7. Toward the Self-sustaining Virtual Society

7.1. Introduction

New forms of culture and social phenomenon are springing on everywhere in a network environment as communication technologies shrink the distance between users and computer technologies enable creating a sense of "place" or "society" in the environment. Also, our research goal --- a 3D virtual society --- can provide an enough opportunity for people to create culture and for social phenomena to occur in the environment. As can be seen in this thesis, technically, 3D MUSVEs enable the realization of a 3D "virtual place" populated by many people and multi-user social activities among them within computer system over the network. Especially, 3D graphical nature of the virtual environments enables more various user activities than other types of collaborative virtual environment such as chat system. Unlike other human interface systems, 3D virtual environments have more possibility of organizing more user-driven cultural/creative activities inside them naturally like the real world.

As we described in this thesis, our virtual society, i.e. PAW^2 has successfully created a sense of "place" populated by many people over the Internet. This virtual place has many the social attributes, and many of the usual social mechanisms operate there. Certain attributes of this virtual place tend to have significant effects on culture and social phenomena in the environment, leading to new mechanisms and new modes of behavior that are rarely seen in the real life. We consider that the next important step is to extend these environments into a "self-sustaining" virtual society.

In this chapter, we introduce a notion of self-sustaining virtual society and discuss the possibility of realizing it from the results described in this thesis and its issues. In addition, we describe some plans for the future work about MUSVEs and personal agents.

7.2. Definition

We define a "self-sustaining virtual society" as a virtual society which provides creative positive feedback loops to enrich the environment and can maintain the society by itself. We can say that the self-sustaining virtual society is the "cultural" version of closed ecological systems, e.g. "Biosphere 2"[1]. Biosphere 2 is a structure built in an attempt to build an "earth-like" system to sustain life without any supports from outside. It was privately constructed in the late 1980s and tried to discover if eight people could sustain themselves in a sealed, energy-rich environment.

If users can create their own culture or content in a virtual society, it can be a virtual society which sustains itself. Consequently, we can expect the users to be from content-eater to content-creator and the virtual society to be a self-sustaining virtual society. In this environment, some user-created culture or content can become new content in the virtual world and also attract other new users to visit the world. Again, the new users may be able to create more new culture or content on top of the pervious one. A self-contained virtual society can provide this kind of creative positive feedback loop to enrich the environment. We can expect the followings about the self-sustaining virtual society:

- It can realize more attractive user-oriented virtual society naturally.
- It may well produce new types of culture and content that have not seen in the real life.
- It can operate autonomously without any user support from the system administration side.

Making a self-contained virtual society without any user support from the system administration side is the ultimate goal of all research on virtual societies. So research on supporting/realizing such a creative positive feedback loop to achieve the goal will be more important after the technology for achieving a virtual society itself has been established.

In the initial report of Habitat, Morningstar and Farmer (1991) also reported as follows: the most important factor to form a cyberspace itself is users' interactions

[1] See also http://www.bio2.edu/. Earth, by the way, is Biosphere 1.

in the cyberspace and operation method to allow users to establish their own goals. It was effective to form and enrich cyberspace.

7.3. Discussion

As we described, PAW^2 has very limited social functions for supporting user's creative or cultural activities. In such a sense, PAW^2 world looks like the pre-Stone-Age world, which had almost no tools. Nevertheless, as described in this thesis, we observed that several unique social phenomena arose: many communities with various features were formed, various user events were hosted, original rules were established, and users skillfully took advantage of PAW^2's social infrastructure to improve their life in PAW^2. In particular, in the user events, we observed that users skillfully utilized the limited PAW^2's functions to hold their events. Also, users applied PAW^2's social infrastructure in their own ways, which we (PAW^2 builders) had not imagined, to improve their lives in PAW^2. For example, handle name system for announce short messages to unspecified users. In the sense of the improvement of their lives by users in PAW^2, we can recognize that users are trying to form their own culture in the virtual society.

PAW^2 users were observed to establish their own way to enjoy life in PAW^2 while using several functions of PAW^2 in combination for a long time. For example, after chatting with a certain user, some PAW^2 users played with their personal agents, participated in an event, or wrote in-world mail until they could find other users to talk with. We can see here their creation of a "life style" in the virtual environment. The life style also influences those of other users. For example, "Flower Garden Plan" event and its mail reporting the results of the event is a good example (Section 5.3.2(3)). We can see here that a useful creative positive feedback loop is being established and the virtual world itself is functioning as a kind of medium. User events also produce a feedback loop and are important as user content to make a self-sustaining virtual society. To enhance the loop, it is useful to provide functions that help users to hold user events, such as a function to create special items for the events, add a script to them to carry out event-specific behavior, and arrange them in the PAW^2 world. As we observed in unique user activities in web-based PAW^2 magazines, some advanced users were publishing their own articles or stories in the virtual world (Section 5.3.2(2)). The content itself is based upon interviews with other PAW^2 users, their activities in PAW^2, user events, or happenings in PAW^2. This is another good example of users finding their own way of enjoying PAW^2 and then their activities become new content.

To expand these cultural or creative activities of PAW^2 users to create a richer culture and content, the underlying system needs to provide more flexibility

and functions to enable the users to create culture or develop content by themselves. However, the big issues here are security and morals about the functions. As we described in this thesis, some anti-social behaviors occurred even in a society like PAW^2 which did not provide much flexibility compared with real society. Conversely, we need to consider that the flexibility that is provided by the system has a strong possibility of being used for anti-social behavior which the flexibility makes possible. This is a kind of a double-edged sword in terms of culture formation or creative activities in the environment and we need to deal with it carefully. It is because, as we described in Section 5.3.2(8), most users are very sensitive to such anti-social behaviors.

To prevent such anti-social behaviors, the functions provided by a virtual society should be socially acceptable ones and they should support a socially-acceptable interface described in Section 4.7.8. In addition, it is important to design the socially-acceptable functions from the viewpoint of recipients in interaction, such as the mute function or function to pass objects (Section 4.7.8). The mechanism for unshared information proposed in (Matsuda, 2001) is also a useful tool for designing socially-acceptable functions (see also Section 7.5.8). Meanwhile, we can extend the mechanism of avatar to prevent such behaviors by changing avatar's behavior according to the context or situation in a MUSVE. Currently, the avatar is a "direct" proxy of user and the user's operations to her avatar are "directly" converted into her avatar's behaviors in a MUSVE. Therefore, for example, a user can throw her personal agent's dung both at recycle boxes and at other users (Section 5.3.2(8)). These behaviors should be changed according to the context in the virtual environment. For this purpose, we can introduce a kind of "behavior interpreter" module between user's operations and her avatar's behaviors (Matsuda, 2001; Matsuda 2002). The interpreter can detect the context or situation in a MUSVE to stop anti-social behaviors by "not" converting user's operations directly into her avatar's behaviors. We will describe this idea in Section 7.5.8.

We believe that anti-social behaviors should be dealt with by the underlying system as much as possible and should not be occurred by technical means. However, some problems that cannot be dealt with by the underlying system may occur depending on the social system provided to users. In the near future, in addition to the technical development to prevent anti-social behaviors, support from the system administration side may also be needed, especially in the initial stage of a self-sustaining virtual society. For example, adequate checks on user identity at the time of registration, logging each user's activities, and introducing some kind of user surveillance mechanism (e.g. a police force) to fix "broken windows" in a virtual society. The "broken windows" indicates light crimes in a society (Kelling et al, 1997). Kelling and others suggested in their "broken windows" theory that the best way to prevent crimes is to prevent the disorder that precedes them, such as graffiti,

panhandling, uncollected trash and unrepaired buildings. We also may be able to apply this theory to a virtual society. In terms of provision of a police force, for example, MUD supported the privileged users called "wizards" or "gods". They have permission to limit the freedom of the user who caused a social problem. Curtis (1996) reported that this approach was useful to prevent social problems.

However, essentially, the main thing is to educate users about the virtual society and the education which should be carried out in the real world. For example, users should consider the question: "The virtual world is a public place opened to anyone like a park in the real world. How should you behave there?"

7.4. Approaches

To sublimate a virtual society into a self-sustaining virtual society, it is important to clarify what is essential to realize creative positive feedback loops in a virtual society, which enriches the environment. We could find out several primary positive feedback loops within PAW^2 by making observations about user activities and phenomena of cultural formation or content creation in the virtual society. We believe that such an observation-based approach is useful to investigate the essential factors to realize the creative positive feedback loops toward the self-sustaining virtual society.

Based upon the observation, we need to evaluate what kind of functions or system should be provided to form culture or develop content. There are two types of approach to develop them: inductive approach and deductive approach. In the inductive approach, we should develop the necessary functions/system based upon analyzing the existing primary positive feedback loops in PAW^2. This approach also includes the development of functions/system to enhance the loops themselves. On the other hand, in the deductive approach, we can develop new functions/system based upon a new hypothesis for enhancing the loops to apply them to PAW^2. Both approaches should be taken. In terms of designing the functions/system, in particular, socially acceptability of the functions/interfaces will become more important.

Also, it is important to evaluate what types of culture/content are formed/developed after the provision of the functions/system. Based upon the result of the evaluation of the culture and content, they should be refined. Then, the evaluation of the refined function/system should be done, again. This evaluation loop is important in this research. We also need to establish evaluation criteria about the culture and content developed in a virtual society. We consider that the results of cultural anthropologic studies can also be applied to the evaluation.

7.5. Other Research Topics

Research on a personal agent-oriented virtual society involves many issues beyond those described in this thesis. Finally, we describe some plans for our future works. This is considered in terms of: virtual society middleware, extension of a personal agent, research framework for physical personal agent, support for e-commerce in a virtual society, support for personal virtual societies, asynchronous interactions with a virtual society, mixed reality and ubiquitous computing, and "unshared" shared virtual environment.

7.5.1 Virtual society middleware

Several functions described in this thesis constitute components of middleware for realizing a virtual society. Based upon our experience on implementation of CP system and PAW^2 and its operation over the Internet, we found that a middleware is important to realize a virtual society from both user's and developer's perspectives. We also have developed PAW^2 based upon our middleware that was in turn based upon AO to realize a virtual society. Most content-dependent functions are implemented using the middleware. It gives a fundamental to "Virtual Society Middleware." Virtual Society Middleware is a bunch of libraries that enable world developers to easily build a virtual society. It also allows users to organize social activities and to easily live in a virtual society.

It consists of the following three types of middleware: in-world middleware, inter-world middleware, and outer-world middleware. Figure 7.1 shows architecture of these middleware.

(1) In-world middleware: allows world developers to implement functions inside a virtual world. It provides libraries for user management, clock

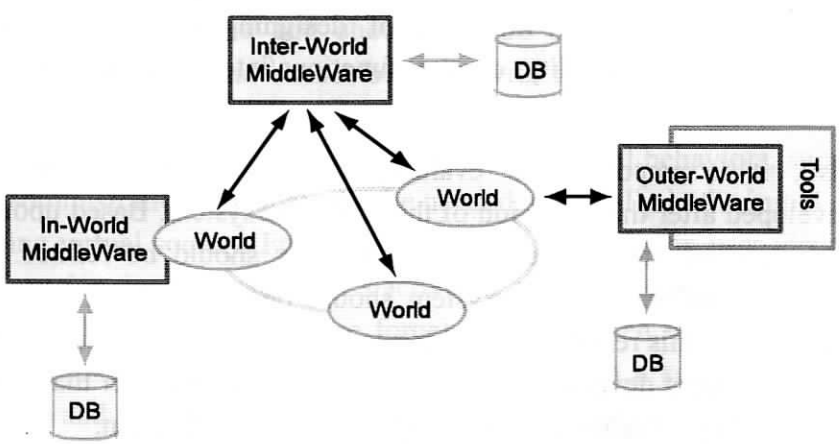


Figure 7.1 An architecture of Virtual Society Middleware

system, object management, barter system, scene management, in-world communication systems, and so on. This middleware is of prime importance to provide social and environmental infrastructure of MUSVEs.

(2) Inter-world middleware: allows world developers to implement functions among multiple virtual societies. In simultaneously providing multiple MUSVEs that have some relationships among them, developers use this middleware to implement several functions supporting activities carried out between the worlds. This middleware provides libraries for some kind of RMI (Remote Method Invocation) function from one MUSVE to another, inter-world communication systems (e.g. inter-world mail system, pager, and phone system), an inter-world object delivery system that enables users to send an object from one MUSVE to another, such as a courier service or postal mail system.

(3) Outer-world middleware: allows world developers to implement functions to control or manage MUSVEs from the outside. This middleware provides libraries for administration of MUSVEs (e.g. managing users or checking the status of CP bureau/AOs/database which manage the MUSVE), displaying urgent messages to CP browsers (users) accessing the MUSVE, controlling in-world functions, and so on. It also provides remote control tools based upon these functions.

It is important to evaluate the reusability of these middleware for other virtual society content, and, if necessary, it may be necessary to generalize them for reuse. In addition, our research on PAW^2 mainly has focused on in-world middleware. However, based upon the experience with PAW^2, we consider that outer-middleware is also important.

7.5.2 Personal agents

To make a personal agent-oriented virtual society like PAW^2 more attractive and to enrich its user interface, it is important to enhance the user interface of a personal agent. We consider that there are two approaches: enhancement of the personal agent's natural language interface and linking the personal agent with the real world.

As described in Section 4.7.3, a personal agent supports a limited natural language interface extended upon ELIZA. Even if it provides the limited functions, we have observed that the natural language interface seems to be attractive to PAW^2 users (Section 5.2.2). In addition, this interface is especially efficient in an environment like PAW^2 where many mediums complexly exist. This is because it

is impossible to add all functions into a GUI such as a control panel. As the following step, it is important to enhance the natural language interface of the personal agent by using A.I. technology. We can introduce various applications using this interface. They include education of a personal agent, sending a message to other users via a personal agent, and an interface to on-line services such as weather information in the real world.

We can see that one of approaches to make a personal agent more friendly to users is to link the personal agent with a pet robot (e.g. SONY "AIBO") (Fujita et al., 1998) in the real world. In other words, the pet robot is treated as an "agent" (proxy) of the personal agent in a virtual society. We can introduce various applications with this combination approach. For example, the personality of a personal agent in a virtual society can reflect to a pet robot. Similarly, the information retrieved by the robot in the real world can reflect back to the personal agent in the virtual society. In addition, this approach allows a user to pass the personality of her personal agent to another user's one as if she placed her "real" pet to a friend in the real world. The friend can download the personality into her pet robot in the real world. The robot can have a shared experience with the friend and reflect back to the original personal agent. The original agent will talk about the experience to her owner as if people were talking about their experience on a trip (e.g. "I had an interesting experience with your friend" or "I saw a beautiful scene in Hokkaido").

As another approach for introducing a personal agent into the real world, we can utilize AR technology to merge the "virtual" personal agent directly within the real world (Section 7.5.7).

7.5.3 Support for real-world personal agents

As described in the previous section, recent robotics technology allows people to personally hold a pet robot, such as AIBO. AIBO (Fujita et al., 1998) is an entertainment pet robot with ability to communicate with its senses, such as touch, sight, hearing, and balance. It can communicate with people by using behaviors and sound. Currently, such pet robots mainly support a functionality to interact with people as a pet and have not assisted people like a personal agent in PAW^2. However, robotics technology and agent technology will be able to realize a personal agent with a physical body in the real world in the near future. We call such a personal agent "real-world personal agent". The real-world personal agent needs to interact with the real world or access the owner's personal information to take appropriate actions to assist the owner.

We believe that MUSVEs can be utilized as a software framework for research on such real-world personal agents. In particular, social MUSVEs (virtual societies) can act as a simulation of the real world. In the real world, it is hard to access information about the environment. Ubiquitous computing (Weiser, 1993) can provide the solution. However, in a virtual society, a personal agent can easily retrieve and take advantage of information about the environment and the owner's personal information to support her. We can also provide multi-user interactions to the personal agent to simulate a similar situation in the real world.

In MUDs, i.e. text-based virtual environments, Doyle and Hayes-Roth (1997) introduced the idea of "annotated environment" to allow a personal agent to learn about the environment in the same way a user would. World builders can add annotated information (e.g. name of place and its functions) to the environment and the personal agent can retrieve the information. In PAW^2, a personal agent can easily access her owner's information to decide her behaviors. However, PAW^2's infrastructure does not provide a uniform interface to access the information about the virtual environment itself. This interface should provide information in two ways: (1) A personal agent actively uses the interface to retrieve the information. (2) The virtual environment itself pushes the information to the agent according to the context in the environment (e.g. location), if necessary. By using the mechanism of interaction between AOs, we can realize these functions.

The most important feature of this approach is that we can easily control the scalability of information provision to a personal agent. The dynamics of the real world are unpredictable and a real-world personal agent could receive noisy input from the environment. In her initial learning, the personal agent does not need to be given all the details of its environment. We can start by intentionally restricting the information in a MUSVE for specific purposes of research on a real-world personal agent. Based upon the result of the initial learning, we can relax the restriction seamlessly in the same MUSVE.

7.5.4 Support for e-commerce in a virtual society

As described in Chapter 6, in our e-commerce experiment, the target user might be limited to the small portion of all users visited to PAW^2. There are two reasons why the target user group of our experiment might have been limited to a subset of the all users during the experiment: users could not choose payment methods other than credit cards, and the virtual goods could only be purchased from a particular retail Web page.

A solution for the first problem would be to provide a cash-based payment method like bank transfers or pre-paid type electronic money[1]. The second problem could be solved by providing a virtual society-oriented or object-oriented sales method. Figure 7.2 shows an example. In the figure, when a user (refer as "user A") shows an item, by using a right hand, to other user (refer as "user B") in a virtual world, user B can purchase the item immediately on the spot by clicking it. This is a method which combines user's social interaction in a virtual society and a purchase system. This approach is also socially-acceptable (Section 4.7.8). As another example, users can purchase items immediately by clicking them on their advertising boards while in the virtual world.

We believe that offering a variety of attractive purchase methods is important in a virtual society by taking advantage of its characteristics, i.e. the freedom from the constraints of the real world. We believe that it is also the most attractive point in the case of using a virtual society as a distribution medium of virtual objects.

In addition, we could introduce a new type of C2C (Customer to Customer) business model based upon virtual societies. For example, one of the models is to share the revenues between users and service provider by selling items created by them in a virtual society and collecting the sales commission from the revenue of the items. We consider that this type of business model will make a virtual society more active and attractive.

We will also further investigate other content payment methods, the pattern of user purchase behavior and the life cycle of goods in a virtual society by gathering statistical information about the purchase behavior and user activities related to the goods. In our opinion, further experimentation into the e-commerce

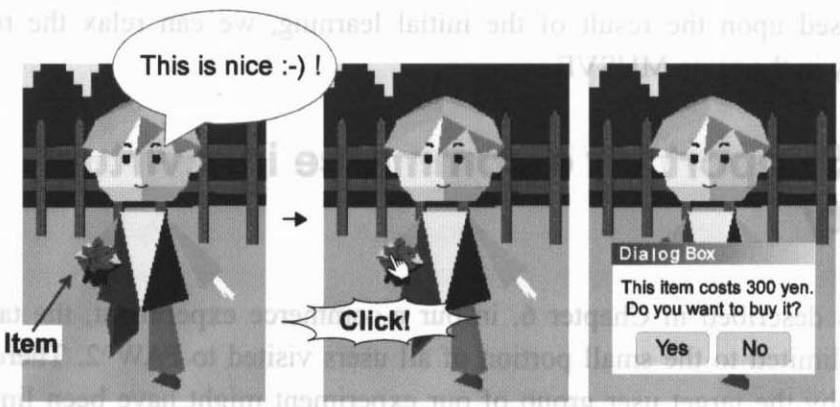


Figure 7.2 Example for virtual society-oriented sales method.

[1] For this purpose, "WebMoney" is available in Japan. See http://www.webmoney.ne.jp/ (in Japanese).

possibilities of goods that enhance social life within a virtual society is a rich area for future research.

7.5.5 Support for personal virtual societies

We can allow individual users to host their own virtual societies based upon CP system and PAW^2 system. As described in this thesis, one CP bureau supports one thousand users simultaneously. It means that, for example, we can realize twenty separate small scale virtual worlds with fifty users in a single CP bureau. If each world is located more than the size of user's aura away from the other worlds, the users in one world are unaware of other worlds (Figure 7.3). This approach also allows users to easily teleport to other worlds, for example, by using functions like "Meet" function in a name card system (Section 4.4.1).

We can apply the PAW^2 system described in this thesis to these multiple small worlds to realize multiple small PAW^2 worlds. These worlds are equivalent to the multiple islands in PAW^2. This approach allows the users in each world to utilize PAW^2's social infrastructure as if they were in a single PAW^2 world. We can extend the Universal AO to display the list of the multiple worlds when a user accesses them. After selecting the target world from the list, the Universal AO locates the user to the world. Each small world should be manages by AO ("PVS AO" in Figure 7.3) to prevent unauthorized users from entering the other small

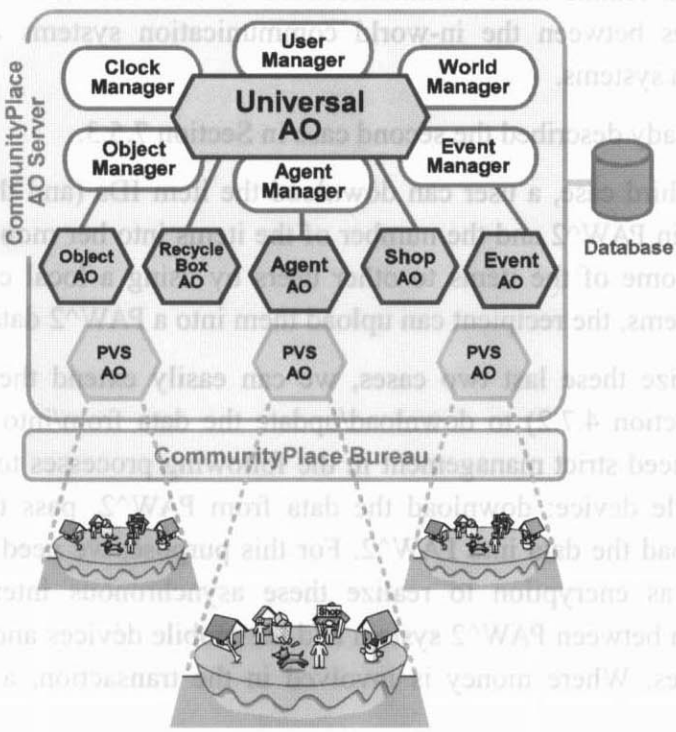


Figure 7.3 Personal virtual society architecture

worlds. To increase the number of the worlds, the WLS (Section 3.3.5) can provide the solution.

7.5.6 Asynchronous interactions with a virtual society

Recent computer and network technology allows people to access the Internet from mobile devices, such as cell phones or PDAs. Furthermore, recent 3DCG technology enables these devices to display 3D virtual worlds on their screens[1]. These technologies allow people to access a virtual society from anywhere.

These environments allow people to access a virtual society not only synchronously but also asynchronously. In the synchronous environment, of course, we can provide a virtual society that allows them to directly access from the mobile devices. These environments will lead to the mixed reality environments described in the next section. In the asynchronous environment, we can provide the following three interactions to users: communication with in-world communication systems, interaction with a personal agent, and passing items to other users.

In the first case, a user can communicate with in-world communication systems from her mobile devices. For example, the user can send an in-world mail to her friends in PAW^2 from the Internet as if she were sending an e-mail to her friends. We can realize these communications by introducing the gateway system which mediates between the in-world communication systems and the Internet communication systems.

We already described the second case in Section 7.5.3.

In the third case, a user can download the item IDs (and their bitmap data) that she holds in PAW^2 and the number of the items into her mobile device. Then, she can pass some of the items to other users by using a local connection. After receiving the items, the recipient can upload them into a PAW^2 database.

To realize these last two cases, we can easily extend the User Object or Agent AO (Section 4.7.2) to download/update the data from/into PAW^2 system. However, we need strict management in the following processes to protect the data in user's mobile device: download the data from PAW^2, pass the data to other users, and upload the data into PAW^2. For this purpose, we need to have security features such as encryption to realize these asynchronous interactions in both communication between PAW^2 system and the mobile devices and communication between devices. Where money is involved in the transaction, a tamper-resistant

---

[1] http://www.access.co.jp/english/press/030214.html

hardware device is necessary to protect the data in the mobile device from making unexpected copies.

7.5.7 Mixed realities and ubiquitous computing

MUSVE is a technology for realizing everything within the computer system, i.e. within the virtual environment. For example, users and objects exist within the environment and user activity and communication also take place through them within the environment. Realizing a virtual society within the computer system is an important research topic. However, we have another approach to extending a new type of virtual society into the real world by realizing virtual avatars, objects, and also a personal agent within the real world.

AR technology (Bajura et al., 1992) can break down the boundary between "real" and "virtual" by overlaying the real world with information and virtual objects. This technology also allows introduction of virtual avatars and objects into the real world. It enables each participant in the real world to share the same real world environment via a HMD (Head Mounted Display) with the avatars accessing the environment remotely. Furthermore, the same virtual objects or personal agents in the environment can be shared among them. In terms of interaction with the real world objects in the environment, ubiquitous computing (Weiser, 1993) can provide the solution. The distant users represented by these avatars can also share the same environment realized in a MUSVE system remotely via a computer display or HMD. They can see the participants as an avatar in the environment.

In the environment, for example, you can communicate with the distant users at a street corner in the real world as if they were standing there. The idea is also mixed reality. We can combine both results of research on mixed reality and MUSVE to realize a new generation of MUSVE mixing real and virtual.

7.5.8 "Unshared" shared virtual environment

As described in Section 2.1, MUSVE systems have already been explored in a number of experimental research platforms. However, in the majority of cases the works have been confined to realization of strict "shared" virtual environments. Or, most works have focused on the environment achieving "what you see is **same as** what I see": actions that occur in MUSVE must be propagated to all users sharing the environment (Section 3.6.1). So "shared-ness" has been the most important research issue in the previous works.

However, as described in this thesis, we found that "unshared-ness" is also useful in MUSVEs. It positively achieves "what you see is **different** from what I

see". This environment allows you to see a red ball which other users see as a white ball. We have already introduced this mechanism of unshared information In PAW^2 system (Matsuda, 2001). For example, a personal agent often gives a hint to her owner in PAW^2's events. This behavior should be unshared between her and other users surrounding her. "Mute" function (Section 4.7.8) and "invisible human" function (Section 5.3.2(8)) are also good examples of this mechanism. The common important point among these examples is "its syntax is not shared but semantics is shared". We can extend this idea to bridge the cultural gaps between people sharing the environment. For example, the syntax of beckoning gesture is different between Japan and United States (Williams, 1998) but its syntax is same. The mechanism of unshared information can provide the solution. As described in Section 6.6, we can utilize the idea of "behavior interpreter" of avatar not only to prevent anti-social behaviors but also to bridge this gap. In terms of prevention of anti-social behaviors, we may extend the behavior interpreter into a kind of "conscience" mechanism of avatar, which we already have in our mind. The user of this avatar with the mechanism needs to keep the avatar's conscience.

This is an area which has begun to be explored by us and others (Matsuda, 2001; Greenhalgh et al., 2000). This idea can also be applied not only to MUSVEs but also to all distributed shared communication environments including MR environments.

7.6. Summary of this Chapter

In this chapter, we introduced a notion of a self-sustaining virtual society, and discussed the possibility of realizing it. We also presented some plans for the future work.

A self-sustaining virtual society is defined as a virtual society which provides creative positive feedback loops to enrich the environment and maintain the society by itself. As we described in this thesis, we have observed primary positive feedback loops occurring naturally in PAW^2 even if PAW^2 does not support enough functionality. Based upon our operation of PAW^2, we consider that a 3D virtual society on the Internet has enough potential to form/develop its own culture/content and sublimate into a self-sustaining virtual society by enhancing the feedback loops. To enhance the loop, the underlying system should be more flexible and has to be able to prevent anti-social behaviors being enabled by the flexibility.

We considered the future work in terms of: virtual society middleware for allowing application developers to construct and control virtual societies easily, enhancement of a personal agent to make her more attractive and enrich her user interface, system framework for research on real world personal agents, support for

e-commerce in a virtual society, support for personal virtual societies to allow many people to host their own worlds in the Internet, realization of asynchronous interactions with a virtual society by using mobile devices, and mixed reality and ubiquitous computing for merging real society and virtual societies together. Finally, we considered introduction of "unshared-ness" of information as new interaction methods not only in MUSVEs but also in all distributed shared communication environments including MR environments.

8. Conclusion

In this thesis, we proposed and evaluated a new approach to supporting social activities via computer technologies and computer networks. First of all, we designed and constructed a "CommunityPlace" system for realizing 3D MUSVEs over the Internet. We used the system to design and construct a personal agent-oriented virtual society "PAW^2" that allows many people to organize multi-user social activities in the virtual environment. Next, we exposed the PAW^2 into the Internet and then conducted experiments based upon the environment to evaluate a virtual society. We evaluated PAW^2 from the following viewpoints: its design policy, user profile, user activities and social activities taking place in the environment, and e-commerce and business model through the environment. As we described in this thesis, we have observed that initially most users were accessing PAW^2 simply to enjoy the service provided by the system but its personal agent and social infrastructure gradually encouraged them to organize social activities in the environment. After that, they have formed communities, hosted their own events, and organized creative/cultural activities to enrich their life and the environment. These results illustrate that we have achieved a first step toward our goal to extend 3D MUSVE into a virtual society. In this chapter, we summarize the contributions in this thesis.

We proposed a new distributed system framework for realizing 3D MUSVEs based upon the WWW architecture. One implementation of this framework consists of a distributed system platform: CommunityPlace. It supports a CommunityPlace browser, a CommunityPlace bureau, and an AO. The CommunityPlace system is a natural extension to the WWW system and enables world developers to construct 3D MUSVEs easily and seamlessly fit them into the Internet environment. It also enabled the MUSVE to extend itself dynamically by using shared applications called "AO" without stopping the service. In addition, the system allows AO to execute in a different computer where CommunityPlace bureau is running. This model also allows separation between server (bureau) manager and service providers and provision of secure MUSVEs. The system supports both client-server system and

peer-to-peer system and artifact-based AOI (Area Of Interest) algorithm to enable many users to access the environment simultaneously. With a dummy user application, we showed that one thousand users can access the environment simultaneously.

Based upon the CommunityPlace system, we designed, implemented, and evaluated a 3D virtual society: PAW^2. In addition to functionalities of existing MUSVEs, PAW^2 provides a social and environmental infrastructure where each user has her own personal agent and ability to organize social interactions. We showed the design policy for realizing a virtual society, its framework, and implementation. The PAW^2 system consists of individual AOs representing objects in a virtual society and their managers. The AO maintains the functions and local consistency of the objects. Each manager maintains global consistency of the AOs managed by the manager. We evaluated the policy and PAW^2 by exposing the environment into the Internet. It allowed all Internet users to access PAW^2 freely. Within the initial 8 months, PAW^2 achieved more than 30,000 registered users with hundreds of simultaneous user accesses and thousands of accesses. In addition, the results of the questionnaire and database analysis about PAW^2 showed that we had successfully introduced a personal agent into the 3D MUSVE and realized a 3D virtual social space over the Internet.

We analyzed user activities and social activities taking place in PAW^2. In terms of user activities, we evaluated usage patterns and user's motivation to access PAW^2 using the statistical information on user activities. We extended the system to log user activities automatically and used statistical methods to analyze the data. The results showed that the most frequent user activities were basic system functions plus activities picked up by user's observing other users' behaviors. The trend analysis of user activities indicated that the rate of heavy users was independent of the previous PAW^2 experience and the combinations of user activities changed gradually. We learned that a personal agent had a significant impact on first-time users and then user activities gradually shifted to communication and events and games-related activities. In terms of social activities, we clarified social phenomena and the anti-social activities observed in PAW^2. In these observations, we found a kind of cultural/creative activity formed on the basis of PAW^2's functions, that we never anticipated, and primary positive feedback loops to enrich the environment itself. PAW^2 is functioning as a kind of medium for these phenomena/activities. We compared our observations with other virtual worlds and found some commonality. These results show that initially most users were accessing PAW^2 simply to enjoy the service provided by the system but they have gradually organized the unique social/cultural phenomena and creative activities in PAW^2.

We proposed a new business model based upon a virtual society and a framework of e-commerce extension to a virtual society based upon the model. We

actually extended PAW^2 to support the model and evaluated it by using the environment. Our underlying hypothesis for this evaluation was that defining the meaning or value of a virtual object in a virtual society makes it possible for that object to be successfully sold for money like real goods. We carried out the e-commerce experiment in selling a virtual object for real money based upon the hypothesis for two months. The results showed that the hypothesis was successfully supported. A comparison between the results of this experiment and those of an investigation on user purchase behaviors for digital content showed that although the sales figures were similar for the two, user purchase behavior in PAW^2 has room to increase the number of buyers. The results also showed a business potential for agent technology-based B2C business.

Finally, we proposed the notion of a self-sustaining virtual society as the next generation of a virtual society. This would allow users to create culture and content of their own and provide various positive cultural/creative feedback loops to enrich the environment based upon the content. The self-sustaining virtual society can extend the environment in a natural self-sustaining way. Based upon the results described in this thesis, we consider that a 3D virtual society on the Internet has enough potential to develop its own culture/content and sublimate it into a self-sustaining virtual society by enhancing the loops. The notion of a self-sustaining virtual society is important in realizing the next generation of more attractive user-oriented virtual societies.

Previously, Asimov (1994) has created "Three Laws of Robotics" as a way of governing the behavior of the robots in his SF stories. Based upon our experience of operation of PAW^2, we propose the following five design policies for building a virtual society:

- Introduction of an (personal) agent to help people to join the society seamlessly and mingle with others.
- Introduction of a social and environmental infrastructure to allow people to organize social activities.
- Introduction of a function to encourage people to stay in the world for a long time.
- Introduction of a function to allow people to have various shared experiences.
- A peaceful scene of the world to promote a harmonious environment.

Research on user activities and the methodologies to allow users to form/develop a culture/content in a virtual society is also important to build a "better" virtual society. These works could lead to studies on cultural anthropology

or sociology in a cyberspace ("virtual cultural anthropology" and "virtual sociology") and establish a new academic discipline in the existing computer science. We believe that a virtual society itself will become the next generation of user interfaces not only for Internet but also for broadcast media.

Appendix A.
CommunityPlace Browser Manual

A.1 Introducing Community Place Browser

Community Place Browser software enables you to navigate through virtual worlds on the Internet. With this software, you can freely travel through the virtual world in various modes; you can walk about, turn a corner, change directions, change the speed of your movement, look up, down, and all around. In the virtual world you can also listen to music, watch a movie, and interact with active animated objects. Some worlds will have cartoon characters, animated actors and even games for you to enjoy.

If you happen to share the same virtual world with other people, you can interact with them: chatting, taking part in the same event or discussing what else is happening in this and other virtual worlds.

Community place is just that, a real 3D electronic place where you can meet friends, chat with new people and interact with a whole host of exciting characters and scenes scattered throughout the internet.

Starting Community Place browser

Once you have installed Community Place correctly, you are ready to start using it. When you click the link toVRML file, you will start the Browser and display the VRML world (Figure A.1).

There are two additional methods for starting Community Place:

(1) Select the [Start]... [Program Files]... [Community Place Browser]... [Community Place Browser].

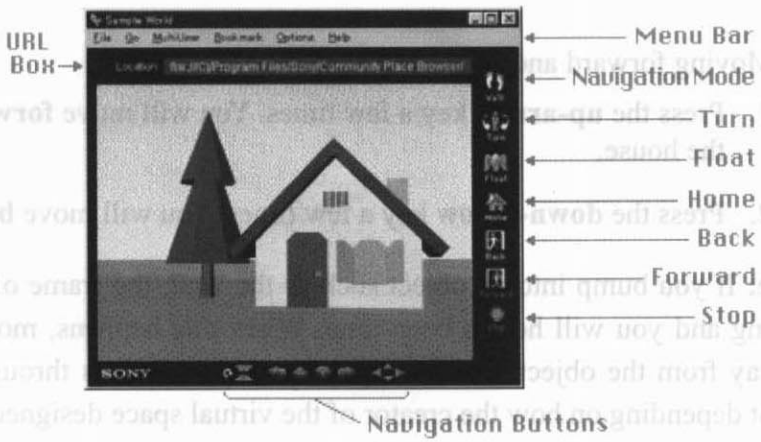


Figure A.1 CommunityPlace Browser

(2) Double-click **cpbrowse.exe** within the Community Place Browser bin directory.

In these cases, the entry room to the VRML world in world folder ("C:¥Program files¥Sony¥Community Place Browser¥world") which level is the same as the folder installed Community Place Browser will be showed.

Quitting Community Place browser

You can terminate the Community Place Browser software using either one of the following methods:

- Click the [Quit] button on the title bar.
- Select **Exit** from the **File** menu.

A.2 Using the Browser

This section explains, in a step-by-step fashion, the basic operation of the Community Place Browser software. This section guides you through a sample world containing a 3D house (see Figure A.1).

Step 1. Traveling through the virtual world (Navigation)

You will see a house within the Browser's window. Let's travel around a little by using **the arrow keys** on the keyboard. You can move right, left, forward, and backward as follows:

(1) Turning left and right.

1. Press the **right-arrow** key a few times. You will turn to the **right.**
2. Press the **left-arrow** key a few times. You will turn to the **left**.

(2) Moving forward and backward.

1. Press the **up-arrow** key a few times. You will move **forward** towards the house.

2. Press the **down-arrow** key a few times. You will move **backward**.

Note: If you bump into an object such as the wall, the frame of the window starts blinking and you will hear a beep tone. When this happens, move backward and stay away from the object. Some objects allow you to pass through them, and others do not depending on how the creator of the virtual space designed them.

If you are more comfortable using the mouse, you may move in the same way by using the mouse.

You can control your movement by dragging the mouse with its **left button** pressed.

(1) To move **forward**, drag the mouse **upward**.

(2) To move **backward**, drag the mouse **downward**.

(3) To move to the **right**, drag the mouse **rightward**.

(4) To move to the **left**, drag the mouse **leftward**.

Operational hints:

(1) When using the keys, you can increase the speed of your movement by keeping the keys pressed. If you are using the mouse, then the distance you drag the mouse controls the speed you move at - try it!

(2) If you get lost while you are traveling around, press the **Home** icon 

, and you will return to your original position.

Step 2. Looking around (Turn)

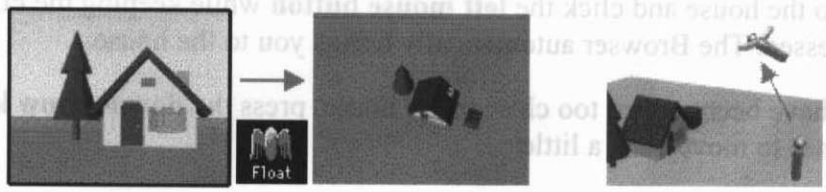
Community Place Browser allows you to easily look around you with a simplified operation. This function is especially helpful when locating the spot where you want to go or just to have a quick check around your area.

(1) Click the **Turn icon** 

on the toolbar. Clicking the person icon at the center will turn you 360 degrees at the current position.

(2) Clicking the left-arrow will turn you 90 degrees leftward at the current position. Clicking the right-arrow will turn you 90 degrees rightward.

Step 3. Taking a bird's-eye view (Float)



While you are traveling around the virtual space, you may get lost. What happens if you get lost in the real world? You might think of looking down from the top of a tall building to find out where you are. Community Place Browser gives you the bird's-eye view of the virtual space you are in. This feature helps you locate where you are and where you want to go.

(4) Click the **Float icon** 

on the toolbar. You will automatically be moved higher and higher. The Float icon will change its form to 

.

(5) When you are taking a bird's-eye view of the virtual world, you can move forward/backward, by pressing the **up-arrow** or **down-arrow key** or using the mouse.

(6) To return to earth you can:

- Returning to your original position

  Click 

again.

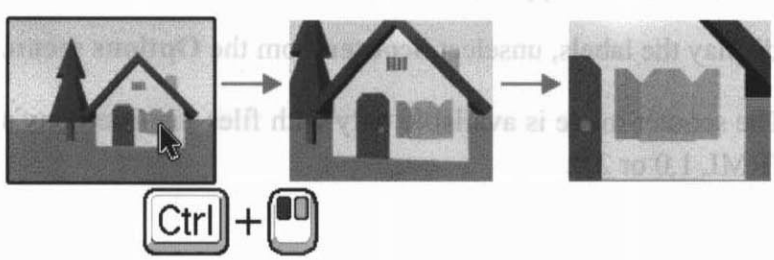
- Return to a new position on the ground:

You can also take a bird's-eye view by using the menu. Select **Float item** from the **Go menu**.

Note: If the virtual space you are in is a room with an object such as a ceiling above you, you are not allowed to move higher than the object.

Step 4. To teleport to a specified object

If you want to see an object that is located far away from you, you may use the normal method of navigation, sometimes through such drag. You can go swiftly to the object, however, by using the following method:



(1) Point to the house and click the **left mouse button** while keeping the **ctrl key** pressed. The Browser automatically brings you to the house.

(2) If you have been moved too close to the house, press the **down-arrow** key a few times to move back a little.

Step 5. Selecting an object

If the cursor happens to be placed on certain objects while you are traveling through the world, it may change to hand cursor. This cursor shows that the object has a pre-programmed "action" in it.

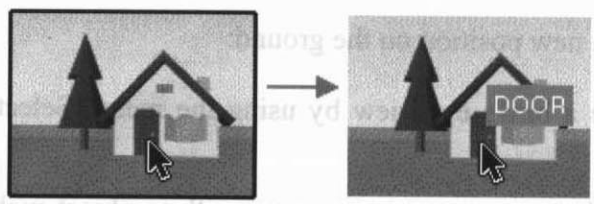
Click the object with the **left mouse button**. The object starts it action. Let's see what happens if you click the red cube to the right of the house.

Place the cursor on the red cube and click with the left mouse button. The red cube starts moving.

1. Clicking it again, the red cube stops moving.

Note: The actions of objects vary depending on how the creators of the virtual world designed them. Clicking an object such as an automobile, may start it moving. An object such as a radio may start playing music and a taxi may take you for a ride.

Step 6. Displaying information (Scouter)



The house displayed on the screen has a label which is not yet visible. You can visualize the label at any time by placing the cursor on an object. This function helps you understand what items exist in the virtual world.

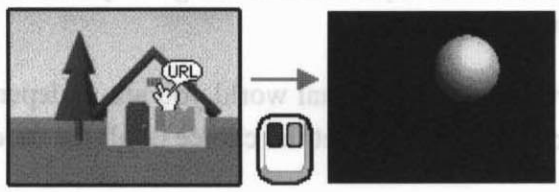
Select Scouter from the Options menu.

(1) Place the cursor on the door.
The "DOOR" label has appeared on the screen.

(2) Not to display the labels, unselect **Scouter** from the **Options menu**.

Note: The scouter mode is available only with files that use Sony's extended functions for VRML 1.0 or 2.0.

Step 7. To warp to another world (Activating a link)



If the cursor happens to point to certain objects while you are traveling around the world, the cursor may change to the hand cursor with "URL". This indicates that objects have links to other virtual worlds. The following function allows you to travel between the worlds.

(1) When you place the cursor over the ventilating opening on the house, the cursor changes to the hand cursor with URL. The link address will be displayed around the bottom of the Community Place Browser window.

(2) Click the **left mouse button**. The link is loaded, and you are transferred to the other virtual world.

(3) To return to the original world, press the **Back** button of Community Place Browser.

Note: Objects on which the cursor changes to the hand cursor with URL are not always linked to other worlds. They can also be linked to a HTML document. In that case, the contents of the HTML document will be displayed in Netscape Navigator.

Step 8. Changing the navigation mode

In the virtual world, you get a choice of navigation modes. You can select your desired navigation mode with the **Navigation Mode button** (each icon represents one of 4 modes) or with the **Navigation Mode** items in the **Options menu.** You can choose one of the following modes:

1. 

Walk mode: You will feel as if you are walking in the real world in this mode. Collisions with the wall will be detected, and your movements will be affected by the force of gravity. It affects only the people who are walking around in the virtual world, and not to the objects within it.

2. 

Examine mode: Objects can be seen from various angles in this mode. You can turn an object by dragging it with the left mouse button pressed.

3. 

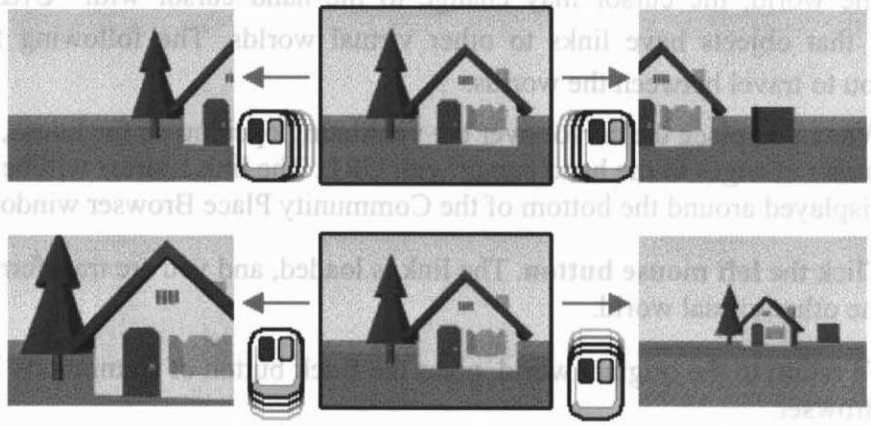
Fly mode: This mode is the same as the Walk mode, except that you are free from the force of gravity.

4. 

None mode: In this mode, you cannot navigate by ordinal user interface such as mouse dragging.

Note: The modes available in the virtual world you are in depend on how its creator designed it. Sometimes you will not get the choice of all the modes.

Moving with the mouse



Drag the mouse with its **left button** pressed to a desired direction.

(1) To move **forward**, drag the mouse **upward**.

(2) To move **backward**, drag the mouse **downward**.

(3) To turn **right**, drag the mouse to the **right**.

(4) To turn **left**, drag the mouse to the **left**.

Additional functions

Community Place Browser supports the following additional functions.

(1) Looking up/down: You can look up or down by using the  button located on the left end of the navigation bar. To look up, click the upper button. To look down, click the lower button. To return to the original head position, click the whirl to the left.

(2) Moving up/down/right/left: You can move up/down/right/left by using the  buttons located on the right end of the navigation bar. Clicking the buttons, you can move in the direction indicated by the arrows.

A.3 Multi-user Functions

Community Place Browser provides additional multi-user functions, which enable you to interact with other people who watch the same virtual world as you do. These functions, however, can be used only in the virtual spaces which have been designed for use by multiple users. **To use the multi-user functions, your computer must be connected to the Internet.**

In the virtual world designed for multiple users, you can still use the movement functions (e.g. traveling around in the space or moving swiftly towards an object) as explained in the earlier sections. The only difference is that the Avatars (representations) of other users will also be traveling around the same shared world.

Certain objects have pre-programmed "actions" in the shared virtual world, too, however in many cases, they are shared by multiple users. For example, the "Circus Park", the shared virtual space presented by Sony, may change the colors and shapes of its objects even if you never touch them. This shows that other users in the same world clicked the objects from their browser, and you share the same view with others!

Entering a shared virtual world

You can enter a shared virtual world in the same way as described in the earlier sections. You don't need to do anything special. If the CP browser sees that this is a shared world it will automatically connect to a multi-user Bureau.

If you download a shared virtual world, the icon  appears on the lower right of the Browser window (Figure A.2), and the Multi-User window appears on the screen. You will also see the "Connection to Community Place Bureau Complete" message scroll across the screen.

All multi-user functions for shared virtual spaces can be obtained in the Multi-User window and from the Multi-User menu. You can also display the Multi-User window by selecting the **Multi-User Window item** from the **Multi-User menu** of Community Place Browser.

Note: The Multi-User window will not appear if you fail to enter the virtual world. This will happen, for example, if your machine is not connected to the Internet. When you have failed to enter, the multi-user functions are not available for use. However, you are still allowed to travel around the shared virtual space, activate the pre-programmed "actions" and so forth. Only the difference is that there you will never come across other Avatars. That is, the space you are in is no longer a shared virtual world but is a simple virtual world.

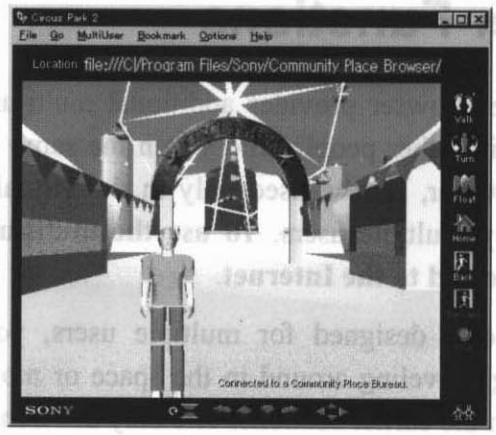


Figure A.2 What the shared virtual space looks like.

Chat function

The chat function in the Multi-User window enables you to easily send text messages to other users.

Type a message in the text input field at the bottom of the Multi-User window. Pressing the Return key, the message will be sent to all the users in your proximity. If there are other users chatting besides you, their messages will be displayed in your Multi-User window.

The message you have typed will be displayed in the Browser's main window as "Text Balloon", above your avatar. It will also be displayed in the Multi-User window. You cannot see your "Text Balloon" message because it is

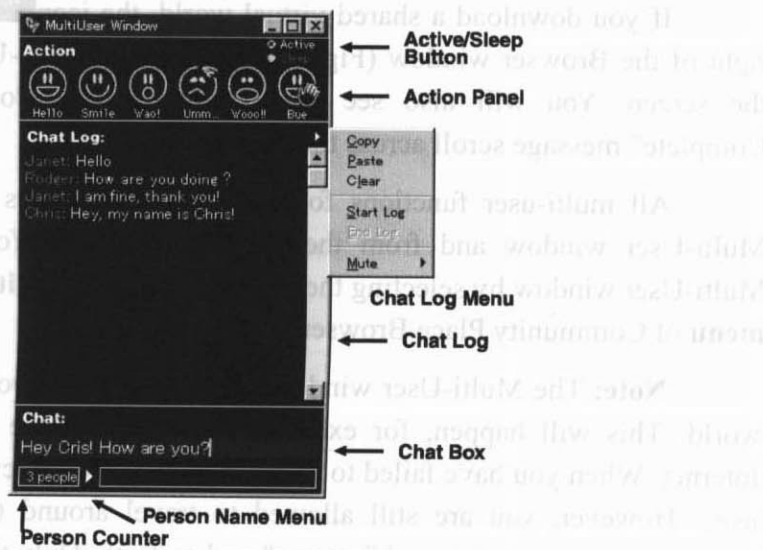


Fig: Chat function

above your head, but can see the messages from others in their Text balloons.

You can store and edit the chat log by using the chat log menu. The menu can be displayed by pressing the triangle button on the right corner of the Multi-User window.

Fig: Chat log menu

Mute function

The mute menu enables you to specify users whose messages are not displayed in your browser. If you are annoyed with some users' messages, you can use this function.

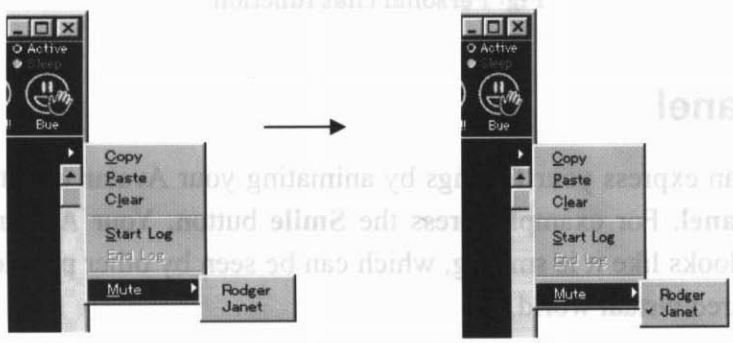


Fig : Mute function

Your message can only be seen by other users who are in your proximity. This has two advantages.

(1) First, this avoids the need to display all the messages received from all over the world, in your chat window. This would be confusing in a crowded world.

(2) Second, if you want to chat in privacy with a few people, all you have to do is to go to a quiet corner and stay away from other users. Now you can rely on Community Place Browser to keep your privacy.

If other users come close to you, however, they can hear you. To avoid this, use the personal chat function.

Personal chat function

While keeping the **Shift key** pressed, place the cursor on the avatar with whom you wish to chat privately. If the cursor changes to look like , you can use the personal chat function with him/her. He or she might, however, turn down your offer.

The personal chat request will pop up a small chat window which is used to display your one on one chat messages. You can also store and edit the personal chat log by using the chat log menu.

Fig: Personal chat function

Action panel

You can express your feelings by animating your Avatar with the buttons on the **Action panel**. For example, press the **Smile** button. Your Avatar will start an "action" that looks like it is smiling, which can be seen by other people who can see you in the shared virtual world.

Note: The actions caused by pressing the buttons will vary depending on how the creator designed the world.

Displaying the radar

To locate other users in the shared virtual space, it is recommended that you use the radar. Select the **Radar Map** item from the **Multi-User** menu. A cross-shape

Fig: Displayed radar

radar appears on the upper right of the screen. You are just at the center of the cross. The others users around you are indicated with red dots.

Displaying the names of other users

You can list the names of other people around you within the shared virtual space. Click the button with the triangle mark located at the bottom of the Multi-User window, and all the names of people near you will be displayed. By selecting a name, you will automatically be teleported to the person with that name.

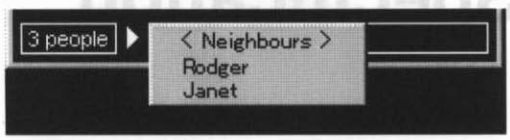


Fig: Listing the names of other users in the shared virtual world

The number displayed on the right side of the button indicates the total number of users within the shared virtual world. Only the names of people near you will be displayed in the list. This means that the number of people sharing the world, and the number of names listed are not necessarily the same.

Modifying the avatar (Representation)

Select the **Select Avatar ...** item from the **Multi-User** menu of Community Place Browser. All the available Avatars will be listed. Select your desired Avatar and click the [OK] button. The selected Avatar now represents you and will be seen in other users' browsers. You can also change its color by using **Edit Avatar...** within the **same menu**.

Note: You may not be able to change the Avatar's color in some virtual worlds. Again, it depends on how the creators designed the world.

Automatic disconnect function

If you don't move, or chat, for a certain period of time, the connection from your computer to the multi-user server will be automatically disconnected (10 minutes with the Circus Park). If you are away from the computer for a while, and want to stay connected, choose **Sleep** from the **[Activate/Sleep] button** located on the upper right of the Multi-User window. Press **[Activate]** when you return. In the Circus Park, for example, if you click the [Sleep] button, your Avatar appears to be sitting down to the other uses. If other Avatars are sitting, you will know that their [Sleep] buttons were pressed. Even with the [Sleep] button pressed, the connection will automatically be disconnected for the Circus Park if you do not interact with the space for 60 minutes.

Appendix B.
E-VRML Specification

B.1 Introduction

Following is a list of nodes that constitute our extensions to the current VRML 1.0 standard. We extended VRML 1.0 about scripting, object attributes and sound features.

There are seven extension nodes for it: Script node, EventHandler node, SporadicTask node, PeriodicTask node, CalendarTask node, Attributes node, and AmbientSound node. Notice that "fields" field is necessary in extension nodes but it is removed from the following descriptions for simplifying.

B.2 Scripting

There are seven extension nodes for it: Script node, EventHandler node, SporadicTask node, PeriodicTask node, and CalendarTask node.

Requirement

- Syntactic extensions to VRML: syntactic extensions are necessary for defining functions and registering tasks.
- Basic task support: VRML's scripting nodes should support the following basic tasks: event handler, periodic task, sporadic task and calendar task.
- Naming 3D objects and accessing methods to them from scripts
- Libraries for manipulating 3D objects

- Multiple scripting language support: there are many script languages in the world: e.g. TCL, python and java. VRML's scripting nodes should be language independent.
- Inline scripting is necessary.

Syntactic extension

Five node types are necessary for the scripting extension to VRML: Script node, EventHandler node, SporadicTask node, PeriodicTask node and CalendarTask node. The first one is for defining functions; others are for registering them to objects.

Defining functions

This node defines a procedure for EventHandler nodes, SporadicTask nodes, PeriodicTask nodes and CalendarTask nodes. This node does not affect any other nodes. Script node can specify inline script in the procedure field. More than one script function can be specified in the field.

```
FILE FORMAT/DEFAULT

Script {
    procedure ""          # SFString
    scriptType NONE       # SFEnum
}

EXAMPLE

Script {
        procedure       "proc change_color { obj event userData } {¥n¥
                                vsSetObjDiffuse $obj $userData;¥n¥
                        }
                        proc move { obj event userData } {¥n¥
                                vsObjTranslate $obj 10 0 0;¥n¥
                        }"      # SFString
        scriptType TCL          # SFEnum
}

SCRIPT TYPE ENUM
        TCL, PYTHON, JAVA, VB...
```

B.3 Registering Tasks

EventHandler node

This property node defines an event handler. It is called back from a browser when an event that matches the eventType occurs on the object.

```
filename:   specifies URL of a script file. The script file can contain
                more than one script function. If the URL is empty string,
               "function" name is searched from Script node.
eventType:  specifies the event type for which to call the handler
userData:   specifies additional data to be passed to the event handler
function:   specifies the function name that is to be added or
               simple method(see below).
scriptType:  specify the script type

FILE FORMAT/DEFAULT

EventHandler {
        filename ""          # SFString
        eventType NOEVENT     # SFBitMask
        userData ""          # SFString
        function ""          # SFString
        scriptType INLINE     # SFEnum
}

EXAMPLE

EventHandler {
        filename "change.tcl"   # SFString
        eventType GRAB         # SFBitMask
        userData "red"         # SFString
        function "change_color" # SFString
        scriptType TCL         # SFEnum
}

EVENT TYPE MASK
        GRAB, RELEASE, COLLISION_IN, COLLISION_OUT, WORLD_IN, WORLD_OUT,
        KEY_DOWN, KEY_UP...

SCRIPT TYPE ENUM
        INLINE, C, PYTHON, TCL, JAVA, VB...
```

About "scriptType", see "Multiple scripting languages example". The script type INLINE can be used for describing a simple method directly. The script type "C" can be used only for specifying the built-in C libraries (see B.8).

The EventHandler procedure is called back from a browser in the following form:

```
void EventHandler(VsObj *obj, VsEvent *event, VsString userData);

obj: is specified the object which has the event handler.
event: is specified the event which is occurred. It has the
         following information.
        - the object on which the event has occurred. This object
               could be different from the "obj".
        - event type
        - key or button
        - modifier
        - (x, y, z)
        - time stamp
        - event type dependent information
          In case of COLLISION_IN/COLLISION_OUT event, the object
           with which "obj" collides is specified. In case of
           WORLD_IN/WORLD_OUT, information about the world is specified.
userData: is specified the data which is specified in the
        "userData" field in an EventHandler node.
```

In this example, the "change_color" function is defined in the Script nodes.

SporadicTask node

This node defines a sporadic task. It is called back from a browser when the specified time expires, and the task is removed. This node has five fields: filename, millisecond, userData, function, and scriptType. The "millisecond" field specifies the time when the task is executed.

```
FILE FORMAT/DEFAULT

SporadicTask {
        filename ""             # SFString
        millisecond 100         # SFLong
        userData ""             # SFString
        function ""             # SFString
        scriptType INLINE       # SFEnum
}

EXAMPLE

SporadicTask {
        filename "change.tcl" # SFString
        millisecond 100        # SFLong
        userData "red"         # SFString
        function "change_color" # SFString
        scriptType TCL         # SFEnum
 }
```

The SporadicTask procedure is be called back from a browser in the following prototype:

void SporadicTaskProc(VsObj *obj, VsEvent *event, VsString userData);

PeriodicTask node

This node defines a periodic task. It is called back from a browser periodically when the specified time expires. This node has the same fields as SporadicTask node has. The task is stopped when the function returns TRUE.

```
FILE FORMAT/DEFAULT

PeriodicTask {
        filename ""             # SFString
        millisecond 100         # SFLong
        userData ""             # SFString
        function ""             # SFString
        scriptType INLINE       # SFEnum
}

EXAMPLE

PeriodicTask {
```

```
        filename "change.tcl"   # SFString
        millisecond 100         # SFLong
        userData "red"          # SFString
        function "change_color" # SFString
        scriptType TCL          # SFEnum
}
```

The PeriodicTask procedure is be called back from a browser in the following prototype:

```
VsBoolean PeriodicTaskProc(VsObj *obj, VsEvent *event,
                                       VsString userData);
```

CalendarTask node

This node defines a calendar task. It is called back from a browser when the specified date comes, and the task is removed. This node has five fields: filename, date, userData, function and scriptType. The "date" field specifies the date when the task is executed (e.g. "Jun 15 15:51:16 2995").

```
date:  specify the date. e.g. "Jun 15 15:51:16 2995"

FILE FORMAT/DEFAULT

CalendarTask {
        filename ""                     # SFString
        date ""                         # SFString
        userData ""                     # SFString
        function ""                     # SFString
        scriptType INLINE               # SFEnum
}

FILE FORMAT/EXAMPLE

CalendarTask {
        filename "change.tcl"           # SFString
        date "Jun 15 15:51:16 2995"     # SFString
        userData "red"                  # SFString
        function "change_color"         # SFString
        scriptType TCL                  # SFEnum
}
```

The CalendarTask procedure is be called back from a browser in the following prototype:

```
void CalendarTaskProc(VsObj *obj, VsEvent *event, VsString userData);
```

Method

A tiny language can be embedded in the "function" field of EventHandler node, SporadicTask node, PeriodicTask node, and CalendarTask node directly to describe some simple methods. The methods also are invoked by events or messages according to the node type. For example,

```
EventHandler {
```

```
        eventType GRAB
        function "set diffuseColor 1 0 0"
}
Cube {}
```

In this example, when user clicks the cube, its diffuse color is changed to red. It is not necessary to specify scriptType, because INLINE is default script type. Only the following syntax can be used for describing method.

- set RESERVED-WORD CONSTANT

  RESERVED-WORD:
  diffuseColor, transparency, ambientColor...

- load URL

B.4 Object Attributes

Attributes node

Attributes node defines the object name and its attributes. The "name" can be accessed only in scripts. Although VRML1.0 supports DEF keyword for naming node, the names do not have to be unique. It is difficult to use for accessing objects in scripts.

The name must be unique between top objects and its sub objects.

```
name:       object name
grasp:      possible to grasp object
solid:      determines if the object is passable.
collision:  determines if a collision check should be performed
             when the object moves.
mobile:     determines if object is static
backface:   backface is rendered
gouraud:    gouraud shading
propagateMask: determines if event is propagated to the object's  parent
assignId:   determines if id is assigned. If id is assigned to an object,
                it is shared among all users under multi-user environment.

FILE FORMAT/DEFAULTS

Attributes {
        name ""                 # SFString
        visible TRUE            # SFBool
        grasp TRUE              # SFBool
        solid TRUE              # SFBool
        collision TRUE          # SFBool
        mobile TRUE             # SFBool
        backface TRUE           # SFBool
        gouraud TRUE            # SFBool
        propagateMask all-one   # SFBitMask
        assignId FALSE          # SFBool
}
```

Note: Solid, "backface" and "gouraud" fields should be the fields of ShapeHint. "assigned" should be "shared".

B.5 Node Scope

The above nodes affect to the object which comes after it in the declaration order. In the following example, the second cube does not have event handler.

```
Separator {
        Separator {
                Attributes {
                        name "foo"
                }
                EventHandler {
                        filename "change.tcl"
                        eventType GRAB
                        userData "red"
                        function "change_color"
                        scriptType TCL
                }
                Cube {} # This cube is named "foo" and has the
                        # event handler.
                Cube {} # This cube has no name and no event handler.
        }
        Cube {} # This cube has no name and no event handler.
}
```

B.6 AmbientSound Node

This property node defines an ambient sound source. If user enters the bounding cube of it, it plays sound automatically and adjusts its volume according to a distance between the node and user.

```
filename:       specify the sound data(URL).
                .wav    Windows WAVE (PCM)
                .mid    Windows MIDI
bboxSize, bboxCenter:    specify the bounding cube.
autoPlay:       specify if the sound plays automatically when user enters
                 the bounding cube.
loop:           auto repeat

FILE FORMAT/DEFAULT

AmbientSound {
        filename    ""          # SFString
        bboxSize    0 0 0       # SFVec3f
        bboxCenter  0 0 0       # SFVec3f
        autoplay    TRUE        # SFBool
        loop        TRUE        # SFBool
}
```

B.7 Scripting Examples

In the following examples, we suppose that content of "change.tcl" likes this:

```
Separator {
        Separator {
                Attributes {
                        name "foo"
                }
                EventHandler {
                        filename "change.tcl"
                        eventType GRAB
                        userData "red"
                        function "change_color"
                        scriptType TCL
                }
                Cube {} # This cube is named "foo" and has the event handler.
        }
        Cube {} # This cube has no name and no event handler.
}
```

Script sharing example

In the following example, two event handlers share the same script.

```
Separator {
        Separator {
                EventHandler {
                        filename "change.tcl"
                        eventType GRAB
                        userData "red"
                        function "change_color"
                        scriptType TCL
                }
                Cube {} # This cube has the event handler to change its
                        # color to red.
        }
        EventHandler {
                filename "change.tcl"
                eventType GRAB
                userData "blue"
                function "change_color"
                scriptType TCL
        }
        Cube {} # This cube has no name but has event handler to change
                # its color to blue.
}
```

Attaching two event handlers example

In the following example, two event handlers are attached to the same cube.

```
Separator {
        Separator {
                EventHandler {
                        filename "change.tcl"
                        eventType GRAB
                        userData "red"
                        function "change_color"
```

```
                scriptType TCL
        }
        EventHandler {
                filename "change.tcl"
                eventType RELEASE
                userData "blue"
                function "change_color"
                scriptType TCL
        }
        Cube {} # This cube has the two event handlers to change
                its color.
    }
}
```

When user presses button, the color of cube is changed to red, then button is released, that of cube is changed to blue.

Multiple scripting languages support example

In the following example, the first cube has TCL event handler and the second cube has PYTHON event handler to change their color.

```
Separator {
    Separator {
        EventHandler {
                filename "change.tcl"
                eventType GRAB
                userData "red"
                function "change_color"
                scriptType TCL
        }
        Cube {} # This cube has the TCL event handler
                # to change its color to red.
    }
    EventHandler {
            filename "change.py"
            eventType GRAB
            userData "blue"
            function "change_color"
            scriptType PYTHON
     }
     Cube {} # This cube has PYTHON event handler to change its color
            # to blue.
}
```

Built-in C function example

If a browser has a built-in function "change_color", an event handler can be specified like the following:

```
Separator {
    EventHandler {
            eventType GRAB
            userData "red"
            function "change_color"
            scriptType C
    }
    Cube {} # This cube has the built-in C event handler
            # to change its color to red.
}
```

Multiple events example

In the following example, a cube has the same event handler for GRAB and KEY_DOWN.

```
Separator {
        EventHandler {
                filename "change.tcl"
                eventType ( GRAB | KEY_DOWN )
                userData "red"
                function "change_color"
                scriptType TCL
        }
        Cube {} # This cube has the event handler for two events.
}
```

Inline scripting example

```
Separator {
        Script { # defines script procedure.
                procedure
                        "proc change_color { obj event userData } {¥n¥
                                vsSetObjDiffuse $obj $userData;¥n¥
                        }"
                scriptType TCL  # SFEnum
        }
        EventHandler {
                filename ""
                eventType GRAB
                userData "red"
                function "change_color"
                scriptType TCL
        }
        Cube {} # This cube has the handler for GRAB event.
}
```

Appendix C. VRML

Virtual Reality Modeling Language (VRML) is 3D file format designed to create 3D virtual worlds for delivery over the Internet. The language used for text documents in the Internet is Hyper-Text Markup Language (HTML). VRML can be thought as a 3D version of HTML and is being standardized by the International Standards Organization (ISO). The latest version of the standard is know as VRML2.0 (VRM97[1]) and has been adopted by most major VRML browser and tool vendors.

VRML2.0 described 3D worlds from a series of transformation node, i.e. spatial position marker. Each node can contain sub-nodes forming tree (called "scene graph"). VRML2.0 provides a set of geometry nodes (box, sphere, etc.), property nodes (color, texture, etc.), grouping nodes and special nodes (camera, light, etc.). In addition, it supports multimedia nodes which enable ambient sound or spot sound and video. Each node type has a fixed set of fields, i.e. attribute of the node. Fields may contain various kinds of data and one or many values. Each field has a default value.

Figure C.1 shows the example of a red sphere. The first line indicates the version of VRML. Shape node (line 4) defines a 3D object (red sphere). It supports two fields for describing its attributes, i.e. attributes of the object; "appearance" field (line 5) and "geometry" field (line 8). In this example, the geometry field specifies Sphere node to describe a sphere. The appearance field describes its color (i.e. red, "1 0 0" in RGB format) by using Material node (line 6). Transform node (line 2) defines the transformation of the object described by the Shape node.

VRML2.0 also adds a mechanism to associate a programming language fragment with 3D scene objects. The mechanism consists of four sub-mechanisms;

---

[1] ISO version of VRML2.0

```
 1 #VRML V2.0 utf8
 2 Transform {
 3    children [
 4       Shape {
 5          appearance Appearance {
 6             material Material { diffuseColor 1 0 0 }
 7          }
 8          geometry Sphere {}
 9       }
10    ]
11 }
```

Figure C.1 A red sphere

sensor, event, routing and script. The sensor is the mechanism to detect external events and convert them to the internal VRML events. For example, VRML2.0 supports a TouchSensor to detect user's mouse operation to 3D objects associated with the sensor and generate internal VRML events. The event is the mechanism that passes information between nodes and goes so by using a routing mechanism. The target for routing events may be graphical nodes, or more typically, script nodes. When the events are routed to a script node, the associated program is executed and passed an events send to the script node. The program can then generate new events that are passed back to the script node. These events may then be routed to other nodes and change the scene graph (for example, changing color of 3D objects or moving them). Currently either Java or JavaScript is available as a scripting language.

Figure C.2 and Figure C.3 show the example which changes the color of sphere from red to blue by using Java as a script language. Figure C.2 describes a

```
 1 #VRML V2.0 utf8
 2 Transform {
 3    children [
 4       DEF TS TouchSensor {}
 5       Shape {
 6          appearance Appearance {
 7             material DEF SphereColor Material { diffuseColor 1
0 0 }
 8          }
 9          geometry Sphere {}
10       }
11    ]
12 }
13 # Script node
14 DEF ChangeColor Script {
15    url "ChangeColor.class"
16    eventIn SFBool clicked
17    eventOut SFColor newColor
18 }
19 # Routing
20 ROUTE TS.isActive TO ChangeColor.clicked
21 ROUTE ChangeColor.newColor TO SphereColor.set_diffuseColor
```

Figure C.2 A red sphere with script

world and Figure C.3 shows the script bound to the world.

In FigureC.2, "DEF" defined the name of a node (line 4) and "ROUTE" defines a route for passing event (line 20). When the user clicks the sphere, the TouchSensor (line 4) detects the event and then generates a VRML event from the isActive field of the TouchSensor. This event is routed to clicked field of Script node (line 16) according to the specified routing (line 20). The Script node binds the red sphere to ChangeColor.class (Figure C.3) by using its url field (line 15). Then the event is passed to ChangeColor.class. After completing some calculation in the program, it generates an event (color) on newColor field of the Script node (line 17). It is routed to diffuseColor field (line 21). Consequently, the sphere's diffuse color is changed.

When the event is passed to ChangeColor.class, processEvent() method (line 16) is called with the event as an argument. In the method, a new color is set to the reference to newColor field. The reference is gotten using getEventOut() in initialize() method (line 13) that is called before any event is generated.

In addition to the VRML2.0's behavior mechanism, the CommunityPlace system supports Java classes to enable to build multi-user applications. Its fundamental function is to send messages to the CommunityPlace Bureau using VSCP. The messages from other browsers appear on Script node's eventIn field and it is also passed to the processEvent() method.

```
 1 // ChangeColor.java
 2 import vrml.*;
 3 import vrml.field.*;
 4 import vrml.node.*;
 5
 6 public class ChangeColor extends Script{
 7     private boolean on = false;     // status of on/off
 8     float red[] = { 1, 0, 0 };      // RGB(Red)
 9     float blue[] = { 0, 0, 1 };     // RGB(Blue)
10     private SFColor newColor;       // reference to "newColor"
                                       //  field of Script node
11
12     public void initialize(){
13         newColor = (SFColor) getEventOut("newColor");
14     }
15
16     public void processEvent(Event e) {
17         ConstSFBool v = (ConstSFBool)e.getValue();
18
19         if(v.getValue()){
20             if (on) {
21                 newColor.setValue(red); // set red to 'newColor'
22             } else {
23                 newColor.setValue(blue); // set blue to 'newColor'
24             }
25             on = !on;
26         }
27     }
28 }
```

Figure C.3 Script to change a color of sphere

Bibliography

Asimov, I. (1994), I, Robot, Bantam Books, ISBN 0553294385.

Bajura, M., Fuchs, H., and Ohbuchi, R. (1992), "Merging virtual objects with the real world: seeing ultrasound imagery within the patient", in Proceedings of ACM SIGGRAPH Computer Graphics, Vol.26, No. 2, pp. 203-210.

Barrus, J., Waters, R., and Anderson, D. (1996), "Locales: Supporting Large Multiuser Virtual Environments", IEEE Computer Graphics and Applications, No. 16, Vol. 6, pp. 50-100.

Bartle, R.(1990), "Interactive multi-user computer games", Technical report, British Telecom. Electronic version available at http://www.mud.co.uk/richard/imucg.htm.

Bates, J. (1997), "The Role of Emotion in Believable agents, Communications of the ACM", No. 37, Vol. 7, pp. 122-125.

Becker, B., and Mark, G. (1998), "Social conventions in collaborative virtual environments", in Proceedings of Collaborative Virtual Environments 1998 (CVE '98), pp. 47-56.

Benedens, O. (1999). "Geometry-based watermarking of 3d models", IEEE Computer Graphics and Applications, pp. 46-55.

Benedikt, M.(ed) (1991), Cyberspace: First Steps. Cambridge, MA: MIT Press, ISBN 0-262-52177-6.

Benford, S., and Fahlen, L. (1993), "A spatial model of interaction in large virtual environments", in Proceedings of the 3rd European Conference on Computer Supported Cooperative Work (ECSCW'93), pp. 109-124.

Benford, S., Fahlen, L., Greenhalge, C., and Bowers, J. (1994), "Managing Mutual Awareness in Collaborative Virtual Environments", in Proceedings of ACM SIGCHI conference on Virtual reality and technology (VRST'94), pp. 223-236.

Benford, S., and Greenhalgh, C. (1997), "Introducing Third Party Objects into the Spatial Model of Interaction", in Proceedings of European Conference on Computer Supported Cooperative Work, pp. 189-204.

Billinghurst, M., and Savage, J. (1996), "Adding Intelligence to the Interface", in Proceedings of IEEE Virtual Reality Annual International Symposium, pp. 168-175.

Bowers, J., Pycock, J., and O'Brien, J. (1996), "Talk and Embodiment in Collaborative Virtual Environments", in Proceedings of Computer human interactions 1996 (CHI '96), pp. 58-65.

Bruckman, A. (1997), "MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids", PhD Dissertation, MIT.

Bruckman, A., Curtis, P., Figallo, C., and Laurel, B., (1994) "Approaches to managing deviant behavior in virtual communities", Panel presented at CHI 94, pp. 183-184.

Carlsson, C., and Hagsand, O. (1993), "DIVE - A platform for multi user virtual environments", Computer and Graphics, Vol. 17, No. 6, pp. 663-669.

Chavez, A., and Maes, P. (1996), "Kasbah: An Agent Marketplace for Buying and Selling Goods", in Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, pp. 75-90.

Cherny, L. (1995), "The Mud Register: Conversational Modes of Action in a Text-Based Virtual Reality", Ph.D. dissertation, Stanford University.

Cherny, L. (1999), Conversation and Community: Chat in a Virtual World, CLI Publications, ISBN1-575-86154-2.

Churchill, E., and Bly, S. (1999), "Virtual Environments At Work: ongoing use of MUDs in the Workplace", in Proceedings of Work Activities Coordination and Collaboration (WACC'99), pp. 99-108

Craven, M., Benford S., Greenhalgh, C., Wyver J., Brazier C., Oldroyd, A., and Regan, T. (2000), "Ages of avatar: community building for inhabited television", in Proceedings of Collaborative virtual environments 2000 (CVE2000), pp. 189-194.

Cruz-Neira, C., Sandin, D., and DeFanti, T. (1993), "Surround-screen projection-based virtual reality: The design and implementation of the CAVE", in Proceedings of SIGGRAPH '93, pp. 135-142.

Curtis, P. (1996), Mudding: Social Phenomena in Text-Based Virtual Realities, Internet Dreams: Archetypes, Myths, and Metaphors, MIT Press, pp. 265-292.

Damer, M. (1998), Avatars! Exploring and Building Virtual Worlds on the Internet, CA: Peachpit Press, ISBN0-201-68840-9.

Dell Computer Corp. (1999), Fiscal 1999 in Review, Annual Report, Round Rock: http://www.dell.com/html/us/corporate/annualreports/report99/Dell_1999_Fiscal_in_Review.pdf.

Dibbell, J. (1994), "A rape in cyberspace: or how an evil clown, a Haitian trickster spirit, two wizards and a cast of dozens turned a database into a society", in M. Deny (ed) Flamewars: the Discourse of Cyberculture, London, Duke University Press, pp. 237-261.

DiPaola, S., and Collins, D. (1999), "A 3D Natural Emulation Design Approach to Virtual Communities", in SIGGRAPH99 Conference Abstracts and Applications, p. 208.

Doyle, P., and Hayes-Roth, B. (1997), "An Intelligent Guide for Virtual Environment", in Proceedings of Animated Interface Agent: Making them Intelligent, pp. 77-84.

Evard, R. (1993), "Collaborative Networked Communication: MUDs as Systems Tools", in Proceedings of USENIX Seventh System Administration Conference (LISA '93), pp. 1-8.

Ellis, C., Gibbs, S. J., and Rein, G. (1988), "Design and use of a group editor", Technical report STP-263-88, MCC.

Ellis, C., Gibbs, S. J., and Rein, G. (1991), "Groupware: some issues and experiences", Communication of the ACM, Vol. 34, No. 1, pp. 38-58.

Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pedersen, D., Pier, K., Tang, J., and Welch, B. (1992), "Liveboard: A large interactive display supporting group meetings, presentations and remote collaboration", in Proceedings of International Conference on Human Factors in Computing Systems 1992 (CHI '92) , pp. 599-607.

Fahlen, L. E., and Brown, C.G. (1992), "The Use of a 3D Aura Metaphor for Computer Based Conferencing and Teleworking", in Proceedings of the 4th Multi-G Workshop, pp. 69-74.

Farmer, F. (1994), "Social Dimensions of Habitat's Citizenry", The Virtual Reality Casebook. C. E. Loeffler, and T. Anderson, (eds.) New York: Van Nostrand Reinhold.

Foner, L. (1997), "Entertaining Agents: a sociological case study", in Proceedings of the First International Conference on Autonomous Agents 1997 (Agents '97), pp. 122-129.

Frecon, E., and Stenius, M. (1998), "DIVE: A scalable network architecture for distributed virtual environments", Distributed Systems Engineering Journal, Vol. 5, Special Issue on Distributed Virtual Environments, pp. 91-100.

Fujita, M., Kitano, H., and Kageyama, K. (1998), "Reconfigurable Physical Agents", in Proceedings of the 2nd International Conference on Autonomous Agents, pp. 54-61.

Funkhouser, T. A. (1995), "RING: A Client-Server System for Multi-User Virtual Environments. Computer Graphics", in Proceedings of SIGGRAPH Symposium on Interactive 3D Graphics, pp. 85- 92.

Gibson, W. (1984), Neuromancer, New York : Ace Books, ISBN 0441569595.

Greenhalgh, C., and Benford, S. (1999), "Supporting Rich and Dynamic Communication In Large Scale Collaborative Virtual Environments", Presence: Teleoperators and Virtual Environments, Vol. 8, No. 1, pp. 14-35.

Greenhalgh, C., and Benford, S. (1995), "MASSIVE: a Distributed Virtual Reality System Incorporating Spatial Trading", in Proceedings of IEEE the 15th International Conference on Distributed Computing Systems (ICDCS '95), pp. 27-34.

Greenhalgh, C., Purbrick, J., and Snowdon, D. (2000), "Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring", in Proceedings of Collaborative Virtual Environment 2000 (CVE2000), pp. 119-127.

Hartman, J., and Wernecke, J. (1996), The VRML2.0 handbook, MA: Addison Wesley Developers Press, ISBN 0-201-47944-3.

Hirooka, Y., and Tsunematsu, N. (1998), "Analysis of membership attrition process on a WWW service", Virtual Reality Society of Japan Research Report, Vol. 2, No.3, pp. 31-36 (in Japanese).

Honda, Y., Matsuda, K., Rekimoto, J., and Lea, R. (1995), "Virtual society: Extending the WWW to Support a Multi-User Interactive Shared 3D Environment", in Proceedings of Virtual Reality Modeling Language 1995 (VRML '95), pp. 109-116.

Inoue, M., Murakami K., Kiyosue, Y., and Ishibashi, S. (2000), "A Study on the Organized Process of Communities and the Usage of Media in the Three Dimensional Cyber Society "InterSpace"", Vol. 41, No. 10, pp. 2670-2677 (in Japanese).

Isbister, K. (2000), "A Warm Cyber-Welcome: Using an Agent-led Group Tour to Introduce Visitors to Kyoto", in Digital Cities: Experiences, Technologies and Future Perspectives, eds. T. Ishida and K. Isbister, Lecture Notes in Computer Science 1765, pp. 397-406, London: Springer-Verlag.

Isbister, K., Nakanishi, H., Ishida, T., and Nass C. (2000), "Helper Agent: Designing an Assistant for Human-Human Interaction in a Virtual Meeting Space", in Proceedings of International Conference on Human Factors in Computing Systems 2000 (CHI2000), pp. 57-64.

Ishida, T., and Isbister, K. (Eds.)(2000), Digital Cities: Experiences, Technologies and Future Perspectives, Lecture Notes in Computer Science 1765, London: Springer-Verlag, ISBN3-540-67265-6.

Jarvenpaa, L., and Todd, P.A. (1997), "Consumer reactions to electronic shopping on the World Wide Web", International Journal of Electronic Commerce, Vol. 1, No. 2, pp. 59-88.

Johnson, L., Rickel, J., Stiles, R., and Munro, A. (1998), "Integrating Pedagogical Agents into Virtual Environments", Presence: Teleoperators and Virtual Environments Vol. 7, No. 6, pp. 523-546.

Kadobayashi, R., and Mase, K. (1998), "Seamless Guidance by Personal Agent in Virtual Space Based on User Interaction in Real World", in Proceedings of the Third International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM98), pp. 191-200.

Kalakota, R., and Whinston, A. (1997), Electronic Commerce: A Manager's Guide, MA: Addison Wesley Longman Inc, ISBN 0-201-88067-9.

Kelling, L. G., Coles, M. C., Wilson, Q. J. (1997), Fixing Broken Windows: Restoring Order and Reducing Crime in Our Communities, Touchstone Books, ISBN 0-684-83738-2.

Koike, H. (1997), "Communication & Shopping town Machiko", Virtual Reality Society of Japan Research Report, Vol. 1, No.1, pp. 25-30.

Leach, E. R. (1976), Culture and Communication, Cambridge University Press, ISBN 0-521-29052-X.

Lee, J., Kim, J., and Moon J. Y. (2000), "What makes internet users visit cyber stores again? key design factors for customer loyalty", in Proceedings of International Conference on Human Factors in Computing Systems 2000 (CHI 2000), pp. 305-312.

Leland, M. D. P., Fish, R. S., and Kraut, R. E. (1998), "Collaborative document production using Quilt, in Proceedings of CSCW '88, pp. 206-215.

Lohse, G. L., and Spiller, P. (1998), "Electronic shopping: The effect of customer interfaces on traffic and sales", Communications of the ACM, Vol. 41, No. 7, pp. 81-87.

Macedonia, M., and Brutzman, D. (1994), "MBone Provides Audio and Video Across the Internet", IEEE Computer, Vol.27 No. 4, pp. 30-36.

Macedonia, M., and Zyda, M. (1997), "A taxonomy for networked virtual environments", IEEE Multimedia, Vol. 4, No. 1, pp. 48-56.

Macedonia, M., Zyda, M., Pratt, D., Barham, P., and Zeswitz, S. (1994), "NPSNET: A Network Software Architecture for Large-Scale Virtual Environments", MIT Presence: Teleoperators and Virtual Environments, Vol. 3, No. 4, pp. 256-287.

Maes, P. (1995), "Artificial life meets entertainment: lifelike autonomous agents", Communications of the ACM, Vol. 38, No. 11, pp. 108-114.

Maes, P., Darrell, T., Blumberg, B., and Pentland, A. (1997) "The ALIVE System: Wireless, Full-Body Interaction with Autonomous Agents", Multimedia Systems, Vol. 5, No. 2, pp. 105-112.

Maes, P., Guttman, R. H., and Moukas, A. G. (1999), "Agents that Buy and Sell", Communications of the ACM, Vol. 42, No. 3, 81-91.

Mass, Y., and Herzberg, A. (1999), "VRCommerce --- electronic commerce in virtual reality", in Proceedings of the first ACM Conference on Electronic Commerce, pp. 103-109.

Matijasevic, M. (1997), "A Review of Networked Multi-User Virtual Environments", http://www.acim.usl.edu/~maja/tr_97-8-1.ps.gz.

Matsuda, K. (2001), "Unshared shared virtual world", in Proceedings of the 62nd conference of IPSJ (The Information Processing Society of Japan), Special 1, 3, pp. 99-102 (in Japanese).

Matsuda, K. (2002), "Distributed Shared Virtual Environment", Journal of the virtual reality society of Japan, Vol. 7, No.3, pp. 25-29 (in Japanese)

Matsuda, K., and Miyake, T. (2000), "Construction and evaluation of Personal agent-oriented virtual society PAW^2 (the 2nd version)", IPSJ (The Information Processing Society of Japan) journal, Vol. 41, No. 1, pp.2698-2707, (in Japanese).

Mauldin, M. (1994), "Chatterbots, Tinymuds, and the Turing test: Entering the Loebner prize competition," in Proceedings of the Twelfth National Conference on Artifical Intelligence, pp. 16-21.

Meeker, M., and Pearson, S. (1997). The Internet Retailing Report (May 28, 1997), Morgan Stanley, Dean Witter, Discover & Co., New York, http://www.morganstanley.com/institutional/techresearch/inetretail.html.

Microsoft Corporation (1997), Developing for Microsoft Agent (Microsoft Professional Editions), Microsoft Press, ISBN 1-572-31720-5.

Miyake, T., Matsuda, K., and Miyashita, K. (2002), "Construction of the interface agent using Web3D," IPSJ (The Information Processing Society of Japan) SIGNotes Human Interface, Vol. 2002, No. 6, pp. 39-46 (in Japanese).

Mohageg, M., Myers, R., Marrin, C., Kent, J., Mott, D., and Isaacs, P. (1996), "A user interface for accessing 3D content on the World Wide Web", in Proceedings of Computer human interaction 1996 (CHI'96), pp. 446.

Morningstar, C., and Farmer, F. R. (1991), The Lessons of Lucasfilm's Habitat, Cyberspace, MIT Press.

Mosberger, D. (1993), "Memory Consistency Models", Operating Systems Review, Vo. 27, No. 1, pp. 18-26.

Muramatsu, J., and Ackerman, M. S. (1998), "Computing, Social Activity, and Entertainment: A Field Study of a Game MUD", Computer-Supported Cooperative Work Journal, Vol.7-1/2, pp. 87-122.

Nakanishi, H., Nishimura, T., and Ishida, T. (1998), "Effects of a Three-dimensional Virtual Space in Desktop Conferences," IPSJ (The Information Processing Society of Japan) journal, Vol. 39, No. 10, pp. 2770-2777 (in Japanese).

Nakanishi, H., Yoshida, C., Nishimura, T., and Ishida, T. (1996), "FreeWalk: Supporting Casual Meetings in a Network", in Proceedings of International Conference on Computer Supported Cooperative Work (CSCW '96), pp. 308-314.

Nagao, K., and Takeuchi, A. (1994), "Social interaction: Multimodal conversation with social agents", in Proceedings of the twelfth National Conference on Artificial Intelligence 1994 (AAAI-94), pp. 22-28.

Nishida, T. (1997), "Agents in the Network Society: Personalized Artificial Systems", IPSJ (The Information Processing Society of Japan) MAGAZINE, Vol. 38, No. 1, pp. 10-16 (in Japanese).

Ochiai, K. (1997), "Current Status, Base Technologies, and Future Outlook of People Space - Commercial based Virtual Society ---", Virtual Reality Society of Japan Research Report, Vol. 1, No.1, pp. 13-18 (in Japanese).

Randall, N. (1994), Teach Yourself the Internet, Around the World in 21 Days, IN: Sams Publishing, ISBN 0-672-30519-4.

Rekimoto, J., Ayatsuka, Y., and Hayashi, K. (1998), "Augment-able Reality: Situated Communication through Physical and Digital Spaces", in Proceedings of the 2nd International Symposium on Wearable Computers (ISWC'98), pp. 68-75.

Rickel, J., and Johnson, W. L. (1997), "Steve: An Animated Pedagogical Agent for Procedural Training in Virtual Environments", in Proceedings of Animated Interface Agents: Making Them Intelligent, pp. 71-76.

Schiano, D. J. (1999), "Lessons from LambdaMOO: A Social, Text-Based Virtual Environment", Presence: Teleoperators and Virtual Environments, Vol. 8, No. 2, pp. 127-139.

Schiano, J. D., and White, S. (1998), "The First Noble Truth of CyberSpace: People are People (Even When They MOO)", in Proceedings of Computer human interaction 1998 (CHI '98), pp. 352-359.

Schroeder, R., Heather, N., Lee, and Raymond, M. (1998), "The Scared and Virtual: Religion in Multi-User Virtual Reality, Journal of Computer-Mediated Communication, http://www.ascusc.org/jcmc/vol4/issue2/schroeder.html

Shade, J., Gortler, S., He, L. W., and Szeliski, R. (1998), "Layered Depth Images", in Proceedings of ACM SIGGRAPH '98, pp. 231-242.

Singhal, S., and Cheriton, D. (1995), "Exploiting position history for efficient remote rendering in networked virtual reality", Presence: Teleoperators and Virtual Environments, Vol. 4, No. 2, pp. 169-193.

Smed, J., Kaukoranta, T., and Hakonen, H. (2002), "A Review on Networking and Multiplayer Computer Games", Technical Report 454, Turku Centre for Computer Science.

Stephenson, H. (1992), Snow Crash, NY: Bantam Books, ISBN 0-553-56261-4.

Snowdon, D., and West, A. (1994), "AVIARY: Design issues for future large-scale Virtual Environments", Presence: Teleoperators and Virtual Environments, Vol. 3, No. 4, pp. 288-308.

Strauss P., Carey, R. (1992), "An Object-Oriented 3D Graphics Toolkit", in Proceedings of ACM SIGGRAPH'92, pp. 341-349.

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., and Suchman, L. (1987), "Beyond the chalkboard: Computer support for collaboration and problem solving in meetings", Communications of the ACM Vol. 30, No. 1 pp. 32-47.

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

Sumi, Y., Etani, T., Fels, S., Simonet, N., Kobayashi, K., and Mase, K. (1998), "C-MAP: Building a context-aware mobile assistant for exhibition tours", Toru Ishida ed., Community Computing and Support Systems, LNCS 1519, pp.137-154, London: Springer-Verlag.

Takeuchi, A., and Nagao, K. (1995), "Situated Facial Displays: Towards Social Interaction", in Proceedings of SIGCHI '95, pp. 450-455.

Taylor, T. L. (2002), "Living Digitally: Embodiment in Virtual Worlds", The Social Life of Avatars: Presence and Interaction in Shared Virtual Environments, London: Springer-Verlag.

Thorpe, J. (1987), "The new technology of large scale simulator networking: Implications for mastering the art of warfighting", in Proceedings of the 9th Interservice/Industry Training Systems Conference, pp. 492-501.

Tsuchiya, T. (1990), "Oracle Layza's Tales from Fujitsu Habitat" Unpublished manuscript. http://sunsite.tus.ac.jp/pub/academic/communications/papers/habitat/layza.txt.

Tsuji, T., Matsuda, K., and Yajima, W. (1999), "Implementation and Evaluation of A Simplified Cellular Phone on the Virtual Society --- PAW", IPSJ (The Information Processing Society of Japan) SIG Notes Human Interface, Vol. 85, No.4, pp. 19-24.

Underkoffler, J., and Ishii, H. (1999), "Urp: A Luminous-Tangible Workbench for Urban Planning and Design", in Proceedings of ACM SIGCHI '99, pp. 386-393.

Valerie, J. J. (1998), Stretching Exercises for Qualitative Researchers, Thousand Oaks, CA: Sage Publications, ISBN 0-7619-0255-4.

Watanabe, K., Sakata, S., Maeno, K., Fukuoka, H., and Ohmori, T. (1990), "Distributed Multiparty Desktop ConferencingSystem: MERMAID", in Proceedings of Computer Supported Cooperative Work 1990 (CSCW90), pp. 27-38.

Waters, R., Anderson, D., Barrus, J., Brogan, D., Casey, M., McKeown, S., Nitta, T., Sterns, I., and Yerazunis, W. (1997), "Diamond Park and Spline: Social Virtual Reality with 3D Animation, Spoken Interaction, and Runtime Extensibility", Presence: Teleoperators and Virtual Environments, Vol. 6, No. 4, pp. 461-480.

Weiser, M. (1993), "Some Computer Science Issues in Ubiquitous Computing", Communications of the ACM, Vol.36, No.7, pp. 74-84.

Weizenbaum, J. (1966), "ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine", Communications of the ACM, Vol.9, No.1, pp. 36-45.

Williams, S. (1998), The Illustrated Handbook of American and Japanese Gestures, Tokyo: Kodansha, ISBN4-7700-2344-8

Publications

Major Publications

(1) 松田晃一, 上野比呂至, 三宅貴宏, パーソナルエージェント指向の仮想社会「PAW^2」の評価, 電子通信学会論文誌 D-II, Vol. J82-D-II No.10 pp.1675-1683, 1999

(2) 松田晃一, 三宅貴宏, パーソナルエージェント指向仮想社会 PAW^2(第 2 版)の構築と評価, 情報処理学会論文誌, Vol.41, No.1, pp.2698-2707, 2000

(3) 松田晃一, 杉野浩之, 上野比呂至, パーソナルエージェント指向の仮想社会における e コマース実験と評価, 電子通信学会論文誌 B, Vol. J84-B No.12 pp.2392-2400, 2001

(4) Matsuda, K., Evaluation of Personal Agent-oriented Virtual Society PAW^2, MIT Presence: Teleoperators and Virtual Environments, Vol. 10, No. 2, pp. 170-184, 2001

(5) Matsuda, K., Can we sell a virtual object in a virtual society? E-commerce evaluation in PAW^2, a personal agent-oriented virtual society, MIT Presence: Teleoperators and Virtual Environments, Vol. 12, No. 6, 2003 (to be appeared)

(6) 松田晃一, 仮想社会におけるユーザアクティビティの分析と考察, 電子通信学会論文誌 B, Vol. J86-B, No. 12, pp. 129-136, 2003

(7) 内藤剛人, 松田晃一, エージェント指向仮想社会の構築 – PAW –, 日本バーチャルリアリティ学会論文誌, Vol. 4, No.2, pp. 399-406, 1998

(8) Lea, R., Honda, Y., and Matsuda, K., Virtual Society: Collaboration in 3D Spaces on the Internet, pp. 227-250, Computer Supported Cooperative Working, vol. 6, no. 2/3, 1997

International Conferences

(1) Matsuda, K., Honda, Y., and Lea, R., Virtual Society: Multi-user Interactive Shared Space on WWW, in Proceedings of the 6th International Conference on Artificial Reality and Tele-Existence (ICAT '96), pp. 83-95, 1996

(2) Matsuda, K., and Naito, T., Construction of Agent-Oriented Virtual Society --- PAW^2, in Proceedings of IEEE 1999 international conference on consumer electronics (ICCE 99), pp. 152-153, 1999

(3) Matsuda, K., Miyake, T., and Kawai, H., Culture Formation and its issues in Personal Agent-oriented Virtual Society --- "PAW^2", in Proceedings of the forth international conference on Collaborative Virtual Environments 2002 (CVE2002), pp. 88-96, 2002

(4) Honda, Y., Matsuda, K., Rekimoto, J., and Lea, R., Virtual society: Extending the WWW to Support a Multi-User Interactive Shared 3D Environment, in Proceedings of Virtual Reality Modeling Language 1995 (VRML '95), pp. 109-116, 1995

(5) Lea , R., Honda, Y., Matsuda, K., and Matsuda, S., Community Place: architecture and performance, in Proceedings of the second symposium on Virtual reality modeling language 1997 (VRML 97), pp.41-50, 1997

Domestic Conferences

(1) 松田晃一, 谷島亘, パーソナルエージェント指向仮想社会 PAW におけるユーザアクティビティの分析と考察, 情報処理学会 DICOMO2000, pp. 775-780, 2000

(2) 松田晃一, Lea, R., 本田康晃, Virtual Society: インターネット上の共有仮想世界構築基盤, 日本バーチャルリアリティ学会仮想都市研究会第 1 回シンポジウム, Vol.1, No.1, pp. 37-42, 1997

(3) 松田晃一, 非共有型共有仮想世界, 情報処理学会第 62 回全国大会, 特 1, 3, pp. 99-102, 2001

(4) 松田晃一, 谷島亘, 文化における課題と展望~インターネット上の仮想空間 PAW を通じて~, 日本社会情報学会第 14 回全国大会, pp.37-40, 1999

(5) 松田晃一, Lea, R., 本田康晃, Realizing multi-user interactive shared space using VRML, 情報処理学会オーディオビジュアル複合情報処理, No. 15, No. 2, pp.7-14, 1996

(6) 河合裕文, 松田晃一, 共有仮想社会 PAW^2 における非共有サービスの実現と評価, VR 学研報 サイバースペースと仮想都市研究会, CSVC2001-26, Vol.6, No.1, pp. 51-58, 2001.

(7) 平石絢子, 松田晃一, メッシュ方式を用いたオーラ計算の考察と評価, 情報処理学会ヒューマンインタフェース学会研究報告集, Vol.4, No.3, pp. 55-60, 2002

(8) 辻貴孝, 松田晃一, 谷島亘, 仮想社会 PAW における携帯電話機能の実装と評価, 情報処理学会ヒューマンインタフェース学会研究報告集, Vol. 85, No.4, pp. 19-24, 1999

(9) 上野比呂至, 松田晃一, 辻貴孝, 谷島亘, 仮想社会 PAW における携帯電話機能の利用形態とその影響, 電子通信学会 ネットワークとライフスタイル研究会, NTSL 資料 No.1, pp. 7-12, 1999

(10) 真有浩一, 松田晃一, 賀川能明, パーソナルエージェント指向仮想社会 PAW^2 におけるビジネスモデルの構築とその評価, 情報処理学会第 62 回全国大会,pp. 103-106, 特 1, 3, 2001

(11) 三宅貴浩, 松田晃一, 宮下健, Web3D を用いたインターフェースエージェントの構築, 情報処理学会ヒューマンインタフェース学会研究報告集, Vol. 2002, No. 6, pp. 39-46, 2002

Patents

(1) Matsuda, K., Three-dimensional virtual reality space display processing apparatus, a three-dimensional virtual reality space display processing method, and an information providing medium, United States Patent 6,346,956, February 12, 2002

(2) Matsuda, K. et al., Client apparatus, image display controlling method, shared virtual space providing apparatus and method, and program providing medium, United States Patent 6,268,872, July 31, 2001

(3) Matsuda, K. et al., Client apparatus, image display controlling method, shared virtual space providing apparatus and method, and program providing medium, United States Patent 6,253,167, June 26, 2001

(4) Matsuda K. et al., Collision detection apparatus and method for avatars in a three-dimensional virtual space, United States Patent 6,496,207, December 17, 2002

(5) Matsuda K. et al., Information processing apparatus and method, information processing system and program providing medium, United States Patent 6,405,249, June 11, 2002

(6) Matsuda K. et al., Information processing apparatus, information processing method and information providing medium, United States Patent 6,292,198, September 18, 2001

Standardization

VRML97 and VRML97 EAI, International Standards, ISO/IEC 14772-1:1997

Book Chapters

(1) 松田晃一, 仮想世界におけるパーソナルエージェント, エージェントとつくるインタラクティブネットワーク, 培風館, 2003

(2) 松田晃一, サイバースペースにおける文化の形成とその課題,「環境としての情報空間」, アグネ承風社, 2001

(3) 松田晃一, VR とゲーム, 映像情報メディアハンドブック, 映像情報メディア学会, 2000

(4) Rodger, L., Matsuda, K., Miyashita, K., Java for 3D and VRML world, New Riders Publishing, 1999

(5) 松田晃一, PAW ザ・セカンドワールドオフィシャルハンドブック, ローカス, 1999

(6) 松田晃一, VRML と Java, Java プログラミング例題集, 共立出版, bit 別冊, 1997

Tutorials

(1) 松田晃一, 共有仮想世界におけるコミュニケーション, システム制御情報学会学会誌, Vol. 47, No. 10, pp. 469-474, 2003

(2) 松田晃一, 場、環境におけるエージェント, ネットワークと産業フロンティア情報技術に関する調査報告 － ネットワークエージェント技術 －, 日本情報処理開発協会, pp. 187-194, 2000

(3) 松田晃一, 分散型共有仮想環境, 日本バーチャルリアリティ学会誌, Vol. 7, No.3, pp. 25-29, 2002

(4) 松田晃一, VRML と分散仮想環境, 日本ソフトウェア科学会チュートリアル, 1996

(5) Matsuda, K., Miyake, T., and Kawai, H., JAVA+VRML: The future of interactive 3D worlds, in Proceedings of NICOGRAPH International, pp. 3-10, 2002

(6) 松田晃一, VRML の現状と将来展望, バーチャルリアリティ産学研究開発推進委員会 定例活動報告会 配布資料-3, (財) イメージ情報科学研究所, pp. 65-96, 1998

(7) 松田晃一, VR インタフェース, 光技術動向調査報告書, 光産業技術振興協会, pp. 470-473, 2001

(8) 松田晃一, VRML の最新動向とソニーの取り組み－インターネットにおける三次元世界の実現に向けて－, 電子ネットワーク協議会 5 月度月例セミナー, 電子ネットワーク協議会, pp.27-45, 1997

(9) 松田晃一, VRML とサイバースペース, 名古屋大学太陽地球環境研究所共同観測情報センター研究集会 第3回太陽地球環境研究のコンピューティング研究会 講演要旨集, pp. 191-224, 1999

(10) 松田晃一, 本田康晃. 3 次元仮想空間を記述する VRML, 最新インターネットテクノロジ, pp.58-69, 日経 BP, 1996

(11) 松田晃一, インターネット上の共有仮想世界 PAW, bit, 共立出版, Vol.30, No.9, pp. 2-10,1998

(12) 松田晃一, VRML2.0 の概要と CyberPassage が実現する共有仮想空間, Computer Today, pp. 18-24, 9, 1996

(13) 松田晃一, VRML97—インターネット上の仮想空間へようこそ, Software Design, 1997

(14) 松田晃一, VRML の動向と CyberPassage (1), bit, 共立出版, Vol.28, No.7, pp.29-36, 1996.

(15) 松田晃一, VRML の動向と CyberPassage (2), bit, 共立出版, Vol.28, No.8, pp.57-65, 1996

(16) 松田晃一, 杉野浩之, VRML2.0 最終仕様 [Java バインディング], Vol. 28, No. 11, pp. 97-104, 1996

(17) 松田晃一, ビジュアル言語 VRML, bit, 共立出版, Vol. 30, No.1, pp. 34-48, 1998

(18) 松田晃一, VRML の最新動向と今後の可能性, IVR'96: Industrial Virtual Reality Show and Conference, 1996

Awards

(1) 1997年 日経BP技術大賞受賞(VRMLの国際標準化と関連ソフトウェア開発)

(2) 1998年 日経CG VRML コンテスト98優秀賞(PAW^2の実現)

(3) 1998年 情報処理学会DICOMO98ベストカンバーサント賞

(4) 1999年 情報処理学会DICOMO99ベストプレゼンテーション賞

(5) 2001年 情報処理学会優秀発表賞

(6) 2001年 情報処理学会DICOMO2001ベストプレゼンテーション賞

(7) 2002年 日本VR学会仮想都市研究会サイバースペース研究賞