

# Participatory Simulation Environment gumonji/Q: A Network Game Empowered by Agents

Shohei Yamane<sup>1</sup>, Shoichi Sawada<sup>1</sup>, Hiromitsu Hattori<sup>1</sup>, Marika Odagaki<sup>2</sup>,  
Kengo Nakajima<sup>2</sup>, and Toru Ishida<sup>1</sup>

<sup>1</sup> Department of Social Informatics, Kyoto University  
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan

{yamane,sawada}@ai.soc.i.kyoto-u.ac.jp, {hattori,ishida}@i.kyoto-u.ac.jp

<sup>2</sup> Community Engine Inc., 4-31-8 Yoyogi Shibuya-ku Tokyo 151-0053 Japan

**Abstract.** Network games are attracting attention as simulation platforms for social experiments because of their rich visualization performance and scalability. Our objective in this study is to develop a participatory simulation platform on a network game. Unlike non player characters (NPCs) in network games, agents in a participatory multiagent-based simulation (PMAS) should behave as real-world humans according to behavior models. We developed a novel networked participatory simulation platform called *gumonji/Q* by integrating scenario description language *Q* with the network game *gumonji*. This paper details the implementation of *gumonji/Q*. In order to connect *Q* and *gumonji*, we implement communication sub-components that realize TCP/IP communication between them, and a scenario translator to convert a request from *Q* into a sequence of operators. This makes it possible for the *gumonji* simulator to deal with human-controlled avatars and *Q*-controlled agents in a unified way.

**Keywords:** Multiagent Simulation, Simulation Platform, Participatory Simulation, Gaming, Networked Simulator.

## 1 Introduction

Network games are attracting attention as simulation platforms for social experiments. Virtual worlds can be made available on the network, and many humans can participate in them. Network games can be used as simulation environments since they enable us to observe the social phenomena caused by the interactions among multiple actors. In addition, many network games have rich 3D visualization performance. This helps users to observe the simulation environment and other users' activities. These features of network games can be great advantage if we can use them as platforms for participatory multiagent-based simulation (PMAS) in which some humans participate by controlling agents. PMAS enables us to observe how humans behave in the simulation and is used as a platform for participatory modeling[1,2,3,4].

Our objective in this study is to develop a new participatory simulation platform based on a network game. We address the following two issues; 1) Since we do not target a specific simulation environment, extensibility is a key requirement. The platform should have extensibility to support the customization of the environment and agents' actions; 2) When a PMAS is used for designing socially embedded systems, definitions of agent behavior are given by the PMAS conductor and are changed repeatedly according to the simulation process.

Against this background, we have developed `gumonji/Q`, a novel networked participatory simulation platform, by integrating the scenario description language `Q` and the network game `gumonji`<sup>1</sup>. `Q` is a scenario description language that enables us to describe complex social interactions among agents and their surrounding world (humans, other agents, simulation environment)[5]. `gumonji` is a network game that offers visually rich playing fields for virtual life. Users can engage in a large variety of activities via characters on that field as though they are in the real-world. “`gumonji`” also has extensibility for the customization of the environment by plugins. `gumonji/Q` is thus able to realize a participatory simulation environment where “human-agent” and “human(agent)-environment” relationships are well reproduce; it allows us to understand a large variety of social phenomena.

The remainder of the paper is organized as follows. First, we show the background of our research; related works and the two primary software programs (`Q` and `gumonji`). Second, we present the new participatory simulation platform `gumonji/Q`. We then detail the implementation of `gumonji/Q` and show an example of participatory modeling on `gumonji/Q`. Concluding remarks are given in the final section.

## 2 Background

### 2.1 Related Works

FreeWalk is a platform for constructing virtual collaborative events in which agents and humans can socially interact with each other in a virtual space[6]. 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. The primary objective of FreeWalk is to realize a virtual space for communication and collaboration between humans and agents. However, while humans and agents can engage in diverse social interactions in FreeWalk, it does not simulate the environment in detail, which means humans cannot alter their surrounding environment and receive feedback. To provide better real-world experience, we need an interactive simulation environment.

CORMAS can be used to build simulation models of coordination modes between individuals and groups who jointly exploit the resources[7]. In CORMAS, users can define the diffusion of environmental changes, and agent's behaviors governed by the surrounding environment. The computational model behind

---

<sup>1</sup> “`gumonji`” is a product of Community Engine Inc. (<http://www.gumonji.net/>)

CORMAS simulations is a cellular automaton. A natural environment is modeled as a two dimensional mesh, and the diffusion between neighboring cells is calculated at each unit time. CORMAS is useful to describe interactions between a natural environment and humans. However, while FreeWalk emphasizes graphics functions, CORMAS just shows abstract graphics; its low level visual information makes it is hard for humans to behave as they would in the real-world.

## 2.2 Scenario Description Language *Q*

*Q* is a scenario description language for multiagent systems that allows us to define how agents are expected to interact with their environment including humans and other agents[5]. *Q* is suitable for describing complex social interactions[8,9]. *Q* scenarios foster the emergence of dialogs between agent designers (computing professionals) and application designers (scenario writers)[10]. The computational model behind *Q* scenarios is an extended finite state automaton, which is commonly used for describing communication protocols. By using *Q*, users can directly create scenario descriptions from extended finite state automata. *Q*'s language functionality is summarized as follows:

- **Cues and Actions**

An event that triggers interaction is called a cue. Cues are used to request agents to observe their environment. Cues keep waiting for the event specified until observation is completed successfully. Comparable to cues, actions are used to request agents to change their environment.

- **Scenarios**

Guarded commands are introduced for the situation wherein we need to observe multiple cues simultaneously. A guarded command combines cues and actions. After one of the cues becomes true, the corresponding action is performed. A scenario is used for describing protocols in the form of an extended finite state machine, where each state is defined as a guarded command. Scenarios can be called from other scenarios.

- **Agents and Avatars**

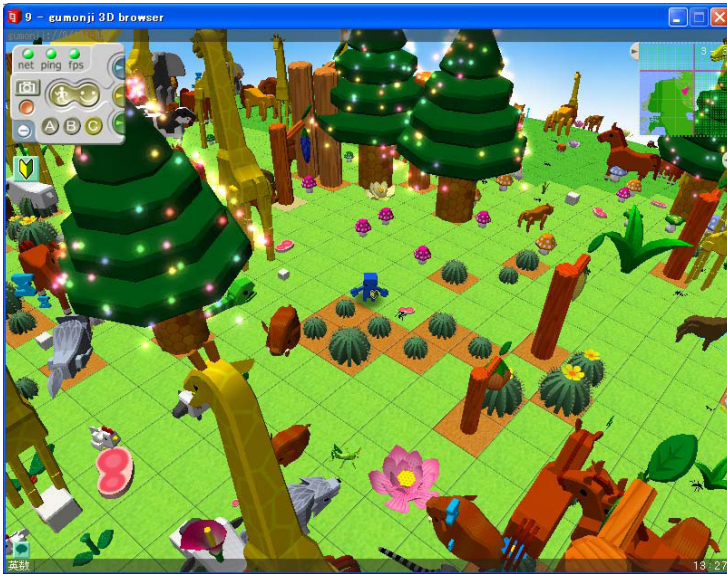
Agents, avatars and a crowd of agents can be defined. An agent is defined by a scenario that specifies what the agent is to do. Even if a crowd of agents executes the same scenario, the agents exhibit different actions as they interact with their local environment (including other agents and humans). Avatars controlled by humans do not require any scenario. However, avatars can have scenarios if it is necessary to constrain their behavior.

## 2.3 Network Game *gumonji*

*gumonji* is a network game developed and released by Community Engine Inc. that offers environmental simulations. Figure 1 shows a snapshot of the virtual space of *gumonji*. In *gumonji*, animals and plants exist in a virtual space, and the atmosphere and water circulate according to realistic physical laws. Users participate in the simulation environment by operating characters (avatars<sup>2</sup>)

---

<sup>2</sup> In this paper, we call a human-controlled character “avatar” in order to distinguish human-controlled character(agent) and *Q*-controlled agent



**Fig. 1.** A Screenshot of gumonji

in the virtual space, and can manipulate living things through the actions of their characters. The changes made to a living thing persist over time, and the change in the environment is reflected in the 3D virtual space. Many users can participate in the same environment, and communicate with other users through the actions and dialogs of the characters.

In *gumonji*, all users have own simulation environment on individual computers and participate in other users' simulation environment through P2P network. By using a P2P network, users can move freely across multiple simulation environments and participate anytime and anywhere.

Since *gumonji* is a network game where many users participate and communicate with each other, the actors in the environment are the characters operated by users. Therefore, *gumonji* has no function to construct or run autonomous agents. To realize participatory simulations, it is necessary to add a function that can ascribe diverse interaction patterns to different agents.

### 3 Participatory Simulation Platform *gumonji/Q*

Our approach has the following advantages.

- **Network games have interfaces that enable humans to act in the virtual space.**

Humans observe the virtual space and operate avatars visually in network games, so humans can act naturally in the virtual space.

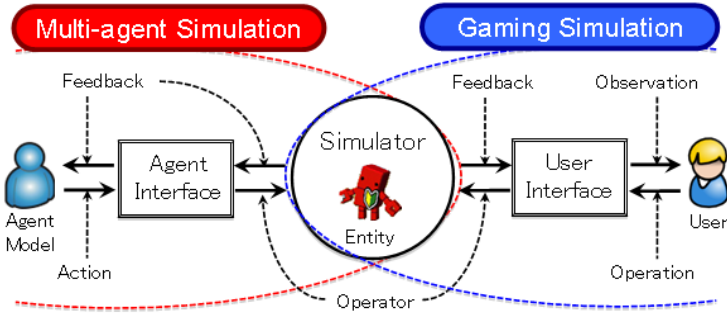


Fig. 2. Participatory simulation with network game and multiagent system

– **Many humans can interact with each other in the virtual environment.**

Network games provide virtual environments wherein many humans can participate. Humans can participate in the environment freely and interact with each other in the virtual space.

– **Humans can participate from everywhere at anytime via the Internet.**

Due to the Internet, human participation is not restricted to specific geographical areas.

– **The virtual environment exhibits persistence.**

The virtual environment exists continuously regardless of the participation of humans. Though the number of participants fluctuates, the virtual environment progresses continuously and keeps changing.

We extended a network game by integrating into it a multi-agent system. Figure 2 shows how this was done. Right side of the figure shows the network game part and the left side shows the multi-agent system part. This integration realizes a platform where humans and agents can jointly participate in simulations.

Humans participate through the user interface provided by the game. The user interface conveys the operations by humans to the simulator, and an output function, which conveys the latest virtual environment to the humans. The input devices include the mouse and keyboard. Input operations are converted into commands that control avatar actions. The actions in and changes to the simulation environment are converted into graphics and sounds, and conveyed to humans through displays and speakers.

An agent interface is newly added to allow agents to participate in the simulation. Its input function conveys requests by agents to the simulator and its output function conveys the latest virtual environment to the agents. Based on given scenarios, agents decide actions according to the surrounding circumstances and send requests for actions. Requests are converted into the operators that trigger the actions of avatars. The results of all actions and changes to the

simulation environment are converted into the data that can be processed by the agents.

Our platform for participatory simulation consists of the server part, which is responsible for the simulation environment, and the client part, which is the user interface for human participation. We used the `gumonji` client as it is, and established a new server-based simulator responsible for the simulation environment in which both humans and agents can participate.

## 4 Implementation of `gumonji/Q`

Any truly useful participatory simulator must support two types of interaction: interaction between entities (humans, agents, etc.) and interaction between entities and the environment. As we mentioned in the previous section, to control the interaction between entities, the scenario description language `Q` is useful, and we can utilize `gumonji` to establish the interactive simulation environment. This section shows how we integrated these two components in constructing the simulation platform `gumonji/Q`.

### 4.1 Overview

Figure 3 shows architecture of `gumonji/Q`. Colored boxes are newly implemented sub-components. As shown in this figure, `gumonji/Q` consists of `gumonji`'s primary components, `gumonji` Zone Server and `gumonji` Client, and a `Q` processor. Zone Server has an environment simulator (just written as “simulator” in

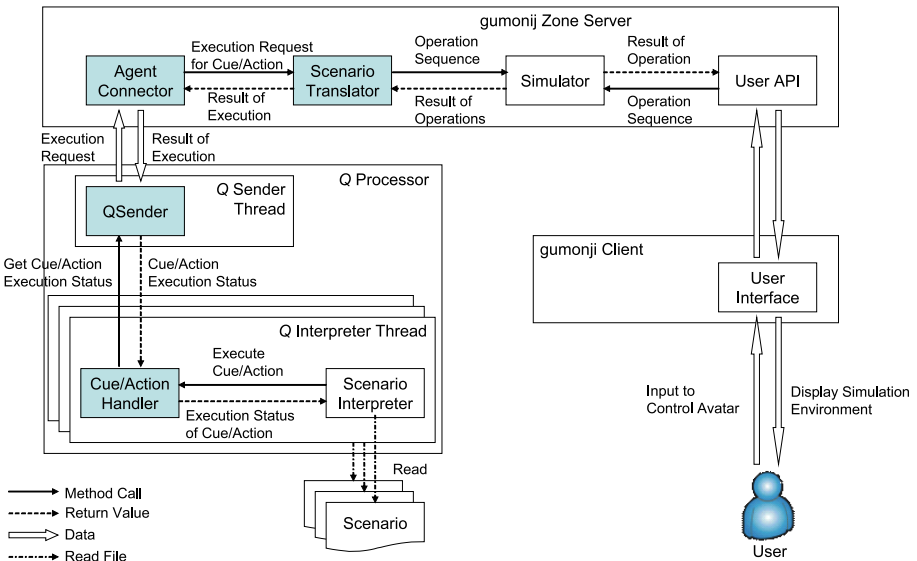


Fig. 3. An overview of `gumonji/Q`

the figure) as its sub-component which simulates the natural environment and maintains all objects, such as human-controlled avatars, plants, animals, in the environment. A user can access the Zone Server from a *gumonji* Client on each computer, and control an avatar in the environment. To put it concretely, user's input data is converted to a sequence of operators that control an avatar based on *gumonji*'s User API. The *Q* interpreter, which transforms a *gumonji* character into an autonomous agent, is connected to the Zone Server. In order to achieve a connection between *gumonji*'s Zone Server and *Q* processor, we implemented two sub-components, "Agent Connector" and "*Q* Sender", within the Zone Server and *Q* processor, respectively. These components communicate via TCP/IP. A *Q*-controlled *gumonji* character can act as an autonomous agent. A crucial point is that the environment simulator in Zone Server deals with human-controlled avatars and *Q*-controlled agents in a unified way.

Cue/Action execution requests are sent to Zone Server via *Q* Sender and Cue/Action Handler. Cues and actions are defined as functions in Cue/Action Handler. These functions first call a function for sending execution request messages in *Q* Sender, and then keep checking cue/action execution status to determine if execution is finished or not. One *Q* interpreter thread is run for each scenario and agent pair. When one scenario description is assigned to two agents, two *Q* interpreter threads are created.

*Q* Sender Thread, on which the *Q* Sender process runs, is created only once in each *Q* processor. *Q* Sender manages the cue/action execution list. When the cue/action handler starts a certain cue/action, *Q* Sender adds its execution status to the execution list, and sends an execution request to the agent connector as a TCP/IP message. When *Q* Sender gets a message from the agent connector indicating the completion of a cue/action, *Q* Sender updates the execution status of the cue/action in the cue/action execution list.

*gumonji* cannot interpret the requests issued by the *Q* processor. Thus, we implemented a sub-component, Scenario Translator, which translates an execution request for Cue and Action into an operator sequence. As shown in Figure 3, the User API and Scenario Translator yield identical forms of input. Therefore, the environment simulator can deal with human-controlled avatars and *Q*-controlled agents in a unified way. The Scenario Translator first translates the execution result into a format suitable for *Q* Interpreter. The result is sent to *Q* Interpreter via Agent Connector.

## 4.2 How to Control *gumonji* Avatar by *Q*

We implemented the functions for processing cues and actions. Agents operate avatars using cues and actions according to the described agent models. Figure 4 shows the process of handling cues and actions. We explain the process with example of executing cues and actions. Please note that *Q* sender and cue/action handler are combined and named "*Q* Connector" in Figure 4.

Cues are used to get information of surrounding environment for the purpose of observing the events used to trigger interactions. Agents set cues according to

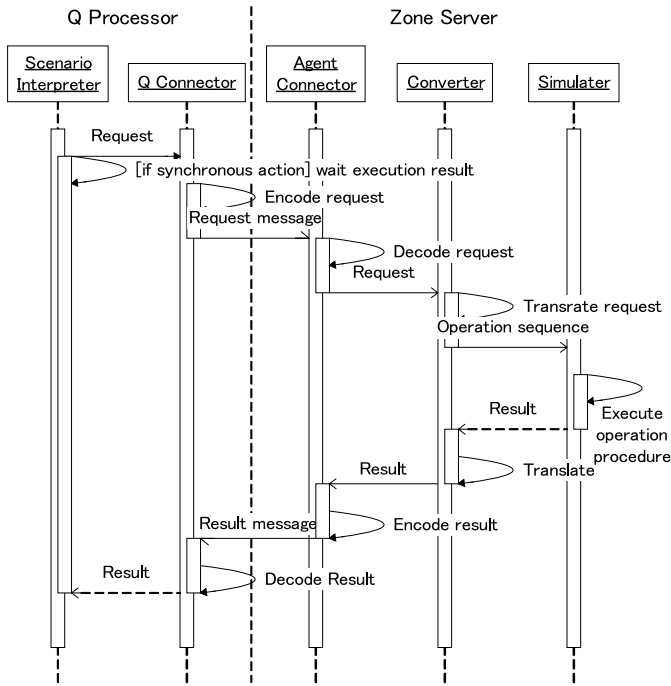


Fig. 4. Process of executing cues and actions

the agent models described in scenarios. Each cue specifies the originating agent and the desired trigger. Requests for cues are sent from *Q Processor* to *gumonji Zone Server*. When *gumonji Zone Server* receives a cue request, it executes observations according to the specified condition. When the specified event is observed, the observation’s results are returned to *Q Processor*. When the agent receives the results of the cue, it begins to perform the actions specified by the scenario.

We implemented seven observation functions for executing cues, such as “hear” and “see”. We implemented general-purpose functions for cues in order for agents to observe events which *gumonji* users can observe via GUI. Agent abilities such as the range of observations can be constrained by the scenario. Scenario writers can constrain the observation abilities of agents freely according to the purpose of the simulation and the situations of agents.

We explain the process of cue execution using the example of the cue “hear”. Figure 5 shows the situation wherein the cue “(?hear-message :message "Hello!" :id \$id)” is executed. This cue means that the agent observes whether the word “Hello!” is said, and gets the ID of the speaker when this expression is spoken. If agent Jiro, who is a yellow colored avatar, said the word “Hello!” as shown in the right figure, the cue is observed and the ID of Jiro is acquired. On the other



Fig. 5. Executing a cue “?hear”

hand, if agent Jiro said the word “Good morning!”, as shown in the left figure, the cue is not observed.

Scenario Interpreter interprets scenarios and requests cue execution according to the agent models described in scenarios. Cue requests are encoded into strings and sent from *Q* Connector to Agent Connector by TCP/IP messaging. Agent Connector decodes received messages and sends them to Cue & Action Converter. In Cue & Action Converter, cue requests are converted into operators which establish observation of the virtual space. In this example, the cue request “(?hear-message :message "Hello!" :id \$id)” is converted into operators that search for the word “Hello!” and get the ID of speaker once the word is captured. The ID of the speaker is returned as the result of cue execution. The results are sent to Cue & Action Converter and converted into a format appropriate for *Q* Interpreter. The acquired ID of speaker is added to the result message at the same time. The result message is encoded as a string and sent from Agent Connector to *Q* Connector via TCP/IP. When *Q* Connector receives the result message, it is decoded and the results are sent to Scenario Interpreter. Through these processes, agents receive the results of cue execution such as ID of Speaker. Acquired information is used in subsequent actions.

We replicated 45 actions that are reasonable for humans to execute via the user interface. The actions include “walk”, “speak”, “pickup”... We gave agents the ability to operate avatars to the same extent as humans. In other words, agents can operate avatars to the same extent as humans.

## 5 Sample of Participatory Simulation

We used *gumonji/Q* to conduct a participatory simulation based on five farmers and 16 farms. Each farmer owns some of the 16 farms. Each farmer is controlled by human participant or *Q* scenario. The aim of each farmer is to maximize

his/her earnings by cultivating, selling, buying, lending and renting the farms. Each step of the simulation consists of following phases.

**Negotiation phase.** A farmer negotiates with other farmers to sell, buy, lend or hire one or more farms. If the farmer rents a certain farm, he/she has to pay the agreed rent to the owner though he/she can get the harvest from the farm.

**Cultivation phase.** For each farm, the farmer decides whether to cultivate the it or not. If he/she thinks the farm is not worth cultivating, he/she can abandon it.

**Simulation phase.** Farmers cultivate the farms, cast seeds and gather the harvest. The growth of crop plants and the condition of the earth are calculated based on realistic models.

**Settlement phase.** Settlement; income and expenditure are displayed for each farmer.

A typical scenario is shown in Figure 6. This scenario describes the behavior model of the generic farmer in the negotiation phase. Based on such scenarios, the agents interact with other agents and participants.

Figure 7 is a screen shot of this simulation. Agents are cultivating the earth and casting seeds. Seeds and plants are drawn using the same color as their owner. The participants can visually see the condition of the farms (how plants grow) and agents' activities. The participants use this visualized information in the negotiation phase. For example, if plants do not grow well in the farm currently being cultivated, the farmer may stop cultivating the farm or lend or sell the farm. If the other farmers can discover the low output of the farm, they may refuse to buy it or set a low price on it. The agents' scenarios can be improved based on the analysis of such participants' behavior.

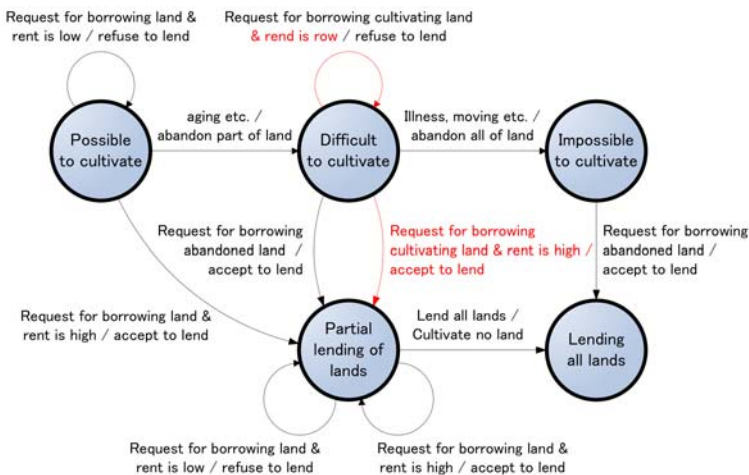


Fig. 6. Behavior Model of a Generic Farmer

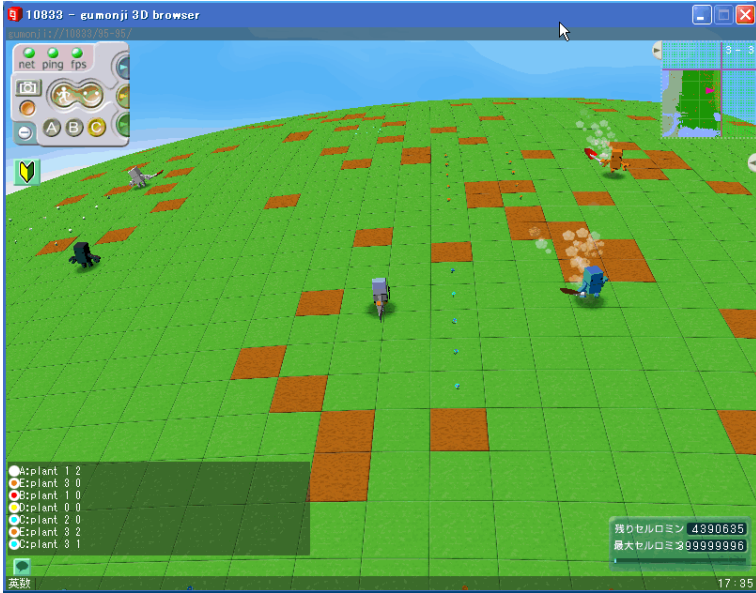


Fig. 7. Executing example scenario

## 6 Conclusion

In this paper, we detailed our implementation of the participatory simulation environment `gumonji/Q`, which is a platform that integrates the scenario description language  $Q$  and the network game `gumonji`. By `gumonji/Q`, A PMAS conductor can design interaction patterns between entities using  $Q$ . He/She can also perform simulations in the realistic virtual space provided by `gumonji` where the PMAS participants can interact with the environment and make reasonable decisions.

Our implementation of `gumonji/Q` is characterized by the following points.

**Customizable simulation environment.** Environment in `gumonji` can be customized by plugins.

**Scenario-controlled agents.** PMAS conductor can easily repeat simulation changing agents' behavior without rebuilding the simulator.

We implemented communication sub-components to achieve TCP/IP communication between the parts of the platform, and a scenario translator to convert  $Q$  requests into/ from sequences of operators.  $Q$  Processor controls avatars that exist in `gumonji` using the functions provided for user control of characters, this makes it possible for agents to perform the same actions as the user-controlled characters and for the `gumonji` simulator to deal with human-controlled avatars and  $Q$ -controlled agents in a unified way.

By using `gumonji/Q` as a simulation platform, we can expect that users will become more interested in participating in simulations given the excellent graphics provided by `gumonji`. The network capabilities of `gumonji` allow users to participate in simulations via the Internet.

## References

1. Gilbert, N., Maltby, S., Asakawa, T.: Participatory simulations for developing scenarios in environmental resource management. In: Third workshop on agent-based simulation, pp. 67–72 (2002)
2. Torii, D., Ishida, T., Bousquet, F.: Modeling agents and interactions in agricultural economics. In: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 81–88 (2006)
3. Colella, V., Borovoy, R., Resnick, M.: Participatory simulations: using computational objects to learn about dynamic systems. In: CHI 1998: CHI 98 conference summary on Human factors in computing systems, pp. 9–10 (1998)
4. Murakami, Y., Sugimoto, Y., Ishida, T.: Modeling human behavior for virtual training systems. In: Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005), pp. 127–132 (2005)
5. Ishida, T.: Q: A scenario description language for interactive agents. *Computer* 35(11), 42–47 (2002)
6. Nakanishi, H., Ishida, T.: Freewalk/q: social interaction platform in virtual space. In: VRST 2004: Proceedings of the ACM symposium on Virtual reality software and technology, pp. 97–104 (2004)
7. Bousquet, F., Bakam, I., Proton, H., Page, C.L.: Cormas: Common-pool resources and multi-agent systems. In: Mira, J., Moonis, A., de Pobil, A.P. (eds.) IEA/AIE 1998. LNCS, vol. 1416, pp. 826–837. Springer, Heidelberg (1998)
8. Murakami, Y., Ishida, T., Kawasoe, T., Hishiyama, R.: Scenario description for multi-agent simulation. In: AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 369–376 (2003)
9. Nakanishi, H., Nakazawa, S., Ishida, T., Takanashi, K., Isbister, K.: Can software agents influence human relations?: balance theory in agent-mediated communities. In: AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 717–724 (2003)
10. Ishida, T.: Society-Centered Design for Socially Embedded Multiagent Systems. *Lecture notes in computer science*, pp. 16–29 (2004)