基于最优路径存储的游戏寻径算法①

宋广涛[1,2], 马 跃[2], 廉东本[2], 孙 咏[2]

[1](中国科学院 研究生院, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)

摘 要: 研究了应用于游戏中的多个路径搜寻算法, 以及游戏路径搜寻的一些特点, 提出了基于最优路径存储的寻径算法. 主要是通过最优路径矩阵存储部分的最优路径, 减少大量路径的重复计算, 提高游戏中的路径计算效率. 针对游戏场景角色的移动引起路径点通行状态的变化导致当前的最优路径失效, 提出了路径更新算法, 更新最优路径矩阵当中的最优路径. 另外, 针对地图路径点规模增大的情况, 提出了地图路径点分块处理的策略, 然后对每一子块分别使用最优路径矩阵进行路径存储.

关键词: 最优路径; 路径更新; 游戏寻径; 分块处理

Game Path Finding Algorithm Based on the Optimal Path Storage

SONG Guang-Tao[1,2], MA Yue[2], LIAN Dong-Ben[2], SUN Yong[2]

[1](Graduate School, Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)

Abstract: Researching some path finding algorithms used in the game, as well as some of the features of game path finding, and giving the routing algorithm based on the optimal path storage. Storing part of the optimal path through the optimal matrix to reduce the large number of repeated path computation, and improve the efficiency of path computation in the game. For the role movement in game scenes changing the state of path points to pass lead to the optimal path of the optimal path matrix invalid. Path update algorithm is given to update the optimal path of the optimal path matrix. As map path points become larger and larger, map paths are divided; then paths are stored for each sub-block using the optimal path matrix.

Key words: optimal path; path update; game path finding; block processin

为减少人员伤亡, 财产损失, 提高在特殊火灾和灾害事故处置行动中的成功率, 近年来基于游戏模式的模拟演练系统得到了快速的发展, 游戏寻径对于模拟演练系统又是不可或缺的一部分.

模拟演练系统中的游戏寻径算法必须能够及时的返回角色所需要的行走路径, 能够应对模拟演练系统当中短时间内发出的成百上千的路径请求, 增强了模拟演练的真实性. 寻求一个高效的游戏寻径算法对于模拟演练系统是非常重要的.

游戏当中的路径搜寻有其自身的一些特点与要求: (1)算法执行时间要短, (2)所得到的路径要平滑, (3)能够应对游戏场景当中路径点通行状态的变化, (4)大量的行走路径之间有重叠部分.

1 游戏寻径算法介绍

当前游戏当中的路径搜寻算法主要分为: 盲目式和启发式搜索两大类. 盲目式搜索是对地图当中的所有可能的路径开销进行比较, 然后得到一条最小开销的路径. 比较典型的算法有: Dijkstra, Floyd-War shall, 深度优先搜索(BFS), 广度优先搜索(DFS)等. 启发式搜索则是在路径搜索的过程当中提供一些启发信息, 进行节点扩展时选择优先级最高的路径节点, 直到扩

① 基金项目:国家水体污染控制与治理科技重大专项(2012ZX07505003)
收稿时间:2012-07-06;收到修改稿时间:2012-09-10

展到目标节点为止. 在游戏寻径当中运用较多的启发式路径搜索算法就是 A*算法.

当前所使用的游戏寻径算法, 无论是盲目式还是启发式都是从源点开始一步步的以地图路径的邻接矩阵对节点进行扩展, 最后找出最优路径. 算法得到的最优路径并没有被再次使用, 由于游戏当中有大量重叠的行走路径, 不可避免的会有大量的重复计算. 另外, 当路径长度规模增加时, 算法搜索空间增大, 效率将急剧下降.

为了有效利用已经得到的最优路径, 避免大量的重复计算, 提高最优路径的计算效率, 提出: 基于最优路径存储的路径搜寻算法. 该算法主要是通过维护一个记录着最优路径的矩阵, 最优路径计算时使用矩阵当中的最优路径对节点进行扩展, 相当于一次扩展走了多步, 从而提高路径计算效率.

重点研究内容: 首先, 最优路径矩阵的构造; 其次, 算法要能够适应场景当中路径点状态的变化; 最后, 针对路径点规模过大提出改进策略, 增强了算法的通用性.

2 基于最优路径存储的寻径算法

2.1 数据结构的准备:

地图当中路径点 NodeType 以及最优路径矩阵存储节点 PathMatrixNode 的设计, 如图 1 中所示.

下面是矩阵功能的介绍:

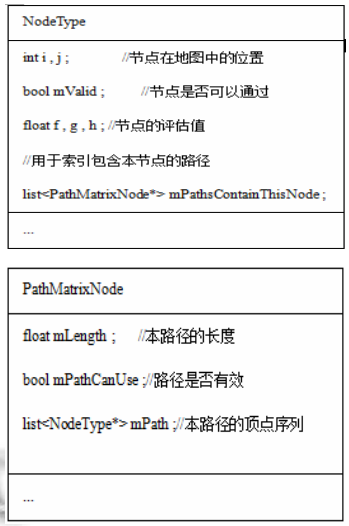

图 1 算法所用到数据的类图

NodeType Mn, 主要用于存储地图路径节点信息, 其中的最后一个数据项记录着通过此路径节点的全部最优路径, 方便在节点阻塞时修改 Mp 中路径的状态;

PathMatrixNode Ml, Ml 是地图路径的邻接矩阵, 主要是用于计算最优路径矩阵(Mp), 以及新的最优路径的计算;

PathMatrixNode Mp, Mp 是上面所提到的最优路径矩阵, 主要用于最优路径的存储与查询, 以及新的最优路径的计算, mPath 专门用于存储最优路径, mPathCanUse 标记最优路径是否可用;

各个矩阵之间的转换关系如图 2 所示:

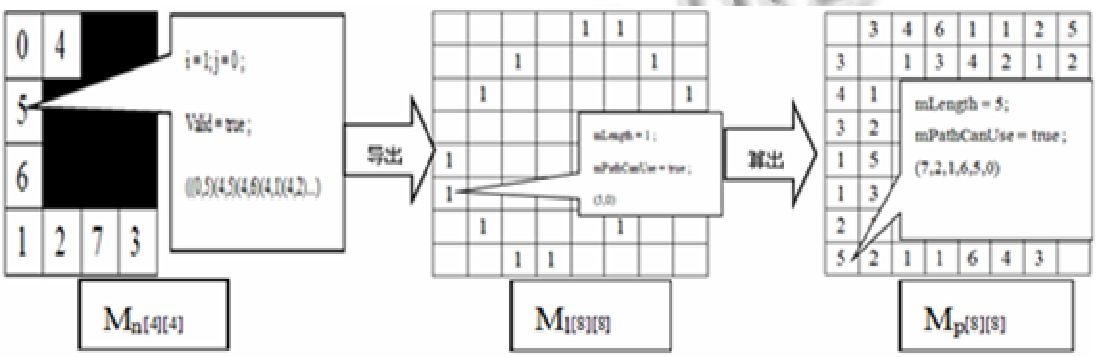

图 2 从地图路径矩阵到最优路径矩阵的转化过

2.2 算法的具体设计

2.2.1 最优路径矩阵的构建:

最优路径矩阵的构建就是 2.1 当中的 M_p, 首先, 必须对 M_p 进行初始化, 初始化是放在游戏运行前的预处理进行的.

M_p 矩阵初始化数据的获得, 使用 Floyd- Warshall 最短路径算法:

M_p <== M_l

For k from 1 to NODE_NUM

For i from 1 to NODE_NUM

For j from 1 to NODE_NUM

If M_p[i][j].mLength > M_p[i][k].mLength +M_p[k][j].mLength

{M_p[i][j].mLength = M_p[i][k].mLength + M_p[k][j].mLength ;

M_p[i][j].mPath = M_p[i][k].mPath+ M_p[k][j].mPath;}

2.2.2 路径更新算法:

当 M_p 中 M_p(S,D)的路径已经变为无效, 则需要进行重新寻径, 更新 M_p(S,D). 重新寻径使用改进的 A*算法, 具体过程如下:

(1) > 把起点 S 放入 OPEN 表, 记该结点的 f=h, 令 CLOSED 为空表.

(2) >重复下列过程, 直至找到目标节点为止, 若 OPEN 为空表, 则宣告失败.

(3) >: 利用 f=g+h, 选取 OPEN 表中未设置过的 f 具有最小值的节点为最佳节点 BESTNODE, 并把它放入 CLOSED 表.

(4) >若 BESTNODE 为目标节点, 从目标点回溯得到最优路径, 更新 Mp(S,D)中的路径, 结束算法.

(5) >若 BESTNODE 不是目标节点, 则需要对 BESTNODE 进行扩展, 若 Mp(BESTNODE, SUCCSSOR)中的路径为有效的, 则使用 Mp(BESTNODE, SUCCSSOR) 进行多步扩展, 否则使用 Ml(BESTNODE, SUCCSSOR)进行扩展, 产生后继节点 SUCCSSOR.

(6) >对每个 SUCCSSOR 进行下列过程: ①建立从 SUCCSSOR 返回 BESTNODE 的指针; ②计算 g (SUC)=g (BES)+g (BES,SUC); ③如果 SUCCSSOR∈OPEN, 则称该节点为 OLD; ④比较新旧路径代价. 如果 g(SUC)<g(OLD), 则重新确定 OLD 的父辈节点为 BESTNODE, 修改 g (OLD)=g (SUC), 并修正 f (OLD)值; ⑤若 OLD 节点的代价较低或是一样, 则停止扩展该节点; ⑥若 SUCCSSOR 不在 OPEN 表中, 则看其是否在 CLOSED 表中; ⑦若 SUCCSSOR 在 CLOSED 表中, 则转向③; ⑧若 SUCCSSOR 既不在 OPEN 表中, 又不在 CLOSED 表中, 则把它放入 OPEN 表中, 然后转向 2>.

2.2.3 基于最优路径存储的路径搜寻算法的具体过程:

算法的具体运行过程如图 3 中所示:

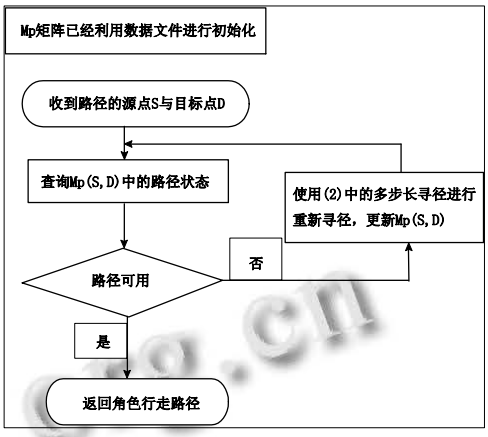

图 3　寻径算法的流程图

首先是游戏启动时, 利用预处理的数据初始化 M_p; 接下来是具体的寻径算法:

(1) >输入路径的源点 S 与目标点 D;

(2) >查找最优路径矩阵(M_p)中的 M_p(S,D)的值;

(3) >若 M_p(S,D)中的路径状态为有效, 则返回 M_p(S,D)当中的最优路径, 算法结束;

(4) >若 M_p(S,D)中的路径状态为无效, 使用(2)中的更新算法, 获得 S, D 之间的一条最优路径, 然后更新 M_p(S,D), 返回 2>;

本算法重点是对最优路径矩阵 M_p 的构建使用和更新. 步骤(4)>当中所用到的更新算法是本算法的一个重点, 在 2.2.2 当中所提出的更新算法就是一个改进的 A*算法. 改进的方面是: 节点扩展时优先使用最优路径矩阵的最优路径, 邻接矩阵作为辅助, 而这个同 A*算法的不同就是重复利用了已经得到的最优路径集合, 消除了大量的重复计算.

2.3 算法的可行性分析

算法当中的 M_p 的初始化的时间复杂度为 o(V3), 但是这个最优路径矩阵数据的生成是放在游戏运行之前的预处理进行的, 并不会影响路径的计算时间.

路径算法当中对于路径的获得主要是通过最优路径矩阵 M_p, 查询相应 M_p(S,D)中的最优路径, 当路径可用时, 其时间复杂度是 o(1); 而当 M_p(S,D)当中的路径由于地图路径节点阻断而变得不可用时, 需要使用更新算法进行重新寻径, 由于使用了最优路径矩阵进行节点扩展, 更新算法时间复杂度将远远小于 A*的 $o(V^2)$.

模拟演练系统中路径点状态只是局部变动, 而且路径点状态是趋于稳定的, 大部分路径可通过最优路径矩阵直接获得, 所以基于最优路径存储的路径搜寻算法的效率趋向于 o(1), 能够高效的计算出最优行走路径.

2.4 算法使用时应当注意的问题:

(1) >添加地图当中的路径节点时, 要尽量少的添加路径节点, 目的是减小 Mp 矩阵的存储规模, 不然矩阵的存储规模过大, 导致该算法失效. 选取的原则是: 将长期不能通过的路径节点排除掉.

(2) >在游戏运行过程当中, 应当判断导致路径不可用的动态角色的类型, 对于长期改变某一路径节点状态的动态角色(比如建筑物, 路障等)应当进行重新寻径; 而对于经常移动的角色(比如大兵, 龙骑等), 等待一个小的时间或是绕过这个角色, 不然将频繁修改最优路径矩阵, 影响寻径算法的效率.

3 地图路径点规模过大的改进策略

由于本算法的关键是最优路径矩阵的存储, 当路径顶点规模超过 $o(10^3)$时, 最优路径矩阵规模达到 $o(10^6)$, 算法变的不可行. 改进策略: 借助分治法的思想对地图路径点进行分割, 分割以后产生了各个独立的路径点区域, 然后对每个区域当中的路径点分别构建一个最优路径矩阵, 各个区域通过边界顶点构建的最优路径矩阵进行联系.

具体的路径点分割过程如图 4 所示:

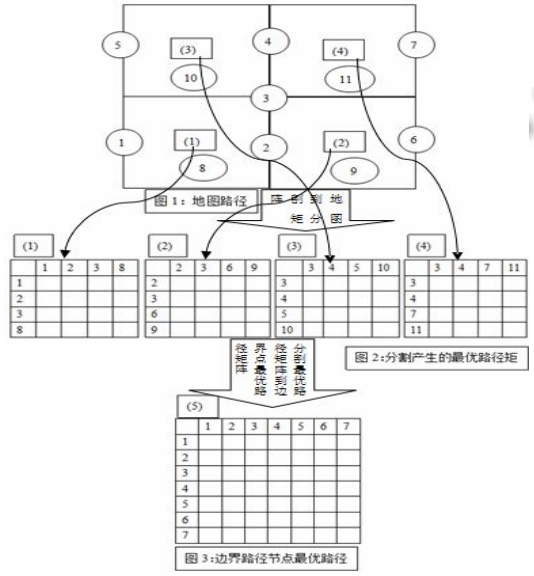

图 4 分块处理地图路径点的过程

(注: 圆形标号为路径点, 方形标号为区域号)

首先是对地图当中的路径点进行划分, 产生了四个独立的区域, 如图 4 第一层当中的(1)(2)(3)(4), 然后对应四个独立区域当中的路径点独立构建了第二层当中的四个最优路径矩阵, 同时为了将独立区域中的路径点进行联系, 通过边界点构建了第三层当中的最优路径矩阵(5).

经过图 4 当中的处理之后, 整个地图路径将产生五个最优路径矩阵, 大大的减少了内存消耗, 存储了一部分顶点对之间的最优路径. 当所需的顶点对不包含在这些最优路径矩阵中时, 可以依据这些最优路径矩阵使用更新算法进行计算.

4 结语

本文结合游戏当中角色行走路径的高重复性, 通过构建最优路径矩阵存储已经得到的最优路径供再次使用, 减少了大量的重复计算, 从而提高了寻径算法的效率. 另外, 提出了路径更新算法维护最优路径矩阵, 可应对游戏运行时对路径点通行状态的改变. 最后, 针对顶点规模过大的情况提出了一种分治的改进策略, 增强了算法的通用性.

参考文献

1 孙成江,刘林.应急救援模拟演练系统设计与实现初探.石油工业计算机应用,2010,67(3):3–6.

2 廖宁,马鸣辉.模拟演练系统综述.理论探索,2010,23(5):1–2.

3 陈和平,张前哨.A* 算法在游戏地图寻径中的应用与实现.计算机应用与软件,2005,22(12):118–120.

4 徐孝凯.数据结构实用教程.第 2 版.北京:清华大学出版社,2006.

5 Bryan S. The Basics of A* for Path Planning. Game Programming Gem. Charles River Media, 2005,254–263.