UbiREMOTE: Framework for Remotely Controlling Networked Appliances through Interaction with 3D Virtual Space

Kohta Kiyokawa[†], Shinya Yamamoto [‡,¶], Naoki Shibata [§,¶], Keiichi Yasumoto [†,¶] and Minoru Ito[†]

[†] Nara Institute of Science and Technology, Ikoma,Nara 630-0192, Japan
[‡] Tokyo University of Science Yamaguchi, Yamaguchi 756-0884, Japan
[§] Shiga University, Hikone,Shiga 522-8522, Japan
[¶] Japan Science and Technology Agency, CREST

kota-k@is.naist.jp, shiny-ya@ed.yama.tus.ac.jp, shibata@biwako.shiga-u.ac.jp, yasumoto@is.naist.jp, ito@is.naist.jp

ABSTRACT

In this paper, we propose a framework named "UbiREMOTE" for controlling information appliances connected to a home network with a unified and intuitive user interface from a remote place. The UbiREMOTE framework provides users with a way to control appliances in a home through a virtual space drawn on a mobile terminal screen which reflects the latest conditions of the real appliances and the rooms in the home. With UbiREMOTE, a user controls appliances by (1) moving to the front of an appliance, (2) choosing the appliance to control and (3) pushing buttons on the virtual remote controller which imitates the real remote controller for the appliance or the real console. In this paper, we propose a method to improve the drawing speed of 3D virtual space on mobile terminals and a method for automatically reflecting condition changes of the real space in the virtual space. We implemented the methods and evaluated the performance. The results showed that the proposed methods can be practically used on small mobile terminals.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: Graphical user interfaces (GUI)

General Terms

Algorithms

Keywords

information appliances, mobile terminals, remote control, lightweight 3D rendering

MMSys'10, February 22–23, 2010, Phoenix, Arizona, USA.

1. INTRODUCTION

In recent years, appliances and information devices that can be connected to a network (*information appliances*, hereafter) have been becoming popular. In order to control information appliances via network, frameworks such as UPnP[1], DLNA[2], OSGi[3] and ECHONET[4] have been developed. One of the objectives of connecting information appliances to a network is to control those devices remotely. One might want to control an air conditioner at home or start recording a TV program before leaving the workplace. It would also be useful to remotely watch elders or the stuation in a house through a camera. Some commercial companies are providing these services[5]. However, most of the existing services/techniques for remotely controlling information appliances are text-based, which may not be intuitive and could sometimes be confusing if many devices are connected to the network.

In this paper, we propose a framework named "UbiREMOTE" for controlling many information appliances connected to a home network with a unified and intuitive user interface from a remote place. UbiREMOTE provides the following four functions: (i) an intuitive display of the real appliances and the rooms in the home through 3D virtual space graphics, rather than a text-based display, (ii) an interface for users to intuitively choose the appliances to control by moving the viewpoint in the virtual space, (iii) displaying the latest conditions and positions of appliances and the latest environmental conditions (such as temperature) in the room through the 3D view, and (iv) a mechanism to improve the drawing speed of the 3D virtual space on mobile terminals without a powerful 3D rendering capability.

We assume that UbiREMOTE is used with UPnP-capable information appliances. However, Simple Service Discovery Protocol (SSDP) used in UPnP requires IP multicast, which is not always supported by the Internet. In order to control appliances from a remote place with UPnP, UbiREMOTE encapsulates the UPnP communication with TCP packets.

In order to improve the drawing speed of 3D graphics on mobile terminals, we can use server-side rendering techniques in which 3D scenes are rendered in a server and converted to an MPEG movie, and played back on the terminal. However, delay could be problematic when interacting with the 3D graphics scenes. In the proposed method, we create

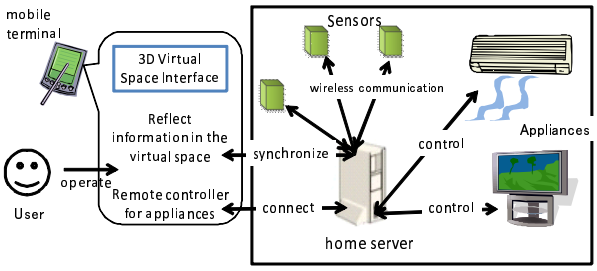

Figure 1: Internal Behavior of UbiREMOTE

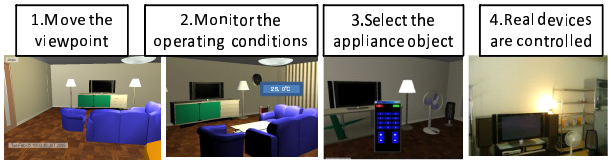

Figure 2: Operation on UbiREMOTE (Turning on a light)

many still pictures of 3D virtual space taken from various directions, and surround the viewpoint with these pictures. In this way, we can reduce the number of 3D objects rendered by the terminal.

As for (iii) reflecting the latest positions of appliances in the virtual space, we attach an accelerometer with a ZigBee communication device to each appliance, and detect the movement of appliances. We also discuss attaching several ZigBee devices on the walls of the room, and detect the positions of appliances based on the localization technique with the ZigBee devices. As for changes of environmental conditions (e.g., temperature or humidity), we discuss a successive synchronization technique to a UbiREMOTE terminal.

In order to evaluate UbiREMOTE, we implemented a UbiREMOTE remote controller prototype on a tablet PC, and conducted an experiment with three information appliances in a room. The experiment showed that the time required for the appliances to react to a user's operation is about 3 seconds, which is fast enough to control appliances from a remote place. We also confirmed that the proposed lightweight 3D rendering technique improved the drawing speed by about 90 times.

The remainder of this paper is organized as follows. Section 2 gives an overview and the challenges for implementing UbiREMOTE. Section 3 explains the technique for improving the drawing speed of 3D virtual space on mobile terminals. Section 4 explains the method for reflecting the changes of real appliances on the virtual space. Section 5 presents the implementation. We give the experimental evaluation and the related work in Sections 6 and 7, respectively. Finally, we conclude the paper in Section 8.

2. UBIREMOTE OVERVIEW

In this section, we first describe the target environment and the objective of the UbiREMOTE framework, and clarify the desirable features and the basic structure. Then, we give the structure of the proposed framework.

2.1 Target Environment and Objective

UbiREMOTE framework handles a service that allows a user to control information appliances in the home from a remote place (*remote controller service*, hereafter). The operating environment of UbiREMOTE is shown in Fig. 1. We suppose that all of the target appliances are connected to the network, and a home server can be connected from the Internet.

The objective of the UbiREMOTE framework is to satisfy the following conditions upon providing the remote controller service.

(1) Everyone can intuitively control appliances.

(2) Appliances can be controlled from a remote place via the Internet.

(3) Operating conditions of appliances can be monitored.

(4) Room conditions can be monitored.

As for (1), in order to solve the low usability in text-based remote control methods of home appliances, we reproduce the space with networked home appliances (target space) as a 3D virtual space, in which users can control and monitor the appliances. When users interact with the virtual space, we let them freely move the viewpoint in the virtual space by operating the remote terminal, as if they were in a real home. For this purpose, as shown in Fig. 2, we provide a user interface in which a user moves the viewpoint to the front of the target appliance, selecting the appliance object to pop up a control window, and operating the appliance through the window. This feature provides users an intuitive and familiar way of controlling appliances by using a control window imitating the real remote controller for each appliance.

As for (2), we target a home network and appliances based on UPnP (Universal Plug and Play) and provide a mechanism to control each appliance from a remote place with a UPnP-based protocol.

As for (3), we reduce the labor of users in checking and configuring a lot of appliances distributed around the home. In order to achieve this, we provide functions for monitoring the operating conditions of appliances through the virtual space displayed in the user's terminal.

As for (4), we allow the users to grasp the condition of the real space by indicating the positions and the operating conditions of appliances and room conditions such as temperature. In order to achieve this, we provide functions for measuring positions of appliances and room condition and displaying the latest measurement results. With these functions, users can intuitively control the temperature settings of air conditioners before leaving their workplaces or check if they turned off heaters after they left home.

2.2 Structure of UbiREMOTE

As in Fig. 1, the mobile terminal controls information appliances through the home network. The sensors in the home measure environmental conditions such as temperature and humidity in the rooms, and positions and operating conditions of the appliances. These data are collected in the server and reflected in the mobile terminal screen through the synchronization of the data between the server and the mobile terminal.

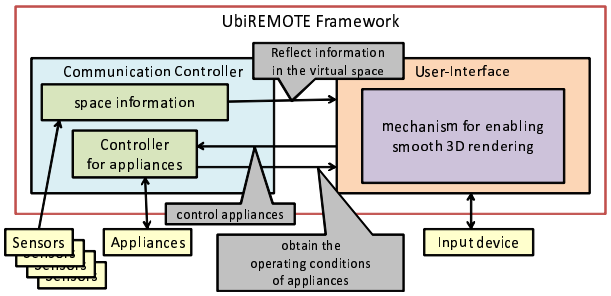

Figure 3: Structure of UbiREMOTE framework

Figure 4: Turning on a light by control window

The UbiREMOTE framework roughly consists of (1) a user interface module and (2) a communication controller module, as shown in Fig. 3.

2.2.1 User Interface

This module has the following three functions: (1) lightweight 3D rendering, (2) viewpoint movement and selecting an appliance in the virtual space, and (3) controlling and monitoring the selected appliance.

As for (1), we display the arrangement and the operating conditions of the appliances (light, TV and air conditioner) in the target space with 3D graphics from any viewpoint on the remote terminal screen. We provide a function for a user to move the viewpoint in the virtual space as if he/she is walking in the real house, and view the space with appliances from any angle. We also provide a function for collaborating with the communication controller module to obtain the operating conditions (e.g., on-off status of power, target temperature/humidity of air-conditioner, channel/volume settings of TV, etc.) of each appliance and room conditions sensed by sensors, and to reflect the obtained information in the virtual space. For example, if the viewpoint is moved and it is looking at a light in the living room, the communication controller module communicates with the real light device and obtains its operating conditions (on/off, or brightness setting, etc.) and a scene showing that the light is turned on is rendered as shown in Fig. 4.

UbiREMOTE is used with a small mobile phone with a touch panel (like iPhone). However, displaying the virtual space requires powerful 3D graphics processing capability, and thus making mobile phones render the virtual space smoothly is difficult. We built a mechanism for enabling smooth 3D rendering on these small mobile phones without powerful hardware. We will describe this mechanism in Section 3.

As for (2), we create a mechanism for users to move the viewpoint or select an appliance by using the touch panel on the phone.

As for (3), we create a mechanism for users to control an appliance through a popup control window imitating the real remote controller displayed on the mobile terminal. Fig. 4 is a snapshot of when a control window has popped up.

2.2.2 Communication Controller

This module has the following three functions: (1) communicating with the server and an appliance in the target space (home), and controlling the appliance as the user specifies, (2) obtaining the operating conditions of appliances and room conditions sensed from the sensors in the appliances, and (3) obtaining the room conditions or the positions of the appliances from the multiple sensors deployed in the room. We describe the details of (3) in Section 4.

3. LIGHTWEIGHT 3D GRAPHICS PROCESSING IN UBIREMOTE

In this section, first we give requirements for 3D graphics processing in a lightweight computing device, and propose basic ideas. Then, we describe the details of our proposed method.

3.1 Requirements

We set the following requirements for lightweight 3D graphics processing.

(R1) The view of the 3D virtual space should be smoothly drawn and updated in a lightweight computing device.

(R2) Perceived quality of the resulting view should not be greatly spoiled by lightweight 3D graphics processing.

(R3) Response time until the latest view of the target space is reflected should be reasonably short.

For (R1), we need a technique to simplify 3D graphics rendering that matches the 3D walkthrough-type interface. For this purpose, most 3D graphics processing should be moved from a client to a server. For (R2), we need to reduce 3D graphics quality for lightweight processing, but we also need to maintain sufficient usability. 3D graphics quality should be reduced for lightweight processing while we maintain sufficient usability. For (R3), when a user operates an appliance, its status should be immediately reflected in the view of the virtual space displayed on the client's screen.

3.2 Basic Ideas for Lightweight 3D Rendering

First, we consider a server-client based lightweight 3D processing technique as shown in Fig. 5, where a server renders the 3D graphics of the virtual space including all objects, captures a 2D image of the graphics from a specified viewpoint, and sends the image to a client. This technique allows

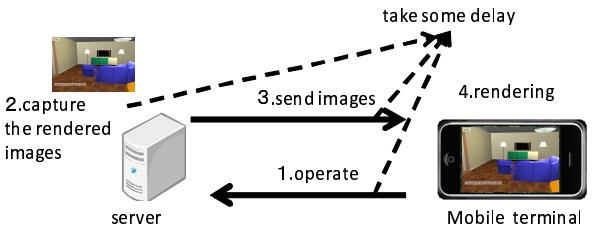

Figure 5: Server-client based 3D graphics processing

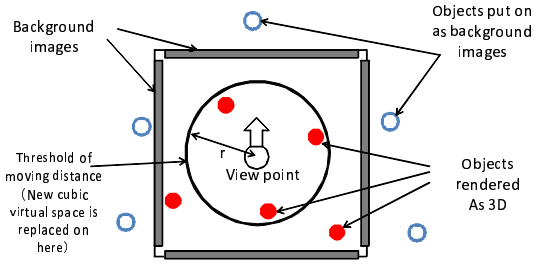

Figure 6: Cubic virtual space

the client to draw the 3D virtual space without 3D graphics processing. However, it takes time to create and transmit a 2D image to the client. Such delay occurs in the following cases: (1) when the user changes viewpoints or directions in the virtual space; and (2) when the user operates an appliance and the appearance of its status in the virtual space should be updated. Since the above technique generates some delay whenever the user moves or operates appliances, the walkthrough-type interface cannot be achieved effectively.

In order to cope with requirement (R1), we adopt a technique that creates 2D images corresponding to multiple views from a viewpoint with different directions for the virtual space, and sends the created 2D images to the client (see Fig. 6). This technique reduces the graphics update delay as well as the number of 3D objects that must be drawn in the client, since the 2D images surround the viewpoint, and therefore the 3D graphics re-rendering is not required even when the user moves the viewpoint or direction in the virtual space. For requirement (R2), we adopt a technique in which the client renders the 3D graphics of only appliances near the viewpoint. The reason is that a user is likely to move the viewpoint to near an appliance when he/she wants to operate it. It would also be better to draw objects near the viewpoint as 3D graphics, since the 2D image of nearby objects tends to be distorted by changing view direction.

The above techniques for coping with (R1) and (R2) are illustrated in Fig. 7. The client draws only a small number of 3D objects in real time within its 3D graphics processing power, and the server draws the virtual space with other 3D objects, creates 2D background images, and sends the images to the client. Then, the client composes the 2D background images and the 3D graphics of some of the objects into one image to be shown on the screen. We call the resulting image drawn by this technique *pseudo 3D virtual space*,hereafter.

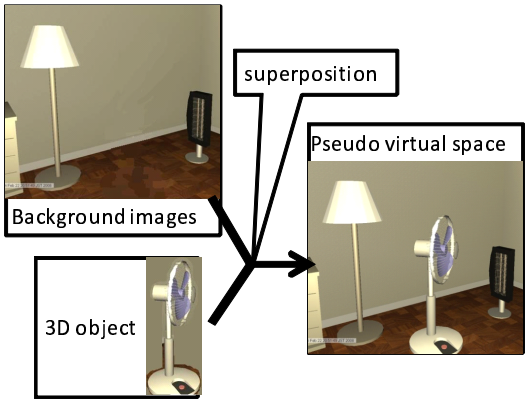

Figure 7: Superposing 3D object on 2D background image

In the following subsections, we describe details of the proposed technique to draw and update the pseudo 3D virtual space.

3.3 Pseudo Virtual Space Drawing Technique

As shown in Fig. 6, the proposed technique uses a virtual cube with a certain size that surrounds a viewpoint. The objects in the cube are drawn as 3D graphics and the other objects outside the cube are drawn as six background images put on the cube faces. The cube size is determined in advance, taking into account the 3D graphics processing power of the client machine. When too many objects are included in the cube, the cube size is reduced.

3.4 Updating Pseudo Virtual Space

In order to maintain the perceived quality of the pseudo virtual space, we update the background images put on cube faces whenever the viewpoint moves a specified distance. That is, as shown in Fig. 8, we consider the circle with radius r at the center of the cube, and we create the new background images for the new viewpoint when the viewpoint moves outside the circle. The server receives the new viewpoint and direction, and creates new background images for the cube faces. Here, the radius r is determined, taking into account the client's 3D graphics processing power, graphics update delay, etc.

As explained above, the pseudo virtual space is smoothly drawn and updated on the client's screen. When the user operates an appliance inside the cube, its status is immediately reflected on the screen since the client draws the object by itself. However, the status of objects outside the cube are not reflected until the background images for cube faces are updated. This would not be a big problem since the user should move the viewpoint near the target appliance before operating it.

Fig. 9 shows the message sequence between the server and the client. When the viewpoint moves outside the circle (moving distance is more than r), the client sends an update request with the new viewpoint and the direction to the server. When the server receives the request, it creates the background images and sends them with 3D object data inside the new cube back to the client. Finally, the client displays the new pseudo virtual space on its screen by drawing the 3D objects inside the cube on the new background images.

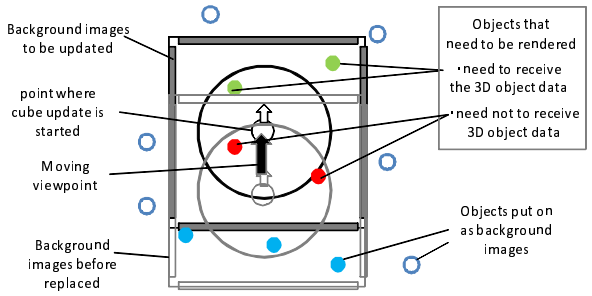

Figure 8: Updating pseudo virtual space according to viewpoint movement

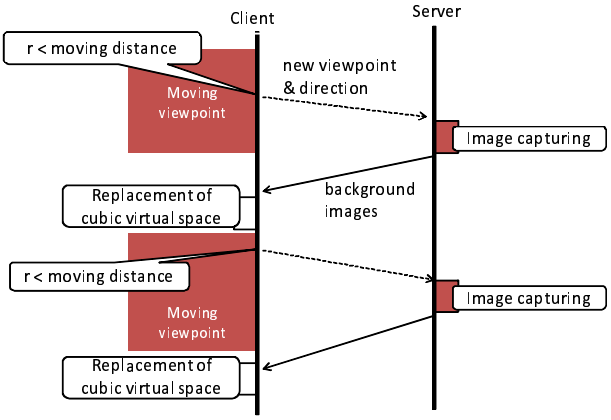

Figure 9: Message sequence between server and client

4. REFLECTING REAL SPACE STATUS INTO VIRTUAL SPACE

In this section, first we give requirements for reflecting change in the real space into the virtual space, then describe the proposed method for reflecting changes of the physical quantities and the appliance locations in the real space into the virtual space.

4.1 Requirements

In a real space, locations and operating conditions of appliances are likely changed by inhabitants' manual operations. If the virtual space information does not reflect the real space status, an unexpected result may be produced when a user operates an appliance. For instance, suppose that a user wants to turn on an air conditioner to cool a room when the room temperature shown in the virtual space is high. However, if another user has already turned on the air conditioner by manual operation and the room temperature has already been regulated to a comfortable degree, further operation to the appliance will be unnecessary. Also, if the location of an appliance has been changed but is not reflected in the virtual space, then an unexpected result may occur. For instance, a user turns on a heater that seemed to be in a safe location in the virtual space, but the heater has been moved near a curtain in the real space.

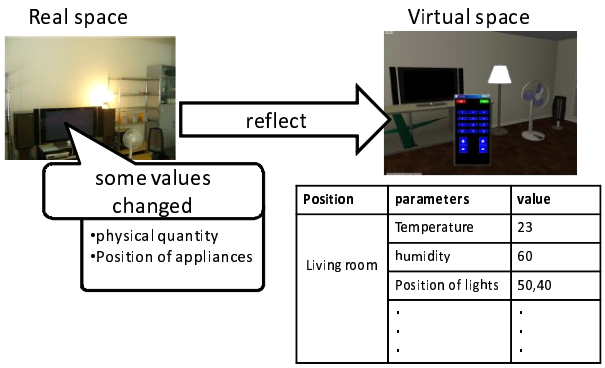

Figure 10: Reflecting real space information into virtual space

Accordingly, the following information should be reflected in the virtual space:

1. actual room conditions such as temperature and humidity in each room,
2. actual location of each appliance, and
3. actual operating condition of each appliance.

In the following subsections, we propose techniques to reflect the above information about the real space into the virtual space displayed on a terminal, as shown in Fig. 10.

4.2 Reflecting room conditions

We deploy some sensor nodes in the real space that sense temperature, humidity, illuminance, etc. and communicate with a home server via ZigBee[6]. We can use SunSPOT as a sensor node, which can sense temperature, illumination, and acceleration at the location where it is deployed. Multiple SunSPOTs can form a wireless sensor network and exchange the sensed data with each other. It is also easy to add a new node or remove an existing node to/from an existing network. Since ZigBee's wireless transmission range is between 10m and 75m, we deploy one SunSPOT in each room. Even in this case, multiple SunSPOTs deployed in different rooms can communicate with each other.

We suppose that the home server is equipped with a ZigBee device and thus can communicate with the deployed SunSPOTs. Each SunSPOT sends the sensed value with its ID to the home server when it detects a certain change between the current value and the previous value shown in the virtual space for each target monitoring item. For example, if the room temperature changes by 2 degrees (this threshold can be specified by the user), the SunSPOT in the room sends the new temperature to the home server so that the value is sent to the client and reflected in the virtual space on the client screen as shown in Fig. 11.

4.3 Detecting and reflecting change of appliance location

Location of an appliance in the virtual space is based on the initial locations given beforehand. When an appliance is moved, it should be accelerated. By attaching SunSPOT to each movable appliance (e.g., TV, cooling fan, heater,

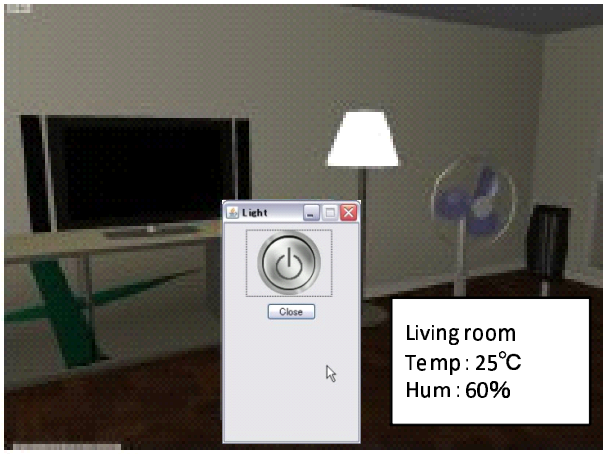

Figure 11: Displaying latest real space information on virtual space

etc.), we can automatically detect the moving action of the appliance. When acceleration disappears for a certain time interval, we regard that the appliance is placed at a new location and fixed. Then, we apply the localization technique to estimate its new position. For example, we can use a technique proposed in [7] that uses ZigBee nodes. In each room, we deploy several anchor nodes with their accurate positions at the fixed location on a wall, ceiling, etc. In localization, when appliance movement is detected, each anchor node sends a beacon packet to the SunSPOT attached to the target appliance. Then the SunSPOT measures RSSI (received signal strength indicator) of the packet for each anchor and sends the result to the home server. By using the RSSI information, the server determines the new appliance position and subsequently updates the position in the virtual space.

5. UBIREMOTE IMPLEMENTATION

In this section, we describe an example case to implement the remote controller software based on the UbiREMOTE framework on a tablet PC with Windows XP Tablet PC Edition. We implemented the software in Java and Java Runtime Environment 1.6.5. We used OpenGL 2.1 (JOGL) as a 3D graphics processing environment and CyberLink for Java [8] as a UPnP library.

5.1 Implementation of user interface module

This module achieves the 3D virtual space drawing function and the intuitive user interface function.

The 3D virtual space drawing function visualizes the dynamic behavior of the appliances and the change of physical quantities in the real space by changing the shape and/or color of the appliances, lighting, etc. in the 3D virtual space. We draw the virtual space on the terminal screen from the first person view.

We implemented the lightweight 3D rendering mechanism proposed in Sect. 3. Fig. 12 depicts the virtual space appearance from the inside (left figure) and the outside (right figure) of the cube. To create the proposed lightweight 3D rendering mechanism, we need to implement mechanisms to (1) draw the pseudo virtual space, (2) appropriately update

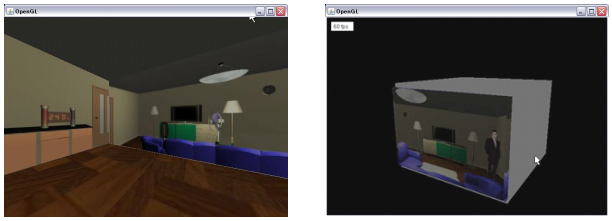

Figure 12: Virtual space appearance from inside (left) and outside (right) of cube

background images put on cube faces, and (3) download the 3D object data of each appliance inside the cube. Mechanism (1) is implemented by letting the home server render the image of the virtual space and send it as a JPEG image to the remote controller terminal via HTTP. We used an image of 220×320 pixels for the side faces of the cube and 128×128 for the upside (ceiling) and the downside (floor) faces. For mechanism (2), as explained in Sect. 3.4, we specify a circle with radius r (we used 1.5 m in the implementation) in the cube and let the terminal send an update request when the viewpoint moves out of the circle. After sending the request, the terminal waits to receive all images for the new cube. When all the images are received, it changes the position and the direction of the cube and replaces the images on the cube faces. For mechanism (3), we did not implement the dynamic object data download mechanism in this version. Instead, we let the terminal have all object data beforehand.

As the intuitive user interface function, we implemented a mechanism that allows the user to select an appliance and to move the viewpoint using the features of the touch panel screen interface. We implemented the following functions:

- changing the view direction or moving the viewpoint by touching a specific area on the screen,
- selecting an appliance just by touching its graphic on the screen
- showing a popup control window dedicated to each appliance and operating the appliance by touching the buttons/switches on the window.

In Fig. 13, we show example operations for turning on a lamp. Initially, the lamp is turned off as shown in the upper part of Fig. 13 (a). First, the user touches the lamp object on the screen, then a control window for the lamp pops up (the bottom of Fig. 13(a)). By tapping the power button on the window, the lamp in the real space and that in the virtual space are turned on as shown in Fig. 13(b). The lamp is turned off by similar operations. Other kinds of appliances can also be operated in a similar way. For instance, if a user wants to change the channel of a TV from channel 1 (soccer) to channel 2 (news), the user just touches the screen area, causing the TV object to display the TV's control window, and taps the button representing channel 2.

As shown in the above example, UbiREMOTE allows users to operate appliances with intuitive user interfaces. When we want to add a new appliance so that it can be operated by UbiREMOTE, we simply prepare 3D object data for the appliance and its control window.

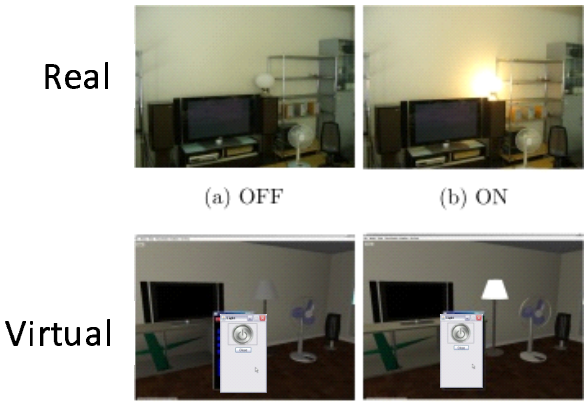

Figure 13: Operation for turning on/off lamp

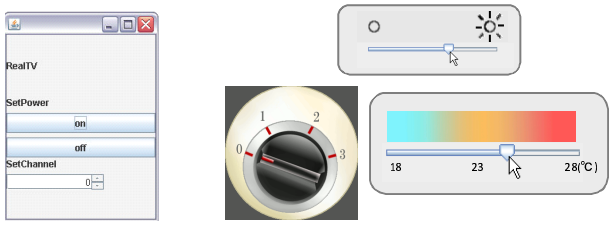

Figure 14: Automatically generated control window (left-side) and customized windows (right-side)

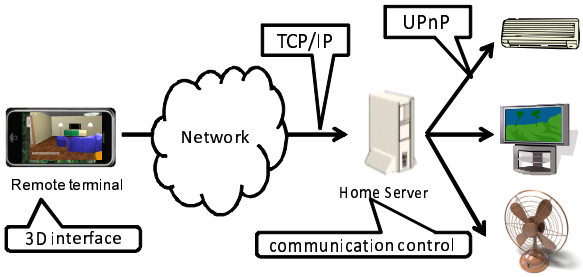

Figure 15: Communication architecture

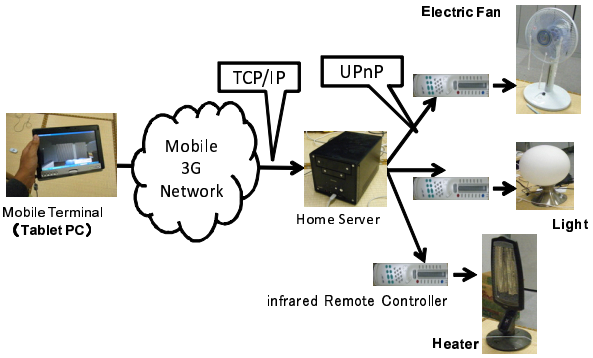

Figure 16: Experimental configuration

The control window of each appliance can automatically be generated based on changeable parameters. In this case, the user needs to customize the window according to the controllable parameters of the appliance and the user's preference as shown in Fig. 14. By defining the control window for each sensor device deployed in the real space, we can show the sensed value to the user through the control window.

5.2 Implementation of communication controller module

In order to operate appliances via the Internet, we implemented this module to communicate with appliances via the home server. The communication architecture is shown in Fig. 15. The user interface module and the communication controller module are executed on the remote controller terminal and the home server, respectively.

To allow the client terminal to communicate with UPnP-ready appliances based on UPnP protocols, we create a TCP/IP tunnel between the client and the server that encapsulates the UPnP communication. We implemented this tunneling mechanism by using the network simulator used in the UbiREAL simulator [9]. The network simulator provides APIs of the TCP/IP protocol stack that are compatible with `java.net` so that virtual network devices executed on the simulator can communicate with each other as if they were connected to the same LAN. The provided APIs can be invoked from a remote computer through TCP/IP so that any protocols including UPnP protocols can be executed on the simulator. It also provides a gateway function between virtual devices (on the simulator) and real networked devices. This technique enables the remote control of UPnP appliances through the Internet where UPnP protocols are not routed in general.

The client-side software (UI module) can monitor the IDs, allowed operations, operating conditions, etc. by the UPnP functionality. When a user initially executes the software on the terminal, it receives from the home server the information of the viewpoint, background images for the cube, position and object data for each appliance inside the cube, and draws the pseudo virtual space on the terminal screen. When a user remotely operates an appliance, the terminal invokes UPnP methods to operate the target appliance and also changes/redraws the graphics of the corresponding 3D object on its screen.

6. EVALUATION

In this section, we evaluate the proposed framework by measuring the time for appliances to react to the operation by a remote terminal. We also evaluate the performance improvement by the proposed lightweight 3D rendering technique.

6.1 Response Time

We first measured the response time when operating appliances by a remote controller terminal (referred to by *client*, hereafter). The setting for this experiment is shown in Fig. 16.

We evaluated two cases: (a) operating appliances in the home where they are placed, and (b) remotely operating appliances from a remote place via the Internet and a cellular network. In case (a), the client is connected to the home

Table 1: Delay in controlling by UbiREMOTE

appliances		delay for controlling with infrared controller	overall delay with UbiREMOTE	
			(a)WLAN	(b)cellular
electric fan	ON	0.540 sec	0.94 sec	3.22 sec
	OFF	0.550 sec	0.92 sec	3.04 sec
light	ON	0.133 sec	0.73 sec	2.98 sec
	OFF	0.204 sec	0.61 sec	3.13 sec
heater	ON	0.413 sec	0.93 sec	3.18 sec
	OFF	0.440 sec	0.84 sec	3.30 sec
average		0.380 sec	0.828 sec	3.141 sec

Table 2: Frame rate

	number of appliance objects (in cube)	polygons	frame rate (fps)
method to draw all 3d objects	20	200000	1
proposed method	20(0)	108	120
proposed method	20(1)	2100	100
proposed method	20(2)	4208	90

network using a wireless LAN (IEEE 802.11b). In case (b), we used a 3G cellular network to connect the client to the home server via the Internet. The home server is connected to the Internet by a wired LAN in both cases.

We used a ThinkPad X61 notebook PC with Core 2 Duo L7500 1.60GHz, 2GB of memory, and Intel(R) 965 graphics processor as a client terminal. We used a desktop PC with Core 2 Quad 2.4GHz, 3GB of memory, and Geforce 8800GT GPU as a home server. Since only few UPnP-capable information appliances are available on the market, we used a USB infrared remote controller to actually control each appliance. We executed a network simulator that tunnels UPnP-based protocols between the client and the server, software for making each appliance visible/controllable as a UPnP device, and the server-side software on the home server.

We measured the time for each appliance to react to ON/OFF operations by a user. We compared the measured time with the case where the appliances are operated using an ordinary infrared remote controller.

The average response time of 10 trials is shown in Table 1. The result shows that appliances can be controlled in 0.83 sec. via a wireless LAN, and 3.14 sec. via a cellular network. Since the time includes the delay by an infrared remote controller, the actual overhead of UbiREMOTE is 0.45 sec. and 2.76 sec, respectively. The time is fast enough to control appliances from a remote place.

6.2 Rendering Speed

6.2.1 Data Size

We captured images of the virtual space and measured the image size. We determined the size of the image to be 220×320 pixels according to the display size of an ordinary mobile terminal. We used 128×128 pixel images for the ceiling and the floor of the cube, since we do not need fine images for those cube faces. All these images are compressed by JPEG, and the total size of six images for the cube was less than 80KB.

6.2.2 Time for Downloading Images

We measured the time for downloading images by a mobile terminal. We uploaded 300-KB images to a web server using a SoftBank 930SH cell phone. This cell phone has the capability to communicate with HSDPA. We tried downloading the images 10 times from a place with a good radio condition. The average download rate (throughput) was 779.2 Kbps. From the measured data size in the previous section, the time to download images would be 0.62 sec, which is fast enough for prefetching images in cube update.

6.2.3 Frame Rate

We measured the frame rate for rendering the virtual space. In this experiment, we measured the screen update frequency (frame rate) of OpenGL in the case of no communication. The rendering speeds for the method that draws all 3D objects and the proposed method when 1 - 3 objects are included in the cube are shown in Table 2. In this experiment, we used the tablet PC with Microsoft Windows XP Professional, Core 2 Duo, 2.0GB of memory and Mobile Intel(R) 965 Chipset.

Compared with the method that draws all objects, the rendering speed was improved to a great extent in the proposed method. The frame rate was high enough even when we added two objects (each has about 2,000 polygons) in the cube.

6.3 Appearance

We checked the feeling of discomfort by the appearance of the pseudo virtual space. We asked 5 graduate students with ages between 22 and 25 to answer the 3 questions below after walking around in the pseudo virtual space drawn by the proposed method.

(1) Whether he/she felt strange or not

All participants answered that they did not feel anything wrong if the viewpoint was close to the center of the cube. They also answered that the 3D objects placed within the cube, as shown in Fig. 17, looked natural. However, they felt strange to some extent when the viewpoint was far from the center of the cube, as shown in Fig. 18.

(2) Whether operations were comfortable or not

All of them answered that they had no problem.

(3) Whether he/she could distinguish each appliance from others or not

All of them answered that it was basically OK, but one of them answered that it was hard to identify distant appliances.

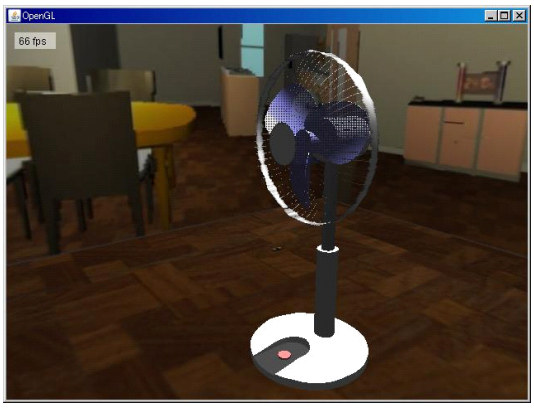

Figure 17: Inside-cube object in pseudo virtual space

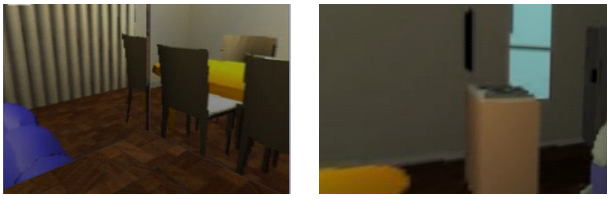

Figure 18: Appearance of pseudo virtual space from near corner of cube (left) and from position near wall (right)

6.4 Observation

Through experiments, we confirmed that the downloading time and rendering speed were satisfactory, and the data transfer via a cellular network did not have a large delay.

Also, we confirmed that the appearance of the graphical interface did not spoil the usability so much. Although the testees pointed out that they felt strange when the viewpoint was far from the center of the cube, we can cope with this problem by adjusting the cube size when the viewpoint gets too far from the center.

7. RELATED WORK

In this section, we address existing studies related to the UbiREMOTE framework and the proposed lightweight 3D rendering technique, according to the following two categories: remote control techniques for information appliances and lightweight 3D rendering techniques.

7.1 Remotely Controlling Information Appliances

So far, some studies have been proposed for remotely controlling multiple information appliances over a network.

Smetters et al. proposed a software environment named Instant Matchmaker for controlling information appliances[10]. This environment mediates between a user and appliances, and he/she can remotely turn on/off appliances with a mobile phone. To improve reliability, only registered mobile phones can be used for remote control.

Toshiba Corporation proposed a service named TOSHIBA FEMINITY[11] that allows users to control information appliances using a PC or a mobile phone from a remote location. This service is realized by Bluetooth communication between a home server and appliances. With this service, a user usually controls an appliance through a function menu, which does not imitate the dedicated remote controller for the appliance. Therefore, it may be sometimes difficult for users to understand which appliance he/she is currently controlling, and he/she may control the wrong appliance.

Nakamura et al. proposed a protocol called Remote Appliance Control Protocol (RACP) for controlling information appliances over a network[12]. They evaluated the overhead and the usability when a user remotely controls an ordinary appliance using an infrared remote controller with RACP. However, they have not evaluated the cases of operating appliances connected to a home network. Also, this protocol is not intended to improve the usability for appliance operations.

These traditional studies for remotely controlling information appliances assume that users use a simple interface that does not imitate the real controller. This makes these methods difficult to use for inexperienced users.

7.2 Light-Weight 3D Rendering Techniques

The existing techniques for reducing the computational load for 3D rendering can be roughly classified into three categories: server-side rendering, 3D geometry data compression, and video streaming.

7.2.1 Server-side rendering

Server-side rendering methods use image warping techniques. Mark et al. proposed a server-side rendering method for displaying a 3D image when a user moves a viewpoint in an interactive 3D rendering like virtual space walk-through[13]. In this method, when the server receives the new viewpoint information, it creates an image for the 3D space from this viewpoint and sends the image to the client.

Chang et al. proposed a method using a similar technique for a light-weight client such as a PDA[14]. However, it is difficult to achieve a function to choose and operate a 3D object of a scene in a short response time, since this method displays only a 2D reference image.

Thomas et al. proposed another server-side rendering method[15]. In this method, the position and the direction for capturing images are optimized to construct the virtual space with a small number of images. The server renders a new scene only when the user moves into a region where the images are not prepared. While this method saves required network bandwidth, new images have to be rendered every two to four frames. Furthermore, the camera selection algorithm influences the appearance of the virtual space since the amount of error in the warping process is influenced by the algorithm.

7.2.2 3D Geometry Data Compression

There are many studies on reducing the number of polygons in 3D geometry data[16]. However, data compression is inevitable for our purpose since the available wireless bandwidth is limited.

Aliaga et al. proposed a software architecture for walkthrough between buildings in a virtual space[17]. In this method, the amount of data is reduced by searching for similar images from the previously transmitted images. How-

ever, it is difficult to use this method in our framework since many images have to be transmitted to a client beforehand.

7.2.3 Video Streaming

The video streaming technique is another client/server technique in which the server renders images for the 3D space from the user's viewpoint and transmits the images to a client as video data. Some of the well-known video codecs support video streaming[18, 19, 20]. It is possible to capture 3D scenes and encode them to a video of some codec at a server, and transmit the video from the server to a client. Winter et al. proposed a method based on video streaming to reduce the processing load for thin-clients[21]. However, this method supposes to use a high-speed wired network of 100Mbps for the high-resolution video streaming that is basically impossible in a mobile phone. Also, there is the problem of a large response time for each user's operation of an appliance.

8. CONCLUSION

In this paper, we proposed the UbiREMOTE framework for remotely controlling multiple information appliances connected to a home network with a unified and intuitive user interface. We also proposed a lightweight 3D graphics processing technique for mobile terminals to improve the drawing speed of a 3D virtual space and a technique for automatically reflecting changes in the target real space to the virtual space. UbiREMOTE allows inexperienced users to intuitively operate and monitor information appliances through the Internet anytime from anywhere. We implemented a UbiREMOTE prototype on a tablet PC and conducted evaluation experiments. As a result, we confirmed that the proposed techniques could achieve remote controlling and monitoring of information appliances with a small delay. Moreover, the proposed light-weight 3D rendering technique achieved sufficient rendering speed to be used in practical environments.

As part of future work, we will evaluate the usability of the proposed 3D virtual space interface by comparison with existing user interfaces in terms of the consumed time and the error rate in operating appliances. Also, we are planning to implement the proposed methods on mobile phones such as iPhone and evaluate the performance.

9. REFERENCES

[1] The UPnP Forum : UPnP Forum, http://www.upnp.org/.

[2] Digital Living Network Alliance: Digital Living Network Alliance, http://www.dlna.org/.

[3] OSGi Alliance: OSGi Alliance, http://www.osgi.org/.

[4] ECHONET CONSORTIUM: ECHONET CONSORTIUM, http://www.echonet.gr.jp/.

[5] NTT Neo Service: U-Concent service, http://www.ntt-neo.com/news/2007/070419.html/.

[6] ZigBee Alliance: ZigBee, http://www.zigbee.org/.

[7] Gonçalo, G. and Helena, S.: "Indoor Location System using ZigBee Technology," *Proc. of 3rd Int'l. Conf. on Sensor Technologies and Applications (SENSORCOMM 2009)*, pp. 152–157 (2009).

[8] Konno, S.: "CyberLink Development Package for UPnP Devices," http://cgupnpjava.sourceforge.net/.

[9] Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K., and Ito, M.: "UbiREAL: Realistic Smartspace Simulator for Systematic Testing," *Proc. of UbiComp2006*, LNCS4206, pp. 459-476 (2006).

[10] Smetters, D.K., Balfanz, D., Durfee, G., Smith, T.F., and Lee,K.H.: "Instant Matchmaking: Simple and Secure Integrated Ubiquitous Computing Environment," *Proc. of 8th Int'l. Conf. on Ubiquitous Computing (UbiComp 2006)*, LNCS4206, pp. 477–494 (2006).

[11] Masao, S., Shunro, K., and Morio, H.: "Extension of FEMINITY (TM) Series Home Network System for Toshiba Network Home Appliances," *TOSHIBA REVIEW*, Vol.57, No.10 (2002). http://www.toshiba.co.jp/tech/review/2002/10/index.htm.

[12] Nakamura, M., Tanaka, A., Igaki, H., Tamada, H., and Matsumoto, K.: "Adapting Legacy Home Appliances to Home Network Systems Using Web Services," *Proc. of Int'l. Conf. on Web Services (ICWS 2006)*, pp.849–858 (2006).

[13] Mark, W.: "Post-Rendering 3D Image Warping: Visibility, Reconstruction, and Performance for Depth-Image Warping," *Technical Report: TR99-022 of University of North Carolina at Chapel Hill* (1999).

[14] Chang, C. and Ger, S.: "Enhancing 3D Graphics on Mobile Devices by Image-Based Rendering," *Proc. of 3rd IEEE Pacific Rim Conf. on Multimedia*, pp. 1105–1111 (2002).

[15] Thomas, G., Point, G., and Bouatouch, K.: "A Client-Server Approach to Image-Based Rendering on Mobile Terminals," *Technical Report RR-5447 of INRIA* (2005).

[16] Shikhare, D., Babji, S.V., and Mudur, S.P.: "Compression techniques for distributed use of 3D data," *Proc. of 15th Int'l. Conf. on Computer Communication*, pp.676 – 696 (2002).

[17] Aliaga, D., Rosen, P., Popescu, V., and Carlbom, I.: "Image warping for compressing and spatially organizing a dense collection of images," *Signal Processing: Image Communication*, Vol. 21, Issue 9, pp.755–769 (2006).

[18] ISO/IEC: "Information technology –Coding of audio-visual objects– Part 2: Visual," *ISO/IEC 14496-2:2001(E)* (2001).

[19] ITU-T: "Recommendation H.264: Advanced video coding for generic audiovisual services," *also an ISO standard as ISO/IEC 14496-10* (2005) .

[20] BBC: "Dirac Specification, Version 2.2.0," http://dirac.sourceforge.net/DiracSpec2.2.0.pdf.

[21] De Winter, D., Simoens, P., and Deboosere, L.: "A hybrid thin-client protocol for multimedia streaming and interactive gaming applications," *Proc. of 2006 Int'l. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2006)* (2006).