**关于目标驱动的状态机的设计方案**

**Virtual Bicycle**

**目前对于目标这样描述:**

一个物体有多个目标,放在一个目标队列里面

每个目标允许有多个子目标

**自行车的目标:**

MoveForward(在其中判断具体是处于 跟随/冲刺/普通移动 状态, 以及速度)

MoveTo(pos),根据MoveForward得到的路径,进行移动

CrossEdge,如果是到达了路径的交接部分或者路口处,调用此方法

Waiting,如果是决策发现前方被堵了或者是红绿灯等逻辑,可以停下来等待 或者"3,2,1.."的时候)

汽车的目标其实是差不多的.只是把一些竞技性的内容去掉

**自行车MoveForward的调用按照下面的步骤进行:**

1)根据MoveForward的最终要求(一般是终点),在该交通工具前方一个预判的距离内进行插值得到一系列的移动点(一般5个左右)

2)把第一个点加入目标队列

3)如果在进行MoveTo的时候发现,当前的目标点是危险的,就重新决策路径

4)假设预判了5个点作为路径,则到达了第三个点的时候就重新决策

5)如果在查找路径的时候发现到达了一条道路的终点了,则加入决策CrossEdge

**汽车的MoveForward调用和自行车很类似,除了加入遵守交通规则以外,这里就不多说了**

**下面需要解决的问题就是状态机的调用,状态机怎么和目标队列联系起来**,因为如果是按照我们的这种做法,目标决策结构也很类似于一个轻量的状态机,如果是两边同时决策,就很容易冲突了.

我的解决办法是状态机只是根据目标队列里面的第一个原子目标(没有子目标,无法细分的目标),把原子目标的状态(目标点,最大速度)拿来进行计算.

这样的话,状态机可以很简单,把大部分的任务都放在目标管理里面.

**接下来要解决的内容是状态机的目标队列的组织问题**

在Bicycle里面有一个GoalManager,管理目标,可以得到下个目标等等

在Bicycle里面有一个StateMachine,可以根据GoalManager的数据,来管理自行车的运行状态

GoalManager和StateMachine分别管理各自的状态,此处略过不提

Bicycle的目标驱动已经完成了部分了,下一步要做的就是完善,最近几天需要做的事情:

1. 添加公路网信息,使得Bicycle可以正常的寻路
2. 添加规划Bicycle路径的方法,使得Bicycle可以在移动中躲避其他车辆,并且有着最快的速度
3. 按照Bicycle的内容,依葫芦画瓢来做出汽车的移动(\*)
4. 按照Bicycle的规划,使得汽车可以在城市中穿行(\*)