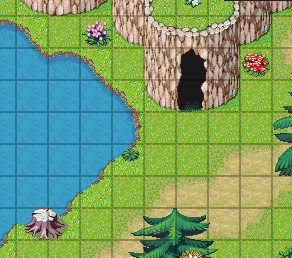
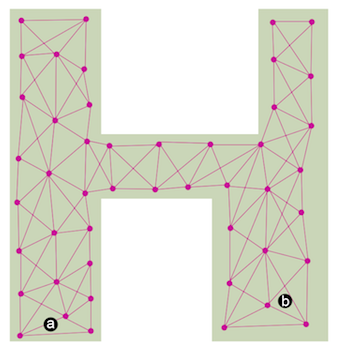
作者：伍一峰  
链接：https://www.zhihu.com/question/20298134/answer/22861904  
来源：知乎  
著作权归作者所有，转载请联系作者获得授权。

rts中的寻路系统一般需要满足有以下几个条件，  
1. 效率高，因为rts普遍地图大，单位多，所以处理效率很重要  
2. 易编辑，以便于level design  
3. 效果真实，如能找出最优（或者是看上去合理）  
4. 可以应对动态的游戏世界，例如起建筑  
  
如 [@王亞暉](//www.zhihu.com/people/461de99bf9c5ee3c71b375c803919376) 所说，一般用于寻路的**算法**是A Star，  
首先是A Star有利用到启发式函数（Heuristic Function）[1]，和另一个算法Dijkstra（A Star的无启发函数版）相比可能会更有效率，因为启发函数设计得当，可以大大减少计算的数量。  
因为启发函数的估计往往不是精确的，所以A Star [删：不像Dijkstra，] 不一定能找出人类人之上的最优解，但是对于游戏来说，看上去合理就行。  
  
然而用A Star作为寻路算法，仅仅是寻路系统的基本部分。  
作为系统，它需要有易编辑的特性。  
这就涉及到A Star中每个节点（Node）的**表现方式**。  
  
最基本的表现方式是方块（Tile），如下图 [2]  
<img src="https://pic1.zhimg.com/96ce8f90c08c851f58d74671441d81f8\_b.jpg" data-rawwidth="292" data-rawheight="258" class="content\_image" width="292">  
其中，可以将山洞所占的的几个方块设为“Not Movable”，这样A Star就会不会考虑到这几个方块，系统所生成的路径就不会碰到山洞。  
用方块作为A Star节点优点是简单，  
不过也有比较多的问题，  
第一是，如果地图很大的话，方块就会很多，这样A Star的节点就会大大增加，处理的时间相应地会增大。  
第二是，单位的移动只能是上下左右，最多加上斜行，总共八个方向，不够真实  
第三是，单位的体积大小不一样的话，大单位的图像可能会覆盖到“Not Movable”部分。以上面的图片为例，一条路径会经过在山洞边边，一个占四个方块大小的巨人走过的话，就会走在山洞上面。  
  
为了解决上面的一些问题，我们可以使用路经点（Waypoint）来做A Star节点，如下图 [3]  
<img src="https://pic1.zhimg.com/1539b841d4d4570edbb179faaff9eefc\_b.jpg" data-rawwidth="341" data-rawheight="347" class="content\_image" width="341">  
图中的红色的路径点代替了方块，成为A Star节点，这样的好处是我们可以自由地添加路径点，可以相对地减少A Star节点数目，  
同时也单位也可以按照设计师设计的方法去走。  
然而，从上图也可以看出它的问题不少，  
第一是，如果是大地图，路径点数量太少会显得生硬。  
第二是，需要考虑得面面俱到，不然一条直路忘了加路径点，单位就会“绕”（看上去）过去。  
  
为了更好地解决以上所述的问题，导航网格（Navmesh or Navigation Mesh）出现了，如下图所示 [4]  
<img src="https://pic3.zhimg.com/2d9d2bdb0a1bfde30b5f729bb52f1632\_b.jpg" data-rawwidth="538" data-rawheight="430" class="origin\_image zh-lightbox-thumb" width="538" data-original="https://pic3.zhimg.com/2d9d2bdb0a1bfde30b5f729bb52f1632\_r.jpg">  
现在，灰白色的多边形成为了A Star节点。  
它解决了上面所出现的所有问题，  
第一，从图中可以看出，节点的数目大大减少，因为多边形可以覆盖任意区域，不用限制成方块或点。除了提升计算速度之外，编辑导航网格的效率也大大增加。  
第二，通过计算直线两点和导航网格的相邻点（上图蓝色点）的位置关系，可以计算出两点是不是可以直接行走而没有阻碍物。例如上图从A点到B点通过计算可以得出可以直线行走，不用想方块和导航点那样绕来绕去。  
第三，在转角位不一定要经过相邻点，可以加上单位的体积半径，这样不同体积的单位都可以合理地通过转角。  
  
  
对于**建筑**的考虑  
在RTS中的寻路系统，还有一个很重要的话题，就是要可以应对动态的游戏世界。  
一个简单的例子就是起建筑。  
在一些需要频繁修改游戏世界的场景中，以方块为节点会更加容易作出修改 [14] ——只需要将建筑所占的方块的“Not Movable”修改成“Movable”。例如著名的塔防游戏《Field Runner》，应该是利用这种方法来实现的，而且作为塔防，《Field Runner》可以只在建塔之后寻路一次，缓存起来就行。所以在这一场景中方块又成为了一个方便快捷的选择。  
然而，导航网格也是可以动态修改的，不过开发难度会更大，而且运行中动态修改可能会造成延迟。有一些方法可以优化，例如动态地修改局部导航网格 [12]，或者是完全不修改，而将建筑看作局部的障碍物用另一套机制来应对 [13]。  
  
  
  
其实除了A Star算法之外，还有其他算法，或者技巧，可以用于RTS的寻路系统，这里简单地介绍一下，  
  
例如**Potential Field**，  
它是将地图用一个矩阵来表示，矩阵储存着大小不同的电势（整数）。  
例如，正电势表示吸引，负电势表示排斥。  
而游戏中的单位本身是一个负电势，游戏以一个数组储存所有单位的电势和位置 [7]。  
这样，在计算一个单位需要怎么从A点到B点时，我们可以用一个新的矩阵将目的地B点设成正电势，并以不同方式（如圆形、四边形等）辐射开来，离B点越远电势越低，直到0。  
然后将地图矩阵，目的地矩阵，和所有单位数组的电势相加，得出一个新的、反映当前游戏世界的电势矩阵，  
然后单位再选择周围所有电势点中的最高电势点去走。  
不过这里坑很多，因为它本质上是Greedy Algorithm，所以它未必能找出解。[5]  
然而在某些设定中，例如在没有过于复杂地形，并且需要单位自动不相互覆盖的情况下，Potential Field还是可以完成任务 [8]。  
因为相比A Star的寻路系统来说，这个方法会比较简单。  
  
还有**Flocking Behavior**，  
在对于一大群单位的寻路，计算量是很大的，而且往往会有很多的重复，这些都是可以避免的。  
如果单位的移动是利用Steering Behavior [9] 来实现的话，  
那么就可以为其中一个单位，称之为Leader，计算路径（例如用导航网格），  
然后其他单位按照以下Flocking原则来移动：  
1. 分离，避开相邻单位  
2. 一致，和整体的移动方向一致，这里应该是Leader的移动方向  
3. 聚合，向整体的平均位置靠拢  
这样的话，就可以降低寻路的计算量，并且得到更加真实的群体单位行进效果。  
  
另外一个技巧和Flocking Behavior类似 [10]，  
对于不用Steering Behavior的一大群单位，  
可以将他们设为一个组，计算这个组的路径（并且要考虑到这个组的半径以便通过转角位），  
然后给每个单位offset一个适当的距离，  
如果遇到小的通道，例如门，可以适当调整offset。  
《全面战争》里面一个队伍40人，大概用的就是这种方法 [11]。  
  
还有一个优化技巧是**Chunk** [15]。  
这个技巧和 [@王亞暉](//www.zhihu.com/people/461de99bf9c5ee3c71b375c803919376) 所提到的“先切分地图然后分块去做”应该是一致的。  
在规模宏大的地图中，为了进一步提高寻路速度，可以在编辑地图时将一些节点处理成一个Chunk，它有入口和出口，并且不同Chunk之间需要连接起来。  
从A点移动到B点，首先先在Chunk之间做寻路，得到一系列的Chunk，  
在Chunk 1的时候只需要在Chunk 1中寻路，去到Chunk 2的时候就只在Chunk 2中寻路。  
它本质上是将地图分为两种维度，一种是粗略的Chunk，一种是Chunk里面的节点（可以是方块，路径点，导航网格），并分开进行处理。有种空间分割（Space Partition）的味道在里面。  
这个方法我没有真正用过，还望大家补充。  
  
还有D Star，它主要运用在机器人领域 [6]，可以在未知环境中寻路，不过我没接触过。  
  
  
--------------Update 1----------------  
1. 增加了Potential Field的简单说明  
2. 增加了常用的启发函数例子  
3. 完善了A Star说明，指出它不一定能找出最优解  
  
--------------Update 2----------------  
1. 增加了Flocking Behavior在大群单位寻路的应用  
2. 增加了Flocking Behavior的替代技巧  
  
--------------Update 3----------------  
1. 增加了对于动态地修改游戏世界的考虑（如建筑）  
  
--------------Update 4----------------  
1. 增加了Chunk 优化技巧  
  
--------------Update 5----------------  
1. 在 [@金秉文](//www.zhihu.com/people/598b0f71a684808bd11e76b504a6def5) 的帮助下，发现A Star和最优解的一个错误，已更正。A Star的启发函数在单调的情况下是可以找出最优解的，但是这个最优解未必符合人类认知上的最优解，因为启发函数未必准确。  
  
  
**注释和资料来源：**  
[1] 启发式函数 Heuristic Function：估计路径所需的资源花费的函数，资源可以是“时间”，“体力”等等。对于精度要求不高的游戏来说，常用的启发函数是估算曼哈顿距离。  
[2] 图片来源： [Implementing Auto-tiling Functionality in a Tile Map Editor](//link.zhihu.com/?target=http://www.codeproject.com/Articles/106884/Implementing-Auto-tiling-Functionality-in-a-Tile-M" \t "_blank)  
[3] 图片来源：[http://mgrenier.me/2011/06/pathfinding-concept-the-basics/](//link.zhihu.com/?target=http://mgrenier.me/2011/06/pathfinding-concept-the-basics/" \t "_blank) （这篇博客也有讲述寻路的概念，是一个不错的学习资源）  
[4] 图片来源：[Game/AI: Fixing Pathfinding Once and For All](//link.zhihu.com/?target=http://www.ai-blog.net/archives/000152.html" \t "_blank) （这篇博客更加全面地讲述各种寻路系统的节点代表方式，值得一看）  
[5] 推荐参考：[Using Potential Fields in a Real-time Strategy Game Scenario (Tutorial)](//link.zhihu.com/?target=http://aigamedev.com/open/tutorials/potential-fields/" \t "_blank)  
[6] 参考来源：[http://en.wikipedia.org/wiki/D](//link.zhihu.com/?target=http://en.wikipedia.org/wiki/D" \t "_blank)\*  
[7] 单位可以移动，所以以数组来储存会比较方便，不用频繁更新矩阵。  
[8] 一个成功的例子：[n-created](//link.zhihu.com/?target=http://www.n-created.com/2013/08/24/potential-fields-ai-13-hello-potentials/" \t "_blank)  
[9] Steering Behavior，将一个单位考虑成一个受力点，通过增加不同的力，如吸引的，排斥的等等，实现如搜索、逃跑、躲避障碍和Flocking等行为。  
[10] [11] 资料来源：[Flanking Total War’s AI: 11 Tricks to Conquer for Your Game](//link.zhihu.com/?target=http://aigamedev.com/open/review/total-war-ai/" \t "_blank)  
[12] 动态地修改局部导航网格：[Dynamic Navigation Mesh](//link.zhihu.com/?target=http://www.staff.science.uu.nl/~gerae101/motion_planning/dynamic_navmesh.html" \t "_blank)  
[13] RV Obstacles：[http://gamma.cs.unc.edu/RVO/](//link.zhihu.com/?target=http://gamma.cs.unc.edu/RVO/" \t "_blank)  
[14] 资料来源：[A\* Pathfinding Project](//link.zhihu.com/?target=http://arongranberg.com/astar/features" \t "_blank)  
[15] 资料来源：[RTS寻路系统概要](//link.zhihu.com/?target=http://inpla.net/forum.php?mod=viewthread&tid=7300&page=1%23pid45923" \t "_blank) 中orange030的补充