Strategic Path Planning on the Basis of Risk vs. Time

Ashish C. Singh and Lawrence Holder

School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164
ashish.singh@ignitionflorida.com, holder@wsu.edu

Abstract. The selection of path in an urban combat setting determines the survival to a greater extent. In this paper we propose an algorithm that finds strategic paths inside a map with a set of enemies without using predetermined waypoints. The strategic path calculation is based upon the hit probability calculated for each enemy's weapons and the risk vs. time preference and it is done at multiple levels of abstractions to address trade-off of efficiency and accuracy and the strategic path calculation minimizes both time and risk as per mission objectives.

Keywords: Strategic Path Planning, Visibility Algorithm, Risk, Time, Non-player character (NPC), Line-of-Sight (LOS), Heuristic Space Search (HSS), Military perations on Urban Terrain (MOUT).

1 Introduction

In this work we present a technique by which without using the fixed set of waypoints we can compute almost unlimited variation of paths based upon the path's risk evaluation. Thus, from the game designer's standpoint it can add to unlimited variations to the gameplay without requiring any manual marking of *navigation* or *cover point* on the map.

We have developed a strategic path planning algorithm that is based upon in-depth risk evaluations along all the possible paths that can lead to the goal. We use the hit probability for calculating the risk involved on a path. The risk calculation takes into account all the enemies.

In our work Risk is defined as the ability to shoot the player in terms of hit probability (HP). Each weapon has a different hit accuracy, rate of fire and hit ratio per bullet fired. We used weapon details [1] to obtain a HP based on distances from a set of enemies. Each weapon has a different HP. The enemy's ability to shoot the agent depends upon three factors: (1) the agent's visibility from the enemy's location, (2) the distance from the enemy, and (3) the lethality of the enemy's weapon.

A strategic path is a trade-off between the time of traversal and the risk along the path. Not all the areas along all the possible paths are completely covered. Therefore the risk evaluation must consider all of the three components of risk. We developed the strategic path computational model using these techniques in the context of a *MOUT* scenario within the Quake3 first-person shooter game.

S.M. Stevens and S. Saldamarco (Eds.) ICEC 2008, LNCS 5309, pp. 77–87, 2008.

In section 2, we compare our work with the research work done in this field. Section 3 discusses our testbed environment, the agent's interface and graph conversion of the map. Section 4 contains important concepts of strategic path computation. Section 5 presents our experimental results, and section 6 presents our conclusions.

2 Related Work

Path planning and collision prevention for single and multiple players has been extensively studied [2]. But strategic path planning has not received as much study. Shortest path planning can be done on waypoints by applying the A* algorithm [5]. But this approach neglects the strategic importance of waypoints.

Using the *BitStrings* technique Liden [4] has done strategic path planning to exhibit MOUT tactics like *flanking*. In Liden's work risk has not been studied in detail. Risk is defined by the ability of an enemy to kill the agent. Liden makes three simplifying assumptions. First, a distant enemy is considered equally risky compared to a short distant enemy and also the variation in firepower is neglected. Third, *BitStrings* can only be used for a fixed set of waypoints. In real 3D environments, the visibility complexity increases and a set of fixed waypoints cannot accurately address the strategic importance of visibility and also computed paths are limited in count. Our work addresses each of these assumptions to yield more strategically-realistic paths.

3 Urban Combat Testbed

Our experiments were performed using the *Urban Combat Testbed (UCT)* [9]. UCT is a mod of the *Quake3* first-person shooter game. The agent program exchanges percepts and actions with the UCT using a shared memory interface that allows lower communication latency and lower computational burden on the game engine.

The percepts are of two types: dynamic and static. The dynamic percepts include information about current location, health, weapons and ammunition, i.e., these percepts are meant to change with the game play. There are 33 different dynamic percepts related to the player, 11 different percepts about entities which include opponents if they are present, 4 different percepts about weapons, and all the different dynamic objects. The static percepts contain the map information. The static map information is passed to the agent using an XML description from UCT's Static Spatial Perception Service (SSPS).

The agent program can choose from 29 different actions that can be sent to the game. The actions are of very primitive form (WALK_FORWARD, TURN_RIGHT, TURN_LEFT, etc). The agent program can write actions into the assigned shared memory. The UCT reads these actions and executes them accordingly. For our study, we used the *Reykjavik map* (figure 1- left), which models an urban area.

3.1 Areas and Gateways

The walk-able surfaces in the map have been defined as *areas* [9]. These areas are 3D convex polygons. Areas have been constructed from the 3D *brushes* defined in a Quake3 map. Figure 1 (Right) shows the areas computed for the map in figure 1 (Left).

All walk-able areas are connected using *gateways*. Gateways also contain information about the type of action required to cross the gateway from one area to another area (actions like Jump, Walk, Fall, etc.). World coordinates of *areas*, *objects* and *gateways* are initially parsed using an XML file. From the dynamic and the static percepts the agent calculates the current *area* information. For traversing into another area the agent finds the *gateway* information corresponding to the present *area* and the desired next *area*. The agent sends the relevant actions in order to cross the found *gateway* to the next *area*.

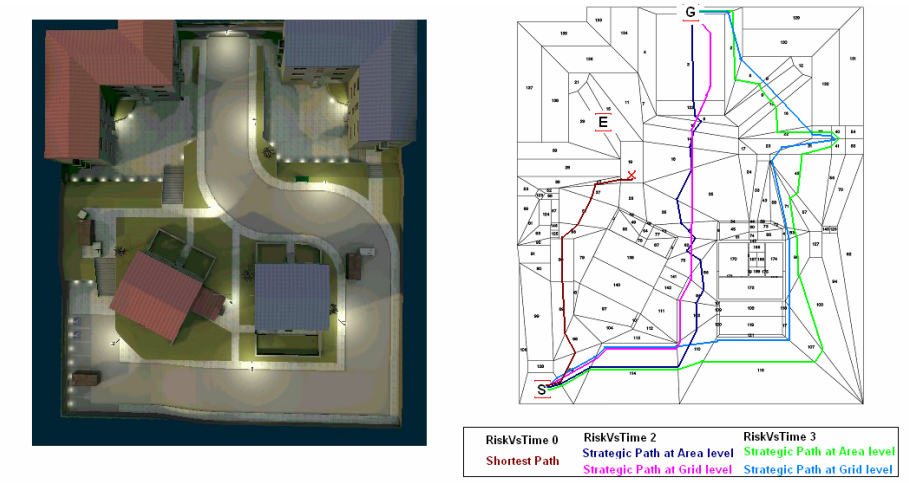

Fig. 1. (Left) The Reykjavik map [3,7] in UCT and (Right) Avg. distance of shortest paths and strategic paths over Area and Grid level

3.2 Area Connectivity Graph

Finding a path between areas becomes a problem of finding a path in the graph constructed from the connectivity information of the map. The connectivity between areas resembles edges between vertices. As in figure 2, areas and their connectivity can

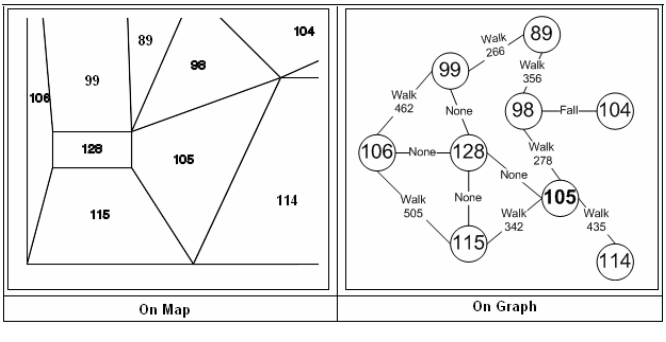

Fig. 2. Graph representing a set of Areas and Gateways

be formalized as vertices and edges of a graph. The Euclidean distances between areas become weights on the edges. In our strategic path calculation we modify these weights to incorporate risk, and then use the same shortest path algorithm to find a strategic path.

3.3 Visibility and 3D Volume Search

Polygonal areas are used both for visibility calculations and also for walk-able path calculations. We have developed the Heuristic Space Search (HSS) technique [6] that can limit the number of visibility tests to a small number of 3D areas and objects. In this technique we index the complete 3D map to a small 3D pointer array, where each pointer points to a list of 3D volumes occupying a fixed 3D space.

In strategic path calculations *brushes* that make an object have been grouped to represent one object, and similarly a set of *brushes* meant to represent a walk-able area represent one area (concept of area abstraction). Therefore, the abstracted areas and objects are larger 3D volumes, which reduce 3D related computations. The HSS constructed from these abstracted 3D volumes is better indexed. Thus, the HSS minimizes the potential 3D volumes for visibility tests and other geometric search tasks (e.g., point containment). Theoretically on perfect indexing (by using a very small HSS edge length) the visibility and other related calculations will be constant time operations.

When compared to the *Binary Space Partitioning (BSP)* technique, the *Heuristic Search Space* technique will directly reach the potential candidates while the BSP technique will search through the root node to the potential set of 3D volumes (*brushes)* by comparing *log(n)* partition planes where *n* is the total number of partition planes.

4 Strategic Path Computation

The strategic path planning is done at two levels. At the *Area level*, a higher level of abstraction, the computation gives the *areas* to walk over. At the *Grid level*, a higher level of detail, the computation gives the within-area *grid points* to walk through after reaching a selected area.

For the strategic path computation movement across the risky areas in the map have been penalized by computing a *Meta-Weight* that is based upon the *hit probability* (from all the enemies) and the given *RiskVsTime* factor. Thus priorities based upon enemies' lethality and preference for safety with respect to time of traversal can be considered together for strategic path planning. The strategic path computations have been done by modifying the weights in the weight matrix *W* of the connectivity graph of *areas* and then within each area over a set of *grid points*. We use *Dijkstra's* algorithm (computational cost: O(|E|log|V|) if a binary heap is used and O(|E| + |V|log|V|) if a Fibonacci heap is used) to compute the shortest path, which is the *strategic path*. Given the use of the HSS technique that theoretically allows visibility and area search related operations in constant time, the computational complexity of the strategic path computation is bounded by the computation complexity of the shortest path computation.

4.1 Forming a Small Set of Areas

All the walk-able surfaces in the *Reykjavik* map (figure 1 - Left) have been manually converted into a small set of large convex polygonal areas (figure 2). The *Reykjavik* map contains 125 open areas and 50 closed areas (inside buildings, etc.). These areas are connected using *gateways* which contains connection information and also the action required to move from one area to another (walk, jump, fall, none, etc.). Thus, for a small set of areas the graph representation and further application of the shortest path algorithm are computationally feasible. The abstraction of the map into a small set of areas adds to simplicity and efficiency for strategic path computation at the area level.

4.2 Hit Probability Calculation

From a start area to a goal area there can be many paths. A path is defined by a set of walk-able polygonal areas. These areas can be visible to enemies. The risk factor of a path is determined by calculating the *total hit probability* for the entire distance covered on that path.

The *hit probabilities* have been calculated from the realistic data obtained from [1]. The realistic data gives a rough conversion of distances to static hit probability for various weapons computed for a standing soldier. We convert the given static hit probability to a dynamic hit probability by considering it to be 0.25 times the static hit probability. When the agent or the enemy is moving, it will be harder to hit the agent, so taking a fraction approximates the dynamics of the situation. We considered 3 types of weapons: assault rifle (AK 47), sniper rifle (SKS-84M) and sub-machine gun (MP5).

The strategic distance between an enemy and the agent depends on how dangerous the enemy is. For example an enemy with a sniper rifle is considered more dangerous than an enemy with a lower-power weapon. Thus, the strategic path computation takes into consideration the variation in the tactical distances of the enemies.

4.2.1 Checkpoints

As shown in figure 3 (Left), a point on a path where the risk is computed has been defined as a *checkpoint*. In order to calculate the risk associated with a distance, the *checkpoints* are uniformly distributed along the path at a fixed interval. The risk associated with a distance of walk is computed as the total hit probability of receiving one hit along the *checkpoints.* In the equation 1 HP_{total} represents the total *hit probability* over the given path, and HP_i represents the *hit probability* for *checkpoint* P_i. The total *hit probability* is computed as:

$$HP_{total} = HP_1 + HP_2(1-HP_1) + \ldots + HP_n(1-HP_{n-1})(1-HP_{n-2})\ldots(1-HP_1) \quad (1)$$

Here $(1-HP_1)$ is the probability of not getting hit at checkpoint P_1 and $HP_2(1-HP_1)$ is the probability of only being hit at checkpoint P_2. Similarly, $HP_n(1-HP_{n-1})(1-HP_{n-2})\ldots(1-HP_1)$ represents the probability of only taking a hit at checkpoint HP_n (after not taking hits over the previous *checkpoints).* The *total hit probability* HP_{total} represents the probability of receiving one hit at one of the *checkpoints*.

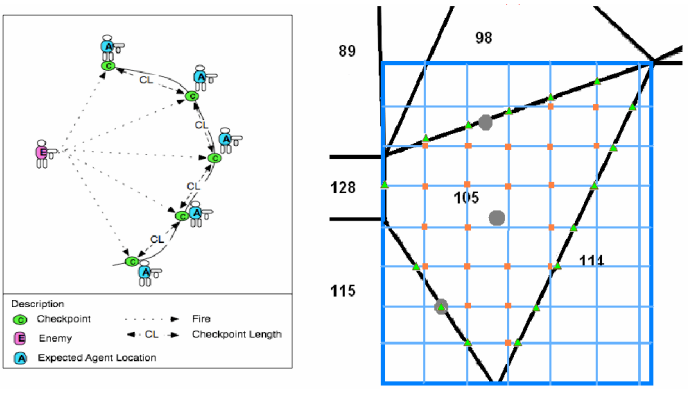

Fig. 3. (Left) Checkpoint distribution over a path and (Right) the strategic path at Grid level for Area-105

For the strategic path computation, initially the risk for each pair of neighboring areas is computed. Here, the risk is the total hit probability along the *checkpoints.* The first *checkpoint* is allocated to the center of the start area and the rest of the *checkpoints* are distributed at fixed intervals (e.g., 1m) along the path to the center of the stop area. This path goes through the connecting gateway of the two neighboring areas. Thus, the risk over a pair of areas is the total hit probability across the distributed checkpoints. The *hit probability (HP)* over this path is used for the Meta-Weight computation.

4.3 Risk vs. Time Preference Factor

Risk is attributed to the probability of a hit. In order to succeed on a mission the agent must maintain a minimum health and minimize health damage. This can be done by taking a route that keeps the agent hidden from most of the threats on the map. But not all the paths are threat free.

As per mission objectives the agent may want to reach a goal location as soon as possible and must take the shortest route towards the goal. But the shortest route may contain threats. Thus, the agent must make a trade-off in selecting a path that can minimize risks and time. Depending upon the mission objectives the preference for the shortest route compared to the preference for safety may vary. Thus, in order to maintain a good balance between the safest path and the shortest path the agent must define its risk vs. time preference factor. A high value will prioritize safety and a low value will prioritize time of traversal.

The *RiskVsTime* factor penalizes an exposed path by linearly increasing the cost of traversal of the path (Euclidean distance). The *RiskVsTime*=1 factor will double the cost of traversal over an area with hit probability 1. The *RiskVsTime*=0 factor will keep the cost of traversal over that area unaffected. As the *RiskVsTime* factor increases the strategic path computation tends toward more safety with a trade-off of longer route to the goal.

4.4 Meta-weight Calculation

After determining the risk vs. time preference and the hit probability on the connecting distance across the neighboring areas, we can compute a meta-weight. This meta-weight is multiplied to each of the weights in the weight matrix. This meta-weight represents a penalizing factor meant to symbolize the extra cost for being exposed to enemies:

$$Meta\text{-}Weight = (HP * RiskVsTime + 1) * Euclidean\ Dist. \quad (2)$$

Euclidean distance is the distance between area centers through the connecting gateway, and *HP* is the total *hit probability* over this Euclidean distance. The equation 2 takes care of the tactical priorities between more dangerous and less dangerous enemies as well as maximizing the strategic distance from the enemies based upon the *RiskVsTime* factor.

4.5 Strategic Path at Grid Level

After the strategic path at the Area level is computed, the strategic path at the Grid level (higher level of detail) is computed for each selected area while traversing the strategic path at Area level. As shown in figure 3 (Right) a selected area is further divided using a grid formation with fixed *grid unit length*. A larger *grid unit length* means more detail and more computational cost. This computation gives the within-area *grid points* to walk through for each selected area. This is done during the path traversal, so it takes into consideration any enemy movement. A selected area is subdivided using a Grid formation, and a strategic path is computed over the *grid points*.

The strategic path at Grid level computation is a two step process. First, a *gateway point* is computed on the gateway between the current area and the next selected area. The *Gateway point* is computed with the same strategic path principle, and in addition, the distance from the goal location is minimized and paths are made smoother. In the second step, *grid points* are distributed on the current area, connected to each other and to the two *gateway points* according to adjacency. The strategic path between the two *gateway points* is then computed as the strategic path at the Grid level. This technique is applied over all the selected areas (selected by the strategic path at Area level) while traversing each of the selected areas.

5 Experimental Results

We present experimental results for both *out-game* and *in-game* trials. *Out-game* trials are meant to simulate *in-game* trials so that we can perform a more systematic analysis of our approach. We performed the *Out-Game* trials using the realistic weapon details [1]. By using the *checkpoint* technique, we computed the hit probability of the generated strategic paths. The *In-Game* trials were done using the Urban Combat Testbed (a modification of Quake3). For each experimental condition (start area, goal area, enemy area), we ran 10 trials and averaged the results. For example in an experiment where the agent was hit 4 times and successfully reached the goal without getting a hit 6 times, the hit probability is 0.4. We performed the in-game trials to validate the accuracy of the out-game computation.

We computed the difference between the *in-game* and *out-game* trials (*hit probabilities* for the traced paths) with *RiskVsTime*=10 and an Assault Rifle for the enemy's weapon. For the strategic path at Area level the difference was 0.17761, and for the strategic path at Grid level the difference was 0.1896. These differences between the *in-game* and the *out-game* trials for the strategic paths are good considering the differences between real-world weapon performance and *Quake3*'s implementation of a similar weapon (similar but not the same).

The *out-game* trials were based on the realistic risk evaluation computed from available weapon details and were free from the implementation details of the Quake3 game engine. We performed *out-game* trials varying the *RiskVsTime* factor from 0 to 30. We observed that the computed strategic paths (Area and Grid levels) for a high *RiskVsTime* factor were consistently safer (consistency was statistically significant) than the shortest paths. The strategic paths at Grid level were safest among the three paths. We observed that the Grid paths were of shorter length, but in *in-game trials* it took more time to trace these paths compared to the Strategic paths. Also, the effect of variation in *checkpoint length* showed that for a shorter *checkpoint length* the *hit probability* computed for Meta-Weights was closer to the paths' evaluated *hit probability*.

5.1 Out-Game Trials for Paths

The hypothesis of this experiment was that the difference in hit probability between the shortest paths and the strategic paths is statistically significant. We computed strategic paths for 50 random experiments with one enemy. In these experiments a start area, an enemy area and a goal area were randomly selected from the available open areas. The strategic paths were computed for *RiskVsTime*=10 for an enemy with an assault rifle. We compared the shortest path and the strategic path at Area level using a t-test and found that the difference in path safety was statistically significant at the p=0.0056 level. Between the shortest path and the strategic path at Grid level the difference in path safety was statistically significant at the p= 0.00076 level. Between strategic paths at Area level and Grid level we found the difference in path safety was statistically significant at the p=0.00085 level. This confirms the claim that when seen in abstraction, an Area gives a rough estimation about its safety. And when that Area is reached and the Grid Path is then computed for that Area, this Grid Path can be consistently traversed with same or lesser risk.

5.2 Out-Game Trials for HP Variation

The hypothesis of this experiment was that with an increase in the *RiskVsTime* factor the generated strategic paths will become safer. Also with a decrease in the checkpoint length we will see a smaller difference between *hit probabilities (HP)* computed by the *checkpoint technique* for *Meta-Weight computation* and *risk evaluation* of the traced path.

Figure 4 shows that as *RiskVsTime* factor increases the *hit probability (HP)* decreases and thereby the computed path becomes safer. Also, Grid Paths are consistently safer compared to other paths. The *RiskVsTime* factor has no effect on the shortest path. When checkpoints are of smaller lengths the difference between *HP* for Meta-Weight and the Path's risk evaluation significantly decreases, and they tend to approach higher values compared to shorter checkpoint lengths.

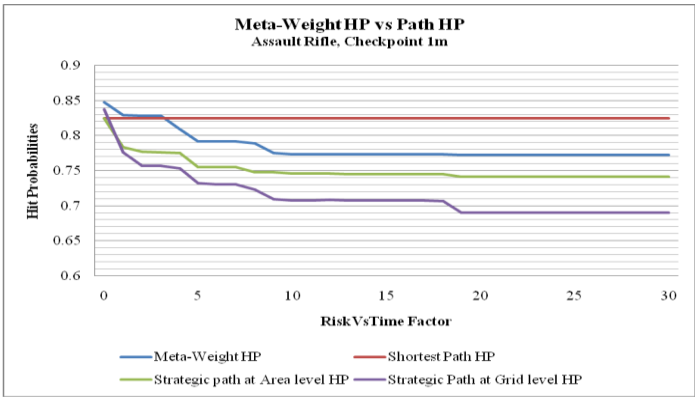

Fig. 4. HP variation between Meta-Weight computation and path's risk evaluation for 1m checkpoint length

5.3 Out-Game Trials for Path Distance

The hypothesis of this experiment was that with an increase in the *RiskVsTime* factor the distance of traversal will increase. In figure 5, for *out-game* trials as the *RiskVsTime* factor increases the strategic paths become safer at a cost of longer distances. The strategic path computation selects the shortest penalized path and in this process it tends to minimize both the risk and the distance of traversal.

As shown in figure 1 (Right) and figure 5, in the case of the strategic path at Area level the distance is the shortest distance between the *gateway points* lying at the centers of *Gateways*. And in the case of the strategic path at Grid level, these *gateway points* tend toward the goal area and strive to remain smooth over the irregular areas (using a technique similar to A*). As a result the distance is further minimized, possibly even below the shortest path length.

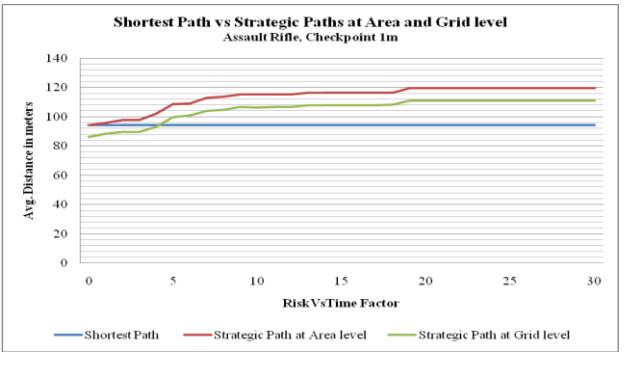

Fig. 5. Avg. distance of shortest paths and strategic paths over Area and Grid level

5.4 In-Game Trials

The hypothesis of this experiment was that the difference between the shortest paths and the strategic paths, as measured within the UCT game, is statistically significant.

For *RiskVsTime* = 10 we ran the same set of 50 experiments and compared the shortest path and the strategic path at Area level using a t-test and found that the difference in path safety was statistically significant at the p=0.006 level for the in-game trials. Between the shortest path and the strategic path at Grid level the difference in path safety was statistically significant at the p=0.0001 level for the in-game trials. Between strategic paths at Area level and Grid level we found the difference in path safety was statistically significant at the p=0.0036 level for the in-game trials.

During the in-game trials the shooting accuracy is based upon the angling calculations (weapon and the target), and the movement accuracy is based upon the bounding box [8] calculations (between objects and the agent). The movement computation tries to minimize any collision with the walls and the objects. Thus, the *in-game trials* contain many details where any technical inaccuracy could have a negative impact on the results. On the other hand for the case of *out-game trials* these game details are abstracted and do not adversely affect the analysis.

The hypothesis of this experiment was confirmed. The importance of the *in-game trials* was to check the accuracy of the *strategic path computation model*, and we found the model was consistent with the *in-game trials*.

6 Conclusions

The overall goal of this work is to improve the realism of paths taken by players in an urban warfare game. We have developed techniques for constructing strategic paths that take into account the desired risk vs. time tradeoff to find safer paths based on a model of an enemy and their different weapons. This model allows the computation of the probability that the player will be hit by the enemy while traversing the path, and therefore allows the tradeoff between risk and time. This model was evaluated using simulated trials (out-game), and the results were verified through comparison with actual in-game trials using the Urban Combat Testbed, a modification of Quake3. Results show that the model allows the selection of significantly safer paths, and that the path hit probabilities for the out-game trials are similar to those observed for the in-game trials. Future directions for this work include the further refinement of the strategic path model, extension of the approach to other maps and other MOUT games and simulators, further automation of the mechanisms for decomposing a map into areas to support area-level and grid-level strategic path planning, and ultimately integration of these techniques into MOUT game players to improve performance and realism.

Acknowledgements

This material is based on research sponsored by DARPA under agreement number FA8750-05-2-0283. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

1. Ghost Recon tactical information, http://www.ick.bz/pdfs/GR2_weapons-2005-03-25_1019.pdf
2. Kamphuis, A., Overmars, M.: Tactical Path Finding in Urban Environments. In: First International Workshop on Crowd Simulation (2005)
3. Kondeti, B., Nallacharu, M., Youngblood, G.M., Holder, L.B.: Interfacing the D'Artagnan Cognitive Architecture to the Urban Terror First-Person Shooter Game. In: IJCAI Workshop on Reasoning, Representation and Learning in Computer Games (2005)
4. Liden, L.: Using Nodes to Develop Strategies for Combat with Multiple Enemies. In: AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment (2001)
5. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs (1994)
6. Singh, A.: Improving the Survivability of Agents in a First-Person Shooter Urban Combat Simulation by Incorporating Military Skills. Master of Science Thesis, Washington State Univ. (2007)
7. Urban Terror, http://www.urbanterror.net
8. Waveren, J.P.: The Quake III Arena Bot. Master of Science thesis, Delft University of Technology (2001)
9. Youngblood, G.M., Nolen, B., Ross, M., Holder, L.B.: Building Test Beds for AI with the Q3 Mod Base. In: Artificial Intelligence in Interactive Digital Entertainment Conference (2006)