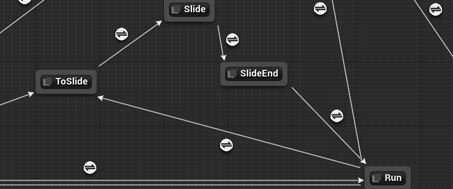
**状态机**

**State Machines（状态机）** 提供了一种图形化的方式来把骨架网格物体的动画分解为一系列 *状态* 。然后，通过 *Transition Rules（变换规则）* 来管理这些状态，这些变换规则控制如何从一个状态混合到另一个状态。它们作为一个工具，大大地简化了骨架网格物体动画的设计过程，因为您可以创建一个图表，使用它来控制您的角色在不同动画类型间的变换，而不必创建复杂的蓝图网络。

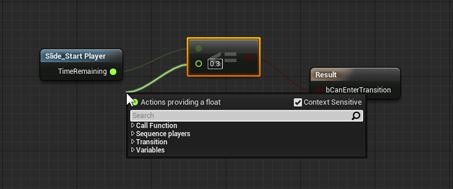
它们也简化了您的动画图表构想过程。当您设计动画时，可以先从构思所需要的状态及角色如何从一个状态变换到另一个状态开始。最简单的一种构思方法就是，把您的角色的所有动画分解为一个易读的流程成图，在这里您可以看到骨架网格物体的状态在该图表中的移动走势。

比如，当一个角色不动时，可能有一个“空闲”状态。一旦角色的速度超过某个特定的值，那么角色可以混合到另一个“移动”状态。还可以有很多种其他状态，比如下降、跳跃、蹲下、疾跑等。通过使用状态机，动画师和美术师可以快速地查看其动画网络的数据流动情况。

[[](http://api.unrealengine.com/CHN/Engine/Animation/StateMachines/CreatingStateMachines/index.html)](http://api.unrealengine.com/CHN/Engine/Animation/StateMachines/CreatingStateMachines/index.html)

**[创建状态机](http://api.unrealengine.com/CHN/Engine/Animation/StateMachines/CreatingStateMachines/index.html)**

[关于创建在动画资源的动画图表中应用的新状态机的指南。](http://api.unrealengine.com/CHN/Engine/Animation/StateMachines/CreatingStateMachines/index.html)

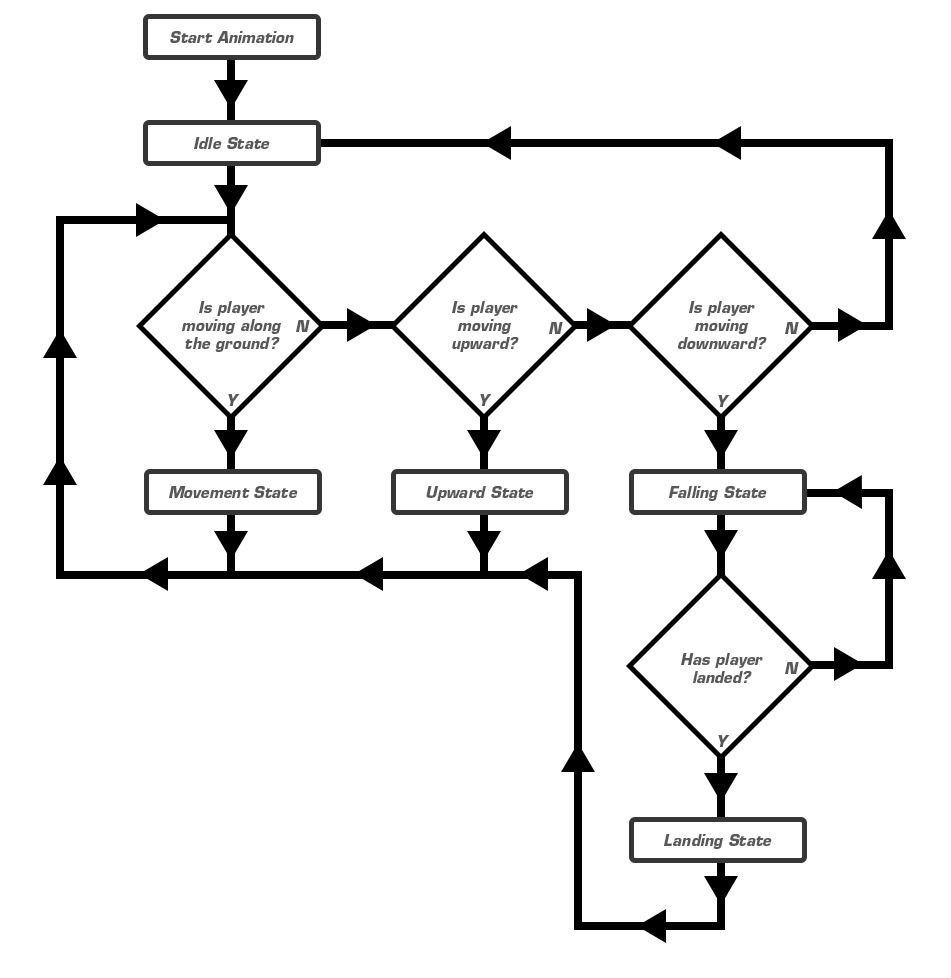
[[](http://api.unrealengine.com/CHN/Engine/Animation/StateMachines/EditingStateMachines/index.html)](http://api.unrealengine.com/CHN/Engine/Animation/StateMachines/EditingStateMachines/index.html)

**[编辑状态机](http://api.unrealengine.com/CHN/Engine/Animation/StateMachines/EditingStateMachines/index.html)**

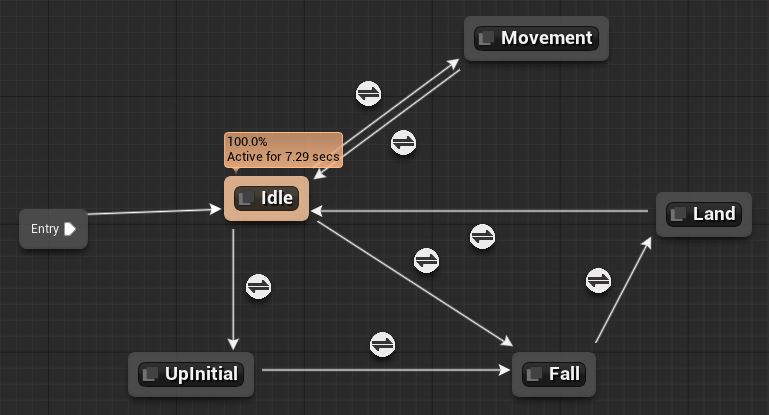
[关于编辑动画状态机的指南。](http://api.unrealengine.com/CHN/Engine/Animation/StateMachines/EditingStateMachines/index.html)

**状态机的工作原理**

动画状态机允许您以更加模块化的方式来处理动画蓝图。用户简单地定义角色或骨架网格物体可能处于各种状态。接下来，他们定义角色或网格物体进入及退出每个状态的情境。这和流程图类似。比如，您可能有一个定义了角色动画过程的流程图，它可能如下所示：



如果您想为上面的流程设置一个状态机，那么其最终效果将如下所示：



**状态机解析**

状态机主要由状态网络及定义进入或退出一个状态的变换规则构成。每个状态(及其附带的每组规则)是其自身压缩的蓝图网络。这使得处理复杂的动画变得非常简单，而不必使用非常复杂的动画图表。以下是状态机的关键组成部分：

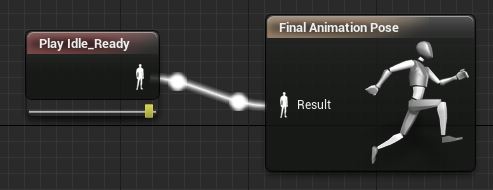
**状态**

从概念上讲，最好把状态想象成为一个组织良好的动画图表部分，我们知道角色或骨架网格物体将会定期地混入及混出该状态。然后，您可以变换到图表的这部分中或者从这部分变换出来。比如，您的角色动画可能有一个空闲状态，那么该状态可以仅包含一个单独的动画。

这里是在状态机中看到的这样一个空闲状态。



当双击它时，会显示空闲状态中的简单图表。



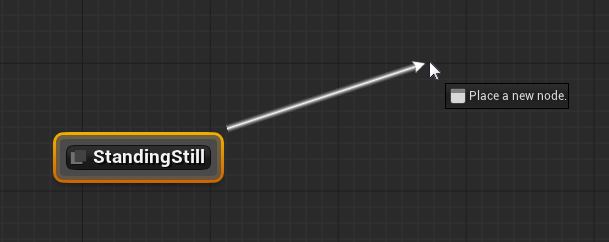
一旦角色开始移动，它们可以变换到Movement(移动)状态中 - 该状态可以定义角色在表面上横向移动的效果，比如走路或跑动。



在这个状态中，可以包含一个[混合空间](http://api.unrealengine.com/CHN/Engine/Animation/Blendspaces/index.html) ，该混合空间由角色的运动速率和方向定义。

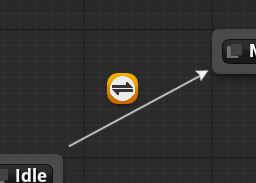


状态内部及其本身都没有引脚，因为我们想让它们具有更加灵活的布局。要想把一个状态连接到另一个状态上，仅需简单地从该状态的外边缘拖拽一条线即可。这个操作也可以用于创建新状态，就像在蓝图中创建一个新节点一样。

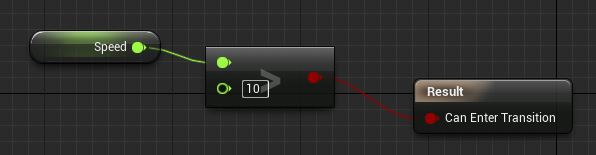


**变换规则**

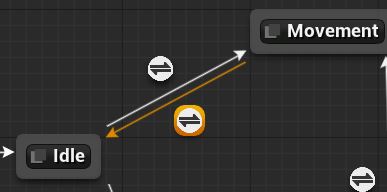
一旦您定义完了状态，您就需要控制您的网格物体如何从一个状态变换到另一个状态。这就是变换规则的作用。变换规则是沿着连接状态和通道的连线自动创建的。



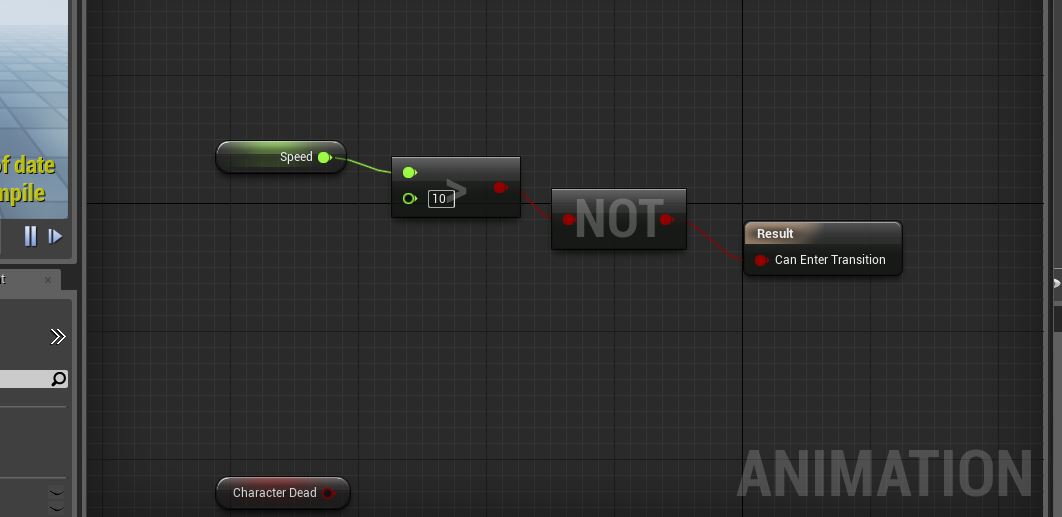
变换规则可以对多个变量的值进行判断及检测，最终的目的是输出一个 True/False 值。这个布尔值输出决定了该动画是否可以执行该变换。



在这个示例中，我们检测了角色的Speed(速率)的值。一旦该值大于10，我们将启用该变换。在这种情况下，角色可以从一个Idle (空闲状态)移动到一个Movement（运动）状态。然后，我们可以使用一个完全独立的规则来返回到空闲状态。比如，这里我们有一个从运动状态返回到空闲状态的规则。



在内部，这个规则简单地判断我们变换为运动状态所使用的同一个值是否还为true。



关于

变换规则及其属性的更多信息，请参照变换规则文档。

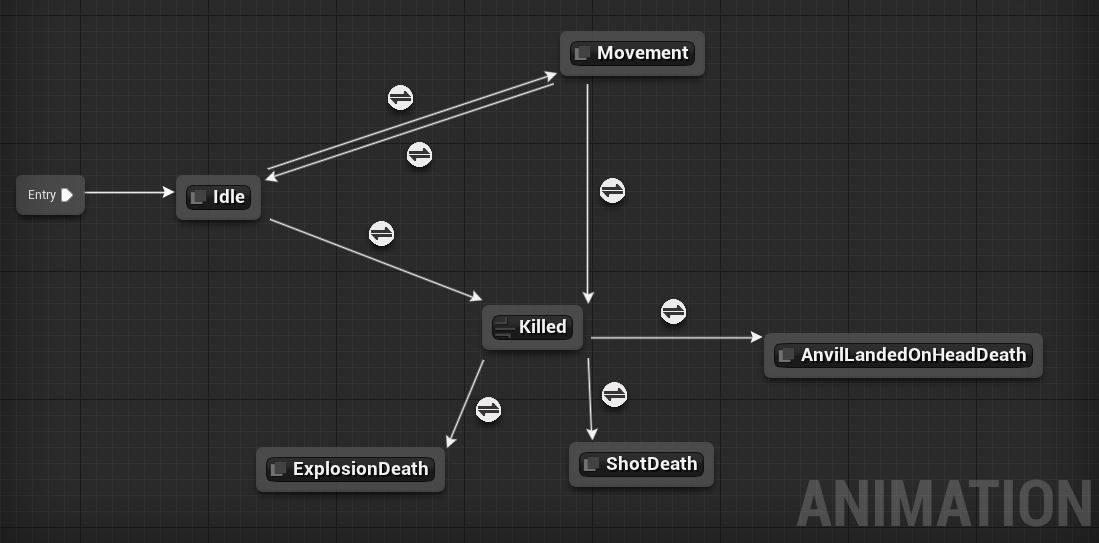
**通道**

Conduits(通道)是一种更加高级的、共享的变换资源。变换规则始终简单地是从一个状态变换到另一个状态的 1-对-1的变换关系，而通道可以实现 1-对-多、多-对1、多-对-多的变换关系。



比如，您可能会遇到这样的情况，当一个角色被杀死时，您有好多个可以播放的动画。首先，您可以使用简单的变换规则，它规定当角色的生命值小于等于零时，角色进入Death（死亡）状态。但是，如果要进入的状态不只一个，那么该变换就可以使用通道，您可以执行一系列的判断来确定玩家是如何死的(射中、刺伤、爆炸、撞车、铁砧打入头部等)，然后从此开始展开分支，对应进入不同的独立的死亡状态。

这样的图表如下所示（可能有一点简化）:



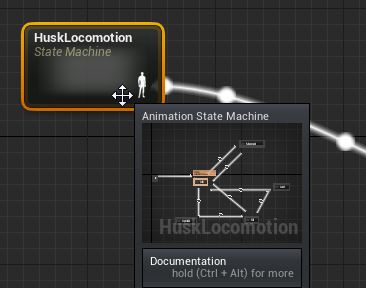
在内部，通道图表和变换规则看上去是一样的：



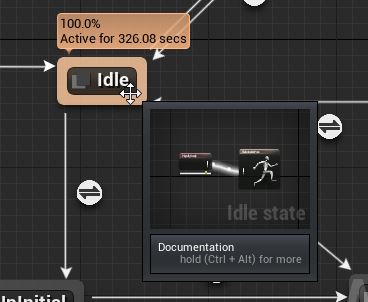
然后，通道中存在的真正的变换规则可以决定死亡的类型，并允许混合到适当的状态中。

**鼠标悬停预览**

状态机提供了鼠标悬停预览状态机本身及其组件节点的功能。比如，如果您把鼠标悬停到一个状态机上，那么预览窗口将会弹出，为您提供了状态机的高层次视图。这和合并的蓝图网络类似。



这些预览功能也可以扩展到状态机中的各种状态和变换规则。在其上面悬停悬停鼠标将会显示其内部构成的预览效果。



对于变换规则来说，您将不仅会看到这些节点的预览效果，同时也会看到规则的详细语言解释。

