

Context-aware Media Agent for Public Spaces

Ichiro Satoh¹

Abstract. This paper presents an agent-based framework for building and operating context-aware media for digital signage in public spaces. The framework is managed in a non-centralized manner by using mobile agent technology and enables user assistant media agents to follow the movement of their users between stationary computers, e.g., public terminals, by using active RFID-tags. When users move between locations in a public space, their agents provide annotations in personal forms according to their current context. To demonstrate the utility and effectiveness of the system, we describe two applications of the frameworks.

1 Introduction

This paper addresses a general-purpose agent-based framework for providing not only individual users but also groups or communities with context-aware services in large-scale public spaces, e.g., city-wide spaces. The framework aims at providing context-aware media for digital signage in public spaces. In fact, digital signage has been expected to play an important role for advertising, navigation, assistance, or entertainment in public and private environments, such as public museums, retail stores and corporate buildings. Advertising using digital signage should be a form of out-of-home advertising (OOH) in which content and messages are displayed on digital signs with a common goal of delivering targeted messages for specific consumers in specific locations at specific times. Users tend to want personalized information or services rather than universal information or services. The framework can also customize user/location-aware media according to individuals or the members of user groups who are in locations.

Case study: personal annotation in public museum

To illustrate our goal, we describe our case scenario. Please suppose public museums. Visitors move from exhibit to exhibit in a museum. Most visitors want supplementary annotations to understand on exhibitions and the visitors' knowledge and experiences are varied. They may become puzzled (or bored) if the annotations provided to them are beyond (or beneath) their knowledge or interest. Context-aware services in museums should be able to be provided from stationary computers rather than portable terminals. Context-aware services in museums should be able to be provided from stationary computers rather than portable terminals. This is because, if services are provided from such terminals, visitors tend to pay attention to terminals instead of the exhibitions. Not users have portable terminals, e.g., cellular phones, smart phones, and PDAs. Users tend to dislike complex operations and interactions with systems. It is also difficult for visitors, particularly children, the elderly, and handicapped people, to

interact with agents through the buttons or touch panels of portable computers, cellular phones, and stationary terminals.

2 Approach

Context-awareness should provide users with services according to the users' context. This paper presents a framework for building and managing context-aware advertising on digital signage in public places, in-transit, or waiting. We have four key ideas behind it.

- Content, which are played on digital signage, can be anything, including text, images, animations, video, audio, and interactivity. The framework should provide a variety of multimedia content, including interactive content, for consumers, from digital signage.
- To enhance customer experience, the framework can monitor contextual information in the real world by means of sensing systems and select and adapt content according to changes in the real world.
- The framework needs to provide massive users with context-aware services at massive ubiquitous and mobile computing devices in their personalized forms.
- It provides context-aware content with the ability to monitor and response to their users' behaviors, because advertising or user assistant services provided as context-aware services should adapt to the users' profiles.

We cannot assume that all people in cities always have their own portable terminals. Therefore, context-aware services should be provided from not only portable terminals but also stationary terminals. Unlike portable terminals, there may be multiple users in front of stationary computers located at public spaces. Context-aware services are required to be aware of multiple users, since cities are sometimes crowded. For example, only while only one user is front of a terminal, the terminal should provide him/her with services in his/her personal form. When another person comes close to the terminal, the terminal should provide services in user-independent forms.

The scale and complexity of city-level context-aware systems are beyond our ability to manage these using traditional approaches, such as those that are centralized and top-down, because such systems consist of numerous devices and support numerous people. Our system needs to be managed in a non-centralized and autonomous manner. The system itself must be independent of any application-specific services, because it needs to provide a variety of services. It is required to be scalable and robust, even when it consists of numerous devices that may occasionally be inactive, and it needs to support massive numbers of users who may behave unexpectedly.

2.1 Design principles

This framework builds and manages context-aware media content on digital signage. It enables such content to be implemented as mobile

¹ National Institute of Informatics, 2-1-2 Hitotsubashi Chiyoda-ku Tokyo 101-8420 JAPAN, e-mail: ichiro@nii.ac.jp

agents so that they can be dynamically deployed at computers close to users according to context in the real world, e.g., the locations of users and physical objects by using locating systems.

- To support large-scale context-aware systems, they need to be managed in a non-centralized manner. Mobile agents can be managed without any centralized servers.
- Each mobile agent is a programmable entity with data store. Therefore, each mobile agent-based services can define programs to play its visual/audio content and interact with users inside it. Therefore, the framework itself is independent of application-specific tasks and provides multiple kinds of multimedia content, because such tasks are performed within mobile agents.
- After arriving at its destination, a mobile agent can continue working without losing the results of working, e.g., the content of instance variables in the agent's program, at the source computers. Therefore, users can continue to watch or listen to content from computers close to their current positions, even when the users move from location to location.
- The framework deploys and execute mobile agents bound to physical entities or people at computers near the position of the user instead of any remote servers. As a result, mobile agent-based content can directly interact with the user, where RPC-based approaches, which other existing approaches are often based on, must have network latency between computers and remote servers.
- Mobile and ubiquitous computers often have only limited resources, such as restricted levels of CPU power and the amounts of memory. Mobile agents can help to conserve these limited resources, since each agent needs to be present at the computer only while the computer needs the content provided by that agent.

2.2 Related work

There have been several commercial projects for providing context-aware content on digital signage, but they were constructed based on ad-hoc manners. On the other hand, several researchers have explored context-aware services independently of the literature of digital signage. Cambridge University's Sentient Computing project [4] provides a platform for location-aware applications using infrared-based or ultrasonic-based locating systems in a building. Microsoft's EasyLiving project [1] enabled services running on different computers to be combined dynamically according to contextual changes in the real world.

One of the most typical approaches in public spaces, including museums, has been to provide visitor with audio annotations from portable-audio players. These have required end-users to carry players and explicitly input numbers attached to exhibits in front of them if they wanted to listen to audio annotations about the exhibits. Many academic projects have provided portable multimedia terminals or PDAs to users. These have enabled users to interactively view and operate annotated information displayed on the screens of their terminals, e.g., the Electronic Guidebook, [3], the Museum Project [2], ImogI [5], and Rememberer [3]. They have assumed that visitors are carrying portable terminals, e.g., PDAs and smart phones and they have explicitly input the identifiers of their positions or nearby exhibits by using user-interface devices, e.g., buttons, mice, or the touch panels of terminals. However, such operations are difficult for visitors to undertake, particularly children, the elderly, and handicapped people, and tend to prevent them from viewing the exhibits to their maximum extent.

We discuss differences between the framework presented in this paper and our previous frameworks. We constructed a location model for ubiquitous computing environments. The model represents spatial relationships between physical entities (and places) as containment relationships between their programmable counterpart objects and deploys counterpart objects at computers according to the positions of their target objects or places [7]. This was a general-purpose location-model for context-aware services, but was not an infrastructure for deploying and operating such services. We presented context-aware user navigation between exhibits in public museums in our early versions of this paper [8, 9, 10], whereas this paper addresses mobile agent-based context-aware media for digital signage.

3 Context-aware Mobile Agent Runtime

The framework consists of three parts: (1) mobile agents, (3) agent runtime systems, and (3) location information servers, called LISs (Fig. 1). The first offers application-specific content, which are attached to physical entities and places, as collections of mobile agents. The second is running on digital signage and responsible for executing and migrating mobile agents. The third provides a layer of indirection between the underlying locating sensing systems and mobile agents. Each LIS manages more than one sensor and provides the agents with up-to-date information on the state of the real world, such as the locations of people, places, and things, and the destinations that the agents should migrate themselves to.

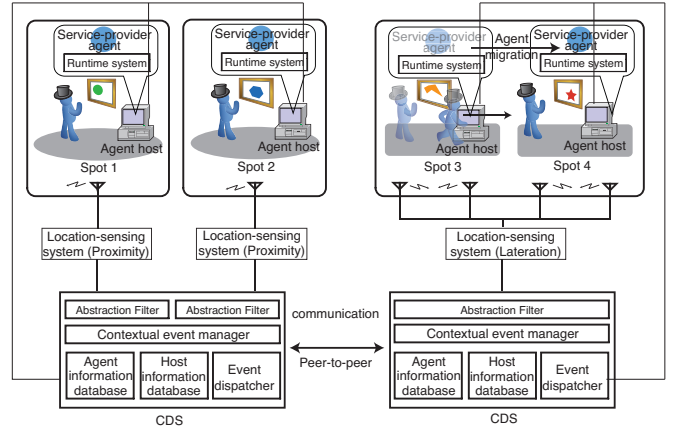


Figure 1. Architecture

3.1 Location Information Server

Each LIS can run on a stationary or mobile computer. It manages multiple sensors that detect the presence of tags and maintains up-to-date information on the identities of tags that are within the zone of coverage of its sensors. This is achieved by polling sensors or receiving the events issued by the sensors themselves. An LIS does not require any knowledge of other LISs. To conceal the differences among the underlying locating systems, each LIS maps low-level positional information from each of the locating systems into information in a symbolic model of location. An LIS represents an entity's location, called *spot*, e.g., a few feet spaces, which distinguishes one or more portions of a room or building. When an LIS detects a new tag in a

spot, it multicasts a query that contains the identity of the new tag and its own network address to all the agent runtime systems in its current sub-network to discover agents or computers (runtime systems) attached to the tag. When there are multiple candidate destinations, each of the agents that is tied to a tag can select one destination on the basis of the profiles of the destinations. When the absence of a tag is detected in a spot, each LIS multicasts a message with the identifier of the tag and the identifier of the spot to all runtime systems in its current sub-network.

3.2 Agent runtime system

Each runtime system migrate agents to other runtime systems running on different computers through a TCP channel using mobile-agent technology (Fig. 2). It is built on the Java virtual machine (Java VM) version 1.5 or later versions, which conceals differences between the platform architectures of the source and destination computers. It governs all the agents inside it and maintains the life-cycle state of each agent. When the life-cycle state of an agent changes, e.g., when it is created, terminates, or migrates to another runtime system, its current runtime system issues specific events to the agent.

When an agent is transferred over the network, not only the code of the agent but also its state is transformed into a bitstream by using Java's object serialization package and then the bit stream is transferred to the destination. Since the package does not support the capturing of stack frames of threads, when an agent is deployed at another computer, its runtime system propagates certain events to to instruct it to stop its active threads. Arriving agents may explicitly have to acquire various resources, e.g., video and sound, or release previously acquired resources.

The framework only maintains per-user profile information within those agents that are bound to the user. It promote the movement of such agents to appropriate runtime systems near the user in response to the user's movements. Thus, the agents do not leak profile information on their users to other parties and they can interact with their mobile users in personalized form that has been adapted to respective, individual users. The runtime system can encrypt agents to be encrypted before migrating them over a network and then decrypt them after they arrive at their destination.

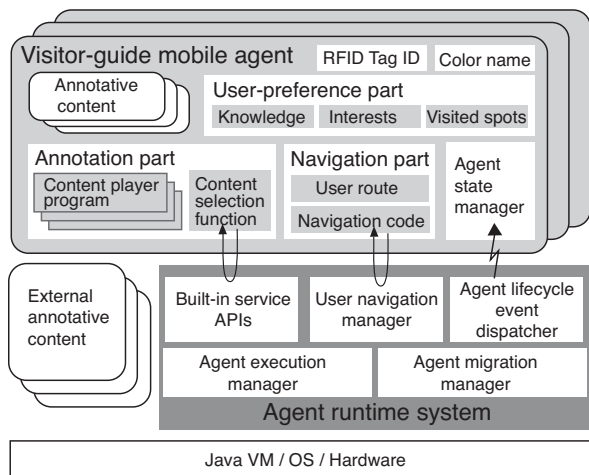


Figure 2. Architecture of runtime system for service-provider agent .

4 Context-aware media mobile agent

Each agent is attached to at most one visitor and maintains per-user preferences on a user and record his/her behavior, e.g., exhibits that they have looked at. It keeps the identifier of the tag attached to its visitor and has facilities to communicate with other agents (Fig. 2). The agent can also define user-personalized services adapted to him/her and access location-dependent services provided at its current computer. To support user/location-dependent services, each agent is dynamically assembled from three parts:

- **The annotation part** defines a task for playing annotations about exhibits or interacting with visitors.
- **The navigation part** defines a task for navigating visitors to their destinations.
- **The user-preference part** maintains and records information about visitors, e.g., their knowledge, interests, routes, names, and the durations and times they spend at exhibits they visit.

Annotation part:

This part is responsible for selecting and playing annotations according to the current spot and route in addition to the information stored in the user-preference part and it plays the content in the personalized form of its user. It is defined as a set of content-selection functions and programs for playing the selected content.

Content-selection function: The function maps more than one argument, e.g., the current spot, the user's selected route, the number of times he/she has visited the spot, and the member of users who are in the spot into a URL referring to the annotative content. The content can be stored in the agent, the current runtime system, or external http servers. That is, each agent can carry a set of its content, play the selected content at its destinations, directly play the content stored at its destinations, or download and play the content stored in Web servers on the Internet. When agents have no services for multiple users at locations, they can fetch agents that can provide the services for the locations.

Content player: Annotation content is various, e.g., text, image, video, and sound. The annotation part defines programs for playing this content. The current implementation supports (rich) text data, html, image data, e.g., JPEG and GIF, video data, e.g., animation GIF and MPEG, and sound data, e.g., WAV and MP3. The format for content is specified in an MIME-based attribute description. Since the annotation part is defined as Java-based general-purpose programs, we can easily define interactions between visitors and agents. The current implementation can divide this part into three sub-parts: opening, annotation, and closing, which are played in turn.

Navigation part

Our agents are required to navigate visitors to their destinations along routes recommended by museums or visitors. After executing their annotation part, the navigation part is invoked by the runtime system to provide visual (or audio) information on the screens of the displays (or from loudspeakers) of the current runtime system. For example, the agents display directions to exhibits that their visitors should next see. We also introduced visitor movements between exhibits as an implicit operation for selecting the routes that they wanted and evaluating what they had learned from the exhibits, because visitor movement is one of the most basic and natural behaviors in museums. As

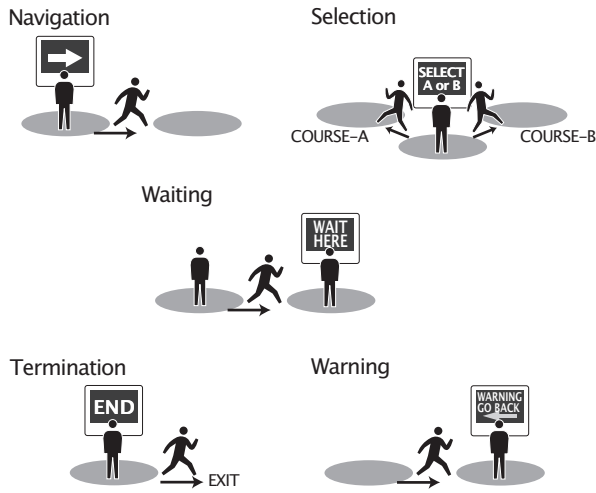


Figure 3. User navigation patterns.

shown in Fig. 3, the current implementation supports the five navigation patterns:

- *Navigation* instructs users to move to at least one specified destination spot.
- *Selection* enables users to explicitly or implicitly select one spot or route from one or more spots or routes close to their current spots by moving to the selected spot or one spot along the selected route.
- *Waiting* informs users to wait for the arrival of other users. When other visitors do not visit within a specified time, this may be inactive.
- *Termination* informs users that they have arrived at the final destination spot.
- *Warning* informs users that they had missed their destination exhibit or their routes.

The user's route is described as a sequence of primitives corresponding to the five patterns. The user-preference part maintains the route inside it. No agent knows the spatial directions to the destinations because the directions themselves depend on the spatial relationships between the locations of the current runtime system and the locations of the destinations, as well as the direction to the current computer's screen. The current implementation permits administrators to manually input the directions of possible destinations on the maps of their target exhibition rooms displayed on the screen. Runtime systems provide several built-in APIs to their visiting agents. For example, if an agent has at least one destination, it invokes a specified API corresponding to the first pattern with the name of the destination; its current runtime system returns the direction to the destination to it or displays the direction on the screen on its behalf. The system can explicitly suspend and store agents on secondary storage after they terminate so that they can continue to assist their users when the users visit museums again.

4.1 Current Status

Although the current implementation was not built for performance, we measured the cost of migrating a null agent (a 5-KB

agent, zip-compressed) and an annotation agent (1.2-MB agent, zip-compressed) from a source computer to a destination computer that was recommended by the LISs. The latency of discovering and instructing an agent attached to a tag after the LIS had detected the presence of the tag was 420 ms and the respective cost of migrating the null and annotation agent between two runtime systems running on different computers over a TCP connection was 41 ms and 490 ms. This evaluation was operated with three computers (Intel Core 2 Duo 2 GHz with MacOS 10.5 and Java version 5) connected via a fast Ethernet. This cost was reasonable for agent migration between computers to follow that visitors moving between exhibits.

Security is essential in deployable software like this system. To prevent malicious agents from being passed between computers, each runtime system supports a Kerberos-based authentication mechanism for agent migration. It authenticates users without exposing their passwords on the network and generates secret encryption keys that can be shared selectively between parties that are mutually suspicious. Since it can inherit the security mechanisms provided in the Java language environment, the Java VM explicitly restricts agents so that they can only access specified resources to protect computers from malicious agents.

5 Experience

This section shows our experiences with this framework.

5.1 Mobile agent for digital signage

We experimented and evaluated mobile agent-based active media for appliances, e.g., electric lights. It is unique among other existing active media because it does not support advertising for its target appliance but also assist users to control and dispose the appliance. We attached an RFID tag to an electric light and provide a mobile agent as an active media for the light. The media is attached to its target item and is deployed at computers close to the current position of the item. The current implementation assumes that an agent for managing active media for its target appliance is created when the appliance is shipped from its factory.

Since the agent defines programs to display three kinds of active media content inside it, it selects them according to their spaces. It supports the lifecycle of the item from shipment, showcase, assembly, using, and disposal.

- **In warehouse:** While the light is in a warehouse, its agent is deployed at a computer in the warehouse. It notifies its specification, e.g., its product number, serial number, the date of its manufacture, the size and weight of it, to a server of warehouse.
- **In store:** While the light is at a showcase in a store, its agent is deployed at a computer close to its target object to display advertising media to sale it customers who visit the store. Figure 4 a) and b) are two images maintained in the agent and displays the price, product number, and manufacture name on its current computer.
- **In house:** When the light is bought and carried to the house of its owner, its agent migrates to a computer in the house and illustrate how to assemble it. Figure 4 c) is the active media for assembly guide. The agent also illustrate how to use it as shown in Figure 4 d). When it is disposed, the agent shows its active media for disposal guide. Figure 4 e) shows the image that illustrates how to dispose the appliance.

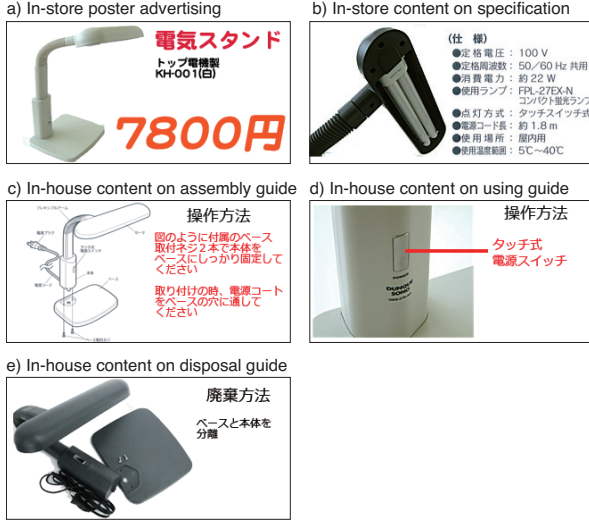


Figure 4. Digital signage for supporting appliance

In house-setting, we can define agents that control appliances, which may not have any network interfaces. In both of the approaches we describe here, the lights are controlled by switching their power sources on or off through a commercial protocol, called X10.

The first can autonomously turn room lights whenever a user with a tagged user is sufficiently close to them. The agent attached to the light can also work as our X10-based server's client and is running on the stationary runtime system in the room. When a tagged user approaches a light, an LIS in the room detects the presence of his/her tag in the cell that contains the light. The LIS then moves the agent that is bound to his/her tag to the runtime system on which the light's agent is running. The user's agent then requests that the lights' agent turns the light on through the inter-agent communication.

The second allows us to use a PDA to remotely control nearby lights. In this system, place-bound controller agents, which can communicate with X10-base servers to switch lights on or off, are attached to places with room lights. Each user has a tagged PDA, which supports the runtime system with WindowsCE and a wireless LAN interface. When a user with a PDA visits the cell that contains a light, the framework moves a controller agent to the runtime system of the visiting PDA. The agent, now running on the PDA, displays a graphical user interface to control the light. When the user leaves that location, the agent automatically closes its user interface and returns to its home runtime system.

5.2 Context-aware Annotation in Museum

The experiment consisted of four spots in front of exhibits, which displayed specimens of stuffed animals, i.e., a bear, deer, racoon dog, and wild boar in an exhibition room of the museum. Each spot could provide five different kinds of animation-based annotative content about the animals, e.g., their ethology, footprints, feeding, habitats, and features, and had a display and Spider's active RFID tag reader with a coverage range that almost corresponded to the space.

All visitors in the experiment were provided with a colored pendant including RFID tags, because it was designed to enable visitors to imagine that their agents, which were virtual owls, were within their pendant. When a visitor with his/her pendant first participated

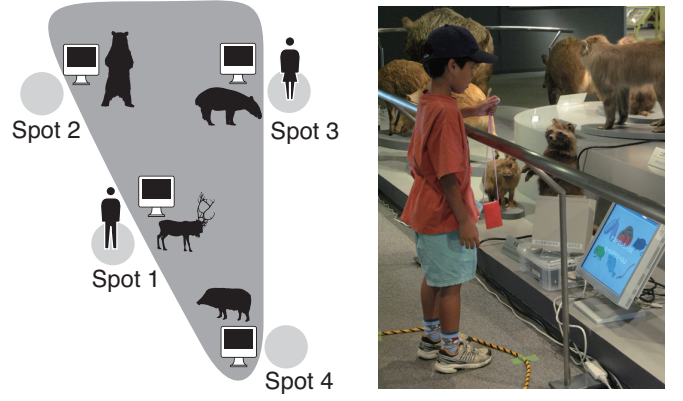


Figure 5. Map in experiment space

in the experiment, an operator input the point of interest and the route for the new visitor and created his/her agent. These pendants were green, orange, blue, yellow, or red. A newly created agent was provided with its own color corresponding to the color of the pendant attached to the agent, because visitors could distinguish between their agents and others' agents through their pendants' colors. This may seem a little naive but it effectively enabled visitors to readily identify their agents. For example, suppose that a visitor enters the spot with the specimen of a racoon dog, then his/her agent migrated from his/her pendant to the display located in the spot. As shown in Fig. 6, an agent tied to the orange pendant plays the opening animation to inform us that its target is a visitor with an orange pendant, where the animation shows the agent's character appearing as an orange pendant. It next plays the annotation and then the closing animation. The durations of the opening animation, annotation, and closing are 7 sec, shorter than 40 sec, and 5 sec.

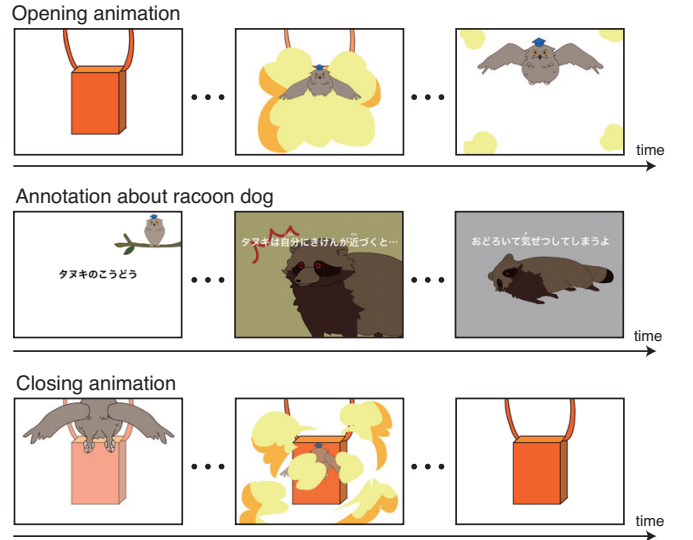


Figure 6. Opening animation, annotation animation, and closing animation

To evaluate the multiple users supports of our system, we provided two kinds of annotation services: annotations about exhibits and quizzes about the annotations that users previously viewed. The

first was to exclusively execute agents for multiple simultaneous users by negotiating between the agents. The second are configured for multiple simultaneous users. Figure 7 shows that users are viewing a quiz about animals displayed on the screen at a spot, where the quiz is configured by the combination of users in spot. They can answer one of two animals by moving to the directions of their selecting animal. Agents for multiple simultaneous users merged their quizzes and enables a group of users by selecting neighboring spots corresponding to their answers in specified spots at specified times. In the current experiment, since merged quizzes were designed for having the same answers, the group of multiple simultaneous users were expected to the same spots to answer the quizzes.

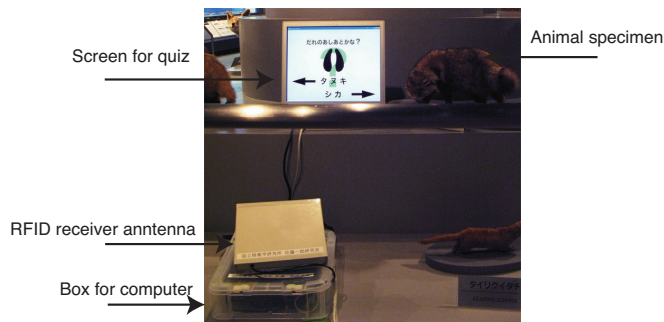


Figure 7. Answering quiz by moving to directions of animals

We did the experiment over a two-week period in addition to another experiment for individual users. Each day, more than 60 individuals or groups took part in the experiment. The experimental system consisted of one LIS and runtime systems running on four computers.

When the LIS detected the presence of a tag bound to a visitor at a spot, it instructed the agent bound to the user to migrate to a computer contained at the spot. After arriving at the computer, the runtime system invoked a specified callback method defined in the annotative part of the agent. The method first played the opening animation for the color of the agent and then called a content-selection function with his/her route, the name of the current spot, and the number of times that he/she had visited the spot. The latency of migrating an agent and starting its opening animation at the destination after visitors had arrived at a spot was within 2 seconds, so that visitors could view the opening animation soon after they began standing in front of the exhibits. The method next played the selected content and then played the closing animation. After that, the runtime system invoked a specified callback method defined in the navigation part.

When an agent bound to a user provided quizzes, it enabled the user to select one of two destination spots by using the *selection* and *waiting* patterns provided on its current computer. When a visitor moved to one of the spots, his/her agent could record their selection. If the selection corresponded to a quiz choice, when a user moved to a spot corresponding to a correct or incorrect answer, their agent modified the visitor's profile that was maintained within it. Furthermore, if a user left out his/her route, the navigation part invoked a method to play warning content for him/her to return to his/her previous spot. Our system could use visitor-movement between exhibits as a natural interaction between users and agents.

6 Conclusion

We designed and implemented a context-aware infrastructure for building and managing mobile agent-based content displayed on digital signage in public spaces. It provides users and physical entities with mobile agent-based content to support and annotate them. Using location-tracking systems, it can migrate content to stationary or mobile computers near the locations of the users and physical entities to which the agents are attached. That is, it allows a mobile user to access its personalized services in an active computing environment and provides user/location-dependent active media to a user's portable computer or stationary computer. It is managed on a decentralized manner.

Finally, we would like to identify further issues that need to be resolved. Some may consider that the non-centralized management architecture we used was needed to operate the context-aware visitor-guide services in museums, as described in this paper. However, our final goal is to provide these large-scale context-aware services without a non-centralized management architecture in large spaces, e.g., cities. We therefore need to demonstrate the scalability of the framework for large-scale context-aware services. This paper only described the design and implementation of agents and the effectiveness and utility of the model; however, the experiments provided us with other interesting experiences, e.g., usability, human aspects, and user modeling. We intend to present these experiences in future papers. We need a mechanism for dynamically making composite services for multiple users from services for individual users.

REFERENCES

- [1] B.L. Brumitt, B. Meyers, J. Krumm, A. Kern, S. Shafer: EasyLiving: Technologies for Intelligent Environments, Proceedings of International Symposium on Handheld and Ubiquitous Computing, pp.12-27, 2000.
- [2] C. Ciavarella and F. Paterno, The Design of a Handheld, Location-aware Guide for Indoor Environments, Personal and Ubiquitous Computing, vol.8 no.2, pp.82-91, 2004.
- [3] M. Fleck, M. Frid, T. Kindberg, R. Rajani, E. O'Brien-Strain, E. and M. Spasojevic, From Informing to Remembering: Deploying a Ubiquitous System in an Interactive Science Museum. IEEE Pervasive Computing vol.1, no.2, pp.13-21, 2002.
- [4] A. Harter, A. Hopper, P. Stegges, A. Ward, and P. Webster: The Anatomy of a Context-Aware Application, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 59-68, ACM Press, August 1999.
- [5] K. Luyten and K. Coninx, ImogI: Take Control over a Context-Aware Electronic Mobile Guide for Museums, In Workshop on HCI in Mobile Guides, in conjunction with 6th International Conference on Human Computer Interaction with Mobile Devices and Services, 2004.
- [6] C. Rocchi, O. Stock, M. Zancanaro, M. Kruppa, A. Kruger: The Museum Visit: Generating Seamless Personalized Presentations on Multiple Devices, Proceedings of 9th international conference on Intelligent User Interface, pp.316-318, ACM Press, 2004.
- [7] I. Satoh: A Location Model for Pervasive Computing Environments, Proceedings of IEEE 3rd International Conference on Pervasive Computing and Communications (PerCom'05), pp.215-224, IEEE Computer Society, March 2005.
- [8] I. Satoh: Experience of Context-aware Services in Museums, in Proceedings of International Conference on Pervasive Services (ICPS'2008), pp.81-90, ACM Press, July 2008.
- [9] I. Satoh: Context-aware Agents to Guide Visitors in Museums, in Proceedings of 8th International Conference on Intelligent Virtual Agents (IVA'08), Lecture Notes in Artificial Intelligence (LNAI), vol.5208, pp.441-455, September 2008.
- [10] I. Satoh: A Context-aware Service Framework for Large-Scale Ambient Computing Environments, in Proceedings of ACM International Conference on Pervasive Services (ICPS'09), pp.199-208, ACM Press, July 2009.