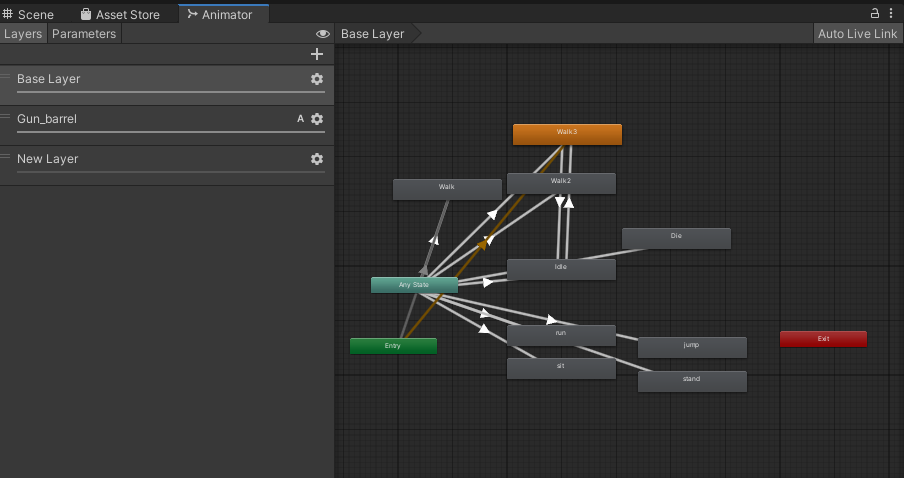
Unity四足机器人IK模拟

**简介：**

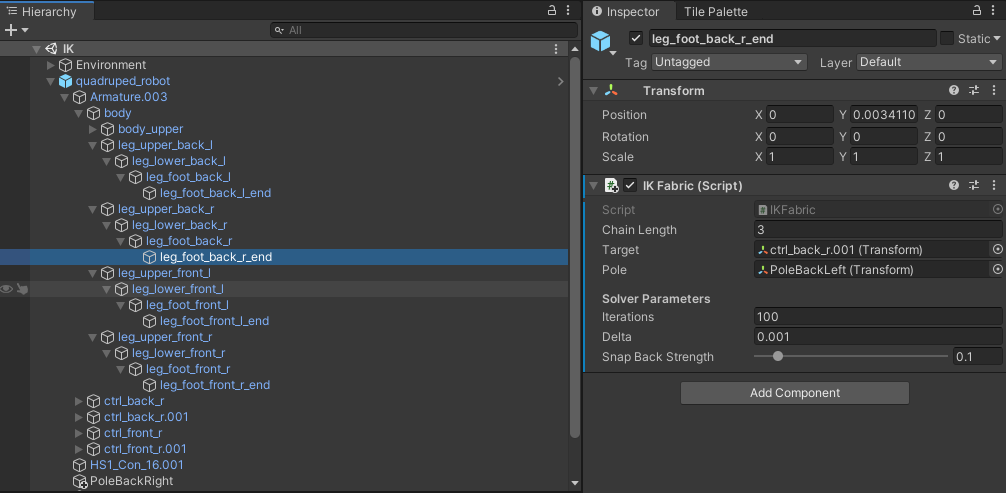
反向动力学（Inverse kinematics）是一种通过先确定子骨骼的位置，然后反求推导出其所在骨骼链上n级父骨骼位置，从而确定整条骨骼链的方法。它的特点是工作效率高，大大减少了需要手动控制的关节数目，且环境互动效果好。

**实现细节**：

首先，从Unity Asset Store购买了Quadruped Robot Asset。该Asset包含一个四足机器人模型和各种动作对应的骨骼动画。



在这之后，将模型导入到场景中，然后找到模型骨骼的四肢根部，将写好的代码绑定上去。调整如迭代率，允许误差，弹性强度等部分参数，并且设置好关节数量和约束条件。就可以看到反向动力学模拟的动画效果了。



代码进行了简单的IK模拟，具体实现如下：

I**nit函数：初始化骨骼结构**。

从代码被挂载的骨骼开始，按照设置好的关节数量逐步遍历父节点。找到每一级关节点的位置、关节的旋转角度、两个关节之间的距离、方向等，如图所示：



需要注意的是，此时Target设置为与骨骼链的末节点位置相同，因为需要与骨骼动画相匹配。

**ResolveIK函数：通过迭代的方式求解骨骼链位置**



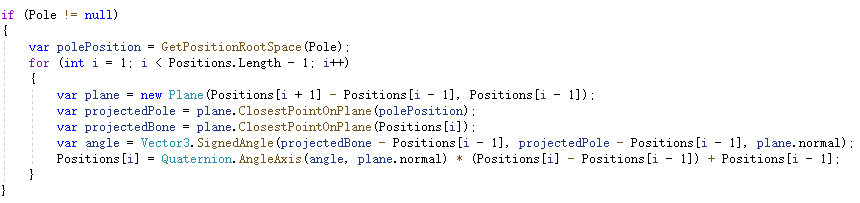
首先，需要判断目标的可达性，如果目标点到根节点的位置大于整个骨骼链的长度，那么就认为目标不可达，此时直接以根节点和目标点连线为方向拉直整个骨骼。

如果目标可达，则从目标点开始进行迭代。因为第一关节点和目标点的位置是固定的，我们只需要求解中间关节点的位置。所以先将最后关节点的位置设为目标点位置，然后从最后关节点开始根据原倒数第二关节点的位置计算方向，然后更新位置，即Position(倒数第二关节点)=Position(最后关节点)+方向\*两关节之间骨骼长度。就这样反向计算每个关节点的位置直到第二关节点。

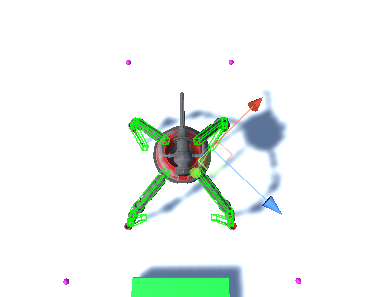
之后，再正向迭代，从第一关节点出发（固定位置），计算第一关节点和第二关节点之间的方向，然后Position(第二关节点)=Position(第一关节点)+方向\*两关节之间骨骼长度。正向计算至倒数第二关节点，即完成一个迭代过程。

重复迭代一定次数或者误差小于某个值之后，完成迭代。

上述迭代过程仅仅解决了位置关系，并未解决角度问题，因为空间是三维的，理论上存在无穷多解。为了防止骨骼向奇怪的方向弯曲，需要设定角度的约束。



Pole即为约束条件。对于中间的关节，以其前后关节连线为法向，在前一关节处构建平面（因为中间关节绕前后关节连线旋转不会影响其它关节的位置）。然后分别计算Pole点和中间关节在平面上的投影点，得到两个投影点到前一关节的向量之间夹角，然后旋转该关节相应的角度。这样就可以保证整个骨骼链不会变成奇怪的形状，符合我们的认知。



图中的四个紫色原点，分别对应四足的约束。

最后将计算得到的相对位置和旋转角度应用到骨骼链即可。

**注：**

1. Asset Store四足机器人链接

（<https://assetstore.unity.com/packages/3d/characters/robots/quadruped-robot-128292>）

1. 生成的可执行文件位于IK/Game目录下
2. 四足机器人模型和动画位于IK\Assets\QuadrupedRobotPack目录下
3. 代码位于IK\Assets\IK\Scripts目录下，其中IKFabric.cs实现IK模拟，RobotMove.cs控制机器人移动。