Virtual Agent Navigation in Open Spaces using Iterative Shrinking Polygons

M. Haciomeroglu R. G. Laycock A. M. Day

School of Computing Sciences, University of East Anglia, Norwich, NR4 7TJ, UK

Abstract

Populating an urban environment with a virtual crowd provides a dynamic element to an otherwise static scene; bringing the virtual environment to life. One of the fundamental components governing the fidelity of the scene is the realistic simulation of the crowd behaviour. To create a believable crowd simulation one group of methods considers constructing a graph covering the space available to the virtual agents and subsequently performing path planning to allow the agents to navigate their environment by traversing the edges of the graph. To avoid computationally expensive path planning algorithms there exists a tradeoff between the number of edges in the graph and the amount of available space which an agent can visit. In order to alleviate this problem we propose to compute the straight skeleton to provide an initial covering of the environment. This is subsequently augmented using iterative shrinking polygons to generate additional edges in the larger open spaces. The technique developed requires limited knowledge of the urban environment, processes the relevant information automatically and is illustrated in this paper to control the behaviour of a virtual crowd in real time.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Geometric algorithms, languages and systems

1. Introduction

The field of virtual human autonomous behaviour has become of great interest in recent years. Games, real time simulation systems, animated films and cinema all benefit by incorporating thousands of virtual humans, which traverse their environment automatically. If we imagine an animation system that has thousands of virtual agents, which are moving and interacting with their environment it is impossible for an animator to deal with all of these virtual agents in a reasonable amount of time. Consequently some techniques are required for automatically controlling the behaviour of virtual humans in a realistic manner. The most essential problems in the behaviour simulation are path finding and steering. Members of the virtual crowd have to automatically find their paths and follow them by using a steering model to avoid collisions with obstacles and other virtual agents. The path finding problem heavily depends on the data structure that defines the potential routes through the available free space in the environment. The main motivation of this work is to properly cover the entire free space with a graph, where each vertex represents a source or destination in the path planning problem. Computing the Medial axis of an environment is a well known technique for generating such a graph. However, the Medial axis of the environment is not sufficient for covering large free spaces. Due to the characteristics of the Medial axis computation only one path can be generated for large areas. This causes unrealistic movement effects, for instance agents move in a line even though the free space is large (for example, simulating pedestrians in a town square). In this paper an extension to the Medial axis calculation is proposed to increase the number of available routes in large areas. A combination of path population and path distance is used to decide which alternative path to choose at any given time.

2. Related Work

To enable a virtual agent to traverse an environment each agent requires knowledge of its surroundings. This knowledge is typically encoded into an environment map and is utilised to aid an agent's navigation. There are three dominant ideas governing map generation: Roadmaps [ACF01] [PLT05] [AAC*07], Convex Polygons [KBT03] [LD04] and Uniform Grids [ST05] [TLC02].

© The Eurographics Association 2007.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

Whichever method is employed each approach typically commences from a two dimensional image describing the free space in an environment based upon the location of buildings on the terrain. Our paper focuses on the roadmap technique, which comprises of a graph of way-points connected with edges. The graph is suitable for efficiently determining paths through large environments and simplifies the virtual agent steering by providing obstacle free direct lines between the way-points. However, the roadmap technique has one common disadvantage, which is that only one path for virtual agents can be defined in large areas.

Several authors try to generate alternative paths for their virtual agents to get more realistic movement. Latif and Widyarto [LW04] vary the path for a group of characters by adding a secondary heuristic function to the A* path searching algorithm on a grid based map. However, this work is limited to a group of characters that have the same source and same destination and can not take advantage of road map techniques. Arikan et. al. [ACF01] used visibility which is based on obstacle corners for navigation. In [ACF01] different sides of the obstacles are used to vary the agent paths but since this technique relies on the presence of obstacles to vary the paths it lacks path alternatives in wide free spaces. Musse and Thalmann [MT01] used different Bézier curves to define multiple paths between two way-points for a group of agents. Each Bézier curve can be assigned to only one agent at a time and has to be calculated in real-time. Pettre and Thalmann [PT05] provides different paths for virtual agents by using Probabilistic Roadmaps (PRM). In [PT05] the roadmap is generated randomly (PRM) and a shortest path finding algorithm processes the PRM graph. Path variations are obtained by removing edges from the previously found path and recalculating the shortest path on the whole graph. However, the characteristics of the PRM causes many redundant edges that effects the system performance and also can only be used for groups of agents that have the same source and the same destination. Pettre et. al. [PLT05] calculated the Voronoi diagram of the environment to extract way-points (roadmaps). In their approach the clearance between way-points is represented using cylinders and gates are defined by intersecting those cylinders. They vary the paths by assigning different way-points, which lie on the gates, for every individual. However, it is not guaranteed to cover wide free spaces due to the graph structure.

3. Overview

To ensure that the proposed method is readily employed for a variety of applications, the algorithms developed require limited knowledge of the urban environment and process this information to populate the scene automatically. The knowledge concerning the urban environment is encoded into a free space map as illustrated in Figure 1. Figure 1 depicts a plan view of the entire scene with buildings in grey, roads in red and the free space in black. One of the significant problems faced when introducing virtual agents in an urban environment is concerned with controlling each agent's traversal through the free space. This traversal should be free from collision with both static and dynamic objects, spend limited time on the roads and provide good coverage of the available free space.

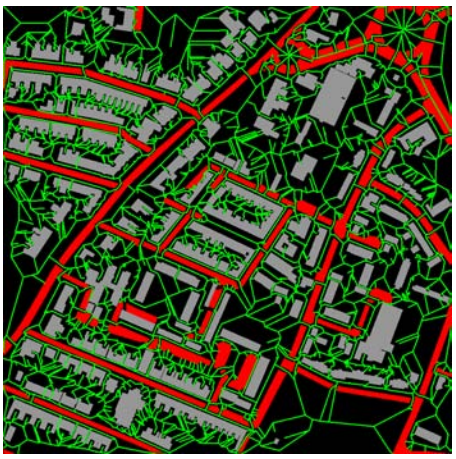

Figure 1: *Computation of the Straight skeleton: Buildings are in grey, roads in red, straight skeleton in green and free space in black.*

By computing the straight skeleton a set of potential paths through the environment is generated for the virtual agents. The straight skeleton may be computed for a polygon with holes and is formed by shrinking the polygon's boundary inwards and reducing the edge lengths as this shrinking process is performed. The vertices of the polygon boundary will move along the bisectors emanating from each vertex. As the polygon shrinks in size two events can occur. These are called the edge and split events and are defined by Aicholzer [AAAG95] as follows. An edge event occurs "when an edge shrinks to zero, making its neighbouring edges adjacent". The split event occurs when "a reflex vertex runs to an edge and splits it, thus splitting the whole polygon". When an event occurs a new bisector is constructed and the process continues. To alleviate the iterative shrinking polygon procedure the implementation as documented in [FO98] has been used. Figure 1 illustrates the straight skeleton using green line segments to represent the edges. The straight skeleton is an ideal structure for covering the space with the added benefit of requiring less vertices and edges than the Medial axis. Furthermore it provides reasonable paths for the agents in narrow sections of the free space map, such as between building and road polygons. However, in larger open spaces a single route through the space is not sufficient for realistic behaviour, resulting in agents bunching together forming noticeable corridors.

4. Iterative Shrinking Polygons

Whilst extending path following algorithms to permit agents to deviate (by a user specified amount) from the straight skeleton can lead to a reduction in this bunching effect, it

© The Eurographics Association 2007.

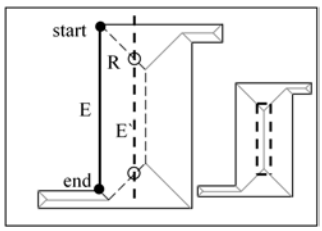

Figure 2: *Left: For each polygon edge an associated polygon, R, is computed. The supporting edge, E, is moved inwards to E' and clipped against R. Right: illustrates the additional edges created in the open space.*

will not completely reduce it. In the open spaces of the free space map the graph resulting from the straight skeleton calculation should be augmented with additional edges. These edges will aid in offering alternatives to the virtual agents. The introduction of these edges is achieved using a technique called Iterative Shrinking Polygons.

In Section 3 we discussed how the straight skeleton could be computed by iteratively scaling the polygon boundary inwards. The scaling of the polygon could be stopped at any time to yield a new set of edges which could be added to the straight skeleton graph for path planning. Simply scaling the polygon is not sufficient, since one of two edge events can occur causing the polygon to either reduce in its number of vertices or to split into two new polygons. Therefore the algorithm proceeds as follows. First each polygon edge, E, is processed to generate an associated polygon denoted by R. R is formed by combining the polygon edge with a subset of the edges, L, from the straight skeleton graph. L are all those edges from the straight skeleton graph which form a connected set of line segments from one end of the polygon's edge, E, to the other. These are illustrated using a thin dashed line in Figure 2. L is constructed by performing a boundary walk from the polygon edge around the skeleton edges taking the edge which makes the least interior angle to the last selected edge. As the polygon is scaled each edge moves inwards and its intersection with the neighbouring polygon edges will lie on the edges of R. Consequently to obtain a new edge, a line collinear to the polygon edge is moved inwards and it is subsequently clipped against the edges of R. If the edge does not intersect with any edge of R then it is discarded. Figure 2 illustrates this procedure on a small non-intersecting polygon.

The iterative shrinking approach has been applied to the road and building polygons illustrated in Figure 1. By shrinking the polygons inwards by an increasing distance from their original position a number of new edge sets are created and these are illustrated in Figure 3 using yellow line segments. All edges in the graph are labelled with an integer indicating their level, with edges on the straight skeleton being assigned level zero and other iteratively generated edges being assigned increasing labels (multi-level edges).

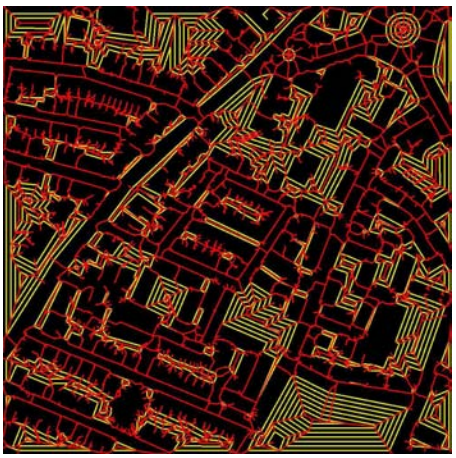

Figure 3: *Iterative Shrinking Polygons: Straight skeleton in red, free space in black and iterative shrinking polygons in yellow.*

To complete the graph, the edges of the straight skeleton are split at the point of intersection with a newly created edge to ensure all edges are connected. The resulting graph is passed to the behaviour module where each vertex represents a way point in the roadmap technique.

5. Using Path Variations

Initially every virtual agent is placed on a random way point and is assigned a random destination. Paths on the way-points graph are calculated in real time by using Dijkstra's shortest path algorithm (DSP). Two versions of DSP are used: one is for global path calculation and the other is for sub-path calculation. The global path is calculated by using only the original straight skeleton edges that are so called level zero edges. The multi-level edges are used only when a virtual agent reaches large free spaces. When a pedestrian reaches a way point that connects to one or more higher level edges it is possible for the agent to take several alternative sub-paths before continuing to their final destination. The source and destination of the sub-path are found by observing the first transition from a level zero edge to a level one edge and the following way point which lies on a level one edge respectively. Therefore, the source of the sub-path is the beginning point of the large area and the destination is the end of the large area, as is illustrated in Figure 4.

The sub-path calculation takes into account the edge population. Edge population is a dynamic variable of an edge and it equals the square root of the number of pedestrians currently on that edge. The square root is for normalizing the population effect. Finally the edge density that the sub-path finder is going to use is calculated by multiplying the edge population and edge distance. Edge distance is the scalar edge distance in the way-points graph and the global path finder directly uses it as an edge weight. By using edge density in the sub-path finder, pedestrians are forced to take al-

© The Eurographics Association 2007.

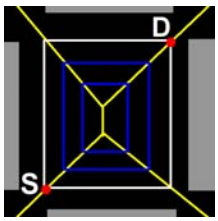

Figure 4: *S and D represent the partial source and destination way-points, the straight skeleton edges are in yellow, 1^{st} level edges in white, 2^{nd} and 3^{rd} level edges in blue.*

ternative paths depending on how crowded a particular path is.

Finally the resulting sub-path is inserted into the global path and the virtual pedestrians continue to follow their way-points lists sequentially. By using the techniques explained above, the overhead of including additional multi-level edges in path finding is reduced whilst forcing virtual agents to use alternative paths.

6. Results

The system has been tested on a $500 \times 500 m^2$ area consisting of 174,690 triangles with 1000 pedestrians on a computer with an Intel Pentium D 2.8GHz processor, 1GB RAM and an ATI 9700 graphics card. There are 5387 level zero edges and 1908 higher level edges found on the tested system during straight skeleton calculation. The behaviour system is tested on the render engine described by Ryder and Day in [RD06]. Pedestrian steering between waypoints is implemented using Reynold's steering engine, [Rey87]. First the system was tested without path variations (only zero level edges) and the system performance was measured at around 35 frames per second. Then the system was tested by including all path variations and sub-path calculations (multi level edges) and there was no significant frame rate drop observed (only 4-5fps) thanks to the partial path calculation.

7. Conclusions

In this paper an algorithm is proposed for handling large free spaces in a virtual crowd simulation. By using this method more realistic looking crowd behaviours can be easily observed, especially in large areas, as illustrated in Figure 5 and the accompanying video. However, because of the greatly increased number of edges the path finding computation cost could be increased. By dividing the path finding problem into two parts: global path finding and sub-path finding for large areas, the additional computational cost is reduced.

References

[AAAG95] AICHOLZER O., AURENHAMMER F., ALBERTS D., GARTNER B.: A novel type of skeleton for polygons. *Journal of Universal Computer Science* (1995).

[AAC*07] AVNEESH S., ANDERSEN E., CURTIS S., LIN M., MANOCHA D.: Real-time path planning for virtual agents in dynamic environments. In *IEEE Virtual Reality 2007 (to appear)* (2007).

[ACF01] ARIKAN O., CHENNEY S., FORSYTH D. A.: Efficient multi-agent path planning. In *Eurographics'01 Workshop on Animation and Simulation* (2001).

[FO98] FELKEL P., OBDRŽÁLEK S.: Straight skeleton implementation. In *(SCCG'98)* (1998), pp. 210–218.

[KBT03] KALLMANN M., BIERI H., THALMANN D.: Fully dynamic constrained delaunay triangulations. In *Geometric Modelling for Scientific Visualization* (2003).

[LD04] LAMARCHE F., DONIKIAN S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *CGF 23*, 3 (2004), 509–518.

[LW04] LATIF M. S. A., WIDYARTO S.: The crowd simulation for interactive virtual environments. In *SIGGRAPH* (Singapore, 2004).

[MT01] MUSSE S. R., THALMANN D.: Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Vis. and Comp. Grap. 7* (2001), 152–164.

[PLT05] PETTRE J., LAUMOND J., THALMANN D.: A navigation graph for real-time crowd animation on multi-layered and uneven terrain. In *(V-CROWDS'05)* (2005).

[PT05] PETTRE J., THALMANN D.: Path planning for crowds: From shared goals to individual behaviors. In *EUROGRAPHICS Short Papers* (2005).

[RD06] RYDER G., DAY A. M.: High quality shadows for real-time crowds. In *EG Short Papers* (2006), pp. 37–41.

[Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH* (1987), vol. 21, pp. 25–34.

[ST05] SHAO W., TERZOPOULOS D.: Environmental modeling for autonomous virtual pedestrians. In *SAE Symposium on Digital Human Modeling for Design and Engineering* (2005).

[TLC02] TECCHIA F., LOSCOS C., CHRYSANTHOU Y.: Visualizing crowds in real-time. *CGF 21*, 4 (2002), 753–765.

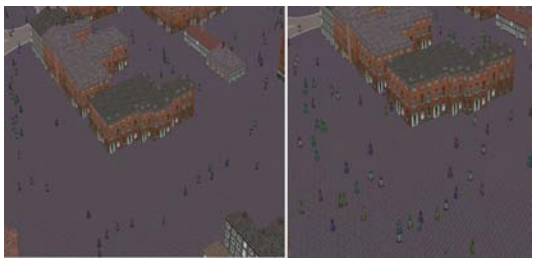

Figure 5: *Comparison of the behaviour simulation using the straight skeleton edges (left) and including multi level edges (right)*

© The Eurographics Association 2007.