ID: 68

Proceedings of the International Workshop
on Modern Science and Technology
Kitami, Japan/September 2010

Ubiquitous User Interfaces Framework Using Augmented Reality as a Platform

Svetoslav Dankov, Rafal Rzepka, and Kenji Araki

Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan
{dankov, kabura, araki}@media.eng.hokudai.ac.jp

Abstract: Augmented Reality applications are becoming more popular with the continued miniaturization of technology. With the increasing use of smart phones, which often provide increased processing power, enhanced and open software platforms, Augmented Reality has become instrumental in the way we perceive our surroundings and the information that it carries. It is now possible to implement an Augmented Reality system without carrying bulky and expensive equipment. Currently, there are many systems that implement some form of Augmented Reality to provide a specialized interaction to users. However, those systems usually employ expensive, immobile components with highly specialized interfaces. In this paper we present a novel approach for building interactive interfaces using Augmented Reality. We present a software framework for ubiquitous Augmented Reality enhancement for human-computer interaction. Our framework improves on four areas in Augmented Reality development that we currently see lacking.

Keywords: augmented reality, human-computer interaction, ubiquitous user interfaces

1 Introduction

Augmented Reality (AR) is a fairly young area of research which is currently expanding in many of the already existing fields of Human-Computer Interaction, Computer Interfaces, etc. In our research we implement an Augmented Reality system which will serve as an extension to existing computer interfaces, provide enhanced user-experience, and define virtual objects and their actions in an ubiquitous way.

To implement such a system, however, we must first address three major areas where we think Augmented Reality user interfaces can be improved.

Firstly, Augmented Reality objects are hard coded into applications, which makes them highly specialized and not ubiquitous. Secondly, there is no standard for defining Augmented Reality interfaces and how they react to human interaction. Finally, there is no implementation of natural interaction with such interfaces and objects.

We begin by presenting a summary of the related research, connecting our approach with previous work.

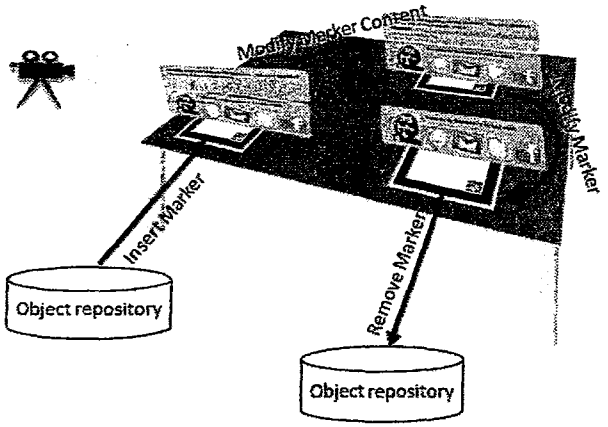

Figure 1: Example usage scenarios

Next we present use cases which describe specific functionalities that our framework enables and describe the purpose of the framework. This is followed by a description of the software tools and libraries we take advantage of while implementing the framework. We conclude by giving short review of the steps ahead.

2 Related Work

The existing research into Augmented Reality and Human Computer Interaction that is relevant to this study can be roughly divided in three areas: finger and hand-based interfaces, paper-based interfaces, and Augmented Reality applications. We will describe each one briefly.

2.1 Hand and Finger-based Interfaces

The technologies for hand and finger-based interfaces can be roughly split in two categories - sensing-based and computer-vision based. Sensing-based systems like [16] are very robust but are often limited to detecting only "touch" behavior, not able to recognize hands or other physical object that come into view. Computer-vision based systems like [4], [5], [13] are often limited by the lighting conditions and may not respond well to sudden changes in the field of view. However, systems like [8], [9], [7] have proven to be robust and accurate enough. We are using a computer-vision based system since wearing special hardware to enable "touch" capability reduces the mobility of the system.

2.2 Paper-based Interfaces

Most of the existing paper-based interfaces fall into three categories - using paper alongside digitizing tablets like [12], using digital paper technologies like [3], and using paper tagged with markers (barcodes, fiducial markers, etc.) like [13]. In our system we will be using only 2D paper tags (AR markers) for 3D positioning of the visual objects, unlike [9] where the hand position and direction is used to determine the position of the virtual object. The interaction between the user and the interface will be entirely virtual or conveyed through AR marker motion.

2.3 Augmented Reality Applications

There have been many Augmented Reality applications, using either multiple-camera hand and object tracking or a single camera (like a webcam). Those applications vary in both their mobility and complexity. Our project was inspired for the most part by the Sixth Sense project developed by Pranav Mistry in the MIT Media Lab [14]. As is the purpose of [14], we strive to provide mobility, affordability and ubiquity to Augmented Reality applications.

There are two major differences between the paradigm employed by [14] and our framework. We let the user utilize any device to view the AR environment (mobile phone, webcam plus desktop, AR goggles, etc.) where [14] projects the AR environment over objects themselves. We also use 2D paper AR markers to determine correct 3D coordinates and scale for object placement.

2.4 Augmented Reality Frameworks and Authoring Tools

With the popularization of software libraries like ARToolKit [6] there has been a lot of development to bring AR authoring tools in the hands of researchers and developers. However, frameworks like DWARF [1] and osgART [11] are quite complex and require an expert programmer. Our framework on the other hand gives developers with enough programming experience in ActionScript3 the ability to construct and distribute user interfaces, interactive objects, etc. with ease.

3 Usage Scenarios

The usage-scenarios described below serve to describe specific features of our framework that are not available in current AR systems. Via those usage scenarios we want to illustrate the particular unique functionalities that our framework provides.

3.1 Universal Marker Registry

In our system we plan to use 2D paper markers for virtual object placement. Currently, there are multiple ways to create such a marker with the only restriction that the pattern not be too complex. This improves the marker recognition which in turn allows for a scalable AR experience. The relation between the virtual object and the marker pattern is embedded in the software.

We plan to implement a universal AR marker registry so

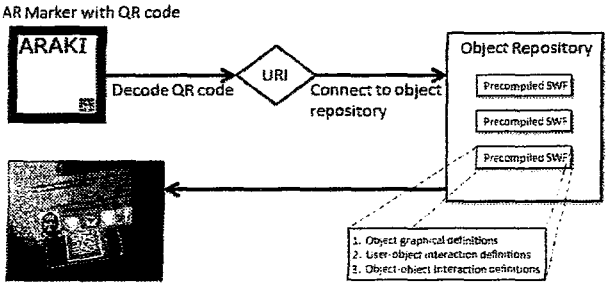

Figure 2: Basic framework model

that information about the object is stored globally. Using this registry, users will be able to point their AR device to any AR marker and display its contents, regardless of whether they have seen the marker before or not. The system will recognize the URL encoded in the marker, download and display the virtual object. We are planning to include a QR code within the AR marker pattern and extract the URL from it.

This will allow for more ubiquitous AR applications. The user will no longer be restricted to using markers specifically designed for his AR system. We hope the implementation of such registry will attract interest from the Augmented Reality community and help construct a large ecology of AR objects. It will also enable developers to construct their own AR Marker ecologies, independent of the main system.

3.2 AR Object-object and Object-user Interaction Definition

The next step in our system is defining virtual object actions as part of their registry information described in the previous section. This way the AR system will know both what objects to display, as well as how those objects are supposed to interact with the user and other objects.

For the most part our virtual objects are user-interfaces. As such, their actions are defined either as user initiated or object initiated. We would like to implement both ubiquitous object-object and user-object interaction. Object-object interactions will allow us to define how virtual objects behave when in proximity of one another. A simple example would be two virtual objects positioned by 2D markers on the field of view, both objects representing a single Skype chat window with different users. If both markers are positioned close to one another the resulting action will be to open a single Skype window making a conference call to both users. User-object interactions will be described as the services the virtual object can perform upon user actions. For example, a virtual object displaying information on a person (a virtual business card) can provide information upon request, provide an email interface, a Facebook or Twitter message interface, current location, etc.

Building such a repository of objects will provide both a functional and a graphical description of the AR objects in an Augmented Reality environment which in turn will make AR applications more ubiquitous.

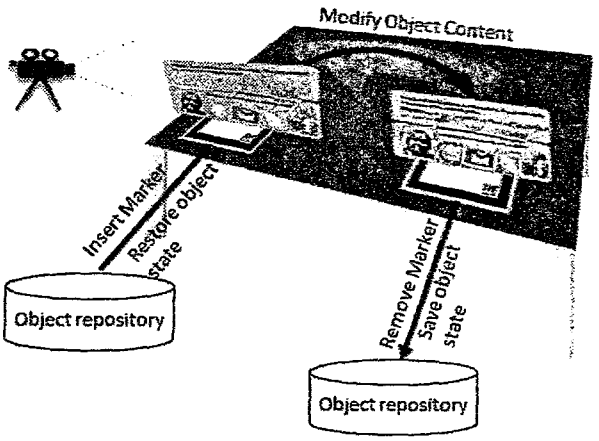

Figure 3: Object Persistence

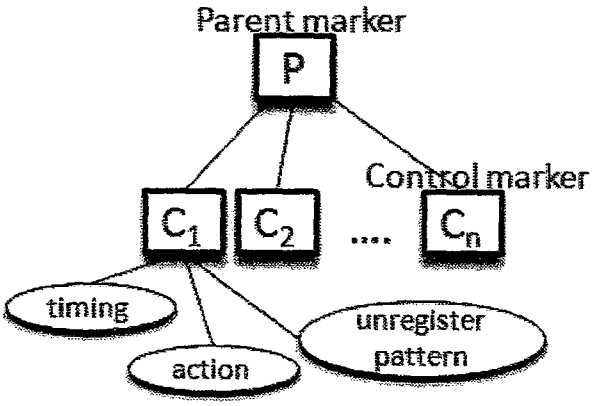

Figure 4: AR Object Control Scheme

4 Framework purpose

In the following list we formally introduce the problems and issues current AR systems suffer from and briefly outline how our framework tackles them:

- **Ubiquity of Object presence**
  Objects associated with one AR marker must remain the same independent of system used to view the AR environment. Objects are first registered via a QR code which encodes the URI of the repository from which the object graphics model and other data is to be drawn. This gives each AR marker (independent of its graphical representation) an unique identifier. This unique identifier allows us to implement the next three objectives.

- **Object Persistence**
  Objects must carry associated data and object states across different AR environment viewers. Object data and object states are stored in the database defined via the AR marker's URI.

- **Ubiquity of Object Interactivity**
  Objects must behave the same way independent of AR environment viewer. Object's interaction definitions are stored in the database defined via the AR marker's URI.

- **Definition of Object Interaction Models**
  Interaction models with AR systems have so far been system/application dependent. Each system defines for itself how users interact with the AR objects and the interaction model cannot be extended or redefined.
  Our framework allows developers to define how a specific AR Object will interact with the users and with other AR Objects introduced to the scene. They do so by assigning behaviors to extra control markers associated with the AR Object via the AR Object's URI. Figure X shows an example of AR Objects and controls.

Figure 1 shows an example usage scenario encompassing all four focus areas described above. As the AR marker is inserted into the scene, the marker's graphical and functional representation is obtained from the object repository on the server encoded in the QR code. In this case the AR marker's object is a simple business card showing a photograph, 4 buttons which when activated would present a different interface depending on the button, and a message board. The user can modify the content - in this case leave a new message on the message board or modify the object the marker represents - in this case by closing the message board section. Once the marker is removed from the scene it's object properties are saved and the next time the marker is introduced it will remember them.

5 System design, software components and implementation

5.1 Software components

Our system is based on several existing technologies that allow us to perform AR overlay, QR decoding, marker recognition, tracking and handling and draw our interfaces programatically. In this section we will look at each one in more detail.

- **AR overlay:** The original AR toolkit was first developed by Dr. Hirokazu Kato from the University of Washington [6] and is currently supported by the Human Interface Technology Lab at the University of Canterbury in New Zealand [2]. As we are building our framework in Adobe ActionScript programming language, we are using a language port of the ARToolKit to AS3 provided by Saqoosha [17], Nyatla [15] and Sparklib [10] named FLARToolKit.

- **QR decoding:** For decoding QR codes in ActionScript we use the QR library provided by Sparklib [10].

- **Marker handling:** To manage marker registration efficiently for multiple markers and predict marker motion we use the FLARManager 0.7 toolkit which is provided by Eric Socolofsky [18].

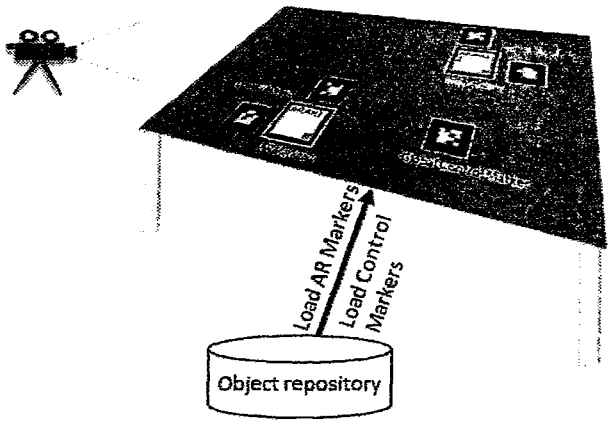

Figure 5: User-object interaction

- **Interface Design:** To design, draw and define our interfaces we use PaperVision3D library provided by [19]. PaperVision3D is a set of libraries that give ActionScript developers a 3D engine for Flash.

All of the above mentioned libraries are distributed under licenses allowing developers to use them free of charge for non-commercial purposes. Our framework is built using Flash Builder 4 and ActionScript 3.5 SDK. Developers can produce their modules using any ActionScript compiler as long as they run the same SDK and use the same versions for PaperVision3D, FLARToolKit, and FLARManager.

5.2 System model

Figure 2 describes the basic model of our system. Here we will look at each component.

- **AR Marker with QR:** We designed our AR markers to include QR codes encoding the Unique Resource Identifiers for the object that the AR marker identifies. This allows the developer to define his own AR marker patterns and objects independent of the viewer. It also allows the AR environment viewer to recognize AR markers without the need to include the patterns in the program. The QR code can be placed either inside of the AR marker as part of the pattern or on the back of the AR marker. Note that if the QR code becomes a part of the AR marker's pattern it must do so in an asymmetrical fashion, since AR marker patterns must be asymmetrical to enable correct marker detection.

- **Database:** The database component of the system implements a simple MySQL scheme with database entries containing developer information and pointing to a local directory for specific marker id. The physical file is a precompiled Adobe SWF file that contains the AR Object's graphical and interaction definitions.

With object persistence we ensure that an AR object will retain its information and state in case it is removed from the AR environment. Figure 3 provides an example of object persistence. If an AR Marker is introduced and the user makes a modification to the state of the object it represents, the system will relate that change to the database. The next time that marker is introduced to the scene, the system will display it's previously modified state.

5.3 System interaction

In order to continue to the next section we must define the control scheme for AR Objects. Figure 4 shows how we implement interaction with our objects. In the database, each parent AR Marker has associated with it a set of control markers that define a single action. Those control markers are our equivalents of a "button". Each control marker is defined by a "timing" parameter and an "action" parameter. To initiate the control one must simply introduce the control marker into the scene. The system detects that the control is activated if it is not registered longer than the "timing" parameter specifies, and performs the action based on the "action" parameter. To unregister the control marker from the scene one must introduce an "unregister" pattern as defined per marker (that pattern can simply be printed on the back of the control pattern).
We can now describe how our system's user-object and object-object interaction paradigms.

5.3.1 User-object Interaction

Figure 5 describes how the users will interact with the AR Objects. For each marker ecology (defined by the database the system is connecting to) there will be a single marker pattern for a global control marker. The purpose of the global control marker is to select between the active AR Objects on the scene. The detection technique the system uses is the same described in section **5.3**. Once an AR Object is active the user can move it around and perform actions as defined by its control markers, as described in section **5.3**.

5.3.2 Object-Object Interaction

The last type of interaction we define in our system is object-object interaction. Figure 6 gives an example of one AR object being aware of another. In this paradigm the developers of the AR Objects are allowed to define an "collision area" defined by an offset to the area of the AR marker. Each marker can either be on the receiving end (marker stationary) or the sending end (marker moving). Each AR object is associated with both a receiving action and a sending action. One simple example of such object-object interaction is when both AR objects are business cards. When business card A detects the proximity of business card B, the information on B will be attached to A's contact list.

6 Conclusions

We are currently deploying a prototype version of our framework and doing preliminary testing of its usability,

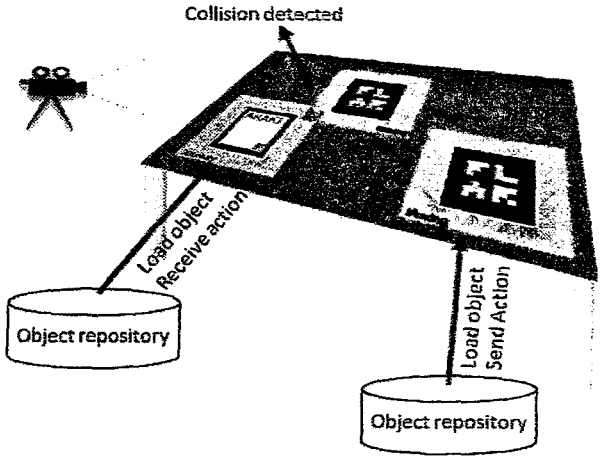

Figure 6: Object-object interaction

robustness and feasibility, both from a developer and user standpoint. Our system is currently suited only for desktop/webcam configuration although we plan to change that, either by implementing an AR viewer on mobile devices or by introducing Virtual Reality (VR) goggles.

7 Future Work

After carefully evaluating the implementation of our system, we plan to implement an AR viewer for mobile devices using an HTC developer device running Android 2.1 OS. We are always looking for better interaction techniques and we plan to use colored fingertip markers to implement additional user-object interaction. In order to improve the overall usability of the system, we would like to implement marker-less fingertip detection in the future.

Another area of interest involves the possible security implications such a framework would entail (an object implementing a business card should be visible to anyone, while a private message window should be visible only to users authorized to view it). We plan to look into implementing user authentication and different user ownership for AR objects.

References

[1] 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. *Augmented Reality, International Symposium on*, 0:45, 2001.

[2] M. Billinghurst. Artoolkit. `http://www.hitlabnz.org`, June 2009.

[3] P. Brandl, M. Haller, J. Oberngruberand, and C. Schafleitner. Bridging the gap between real printouts and digital whiteboard. *AVI'08*, 2008. In Proceedings of the conference on Advanced Visual Interfaces.

[4] S. Do-Lenh, F. Kaplan, A. Sharma, and P. Dillenbourg. Multi-finger interactions with papers on augmented tabletops. *TEI2009*, pages 16–18, 2009. The 3rd International Conference on Tangible and Embedded Interaction, Cambridge, UK.

[5] D. Holman, R. Vertegaal, M. Altosaar, N. Troje, and D. Johns. Paper windows: interaction techniques for digital paper. *CHI*, pages 591–599, 2005. Proceedings of CHI'05.

[6] ARToolworks Inc. Artoolkit. `http://www.hitl.washington.edu/artoolkit/`, June 2009.

[7] C. Keskin, A. Erkan, and L. Akarun. Real time hand tracking and 3d gesture recognition for interactive interfaces using hmm. *ICANN/ICONIP*, June 2003. Istanbul.

[8] B. Lee and J. Chun. Manipulation of virtual objects in markerless ar system by fingertip tracking and hand gesture recognition. *ICCIT'09*, 2009. Seoul.

[9] T. Lee and T. Hllerer. Handy ar: Markerless inspection of augmented reality objects using fingertip tracking. *ISWC*, October 2007. In Proceedings for IEEE International Symposium on Wearable Computers, Boston, MA.

[10] Spark Project Actionscript Class Library. `http://www.libspark.org/`, June 2009.

[11] J. Looser, R. Grasset, H. Seichter, and M. Billinghurst. Osgart - a pragmatic approach to mr. *ISMAR'06*, 2006.

[12] W. Mackay, G. Pothier, C. Letondal, K. Boegh, and H. Erik Sorensen. The missing link: augmenting biology laboratory notebooks. *UIST'02*, pages 41–50, 2002. In ACM Symposium on User Interface Software and Technology.

[13] C. McDonald, G. Roth, and S. Marsh. Red-handed: collaborative gesture interaction with a projection table. *FG2004*, pages 773–778, 2004. International Conference on Automatic Face and Gesture Recognition.

[14] P. Mistry, P. Maes, and L. Chang. Wuw - wear ur world - a wearable gestural interface. *CHI'09*, 2009. In the CHI '09 extended abstracts on Human factors in computing systems, Boston, MA.

[15] Nyatla. Nyartoolkit for as3. `http://d.hatena.ne.jp/nyatla/`, June 2009.

[16] J. Rekimoto. Smartskin: An infrastructure for freehand manipulation on interactive surfaces. *SIGCHI*, pages 113–120, 2002. Conference on Human Factors in Computing Systems.

[17] Saqoosha. Flartoolkit. `http://saqoosha.net/`, June 2009.

[18] E. Socolofsky. Flarmanager. `http://words.transmote.com/wp/flarmanager/`, December 2009.

[19] C. Ulloa, J. Grden, R. Hauwert, T. Knip, and Andy Zupko. Papervision3d: 3d engine environment for adobe flash. `http://www.papervision3d.org/`, June 2009.