ציור בפיקסלים

# לולאות ופעולות

קבצים p1, p2

בדף עבודה זה נתנסה בעבודה במצב גראפי. (בשונה ממצב טקסט)

נצייר ביחד, קווים אנכיים וקווים אופקיים.

# מעבר למצב גרפי

המעבד יכול לעבוד בשני מצבים הראשון מצב טקסט (בו עבדנו עד היום) והשני מצב גרפי בו משתמשים בגרפיקה (ציור ושרטוט על המסך) כדי לעבור למוד גראפי נשתמש בפסיקת ה bios 10h

; Enter Graphic mod 320 \* 200 pixels, 256 colors

mov ax, 13h

int 10h

צרו תכנית העוברת למוד גראפי. בדקו בעזרת ה turbo debugger (Alt -F5) מה קורה למסך ה – DosBox במוד זה?

**חזרה ממצב גרפי**:

הוסיפו בסוף התוכנית שלכם את ה interrupt החוזר למצב text.

הריצו את התכנית ב – Turbo Debugger, כדי לצפות במסך המשתמש הקישו **Alt + F5**.

חזרה למסך ה – TD הקישו **Esc**.

; return to text mode 80\*20

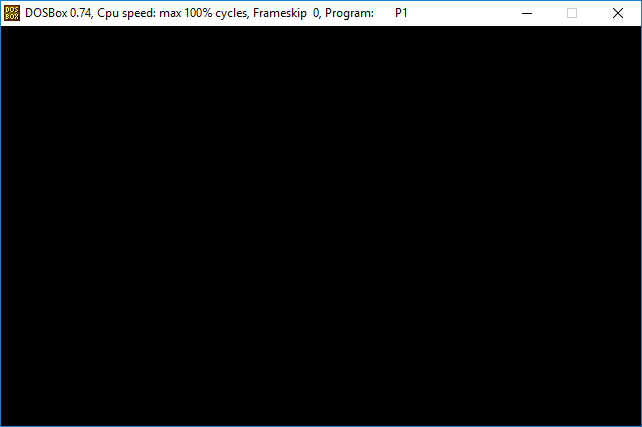
mov ax, 2

int 10h

# ציור במצב גרפי – ציור פיקסל (נקודה על המסך)

המסך מורכב מנקודות. כל נקודה נקראת פיקסל. במצב גרפי המסך מורכב מ 200 שורות ו 320 עמודות של פיקסלים. לכל פיקסל מיקום וצבע.

המיקום מורכב משורה ועמודה ⇦ row, column



נוסיף פקודה לתכנית שבהקשת מקש (כל שהוא) נצא מהמצב הגראפי ונחזור למוד של טקטס.

; Wait for key press

mov ah,0h

int 16h

319,199

0,199

319,0

0,0

start:

mov ax, @data

mov ds, ax

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

; Enter Graphic mod 320 \* 200 pixels, 256 colors

mov ax, 13h

int 10h

; Wait for key press

mov ah, 0h ; get keystroke from keyboard

int 16h

; Return to text mod 80\*20

mov ax, 2

int 10h

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

exit:

mov ax, 4c00h

int 21h

END start

# ציור במצב גרפי – ציור פיקסל (נקודה על המסך)

**הצבע:**

בזיכרון המחשב יש טבלה בגודל 256 תאים ובכל תא שלה מצוי מספר המייצג צבע.

כדי לצייר פיקסל למסך, צריך לציין שלושה נתונים: מיקום על ציר x, מיקום על ציר y וצבע.

לדוגמה: שורה 0

עמודה 0

צבע 4

יצבע בצבע אדום את הפיקסל בפינה השמאלית העליונה

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **319,0** |  |  |  |  |  |  | **0, 0** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **319, 199** |  |  |  |  |  |  | **0,199** |

האינטרפט (פסיקה) המצייר פיקסל למסך, הוא פסיקת bois - **int 10h**

**הפרמטרים עבור אינטרפט זה:**

ah = 0ch – קוד הפסיקה

bh = 0 – איפוס רגיסטר bh

cx – עמודה

dx – שורה

ax - צבע

; Draw Pixel

xor bh, bh

mov cx, [x\_coordinate]

mov dx, [y\_coordinate]

mov ax, [color]

mov ah, 0ch

int 10h

כתבו תכנית המציירת פיקסל אדום )תא 4 בפלטת הצבעים) בשורה 20, עמודה 50 ובצבע 11.

שימו לב 🎔: אל תשכחו להצהיר על המשתנים x\_coordinate ו - y\_coordinate במקטע הנתונים ולתת להם ערכים.

בהרצת התכנית ב – TD ניתן לראות את הציור על מסך (לאחר הפעלת הפסיקה) ע"י הקשה על Alt + F5 או בחירה בתפריט Window ו - User screen.

DATASEG

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

; Your variables here

x\_coordinate dw 50 ; in column

y\_coordinate dw 20 ; in line

color dw 11

CODESEG

start:

mov ax, @data

mov ds, ax

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

; Enter Graphic mod

mov ax, 13h

int 10h

; draw pixel

xor bh, bh

mov cx, [x\_coordinate] ;in column

mov dx, [y\_coordinate] ; in line

mov ax, [color]

mov ah, 0ch

int 10h

; Enter text mod

mov ax, 2

int 10h

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

# פרוצדורה

שנו את קטע הקוד המצייר פיסל על המסך לפרוצדורה.

כדי לשמור על מצב הרגיסטרים בכניסה לכל פרוצדורה יש צורך לשמור את מצב הרגיסטרים בתחילתה ע"י פקודת push למחסנית, ולמשוך את ממצב הרגיסטרים בסופה pop מהמחסנית.

ההוראה p186 מאפשרת לעשות פוקדת push ו – pop לכל הרגיסטרים בהוראה אחת.

IDEAL

MODEL small

STACK 100h

**p186**

פקודת pusha - דוחפת את כל הרגיסטרים למחסנית

פקודת poa – מושכת את כל הרגיסטרים מהמחסנית

**pusha**

**popa**

proc drawPixel

; draw pixel

xor bh, bh

mov cx, [x\_coordinate] ; place in column

mov dx, [y\_coordinate] ; place in line

mov ax, [color]

mov ah, 0ch

int 10h

ret

endp drawPixel

**הריצו ובדקו.**

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

# ציור קו אופקי

קו הוא אוסף של נקודות ............... על ציר ה – x

כתבו פרוצדורה המציירת קו על המסך (לרוחב).

* הפרוצדורה עושה שימוש בקואורדינטת של תחילת השורה על ציר ה – x
* בכל איטרציה של הלולאה לאחר ציור הפיקסל למסך נגדיל את המיקום שלו על ציר ה – x.
* לצורך ביצוע השרטוט הקו עלינו ליצור משתנה חדש שבו נשמור את ארוך הקו, [len]

**ניצור פרוצדורה חדשה המציירת קו למסך - drawHorizontalLine**

* בתחילת הפרוצדורה נבצע פקודת push לכל הרגיסטרים כדי לשמור את ערכם למחסנית.

בפרוצדורה זו יש שימוש בלולאה ולכן חשוב מאוד לשמר את הרגיסטרים בכלל ואת רגיסטר cx בפרט.

* נוסיף משתנה בגודל מילה לאורך הקו בפיקסלים: len ונקבע לו את הערך 20 ⇦ len dw 20
* ניצור לולאה שערכה הוא כאורך הקו בפיקסלים (נעביר לרגיסטר cx את הערך של משתנה len).
* נזמן את הפעולה של ציור הפיקסל מתוך פרוצדורה זו.
* נגדיל את ערכו של [x\_coordinate]
* בסיום הפרוצדורה נבצע פעולת **popa** לערכים של כל הרגיסטרים.

זמנו את הפעולה, הריצו ובדקו את התוצאה.

proc drawHorizontalLine

; Draw line of pixcels

pusha

mov cx,[len]

HorizontalLine:

call draePixel

inc [x\_coordinate]

loop HorizontalLine

popa

ret

endp drawHorizontalLine

# ציור קו אנכי

כתבו פעולה המציירת קו אנכי.

קו אנכי הוא אוסף נקודות על ציר y

כתבו פרוצדורה חדשה drawVerticalLine

שימו לב 🎔: אל תשכחו לפני הקריאה לפרוצדורה להשים ערכים למשתנים של תחילת ציור הקו על ציר ה – x ועל ציר ה – y.

mov [x\_coordinate], 30

mov [y\_coordinate], 30

call drawVerticalLine

קישור לטבלת צבעים

<https://upload.wikimedia.org/wikipedia/commons/4/49/VGA_palette.svg>

**הריצו ובדקו.**

נסו לשרטט מרובע על המסך.

חשבו על מיקום, וצבע.

IDEAL

MODEL small

STACK 100h

**p186**

DATASEG

x\_coordinate dw 50 ; place in line

y\_coordinate dw 20 ; place in column

len dw 20

color dw 5

CODESEG

proc drawHorizontalLine

; draw line of pixels

**pusha**

mov cx, [len]

drawLine:

call drawPixel

inc [x\_ coordinate]

loop drawLine

**popa**

ret

endp drawHorizontalLine

start:

mov ax, @data

mov ds, ax

; Enter Graphic mod

mov ax, 13h

int 10h

call drawHorizontalLine

call drawVerticalLine

proc drawVerticalLine

; draw column of pixels

pusha

mov cx, [len]

drawColum:

call drawPixel

inc [y\_ coordinate]

loop drawColum

popa

ret

endp drawVerticalLine

; Wait for key press

mov ah,00h

int 16h

;Retern text mod

mov ax, 2

int 10h

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

exit:

mov ax, 4c00h

int 21h

END start

שרטוט ריבוע:

DATASEG

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

x\_coordinate dw 100

y\_coordinate dw 50

color dw 11

len dw 50

call drawHorizontalLine

call drawVerticalLine

mov [x\_coordinate], 100

mov [y\_coordinate], 50

call drawVerticalLine

call drawHorizontalLine

