FC_Area

Olivier LAVIALE 2004

COLLABORATORS			
	TITLE : FC_Area		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Olivier LAVIALE 2004	January 13, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FC_Area

1.1 Feelin : FC_Area

FC_Area

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

This is the most important gadget class in Feelin. It is the superclass of almost all the other Feelin gadget classes at one point or another. It holds informations about an object current position, size and weight. It manages fonts, colors, frames (through the FC_FrameDisplay class) and backgrounds (through the FC_ImageDisplay class).

Additionnally, the class handles events and user inputs. By setting an object FA_InputMode, you can make it behave like a button or like a toggle gadget. That's why there is no extra button class. A button is simply a FC_Text object with a raised frame and a FV_InputMode_Release input mode.

Since especially FC_Group is a subclass of FC_Area, you can create rather complex buttons consisting of many other display elements.

METHODS

FM_Setup FM_Cleanup

FM_Show FM_Hide

FM_AskMinMax FM_Layout

FM_Draw FM_Earse

FM_GoActive FM_GoInactive

FM_GoEnabled FM_GoDisabled

FM_HandleEvent FM_ModifyHandler

FM_BuildContextHelp FM_BuildContextMenu

FM_DnDQuery FM_DnDBegin

FM_DnDFinish FM_DnDReport

FM_DnDDrop

ATTRIBUTES

FA_AreaData FA_Box

FA_Left FA_Top

FA_Right FA_Bottom

FA_Width FA_Height

FA_MinWidth FA_MinHeight

FA_MaxWidth FA_MaxHeight

FA_FixWidth FA_FixHeight

FA_FixWidthTxt FA_FixHeightTxt

FA_Fixed FA_FixedWidth

FA_FixedHeight FA_Active

FA_Disabled FA_Selected

FA_Pressed FA_Timer

FA_Hidden FA_Draggable

FA_Dropable FA_Weight

FA_Horizontal FA_Font

FA_ChainToCycle FA_ControlChar

FA_ContextHelp FA_ContextMenu

TYPES

FAreaData FBox

FRect FMinMax

MACROS

_areadata _box

_x _y

_w _h

_x2 _y2

_inner _bl

_br _bt

_bb _mx

_my _mw

_mh _mx2

_my2 _minmax

_minw _minh

_maxw _maxh

_weight _parent

_flags _pens

_font _render

_display _app

_win _rp

1.2 FC_Area / FM_AskMinMax

NAME

FM_AskMinMax -- (00.00) [For use within classes only]

FUNCTION

This method is sent to your object when Feelin wants to know minimun and maximum sizes of your object. These values are needed for the correct layout, depending on the type of group that contains your object.

EXAMPLE

/* Assuming the member "AreaData" of type "FAreaData *" in the LocalObjectData structure of your object */

F_METHOD(ULONG,mAskMinMax) { struct LocalObjectData *LOD = F_LOD(Class,Obj);

/* We add values specific to our object *before* calling our superclass, FC_Area will adjust them depending on FA_MinXxx, FA_MaxXxx or FA_FixedXxx. note that we indeed need to *add* these values, not just set them! */

// x-size depending on objects font

_minw += _font -> tf_XSize * 10; _maxw += _font -> tf_XSize * 20;

// y-size

_minh += _font -> tf_YSize; _maxh += _font -> tf_YSize;

return F_SUPERDO(); }

SEE ALSO

FM_Layout

1.3 FC_Area / FM_BuildContextHelp

NAME

FM_BuildContextHelp -- (02.00)

SYNOPSIS

F_DoA(Obj,FM_BuildContextHelp,struct FS_BuildContextHelp *);

FUNCTION

Allows dynamic creation of context help.

When Feelin is about to show a new context help, it sends a FM_BuildContextHelp to the FC_Window object. The method travels throught the object tree until an object returns the FS_BuildContextHelp structure, previously filled with appropriate information.

When FM_BuildContextHelp reaches FC_Area, it just fill the field Help with the contents of FA_ContextHelp (if any). You needn't care about this method if you only have static, non-changing context helps.

However, if your context helps depend on some internal states of your objects or on the mouse position within your objects, you have to have a subclass which overrides FM_BuildContextHelp, creates a nice string and set it.

INPUTS

A struct FS_BuildContextHelp holding current MouseX and MouseY (relative to window coordinates), and the entry Help which should be filled with the appropriate string.

RESULT

A pointer to the struct FS_BuildContextHelp passed in if you have set the field Help with your string, NULL otherwise.

SEE ALSO

FA_ContextHelp FA_ContextMenu

FC_PopHelp

1.4 FC_Area / FM_BuildContextMenu

NAME

FM_BuildContextMenu -- (02.00)

SYNOPSIS

F_DoA(Obj,FM_BuildContextMenu,struct FS_BuildContextMenu *);

FUNCTION

Allows dynamic creation of context menus.

When Feelin is about to show a new context menu, it sends a FM_BuildContextMenu to the root object of the FC_Window object. The method travels throught the object tree until an object returns the FS_BuildContextMenu structure, previously filled with appropriate information.

When FM_BuildContextMenu reaches FC_Area, it just fill the field Menu with the contents of FA_ContextMenu (if any) and set the field ContextOwner to itself. You needn't care about this method if you only have static, non-changing context menus.

However, if your context menus depend on some internal states of your objects or on the mouse position within your objects, you have to have a subclass which overrides FM_BuildContextMenu, creates a nice Menu object and returns it.

RESULT

Fill the field Menu of the FS_BuildContextMenu structure with a pointer to your FC_Menu object, and the field ContextOwner with a pointer to your object.

The fields Menu and ContextOwner should be left untouch if you failed to create a FC_Menu object. This will allow the FC_Window object to open it's own instead, but please warn the user in this case e.g. using F_LogA(), as he may gets confused...

EXAMPLE

/* This example is directly taken from FC_Area.*/

F_METHODM(APTR,Area_ContextMenu,FS_BuildContextMenu) { struct LocalObjectData *LOD = F_LOD(Class,Obj);

if (LOD -> ContextMenu) { if (between(Msg -> MouseX, _x1, _x2) && between(Msg -> MouseY, _y1, _y2)) { Msg -> Menu = LOD -> ContextMenu; Msg -> ContextOwner = Obj;

return Msg; } }

return NULL; }

NOTE

Feelin will never dispose the object you return. You must take care of this yourself, e.g. by storing a pointer somewhere in your instance data and killing it on the next invocation of FM_BuildContextMenu or on FM_Dispose.

SEE ALSO

FA_ContextHelp FA_ContextMenu

1.5 FC_Area / FM_Cleanup

NAME

FM_Cleanup -- (00.00) [For use within classes only]

FUNCTION

This method is sent to your object when e.g. the window containing your object is closed. You have to free all resources allocated during FM_Setup .

1.6 FC_Area / FM_Draw

NAME

FM_Draw -- (00.00) [For use within classes only]

SYNOPSIS

F_DoA(Obj,FM_Draw,FS_Draw *);

F_Do(Obj,FM_Draw,ULONG Flags);

FUNCTION

Whenever Feelin feels (indeed) that your object should render itself, it sends you a FM_Draw method. This happens e.g. when a window is openend for the first time, after a window was resized.

Together with FM_Draw comes a flag value that indicates which parts of the object are to be redrawn. The only interesting bits in this flag value are FF_Draw_Object and FF_Draw_Update. When FF_Draw_Object is set, Feelin wants you to do a complete redraw of your object. FF_Draw_Update is used by Feelin when the frame doesn't have to be redrawn, only the inner area of the object will be cleared and redrawn. You can use this flags for private requirement with the F_Draw() function call. You can also use one of the FF_Draw_Custom_X flags if you need something special.

A FC_Render object holds information about the rendering evironment (display, application, window, rastport, pens...). You can obtain a pointer to this object from the FAreaData structure of your object. You can also use the _render macro.

NOTE

FM_Draw is the only place where you are allowed to render!

SEE ALSO

F_Draw()

1.7 FC_Area / FM_Erase

NAME

FM_Erase -- (01.00)

SYNOPSIS

F_DoA(Obj,FM_Erase,struct FS_Erase *);

F_Do(Obj,FM_Erase,struct FeelinRect *Rect,ULONG Flags);

FUNCTION

Use this method to draw a specific part of an object's background.

INPUTS Rect

Pointer to a struct FeelinRect describing the area to fill.

Flags

FF_Erase_Region If this flag is set the struct FeelinRect is used as a struct Region (holding multiple rectangles). This region is a standard graphics.library region, created with NewRegion() and setup with appropiate function e.g. OrRectRegion().

Using regions facilities you can fill multiple areas in a single step and also create rather complex rendering and refreshing. Such facilities are used by FC_Group when clearing damaged zones in complex rendering mode.

FF_Erase_Fill This method takes care of rendering context and possibilities e.g. the rectangle or region is not filled if the window is in refresh mode and a same background has already been used to clear the region, unless the flag FF_Erase_Fill is set.

If this flag is set the background is always drawn (even if not necessary).

NOTE

Don't worry about bitmap mapping, FC_ImageDisplay is used to render graphics and is clever enough to adjust patterns by itself and render them nicely.

SEE ALSO

FA_Back

1.8 FC_Area / FM_GoActive

NAME

FM_GoActive -- (00.00) [For use within classes only]

FUNCTION

The methods FM_GoActive / FM_GoInactive are sent to an object when it gets activated / desactivated.

You may not send these methods yourself, they are sent to you by Feelin when your object becomes the active one. If you pass these methods to your superclass, FC_Area will draw the active object frame. If you don't, nothing will be drawn and you're responsible for displaying this state yourself, e.g. with a cursor (as FC_String objects).

SEE ALSO

FA_Active

1.9 FC_Area / FM_GoInactive

NAME

FM_GoInactive -- (00.00) [For use within classes only]

FUNCTION

The methods FM_GoActive / FM_GoInactive are sent to an object when it gets activated / deactivated.

You may not send these methods yourself, they are sent to you by Feelin when your object becomes the active one. If you pass these methods to your superclass, FC_Area will draw the inactive object frame. If you don't, nothing will be drawn and you're responsible for displaying this state yourself, e.g. with a cursor (as string objects).

SEE ALSO

FA_Active

1.10 FC_Area / FM_HandleEvent

NAME

FM_HandleEvent -- (00.00) [For use within classes only]

SYNOPSIS

F_DoA(Obj,FM_HandleEvent,struct FS_HandleEvent *);

F_Do(Obj,FM_HandleEvent,struct FeelinEvent *FEv);

FUNCTION

This method is invoked whenever one of an event handlers trigger signals arrives.

INPUTS

FEv - Pointer to a struct FeelinEvent created within the FM_Application_Run method and sent to the FC_Window object.

RESULT

Return FF_HandleEvent_Eat if the event was for you and you want Feelin to stop calling other event handlers in the chain.

SEE ALSO

FM_ModifyHandler FM_Window_AddEventHandler

1.11 FC_Area / FM_Hide

NAME

FM_Hide -- (00.00) [For use within classes only]

FUNCTION

Your object will receive this method right before the window that contains your object is closed.

SEE ALSO

FM_Show

1.12 FC_Area / FM_Layout

NAME

FM_Layout -- (00.00) [For use within classes only]

FUNCTION

Your object will receive this method after the function F_Layout() has modified its coordinates. This method is curently only implemented by FC_Group and is used to compute and set coordinates of its children.

NOTE

You should rarely implement this method in a class.

1.13 FC_Area / FM_ModifyHandler

NAME

FM_ModifyHandler -- (03.00)

SYNOPSIS

F_Do(obj,FM_ModifyHandler,ULONG nAdd,ULONG nRem);

FUNCTION

Event handlers introduced in FC_Window V6.12 allow custom classes to receive IDCMP events. You need to manage a FeelinEventHandler to receive these events. The FM_ModifyHandler method creates and disposes this precious structure on the fly, depending on the modifications applied to the events requested.

INPUTS

nAdd - IDCMP events to add. nRem - IDCMP events to remove.

RESULT

A pointer to a struct FeelinEventHandler if you have requested some events, or NULL if either modifications set events to NULL or allocation of the structure failed.

EXAMPLE

FM_Setup: F_Do(obj,FM_ModifyHandler,IDCMP_RAWKEY | IDCMP_MOUSEBUTTONS,NULL);

FM_Cleanup: F_Do(obj,FM_ModifyHandler,NULL,IDCMP_RAWKEY | IDCMP_MOUSEBUTTONS);

EXAMPLE

F_METHODM(ULONG,mHandleEvent,FS_HandleEvent) { struct LocalObjectData *LOD = F_LOD(Class,Obj); struct FeelinEvent *FEv = Msg -> FEv;

if (FEv -> Key) { switch (FEv -> Key) { case FV_KEY_LEFT: LOD -> sx = -1; break; case FV_KEY_RIGHT: LOD -> sx = 1; break; case FV_KEY_TOP: LOD -> sy = -1; break; case FV_KEY_BOTTOM: LOD -> sy = 1; break; default: return NULL }

F_Draw(Obj,FF_Draw_Update);

return FF_HandleEvent_Eat; } else { switch (FEv -> Class) { case IDCMP_MOUSEBUTTONS: { if (FEv -> Code == SELECT-DOWN) { if (between(FEv -> MouseX,_x(Obj),_x2(Obj)) && (between(FEv -> MouseY,_y(Obj),_y2(Obj))) { LOD -> x = FEv -> MouseX; LOD -> y = FEv -> MouseY;

F_Draw(Obj,FF_Draw_Update);

/* Only request IDCMP_MOUSEMOVE if we really need it */

F_Do(Obj,FM_ModifyHandler,IDCMP_MOUSEMOVE,0);

return FF_HandleEvent_Eat; } } else { /* reject IDCMP_MOUSEMOVE because lmb is no longer pressed */

F_Do(Obj,FM_ModifyHandler,0,IDCMP_MOUSEMOVE); } break;

case IDCMP_MOUSEMOVE: if (between(FEv -> MouseX,_x(Obj),_x2(Obj)) && (between(FEv -> MouseY,_y(Obj),_y2(Obj))) { LOD -> x = FEv -> MouseX; LOD -> y = FEv -> MouseY;

F_Draw(Obj,FF_Draw_Update);

return FF_HandleEvent_Eat; } break; }

/* Passing FM_HandleEvent to the super class is only necessary if you rely on FC_Area input handling (FA_InputMode). */

} return NULL; }

Take a look at class3 source code to see how things work.

NOTE

FM_ModifyHandler invokes automatically FM_Window_AddEventHandler and FM_Window_RemEventHandler on the window object of the object.

1.14 FC_Area / FM_Setup

NAME

FM_Setup -- (00.00) [For use within classes only]

SYNOPSIS

F_DoA(Obj,FM_Setup,struct FS_Render);

F_Do(obj,FM_Setup,struct FeelinRender *Render);

FUNCTION

Since your object doesn't know anything about display environment after it is created with FM_New, Feelin will send you a FM_Setup when it is about to open a window containing your object.

The first thing you have to do is to pass FM_Setup to your super class an return FALSE on failure. After this, you can calculate some internal data or allocate some display buffers. Return TRUE if everything went ok or FALSE when you discovered any errors. If you return FALSE, the root object of the window containing the object will recieve a FM_Cleanup method.

INPUTS

Render - a pointer to a struct FeelinRender. This structure is created by the window object and shared by all objects of the window. This structure remains valid between FM_Setup / FM_Cleanup.

SEE ALSO

FM_Cleanup

1.15 FC_Area / FM_Show

NAME

FM_Show -- (00.00) [For use within classes only]

FUNCTION

Once the window is opened, your object will receive a FM_Show. If you have some window/rastport environment dependant things to do, FM_Show is the correct place. Intuition like gadgets would for example do an AddGadget() here.

NOTE

Note that you should not render during FM_Show. Usually, Feelin classes won't need to implement this method. And please do not use intuition gadgets but create your own. Feelin purpose is to get rid of intuition so please follow the rule.

SEE ALSO

FM_Hide

1.16 FC_Area / FA_Active

NAME

FA_Active -- (00.00) [ISG], BOOL

FUNCTION

Get and set the active state of an object. If the object is member of the window's cycle chain, this attribute is triggered each time the object become the active one or is no longer the active one. If FA_Active is set to TRUE the FM_GoActive method will be invoked on the object by FC_Area, else the FM_GoInactive method will be invoked instead.

SEE ALSO

FA_Selected FAreaData

1.17 FC_Area / FA_AreaData

NAME

FA_AreaData -- (04.00) [..G], FAreaData *

FUNCTION

Address of the FAreaData structure of the object.

The FAreaData structure is very important to obtain information about the application, the window and the group an object belongs to. It is also very important to know how the object should be rendered...

"_mx(Obj)" is not allowed in Feelin. Any information about the area *MUST* be obtained from the FAreaData structure. Of course, attributes can also be used to obtain everything needed from an object but it's harder to use multiple attributes than a single structure.

Generaly, you obtain the FAreaData pointer within the FM_New method. The pointer is valid during the lifetime of the object, because the FAreaData structure is a local data of the object and not an allocated structure.

Save this pointer in your own local object data (LOD) to use it later. I recommand you to name the field "AreaData", so you'll be able to use macros to easly access FAreaData fields.

EXAMPLE

struct LocalObjectData { FAreaData *AreaData; struct FeelinSignalHandler SignalHandler; };

F_METHOD(ULONG,mNew) { struct LocalObjectData *LOD = F_LOD(Class,Obj);

LOD -> AreaData = (FAreaData *) F_Get(Obj,FA_AreaData);

LOD -> SignalHandler.Object = Obj; LOD -> SignalHandler.Method = FM_Strobo; LOD -> SignalHandler.Flags = FF_SignalHandler_T
LOD -> SignalHandler.fsh_Secs = 0; LOD -> SignalHandler.fsh_Micros = 30000;

return F_SUPERDO(); }

F_METHOD(ULONG,mAskMinMax) { struct LocalObjectData *LOD = F_LOD(Class,Obj);

_minw += 30; _minh += 30;

return F_SUPERDO(); }

F_METHOD(void,mDraw) { struct LocalObjectData *LOD = F_LOD(Class,Obj); struct RastPort *rp = _rp;

F_SUPERDO();

_APen(Rnd(1 << rp -> BitMap -> Depth)); _Boxf(_mx,_my,_mx2,_my2); }

...

SEE ALSO

FAreaData

1.18 FC_Area / FA_Bottom

NAME

FA_Bottom -- (00.00) [..G], ULONG

FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

SEE ALSO

FA_Left FA_Top

FA_Right FA_Width

FA_Height FAreaData

1.19 FC_Area / FA_ChainToCycle

NAME

FA_ChainToCycle -- (01.00) [I..], BOOL

FUNCTION

Feelin GUI can be fully controled with the keyboard. All you have to do to add an object to the window's cycle chain is to set the attribute the TRUE. Setting the attribute to FALSE disabe the automation.

Default: TRUE.

SEE ALSO

FM_Window_ChainAdd FM_Window_ChainRem

1.20 FC_Area / FA_ContextHelp

NAME

FA_ContextHelp -- (02.00) [ISG], STRPTR

FUNCTION

Specifies a context sensitive popup help for the current object. Popup helps are simple string used by a FC_PopHelp object managed by the Window object to as display context help.

Have a look at FC_PopHelp documentation to know how things are handled.

NOTE

Feelin uses the same tree-like technique as always (e.g. with Drag'n'Drop) to find out whichs context help to use on a certain mouse position. This allows you to have a context help for a group and different context helps for its children.

SEE ALSO

FM_BuildContextMenu FA_Menu_Selected

1.21 FC_Area / FA_ContextMenu

NAME

FA_ContextMenu -- (02.00) [ISG], FObject

FUNCTION

Specifies a context sensitive popup menu for the current object. Popup menus are created using FC_Menu and subclasses, like standard Feelin menus.

Whenever the user hits the RMB and the mouse is above the parent object, Feelin will present the popup menu instead of the windows menu.

Note: Feelin will not dispose the FA_ContextMenu object when the object is disposed. You must take care of this object yourself. This is because FC_Menu object of FA_ContextMenu do not actually belong to their parent objects, it's just a reference. You are allowed to use a single FC_Menu object, or even a submenu of a FC_Menu object, as FA_ContextMenu for different objects.

Menus (and submenus) can be shared with objects in the same window, other windows and even application. FC_Menu is very flexible and easy to setup.

NOTE

Feelin uses the same tree-like technique as always (e.g. with Drag'n'Drop) to find out whichs context menu to use on a certain mouse position. This allows you to have a context menu for a group and different context menus for its children.

SEE ALSO

FM_BuildContextMenu FA_Menu_Selected

1.22 FC_Area / FA_ControlChar

NAME

FA_ControlChar -- (01.00) [ISG], CHAR

FUNCTION

This character will be used as a shortcut to your object.

NOTE

FC_Text objects modify this attribute when they find an underscore in their string. You will rarely need to set this attribute yourself.

SEE ALSO

FA_Text

1.23 FC_Area / FA_Disabled

NAME

FA_Disabled -- (01.00) [ISG], BOOL

FUNCTION

Disable or enable an object. Setting this attribute causes an object to become disabled, it gets a ghost pattern and doesn't respond to user input any longer.

Disabled objects cannot be activated with the cycle keys.

Using FA_Disabled on a group of objects will disable all objects within that group.

SEE ALSO

FAreaData

1.24 FC_Area / FA_Font

NAME

FA_Font -- (01.00) [ISG], SPTPTR

SPECIAL INPUTS

FV_Font_Inherit

FUNCTION

Every FC_Area object can have its own font.

The font is loaded through the FM_Application_LoadFont method. See method's documentation to know all possible values.

If the font is modified while the object is displayed a new layout is computed is needed and the object is updated.

NOTE

Getting this attribute returns the struct TextFont * of the opened font.

SEE ALSO

FAreaData

1.25 FC_Area / FA_Height

NAME

FA_Height -- (00.00) [..G], ULONG

FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

SEE ALSO

FA_Left FA_Top

FA_Right FA_Bottom

FA_Width FAreaData

1.26 FC_Area / FA_Hidden

NAME

FA_Hidden -- (00.00) [ISG], BOOL

FUNCTION

Objects with this attribute set are not displayed. You can set FA_Hidden at any time, causing objects to appear and to disappear immediately. A new layout is calculated whenever some objects are shown or hidden. When necessary, Feelin will resize the parent window to make place for the new objects.

SEE ALSO

FAreaData

1.27 FC_Area / FA_Horizontal

NAME

FA_Horizontal -- (00.00) [I.G], BOOL

FUNCTION

A general purpose attribute. It's an easy way telling your object how it has to look like.

This attribute is especially used by groups to make their layout horizontal or vertical, or by sliders to define their orientation.

SEE ALSO

FAreaData

1.28 FC_Area / FA_Left

NAME

FA_Left -- (00.00) [..G], ULONG

FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

SEE ALSO

FA_Left FA_Right

FA_Bottom FA_Width

FA_Height FAreaData

1.29 FC_Area / FA_Pressed

NAME

FA_Pressed -- (00.00) [ISG], BOOL

FUNCTION

Get and set the pressed state of an object. This attribute is triggered by some user action, depending on the input mode:

FV_InputMode_Release: - TRUE when left mouse button is pressed.

- FALSE when left mouse button is released and the mouse is still over the object box (otherwise it will be cleared too, but without triggering a notification event).

FV_InputMode_Immediate - undefined, use FA_Selected for this.

FV_InputMode_Toggle - undefined, use FA_Selected for this.

Waiting for FA_Pressed getting FALSE is the usual way to react on button objects.

NOTE

While an object with a FV_InputMode_Release is pressed the attribute FA_Timer is increased every INTUITICK.

SEE ALSO

FA_Selected FA_Timer

FAreaData

1.30 FC_Area / FA_Right

NAME

FA_Right -- (00.00) [..G], ULONG

FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

SEE ALSO

FA_Left FA_Top

FA_Bottom FA_Width

FA_Height FAreaData

1.31 FC_Area / FA_Selected

NAME

FA_Selected -- (00.00) [ISG], BOOL

FUNCTION

Get and set the selected state of an object. This attribute can be triggered by the user clicking on the object (or using the keyboard), depending on the input mode:

FV_InputMode_Release: - TRUE when left mouse button is pressed. - FALSE when left mouse button is released. - FALSE when the object is selected and the mouse leaves the gadget box. - TRUE when the mouse renters the object box.

FV_InputMode_Immediate: - TRUE when left mouse button is pressed.

FV_InputMode_Toggle: - Toggled when left mouse button is pressed.

Of course you may set this attribute yourself, e.g. to adjust the state of a checkmark object.

A selected object will display its alternative frame and get the alternative background.

SEE ALSO

FA_InputMode FA_Pressed

FA_Timer FAreaData

1.32 FC_Area / FA_Timer

NAME

FA_Timer -- (00.00) [..G], LONG

FUNCTION

FA_Timer gets triggered when a FV_InputMode_Release object is pressed and (after a little delay) increases every INTUITICK as long as the mouse remains over the object.

This makes it possible to have buttons repeatedly cause some actions, just like the arrow objects of a scrollbar.

SEE ALSO

FA_Pressed FA_Selected

1.33 FC_Area / FA_Top

NAME

FA_Top -- (00.00) [..G], ULONG

FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

SEE ALSO

FA_Left FA_Right

FA_Bottom FA_Width

FA_Height FAreaData

1.34 FC_Area / FA_Weight

NAME

FA_Weight -- (00.00) [I..], UWORD

FUNCTION

The weight of an object determines how much room it will get during the layout process. Imagine you have a 100 pixel wide horizontal group with two string objects. Usually, each object will get half of the room and be 50 pixels wide. If you feel the left gadget is more important and should be bigger, you can give it a weight of 200 (and 100 for the right gadget). Because the left gadget is twice as "heavy" as the right gadget, it will become twice as big (about 66 pixel) as the right one (34 pixel).

Of course giving weights only makes sense if the object is resizable. An object with a weight of 0 will always stay at its minimum size.

By default, all objects have a weight of 100.

SEE ALSO

FAreaData

1.35 FC_Area / FA_Width

NAME

FA_Width -- (00.00) [..G], ULONG

FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

SEE ALSO

FA_Left FA_Top

FA_Right FA_Bottom

FA_Height FAreaData

1.36 FC_Area / FAreaData

NAME

FAreaData -- (04.00)

STRUCT

struct FeelinAreaData { FObject Parent; FObject Next; FObject Prev;

FRender *Render; ULONG Flags;

FBox Box; FInner Inner; FMinMax MinMax;

ULONG *Pens; struct TextFont *Font; UWORD Weight; };

FUNCTION

This structure holds precious information such as the application, the window and the group the object belongs to, and general layout and graphic information.

Because FC_Object uses an opaque structure to keep as much independence as possible from its subclasses, its not a good idea (not to say dangerous) to make any assumption about the offset from the start of the object to this local data. In order to obtain a pointer to the FAreaData structure you *MUST* use the FA_AreaData attribute.

Using "_app(Obj)" like MUI does is not possible with Feelin. You may complain about this but since 'Obj' may refers to nearly anything, it's a security gap that Feelin doesn't permit.

FIELDS Parent (FObject)

Parent of this object. Usualy the parent is a sub-class object of FC_Group}, but if the object is the root object of a window, its parent will be the window itself. You can also obtain the parent of an object with the FA_Parent attribute.

Next, Prev (FObject)

These fields are used by FC_Group to link objects one to another.

Render (FRender *)

The shared FC_Render object used by every object in a window.

Flags (ULONG)

GENERAL LAYOUT FLAGS

FF_Horizontal - If set, the layout of the object should be horizontal, otherwise the object should have a vertical layout. This flag reflects the FA_Horizontal attribute.

FF_Area_SetMinW, FF_Area_SetMinH - When these flags are set, the minimum dimensions should be set in the FMinMax structure during the FM_AskMinMax method.

FF_Area_SetMaxW, FF_Area_SetMaxH - When these flags are set, the maximum dimensions should be set in the FMinMax structure during the FM_AskMinMax method.

GENERAL STATE FLAGS

FF_Area_Selected - If set, the object is currently selected (e.g. a button pressed down, or a check-box checked). This flag should be used to render the "selected" state. It reflects the FA_Selected attribute.

FF_Area_Pressed - If set, the object is currently pressed (e.g. a button pressed down). This flag should not be used to render any state, its only goal is to triggers the FA_Pressed attribute.

FF_Area_Active - If set, the object is currently active (e.g. an object in the cycle chain, a FC_String object). This flag can be read if you handle the active and inactive state of an object (and the rendering of these states). It reflects the FA_Active attribute.

FF_Area_Disabled - If set, the object is currently disabled (e.g. an object ghosted which receives no inputs). This flag can be read if you handle the disabled and enabled state of an object (and the rendering of these states). It reflects the FA_Disabled attribute.

GENERAL RENDER FLAGS

FF_Area_CanShow - If set, the object can be shown and participate in the layout. Only "shown" object can be drawn and receive inputs. This flag reflect the FA_Hidden attribute.

FF_Area_CanDraw - If set, the object can be drawn. This flag is only set when all conditions are meet for the object to actualy be drawn : object has been setuped successfully, object can be shown, object is currently within a displayable region, rendering is not forbidden.

Box (FBox)

This FBox structure holds the coordinates of the object. These coordinates are only valid between the FM_Show and FM_Hide methods.

Inner (FInner)

This FInner structure holds the spacing dimensions between the object itself and its frame.

MinMax (FMinMax)

This FMinMax structure holds the minimum and maximum dimensions of the object. These dimensions are computed during the FM_AskMinMax method and are only valid between the FM_Show and FM_Hide methods.

Pens (ULONG *)

An array of ULONG values describing which pens should be used to render the object. This pointer may change as the state of the object changes e.g. if the user has defined soften colors for the "disabled" state.

Font (struct TextFont *

Text font used by this object. Text rendered in this object should use this font. The font can be modified with the FA_Font attribute. Note that the field is a pointer to a struct TextFont (the opened font structure).

Weight (UWORD)

Weight of the object. This field has no general use, but can be used to create balancing stuff (it is actualy used by FC_Balance). This field is usualy set at initialisation time with the FA_Weight attribute.

1.37 FC_Area / FBox

NAME

FBox -- (00.00)

STRUCT

struct FeelinBox { WORD x,y; UWORD w,h; };

FUNCTION

This structure holds the coordinates and dimensions of an object.

FIELDS x,y

Coordinates of an object.

w,h

Dimensions of an object.

1.38 FC_Area / FRect

NAME

FRect -- (00.00)

STRUCT

struct FeelinRect { WORD x1,y1; WORD x2,y2; };

FUNCTION

This structure holds the coordinates of an object.

FIELDS x1,y1

Up corner left of an object.

w,h

Down corner right of an object.

NOTE

This structure is often used to represent a region.

1.39 FC_Area / FMinMax

NAME

FMinMax -- (00.00)

STRUCT

struct FeelinMinMax { UWORD MinW,MinH; UWORD MaxW,MaxH; };

FUNCTION

This structure holds the minimum and maximum dimensions of an object.

FIELDS MinW,MinH

Minimum width and height of an object.

w,h

Maximum width and height of an object.

1.40 FC_Area / _areadata

NAME

_areadata -- (04.00)

DEFINE

#define _areadata ((FAreaData *)(LOD -> AreaData))

FUNCTION

Use this macro to obtain the pointer of the FAreaData structure stored in the LocalObjectData structure of your object.

This macro needs the variable 'LOD' to be defined, with the member 'AreaData' of type FAreaData . All other FC_Area macros depend on the _areadata macro. You can easily redefine the macro to your needs and use the other ones without modifing them.

EXAMPLE

struct LocalObjectData { FAreaData *AreaData; ... };

F_METHOD(ULONG,mNew) { struct LocalObjectData *LOD = F_LOD(Class,Obj);

_areadata = (FAreaData *) F_Get(Obj,FA_AreaData);

...

return F_SUPERDO(); }

F_METHOD(ULONG,mAskMinMax) { struct LocalObjectData *LOD = F_LOD(Class,Obj);

_minw += 10; _maxw += 100; _minh += 10; _maxh += 100;

return F_SUPEROD(); }

SEE ALSO

FA_AreaData

1.41 FC_Area / _render

NAME

_render -- (00.00)

DEFINE

#define _render (_areadata -> Render)

FUNCTION

Use this macro to obtain the pointer of the FC_Render object stored in the FAreaData structure of your object.

SEE ALSO

_app _display

_rp _win

1.42 FC_Area / _parent

NAME

_parent -- (00.00)

DEFINE

#define _parent (_areadata -> Parent)

FUNCTION

Use this macro to obtain the parent of an object.

Usualy the parent is a sub-class object of FC_Group}, but if the object is the root object of a window, its parent will be the window itself. You can also obtain the parent of an object with the FA_Parent attribute.

SEE ALSO

_app _win

1.43 FC_Area / _flags

NAME

_flags -- (00.00)

DEFINE

#define _flags (_areadata -> Flags)

FUNCTION

Use this macro to read the flags of an object.

Please, *do not* modify the flags !

GENERAL LAYOUT FLAGS

FF_Horizontal - If set, the layout of the object should be horizontal, otherwise the object should have a vertical layout. This flag reflects the FA_Horizontal attribute.

FF_Area_SetMinW, FF_Area_SetMinH - When these flags are set, the minimum dimensions should be set in the FMinMax structure during the FM_AskMinMax method.

FF_Area_SetMaxW, FF_Area_SetMaxH - When these flags are set, the maximum dimensions should be set in the FMinMax structure during the FM_AskMinMax method.

GENERAL STATE FLAGS

FF_Area_Selected - If set, the object is currently selected (e.g. a button pressed down, or a check-box checked). This flag should be used to render the "selected" state. It reflects the FA_Selected attribute.

FF_Area_Pressed - If set, the object is currently pressed (e.g. a button pressed down). This flag should not be used to render any state, its only goal is to triggers the FA_Pressed attribute.

FF_Area_Active - If set, the object is currently active (e.g. an object in the cycle chain, a FC_String object). This flag can be read if you handle the active and inactive state of an object (and the rendering of these states). It reflects the FA_Active attribute.

FF_Area_Disabled - If set, the object is currently disabled (e.g. an object ghosted which receives no inputs). This flag can be read if you handle the disabled and enabled state of an object (and the rendering of these states). It reflects the FA_Disabled attribute.

GENERAL RENDER FLAGS

FF_Area_CanShow - If set, the object can be shown and participate in the layout. Only "shown" object can be drawn and receive inputs. This flag reflect the FA_Hidden attribute.

FF_Area_CanDraw - If set, the object can be drawn. This flag is only set when all conditions are meet for the object to actualy be drawn : object has been setuped successfully, object can be shown, object is currently within a displayable region, rendering is not forbidden.

1.44 FC_Area / _box

NAME

_box -- (00.00)

DEFINE

#define _box (_areadata -> Box)

FUNCTION

Use this macro to access the FBox structure holding the coordinates and dimensions of an object.

SEE ALSO

_x _y

_w _h

1.45 FC_Area / _x

NAME

_x -- (00.00)

DEFINE

#define _x (_box .x)

FUNCTION

Use this macro to obtain the left edge coordinate of an object.

SEE ALSO

_y _w

_h _box

_mx

1.46 FC_Area / _y

NAME

_y -- (00.00)

DEFINE

#define _y (_box .y)

FUNCTION

Use this macro to obtain the top edge coordinate of an object.

SEE ALSO

_x _w

_h _box

_my

1.47 FC_Area / _w

NAME

_w -- (00.00)

DEFINE

#define _w (_box .w)

FUNCTION

Use this macro to obtain the width of an object.

SEE ALSO

_x _y

_h _box

_mw

1.48 FC_Area / _h

NAME

_h -- (00.00)

DEFINE

#define _h (_box .h)

FUNCTION

Use this macro to obtain the height of an object.

SEE ALSO

_x _y

_w _box

_mh

1.49 FC_Area / _x2

NAME

_x2 -- (00.00)

DEFINE

#define _x2 (_x + _w - 1)

FUNCTION

Use this macro to obtain the right edge coordinate of an object.

SEE ALSO

_mx2 _y2

1.50 FC_Area / _y2

NAME

_y2 -- (00.00)

DEFINE

#define _y2 (_y + _h - 1)

FUNCTION

Use this macro to obtain the top edge coordinate of an object.

SEE ALSO

_my2 _x2

1.51 FC_Area / _inner

NAME

_inner -- (00.00)

DEFINE

#define _inner (_areadata -> Inner)

FUNCTION

Use this macro to access the FInner structure holding inner dimensions (offset between the frame's border an the object itself).

SEE ALSO

_bl _br

_bt _bb

1.52 FC_Area / _bl

NAME

_bl -- (00.00)

DEFINE

#define _bl (_inner .l)

FUNCTION

Use this macro to obtain the dimension of the left space between the frame's border and the actual itself.

SEE ALSO

_br _bt

_bb

1.53 FC_Area / _br

NAME

_br -- (00.00)

DEFINE

#define _br (_inner .r)

FUNCTION

Use this macro to obtain the dimension of the right space between the frame's border and the actual itself.

SEE ALSO

_bl _bt

_bb

1.54 FC_Area / _bt

NAME

_bt -- (00.00)

DEFINE

#define _bt (_inner .t)

FUNCTION

Use this macro to obtain the dimension of the top space between the frame's border and the actual itself.

SEE ALSO

_bl _br

_bb

1.55 FC_Area / _bb

NAME

_bb -- (00.00)

DEFINE

#define _bb (_inner .b)

FUNCTION

Use this macro to obtain the dimension of the bottom space between the frame's border and the actual itself.

SEE ALSO

_bl _br

_bt

1.56 FC_Area / _minmax

NAME

_minmax -- (00.00)

DEFINE

#define _minmax (_areadata -> MinMax)

FUNCTION

Use this macro to access the FMinMax structure holding the minimum and maximum dimensions of an object.

SEE ALSO

_minw _minh

_maxw _maxh

1.57 FC_Area / _minw

NAME

_minw -- (00.00)

DEFINE

#define _minw (_minmax .MinW)

FUNCTION

Use this macro to obtain the minimum width of an object.

SEE ALSO

_minh _maxw

_maxh

1.58 FC_Area / _minh

NAME

_minh -- (00.00)

DEFINE

#define _minh (_minmax .MinH)

FUNCTION

Use this macro to obtain the minimum height of an object.

SEE ALSO

_minw _maxw

_maxh

1.59 FC_Area / _maxw

NAME

_maxw -- (00.00)

DEFINE

#define _maxw (_minmax .MaxW)

FUNCTION

Use this macro to obtain the maximum width of an object.

SEE ALSO

_minw _minh

_maxh

1.60 FC_Area / _maxh

NAME

_maxh -- (00.00)

DEFINE

#define _maxh (_minmax .MaxH)

FUNCTION

Use this macro to obtain the maximum height of an object.

SEE ALSO

_minw _minh

_maxw

1.61 FC_Area / _pens

NAME

_pens -- (00.00)

DEFINE

#define _pens (_areadata -> Pens)

FUNCTION

Use this macro to obtain an array of ULONG values describing which pens should be used to render the object. This pointer may change as the state of the object changes e.g. if the user has defined different colors for the "disabled" state.

SEE ALSO

FA_ColorScheme

1.62 FC_Area / _font

NAME

_font -- (00.00)

DEFINE

#define _font (_areadata -> Font)

FUNCTION

Use this macro to obtain the Text font used by this object. Text rendered in this object should use this font. The font can be modified with the FA_Font attribute. Note that the field is a pointer to a struct TextFont (the opened font structure).

SEE ALSO

FA_Font

1.63 FC_Area / _weight

NAME

_weight -- (00.00)

DEFINE

#define _weight (_areadata -> Weight)

FUNCTION

Use this macro to obtain the weight of an object. This field has no general use, but can be used to create balancing stuff (it is actualy used by FC_Balance). This field is usualy set at initialisation time with the FA_Weight attribute.

1.64 FC_Area / _mx

NAME

_mx -- (00.00)

DEFINE

#define _mx (_x + _bl)

FUNCTION

Use this macro to obtain the left edge of the object itself, not its frame, like _x does.

SEE ALSO

_my _mw

_mh _mx2

1.65 FC_Area / _my

NAME

_my -- (00.00)

DEFINE

#define _my (_y + _bt)

FUNCTION

Use this macro to obtain the top edge of the object itself, not its frame, like _y does.

SEE ALSO

_mx _mw

_mh _my2

1.66 FC_Area / _mw

NAME

_mw -- (00.00)

DEFINE

#define _mw (_w - _bl - _br)

FUNCTION

Use this macro to obtain the width of the object itself, exluding its frame.

SEE ALSO

_mx _my

_mh

1.67 FC_Area / _mh

NAME

_mh -- (00.00)

DEFINE

#define _mh (_h - _bt - _bb)

FUNCTION

Use this macro to obtain the height of the object itself, exluding its frame.

SEE ALSO

_mx _my

_mw

1.68 FC_Area / _mx2

NAME

_mx2 -- (00.00)

DEFINE

#define _mx2 (_x2 - _br)

FUNCTION

Use this macro to obtain the right edge of the object itself, not its frame, like _x2 does.

SEE ALSO

_my2

1.69 FC_Area / _my2

NAME

_my2 -- (00.00)

DEFINE

#define _my2 (_y2 - _bb)

FUNCTION

Use this macro to obtain the bottom edge of the object itself, not its frame, like _y2 does.

SEE ALSO

_mx2

1.70 FC_Area / _display

NAME

_display -- (00.00)

DEFINE

#define _display (_render -> Display)

FUNCTION

Use this macro to obtain a pointer to the FC_Display object managing the display context in which the window is opened.

NOTE

This macro assumes that _render is valid (the object is ready).

SEE ALSO

_app _rp

_win

1.71 FC_Area / _app

NAME

_app -- (00.00)

DEFINE

#define _app (_render -> Application)

FUNCTION

Use this macro to obtain a pointer to the application an object belongs to.

NOTE

This macro assumes that _render is valid (the object is ready).

SEE ALSO

_display _rp

_win

1.72 FC_Area / _win

NAME

_win -- (00.00)

DEFINE

#define _win (_render -> Window)

FUNCTION

Use this macro to obtain a pointer to the FC_Window object an object belongs to.

NOTE

This macro assumes that _render is valid (the object is ready).

SEE ALSO

_app _display

_rp

1.73 FC_Area / _rp

NAME

_rp -- (00.00)

DEFINE

#define _rp (_render -> RPort)

FUNCTION

Use this macro to obtain a pointer to the RastPort in which objects should be rendered. This RastPort may be different than the window's RastPort.

NOTE

This macro assumes that _render is valid (the object is ready).

SEE ALSO

_app _display

_win