משחקים בתווים

בדף עבודה זה נתנסה בעבודה במצב טקסט.

ניצור מחרוזת המשתרעת על פני כמה שורות, נוסיף שחקן ונזיז אותו על פני המסך

# מצב טקסט

מסך ה – dos box מחולק לשורות ועמודות, ישנם שני מצבים של תצוגת הטקסט על המסך:

* מצב של 80 עמודים ו 25 שורות (המצב ברגיל בו אנו עובדים)
* ומצב של 40 עמודות ו – 25 שורות.

הפסיקות הקובעות את מצב תצוגת המסך והמתנה למקש

|  |  |
| --- | --- |
| כניסה למוד טקסט 25\*80 25 שורות ו - 80 עמודות **תווים** | mov al, 03h  mov ah, 0 int 10h |
| כניסה למוד טקסט 25\*40 25 שורות ו - 40 עמודות **תווים** | mov al, 13h mov ah, 0 int 10h |
| המתנה למקש  התו נשמר לרגיסטר al, והתכנית עוצרת עד להקשה על מקש. | mov ah, 0h  int 16h |

כתבו תכנית קצרה שתעזור לנו לעבוד בצורה רציפה ונוחה.

* מעבר למוד של 40\*25
* המתנה למקש
* יציאה למוד טקסט רגיל.

**הריצו ובדקו את מצב המסך.**

תקבלו מסך גרפי שחור ובהקשה על מקש נצא למוד טקסט רגיל.

# יצירת מחרוזת המשתרעת על כמה שורות.

מחרוזת בנויה ממספר תווים, כאשר כל תו גודלו בית והסימן $ מסמן לאסמבלי על סוף המחרוזת.

דוגמה למחרוזת ב – datasegment:

memu1 db ' text \*\*\*\* ', 10, 13

db '++++++++++++++', 10, 13

שם משתנה המחרוזת,

גודל בית

'טקטסט המחרוזת בגרשיים'

10**,**13

הספרות 10, ו – 13 מסמנות לאסמבלי סוף שורה ומעבר לשורה הבאה

בסיום המחרוזת לא לשכוח את הסימן '$'המסמן לאסמבלי סוף המחרוזת.

db '+ +', 10, 13

db '+ +', 10, 13

db '+ +', 10, 13

db '+ +', 10, 13

db '++++++++++++++', 10, 13

db ' ', 10, 13

db 'press p to play', 10, 13

db 'press Esc', 10, 13, ,'$'

נפעיל את הפסיקה להדפסת מחרוזת ונבדוק את התוצאה על המסך.

את הפסיקה נכתוב לפני הפסיקה של ההמתנה למקש כדי שנוכל לראות את התוצאה.

בהקשה על מקש נצא מהתכנית.

; print string to screen

mov dx, offset memu1  
mov ah, 9h  
int 21h

# מיקום תו על המסך

ראשית ממקמים את הסמן על גבי המסך ואחר כך מציירים את התו הרצוי:

לשם כך נשתמש בשלושה משתנים אשר יציינו את:

* מיקום התו בשורה – צירה ה – x

y\_row db 4

x\_column db 6

colour db 11 ; Light Cyan

* מיקום התו בעמודה ציר ה – y
* צבע התו

|  |  |
| --- | --- |
| **פסיקה למיקום סמן על המסך.**  mov bh, 0  mov dh, [y\_row] ; Row number  mov dl, [x\_column] ; Column number  mov ah, 2  int 10h | **פסיקה המשרטטת את השחקן תו ascii - 3**  mov ah, 9 ; Interrupt code  mov al, 3 ; Character to display -asci code  mov bh,0h ; Page - always 0  mov bl, [color]; BL = Foreground  mov cx, 1 ; Number of times to write character  int 10h ;int 10h Bios |

כתבו שתי פרוצדורות למיקום סמן וציור שחקן.

|  |  |
| --- | --- |
| מיקום סמל אל פני המסך:  proc Cursor\_Location  ;Place the cursor on the screen  mov bh, 0  mov dh, [y\_row] ; in row  mov dl, [x\_column]  mov ah, 2  int 10h  ret  endp Cursor\_Location | ציור תו על המסך:  proc DrawPlayer  ; draw player in cursor position  mov ah, 9  mov al, 2 ;AL = character to display  mov bh, 0h ;BH=Page  mov bl, [colour] ; BL = Foreground  mov cx, 1 ; number of times to write character  int 10h ; Bois -> show the character  ret  endp DrawPlayer |

נזמן את הפרוצדורות לאחר שרטוט לוח המשחק (המחרוזת שלנו), ונריץ את התכנית.

call Cursor\_Location

call draw\_player

**הריצו ובדקו שאתם רואים על המסך את השחקן**

**שמרו את הקובץ, בשם חדש והמשיכו בקובץ החדש.**

# הוספת מקש

נוסיף לתכנית שלנו :

תווית המתנה למקש

בדיקה האם הוקש מקש במקלדת

אם הקישו על מקש p נתחיל במחשק קפיצה לתווית Move

אם הקישו על מקש Esc נצא מהמשחק קפיצה לתווית TheEnd

קפיצה לתווית – המתנה למקש

מבנה התנית לאחר הדפסת המחרוזת למסך

call Cursor\_Location

call DrawPlayer

; Read character and save it to AL

WaitForCharacter:

mov ah, 0h

int 16h

; Was Esc pressed?

cmp al, 27

je TheEnd

; Was p pressed?

cmp al, 'p'

je Move

jmp WaitForCharacter

Move:

TheEnd:

; Return text mode 80x25 (x,y)

mov al, 03h

mov ah, 0

int 10h

**הריצו ובדקו.**

שימו לב 🎔: רק כאשר נקיש על מקש ה – Esc או מקש p נצא מהתכנית.

**שמרו את הקובץ, בשם חדש והמשיכו בקובץ החדש.**

# הזזת השחקן

כדי להזיז את השחקן "מוחקים אותו":

* מציירים את השחקן במיקום שהוא נמצא בצבע שחור (הערך של הצבע השחור הוא : 0)
* מזיזים את הסמן למיקום הבא
* ממקמים את הסמן במיקום הבא
* משנים את המשתנה של color לצבע
* מציירים את השחקן בצבע במיקום החדש.

כדי לזיז את השחקן ימינה, יש להוסיף 1 למיקום שלו על ציר ה – X ⇦ [x\_column]

ניצור פרוצדורה של הזזת השחקן ימינה.

proc RighttKeyPress

mov [colour], 0 ; Bleck color

call DrawPlayer ; Draw black character

add [x\_column], 1 ; move player >>

mov [colour], 11 ; Color to character

call Cursor\_Location ; Move cursor to new location

call DrawPlayer ; Draw character in color

ret

endp RighttKeyPress

נבדוק בתכנית הראשית האם הקישו על מקש ⇦ חץ ימני.

מתחת לתווית של **Move** ⇦ התווית שקבענו לתחילת המשחק, נבדוק האם הקישו על מקש

נבדוק את ה – **T**ype **A**head **B**uffer (מקום בזיכרון בו המעבד שומר את המידע מהמקלדת.

עבור כל תו המעבד שומר את ה Scan Code ואת התרגום שלו ל ASCII).

**Move:**

; Check Type Ahead Buffer Status

mov ah, 1h

int 16h

jz Move ; no key

אם יש תו מוכן, al ו-ah יקבלו את ערכי ה־ASCII וה־code scan של התו.

; Wait for char – read character and save it to register al

mov ah, 0h

int 16h

נבדוק מהו התו:

אם התו חץ ימיני ⇦ נקפוץ לתווית **RighttKey**   
(אחרי תווית זו נזמן את הפעולה של הזזת השחקן ימינה, ונקפוץ שוב לתווית Move נמתין להקשת המקלדת הבאה)

; Was Esc pressed?

cmp al, 27

je theEnd

; Was right Key Pressed?

תווית **RightKey:**

תעבור לפעולה של בדיקת התו הבא, וציור השחקן במיקום החדש. לאחר החזרה נעבור לתווית **Move:** כדי להמתין למקש הבא

cmp ah, 4dh

je RighttKey

jmp Move

RightKey:

call RightKeyPress

jmp Move

theEnd:

; return text mode 80x25 (x,y)

mov al, 03h

mov ah, 0

**הריצו ובדקו.**

**שמרו את הקובץ, בשם חדש והמשיכו בקובץ החדש.**

int 10h

# בדיקת מיקום השחקן

כאשר משחקים במבוך אנו לא רוצים שהשחקן יגע בגבולות של המבוך / יפסל כאשר הוא נגוע בהם.

השחקן זז בעזרת החיצים, לכן לפני שנעביר את השחקן למיקום הבא:

נעביר את הסמן למיקום

* נבדוק מהו ערך התו במיקום החדש
* אם התן הוא רווח (ריק) נעביר את השחק למיקום החדש.
* אם במיקום החדש יש סימן של גבול המבוך, לא נאפשר לשחקן לזוז (נחזיר את הסמן למיקום הקודם שלו).

פסיקה הבודקת מהו ה"תו" במיקום של הסמן ושמה את הערך ברגיסטר al.

; check the character on coordinates

mov bh, 0h ; Page=1

mov ah, 08h ; Read character function

int 10h ;return the character value to AL

**נשנה את הפעולה המזיזה את השחקן ימינה.**

* נזיז את המשתנה האחראי על מיקום השחקן בשורה (ציר x) למיקום הבא
* נזיז את הסמן למיקום הבא
* נבדוק מהו הערך של התו במיקום החדש של הסמן
* אם הערך הוא לא "+" הגבול של המבוך
* נעבור לתווית של המשך תזוזת השחקן (הזזת הסמן למיקום הקודם, ציור שחק שחור הזזת השחקן למיקום החדש ציור שחקן צבעוני.
* אם הערך הוא "+" גבול, נחזיר את הסמן למיקום הקודם ונצא מהפעולה של הזזת השחקן ימינה.

proc RightKeyPress

inc [x\_column]

call Cursor\_Location ; move cursor to new place

;check the character on coordinates

mov bh, 0h ; Page=1

mov ah, 08h ; Read character function

int 10h ;Return the character value to AL

cmp al, '+'

jne MovePlayerRigt ; Jump to draw player

dec [x\_column]

call Cursor\_Location

jmp EndRightKeyPress

MovePlayerRigt:

dec [x\_column]

call Cursor\_Location ; Return cursor to player position

mov [colour], 0

call DrawPlayer

add [x\_column], 1

mov [colour], 11

call Cursor\_Location

call DrawPlayer

EndRightKeyPress:

**הריצו ובדקו.**

**שמרו את הקובץ, בשם חדש והמשיכו בקובץ החדש.**

ret

endp RightKeyPress

## תזוזה של השחקן

צרו פעולות לתזוזת השחקן

* חץ שמאלי ⇦ (4bh)
* חץ מעלה ⇧ (48h)
* חץ מטה ⇩ (50h)

שימו לב 🎔: באסמלי קשה למצוא טעויות.

המשיכו את התכנית שלב אחרי שלב,

בכל שלב שמרו את קטע התכנית העובד בשם חדש והמשיכו לכתוב את קטע הקוד בתכנית החדשה.

* צרו פעולה לתזוזת השחקן
* צרו תנאי המפעיל אותה ע"י בדיקה אם הקישו על המקש במקלדת

; Was right Key Pressed?

cmp ah, 4dh

je RightKey

שימו 🎔 לדוגמה:

מיקמתי את הבדיקה של החץ השמאלי לאחר הבדיקה של החץ הימני.

; Was left Key Pressed?

cmp ah, 4bh

je LeftKey

* קפצו לתווית המפנה לפעולה.

RightKey:

call RightKeyPress

שימו 🎔 לדוגמה:

מיקמתי את התווית של החץ השמאלי לאחר התווית של החץ הימני.

jmp Move

LeftKey:

call LeftKeyPress

jmp Move

* שמרו והריצו.

לאחר שעובד, שמרו לקובץ חדש והמשיכו לכיוונים נוספים.

# הרחבות

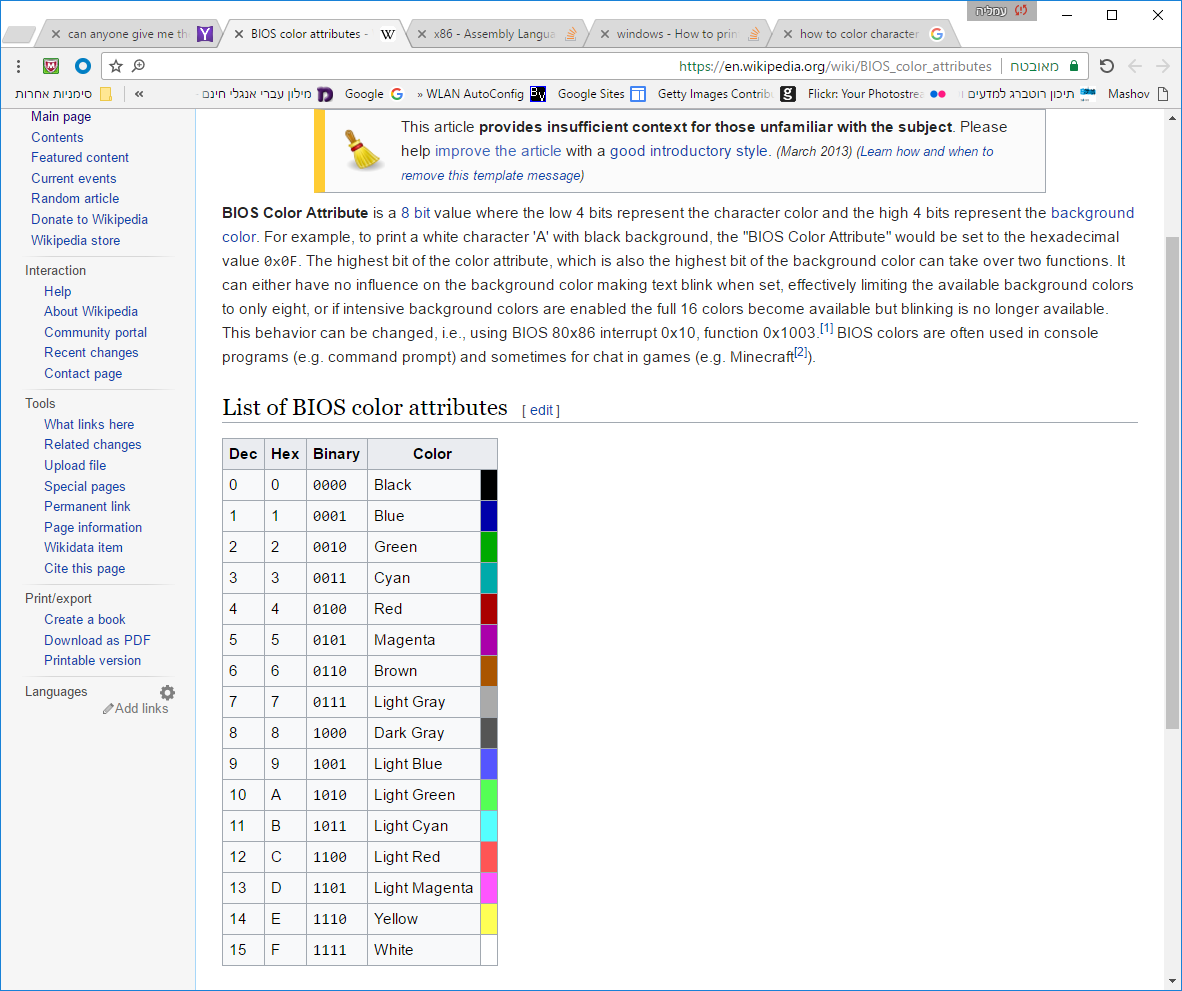
שפרו את התכנית.

ציירו מבוך בסימנים אחרים על פני כל המסך.

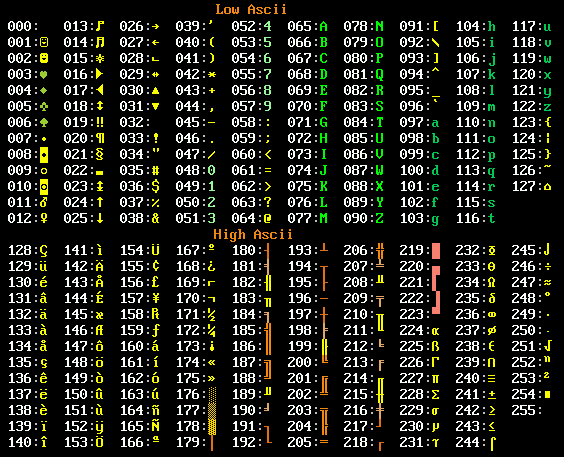
שנו את המשחק, אם השחקן נוגע בדופן המבוך הוא חוזר לנקודת ההתחלה

הוסיפו ניקוד

ועד כיד הדמיון הטובה עליכם.

טבלת צבעים

טבלת תווים (Ascii)



משתנים:

DATASEG

y\_row db 4

x\_column db 6

colour db 11

memu1 db ' text \*\*\*\* ', 10, 13

db '++++++++++++++', 10, 13

db '+ +', 10, 13

db '+ +', 10, 13

db '+ +', 10, 13

db '+ +', 10, 13

db '++++++++++++++', 10, 13

db ' ', 10, 13

db 'press p to play', 10, 13

db 'press Esc', 10, 13,'$'

פעולות:

**ציור שחקן**

**CODESEG**

; draw player in cursor position

proc DrawPlayer

mov ah, 9 ;Write character and attribute at cursor position

mov al, 2 ;AL = character to display

mov bh, 0h ;BH=Page

mov bl, [colour] ; BL = Foreground - (חזית)

mov cx, 1 ;CX = number of times to write character

int 10h ;int 10h Bois -> show the character

ret

endp DrawPlayer

**מיקום סמן**

proc Cursor\_Location

;Place the cursor on the screen

mov bh, 0

mov dh, [y\_row] ; position in raw

mov dl, [x\_column] ; position in column

mov ah, 2

int 10h

ret

endp Cursor\_Location

**תזוזת שחקן ימינה**

proc RightKeyPress

inc [x\_column]

call Cursor\_Location ; move cursor to new place

;check the character on coordinates

mov bh, 0h ; Page=1

mov ah, 08h ; Read character function

int 10h ;return the character value to AL

cmp al, '+'

jne MovePlayerRigt ; draw player

הזזת סמן ימינה, בדיקה של התו במיקום הסמן

אם **לא** גבול מבוך

מחזירים סמן למיקום המקורי

מציירים דמות שחורה

מזיזים זמן

מציירים דמות בצבע.

אם גבול מבוך

מחזירים סמן למיקום מקורי

יציאה מהפעולה

dec [x\_column]

call Cursor\_Location

jmp EndRightKeyPress

MovePlayerRigt:

dec [x\_column]

call Cursor\_Location

mov [colour], 0

call DrawPlayer

add [x\_column], 1

mov [colour], 11

call Cursor\_Location

call DrawPlayer

EndRightKeyPress:

ret

endp RightKeyPress

**תזוזת שחקן שמאלה**

proc LeftKeyPress

dec [x\_column]

call Cursor\_Location ; move cursor to new place

;check the character on coordinates

mov bh, 0h ; Page=1

mov ah, 08h ; Read character function

int 10h ;return the character value to AL

cmp al, '+'

jne MovePlayerLeft ; draw player

inc [x\_column]

call Cursor\_Location

jmp EndLeftKeyPress

MovePlayerLeft:

inc [x\_column]

call Cursor\_Location

mov [colour], 0

call DrawPlayer

sub [x\_column], 1

mov [colour], 11

call Cursor\_Location

call DrawPlayer

EndLeftKeyPress:

ret

endp LeftKeyPress

**תזוזת שחקן למעלה**

proc UpKeyPress

dec [y\_row]

call Cursor\_Location ; move cursor to new place

; Check the character on coordinates

mov bh, 0h ; Page=1

mov ah, 08h ; Read character function

int 10h ;Return the character value to AL

cmp al, '+'

jne MovePlayerUp ; draw player

inc [y\_row]

call Cursor\_Location

jmp EndUpKeyPress

MovePlayerUp:

inc [y\_row]

call Cursor\_Location

mov [colour], 0

call DrawPlayer

sub [y\_row], 1

mov [colour], 11

call Cursor\_Location

call DrawPlayer

EndUPKeyPress:

ret

endp UpKeyPress

**תזוזת שחקן למטה**

proc DownKeyPress

inc [y\_row]

call Cursor\_Location ; Move cursor to new place

; Check the character on coordinates

mov bh, 0h ; Page=1

mov ah, 08h ; Read character function

int 10h ; Return the character value to AL

cmp al, '+'

jne MovePlayerDown ; Draw player

dec [y\_row]

call Cursor\_Location

jmp EndDownKeyPress

MovePlayerDown:

dec [y\_row]

call Cursor\_Location

mov [colour], 0

call DrawPlayer

add [y\_row], 1

mov [colour], 11

call Cursor\_Location

call DrawPlayer

EndDownKeyPress:

ret

endp DownKeyPress

תכנית ראשית:

**start:**

**Move:**

; Check Type Ahead Buffer Status

mov ah, 1h

int 16h

jz Move

; Wait for char

mov ah, 0h

int 16h

; Was Esc pressed?

cmp al, 27

je theEnd

; Was right Key Pressed?

cmp ah, 4dh

je RightKey

; Was left Key Pressed?

cmp ah, 4bh

je LeftKey

; Was up Key Pressed?

cmp ah, 48h

je UpKey

; Was Down Key Pressed?

cmp ah, 50h

je DownKey

jmp Move

\*\*\*

mov ax, @data

mov ds, ax

; Enter text mode 40x25 (x,y)

mov al, 013h

mov ah, 0

int 10h

; Print string to screen

mov dx, offset memu1

mov ah, 9h

int 21h

call Cursor\_Location

call DrawPlayer

; Wait for char

WaitForCharacter:

mov ah, 0h

int 16h

; Was Esc pressed ?

cmp al, 27

je theEnd

; Was p pressed?

cmp al, 'p'

je Move

jmp WaitForCharacter

**Move:**

**\*\*\***

RightKey:

call RightKeyPress

jmp Move

LeftKey:

call LeftKeyPress

jmp Move

UpKey:

call UpKeyPress

jmp Move

DownKey:

call DownKeyPress

jmp Move

theEnd:

; Return text mode 80x25 (x,y)

mov al, 03h

mov ah, 0

int 10h

; --------------------------

exit:

mov ax, 4c00h

int 21h

END start