Apple II Technical Notes

Developer Technical Support

Apple IIGS
#4: Changing Graphics Modes in Mid-Application

Revised by: Dave "Dave" Lyons, C.K. Haun, & Dan Oliver
January 1991
Written by: Dan Oliver
October 1986

This Technical Note discusses how to switch between the two graphics modes, 320 and 640 horizontal resolution, while running an application which uses the Window, Control, and Menu Managers.

Changes since May 1990: Added information about reinstalling fonts after restarting QuickDraw II.

Why Change Resolution?

Why not? There are certain applications where the ability to run in both modes is essential; most graphics applications fall into this category. Other applications might switch modes to provide features which their competitors lack; a financial application might display figures in 640 mode and charts in 320 mode. Still other applications may want to give the user the choice. A word processor might seem useful only in 640 mode, but what if the user wants to print greeting cards with pictures? The user does not need the line length provided in 640 mode but does need the added color of 320 mode for the pictures.

Let me preach a little. I have worked on other machines with different graphic modes and learned some things that might be of use to application programmers. Many application programmers fight mode switching with either rhetoric or apathy, then when users expect their software to run in either mode, they become frustrated when it does not allow switching. To avoid the problem of frustrating the user, you can provide mode switching (which is not as hard as you might think).

How To Change Modes

First, assume you are in an application which is running with a system menu bar, a few visible windows with scroll bars, and one window with some standard controls. At some point, the user decides to change modes, possibly via a menu item thoughtfully provided by the application programmer. Your change mode handler might look like the following:

```
;
; --- This step is necessary if QuickDraw Auxiliary is started -----------------------------
              _QDAuxShutDown              ;Shut down QDAux first
; ------------------------------------------------------------------------------------------
              _QDShutdown                 ;Shut down QuickDraw.
                                          ;This will turn graphics off so you will see
                                          ;the text screen for a second (a advertisement
                                          ;might go here).
              lda     <mode               ;Variable that holds current resolution.
              eor     #$0080              ;Flip the mode bit, $0000 = 320, $0080 = 640.
              sta     <mode               ;New value will be used to start the new mode.
;
              pei     <QDzpage            ;Pass the direct pages allocated for QuickDraw.
              pei     <mode               ;New mode.
              pei     <QDwidth            ;0 for screen width; other numbers for printing
              pei     <MyID               ;Pass my ID number.
              _QDStartup                  ;Restart QuickDraw in the new mode.
;
              _GrafOff                    ;Turn screen off because changing mode
                                          ;may not be pretty.
; --- This step is necessary if you need QuickDraw Auxiliary  -----------------------------
              _QDAuxStartUp               ;Start QDAux again
; ------------------------------------------------------------------------------------------
;
;
; --- Fix up the cursor for the new mode ---------------------------------------------------
;
              pea     0                   ;Pass minimum cursor X position.
              lda     #319                ;Maximum X position for 320 mode.
              ldx     <mode               ;320 or 640 mode?
              beq     store
              lda     #639                ;Maximum X position for 640 mode.
store          pha                         ;Pass maximum cursor X position.
              pea     0                   ;Pass minimum Y cursor position.
              pea     199                 ;Pass maximum Y cursor position.
              _ClampMouse                 ;Clamp the cursor to the new screen size.
;
              _HomeMouse                  ;Move the cursor to 0,0 to make sure
                                          ;it is on screen.
              _ShowCursor                 ;Make cursor visible.
;
;
; --- Tell tools about the change ----------------------------------------------------------
;
              _WindNewRes                 ;Tell Window Manager about the change.
              _MenuNewRes                 ;Tell Menu Manager about the change.
              _CtlNewRes                  ;Tell Control Manager about the change.
;
;
; --- Fix the screen to look good ----------------------------------------------------------
;
;               Here you might want to change the color of the desktop, windows, menus or
;               controls to look good for the new mode.
```

```
;
;               See example below.
;
; --- Redraw the screen in the new mode ------------------------------------------------
;
              pea    0                    ;Pass flag to draw entire screen.
              pea    0
              _RefreshDesktop             ;Draw entire screen.
;
              _GrafOn                     ;Now show the new screen.
;
```

That is not too bad, but I left out the fun part. Before the `RefreshDesktop` there is a section named "Fix up the screen to look good." This section is where you might want to put some color into windows, controls, and menus if you are switching to 320 mode; changing colors is not required, but there are some things which are.

When switching from 640 mode to 320 mode, some windows (both visible and invisible) might be positioned off the screen in 320 mode. The first way to handle this problem is easy for you, the programmer, but not so great for the user: close all the windows before changing modes, then position them correctly when the user opens them in the new mode. The second way to handle the problem is to walk the window list and move all the windows, maybe even change their sizes. You could double each window's horizontal starting position and width when switching from 320 mode to 640 mode and halve it when changing from 640 mode to 320 mode. The vertical position and height are okay. An example of the second method is given below.

Windows with vertical scroll bars in the window frame are the same width when you change modes, so switching from 320 mode to 640 mode results in a narrower bar while changing from 640 mode to 320 mode produces a wider bar. The bars change to the correct size as soon as the user resizes the window, since `SizeWindow` deletes the old scroll bars and allocates new ones according to the current mode. If, as suggested above, you resize all the windows after the mode change and before calling `RefreshDesktop`, you should be in good shape. If you choose not the follow this recommendation, you should call `SizeWindow` for every window with scroll bars and change the size of each window at least one pixel since `SizeWindow` does not do anything if the passed size is not different than the current size.

You should dispose of scroll bars in a window's content region and recreate them; this is not nice, but very few applications have scroll bars in a window's content region.

You should not resize any open new desk accessory (NDA) windows. NDAs may be dependent on screen mode, or their current position, or other such things which may change with resolution. To be kind to the NDAs, you should issue a `CloseAllNDAs` call. This call allows the NDAs to go through their normal close procedures. If a user wants an NDA open in the new screen resolution he must reopen it. This assures that the NDA always knows its own position and the current screen resolution.

`WindNewRes` resets the desktop shape and pattern and the Window Manager's icon font to their defaults for the new mode, so if you changed any of these, you must add to or subtract from the desktop again and reinitialize to your custom pattern or icon font again.

`CtlNewRes` resets the Control Manager's icon font to the default for the new mode, so if you changed the Control Manager's icon font, you must reinitialize to your icon font again.

Reinstalling Large Fonts

After restarting QuickDraw II, you should call `InstallFont` again on the fonts your application is using. This causes the Font Manager to call `InflateTextBuffer` so that QuickDraw can draw text correctly in large font sizes.

Repositioning and Resizing Windows in the New Mode

Here is an example of how to reposition and resize windows in the new mode.

```
;             QuickDraw and the tools have already been reinitialized in the new mode.
;             mode = $0000 if in 320 mode, $0080 if in 640 mode.
;
BoundsRect    equ    8                  ;Offsets in port record from QuickDraw document.
PortRect      equ    16
;
              _CloseAllNDAs              ; close all open NDA windows
              pha                        ;Space for result.
```

```
              pha
              _FrontWindow                ;Start with the top most window, this assumes
              bra    enter                ;there are no invisible windows ahead of the
                                          ;active window in the window list.
              ldy    #BoundsRect+2
              lda    [window],y           ;Get window's starting horizontal position.
              eor    #$FFFF               ;Convert to screen coordinate (negate it).
              inc    a
              asl    a                    ;Double it if we're going to 640 mode.
              ldx    <mode                ;Going to 320 or 640 mode?
              bne    store1               ;Ready if we're going to 640.
              lsr    a                    ;Otherwise, undo the doubling,
              lsr    a                    ;and halve the starting horizontal position.

store1         pha                         ;Pass window's new X starting position.
              ldy    #BoundsRect
              lda    [window],y           ;Get window's starting vertical position.
              eor    #$FFFF               ;Convert to screen coordinate.
              inc    a
              pha                         ;Pass window's current Y starting position.
              pei    <window+2            ;Pass window to move.
              pei    <window
              _MoveWindow                 ;Move the window to its new position.
;
              ldy    #PortRect+6          ;Get window's current width.
              lda    [window],y           ;(This assumes the window's origin is 0,0.)
              asl    a                    ;Double the window's width if going to 640 mode.
              ldx    <mode                ;Going to 320 or 640 mode?
              bne    store2               ;Ready if we're going to 640.
              lsr    a                    ;Otherwise, undo the doubling,
              lsr    a                    ;and halve the window's width.
store2         pha                         ;Pass window's new width.
              ldy    #PortRect+4
              lda    [window],y           ;Get window's height.
              pha                         ;Pass window's current height.
              pei    <window+2            ;Pass window to resize.
              pei    <window
              _SizeWindow                 ;Resize the window.
;
              pha                         ;Space for result.
              pha
              pei    <window+2            ;Pass pointer to window we just processed.
              pei    <window
              _GetNextWindow              ;Get the pointer to the next window.
;
enter          pla                         ;Remember the pointer to this window.
              sta    <window
              pla
              sta    <window+2
;
              ora    <window              ;Are there any more windows?
              bne    loop
;
```

WindNewRes

Generally, `WindNewRes` does the following:

- closes its port
- opens its port again, now in the new mode
- reinitializes the desktop size
- chooses the proper icon font for close and zoom boxes
- reinitializes the desktop pattern
- changes the SCB byte of each window's port to the new mode
- recomputes the `VisRgn` for each window

MenuNewRes

Generally, `MenuNewRes` does the following:

- closes its port
- opens its port again, now in the new mode
- reinitializes internal parameters, like vertical line width, for the new mode
- reinitializes the color palette via `InitPalette`
- subtracts the system menu bar from the desktop (this is why you must call `WindNewRes` first)
- draws the system menu bar

CtlNewRes

Generally, `CtlNewRes` does the following:

- chooses the proper icon font for radio button, check box, grow box and scroll bar arrows
- reinitializes internal parameters, like vertical line width, for the new mode

Further Reference

- *Apple IIGS Toolbox Reference*