FreeWalk/*Q*: Social Interaction Platform in Virtual Space

Hideyuki Nakanishi[1], Toru Ishida[1,2]
[1]Department of Social Informatics, Kyoto University
[2]JST CREST Digital City Project
Kyoto 606-8501, JAPAN
+81-75-753-4821
{nakanishi, ishida}@i.kyoto-u.ac.jp

ABSTRACT
We have integrated technologies related to virtual social interaction, e.g. virtual environments, visual simulations, and lifelike characters. In our previous efforts to integrate them, the asymmetry between agents and avatars made the systems too complex to be used widely. Another crucial problem we faced is that it took a long time to construct agents that play various roles, since each role needs its specific behavioral repertory. To eliminate these problems, we developed a general-use platform, FreeWalk/*Q*, in which agents and avatars can share the same interaction model and scenario. We created a control mechanism to reduce the behavioral differences between agents and avatars, and a description method to design the external role rather than the internal mechanism. In the development, we found that it was necessary to prepare several topologies of control mechanism and several granular levels of description method.

Categories and Subject Descriptors
H.5.3 **[Information Interfaces and Presentation]**: Group and Organization Interfaces – *Synchronous interaction.*

General Terms
Design, Human Factors, Languages

Keywords
Virtual space, social interaction, agent, avatar, interaction platform, scenario description, virtual community, virtual training, virtual city.

1. INTRODUCTION
We have developed a platform for supporting and simulating social interaction in virtual space called "FreeWalk/*Q*." Our goal is the integration of diverse technologies related to virtual social interaction, e.g. virtual environments, visual simulations, and lifelike characters [24]. These interaction technologies should be integrated in order to get widely spread and utilized, since the application of each technology is limited and their applications overlap with each other. This paper shows how interaction technologies can be integrated and what applications can be enabled by the integration.

VRST'04, November 10-12, 2004, Hong Kong.

In FreeWalk/*Q*, lifelike characters enable virtual collaborative events such as virtual meetings, virtual trainings, and virtual shopping in distributed virtual environments. You can conduct distributed virtual trainings [4][16][30] where you can use lifelike characters as the colleagues of human trainees [26]. FreeWalk/*Q* is not only for training but also for communication and collaboration [3][14][27]. You can use lifelike characters as the facilitators of virtual communities [7]. These characters and the human participants can use verbal and nonverbal communication skills to talk with one another [1]. FreeWalk/*Q* can also be a browser of 3D geographical contents [15] in which lifelike characters guide your navigation [12] and also populate the contents [28].

In our previous efforts to integrate interaction technologies [8][20], we found that the asymmetry between agents (the characters controlled by programs) and avatars (the characters controlled by users) made the systems too complex to be used widely. But a general-use platform should be kept simple so that many people can use it for many applications. The extreme example of the asymmetry is agents and avatars that are not sharing the same virtual space [1]. Even in the existing systems in which agents and avatars can share the same space, the symmetry in the interaction between them has not been considered extensively enough [26].

Another crucial problem we faced is that it took a long time to construct agents that can socially interact with people, since such agents need to play various roles and each role needs its specific behavioral repertory. It should be easier to design the external role of an agent instead of its internal mechanism. Previous studies have focused on the internal mechanism rather than the external role [12].

To solve the symmetry problem, we developed the FreeWalk interaction platform. FreeWalk has a common interaction model for both agents and avatars while at the same time having different interfaces for them, so they can interact with each other based on the same model. Agents are controlled through the application program interface (API). Avatars are controlled through the user interface (UI). FreeWalk does not care about whether each character is an agent or an avatar.

To solve the construction problem, we developed the scenario description language *Q*. *Q* helps the designer of agents to describe their external roles as interaction scenarios that define the

interaction rules of each role. The language processor of *Q* is connected to the API of FreeWalk so that each agent behaves according to the assigned interaction scenario. *Q* enables the agent designer to construct agents rapidly for any virtual collaborative event.

2. INTERACTION PLATFORM

FreeWalk is a platform for constructing virtual collaborative events in which agents and people can socially interact with each other in a distributed virtual space. Figure 1 roughly shows the distributed architecture of FreeWalk. An agent is controlled through the platform's API. A human participant enters the virtual space as an avatar, which he/she controls through the UI devices connected to the platform. Each character can be controlled from any client. FreeWalk is a hybrid architecture in which the server administrates only the list of current members existing in the virtual space and each client administrates the current states of all characters. This architecture enables agents and people to interact with each other based on the same interaction model.

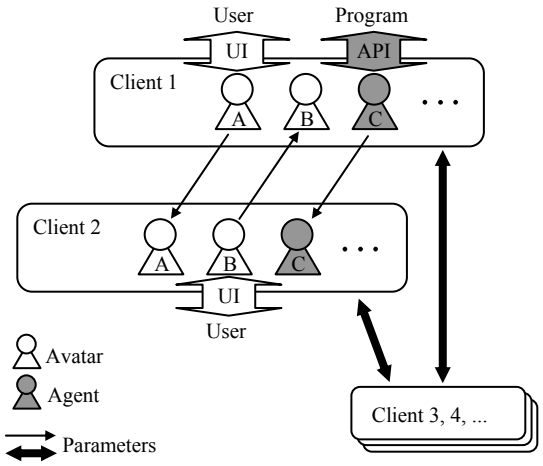


Figure 1. Architecture of FreeWalk

2.1 Interaction Model

The interaction model was designed to provide building blocks for constructing virtual collaborative events. Each character has the following parameters that constitute its actions:

- Location, velocity, and acceleration of the character for its walking movement
- Angle, angle velocity, and angle acceleration of the rotations of the body, the head, and the arms for the deictic gestural movements
- Message and volume of the utterance

These action parameters can be perceived by another character according to the following parameters that restrict its perceptual ability:

- Visual field that limits the angle range of the observable area
- Visual power that limits the distance range of the observable area
- Hearing power that limits the distance range of the audible area

Based on these perception and action parameters, verbal cues and nonverbal cues including the following examples are transmitted by the virtual space:

- Interpersonal distance [5]
- Gaze direction [13]
- Pointing gesture
- Awkward pauses in a conversation [2]

Based on these cues, group behaviors including the following examples finally occur:

- Following others [25]
- Forming a circle to have a conversation [13]
- Eavesdropping on a conversation [2]

Figure 2 illustrates this model. The action parameters of each character produce a cue. The perception parameters of each character determine whether it perceives the cue. When a character perceives cues, its next action is influenced by the cues. For example, the walking direction of a character may change when the character observes a pointing gesture produced by the gestural parameters of another character.

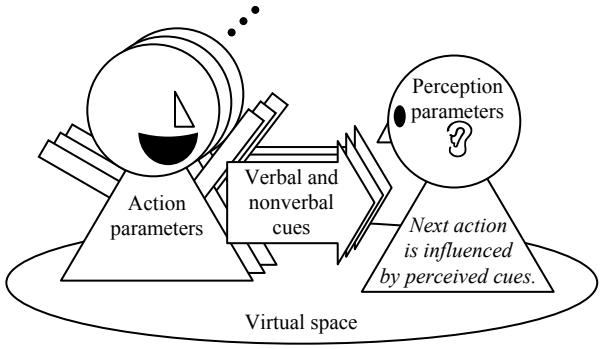


Figure 2. Interaction model

2.2 Interaction Interfaces

The API provides action and perception functions that can be called by programs such as the language processor of *Q*. Some examples of the action functions are *Walk*, *Turn*, *Face*, *Point* and *Speak*. Examples of the perception functions are *Position*, *Observe*, and *Hear*. When an action function is called, the action parameters of the character are modified based on the arguments that specify how to execute the action. When a perception function is called, the action parameters of the characters within the observable or audible area defined by the perception parameters are examined. The purpose of this examination is to find the cues specified by the arguments.

The UI connects the platform with input/output devices such as a keyboard, a mouse, a joystick, a display, a HMD, a speaker, and so on. The action parameters are modified based on how the input devices are controlled. The perception parameters determine the

field of view displayed on the screen, the distance of the far clipping plane, and the sound volume.

The API and UI eliminate the difference between agents and avatars so that programs and human participants have equal ability to produce cues. To eliminate the difference, we created four topologies to connect the platform with the API and the UI. The first one is the sharing topology that is used for the *Walk* function. The second one is the hierarchical topology that is used for the *Turn*, *Face*, and *Point* functions. The third one is the peer-to-peer topology that is used for the *Speak* and *Hear* functions. The last one is the divisional topology that is used for the *Position* and *Observe* functions. Figure 3 summarizes these topologies which are explained below.

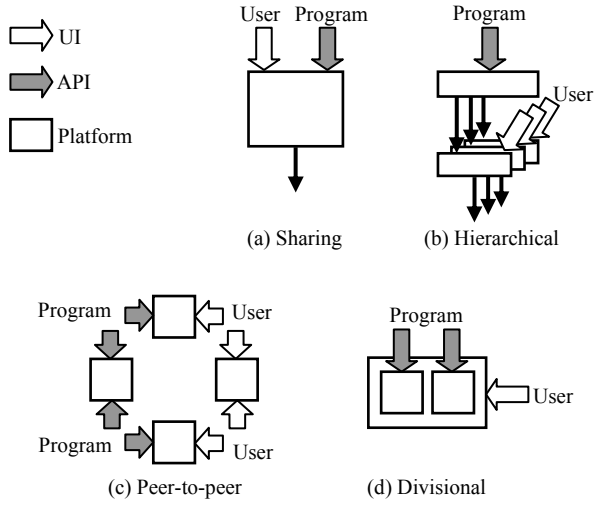


Figure 3. Topology of the control mechanism

2.2.1 Sharing Topology: Walk

We tried to make the walking mechanism be shared by programs and human participants as much as possible. We adopted the sharing topology in which the API and the UI share the process to generate walking movements. Figure 3(a) shows this topology.

The *Walk* function receives the destination of the agent from the program that controls it. Then, the function calculates the walking direction and transfers it to the platform. A human participant manipulates a keyboard or other devices to input to which direction his/her avatar walks. When the platform receives the direction to walk in, it tries to detect collisions with other characters and with the spatial structures such as walls and pillars, based on the pedestrian model [23]. Based on the result of the collision detection, the gait animation is generated.

To keep the correspondence of the walking movement with the graphical representation of the virtual space, the platform uses neither prepared animation data nor simplified collision models. A VRML model that is basically used for drawing the virtual space is also used as a geometric model to detect collisions with the spatial structure and to generate gait animations. The animations are generated based on the hybrid algorithm of kinematics and dynamics [29].

2.2.2 Hierarchical Topology: Turn, Face, and Point

It is annoying to shape deictic gestures that require the coordination of the body, the head, and the arms. For example, a character rotates its body, head, and arm when it explains some object to another character. It is more convenient for the agent designer to choose from prepared gestures. However, it is annoying for a human participant to choose from prepared gestures. It is natural for him/her to move freely his/her body, head, and arms, especially when wearing VR input devices.

To make deictic gestures easy to control through both the API and the UI, we adopted the hierarchical topology in which the API is connected to the abstract layer of the gesturing mechanism while the UI is connected to its primitive layer. Figure 3(b) shows this topology. Each function of the abstract layer is a combination of the functions of the primitive layer. For example, the abstract function *Explain* is a combination of the primitive functions *Turn*, *Face*, *Point*, and also *Walk*. When you call the *Explain* function, you have to specify only to whom and which object to explain. The *Explain* function determines how to move the body, the head, and the arm according to social manners.

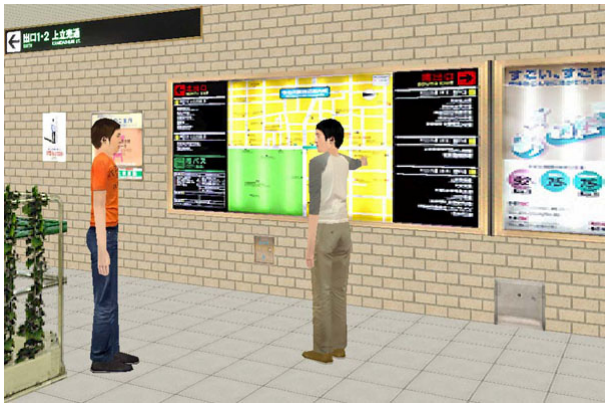
Figure 4 shows how the *Explain* action adapts to the positional relationship among characters. In this figure, the agent explains the map to the avatar in a subway station. You can see that the standing position of the agent changes according to the side from which the avatar comes. The standing position is determined by the following three rules: 1) the agent should not block the view between the map and the avatar; 2) the agent should not invade the personal space of the avatar; and 3) the agent should be seen by the avatar. The direction from the standing position to the avatar becomes the target angle of the head. The direction to the map becomes the target angle of the arm. The body's target angle is decided according to the head's and the arm's target angles. Based on these target angles, the *Turn*, *Face*, and *Point* functions change the angle velocities of the rotations of the body, the head or the arm.

2.2.3 Peer-to-peer Topology: Speak and Hear

Human participants can talk with each other through vocal channels. They can use vocal channels to talk also with agents via speech-to-text (STT) and text-to-speech (TTS) engines. However, agents do not use vocal channels to talk with each other and use text channels instead. To deal with this complex condition of communication channels, we adopted the peer-to-peer topology in which the API and the UI are connected with each other through the platform as shown in Figure 3(c) and explained below.

Figure 5 shows the implementation of this topology. Vocal messages are exchanged between the UIs. Text messages are exchanged between the APIs and also between the API and the UI. The UI is responsible for STT and TTS conversions so that only the channels between the UIs need to transmit audio data. To enable the *Hear* function to search for key phrases, the API stores the text messages that are received from other characters.

The *Speak* function of the API receives the message and the volume of the utterance from the caller program while the UI converts the recorded voice into its message and its volume. The platform transmits the messages and the volumes. In this transmission, the volume attenuates according to each distance between the sending character and each receiving character. If the



Avatar from the left side

Avatar from the right side

Figure 4. Adaptive function *Explain*

perception parameter does not allow a character to hear the message, it is not transmitted. After these processes, the *Hear* function can search the message for the key phrases received from the caller program.

2.2.4 Divisional Topology: Position and Observe

To perceive a situation in the virtual space, human participants see their avatars' views displayed on the screens. On the other hand, programs call the perception functions that examine the action parameters. Since an agent's ability to perceive the situation seriously affects its behavior, the perception functions must be carefully designed to provide ability equal to that of a human participants'. Although equal ability can be achieved by having a single function able to examine all of the action parameters, it becomes too complex to construct humanlike perceptual behaviors. Each function should be kept simple. However, the collection of functions each of which can sense a single cue cannot achieve this equality of abilities.

In the divisional topology, each API function is a division of the functionality of the UI as shown in Figure 3(d). We divided the UI's perceptual functionality into two API functions. The *Position* function can examine the location and the body angle. The *Observe* function can examine the gestural parameters. We think this is an intuitive division. The *Position* function enables programs to obtain the positional and directional relationship

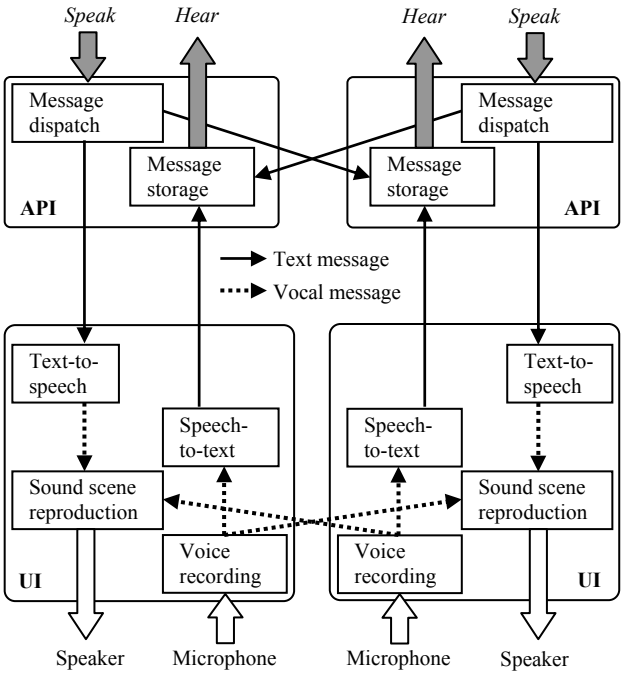


Figure 5. Implementation of communication channels

among characters, e.g. approaching, following, and gathering. The *Observe* function enables programs to know what the character is doing, e.g. inviting and leading.

3. SCENARIO DESCRIPTION LANGUAGE

Q is a language for describing an agent's external role through an "interaction scenario", which is an extended finite state machine whose input is the perceived cues, whose output is an action, and where each state corresponds to each scene. Each scene includes a set of interaction rules, each of which is a couple made up of a conditional cue and the consequent series of actions. Each rule is of the form: "if the agent perceives the event *A*, then the agent executes the actions *B* and *C*."

In *Q*, it is easy to describe an agent's complex role, since its scenario can be divided into a number of scenes. When a scenario writer specifies the rule set of each scene, he/she needs to consider only what may happen within the scene. Moreover, scenarios can be written either with a lot of detail or roughly, since the syntax of *Q* does not care about the granularity of cues and actions. When scenario writers want to specify the scenario in detail, they can use fine-grained cues and actions.

In FreeWalk, each agent behaves according to the assigned scenario. FreeWalk and the language processor of *Q* are connected by a shared memory through which the *Q* processor calls FreeWalk's API functions to evaluate cues and actions described in the current scene.

3.1 Specifying Scenarios

Q is an extension of Scheme, a Lisp programming language dialect. A very short example of a scenario is presented below. Cues start with a question mark and actions start with an

exclamation point. Cues are events that trigger interaction. Cues keep on waiting for the specified event until it is observed successfully. No cue can have any effect on the environment. It is actions that change the environment. When cues and actions are evaluated, the corresponding API functions are called.

```
(defscenario reception
  (scene1
   ((?hear "Hello" :from $x)
    (!speak "Hello" :to $x)
    (go scene2))
   ((?hear "Bye")
    (go scene3)))
  (scene2
   ((?hear "Hello" :from $x)
    (!walk :to $x)
    (!speak "Yes, may I help you?" :to $x))
   (otherwise (go scene3)))
  (scene3 ...))
```

In this example, the scenario called 'reception' includes scene1, scene2, and so on. In each scene, multiple cues can be evaluated simultaneously. After either cue becomes true, the corresponding action is evaluated. If none of the cues are satisfied, the 'otherwise' clause is evaluated. Though scenarios are written in the form of simple scene transitions, it is possible to describe complex roles since each scene can contain its unique observational rules for agents. You can see that the same observation yields different actions in different scenes. In scene1, the agent says "Hello" when it hears someone say "Hello." In scene2, however, the agent approaches the person and responds with "Yes, may I help you?" when it hears "Hello" again.

Since *Q* is a general-purpose scenario description language, with a lot of description freedom, special-purpose application design at the low level would be inadequate. We thus introduced Interaction Pattern Cards (IPC) to capture the interaction patterns in each application, therefore providing a higher level of abstraction. Figure 6(b) shows the scenario for a virtual evacuation and Figure 6(a) shows an IPC equivalent of the scenario [17]. Scenario writers can use spreadsheet software to fill in the card. IPC provides a pattern language, and so it should be carefully designed by analyzing the interactions in each application. You can learn more about *Q* in [10].

3.2 Level of Detail of Interactions

Just like the level of detail technique is useful for rendering a landscape, it is useful for describing interaction. *Q* can provide different levels of cues and actions. FreeWalk/*Q* provides both fine-grained and coarse-grained actions. When you do not care about details, you can use `!explain` instead of `!walk`, `!turn`, `!face`, and `!point`. It is easy to prepare and use different levels of actions, since unlike functions in programming languages, *Q* does not define the semantics of actions.

In *Q*, actions are basically sequential to avoid complexity. The next action begins when the previous action is completed. However, it is difficult to use fine-grained actions such as `!walk` and `!speak` to describe more abstract behaviors if the overlapping between actions cannot be allowed at all. Thus, *Q* supports non-blocking actions that can be executed in parallel. For example, `!walk` can be a non-blocking action, since we can speak and walk at the same time. To represent non-blocking actions, we use the notation `!!walk` as you can see in Figure 6(b). If we use `!!walk` in the preceding example, the agent says "Yes, may I help you?" just after he starts walking.

3.3 Executing Scenarios

Q's processes are event-driven, and call FreeWalk's API functions asynchronously, while FreeWalk is a cyclic process like other visual simulators and can execute the called functions at regular intervals. We elaborated a connection mechanism to cope with this incompatibility.

When *Q*'s processes call functions, the called functions are listed in the shared memory. FreeWalk repeats the cycle of changing the action parameters and drawing characters based on the amount of change per cycle. When starting the next cycle, FreeWalk begins executing the listed functions. If the function is an action that takes some time to complete, FreeWalk continues to change the action parameters across several cycles. The amount of change in each cycle is determined by the period of the time elapsed since the previous cycle. For example, the *Walk* function is repeated to draw a character that goes forward another more in each frame until it reaches the destination indicated by the caller program.

4. APPLICATIONS

In this section, we describe our previous studies on applying virtual space to social interaction rather than preliminary or imaginary applications of FreeWalk/*Q*. FreeWalk has been in development for more than eight years and the FreeWalk/*Q* is the latest product [21]. During this period of time, we have conducted a lot of experiments to investigate how virtual space can support and simulate social interaction. In these experiments, diverse potential applications have emerged.

4.1 Virtual Communities

It has become possible to enjoy worldwide communication from a living room. However, this kind of distant communication is limited to discussions related to shared goals or interests. This limitation had already been reported before the wide spreading of the Internet. In the early 1990's, many researchers tried to produce informal daily conversations among distributed workplaces such as different floors and distant buildings. It was known that daily conversation sustains human relations, contributes to collaborative working, and is indispensable to an organization. We developed the first version of FreeWalk, which combines video-mediated communication with virtual space, to convey conversational awareness through freely walking video avatars [19]. Although the researchers proposed various brilliant ideas to extend videoconferencing, they could not generate daily conversations between remote places that would take place as frequently as in face-to-face environments [11].

Card ID	1	Card Name	Follow-me	Card Type	Guidance Card
Initiate Guidance		Object of Observation	State	Guiding Actions	
		Exit A	Open	Put on a cap	
				Choose an evacuee (one closest to me)	
				Approach (Evacuee)	
				Speak (Follow me)	
				Start to walk (Point Y, Exit B)	
Guidance [Condition]		Evacuee's position/direction	My position/direction	Guiding Actions	
Guidance [Repeat]		Evacuee's position/direction	My position/direction	Guiding Actions	
		More than 3.0m apart from me		Stop	
				Turn direction (Evacuee)	
		Within 1.5m from me	Positioned at the left room	Start to walk (Point Y, Exit B)	
			Positioned at the right room	Start to walk (Exit B)	
		Exit B		[Terminate Repeat]	
Terminate Guidance		Guiding Actions			
		Walk (Outside the room)			

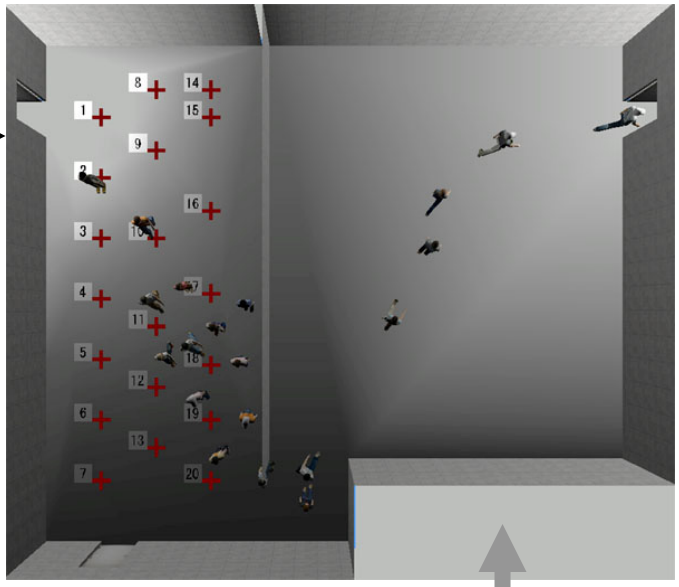
(a) Interaction Pattern Cards

```
(defscenario Follow-Me-Method ()
  (let ((target '()))
    (scene1
     ((?observe :name Exit-A :state Open)
      (!change :image Putting-on-a-Cap)
      (set! target (Choose-Closest-Evacuee))
      (!approach :to target)
      (!speak :to target :sentence "Follow me")
      (!!walk :route (list Point-Y Exit-B))
      (go scene2)))
    (scene2
     ((?position :name target :distance '(3.0 9.9))
      (!finish :action "walk")
      (!!turn :to target)
      (go scene2))
     ((?position :name target :distance '(0.0 1.5))
      (guard
       ((?position :name me :at Left-Room)
        (!!walk :route (list Point-Y Exit-B))
        (go scene2))
       ((?position :name me :at Right-Room)
        (!!walk :route (list Exit-B))
        (go scene2))))
     ((?position :name target :at Exit-B)
      (go scene3)))
    (scene3
     (#t
      (!walk :route (list Outside))))))
```

(b) *Q* scenario

FreeWalk simulation

(c) First-person view



Experience (d) Bird's-eye view Observe

(e) Human participants

Figure 6. Evacuation simulation

Currently, virtual communities are very common. However, they are still far from being a substitute for real-world communities. Internet BBSs seem to be regarded as online places for free discussions without worrying about human relations, rather than for maintaining human relations. Even though the importance of international collaboration is increasing, we have not yet found a

way to support daily conversation between geographically separated places. It can be said that it is almost impossible to support daily conversation by providing only environments such as videoconferencing systems and BBSs. Since this conclusion suggested us to upgrade the level of communication support from the environments, we tried to place a third-party agent as virtual space conversation facilitator [8]. Such an agent is not the same as usual software agents because it has the capability to socially interact with people. This kind of agent is called "social agent" [18].

We conducted several experiments in which a social agent intervenes in human conversations in FreeWalk virtual space. Our social agent played the role of facilitator of the first time meetings and provided human subjects a topic to resume their faltering conversations. The agent had communication skills necessary to become a facilitator such as adjusting interpersonal distance [5], controlling gaze direction [13], and eavesdropping on a conversation to detect its faltering [2]. The experiments clarified the interesting effects of social agents on virtual communities. For example, we found that social agents can be influential enough to control human relations according to the balance theory [20].

4.2 Virtual Trainings

Since smooth evacuation is related to the safety of our lives, we are taught how to evacuate in preparation for a disaster. For example, fire drills are conducted in schools. However, it is rare to conduct fire drills in large-scale public spaces such as a central railway station, even though it is a place where a vast amount of people gathers. It is obvious that we can significantly benefit from crowd simulations for learning about evacuation in such city spaces. Multi-agent simulations can effectively deal with the complex behavior of escaping crowds in evacuation [6][23]. However, conventional multi-agent simulations are not tailored for learning evacuation. Since they are designed for analyzing crowd behavior, they do not take human involvement much into account. For example, crowds are usually represented as moving particles. It is not very easy to interpret such a symbolic representation. Moreover, it is almost impossible for users to become simulated crowds to experience a virtual evacuation. FreeWalk/*Q* can compensate these incapabilities.

In FreeWalk/*Q*, a virtual city space and virtual crowds are represented as 3D photo-based models. Since users can change the viewpoint of a simulation freely, they can observe the simulation through a bird's-eye view (BE) and also experience it through their first-person views (FP). In FP, users can practice decision making, since they control their avatars according to their decisions made based on what the view informs (Figure 6(c)). BE is more effective in understanding an overall crowd behavior than FP (Figure 6(d)). Since both views have different efficacies, we compared them and also derived their synergic effects to find the best way to learn evacuation and what kind of superiority FP has. A lot of simulations were conducted to enable this comparison. In the FP simulations, six evacuees were subjects' avatars (see Figure 6(e)) and the other ten evacuees were agents. BE simulations included only agents. You can find more details of this experiment in [22]. As a result of our analysis, we found that the best way to learn evacuation in a virtual city is that learners control their avatars in the crowd simulation after they observe the overall crowd behavior.

Figure 7. Shopping street

4.3 Virtual Cities

In our project [9], we constructed the 3D model of a real-world shopping street and agents that can guide people in the virtual shopping street of FreeWalk/*Q* and also populate it as shown in Figure 7. We constructed them not for online shopping but for real-world shopping. Visitors are able not only to feel the atmosphere but also plan and practice the routes to walk around before visiting the real-world site. Residents can easily track renewals and replacements of shops. We found that the combination of pictures taken by digital cameras and a simple geometric model based on the map worked well for these purposes. We also found that an avatar's sighting behavior, which indicates his/her attention and interests, could be a strong cue that invokes some introduction from the guide agent.

5. CONCLUSION

In a general-use platform for virtual social interaction, a lot of heterogeneous actors and roles must be involved. We described how to implement and use a platform in which software agents and people's avatars can share the same environment, the same interaction model, and the same interaction scenario. In the design of such an integrated platform, the integration between agents and avatars is a key issue. We proposed the control mechanism and the description method to integrate their behaviors. We found that the topology of control mechanism and the granular level of description method depend on each behavior. Therefore, it is necessary to prepare several topologies and description levels.

6. ACKNOWLEDGMENTS

We thank Ken Tsutsuguchi, Kaori Sugiyama, Toyokazu Itakura, CRC Solutions, Mathematical Systems, and CAD Center for their efforts in the development. We received a lot of support in the construction of the evacuation simulation from Toshio Sugiman and Shigeyuki Okazaki. The source code is available at http://www.lab7.kuis.kyoto-u.ac.jp/freewalk/ and http://www.digitalcity.jst.go.jp/Q/.

7. REFERENCES

[1] Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjalmsson H. and Yan H. Embodiment in Conversational Interfaces: Rea. CHI99, 520-527, 1999.

[2] Clark, H.H. Using Language, Cambridge University Press, 1996.

[3] Greenhalgh C. and Benford S. Massive: A Collaborative Virtual Environment for Teleconferencing. ACM Transactions on Computer-Human Interaction, 2(3), 239-261, 1995.

[4] Hagsand, O. Interactive Multiuser VEs in the DIVE System. IEEE MultiMedia, 3(1), 30-39, 1996.

[5] Hall, E.T. The Hidden Dimension. Doubleday, 1966.

[6] Helbing, D., Farkas, I.J. and Vicsek, T. Simulating Dynamical Features of Escape Panic. Nature, 407(6803), 487-490, 2000.

[7] Isbell, C.L., Kearns, M., Kormann, D., Singh, S. and Stone, P. Cobot in LambdaMoo: A Social Statistics Agent. AAAI2000, 36-41, 2000.

[8] Isbister, K., Nakanishi, H., Ishida, T. and Nass, C. Helper Agent: Designing an Assistant for Human-Human Interaction in a Virtual Meeting Space. CHI2000, 57-64, 2000.

[9] Ishida, T. Digital City Kyoto: Social Information Infrastructure for Everyday Life, CACM, 45(7), 76-81, 2002.

[10] Ishida, T. *Q*: A Scenario Description Language for Interactive Agents. IEEE Computer, 35(11), 54-59, 2002.

[11] Jancke, G., Venolia, G.D., Grudin, J., Cadiz, J.J. and Gupta, A. Linking Public Spaces: Technical and Social Issues. CHI2003, 530-537, 2003.

[12] Johnson, W.L., Rickel, J.W. and Lester, J.C. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. International Journal of Artificial Intelligence in Education, 11, 47-78, 2000.

[13] Kendon, A. Spatial Organization in Social Encounters: the F-formation System. A. Kendon, Ed., Conducting Interaction: Patterns of Behavior in Focused Encounters, Cambridge University Press, 209-237, 1990.

[14] Lea, R., Honda, Y., Matsuda K. and Matsuda, S. Community Place: Architecture and Performance. VRML97, 41-50, 1997.

[15] Linturi, R., Koivunen, M. and Sulkanen, J. Helsinki Arena 2000 - Augmenting a Real City to a Virtual One. T. Ishida, K. Isbister Ed., Digital Cities, Technologies, Experiences, and Future Perspectives, LNCS 1765, 83-96. 2000.

[16] Macedonia, M.R., Zyda, M.J., Pratt, D.R., Barham, P.T. and Zeswitz, S. NPSNET: A Network Software Architecture for Large-Scale Virtual Environments. Presence, 3(4), 265-287, 1994.

[17] Murakami, Y., Ishida, T., Kawasoe, T. and Hishiyama, R. Scenario Description for Multi-agent Simulation. AAMAS2003, 369-376, 2003.

[18] Nagao, K. and Takeuchi, A. Social Interaction: Multimodal Conversation with Social Agents. AAAI94, 22-28, 1994.

[19] Nakanishi, H., Yoshida, C., Nishimura, T. and Ishida, T. FreeWalk: A 3D Virtual Space for Casual Meetings. IEEE MultiMedia, 6(2), 20-28, 1999.

[20] Nakanishi, H., Nakazawa, S., Ishida, T., Takanashi, K. and Isbister, K. Can Software Agents Influence Human Relations? - Balance Theory in Agent-mediated Communities -. AAMAS2003, 717-724, 2003.

[21] Nakanishi, H. FreeWalk: A Social Interaction Platform for Group Behavior in a Virtual Space. International Journal of Human Computer Studies, 60(4), 421-454, 2004.

[22] Nakanishi, H., Koizumi, S., Ishida, T. and Ito, H. Transcendent Communication: Location-Based Guidance for Large-Scale Public Spaces. CHI2004, 655-662, 2004.

[23] Okazaki, S. and Matsushita, S. A Study of Simulation Model for Pedestrian Movement with Evacuation and Queuing. International Conference on Engineering for Crowd Safety, 271-280, 1993.

[24] Prendinger, H. and Ishizuka, M. Life-Like Characters: Tools, Affective Functions, and Applications. Springer Verlag, 2004.

[25] Reynolds, C.W. Flocks, Herds, and Schools: A Distributed Behavioral Model. SIGGRAPH87, 25-34, 1987.

[26] Rickel, J. and Johnson, W. L. Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. Applied Artificial Intelligence, Vol. 13, pp. 343-382, 1999.

[27] Sugawara, S., Suzuki, G., Nagashima, Y., Matsuura, M., Tanigawa H. and Moriuchi, M. InterSpace: Networked Virtual World for Visual Communication. IEICE Transactions on Information and Systems, E77-D(12), 1344-1349, 1994.

[28] Tecchia, F., Loscos, C. and Chrysanthou, Y. Image-Based Crowd Rendering. IEEE Computer Graphics and Applications. 22(2), 36-43, 2002.

[29] Tsutsuguchi, K., Shimada, S., Suenaga, Y. Sonehara, N. and Ohtsuka, S. Human Walking Animation based on Foot Reaction Force in the Three-dimensional Virtual World. Journal of Visualization and Computer Animation, 11(1), 3-16, 2000.

[30] Waters, R.C. and Barrus, J.W. The Rise of Shared Virtual Environments. IEEE Spectrum, 34(3), 20-25, 1997.