In Applied Artificial Intelligence Journal 14(1):69-91, January 2000

A Paradigm for Controlling Virtual Humans in Urban Environment Simulations

Nathalie Farenc, Soraia Raupp Musse, Elsa Schweiss,
Marcelo Kallmann, Olivier Aune, Ronan Boulic, Daniel Thalmann
Computer Graphics Lab. Swiss Federal Institute of Technology
EPFL, DI-LIG, CH 1015 Lausanne, Switzerland
{nathalie,soraia,schweiss,kallmann,aune,boulic,thalmann}@lig.di.epfl.ch

This paper presents a new architecture for simulating virtual humans in complex urban environments. Our approach is based on the integration of six modules. Four key modules are used in order to: manage environmental data, simulate human crowds, control interactions between virtual humans and objects, and generate tasks based on a rule-based behavioural model. The communication between these modules are made through a Client/Server system. Finally, all low-level virtual human actions are delegated to a single motion and behavioural control module. Our architecture combines various human and object simulation aspects, based on the coherent extraction and classification of information from a virtual city database. This architecture is discussed in this paper, together with a detailled case-study example.

***Keywords**: virtual urban simulation, rule-based behaviours, object interaction, human crowd control.*

INTRODUCTION

For several years, we have been working on modelling and simulations of realistic virtual humans. A most important aspect of our work was to investigate virtual human behaviours (for instance: Becheiraz, 1996; Noser, 1996) and body motion control (Boulic et al 1997). However, less attention has been given towards the integration of these techniques in larger applications, where interaction with the virtual environment is needed. This paper presents an architecture integrating different aspects of human and object simulations in an urban environment using information extracted from a virtual city database. All such issues are addressed within the scope of our current application: populating virtual environments with virtual humans that can interact with objects and can walk within specified environmental constraints. One key aspect is the management of numerous virtual humans with the crowd module.

Throughout the text, the term *agent* refers to a virtual human agent and the term *crowd* refers to a group of agents.

Related Work

We present here some works related to the following topics: construction of a virtual city, human crowd management, agent-object interaction, rule-based behaviour, and client-server message control.

Several works have been done on virtual cities: reconstructing an existing city with image processing (Donikian, 1997; Russ et al, 1996; Jepson and Friedman, 1996), using urban knowledge to deal with traffic problems (Jepson and Friedman, 1996; Howard, 1997; Mokhtarian, 1997), considering the rules of urban displacement (Ingram et al, 1996; Jepson and Friedman, 1996; Aufaure et al, 1994) or designing a virtual city (Fuji et al, 1995; Jepson and Friedman, 1996). To produce realistic simulations the environment has to integrate semantic notions about specific areas algorithmically defined, supermarkets or town squares or other real places which can be modelled. Such notions correspond to natural human analysis using basic urban knowledge, e.g. “a sidewalk is for pedestrians”. Our environment is decomposed into entities corresponding to semantic information, thus leading to the concept of an Intelligent Virtual Environment (IVE), where appropriate information is given to virtual creatures.

Considering crowd simulations, there are several approaches which might be used such as particle systems and behavioural systems. These techniques are characterised by the possible number of individuals to be simulated, their autonomy level and decision-making ability, the associated collision avoidance method etc. Several authors have worked in order to provide techniques to control with physical rules many autonomous agents using particle systems (Bouvier et al, 1997; Brogan and Hodgins, 1997). Behavioural systems consider the autonomous agent as an intelligent agent who can make decisions using specific rules (Noser and Thalmann, 1996; Bécheiraz 1998, Reynolds, 1999; Tu and Terzopoulos, 1994). Our typical application uses a crowd management method (Musse and Thalmann, 1997; Musse et al, 1998) for which we need a virtual city database in order to establish the connection with the virtual environment and to synchronise various actions made by agents.

Concerning agent-object interaction tasks, we include semantic and behavioural information within each object description. Some proposed systems already use this kind of approach. In particular, *object specific reasoning* (Levison, 1996) creates a relational table to inform an agent of an objects' purpose, hand shapes and grasp approach direction. This set of information is intended to be sufficient to perform a grasping task. Kallmann and Thalmann (1998) introduced the concept of *Smart Objects,* integrating object specific interaction features such as intrinsic object properties, information on how-to-interact with it, objects functionality, and as well as expected agent behaviours.

Rule-based behavioural systems are well known and often exploited to simulate behaviours (Reynolds, 1999 ,Chaib-draa, 1997, Neves and Oliveira, 1997, Zeltzer, 1997). Several authors have used it in order to describe human and animal behaviours. We present an RBBS (Rule Based Behaviour System) which describes a syntactic analyser of natural language in order to specify behavioural rules to guide crowds. We base our rule-based behavioural system module on the hierarchical structure defined by the *Skill-Rule-Knowledge* levels of Rasmussen (1996).

We adopt a synchronisation method based on a Client/Server system (Guzzoni et al, 96) for the communication between the different clients. The central idea here is to have an open architecture where several clients can be integrated, resulting in a general and extendable framework to simulate a virtual city.

Overview of the Architecture

Our architecture is based on the integration of six distinct modules, each one having its own specific purpose, as described by the following:

- RBBS *(Rule-Based Behavioural System*), generates intentions and tasks that can be applied to agents and crowds.
- CITY *(Virtual City Database),* maintains the database of the virtual environment. This module is able to provide information from the database during the simulation.
- CROWD *(Human Crowd Management),* controls the motions and behaviours of groups of agents. Crowds can autonomously perform pre-programmed behaviours, or be *guided* by a controlling agent or group of the crowd.
- SMART OBJECT *(Smart Object Management),* is the module responsible for performing agent-objects interactions. Smart Objects store, for each possible interaction, pre-defined plans that can be are used by the virtual humans.
- AGENTlib *(Agent Management),* is the library responsible for performing individual agent actions, such as behaviours and motions. This library is directly called by the CROWD and SMART OBJECT module and guarantees motion coherence.
- Controller *(Communication System Server),* synchronises the communication system between the first four modules (its clients), by knowing which messages each client can understand. In this way a client can make any request to the Controller that will redirect the request to the correct module.

Modules need to communicate with each other to achieve a given task. This implies that there must be a well-defined message exchange system. An overview of the dependencies between these modules is schematically described in Fig. 1. More details about this are presented in Section COMMUNICATION BETWEEN MODULES.

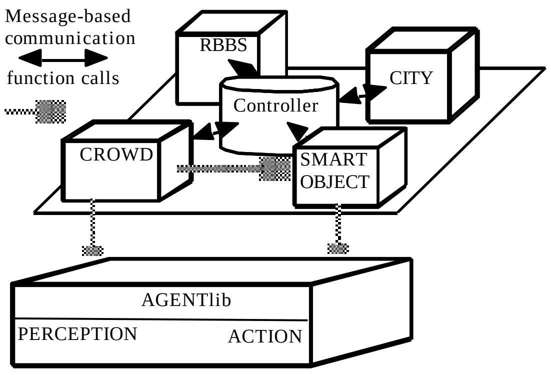

Figure 1 - ***Representation of the system architecture***

Organisation of This Paper

The paper is structured as follows: in the first section we present how our virtual city is structured. In the next section, the designer's viewpoint discusses existing modelling constraints. The subsequent three sections present, respectively, the modelling of Smart Objects, major concepts of human crowd management, and the rule-based control module. The last two sections detail how the message exchange protocol works, and presents the entire system functionality with a case study application.

VIRTUAL CITY DATA BASE

What is required for virtual human behavioural simulation in a complex environment like a city? Bearing this question in mind, we can assume that urban displacements depend to a large extent on geometrical data and urban features knowledge. For us an urban environment is a place where information (semantic and geometric) is dense and can be structured using rules. We define urban knowledge for this work as a set of urban structural information, objects usable according to some conventions, and an association between places (geometric areas) and some semantic information.

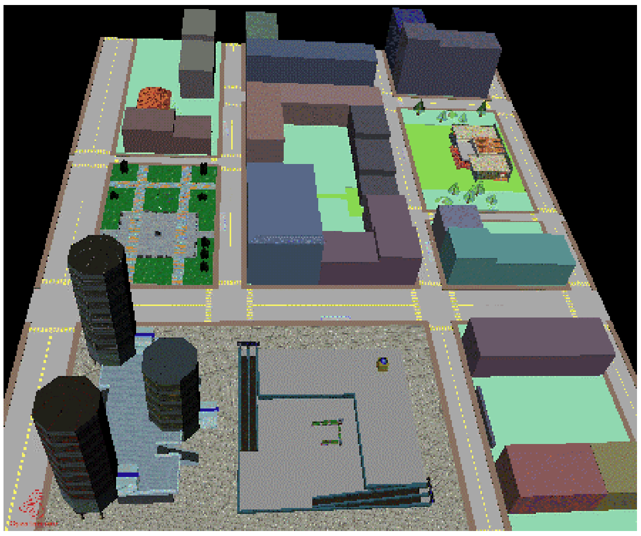

Figure 2 - ***Overhead of the virtual city***

Our aim is to be able to deduce which actions are possible according to this information. For example, in a city, we can denote sidewalks as areas for pedestrian entities, so that in such places they may walk or have social encounters (Doyle and Hayes-Roth, 1997). The city database has to inform all "mobiles" (objects with mobility such as pedestrians, cars, buses, and bicycles) as to where they may circulate. In some cases, even if pedestrians may walk across an area, their motion has to be synchronised with other mobiles depending on the traffic signal's status. Then the environment has to inform mobiles that they are in a synchronisation area and have to deal with other mobiles. How might one organise them to access information efficiently? Our approach is to define areas. The areas can be subdivided into sub-areas, or grouped, depending on the 'level of information'. This decomposition tidies up and links the semantic data. The city is decomposed into several areas, depending on their geographical and functional properties. At each level in the environmental representation, the accessible information corresponds to a level of abstraction. At a town level we are interested in knowing the name of different quarters and not the position of the next pedestrian crossing. Moreover, information for crossing an area can be "how to cross a city?" (reply concerning motor ways) or "how to cross a street?" with information for pedestrians informing them about sidewalks or pedestrian crosswalks or other objects in order that they will avoid collisions.

Specific areas of the scene are defined as "Environment Entities" (*ENV*) because they possess geometrical and semantic information. According to figure 3, the city is decomposed into "quarters", and the quarters are then decomposed into block levels. We define a block as a part of a quarter, composed of streets, junctions and "parcels". Junctions are crossroads that connect streets. We consider as a parcel a portion of land that has no streets or junctions inside, like a park or a piece of land with buildings. In this way, all the space covering a block can be marked out and all points on the surface of a block can be associated with an ENV.

All these entities are themselves composed of sidewalks, pedestrian crossings, rolling stock (traffic) for the street, travelling or non-travelling areas for parcels. Figure 2 represents a view of our modelled block and Figure 3 shows a representation of the ENV hierarchy.

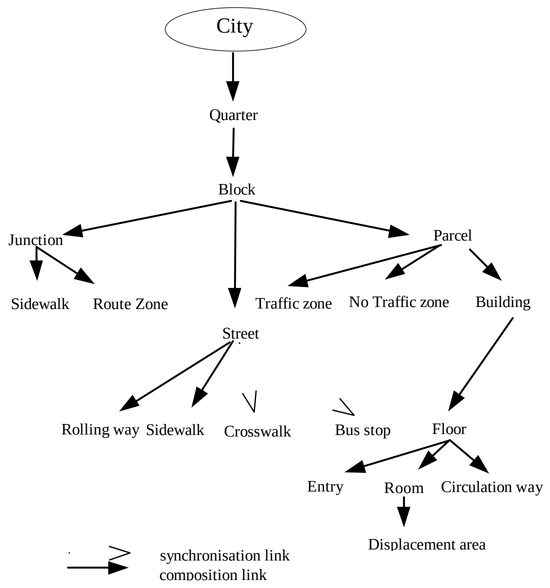

Figure 3 - ***City hierarchical decomposition in ENV***

To create a database based on *ENVs* we use a scene graph. In the scene some objects are named so that they can be detected as *ENVs* and analysed to extract their geometrical data. From a set of *ENVs* we calculate the hierarchical dependency between all the *ENVs*(according to the hierarchical decomposition model shown in Figure 3), all the geometrical information (walking area is defined in a parcel entity) and the connectivity information between *ENVs*. The connectivity data is used to derive the authorised paths depending on the considered mobile type. Some simple semantic information is associated with each ENV (area for car, pedestrian or bus) identifying who can use which *ENV*. More information can be added concerning more specific action such as "playing at the playground". Figure 4 shows the connectivity graph for the *ENV* used by pedestrian mobiles. This figure highlights the objects usable only by pedestrian mobiles (crosswalk, sidewalk and pedestrian traffic zone). The cylinders represent the links between the ENV nodes in the connectivity graph. Their location is used for the computation of travelled distances through the city (the full description of this topic is beyond the scope of this paper, one can refer to (Farenc et al, 1999) for details). The naming convention of objects is based on a precise definition. Objects are parsed using their names, and put in the ENV hierarchy according to the information provided by the name analysis. The next section presents a city designer's viewpoint of this kind of constraint.

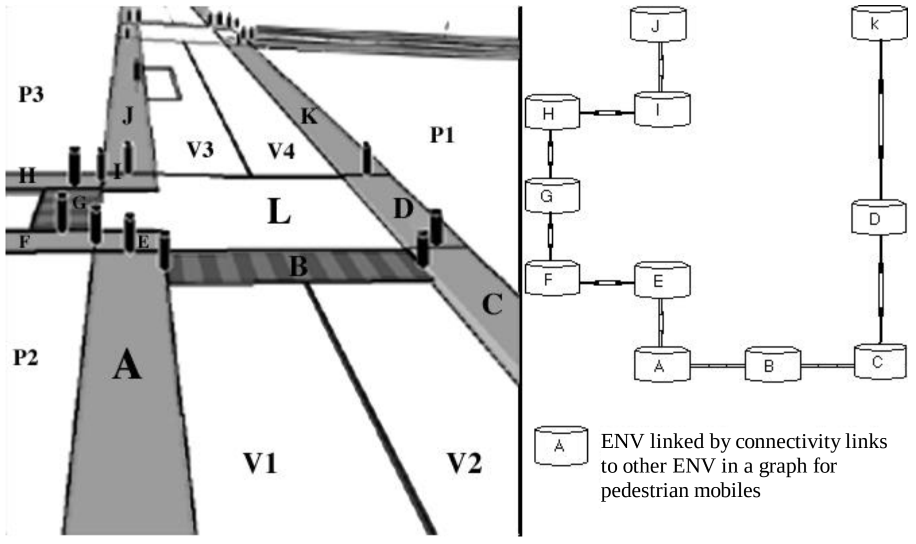
P3
J
K
P1
V3
V4
H
I
G
L
D
F
E
B
C
P2
A
V1
V2
J
k
H
I
G
D
F
E
A
B
C
A
ENV linked by connectivity links to other ENV in a graph for pedestrian mobiles

Figure 4 - ***Connectivity graph for pedestrian ENV represented by real visual objects in the block and by the graph associated***

THE DESIGN ASPECTS

In an interactive walkthrough, only the geometry is relevant while the user interprets the semantic and spatial aspects of the graphical database in an intuitive way. To provide an autonomous behaviour to the agents, they have to be informed about the significance of the geometry they must deal with. In this context our goal is to build some semantic tools, which allow one to encode in and extract from the different components of the virtual world the own functionalities dedicated to urban living. An IVE (intelligent virtual environment) needs rules to conceptualise in a precise way how the designer team will produce a graphical and semantic environment usable by developers. Which syntax should one choose? The concerns of the developers and those of the designers are quite different. The designers have to manage different definitions of the objects: geometric, visual and spatial. They have to ensure the compatibility of the graphical database with a range of formats involved in its constitution. Responsible for the major parts of the data entry, designers are prone to acts of omission or creating wrong semantic entries, hence the rules have to be as intuitive as possible. In fact an automated encoding based on feature recognition is desirable wherever possible.

Different Categories of Autonomous Entities

It is important to make a distinction between the behaviour of a virtual creature and the functionality of an object composing the virtual world. The functionality of an object is defined at the time of the virtual world design so that its whole range of state is made explicit or clearly bounded. On the other hand, the behaviour of a virtual creature results from the exploitation of object features by applying an additional perception-action model. By construction it is more difficult to predict its whole range of potential states. It can only be reasonably bounded at an individual level for a given application. Finally, the populated virtual city itself can be viewed as an entity of higher complexity that can be the source of unforeseen behavioural patterns. Identifying such emergent situations is extremely important for characterising and validating our model. This is one of our long-term goals.

In an IVE, the concept of an object has to be precisely defined with respect to the type of information it possesses: visual, spatial, and/or behavioural.

Displayed Objects : They have only a visualisation purpose.
Objects Box : They are associated with displacement surfaces for the calculation of local collisions. These objects can be associated to Displayed Objects for visualisation.
Smart Objects : Some objects of the 3D scene integrate some functionalities, which ranges from the control of some internal degree of motion (a door movement) to the diffusion of information (a signpost). This is discussed further in the next section.

MODELLING SMART OBJECTS

In order to easily control agent-object interactions, we use the Smart Object framework (Kallmann and Thalmann, 1998). This framework permits the designer of an object to model not only the object's geometry but also extra information like specific parts to interact with and its functionalities. Having done this, the simulator program can use this information to guide an interaction with the virtual actor. Then, the designer can have the feedback of the simulator program to adjust the object design, taking control of the entire cycle of object design and testing.

This approach of including information inside the object description is known in the literature as “Feature Modelling” and is most often discussed within the CAD/CAM domain. As commonly stated, a feature is a region of interest on the surface of a part (Pratt and Wilson, 1985; Parry-Barwick and Bowyer, 1993). In our case, we identify “interaction features” which are parts of the object that may offer some kind of interaction. Consider, for example, a button to be pressed, the knob of a door, or a drawer in a table. These features are identified and associated with the object’s functionality, which is defined with a dedicated script language. Some examples of such interaction-features are: hand shapes, special locations where the agent should be during an interaction, and object movements. In this way, the application-specific object reasoning will have sufficient information to perform its tasks with the object.

Modelling Interaction Information

A typical Smart Object description contains, among other operators, a collection of hierarchical *parts*, *actions*, *commands*, *gestures* and *functionality rules*. Each action represents a movement that can be applied to any part of the object. A command is an entity that links a part to an action, and may be triggered after an agent’s gesture. For example, the motion of the arm and hand of an agent pressing a button is considered to be a gesture.The object's functionality rules are described by a list of *behaviours* and *state variables*. Each behaviour is described by a list of low level instructions that can be directly interpreted. Agent-related instructions are performed by requesting the corresponding motion motor from the AGENTlib module.

All these interaction features are identified and defined during the modelling phase by using a graphical interface modeller. Figure 5 shows an example of an automatic door, which opens after a button is pressed, being modelled by many different kinds of information. For example, we can note a positional vector that is used by the agent to get closer to the door, and a hand shape used to press the button.

A simple scripted behaviour is saved within the object forming a pre-defined plan that can be directly interpreted during run time. For example, when the agent “wants” to enter a room by opening the door, the related interaction script is interpreted and will generate a sequence of AGENTlib actions: the agent’s hand will move towards the button using an inverse kinematics motor (Baerlocher and Boulic, 1998), the hand will be attached to the button, the button will move, and finally the door will move and open.

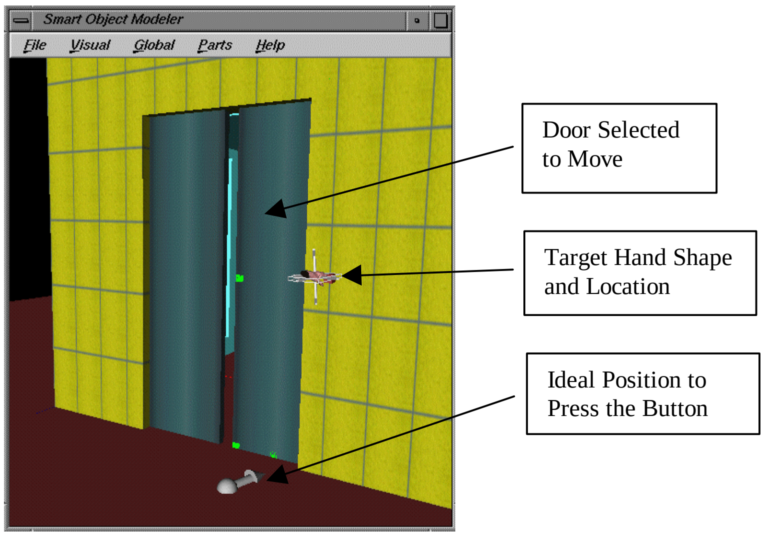

Figure 5 - ***Modelling some Interaction features of a lift door***

Smart Objects and the Virtual Environment

In our virtual city simulation, the RBBS will not take care of some low-level decisions. Considering the example of an actor passing through a door, it is not worthwhile to do a complex reasoning algorithm to decide things like which hand shape best fits the doorknob. Moreover, this is not the role of the RBBS module. In such a case, it's simpler to store the best pre-defined hand shape to be used within the definition of the object door itself.

There is a compromise when deciding what kind of information should be included within the object, and what should be left to the object's reasoning algorithm. The idea is to be able to model, in the design phase of the object, as much generic information as possible. And then, for each situation, the application can decide whether the object information can be used or needs to be calculated. For example, the door can have a pre-defined agent position to be opened with the right hand, but the reasoning module may decide for itself which hand to use for any given position.

The Smart Object reasoning in a virtual city environment might have perceptions and communicate with other entities. For example, if the automatic door does not have a button, it might have a perception to test if there is an actor nearby, so that the door automatically opens. On the other hand, the actor perceives that there is a door, through which it can go and enter without any interaction, as it is an automatic door. Once the plus important needs of the application are well defined, the Smart Object framework provides a versatile way to model many possible interactions.

The following characteristics can be identified in a simulation system based on the Smart Object framework:

• Decentralisation of the animation control. By interpreting plans stored in a Smart Object, many object-specific computations are released from other modules.

• Reusability of designed Smart Objects. A Smart Object can first be modelled for a specific situation, and updated if needed for other situations or interactions without changing the usability of the original design.

• A simulation-based design is naturally achieved. The designer can take control of the loop: design, test and re-design. A designed Smart Object can be easily inserted into a simulation program, which gives feedback for design improvements.

Figure 6 shows an example of two agents before entering into a two-stage lift. All interaction information (gestures to press the button, movements of doors and cabin, etc) were modelled together with the lift functionality. The default behaviour of the lift describes a sequence of instructions to move an agent from its current floor to the next one. Many details are handled, e.g. only the first agent arriving will press the call button. More details about how instructions are handled are described by Kallmann and Thalmann (1999).

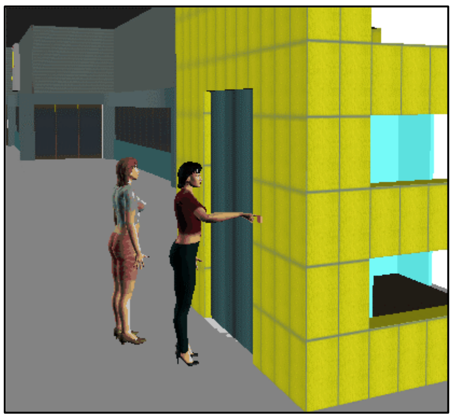

Figure 6 - ***Agents interacting with a Smart Object Lift***

HUMAN AND CROWD CONTROL

Simulation of human crowds for populating virtual worlds provides a more realistic sense of virtual group presence (Benford et al, 1997). There are several approaches for modelling autonomous crowds (Bouvier et al, 1997; Brogan and Hodgins, 1997; Reynolds , 1999; Tu and Terzopoulos, 1994). In a virtual urban environment, it is useful to simulate autonomous populations, i.e. agents that can have a kind of environmental knowledge (CITY), and are able to move and interact with other agents and their environment. In addition, the user can define crowd intentions which will be achieved by the groups of agents. We have defined a crowd model where autonomy and interactivity control can be mixed in the same simulation (Musse and Thalmann, 1998). We have defined an ***autonomous crowd*** to be the type of crowd which acts according to the pre-defined intentions and the environmental specifications, which will be achieved by the groups of agents without user intervention during the simulation.

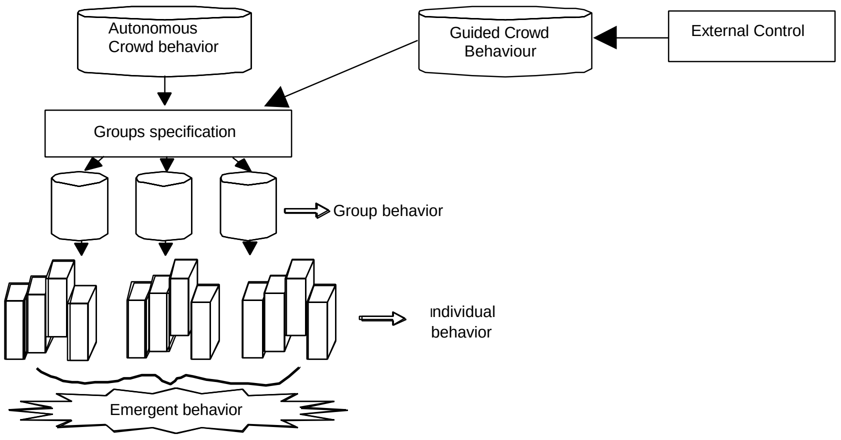

Figure 7 - ***The architecture of the CROWD system***

The other kind of crowd control present in our work is the ***guided crowd*** that can be guided by an external process. This could be done through an interface with an avatar (virtual human representation controlled by an end user) (Musse et al, 1998) or an intelligent autonomous agent (Schweiss et al, 1999). The guided crowd responds to dynamic behaviours which can change as a function of time whereas the autonomous crowd applies pre-defined behaviours without user intervention during the simulation. Examples of behaviours are: interaction with objects (Kallmann and Thalmann, 1998), the displacement to reach a specific point of interest, a keyframe sequence to be played (Boulic et al, 1994), etc. The crowd behaviour is distributed among the groups and individuals, according to group specifications (Musse and Thalmann, 1997). The data structure of our crowd model is presented in Figure 7. At the highest information level, a crowd is treated as a single entity, which is formed by agent groups that can have the following specific behaviours:

- seek goal (the groups have one or more goals to follow);
- flocking motion (group ability to walk together);
- collision avoidance;
- formation to reach a goal or an interest point (when a goal or an interest point is shared by many individuals);
- action to be performed when the agent arrives at a specific goal (Boulic et al, 1997);
- following motion (group's ability to follow another group);
- group control (which can be autonomous or guided).

The connection with the virtual environment provides information to model crowd motion, for instance, the physical locations of the goals or interest points, positions and dimensions of each obstacle. We can see in Figure 8 some interest points (represented by white cylinders) in the virtual city and which can serve as goals for the individual groups. These goals are distributed for the different groups to construct the crowd motion.

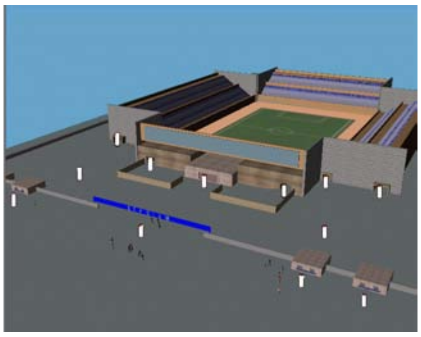
Figure 8 - ***Interest points and goals to drive the crowd***

Next, the group behaviours are generated for the autonomous crowd or directly provided to the guided crowd, during the simulation. In Figures 9a and 9b, we can see different groups going to different locations enacting different goals and behaviours.

Figures 9a and 9b - ***Different groups entering in the stadium***

RULE-BASED BEHAVIOUR CONTROL

The rule-based behaviour control was designed to guide crowds in the virtual city (Schweiss, 1999). It sends high level orders to the Controller, like "Go to supermarket", chosen according to behavioural rules and crowd states. The rules are pre-defined by the user using a pseudo natural language interface. Then, these rules are interpreted during the simulation to guide agents and crowds.

The rule-based behaviour control analyses the context considering three main criteria chosen for the simulation: type of behaviours, agent relations and behaviour treatments. Types of behaviours are used to input and treat daily life behaviours such as "Need for buying the bread", "Wish to visit a museum", or "Replying to a phone call". The agent relations module permits one to specify humanoid interactions and relationships. Finally, according to the system state, the behaviour treatment module selects rules that produce state changes (Norvig, 1992).

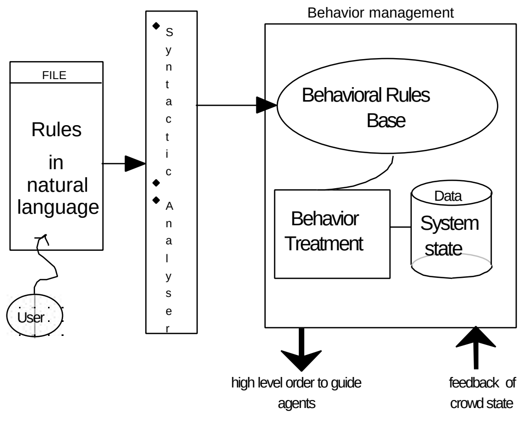

Figure 10 - ***General organisation***

After rules are entered via natural language, they are interpreted by a syntactic analyser and added to the behavioural rules base (Figure 10), allowing non-programmers to design basic rules for daily life simulations. This analyser reads and translates the user's behavioural rules for the system. The system then works with a rule base composed of predefined basic rules and the user's rules.

User's rules have a fixed syntax based on the following semantic elements (Wilks, 1986) :

1 - ENTITIES : crowds, agents, objects
2 - ACTIONS : to be, to pick, to have to, to cause, to feel
3 - TYPE INDICATORS : a kind of, how
4 - SORTS : man, woman, any character
5 - CASES : source, location, direction, goal, in, possessed

Rules are composed of two parts (Norvig, 1992) : a premise corresponding to the condition or test which determines when the rule can be selected, and a conclusion or actions implied when the premise test is validated. *Rule structure* **:** IF (condition) THEN (action).

A condition can be classified into three categories:

1 - Is an agent a kind of a given sort?
2 - For an agent, are actions active?
3 - How is an action performed?

The rule conclusion represents the state when the premise is validated by the current state of the virtual world.

For a daily life behaviour treatment, the rules are structured into different sets, and organised in trees. Root rules - called 'start rules' – are used to determine which the sequential events of daily life occurred. The system, then, deduces a sequence of responses, by exploring the rules of the tree whose root was selected. When a rule condition is validated, the rule produces some activation and deactivation of actions on the virtual agents. Consequently, the state of the system changes, and only then, children rules in the tree are taken into consideration. A daily life behaviour treatment ends with the validation of a leaf rule.

COMMUNICATION BETWEEN MODULES

As previously mentioned in the introduction, our architecture is based on the integration of six distinct modules, each one having its own specific purpose (see figure 1). These modules are named as follows: RBBS (rule-based behavioural system), CITY (virtual city database), CROWD (human crowd management), SMART OBJECT (agent-object interaction control), AGENTlib (controls agents actions), and the Controller (that synchronises message exchange).

The Controller knows which messages the modules CITY, CROWD, SMART OBJECT and RBBS can respond to. In this way, any module can perform a query to the Controller that will correctly redirect the query to the module which will understand it (the request). This kind of architecture is open, and allows us to connect also external modules to monitor the simulation state during run time

Module can ask for ->	CITY	CROWD	SMART OBJECT	AGENTlib
CITY		Status of agents and groups	Status of smart objects	
CROWD	Paths to go to specific location		Status and configuration of smart objects	Perception of objects and agents
RBBS	Paths to go to a specific location	Status of agents and groups	Status of smart objects	
SMART OBJECT		Status of agents and groups		Perception of objects and agents

Table 1: The dependencies between the modules.

Table 1 shows the inter-dependence between the different modules. Note that the module AGENTlib is not a client directly connected to the Controller. Instead, it is accessed through clients CROWD and SMART OBJECT. Moreover, AGENTlib does not ask information of other clients whereas RBBS asks information of other clients but does not give any information back.

In our client-server communication system, all messages are sent to the central Controller, which analyses the label messages and redirects it to the proper recipient. The main drawbacks of this method are the volume of messages sent to the Controller, the complexity of the Controller for message treatment and the low-level control for the rule-based behaviour control. The Controller can be saturated, and we may run into some problems of synchronisation between messages and the applications treatment.

As an alternative, we minimise the transmitted data by delegating low-level decision to other modules directly connected to AGENTlib. Crowd humanoids and smart objects are defined as agents in the common AGENTlib software architecture (Boulic, 1997). AGENTlib maintains a database concerning all the agents created during one simulation. We can also define some perceptions, which allows agents to extract information about neighbours . A perception can be specialised by adding a selection criterion for some types of agents in a limited space (Bordeux et al., 1999).

Using such features, we can refine our model for an integration as follows: if we analyse the modules as linked or not with AGENTlib (Fig. 1), we can see that CITY and RBBS are independent and can run only when connected with the Controller. However, CROWD and SMART OBJECT modules are defined on top of the AGENTlib layer.

For example, for synchronisation with traffic lights we use some smart objects dedicated to synchronous area management. This specialised action is a predefined behaviour activated for each agent arriving in a synchronous area. In this case, all messages for traffic lights management are internal and do not pass through the Controller. In such a configuration, the CROWD module gives access to the perception for the leaders, and if needed delegates the control to the SMART OBJECT module. Figure 1 depicts our communication approach.

The next section concretely exemplifies how messages are exchanged in order to perform a case study simulation.

A CASE STUDY

Now that all of the different modules have been presented, let us examine a concrete example to see how data is exchanged to perform realistic simulation of humanoids in an urban environment. The test case is: “an autonomous group named G, composed of agents h3 and h4, wants to go from its home to the supermarket”. We have three groups ((h1, h2), (h3, h4), (h5, h6)), three smart objects that are traffic lights L1, L2 and door D, and three possible paths. The group’s intention is decided by the rule-based behaviour control. The city database computes a path and the crowd module performs the distribution of crowd information among the individuals as well as the displacement in the urban environment. During the displacement, the agent of the crowd can meet a smart object. In this case, depending on the smart object type, the interaction can influence the smart object’s state or the autonomous agent or both. In the case where the agent goes to a door, we state that the door automatically opens itself when an agent is near. Figure 11 shows the representation of the context. In the path we can see that the agent has to go through a synchronisation area with the traffic lights, and an automatic door D has to open when the agent arrives at the supermarket.

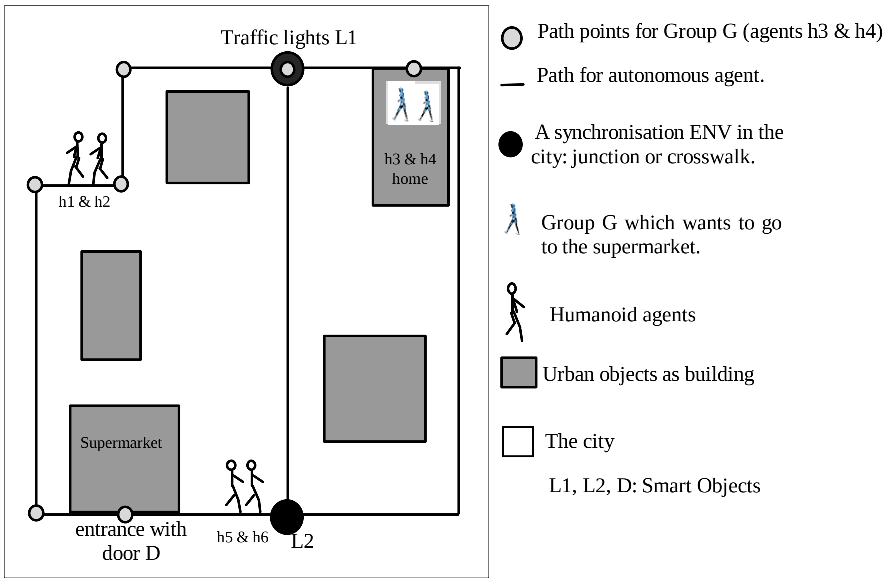
Traffic lights L1
h1 & h2
h3 & h4 home
Supermarket
entrance with door D
h5 & h6
L2
Path points for Group G (agents h3 & h4)
Path for autonomous agent.
A synchronisation ENV in the city: junction or crosswalk.
Group G which wants to go to the supermarket.
Humanoid agents
Urban objects as building
The city
L1, L2, D: Smart Objects

Figure 11 - ***Simple representation of agent’s routing in the block***

When the RBBS module sends a message “Group G : go to the supermarket”, the CROWD module receives it and asks CITY to know how the Group G located in (x0, y0, z0) can go to the supermarket (group goal) walking only on allowed regions. The city database locates the start and the end points (location of the place named supermarket in the city) and computes the best path. This path is a list of surfaces and points to reach and it is sent back to CROWD. With this known path, the CROWD module guides the Group G providing points to be reached sequentially, guaranteeing that both h3 and h4 will not walk outside the given regions. During the agents’ displacements, the crowd module requires crowd (internal to CROWD module) and individual (AGENTlib) perceptions in order to detect close smart objects. When agents are near to the smart object L1, the crowd module internally calls the smart object functionality to manage the interaction between L1 and each agent. When the object-interaction is finished the crowd module continues to inform the agents with locations to reach in order to arrive at a programmed goal.

Starting from the high level command “Group G : go to the supermarket”, a more detailed sequence of exchanged messages is listed in table 2.

Sender	Received by Controller and send to module	Message or Action
RBBS	CROWD	Message : “Group G: go to supermarket ” *CROWD analyses the message, defines the location of the Group G and creates a new message.*
CROWD	CITY	Message : “path from position (x0, y0, z0) to the supermarket” *CITY locates the starting point in the city database and the place named supermarket. Then, a collision-free path between these locations is computed.*
CITY	CROWD	Message : “path (lists of connected surfaces and points to walk)”
CROWD actions: Group G follows the path, using the surfaces associated, and calling AGENTlib perceptions to detect some smart object in the way.		
Activated perceptions in AGENTlib by CROWD. Get information that agent traffic light L1 is near. Connection between CROWD and SMART OBJECT for the interaction. Internal call to SMART OBJECT for object L1 which handles the synchronisation with other agents. When it is done, the control is given back to CROWD. Then, the CROWD module continues to reach the path points.		
The Smart Object door D has the perception of agents of Group G near the door D. The door D has a closed status, so that it automatically opens. Status door D becomes open and the agents of Group G are guided to pass the door		
CROWD continues to reach paths’ points.		
CROWD	CITY	“Group G position (x,y,z) in supermarket ?” *CITY extracts the list of ENVs corresponding to the point (x,y,z) in the hierarchy and verifies whether the point is in the supermarket ENV.*
CITY	CROWD	“Group G in supermarket”
CROWD	RBBS	“Group G arrived in the supermarket”

Table 2. ***Message exchange to achieve the task “go to supermarket”***

Figure 12 shows the action decomposition and their links with the different modules. All actions represented in the figure are generated from the main high level order: “go to the supermarket”. This order is then gradually decomposed into actions until the AGENTlib level is reached. Note that in this case study, the only low level motion motor called by SMART OBJECT concerns the walk motor. However, other interactions with smart objects can also be associated with other Body Motion Control motors of AGENTlib using for example, some inverse kinematics methods.

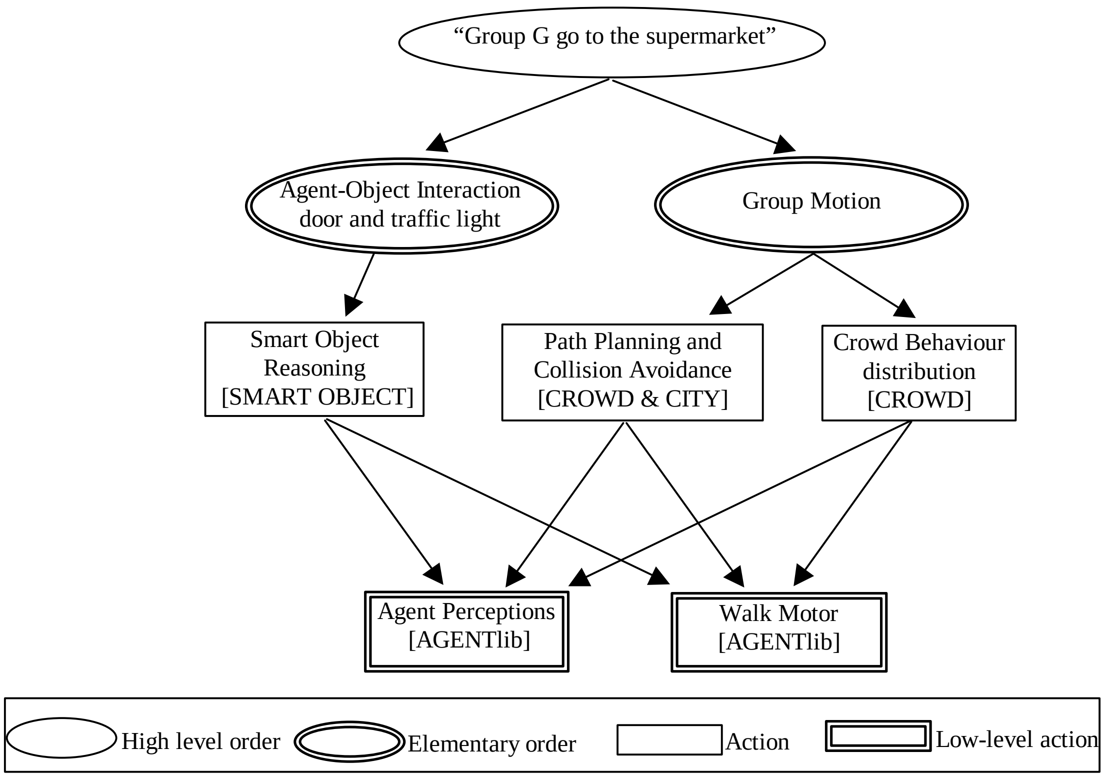
"Group G go to the supermarket"
Agent-Object Interaction door and traffic light
Group Motion
Smart Object Reasoning [SMART OBJECT]
Path Planning and Collision Avoidance [CROWD & CITY]
Crowd Behaviour distribution [CROWD]
Agent Perceptions [AGENTlib]
Walk Motor [AGENTlib]
High level order
Elementary order
Action
Low-level action

Figure 12 – ***Levels of action complexity***

Figure 13 shows a snapshot of a park in the inhabited virtual city simulated by the system. The CITY module can calculate many paths around the city allowing the humanoids to walk on sidewalks and crosswalks. Agents walk autonomously following their autonomous crowd intentions and can be guided by eventual RBBS tasks.

As an extension, a bus has been implemented and synchronised with the virtual humans using the crowd events and reactions in order to inform them when the bus arrives. As a result the crowd management is able to perform the appropriate reaction, i.e. get on the bus and attach the crowd motion to the bus movement.

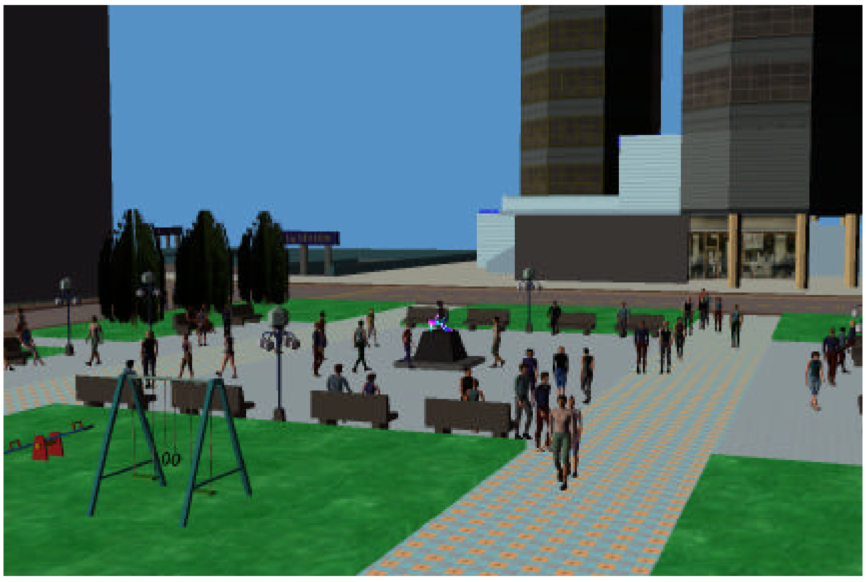

Figure 13 - ***View of humanoids in the city***

CONCLUSIONS

In this paper we have presented the integration of various key modules to perform simulations in an urban environment. We have shown how we separated modules that have a straight connection with the graphical output from those that can run independently, connected by a client-server communication system.

In order to minimise the volume of message exchange, our Controller has been designed to emphasise the management of messages concerning queries between the modules RBBS, CROWD, and CITY, that are messages concerning high level information. Other low-level messages are internal, and so are treated with function calls. For example, data related to agents, as human joint positions and orientation, are directly controlled by AGENTlib. Modules CROWD and SMART OBJECT directly access other important features as perceptions, without message exchange.

In the same frame of mind, the behavioural control is distributed along three co-operating modules efficiently handling specific tasks: the rules-based behaviour deals with wide-range decision making, the crowd module distributes pre-programmed crowd behaviours among autonomous groups and individuals and deals also with the direct control of behaviours provided by RBBS, and the smart object module deals with operating specific devices.

Moreover, we have described how we build semantic environments constituted of interactive objects and a hierarchical virtual city. As a consequence, we can note that the work of urban model designers is evolving significantly as they also have to create the basis of an intelligent city database. The encoding of semantic information in the virtual environment is the prerequisite to efficient simulation of populated virtual city.

Our current prototype implementations show that the open aspect of the client-server architecture is very important for controlling the simulation. Besides the pre-programmed RBBS system, the use of additional modules to send messages directly typed by the user is an important feature that has been included. Such external modules can also query for simulation states in order to permit data analysis.

Another advantage of our architecture has shown to be its suitability for interactive simulations with a relatively big number of agents. Comparing to standard agent-based approached which try to mimic the real world by modelling each agent as a separated asynchronous process, our architecture organises independent and time-consuming modules in separated processes. In this way, our approach clearly minimises messages exchange and thus permits faster simulations.

Our experience has shown that this system facilitates the creation of simple “every-day” behaviours for a relatively large number of agents, with a good level of control.

ACKNOWLEDGEMENTS

The authors thank to Dr. Srikanth Bandi, Amaury Aubel, and the anonymous referees for their suggestions. This research was supported by the Swiss National Research Foundation and the Office Federal for Education and Science (ESPRIT project eRENA), FUNDEPE, CNPq and CAPES.

REFERENCES

Aufaure Portier, M. A., P. Berthet, and J. L. Moreno. 1994. Optimized Network Modelling for Route Planning. *Proceding of the 4th Eur. Conference on GIS, EGIS’94 Paris, 1817 – 1824.*

Baerlocher, P., and R. Boulic. 1998. Task priority formulations for the kinematic control of highly redundant articulated structures, *IEEE IROS '98,Victoria (Canada), pp. 323-329.*

Becheiraz, P., and D. Thalmann. 1996. A Model of Nonverbal Communication and Interpersonal Relationship between Virtual Actors. *Proceedings of Computer Animation, June 3-5, Geneva, Switzerland.*

Becheiraz, P., and D. Thalmann. 1998. A Behavioral Animation System for Autonomous Actors Personified by Emotions, *Proc. of first Workshop on Embodied Conversational Characters (WECC’98), Lake Tahoe, California.*

Benford, S. D., C. M. Greenhalgh and D. Lloyd. 1997. Crowded Collaborative Virtual Environments. *Proceedings of ACM Conference on Human Factors in Computing Systems, Atlanta, Georgia, US, March 22-27.*

Bordeux, C., Boulic, R. and D. Thalmann. 1999. An Efficient and Flexible Perception Pipeline for Autonomous Agents, *Proc. of Eurographics’99, Milan September 1999*

Boulic, R., Z. Huang, and D.Thalmann. 1994. Goal Oriented Design and Correction of Articulated Figure Motion with the TRACK System. *Journal of Computer and Graphics, v.18, n.4, pp. 443-452, Pergamon Press, October.*

Boulic, R., P. Becheiraz, L. Emering, D. Thalmann. 1997. Integration of Motion Control Techniques for Virtual Human and Avatar Real-Time Animation. *ACM VRST'97, Lausanne, Switzerland, 111-118, ISBN 0-89791-953-x.*

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

Brogan, D. and J. Hodgins. 1997. Group Behaviors for Systems with Significant Dynamics. *Autonomous Robots, 4:137-153.*

Chaib-draa, B. 1997. Connection Between Micro and Macro Aspects of Agent Modelling. Proceedings of Autonomous Agents.

Donikian, S. 1997. VUEMS : A virtual Urban Environment Modeling System. *Computer Animation 97, 127-133.*

Doyle, P., and B. Hayes-Roth. 1997. Agents in Annotated Worlds. *Report No KSL 97_09, Knowledge Systems Laboratory, Stanford University, California.*

Farenc, N., R. Boulic, D. Thalmann. 1999. An informed Environment dedicated to simulation of virtual humans in urban context. *Proceedings of Eurographics'99, to appear.*

Fuji, T., K. Imamura, T. Yasuda, S. Yokoi, and J. Toriwaki. 1995. A Virtual Scene System for City Planning. *Computer Graphics : Development in Virtual Environments 1995, 485-496.*

Guzzoni, D., A. Cheyer, L. Julia, and K. Konolige. 1996. Many Robots Make Short Work. SRI International, AAAI Robot Contest.

Howard, K. R. 1997. Unjamming Traffic with Computers. *Scientific American October.*

Ingram, R., S. Benford, and J. Bowers. 1996. Building Virtual Cities: applying urban planning principles to the design of virtual environments. *VRST'96.*

Jepson, W., R. Liggett, and S. Friedman. 1996. Virtual Modeling of Urban Environments. *Presence, 5:83-95.*

Kallmann, M. and D. Thalmann. 1998. "Modeling Objects for Interaction Tasks". *Proceedings of the Eurographics Workshop on Computer Animation and Simulation (EGCAS'98), Lisbon, Portugal.*

Kallmann, M. and D. Thalmann. 1999. A Behavioral Interface to Simulate Agent-Object Interactions in Real-Time, *Proceedings of Computer Animation 99, IEEE Computer Society Press.*

Modeling Objects for Interaction Tasks". *Proceedings of the Eurographics Workshop on Computer Animation and Simulation (EGCAS'98), Lisbon, Portugal.*

Levison, L. 1996. Connecting Planning and Acting via Object-Specific reasoning. PhD thesis, Department of Computer & Information Science, University of Pennsylvania.

Mokhtarian, P. L. 1997. Now That Travel Can Be Virtual, Will Congestion Virtually Disappear?. *Computers Scientific American, October.*

Musse, S. R., and D. Thalmann. 1997. A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis. *Proceedings of the Eurographics Workshop on Computer Animation and Simulation, September, Budapest, Hungary.*

Musse, S. R., C. Babski, T. Capin, and D. Thalmann. 1998. Crowd Modelling in Collaborative Virtual Environments, *ACM VRST'98, Taiwan.*

Neves, M. C., and E. Oliveira. 1997. A Control Architecture for an Autonomous Mobile Robot. *Proceedings of Autonomous Agents.*

Norvig, P. 1992. Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp. *M Kaufmann, ISBN 1-55860-191-0, 946 pages.*

Noser, H., and D. Thalmann. 1996. The Animation of Autonomous Actors Based on Production Rules. Proceedings of Computer Animation, Geneva, Switzerland.

Parry-Barwick, S., and A. Bowyer. 1993. Is the Features Interface Ready?. *In "Directions in Geometric Computing", Ralph Martin ed., chap 4, 130-160.*

Pratt, M. J., and P. R. Wilson. 1985. Requirements for Support of Form Features in a Solid Modeling System. *Report R-85-ASPP-01, CAM-I.*

Rasmussen, J. 1986. Information Processing and Human Machine Interaction: An Approach to Cognitive Engineering. North Holland.

Reynolds, C. 1999. Steering Behaviors For Autonomous Characters. *Game Developers Conference ,San Jose CA, 15-19, March.*

Russ, T. A., R. M. MacGregor, and B. Salemi. 1996. VEIL: Combining Semantic knowledge with Image Understanding. *ARPA Image Understanging Workshop.*

Schweiss, E., S. R. Musse, and F. Garat. 1999. An Architecture to Guide Crowds based on rule-based systems. Autonomous Agents'99, Seattle, Washington, USA.

Tu, X. and D. Terzopoulos. 1994. Artificial Fishes: Physics, Locomotion, Perception, Behavior. *Proceedings of SIGGRAPH '94, Computer Graphics, July*.

Wilks, Y. 1986. An Intelligent Analyser and Understander of English. *Readings in Natural Language processing, M. Kaufmann Publishers Inc, ISBN 0-934613-11-7, 664 pages.*

Zeltzer, D., R. K. Addison. 1997. Responsive Virtual Environments. *Communication of the ACM, Vol. 40, Number 8, 61-64, August.*