

Description

This invention relates to electronic systems and, more particularly, to electronic instruments for measuring electrical signals. Specifically, the invention is directed to displays for electronic systems.

In one known instrument, a display is provided which can display as many as four windows. An instrument can comprise a module or a combination of modules. A different instrument state can be displayed in each of the four windows simultaneously. That is, a display of the state of a module or combination of modules can be displayed in each of the four windows concurrently.

Unfortunately, once the different instrument states are displayed, it is very cumbersome to access an individual instrument, that is, a module or combination of modules, to change instrument settings. For example, one exemplary instrument, the HP 70000 series modular measurement system display, the HP 70206A display, available from Hewlett-Packard Company, Signal Analysis Division, Rohnert Park, California, requires numerous keystrokes to change instrument settings. Specifically, the user needs to depress the "DISPLAY" hardkey, then push the "Assign Keyboard" softkey, next select the window corresponding to the instrument whose settings are to be changed by means of a numeric keyboard or a rotary pulse generator knob, and depress the "MENU" hardkey to enable the instrument setting softkeys. This operation necessitates an in-depth understanding of the operation of the instrument. This operation also usurps the time to enter many keystrokes irrespective of the level of skill of the user. It is therefore desirable to facilitate the change of instrument settings in such an instrument.

Additionally, many color editors are known. One approach for enabling a user to select a set of colors is to display multiple colors from which the user can choose on the display screen simultaneously, and the user can select colors by codes associated with the colors. Typically, the system sets defaults if the user does not elect to choose the colors. For example, in the case sixteen colors are available, the background can be defaulted to black (0), the graticule can be defaulted to gray (1), the first trace can be defaulted to yellow (2), the second trace can be defaulted to cyan (blue) (3), the third trace can be defaulted to pink (4), and so on up to 16 colors (i.e., colors 0-15). Therefore, color choices and the ability to adjust colors for various reasons, such as the level of ambient light, color blindness of the user, etc., are limited. It is therefore desirable to provide a broader range of color choices for color displays.

Finally, dot or pixel stretching to achieve a clearer, more refined display is known. Previously, however, pixel stretching has been achieved by means of hardware, such as in the HP 70205 and HP 70206 displays, by ORing adjacent pixels. More recently, pixel stretching has been achieved by means of a PAL (program array logic) circuit. However, in both of these instances, only the pixels of traces of data are stretched. It is desirable that pixel stretching be applied to the entire display.

Previous color editors allowed the user to adjust only the HSL values, as disclosed in "Microsoft Windows Presentation Manager Benutzerhandbuch Version 2," Microsoft Corporation, 1987, or the RGB values. Specifically, this document does not disclose adjustment or display of the red value, green value, or blue value. That is, in the data processing system disclosed in this document, the user is never shown the red value, green value, or blue value.

It is an object of the invention to provide a method for adjusting a plurality of colors which allows full flexibility of color adjustment and teaches the user the relationship between the different color systems.

This object is achieved by a method for adjusting a plurality of colors in accordance with claim 1.

Summary of the Invention

One embodiment of the present invention provides an instrument ("INSTR") hardkey to facilitate a change in settings of any instrument displayed in any of the windows that appears on a display. The user merely depresses the "INSTR" key and selects the instrument whose settings are to be changed by means of a softkey menu that is displayed when the "INSTR" hardkey is depressed. In the case of a color display, different colors can be employed to show the correspondence between the instruments shown in the windows and the softkey menu instrument choices (i.e., the instrument and corresponding softkey menu selections are color coded). Also, or alternatively, the model number of the instruments shown in the windows can be shown in the softkey menu. Both color coding and model number coding are preferred. Furthermore, in systems where multiple modules having the same model number are employed, the row and column addresses of the modules having the same model number are displayed to distinguish between instruments having the same model number.

One embodiment of the invention also enables the user to adjust the colors of the color display. The color editor is preferably resident in the display. An instrument connected to the display preferably labels softkeys for allowing the user to adjust the colors based upon the instrument actually controlling the display, that is, the module or combination of modules controlling the display determines what to call the aspects of the display

which are enabled for color adjustment.

When the user selects color adjustment, the display provides a superimposed window and softkey menu for enabling the user to adjust colors. The menus can provide various selections, including the selection of default colors and preferably colors that compensate for various types of color blindness, which are stored in a read only memory. The user also has the softkey menu option to select "Adjust Color."

The "Adjust Color" softkey allows the user to adjust one or all of six factors: red, green, blue, hue (tint), saturation (color or color level), and luminosity (brightness). The window preferably displays red, green, and blue bars in a bar graph format, which change in height as the user changes the red, green, and/or blue settings by means of a rotary pulse generator or a numeric keyboard. The window also shows readouts for hue, saturation, and luminosity which change as the user adjusts the values with the rotary pulse generator or the numeric keyboard. The ability to adjust and view any or all of the six attributes (red, green, blue, hue, saturation, and/or luminosity) enables the user to select any desired color.

Finally, one embodiment of the invention provides a method implemented in software, instead of hardware, for providing pixel stretching. In accordance with the pixel stretching method of the invention, any pixel stretching can be applied to any item on the entire display to provide clearer, more refined images.

Brief Description of the Drawings

The above and other features of the invention and the concomitant advantages will be better understood and appreciated by persons skilled in the field to which the invention pertains in view of the following description given in conjunction with the accompanying drawings. In the drawings:

Figs. 1-3 show example screens for one, two, and four window cases, respectively, useful in describing the "INSTR" key provided in accordance with one embodiment of the invention;

Fig. 4 shows an example for editing COLOR 0 in accordance with one embodiment of color adjustment method in accordance with the invention;

Fig. 5 shows an example of a softkey menu which indicates that RED is active and RGB selected in connection with adjusting the colors of a color display; and

Fig. 6 illustrates an example of pixel stretching in accordance with the invention.

Detailed Description of the Preferred Embodiments

A key not previously provided, the "INSTR" key, is used to move the keyboard between modules on the screen of a modular measurement system display, such as the HP 70004A display.

Previously, it was necessary to enter a menu called "assign keybd" in which a window could be selected by entering the appropriate window number with a numeric keyboard or a rotary pulse generator knob, then depressing MENU again.

The INSTR key allows two-stroke access to any window displayed on-screen, thus greatly easing the burden of controlling instruments in several different windows.

The "INSTR" key functions as follows. When the "INSTR" key is depressed, the display places colored borders around the currently defined windows. The pens used are: window 1, pen 2; window 2, pen 3; window 3, pen 4; window 4, pen 5. Typically, these pens are yellow, cyan, pink, and green, respectively. In the corner nearest the default position of these windows (window 1, lower left, window 2, lower right, window 3, upper left, window 4, upper right), a softkey containing the first seven characters in the label identification for the module is displayed in inverse video in the same color as the associated window border. If any of the labeled softkeys is depressed, a keyboard link is offered to the module corresponding to that window. If the module accepts the link, it is given the keyboard and will put up its menu keys as though the "MENU" key had been depressed.

Figs. 1-3 show example screens for one, two, and four window cases, respectively.

The listing of the firmware to implement the INSTR key follows:

```

"68000" PREPROCESS

{$Log: instr.P,v $
Revision 6.0.3.1 89/04/19 15:30:40 15:30:40 coryb (Cory Boyan)
5 MR release

Revision 6.0 89/04/19 15:30:05 15:30:05 coryb (Cory Boyan)
Rev 6.0 to create
}

10 PROGRAM instr_P;

$EXTENSIONS ONS
#include define.H
{*****}

15 #include text.H

{*****}

#define FILE_NAME instr_P
20 #include asm_dec.H
#include psc1_dec.H

$GLOBPROC ONS

{*****}
25 { This file handles the processing for the INSTR KEY functions. }
{ It was added as of 7.0 and only pertains to the new hardware. }
{*****}

PROCEDURE COLOR_FRAME(Window: BYTE); (7.0)

30 { This procedure activates and colors the thick frame around a }
{ window (if it exists) and colors it Color. }

VAR Color: SIGNED_16;

35 BEGIN
CASE Window OF
1: Color:= YELLOW;
2: Color:= CYAN;
3: Color:= PINK;
40 4: Color:= TRD_COLOR;
OTHERWISE Color:=0;
END;
SET_VLIST((Frame_pgm.Thickframes[Window-1].Area_pen),Color);
SET_VLIST((Frame_pgm.Thickframes[Window-1].Area_fill1),AREA_FILL);
SET_VLIST((Frame_pgm.Thickframes[Window-1].Area_fill2),AREA_FILL);
45 SET_VLIST((Frame_pgm.Thickframes[Window-1].Area_fill3),AREA_FILL);
SET_VLIST((Frame_pgm.Thickframes[Window-1].Area_fill4),AREA_FILL);
END;

{*****}

50 PROCEDURE SHOW_COLORS; (shows existing windows with colored frames)

VAR Point: POINT_STACK;
1: BYTE;

55 BEGIN

```

```

FOR I:=1 TO 4 DO
  BEGIN
    Point:=FIND_TASK(-1,I);
    IF (Point <> NIL) THEN COLOR_FRAME(I);
  END;
END;

(*****);
(
( Usage of Update variables
(
( Update_value1:           Task we are scanning for
( Update_value3:           Address of task (I/O)
( Update_value7:           character number to be loaded
(
(*****);

PROCEDURE LOAD_INSTR_KEY(Task: SIGNED_16);
(setup the required menu key)

VAR Key:      BYTE;
    Pen_loc: DATA_POINT;
    Color:    SIGNED_16;

BEGIN
CASE Task OF
  1 : BEGIN
    Key := 28;    (bottom left)
    Pen_loc.Value:=INTEGER(ADDR(SYSKEY1))+2;
    Color:= YELLOW;
    END;
  2 : BEGIN
    Key := 21;    (bottom right)
    Pen_loc.Value:=INTEGER(ADDR(SYSKEY8))+2;
    Color:= CYAN;
    END;
  3 : BEGIN
    Key := 22;    (top left)
    Pen_loc.Value:=INTEGER(ADDR(SYSKEY1))+2;
    Color:= PINK;
    END;
  4 : BEGIN
    Key := 15;    (top right)
    Pen_loc.Value:=INTEGER(ADDR(SYSKEY8))+2;
    Color:= TRD_COLOR;
    END;
    OTHERWISE Key:=0;
    END;
IF (Key<>0) THEN
  BEGIN
    OPEN_KEY(Key,Instrkey_data);           ( "Open" the key. )
    Instrkey_data.Enhancement:=INVERSE;
    Pen_loc.Value:=Pen_loc.Value+KEY_OVER+KEY_PEN+
      INTEGER(KEY_LENGTH*(SIGNED_16(Key-1) MOD 7));
    SET_VLIST(Pen_loc.Point_word^,Color);  ( Color the key )
    END;
  END; (LOAD_INSTR_KEY)

(*****);

PROCEDURE INSTR_SETUP;

( This procedure sets up the menu keys and begins the scanning )
( of addresses for the INSTR KEY function. It defines which )

```

```

      ( routines handle NMAA and immediate data. )

VAR Pointer:    POINT_STACK;
    Flag, Key:  BYTE;

5
BEGIN
    Flag:=DISABLE_MENU;                ( Disable menus & RPG )

    FOR Key:=15 TO 28 DO                ( Blank the softkeys )
        BEGIN
10      OPEN_KEY(Key,Instrkey_data);    ( "Open" the key. )
        CLOSE_KEY(Instrkey_data);      ( "Close" the key. )
        END;

    Update_value1:=0;                  (find first task )

15  REPEAT
    Update_value1:=Update_value1+1;
    Pointer:=FIND_TASK(-1,Update_value1);
    IF (Pointer<>NIL) THEN
        BEGIN
20      Update_value3:=Pointer^.Channel;
        IF (Update_value3=UNUSED) THEN  (window unassigned as yet )
            Pointer:=NIL                (get another task )
        ELSE IF (Update_value3=HP1B) THEN (treat HP-1B special )
            BEGIN
25      LOAD_INSTR_KEY(Update_value1);
        PRINT_STRING(Instrkey_data,' HP-1B ');
        Pointer:=NIL;                  (get another task )
            END;
        END;
    UNTIL (Update_value1=4) OR (Pointer<>NIL);

30  IF (Pointer<>NIL) THEN
    BEGIN
        LOAD_INSTR_KEY(Update_value1);    (prepare approp. label )
        Nmaa_routine:=IMAP_TABLE[0];      ( Define execution )
        Immed_routine:=IMAP_TABLE[1];      ( routines. )
        Update_value7:=1;                  ( Process chars. )
35      SEND_COMMAND(Update_value3,MS1B_1D); ( Start the riot. )
        END;
    ELSE
        BEGIN
40      Nmaa_routine:=0;                    ( Terminate mode. )
        Immed_routine:=0;
        END;

    ENABLE_MENU(Flag);                  ( Restore menu & RPG )
END;

45  (*****);

PROCEDURE INSTR_NMAA(Address: SIGNED_16);

    ( This procedure handles the case where a window exists with no module )

50  VAR Flag:    BYTE;
    Pointer:    POINT_STACK;

    BEGIN
        IF (Address = Update_value3) THEN    ( If from current addr )
55      BEGIN
          Flag:=DISABLE_MENU;                ( Disable keys & RPG )

```

```

Pointer:=NIL;

IF (Update_value1<4) THEN                                ( more tasks )
  REPEAT
5    Update_value1:=Update_value1+1;
    Pointer:=FIND_TASK(-1,Update_value1);
    IF (Pointer<>NIL) THEN
      BEGIN
        Update_value3:=Pointer^.Channel;
        IF (Update_value3=UNUSED) THEN                (window unassigned as yet )
          Pointer:=NIL                                (get another task )
        ELSE IF (Update_value3=HP1B) THEN              (treat HP-1B special )
          BEGIN
            LOAD_INSTR_KEY(Update_value1);
            PRINT_STRING(Instrkey_data,' HP-1B ');
            Pointer:=NIL;                              (get another task )
          END;
        END;
      UNTIL (Update_value1=4) OR (Pointer<>NIL);

  IF (Pointer<>NIL) THEN
20    BEGIN
      LOAD_INSTR_KEY(Update_value1);                  ( Prepare key label )
      Update_value7:=1;                               ( Process chars. )
      SEND_COMMAND(Update_value3,MS1B_ID);            ( Start the next riot. )
    END
  ELSE
25    BEGIN
      Nmaa_routine:=0;                                ( Terminate mode. )
      Immed_routine:=0;
    END;

  ENABLE_MENU(Flag);                                ( Re-enable keys & RPG )
30  END;
END;

(*****);

35  PROCEDURE INSTR_DATA(Address,Data: SIGNED_16);

  ( This routine processes data for the INSTR KEY function. )

  VAR Character: WORD;
      Flag, I: BYTE;

40  BEGIN
    IF (Address = Update_value3) THEN                ( If from current addr.)
      IF (Data = END_DATA) THEN
        INSTR_NMAA(Address)                          ( End immediate: Finish)
      ELSE
45    BEGIN
        BEGIN                                         ( Else display data. )
          Flag:=DISABLE_MENU;                        ( Disable keys & RPG )

          Character.Word:=Data;
          IF (Update_value7<8) THEN                  ( If processing chars )
            IF (Character.Low_byte=BYTE(",")) THEN    ( Comma is end of line.)
50              BEGIN
                FOR I:=Update_value7 TO 7 DO
                  BEGIN
                    Instrkey_data.Character:= ' ';
                    PRINT_CHAR(Instrkey_data);        ( Add to label )
                  END;
                Update_value7:=8                      ( Process no more chars)
              END
            END
          END
        END
      END
    END
  END

```

```

ELSE BEGIN
  Instrkey_data.Character:=CHR(Character.Low_byte);
  PRINT_CHAR(Instrkey_data);          ( Add to label      )
  Update_value7:=Update_value7+1;
  END;
5
  ENABLE_MENU(flag);                  ( Re-enable keys & RPG )
  END;
  {*****}
10

```

The "Adjust Color" menu is used to modify the colors displayed by the HP 70004A display. The "Adjust Color" menu is:

- * RECALL COLORS
- 15 * SAVE COLORS
- * edit colors
- *
- * DEFAULT COLORS
- * MONOCHROME
- 20 * special colors

RECALL COLORS and SAVE COLORS load and store the current color map from/to any of four color map registers. The user is prompted to enter a register number when one of these keys is depressed and concludes the entry with the "ENTER" softkey.

- "DEFAULT COLORS" loads the current color map from a preset color map in read only memory (ROM).
- 25 "MONOCHROME" loads the current color map with shades of green. The "special colors" menu loads predefined color maps out of ROM. The three choices are "VISION ENHNC1," "VISION ENHNC 2," and "OPTICAL FILTER," which are color maps optimized for two common visual impairments and for people working in red goggles often found in optics laboratories.

- The "edit colors" menu allows editing of pens 0-11. This editing affects the current color map, regardless of how it was originally loaded. For example, the MONOCHROME palette can be loaded, and then one or more of the trace colors can be modified using "edit colors."
- 30

The "edit colors" menu labels look like:

- * BACK GROUND
- * COLOR 1
- 35 * COLOR 2
- * COLOR 3
- * COLOR 4
- * COLOR 5
- * MORE 1 of 2
- 40 * COLOR 6
- * COLOR 7
- * COLOR 8
- * COLOR 9
- * COLOR 10
- 45 * KEY LABELS
- * MORE 2 of 2

When any of these colors is selected, the following menu goes up:

- * HUE
- * SATURATION
- 50 * LUMINOSITY
- * HSL/RGB
- * UNDO
- * PLOTTER PEN
- * prev menu

- 55 An example, shown for editing COLOR 0, is shown in Fig. 4.

The active function (HUE, SATURATION, or LUMINOSITY) is underlined. HUE is active on entry to this menu. HSL is underlined as well. If the HSL/RGB key is depressed, the menu changes to:

- * RED

- * GREEN
- * BLUE
- * HSL/RGB
- * UNDO
- 5 * PLOTTER PEN
- * prev menu

with RED active and RGB underlined. An example is shown in Fig. 5.

10 In either case a box appears on the display screen which overrides a part of the on-screen data. In the box is a patch of the color being edited, and also red, green, and blue bars showing graphically the red, green, and blue content of the color. This is significant because the hardware implements the RGB model, and hence the bars show the exact hardware settings (and hence imply the parameters for the PC command) for that color. The values of HUE, SATURATION, LUMINOSITY, RED, GREEN, and BLUE are also displayed textually. One or more of these values changes when the rotary pulse generator knob is turned or a number entered.

15 Previous color editors allowed the user to adjust only the HSL values or the RGB values. The color editor in accordance with the invention allows adjustment of any of the six attributes of a color (RED value, GREEN value, BLUE value, HUE, SATURATION, or LUMINOSITY) and simultaneous display of all six attributes, along with a graphical readout of RGB. This allows the user to truly understand exactly how the color mix is being impacted by any changes being made to one of the six attributes.

20 The UNDO key restores the values for the current pen that existed when that pen was selected to edit. However, if the "prev menu" key is depressed (thus returning to the menu with the pen selections), the newly edited values are "latched" into that pen.

The algorithm for converting RED/GREEN/BLUE (RGB) colors to/from HUE/SATURATION/LUMINOSITY (HSL) colors is shown in the respective RGB_TO_HSL and HSL_TO_RGB routines in the listing that follows.

25

30

35

40

45

50

55

```

"68000" PREPROCESS

($Log: color.P,v $
Revision 6.0.3.1 89/04/19 15:30:37 15:30:37 coryb (Cory Boyan)
5 MR release

Revision 6.0 89/04/19 15:29:40 15:29:40 coryb (Cory Boyan)
Rev 6.0 to create
)

10 PROGRAM color_P;

$EXTENSIONS ON$
#include define.H
(*****);

15 #include text.H

COLED_PATCH,COLED_RED,COLED_GREEN,COLED_BLUE,COLED_NUMBER,
COLED_NAME,COLED_REDT,COLED_GREENT,COLED_BLUET,COLED_HUET,
COLED_SATT,COLED_LUNT,COLED_PROGRAM : SIGNED_16;

20 HUE_MSG: LONG_LANGUAGE;

(*****);

$GLOBVAR ON$

25 VAR
Old_red, Old_green, Old_blue : SIGNED_16;

#define FILE_NAME color_P
#include asm_dec.H
30 #include pscl_dec.H

$GLOBPROC ON$

( ***** NOTES ON H S & L ***** )
35 ( Hue changes only 1 color between max & min determined by Saturation )
( and Luminosity while others stay at levels determined by S & L. )
( Luminosity determines the max any color can get; if Luminosity is )
( 100%, at least one color is at max. )
( Saturation determines the minimum any color can get; if Saturation is )
( 100%, there is no white, which means at least one color is off. )
40 ( Hence a common scenario is to adjust hue with Sat and Lum both 100%, )
( in which case one color is at 0, one is full on, and the other goes )
( from 0 to 100 as hue changes; when it gets to 100, the color already )
( at 100 starts toward 0, and so on. Only one changes at a time while )
( one of the others is full on and the other is full off. )

45 (*****);
(
( Usage of Update variables )
(
( Update_point1: )
50 ( Update_point2: )
( Update_point3: )
( Update_point4: )
( Update_point5: )

```

```

( Update_point6:                                )
(                                                )
( Update_value1:                                Pen being edited )
( Update_value2:                                Prev value storage )
5 ( Update_value3:                                Red )
  ( Update_value4:                                Green )
  ( Update_value5:                                Blue )
  ( Update_value6:                                Hue )
  ( Update_value7:                                Saturation )
  ( Update_value8:                                Luminosity )
10 ( )
  (*****)

```

(The USER_PALETTE is a 0..15 array of SIGNED_16, each entry of which contains:

15

Attr	Red	Green	Blue
4	4	4	4

20 Red, Green, and Blue go between 0 and 15.
 Hue, Saturation and Luminosity go between 0 and 100.)

(*****)

25 PROCEDURE HSL_TO_RGB; {derives the RGB values from the HSL}

VAR Sextant, Low_color, High_color, Frac, Q, T : SIGNED_16;

```

BEGIN
  IF (Update_value7 = 0) THEN { ignore Hue, no saturation }
    BEGIN { set colors = Luminosity }
30   {add 50 to round}
      Update_value3:=((15*Update_value8)+50)/100; {red}
      Update_value4:=Update_value3; {green}
      Update_value5:=Update_value3; {blue}
    END
  ELSE BEGIN
35   IF Update_value6>50 THEN Sextant:=(Update_value6+1)/17
      ELSE Sextant:=(Update_value6)/17; {determine sextant from Hue:
                                         0-16 0    50-66 3
                                         17-33 1    67-83 4
                                         34-49 2    84-99 5
                                         }
40   Frac:=(Update_value6*6)-(Sextant*100); {Frac = how far into sextant
                                             Hue is, *100. Eg, if it is
                                             halfway in, Frac = 50. }

```

45 {The color (R,G,or B) at the lowest level is the product of Luminosity and 1-Saturation. Normally these values go from 0-1, but here they go 0-100. Hence the result is x 10000. But since the result normally goes 0-1 but in our case goes 0-15, it is really only x 10000/15 = 666.67. We add 333 to effect rounding: }

50 Low_color:=((Update_value8*(100-Update_value7))+333)/667;

{The color at the highest level is the same as Luminosity:}

High_color:=((15*Update_value8)+50)/100; {add 50 to round}

55 { Q and T, normally values between 0 and 1, are determined by the following formulae:

Q=Luminosity*(1-(Saturation*Frac))

```

5          T:=Luminosity*(1-(Saturation*(1-Frac)))
      Because of the complexity of these formulae only the desired
      one is executed for any given color:

      O:=(Update_value8*((10000-(Update_value7*Frac))/100))/667;
      T:=(Update_value8*((10000-(Update_value7*(100-Frac))/100))/667; )

10  ( The descriptions below apply for the fully saturated 100% Luminosity case: )

      CASE Sextant OF
          0: BEGIN ( Full red, increasing green, no blue )
              (R) Update_value3:=High_color;
              (G) Update_value4:=((Update_value8*((10000-(Update_value7*(100-Frac))/100))+333)/667; (T)
              (B) Update_value5:=Low_color;
              END;
          15  1: BEGIN ( Full green, decreasing red, no blue )
              (R) Update_value3:=((Update_value8*((10000-(Update_value7*Frac))/100))+333)/667; (O)
              (G) Update_value4:=High_color;
              (B) Update_value5:=Low_color;
              END;
          20  2: BEGIN ( Full green, increasing blue, no red )
              (R) Update_value3:=Low_color;
              (G) Update_value4:=High_color;
              (B) Update_value5:=((Update_value8*((10000-(Update_value7*(100-Frac))/100))+333)/667; (T)
              END;
          25  3: BEGIN ( Full blue, decreasing green, no red )
              (R) Update_value3:=Low_color;
              (G) Update_value4:=((Update_value8*((10000-(Update_value7*Frac))/100))+333)/667; (O)
              (B) Update_value5:=High_color;
              END;
          30  4: BEGIN ( Full blue, increasing red, no green )
              (R) Update_value3:=((Update_value8*((10000-(Update_value7*(100-Frac))/100))+333)/667; (T)
              (G) Update_value4:=Low_color;
              (B) Update_value5:=High_color;
              END;
          5: BEGIN ( Full red, decreasing blue, no green )
              (R) Update_value3:=High_color;
              (G) Update_value4:=Low_color;
              (B) Update_value5:=((Update_value8*((10000-(Update_value7*Frac))/100))+333)/667; (O)
              END;
          OTHERWISE;
          END; (CASE)
      END; (IF (Update_value7 = 0))
35  END;

      (*****)

      PROCEDURE RGB_TO_HSL; (derives the HSL values from the RGB)

      VAR Max, Min, Diff : SIGNED_16;

40  BEGIN
      IF (Update_value3>Update_value4) AND (Update_value3>Update_value5)
      THEN Max:=Update_value3 (red)
      ELSE IF (Update_value4>Update_value5) AND (Update_value4>Update_value3)
      THEN Max:=Update_value4 (green)
      ELSE IF (Update_value5>Update_value4) AND (Update_value5>Update_value3)
45  THEN Max:=Update_value5; (blue)

      IF (Update_value3<Update_value4) AND (Update_value3<Update_value5)
      THEN Min:=Update_value3 (red)
      ELSE IF (Update_value4<Update_value5) AND (Update_value4<Update_value3)
      THEN Min:=Update_value4 (green)
      ELSE IF (Update_value5<Update_value4) AND (Update_value5<Update_value3)
50  THEN Min:=Update_value5; (blue)

```

55

```

      THEN Min:=Update_value5;      (blue)

      Diff:=Max-Min;                ( Difference will be used later      )
      Update_value8:=((100*Max)+7)/15; ( Luminosity is equal to the brightest color.)
5      Update_value8:=((100*Max)+7)/15; ( (7 is added to round.) )
      IF (Max<>0) THEN                ( Saturation is the difference over the max )
      Update_value7:=((100*Diff)+(Max/2))/Max ( Max/2 added to round. )
      ELSE
      Update_value7:=0;

10     IF (Update_value7<>0) THEN      ( Don't change Hue if Saturation is 0 )
      BEGIN                          ( 3 is added to round. )
      IF Update_value3=Max THEN      ( Max RED )
      Update_value6:=(((Update_value4-Update_value5)*100)/Diff)+3)/6
      ELSE IF Update_value4=Max THEN ( Max GREEN )
      Update_value6:=((200+((Update_value5-Update_value3)*100)/Diff)+3)/6
15     ELSE IF Update_value5=Max THEN ( Max BLUE )
      Update_value6:=((400+((Update_value3-Update_value4)*100)/Diff)+3)/6;
      IF Update_value6<0 THEN Update_value6:=100+Update_value6;
      IF Update_value6>99 THEN Update_value6:=100-Update_value6;
      END;
      END;

20     (*****

PROCEDURE UPDATE_READOUT(Value,Digits: SIGNED_16; Where: POINT_WORD);
VAR  I:      SIGNED_16;
      Temp:   STRING;
25     Point:  POINT_WORD;
      BEGIN
      CONVERT_TO_TEN(Temp,Value,Digits);
      Point:=Where;
      FOR I:=1 TO Digits DO
      BEGIN
30         SET_VLIST(Point^,SIGNED_16(UNSIGNED_8(Temp[I])));
      Point:=POINT_WORD(INTEGER(Point)+2);
      END;
      END;

      (*****

35     PROCEDURE UPDATE_COLED;
      BEGIN
      SET_VLIST(COLED_RED,Update_value3*3);
      SET_VLIST(COLED_GREEN,Update_value4*3);
      SET_VLIST(COLED_BLUE,Update_value5*3);
40     UPDATE_READOUT(Update_value3,3,ADDR(COLED_REDT));
      UPDATE_READOUT(Update_value4,3,ADDR(COLED_GREENT));
      UPDATE_READOUT(Update_value5,3,ADDR(COLED_BLUET));
      UPDATE_READOUT(Update_value6,4,ADDR(COLED_HUET));
      UPDATE_READOUT(Update_value7,4,ADDR(COLED_SATT));
      UPDATE_READOUT(Update_value8,4,ADDR(COLED_LUMT));
45     USER_PALETTE[Update_value1]:=256*Update_value3 (red)
      +16*Update_value4 (green)
      +Update_value5;(blue)
      New_checks:=CHECKSUM(ADDR(USER_PALETTE),NEW_SIZE);
      SET_PALETTE(ADDR(USER_PALETTE));
      END;

50     (*****

PROCEDURE OLD_ONES; (restores old colors)
      BEGIN
      Update_value3:=Old_red;
55

```

```

Update_value4:=Old_green;
Update_value5:=Old_blue;
Update_value6:=0; {Initialize Hue to 0 in case Saturation is 0}
5  RGB_TO_HSL;      {Derive H,S,L}
  UPDATE_COLED;
  POKE_KEY(SOFTKEY+1);      {push RED or HUE      }
  END;

{*****}

10  PROCEDURE LOAD_MONO;
    BEGIN
      IF GET_VLIST(Update_pgm)=BS_COLED_FLAG THEN
        SET_VLIST(Update_pgm,RETURN);      { Disable the pgm }
      USER_PALETTE:=MONO_PALETTE;
15  New_checks:=CHECKSUM(ADDR(USER_PALETTE),NEW_SIZE);
      SET_PALETTE(ADDR(USER_PALETTE));
      END;

{*****}

20  PROCEDURE LOAD_DEFCOLOR;
    BEGIN
      IF GET_VLIST(Update_pgm)=BS_COLED_FLAG THEN
        SET_VLIST(Update_pgm,RETURN);      { Disable the pgm }
      USER_PALETTE:=DEFAULT_PALETTE;
25  New_checks:=CHECKSUM(ADDR(USER_PALETTE),NEW_SIZE);
      SET_PALETTE(ADDR(USER_PALETTE));
      END;

{*****}

30  PROCEDURE COLED_SETUP;
    VAR Pointer,Temp:      DATA_POINT;
    BEGIN
      {Fetch R,G,B}
35  Update_value3:=SIGNED_16(BINARY_WORD(SHIFT(USER_PALETTE[Update_value1],-8))
      *BINARY_WORD(OF00H));
      Update_value4:=SIGNED_16(BINARY_WORD(SHIFT(USER_PALETTE[Update_value1],-4))
      *BINARY_WORD(OF00H));
      Update_value5:=SIGNED_16(BINARY_WORD(USER_PALETTE[Update_value1])
40  *BINARY_WORD(OF00H));

      Old_red:=Update_value3; {save old colors for UNDO}
      Old_green:=Update_value4;
      Old_blue:=Update_value5;

45  Update_value6:=0; {Initialize Hue to 0 in case Saturation is 0}
      RGB_TO_HSL;      {Derive H,S,L}

      Pointer:=ROUTINE_BS(DATA_POINT(ADDR(COLED_PROGRAM)), {load color editor}
      DATA_POINT(ADDR(Update_pgm))); {dot gen pgm      }

50  SET_VLIST(COLED_PATCH,Update_value1); {Set patch color }
      UPDATE_READOUT(Update_value1,3,ADDR(COLED_NUMBER));

      UPDATE_COLED;      { set up initial values }
      SET_VLIST(Update_pgm,BS_COLED_FLAG); { Enable the pgm }

55  END;

```

```

(*****)
PROCEDURE COLOR_EDITOR;
5  ( This procedure is called via the DISPLAY menu. It      )
  ( starts the process of editing the selected color.      )

VAR Flag:      BYTE;

10 BEGIN
   Flag:=DISABLE_MENU;           ( Disable menu during )
                                  ( processing           )
   COLED_SETUP;                  ( Build the B.S. prog. )

   POKE_KEY(SOFTKEY+1);          ( Enable HUE adjust  )

15  ENABLE_MENU(Flag);           ( Re-enable menus    )
END;

(*****)

20  PROCEDURE ADJ_COLOR0;
    BEGIN
      Update_value1:=0;
      COLOR_EDITOR;
    END;

25  PROCEDURE ADJ_COLOR1;
    BEGIN
      Update_value1:=1;
      COLOR_EDITOR;
    END;

30  PROCEDURE ADJ_COLOR2;
    BEGIN
      Update_value1:=2;
      COLOR_EDITOR;
    END;

35  PROCEDURE ADJ_COLOR3;
    BEGIN
      Update_value1:=3;
      COLOR_EDITOR;
    END;

40  PROCEDURE ADJ_COLOR4;
    BEGIN
      Update_value1:=4;
      COLOR_EDITOR;
    END;

45  PROCEDURE ADJ_COLOR5;
    BEGIN
      Update_value1:=5;
      COLOR_EDITOR;
    END;

50  PROCEDURE ADJ_COLOR6;
    BEGIN
      Update_value1:=6;
      COLOR_EDITOR;
    END;

55  PROCEDURE ADJ_COLOR7;

```

```

BEGIN
  Update_value1:=7;
  COLOR_EDITOR;
END;

5  PROCEDURE ADJ_COLOR8;
  BEGIN
    Update_value1:=8;
    COLOR_EDITOR;
  END;

10  PROCEDURE ADJ_COLOR9;
  BEGIN
    Update_value1:=9;
    COLOR_EDITOR;
  END;

15  PROCEDURE ADJ_COLOR10;
  BEGIN
    Update_value1:=10;
    COLOR_EDITOR;
  END;

20  PROCEDURE ADJ_COLOR11;
  BEGIN
    Update_value1:=11;
    COLOR_EDITOR;
  END;

25  (*****);

  PROCEDURE RED_ADJUST;

30  BEGIN
    Update_value2:=Update_value3;
    CASE Knob_mode OF
      ( save old )
      ( Adjust variable )

      KNOB_SET:
        Update_value3:=Knob_variable;

35      KNOB_UPDATE:
        BEGIN
          IF (Intr_count<0) THEN
            Update_value3:=Update_value3-1;
          IF (Intr_count>0) THEN
            Update_value3:=Update_value3+1;
          END;

          OTHERWISE;
          END; (CASE)

45      IF Update_value3<0 THEN Update_value3:=0;
      IF Update_value3>15 THEN Update_value3:=15;

      Intr_count:=0;

50      IF Update_value3<>Update_value2 THEN ( act on change )
        BEGIN
          RGB_TO_HSL; ( Derive HSL values from RGB values )
          UPDATE_COLED; ( Update color editor screen )
        END;

55      Knob_variable:=SIGNED_16(Update_value3); ( Put Update_value3 in )
      ( variable for cmd. )

```

```

                                ( line use.          )
END;

(*****)
5
PROCEDURE GREEN_ADJUST;

BEGIN
    Update_value2:=Update_value4;           { save old          }
    CASE Knob_mode Of                     { Adjust variable    }
10
        KNOB_SET:
            Update_value4:=Knob_variable;

        KNOB_UPDATE:
            BEGIN
15
                IF (Intr_count<0) THEN
                    Update_value4:=Update_value4-1;
                IF (Intr_count>0) THEN
                    Update_value4:=Update_value4+1;
                END;

20
            OTHERWISE;
            END; {CASE}

            IF Update_value4<0 THEN Update_value4:=0;
            IF Update_value4>15 THEN Update_value4:=15;
25
            Intr_count:=0;

            IF Update_value4<>Update_value2 THEN      { act on change }
            BEGIN
                RGB_TO_HSL;           { Derive HSL values from RGB values }
30
                UPDATE_COLED;         { Update color editor screen   }
                END;

                Knob_variable:=SIGNED_16(Update_value4); { Put Update_value4 in }
                                                         { variable:for cmd.    }
                                                         { line use.          }
35
            END;

(*****)

PROCEDURE BLUE_ADJUST;
40
BEGIN
    Update_value2:=Update_value5;           { save old          }
    CASE Knob_mode Of                     { Adjust variable    }

        KNOB_SET:
45
            Update_value5:=Knob_variable;

        KNOB_UPDATE:
            BEGIN
                IF (Intr_count<0) THEN
                    Update_value5:=Update_value5-1;
50
                IF (Intr_count>0) THEN
                    Update_value5:=Update_value5+1;
                END;

            OTHERWISE;
            END; {CASE}
55

            IF Update_value5<0 THEN Update_value5:=0;

```

```

      IF Update_value5>15 THEN Update_value5:=15;

      Intr_count:=0;

5      IF Update_value5<>Update_value2 THEN      ( act on change )
      BEGIN
        RGB_TO_HSL;      ( Derive HSL values from RGB values )
        UPDATE_COLED;      ( Update color editor screen      )
        END;

10     Knob_variable:=SIGNED_16(Update_value5);      ( Put Update_value5 in )
                                                    ( variable for cmd.      )
                                                    ( line use.              )

      END;

15     (*****);

      PROCEDURE HUE_ADJUST;

      BEGIN
        IF (Update_value7=0) THEN
17         BEGIN
          PRINT_STRING(Private_line,HUE_MSG[Language]^);
          Current_digits:=0;      (disable SHOW_ENTRY)
          END

        ELSE BEGIN
25         Update_value2:=Update_value6;      ( save old      )
          CASE Knob_mode OF      ( Adjust variable      )

            KNOB_SET:
              Update_value6:=Knob_variable;

30             KNOB_UPDATE:
              BEGIN
                IF (Intr_count<0) THEN
                  Update_value6:=Update_value6-1;
                IF (Intr_count>0) THEN
35                 Update_value6:=Update_value6+1;
              END;

              OTHERWISE;
              END; (CASE)

40             (HUE wraps around:)
              IF Update_value6<0 THEN Update_value6:=100+Update_value6;
              IF Update_value6>99 THEN Update_value6:=100-Update_value6;

              Intr_count:=0;

45             IF Update_value6<>Update_value2 THEN      ( act on change )
              BEGIN
                HSL_TO_RGB;      ( Derive RGB values from HSL values )
                UPDATE_COLED;      ( Update color editor screen      )
                END;

50             Knob_variable:=SIGNED_16(Update_value6);      ( Put Update_value6 in )
                                                            ( variable for cmd.      )
                                                            ( line use.              )

              END;
            END;

55     (*****);

```

```

PROCEDURE SAT_ADJUST;
BEGIN
  Update_value2:=Update_value7;           ( save old           )
  CASE Knob_mode OF                       ( Adjust variable      )
    5
      KNOB_SET:
        Update_value7:=Knob_variable;

      KNOB_UPDATE:
        BEGIN
          IF (Intr_count<0) THEN
            Update_value7:=Update_value7-1;
          IF (Intr_count>0) THEN
            Update_value7:=Update_value7+1;
          END;
        10
        OTHERWISE;
        END; (CASE)

      IF Update_value7<0 THEN Update_value7:=0;
      IF Update_value7>100 THEN Update_value7:=100;
        20

      Intr_count:=0;

      IF Update_value7<>Update_value2 THEN      ( act on change )
        BEGIN
          HSL_TO_RGB;      ( Derive RGB values from HSL values )
          UPDATE_COLED;    ( Update color editor screen      )
          END;
        25

      Knob_variable:=SIGNED_16(Update_value7);  ( Put Update_value7 in )
                                                ( variable for cmd.   )
                                                ( line use.          )
        30
    END;

    (*****)

  PROCEDURE LUM_ADJUST;
    35
  BEGIN
    Update_value2:=Update_value8;           ( save old           )
    CASE Knob_mode OF                       ( Adjust variable      )
      KNOB_SET:
        40
        Update_value8:=Knob_variable;

      KNOB_UPDATE:
        BEGIN
          IF (Intr_count<0) THEN
            Update_value8:=Update_value8-1;
          IF (Intr_count>0) THEN
            Update_value8:=Update_value8+1;
          END;
        45
        OTHERWISE;
        END; (CASE)
        50

      IF Update_value8<0 THEN Update_value8:=0;
      IF Update_value8>100 THEN Update_value8:=100;

      Intr_count:=0;
        55

      IF Update_value8<>Update_value2 THEN      ( act on change )

```

```

      BEGIN
      HSL_TO_RGB;          ( Derive RGB values from HSL values )
      UPDATE_COLED;        ( Update color editor screen          )
      END;
5
      Knob_variable:=SIGNED_16(Update_value8); ( Put Update_value8 in )
                                              ( variable for cmd.      )
                                              ( line use.                )

      END;
10
      {*****}

      PROCEDURE PEN_COLOR;

      { PC Pen,Red,Green,Blue }
      {
      { This routine implements the HPGL 'PC' command. All the parms
      { are optional and default to 1,existing,existing,existing. }
      {
      { PC -1 causes the USER_PALETTE to get loaded into the task's
      { Color_map palette. }
      {
      { The actual color map hardware is only output to if the task
      { sending PC has a keyboard or control link. }
20
      VAR Lred,Lgreen,Lblue,Lpen: SIGNED_16;

      BEGIN
      WITH Task_point^ DO
      BEGIN
25
          Lpen :=DEFAULT_WORD(1,GET_PARM_WORD);

          IF (Lpen = -1) THEN (restore user's palette)
              Color_map := USER_PALETTE

          ELSE BEGIN

30
              Lred :=SIGNED_16(BINARY_WORD(SHIFT(Color_map[Lpen],-8))*BITS_0to3);
              Lgreen:=SIGNED_16(BINARY_WORD(SHIFT(Color_map[Lpen],-4))*BITS_0to3);
              Lblue :=SIGNED_16(BINARY_WORD(Color_map[Lpen])*BITS_0to3);

              IF (TERMINATOR(Task_point^.Input_term) <> True) THEN
35
                  BEGIN
                  Lred :=DEFAULT_WORD(Lred,SHIFT(GET_PARM_WORD,-4));
                  IF (TERMINATOR(Task_point^.Input_term) <> True) THEN
                      BEGIN
                          Lgreen :=DEFAULT_WORD(Lgreen,SHIFT(GET_PARM_WORD,-4));
                          IF (TERMINATOR(Task_point^.Input_term) <> True) THEN
40
                              Lblue :=DEFAULT_WORD(Lblue,SHIFT(GET_PARM_WORD,-4));
                              END; {IF}
                          END; {IF}

                          Color_map[Lpen]:= 256*Lred+16*Lgreen+Lblue;
                          END; {IF Lpen = -1}

45
                          (If this task has a keyboard or control link then)
                          SET_PALETTE(ADDR(Color_map)); (send new values to GSP (NOP if old hardware) )
                          (also do whenever keyboard or control link established)
                          (and reset hardware to USER if task w/ keyboard is purged)
                          (in fact modify USER_PALETTE also if control link )

50
                          EOL_CHECK;
                          END;

      END;

55

```

The graphics hardware in the HP 70004A display comprises a 12.5 MHz Motorola 68000 microprocessor which handles all I/O, graphics scaling, and stroke generation functions, and a 50 MHz TMS34010 Graphics System Processor (GSP) which handles the raster conversion of vector information into colored dots on the

screen.

The two processors run asynchronously, with the GSP constantly parsing the vector list and writing dots into the bit map. To avoid any artifacts produced by viewing data on the screen while the GSP is updating it, two separate bit maps are used, one of which is viewed while the other is drawn. They are swapped each time the GSP finishes one complete scan of the vector list.

The resolution is 1024 X 400 dots. This is unique in that graphics systems have aspect ratios near 4 x 3; here we have an aspect ratio near 8 x 3. Therefore, this system has effectively double the typical horizontal resolution. This allows dot positioning to twice as fine a degree as is typical, which allows the display of twice as many points of data on the horizontal axis with no loss of resolution. However, given the approximately 147 x 105 mm display area, each pixel is essentially .14 mm wide x .26 mm high, or approximately twice as high as wide. It is desirable for pixels to be square, as this provides for smoother vertical transitions, and equalizes the brightness of vertical and horizontal lines. Therefore, all dots are stretched to be two pixels wide. Fig. 6 shows how this helps make traces overlap and thus smooths the transitions.

Dot or pixel stretching in accordance with the invention is implemented by drawing every line twice. The second time that the line is drawn, the start and the end points are moved one pixel to the right. This works regardless of the orientation of the line.

A listing of the line drawing routine, showing dot stretching, appears below. The language is Texas Instruments TMS34010 assembly language.

```

draw_line:
    move    CURXY,DADDR      ; starting point of line
    move    NEWXY,SADDR      ; ending point of line
    clr     DYDX              ; [0,0]
    subxy   DADDR,SADDR      ; SADDR = [b,a] = end-start
    jrzc    horiz            ; IF DeltaY == 0, Horizontal
    jrn     vert              ; IF DeltaX == 0, Vertical
    subb    INC1,INC1         ; fast way to set all bits = to
                                ; carry from above. INC1 now
                                ; = [-1,-1] if b<0 or [0,0] if b>=0
                                ; Will be used later.
                                ; assume a<0 and b<0, fix later if not
    movk     1,COUNT          ; [-b,-a]
    subxy    SADDR,DYDX       ; branch if -b<0
    jrnc     line1            ; -b>=0, DYDX=[b,-a]
    movy     SADDR,DYDX       ; [-1,-1]
    not      INC1             ; [+1,-1]
    srl      15,INC1          ; DYDX MSW is now positive, INC1 MSW set
                                ; branch if -a<0
                                ; -a>=0, DYDX=[b,a]
                                ; [+1,+1]
    line1:   jrnv             line2
    movx     SADDR,DYDX       ; INC1 & DYDX now set, DYDX MSW & LSW pos
    movx     COUNT,INC1       ; [0,0]
                                ; [b',a'] (b' & a' are absolute values)
                                ; [0,b']
                                ; compare a and b
                                ; branch if a' < b'
                                ; b' >= a', INC2=[0,INC1(X)]
                                ; b' >= a', no swap required
                                ; a' < b', swap a' and b'
                                ; [0,b'] b'='a'
                                ; [b'',a''], b''=a', a''=b', b'' >= a''
                                ; INC2=[INC1(Y),0]
                                ; a and b now both positive and b >= a
                                ; 2 * b
                                ; a
                                ; d=2 * b - a
                                ; a + 1
    line2:   clr             INC2
    move     DYDX,SADDR       ; save evrything
    srl      16,SADDR         ; draw line
    cmpxy    SADDR,DYDX       ; recall everything
    jrv      line3            ; move vertices right one
                                ; draw line again to implement
                                ; DOT STRETCHING.
    line3:   movx             DYDX,SADDR
    rl       16,DYDX          ;
    movy     INC1,INC2        ;
    line4:   add             SADDR,SADDR
    movx     DYDX,COUNT       ;
    sub      COUNT,SADDR      ;
    addk     1,COUNT          ;
    MMTM     SP,SADDR,DADDR,COUNT,PATRN ;
    LINE     0                ;
    MMFM     SP,SADDR,DADDR,COUNT,PATRN ;
    ADDI     [1,0],DADDR      ;
    LINE     0                ;
    jrnc     done             ;
    horiz:   jrn             gpixel
    jrnv     do_fill          ;
    subxy    SADDR,DYDX       ; make DeltaX positive : DYDX = 0-(b,a)
    move     DYDX,SADDR       ;
    subxy    SADDR,DADDR      ; change start to (y1,x1)
    jrnc     do_fill          ;
    vert:    jrnv             do_fill
    neg      SADDR             ; make DeltaY positive
    subxy    SADDR,DADDR      ; change start to (y1,x1)
    do_fill: move             SADDR,DYDX
    addi     020001H,DYDX     ; add 1 to each dimension so line covers all dots
                                ; and is a dot thick; add 1 more to x so 2 dots thick
                                ; thus implementing DOT STRETCHING
    fill     XY

```

```

    jruc    done
gpixel:
    movi    [1,0],INC2    ; just a dot
    DRAV    INC2,DADDR    ; draw dot
5         DRAV    INC2,DADDR    ; draw again to implement DOT STRETCHING
done:
    RETS

```

```

* END DRAW LINE ROUTINE

```

10

Claims

- 15 1. A method for adjusting a plurality of colors associated with a plurality of images displayed on a screen of a color display system, comprising the steps of:
 - providing a user with a softkey menu for color adjustment, which enables the user to adjust each of six attributes of a color, the attributes comprising red value, green value, blue value, hue, saturation, and luminosity; wherein the attributes red value, green value, and blue value belong to one colour system and the attributes hue, saturation, and luminosity belong to another colour system;
 - 20 displaying a user adjustment of one of said colour attributes as a corresponding change in the attributes of the other colour system;
 - thereby enabling the user to select any desired color.
- 25 2. The method according to claim 1, further comprising the step of providing a color editor resident in the display and a superimposed window on the screen, and wherein the window displays red value, green value, and blue value as at least one of a) numeric readouts and b) bars in a bar graph format which change in height, as the user changes the red, green, and blue values.
- 30 3. The method according to claim 2 wherein the window further displays readouts for hue, saturation, and luminosity, which change as the user adjusts values, thereby providing a simultaneous display of all six attributes to enable the user to view how the color is being impacted by any changes being made to one of the six attributes.
- 35 4. The method according to claim 1, 2, or 3 wherein the softkey menu for color adjustment further enables the user to save colors for storing a color map in a resident color map register, further comprising the step of prompting the user to enter a register number after the user depresses the save colors softkey and thereafter prompting the user to depress an enter softkey.
- 40 5. The method according to claim 1, 2, 3, or 4 wherein the softkey menu for color adjustment further enables the user to select monochrome for loading a color map with shades of green.
- 45 6. The method according to claim 1, 2, 3, 4, or 5 wherein the softkey menu for color adjustment further enables the user to select a special colors menu for providing a softkey menu enabling the user to select one of at least three color maps resident in read only memory, including two color maps optimized for two common visual impairments, respectively, and one color map for a user working in red goggles often found in optics laboratories.
- 50 7. The method according to claim 1, 2, 3, 4, 5, or 6 wherein the softkey menu for color adjustment further enables the user to select an edit colors menu, further comprising the step of prompting the user to enter a pen number corresponding to one of the images displayed on the screen after the user depresses the edit colors key for editing the color in which the selected image appears.
- 55 8. The method according to any preceding claim further including user interaction with the display, wherein the screen displays a plurality of windows, the system comprising a plurality of instruments wherein instrument states for each of at least two of the plurality of instruments are displayed in respective windows simultaneously, the method comprising the steps of:
 - providing an instrument key to initiate a sequence to access at least one of the instruments, whose instrument state is being displayed in one of the plurality of windows, to change instrument settings, the



```

      jruc      done
gpixel:
      movi      {1,0},INC2      ; just a dot
      DRAY      INC2,DADDR      ; draw dot
5      DRAY      INC2,DADDR      ; draw again to implement DOT STRETCHING
done:
      RETS

* END DRAW LINE ROUTINE

```

10

Claims

- 15 1. A method for adjusting a plurality of colors associated with a plurality of images displayed on a screen of a color display system, comprising the steps of:
 - providing a user with a softkey menu for color adjustment, which enables the user to adjust each of six attributes of a color, the attributes comprising red value, green value, blue value, hue, saturation, and luminosity; wherein the attributes red value, green value, and blue value belong to one colour system and the attributes hue, saturation, and luminosity belong to another colour system;
 - 20 displaying a user adjustment of one of said colour attributes as a corresponding change in the attributes of the other colour system;
 - thereby enabling the user to select any desired color.
- 25 2. The method according to claim 1, further comprising the step of providing a color editor resident in the display and a superimposed window on the screen, and wherein the window displays red value, green value, and blue value as at least one of a) numeric readouts and b) bars in a bar graph format which change in height, as the user changes the red, green, and blue values.
- 30 3. The method according to claim 2 wherein the window further displays readouts for hue, saturation, and luminosity, which change as the user adjusts values, thereby providing a simultaneous display of all six attributes to enable the user to view how the color is being impacted by any changes being made to one of the six attributes.
- 35 4. The method according to claim 1, 2, or 3 wherein the softkey menu for color adjustment further enables the user to save colors for storing a color map in a resident color map register, further comprising the step of prompting the user to enter a register number after the user depresses the save colors softkey and thereafter prompting the user to depress an enter softkey.
- 40 5. The method according to claim 1, 2, 3, or 4 wherein the softkey menu for color adjustment further enables the user to select monochrome for loading a color map with shades of green.
- 45 6. The method according to claim 1, 2, 3, 4, or 5 wherein the softkey menu for color adjustment further enables the user to select a special colors menu for providing a softkey menu enabling the user to select one of at least three color maps resident in read only memory, including two color maps optimized for two common visual impairments, respectively, and one color map for a user working in red goggles often found in optics laboratories.
- 50 7. The method according to claim 1, 2, 3, 4, 5, or 6 wherein the softkey menu for color adjustment further enables the user to select an edit colors menu, further comprising the step of prompting the user to enter a pen number corresponding to one of the images displayed on the screen after the user depresses the edit colors key for editing the color in which the selected image appears.
- 55 8. The method according to any preceding claim further including user interaction with the display, wherein the screen displays a plurality of windows, the system comprising a plurality of instruments wherein instrument states for each of at least two of the plurality of instruments are displayed in respective windows simultaneously, the method comprising the steps of:
 - providing an instrument key to initiate a sequence to access at least one of the instruments, whose instrument state is being displayed in one of the plurality of windows, to change instrument settings, the



instrument key to facilitate a change in settings of any instrument displayed in any of the windows that appears on the screen, the instrument key being depressed by a user for initiating a sequence for selecting the instrument whose settings are to be changed;

responding to user depression of the instrument key by displaying a first softkey menu having a plurality of user selections corresponding to the instruments whose instrument states appear in the windows displayed on the screen;

responding to user depression of a softkey that appears in the first softkey menu by displaying a second softkey menu having a plurality of user selections corresponding to available settings of the selected instrument; and

responding to user depression of a softkey that appears in the second softkey menu by enabling means for adjusting a selected setting of the selected instrument;

thereby providing two-stroke access to an instrument displayed in any window displayed on-screen, thus facilitating control of instruments whose instrument states appear in the plurality of windows.

9. The method according to claim 8 wherein different colors are used to indicate correspondence between the instruments displayed in the windows and instrument selections on the first softkey menu.
10. The method according to claim 8 or 9 wherein label identifications of the instruments displayed in the windows appear in the first softkey menu.
11. The method according to claim 9 wherein each instrument is provided with a label identification and a softkey containing at least a portion of the label identification for the instrument is displayed in inverse video in the same color as the associated window border.
12. The method according to claim 10 wherein multiple instruments having the same model number are used, and the row and column addresses of the instruments having the same model number are displayed to distinguish between instruments having the same model number.
13. The method according to claim 11 wherein the label identifications are model numbers, multiple instruments having the same model number are used, and the row and column addresses of the instruments having the same model number are displayed in the first softkey menu to distinguish between instruments having the same model number.
14. The method according to any preceding claim further including enhancing selected images displayed on the screen of the display, the screen comprising a plurality of rows of pixels, each row of pixels comprising a line of the display, the method comprising the steps of:
 - drawing every line of the display once; and
 - drawing selected lines of the display twice, wherein the second time that the line is drawn, the start and the end points are moved one pixel horizontally to an adjacent pixel in the respective rows of pixels containing the start and the end points, respectively;
 - thereby applying pixel stretching to any selected line on the screen to provide clearer, more refined images.
15. The method according to claim 14 wherein the display system comprises a microprocessor, which handles all I/O, graphics scaling, and stroke generation functions, and a graphics system processor, which handles raster conversion of vector information into dots on the screen, the two processors being operated asynchronously, with the graphics system processor constantly parsing a vector list and writing dots into a bit map, wherein to avoid any artifacts produced by viewing data on the screen while the graphics system processor is updating the bit map, two separate bit maps are used, one of which is viewed while the other is drawn, the two bit maps being swapped each time the graphics system processor completes one complete scan of the vector list.

Patentansprüche

1. Ein Verfahren zum Einstellen einer Mehrzahl von Farben, die einer Mehrzahl von Bildern zugeordnet sind, die auf einem Bildschirm eines Farbanzeigesystems angezeigt sind, mit folgenden Schritten:
Bereitstellen eines Soft-Tastenmenüs zur Farbeinstellung für einen Anwender, das es dem Anwender ermöglicht, jedes von sechs Attributen einer Farbe einzustellen, wobei die Attribute einen Rot-Wert, Grün-



- Wert, Blau-Wert, den Farbton, die Sättigungsstufe und die Helligkeit umfassen; wobei die Attribute Rot-Wert, Grün-Wert und Blau-Wert zu einem Farbsystem gehören und die Attribute Farbton, Sättigungsstufe und Helligkeit zu einem anderen Farbsystem gehören;
Anzeigen einer AnwenderEinstellung eines der Farbattribute als eine entsprechende Änderung der Attribute des anderen Farbsystems;
wodurch es dem Anwender ermöglicht wird, irgendeine erwünschte Farbe auszuwählen.
2. Verfahren nach Anspruch 1, das ferner den Schritt des Bereitstellens eines Farbeditors umfaßt, der in der Anzeige und in einem überlagerten Fenster auf dem Bildschirm vorhanden ist, und wobei das Fenster den RotWert, Grün-Wert und Blau-Wert als zumindest a) numerische Ausgaben oder b) Balken in einem Balkengraphformat anzeigt, das seine Höhe ändert, wenn der Anwender den Rot-, Grün- und Blau-Wert ändert.
3. Das Verfahren nach Anspruch 2, bei dem das Fenster ferner Ausgaben des Farbtons, der Sättigungsstufe und der Helligkeit anzeigt, die sich verändern, wenn der Anwender Werte einstellt, wodurch eine gleichzeitige Darstellung aller sechs Attribute geschaffen wird, um es dem Anwender zu ermöglichen, zu sehen, wie die Farbe durch irgendwelche Änderungen, die bei einem der sechs Attribute durchgeführt wird, beeinflusst wird.
4. Das Verfahren nach Anspruch 1, 2 oder 3, bei dem das Soft-Tastenmenü zur Farbeinstellung dem Anwender ferner ermöglicht, Farben zu speichern, um eine Farbtabelle in einem residenten Farbtabellenregister zu speichern, das ferner den Schritt des Aufforderns des Anwenders umfaßt, um eine Registernummer einzugeben, nachdem der Anwender die Soft-Taste zum Sichern der Farben niederdrückt, und den Anwender danach auffordert, eine Eingabe-Soft-Taste niederzudrücken.
5. Das Verfahren nach Anspruch 1, 2, 3 oder 4, bei dem das Soft-Tastenmenü zur Farbauswahl es dem Anwender ferner ermöglicht, monochrom auszuwählen, um eine Farbtabelle mit Grünschattierungen zu laden.
6. Das Verfahren nach Anspruch 1, 2, 3, 4 oder 5, bei dem das Soft-Tastenmenü zur Farbeinstellung es dem Anwender ferner ermöglicht, ein spezielles Farbenmenü auszuwählen, um ein Soft-Tastenmenü zu schaffen, das es dem Anwender ermöglicht, zumindest eine von drei Farbtabellen, die in einem Nur-Lese-Speicher vorhanden sind, auszuwählen, die zwei Farbtabellen einschließen, die für zwei verbreitete Sehbeeinträchtigungen optimiert sind, und die eine Farbtabelle für einen Anwender, der mit einer roten Schutzbrille arbeitet, wie sie oft in optischen Laboratorien angetroffen wird, einschließt.
7. Das Verfahren nach Anspruch 1, 2, 3, 4, 5 oder 6, bei dem das Soft-Tastenmenü zur Farbeinstellung es dem Anwender ferner ermöglicht, ein Menü zur Bearbeitung der Farben auszuwählen, das ferner den Schritt des Aufforderns des Anwenders umfaßt, eine Stiftnummer einzugeben, die einem der auf dem Bildschirm angezeigten Bilder zugeordnet ist, nachdem der Anwender zur Bearbeitung der Farbe, in der das ausgewählte Bild erscheint, die Bearbeite-Farben-Taste niederdrückt.
8. Das Verfahren nach irgendeinem vorhergehenden Anspruch, das ferner eine Anwenderwechselwirkung mit der Anzeige einschließt, wobei der Bildschirm eine Mehrzahl von Fenstern anzeigt, wobei das System eine Mehrzahl von Geräten umfaßt, wobei die Gerätezustände für jedes von zumindest zwei der Mehrzahl von Geräten in jeweiligen Fenstern gleichzeitig angezeigt werden, wobei das Verfahren folgende Schritte umfaßt:
Bereitstellen einer Geräte-Taste, um eine Sequenz zu initiieren, um auf zumindest eines der Geräte zuzugreifen, dessen Gerätezustand in einem der Mehrzahl der Fenster angezeigt ist, um Geräteeinstellungen zu ändern, wobei die Geräte-Taste eine Änderung der Einstellungen irgendeines Geräts, das in irgendeinem der Fenster angezeigt ist, das auf dem Bildschirm erscheint, vereinfacht, wobei die Geräte-Taste durch einen Anwender niedergedrückt wird, um eine Sequenz zur Auswahl des Geräts, dessen Einstellungen verändert werden sollen, zu beginnen;
Reagieren auf das Niederdrücken der Geräte-Taste durch den Anwender durch Anzeigen eines ersten Soft-Tastenmenüs mit einer Mehrzahl von Anwenderauswahlen, die den Geräten entsprechen, deren Gerätezustände in den Fenstern erscheinen, die auf dem Bildschirm angezeigt sind;
Reagieren auf ein Niederdrücken einer Soft-Taste durch den Anwender, die in dem ersten Soft-Tastenmenü erscheint, durch Anzeigen eines zweiten Soft-Tastenmenüs mit einer Mehrzahl von Anwenderauswahlen, die den möglichen Einstellungen des ausgewählten Geräts zugeordnet sind; und



- Reagieren auf das Niederdrücken einer Soft-Taste durch den Anwender, die in dem zweiten Soft-Tastenmenü erscheint, durch eine Aktivierungseinrichtung zum Einstellen einer ausgewählten Einstellung des ausgewählten Geräts;
 wodurch ein Zugriff mit zwei Tastenbetätigungen auf ein Gerät geschaffen wird, das in irgendeinem Fenster angezeigt ist, das auf dem Bildschirm angezeigt ist, wodurch die Steuerung der Geräte, deren Gerätezustände in der Mehrzahl von Fenstern erscheinen, vereinfacht wird.
9. Das Verfahren gemäß Anspruch 8, bei dem unterschiedliche Farben verwendet werden, um die Zuordnung zwischen den in den Fenstern angezeigten Geräten und den Geräteauswahlen in dem ersten Soft-Tastenmenü anzuzeigen.
 10. Das Verfahren nach Anspruch 8 oder 9, bei dem die Markierungsidentifikationen der in den Fenstern angezeigten Geräte in dem ersten Soft-Tastenmenü erscheint.
 11. Verfahren gemäß Anspruch 9, bei dem jedes Gerät mit einer Markierungsidentifikation versehen ist und bei dem eine Soft-Taste, die zumindest einen Abschnitt der Markierungsidentifikation für das Gerät enthält, in inverser Darstellung in derselben Farbe wie die zugeordnete Fensterumrandung gezeigt ist.
 12. Das Verfahren gemäß Anspruch 10, bei dem mehrere Geräte, die dieselbe Modellnummer haben, verwendet werden, und die Zeilen- und Spaltenadressen der Geräte, die dieselbe Modellnummer haben, angezeigt werden, um zwischen den Geräten mit derselben Modellnummer zu unterscheiden.
 13. Das Verfahren gemäß Anspruch 11, bei dem die Markierungsidentifikationen Modellnummern sind, wobei mehrere Geräte mit derselben Modellnummer verwendet werden, und die Zeilen- und Spaltenadressen der Geräte mit derselben Modellnummer in dem ersten Soft-Tastenmenü angezeigt sind, um zwischen Geräten mit derselben Modellnummer zu unterscheiden.
 14. Das Verfahren gemäß irgendeinem vorhergehenden Anspruch, das ferner die Verbesserung von ausgewählten Bildern einschließt, die auf dem Bildschirm der Anzeige angezeigt sind, wobei der Bildschirm eine Mehrzahl von Pixelzeilen umfaßt, wobei jede Pixelzeile eine Linie der Anzeige umfaßt, wobei das Verfahren folgende Schritte umfaßt:
 Zeichnen jeder Linie der Anzeige einmal; und
 Zeichnen ausgewählter Linien der Anzeige zweimal, wobei beim zweiten Mal, wenn die Linie gezeichnet wird, der Start- und Endpunkt um ein Pixel horizontal zu einem benachbarten Pixel in der jeweiligen Pixelzeile, die den Start- bzw. Endpunkt enthält, bewegt werden;
 wodurch auf jegliche ausgewählte Linie auf dem Bildschirm eine Pixel-Streckung angewandt wird, um klarere, feinere Bilder zu schaffen.
 15. Das Verfahren gemäß Anspruch 14, bei dem das Anzeigesystem einen Mikroprozessor, der alle E/A, Graphikskalierungen und Tastenbetätigungserzeugungsfunktionen handhabt, und einen Graphiksystemprozessor umfaßt, der die Rasterumwandlung von Vektorinformationen in Punkte auf dem Bildschirm handhabt, wobei die zwei Prozessoren asynchron betrieben werden, wobei der Graphiksystemprozessor konstant eine Vektorliste analysiert und Punkte in eine Bittabelle schreibt, wobei, um irgendwelche Artefakte zu vermeiden, die durch Darstellung von Daten auf dem Bildschirm erzeugt werden, während der Graphiksystemprozessor die Bittabelle aktualisiert, zwei getrennte Bittabellen verwendet werden, von denen eine angezeigt wird, während die andere gezeichnet wird, wobei die zwei Bittabellen jedesmal ausgetauscht werden, wenn der Graphiksystemprozessor eine vollständige Abtastung der Vektorliste beendet.

Revendications

1. Un procédé d'ajustement d'une série de couleurs associées à une série d'images affichées sur un écran d'un système d'affichage en couleurs comprenant les étapes consistant à:
 fournir à l'utilisateur, pour l'ajustement de couleurs, un menu à touches programmables qui permet à l'utilisateur d'ajuster chacun des six attributs d'une couleur, les attributs comprenant la valeur rouge, la valeur verte, la valeur bleue, la teinte, la saturation, et la luminosité; les attributs de valeur rouge, de valeur verte, et de valeur bleue appartenant à un système de couleurs et les attributs de teinte, de saturation et de luminosité appartenant à un autre système de couleurs;
 afficher un ajustement, effectué par l'utilisateur, de l'un desdits attributs de couleurs en tant que



variation correspondante dans les attributs de l'autre système de couleurs;
en permettant ainsi à l'utilisateur de choisir une couleur souhaitée quelconque.

2. Le procédé selon la revendication 1, qui comprend en outre l'étape consistant à fournir un éditeur en couleur résident dans l'affichage et une fenêtre superposée sur l'écran, la fenêtre affichant en outre la valeur rouge, la valeur verte et la valeur bleue, sous au moins l'une de deux formes a) des lectures numériques et b) des barres, dans un format de graphe à barres, qui varient en hauteur au fur et à mesure que l'utilisateur modifie les valeurs rouge, verte et bleue.
3. Le procédé selon la revendication 2, dans lequel la fenêtre affiche en outre, pour la teinte, la saturation et la luminosité, des lectures qui varient au fur et à mesure que l'utilisateur ajuste des valeurs, en réalisant ainsi un affichage simultané de la totalité des six attributs pour permettre à l'utilisateur de voir l'impact, sur la couleur, de toutes modifications en cours d'exécution sur l'un des six attributs.
4. Le procédé selon la revendication 1, 2 ou 3 dans lequel le menu à touches programmables destiné à l'ajustement de couleurs permet en outre à l'utilisateur de sauvegarder des couleurs pour mémoriser une carte de couleurs dans un registre résident de cartes de couleurs, le procédé comprenant en outre l'étape consistant à inviter l'utilisateur à entrer un numéro de registre quand l'utilisateur a actionné la touche programmable de sauvegarde de couleurs et à inviter ensuite l'utilisateur à actionner une touche programmable d'entrée.
5. Le procédé selon la revendication 1, 2, 3, ou 4 dans lequel le menu à touches programmables destiné à l'ajustement de couleurs permet en outre à l'utilisateur de choisir une touche désignée comme monochrome pour charger une carte en couleurs à nuances de vert.
6. Le procédé selon la revendication 1, 2, 3, 4, ou 5 dans lequel le menu à touches programmables destiné à l'ajustement de couleurs permet en outre à l'utilisateur de choisir un menu spécial de couleurs pour réaliser un menu à touches programmables qui permet à l'utilisateur de choisir une carte au moins parmi trois cartes en couleurs, résidentes en mémoire morte, qui comprennent deux cartes en couleurs optimisées pour deux influences défavorables visuelles communes, respectivement, et une carte en couleurs destinée à un utilisateur qui travaille avec des lunettes rouges que l'on trouve fréquemment dans des laboratoires optiques.
7. Le procédé selon la revendication 1, 2, 3, 4, 5, ou 6 dans lequel le menu à touches programmables destiné à l'ajustement de couleurs permet en outre à l'utilisateur de choisir un menu d'édition de couleurs, le procédé comprenant en outre l'étape consistant à inviter l'utilisateur à entrer, quand l'utilisateur a actionné la touche d'édition en couleurs, un numéro de plume qui correspond à l'une des images affichées sur l'écran pour éditer la couleur dans laquelle l'image choisie apparaît.
8. Le procédé selon l'une revendication précédente quelconque qui inclut en outre une interaction de l'utilisateur avec l'affichage, l'écran affichant une série de fenêtres, le système comprenant une série d'instruments, des états d'instruments qui concernent chacun d'au moins deux instruments de la série des instruments étant affichés simultanément dans des fenêtres respectives, le procédé comprenant les étapes consistant à:
 - prévoir une touche d'instrument pour lancer une séquence permettant d'accéder à au moins l'un des instruments, dont l'état d'instrument est en cours d'affichage dans l'une des fenêtres de la série, afin de modifier des réglages d'instrument, la touche d'instrument servant à faciliter une modification des réglages d'un instrument quelconque choisi dans une fenêtre quelconque qui apparaît sur l'écran, la touche d'instrument étant actionnée par un utilisateur pour lancer une séquence de choix de l'instrument dont les réglages doivent être modifiés;
 - répondre à un actionnement de la touche d'instrument par l'utilisateur en affichant un premier menu à touches programmables qui inclut une série de choix offerts à l'utilisateur qui correspondent aux instruments dont les états d'instruments apparaissent dans les fenêtres affichées sur l'écran;
 - répondre à l'actionnement d'une touche programmable par l'utilisateur, qui apparaît dans le premier menu de touches programmables, en affichant un deuxième menu de touches programmables qui inclut une série de choix offerts à l'utilisateur qui correspondent à des réglages disponibles de l'instrument choisi; et
 - répondre à un actionnement, par l'utilisateur, d'une touche programmable qui apparaît dans le deuxième menu de touches programmables en validant des moyens d'ajustement d'un réglage choisi de



l'instrument choisi.

en réalisant ainsi un accès en deux actionnements à un instrument affiché dans une fenêtre quelconque affichée sur l'écran, ce qui facilite une commande des instruments dont les états d'instruments apparaissent dans la série des fenêtres.

5

9. Le procédé selon la revendication 8, dans lequel des couleurs différentes sont utilisées pour indiquer une correspondance entre les instruments affichés dans les fenêtres et des choix d'instruments sur le premier menu à touches programmables.

10

10. Le procédé selon la revendication 8 ou 9 dans lequel des identifications par étiquettes sur les instruments affichés dans les fenêtres apparaissent dans le premier menu à touches programmables.

15

11. Le procédé selon la revendication 9, dans lequel chaque instrument est pourvu d'une identification par étiquette, et une touche programmable contenant au moins une partie de l'identification par étiquette pour l'instrument est affichée en vidéo inverse à la même couleur que le cadre de fenêtre associé.

20

12. Le procédé selon la revendication 10, dans lequel de multiples instruments dont le numéro de modèle est le même sont utilisés, et les adresses de rangée, de colonne des instruments dont le numéro de modèle est le même sont affichés de manière à distinguer entre des instruments dont le numéro de modèle est le même.

25

13. Le procédé selon la revendication 11 dans lequel les identifications par étiquettes sont des numéros de modèle, de multiples instruments dont le numéro de modèle est le même sont utilisés, et les adresses de rangée et de colonne des instruments dont le numéro de modèle est le même sont affichées dans le premier menu à touches programmables afin de distinguer entre les instruments dont le numéro de modèle est le même.

30

14. Le procédé selon une revendication précédente quelconque incluant en outre une amélioration d'images choisies affichées sur l'écran de l'affichage, l'écran comprenant une série de rangées de pixels, chaque rangée de pixels comprenant une ligne de l'affichage, le procédé comprenant les étapes consistant à:

tracer une fois chaque ligne de l'affichage; et

tracer deux fois des lignes choisies de l'affichage, les points de début et de fin étant déplacés horizontalement, la deuxième fois où la ligne est tracée, vers un pixel adjacent dans les rangées respectives de pixels contenant les points de début et de fin, respectivement;

35

en appliquant une extension de pixels à une ligne choisie quelconque sur l'écran afin de réaliser des images plus claires, plus affinées.

40

15. Le procédé selon la revendication 14, dans lequel le système d'affichage comprend un microprocesseur qui traite toutes les fonctions d'entrée/sortie, de mise à l'échelle de graphiques, et de génération d'actionnement, et un processeur du système de graphiques qui traite une conversion en réseau d'une information de vecteur en points sur l'écran, les deux processeurs étant mis en oeuvre de façon asynchrone, le processeur du système de graphiques analysant constamment une liste de vecteurs et écrivant des points dans une carte de bits, deux cartes de bits dont l'une est vue tandis que l'autre est en cours de traçage étant utilisées afin d'éviter tous parasites produits par l'observation de données sur l'écran tandis que le processeur du système de graphiques est en train de mettre à jour la carte de bits, les deux cartes de bits étant échangées l'une contre l'autre chaque fois que le processeur du système de graphiques complète un balayage complet de la liste des vecteurs.

45

50

55