Intelligent Virtual Environment and Camera Control in Behavioural Simulation

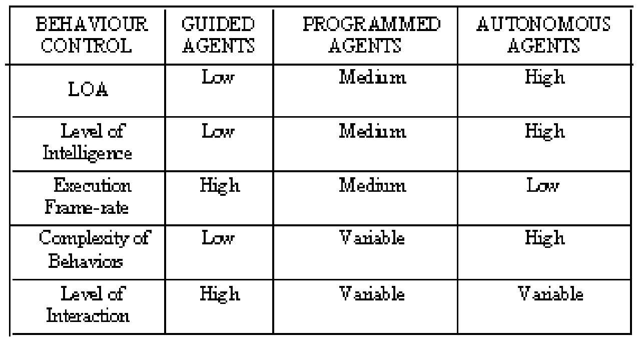
FLAVIA PEREIRA FERREIRA[1], GIORGENES GELATTI[1], SORAIA RAUPP MUSSE[1]

[1]UNISINOS– PIPCA - Masters in Applied Computing
Av. Unisinos, 950 São Leopoldo, RS, Brazil, CEP 93022-000
{flaviapf,gelatti,soraiarm}@exatas.unisinos.br

Abstract. This paper presents a model describing Intelligent Information present in the Virtual Environment, as well as the Camera Control. We are interested in modeling Intelligent Virtual Environments, integrated with our Intelligent Camera control in order to provide information to be used in behavioural simulations. Our main goal is to be able to populate Virtual Environments with virtual agents endowed with different Levels of Autonomomy: from guided to autonomous.

1 Introduction

Various researches on behavioural modeling have mainly focused on several aspects of virtual human control and autonomy. This paper presents a model describing Intelligent Information, required in order to perform behavioural simulations, outside the virtual agents. In our case, we are interested in modeling Intelligent Virtual Environments (IVE) integrated with our Intelligent Camera control (IC) in order to provide information to be used in behavioural simulations. Our main goal is to be able to produce behavioural simulations in Virtual Environments using virtual agents endowed with different Levels of Autonomy (LOA): from low to high. Table.1 shows the classification we used in this work [21].



BEHAVIOUR CONTROL	GUIDED AGENTS	PROGRAMMED AGENTS	AUTONOMOUS AGENTS
LOA	Low	Medium	High
Level of Intelligence	Low	Medium	High
Execution Frame-rate	High	Medium	Low
Complexity of Behaviors	Low	Variable	High
Level of Interaction	High	Variable	Variable

Table 1: Characteristics of different agents' control: Guided agents deal with LOA and autonomous agents with High LOA.

In our model, IVEs are modeled including geometric and semantic information. Geometric information is related to locations of places in the IVE, paths to go to specific locations, regions to walk, objects to avoid, etc. Semantic information deals with behavioural aspects of the spaces: specific actions or movements to be applied, emotional states to be assumed, information related to the characteristics of the space (like comfortable, beautiful, etc). Afterwards, the autonomous agents are able to evolve in the IVE using only the information found in the space which can be used in the decision process.

Concerning the visualization of complex simulations (with multiple agents in complex environments), sometimes it is not possible to cover the events we are interested in only by using an interactive camera, because there are many agents doing different things in different places of the environment, simultaneously. That is the main reason why we proposed the IC module in our simulations.

The goal of the IC module is to find the events that the user wants to monitor during the simulation. The IC input is a script file in which is possible to write the events that the camera has to detect during the simulation. Then, the camera can locate spatial or temporal events which are triggered: the application of specific tasks by the agents, the states of an agent, the location of an agent, etc.

The main idea of this work is to distribute the information and "intelligence", which are needed to simulate and visualize virtual agents, among the Virtual Environment (VE) and the camera control. In this way, the simulation as well as the visualization can be facilitated from the user point of view.

In next sections we discuss related works. In section 3 we present our model as well as the system architecture of our framework including IVE, IC, Simulators and Visualizers. In sections 4 and 5 we discuss more details about the IVE and IC modules. Section 6 draws the results, and section 7 presents some conclusions.

2 Related Works

Many works have presented different ways for controlling the virtual humans in order to improve their ability to evolve in an autonomous way. Several authors agree with the concept of autonomous or ”intelligent” agent requirements: autonomous behaviour, action, perception, memory, reasoning, learning and self-control [4, 5, 20, 34].Also, several methods have been developed in order to model autonomous agents: L-systems [17, 23, 28], vision systems [18, 26]; rule-based systems [14]; learning methods [27, 30], evolution [17], etc.

Becheiraz [3] presented a model of emotion to represent the behaviour of autonomous agents. Hyvrinen and Honkela [15] presented a mathematical model to describe emotional disorders in autonomous agents. Ashida [1] created agent behaviours through statistical analysis of observation data.

Specifically concerning VE, some authors have discussed methods for building informed virtual scenes focused on urban context [8, 10, 12, 31, 32]. Donikian [9] have studied the animation and simulation of autonomous vehicles in urban environments. Farenc [13] described a database model in a framework to simulate virtual humans.

Musse [22] proposed ViCrowd: a model to describe crowd behaviours allowing the virtual agents to interact among them as well as with the VE. ViCrowd’s goal is to simulate groups of autonomous agents endowed with different levels of autonomy. Ulicny [33] presented a model to provide virtual reality training in emergency situations.

Marchand [19] presented a general framework that allows the automatic control of a camera in a dynamic environment based on the image-based control or visual servoing approach. It consists in positioning a camera according to the information perceived in the image, to be ready to automatically respond to modifications of the environment.

Drucker and Zeltzer [11] describe a framework to control an Intelligent Camera in a 3D Virtual Environment, using a virtual museum as prototype. The Camera control is based on an analysis of what tasks are to be required in a specific environment.

This paper presents the IVE and IC modules used in order to provide behavioural simulation and visualization. The main idea of this work is to allow very simple agents to be ”convincingly” animated in large and complex virtual environments. Next sections show information about our work.

3 A Framework to Simulate Virtual Agents

The goal of this framework is to manage information from any VE (IVE module), to provide a convenient visualization in order to verify information about the simulations (IC module) and to visualize virtual humans in different levels of detail (LOD’s). The visualizers we developed are Caterva, RTKrowd and POVo [2]. Figure 1 shows the system architecture including the three behavioural modules: IVE, IC and AgentSim which is responsible by the agents control. The integration among the modules is made using Python [25], consequently all modules are independent of each other.

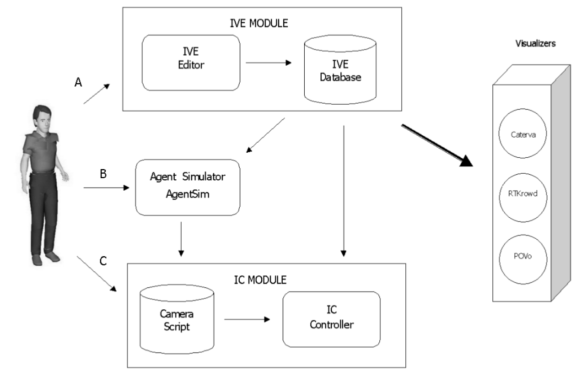


Figure 1: System Architecture. Showing how is the integration among the modules.

The user can edit the semantic information of the VE using IVE Editor (figure 1 - A) as well as to inform the events that the IC module should find during the simulation (figure 1 - C).

The IC should have information from IVE (PLACE position) and AgentSim (agent position) in order to check the triggered events. Once having this information the IC is able to discover if events are occurring. Then, the IC changes camera’s parameters and send them to the visualizers. Further details about the IC, IVE and AgentSim Modules are presented in the next sections.

Concerning the visualizers, Caterva is a simple viewer based on OpenGL based viewer designed to display simulations with a very large number of agents in real-time. It is limited to receive and display the position, orientation and color of each agent (Figure 2).

RTKrowd is a more complex viewer, based on RTK Motion (Softimage) [29], capable of visualizing more complex models as well as animating structures. In addition, it is not limited to display only the agents’ position and orientation: RTKrowd’s actors are capable to perform keyframed animations. This viewer is also targeted to real-time simulations (Figure 3).

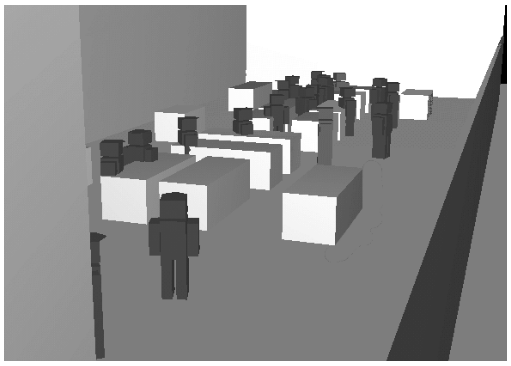


Figure 2: Caterva Viewer.

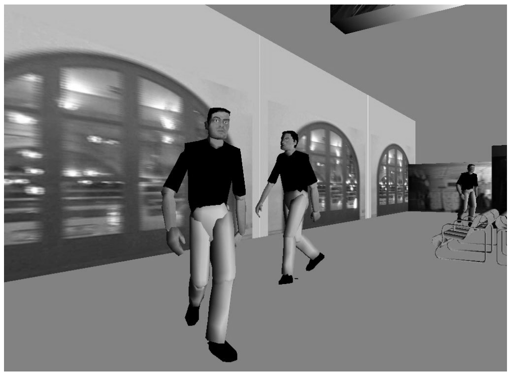


Figure 3: RTKrowd Viewer.

POVo is not a real-time viewer. It writes simulation data to files that can be rendered with the POV-Ray ray-tracer [24]. Furthermore, it is possible to create realistic images and videos from simulations (Figure 4).

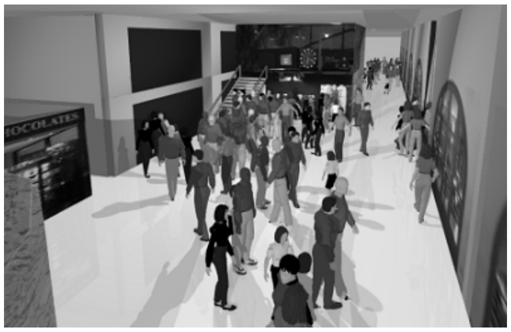


Figure 4: POVo Viewer.

4 Intelligent Virtual Environment

The goal of the Intelligent Virtual Environment (IVE) module is to provide information from the VE in order to be used in behavioral simulations of virtual agents. Using EnvEditor, it is possible to define geometric and semantic information from the VE, which is saved in a VE Database (in XML format). Afterwards, the virtual agents are able to use the information in order to evolve in the IVE.

4.1 EnvEditor

EnvEditor follows the approach discussed in some recent works on the domain [9, 12, 16] in order to remove part of the complexity regarding some complex tasks processed by the agent and to transfer this information to the VE. As a consequence, a virtual agent can ”ask” to the IVE the position of a specific place, the best way to go to a position, the semantic of a place or a path, or other needed information to provide the agents’ evolution in the virtual environment.

The following information can be found in our IVE and informed to the agents during the simulation (further information about these items are described in the next sections).

- Interest Points (IPs) information (IPs are physical positions in the VE where the agents can go).
- IPs connection forming graphs that represent the possible paths to be applied by the agents.
- Visibility graphs generation means the automatic generation of paths to link two or more IPs.
- Criteria specification associated to the paths that could be used by the agents during the simulation.
- FSM (Finite State Machine) specification associated to the space (paths or specific places) containing tasks information (action, motion, state changes),

All this information is saved in XML format database. An example is shown in the annex.

4.1.1 Interest Points (IPs)

An IP is a geometric location in space that defines an interest point for the agent, that is, a point where the agent can pass [22].

Each IP is surrounded by a set of points that form a limit region around the IP. This region is used to distribute the agents in the case of multiple agents simulation. These region points can be connected among themselves, forming access regions between one IP and other (Figure 5).

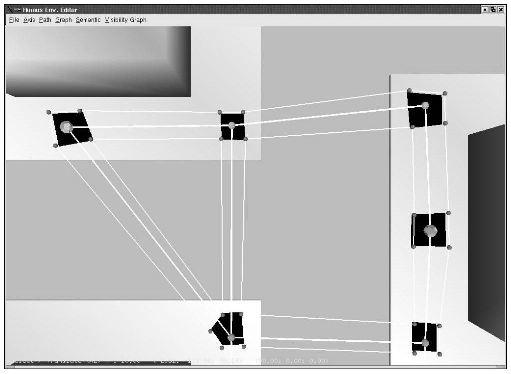


Figure 5: The path are represented by the lines (graph connections), the IPs are represented by the points in the center of the dark regions. The points around the dark regions represent the IP limit region.

Moreover, IPs and Places are semantically different. While one IP is only an intermediate point, for instance a corner, where the agent has to pass in order to cross the street, a Place is an interest point where the agent would like to go, for example the entrance to a restaurant.

4.1.2 Graph Paths

The paths are formed by the connection of two IPs (or places). Once there is a path between two IPs, an agent is able to apply this path. Only existent paths in the database can be understood by the agents. The paths can be automatically generated using visibility graph or by the user.

The visibility graph is generated in three steps: i) the 3D model of the environment is transformed into a 2D plane; ii) the polygon vertices are linked with intersections between them and a new polygon is calculated using a convex hull algorithm [6]; iii) The visibility graph is generated.

In order to generate the visibility graph we used the dijkstra algorithm [7] (Figure 6). The boxes represent obstacles (buildings, for example), and the lines represent the graphs connexion.

4.1.3 Criterias

The criteria are attributes that can be associated to paths, places or IPs. For each of them it is possible to define a different criteria, for instance, specify a sunny path, a crowded place, a dangerous or dark path. Since there are many paths and places in the database, the agent can use the criteria information in order to generate its path planning.

The criteria list can be specified by the user utilizing the EnvEditor. Each criteria is defined by a keyword, an identifier, that can be used by the agents in order to make decisions. For instance in Figure 7 we can see the list of keywords which represent the spaces criteria.

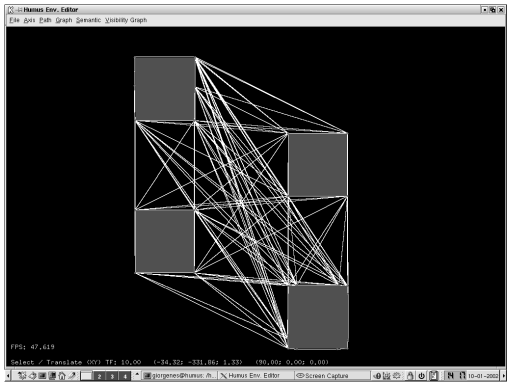
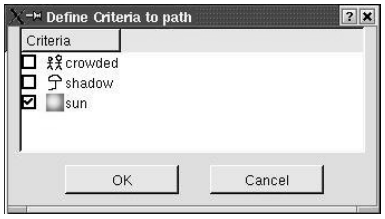


Figure 6: The environment visibility graph.



Define Criteria to path

Criteria

- [ ] crowded
- [ ] shadow
- [✓] sun

OK Cancel

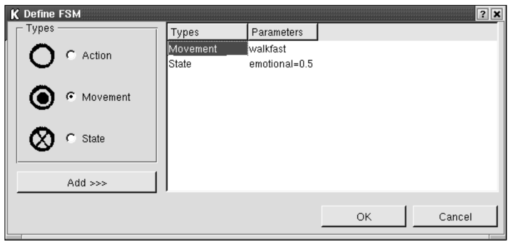
Figure 7: The criteria list of the path.

4.1.4 State Machines

The FSM (Finite State Machines) can be defined, using EnvEditor, in order to specify tasks and states changes which are associated to paths, places or IPs of IVEs. For example, specify an action (play a keyframe animation), change the emotional state of an agent in a certain place, etc.

Moreover, parameters can be associated to the FSM, being informed by the user and interpreted by the agents. Figure 8 shows the FSM specification interface. There are three possibilities of FSM information:

- *Action, the agent applies a specific action, for example clapping (executing a pre-calculated animation);*
- *Movement, the agent applies a specific motion, for instance go to a determined place;*
- *State, the agent changes its emotional state.*



Define FSM

Types: Action [✓ Movement] State

Add >>>

Types	Parameters
Movement	walkfast
State	emotional=0.5

OK Cancel

Figure 8: The interface of FSM.

4.2 AgentSim

This module was developed in order to simulate very simple agents evolving in IVEs. The agents in AgentSim are not able to take decisions, so they behave exactly as informed by the IVE. Our idea is to integrate with very simple as well as sophisticated agents. In the first case the agent just does what the environment says, in the latest, the agent decides which information coming from the environment is going to be used and in which manner.

The integration between the modules (IVE, IC and AgentSim) is made using Python language. In a script the user can call Python functions to manage the agent and ask information from other modules.

EnvLib (library responsible by IVE) offers a set of functions that allow access to the XML database. For example: *GetPathFromPos* those parameters are: current agent position (x,y,z) and PLACE (that means a specific location in the database where the agent should go). This function returns a list of positions to be reached by the agent in order to arrive at the final destination (PLACE).

AgentSim provides creation, manipulation and visualization of virtual agents. For example: *CreateAgent* which parameters are agent identification, initial emotional state and start position.

In a script, we call functions from each module appropriately, loading the generated information by the EnvEditor, choosing the paths, creating the agents, and informing the agents to follow a determined path. The example below shows a simulation script including one agent:

Loading the intelligent environment module
```
import Env
```
Loading the AgentSim module
```
import agent
```
Loading the data base with the virtual environment semantic
```
Env.LoadEnv(”city.xml”)
```
Getting the list of IPS to go to the bank from a specific position (0,0,0)
```
l =Env.GetPathsFromPos(0.0, 0.0,0.0, ”bank”)
```
Getting the paths semantics that are into path ”l”
```
ls=Env.GetFSMFromPath(l)
```
Starting the AgentSim module
```
agent.init()
```
Defining the geometric model of the city
```
agent.setCityModel(”city.obj”)
```
Creating an agent called Bob
```
agent.create(”bob”)
```
Calling agent control
```
agent.perform()
```
Starting the graphical visualization
```
agent.visualize()
```
Sending a task of type ”GO” to the agent called ”Bob”, applying the paths ”l” with the specified semantic defined in ”ls”
```
agent.go(”bob”, l, ls)
```

When the script is loaded by the python interpreter, it starts the simulation, then the agent start to walk to the PLACE. Figure 9 shows an agent walking in the city.

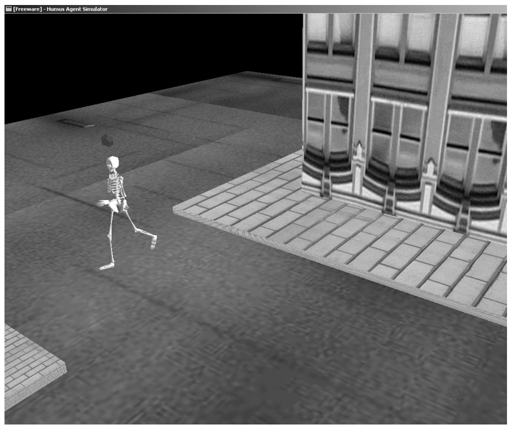


Figure 9: Agent walking in the city.

5 Intelligent Camera

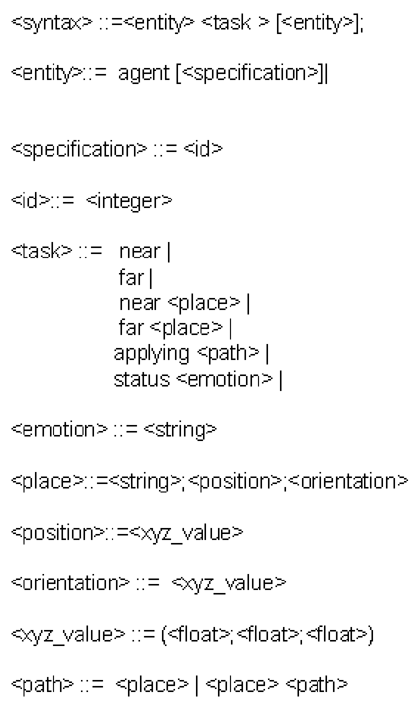
When we deal with complex simulations containing many agents in a large environment, it can be difficult to visualize all the events that were triggered, because more than one event can happen at the same time. The IC has the main function to provide an easy visualization of the events that occur during the simulation. IC automatically adjusts the viewing parameters in order to have proper view of the event.

The IC behaviour is based on events which trigger as a function of IVE and Agents information. A script containing events definition has to be informed to IC Controller in order to deal with any event and react in the appropriated manner.

As can be see in table 2, the entity can be a specified agent (informed identifier) or not. For instance *agent <04>* and *agent* which triggers for any agent in simulation 2. The possible tasks can be NEAR, FAR, STATUS, APPLYING as presented in Figure (10). PLACE and STATUS are related to position in the IVE and emotional state of agents, respectively.

Entity	**Task**	**Entity —Place— State**
agent	near	agent<03>
agent	status	happy
agent<04>	far	agent<01>
agent	near	library
agent	applying	restaurant-path

Table 2: IC Script Example.



```
<syntax> ::=<entity> <task > [<entity>];

<entity>::=  agent [<specification>]|


<specification> ::= <id>

<id>::=  <integer>

<task> ::=   near |
             far |
             near <place> |
             far <place> |
            applying <path> |
            status <emotion> |

<emotion> ::= <string>

<place>::=<string>;<position>;<orientation>

<position>::=<xyz_value>

<orientation> ::=  <xyz_value>

<xyz_value> ::= (<float>;<float>;<float>)

<path> ::=  <place> | <place> <path>
```

Figure 10: The camera script format.

In order to check the triggered events, the IC camera asks information from the IVE and the AgentSim modules. For instance, if the event is: ”Agent Bob NEAR to School”. IC modules needs to know the location of Bob and School in order to decide whether the distance is near or not. Figure 11 shows IC automatically localizing the agent.

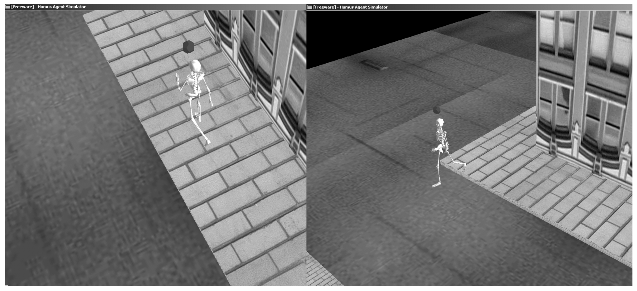


Figure 11: The IC localizing the agent.

In order to provide a proper view of the triggered events, the IC module describes the behavior to be adopted: following the entity or standing still, depending on the agents motion.

6 Results

For this case study, we simulated agents endowed with very little intelligence. In this case, the agents change their emotional states based on the semantic information associated to the environment, for instance one agent can become ”sad” when it is near to a cemetery, or ”happy” when it passes in front of a party.

The example below shows the following situation: we load a 3D city model that contains IPs and PLACE (park) specification. The IPs and PLACES are connected using the visibility graph. Then, AgentSim module runs a Python script in order to start the simulation. The script information is presented as follows (we avoided the comments in some lines because they were explained in Section 5.1):

import Env
import agent
import IC
Env.Init(””)
Env.LoadEnv(”city.xml”)
l =Env.GetPathsFromPos(0.0, 0.0,0.0, ”park”)
ls=Env.GetFSMFromPath(l)
Load the IC script file
IC.File(”script.cam”)
Ask to AgentSim module the agents position
IC.Reg(agent.query)
Ask to Env module the places position
IC.RegEnv(Env.query)
send to Env the camera position
IC.RegCam(agent.lookAt)
agent.init()
agent.setCityModel(”city.obj”)
agent.create(”bob”)
agent.setPos(”bob”, 100.0, 0.0,140.0)
agent.perform()
agent.visualize()
agent.go(”bob”, l, ls)
It starts the camera perform
IC.per()

It is possible to make a simulation with many agents, as they are very unsophisticated ones. We simulated seven agents in a frame rate of 25 FPS, using a desktop Pentium III 800 mhz with 128 MB of memory. In this case, each agent starts in a different place and has different paths, e.g. one is going to the bank, two are going to the restaurant, one is going to the park, two are going to the school and one is going to the administration building. The models used have been designed using 3DS.

Further information and a video (AVI format) of the simulation is available for viewing at
http://inf.unisinos.br/cglab/equipe/flaviapf/index.html

7 Conclusions

This paper discusses the possibility to provide information from VE integrated with a module to help in the production and visualization of virtual human simulations. Four modules were presented in this framework (IC, IVE, AgentSim and Visualizers) and discussed in this paper.

The main goal of this work is to demonstrate that not sophisticated virtual humans can behave in a convincing way only by interacting with IVEs. Figure.12 shows three available configurations of simulation concerning different levels of autonomy (LOA) for virtual humans (VH) as well as the level of informed information (LOII) contained in the VEs. According to Figure 12 (c), the agents should have a high level of autonomy in order to be able to evolve in non-Intelligent VEs (Low LOII). On the other hand, IVEs can be easily integrated with agents endowed with different levels of autonomy (Figure 12 (a) and (b)).

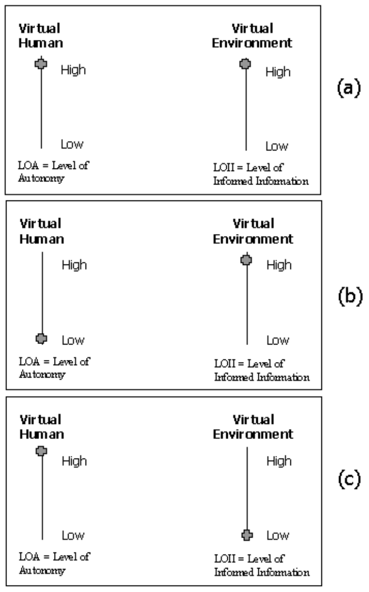


Figure 12: (a) High LOA for VH and High LOII for VE; (b) Low LOA for High LOII; (c) High LOA for Low LOII.

In despite this discussion, it is hard to say the exact LOA for VH or LOII for VE that allow to obtain convincing behavioural simulations. It means which level of LOII and LOA are minimum acceptable or really needed for each type of application. As future work we are interested on integrating this framework with other VH simulators, where the agents can be more sophisticated. Futhermore, we are studying some metrics to be observed in order to evaluate the believability of simulations.

References

[1] K. Ashida,et al. "Pedestrians: Creating Agent Behaviors through Statistical Analysis of Observation Data. *Computer Animation 2001*, National Seoul University, (Nov.7-8, 2001).

[2] L. M. Barros, T. F. Evers, S. R. Musse. A Framework to Investigate Behavioural Models. *Journal of WSCG*. 10,(1), 40-47. (Plzen, Feb 2002).

[3] P. Becheiraz. "Un Modèle Comportemental et É motionnel pour l'Animation d'Acteurs Virtuels." *PhD Thesis*, EPFL. Lausanne, (Switzerland, 1998).

[4] C. Bordeux, et al. "An Efficient and Flexible Perception Pipeline for Autonomous Agents." *Proceedings of EUROGRAPHICS '99*. 18, (3),23–29, (1999).

[5] A. Chaves, et al. "Challenger: A Multi-agent System for Distributed Resource Allocation." *Proceedings of Autonomous Agents'97*. ACM Press, 323–332, (Marina Del Rey, CA USA, Feb 1997).

[6] K. Clarkson. *http://www.netlib.org/voronoi/hull.html* (2002).

[7] E. Dijkstra. "A note on two problems in connexion with graphs." *Numerische Mathematik* 1, 269-271, (1959).

[8] S. Donikian, E. Rutten. "Reactivity, Concurrency, Data-flow and Hierarchical Pre-emption for Behavioural Animation." *Paradigms in Graphics'95*, Eurographics collection. (Springer-Verlag, 1995).

[9] S. Donikian. ''VUEMS: A virtual urban environment modeling system." *Computer Animation '97*. 127-133, (1997).

[10] S. Donikian. "Towards Scenario Authoring for Semi-Autonomous entities." *International Conference on Visual Computing (ICVC99)*. (Goa, India, Feb 1999).

[11] S. Drucker, D. Zeltzer. ''Intelligent Camera Control in a Virtual Environment." *Graphics Interface '94*. 190–199 Canadian Human-Computer Communications Society. (May 1994).

[12] N. Farenc, R. Boulic, D. Thalmann. "An informed environment dedicated to the simulation of virtual humans in urban context." *Proc. Eurographics '99, Computer Graphics Forum*, 18,(3),C–309–C–317, (1999).

[13] N. Farenc, et al. "A paradigm for controlling virtual humans in urban environment simulation." *Journal Applied Artificial Intelligence*, 14,(1),69–91, (2000).

[14] M. Girard, S. Amkraut. "Eurhythmy: Concept and Process." *The journal of Visualization and computer animation*, 1, 15–17, (1990).

[15] A. Hyvärinen, T. Honkela. "Emotional Disorders in Autonomous Agents." *Proceedings of ECAL99*. (Springer, 1999).

[16] M. Kallmann. “Object Interection in Real-Time Virtual Environments.” DSC Thesis” *DSC Thesis* , (2001).

[17] D. Luigi,V. Maniezzo. “The Rise of Interaction: Intrinsic Simulation Modelling of the Onset of Interacting Bahavior” *Proceedings of First International Conference on Simulation of Adaptative Behavior*. MTI Press, (1990).

[18] Marr, Vision, Freeman. (New York, 1982).

[19] E. Marchand, N. Courty. “Controlling a camera in a virtual environment.” *The visual Computer Journal*, 18, (1), (February, 2002).

[20] J. A. Meyer, A. Guillot. “From SAB90 to SAB94: Four Years of Animat Research.” *Proceedings of Third International Conference on Simulation of Adaptive Behaviour*, (Brighton, England, 1994).

[21] S. Musse, M. Kallmann, D. Thalmann. “Level of Autonomy for Virtual Human Agents.” *Proceedings of ECAL ’99*. (Springer, 1999).

[22] S. Musse. ”Hierarchical Model for Real Time Simulation of Virtual Human Crowds.“ *IEEE Visualization and Computer Graphics* 7, 152, (2001).

[23] H. Noser. “A Behavioral Animation System Based on L-Systems and Syntetic Sensors for actors.” *PhD Thesis*, EPFL. (Lausanne, Switzerland, 1996).

[24] POV-Ray. http://www.povray.org. (2002).

[25] POV-Ray. http://www.python.org. (2002).

[26] O. Renault, et al. “A Vision-based Approach to Behavioral Animation.” *The journal of Visualization and computer animation*, 1, (1),18–21, (1990).

[27] W. D. smart, J. Hallam “Location Recognition in Rats and Robots.” *Proceedings of Third International Conference on Simulation of Adaptive Behaviour*. (Brighton, England, 1994).

[28] A. R. Smith “Plants, Fractals, and Formal Languages.” *Proceedings of First International Conference on Simulation of Adaptive Behaviour*, MTI Press. (1990).

[29] Softimage Co. http://www.softimage.com. (2002).

[30] R. S. Sutton. “Reinforcement Learning Architectures for Animats.” *Computer Graphics*, 18, (3),1–10, (July 1984).

[31] G. Thomas, S. Donikian. “Modelling virtual cities dedicated to behavioural animation. *Eurographics 2000*, (Interlaken, Switzerland, 2000).

[32] G. Thomas, S. Donikian. “Virtual humans animation in informed urban environments *Computer Animation 2000*, IEEE, (Philadelphie, USA, May 2000).

[33] B. Ulicny, D. Thalmann. “Crowd simulation for interactive virtual environments and VR training systems *Proc. Eurographics Workshop on Animation and Simulation’01*, 163–170, (Springer-Verlag, 2001).

[34] M. Wooldrigde, N. Jennings. “Intelligent Agents: Theory and Practice.” *Knowledge Engeneering Review*, 10, (2) Cambrigde University Press, (June 1995).

8 Annex

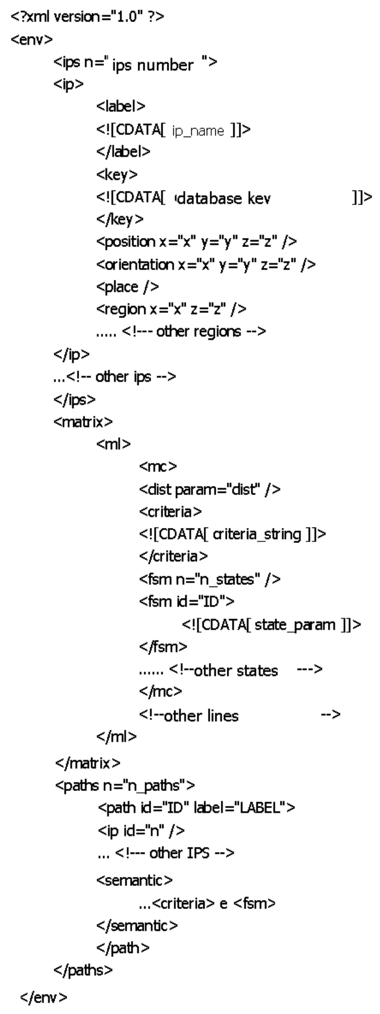

```
<?xml version="1.0" ?>
<env>
        <ips n=" ips number ">
        <ip>
                <label>
                <![CDATA[ ip_name ]]>
                </label>
                <key>
                <![CDATA[ database kev ]]>
                </key>
                <position x="x" y="y" z="z" />
                <orientation x="x" y="y" z="z" />
                <place />
                <region x="x" z="z" />
                ..... <!--- other regions -->
        </ip>
        ...<!-- other ips -->
        </ips>
        <matrix>
                <ml>
                        <mc>
                        <dist param="dist" />
                        <criteria>
                        <![CDATA[ criteria_string ]]>
                        </criteria>
                        <fsm n="n_states" />
                        <fsm id="ID">
                                <![CDATA[ state_param ]]>
                        </fsm>
                        ...... <!--other states    --->
                        </mc>
                        <!--other lines                -->
                </ml>
        </matrix>
        <paths n="n_paths">
                <path id="ID" label="LABEL">
                <ip id="n" />
                ... <!--- other IPS -->
                <semantic>
                        ...<criteria> e <fsm>
                </semantic>
                </path>
        </paths>
    </env>
```

Figure 13: XML Data Base