

Master Thesis

**Extending Network Game Environment
for Persistent Participatory Simulation**

Supervisor Professor Toru Ishida

Department of Social Informatics
Graduate School of Informatics
Kyoto University

Shoichi SAWADA

February 5, 2009

Extending Network Game Environment for Persistent Participatory Simulation

Shoichi SAWADA

Abstract

Multi-agent simulation is becoming popular for understanding complex social phenomena and designing social systems. However, it is difficult to construct complete agent models which can decide behaviors automatically in all situations. Participatory multi-agents simulation (PMAS), where humans and agents jointly participate in and interact with each other in virtual space, attracts attentions. PMAS can take into account complex decisions of humans, and is useful for the purpose of agent modeling.

Meanwhile, network games are attract attentions as simulation platforms for social experiments. Virtual worlds are developed on network, and many humans can participate in them. Network games are used as simulation environment which enable us to observe social phenomena caused by interactions between multiple actors.

Changes in the environment and permeation of new systems do not give humans change immediately. Therefore, for the purpose of observing phenomena which are caused by accumulation of interactions between multiple actors, it is necessary to conduct simulations for a long time. In this research, I propose architecture for developing platforms for persistent participatory simulation by connecting multi-agent systems with network games. Persistent participatory simulation means that humans and agents jointly participate in the simulation and conduct simulation continuously for a long time.

In order to develop platforms for persistent participatory simulation by integrating network games and multi-agent systems, there are two issues as follows.

1. Interactions between humans and agents

In PMAS, humans and agents jointly participate in the simulation by operating avatars in virtual spaces. For the purpose of observing phenomena caused by individual interactions, it is necessary to enable

humans and agents to interact with each other.

2. **Switching operations between humans and agents.**

By entrusting operations of avatars to agents, it becomes possible to continue simulations even when humans are not participating in simulations. In order to realize this function, it is necessary to switch controller of avatars between humans and agents.

In order to enable agents to participate in simulations which conducted in the virtual spaces of network game, I integrate network games and multi-agent systems. By processing operations from agents and humans in the same way, I enable both humans and agents to interact with each other without distinguishing between them. I enable humans and agents to operate the same avatars, and develop meta-level control functions to control operations of avatars. By entrusting operations of avatars to agents, it becomes possible to continue simulation even when humans are not participating in the simulation. In this way, I realize platforms which resolve above mentioned two issues.

I implemented a platform for persistent participatory simulation gumonji/Q by integrating network game gumonji and scenario description language Q. The gumonji is a network game which has functions of environmental simulation. Q is an interaction design language which allows us to define interaction models. By integrating the gumonji and Q, I enable agents to operate avatars on virtual space of the gumonji from outside. I extended meta-level control functions of Q and enabled switching controllers of avatars dynamically between humans and agents.

I conducted a simulation experiment about sustainable use of natural resources using gumonji/Q, for the purpose of verifying availability of persistent participatory simulation. In the experiment, it was possible to observe spread of activities from individual to many actors include humans and agents by interactions between them. It was also possible to observe gradual changes of humans' behaviors according to the changes of surrounding environment during a long time simulation. Thus, a platform for persistent participatory simulation, which is useful for reproducing complex social phenomena and designing social systems, was realized.

永続的な参加型シミュレーションのための ネットワークゲームの拡張

澤田 祥一

内容梗概

マルチエージェントシミュレーションは、複雑な社会現象の理解や、新たな社会システムのデザインのための手段として有用である。しかし、ソフトウェアエージェントのみによるシミュレーションでは、複雑な社会現象を再現することは困難である。そこで、人間とソフトウェアエージェントが仮想空間上でインタラクションをしてシミュレーションを進める、参加型シミュレーションが注目されている。人間が参加することで、より現実的なシミュレーションの実現が期待できる。また、現実的な人間の振る舞いの情報を獲得し、エージェントモデルの構築に活かす、参加型モデリングの手段としても有用である。

一方、社会実験のための環境として、ネットワークゲームが注目されている。ネットワーク上に、現実世界を模擬した大規模な仮想空間が構築されており、多数の人間が参加してインタラクションすることができる。そのため、多数の主体の相互作用による社会現象を観察するためのシミュレーション環境としての利用が広がっている。

環境の変化や新しいシステムの社会への浸透などは、人間に即時的な変化を与えるものではなく、長時間かけて徐々に人間の活動に変化をもたらす。従って、大規模な社会システムの分析のためには、多数の主体が参加したシミュレーションを長期間継続して実施するための枠組みが必要となる。そこで本研究では、ネットワークゲームをエージェントシステムと結合して拡張することで、永続的な参加型シミュレーションのためのプラットフォームを構築する。永続的な参加型シミュレーションとは、人間とエージェントが参加し、長期間継続して実施するシミュレーションである。

ネットワークゲームとエージェントシステムを結合することによって永続的な参加型シミュレーションのためのプラットフォームを構築するためには、以下の2つの課題を解決する必要がある。

1. 人間とエージェントのインタラクション

参加型シミュレーションには、人間が操作したアバタと、エージェントモデルに従って自律的に振舞うエージェントが参加する。従って、主体の相互作用によって現象を再現するためには、人間とエージェントの間でのイ

インタラクションを可能にする必要がある。

2. 人間とエージェントの操作の切り替え

人間が操作しない間のアバタの操作をエージェントに委託することにより、人間が参加していない間のシミュレーションの継続を可能とする。そのため、アバタの操作を人間とエージェントの間で切り替える機能が必要となる。

エージェントがネットワークゲームの仮想空間上で実施されるシミュレーションに参加するため、ネットワークゲームとエージェントシステムを外付けで結合する。その際、エージェントによる動作要求と人間による操作入力を同一の過程で処理するように設計することにより、人間とエージェントが同様の方法でインタラクションを行うことを可能にする。また、同一のアバタに対して人間とエージェントが操作を行うことを可能にし、メタレベルでアバタの操作を制御する機能を設ける。これにより、人間が操作していない間のアバタの操作をエージェントに委託することで、人間が参加していない間でもシミュレーションを継続することを可能にする。以上により、上記の 2 つの課題を解決した、永続的な参加型シミュレーションのためのプラットフォームを実現する。

永続的な参加型シミュレーションのためのプラットフォームとして、ネットワークゲーム **gumonji** とシナリオ記述言語 **Q** を結合して、**gumonji/Q** を実装した。**gumonji** は、環境シミュレーション機能を持つネットワークゲームであり、人間は 3 次元仮想空間上のアバタを操作することによって他者や環境とインタラクションする。**Q** は、エージェントのインタラクションを設計するための言語であり、インタラクションの手順を拡張状態遷移機械モデルで記述する。**gumonji** と **Q** を結合することで、**gumonji** 上のアバタを外部からエージェント化した。そして、**Q** のメタレベル制御機能により人間の操作とシナリオによるエージェントの動作を動的に切り替えることを可能にした。

永続的な参加型シミュレーションを検証するため、**gumonji/Q** を用いて環境問題を扱ったシミュレーション実験を行った。実験では、個人の活動が徐々に周りに影響を与え、人間とエージェントとのインタラクションにより活動が広まっていく様子が観察された。また、長期間のシミュレーションにより、環境の変化によって徐々に人間の行動に変化が現れることが観察された。これにより、複雑な社会現象の再現や社会システムのデザインのために有用な、永続的な参加型シミュレーションのための環境が実現されたと言える。

Extending Network Game Environment for Persistent Participatory Simulation

Contents

Chapter 1 Introduction	1
Chapter 2 Background	8
2.1 Participatory Multi-Agent Simulation	8
2.2 Related Works	8
Chapter 3 Platform for Persistent Participatory Simulation	11
3.1 Persistent Participatory Simulation.....	11
3.2 Integrating Network Game & Multi-Agent System	12
3.3 Agent participation	17
3.4 Switching Controller	19
Chapter 4 gumonji/Q	25
4.1 gumonji & Q	25
4.1.1 Network Game gumonji	25
4.1.2 Scenario Description Language Q.....	27
4.2 Integrating gumonji & Q.....	28
4.3 Avatar Control by Agent.....	31
4.4 Meta-Level Control	36
Chapter 5 Experiment	39
5.1 Overview of Experiment	39
5.2 Result.....	41
5.3 Discussion.....	43
Chapter 6 Summary	46
Acknowledgments	49
References	50
Appendix:	1
A.1 How to Use gumonji/Q.....	1

A.2 Action Reference	9
A.3 Cue Reference.....	16

Chapter 1 Introduction

Social simulations are becoming popular for understanding complex social phenomena and designing social systems. Multi-agent simulation is a promising approach for social simulations, which are based on decisions of humans [Drogoul94, Axelrod97]. In multi-agent simulation, behavior models of individuals and groups are described as agent models. Multi-agent simulation reproduces whole behavior by accumulation of individual interactions. Multi-agent system can reproduce complex systems such as human society based on a bottom-up approach, and we can observe and analyze behaviors of individuals and whole phenomena caused by accumulation of individual interactions [Deguchi98]. However, it is difficult to construct complete agent models which can decide behaviors automatically in all situations [Gunderson99]. Therefore, it is difficult to reproduce complex social phenomena by simulations which consist only of software agents.

Participatory multi-agent simulation (PMAS), where humans participate in multi-agent simulation, attracts attentions. In PMAS, humans and agents jointly participate and interact with each other in virtual space. Thanks to human participation, PMAS can take into account complex decisions of humans and compensate incompleteness of agent models. Moreover, PMAS is useful for the purpose of agent modeling, where we can observe practical behaviors of humans in virtual environment and construct models from observed behaviors [Colella00, Bousquet02]. In addition, PMAS can be used as a tool for educational simulations where participants learn knowledge and techniques by experience in virtual environment [Rickel99, Prensky03].

In previous researches, PMAS was used for analyzing individual phenomenon under the particular situation. In Modeling Human Behavior for Virtual Training Systems [Murakami05], simulation was conducted for the purpose of constructing evacuation models by observing behaviors of human participants who evacuate from a building in virtual space. Participants operated avatars in 3D virtual space and evacuated from a building. In this experiment, simulation was conducted in narrow space and short period. In

Layering Social Interaction on Environmental Simulation [Torii04], CORMAS [Bousquet98] was used as a platform for reproducing interactions of fire fighters. In this simulation, interactions between fire fighters and leader while they are extinguishing forest fire are reproduced. Thus, if target of observation and analysis is particular behavior or phenomenon, it is possible to construct a simulation environment where particular situation is reproduced.

Meanwhile, network games, such as Second Life and World of Warcraft, attract attentions as simulation platforms for social experiments [Bainbridge07]. By the progress of information technologies, virtual spaces which reproduce world-like environment are developed on network. In networked environment, large-scale virtual spaces are developed, and many humans can participate in virtual spaces. Behaviors of humans are reproduced as movements of avatars on the virtual spaces. Humans can interact with avatars which are operated by other humans and virtual objects. Virtual environment changes continuously by interactions between avatars.

In the fields of social, behavioral, or economic sciences, use of network games are spreading as simulation platforms for analyzing cooperating behaviors of humans or economic markets. Experiments using Second Life include sensor-based tracking simulation [Brandherm08] and social interaction simulation [Rehm08]. Advantages of using network games as simulation platforms are following four points. (1) We can construct large scale experimental environments. (2) Multiple and various humans can participate in simulations. (3) We can conduct simulations for a long time. (4) It does not require special equipment. Thus, simulation environments, where many humans can participate in and social phenomena caused by interactions between multiple actors are reproduced, are required.

Network games have been used for simulation environments for social experiments as they are, in past research. Network games are used as platforms for gaming simulations where humans participate in. Actors in virtual environment consist of only avatars operated by humans. Therefore, in order to conduct large scale simulation, it is necessary to assemble many human subjects. In network games, time of virtual environment always progresses, and

humans can participate in the environment at any time. Accordingly, humans do not participate always in simulations, and number of participants varies from hour to hour. When few humans participate in simulations, interactions between them decrease. Therefore, the period is limited, when complex social phenomena caused by interactions between multiple actors can be observed. So, it is difficult to continue simulations for a long time if network games are used as simulation platforms without modification.

For the purpose of observing phenomena caused by accumulation of interactions between multiple actors, it is necessary to conduct simulations for a long time. For example, if new products such as mobile phones, which give impact to human activities, are released, not all humans begin to use them immediately. Number of owner increases gradually by advertisement of companies and word-of-mouth of many people. Moreover, spreading of new products could make an impact on whole social system. Thus, changes in the environment and permeation of new systems do not give humans change immediately. Activities of humans change little by little over the long time. Therefore, frameworks which enable us to assemble many human objects and conduct simulations continuously are needed.

In this research, I propose architecture for developing platforms for persistent participatory simulation by integrating network games and multi-agent systems. Persistent participatory simulation is a method which humans and agents jointly participate in and conduct simulation continuously for a long time. It becomes possible to continue simulations for a long time by entrusting operations of avatars to agents while humans do not participate. Participation both of humans and agents enables to conduct simulations where many subjects participate in and interact with each other.

In order to develop platforms for persistent participatory simulation by integrating network games and multi-agent systems, there are two issues as follows.

1. Interactions between humans and agents

In PMAS, humans who operate avatars and agents who behave autonomously according to agent models jointly participate in the

simulation. For the purpose of observing phenomena caused by individual interactions, it is necessary to enable humans and agents to interact with each other.

2. Switching operations between humans and agents.

By entrusting operations of avatars to agents, it becomes possible to continue simulations even when humans are not participating in simulations. In order to realize this function, it is necessary to switch controller of avatars between humans and agents.

In order to enable agents to participate in simulations which conducted in the virtual spaces of network games, I integrate network games and multi-agent systems. Agents decide actions according to the acquired information by observing surrounding environment. In order to realize autonomous actions of agents, I develop functions which enable agents to observe surrounding environment and operate actions of avatars. Agents participate in simulations by requesting actions of avatars from outside. I develop functions to handle requests from agents and operations from humans in the same process. In this way, I give agents the ability to operate avatars as same as humans and interact with both humans and agents in the same way.

In order to enable switching controller of avatars between humans and agents, I design avatar control functions where humans and agents can operate the same avatars. By switching controllers of avatars, human participants can entrust operations of avatars to agents, when they are not operating avatars. In this way, it becomes possible to continue simulations even when humans are not participating in the simulation, thanks to autonomous operations by agents substitute for humans.

For the purpose of controlling operations from humans and agents, I develop meta-level control functions. Meta-level control enables humans to switch controller of avatars and entrust operations to agents at any time. I also enable humans to take back control of avatars at any time even when agents are executing actions of avatars. On the other hand, I enable agents to switch controllers of avatars autonomously. While agents are observing operations of humans, if humans are not operating avatars, agents switch controllers of

avatars and start operating avatars autonomously. While agents are operating avatars, if agents become impossible to decide behaviors autonomously, agents request humans to operate avatars. In this way, it becomes possible to switching controllers of avatars dynamically according to the situation.

I implement a platform for persistent participatory simulation gumonji/Q. I developed gumonji/Q by integrating network game gumonji and scenario description language Q [Ishida02].

The gumonji is a network game, which has functions of environmental simulation. In gumonji, animals and plants exist in a virtual space, and the atmosphere and water circulate according to physical law. Humans participate in the simulation environment by operating avatars in the virtual space. Humans can interact with avatars operated by other human participants through actions and chats of avatars. Humans also can manipulate living things and landform in virtual space through the actions of avatars. The influences given to the living things and landscape spread widely, and the environmental changes are displayed in 3D virtual space. Many humans can participate in the same virtual space of gumonji through the Internet.

Scenario description language Q is an interaction design language for multiagent systems. By using Q, it is possible to define how agents are expected to interact with its environment involving humans and other agents. Q has functions to control execution of scenarios in meta-level. By controlling scenario executions in meta-level, it is possible to observe states of agents and changes of simulation environment, and control scenario executions according to the situation.

By integrating the gumonji and Q, agents become possible to operate avatars from outside. In the process of communicating between the gumonji and Q, requests of operating avatars from agents are converted into operators which activate actions of avatars, and results of executing actions are converted into the format which agents can deal with. In this way, agents can acquire the situation of simulation environment and execute actions of avatars according to the situation. In addition, requests from agents and operations from humans are converted into the same format and processed in the same way. By doing so,

it becomes possible to process operations without distinguishing whether they are from humans or agents. In this way, agents become able to operate avatars substitute for humans.

I use meta-level control functions of Q, for the purpose of switching controllers of avatars. Meta-level control functions enable observing scenario execution in meta-level and switching scenarios according to the situation. I extend these functions and enable to observe and control operations from humans. By doing so, it becomes possible to observe behaviors both of agents and humans, and switch controllers of avatars between humans and agents.

I conduct a simulation experiment about sustainable use of natural resources using gumonji/Q, for the purpose of verifying availability of persistent participatory simulation. I conduct simulation continuously for one week. In the experiment, I observe spread of activities from individual to many actors include humans and agents by interactions between them. I also observe gradual changes of humans' behaviors according to the changes of surrounding environment during a long time simulation. By comparing behavior of many humans while they are interacting with other humans and agents, I take out the characteristic behaviors of individuals. By observing activities for a long time, I verify gradual changes of behavior patterns of humans. By observing interactions between multiple humans, I verify spread of activities from human to human. In this way, I evaluate functions of persistent participatory simulations.

By conducting simulation using network games, humans can easily participate in the simulation. This can be considered as another advantage of using network games. By enabling many humans' participation, it becomes possible to conduct larger scale simulations.

In addition, by switching operations at any time, it is possible to entrust operations to avatars arbitrarily such as when humans judge that they do not have to operate directly. Since agents can automatically request humans to operate avatars, agents can switch controller such as when they can not decide actions autonomously or they need decisions of humans. In this way, it becomes possible to request humans to participate at the important point in

the simulation. If simulator skip process of operation and display, when humans are not participating in the simulation, it becomes possible to reduce computational load.

The reminder of the paper is organized as follows. First, I show the background of this research and some related works in chapter 2. Second, I present an approach to develop a platform for persistent participatory simulation in chapter 3. Then, I introduce gumonji/Q, which is implemented according to my approach in chapter 4. In chapter 5, I evaluate my approach by conducting simulation experiment on gumonji/Q. Finally, I conclude in chapter 6.

Chapter 2 Background

In this chapter, I describe about participatory multi-agent simulation (PMAS), as a background of this research. Then, I pick up some related works about PMAS and consider issues for conducting social simulations and reproducing human behaviors.

2.1 Participatory Multi-Agent Simulation

Participatory multi-agents simulation (PMAS), is a form of multi-agent simulation. In PMAS, humans and agents jointly participate in simulations and interact with each other in virtual space. In PMAS, whole phenomena are reproduced by accumulation of individual interactions. Thanks to humans' participation, it is possible to take into account complex decisions of humans and compensate incompleteness of agent models. Moreover, PMAS is useful for the purpose of agent modeling, where we observe practical behaviors of humans on virtual environments and construct models from observed behaviors [Colella00, Bousquet02]. In addition, PMAS can be used as a tool for educational simulations where participants learn knowledge and techniques by experiences in virtual environment [Rickel99, Prensky03].

2.2 Related Works

FreeWalk [Nakanishi04] is a virtual urban space simulator, which has functions of displaying 3D virtual space. In FreeWalk, agents are controlled by scenarios, and humans can participate in the virtual space by operating avatars. The primary objectives of FreeWalk are to realize a virtual space for communication and collaboration between humans and agents. FreeWalk has functions for interactions between humans and agents, but does not simulate the environment in detail. Humans can not affect surrounding environment and receive feedback. Therefore the use of FreeWalk as a participatory simulation platform is limited.

FreeWalk is used as a participatory simulation platform in the research of Modeling Human Behavior for Virtual Training Systems [Murakami05]. Humans participated in the simulation by operating avatars in a building.

Simulation was conducted for the purpose of constructing evacuation models by observing behaviors of human participants who evacuate from a building in virtual space. Participants operated avatars on 3D virtual space and evacuated from a building. In this experiment, simulation was conducted in narrow space and short period. In contrast, I aimed to develop simulation platforms not for reproducing particular situation but for reproducing change of social systems and transition of behavior pattern of humans in long term.

CORMAS [Bousquet98] is a platform for simulating interactions and cooperation between individuals and groups who jointly exploit the resources. In CORMAS, users can define the diffusion of environmental changes, and agents' behaviors followed by the surrounding environment. The computational model of CORMAS is a cellular automaton. Natural environment is modeled as a two dimensional mesh, and diffusion between neighboring cells is calculated at each unit time. CORMAS is useful to reproduce interactions between natural environment and humans. However, CORMAS shows simulation environment in abstract graphics, so that it does not provide enough visual information. Therefore, it is hard for humans to behave like in the real world on the environment described in two dimensional meshes.

CORMAS is used in the study of Layering Social Interaction on Environmental Simulation [Torii04]. CORMAS was used as a platform for reproducing interactions of fire fighters. In this experiment, interactions between fire fighters and a leader when they are extinguishing forest fire was reproduced. Fire fighters observed situation of forest fire in the two dimensional map in CORMAS and selected operations. Objective of this experiment was analyzing interactions between fire fighters and a leader and decisions of fire fighters. Therefore, it was enough to conduct simulations where situations of simulation environment were displayed in abstract graphics. However, when it is not clear where humans' characteristic behaviors appear, it is impossible to observe humans' practical behaviors in such simulation.

SOARS [Deguchi04] is a modeling language for agent-based social simulations. In SOARS, autonomous agents, who perform complex social

activities according to individual rules, interact with each other. A kind of complex systems is constructed by interactions such as cooperation, opposition and asymptomatic behaviors. Models are constructed by combination of commands which represent abstracted behavior rules of humans. Result of simulation is displayed in animations on two-dimensional space. Humans can participate in simulations as decision-making actors by imputing numeric or characters.

Chapter 3 Platform for Persistent

Participatory Simulation

3.1 Persistent Participatory Simulation

In PMAS, humans and software-agents jointly participate in the simulation environment built in virtual space. Behaviors of humans are reproduced as movements of avatars in virtual space. Simulation progresses by interactions among human controlled avatars, agents and virtual objects. Humans participate in the simulation for the purpose of taking into account humans' complicated decision-making processes. Observed behaviors of humans are also used for building agent models. Because decision-making processes of humans are continuous, it is not clear where and when characteristic decisions and behaviors appear. Therefore simulation platforms for observing decision-making of humans are required for providing the environment where sequence of humans' behaviors can be observed.

Changes in the environment and permeation of new systems do not give humans change immediately. Activities of humans change little by little over the long time. For example, when cellular phones, which would give significant changes in behavior patterns of humans, are released, they will not spread over people immediately. Number of owners increases gradually by advertisement of companies and word-of-mouth of many people. Moreover, spreading of new products could make an impact on whole social system. Therefore, for the purpose of observing influences of introducing new systems and changes in human behavior patterns, simulation has to be conducted for a long term.

In this research, I proposed architecture for developing platforms for persistent participatory simulation by integrating network games and multi-agent systems. Persistent participatory simulation means that humans and agents jointly participate in and conduct simulation continuously for a long time. It makes it possible to continue simulations for a long time by entrusting operations of avatars to agents while humans do not participate in.

Participation both of humans and agents enables to conduct simulations where many subjects participate in and interact with each other.

I describe architecture for developing platforms for persistent participatory simulation in following sections.

3.2 Integrating Network Game & Multi-Agent System

I realized a platform for persistent participatory simulation by integrating a network game and a multi-agent system.

The advantage of using network games for persistent participatory simulation is as follows.

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

Humans can observe virtual space and operate avatars visually in network games. So humans can act practically in the virtual space.

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

Network games provide virtual environment where many humans can participate in. 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.**

Many and various humans can participate in the environment built in virtual space thanks to participating via the Internet.

- **The virtual environment progresses persistently.**

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

However, by using network games without modifying, persistent participatory simulation can not be achieved. It is assumed that humans operate avatars and interact with other human participants in network games. So, avatars perform actions only when operated by humans. While humans do not participate, avatars do not act at all. Therefore, if a human get out of the

virtual environment, other humans who are interacting with him would lose a partner and it would become difficult to continue activity performed before. For the purpose of realizing persistent participatory simulation, a framework is needed which enables continuing simulation even while humans do not participate.

I extended a network game by connecting with a multi-agent system. Figure 1 shows how to realize a participatory simulation by integrating a network game and a multi-agent system. Right side of the figure shows the part of network game and left side shows the part of multi-agent system.

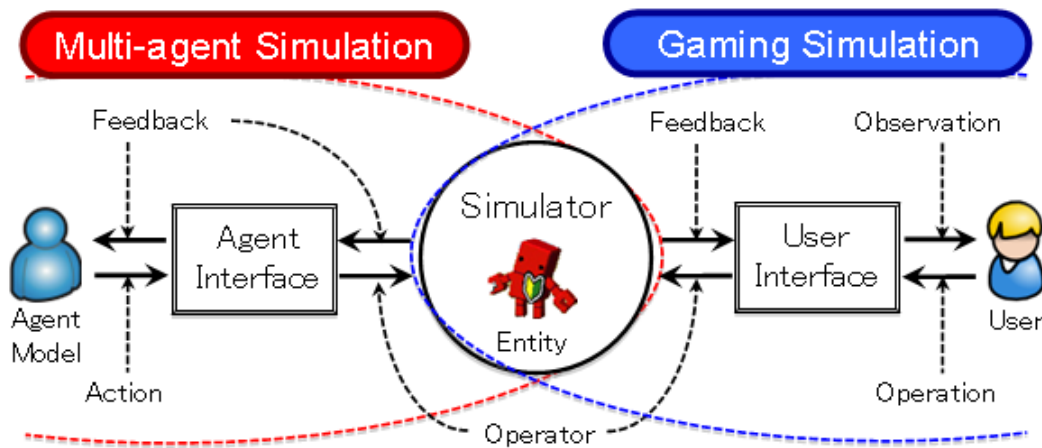


Figure 1: Participatory simulation with network game and multi-agent system

It can be said that a network game is a platform for gaming simulation where humans act in the virtual space. On the other hand, in a multi-agent system, agents act in the virtual environment following described agent models. By integrating a network game and a multi-agent system, I realize a platform where humans and agents jointly participate in the simulation.

Humans participate in the simulation performed in the virtual space through user interface. User interface has the input function which conveys operations by humans to simulator and the output function which conveys the situation of virtual environment to humans. Humans input operations using

input devices such as mouse and keyboard. Input operations are converted into the operators which activate actions of avatars, and the operators are conveyed to the simulator. Result of the actions and changes of the simulation environment which outputted from simulator are converted into graphics and sounds, and conveyed to humans through display and speaker.

On the other hand, agents participate in the simulation through agent interface. Agent interface has the input function which conveys requests by agents to simulator and output function which conveys the situation of virtual environment to agents. Agents decide actions according to the surrounding circumstances and send requests for actions. Requests are converted into the operators which activate actions of avatars, and the operators are conveyed to the simulator. Results of the actions and changes of the simulation environment which outputted from simulator are converted into the format available to agents and conveyed to agents.

In this research, I developed a platform for persistent participatory simulation by extending a network game connecting with a multi-agent system. A network game consists of the server part which computes simulation environment, and the client part which becomes the user interface for humans' participation. I used the client as it is, and use the server as a simulator which computes simulation environment where both humans and agents participate in. By connecting a multi-agent system to the server of a network game, I enabled agents to participate in the virtual environment of a network game.

Figure 2 shows architecture for integrating a network game and a multi-agent system. Upper part of the figure shows the server of a network game, lower right part shows the client of a network game, and lower left part shows a multi-agent system. A multi-agent system is connected to the server in the same way as the client.

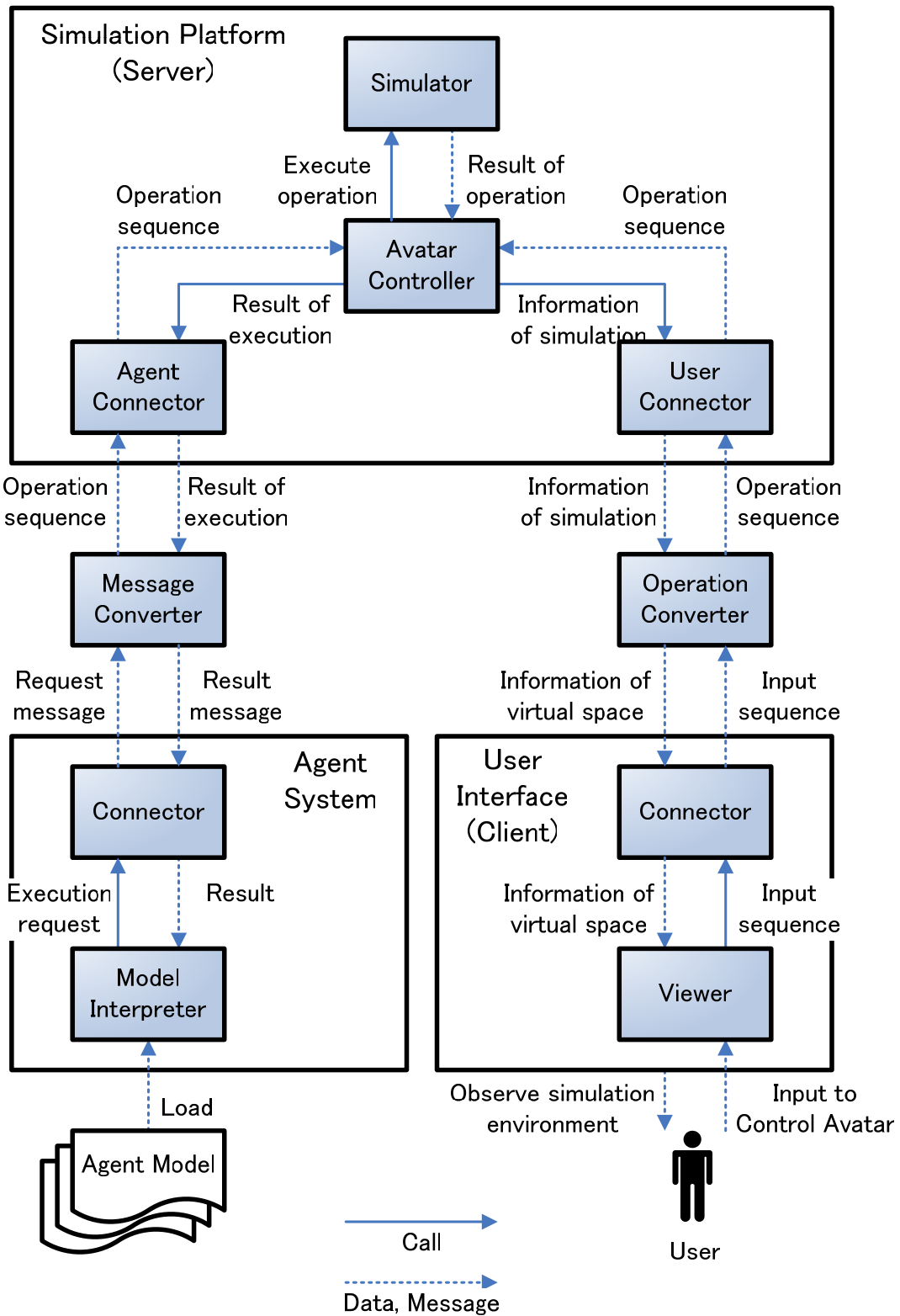


Figure 2: Integrating network game and multi-agent system

Humans participate in a network game by operating avatars in a virtual space. Human participants operate avatars through user interfaces such as keyboard and mouse. Operations input on the user interfaces are converted into the operators to make avatars move in the virtual space and conveyed to the simulator. The simulator outputs the results of operations and the information of the virtual space. Outputted information is converted into graphics and sounds which humans can understand, and displayed on the viewer. Humans can observe situation of virtual space and get feedbacks of actions by the information output through a display and a speaker.

When agents operate avatars, operations from agents are handled in the same process as operations from humans. Request messages from agents are sent to Simulator after converting into operators which activate actions of avatars in virtual space at Message Converter. In this process, request messages are converted into as same format as operations from humans. By doing so, Simulator receives operators which are converted in advance, and Simulator can handle operators without distinguishing operations from humans or agents. In other words, Simulator can handle operations from both humans and agents in the same way. Simulator returns feedbacks of actions and information of virtual space to agents. In this time, information and feedbacks are converted into format which agents can handle, and sent to agents by Message Converter. In above process, agents can receive information of virtual space and feedbacks of actions, and continue executing next actions according to the received information and feedbacks.

In this architecture, operations input from humans and requested by agents are handled in the same process. By doing so, agents have the same abilities for operating avatars as humans. Therefore, agents can participate in simulation substitute for humans. Moreover, Simulator can handle both avatars operated by humans and agents in the same way, so that participants can interact with avatars without distinguishing whether humans or agents are operating avatars.

3.3 Agent participation

Agents participate in the simulation by requesting actions of avatars from outside. Agents observe surrounding environment and decide actions according to the obtained information. In order to realize autonomous behavior of agents, agents have to be able to observe surrounding environment and control actions of avatars.

Figure 3 shows the flow of the processing observations and actions of agents from outside. Simulator and Agent System communicate with each other through the Message Converter. Message Converter absorbs the difference of the information format between Simulator and Agent System. Observation of environment and execution of actions are handled in the same process. Agents request observations by specifying the situation they want to know. Then, Simulator returns the information of virtual environment which match the requested situation. Agents also request actions by specifying the movement of avatars. Then, actions of avatars are executed in the virtual space and feedbacks of the actions are returned.

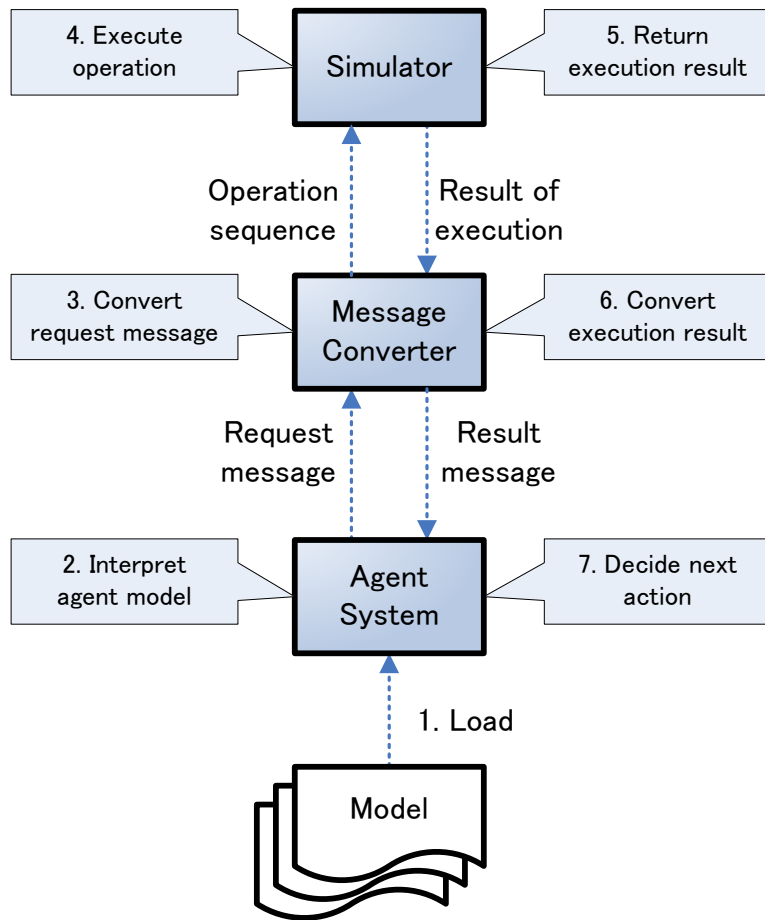


Figure 3: Operation process of agents

I explain the process of observing environment following the figure. First, 1) agents send messages for requesting the observation of environment by specifying the condition of environment they want to know. 2) The messages sent from agents are converted into the operators for executing observations on the virtual space through the Message Converter. 3) When simulator receives the operators of observation, execute observing virtual environment around the avatars. Then, 4) Simulator returns the information of virtual environment which matches the specified situation. 5) The results of observations are converted into the format which agents can handle through the Message Converter, and sent to Agent System. 6) Agents receive the information of surrounding virtual environment of avatars as results of observations. In above

process, Agents can acquire information of surrounding environment which is necessary for behaving autonomously.

Next, I explain the process of executing actions. First, 1) agents send messages for requesting the actions of avatars by specifying the movements in the virtual space. 2) The messages sent from agents are converted into the operators for executing actions of avatars through the Message Converter. 3) When simulator receives the operators of actions, execute actions of avatars in virtual space. Then, 4) Simulator returns the results of actions as feedbacks to the agents. 5) The results of actions are converted into the format which agents can handle through the Message Converter, and sent to Agent System. 6) Agents receive the results of actions as the feedbacks of operating surrounding environment. In above process, Agents can operate avatars in the virtual space and execute the actions of avatars which make changes of surrounding environment.

Agents can operate avatars autonomously from outside by repeating process of observations and actions. In this way, I realized autonomous behavior of agents from outside.

3.4 Switching Controller

It is difficult to continue conducting simulations for a long time with all human subjects participating. Therefore, it is necessary to continue simulations even while part or all of human subjects are not participating in. So, I developed functions to switch controllers of avatars between humans and agents. By switching controllers of avatars from humans to agents and entrusting control of avatars to agents, it becomes possible to continue activities of avatars even while human subjects are not operating avatars. In this way, it becomes possible to continue simulations persistently regardless of humans' participation.

In order to entrust control of avatars to agents, the platform have to enable humans and agents to operate the same avatar and enable to switch controllers of avatars dynamically between humans and agents. So, I developed three layer avatar control architecture. The three-layer structure consists of

Decision-Making Layer, Control Layer, and Simulation Layer. This architecture enables multiple actors to control the same avatars.

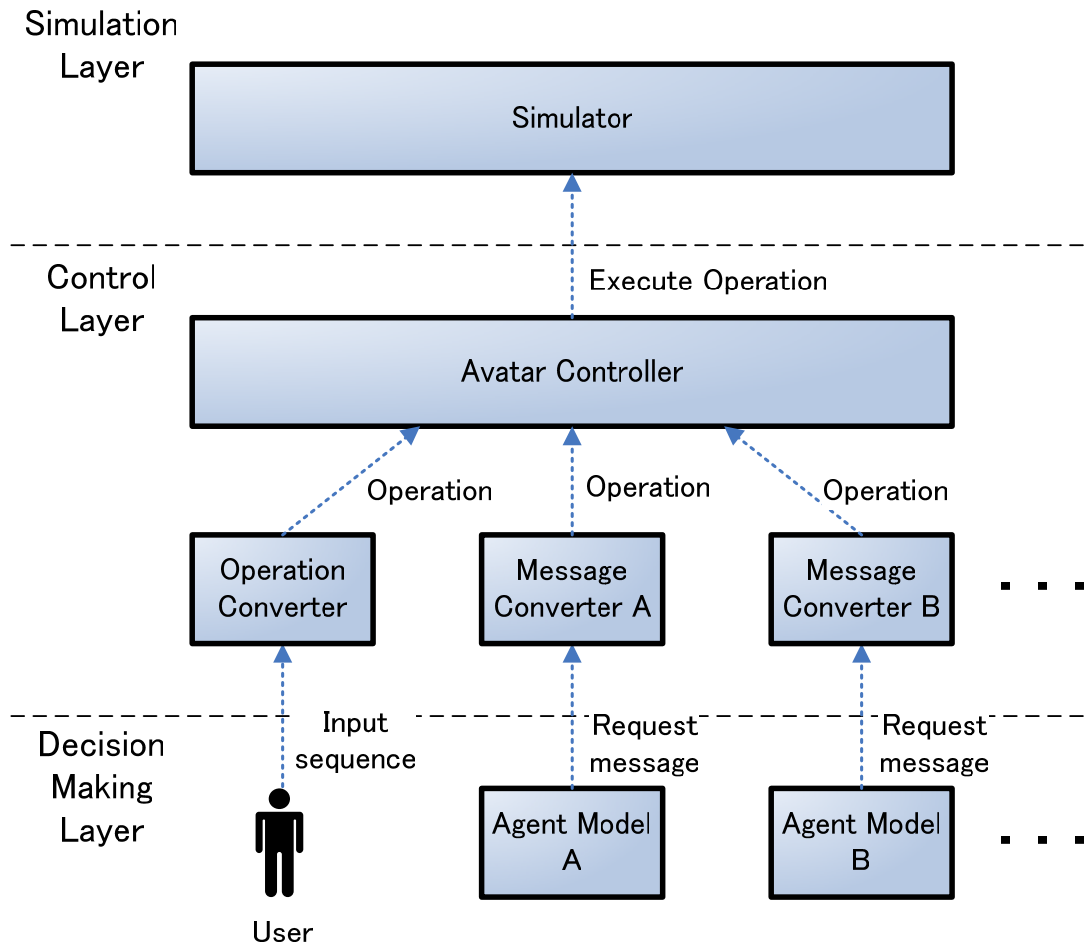


Figure 4: Three-layered avatar control

Figure 4 shows the three-layer structure of avatar control. Decision-Making Layer processes decisions of operating avatars. Control Layer processes requests of operating avatars. And Simulation Layer processes executions of operating avatars.

Decision-Making Layer has functions to process decisions of operating avatars. Decision-Making Layer includes actors who autonomously decide operations of avatars. Decision-making actors include human participants and

autonomous agents who decide behaviors according to described various agent models. Both humans and agents decide behaviors according to individual decision-making model and request actions of avatars. Decisions of humans are input from user interface. Input sequences of operating avatars are sent to Operation Converter. Decisions of agents are input as request messages. Request messages for operating avatars are sent to Message Converter specific to agent models.

Control Layer has functions to process requests of operating avatars. In this layer, requests of operations sent from Decision-Making Layer are converted into operators which activate actions of avatars. Operation sequences input from humans and request messages sent from agents are converted into operators. In this process, humans' inputs and agents' requests are converted into the same format of operators. All operators sent from individual decision-making actors are gathered in Avatar Controller. Then, Avatar Controller sends operators to Simulator for executing actions of avatars.

Simulation Layer has functions to process executions of operating avatars. Simulator executes operators sent from Avatar Controller which activate avatars actions. Since all operators requested from multiple actors are sent via Avatar Controller, Simulator does not have to distinguish whether operators are sent from humans or agents.

This three-layer structure of avatar control wraps the difference of decision-making actors. Decision-Making Layer is wrapped by Control Layer. So, Simulation Layer does not have to distinguish decision-making actors. In other words, Simulator can execute operators for activating actions of avatars without distinguishing whether operators are sent from humans or agents. In this way, both humans and agents can operate the same avatars.

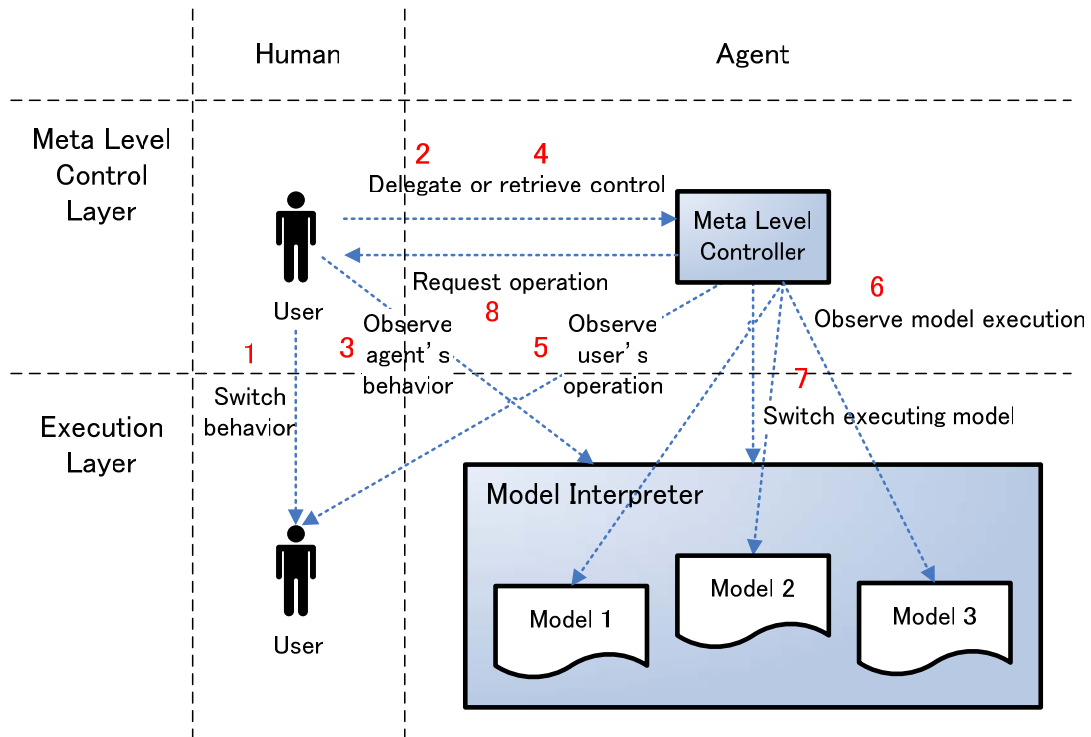


Figure 5: Meta-level operation control

Figure 5 shows meta-level control framework for controlling operations of avatars. This meta-level control framework consists of two layers, which are Meta-Level Control Layer and Execution Layer. This framework enables dynamically switching controllers of avatars. Meta-Level Control Layer observes actions of avatars and surrounding environment, and controls operations of avatars. Execution Layer executes actions of avatars in the virtual space according to individual decision-making models.

Humans' behaviors can be divided meta-level consideration and actual actions. Humans consider behavior at the big picture according to time and circumstance, and perform actions to interact with surrounding environment and other actors. In a simulation, humans consider how to behave according to the circumstance of virtual environment at meta-level. And then humans operate actions of avatars in the virtual space.

Behaviors of Agents can be separated into Meta-Level Control Layer and Execution Layer, in the same way as humans. In Meta-Level Control Layer,

agents observe surrounding environment and switch agent models suitable for the circumstance. In Execution Layer, agents execute actions to interact with surrounding environment.

Execution Layer includes humans who operate avatars and Model Interpreter which interpret agent models and decide actions according to the model. Humans observe surrounding virtual environment of avatars and decide operations to activate actions of avatars. On the other hand, agents observe surrounding virtual environment and decide actions according to the agent model.

Meta-Level Control Layer includes humans who switch how to operate avatars according to the circumstance, and Meta-Level Controller which controls executions of agent models.

On the human side, humans observe the environment of virtual space and consider the behavior according to the circumstance. 1) When humans are directly operating avatars, observe virtual space around avatars and switch behaviors suitable for the circumstance. 2) If humans judge that they do not have to operate avatars by themselves, they can entrust operations to agents. 3) When agents are operating avatars, humans observe behaviors of avatars. If humans judge that behaviors of avatars are improper, they can direct agents to switch executing models. 4) If humans judge that they should operate avatars directly, they can stop operations of agents and take back the control of avatars.

On the agent side, Meta-Level Controller observes environment of virtual space and behaviors of avatars, and control executions of agent models. When humans are operating avatars, Meta-Level Controller observes operations by humans. 5) If Meta-Level Controller observes that avatars are not acting because humans are away from keyboard, it switches operations of avatars to agents. If humans direct to switch control, Meta-Level Controller starts executions of agent models. 6) When agents are operating avatars, Meta-Level Controller observes state of executing agent models and switch executions of agent models depending on the situation. 7) When agents have to switch behaviors according to the changes of situation, switch executing models and start executing other agent models. In the case when it is difficult for agents to

decide actions autonomously, Meta-Level Controller stops executing agent models and requests humans to operate avatars.

Chapter 4 gumonji/Q

I implemented a participatory simulation platform gumonji/Q, which enables conducting simulation persistently according to the architecture I described in previous chapter. In this chapter, I describe the system structure and functions of gumonji/Q for realizing persistent participatory simulation.

4.1 gumonji & Q

I used network game gumonji and scenario description language Q to develop gumonji/Q. In this section, I introduce the gumonji and Q which is the component of gumonji/Q.

4.1.1 Network Game gumonji

The gumonji¹ is a network game developed and released by Community Engine Inc. The characteristics of gumonji are functions of environmental simulation. In the gumonji, animals and plants exist in a virtual space, and the atmosphere and water circulate according to physical law. Humans participate in the simulation environment by operating avatars in the virtual space. Humans can interact with avatars operated by other human participants through actions and chats of avatars. Humans also can manipulate living things and landform in virtual space through the actions of avatars. The influences given to the living things and landform spread widely, and the environmental changes are displayed in 3D virtual space as shown in Figure 6.

The gumonji consists of Zone Server which computes simulation of virtual space and Client which is a user interface for human participants. Zone Server has functions to calculate environmental simulation according to physical law. Zone Server reflects actions of avatars sent from Client in the virtual space. Zone Server calculates environmental change according to the actions of avatars and sends information of environmental changes to Client. Client is a user interface which has functions of operation and display. Humans participate in the virtual space by operating avatars displayed on Client. Input operations by humans are sent from Client to Zone Server and give influences

¹ <http://www.gumonji.net/>

to the virtual environment through the actions of avatars. Client dynamically displays the state of virtual space which sent from Zone Server in real-time.



Figure 6: 3D virtual space of gumonji

In this paper, I used the gumonji for developing a platform for persistent participatory simulation. The gumonji has the functions to enable humans to act in the virtual space and participate at any time, which is necessary for realizing persistent participatory simulation. Human participants can operate avatars in the virtual space and interact with each other using motions and chats of avatars. Humans can log in and log out simulation at anytime, and the environment of virtual space keeps changing regardless of human participants. Further more, the gumonji has functions which enable humans to interact with natural environment in the virtual space and calculate the changes of natural environment. The influences of actions of avatars spread to surrounding environment and give feedback to humans as the change of environment. Thanks to the functions of environmental simulation of the gumonji, it becomes possible to simulate social phenomena caused by interactions between human society and natural environment.

In other words, it is possible to observe bidirectional influences between activities of humans and changes of environment. Activities of humans influences natural environment, and changes of natural environment also influences humans.

However the gumonji is a game where humans operate avatars in the virtual space and interact with each other. So, decision-making actors in the simulation are only humans. In order to activate actions of avatars, humans have to operate all actions of avatars. Therefore, avatars act only when humans are participating in the simulation. When humans log out of simulation, avatars temporally disappear from virtual space. In order to use the gumonji as a platform for persistent participatory simulation, it is necessary that functions which enable to continue simulation even when humans are not participating.

4.1.2 Scenario Description Language Q

Scenario description language Q [Ishida02] is a interaction design language for multiagent systems that allows us to define how agents are expected to interact with its environment involving humans and other agents. Q does not refer to inside mechanism of agents and does not consider how agents are described. Q is designed for connecting with existing simulation platform and controlling agents from outside without changing internal implementation.

In Q, agent models are described in scenarios. The computational model behind a Q scenario is an extended finite state automaton, which is commonly used for describing communication protocols. Q is suitable for describing complex social interactions [Murakami05]. Q scenarios foster the emergence of dialogs between agent designers (computing professionals) and application designers (scenario writers) [Ishida04]. 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 on waiting for the event specified until the 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 scenarios that specify 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.

Q has the functions which controls executions of scenarios in meta-level [Yamane 06]. Models of meta-level control are described as meta-scenarios. Meta-level control functions observe states of scenario executions and changes of simulation environment, and control scenario executions according to the situation. It enables to assign new scenarios and change executing scenarios.

Q has been used by connecting with existing agent systems because it does not have the functions of simulation. Q has been connected with FreeWalk [Nakanishi04], CORMAS [Torii04], MS Agent, and Caribbean [Nakajima06]. In this paper, I developed a platform for persistent participatory simulation by connecting Q with gumonji.

4.2 Integrating gumonji & Q

I integrated the gumonji and Q according to the architecture of developing platforms for persistent participatory simulation by connecting network games and multi-agent systems. Figure 7 shows the system structure of gumonji/Q. Upper part of the figure is gumonji Zone Server, Lower right part is gumonji Client, and lower left part is Q processor.

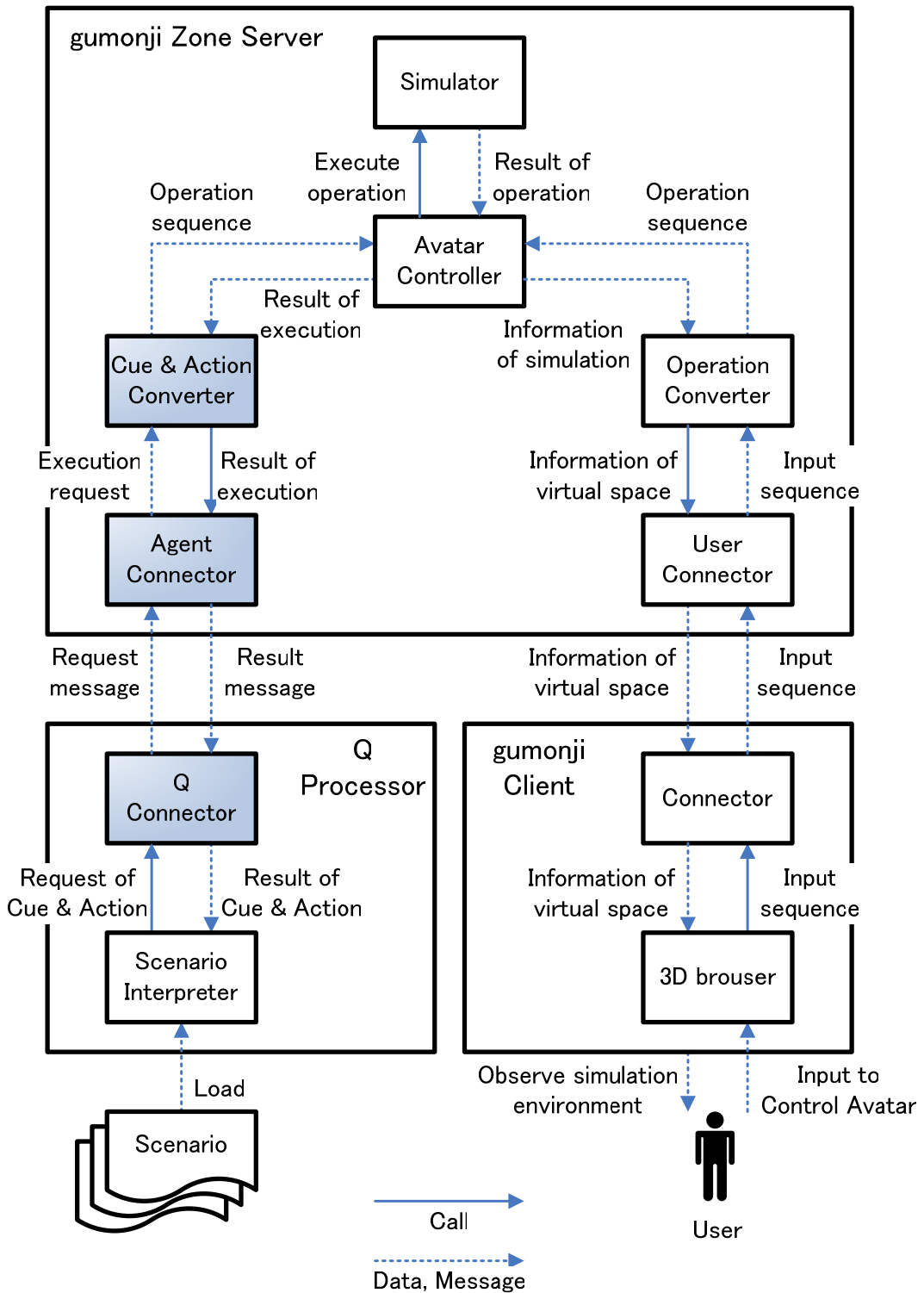


Figure 7: System structure of gumonji/Q

I used gumonji Zone Server as a simulator and gumonji Client as a user interface in gumonji/Q. So, the components newly implemented in this research are colored components in the figure such as Cue & Action Converter, Agent Connector and Q Connector.

I implemented connection between the gumonji and Q according to the architecture described in previous chapter. I implemented gumonji/Q to handle requests from user interface and agent system in the same process. I implemented Cue & Action Converter in Zone Server in the perspective of processing efficiency.

I used TCP/IP for connecting the gumonji and Q. This is because TCP/IP is a legacy protocol and can be used in many programming platforms. Q Processor is executed on JAVA framework and gumonji is implemented in C++. So, connecting functions can be easily added in both the gumonji and Q. Furthermore, by using legacy protocol, it can be easily extended by connecting with other existing agent platforms and enable avatar control by agents described in other agent models.

For connecting the gumonji and Q using TCP/IP, I implemented Q Connector in Q Processor and Agent Connector in gumonji Zone Server. Q Connector in Q Processor has functions to send request messages and receive result messages. Q Connector encodes request messages of cues and actions, and sends them using TCP/IP connection. When Q Connector receives result messages, decodes them and sends to agents. Agent Connector in gumonji Zone Server has functions to receive request messages and send result messages. When Agent Connector receives request messages, decodes them and sends to Cue & Action Converter. And Agent Connector encodes result messages into string and sends to Q Connector.

Cue & Action Converter has functions to convert data format between Q Processor and gumonji Zone Server. Q Processor handles behaviors of agents using cues and actions. Cue & Action Converter has the role of relating cues to observation of virtual space and relating actions to movement of avatars. It converts cues into operators for observing surrounding environment of avatars in virtual space, and convert actions into operators for activating movements of

avatars. I designed Cue & Action Converter to convert actions into the format as same as operations input from humans. By doing so, it becomes possible to handle operations from humans and agents in the same way.

4.3 Avatar Control by Agent

I implemented the functions for processing cues and actions. Agents operate avatars using cues and actions according to the described agent models. Figure 8 shows the process of handling cues and actions. I explain the process with example of executing cues and actions.

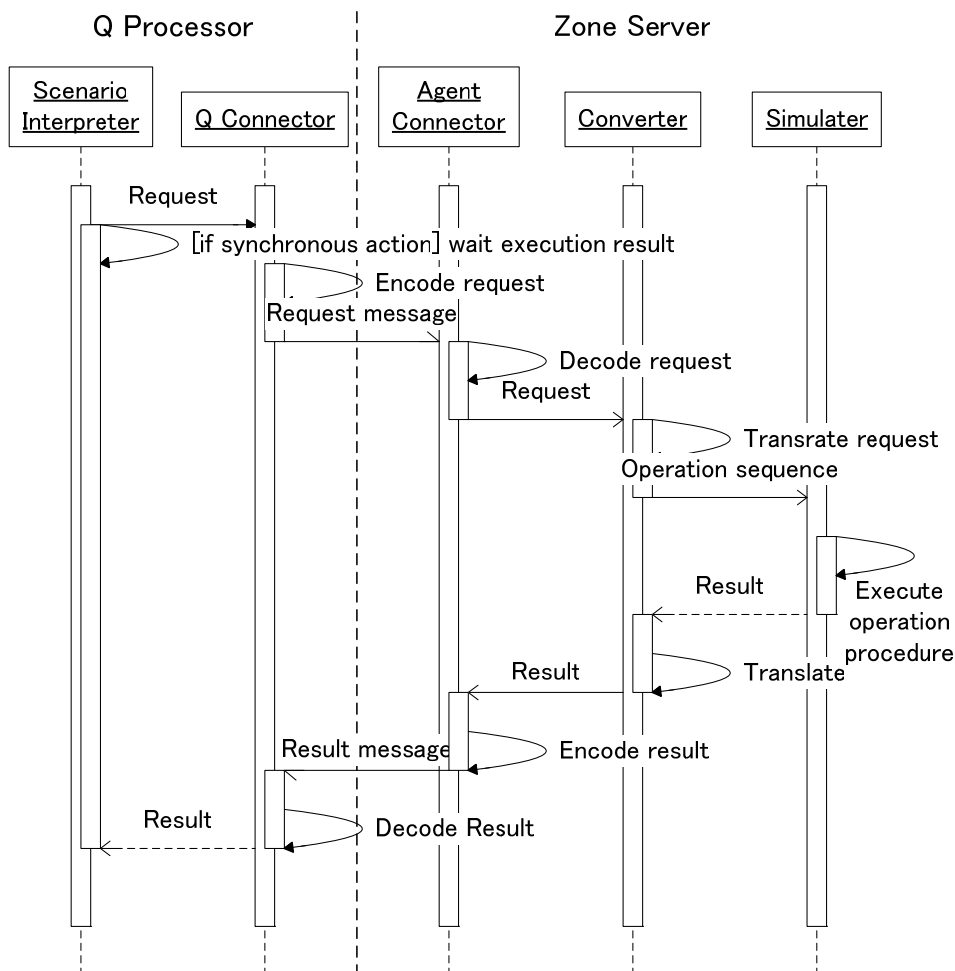


Figure 8: Process of executing cues and actions

Cues are used to get information of surrounding environment in the virtual space, for the purpose of observing events which are used for the triggers of interactions. Agents request for executing cues according to the agent models described in scenarios. Agents request cues with specifying the situations they want to know. Requests of cues are sent from Q Processor to gumonji Zone Server. When gumonji Zone Server receives requests of cues, it executes observations according to the specified condition. When the specified events are observed, results of observations are returned to Q Processor. When Agents receive the results of cues, they begin to perform subsequent actions. In other words, Cues are the triggers of beginning actions. In this way, I realized the functions for executing cues.

I implemented seven observation functions for executing cues, such as “hear”, “see” I implemented general-purpose functions for cues. The abilities of agents such as ranges of observations can be constrained in scenarios. Scenario writers can constrain the observation abilities of agents freely according to the purpose of simulations and situations of agents.

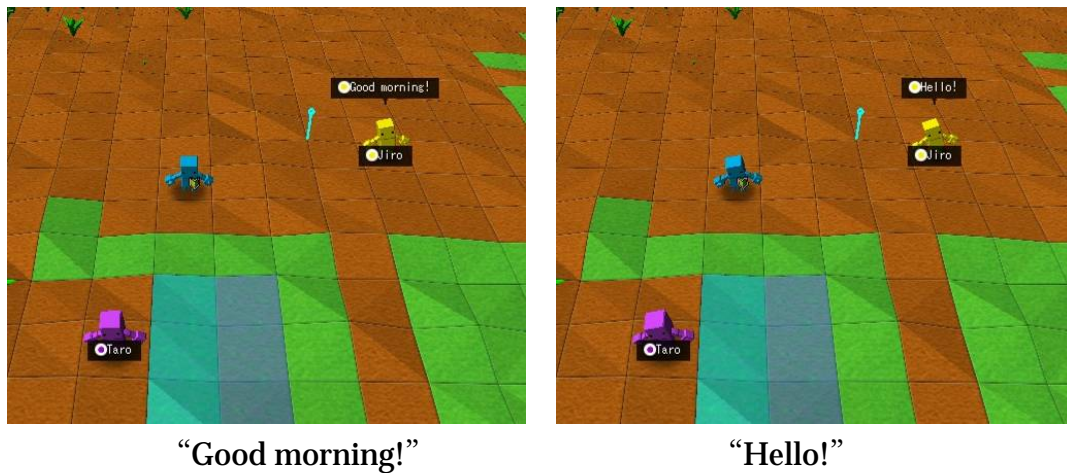


Figure 9: Executing a cue “?hear”

I explain the process of executing cues with an example of executing a cue “hear” according to the process shown in figure 8. Figure 9 shows the situation

when a cue “(?hear-message :message “Hello!” :id \$id)” is executed. This cue means that an agent observes whether the word “Hello!” is said, and get the ID of the speaker when someone said the word. If agent Jiro, who is a yellow colored avatar, said the word “Hello!” as shown in the right figure, the cue is observed and ID of Jiro is acquired. On the other 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 for execution of cues according to the agent models described in scenarios. Request of cues are encoded into string and sent from Q Connector to Agent Connector through TCP/IP connection. Agent Connector decodes received messages and sends them to Cue & Action Converter. In Cue & Action Converter, requests of cues are converted into operators which execute observation of virtual space. In this example, the request of cue “(?hear-message :message “Hello!” :id \$id)” is converted into the operators which search the word “Hello!” and get the ID of speaker. When the word is observed, the ID of speaker is returned as a result of executing the cue. The results are sent to Cue & Action Converter and converted into the format appropriate for Q Interpreter. The acquired ID of speaker is assigned into the result message at the same time. The result messages are encoded into string and sent from Agent Connector to Q Connector through TCP/IP. When Q Connector receives result messages, decodes them and sends results to Scenario Interpreter. In these processes, agents receive the results of executing cues such as ID of Speaker. Acquired information is used in following actions.

Actions are used to activate actions of avatars and change surrounding environment. Agents request for executing actions according to the agent models described in scenarios. Agents request actions with specifying the motions and parameters for activating actions of avatars. Requests of actions are sent from Q Processor to gumonji Zone Server. When gumonji Zone Server receives requests of actions, it executes actions of avatars according to the specified motions and parameters. When finish execution of actions, feedbacks of actions are returned to Q Processor. When Agents receive the feedbacks of actions, they continue to next actions according to the results.

I implemented actions correspond to operations which humans can execute on the user interface. I implemented 45 actions such as “walk”, “speak”, “pickup” I gave agents the ability to operate avatars as same as humans. In other words, agents can operate avatars substitute for humans.

Actions are divided into synchronous actions and asynchronous actions. In the case of synchronous actions, agents wait until receiving results of executing actions. While, in the case of asynchronous actions, agents do not wait for finishing actions but begin immediately next actions.

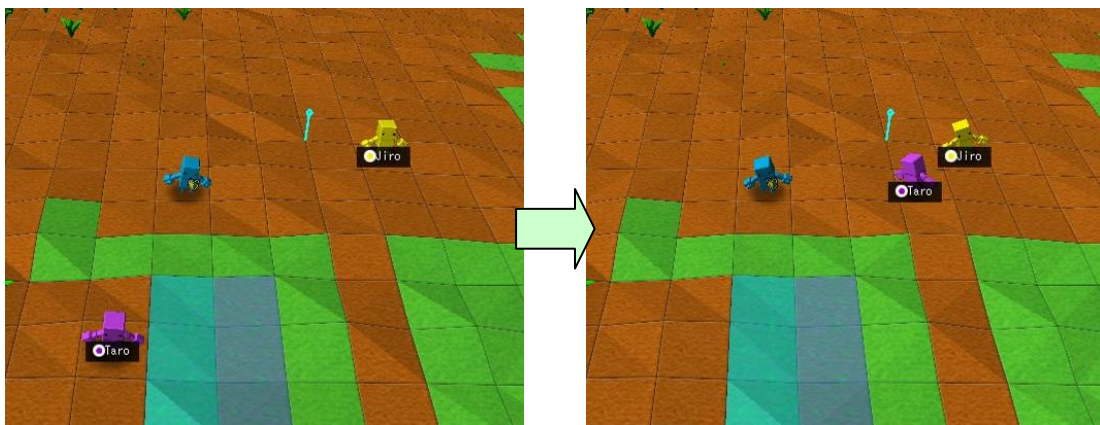


Figure 10: Executing an action “!approach”

I explain the process of executing actions with an example of executing an action “approach” according to the process shown in Figure 8. Figure 10 shows the appearance of virtual space on gumonji when an action “(!approach :id \$id)” is executed. This action means that an agent moves to near the avatar who is specified by the ID “\$id”. When agent Taro, who is a purple colored avatar, executes an action “!approach” in the situation of the left figure, Taro moves to near Jiro, who is yellow colored avatar, as shown in the right figure.

Scenario Interpreter interprets scenarios and requests for executing actions according to the models described in scenarios. Requests of actions are encoded into string and sent from Q Connector to Agent Connector through TCP/IP connection. Agent Connector decodes received messages and sends them to Cue & Action Converter. In Cue & Action Converter, requests of actions

are converted into operators which activate actions of avatars in virtual space. In this example, the request of action “(!approach :id \$id)” is converted into the operators which get the position of the avatar who has the ID specified in “\$id”, and change position of the avatar. When the avatar finishes moving, the new position of avatar is returned as a feedback of executing the action. The results are sent to Cue & Action Converter and converted into the format appropriate for Q Interpreter. The new position of avatar is assigned into the result message at the same time. The result messages are encoded into string and sent from Agent Connector to Q Connector through TCP/IP. When Q Connector receives result messages, decodes them and sends results to Scenario Interpreter. In these processes, actions of avatars are executed in virtual space, and agents receive the feedbacks of executing actions.

Q has the functions of guarded commands which are used to request multiple cues simultaneously. Guarded commands are used to observe multiple events simultaneously, and change states of scenarios according to the observed events. In order to enable executing guarded commands, I implemented the functions for requesting execution of multiple cues concurrently.

When Q Interpreter requests guarded commands, all requests of cues are sent to gumonji Zone Server at the same time. When gumonji Zone Server receives requests of guarded commands, all observations are executed simultaneously. Observations are continued until either of cues is observed. When one of cues in guarded command is observed, observations of other cues are stopped. As results of executing guarded commands, the information of which cue is observed and the information of surrounding environment are returned to Q Interpreter. When agents receive the results of guarded commands, decide actions and change states according to the observed information. In this way, agents can decide behaviors depending on the observed situation.

Thanks to the above functions, agents can autonomously operate avatars according to the models described in Q scenarios.

4.4 Meta-Level Control

Q has meta-level control functions which enable to control executions of scenarios in meta-level. I extended meta-level control functions and enabled to control both scenario executions of agents and avatar operations of humans.

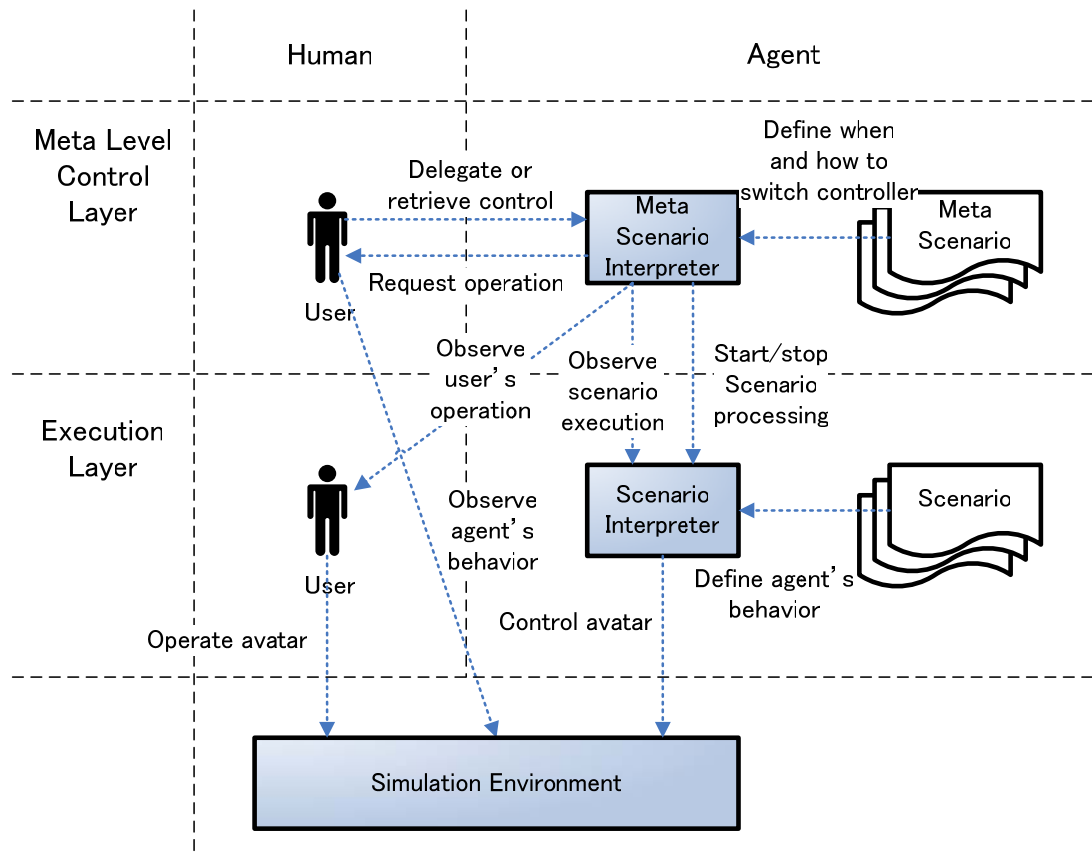


Figure 11: Meta-level avatar control

Figure 11 shows the functions of switching avatar control by meta-level control. These functions consist of two layers such as Execution Layer and Meta-Level Control Layer. These functions enable to switching controllers of avatars between humans and agents.

Humans' operations can be divided into meta-level considerations which switch behaviors according to the situation, and actual operations which execute actions of avatars. Agents' behaviors also can be divided into

Meta-Level Control Layer and Execution Layer. In Meta-Level Control Layer, Meta-Scenario Interpreter controls scenario executions according to the control models described in Meta-Scenarios. In Execution Layer, Scenario Interpreter decides operations of avatars according to the agent models described in Scenarios.

On the human side, humans observe simulation environment in virtual space in meta-level and consider behaviors according to the situation. When humans are operating avatars, humans observe virtual space displayed in gumonji 3D Browser, and switch behaviors according to the surrounding environment of avatars. When humans judge that they do not have to operate avatars directly, they can indicate Meta-Scenario Interpreter to operate avatars autonomously according to described agent models. In this way, humans can entrust operations of avatars to agents at any time. When avatars are operated by agent models, humans can observe behaviors of avatars displayed in gumonji 3D Browser, and if humans consider that executed scenarios are inappropriate, they can indicate Meta-Scenario Interpreter to switch executing scenarios. Furthermore, if humans judge that they should operate avatars directly, they can stop execution of scenarios and take back control of avatars. In this way, humans can switch controller of avatars arbitrarily at anytime.

On the agent side, Meta-Scenario Interpreter controls operations of avatars according to the control models described in Meta-Scenarios. Meta-Scenario Interpreter interprets Meta-Scenarios and observes situation of virtual environment and condition of avatars. Meta-Scenario Interpreter has roles of controlling executions of scenarios according to the situation. The models of meta-level control are described in Meta-Scenarios, such as switching controller from humans to agents if humans are not operating avatars, or starting execution of scenarios if humans log out of simulation. By doing so, it becomes possible to switch controllers of avatars from humans to agents according to the situation. Meta-Scenario Interpreter switches controllers and start executions of scenarios also when humans entrust operations of avatars to agents. The models of requiring operations to humans are also described in Meta-Scenarios, such as stopping executions of scenarios and switching

controllers of avatars from agents to humans if the situation which is not described in scenarios is observed or the feedbacks of actions are changed. By doing so, agents can autonomously require humans to operate avatars, when agents can not make decision autonomously or agents come across the situations which requiring decisions of humans.

By the above meta-level control functions, I enabled controlling operations of avatars in meta-level and switching controllers of avatars dynamically between humans and agents. By switching controllers of avatars dynamically, it becomes possible to continue simulations persistently even when humans are not participating in the simulation.

Chapter 5 Experiment

5.1 Overview of Experiment

I conducted a simulation experiment for the purpose of evaluating the architecture of a platform for persistent participatory simulation. I developed a simulation environment in the virtual space of gumonji/Q about maintenance of natural environment and production of natural resources. In order to verify that persistent participatory simulation is realized, I evaluated how the functions of switching controls work. And in order to verify the availability of persistent participatory simulations, I verified that continuous social changes and behavior patterns of humans can be observed in the simulation.

I enabled to switch controllers between humans and agents by the functions of meta-level control. In order to verify that switching controllers are realized, I checked the operations of meta-level control. I described models of meta-level control in meta-scenarios. In meta-scenario, I described how to switch controllers from humans to agents, and how to switch controllers from agents to humans.

I defined switching controllers of avatars from humans to agents in the following case.

- **When humans log out.**

Human participants previously select the scenarios, which are assigned to agents substitute for operations by humans. When humans log out of simulation, agents automatically start executions of selected scenarios.

- **When humans entrust operations to agents.**

When humans indicate agents to operate avatars, agents start executions of scenarios. Scenarios which are assigned to agents are indicated by humans when they entrust operations.

- **When humans are not operating for more than 10 minutes.**

When avatars are not operated by humans for several minutes, in the case such as humans are away from keyboard, agents automatically start scenario executions.

I defined switching controllers of avatars from agents to humans in the following case.

- **When humans log in.**

When humans log in to the simulation, agents stop executing scenarios, which are executed substitute for humans, and switch controllers of avatars to humans.

- **When humans indicate to stop operations of agents.**

When humans indicate to stop executing scenarios, stop operations of agents and switch controllers of avatars to humans.

- **When scenarios stop executions.**

When all cues are not observed, scenarios can not transit to next state and stop executing. In this case, agents require humans to operate avatars directly.

I conducted a simulation about maintaining natural environment and sustainable use of natural resources. Natural resources such as woods, wildlife animals and aquatic resources are sustainable resources, which regenerate themselves naturally. If the amount of consuming natural resources is less than the amount of regeneration, resources can be used continuously. However, if the amount of consumption is over regeneration, resources would be exhausted. For the purpose of using natural resources continuously, it is necessary to maintain natural environment and manage reproduction of natural resources. In this simulation, I observed continuous changes of natural environment and activities of maintaining natural environment in the virtual space, and verified that it is possible to observe continuous changes of society and behavior patterns of humans in persistent participatory simulation.

I developed simulation environment in the virtual space of gumonji, which is consists of forests, wildlife animals, aquatic lives, and avatars operated by humans and agents. Thanks to the functions of simulating natural environment in gumonji, circulation of atmosphere, water and resources are calculated according to physical law. Therefore, changes of natural environment, which is caused by activities of humans, and spreads of environmental changes are reproduced in virtual environment. In other words, it is possible to simulate

continuous interactions between humans and environment.

As human subjects, general users of gumonji participate in the simulation examination. Participants act in the simulation for the purpose of using natural resources continuously with maintaining natural environment. I observed participants operating natural living things and changes of environment caused by operations of participants. By observing behaviors of participants and changes of environment continuously, I verify that it is possible to observe changes of behavior patterns of humans according to changes of natural environment.

5.2 Result

I conducted a simulation continuously for a week. During conducting a simulation, number of participants fluctuated according to hours. Even if number of human participants was changed, it was possible to continue a simulation by switching controllers of avatars from humans to agents when humans are not participating in the simulation. When humans logged out of virtual space, avatars' operations were switched from humans to agents. On the other hand, when humans logged in to virtual space, avatars' operations were switched from agents to humans. By doing so, it was prevented that avatars go out of action when humans do not operate avatars. Thanks to the functions of switching controllers of avatars, it was possible to continue a simulation for one week regardless of humans' participation.

Humans were able to participate in the simulation at any time. Because of not determining the hours of participation, it was possible to assemble many human participants. However, because of not forcing participation, the number of human participants decreased gradually. In other words, the period when humans are operating avatars became shorter, and the period when agents were operating avatars became longer. Though it was possible to continue a simulation thanks to autonomous operations by agents, it is necessary to assemble more human participants for the purpose of observing complex human behaviors and interactions.

At first, interactions between humans and agents were observed frequently.

However, number of humans who try interacting with agents decreased gradually. It was because agents can not adequately interact with humans. In other words, the scenarios assigned to agents were not enough for interacting with humans. In order to enable interactions between humans and agents, it is necessary to improve agent models described in scenarios.

I developed functions which enable human participants to switch controllers of avatars arbitrarily. Therefore, humans can entrust operations of avatars to agents and take back operations of avatars from agents at any time. In the experiment, it was observed that humans entrusted operations when they went away from keyboard or entrusted agents to repeat routine works automatically. When agents acted despite humans' intentions, humans stopped operations by agents and switched to operating avatars manually. However, the functions of switching controllers of avatars were not used so much. It was because agent models described in scenarios were not enough to operate avatars as humans intended. In order to enable entrusting operations to agents, it is necessary to develop agent models which are enough to operate avatars instead of humans.

I developed functions which enable agents to switch controllers of avatars to humans, when executions of scenarios stopped. Agents were able to request humans to operate avatars manually, in the case such as agents observed situations which were not described in scenarios or agents can not decide operations autonomously. In the experiment, when agents requested humans to operate avatars, controllers of avatars switched to humans, and humans started operating avatars. By doing so, even when agents can not operate avatars autonomously, it was possible to continue activities of avatars. However, it was rarely observed that humans immediately start operations of avatars when agents requested operations to humans. This was because it can not be expected that humans are always observing simulations while entrusting operations to agents. The way of requesting operations of avatars from agents to humans has room for improvement.

This simulation experiment deals with maintaining natural environment and sustainable use of natural resources. At first, there were enough resources to use and participants did not have to be conscious of maintaining natural

environment. Therefore, there were few activities for maintaining natural environment. Almost all activities were consumption of natural resources. Since the amount of consumption exceeded the amount of regeneration of natural resources, production of natural resources decreased gradually. Then, decrease of production gave change to activities of humans. Humans who recognized the deterioration of environment began maintaining natural environment. Thus, it can be observed in the simulation that humans gradually changed behaviors according to the environmental changes.

Activities of maintaining natural environment were not performed uniformly. At first, only a part of human participants were performing activities for maintaining natural environment. After some of them began maintaining natural environment, activities were spread gradually to others. This was because human participants who recognized that they had to maintain natural environment told others to begin activities. And that was also because humans recognized the importance of maintaining natural environment by observing the activities of others. Thus, spread of activities from humans to humans can be observed by conducting simulation for a long time.

During the simulation, activities of maintaining natural environment and consuming natural resources were changed gradually. At first, most of human participants performed activities individually. However, they became to perform activities cooperating with others. It can be considered that since humans who performed activities for maintaining natural environments increased gradually, humans who had the same motivations began performing activities cooperatively. Thus, changes of behavior patterns can be observed by conducting simulation continuously.

5.3 Discussion

In the experiment, humans were able to participate in the simulation at any time. By doing so, many general users of gumonji can easily participate in the simulation. However, by enabling humans to participate at any time, periods of participation varied from human to human. Therefore, the number of participants varied from hour to hour. Of course, it was possible to continue a

simulation regardless of humans' participation, by entrusting operations of avatars when humans are not operating avatars. However, for the purpose of taking into account behavior of humans and observing decision-making process of humans, mechanisms for assembling more human participants should be developed.

In this experiment, scenarios which were assigned to agents were selected from those which were prepared before or written by participants. Therefore, scenarios which participants can select were limited and not enough to operate avatars substitute for humans. In order to solve this issue, mechanisms for constructing scenarios from observation of humans' behaviors can be useful. In the platforms for persistent participatory simulation, it is possible to switch controllers of avatars dynamically between humans and agents. So, by recording behaviors of humans while humans are operating avatars, agents can observe humans' behaviors and learn behavior models. Moreover, by requesting humans to operating avatars when agents can not decide actions autonomously, agents can improve their models by taking into account decisions of humans. In this way, it becomes possible to learn agent models which reflect humans' decision-making processes. As the mechanism for constructing agent models, such a way of learning functions should be developed.

I developed meta-level control functions for the purpose of switching controllers of avatars between humans and agents. In the experiment, by describing control models in meta-scenarios, I enabled to switching controllers of avatars dynamically. However, it was not enough to switching controllers according to the situations. As the theory of switching controllers between humans and agents, there are researches of Adjustable Autonomy. It remains to be solved to develop mechanisms for switching controllers between humans to agents by taking into account the theory of Adjustable Autonomy.

In this research, I used a network game for developing a platform for persistent participatory simulation. This was because network games have functions useful to realize persistent participatory simulation. Moreover, network games have other characteristics that human participants can enjoy

playing activities in the virtual space. By taking into account advantages of these characteristics, this platform can be used for learning. As an application of gumonji/Q, participatory simulation can be used for the purpose of learning. By using gumonji/Q, human participants can learn and discuss about natural environment through experiences in virtual environment.

Chapter 6 Summary

In this research, I proposed architecture for developing platforms for persistent participatory simulation using network games. Persistent participatory simulation means that both humans and agents participate in and conduct simulations continuously for a long time. Persistent participatory simulation enables to observe changes of social systems and behavior patterns of humans caused by interactions among multiple decision-making actors.

For the purpose of developing a platform for persistent participatory simulation using network game, there were two issues such as realizing interactions between humans and agents and switching controllers of avatars between humans and agents. I solved these issues by following two approaches.

1. **Unified process of operating avatars**

For the purpose of enabling agents to operate avatars, I developed functions which enable agents to observe surrounding environment of virtual space and request actions of avatars in real time. In this process, requests of operating avatars from agents are sent to simulator via Message Converter. Message Converter converts requests into operators as same format as operations from humans. By processing operations from humans and agents in the same way, I enabled to execute actions of avatars without distinguishing humans and agents. In this way, it became possible that both humans and agents interact with each other in the same way.

2. **Meta-level control of avatar operations**

I enabled both humans and agents to operate the same avatars, and developed meta-level control functions which observe operations of humans and conditions of agents and control operations of avatars. Meta-level control enabled switching controllers of avatars dynamically between humans and agents.

I implemented a platform for persistent participatory simulation gumonji/Q by integrating network game gumonji and scenario description language Q. The gumonji is a network game which has functions of environmental simulation. Scenario description language Q is an interaction design language

which allows us to define interaction models in extended finite state machine model. By integrating the gumonji and Q, I enable agents to operate avatars in virtual space of the gumonji from outside. Moreover, I extended meta-level control functions of Q and enabled switching controllers of avatars dynamically between humans who operate avatars on gumonji Client and agents who request actions according to the agent models described in scenarios. In addition, for the purpose of using the gumonji, which is open to public, as a platform for simulation, I developed gumonji/Q jointly with Community Engine Inc.

I conducted a simulation experiment about sustainable use of natural resources using gumonji/Q, for the purpose of verifying availability of persistent participatory simulation. In the experiment, it was possible to observe spread of activities from individual to many actors include humans and agents by interactions between them. It was also possible to observe gradual changes of humans' behaviors according to the changes of surrounding environment, by conducting simulation for a long time. Thus, it can be said that a platform for persistent participatory simulation was realized, which are useful for reproducing complex social phenomena and designing social systems.

In this research, I developed a framework which enables switching decision-making actors dynamically, and it became possible to switch controllers of avatars between humans and agents. In this framework, I would like to design mechanisms of switching controllers of avatars as a future works.

As a future work, gumonji/Q can be used as a simulation platform for environmental education. Thanks to the gaming interface of the gumonji, participants can enjoy playing in the virtual simulation environment and experience activities virtually. Participants can learn about natural environment and the influences of activities to living things, by trying actions in virtual space and observing changes of environment. For example, if participants experience that not only trees decrease but also living things around the forests become extinct when they cut forests excessively, they can learn from experience that maintaining forests is important. Thus, gumonji/Q

is useful as a platform for educational simulations where participants learn from experiences in virtual environment.

Acknowledgments

The author would like to express his sincere gratitude to the supervisor, Professor Toru Ishida at Kyoto University, for his continuous instruction, valuable advice and helpful discussions.

The author would like to express his thanks to the advisers, Professor Tetsuro Sakai at Kyoto University, Professor Osamu Katai at Kyoto University, and Professor Yasuhiko Kitamura at Kwansei Gakuin University, for his valuable advice and discussion.

The author would like to express his thanks to Associate Professor Shigeo Matsubara at Kyoto University, and Assistant Professor Hiromitsu Hattori at Kyoto University, for his valuable advice and discussion.

The author would like to express his thanks to Mr. Kengo Nakajima and Ms. Marika Odagaki at Community Engine Inc., for their various discussion and technical support.

Finally, the author would like to thank all members of Ishida & Mastubara laboratory for their various supports and discussions.

References

- [Drogoul94] Drogoul, A., Ferber, J., Multi-agent Simulation as a Tool for Modeling Societies: Application to Social Differentiation in Ant Colonies, Proceedings of the 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-92), pp. 3--23, 1994.
- [Axelrod97] Axelrod, R., Advancing the art of simulation in the social sciences, Complexity, Vol. 3, No. 2, pp. 16--22, 1997.
- [Deguchi98] Deguchi, H., Agent Based Approach for Social Complex Systems - Management of Constructed Social World, Lecture Notes In Computer Science, Vol. 1519, pp. 61--76, 1998.
- [Gunderson99] Gunderson, J., P. and Martin, W., N., Effects of uncertainty on variable autonomy in maintenance robots, Proc. of Agents' 99, Workshop on Autonomy Control Software, 1999.
- [Colella00] Colella, V., Participatory Simulations: Building Collaborative Understanding Through Immersive Dynamic Modeling, The Journal of the Learning Sciences, Vol. 9, No. 4, pp. 471--500, 2000.
- [Bousquet02] Bousquet, F., Barreteau, O., d'Aquino, P., Etienne, M., Boissau, S., Aubert, S., Le Page, C., Babin, D., and Castella, J.C., Multi-agent Systems and Role Games: An Approach for Ecosystem Co-Management, Complexity and Ecosystem Management: The Theory and Practice of Multi-agent Approaches, Edward Elgar Publishers, pp. 248--285, 2002.
- [Rickel99] Rickel, J., Johnson, W. L., Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control, Applied Artificial Intelligence, Vol. 13, No. 4-5, pp. 343-382, 1999.
- [Prensky03] Prensky, M., Digital game-based learning, Computers in Entertainment (CIE), Vol. 1, No. 1, pp. 21-21, 2003.
- [Murakami05] Murakami, Y., Sugimoto, Y., and Ishida, T., Modeling human behavior for virtual training systems, Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05), pp. 127--132, 2005.
- [Torii04] Daisuke Torii and Toru Ishida, Stéphane Bonneaud and Alexis Drogoul. Layering Social Interaction Scenarios on Environmental Simulation.

Workshop on Multiagent and Multiagent-based Simulation (MAMABS), International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04), pp. 61-70, 2004.

[Bainbridge07] Bainbridge, W.S., The Scientific Research Potential of Virtual Worlds, *Science* Vol. 317, No. 5837, pp. 472-476, 2007.

[Brandherm08] Brandherm, B., Ullrich, S., Prendinger, H., Simulation of sensor-based tracking in Second Life, Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS08), pp. 1689-1690, 2008.

[Rehm08] Rehm, M. and Rosina, P., SecondLife as an evaluation platform for multiagent systems featuring social interactions, Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS08), pp. 1663-1664, 2008.

[Ishida02] Ishida, T., Q: A Scenario Description Language for Interactive Agents, *IEEE Computer*, Vol. 35, No. 11, pp. 42--47, 2002.

[Nakanishi04] Nakanishi, H. and Ishida, T., FreeWalk/Q: Social Interaction Platform in Virtual Space, ACM Symposium on Virtual Reality Software and Technology (VRST2004), pp. 97--104, 2004.

[Bousquet98] Bousquet, F., Bakam, I., Proton, H. and Le Page, C., Cormas: common-pool resources and multi-agent Systems, *Lecture Notes in Artificial Intelligence*, Vol. 1416, pp. 826--838, 1998.

[Deguchi04] Deguchi, H., Tanuma, H. and Shimizu, T., SOARS: Spot Oriented Agent Role Simulator-Design and Agent Based Dynamical System, Proceedings of the Third International Workshop on Agent-based Approaches in Economic and Social Complex Systems, pp. 49--56, 2004.

[Yamane06] Shohei Yamane and Toru Ishida. Meta-level Control Architecture for Massively Multiagent Simulations. Winter Simulation Conference (WSC-06), pp. 889-896, 2006.

Appendix:

A.1 How to Use gumonji/Q

1. gumonji/Q とは

1.1. gumonji/Q の紹介

gumonji/Q では、**gumonji** 上のキャラクターや動物にシナリオを割り当てることで、エージェントとして自動的に動かすことが可能になります。

エージェントは、与えられたシナリオに従って自動的に行動します。つまり、シナリオを与えるということは、キャラクターや動物に命を吹き込むということです。キャラクターに雑草を刈るシナリオを与えてあげれば、自動的に草刈をしてくれるようになるでしょう。また、動物に言葉に反応して動作するシナリオを与えてあげれば、自分の指示に答えて動いてくれるようになるでしょう。

オリジナルのシナリオを作って、世界のキャラクターや動物に与えてあげることで、あなただけの世界の住人をつくってみてください。

1.2. gumonji/Q の動作概要

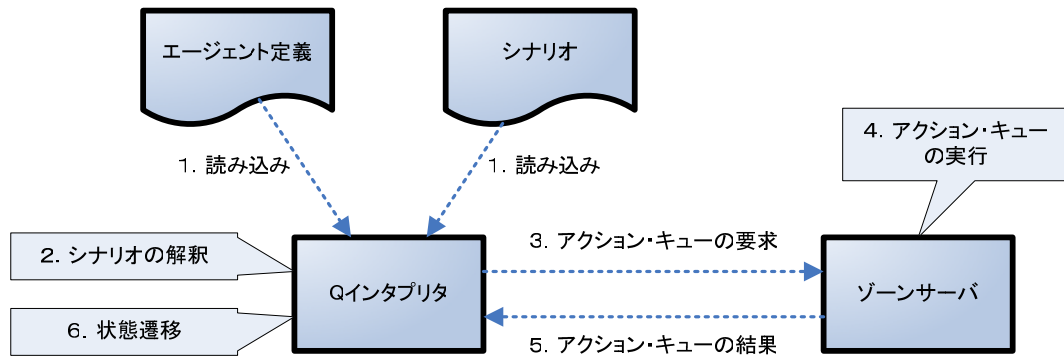
シナリオは、ある事柄を観測するとそれに応じた動作を行って次の状態に移るといふ、状態遷移機械モデルで記述します。観測はキューとして、動作はアクションとして指定します。

エージェントは、状態に応じて 1 つ以上の事柄を観測し、その中のいずれかの条件が満たされるとその条件に応じた動作を行い、次の状態に移行するという行動を繰り返します。

シナリオの解釈は **Q** インタプリタで行い、観測や動作の実行はゾーンサーバで行います。ゾーンサーバと **Q** インタプリタの間での処理の流れを図 1 に示します。

1. **Q** インタプリタでエージェント定義とシナリオを読み込む
2. **Q** インタプリタでシナリオを解釈し、次の行動を決定する
3. **Q** インタプリタからゾーンサーバにアクション・キューの実行を依頼する
4. ゾーンサーバで、依頼されたアクション・キューを実行する

5. ゾーンサーバから Q インタプリタにアクション・キューの実行結果を送る
6. Q インタプリタでアクション・キューの実行結果に応じてエージェントの状態を更新する
7. 終了状態でなければ 2 に戻る



(図 1) シナリオ実行の流れ

2. gumonji/Q の実行方法

2.1. JAVA ランタイムのインストール

以下の手順で、**JAVA** ランタイムをインストールしてください。すでにインストールされている場合は、次に進んでください。

1. <http://java.com> にアクセスして、最新の **JAVA** ランタイムをダウンロード
2. **JAVA** ランタイムをインストール

2.1. Q の実行許可

以下の手順で、ゾーンサーバで Q シナリオの実行を許可するように設定してください。

1. **gumonji** ゾーンサーバの“設定”ボタンを押す
2. “Q の設定”タブを選択する
3. “Q シナリオの実行を許可する”をチェックする
4. “OK”を押す

2.2. Q 処理系の起動

以下の手順で、Q 処理系を起動し、シナリオの実行に必要なライブラリとシ

ナリオを読み込んでください。

1. “runManager.bat” もしくは “runManager_auto.bat” を実行

※ “runManager_auto.bat” を実行すると、2.3 で “(start)” を入力しなくても自動でシナリオの実行を開始します。

コマンドプロンプトと Q ウィンドウの 2 つのウィンドウが開きますが、使用するのは Q ウィンドウのみです。コマンドプロンプトにはログやエラーメッセージなどが表示されます。

“scenario/main.q” に、シナリオとエージェントの定義を読み込んで実行を開始するまでのスクリプトが書かれています。自分で定義したファイルを追加で読み込みたいときは、適宜書き換えてください。

例えば、scenario フォルダ内に myscenario.q というファイルでシナリオを定義したときは、以下の記述を追加してください。

```
(load "../scenario/myscenario.q")
```

※カレントディレクトリは、“qv2-src” となっているので注意

2.3. シナリオの実行開始

以下の手順で、シナリオの実行を開始してください。

1. シナリオの実行開始 (Q ウィンドウに以下のコマンドを入力)

```
(start)
```

※ 2.2.Q 処理系の起動で、“runManager.bat” の代わりに “runManager_auto.bat” を実行すると、“(start)” を入力しなくても自動でシナリオの実行を開始します。

2.4. シナリオの実行停止

Q ウィンドウを閉じることで、シナリオの実行を終了します。

2.5. gumonji/Q の終了

ゾーンサーバを停止することで、**gumonji/Q** の実行を終了します。

エージェントの位置や所持アイテムなどの情報は、ゾーンのデータの中に保存されます。

※**Q** ウィンドウ上で実行可能なコマンドの詳細については、**gumonji/Q** アクション・キューリファレンスの、“**4.Q** のコマンド” を参照してください。

3. シナリオの書き方

サンプルシナリオのファイルに追加で記述するか、別のファイルにシナリオを記述して **Q** ウィンドウで読み込むことで、自由にシナリオを作ることができます。拡張子が “.q” や “.scm” のファイルは、テキスト形式のファイルです。適当なテキストエディタで閲覧・編集することが可能です。

定義したシナリオは、エージェントに割り当てることで実行が開始されます。また、複数のキュー・アクションから成る機能のまとまりを **1** つのシナリオとして記述して、別のシナリオから呼び出すこともできます。

詳しくは、“シナリオ記述言語 **Q** 言語仕様書” を参照してください。

3.1. シナリオの記述

以下の形式でシナリオを定義します。

```
(defscenario シナリオ名 (引数リスト) シナリオ実体)
```

3.2. エージェントの作成

以下の形式でエージェントを定義します。

```
(defagent エージェント名 パラメータリスト)
```

※詳しくは、アクション・キューリファレンスを参照してください。

3.3. シナリオの割り当て

以下の形式でシナリオをエージェントに割り当てます。

```
(assign-scenario 'シナリオ名 'エージェント名 パラメータリスト)
```

3.4. シナリオの呼び出し

シナリオ内で以下のように記述することで、シナリオから別のシナリオを呼び出すことができます。

```
(シナリオ名 self パラメータリスト)
```

4. シナリオ自動割り当て

以下の方法で、エージェントを自動的に作成してシナリオを割り当てます。

“**robot-scenario**” もしくは “**animal-scenario**” のフォルダ内に、“エージェント名.q” というファイル名でシナリオを記述したファイルを置くことで、エージェントに自動的にシナリオが割り当てられます。

1. ファイルの作成

“エージェント名.q” という名前でファイルを作成してください。ファイル名をそのままエージェント名としてエージェントが作成されます。

2. シナリオの記述

エージェントに割り当てたいシナリオを、作成したファイル内に記述してください。ファイル内に記述したシナリオが自動的にエージェントに割り当てられます。

3. ファイルの配置

ロボットの場合は “**robot-scenario**”，動物の場合は “**animal-scenario**” というフォルダ内にファイルを置いてください。

※**robot-scenario** フォルダに記述したものだけを実行したいときは、

“scenario.q”と“agent.q”を読み込まないようにしてください。
“scenario/main.q”の、(load "../scenario/scenario.q")および (load "../scenario/agent.q")の行頭に; (セミコロン) を加えてコメントアウトすることで、読み込まないようにすることができます。

※robot-scenario 内のシナリオを書き換えて再割り当てをするためには、シナリオを開放した上で、Q ウィンドウに以下のコマンドを打ち込んで再読み込みを行ってください。

```
(load "../lib/scenarioloader.scm")
```

5. サンプルシナリオ

Q の使い方の例として、いくつかのサンプルシナリオを用意しました。サンプルシナリオの動作を簡単に説明します。

初期状態では、シナリオ **1~8** が実行されるようになっています。実行するシナリオを変えたいときは、“scenario.q”の最後のシナリオ割り当ての部分を書き換えてください。行頭に“;”を付けるとコメントアウトされますので、実行させたいときは“;”を消し、実行させたくないときは“;”を付けるようにしてください。

動物にシナリオを実行させるためには、指定した名前の動物が存在している必要があります。シナリオを実行させたい動物に名前を付けるか、エージェントの作成の際に存在している動物の名前を指定してください。

1. 同じ発言を繰り返すシナリオ (scenario-speak)

“こんにちは”、“お元気ですか?”、“天気がいいですね!”、“さようなら”という言葉を繰り返します。

2. ユーザからの発言に返答するシナリオ (scenario-hear)

ユーザまたは周囲のキャラクターの言葉を聞いて、それに対して返答します。近くに **scenario-speak** を与えたエージェントがいる場合は、そのエージェントと対話します。

3. 円を描いて走り回るシナリオ (**scenario-move**)

円を描いてキャラクターが走り回ります。

4. **\$name** に指定したキャラクターを追い掛け回すシナリオ (**scenario-follow**)

指定したキャラクターを追い掛け回します。 **\$name** に別の名前を指定すると、そのキャラクターを追い掛け回すようにできます。

5. アクションを繰り返すシナリオ (**scenario-action**)

横たわる、ジャンプ、座る、手を振る、おじぎのアクションを繰り返します。

6. 表情の変化を繰り返すシナリオ (**scenario-express**)

ほほを赤くする、大粒涙をこぼす、ときめく、目からビームを出す、はてなマークの表情を繰り返す。

7. **\$name** に指定した動物の名前を付け替えるシナリオ (**scenario-name**)

動物の名前を付け替える。 **\$name** に別の名前を指定すると、その動物の名前を付け替えるようにできます。

8. バッジの付け替えを繰り返すシナリオ (**scenario-badge**)

バッジを順番に付け替えます。

9. アイテムの装備を繰り返すシナリオ (**scenario-equip**)

缶、バケツ、シャベル、手斧の装備を繰り返します。

10. 持っているアイテムを置いて拾うシナリオ (**scenario-put**)

缶、インクローラー、シャベル、バケツを周囲に置いてそれを拾うという動作を繰り返します。

11. シャベルで地面を掘ったり埋めたりするシナリオ (**scenario-shovel**)

地面を掘って、移動して、土を埋めるという動作を繰り返します。

12. シャベルで地面を掘ったり埋めたり固めたり、バケツで水をすくったり撒

いたりするシナリオ (**scenario-shovel2**)

地面を掘って、埋めて、平らにして、なめらかにして、固めて、水をすくって、撒くという動作を繰り返します。

13. 植物の種を収穫して刈るシナリオ (**scenario-harvest**)

指定した植物を探して刈ります。種がついている場合は種を収穫してから刈ります。近くに指定した植物がない場合にはランダムに歩き回ります。“雑草”を別の植物に替えると、その植物を刈るようにできます。

14. 動物を捕獲するシナリオ (**scenario-fish**)

指定した動物を探して狩り、肉を拾います。1000 モルより小さい動物は刈りません。肉を拾った場合は、今までに拾った肉の合計のモル数を言います。近くに指定した動物がいない場合はランダムに歩き回ります。“フナ”を別の動物に変えると、その動物を狩るようにできます。

15. ランダムに歩き回るシナリオ (**scenario-randomwalk**)

ランダムに歩き回ります。

16. アイテムの装備を繰り返すシナリオ 2 (**scenario-equip2**)

手斧、バケツ、シャベルの順に装備したあと缶を装備し、すべての装備を外すという動作を繰り返します。

17. ロボット全員で踊るシナリオ (**scenario-dance**)

全体アクションの動作例を示すシナリオです。ロボット全員で同じ動きをして踊ります。

6. 各ファイルの説明

- **runManager.bat**

Q 処理系の起動, **lib/main.scn** の読み込み

- **runManager_auto.bat**

Q 処理系の起動, **lib/main_auto.scn** の読み込み

- **lib/main.scm**
gumonji/Q の起動に必要なファイルの読み込み
- **lib/main_auto.scm**
gumonji/Q の起動に必要なファイルの読み込み（自動実行）
- **lib/nodelib.scm**
gumonji/Q ライブラリ
- **lib/handler.scm**
ハンドラの記述
- **lib/cueadapter.scm**
キュー（観測）の記述
- **lib/actionadapter.scm**
アクション（行動）の記述
- **lib/definition.q**
キュー・アクションの設定
- **lib/scenarioloader.scm**
シナリオの自動割り当て
- **scenario/main.q**
シナリオとエージェント定義の読み込み
- **scenario/scenario.q**
シナリオの記述
- **scenario/agent.q**
エージェントの設定

A.2 Action Reference

!walk

移動先を指定して歩く

:x :y	in	整数	移動先の絶対座標を指定	指定しない場合は現在地
:dx :dy	in	整数	移動先の相対座標を指定	指定しない場合は 0

※!walk による移動は、ゾーンサーバ上では瞬時に指定された場所へ移動し、3D ブラウザが移動を遅らせて表示することで移動に時間がかかっているよう

に見せかけている。従って、**!walk** のアクションのみでは移動にかかる時間を確保することができないため、移動時間を確保するためには**?wait** を用いる必要がある。なお、移動時間の確保の仕方については、今後のバージョンアップにより仕様変更を行う可能性がある。

!approach

対象を指定して歩いて近づく

:id in 文字列 対象の ID を指定
:dx :dy in 整数 対象との相対座標を指定 指定しない場合は 0

!turn

方角を指定して方向転換

:dir in 整数 方角を指定(北:0 東:1 南:2 西:3)

!speak

対象と発言内容を指定して話す (ロボットのみ)

:id in 整数 対象の ID を指定
:message in 文字列 発言内容を指定

!send

対象のエージェントを指定してメッセージを送る (ロボットのみ)

:name in 文字列 対象エージェントの名前を指定
:message in 文字列 メッセージの内容を指定

!express

表情を指定する (ロボットのみ)

:face in 文字列 表情を指定 ※表 1 参照

※現在対応していません

!perform

アクションを実行する

:action in 文字列 状態を指定 ※表 2 および表 3 参照

※現在一部のアクションのみしか対応していません

!look

対象を指定して状態を調べる

:id	in	整数	対象の ID を指定	
:keys	in	文字列	対象から受け取る変数名のリストを指定	※表 4 参照
:values	out	文字列	指定した変数値のアソシエーションリストを取得	

※取得した変数の値は、(get-value-str 変数名 アソシエーションリストへの参照) で得られる。

!observe

地点を指定して地面の状態を調べる

:x :y	in	整数	絶対座標を指定	指定しない場合は現在地
:dx :dy	in	整数	相対座標を指定	指定しない場合は 0
:keys	in	文字列	対象から受け取る変数名のリストを指定	※表 5 参照
:values	out	文字列	指定した変数値のアソシエーションリストを取得	

!name

対象を指定して名前を付ける

:id	in	整数	対象の ID を指定	対象は動物
:name	in	文字列	名前を指定	

!denominate

対象を指定してグモ学名を付ける

:id	in	整数	対象の ID を指定	対象は動物または植物
:name	in	文字列	グモ学名を指定	

!rotate

対象と回転角を指定して回す

:id	in	整数	対象の ID を指定	対象は植物またはアイテム
:angle	in	整数	角度を指定	

!harvest

対象を指定して種を収穫する (ロボットのみ)

:id	in	整数	対象の ID を指定	対象は実を付ける植物
-----	----	----	------------	------------

!attach

バッヂを付ける (ロボットのみ)

:badge in 整数 バッヂの種類を指定 ※表 6 参照

!get-item-id

指定した種類のアイテムの中で, **index** 番目のアイテムの **ID** を得る

アイテムの種類を指定しない場合は, アイテム欄内での **index** 番目の位置にあるアイテムの **ID** を得る

:index	in	整数	アイテムの位置を指定	指定しない場合は 1 ※1~48
:type	in	文字列	アイテムの種類を指定	指定しない場合はすべて ※(葉っぱ, うまのタマゴなど)
:id	out	整数	アイテムの ID を取得	

!equip

アイテムを指定して装備する

:id	in	整数	アイテムの ID を指定	
:type	in	文字列	種類を指定	※(缶, シャベルなど)

!put

アイテムを指定した地点に置く

:id	in	整数	アイテムの ID を指定	
:x :y	in	整数	絶対座標を指定	指定しない場合は現在地
:dx :dy	in	整数	相対座標を指定	指定しない場合は 0

!merge

アイテムを指定して同種類のアイテムをまとめる

:id	in	整数	アイテムの ID を指定	
-----	----	----	--------------	--

!overlay

指定した同種類のアイテム 2 つを重ねる

:dest	in	整数	重ねる先のアイテムの ID を指定	
:from	in	整数	重ねる元のアイテムの ID を指定	

!divide

指定したアイテムを半分に分割する

:id in 整数 アイテムの ID を指定

!price

指定したアイテムに指定した値段を付ける

:id in 整数 アイテムの ID を指定

:price in 整数 値段を指定

!takeout

指定したアイテム（缶・バケツ・タンク）から指定した量のアイテムを取り出す

:id in 整数 アイテムの ID を指定

:amount in 整数 取り出す量を指定

!store

指定したアイテムを缶・バケツ・タンクに収納する

:id in 整数 アイテムの ID を指定

!pickup

指定したアイテムを拾う

:id in 整数 アイテムの ID を指定

!bury

指定したアイテムを埋める

:id in 整数 アイテムの ID を指定

!plant

指定した種を植える

:id in 整数 アイテムの ID を指定

!incubate

指定した卵を孵化させる

:id in 整数 アイテムの ID を指定

!write

指定したアイテムにコメントを書く

:id in 整数 アイテムの ID を指定
:message in 文字列 コメントの内容を指定

!read

指定したアイテムのコメントを読む

:id in 整数 アイテムの ID を指定
:message out 文字列 コメントの内容を取得

!delete

指定したアイテムを削除する

:id in 整数 アイテムの ID を指定

!dig

地点・範囲・深さを指定して地面を掘る

:x :y in 整数 絶対座標を指定
:dx :dy in 整数 相対座標を指定
:range in 整数 範囲を指定
:height in 整数 深さ(mm) (差分を指定)

指定しない場合は現在地
指定しない場合は 0
1辺の長さが $(range+1)*2$ の正方形
※0~2

!raise

地点・範囲・高さを指定して地面を盛る

:x :y in 整数 絶対座標を指定
:dx :dy in 整数 相対座標を指定
:range in 整数 範囲を指定
:height in 整数 高さ(mm) (差分を指定)

指定しない場合は現在地
指定しない場合は 0
1辺の長さが $(range+1)*2$ の正方形
※0~2

!flatten

地点・範囲を指定して、地面をエージェントがいる地点の高さにそろえる

:x :y in 整数 絶対座標を指定
:dx :dy in 整数 相対座標を指定
:range in 整数 範囲を指定

指定しない場合は現在地
指定しない場合は 0
1辺の長さが $(range+1)*2$ の正方形
※0~2

!smooth

地点・範囲を指定して地面をなめらかにする

:x :y in 整数 絶対座標を指定
:dx :dy in 整数 相対座標を指定
:range in 整数 範囲を指定

指定しない場合は現在地
指定しない場合は 0
1辺の長さが $(range+1)*2$ の正方形
※0~2

!harden

地点・範囲を指定して地面を固める

:x :y in 整数 絶対座標を指定
:dx :dy in 整数 相対座標を指定
:range in 整数 範囲を指定

指定しない場合は現在地
指定しない場合は 0
1辺の長さが $(range+1)*2$ の正方形
※0~2

!scoop

地点・範囲・量を指定して水をすくう

:x :y in 整数 絶対座標を指定
:dx :dy in 整数 相対座標を指定
:range in 整数 範囲を指定
:amount in 整数 各セルからすくう水の量を指定 (mol)

指定しない場合は現在地
指定しない場合は 0
1辺の長さが $(range+1)*2$ の正方形
※0~8
指定しない場合は 1000

!spill

地点・範囲・量を指定して水を撒く

:x :y in 整数 絶対座標を指定
:dx :dy in 整数 相対座標を指定
:range in 整数 範囲を指定
:amount in 整数 各セルに撒く水の量を指定 (mol)

指定しない場合は現在地
指定しない場合は 0
1辺の長さが $(range+1)*2$ の正方形
※0~8
指定しない場合は 1000

!fell

対象を指定して植物を刈る

:id in 整数 植物の ID を指定

!hunt

対象を指定して動物を狩る

:id in 整数 動物の ID を指定

!chemical

指定したアイテムを作る

:item in 文字列 アイテムの名前を指定

※ケミカルで作成できるアイテム名
(布, 木の机など)

!play-note

音を鳴らす

:height in 整数
:pitch in 整数 音の種類を指定
:note in 整数 音の高さを指定
:length in 整数 音の長さを指定

A.3 Cue Reference

?hear-message

指定した内容に一致する発言が聞こえると、発言した対象の ID を得る

:message in 文字列 発言内容を指定
:id out 整数 発言したキャラクターまたはロボットの ID を取得

?hear-from

指定した対象の発言を聞くと、その発言内容を得る

:id in 整数 キャラクターまたはロボットの ID を指定
:message out 文字列 発言内容を取得

?receive-message

指定した内容に一致するメッセージを受け取ると、送り主の名前を得る

:message in 文字列 メッセージ内容を指定
:name out 文字列 キャラクターまたはロボットの名前を取得

?receive-from

指定した対象からメッセージを受け取ると、その内容を得る

:name in 文字列 キャラクターまたはロボットの名前を指定
:message out 文字列 メッセージ内容を取得

?see

指定した条件に一致する対象が指定した範囲内に存在すると、その ID を得る

:category	in	文字列	カテゴリを指定	指定しない場合はすべて ※表 7 参照
:type	in	文字列	種類を指定	指定しない場合はすべて ※(葉っぱ, うま, 丸い木など)
:name	in	文字列	名前・グモ学名を指定	指定しない場合はすべて
:color	in	文字列	色を指定	指定しない場合はすべて ※表 8 参照
:x :y	in	整数	対象地点の絶対座標を指定	指定しない場合は現在地
:dx :dy	in	整数	対象地点の相対座標を指定	指定しない場合は 0
:dist	in	整数	対象地点からの範囲(距離)を指定	指定しない場合は 0
:id	out	整数	対象を特定する ID を取得	

?wait

指定した時間が経過する

:time	in	整数	待機する時間を指定(gumonji 上の分単位)	指定しない場合は 0(必ず発火)
-------	----	----	--------------------------	------------------

※ゾーンの時間が止まっていると発火しない

?is-equip

指定した条件に一致するアイテムを装備しているかを調べる

:id	in	整数	アイテムの ID を指定	
:type	in	文字列	種類を指定	※(缶, シャベルなど)

※種類または ID のいずれかの条件に一致するアイテムを装備していれば発火