DOI：10.13274/j.cnki.hdzj.2010.12.032
2010年第12期
信息技术 Information Technology
中图分类号：TP301　文献标识码：A　文章编号：1009－2552(2010)12－0125－03

游戏地图中寻找路径的改进算法研究

周小镜

(西南大学计算机与信息科学学院，重庆 400715)

摘　要：随着计算机技术和网络技术的发展，以及人们生活水平的提高，网络游戏已经成为人们休闲娱乐生活不可或缺的一部分。网络游戏中角色按照什么路径行走，直接影响到游戏的质量。然而，由于游戏地图资源庞大和计算机资源等的限制，使得游戏中的寻路成为一个难点。通过分析现有常见的两种游戏地图寻径算法 A* 算法和单个物体寻径算法的原理及优缺点，提出一种将两种算法结合使用的改进算法，它综合了两种算法的优点，在寻径的速度和找到路径的长度方面都有显著的优势。最后，用 C++语言实现了改进的寻路算法，并和原来的算法对比，实验结果显示提出的改进算法的合理性和有效性。

关键词：地图寻径；A* 算法；单个物体寻径算法

Research on pathfinding improved algorithm in game map

ZHOU Xiao-jing

(School of Computer and Information Science, Southwest University, Chongqing 400715, China)

Abstract: With the development of computer technology and network technology, and improvement of living standard, online games have become an integral part of recreational life. The path directly affects the quality of the game. However, the huge game maps resources and the limit computer resources make find its way in games difficultly. By analyzing both A* algorithm and a single object pathfinding algorithm, including the advantages and disadvantages and the principles, the paper proposed a combination of the two methods, which combines the advantages of two algorithms in the speed of routing and the length of the path. Finally, it realizes the pathfinding improved algorithm with C++, and comparing the original algorithm. The experimental results show this improved rationality and effectiveness of the algorithm.

Key words: map pathfinding; A* algorithm; a single object pathfinding algorithm

0　引言

随着计算机和 Internet的飞速发展，网络游戏已经成为了人们休闲娱乐生活中的重要组成部分。而游戏中的路径搜索是实现游戏中 NPC(Non Player Character“非玩家控制角色”的缩写)逼真行走的关键，同时也反映了一个游戏的质量。由于计算机内存资源的限制、游戏地图的复杂性，以及对真实性的要求。游戏路径搜索成为了一个热点问题，同时也是一个难点问题，没有一个最优的解决方法，通常要综合多种方法，并且结合实际应用才能找到较好的方案。本文就游戏中的路径搜索算法展开讨论。

1　A* 算法分析

A* 算法是现目前用来解决路径搜索问题最有效的算法。A* 算法的原理是设计一个估价函数 F(n)＝G(n)＋H(n)，其中 F(n)是节点 n的估价函数，G(n)是从初始节点到 n节点的实际代价，H(n)是从节点 n到目标节点最佳路径的估计代价。根据这个函数可以计算出每个节点的代价，每次搜索时，通过这个估价函数对下一步能够到达的每一个点进行评价，找到估价值最小的点，继续往下搜

收稿日期：2010－06－18

作者简介：周小镜(1986－)，女，硕士研究生，研究方向为软件工程。

索。A* 算法的具体步骤如下：

(1)把起始节点放到开启列表 OPEN表中。

(2)如果 OPEN表为空，则失败退出，没有找到路径。

(3)如果 OPEN表不为空，就从表中选择一个 F值最小的节点 n。

(4)把 n从 OPEN 表中移出，放入关闭列表 CLOSE表中。

(5)如果 n是个目标节点，则成功退出，求得一个解，找到最优路径。否则跳转到第(6)步。

(6)扩展节点 n，生成其全部后继节点。对于 n的每一个后继节点 m，计算 F(m)。

①如果 m不在 OPEN表和 CLOSE表中，把它添入 OPEN表。给 m加一指向其父节点 n的指针。以便找到目标节点时记住路径。

②如果 m已在 OPNE表中，则比较刚计算的 F(m)新值和该节点 m在表中的 F(m)旧值。如果 F(m)新值较小，表示找到一条更好的路径，则以此新 F(m)值取代表中该节点 m的 F(m)旧值。将节点 m父指针指向当前的 n节点。

③如果 m在 CLOSE表中，则跳过该节点，返回(6)继续扩展其它节点。

(7)跳到步骤(2)，继续循环，直到找到解或无解退出。

由此可知，A* 算法每进行下一节点搜索，都是选择 F值最小的节点，因此找到的是最优路径。但是正因为如此，A* 算法每次都要扩展当前节点的全部后继节点，计算它们的 F值，找到 F值最小的节点，这样导致开启列表 OPEN表记录大量的节点信息，不仅存储量非常大，而且在查找 F值最小的节点时，非常耗时。如果游戏地图大，路径复杂，路径搜索过程则可能要计算成千上万个节点，计算量非常巨大，因此，搜索到一条路径需要一定的计算时间，这就意味着游戏运行速度降低。如果多个单位同时应用算法进行寻径时，计算量将会成倍增加，搜索效率更低，因此标准 A* 算法比较费时，会令很多玩家无法接受。

2 单个物体寻径算法分析

单个物体寻径算法的原理是从起点到终点拉一条直线 L，沿着直线 L朝终点前进，一遇到障碍物便按顺时针方向或者逆时针方向绕着障碍物行走，直到碰到直线 L为止，重复上述步骤，便一定能到达目的地。单个物体寻径算法的具体步骤如下：

(1)从起点到终点拉一条直线 L。

(2)沿着直线 L朝终点行进，一遇到障碍物便按顺时针方向或者逆时针方向绕着障碍物行走，直到碰到直线 L为止。

(3)重复步骤(2)，便一定能到达目的地。

由此可知，单个物体寻径算法，它是沿着起始点到终点的一条直线前进，如果没有障碍物，起始点到终点就是一条直线路径，当有障碍物时，按顺时针或者逆时针方向绕过障碍物之后再沿直线行走，这样计算机处理速度非常快。但是有个缺点就是按照单个物体寻径算法找到的路径不是最短的路径，有时可能围着障碍物绕很大一个圈，在游戏中走出不切实际的很奇怪的路，也是令玩家无法接受的。

3 游戏地图路径搜索改进算法

3.1 改进算法思想

通过以上分析发现，在没有障碍物的情况下，所有的最佳路径都是起始点到目标点的一条直线，如果有障碍物，则每一个拐弯点必然是障碍物的凸多边形的顶点。基于上述思想，本文提出一种改进的寻径方法，即在没有障碍物的路径上非玩家角色沿着起始点和目标点确定的这条直线朝着目标行进，当非玩家角色与障碍物的距离达到设定阈值 k时，调用 A* 算法进行路径搜索，选择最佳节点，当玩家角色与障碍物的距离超过设定阈值 k时，再次使用单个物体寻径算法，如此循环，直到到达目标点。因为两点间直线方向运动，扩展节点很少，所以计算量非常小。

本文采用的改进的寻径算法的示意图如图 1所示，其中，节点 A为起始点，节点 D为目标点，从起始点 A节点处 NPC采用的单个物体寻径算法即直线运动，当 NPC运动到节点 B时，NPC与障碍物的距离达到阈值 k时，则调用 A* 算法进行路径搜索，当 NPC运动到节点 C时，NPC与障碍物的距离大于阈值 k时，则再次使用单个物体寻径算法进行直线运动，直到到达目标节点 D。

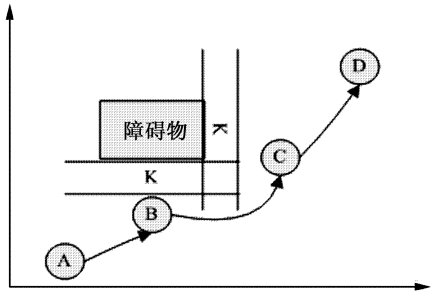

图 1 改进寻径算法示意图

3.2 改进算法的步骤

通过上面所述改进算法的思想，得出改进的路径搜索算法步骤如下，改进路径搜索算法流程图如图 2所示。

(1)设非玩家角色坐标点 S为起始点，需要到达的目的地坐标点 G为目标节点。

(2)单个物体寻径，沿着朝着目标节点方向行进。

(3)如果路径中有障碍物 O，计算 |O－S|。

①如果 |O－S|≤k(设定的阈值)，则调用 A*算法。

②否则，返回(4)。

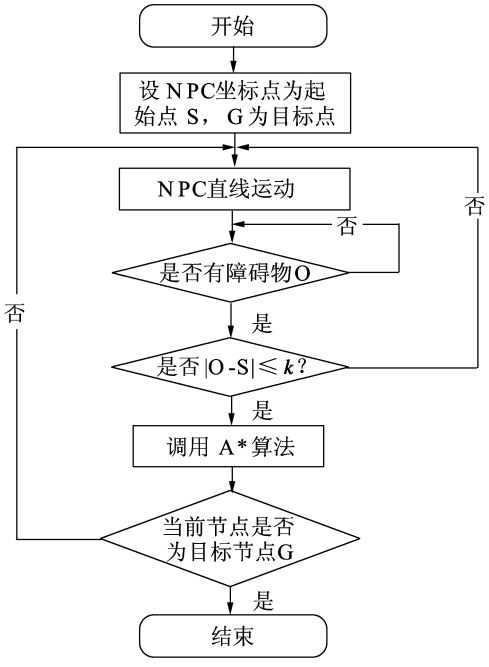

图 2 改进路径搜索算法流程图

3.3 实验与结果分析

本文采用 C++语言实现了非玩家角色的寻路功能。对改进的路径搜索算法需要建立数组来保存以下信息：当前节点的代价 g(n)，当前节点到目标节点的估计代价 h(n)，总代价 f(n)，指向父节点的指针、标识节点、非玩家角色坐标 S目的地坐标 G等数据，同时还需要存储调用 A*算法的阈值 k。

采用标准 A*算法所所得的路径如图 3所示，其中黑点表示 OPEN表中曾记录过的点。图 3为采用单个物体寻径算法所得的路径。图下为采用改进算法所得的路径，其中黑点表示在遇到障碍物时调用 A*算法，OPEN表中曾记录过的点。从图 3中，可以看出改进的寻径算法比 A*算法记录的点少，不仅减少了存储空间，而且减少了内存访问量，提高了寻径速度；改进的寻径算法比单个物体寻径算法所得路径简短。根据实验数据得出，使用改进算法使游戏的内存有效使用率提高了 29.34%。从实验结果看，采用改进算法的游戏运算速度流畅，路径真实。

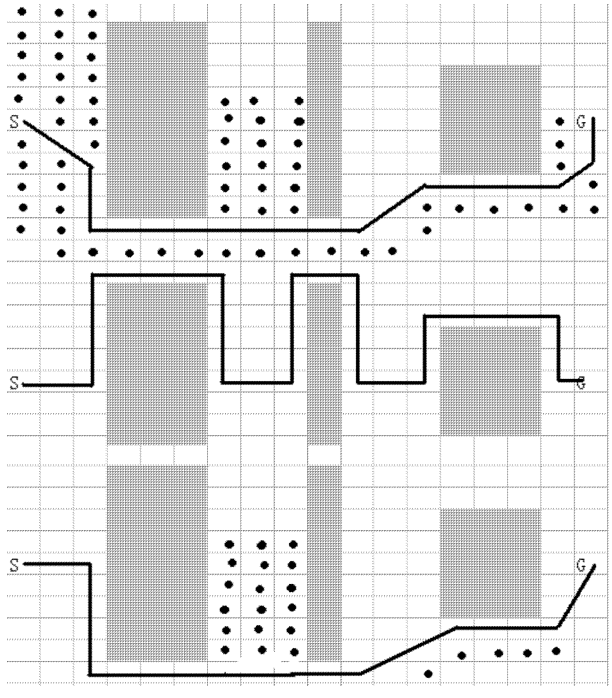

图 3 3种算法路径(A*算法(上)；单个物体寻径(中)；改进算法(下))

参 考 文 献：

[1] 陈刚，付少锋，周利华. A*算法在游戏地图寻径中的几种改进策略研究[J]. 科学技术与工程，2007，7(15).

[2] 何国辉，陈家琪. 游戏开发中智能路径搜索的算法研究[J]. 计算机工程与设计，2006，27(13)：2334－2337.

[3] 钟鼎一. 用 VC开发游戏的引擎架构设计技术[J]. 微型电脑应用，2006，22(8)：35－38.

[4] 杨青，杨磊. 3D游戏编程[M]. 北京希望电子出版社，2004.

[5] Rabin S 人工智能游戏编程真言[M]. 庄越挺，吴飞，译. 北京：清华大学出版社，2005.

[6] 李远静，莫诚生. Windows游戏编程[M]. 北京，清华大学出版社，2004.

[7] 苏羽，王媛媛. VisualC++网络游戏建模与实现[M]. 北京科海电子出版社，2003.

[8] 杨素琼，等. 基于 A*算法的地图路径搜索的实现[J]. 铁路计算机应用，2001，9(4).

[9] GameRes游戏开发资源网[EB/OL]. http://www.gameres.com.

责任编辑：肖滨