Grid-Based Path-Finding

Peter Yap

Department of Computing Science, University of Alberta
Edmonton, Canada T6G 2E8
peteryap@peteryap.com

Abstract. Path-finding is an important problem for many applications, including network traffic, robot planning, military simulations, and computer games. Typically, a grid is superimposed over a region, and a graph search is used to find the optimal (minimal cost) path. The most common scenario is to use a grid of tiles and to search using A*. This paper discusses the tradeoffs for different grid representations and grid search algorithms. Grid representations discussed are 4-way tiles, 8-way tiles, and hexes. This paper introduces *texes* as an efficient representation of hexes. The search algorithms used are A* and iterative deepening A* (IDA*). Application-dependent properties dictate which grid representation and search algorithm will yield the best results.

1 Introduction

Commercial games were a $9 billion (US) industry in 1999, and the rapid rate of growth has not abated [10]. In the past, better computer graphics have been the major technological sales feature of games. With faster processors, larger memories, and better graphics cards, this has almost reached a saturation point. The perceived need for better graphics has been replaced by the demand for a more realistic gaming experience. All the major computer games companies are making big commitments to artificial intelligence [3].

Path-finding is an important problem for many applications, including transportation routing, robot planning, military simulations, and computer games. Path-finding involves analyzing a map to find the "best" cost of traveling from one point to another. Best can be a multi-valued function and use such criteria as the shortest path, least-cost path, safest path, etc. For many computer games this is an expensive calculation, made more difficult by the limited percentage of cycles that are devoted to AI processing.

Typically, a grid is superimposed over a region, and a graph search is used to find the best path. Most game programs conduct path-finding on a (rectangular) tile grid (e.g., *The Sims*, *Ages of Empire*, *Alpha Centauri*, and *Baldur's Gate*). Each tile has a positive weight that is associated with the cost to travel into that tile. The path-finding algorithm usually used is A* [2]. A few games use IDA* (Iterative Deepening A*) [4], which avoids A*'s memory overhead usually at the cost of a slower search. It is worth noting that the commercial computer games industry "discovered" A* in 1996 [9].

R. Cohen and B. Spencer (Eds.): AI 2002, LNAI 2338, pp. 44–55, 2002.

Path-finding in computer games may be conceptually easy, but for many game domains it is difficult to do well [1]. Real-time constraints limit the resources—both time and space—that can be used for path-finding. One solution is to reduce the granularity of the grid, resulting in a smaller search space. This gives a coarser representation, which is often discernible to the user (characters may follow in contorted paths). Another solution is to cheat and have the characters move in unrealistic ways (e.g., teleporting). Of course, a third solution is to get a faster processor. Regardless, the demands for realism in games will always result in more detailed domain terrains, resulting in a finer grid and a larger search space.

Most game programs decompose a terrain into a set of squares or tiles. Traditionally, one is allowed to move in the four compass directions on a tile. However, it is possible to also include the four diagonal directions (so eight directions in total). We call the latter an *octile* grid and the former a *tile* grid. Once the optimal path is found under the chosen grid, smoothing is done on this "grid-optimal" path to make it look more realistic [8].

This paper presents several new path-finding results. Grid representations discussed are tiles, octiles and the oft-overlooked hexes (for historical reasons, usually only seen in war strategy games). This paper introduces *texes* as an efficient representation of hexes. The search algorithms used are A* and iterative deepening A* (IDA*). Applicant-dependent properties dictate which grid representation and search algorithm will yield the best results. This work provides insights into different representations and their performance trade-offs. The theoretical and empirical analysis show the potential for major performance improvements to grid-based path-finding algorithms.

2 Path-Finding in Practice

Many commercial games exhibit path-finding problems. Here we highlight a few examples that we are personally familiar with. It is not our intent to make negative remarks about these products, only to illustrate that there is a serious problem and that it is widespread.

Consider Blizzard's successful multi-player game *Diablo II*. To be very brief, the player basically runs around and kills hordes of demonic minions... over and over again. To finish the game, the player usually exterminates a few thousand minions. The game involves quite a lot of path-finding, since each of these minions either chases the player, or (less commonly) runs away from the player. In the meantime, the player is rapidly clicking on the screen in an effort to either chase the minion, or (more commonly) to run away from the minion and company. All this frantic running and chasing requires path-finding computations. To complicate the matter, the player is allowed to hire NPCs (non-player characters, called hirelings) or play with other humans in an effort to kill more minions. This significantly adds to the complexity of the game in terms of path-finding.

Consider the scenario whereby a party of human players with hirelings is attacked by a very large horde of minions. From the path-finding point of view,