; This version is in Dewar Assembler and supports the Mouse
;
; This program implements a simple one player game on the IBM PC.
; It requires the color/graphics adapter (although it operates only
; in character mode, and could be run on a monochrome monitor with
; a suitable change in the screen attributes).
;
; The general picture is a blue screen with green walls on the top
; and on the two sides. On the bottom line is a paddle which consists
; of a red strip, five character positions long, which can be moved
; from side to side using the mouse.
;
; On pressing a mouse button, a "ball" (a yellow happy face
; on a blue background) is released and travels around bouncing
; off the walls and paddle until it is missed and falls through the
; bottom. A game lasts three balls, after which control returns to DOS.
;jmp main
;
; Definitions of colors to be used
;
WALLC EQU 22H ; wall is all green
FIELD EQU 1FH ; playing field is white on blue
PADLC  EQU 44H ; paddle is all red
BALLC  EQU 1EH ; ball is yellow on blue background
NORMC  EQU 07H ; grey on black (for return to DOS)
;
; Data areas, first the position and direction of the ball. If no ball is
; in play, then BROW is zero and the other fields are irrelevant. If
; the ball is in play, then BROW, BCOL give its current position, and
; BVDIR, BHDIR give its direction (one of four possibilities since the
; ball always moves in a diagonal direction.

BROW DB ? ; current line of ball (0 = no ball)
BCOL DB ? ; current col of ball
BVDIR DB ? ; vertical direction (-1=up, +1=down)
BHDIR DB ? ; horiz direction (-1=left, +1=right)

; The following location gives the current paddle position. It is in the
; range 3-76, since it gives the position of the middle of the paddle.
; We start out with the paddle in the middle of the screen

PADPOS DB 40 ; current paddle position
PADML EQU 4 ; minimum position
PADMX EQU 75 ; maximum position

; Count balls in play, and score (count of bounces off paddle)

BCOUNT DB 0 ; number of balls played
SCORE DW 0 ; score so far

; Initial notes of funeral march (used when ball is missed)
; See TUNE routine for definition of the format of this list

FUNR1 DW 7400,262,600,0
DW 5300,262,600,0
DW 1400,262,600,0
DW 7400,262,600,0
DW 0 ; end of first half of tune

FUNR2 DW 5300,311,600,0
DW 1400,294,600,0
DW 5300,294,600,0
DW 1400,262,600,0
DW 5300,262,600,0
DW 1400,247,600,0
DW 7400,262,600,0
DW 0

; Beep data for ball bouncing (see NOTE routine)

BEEPL EQU 1000 ; frequency for beep off left wall
BEEPR EQU 1500 ; frequency for beep off right wall
BEEPT EQU 750 ; frequency for beep off top
BEEPP EQU 1250 ; frequency for beep off paddle
BDUR EQU 200 ; length of bounce beeps (0.07 secs)

; Here is the main program. We use a top down approach in which
; the functions are distributed to lower level modules. The first step
; is to initialize the screen (playing field and wall display).

MAIN: CALL INIT ; initialize screen

; Loop to play 3 balls

MAIN1: INC BCOUNT ; bump count of balls played
CALL PLAY ; play one ball

MOV AH, 1 ; check if key pressed - means quit
INT 16H
JNZ GETKEY

CMP BCOUNT,3 ; check three balls played
JNE MAIN1 ; loop back for another if not
LEA SI, FUNR2 ; finally, play second half of tune
CALL TUNE
JMP CLEANUP

GETKEY: MOV AH, 0 ; key was pressed -- read it
INT 16H ; and throw it away!

; Reset screen to normal, and return to DOS

CLEANUP:
MOV DX,0 ; reset screen to normal color
MOV CX,25*80
MOV BL,NORMC
MOV   AL,' '  
CALL  PAINT  
MOV   AX, 4C00h  
INT   21H ; then return to DOS

; Initialization subroutine. Make playing field and walls

INIT      PROC
PUSH  AX ; save registers
PUSH  BX
PUSH  CX
PUSH  DX
MOV   DX,0 ; home cursor
MOV   CX,25*80 ; clear screen to all blue
MOV   BL,FIELDC ; set proper color
MOV   AL,' '  
CALL  PAINT ; paint playing field
MOV   BL,WALLC ; set color for wall
MOV   CX,80 ; paint top wall
CALL  PAINT

; Loop to paint side walls

INIT1: ADD   DH,1 ; bump to next line
MOV   DL,0 ; set left column
MOV   CX,1 ; one wall piece on left
CALL  PAINT
MOV   DL,79 ; one wall piece on right
CALL  PAINT
CMP   DH,24 ; loop till all set
JNE   INIT1

; Walls are complete, initialize ball and score counts and we are done

; CALL  UTEXT ; initialize text messages
MOV   AH, 0 ; initialize mouse
INT   33H
POP   DX ; restore registers
POP   CX
POP   BX
POP   AX
RET ; then return to caller

INIT      ENDP

; Main controlling routine for playing one ball

PLAY      PROC
PUSH  AX ; save registers
PUSH  BX
PUSH  CX
PUSH  DX
CALL DISPD ; display initial position
;
; Wait for ball to be launched (any key pressed)
;
PLAY1:
CALL NEWP ; update paddle position
CALL DISPD
CALL DELAY
MOV AH, 1 ; check for quitting
INT 16H
JNZ ENDPL
MOV AX, 3 ; check for button pressed?
INT 33H
OR BX, BX
JZ PLAY1 ; loop if no button pressed
;
; Here we put the ball into play

MOV AX, BEEPP ; set beep off paddle
MOV DX, BDUR
CALL NOTE
MOV BROW, 23 ; ball is on bottom line, just above paddle
MOV AL, PADPOS
MOV BCOL, AL
MOV BVDIR, -1 ; going up
MOV BHDIR, +1 ; to the right
;
; Loop through ball and paddle positions as play proceeds

PLAY2: CALL NEWP ; update paddle position
CALL DISPD ; display paddle in current position
MOV DH, BROW ; point to ball position
MOV DL, BCOL
MOV AL, 2 ; ball is ASCII 2 (happy face)
MOV BL, BALLC ; set proper color
MOV CX, 1
CALL PAINT ; display ball at current position
CALL DELAY ; wait for a moment
MOV AL, '' ; clear old ball position
MOV BL, FIELDC
CALL PAINT
CALL UTEXT ; update text messages
CALL NEWP ; set new paddle position
CALL NEWB ; set new ball position and direction
CMP BROW, 0 ; ball still in play?
JNZ PLAY2 ; loop if so
;
; Ball is out of play, we are done

ENDPL: POP DX ; restore registers
POP CX
POP BX
POP AX
213       RET               ; return to caller
214       PLAY              ENDP
215       
216       ; Routine to display paddle. We paint the paddle first, then the
217       ; spaces on either side. Things are done in this order to avoid turning
218       ; off the paddle completely, which might cause some flicker.
219       
220       DISPD   PROC
221       PUSH AX          ; save registers
222       PUSH BX
223       PUSH CX
224       PUSH DX
225       
226       ; Paint the paddle itself
227       
228       MOV   DH,24          ; set left position of paddle
229       MOV   DL,PADPOS
230       SUB   DL,3
231       MOV   CX,7           ; paint seven squares for paddle
232       MOV   BL,PADLC
233       MOV   AL,' '
234       CALL  PAINT          ; display paddle
235       
236       ; Paint space to right of paddle
237       
238       ADD   DL,7           ; point to space on right
239       MOV   CX,79          ; space on right is 79
240       SUB   CL,DL          ; minus starting pos of space
241       MOV   BL,FIELDC      ; set proper color for space
242       MOV   AL,' '
243       CALL  PAINT          ; paint spaces on the right
244       
245       ; Paint space to left of paddle
246       
247       MOV   DL,1           ; point to space on left
248       MOV   CL,PADPOS      ; space on left is paddle position
249       SUB   CL,4           ; minus 3 for start of space
250       CALL  PAINT         ; paint space on the left
251       
252       ; Paddle display is complete
253       
254       MOV   DH, 25         ; move the cursor off the screen
255       MOV   DL, 0
256       CALL  SETPOS
257       POP   DX          ; restore registers
258       POP   CX
259       POP   BX
260       POP   AX
261       RET               ; return
262       DISPD   ENDP
263       
264       ; Routine to set new paddle position
265       ;
266  NEWP  PROC
267      PUSH  AX          ; save registers
268       ;
269      ;  See if mouse moved
270       ;
271      MOV  AX,3          ; get mouse position
272      INT   33H
273      SHR  CX, 1        ; col for text mode resolution
274      shr  cx, 1
275      shr  cx, 1        ;i.e., shr cx, 1 since shr cx, 3 not supported!
276
277      CMP   CX, PADMN
278      JAE   NEXT
279      MOV   CX, PADMN
280  NEXT:  CMP   CX, PADMX
281      JBE   NEXT1
282      MOV   CX, PADMX
283  NEXT1:
284      MOV   PADPOS, CL
285      POP   AX          ; restore registers
286      RET               ; return to caller
287  NEWP  ENDP
288       ;
289 ;  Routine to set new ball position, first deal with vertical position
290       ;
291  NEWB  PROC
292      PUSH  AX          ; save registers
293      PUSH  BX
294      PUSH  DX
295      PUSH  SI
296      MOV   BL,BROW     ; get current line
297      ADD   BL,BVDIR    ; adjust by adding direction to get new line
298      JNZ   NEWB1        ; jump if not on top wall
299       ;
300 ;  If we bounce off top wall, reset direction to down
301       ;
302      MOV   AX,BEEPT     ; set beep off top
303      MOV   DX,BDUR
304      CALL   NOTE
305      MOV   BL,1         ; reset position to top of play area
306      MOV   BVDIR,+1     ; reset direction to down
307       ;
308 ;  Now deal with horizontal position
309       ;
310  NEWB1:  MOV   BROW,BL  ; first store new vertical position
311      MOV   BL,BCOL     ; get current char
312      ADD   BL,BHDIR    ; add horizontal direction to get new char
313      JNZ   NEWB2       ; jump if not at left wall
314       ;
315 ;  If we are at left wall, reset horizontal direction for bounce
316       ;
317      MOV   AX,BEEPL     ; set beep off left
318      MOV   DX,BDUR
CALL NOTE
MOV BL,1 ; reset position to left of play area
MOV BHDIR,+1 ; reset direction to right
;
; If at right wall, reset direction for bounce
;
NEWB2: CMP BL,79 ; jump if not at right wall
JNE NEWB3
MOV BL,78 ; else reset position to right of play area
MOV BHDIR,-1 ; reset direction to left
MOV AX,BEEPR ; set beep off right
MOV DX,BDUR
CALL NOTE
;
; Here we have new position, check for paddle bounce
;
NEWB3: MOV BCOL,BL ; store new character position
CMP BROW,24 ; on bottom line?
JNE NEWB8 ; jump if not, all set
;
; Check for case of bouncing off paddle (ball is on bottom line).
; First we see if the ball is hitting the center of the paddle.
;
MOV BH,PADPOS ; get paddle position
MOV BL,BCOL ; get horizontal position of ball
CMP BL,BH ; is ball hitting center of paddle?
JE NEWB4 ; jump if so
;
; Test to see if ball is hitting either of the paddle squares on the left
;
DEC BH ; point to square to left of center
CMP BL,BH ; did ball hit this square?
JE NEWB5 ; jump if so
DEC BH ; point to leftmost paddle square
CMP BL,BH ; did ball hit this square?
JE NEWB5 ; jump if so
DEC BH ; one more time
CMP BL,BH
JE NEWB5
;
; Test to see if ball is hitting either of the paddle squares on the right
;
ADD BH,4 ; point to square to right of center
CMP BL,BH ; did ball hit this square?
JE NEWB6 ; jump if so
INC BH ; point to rightmost paddle square
CMP BL,BH ; did ball hit this square?
JE NEWB6 ; jump if so
INC BH
CMP BL,BH
JE NEWB6
;
; This is where ball has missed the paddle and goes out of play
; MOV BROW,0          ; indicate ball no longer in play
374 MOV DH,25         ; set cursor off screen
MOV DL,0
CALL SETPOS
377 LEA SI,FUNR1      ; play funeral march (1st half)
378 CALL TUNE
379 JMP NEWB8         ; exit
380 ;
381 ;  Ball hit middle of paddle, keep horizontal direction
382 ;
383 NEWB4: MOV BROW,23 ; reset line to bottom of play area
384 JMP NEWB7         ; exit
385 ;
386 ;  Ball hit left of paddle, reset direction to left
387 ;
388 NEWB5: MOV BROW,23 ; reset line to bottom of play area
389 MOV BHDIR,-1     ; reset direction to left
390 JMP NEWB7        ; exit
391 ;
392 ;  Ball hit right of paddle, reset direction to right
393 ;
394 NEWB6: MOV BROW,23 ; reset line to bottom of play area
395 MOV BHDIR,+1      ; reset direction to right
396 ;
397 ;  Merge here for ball bouncing off paddle, horizontal direction set
398 ;
399 NEWB7: MOV AX,BEEPP ; set beep off paddle
400 MOV DX,BDUR       ;
401 CALL NOTE
402 MOV BVDIR,-1      ; reset direction to up
403 INC SCORE         ; bump score
404 ;
405 ;  Exit here after setting new ball position and direction
406 ;
407 NEWB8: POP SI       ; restore registers
408 POP DX
409 POP BX
410 POP AX
411 RET                ; return
412 NEWB      ENDP
413 ;
414 ;  Routine to update text messages (ball count and score)
415 ;
416 UTEXT PROC
417 PUSH AX           ; save registers
418 PUSH DX
419 ;
420 ;  Output score
421 ;
422 MOV DX,0204H       ; set position for score
423 CALL SETPOS       ;
424 MOV AL,'S'      ; set message SCORE:
CALL WCHAR
MOV AL,'C'
CALL WCHAR
MOV AL,'O'
CALL WCHAR
MOV AL,'R'
CALL WCHAR
MOV AL,'E'
CALL WCHAR
MOV AX,SCORE         ; output score
CALL UTXTI

; Output ball count

MOV DL,69          ; set position for ball count
CALL SETPOS
MOV AL,'B'         ; set message BALL:
CALL WCHAR
MOV AL,'A'
CALL WCHAR
MOV AL,'L'
CALL WCHAR
MOV AL,'L'
CALL WCHAR
MOV AL,':'
CALL WCHAR

; Complete output of ball count

MOV AL,BCOUNT      ; get ball count
OR AL,'0'         ; in ASCII
CALL WCHAR          ; write ball count
POP DX             ; restore registers
POP AX
RET               ; return to caller

UTEXT      ENDP

; Routine to output integer value from AX

UTXTI    PROC
PUSH  CX             ; save registers
PUSH  DX
MOV  CX,10          ; divide input argument by 10
SUB  DX,DX
DIV   CX

; We have the number / 10 in AX, and number mod 10 in DX. If the
value in AX is non-zero, we make a recursive call to output it

; test non-zero quotient
JZ UTX2 ; jump if zero
CALL UTXTI ; else output upper digits

; Here we output the remainder from the division as the last digit

UTX2: MOV AL,DL ; remainder to AL
OR AL,'0' ; convert to ASCII
CALL WCHAR ; write last digit
POP DX ; restore registers
POP CX
RET ; return to caller
UTXTI ENDP

; Delay routine, if we had no delay, then things would move so rapidly
; we could not tell what was going on, and could not move the paddle
; fast enough to keep up with the play!

DELAY PROC
PUSH AX ; save registers
PUSH DX
MOV DH,25 ; get cursor off screen area
MOV DL,0 ; (cleaner appearance)
CALL SETPOS
SUB AX,AX ; zero frequency for rest
MOV DX,1800 ; delay of 0.06 secs is reasonable
CALL NOTE ; execute delay
POP DX ; restore registers
POP AX
RET ; return to caller
DELAY ENDP

; Routine to play tune on speaker

(TSI) Pointer to tune list
CALL TUNE

; The tune list is a series of word pairs. The first word is the
duration in units of 1/100th of a second, and the second word
is the frequency. An entry with zero duration ends the tune list
and a zero frequency is a rest (period of silence).

TUNE PROC
PUSH AX ; save registers
PUSH DX
; Loop through notes of the tune
TUN1: LODSW ; load duration
OR AX,AX ; test zero duration ending the list
JZ TUN2 ; if so, end of tune

531 ;
532 ; Play next note
533 ;
534 XCHG AX,DX ; put duration in DX
535 LODSW ; load frequency
536 CALL NOTE ; play the note
537 JMP TUN1 ; and loop back
538 ;
539 ; Here at end of tune
540 ;
541 TUN2: POP DX ; restore registers
542 POP AX
543 RET ; return to caller
544 TUNE ENDP
545 ;
546 ; Routine to play note on speaker
547 ;
548 ; (AX) Frequency in Hz (32 - 32000)
549 ; (DX) Duration in units of 1/100 second
550 ; CALL NOTE
551 ;
552 ; Note: a frequency of zero, means rest (silence) for the indicated
553 ; time, allowing this routine to be used simply as a timing delay.
554 ;
555 ; Definitions for timer gate control
556 ;
557 CTRL EQU 61H ; timer gate control port
558 TIMR EQU 00000001B ; bit to turn timer on
559 SPKR EQU 00000010B ; bit to turn speaker on
560 ;
561 ; Definitions of input/output ports to access timer chip
562 ;
563 TCTL EQU 043H ; port for timer control
564 TCTR EQU 042H ; port for timer count values
565 ;
566 ; Definitions of timer control values (to send to control port)
567 ;
568 TSQW EQU 10110110B ; timer 2, 2 bytes, sq wave, binary
569 LATCH EQU 10000000B ; latch timer 2
570 ;
571 ; Define 32 bit value used to set timer frequency
572 ;
573 FRHI EQU 0012H ; timer frequency high (1193180 / 256)
574 FRLO EQU 34DCH ; timer low (1193180 mod 256)
575 ;
576 NOTE PROC
577 PUSH AX ; save registers
578 PUSH BX
579 PUSH CX
580 PUSH DX
581 PUSH SI
582 MOV BX,AX ; save frequency in BX
583 MOV CX,DX ; save duration in CX
We handle the rest (silence) case by using an arbitrary frequency to program the clock so that the normal approach for getting the right delay functions, but we will leave the speaker off in this case.

```
    MOV SI,BX          ; copy frequency to BX
    OR BX,BX          ; test zero frequency (rest)
    JNZ NOT1           ; jump if not
    MOV BX,256         ; else reset to arbitrary non-zero

    NOT1: MOV AL,TSQW          ; set timer 2 in square wave mode
    OUT TCTL,AL
    MOV DX,FRHI          ; set DX:AX = 1193180 decimal
    MOV AX,FRLO          ; = clock frequency
    DIV BX               ; divide by desired frequency
    OUT TCTR,AL          ; output low order of divisor
    MOV AL,AH            ; output high order of divisor
    OUT TCTR,AL

    ; Turn the timer on, and also the speaker (unless frequency 0 = rest)
    IN AL,CTRL          ; read current contents of control port
    OR AL,TIMR          ; turn timer on
    OR SI,SI            ; test zero frequency
    JZ NOT2             ; skip if so (leave speaker off)
    OR AL,SPKR          ; else turn speaker on as well

    NOT2: OUT CTRL,AL          ; rewrite control port
    XCHG AX,BX           ; frequency to AX
    MUL CX               ; frequency times secs/100 to DX:AX
    MOV CX,100           ; divide by 100 to get number of beats
    DIV CX
    SHL AX,1             ; times 2 because two clocks/beat
    XCHG AX,CX           ; count of clock cycles to CX

    ; Loop through clock cycles
    NOT3: CALL RCTR          ; read initial count

    ; Loop to wait for clock count to get reset. The count goes from the value we set down to 0, and then is reset back to the set value

    NOT4: MOV DX,AX          ; save previous count in DX
    CALL RCTR             ; read count again
    CMP AX,DX             ; compare new count : old count
    JB NOT4               ; loop if new count is lower
    LOOP NOT3             ; else reset, count down cycles

    ; Wait is complete, so turn off clock and return
```
; read current contents of port
AND AL,0FFH-TIMR-SPKR ; reset timer/speaker control bits
; note that the above statement is an equation
OUT CTRL,AL ; rewrite control port
POP SI ; restore registers
POP DX
POP CX
POP BX
POP AX
RET ; return to caller

NOTE ENDP

; Routine to read count, returns current timer 2 count in AX

RCTR PROC
MOV AL,LATCH ; latch the counter
OUT TCTL,AL ; latch counter
IN AL,TCTR ; read lsb of count
MOV AH,AL
IN AL,TCTR ; read msb of count
XCHG AH,AL ; count is in AX
RET ; return to caller
RCTR ENDP

; Routine to set cursor position

; (DH,DL) Cursor line, column
; CALL SETPOS

SETPOS PROC
PUSH AX ; save registers
PUSH BX
MOV BH,0 ; set cursor
MOV AH,2
INT 10H
POP BX ; restore registers
POP AX
RET ; return to caller
SETPOS ENDP

; Routine to paint characters on screen

; (DH,DL) Line, column of start of area to paint
; (CX) Number of characters to paint
; (BL) Required color
; (AL) Character code
; CALL PAINT

PAINT PROC
PUSH AX ; save registers
PUSH BX
CMP CX,0 ; skip if no chars to paint
JE PAINT1
CALL SETPOS ; set cursor position
MOV BH,0 ; write the characters
MOV AH,9
INT 10H

; Here with characters painted
PAINT1: POP BX ; restore registers
POP AX
RET ; return to caller
PAINT ENDP

; Routine to write one character to screen
; (AL) Character to write
CALL WCHAR

WCHAR PROC
PUSH AX ; save registers
PUSH BX
PUSH BP
SUB BH,BH ; set page zero
MOV AH,14 ; set code for write teletype
INT 10H ; write character
POP BP ; restore registers
POP BX
POP AX
RET ; return to caller
WCHAR ENDP