Interface Abstractions for an *naplps* Page Creation System

Ernest Chang
Department of Computer Science
University of Victoria
Victoria, B.C. V8W 2Y2

ABSTRACT

Computer programs that allow humans to create pictures are called *paint* systems. Those that use geometrical shapes as building blocks, rather than brushes, are *element-based*, rather than *canvas-based*. The *naplps* graphics encoding standard, used primarily for videotex, lends itself to the creation of element-based paint systems. The user interface for such a system can be rather complex, because of the need to choose shapes, colours, textures, and the need for editing functions. This paper presents some interface techniques that present the user with a number of easily understood abstractions such as a colour bar and function buttons, that facilitate the creation of *naplps* graphics.

RESUME

Il y a les systèmes de *peinture* qui permit aux humains de façonner les images. Ces qui utilize les configurations géométrique au lieu des pinceaux s'appelent *element-based*, au lieu de *canvas-based*. La convention *naplps*, qu'on utilize dans le principe pour videotex, est convenable à réalization des systémes de peinture fondants sur les éléments. L'interaction avec ses systèmes deviendrait difficile à cause de la nécessité de choisir les configurations, les couleurs, les textures, et la complication des fonctions pour changer ces détails. Nous presentons quelques méthodes de faire présenter plusieurs abstractions comme un segment de couleur et les poussiors pour fonctions, qui facilitent la création des images *naplps*.

1. Introduction

A paint system[3,4,6] is a computer program that facilitates the interactive creation of graphic images, using techniques analagous to those found in traditional artistic media. Some paint systems treat the screen as a simple canvas, which is successively covered with colours and textures. The software, and user interface, is correspondingly simple since there is only one major function to support, that of adding more colour. The performance of this class of *canvas-based* paint systems is very fast when implemented on a frame buffer.

More flexible paint systems can define graphic elements, that are remembered as unique entities in display lists, even if they are covered on the screen by other elements. These entities, and subsets of them, can be selected, deleted, copied, moved, or modified. These *element-based* paint systems typically require more complex software, and a user-interface that is no longer analogous to traditional techniques in art. This problem is compounded by the addition of facilities for entering text in different sizes, colours, orientations and fonts. This paper deals with the demands that an element-based paint system makes of the user interface, in the context of the **naplps**[1] videotex environment, and the use of abstractions in its implementation.

2. naplps and Videotex

The North American Presentation Level Protocol Syntax[1], known commonly as *naplps*, is a communications standard using 7-bit or 8-bit codes to represent graphical and text images. The idea is to use the same bit combinations to define multiple code sets, with a mechanism for invoking a particular set as the **active** code set. Graphic elements, and their attributes, are represented by one code set, alphanumeric text by another, and so on.

An underlying assumption in **naplps** is that a function or attribute, once invoked, remains active until specifically reset. There is always a current graphical element, a current colour, etc. Another fundamental premise is that graphical images are constructed from dots, lines, arcs, rectangles and polygons. These are the building blocks for the image creator. The **naplps** approach to colour is also important: it assumes that a colour map is used, which permits a small number of colour *entries* to take on values from a larger range of colours.

Some implications of these characteristics are that the encoding is **sequential**, that the user **selects** the current active attributes, and that **colours** can be modified. To understand how these affect the user-

interface, we must refer to the physical context in which **naplps** is used.

To date, **naplps** has been used mainly in videotex[2], which transmits combined text and graphics as *pages* over low-speed communications lines to large numbers of subscribers. The cost-performance characteristics of the videotex market has produced hardware **naplps** decoders costing about $1000 that typically give 256 X 200 resolution, support 16 displayable from 4096 colours, communicate at 300 to 9600 baud, and use 8-bit microprocessors. This relatively slow graphics system yields figures that take a long time to fill, especially if texture patterns are used. Pictures are therefore built up sequentially over long periods of time, possibly several minutes. To get the actual effect of a page, the page making system must use a decoder similar to the end-user's. Thus, slowness and sequentiality is an inherent property of such systems, which cannot make use of the speed of frame buffers, nor the kinds of icon-based menu interfaces that they can support[2,4].

3. User Interface Requirements for Page-Making Systems

Although it is possible to simply establish a one-to-one correspondence between **naplps** codes and page-creation commands, such a system would be so cumbersome as to be unusable. A more appropriate interface should reflect both the characteristics of the structures being used, as well as the forms of human motion and perception. Functionally, it must support three classes of operations: creating a new element, modifying the current picture, and changing the active environment.

The major user-interface design problems lie in the implementation of this informal set of specifications. They are: methods for accepting input, displaying and changing the current active environment, dealing with colours, and editing the current picture. The user interface must integrate the solutions in as clear and simple a manner as possible.

4. PCS-UVIC and Interface Abstractions

The Page Creation System developed by the author at the University of Victoria is based on an IBM Personal Computer, using an external decoder with a graphics monitor and optional input devices. The graphics monitor is treated as the drawing and page display screen, and the IBM monitor as the menu screen. The user-interface is based on a small number of *abstractions*, which present the user with easily understandable objects that are simple to manipulate. The most important of these are: the *cursor*, *virtual buttons*, and the *colour bar*.

4.1. The Cursor Abstraction

The user interacts with the system either through the cursor or by entering text with the keyboard. The cursor is active and displayed on only one of the drawing or menu screens. The functions for moving the cursor, 'accepting' its present position, and switching it between screens are mapped onto either the keyboard or the optional pointing device, which is a mouse or digitizer tablet. Accepting the cursor while in the drawing screen includes the point into the current graphical element, and in the menu area activates its corresponding function.

4.2. The Virtual Button Abstraction

The menu screen [Figure 1] is an object used to display amd change the current active environment, to invoke functions for editing the page, and to support the colour map. It is divided into a number of contiguous *virtual buttons*, each of which has a display mode. A button is 'pressed' by moving the menu cursor there and 'accepting' it. As the cursor moves, the current button is outlined in high intensity.

There are four flavours of buttons: *select* buttons, *toggle* switches, *action* buttons, and *function* buttons. Select buttons allow the user to pick one of a group to set an attribute. For example, the user selects one of the buttons DOT, LINE, ARC, RECTANGLE, to set the current drawing element. The selected button is displayed in reverse video.

A toggle button is shown on the menu screen as a box with <a>/<b> on it, where <a> and <b> are its possible values. Only one of these is active, with its descriptor in high intensity. Pressing the button causes the other attribute to become active.

An action button is similar to a firing trigger, whose action is invoked each time it is selected. Every time an action button is pressed, it flashes in reverse video. A colour can be modified by changing its HSV values using action buttons.

Some functions in PCS-UVIC may cover several steps. For example, to modify the current colour, it can be continuously varied until the user is satisfied. When a function button is pressed, it flashes continuously in high intensity, and the function remains in effect until the button is pressed again.

4.3. The Colour Bar Abstraction

The set of displayable colours are presented as a

series of contiguous rectangles on the drawing screen, with the current colour outlined. The user can select a different colour by invoking the 'New Colour' function, moving the cursor to the colour, and accepting it. The colour bar can be scaled in size, oriented horizontally or vertically, and repositioned, or removed. The system covers its previous position with the background colour, which may render the picture temporarily incorrect, until redrawn. Actions which require the colour bar to be present, such as selecting a new colour, will redraw it on the screen if necessary.

To support the use of only 16 displayed colours out of 4096, the current colour value is displayed on the menu screen in its RGB and HSV coordinates, and can be modified either by entering the exact RGB or HSV values, or step by step using rate buttons. The step size can itself by changed with the menu. What the colour bar represents is a *colour map* from display space (16 possibilities) into value space (4096). The current colour map can be included in the **naplps** encoded picture, to support colour map animation[5]. This also provides a mechanism for the page creator to save and recall previously used colour maps.

4.4. Other Interface Abstractions

Other techniques used in the user-interface include the provision of a keyboard to menu mapping, so that buttons can be selected using keys assigned by a user-definable configuration file. Sequences of such keys-trokes can be remembered as keyboard macros. The system can define a sequence of elements as a *group*, which can be given a name, saved on file, and included in new pictures. The system also has the concept of a background of up to ten previously created pages, which can be used for inheriting colour maps, texture maps, text fonts, as well as a mechanism for showing commonly used visual backdrops.

5. Discussion

The **naplps** environment is by no means a simple one, and many page creation systems require a significant period of time to learn. PCS-UVIC has been used by novices after only 15 minutes of instruction to create non-trivial pictures. An interesting aspect of designing user interfaces is to allow entry level skills to be acquired quickly, while providing mechanisms for the professional user to build short cuts. This is the reason why tree-structured menus were rejected in favour of buttons and configurable keyboards with macros.

REFERENCES

1. Canadian Standards Association. Videotex/Teletext Presentation Level Protocol Syntax. Toronto. 1983.

2. Miller D. Videotex: Science Fiction or Reality? BYTE:42-56. July 1983.

3. Plebon DA, Booth KS. Interactive Picture Creation Systems. Department of Computer Sciene Technical Report CS-82-46. University of Waterloo. 1982.

4. Smith AR. Paint. NYIT Technical Memo. July 1978.

5. Shoup RG. Color Table Animation. Computer Graphics 13(2):-13. Aug. 1979.

6. Tanner P et al. Colour Selection, Swath Brushes and Memory Architectures for Paint Systems. Proc. Graphics Interface 83. Edmonton. 1983.

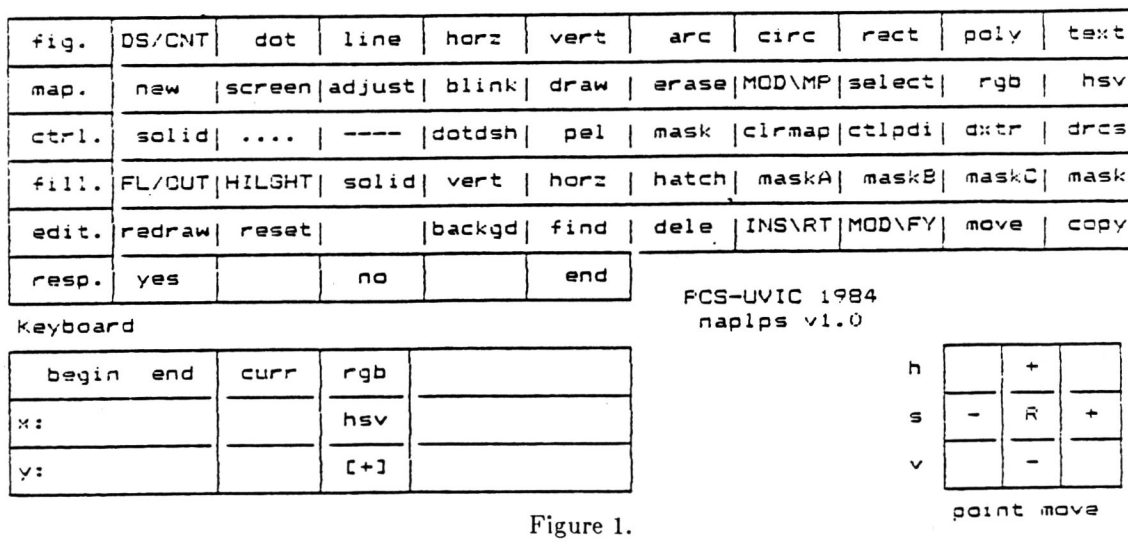

Figure 1.