Ja-PONG

a MSX game

Written/Coded by Alfonso D. C. (Dioniso)
24-06-2005, Huelva (Spain)

Source code for the MSX game Ja-PONG, coded for the 1K PONG contest:

http://karoshicorp.proboards40.com/

Program used: Chaos Assembler 3, for PC

As you can see, the code is quite a mess ... enjoy!

```
        .org $c400-$07

        .db $fe
        .dw startProgram,endProgram,startProgram

startProgram:

;some variables in RAM
ball:           .equ $d002
print_add:      .equ $d004
digit_add:      .equ $d006
speed_ball:     .equ $d008
speed1:         .equ $d00a
speed2:         .equ $d00b
speed3:         .equ $d00c

        ld a,1
        ld ($f3ea),a
        ld ($f3eb),a
        ld a,15
        ld ($f3e9),a    ;colour 1,15,15
        xor a
        ld ($f3db),a    ;no keyboard-click
        call $cc        ;functions off
        call $6f        ;screen 1

        call $41        ;turn off screen

        ld a,($F3E0)
        or %00000010
        and %11111110
        ld b,a
        ld c,1
        call $47        ;sprites 16*16

;some character definitions
        ld a,$ff
        ld hl,840
        ld bc,8
        call $56
        ld a,$db
        ld hl,6176
        ld bc,736
        call $56
        ld a,$20
        ld hl,6208
        ld bc,672
        call $56
        ld hl,14336
        ld bc,2048
```

```
        xor a
        call $56        ;clear all sprites definitions

        ld hl,spr_definition
        ld de,14336
        ld bc,160       ;(5 sprites * 32 datas)
        call $5c

        ld hl,spr_attributes
        ld de,6912
        ld bc,20        ;(5 sprites * 4 attributes)
        call $5c

;some colour: blue border, red numbers
        ld a,$41
        ld hl,8219
        call $4d
        ld a,$81
        ld hl,8205
        call $4d

zero:
;let's initialize some VAR
        ld a,2  ;2=left-down (ball direction)
        ld (mode),a

        call init_some
        ld (score1),a
        ld (score2),a
        call print_score        ;print scores 00-00

        ld a,86
        ld (bat1),a     ;Y
        ld (bat2),a     ;Y
        ld a,235
        ld (ball+1),a   ;X
        call writey

        call $44        ;turn on screen

;player1 moves
main_loop:
sound_cnt:      .equ $+1
        ld a,$3e        ;check the 6/50 second of the rebound sound.
        or a
        jr z,kr
        dec a
        or a
        call z,silence          ;turn off sound
        ld (sound_cnt),a
```

```
kr:
        call kronos     ;check timer

joy:
      xor    a
      call   $d5
      or     a
      jr     nz,no_joy
      ld     a,1
      call   $d5
; Move P1 paddle
no_joy:

up1:
      dec a
      jr     nz,down1
bat1:   .equ $+1
      ld     a,$3e
      cp     14
      jr     z,player2
      add a,-3
      ld     (bat1),a
      jr     player2
down1:
      cp     4   ;CP 5
      jr     nz,player2
      ld     a,(bat1)
      cp     152
      jr     z,player2
      add a,3
      ld     (bat1),a

player2:
      ld     a,2
      call   $d5
up2:
      dec a
      jr     nz,down2
bat2:   .equ $+1
      ld     a,$3e
      cp     14
      jr     z,ball_move
      add a,-3
      ld     (bat2),a
      jr     ball_move
down2:
      cp     4   ;CP 5
      jr     nz,ball_move
      ld     a,(bat2)
```

```
        cp      152
        jr      z,ball_move
        add a,3
        ld      (bat2),a

ball_move:
;let's check ball-paddle contact
mode:  .equ $+1
        ld a,$3e
        or a
        jr z,if_pad2
        dec a    ;CP 1
        jr z,if_pad2

;if_pad1:
        ld a,(ball+1)
        cp 12
        jr c,check_pad1
        jr real_move_ball       ;it doesn't touch paddle1

if_pad2:
        ld a,(ball+1)
        cp 241
        jr nc,check_pad2
        jr real_move_ball       ;it doesn't touch paddle2

check_pad2:
        cp 246
        jr nc,real_move_ball
        ld a,(bat2)
        add a,-4
        ld e,a              ;e=ball Y
        ld a,(ball)         ;a=pad1 Y
        sbc a,e             ;subtract
        ld hl,table_pad_values-2
        ld b,2
        jr work_table_pad

check_pad1:
        cp 7
        jr c,real_move_ball
;let's check 6 parts of the paddle to see direction of the ball
        ld a,(bat1)
        add a,-4
        ld e,a              ;e=paddle1 Y
        ld a,(ball)         ;a=ball Y
        sbc a,e             ;subtract
        ld hl,table_pad_values-2
        ld b,0
```

```
work_table_pad:
        sub 6   ;CP 6
        ld e,2
        jr c,set_pad_values
        sub 6   ;CP 12
        ld e,4
        jr c,set_pad_values
        sub 6   ;CP 18
        ld e,6
        jr c,set_pad_values
        sub 6   ;CP 24
        ld e,8
        jr c,set_pad_values
        sub 6   ;CP 30
        ld e,10
        jr c,set_pad_values
        sub 6   ;CP 36 (32 pixels of paddle + 4 pixels of the ball)
        ld e,12
        jr c,set_pad_values
        jr real_move_ball

table_pad_values:
        .db 0,speed3,0,speed2,0,speed1,1,speed1,1,speed2,1,speed3

set_pad_values:
        ld d,0
        add hl,de
        ld a,(hl)
        add a,b
        ld (mode),a
        inc hl
        ld a,(hl)
        ld (speed_ball),a
        call play_it

;-------------------------------------------------------------------------------
; Movement of the ball
;-------------------------------------------------------------------------------
real_move_ball:
        ld a,(mode)     ;0-rght-up, 1-rght-down, 2-lft-up, 3-lft-down
        or a
        jr z,mode0
        dec a   ;CP 1
        jr z,mode1
        dec a   ;CP 2
        jr z,mode2

mode3:
        call short_mode         ;just to shorten routines mode0 to mode3
        add a,d
```

```
        ld (ball),a
        cp 192-12       ;border
        call nc,setmode2
        jr xmode1

mode0:
        call short_mode
        sbc a,d
        ld (hl),a
        cp 16           ;border
        call c,setmode1
        jr xmode2
mode1:
        call short_mode
        add a,d
        ld (hl),a
        cp 192-12       ;border
        call nc,setmode0
        jr xmode2
mode2:
        call short_mode
        sbc a,d
        ld (ball),a
        cp 16           ;border
        call c,setmode3

xmode1:         ;this is the speed level; 0, 1, 2
        ld a,(xmode)
        or a
        ld e,-1
        jr z,xmode1_add
        dec a   ;CP 1
        ld e,-2
        jr z,xmode1_add
        ld e,-3
xmode1_add:
        ld a,(ball+1)
        add a,e         ;we subtract ‘e’ pixels to X coordinate of the ball
        ld (ball+1),a
        cp 4
        jp c,player2scores
        jr write

xmode2:
        ld a,(xmode)
        or a
        ld e,1
        jr z,xmode2_add
        dec a
        ld e,2
```

```
        jr z,xmode2_add
        ld e,3
xmode2_add:
        ld a,(ball+1)
        add a,e         ;we add 'e' pixels to X coordinate of the ball
        ld (ball+1),a
        cp 5
        jr c,player1scores

write:
        call writey
        jp main_loop

writey:
        halt
        ld hl,6912      ;paddle 1
        ld a,(bat1)
        call $4d
        ld hl,6916
        ld a,(bat1)
        ld b,16
        add a,b
        call $4d

        ld hl,6920      ;paddle 2
        ld a,(bat2)
        call $4d
        ld hl,6924
        ld a,(bat2)
        ld b,16
        add a,b
        call $4d

        ld hl,6928      ;ball
        ld a,(ball)
        call $4d
        ld hl,6929      ;or INC HL
        ld a,(ball+1)
        call $4d

        ret

both_cont:
        call init_some
        ld bc,$0807
        call $47
        halt
        halt
        halt
        halt
```

```
        halt
        ld bc,$0107
        call $47
        jr write

;.....................................................................
;player 1 or 2 scores
;.....................................................................
player1scores:
score1:         .equ $+1
        ld a,$3e        ;$3e for better compression ratio
        inc a           ;INC score
        ld (score1),a
        call print_score        ;print it
        ld a,(score1)
        sub 5   ;CP 5 to check if player 1 wins
        jr z,win_1
        ld a,2   ;left-up
        ld (mode),a
        ld a,250
        ld (ball+1),a    ;X of ball
        jr both_cont

player2scores:
score2:         .equ $+1
        ld a,$3e
        inc a
        ld (score2),a
        call print_score
        ld a,(score2)
        sub 5   ;CP 5 to check if player 2 wins
        jr z,win_1
        xor a           ;rght-up
        ld (mode),a
        ld a,5
        ld (ball+1),a    ;X of ball
        jr both_cont
win_1:
        call silence
win_2:
        xor a
        call $d5
        or a
        jr z,win_2
        jp zero

play_it:
        call sound1
        ld a,6
        ld (sound_cnt),a
```

```
        ret

short_mode:
        call mode2ball
        ld hl,ball
        ld a,(hl)
        ret

setmode1:
        ld a,15         ;ball up-top
        ld (ball),a
        ld a,1
        jr short_set_pad
setmode0:
        ld a,192-14     ;ball down-top
        ld (ball),a
        xor a
        jr short_set_pad
setmode3:
        ld a,15         ;ball up-top
        ld (ball),a
        ld a,3
        jr short_set_pad
setmode2:
        ld a,192-14     ;ball down-top
        ld (ball),a
        ld a,2
        jr short_set_pad

short_set_pad:
        ld (mode),a
        call play_it
        ret

mode2ball:
        ld hl,(speed_ball)
        ld a,(hl)
        ld d,a
        ld e,0
        ret

;.....................................................................
;routine to print big scores 00-00
;.....................................................................
print_score:

        ld hl,6247      ;player 1
        ld (add_change),hl
        ld a,(score1)
        call print_player_score
```

```
        ld hl,6258      ;player 2
        ld (add_change),hl
        ld a,(score2)

print_player_score:
        inc a
        ld b,a
        ld hl,digit_00-5
loop_guess_digit:
        ld de,$0005
        add hl,de
        djnz loop_guess_digit
        ld (digit_add),hl

        ld b,e
loop_p1:
        push bc
        ld hl,(digit_add)
        ld a,(hl)
        inc hl
        ld (digit_add),hl

        ld hl,$d300-1
        bit 6,a
        call blue_green
        bit 5,a
        call blue_green
        bit 4,a
        call blue_green
        bit 3,a
        call blue_green
        bit 2,a
        call blue_green
        bit 1,a
        call blue_green
        bit 0,a
        call blue_green

        ld hl,$d300
        ld bc,7
add_change:   .equ $+1
        ld de,$0000    ;address to change
        call $5c

        ld hl,(add_change)
        ld de,32
        add hl,de
        ld (add_change),hl
```

```
        pop bc
        djnz loop_p1
        ret

blue_green:
        inc hl
        push af
        jr nz,print_green
        ld a,32 ;black square to buffer
        ld (hl),a
        pop af
        ret

print_green:
        ld a,'  i';red square to buffer
        ld (hl),a
        pop af
        ret

;-------------------------------------------------------------------------------
kronos:
        ld a,(xmode)
        cp 2
        ret z

clock:  .equ $+1
        ld hl,$3232
        ld de,2
        sbc hl,de       ;or LD DE,-2 / ADD HL,DE
        jr nc,kronos_
        jr xmode_change     ;we change speed (xmode)

kronos_:
        ld hl,(clock)
        dec hl
        ld (clock),hl

;if you run the game at 60hz you better load the correct values
;time_02 with 600 and time_03 with 60.

        ld b,0
time_02:    .equ $+1
        ld de,500       ;600
        call cifrar2
        ld hl,$d300-2
        call convertion
        ld hl,(digit_add)

        ld b,0
time_03:    .equ $+1
```

```
        ld de,50        ;60
        call cifrar2
    ld hl,$d300-1
        call convertion

        ld hl,$d300-2
        ld bc,2
        ld de,6144+15
        call $5c
        ret

xmode_change:
xmode:          .equ $+1
        ld a,$3e
        or a
        jr nz,xm1
        inc a
        ld (xmode),a
        ld hl,speed_buffer+3  ;+0, +3, +6
        call short_xm
        ld hl,1050
        ld (clock),hl
        ret
xm1:
        inc a
        ld (xmode),a
        ld hl,speed_buffer+6  ;+0, +2, +4
        call short_xm
        ret

cifrar2:
        and a
loop02:
        ld (digit_add),hl
        sbc hl,de
        inc b
        jr nc,loop02
        dec b
        ret
convertion:
        ld a,b
        add a,48
        ld (hl),a
        ret

;-------------------------------------------------------------------------
;some definitions
;-------------------------------------------------------------------------
speed_buffer:  .db 1,2,4,2,4,6,5,6,8
```

```
digit_00:       .db %01110111, %01010101, %01010101, %01010101, %01110111
digit_01:       .db %01110001, %01010001, %01010001, %01010001, %01110001
digit_02:       .db %01110111, %01010001, %01010111, %01010100, %01110111
digit_03:       .db %01110111, %01010001, %01010011, %01010001, %01110111
digit_04:       .db %01110101, %01010101, %01010111, %01010001, %01110001
digit_05:       .db %01110111, %01010100, %01010111, %01010001, %01110111
;digit_06:      .db %01110111, %01010100, %01010111, %01010101, %01110111
;digit_07:      .db %01110111, %01010001, %01010010, %01010010, %01110010
;digit_08:      .db %01110111, %01010101, %01010111, %01010101, %01110111
;digit_09:      .db %01110111, %01010101, %01010111, %01010001, %01110001
;digit_10:      .db %00010111, %00010101, %00010101, %00010101, %00010111
;digit_11:      .db %00010001, %00010001, %00010001, %00010001, %00010001
;digit_12:      .db %00010111, %00010001, %00010111, %00010100, %00010111
;digit_13:      .db %00010111, %00010001, %00010011, %00010001, %00010111
;digit_14:      .db %00010101, %00010101, %00010111, %00010001, %00010001
;digit_15:      .db %00010111, %00010100, %00010111, %00010001, %00010111
;actually, from 00 to 15 can be defined but ...

;sprites datas + attributes.....................................................;
spr_definition:
        ; Sprite patterndata for sprite 1
        .db $e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0
        .db $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        ; Sprite patterndata for sprite 2
        .db $e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0
        .db $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        ; Sprite patterndata for sprite 3
        .db $e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0
        .db $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        ; Sprite patterndata for sprite 4
        .db $e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0,$e0
        .db $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        ; Ball
        .db $60,$F0,$F0,$60,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
        .db $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00

spr_attributes:
        .db 88,8,0,15           ;sprite 1 (Y,X,plane,colour)
        .db 88+16,8,4,15        ;sprite 2 (Y,X,plane,colour)
        .db 88,245,8,15         ;sprite 3 (Y,X,plane,colour)
        .db 88+16,245,12,15     ;sprite 4 (Y,X,plane,colour)
        .db 88+16,235,16,15     ;sprite 5 (Y,X,plane,colour)   ball
;.....................................................................;
;a small sound for ball rebounds
;.....................................................................;

sound1:
        ld e,$20
        xor a
        call $93
```

```
        ld e,%00111110
        ld a,7
        call $93
        ld e,16
        ld a,8
        call $93
        ld e,10
        ld a,13
        call $93
        ld e,%68
        ld a,11
        call $93
        ret

silence:
        ld e,0
        ld a,8
        call $93
        ret

;some loose routines
init_some:
        ld hl,speed1
        ld (speed_ball),hl
        ld a,96
        ld (ball),a       ;Y
        ld hl,550
        ld (clock),hl
        ld hl,speed_buffer+0  ;+0, +2, +4
        call short_xm
        ld (xmode),a   ;0=1, 1=2, 2=3 (X speed)
        ret

short_xm:
        ld de,speed1
        ld bc,4
        ldir
        xor a
        ret

endProgram:
.end
```