Subject: : Official AmigaOS4 feedback
Topic: : Public screen closing bug in OS4.1 (FIXED)
Re: Public screen closing bug in OS4.1
Author: : ChrisH
Date: : 2011/6/16 12:06:48
URL:

@Chris
That does not appear to be the case. At least the SDK says this:

Quote:

NAME
LockPubScreen -- Prevent a public screen from closing. (V36)

SYNOPSIS
struct Screen *screen = LockPubScreen(CONST_STRPTR name);

FUNCTION
Prevents a public screen (or the Workbench) from closing
while you examine it in preparation of opening a visitor window.

The sequence you use to open a visitor window that needs to examine fields in the screen it is about to open on is:
LockPubScreen()
... examine fields ...
OpenWindow() on public screen
UnlockPubScreen()
... use your window ...
CloseWindow()

NOTE
You needn't hold the "pubscreen lock" for the duration that
your window is opened. **LockPubScreen() basically has the same effect as an open visitor window: it prevents the screen from being closed.**

If you pass the string "Workbench" or you pass NULL and there
is no default public screen, the Workbench screen will
be automatically opened if it is not already present.

INPUTS
name - name string for public screen or NULL for default public
screen. The string "Workbench" indicates the Workbench

screen.

RESULT
screen - **Returns pointer to a screen if successful or NULL.
The call can fail for reasons including that the named
public screen doesn't exist or is in private state.**

Quote:

NAME
CloseWindow -- Close an Intuition window.

SYNOPSIS
CloseWindow(Window)

VOID CloseWindow(struct Window *);

FUNCTION
Closes an Intuition window. Unlinks it from the system, deallocates
its memory, and makes it disappear.

...

**New for V36: If your window is a "Visitor Window" (see OpenWindow)
CloseWindow will decrement the "visitor count" in the public screen
on which the window was open. When the last visitor window is
closed, a signal will be sent to the public screen task, if this
was pre-arranged (see OpenScreen).**

Quote:

NAME
OpenWindow -- Open an Intuition window.

SYNOPSIS
Window = OpenWindow(NewWindow)

struct Window *OpenWindow(struct NewWindow *);

FUNCTION

...

**WA_PubScreenName - This tag item declares that you want your window
to open as a visitor window on the public screen whose name
is pointed to by (STRPTR) ti_Data.**

WA_PubScreen - Open as a visitor window on the public screen whose address if provided as (struct Screen *) ti_Data. To ensure that this screen remains open long enough, you must either:
1) Be the screen's owner
2) have another window already open on the screen
3) use LockPubScreen()
Using exec.library/Forbid() is not sufficient.

You can provide ti_Data to be NULL (zero), without any of the above precautions, to specify the default public screen.

WA_PubScreenFallBack - This Boolean attribute specifies that a visitor window should "fall back" to opening on the default public screen if the explicitly specify public screen is not available.

...

NOTES
Regarding Public Screens, you can specify a window to be a "visitor window" on a public screen in one of several ways. In each case, you must be sure not to specify a NewWindow type of CUSTOMSCREEN. You should use the value PUBLICSCREEN.

There are actually several ways you can specify which screen you want a visitor window to be opened on:

1) Specify the name of the public screen WA_PubScreenName, or a NULL pointer, in ti_Data. The name might have been provided by the user. A NULL pointer means to use the default public screen.

If the named screen cannot be found, the default public screen will be used if the Boolean attribute WA_PubScreenFallBack is TRUE.

2) Specify a pointer to a public screen using the WA_PubScreen tag item. The WA_PubScreenFallBack attribute has no effect. You can specify the default public screen by providing a NULL pointer.

You can also specify the pointer by setting NewWindow.Type to PUBLICSCREEN, and specifying the public screen pointer in NewWindow.Screen. The WA_PubScreen tag item has precedent over this technique.

Unless NULL, the screen pointer provided MUST be a valid public screen. You may ensure this several ways:

- Be the owner of the screen.

- Have a window already open on the screen.
- Use LockPubScreen() to prevent the screen from closing.
- specifying the WFLG_VISITOR bit in NewWindow.Flags is not supported.

It is anticipated that the last will be the most common method of opening public screens because you often want to examine properties of the screen your window will be using in order to compensate for differences in dimension, depth, and font.

The standard sequence for this method is as follows:
LockPubScreen() - obtain a pointer and a promise
layout window - adapt your window to the screen you will use
OpenWindow() - using the pointer you specify
UnlockPubScreen() - once your window is open, you can let go of the lock on the public screen
... normal window even processing ...
CloseWindow().

Quote:

NAME
OpenScreen -- Open an Intuition screen.

SYNOPSIS
Screen = OpenScreen(NewScreen)

struct Screen *OpenScreen(struct NewScreen *);

or

struct Screen *OpenScreen(struct ExtNewScreen *);

FUNCTION

...

The following two tag items specify the task and signal to be issued to notify when the last "visitor" window closes on a public screen. This support is to assist envisioned public screen manager programs.

SA_PubTask: Task to be signalled. If absent (and SA_PubSig is valid), use the task which called OpenScreen() or OpenScreenTagList()).

SA_PubSig: Data is a UBYTE signal number (not flag) used to notify a task when the last visitor window closes on a public screen.[b]

...

[b]A_ErrorCode: ti_Data points to a ULONG in which Intuition will stick an extended error code if OpenScreen[TagList]() fails.
Values are of this include 0, for success, and:
OSERR_NOMONITOR - monitor for display mode not available.
OSERR_NOCHIPS - you need newer custom chips for display mode.
OSERR_NOMEM - couldn't get normal memory
OSERR_NOCHIPMEM - couldn't get chip memory
OSERR_PUBNOTUNIQUE - public screen name already used

This all seems to indicate that "visitor windows" should NOT try opening or closing the screen. There does not appear to be any counter operating in the fashion you suggest.

It also provides a way that "Open/close automatically" COULD work: If public screen FooBar has that feature enabled, then when a program calls LockPubScreen("FooBar"), or if it opens a window with WA_PubScreenName="FooBar", then AmigaOS can automatically open that screen for the program.

This also means that AmigaOS must automatically close the screen when the last visitor window is CloseWindow()ed... except it doesn't in my tests.