**ספר הפרויקט**

מגיש: יואב קולט פרויקט: מבוך

חלק א: הפרויקט הוא מבוך בעל 3 שלבים הדרכה , פתיחה , וסיום.

בפרויקט ממומש פתיחת 7 תמונות, קליטת תווים מן המקלדת,

שימוש בעכבר,השמעת צלילים בין כל שלב, יצירת פיקסלים על המסך,

קריאת פיקסלים מהמסך והשוואת צבע הפיקסלים.

חלק ב רפלקציה:

1. מדוע בחרתם בפרויקט זה?

בחרתי בפרויקט זה מכיוון שרציתי פרויקט שמשתמש בתמונות והזזה של הפיקסלים, בנוסף רציתי פרויקט יחסית "קל" כדי שאני אוכל להשקיע בו יותר.

2. מהם האתגרים שנתקלתם ביישום הפרויקט?

האתגרים בהם נתקלתי הם:

1. השוואת צבע הפיקסלים על המסך כדי שהשחקן לא יעלה על הגבולות ושהוא יוכל לעבור שלב.
2. הגדלה והקטנה של השחקן.
3. שימוש בעכבר.

3. האם מימשתם את כל מה שתכננתם?

כן, מימשתי את כל מה שתכננתי לעשות בפרויקט ואפילו טיפה מעבר.

4. אם יכולתם להמשיך, מה הייתם מוסיפים?

אם הייתי יכול להמשיך הייתי מוסיף עוד שלבים.

חלק ג:

הפרוצדורות המעורבות הם:

קריאת תמונות:

;================ ===============stat files

טענת כניסה: מקבלת קובץ קובץ BMP

טענת יציאה: קוראת את הקובץ

proc ReadHeader

; Read BMP file header, 54 bytes

mov ah,3fh

mov bx, [filehandle]

mov cx,54

mov dx,offset Header

int 21h

ret

endp ReadHeader

טענת כניסה: מקבלת את הפלטה של הצבעים

טענת יציאה: קוראת את הפלטה של הצבעים

proc ReadPalette

; Read BMP file color palette, 256 colors \* 4 bytes (400h)

mov ah,3fh

mov cx,400h

mov dx,offset Palette

int 21h

ret

endp ReadPalette

טענת כניסה: מקבל את הפלטה של הצבעים

טענת יציאה: מעתיק את הפלטה של הצבעים

proc CopyPal

; Copy the colors palette to the video memory

; The number of the first color should be sent to port 3C8h

; The palette is sent to port 3C9h

mov si,offset Palette

mov cx,256

mov dx,3C8h

mov al,0

; Copy starting color to port 3C8h

out dx,al

; Copy palette itself to port 3C9h

inc dx

PalLoop :

; Note: Colors in a BMP file are saved as BGR values rather than RGB .

mov al,[si+2] ; Get red value .

shr al,2 ; Max. is 255, but video palette maximal

; value is 63. Therefore dividing by 4.

out dx,al ; Send it .

mov al,[si+1] ; Get green value .

shr al,2

out dx,al ; Send it .

mov al,[si] ; Get blue value .

shr al,2

out dx,al ; Send it .

add si,4 ; Point to next color .

; (There is a null chr. after every color.)

loop PalLoop

ret

endp CopyPal

טענת כניסה: מקבל קובץ BMP

טענת יציאה: קורא את התמונה ומדפיס אותה

proc CopyBitmap

; BMP graphics are saved upside-down .

; Read the graphic line by line (200 lines in VGA format),

; displaying the lines from bottom to top.

mov ax, 0A000h

mov es, ax

mov cx,200

PrintBMPLoop :

push cx

; di = cx\*320, point to the correct screen line

mov di,cx

shl cx,6

shl di,8

add di,cx

; Read one line

mov ah,3fh

mov cx,320

mov dx,offset ScrLine

int 21h

; Copy one line into video memory

cld ; Clear direction flag, for movsb

mov cx,320

mov si,offset ScrLine

rep movsb ; Copy line to the screen

;rep movsb is same as the following code :

;mov es:di, ds:si

;inc si

;inc di

;dec cx

;loop until cx=0

pop cx

loop PrintBMPLoop

ret

endp CopyBitmap

;======================= =======end files

טענת כניסה: מקבלת את הקובץ של פתיחת המשחק

טענת יציאה: פותחת את הקובץ של פתיחת המשחק

;======================stat of game opening

proc OpenFile\_game\_opening

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_game\_opening

int 21h

jc openerror\_game\_opening

mov [filehandle], ax

ret

openerror\_game\_opening:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_game\_opening

;============== =========end of game opening

טענת כניסה: מקבלת את הקובץ הוראות

טענת יציאה: פותחת את הקובץ הוראות

;===== =============start of instructions

proc OpenFile\_instructions

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_instructions

int 21h

jc openerrorinstructions

mov [filehandle], ax

ret

openerrorinstructions:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_instructions

;======== ==================end of instructions

טענת כניסה: מקבלת את הקובץ הדרכה

טענת יציאה: פותחת את הקובץ הדרכה

;============= ============stat of tutorial

proc OpenFile\_tutorial

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_tutorial

int 21h

jc openerror\_tutorial

mov [filehandle], ax

ret

openerror\_tutorial:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_tutorial

;============= ===============end of tutorial

טענת כניסה: מקבלת את הקובץ שלב 1

טענת יציאה: פותחת את הקובץ שלב 1

;========== =============start of background level 1

proc OpenFile\_level1

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_level1

int 21h

jc openerror1

mov [filehandle], ax

ret

openerror1:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_level1

;==================== =====end of background level 1

טענת כניסה: מקבלת את הקובץ שלב 2

טענת יציאה: פותחת את הקובץ שלב 2

;======== ====== =========start of background level 2

proc OpenFile\_level2

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_level2

int 21h

jc openerror2

mov [filehandle], ax

ret

openerror2:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_level2

;============ ===========end of background level 2

טענת כניסה: מקבלת את הקובץ שלב 3

טענת יציאה: פותחת את הקובץ שלב 3

;========== = ==========start of background level 3

proc OpenFile\_level3

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_level3

int 21h

jc openerror3

mov [filehandle], ax

ret

openerror3:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_level3

;====== ===========end of background level 3

טענת כניסה: מקבלת את הקובץ של סוף המשחק

טענת יציאה: פותחת את הקובץ של סוף המשחק

;========== ==============start EndGame

proc OpenFile\_EndGame

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_EndGame

int 21h

jc openerrorEndGame

mov [filehandle], ax

ret

openerrorEndGame:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_EndGame

;============= ==========end of EndGame

ציור השחקן:

טענת כניסה: מקבל את ה X והY של השחקן ואת הגודל שלו

טענת יציאה: מדפיס את השחקן לפי הגודל

proc player;מצייר את השחקן

mov [y\_count], si

draw1:

mov [x\_count], si

draw2:

call drawpixel

inc [player\_x]

cmp [x\_count],0

dec [x\_count]

jne draw2

inc [player\_y]

sub [player\_x],si

cmp [y\_count],0

dec [y\_count]

jne draw1

sub [player\_y],si

ret

endp player

ציור פיקסל:

טענת כניסה: מקבל את ה X והY של השחרן ואת הצבע שלו

טענת יציאה: מדפיס את הפיקסל

proc drawpixel; מצייר פיקסל על המסך

mov bh,0h

mov cx,[player\_x]

mov dx,[player\_y]

mov al,[color]

mov ah,0ch

int 10h

ret

endp drawpixel

ניקוי פיקסל:

טענת כניסה: מקבל את הX והY של השחקן

טענת יציאה: מדפיס פיקסל עם הצבע של הרקע

proc clearpixel; מנקה פיקסל על המסך

mov bh,0h

mov cx,[player\_x]

mov dx,[player\_y]

mov al,[level\_color]

mov ah,0ch

int 10h

ret

endp clearpixel

ניקוי שחקן:

טענת כניסה: מקבבל את הX והY ואת הצבע של הרקע

טענת יציאה: מדפיס את הצבע של הרגע אחרי השחקן

proc clear\_player; מנקה את השחקן

mov [y\_count], si

clear1:

mov [x\_count], si

clear2:

call clearpixel

inc [player\_x]

cmp [x\_count],0

dec [x\_count]

jne clear2

inc [player\_y]

sub [player\_x],si

cmp [y\_count],0

dec [y\_count]

jne clear1

sub [player\_y],si

ret

endp clear\_player

קריאת צבע הפיקסלים מימין , שמאל, למטה ולמעלה של השחקן:

;===============================read player

טענת כניסה: מקבל את הX והY שהוא צריך לקרוא

טענת יציאה:קורא את הצבע ושומר אותו בAL

proc read\_player\_up; קורא את הצבע

mov cx,[player\_x]

mov dx,[player\_y]

dec dx

mov [read\_count],si

readUP:

mov bh,0h

mov ah,0dh

int 10h ; return al the pixel value read

call comper

inc cx

cmp [read\_count], 0

dec [read\_count]

jne readUP

ret

endp read\_player\_up

טענת כניסה: מקבל את הX והY שהוא צריך לקרוא

טענת יציאה:קורא את הצבע ושומר אותו בAL

proc read\_player\_right; קורא את הצבע

mov cx,[player\_x]

mov dx,[player\_y]

add cx,si

mov [read\_count],si

readRight:

mov bh,0h

mov ah,0dh

int 10h ; return al the pixel value read

call comper

inc dx

cmp [read\_count], 0

dec [read\_count]

jne readRight

ret

endp read\_player\_right

טענת כניסה: מקבל את הX והY שהוא צריך לקרוא

טענת יציאה:קורא את הצבע ושומר אותו בAL

proc read\_player\_down; קורא את הצבע

mov cx,[player\_x]

mov dx,[player\_y]

add dx, si

mov [read\_count],si

readDown:

mov bh,0h

mov ah,0dh

int 10h ; return al the pixel value read

call comper

inc cx

cmp [read\_count], 0

dec [read\_count]

jne readDown

ret

endp read\_player\_down

טענת כניסה: מקבל את הX והY שהוא צריך לקרוא

טענת יציאה:קורא את הצבע ושומר אותו בAL

proc read\_player\_left; קורא את הצבע

mov cx,[player\_x]

mov dx,[player\_y]

dec cx

mov [read\_count],si

readLeft:

mov bh,0h

mov ah,0dh

int 10h ; return al the pixel value read

call comper

inc dx

cmp [read\_count], 0

dec [read\_count]

jne readLeft

ret

endp read\_player\_left

;================================end of player

תחילת השמע:

טענת כניסה: מקבל תדר לצליל

טענת יציאה: פותח את הרמקולים ומשמיע את הצליל

proc startsound

; open speaker

in al, 61h

or al, 00000011b

out 61h, al

; send control to change frequency

mov al, 0B6h

out 43h, al

; play sound

mov ax, [note]

out 42h, al ; Sending lower byte

mov al, ah

out 42h, al ; Sending upper byte

ret

endp startsound

סוף השמע:

טענת כניסה: סוגר את הרמקולים

טענת יציאה: סוגר את הרמקולים

proc endsound

; close the speaker

in al, 61h

and al, 11111100b

out 61h, al

ret

endp endsound

קריאת כל השמע:

טענת כניסה: קורא לפרוצדורות של השמע

טענת יציאה: מחזיר צליל קצר

proc sound

call startsound

call timer

call endsound

ret

endp sound

טיימר בשביל השמע:

טענת כניסה: מקבל זמן מוגדר

טענת יציאה: משמיע את השמע לבערך חצי שנייה

proc timer

; wait for first change in timer

mov ax, 40h

mov es, ax

mov ax, [Clock]

FirstTick:

cmp ax, [Clock]

je FirstTick

mov cx, 3

DelayLoop:

mov ax, [Clock]

Tick :

cmp ax, [Clock]

je Tick

loop DelayLoop

ret

endp timer

עכבר:

טענת כניסה: מדפיס את העכבר על המסך

טענת יציאה: מחכה ללחיצה ומדפיס את העכבר על המסך

proc mouse

mov ax, 1h

int 33h

; Loop until mouse click

Mouse1:

mov ax, 3h

int 33h

cmp bx, 01h ; check left mouse click

jne Mouse1

shr cx, 1; adjust cx to range 0-319, to fit scree

dec dx

ret

endp mouse

השוואת צבעים:

טענת כניסה: מקבל ערך של צבע בAL

טענת יציאה: קופץ למקום הנכון לפי הערך של הצבע

proc comper

cmp al,0h;black

je Game2

cmp al,7h;gray

jne level1;eny other color

ret

endp comper

הקוד הראשי:

הקוד פותח תמונות ועובר לתזוזה של המשחק.

בתזוזה של המשחק קולטים תו מהמקלדת ואז משווים את התו לכיוון שהשחקן צריך ללכת.

חלק ד:

השתמשתי ב3 אינטרפטים שונים:

Int 21

Int 10

Int 33

השתמשתי בפרוצדורות:

Int 21h

ReadHeader- כדי לקרוא קובץ

ReadPalette – כדי לקרוא קובץ

CopyBitmap – לקרוא קובץ

כל ה-OpenFile- פתיחת קובץ, לרשום string

Int 10h

Drawpixel- רושם פיקסל על המסך

Clearpixel- רושם פיקסל על המסך

כל ה read player- קורא פיקסל המסך

Int 33h

Mouse- מראה את העכבר על המסך, מקבל את מצב העכבר ואת סטטוס הלחצן

**הקוד:**

[להורדת הקובץ:](https://docs.google.com/document/d/1m8wWr0YEEJy5szd_irK0iH7g_McgXlSgI1NsOq-YC4o/edit)

IDEAL

MODEL small

STACK 100h

DATASEG

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

;files

filehandle dw ?

Header db 54 dup (0)

Palette db 256\*4 dup (0)

ScrLine db 320 dup (0)

;game opening

filename\_game\_opening db 'game.bmp',0 ;game opening image

;instructions

filename\_instructions db 'I.bmp',0 ;game opening image

;tutorial opening

filename\_tutorial db 'tutorial.bmp',0 ;game opening image

;background level 1

filename\_level1 db 'level\_1.bmp',0 ;1st background image

;background level 2

filename\_level2 db 'level\_2.bmp',0 ;2nd background image

;background level 3

filename\_level3 db 'level\_3.bmp',0 ;3rd background image

;background EndGame

filename\_EndGame db 'EndGame.bmp',0 ;3rd background image

;-----------------------------------messeges

;Error messege

ErrorMsg db 'Error', 13, 10 ,'$'

;mouse coordes

corner1\_Instructions dw 184, 46

corner2\_Instructions dw 310, 64

corner1\_InstructionsExit dw 2, 4

corner2\_InstructionsExit dw 13, 12

corner1\_tutorial dw 116, 85

corner2\_tutorial dw 198, 103

corner1\_startGame dw 16, 47

corner2\_startGame dw 139, 67

corner1\_yes\_EndGame dw 47, 70

corner2\_yes\_EndGame dw 94, 86

corner1\_no\_EndGame dw 239, 74

corner2\_no\_EndGame dw 262, 86

note dw 6ADh

Clock equ es:6Ch

;player settings

player\_x dw 0

player\_y dw 0

player\_tutorial\_x dw 35

player\_tutorial\_y dw 107

player\_x1 dw 19

player\_y1 dw 175

player\_x2 dw 16

player\_y2 dw 184

player\_x3 dw 11

player\_y3 dw 189

x\_count dw ?

y\_count dw ?

levelcount db 0

read\_count dw,5

level\_color db 7h

color db 0h

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

CODESEG

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

;=========================================stat files

proc ReadHeader

; Read BMP file header, 54 bytes

mov ah,3fh

mov bx, [filehandle]

mov cx,54

mov dx,offset Header

int 21h

ret

endp ReadHeader

proc ReadPalette

; Read BMP file color palette, 256 colors \* 4 bytes (400h)

mov ah,3fh

mov cx,400h

mov dx,offset Palette

int 21h

ret

endp ReadPalette

proc CopyPal

; Copy the colors palette to the video memory

; The number of the first color should be sent to port 3C8h

; The palette is sent to port 3C9h

mov si,offset Palette

mov cx,256

mov dx,3C8h

mov al,0

; Copy starting color to port 3C8h

out dx,al

; Copy palette itself to port 3C9h

inc dx

PalLoop :

; Note: Colors in a BMP file are saved as BGR values rather than RGB .

mov al,[si+2] ; Get red value .

shr al,2 ; Max. is 255, but video palette maximal

; value is 63. Therefore dividing by 4.

out dx,al ; Send it .

mov al,[si+1] ; Get green value .

shr al,2

out dx,al ; Send it .

mov al,[si] ; Get blue value .

shr al,2

out dx,al ; Send it .

add si,4 ; Point to next color .

; (There is a null chr. after every color.)

loop PalLoop

ret

endp CopyPal

proc CopyBitmap

; BMP graphics are saved upside-down .

; Read the graphic line by line (200 lines in VGA format),

; displaying the lines from bottom to top.

mov ax, 0A000h

mov es, ax

mov cx,200

PrintBMPLoop :

push cx

; di = cx\*320, point to the correct screen line

mov di,cx

shl cx,6

shl di,8

add di,cx

; Read one line

mov ah,3fh

mov cx,320

mov dx,offset ScrLine

int 21h

; Copy one line into video memory

cld ; Clear direction flag, for movsb

mov cx,320

mov si,offset ScrLine

rep movsb ; Copy line to the screen

;rep movsb is same as the following code :

;mov es:di, ds:si

;inc si

;inc di

;dec cx

;loop until cx=0

pop cx

loop PrintBMPLoop

ret

endp CopyBitmap

;=========================================end files

;=========================================stat of game opening

proc OpenFile\_game\_opening

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_game\_opening

int 21h

jc openerror\_game\_opening

mov [filehandle], ax

ret

openerror\_game\_opening:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_game\_opening

;=========================================end of game opening

;=========================================start of instructions

proc OpenFile\_instructions

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_instructions

int 21h

jc openerrorinstructions

mov [filehandle], ax

ret

openerrorinstructions:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_instructions

;=========================================end of instructions

;=========================================stat of tutorial

proc OpenFile\_tutorial

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_tutorial

int 21h

jc openerror\_tutorial

mov [filehandle], ax

ret

openerror\_tutorial:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_tutorial

;=========================================end of tutorial

;=========================================start of background level 1

proc OpenFile\_level1

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_level1

int 21h

jc openerror1

mov [filehandle], ax

ret

openerror1:

mov dx, offset ErrorMsg

mov ah, 9h

ret

endp OpenFile\_level1

;=========================================end of background level 1

;=========================================start of background level 2

proc OpenFile\_level2

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_level2

int 21h

jc openerror2

mov [filehandle], ax

ret

openerror2:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_level2

;=========================================end of background level 2

;=========================================start of background level 3

proc OpenFile\_level3

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_level3

int 21h

jc openerror3

mov [filehandle], ax

ret

openerror3:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_level3

;=========================================end of background level 3

;=========================================start EndGame

proc OpenFile\_EndGame

; Open file

mov ah, 3Dh

xor al, al

mov dx, offset filename\_EndGame

int 21h

jc openerrorEndGame

mov [filehandle], ax

ret

openerrorEndGame:

mov dx, offset ErrorMsg

mov ah, 9h

int 21h

ret

endp OpenFile\_EndGame

;=========================================end of EndGame

;=========================================start of player

proc drawpixel; מצייר פיקסל על המסך

mov bh,0h

mov cx,[player\_x]

mov dx,[player\_y]

mov al,[color]

mov ah,0ch

int 10h

ret

endp drawpixel

proc player;מצייר את השחקן

mov [y\_count], si

draw1:

mov [x\_count], si

draw2:

call drawpixel

inc [player\_x]

cmp [x\_count],0

dec [x\_count]

jne draw2

inc [player\_y]

sub [player\_x],si

cmp [y\_count],0

dec [y\_count]

jne draw1

sub [player\_y],si

ret

endp player

proc clearpixel; מנקה פיקל על המסך

mov bh,0h

mov cx,[player\_x]

mov dx,[player\_y]

mov al,[level\_color]

mov ah,0ch

int 10h

ret

endp clearpixel

proc clear\_player; מנקה את השחקן

mov [y\_count], si

clear1:

mov [x\_count], si

clear2:

call clearpixel

inc [player\_x]

cmp [x\_count],0

dec [x\_count]

jne clear2

inc [player\_y]

sub [player\_x],si

cmp [y\_count],0

dec [y\_count]

jne clear1

sub [player\_y],si

ret

endp clear\_player

;==============================================read player

proc read\_player\_up; קורא את הצבע

mov cx,[player\_x]

mov dx,[player\_y]

dec dx

mov [read\_count],si

readUP:

mov bh,0h

mov ah,0dh

int 10h ; return al the pixel value read

call comper

inc cx

cmp [read\_count], 0

dec [read\_count]

jne readUP

ret

endp read\_player\_up

proc read\_player\_right; קורא את הצבע

mov cx,[player\_x]

mov dx,[player\_y]

add cx,si

mov [read\_count],si

readRight:

mov bh,0h

mov ah,0dh

int 10h ; return al the pixel value read

call comper

inc dx

cmp [read\_count], 0

dec [read\_count]

jne readRight

ret

endp read\_player\_right

proc read\_player\_down; קורא את הצבע

mov cx,[player\_x]

mov dx,[player\_y]

add dx, si

mov [read\_count],si

readDown:

mov bh,0h

mov ah,0dh

int 10h ; return al the pixel value read

call comper

inc cx

cmp [read\_count], 0

dec [read\_count]

jne readDown

ret

endp read\_player\_down

proc read\_player\_left; קורא את הצבע

mov cx,[player\_x]

mov dx,[player\_y]

dec cx

mov [read\_count],si

readLeft:

mov bh,0h

mov ah,0dh

int 10h ; return al the pixel value read

call comper

inc dx

cmp [read\_count], 0

dec [read\_count]

jne readLeft

ret

endp read\_player\_left

;=========================================end of player

;=========================================sound

proc startsound

; open speaker

in al, 61h

or al, 00000011b

out 61h, al

; send control to change frequency

mov al, 0B6h

out 43h, al

; play sound

mov ax, [note]

out 42h, al ; Sending lower byte

mov al, ah

out 42h, al ; Sending upper byte

ret

endp startsound

proc endsound

; close the speaker

in al, 61h

and al, 11111100b

out 61h, al

ret

endp endsound

;=========================================sound

;=========================================other

proc timer

; wait for first change in timer

mov ax, 40h

mov es, ax

mov ax, [Clock]

FirstTick:

cmp ax, [Clock]

je FirstTick

mov cx, 3

DelayLoop:

mov ax, [Clock]

Tick :

cmp ax, [Clock]

je Tick

loop DelayLoop

ret

endp timer

proc sound

call startsound

call timer

call endsound

ret

endp sound

proc mouse

; Show mouse

mov ax, 1h

int 33h

; Loop until mouse click

Mouse1:

mov ax, 3h

int 33h

cmp bx, 01h ; check left mouse click

jne Mouse1

shr cx, 1; adjust cx to range 0-319, to fit scree

dec dx

ret

endp mouse

proc comper

cmp al,0h;black

je Game2

cmp al,7h;gray

jne level1;eny other color

ret

endp comper

;=========================================other

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

start:

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

mov ax, @data

mov ds, ax

;=========================================backgrounds

;Game opening

game\_opening:

; Graphic mode

mov ax,13h

int 10h

call OpenFile\_game\_opening

call ReadHeader

call ReadPalette

call CopyPal

call CopyBitmap

label\_mouse:

call mouse

comper\_tutorial:

cmp cx, [ corner1\_tutorial]

jle comper\_Instructions

cmp dx, [corner1\_tutorial+2]

jle comper\_Instructions

cmp cx, [corner2\_tutorial]

jge comper\_Instructions

cmp dx, [corner2\_tutorial+2]

jge comper\_Instructions

jmp tutorial

comper\_Instructions:

cmp cx, [corner1\_Instructions]

jle comper\_startGame

cmp dx, [corner1\_Instructions+2]

jle comper\_startGame

cmp cx, [corner2\_Instructions]

jge comper\_startGame

cmp dx, [corner2\_Instructions+2]

jge comper\_startGame

jmp Instructions

comper\_startGame:

cmp cx, [corner1\_startGame]

jle label\_mouse

cmp dx, [corner1\_startGame+2]

jle label\_mouse

cmp cx, [corner2\_startGame]

jge label\_mouse

cmp dx, [corner2\_startGame+2]

jge label\_mouse

mov [levelcount],1

jmp level\_1

level1:

jmp level

Game2:

jmp Game1

Game\_open:

jmp game\_opening

;Instructions

Instructions:

; Graphic mode

mov ax,13h

int 10h

call OpenFile\_instructions

call ReadHeader

call ReadPalette

call CopyPal

call CopyBitmap

label\_mouse\_instructions:

call mouse

comper\_instructions\_page:

cmp cx, [corner1\_InstructionsExit]

jle label\_mouse\_instructions

cmp dx, [corner1\_InstructionsExit+2]

jle label\_mouse\_instructions

cmp cx, [corner2\_InstructionsExit]

jge label\_mouse\_instructions

cmp dx, [corner2\_InstructionsExit+2]

jge label\_mouse\_instructions

jmp game\_opening

level:

call sound

inc [levelcount]

;jump tutorial

cmp [levelcount],1

je Game\_open

;jump level 2

cmp [levelcount],2

je level\_2\_1

;jump level 3

cmp [levelcount],3

je level\_3\_1

;jump end game

cmp [levelcount],4

je end1

Game1:

jmp Game

exit2:

jmp exit1

; Process tutorial file

tutorial:

; Graphic mode

mov ax,13h

int 10h

mov [levelcount],0

mov ax, [player\_tutorial\_x]; sets player x

mov [player\_x],ax

mov ax, [player\_tutorial\_y]; sets player y

mov [player\_y],ax

call OpenFile\_tutorial;open file

call ReadHeader

call ReadPalette

call CopyPal

call CopyBitmap

mov si,7

jmp Game

; Process level 1 file

level\_1:

; Graphic mode

mov ax,13h

int 10h

mov ax, [player\_x1]; sets player x

mov [player\_x],ax

mov ax, [player\_y1]; sets player y

mov [player\_y],ax

call OpenFile\_level1;open file

call ReadHeader

call ReadPalette

call CopyPal

call CopyBitmap

mov si,5

jmp Game

level\_2\_1:

jmp level\_2

end1:

jmp EndGame

level\_3\_1:

jmp level\_3

;Process level 2 file

level\_2:

; Graphic mode

mov ax,13h

int 10h

mov ax, [player\_x2]; sets player x

mov [player\_x],ax

mov ax, [player\_y2]; sets player y

mov [player\_y],ax

call OpenFile\_level2;open file

call ReadHeader

call ReadPalette

call CopyPal

call CopyBitmap

mov si,3

jmp Game

checkpoint\_level\_1:

jmp level\_1

checkpoint2:

jmp start

;Process level 3 file

level\_3:

; Graphic mode

mov ax,13h

int 10h

mov ax, [ player\_x3]; sets player x

mov [player\_x],ax

mov ax, [ player\_y3]; sets player y

mov [player\_y],ax

call OpenFile\_level3;open file

call ReadHeader

call ReadPalette

call CopyPal

call CopyBitmap

mov si,2

jmp Game

;==================================================backgrounds

;==================================================player movement

Game: ;moving and controlling the player

call player;drawing the player

mov ah,8h

int 21h

;=============================================Errow keys

cmp al,48h

je moveup

cmp al,4bh

je moveleft

cmp al,4dh

je moveright

cmp al,50h

je movedown

;=============================================Errow keys

cmp al,27

je exit1

jmp Game

moveup:

call read\_player\_up

call clear\_player

inc [color]

dec [player\_y]

jmp Game

checkpoint1:

jmp checkpoint2

moveleft:

call read\_player\_left

call clear\_player

dec [color]

dec [player\_x]

jmp Game

moveright:

call read\_player\_right

call clear\_player

inc [color]

inc [player\_x]

jmp Game

movedown:

call read\_player\_down

call clear\_player

dec [color]

inc [player\_y]

jmp Game

exit1:

jmp exit

;==================================================player movement

EndGame:

; Graphic mode

mov ax,13h

int 10h

call OpenFile\_EndGame

call ReadHeader

call ReadPalette

call CopyPal

call CopyBitmap

label\_mouse\_EndGame:

call mouse

comper\_EndGame\_yes:

cmp cx, [corner1\_yes\_EndGame]

jle comper\_EndGame\_no

cmp dx, [corner1\_yes\_EndGame+2]

jle comper\_EndGame\_no

cmp cx, [corner2\_yes\_EndGame]

jge comper\_EndGame\_no

cmp dx, [corner2\_yes\_EndGame+2]

jge comper\_EndGame\_no

jmp checkpoint1

comper\_EndGame\_no:

cmp cx, [corner1\_no\_EndGame]

jle label\_mouse\_EndGame

cmp dx, [corner1\_no\_EndGame+2]

jle label\_mouse\_EndGame

cmp cx, [corner2\_no\_EndGame]

jge label\_mouse\_EndGame

cmp dx, [corner2\_no\_EndGame+2]

jge label\_mouse\_EndGame

jmp exit

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

exit :

mov [levelcount],0

text\_mode:

; Back to text mode

mov ah, 0

mov al, 2

int 10h

mov ax, 4c00h

int 21h

END start