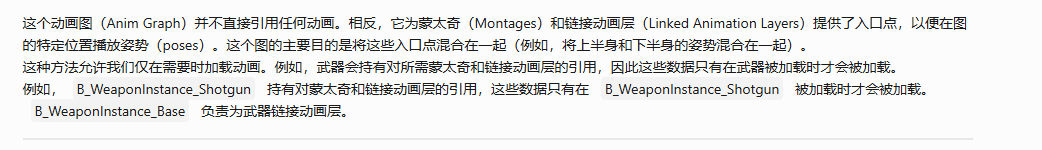




3

**动画蓝图教程 #4**  
这个状态机处理角色高层次状态之间的转换。  
每个状态的行为主要由 ABP\_ItemAnimLayersBase 中的层（layers）来处理。

**动画蓝图教程 #5**  
与 AnimBP\_Mannequin\_Base 一样，这个动画蓝图在 BlueprintThreadSafeUpdateAnimation 中执行其逻辑。  
此外，这个动画蓝图可以通过 ​**属性访问（Property Access）​** 和 GetMainAnimBPThreadSafe 函数访问 AnimBP\_Mannequin\_Base 中的数据。  
下面是一个示例。

**动画蓝图教程 #5**  
与 AnimBP\_Mannequin\_Base 一样，这个动画蓝图在 BlueprintThreadSafeUpdateAnimation 中执行其逻辑。  
此外，这个动画蓝图可以通过 ​**属性访问（Property Access）​** 和 GetMainAnimBPThreadSafe 函数访问 AnimBP\_Mannequin\_Base 中的数据。  
下面是一个示例。

**动画蓝图教程 #6**  
这个动画蓝图被设计用于处理常见武器类型（如步枪和手枪）的逻辑。  
如果需要自定义逻辑（例如弓这样的武器），可以编写一个实现了 ALI\_ItemAnimLayers 接口的不同动画蓝图。  
与其直接引用动画资源，这个动画蓝图通过一组变量来控制逻辑，这些变量可以被子动画蓝图覆盖。  
这些变量可以在 ​**​“Anim Set - X”​** 类别中找到（位于 ​**My Blueprint** 面板中）。  
这使我们能够为多种武器复用相同的逻辑，而无需在一个动画蓝图中引用（并因此加载）每种武器的动画内容。  
有关示例，请参见 ABP\_RifleAnimLayers，它是一个子动画蓝图，为每个 ​**​“Anim Set”​** 变量提供了具体值。

**动画蓝图教程 #7**  
这个动画蓝图为 AnimBP\_Mannequin\_Base 中的每个状态实现了一个层（Layer）。  
层可以播放单个动画，也可以包含复杂逻辑（如状态机）。

**动画蓝图教程 #8**  
这是一个 ​**动画节点函数（Anim Node Functions）​** 的示例用例。  
动画节点函数可以运行在动画节点上。它们仅在节点处于激活状态时运行，这使我们能够将逻辑限定在特定的节点或状态中。  
在这个例子中，一个动画节点函数在节点变为相关状态时选择要播放的动画。另一个动画节点函数管理动画的播放速率。

**动画蓝图教程 #9**  
这是一个使用 ​**距离匹配（Distance Matching）​** 的示例，用于确保起始动画（Start animation）所行进的距离与 Pawn 拥有者的行进距离相匹配。  
这样可以防止脚部滑动（foot sliding），通过保持动画和运动模型（motion model）的同步来实现。  
这实际上是对起始动画的播放速率（play rate）进行控制。我们对有效的播放速率进行限制（clamp），以防止动画播放得太慢或太快。  
如果有效播放速率被限制，我们仍然会看到一些滑动。为了解决这个问题，稍后会使用 ​**步幅校正（Stride Warping）​** 来调整姿势，以修正剩余的差异。  
要使用距离匹配功能，需要安装 ​**动画运动学库插件（Animation Locomotion Library plugin）​**。

**动画蓝图教程 #10**  
这是一个通过 ​**扭曲（Warping）​** 动画预设姿势（authored pose）来匹配 Pawn 拥有者实际行为的示例。  
**方向扭曲（Orientation Warping）​** 会旋转角色下半身的姿势，使其与 Pawn 拥有者的移动方向对齐。  
我们仅设计了前/后/左/右四个方向的动画，并依赖扭曲技术来填补其他方向的空白。  
随后，​**方向扭曲** 会重新调整角色的上半身姿势，使角色继续朝向摄像机视角的方向瞄准。  
**步幅扭曲（Stride Warping）​** 会在动画预设的步幅速度与 Pawn 拥有者的实际速度不匹配时，缩短或延长腿部的步幅。  
要使用这些节点，需要安装 ​**动画扭曲插件（Animation Warping plugin）​**。