CHI'92 DEMONSTRATION OF *ART*

Larry Stead, Jim Hollan, Mark Rosenstein, and Will Hill

Computer Graphics and Interactive Media Research Group
Bell Communications Research
445 South Street, Morristown, NJ 07962-1910

Email: lstead@bellcore.com hollan@bellcore.com, mbr@bellcore.com, willhill@bellcore.com

ABSTRACT

Most current user interface development environments and the underlying window systems upon which they are built are based on a set of presuppositions about low-level machine capabilities that are rapidly being invalidated by current hardware advances. *ART* is a software environment designed in our research group that leverages hardware advances to facilitate rapid prototyping of novel interactive visualizations and exploration of new interface possibilities.

Keywords: Window Systems, Interactive Graphics, 3D, Graphical User Interfaces, Visualization.

INTRODUCTION

Art does not reproduce the visible; rather, it makes visible. Paul Klee

Thanks to Art, instead of seeing one world, our own, we see it multiplied and as many original Artists as there are, so many worlds are at our disposal. Marcel Proust

Most contemporary window systems are based on assumptions of single buffering, relatively slow rendering speed, two-dimensional fixed orthographic views, and meager sound facilities. Changes in current hardware are invalidating these presuppositions. For example, the decreasing cost of memory enables widespread use of double buffering. In double buffering the bitplanes of the framebuffer are partitioned into two buffers. One is visible and the other is invisible. An entire frame is rendered in the invisible buffer and then it is swapped with the visible buffer during the vertical retrace of the screen. This enables animation and simplifies programming. Faster CPU's and increasing use of the CPU to perform graphics operations, that at one time could only be effectively provided by special purpose hardware, now allow rendering speed to scale with CPU speed and makes code much more portable. This in turn can be expected to lead to an increasing use of animation techniques within the user interface.

Another consequence of decreasing memory cost and increasing CPU speed is the ability to employ 3-D. For example, the framebuffer can use additional bits to provide a *z-buffer*. It can be thought of as a set of integers associated with each pixel on the screen that, among other things, can represent the distance to the eye-point in the 3-D scene and thus be used to do hidden surface removal. Having additional display memory also enables the use of color. The use of color, shading, texture mapping, and other animation techniques make it possible to provide much more realistic interface objects.

In our view these and other hardware changes, such as availability of CD-quality sound and video, will fundamentally change the paradigm for interactive interface development. This paradigm shift raises new interface research issues. We not only need to understand what we can apply from what we know about the predominantly silent 2-D interfaces of the present but also we need to develop appropriate metaphors and ways of thinking about the new interface possibilities enabled by sound, 3-D, animation, video, and new methods of interaction such as sketching and 3-D input. The addition of all these new interface dimensions radically expands the space of interface possibilities.

CHI'92 Demonstration

In our demonstration at CHI'92 we introduce *ART*, a software environment designed to facilitate rapid prototyping of new interface possibilities. In the demonstration we describe some of the issues associated with going from 2-D to 3-D, discuss the underlying 3-D graphics model upon

which *ART* is based, describe a novel 3-D input facility we have found to be very effective, present a series of simple demonstrations, summarize the components of *ART*, give a more detailed description of what goes on behind the scenes when one does 3-D input and output, and end with a summary of the status of the current implementation and applications. Here we can only briefly allude to considerations associated with moving from 2-D to 3-D and to our motivations for implementing *ART*. More detail on each is forthcoming [3, 2].

Moving From 2-D to 3-D

A traditional 2-D window system consists of a finite viewing plane divided into multiple rectangular input and output regions, *windows*. In 3-D the model is an infinite space of three dimensions in which *objects* can be positioned. Most 2-D systems permit windows to overlap. In 3-D one object can partially obscure another object in a similar way by being *in front* of it, but objects may also interpenetrate each other as well as be positioned anywhere in the 3-D space. As viewing position changes the interrelationship of the objects, the way they look, also changes. Unlike simple 2-D window objects, 3-D objects are not constrained to be contiguous or connected. For example, a single object might consist of a cloud of particles. Possible relationships between hierarchically composed objects can also be more varied than in a traditional window system in which a child is typically clipped to a parent. In *ART* we are exploring a variety of relationships that we will demonstrate.

While one can very the size of a window in 2-D window systems, the contents of the window typically do not scale. Windows usually exist full size or iconified. The iconified window is a fixed size and maintains only an abstract relationship to the functional window. In 3-D, objects may be scaled arbitrarily rather than being restricted to the binary choice of icon versus window.

In 2-D, the standard input device is the mouse. The mouse is used to select windows for operations like moving and resizing, to control the input focus for devices like a keyboard, to position objects, and to choose items from menus. In 3-D some of these tasks, such as moving, are more complex. While there are many techniques for using the mouse as a 3-D input devices, it is the case that the mouse is fundamentally a 2-D device. In *ART* we are exploring the use of the mouse in concert with a true 3-D input device. The goal is to give users the impression that they can grab an object and naturally move it around in 3-D.

This is more complex than it might at first seem. One needs to deal with the coordinate systems of the object, the 3-D input device, as well as the camera. If, for example, the camera is on one side of the object versus the opposite, what is intuitively interpreted as left and right reverse. There are many subtle factors involved in developing a *physics* for the device that is natural.

Informational Physics

Of greater interest to us is the development and exploration of informational physics for *ART* objects. Our vision is of more dynamic informational entities and computationally-based work materials [1, 2] that exploit representations of tasks, semantic relationship explicit and implicit in information and in our interactions with it, and user-specified taliorings to provide effective, enjoyable, and beautiful places to work.

IMPLEMENTATION

ART is implemented in Common Lisp using CLOS (Common Lisp Object System). It provides a basic set of classes and mixins to support the interactive use of animation techniques and the exploration of new forms of information visualization. On top of these basic facilities a growing number of task-based information visualization applications are being implemented. We will demonstrate a number of these at the conference.

ART currently runs on Silicon Graphics machines.

REFERENCES

1. Hill, W., Hollan, J. Wroblewski, D. , & McCandless, T. Read Wear and Edit Wear, this proceedings.

2. Hollan, J., & Hill, W. Towards An Informational Physics Perspective For Interface Desgin, Bellcore techincal report, in prepartion.

3. Stead, L. The *ART* of User Interfaces, Bellcore technical report, in preparation.