Painter Tech Note #1

Color Set Text File Syntax

Mark Zimmer
Fractal Design Corporation

Special Palettes and Color Sets

Painter maintains color sets in files. These palettes are maintained as text files so they can be edited in Teach Text, Microsoft Word, or a programming text editor, like Think C.

The File "Painter Colors"

When Painter starts up, it reads the default color set file "Painter Colors" (PAINTER.PCS on a PC). This file must be in the Painter folder. Color set files each contain a set of colors and a format specification for the color set window. The "Painter Colors" file currently contains a standard set of colors somewhat corresponding to the standard 216-color RGB color cube palette favored by Apple. So the user can now easily choose strong colors. Also, you can use this file (and others which we supply) as a working example of color set text file syntax.

The Color Set Window

Painter maintains a new window, called the Color Set, which appears on the Windows menu. Command-equals brings the window up or puts it away, toggling its visibility. The Color Set contains patches of colors and, optionally, text for each patch. Clicking the cursor in one of these patches picks up the patch as the current color. The Color Set window's format, size, and shape are defined in the color set file. So, the actual size of the window is controlled by this formatting information.

Color Set File Format

Color Sets are maintained in text files. Each line in the text file must be a **format line** or a **color line**. When a color set file is read in to Painter , the color set window is actually resized and filled with the new colors.

Color Lines

Let's start by describing color lines. Color lines describe the color as an RGB value followed by a name, as follows:

```
R: ###, G: ###, B: ### the color's name
```

Here, all punctuation is mandatory. Here spaces may be added for formatting purposes, but not inside numbers. Letters may be in lower or upper case. The ### values are decimal numbers between 000 and 255, with the leading zeros (which are optional, to make the file more readable). After the corresponding blue ### value you must include a space and then the color's name. Names can be up to 31 characters long. Here is an example of some color lines:

```
R: 059, G:033, B:023 YR-5M
R: 095, G:056, B:036 YR-10M
R: 177, G:098, B:058 YR-15
R: 144, G:065, B:045 YR-20
R: 126, G:075, B:044 YR-20M
R: 248, G:103, B:062 YR-40
R: 168, G:111, B:072 YR-40M
```

Format Lines

A format line is always a word followed by a decimal number. A typical color set file begins with format information, and here is an example of the format header:

```
ROWS 10
COLS 6
WIDTH 38
HEIGHT 16
TEXTHEIGHT 12
SPACING 1
```

The ROWS Line

The ROWS format line describes the number of rows of colors displayed in the color set window.

The COLS Line

The COLS format line likewise describes the number of columns of colors displayed in the color set window.

The WIDTH Line

The WIDTH format line describes the width of a color patch in pixels.

The HEIGHT Line

HEIGHT format lines descibe the height of a color patch in pixels. All color patches are the same size.

The TEXTHEIGHT Line

The TEXTHEIGHT field specifies the number of pixels of vertical space for the text. A value of 0 in this format line specifies a color set with no text. A value of 12 gives the proper amount of space for geneva 9 text. The text is underneath the color patch, in geneva 9. Note that the WIDTH specification must be sufficiently wide for all text, or the text will be shortened and an ellipsis (...) will be inserted automatically.

The SPACING Line

The SPACING format line controls the number of pixels spacing in between the patches. A value of 0 is allowed, but 1 or 2 is more common.

Color Set Window Size

You can compute the color set window's size from the formatting information in the following way:

The width of the color set window is given by:

SPACING + (WIDTH + SPACING) * COLS

and the height of the color set window is given by the following expression:

SPACING + (HEIGHT + SPACING) * ROWS.

Painter Tech Note #2

Mark Zimmer, Fractal Design Corporation

Painter Script Text File Syntax

In Painter 2.0 and later versions, you can record and play back sessions. A session is like a list of commands which are executed one at a time. The sessions are kept in an internal "binary" format. In Painter 3, the sessions can be exported to a text file, edited (presumably) and then imported back into Painter and run. So the export command converts the binary format to text and the import format reads the text format and converts it to the binary format.

This document is a reference for the text file script syntax. This allows you to edit scripts, process existing Painter-made scripts, and also to create your own scripts to be played by Painter.

A Note on Meta-Syntax

To describe the syntax of script lines, a meta-syntax is used here. Words or phrases surrounded by angle brackets, like <opcode> for instance, are logical quantities (generally identifiers) which are to be replaced by legal words, like stroke_start for example. Sometimes numeric parameters are indicated in this manner. Optional segments of a script line are surrounded by square brackets. When in doubt, we try to indicate what's what by giving a text example.

Script Lines

A script file is a standard text file composed of a list of commands. Most commands fit on one text line. Some operations, like Apply Lighting, Brush Strokes, and Frisket definitions are composed of multiple line definitions. Most lines in the script definition take on the form

```
<opcode name> <operand> <operand>...
```

Some commands have no parameters (like the deselect command, for one). Many commands take as single parameter. Others take multiple parameters (called operands above). The opcodes are all "identifiers", in other words, a string of alphabetic or numeric characters (or the underscore character "_", which is legal as part of an identifier).

Making Brush Strokes

stroke_start
pnt
stroke_end

Brush strokes are formatted in a fairly simple way. The stroke is preceeded with a stroke_start command and ended with a stroke_end command. Between these two commands, the stroke is encoded as points. Both stroke_start and stroke_end take no parameters, so they each get a line to themselves. The points of the stroke are encoded as pnt commands. The pnt command syntax is as follows:

```
pnt x <x val> y <y val> time <time> [prs <pressure>] [tlt <tilt>] [brg <bearing>]
```

Here the pressure clause may be omitted if the pressure is all the way up (has a value of 1.0). Pressure, tilt, and bearing are all normalized to a range of 0...1. Tilt and bearing are parameters of control (and come only from CalComp tablets presently). Omit tilt and bearing unless you want to control those values. Here's a real example of a stroke.

```
stroke_start
pnt x  244.50 y   98.95 time 0 prs 0.09
pnt x  224.28 y  108.97 time 4 prs 0.11
pnt x  200.24 y  121.47 time 15 prs 0.14

        <many points omitted here>

pnt x  315.92 y  275.25 time 974 prs 0.17
pnt x  351.29 y  258.86 time 985 prs 0.04
pnt x  387.67 y  241.65 time 995 prs 0.00
stroke_end
```

The x and y coordinates are in picture space (do not include the picture-to-screen transform). Subpixel imformation is gladly accepted here: that's the way Painter works internally. The time stamps are measured in milliseconds. Typical tablets feed Painter a sample every 4-15 milliseconds, depending upon the tablet. The brush stroke points can be any distance apart. Painter automatically vectorizes the stroke between the two points as needed.

Selecting Brushes, Colors, and Paper Textures

variant

Brush selection is encoded in the variant command. The syntax of a variant command is:

```
variant "brush library name" "category name" "variant name"
```

The brush library name is a simple file name (this file has to exist in the Painter folder). The category and variant names are all standard brush and variant names from Painter. An example of such a line is:

```
variant "Painter Brushes" "Pencils" "Colored Pencils"
```

color

Color selection is encoded in the color command. Colors are maintained in RGB (red green blue) color space. The syntax is:

```
color red <red value> green <green value> blue <blue value>
```

The red, green, and blue values are component values between 0 and 255 inclusive. 0 means that the component is all the way off, 255 means that it is all the way on. So the command to select the color black is:

```
color red 0 green 0 blue 0
```

background_color

The secondary color (in the color palette) is addressable by using the background_color command. The syntax is:

```
background_color red <red value> green <green value> blue <blue value>
```

The red, green, and blue values are component values between 0 and 255 inclusive. 0 means that the component is all the way off, 255 means that it is all the way on. So the command to select the color white as a secondary color (for color ramp usage) is:

```
background_color red 255 green 255 blue 255
```

texture

Selecting a paper texture from a paper library can be done using the texture command. This command references libraries and paper textures by name. The syntax is:

```
texture <library name> <texture name>
```

Here the library name and the texture name are both separately enclosed in quotes. An example if this is the following:

```
texture "Paper Textures" "Basic Paper"
```

scale_slider

To set the scale of the paper texture, another command, scale_slider is used. This command has a simple syntax:

```
scale_slider <value>
```

An example of this is the default 100 percent scale, which is expressed as:

```
scale_slider 1.0
```

If the session is being played back at a different scale, the value is scaled up or down accordingly.

grain_inverted

To set the Inverted check box in the paper palette, the grain_inverted command is used. It's syntax is like other check box commands:

```
grain_inverted <option>
```

Here the <option> string can be unchecked or checked. So the default is expressed in the following example:

```
    grain_inverted unchecked
```

random_grain

Similarly, you can access the Random Grain Brush Strokes check box by using the random_grain command. The syntax is as follows:

```
    random_grain <option>
```

Here the <option> string can be unchecked or checked. To make the brush work like the spatter airbrush, use the following example:

```
    random_grain checked
```

grain_position

This command controls the position of the grain, its mapping onto the image. This is similar to the dialog you get when using the Position button in the Paper Palette, but more powerful. The format is:

```
    grain_position oper <p1> row <p2> col <p3>
```

Here <p1> is an operator type, taking on the following values:

0	no operation
1	set grain position to (row, col)
2	reset to (0,0)

The <p2> and <p3> parameters are row and column position for the grain.

Working With Files

new

This command is used to create a new image file (within Painter, but not yet attached to a disk file). Its format is:

```
    new width <w> height <h> resolution <r> width_unit <wu> height_unit <hu>
      resolution_unit <ru> paper_color red <v1> green <v2> blue <v3>
```

Note that this command is expressed as one line, without an intermediate carriage return. Here is an example of a new command:

```
    new width 400 height 400 resolution   75.00000 width_unit 1 height_unit 1
      resolution_unit 1 paper_color red 255 green 255 blue 255
```

In the format, <w> and <h> define the width and height of the new file in whatever units are defined by <wu> and <hu>. It's easy to set <wu> and <hu> to a value of 1 and then <w> and <h> are in pixels. But technically, <wu> and <hu> are values between 1 and 6, defined as follows:

1	Pixels

2	Inches
3	Centimeters
4	Points
5	Picas
6	Columns

The <r> value is the resolution of the image, defined in pixels per inch is <ru> is 1 or pixels per centimeter if <ru> is 2. Note that <v1>, <v2>, and <v3> define the paper color for the new image. They are all on the range 0...255 inclusive, so that 255,255,255 defines the color white.

open

This command opens a file by name. It first determines its type and then opens it accordingly. The format is:

```
open "filename"
```

Here the filename is the name of the file. Use the colon character do separate volume or folder (directory) names from the file name at the end.

clone_open

This command establishes the currently opened file as the clone source, and then opens a file by name as its clone. The format is:

```
clone_open "filename"
```

Here the filename is the name of the file. Use the colon character do separate volume or folder (directory) names from the file name at the end.

close

This command closes the current image. It does not interactively ask the user if the file is to be saved.

clone

This command clones (duplicates) the current image. The current image is marked as clone source before the duplication, so cloning can proceed.

change_file

This command allows you to change between one image and another in a session. Its format is:

```
change_file "filename"
```

The filename is a quoted string (as shown) and must be unique to have the correct effect. This specifies the image file (which must already be open) that is to become current. Note: a new command generates files which are called "Untitled", and if there are more than one of them, the change_file command cannot be used to switch between them properly.

annotation

This command creates an annotation within an image. The format is:

```
annotation source x <p1> y <p2> dest x <p3> y <p4>
```

Here, <p1> and <p2> define the location in image space of the pixel being annotated: it defines the color being annotated. The <p3> and <p4> parameters define the destination: the position of the annotation tag in image space. The tag is filled up with whatever color in the color set matches the annotated color.

Applying Effects to Selections and Floating Items

motion_blur

This command applies motion blur to the current selection or selected floating item. Its format is:

```
motion_blur radius <v1> angle <v2> thinness <v3>
```

Here, <v1> is the size of the motion blur effect, on the range 1.0 through 100.0 inclusive. The angle of motion is defined by <v2>, which is on the range 0.0...3.1416, as an angle in radians. The thinness parameter <v3> is on the range 0.01...1.0, and defines the sharpness in the direction perpendicular to the motion direction. For those who understand signal processing, this command accomplishes convolution with a arbitrarily-oriented elliptical disk.

enforce_printables

This command converts the selection or floating item so that all colors in the selection or item are printable (based on the current printer definition).

apply_screen

This command executes the Apply Screen... command, thresholding a continuous tone image into two or three flat colors. Its format is:

```
apply_screen using <p1> threshold1 <p2> threshold2 <p3> red1 <p4>
  green1 <p5> blue1 <p6> red2 <p7> green2 <p8> blue2 <p9>
  red3 <p10> green3 <p11> blue3 <p12>
```

Note that this command is expressed as one line, with no intermediate carriage returns. Here the <p1> parameter defines what is used as the layer of texture to convolve the image with. It must be an integer between 1 and 4 inclusive. The following table defines the using parameter here:

1	Paper Grain
2	Frisket
3	Image Luminance
4	Original Luminance

Note that <p2> and <p3> define the primary and secondary threshold values. Note that the image luminance and the <using> channel luminance are both on the range 0...100, so the sum of the two is on the range 0...200. Thus the threshold values should also be on the range 0.0 through 2.0 (since they are real numbers rather than percentages). Here <p4>, <p5>, and <p6> define the red, green, and blue component of the first color. Also, <p7>, <p8>, and <p9> define the red, green, and blue component of the second color, and <p10>, <p11>, and <p12> define the red, green, and blue component of the third color. These components are all on the range 0 through 255 inclusive.

apply_surface_texture

This command applies surface texture to the current selection or floating item. The format is:

```
apply_surface_texture amount <value1> light_direction <value2>
  using <value3> [shiny]
```

This command is expressed as a single line with no intermediate carriage returns. The <value1> is an amount setting for the command, with a value between 0 and 100 inclusive. The light direction is a value between 0 and 7 inclusive, defined as follows:

0	North
1	Northeast
2	East
3	Southeast
4	South
5	Southwest
6	West
7	Northwest

The using value is a value between 1 and 5 inclusive, defined as follows:

1	Paper Grain
2	3D Brush Strokes
3	Frisket
4	Image Luminance
5	Original Luminance

The shiny code may optionally be appended to create a glossy result like plastic as opposed to diffuse result like paper.

adjust_dye_concentration

This command adjusts the dye concentration of the image inside the current selection or floating item. The format is:

```
adjust_dye_concentration maximum <value1> minimum <value2>
  using <value3>
```

This command is expressed as a single line with no intermediate carriage returns. Here <value1> is a floating point value between 0.0 and 8.0 inclusive, indicating the maximum amount of concentration of dyes within the selection. The <value2> represents the minimum concentration of dyes, and is between 0.0 and 8.0. The using parameter is a value between 1 and 8, defined as follows:

1	Uniform Adjustment

2	Paper Grain
3	Frisket
4	Image Luminance
5	Original Luminance

color_overlay

This command overlays the current color over a selection or floating item, modulated by a using parameter. The format is:

```
color_overlay opacity <value1> using <value2> model <value3>
```

Here, value1 is a value between 0.0 and 1.0 for the hiding power model, and between 0.0 and 8.0 for the dye-concentration model. The <value2> parameter defines which channel is used to modulate the overlay process, and is an integer value between 1 and 5 inclusive, defined as follows:

1	Uniform Color
2	Paper Grain
3	Frisket
4	Image Luminance
5	Original Luminance

The <value3> is 0 for a hiding power (additive) color model, and 1 for a dye-concentration (subtractive) color model.

sharpen

This command sharpens (via unsharp masking) the current selection or the currently selected floating item. Its format is:

```
sharpen radius <value1> highlight <value2> shadow <value3>
```

The <value1> represents the size of the detail which is sharpened, in pixels. It ranges between 0.1 and 200.0 inclusive. The <value2> is a floating point number between 0.0 and 1.0 inclusive which defines the amount of effect in the highlights. The <value2> is a floating point number between 0.0 and 1.0 inclusive which defines the amount of effect in the shadows.

soften

This command softens (blurs) the image in the current selection or the currently selected floating item. Its format is:

```
soften radius <value>
```

Here <value> is a floating point number between 0.1 and 200.0 inclusive defining the size of the blur, in pixels. The larger the value, the more the blur. For those who understand signal processing, this command accomplishes convolution with a circular disk.

highpass

This command highpasses the image in the current selection or the currently selected floating item, leaving only the details, and eliminating the large features. Its format is:

```
highpass radius <value>
```

Here <value> is a floating point number between 0.1 and 200.0 inclusive. The value is in pixels and divides the level of details which are kept from those which are thrown away. For those who understand signal processing, this command accomplishes the image minus the image convolved with a circular disk and is biased by gray.

posterize

This command limits the number of levels of color in a selection or floating item by limiting each component of color (red, green, and blue) to a smaller number of possible values. Its format is:

```
posterize levels <value>
```

Here <value> is an integer between 2 and 255 inclusive. A value of 2 turns a grayscale image into a pure thresholded black and white image, where each pixel is either black or white.

negative

This command inverts the colors within the current selection or the currently selected floating item. It accomplishes direct inversion, by inverting the color's red, green, and blue components.

delete

This command deletes the current selection or floating item. If it was a selection in the image, a hole is left behind, filled with the paper color for the image. If it was a floating item, the floating item is removed.

equalize

This command equalizes the range of luminance tones in the current selection or currently selected floating item. The format is:

```
equalize black_point <p1> white_point <p2> gamma <p3> gamma_width <p4>
  [entire_image]
```

This command is expressed as a single line with no intermediate carriage returns. Here <p1> represents the black point luminance value (on the range 0...255) and <p2> represents the white point luminance value (also on the range 0...255). These two values specify the range of tones that are expanded to the full range from black to white. Each of the three components of the color are expanded similarly.

The <p3> and <p4> parameters define the chosen midpoint and the equalize slider width (a fixed value of 178). The optional entire_image keyword may be appended to indicate that the operation is to be applied to the entire file even if a selection is present.

blobs

This command deposits blobs in the selection or currently selected floating item, simulating a liquid effect used in marbling. The format is:

```
blobs n <count> min_size <p1> max_size <p2> subsample <n>
```

Here the <count> parameter defines the total number of blobs deposited. The blobs range randomly from <p1> pixels to <p2> pixels in size (values are in floating point, between 0.1 and 1000.0 inclusive). The <n> value defines the number of subsamples in each direction to use for anti-aliasing the blobs.

apply_marbling

This command passes a rake or comb through a liquid pan, which has a color image in it, as contained in the current selection or floating item. The rake or comb is defined in terms of evenly spaced teeth. The parameters for the command define the comb or rake and its path as well as other rendering attributes for the marbling pass. The format is:

```
apply_marbling spacing <p1> offset <p2> pull <p3> waviness <p4>
  wavelength <p5> phase <p6> quality <p7> <direction>
```

This command is expressed as a single line with no intermediate carriage returns. Here <p1> is the inter-tooth spacing value from 0.001 to 1.0, which is defined as a fraction of either the width (for vertical passes) or the height (for horizontal passes). Also, <p2> defines the offset of the first tooth from the left (for vertical passes) or from the top (for horizontal passes). An offset value of 0.0 through 1.0 are allowed. The <p3> value specifies the pull amount and whether the motion of the rake is forwards of backwards. Floating-point values of -10.0 through 10.0 are allowed. A positive pull value means a left-to-right motion (for horizontal passes) or a top-to-bottom motion (for vertical passes). A negative pull value means a right-to-left motion (for horizontal passes) or a bottom-to-top motion (for vertical passes). The <p4> value defines the size of sinusoidal waves that the path of the rake undergoes, measured as a fraction of the intertooth spacing. Waviness can take on values from 0.0 through 10.0. Parameter <p5> defines wavelength of the sinusoidal wave as a fraction of the width (for horizontal passes) or of the height (for vertical passes). Values between 0.0 and 1.0 are allowed. And <p6> defines the within the cycle of a full sine wave which the rake starts at, normalized onto the unit range of 0.0 through 1.0. The <p7> value is an integer between 1 and 4 inclusive which defines the quality of the marbling rendering. The higher the value, the better the rendering. Finally, the <direction> parameter is the direction, either the keyword horizontal or the keyword vertical is placed here.

apply_lighting
light
end_lighting

This command applies lighting in the 3D sense upon the selection or the floating item. This command is split into a single header command, followed by light commands, and terminated with the end_lighting command, as follows:

```
apply_lighting exposure <p1> ambient red <p2> green <p3> blue <p4> lights <n>
```

Here the <p1> command is the exposure for the entire lighting, which is a floating point value between 0.0 and 2.0, inclusive. The <p2>, <p3>, and <p4> parameters define the ambient light color and amount as red, green, and blue fractions. This is added to the cumulatively lighted result before compensating for exposure. Values lie between 0.0 and

1.0, inclusive. The <n> value defines the number of lights, between 0 and 1000, inclusive. This command line is fillowed by 0 or more light commands, formatted as follows:

```
light x <p5> y <p6> dist <p7> vx <p8> vy <p9> vz <p10>
  spread <p11> brightness <p12> color red <p13> green <p14> blue <p15>
```

This command is expressed as a single line with no intermediate carriage returns. Here, <p5> and <p6> define the position the light is pointed at (not necessarily where the shine occurs) as an unconstrained fraction relative to selection coordinates, represented in floating point. For instance, the center of the selection would be x 0.5 y 0.5.

The distance parameter <p7> represents the distance of the light from the plane of the image along the direction vector to the light. Values between 0.1 and 10.0 are allowed. The direction vector is defined by <p8>, <p9>, and <p10>, which are a three-dimensional unit-length vector to the light. They all generally range between -1.0 and 1.0. The <p11> parameter defines the spread angle of the light source, measured in radians. A value between 0.001 and 3.1416 are allowed. Also, <p12> represents the brightness of the light, between 0.0 and 2.0 inclusive. The light's color is defined by <p13>, <p14>, and <p15>. These are all integers between 0 and 255 inclusive and specify the color in RGB.

At the end, an end_lighting command is placed. Its format is simply:

```
end_lighting
```

glass_distortion

This command applies glass distortion to the selection or floating item. Its format is:

```
glass_distortion scale <p1> using <p2> normalize <p3> nsamples <p4>
```

The <p1> parameter define the scale of the distortion (the higher the value, the more the distortion). A value between 0.1 and 10000.0 is allowed. The <p2> parameter is an integer on the range 1 through 6 inclusive which define which channel is used to define the distortion. The values are defined as follows:

1	Paper Grain
2	3D Brush Strokes
3	Frisket
4	Image Luminance
5	Original Luminance

The <p3> parameter is a normalizer, which helps to normalize the texture and allow for a huge range of possible inputs. Its value represents the average of the length of the gradient vector over all pixels. The gradient vector at a pixel depends upon its neighbors, and is equal to (delta-x, delta-y). The <p4> parameter specifies the number of glass distortion samples which are averaged (as increasing distortion). A special value of 20 specifies a displacement map instead of a refraction based upon the gradient of the distortion channel.

flip_vertical

This command causes the current selection to be flipped vertically, like a reflection in a lake. This command is not legal on a frisket. It operates on floating items of any kind, though.

flip_horizontal

This command causes the current selection to be flipped horizontally, like a reflection in a mirror. This command is not legal on a frisket. It operates on floating items of any kind, though.

fill

This command causes the current selection in the background or current floating item to be filled with the current color.

distort
quad1
quad2

This command distorts the current selection or the currently selected floating item using four point perspective distortion. If the operation is on a selection in the image, then the selection is floated first. The distortion is defined by displacing the four corner points of the selection to four new arbitrary points, mapping a rectangle into a quadrilateral. This command is expressed as three lines, as a distort command, a quad1 command, and a quad2 command, as follows:

```
distort top <p1> left <p2> bottom <p3> right <p4> [oversampled]
quad1 p0X <p5> p0Y <p6> p1X <p7> p1Y <p8> p2X <p9>
  p2Y <p10> p3X <p11> p3Y <p12>
quad2 p01X <p13> p01Y <p14> p12X <p15> p12Y <p16>
  p23X <p17> p23Y <p18> p30X <p19> p30Y <p20>
```

This command is expressed as three single lines, each with no intermediate carriage returns. The parameters <p1>, <p2>, <p3>, and <p4> define the top, left, bottom and right pixel values of the original rectangle selection. If it is operating on a floating item, these values are in the space of the floating item and are offset by the top left of the floating selection.

The oversampled keyword is optional. It causes oversampling to occur, making the operation much (almost ten times) slower but more accurate in cases where there is a pinch or an extreme widening in the quadrilateral.

The quadrilateral coordinates are defined in floating point by the quad1 line. The top left point maps into p0, which is defined by <p5> and <p6>. The top right point maps into p1, which is defined by <p7> and <p8>. The bottom right point maps into p2, which is defined by <p9> and <p10>. The bottom left point maps into p3, which is defined by <p11> and <p12>.

The quad2 line contains values which are actually dependent upon p0, p1, p2, and p3, and thus are expressed redundantly. Basically, p01 is (p0 + p1)/2, p12 is (p1 + p2)/2, p23 is (p2 + p3)/2, and p30 is (p3 + p0)/2.

rotate

This command rotates the current selection or the currently selected floating item. If the operation is on a selection in the image, then the selection is floated first. The format is:

```
rotate <angle>
```

Here <angle> is defined as a floating point number, expressed in degrees, between -360.0 and 360.0 inclusive. A positive value indicates counterclockwise rotation, a negative value indicates clockwise rotation.

scale_selection

This command scales the current selection or the currently selected floating item. If the operation is on a selection in the image, then the selection is floated first. The format is:

```
scale_selection xfactor <x> yfactor <y>
```

Here <x> and <y> are defined as floating point numbers, scale factors, between 0.1 and 1000.0 inclusive.

video_legal

This command converts the colors within the current selection or floating item so that they are video legal. The format is:

```
video_legal system <number>
```

Here <number> is 1 or 2. The value 1 stands for NTSC encoding, and the value 2 stands for PAL encoding.

brightness_contrast

This command alters the brightness and contrast of the current selection or floating item. The format is:

```
brightness_contrast brightness <value1> contrast <value2>
```

Here <value1> and <value2> are real numbers taken from the dialog. Their semantic values are not yet documented.

auto_clone

This command places random dabs of the current brush into the current selection or currently selected floating item. In interactive Painter, the user stops this command using a click of the mouse or stylus. When in a session, a count of the total number of dabs is included. The format is:

```
auto_clone <number>
```

Here <number> is a 32-bit integer.

grid_paper
grid_paper2

This command fills the current selection or currently floating item with grid paper. Its format consists of two lines, as follows:

```
grid_paper kind <p1> offset x <p2> y <p3> spacing x <p4> y <p5>
  thickness <p6> units x <p7> y <p8> thickness <p9> [transparent]
grid_paper2 color red <p10> green <p11> blue <p12>
  background_color red <p13> green <p14> blue <p15>
```

This command is expressed as two single lines, each with no intermediate carriage returns. Here the <p1> parameter is an integer between 1 and 4 inclusive, defined as follows

1	rectangular criss-cross lines
2	vertical lines
3	horizontal lines
4	rectangular dot array (lattice)

The offset x and y values are defined by parameters <p2> and <p3> and are all offsets from (0,0) (the top-left point in the image) of the first grid origin, measured in x and y units as given below in parameters <p7> and <p8>. The spacing of the grid in x and y are defined separately by <p4> and <p5>, also controlled by the x and y units. The thickness of the grid lines are defined by <p6> which is given in units defined by <p9>. The parameters <p7>, <p8>, and <p9> define the units of measurement for x, y, and thickness amounts. These are integers between 1 and 6 inclusive, defined by the following values:

1	Pixels
2	Inches
3	Centimeters
4	Points
5	Picas
6	Columns

The parameters <p10>, <p11>, and <p12> define the color of the grid lines as an RGB color. Each of them is defined to be an integer on the range 0...255. Similarly, <p13>, <p14>, and <p15> define the background color behind the grid.

plugin_filter
hex_data

Plug-in's are supported and their local settings are stored in hex data records. The format of this command is:

```
plugin_filter "menu" "command" size <p1>
```

Here the plug-in submenu name and the plug-in command name are given in quotes, as shown. The <p1> parameter is the size of the plug-in-specific data encoded on the following lines. They are structured as follows:

```
hex_data <hex data>
```

Any number of these lines can be included, where the number of actual hex bytes is given by the <p1> parameter. The <hex data> itself is formatted as ASCII data, for example:

```
hex_data 0F 3B 27 1A 99 00 01 2D EE
```

The hex_data lines are terminated by carriage returns. Note that whitespace (a blank or tab) is allowed between hex bytes.

Controlling Selections in an Image

rectangle_selection

This command establishes a rectangular selection in the current image (not within a floating selection). Preceed this command with deselect, to be sure the last selection is not in effect. The format is:

```
rectangle_selection top <p1> left <p2> bottom <p3> right <p4>
```

Here <p1>, <p2>, <p3>, and <p4> are integer values ranging between -32767 and 32767 inclusive. Note that <p1>, the top y value of the selection, is less than <p3>, the bottom y value of the selection, and <p2>, the left x value of the selection, is less than <p4>, the right x value of the selection. All values are in pixels and may not be outside the image space.

deselect

This command deselects the current selection or the current floating object. Note: in Painter and the Painter engine, this command does not drop a floating selection if it is currently selected, it just makes it deselected.

frisket_selection
frisket_selection_end
frisket
end_frisket
frisket_start_points
fpnt
frisket_end_points
frisket_hole
frisket_end_hole

These commands are used to define an irregular selection by outline (a frisket). Friskets are automatically rendered into the mask layer by Painter when established. A frisket is defined by a set of points (in floating point, for subpixel accuracy) which make up its outline, and a set of holes each of which are, in turn, friskets. A frisket selection is defined as a set of friskets. Friskets can be defined as positive or negative. A particular syntax is required to define friskets properly.

```
frisket_selection top <p1> left <p2> bottom <p3> right <p4> count <p5>
  feather <p6>
```

This command is expressed as a single line with no intermediate carriage returns. This line begins a frisket. The <p1>, <p2>, <p3>, and <p4> parameters are rectangle information (measured in pixels in image space) as shown. This rectangle defines the bounds of the frisket set, including any feather. They all are integer values on the range -32767 through 32767 inclusive. The <p5> parameter is the count of the number of top-level friskets in the set. This count ranges from 0 through 32767, inclusive. The <p6>

parameter is the feather radius to be applied to the frisket set, and it ranges between 0.0 and 1000.0 pixels.

```
frisket_selection_end
```

This line ends a frisket definition. Place it at the very end after the last frisket.

```
frisket points <p7> holes <p8> [inverted]
```

This line begins an actual frisket (or hole) that is part of a frisket set. The <p7> parameter gives the number of points in the frisket. This is an integer generally greater than or equal to 3 and less than 10,000,000. The <p8> parameter defines the number of holes in the frisket. For example, the text letter "o" has one outline and one hole (generally). The inverted keyword can optionally be appended to indicate a top level frisket which is negative.

```
frisket_start_points
```

This line is used as a start bracket for the points which make up the frisket. It is used for any top level frisket or any hole within a top level frisket. Each frisket_start_points command must be bracketed by a following frisket_end_points command.

```
frisket_end_points
```

This line is placed after the last point in a frisket outline point list, as indicated above.

```
fpnt x <p9> y <p10>
```

This line is used to encode a frisket point. The <p9> and <p10> parameters are the x and y values of the coordinate. They are measured in pixels and are both floating point values. Note: a frisket may be partially outside the image area. Note that the last fpnt of a frisket outline definition *must* match the first fpnt of a frisket outline.

```
frisket_hole <p11>
```

This line is used to mark a frisket hole definition. It is always followed by a frisket command, then a frisket start_points command, then a number of fpnt commands, and finally a frisket_end_points command. The <p11> parameter must be the index of the frisket hole within its containing frisket, starting at 0.

```
frisket_end_hole <p12>
```

This line is used to end a frisket hole definition. It is always preceded by an end_frisket command. The <p12> parameter must be the index of the frisket hole within its containing frisket, starting at 0. This must match the <p11> parameter of the matching frisket_hole command.

```
end_frisket
```

This command ends a frisket definition, or the definition of a hole (which is an entire frisket unto itself). This line matches the frisket line.

An example of a frisket is given here. This frisket set is a collection of two friskets. The first frisket is simply an outline of

```
frisket_selection top 52 left 27 bottom 287 right 179 count 2 feather    0.00
frisket points 461 holes 0
frisket_start_points
  fpnt x  178.29 y   75.21
  fpnt x  176.92 y   73.58
  fpnt x  175.49 y   72.02

            (some lines omitted here)

  fpnt x  145.63 y   98.95
  fpnt x  146.84 y  100.12
  fpnt x  178.29 y   75.21
frisket_end_points
end_frisket
frisket points 237 holes 1
frisket_start_points
  fpnt x  341.43 y  198.50
  fpnt x  341.41 y  197.00
  fpnt x  341.36 y  195.50

            (some lines omitted here)

  fpnt x  341.31 y  203.96
  fpnt x  341.40 y  201.25
  fpnt x  341.43 y  198.50
frisket_end_points
frisket_hole 0
frisket points 149 holes 0
frisket_start_points
  fpnt x  300.33 y  203.79
  fpnt x  300.28 y  207.73
  fpnt x  300.15 y  211.56

            (some lines omitted here)

  fpnt x  300.27 y  199.77
  fpnt x  300.32 y  201.76
  fpnt x  300.33 y  203.79
frisket_end_points
end_frisket
frisket_end_hole 0
end_frisket
frisket_selection_end
```

clear_friskets

This command is used to completely clear out the frisket definition.

frisket_draw_mode

This command is used to change the way friskets protect painting operations. The format is:

```
frisket_draw_mode <p1>
```

Here <p1> is an integer with the value 0, 1, or 2. The value 0 causes the frisket not to contstrain any brushstrokes. The value 1 causes the frisket to only allow brushstrokes that are inside the frisket. The value 2 causes the frisket to only allow brushstrokes outside the frisket.

frisket_display_mode

This command is used to change the way friskets draw to the screen. This often will control what you can do with them. The format is:

```
frisket_display_mode<p1>
```

Here <p1> is an integer with the value 0, 1, or 2. The value 0 causes the frisket not to be displayed. The value 1 causes the frisket to be drawn as a color overlay. The value 2 causes the frisket to be drawn as an outline marquee. Note: value 2 is required so that a frisket can be used as a selection properly.

undo

This command causes the last action to be undone (or redone if the last command is undo).

fade

This command accomplishes partial undo (or partial redo, if the last command is an undo command). Its format is:

```
fade <number>
```

Here <number> is an integer between 0 and 100 inclusive which defines the percent of fade. Note that 100% fade is equivalent to an "undo" operation.

Working With the Clipboard

cut

This command causes the current selection (in the background or a floating selection) to be copied to the clipboard and removed from the image.

copy

This command causes the current selection (in the background or a floating selection) to be copied to the clipboard, but unlike cut the selection is not removed from the image.

paste

This command causes the current clipboard contents to be pasted into the image as a new floating item. In interactive Painter, this floating item is centered in the image window. In the Painter engine, this item is centered within the entire image.

paste_into_new_picture

This command causes the current clipboard contents to be pasted into a new image window.

clear

This command clears the selection or removes the floating item, if one is selected. If it is a background selection, then the selection is filled with the paper color (but no deselect is done, unlike for cut).

Working With the Paint Bucket

These commands control the use of the paint bucket for filling areas within the image.

paint_bucket_options

This command controls the options for a subsequent paint bucket click command. It controls what is filled, what it is filled with, and other things. The format is:

```
paint_bucket_options what <p1> with <p2> direction <p3>
```

The <p1> parameter is an integer between 0 and 4, inclusive, defined as follows:

0	contuguous pixels
1	rectangular selection
2	frisket selection
3	entire image
4	cartoon cel

Note: frisket selection fill is sensitive to the frisket_draw_mode as to whether it will fill inside or outside the frisket area. Note: entire image is used for filling floating selections always.

The <p2> parameter is an integer between 1 and 3 inclusive, defined as follows:

1	flat color
2	color ramp
3	clone

The <p3> parameter defines the direction of a color ramp. Leave this value 0 for a <p2> which is not equal to 2. But for color ramp fills, the values of <p3> are integers between 0 and 7 inclusive, defined as follows:

0	north
1	northeast
2	east
3	southeast
4	south
5	southwest
6	west
7	northwest

paint_bucket_click

This command is like the click of the paint bucket tool within an image. Its format is:

```
paint_bucket_click x <p1> y <p2>
```

Here <p1> and <p2> are the x and y values of a coordinate (measured in pixels) within the image. They are integers. This coordinate may also lie within a floating selection, if the floating selection is selected. In that case, the floating selection is filled. This command is generally preceded by a paint_bucket_options command.

paint_bucket_limit_rect

This command can specify the limit rectangle of a paint bucket fill (especially a contiguous pixels fill). Its format is:

```
paint_bucket_limit_rect top <p1> left <p2> bottom <p3> right <p4>
```

Here, the <p1>, <p2>, <p3>, and <p4>, parameters are integer values defining the top y, left x, bottom y, and right x ordinates of a rectangle in image space. These values are measured in pixels.

seed_fill_data

This command is used to define the magic wand seed fill color set. Its format is:

```
seed_fill_data hue min <p1> max <p2> [wraps] saturation
  min <p3> max <p4> value min <p5> max <p6>
```

This command is expressed as a single line with no intermediate carriage returns. All parameters are on the range 0 through 255 inclusive. The <p1> and <p2> parameters define a range of hues. If the optional wraps keyword is included, the range wraps around (includes blue) and in general includes the opposite set of hues. The <p3> and <p4> parameters define the saturation range. The <p5> and <p6> parameters define the value range. Note: <p1> is less than or equal to <p2>, <p3> is less than or equal to <p4>, and <p5> is less than or equal to <p6>.

mask_threshold_slider

This command controls the Mask Threshold slider in the Fill Palette. This adjusts the maaximum mask value which will stop a Cartoon Cel fill. The format is:

```
mask_threshold_slider <p1> percent
```

The <p1> value is an integer which ranges from 0 to 100.

Controlling Session Playback

playback_reference_rectangle

Include this command to insert a reference rectangle which defines the space of the drawing which follows. This rectangle is recorded when an image is open and a selection is present when you start recording. The format is:

```
playback_reference_rectangle top <p1> left <p2> bottom <p3> right <p4>
```

Here parameters <p1> through <p4> are integer values which represent coordinates in the current working image. Specifically, <p1> and <p3> represent the top and bottom y coordinates (y is 0 at the top of the image). Also, <p2> and <p4> represent the left and right x coordinates. The four parameters define a non-zero area rectangle in image pixels.

playback_session

This command plays back a session in the same session library as the currently executing session. Its format is:

```
playback_session "session name"
```

Here, the session name is enclosed in quotes, as shown. There is no limit of session call levels, but unlimited "x calls x" recursion is not allowed.

stroke_record

The next stroke, after this command is issued, is recorded into a single stroke recording buffer. This stroke can be repeatedly played back at different locations using the stroke_playback command.

stroke_start_playback

Use this command to start a section of stroke_playback commands.

stroke_stop_playback

Use this command to finish a section of stroke_playback commands.

stroke_playback

This command plays back a recorded stroke. This command can be included any number of times inside the stroke_start_playback and stroke_stop_playback commands. The format is:

```
stroke_playback x <p1> y <p2> [centered]
```

The <p1> and <p2> parameters are floating point locations in pixel coordinates within the current working image. This coordinate is used to place the stroke. If the optional centered keyword is included, the stroke is centered about the location given. If the optional centered keyword is omitted, then the start of the stroke is relocated to this point before the stroke is played back.

auto_playback

This command is used to automatically play back the recorded stroke within the current selection in the current image window. The selection does have to be explicit for this command, unlike for other effects. The format of this command is:

```
auto_playback <p1>
```

Here, the <p1> parameter is a 32-bit integer value which contains the total number of strokes to be played back within the selection. The strokes are randomly placed.

Controlling Brushes

There are many features in Painter which control the effects which come out of a brush. This section goes into them. The features predominantly lie in the Brush Palette, the Brush Size window, the Brush Behavior window, and the Expression Palette.

size_slider

This command controls the Size slider in the Brush Size window. The format is:

```
size_slider <p1>
```

Here <p1> is the natural logarithm of the size of the brush in pixels. For brushes which vary in size, <p1> represents the logarithm of the geometric mean of the brush sizes.

plus_or_minus_size_slider

This command controls the ±Size slider in the Brush Size window. This is used to create brushes which vary in size. The format is:

```
plus_or_minus_slider <p1>
```

Here <p1> represents the natural logarithm of the variation factor in size from the mean size to the largest brush size, or from the smallest brush size to the mean size.

percent_of_size_slider

This command controls the minimum tolerance between one brush size and another. For brushes which vary in size, a number of brush sizes are constructed. This command controls the size step. The format is:

```
percent_of_size_slider <p1>
```

Here <p1> represents the amount of change between one radius and another, values ranging between 1.01 and 2.0 inclusive.

base_angle_slider

This command controls Angle slider in the Brush Size window. This adjusts the base angle for brushes which are angled, or for those which vary in angle. The format is:

```
base_angle_slider <p1> degrees
```

Here <p1> is an angle expressed in degrees. Values between 0.0 and 180.0 are allowed. Angled brushes take longer to build than non-angled brushes. Also set the thinness_slider to see angled brushes.

delta_angle_slider

This command controls Ang Rng slider in the Brush Size window. This adjusts the range of angles (starting at the base angle as given above) for brushes which vary in angle. The format is:

```
delta_angle_slider <p1> degrees
```

Here <p1> is an angle expressed in degrees. Values between 0.0 and 180.0 are allowed.

angle_delta_slider

This command controls the Ang Delta slider in the Brush Size window. This adjusts the minimum angle tolerance for brushes which vary in angle. The format is:

```
angle_delta_slider <p1> degrees
```

Here <p1> is an angle expressed in degrees. Values between 0.0 and 180.0 are allowed. Typical values are 5 to10 degrees. The smaller the value, the longer the brush takes to build.

thinness_slider

This slider controls the Thinness slider in the Brush Size window. This adjusts the thinness of the brush disk, and is employed in all angled brushes. The format is:

```
thinness_slider <p1> percent
```

Here <p1> is an integer value between 1 and 100, inclusive. A value of 100 is used for non-angled brushes.

brush_tip

This command controls the choice of the brush tip in the Brush Size window. The format is:

```
brush_tip <p1>
```

Here <p1> is an integer, taking on the following values:

1	cusp (concentrated at center)
2	cubic (neat feathered spray)
3	linear (triangular pointed tip)
4	rounded (for pencils)
5	watercolor (ringed edge)
6	flat (one pixel edge)

build

This command is issued after changing anything in the brush size window, to rebuild the brush internal definition for painting. This command must be invoked before painting or else a dialog will appear. This command takes no parameters, and is equivalent to hitting the Build button in the Brush Size window.

penetration_slider

This command allows you to directly set the penetration slider in the Brush Palette. This slider controls the penetration of brushes into the paper grain of brushes which employ the following methods:

Grainy Edge Flat Buildup
Grainy Hard Buildup
Soft Variable Buildup
Grainy Edge Flat Cover
Grainy Hard Cover
Drip
Hard Drip
Grainy Drip
Grainy Hard Drip
Grainy Hard Cover Cloning

For all but the drip methods given above, the slider controls penetration into paper grain. For the drip brushes, penetration controls the amount of "pull" the brush exerts.

The value is set to a given value in percent. The format is:

```
penetration_slider <p1> percent
```

Here <p1> is an integer between 0 and 100 inclusive representing percentage penetration. A value of 0 will generally cause a brush not to draw, when penetration affects it. A typical setting of 19 to 25 percent gives good results.

concentration_slider

This command allows you to directly set the concentration slider in the Brush Palette. Concentration, for all brushes, controls the amount of dye or paint which comes out through the brush. The value is set to a given value in percent. The format is:

```
concentration_slider<p1> percent
```

Here <p1> is an integer between 0 and 100 inclusive representing percentage penetration. Note that a value of 0 will generally cause a brush not to draw, because it is completely transparent.

method

This command controls the Method slider in the Brush Palette. The method specifies the inner workings of a brush, in particular, the function which transfers a dab of a brush stroke to the image layer, water color layer, or mask layer. The format is:

```
method <p1>
```

Here <p1> is an integer with a value of one of the following list:

1	flat cover
2	soft cover
3	soft buildup
4	grainy flat cover
5	grainy soft cover
6	grainy soft buildup

7	grainy edge flat cover
8	grainy hard cover
9	grainy hard buildup
10	soft variable buildup
11	soft cover cloning
12	grainy soft cover cloning
13	hard cover cloning
14	grainy hard cover cloning
15	soft paper color
16	soft mask colorize
17	drip
18	drip cloning
19	soft paint remover
20	soft paint thickener
21	grainy edge flat buildup
22	grainy wet abrasive
23	wet remove density
24	grainy wet buildup
25	grainy drip
26	hard drip
27	grainy hard drip
28	soft mask cover
29	flat mask cover
30	grainy hard mask cover
31	grainy edge flat mask
32	grainy soft mask cover
33	linoleum

jitter_slider

This command controls the Dab Location Variability Amount slider in the Brush Behavior window. This command is formatted:

```
jitter_slider <p1>
```

Here <p1> can be any value between 0.0 and 4.0. A value of 0.0 is typical for most brushes. Higher values cause the brush dab locations to be distributed about the brush stroke, rather than on center.

advance_slider

This command controls the Dab-to-Dab Spacing Amount slider in the Brush Behavior window. This controls how far apart the dabs are which make up a brush stroke. This value is a percent of the current brush radius (half-diameter). The format is:

```
advance_slider <p1>
```

Here <p1> is a fraction between 0.0 and 1.0 generally.

min_advance_slider

This slider controls the Dab-to-Dab Spacing Min Spacing slider in the Brush Behavior window. This controls the minimum spacing between brush dabs, and is generally a value in image pixels. For instance, a value of 20.0 forces each dab to be twenty pixels from the last dab. The format is:

```
min_advance_slider <p1>
```

Here <p1> is a value in pixels, generally between 0.0 and 50.0.

n_multi_slider

This command controls the No. of Bristles slider in the Brush Behavior window. This adjusts the number of strokes in both kinds of multi-bristle brushes in Painter. The setting is between 2 and 20 strokes. The format is:

```
n_multi_slider <p1>
```

Here <p1> is an integer value between 2 and 20, inclusive.

resaturation_slider

This command controls the Brush Paint Reservoir Resaturation slider in the Brush Behavior window. This adjusts the flow of paint to the tip of the brush. The format is:

```
resaturation_slider <p1> percent
```

The <p1> parameter varies between 0 and 100 percent. A value of 0 will cause no paint to be applied. This is useful, along with a non-zero bleed value, for creating water or smudge brushes.

bleed_slider

This command controls the Brush Paint Reservoir Bleed slider in the Brush Behavior window. This adjusts the color pick-up from the image to the brush. The format is:

```
bleed_slider <p1> percent
```

Here <p1> is a value between 0 and 100 percent. Set this value to 0 to eliminate smudginess in the brush.

clone_jitter_amount_slider

This command controls the Clone Location Variability Amount slider in the Brush Behavior window. This adjusts the amount of jittering for certain kinds of cloning brushes. The brushes affected are those which employ the following methods:

Hard Cover Cloning
Soft Cover Cloning
Grainy Hard Cover Cloning
Grainy Soft Cover Cloning
Drip Cloning

The format for this command is:

```
clone_jitter_amount_slider <p1>
```

Here <p1> is an amount of jitteryness, in pixels. A minimum value of 0.0 and a maximum value of 50.0 are supported.

clone_jitter_time_slider

This command controls the Clone Location Variability How Often slider in the Brush Behavior window. This adjusts the amount of jittering for certain kinds of cloning brushes. The brushes affected are those which employ the following methods:

Hard Cover Cloning
Soft Cover Cloning
Grainy Hard Cover Cloning
Grainy Soft Cover Cloning
Drip Cloning

The format for this command is:

```
clone_jitter_time_slider <p1>
```

Here <p1> is an amount of time, measured in 60ths of a second, between changed in random offset during cloning. A minimum value of 1 and a maximum value of 60 are supported.

hue_variability_slider

This command controls the Color Variability ±H slider in the Color Palette. This adjusts the amount of hue jitter in subsequent brush strokes. The format is:

```
hue_variability_slider <p1> percent
```

Here <p1> takes on integer values between 0 and 50, inclusive.

saturation_variability_slider

This command controls the Color Variability ±S slider in the Color Palette. This adjusts the amount of colorfullness jitter in subsequent brush strokes (varying between grayish and saturated colors). The format is:

```
saturation_variability_slider <p1> percent
```

Here <p1> takes on integer values between 0 and 50, inclusive.

value_variability_slider

This command controls the Color Variability ±V slider in the Color Palette. This adjusts the amount of luminance jitter in subsequent brush strokes. The format is:

```
value_variability_slider <p1> percent
```

Here <p1> takes on integer values between 0 and 50, inclusive.

use_clone_color

This command controls the Use Clone Color check box in the Color Palette. When checked, this specifies that color is picked up (at any part of the stroke for single-bristle brushes, at the start of the stroke for multi-bristle brushes) from the original and is transferred through the brush into the clone. The format is:

```
use_clone_color <option>
```

Here <option> is checked or unchecked.

random_cloning

This command controls the Random Cloning check box in the Color Palette. This turns on a random lookup in the clone source, so that an original texture can be used to create a new random cloned texture. The format is:

```
random_cloning <option>
```

Here <option> can be checked or unchecked.

multiple_bristles

This command controls the Multiple Bristles check box in the Brush Behavior window. Check this command for creating multi-bristle non-real-time brushes such as the VanGogh brush. The format is:

```
multiple_bristles <option>
```

Here option is checked or unchecked, as for other check boxes. As an example, a typical setup for a VanGogh brush is:

```
multiple_bristles checked
n_multi_slider 10
value_variability_slider 25 percent
```

expression_palette_radius

This command controls the Size popup in the Expression Palette. The size of the brush, for brushes which vary in size, is controlled by whichever option is specified. The format is:

```
expression_palette_radius <p1>
```

Here <p1> is an integer, and takes on the following values:

1	none
2	velocity
3	direction
4	pressure
5	tilt
6	bearing
7	original luminance
8	random

expression_palette_jitter

This command controls the Jitter popup in the Expression Palette. The amount of Dab Location Variability is scaled from 0 to its current setting in the Brush Behavior window by whichever option is specified. The format is:

```
expression_palette_jitter <p1>
```

Here <p1> is an integer, and takes on the following values:

1	none
2	velocity
3	direction
4	pressure
5	tilt
6	bearing
7	original luminance
8	random

expression_palette_penetration

This command controls the Penetration popup in the Expression Palette. The grain penetration amount in the Brush Palette is scaled by whichever option is specified. The format is:

```
expression_palette_penetration <p1>
```

Here <p1> is an integer, and takes on the following values:

1	none
2	velocity
3	direction
4	pressure
5	tilt
6	bearing
7	original luminance
8	random

expression_palette_concentration

This command controls the Concentration popup in the Expression Palette. The concentration amount in the Brush Palette is scaled by whichever option is specified. The format is:

```
expression_palette_concentration <p1>
```

Here <p1> is an integer, and takes on the following values:

1	none
2	velocity
3	direction
4	pressure
5	tilt
6	bearing
7	original luminance

8	random

expression_palette_color

This command controls the Color popup in the Expression Palette. The color of the brush is varied between the secondary color and the primary color by whichever option is specified. The format is:

```
expression_palette_color <p1>
```

Here <p1> is an integer, and takes on the following values:

1	none
2	velocity
3	direction
4	pressure
5	tilt
6	bearing
7	original luminance
8	random

expression_palette_angle

This command controls the Angle popup in the Expression Palette. The angle of the brush, for brushes which vary in angle, is controlled by whichever option is specified. Multi-angle brushes are set up in the Brush Size window using the base_angle_slider, delta_angle_slider, angle_delta_slider, and thinness_slider commands. The format is:

```
expression_palette_angle <p1>
```

Here <p1> is an integer, and takes on the following values:

1	none
2	velocity
3	direction
4	pressure
5	tilt
6	bearing
7	original luminance
8	random

expression_palette_resaturation

This command controls the Resaturation popup in the Expression Palette. The resaturation amount in the Brush Behavior window is scaled by whichever option is specified. The format is:

```
expression_palette_resaturation <p1>
```

Here <p1> is an integer, and takes on the following values:

1	none
2	velocity
3	direction

4	pressure
5	tilt
6	bearing
7	original luminance
8	random

expression_palette_bleed

This command controls the Bleed popup in the Expression Palette. The bleed amount in the Brush Behavior window is scaled by whichever option is specified. The format is:

```
expression_palette_bleed <p1>
```

Here <p1> is an integer, and takes on the following values:

1	none
2	velocity
3	direction
4	pressure
5	tilt
6	bearing
7	original luminance
8	random

set_clone_source

This command sets an image to be the clone source for future cloning operations. The format is:

```
set_clone_source "filename"
```

The single parameter is the name of a file, quoted as shown. In order to properly handle this, the file name must be unique among those image files currently opened in Painter.

new_brush_model

This command controls the New Brush Model check box in the Brush Behavior window. This turns on (or off) the use of real-time multi-bristle brushes. The format is:

```
new_brush_model <option>
```

Here <option> is checked or unchecked.

spread_bristles

This command controls the Spread Bristles check box in the Brush Behavior window. This turns on (or off) the use of pressure as a control for the width of real-time multi-bristle brush strokes. The format is:

```
spread_bristles <option>
```

Here <option> is checked or unchecked.

edge_soften

This command controls the Edge Soften check box in the Brush Behavior window. This turns on (or off) the softening of the edge bristles of a real-time multi-bristle brush. The format is:

```
edge_soften <option>
```

Here <option> is checked or unchecked.

contact_slider

This command controls the Contact Angle slider in the Brush Behavior window. This adjusts the number of bristles in a real-time multi-bristle brush which touch the page. The format is:

```
contact_slider <p1> degrees
```

The <p1> value can be between 0 and 360 degrees. As a brush contacts a piece of paper, pressure controls the number of bristles (as they flatten out) which touch the surface.

bristle_scale_slider

This command controls the Brush Scale slider in the Brush Behavior window. This adjusts the scale of the real-time multi-bristle brushes. The format is:

```
brush_scale_slider <p1> percent
```

Here <p1> takes on values between 0 and 1000 percent. The larger values create more bristle "splay".

turn_angle_slider

This command controls the Turn Angle slider in the Brush Behavior window. This adjusts the amount of turn of a real-time multi-bristle brush as the brush stroke changes direction. The format is:

```
turn_angle_slider <p1> degrees
```

The <p1> value is an angle in degrees which ranges from 0 to 360. This helps to simulate brush "roll".

wet_paint

This command turns wet paint on or off. Its format is:

```
wet_paint [<key>]
```

Here the optional <key>parameter is either the word on or off. If the <key> parameter is not specified, then wet paint is turned on. If wet paint is turned off, an implicit dry command is done.

dry

This command causes the current water color layer to be dried and composed with the background. It does not turn off wet paint, though.

wet_fringe_slider

This command controls the Water Colors Wet Fringe slider in the Brush Behavior window. The format is:

```
wet_fringe_slider <p1>
```

The <p1> value is a fraction between 0.0 and 1.0 which controls the amount of wet fringe on watercolor strokes.

post_diffuse_slider

This command controls the Post Diffuse slider in the Brush Behavior window. This adjusts the amount of post diffusion which is applied to water color strokes. The format is:

```
post_diffuse_slider <p1>
```

Here <p1> is a number of times that post diffusion is run on the stroke after it is applied to the water color layer. The value ranges between 0 and 20 times.

Operating on Floating Items

We've added several commands you can use to access the multiple floating item capability of Painter.

float_selection

This command floats a selection in the background layer with various options. The format is:

```
float_selection top <p1> left <p2> bottom <p3> right <p4>
  [rectangular] [option]
```

This command is expressed as a single line with no intermediate carriage returns. Here <p1>, <p2>, <p3>, and <p4> represent the top, left, bottom, and right coordinates in image space of a selection to be floated. The keyword rectangular is added to indicate that a rectangular selection is being floated. If this keyword is omitted, the selection is assumed to be a frisket (outline) selection. The keyword option is added to indicate that no hole is to be left behind in the background. Omitting this keyword will cause the selected area in the background to be filled with that image's paper color after the selection is floated.

float_move

This command is used to move the selected floating item. It moves the item to a location within the image (in pixels). This location is scaled when replayed at a different scale. The format is:

```
float_move x <p1> y <p2>
```

Here <p1> and <p2> are integer values which specify the destination for the current floating object.

delete_floating_selection

This command causes the current floating selection to be deleted from the image.

drop_floating_selection

This command causes the current floating selection to be dropped into the background. The format is:

```
drop_floating_selection [masked]
```

The masked keyword is included if the selection is to be masked in any way, omitted if it is to be dropped as a rectangle.

generate_mask

This command generates a mask layer in the current image or in a floating selection within the current image, if one is selected. The format is:

```
generate_mask <p1> [inverted]
```

Here <p1> is an integer value which takes on the following values:

1	Paper Grain
2	3D Brush Strokes
3	Original Mask
4	Image Luminance
5	Original Luminance
6	Current Color

The inverted keyword is omitted if the mask is to be generated based on the value given. It is included if the mask is inverted (blask-white reversal) based on the value given.

portfolio_change

This command changes the current portfolio. The format is:

```
portfolio_change "library"
```

Here the portfolio name is included within quotes as shown.

portfolio_place

This command places a portfolio object into the current image. The format is:

```
portfolio_place "name" x <p1> y <p2>
```

Here the item name is givcen in quotes as shown. The position of the object within the image is given by <p1> and <p2> which are x and y offsets in pixels within the image (from the top left point).

drop_command

This command drops the single selected floating item, compositing it using its masking into the background layer.

floating_select

This command allows you to select a floating item. Its format is:

```
floating_select "string"
```

Here <string> is a quoted string (as shown) which is a path to the selected item or items. The string "/" is used to define the hierarchy, like UNIX pathnames. A floating selection is not referred to by name, but by number (order within the group or file). The number 1 represents the first (frontmost) item in a group, etc. The exception is that the number 0 represents the backmost floating item in the group or file. Multiple floating items are represented as a list, separated by commas. Multiple selections must all reside at the same level, by definition. For instance, if there is only one floating item and it is selected, its path would be /0. If there are only three floating items and they are all selected, their path is /1,2,3 (or alternatively /1,2,0).

floating_name_and_options

This command allows you to name and reposition the currently selected floating item. The format is:

```
floating_name_and_options top <top> left <left> algorithm <number>
```

The <top> and <left> parameters define the new top left corner of the item, defined in absolute pixel coordinates. These coordinates are scaled if the session is played back at a different scale. The <number> value is an integer from 0 to 5 which defines the compositing method. The values are given in the table below:

0	Default
1	Dye Concentration
2	Colorize
3	Reverse-Out
4	Shadow Map
5	Magic Combine

floating_duplicate_selection

This command duplicates the currently selected floating item in place and leaves it on top of the original. The duplicate is left selected and the original is deselected.

floating_feather

This command changes the feather amount of the currently selected floating item. The format is:

```
floating_feather <number>
```

Here <number> is a floating point value between 0.0 and 50.0 inclusive. This feather value is scaled appropriately if the session is played back at a different scale.

floating_opacity

This command changes the opacity of the currently selected floating item. The format is:

```
floating_opacity <number>
```

Here <number> is a floating point value between 0.0 and 1.0. The value 0.0 implies totally transparent, and the value 1.0 implies totally opaque.

floating_item_masking

This command allows you to select the masking attributes of the currently selected floating item. It accomplishes "item" masking, which means that you are selecting whether the frisket layer of the floating object is ignored, whether everything inside the mask is masked off, or whether everything outside the mask is masked off. The format is:

```
floating_item_masking <number>
```

Here <number> is 1, 2, or 3. The value 1 means no item masking (unmasked). The value 2 means Mask Inside and the value 3 means Mask Outside.

drop_all

This command performs a "drop" on all floating items, arranged back to front. It creates one flat layer, the background image, which visually matches the floating layers.

floating_restore_frisket

This command operates on the mask of the currently selected floating item. It restores the mask of the item based on its outline definition, throwing away any changes which may have been made by painting into its mask.

floating_open_group

This command opens the selected floating item, which must be a group. This makes the items it contains visible in the floating item list.

floating_close_group

This command closes the selected floating item, which must be a group. This hides the items it contains in the floating item list.

floating_group_items

This command groups the selected floating items into an open group.

floating_ungroup_items

This command ungroups the selected floating item, which must be a group.

floating_bring_to_front

This command moves the selected floating item all the way to the front (in the front-to-back heirarchy).

floating_send_to_back

This command moves the selected floating item all the way to the back (in the front-to-back heirarchy).

floating_forward_one

This command moves the selected floating item forwards one level in the heirarchy (towards the front).

floating_backward_one

This command moves the selected floating item backwards one level in the heirarchy (towards the back).

floating_background_masking

This command allows you to select the masking attributes of the currently selected floating item. It accomplishes "background" masking, which means that you are selecting whether the frisket layer of the background is ignored, whether everything inside the background mask is masked off, or whether everything outside the background mask is masked off. The format is:

```
floating_background_masking <number>
```

Here <number> is 1, 2, or 3. The value 1 means no background masking (unmasked). The value 2 means Mask Inside and the value 3 means Mask Outside.

float_move_delta

This command is used to move the selected floating item. It moves the item by a vector which is represented in delta coordinates (in pixels). This delta vector is scaled (without offsets) when replayed at a different scale. The format is:

```
float_move_delta x <p1> y <p2>
```

Here <p1> and <p2> are integer values which specify the move for the current floating object, relative to its current position.

Assorted Commands

These are commands which do not operate on a selection or have anything to do with editing in general. We just included them because they're part of scripts that Painter generates.

start_time

This command specifies the start time of the script. The format is

```
start_time date <date> time <time>
```

Here <date> is locality dependent, but in the US an example is Wed, Sep 1, 1993. Also, <time> is also locality dependent, but the example 1:21 PM suffices to show that both date and time are String2Date and String2Time compatible (Macintosh OS calls, but easily emulated). The start_time and end_time commands are not mandatory in a script.

end_time

This command is formatted exactly like start_time, except the format is:

```
end_time date <date> time <time>
```

So the interpretation of <date> and <time> are identical to the above.

delay

This command causes the script which is playing to stop for a while. The format is:

```
delay <seconds>
```

Here <seconds> is an integer which defines the number of seconds to pause.

artist_name

This command encodes the artist's name in the script. The format is

```
artist_name "string"
```

Here the string is a quoted string as shown which encloses the artist's name. An example is:

```
artist_name "John Derry"
```

new_tool

This command changes the current tool in the toolbox. The format is:

```
new_tool <value>
```

Here <value> is a number between 0 and 10, indicating the tool number. This command is generally cosmetic, affecting only the current tool chosen.

Painter Tech Note #3
ColorTalk™ Annex Notes

Mark Zimmer
September 1994

The ColorTalk Annex works on a rectangular selection. It is a language which can be used to accomplish channel operations on the selection. This annex has the additional capability of being able to operate on a floater AND the image underneath it, for more complicated compositing operations.

The language is akin to C in expression syntax. There are predefined identifiers for referring to the various components of the image or the copy. These are detailed in definitions below. For instance, if you need to add green into red, you would use:

```
r = r + g;
```

Here the red component is referred to as "r" and the green component is referred to as "g". This statement is applied individually to every pixel selected.

A ColorTalk program can be a number of statements. Statements all end with a semicolon.

ColorTalk has a number of predefined functions which come with it. These functions provide for swapping components, linear interpolation, and the usual math stuff.

ColorTalk Definitions

Predefined Identifiers
can all be used as a source

name	refers to	can be stored into
r	Image Red	yes
g	Image Green	yes
b	Image Blue	yes
a	Image Alpha (Mask)	yes
h	Image Hue	yes
s	Image Saturation	yes
v	Image Value	yes
pc	Image Process Cyan	no
pm	Image Process Magenta	no
py	Image Process Yellow	no
pk	Image Process Black	no
cr*	Background Red	yes
cg*	Background Green	yes
cb*	Background Blue	yes
ca*	Background Mask	yes
ch*	Background Hue	yes
cs*	Background Saturation	yes
cv*	Background Value	yes
cpc*	Background Process Cyan	no
cpm*	Background Process Magentano	

cpy*	Background Process Yellow	no
cpk*	Background Process Black	no
x	Selection X Fraction	no
y	Selection Y fraction	no
noise	Noise function	no
xnoise	X-dependent Noise	no
ynoise	Y-dependent Noise	no
angle	Angle from selection center	no
distance	Distance from selection center	no
grain	Grain function	no

* name available for use only when operating on a floater.

Predefined Functions and Procedures

calling example	what it does
a = min(b, c);	a is replaced by the lesser of b and c
a = max(b, c);	a is replaced by the greater of b and c
a = pow(b, c);	a is replaced by b raised to the power c
a = log(b);	a is replaced by the natural logarithm of b
a = exp(b);	a is replaced by "e" raised to the power b
a = sin(b);	a is replaced by the trigonometric sine of b
a = cos(b);	a is replaced by the trigonometric cosine of b
a = lerp(b, c, d);	a is replaced by b*(1-d) + c*d (uses d to mix between b and c)
swap(a, b);	a's and b's values are swapped
a = sqrt(b);	a is replaced by the square root of b
a = usin(b);	a is replaced by (sine(b*2*PI) + 1) / 2
a = ucos(b);	a is replaced by (cosine(b*2*PI) + 1) / 2
a = atan2(y, x);	a is replaced by the arctangent of y over x, with the proper sign
a = uatan2(y, x);	a is replaced by atan2(y, x) / (2*PI)
a = step(b, c);	if b is greater than c, then a is replaced by 1, otherwise 0
a = uclip(b);	a is replaced by b mod 1.0
a = xfposmap(b, c);	a is replaced by b remapped by the x fraction index function in c
a = abs(b);	a is replaced by the absolute value of b

Numbers

any integer or floating point number from -8 to 7.99 is legal

Statements and Delimiters

delimiter	usage
;	semicolons are used to separate multiple statements
,	commas are used in procedure and function calls
( )	parentheses bound expressions, and are used in calls

Operators

notation is standard infix, related to C expressions

operator	name	example	same as
=	assignment	a = b;	

+	addition	a = b + c;	
-	subtraction	a = b - c;	
*	multiplication	a = b * c;	
/	division	a = b / c;	
+=	add into	a += b;	a = a + b;
-=	subtract out of a -= b;		a = a - b;
*=	multiply into	a *= b;	a = a * b;
/=	divide out of	a /= b;	a = a / b;

Example Programs

(1) luminance evaluation

This program transfers luminance to mask (denoted by A), like "copy to mask luminance". The NTSC definition of luminance is used here. As you can see, simple mathematical expressions can be used.

```
a = r*0.30 + g*0.59 + b*0.11;
```

(2) Hue Value Chart

This program uses the built-in values x and y to make a two-dimensional chart. It first sets up a value ramp vertically, then sets saturation to 1, then sets up a hue ramp horizontally.

```
v = y; s = 1; h = x;
```

(3) b/w at left to color on the right

This program takes any color image and sets up a horizontal saturation ramp. The right side of the image is kept as full color. The color drops off continuously so the the image becomes black and white at the left. Note that rather than replacing saturation, we are multiplying it by a ramped value, which is automatically available in x.

```
s *= x;
```

(4) increase the saturation of an image

This program uses the POW function to gamma correct saturation. This increases it in a continuous way across the full range of saturations. This can be applied to any color image.

```
s = pow(s, 0.75);
```

(5) diagonal ramp

This program constructs a diagonal gray ramp in the selection. The upper left corner will be black and the lower right corner will be white. Different values of the linear interpolation fraction simply lead to different ramp angles.

```
v = lerp(x, y, 0.5);
```

(6) hard contrast RGB

This program runs on any image, and converts a soft contrast to a hard one. A simple cubic function is used on all 3 components.

```
r = 3*r*r - 2*r*r*r; g = 3*g*g - 2*g*g*g; b = 3*b*b - 2*b*b*b;
```

(7) simple gamma correction

This program raises the red, green, and blue components to a power in order to darken the picture. The gamma factor is 1.8.

```
r = pow(r, 1.8); g = pow(g, 1.8); b = pow(b, 1.8);
```

Painter Tech Note #5
Painter 3 Image Hose™ Nozzle Files

Mark Zimmer
John Derry

January 1995

Introduction

The image hose is a new type of mark-making tool. Instead of painting in an image with individual dabs of paint, the image hose paints with a series of image elements. The source of these image elements is a nozzle file. A nozzle file organizes image elements into a regular grid. Each grid unit holds one image element. The image hose controls the manner in which the individual elements are selected and applied to an image; the nozzle file defines the image element content of the image hose. These nozzle files are selected in the **Brush Controls: Nozzles** palette. Image elements can be come from various sources. They can be created with any tools in Painter or other image applications, or they can come from existing imagery such as photographs. Nozzle files can be organized by "ranks" which allow image elements to be sequenced by the image hose in a complex manner.

There are two methods for creating nozzle files. The simplest method is done by grouping any number of floaters together and using the **Tools: Image Hose: Make Nozzle From Group** command. The second method requires creating a file with a predetermined grid unit for placing individual image elements. This method offers much more control over the organization of the elements, particularly in regard to multiple ranks.

Creating Nozzle Files From Floater Groups

The simplest method of constructing a nozzle file is done with the **Tools: Image Hose: Make Nozzle From Group** command. Any number of floaters can be included in a group. A nozzle created this way can have only one rank.

Trimming Floater Elements

Nozzle files are based on a regular grid. To minimize a nozzle file's space requirement, it is desirable that the required grid be no larger than necessary. When using the **Make Nozzle From Group** command, this grid is automatically derived from an analysis of the maximum height and width values present within the floater group, taking both the 24-bit image layer and the 8-bit mask layer into account. The **Trim** button on the Objects: Floater List palette eliminates any blank areas in the image and mask layers of a floater and makes it as small as possible. Trimming can be operated on all floaters in a group by selecting the group name and applying the Trim command.

Collapsing Nested Groups

Normal grouping technique includes nesting groups within groups. For example, when the **Effects: Objects: Create Drop Shadow** command is applied to a group, each individual floater will become a nested group that includes the newly created drop shadow for that element. As a result, all the individual floaters in the group will become a set of individual groups within the original group. This technique is often desirable as it allows for floater re-positioning among related floaters. When preparing floaters for a nozzle file, all nested groups must be individually collapsed into single floaters. This is done with the **Collapse** button on the Objects: Floater List palette. In the case of the Create Drop Shadow dialog, the **Collapse to One Layer** checkbox is available. If there are any nested groups within a group the Make Nozzle from Group command will not create a nozzle file.

Nozzle Element Sequence Organization

The top-to-bottom order (front-to-back on-screen) of all floaters is indicated in the **Objects: Floater List** palette by their top-to-bottom order in the list. This order is used by the **Make Nozzle From Group** command to sequence the elements in a first-to-last order in a nozzle file, the top floater being the first nozzle element and the bottom floater being the last nozzle element. This is useful when creating nozzles in which a specific order is required. For example, a nozzle file that uses an image hose controlled by pressure to dictate the size of the nozzle element must have the elements sequenced in their small-to-large order.

Nozzle File Get Info Text Field

A nozzle file must contain specific information to be recognized as such when accessed via the **Load** button on the **Brush Controls: Nozzle** palette**.** When a nozzle file is created with the **Make Nozzle From Group** command this information is automatically inserted into the nozzle file's Get Info text field. The **Get Info** dialog is accessed via the File menu. An example of a one-rank nozzle Get Info text field format is shown below:

image hose 19 items (height 87, width 93)

image hose must be the first text in the string. This tells Painter it is a nozzle file.
n items describes the number (n) of image elements in the file.
(height y, width x) describes in pixels (y,x) the regular grid unit used in the file.

Creating Nozzle Files From Grids

Nozzle files created from grouped floaters are limited to a single rank. Nozzle files using multiple ranks must be constructed manually in concert with a regular grid. Because of this, a bit of planning is required to construct these files. The first step is to determine the maximum grid unit size necessary to bound the largest image element (and its drop shadow, if one is used) that will be used in the nozzle file. Use **Canvas: Grid Options** to construct these grids.

The second step is to create a new file with the correct dimensions. The height and width of the new nozzle file must be an even multiple of the corresponding height and width of the regular grid unit arrived at previously. The actual x and y multiples required to calculate the file dimensions are dependent upon the number of image elements to be used. For example, if a grid unit is (height 50, width 60), then the corresponding nozzle file dimensions must be (height 100, 150, 200, etc.; width 120, 180, 240, etc.). The nozzle file must be saved in RIFF format.

Understanding Multiple Ranks

Nozzle image elements are organized by rank; there can be up to 3 ranks in a Nozzle file. The **Brush Settings: Image Hose** dialog is used to assign one of 9 settings to each rank. These settings control the manner in which Nozzle elements are applied to the image. During this discussion, the examples are shown (in a schematic sense) in Figure 1 of this Tech Note (see the last page). Refer to the three diagrams there as necessary. A multiple-rank Image Hose can be created with a Nozzle file by opening it, selecting **File: Get Info**, and editing the info text for the file. For instance, a three-element single-rank nozzle file would have an info text like so:

image hose 3 items (height 50, width 50)

When a rank 2 nozzle file is created, this info text has to be a little different. This example describes a 12-element two-rank nozzle file with 3 columns and four rows:

image hose 3 by 4 items (height 50, width 50)

When creating a three-rank nozzle file, the info text is just an extension of the two-rank syntax. Here we describe a three column by four row by five layer nozzle file. It has 60 elements and has info text like this:

image hose 3 by 4 by 5 items (height 50, width 50)

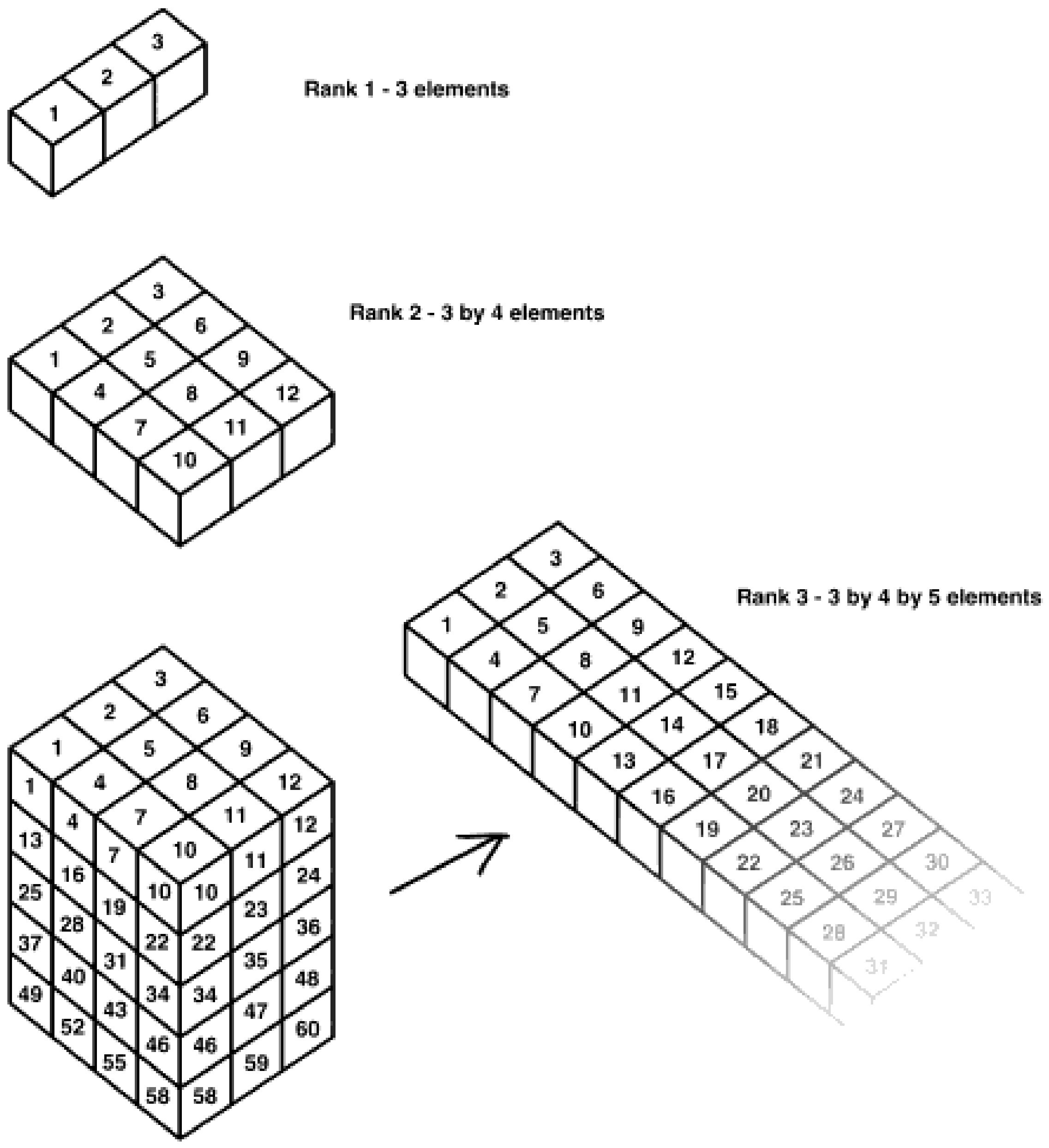
Nozzle File Layout

As shown in Figure 1, the rank 1 image hose file is represented with a single row of elements. The rank 2 nozzle expands into rows and columns. The rank 3 nozzle file has rows and columns and layers. But, as you know, nozzle files are only two dimensional. As the diagram schematically indicates, a three-dimensional nozzle file must be unstacked in order to be represented in an image file. Actually, the shape of the nozzle file really only depends upon the number of and shape of the elements in the file. Note that Painter 3 uses what we call “ravel order” (left-to-right and then top-to-bottom, like the order in which English text is read) to order the elements within the file.

What Ranks Mean

Each rank in a nozzle file is a dimensional index. In the case of a two-dimensional nozzle file, imagine a row is a set of the same object at different scales, small-to-large from left-to-right. Now imagine that each row has a different object, for example, a different leaf shape. So rank 1 represents the size of the object and rank 2 represents different objects. When loading this nozzle, you might want pressure to control rank 1 and random to control rank 2 in this case.

So, the more ranks, the more degrees of control you can have over your nozzle file, (the more elements the nozzle file has, etc.) the more work goes into creating it.



Nozzle Ranking - Figure 1