ARWin - A Desktop Augmented Reality Window Manager

UCSB Tech Report 2003-12

Stephen DiVerdi, Daniel Nurmi, Tobias Höllerer
Department of Computer Science
University of California, Santa Barbara, CA 93106
{sdiverdi,nurmi,holl}@cs.ucsb.edu

May 2003

Abstract

We present ARWin, a single user 3D augmented reality desktop window manager, placing 3D user interfaces into a physical desktop workspace. We explain our design considerations and system architecture, exhibiting the ease with which such a system can be developed and used. We showcase a number of novel 3D applications, which take advantage of the environment to provide more powerful interactions and data visualizations. We elaborate on how to support 2D legacy application functionality. Finally, we discuss future possibilities for application and interaction development.

1 Introduction

For the past twenty years, desktop computer use has been dominated by the 2D Windows-Icons-Menus-Pointing (WIMP) paradigm. Through major advances in computing power and systematic improvements in 3D graphics and tracking support, it is now possible to create 3D augmented environments with unprecedented ease. Over a period of less than three months we created a 3D augmented desktop that can run and display 2D legacy applications, some specifically enhanced by 3D data visualizations, and a variety of hybrid data handling and interaction techniques.

Standard desktop computers, coupled with a head-worn display and camera, are adequately equipped for the task of presenting the user with an augmented version of his physical desktop workspace. Such an environment can make available a variety of new interaction techniques. Computer programs can populate physical space and make use of electronically enhanced versions of the physical objects commonly used in an office environment (see Figure 1).

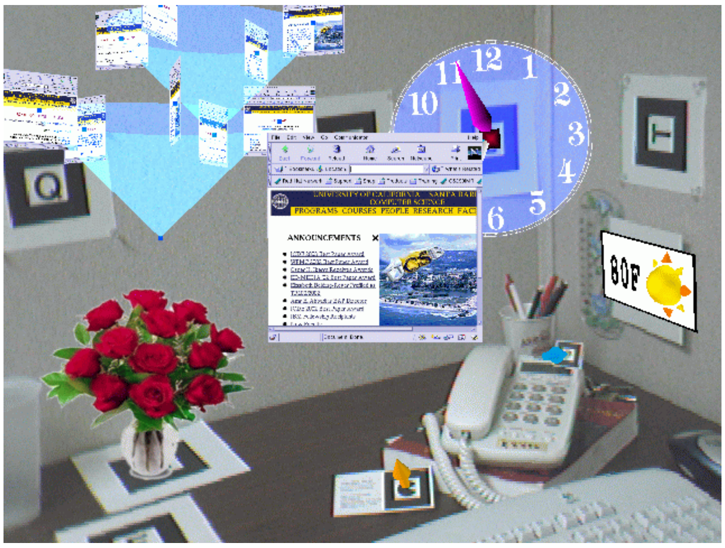

Figure 1: A typical ARWin desktop, as experienced through a video see-through head-worn display (Sony LDI-A55 with Point Grey Firefly camera). Applications are (clockwise from right) weather report, tagged phone, business card, flowers, web browser, clock.

Research in 3D data visualization [6], vision-based tracking and tangible interfaces [3], migration of 2D windows into 3D environments [2, 7], as well as 3D AR workspaces [8], has advanced the state of the art of 3D user interfaces to a point where we can put them to a rigorous test: everyday office productivity.

ARWin is our prototype environment for augmented reality desktop management. It allows the user to work in a familiar fashion with traditional 2D GUI applications, while introducing novel applications that are developed specifically with the 3D workspace in mind. These applications' geometry can mimic or extend traditional desktop objects such as a clock or calendar, or can spatially visualize information, such as web or file hierarchies. Thanks to the extra dimension in a volumetric workspace, these applications can also interact in ways previously impossible due to the limitations of a two-dimensional desktop. Placing the applications in space rather than minimizing them to a dock or menu allows persistent inter-application links and dataflows to be visualized graphically. It also makes possible the interaction of applications based on physical proximity.

While AR lends itself very well to multi-user collaborative work [1, 8], our application scenario purposefully focuses on usability benefits in the single-user case, which is how most computer users spend the majority of their time.

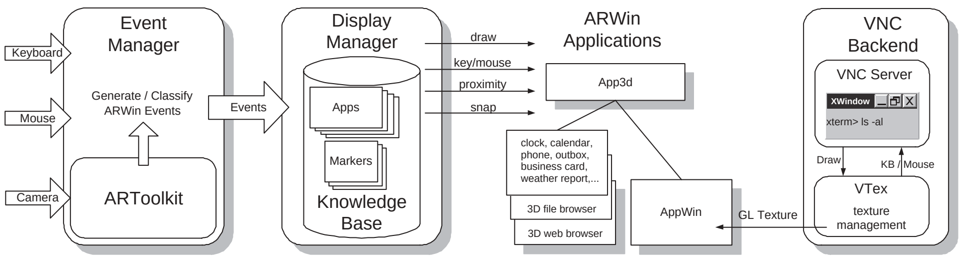

Figure 2: ARWin system architecture. The event manager turns hardware input into events. The display manager distributes events among applications which subclass App3d. AppWin uses a texture map of a VNC server to display X-applications.

2 Architecture

ARWin essentially replaces the combination of a display manager, such as X windows, and a window manager, such as TWM (see Figure 2).

The event manager's primary responsibility is to process input and generate ARWin specific events. For example, the camera input prompts the event manager to perform marker recognition and tracking. Markers affixed to the desk and walls in the user's workspace are detected by the ARToolkit [3], which calculates orientation and position information. This provides a global coordinate space in the volume above the user's desk where applications may be placed. *Floating* markers, which the user can manipulate freely within the environment, are also used to dynamically attach individual applications to. Input from standard hardware devices (eg. keyboard and mouse) is captured using the Simple DirectMedia Library. These events can either be passed along to the display manager unmodified, or may trigger more complicated events, such as window dragging or menu interaction. Control is then passed to the display manager.

The display manager's primary responsibility is to use the application knowledge base to determine where events should be delivered. The *draw* event, for instance, must be delivered to all visible applications while keyboard and mouse events need only be delivered to the application that is currently in focus. The display manager also sets up the context for these events - when a draw event is issued to an application, the display manager first sets up the transformation matrix to provide the application with a local coordinate system to draw in. Certain events are handled by the display manager as well, such as mouse events that move applications within the environment.

Applications designed for ARWin implement a generic App3d superclass that provides callback functions for each type of event ARWin can generate. To handle various events, the developer need only implement the appropriate functions. Applications have control over their appearances via member fields in the App3d class that the display manager reads for look and feel (eg. border style, default position).

A special case of application which we call AppWin allows users to operate their legacy X applications (xterm, emacs) within ARWin. AppWin first runs the X application on a virtual VNC enabled X server [5]. Our custom VNC client connects to this server and draws the screen image data into an OpenGL texture map, which is then applied to a polygon in the 3D space above the user's desk. To maximize application visibility, the focused application is always oriented to face the user head-on.

3 Applications

Developing intuitive 3D applications constituted a major part of our work (see Figure 1). We've devised an informal taxonomy to classify these applications in terms of the physical or virtual nature of their geometry and functionality.

On the physical end of the spectrum, there are tagged physical objects, such as the phone or business card. They are fully functional, regular desk objects that have a small marker attached to them. These markers provide ARWin with location information, as well as metadata (from a knowledge base) about the physical object, enabling their interactions with other ARWin applications.

Next we have hybrid physical / virtual objects, such as the weather report. These are tagged physical objects (a wall thermometer) which have some associated virtual geometry (the sunny / cloudy indicator). The virtual component augments the functionality of the otherwise unmodified physical object.

On the fully virtual end of the spectrum, we have virtual objects that mimic the geometry and functionality of their physical counterparts, such as the wall clock or flowers. These applications perform the same useful behaviors, but without the downsides. For example. the flowers add a nice aesthetic touch to a virtual desktop, but without the need for regular maitenance.

Finally, there are virtual objects with no physical counterpart, such as the web and file browsers. These applications are extensions of the traditional counterparts, specifically enhanced to take advantage of the 3D environment. Each uses cone trees [6] to enhance visualization and manipulation of large data sets. The file browser displays the file system in a cone tree format to allow for ease of selection large numbers of diverse groups of files and directories. Filters can also be applied to the conetree to select a subset of the nodes based on various filesystem metadata. The web browser allows for normal navigation of web pages, but at the same time uses a cone tree to display the web hierarchy from the current viewed page. The hierarchy can include both a precache of web pages linked to from the current page, as well as an organized history of previously viewed pages.

4 Interactions

To take full advantage of the 3D workspace, ARWin extends traditional application interaction by providing a *proximity* event. When two (or more) applications are

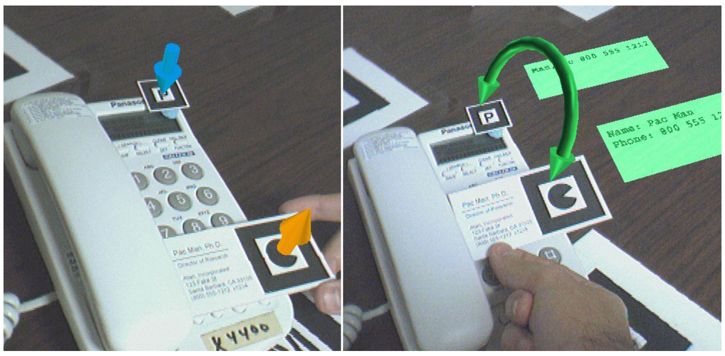

Figure 3: Application interaction. *left*: phone (data handler) and business card (data container). *right*: phone receives business card data, displaying and storing the name and phone number.

placed near each other (defined by an application-specific threshold), each application receives an event notifying it of the members of its neighborhood. What the applications do with this knowledge is left to the developer.

For our example applications, we've classified each in terms of their roles during a proximity event. Here, *metadata* refers to information about the application in its current state, such as the time on the clock or the temperature on the weather indicator. Objects which contain useful metadata are called data containers and are represented by an orange arrow pointing away from the object. Objects which can operate on metadata are data handlers and are represented by a blue arrow pointing towards the object. When a handler and a container are placed near each other, the handler can request the metadata from the container and then process it appropriately (see Figure 3). This may include storing the data, displaying it, or something else entirely.

ARWin provides an additional new event to applications, called a *snap* event. For some applications, such as a web browser or xterm, it is appropriate to provide two modes of operation - one in 3D and one in 2D. The snap event gives the target application control over the display so it can draw at full resolution for maximum visibility. The input events can be handled differently as well, to provide an interface appropriate to this 2D representation. This is particularly useful for AppWin objects - when snapped, the user is presented with the traditional interface to the X windows application.

5 Future Work

A major benefit of a system like ARWin is that it's provides a useful testbed for new ideas and integration of established technologies in an augmented reality environment. We have many ideas for features we'd like to add and new concepts we'd like to test out in a developed system.

One of our next goals will be to replace the traditional mouse interface with something more appropriate to the three-dimensional workspace. At the moment, ARWin uses a screen stabilized pointer controlled by a mouse. This is unintuitive for a number of reasons, largely because the user has an expectation of how a mouse will perform which does not clearly translate into ARWin. One major problem is the mouse is screen stabilized - that is, when the user moves his or her head, the mouse moves with it, rather than holding it's world position. This is unintuitive - a user would expect a pointing device to remain world stabilized. We are considering a variety of replacements, from using a regular mouse to control a 3D cursor, to using a more intuitive 3DOF device like a spaceball. The most attractive possibility is to integrate hand tracking and gesture recognition algorithms so the user's hand could be the pointing device.

We also have many different avenues to explore in the realm of inter-application interactions. ARWin's proximity events and data container / handler abstraction allows for the easy transfer and manipulation of small metadata, but the potential for connecting flows of more significant application data and control information is obvious, much along the same lines as data tiles' [4] information sharing. Once these inter application streams are established the question of how best to visualize the sharing of data is immediately raised. Simply drawing static lines to represent all links quickly becomes muddled and annoying, but some visual cue to the user indicating the link and transfer of data is obviously necessary.

6 Conclusions

We have presented ARWin, a practical augmented reality desktop environment. We've demonstrated ARWin's ability to execute traditional X applications as well as novel 3D applicationss which showcase ARWin's enhancements.

In the future, ARWin will be an invaluable testing environment for technology integration. We will explore user interaction techniques such as hand tracking and gesture recognition, as well as more advanced application interaction and data visualization technologies.

It's important to understand the limitations of ARWin. Tracking with the ARToolkit is currently limited by camera resolution and the careful calibration of known markers attached to the desk and walls. Techniques have been developed to address the marker placement issue, and would be appropriate to integrate with ARWin. The low resolution of HMDs is another fundamental limitation, which we'd like to address by allowing applications to snap to a high resolution desktop LCD monitor instead of the HMD.

References

[1] A. Butz, T. Höllerer, S. Feiner, B. MacIntyre, and C. Beshers. Enveloping users and computers in a collaborative 3D augmented reality. In *IWAR '99*, pages 35–44, San

Francisco, CA, Oct. 20–21 1999.

[2] S. Feiner, B. MacIntyre, M. Haupt, and E. Solomon. Windows on the world: 2D windows for 3D augmented reality. In *ACM UIST '93*, pages 145–155, Atlanta, GA, Nov. 3–5 1993.

[3] I. Poupyrev, D. Tan, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsutani. Developing a generic augmented-reality interface. *Computer*, 35(3):44–50, Mar. 2002.

[4] J. Rekimoto, B. Ullmer, and H. Oba. Datatiles: a modular platform for mixed physical and graphical interactions. In *CHI*, pages 269–276, 2001.

[5] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. In *IEEE Internet Computing*, pages 33–38, Jan./Feb. 1998.

[6] G. Robertson, J. Mackinlay, and S. Card. Cone trees: Animated 3D visualizations of hierarchical information. In *ACM CHI'91*, pages 189–194, 1991.

[7] G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. The task gallery: a 3D window manager. In *ACM CHI '00*, pages 494–501, Apr. 1–6 2000.

[8] D. Schmalstieg, A. Fuhrmann, and G. Hesina. Bridging multiple user interface dimensions with augmented reality. In *IEEE/ACM ISAR 2000*, pages 20–29, Oct. 5–6 2000.