Developing Interactive Augmented Reality Modelling Applications

Wayne Piekarski and Bruce H. Thomas

Abstract— This paper presents our position on the development of software for interactive AR applications, particularly those deployed in mobile outdoor environments. We describe previous work that has been performed in the field and how the main focus has been on low level problems such as hardware abstraction. To develop complex future AR applications, higher level user interface abstractions and supporting toolkits will be required. Working in outdoor AR environments with mobile standalone systems also introduces new limitations not previously encountered indoors, further complicating the problem. Furthermore, this software will have to be designed with the particular environments and hardware in mind to ensure the user interface is optimal. Another problem currently unsolved is the development of content for AR systems. Rather than using traditional 2D desktop applications, the development and consumption of AR information should both be possible on the mobile system. This allows the instant verification of 3D models in AR, which is a core part of the overall design process but not possible on a traditional desktop system.

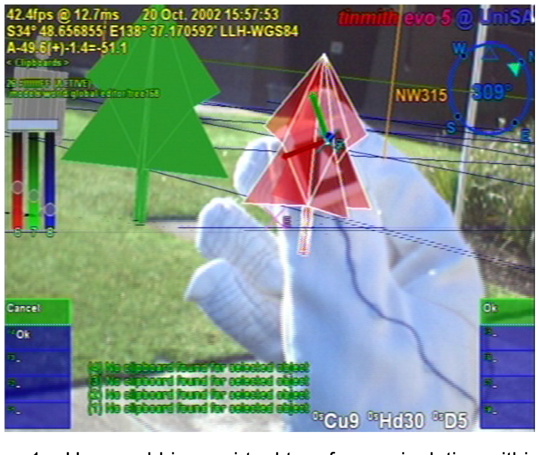
Index Terms— I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Virtual Reality; J.9.e [Mobile Applications]: Wearable computers and body area networks.

1 Introduction

We have been developing prototype augmented reality systems for use in outdoor environments for a number of years. As part of developing these prototype systems, we have performed research into the development of appropriate software architectures that simplify the development of applications. Our current software architecture is named Tinmith-evo5 [22], and combines a variety of novel techniques for developing applications that use less powerful wearable hardware, mobile 3D graphics, and high level user interfaces. We have used this software architecture to develop the Tinmith-Metro application [19] [21], which supports interactive 3D modelling in outdoor environments using the hands and the body to control the system. Figure 1 shows an example of this application in use with a landscape gardening example.

The development of software to support AR environments is much more challenging compared to other environments such as traditional 2D desktops. In 2D environments, there has been a lot of existing research and implementations have converged on a generally agreed best practice. In contrast, 3D environments such as AR and VR suffer from many different types of applications, differing opinion on the best user interfaces to deploy, and non-standardised and changing hardware. Furthermore, there are only a small number of software toolkits available to simplify applications development, so application developers must focus on reimplementing the wheel each time for their particular requirements. As discussed by Shaw et al [28], the development of higher level software toolkits is not possible until there are a stable set of low level toolkits to support them. For AR environments to become mainstream and easy to develop for (similar to the ubiquitous 2D desktop), high level toolkits with a rich feature set are required.

This paper describes our position on the development of AR applications based on previous work by ourselves and others. Initially this paper starts with a review of some existing work in the field, followed by our position on various areas of interest to AR researchers. We then discuss how AR applications may be described relative to existing VR based techniques, how outdoor AR has unique requirements compared to indoor AR, and how the only way to author content for AR is directly in the AR environment itself.



```
42.4fps @ 12.7ms    20 Oct, 2002 15:57:53
S34° 48.656855' E138° 37.170592' LLH-WGS84
A-49.6(+)-1.4=-51.1
```

```
[4] No clipboard found for selected object
[3] No clipboard found for selected object
[2] No clipboard found for selected object
[1] No clipboard found for selected object
```

Figure 1 – User grabbing a virtual tree for manipulation within the Tinmith-Metro mobile outdoor AR modelling application

• *Wayne Piekarski is with the Wearable Computer Lab at the University of South Australia, Mawson Lakes, Adelaide, SA, 5095, Australia. E-mail: wayne@cs.unisa.edu.au*
• *Bruce H. Thomas is with the Wearable Computer Lab at the University of South Australia, Mawson Lakes, Adelaide, SA, 5095, Australia. E-mail: thomas@cs.unisa.edu.au*

2 RELATED WORK

This section describes a number of different contributions that have been made in the development of toolkits to support AR and VR applications. As will be described later, we believe that AR and VR systems share very similar properties and so we describe them both together.

The first type of software toolkits for VR applications were hardware abstraction layers, providing generic interfaces to the non-standardised hardware available. Some early contributions were MR Toolkit [28] and VRPN [29], and some other recently developed software systems are DIVERSE [11], MAVERICK [10], OpenTracker [24], and VrJuggler [2]. Many of these systems are freely available and provide similar types of interfaces to trackers and renderers, indicating that these low level toolkits have converged toward some form of best practice. Other toolkits such as dVS [9] and World Toolkit [27] extend this model to provide scene graphs, event triggering, and distributed processing.

To support the definition of more complex software, there is a need to break down an application into objects or modules that communicate through some abstraction mechanism. The VRML 2.0 language [31] supports the definition of scene graphs, and includes fields and routes to allow 3D objects to be connected together. The Coterie system [14] implements object communications via a distributed shared memory, allowing applications to be spread across multiple systems. Coterie integrates a distributed scene graph with animation, an interpreted language, and tracking abstractions. The Studierstube system [26] is also used to develop distributed applications, but in contrast to Coterie it embeds the application into a distributed scene graph. Studierstube is based on a distributed version of Open Inventor, and to be distributed all components must be expressed using these interfaces.

There are a number of other systems for developing 3D applications. DIVE [7] supports distributed scene graphs with multicast to improve scalability. Avocado [30] provides features similar to VRML fields and routes with a scripting language. Lightning [3] implements data flow between objects to support distribution. VB2 [8] uses a constraint engine to implement relationships between objects. DWARF [1] uses a services based framework to connect components over a network. Although there has been a wide range of research in software engineering for AR and VR applications, the state of the art is still quite primitive compared to the maturity we have in 2D desktop applications.

3 AR AS AN EXTENSION OF VR

Virtual reality systems have traditionally been designed around a sensor model where tracking devices are read in, processed, and then used to draw output to a head mounted display. The simplest VR systems use an input device that tracks the head of the user, and this tracking data is used to generate a matching viewpoint for a head mounted display. When viewed in the context of Milgram and Koshino's reality-virtuality continuum [15], VR and AR are two locations along this spectrum. Therefore, AR systems can be modelled similar to VR systems, although there are some additional requirements. The first is that a view of the physical world must be provided for an augmented overlay. This overlay may be performed quite simply using video cameras and software overlay, or optical combination in the HMD. Secondly, virtual objects must be stored relative to physical world coordinates, and performing VR actions such as flying [25] or scaled world techniques [16] does not make sense when virtual objects must register with physical objects.

Our latest generation of augmented reality software is Tinmith-evo5 [22], which is based around processing objects that are connected together to flow data through the system, processing inputs to produce outputs. The sensor model in use is similar to the traditional VR model and is depicted in Figure 2, where sensors are processed using application code and run time configurations to render data to a HMD. Internally, the sensors are abstracted as objects and when new data arrives, these objects notify other listeners that updates are available. Processing objects connect to these data objects and receive notification messages, calculating a new result that is then made available to other processing objects. These objects are connected together into a directed graph and the final calculated values are used to render to the HMD. The objects are stored in an object repository based on a file system model, and a consistent methodology is used for the design of each set of objects.

The use of a sensor model processing values for rendering to a HMD is common for many AR and VR systems. The tracking of the head is a relatively simple operation, and there are no gestures or commands to interpret – the user simply walks around to interact with the computer. These systems are therefore quite simple to implement, since the software is basically a renderer controlled by a tracker. The most difficult problem currently facing both AR and VR systems is the user interface technology however, and this has not been thoroughly addressed yet. Without good user interfaces, we cannot develop applications more complex than the simple rendered AR demonstrations currently available.

On a 2D desktop, devices such as mice and keyboards are commonly used to interact with intricate graphical user interfaces, and these interfaces have been highly refined over many years. For 3D environments, there are currently a number of different input methodologies depending on the designer and the particular application in mind, and there is still disagreement on the most intuitive interaction

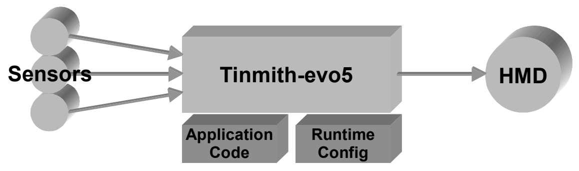


Figure 2 - Overall Tinmith-evo5 architecture with sensors processed using libraries and application components, and then rendered to the HMD

methods. For example, 3D systems use a wide range of input devices such as trackballs, gyro mice, keyboards, dials, speech, 3D wands, props, and tracked gloves. For user interfaces, researchers have used direct 3D manipulation, action at a distance techniques, the use of 2D interfaces in 3D environments, and command entry. In many cases, user interfaces are borrowed from 2D environments and used directly in 3D, with unintuitive controls for the user given the completely different environment. We do not believe that 2D user interfaces or input devices should be used in the development of AR applications since there is a mismatch in capabilities.

We see augmented reality as enhancing a user's perception of the environment, and believe the best way to currently achieve a sense of presence is through the use of HMDs. When wearing a HMD, we need input devices and user interfaces that are designed for this type of environment. In an immersive environment such as this, we think that using intuitive gestures of the hands and the body is the best way of interfacing to the user. While in VR techniques such as flying [25] and scaled world [16] are used, these are not useful for AR since it requires the user to break registration with the environment. Devices such as keyboards and 2D mice controlling a cursor are designed for the desktop and add levels of indirection that are unintuitive for a user. In the Tinmith-Metro application [19] [21], we use gloves that map hand motions and finger pinches to a pointing and command entry system designed specifically for the mobile environment. The gloves are currently a technological limitation, and in the future we would like to make it possible for a user to interact with the system without having to wear anything on the hands. We envisage the user controlling the wearable in the same way that two workers may discuss a task on a construction site. We have also been inspired by the Put-That-There system [4], where a user can work with shapes by pointing at them and speaking commands to perform. Working in a 3D environment is different from 2D interfaces, because objects can be rendered as they are normally shown in the real world and are not just abstract icons or 2D pictures on a monitor. With the future ability to render realistic 3D objects that merge seamlessly with the environment, having interfaces that match what users experience in daily life is definitely desirable. When designing these user interfaces, we feel that new input methodologies must be designed for 3D environments rather than just 'hacking' existing 2D devices into immersive applications. As an extension to the previous arguments, since 3D environments are inherently artificial it should also be possible to perform actions that are not possible in the real world. Some artificial methods developed for selecting 3D objects at a distance are Spot Lights [13], Apertures [6], and Image Plane Techniques [23]. One limitation of many existing demonstrations is that each technique is only demonstrated individually and there are no controls to switch between many functions. Many systems such as SmartScene [17] have intuitive interfaces for editing existing 3D worlds, but performing abstract creation concepts from scratch is not as simple or not possible. Therefore, an equally important consideration in user interfaces is command entry, where the user is free to activate appropriate techniques at any time in an easy and intuitive way. While our Tinmith-Metro menu system attempts to address the problem of building user interfaces that support many techniques with flexible command entry, it is far from perfect and is still in its infancy.

As stated earlier, an important problem for AR systems (and 3D environments such as VR) is the development of powerful user interface techniques. With the development of interfaces that are as refined as desktop computers, it will be possible to then build real applications with similar usefulness. Based on this argument, developing software architectures that provide hardware abstractions is not enough, and more higher level functionality is required. While low level software interfaces to the hardware, higher level software must interface with humans and user interfaces make this happen.

4 Mobile outdoor AR

We believe that the most powerful augmented reality applications are those where the user is able to freely roam about any environment experiencing overlaid virtual objects. While performing AR at a desk or physically tethered to a piece of infrastructure allows applications for surgery or maintenance, a much more general and useful system should not be limited to a small working area. We are currently exploring the development of autonomous mobile AR systems that are worn by the user, such as the Tinmith-Endeavour backpack depicted in Figure 3. This backpack is portable and may be used in both indoor and outdoor environments. When working outdoors, tracking technology such as GPS and hybrid inertial/magnetic sensors allow an almost unlimited tracking range in conditions where the sky is mostly visible. When indoors, we have used tracking technology such as fiducial markers [20] although there are other technologies available. Our research work is not focussed on the development of improved trackers but on what is currently possible.

Working outdoors imposes a number of problems not normally experienced when indoors, such as uncontrollable lighting, environmental noise for various sensing technologies, a lack of infrastructure, and the availability of less tracking hardware that can operate under these conditions.

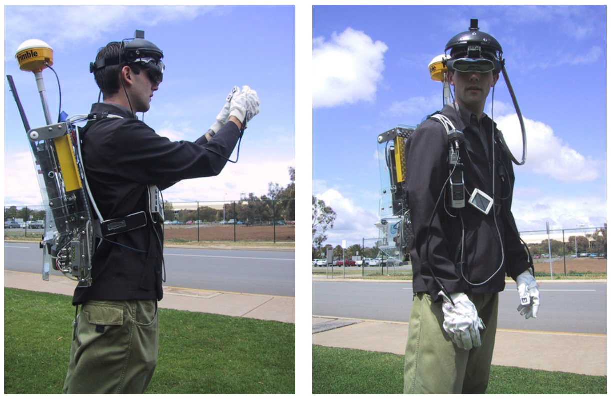


Figure 3 – Tinmith-Endeavour backpack with side and front views

These problems are supplemented by other problems associated with performing mobile computing and there are limitations on weight, size, and power consumption since a human must be able to carry the equipment. Rather than testing our software on simulators and planning for technology to improve in the future, we have designed our software so that it can be tested and demonstrated on currently available technology. When designing software architectures, it is important to keep the limitations of mobile outdoor hardware in mind so that it will be possible to implement. For use on low end wearable computers, the Tinmith-evo5 software architecture has been implemented using C++ and the infrastructure provided is designed to be light weight and optimised for the most common applications. For example, object communication is by default performed using function calls, and network distribution (with associated performance penalties) is only used when required in rare cases.

Interfacing to mobile outdoor hardware also has some other unique properties compared to working fixed indoors. The backpack's batteries discharge after an hour or two of operation, and so it is desirable that the user can swap the batteries without restarting. Some devices such as cameras need to be reinitialised before the software can continue operation, and applications should be designed to handle these restart conditions if possible. Some operating systems may suffer internal errors when cameras are disconnected during use however, and there may be no graceful recovery mechanism for software applications. Another problem with mobile outdoor AR is that the hardware in use is constantly changing. For example, in the last 2-3 years, slow USB cameras appeared and were then replaced by Firewire, and USB interfaces have allowed the chaining of virtually any number of devices rather than being limited to only the fixed serial and PCMCIA ports on a laptop.

Apart from just interfaces, the tracking technology being used also is quite limited in comparison to outdoor environments. For example, for many fixed indoor AR and VR applications there are a number of excellent tracking systems based on ultrasonics, active magnetics, and fiducial markers. When working outdoors however, none of these systems are available for use since they require fixed mounting or are affected by noise generated outdoors. The equivalent hardware for outdoor tracking tends to have accuracy values at least an order of magnitude worse (GPS is at best 2 cm, while indoor trackers may be submillimetre) and so improvements are constantly desired. As funding is acquired or as technology improves, these devices are replaced and so having suitable hardware abstraction and configuration layers that can support this helps to improve application portability.

Given that outdoor tracking hardware is generally at least an order of magnitude worse in accuracy than indoors, this must be kept in mind when implementing user interfaces. Many VR systems implement menus, toolbars, or objects that are grabbed directly by the user with fine grained hand tracking, but this is difficult to achieve outdoors. We believe that the user interface must be tailored to the particular tracking and display technologies available, as these will never be perfect and so will always be a concern. While it is possible to borrow ideas from existing domains such as VR, some of these will not work with lower resolution tracking or the different display type and so alternatives may be required. While we previously stated that working with AR is similar to VR type problems, differences between user interfaces may be required because the technology is not quite the same. Without considering these differences carefully, ideas which seem useful from one domain may not be possible to implement in another.

5 Content design

Content for AR and VR systems is traditionally designed on 2D desktop systems running software such as 3D Studio Max or AutoCAD. These programs allow the specification of models that may then be displayed on a HMD with a separate AR system. However, a main requirement of AR is that the models are overlaid onto the physical world and a desirable goal is for the user to achieve a sense of presence with the virtual objects. Actual physical coordinates must be used to represent object locations and the object must be presented correctly to the user. Since the previously mentioned design tools are both desktop based, there is no way to verify the models without loading them onto an AR system and trying them out. While the modeller can perform careful and accurate measurements of most of the object properties, this is tedious and even then there is still no guarantee the user will be comfortable with the model or find it useful. Verification allows us to answer the following questions: How do you know if the object is correctly placed into the physical world? How do you know if the object has the correct proportions, colours, shading, and other visual effects? Is there too much or too little detail for the user's needs? These are not just questions about supplying accurate measurements and tracking, but human perception issues that are important to get right because otherwise the models may be confusing or not seem realistic enough.

Rather than having the user switch between a desktop system for modelling and an AR system for verification, we propose the use of a single AR system to perform both actions simultaneously. An AR system would then be a standalone application that is independent of any desktop computers and users, and would allow a system initially containing a blank universe to be used for model creation rather than just consumption. During the modelling process, the user can verify the current model against the physical world, instantly seeing where changes need to be made because the process is performed in situ. Brooks also describes iterative modelling approaches as a good way of capturing 3D models of objects in the physical world for VR applications [5]. By creating an initial approximate model and iteratively refining those parts that require it according to the judgement of the user, a model can be created that adequately meets the requirements without wasting time on unnecessary detail.

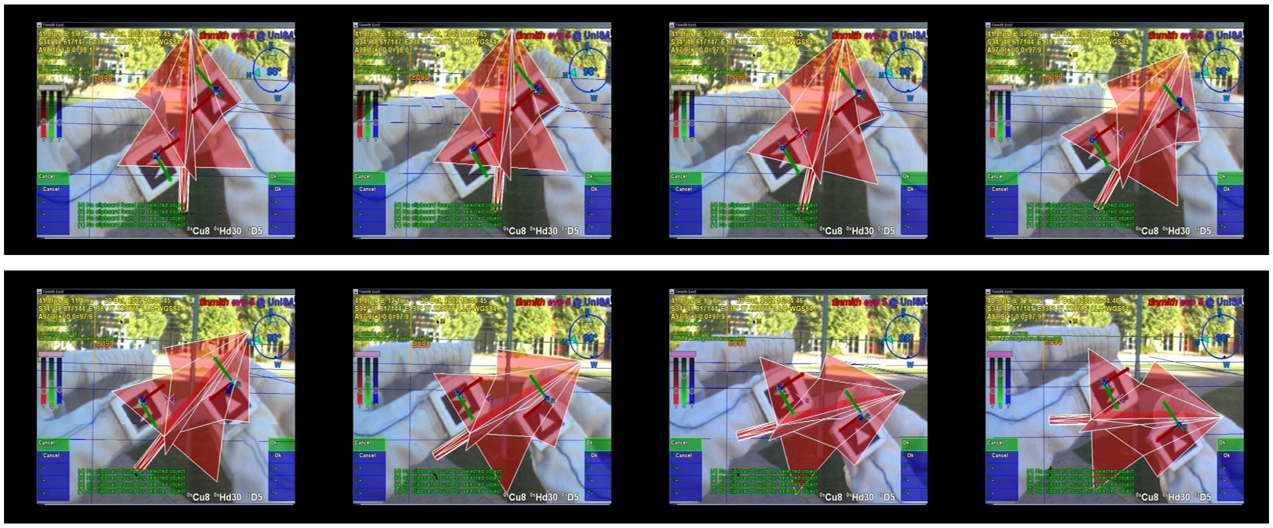


Figure 4 – Filmstrip showing a rotation operation being performed on a virtual tree at a distance using two handed input techniques

The Tinmith-Metro system [19] [21] takes the previously described concepts of real time modelling and applies them to the AR domain. In Tinmith-Metro, the user is able to walk around outdoors and use specialised techniques we have developed named *construction at a distance* to form planes, delimit large ground features, form solid shapes, specify constructive solid geometry operations, manipulate objects, and carve away parts of objects. Figure 4 shows a filmstrip of the view through the HMD showing a user rotating at a distance a 3D object placed in the environment. A key part of the Tinmith-Metro modelling system is that most modelling can occur without the user having to physically touch the object, allowing the capture of objects at large distances or at scales beyond the user's reach. We believe that AR systems implementing modelling techniques such as these are the only way to design accurate static content for AR because they support critical verification that is not possible on a desktop.

Another area we have performed some initial experimentation in is with the use of alternative views. Figure 5 shows an example where an external VR view is integrated into the application, allowing the user to improve their situational awareness by seeing the world from different viewpoints within the HMD. This type of view was initially introduced to VR by Koller [12], and we have implemented a number of other world and body relative views as well as

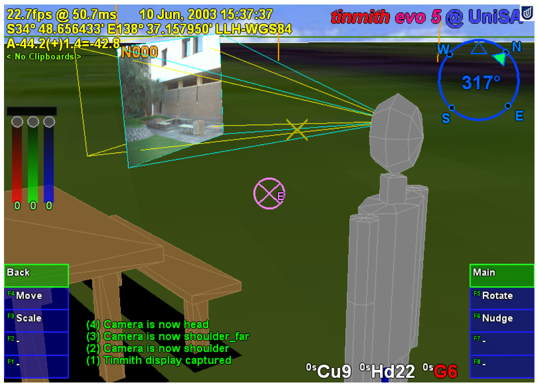

```
22.7fps @ 50.7ms     10 Jun, 2003 15:37:37
S34° 48.656433' E138° 37.157950' LLH-WGS84
A-44.2(+)1.4=-42.8
< No Clipboards >
```
```
(4) Camera is now head
(3) Camera is now shoulder_far
(2) Camera is now shoulder
(1) Tinmith display captured
```

Figure 5 – External VR view of an AR user near a virtual table. A live video stream is shown in the user's frustum to indicate what they are seeing. This view is useful for collaborative applications.

a polygon showing live video from the user's AR head camera. This same view is also useful for collaborative applications where an indoor user wants to monitor the actions of an outdoor wearable user performing a modelling task.

The development of dynamic content such as animations or attaching scripts to AR objects further increases the demands on user interface design and has not been addressed yet. High level authoring tools such as ALICE [18] have been developed to allow users to implement simple desktop VR applications with a point and click mouse based scripting language. We envisage modelling systems such as Tinmith-Metro extended to support the powerful capabilities and ease of use of ALICE, but modified to work intuitively in an AR environment. To support this, development of new user interfaces for AR that are tailored for the environment will be required to support the rich command set needed.

6 Conclusion

In this paper we have described our position on a number of areas that are related to software engineering for augmented reality. A comparison between AR and VR was made to explain how it is possible to leverage similar techniques and existing research from VR into AR. A common problem with both AR and VR is that they share a lack of higher level software toolkits, especially those that provide user interface abstractions. Good user interface design is the key to developing useful AR applications that can perform many functions and yet still remain easy to use. The Tinmith-evo5 software architecture was presented as a way of supporting the development of AR applications, with a particular focus on mobile outdoor AR environments although not limited to only these. The Tinmith-Metro application is presented as an example of a user interface that can support complex modelling tasks, allowing it to address problems with the design of content for AR systems. By using an AR system to design content rather than a non-AR desktop machine, we provide the ability to perform verification in real time with the modelling operation.

7 References

[1] Bauer, M., Bruegge, B., Klinker, G., MacWilliams, A., Reicher, T., Ris, S., Sandor, C., and Wagner, M. Design of a Component-Based Augmented Reality Framework. In *2nd Int'l Symposium on Augmented Reality,* pp 45-54, New York, NY, Oct 2001.

[2] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Cruz-Neira, C. VR Juggler: A Virtual Platform for Virtual Reality Application Development. In *IEEE Virtual Reality 2001,* pp 89-96, Yokohama, Japan, Mar 2001.

[3] Blach, R., Landauer, J., Rosch, A., and Simon, A. *A Highly Flexible Virtual Reality System.* Future Generation Computer Systems, 1998.

[4] Bolt, R. A. "Put-That-There" : Voice and Gesture at the Graphics Interface. In *ACM SIGGRAPH 1980,* pp 262-270, Seattle, Wa, Jul 1980.

[5] Brooks, F. P. *What's Real About Virtual Reality?* IEEE Computer Graphics and Applications, Vol. 19, No. 6, pp 16-27, 1999.

[6] Forsberg, A., Herndon, K. P., and Zeleznik, R. Aperture Based Selection for Immersive Virtual Environments. In *9th Int'l Symposium on User Interface Software and Technology,* pp 95-96, Seattle, Wa, Nov 1996.

[7] Frecon, E. and Stenius, M. *DIVE: A Scaleable Network Architecture For Distributed Virtual Environments.* Distributed Systems Engineering Journal, Vol. 5, No. 3, pp 91-100, 1998.

[8] Gobbetti, E. and Balaguer, J.-F. VB2: An Architecture For Interaction In Synthetic Worlds. In *6th Int'l ACM Symposium on User Interface Software and Technology,* pp 167-178, Atlanta, Ga, Nov 1993.

[9] Grimsdale, G. dVS - distributed virtual environment system. In *Proc. Computer Graphics 1991 Conference,* London, UK, 1991.

[10] Hubbold, R., Cook, J., Keates, M., Gibson, S., Howard, T., Murta, A., West, A., and Pettifer, S. GNU/MAVERICK - A micro-kernel for large-scale virtual environments. In *ACM Virtual Reality Software Technology,* pp 66-73, London, UK, Dec 1999.

[11] Kelso, J., Arsenault, L. E., Satterfield, S. G., and Kriz, R. D. DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments. In *IEEE Virtual Reality 2002,* Orlando, Fl, Mar 2002.

[12] Koller, D. R., Mine, M. R., and Hudson, S. E. Head-Tracked Orbital Viewing: An Interaction Technique for Immersive Virtual Environments. In *9th Int'l Symposium on User Interface Software Technology,* pp 81-82, Seattle, Wa, Nov 1996.

[13] Liang, J. and Green, M. Geometric Modelling Using Six Degrees of Freedom Input Devices. In *3rd Int'l Conference on CAD and Computer Graphics,* pp 217-222, Beijing, China, Aug 1993.

[14] MacIntyre, B. and Feiner, S. Language-Level Support for Exploratory Programming of Distributed Virtual Environments. In *9th Int'l Symposium on User Interface Software and Technology,* pp 83-94, Seattle, WA, Nov 1996.

[15] Milgram, P. and Kishino, F. *A Taxonomy of Mixed Reality Visual Displays.* IEICE Trans. Information Systems, Vol. E77-D, No. 12, pp 1321-1329, 1994.

[16] Mine, M., Brooks, F. P., and Sequin, C. H. Moving Objects In Space: Exploiting Proprioception In Virtual-Environment Interaction. In *ACM SIGGRAPH 1997,* pp 19-26, Los Angeles, Ca, Aug 1997.

[17] Multigen. *SmartScene.* http://www.multigen.com

[18] Pausch, R., *et al. Alice: A rapid prototyping system for 3D graphics.* IEEE Computer Graphics and Applications, Vol. 15, No. 3, pp 8-11, 1995.

[19] Piekarski, W. and Thomas, B. H. Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer. In *5th Int'l Symposium on Wearable Computers,* pp 31-38, Zurich, Switzerland, Oct 2001.

[20] Piekarski, W., Avery, B., Thomas, B. H., and Malbezin, P. Hybrid Indoor and Outdoor Tracking for Mobile 3D Mixed Reality. In *2nd Int'l Symposium on Mixed and Augmented Reality,* Tokyo, Japan, Oct 2003.

[21] Piekarski, W. and Thomas, B. H. Interactive Augmented Reality Techniques for Construction at a Distance of 3D Geometry. In *Immersive Projection Technology / Eurographics Virtual Environments,* Zurich, Switzerland, May 2003.

[22] Piekarski, W. and Thomas, B. H. An Object-Oriented Software Architecture for 3D Mixed Reality Applications. In *2nd Int'l Symposium on Mixed and Augmented Reality,* Tokyo, Japan, Oct 2003.

[23] Pierce, J. S., Forsberg, A., Conway, M. J., Hong, S., Zeleznik, R., and Mine, M. R. Image Plane Interaction Techniques in 3D Immersive Environments. In *1997 Symposium on Interactive 3D Graphics,* pp 39-43, Providence, RI, Apr 1997.

[24] Reitmayr, G. and Schmalstieg, D. An Open Software Architecture for Virtual Reality Interaction. In *Virtual Reality Software Technology,* Banff, Canada, Nov 2001.

[25] Robinett, W. and Holloway, R. Implementation of Flying, Scaling, and Grabbing in Virtual Worlds. In *1992 ACM Symposium on Interactive 3D Graphics,* pp 189-192, Cambridge, Ma, Mar 1992.

[26] Schmalstieg, D., Fuhrmann, A., and Hesina, G. Bridging Multiple User Interface Dimensions with Augmented Reality. In *3rd Int'l Symposium on Augmented Reality,* pp 20-29, Munich, Germany, Oct 2000.

[27] Sense8 Incorporated. *World Toolkit.* http://www.sense8.com

[28] Shaw, C., Green, M., Liang, J., and Sun, Y. *Decoupled Simulation in Virtual Reality with The MR Toolkit.* ACM Transactions on Information Systems, Vol. 11, No. 3, pp 287-317, 1993.

[29] Taylor, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T. VRPN: A Device-Independent, Network-Transparent VR Peripheral System. In *ACM Virtual Reality Software Technology,* pp 55-61, Banff, Canada, Nov 15-17, 2001.

[30] Tramberend, H. Avocado: A Distributed Virtual Reality Framework. In *IEEE Virtual Reality 1999,* pp 14-21, Houston, Tx, Mar 1999.

[31] VRML Consortium Incorporated. *The Virtual Reality Modeling Language.* In ISO/IEC 14772-1:1997, 1997.