Chapter 18
Agent Tools, Techniques and Methods for Macro and Microscopic Simulation

Ateen Patel and Andrew Hudson-Smith

Abstract Many situations exist that require virtual crowds to be modelled via computer simulations on varying scales. Such simulations often have conflicting goals; the need for large and complex worlds with rich behaviours in agents, but at the same time, the need for fast performance provided by simpler agents with reasonable crowd authoring. Our goal in this chapter is to establish the tools and techniques required for simulating large-scale virtual crowds. We identify both macroscopic and microscopic simulation methods and detail application areas where there is the need for navigation and behaviour of agents around the simulation environment, to correspond to realism. Hence, we actively identify different classes of applications that form balances between the conflicting goals that exist in simulating virtual populations.

18.1 Introduction

Different applications have specific needs, and therefore, dedicated techniques to solve navigation and behavioural problems. Applications for these macroscopic and microscopic simulations can be described in three broad classes: safety and urban planning, entertainment, including the video games and the movie industry, and lastly virtual reality.

Safety and urban planning models require simulations that correspond to realism related to calibrated environments. The navigation of the crowds and their behaviour has to correspond to reality. The general objective of these types of simulations are to model the flow of pedestrians around a building, e.g. when trying to exit a building or fire, to test the suitability of the design of a building, or as an aid to large

A. Patel (✉) • A. Hudson-Smith
Centre for Advanced Spatial Analysis (CASA), University College London, Westminster, UK
e-mail: ateen.patel@ucl.ac.uk; asmith@geog.ucl.ac.uk

scale planning, such as shopping centres, stadiums or town planning. Therefore, the simulation has to correspond to real world conditions; however interactivity and real time simulation is not a requirement. The data are generally analysed during a post processing phase.

Entertainment applications, such as video games or movies, require the behaviour and navigation of agents to appear real, but exact realism is not necessary. Interactivity is a crucial requirement for video games. Examples include recent games that provide an open world, such as Crysis 2 for the Xbox 360,[1] where players inhabit a virtual city. The realism here is enhanced by simulations of computer controlled agents with subsets reacting to real time human input.

While safety and urban simulations require exact simulations, and entertainment applications require interactivity, virtual reality simulations require both interactivity and believability. Virtual reality models can also enhance urban planning models by adding a level of immersion that gives the realism required to aid town planning. Here, the immersion of a user into the virtual world requires pedestrians moving around the scene to look and behave in a natural and real manner. A number of resources are used in a virtual reality application, from the rendering of the scene in the virtual world, to the pedestrians walking around the city during the crowd simulation. In order to solve the navigation problem, the global movement of a pedestrian around the virtual world is generally planned (see Sect. 18.2). We will explore the meaning of planned movement in the following sections. In addition to global planned behaviour, local behaviour of pedestrians interacting at street level must be taken into account. The local behaviour is commonly reactive, taking into account the interactions of pedestrians with the environment and other pedestrians (see Sect. 18.3).

The section that follows will survey some of the relevant models for pedestrian simulations to give a broad idea of techniques used and the tools that exist as well as alternatives to these tools. Further surveys can be found in Pelechano et al. (2008) and Thalmann and Musse (2007).

18.2 Macroscopic Agent Behaviour

In order to simulate crowds, individuals need to behave realistically at a local level as well as at a global level when navigating the environment. Therefore, coordinating the movements of these individual agents plays an important role. When simulating large and complex environments, local behavioural techniques such as the social forces model (Helbing and Molnár 1995; Helbing et al. 2000) and the boids model (Reynolds 1987) can get caught up in local minima. Several techniques have been proposed in order to deal with the crowd navigation problem at a global scale. The techniques can be divided into three major classes: navigation graphs (Pettre

[1] http://www.crytek.com

et al. 2005; Pelechano and Badler 2006; Lerner et al. 2006), potential fields (Loscos et al. 2003; Chenney 2004; Treuille et al. 2006), and roadmaps (Bayazit et al. 2002; Sung et al. 2005; Sud et al. 2007). This section will describe the techniques for each class of navigation.

18.2.1 Navigation Graphs

Graphs were first introduced by Teller (1992), with the aim to aid architectural walk-throughs in real time. Graphs are made up of a series of nodes and portals. The agents need to navigate through these nodes and portals defined by the graph to reach a destination. Nodes may represent rooms or corridors indoors and pavements outdoors, and portals may represent doors internally and crossings outdoors.

In Pettre et al. (2005), navigation graphs use a cell decomposition technique represented as a data structure that captures the topological structure of the navigable space in the environment. The navigable space is divided into a set of intersecting circles. Edges are defined as the line segments that join the intersecting circles, which divide the adjacent navigable areas. Navigation graphs only have to be computed once for each environment. Navigation for agents from a source to a destination is performed by scattering the number of people navigating from a source A to a destination B by *x*% using Dijkstra's graph search algorithm. Alternative paths for pedestrians to follow are then computed depending on the scattering parameter *x*. This scattering provides for variety, saving computation time during simulation. Collision avoidance is, therefore, disabled for agents at far distances, as collisions are not detectable by a viewer.

In Pelechano and Badler (2006), Helbing and Molnár's (1995) social forces model is used for the local behaviour, and combined with a wayfinding approach to build up a high level global view of the environment. This global view is constructed via agents exploring the environment using current knowledge. Wayfinding is computed by using four components: building a mental model of the environment, its current position within the mental model, processes that help it learn features such as doors and walls within the environment, and finally, the navigation process that allows it to move through the environment. The mental map is made up of a cell and portal graph with nodes added as the agent navigates the environment. The algorithm calculates the shortest path to the exit based on the agent's knowledge of the environment. When agents interact, they can share information to the adjacent cells only, to share knowledge such as whether it is passable or there is danger through the portal (e.g. a door).

Agents have different levels of knowledge to conform to real world situations within three classes: agents who are leaders and trained, i.e. they have complete knowledge of the environment; those that do not have complete environmental knowledge and make their own decisions in stressful environments such as a fire evacuation; and agents who are not leaders and untrained, i.e. they do not have complete environmental knowledge and may be incapable of making their own decisions in a highly stressful environment. High level wayfinding then takes place. Leaders share their mental map with other agents. Agents then check their shortest known path based on their own and shared knowledge, reacting differently to a hazard based on the agent class.

In Lerner et al. (2006), outdoor environments are represented using cells and portals. The cells and portals are created by a two-pass algorithm; the first pass creates the initial partition, which is refined by a second pass. It proves more efficient than traditional Binary Space Partitioning (BSP), and creates an efficient automatic cell and portal partitioning; as such it can deal with arbitrary orientation of walls. Half edges, i.e. a single sided oriented edge obtained from a 3D model, are passed to the algorithm in 2D, which will create the partitioning. Pedestrian paths are represented as cells, whereas the intersections between pedestrian paths and crossings are represented by portals. The average number of visible cells and portals in a rendered scene is significantly lower than a BSP tree, which allows for more agents to be accommodated in real time.

18.2.2 Potential Fields

In methods that involve potential fields, the environment is modelled as a 2D discretised grid. Each cell in the grid has a certain potential. The goal has attractive potentials while the obstacles have repulsive potentials. A gradient can therefore be applied that guides the agents from the source to the destination. These methods help reduce the complexity at a macroscopic level. In Thompson and Marchant (1994), Simulex used potential fields for the agents to navigate around the scene.

Loscos et al. (2003) also uses a potential field method whereby the space is subdivided into a 2D grid to indicate the areas that are accessible such as crossings and pathways, and collision detection with buildings is achieved with a collision map (Tecchia and Chrysanthou 2000). The agents are not entirely represented as flows, but have individual-based behaviour. A graph of goals is around the urban environment, covering each pavement area. Individual behaviour is achieved by creating a trajectory, where the agents start at one of the goals in the graph, and are assigned another goal as a destination. The flows are achieved by using local information rather than global, making use of the graph of goals. When an agent reaches a goal, it stores a direction vector that fades over time. When another agent reaches that goal, it checks for a previous direction vector, and if found, makes a decision based on that vector. These methods have the potential of getting stuck in local minima. Figure 18.1 shows a representation of the agents moving around in the environment.

Another similar technique to potential fields is flow tiles (Chenney 2004). Each flow tile contains a precomputed velocity field. These flow tiles can be combined to form a continuous flow for various applications, where the edges and corners maintain continuity. For a crowd simulation, the streets can be represented with flow tiles (Fig. 18.2). Internal boundary conditions on the tiles can denote buildings preventing the agents bumping or walking through buildings.

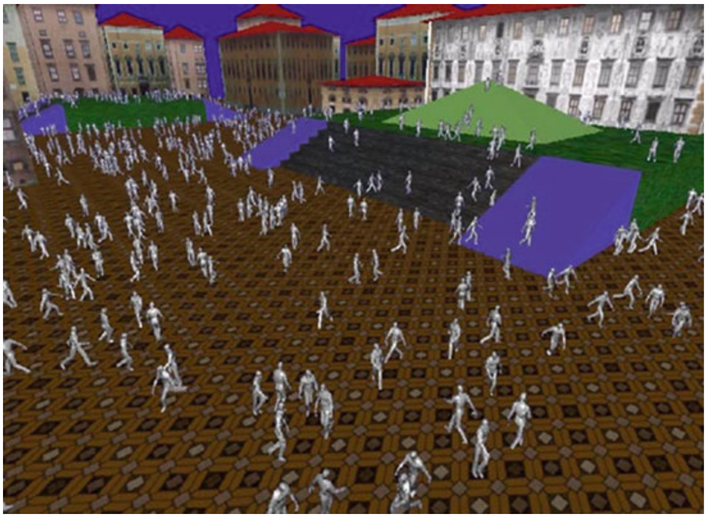


Fig. 18.1 Behaviours achieved by agents based on underlying grids containing behavioural maps (Taken from Tecchia and Chyrsanthou (2000))

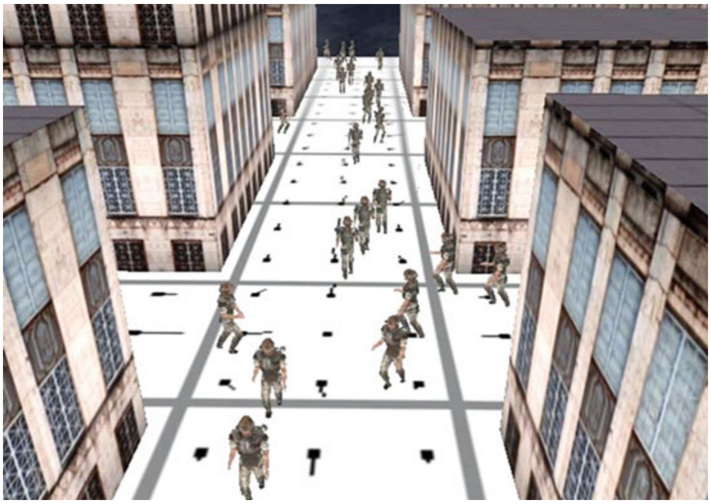


Fig. 18.2 Flow tiles on the streets drive the flow of agents through the city (Taken from Chenney (2004))

In Treuille et al. (2006), the model uses a continuum-based approach. The environment is modelled as a 2D grid including the mapping of uneven terrain. Large numbers of agents can be accommodated as the computations are based spatially rather than per-each individual agent. The motion of crowds is controlled by a dynamic potential field, which allows it to avoid moving obstacles without the need

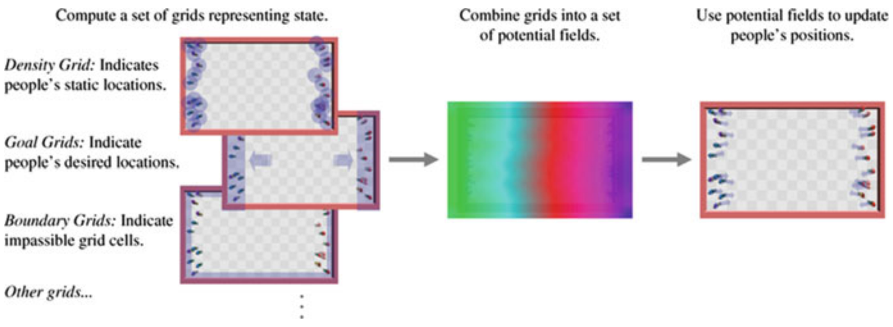

Compute a set of grids representing state.
Density Grid: Indicates people's static locations.
Goal Grids: Indicate people's desired locations.
Boundary Grids: Indicate impassible grid cells.
Other grids...
Combine grids into a set of potential fields.
Use potential fields to update people's positions.

Fig. 18.3 An underlying dynamic potential field representing density, goals and boundaries. Other conditions control the positions of the agents (Taken from Treuille et al. (2006))

for explicit collision avoidance. The potential field is composed of superimposing grids made up of different types of information, such as the density grid (the location of people), the goal grid (their goals), boundary grids (impassible grid cells), etc. (Fig. 18.3). The environment uses a cost function that determines the speed of the agents. The cost function takes in factors such as the slope of the terrain, the density of the agents, obstacles, etc. This cost function helps compute the optimal path for the agents to travel through by using the gradient of the computed potential field. Most crowd simulations combine global path planning and local behaviour planning methods with the potential of conflicts between the two methods. The continuum dynamics method integrates local path planning with the global navigation for the agents. A dynamic potential field integrates agents as moving obstacles, and therefore, a dynamic global path can be planned that will avoid static obstacles as well as other agents. As the path is dynamic and is updated based on the current environment, agents do not get stuck in local minima.

18.2.3 Road Maps

Road maps use robotics as a basis, as well as algorithms for route planning of agents around an urban environment developed from the motion planning solutions of robots. The real time global navigation of only a small number of agents will run efficiently when computing road maps based solely on the robot motion planning solution. When the number of agents increases, global collision-free routes have to be computed for each independent agent, which can result in exponential computation time, therefore becoming a bottleneck. Dynamic objects can cause an even greater problem for motion planning with the potential of agents getting stuck in local minima. Various solutions have been developed from road maps that will allow large numbers of agents to navigate around scenes that contain static as well as dynamic objects.

Bayazit et al. (2002) incorporates Reynold's seminal work on flocking (Reynolds 1987) with probabilistic roadmaps (PRM) to allow global navigation of agents.

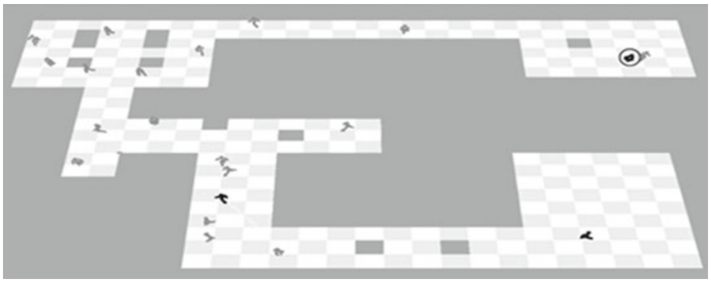


Fig. 18.4 The black character uses roadmaps to navigate through the rooms avoiding obstacles to lift a box (*circled*) in the *upper right* room (Taken from Sung et al. (2005))

Flocking behaviour uses local behaviour planning, where each flock member's behaviour is determined by its local environment. Integrating a roadmap based path planning method with the flocking behaviour allows complex group behaviours among the flock such as homing towards a goal, exploring an environment and shepherding. The roadmaps that are computed have maximum clearance from obstacles, allowing the flock to pass through a collision-free path. Individual flock members may have sub-goals within the group, and on reaching the sub-goal, continues along the global path of the group.

Lien et al. (2005) extend the road map integration of Bayazit et al. (2002), specifically the shepherding behaviour of the flock when following a global path. Shepherding behaviour is a type of group behaviour where one or more group members shepherds or controls the motion of another group, i.e., the flock. Lien et al. (2005) introduce multiple shepherds that control the flock individually without communicating with each other, and still manage to efficiently control a flock. The environment, once divided into a roadmap, contains milestones that are the nodes of the roadmap. The shepherds attempt to steer the flocks towards the milestones in order to help the flock reach the goal. The roadmap here is dynamic.

Kamphuis and Overmars (2004) created roadmaps for groups to follow. A path is created for a single agent known as the backbone path. The clear area around the path is termed the corridor, in which other agents in the group must stay. The corridor's width and the extent to which other agents can move away from the main group is limited by using a social forces model. This ensures the group stays together, and allows agents in the group to catch up or even pass through obstacles from the same side.

Sung et al. (2005) used the concept of roadmaps to create a global motion planning solution. A probabilistic roadmap (PRM) was computed for the pedestrians to navigate through the complex environments to avoid collisions (Fig. 18.4). The PRM does not create an accurate navigation path, but provides an initial guess for the search that approximately satisfies the constraints. It then uses Dijkstra's search to further refine the search to allow for smooth adjustments to the position, and direction of the pedestrian, providing an accurate motion that exactly satisfies the constraints. All the possible sets of actions are represented in a motion graph, and the search algorithms just described are used to generate motion. The crowd motion is authored by taking into account constraints such as duration, position, orientation and body pose. This allows two characters to meet at particular places face-to-face at specified times, or even to go to certain events together.

Lau and Kuffner (2005) introduced a behavioural motion planning approach using a finite-state machine (FSM), which determines the motion of the pedestrians by avoiding collisions with other pedestrians as well as obstacles. This motion is determined during runtime via a planning algorithm that uses the FSM to create a global search to create the trajectory that the pedestrian has to follow. This technique was further extended by Lau and Kuffner (2006) to accommodate a larger number of pedestrians in the simulation. It also allowed for the motion planning technique to re-plan in real time, allowing for a more dynamic environment. The environment here is modelled as a 2D grid where the cells are determined to either contain an obstacle or not. A search tree is precomputed to represent all the behavioural states that are reachable based on the current state of the pedestrian. A search algorithm also computes a global path to the goal of each pedestrian by avoiding obstacles mapped onto the environment grid map. A path finding algorithm using reverse path lookup is then used to calculate the shortest path to the destination for the pedestrian to follow. Using the behavioural states that were determined using the precomputed search tree, the path calculated is able to handle the presence of certain obstacles and characters by making the pedestrians jump or pass under them. A sequence of behaviours is then converted to actual believable motion using a motion synthesis system blending frames at transition points.

Sud et al. (2007) introduced a technique called Adaptive Elastic Roadmaps (AeRO). The roadmap is based on a Voronoi diagram. It uses an approach that allows agents to have distinct goals and individual-based behaviour without getting stuck in local minima similar to Treuille et al. (2006). The road map algorithm adapts to dynamic environments and computes a dynamic global navigation path for the agents by taking into account moving obstacles and inter-agent collision avoidance. The global road map updates incrementally, which reduces the computation time and offsets the requirement of continuous updating by introducing a concept called *link bands*. Link bands are introduced at a local level to augment the global navigation. The link bands deform and reform the road map at a local level, and therefore, guide agents through a collision free path avoiding other agents and moving obstacles.

18.3 Microscopic Agent Behaviour

There are varying techniques used to solve the crowd motion planning problem at a local scale. The technique used depends on the application that needs to be solved, and the specific requirements based on the crowd density and the scale of the area in which it needs to be simulated. This section will focus on the crowd behaviour at a local scale, and will divide local navigation techniques based on the application, as the application's requirements define the technique.

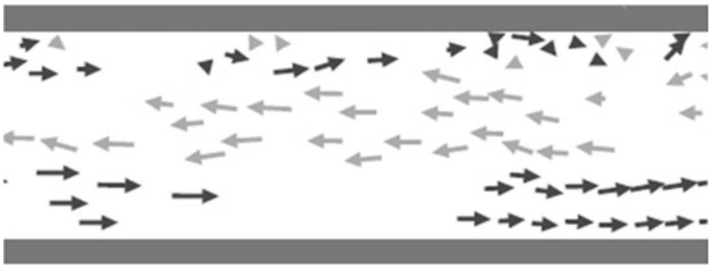


Fig. 18.5 The flow of agents leads to lane formation in this example. The direction and *arrow* lengths represent the walking direction and the speed of the agents (Taken from Helbing et al. (2001))

18.3.1 Safety, Urban Modelling and Architectural Applications

The analysis of video films of individual and groups of pedestrians has given a good insight into the way pedestrians behave, and it could give us an idea of how they influence crowds – see also Johnasson and Kretz (2012). There are certain observations made by Helbing et al. (2001) and Loscos et al. (2003). With regards to groups of pedestrians, they will walk at the same speed, follow the same goals, and will wait for each other if one goes missing. In order to simulate these pedestrians, the main goal of the simulator is to provide results that match real world data, e.g. evacuation times, walking speed, influence among individuals, individual personal space. Therefore, it is important to calibrate the models with real world data. There are three common approaches that are used to simulate pedestrians for safety applications. These can be classed as fluid, cellular automata (CA) and agent-based models (ABMs). Details of CA and ABMs can be found in Iltanen (2012) and Crooks and Heppenstall (2012), respectively.

Flow-based simulations are used to model pedestrians at a macroscopic scale, where an individual does not have a powerful influence on the behaviour of the crowd. These pedestrians are, therefore, not individual entities but are part of a network of flows. These flows are based on the Navier-Stokes equations, where Henderson's (1971) early work showed that it was possible to model pedestrians based on these equations. The Navier-Stokes equations describe the motion of fluid substances based on Newton's second law. These equations describe compressible and incompressible fluids, but to apply it to pedestrians, only the incompressible part of the equations need to be used. Helbing et al. (2001) and Hughes (2003) are examples of recent work that has used fluids to simulate crowds. Figure 18.5 illustrates pedestrian flow represented by arrows from Helbing et al. (2001).

CA approaches include the environment that is divided into a group of cells. Each cell has a finite amount of states. The pedestrians occupy the cells, and they move from cell to cell based on parameters such as crowd density and walking speed. ABMs are quite popular among safety sciences, social sciences and the urban modelling area. An agent as defined by Jennings et al. (1998) is "a computer

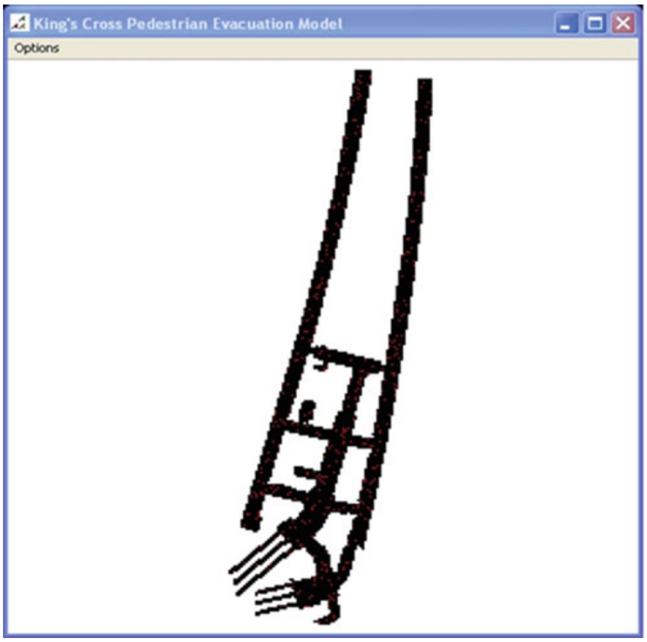


Fig. 18.6 Kings Cross St. Pancras underground station evacuation model (Taken from Castle (2006))

system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objective". These agents can also represent pedestrians at an individual level, unlike flows. This allows us to set micro-specifications of a crowd system, which generate emergent behaviours at a macro-scale. Castle's (2006) work is one example of an ABM for evaluating evacuation scenarios at King's Cross St Pancras underground station in London (Fig. 18.6). Batty et al.'s (2003) simulation of the Notting Hill carnival is another example of the use of an ABM in this area.

Helbing and Molnár (1995) introduce a 'social forces model' for pedestrians. They describe a model where a sum of the forces exists that influences the locomotion of pedestrians. There are three forces. The first is acceleration since the velocity of the pedestrian can vary over time. The second is repulsion, which is the repulsive force experienced from other pedestrians and obstacles, and the third is attraction, which is the attractive force experienced by other people (e.g. friends, family) or other objects (e.g. popular attractions, window displays). These forces are combined with a term to account for behavioural fluctuations to form the equation for the 'social forces model'. This model is one of the most popular approaches to pedestrian modelling in safety applications, and has been reused and refined in many other approaches such as Helbing et al. (2001) to simulate escape panic, and others used for modeling evacuation scenarios (Braun et al. 2003; Cordeiro et al. 2005).

18.3.2 Entertainment Applications

The aim of crowd simulation in entertainment applications is interactivity, especially in video games. Interactivity is also needed in the movie industry, but applied in a different context to video games. Whereas video games require real-time interaction with crowds for the end user, the movie industry handles all the crowd simulation offline in order to create impressive scenes containing huge crowds. The user interacts with the simulator running several trials and set-ups to create the final scene. This also gives the user global and local control, allowing precise control over the crowds' movements. Although the simulation is computed offline, it has to be fast enough for the several trials that will take place.

Work by Reynolds (1987) first introduced artificial life to the computer animation/graphics area. He demonstrated how the behaviour of bird flocks could emerge from an extension of a particle system, where the birds act as the particles. This was a computer model of coordinated animal motion such as bird flocks or fish schools, which accounted for collision avoidance and goal seeking (Fig. 18.7). This behaviour can be applied equally to crowds, although commercially, this model has had various applications. One such application is the bat swarms and penguin flocks in the 1992 movie 'Batman Returns'. More recent work by Reynolds (1999) extended the behaviours that were simulated, and introduced steering behaviours in addition to the collision avoidance and goal seeking introduced in Reynolds (1987). The steering behaviours involved seeking, fleeing, pursuit, evading, path following, wall following, etc. The crowd simulation has been further extended, and moved onto the Sony PlayStation 3 platform (Reynolds 2006). The general navigation and behaviour remains the same, but accommodates more agents in real time (Fig. 18.8).

Particle systems have been used in other ways to simulate crowds (Fig. 18.8). Bouvier et al. (1997) uses a particle system combined with Newton's law of motion to model crowds. This again uses a microscopic approach, which allows for the navigation to be planned at a local level. The local planning is based on taking into account reactive behaviour, where individuals avoid each other based on density as well as the avoidance of obstacles. The model by Bouvier et al. (1997) also uses higher level decision making to create the simulation, taking into account the parameters such as destination, density, the duration of stay for visiting certain places and

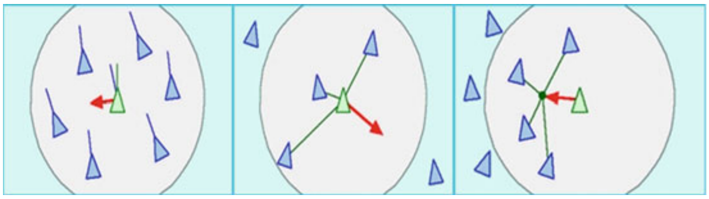


Fig. 18.7 Animated flock showing behaviours of separation (steering to keep a minimum distance among flockmates), alignment (steering to keep aligned with the direction of local flockmates) and cohesion (steering to move closer to flockmates) (Taken from Reynolds (1999))

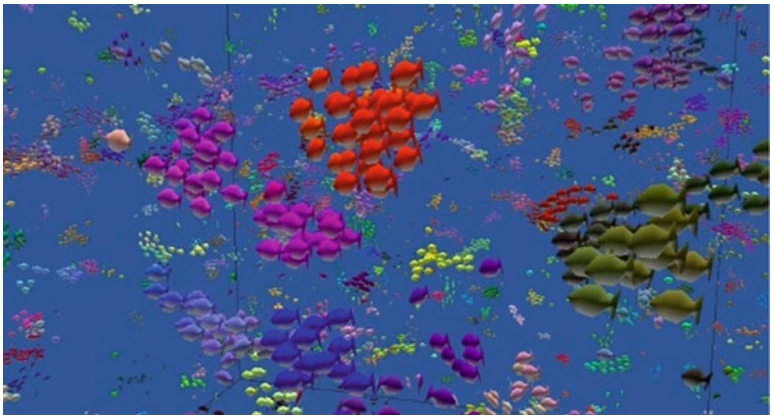


Fig. 18.8 Flock of 10,000 fish rendered on a Sony PlayStation 3 (Taken from Reynolds (2006))

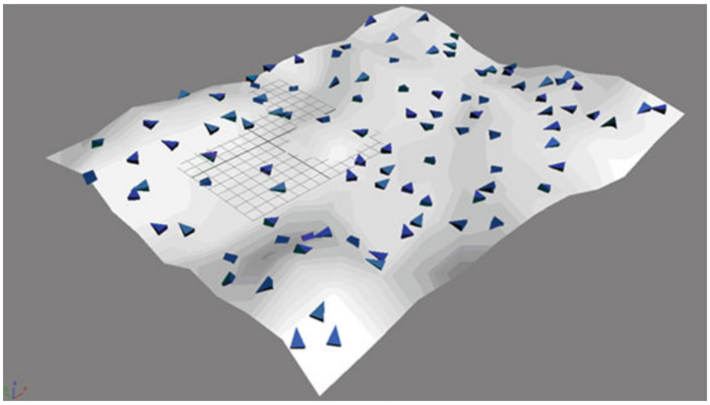


Fig. 18.9 3ds Max crowd and delegate system: follow surface and wander behavior

other global events such as taking into account the presence of obstacles. This higher level is based on Newton's law.

Models created for the entertainment industry are often created in packages outside of the custom created research toolkits. While not being a specialised modelling system per se, 3ds Max from Autodesk is an industry leader in game production linking into the wider Autodesk suites. Built into the package is a 'Crowd and Delegate' system allowing groups of 3D objects and characters to be animated according to a series of computed behaviours. Using simple rules such as 'avoid' 'follow', 'seek' or custom written in the form of scripts, one can create crowds with highly complex behaviours. Figure 18.9 illustrates the most basic level of model behaviour in 3ds Max using the 'follow surface' and 'wander behaviour'. Each delegate is represented by the triangles forming part of a team assigned with a wander rule for a set period of time while following the 3D object surface. Once set up, key frames are assigned according to a predefined time sequence, and the simulation is pre-computed allowing for later export or rendering. Using such a method, it can be quickly extended to cityscapes, fictional landscapes or any realm.

Models created in such a software not only utilize advances in graphic card technology, but also advances in physics-based engines such as Havok Physics, which allows the user to easily add additional elements such as mass and gravity to influence the behavior of the agents.

The various built-in components of 3ds Max enables high quality graphic outputs as well as real-time previews and outputs to game engines such as Crysis. This allows researchers to achieve ‘semi movie-like’ results. Indeed 3ds Max is arguably the most powerful mass market simulation engine although it does need viewing in context. The package itself is essentially a ‘black box’ compared to more traditional approaches with routes and behaviours pre-computed with minimal feedback on the underlying dynamics. We would argue, however, that it is within the entertainment industry with increasing consumer demand for realistic reaction from agents within gaming environments where future innovation will emerge.

18.3.3 Virtual Reality Applications

The techniques used for virtual reality applications combine the techniques used in the previous two sections. This is an area where we see more of a convergence between the two broad areas of crowd simulation, i.e. realism of crowd behaviour, and high quality visualisation. This convergence helps to immerse the user into the virtual crowd.

Tecchia et al. (2001) introduced a layered approach to simulate agent behaviour. The approach is based on using a 2D grid with four layers. Two layers are used for collision detection with the environment, and between other agents. The other two layers are used for other complex behaviours, one layer which involves behaviours such as waiting, turning a certain direction or computing a new direction based on the environment. The other layer deals with highly complex local behaviour such as pushing a button to call a lift (Fig. 18.10). This layered approach allows for highly realistic behaviour to be simulated at a real-time rate, with thousands of agents.

Musse and Thalmann (2001) introduced scalability in simulating virtual crowds. The crowd simulation used an agent-based approach. The hierarchy was comprised of virtual crowds, groups and individuals where the groups are the more intelligent and complex structures. The individual follows the specification of the group. The group can be controlled at different levels of the autonomy, which refers to the independence of the agents without the need for user intervention including the amount of information needed for simulating the crowds. There are three different levels of control of the groups. The first is through the use of scripted behaviour for the crowds to follow, the second is by defining behavioural rules to user events and reactions to create more complex behaviours, and the third is by guiding crowds to follow orders by the user during run-time. Individual agents, therefore, group together based on having common sociological factors defined for the group.

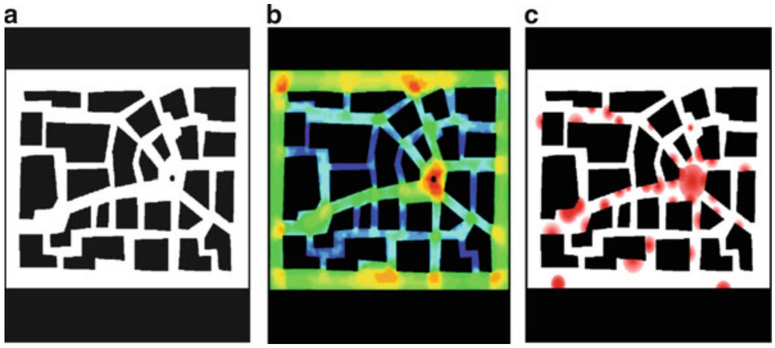


Fig. 18.10 (**a**) An example of a collision map. The regions where agents can move are encoded in *white* and inaccessible regions in *black*. Examples of behaviour maps: (**b**) visibility map (**c**) attraction map (Taken from Tecchia et al. (2001))

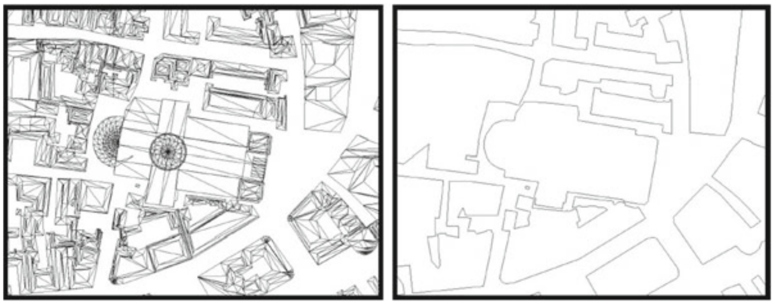


Fig. 18.11 A 2D map extracted from a 3D database (Taken from Lamarche and Donikian (2004))

Lamarche and Donikian (2004) introduce a different type of hierarchical approach using the geometry of the virtual environment. The topological structure of the geometry enables the capability for global path planning in real-time for each agent while taking into account visibility and reactive behaviours. To create the topological structure, the 3D geometry in the database is converted into a 2D map (Fig. 18.11). This is done by using two parallel planes separated by the height of a humanoid, and cutting the database geometry starting at the floor. This area cut is the navigation area, which is projected onto an XY plane to create the 2D map. A cell decomposition technique is then used on the 2D map, and the resulting triangulation is then optimised to create convex cells of the environment, which is then captured in a graph.

The cell decomposition enables the ability to generate road maps for path planning. The topological abstraction of the geometry reduces the path planning graph size, enabling real time path planning computations. While agents are navigating the path, a reactive technique is used to take into account the presence of other agents in each cell. An iterative optimisation approach is used to reach the goal and

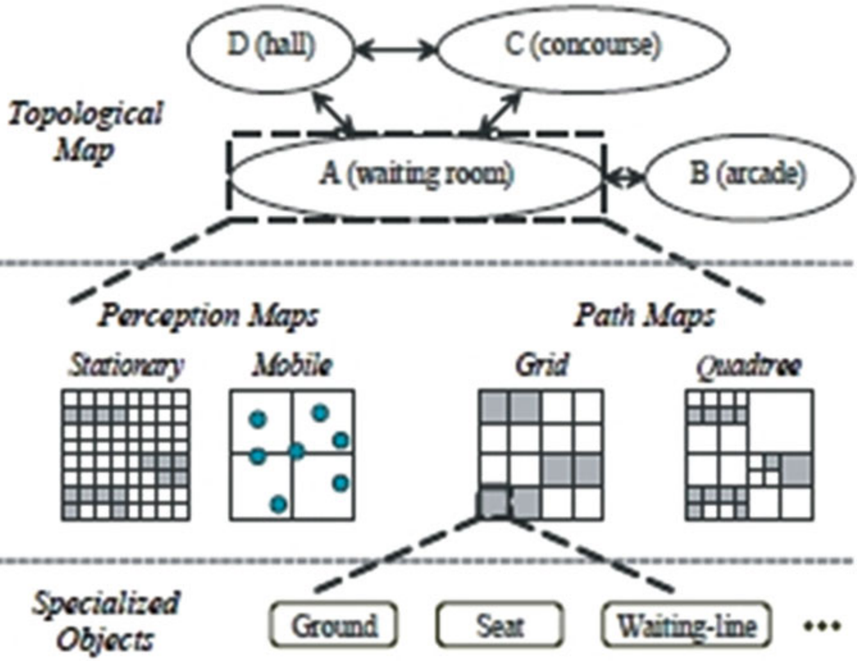


Fig. 18.12 Hierarchical collection of maps representing the environment (Taken from Shao and Terzopoulos (2005))

avoid collisions. The way the trajectory of the agent is computed depends on the pedestrian viewing angle and the next cell. A free personal space is kept around each agent, which is a minimum distance between other agents, and obstacles. By predicting the positions of the other agents, the model can predict if a collision may occur, and in turn modify its trajectory.

Shao and Terzopoulos (2005) use a conceptually similar hierarchical structure for the virtual environment to Lamarche and Donikian (2004), but with different methods. Their method uses a hierarchical collection of maps (Fig. 18.12). At the highest level, it includes a topological map that represents the structure of the different parts of the virtual world. This topological structure results in a graph structure that represents the environment as a set of rooms or corridors that are interconnected. At the middle level, perception maps provide information related to perceptual queries by storing lists of stationary objects as well as agents in the vicinity. This level also includes grid maps and quadtree maps for the geometry of the environment. The lowest level consists of path maps.

These maps enable path planning online for navigating through the environment. The level also includes special representations such as seats and waiting lines. Pedestrians in this model use a reactive and deliberative approach. The reactive behaviours with which the pedestrians are equipped are primitive and based on Tu

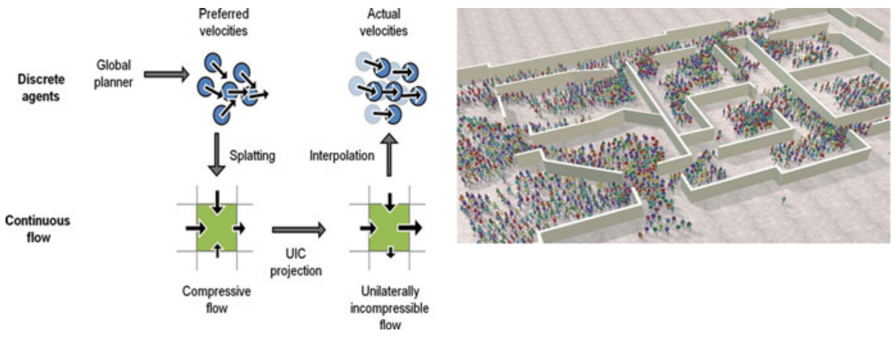


Fig. 18.13 Overview of the algorithm (*left*) with a screenshot of the simulation (*right*) (Taken from Narain et al. (2009))

and Terzopoulos (1994) using a bottom-up strategy. The behaviours include standing still and moving forward as well as collision avoidance. These local behaviours in turn support more complex motivational behaviours. The motivational and navigational behaviour enable the pedestrians to reach their goal. The topological map introduced here allows for the achievement of this task. The graph structure is used to find the global path for the pedestrian, which is then refined using the grid maps.

In Narain et al. (2009), a dual representation approach is used, where agents are represented as both discrete, and as part of a continuous system which is described by density and flow velocity. Therefore, the discrete representation of agents is combined with the representation of the crowd as a continuum fluid. Local collision avoidance is mapped into the continuous domain. A Unilateral Incompressibility Constraint (UIC) is introduced in this mapping to obtain a variational constraint on the crowd flow depending on the region of the environment. The constraint accelerates the collision avoidance step, as it increases the density constraint to maximum where obstacles exist.

Figure 18.13 illustrates the UIC algorithm as well as a rendered scene of an evacuation. This approach can be used as a local planner along with a global planning approach such as a road-map based algorithm (Sud et al. 2007) or continuous based optimisation (Treuille et al. 2006) on a coarse grid. The global planning approach needs to be computed to determine a preferred velocity for each agent. The global path is computed by avoiding large static obstacles in the environment, and ignoring the presence of agents. The UIC solver introduced is then used for collision avoidance among agents, and therefore, computes the corrected velocity field for the crowds, taking agents into account. Using the corrected crowd velocity field, the agent flow is constrained in high density regions, but the agents flow will be over-constrained in low density region. In order to overcome this problem, the velocity of the agent is computed by interpolating between the continuum velocity and the agents own preferred velocity. A final step is then added to create a personal space around each agent by introducing a minimum distance based on the method used in (Treuille et al. 2006).

18.4 Tools

18.4.1 Safety and Urban Planning

A number of commercial tools exist for creating pedestrian simulators for safety applications, particularly for evacuation scenarios. We will cover the most popular tools, and we will try to describe the technique used behind each tool where possible. Due to the commercial nature of some of the tools, the information in the literature is gathered from publicly available data.

Legion™ is a popular pedestrian simulator used in the commercial sector, which resulted from extensive research at the Maia Institute in Monaco. It uses ABMs for all of its simulations, with the environment layouts based on computer aided design (CAD). Therefore, agents are treated as individuals. Legion™ is based on work carried out by Still (2000), trying to simulate and analyse crowd dynamics in environments such as stadiums. Environments in the simulation are known as *iSpace*, where agents and the environment communicate with each other. Agents inform the environment with their observations such as obstacles, while it asks the environment for the direction to take based on its objective. In this way, the agents can communicate using the environment. The path planning algorithm has not been disclosed, but the agent uses a least effort algorithm to reach its goal, based on satisfying a number of constraints, such as speed distribution and collision (Thalmann et al. 2005). The least effort means the maximum speed, but a minimum cost, time and congestion, and therefore, optimality may not be guaranteed. The path costs are based on their length, travel time and effort. A number of parameters in Legion™ for the agents give the simulations a factor of believability, e.g. they may move around randomly in a particular space, and come together to congregate before moving apart again. Other random events can be entered into the simulation such as size, speed and age.

As mentioned in the previous sections, flow-based models are based on a macro-scale, where the individual is not significant, and pedestrians are part of a network of flows. EVACNET4 (Kisko et al. 1998) is an example of a flow-based model used to simulate evacuation scenarios. The environment of the simulator is represented as a network of nodes. This network is called the EVACNET network model, and consists of a set of nodes and arcs. The nodes are the building components or the bounded parts of the environment, such as rooms, corridors, lobbies and halls. The arcs are the passageways that connect these parts. Each arc has to be set up with a flow capacity and a traversal time by the user. The user also has to set the initial placement of occupants, i.e. each node has an initial state, such as the number of people, and the destination (fire exits). The maximum number of people for each node must also be defined. The program then searches for a minimum evacuation time, and the network flow algorithm finds a solution for the evacuation. Note that the model does not have the environment described in explicit geometry, and it is rather an implicit model using the nodes. There are other examples of flow-based models such as EESCAPE and FIREWIND.

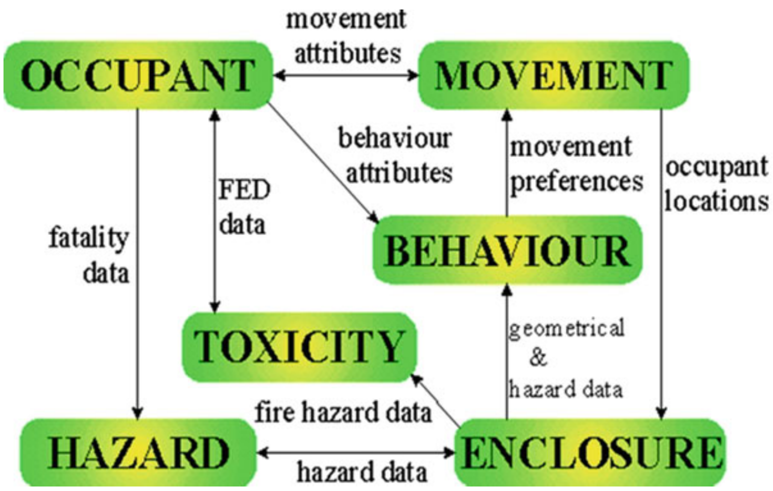


Fig. 18.14 The submodels representing the algorithm of the EXODUS model

Simulex (Thompson and Marchant 1994) is another agent-based simulator. Like Legion™, Simulex uses CAD files, but uses these files to define 2D grids which represent floor plans. It uses distance maps on top of the 2D grid, which is a discrete vector field. The aim of Simulex is to model evacuation scenarios, and individual agents have various attributes such as walking speed, acceleration, position, and angle of orientation. Agents use the gradient of the vector field to reach their goal, while the attributes help solve the motion planning problem.

EXODUS is a simulation tool created by the Fire Safety Engineering Group of the University of Greenwich.[2] This tool is also an agent-based simulator. It is used for both evacuation scenarios and pedestrian dynamics, and was first introduced in Galea et al. (1993). It is made up of five interactive submodels, namely, occupant, movement, behaviour, toxicity and hazard (Fig. 18.14). The navigation of the occupants uses a rule-based system and is adaptive. CAD files are used again to define 2D grids for the floor plans as well as the general geometric structure. Each grid is made up of nodes and arcs, but unlike EVACNET4, these nodes represent a small region of space on the grid, while the arcs represent the distance between nodes. Individual agents travel using the arcs from node to node to reach their goal. This model incorporates many psychological factors for the simulation.

In addition to creating the first prototype for Legion™, Still (2000) developed Myriad II along with colleagues at Crowd Dynamics Ltd.[3] Myriad II is unique compared to other tools seen thus far, where it integrates three different techniques to create the modelling tool (Fig. 18.15). It comprises network analysis, spatial analysis, and agent-based analysis. This integration enables it to create the best possible model based on the situation in the environment, e.g. a network model representing

[2] http://fseg.gre.ac.uk/exodus/

[3] http://www.crowddynamics.co.uk/

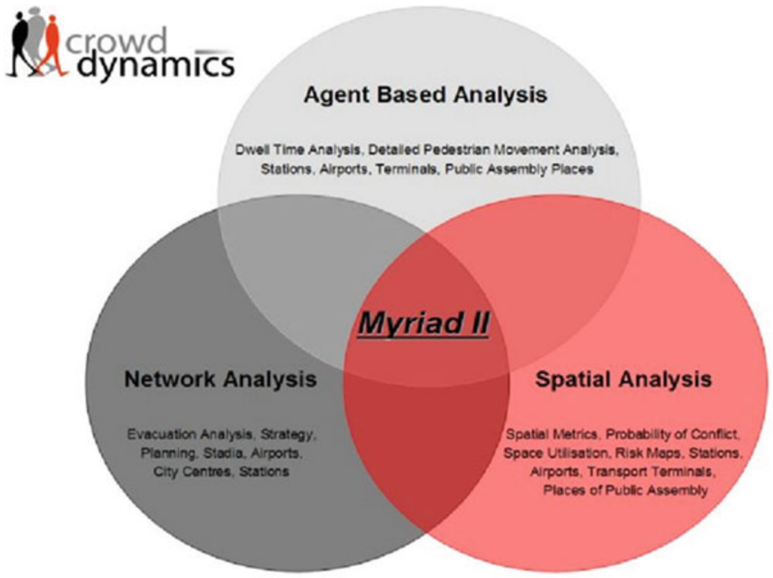

crowd dynamics
Agent Based Analysis
Dwell Time Analysis, Detailed Pedestrian Movement Analysis, Stations, Airports, Terminals, Public Assembly Places
Myriad II
Network Analysis
Evacuation Analysis, Strategy, Planning, Stadia, Airports, City Centres, Stations
Spatial Analysis
Spatial Metrics, Probability of Conflict, Space Utilisation, Risk Maps, Stations, Airports, Transport Terminals, Places of Public Assembly

Fig. 18.15 The overlapping of different methods in Myriad II

simple roads can be integrated into an ABM representing complex interactions in parts of the environment such as a shopping centre. Data can then be passed between the two models of the environment. Four attributes define the crowd behaviour, namely, objective, motility (walking speed), constraint (eg. crowd density) and assimilation (acceleration) (Still 2000).

Mass Motion (Challenger et al. 2009) is a 3D agent-based simulation tool developed by Erin Morrow at Arup.[4] The ABM's simulation core runs on a 64 bit multi-core architecture, and it may be the only currently available commercial simulation software that is multi-threaded. Each individual agent has a position, orientation and velocity. Each agent has a goal, and has to reach it in the minimum amount of time by using both local and global navigation. Figure 18.16 shows a crowd simulation using the Mass Motion software.

Each individual agent is aware of the environment using a 2D projection of all static obstacles within a defined volume in order to map the obstacles (Fig. 18.17). A modified version of Dijkstra's algorithm is used to define the complete paths between origin and destination within the map. Each agent also accounts for the visibility of other agents within their immediate neighbourhood.

Figure 18.17 shows the structure of a simulation environment within Mass Motion via a sparse node network and its implicit relationships between floors via links. The distance for each agent to the exit is precomputed using the modified Dijkstra's algorithm and is stored at link nodes. This helps in considering the best possible route to the destination (exit) via a perceived cost, which is randomised per

[4] http://www.arup.com/

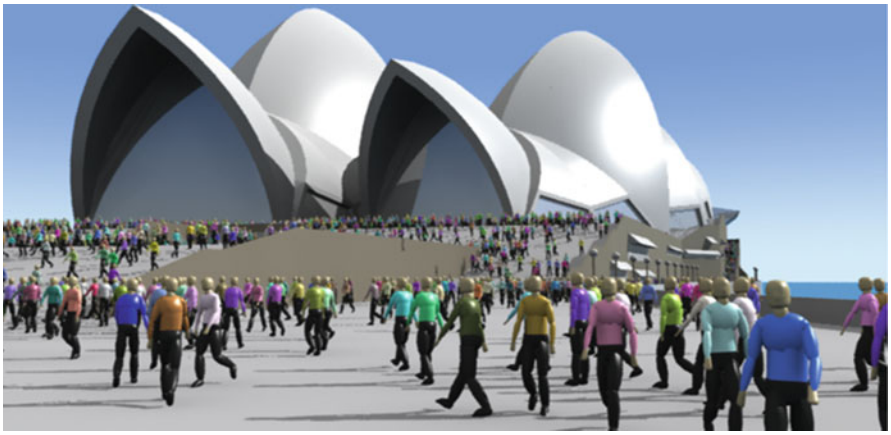


Fig. 18.16 Screenshot of Mass Motion simulation software (Transbay terminal in San Franciso) (Image courtesy of Erin Morrow, Arup)

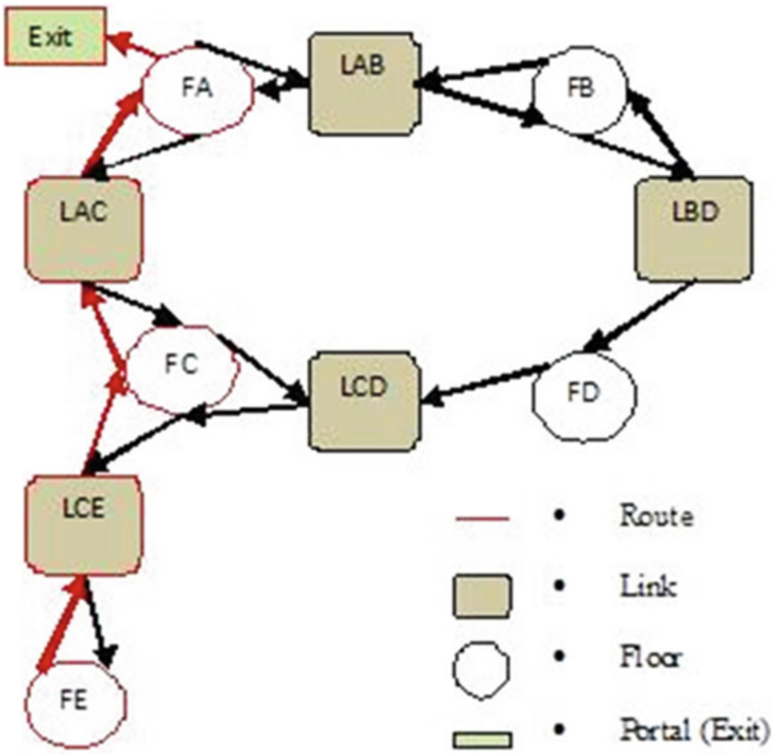


Fig. 18.17 Sparse node network structure of a simulation environment within Mass Motion (Image courtesy of Erin Morrow, Arup)

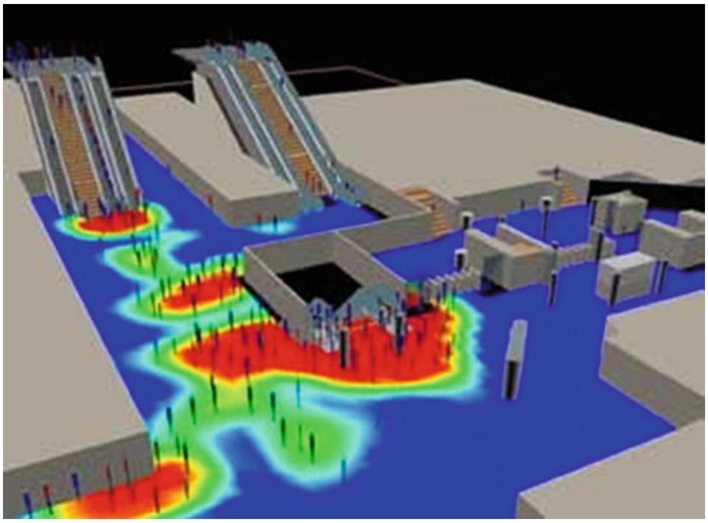


Fig. 18.18 Screenshot of the STEPS software

agent using randomised weights for the cost components of the routes. A simplified algorithm for the total route cost is given as:

$$cost = W_D * \left(\frac{D_G}{V}\right) + W_Q * Q + W_L * L \tag{18.1}$$

where W_D is the distance weight; D_G is the total distance from agent position to goal; V is the agent's desired velocity; W_Q is the queue weight, Q is the expected time in the queue before reaching the link entrance; W_L is the link traversal weight; and L is the link type cost (level, ramp, stair, etc.). The variables W_D, V, W_Q and W_L are random agent properties. The structure shown in Fig. 18.17 has the advantage of allowing the environmental geometry to be easily replaced, where the sparse node network will then update itself based on the new geometric relationships.

STEPS[5] is another agent-based simulation tool developed by Mott MacDonald and uses coarse grid geometry. Agents do not have a global view of the environment but move around towards a final exit. STEPS takes in CAD files for modelling geometry. It uses a CA where each individual occupies one cell at a given time. Agents know the shortest route but not the density or obstacles that they may encounter. Each agent then moves in the desired direction if the next cell is empty. The routes that agents take can generally be defined by a crude form of global view, mainly for normal operation, i.e. not during evacuation scenarios. Costs can be accounted for by the route taken. Each agent also has its own characteristics and familiarity behaviour. Figure 18.18 shows a screenshot of the software.

[5] http://www.mottmac.com/skillsandservices/software/stepssoftware/

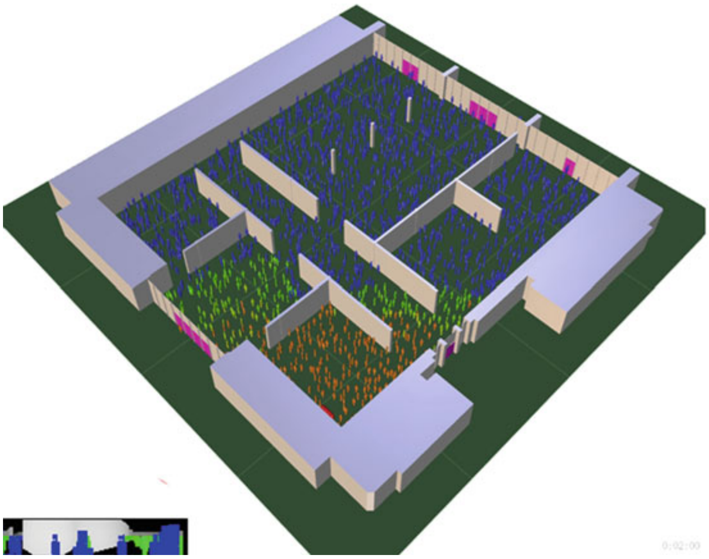


Fig. 18.19 Screenshot of the MASSIVE software

18.4.2 Entertainment

As the entertainment field is based on interactivity and real time simulation where computer games are concerned, a number of tools exist to facilitate the creation of interactive environments for a cinematographer or the user.

MASSIVE (Multiple Agent Simulation System in Virtual Environment) is one of the most advanced crowd animation and simulation systems for off-line productions (Fig. 18.19). It was first developed by Stephen Regelous in order to address the animation problems for the Lord of the Rings[6] trilogy, and is now used in many visual effects systems. The software does not contain any explicit Artificial Intelligence (AI), the agents can be assigned AI, and it uses a simple way to deploy intelligent agents (Thalmann et al. 2005). It does not use global path planning, and the user can author the entire crowd motion although autonomous agents can be introduced. It uses a hierarchical system where behavioural systems vary (Pettre et al. 2006), and at the lowest level, techniques such as potential fields can control the flow of the crowds (Thalmann and Musse 2007). MASSIVE uses a bottom-up approach, which leads to emergent behaviour based on reactive motion planning, with high user control.

The user specifies inputs such as vision and internal states. Through the use of fuzzy logic, agents then respond to the environment after taking into account the static and dynamic obstacles (such as other agents). In this way agents can endlessly walk

[6] http://www.lordoftherings.net/

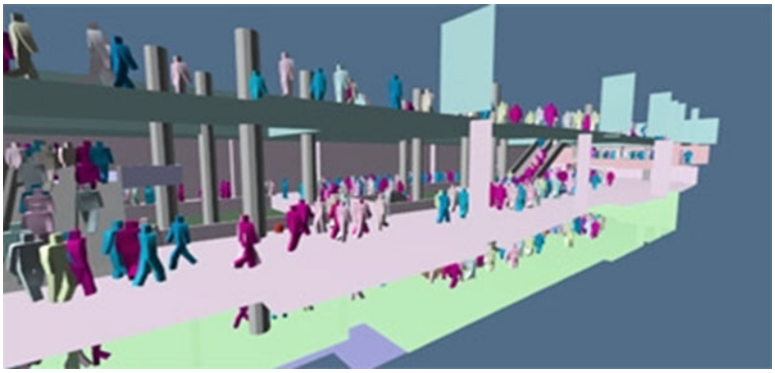


Fig. 18.20 Screenshot of the Spirops Crowd software

through the environment without the need for elaborate path planning. The realisation of long term goals of the agents in this case is not important, as cinematic shots average at around 5 s, and therefore, the main aim is high quality visualisation during the flow. The design decision in this case allows for a very large number of agents (hundreds of thousands) to be accommodated during the simulation. MASSIVE Insight[7] is a spin-off from MASSIVE, and is at a beta stage that is aimed for safety, architecture and urban applications. It uses the original bottom-up approach of MASSIVE and a global planning algorithm will be incorporated in the system.

There are a number of middleware tools available to author crowds for video games. Spirops[8] is a middleware tool for game development, which focusses on AI problems in the game development cycle (see Fig. 18.20). This allows the in-game characters to behave realistically. Spirops crowd is a component of Spirops focused on crowd simulation. It creates paths for the pedestrians to follow and avoid collisions. It was first developed by Axel Buendia based on his research thesis. Due to its commercial nature, not many details are available with regards to the path planning of pedestrians. However, it currently uses a hard linked behaviour to plan the global path although a future release will include dynamic planning. A screenshot is provided in Fig. 18.20.

PathEngine[9] is another toolkit to provide realistic movements of agents in virtual worlds such as games. The toolkit provides collision avoidance and paths for agents to follow. However, path finding is based on visibility of agents, and it does not provide a global path planning solution.

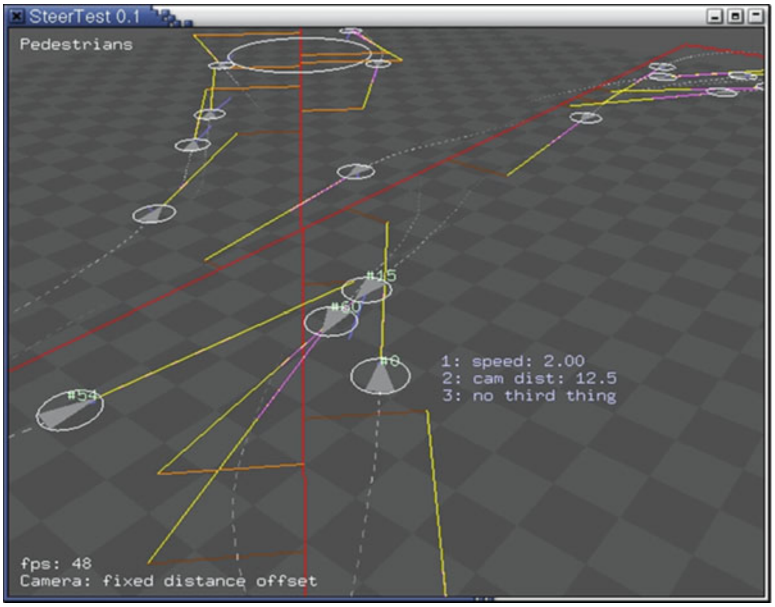
OpenSteer[10] has been developed by Reynolds based on his boids work (Reynolds 1987, 1999). It is an open source library written in C++ and uses OpenGL, which is available to build the steering behaviours of autonomous agents. It is a cross

[7] http://www.massivesoftware.com/real-world-simulation/

[8] http://www.spirops.com/

[9] http://www.pathengine.com/

[10] http://opensteer.sourceforge.net/doc.html.



```
1: speed: 2.00
2: cam dist: 12.5
3: no third thing
fps: 48
Camera: fixed distance offset
```

Fig. 18.21 OpenSteer demo window

platform toolkit available as a demo which demonstrates a number of steering behaviours supplied as sample plug-ins (Fig. 18.21). Bespoke plug-ins can be written by using the steering library provided, which defines the behaviours for the autonomous agents such as wander, goal seeking, flee, path following as well as collision avoidance. The individual steering behaviours as described in Reynolds (1999) combine to form a more complex behaviour. There are two ways it happens: either by switching between individual behaviours as the environment changes, or by combining the individual behaviours together, therefore, working in parallel with each other.

18.4.3 Virtual Reality

As this section deals with the convergence between two fields, not many tools exist. Research in academia has focused on the realistic behaviour of individuals combined with rendering huge numbers (hundreds of thousands) on desktop computers. Crowd simulation systems in this area have mainly been developed in the academic field. A few tools mainly relevant to this section will be described in greater detail.

ViCrowd is an academic tool developed by Musse (2000) and detailed in Musse and Thalmann (2001). It includes the three-tier hierarchy for the different degrees of autonomy, and uses Reynold's flocking model for behavioural rules. A screenshot

Fig. 18.22 Screenshot of ViCrowd (Taken from Musse and Thalmann (2001))

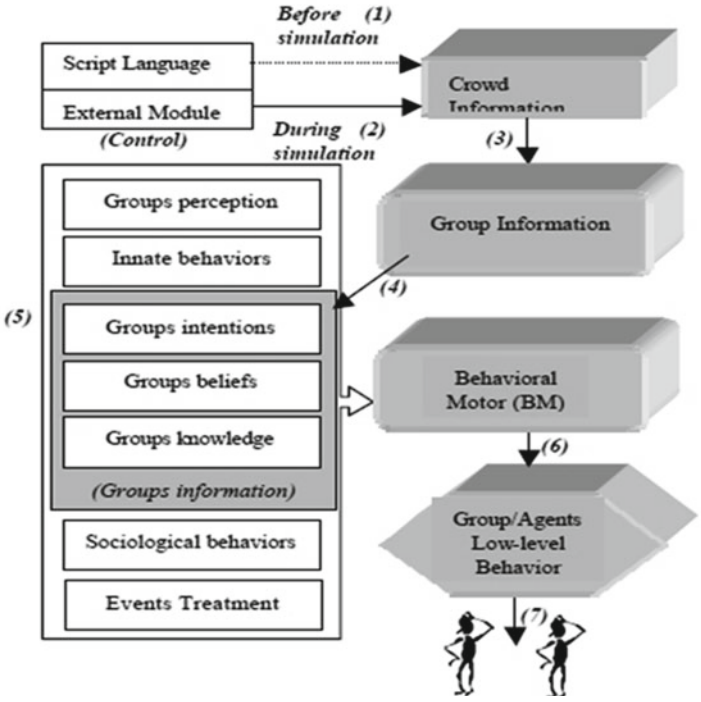


Fig. 18.23 Diagram representing the behavioural algorithm (Taken from Musse and Thalmann (2001))

of ViCrowd can be seen in Fig. 18.22. The architecture for the tool is shown in Fig. 18.23. A behaviour motor exists to process various low level behaviours such as perception and other sociological behaviours. The architecture of ViCrowd is described below, and the steps are indicated in Fig. 18.23.

18.4.3.1 Procedure

Step 1: Scripted behaviours are specified before the simulation takes place

Step 2: The user may also provide behaviours for the crowd to follow during the simulation.

Step 3: The behaviours indicated in the last two steps, as well as information emerging using behavioural rules, are then distributed among the groups.

Step 4: The group information that is passed describes the groups' goals, intentions and knowledge.

Step 5: This information is then combined with other information such as the events and other sociological behaviours. These sociological behaviours include beliefs describing the internal state of the groups, and knowledge representing the environmental information.

Step 6: The behavioural motor represents the process where the low level behaviour of the groups and individuals is generated.

Step 7: The low level behaviour of the groups is then created such as goals, walking speed and actions.

The behaviour motor's process uses a number of priority rules for the behaviours. It first analyses the group information where the low level behaviours for each group are created. If sociological factors exist, the low level behaviours can be changed again by getting the agents to change groups. If certain events occur, the reaction defined by the user or scripted behaviour can also change the group's low level behaviour. This low level behaviour is then passed to the individual agents of the group, where perception and innate abilities are used to satisfy the group's intentions. This procedure described allows for complex behaviours of crowds to be simulated in real time.

Crowdbrush is another tool that comes out of the same research lab as ViCrowd. This tool was designed and developed by Ulicny et al. (2004) to make the authoring of complex crowd scenes with different scenarios simpler. It introduces the capability to create, modify and control the crowd member of a scene in real time with visual feedback. A brush metaphor allows for crowd authoring. The main focus of the tool is to author crowd scenes, which is beyond the scope of this section. Crowdbrush does allow the creation of pedestrian paths, but is done manually, and no global motion planning algorithms are involved. Low level behaviours are defined using a rule-based behaviour engine (Ulicny and Thalmann 2002). A simple reactive rule system is defined in order to achieve fast real-time simulation of thousands of agents. The agents use simple displacement of humans when reacting to any type of internal or external event defined by the behavioural rules. A simple collision avoidance system also exists based on Helbing and Molnár's (1995) social forces model. Collision queries are minimised by using a bin-lattice space subdivision (Reynolds 2000). The behaviour can be applied both directly and indirectly in real-time using the same brush metaphor used for authoring crowd scenes by sending events to activate behavioural rules or tagging agents with a tagging brush that selectively triggers different behavioural rules for different agents. This allows for various actions happening in different parts of the crowd scene.

18.5 Summary

As demonstrated through the three application areas within this chapter, there are multiple micro and macro simulation techniques dependent on the environment to be modelled and the industry involved. In-house or third party toolkits are common place, each with their own pros and cons and methods to achieve the desired simulation. Packages such as 3ds Max are available free of charge to the global academic community, yet their use is limited outside of the entertainment field. As the simulation industry widens in scope and moves towards more real time inputs and outputs rather than pre-computed ones, we expect this to change. Simulation is increasingly moving towards a real-time input/output environment as processing speeds and data collection techniques advance.

References

Batty, M., Desyllas, J., & Duxbury, E. (2003). The discrete dynamics of small-scale spatial events: Agent-based models of mobility in carnivals and street parades. *International Journal of GIS, 17*(7), 673–697.

Bayazit, O. B., Lien, J.-M., & Amato, N. M. (2002). Roadmap-based flocking for complex environments. In *Proceedings of the 10th Pacific conference on computer graphics and applications* (p. 104). New York: IEEE Computer Society.

Bouvier, E., Cohen, E., & Najman, L. (1997). From crowd simulation to airbag deployment: Particle systems, a new paradigm of simulation. *Journal of Electronic Imaging, 6*(1), 94–107.

Braun, A., Musse, S. R., de Oliviera, L. P. L., & Bodmann, B. E. J. (2003). Modeling individual behaviors in crowd simulation. *Proceedings of the 16th International Conference on Computer Animation and Social Agents*. Los Alamitos: IEEE Computer Society. http://doi.ieeecomputersociety.org/10.1109/CASA.2003.1199317

Castle, C. J. E. (2006). *Developing a prototype agent-based pedestrian evacuation model to explore the evacuation of King's Cross St Pancras underground station*. London: Centre for Advanced Spatial Analysis (UCL). http://eprints.ucl.ac.uk/3327/. Sept 2006.

Challenger, W., Clegg W. C., & Robinson A. M. (2009). *Understanding crowd behaviours: Guidance and lessons identified*. Technical Report prepared for UK Cabinet Office, Emergency Planning College, University of Leeds, 2009.

Chenney, S. (2004). Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 233–242). Grenoble: Eurographics Association.

Cordeiro, O. C., Braun, A., Silveria, C. B., Musse, S. R., & Cavalheiro, G. G. (2005). Concurrency on social forces simulation model. *First International Workshop on Crowd Simulation*. EPFL, Lausanne.

Crooks, A. T., & Heppenstall, A. J. (2012). Introduction to agent-based modelling. In A. J. Heppenstall, A. T. Crooks, L. M. See, & M. Batty (Eds.), *Agent-based models of geographical systems* (pp. 85–105). Dordrecht: Springer.

Galea, E. R., Galparsoro, J. M. P., & Pearce, J. (1993). A brief description of the EXODUS evacuation model. *Proceedings of the International Conference on Fire Safety*, *San Francisco*, Vol 18 (pp. 149–162).

Helbing, D., Farkas, I. J., & Vicsek, T. (2000). Simulating dynamical features of escape panic, Nature, 407, pp. 487–490.

Helbing, D., & Molnár, P. (1995). Social force model for pedestrian dynamics. *Physical Review E, 51*(5), 4282–4286.

Helbing, D., Molnár, P., Farkas, I. J., & Bolay, K. (2001). Self-organizing pedestrian movement. *Environment and Planning B: Planning and Design, 28*(3), 361–383.

Henderson, L. F. (1971). The statistics of crowd fluids. *Nature, 229*(5284), 381–383.

Hughes, R. L. (2003). The flow of human crowds. *Annual Review of Fluid Mechanics, 35*(1), 169–182.

Iltanen, S. (2012). Cellular automata in urban spatial modelling. In A. J. Heppenstall, A. T. Crooks, L. M. See, & M. Batty (Eds.), *Agent-based models of geographical systems* (pp. 69–84). Dordrecht: Springer.

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. http://eprints.ecs.soton.ac.uk/2112/

Johnasson, A., & Kretz, T. (2012). Applied pedestrian modeling. In A. J. Heppenstall, A. T. Crooks, L. M. See, & M. Batty (Eds.), *Agent-based models of geographical systems* (pp. 451–462). Dordrecht: Springer.

Kamphuis, A., & Overmars, M. H. (2004). Finding paths for coherent groups using clearance. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 19–28). Grenoble: Eurographics Association.

Kisko, T. M., Francis, R. L., & Nobel, C. R. (1998). *Evacnet4 user's guide*. University of Florida.

Lamarche, F., & Donikian, S. (2004). Crowd of virtual humans: A new approach for real time navigation in complex and structured environments. *Computer Graphics Forum, 23*(3), 509–518.

Lau, M., & Kuffner, J. J. (2005). Behavior planning for character animation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 271–280). Los Angeles: ACM.

Lau, M., & Kuffner, J. J. (2006). Precomputed search trees: Planning for interactive goal-driven animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 299–308). Vienna: Eurographics Association.

Lerner, A., Chrysanthou, Y., & Cohen-Or, D. (2006). Efficient cells-and-portals partitioning: Research articles. *Computer Animation and Virtual Worlds, 17*(1), 21–40.

Lien, J- M., Rodriguez, S., Malric, J., & Amato, N. M. (2005). Shepherding behaviors with multiple shepherds. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* (pp. 3402–3407).

Loscos, C., Marchal, D., & Meyer, A. (2003). Intuitive crowd behaviour in dense urban environments using local laws. In *Theory and practice of computer graphics (TPCG'03)* (pp. 122–129). Los Alamitos: IEEE Computer Society.

Musse, S. R. (2000). *Human crowd modelling with various levels of behaviour control*. Unpublished PhD thesis, EPFL.

Musse, S. R., & Thalmann, D. (2001). Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions Visualization and Computer Graphics, 7*(2), 152–164.

Narain, R., Golas, A., Curtis, S., & Lin, M. C. (2009). Aggregate dynamics for dense crowd simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics symposium on computer animation*. New York: ACM.

Pelechano, N., Allbeck, J. M., & Badler, N. I. (2008). Virtual crowds: Methods, simulation and control. *Synthesis Lectures on Computer Graphics and Animation, 3*(1), 1–176.

Pelechano, N., & Badler, N. I. (2006). Modeling crowd and trained leader behavior during building evacuation. *IEEE Computer Graphics and Applications, 26*(6), 80–86.

Pettre, J., Ciechomski, P. H., Maim, J., Yersin, B., Laumond, J.-P., & Thalmann, D. (2006). Real-time navigating crowds: Scalable simulation and rendering. *Computer Animation and Virtual Worlds, 17*(3–4), 445–455.

Pettre, J., Laumond, J., & Thalmann, D. (2005). A navigation graph for real-time crowd animation on multilayered and uneven terrain. Presented at the *First International Workshop on Crowd Simulation (V-CROWDS'05),* Lausanne.

Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. *Computer Graphics, 21*(4), 25–34.

Reynolds, C. (1999). Steering behaviours for autonomous characters. *Proceedings of the 1999 Game Developers Conference, San Jose, California* (pp. 763–782). URL: http://www.red.com/cwr/steer/

Reynolds, C. (2000). Interaction with groups of autonomous characters. *Proceedings of the 2000 Game Developers Conference,* San Francisco.

Reynolds, C. (2006). Big fast crowds on PS3. In *Proceedings of the 2006 ACM SIGGRAPH symposium on videogames* (pp. 113–121). Boston: ACM.

Shao, W., & Terzopoulos, D. (2005). Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 19–28). Los Angeles: ACM.

Still, G. (2000). *Crowd dynamics*. Unpublished PhD thesis, Warwick University, Coventry.

Sud, A., Gayle, R., Andersen, E., Guy, S., Lin, M., & Manocha, D. (2007). Real-time navigation of independent agents using adaptive roadmaps. In *Proceedings of the 2007 ACM symposium on virtual reality software and technology* (pp. 99–106). Newport Beach: ACM.

Sung, M., Kovar, L., & Gleicher, M. (2005). Fast and accurate goal-directed motion synthesis for crowds. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 291–300). Los Angeles: ACM.

Tecchia, F., & Chrysanthou, Y. (2000). Real-time rendering of densely populated urban environments. In *Proceedings of the eurographics workshop on rendering techniques 2000* (pp. 83–88). New York: Springer.

Tecchia, F., Loscos, C., Conroy-Dalton, R., & Chrysanthou, Y. L. (2001). Agent behaviour simulator (ABS): A platform for urban behaviour development. In *Proceedings of the first international game technology conference and idea expo (GTEC'01)*. Hong Kong: ACM.

Teller, S. J. (1992). *Visibility computations in densely occluded polyhedral environments*. Unpublished PhD thesis, University of California, Berkeley.

Thalmann, D., Kermel, L., Opdyke, W., & Regelous, S. (2005). Crowd and group animation. In *ACM SIGGRAPH 2005 courses* (p. 1). Los Angeles: ACM.

Thalmann, D., & Musse, S. R. (2007). *Crowd simulation*. London: Springer (Online service, and Adriana Braun).

Thompson, P. A., & Marchant, E. W. (1994). *Simulex; developing new computer modelling techniques for evaluation*. Unit of Fire Safety Engineering, University of Edinburgh.

Treuille, A., Cooper, S., & Popovic, Z. (2006). Continuum crowds. *ACM Transaction on Graph, 25*(3), 1160–1168.

Tu, X., & Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on computer graphics and interactive techniques* (pp. 43–50). New York: ACM.

Ulicny, B., Ciechomski, P., & Thalmann, D. (2004). Crowdbrush: Interactive authoring of real-time crowd scenes. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 243–252). Grenoble: Eurographics Association.

Ulicny, B., & Thalmann, D. (2002). Towards interactive real-time crowd behavior simulation. *Computer Graphics Forum, 21*(4), 767–775.