To: Graphics Guys *Date:* February 4, 1975

From: Dick Shoup *Loc.:* Palo Alto

Subject: Old Software on the Color Video System *Org.:* PARC/CSL

This document describes very briefly the capabilities available to a user at the terminal of the color video system (except for SUPERPAINT, described in another memo). This is not a user manual for the system but merely a compendium of frequently used programs. Its really not fair to call the collection of separate programs described here a "command language" nor has that been claimed. What can be seen here are some of the capabilities which are available. Most of these capabilities have been integrated into the SUPERPAINT program with a simple tablet directed command structure.

Using these routines requires calling them individually from the DOS exec by typing their names in the usual way (with the usual delay). Thus one might use the PAINT program to create some subimages, position them using MOVEPA, adjust colors using SETCOLOR, and save the result on disk with SAVPA. Subimages might then be combined using RESPA and resaved, etc. The file format for point array files is given in the appendix.

What follows is organized like so: a brief description of the overall system, then descriptions of painting, etc. programs, point array routines, color/value routines, and demo and miscellaneous programs.

COLOR GRAPHICS SYSTEM

The graphics Nova is interfaced to a number of peripherals, notably the picture memory (PM) which holds one color digital video image (480x640 points), the tablet, and the video disk. The PM hardware also includes a 256-color map and some special logic (such as run length encoding and decoding) to aid in manipulating point array images. All this is under control of the Nova.

Software existing on the system consists of a large variety of separate programs, most of which run under the Nova DOS. These include painting programs, point array manipulating routines, spline curve editors, color space manipulators, jaggy removal demos, a version of Smalltalk, etc. Most of the recent programs are written in BCPL and run with a set of support routines called G06. G06 software includes routines for reading and writing the PM, reading and writing point array files, handling the color space, tablet, etc.

Throughout this document, a small rectangular area on the tablet frame will be referred to as the "pad". This area is touched with the pen to indicate an escape, a default, etc., as described below. Also, some of the programs (the more recent ones) use the DEFAULT and GO areas marked on the tablet (like SUPERPAINT does).

PAINTING, ETC. PROGRAMS

PAINT — This is the main painting program. It handles the palette, the brushes, etc. and has the area-filling routine within it. Brushes and colors are selected by touching them via the tablet. To invoke area-filling: touch the desired palette color, touch the upper left-hand corner of the screen, touch the desired area.

Global switches:

/C — Causes the "canvas" part of the screen to be cleared before executing (normally it is left unchanged).

Local switches:

file/P — Causes *file* to be used as the palette (default is STDPALETTE.PA). Any point array file may be used (see SAVPA) and will be positioned centered on the palette area.

file/B — Causes *file* to be interpreted as the list of brush names (see MAKELIST). Brushes are just point array files and *file* is a text file. Default is STDBRUSHES.LI.

NPAINT — This is a version of PAINT with two pictures into which to paint. One has the brushes and palette on the edges like PAINT, the other (accessed by touching the pen down on the small pad on the tablet frame) is completely free for painting. To switch back and forth betwen the two pictures, hit the pad again. A smaller palette is used (NPAL.PA) since only 16 colors are available in each picture. No area filling facility.

Global switches:

/B — Specifies that *both* pictures are to be displayed simultaneously. This causes interesting mixing and overlay effects in the resultant color space.

Local switches: (same as PAINT).

MAKELIST *filename* — This routine creates a text file named *filename* and puts in it a list of the names you type (after the colon prompt). This list file (usually with extension .LI) can hold the names of point array files (six or fewer) and can be used as a brush list to the PAINT program with /B. E.g. if you create files B1, B2 and B3 with SAVPA, then do:

```
MAKELIST BL.LI [carriage return]
:B1 B2 B3 [carriage return]
PAINT BL.LI/B [carriage return]
```

You will then be able to paint with brushes B1, B2 and B3.

POINT ARRAY MANIPULATING ROUTINES

SAVPA — Save point array file. This is the main routine for saving part or all of the PM contents in a point array file (usually extension .PA). The desired file name is given in response to a TTY prompt. Allows specification of the desired rectangular area by indicating two corners and an origin (arbitrary positioning point) via the tablet. Default (if pen is touched down in DEFAULT area of the tablet) is the whole screen.

RESPA — Restore point array file. The desired file name is given in response to a TTY prompt. A single point is specified via the tablet and the origin point of the PA is placed there. If the DEFAULT area is touched, then the PA is positioned where it was when it was saved.

TRANSPA — Transform (translate, rotate and scale) point array. This program does a transformation on a point array from the PM and places it back in the PM. There are no switches and the parameters are given via tablet and TTY in response to prompts on the TTY. This routine is not very smart and can only handle PAs which can fit in core.

MOVEPA — Does a simple translate of a point array. You specifiy a rectangular area, and origin point and a destination point.

ZOOMIN — Using two corner cursors supplied sequentially, an origin for the area, and a destination origin for the result, this routine scales a point array up by a factor of two. You specify a rectangular area, an origin for the area, and a destination origin for the result. The default area (indicated by pen down on DEFAULT) is the entire screen.

ZOOMOUT — Similar to ZOOMIN but shrinks everything by a factor of 2.

REFLECT — This routine provides a mirror image reflection about the 0, 90, 180 or 270 degree axis (specified via the TTY). The reflection axis is a line which passes through the center of the rectangular area.

Global switches:

/T — Allows specification of the point array from the tablet. Default (pen down on the pad) is the whole screen.

COLOR/VALUE ROUTINES

At each point in the picture memory, a *value* is stored. A value is an 8 bit

number. It is these values that make up a point array. Associated with each of the 256 values is a *color definition.* Color definitions are placed in the color map and the PM hardware accesses this map to generate the red, green and blue signals which produce the appropriate colors on the screen.

The hardware also has the capability to replace all occurrences of a given value with another given value throughout the PM in one frame time. The following three routines use this capability.

SETVALUE *value newvalue* This routine causes all occurrences of *value* to be replaced by *newvalue* throughout the picture.

Global switches:

/T Allows *value* and *newvalue* to be specified from the tablet via two successive pen downs.

SWAP This routine exchanges all occurrences of two values which you specify via the tablet.

FLASH This routine produces an entertaining effect by SWAPing all values sequentially through the value space.

ZAP Sets the entire PM to the value given in response to a TTY prompt.

The following routines manipulate the color definitions in the color map rather than the values:

SETCOLOR This routine allows the user to set the color definition of any value individually. Three slider scales are displayed on the screen representing the red, green and blue components of the color. A particular value is first selected via the tablet (by touching any point outside the slider area). Then the sliders can be moved to alter the color definition of the selected value. All occurrences of the selected value will be seen to change color as the sliders are moved.

Global switches:

/H The three sliders represent hue, saturation and brightness instead of red, green and blue respectively. The program does the appropriate computation to transform from HSV space to RGB space.

/N Does not display the sliders but otherwise operates identically.

SET~~HUE~~ MAP *filename* This program sets the colormap from the color definitions given in the PA file *filename.* If any of the following switches are set, the program sets all 256 color definitions to shades of a single hue with high saturation. The hue is specified by the switch:

Global switches:

/B	Blue.
/C	Cyan.
/G	Green.
/M	Magenta.
/R	Red.
/W	White ("grayscale").
/Y	Yellow.

If no file name is given and no switch is given, the colormap is set to the color definitions which were hardwired into the PM before the colormap existed. In those days, the 8 bits were allocated RRRBGGGB, thus providing 3 bits of red, 3 bits of green, and 2 bits of blue information.

COLORCUBE — This is an elaborate program for setting the entire color space, collectively or individually, to colors of constant primary, colors of constant hue, saturation or brightness, demonstrating simultaneous color perception phenomena, equal energy planes, etc. Its operation is controlled entirely from the tablet and can best be understood by observing a demo.

FADE — This program allows all the existing color definitions to be faded to black (like a conventional video fade). The y coordinate of the tablet is used, full color being near the top of the tablet and total black near the bottom.

CYCOLORS — This program provides an entertaining effect by shifting all the color definitions sequentially through the value space.

Global switches:

/B	Leaves the background (value 0) unchanged during cycling.
/V	Waits for vertical interval (1/60 second apart) before each color map change.

MISCELLANEOUS AND DEMO PROGRAMS

FASTFILL — This is the same area filling program used in PAINT. It requires a tablet point to specify the desired value and a tablet point to specify the area to be filled.

WHPM — White PM. Clears the entire PM to the background value (0) and sets the color definition of the background to white.

BLPM — Black PM.

ZPM — Resets all PM logic and does BLPM.

COLORS — Writes into the PM a pattern of 256 squares representing the 256 possible values.

COLORBARS — Writes the standard color bar pattern into the PM.

ATOD — Causes the output of the real time A/D converter to be continuously dumped into the PM whenever the pen is down. When the pen is then lifted, the last image stored will be retained. To set this up, some switches need to be in the right positions. Ask somebody how to do this. If the colorspace has been set using SETHUE, a monochrome image will result.

PICRUNS — This program types out some statistics (the number of runcodes, etc.) within a rectangular area specified via the tablet.

TYPECOLOR — Types the value and color definition of a point selected via the tablet. Repeats until the pen is touched down on the pad.

TYPECS — Types the entire color space and a count of those values currently in use.

PLVD *frame1 frame2* — Plays video disk from *frame1* to *frame2*.

Global switches:

/C — Repeats.

PCM *file* — P-curve movie. Takes a path (sequence of points) from the tablet, then records on the video disk a sequence of frames representing the motion of the point array named *file* through that path.

Global switches:

/C — Takes a continuous path rather than a sequence of discrete points.

VTPM — This is a character generator program which writes characters into the PM. Various escape characters are used to specify font, color, background color, etc:

$C character color
$F defines the font
$P take position from tablet

$T	use transparent background
$B	background color

GSMALL
BOBSMALL

These are versions of Smalltalk which run on the PM system. The turtle (with a colored tail of course) draws into the PM. Details are available elsewhere.

JO *file*

This is the jaggy removal demo program. It scan converts line segments specified in *file* and writes them into the PM using monochrome values to remove the quantization effects. Best to do SETHUE/W before this. Suitable demo files are: WHEEL, UHL (Upper Horizontal Lines), LHL (Lower Horizontal Lines), GUY.L, MAO.L ("long live Chairman Mao").

JOQ *file*

Same as JO but leaves the quantization effects ***in.***

FONTEDIT

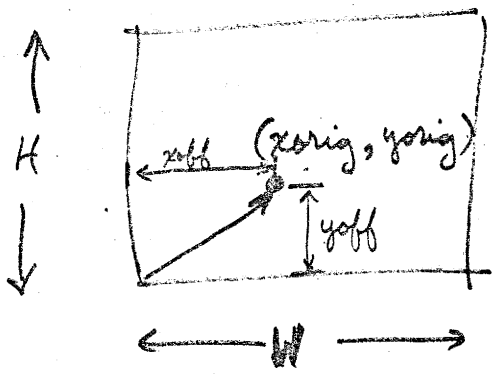
This program allows the user to define and edit font character outlines via spline curves and write the results out on files. Documentation is available from Bob Flegal.

APPENDIX A.
POINT ARRAY FILE DEFINITIONS


```
//          ---- STRUCTURES ----

STRUCTURE COLOR:
        [ VALUE         WORD 1
          BLANK         BIT 4
          C0            BIT 1     //      IN USE BIT
          C1            BIT 1
          RED           BIT 10
          BLANK         BIT 4
          C2            BIT 1
          C3            BIT 1
          GREEN         BIT 10
          BLANK         BIT 4
          C4            BIT 1
          C5            BIT 1
          BLUE          BIT 10
                        ]

MANIFEST [ COLORLEN=4 ]


STRUCTURE PAFH:                         // POINT ARRAY FILE HEADER
        [ PAFID         WORD 1  // PAF IDENTIFIER
          SPARE1        WORD 1
                                        #120000  Q0 only
                                        #050000  Q1 only
                                        #170000  both Q
          MODE          WORD 1
          SPARE3        WORD 1
          WINDOW        WORD 1  // SOURCE WINDOW (see below)
          SPARE5        WORD 1          PTR

          COLORTABPTR   WORD 1
          NUMVALS       WORD 1
          SPARE8        WORD 1
          SPARE9        WORD 1

          SPARE10       WORD 1
          SPARE11       WORD 1

          FORMAT        WORD 1  // TELLS HOW DATA IS STORED  0
          DATAPTR       WORD 1
          NUMTABS       WORD 1  — max no. of runs on a scan line in this PA
          SPARE15       WORD 1  ]

                        // WINDOW, COLORS (DATA)  — see below
                        //    FOLLOW (IN THAT ORDER)


MANIFEST [ PAFHLEN=16 ]         // HEADER LENGTH

                        // LENGTH OF PAF = PAFHLEN
                        //     + WINLEN + 256*COLORLEN
                        //     + NUMTABS*WINDOW>>WIN.H

MANIFEST [ PAFID=#66666 ]       // IDENTIFIER
```

ptrs relative to beginning of paf

```
STRUCTURE WIN:                          // WINDOW
        [ YORIG         WORD 1
          XORIG         WORD 1
          H             WORD 1
          W             WORD 1
          YOFF          WORD 1
          XOFF          WORD 1 ]

MANIFEST [ WINLEN=6 ]
```



Window def.

Runtable order: