PLAYER-TRACED EMPIRICAL COST-SURFACES FOR A* PATHFINDING

Sam Redfern
National University of Ireland, Galway
Galway
Ireland
E-mail: sam.redfern@nuigalway.ie

KEYWORDS

A* Pathfinding, Player Modelling, Player Tracing

ABSTRACT

This paper discusses the use of empirical cost-surfaces derived from substantial amounts of player-traced movements in an online vehicular combat game, for the purposes of improving A* pathfinding by AI vehicles. Our goals include the improvement of path travel times, aesthetic improvements, and the reduction of damage sustained while travelling across the map.

The results presented include quantifiable timings and observational characteristics. Quantifiable improvements include both algorithmic efficiency and travel time efficiency, while observations include the improved ability to avoid risky terrain features as well as other subtle human-like behaviours.

A best-performing non-linear cost function for the A* algorithm, based on player data, is suggested. Continued and future work on the AI in the game is discussed.

INTRODUCTION

This paper discusses the development of empirically-derived (player-mimicking) cost surfaces for AI pathfinding in the online vehicular combat game "Darkwind: War on Wheels". This game has been developed by the author since 2005 and, since it provides a substantial player-base and thousands of live games per week, is an ideal test-bed for AI research (Redfern 2007, 2010).

Although pathfinding in general, and the A* ("A Star") algorithm in particular, are well established techniques in computer games, improvements continue to be proposed in terms of aesthetics (Coleman 2009) – producing believable 'human-like' routes – and in experimental refinements for complex environments (Hale et al. 2010). The pathfinding requirements of typical open-terrain First Person Shooter (FPS) and Real Time Strategy (RTS) games, are fundamentally simpler than those of a vehicular combat game with realistic physics, tyre and chassis degradation, and collision damage models. In Darkwind, a car's momentum is critically important to its tactical movement and performance during combat; cars receive damage from poor driving and poor surfaces; various surface characteristics exist (e.g. sand, dirt, tarmac); and, effective routes across the terrain require cover from enemy fire. Furthermore, it is often appropriate to maintain a safe distance around dangerous obstacles such as cliff edges rather than choose an absolute shortest route.

Our core hypotheses are that (i) there are a number of subtle factors related to both effectiveness and aesthetic value, which define optimal routes around the terrains, and that (ii) it may not be feasible to deal with these factors algorithmically. We aim to achieve efficient, believable ('human-like') routes which navigate terrain features and surface types sensibly, are safe from collision damage and, where possible, enemy fire. Since Darkwind is a well established online multiplayer game, it provides substantial amounts of empirical evidence about player-chosen routes. This paper describes the use of this evidence to improve AI pathfinding.

A* PATHFINDING

The A* algorithm was first proposed in 1968 (Hart et al. 1968) and has been the most widely used pathfinding technique by games programmers, due to its effectiveness and efficiency. Numerous introductory explanations are available in the literature; a particularly good online description, for example is (Lester 2005).

The fundamental operation of A* is to traverse a map by exploring promising positions (nodes) beginning at a starting location, with the goal of finding the best route to a target location. Each node has four attributes other than its position on the map:

- *g* is the cost of getting from the starting node to this node
- *h* is the estimated (heuristic) cost of getting from this node to the target node. It is a best guess, since the algorithm doesn't (yet) know the actual cost
- *f* is the sum of *g* and *h*, and is the algorithm's best current estimate as to the total cost of travelling from the starting location to the target location via this node
- *parent* is the identity of the node which connected to this node along a potential solution path

The algorithm maintains two lists of nodes, the *open* list and the *closed* list. The former consists of nodes to which the algorithm has already found a route (i.e, one of its connected neighbours has been evaluated or *expanded*) but which have not themselves, yet, been expanded. The latter (closed) list consists of nodes that have been expanded and which therefore should not be revisited.

Progress is made by identifying the most promising node in the open list (i.e., the one with the lowest *f* value) and expanding it by adding each of its connected neighbours to the open list, unless they are already closed. As nodes are expanded, they are moved to the closed list. As nodes are added to the open list, their *f, g, h* and *parent* values are recorded. The *g* value of a node is, of course, equal to the *g* value of its parent plus the cost of moving from the parent to the node itself. If a node is already on the open list when it is evaluated, its *f, g, h* and *parent* values are only updated if the new *f* value is lower than the previously recorded one – this means a better path to the node has been found than the previous one.

The algorithm concludes when the target node is found, or when the open list is empty – the latter case means that a path does not exist from source to target, which is possible when you consider that some positions on the map may be non-traversable (e.g., mountains, lakes, walls, buildings).

There are various ways of calculating the cost of moving from a node to a connected node – the simplest and most common is to use Euclidean distance. It is also very common to take into account factors such as terrain cover or elevation changes. By applying a higher cost to difficult or steep terrain, the algorithm will be encouraged to find cheaper routes around these features rather than simply finding the shortest path. The current paper is primarily concerned with the identification of an appropriate mechanism for calculating costs, based on recorded player behaviour.

The choice of heuristic function *h(n),* which estimates the *h* value for a node (the cost of getting from the node to the target location), has a strong influence on the optimality and accuracy of the identified solution. If *h(n)* is always lower than (or equal to) the cost of moving from a node to the target, then A* is guaranteed to find a shortest path. The lower *h(n)* is, the more nodes A* expands, making it slower. If *h(n)* is sometimes greater than the cost of moving from *n* to the goal, then A* is not guaranteed to find a shortest path, but it can evaluate faster (Patel, 2011). In practice, therefore, it may be possible to dynamically modify the heuristic function in order to trade-off speed and accuracy as required during a game, if this is appropriate to the game.

PREVIOUS WORK

In recent years, the research literature has increasingly stressed the fact that game AI is not simply about winning the game or discovering the most optimal solution, but more critically is about making the game fun for the human player. From a pathfinding perspective, this means avoiding mechanical-looking routes in favour of believable, human-like ones – straight lines look better and more plausible, for example, than routes which zigzag and track around obstacles (Coleman 2009). Rabin (2000) uses splines and hierarchical approaches to introduce aesthetics into routes, while Higgins (2002a) and others use a second pass through a route in order to apply "aesthetic corrections". Coleman (2007) proposes a metric based on second-order derivatives and obstacle tracking in order to quantify the "beauty intensity" of paths, and later refines this approach to include fractal dimensions and rescaled range analysis (Coleman 2009).

John et al. (2008) propose a novel approach based on probabilistic pathfinding to produce varied high-quality routes and thereby improve game replayability – their examples presented provide a convincing argument for this approach in a team-based AI combat in a maze-like environment.

Few previous papers have discussed the use of recorded player behaviours in order to train AI systems – one exception is a case-based reasoning system developed to learn high-level strategies by mining recordings of expert human players playing a real time strategy game (Ji-Lung and Chuen-Tsai 2008). No previous work that we are aware of has taken this approach for navigation purposes. However, the increasing industry emphasis on logging player interactions and movements for other purposes, such as player category modelling and game personalization (Thawonmas et al 2009; Oda et al. 2009) is expected to be reflected in an increased research interest in this area. Online games are particularly suited to this approach, since the server can easily record data centrally, and since regular updates to the game are a normal part of the lifecycle after initial release: we can gather data over a period of time and use this to incrementally improve the AI in the live game.

One technique of interest is the 'heatmap' which can be used to visualise regular patterns of player behaviour over a spatial domain (Youngblood et al. 2011). Related work also includes the use of graph-based discovery algorithms to perform supervised learning (Cook et al., 2007), and the dynamic modification of navigation meshes based on the experiences of AI agents in complex game worlds (Hale et al., 2010).

EMPIRICAL COST SURFACES FROM PLAYER TRACES

Since 2008 we have been recording player movements on the game maps of Darkwind, and constructing A* nodes from these. However, a voting system was not established until June 2010: prior to this our data simply recorded where player vehicles had ever safely travelled. We now have 12 months of voting data collected: each time a node is revisited safely, a vote is accumulated for it. There are an average of about 3000 combats played per week, and an average of about 4 player-controlled vehicles per combat, spread across about 40 game maps. Vehicles typically travel 1-2km during a combat.

In order to ensure that we record only suitable votes, a 5-second cache of recently visited nodes is stored for player vehicles; if any damage is received due to collisions with terrain or other static obstacles, the cache is emptied without committing its data.

Figure 1 provides a visualisation of the A* vote nodes stored in the region of a desert mountain in the game. Each blue square represents a node, with both size and brightness proportional to the relative number of votes accumulated at that node. In this case, the cost of travelling to a node which

has accumulated x votes, from a previous node at distance d metres, is taken to be $d / \sqrt{x}$.

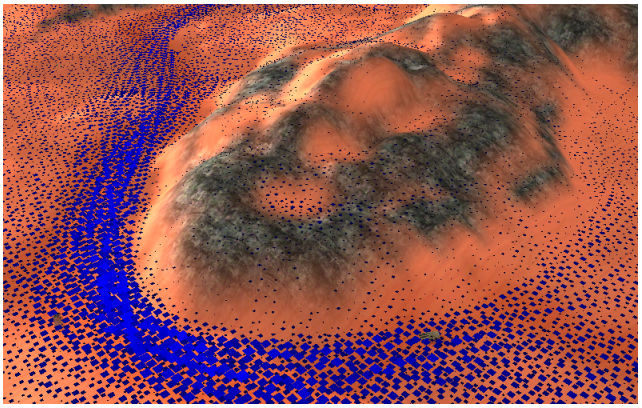

Figure 1: A Visualisation of the A* Player-Traced (Vote) Nodes Stored in the Region of a Desert Mountain in the Game

ADDITIONAL MODIFICATIONS TO THE A* ALGORITHM

Our implementation of the A* algorithm includes a number of modifications to improve efficiency and suitability for our requirements.

In order to provide rapid identification of the node closest to a world co-ordinate, nodes are pre-sorted into a world location-indexed hash table. Long distance searches are calculated using a pessimistic (high) heuristic – speeding up the search substantially, while accepting sub-optimal routes. Since the AI drivers typically re-evaluate their paths every few seconds, a guaranteed shortest route is not needed on long routes.

We also maintain a sorted shortlist of 'promising' open nodes (i.e., those with the lowest f values), which allows rapid identification of the next node to expand without the need to maintain all open nodes in a sorted list. When the shortlist is emptied, the entire open list is searched in order to refill it, and if a newly opened node has a lower f value than the worst of the 'promising' nodes, it is added to the 'promising' list. This latter performance improvement is discussed in (Higgins 2002b).

We also treat separately by direction the edge (connection) between two nodes – since in rough terrain a path may be popular in one direction but unpopular (or impossible) in the other. In figure 1, for example, the nodes on the steep sides of the mountain are effectively only connected in the downwards direction.

EVALUATION OF PLAYER-TRACED VERSUS ELEVATION-BASED COSTING

The most obvious, and often well-performing function for algorithmically defining a cost-surface is the slope (local change of elevation) of the terrain. For purposes of comparison with our votes-based function, we therefore computed costs at each node based on the average absolute difference between that node's z position (i.e., on the world's 'up' axis), and the z position of its connected neighbours. Figure 2 provides a visualisation of this scheme: the size of the squares is inversely proportional to the node's cost.

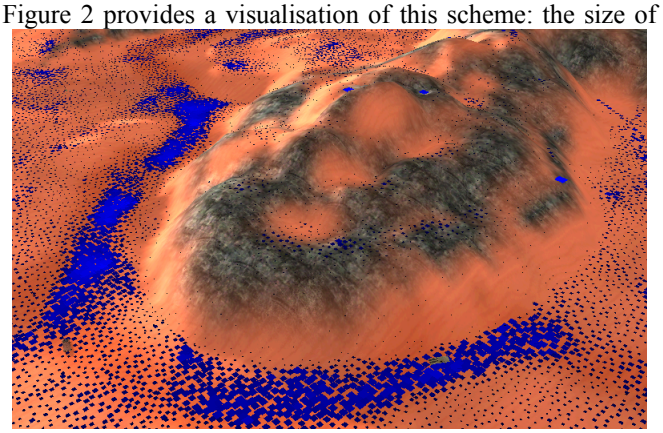

Figure 2: A Visualisation of the A* Elevation-Based Nodes Calculated Near the Same Desert Mountain

In many cases, the routes obtained when using elevation-based costing appeared very similar to those taken by the player-tracing approach. Although there were some exceptions, the general rule upon running time-trials was that the player-traced route was faster, on average by about 3%.

More importantly, however, the player-traced routes were frequently safer: elevation-based costing tended to produce routes closer to dangerous features such as cliff edges and trees. In figure 3, the route taken by the elevation-based approach was too close to the cliff, and the AI vehicle tumbled over the edge; in figure 4, the route taken through the garden caused a collision with both fencing and a tree, leading to a poor travel time. In figure 5, the route taken towards the town gates, while comparably fast, caused damage to the vehicle as it collided with the terrain while negotiating the small hills.

The game map illustrated in figure 4 consists of a ruined town with a good, wide road through its centre. The accumulation of votes indicates a very strong player preference for driving along the centre of the road. Players tend to drive fast on this road, and want to avoid collisions with fences or trees if their car spins or loses control due to weapons fire. This is a good example of subtle player behaviour that would be very hard to produce with algorithmic AI.

In terms of aesthetics, sometimes the elevation-based routes looked unnatural, especially on flat ground where features such as pits were 'edge-hugged' rather than driven around in a natural-looking way. It is probably also useful to note that, due to the underlying physical simulation, vehicles in the game are incapable of following zigzag paths due to their momentum – therefore the unsmoothed appearance of the routes illustrated in the images in this paper do not cause a problem aesthetically in the live game: we had no need to perform 'aesthetic improvement' calculations on them.

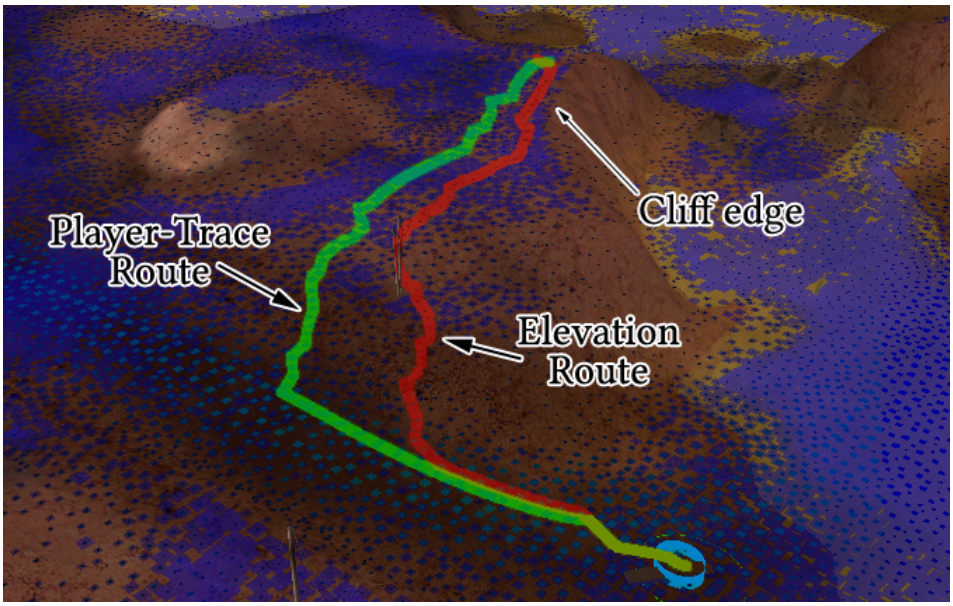

Figure 3: Elevation (Red) and Player-Traced (Green) Routes Near a Cliff

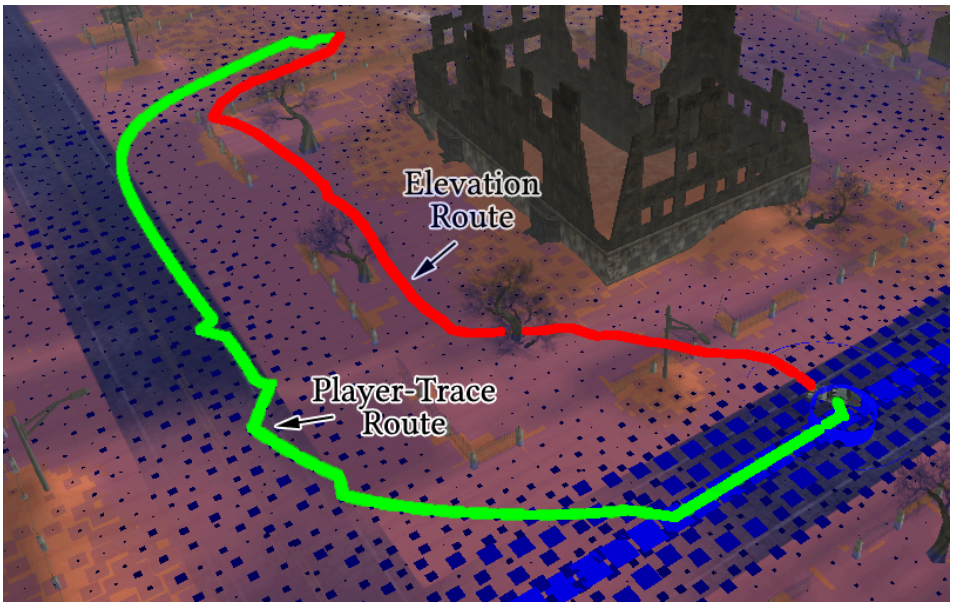

Figure 4: Elevation (Red) and Player-Traced (Green) Routes Around a Ruined House and Fenced Garden

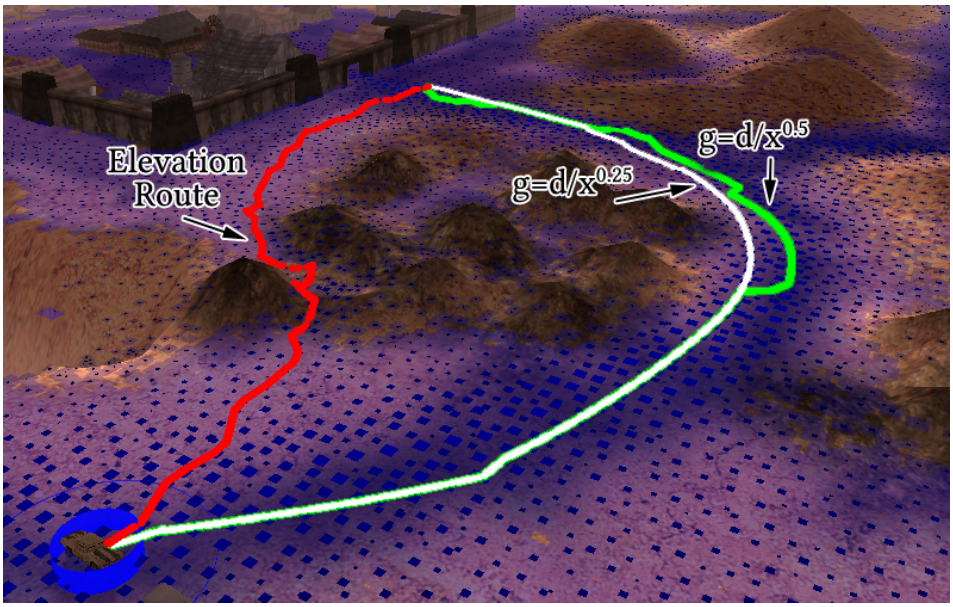

Figure 5: An Elevation (Red) Route Compared with Two Player-Traced (Green and White) Routes Which Use Different Non-Linear *g* (Cost) Functions

We frequently found the player-tracing costing approach to be more computationally efficient than the elevation approach, since it may direct the search far more tightly, expanding less nodes. This is clearly because the elevation approach often produces numerous almost-identically-scoring nodes close together. From a number of randomly-chosen tests across several maps, the performance benefit versus elevation-based costing ranged from zero (on hilly maps) to several hundred percent (on flat maps).

DEFINING THE A* PARAMETERS

We experimented with a variety of *g* cost functions, which is used to define the cost of travelling to a node based on the number of votes it has received. On safe, wide roads, it was found that a function which discriminated weakly between low amounts and high amounts of votes was more effective: for example in figure 5, the function $g=d/x^{0.25}$ produced a quicker (to travel) route than our generally best-performing function $g=d/x^{0.5}$ by about 4% - weak discrimination has a tendency to choose a shorter path towards the inside edge of corners. However, there is clearly a trade-off between speed and safety, and on routes such as the mountain in figure 1, the function $g=d/x^{0.25}$ performed poorly due to routing the car over rough terrain too close to the base of the mountain, and losing momentum: in this case, $g=d/x^{0.5}$ was quicker to travel by about 11%.

We found, again with some exceptions, that functions which discriminated very strongly between low amounts and high amounts of votes, such as $g=d/x$ or $g=d/x^2$, tended to produce erratic behaviours as the AI focused too strongly on finding 'popular' nodes, to the detriment of the overall route. This is clearly a complex situation, where factors such as the absolute number of votes cast on the map and the frequency with which games have been played on the current section of the map will have an effect. The general rule over 50 randomly-chosen test routes on various maps was that $g=d/x^{0.5}$ performed the best, on average, in terms of travel time, safety, and aesthetic value.

In order ensure an optimal path, we generally use a highly optimistic heuristic: we calculate $h = d * b$, where d is the Euclidean distance from the node to the target location and b is the cost attributed to the best-scoring node on the map. As mentioned previously, we do however vary this dynamically: a more pessimistic heuristic is used for long paths, in order to speed up the search process by expanding fewer nodes far away from the target location. To achieve this, we simply raise d to the power of 1.5 if it is larger than 50m.

CONCLUSIONS AND FUTURE WORK

We have described a novel use of player-traced navigation information as part of a voting system to inform cost-surfaces in AI A*-based navigation in a vehicular combat game with accurate physics. Experimental tests have validated the superiority of this approach over a cost-surface implementation based on local elevation changes.

We have also observed subtle behaviours in the player-traced approach, for example the avoidance of cliff edges and the preference for wide, flat routes rather than narrow gaps between terrain features. While we acknowledge that algorithmic terrain analysis could (with effort) provide *some* of these behaviours, our contention is that *every* subtlety of effective terrain navigation in this specific game has already been taken into account implicitly in the player traces.

This paper describes what is essentially a work in progress; although results are very promising and indeed Darkwind players report that they have witnessed a substantial improvement in AI navigational behaviour since the new cost function was implemented, we still have more work to do.

We intend to work on algorithmic terrain analysis, in order to produce comparable behaviours to those witnessed by the player-tracing AI. This will allow for more challenging (and therefore meaningful) comparisons between player-traced navigation behaviour and purely algorithmic AI. It will also, we hope, provide some useful algorithms of interest to AI navigation systems which cannot benefit from the wealth of player data that Darkwind has available – for example, offline single-player games and games with player-produced maps.

This paper has focused purely on the low-level navigational pathfinding task of the game AI – little has been said about the higher-level decision making which decides what the actual target locations for travel should be. The current situation in Darkwind is that a mixture of algorithmic AI techniques are used by a finite-state machine to, ultimately, produce these target locations. These techniques include simple terrain analysis (e.g. looking for 'sniper' points), group behaviours (such as re-grouping when separated, or scattering when receiving heavy ballistic damage), and outflanking behaviours which identify and respond to a 'gun line' by approaching enemies from the side. The latter tactic is in direct response to a favourite player technique in Darkwind by which a number of heavy vehicles form a static line and ambush the AI vehicles at choke points in the terrain. These techniques work reasonably in many cases, but there is generally a lack of high-level strategy or group co-ordination. The ability to navigate well across the terrain is not much use if you don't know where you want to go in the first place!

Generally, higher-level decision making needs to be improved in Darkwind, with influence maps (Tozour 2001) a likely candidate to supplement or replace the current rules-based AI. We have recently started gathering data for 'danger' influence maps, by logging the source and target positions of all successful gunfire attempts, along with the type of weapon and gunnery skill of the game character firing the weapon. Not only will this provide the data needed for 'danger' influence maps, but will also allow us to investigate our belief that gunfire avoidance is one of the subtle behaviours embedded in the player-traced data described in this paper. Additionally, we intend to experiment with an interesting approach to combining line-of-sight 'threats' into influence maps and thereby directly informing the cost function in A* path finding, as described in (van der Sterren 2002).

REFERENCES

Coleman, R. 2007. "Operationally Aesthetic Pathfinding". In *Proceedings of the International Conference on Artificial Intelligence*, 159-163.

Coleman, R. 2009. "Long Memory of Pathfinding Aesthetics". *International Journal of Computer Games Technology 2009,* 9 pages.

Cook, D.J; L.B. Holder, and G.M. Youngblood, 2007. "Graph-Based Analysis of Human Transfer Learning Using a Game Testbed". *IEEE Transactions on Knowledge and Data Engineering*, 19(11), 1465-1478.

Hale, D.H; G.M. Youngblood and N.S. Ketkar, 2010. "Using Intelligent Agents to Build Navigation Meshes". In *Proceedings FLAIRS Conference 2010*.

Hart, P.E.; N.J. Nilsson and B Raphael. 1968. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107.

Higgins, D. 2002a. "Pathfinding Design Architecture". In *AI Game Programming Wisdom*, S. Rabin (ed.), 133-145. Charles River Media.

Higgins, D. 2002b. "How to Achieve Lightning-Fast A*". In *AI Game Programming Wisdom*, S. Rabin (ed.), 114-121. Charles River Media.

Ji-Lung, H. and S. Chuen-Tsai. 2008. "Building a player strategy model by analyzing replays of real-time strategy games". *IEEE World Congress on Computational Intelligence*, 3106-3111.

John, T.C.H; E.C. Prakash and N.S. Chaudhari, 2008. "Strategic Team AI Path Plans: Probabilistic Pathfinding". *International Journal of Computer Games Technology 2008,* 6 pages.

Lester, P. 2005. "A* Pathfinding for Beginners". Retrieved 17th June 2011, http://www.policyalmanac.org/games/aStarTutorial.htm

Oda, J; R. Thawonmas and Chan, K-T. 2009. "Comparison of User Trajectories Based on Coordinate Data and State Transitions". *Proc. Fifth Int. Conf. On Intelligent Information Hiding and Multimedia Signal Processing*, 1134-1137.

Patel, A. 2011. "Heuristics". Retrieved 13th June 2011, http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Rabin, S. 2000. "Aesthetic Optimizations". In *AI Game Programming Gems*, M. DeLoura (ed.), 264-271. Charles River Media.

Redfern, S. 2007. "Psychic Software and Darkwind: War on Wheels." *New Age Gamer Magazine* 10(6), 32-35.

Redfern, S. 2010. "Evolving Racetrack Knowledge in a Racing Game". In *Proceedings AICS 2010*, 220-229.

Thawonmas, R.; J. Oda and K-T. Chen. 2009. "Analysis of User Trajectories Based on Data Distribution and State Transition: a Case Study with a Massively Multiplayer Online Game Angel Love Online". *Proc. GAME-ON2009*, 56-60.

Tozour, P. 2001. "Influence Mapping" In *Game Programming Gems 2*, M. DeLoura (ed.), 287-297. Charles River Media.

van der Sterren, W. 2002. "Tactical Path-Finding with A*". In *Game Programming Gems 3*, D. Treglia (ed.), 294-306. Charles River Media.

Youngblood, G.M.; F.W.P. Heckel; D.H. Hale, and P.N. Dixit, 2011. "Embedding Information into Game Worlds to Improve Interactive Intelligence". In *Artificial Intelligence for Computer Games*, P.A. González-Calero and M.A. Gómez-Martín (eds.), 31-53. Springer, New York.

BIOGRAPHY

SAM REDFERN attended the National University of Ireland, Galway, where he studied for a B.A. in English and Archaeology (1992), followed by an M.Sc. (1994) and Ph.D. (1998) in Information Technology. He has worked as a lecturer in Galway since 1996, and has published in the areas of digital image processing, various types of A.I., graphics, collaborative virtual environments and serious games. He has been an independent game developer in his spare time since 1984, with games published on the BBC Micro, Amiga, PC, Mac, iPhone and Android.