Scalable Architecture and Content Description Language for Mobile Mixed Reality Systems

Fumihisa Shibata, Takashi Hashimoto, Koki Furuno,
Asako Kimura, and Hideyuki Tamura

Graduate School of Science and Engineering, Ritsumeikan University
1-1-1 Nojihigashi, Kusatsu, Shiga 525-8577, Japan
fshibata@is.ritsumei.ac.jp

Abstract. We propose a new scalable architecture for mobile mixed reality (MR) systems and a content description language to be used in such architecture. Several architectures already exist to realize mobile MR systems, however, most of them are device specific. The architecture we propose here is able to accommodate a variety of devices, from mobile phones to notebook PCs. We have already designed a concrete specification for our architecture and content description language. We have also confirmed their viability by implementing several applications on various mobile devices.

Keywords: Mixed Reality, Augmented Reality, Scalable Architecture, Content Description Language, Mobile Computers.

1 Introduction

Outdoor application of augmented reality (AR) and mixed reality (MR) systems using mobile devices is expected to be a rich field [1]. Though there are some developments that implement AR/MR functionality on mobile phones and PDAs, their performance is inadequate [2]-[5]. In general, the main reason for this is that today's mobile devices have neither enough computational power nor memory.

These limitations could be eliminated soon. Mobile phones will become multifunctional, and we will soon have Ultra-Mobile PCs (UMPC). However, conventional desktop and laptop PCs probably may offer more performance than such UMPCs. AS a result, we need to consider frameworks that can overcome the performance differences between various mobile devices, instead of hastily attempting to implement AR/MR functions on a few low-performance devices. We must ensure that we accommodate future device developments, as well as existing performance differences between types and models of devices.

Based on these principles, we proposed a scalable architecture for various mobile MR systems and a core content description language "SKiT-XML." Our architecture is based on a client-server model that distributes the functionalities required to realize MR systems to the clients and a server. The architecture's design and implementation is explained in [6] and in the panel of [1].

Since then, our architecture has improved. Now, most of the primary functions have been implemented. We also tested the effectiveness of our scalable architecture

Z. Pan et al. (Eds.): ICAT 2006, LNCS 4282, pp. 122 – 131, 2006.

using various mobile devices and multiple applications. This paper provides an overview of the architecture and language as well as the applications used for confirmation.

In Section 2, we present related work. Section 3 describes our scalable architecture. In Section 4, we present our content description language, which describes information exchanged between the server and the clients. Section 5 presents the experimental results of three application types based on our architecture. Finally, Section 6 summarizes this paper and describes future directions.

2 Related Work

Some previous works in virtual and augmented reality have addressed a framework or a toolkit for developing VR/AR systems. MacIntyre and Feiner proposed a toolkit, called Columbia object-oriented testbed for exploratory research in interactive environments (COTERIE), which provides language-level support for building distributed virtual environments [7]. Feiner *et al.* also developed a Touring Machine [8] based on COTERIE. The Touring Machine is an application that provides information about Columbia's campus. COTERIE realized a shared, distributed database, and handles various trackers. Bauer and his colleagues proposed distributed wearable augmented reality framework (DWARF), which is a CORBA-based framework that allows rapid prototyping of distributed AR applications [9]. DWARF consists of reusable distributed services such as trackers, middleware to dynamically match these services, and extensible software architecture. Unfortunately, this work does not apply to mobile phones. Mobile phones with digital cameras are widespread and no other device seems to be an ideal candidate as a handy AR/MR platform. Although current mobile phones do not have sufficient CPU power to display videos and calculate camera parameters, AR/MR on mobile phones will be used in the near future. Accordingly, it is important for AR/MR frameworks to support mobile phones.

Miyamae *et al.* proposed a navigation platform for wearable AR/MR systems [10]. Miyamae's platform can address a wide variety of navigation scenarios, based on event, condition, and/or action-driven navigation systems. However, this platform is not versatile, as only navigation applications for wearable computers are assumed in this platform.

A software framework, Studierstube 4.0, was proposed in the Studierstube Augmented Reality Project [11][12]. This framework aims at creating a complete solution for AR on PDAs. Studierstube 4.0 is a cross-platform software framework that includes various components, such as graphics API, scene graph libraries, tracking middleware, multimedia playback tools, and so on. Wagner developed an application named "Signpost" based on this framework [13]. The application guides the user through a building by showing a map and an arrow, which indicates the direction. The framework mentioned in [13] supports not only a fully self-contained AR/MR system, but also a server-assisted AR/MR system. However, this framework cannot be applied in extremely limited devices such as mobile phones.

3 Scalable Architecture

3.1 Design Concept

There are varieties of mobile devices, ranging from mobile phones to notebook PCs, and there are significant performance gaps between them. For example, some contemporary notebook PCs have almost the same capability as desktop PCs. However, mobile phones used in Japan do not have enough memory and have many security limitations [14][15]. Accordingly, it is difficult to develop and run complex applications on such devices. It is assumed that these performance gaps will continue to exist in the future.

In order to absorb such performance gaps within the system, we decided to adopt a client-server model, allowing the server to compensate for low-performance clients.

In general, a mixed reality system is comprised of the following six modules:

(1) Camera Module
The Camera Module captures raw images (real world scenes), on which virtual objects are superimposed, augmenting the user's perception.

(2) User-Interface Module
This module is an interface for users. It can present users with raw images superimposed with MR information, and receives explicit input from users.

(3) Tracking Module
The Tracking Module is one of the most important components of an MR system. This module detects the position and orientation of the client.

(4) Rendering Module
The Rendering Module draws virtual objects with the correct distance, position, and orientation, based on the client's position and orientation.

(5) Content-Management Module
This module manages the content database in the database system. The content database includes 3D graphic models, text annotations, and 2D images.

(6) Application Module
The Application Module handles application-specific logic. It performs processes such as switching current contents based on the user's context, user position, and/or input from the user.

In addition to these modules, one additional module completes the client server/system.

(7) Client Management Module
This module manages client information and user context.

3.2 System Architecture

In order to accommodate the performance gaps between various types of mobile devices, we distributed modules, one through seven, to the server and clients. Fig. 1 shows the system architecture. In this architecture, mobile devices are classified into three groups: light-load clients (LLCs), middle-load clients (MLCs), and heavy-load clients (HLCs).

LLCs are also called thin clients. For LLCs, the system places only the camera and the user-interface modules in client devices. All other functions are performed by modules on the server. Since the vision-based tracking method is used for LLCs, an LLC client has to communicate with the server each time it displays an image. Therefore, it is not suitable for continuous presentation of the MR images. However, since the client has only two modules, it is fairly easy to implement. For example, mobile phones of NTT DoCoMo, the most widely used phones in Japan, can run programs in an environment called "the i-αppli Runtime Environment." In this environment, only one i-αppli can run at a time because of security limitations [16]. In other words, when the camera function is used, i-αppli execution is interrupted and the actual photographing depends completely on the user—the application program cannot operate the shutter. This means that users have to press the shutter of the built-in camera to capture every scene. The process flow for LLCs stated before is assumed to be used in such limited environment.

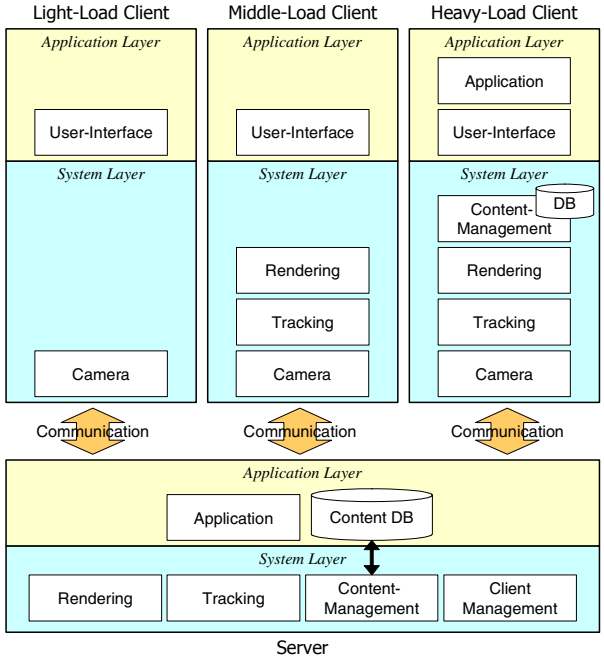

Fig. 1. The system consists of a server (the lower box) and clients (the upper boxes). Both the server and clients have two layers: *System* and *Application Layer*.

MLCs are clients that have tracking and rendering modules in addition to the camera and user-interface modules. MLCs detect and send position and orientation information to the server and receive local content related to the surrounding real world. It is possible to continuously present MR images by drawing MR information based on contents received from the server. However, since the application module on the server selects the contents to be sent to the client, the clients can only render images based on the contents sent by the server.

HLCs are self-contained clients that have all modules required to realize mixed reality themselves. Because of this, HLCs must be relatively high-performance devices. If the client is an HLC, a user can experience high-quality mixed reality in real time by wearing a head mounted display (HMD). HLCs store and manage perfectly duplicated copies of the server database. The HLC synchronizes the content database with the server, so that the database is always up to date. Since the database itself is on the HLC, the application module in the client can select the contents to be presented to the user.

However, the server must have all five modules, other than the camera and user-interface module, so that it can compensate for shortcomings of any client. The server responds to requests from clients by providing the required functionalities, based on the client type.

Both the server and clients have two layers: system and application layer. In any client, the camera and tracking modules are placed in the system layer, as they are shared by all applications. However, the application and user-interface modules reside in the application layer, because different applications use different modules. Reference implementations of modules placed in the application layer can be obtained. However, application-layer modules must be developed and installed by the developers.

4 Content-Description Language

In this architecture, each client must communicate with the server to display MR information. The architecture proposed in this paper is based on the client-server model; communication between client and server is through the HTTP protocol. Information exchanged between clients and the server is roughly divided into two groups: requests from clients and responses from the server.

The information exchanged between the server and the clients varies according to the client type, as shown in Fig. 2. SKiT-XML serves as the content-description language used in such communication. SKiT-XML is one instance of XML [17].

The most important point in this architecture concerns sharing identical content among various clients or terminals. Note that elements such as 3D graphic models and text annotations contained in the contents, are defined as virtual objects. The clients have to display identical motion of virtual objects to the users even when the objects change over time.

Two kinds of motion of virtual objects are to be considered: shape changes of an object, and movement of an object within the environment. Note that it is impractical to describe all object position and orientation parameters for each frame. Therefore, we describe parameters of each object at a specified timing, assuming that each client's system time is synchronized with the server. We call the parameter list the Sequence, and the sequence is used to represent object motions. In the sequence, the states of objects between one parameter set and the next are obtained by linear interpolation. The same method is used for animation description in scalable vector graphics (SVG) [18].

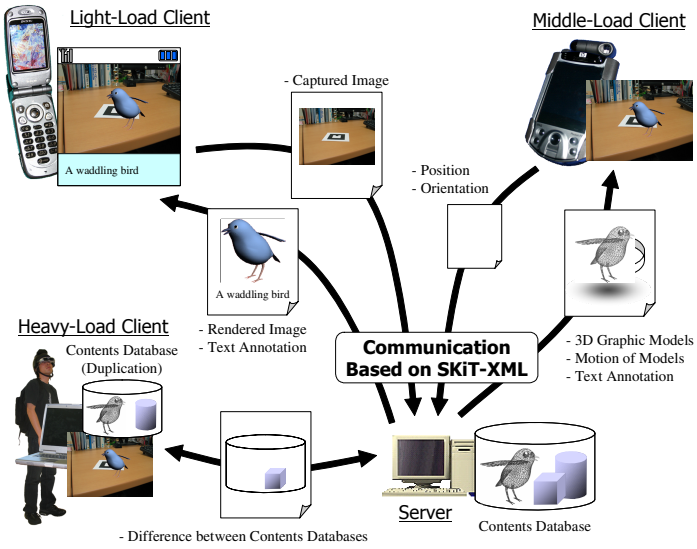

Fig. 2. Information exchanged between the server and the clients differs according to the client type. The LLC sends a captured image to the server and receives a rendered image and text annotations. The MLC sends its position and orientation and receives 3D graphics models, motion of the models, and text annotations. The HLC synchronizes the content database with the server.

All sequences of the virtual objects are stored in the server's content database. The application module changes the records in the content database according to the user's context, or while interacting with the user.

When we describe contents in the way shown above, the following data exchange is required, depending on the client type.

- LLC sends a camera image to the server in order to detect its position and orientation. The server renders the contents at that time, based on the corresponding sequence, and sends the rendered image to the client.
- MLC sends its position and orientation to the server. The server returns a sequence of virtual objects surrounding the client.
- HLC checks periodically for any changes in the server's content database. The server returns SQL statements used to change the contents in the database, to the client.

5 Experimental Results

We examined the effectiveness of our architecture by installing the system on various terminals. Table 1 shows specifications of clients and the server used in these experiments.

Table 1. This table shows the hardware specifications of the clients and the server used in these experiments

Hardware	Type	Resolution	Graphics Chip	Camera	Network
Dell Precision 450	Server	---	nVIDIA Quadro FX500	---	---
NTTDoCoMo SH901iC	LLC	QVGA	---	Built-in	Packet Network
Sharp Zaurus SL-6000W	LLC, MLC	VGA	---	CE-AG06	IEEE 802.11b
HP iPAQ h5550	MLC	QVGA	---	FlyCAM-CF1.3	IEEE 802.11b
Xybernaut MA-V	MLC, HLC	SVGA	ATI RAGE Mobility-M	CMS-V13	IEEE 802.11g
Sony VAIO Type U	MLC, HLC	SVGA	Intel 855GM	CMS-V13	IEEE 802.11g
Dell Precision M60	MLC, HLC	SVGA	nVIDIA Quadro FX Go1000	CMS-V13	IEEE 802.11g

5.1 Hidden Cable Viewer

An application called “Hidden Cable Viewer” was developed and installed on the server and the clients. The application visualizes cables hidden beneath the panels of a free-access floor, so that re-arrangement of such cables is possible.

The objective area consists of 4×6 panels, each 50 cm square. Beneath the panels are power and LAN cables. The ARToolKit [19] is used as a tracking mechanism, and 12 cm square markers are respectively placed on each tile. The ARToolKit is used only to ensure solid and stable positioning and we do not recommend using such markers.

Fig. 3 shows an example application. LLCs and MLCs of multiple types offered the expected performance.

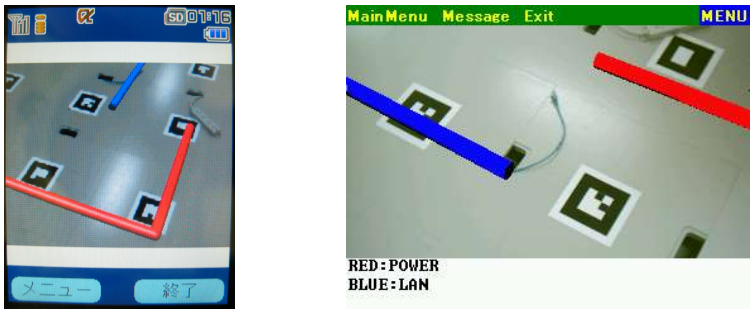

Fig. 3. The left figure is a screenshot of SH901iC, which is implemented as an LLC. The right figure is a screenshot of iPAQ h5550, which is implemented as an MLC.

When the shutter is pressed on an LLC, it sends the image taken by the camera to the server. The server computes client’s position and generates MR information, which is then returned to the client. The client, then, displays the returned MR image. It takes approximately 15 seconds for the SH901iC mobile phone and 3 seconds for SL-6000W PDA to display an MR image to the user from the moment the shutter is pressed. Note that most of this time is spent in communication.

Since an MLC can detect position and orientation, it does not need to send an image to the server. It sends only the required position and orientation information. Similarly, the server can return only wiring information to the client. This reduces the communication load, allowing a movie-like presentation. The user can change the

direction of the PDA and can see MR images for that direction with only a small delay.

5.2 Navigation System

Another application we used to examine our architecture is an MR-based navigation system. In this system, a user follows a path to a goal by following a flapping virtual bird. The system provides navigation at cross points, switching modes to show an arrow indicating the direction to go.

Fig. 4 shows sample images of this application. A bird at the center of the screen flies by changing position and orientation. This motion is described by the sequence explained in Section 3. A Precision M60 is used as a client. Its screen shows images from a USB CMOS camera with the superimposed virtual bird. This is an HLC, and the system ran at approximately 20 fps (frames per second). ARToolKit provided solid and stable position information.

Fig. 4. The left figure is a screenshot of a Precision M60, which is implemented as an HLC. A fluttering bird guides the user to his/her destination. The right figure is a screenshot of VAIO Type U, which is implemented as an MLC. A user heading and an annotated image of a room are displayed.

5.3 Campus Guide System

We also tested our architecture by implementing the Campus Guide System. Fig. 5 shows examples. Since position and orientation detection by the ARToolKit cannot be applied outdoors, we adopted a gyro-aided, dead-reckoning module (Gyro-DRM-III) and an inertial-based 3-DOF sensor (InertiaCube3), to build the tracking module. Since Gyro-DRM-III only detects positions within one foot, we obtained positions between each step by linear interpolation of the detected positions. The tracking module integrates the output of Gyro-DRM-III and InertiaCube3, which provides full 360-degree tracking in all axes. The error of Gyro-DRM-III is within 5 m for 100 m, and is not accurate enough. However, it is possible to use it in applications such as in outdoor guide system.

The system frame rate is nearly 25 fps. The frame rate is mainly dependent on the performance of the USB camera. When a USB CCD camera is used (Buffalo BWC-C35H01/SV), the frame rate is 28 fps. However, the USB CCD camera is slightly unstable.

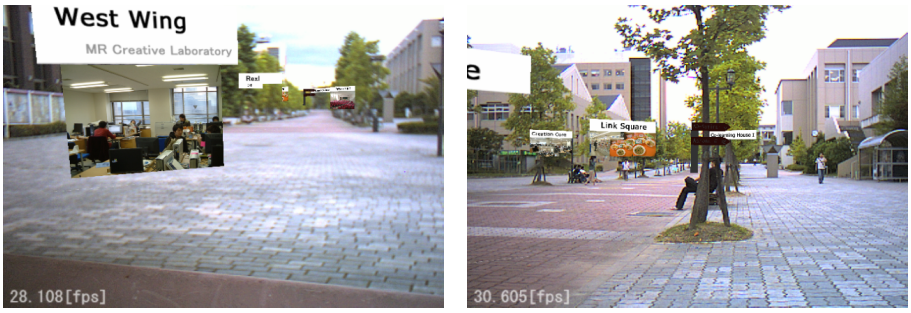

Fig. 5. These figures are screenshots of a Precision M60, which is implemented as an HLC. Annotations are displayed in front of the corresponding buildings.

6 Conclusion

In this paper, we proposed a scalable architecture that can accommodate various types of mobile devices to realize AR/MR functionality, and described the results of its implementation in actual applications. These results show that the significance of our scalable architecture will increase as mobile devices evolve.

Though it is out of the scope of our concept, one problem to be resolved is building a stable tracking method for outdoor use. We used Gyro-DRM-III as a position detection device, however, the results indicated that it did not have the required accuracy. Many researchers are working on sensing devices for outdoor use. We believe that our application- and sensor-independent architecture can play a great role in outdoor use also when this research is expanded.

Acknowledgement

This research was partially supported by a Grant-in-Aid for Scientific Research (B), of the Ministry of Education, Culture, Sports, Science and Technology, No.17200039, 2005. The authors would like to thank Dr. Ryuhei Tenmoku for fruitful discussions.

References

[1] Handheld Augmented Reality, Proc. of 4th IEEE and ACM Int. Symp. on Mixed and Augmented Reality, pp.xix–xxi, 2005.

[2] J. Fruend, C. Geiger, M. Grafe, and B. Kleinjohann: The Augmented Reality Personal Digital Assistant, Proc. of 2nd Int. Symp. on Mixed Reality, pp.145–146, 2001.

[3] D.Wagner and D.Schmalstieg: First Steps towards Handheld Augmented Reality, Proc. of 7th IEEE Int. Symp. on Wearable Computers, pp.127–135, 2003.

[4] W. Pasman and C. Woodward: Implementation of an Augmented Reality System on a PDA, Proc. of 2nd IEEE and ACM Int. Symp. on Mixed and Augmented Reality, pp.276–277, 2003.

[5] M. Möhring, C. Lessig, and O. Bimber: Video See-through AR on Consumer Cell-phones, Proc. of 3rd IEEE and ACM Int. Symp. on Mixed and Augmented Reality, pp.252–253, 2004.

[6] F. Shibata, A. Kimura, T. Hashimoto, K. Furuno, T. Hiraoka, and H. Tamura: Design and Implementation of General Framework of Mixed Reality Systems Applicable to Various Mobile and Wearable Platforms, Transactions of the Virtual Reality Society of Japan, Vol.10, No.3, pp.323–332, 2005(in Japanese).
[7] B. MacIntyre and S. Feiner: Language-level Support for Exploratory Programming of Distributed Virtual Environments, Proc. of 9th ACM Symp. on User Interface Software and Technology, pp.83–94, 1996.
[8] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway: Exploring Mars: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System, Computers and Graphics, Vol.23, No.6, pp.779–785, 1999.
[9] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riß, C. Sandor, and M. Wagner: Design of a Component-based Augmented Reality Framework, Proc. of 2nd IEEE and ACM Int. Symp. on Augmented Reality (ISAR '01), pp.45–54, 2001.
[10] M. Miyamae, T. Terada, Y. Kishino, S. Nishio, and M. Tsukamoto: An Event-driven Navigation Platform for Wearable Computing Environments, Proc. of 9th IEEE Int. Symp. on Wearable Computers (ISWC 2005), pp.100–107, 2005.
[11] Studierstube Augmented Reality Project Homepage, http://studierstube.icg.tu-graz.ac.at/
[12] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavari, L. M. Encarnação, M. Gervautz, and W. Purgathofer: The Studierstube Augmented Reality Project, PRESENCE - Teleoperators and Virtual Environments, Vol.11, No.1, pp.33–54, 2002.
[13] D. Wagner and D. Schmalstieg: First Steps towards Handheld Augmented Reality, Proc.of 7th Int. Symp. on Wearable Computers (ISWC 2003), pp.127–135, 2003.
[14] NTT DoCoMo, Let's make i-mode contents: i-αppli, http://www.nttdocomo.co.jp/english/p_s/i/make/java/index.html
[15] KDDI au, EZfactory, http://www.au.kddi.com/ezfactory/index.html (in Japanese)
[16] NTT DoCoMo, i-αppli Content Developer's Guide for DoJa-3.0, 2003.
[17] Extensible Markup Language (XML) 1.0 (Third Edition), http://www.w3.org/TR/REC-xml/
[18] Scalable Vector Graphics (SVG) 1.1 Specification, http://www.w3.org/TR/SVG11/
[19] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana: Virtual Object Manipulation on a Table-top AR Environment, Proc. of Int. Symp. on Augmented Reality, pp.111–119, 2000.