GetWMgrPort Obtain a pointer to the Window Manager port

#include <Windows.h> **Window Manager**

void	**GetWMgrPort(***wPort* **);**	
GrafPtr	***wPort**;	receives the pointer value

This function obtains a GrafPtr to allow you access to the Window Manager's GrafPort.

wPort is a pointer to a GrafPtr. Upon return, it will contain a GrafPort value that may be used to find out about the Window Manager's graphics environment.

Returns: none

Notes: We are warned against modifying fields of the Window Manager's GrafPort. However, as a read-only data area, it can be useful in finding out such facts as the height and width of the primary screen. See the discussion of the fields of the GrafPort structure.

To simply get the size of the primary screen, access the global variable screenBits. The global variable GrayRgn may be more useful; it is the region below the menu bar, including all real estate on multiple screens.

In order to be MulitFinder compatible, applications shouldn't draw outside their windows. MultiFinder documentation warns against drawing in WMgrPort since the system "owns" the desktop and windows of other applications are drawn within it. However, if you must draw outside your window, you should follow these rules.

WMgrPort has its visRgn set to include all active screens. Its clipRgn is initially set to the rectangle -32767, -32767, 32767, 32767. This GrafPort should be treated as read-only, with global variable GrayRgn equal to the WMgrPort's visRgn less the menu bar. You should use GrayRgn to find the shape, size and coordinates of the screen. You should never have to use the WMgrPort directly and you shouldn't call GetWMgrPort ever.

Rules

Only draw to the whole screen in a "modal" way. This could be an animation across windows or visual feedback from dragging from one to another. It is important to know that no other application, including the Finder, will draw until you have finished. In the case of an animation effect, the drawing shouldn't take much time. In the case of a drag, you should only draw while the mouse button is down.

All operations should end with nothing left drawn outside your windows. Under MultiFinder it is alright not to call GetNextEvent, EventAvail or WaitNextEvent while drawing outside your windows. Use the StillDown or WaitMouseUp functions for loops that wait for the mouse button to go up.

Never draw something on the desktop and leave it there. There is no way to tell the system that you have drawn on the desktop, so the Finder may draw right over you.

Examples and How To

One famous animation effect is the ZoomRect routine. It is used by the Finder to draw a series of nested rectangles around an icon that is being opened. The rectangles zoom out to where the window for the icon will be. Another example is when you drag something to another window. This should be done with DragGrayRgn, which does the right thing and doesn't call GetNextEvent, etc.

To make these examples happen, you should use a GrafPort, not a window or WMgrPort, that covers all the screens. OpenPort will set up most of the fields of the GrafPort correctly. You have to change the visRgn of your port to a copy of GrayRgn and put the GrayRgn's rgnBox into your portRect. Don't directly manipulate the visRgn of a window under MultiFinder!

Draw using srcXor mode; this will allow you to erase as you go, by drawing each object a second time, also in srcXor mode. Remember to leave all areas outside your window exactly as you found them.

WDEFs and MDEFs

Window and menu definition procedures draw in the current port, which is set to the WMgrPort by the WindowManager and the Menu Manager. This means you don't ever have to call GetWMgrPort. You shouldn't ever draw into it except from one of these procedures!