```
;                       ****************************
;                       *     ST Star Scroller     *
;                       * ------------------------ *
;                       * by Paul Lay, August 1986 *
;                       ****************************

; ***************************************************************************
; Equates.

gemdos          equ     $01                     ; gemdos trap vector
super           equ     $20                     ;       supervisor entry
keep_process    equ     $31                     ;       exit maintaining store

xbios           equ     $0E                     ; xbios trap vector
physbase        equ     $02                     ;       return screen RAM address
getrez          equ     $04                     ;       return screen resolution
random          equ     $11                     ;       return random number

vblqueue        equ     $0456                   ; VBI queue


; ***************************************************************************
; Initialise the stars and set up the VBI.

start
                move.w  #physbase,-(sp)         ; find screen memory
                trap    #xbios
                addq.l  #$02,sp
                move.l  D0,scrn_mem_pointer     ; save pointer

                move.w  #getrez,-(sp)           ; find screen resolution
                trap    #xbios
                addq.l  #$02,sp
                asl.l   #$01,D0
                lea     masks,A0                ; save initial position mask
                move.w  $00(A0,D0.l),bit_mask
                lea     sizes,A0                ; save scan line size (words)
                move.w  $00(A0,D0.l),line_size
                lea     alongs,A0               ; save course scroll step
                move.w  $00(A0,D0.l),bytes_along

                lea     star_x,A5               ; initial star positions
                move.w  #$C8,D7
get_position
                move.w  #random,-(sp)           ; get random position
                trap    #xbios
                addq.l  #$02,sp
                and.w   bit_mask,D0             ; mask appropriately
                cmp.w   line_size,D0            ; and check range
                bcc     get_position
                move.w  D0,(A5)+                ; set position
                dbra    D7,get_position

                bsr     draw_stars              ; draw the stars

                clr.l   -(sp)                   ; enter supervisor mode
                move.w  #super,-(sp)
                trap    #gemdos
                addq.l  #$06,sp
                move.l  D0,-(sp)

init_vbi
                move.l  vblqueue,A0             ; search VBI queue
                move.w  #$08,D0
test
                tst.l   (A0)+
                beq     set_vbi                 ; check for free entry
                dbra    D0,test
                bra     exit_supervisor
set_vbi
                move.l  #my_vbi,-(A0)           ; vector VBI routine

exit_supervisor
                move.w  #super,-(sp)            ; return to user mode
                trap    #gemdos
                addq.l  #$06,sp

                move.w  #$00,-(sp)              ; terminate program
                move.l  #$800,-(sp)             ; of size $800 bytes
                move.w  #keep_process,-(sp)     ; exit program maintaining store
                trap    #gemdos


; ***************************************************************************
; The star scrolling VBI.

my_vbi
                bsr     erase_stars             ; remove stars
                bsr     move_stars              ; update positions
                bra     draw_stars              ; redraw stars


; ***************************************************************************
; Draw and erase the stars (exclusive_or).

draw_stars
erase_stars
                move.l  scrn_mem_pointer,A0     ; locate screen memory
                move.l  #star_x,A5              ; locate star positions
                move.w  #$64,D0
                move.w  even_bits,D1            ; bit image for even stars
                move.w  odd_bits,D2             ; bit image for odd stars
draw_next
                move.w  (A5)+,A1                ; locate star offset
                eor.w   D1,$00(A0,A1.w)         ; store even bit image
                add.l   #$A0,A0
                move.w  (A5)+,A1                ; locate star offset
                eor.w   D2,$00(A0,A1.w)         ; store odd bit image
                add.l   #$A0,A0
                dbra    D0,draw_next
                rts


; ***************************************************************************
; Move the stars.

move_stars
                move.w  even_bits,D0            ; test even bit image
                cmpi.w  #%1000000000000000,D0
                bne     okay1
                bsr     course_even_scroll      ; course scroll even stars
okay1
                rol.w   #$01,D0                 ; fine scroll even bit image
                move.w  D0,even_bits

                move.w  odd_bits,D0             ; test odd bit image
                cmpi.w  #%0100000000000000,D0
                bne     okay2
                bsr     course_odd_scroll       ; course scroll odd stars
okay2
                rol.w   #$02,D0                 ; fine scroll odd bit image
                move.w  D0,odd_bits
                rts


; ***************************************************************************
; Course scroll the even stars (slow plane).

course_even_scroll
                move.l  #star_x,A5              ; set address of first even star
                bra     all_scroll              ; perform the scroll


; ***************************************************************************
; Course scroll the odd stars (fast plane).

course_odd_scroll
                move.l  #star_x+$02,A5          ; set address of first odd star


; ***************************************************************************
; Perform the scroll.

all_scroll
                move.w  #$64,D1
next_scroll
                move.w  (A5),D2                 ; get star offset
                bne     okay3                   ; check if zero
                move.w  line_size,D2            ; reset position
okay3
                sub.w   bytes_along,D2          ; move left (course)
                move.w  D2,(A5)                 ; store new offset
                addq.l  #$04,A5                 ; move onto next star
                dbra    D1,next_scroll
                rts


; ***************************************************************************
; Variables.

even_bits                                       ; even bit image
                dc.w    %0000000000000001
odd_bits                                        ; odd bit image
                dc.w    %0000000000000001
masks                                           ; masks for initial positions
                dc.w    $F8,$FC,$7E
sizes                                           ; sizes of scan lines (words)
                dc.w    $A0,$A0,$50
alongs                                          ; course scroll steps
                dc.w    $08,$04,$02
scrn_mem_pointer                                ; pointer to screen memory
                ds.l    $01
bit_mask                                        ; bit mask used
                ds.w    $01
line_size                                       ; line size used
                ds.w    $01
bytes_along                                     ; course scroll step used
                ds.w    $01
star_x                                          ; star positiions
                ds.w    $C8

; ***************************************************************************

                end
```

**Star scroller source code**