*To be published in the Proceediings of the COSIT'03 Conference*
*Ittingen (Switzerland) September 2003, Kuhn, Timpf, Worboys (eds.)*
*Springer Verlag Lecture Notes in Computer Science*


# MAGS Project: Multi-Agent GeoSimulation and Crowd Simulation[1]

Bernard Moulin, Walid Chaker, Jimmy Perron, Patrick Pelletier, Jimmy Hogan, Edouard Gbei

Computer Science Department and Center for Research in Geomatics,
Laval University, Ste Foy, QC G1K 7P4, Canada
{bernard.moulin, walid.chaker, jimmy.perron, patrick.pelletier, edouard.gbei} @ift.ulaval.ca,

**Abstract**. Geosimulation aims at modeling systems at the scale of individuals and entity-level units of the built environment and provides a new way to simulate how geographic spaces can be used by their future users, particularly in urban environments. In the MAGS Project we are developing a generic software platform for the creation of Multi-Agent Geo-Simulations involving several thousand agents interacting in virtual geographic environments (in 2D and 3D) and endowed with spatial cognitive capabilities (perception, navigation, reasoning). Our approach is currently applied to the simulation of crowd behaviors in urban environments.

## 1 Introduction

Constrained by the structure of space and communication networks, urban regions form a complex system of interactions which evolves over time. Actors in the system strategically adapt their behaviors according to their own priorities among perceived opportunities. Government agencies need tools that take into account these complex interactions in order to compare planning scenarios on the basis of their global and long-term effects. However, statistical modeling remains unable to simulate these urban processes appropriately. The 'traditional' urban simulation models have been criticized [20] because of their centralized approach, a poor treatment of dynamics, a reduced flexibility and a lack of realism [31]. In addition, an increasing number of researchers think that an adequate forecast of transportation demand should be based on the study of individual mobility behaviors [6]. In such an approach, individual behaviors should be modeled at a spatio-temporal scale which is appropriate for characterizing the nature and importance of the decision processes influencing the transformation of the urban environment. However, applying such an approach involves the development of simulation systems that are able to deal with the simultaneous actions of thousands of actors, integrating their interactions [7], and taking into account the structural constraints

[1] The MAGS project is supported by GEOIDE, the Network of Centers of Excellence in Geomatics, Defense Research and Development-Valcartier, the Natural Science and Engineering Council of Canada and le Fonds Québécois de Recherche sur la nature et les technologies

of the urban environment (transportation network, localization of infrastructures, distribution of activities and services, etc).

Geo-simulation is an approach which aims at modeling systems at the scale of individuals and entity-level units of the built environment. It provides a new way to simulate how geographic spaces can be used by their future users, particularly in urban environments. Torrens [31] indicates that geo-simulation models are in their relative infancy as applied to urban simulation and constitute a new class of simulation models which borrow heavily from developments in geographic information science, artificial intelligence, artificial life, complexity studies and simulation in natural sciences and social sciences outside geography. Applied to urban design, geo-simulation provides the means to study the characteristics of the urban environment by analyzing the interactions of moving agents simulating the behavior of various actors (such as pedestrians and automobiles) in a urban landscape. In distributed artificial intelligence, researchers developed techniques that are used to create multi-agent systems (MASs) composed of agents which are autonomous programs collaborating together to solve problems [18] [32]. MASs are particularly adapted to the simulation of population dynamics in large-scale environments [22] [8] [11]. They are well suited to the exploration of dynamic phenomena in which the interactions of individual entities can be studied at a micro-level and the emergence of behavioral patterns can be observed at a macro-level [29] [27].

Microsimulation has been frequently used in the field of transportation systems analysis during the past decade, especially to model urban travel behaviors in order to predict the spatial and temporal distributions of trips in urban areas. For example, the TRANSIMS system [1], a generic platform for modeling and simulating complex behaviors using actors, provides a series of integrated transportation and air quality analysis models and attempts to simulate the aspects of human behavior that are relevant to transportation planning. Although traffic models often use a multi-actor approach, they typically do not contain models of cognitive aspects of human spatial behavior[2] [8] [31]. Indeed, most traffic models simulate urban phenomena [24] using a cellular automata approach [33] in which space is represented as a uniform lattice of cells. Each cell may be in a finite number of discrete states and can change its state at discrete time steps during the simulation. The cells are subject to a uniform set of rules which drive the overall behavior of the system. Such an approach does not enable actors to move in space autonomously and does not provide mechanisms to simulate basic spatial cognitive capabilities such as perception and memory. Introducing agent's autonomy and cognitive spatial capabilities in geo-simulation models would offer new possibilities for the analysis of phenomena resulting from the decisions and actions of a large number of actors resulting from their interactions with their spatial environment and other actors.

In the MAGS Project we aim at developing a generic software platform for the creation of Multi-Agent Geo-Simulations (MAGS) involving several thousands of agents interacting in virtual geographic environments and endowed with spatial cognitive ca-

[2] In this paper we cannot review the large body of literature dealing with the cognitive aspects of human spatial behavior. The interested reader can consult an in-depth review on the subject in [21] as well as several papers in the proceedings of the COSIT Conference.

pabilities. The application domain in which we are currently applying our geo-simulation approach is the simulation of crowd behaviors in urban environments.

Section 2 presents the requirements that we selected for the development of the MAGS platform in the context of recent work on the simulation of crowd behavior and pedestrian flows. As an illustration of the agent's cognitive spatial capabilities, Sections 3 and 4 present the main characteristics of the perception and navigation mechanisms implemented in the MAGS platform. Section 5 briefly outlines the other main agents' characteristics (needs, objectives, etc.) and presents an example of a simulation involving several hundreds of agents moving in a portion of Quebec city. Section 6 presents some performance results and concludes the paper.

## 2 Requirements for the MAGS Project

Several studies have been carried out on crowd movements in a portion of a urban environment represented on a 2D map and successful simulations have been created on the basis of mathematical models used in physics to simulate flows of particles in constrained environments. At medium and high pedestrian densities, the motion of pedestrian crowds shows similarities with the motion of fluid particles, giving rise to self-organization phenomena [15]. Such approaches have been used to simulate the formation of lanes of pedestrians on busy pavements [14] and to study evacuation strategies [13]. These approaches, which model the interactions between individuals in a quite simplified way (in terms of physical interactions of particles) are successful when simulating the flow of dense crowds in various situations. However, they cannot differentiate between different types of individuals with different goals and behaviors. Jager and his colleagues [17] developed a multi-agent system to simulate clustering and fighting behaviors of two-party crowds. They provided agents with simple rules based on the recognition of own-party agents and other-party agents (as a result of scanning an area 40 by 40 cells around the agent) and the agent's level of aggression motivation. Some simple clustering behaviors emerged from the simulations, which resembled real-world crowd phenomena.

Several systems have been designed to study pedestrian flows and movement at a strategic level. The STREETS System [28] applies an approach similar to the TRANSIMS model in which the simulation of the activities of pedestrians in urban districts follows a two-stage approach. In the first stage, the system exploits socio-economic data sets to predetermine the pedestrians' intended activity schedules which are used to "load" the agents into the simulation component. In the second stage, a simulation generates the movements of a population of agents representing pedestrians [12]. The simulation is influenced by the urban district's spatial configuration, pre-determined activity schedules and the distribution of land-uses. The PEDFLOW System [19] is another multiagent microsimulation system which is used to study conflicting pedestrian flows on a section of sidewalk or in an open or enclosed space with obstacles. Each pedestrian is represented as a single process which makes decisions about the pedestrian's movements. The simulated space is mapped onto a grid. A pedestrian agent occupies a grid element for a length of time required by its walking speed. A shared data structure

is used to record the current position of every pedestrian as well as location information about obstacles and the pavement. Displacement behavior is specified in the form of rules which take into account a number of parameters such as preferred gap size, desired walking speed and personal space measure.

The currently available systems for crowd and pedestrian flow simulation rely on very simple agent movements on grids (possibly directed by cellular automata) which do not take into account terrain characteristics and environmental factors that may influence the agents' perception and navigation. In addition, their decision making capabilities are quite basic (simple displacement rules) and group behaviors are non existent or at best very simple. In order to allow agents to simulate cognitive spatial activities in geosimulation systems we agreed upon basic principles: 1) an agent should be able to perceive the spatial environment as well as the objects and other agents surrounding it; 2) the spatial environment and the static objects it contains should be generated from data contained in geographic information systems (GIS) and related databases; 3) agents should be able to efficiently plan their activities based on their internal states and goals as well as the information they perceive in the virtual space.

Taking advantage of our previous experience with the development of *PADI-Simul*, a multi-agent system simulating basic navigation behaviors of hundreds of agents in a 2D sketch of a natural park [5] [23], we selected a set of requirements which involve integrating several technologies into the MAGS platform: GIS, multi-agent systems, 3D real-time animation engine and parallel processing. The set of requirements that we selected is as follows:

*We need to create a virtual geographic environment (VGE) in 2D (and possibly 3D) from reliable GIS sources.*

*We need to create agents of various types, each agent being individualized.*

*An agent must be able to perceive its environment, to navigate autonomously and to react to changes occurring in the VGE.*

*The agents' characteristics must reflect various possible states (static, dynamic, possession, etc.) and the agents' behaviors must offer efficient planning capabilities (reactive planning based on objectives).*

*Agents must be able to display group behaviors and to communicate with other agents.*

*The system must be optimized and allow simulations involving several thousand agents in relatively large spaces (a portion of a city for example).*

*An agent needs a memory capability in order to organize the knowledge about the VGE that it obtained from past experience.*

*Simulation scenarios must be specified easily, including the initialization of agents and the VGE and the introduction of specific events influencing the simulation.*

Several teams work on the development of digital representations of urban spaces (see for example [4]) in order to create "virtual cities" that can be explored by designers, urban planners and citizens in order to assess various characteristics of urban projects. People explore these virtual cities thanks to a 3D visualization engine that enables them to control a point that moves in the virtual space as well as virtual cameras that are used to observe the landscape. However, we do not know any system that enables a large number of virtual agents to move in a virtual city, perceiving the landscape around them and acting according to their perceptions. So, we first worked on an ap-

proach to generate the VGE as a 3D model in which agents can navigate. To this end, we elaborated a suite of transformation processes based on Geomedia [16] and 3ds Max software [3]. The VGE used in our current simulation is created from topographic data of a portion of Quebec city (scale: 1:20000), a digital elevation model and a data base giving the characteristics of the main buildings. In addition, a module of the MAGS System generates a collection of bitmaps which are used by the agents to obtain knowledge about the space surrounding them (Section 3). At the heart of the MAGS simulation engine there are 5 main modules: 1) a thread manager which coordinates the activities of the other modules; 2) a 3D engine which manages the display of the 2D or 3D VGE and cameras; 3) the agent manager and related modules; 4) the VGE manager and 5) the user interface manager.

As an illustration of the agent's cognitive spatial capabilities, the two following sections present the main characteristics of the perception and navigation mechanisms implemented in the MAGS platform.

## 3 Agent Perception

Perception is an important agent ability which must be carefully simulated in a 3D VGE if we want that agents exhibit plausible cognitive spatial behaviors [8]. We must bear in mind that simulating visual perception is a resource intensive process which must be optimized if we want to enable thousands of agents to perceive the VGE in real-time. By analogy to human spatial perception, we identified several perception modes for MAGS agents: 1) perception of terrain characteristics (elevation and slopes) in the area surrounding the agent; 2) perception of the landscape surrounding the agent (including buildings and static objects); 3) perception of other mobile agents navigating in the agent's range of perception; 4) perception of dynamic areas with specific properties such as a smoky area or zones having pleasant odors; 5) perception of special events (detonation, explosion, etc.) occurring in the agent's vicinity; 6) perception of messages communicated by other agents. We developed several mechanisms that enable MAGS agents to take advantage of these perception modes.

***Encoding spatial data.*** Spatial information is recorded in a raster mode which enables agents to access the information contained in various bitmaps that encode different kinds of information about the terrain characteristics and the objects contained in the VGE. The *HeightMap* is a 2D matrix (or grid) which represents the space of the VGE. It is generated from data contained in a digital elevation model and different layers of the GIS data base defining buildings and all the information that may influence the agents' perception and navigation. Every cell contains a single value indicating the height of the corresponding point relative to the point of lowest elevation in the VGE. Figure 2 presents the plan (x, z) of section A shown in the Height Map of Figure 1. We can observe that building2's height is higher than building1's height. But, the Height Map encodes that the top of building1 is higher than the top of building2 relative to sea level. The agent perception module uses this simple structure to determine the visibility of the matrix cells (using the elevation information). Hence, an agent can perceive the

terrain characteristics (slope, elevation) as well as objects that are in its range of perception. The agents' positions are recorded and updated in another data structure called *LocationMap*, a 2D matrix of cells in which every cell contains a pointer to the agent occupying the corresponding position. If an agent has a dimension allowing it to occupy several cells, several pointers refer to the same agent. Hence, it is possible to analyze the visibility of a cell using the *HeightMap* and to determine, using the *LocationMap*, if an agent located at the corresponding position is visible.

***Visual perception.*** Several researchers have already studied the problem of simulating perception in an environment represented by a height map [9] [2]. The goal of these techniques was to determine the visibility of all the cells of the height map which are in an observer's field of vision. They use lines of sight in order to test the cells' visiblity (labelled as visible or not) from the observer's location. If the observer moves, it is necessary to compute the visibility map corresponding to the new position. This technique is by far too inefficient to simulate the perception of thousands of agents moving in real time. We propose a solution extending Franklin's algorithm in a way that enables agents to perceive the environment as well as other agents in real time. The agent's perception field is represented by an isosceles triangle, the main vertex being at the agent's location, the congruent sides of the triangle limiting the perception field and the bisector of the main angle corresponding to the agent's direction of movement. The length of the bisector corresponds to what we call the perception radius. The angle of perception (Figure 3B) is a parameter that can be adjusted (currently set at 90 degrees).

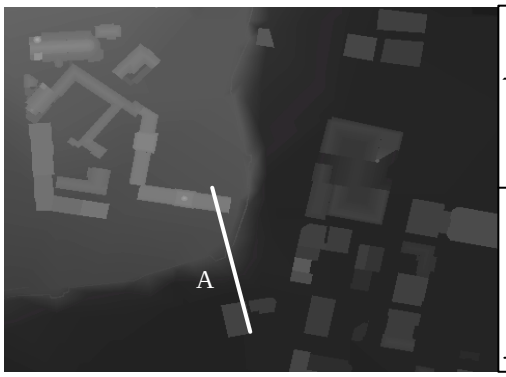


**Fig.1.** Height Map of a part of Quebec city

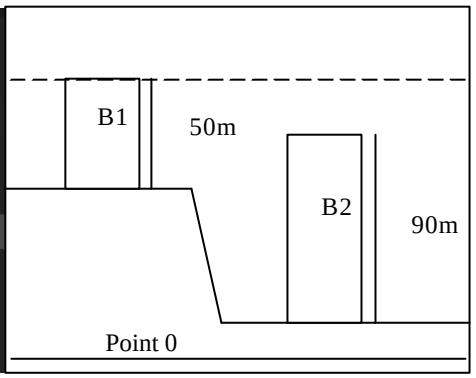


**Fig.2.** Plan (x,z) of section A in Fig. 1

We developed an algorithm which computes, for each cell in the agent's field of perception, the line of sight linking the agent's position to the cell in order to determine if the cell is visible or not. This computation takes into account the cell's height and the visibility of the other cells that may block the line of sight. As an illustration, our algorithm browses an environment of 300x300 cells and computes a visibility map in 27 ms on a Pentium 1000Mhz. This algorithm has a complexity $O(n^2)$ in which $n$ represents the perception radius. During a simulation, each agent must compute its own

visibility map at every iteration of the simulation engine. It is obviously impossible to allocate 27 ms to each agent in order to browse a height map of 300x300, using a perception radius of 300 cells. The solution is to reduce the length of the perception radius in order to decrease the computing time. Figure 3A shows how the perception computing time increases when we increase the perception radius. In the context of our simulator in which one pixel represents one meter approximately, we can think about limiting the perception of an agent's immediate neighborhood to 20 meters. In these conditions, the necessary time to process an agent's perception is 0,12 ms with a 90 degree angle of vision and a perception radius of 20 cells. If 1000 agents are moving at the same time, the global perception time would be 120 ms. Considering that a cycle in the simulator takes 33ms (to emulate real time), we would allow approximately 250 agents to perceive at every cycle, knowing that in a real-time 3D engine we use 30 cycles per second (hence, 33 ms / cycle).

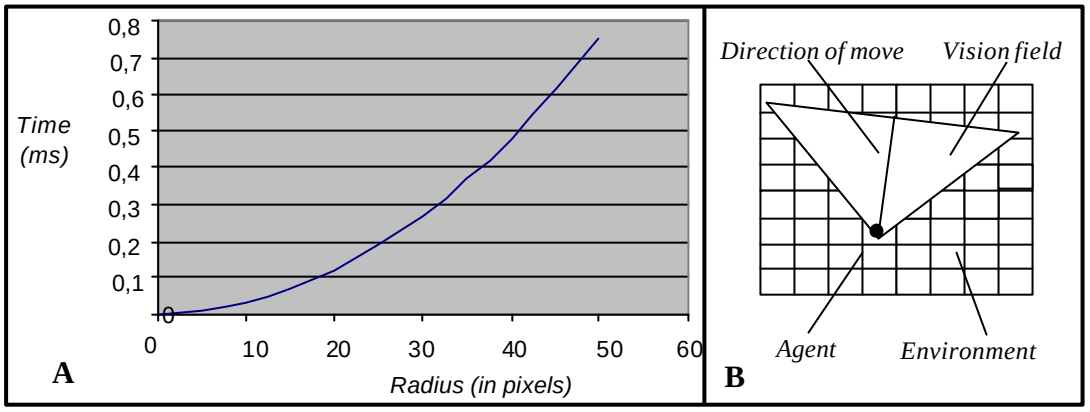


**Fig.3A.** Calculation time for the vision algorithm **Fig.3B.** Agent's field of perception

However, limiting the perception to 20 meters does not correspond to people's experience in the real world. Buildings distant from several kilometers can been seen in good visibility conditions. However, it is rather rare that we can distinguish smaller objects moving several kilometers away. Thus, we had the idea to compute a *static visibility map* in a pre-processing phase taking place before the simulations. Each cell of the static visibility map has a list of pointers to the static objects that are visible from this cell. Hence, an agent can directly access the static visibility map in order to determine which static objects (buildings, etc.) can be perceived from its position. This perception mode which is called *static perception* complements the *dynamic perception* mode that allows an agent to perceive in real time all the objects located within its perception radius. It is clear that this perception radius can be increased depending on the performance of the computer and the number of agents that need to perceive during each cycle of the simulation engine. Our approach allows an agent to have a global vision of the VGE (static perception) while keeping a focus on what is happening in front of it (dynamic perception).

***Perception of roads, paths and regions.*** In order to optimize the agent navigation function, we generate a bitmap called an *Ariadne Map* from the GIS data. In this bitmap streets and paths are colored in red (Figure 4A), and an agent can directly access it to determine which cells around it correspond to a street or path. The Ariadne Map can be generated in several colors in order to differentiate particular categories of ways such as sidewalks, roads, paths, bike paths, etc. Specific areas such as public squares can be perceived by the agents in the VGE. These areas are identified in the GIS and a bitmap called the *Zone Map* is automatically generated (Figure 4B). Each area is associated with a list of properties recorded in an auxiliary file. When an agent enters such a zone, it can access the corresponding information and act accordingly.

***Perception of dynamic areas (or volumes).*** Certain gaseous phenomena such as smoke, dense gases and odors are related to the VGE atmosphere and cannot be modeled using static objects or moving agents. They are associated with areas or volumes whose properties (boundaries, local density, etc.) change dynamically under the influence of external forces like the wind. A good way to simulate such phenomena is to use particle systems [30]. We developed such a module, but we will not go into detail about it in this paper. What is of interest here is the way that agents perceive these phenomena. Our particle system encodes the position of each particle in a *Gas Bitmap* as well as a pointer to a file in which the particle characteristics are encoded (particle type, density). An agent can access this gas bitmap in order to determine if there are some particles at its location. If yes, it can get the information about the particles directly and this information is taken into account by the agent behavior module. Indeed, gaseous phenomena are dynamic and the areas/volumes that they occupy change. Our particle system computes the particles' trajectories and takes snapshots of their positions (encoded in the Gas Bitmap) and sends them to the simulation system every k simulation cycles (k depends on how rapidly the phenomenon changes).

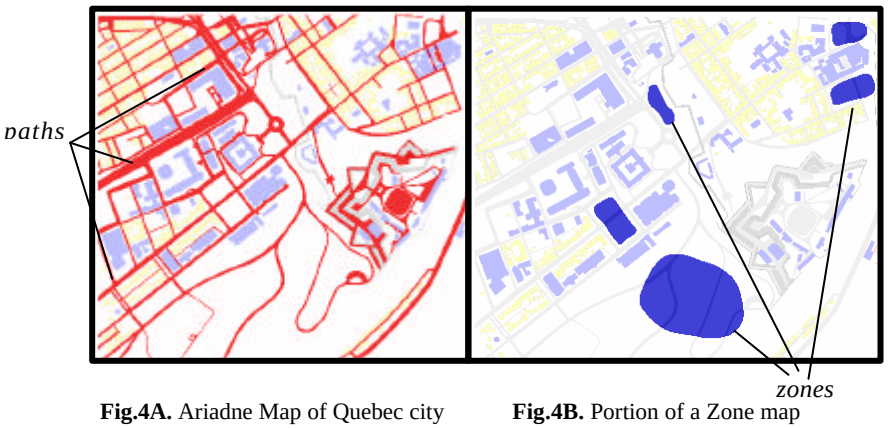


**Fig.4A.** Ariadne Map of Quebec city **Fig.4B.** Portion of a Zone map

***Perception of the effects of field-generating objects.*** Agents may react to events occurring in the VGE such as the detonation of a fire cracker and the explosion of a tear gas canister. These events are simulated by what we call *field-generating objects*, which emulate the propagation of sound or light in the atmosphere. The user may specify in the simulation scenario when and where in the VGE such an object will be

triggered in order to study crowd behavior characteristics through the reactions of individual agents. These objects may also be created by agents in the VGE such as a policeman throwing a tear gas canister or using an ultra-sound whistle to call his dog. Hence, agents may react in different ways to the effects of field-generating objects. A dog will react to the sound of the ultra-sound whistle, but a person will not. In the MAGS system a field-generating object is implemented as a special type of agent whose behavior reflects the consequences of triggering the object. The field-generating object identifies which agents are susceptible to perceive its effects according to the distance and to the agents' perception capabilities, and sends them a message to warn them about the occurrence of the corresponding event. We chose this approach[3] for efficiency purposes, because it would have been very costly to use a function that enables agents to scan the VGE in order to monitor the activation of field-generating objects. In fact, the messages emitted by field-generating objects simply simulate the transmission of information to the agents perceiving the associated phenomenon and mimic the transmission of sound or light in the atmosphere. A field-generating object such as the explosion of a tear gas cannister may also trigger a particle system which generates a gazeous phenomenon (see previous sub-section). Consequently, an agent may first react to the perception of a canister's explosion (after receiving the message from the corresponding field-generating object) and then to the perception of the tear gas emitted by the canister (after accessing the corresponding gas bitmap).

***Communication between agents.*** In the real world people communicate verbally in different ways, speaking or shouting for example. We simulate verbal communications in a simple way: a MAGS agent can send messages to one or several agents if it wants to communicate with them. A simple computation can limit the maximum distance at which a message broadcasted by an agent can be perceived by other agents depending on the distance separating them from the emitting agent.

## 4 Agent Navigation

While navigating, agents may either follow paths (we call this navigation mode « *following-a-path-mode* ») or move through open spaces (we call this navigation mode « *obstacle-avoidance-mode* »). When an agent is in the *obstacle-avoidance-mode*, the navigation module accesses the Height Map's portion which is visible by the agent (obtained from the dynamic perception function) in order to evaluate the difficulty of crossing the space separating the agent from its destination. When an agent is in the *following-a-path-mode*, the navigation module accesses the Ariadne Map's portion which is visible by the agent (obtained from the dynamic perception function) in order to compute its next move: this navigation mode requires less computational resources than the obstacle-avoidance-mode. An agent can opportunistically change its navigation mode in order to draw nearer to its destination in the VGE. We also developed a

[3] There is a similarity with the notion of *affordance* [10]. Affordances are defined as what objects or things offer people to do with them. In the same way field-generating objects indicate to the agents how to react to the events that they trigger.

function for collision avoidance which is used to manage the agents' local interactions when they move on the Ariadne Map. As an illustration, Figure 5 shows 12 agents in different situations and aiming at the same destination. Some of them (n° 1 to n° 5) are in the *following-a-path-mode* and the others (n° 6 to n° 12) are in the *obstacle-avoidance-mode*. Let us briefly comment upon the agent navigation behavior using this example. In order to determine its next move, an agent must choose one of the eight cells surrounding its current position. To this end, the agent goes through 6 steps :

***Step 1: Direct beam computing***

Using a ray tracing function, the system computes the direction of the *direct Beam* originating from the agent's current position and aiming at its current destination

***Step 2: Path_mode checking***

Whenever it is possible, an agent tries to follow a path. Hence, when the agent is not following a path (it is in the *obstacle-avoidance-mode*) the system calls the perceptual function that tries to return the nearest visible position on the path of the Ariadne map. If such a position is found, the agent switches to the *following-a-path-mode* and takes this position as an intermediate destination (ex: agent n° 9 in Figure 5).

***Step 3: Beam Tracing***

A beam is traced in the *direct beam* direction until reaching a predetermined distance (for example 20 pixels) called the *Beam_range* which is less than or equal to the perception radius. The *Beam_range* value is much less important when the agent is following a path (ex: agents n° 1 to n° 5 in Figure 5) than when it is moving in an open space (ex: agents n° 6 to n° 12). In the *following-a-path-mode*, the tracing function succeeds when the beam crosses an Ariadne pixel (ex: agents n° 4 to n° 5). In the *obstacle-avoidance-mode* the tracing function succeeds when the beam does not hit an obstacle (ex : agent n° 10).

In both navigation modes, when the beam tracing function fails, the system sends a second beam to the right of the *direct beam*, changing its direction by a *Beam_variation_angle*. If this new beam also fails, the system sends a third beam symmetrical to the second, but this time to the left of the *direct beam*. This process goes on until either the beam tracing succeeds, or the angle between the beam direction and the *direct beam* direction goes beyond a given threshold called the *Beam_maximum_angle*[4]. In Figure 5, agents n° 1, 2 and 3 are following a path, each of them moving in a direction (represented by its beam on the figure) which enables it to stay on the path, since the corresponding *Beam_maximum_angle* is not yet exceeded. Agents n° 6, 7 and 8 are in the *obstacle-avoidance-mode*: the function looks for the direction nearest to the *direct beam* that can be followed without hitting an obstacle.

***Step 4: Beam Tracing Exception***

If Step 3 failed to trace a beam and the agent is in the *following-a-path-mode*, it goes into the *obstacle-avoidance-mode*. This is the case of agent n° 11 for which the

[4] The *Beam_variation_angle* and the *Beam_range* are parameters that are associated to the agent profile and can be adjusted. The *average_beam_range* value is 20 pixels. The average *beam _variation_angle* is $\pi/32$ (or 5.625 degrees). The *Beam_maximum_angle* is another parameter that can be adjusted. The average value is 45 degrees on each side of the direct beam.

*Beam_maximum_angle* is exceeded: it is now in the *obstacle-avoidance-mode*. If the agent is already in the *obstacle-avoidance-mode*, it is blocked by an obstacle because the *Beam_maximum_angle* is exceeded. In that case a function called *GetOutOfBlockage* moves the agent away from the obstacle. This is the case of agent n° 12 which is currently blocked. The function *GetOutOfBlockage* will enable it to jump to the neighboring Ariadne path.

***Step 5: Oscillation detection***

In order to avoid certain critical situations in which the agent could get lost by following cyclic trajectories, we call a function that detects oscillations. This function records the agent's last position every *k* steps and analyzes the evolution of the distance between this position and the agent's current position. If an oscillation is detected, we call the *GetOutOfBlockage* function.

***Step 6: Collision detection and agent displacement***

*CollisionDetection* and *AgentDisplacement* are functions accessing the Location Map. If the location *l* chosen for the agent's displacement is already occupied by another mobile agent, a simple displacement algorithm determines a position next to location *l* .

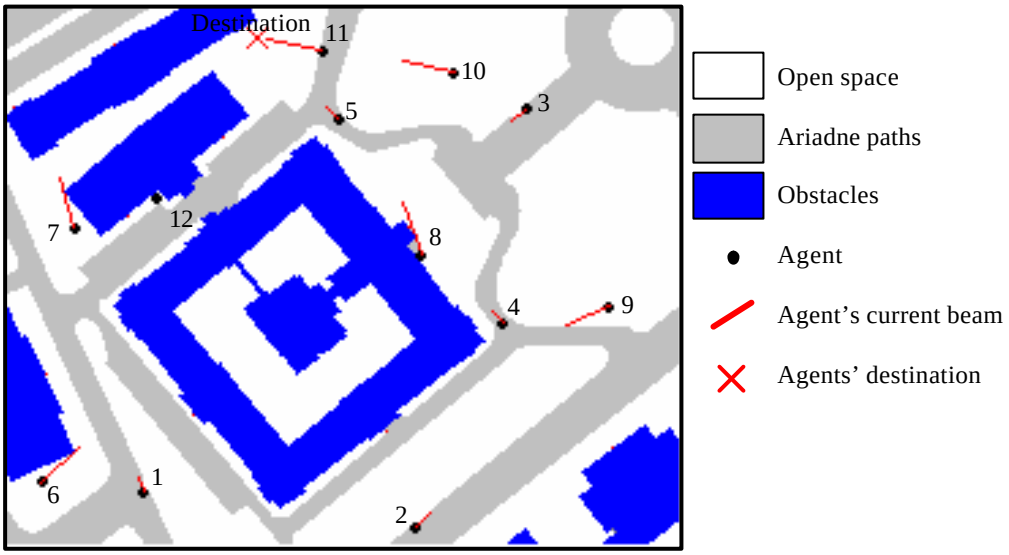


**Fig. 5.** Different navigation modes

To sum up, the agent navigation module takes advantage of the Height Map, the Ariadne Map, the Location Map and the ray tracing function in order to move the agents to their destinations by opportunistically choosing either the *following a path* or the *obstacle avoidance* modes. This simple mechanism allows the system to perform the simultaneous navigation of a large number of agents (several thousands of agents on a grid of 2048 x 2048 pixels). Besides this navigation mechanisms, we developed a set of basic navigation functions that can be used by the behavior module:

- `ChangeDesiredSpeed` : change the agent's desired speed
- `Flee` : flee from another agent's position or from a given location
- `MoveInsideDisc` : useful to keep an agent moving in a circular area

- `MoveAroundDisc` : useful to keep an agent moving around a circular area
- `Goto` : go to an object's location or to a given location on the map
- `Follow` : follow a mobile agent
- `Walk` : walk around the VGE by choosing destinations randomly.

## 5 The Agent's Knowledge

In order to behave autonomously, agents must be able to interact with their environment (VGE, objects and other agents), make decisions with respect to their own states and preferences and act accordingly. We consider five types of agents: mobile, object, field-generating object, cluster and group. *Mobile agents* such as persons and cars are able to move in the VGE. *Object agents* such as buildings and trees are not able to move. *Cluster agents* are dynamically created when several agents gather in a definite area for a certain duration. Moreover, clusters allow more control over member agents at a higher level of abstraction. *Group agents* associate agents having a common characteristic such as policemen. We can also define a group containing other groups. For example, a buildings group contains the different groups of buildings having the same type such as residential, industrial, governmental and commercial buildings. Each agent of any type has its own behavior which depends on its profile.

An agent is characterized by a number of variables whose values describe the agent's state at any given time. We distinguish *stable states* and *dynamic states*. A *stable state* does not change during the simulation and is represented by a variable and its current value. For example, the fact that an agent is respectful of city regulations will not change during the simulation. A *dynamic state* is a state which can possibly change during the simulation. For example, an agent's tiredness can change during the simulation. A dynamic state is represented by a variable associated with a function which is used to compute how this variable changes values during the simulation. The variable is characterized by an initial value, a maximum value, an increase rate, a decrease rate, an upper threshold and a lower threshold which are used by the function. Using these parameters, the system can simulate the evolution of the agents' dynamic states and trigger the relevant behaviors. An agent is also associated with a set of objectives that it tries to reach. The objectives are organized in hierarchies such that elementary objectives are associated with actions that the agent can perform. Each agent owns a set of objectives corresponding to its needs. An objective is associated with rules containing constraints on the activation and completion of the objective. Constraints are dependent on time, on the agent's states, and on the environment's states. The selection of the current agent's behavior relies on the priority of its objectives. Each need is associated with a priority which varies according to the agent's profile. An objective's priority is primarily a function of the corresponding need's priority. It is also subject to modifications brought about by the opportunities that the agent perceives and by the temporal constraints applying on the objective. Each agent of any type has its own behavior depending on its profile. A profile contains a set of roles that an agent can play during a simulation. It is also used to personalize the agent's needs. A role is represented as a tree of objectives. The tree structure allows us to define a behavior at different levels of abstraction. The root nodes are composite objectives and leafs are elementary objectives. The actions are associated with the elementary objectives.

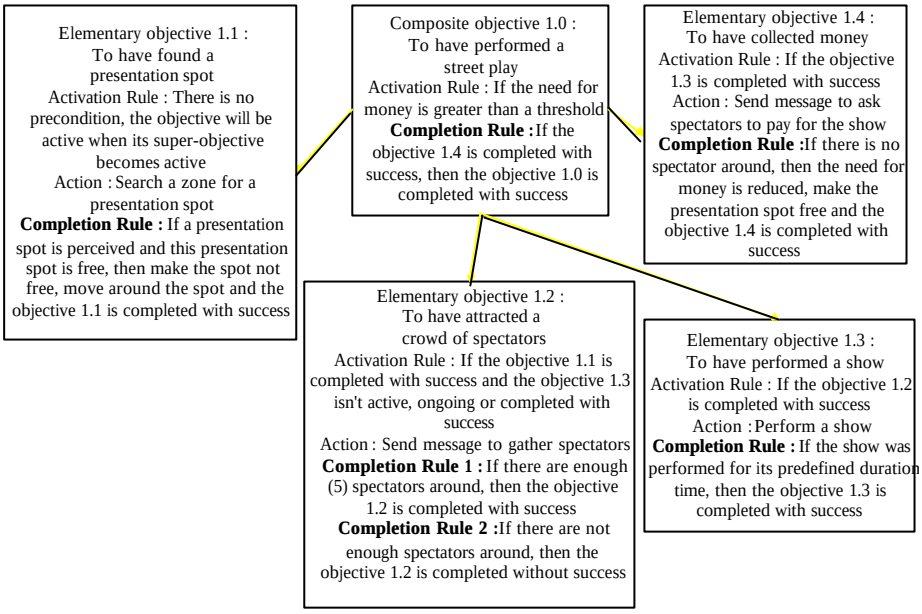


**Fig. 6.** The buskers' tree of objectives

As an illustration we developed a simple scenario in which mobile agents represent people wandering in a VGE of a portion of Quebec city. We also specified a group of buskers who try to find spots in order to entertain the passers-by. Figure 6 presents the simple objective tree of buskers. The main (composite) objective (1.0) is activated if the busker's need for money is greater than a given threshold. Four elementary objectives are associated with *Objective 1.0*. *Objective 1.1* is activated when *Objective 1.0* is active and aims at finding a presentation spot: it triggers an action that makes the busker wander through the city in search of a presentation spot. If the busker perceives a free presentation spot, it reserves this spot, starts to move around it and *Objective 1.1* is completed with success. *Objective 1.2* is activated when *Objective 1.1* is completed with success. The action triggered by *Objective 1.2* is the sending of messages to agents passing by in order to advise them that a show will start soon at this spot. If at least 5 passers-by stay near the spot, *Objective 1.2* is completed with success. If after a certain time there are not enough spectators around the busker's spot, *Objective 1.2* is completed without success. *Objective 1.3* is activated when *Objective 1.2* is completed with success. The action makes the busker agent present its show for a given period of time. Then, *Objective 1.3* is completed with success, *Objective 1.4* is activated and the busker agent sends a message to the spectator agents which are still near the presentation spot in order to ask them for some money.

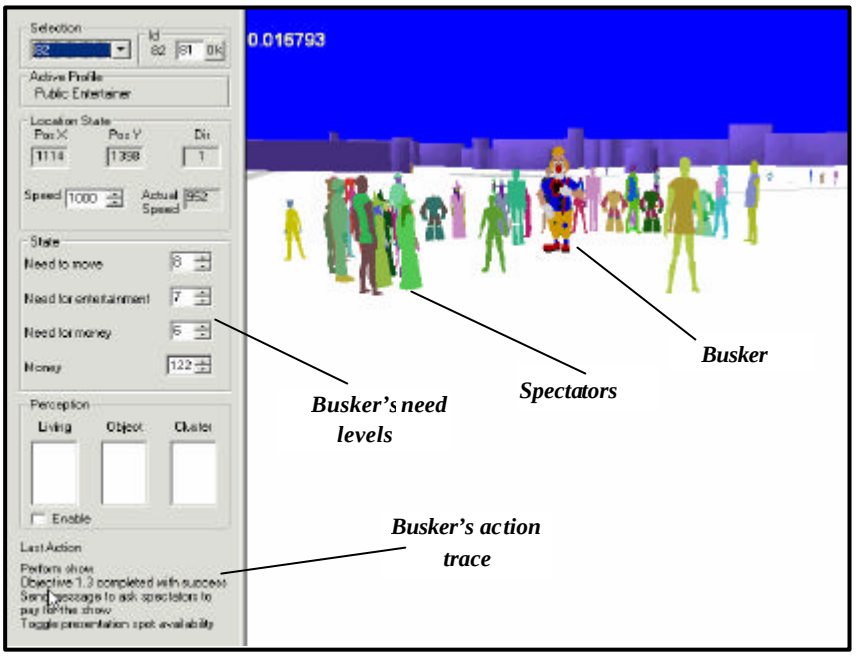


**Fig. 7.** Busker and audience

After a certain time the busker agent's need for money decreases, it frees the spot and completes *Objective 1.4* with success which leads to the completion with success of *Objective 1.0*. Then, the busker activates another need (which takes the highest priority) that makes the agent wander around in the VGE until its need for money reaches the threshold that triggers *Objective 1.0* again. Passers-by also have an objective tree that emphasizes their need for being entertained by buskers. Space limitations prevent us from presenting this tree in this paper.

Figure 7 presents a snapshot of the simulation which displays the passers-by gathered around a busker in a 3D model of the city. On the left-hand side of the screen we see various characteristics of the selected agent (the busker in this case). At the bottom of this part of the screen, the *Last Action* section shows that the busker has completed *Objective 1.3* and is performing the actions of *Objective 1.4.*

Figure 8 presents a 2D view of people (little dots on the streets) marching in a portion of Quebec city. Figure 9A offers a 3D view that shows people marching and gathering on a square of the city. Figure 9B presents the agent's point of view when moving among other agents. Our system enables the user to manipulate a camera to explore the 3D VGE according to various modes. In Figure 9A we view the scene from above (third person view). In Figure 9B we view the scene from the position of an agent (first person view). The user may view the landscape from any agent's point of view.

**Fig. 8.** A peace walk in Quebec city (2D)

The 3D characters associated with the agents are not elaborate since the emphasis of the MAG project was not put on the creation of character animations.

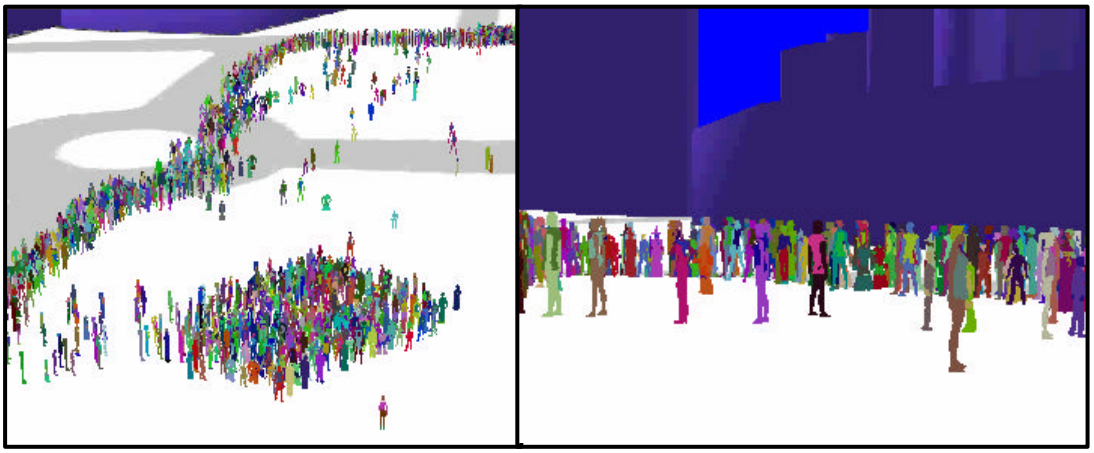

**Fig. 9A.** Observing a march in Quebec city (from above)

**Fig. 9B.** Observing the scene from an agent's point of view

## 6 Discussion and conclusions

Since perception and navigation are the two fundamental spatial cognitive capabilities available to MAGS agents, we wanted to assess the efficiency of our current implementation with respect to the durations of the navigation and perception cycles. Let us recall that navigation and perception are performed by two separate threads in our architecture. We carried out tests on the example of agents marching from one location to another in Quebec city (represented on an image of 2048x2048 pixels). The tests were performed on a Pentium P4, 2.66 GHz with 2 Go of RAM and a Radeon 9700 Pro graphic card. We grouped agents in clusters in order to lessen the load on the perception thread. In each cluster we have a leader that fully perceives (*Beam_range* of 30 pixels in the *obstacle-avoidance-mode* and 13 pixels in the *following-a-path-mode*). Agents playing the role of followers in the cluster follow the leader and have a *Beam_range* of 1 pixel. We carried out the tests with various populations of agents (1000, 2000, 3000, 5000, 10 000 agents) and with different cluster sizes (100, 50, 25 and 15 agents per cluster). In order to make the measurements, we had to run the system in a debug mode which is less efficient than the normal simulation mode. The navigation and perception cycles do not have the same duration because they require different computing resources. Figure 10 shows the duration of the navigation cycle for the various populations and cluster sizes. Since an agent moves at most one pixel (equivalent to about one meter according to the map scale) per cycle, this means that in the worst case (10 000 agents clustered in groups of 100), the agents move 6.25 pixels per second. Considering that the average speed of a fast walking person is 5 km/hour or 1.38 m/s (corresponding to 1.38 pixel/s in the simulation), our simulation shows displacements accelerated by an average factor of 4.5. This leaves time for the system to devote computing resources to the behavior thread.

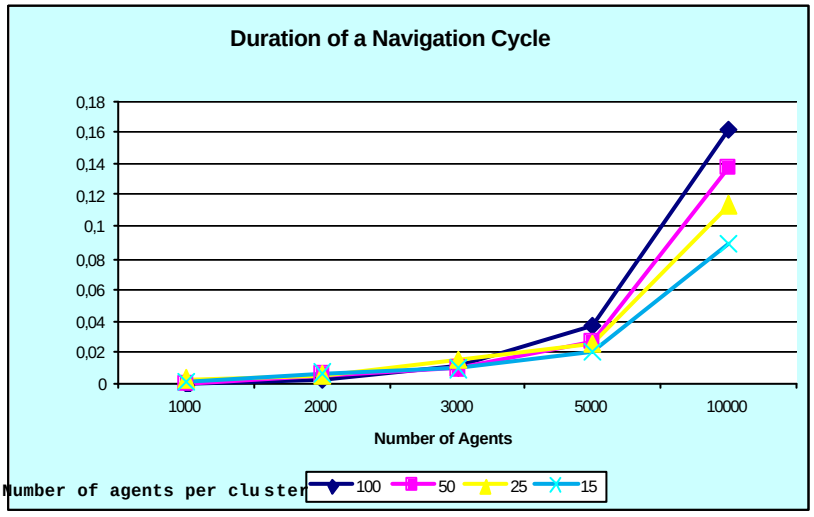


**Fig. 10.** Duration of a navigation cycle

Perception takes more time than navigation. Figure 11 shows the ratio of the navigation cycle duration and the perception cycle duration in the different cases of our experiments. Since the navigation and perception threads run in parallel, a ratio of 4 means that an agent will be able to move 4 pixels at most before it gets new information from the perception module. This is not a problem since basic functions such as collision avoidance are part of the navigation cycle. Hence, it is quite acceptable that an agent moves 4 to 6 pixels (equivalent to 4 to 6 meters) before taking into account new information from the dynamic perception function. However, for the worst case of 15 agents per clusters (see Figure 11) the ratio value varies between 9 and 15 times. This is too much. Hence, in our current implementation clusters of at most 25 agents give acceptable results in the debug mode. The performance is better in the simulation mode. Considering the improvement of hardware performance anticipated for the coming years, we expect that the current performance limitations of the MAGS platform will be overcome.

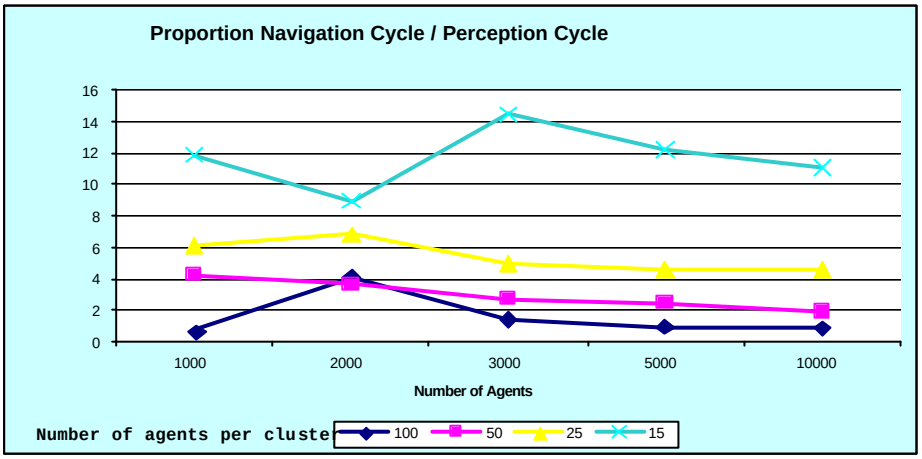


**Fig. 11.** Ratio of the navigation and perception cycles' durations

The MAGS system is not fully completed and optimized. For example, we are currently working on the development of the scenario specification module and on the agent's spatial memory capability. We are also exploring various alternatives to distribute the computing load on networked computers. For example, the particle systems used to simulate dynamic areas, such as dense gaz and smoke, require a large amount of computing resources, especially when simulating several phenomena at once (i.e. several tear gaz cannisters exploding in front a group of demonstrators). We developed an initial version of a distributed system in which MAGS and the particle system simulator are on two different computers, the particle system simulator, sending an update of the gas bitmap (see Section 3) at regular time intervals that can be adjusted by the user. However, we need to dedvis e a complete distributed solution for MAGS considering the possible distribution strategies available for this kind of applications [25] [26]. However, the current version of the MAGS system and its application to the simulation of crowd behavior show the interest of building agent-based geo-simulation environ-

ments as well as the potential of this kind of approach for various kinds of simulations in which agents should exhibit plausible cognitive spatial behaviors [8].

## References

1. Beckman, R. J. (ed.): "The Dallas – Fort Worth Study", Los Alamos unclassified report LAUR-97-4502LANL, Los Alamos National Laboratory, Los Alamos NM, available at http://transims.tsasa.lanl.gov/ (1997)
2. De Floriani, L, Magillo, P.: Visibility algorithms on DTMs, *International Journal of Geographic Information Systems*, 8(1), 13–41(1994)
3. Discreet: http://www.discreet.com/products/3dsmax/, (Last visit March 2003)
4. Dodge, M., Doyle, S., Smith, A., Fleetwood, S.: Towards the Virtual City: VR & Internet GIS for Urban Planning, Virtual Reality and Geographical Information Systems Workshop, Birkbeck College (1998)
5. Epstein, S. L., Moulin, B., Chaker, W., Glasgow, J., Gancet, J.: Pragmatism and spatial layout design, in D. Montello (edt.), *Spatial Information Theory: Foundations for Geographic Information Science*, Springer Verlag LNCS 2205,189-205 (2001)
6. Ettema, D., Timmermans, H.: *Activity Based Approaches to Travel Analysis*, Elsevier Science, Amsterdam (1999)
7. Fotheringham, A. S., O'Kelly, M. E.: *Spatial Interaction Models : Formulation and Applications*, Kluwer Academic Publishers, Dordrecht (1989)
8. Frank, A.U., Bittner, S., Raubal, M.: Spatial and cognitive simulation with multi-agent systems, in D. Montello (edt.), *Spatial information Theory: Foundations for Geographic Information Science*, Springer Verlag, LNCS 2205,124-139 (2001)
9. Franklin, W.R.: Applications of analytical cartography, *Cartography and Geographic Information Systems*, 27(3), 225–237 (2000)
10. Gibson, J.: *The Ecological Approach to Visual Perception*, Houghton Mifflin Company, Boston (1979)
11. Gimblett, H. R.: *Integrating Geographic Information Systems and Agent-Based Modeling Techniques for Simulating Social and Ecological Processes*, Oxford University Press (2002)
12. Hacklay, M., O'Sullivan, D., Thurstain-Goldwin, M., Schelhorn, T.: « So go downtown » : simulating pedestrian movements in town centres, *Environment and Planning B: Planning and Design*, 28(3), 343-359 (2001)
13. Helbing, D., Farkas, I. J., Vicsek, T.: Simulating dynamic features of escape panic, *Nature*, n. 407, 487-490 (2000)
14. Helbing, D., Molnar, P., Schweitzer, F. Computer simulation of pedestrian dynamics, In proceedings of the 3rd International Symposium of SFB 230, *Evolution of Natural Structures*, Sonderforschungsbereich, Stuttgart, Germany, 229-234 (1999)
15. Helbing, D., Molnar, P., Farkas, I. J., Bolay, K.: Self-organizing pedestrian movements, *Environment and Planning B: Planning and Design*, 28(3), 361-383 (2001)
16. Intergraph: Geomedia Professional, http://www.intergraph.com/gis/gmpro (2003)

17. Jager, W., Popping, R., van de Sande, H.: Clustering and fighting in two-party crowds: simulating approach-avoidance conflict, *Journal of Artificial Societies and Social simulation*, vol.4 n.3, http://jasss.soc.surrey.ac.uk/JASSS/4/3/7.html (2001)
18. Jennings, N., O'Hare, G.: *Foundations of Distributed Artificial Intelligence*, Wiley (1996)
19. Kerridge, J., Hine, J., Wigan, M.: Agent–based modelling of pedestrian movements: the questions that need to be asked, *Environment and Planning B: Planning and Design*, 28(3), 327-341 (2001)
20. Lee, D. B.: Retrospective on large-scale urban models, *Journal of the American Planning Association*, 60, 35-40 (1994)
21. Mark, D.M., Freksa, C., Hirtle, S.C., Lloyd, R., Tversky, B.: Cognitive models of geographic space, *International Journal of Geographical Information Science*, vol. 13 no. 8, 747-774 (1999)
22. Moss, S., Davidsson, P.: Multi-Agent-Based Simulation, Proc. of the 2nd Internat. Workshop MASB 2000, Springer Verlag, LNAI, n.1979 (2000)
23. Moulin, B., Chaker, W., Gancet, J. : PADI-Simul, an agent-based software which simulates the behaviors of hundreds of actors in a geographic space, To appear in *Journal on Computers, Environment and Urban Systems* (2003)
24. O'Sullivan, D., Torrens, P.: Cellular models of urban systems, in S. Bandini and T. Worsch (edts.), *Theoretical and Practical Issues on Cellular Automata*, Springer Verlag, also available from Centre for Advanced Spatial Analysis, Working Paper 22, June 2000, www.casa.ucl.ac.uk (2000)
25. Ray, C., Claramunt, C.: Atlas : A distributed system for the simulation of large-scale systems, In Chen, S.-C. et Voisard, A.(edts.), Proceedings of 10th ACM International Symposium On Advances In Geographic Information Systems, 155–162, McLean, VA, ACM Press (2002)
26. Righter, R., Walrand, J. C.:Distributed simulation of discrete event systems. *Proceedings of the IEEE*, 77(1), 99–113 (1989)
27. Sawyer, R.K.: Artificial societies: multiagent systems and the micro-macro link in sociological theory, *Sociological Methods and Research*, vol 31 n3, 325-363 (2003)
28. Schelhorn, T., O'Sullivan, D., Haklay, M., Thustain-Goodwin, M.: STREETS : An agent-based pedestrian model, CASA Working Paper 9, www.casa.ucl.ac.uk (1999)
29. Schillo, M., Fischer, K., Klein, C.T.: The micro-macro link in DAI and sociology, in S. Moss & P. Davidsson (edts.), *Multi-Agent Based Simulation*, Springer Verlag, Lectures Notes in Artificial Intelligence, n. 1979, 133-148 (2000)
30. Stam, J. : Interacting with Smoke and Fire in Real Time, *Communications of the ACM*, vol 43, n 7, 76-83(2000)
31. Torrens, P.M.: Can geocomputation save urban simulation? Throw some agents in the mixture, simmer and wait…, CASA Working Paper 32, www.casa.ucl.ac.uk (2001)
32. Weiss, G. (ed.): Multi-Agent systems, MIT Press (1999)
33. Wolfram, S.: *Cellular Automata and Complexity : Collected papers*, Addison-Wesley (1994)