doi:10.3969/j.issn.1673-9833.2012.01.015

2D游戏地图的寻路实现

邱 磊，张 辉

（武汉船舶职业技术学院 电气与电子工程学院，湖北 武汉 430050）

摘 要：对2D游戏领域的地图寻路算法进行了研究。阐述了A*寻路算法和D*寻路算法，并在Blitz Basic 2D游戏开发平台仿真实现了2种路径搜索算法。实验结果表明：A*算法适合在静态地图中搜索路径，而D*算法在动态地图中搜索路径更迅速、有效，且随着结点数量的增加，D*算法优于A*算法。

关键词：人工智能；游戏地图；A*算法；D*算法

中图分类号：TP312 **文献标志码**：A **文章编号**：1673-9833(2012)01-0066-04

Implementation of Path Finding on 2D Game Maps

Qiu Lei，Zhang Hui

(College of Electrical and Electronic Engineering，Wuhan Institute of Shipbuilding Technology，Wuhan 430050，China)

Abstract：Researched into path finding algorithm on map in the field of 2D game. Expounded A* algorithm and D* algorithm and simulated two path-search algorithms on the Blitz Basic 2D game development platform. Experimental results indicate that A* algorithm is suitable for searching path on static game map, D* algorithm is very quickly and effectively when searching path in dynamic game map environment, and as the number of nodes increasing, the performance of D* is better than A*.

Keywords：artificial intelligence；game map；A* algorithm；D* algorithm

路径搜索算法被广泛地应用于游戏设计、地理信息系统（geographic information system，GIS）、车辆导航、计算机网络等领域，是人工智能应用于游戏中需要解决的最基本的问题之一[1]。正如Steven Woodcock所说：“AI开发者就是路径寻找者”。在游戏设计中，A*算法是被大家较广泛使用的人工智能路径搜索算法，是解决静态游戏地图中寻路问题的较好选择，但在权重等不断变化的动态地图环境下，其路径搜索效率较低；D*算法[2-3]是在地图环境不断发生变化即不能计算估价的情况下求解最短路径的典型算法。本文拟对A*和D*算法进行分析与研究，在Blitz Basic 2D游戏开发平台下分别仿真实现基于A*和D*寻路算法，并对2种算法的运算效率进行了比较。

1 A*算法和D*算法介绍

1.1 A*算法

A*算法在人工智能中是一种典型的启发式搜索算法。如果将问题求解过程表现为从初始状态到目标状态寻找最优路径的过程，则启发式搜索是在状态空间中对每一个搜索的位置进行评估，得到最好的位置，再从这个位置进行搜索直到目标状态。这样，可省略大量无谓的搜索路径，提高搜索效率。美

收稿日期：2011-08-02

基金项目：湖北省优秀中青年人才科研基金资助项目（Q20106101）

作者简介：邱 磊（1982-），男，黑龙江齐齐哈尔人，武汉船舶职业技术学院教师，主要从事计算机软件理论方面的教学与研究，E-mail：tsuly@163.com

国微软公司出品的即时战略游戏《帝国时代》中采用的寻路算法就是 A* 算法。

A* 算法所搜索的空间可以是二维网格或三维区域，也可以是传统的地图。在地图中，各个结点之间的联系要有很好的描述，且每个结点都能得到相应的子结点。如在二维网格中，某网格可以与周围上下左右 4 个方向的网格相连通，也可以与周围 8 个方向的网格相连通。A* 算法是在地图中的 2 点间找到一条最好的路径，即最短路径，路径的好坏由代价来判断。以拼接地图为例，树表示障碍物，每个网格与周围 4 个网格相连接，代价为距离，小精灵要从地图左下角走到右上角，且只能上下左右移动，小圆点表示 A* 算法计算出的小精灵到达目标结点的最短路径，如图 1 所示。

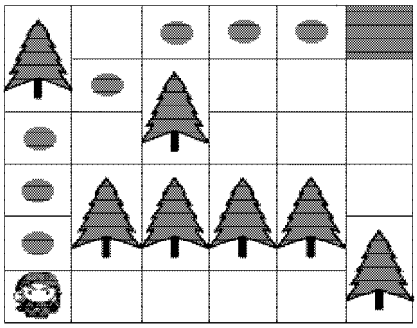

图 1　A* 算法找到的路径

Fig. 1　The path computed by A* algorithm

A* 算法通过在搜索中加入了与问题有关的启发性信息，指导搜索朝着最有希望的方向前进，加速问题的求解过程，并找到最优解[4-5]。如果一个估价函数可以找出最短的路径，则具有可采纳性。一般估价函数为：

$$f_n=g_n+h_n,$$

式中：g_n 是初始结点到任意结点 n 的代价；

h_n 是结点 n 到目标结点的启发式评估代价（heuristic estimated cost）；

估价函数 f_n 是经过结点 n 的路径估计值，值越小表示该路径越好。

A* 算法改变其行为的能力是基于启发函数 h_n，因此，提高函数 h_n 的信息量可加快路径搜索的速度和提高搜索的准确性。如果函数 h'_n 是结点 n 到目标结点的实际值，则当 $h_n \leqslant h'_n$ 时，才能确保找到一条最短路径。在 2D 地图中，结点坐标表示为（X，Y），路径长度表示代价，因此，结点 A 到目标结点 B 的启发函数 h_n 可用 2 点间最短路径表示：

$$h_A=\sqrt{(X_A-X_B)^2+(Y_A-Y_B)^2}。$$

1.2　D* 算法

在静态游戏地图中，A* 算法寻路效率较高，但在如权重等不断发生变化的动态环境中，游戏地图每变化 1 次，就要对其重新进行计算，以至于寻路效率较低。动态地图寻路算法是在外界环境不断发生变化时求解路径的方法，如在游戏中敌人或障碍物不断移动的情况下，可用 D* 算法求解寻路问题。D* 算法是动态 A*（dynamic A*）算法，由卡内及梅隆机器人中心的 Anthony Stentz 在 1994 年提出，主要用于机器人探路。1996 年 12 月美国发射的“火星探路者”探测器所携带的火星车“索杰纳”，采用了 D* 寻路算法[6-8]。

相对 A* 算法，D* 算法的主要特点是可应用于环境仅为部分已知或环境不断变化的情况下的路径寻优。该算法可根据当前已知环境信息沿着最有启发性的结点搜索，如探测到下一结点将会发生阻塞，则适时调整估价函数，改变方向继续搜索，从而最终得到整个路径。D* 算法弥补了 A* 算法必须事先知道全部环境信息的缺点。

D* 算法求解流程如下：

1）获取最短路径信息。利用 Dijkstra 算法计算出目标结点 G 到初始结点的最短路径，k 储存游戏地图中各个结点到目标结点 G 的最短路径值，h 存储各个结点到目标结点 G 的实际值。k 的初始值为 h，当某个结点的 h 值发生变化时，将该结点原 h 值与之比较，选取较小值存储到相应的 k 值中。另外，每个结点还包含下一结点到目标结点的最短路径信息，如 n_1, n_2, n_4, n_5 结点的路径信息分别为：$n_1(n_2)$, $n_2(n_5)$, $n_4(n_7)$, $n_5(n_4)$，则结点 n_1 到结点 n_4 的最短路径为 $n_1 \rightarrow n_2 \rightarrow n_5 \rightarrow n_4$，将结点信息保存在 Dijkstra 算法的 OPEN 表和 CLOSED 表中。

2）求解新的最短路径值。假设由当前位置 Y 移动到下一结点 X，如果结点 X 的状态没有变化，则无需计算，只要利用上一步 Dijkstra 算法计算的最短路径信息，从起始结点向后追溯；如果结点 X 的状态发生改变（例如堵塞），则需要调整当前位置 Y 到目标结点 G 的实际值 h_Y：

$$h_Y=C_{X,Y}+h_X,$$

式中：$C_{X,Y}$ 表示 X 到 Y 的新权值；

h_X 表示 X 的原实际值。

Y 到目标结点的方向为 $Y \rightarrow X \rightarrow G$。原 h_Y 值与变化后的 h_Y 值比较，h_Y 取较小值。

用A*算法遍历 Y 的子结点，将 Y 结点放入CLOSED表，调整子结点 X 的 h_X 值，令 $h_X=C_{Y,X}+h_Y$，其中 $C_{Y,X}$ 为 Y 到子结点 X 的权重；再判断 X 结点是否存在于 OPEN 表和 CLOSED 表中，其伪代码如下：

while( )
{
从 OPEN 表中取 k 值最小的结点 Y；
遍历 Y 的子结点 X，重新计算 $h_X=C_{Y,X}+h_Y$；
{
if(X 在 OPEN 表中) 比较 X 的 2 个 h 值；
if(X 的新 h 值小于 OPEN 表中 X 的 h 值) {
更新 OPEN 表中 X 的 h 值；
k 值取最小的 h 值；
// 有未受影响的最短路径存在
break;
}
if(X 在 CLOSED 表中) 比较 X 的两个 h 值；//注意是同一个结点的两个不同路径估价值
if(X 的新 h 值小于 CLOSE 表的 h 值) {
更新 CLOSED 表中 X 的 h 值；
k 值取最小的 h 值；
将 X 结点放入 OPEN 表；
// 有未受影响的最短路径存在
break;
}
if(X 既不在 OPEN 表也不在 CLOSED 表)
将 X 插入到 OPEN 表中；
// 此时 OPEN 表还没有排序
}
将 Y 放到 CLOSED 表中；
对 OPEN 表按 k 值大小进行排序；
}

小车在 Y 结点处找出新的子结点 X，按照新的子结点 X 指引行走，直至找到目标结点 G[9-10]。

2 算法实现与比较

2.1 A* 算法的仿真实验

A* 算法程序在 Blitz Basic 2D 游戏开发平台下实现，该算法的寻路过程如图 2 所示。

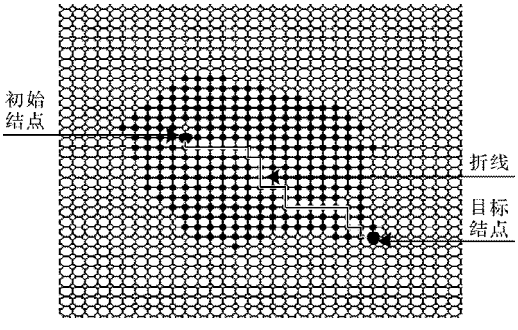

图 2 A* 算法寻路演示

Fig. 2 Path finding demo of A* algorithm

A* 算法寻路过程为：在 44 × 44（1 936 个结点）的游戏地图中选取初始结点和目标结点，实心圆为经遍历计算后所得结点，计算的结点从初始结点逐渐向目标结点的方向扩展，折线为求解的最优路径。与典型的 Dijkstra 最短路径算法相比，A* 算法是根据启发式函数的导向信息进行定向搜索，仅遍历了最短路径附近的相关结点，省去了对大量无关结点的访问，计算量明显减少，较大地提高了路径搜索效率。仿真实验结果证明了 A* 算法是处理静态游戏地图寻路问题的较佳选择。

2.2 D* 算法的仿真实验

在相同的实验平台中仿真实现 D* 算法，该算法的寻路过程如图 3 所示。

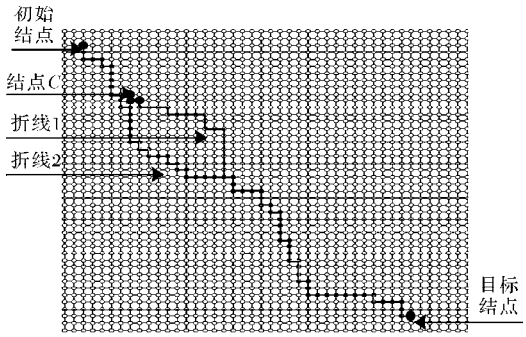

图 3 D* 算法的寻路演示

Fig. 3 Path finding demo of D* algorithm

D* 算法寻路过程为：在 44 × 44（1 936 个结点）的游戏地图上选取初始结点和目标结点，折线 1 是第 1 次计算出的最短路径；结点 C 是路径上发生了变化的障碍点，当游戏主角位于结点 C 时，检测到前面发生了路段堵塞，需要更新环境信息，该点再根据新的环境信息重新计算路径；折线 2 所经过的结点为重新计算后遍历过的结点，在向目标点移动的过程中，D* 算法仅计算了很少的结点就找到了最短路径，也就是只检查最短路径上下结点或临近结点的变化情况，修改时也是只修改障碍物附近的局部结点。折线 2 为计算出的避开了障碍物后新的最短路径，对于环境变化，整个寻路过程影响不大，且能快速完成。由仿真实验结果可知，D* 算法在动态地图环境中寻路迅速且有效。

2.3 算法比较

在 Sun Microsystems SPARC−10 处理器上，利用 Blitz Basic 2D 游戏开发平台对 2 种寻路算法的运行效率进行了比较。在 4 种不同的结点数量（从 1 000 个结点到 1 000 000 个结点）的地图中，随机产生 5 幅地图，并进行 5 次测试，分别计算从初始结点移动到目标结点所花费的 CPU 时间，再计算其 CPU 执行时

间的平均值，A* 算法与 D* 算法寻路时间比值，所得结果如表 1 所示。

表 1　A* 算法和 D* 算法的运行时间比较

Table 1　The runtime comparison of A* algorithm and D* algorithm

结点数量 / 个	算法运行时间 /s		时间比值
	A* 算法	D* 算法	
1 000	0.346	0.214	1.62
10 000	8.720	1.360	6.41
100 000	651.600	10.930	59.61
1 000 000	3 049.200	16.830	181.18

随着结点数量的增加，局部轨迹的复杂性不变，全局轨迹的复杂性将增加，当探测到未知障碍物时，D* 算法是就地重新规划轨迹，而 A* 算法要产生一个新的全局轨迹。实验结果表明，D* 算法的性能显著优于 A* 算法，且 D* 算法的运行时间高度依赖于地图的复杂度，即障碍物的数量、大小和位置，及未知障碍物占地图的比率。

3　结语

当今游戏越来越复杂，在地图寻路方面还需要针对各种特殊、复杂的情况去改进和扩展寻路算法。本文只局限于在 2D 游戏地图中运用 D* 算法求解最短路径，以后将进一步探究 A* 算法和 D* 算法在 3D 游戏地图领域的应用。

参考文献：

[1] Le Minh Duc，Amandeep Singh Sidhu，Narendra S Chaudhari. Hierarchical Pathfinding and Al-Based Learning Approach in Strategy Game Design[J]. International Journal of Computer Games Technology，2008，2008：1-11.

[2] Stentz Anthony. The Focussed D* Algorithm for Real-Time Replanning[C]//Proceedings of the International Joint Conference on Artificial Intelligence. San Mateo：Morgan Kaufmann Publishers，1995：1652-1659.

[3] Stentz Anthony. Optimal and Efficient Path Planning for Partially-Known Environments[C]//Proceedings IEEE International Conference on Robotics and Automation. San Diego：[s.n.]，1994：3310-3314.

[4] 权建洲，韩明晶，李　智. 基于改进A*算法的电子制造装备布线方法研究[J]. 中国科技论文在线，2009，4(8)：555-559.
Quan Jianzhou，Han Mingjing，Li Zhi. Study on the Wiring of Electronic Manufacturing Equipments Based on a Modified A* Algorithm[J]. Sciencepaper Online，2009，4(8)：555-559.

[5] Kumar Pawan，Bottaci Len，Mehdi Qasim，et al. Efficient Path Finding for 2D Games[C]//Proceedings of CGAIDE' 2004. Wolverhampton：University of Wolverhampton，School of Computing and Information Technology，2004：265-266.

[6] 刘　亮，郭建明. 环境部分未知的移动机器人路径规划D*算法改进[C]// 第一届智能交通与人工智能学术研讨会论文集：智能交通技术研究与应用. 广州：华南理工大学出版社，2006：260-264.
Liu Liang，Guo Jianming. Improved on D* Algorithm of Mobile Robot Path Planning in Partial Unknown Environment[C]//First Symposium on Artificial Intelligence and Intelligent Transportation：Research and Applications of Intelligent Transportation Technology. Guangzhou：South China University of Technology Press，2006：260-264.

[7] 张　颖. 基于 D* 思想的 ASON 动态均衡恢复策略研究[D]. 镇江：江苏大学，2007：27-28.
Zhang Ying. Research on D* Principle-Based Dynamic Equilibrated Restoration Strategy in ASON[D]. Zhenjiang：Jiangsu University，2007：27-28.

[8] 高　博，徐德民，张福斌. 动态目标的 Field D* 算法及路径的提取计算[J]. 火力与指挥控制，2010，35(8)：98-102.
Gao Bo，Xu Demin，Zhang Fubin. The Algorithm for the Dynamic Object Based on Field D* Algorithm and the Method of Path Extraction[J]. Fire Control & Command Control，2010，35(8)：98-102.

[9] Mackay D. Path Planning with D* Lite：Implemetation and Adaptation of the D* Lite Algorithm[R]. [S.l.]：Defence R&D Canada，2005：12.

[10] Ferguson Dave，Stentz Anthony. Field D*：An Interpolation-Based Path Planner and Replanner[M]// Springer Tracts in Advanced Robotics：Robotics Research Results of the 21th International symposium ISRR. [S.l.]：Springer, 2007：239-253.

（责任编辑：邓　彬）