Research Journal

COMP130

Student number: 1607539

March 29, 2017

"A journey of a thousand miles begins with a single step - Lao Tzu"

"On second thoughts, let's not go to Camelot. 'Tis a silly place - Graham Chapman"

Pathfinding in games

Pathfinding stands out as one of the most common applications of AI research to games [1]. In practice the process must be as efficient as possible as there many are many other demands on the CPU, such as physics and graphics [2]. For the optimum solution, A* is known as one of the best search algorithms [3].

A*

By extending Dijkstra's algorithm to use heuristics relating to an end node (in Dijkstra's algorithm H is always 0) [4], A* is the *de facto* pathfinding algorithm [5], and with an admissible heuristic, returns optimal solutions. Dijkstra's algorithm was the only choice for pathfinding until the last two decades [6], and should still be used when there is no end node, for example, were there several targets and an actor were needed to move to the closest [4].

Heuristics

The Oxford dictionary defines Heuristics in regards to computing as proceeding to a solution by trial and error or by rules that are only loosely defined. A* uses a weighting for each tile defined as the sum of the distance from the start node, and the distance from the target as a heuristic (H), to find the path of least cost. Commonly written as finding the smallest value of $f(n) = g(n) + h(n)$.

The Manhattan distance is a well known admissible heuristic, and is defined as $\Delta x + \Delta y$ [1] which can be seen on an obstacle free grid as the direct route and distance. The octile distance is a variant of the Manhattan distance adapted to 8-connected maps. It is defined as $\sqrt{2} \cdot m + (M - m)$, where $m = min(\Delta x, \Delta y)$ and $M = max(\Delta x, \Delta y)$ [1], essentially examining the hypotenuse of an equilateral right angled triangle with sides of length m. $\sqrt{2}$ features prominently in square grid pathfinding although it is often approximated. The degree of accuracy should be highly pertinent, although for short paths, $\sqrt{2} \approx 1.4$ can be sufficiently accurate.

Optimizations

Given the success and popularity of A* pathfinding, there has been considerable effort to improve and refine it. Typical speed-up enhancements include reducing the size of the search space using abstraction, and building more informed heuristics [7], although the execution time, memory overhead, and whether the environment of the search system is static, dynamic, or real-time [8] must be considered. Static and dynamic systems are self explanatory, real time search works under the assumption that there is a limited amount of time to make a move. Specific per case optimization is most important. For instance, a slightly overestimating heuristic increases the speed at the expense of accuracy [9], or were the map not to feature dead ends that would require back tracking then using the node with the best heuristic as far as a goal node is favourable [10].

Dynamically changing environments [11] are a common occurrence in games and

A* has difficulty in solving the problem of random dynamics obstacles avoidance [12]. Sometimes we just want to start the movement in the correct direction before its ultimate outcome is known, as replanning from scratch every time the graph changes can be very computationally expensive [13]. Pathfinding in a smaller space abstraction can reduce the memory footprint [14] especially when the end node of the solution is subject to change. A lookahead search is also effective for such problems [15], but although looking ahead can improve the solution quality, suboptimal solutions can be produced even with admissible and consistent heuristics [16].

When maintaining the open nodes of an A* search, it is better to use an array of stacks than a priority queue. If N is the number of elements in the priority queue, the insertion and the extraction of an element both take $O(logN)$ which is faster than a constant time for inserting and a linear time for extracting as in the list implementation [17]. Correctly maintaining the priority queue is key for optimization, allowing a lazier initialisation for the node cache; it is more efficient to use an array of the size of the map as the game map already fits in memory [17].

Variations

Abstraction can be found in implementing hierarchy (HPA*) or by using a navmesh [18]. HPA* breaks down the map into sections and is significantly faster than low level A* [19], speeding up a search by replacing it with a series of smaller searches [1] gradually refined into a low level solution. It is also possible to build more than one level of abstraction for HPA* [20]. By precomputing ideal routes across these larger sections; encouraging the algorithm to expand nodes from the perimeter rather than the interior [21], or node pruning with jump points [22], symmetries can be drastically reduced, trading off the real time CPU resources with optimality of the solution.

Iterative A* is a space-efficient version of A*, but it suffers from cycles in the search space (the price for using no storage), and repeated visits to states (the overhead of iterative deepening) [23].

Block A* pre-computes a database of all possible block topologies. For each block, optimal distances between pairs of boundary cells are stored. At runtime, Block A* expands an entire block at a time, as opposed to one individual cell [1]

Always with the triangles, your solution to everything is triangles

The most common method for building navigation meshes is by vertex-based decomposition [24] collapsing corridors and tree components [25], essentially splitting the map into triangles. This has the advantage that the obtained triangular cell decomposition of the environment has $O(n)$ cells, where n is the number of segments used to describe the obstacles in the environment [26]. In fact, a triangulation A* variant is being used in the StarCraft 2 game engine [27].

Scrubbing

Due to dead ends and corners, or when the initial heuristic values are overly optimistic [28], agents can "scrub" (repeatedly revisit) the state space. State revisitation is inherently suboptimal, and can look unnatural which can reduce player immersion. It can be reduced by setting sub goals (end nodes) in the state abstractions [29].

A hill climbing agent performs no heuristic updates [30] and can brute force its way past scrubbing behaviour by only examining its neighbour. Although it must be used with a cut off number of steps to avoid unnatural behaviour, as it is vulnerable to local plateaus [31].

References

[1] A. Botea, B. Bouzy, M. Buro, C. Bauckhage, and D. Nau, "Pathfinding in games," in *Dagstuhl Follow-Ups*, vol. 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[2] N. R. Sturtevant, “Memory-efficient abstractions for pathfinding.” *AIIDE*, vol. 684, pp. 31–36, 2007.

[3] Z. A. Algfoor, M. S. Sunar, and H. Kolivand, “A comprehensive study on pathfinding techniques for robotics and video games,” *International Journal of Computer Games Technology*, vol. 2015, p. 7, 2015.

[4] P. Lester, “A* pathfinding for beginners,” *online]. GameDev WebSite. http://www. gamedev. net/reference/articles/article2003. asp (Acesso em 08/02/2009)*, 2005.

[5] R. Leigh, S. J. Louis, and C. Miles, “Using a genetic algorithm to explore a*-like pathfinding algorithms,” in *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on.* IEEE, 2007, pp. 72–79.

[6] X. Cui and H. Shi, “Direction oriented pathfinding in video games,” *International Journal of Artificial Intelligence & Applications*, vol. 2, no. 4, p. 1, 2011.

[7] D. Harabor, A. Botea *et al.*, “Breaking path symmetries on 4-connected grid maps.” in *AIIDE*, 2010.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[9] I. Millington and J. Funge, *Artificial intelligence for games.* CRC Press, 2016.

[10] J. Togelius, S. Karakovskiy, and R. Baumgarten, “The 2009 mario ai competition,” in *Evolutionary Computation (CEC), 2010 IEEE Congress on.* IEEE, 2010, pp. 1–8.

[11] A. Botea *et al.*, “Ultra-fast optimal pathfinding without runtime search.” in *AIIDE*, 2011.

[12] J.-Y. Wang and Y.-B. Lin, "Game ai: Simulating car racing game by applying pathfinding algorithms," *International Journal of Machine Learning and Computing*, vol. 2, no. 1, p. 13, 2012.

[13] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, 2005, pp. 9–18.

[14] R. Lawrence and V. Bulitko, "Database-driven real-time heuristic search in video-game pathfinding," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 3, pp. 227–241, 2013.

[15] R. E. Korf, "Real-time heuristic search," *Artificial intelligence*, vol. 42, no. 2-3, pp. 189–211, 1990.

[16] V. Bulitko, "Lookahead pathologies and meta-level control in real-time heuristic search," in *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, 2003, pp. 13–16.

[17] T. Cazenave, "Optimizations of data structures, heuristics and algorithms for path-finding on maps," in *Computational Intelligence and Games, 2006 IEEE Symposium on*. IEEE, 2006, pp. 27–33.

[18] X. Cui and H. Shi, "A*-based pathfinding in modern computer games," *International Journal of Computer Science and Network Security*, vol. 11, no. 1, pp. 125–130, 2011.

[19] A. Botea, M. Müller, and J. Schaeffer, "Near optimal hierarchical path-finding," *Journal of game development*, vol. 1, no. 1, pp. 7–28, 2004.

[20] M. R. Jansen and M. Buro, "Hpa* enhancements." *AIIDE*, vol. 7, pp. 84–87, 2007.

[21] D. Harabor, "Fast pathfinding via symmetry breaking," 2012.

[22] D. D. Harabor, A. Grastien *et al.*, "Online graph pruning for pathfinding on grid maps." in *AAAI*, 2011.

[23] Y. Björnsson, M. Enzenberger, R. C. Holte, and J. Schaeffer, "Fringe search: Beating a* at pathfinding on game maps." *CIG*, vol. 5, pp. 125–132, 2005.

[24] D. H. Hale and G. M. Youngblood, "Full 3d spatial decomposition for the generation of navigation meshes." in *AIIDE*, 2009.

[25] D. J. Demyen and M. Buro, "Efficient triangulation-based pathfinding," in *Aaai*, vol. 6, 2006, pp. 942–947.

[26] M. Kallmann, "Navigation queries from triangular meshes," in *International Conference on Motion in Games.* Springer, 2010, pp. 230–241.

[27] S. Ontanón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss, "A survey of real-time strategy game ai research and competition in starcraft," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 5, no. 4, pp. 293–311, 2013.

[28] V. Bulitko, Y. Björnsson, N. R. Sturtevant, and R. Lawrence, "Real-time heuristic search for pathfinding in video games," in *Artificial Intelligence for Computer Games.* Springer, 2011, pp. 1–30.

[29] V. Bulitko, Y. Björnsson, and R. Lawrence, "Case-based subgoaling in real-time heuristic search for video game pathfinding," *Journal of Artificial Intelligence Research*, vol. 39, no. 1, pp. 269–300, 2010.

[30] D. Huntley and V. Bulitko, "Search-space characterization for real-time heuristic search," *arXiv preprint arXiv:1308.3309*, 2013.

[31] B. L. Richards and R. J. Mooney, *Learning relations by pathfinding.* Artificial Intelligence Laboratory, University of Texas at Austin, 1992.