```
; File IconBounce.TXT
;--------------------------------------------------------------------------
;
;   IconBounce uses custom plotting routines to bounce a lot of icons
;   around on the deskTop.
;
;     written by Andy Hertzfeld, Nov 17 1985
;
;--------------------------------------------------------------------------

INCLUDE MacSys:MacTraps.D

; System definitions, etc.

IOCompletion    EQU     12              ;offset to completion routine address
IOFileName      EQU     18              ;offset to fileName
IOVRefNum       EQU     22              ;offset to volume refNum
IOFileType      EQU     26              ;offset to type byte, permissions
IOMisc          EQU     28              ;offset to misc param
IOBuffer        EQU     32              ;offset to buffer pointer
IOByteCount     EQU     36              ;offset to count
IONumDone       EQU     40              ;offset to number done
IOPosMode       EQU     44              ;offset to positioning mode
IOPosOffset     EQU     46              ;offset to position value

EvtMsg          EQU     2               ;offset to message field
EvtMeta         EQU     14              ;offset to metaKey field
Where           EQU     10              ;mouse offset in event record
portRect        EQU     16              ;offset to portRect

ScreenRow       EQU     $106            ;rowBytes of screen [word]
Ticks           EQU     $16A
KeyMap          EQU     $174
Time            EQU     $20C
ScrnBase        EQU     $824
CurApRefNum     EQU     $900
WmgrPort        EQU     $9DE

MaxX            EQU      512
MaxY            EQU      342

; Icon Data Structure

NextIcon        EQU     0               ;handle of next structure
IconData        EQU     4               ;128 byte 32 by 32 bitmap
IconMask        EQU     132             ;128 byte mask
IconPosition    EQU     260             ;longword position
IconVelocity    EQU     264             ;velocity

IconDSSize      EQU     268             ;total data structure size

; Global Variable Definitions

QuickBase       EQU     -4              ;quickDraw globals
MyEvent         EQU     QuickBase-200   ;my event record
QuitFlag        EQU     myEvent-2       ;boolean for exiting
WhichWindow     EQU     QuitFlag-4      ;whichWindow result

NumIcons        EQU     WhichWindow-2   ;# of icons allocates
FirstIcon       EQU     numIcons-4      ;handle of 1st one
LastIcon        EQU     FirstIcon-4     ;handle of last one

BigBuffer       EQU     LastIcon-4      ;pointer to big buffer


XDEF            START

START:

; first allocate some zeroed space by clearing it off the stack

                MOVE    #511,D0         ;need about 2K bytes
ClearLoop
```

```
                CLR.L   -(SP)
                DBRA    D0,ClearLoop

; now grow the heapZone as large as we can make it

                MOVEQ   #64,D0
                SWAP    D0                  ;get huge number
                _NewHandle                  ;grow out the heap
                BNE.S   InitWorld           ;we expect the error
                _DisposHandle               ;if it not, dispose it

; initialize QuickDraw and the toolBox

InitWorld
                PEA     QuickBase(A5)       ;push address of QuickDraw vars
                _InitGraf                   ;initialize QuickDraw
                _InitFonts                  ;initialize the font manager
                _InitCursor                 ;get the arrow cursor
                _InitWindows                ;initialize the window manager
                _InitMenus                  ;ditto for menus

                CLR.L   -(SP)               ;our recovery proc is  NIL
                _InitDialogs                ;initialize dialogs
                _TEInit                     ;and text edit, too

                BSR     SetHourGlass

; initialize our globals

                CLR.B   QuitFlag(A5)
                CLR.W   numIcons(A5)

                CLR.L   FirstIcon(A5)
                CLR.L   LastIcon(A5)

; allocate the big buffer

                MOVE.L  #24000,D0
                _NewPtr
                BNE     ErrorExit
                MOVE.L  A0,BigBuffer(A5)

; display title message

                MOVE.L  WmgrPort,-(SP)
                _SetPort
                PEA     BigRect
                _ClipRect

                MOVE.L  #$000E0038,-(SP)
                _MoveTo

                PEA     TitleString
                _DrawString

; allocate the icons

                BSR     AllocIcons

                _HideCursor

                MOVEQ   #31,D0
                _FlushEvents

; start the main event loop

MainLoop
                _SystemTask

                BSR     HandleEvent     ;check for events and handle them
                BSR     AnimateIcons

                TST.B   QuitFlag(A5)
```

```
                BEQ.S   MainLoop

                _ExitToShell                ;back to finderLand

; HandleEvent checks for events and handles them as necessary.  It handles
; the menu commands and all interaction with the user.

HandleEvent
                SUBQ    #2,SP               ;make room for result
                MOVE.W  #-1,-(SP)           ;we want every event
                PEA     myEvent(A5)         ;stick it in our global
                _GetNextEvent               ;get the event

                TST.B   (SP)+               ;did we get one?
                BEQ.S   NoEvent             ;if not, we're done

                MOVE.W  myEvent(A5),D0      ;get the event number
                BEQ.S   NoEvent             ;ignore the Null event
                CMP     #9,D0               ;only care about 1st 9 events
                BGE.S   NoEvent

                ADD     D0,D0               ;double for word index
                LEA     EvtDispatch,A0      ;get the address of the table
                ADD.W   0(A0,D0),A0         ;get routine address
                JMP     (A0)                ;go to it!

; here is the event dispatch table

EvtDispatch
                DC.W    NoEvent-EvtDispatch
                DC.W    MyMouseDown-EvtDispatch
                DC.W    MyMouseUp-EvtDispatch
                DC.W    MyKeyDown-EvtDispatch
                DC.W    NoEvent-EvtDispatch
                DC.W    MyKeyDown-EvtDispatch
                DC.W    MyUpdateEvt-EvtDispatch
                DC.W    MyDiskInsert-EvtDispatch
                DC.W    MyActivate-EvtDispatch
MyDiskInsert
MyMouseUp
MyActivate
NoEvent
                RTS


; Handle keyboard events

MyKeyDown
                ;ST      QuitFlag(A5)
                RTS

; handle update events

MyUpdateEvt
                RTS

; the following routine handles mouseDowns.  First call FindWindow
; to classify where the mouse went down

MyMouseDown

                ST      QuitFlag(A5)
                RTS

; AllocIcons opens the desktop, and allocates an icon data structure for
; each ICN# in the file.

AllocIcons
                SUBQ    #2,SP
                PEA     DeskTopName
                _OpenResFile
                TST     (SP)
                BMI     ErrorExit
```

```
                SUBQ    #2,SP
                MOVE.L  ICNRType,-(SP)          ;push ICN# type
                _CountResources                 ;get # of resources
                MOVE.W  (SP)+,D3                ;keep in D3
                BLE     ErrorExit

; limit the # of icons to 64, unless the option key is down

                BTST    #2,KeyMap+7
                BNE.S   NoILimit

                CMP     #64,D3
                BLE.S   NoILimit

                MOVEQ   #64,D3

; OK, now loop for each icon

NoILimit
                MOVEQ   #1,D4                   ;init index
AllocIcLoop
                SUBQ    #4,SP                   ;make room for result
                MOVE.L  ICNRType,-(SP)          ;push ICN#
                MOVE.W  D4,-(SP)                ;push index
                _GetIndResource                 ;get it
                MOVE.L  (SP)+,D5                ;got it?
                BLE     DoneAllocIcon

; we have the icon handle, so allocate the structure

                MOVE.L  #IconDSSize,D0
                _NewHandle
                BNE     ErrorExit

                MOVE.L  A0,A3                   ;get new handle
                MOVE.L  (A3),A2                 ;handle->ptr
                CLR.L   (A2)+                   ;link is zero
                MOVE.L  A2,A1                   ;set up dest

                MOVE.L  D5,A0
                MOVE.L  (A0),A0                 ;set up source
                MOVE.L  #256,D0                 ;256 bytes to move
                _BlockMove

                ADD.L   #256,A2                 ;skip over save area

; generate positions 0 < x < 512,  0 < y < 302

                SUBQ    #2,SP
                _Random
                MOVEQ   #0,D0
                MOVE.W  (SP)+,D0
                DIVU    #302,D0
                SWAP    D0
                ADDQ    #1,D0
                MOVE    D0,(A2)+

                SUBQ    #2,SP
                _Random
                MOVE.W  (SP)+,D0
                AND.W   #511,D0
                ADDQ    #1,D0
                MOVE    D0,(A2)+

; generate velocitys from 1 to 8

                SUBQ    #2,SP
                _Random
                MOVE.W  (SP)+,D0
                AND     #7,D0
                ADDQ    #1,D0
```

```
                MOVE.W  D0,(A2)+

                SUBQ    #2,SP
                _Random
                MOVE.W  (SP)+,D0
                AND     #7,D0
                ADDQ    #1,D0
@2
                MOVE.W  D0,(A2)+

; link it in the list

                MOVE.L  LastIcon(A5),D0
                BNE.S   LinkItIn

                MOVE.L  A3,FirstIcon(A5)
                MOVE.L  A3,LastIcon(A5)
                BRA.S   BumpICount
LinkItIn
                MOVE.L  A3,LastIcon(A5)
                MOVE.L  D0,A0
                MOVE.L  (A0),A0
                MOVE.L  A3,(A0)                 ;link it in
BumpICount
                ADDQ    #1,numIcons(A5)

DoNextIcon
                ADDQ    #1,D4                   ;bump index
                CMP     D3,D4                   ;done?
                BLT     AllocIcLoop
DoneAllocIcon
                _CloseResFile

                RTS

; ShowIcon is the routine that plots an icon.  It is adopted from the
; BigCursor routines.  The handle of the icon data structure is passed
; in A3.

ShowIcon
        MOVEM.L D0-D7/A0-A4,-(SP)       ; save registers
        MOVE.L  (A3),A3                 ; de-reference icon data structure

        MOVEQ   #32,D5                  ; size of icon
        MOVEQ   #16,D6                  ; half size

        LEA     IconData(A3),A2         ; cursor data bitmap address
        LEA     IconMask(A3),A4         ; cursor mask bitmap address

;  first handle the x coordinate

        MOVE    IconPosition+2(A3),D0   ; get left
        MOVEQ   #15,D2                  ; upper left X-coordinate
        AND.W   D0,D2                   ; bit offset within word

        AND     #$FFF0,D0               ; truncate to nearest word
        BGE.S   @0                      ; if positive, skip

        MOVEQ   #0,D0                   ; minimum upper left X-coord of 0
        ADD.W   D6,D2                   ; adjust right shift count

; if shift count > 15, just move over a word

@0
        CMP     D6,D2                   ;is it?
        BLT.S   @7                      ;if not, skip

        SUB     D6,D2                   ; reduce bit index
        ADD.W   D6,D0                   ; bump base point
@7

; establish "last word" boolean
```

```
        CLR.W   -(SP)

        MOVE    D0,D1                   ; copy coordinate
        SUB.W   #MaxX-32,D1             ; upper left X-coord <= 512-32
        BLT.S   @2                      ; branch if <= 512-32

        MOVE.W  #MaxX-32,D0             ; maximum X-coord of 512-32
        ADD.W   D1,D2                   ; adjust left shift count
        ST      (SP)                    ; set the boolean

; handle the y coordinate

@2      MOVE.W  D5,D4                   ; 32 rows

        MOVE.W  IconPosition(A3),D1     ; get Y-coordinate

;   Display the icon on the screen.

        MOVE.L  BigBuffer(A5),A1        ; offscreen memory address
        LSR.W   #3,D0                   ; convert X-coord to bytes
        ADD.W   D0,A1                   ;  and add to screen address
        MOVE.W  ScreenRow,D5            ; bytes per row on screen
        MULU    D5,D1                   ;  * Y-coord
        ADD.L   D1,A1                   ;  added to screen address

        SUBQ    #4,D5                   ; bias D5 for loop

        TST     D2                      ; is shiftcount = 0?
        BEQ.S   BotFastLoop             ; if so, go ultra fast

; OK, for added speed, two different loops, depending on the
; if we need the 3rd word (as specified by the top of stack boolean)

        TST.B   (SP)+
        BNE     BotCurlLoop
        BRA.S   BotCurLoop              ; test for rows=0

; here is the icon plotting loop.  First do the leftmost 32 bits

ShowCurLoop
        MOVE.L  (A2)+,D0                ; get the data
        MOVE.L  D0,D6                   ; copy for later
        LSR.L   D2,D0                   ; shift into place

        MOVE.L  (A4)+,D1                ; get the mask
        MOVE.L  D1,D7                   ; copy for later
        LSR.L   D2,D1                   ; shift into place
        NOT.L   D1                      ; complement mask

        AND.L   D1,(A1)                 ; bit-clear with the mask
        OR.L    D0,(A1)+                ; plot the data

; now handle the rightmost 16 bits

        MOVEQ   #16,D1
        SUB.W   D2,D1                   ; compute left shift count

        ASL.W   D1,D6                   ; shift the data
        ASL.W   D1,D7                   ; shift the mask
        NOT.W   D7                      ; complement the mask

        AND.W   D7,(A1)                 ; bit clear the mask
        OR.W    D6,(A1)                 ; plot the data
@0
        ADD     D5,A1                   ; bump to next row
BotCurLoop
        DBRA    D4,ShowCurLoop          ; loop till done

DoneShowLoop
        MOVEM.L (SP)+,D0-D7/A0-A4       ; restore registers
DoneShow
        RTS
```

```
; this loop is used when we're near the right edge and don't have to
; plot the third word

ShowCur1Loop
        MOVE.L  (A2)+,D0                ; get the data
        LSR.L   D2,D0                   ; shift into place

        MOVE.L  (A4)+,D1                ; get the mask
        LSR.L   D2,D1                   ; shift into place
        NOT.L   D1                      ; complement mask

        AND.L   D1,(A1)                 ; bit-clear with the mask
        OR.L    D0,(A1)+                ; plot the data

        ADD     D5,A1                   ; bump to next row
BotCur1Loop
        DBRA    D4,ShowCur1Loop         ; loop till done

        BRA     DoneShowLoop

; special fast loop for the 6% case where we don't have to shift

FastLoop
        MOVE.L  (A2)+,D0                ;fetch the data
        MOVE.L  (A4)+,D1                ;fetch the mask
        NOT.L   D1                      ;complement mask

        AND.L   D1,(A1)                 ;plot the mask
        OR.L    D0,(A1)+                ;plot the data

        ADD     D5,A1
BotFastLoop
        DBRA    D4,FastLoop

        ADDQ    #2,SP                   ;discard boolean
        BRA     DoneShowLoop

; Error handling routines

ErrorExit
        DC.W    $F123

        RTS


; AnimateIcons is the mainline routine that bounces the icons

AnimateIcons
        BSR     GrayBuffer              ;fill big buffer with gray

        BSR     UpdateIconPositions
        BSR     DrawIntoBuffer

; now move the buffer onto the screen with blockMove

        MOVE.W  ScreenRow,D0
        MULU    #20,D0
        ADD.L   ScrnBase,D0
        MOVE.L  D0,A1                   ;screen is destination
        MOVE.L  BigBuffer(A5),A0        ;big buffer is source

        MOVE.L  #20608,D0
        _BlockMove

        RTS

; DrawIntoBuffer goes through the icon data structure, drawing each icon

DrawIntoBuffer
        MOVE.L  A3,-(SP)                ;save work reg

        MOVE.L  FirstIcon(A5),D0        ;get first one
        BEQ.S   DoneDIB                 ;if empty, skip
```

```
DIBLoop
        MOVE.L  D0,A3

        BSR     ShowIcon

        MOVE.L  (A3),A0
        MOVE.L  (A0),D0
        BNE.S   DIBLoop
DoneDIB
        MOVE.L  (SP)+,A3
        RTS

; UpdateIconPositions animates the icon positions

UpdateIconPositions

        MOVE.L  FirstIcon(A5),D0        ;get first one
        BEQ     DoneUIP                 ;if empty, skip

UIPLoop
        MOVE.L  D0,A0
        MOVE.L  (A0),A0

        MOVE.L  IconPosition(A0),D0
        MOVE.L  IconVelocity(A0),D1

; OK, bounce in the x position

        ADD.W   D1,D0                   ;compute new position
        BGE.S   @0                      ;if > 0, skip

        SUB.W   D1,D0                   ;undo it
        NEG.W   D1                      ;toggle velocity

@0
        CMP.W   #510,D0
        BLT.S   BounceY

        SUB.W   D1,D0
        NEG.W   D1

BounceY
        SWAP    D0
        SWAP    D1

        ADD.W   D1,D0
        BGE.S   @0

        SUB.W   D1,D0
        NEG.W   D1
@0
        CMP     #302,D0
        BLT.S   NextBounce

        SUB.W   D1,D0
        NEG.W   D1

NextBounce
        SWAP    D0
        SWAP    D1
        MOVE.L  D0,IconPosition(A0)
        MOVE.L  D1,IconVelocity(A0)

        MOVE.L  (A0),D0
        BNE     UIPLoop
DoneUIP
        RTS


; GrayBuffer fills the 322 scan lines at GrayBuffer with gray

GrayBuffer
```

```
        MOVE    #160,D2                 ;# of scan line pairs - 1
        MOVE.L  BigBuffer(A5),A0        ;point to the buffer
FillGLoop
        MOVE.L  #$55555555,D0           ;get gray
FillGLoop2
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long
        MOVE.L  D0,(A0)+                ;fill a long

; OK, now a scan line is done, so flip the gray

        NOT.L   D0
        BMI.S   FillGLoop2

        DBRA    D2,FillGLoop2

        RTS

; SetHourGlass installs the hourGlass (watch) cursor

SetHourGlass

        SUBQ    #4,SP
        MOVE    #4,-(SP)
        _GetCursor

        MOVE.L  (SP)+,A0
        MOVE.L  (A0),-(SP)
        _SetCursor

        RTS


; Constants, etc.

ICNRType
        DC.B    'ICN#'
DeskTopName
        DC.B    7,'DeskTop'
BigRect
        DC.W    0,0,1000,1000
TitleString
        DC.B    60,'IconBounce by Andy Hertzfeld...  Press mouse button '
        DC.B    'to exit. '
```