The Immersion Interface Concept

Dustin Silverman
dustbin@cs.utexas.edu
University of Texas

Donald S. Fussell
Department of Computer Sciences
The University of Texas at Austin

Abstract

This paper presents a concept for organizing and interfacing with 3D windowing systems. Its intention is to provide a means of replacing the 2D desktop metaphor. The replacement metaphor is that of an office, a concept that more directly maps to 3D space.

1 Introduction

The WIMP interface concept and the desktop metaphor concept were originally developed at Xerox PARC in 1973, approximately thirty-three years ago as of the time of this writing. There have been obvious improvements in the concept over time, though overall it has changed very little. The result of this is that certain restrictions in the concept are now more obvious due to how much both hardware and software have changed.

One of the most basic restrictions is the restraining of the usable space to the resolution of the monitor, which often results in a cluttered virtual workspace filled with overlapping windows [4]. There have been various attempts to solve this restriction, the most common of which is adding the ability to switch between several desktop spaces [2]. However this does not solve the related difficulty of having multiple open windows visible at the same time. These restrictions are less of a difficulty on larger monitor and multi monitor systems [4], though these solutions are expensive and consume physical space.

Given the prevalence of graphics cards in recent computers, it is now practical to have a 3D windowing system, an engine that allows for 3D graphics to be used in rendering applications and the environment in which they are placed, in a widely distributed operating system. The result of this as it applies to academia is that 3D windowing systems are now a viable area to research for solutions to the restrictions that exist in the current desktop metaphor. Also it should be noted that, while several companies have begun to create 3D windowing systems, none have presented a new concept for organizing and interacting with applications in 3D space. Instead they have retained the desktop concept [6]. While 3D windowing systems implementing the desktop concept can have a few new interesting visual effects and additional capabilities, they cannot take full advantage of the possibilities provided by 3D windowing systems, as they are in effect still 2D systems with 3D windows [5] or, as it is sometimes called, 2½D systems [3].

With these things as incentive, we decided to investigate the usability of the three dimensional metaphor of an office as a replacement for the desktop metaphor as the layout concept. We have named this system Immersion, as it places the user within a 3D

space that virtually surrounds them. Additionally a concept for replacing the HUD-like portions of the desktop interface, the portions conceptually attached to the monitor such as the start menu and task bar, is explained. It is a tree structure consisting of what we refer to as workspace nodes, where each node is similar to a folder in that it contains files and links to applications, but additionally each contains active applications. The layout of each node uses the Immersion concept, with its office metaphor. In this paper we will outline these systems and how we have implemented them for the purpose of future testing and refining of the ideas.

2 Immersion

In the Immersion concept, the active applications are placed surrounding the user in a 3D virtual space, with the monitor or monitors acting as viewing portals providing the user with a means of looking into the virtual space. Applications can be dragged vertically, dragged horizontally in an orbit, and moved closer to or further from the user. The user's view may be rotated about the vertical axis using an additional input device or onscreen interface.

Optionally, but within the scope of the Immersion concept, the user's view may be controlled in the other five degrees of freedom available in 3D space. Of the other degrees of freedom, two could easily be used. They are: pitch, which is rotation about a horizontal axis perpendicular to the user's primary view direction, and movement along the vertical axis. Which of these additional options are used depends primarily on the design of the view control input device.

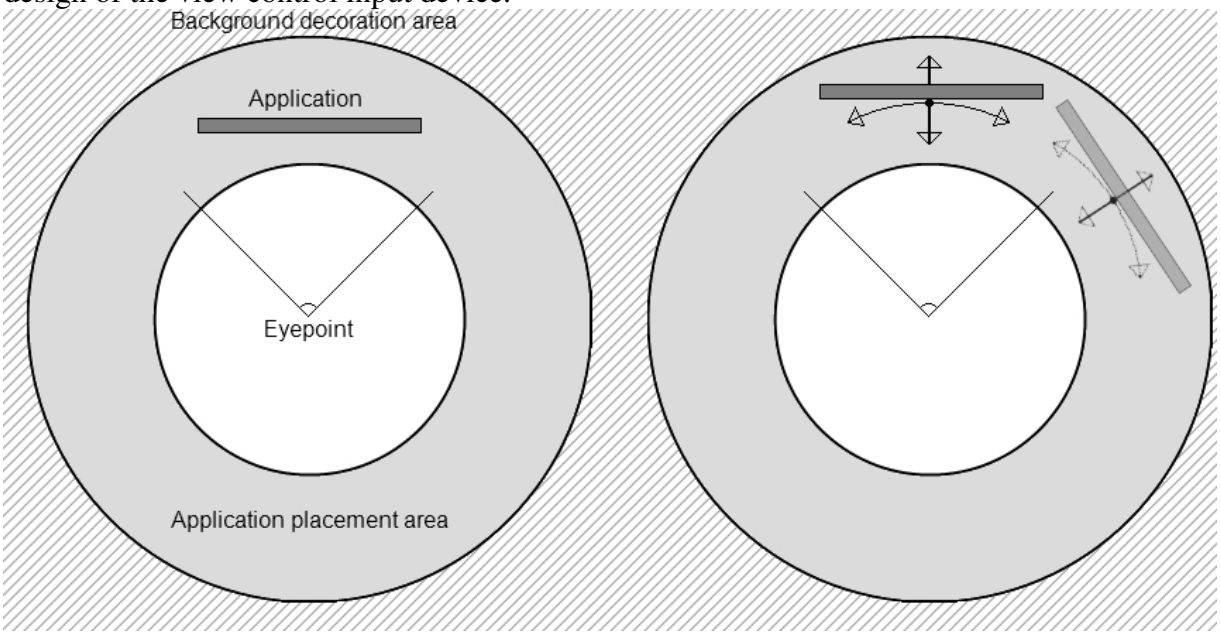

Figure 1: Left: the layout of the Immersion concept.
Right: an example of moving an application in an orbit.

The above descriptions give all the information that is necessary to implement something that fulfills the Immersion concept. However with only the basic ideas implemented, the concept barely begins to take advantage of the potential offered by a 3D windowing system. To continue exploring the potential, several other unique ideas

will be presented and explored further in the next few sections. The hope is that in the same way that a start menu is not an integral part of the desktop concept, but is commonly used by many systems, these additional ideas, while not integral to the Immersion concept, will be found to be useful and be commonly adopted.

The Immersion concept is relatively unique among what has been developed for 3D windowing systems. Most single user window management concepts have instead focused on using 3D windows that work with the 2D desktop concept [3][5][6]. Most systems which intend to provide multiple users with a common workspace allow windows to be placed anywhere in a virtual environment, leaving it up to the users to decide how to arrange windows [7]. Such systems, in placing the burden on the user, require that the interaction system be more general, resulting in a need for a more complex control scheme than is wanted in common user interaction.

For the purpose of testing the concepts outlined, we chose to implement the Immersion system using the Looking Glass 3D windowing system. Additionally we chose to use as a means of interface a chair with a monitor or monitors mounted on it and a rotation sensor attached to the base so that the virtual space could be more directly correlated to the real world. This is explained further in section four. Another feature present in the test implementation of Immersion is a system of nodes that act as in a manner that will be referred to as workspaces. This will be explained further in the next section.

3 Workspaces

Workspaces are analogous to folders in that they are visual representations of system resources. They go beyond folders in that they can hold both the means of initiating an application, such as a file or a shortcut to an application, and an active application. This is conceptually useful in that it can provide a similar functionality to having multiple desktops, but with the added advantage of being able to be organized in a manner similar to folders.

A potential difficulty with this system is keeping track of where running applications are located. As an intuitive, at least partial, solution to this, paths to other nodes will be color-coded. To nodes containing active applications there will be blue paths. To nodes containing no active applications, but with paths from them to nodes containing active applications or nodes of this type, there will be green paths. To nodes containing no active applications and with no paths to the two previously described types of nodes there will be red paths [Figure 2]. Paths to nodes closer to the root of the tree will be darker. Paths to nodes further from the root of the tree will be lighter. This may be all that is needed for the users to have an intuitive understanding as to where active applications are located, however it may also not be enough information. Determining this is a subject for further research involving user testing.

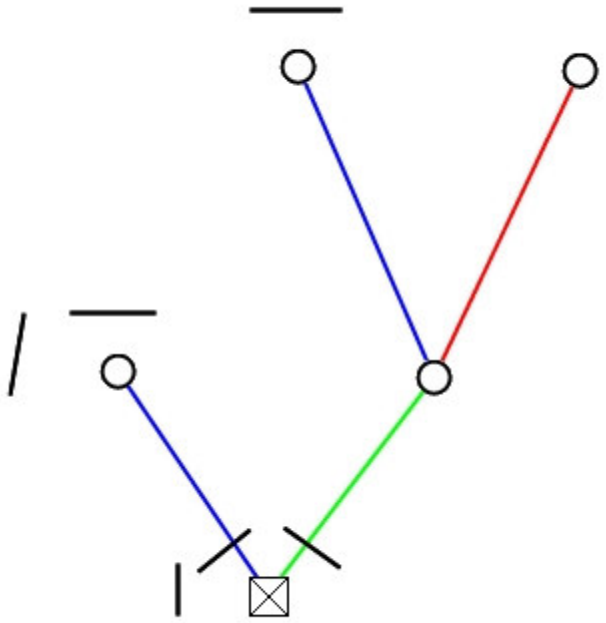

Figure 2: Shown here is a sketch of an example workspace tree with the circles representing nodes, the box representing the user's location at a node and the black lines representing active applications.

The prototype workspace system will be fairly limited, with additional features added as needed. The goal functionality for workspaces is to represent a particular folder's contents in some customizable iconic form and provide a location around which open applications will be displayed. The goal interface system for workspaces will allow the user to attach applications either to a workspace node or to the user, thereby allowing users to keep applications that are needed for multiple tasks always with them. Additionally the goal interface will provide a means of viewing the node system.

We envision workspaces as a means of organizing files that are accessed and created by the user. To that end each workspace will follow a template defined by the user or by an installed application that relates to what is in that workspace. Templates could be for associated file types such as music files, office documents or images. Templates could also be for individual projects such as programming projects or business projects.

In an example scenario for the first case, a user creates a new workspace node for a collection of pictures. The layout and visual style is then based on the existing template for collections of images. A likely visual style would be to have all the images immediately visible in a thumbnail form and tiled around the user for easy viewing. The workspace would include links to relevant applications, such as image manipulation software. The template may define that file types other than images not be visually represented, as they are not relevant to the workspace node.

In an example scenario for the second case, a user creates a new project in an IDE. The user's project is then created in a new workspace node that is accessible by a

path from the IDE's workspace node. Layout and visual style for the new node come from the IDE's template for project workspace nodes. Additional applications that are commonly used in association with projects would be accessible by links included in the template. Relevant documentation and tutorials would be accessible by links included in the template. The user could add additional subprojects that would be workspace nodes accessible by paths from the project node.

4 The Chair

In conceptualizing Immersion it was decided that a second human interface device was needed to provide a means of rotating the user's view. The simplest solution to this was to add a second mouse with horizontal rotation mapped to the x-axis. There are other fairly traditional solutions to this problem that could be considered such as a rotating knob, a thumb stick attached to the keyboard or, if the user wanted only one input device, an on-screen area could be set aside for use as a kind of control dial. In the spirit of considering non-traditional solutions that could more elegantly solve the problem, we considered the user's chair.

To understand why we might consider using a chair as the view controller, consider the metaphor of the office in which the rotation of the view to other portions of the desk is accomplished by the rotation of the user's chair. Then take that idea and consider if a screen were directly mounted to the user's chair and a rotation sensor to the base of it. It would then be possible for the user to rotate the chair to change their view in the virtual world as they would in an office in the real world.

The chair is a standard office chair with a wheeled structure attached in front that supports the monitors and the keyboard and a mouse attached to the bottom that reads the amount of rotation of the chair [Figure 3]. The mouse is attached to the seat portion and a patterned surface is attached to the base when the chair is rotated the mouse is moved over the surface. The mouse as a rotation sensor system is not ideal, but in place of a true rotation sensor it works as required for our purposes.

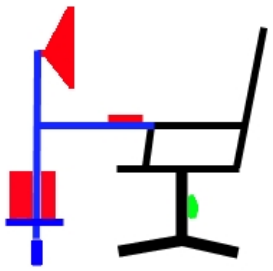

Figure 3: Side view of chair and framework: in red keyboard, mouse, monitor and computer; in green mouse used as rotation sensor; in blue wheeled framework; and in black rotating chair.

The ideal controller for the Immersion concept would be an augmented reality head mounted display that contains rotation sensors and motion sensors. Given a system of this kind, the user's view into the virtual space would be directly mapped to the real world. If the user moved their head forward in the real world, a corresponding movement would happen in the virtual and etcetera. As a result the control provided by the chair would inherently exist, as rotation of the chair would result in rotation of the user's head, which would be sensed by the display's rotation sensors. However, short of an augmented reality system, the chair's ability to provide the user with a more direct link between reality and a virtual space than a hand manipulated input will help to make human interface more intuitive.

The current drawback of the chair system is that it can only provide one degree of motion input (rotation about the vertical axis.) With hand manipulated input devices such as a second mouse or a thumbstick, two degrees of motion input are provided. The resulting restriction of freedom of movement in the virtual space is not very noticeable, but a solution to the problem should be considered for future research.

note as of the time of this writing the chair interface system has not been built due to mechanical engineering issues

5 Looking Glass

Looking Glass is a 3D windowing system for Linux built in java. It is capable of displaying existing X11 applications by rendering the X11 content and then applying it as a texture to a 3D window. The existing capabilities of this system gave us a good starting point for building an implementation of the Immersion concept, as our test system is able to run most applications created to run in X11. However, Looking Glass' interface concept is the desktop metaphor, and much of the code structure is built with this in mind, therefore extensive modification of the code was necessary to implement the Immersion concept.

Another difficulty encountered in implementing Immersion in Looking Glass came from a limitation in the input event system in java. Specifically, in java the built input events system is for only one mouse and one keyboard, and it requires any additional mice or keyboards to be implemented by the user outside of the existing java input device framework. Therefore, to receive input from a second mouse requires the addition of a new input handling thread, which for the Immersion Looking Glass implementation reads directly from a mouse stream. The components needing the second mouse's input then register with this thread and receive events when the second mouse moves or has a button clicked. With some additional work, this system was made compatible with existing mouse movement interpreting systems within Looking Glass so that the movement animation systems could be used.

Overall the decision to use Looking Glass has proven to be a good choice as we have benefited from being able to create the test version of the Immersion concept in a real operating system. Before finally finding and deciding on Looking Glass, we had considered creating a demonstration windowing system as an application built on a 3D engine such as OGRE. We had also considered some other windowing systems, but we

found that at the time no other system provided as close an integration with an existing framework or as truly 3D a windowing system as that of Looking Glass.

6 User Testing

The test implementation of Immersion that has been described here will eventually be used for user testing to determine the intuitiveness of the concept and how users respond to the various new ideas. Also in some areas of the concept there are several options to choose from in deciding how to implement, but no clear best option. In these cases the optimal way for us to choose the best option is to do user testing.

A feature that has several choices with no clear best option is the mapping of the y-axis of the second input device. The options are rotation on a horizontal axis perpendicular to the primary view direction, movement along the vertical axis, or no mapping at all. A case could be made for the rotation on a horizontal axis option because it will resemble the user rotating their head to look up or down. On the other hand the movement along the horizontal axis option would be useful, as it provides the user with essentially unlimited space in which to place applications and icons. The third option of having no change based on movement on the y-axis makes sense if a chair is being used as a means of control, as there is no means of physically moving on the y-axis with the chair.

Another area that would benefit from user testing is the navigation system for the workspace nodes. One option would be an extension of any folder-viewing concept, as the workspace system itself is an extension of the folder organized file system concept. Potentially it could also be a zoomed out three quarter view of the nodes themselves. Another possible option would be not to have any external view of the nodes, but instead have links within the nodes that are color coded to indicate whether the linked node contains active applications or nodes with active applications.

7 Conclusion

We have described how Immersion and associated concepts can provide solutions to the problems outlined in the introduction. The most important is that the space in which applications are placed is no longer limited by the resolution of the monitor. The problem of organizing applications and files into relevant groups is solved by the addition of workspace nodes. The ability to use a chair's rotation to provide input on how one looks into the virtual space provides a more direct interaction between the physical and the virtual, another step in the direction of augmented reality systems.

Now that the Immersion system and associated concepts have been described and a test system implemented, the next step will be refinement through user testing. In addition to deciding how to best implement the portions of the design as described in section six, this should include testing for the purpose of smoothing away rough edges and finding and fixing things that are not intuitive to the average user.

[1] Bowman, D., Kruijff, E., LaViola, J. & Poupyrev, I. (2001). An Introduction to 3D User Interface Design. *Presence, 10*(1), 96-108

[2] Ringel, M. When one isn't enough: an analysis of virtual desktop usage strategies and their implications for design. *CHI Extended Abstracts 2003*, ACM Press, 762-763.

[3] Leach, G., Al-Qaimari, G., Grieve, M., Jinks, N., & Makay, C. (1997). *Elements of a three-dimensional graphical user interface*. Retrieved May 1, 2006, from http://goanna.cs.rmit.edu.au/~gl/research/HCC/interact97.html

[4] Hutchings, D. R., Smith, G., Meyers, B., Czerwinski, M., & Robertson, G. Display space usage and window management operation comparisons between single monitor and multiple monitor users. *Advanced Visual Interfaces 2004*, ACM Press, 32 – 39.

[5] Chapuis, O. & Roussel, N. Metisse is not a 3D desktop! TR 1407, LRI, Université Paris-Sud, 2005.

[6] Kawahara, H., Byrne, P., Johnson, D. & Gadepalli, K. Project Looking Glass: A Comprehensive Overview of the Technology *Rev. 0.2 March 14, 2006.* Retrieved May 1, 2006, from https://www.dev.java.net/files/documents/1834/30923/LG3D-Overview.pdf

[7] Smith, D., Raab, A., Reed, D., & Kay, A. Croquet: A menagerie of new user interfaces. In Proceedings of C5 2004, pages 4–11. IEEE Computer Society, January 2004.