ICAT 2005
December 5-8, Christchurch, New Zealand

Distributed Autonomous Interface using ActiveCube for Interactive Multimedia Contents

Ryoichi Watanabe[1], Yuichi Itoh[1], Yoshifumi Kitamura[1], Fumio Kishino[1], and Hideo Kikuchi[2]

[1] Human Interface Engineering Lab.,
Graduate School of Information Science and Technology, Osaka University
2-1 Yamada-oka, Suita, Osaka, 565-0871, Japan
ryoichi@ist.osaka-u.ac.jp
[2] System Watt Co., Ltd. 4-1-38-4F, Isobedori, Chuo-ku, Kobe, 651-0084, Japan

Abstract

In this paper, we describe a unique and novel function of ActiveCube that develops interactive multimedia contents; a new method that reduces the load of the host PC by distributed control between the host PC and cubes. Furthermore, we propose and implement a method that restricts unnecessary packets in the network between the host PC and cubes by realizing autonomous activity of each cube and encapsulation of input/output devices. Then we conducted experiments to evaluate the performance of the proposed system to achieve natural and intuitive interactions.

Key words: tangible user interface, 3D modeling, real-time interaction, bidirectional interface, distributed control, autonomous activity

1. Introduction

Recently, technological advancements in personal computers provide users an opportunity to use and enjoy various kinds of rich multimedia contents, including 3D video games, interactive digital art work, audio and video delivered over the web, and so on. Such interactive contents cannot be realized by using traditional media (e.g., books, audio records, still images, and films). Though multimedia contents have grown and are widely spread, interactions with these contents are generally performed by a keyboard, pointing devices, and a 2D desktop monitor. Since such interactions are very different from those in the real world where we live and act, users have difficulty interacting with multimedia contents.

On the other hand, much research has focused on user interfaces that can make interactions with a computer more intuitive. Some researchers utilize physical electronic objects as user interfaces [1-10]. These interfaces have improved the intuitiveness of 3D shape modeling or interactive manipulation. In addition, if users construct 3D objects on a computer by simply combining physical objects, the user interface for 3D shape modeling becomes more intuitive. If constructed physical objects were capable of accepting user input and expressing simulated output results, users could directly interact with 3D environments by using physical objects instead of ordinary interaction devices (e.g., a mouse, a keyboard, and a display monitor). Consequently, user interfaces become more intuitive, and it would be easier to understand what is happening in a virtual environment because the constructed object in the real world acts as a physical replica of the virtual structure.

Thus, if users use physical electronic objects as user interfaces during interactions with multimedia contents, they could create and change contents by simply assembling physical blocks. Moreover, they could interact with the contents by using the constructed physical structures. As a result, we could develop useful systems in educational fields for children or in the medical field to prevent dementia. To realize such a user interface that allows users to intuitively interact with rich multimedia contents, we developed the ActiveCube system [11-13] that allows users to construct and interact with 3D environments using physical cubes as bidirectional user interfaces. By using this system, we developed various applications using multimedia contents: retrieval of 3D shape models [14], cognitive assessment [15, 16], and an interactive edutainment system [17]. However, as the contents become richer, procedural problems in the computers increase and become more complex.

In this paper, we describe a new ActiveCube function that reduces the host PC's load by sharing procedures between the host PC and cubes. Furthermore, we propose and implement a method that restricts unnecessary packets in the network between the host PC and cubes by realizing autonomous activity of each cube. We also conducted experiments to evaluate the performance of the proposed methods in achieving natural and intuitive interactions.

2. Previous Work

2.1 Related Work

3D shape modeling by assembling physical objects is a solution that solves the complexity problem in 3D space and offers an easy way to recognize spatial configuration in 3D environments. Research on 3D shape modeling using physical objects to achieve an intuitive interface was carried out in the early 1980s with architecture designs [1] using "Machine-Readable Models" [2, 3]. These ideas were succeeded by further efforts [4, 5]. Recently, a modeling system was proposed in which the geometry of completely assembled Lego-type blocks is recognized by a computer after being connected to it and turned on [6]. In this system, users utilize 3D models in the game. However, the recognition of the geometry of connected blocks in this approach is an offline process that does not work in real time while a user is connecting or disconnecting blocks.

Some interaction systems use physical objects. In the "Triangles" system [7], users can enjoy such computer interaction as opening a web page or interacting with a story by connecting triangular planes to each other. However, users can only input the shape of the structure and cannot acquire the results of interaction via physical objects due to a lack of output devices on the planes. "AlgoBlock" [8] has a program-coding interface that uses physical blocks with which users can easily program Logo-like language by assembling physical blocks. However, the environment of execution is separated from coding, forcing each learner to understand the positioning of each environment in addition to its role. "Electronic Blocks" [9] is computational Lego blocks that have inputs or outputs. Users create simple programs by assembling blocks to learn programming and logic through play. "Navigational Blocks" [10] was designed to achieve easy interaction with a database through tactile manipulation and haptic feedback. Using physical blocks as representations of data queries effectively provides users with an easily understood, creative, and explorative means of learning.

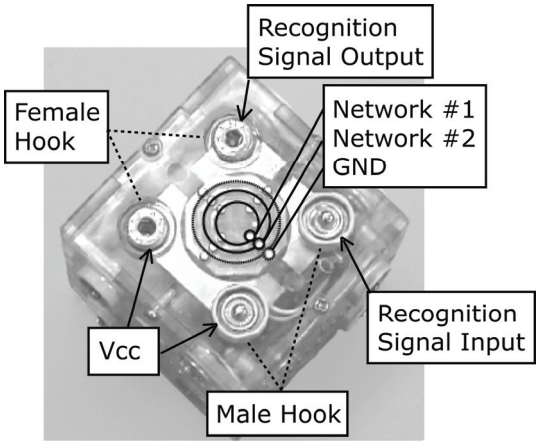

Fig. 1 Arrangement of contact terminals

2.2 ActiveCube System

In this section, we describe the functions and features of the ActiveCube system related to this paper. ActiveCube is a novel user interface that consists of a set of rigid cubes. It has three features: real-time 3D modeling, real-time interaction, and a bidirectional interface. To achieve these features, we need technology that can obtain and control the information of connected faces and sensor values in real time. So, we use a real-time network management system called Local Operating Network (LON) technology (Echelon Corporation). Each cube is equipped with a microprocessor called a Neuron Chip (NC, Toshiba Corporation, TMPN3120FE3M) that allows us to control all cubes in real time. It has 2 KBytes RAM, 16 KBytes ROM, 2 KBytes Electrically Erasable Programmable ROM (EEPROM), and can run at 20 MHz. The chip achieves a speed of 39 kbps for communications with other chips. There are several input/output cubes those are equipped with input/output devices and control them. In addition, there are also plain cubes that enrich only the shape representation without input/output devices.

An ID number (called a cube ID) is assigned to each cube for the unique identification of cubes. An ID number (called a face ID) is also assigned to each face of the cube to identify connecting faces. Four electric communication lines are required on each face for communication and power between cubes. For this purpose, four contact terminals are arranged on each face. Three contact terminals allocated on a concentric circle at the center of the face are used for networks #1 and #2 and the ground (GND), as shown in Figure 1. Two of the four hooks for physical connections between cubes are used to supply power (Vcc), and the other two hooks are used to recognize connections with other cubes.

The connected cubes constitute a network bus made up of RS-485 components. Therefore, all cubes and the host PC can communicate directly in real time without depending on the topology. The cubes are connected to the host PC through a special cube called a base cube

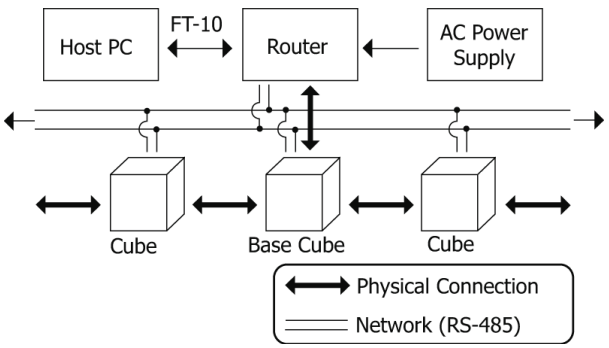

Fig. 2 ActiveCube system configuration

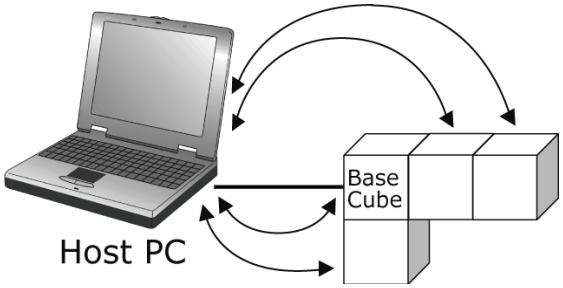

Fig. 3 Central control by host PC

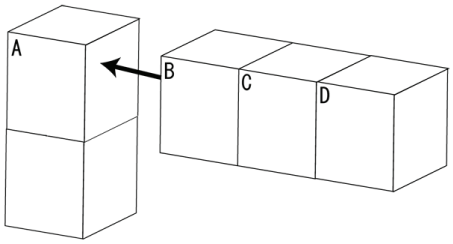

Fig. 4 Multiple-cube connection

(Figure 2). Communication between cubes and the host PC is achieved by translating RS-485 to FT-10 on a router. By using a special network card, the host PC can access the network constituted of cubes. The software installed in the host PC uses the ActiveCube library based on a toolkit called LonWorks Network Services (LNS), Echelon Corporation. By using this library, the host PC can control valuables in cubes.

As shown in Figure 3, the host PC directly communicates with cubes and obtains their connected cube and face IDs in real time. The host PC manages connection status information in the tree structure. As a result, the host PC can correctly recognize the 3D structure. Moreover, to realize such flexible 3D modeling as multiple-cube connections (Figure 4), the host PC controls output signals for face recognition [13]. When the host PC obtains data from the input cubes, it performs polling to input cubes at regular intervals. The causal relationship between the input and output devices can be determined by applications in the host PC.

Thus, because the host PC must communicate directly with all of the cubes and manage the system in real time, the host PC requires a lot of CPU resources. Consequently, it is difficult to fully realize applications using rich multimedia contents that require huge CPU power. Toward intuitive interaction with multimedia contents, we have to improve several points of the previous ActiveCube system. Moreover, in applications for children, the ActiveCube system requires quick responses because children sometimes quickly and repeatedly connect and disconnect them.

In the next section, we describe a novel function of the implementation of ActiveCube: a method that reduces the host PC's load. We propose a method to restrict unnecessary packets in the network between cubes and the host PC.

3. Distributed Control by Base Cube

3.1 Addition of Microprocessor

To reduce the host PC's load, we propose a method to share procedures between the host PC and the base cube. Though we incorporated NCs into all of the cubes, the NC already has a lot of tasks, such as communication with other cubes and the control of input/output devices in the cubes. Therefore, it is difficult for the NC to perform additional procedures. Instead of designing completely new infrastructure for our implementation, we kept the NC and incorporated a second microprocessor into the base cube.

Because the second microprocessor must be fast and have a large amount of programmable memory to share tasks with the host PC, we chose H8S/2633 from the H8S family [Renesas Technology Corporation]. It is a high-performance microprocessor with a 32-bit H8S/2600 CPU core and a variety of built-in peripheral functions. It has 256 KBytes reprogrammable flash memory and 16 KBytes data RAM. The maximum clock frequency is 25 MHz. The program is coded in C language and assembler and downloaded into flash memory via the RS-232C serial port by using a toolkit called Flash Development Toolkit [Renesas Technology Corporation].

3.2 System Configuration

Figure 5 shows the proposed system configuration. The base cube has two microprocessors, a NC and a H8S. The host PC communicates with H8S via RS-232C port. The two microprocessors in the base cube communicate with each other via parallel communication. Each cube communicates with other cubes via a RS-485 network

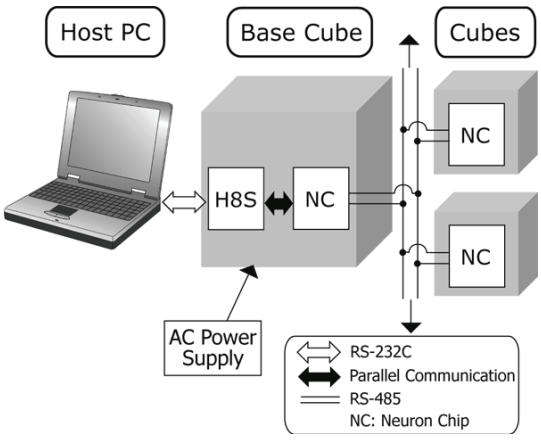

Fig. 5 Proposed system configuration of ActiveCube system

bus, as in the previous implementation.

All the cubes constitute a network bus comprised of RS-485 components, as in the previous implementation, through which the host PC and cubes directly communicate. In addition, in the proposed method all cubes can communicate with each other by using "explicit messages," which are supported by Neuron C (programming language for the NC). The program can send or receive a maximum of 228 Bytes of data between NCs. The unique address of a NC is defined in the program and written into the EEPROM.

As described above, the two microprocessors in the base cube communicate with each other via parallel communication. Neuron C supports a parallel I/O object, which allows bidirectional communication with the other microprocessor at 3.3 Mbps. The physical interface of the parallel I/O object is realized by using all eleven I/O ports. To synchronize communication and prevent collisions of sending data, a token passing protocol is built in the NC firmware. The microprocessor having a token owns the right to send data. In communication between two microprocessors, we implement H8S as a master and NC as a slave. When H8S has data to send to NC, H8S sets the first bit of data high and sends data from the next bit. When H8S doesn't have any data to send, it delegates the right by setting the first bit low. On the contrary, when NC sends data to H8S, we utilize a built-in function for parallel communication of the NC. Since in the base cube all NC I/O ports are used for parallel communication, we implemented the functions to recognize connected faces or to control input/output devices into the H8S.

The H8S in the base cube communicates with the host PC through the RS-232C serial connection at the baud rate of 9.6 kbps. The H8S and the host PC send and receive data in one Byte units. When one cube sends data to the host PC, it first sends data to NC in the base cube, and NC transmits data to H8S. Then H8S converts the explicit message into a protocol that the host PC can understand. On the other hand, when the host PC sends data to a cube, the H8S adds header information to the data and transmits it to NC in the base cube through the parallel interface. NC converts the received data into an explicit message and sends it to the cube whose address is included in the message.

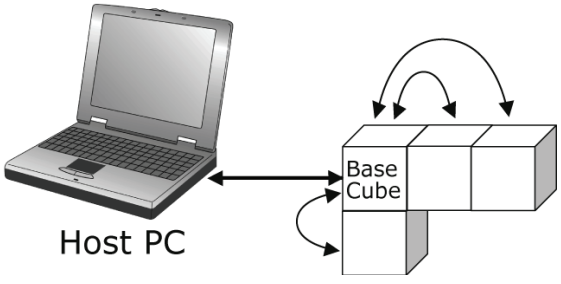

Fig. 6 Distributed control by base cube

We developed the ActiveCube Library offered as a class of C++ and Dynamic Link Library (DLL). With this library, the program in the host PC can recognize the constructed structure by using connection information from the base cube and control I/O devices equipped on each cube.

3.3 Distributed control by Base Cube

We describe the details of the proposed method in which the host PC and the base cube share the procedures of controlling cubes, as shown in Figure 6. When a new cube (a child cube) is connected to a cube that has already been connected to the network (a parent cube), the base cube communicates with these cubes by an explicit message, obtains their cube and face IDs, and sends them to the host PC. The base cube controls the output signals for face recognition instead of the host PC. For example, when multiple cubes (B, C, and D) are connected to parent cube (A), as shown in Figure 4, the base cube controls the output signal from the cube near cube A and creates the same situation when the cubes are connected one by one. As a result, the host PC correctly recognizes the 3D structure.

Since the host PC doesn't care about shape recognition procedures any more, we can reduce the host PC's load. Consequently, we can allocate resources in the host PC to the application software with rich multimedia contents.

4. Autonomous Cube Activity

In previous implementations, when the host PC obtains data from input cubes, it performs polling to all of the input cubes at regular intervals as shown in Figures 7 (1), (2). After receiving data from the input cubes, the host PC retrieves the causal relationship between the input and output devices determined in the software (Figure 7 (3)) and then controls the output devices (Figure 7 (4)). This also creates a situation where the host PC always requires a lot of CPU resources for

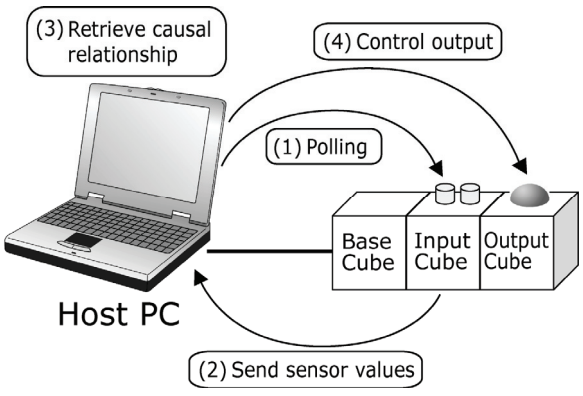

Fig. 7 Interaction with input/output devices by polling

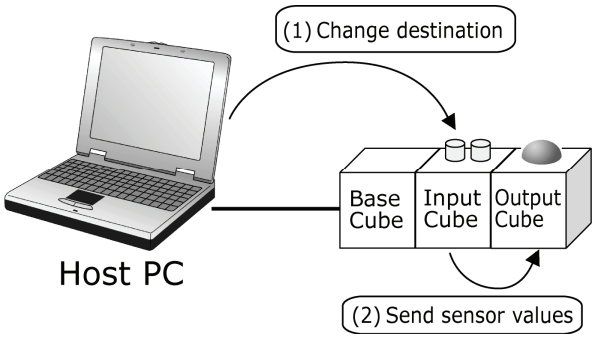

Fig. 8 Input cube sends sensor values directly to output cube

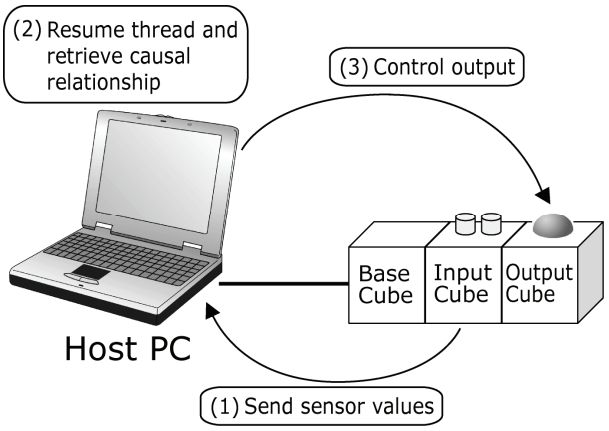

Fig. 9 Encapsulation of input/output devices

polling. Moreover, there are sometimes unnecessary packets in the network between the host PC and cubes. In this section, we present a method to resolve these issues.

4.1 Autonomous Activity of Input Cubes

The most appropriate solution for reducing the host PC's load is to quit polling. For this purpose, we propose a method to let input cubes work autonomously and observe sensor values. When the application doesn't require the sensor value of an input cube, the host PC stops sending cube data to prevent unnecessary packets in the network between the host PC and cubes. If an input cube is permitted to send data by the host PC and detects changes in sensor values, it sends them to the host PC by explicit message. Thus, since the host PC doesn't need to perform polling at regular intervals, unnecessary packets between connected cubes and the host PC are restricted, reducing the load of the host PC.

4.2 Encapsulation of Input/Output Devices

When users interact with the ActiveCube system using input/output devices, the best solution for reducing the host PC's load is to directly control output devices in an output cube with an input cube's sensor value (Figure 8). In this method, first the host PC changes the destination of an input cube's sensor value from the host PC to an output cube, which shows interaction results to the sensor. Secondly, when the input cube detects changes in sensor value, it directly sends the value to the output cube. Once the destination is changed, the input and output cubes autonomously and directly communicate with each other, reducing the host PC's load. However, since the causal relationship between the input and output devices is programmed into the cubes in advance, it is difficult to dynamically change the relationship by connection procedures or constructed structure.

To dynamically change the relationship between I/O cubes, we propose a method to create threads where causal relationships are described at each input cube in the host PC (Figure 9). By using this method, the causal relationship is encapsulated at each input cube. When an input cube is connected, a thread for the input cube is created. When the input cube detects changes of sensor value, it sends the sensor value to the host PC. After that, the host PC resumes the thread for the input cube and searches for the causal relationship described in the thread. Then the host PC controls the output device based on the causal relationship. We incorporated this function into the ActiveCube library for application developers.

Because interaction between the input and output devices performed in the thread differs from the main process, the main process's load is reduced, simplifying application operations that use rich multimedia contents. In addition, since the causal relationship between the input and output devices is described at each input cube, the relationship is clarified. Consequently, it becomes easy for developers to design interaction parts.

5. Experiment

When users interact with multimedia contents using the ActiveCube system, quick response is crucial in such tasks as connection/disconnection of the cubes and interaction with input/output devices. We conducted experiments to evaluate the real-time performance of the proposed system configuration. Furthermore, we allowed several children to construct and interact with the 3D structures using ActiveCube and discussed our system's usability.

5.1 Experimental Method

In the experiments, we used four kinds of cubes: light cubes (that emit RGB colors with 256 brightness levels), plain cubes, a base cube, and a ping cube (that immediately replies to a message from the host PC after receiving a message from it). We conducted the following three experiments both in the previous implementation and in the proposed system.

Experiment 1: Communication time between the host PC and cubes

Communication time between the host PC and the cubes was measured. A message from the host PC reaches the ping cube via the base cube and some plain cubes. After receiving this message, the ping cube sends a message to the host PC via the same cubes whose time is measured by the timer in the software on the host PC. We conducted this experiment five hundred times.

Experiment 2: Interaction time between output cubes and host PC

We measured the time from the order transmission by host PC to the actual light-up of the light cube.

First, a base cube and a light cube were connected to the network. Next, when a certain button on the display of the host PC is pressed, the host PC sends a command to the light cube to turn on its light. The color of the label on the host PC's display is simultaneously changed. We carried out ten measurements from the change of the color of the label to the light-up of the cube. As measurement apparatus, we use a digital video camera (NTSC, 29.97fps=33.4 msec/frame) because in the cube it is impossible to synchronize a clock between the host PC and the microprocessor.

Experiment 3: Recognition time of connection and disconnection

The recognition time of connection and disconnection was measured. Concretely, we connected and disconnected the plain cube to/from the base cube and then measured the time required for the host PC to recognize connections/disconnections. At first, when a plain cube was connected to the base cube, the power LED was turned on as soon as it received power. If the host PC recognized this connection, the label's color on the display of the host PC changed. Similarly, when the plain cube was disconnected, the power LED turned off, and if the host PC recognized this disconnection, the label's color changed. We measured the time required for the host PC to recognize the connection/disconnection ten times with a digital video camera.

5.2 Results

From the results of experiment 1, in the previous implementation, the average communication time between the host PC and cubes was 248.2 msec; in the proposed method, the average communication time was 80.8 msec. From the results of experiment 2, in the previous implementation, the average interaction time between the host PC and the output cubes was 447.6 msec. On the contrary, in the proposed method, this time was 33.4 msec. From experiment 3, in the previous implementation, the average connection recognition time was 1409.5 msec (standard deviation is 253.8 msec), and disconnection was 865.1 msec (standard deviation is 148.1 msec). In the proposed method, the average connection recognition time was 370.7 msec (standard deviation is 19.0 msec) and disconnection was 126.9 msec (standard deviation is 56.3 msec).

5.3 Discussion

The experiments proved that the proposed method achieves much quicker response than the previous implementation. We assume this difference reflects improvements of the communication system. In the previous implementation, we utilized a LNS toolkit to control each cube; however, since this toolkit could not directly control the communication layer, it required much time in the communication process. In contrast, with the proposed method, cubes and the host PC directly communicate with each other, dramatically improving the processing speed in the communication part. Moreover, in experiment 3 the recognition time of connection/disconnection became shorter because the base cube controls the face recognition signal instead of the host PC, and so the communication route was shortened.

Generally, the response time of a user interface should be appropriate to the tasks. For instance, in simple and frequent tasks, users probably feel comfortable if the response time is around one second [18]. From all the results measured by experiments, response times of interaction with ActiveCube are less than one second, and these results show that the proposed methods have sufficient performance to support natural and intuitive interactions.

As shown in Figure 10, several children played with a simple ActiveCube application in which users construct 3D structures by computer and interact with them using input/output cubes (e.g., a gyroscopic sensor cube, an ultrasonic sensor cube, a light cube, and so on). After a few minutes, most children understood how to use

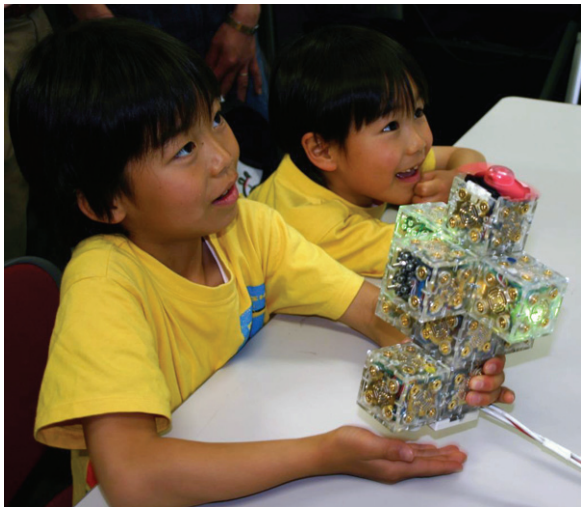

Fig. 10 Children's interaction with ActiveCube

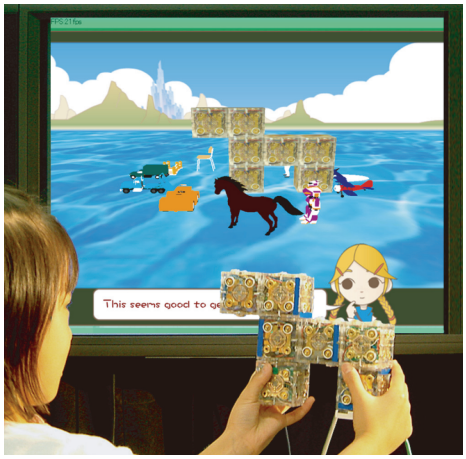

Fig. 11 3D shape model retrieval in TSU.MI.KI system

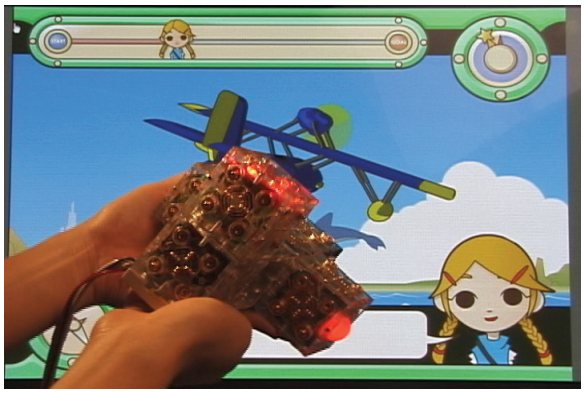

Fig. 12 Interaction with virtual plane

ActiveCube and the causal relationship between the input and output devices (e.g., the brightness of a light is controlled by the distance measured by an ultrasonic sensor). Children especially seemed to enjoy interacting with the input/output devices. Some searched for a new causal relationship by connecting additional input/output cubes. However, some children broke the structure during interaction. For them it seemed difficult to correctly construct the 3D structure because of the connectors between cubes. We have to improve the system's robustness and connectivity.

6. Applications

By using the proposed system, it has become possible to realize applications that utilize rich multimedia computer contents. We developed educational applications for children called "TSU.MI.KI," based on a traditional Japanese toy [17]. Children play in the virtual environment by constructing and manipulating physical cubes and using input/output devices equipped on cubes.

As one example using the TSU.MI.KI system, we developed a storytelling system for children to seek the virtual world (Figures 11 and 12). Users have to find a virtual object to overcome difficulty in the virtual world and create a virtual object by assembling physical cubes. The system retrieves and shows candidate objects similar to the constructed shape (Figure 11). If users select one of these candidates, they can manipulate it by using the constructed structure equipped with input/output devices and travel in the virtual world. For example, if users select a plane and encounter a dangerous situation, such output cubes as lights or vibrators show warning messages (Figure 12).

TSU.MI.KI provides edutainment (educational-entertainment) experiences for children. We believe this application can stimulate their creativity and imagination.

7. Conclusion

We presented a unique and novel function of ActiveCube that easily and intuitively interacts with rich multimedia contents; the method reduces the host PC's load by sharing procedures between the host PC and cubes. Moreover, we proposed and implemented a method that restricts unnecessary packets in the network between cubes and host PC by realizing autonomous activity of each cube and encapsulation of input/output devices. Experimental results showed that ActiveCube had sufficient performance to achieve natural and intuitive interactions.

As future works, we are planning to implement more sophisticated I/O devices into this system (e.g., a liquid crystal display or a camera for image recognition), to incorporate wireless communication between the host PC and cubes, and to develop applications that fully exploit the ActiveCube's functionality.

Acknowledgements

This research was supported in part by Special Coordination Funds of the Science and Technology Agency of the Japanese Government, and "The 21st Century Center of Excellence Program" of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and the Exploratory Software Project grant of Information-technology Promotion Agency, Japan.

References

1. R. Aish and P. Noakes, "Architecture without numbers - CAAD based on a 3D modeling system," Computer-Aided Design, vol.16, no. 6, pp. 321-328, 1984.
2. J. H. Frazer, J. M. Frazer, and P. A. Frazer, "Three dimensional data input devices," Proc. of Conference on Computers/Graphics in the Building Process, pp. 409-416, 1982.
3. J. H. Frazer, "An evolutionary architecture," Architectural Association, 1995.
4. G. Anagnostou, D. Dewey, and A. T. Patera, "Geometry-defining processors for engineering

design and analysis," The Visual Computer, vol. 5, pp. 304-315, 1989.

5. M. Resnick, F. Martin, R. Sargent, and B. Silverman, "Programmable bricks: toys to think with," IBM Systems Journal, vol. 35, no. 3-4, pp. 443-452, 1996.
6. D. Anderson, J. Frankel, J. Marks, A. Agarwala, P. Beardsley, J. Hodgins, D. Leigh, K. Leigh, K. Ryall, E. Sullivan, and J.S. Yedidia, "Tangible interaction + graphical interpretation: a new approach to 3D modeling," Proc. of SIGGRAPH2000, pp. 393-402, 2000.
7. M.G. Gorbet, M. Orth, and H. Ishii, "Triangles: tangible interface for manipulation and exploration of digital information topography," Proc. of Conference on Human Factors in Computing Systems (CHI '98), pp. 49-56, 1998.
8. H. Suzuki and H. Kato, "Algoblock: a tangible programming language, a tool for collaborative learning," Proc. of 4th European Logo Conference, pp. 297-303, 1993.
9. P. Wyeth and G. Wyeth, "Electronic blocks: tangible programming elements for preschoolers," Proc. of INTERACT' 01, pp. 496-503, 2001.
10. K. Camarata, E. Y. Do, B. R. Johnson, and M. D. Gross, "Navigational blocks: navigating information space with tangible media," Proc. of International Conference on Intelligent User Interface (IUI '02), pp. 31-38, 2002.
11. Y. Kitamura, Y. Itoh, and F. Kishino, "Real-time 3D interaction with ActiveCube," CHI 2001 Extended Abstracts, pp. 355-356, 2001.
12. R. Watanabe, Y. Itoh, M. Kawai, Y. Kitamura, F. Kishino, and H. Kikuchi, "Implementation of ActiveCube as an intuitive 3D computer interface," Proc. of 4th International Symposium on Smart-Graphics, pp. 43-53, 2004.
13. R. Watanabe, Y. Itoh, M. Asai, Y. Kitamura, F. Kishino, and H. Kikuchi, "The soul of ActiveCube - implementing a flexible, multimodal, three dimensional spatial tangible interface," Computers in Entertainment, vol. 2, no. 4, 2004.
14. H. Ichida, Y. Itoh, Y. Kitamura, and F. Kishino, "Interactive retrieval of 3D shape models using physical objects," Proc. of the 12th ACM International Conference on Multimedia 2004, pp.692-699, 2004.
15. E. Sharlin, Y. Itoh, B. Watson, Y. Kitamura, S. Sutphen, and L. Liu, "Cognitive cubes: a tangible user interface for cognitive assessment," Proc. of Conference on Human Factors in Computing Systems (CHI '02), pp.347-354, 2002.
16. E. Sharlin, Y. Itoh, B. Watson, Y. Kitamura, S. Sutphen, L. Liu, and F. Kishino, "Spatial tangible user interfaces for cognitive assessment and training," Proc of Bio-ADIT 2004, pp. 410-425, 2004.
17. Y. Itoh, S. Akinobu, H. Ichida, R. Watanabe, Y. Kitamura, and F. Kishino, "TSU.MI.KI: Stimulating children's creativity and imagination with interactive blocks," Proc. of The Second International Conference on Creating, Connecting and Collaborating through Computing (C5), pp. 62-70, IEEE Computer Society, 2004.
18. T. W. Butler, "Computer response time and user performance," Proc. of Conference on Human Factors in Computing Systems (CHI '83), pp. 56-62, 1983.