New Technical Notes

Macintosh

Developer Support

Monitor Depth : Gimmie Depth Or Gimmie Death

Devices

Revised by: Guillermo Ortiz June 1990
Written by: Guillermo Ortiz April 1990

This Technical Note describes two new system calls that allow an application to change the depth and flags for a given device and also check whether a device supports a particular depth and flags setting. Apple provides these calls to give developers a better way to help **users** make changes when they consider it appropriate. Abusive use of these calls is a sure way to guarantee that the Thought Police come after you to confiscate your Macintoshes, your stock of Mountain Dew®, and your Technical Notes binder. This Note assumes familiarity with *Inside Macintosh*, Volume V, Graphics Devices.

Changes since April 1990: Corrected trap addresses and dispatch numbers in the `SetDepth` and `GetDepth` definitions.

Historic Novella

Since the introduction of the Macintosh II, developers have had the strong urge to change the depth of Macintosh screens under program control. Developers often ask, “How can I change the depth from my application like the Monitors cdev does?” The reasons for this question have varied from pure Macintosh hacking spirit to valid reasons to the lack of finding a good solution which would work regardless of the depth setting the user may choose.

A poor scenario occurs when a developer wants to impose a certain depth and color or black and white setting because the application does not work well, if at all, with any other configuration. The responses from DTS always include questions about what happens to the application when the system in use does not support this “optimal” configuration or when one monitor is set to the magic configuration, but others are not, or when the user brings the application to the front and it finds that the user has changed the setting to something with which it is not equipped to deal.

On the other hand, DTS does see situations where an application that deals with certain image types may work better with a particular setting and would like to present the user with a dialog box similar to the Monitors cdev to allow the user to change the depth and color settings from within the application.

Not everyone agrees on the wisdom of providing facilities for an application to allow users to change depth and color settings from within itself, but all agree that a well-behaved application (remembering that well-behaved applications are more likely to survive system

and hardware changes) should only change depth and color settings in the following circumstances:

- Depth changes can **only** be made with the user's consent; never change depths because it is simply convenient for the application. The user paid for his system and if the user wants color, be prepared to give color. If the user wants millions of colors, don't change the display to one-bit black and white.

- The minimal amount of user input to change depth and color settings should be to provide a preferences dialog box where the user would be given a yes or no choice to change depths when a particular action is chosen. The application should make the "no" choice the default and have a sensible mechanism for handling the situation when the user chooses no.

- Under **no** condition should an application change depth or color settings while in the background. An application should only initiate depth or color changes when it is the frontmost application; do not twiddle with the user's settings while in the background.

The Calls

Beginning with System Software 6.0.5 (regardless of whether or not 32-Bit QuickDraw is installed), applications can make a call to `SetDepth` to change the depth and flag settings for a given device.

```
FUNCTION SetDepth(gd:GDHandle; newDepth,whichFlags, newFlags: Integer):Integer;
   INLINE $203C,$000A, $0013,$AAA2; { Move.L #$000A0013,D0
                                      _PMgrDispatch
                                    }

pascal short SetDepth(GDHandle gd, short  newDepth,short whichFlags, short newFlags)
                     = {0x203C, 0x000A, 0x0013, 0xAAA2};
```

Where `gd` is the device to be changed, `newDepth` is the desired depth (you can pass the bit depth or the mode necessary to set the requested depth,) `whichFlags` is a bit field selector specifying which bits in `newFlags` are meaningful, and `newFlags` are bits to be set in the `gdFlags` field of the device record as specified by `whichFlags`. For example, if you want to set a depth of eight in black and white, the call would be as follows:

```
   someResult := SetDepth(myGDevice,8,1,0);
```

In this call, `newDepth = 8` sets an eight-bit depth, `whichFlags = 1` indicates that only bit one of `newFlags` is important, and `newFlags = 0` clears the `gdDevType` flag in the device record (0 = monochrome, 1 = color). `SetDepth` returns zero if successful or a non-zero value if it cannot impose the desired depth on the requested device.

Also beginning with System Software 6.0.5, applications can make a call to `HasDepth` to verify if a given device supports a mode for the desired depth.

```
FUNCTION HasDepth(gd:GDHandle; newDepth,whichFlags, newFlags: Integer):Integer;
   INLINE $203C,$000A $0014,$AAA2; { Move.L #$000A0014,D0
                                     _PMgrDispatch
                                   }

pascal short HasDepth(GDHandle gd, short  newDepth,short whichFlags, short newFlags)
                     = {0x203C,0x000A 0x0014,0xAAA2};
```

Where `gd` is the device to be verified, `newDepth` is the desired depth, `whichFlags` is a bit field selector specifying which bits in `newFlags` are meaningful, and `newFlags` are bits to be checked in the `gdFlags` field of the device record as specified by `whichFlags`. `HasDepth` returns zero if the desired depth or flag setting is not supported on the given device; otherwise, `HasDepth` returns the `mode` necessary to set the device to the desired depth (which may be passed as the `newDepth` parameter in a call to `SetDepth`).

Further Reference:

- *Inside Macintosh*, Volume V, Graphics Devices
- *Designing Cards and Drivers for the Macintosh Family*, Second Edition