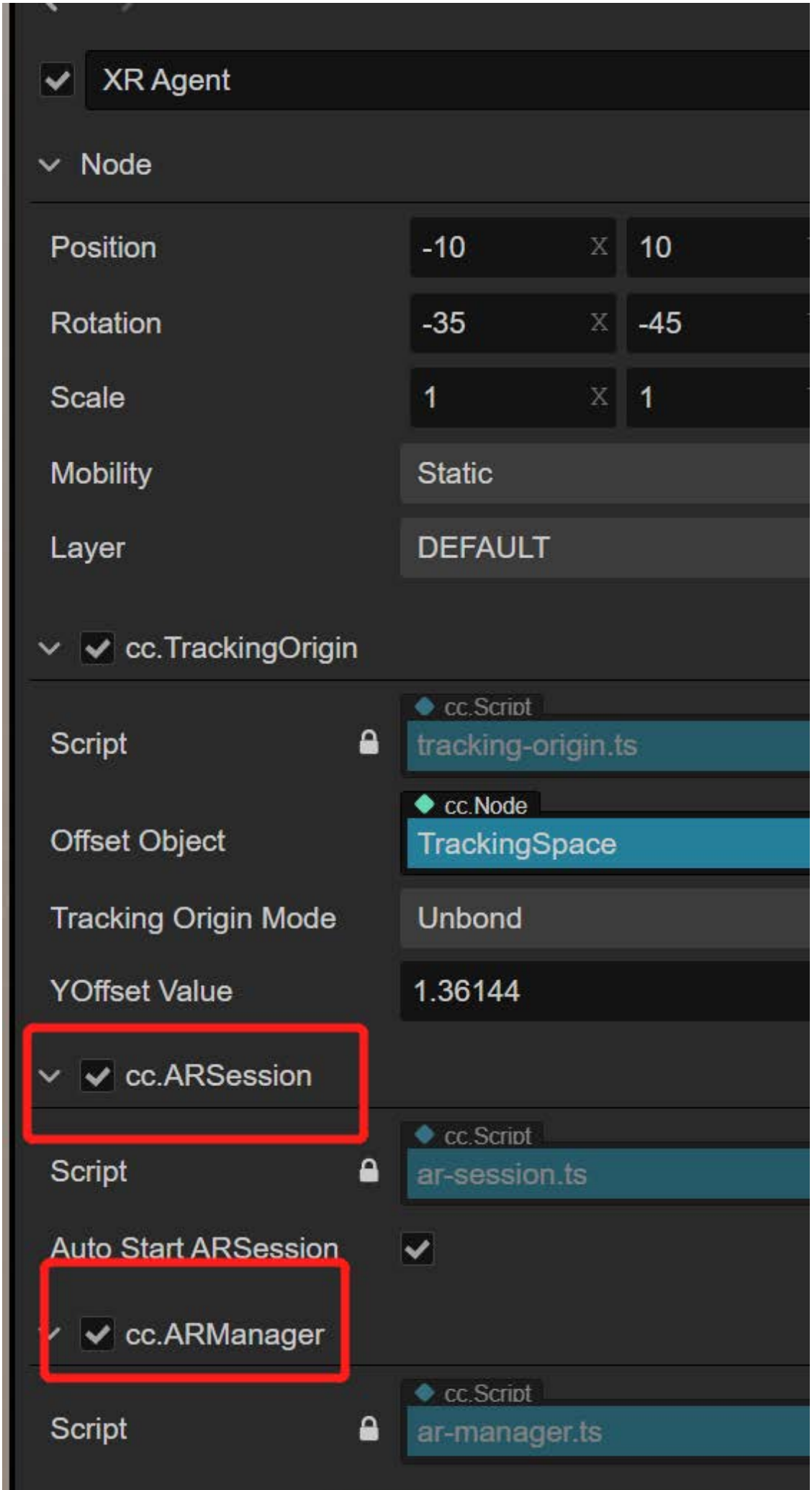
2. 对于空场景或现有项目，可以直接选中场景中主摄像机，右键选择**转为 AR Camera**，即可得到上述默认的结构。

XR Agent 和 AR Camera 及其组件在 AR 项目中扮演着重要的角色。要更详细地了解它们，请分别查阅设备映射和 AR 相机。

Spaces平台项目场景设置

新建空场景，将场景中 Main Camera 右键选择**转为 XR HMD**。

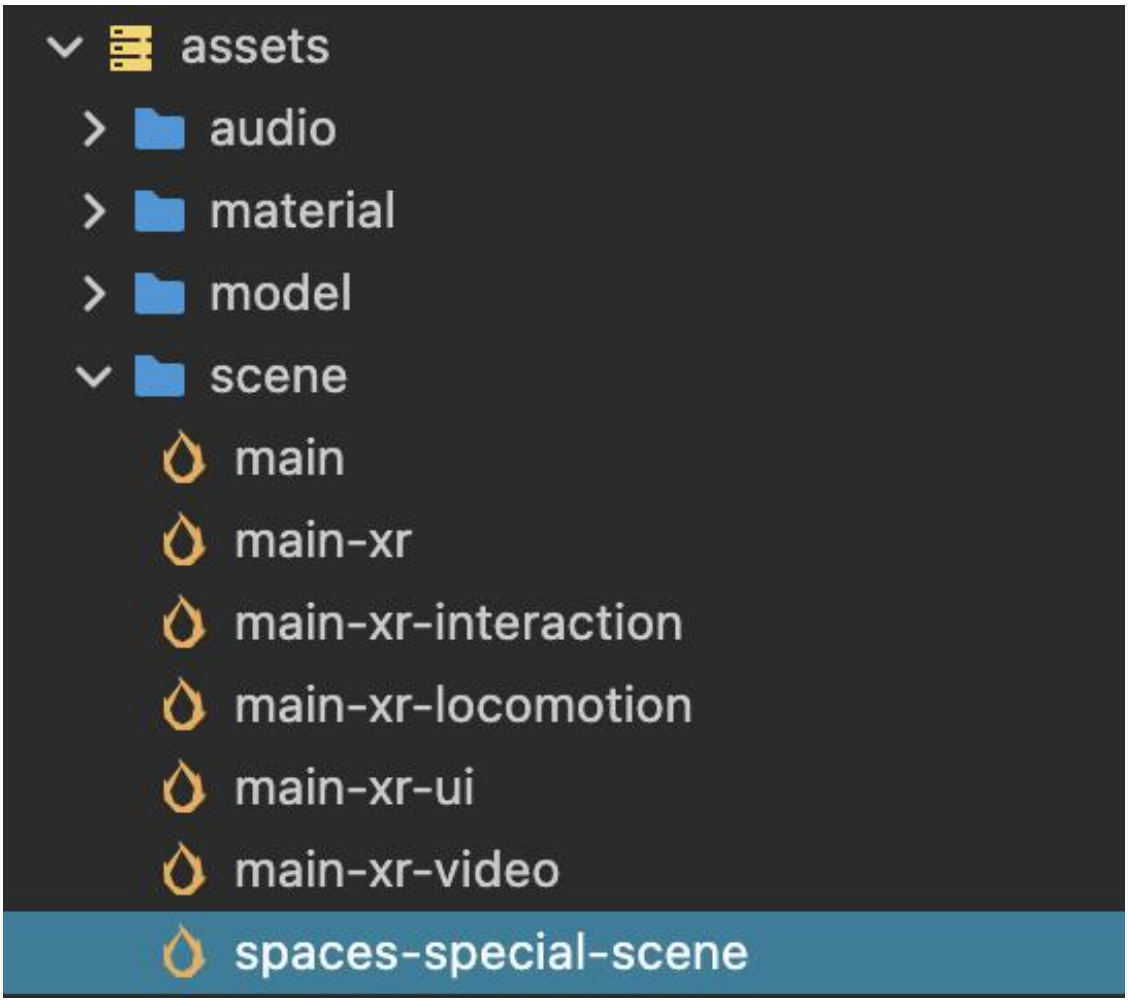
选中 XR Agent，点击 Add Component 添加 ARSession 和 ARManager 组件。

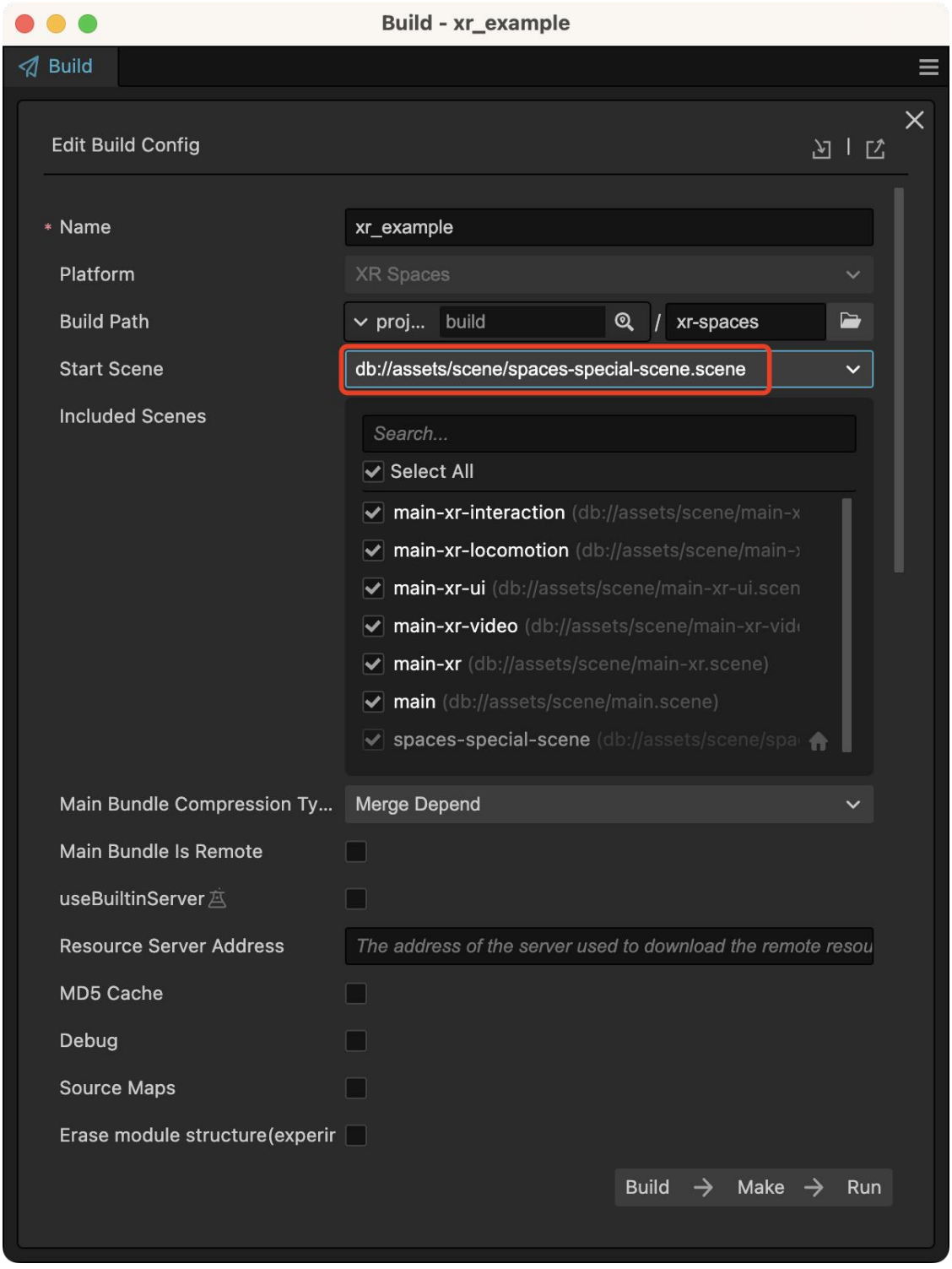

XR Agent
Node
Position
-10
X
10
Rotation
-35
X
-45
Scale
1
X
1
Mobility
Static
Layer
DEFAULT
cc.TrackingOrigin
Script
cc.Script
tracking-origin.ts
Offset Object
cc.Node
TrackingSpace
Tracking Origin Mode
Unbond
YOffset Value
1.36144
cc.ARSession
Script
cc.Script
ar-session.ts
Auto Start ARSession
cc.ARManager
Script
cc.Script
ar-manager.ts

可以参考平面追踪和图像追踪给应用做 AR 赋能。

功能开发完成后可直接打包发布。

在 Dashboard 的 VR 案例中，提供了一个简易的 Spaces 专用场景，可以直接打包应用并将此场景设为启动场景。



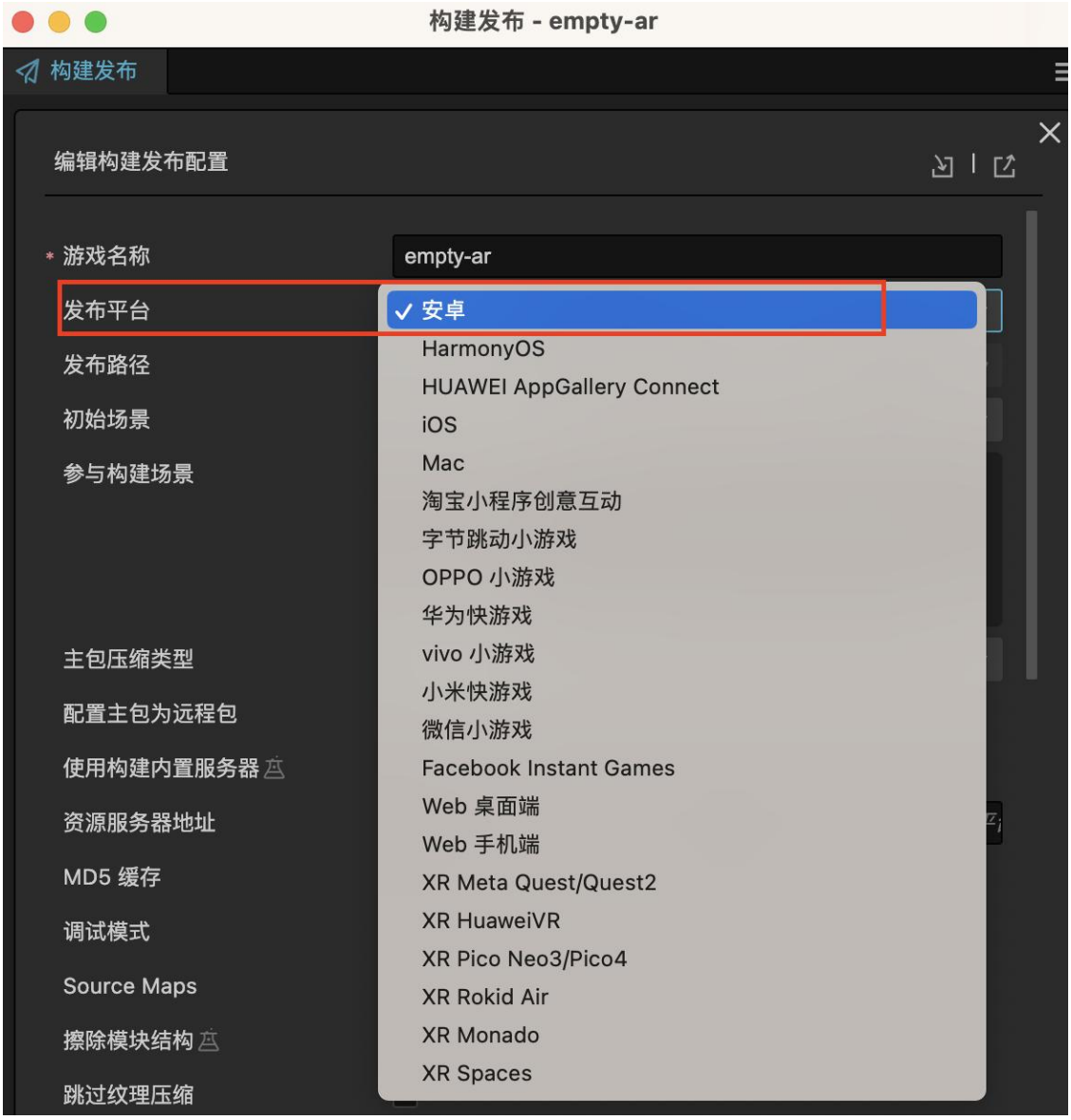


AR项目构建与发布

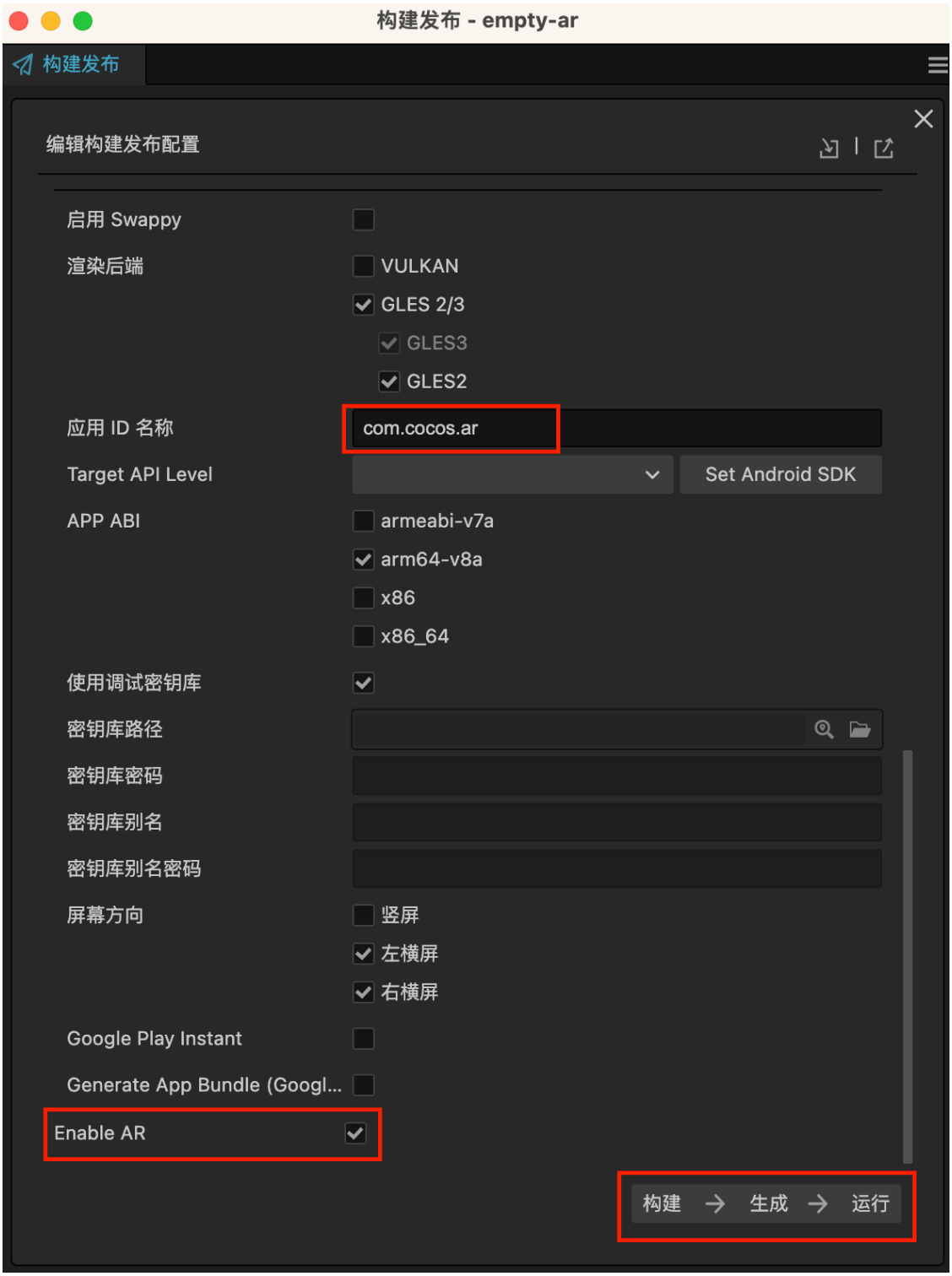
完成 AR 应用的项目设置并完成项目开发之后，即可打包 AR 应用。点击**菜单栏 > 项目 > 构建发布**。

ARCore、AREngine

针对于安卓和华为平台的手机发布 AR 应用，新建构建任务，平台选择**安卓**。



填写应用 ID 并勾选 **Enable AR**，连接好移动端设备后点击**构建>生成>运行**即可一键发布 AR 应用。

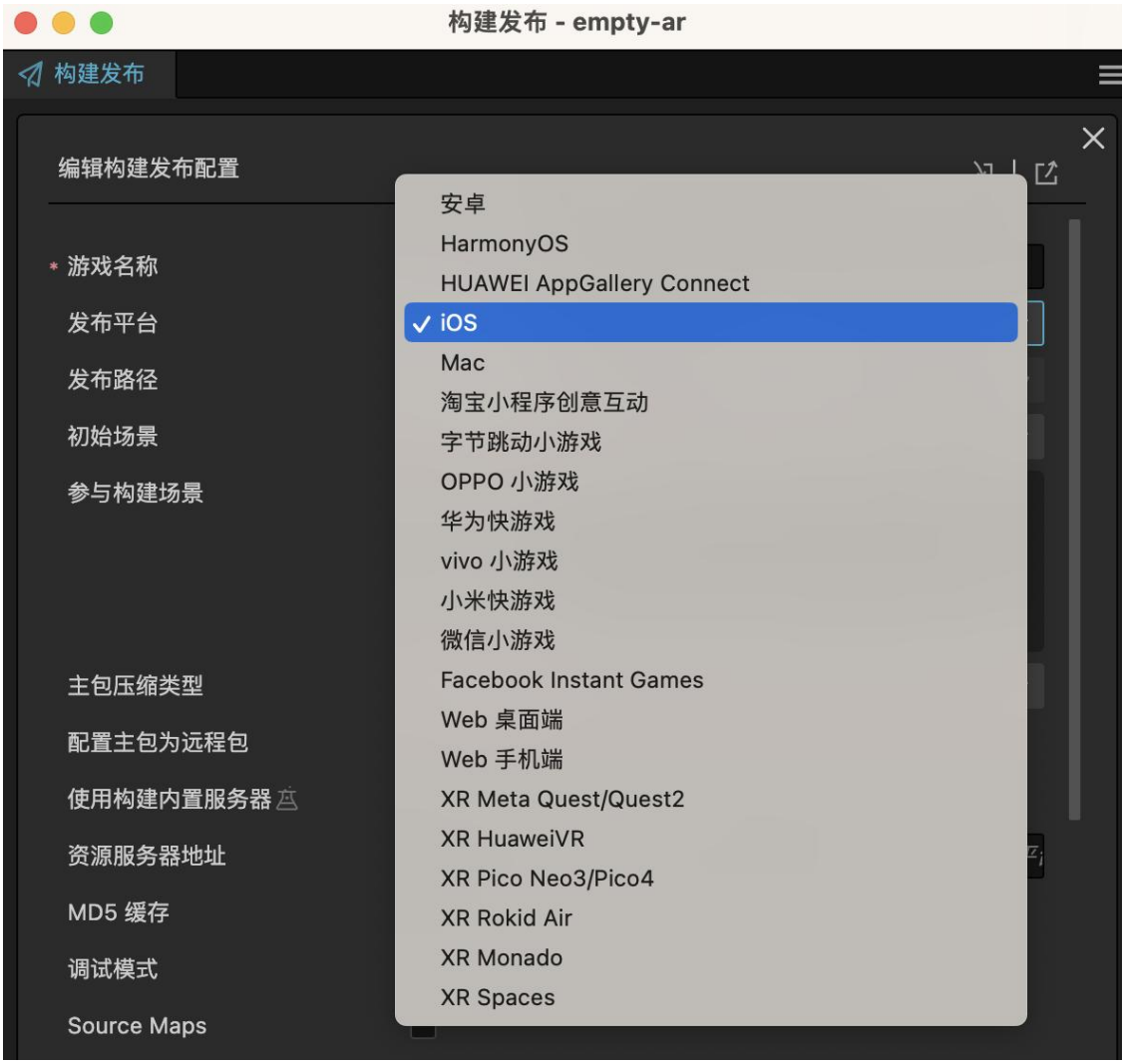


注：安卓平台 AR 应用的渲染后端不支持 VULKAN。

ARKit

iOS 发布需要的各项配置属性请参考 iOS 平台构建选项，需要在 Xcode 中配置好开发者账户。

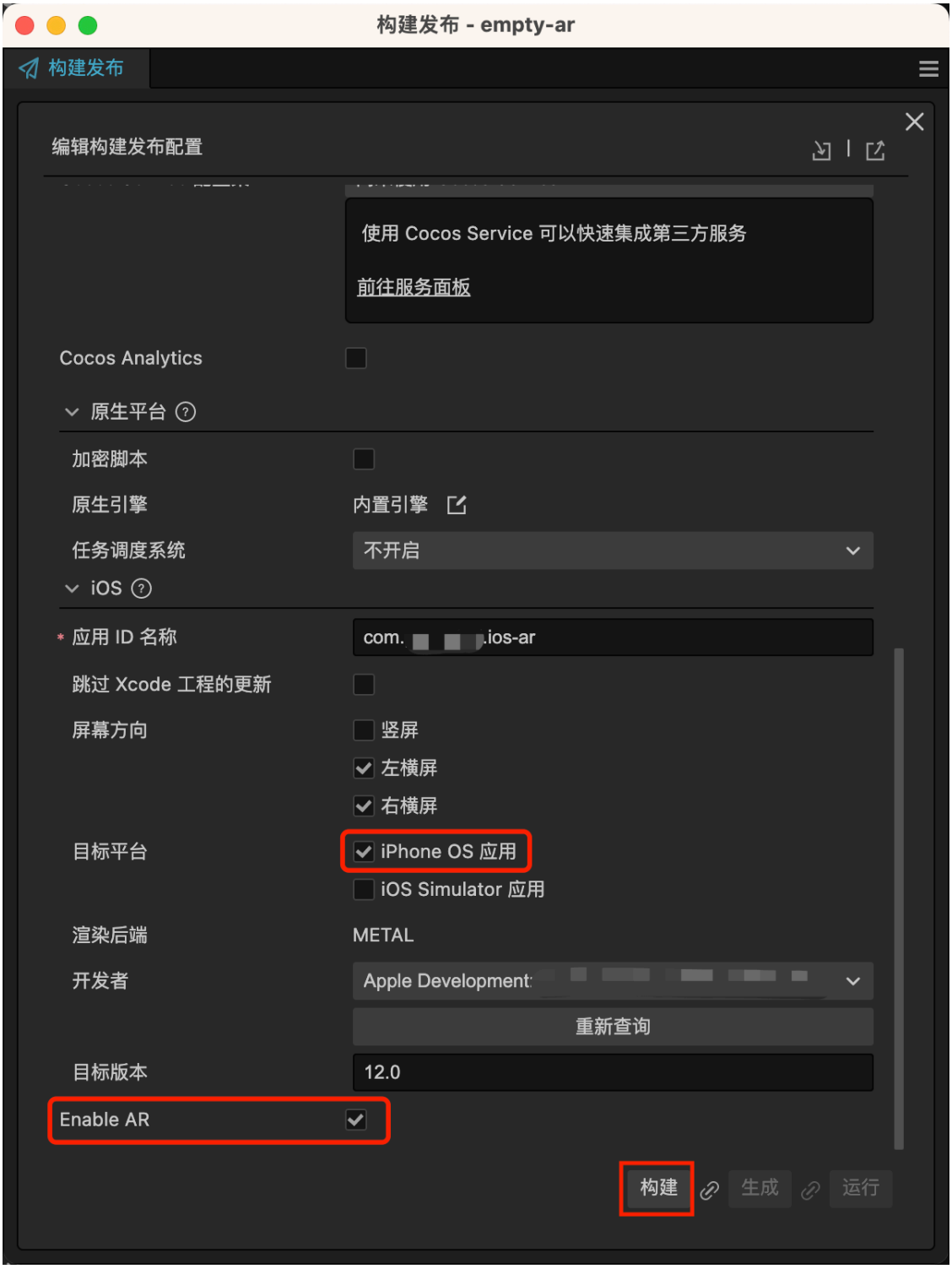
针对于 iOS 平台发布 AR 应用，新建构建任务，平台选择 **iOS**。



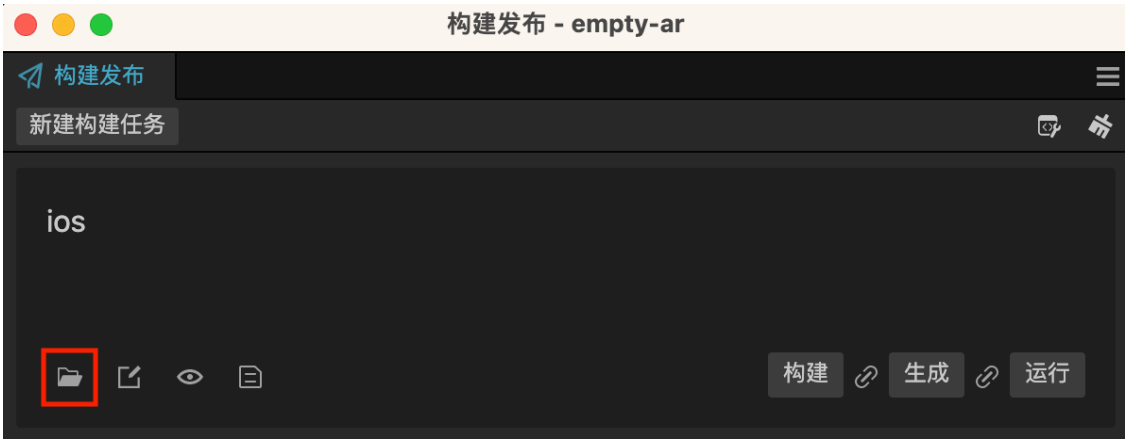
应用 ID 名称第二节建议使用 Xcode 配置的同名开发者账户名，目标平台选择 iPhone OS 应用，勾选 Enable AR。

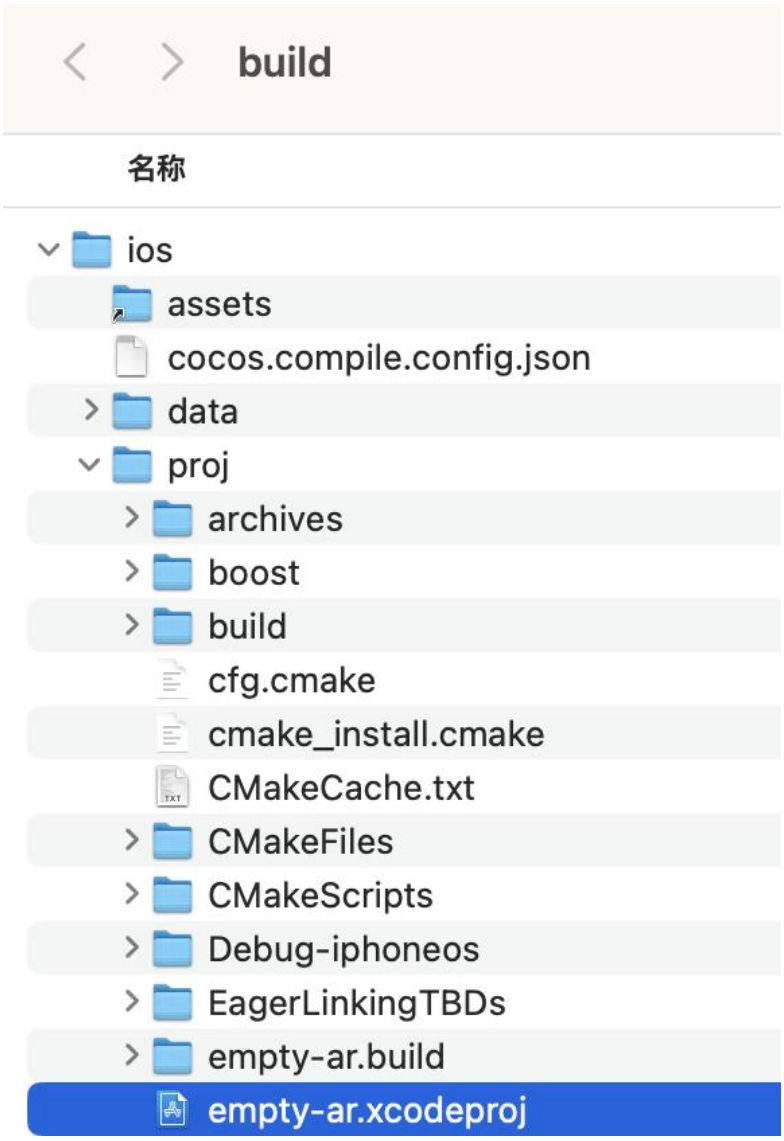
点击构建，生成 Xcode 工程。

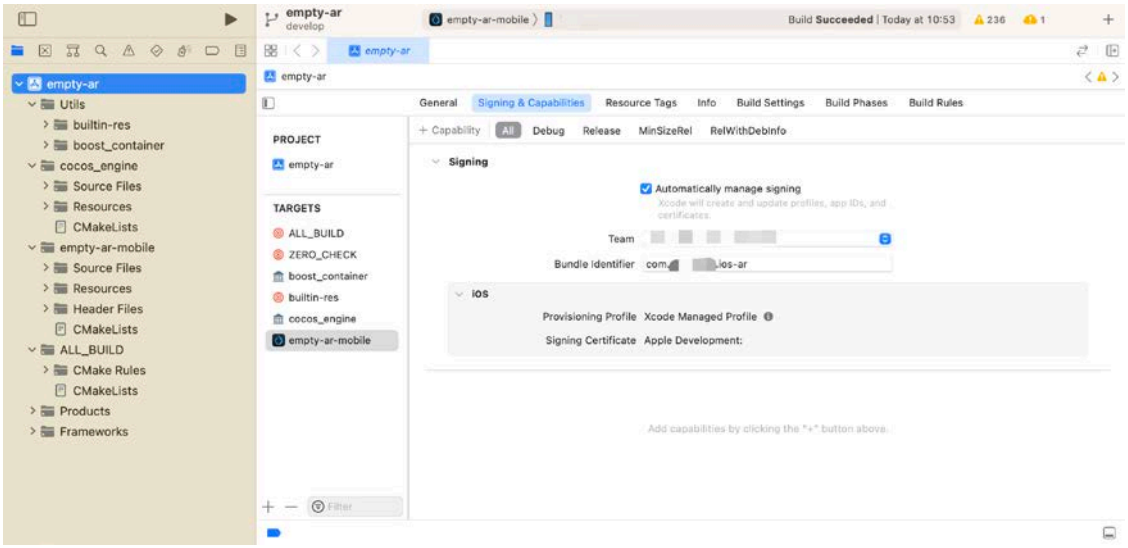
注：目前 Cocos 引擎对 iOS 应用暂时只支持构建工程，编译和运行需要转移至 Xcode 中进行。



构建完成后，找到生成的 xcodeproj 文件，使用 Xcode 打开，配置好签名和开发者团队连接好设备点击运行即可。

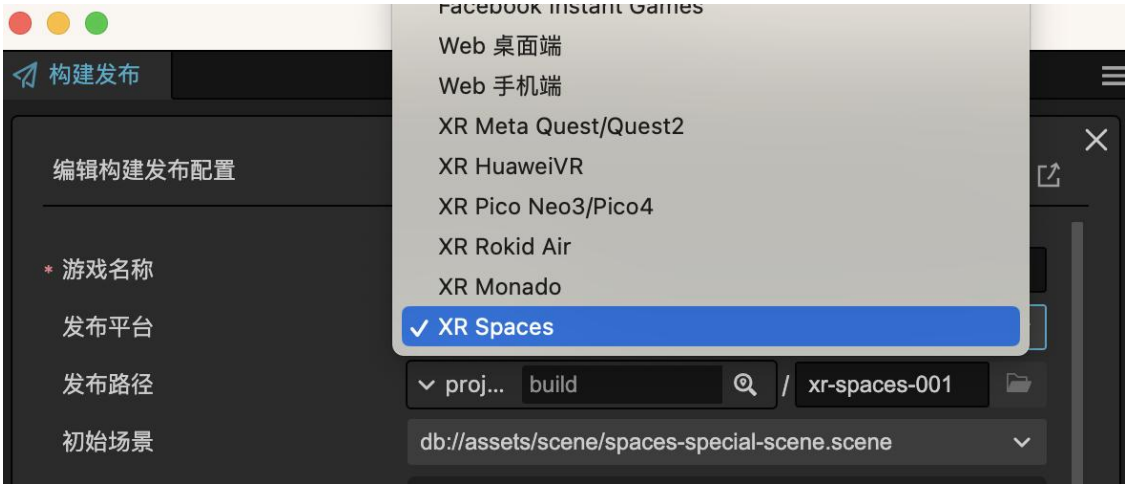

构建发布 - empty-ar
构建发布
新建构建任务
ios
构建
生成
运行


build
名称
ios
assets
cocos.compile.config.json
data
proj
archives
boost
build
cfg.cmake
cmake_install.cmake
CMakeCache.txt
CMakeFiles
CMakeScripts
Debug-iphoneos
EagerLinkingTBDs
empty-ar.build
empty-ar.xcodeproj



Spaces

针对于高通 Spaces 平台的设备发布 AR 应用，新建构建任务，平台选择 **XR Spaces**。



填写好应用 ID，连接好 Spaces 设备（如果是分体式设备请连接移动端）后点击**构建>生成>运行**即可一键发布 AR 应用。