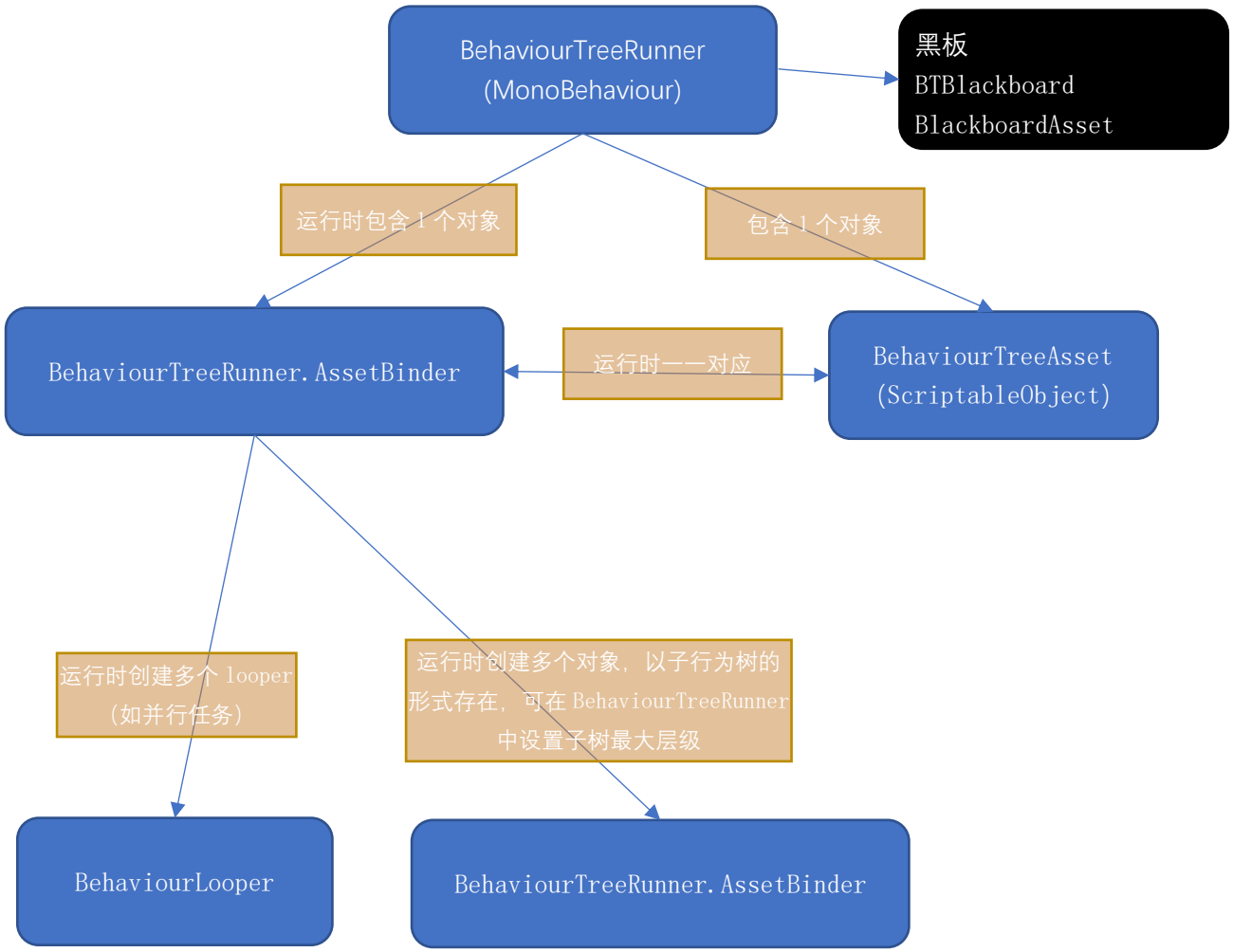
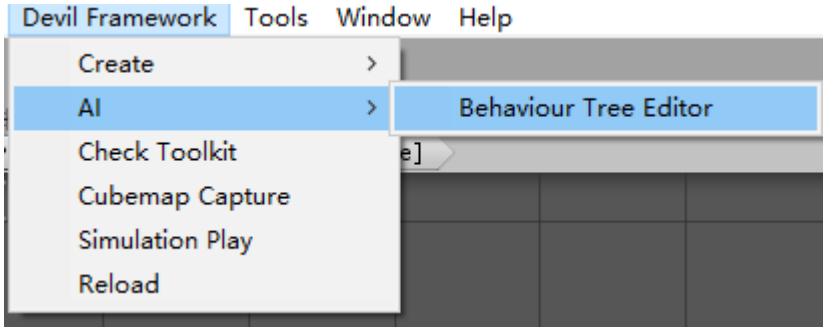
行为树编辑器使用说明

1. 组件说明

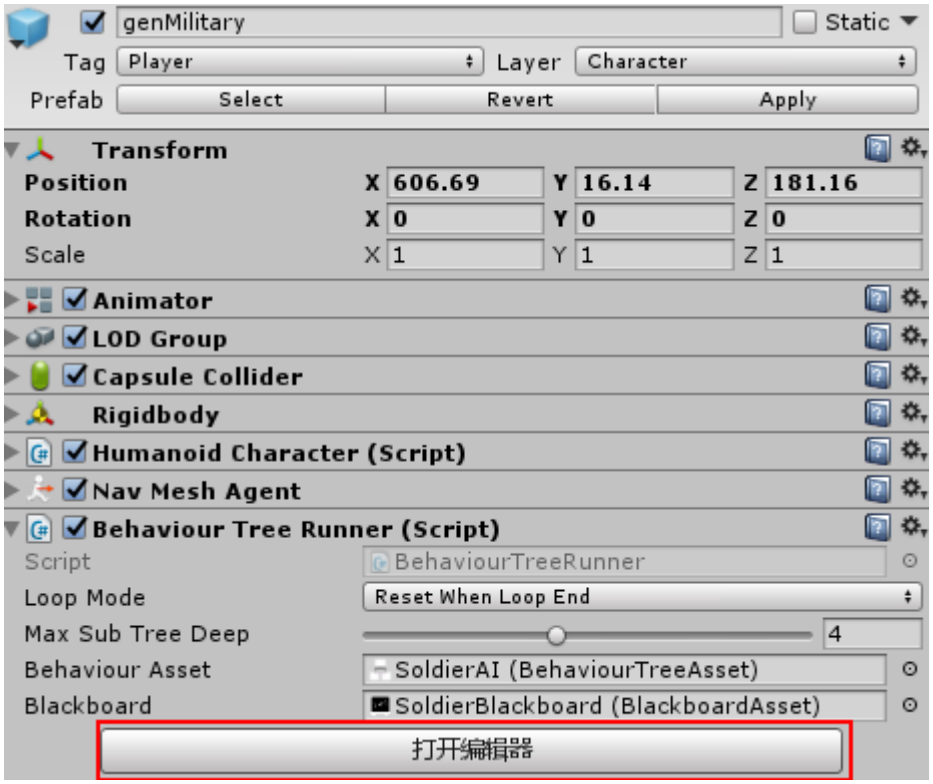


2. 打开行为树编辑器

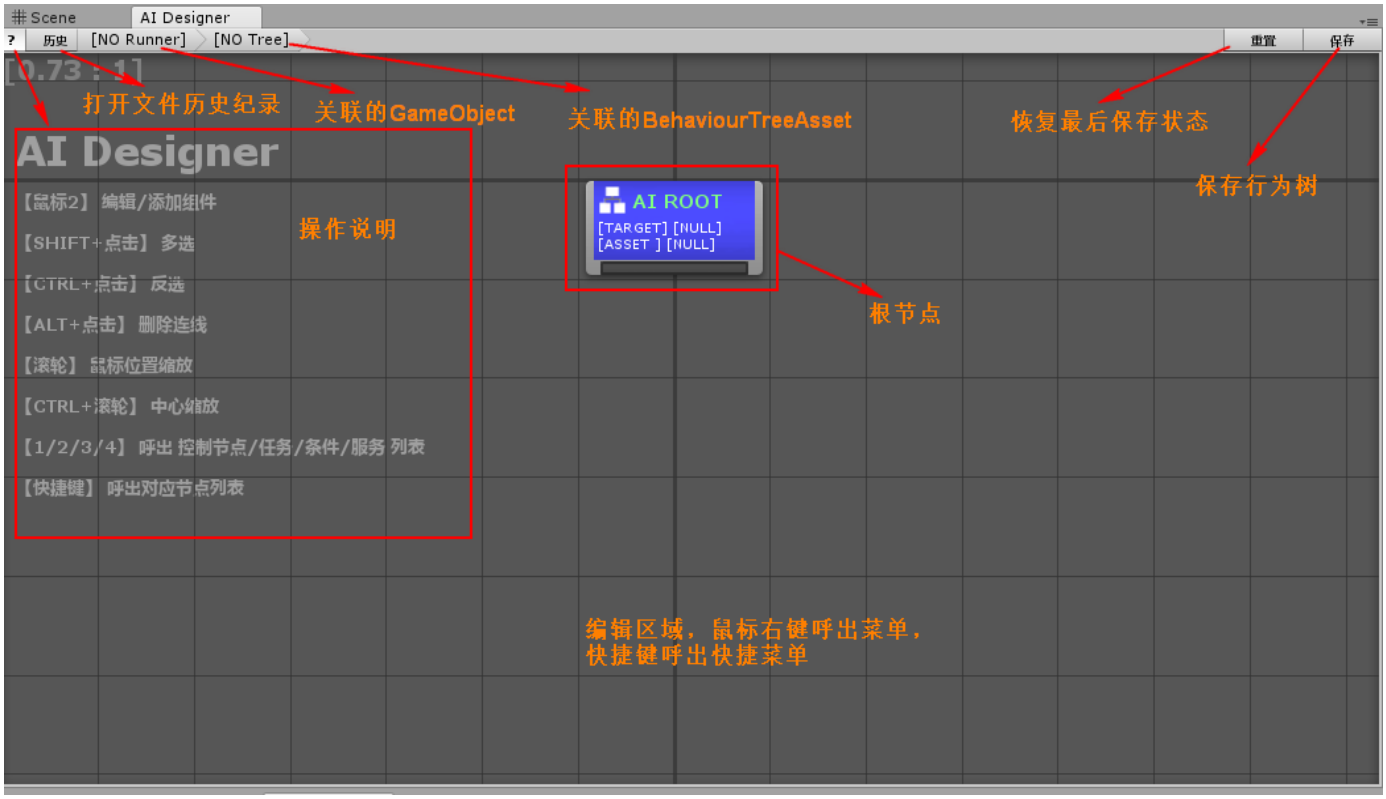
通过菜单“Devil Framework/AI/Behaviour Tree Editor”打开



或者通过“BehaviourTreeRunner”组件的 Inspector 窗口按钮“编辑”。

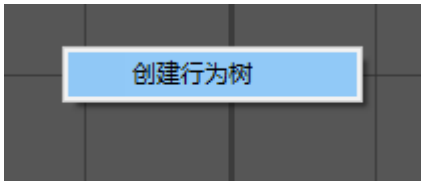


3. 编辑器说明

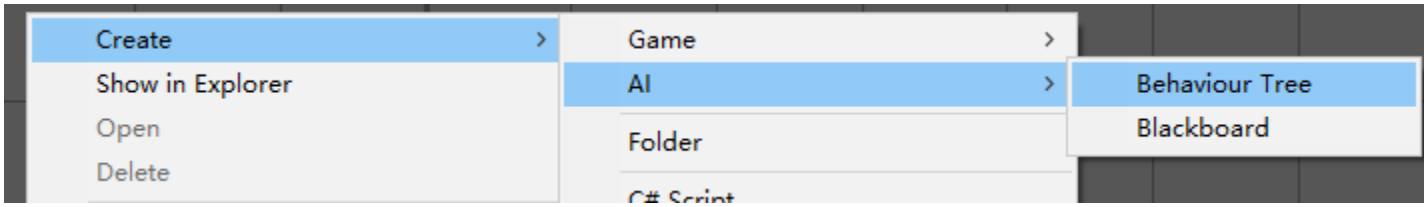


4. 创建/打开行为树资源

- 创建方法 1：编辑窗口空白区域右键菜单“创建行为树”，如下图：



- 创建方法 2：资源窗口创建资源菜单 “Create/AI/Behaviour Tree”，如下图：



- 打开方法 1：将关联的 GameObject 对象或者 BehaviourTreeAsset 对象拖拽至编辑窗口；
- 打开方法 2：在资源或 GameObject 对象的 Inspector 窗口点击打开编辑器按钮，如下图；

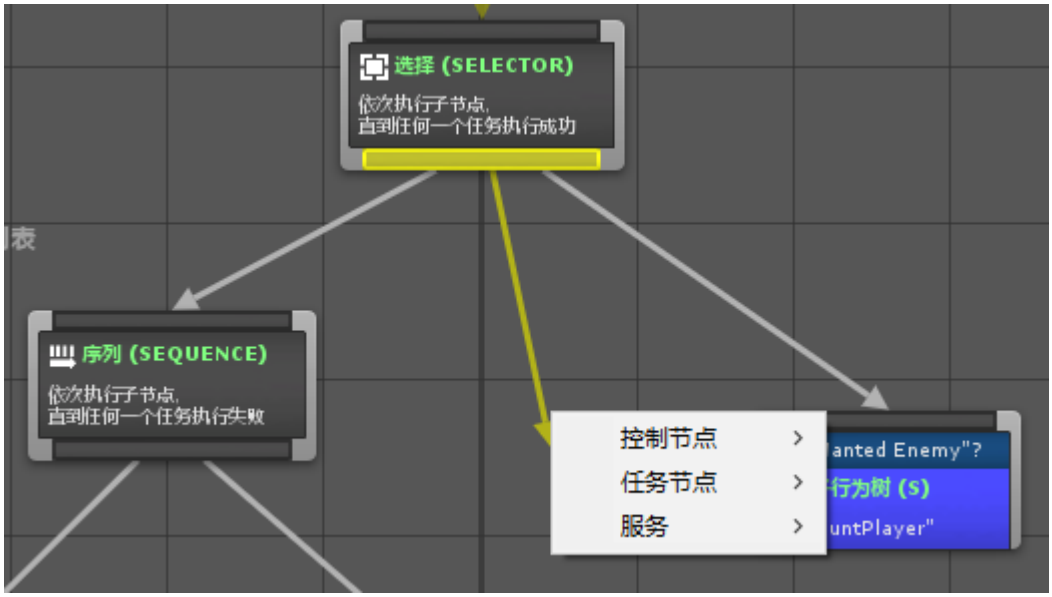
- 打开方法 3：通过编辑器窗口的历史菜单选择要打开的行为树资源，如下图。

5. 编辑行为树

- 创建节点：编辑区右键单击或者快捷键呼出节点菜单，选择需要的节点进行添加，如下图：

- 关联父子节点：点击节点上方或下方的手柄开始连线，停留在指定节点在此点击（右键取消）确认连接，在空

白区域点击创建新的节点，如下图：

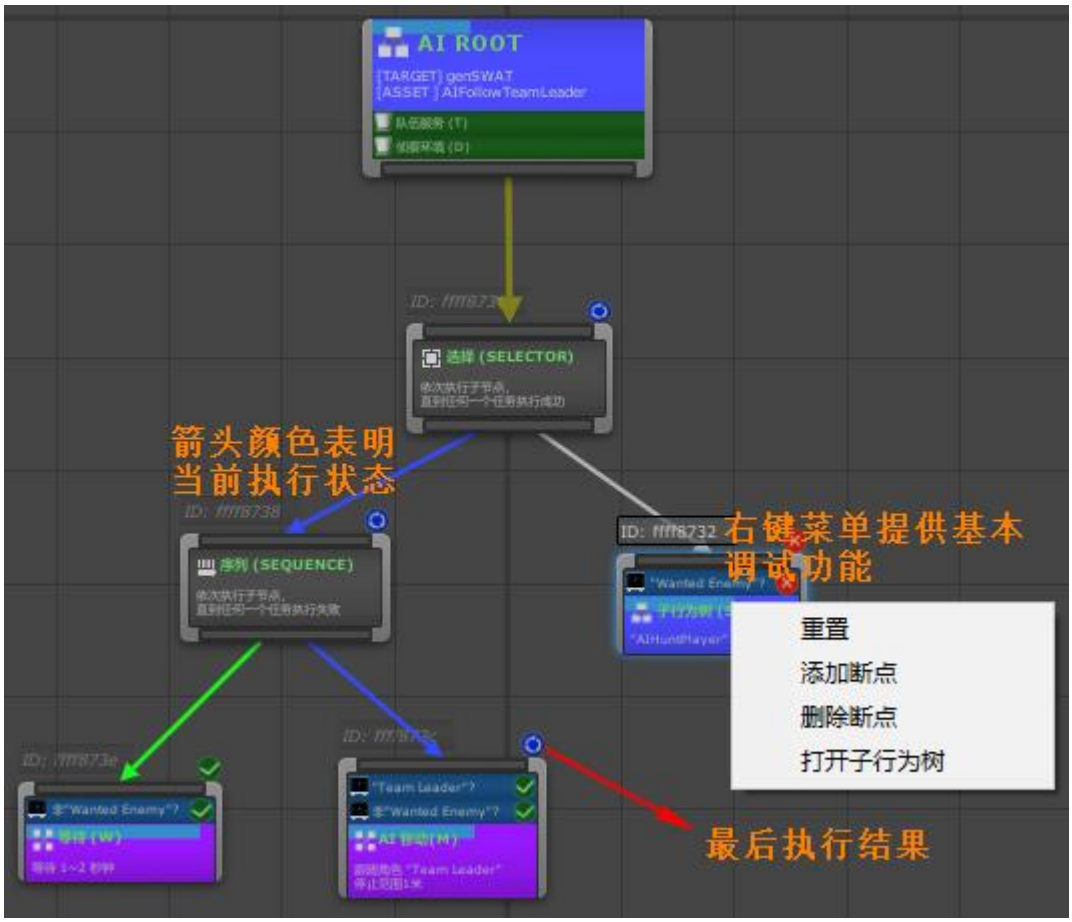


6. 节点说明

- 控制节点：如我们常说的行为树控制节点，实现了基本的逻辑功能。
- 任务节点：如我们常说的任务（行为）节点，实现了具体的行为/任务逻辑。
- 服务节点：被设计为一种装饰节点，它被附于该行为树上，在此行为树运行过程中，将一直处于活动状态，就像后台服务一样。
- 条件节点：被设计为一种装饰节点，可以附加到任何除了根节点以外的节点上，它只提供的条件的检查，用以控制对于任务的某些前提条件的设计（当条件不满足时会向任务或控制节点发送 Abort 信号并且退出执行该节点。

7. 调试运行

箭头颜色表明执行状态，
蓝色表示执行中，绿色表示成功，红色表示失败，灰色表示未执行；
节点上方蓝色横条表示更新状态，在节点执行 Update 的过程中该条会反应节点执行时间的推进，如下图所示：



8. 扩展控制节点

（包括后面的所有扩展，类名必须保持和文件名一致，需遵循 Unity 的 ScriptableObject 序列化规则）

定义类型继承 Devil.AI.BTControllerAsset，实现 IBTController 接口中定义的方法，如：

```
    [BTComposite(Title = "选择 (SELECTOR)",Detail = "依次执行子节点，\n直到任何一个任务执行成功",
        IconPath = "Assets/DevilFramework/Gizmos/AI Icons/selector.png", HotKey = KeyCode.S)]
    public class BTSelector : BTControllerAsset
    {
        int mIndex;

        public override IBTNode GetNextChildTask()
        {
            return mChildren[mIndex];
        }

        public override EBTState OnAbort()
        {
            return EBTState.failed;
        }
```

```
        public override EBTState OnReturn(EBTState state)
        {
            if (state == EBTState.success)
                return EBTState.success;
            mIndex++;
            if (mIndex < mChildren.Length)
                return EBTState.running;
            else
                return EBTState.failed;
        }

        public override EBTState OnStart()
        {
            mIndex = 0;
            return mIndex < mChildren.Length ? EBTState.running : EBTState.failed;
        }

    }
```

其中 BTComposite 可以定义节点的一些描述信息和快捷键等，这个可以完全省略，省略后会以默认显示方式显示节点。

9.扩展任务节点

继承 Devil.AI.BTTaskAsset， 实现 IBTTask 接口中定义的方法。

10. 扩展条件节点

继承 Devil.AI.BTConditionAsset，实现 IBTCondition 接口定义的方法。

11. 扩展服务节点

继承 Devil.AI.BTServiceAsset，实现 IBTService 接口中定义的方法。