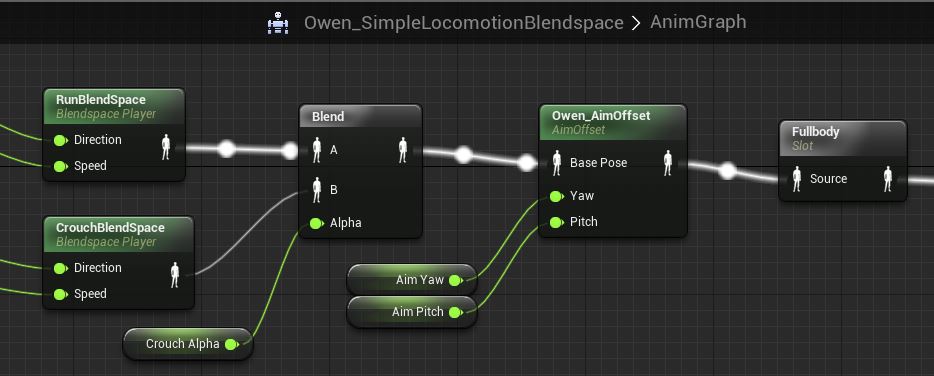
**动画图表**



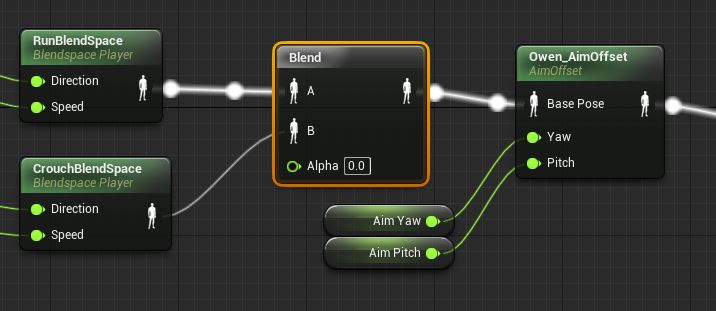
**动画图表** 用于计算当前帧上 骨架网格物体 的最终姿势。 默认情况下，每个 动画蓝图 有一个 **动画图表** ，在该动画图表内可以放置 [动画节点](http://api.unrealengine.com/CHN/Engine/Animation/AnimBlueprints/AnimGraph/index.html#%E5%8A%A8%E7%94%BB%E8%8A%82%E7%82%B9) ，以便对 动画序列 进行采样、执行动画混合或者通过使用 骨架控制器 来控制骨骼变换。 然后,在每帧中将最终的姿势应用到 骨架网格物体 上。



在 **动画图表** 中，您可以使用 **事件图表** 或其他地方（比如代码中）计算的值，且可以使用这些值作为 混合空间 或其它混合节点的输入。 当然，您也可以不进行任何特殊的动画混合，仅是简单地直接采样 动画序列 。 **动画图表** 还允许使用 骨架控制器 来直接地修改骨骼的位置、旋转度及缩放比例。

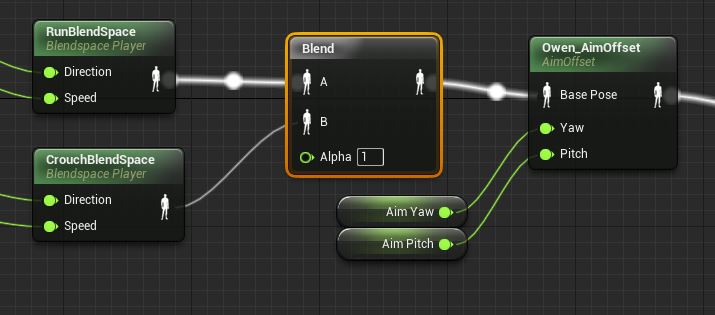
**执行流程**

所有图表都有一个执行流程，呈现为图表节点上的 exec 引脚 间的脉冲连接。 对于常规图表来说，比如一个 蓝图 的 **事件图表** ，这个流程只有在蓝图执行时可见，因为它依赖于事件的触发。 **动画图表** 却有所不同，它任何时候都会显示执行流程， 因为它不是基于事件的，且每帧都需要进行计算。



在 **动画图表** 中，执行流程代表 姿势 从一个节点传入到另一个节点。 某些节点（比如[混合](http://api.unrealengine.com/CHN/Engine/Animation/NodeReference/Blend/index.html) 节点）有多个输入，且可以在内部决定哪个输入是当前执行流程的一部分。 这个决定通常依赖于某个外部输入，比如传入到 数据引脚 的值。

在下面的示例中，这个同上面介绍的一样的混合节点，基于它的 Alpha 引脚（为了简化起见，手动设置该值）的值来在其输入之间切换执行流程。



当Alpha为1.0时进行混合 - 使用输入B

**应用动画图表**

**动画图表** 通过计算一个节点图表进行工作。 其中一些节点在一个或多个输入姿势上执行特定动作， 而其他节点则用于访问或采样其他类型的资源，比如 动画序列 、 混合空间 或 动画蒙太奇 。 以下介绍了这些资源和节点，以及它们是如何融入到 **动画图表** 中的。

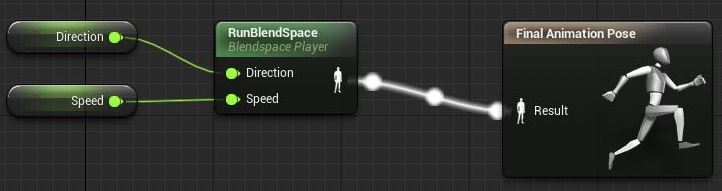
**动画序列**

AnimationSequence(动画序列) 的播放尽管可以通过代码来执行，但一般都通过应用到 骨架网格物体组件 的 动画蓝图 来处理。在 **动画图表** 中，可以对用于创建 动画蓝图 的骨架的所有 动画序列 进行采样。这些动画序列可以作为 序列播放器节点进行放置，其输出是对该 动画序列 采样后而生成的姿势。



**混合空间**

通过在 动画蓝图 的 动画图表 中放置混合空间节点，便可以应用混合空间。在 动画图表 中，您可以向 混合空间 输入数据然后输出混合后的姿势。

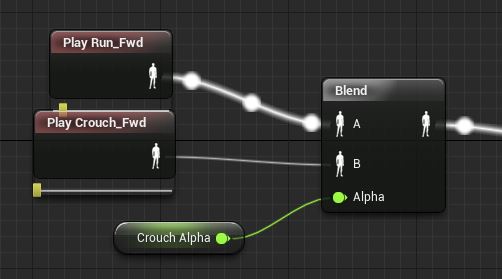


**动画节点**

AnimationBlueprints（动画蓝图） 在它们图表中使用各种节点来操作输入姿势 ，比如混合、直接操作骨骼等。引擎中提供了几种独特类型的动画节点， 这包括事件、混合节点、骨架控制器、空间和转换节点。

**动画混合**

混合节点用于将多个动画混合到一起。这些节点仅能放置在 VimBlueprint 的 **Anim Graph（动画图表）** 中。每个混合节点取入多个姿势、一个alpha值或一个用于计算最终输出姿势中每个姿势所占权重的权重值。某些混合节点可能比其他的节点更复杂，需要输入额外的数据。



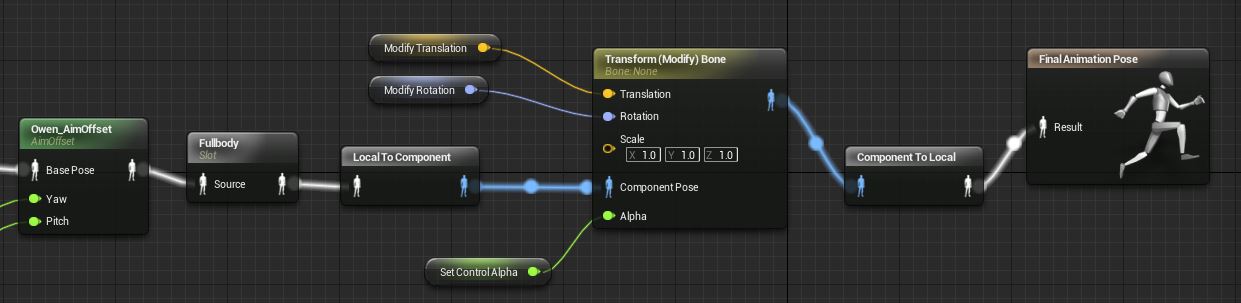
**骨架控制器**

**SkeletalControls(骨架控制器)** ，也称为 **kelControls** ，可以直接控制骨架中的骨骼。这些骨架控制器可以用于在 Animation Blueprints（动画蓝图） 中控制一个单独的骨骼、创建IK链等。这种直接控制底层 骨架 的方式可以创建程序化的、动态可变的动画。一个骨骼的 **变换** 可以用于驱动另一个骨骼；或者当播放一般的走路动画时，可以使用踪迹来使得角色的脚部贴合地面。您可以应用任何类型的修改来调整或完全覆盖 AnimationSequences（动画序列） 对骨骼的 变换操作\_ 。

**空间转换**

动画姿势可以在 本地空间 中也可以在 组件空间 中。动画蓝图 的 **动画图表** 中的 **Convert Spaces（转换空间）** 节点提供了在 本地空间 和 组件空间 间转换姿势的功能。本地空间假设 骨骼的变换是相对于那根骨骼进行的。组件空间假设该骨骼的变换是相对于 SkeletalMeshComponent（骨架网格物体组件） 的。

一般，当在 动画蓝图 中应用姿势时，这些姿势位于本地空间中。但是，某些混合节点和所有的 SkeletalControls（骨架控制） 都是在组件空间中运作的。这意味着，输入姿势在传入到这些类型的节点之前需要先进行转换 。如果输入姿势来自于一个输出到本地空间的节点 ，那么在 SkeletalControl 在其上面执行任何操作之前，必须将其转换到正确的空间中。 在 SkeletalControl 执行完操作之后，所得到的姿势必须转换回到 本地空间中，以便作为其他混合或 **Result** 节点的输入。



**同步组**

**同步组** 用于保持可能具有不同长度的相关动画同步。 比如，您可能有个循环走动动画和跑动动画，您想把它们混合到一起，以便角色可以平滑地加速或减速。 但是，如果这些动画的长度不一样，比如走动动画长度比跑动动画长很多的情况。 在这种情况中，直接将一个动画混合到另一个动画会产生不自然的结果，脚部动画切换时出现难看的“拍打”现象。

**Sync Groups** （同步组）通过使得一个主要的动画节点作为 Leader（主导者） ，所有其他相关动画将简单地缩放它们的时间长度来和主导者的时间长度相匹配。 一般，主导者是混合权重最大的节点。 随着权重混合，当跟随动画的权重超过主导动画的权重时，跟随动画 变为 主导动画。 通过这种方式，两个动画可以平滑地进行协作，产生从一个运动到另一个运动的无缝变换。

但是，应该注意的是，由于动画的时间是变化的，所以需要对某些动画进行一定的考虑。 比如，在混合两个不同长度的走动动画和跑动动画的例子中，您会想确保您的动画全都从同一只脚上开始及结束。 提前建立这些规则有助于使得所有动画进行更加平滑的混合！

