One Step towards Virtual Human Management for Urban Environment Simulation

N. Farenc, S. Raupp Musse, E. Schweiss,
M. Kallmann, O. Aune, R. Boulic, D. Thalmann

Abstract. In this paper , we present an approach to integrate different applications for a virtual human simulation in a complex environment. We present the design of one system and the integration of the various features: creation and use of information extracted from the environment and crowd management.

1 INTRODUCTION

For several years, we have been working on modelling and simulating realistic virtual humans. All simulations were made using environment without semantic or behavioural indication. This project aims at performing simulations of autonomous agents or crowds within complex scenes including some degree of interaction with the environment. This paper presents different aspects of human and object simulation and their integration in an urban environment using information extracted from a virtual city.

To make more realistic simulations the environment must integrate some semantic notion about specific areas. This corresponds to natural human analysis using urban basic knowledge like " a side walk is for pedestrians ". Our environment is decomposed into entities corresponding to semantic or functionality information. Several works have been done on virtual city to reconstruct real city with image processing or sophisticated tools [DON97, RUS96, JEP96], or to use urban context to deal with traffic problem [JEP96, HOW97, MOK97], urban displacement [ING96, JEP96, AUF94] or city modelling [FUJ95, JEP96]. Our aim is to have a hierarchical urban model with integrated knowledge adapted to human behaviour simulation. In the context of crowd management, we need this knowledge to drive the motion of autonomous groups and to synchronise different actions like object interaction, collision avoidance, etc. For the autonomous crowds , we have used the flocking model [REY 87] [MUS 97]. In our case of Intelligent Virtual Environment (IVE), the information has to be interpreted by the virtual creature. The graphical data base conception has to integrate semantic information within the objects. To provide virtual humans with a range of simple responses to daily life event stimulati, we present an approach supporting rules_based behaviours. In addition, a rule analyser module allows end users to define their own set of behavioural rules in natural language. At a low level the interactions between virtual humans and objects are integrated in the objects during their modelling phase so as to provide all the low level parameters needed to perform many object interactions.

The paper is structured as follows : in the next section we present the city model and all the associated information. The subsequent section is about the designer viewpoint on the associated modelling constraints; the fourth section presents the modelling of objects interactions; the fifth section introduces human crowd control; the sixth section talks about a rule-based behaviour control to perform more sophisticated human simulation, and the last section deals with the integration of theses application.

2 CITY DATA BASE

What do we need to perform virtual human behavioural simulation in a complex environment like a city? With this question in mind we can assume that urban displacements are very much dependent on geometrical data and urban knowledge. Accessing a specific location in a big scene is not trivial (a point in front of the door in an office, or the location of the crosswalk entry), and the geometric data extraction is very much dependent on the model used. A city is a complex environment. What does that mean? For us a complex environment is a place where information (semantic and geometric) is dense and can be structured and organised using rules. We define urban knowledge for this work as a set of urban structural information and objects (complex environment) usable according to some conventions. We define the urban knowledge as an association between some places (geometric area) and some semantic information.

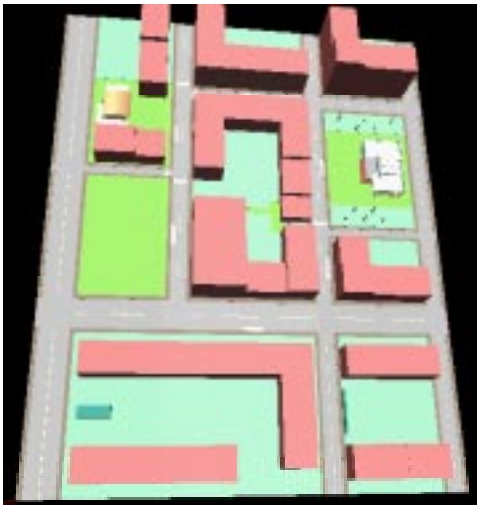

Figure 1 - ***Fly view of a block***

The aim is to be able to deduce what action is possible according to this information. For example, in a city, we can annotate sidewalks like areas for pedestrian mobiles, so that in such places they can walk or have social encounters [DOY97]. The city data base has to informe all "mobiles" (objects with mobility such as pedestrians, cars, buses, bicycles) where they can circulate and in some cases even if pedestrians can cross an area, they have to be synchronised with other mobiles depending on the traffic signals status. Then the environment has to inform mobiles that they are in a synchronisation area and have to deal with other mobiles. A problem is the management of such large quantity of data consisting of thousands of crosswalks, specific points and objects. How to distinguish them, and access their own information (geometrical and specific ones)? How to organise them to access information in a minimum of time? From these observations our approach is to define some areas. The areas are either subdivided into sub_areas or

EPFL Swiss Federal Instute of Technology LIG Computer Graphics Lab CH 1005 Lausanne VD Switzerland

grouped, depending on the 'level of information' in the same way as in a geographical map, we decompose a large area into sub_areas with information inherent to the level of description. (Figure 2). At the city level we have information about the main roads to cross the city and at the block level we can find information on the streets, the locations of parcels and junctions. This decomposition tidies up and links the semantic data.

In order to sort and classify data one solution is to structure the information in a hierarchy. The city has been decomposed into several areas, depending of their geographical and functional properties. At each level in the environment 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 pedestrian informing about sidewalks or pedestrian crossing or objects to avoid collisions. Specific areas of the scene are defined as "Environment Entities" *ENV* because they hold geometrical and semantic information. In the city, at the first stage we start at the "block" level which is under the quarter level. A block is composed of streets, cross-roads and parcels. All these entities are themselves composed of sidewalks, pedestrian crossings, rolling (or traffic) way for the street, travelling or not travelling areas for parcels. Here is a drawing representation of the ENV hierarchy :

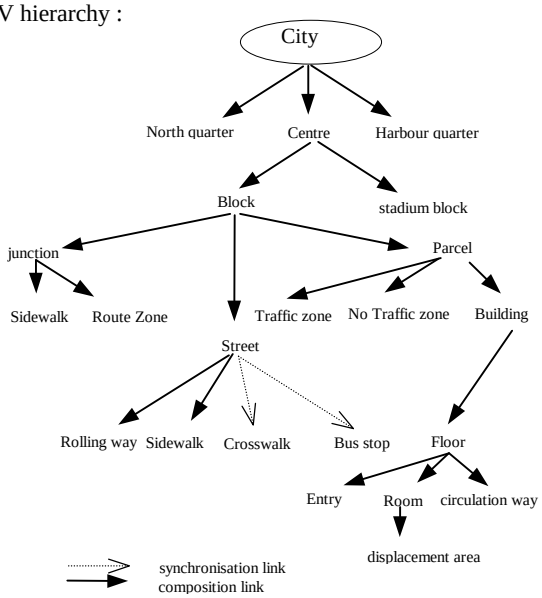

Figure 2 - City hierarchical decomposition in ENV

To create a data base with *ENV* we use a graphic scene. In the scene some objects are named in such a way as to detect them as *ENV* and analysed to extract their name and their geometrical data. From a set of *ENV* we calculate some dependency information (according to the hierarchical model), all the geometrical information (walking area is defined in a parcel world) and the connexity information between *ENV*. The connexity data is used to know the possibilities of reaching some place from one point depending on the mobile type considered. Some simple semantic information is associated with ENV (area for car, pedestrian or bus) showing who can use which *ENV*. Other information can be added concerning more specific action such as "playground". The Figure 1 represents a view of our modelled block. The Figure 3 shows the connexity graph for the *ENV* used by pedestrians mobiles; we can see the objects only usable by pedestrian mobiles (crosswalk, sidewalk and pedestrian traffic zone. The cylinders are points allowing movement from one *ENV* to the next and correspond to the connexity graph links between the nodes (ENV). The naming convention of objects is based on a precise definition and it's an important part of the work. Objects are parsed using their names, and put in the ENV hierarchy according to the information provided by the name analysis. The next section concerns the designers view point about this kind of constraint.

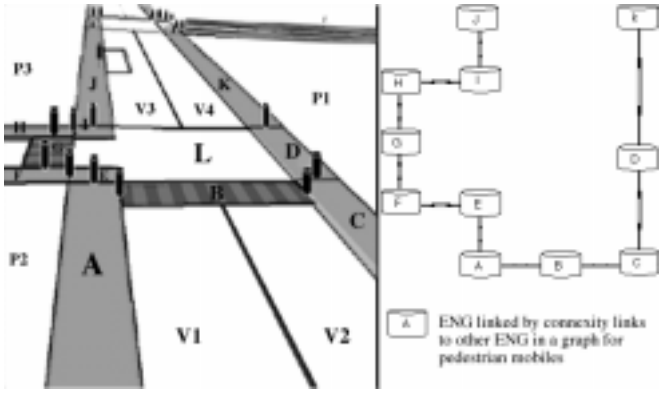

Figure 3 - Connexity graph for pedestrian ENV represented by real visual objects in the block, and by the graph associated.

3 THE DESIGN ASPECTS

In an interactive walkthrough, only the geometry is relevant while the semantic and spatial aspects of the graphical database are interpreted in an intuitive way by the user. To provide an autonomous behaviour to the IVE agents, these have to be informed about the significance of the geometry they are dealing with. In this case the absolute necessity is to build some semantic tools, which inform the different components of the virtual world about their own functionality. As in the real word an IVE needs common rules. It means we have to conceptualize of such environments in a precise way to ensure that the designer team will produce a graphical and semantic environment usable by developers. Which syntax ? The preoccupations of the developers and 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 the designers are prone to acts of omission or creating wrong semantic entries, hence the rules involved have to be as simple as possible. In most of the cases it would be better to automate the translation between the graphical information and the needs of the routines, such as those used for collision detection or recursive naming.

3.1 Virtual Creature Behaviours and the Objects

It's important to make the difference between the behaviour of a virtual creature and the potential functionality of an object composing the virtual world. For example, the potential functionality of a signpost is to inform, and of a door is to open itself in a specific way. The behaviour of an humanoid is to get and interpret the information on the signpost or to turn the knob of the door as the case may be. On the other hand, a virtual dog would stand on its leg by the signpost or scratch the door. Although all this information is important, the implementations are quite different. Objects functionality should be defined at the time of the virtual world design while creature behaviour has to be done later, since it needs the scene and objects definitions.

3.2 Different Categories of Objects

As every objects in the real world, have a particular functionality, in an IVE, the concept of an object also has to be precisely defined regarding the type of information it owns: visual, spatial, behavioural.

The Displayed Objects : The graphical database is composed of different kinds of objects some of which are only used to perform some specific computations such as collision detection or behaviour information. Hence we call Displayed Objects only those which are visualised in the final scene.

The Objects Box : These are associated with walking surfaces and used to perform local collision detection in this area. In doing so, we have the advantage of optimising database traversal, avoiding any object that is present on the walking surfaces. These objects could correspond to Displayed Objects or not, in the later case they are used only for calculation.

The Smart Objects : Some objects of the 3D scene need functionality, which could either be internal to the object (the door movement) or external (the signpost). In the latter case it informs the other agents of the IVE. The Smart Object refers to two kind of data files : the graphical description and the functionality description. The next section describes this class of objects.

4 MODELLING SMART OBJECTS

The main objective here is to construct a complete framework where the designer of an object can model not only the object's geometry but also extra information like specific parts to interact with and its functionality. In doing so, some simulator program can read object's features that will guide the interaction with a virtual actor. In this way, the designer can have the feedback of a simulator program to adjust the object design, taking control of the whole cycle of the object design and testing. This approach of including information inside the object description is known in the literature as Feature Modelling. As commonly stated, a feature is a region of interest on the surface of a part [PRA85] [PAR93]. In our case, we are trying to identify "interaction features"; these are parts in 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 an object's functionality, with specific data such as hand shapes to manipulate them, and also dedicated locations to interact with them. In this way, the application-specific object reasoning will have sufficient information to perform its tasks with the object.

4.1 Modelling Interaction Information

We term *Smart Object* as the object being modelled along with its interaction and functionality information. A typical Smart Object description contains, among others features, a collection of hierarchical *parts*, *actions*, *commands*, *gestures* and *functionality rules*. Each action represents a movement, a change in the material attributes or a sound to be performed. An action is independent of a part and can be parameterized. A command is an entity that links a part to an action that may be triggered after an actor's gesture. Object's functionality rules are described by a list of *behaviours* and *state variables*. Each behaviour is described by a list of instructions, in a script language format..Examples of such insctructions are: a variable value to change/ check, a gesture to be performed, or a command to execute.

4.2 Smart Objects and Virtual Humans

In the case of a virtual city environment, the "intelligence" of the actor should not take care of some low level decisions. Take the example of an actor passing through a door. In this case, it is not worthwhile to do a complex reasoning algorithm to decide things like which hand shape best fits the door knob. In this case, it's simpler to store the best pre-defined hand shape to be used in the door object itself. And this should be done in the design phase of the door. 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, each application-specific reasoning can decide how far object's information is to be used or 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 itself the 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, an automatic door might have a perception to test if there is some actor near it, 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 main needs of the application-specific reasoning are well defined, the Smart Object framework provides a versatil way to model many possible interactions.

5 HUMAN AND CROWD CONTROL

Simulation of human crowds for populating virtual worlds provides a more realistic sense of virtual group presence [BEN97]. There are several approaches to model autonomous crowds [BOUV97] [BRO97]. In a Virtual Urban Environment, it is useful to simulate autonomous populations, i.e. agents which have a kind of environment knowledge (section 2) and are able to move and interact with their environment. This type of crowd is called ***autonomous crowd*** that must obey the environment specifications, the programmed interaction with objects, etc. This information must be specified at the beginning of the simulation.

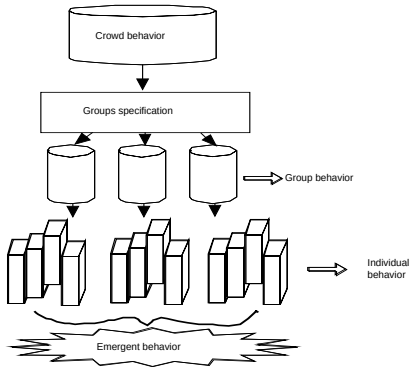

Figure 4 ***The architecture of the CROWD system***

The other kind of crowd control present in our work is the ***guided crowd*** which must always be guided by a leader. This leader can be an avatar (virtual human representation controlled by an end user) or an intelligent autonomous agent (section 6). The guided crowd responds to dynamic information (which can change as a function of time), such as the interaction with the objects, the displacement to reach a specific point of interest, etc provided by the leader. We have defined a crowd as a set of groups composed of human agents.

The crowd behaviour is distributed across a number of groups, and the individual behaviours conform to corresponding group specification [MUS 97].The data structure of our crowd model is presented in Figure 4. At a highest information level, a crowd is treated as a single entity which is formed by agent groups which 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 [BOUL97];
- following motion (group's ability to follow another group);
- group control (which can be autonomous or guided).

Some group behaviours can occur if the simulation environment is known (section 2), It means the physical locations of the goals or interest points, positions and dimensions of each obstacle, etc are known. We can see in figure 5 some interest points (represented by cylinders) which are distributed for the different groups to construct the group's paths.

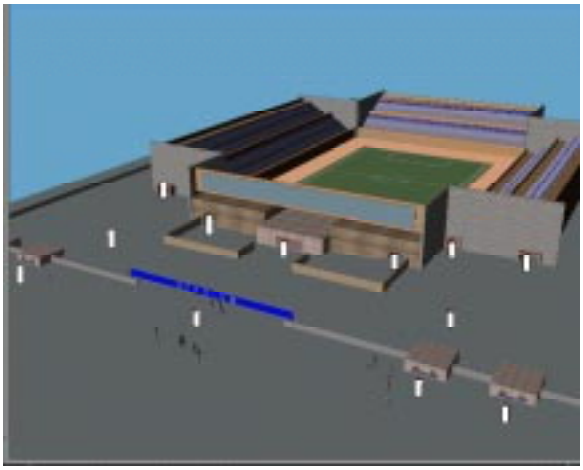

Figure 5 - ***Interest points and goals to drive the crowd***

Based on these goals, the autonomous crowd motion occurs always considering the behaviours of all the others groups. Figure 6 shows one image of this simulation.

Figure 6- ***Different groups entering in the stadium***

6 RULE-BASED BEHAVIOR CONTROL

The rule-based behaviour control guides the crowd in the city. It sends high level orders like "Go to supermarket" choosen according to behavioural rules and crowd states.

Main criteria of the rule-based behaviour control

Type of behaviour : In this part of the behaviour control, we are treating daily life behaviours like either "Need for buying the bread", or "Wish to visit a museum", or "Replying to a phone call".
Agent relations : The features of this module are adapted not only to object and humanoid interactions but also to high level humanoid relationships.
Behaviour treatment : According to the system state, the module selects rules which produce state changes [NOR92].

Rule syntactic analyser

Rules are interpreted by the syntactic analyser and added in the behavioural rules base (Figure 7). We have developed a syntactic analyser to allow non programmers to design daily life simulations by defining a set of rules in a pseudo-natural language. Indeed, the user can write his/her own rules in a file in order to simulate not only daily life behaviours, but also human characters. This analyser reads and translates the user's behavioural rules for the system. The system, then, works with the rule base composed of predefined basic rules and user's rules. The analyser the rules to have a fixed syntax and be based on the following semantic elements.

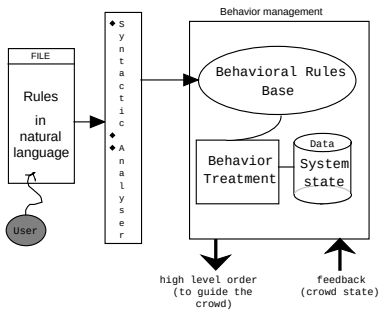

Figure 7 : ***General organisation***

Semantic elements

The basic semantic elements [WIL86] used in our system are :

1 - ENTITIES : agents (i.e. human being and physical object)
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 characters
5 - CASES : source, location, direction, goal, in, possessed

6.1 Rules architecture

Data structure

The rules are composed of two parts [NOR92] : 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)
In our rule based system, the premises tests 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.

A daily life behaviour treatment

The rules are structured into different sets of rules organised in trees. The role of the root rules - called 'start rules' - is to determine

which daily life order occurred. The system, then, deduces a sequence of responses, by exploring the rules of the tree which root was selected. When a rule premise is validated, the conclusion produces some activation and desactivation of actions for the virtual agents. Consequently, the state of the system changes and only the 'next rules' down in the tree are then taken into consideration. A daily life behaviour treatment ends with the validation of a 'leaf rule' (i.e. with no 'next rule').

7 - A CASE STUDY

Now that all the different applications have been briefly presented, we examine a concrete example to see how data can be exchanged to perform realistic simulation of humanoids in an urban environment. The test case: "an autonomous agent named h4 wants to go from its home to the supermarket". We have four agents, three smart objects which are traffic lights L1, L2 and door D, and three possible paths. The agent's wish is decided by the rule-based behaviour control. The rule-based behaviour control asks the city data base to know how h4 can go to the supermarket; it receives some lists of ENV/points. These lists start (normally not for the first one) with an entrance in a synchronisation area which means the autonomous agent (the crowd leader in our case) has to use its perception or to have a way to know what it can do. The crowd module is the agent manager making them walk and avoiding collisions. During the displacement, the autonomous agent can meet a smart object in which case, depending on the smart object type, the interaction can influence the smart object state or the autonomous agent or both. In the case where the agent goes to a door, we can imagine that the door is automatic and has to open when an agent is near. The Figure 8 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 is arriving at the supermarket.

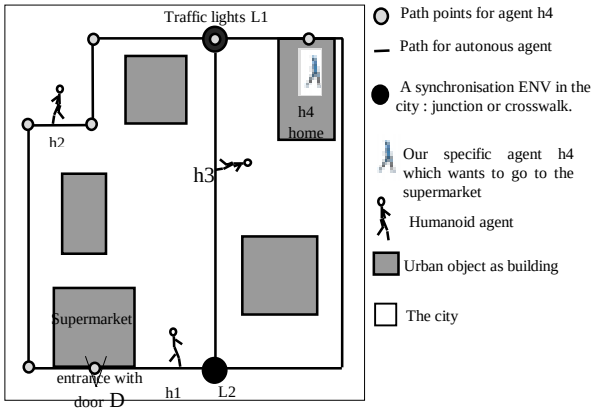

Figure 8 - ***Simple representation of agent's routing in the block***

How to communicate in an open system with applications running simultaneously and using information from others tools without too much data exchange? We have thought about a kind of " client_server " system where all messages are sent to a central receptor which analyses the label message and redirects it to the proper recipient.

The main drawbacks of this method are the volume of message 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 run into some problems of synchronisation between messages and the applications treatment. As an alternative, we can minimise the data transmitted by devoting low level decision to other application belonging to the agent application. The crowd humanoids and the smart objects are defined as agents in a software architecture called AGENTlib [BOUL97]. The AGENTlib maintains a data base concerning all the agents created during one simulation. We can also define some perception, which allow agents to extract information about neighbours for example. The perception can be specialised by adding a selection criteria for some types of agents in a limited space. With these features we can refine our model for the integration as follow. If we analyse the different applications as application linked or not with AGENTlib, we can see that the data base for the city and the rule-based behaviour control are independent and can run with only connection with the controller. The Crowd management and the smart object application are defined on top of AGENTlib layer. To manage situation like synchronisation with traffic lights we can use some specialised action dedicated to synchronous area management. This specialised action is a predefined behaviour activated for each agent arriving in a synchronous area. Then in this case all messages for traffic lights management are internal and don't pass through the controller. In such a configuration, the crowd application gives access to the perception for the leaders, and if needed delegates the control to the smartobject application or to a specialised action. The next Figure 9 represents our communication approach.

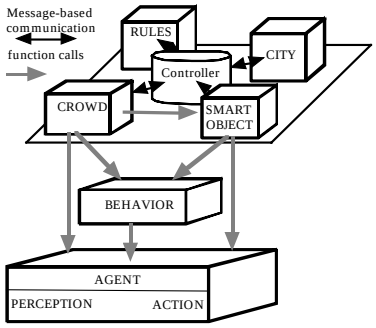

Figure 9 - ***Representation of the system with specialised action***

If we analyse " Agent X wants to go to the supermarket " we can find the following format data exchange :

Sender	Receiver via controller	Message or *Action*
RBB	City	" path to supermarket form (x0, y0) for Agent X " *calculate for the best path*
City	Crowd	" list_path(lists of points) for Agent X " *Group X walk to (xn, yn,zn)*
Activation perception in AGENTlib by the crowd control. Get information that agent traffic light L1 is near. Internal call to a specialised action which handles the synchronisation with other agents. When it is done the control is given back to the crowd management. The crowd continues to reach the path points. smobj : perception of Agent X near the door D status closed smobj : door D open, new status door D open. The crowd continue to reaches the path' points		
Crowd	City	" Agent X arrived at the end of the path position (x,y,z) in supermarket ? "
City *define the list of the ENV corresponding to this point in the hierarchy and verification whether the point (x,y,z) is in the supermarket ENV.*		
City	RBB	" Agent X arrived in the supermarket "

Abbreviations :

- rule-based behaviour : RBB
- data base system for the city: City
- crowd management : Crowd

- smart object application : Smobj

In this configuration the controller manages messages concerning only queries for the rule-based behaviour or for the data base system for the city. Others messages are internal and so are some function calls . The internal perception avoids some messages concerning smart-objects status. The specialised action deals with the synchronisation problems internally. The drawbacks is integration of more applications and perhaps more memory space allocated for specialised action. The controller still has to parse messages to send good query depending on the incoming message. As the interesting points we can note that all these applications are well integrated, each has a well defined job, messages passing through the controller are very limited and the use of state finite automaton is a good solution for synchronisation problems.

Some results of a simple integration

Below, there is a picture of simple integration of city environment data and crowd application. Some paths have been defined to move around the city and the humanoids are walking on sidewalks and crosswalks. A bus has been implemented and it is synchronised with the humanoid in a very simple way using tags to inform of each other presence. In such cases, the specialised module can synchronise agent and bus.

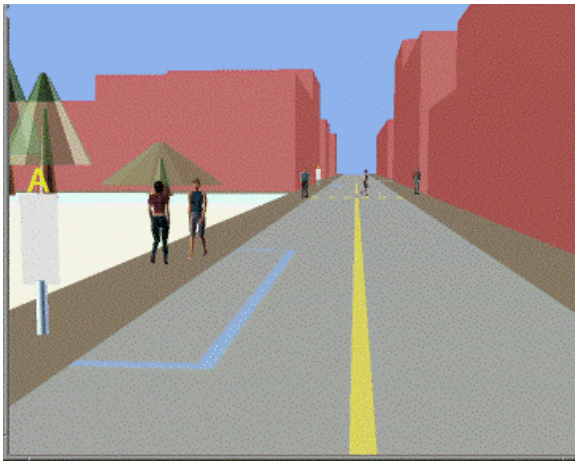

Figure 10 - *View of humanoids in the city*

8 - CONCLUSION

In this paper we have presented our various applications and their integration to perform realistic simulation in an urban environment. We have started the integration between the data base system for the city and the crowd management application, and also the crowd management and the rule-based behaviour control. Our future goal is to create the controller and some functions to treat message in the first stage between the crowd application and the rule-based behaviour. The future integration of other applications will depend on the quantity of messages exchanged. This integration has some consequences on the development of all the applications involved. Besides, as the work of 3D designers is constantly evolving, they no longer have the choice of staying under the shadow of the artistic rules. Their understanding of the technological environment has to be strong enough to manage intelligent databases. Tomorrow's designers will learn to use high level syntax to be able to design graphical, behavioural and semantic environments.

ACKNOLEDGEMENTS

Thanks to Srikanth Bandi for its proof reading, Joaquim Esmerado and Luciana Porcher Nedel for their help concerning the paper production.

REFERENCES

[AUF94] M. A. Aufaure Portier, P. Berthet, J. L. Moreno (1994), "Optimized Network Modelling for Route Planning", *Proceding of the 4th Eur. Conf. on GIS, EGIS'94 Paris 1994 p.1817 - 1824*

[BEN97] S. D. Benford, C. M. Greenhalgh and D. Lloyd. "Crowded Collaborative Virtual Environments, *Proc. 1997 ACM Conference on Human Factors in Computing Systems (CHI'97), Atlanta, Georgia, US, March 22-27, 1997*

[BOUL97] R. Boulic, P. Becheiraz, L. Emering, D. Thalmann. "Integration of Motion Control Technique for Virtual Human and Avatar Real-Time Animation". ACM VRST'97, Sept. 97, Lausanne Switzerland, pp111-118, ISBN 0-89791-953-x

[BOUV97] E. Bouvier, E. Cohen and L. Najman. "From crowd simulation to airbag depolyment: particle systems, a new paradigm of simulation". *Journal of Electronic Imaging 6(1), 94-107 (January 1997).*

[BRO97] D. Brogan and J. Hodgins. "Group Behaviors for Systems with Significant Dynamics". *Autonomous Robots, 4, 137-153. 1997*

[DON97] S. Donikian (1997) VUEMS : A virtual Urban Environment Modeling System *in Computer Animation 97 from p.127 to p.133*

[DOY97] P. Doyle, B. Hayes_Roth (1997), "Agents in Annotated Worlds", *Report No KSL 97_09 Knowledge Systems Laboratory Stanford University California 94305*

[FUJ95] T. Fuji, K. Imamura, T. Yasuda, S. Yokoi, J. Toriwaki (1995), "A Virtual Scene System for City Planning", *Computer Graphics : Development in Virtual Environments 1995 from p.485 to p. 496*

[HOW97] Kenneth R Howard (1997), "Unjamming Traffic with Computers", *Scientific American October 1997*

[ING96] R. Ingram, S. Benford, J. Bowers (1996), "Building Virtual Cities : applying urban planning principles to the design of virtual environments", in *VRST'96 from p.83 to p.95*

[JEP96] W Jepson, R. Liggett, S. Friedman (1996), "Virtual Modeling of Urban Environments", *Presence Vol. 5 Winter 1996 p.83 - p.95*

[MOK97] P. L Mokhtarian (1997), "Now That Travel Can Be Virtual, Will Congestion Virtually Disappear?". *Computers Scientific American October 1997*

[MUS97] S. R. Musse and D. Thalmann. "A Model of Human Crowd Behavior : Group Inter-Relationship and Collision Detection Analysis". *Proc Workshop of Computer Animation and Simulation of Eurographics'97, Sept, 1997. Budapest, Hungary*

[NOR92] P. Norvig, Paradigms of Artificial Intelligence Programming : Case Studies in Common Lisp, 1992, M Kaufmann, ISBN 1-55860-191-0, 946 pages

[PAR93] Parry-Barwick S. and Bowyer, A., "Is the Features Interface Ready?", *In "Directions in Geometric Computing", Ralph Martin ed., chap 4, 130-160*

[PRA85] Pratt M. J. and Wilson P. R., "Requirements for Support of Form Features in a Solid Modeling System", *Report R-85-ASPP-01, CAM-I, 1985*

[REY87] C.Reynolds. "Flocks, Herds and Schools: A Distributed Behavioral Model". *Proc. SIGGRAPH'87, Computer Graphics, v.21, n.4, July, 1987*

[RUS96] T. A. Russ, R. M. MacGregor, B. Salemi (1996), "VEIL: Combining Semantic knowledge with Image Understanding", *ARPA Image Understanging Workshop 1996*

[WIL86] Y. Wilks "An Intelligent Analyser and Understander of English" *11 pages - Readings in Natural Language processing, M. Kaufmann Publishers Inc, ISBN 0-934613-11-7. 1986, 664 pages*