Corel Painter Tech Note -- ColorTalk™ Annex Notes

Created By: Mark Zimmer

The ColorTalk Annex works on a rectangular selection. It is a language that 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 Magenta	no
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;

<u>Example Programs</u>

(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);
```