Environment Model for Multiagent-Based Simulation of 3D Urban Systems

Stéphane GALLAND, Nicolas GAUD,
Jonathan DEMANGE, and Abderrafíâa KOUKAM

Multiagent Systems Group,
System and Transport Laboratory
University of Technology of Belfort Montbéliard
90010 Belfort cedex, France
{stephane.galland,nicolas.gaud,
jonathan.demange,abder.koukam}@utbm.fr
http://set.utbm.fr

Abstract. This paper presents the JaSIM environment model. It gathers experiences collected on design of the environment — as a first-class entity — coming from both multiagent and computer graphics domains. Its main goal consists in providing a sets of models and tools to easily implement crowd and traffic MABS[1]. JaSIM integrates efficient hierarchical and graph-based data structures for supporting $1D$ to $3D$ actions and perceptions of numerous agents. It provides geometrical, topological and semantical data about environmental objects to agents. This paper discusses the overall architecture and $3D$ modules of JaSIM environment model, and illustrates platform performances on two experiments.
Key words: Environment Model, Multiagent-based simulation, Urban Simulation, $3D$ Environment, Bounding Volume Hierarchy.

1 Introduction

This paper introduces JaSIM[2], a multiagent environment model dedicated to multiagent-based simulation in 3D virtual environments. JaSIM is an innovative simulation platform for implementation and deployment of urban simulations. JaSIM follows a microscopic simulation approach and it is mainly designed for traffic and crowd simulations. JaSIM platform is designed as an extension of Janus [9] combining organisational agent-oriented approach of Janus with object-oriented structure of JaSIM simulation model. This paper does not detail the entire JaSIM platform but focus on its environment model. This model tries to gather experiences collected on environment design coming from both multiagent and computer graphics domains. It thus integrates a full set of efficient hierarchical and graph-based data structures for supporting actions and perceptions of numerous autonomous entities. These structures are designed to

[1] MABS : MultiAgent Based Simulation
[2] stands for "JAva SImulation Model"

assemble geometrical, topological and semantical data in single model integrating all information required to manage realistic behaviours of pedestrians and vehicle drivers. JaSIM is designed to fully respect the key principles that govern multiagent-based simulation such as environmental integrity constraint, agent autonomy, and the clear distinction between agent's mind and body [13]. It also provides a collection of well-known design-patterns for an easy and reusable implementation of traffic and crowd simulation. JaSIM supports simulations in $1D$[3], $2D$ or $3D$ environments, however this paper focus on the implementation of $3D$ environment. The simulation in $3D$ environment is not always required, in most of cases simpler environmental structures may be used [21]. However many applications require a maximal level of accuracy and the use of a true $3D$ environment, mainly to enable agent $3D$ perception. Allowing agents to perceive their world in $3D$ enables the simulation of complex real situations like smoky environments, testing the visibility of security features like emergency exit... Physics simulation also requires a complete 3D simulated world to enable the computation of precise physics laws (i.e. "sub-microscopic simulation").

This paper is structured as follows. In section 2, a brief overview of previous works on notion of environment in situated multiagent systems is given. Section 3 introduces JaSIM environment model and detail various elements composing this model. Section 4 provides experimental results. Finally we draw conclusions and look at challenges for future works on JaSIM platform.

2 Related Works on Environmental Models

2.1 Background on Environment for Situated Multiagent Systems

In this article, environment is considered as a explicit part of multiagent system. This section focus on its role in situated multiagent systems and multiagent-based simulations. Three different points of view may be adopted to study the notion of environment in situated MAS: environment as the external world, environment as a medium for coordination, environment architecture [24]. This section mainly focus on the two last perspectives.

According to Weyns et al. [24], the two types of difficulties most often reported as being encountered when studying the MAS literature are: (i) the term *environment* has several different meanings, causing a lot of confusion, (ii) the functionnalities associated with the environment are often implicitly handled, or integrated in the MAS in an ad-hoc manner. This point indicates that the environment is failed to be considered as a *first-class module*.

The confusion on the environment definition is mainly caused by mixing up concepts and infrastructure. Sometimes the environment refers as the logical entity of a MAS in which the agents and other entities and resources are embedded. Sometimes the notion of environment is refering to the software infrastructure on which the MAS is executed or it may even refer to the underlying hardware infrastructure on which the MAS is running.

[3] 1D environment: graph-based model

Environment Architecture. Ferber's model and its extension [13] are based on three main principles: respect the environment integrity, agent autonomy, and clearly distinct agent's mind from its body. In other words, an agent should not directly modify the environment, and one agent may not decide for the others. In this theory, an agent does not perform actions but produces influences. Influences do not directly modify the environment and, from an agent point of view, nothing can be guaranteed about their result. Applying this theory at implementation level thus requires a two phases mechanism that firstly collects influences: *Influence phase*, and then computes results of their combination: *Reaction phase*. Reaction, which is managed by the environment itself, modifies the state of the environment by combining the influences of all agents according to a previous local state of the environment and the laws that govern the simulated world. This clear distinction between the products of the agent's behaviours and environmental reactions provides a way to handler simultaneous action in MAS and contributes to a better respect of modeling and simulation relations [25].

Beside the activity of the agents, resources or entities can produce activity in the environment too. Maintaining such dynamics is an important functionnality of the environment [2, 23]. Influences-Reaction model also enables modeling of environmental endogenous dynamics by considering the environment as being able to produce influences.

Environment as a medium for coordination. According to Odell et al. [15], "*the environment provides the conditions under which an entity (agent or object) exists.*" The authors distinguish between the *physical environment* and the *communication environment*. The physical environment provides the laws, rules, constraints and policies that govern and support the physical existence of agents and entities. The communication environment provides (i) the principles and processes that govern and support exchanges of ideas, knowledge and information, and (ii) the functions and structures that are commonly deployed to exchange communication, such as roles, groups and interactions protocols between roles and groups. Odell et al. [15] define an agent's *social environment* as "*a communication environment in which the agents interact in coordinated manner*". This approach is shared by Ferber et al. [7] who proposed to integrate the environment with the Agent-Group-Role organisational model.

2.2 Environment in 3D Urban Simulation

This section gives a short overview on various kinds of 3D simulation application domains targeted by JaSIM, with a focus on environment. In this paper three main types of urban simulation are focusing: traffic, crowd and city simulations. City simulation refers to 3D urban simulations aiming at simulating both life in streets and inside buildings. This implies to simulate both pedestrians and vehicles evolving in a virtual 3D urban environment. In 3D simulation, two kinds of data have to be clearly distinguished: the required data for rendering 3D scene and data for supporting the simulation and enabling complex autonomous

behaviours. This section only focus on the simulation's environment and its associated data.

Crowd Simulation. To evolve in a virtual world and exhibit a complex behavior, an agent requires several semantical, symbolical and topological data about its environment. Simplest behaviors consist in avoiding static and dynamic obstacles, and it thus requires only geometrical information on the various objects in the 3D world. More complex behaviors usually depend on the nature of the objects. This requires to add semantical information into the environment model. This semantic level intends to classify the various objects at different abstraction levels. Finally, to enable an agent to efficiently compute the best path between two points in the 3D world, topological informations on this world are required. This is usually done thanks to navigation graphs that are precomputed from the environment mesh [18]. Several platforms such as Massive [10] and Breve [12] are interesting about the physical interactions of the simulated entities. In a word, an environment model dedicated for crowd simulation should at least integrate three different layers of information: structural, semantical and topological [4, 6, 22].

Traffic Simulation. In traffic simulation model, we focus on reproducing the street life and it usually does not reflect what happens in buildings. This kinds of simulation requires informations about the road network (graph-based structure: lanes, junctions, intersections, interchanges), its rules (priority, road-sign, speed limits) and its neighbouring environment (building, square...). Semantic informations (streets names, buildings info) are also required and have to be added to the topological information of the road network to enable autonomous driving behavior [17]. Usually this kinds of information is stored in a specific database [4] generally called Urban Information System (UIS).

To conclude this section on related works, a small summary of the various missions assigned to the environment [24] in a classical MABS in virtual environment is provided.

M1 - Sharing informations: Environment is a shared structure for agents, where each of them perceives and acts. Associated data are usually composed of hierarchical structures gathering all the objects that make up the virtual world and are useful to agents. The environment is thus hierarchically decomposed into a set of areas, sub-areas and so on, each area being associated with the set of its objects. At each level of this hierarchy, objects are themselves connected through a set of graph-based structure to maintain the topological information, associated to a given sets of semantics. This gives a rise to a complex data structure that may be considered as a kind of clustered graph. This part of the environment contains all the structures used to organise structural, semantical and topological informations such as graphs, octrees, quadtrees, grids, etc.

M2 - Managing agents actions and interactions: This aspect is related to the management of agents' simultaneous and joint actions and to the preservation of the environmental integrity. For example, when two agents are

pushing the same box, the environment may compute the real location of the box according to physics laws. On this aspect, the influence-reaction [13] model may provide an efficient solution.

M3 - Managing perception and observation: The environment must be locally and partially observable. Thus agents can also manage the access to environmental informations and guarantee the partialness and localness of perceptions.

M4 - Maintaining endogenous dynamics: The environment is an active entity; it can have its own processes, independently of the ones of the agents. A typical example is the evaporation of artificial pheromones in ant colony based algorithms.

All these missions as assumed as essential for an environment model. The next section is dedicated to the description of the JaSIM environment model and describes how this set of environmental missions is implemented within JaSIM. Two constraints are integrated during the design of the environment model: (i) agent implementations may be independent from the environment implementation; and (ii) environment model may be modular to plug and unplug components as required by applications.

3 JaSIM Environment Model and Platform

JaSIM environment model is a multiagent situated environment model dedicated to the simulation in 1D, 2D and 3D virtual environments. It is mainly designed for traffic and crowd simulations. JaSIM platform is directly designed as an extension of the Janus platform [9] combining organisational agent-oriented approach from Janus with object-oriented structure of the JaSIM simulation model. This model is designed to provide realistic perception and action mechanisms to agents. It is a framework[4] providing a collection of well-known design-patterns for an easy and reusable implementation of traffic and crowd simulation.

This section is structured as follows. Subsections 3.1 describes the overall architecture of the JaSIM platform and the various features provided by each module. Subsection 3.2 describes the key principles and concepts of the JaSIM simulation model.

3.1 Overall Architecture

The overall architecture of the JaSIM platform is shown in Figure 1. JaSIM is developed in Java 1.6 and composed of four main modules representing a total of 88,805 lines of code. The various features provided by each module are briefly described below:

- Janus *Wrapper* module is an extension of the Janus platform, and consitutes the agent-oriented part of the JaSIM platform. It provides a collections of organisations and agent architectures dedicated to simulation. It

[4] according to the design-pattern definition of "framework" [8]

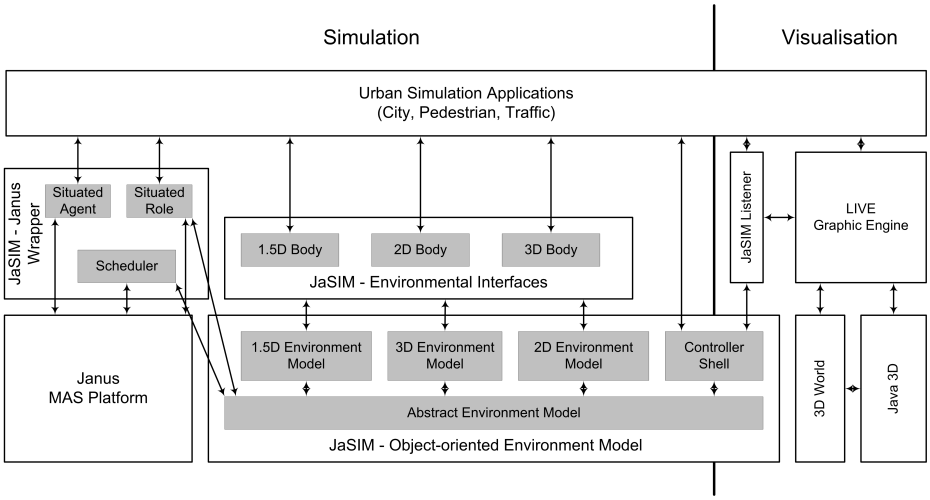


Fig. 1. Overall Architecture of JaSIM-Janus MABS Platform

also extends Janus agent and role concepts to speed up the development of multiagent-based simulation. A collection of tools to manage agent scheduling and probing is also provided by this module.

- *OO Environment Model* is the main part of the JaSIM platform and provides all necessary tools to model and manage creation and modification of environmental structures. It contains in particular all the hierarchical and graph structures used to manage agent's perceptions. This module is more detailed in the next subsection.
- *Environmental Interfaces* enable connection between a specific application and JaSIM platform. They provide a full set of agent's body architectures for various types of virtual environments. Each kind of body provides to application agents a collection of tools for perceiving its environment and emitting influences.
- *Listener* module is intended to provide a common interface for managing event-based communication with external tools and graphical user interfaces. A specific feature of this module enables network-based communication with a remote application. It is used for managing communication with the *Live* graphics engine[5].

3.2 JaSIM Environment Model

According to Russel and Norvig [19], JaSIM environment model may be considered as inaccesible, non-deterministic, dynamic, and continuous. This model is strongly inspirated by the *Influence-Reaction* approach [7, 13].

[5] Live is developed by the Systems and Transport Laboratory

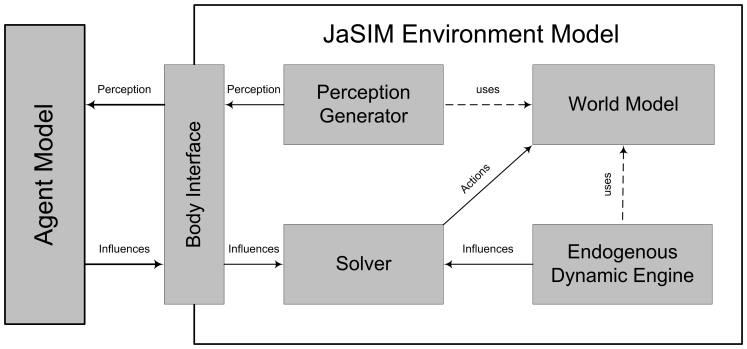


Fig. 2. The JaSIM Environment Model

The JaSIM environment model does not make any assumption about the multiagent platform used to execute and design agents. Currently the Janus platform[6] and the TinyMAS platform[7] are usable. The only one requirement to support another platform is to develop a piece of code which is (i) creating an agent instance, and binding it to an environmental body; and (ii) providing a scheduling policy which is compatible with the environment execution policy.

As the JaSIM model should be usable even for pedestrian and vehicle simulations, it is primilary defined with the common features for both types of simulation. Figure 1 shows the global architecture of an application running the JaSIM model.

In this section, the JaSIM environment model is described as a design pattern which is applied to produce an executable platform for all supported dimensions ($1D$ to $3D$, in section 3.3). The JaSIM model assumes that:

- an agent owns a body which is its representation in the physical environment, this body is part of the environment;
- a body provides a set of sensors (frustums...) and effectors to its associated agent;
- a body may provide active perceptions;
- an agent can perceive or act on the physical environment only through its body;
- an agent can not directly change the state of the environment to avoid temporal conflicts of the actions; and
- the environment has its own endogenous dynamics that may be specialised according to application requirements.

According to these assumptions the JaSIM model is designed with two "*pipelines*": the first one computes the agent's perceptions from the world hierarchical or graph-based structures (or *world model* thereafter), and the second

[6] http://www.janus-project.org
[7] http://www.arakhne.org/tinymas

pipeline collects the agent's influences and computes the resulting reactions on the world model (Figure 2). All these components are detailed in the following subsections, and they are defined in the UML class diagram in Figure 3.

Perceivable, *Influencable* and *Influencer* interfaces describes the main roles of the environmental components according to the *Influence-Reaction* approach. *Perceivable* interface is implemented by all the entities which may be perceived by agents. Entities which may be the target of an agent influence must implement *Influencable*. And *Influencer* interface is implemented by the bodies of influence emitters.

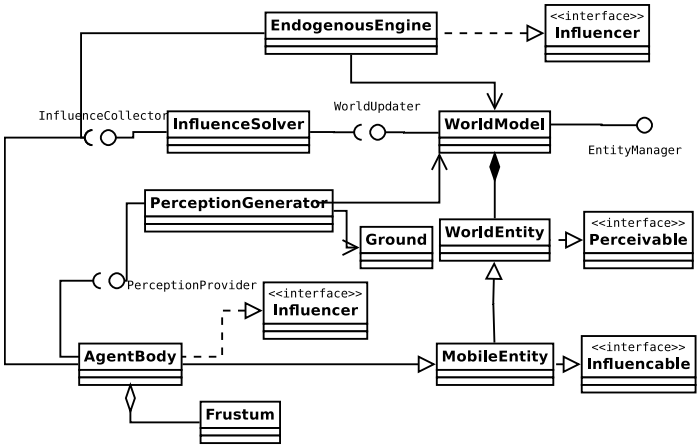


Fig. 3. Simplified UML class diagram for the JASIM environmental model

3.3 World Model

The model of world contains all data structures required to store and describe the entities — an agent body is a kind of entity — in the environment. The world model is divided into two main groups of data structures: (i) the ground descriptions, and (ii) the object descriptions.

Ground Model This model is mainly used by the perception pipeline and the "keep-on-floor" heuristic. A ground description is basically a discrete heightmap which is able to quickly compute the height of each continuous point by using tri-interpolation equations. More formally, the ground is defined by a surjective function $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ where the parameters are the 2D coordinates of a point on the ground and the replied value is the interpolated-height. The implementation of the ground function may be based on a regular or an irregular grid [16, 3].

Ground is also used to quickly detect non-traversable areas by the influence-reaction pipeline presented below.

Object-Space Model An object-space model is able to quickly provide information on the location and the orientation of each entity in the world, and to manipulate them. Here the dimension of the space is a important choice. JaSIM environment model is developed to support $1D$, $1\frac{1}{2}D$, $2D$, and $3D$ dimensions. Each implementation provides a specific world model to improve query performances, and respecting the environment's mission M1.

The $1-$dimension world model is based on a directed or non-directed graphs (depending on GIS imported data) on which each entity is located according to a curvilinear abscissa x. The orientation of the entity may be colinear to the graph's segment or defined by a rotation angle [14]. The $1\frac{1}{2}-$dimension world model is an extension of the $1-$dimension model. Each entity is located with its curvilinear abscissa x and a shifting distance y from the graph's segment (along the normal segment). These two dimensions are assumed to be out of the scope of this paper.

The $2-$ and $3-$dimension world models are similar and uses equivalent data structures: spatial trees. According to [20] and for performance reasons, the JaSIM environment model is using two trees: (i) the first one is containing the immobile entities, and (ii) the second, the mobile entities. Mobile entities may be agent's bodies or non-autonomous objects.

A spatial tree uses a partition heuristic to hierarchically decomposed the space into subspaces. The JaSIM model is providing different implementations of well-known trees traditionally used in Game and Serious-Game theories: binary space partition (BSP) tree, quadtree, and octree. They respectively divide the space into two, four and eight subspaces. In JaSIM the cutting planes[8] are assumed to be parallel to the world's axis by default. This choice is introduced to simplify and speed up the building process of the trees.

Two key points have still a great influence on the access/query performances of the trees: (i) the position of the cutting planes, and (ii) the heuristic used to classify the entities which are intersecting one or more cutting planes. The cutting planes may be located at the center of the space to partition or at the barycenter of the entities located in that subspace. Four possibilities are theoritically available to classify an entity which is intersecting one or more cutting planes:

1. split the entity mesh into several parts and put each of them into a child node;
2. clone the entity and put the clones into the child nodes;
3. put the entity in the current node, not in the child node;
4. create a new specific child node — icosep node [20] — and put the intersecting entity into.

[8] A cutting plane is a plane which is separating two adjacent subspaces.

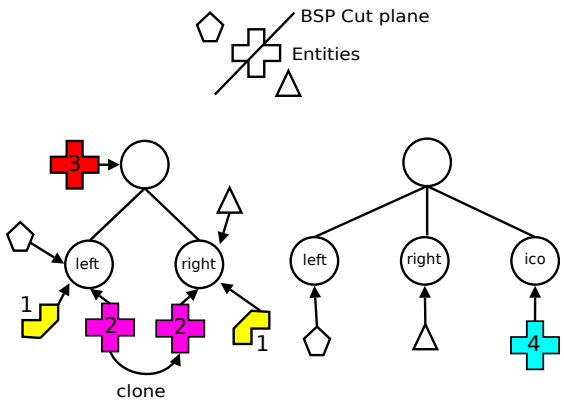


Fig. 4. Cutting plane heuristics on a BSP example

Figure 4 illustrates these four approaches. In this example three entities are located in space; the upper part of the figure illustrates the locations of three entities and the cutting plane. The lower-left tree is a standard BSP tree. According to the cutting plane classification, the hexagon and the triangle are respectively located in the left and right child. As the cross is located on the cutting plane, it is classified according to the previous approaches — the number near/on the cross identifies the selected case. The lower-right tree is a BSP tree using the icosep heuristic. In this case the intersecting entity is directly put into the icosep node.

Cases 1 and 3 are too much time consuming: continously splitting and merging the entity meshes firstly, and secondly retreiving all the entity clones in the tree harm the initial logarithmic complexity of tree queries. Case 4 is prefered to Case 3 because it enables to recursively partition the *icosep node* according to a dedicated heuristic. In [20] each half-cutting plane has its own icosep node but this last is not splitted in turn. This may lead to big tree nodes that contain an important number of objects. Traversing such nodes significantly increases the overall computational cost of a tree query.

A key point in our world model is that the entities are represented by bounding boxes and simplified meshes. The bounding boxes are used as default volumetric representation of the entities. Meshes are only used to compute the bounding boxes or to support the occlusion culling. Bounding boxes are basicaly used in 3D rendering to preserve computational time because they provide quick intersection tests.

3.4 Perception Pipeline

The perception pipeline aims at computing a set of perceptions for each agent (environment mission M3) according to the environment's laws and agent's preferences specified in its body.

In this paper, we refer to perception as a generic term which encapsulate all the classical human senses. The current implementation of the JaSIM model is providing two cascading vision cullers (frustum culler and occlusion culler). The smell, hearing, taste and touch perceptions are not provided yet.

Visual perception is firstly based on a frustum culler which selects the entities that are partly or entirely inside the agent's frustum. They are selected by recursively traverse the tree nodes in intersection with the frustum. Bounding boxes permit to quickly exclude the nodes — consequently the child nodes — which have no possible intersection with the frustum. Finally from each selected nodes, entities are matched against the frustum and replied if they have an intersection. The frustum culler is able to quickly determine the type of intersection: outside, entirely inside, partly inside, or enclosing the frustum. The complexity of a culling query is $O(\log n)$ where n is the count of nodes in the tree, experimental results are given in table 1.

Then, an occlusion culler is used to determine if an entity is partly or completely occluded by another one, or not. The occlusion culler uses a depth-buffer approach [1], well-known in 3D rendering. As this culler is very time consuming, it is not activated by default.

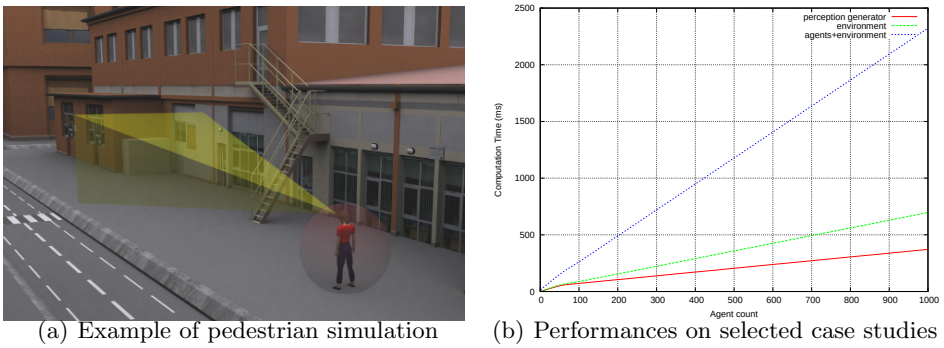
3.5 Influence-Reaction Pipeline

The influence-reaction pipeline is inspirated by [13] and manages environment's mission M2. First, an influence collector is collecting all the influences emitted by the agents through their bodies, or by the environment itself. When all the influences are collected, the environment is trying to detect conflicts according to its internal laws. For each nonconflicting influence a reaction is directly computed. For conflicting influences a reaction is computing according to resolution heuristics. The JaSIM model uses a "keep-on-floor" heuristic, which forces the bodies to be on the ground, and a collision detection algorithm [5].

3.6 Body Interface

The body interface is the mode used by the agents to interact with the physical space. In addition to the definition of the physical location, the orientation and the shape of an agent, the body acts as the filter between the agent model and the environment model. It owns various perception frustums. The JaSIM environment model is providing three predefined types of frustums: a sphere, a truncated pyramid, and a pedestrian frustum — a frustum composed by a sphere (short-range perception) and a truncated pyramid (long-range perception). The agent may put in its body several functions which are acting as filters on the perceptions. These functions, also named "active perception functions", enable to speed up the selection of the perceived entities. A body has also a set of kinematic properties updated by the environment (acceleration, speed...) according to the reactions to the influences.

Finally, a body is able to convert data from the environment's format to the agent's desired format. This last point permits in particular to implement an



(a) Example of pedestrian simulation (b) Performances on selected case studies

Fig. 5. Study Case Experiments

agent model with a given space dimension ($1D$ to $3D$) independently from the dimension of the environment. Data conversion is done thanks to convertion rules defined in the bodies. These rules are based on the mathematical transformations of the points and the vectors from a coordinate system to the other. For example, from the $3D$ left-handed coordinate system with z axis up to the $2D$ right-handed coordinate system, the point $\langle 1, 2, 3 \rangle$ is transformed to $\langle 1, -2 \rangle$.

3.7 Endogenous Dynamics

The environment has also its intrinsic dynamic [11], and it may evolve beyond the agent control. This phenomena is modelled by the endogenous dynamics engine in JaSIM (mission M4). All the physics laws from the real world may be implemented in this engine. The JaSIM implementation is able to use the NVidia PhysX®[9] and the ODE[10] physics engines well-known in 3D real-time applications.

4 Experimental Results

The JaSIM platform is deployed and tested on two case studies. The goal of these experiments is to compare the performances of the different modules of the environment model against the size of the agent population. The first study case is related to a crowd simulation during a famous european rock festival which is taking place near Belfort. The second case study is a simulation of pedestrians in a Belfort's industrial area. Both case studies focus on bottleneck detection and flow quantification. They are generated from real data sets (geographical information systems, buildings photos...). Figure 5(a) illustrates the second case study with a pedestrian and its pedestrian frustum.

[9] http://www.nvidia.com/object/physx_new.html
[10] http://www.ode.org

	Frustum		
	Sphere	Pyramid	Pedestrian
BSP	$\mu = 0.02$ $\sigma = 0.01$	$\mu = 0.04$ $\sigma = 0.01$	$\mu = 0.04$ $\sigma = 0.01$
Icosep-BSP	$\mu = 0.02$ $\sigma = 0.01$	$\mu = 0.03$ $\sigma = 0.01$	$\mu = 0.04$ $\sigma = 0.01$
QuadTree	$\mu = 0.02$ $\sigma = 0.01$	$\mu = 0.04$ $\sigma = 0.01$	$\mu = 0.04$ $\sigma = 0.01$
Icosep-QuadTree	$\mu = 0.03$ $\sigma = 0.01$	$\mu = 0.04$ $\sigma = 0.01$	$\mu = 0.04$ $\sigma = 0.01$

Table 1. Numerical results for one frustum culling query – average μ and standard deviation σ in ms

Benchmarks are run on Linux operating system with a 1.2GHz dual-core processor and 2Go of RAM. Simulation is running under the 1.6 Sun's Java virtual machine with 64Mo of allocated memory. Agent models are implemented as holons on the Janus platform. Figure 5(b) illustrates the performances of the simulation.

Figure 5(b) shows that the major part of the environment's computation time is consumed by the perception generator. Indeed, for each agent's body in the environment a visual perception is computed. The complexity of the perception generator is $O(k.m \log n)$ where k is the count of agent's bodies in the environment, $\log n$ is the complexity of a query on the trees when it is reasonably balanced. m is the factor introduced by the current implementation of JaSIM. The poor performances of the perception generator implementation is currently due to internal copies of the perception collections before passing them to the agents. Table 1 contains the computation times for the queries on the three types of tree with the three types of supported frustums. The influence of each data structure on the overall simulation time is limited and follows a logarithmic curve.

5 Conclusion

Situated environment is a key concept in MABS, however it remains barely considered as a first-class entity. JaSIM environment model gathers the environment definition from MAS literature, the *Influence-Reaction* approach, with the standard data structures used in real-time 3D rendering. Combined with Janus, JaSIM is able to simulate crowd and traffic flows with 3D-based senses and actions. This paper describes the platform architectures and illustrates its proof of concept on two case studies. The platform is successfully used on the simulation of a rock festival (up to 5,000 pedestrians), industrial area traffic (1,000 vehicles), and Orly airport hall (3,000 pedestrians). In the opposite of Massive or Breve, JaSIM is not focussed on the simulation of the human bodies and there physical interactions. Its first purpose is to provide efficent, accurate

and realistic perceptions to numerous agents. Another key point of JaSIM is its ability to merge different points of view (pedestrian, vehicle, cycle...) in the same simulation model.

To improve the support of large-scale simulations (more than 5,000 entities) by the platform, multilevel models will be introduced both in the agents and environment models. In conjunction with environmental topological hierarchies it should permit to enlarge the number of simulated entities as well as the size and complexity of the simulated space. Relationships between the different points of view (vehicle, pedestrian, economical entities...) of the system will be also studied and improved in future works.

Acknowledgement. The JaSIM platform is distributed as part of the SIMULATE commercial offer of the Voxelia SAS[11] company, France. The authors would like to thank Mickael Goncalves, Renan Zeo and Olivier Lamotte for their support and contributions.

Bibliography

[1] Thomas Akenine-Möller, Eric Haines, and Natty Hoffman. *Real-Time Rendering*. A.K. Peters, Ltd., Natick, MA, USA, 3rd edition edition, 2008.

[2] S. Brueckner. *Return from the Ant*. Computer science, Humboldt-Universität, Berlin, Germany, 2000.

[3] L. De Floriani and P. Magillo. Regular and Irregular Multi-Resolution Terrain Models: a Comparison. In A. Voisard and S.-C. Chen, editors, *10th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS'02)*, pages 143–148, McLean, VA, November 2002.

[4] Stephane Donikian. Vuems: A virtual urban environment modeling system. *Computer Graphics International Conference*, 0:84, 1997. doi: http://doi.ieeecomputersociety.org/10.1109/CGI.1997.601278.

[5] Christer Ericson. *Real-Time Collision Detection*. Interactive 3D Technology. Morgan Kaufmann, 2004.

[6] Nathalie Farenc, Ronan Boulic, and Daniel Thalmann. An informed environment dedicated to the simulation of virtual humans in urban context. In *Proceedings of EUROGRAPHICS99*, pages 309–318, 1999.

[7] J. Ferber, F. Michel, and J. Baez. AGRE: Integrating environments with organizations. In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *Third International Workshop (E4MAS 2006)*, volume 4389 of *Lecture Notes in Artificial Intelligence*, pages 48–56. Springer, Hakodate, Japan, may 2006.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing. Addison-Wesley Professional, November 1994.

[9] Nicolas Gaud, Stéphane Galland, Vincent Hilaire, and Abderraffíâa Koukam. An Organisational Platform for Holonic and Multiagent Systems. In *PROMAS-6@AAMAS'08*, Estoril, Portugal, May 12-16th 2008.

[10] C. Greenhalgh and S. Benford. Massive: a distributed virtual reality system incorporating spatial trading. In *ICDCS '95: Proceedings of the 15th International*

[11] http://www.voxelia.com

Conference on Distributed Computing Systems, page 27, Washington, DC, USA, 1995. IEEE Computer Society.

[11] A. Helleboogh, G. Vizzari, A. Uhrmacher, and F. Michel. Multi-Agent Modeling and Simulation: Dynamis in the Environment. *Journal of Autonomous Agents and Multi-Aent Systems (JAAMAS)*, 14(1):87–116, feb 2007. ISSN 1387-2532.

[12] Jon Klein. breve: a 3d environment for the simulation of decentralized systems and artificial life. In *ICAL 2003: Proceedings of the eighth international conference on Artificial life*, pages 329–334, Cambridge, MA, USA, 2003. MIT Press. ISBN 0-262-69281-3.

[13] Fabien Michel. The IRM4S model: the influence/reaction principle for multiagent based simulation. In *Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS07)*. ACM, May 2007. ISBN 978-81-904262-7-5. doi: http://doi.acm.org/10.1145/1329125.1329289.

[14] Ian Millington. *Artificial Intelligence for Games*. Interactive 3D Technology. Morgan Kaufmann, San Francisco, CA, 2006.

[15] J. Odell, H.V.D. Parunak, M. Fleisher, and S. Brueckner. Modeling Agents and their Environment. In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Agent-Oriented Software Engineering III*, volume 2585 of *Lecture Notes In Computer Science*, N.Y. (USA), 2002. Springer-Verlag.

[16] Renato Pajarola and Enrico Gobbetti. Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, 23(8):583–605, aug 2007.

[17] Y. Papelis and S. Bahauddin. Logical modeling of roadway environment to support real-time simulation of autonomous traffic. In *SIVE95: the First Workshop on Simulation and Interaction in Virtual Environments*, volume 1, pages 62–7, Iowa, Iowa City, U.S.A., July 1995.

[18] Julien Pettré, Pablo de Heras Ciechomski, Jonathan Maïm, Barbara Yersin, Jean-Paul Laumond, and Daniel Thalmann. Real-time navigating crowds: scalable simulation and rendering: Research articles. *Computer Animation and Virtual Worlds*, 17(3-4):445–455, 2006. ISSN 1546-4261.

[19] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition edition, 2003.

[20] Joshua Shagam. *Dynamic spatial partitioning for real-time visibility determination*. Computer science, New Mexico State University, Department of computer science, April 2003.

[21] Wei Shao and Demetri Terzopoulos. Autonomous Pedestrians. *Graphics Models*, 69(5–6):246–274, sep 2007. Special Issue on SCA 2005.

[22] Gwenola Thomas. *Environnements virtuels urbains : modélisation des informations nécessaires la simulation de piétons*. PhD thesis, Informatique, Université de Rennes 1, 16 décembre 1999.

[23] D. Weyns and T. Holvoet. A Formal Model for Situated Multiagent Systems. *Special Issue of Fundamenta Informaticae*, 63(2), 2004.

[24] Danny Weyns, H. Van Dyke Parunak, Fabien Michel, Tom Holvoet, and Jacques Ferber. Environments for Multiagent Systems State-of-the-Art and Research Challenges. In Danny Weyns, H. Van Dyke Parunak, and Fabien Michel, editors, *Third International Workshop (E4MAS 2006)*, volume 4389 of *Lecture Notes in Artificial Intelligence*, pages 1–47. Springer, Hakodate, Japan, may 2006.

[25] Bernard P. Zeigler, Tag Gon Kim, and Herbert Praehofer. *Theory of Modeling and Simulation*. Academic Press, 2nd edition edition, 2000.