基于种子算法的三维导航图自动生成算法

陈慧平,刘东峰,何家峰,程 昱
(广东工业大学 信息工程学院,广东 广州 510006)

摘 要:导航图为人群模拟提供了对应的环境信息,为智能体的移动提供了导航基础。其准确与否对模拟结果的正确性至关重要,是反应智能体自主特征与智能行为的关键技术之一。而目前工作主要针对平坦的地面进行导航图的创建,对实际应用有很大的局限性。文中利用种子填充算法蔓延特性和碰撞检测技术,并根据场景的几何属性自动生成复杂地形的三维导航图,解决了起伏地形、复杂场景导航图自动生成困难的问题。所得结果可以利用到实际人群三维模拟或三维游戏开发中。

关键词:种子算法;导航图;人群模拟
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2013)08-0055-04
doi:10.3969/j.issn.1673-629X.2013.08.014

Automatic Generation Algorithm of 3D Navigation Map Based on Seed Algorithm

CHEN Hui-ping, LIU Dong-feng, HE Jia-feng, CHENG Yu
(College of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: Navigation map provides the corresponding environmental information in the crowd simulation for agent movement, it is navigation foundation. Its accuracy or not for the correctness of the simulation results is very important, is one of key technologies reacting the agent independent characteristics and behavior. The exsiting work is mainly navigation map creation for flat ground, which has significant limitations on the practical application. In this paper, use the spread characteristic of seed filling algorithm and collision detection technology to automatically generate complex three-dimensional navigation map of the terrain, and based on the geometric properties of the scene to solve the problems of undulating terrain, complex scenes automatically generating navigational charts difficult. The results can take advantage of the actual crowd 3D simulation or 3D game development.

Key words: seed algorithm; navigation chart; crowd simulation

0 引 言

随着 3D 图形技术、虚拟现实技术及相关软件和硬件的发展,更加真实,更有视觉投入感的虚拟环境呈现在人们面前。虚拟现实下一个发展方向是提供更加真实可信的人工智能。虚拟环境中智能体的智能行为的基础是对所处环境的感知。存储在地图文件中的环境信息往往不能直接作为智能体感知的输入,而要在此基础上对三维空间进行二次建模[1,2]。

三维空间建模的方法有多种,不同的构建方法适用于不同类型的三维应用:

(1)路点法是根据设计的关卡来手动放置路点,这种方法有着一定的局限性,场景越复杂,需要手工的操作也越多,测试工作量也越大,导致生产成本过高;

(2)导航网格法是一个凸多边形的集合,将三维空间表示为由连接的多边形构成的网格,这种表示方法将多边形数据存储为节点,应用于路径搜索中。缺陷是导航网格中的凸多边形的边数并不是一个定值,对多边形的内存分配就有了更高的要求,并且场景建模中不能出现离散的三角网格,这样会产生中断,影响最终网格效果和游戏性[2,3];

(3)四叉树是将空间划分为 4 个更小的正方形,直到每个正方形都包含基本一致的地形,使用不同面

收稿日期:2012-10-25 **修回日期**:2013-01-30 **网络出版时间**:2013-04-22
基金项目:广东省教育部产学研结合项目(2009B090300401)
作者简介:陈慧平(1986-),女,江西九江人,硕士研究生,主要研究领域为计算机图形学;刘东峰,教授,博士,主要研究领域为计算机三维仿真和虚拟技术。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20130422.1722.037.html

积的正方形表示地形[4,5],更多地用于二维场景中;

(4)栅格法是将三维空间通过插值算法进行栅格化,为了更精确地描述地图,就必须使用边长较小的栅格,这种方法会导致搜索空间的膨胀,对于复杂的三维场景而言,需要大量的内存空间。

导航图是路径规划的基础,除了使用高效的搜索算法,导航图的构造与优化直接影响到路径规划的效率。导航图顶点的生成,一种是将空间分割成区域(如矩形网格图),可以由这些区域生成导航图顶点;另外一种是直接在空间中指定顶点。这两种方法在场景较小时可以很好地进行工作,但是如果场景巨大且细节复杂,第一种方法容易造成失真,并且难以表达场景细节;第二种方法可以很好地表达场景细节,然而如果频繁地更换场景,需要大量的人工手动标记顶点。

针对以上方法多用于平坦地形及部分手工操作困难的问题,文中引入三维空间自动建立搜索空间的方法:种子填充算法自动生成的导航图。将三维场景的复杂信息利用种子算法和碰撞检测技术提取出地形特征,经空间表示抽象为一幅导航图。智能体在导航图的基础上完成路径搜索、移动等智能行为[4,6]。

1 种子填充算法

种子填充算法是区域填充的一种重要算法,它广泛应用于交互式绘图系统和数字图像处理中[7]。文中对平坦地形采用递归四向连通种子填充算法,是从填充区域内部的一点开始,具体算法 1 如下:

1)从填充区域内部的一点 (x,y,z) 开始,其坐标在 stage 中找出大概中心位置,z 坐标离地面约高出一定距离。

2)由此点一定距离找到相邻四个方向上的新种子。

3)对这四个方面的新种子一一检测是否与父种子连线相碰撞,并查表是否该种子已添加作为航点,若均没有则添加新航点,并保存新航点坐标信息。

4)依次对生成的航点四个方向检测,采用递归算法添加新航点循环步骤 2。这个检测、添加过程到区域边界范围内的所有需要添加航点位置为止。

5)对已生成的航点两两相邻之间添加航线。

由于很多游戏场景的地形并非都是平坦的,针对这种复杂的三维地形,文中对上面的算法 1 进行了改进,增加了地面检测功能[8,9]。具体算法 2 如下:

1)从填充区域内部的一点 (x,y,z) 开始,其坐标在 stage 中找出大概中心位置,z 坐标离地面约高出一定距离。此距离由场景大小、个人需要决定。

2)由此点一定距离找到相邻四个方向上的子种子,如分别为 $(x-1,y,z)$、$(x+1,y,z)$、$(x,y-1,z)$、$(x,y+1,z)$。

3)对这四个方面的种子一一检测是否与父种子连线相碰撞。若子种子 $(x-1,y,z)$ 与父种子 (x,y,z) 连线没有发生碰撞则子种子 $(x-1,y,z)$ 向下检测地面得到碰撞点坐标 pos,若发生碰撞则子种子向上检测地面同样得到新碰撞点坐标 pos。图 1 是以左边第二个航点为中心检测坡面蔓延开来的平面示意图。

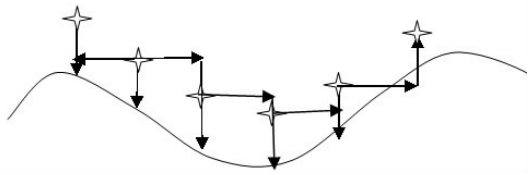

图 1 航点坡面平面图

4)查表种子 $(x-1,y)$ 坐标是否已添加作为航点信息,若没有则添加新航点 $(x-1,y,\text{pos.}z()+0.5)$,否则不添加,并保存新航点坐标信息,留下一次递归作为父种子检测使用。

5)依次对生成的航点四个方向检测,递归到步骤 2。这个检测、添加过程到区域边界范围内的所有需要添加航点位置为止。

6)对已生成的航点两两相邻之间添加航线。由于相邻航点之间的生成距离大致很接近,边的生成可以由计算机自动生成,免去了很多复杂的计算过程。

基本流程图如图 2 所示。

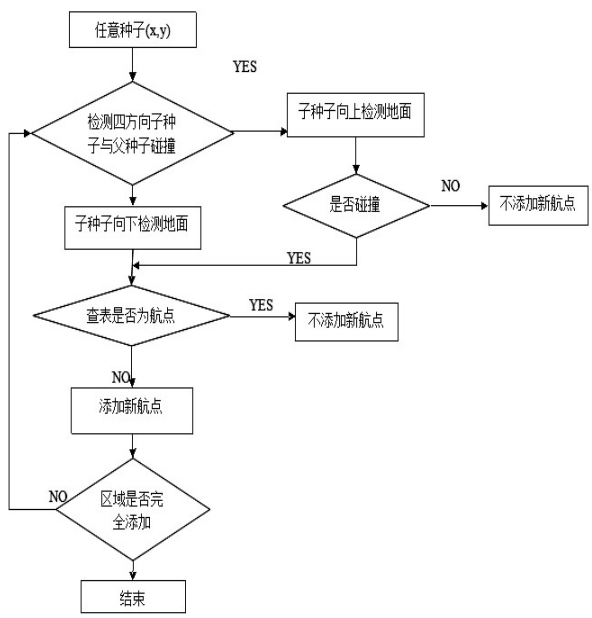

图 2 流程图

由于新产生的航点要继续向下蔓延,每次同样会检测到周围的点是否已添加,而且不同的蔓延顺序和地面的起伏不平可能导致 z 轴上下重复添加航点。此时可行的方法是通过查四周点的 x,y 坐标是否与表中已生成的点 x,y 坐标一致,无需查找航点的 x,y,z 坐标一一对应。

2　算法实现

利用 osg 开发库,delta3d 游戏开发引擎,文中实现了上述算法。

下面给出了其中主要的代码段。

2.1　航点生成

```
dfs_fill(osg::Vec3 vec)
//vec 是已经添加了的航点坐标
        x←vec.x()
        y←vec.y()
        z←vec.z()
if(IntersectStaticMesh(osg::Vec3(x,y,z),osg::Vec3(x-1,y,z))==0)//坐标 x,y,z 与 坐标 x-1,y,z 连线与地面发生碰撞则
{if(DetectPos02(osg::Vec3(x-1,y,z)))//Vec3(x-1,y,z)再向下检测地面
        If(查表 x-1,y)否则添加航点
          a[n]←x-1
          b[n]←y
          c[n]←pos.z()+0.5//记录新航点的坐标信息
}
if(IntersectStaticMesh(osg::Vec3(x,y,z),osg::Vec3(x-1,y,z))==1)//坐标 x,y,z 与 坐标 x-1,y,z 连线没有与地面发生碰撞则
{if(DetectPos01(osg::Vec3(x-1,y,z)))//Vec3(x-1,y,z)再向上检测地面
        If(查表 x-1,y)否则添加航点
        a[n]←x-1
        b[n]←y
        c[n]←pos.z()+0.5//记录新航点的坐标信息
}
```

其他三个方向上的检测依次为同样的方法,在父种子与新种子连线方向上有一点差别,四个方向上检测完毕之后,最后依次迭代蔓延至整个可通行区域为止。

2.2　查　表

```
search_waypoint(osg::Vec3 vec)//查表函数,vec 是所要查询的航点对象
{pContainer←WaypointManager::GetInstance().GetWaypoints();//所有航点均放在容器 pContainer 中
iter2←pContainer.begin()
for(;iter2!=pContainer.end();iter2++)
{osg::Vec3
vec1=(*iter2).second->GetPosition();
if(vec.x()==vec1.x() && vec.y()==vec1.y()) break;
}
if(iter2!=pContainer.end()) return false;
//没有查找完表明已有此航点 x,y 坐标
if(iter2==pContainer.end()) return true;
//直到查找完表明没有找到此航点的 x,y 坐标
}
```

2.3　碰撞检测

```
DetectPos01(osg::Vec3 vec)//检测向上方向上的障碍物的函数
{pIsector→SetDirection(osg::Vec3(0,0,1));
pIsector→SetLength(1);//检测点长度
if(pIsector→Update())
{Pos←pIsector→GetHitList()[0].getWorldIntersectPoint();//pos 得到碰撞点的三维坐标
return true;
}
else return    false;
}
```

检测向下地形的函数基本与上面函数一致,pIsector 检测方面改为向下即可。

```
IntersectStaticMesh(osg::Vec3 pos1,osg::Vec3 pos2)
{IntersectMesh→SetStartPosition(pos1);
IntersectMesh→SetEndPosition(pos2);//分别为检测点的初始点坐标
if(IntersectMesh→Update())
{pos3←IntersectMesh→GetHitList()[0].getWorldIntersectPoint();//pos3 得到两连接点之间与地面相碰撞点的坐标
return true;
}
else return false;
}
```

一般场景航点蔓延之后有些会离障碍物很近,这样导致智能体有可能钻进障碍物停滞不前。由于一般障碍物比凸起地面的垂直高度要高很多的特征,将初步建立的导航图上的航点的 Z 坐标均增加 1.5,此高度要介于凸起地面的垂直高度和障碍物的高度之间[10,11]。再对每个航点向周围八个方向检测,任何一个方向有碰撞则删除航点和边。此方法计算快,程序简单。

3　实验结果

采用 Visual Studio2008 作为编译器,文中测试了两个场景。运行环境为:双核 CPU 酷睿 i3,主频 2.4GHz,内存 2G,操作系统 Windows 7。图 3 显示种子算法蔓延的过程,图 4 显示整个虚拟环境下生成的完整的三维导航图,图 5 是在虚拟环境下,利用上述方法自动生成的楼梯建筑物的导航图。

4　结束语

文中根据区域填充算法蔓延的特性,结合碰撞检测技术来填充整个可通行面的航点。无论是平坦地形还是复杂场景地形,可行性强。该方法的正确性和有效性在试验中得到了验证,不需要手动布点,使计算机能够按照虚拟空间特征自动生成导航。

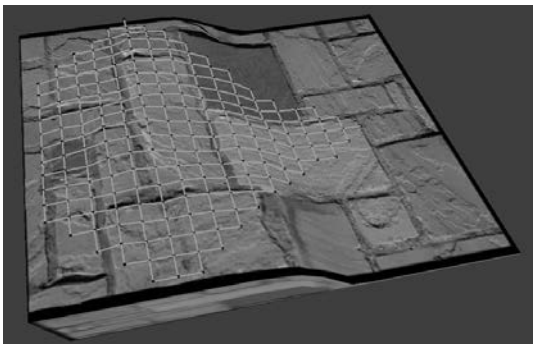

图 3 蔓延过程

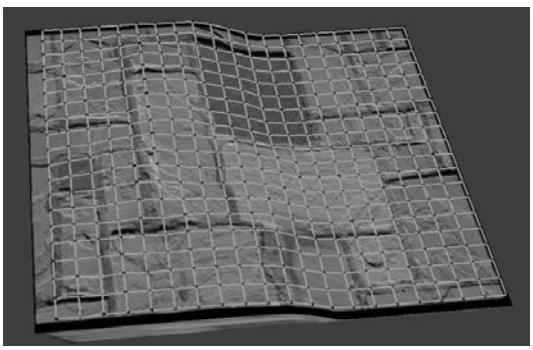

图 4 完整的三维导航图

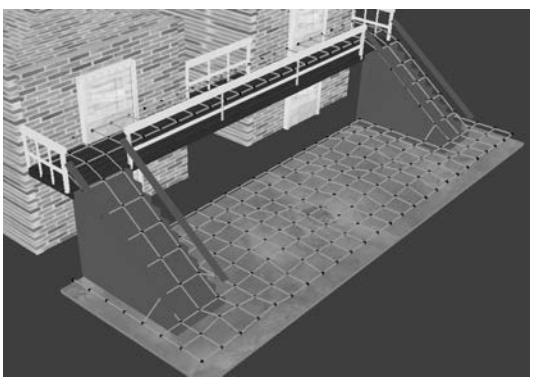

图 5 楼梯建筑物的三维导航图

参考文献:

[1] 彭文刚,彭宝林,柳 胜. 移动机器人导航系统的研究现状与发展趋势[J]. 机电工程,2009(36):69-70.

[2] 曹 雷,饶真珍,贺毅辉. 基于导航网格的三维空间表示[J]. 系统仿真学报,2008(s1):715-716.

[3] 王天顺,张 莉. 一种基于导航网格的路径搜索技术[J]. 电脑知识与技术,2010,6(12):3014-3016.

[4] 殷 宏,许继恒,周良伟,等. 基于限制四叉树的大规模地形可视化及其实现[J]. 计算机应用研究,2005(5):151-153.

[5] 任立群,高志强. 基于虚拟空间本体的路径规划[J]. 江南大学学报,2007,6(3):276-278.

[6] Pettre J, Grillon H, Thalmann D. Crowds of Moving Objects: Navigation Planning and Simulation[C]//Proc. of IEEE International Conference on Robotics and Automation. [s. l.]: [s. n.],2007:79-81.

[7] 胡 云,李盘荣. 一种改进的种子填充算法[J]. 南华大学学报,2008,6(2):57-58.

[8] Mishra S, Bande P. Maze Solving Algorithms for Micro Mouse [C]//Proc. of IEEE International Conference on Signal Image Technology and Internet Based Systems. [s. l.]:[s. n.], 2008:86-92.

[9] Wan T R, Chen H, Earnshaw R. Real-time Path Planning for Navigation in Unknown Environment[C]//Proc. of Theory and Practice of Computer Graphics. [s. l.]:[s. n.],2003: 564-567.

[10] Dang Hongshe, Song Jinguo. An Efficient Algorithm for Robot Maze-solving[C]//Proc. of International Conference on Intelligent Human-machine Systems and Cybernetics. [s. l.]: [s. n.],2010:79-82.

[11] Sharma M, Robeonics K. Algorithms for Micro-mouse[C]// Proc. of International Conference on Future Computer and Communication. [s. l.]:[s. n.],2009:581-585.

(上接第 54 页)

端设备的存储空间,需要时也可以方便的下载。

实验结果表明系统具有良好的稳定性和可行性。系统可进一步扩展其应用,完善系统性能以充分利用云服务。

参考文献:

[1] 刘 鹏. 云计算[M]. 北京:电子工业出版社,2010.

[2] 张建成,宋丽华,鹿全礼,等. 云计算方案分析研究[J]. 计算机技术与发展,2012,22(1):165-167.

[3] Sing M A, Shrivastava D M. Overview of Security Issues in Cloud Computing[J]. International Journal of Advanced Computer Research,2012,2(1):41-45.

[4] 刘会改. 面向云计算架构的资源管理的研究与实现[D]. 北京:北京邮电大学,2011.

[5] Singh G, Garg G, Jain P, et al. The Structure of Cloud Engineering[J]. International Journal of Computer Applications, 2012,33(8):33-39.

[6] 杨文志. Google Android 程序设计指南[M]. 北京:电子工业出版社,2009.

[7] Shabtai A, Kanonov U, Elovici Y. Intrusion detection on mobile devices using the knowledge based temporal-abstraction method[J]. Systems and Software,2010,83(8):1527-1536.

[8] White T. Hadoop 权威指南[M]. 中文版. 北京:清华大学出版社,2010.

[9] Rao B T, Reddy D L S S. Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environ[J]. International Journal of Computer Applications,2012,34(9):28-32.

[10] 刘 亮,霍剑青,郭玉刚,等. 基于 MVC 的通用型模式的设计与实现[J]. 中国科学技术大学学报,2010,40(6):635-639.