**Animator Controllers**

Animator Controllers允许您为角色或其他动画游戏对象安排和维护一组动画。

控制器会引用其中所用的动画剪辑，使用所谓的\_\_状态机\_\_来管理各种动画状态和它们之间的过渡；状态机可视为一种流程图，或者是在 Unity 中使用可视化编程语言编写的简单程序。

以下部分将介绍 Mecanim 为动画的控制和排序提供的主要功能。

# Animator Controller资源

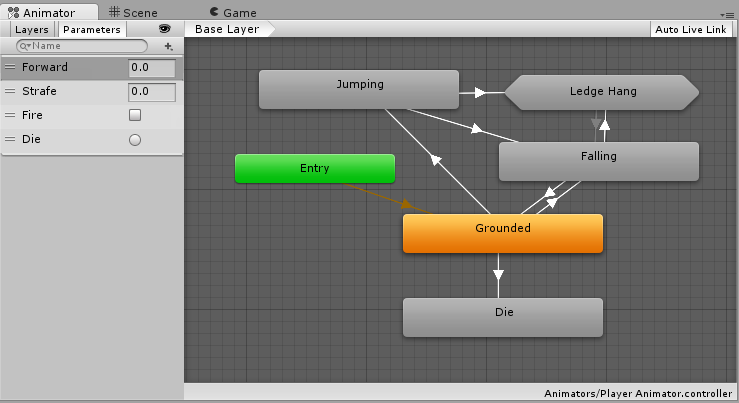
当您准备好使用动画剪辑时，需要使用\_\_Animator Controller\_\_使这些剪辑结合在一起。Animator Controller资源是在 Unity 内创建的，允许您为角色或对象维护一组动画。

项目文件夹中的Animator Controller资源

从 Assets 菜单或从 Project 窗口中的 Create 菜单创建Animator Controller资源。

在大多数情况下，拥有多个动画并在满足某些游戏条件时在这些动画之间切换是很常见的。例如，只要按下空格键，就可以从行走动画切换到跳跃动画。但是，即使您只有一个动画剪辑，仍需要将其放入 Animator Controller 以便将其用于游戏对象。

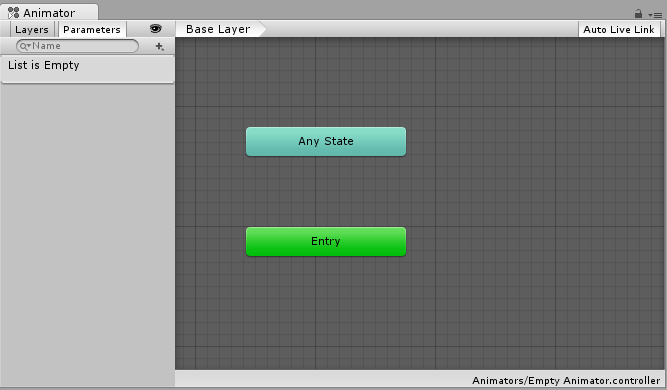
控制器使用所谓的\_\_状态机\_\_来管理各种动画状态和它们之间的过渡；状态机可视为一种流程图，或者是在 Unity 中使用可视化编程语言编写的简单程序。可在[此处](file:///E:\\UnityDocumentation\\Manual\\AnimationStateMachines.html)找到有关状态机的更多信息。可在 [Animator 窗口](file:///E:\UnityDocumentation\Manual\AnimatorWindow.html)中创建、查看和修改Animator Controller的结构。

简单的Animator Controller

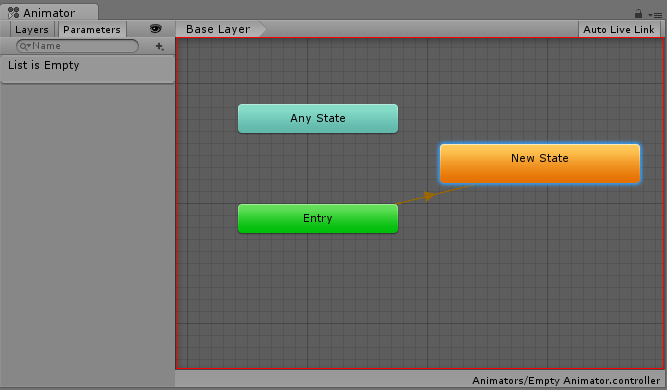
最终会通过连接\_\_Animator\_\_组件（其中引用了Animator Controller）将Animator Controller应用于对象。请参阅关于 [Animator](file:///E:\UnityDocumentation\Manual\class-Animator.html) 组件和 [Animator Controller](file:///E:\UnityDocumentation\Manual\class-AnimatorController.html) 的参考手册页面来了解有关其用法的更多详细信息。

**Animator 窗口**

在 Animator 窗口中可创建、查看和修改 Animator Controller 资源。

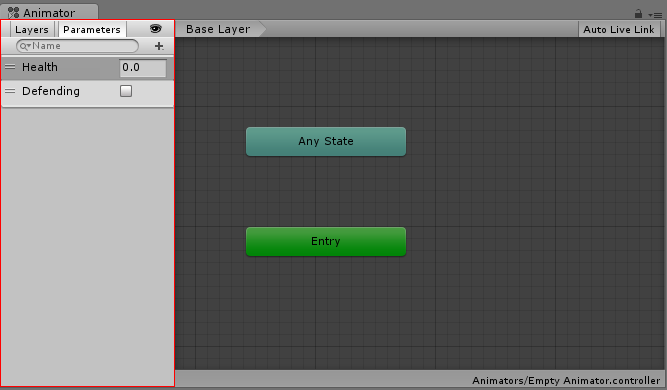
Animator 窗口显示了新的空 Animator Controller 资源

Animator 窗口有两个主要部分：主要网格化布局区域以及左侧 Layers 和 Parameters 面板。

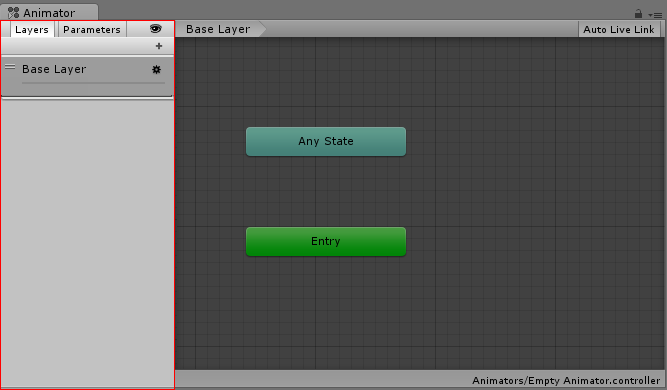
Animator 窗口的布局区域

带有深灰色网格的主要部分是布局区域。使用此区域可在[Animator Controller](file:///E:\UnityDocumentation\Manual\Animator.html)中创建、排列和连接状态。

您可以右键单击该网格以创建新的状态节点。使用鼠标中键或按住 Alt/Option 键拖拽可平移视图。通过单击可选择状态节点以进行编辑，而通过单击并拖动状态节点可重新排列状态机的布局。

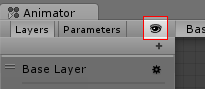
Parameters 视图中已创建两个示例参数。

左侧面板可在 Parameters 视图和 Layers 视图之间切换。Parameters 视图 允许您创建、查看和编辑 [Animator Controller 参数](file:///E:\UnityDocumentation\Manual\AnimationParameters.html)。这些参数是您定义的变量，充当状态机的输入。要添加参数，请单击加号图标，然后从弹出菜单中选择参数类型。要删除参数，请在列表中选择参数，然后按 Delete 键（在 macOS 上使用 fn-Delete 来删除所选参数）。

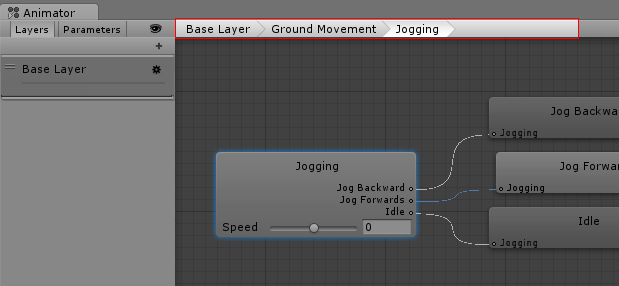
Layers 视图

左侧面板切换到 Layers 视图时，您可以在 Animator Controller 中创建、查看和编辑[层](file:///E:\\UnityDocumentation\\Manual\\AnimationLayers.html)。因此，可在单个动画控制器中同时运行多个动画层，每个动画层由一个单独状态机控制。此情况的常见用途是在控制角色一般运动动画的基础层之上设置一个单独层来播放上身动画。

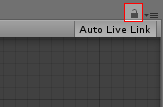
要添加层，请单击加号图标。要删除层，请选择该层并按 Delete 键。

Layers 和 Parameters 隐藏图标

单击“眼睛”图标开关即可显示或隐藏 Parameters 和 Layers 侧面板，从而为您提供更多空间来查看和编辑状态机。

分层示踪导航位置

当前状态机内的“示踪导航”分层位置。状态可包含[子状态](file:///E:\\UnityDocumentation\\Manual\\NestedStateMachines.html)和[树](file:///E:\UnityDocumentation\Manual\class-BlendTree.html)，这些结构可以反复嵌套。当进入子状态时，此处会列出父状态的层级视图以及当前查看的状态。单击父状态可跳回到父状态或直接返回到状态机的基础层。

锁图标

启用锁图标将使 Animator 窗口聚焦于当前状态机。锁图标为关闭状态时，如果单击新的 Animator 资源或带有 Animator 组件的游戏对象，则会切换 Animator 窗口以显示该项的状态机。锁定该窗口后，无论选择其他任何资源或游戏对象，Animator 窗口均一直显示同一状态机。

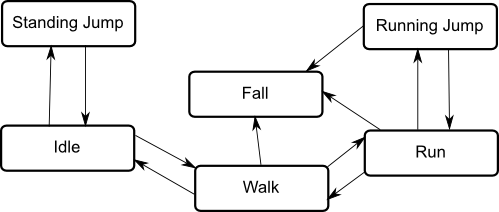
# 动画状态机

角色或其他动画游戏对象通常具有若干不同的动画，这些动画对应于该角色或对象可在游戏中执行的不同动作。例如，角色可以在空闲时轻微呼吸或摇摆，在得到指令时行走，以及从平台上跌落时恐慌地抬起手臂。一扇门可能具有打开、关闭、卡住和撞开动作的动画。Mecanim 使用类似于流程图的可视化布局系统来表示\_\_状态机\_\_，从而控制需要在角色或对象上使用的动画剪辑并对这些动画剪辑排序。本部分将提供有关 Mecanim 状态机的更多详细信息，并说明如何使用这些状态机。

# 状态机基础

状态机的基本思想是使角色在某一给定时刻进行一个特定的动作。动作类型可因游戏类型的不同而不同，常用的动作包括空闲、行走、跑步、跳跃等，其中每一个动作被称为\_\_一种状态**。在某种意义上，角色处于行走、空闲或者其它的“状态”中。一般来说，角色从一个状态立即切换到另一个状态是需要一定的限制条件的。比如角色只能从跑步状态切换到跑跳状态，而不能直接由静止状态切换到跑跳状态。角色从当前状态进入下一个状态的选项被称为**[状态过渡条件](file:///E:\UnityDocumentation\Manual\class-Transition.html)**。状态集合、状态过渡条件以及记录当前状态的变量放在一起，形成了一个\_\_状态机**。

状态及其过渡条件可以通过图形来表达，其中的节点表示状态，而弧线（节点间的箭头）表示状态过渡。您可以将当前状态视为放置在节点之一上的标记或亮点，然后只能沿箭头之一跳转到另一个节点。

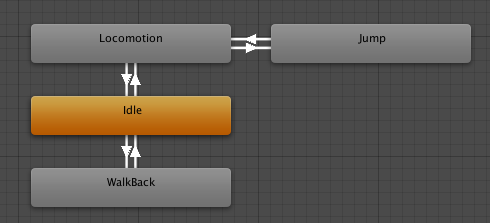


状态机对于动画的重要意义在于用户可以通过很少的代码对状态机进行设计和升级。每一个状态有一个与之关联的运动，只要状态机处于此状态，就会播放此运动。从而让动画师或设计师方便地定义动作顺序，而不必去关心底层代码的实现。

## 状态机

Unity的动画状态机提供了一种纵览角色所有动画剪辑的方法，并且允许通过游戏中的各种事件（如用户输入）来触发不同的动画效果。

动画状态机可以通过 [Animator Controller 窗口](file:///E:\UnityDocumentation\Manual\Animator.html)来创建，这些状态机如下所示：



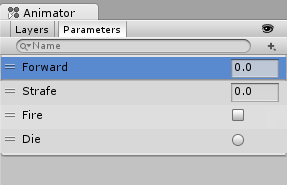
动画状态机包括\_\_动画状态**、**动画过渡\_\_和\_\_动画事件\_\_，而复杂的状态机还可以含有简单的子状态机。请参阅[动画状态](file:///E:\UnityDocumentation\Manual\class-State.html)和[动画过渡](file:///E:\UnityDocumentation\Manual\class-Transition.html)的参考页面来了解更多信息。

# 动画参数

动画参数是在 Animator Controller 中定义的变量，可从脚本访问这些变量并向其赋值。这是脚本控制或影响状态机流程的方法。

例如，可通过[动画曲线](file:///E:\UnityDocumentation\Manual\animeditor-AnimationCurves.html)更新参数的值，然后从脚本访问参数以便可改变音效的音高（就像它是一段动画一样）。同样，脚本可设置被 Mecanim 拾取的参数值。例如，脚本可设置参数来控制[混合树](file:///E:\UnityDocumentation\Manual\class-BlendTree.html)。

可使用 Animator 窗口的 Parameters 部分来设置默认参数值（可在 Animator 窗口的右上角进行选择）。这些参数可分为四个基本类型：



* Int - 整数
* Float - 带小数部分的数字
* Bool - true 或 false 值（由复选框表示）
* Trigger - 当被过渡使用时，由控制器重置的布尔值参数（以圆形按钮表示）

可使用以下 Animator 类中的函数，从脚本为参数赋值：[SetFloat](file:///E:\UnityDocumentation\ScriptReference\Animator.SetFloat.html)、[SetInt](file:///E:\\UnityDocumentation\\ScriptReference\\Animator.SetInteger.html)、[SetBool](file:///E:\\UnityDocumentation\\ScriptReference\\Animator.SetBool.html)、[SetTrigger](file:///E:\\UnityDocumentation\\ScriptReference\\Animator.SetTrigger.html) 和 [ResetTrigger](file:///E:\\UnityDocumentation\\ScriptReference\\Animator.ResetTrigger.html)。

以下是基于用户输入和碰撞检测而修改参数的脚本示例。

using UnityEngine;

using System.Collections;

public class SimplePlayer : MonoBehaviour {

Animator animator;

// 使用此函数进行初始化

void Start () {

animator = GetComponent<Animator>();

}

// 每帧调用一次 Update

void Update () {

float h = Input.GetAxis("Horizontal");

float v = Input.GetAxis("Vertical");

bool fire = Input.GetButtonDown("Fire1");

animator.SetFloat("Forward",v);

animator.SetFloat("Strafe",h);

animator.SetBool("Fire", fire);

}

void OnCollisionEnter(Collision col) {

if (col.gameObject.CompareTag("Enemy"))

{

animator.SetTrigger("Die");

}

}

}

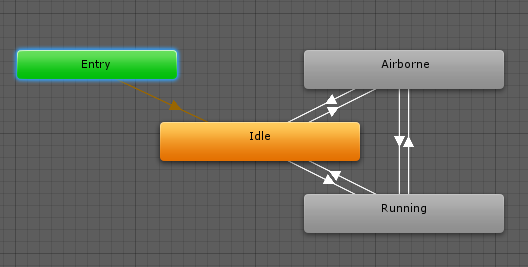
# 状态机过渡

状态机过渡可帮助您简化大型或复杂的状态机。允许对状态机逻辑进行更高级的抽象化。

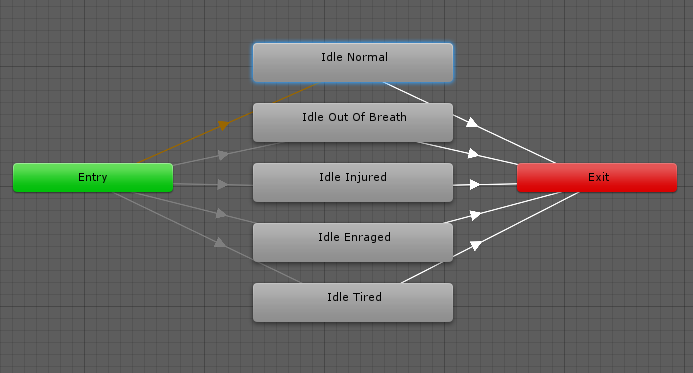
Animator 窗口中的每个视图都有一个进入 (Entry) 和退出 (Exit) 节点。在状态机过渡期间使用这些节点。

过渡到状态机时使用进入节点。进入节点将接受评估，并根据设置的条件分支到目标状态。通过此方式，进入节点可以通过在状态机启动时评估参数的状态来控制状态机的初始状态。

因为状态机始终具有默认状态，所以始终会有从进入节点分支到默认状态的默认过渡。

具有单个默认进入过渡的进入节点

随后可添加从进入节点到其他状态的其他过渡来控制状态机是否应以其他状态开始。

具有多个进入过渡的进入节点

退出节点用于指示状态机应退出。

状态机中的每个子状态都被视为一个独立且完整的状态机，因此通过使用这些进入和退出节点，可以更简练地控制从顶级状态机到其子状态机的流程。

可将状态机过渡与常规状态过渡混合，因此可在状态之间过渡、从状态过渡到状态机以及从一个状态机直接过渡到另一个状态机。