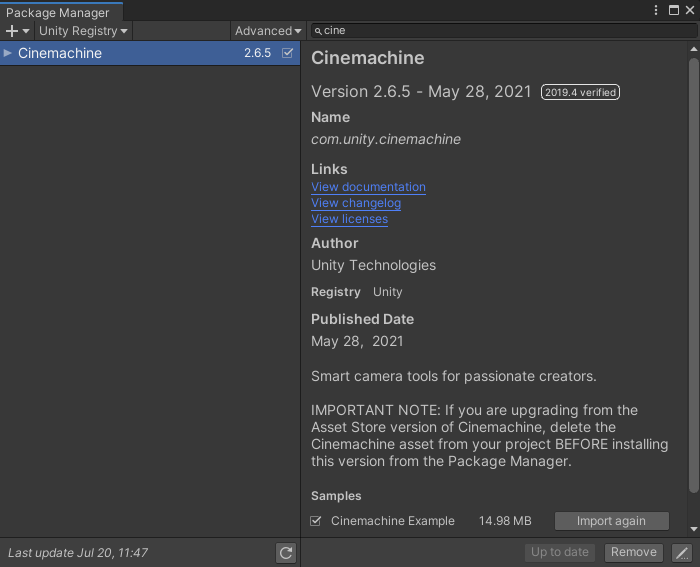
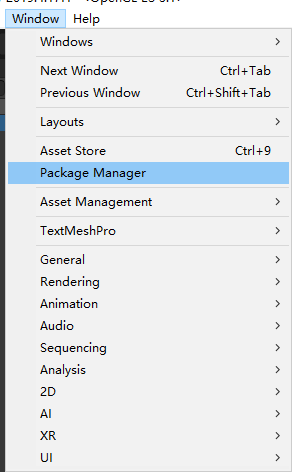
**Cinemachine Camera详细讲解（程序向）**

**前言**

本文档所用的Cinemachine Camera版本为2.6.5

文档可能过长，会尽量详细介绍到每一种不同的Cinemachine Camera类型

**一、 在Unity上导入Cinemachine插件**



将Samples内的Cinemachine Example同步导入之后会结合场景熟悉如何运用各种camera

**二、 什么是Cinemachine Camera**

以下为Untiy官方文档对于CinemachineCamera的详细描述

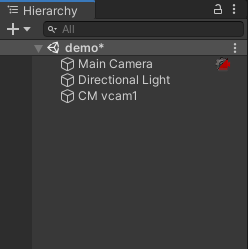
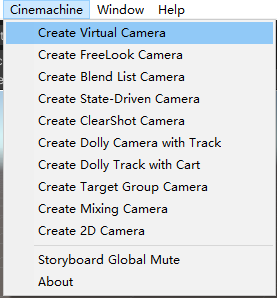
<https://docs.unity3d.com/Packages/com.unity.cinemachine@2.2/manual/index.html>

Cinemachine插件是Unity在2017年推出的一套专门控制Unity Camera的模块，适用于各种游戏场景中物体的移动变化(例如人物对话，人物移动跟随等等)，解决了许多关于摄像机间的复杂控制，混合，切换等复杂数学和逻辑，减少了开发过程中工程师对camera的脚本控制编写所需的时间成本，对一些摄像机功能需求能快速实现，同时能结合Timeline来实现动画效果。

接下来我们以Virtual Cameras来举例说明Cinemachine Camera的参数

**三、 Cinemachine Camera**

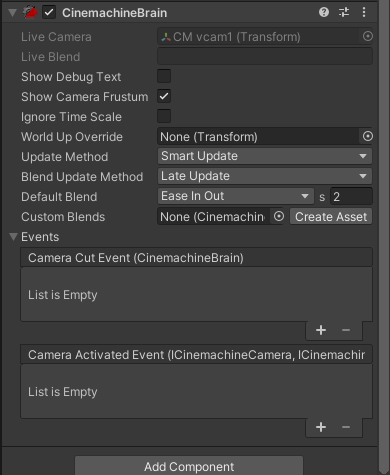
1. **新建Virtual Camera**



按照步骤生成 Virtual Camera， Virtual Cameras并不是Unity上的Camera模块，而是控制Unity相机，一般场景中只会有一个Unity相机，但可以有多个Virtual Cameras，来实现相机模块之间的多发单收的信息机制。

1. **了解Virtual Cameras**
2. **Cinemachine Brain 参数**

完成第一步后点开main camera你会发现多出一个多出了一个组件叫Cinemachine Brain



Cinemachine Brain是UnityCamera上的一个组件，相当于大脑，可以在每一帧来监控并计算场景中所有活动的Virtual Cameras的状态（位置，方向等），同步选择其他具有相同优先级的Virtual Cameras中或切换具有更高优先级的Virtual Camera甚至进行Virtual Cameras之间的混合（切换）。Virtual Cameras的优先级可以通过手动或者编写游戏逻辑来进行操控。

这里需要先了解Blend的概念，Blend（混合）是指用于描述从相机A切换到相机B的过程。

|  |  |
| --- | --- |
| 变量 | 解释 |
| Live Camera | 当前选择的活跃的Virtual Camera（运行时可以看到变化） |
| Live Blend | 当前的镜头切换名称 |
| Show Debug Text | 如果选中，显示调试效果（左上角） |
| Show Camera Frustum | 如果选中，显示相机的视锥体，相比原来自带的边框，这个会裁切掉不在显示范围内的边线。 |
| Ignore Time Scale | 忽视TimeScale的影响，选中后相机会一直响应用户的输入和相机的阻尼运动甚至是慢动作运行 |
| World Up Override | 不选中的时候，所有虚拟相机的Y轴为世界的向上方向（y轴），虚拟相机是通过这个来判断世界环境，如果将放入新物体，那个物体的Y轴会成为所有虚拟相机的世界Y轴 |
| Update Method | 更新所有与该Brain相关的虚拟相机，主要是计算每个VirtualCamera的State。里面具有四种算法选择。推荐默认选择SmartUpdate（下面会细说这四种算法含义） |
| Blend Update Method | 将计算VirtualCameras后更新的结果到Brain后，Brain进行更新的时机。只有两种算法选择。 |
| Default Blend | 默认虚拟相机之间的切换模式 |
| Custom Blends | 这是包含用于场景中特定虚拟摄像机之间混合的自定义设置的资产 |
| Camera Cut Event | 在这个列表上的事件只要当前场景任何一个虚拟相机激活并且没有混合时就会触发 |
| Camera Activated Event | 这个列表上的时间只要任何一个虚拟相机激活。如果涉及混合，则事件将在混合的第一帧上触发。 |

* **Update Method**

1. **FixedUpdate**

虚拟相机与物理硬件模块同步更新，在 FixedUpdate之后

1. **LateUpdate**

虚拟相机在MonoBehaviour LateUpdate时更新

1. **SmartUpdate**

虚拟相机更新根据目标的改变（移动，变形等）来进行动态更新，源码是通过判断在一定时间（具体代码是通过UpdateTracker实现的。这个一定时间其实是一个固定的帧数：UpdateStatus.kWindowSize,是个常数30）该虚拟相机的target在fixedUpdate移动次数多还是在lateupdate移动的次数多。来判断下一段时间用fixedUpdate还是lateUpdate。

1. **ManualUpdate**

如果 UpdateMode 设置为 ManualUpdate，则从外部脚本显式调用此方法以更新虚拟摄像机并定位主摄像机。对于其他更新方式，此方法会自动调用，不应从其他地方调用。

* **Blend Update Method**

1. **FixedUpdate**

与物理硬件模块同步更新，在 FixedUpdate之后，真实相机会将计算的数据同步

1. **LateUpdate**

在MonoBehaviour LateUpdate时更新，真实相机会将计算的数据同步

**(程序向）源码过程（这里只是理清框架，可以打开源码搜索下面说到的算法来观看）**

* 源码上在OnEnable阶段检测是否有帧在栈中，如果没有，新建第一帧放进栈(mFrameStack存储帧，List结构，用ID记录对应帧)中(调用BrainFrame()构造函数)，FrameStack中的第一个Frame是Brain每帧Tick游戏中虚拟相机的结果。其他的是TimeLine的，之后开启协程。
* 协程上运行的AfterPhysics()，根据WaitForFixedUpdate帧率来对虚拟相机得到的计算结果进行更新状态（可以在一帧内多次调用)。
* 在Start()阶段进行UpdateVirtualCameras()更新虚拟相机状态。
* 而在LateUpdate()阶段判断UpdateMethod状态是否为ManualUpdate，如果不是，执行ManualUpdate()来进行更新状态。ManualUpdate()是根据当前UpdateMethod来UpdateCameraState，紧接调用ProcessActiveCamera(deltaTime)来将虚拟相机更新到Unity上的真实相机
* ProcessActiveCamera()用来更新活跃虚拟相机的状态最后调用PushStateToUnityCamera()将虚拟相机状态更新到Unity真实相机，同时检测Camera Cut Event和Camera Cut Event是否挂载事件脚本，是则触发
* private static readonly AnimationCurve mDefaultLinearAnimationCurve = AnimationCurve.Linear(0, 0, 1, 1);
* 该代码为避免GC给每一帧分配内存。
* UpdateFrame0()用来更新帧，其中会有判断是否进行相机混合（切换），更换当前活跃虚拟相机，这块代码会在协程AfterPhysics()以及LateUpdate时中的ManualUpdate()中调用。
* ComputeCurrentBlend()计算当前所有活跃相机的混合，可以排除前n个，在ManualUpdate()里运行
* TopCameraFromPriorityQueue()获取当前可见的最高优先级的虚拟相机，在UpdateFrame0()里运行
* 在更新相机时，会通过UpdateStatus（该项在CinemachineCore里）来保证每个相机每帧不会被多次更新。以免造成性能浪费。

**（扩展）在了解Virtual Camera参数之前了解**

CinemachineCore以及Cinemachine.CameraState这两部分源码可以通过在VS2019的对象浏览器引用里看到调用了该类。如果想要看这两部分的源码，可以通过VS2019里面对象浏览器，搜素类名，点击全局搜索引用可以搜到该源码。但是该源代码只能看到函数命名以及部分值类型参数，如果想要详细浏览函数算法的具体实现，可以通过开启VS2019工具里面的反编译后，点击F12来观看部分算法通过二进制代码重现的源代码。

**了解下面两部分时需要了解的概念**

**CinemachinePipeline：**

计算流程，或者说是串联流水线，就是虚拟相机可以通过不断挂载CinemachineCompent脚本来进行各种叠加运算的过程，每一步得出的虚拟相机数据（位置，旋转，Lens等）都会继承到下一步运算。

* **CinemachineCore**
* 一个全局的管理类，保存当前所有有效的CinemachineBrain、VirturalCameraBase对象引用，定义了各种全局类和函数，用于Cinemachine系统的整体调度。该类是一个密封类，相当于全局单例的使用方法。它只需要正常维护Brain和Vcam列表，计算配对更新采样。还有一些基于进行CinemachinePipeline进行时的优化采样。
* 这里不详细介绍，如有需要建议观看源码。
* **Cinemachine.CameraState**
* CameraState（包含了位置、旋转、视角、额外偏移值等数据）是每一个虚拟相机都有自己独有的状态结构体存储，通过CinemachineBrain将其中的数据同步到真实相机上。CameraState是一个结构体存储方式。

Generated

* **State的计算流程：**

如果想要深入学习State如何计算可以走以下步骤：

根据上面反编译后找到源码，

Generated

Generated

找到该项，可以修改为true

这个时候切换到外面原本挂载CinemachineVirtual Camera脚本分割成三部分

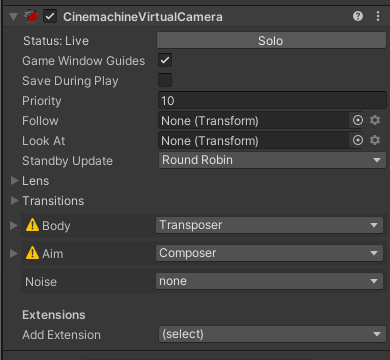
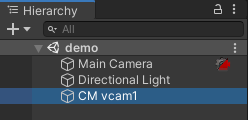
CinemachinePipeline脚本和另外两个CinemechineComposer和CinemechineTransposer组件。

CinemachinePipeline并没有实际的逻辑作用，只是起一个标记作用，代表这个节点是一个Pipeline节点。另外两个CinemachineComponent就是流水线中用于计算State的脚本。可以自行观看。

概括其运算过程如下：获取当前虚拟相机各种参数-->在CinemachinePipeline创建后-->通过CinemachineComponent和扩展Extensions进行State的运算，计算插值，叠加-->最后进入CinemachineBrain进行判断是否进行相机混合。

1. **Virtual Camera参数**

在了解完Cinemachine Brain之后打开场景中的CM vcam1后你会看到虚拟相机下挂载的CinemachineVirtualCamera脚本，该Object其实是一个empty Transform GameObject



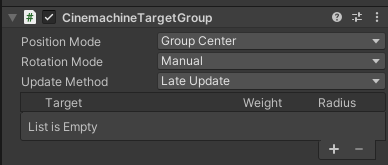
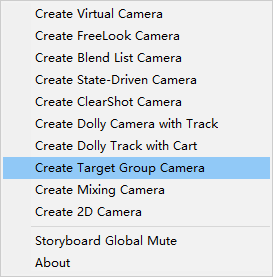
|  |
| --- |
| 表格里高亮的会在下面单独详细解释 |

|  |  |
| --- | --- |
| 变量 | 解释 |
| Status : Live/Standby/Disable | 当前这个虚拟相机的状态。  Live状态为当前该虚拟相机正在操控Unity真实相机，并且正在跟踪其目标并每帧更新一次。  Standby为当前该虚拟相机正在跟踪其目标并每帧更新一次，但它没有控制任何 Unity 摄像机。这是在场景中启用的虚拟摄像机的状态，但其优先级可能低于实时的虚拟摄像机。  Disable为该虚拟相机存在但不被允许使用在当前场景，他不能实时跟踪其目标，不消耗任何系统运算能力，但是可以在Timeline中生成并使用，如果参与混合也可使用。 |
| Game Window Guides | 选中时，Game窗口会显示辅助线，显示出虚拟相机的各个属性。辅助开发调整参数属性。需要LookAt或者Follow属性其中一个以上有设置物体并选择相关算法。 |
| Save During Play | 如果选中，运行中的虚拟相机属性可以被保存下来，退出Play状态不会被重置。 |
| Priority | 优先级，调整该虚拟相机的优先级。CinemachineBrain会根据这一项来选择LiveCamera同步到UnityCamera中。在Timeline中不生效该选项。 |
| Follow | 设置虚拟相机想要跟随的目标（Body Target），CinemachineComponent管道中的Body组件将会引用此目标并根据提供给它的规则和设置来定向 vcam。如果该项为空，则虚拟相机的坐标将会定义相机的坐标。 |
| Look At | 设置虚拟相机想要关注的目标（Aim Target）,CinemachineComponent管道中的Aim组件会将引用此目标并根据提供给它的规则和设置来定向 vcam。如果该项为空，则虚拟相机的变换方向将定义相机的方向。 |
| Standby Update：Never/Always/Round Robin | 当虚拟相机处于Standby状态时的更新方式。  Never方式为不更新  Always方式为每一帧都更新  Round Robin为循环更新，将所有待命的虚拟相机按照优先级大小来进行每一帧的循环更新（例如有10个待命的相机，第一帧更新第一个相机，第2帧更新第2个相机，…，第11帧更新第1个相机，以此类推） |
| Lens | 设置该虚拟相机的Lens属性，当该虚拟相机被激活且为活跃状态时，会将这些参数传递给Unity真实相机。其中Dutch为为镜头旋转（为屏幕空间）。 |
| Transitions | 相机混合时的设置。 |
| Body | CinemachineComponent管道中重要组件，重要设置参数，配合Follow使用 |
| Aim | CinemachineComponent管道中重要组件，重要设置参数，配合LookAt使用 |
| Noise | CinemachineComponent管道中重要组件，重要设置参数，可以实现模拟相机抖动效果 |
| Extensions | 可以给虚拟相机添加额外的扩展。 |

* **Follow&Look At的拓展Target Group应用**
* **TargetGroup**

如果本地目标为空，并且此虚拟相机是元相机集合的一部分，则将使用所有者的目标。这是由于Cinemachine中有一种相机是Target Group Camera，该类相机与VirtualCamera一样，只是在Follow和LookAt中选用了Target Group。

可以根据下列步骤创建



Generated

Target Group可以根据自身需求来添加自己想要观察和跟踪的物体在TargetList上，然后分配相应物体的Weight和Radius

* **Position Mode**

计算TargetGroup的Position计算方式

* GroupCenter：坐标轴对齐边界框的中心，简单来说就是根据所有物体的包围盒计算一个能包含所有物体的大包围盒的中心。然后虚拟相机的位置就是这个大包围盒的中心。
* Group Average：根据所有物体各自的位置加上权重来计算平均值得出虚拟相机的位置。
* **Rotation Mode**

计算TargetGroup的Rotation的计算方式

* Manual：手动设置TargetGroup的Rotation
* GroupAverage：所有成员的Rotation的加权平均值
* **Update Method**
* 更新TargetGroup的Transform的帧率选择
* Update/FixedUpdate/LateUpdate
* **Weight**

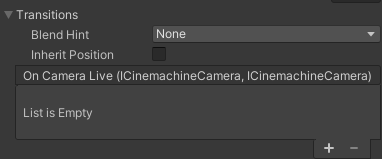
权重是用来计算Position和Rotation平均值的参数

* **Radius**

半径为组内每个物体的包围盒半径，用来计算包围盒，不能为负数

* **Transitions**

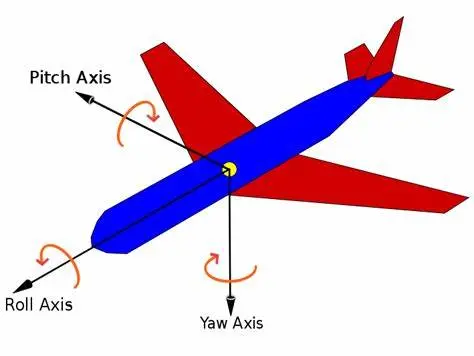
从其他虚拟相机转移过来影响本虚拟相机的参数



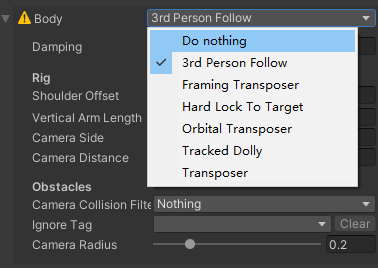
* **BlendHint**
* None：无，默认线性混合
* Spherical Position：根据Look At的物体球面旋转混合
* Cylindrical Position：根据Look At的物体柱面旋转混合（水平方向圆弧，垂直方向线性）
* Screen Space Aim When Target：在屏幕空间瞄准目标
* **Inherit Position**
* 当这个虚拟相机激活成活跃状态时，从上一个相机继承位置，保持两个相机位置相同。
* **On Camera Live**
* 当该相机变为活跃状态时会触发相应的事件脚本。

**以下为重中之重，CinemachineComponent管道的三大重要组件(Body,Aim,Noise)！！！**

在阅读以下文章之前先了解以下概念



* Pitch Axis：俯仰轴
* Roll Axis：翻转轴
* Yaw Axis：航向轴
* 这三个轴与x，y，z轴不同概念
* **Body**



**打开Body组件界面有7种算法**

以下图片出现黄色感叹号是因为Follow没有挂载跟随目标

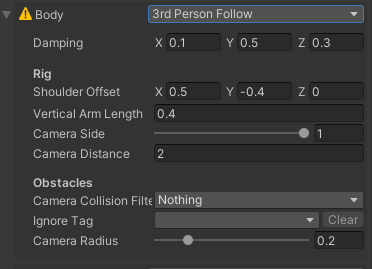
1. **Do nothing**

Generated

选择该算法后，该虚拟相机的坐标不会移动更新，通常为静态镜头选择此算法或使用自定义脚本直接为相机位置设置动画。通常和LookAt配合使用，模拟固定位置的跟随镜头。

1. **3rd Person Follow**

选择该算法后虚拟相机会自动跟随任务移动，就是常见的第三人称视角



|  |  |
| --- | --- |
| 变量 | 含义 |
| Damping | 阻尼系数，指摄像机向x,y,z轴三个方向移动反应速度，系数越小反应越快，系数越大反应越慢，可以通过调整三个不同方向轴的阻尼系数来实现不同的移动效果。 |
| Shoulder Offset | 相对于跟随目标原点的肩膀支点的位置。这个偏移量位于目标模型空间。 |
| Vertical Arm Length | 手相对于肩膀的垂直偏移量。当相机垂直旋转时，arm长度会影响跟踪目标在屏幕空间的偏移。 |
| Camera Side | 明确哪个位置是相机的肩膀（0-1，代表left，in-between，right三种状态） |
| Camera Distance | 相机的手部位置会被放置多远距离 |
| Camera Collision Fiter | 相机碰撞过滤器，相机会自动避免阻碍这些层。有以下成员：Nothing/Everything/default/TransparentFX/Ignore Raycast/Water/UI |
| Ignore Tag | 检测忽略的Tag |
| Camera Radius | 虚拟相机会尽量与遮挡物保持不低于这个距离。如果你看到由于较大的FOV导致相机穿透了遮挡物内部，可以适当增加这个值。 |

1. **Framing Transposer**

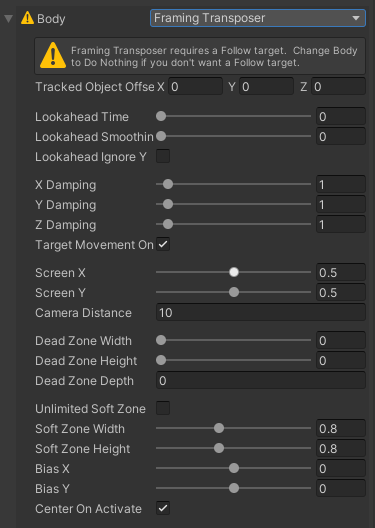
此虚拟相机本身算法将相机以固定的屏幕空间关系移动到跟随目标。您还可以指定偏移、阻尼和合成规则。**Framing Transposer**只改变相机在空间中的位置。它不会重新定向或以其他方式瞄准相机。

**Framing Transposer**专为 2D 和正交相机而设计。但它也适用于透视相机和 3D 环境。

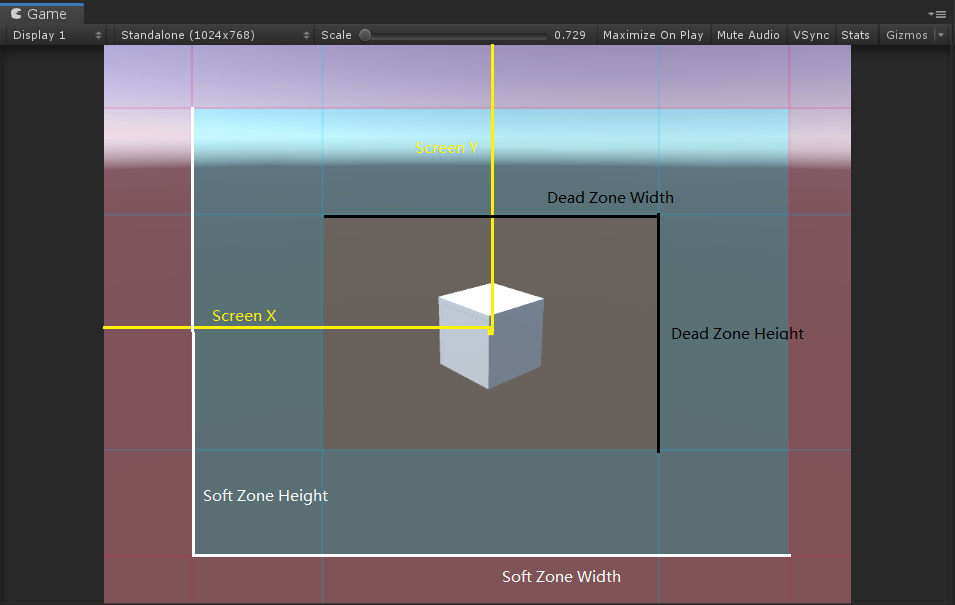
此算法首先沿相机 Z 轴移动相机，直到跟随目标与相机的 XY 平面处于所需距离。然后在其 XY 平面中移动相机，直到跟随目标位于相机屏幕上的所需点。

**注意**：**Framing Transposer**忽略 LookAt 目标 - 仅使用 Follow 目标。

如果**Follow**目标是Target Group，则可以使用其他属性来构建整个组。



|  |  |
| --- | --- |
| 变量 | 含义 |
| **Tracked Object Offse** | 对跟随目标对象的偏移量（在目标-局部坐标中）。相机将尝试框定目标位置加上此偏移量的点。当目标原点不是目标点时，使用它来校正相机 |
| **Lookahead Time** | 根据目标的运动，调整虚拟相机与“跟随”目标的偏移量。Cinemachine预测目标在未来数秒之内到达的位置并提前设置Unity相机的位置。这个功能对微动的动画敏感，并且会放大噪点，导致非预期的相机抖动。如果不能接受目标运动时的相机抖动，减小这个属性可能会使相机动画更流畅。 |
| **Lookahead Smoothing** | 预测算法的平滑度。较大的值可以消除抖动但会使预测滞后。 |
| **Lookahead Ignore Y** | 如果选中，则忽略沿 Y 轴的移动以进行预测计算。 |
| **X ，Y，Z Damping** | 相机尝试保持 x ，y，z轴偏移的响应程度。小数字使相机更灵敏。数字越大，相机响应越慢。每个轴使用不同的设置可以产生广泛的相机行为。 |
| **Target Movement Only** | 如果启用此选项，则阻尼仅适用于目标的运动。相机旋转更改将绕过阻尼。 |
| **Screen X，Y** | 目标的水平，垂直屏幕位置。相机移动的结果是使目标处于此位置。 |
| **Camera Distance** | 沿摄像机Z轴与跟随目标保持的距离。 |
| **Dead Zone Width** | 当目标在此位置范围内时，不会水平移动相机 |
| **Dead Zone Height** | 当目标在此位置范围内时，不会垂直移动相机。 |
| **Dead Zone Depth** | 当跟随目标距离相机在此范围内时，不会沿其z轴移动相机。 |
| **Unlimited Soft Zone** | 如果选中，Soft Zone没有边界 |
| **Soft Zone Width** | 当目标处于此范围内时，会水平移动相机，将目标移回到Dead Zone中。Damping属性会影响摄像机的运动速度。 |
| **Soft Zone Height** | 当目标处于此范围内时，会垂直移动相机，将目标移回到Dead Zone中。Damping属性会影响摄像机的运动速度。 |
| **Bias X，Y** | Soft Zone的中心与目标位置的水平，垂直偏移。 |
| **Center On Active** | 选中时，虚拟相机激活时会将镜头中心对准物体。不选中时，虚拟相机会将目标物体放置在最近的dead zone边缘。 |



非常详细不用介绍了

1. **Hard Lock To Target**

该算法实现虚拟相机和跟随目标使用相同位置。可以用作第一人称

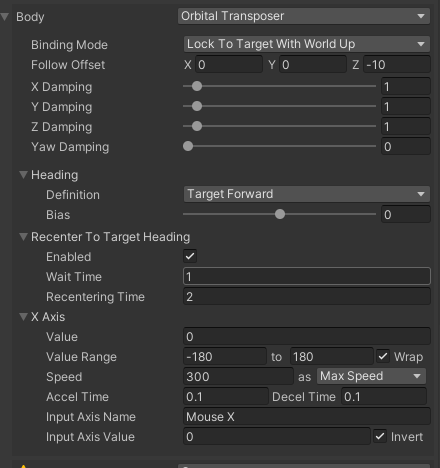
Generated

1. **Orbital Transposer**

这个算法支持相机和目标之间的可变相对位置关系。可以接受玩家的输入，动态的控制相机的位置。

Orbital Transposer引入了一个新的概念叫heading，代表了目标移动的方向或面朝的方向。Orbital Transposer会尝试移动相机，让镜头朝向heading的方向。默认情况下，相机的位置会在target的正后面。也可以通过Heading Bias属性设置。如果给Orbital Transposer添加了输入控制器，玩家就可以控制相机围绕目标旋转。可以设置为Input Manager中的轴，也可以直接用脚本控制。

当Recenter To Target Heading属性选中时，在没有输入时Orbital Transposer可以自动重新居中相机。



|  |  |
| --- | --- |
| 变量 | 含义 |
| **Binding Mode** | 解释虚拟摄像机与跟随目标的偏移时要使用的坐标空间。 |
| **Follow Offset** | 跟随目标时的位置偏移 |
| **X Damping** | 相机在X轴上移动的阻力系数。较小的值会使相机反应更快。较大的值会使相机的反应速度变慢。每个轴使用不同的设置可以制造出各种类型相机的行为。 绑定模式为Simple Follow With World Up时不可用。 |
| **Y Damping** | 相机尝试保持 y 轴偏移的响应速度。小数字使相机更灵敏。数字越大，相机响应越慢。 |
| **Z Damping** | 相机尝试保持 z 轴偏移的响应程度。小数字使相机更灵敏。数字越大，相机响应越慢。 |
| **Yaw Damping** | 相机在y轴旋转的阻力系数。较小的数字会使相机反应更快。较大的数字会使相机的反应速度变慢。每个轴使用不同的设置可以制造出各种类型相机的行为。Binding Mode为Lock to Target With World Up、Lock to Target No Roll、Lock to Target时可用。 |
| **Pitch Damping** | 相机在x轴旋转的阻力系数。较小的数字会使相机反应更快。较大的数字会使相机的反应速度变慢。 Binding Mode为Lock to Target No Roll、Lock to Target时可用。 |
| **Roll Damping** | 相机在z轴旋转的阻力系数。较小的数字会使相机反应更快。较大的数字会使相机的反应速度变慢。 Binding Mode为Lock to Target时可用。 |
| **Heading** | 指定如何计算跟随目标的航向。 |
| **Recenter To Target Heading** | **Orbital Transposer \_\_可选地自动将相机重新居中。**选中**\_\_Recenter To Target Heading 后**，Orbital Transposer 会自动将相机移回目标航向。您可以指定在检测到没有用户输入后要等待的时间长度以及重新定位的速度。 |
| **X Axis** | 航向控制。此处的设置控制相机响应玩家输入的行为。 |

* **Binding Mode**
* Lock To Target On Assign：本地空间，相机被激活或target赋值时的相对位置。
* Lock To Target With World Up：本地空间，保持相机y轴朝上，yaw和roll为0。
* Lock To Target No Roll：本地空间，锁定到目标物体，roll为0。
* Lock To Target：本地空间，锁定到目标物体 。
* World Space：世界空间 。
* Simple Follow With World Up：相对于目标的位置，使用相机的本地坐标系，保持相机y轴朝上。
* **Heading**
* Definition：计算Follow朝向的方法
* 选择 Position Delta 以根据上次更新和当前帧的目标位置差异来计算航向。
* 选择 Velocity 以使用目标刚体的速度。 如果目标没有刚体组件，则恢复到位置增量。
* 选择 Target Forward 以使用目标的本地 Forward 轴作为航向。
* 选择 World Forward 以使用恒定的世界空间 Forward 作为航向。
* Velocity Filter Strength：在定义中使用 Position Delta 或 Velocity 时控制速度的平滑。
* Bias：相对于航向放置相机的轨道中的角度偏移。 以度为单位。 轴值为 0 会将相机放在此处。
* **Recenter To Target Heading**
* Wait Time：如果在轴上没有检测到用户输入，相机会在重新定位之前等待这么长时间（以秒为单位）。
* Recentering Time：重新自动定位居中的过程花费的时间.
* **X Axis**
* Value：当前值
* Value Range ：值范围
* Wrap：如果选中，则轴在 Min 和 Max 值之间形成一个循环。
* Speed ：最大速度（Max Speed）或者最大增加速度（Input Value Gain）
* Accel Time ：加速到最高速度所需要的时间
* Input Axis Name：在 Unity 输入管理器中指定的此轴的名称。 设置为空字符串以禁用此轴的自动更新。
* Input Axis Value：玩家输入轴的值。值0表示没有输入。可以通过编写脚本控制
* Invert：是否反转输入的值（取相反数）

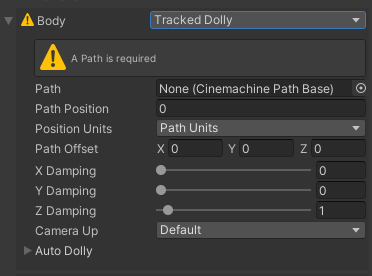
1. **Tracked Dolly**

这个算法可以让相机沿预定路径移动。使用Path Position属性来指定将虚拟相机放置在路径上的位置。需Follow目标移动。

使用Auto-Dolly模式将虚拟相机移动到路径上最接近Follow目标的位置。启用后，Auto-Dolly会自动将虚拟相机的位置移动到最接近目标的路径上的位置。

提示：使用Auto-Dolly模式时，一定要谨慎选择路径形状。在围绕某个点形成弧形的路径上可能会有问题。举一个极端的例子，考虑一条以Follow目标为中心的完美圆形路径。路径上最接近目标的点变得不稳定，因为圆形路径上的所有点都同样接近目标。在这种情况下，将Follow目标移动很小的距离会导致相机在轨道上移动很大的距离。

注意此算法一定要有Path



|  |  |
| --- | --- |
| 变量 | 含义 |
| **Path** | 相机移动的路径。此属性必须引用CinemachinePath或Cinemachine Smooth Path对象。 |
| **Path Position** | 沿路径放置相机的位置。直接给这个属性作动画或启用Auto-Dolly。这个值以Position Units指定的单位为单位。 |
| **Position Units** | 路径位置的度量单位。 |
| **Path Offset** | 相机相对于路径的位置。X 垂直于路径，Y 向上，Z 平行于路径。使用此属性可使相机偏离路径本身 |
| **X ，Y，Z Damping** | 相机尝试保持 x ，y，z轴偏移的响应程度。小数字使相机更灵敏。数字越大，相机响应越慢。每个轴使用不同的设置可以产生广泛的相机行为。 |
| **Camera Up** | 如何为虚拟摄像机设置向上矢量。这会影响屏幕构图，因为相机 Aim 算法尝试尊重向上方向。 |
| **Auto Dolly** | 控制自动轨道位置选择方式。要使用此功能，必须设置Follow目标。 |

* **Path**
* **Cinemachine中有两种Dolly Path组件：**
* **CinemachinePath组件**：每个路径点都可以设置位置、切线和翻转角度，可以最大程度控制路径。但是如果切线设置的不合理，相机在路径上运动时可能会有不稳定的现象。
* **CinemachineSmoothPath组件**：每个路径点都可以设置位置和翻转角度。组件中会使用Bezier算法来推算路径点之间的位置。虽然SmoothPath不能完全控制路径，但是整个路径会更平滑和连续。**推荐优先使用。可以避免出现，尽管路径位置始终是平滑连续的，但沿路径设置动画时仍然可能会产生不稳定的现象。**
* **额外扩展：什么是Bezier算法和Bezier曲线**
* **Bezier曲线**

[怎么理解贝塞尔曲线? - 知乎](https://www.zhihu.com/question/29565629)

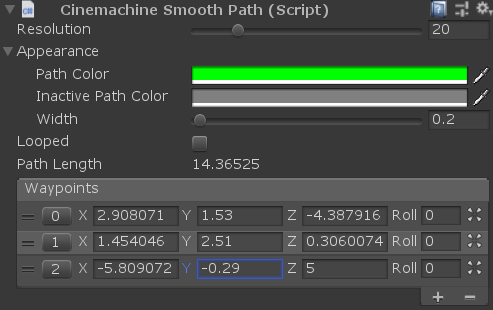
* **Bezier算法**

详细的算法推导可以在上述链接中观看如何逐阶层运算各个阶的Bezier曲线。

Unity本身自带的**CinemachineSmoothPath**脚本源码，他们是定义一个世界空间路径，由一组路点组成，每个路点都有位置和滚动设置。Bezier 插值在航点之间执行，以获得平滑和连续的路径。 计算距离缓存时，对点之间的路径进行多次采样。路径将通过所有航路点，并且（与 CinemachinePath 不同）保证一阶和二阶连续性。

源码都是运算过程，大家可以自行观看。

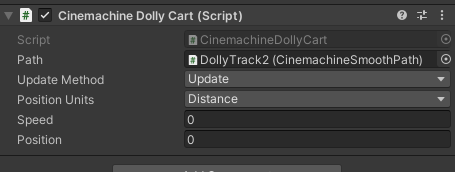
* **如何创建Dolly Path**
* 方法一：**如果是在相机上制作路径动画**，可以直接创建Dolly Camera。菜单栏**Cinemachine > Create Dolly Camera with Track**。点击后会创建一个虚拟相机和dolly path。这个Path默认是SmoothPath。虚拟相机会自动设置好Body属性为Dolly Track并且Path属性赋值为创建出来的Dolly path。
* 方法二：**如果是给GameObject制作路径动画**，可以直接创建Dolly Cart。菜单栏**Cinemachine > Create Dolly Track With Cart**。点击后会创建一个虚拟相机和DollyCart。这个Path默认是SmoothPath。只需要把做路径动画的物体作为Cart的子物体即可。
* 方法三：直接创建Dolly Path。创建一个空物体，给空物体添加CinemachinePath或CinemachineSmoothPath组件。
* **SmoothPath参数详解**



* **Resolution**： 每个路径点之间采样的次数。Cinemachine在计算路径距离时使用此值来限制粒度。场景视图中路径Gizmo上的交叉线反映了该值。
* **Appearance** ：路径在场景中显示的外观。只在编辑器中生效。
* **Path Color** ：选中路径时的颜色。
* **Inactive Path Color** ：未选中路径的颜色。
* **Width** ：路径轨道的宽度。
* **Looped** ：选中后，路径会首尾相连。
* **Path Length** ：整个路径长度，这个数据是只读的，根据所有路径点计算出来的。
* **Waypoints** ：定义路径点的列表。
* **Position** ：位置，在路径局部空间中的位置（即相对于路径GameObject本身的变换）
* **Roll\*** ：翻转角度。朝向的其他轴方向是从切线和世界的向上向量推断出来的。
* **Dolly Cart参数**

**（只有Create Dolly Track With Cart才有）详解**：

这是限制在 CinemachinePath的移动镜头。 它可用于为路径上的任何对象设置动画，或作为 Cinemachine 虚拟摄像机的跟随目标。这个组件就是把当前所附着的GameObject沿着path移动。



Speed ：移动物体的速度。基于Position Unit设置的单位。如果速度不为0，则小车会根据速度自动移动。

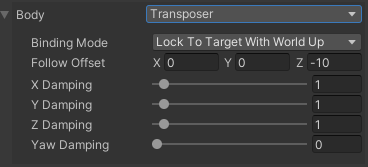
Position :当前物体所处的位置。基于Position Unit设置的单位

* **Position Units**
* Path Units ：使用路径点的序号。比如0代表第一个路径点，1代表第二个路径点。
* Distance ：使用路径的总长度。
* Normalized： 归一化。0代表路径最开始的位置，1代表路径结束的位置。
* **Camera Up**
* Default：不修改虚拟相机up方向，使用Cinemachine Brain的Override的World Up
* Path：使用路径当前的向上向量
* Path No Roll：使用路径节点当前的向上向量，Roll\*为0
* Follow Target：使用跟随目标的向上向量
* Follow Target No Roll：使用跟随目标的向上向量，Roll\*为0
* 不要修改虚拟摄像机的向上方向。 相反，使用 Cinemachine Brain 中的 World Up Override 属性
* **Auto Dolly**
* Enabled：选中使用自动dolly，注意：这对性能可能会有一些影响，却决于Search Resolution
* Position Offset：从路径最近点到跟随目标的偏移量
* Search Radius：对当前路径点的n个单位点进行搜索，如果为0则为整条路径所有点
* Search Resolution：Cinemachine 通过将片段分成许多直线来搜索片段。 数字越大，结果越准确。 但是，对于更高的数字，性能会成比例地变慢。

1. **Transposer**

这个算法将虚拟相机的坐标与跟随目标的坐标有固定的偏移量来进行跟随，也可以使用Damping属性

简单来说就是虚拟相机跟目标会有固定的位置差偏移

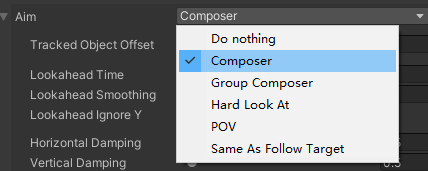


根据BindingMode的不同选择会有不同参数的Damping可以调整

Binding Mode 各种模式可以参考上面Orbital Transposer 的详细介绍

* **Aim**

Aim组件属性负责指定如何旋转虚拟相机。 要更改相机的位置，请使用 Body 属性。



**打开Aim组件界面有6种算法（6种算法源码也一样被隐藏了起来，所以只能先做解释）**

同理想要用Aim组件也需要先挂载LookAt

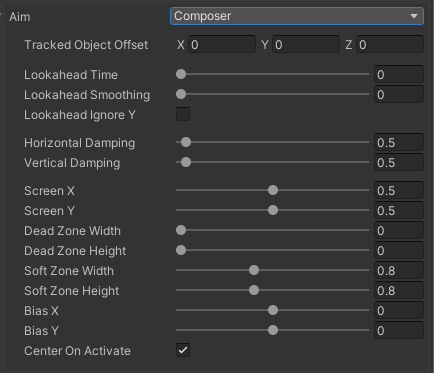
1. **Do nothing**

Generated

此虚拟摄像机瞄准算法不操控虚拟摄像机。为静态镜头选择此算法或使用自定义脚本直接为旋转设置动画。

1. **Composer**

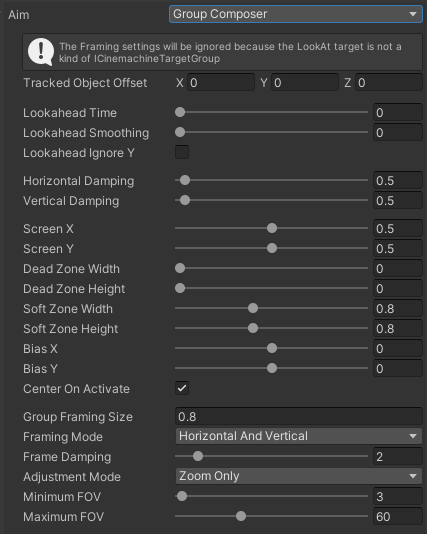
该算法用来控制镜头瞄准目标以面向目标坐标，最常用的算法，通常用来做跟随人物



|  |  |
| --- | --- |
| 变量 | 含义 |
| **Tracked Object Offset** | 相对于跟踪目标的偏移。当注视的位置不是被跟踪对象的中心时，可以通过这个属性微调来确定需要跟踪目标位置。 |
| **Lookahead Time** | 根据 Look At 目标的运动调整偏移量。 该算法估计目标将在未来几秒内出现的点。 此功能对嘈杂的动画很敏感。 它可以放大噪声，导致不良的相机抖动。 如果目标在运动时相机抖动不可接受，请调低此属性或更平滑地为目标设置动画。 |
| **Lookahead Smoothing** | 控制前瞻算法的平滑度。较大的值可以消除抖动预测但会使预测滞后。 |
| **Lookahead Ignore Y** | 预测算法会忽略Y轴的运动。 |
| **Horizontal Damping** | 水平阻尼。相机在屏幕水平方向上对目标的反应速度如何。使用较小的数字可以使照相机更快地旋转，以使目标保持在dead zone。使用较大的数字来模拟较重，响应缓慢的相机。 |
| **Vertical Damping** | 垂直阻尼。相机在屏幕垂直方向上对目标的反应速度如何。使用不同的垂直和水平设置可以模拟不同相机行为。 |
| **Screen X** | dead zone中心的水平屏幕位置，相机旋转保持目标在此处。 |
| **Screen Y** | dead zone中心的垂直屏幕位置，相机旋转保持目标在此处。 |
| **Dead Zone Width** | 目标在这个区域时，相机会忽略目标的任何移动，此属性设置这个区域的宽度。目标位于该区域内的任何位置时，虚拟相机不会更新其旋转角度。这对于忽略较小的目标移动很有用。 |
| **Dead Zone Height** | 目标在这个区域时，相机会忽略目标的任何移动，此属性设置这个区域的高度。如果目标位于该区域内的任何位置，则虚拟相机不会更新其旋转角度。这对于忽略较小的目标移动很有用。 |
| **Soft Zone Width** | soft zone的宽度。如果目标出现在屏幕的此区域中，则相机将旋转，以在Horizontal Damping设置的时间内将其推回dead zone。 |
| **Soft Zone Height** | soft zone的高度。如果目标出现在屏幕的此区域中，则相机将旋转，以在Vertical Damping设置的时间内将其推回dead zone。 |
| **Bias X** | soft zone中心相对于dead zone中心的水平偏移 |
| **Bias Y** | soft zone中心相对于dead zone中心的垂直偏移。 |
| **Center On Activate** | 选中时，虚拟相机激活时会将镜头中心对准物体。不选中时，虚拟相机会将目标物体放置在最近的dead zone边缘。 |

1. **Group Composer**

该算法与上面算法一致，区别在于对象为Gourp，可以用于对多对象的LookAt，需要的组件为上面介绍到的Target Group

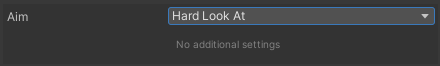


|  |  |
| --- | --- |
| 变量 | 含义 |
| Group Framing Size | 目标应占据的屏幕大小比例。使用1填充整个屏幕，使用0.5填充一半的屏幕，依此类推。 |
| Framing Mode | 指定构图时要考虑的屏幕尺寸。 |
| Frame Damping | 阻尼系数，相机完成构图的快慢，越大越慢。 |
| Adjustment Mode | 指定构图时要考虑的屏幕尺寸。 |
| Minimum FOV | 自动调节FOV时的最小值。 |
| Maximum FOV | 自动调节FOV时的最大值。 |

* **Framing Mode**
* Horizontal 仅考虑水平尺寸。忽略垂直尺寸。
* Vertical 仅考虑垂直尺寸。忽略水平尺寸。
* Horizontal And Vertical 使用水平和垂直尺寸中较大的那个来获得最佳匹配。
* **Adjustment Mode**
* Zoom Only 不移动相机，仅调整FOV。
* Dolly Only 移动相机，不修改FOV。
* Dolly Then Zoom 将相机移动到范围允许的最大范围，然后根据需要调整FOV

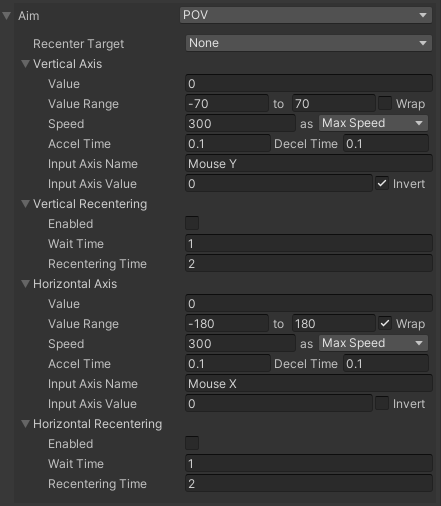
1. **Hard Look At**

保持虚拟相机中心为LoaakAt的中心



1. **POV**

这种算法使虚拟相机的设定取决于用户的设定



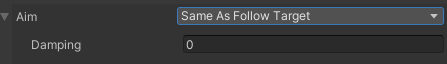
|  |  |
| --- | --- |
| 变量 | 含义 |
| **Recenter Target** | 重置回中心的目标对象。 |
| **Vertical Axis** | 控制虚拟相机目标的垂直朝向。 |
| **Vertical Recentering** | 当接收不到玩家输入时，自动在垂直方向重新居中。 |
| **Horizontal Axis** | 控制虚拟相机目标的水平方向。 |
| **Horizontal Recentering** | 当接收不到玩家输入时，自动在水平方向重新居中。 |

因为参数都是相对的这边只介绍水平，垂直方向相对可读

* **Recenter Target**
* None： 无
* Follow Target Forward： Follow属性的forward
* Look At Target Forward ：Look At属性的forward
* **Vertical Axis**
* Value ：轴的当前值，以度为单位。可接受的值为-90至90。
* Value Range ：虚拟相机的垂直轴的最小值和最大值。
* Wrap： 如果选中，则轴将在Value Range范围内，形成一个循环。
* Max Speed： 该轴的最大速度，以度/秒为单位。
* Accel Time：处于最大值时，加速到最大速度所花费的时间（以秒为单位）。
* Decel Time ：轴减速为零所花费的时间（以秒为单位）。
* Input Axis Name ：输入轴的名称。在Unity Input Manager中指定的该轴的名称。将此属性设置为空字符串来禁用此轴的自动更新。
* Input Axis Value ：输入轴的值。值为0表示无输入。你可以直接从自定义脚本中修改这个值。或者设置Input Axis Name并由Unity Input Manager驱动。
* Invert ：将原始值反转。
* **Vertical Recentering**
* Enable ：是否启用。选中以启用自动垂直居中。
* Wait Time：等待时间。如果在垂直轴上未检测到用户输入，则相机将等待几秒钟，然后再进行重新居中。
* Recentering Time：重新居中所花费的时间。

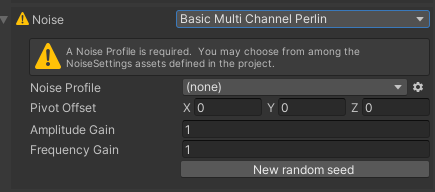
1. **Same As Follow Target**

此虚拟相机瞄准算法与跟随目标的方向相匹配。 如果想让相机和目标保持同样的位置和朝向，可以配合Body 属性中的 Hard Lock to Target 算法一起使用时。



* **Noise**

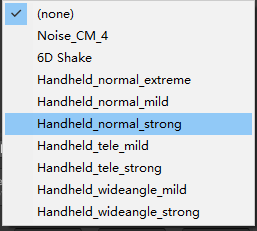
Noise组件平时是默认关闭的，想要使用要选择以下选项



其本质就是通过噪音来实现虚拟相机的模拟抖动。Cinemachine 包含一个**基本的多通道 Perlin**组件，该组件将 Perlin 噪声添加到虚拟摄像机的运动中。**Perlin 噪声**是一种计算具有自然行为的随机运动的技术。

如果想要更加详细了解柏林噪声可以观看这篇文章

<https://blog.csdn.net/u010669231/article/details/97051705>



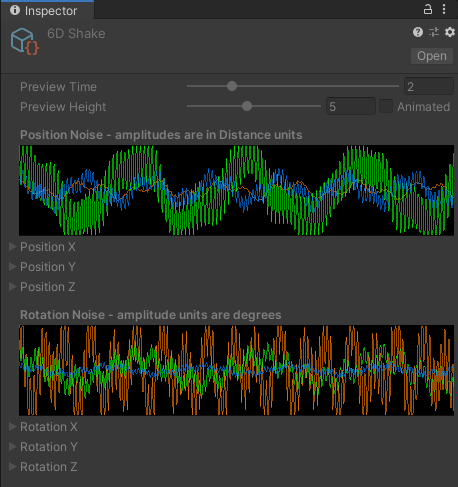
Noise Profile是一个配置文件资产，里面有噪声的相关配置。Cinemachine中自带了几种profile配置，你也可以自己创建profile。Cinemachine在计算相机的位置后会添加噪声造成的偏移。这样，相机噪声不会影响将来相机运动的计算。 在介绍噪音该怎么配置前，先介绍这两个系数。

学过信号与系统都会知道一段信号它具有频率以及幅度两个特性。

|  |  |
| --- | --- |
| 变量 | 含义 |
| **Amplitude Gain** | 应用于噪声配置文件中定义的幅度的增益。使用 1 以使用噪声配置文件中定义的幅度。将此设置为 0 会静音。提示：为该属性设置动画以提高和降低噪波效果。直白说就是越大抖的幅度越大，越小抖得幅度越小 |
| **Frequency Gain** | 应用于噪声配置文件中定义的频率的因子。使用 1 以使用噪声配置文件中定义的频率。使用较大的值可以更快地抖动相机。提示：为该属性设置动画以提高和降低噪波效果。直白说就是越大抖得越剧烈，越小抖得越平稳 |

以下介绍怎么修改或生成你的噪音资产（如果是程序向，个人建议看我上面的链接先了解一段柏林噪声怎么生成的）

以cinemachine自带的资产里面叠加最复杂的6D Shake举例



|  |  |
| --- | --- |
| 变量 | 含义 |
| **Preview Time** | 在 Inspector的图表中显示的秒数。 此属性用于在 Inspector 中进行编辑； 它不会影响您正在编辑的噪声配置文件资产的内容。 |
| **Preview Height** | 在 Inspector的图表中显示的幅度（信号高度） |
| **Animated** | 选中以显示图表中噪声分布示例的移动表示 |
| **Position Noise** | 虚拟相机坐标噪音 |
| **Position X**  **Position Y**  **Position Z** | 每个轴都可以设置多层的噪波。每个轴有一个图形展示。可以设置多层，点击下面的+或-可以添加或删除一层。 |
| **Rotation Noise** | 虚拟相机的旋转噪音 |
| **Rotation X**  **Rotation Y**  **Rotation Z** | 每个轴都可以设置多层的噪波。每个轴有一个图形展示。可以设置多层，点击下面的+或-可以添加或删除一层。 |

**举例Position X**

* Frequency频率： 噪声层中的频率，以Hz为单位。
* Amplitude振幅： 噪声层中波的振幅（高度），以度数为单位。
* Non-random ：选中时不给波形添加柏林噪声。在没有Perlin噪声的情况下，Cinemachine使用规则的正弦波。取消选中可将Perlin噪声应用于该层，从而使频率和振幅随机化，同时保持在所选值的附近。

**备注使用Noise小提示**

真实的的噪声曲线通常将低频，中频和高频混合在一起。创建新的noise profile时，可以从每个轴的这三层噪声开始。

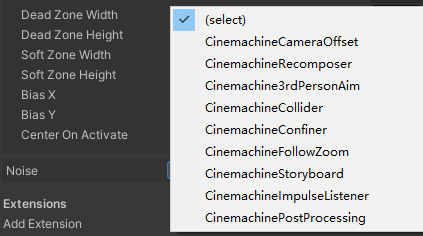
对于振幅（Amplitude），较大的值相机晃动越明显。对于远景，使用较小的振幅值，因为较小的FOV会放大晃动效果。

对于频率（Frequency），典型的低频范围为0.1-0.5 Hz，中频范围为0.8-1.5，高频范围为3-4。最高可用频率取决于游戏的帧频。游戏通常以30或60Hz运行。高于游戏帧速率的噪声频率可能会出现采样出现跳跃的情况。

例如，如果你的游戏以60帧/秒的速度运行，并且将噪声频率设置为100，则相机会发出断断续续的噪声。这是因为你的游戏渲染帧率无法比噪声频率更快。

* **Extensions**

扩展是增强虚拟摄像机行为的组件。 例如，Collider 扩展将相机移出阻碍相机对其目标视野的游戏对象。Cinemachine 包括各种扩展。 通过从 CinemachineExtension 类派生来创建您自己的自定义扩展。

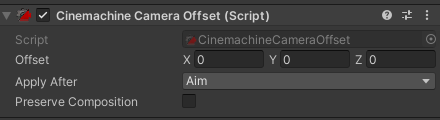


以下是官方Cinemachine2.6.5版本的Extensions

以下仅作参数讲解以及适用于什么工作环境

**CinemachineCameraOffset**

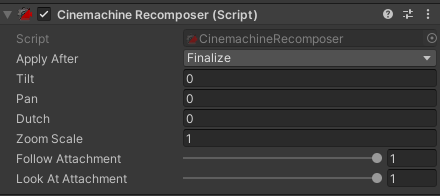
Cinemachine 虚拟摄像机的附加模块，可为摄像机添加最终偏移量



|  |  |
| --- | --- |
| 变量 | 含义 |
| offset | 偏移量 |
| ApplyAfter | 最后加到哪一个参数（Body，Aim，Noise，Finalize） |
| Preserve Composition | 如果在瞄准后应用偏移，请重新调整瞄准以尽可能保留 LookAt 目标的屏幕位置 |

**CinemachineRecomposer**

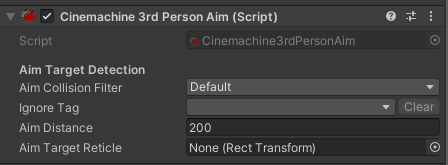
Cinemachine 虚拟摄像机的附加模块，可对摄像机组成进行最终调整。 它旨在用于时间轴上下文，您希望在其中手动调整程序或记录的相机瞄准的输出。



|  |  |
| --- | --- |
| 变量 | 含义 |
| ApplyAfter | 最后加到哪一个参数（Body，Aim，Noise，Finalize） |
| Tilt | 摄像机倾斜 |
| Pan | 摄像机平移 |
| Dutch | 摄像机翻滚 |
| ZoomScale | 按照此比例缩放 |
| FollowAttachment | 降低该值可以降低相机对Follow的灵敏度（正常为1） |
| LookAtAttachment | 降低该值可以降低相机对LookAt的灵敏度（正常为1） |

**Cinemachine3rdPersonAim**

Cinemachine 虚拟摄像机的附加模块，强制 LookAt 指向屏幕中心，消除噪音和其他校正。 这对于需要始终精确瞄准的第三人称风格瞄准相机非常有用，即使存在位置或旋转噪声也是如此。

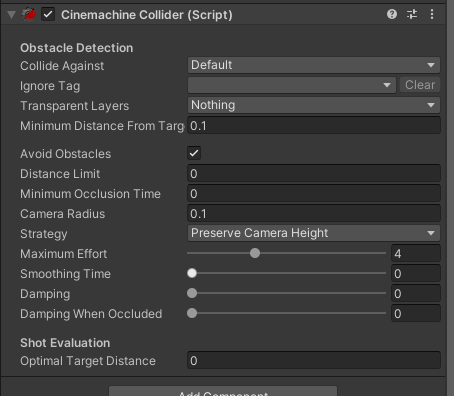


|  |  |
| --- | --- |
| 变量 | 含义 |
| AimCollisionFilter | 检测这些层的碰撞 |
| IgnoreTag | 带有此标签的对象将被忽略。 |
| AimDistance | 物体检测射线的投射距离。 |
| AimTargetReticle | 该 2D 对象将位于游戏视图中光线投射命中点（如果有）的上方，或者如果未检测到命中点，则将保留在屏幕中央。 可能为空，在这种情况下不会出现屏幕指示符 |

**CinemachineCollider**

Cinemachine 虚拟摄像机的附加模块，用于对虚拟摄像机的最终位置进行后处理。 根据提供的设置，碰撞器将尝试通过远离会阻碍视线的物体来保持与虚拟相机的 LookAt 目标的视线。该项可以利用在很多游戏场景。

碰撞器使用物理射线投射器。因此，Cinemachine Collider 要求潜在障碍物具有碰撞体体积。 此要求存在性能成本。 如果此成本在您的游戏中过高，请考虑以不同方式实现此功能



|  |  |
| --- | --- |
| 变量 | 含义 |
| CollideAgainst | 物体在这些选中的层级里面会被检测 |
| IgnoreTag | 带有此标签的对象将被忽略。 |
| TransparentLayers | 透明层，代表不会遮挡视线的层layer |
| MinimumDistanceFromTarget | 如果遮挡物和目标物体很近，小于这个属性的值，就忽略这个遮挡物（不避开它） |
| AvoidObstacles | 勾选时才启用这个自动避开遮挡物的功能 |
| DistanceLimit | 目标物与相机的检测距离限制。在这个距离限制内才会检测遮挡。设置为0即代表不限制 |
| MinimumOcclusionTime | 遮挡超过这个时长以后才调整相机避开遮挡物 |
| CameraRadius | 虚拟相机会尽量与遮挡物保持不低于这个距离。如果你看到由于较大的FOV导致相机穿透了遮挡物内部，可以适当增加这个值。 |
| Strategy | 避开遮挡物的策略。 |
| MaximumEffort | 最多处理多少个遮挡物。数字越大越影响性能，一般4足够了。 |
| SmoothingTime | 平滑时长。到避开遮挡位置至少会停留这么长时间。 |
| Damping | 不遮挡时相机回到正常位置的时长。 |
| DampingWhenOccluded | 机从正常位置移到避开遮挡位置的时长 |
| OptimalTargetDistance | 如果设置大于0的值，在计算相机位置时会更倾向于接近这个距离。 |

**Strategy**

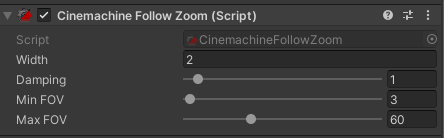
PullCameraForward ：将相机沿着自身的Z轴往前拉，直到在遮挡物前面。

PreserveCameraHeight ：保持相机的高度，同时把相机往前拉。

PreserveCameraDistance ：保持相对于目标物体的距离，同时把相机往前拉

**CinemachineFollowZoom**

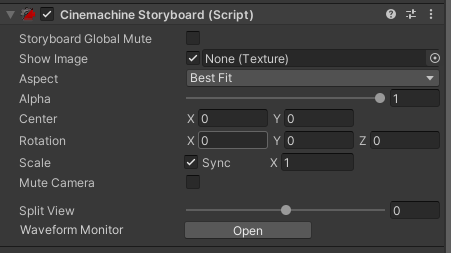
这个扩展的功能是：调整相机的FOV来保持物体在屏幕上的尺寸。



|  |  |
| --- | --- |
| 变量 | 含义 |
| Width | 在目标距离处以世界单位保持的射击宽度。 将尽可能调整 FOV 以将这个宽度保持在与相机的目标距离处。物体宽度 |
| Damping | 增加此值可柔化后续变焦的激进程度。 小数字更灵敏，大数字给出更重的响应缓慢的相机。 |
| MinFOV | 此行为将生成的 FOV 下限。 |
| MaxFOV | 此行为将生成的 FOV 上限。 |

**CinemachineStoryboard**

这个扩展的作用是：最后在屏幕上显示一张图片。

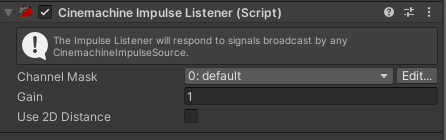


|  |  |
| --- | --- |
| 变量 | 含义 |
| StoryboardGlobalMute | 勾选时全局禁用，不显示所有CinemachineStoryboard的图片。 |
| ShowImage | 是否显示图片以及虚拟相机输出的图片是什么 |
| Aspect | 图片比例 |
| Alpha | 图片透明度 |
| Center | 图片中心的位置 |
| Rotation | 旋转 |
| Scale | 缩放比例，Sync勾选时宽高等比缩放 |
| MuteCamera | 勾选时，这个虚拟相机不会控制Unity相机（相当于只使用StoryBoard的功能） |
| SplitView | 水平擦拭图像 |
| Waveform Monitor | 可以用来查看图片的色阶 |

**CinemachineImpulseListener**

当你给一个虚拟相机添加Impulse Listener扩展时，这个虚拟相机就能对震动做出反应，最直观的就是相机晃动。 该组件需要配合下面提到的Cinemachine Impulse Listener脚本组件一起使用，可以很快的得出物体振动引发的相机颤抖特效。

**提示：**您可以创建自己的 Impulse Listener 以任何您喜欢的方式解释振动信号。



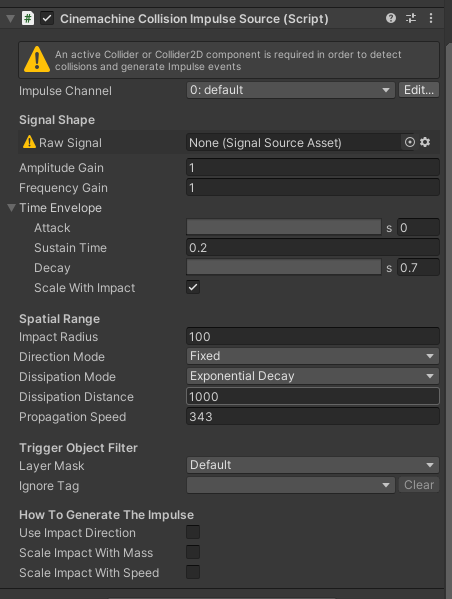
|  |  |
| --- | --- |
| 变量 | 含义 |
| ChannelMask | 未包含在Mask中的通道上的脉冲事件将被忽略。 |
| Gain | 脉冲的增益。1代表正常强度，0代表忽略这个信号。真实世界中，有的相机绑定在类似树枝上，会造成晃动的更厉害。这个值越大，晃动就会越多。 |
| Use2DDistance | 使用2D距离，忽略Z轴 |

**相关组件脚本（直接讲参数，如果想要了解原理可以到下面网站观看）**

**CinemachineCollisionImpulseSource**

该组件继承自CinemachineImpulseSource其区别在于该组件是碰撞体或者触发器来产生，而CinemachineImpulseSource根据事件产生

将**CinemachineCollisionImpulseSource**组件附加到具有 Collider 或 Collider2D 组件的**GameObject**。

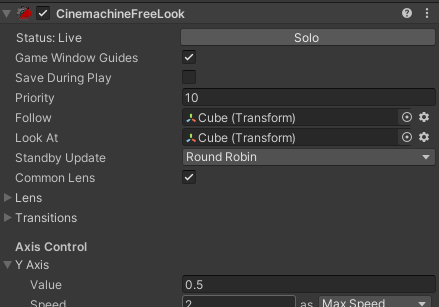


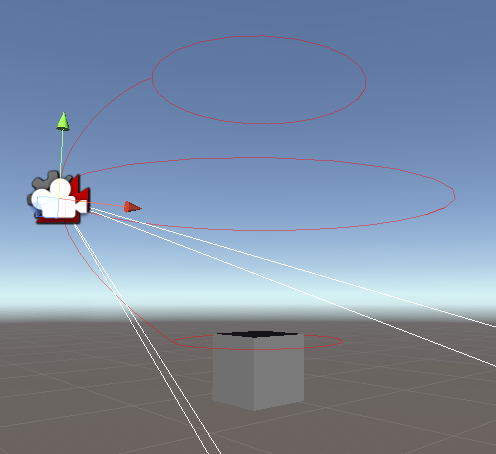
|  |  |
| --- | --- |
| 变量 | 含义 |
| Impulse Channel | 脉冲信号频道。从下拉列表中选择一个或多个频道。单击编辑可修改现有频道或添加新频道。 |
| Signal Shape | 信号形状 |
| Raw Signal | 原始信号形式。将信号资源拖到信号名称上以将信号连接到碰撞脉冲源。  单击“齿轮”图标可查看其他信号选项，包括用于创建新信号的选项。 |
| Amplitude Gain | 设置一个值，用于乘以**原始信号**的幅度。这控制了振动的强度。默认情况下设置为 1。使用大于 1 的值放大信号，使用 1 代表原始信号，使用小于 1 和大于 0 的值来衰减信号，使用 0 使信号静音。 |
| Frequency Gain | 设置一个值来乘以**原始信号**的频率。这控制了振动的速度。默认情况下设置为 1。使用大于 1 的值来增加频率，使用 1 表示原始信号，使用小于 1 和大于 0 的值来降低信号的频率。频率 0 无限期地保持单个噪声值，就好像时间被冻结了一样。 |
| Time Envelope | 时间包络设置，这些属性控制脉冲的持续时间和原始信号在该持续时间内的强度。 |
| Attack | 定义信号如何在波形开始处达到全幅度。单击预览窗格（灰色条）设置曲线的形状，并使用数字字段设置达到最大振幅的时间（以秒为单位）。将预览窗格留空以使用适合大多数用途的默认曲线。 |
| Sustain Time | 设置在起音后保持波形全振幅的时间。 |
| Decay | 定义信号如何从全振幅到波形末端的零。单击预览窗格（灰色条）设置曲线的形状，并使用数字字段设置信号静音的时间（以秒为单位）。将预览窗格留空以使用适合大多数用途的默认曲线。 |
| Scale With Impact | 启用此选项可将信号幅度缩放应用于**时间包络**。如果选中，Cinemachine Collision Impulse Source 组件会根据冲击强度缩放时间包络：更强的冲击持续时间更长，而更温和的冲击持续时间更短。 |
| Spatial Range | 空间范围。这些属性定义了场景中的区域。来自这个脉冲源的脉冲只影响这个区域的脉冲监听器。 |
| Impact Radius | 设置场景中信号保持全振幅的撞击点周围的空间半径。除此之外，信号强度在耗散距离上逐渐减弱。之后，强度为零。换句话说，信号的总影响半径为**Impact Radius** + **Dissipation Distance**。 |
| Direction Mode | 定义 Cinemachine Collision Impulse Source 组件在 Impulse Listener 远离 Impulse Source 时应如何应用信号方向。 |
| Dissipation Mode | 定义当听者在**Impact Radius**之外时信号如何消散。 |
| Dissipation Distance | 设置超出冲击半径的距离，在该距离内信号从全强度衰减到无。 |
| Trigger Object Filter | 触发对象过滤器。这些属性过滤在碰撞或进入触发区域时触发脉冲的游戏对象。**图层蒙版**指定的**图层中的**游戏对象会触发脉冲，除非您使用**Ignore Tag**值标记它们。 |
| Layer Mask | 这些层中的刚体组件与脉冲源碰撞或进入触发区域会导致源广播其信号 |
| Ignore Tag | 带有此标签的游戏对象不会产生脉冲，即使它们位于**Layer Mask 中**指定的**层中**。 |
| How To Generate The Impulse | 通过以冲击速度移动的冲击质量在某个位置触发脉冲。Unity 动态计算触发脉冲的 Rigidbody 或 Rigidbody 2D 组件的质量和速度。该如何产生脉冲，通过控制刚体的质量和速度如何影响信号的强度和方向。 |
| Use Impact Direction | 启用此设置可旋转脉冲信号以对应冲击速度的方向。例如，如果原始信号垂直振动但冲击方向是水平的，您可以检查此属性以使生成的脉冲信号水平振动。  禁用以使用原始信号的方向，而不管冲击方向。 |
| Scale Impact With Mass | 启用此设置可根据碰撞游戏对象的质量增加或减少脉冲信号的幅度。质量在包含 Cinemachine 脉冲源的游戏对象和碰撞游戏对象的 Rigidbody 或 RigidBody2D 组件中指定。禁用此设置以使用恒定质量 1。 |
| Scale Impact With Speed | 启用此设置可根据冲击速度缩放信号幅度。移动得更快的游戏对象具有更大的动量，因此会产生更强的信号。禁用此设置以忽略碰撞游戏对象的速度。 |

**四、 日常开发常用的Camera组件**

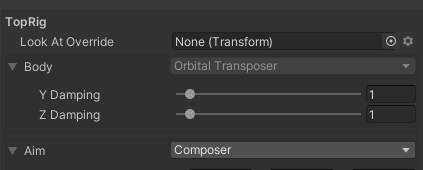
**CinemachineFreeLook**

面向第三人称摄像机体验的 Cinemachine 摄像机。相机围绕其对象运行，三个独立的相机装备定义了围绕目标的环。每个装备都有自己的半径、高度偏移、合成器和镜头设置。根据相机沿连接这三个装备的样条线的位置，对这些设置进行插值以给出最终的相机位置和状态。





其本质的参数与VerticalCamera一致，只是将3个VerticalCamera封装为一个，由Top，Middle，Bottom三个部分操控。都可以选择自己的Soft Zone ，Dead Zone等各种参数数值



其主要特点是3个子空间的body都是选择OrbitalTransposer算法跟随

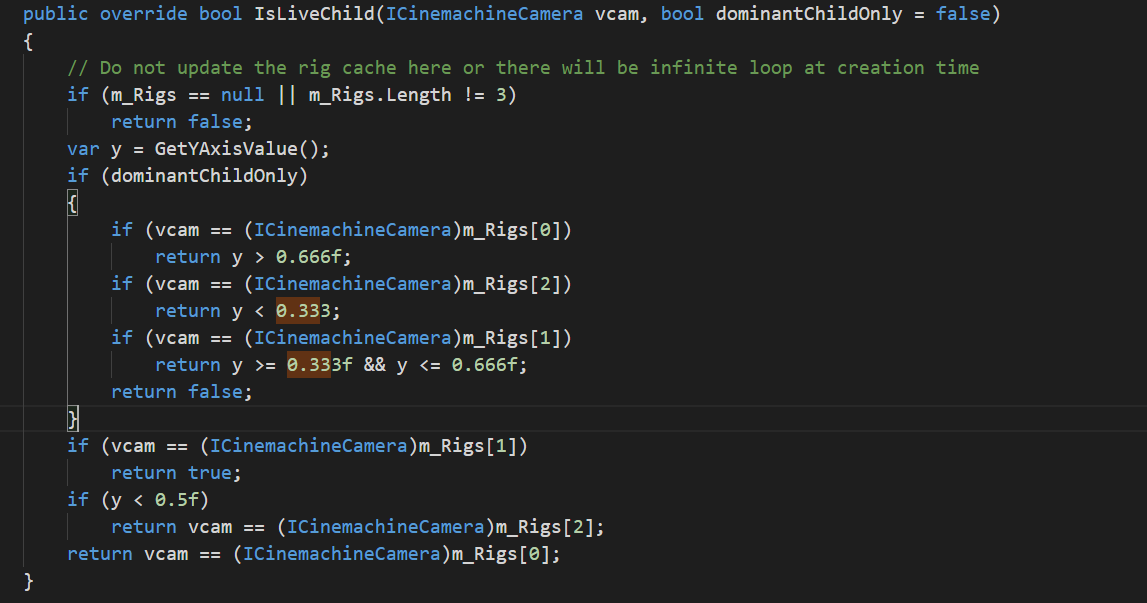
其Aim默认选择是Composer，可以选择其他观察模式

关键在于什么时候切换对应的子虚拟相机

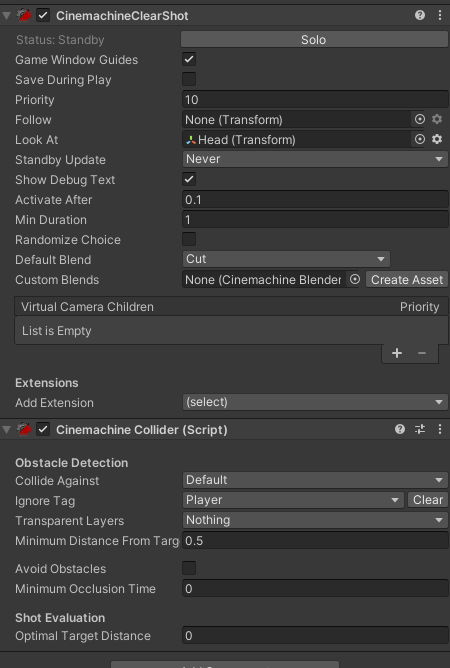
可以在游戏运行时观察

Generated

源码里是这样切换的



**Cinemachine Clear Shot Camera**



点击生成CinemachineClearShotCamera，会产生这两个组件。使用 Clear Shot 设置场景的复杂多摄像机覆盖范围，以保证目标的清晰视野。这可能是一个非常强大的工具。 带有 Cinemachine Collider 扩展的虚拟摄像机子项分析场景中的目标障碍物、最佳目标距离等。 Clear Shot 使用此信息来选择要激活的最佳子虚拟相机。

提示：要为所有虚拟摄像机子项使用单个 Cinemachine Collider，请将 Cinemachine Collider 扩展添加到 ClearShot GameObject 而不是其每个虚拟摄像机子项。 这个 Cinemachine Collider 扩展适用于所有孩子，就好像他们每个人都有那个 Collider 作为自己的扩展。

如果多个子摄像机具有相同的拍摄质量，则 Clear Shot 摄像机选择具有最高优先级的摄像机。

您还可以在 ClearShot 子虚拟相机之间定义自定义混合。

Cinemachine Collider已经介绍过了

|  |  |
| --- | --- |
| 变量 | 含义 |
| **Look At** | 子虚拟相机移动的默认跟随的目标游戏对象。 当孩子未指定此目标时，Clear Shot 相机使用此目标。 如果所有孩子都定义了自己的目标，则可能为空 |
| **Follow** | 将 Unity 相机瞄准的目标 GameObject。当子虚拟相机未指定此目标时，Clear Shot 相机使用此目标。如果所有孩子都定义了自己跟踪的目标，则可能为空。 |
| **Show Debug Text** | 选中以显示实时虚拟摄像机的文本摘要并融入游戏视图。 |
| **Activate After** | 在激活新的子摄像头之前等待几秒钟。 |
| **Min Duration** | 一个活动的相机必须至少在这么多秒内处于活动状态，除非更高优先级的相机变为活动状态。 |
| **Randomize Choice** | 如果多个相机具有相同的拍摄质量，请选中以随机选择一个相机。取消选中以使用子虚拟摄像机的顺序及其优先级。 |
| **Default Blend** | 当您没有明确定义两个虚拟摄像机之间的混合时使用的混合。 |
| **Priority** | 根据其他摄像机和本摄像机的状态确定哪个摄像机处于活动状态。  更高的数字具有更高的优先级。 |
| **Virtual Camera Children** | 子虚拟相机挂载在这里 |

**Cinemachine and Timeline**

Cinemachine很容易和Timeline结合使用，比直接使用相机动画制作镜头更容易。使用Timeline可以激活、停用、混合虚拟相机。Timeline可以将Cinemachine、场景中的GameObject和其他资产组合在一起，以可视化方式创建、调整出丰富的过场动画，甚至是交互式的过场动画。

提示：对于简单的镜头序列，可以不使用Timeline，直接使用Cinemachine中的Blend List Camera。

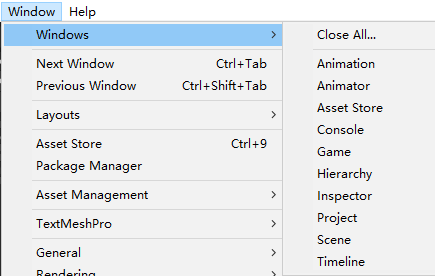
使用Timeline时，Timeline会控制哪个相机激活，覆盖Cinemachine Brain基于虚拟相机优先级的决策。当Timeline播放完后，控制权返还给Cinemachine Brain，它会激活优先级最高的虚拟相机。

你可以使用Cinemachine Shot Clip在Timeline中控制虚拟相机。每个镜头片段都指向一个虚拟相机，Timeline播放时将其激活，播放后再将其禁用。使用一系列镜头片段来指定每个镜头的顺序和持续时间。

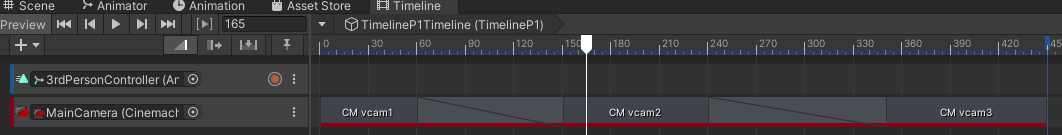
想要两个虚拟相机之间直接切换，可以将clip相邻放置。想要两个虚拟相机之间混合过渡，可以重叠片段。

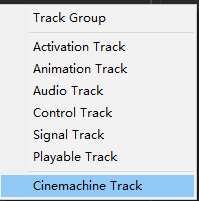
下面为如何设置Timeline来创建相机动画步骤：

在资产界面生产一个EmptyObject，将其命名



点击Timeline生成，该EmptyObject会自动生成Timeline



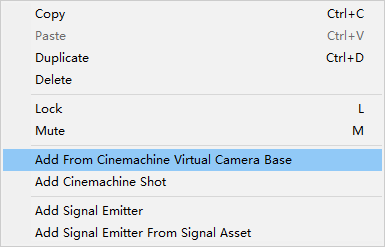


点击CinemachineTrack可以创建相应的轨道

蓝色为Animator轨道，红色为VirtualCamera轨道

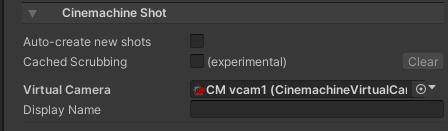
要将 Cinemachine 镜头剪辑添加到 Cinemachine 轨道：

1. 在 Cinemachine Track 中，右键单击并选择**Add Cinemachine Shot Clip**。

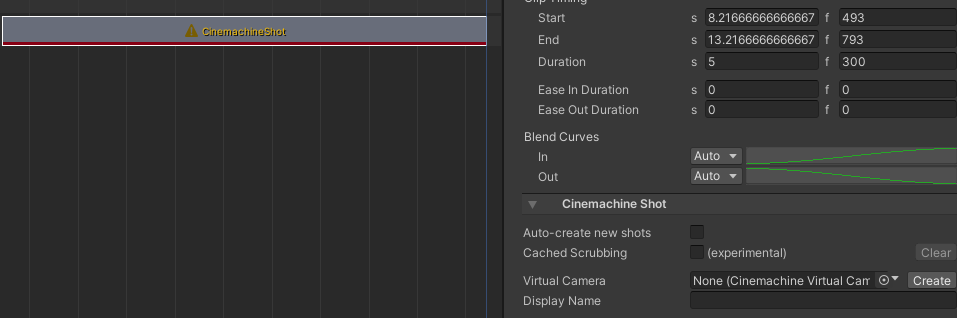


1. 执行以下操作之一：

* 要将现有的虚拟摄像机添加到镜头剪辑中，请将其拖放到 Cinemachine Shot 组件中的虚拟摄像机属性上。Add From Cinemachine Virtual Camera Base，这会自动将选择的虚拟相机设置到Cinemachine Shot组件的Virtual Camera属性中。



* 要创建新的虚拟摄像机并将其添加到镜头剪辑中，请单击 Cinemachine Shot 组件中的创建。要创建一个新的虚拟相机并将其添加到镜头片段中，选择Add Cinemachine Shot，在Cinemachine Shot组件中单击Create按钮。

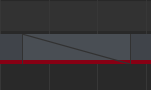


然后可以在时间轴中拖动相应的虚拟机模块

这种两个虚拟相机相切的叫做直接切换

Generated

这种两个虚拟相机混合的会产生相机之间切换动画（混合）



在Timeline编辑器中，调整镜头片段的顺序、持续时间、切换和混合。

调整虚拟相机的属性，主要是设置跟随Follow或者瞄准的Aim的对象。

如果要设置虚拟相机属性的动画，给它创建动画轨道，然后像其他GameObject一样录制动画即可。

**后续可能会继续更新。。。。**

**附录**

本文档用到的部分参考资料和代码均来自以下地方

1. <https://docs.unity3d.com/Packages/com.unity.cinemachine@2.6/manual/index.html>
2. <https://docs.unity3d.com/Packages/com.unity.cinemachine@2.2/manual/index.html>
3. 作者：大智\_Unity技术探路者 https://www.bilibili.com/read/cv4702573 出处：bilibili
4. Unity引擎Cinemachine脚本源码