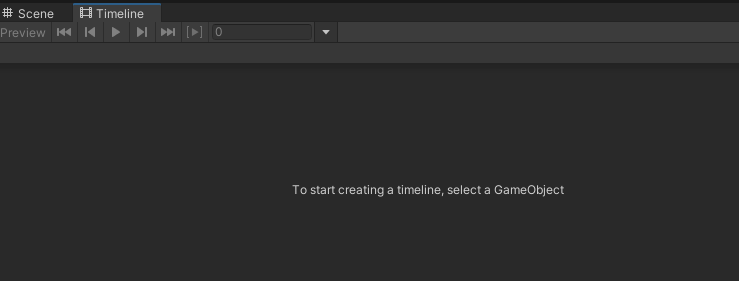
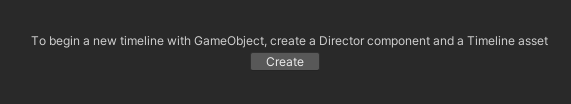
1. 创建Timeline：打开面板Window->Sequencing->Timeline

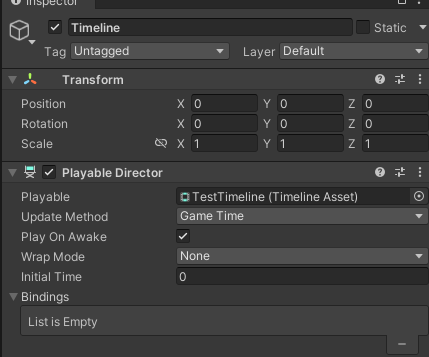


(1.1)选择一个要添加 Timeline 的物体，我创建一个物体就叫 Timeline(可以随意命名)，选择Timeline，然后在面板上显示 Create 按钮，如下

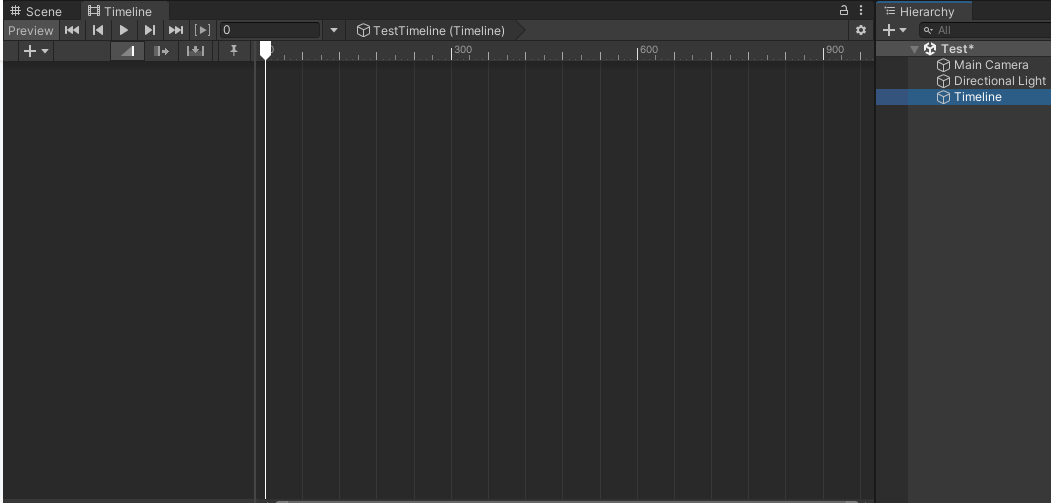


点 Create 按钮，保存一个 \*.playable 的文件,然后自动给Timeline物体添加一个 PlayableDirector 脚本

脚本参数 Playable ：TestTimeline 就是创建的 TestTimeline.playable 文件

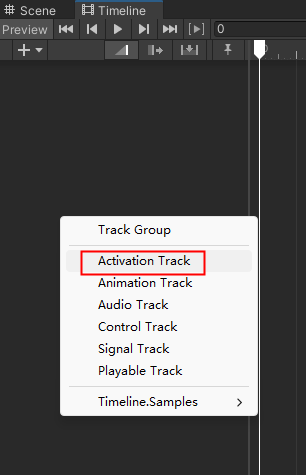


选择 Timeline 面板如下

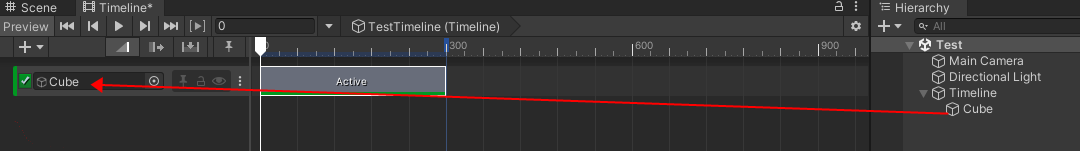


2.创建Activation Track (显示/隐藏物体轨道)

(2.1)在左侧空白处，鼠标右键，选择 ActivationTrack



(2.2) 创建一个 Cube，为了方便管理，放在 Timeline 下方

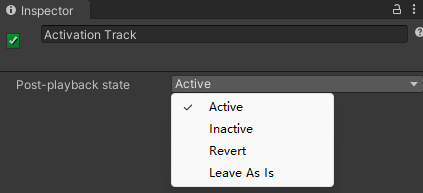


然后拖拽到 Activation 轨道的目标物体上

右侧Active区域是Cube物体显示的时间区域（Timeline执行到这个时间段，物体是显示的,在这个时间段之外时，物体都是隐藏的）

可以左右拖动，可以把鼠标放在左右两侧拉长时间区域

(2.3)点击左侧对勾区域，Post-playback state



Active状态：Timeline完成播放时，将绑定的游戏对象的状态设置为激活(显示)状态

Inactive状态：Timeline完成播放时，将绑定的游戏对象的状态设置为停用(隐藏)状态

Revert状态：Timeline开始播放之前，将绑定的游戏对象的状态设置为停用(隐藏)状态

Leave As Is状态：Timeline完成播放时，绑定的对象状态是显示的就设置为显示状态，Timeline完成播放时，绑定的对象状态是隐藏的，就设置为隐藏状态

名词解释：

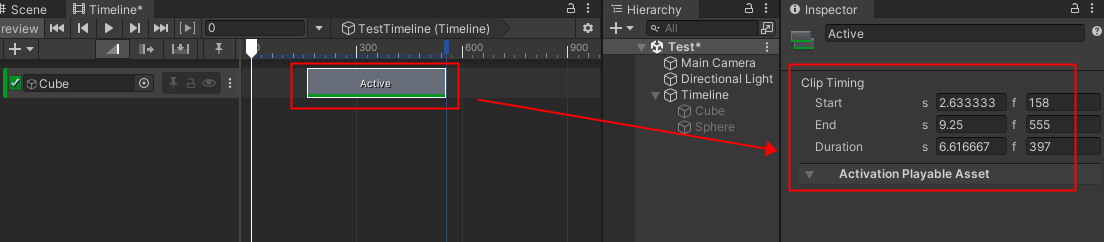
**Timeline完成播放时：**Timeline播放时间到，所有轨道结束时，才是**Timeline**完成播放时

(2.4)选择轨道上的 Active，在 Inspector 面板上可以看到参数

Start：开始时间(秒) f：帧

End：结束时间(秒)

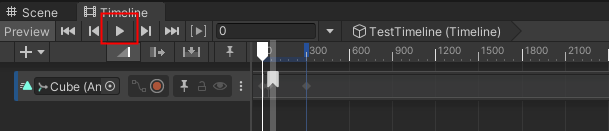
Duration：持续时间(秒)



预览：

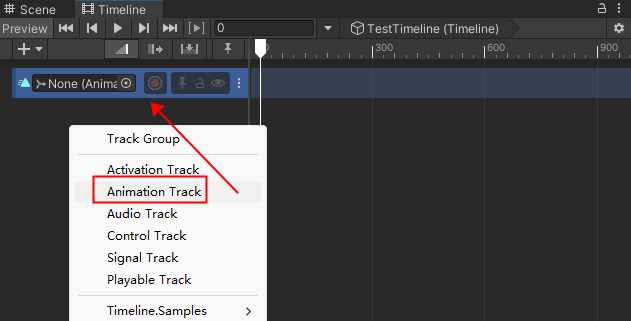
(2.4.1)选择挂Playable Director 脚本的GameObject，在 Timeline窗口左上角，点击三角预览播放

(2.4.2)直接运行Unity也可以预览播放

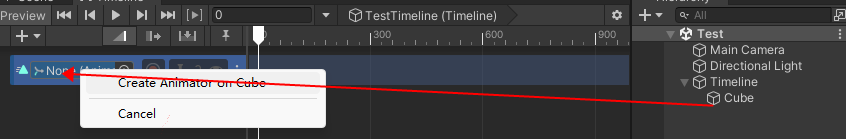


1. 创建 Animation Track (动画轨道)

(3.1)空白位置鼠标右键，选择 Animation Track



创建一个 Cube 拖拽到 None 位置



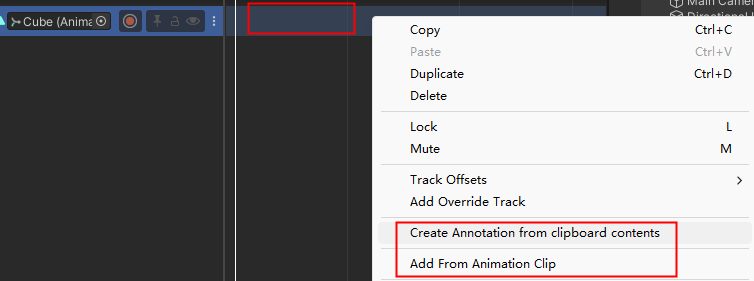
提示 Create Animator on Cube：添加Animator 组件到Cube上，点击Create 即可，自动在 Cube 上挂一个Animator 组件

(3.2)添加动画帧

在轨道空白位置鼠标右键

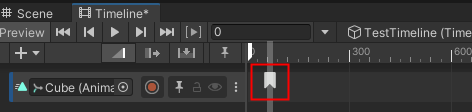
Create Annotation from clipboard contents：创建一个动画帧

AddFrom Animation Clip：直接选择一个现有的动画帧



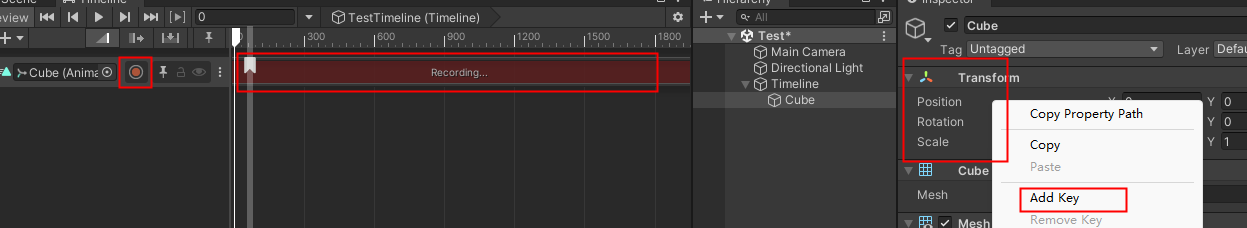
点击 AddFrom Animation Clip弹出选择动画帧面板

点击 Create Annotation from clipboard contents 在动画轨道上出现标签如下

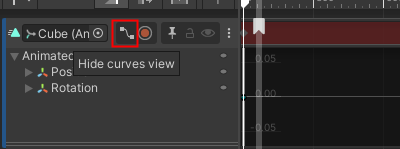


(3.3) 创建 Create Annotation from clipboard contents 后，点击左侧红色圆圈，开始录制，在轨道上显示红色并且有 Recoding... 字体提示

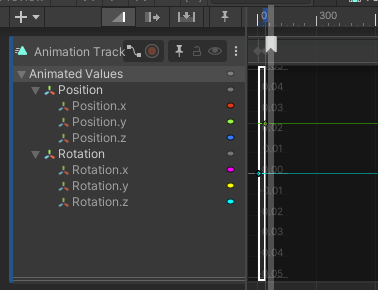
然后选择绑定的对象 Cube，在Inspector面板上，Transform 位置，可以分别在 Position、Rotation、Scale 鼠标右键 弹框选择 AddKey



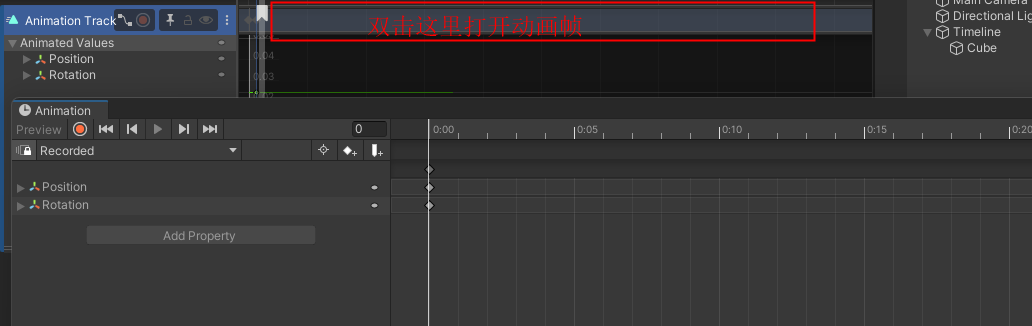
点击 AddKey将在左侧轨道下方出现选择的属性，点击下方图中红色位置，可以显示隐藏编辑属性



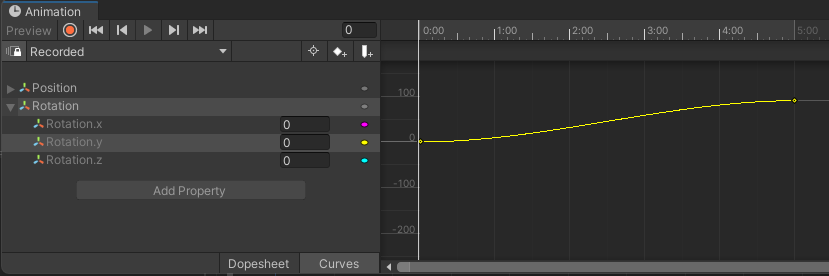
展开如下



在轨道区域双击打开动画编辑窗口

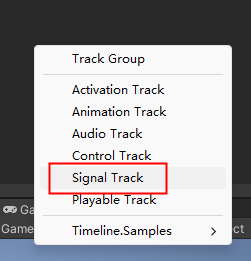


切换到 Curves 编辑动画



5.Signal Track：信号轨道，发射信号，相当于发送一个事件

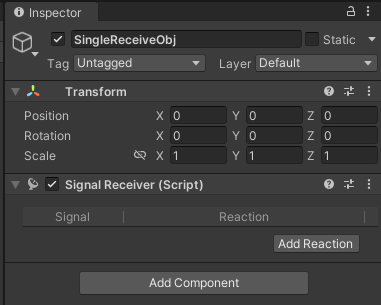
(5.1)在TimeWindow面板左侧空白区域，右键选择 Signal Track



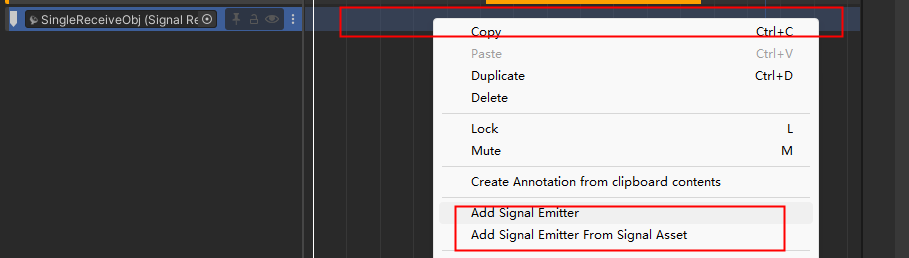


创建后显示如上，

（5.2）创建一个接收信号的 GameObject，我命名为 SingleReceiveObj，然后拖拽到左侧 None(Signal Receiver) 位置，提示添加 Create Signal Receiver 脚本，选择 Create 即可在 GameObject 上自动挂 Signal Receiver 脚本，如下



(5.3) 在信号轨道空白处右键添加一个发射器



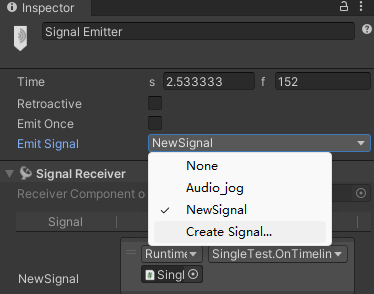
Add Signal Emitter：创建一个标签，然后在 Inspector面板上Emit Signale 位置可以选择已经创建的发射器

Add Signal Emitter From Signal Asset: 会打开选择窗口，可以选择已经创建的发射器

发射器显示如下



点击信号发射器看Inspector 面板



Time：发射信号的时间

Retroactive：

EmitOnce：勾选后只会发射一次信号

Emit Signal：选择一个发射信号Asset，可以复用现有的，也可以选择 Create Signal 创建一个

(5.4)配置接收函数

// 创建一个接收信号的脚本，类名随意

public class SignalReceiveTest : MonoBehaviour

{

public void OnTimelineSignal()

{

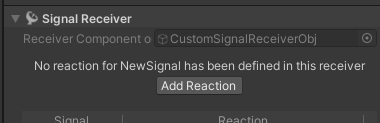
Debug.LogError("哈哈收到信号了");

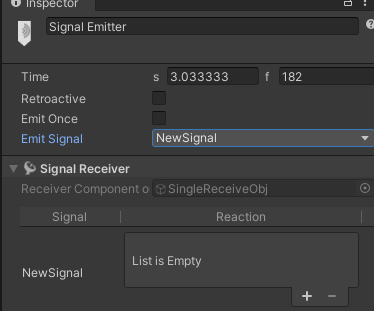
}

}

将 SignalReceiveTest 脚本挂到 SingleReceiveObj 物体上

在 Signal Receiver 下方点击 AddReaction

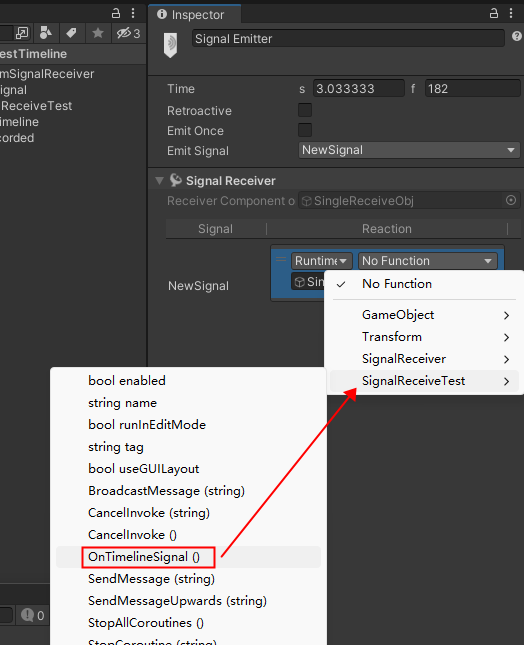




点击 +

选择 Runtime，在 NoFunction 处下拉选择 SignalReceiveTest 脚本上的 OnTimelineSignal 函数

运行当时间轴执行到信号发射器位置时，将会调用到 OnTimelineSignal 函数



6.自定义信号

（6.1）新建信号发射类，继承 SignalEmitter

using UnityEngine.Playables;

using UnityEngine.Timeline;

// 自定义的信号

public class CustomSignal : SignalEmitter

{

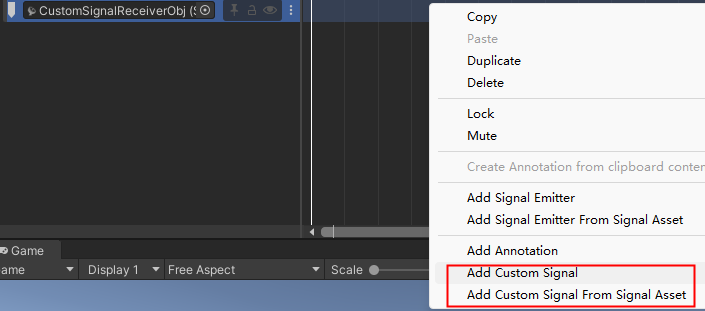
// 自定义参数

public int number;

public int param;

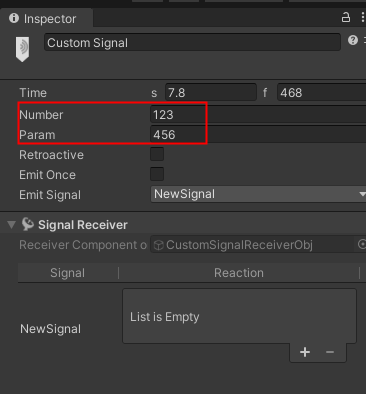
}

创建一个 Signal Track 轨道，然后在右侧右键，如下可以看到新建的信号发射器



创建一个接收信号的 GameObject 我命名为 CustomSignalReceiverObj，添加 SngnalReceiver 脚本，并拖拽到信号轨道

(6.2)选择新创建的 Custom Signal，在Inspector 面板可以看到 CustomSignal 类的两个参数 number 和 param，可以手动输入参数值



(6.3) 创建信号接收类

/// <summary>

/// 自定义信号接收器

/// </summary>

public class CustomSignalReceiver : MonoBehaviour, INotificationReceiver

{

public void OnNotify(Playable origin, INotification notification, object context)

{

var signal = notification as CustomSignal;

if (signal != null

&& signal.asset != null)

{

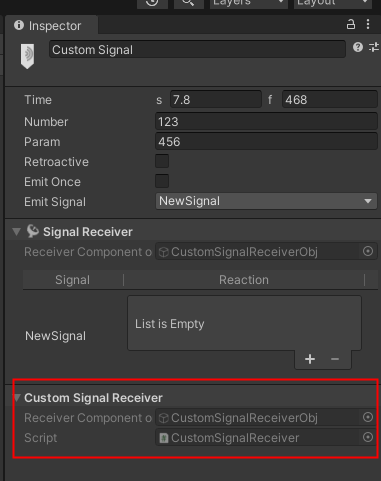
Debug.LogError(signal.number + " " + signal.param);

}

}

}

将脚本 CustomSignalReceiver 挂到 CustomSignalReceiverObj 上，再选择Custom Signal 查看Inspector面板下方显示 CustomSignalReceiver



运行将自动触发 CustomSignalReceiver 类的 OnNotify 函数