Guiding and Interacting with Virtual Crowds in Real-Time

Soraia Raupp Musse, Fabien Garat and Daniel Thalmann

Computer Graphics Lab. Swiss Federal Institute of Technology
EPFL, DI-LIG, CH 1015 Lausanne, Switzerland
{soraia,garat,thalmann}@ lig.di.epfl.ch
http://ligwww.epfl.ch

Abstract. This paper presents some aspects to provide interaction with virtual human crowds. We describe some interaction paradigms present in ViCrowd, a system to model and generate virtual crowds with various degrees of autonomy. In addition, a Client/Server architecture is discussed in order to provide interface to guide and communicate with virtual crowds.

1 Introduction

Virtual humans grouped together to form crowds populating virtual worlds allow a more intuitive feeling of presence. Yet, different applications can have different requirements in terms of crowd control. For instance, one can simulate specialised or "intelligent" behaviours of crowds; another application can be interested in modelling the interaction with virtual crowds as well as some basic behaviour. In each one of these applications, the control and autonomous nature of the crowd and the virtual agents can be more or less sophisticated. We have developed the ViCrowd model [MUS97][MUS98] to simulate crowds with different types of control: *programmed* (pre-defined behaviours using a script language), *autonomous* (rule-based behaviours) and *guided* (interactive control) depending on the goal of the simulation. Table 1 presents some characteristics of crowd control types.

BEHAVIOUR CONTROL	**GUIDED CROWDS**	PROGRAMMED CROWDS	AUTONOMOUS CROWDS
Level of Autonomy	**Low**	Medium	High
Level of "Intelligence"	**Low**	Medium	High
Execution frame-rate	**Low**	Medium	High
Complexity of behaviours	**Low**	Variable	High
Level of Interaction	**High**	Variable	Variable

Table 1. Characteristics of different types of crowd control.

Although the different types of control existing in our model, this paper is more focused on different aspects of interaction with guided crowds. We present, then, the various interaction paradigms that were modelled in order to define the different manners to interact with guided crowds. Yet, a Client/Server system is proposed in order to define an architecture to provide the external control of crowds and communication with other modules.

First of all, we present some useful concepts assumed in this work. A *virtual human agent* (after referred as an agent) is a humanoid whose behaviours are inspired by those of humans' [MEY94]. They can be equipped with sensors, memory, perception, and behavioural motor that allow them to act or react to events. They can also be much simpler, like guided by users in real time or interpreting predefined commands. The term *group* will be used to refer to a group of *agents* whereas a *crowd* concerns a set of *groups*. The *interaction paradigms* define various types of messages to be exchanged between the external processes and the crowd in order to establish the interaction. Concerning the protocol of Client/Server architecture, *stimulus* describe the messages that each client can send to the server; and *response* concerns a list of sub-messages or sub-stimulus to be performed in order to fit the stimulus.

Although several works aiming to improve the autonomy included in virtual agents, there are some recent efforts to integrate autonomy and directability. Zeltzer [ZEL91], Blumberg [BLU95], Perlin [PER96] and Thalmann [THAL96] have presented different classifications and levels of information in order to describe applications managing control of avatars, guided agents, programmed agents, rule-based behaviours and etc.

To model autonomous crowds, there are several approaches such as particle, flocking and behavioural systems. These techniques are characterised by the possible number of individuals to be simulated, their intelligence levels and decision ability, the associated collision avoidance method, the employed control method, etc. Several authors have worked to model many autonomous agents controlled by physical rules using particle systems [BOU97][BRO98]. Behavioural systems consider the autonomous agents as "intelligent" agents that can make decisions using specific rules [LUI90][REY87][NOS96]. We described ViCrowd as a model to manage virtual human crowds with various degrees of autonomy. Yet, several methods concerning distribution of crowd behaviours among individuals as well as the sociological model used for modelling crowds have been presented in recent works [MUS97][MUS98][SCH99]. The Client/Server

architecture is not a novel idea. Several works have used it in order to describe distributed or multi-client systems [COH94][GUZ96][CHA97].

The next section presents some aspects about ViCrowd and Section 3 describes the interaction paradigms, which provide the exchange of information between virtual crowds and real participants. Section 4 addresses the multi-client architecture we propose to guide crowds. The discussion closes with an experiment study, as well as with possible future lines of work.

2 Crowd Model

This section aims at presenting some concepts of our crowd model [MUS97] [MUS98] and more explicitly about the guided crowd. The simulation of human crowds for populating virtual worlds provides a more realistic sense of virtual group presence. In some virtual environments, it would be useful to simulate populations in an autonomous way, thus the agents have a kind of environment knowledge and are able to move and interact within this environment. However, depending on the application, more ways of interaction can be required in order to provide a real time communication between participants and virtual agents. We have worked with three levels of autonomy: guided, programmed and autonomous (see Table 1) in order to establish the required control of crowds, depending on the application. These three levels of autonomy are represented using two kinds of interface: scripted or guided interface. *Scripted interface* uses a script language where action, motion and behavioural rules are defined in order to specify the crowd behaviours. While action and motion describe explicit behaviours of crowd, called ***programmed crowd*** (see Fig. 2), the behavioural rules are used to define ***autonomous crowd***. All these information can also be sent by an external process in order to guide crowds explicitly, during the simulation. We called this type of crowd as ***guided crowd*** (see Fig.1). The small window on bottom right represents the Textual User Interface (TUI) client where textual commands can be specified. Figure 1 shows guided agents reacting according to a textual command: GOTO Station.

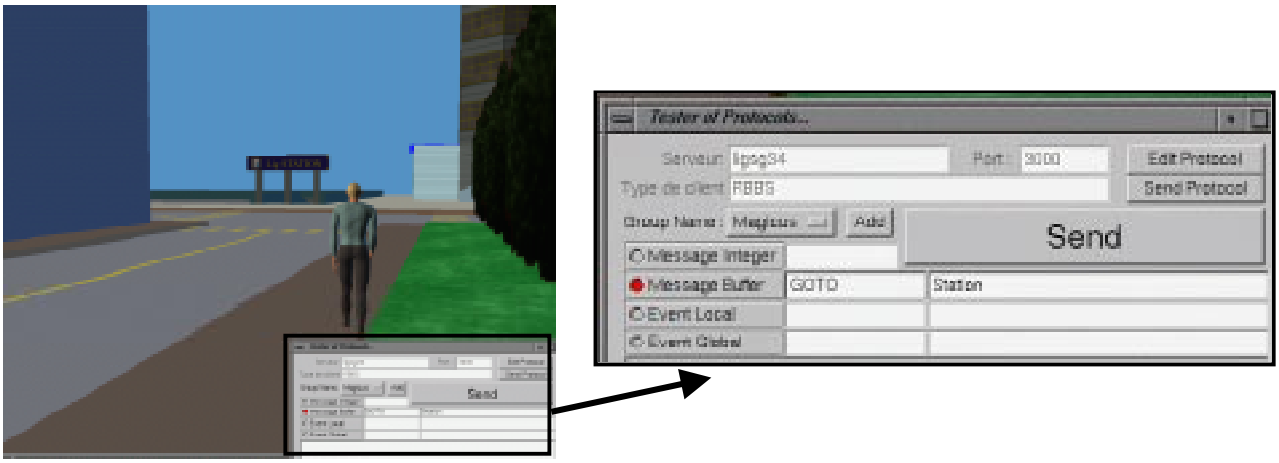

Figure 1: A guided agent going to the train station as specified in the Textual User Interface Client (TUI).

The ***autonomous crowd*** (see Fig. 3) is able to act according to the inherent behaviour (seeking goal, walking, avoiding collision, etc), recognising the presence of behavioural rules and also obeying programmed behaviours of groups in the script language. In order to mix different behaviour natures, we have defined a priority of behaviours, then synchronising different kinds of control: (from high to low priority) guided, autonomous, programmed and inherent. Some examples of mixed control are:

- A group of agents walk on the virtual city (programmed behaviour) avoiding collision with other agents and declared obstacles (inherent behaviour).
- A panic event that was time-defined in the script language occurs (e.g., pre-specified to occurs in frame 1000). The reactions specified in the behavioural rules are activated (the reactive behaviour has more priority than the programmed motions and actions). Then, agents loose their programmed behaviour and employ their programmed reaction (e.g., look for the exit doors in the virtual environment).
- The user/external controller gives a new order to the agents, which form the crowd. For instance, exit in a specific door (external control). Afterwards, the agents stop to react as a function of pre-programmed events and follow the external control.

More details about our crowd model have been published by Musse and Thalmann [MUS97][MUS98]. Although the multi-autonomy level of our crowd approach, this paper is more focused on discussion about the ways of interacting with guided crowds.

Figure 2,3: (left) Programmed crowd walking on the city [FAR98]. (right) Autonomous crowd reacting as a function of a rule based system.

2.2 Guided Crowds

The guided crowd represents groups of virtual agents, which can be externally guided. As the crowd in our approach is goal-based (the groups of agents always try to seek goals), the guided crowd receives dynamic goals in order to reach during the simulation. The goals concerning the information entities that can be controlled, they are:

- Motion (go to a specific position respecting collision avoidance with obstacles)
- Action (apply a specific posture, interact with an object)
- Events (can be triggered as a function of a matched condition or time-based generated)
- Reaction (can be: an action; a motion; attach to the motion of another object; change the internal status of a group/agent; activate one or more events)
- Internal status of groups (e.g., the group emotion)
- Environment information (used to declare obstacles that have to be avoided and regions where the crowd can walk)

Some of these entities information (specified during simulation by an external controller) are independent of script language information (defined before simulation starts). However, some of them need to be referenced or declared before simulation, because it can also affect autonomous crowds. For example, the events and reactions: an event can be a pre-determined fact (time-based defined in the script, e.g., frame 1000), or still can be activated through an external controller. In both cases, this event has to be declared and its parameters defined in the script as well as the associated reactions. Table 2 describes the inter-dependence existing between these entities:

GUIDED ENTITIES	SCRIPT DEPENDENCE	GUIDED CONTROL
Actions	Not existing	Can just be applied by the guided crowd
Motion	Not existing	Can just be applied by the guided crowd
Events	Have to be declared in the script.	Can be activated by the external control. Can influence the autonomous crowd
Reactions	Have to be declared in the script.	Some parameters of reactions can be sent via external controller. Can influence the autonomous crowd
Status	A status changing can match other events	Can influence autonomous crowd
Environment Information	The guided crowd considers the parameters declared in the script.	The autonomous crowd considers the parameters declared in the script

Table 2: Inter-dependence between guided entities and scripted information.

Figure 5 shows some images of a crowd evacuation from a museum during a panic situation caused by a statue that becomes alive. Indeed, the statue is externally controlled, and the crowd initially obey the programmed behaviour: walk and visit the museum. Afterwards, when the external controller applies a motion to the statue, the crowd reacts according to the pre-programmed reaction in the script, so, exiting the museum through the two doors.

Figure 5: Scenes of simulation of evacuation due to a panic situation. Up left and right: Before the event, the crowd walks. Down, left and right: crowd reacts. The statue motion and action are externally controlled.

3 Interaction Paradigms

The interaction paradigms set different ways to interact or guide crowds. The exchanged information is basically classified in 8 types:

1. Selection: Selection of a group/agent to interact.
2. Motion: Defines new motion paradigm to the selected entity.
3. Action: Selection of an action to be applied.
4. State: Changes the internal status of group or agents.
5. Density: Increases or decreases the number of agents in the output area (camera's view).
6. Events/reactions: activates pre-programmed events and change event/reaction parameters.
7. Request: Requires about the selected entity.
8. Knowledge: Defines environmental data.

Each one of the paradigms presents different information that can be dealt in order to interact with a group or agent of the crowd. The next sections show more information about each interaction paradigm.

3.1 Selection paradigm

To manipulate with the groups/agents of crowd, it is necessary to select which group should be dealt with. This paradigm includes functions to select agents or groups. Afterwards, the selected entity might be manipulated with the others paradigms. The way to get information about the entities depending on the interface developed. For instance, a graphic interface can allow graphic functions like pick an agent, in order to select a group near to an object, or more placed in the right side of the output field of view. Considering this paradigm, we can also select agents depending on some conditions, for example, agents near to a specific object (location function), or agents which emotional status is HAPPY (emotional status function).

Selection paradigm:	Conditional functions to select entities:
Select an AGENT	~ Group/Agent status
Select a GROUP	~ Group/Agent location
	~ Group/Agent goal

3.2 Motion Paradigm

This paradigm defines a specific motion to be applied by the selected entity. Yet, the information defined in the script to inform autonomous crowds is also regarded by guided crowds in order to avoid collision with declared objects. Yet, locations and dimension of obstacles can be dynamically defined in the environment paradigm (see section 3.8). If there is no selected entity, this paradigm is applied to everybody.

Motion paradigm:
~ Go to a specific location

3.3 Action paradigm

This paradigm sends actions to be applied by the crowd. These actions can be a body posture or interactions with objects [KAL98]. Also, the action can be applied exactly at the moment sent by the external controller or be synchronised with the motion paradigm. For instance, send a motion task in order to reach a counter and after an action paradigm in order to buy a ticket. This paradigm is applied to the selected entity (agent or group), if there is not a selected entity, so, the action is applied to every agent of crowd.

Action paradigm:
~ Apply a body posture
~ Interact with an object

3.4 State Paradigm

This paradigm is responsible for setting parameters to change the internal state of agents and/or groups. Various parameters can be manipulated within this paradigm in order to set or change entity's state.

State paradigm (divided into three types of information)
~ Behavior data: group behaviors (flocking, following, adaptability, collision avoidance, repulsion, attraction and split) [MUS98]
~ Quantitative data: number of agents and list of agents.
~ Internal status data: emotional status, individual level of domination, way of walk, relationship with other groups/agents, etc. [MUS97]

3.5 Density Paradigm

During the simulation, it is possible to know the location of camera's field vision in order to control which agents and groups could be seen by the application and apply the selection paradigm. So, the density paradigm aims at changing the number of agents located in the output field.

Density paradigm:
~ Increase or decrease.

3.6 Events/Reactions Paradigm

This paradigm is responsible for the activation of crowd events and reactions. Events and reactions in our model consist of behavioural rules to be applied depending on the matched conditions. For instance, an event can be activated as a function of a request paradigm (Section 3.7), status paradigm (section 3.4), or an autonomous behaviour specified in the script language. Events and reactions have to be declared in the script language (see Table 2), anyway some data can be generated in real time during the simulation as well as the activation of events. Normally, we have modelled events which are activated during running time through the external interface. For instance, a panic situation (Fig. 5) is declared in the script language as well as the consequent reactions. Anyway, the information about the fine moment when the panic situation occurs can be specified in real time using the event/reaction paradigm.

Events/reactions paradigm:
~ Activate or desactivate
~ Send specific data

3.7 Request Paradigm

Some information is needed to be accessible by the user in order to achieve the interaction paradigms with crowds in ViCrowd. Some examples of information of crowd can be: position, orientation (for crowd, group or agent), nature of group, emotion of group/agent, if group/agent is reacting to some matched event, goal of group, the current group behaviours, etc.

3.8 Environment Paradigm

Information about regions where we can walk, obstacles to be avoided and etc, can be defined in the script language before starting the simulation. The guided interface can define complementary information as well as re-define already described information.

4.Architecture to Guide Crowds

Considering the many possibilities of interaction with ViCrowd, we decided to propose a multi-client architecture in order to provide several manners to communicate with the virtual crowd.

First of all, the protocol we used to provide the communication between the clients and the server in the context of real time animation, is described. We have defined a server as an application responsible for sharing and distributing messages among the clients. One client is a specific application that can have more than one connection with the server. For instance, in Fig. 6, ViCrowd client has 3 connections (Group1, Group2, Group3) sharing the same protocol in order to communicate with a specific client (*RBBS client – Rule-based behaviour system*). An advantage of a Client/Server architecture is that we can distribute independent process on several computers as well as use other processes as "black boxes". As example, we can separate the knowledge about the virtual environment where the simulation occurs from the rendering client. These processes can run on different computers, and share information via the server. Each client does not have information about other clients except their input and output, so, they can be considered as black box.

As we have oriented the server for managing crowds, we assumed that each client requires a connection for each guided group to be manipulated. At the same time, each client sends information to the server in order to present the input and output that can be understood. This information is described using the protocol to inform the server about the instructions to follow (*responses for each stimulus*) when clients send messages (*stimulus*). Afterwards, the server is capable to understand from which client the message came, to which client it has to be redirected and which responses have to be employed in order to fit the stimulus.

Figure 6 shows an example of architecture formed by 6 different types of clients. Each client can have several connections (e.g. ViCrowd client has 6 connections) that are able to send and receive messages from the server in order to drive the guided groups. There are two types of connection: *request* and *group_name* connections. The first one represents connections accessible by all the connected clients. The *group_name* connection deals with information for specific groups, e.g., group 1 in ViCrowd Client.

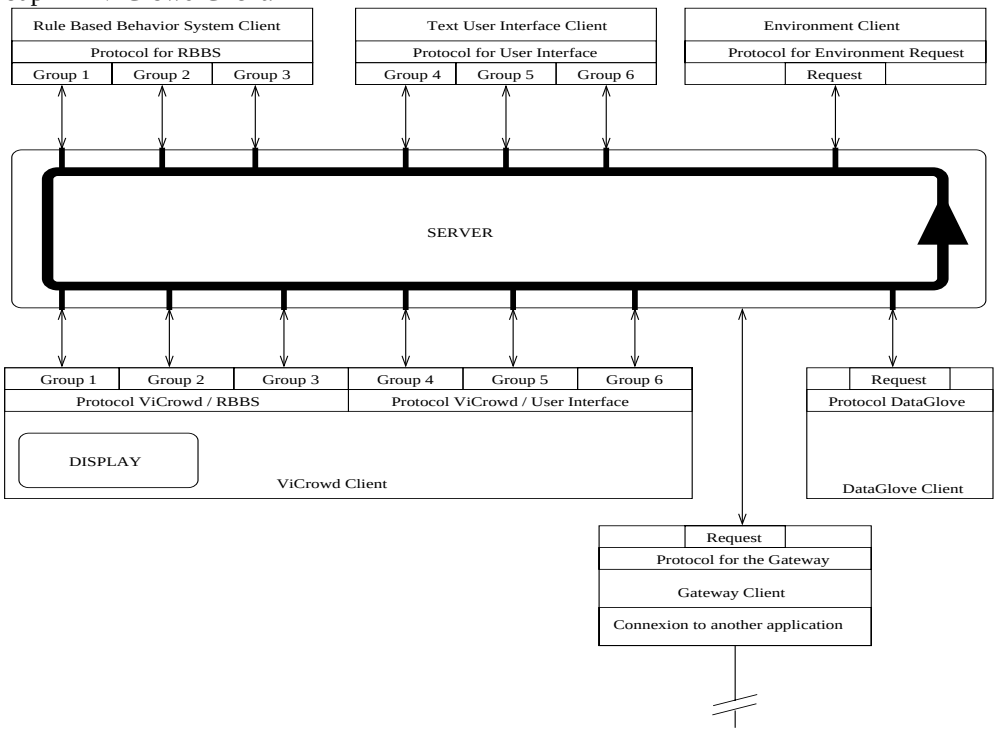

Figure 6: Various clients and connections within the architecture.

4.1 Protocol information

As mentioned before, some information is sent to the server at the beginning of the simulation, in order to present each client. These information are classified in two different parts: *stimuli* and *responses*. Afterwards, using this definitions, the server is able to recognise the client from which the message is coming and what have to be performed for each message: redirect to other clients, translate some data, etc.

The parameters of the protocol are sent by each client in order to describe the following items:

1. Who is the client, describing the *type of client name* and the *group name* which has to be affected by the interaction (e.g., ViCrowd Client, group_1). In the case of generic connections (information to be accessed by all other clients), a special *group name* is sent in order to be identified as a *request* connection.
2. Which *stimuli* can be received from this client;

3. Which *responses* have to be applied by server or clients, in response of a specified stimulus. This protocol can be changed and re-loaded at any moment during the simulation, allowing a re-management of complex clients in real time. One client can deal with more than one pre-defined stimulus. For each stimulus, the server receives associated instructions to be applied (*responses*).

4.1.1 Examples of stimulus/responses using the interaction paradigms

Example 1 shows the manipulation of a variable representing the emotional status of a specific group, using the *state interaction paradigm* (see Section 3.4).

Client Name: Textual User Interface	*(identifier of client)*
Group Name: Group1	*(identifier of group name)*
Stimulus: BE > $Emotion	*(Format of Stimulus sent by Client)*
Response: ViCrowd APPLIES STATE BE <$Emotion	*(ViCrowd changes emotional status of Group1)*
EndResponse:	(End of response associated to Stimulus)

For instance, when the Textual User Interface (TUI) client send stimulus: "BE HAPPY", “HAPPY” is stored in the variable $Emotion and afterwards sent to ViCrowd Client that changes the internal status of Group1.
The next example uses two sorts of interaction paradigms: *motion and request*.

Stimulus: GOTO SPECIFIC_LOCATION	*(Format of Stimulus sent by Client)*
Response: ViCrowd APPLIES **MOTION**_GOTO <$NAME AT 63150 -5000 -12540 1 0 0	
(ViCrowd activates Group 1 to walk until the specified position using Motion paradigm)	
Response: ViCrowd WAIT_FOR (**REQUEST**_GOALREACHED FOR <$NAME) TRUE	
(Request paradigm is used in order to verify the group location)	
EndResponse.	*(End of response)*

In this case, when TUI client sends “GOTO SPECIFIC_LOCATION” using the connection with group "Group1", the server sends to ViCrowd client: "**MOTION**_GOTO Group1 AT 63150 -5000 -12540 1 0 0 ", which uses a motion interaction paradigm using parameters to define position (63150 -5000 –12540) and orientation (1 0 0). The second response waits until ViCrowd client update the field "**REQUEST**_GOALREACHED FOR Group1". It only occurs when the group has finished the previous response.

4.2 Interface with Crowds

We have proposed to use different clients in order to interact with crowds. Using the protocol defined in last section, all clients can present the parameters they can deal with in order to communicate with others clients and send information to guided crowd. There are three types of clients in our proposed architecture.

1. Behavioural Clients concern the clients responsible by the management of behaviours [SCH99],
2. Interface Clients provide different interfaces for interaction with crowds, and
3. Database Client is responsible by providing information from the virtual environment, e.g., regions where the agents can walk, the objects to be avoided, etc [FAR99].

TYPE OF CLIENT	NAME OF CLIENT	CLIENT FUNCTIONS
Behavioural Clients	ViCrowd Client	Manages the crowd and display the animation.
	RBBS Client	Manages the Behaviours of guided groups, using LISP rules
Database Client	Environment Client (ENVIR)	Database about virtual environment and computation of trajectories
Interfaces Clients	Text User Interface (TUI)	Allows user to send stimulus to guided crowd using a textual interface
	Dataglove Client	Recognises hands gesture using a dataglove in order to start events or to give additional data to the multimodal client
	Speech Recognition Client	Recognises sentences and associate them with specific events, and give additional data to the multimodal client
	Sound Client	Generate a sound in association with the others clients

Table 3: Types of clients in Client/Server architecture

Each one of the clients is responsible to send some information to others clients. Basically, there are five kinds of data to send to others clients:

1. Confirmation data (ACK), *e.g., goal reached.*
2. High-level behaviour (HLB), *e.g., Group5 goes to the restaurant to lunch*
3. Low-level behaviour (LLB), *e.g., Group5 GOTO position (X Y Z)*
4. Asking data (ASK), *e.g., ASKPOS (asking for position) “Restaurant*”
5. Occurrence of crowd events (EVE), *e.g., Event PANIC occurred*

Table 4 shows the data each client can send and receive from others clients. The data included in the table is related to the five data types recently mentioned.

SENDS DATA TO COLU	ViCrowd	RBBS	ENVIR	TUI	Dataglove	Speech	Sound
ViCrowd		ACK/EVE	ASK	EVE	EVE	EVE	EVE
RBBS	HLB/EVE			EVE	EVE	EVE	EVE
ENVIR	LLB						
TUI	HLB/LLB/EVE	EVE			EVE	EVE	EVE
Dataglove	HLB/EVE	EVE		EVE		EVE	EVE
Speech	HLB/EVE	EVE		EVE	EVE		EVE
Sound	HLB/EVE	EVE		EVE	EVE	EVE	

Table 4: Inter-dependence between clients. Client from row can send data to column clients.

Several clients can generate events. Depending on the protocol specification, the events are transferred to others clients or not. Table 4 shows the basic configuration we have used in our simulations where all the clients whose are able to generate events, should communicate it to others. For instance, since ViCrowd is able to generate events, for all occurrences of ViCrowd events, RBBS and all Interface Clients (TUI, Dataglove, Speech and Sound) will receive this information. Also, the only client able to receive and treat HLB is ViCrowd, while ENVIR is just responsible for giving LLB, since it concerns a database management.

All the events (EVE) generated by the clients in this context, refer to crowd events and have to be declared at the beginning of the simulation in the script language for controlling of programmed, autonomous and guided crowds. Then, ViCrowd is able to react to all the possible events generated by other clients. An example of event activated by an external client is:

```
Event: PANIC
        WHEN external_info
        WHO ALL AGENTS
Reaction: PANIC
        MOTION: 2 IP DOOR1 DOOR2
```

The event PANIC in the example can only be activated using an external controller. When it occurs, all agents (as specified in the command WHO) will be affected by the event, applying then the programmed reaction. The reaction consists to apply a motion towards two interest points (IP): DOOR1 and DOOR2. The geometric information of IPs have to be defined in the script language too.

Considering that more than one client can send different stimulus for the same guided group (e.g., Dataglove sends PANIC event and Speech sends new HLB to guided Group 5), a synchronisation method is required in order to establish some priority rules. Next section discusses this problem and points the employed solution.

4.3 Clients Synchronisation

Since different clients can send information to same guided group at same time, we have established a simple way to define priorities in order to synchronise stimuli. When a client registers stimuli to the server, it can also define the way each stimulus has to behave. For instance, if a stimulus can be interrupted by another one or not.

There are three levels of priority that the stimuli can fit: *high-level*, for non-interruptible stimuli, *mid-level*, for continuos and interruptible stimuli and *low-level*, for non-continuous and interruptible stimuli. Yet, continuous in this context means that the stimuli will be executed later anyway (pushed in a queue) if others more priority stimuli occur. The command <SET PRIORITY> of the protocol defines the level of priority for each stimulus, e.g. SET PRIORITY HIGH-LEVEL. The default information is LOW-LEVEL for the cases where the type of priority is not defined.

For instance, in Figure 7 a system configuration is presented where four different clients present the level of priority of their stimuli. Each client is responsible for a different interface to guide virtual crowds, as specified in Table 3. The flow of execution, the list of stimuli in the queue to be treated by the server and the performed tasks executed by ViCrowd are presented in Table 5.

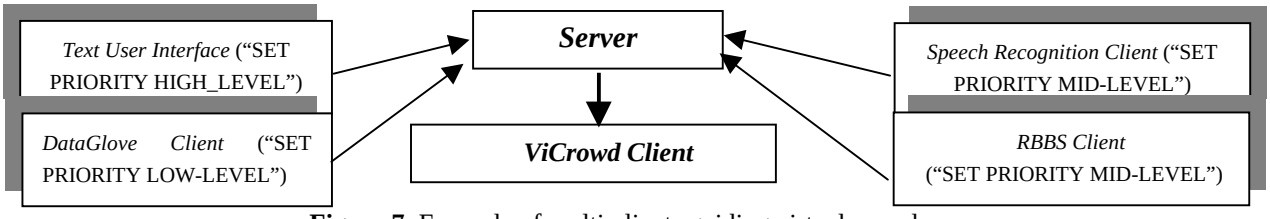

Figure 7: Example of multi-clients guiding virtual crowds

TIME	CLIENT	SET	ORDER SENT	QUEUE IN THE SERVEI	ViCrowd CLIENT ACTION
0	Speech Recognition	MID-LEVEL	GOTO Station	1:GOTO Station	The group goes to train station.
1	Dataglove	LOW-LEVEL	Y KEYFRAME wel	1:PLAY welcome 2: GOTO Station	The group stops to go to the station and plays the welcome keyframe
2	RBBS	MID-LEVEL	GOTO Restaurant	1:GOTO Restaurant 2: GOTO Station	The group stops to play keyframe, and starts to go to the restaurant
3	TUI.	HIGH-LEVEL	GOTO Office	1:GOTO Office 2:GOTO Restaurant 3 GOTO Station	The group stops to go to the restaurant, and starts to go to the office
4	RBBS	MID-LEVEL	GOTO Supermarke	1:GOTO Office 2: GOTO Supermarket 3:GOTO Restaurant 4GOTO Station	The group continues to go to the office, and will go to the Supermarket once reached this position.

Table 5: Flow of execution of synchronisation example

ViCrowd performs all the stimuli pushed inside the server queue. In frame 4, the order sent by RBBS client is on second position of queue because it has more priority that others mid-level priority stimuli already stored in the queue due to time occurrence. For instance, if three sequential mid-level stimuli occur, the last one has more priority.

5 Case-Study Integrating Different Clients in Order to Guide Crowds

The goal of this section is to provide a complete example for interacting with crowds using the Client/Server architecture. Four clients were used in this configuration: ViCrowd, ENVIR, RBBS and TUI. Figure 8 shows the data flow of information sent by different clients. The beginning of simulation concerns an order sent by RBBS client: *Group1 GOTO Counter*.

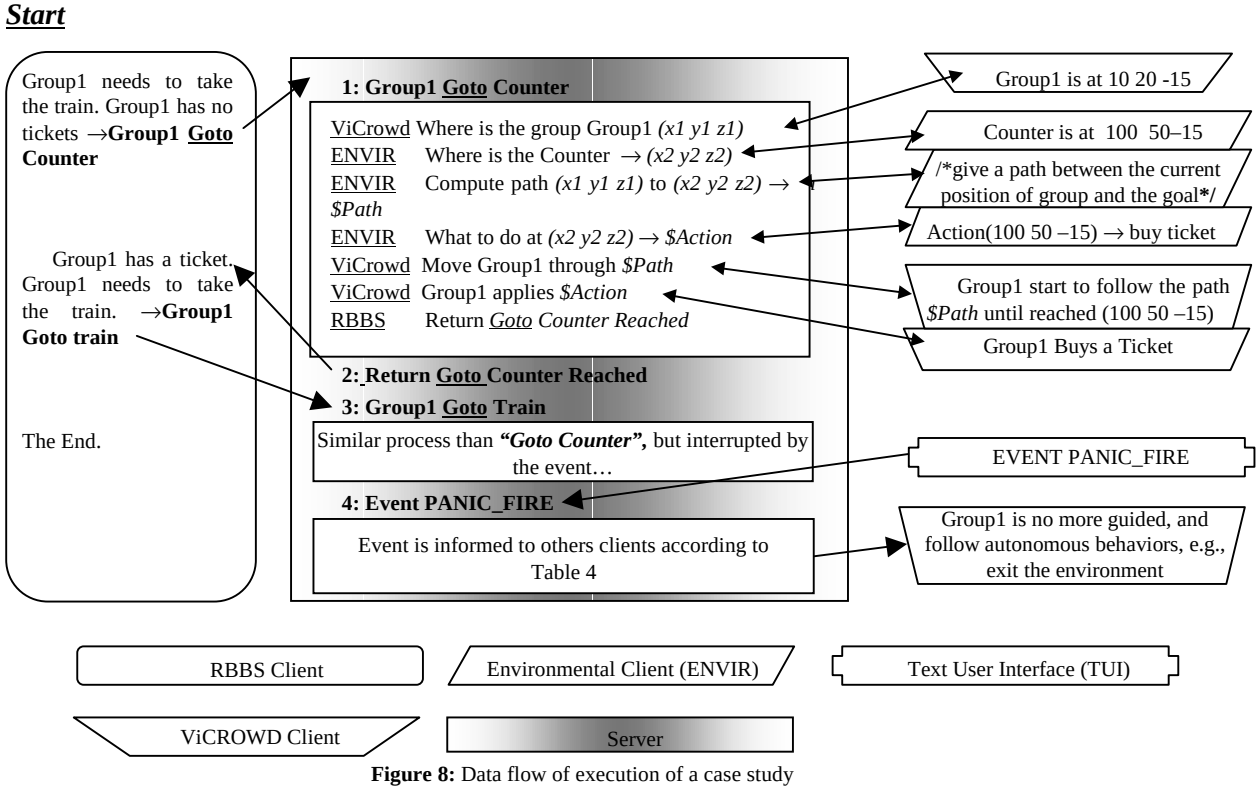

Figure 8: Data flow of execution of a case study

The main stimuli manipulated by the server are: **1: Group1 Goto Counter**, sent by RBBS to Server, **2: Return Goto Counter reached**, send by Server to RBBS, **3: Group1 Goto Train**, sent by RBBS to Server and finally, **4: Event PANIC_FIRE**, sent by TUI to Server. Each one of these stimuli can generate messages to be exchanged with other clients (represented inside the white boxes). For instance, when RBBS sends: **Group1 Goto Counter**, some messages are generated by the Server as part of the responses programmed in the protocol to apply the specific stimulus.

When the stimulus: (**4: Event PANIC_FIRE)** occurs, the command (**3: Group1 Goto Train**) is interrupted according to the priorities rules presented in Section 4.3. Also, in the crowd point of view, events externally defined have higher

priority than guided or autonomous behaviours. Afterwards, the crowd reactions are applied. These reactions are defined in the script language and can be accessible by all different types of crowd: programmed, autonomous or guided. In addition, the reaction in ViCrowd can be: an action; a motion; an attachment to the motion of another object; changing the internal status of a group/agent; activate one or more events, etc. In the case of this example, the reaction concerns a motion of all agents of crowd towards to the exit doors of the environment.



Figures 9,10,11: Guided groups going to the counter to buy tickets, going sit down and checking the timetable

Figures 12,13: Autonomous crowd walking on the train station.

6 Conclusions

We have described in this paper the interaction paradigm existents in the ViCrowd Model and their utilisation in the Client/Server architecture. Basically, there are four main challenges we dealt in this paper are:

1. The interaction paradigms which intend to provide generic functions to interact and guide crowds in ViCrowd
2. Client/Server system and the protocol in order to provide an open architecture where clients can easily communicate
3. The priority aspects which provide some autonomy to the server in order to decide between synchronous events which one have been sent to ViCrowd client
4. Optimisation of exchanged messages due to display associated to client responsible for execution of low-level behaviours. Then, information like position of agents or body posture of each agent in each time of simulation do not require to be transferred to others clients.

Actually, the Client/Server architecture has been used in the framework of The Virtual City, a human populated virtual city where different kinds of simulation can occur [FAR98]. For instance, interaction with “smart objects” [KAL98], simulation of autonomous agents [BOR99]. Future research should focus on giving still more autonomy for the server in order to improve the priority rules. At the moment, each client decides the priority of each stimulus. Our idea is to provide the server with behavioural rules in order to be able to change priorities of stimulus depending on the nature of messages. For example, one EVE stimulus (crowd event type – See Section 4.2) normally has to be higher priority than a HLB stimulus (high-level behaviour stimulus). However, if the clients defined it in a different way, the virtual crowd can react as a function of the HLB, e.g., going to the restaurant during a panic situation.

Acknowledgements

The authors thank to Elsa Schweiss, Nathalie Farenc, Marcelo Kallmann and Angela Caicedo for their participation in the development of some clients. The research was sponsored by the Swiss National Research Foundation, the Federal Office of Education and Science in the framework of the European Project eRENA, FUNDEPE and CAPES (Brazilian offices of Education and Science).

References

[BLU95] Blumberg, B.; Galyean, T. “Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments”. SIGGRAPH - Computer Graphics Proceedings, pp 47-54. Los Angeles, 1995.

[BOU97] Bouvier, E.; Cohen E.; and Najman. L. "From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation". Journal of Electronic Imaging 6(1), 94-107 (January 1997).

[BOR99] Bordeux, C, Boulic, R and Thalmann, D. “An efficient and Flexible Perception Pipeline for Autonomous Agents”, Proc. Of Eurographics 99, Milan, Sept., 199.

[BRO98] Brogan, D.C., Metoyer, R.A. and Hodgins, J.K. "Dynamically simulated characters in virtual environments". In IEEE Computer Graphics and Applications. Vol.18, No5, pp 58-69. Sept. 1998.

[CHA97] Chavez, A., Moukas, A. and Maes, P. “Challenger: A Multi-agent System for Distributed Resource Allocation”. Proceedings of Autonomous Agents’97. ACM Press, Feb, 1997. Marina Del Rey, CA USA, pp. 323 – 332.

[FAR98] Farenc, N.; Boulic, R.; Thalmann, D. “An Informed Environment Dedicated to the Simulation of Virtual Humans in Urban Context”. Proceedings of EUROGRAPHICS'99. Vol. 18 (1999), Sept. 1999. (to appear)

[KALL98] Kallmann, M. and Thalmann, D. “Modeling Objects for Interaction Tasks”, Proc. Eurographics Workshop on Animation and Simulation, 1998.

[LUI90] Luigi, D., Maniezzo, V. “The Rise of Interaction: Intrinsic Simulation Modelling of the Onset of interacting Behavior”. In: Proceedings of First International Conference on Simulation of Adaptive Behavior. MTI Press, 1990.

[MEY94] Meyer, J.A.; Guillot, A. “From SAB90 to SAB94: Four Years of Animat Research”. In: Proceedings of Third International Conference on Simulation of Adaptive Behavior. Brighton, England, 1994.

[MUS97] Musse, S.R. and Thalmann, D. “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.

[MUS98] Musse, S.R., Babski, C., Capin, T. and Thalmann, D. “Crowd Modelling in Collaborative Virtual Environments”. ACM VRST /98, Taiwan

[NOS 96] Noser, H. “A Behavioral Animation System based on L-Systems and Synthetic Sensors for Actors”. PhD thesis. EPFL, Lausanne, Switzerland, 1996.

[PER96] Perlin, K.; Goldberg, A. “Improv:A System for Scripting Interactive Actors I Virtual Worlds”. SIGGRAPH – Computer Graphics Proceedings. Pp. 205-216. New Orleans, 1996.

[REE93] Reeves, W. “Particle Systems – A Technique for Modelling a Class of Fuzzy Objects”, ACM Transactions on Graphics – April, 1993 – Vol. 2, No. 2.

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

[SCH99] 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.

[THAL96] Thalmann, D., “A New Generation of Synthetic Actors: The Interactive Perceptive Actors”. Proc. Pacific Graphics 96, National Chiao Tung University Press, Hsinchu, Taiwan, 1996, pp. 200 – 219.

[TER94] Tu, X. and Terzopoulos, D. “Artificial Fishes: Physics, Locomotion, Perception, Behavior”. Proc. SIGGRAPH ’94, Computer Graphics, July 1994.

[WOO95] Wooldridge, M. and Jennings, N. “Intelligent Agents: Theory and Practice”. Knowledge Engineering Review, 10(2), June 1995. Cambridge University Press, 1995.

[ZEL91] Zeltzer, D. “Task-level Graphical Simulation: Abstraction, Representation and Control”. Making them Move: Mechanics, Control and Animation of Articulated Figures. Edited by N. Badler, B. Barsky and D. Zeltzer. Pp 3-33. 1991.

Filename:	Egcas4.doc
Directory:	Z:\people\soraia\Textes\Papers\EGCAS99
Template:	C:\Antje\temp\TestWV\sv-lncs.dot
Title:	Level of Autonomy for Virtual Human Agents
Subject:	
Author:	So&Mito
Keywords:	
Comments:	
Creation Date:	02/28/99 4:57
Change Number:	42
Last Saved On:	05/30/99 6:52
Last Saved By:	soraia
Total Editing Time:	1,635 Minutes
Last Printed On:	06/11/99 2:00

As of Last Complete Printing

Number of Pages:	11
Number of Words:	5,307 (approx.)
Number of Characters:	30,250 (approx.)