FC_Window

Olivier LAVIALE 2004

COLLABORATORS

	TITLE : FC_Window	

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

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FC_Window

1.1 Feelin : FC_Window

FC_Window

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

This class generates windows and supply a place where Feelin gadgets feel (indeed :) well. It handles the complicated task of window resizing fully automatic, you don't need to worry about that.

Windows are children of an application, you cannot use a FC_Window object without having a parent FC_Application object. On the other side, the gadgets in a window are children of the window, you cannot use gadgets without having a parent window.

Creating a window does not mean to open it instantly. This is done later by setting the window's FA_Window_Open attribute. If your application has several windows, the usual way is to create them all at once at startup time and open/close it later just by setting FA_Window_Open .

There is no difference in talking to gadgets whether their parent window is open or not. If you e.g. set the contents of a string gadget in an open window, the gadget will refresh immediately. If the window is closed, the gadget just remembers its new setting and displays it later.

Window look is not restricted to standard intuition look. Windows can be dressed to your taste by decorators without any restriction.

METHODS

FM_Window_Setup FM_Window_Cleanup

FM_Window_Open FM_Window_Close

FM_Window_ChainAdd FM_Window_ChainRem

FM_Window_HandleEvent FM_Window_AddEventHandler

FM_Window_RemEventHandler FM_Window_Layout

FM_Window_Draw

ATTRIBUTES

FA_Window FA_Window_Open

FA_Window_Activate FA_Window_ActiveObject

FA_Window_CloseRequest FA_Window_Backdrop

FA_Window_Borderless FA_Window_Resizable

FA_Window_GadClose FA_Window_GadDepth

FA_Window_GadDragbar FA_Window_GadIconify

FA_Window_Title FA_Window_ScreenTitle

TYPES

FeelinEventHandler

1.2 FC_Window / FM_Window_AddEventHandler

NAME

FM_Window_AddEventHandler -- (06.12)

SYNOPSIS

F_Do(Obj,FM_Window_AddEventHandler,struct FeelinEventHandler *feh);

FUNCTION

Event handlers introduced in FC_Window v6.12 allow classes to receive IDCMP events. You create a struct FeelinEventHandler (preferably located somewhere in the instance data of your custom class) with the type of events you wish to receive and then call FM_Window_AddEventHandler to add your node to a window's event handler queue.

Subclasses of FC_Area may use the FM_ModifyHandler method that provides an easy interface to the FM_Window_AddEventHandler and FM_Window_RemEventHandler methods.

Whenever an input event arrives, FC_Window iterates through the list of event handlers. The class of the input event is matched against the event handlers class field and in case of a match, the method FM_HandleEvent is invoked on the specified object.

Each event handler may decide to "eat" the input event, this means FC_Window will abort iterating the handler queue after calling this handler.

Feelin follows certain priority rules when iterating through the handler queue of a window. If there are handler entries for the active object (FA_Window_ActiveObject) its node is checked before all other nodes. If there was no active object or if it didn't ate the event, the rest of the handler queue is checked according to their 'Priority' field.

A good place to add/remove event handlers is the FM_Setup/FM_Cleanup method pair of your custom class.

If you wish to change certain fields (e.g. Events) in an handler (currently added or not to a windows), you must deactivate (remove) the handler, make your changes and then add it again. Do not change fields of an active handler without removing it first !

INPUTS

feh - Pointer to an initialized struct FeelinEventHandler .

Since every class should add a handler itself if it needs some input, FM_HandleEvent methods are different from other Feelin methods which usually travel from class to class in an objects class tree. FM_HandleEvent is more treated like a hook function instead of a method: in almost all cases you will want it to be passed directly to a specific class dispatcher (by setting 'Class'). Also, this dispatcher will usually not pass the method to its super class but instead return directly.

RESULT

FM_Window_AddEventHandler cannot fail, the result value of the method is currently undefined.

NOTE

You must match each FM_Window_AddEventHandler with exactly one FM_Window_RemEventHandler method.

EXAMPLE

FM_Setup: LOD -> Handler.Object = Obj; LOD -> Handler.Class = Class; LOD -> Handler.Events = IDCMP_MOUSEBUTTONS; F_Do(_win(Obj),FM_Window_AddEventHandler,&LOD -> Handler);

FM_Cleanup: F_Do(_win(Obj),FM_Window_RemEventHandler,&LOD -> Handler);

Changing the trigger events of an active handler:

F_Do(_win(Obj),FM_Window_RemEventHandler,&LOD -> Handler); LOD -> Handler.Events |= IDCMP_MOUSEMOVE; F_Do(_win(Obj),FM_Window_AddEventHandler,&LOD -> Handler);

More examples can be found in the Feelin demo source codes that deal with custom classes.

SEE ASLO

FM_ModifyHandler FM_Window_RemEventHandler

1.3 FC_Window / FM_Window_RemEventHandler

NAME

FM_Window_RemEventHandler -- (06.12)

SYNOPSIS

F_Do(Obj,FM_Window_RemEventHandler,struct FeelinEventHandler *feh);

FUNCTION

Remove an event handler.

RESULT

FM_Window_RemEventHandler cannot fail, the result value of the method is currently undefined.

INPUTS

feh - event handler node structure you passed to FM_Window_AddEventHandler previously.

NOTE

Every call to FM_Window_AddEventHandler must be matched by exactly one call to FM_Window_RemEventHandler.

SEE ASLO

FM_ModifyHandler FM_HandleEvent

1.4 FC_Window / FA_Window

NAME

FA_Window -- (01.00) [..G], struct Window *

FUNCTION

When your window is open, you can obtain a pointer to the intuition Window structure with this tag.

Since the user can close your window at any time (e.g. iconification), you must be prepared to receive a NULL pointer as result.

1.5 FC_Window / FA_Window_Activate

NAME

FA_Window_Activate -- (01.00) [ISG], BOOL

FUNCTION

Setting this to TRUE will activate the window. Setting this to FALSE has no effect. The attribute will change whenever the user activates/deactivates the window.

Specifying FALSE at object creation time will make the window open in an inactive state.

Default to TRUE.

1.6 FC_Window / FA_Window_ActiveObject

NAME

FA_Window_ActiveObject -- (01.00) [.SG], APTR

SPECIAL INPUTS

FV_Window_ActiveObject_None FV_Window_ActiveObject_Next FV_Window_ActiveObject_Prev

FUNCTION

Set the active object in a window as if the user would have activated it with keyboard.

EXAMPLE

F_Set(window,FA_Window_ActiveObject,okaybutton);

1.7 FC_Window / FA_Window_Backdrop

NAME

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

FUNCTION

Make the window a backdrop window.

Default to FALSE.

1.8 FC_Window / FA_Window_Borderless

NAME

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

FUNCTION

Make the window borderless.

Default to FALSE.

1.9 FC_Window / FA_Window_CloseRequest

NAME

FA_Window_CloseRequest -- (00.00) [..G], BOOL

FUNCTION

When the user hits a windows close gadget, the window isn't closed immediately. Instead Feelin only sets this attribute to TRUE to allow your application to react.

Usually, you will setup a notification that automatically closes the window when a close request appears, but you could e.g. pop up a confirmation requester or do some other things first.

EXAMPLE

/* automagically close a window when the close gadget is pressed */

F_Do(window,FM_Notify, FA_Window_CloseRequest,TRUE, window, FM_Set,2,FA_Window_Open,FALSE);

SEE ASLO

FA_Window_Open

1.10 FC_Window / FA_Window_GadClose

NAME

FA_Window_GadClose -- (01.00) [I.G], BOOL

FUNCTION

Set this to FALSE and your window will not have a close gadget.

Default: TRUE

SEE ALSO

FA_Window_GadNone FA_Window_GadDragbar

FA_Window_GadDepth FA_Window_GadIconify

FA_Window_Resizable

1.11 FC_Window / FA_Window_GadDepth

NAME

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

FUNCTION

Enable or disable the depth gadget. There is no good reason to use this tag.

Default: TRUE

SEE ALSO

FA_Window_GadNone FA_Window_GadDragbar

FA_Window_GadClose FA_Window_GadIconify

FA_Window_Resizable

1.12 FC_Window / FA_Window_GadDragbar

NAME

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

FUNCTION

Tell Feelin to give your window a dragbar. There is no good reason to disable the dragbar!

Default: TRUE

SEE ALSO

FA_Window_GadNone FA_Window_GadClose

FA_Window_GadDepth FA_Window_GadIconify

FA_Window_Resizable

1.13 FC_Window / FA_Window_GadIconify

NAME

FA_Window_GadIconify -- (01.00) [I.G], BOOL

FUNCTION

Set this to FALSE and your window will not have an iconify gadget.

The iconify gadget is not available with standard intuition windows, but decorators can use this attribute to create (TRUE) or not (FALSE) an iconify gadget.

Default: TRUE

SEE ALSO

FA_Window_GadNone FA_Window_GadDragbar

FA_Window_GadDepth FA_Window_Resizable

1.14 FC_Window / FA_Window_GadNone

NAME

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

FUNCTION

Disable all window's gadgets. The window will only have a small border, that can be disabled with the FA_Window_Borderless attribute.

This attribute has no negative form e.i. you can disable all gadgets by setting the attribute to TRUE, but setting the attribute to FALSE does not enable them.

SEE ALSO

FA_Window_GadDragbar FA_Window_GadClose

FA_Window_GadDepth FA_Window_GadIconify

FA_Window_Resizable

1.15 FC_Window / FA_Window_Open

NAME

FA_Window_Open -- (00.00) [ISG], BOOL

FUNCTION

Use this attribute to open (TRUE) or close (FALSE) a window.

You can set this attribute to TRUE in the initial taglist (when creating the object). This is alright because the attribute is more a wish than an order.

It depends much on the application's state (awake, asleep, running or not). It's up to the application to open windows or not (the application may start in a hidden state). The attribute only toggle the state of the window leaving full control to the application. Once the application is running and active (not iconified), setting this attribute have immediate results.

When a window is closed, the object remembers its box (position and size). If the window opens again it gets back its previous box (if possible). If the object has a FA_ID its box is saved (FM_Export) when the application is shutdown, and loaded (FM_Import) when the application is launched again.

NOTE

To know if a window is opened use the FA_Window attribute because FA_Window_Open doesn't reflect reality, but a wish.

SEE ALSO

FM_Application_Run

1.16 FC_Window / FA_Window_Resizable

NAME

FA_Window_Resizable -- (01.00) [..G], BOOL

FUNCTION

Zoom and Size gadgets cannot be requested. These gadgets only appear if a window can be resized. The FA_Window_Resizable attribute can be used by decorators to know if a window can be rezised (to create zoom and size gadgets).

SEE ALSO

FA_Window_GadNone FA_Window_GadDragbar

FA_Window_GadClose FA_Window_GadDepth

FA_Window_GadIconify

1.17 FC_Window / FA_Window_ScreenTitle

NAME

FA_Window_ScreenTitle -- (00.00) [ISG], PTR TO CHAR

FUNCTION

This text will appear in the screens title bar when the window is active.

SEE ASLO

FA_Window_Title

1.18 FC_Window / FA_Window_Title

NAME

FA_Window_Title -- (00.00) [ISG], PTR TO CHAR

FUNCTION

Specify the title of a window.

SEE ASLO

FA_Window_ScreenTitle

1.19 FC_Window / FeelinEventHandler

NAME

FeelinEventHandler -- (06.12)

STRUCT

struct FeelinEventHandler { struct FeelinEventHandler *Next; struct FeelinEventHandler *Prev;

UWORD Flags; BYTE Priority; UBYTE reserved; ULONG Events; FObject Object; struct FeelinClass *Class; };

FUNCTION

Event handlers introduced in FC_Window v6.12 allow classes to receive IDCMP events. You create a struct FeelinEventHandler (preferably located somewhere in your local object data) with the type of events you wish to receive and then call FM_Window_AddEventHandler to add your node to a window's event handler queue.

Whenever an input event arrives, FC_Window iterates through the list of event handlers. The class of the input event is matched against the event handlers class field and in case of a match, the method FM_HandleEvent is invoked on the specified object.

FIELDS Next, Prev

These fields are used by the FC_Window object to link handlers, you don't have to bother with them.

Flags

There is currently no flag defined yet, always set 0 here.

Priority

Event handlers are inserted according to their priority. 0 is fine in almost all cases.

Events

Exclusive OR of all IDCMP_Xxx flags you want this handler to react on.

Object

Object which should receive the FM_HandleEvent method. Insert a pointer to yourself here.

* Class

If you point this to your class, Feelin will invoke FM_HandleEvent with F_ClassDoA(Class,Obj,...) instead as with F_DoA(Obj,...). This will bypass any subclasses you might have and directly hand the input event to the dispatcher. If Class is NULL, FM_HandleEvent is sent like any other method to the true class of your object.

Since every class should add a handler itself if it needs some input, FM_HandleEvent methods are different from other Feelin methods which usually travel from class to class in an objects class tree. FM_HandleEvent is more treated like a hook function instead of a method: in almost all cases you will want it to be passed directly to a specific class dispatcher (by setting Class). Also, this dispatcher will usually not pass the method to its super class but instead return directly.

SEE ALSO

FM_ModifyHandler