文章编号：1005-1228(2018)04-0008-04

MOBA 游戏人工智能的设计与实现

李 焜，李 平，李立波

(1. 长沙理工大学 计算机与通信工程学院，湖南 长沙 410076；
2. 陆军武汉军代局驻长沙地区军代室 湖南 长沙 410014)

摘 要：近年来，手机游戏市场不断的发展壮大。游戏人工智能(AI)对游戏的可玩性有着巨大影响，是游戏能否占有市场的重要元素之一。针对多人在线战术竞技游戏(MOBA)中多虚拟角色的组合决策分析研究，使用分层决策架构设计、消息队列实现游戏单位对环境变化应对及游戏智能间协作，同时适应游戏敏捷开发。通过在游戏中的实际运用证明该方法的可行性和有效性。

关键词：游戏人工智能；层次状态机；系统设计

中图分类号：TP18 文献标识码：A

DOI:10.19414/j.cnki.1005-1228.2018.04.003

Design and Realization of Artificial Intelligence of MOBA Game.

LI Kun, LI Ping, LI Li-bo

(1.College of Computer and Communication Engineering, Changsha University of Science & Technology, Changsha 410076, China;
2.Changsha military representative agency, Wuhan military representative bureau, PLA land force, Changsha 410014, China)

Abstract: In recent years, the mobile game market has been growing. Game artificial intelligence (AI) has a great influence on the playability of the game, and is one of the key elements that make the game marketable. The multiplayer online battle arena (MOBA) combined multi-virtual role decision-making analysis, using hierarchical decision-making structure design, message queuing to achieve the game unit to respond to environmental changes and game intelligence collaboration, and can adapt to the game Agile development. The feasibility and effectiveness of this method are proved by practical application in the game.

Key words: Game Artificial Intelligence; Hierarchical Finite State Machines; System Design

MOBA 游戏中，玩家通常只控制一个角色(英雄)，与其他玩家协作与另一个团队进行竞技，摧毁对方的核心建筑以获胜。此类游戏为提升可玩性，通常会对英雄的种类和特性进行扩展，以提高团队战术组合的策略深度。但对于在刚接触游戏的新手玩家，大量未知的英雄特性会造成其竞技水平偏低，在玩家群体中不易选择合适的对手，难以获得良好的游戏体验。针对这一问题，可以提供 NPC(Non-Player Character，非玩家角色)来作为玩家水平匹配的对手来协助其熟悉游戏。

NPC 由游戏人工智能(AI)控制游戏内行为，其逻辑决策结果直接影响其对战能力和操作表现，因此其智能性非常重要。简单的行为决策系统通常基于有限状态机(Finite State Machine, FSM)或行为树(Behavior Tree)，但其均有短板：FSM 在决策环境复杂时维护难度会急剧上升。行为树每次均需从根部开始决策，当决策节点数量过多后存在执行效率瓶颈，对商业游戏造型运营上升。也有通过强化学习来训练 AI 的尝试，但其需要一定经验数据，训练成本较高，并会由于游戏本身更新数值而导致训练结果失效，无法满足游戏敏捷开发所要求的确定性结果[1,2]。

针对以上问题，结合状态机和行为树的优点，设计并实现了一套多层次决策结构，各个决策体逻辑独立，使用组件式方便进行扩展开发和替换迭代。在此结构下硬编码实现 AI 逻辑，较好保证 NPC 做出类似玩家的行为反馈，保证游戏的趣味性。测试结果显示，决策结构能正确合理的驱动多个 NPC，表现出 AI 之间的合作。

收稿日期：2018-03-09

作者简介：李焜(1993-)，男，硕士研究生，主要研究方向：游戏人工智能；(通讯作者)李平(1972-)，男，教授，博士，主要研究方向：物联网与传感网，数据挖掘与大数据智能处理；李立波(1989-)，男，湖南益阳人，本科，主要研究方向：计算机软件。

1　结构设计

AI 模块设计的核心思想是将复杂的逻辑从结构设计上进行拆解，各个模块高内聚低耦合，并可以单独测试调试，以适应敏捷开发过程。从模块设计角度，只需关注自身的输入和输出，简化了功能设计时考虑的维度，降低制作难度。如图 1 所示，模块整体由环境信息管理器和多层决策器构成，其交互通过消息队列进行。

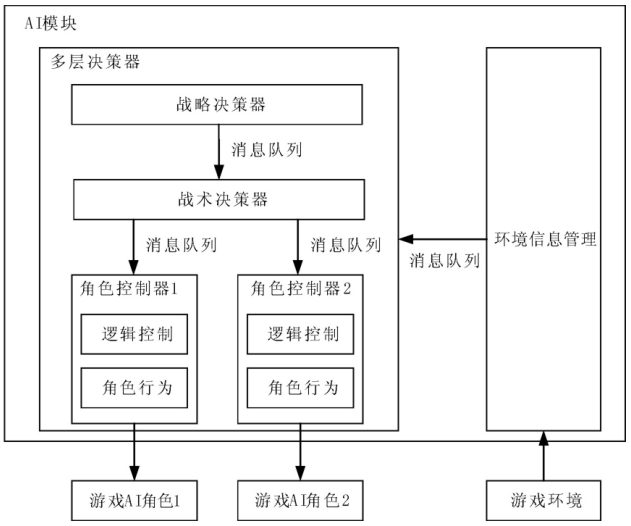


图 1　AI 决策结构图

1.1　环境信息管理

MOBA 游戏中通常包含士兵、防御塔等元素，其信息分别由各自的实体对象管理。AI 进行决策时需要获取这些对象的位置、属性信息、当前逻辑状态等即时信息。若让每个 AI 决策器都直接访问场景中几十个 NPC 对象的信息，势必造成模块间耦合[3]，并存在重复处理造成计算资源浪费。

因此，设计了环境信息管理模块。该模块具备以下功能：

⑴ NPC 信息变化时更新自身缓存数据。

⑵ 对收集信息进行预处理，如：通过经验公式，将生命值、攻防属性组合量化为战斗力；对士兵位置集中化计算交战点；敌方英雄使用指向性技能等。

⑶ 通过消息队列推送给关注的 AI。

1.2　多层决策器

传统有限状态机中，其每个状态之间是可以独立跳转的，但每个状态内部能做的行为是单一的。若希望状态机能执行复杂的逻辑，通常有两个方法：增加状态或状态内添加逻辑判断。但其均存在一定问题：状态机中每两个状态之前是存在转换关系的，在符合条件时会从原状态中跳出进入新状态，故当状态超过一定数量后，新状态加入所带来的维护成本是呈指数上升的。状态机作用即将复杂的 AI 逻辑抽象为 N 个状态来便于理解减少维护成本，若状态内部的逻辑复杂后，其逻辑维护成本又将上升[4]。分层有限状态机的思路，正是将单个状态内部逻辑也使用状态机来进行组织管理，将同类型的状态做成一个子状态机，由大状态机来维护子状态机。但本质上没有减少状态数量。

行为树的实现方式，是将各个逻辑用节点表示，分支逻辑在父节点逻辑下，仅当前置条件满足时才执行。但同样存在问题：每次决策均需从根节点执行，当节点数量达到一定量级后存在效率执行瓶颈，以及实际游戏中不同类型决策的需求频率并不一致[5]。

本文中多层决策器的实现方式，即将不同层级的决策分离到不同的状态机中：战略级、战术级和角色级。每层决策的频率和应用实体均不同，顶层决策器是针对全体 AI 来决策，真正决定角色行为的是底层决策器。决策结果会从上而下推送到下一层决策器，影响下一层状态机的决策逻辑。将决策逻辑分离后，不同层次决策间不存在耦合关系，可以方便调整不同情形下某一层的决策结果。避免单次决策需要考虑的条件较多，执行效率下降。

1.3　消息队列

每个决策器都需要获取外部信息来作为决策输入，并产生决策结果推送给下一层决策器。消息队列是在环境信息和多层决策器间提供信息共享和触发的模块[6]。其设计目的为：解除决策器和环境管理器间直接调用造成的耦合和决策深度超过预期。

单个决策器会接受到的消息类型是有限的。为保证决策器始终响应最为重要的消息，对消息实体引入优先级的概念。当决策器从消息队列中获取新消息时，会按优先级对当前消息组进行排序，高优先级的消息会先执行，并根据其执行结果决定是否继续响应后续消息。这一逻辑类似行为树中 Sequence Node 迭代所有子节点，当子节点返回 False 时停止迭代。与行为树不同的是，行为树要求节点具有静态性，而消息是动态产生并加入消息队列的。这一做法有效避免了决策器盲目响应外部消息，先产生的决策在极短时间内被后续的决策所覆盖而导致的状态抖动，角色行为表现不连贯。

未被执行的消息会保持在消息队列中等待。当一条低优先级的消息在被执行时，与消息产生的环境已经有了非常大的偏差——消息是具有时效性的。因此，对消息实体引入有效时间属性。从加入消息队列开始，在有效时间内还未被获取的消息实体将被丢弃。

通过调整消息优先级和有效时间，可以方便的优

化决策器在对各种外部环境时的反馈，令 AI 表现上符合玩家预期。

2 详细设计

2.1 战略决策器

战略决策器需对敌我实力进行对比来得到结果。使用势力图（Influence Map）对环境信息进行量化分析并预处理。

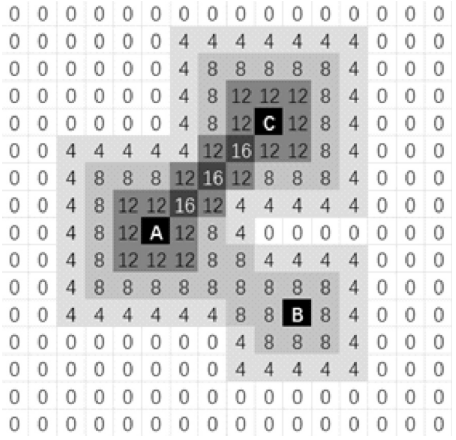


图 2 势力图

地图上每个 NPC 和角色，均根据其属性和状态量化为线性的战斗力数值。由不同角色的攻击力和移动能力，对地图位置的影响范围不同，并距离越远，其所能产生的影响力越弱。同一阵营的影响数值是可以叠加的。如图 2 所示为 32*32 地图范围内，A、B、C3 名角色所生成的势力图。

对战略决策，可以根据游戏实际情况进一步简化。将地图按小兵移动路线划分，通常可以划为三条兵线。将兵线上同一阵营单位势力做线性叠加，可以快速得出势力差值 Q。

$$Q=\sum_{k=0}^{n} f(k) - \sum_{j=0}^{m} f(j) \quad (1)$$

其中，n 为 AI 阵营单位数，m 为敌对阵营单位数，f(k)为战斗力计算式。如图 3 所示，对 Q 的值进行分段处理，通过调整 x,y,z 的值，可快速调整 AI 的决策结果。

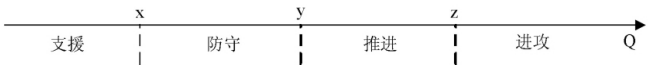


图 3 战略决策取值图

2.2 战术决策器

游戏为促进玩家间合作交流，在数值和平衡性上会避免出现单个角色过于强大的情形，而让角色间的配合更为强大，通常能够产生 1+1>2 的效果。因此，在战略决策器选择战斗方向后，设计战术决策器来针对 1~5 名角色协同对战的情况进行群体战术决策。

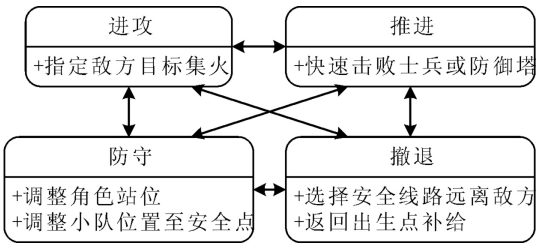


图 4 战术决策状态图

战线协作小队的状态转换图如图 4 所示。依据战略决策结果和局部势力图进行状态跳转。状态内部逻辑会为小队中角色控制器制定战术目标。

2.3 角色控制器

角色控制器内部分为两个部分：

⑴ 角色行为模块。提供 AI 控制游戏角色的行为接口。如：寻路移动至某坐标、使用某技能、购买装备等。将与游戏环境的交互封装起来，令 AI 决策结果标准化。

⑵ 逻辑控制模块。根据传入消息、当前状态、角色自身信息，决策选择 AI 使用的角色行为。

角色控制器可以在接受战术决策器的战术目标并执行的情况下，根据当前战斗环境具有一定自主逻辑。如：收到小队防守消息，角色决策时会根据小队位置和战斗方向，根据自身定位调整团队位置。远程英雄和刺客英雄会站位偏后，近战肉盾型英雄会站位偏前。

逻辑控制模块对不同的逻辑状态内部，又进一步进行决策细分。此时需决策内容已经细化到某一具体行为，决策深度较浅，因此可以选择用行为树来方便实现和维护。攻击行为树如图 5 所示。目标选择、技能选择、路线选择为序列节点（Sequence Node）依次执行。技能选择和路线选择为选择节点（Selector Node），根据条件选择其中一个执行。为防止行为树静态决策造成 AI 表现固化，引入模糊逻辑[7]，选择节点加入与环境信息相关的随机判断，通过调整随机概率令 AI 表现更加真实。

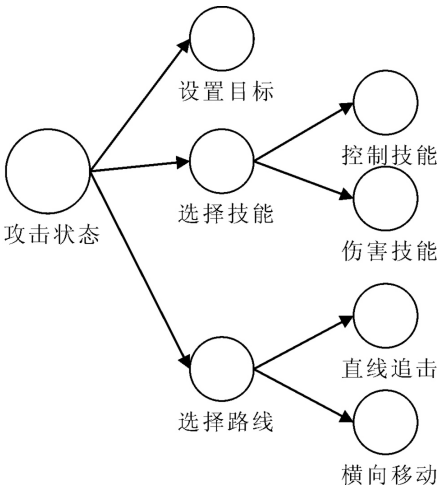


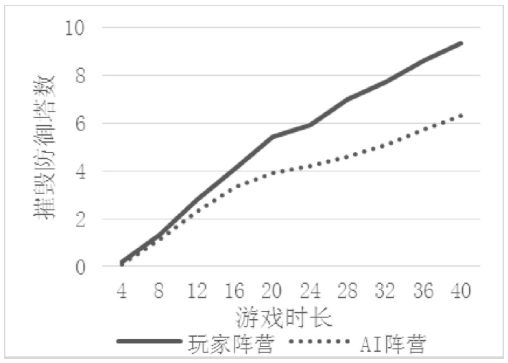
图 5 攻击行为树图

3　游戏应用效果

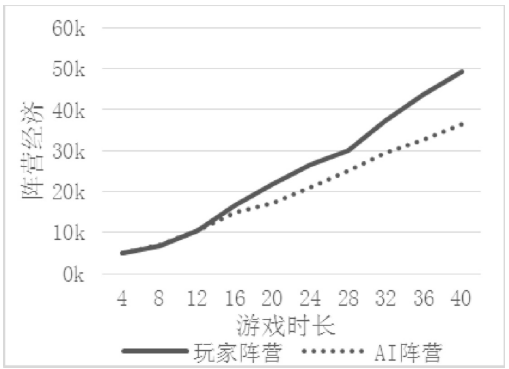
3.1　人工智能表现

根据游戏心流理论，良好的游戏体验应让玩家处于一个挑战和回报都相对匹配的狭小区域内，任意一方的失衡,都会导致焦虑或厌倦的结果[8]。相对的,玩家更喜欢胜利带来的喜悦感。因此,AI 在 MOBA 游戏中的表现,应是有一定的难度和对抗性,战局中先势均力敌,再逐渐落后,最终败北。

游戏数据分析中，通常使用摧毁敌方防御塔数 / 时间和阵营经济 / 时间数据来作为阵营对抗程度指标。从图 6 中可以看出，在游戏开始的 16 分钟内,AI 在补刀和对线上的精准逻辑设计让和玩家阵营在防御塔摧毁数和经济上均处于旗鼓相当的状态。随后游戏进入团战阶段,玩家通过团战配合逐渐拉开与 AI 的差距获得领先,最终平均在 27.3 分钟时结束游戏,玩家胜率为 94.7%。结果表明,AI 表现是符合游戏设计预期的。



(a)防御塔摧毁数对比图



(b)阵营经济对比图

图 6　AI 表现结果分析图

3.2　运行效率

AI 的运行效率影响游戏的运营成本。在 CPU 为 Intel®Xeon®Processor E5- 2643 v4 (20M Cache, 3.40 GHz)的服务器上,对比有无运行 AI 角色对战战局时的单个 CPU 负载情况。测试结果如图 7 所示。结果表明,AI 运行所增加的额外负载仅占游戏本身内容 10% 左右。

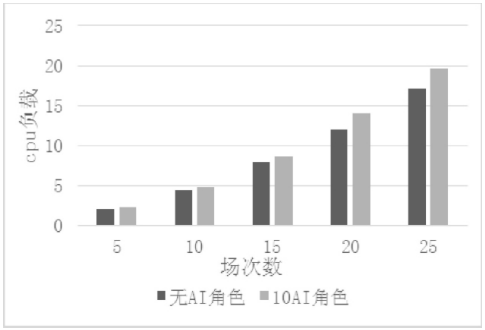


图 7　AI 运行效率分析图

4　结束语

本文提出通过分层决策的方法，将复杂的游戏人工智能决策系统分离为 N 个互不干扰的简单决策。每个简单决策所使用的逻辑方式相对独立，可以根据实际情况选择状态机、行为树等。在此基础上,尝试横向扩展各个决策组件,利用机器学习来进行宏观决策,让 AI 的逻辑行为更加拟人,是下一步的研究方向。

参考文献：

[1] 刘晓伟, 高春鸣. 结合行为树与 Q- learning 优化 UT2004 中 agent 行为决策[J].计算器工程与应用,2016,52(3):113- 118.

[2] Mat Buckland 著，吴祖增,沙鹰译.游戏编程中的人工智能技术[M].北京:清华大学出版社,2006:240- 241.

[3] Ian Millington,John Funge. Artificial Intelligence for Games[M]. Morgan KaufmannPublishers.2009:745- 750.

[4] 黄建国.基于决策模型的 AI 引擎研究与实现[D].广州:暨南大学,2010:10- 15.

[5] 王振宇.计算机游戏中智能角色行为的研究与实现[D].长沙:湖南师范大学, 2010:31- 40.

[6] Robert Nystrom，GPP 翻译组译.游戏编程模式[M].北京:人民邮电出版社,2016.12- 215.

[7] Mat Buckland 著，罗岱译. 游戏人工智能编程案例精粹[M].北京:人民邮电出版社,2008:321- 322.

[8] 腾讯游戏天美工作室群. 造物理论——游戏关卡设计指南[M].北京:电子工业出版社,2016:111- 114.