1. 实体组件方案使用中出现的问题
2. 实体控制问题

实现某些逻辑需求时（例如动画控制系统，新手引导等），想要实现实体顺序启动必须添加新组件，且必须修改原系统，以适应新需求，此操作必然会将未知的bug引入已完成的系统。

1. 系统复用性问题

由于问题1的存在，新项目想要实现某些新需求，可能会需要修改原系统，假设原系统不可修改（比如封装入dll中），则需要重写（或继承）原系统导致重复的工作

1. 行为树

1.优点：基础逻辑的组合，可视化的逻辑编辑，容易拓展，无关节点的复用性极强

2.缺点：使用面向对象编程时，数据封装在子节点内，其他节点访问时会使节点间产生耦合，而数据放置在全局范围内会增加数据访问时的时间消耗

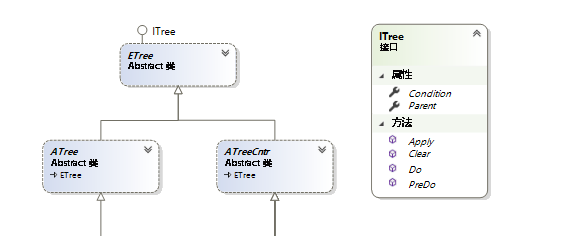
1. 与实体组件结合

能有效解决实体组件的可控性，拓展性问题，由于将节点数据转换为组件形式，消除了行为树节点间的耦合。极大提高到了逻辑部件的复用性

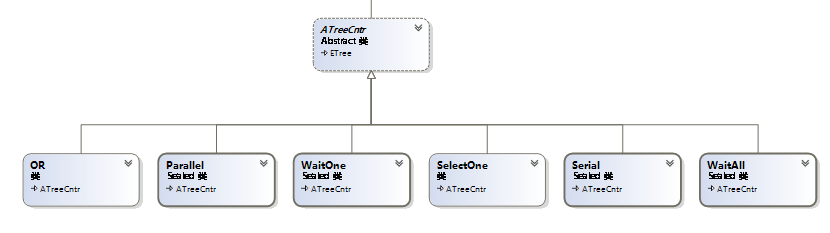
1. 设计与实现
2. 设计

树具备两部分逻辑功能，一部分为行为树部分，一部分为实体部分，且每个逻辑组合即可作为独立的逻辑，又可以作为其他逻辑的子逻辑

1. 实现
2. 实现行为树部分

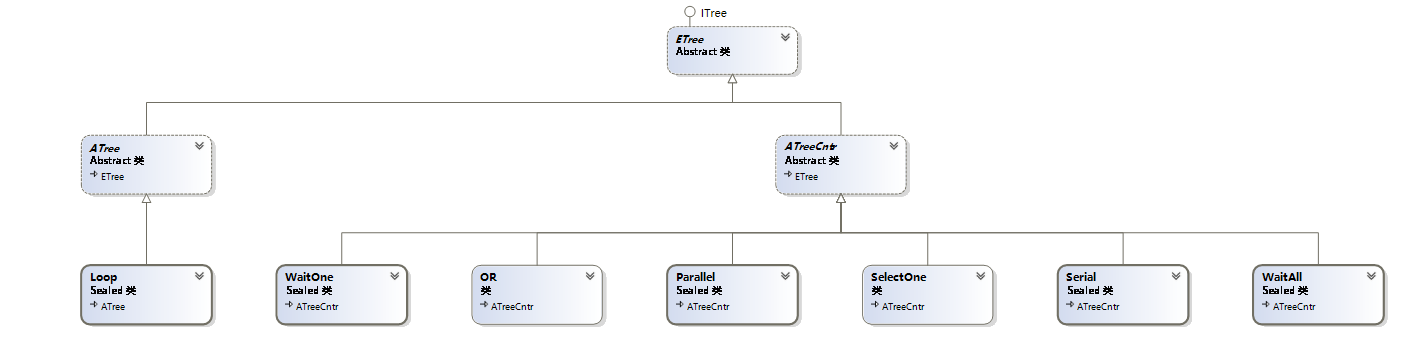


说明：ETree负责保存查找组件，Cntr父节点抽象， ATree叶子节点抽象



上图为不同的逻辑节点

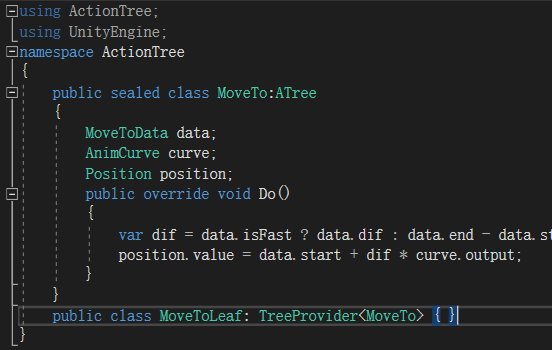
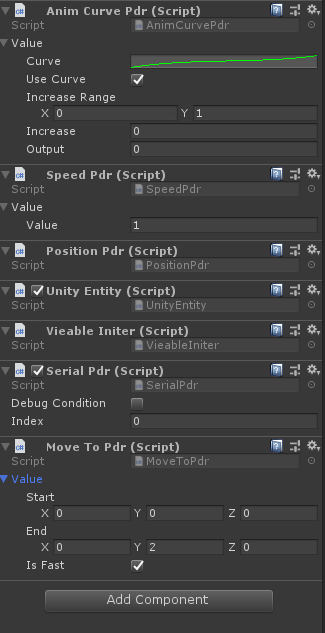
总览：



1. 组件实例的注入

首先在当前节点查找组件，若找到则注入当前实例，如未找到则由当前节点向上查找全部父节点，若存在则注入，若不存在则产生警告。

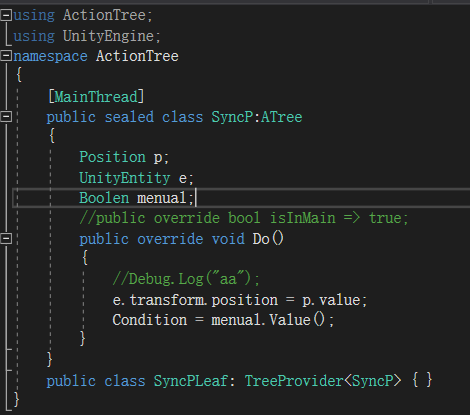
示例：



1. 添加多线程支持

节点默认由主线程以外的线程执行，当需要在主线程执行时需要添加[MainThread]标签

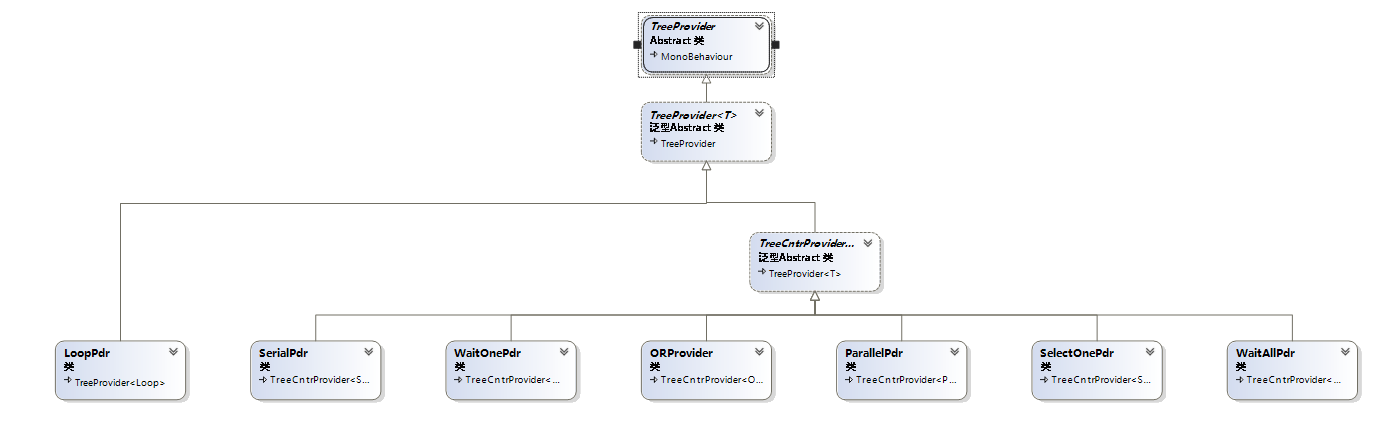
如：



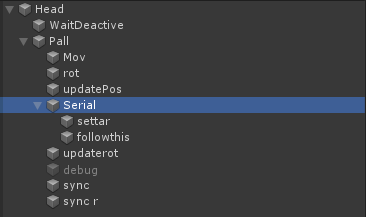
4.Unity 可视化

生成代理的mono使用GameObject进行组合

视图：



示例：



1. 总结---项目适用于中小型项目
2. 执行效率

由于树节点间为嵌套关系，因此整套结构的函数调用很多，单一线程60帧下，估计可容纳500-1000个实体同时执行，多线程下则将其乘以逻辑核心数，可得出最低实体数量约为2000-4000（以双核cpu为例）

1. 存在问题
2. 由于实体间并不只是单纯的并行，可能存在逻辑关系，因此实体间的执行顺序安排不一致可能会导致实体数值抖动的问题，目前解决方案为将其放置于同一线程下执行。
3. 场景切换需要使用专用节点来控制，直接使用unity方法会导致访问已销毁物体错误。
4. 目前编辑逻辑还比较麻烦，需要创建空物体再添加组件
5. 相同逻辑复制使用问题，目前思路为添加一个引用的Provider，暂未实现
6. 配置，资源，音频控制，网络逻辑等还未测试可能会出现其他问题