FC_ImageDisplay

Olivier LAVIALE 2004

COLLABORATORS

	TITLE : FC_ImageDisplay	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Olivier LAVIALE 2004	January 13, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 FC_ImageDisplay **1**

1.1 Feelin : FC_ImageDisplay 1

1.2 FC_ImageDisplay / FA_ImageDisplay_Spec 1

1.3 FC_ImageDisplay / FA_ImageDisplay_State 3

1.4 FC_ImageDisplay / FA_ImageDisplay_Height 3

1.5 FC_ImageDisplay / FA_ImageDisplay_Width 3

1.6 FC_ImageDisplay / FM_ImageDisplay_Setup 3

1.7 FC_ImageDisplay / FM_ImageDisplay_Cleanup 4

1.8 FC_ImageDisplay / FM_ImageDisplay_Draw 4

Chapter 1

FC_ImageDisplay

1.1 Feelin : FC_ImageDisplay

FC_ImageDisplay (03.00)

IDs: Static Super: FC_Object Include: <libraries/feelin.h>

This class is used to create and manage abstract images. An image may be a color, a pen, a picture, a brush, a function... Simple and double images are handled, use the FA_ImageDisplay_State attribute to switch between images. Images can be modified on the fly (you don't have to dispose and create a new object).

You should rarely use this class, unless you want to handle images yourself. This class is used by FC_Area to manage its background, or by FC_Image to display image buttons.

METHODS

FM_ImageDisplay_Setup FM_ImageDisplay_Cleanup

FM_ImageDisplay_Draw

ATTRIBUTES

FA_ImageDisplay_Spec FA_ImageDisplay_State

FA_ImageDisplay_Width FA_ImageDisplay_Height

1.2 FC_ImageDisplay / FA_ImageDisplay_Spec

NAME

FA_ImageDisplay_Spec -- (01.00) [IS.], STRPTR | ULONG

FUNCTION

Define the type of your image.

You can use numeric values such as FI_Shine or FI_Shine_Shadow, or a string containing specifications to create the image. Single and double images (with a render and a select state) can be created. Numeric values can only create single images.

String specification always starts with a digit or a letter, followed by a ":", followed by some parameters e.g. "c:FF0000" for full red. To create a double image, seperate each image specification with a comma. e.g. "c:FF0000,c:0000FF" for a full red render image and a full blue select image.

Currently, the following things are defined (all numeric parameters need to be ascii values!):

0:<d>

A pattern, where <d> is a decimal between FI_Shine and FI_Shine_Highlight e.g. "0:12"

p:<d>

A pen register, where <d> is a decimal value describing one of the pen registers of the current display.

s:<d>

A color-scheme register, where <d> is a decimal value describing one of the scheme entries of the current display.

c:<r,g,b>

A color, where <r,g,b> are hexadecimal values describing an RGB color e.g. "c:FF00FF" for a full red image.

P:<n>

Where <n> is the name of an external picture file that will be loaded with FC_Picture. Watch out the case !

B:<n>

Where <n> is the name of an external Feelin brush. This may result in a single or double brush. A double brush is created from two files: a "*.fb0" and a "*.fb1". If brush's name ends with a ".fb0" an alternate image will be loaded from disk, if this succeed a double brush is created, otherwise the brush remains single. Brushes are currently loaded from the "Feelin:Images/" drawer, but your can specify a full path to override this e.g. "RAM:Brushes/MyBrush.png".

H:<h>

Where <h> is an hexadecimal value describing a pointer to a struct Hook. The hook will be called with a pointer to the struct Hook in A2, a pointer to the object in A0, and a pointer to a FS_ImageDisplay_HookDraw in A1.

Everything is checked (rport != NULL, x1 <= x2, y1 <= y2...) and the rectangle is clipped (if possible) before the function is called.

void main(void) { static char myback[16];

F_RawFormat(myback,"H:%08lx",&Hook_Draw)

... Child, AreaObject, FA_Back, myback, ... }

F_HOOKM(void,Hook_Draw) { struct RastPort *rp = Msg -> Render -> RPort;

_APen(Msg -> Render -> Palette -> Pens[FV_Pen_Highlight]); _Boxf(Msg -> Rect -> x1,Msg -> Rect -> y1,Msg -> Rect -> x2,Msg -> Rect -> y2); }

Use region's coordinates to create stylish effects :

F_HOOKM(void,Hook_Draw,FS_ImageDisplay_HookDraw) { struct RastPort *rp = Msg -> Render -> RPort; ULONG ap = Msg -> Render -> Palette -> Pens[FV_Pen_Fill], bp = Msg -> Render -> Palette -> Pens[FV_Pen_HalfShadow]; WORD x = (Msg -> Region -> x2 - Msg -> Region -> x1 + 1) / 3 + Msg -> Region -> x1;

static UWORD pt[] = {0xFF00,0x7F80,0x3FC0,0x1FE0, 0x0FF0,0x07F8,0x03FC,0x01FE, 0x00FF,0x807F,0xC03F,0xE01F, 0xF00F,0xF807,0xFC03,0xFE01};

_APen(bp); _Boxf(Msg -> Region -> x1,Msg -> Region -> y1,x,Msg -> Region -> y2);

rp -> AreaPtrn = pt; rp -> AreaPtSz = 4;

_APen(ap); _BPen(bp); _Boxf(x+1,Msg -> Region -> y1,Msg -> Region -> x2,Msg -> Region -> y2);

rp -> AreaPtSz = 0; rp -> AreaPtrn = NULL; }

You must keep in mind that your backdrop may be used by childrens, windows... and is called each time an object is activated / desactivated. This is only required by stylish rendering, filling rectangles with a solid color do not require much care.

NOTE

Image can be modified on the fly. You don't have to dispose an object an create another one to be able to use a different image. Don't forget to setup the object with the FM_ImageDisplay_Setup method.

SEE ALSO

FA_ImageDisplay_State

1.3 FC_ImageDisplay / FA_ImageDisplay_State

NAME

FA_ImageDisplay_State -- (01.00) [ISG], ULONG

FUNCTION

Some images offer different states, you can select one of them by setting this attribute. Simply use one of the FV_ImageDisplay_Normal, FV_ImageDisplay_Selected...

SEE ALSO

FA_ImageDisplay_Spec

1.4 FC_ImageDisplay / FA_ImageDisplay_Height

NAME

FA_ImageDisplay_Height -- (01.00) [..G], ULONG

FUNCTION

Return image's height.

All images don't have dimension, currently only brushes and bitmaps will return their dimensions, others will return 10.

SEE ALSO

FA_ImageDisplay_Spec FA_ImageDisplay_Width

1.5 FC_ImageDisplay / FA_ImageDisplay_Width

NAME

FA_ImageDisplay_Width -- (01.00) [..G], ULONG

FUNCTION

Return image's width.

All images don't have dimension, currently only brushes and bitmaps will return their dimensions, others will return 10.

SEE ALSO

FA_ImageDisplay_Spec FA_ImageDisplay_Height

1.6 FC_ImageDisplay / FM_ImageDisplay_Setup

NAME

FM_ImageDisplay_Setup -- (01.00)

SYNOPSIS

F_DoA(Obj,FM_ImageDisplay_Setup,FS_ImageDisplay_Setup);

F_Do(Obj,FM_ImageDisplay_Setup,FRender *Render);

FUNCTION

Since your object doesn't know anything about display environnement after it is created with the FM_New method, you must send the FM_ImageDisplay_Setup method to setup your object. Colors will be allocated, pictures and brushes remaped...

NOTE

The FC_Render object supplied with the method must stay valid until you send the FM_ImageDisplay_Cleanup method. A pointer to this object is kept to allow on the fly image changes.

SEE ALSO

FA_ImageDisplay_Spec

1.7 FC_ImageDisplay / FM_ImageDisplay_Cleanup

NAME

FM_ImageDisplay_Cleanup -- (01.00)

SYNOPSIS

F_Do(Obj,FM_ImageDisplay_Cleanup);

FUNCTION

Send this method to the object before display context changes, or before you want to dispose the object.

SEE ALSO

FM_ImageDisplay_Setup

1.8 FC_ImageDisplay / FM_ImageDisplay_Draw

NAME

FM_ImageDisplay_Draw -- (01.00)

SYNOPSIS

F_DoA(Obj,FM_ImageDisplay_Draw,FS_ImageDisplay_Draw)

F_Do(Obj,FM_ImageDisplay_Draw,FRender *Render,FRect *Rect,ULONG Flags)

FUNCTION

Use this function to draw the image.

INPUTS Render

SEE ALSO