**About the Window Manager** Information on color and multi-screen windows

Window Manager supports both multiple screen desktops and color windows for the Macintosh II-family of machines. Accordingly, special data structures are required, along with a special 'wctb' resource type. Also, to support hierarchical windows on the Mac Plus, the SE and the Mac II, **InitWindows** calls the **Menu Manager** to establish menu bar height and to draw an empty one; and **FindWindow** uses the **Menu Manager** to test whether or not a point on the screen has been selected.

Mac II-specific changes to the **Window Manager** let your applications create color windows; erase each window's area separately; let you choose either a black and white or color desk pattern; and let you drag a window from one monitor to another.

System 7.0 introduced a new look for the Macintosh Desktop. In order to implement those changes, 'wctb' and 'cctb' resources have changed in both form and use. System 7.0 uses 'wctb' resources in a different way than Inside Macintosh Volume V. It is no longer recommended that applications provide their own 'wctb's or that they change the system 'wctb' resources at all. Directly accessing 'wctb' resources may cause system crashes and/or produce ugly results. Apple suggests that developers revise their applications to use the new 'wctb' data structure.

The system will ignore old-style 'wctb' resources; as a result applications that provide their own pre-System 7.0 window color table resources will not get the colors they used to see. The system will use the default look for the windows. If new style resources are provided then the entries will be used according to the new scheme; chances are the results are not going to be as good as those obtained with the system colors. What this all means is that Apple recommends using the window color tables that the system provides, using your own will probably not enhance what you are already getting for free.

Applications that carry their own 'WDEF' and 'CDEF' resources will not get the new windows provided by using the system but they should not experience problems since they are doing all the work for themselves.

Routines for handling color windows include: **NewCWindow**, **GetNewCWindow**, **SetWinColor**, **GetAuxWin**, **GetWVariant**, **GetGrayRgn**, **GetCWMgrPort**, and **SetDeskCPat**. Additionally, all dragging and sizing routines (**MoveWindow**, **DragGrayRgn**, **GrowWindow**, **GrowWindow**, **SizeWindow**, and **ZoomWindow**) now let you drag windows across several screens and provide update events to let their contents display the correct colors. Finally, the GrayRgn **Window Manager** variable contains multi-screen management information that lets your application check the size of the desktop by examining GrayRgn's bounding box.

Customized color windows use a structure called the WMgrCPort to open a cGrafPort. It performs the same function and has the same look as WMgrPort, which opens a standard grafPort. The difference is that it lets your applications draw desktop objects in all the colors the user's Mac (and monitor) can support. The Mac II ROMs include universal definition procedures that allow applications to make use of color, when available, and

black and white when it is not. You will also want to use the universal defproc style if your applications will be used on both color and black and white machines.

A universal definition procedure should first call **SysEnvirons** ( or see the **Gestalt Manager** for more information on environment variables ) to establish what kind of machine your application is running on. If it's non-color, all the old definition procedure rules apply. If it's a color-capable machine, however, you need to add some extra steps.

The first involves changing the current port to a WMgrCPort. After that, the definition procedure needs to update the pen attributes, text attributes and background pattern fields to reflect the values of those same fields in the WMgrPort. Two things you don't have to update, however, are visRgn and clipRgn. **Window Manager** transfers them automatically.

The definition procedures are the only things that keep color and black and white ports running in parallel. Therefore, any defproc that draws in a black and white port should stick to these rules whether or not they operate any differently as a result of changes to the fields.

After the ports are parallel, color drawing can proceed in the same fashion as black and white drawing. The **GetAuxWin** routine delivers color from the auxiliary window list. Exiting from a color defproc, as well as all other features and requirements are the same as they always were.