从AI的难度上来看，应该是：MMO < FPS < RTS < 体育竞技。体育竞技游戏主要用到的是硬派AI。

硬派游戏AI，不是虚无缥缈的神经网络，用神经网络其实是一个黑洞，把问题一脚踢给计算机，认为我只要训练它，它就能解决一切问题的懒人想法。更不是遗传算法和模糊逻辑，你想想以前8位机，16位机上就能有比较激烈对抗的足球游戏、篮球游戏，那么差的处理器能做这些计算么？

硬派游戏AI，就是状态机和行为树。状态机是基本功，行为树可选（早年AI没行为树这东西，大家都是hard code的）。大部分人说到这里也就没了，各位读完还是无法写代码。因为没有把最核心的三个问题讲清楚，即：分层状态机、决策支持系统、以及团队角色分配。

**何为分层状态机？**

每个人物身上，有三层状态机：基础层状态机、行为层状态机、角色层状态机。每一层状态机解决一个层次的复杂度，并对上层提供接口，上层状态机通过设置下层状态机的目标实现更复杂的逻辑。这样各个层独立动作，避免耦合度过高。

* 基础状态机：直接控制角色动画和绘制、提供基础的动作实现，为上层提供支持。
* 行为状态机：实现分解动作，躲避跑、直线移动、原地站立、要球、传球、射球、追球、打人、跳。
* 角色状态机：实现更复杂的逻辑，比如防射球、篮板等都是由N次直线运动+跳跃或者打人完成。

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image8.png)

每一层状态机都是通过为下一层状态机设定目标来实现控制（目标设定后，下层状态机将自动工作，上层不用关心动画到底播到哪了，现在到底是跑是跳），从而为上层提供更加高级拟人化的行为，所有状态机固定频率更新（如每秒10次），用于判断状态变迁和检查底层目标完成情况。最高层的角色状态机的工作由团队AI来掌控，即角色分配的工作。而行为状态机以上的状态抉择，比如回防，到底是跑到哪一点，射球，到底在哪里起跳，路径是怎样的，则由决策支持系统提供支持。

**何为决策支持系统？**

状态机为角色的大脑，而决策支持系统为角色的眼睛和耳朵，常见的工具有势力图（Influence Map）和白板（相当于不同角色间喊话），其中势力图比较常用，篮球游戏AI势力图可以用下面几张图来表示：

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image9.png)

势力图1：防守篮板距离的map，每格分值为最远距离减去该格到篮板所在格子的距离

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image10.png)

势力图2：进攻篮板距离的map，每个分值为最远距离减去该格到篮板距离，篮板后为0

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image11.png)

势力图3：同敌人距离，每个敌人有影响范围，范围内，离敌人越近分越低，范围重叠选低的。

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image12.png)

势力图4：同所有队友目标位置距离map，打分方法类似上图。

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image13.png)

势力图5：与每个队友目标位置距离的map，标识单个队友目标位置距离的map。

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image14.png)

势力图6：现实传球可行性的map，分数越高，越容易把球传到该格子上。

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image15.png)

势力图7：容易把球传出的位置map，越容易直接传球给队友的区域分数越高。

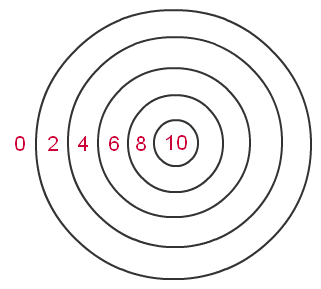
[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image16.png)

势力图8：综合map，把以上map按一定加权求和。球员有合法目标区域，便于实现内线游走和外线游走。

每个球员性格不同，权值也不同，有保守的球员，有喜欢冒险的球员，权值不同而已。这些势力图都是为了给上面的三层状态机和团队状态机提供决策支持的。

**何为势力图？**

所谓**势力图（Influence Map）**，也就是一种对于世界信息的预处理，计算和存储地图上的势力数据。比如说，我站在地图的某个点上，那我所站的这个位置，我就对它产生了最大的“势”，因为别人不可能再占据这个点了，由我这个点向外越远的地方，我对其的影响力就越小，也就是我所产生的势越小，用图画出来的话，就像一个靶子一样，所以可以称之为**势力靶图**，如果我们用分数来表示的话，中心就是10分（最大值），然后随着离中心点距离的增大，造成的势力值越来越小。



一般AI决策前，我们需要很多对于世界数据的收集和抽象，势力图作为其中的一种，可以看到，提供了对当前时刻，当前地图中物体的大致分布情况的抽象，可以很好用来帮助我们做AI寻路的决策，其实不光是寻路，其他AI决策问题，如果需要，也可以用势力图做参考数据，比如足球游戏中，如果我们要选择传球给一个处在空档的球员，我们就可以用势力图，来寻找哪一个球员处在空档的位置，等等。

对于势力图，还有几点需要补充一下：

1. 对于势力图的更新，大可不必每帧计算，因为对于一个场景来说，就算是有运动的物体，一帧的时间里，其变化还是很小的，所以，可以根据需要把更新的周期拉长，以优化势力图的计算
2. 方型的势力靶图一般可以满足需求，比起圆形的来说，势力图计算的性能上更有优势
3. 可以把势力靶图做成运动方向凸起的形状，因为如果是需要转向的物体，他背后的势力值应该是衰减的很快的，或者说延伸的很近的，而在其当前运动方向上的势力值，则延伸的比较远
4. 需要将势力图画出来（无论是在屏幕上，还是在另一个工具中，见我以前的关于AI调试的文章），以帮助调试。

## 关于如何收集游戏世界信息的想法

假设我们写一个篮球游戏中控球队员的AI，如果我们不考虑复杂的战术配合，一般来说，控球队员要么带球突破，要么直接投篮，要么传球给会造成威胁的空位球员，考虑到篮球场上瞬息万变情况来，如果单纯的if-else，会很难罗列出全部的条件，所以我们可能会采用模糊AI的决策逻辑，比如[分数系统](http://www.aisharing.com/archives/42" \t "http://www.aisharing.com/archives/_blank)（分数系统的一个基本思想，是为每个单独的行为打分，根据分数的高低来决定做哪个行为，而所有的因素就是打分的依据。）不过，由于今天我们讨论的是收集和存储游戏世界信息的问题，所以对于AI决策相关的东西，我们暂且不讨论。我们仅仅来看，在这个问题中，AI决策时候可能需要知道哪些游戏世界信息：

1. 场上己方球员的分布
2. 场上对方球员的分布
3. 球场上哪里比较有威胁（靠近篮筐的地方）
4. 球场上哪里比较不安全（比如有强力防守队员，或者防守队员人数很多）
5. 场地的构成（三分线位置，三秒区）
6. 球员的相关信息（比如球员能力，位置，当前行为等等）
7. ……

如果把上面的信息分个类别的话，可以分成以下4种

* 静态实体信息（比如5）
* 静态抽象信息（比如3）
* 动态实体信息（比如1，6）
* 动态抽象信息（比如4）

静态和动态的概念比较好理解，“静态”就是值不随着游戏的进行而变化的信息，“动态”就是随着游戏的进行会一直改变的信息，像场地信息就是静态的，不会改变的，像场上对方球员分布就是属于动态信息，因为他们的位置是一直变动的。而我这边提到的“实体”信息，指的是“真是存在”的信息，“抽象”是指“自定义”的参考信息。像场地信息，就是实体信息，因为类似三分线位置都是实际存在的信息，但像球场上哪里有威胁，那就是我们根据需要，自己定义的信息了，可以不断的调整和修正。

我借鉴了3D渲染中“帧缓冲区”（Frame Buffer）的概念，想用一种类似的方式来存储游戏世界信息，因为我们看到上面我们需要收集的信息中，不管是静态还是动态，实体还是抽象，很多都是和游戏地图相关的（除了6），所以我们就可以用一种“图”的方式来存储信息，称之为“游戏信息图”（Game World Info Map）。

首先我们按需求将游戏地图栅格化，比10×10，当然，粒度的大小取决于对于精度和效率的平衡。每一个格子就相当于“帧缓冲区”中的像素，然后我们可以创建多个这样的“图”，和创建多个“帧缓冲区”一样。每个图都代表上述信息中的一项内容，图中的每个格子都根据信息的内容填入0.0 ~1.0的值。

例如，我们要建立一个“场地威胁图”，我们定义0表示完全没有危险，1表示威胁值最高，那我们就可以这张图的相应的格子中填入相应的值，而且因为这是静态信息，所以只需要在游戏开始时填入就可以了，当我们填完每一个格子的时候，我们就得到了这样一张“场地威胁图”。对于动态信息的情况，我们需要在每一帧（或者每几帧）对“图”中的信息做一次更新，比如“场地危险图”，就是这样的动态“信息图”，需要根据防守队员的情况来实时更新。这样当我们填完所有的“图”信息后，AI决策时就可以知道任意时刻，在地图上的任意点上的相关信息了。

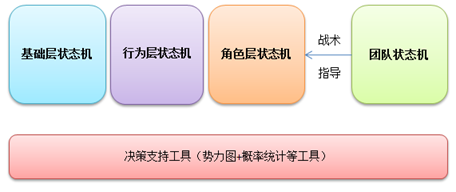
小地图的情况（如上例）可以直接做栅格化，但对于地图比较大的情况，如果直接栅格化的话，更新起来性能太低，这种情况可以考虑采用层次化的图模型，先将地图分成大块的格子，在大的格子里再细分成小格子，当查看距离近的信息的，采用精细的格子信息，查看远处的时候，采用粗略的信息，这样就可以在效率上取得一些平衡。

用“图”来表示世界信息的另一个好处是，可以方便的将信息绘制出来（在地图上，或者在外部的调试工具中），而不用面对一大堆的数据，如果再将不同的值配以不同的颜色来显示的话，那将大大的降低AI调试的难度。

可以看到，其实“图”的概念，就是对于**游戏世界信息中和地图有关的信息的抽象**，像我前段时间提到的[“势力图”（Influence Map）](http://www.aisharing.com/archives/80" \t "http://www.aisharing.com/archives/_blank)，就是“信息图”的一种应用。“信息图”的想法并不是我的独创，其实可能大家或多或少以前在编写游戏AI的时候也用到过，但我觉得整理一下的话，可以作为一种比较通用的结构来提炼出来，在AI中加以运用。希望对大家有所帮助。

**何为团队角色分配？**

每一层状态机为下一层设定一个目标，让下层自动工作，顶层角色层的目标则由最高层的团队ai进行战术指导

[](http://www.skywind.me/blog/wp-content/uploads/2015/04/image17.png)

团队状态机跟据当前的游戏情况确定当前首要目标（进攻或者防守），又根据当前的势力图等信息，确定进攻或者防守的具体战略（比如中路突破、盘路包抄、下底传中等），最终为当前己方的所有角色分配一个新的任务，即设定角色层状态机的新目标，确定他是做主攻还是做助攻，还是联防还是策应。具体该怎么联防，怎么策应，那就是角色层状态机的事情了。

**话题总结**

其实团队AI没那么玄乎，任何问题就是一个编程的建模问题，而最复杂的体育竞技类游戏的AI策略，上文已经给出模型，相信各位略加修改即可使用。写状态机是游戏AI的硬功夫，如果状态机逻辑经常改变或者项目规模大了以后可以考虑引入决策树来控制状态机，程序提供一系列接口，然后用可视化的编辑器进行更改，感兴趣的人可以参考决策树相关文章。

**概率统计**

如果上面这些逻辑都实现了，这时候才可以辅助与概率统计来让角色具备学习特性，比如统计某个策略对对手的成败情况，用来支撑下一次决策，这样能够逐步发现对手的弱点，还可以统计所有用户的大数据，来确定某种情况下，选择什么策略，能够对付60%的用户。

**神经网络**

在上面所有逻辑都实现了，你调试好了，玩着比较顺畅的时候，再在团队角色分配处尝试使用神经网络或者模糊逻辑，同样是学习大数据，来引入一些不可控的人性化的成分，让游戏更加有意思。（EA的 FIFA 20XX号称引入神经网络，Call of Duty的AI也号称引入了神经网络和学习机制）。确实让游戏更有趣一点，但仅仅如此而已。

**使用行为树的优点**

1. 行为树可以扩展和处理更复杂的行为逻辑，比如随机，序列，并行等等。某些请求可能不是用单个行为可以完成的，需要多个行为的配合，比如完成一个技能释放，需要先集气，然后再释放，类似这样的行为，就可以用行为树来实现了，更棒的是，集气这个行为还能被共用。
2. 可重用性
3. 可视化编辑

**行为树过于庞杂了怎么办？**

## CryEngine里的行为选择树（Behavior Selection Tree）

从CryEngine 3.3开始，行为树被舍弃了，取而代之的是称为“行为选择树”的模块。

行为选择树（Behavior Selection Tree），顾名思义，它的重点在于为“行为”和“选择”

* 行为：AI在每一个时刻，都必须处在某一个行为中
* 选择：AI在每一个时候，都会根据外部的条件，选择某一个行为作为当前的行为状态

这两点并不是很难理解，换句话说，就是每一个AI智能体都永远处在它所应该在的行为状态中（当然，这里的“应该”指人为预设的行为选择逻辑）

为了更好的说明上面复用的问题，我们可以举这样一个例子，比如有两种动物，一个是猫，一个是狗，同样是闲逛，猫有猫的方式CatWalk（猫步？），狗有狗的方式DogWalk，同样是吃饭，猫是CatEating，狗是DogEating，当饿的时候，会从闲逛行为切换到吃饭行为，吃饱了，就会再次切换回闲逛，所以对这两种动物来说，它们的行为选择树结构和逻辑上是一样的，不一样的只是他们的具体行为的执行方式，所以他们的行为选择树是可以复用的。

基本上，以上这些就是行为选择树的全部概念了，和行为树比起来，我觉得有两个比较明显的不同：

1. 对于“前提”（Precondition）的处理：在行为树的定义中，“前提”是可以带逻辑的，或是简单的逻辑代码，也或是复杂的逻辑代码，而在行为选择树中，它的前提被定义成了一个和“信号”所绑定的布尔变量，也就是说在它的前提中，不包含对于逻辑的计算，所有的这些计算都是在外部的其他AI模块（比如，感知系统，团队系统等等）中。所以行为选择树的“前提”可以看成是对一般的行为树前提的一个严格限定。
2. 叶节点映射：行为树的定义中，并没有别名系统，即使行为树的结构是一样的，但不同的智能体上也会绑定各自的行为树。行为选择树很好的引入了别名系统，这样就可以使行为树得到了复用，可以说，这一点是对行为树的一个改进。

另外，在实际开发中，会发现对于树（或者树的某个分支）的复用是十分必要的，就像我们不喜欢代码的冗余一样，对于行为树的冗余，有时也是不能接受的，所以不管从树的设计上来说，还是编辑器来说都要提供复用的架构设计，行为选择树就很好的做到了这一点，特别要指出的是，简单的前提设计在复用上有提供了一个很好的帮助，因为它仅仅是一个变量，所以在前提部分我们就不需要特别关注复用的问题了。如果是带复杂逻辑的前提，那势必我们还需要关心前提的复用问题。简单就是美啊!

总结下，行为选择树可以看成是对行为树的部分功能简化，和拓展，为编辑器的设计，为调试带来了很大的帮助，当然，由于它把前提的运算移到了外部，所以就更多的需要外围其他AI系统的支援了。

树只负责逻辑的走向控制， 这样树维护起来很方便，判断逻辑放在外部是为了节省计算资源。同时方便树的共享。  
只用selector树是因为可以用多个单一功能的树进行组合，比如移动树，动画树，旋转树，等等。因为所有单一功能的具体Action节点必然只有一个。也方便action节点的提炼共享。  
这样设计维护都很清晰，而且系统效率也会得到较好控制。

**http://www.aisharing.com/**