A Framework for Interacting Situated Agents in Virtual Environments

Giuseppe Vizzari[a], Giorgio Pizzi[a], Flávio Soares Corrêa da Silva[b]

***Abstract*—This paper presents a framework supporting the definition and implementation of virtual environment inhabited by interacting situated agents defined according to the Multilayered Multi-Agent Situated System model. The framework supports the specification and execution of visually rich 3D virtual environment endowed by the presence of mobile agents acting and interacting inside it according to a multi-agent model. The paper briefly describes the related works and possible application scenarios for the framework, then it introduces the multi-agent model underlying the framework and its basic architecture. Sample applications are also described so as to show the potential of the framework in executing models comprising several hundreds of agents producing an effective visualization of the generated dynamics.**

***Index Terms*— multi-agent systems, virtual environments, simulation, 3D visualization**

I. Introduction

The design and realization of virtual environments inhabited by social entities is a significant application of the conjoint results of various research areas in computer science and engineering. Virtual environments have been exploited in several ways, and in particular:

- to support computer mediated forms of human interaction, characterized by the introduction of Embodied Conversational Agents facilitating users' interactions [18] or supplying awareness information in a visually effective form [20];
- to realize operational laboratories for participatory design, supporting the effective visualization of various alternative design choices to the involved stakeholders [8][13][11];
- to provide effective instruments for the modeling, simulation and visualization of the dynamics of entities situated in a representation of an existing, planned or reconstructed environment or situation [10][19];
- for sake of entertainment, in movies, computer games or in online communities (see, e.g., Second Life[1]).

While all these applications are characterized by a strong requirement for realistic and effective visualization tools (and some of them require a thorough analysis of the system usability, due to the necessary accessibility by non-technically skilled users), they also call for expressive models supporting the specification of behaviours for the entities that inhabit these environments, as well as the interaction among them and with the environment itself. The fact that the overall performance of the system is essentially dependant on the single actions and interactions that are carried out by entities inhabiting the modeled environment leads to consider that the Multi-Agent Systems [12] approach is particularly suited to tackle the modeling issues that are posed by this scenario. This idea is also corroborated by the fact that most of the above introduced references actually describe systems based on this approach, and by specific experiences in applying MAS approaches to specific virtual environments applications such as computer games [16].

In this vein, the main aim of this paper is to show the current advancement of a long term project that provides the realization of a framework supporting the development of MAS based simulations based on the Multilayered Multi-Agent Situated System model provided with an effective form of 3D visualization. The main goal of the framework is to support a smooth transition from the definition of an MMASS based model of given situation (in terms of environment, relevant entities and their behaviours, expressed as individual actions interactions) to the realization of simulation systems characterized by an effective 3D user interface. One of the possible application areas of this kind of system is related to the modeling and simulation of crowds of pedestrians to support architectural design or urban planning [3][4]. In order to have information flowing appropriately from the formal model to design professionals (e.g. architects and urban planners), the MMASS-based simulator must be supported by adequate visualization and animation tools. Such supporting tools are the core issue of the present paper.

The paper breaks down as follows: the following section discusses related works in different application scenarios, while section III briefly introduces the MMASS model and its application to model pedestrians situated and moving in representations of physical spaces. Section IV discusses the architecture of the proposed framework, its main components and the tools supporting developers adopting it. Section V presents two sample applications aimed at showing the potential of the framework in executing models comprising several hundreds of agents producing an effective visualization of the generated dynamics. Conclusions and future developments end the paper.

[a]Complex Systems and Artificial Intelligence research center, University of Milano-Bicocca, via Bicocca degli Arcimboldi 8, 20126 Milano, {giuseppe.vizzari, giorgio.pizzi}@csai.disco.unimib.it

[b]Department of Computer Science, Institute of Mathematics and Statistics, Universidade de São Paulo, fcs@ime.usp.br

[1] http://secondlife.com

II. Related works and Application scenarios

The realization of virtual environments inhabited by autonomous entities and characterized by a realistic three-dimensional form of visualization was the goal of several projects, both commercial and academic, with different aims, features and available documentation. A complete and thorough description of the state of the art in this area, besides being extremely difficult to realize, is out of the scope of this paper; we will rather briefly report the survey activity that was carried out before starting the project and that motivated the effort related to the design and realization of the framework.

The main aim of the research effort is to realize an instrument that, on one hand, supports an effective form of visualization of a virtual environment and, on the other, allows the specification of the behaviours of the autonomous entities that inhabit it in terms of an expressive agent based model. To this purpose, we considered several possible supporting instruments, both commercial and open source, both providing a basic enabling technology and also some projects providing a support to the phases of modeling and definition of agents behaviours.

Some relevant representatives of the category of commercial instruments that can support the design and development of virtual environments are Quadstone Paramics[2] and Massive[3]. Paramics is a traffic micro-simulation software, that is able to generate realistic 3D visualizations of the simulated dynamics. Massive is instead an application specifically devoted to the generation of photorealistic animation of crowd-related visual effects (it was adopted for several films – e.g. to generate the battles of the Lord of the Ring – and commercials – e.g. to create the audience in a stadium). These instruments are generally extremely focused, very powerful, but their internal mechanisms are not well documented. In general they cannot be adapted to tackle application scenarios different from those they were originally conceived for; therefore they do not provide the degree of flexibility required by our project.

Some open source efforts were also considered, and two relevant representatives of the analyzed platforms are Mason[4] [15] and Breve[5] [14]. Mason is a discrete-event multi-agent simulation library, designed to be the foundation for custom-purpose Java simulations. It also includes an optional suite of visualization tools in 2D and 3D. However, this suite does not represent a proper support to the realization of a virtual environment, but rather a library for realizing simple 3D visualization of the simulated system. Breve, on the other hand, is a software package enabling the definition of 3D simulations of multi-agent systems and artificial life. It adopts a specific ad-hoc language (called "steve") for the specification of agents' behaviours. However, it is more focused on providing an abstract and extremely simplified 3D environment for specifying and testing multi-agent models and artificial life models, rather than supporting the realization of a detailed virtual environment.

The analyzed system that was closest to meet our requirements is Freewalk[6], an application that was adopted in some of the previously cited applications of virtual environments. It adopts a scenario specification language, called Q, that supports the specification of the environment (that must be a VRML model) and the behaviours of agents (through the notion of *scenario*). Freewalk, however, is not an open-source project and the Q language is not very focused on the interaction among the agents and the environment (that can be conceived as an element having an influence on their behaviour that goes beyond the fact that it provides obstacles to their movement). These considerations lead us to consider the possibility to adopt a basic enabling infrastructure for an effective visualization of system dynamics and an existing model - MMASS [1] - and platform for the specification of situated MAS supporting the definition of virtual environments. The MMASS model is in fact characterized by the fact that agents' environment is a first class element of the model, and it deeply influences agents' perceptions and actions, supporting forms of interactions that are particularly suited to represent the movement of pedestrians in physical spaces [2].

III. Multilayered multi-agent situated system model

This section will introduce the basic elements of the MMASS model and its application to represent and manage mechanisms of interaction between the environment and active autonomous entities that are useful for the specification of dynamic virtual environments. We will start discussing the elements of a single layered model, a Situated Cellular Agents model, then we will show how it can be applied to represent physical environments and active entities situated and moving in it. The last subsection will discuss how a multilayered structure can be adopted to enhance the model and support more autonomous forms of agents' behaviours.

A. Situated Cellular Agents

A system of Situated Cellular Agents can be denoted by the three-tuple $\langle Space,F,A\rangle$ where *Space* is a single layered environment where agents are situated, act autonomously and interact by means of reaction or through the propagation of fields belonging to the set *F*. Field based interaction is an indirect interaction mechanism that provides a modification of agents' environment that can be perceived by agents according to their context and state; a more thorough description of field based interaction can be found in [17], whereas the specific SCA field-based interaction model is discussed in [6]. Agents belong to *A*, a finite set of agents, each characterized by a type determining their state, perceptive capabilities and behavioural specification. The elements of this three tuple will now be formally described.

Space - The *Space* consists of a set *P* of sites arranged in a network (i.e. an undirected graph of sites). Each *site* $p \in P$ can contain at most one agent and is defined by $\langle a_p,F_p,P_p\rangle$

[2] http://paramics-online.com/
[3] http://www.massivesoftware.com/
[4] http://cs.gmu.edu/~eclab/projects/mason/
[5] http://www.spiderland.org/
[6] http://www.ai.soc.i.kyoto-u.ac.jp/freewalk/

where $a_p \in A \cup \{\perp\}$ is the agent situated in p ($a_p = \perp$ when no agent is situated in p, in other words p is empty); $F_p \subseteq F$ is the set of fields active in p ($F_p = \emptyset$ when no field is active in p); and $P_p \subset P$ is the set of sites adjacent to p. Edges connecting sites represent a constraint to the movement of agents situated in the environment and also on the diffusion of fields, which only propagate through these connections.

Fields - A field $f_\tau \in F$ that can be emitted by agents of type τ is denoted by the four-tuple $\langle W_\tau, Diffusion_\tau, Compare_\tau, Compose_\tau \rangle$ where:

- $W_\tau = S \times N$, where $S \subseteq \Sigma_\tau$, denotes the set of values that the field can assume; given $w_\tau \in W_\tau$, $w_\tau = \langle s_\tau, i_\tau \rangle$, where $s \in S$ represents information brought by the field (i.e. the field payload) and $i_\tau \in N$ represents its intensity.
- $Diffusion_\tau : P \times W_\tau \times P \rightarrow W_\tau$ is the diffusion function for field type τ; $Diffusion_\tau(p_s, w_\tau, p_d)$ computes the value of a field on a given destination site (p_d) taking into account in which site it was emitted (p_s) and with which initial value ($w_\tau \in W_\tau$).
- $Compare_\tau : W_\tau \times W_\tau \rightarrow \{True, False\}$ is the function that compares field values. It is used by the perceptive system of agents to evaluate if the value of a certain field type is such that it can be perceived.
- $Compose_\tau : (W_\tau)^+ \rightarrow W_\tau$ expresses how field values of the same type have to be combined in order to obtain the unique value of a field type at a given site.

Agent Types - The possibility to define different agent types introduces heterogeneity, in other words the chance to define different abilities and perceptive capabilities. Defining T the set of types, it is appropriate to partition the set of agents in disjoint subsets corresponding to different types. The set of agents can thus be defined as $A = \bigcup_{\tau \in T} A_\tau$ where $A_{\tau_i} \cap A_{\tau_j} = \emptyset$ for $i \neq j$. An agent type τ is defined by the three tuple $\langle \Sigma_\tau, Perception_\tau, Action_\tau \rangle$ where:

- Σ_τ defines the set of states that agents of type τ can assume;
- $Perception_\tau : \Sigma_\tau \rightarrow [N \times W_{f_1}] ... [N \times W_{f_{|F|}}]$ is a function associating to each agent state the vector of pairs representing respectively a receptiveness coefficient modulating the intensity of that kind of field and a sensitivity threshold represented by a specific field value; these functions represent the perceptive capabilities specification for that type of agent and their usage will be clarified in the description of agents and their behaviours. Formally, this vector of pairs is defined as

$$\left(c_\tau^1(s), t_\tau^1(s)\right), \left(c_\tau^2(s), t_\tau^2(s)\right), ..., \left(c_\tau^{|F|}(s), t_\tau^{|F|}(s)\right)$$

where for each i ($i = 1...|F|$), $c^i_\tau(s)$ and $t^i_\tau(s)$ express respectively a receptiveness coefficient to be applied to the field value f_i and the agent sensibility threshold to f_i in the given agent state s.

- $Actions_\tau$ denotes the set of actions that agents of type τ can perform, and will be described in the following.

Agents and their Behaviours - An agent $a \in A$ is defined by the three–tuple $\langle s, p, \tau \rangle$, where:

- $s \in \Sigma_\tau$ denotes the *agent state* and can assume one of the values specified by its type;
- $p \in P$ is the site of the *Space* where the agent is situated;
- τ is the *agent type*, which provides the allowed states, perceptive capabilities and behavioural specification for that type of agents.

The first two elements were previously introduced, we will now focus on $Action_\tau$, which is made up of a set of actions and an action selection strategy. Actions can be selected from a set of primitives which include *reaction* (synchronous interaction among adjacent agents), field *emission* (asynchronous interaction among at–a–distance agents through the field diffusion–perception–action mechanism), *trigger* (change of agent state as a consequence of a perceived event) and *transport* (agent movement across the space). The two interaction mechanisms provided by the SCA model (i.e. reaction and field–based interaction) are also described by the diagram in Figure 1. Every primitive will be now briefly described specifying preconditions and effects. It must be noted that an action selection strategy is invoked when the preconditions of more than one action are verified; several possible strategies can be defined, but in this context a non–deterministic choice among possible action was adopted.

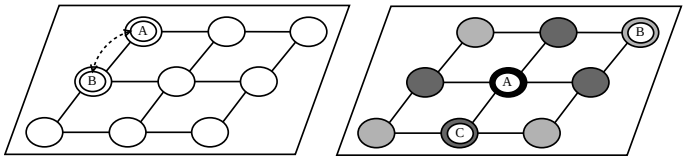

Figure 1 – A diagram showing the two interaction mechanisms provided by the SCA model: two reacting agents on the left, and a field emission on the right.

The behavior of Situated Cellular Agents is influenced by agents situated on adjacent positions and, according to their type and state agents are able to synchronously change their states. Synchronous interaction (i.e. reaction) is a two–step process. Reaction among a set of agents takes place through the execution of a protocol introduced in order to synchronize the set of autonomous agents. When an agent wants to react with the set of its adjacent agents since their types satisfy some required condition, it starts an *agreement* process whose output is the subset of its adjacent agents that have agreed to react. An agent agreement occurs when the agent is not involved in other actions or reactions and when its state is such that this specific reaction could take place. The agreement process is followed by the synchronous reaction of the set of agents that have agreed to it. Reaction of an agent a situated in site $p \in P$ can be specified as:

action:*reaction*($s, a_{p1}, a_{p2}, ..., a_{pn}, s'$)
condition:*state*(s),*position*(p),*agreed*($a_{p1}, a_{p2}, ..., a_{pn}$)
effect:*state*(s')

where *state*(s) and *agreed*($a_{p1}, a_{p2}, ..., a_{pn}$) are verified when the state of agent a is s and agents situated in sites $\{p_1, p_2, ..., p_n\} \subseteq P_p$ have previously agreed to undertake a synchronous reaction. The effect of a reaction is the synchronous change in state of the involved agents; in particular, agent a changes its state into s'.

Other possible actions are related to the indirect interaction mechanism, related to field emission and to the perception–deliberation–action mechanism. Agent emission can be defined as follows:

action:emit(s,f,p)
condition:state(s),position(p)
effect:added(f,p)

where *state*(*s*) and *position*(*p*) are verified when the agent state is *s* and int position is *p*. The effect of the emit action is a change in the active fields related to sites involved in the diffusion, according to $Diffusion_f$. One of the possible effects of an agent perception of a certain field f_i can be defined as

action:trigger(s,f_i,s')
condition:state(s),position(p),perceive(f_i)
effect:state(s')

where *perceive*(f_i) is verified when $f_i \in F_p$ and $Compare_\tau(c^i_\tau \cdot i_{f_i}, t^i_\tau) = true$ (in other words, field intensity modulated by a receptiveness coefficient exceeds the sensitivity threshold for that field). The coefficients c^i_τ and t^i_τ are those determined by the perception function for that type of agent in the state *s*. The effect of the trigger action is a change in agent's state according to the third parameter.

The last possible action for an agent causes a change in its position and can be specified as follows:

action:transport(p,f_i,q)
condition:position(p),empty(q),near(p,q),perceive(f_i)
effect:position(q),empty(p)

where *empty*(*q*) and *near*(*p*,*q*) are verified when $q \in P_p$ and $q = \langle \perp, F_q, P_q \rangle$ (*q* is adjacent to *p* and it does not contain agents). The effect of a transport action is thus to change the position of the related agent.

B. *Modeling Crowds with SCAs*

The basic idea underlying the application of the SCA model to represent environments and entities situated and moving in it is that this kind of movement can be generated by means of attraction and repulsion effects (as also suggested in [9]). These effects are generated by means of fields that can be emitted by specific point of the environment, and that can be perceived as attractive/repulsive or that can even be simply ignored by different types of moving entities in specific states. Also pedestrians themselves are able to emit fields and thus, in turn, they can generate attraction/repulsion effects, and what is called an 'active walker' model.

A thorough discussion of this modeling approach is out of the scope of this paper and it can be found in [2], we will now just give some indications of the main steps that must be followed to define a SCA model starting from an abstract description of a given scenario.

Definition of the **spatial infrastructure** of the environment – a SCA space can represent a discrete abstraction of a physical environment, in which a site corresponds to a portion of space that can be occupied by a pedestrian. For instance, a corridor and the rooms having a door on it could be discretized in 40cm^2 cells characterized by a Von Neumann adjacency.

Definition of **points of interest/reference** in the environment – specific spots of the environment can represent elements of interest, reference points or constraints (e.g. gateways, doorways) influencing pedestrian movements. These elements must be associated with immobile agents (e.g. door jambs) able to emit fields indicating the presence of the point of interest/reference to pedestrians. For instance, considering a corridor the exits should be associated to suitable fields able to guide agents towards them, but also possible doorways leading to rooms should be provided with agents emitting proper fields.

Definition of **mobile entities** of the environment (pedestrians) – the different types of mobile entities, agents representing pedestrians, can be now defined in terms of attitudes towards the movement in the environment (sort of states indicating how an agent interprets fields in choosing where to move). For instance, in the corridor example, agents in different states could be attracted by different exits of the corridors and thus could be attracted by the related field, ignoring fields generated by doors leading to internal rooms. This attitude could change according to internal decisions of the agent, or to an external event perceived by it. Of course, different agent types can have different attitudes; summarizing, different agent types can interpret fields in a different way, and agents of the same type, according to their state, can also react in different way to the perception of the same kind of signal.

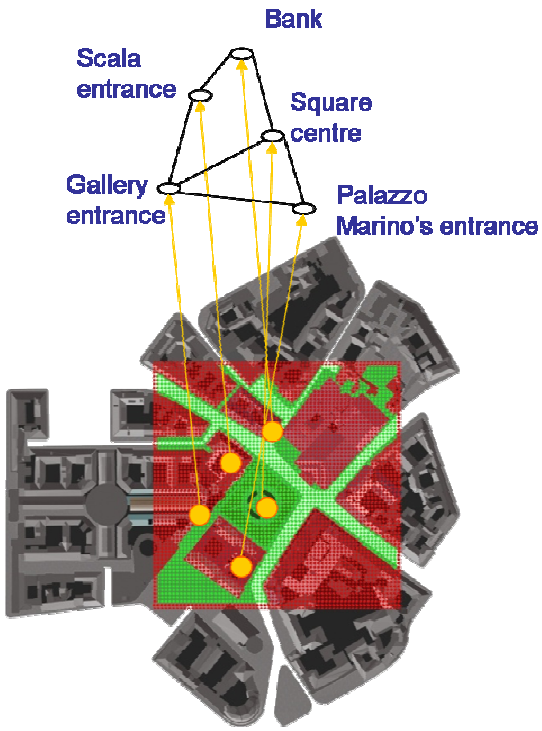

Figure 2 – A diagram illustrating a multilayered environment specification: the bottom layer is a fine grained discretization of Scala Square and the top layer represents its points of interest.

C. *From a Single Layer to Multiple Layers*

The previously introduced representation of the environment can be enhanced by introducing additional representations, for instance representing a different abstraction of the physical space related to the virtual environment. In particular, the different points of interest/reference might be represented on a graph whose links represent proximity or direct reachability relations among the related points, realizing a sort of *abstract map* of the environment. This layer might be interfaced to the previously introduced finer representation of the environment (i.e. the *physical layer*), and it could be the

effective source of fields generated by infrastructural elements, that are diffused to the physical layer by means of interfaces. A sample diagram illustrating this approach to the modeling of a physical environment is shown in Figure 2: the bottom layer is a fine grained discretization of Scala Square and the top layer represents its points of interest, that are associated with agents emitting a proper distinctive presence field.

The abstract map could also be (at least partly) owned by an agent, that could thus make decisions on what attitude towards movement should be selected according to its own goals and according to the current context by reasoning on/about the map, instead of following a predefined script. This kind of considerations do not only emphasize the usefulness of a multiple layered representation of the environment, but they also point out the possibility to enhance the current agents (that are characterized by a reactive architecture) by endowing them with proper forms of *deliberation*, towards a hybrid agent architecture. A complete definition of these deliberative elements of the situated agents is object of current and future works.

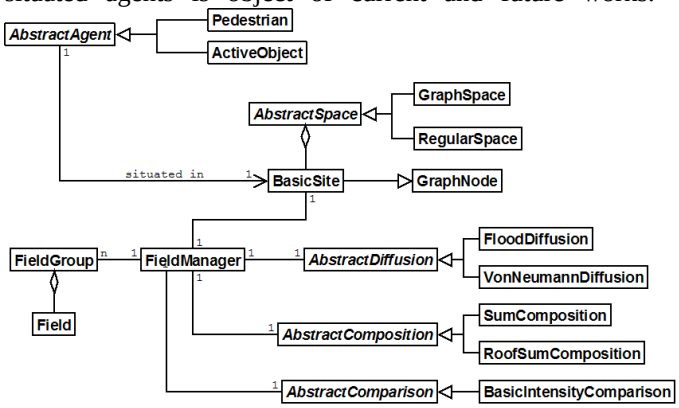

Figure 3 – Simplified class diagram of the part of the framework devoted to the realization of MMASS concepts and mechanisms.

IV. The Execution and Visualization Framework

As discussed in section II, the basic approach that was adopted for this project is to integrate an existing MAS modeling and development framework with an infrastructure supporting an effective form of 3D visualization of the dynamics generated by the model. In particular, to realize the second component we adopted Irrlicht[7], an open-source 3D engine and usable in C++ language. It is cross-platform and it provides a performance level that we considered suitable for our requirements. It provides a high level API that was adopted for several projects related to 3D and 2D applications like games or scientific visualizations. The MAS modeling and development framework we adopted is a C++ porting and relevant refactoring of the original MMASS framework [2], aimed at adapting it to the different programming language and also at optimizing some mechanisms such as commonly adopted field diffusion algorithms. The following subsections will discuss the basic elements of this C++ version of the MMASS framework and the infrastructure interfacing this module with the 3D visualization engine.

[7] http://irrlicht.sourceforge.net/

A. Supporting and Executing MMASS Models

The MMASS framework adopted for this project is essentially a library developed in C++ providing proper classes to realize notions and mechanisms related to the SCA and MMASS models. In particular, a simplified class diagram of the MMASS framework is shown in Figure 3. The lower part of the diagram is devoted to the environment, and it is built around the BasicSite class. The latter is essentially a graph node (i.e. it inherits from the GraphNode class) that is characterized by the association with a FieldManager. The latter provides the services devoted to field management (diffusion, composition and comparison, defined as abstract classes). An abstract space is essentially an aggregation of sites, whose concretizations define proper adjacency geometries (e.g. regular spaces characterized by a Von Neumann adjacency or possibly irregular graphs).

An abstract agent is necessarily situated in exactly one site. Concrete agents defined for this specific framework are active objects (that are used to define concrete points of interest/reference to be adopted in a virtual environment) and pedestrians (that are basic agents capable of moving in the environment). Actual pedestrians and mobile agents that a developer wants to include to the virtual environment must be defined as subclasses of Pedestrian, overriding the basic behavioural methods and specifically the *action* method.

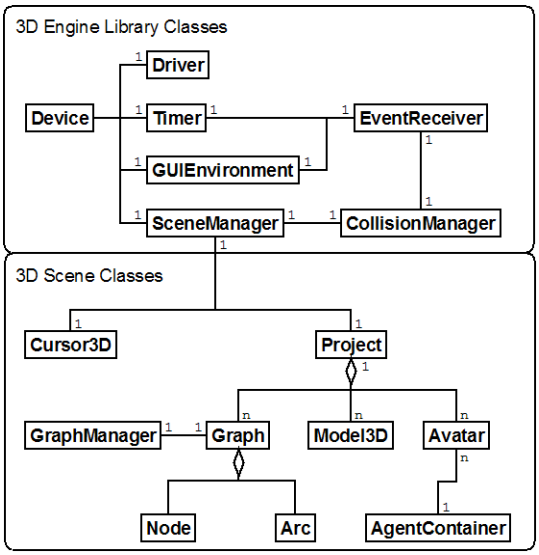

Figure 4 – Simplified class diagram of the part of the framework devoted to the management of the visualization of the dynamics generated by the model.

B. Integrating the Models with a Realtime 3D Engine

While the previous elements of the framework are devoted to the management of the behaviours of autonomous entities and of the environment in which they are situated, another relevant part of the described framework is devoted to the visualization of these dynamics. More than entering in the details of how the visualization library was employed in this specific context, we will now focus on how the visualization modules were integrated with the previously introduced MMASS framework in order to obtain indications on the scene that must be effectively visualized.

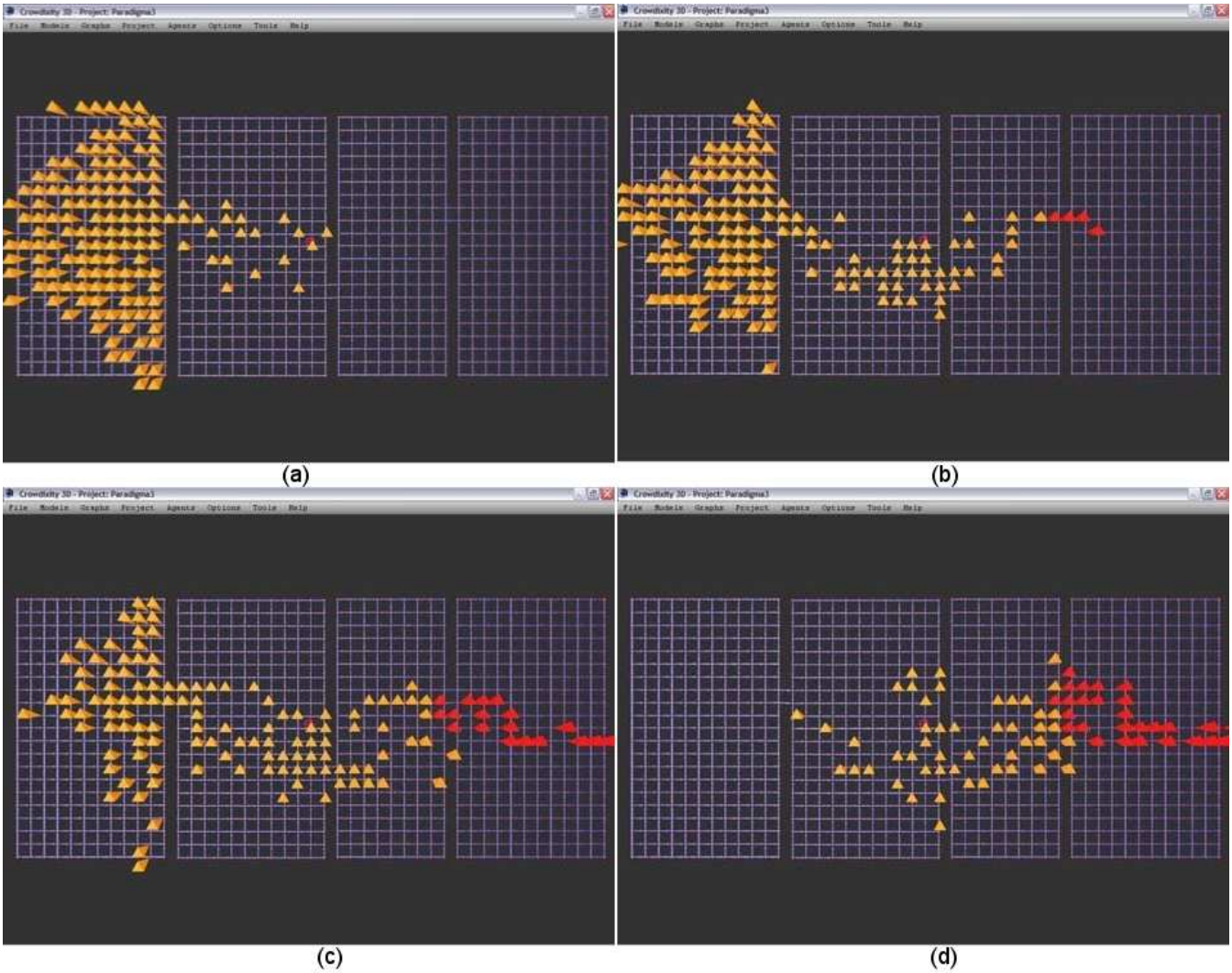

Figure 5 – Four screenshots of the first sample application, showing the movement of very simple agents from a starting room on the left, to an exit in the rightmost room.

Figure 4 shows a simplified class diagram of the main elements of the 3D Engine Library. The diagram also includes the main classes that are effectively in charge of inspecting the state of the MMASS environment and agents, and of providing the relevant information to the SceneManager that will translate it into a scene to be visualized. The *Project* class act as a container of the 3D models providing the graphical representation of the virtual environment (*Model3D* objects), as well as the graph related to the adopted discretization of this physical space (a *Graph* object visually representing the previously discussed *physical layer*). It also includes a set of *Avatar* objects, that are three dimensional representations of *Pedestrian* objects (introduced in the previous subsection).

The framework must be able to manage in a coordinated way the execution of the model defined for the specific virtual environment and the updating of its visualization. To manage this coordinated execution of different modules and procedures three main operative modes have been defined and are supported by the framework. The first two are characterized by the fact that agents are not provided with a thread of control of their own. A notion of *turn* is defined and agents are activated to execute one action per turn, in a *sequential* way or in a conceptually *parallel* way (as for a Cellular Automaton). In this case, respectively after each agent action or after a whole turn the scene manager can update the visualization. On the other hand, agents might be *associated with a thread of control* of their own and no particular fairness policy is enforced. The environment, and more precisely the sites of the MMASS space, is in charge of managing possible conflicts on the shared resource. However, in order to support a fluid visualization of the dynamics generated by the execution of the MAS, the *Pedestrian* object before executing an action must coordinate with the related *Avatar*: if the previous movement was still not visualized, the action is temporarily blocked until the visualization engine has updated the scene. It must be noted that in all the introduced activation modes the environment is in charge of a regulation function [7] limiting agents' autonomy for sake of managing the consistency of the overall model or to manage a proper form of visualization.

V. Sample Applications

The aim of this section is to present some sample applications to show how the framework supports the definition of MMASS models and the realization of an effective three dimensional visualization. The applications were also chosen to show the potential of the framework in terms of execution of a large number of agents. Tests were

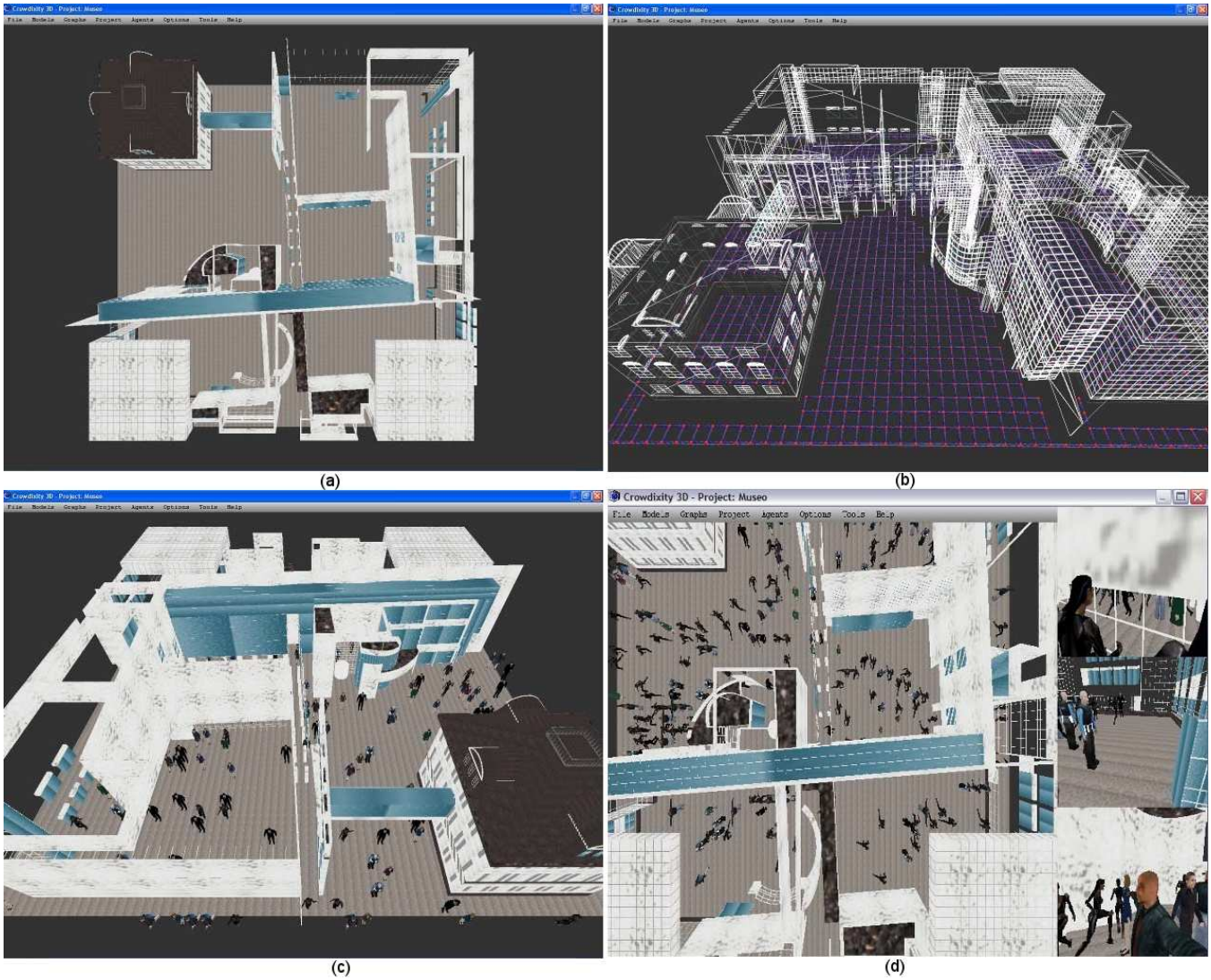

Figure 6 – Four screenshots of the virtual museum application, showing the structure of the environment - (a) and (b) – a perspective view of the evacuation and also a 'bird's eye' view of the environment coupled with three 'first-person' perspectives of agents – (c) and (d).

carried out on a notebook on which the Windows XP Professional operating system was installed; the notebook was provided with an Intel Pentium IV 2.4 GHz processor, with 320 MB RAM and an ATI Raedon IGP graphic card with 128 MB (shared system memory).

The first application is about the simulation of the evacuation of a section of a building, comprising several rooms connected by doors. In this specific scenario agents' behaviours are very simple, and only provide the movement towards specific exits. Agents reaching these exits are simply eliminated from the scenario; some screenshots of this example are shown in Figure 5. In this scenario the environment comprises a graph of around 1000 sites, connected by more than 3500 arcs; 150 agents are situated in the scenario and they are activated according to sequential activation strategy. The analytical results of the simulation are not relevant in this context, also because the agent models were not calibrated against real data; the simulation was executed and visualized with a number of frames per second (FPS) constantly above 60. The speed of the simulation was in fact actually limited to achieve a smooth form of visualization of the system dynamics.

The second example is about the movement of agents inside a virtual museum; the aim of the agents in this scenario is to move outside the buildings to gather in specific areas, as in case evacuation. In this case the environment comprises around 2000 sites (a gross discretization of the represented environment) with around 6000 arcs connecting them; 500 agents were randomly positioned inside buildings, and they were provided with a thread of control of their own. Both the environment and agents were characterized by a 3D visual model, with textures; some relevant screenshots of this sample application are shown in Figure 6. Once again, the analytical results of this simulation are not relevant, since the agent models were extremely simple and they were not calibrated against real data. The simulation was executed and visualized with a number of FPS constantly above 30.

We also executed a stress test on a different hardware configuration, to verify the scalability of the framework; the workstation was based on Windows XP Professional operating system, with an Intel Pentium Core 2 Duo 2.4 GHz, 2 GB RAM and a NVIDIA Quadro FX 3450 graphic card with 256 MB. The test environment was constituted by 11000 sites, connected by around 44000 arcs; 10000 agents, sequentially activated, were positioned in this environment. Their behaviour was simply to move towards the closest source of an 'exit' field; agents reaching the source were

removed from the environment. The system was able to execute and visualize the simulation with 22 FPS, when the structure of the environment was hidden (reducing the number of displayed triangles), and with 3 FPS when it was visualized.

VI. Conclusions and Future Developments

The paper has presented a framework supporting the definition and realization of virtual environment inhabited by interacting situated agents modeled according to the Multilayered Multi-Agent Situated System. The framework supports the specification and execution of visually rich 3D virtual environment characterized by the presence of situated agents acting and interacting inside it. The paper briefly introduced some relevant related works, then it presented the multi-agent model underlying the framework and its basic architecture (with specific reference to the integration of computational support to the formal model and the visualization components). Sample applications were also described in order to show the potential of the framework in executing models comprising several hundreds of agents producing an effective visualization of the generated dynamics.

Future works are aimed, on the one hand, at improving the set of support instruments, both methodological and computational, supporting for instance the definition of the spatial structure of the virtual environment. Some support instruments, such as a tools for a semi-automatic realization of discrete abstractions of an existing 3D model (e.g. a 3D Studio design of an architectural space) was already realized, but it must still undergo a thorough testing phase. Additional relevant future works are instead aimed at providing a more expressive modeling framework, as briefly discussed in Section III-C.

References

[1] S. Bandini, S. Manzoni, C. Simone. Heterogeneous Agents Situated in Heterogeneous Spaces. Applied Artificial Intelligence, 16(9-10):831–852, 2002.

[2] S. Bandini, M. L. Federici, S. Manzoni, G. Vizzari. Towards a methodology for SCA based crowd simulations. In: VI International Workshop Engineering Societies in the Agents' World, vol. 3963 of Lecture Notes in Artificial Intelligence, Springer-Verlag, pp. 203–220, 2006.

[3] S. Bandini, S. Manzoni, G. Vizzari. Situated Cellular Agents: a Model to Simulate Crowding Dynamics. IEICE - Transactions on Information and Systems: Special Section on Cellular Automata, Vol.E87-D(3):669-676, 2004.

[4] S. Bandini, S. Manzoni, G. Vizzari. Multi Agent Approach to Localization Problems: the Case of Multilayered Multi Agent Situated System. Web Intelligence and Agent Systems, IOS Press, 2(3):155-166, 2004.

[5] S. Bandini, S. Manzoni, G. Vizzari. Towards a platform for Multilayered Multi Agent Situated System based simulations: focusing on field diffusion. Applied Artificial Intelligence, Taylor & Francis, 20(4-5):327-351, 2006.

[6] S. Bandini, G. Mauri, G. Vizzari. Supporting Action-At-A-Distance in Situated Cellular Agents. Fundamenta Informaticae, 69(3):251-271, 2006.

[7] S. Bandini, G. Vizzari. Regulation Function of the Environment in Agent-Based Simulation. Environments for Multi-Agent Systems III, Third International Workshop, E4MAS 2006, vol. 4389 of Lecture Notes in Computer Science, Springer-Verlag, pp. 157-169, 2007.

[8] M. Batty, A. Hudson-Smith. Urban Simulacra: From Real to Virtual Cities, Back and Beyond, *Architectural Design*, 75 (6):42-47, 2005.

[9] M. Batty. Agent-based pedestrian modeling. In *Advanced Spatial Analysis: The CASA Book of GIS*, pp. 81-106, 2003.

[10] J. Dijkstra, H. P. J. Timmermans. Towards a multi-agent model for visualizing simulated user behavior to support the assessment of design performance. Automation in Construction 11:135-145, Elsevier, 2002.

[11] J. Dijkstra, J. Van Leeuwen, H. J. P. Timmermans. Evaluating Design Alternatives Using Conjoint Experiments in Virtual Reality. Environment and Planning B 30(3):357–367, 2003.

[12] J. Ferber. Multi-Agent Systems. Addison-Wesley, 1999.

[13] T. Ishida, Y. Nakajima, Y. Murakami, H. Nakanishi. Augmented Experiment: Participatory Design with Multiagent Simulation. IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1341-1346, 2007.

[14] J. Klein. Breve: a 3D simulation environment for the simulation of decentralized systems and artificial life. In Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems. The MIT Press, pp. 329–334, 2002. http://www.spiderland.org/breve/breve.pdf.

[15] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan. MASON: A Multi-Agent Simulation Environment. In Simulation 81(7) :517-527, 2005.

[16] M. Mamei, F. Zambonelli. Motion Coordination in the Quake 3 Arena Environment: A Field-Based Approach. Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004, vol. 3374 of Lecture Notes in Computer Science, Springer-Verlag, pp. 264-278, 2005.

[17] M. Mamei, F. Zambonelli. Field-Based Coordination for Pervasive Multiagent Systems, Springer-Verlag, 2006.

[18] H. Nakanishi, S. Nakazawa, T. Ishida, K. Takanashi, K. Isbister. Can Software Agents Influence Human Relations? - Balance Theory in Agent-mediated Communities. *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, ACM press, pp. 717-724, 2003.

[19] P. Nugues, S. Dupuy, A. Egges: Information Extraction to Generate Visual Simulations of Car Accidents from Written Descriptions. In: Computational Science and Its Applications - ICCSA 2003, vol. 2667 of Lecture Notes in Computer Science, Springer-Verlag, pp. 31-40, 2003.

[20] F. Nunnari, C. Simone. Perceiving awareness information through 3D representations. Proceedings of the working conference on Advanced Visual Interfaces, AVI 2004, ACM Press, pp. 443-446, 2004.

[21] G. Papagiannakis, S. Schertenleib, B. O'Kennedy, M. Arevalo-Poizat, N. Magnenat-Thalmann, A. J. Stoddart, D. Thalmann: Mixing virtual and real scenes in the site of ancient Pompeii. Journal of Visualization and Computer Animation 16(1):11-24, 2005.