Apple II Technical Notes

Developer Technical Support

Apple IIGS
#74: A Faster List Manager Draw Routine

Written by: Jim Mensch
November 1989

This Technical Note presents a method for speeding up custom List Draw routines, with sample source code for the APW assembler.

The List Manager is designed to allow users to select from relatively small lists. In this respect, it does quite well. However, when the width or height of the displayed area gets above a certain size (about 20 chars wide and 6 items tall) the list starts to scroll slower than normal. This is due to the design of the List Manager's scroll routine. To scroll text, the List Manager calls `ScrollRect` to scroll the list, then it redraws all the visible members. On small lists this is fine, but on larger lists it can cause the redrawing of much data that is already on the screen, which can take time. To cure this problem, you can include a simple draw procedure in your program that checks the `clipRgn` before anything is drawn. This way, you will not attempt to redraw items that do not need redrawing.

The custom draw routine in this Note is an example of one such routine that could be used. It first checks the current `clipRgn` (which the List Manager was kind enough to shrink down to include only the portion of the list that needs redrawing) against the passed item rectangle. If the rectangle is in any way enclosed in the `clipRgn`, then the member is redrawn; otherwise the routine simply returns to the List Manager without

drawing. This sample routine is designed to work only with Pascal-style strings, but it can be easily modified to use any other type of string you choose.

```
MyListDraw  Start
;
; This routine draws a list member if any part of the member's
; rectangle is inside the current clipRgn.
;
; Note that the Data Bank register is not defined on entry
; to this routine.  If you use any absolute addressing, you
; must set B yourself and restore its value before exiting.
;
top         equ 0
left        equ top+2
bottom      equ left+2
right       equ bottom+2
rgnBounds   equ 2
;
oldDPage    equ 1
theRTL      equ oldDPage+2
listHand    equ theRTL+3
memPtr      equ listHand+4
theRect     equ memPtr+4
            using globals

            phd
            tsc
            tcd
```

```
            pha
            pha
            _GetClipHandle
            PullLong listHand

            ldy #2
            lda [listhand],y
            tax
            lda [listhand]
            sta listhand
            stx listhand+2

            lda [therect]               ; now test the top
            dec a                       ; adjust and give a little slack
            ldy #rgnbounds+bottom
            cmp [listhand],y            ; rgnRectBottom>=top?
            blt skip2
            brl NoDraw                  ; if not don't draw..
Skip2       ldy #bottom                 ; now see if the bottom is higher than the top
            inc a                       ; give a little slack
            lda [therect],y
            ldy #rgnBounds+top
            cmp [listhand],y
            blt NoDraw
NoTest      ANOP

            PushLong theRect
            _EraseRect              ; erase the old rectangle

            ldy #left
            lda [theRect],y
            tax
            ldy #bottom
            lda [theRect],y
            dec a
            phx
            pha
            _MoveTo
            ldy #2
            lda [memptr],y
            pha
            lda [memptr]
            pha
            _DrawString

            ldy #4
            lda [memPtr],y
            and #$00C0              ; strip to the 6 and 7 bits
            beq memDrawn            ; if they are both 0 the member is drawn
            cmp #$0080              ; member selected?
            bne noSelect            ; member not selectable
            PushLong theRect
            _InvertRect
            bra memDrawn
; if we get here the member is disabled
noSelect    PushLong #DimMask
            _SetPenMask
            PushLong theRect
            _EraseRect
            PushLong #NorMask
            _SetPenMask
memDrawn    ANOP


; exit here
```

```
            pld
            sep #$20
            longa off
            pla
            ply

            plx
            plx
            plx
            plx
            plx
            plx
            phy
            pha
            rep #$20
            longa on
            rtl

DimMask     dc  i1'$55,$AA,$55,$AA,$55,$AA,$55,$AA'
NorMask     dc  i1'$FF,$FF,$FF,$FF,$FF,$FF,$FF,$FF'
            end
```

Further Reference

- *Apple IIGS Toolbox Reference*, Volumes 1 and 3