

colorwheel_gc

COLLABORATORS

	TITLE : colorwheel_gc		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		March 29, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	colorwheel_gc	1
1.1	colorwheel_gc.doc	1
1.2	colorwheel.gadget/colorwheel.gadget	1
1.3	colorwheel.gadget/ConvertHSBToRGB	5
1.4	colorwheel.gadget/ConvertRGBToHSB	5

Chapter 1

colorwheel_gc

1.1 colorwheel_gc.doc

```
colorwheel.gadget()  
ConvertHSBToRGB()  
ConvertRGBToHSB()
```

1.2 colorwheel.gadget/colorwheel.gadget

NAME

```
colorwheel.gadget -- create standard HSB color wheel BOOPSI objects  
                    (V39)
```

FUNCTION

The color wheel class provides the ability to create gadgets enabling the user to control the hue and saturation components of an HSB (Hue-Saturation-Brightness) color space. The companion gradient slider class enables control of the brightness component of the color space.

The color wheel can operate on screens of any depth, and adapts its rendering to the number of colors available. The system's pen sharing mechanism is used in order to maximize the number of colors used by the wheel. A color wheel gadget is (normally) responsible for choosing it's own color pens to draw in (using `graphics.library/ObtainBestPen()`). However, the creator of the gadget can "donate" some pens to the gadget, using the `WHEEL_Donation` tag.

The reason that the color wheel picks its own colors is because it has the ability to display several different layouts depending on the number and variety of colors available. For example, when opening on a screen of low depth or when opening on a screen where all the pens have already been allocated exclusively, the gadget will display a "monochrome" version of the color wheel, where instead of colored segments, the letters "R" (for red), "G" (for green), "B" (for blue), "Y" (for yellow), "C" (for cyan) and "M" (for magenta) will be used as labels.

You can talk to the color wheel using HSB or RGB, even though the color wheel only really deals with HSB in its user-interface. All communications with applications are performed with full 32-bit color component values.

TAGS

WHEEL_Hue (ULONG) -- Set and get the hue component of a color wheel. This is effectively the angle around the wheel where the desired color lies. If the wheel is currently displayed, the position of the selection knob will be moved to reflect the new hue.

A hue value of 0 is all red, and nothing but red. Increasing the value moves the color towards all green at \$55555555, full blue at \$AAAAAAAA, and back to red at \$FFFFFFFF.

If you are setting or getting more than one color component at a time, it is more efficient to use the **WHEEL_HSB** tag.

Default for this tag is 0. Applicability is (ISGNU).
(V39)

WHEEL_Saturation (ULONG) -- Set and get the saturation component of a color wheel. This is effectively the distance from the center of the wheel where the desired color lies. If the wheel is currently displayed, the position of the selection knob will be moved to reflect the new saturation.

A saturation value of 0 puts the knob at the center of the wheel and always yields white. Increasing the value progressively moves the knob farther away from the center. until the value \$FFFFFFFF is reached in which case the knob is as far as it can go.

If you are setting or getting more than one color component at a time, it is more efficient to use the **WHEEL_HSB** tag.

Default for this tag is \$FFFFFFFF. Applicability is (ISGNU).
(V39)

WHEEL_Brightness (ULONG) -- Set and get the brightness component of a color wheel. The color wheel does not itself have any means of displaying or editing brightness, but it does maintain this value internally. Used with **WHEEL_GradientSlider**, this tag lets you control the value of a gradient slider object by passing **WHEEL_Brightness** to a color wheel.

A brightness value of 0 means all black. Increasing the value progressively brightens the current color, until the value \$FFFFFFFF is reached in which case the color is as bright as it gets.

If you are setting or getting more than one color component at a time, it is more efficient to use the **WHEEL_HSB** tag.

Default for this tag is \$FFFFFFFF. Applicability is (ISGU).

(V39)

WHEEL_HSB (struct ColorWheelHSB *) -- Set and get the hue, saturation, and brightness components of a color wheel. This is a buik version of the separate WHEEL_Hue, WHEEL_Saturation, and WHEEL_Brightness tags.

When setting this tag, initialize a ColorWheelHSB structure, and provide a pointer to it. When getting this tag, pass a pointer to a ColorWheelHSB structure, and the color wheel object will fill it in with the current values.

Default for this tag is a hue of 0, a saturation of \$FFFFFFFF, and a brightness of \$FFFFFFFF. Applicability is (ISGU).

(V39)

WHEEL_Red (ULONG) -- Set and get the red component of a color wheel. If the wheel is currently displayed, the position of the selection knob will be moved to reflect the new amount of red.

If you are setting or getting more than one color component at a time, it is more efficient to use the WHEEL_RGB tag.

Default for this tag is \$FFFFFFFF. Applicability is (ISGNU).

(V39)

WHEEL_Green (ULONG) -- Set and get the green component of a color wheel. If the wheel is currently displayed, the position of the selection knob will be moved to reflect the new amount of green.

If you are setting or getting more than one color component at a time, it is more efficient to use the WHEEL_RGB tag.

Default for this tag is 0. Applicability is (ISGNU).

(V39)

WHEEL_Blue (ULONG) -- Set and get the blue component of a color wheel. If the wheel is currently displayed, the position of the selection knob will be moved to reflect the new amount of blue.

If you are setting or getting more than one color component at a time, it is more efficient to use the WHEEL_RGB tag.

Default for this tag is 0. Applicability is (ISGNU).

(V39)

WHEEL_RGB (struct ColorWheelRGB *) -- Set and get the red, green, and brightness components of a color wheel. This is a buik version of the separate WHEEL_Red, WHEEL_Green, and WHEEL_Blue tags.

When setting this tag, initialize a ColorWheelRGB structure, and provide a pointer to it. When getting this tag, pass a pointer to a ColorWheelRGB structure, and the color wheel object will fill it in with the current values.

Default for this tag is a red of \$FFFFFFFF, a green of 0, and a blue of 0. Applicability is (ISGU). (V39)

WHEEL_Screen (struct Screen *) - Indicate the screen the color wheel is to open on. This is a required tag and must be provided when the wheel is created via NewObject().

Applicability is (I). (V39)

WHEEL_Abbrev (STRPTR) - When the color wheel is rendered on a display which doesn't have enough colors to allow it to draw itself in color, it automatically renders itself in monochrome instead. In such a case, the various color segments are identified by six letters G=Green, C=Cyan, B=Blue, M=Magenta, R=Red, and Y=Yellow. You can provide a replacement set of six letters. This is meant for localization of the wheel.

Default for this tag is "GCBMRY". Applicability is (I). (V39)

WHEEL_Donation (UWORD *) - Specifies an array of pens donated by the application for use by the color wheel. The array can contain any number of pens, and is terminated with a pen value of ~0. The wheel will change the RGB values of these pens to suit its needs, so be prepared for this. This means the pens should likely be allocated using the PENF_EXCLUSIVE option of the graphics.library/ObtainPen() function.

Default for this tag is NULL. Applicability is (I). (V39)

WHEEL_BevelBox (BOOL) - Set to TRUE if you want a raised bevelled box to be drawn around the wheel.

Default for this tag is FALSE. Applicability is (I). (V39)

WHEEL_MaxPens (ULONG) - Indicate the maximum number of shared color pens the wheel should attempt to allocate. This tag is useful if you wish to minimize the impact the wheel can have on your screen's pens.

Default for this tag is 256. Applicability is (I). (V39)

WHEEL_GradientSlider (struct Gadget *) - This tag lets you do simple linking of a gradient slider object to a color wheel object. You give this tag a pointer to a gradient slider object obtained previously from NewObject(). Once this is done, anytime the various tags that can affect the brightness component of the current color is sent to the color wheel, the color wheel automatically changes the value of the attached gradient slider to match that new brightness value. Reading the brightness value from the color wheel returns the current value indicated by the gradient slider.

Using this tag effectively allows you to treat the color wheel and gradient slider as a single gadget. Once things are set up, all communications occur through the wheel object, and the

gradient slider can pretty much be ignored by the application.

Default for this tag is NULL. Applicability is (IS). (V39)

GA_ID (UWORD) - Specify the gadget ID of a color wheel object.
Applicability is (ISGNU). (V39)

WARNING

Once a color wheel has been created on a given screen, the wheel object must be deleted using `DisposeObject()` prior to closing the screen. This is because the wheel object allocates pens on the screen.

BUGS

Even though all communication with the color wheel is done using full 32-bit color components, color calculations are currently done using 16-bit math, which can cause certain rounding errors to appear.

Prior to V40, the `WHEEL_Red`, `WHEEL_Green`, and `WHEEL_Blue` tags did not return the correct values when a gradient slider is linked with the color wheel using the `WHEEL_GradientSlider` tag. The workaround is to always use the `WHEEL_RGB` tag, and just extract the values from there.

1.3 colorwheel.gadget/ConvertHSBToRGB

NAME

`ConvertHSBToRGB` -- convert from an HSB color space to an RGB color space. (V39)

SYNOPSIS

```
ConvertHSBToRGB(hsb, rgb);  
    A0    A1
```

```
VOID ConvertHSBToRGB(struct ColorWheelHSB *, struct ColorWheelRGB *);
```

FUNCTION

Converts a color from an HSB representation to an RGB representation.

INPUTS

`hsb` - filled-in `ColorWheelHSB` structure containing the values to convert
`rgb` - structure to receive the converted values

BUGS

Even though all communication with the color wheel is done using full 32-bit color components, color calculations are currently done using 16-bit math, which can cause certain rounding errors to appear.

1.4 colorwheel.gadget/ConvertRGBToHSB

NAME

ConvertRGBToHSB -- convert from an RGB color space to an HSB color space. (V39)

SYNOPSIS

```
ConvertRGBToHSB(rgb, hsb);
                A0   A1
```

```
VOID ConvertRGBToHSB(struct ColorWheelRGB *, struct ColorWheelHSB *);
```

FUNCTION

Converts a color from an RGB representation to an HSB representation.

INPUTS

rgb - filled-in ColorWheelRGB structure containing the values to convert
hsb - structure to receive the converted values

BUGS

Even though all communication with the color wheel is done using full 32-bit color components, color calculations are currently done using 16-bit math, which can cause certain rounding errors to appear.