תיק פרויקט:

GPaint



מתכנת: גיא פורת

ת.ז.: 211675400

מורה: יוסי זהבי

כיתה: י' 4

בית ספר: תיכון הרצוג

1. מבוא

שם העבודה שלי GPaint (משחק מילים לא מאוד מוצלח על Guy Porat ו-paint) והיא פלטפורמה ויזואלית לציור (תכל"ס, צייר).

שם הקובץ (שצריך לקמפל):

PaintTop – הקובץ שמהווה את הקובץ הראשי.

קבצים נלווים:

GenProcs.asm – מכיל פעולות כלליות.

Shapes.asm – מכיל את הפעולות הקשורות לצורות.

Bres\_Rev.asm – מכיל את הפעולה הבסיסית של ציור קו אלכסוני, נכתב על ידי יוסי זהבי ושונה מעט על ידי. בלוגיקה נחשב כתת מחלקה של Shapes.asm.

SaveLoad.asm – מכיל פעולות הקשורות לטעינה ושמירה של קבצי BMP כמו גם פעולות היפוך (ימינה-שמאלה או למעלה-למטה).

Tools.asm – מכיל פעולות לכל הכלים שלא כלולים בקבצים (מחלקות) לעי"ל.

UI.BMP – קובץ תמונה המכיל את תמונת הממשק (וממנו נטען הממשק).

סביבת עבודה: Turbo Assembler (TASM)

סביבת הפיתוח: Notepad++

סביבת הרצה: Dosbox (או כל מחשב/סביבת הרצה מריץ/ה 16- or 32-bit x86 MS-DOS ).

כיצד לקמפל:

יש להפעיל את DosBox ולנווט לתיקייה בה נמצאים הקבצים.

יש לה

1. נושא העבודה

GPaint היא פלטפורמה ויזואלית לציור ומטרתה לאפשר לכל משתמש ליצור ולשנות תמונות בפורמט BMP. בפלטפורמה מצויים כלים שונים ומגוונים המדמים תהליך ציור כגון: צורות, ציור חופשי, מחק וכדומה.

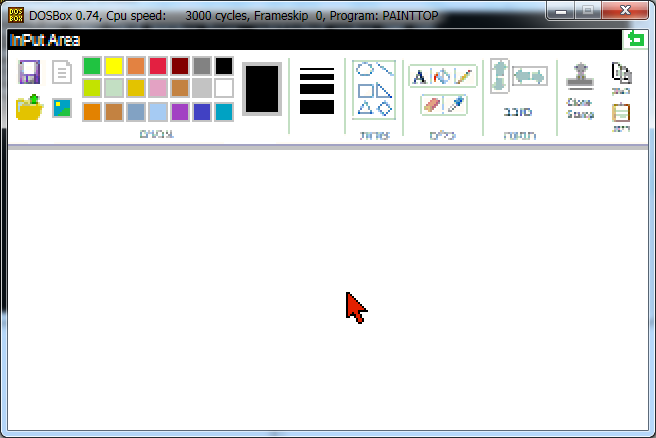
1. אופן ההפעלה:

הפעלה:

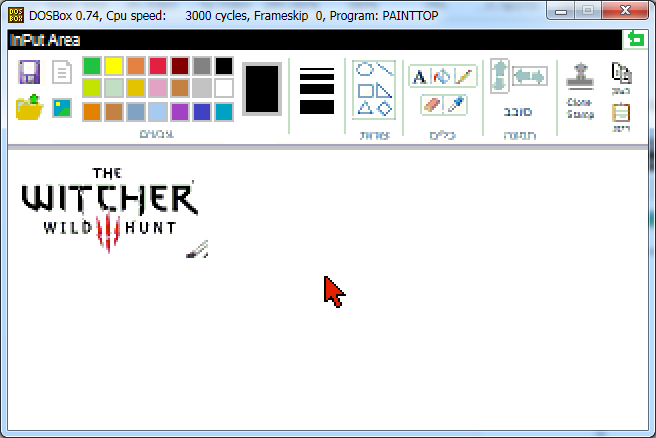
את הצייר ניתן להפעיל בצורה רגילה שבה יפתח עם קנבס ריק (PaintTop.exe) אך אם נוסיף אחרי ה- PaintTop.exeשם של קובץ BMP יפתח הצייר עם התמונה (PaintTop.exe File.bmp)

לדוגמא:

PaintTop.exe



PaintTop.exe Witcher3.bmp



תפעול:

ככלל דומים הכלים לצייר של מיקרוסופט המצוי בווינדוס וכך גם השימוש בהם. חריגות לכלל זה הן:

1. כשכלי נבחר מוגבלת תנועת העכבר לתוך תחומי הקנבס, על מנת להוציא את העכבר מהקנבס ולצאת מהכלי יש ללחוץ קליק ימני. (למעט בפעולות SAVE\_FILE,OPEN\_FILE,LOAD\_IMAGE ו-TEXT)
2. בפעולות SAVE\_FILE ו-OPEN\_FILE, הדרך לצאת מהכלי הינה באמצעות הקשת הלחצן ENTER במקלדת.
3. בפעולות TEXT ו-LOAD\_IMAGE לאחר שנבחר מיקום יש ללחוץ ENTER כדי לצאת משלב הזנת הטקסט (בפעולת TEXT לאחר הלחיצה על ENTER ניתן לצאת מהכלי בעזרת לחיצה ימנית או לחילופין לבחור מיקום נוסף).
4. לחזור אחורה באמצעות כפתור החזרה ניתן רק לעשות לצעד אחד אחורה והמצב יוחזר למצב שהיה לפני בחירת הכלי האחרון.
5. הצורות לא מושפעות מהעובי שנבחר ותמיד יצוירו בעובי 1.
6. בכלי ה-CLONE\_STAMP יש לציין ראשית מקום ממנו יועתק המידע (על מנת לחשב רלטיביות), על כן יש לבחור ראשית את המיקום ממנו יועתק המידע באמצעות הלחצן האמצעי בעכבר (גלגלת).
7. על מנת לסגור את הצייר ולצאת יש ללחוץ על הלחצן ESC, אך היו זהירים! מידע שלא נשמר טרום סגירת התוכנית יימחק.

גרסאות המערכת:

|  |  |
| --- | --- |
| 0.99 - | * הכלים: Copy, Paste, Clone\_Stamp, Turn\_Upwards\_Down, Turn\_Rightside\_Left, Text, Fill, Pencil, Eraser, Sampler, Shapes, Open\_File, Save\_File, Refresh\_Canvas, ו- Load\_Image * האפשרות לבחור צבעים מפלטה בממשק * האפשרות לבחור עובי מברשת (לא משפיע על עובי הצורות) * גרסת בטא סופית |
| 1.0 - | * תיקון באגים * גרסת שיחרור ראשונה |
| 1.1 - | * תיקון באגים * כלי המאפשר לחזור צעד אחורה למצב שלפני בחירת הכלי הקודם |
| 1.2 - | * תיקון באגים * הוספת יכולת לטעון תמונה משורת פקודה |
| 1.201 - | * ייעול משמעותי של Mouse\_Handler * שיפור תיעוד |
| 1.202 - | * תיקון באגים |
| 1.203 - | * שינוי קטן ל-Load\_Bmp\_Data\_To\_Screen |

ארצה לעשות אפשרות הצפנת ופיענוח תמונה (לפי מפתח קבוע או אחד שנוצר מנוסחה קבועה ומנתונים מהתמונה, לא הצפנה מסובכת). כמו כן, ארצה לשנות את פונקציית ציור המלבן כך שתצייר רק את השוליים ולא את המילוי.

הסבר הפיתרון:

אופן המימוש:

התוכנית בנויה בצורה אסינכרונית ומתחלקת לשני נתיבים לוגיים:

1. ישנה לולאה שתנאי העצירה שלה לחיצה על הלחצן 'ESC'.
2. ישנה פעולה אסינכרונית הנקראת בכל לחיצה הבודקת האם ועל איזה כפתור לחץ המשתמש ואם נלחץ אחד הכפתורים היא קוראת לפעולה של הכפתור.

הלולאה כתובה ב-PAINTTOP שמקביל בתוכנית שלי למחלקת MAIN:

ראשית נטען הממשק, הקנבס מאופס והפעולה האסינכרונית נרשמת. לאחר מכן, נכנס הקוד ללופ שכביכול כל מה שהוא עושה זה לבדוק האם תנאי היציאה (לחיצה על הלחצן ESC) התבצע.

בעוד שהלופ רץ, בכל לחיצת עכבר שמאלית הפעולה האסינכרונית (Mouse\_Handler) נקראת. הפעולה האסינכרונית בודקת בכל לחיצה על הכפתור השמאלי של העכבר האם הלחיצה הייתה בתחומי הממשק ואם כן איזה כפתור נלחץ. לאחר מכן, היא קוראת לכפתור (לאחר שהיא מטפלת בנושא Ctrl\_Z\_Bakeup, החבאת העכבר והגבלת תנועת העכבר במידת הצורך).

מכאן מתפצל הקוד לנתיבים רבים, לכל כלי פעולה משלו ולה אלגוריתם עצמאי.

חריגה לכלל זה הן הצורות ולהן אלגוריתם כללי משותף: Print\_Shape\_Button\_Rev. הוא מחשב את הגובה והאורך של הצורה לפני שהוא קורא לפעולה לציור הצורה (לכל צורה פעולה משלה).

את הממשק טוענים לפני תחילת הלולאה הראשית ובכל לחיצה מהקובץ הנלווה UI.BMP.

משתנים חשובים:

**General Vars:**

Color – dictates the current color.

CommandLine – buffer used receive the input from PSP in "Read\_Command\_Line\_From\_PSP" proc (from the command line entered after "PaintTop.exe ")

CommandLine\_Length – after "Read\_Command\_Line\_From\_PSP" proc activation will contain the number of chars put in CommandLine.

**SaveLoad Vars:**

File\_Name – buffer that receives name in "Recive\_Input\_To\_File\_Name\_Buffer" proc in asciiz and is used in "Save\_File" and " and "Open\_File\_Button" procs.

Header – buffer to store the loaded bmp's header.

Pallette – buffer to store the loaded bmp's palette.

FileHandle – buffer to store the loaded bmp's File Handle.

UI\_File\_Name\_Var – stores the name of the ui bmp file in asciiz

UI\_File\_Handle Stores the UI's file handle.

Default\_Header\_To\_Save – An array of bytes used as header for the files saved using Save\_File proc.

**Shapes Vars:**

shp\_Type – dictates what shape the Print\_Shape will draw (Circle = 0, Diagonal Line = 1, Rect = 2, Triangle\_90\_Deg = 3, Isosceles\_Triangle = 4, Rhombus = 5)

shp\_Height, shp\_Length, shp\_PressX, shp\_PressY, shp\_ReleaseX, shp\_ReleaseY – used extensively in calculating where and how to draw the shape.

**Tools Vars:**

Thickness – dictates the thickness of the tools (except for shapes).

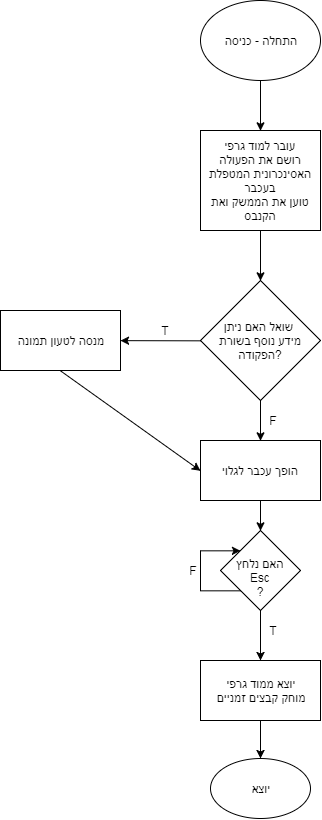
Text - used as buffer for int 09 of the 21h family in Text tool

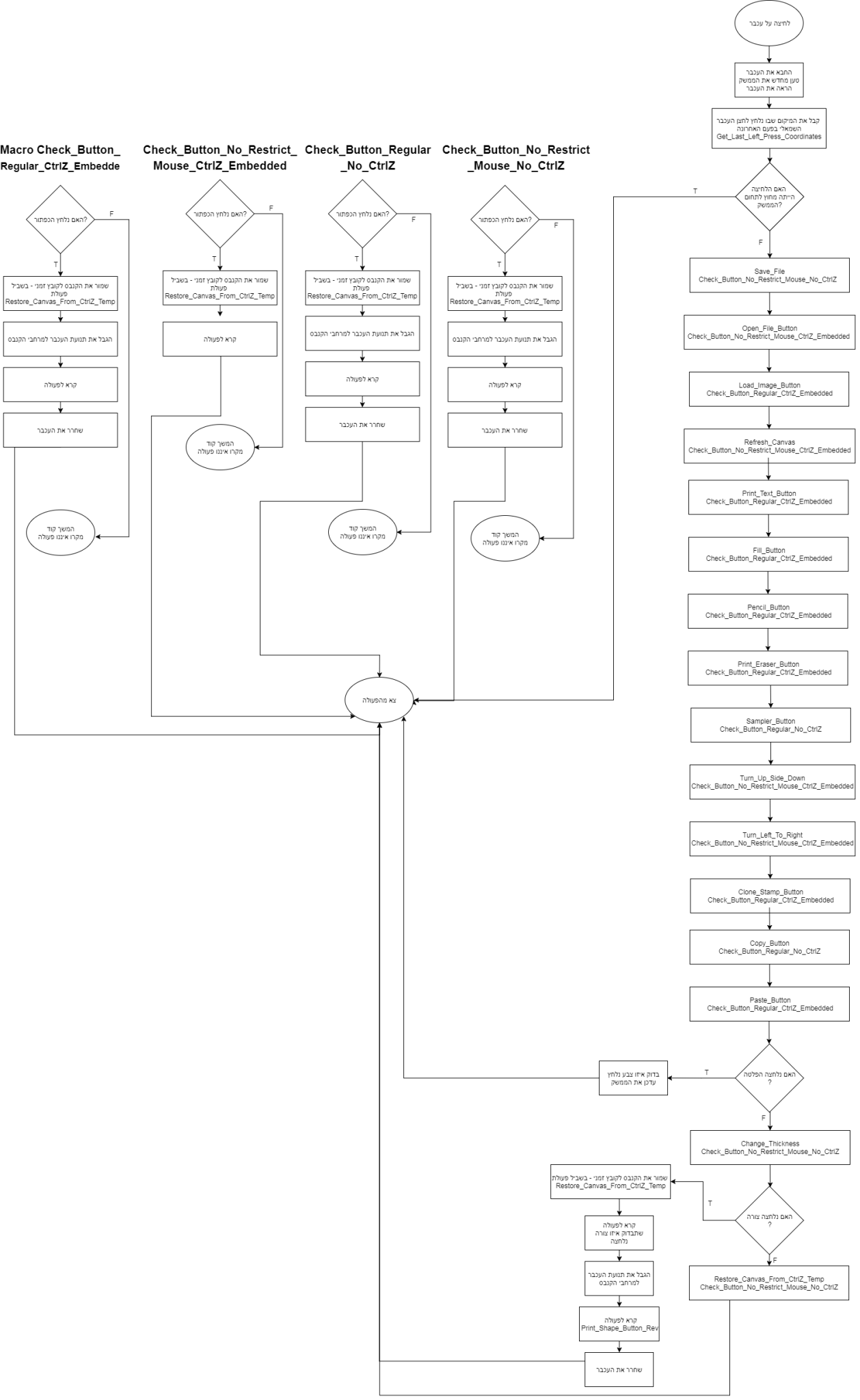
**Copy Paste Vars:**

CP\_Length, CP\_Height, CP\_X, CP\_Y – משמשים בהעתקה והדבקה

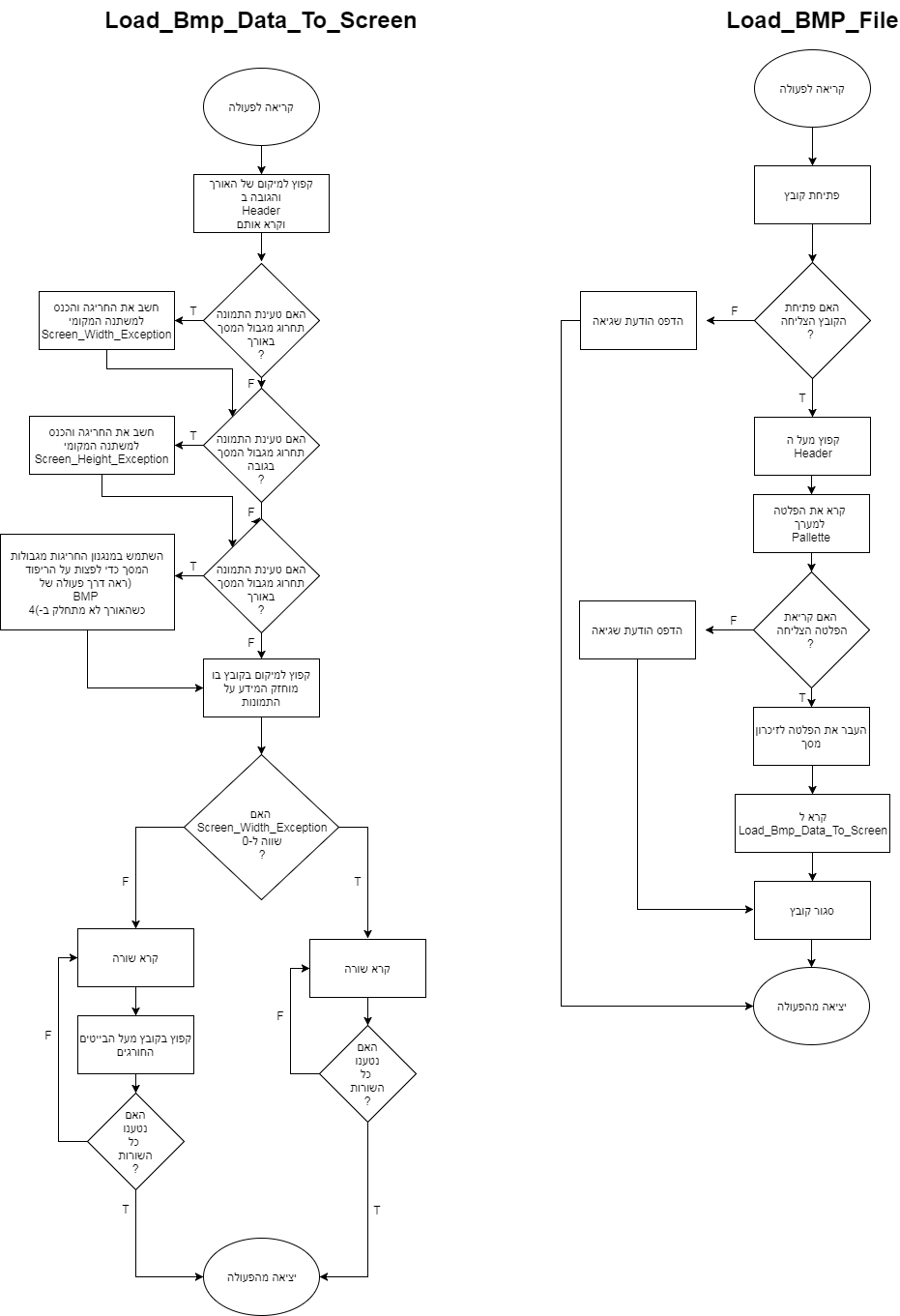
תרשימי זרימה:

מחלקת MAIN



Mouse\_Handler 

טעינת קובץ BMP



(תרשימי זרימה נעשו בעזרת https://www.draw.io/)

רשימת הפעולות:

PaintTop

Proc Set\_Mouse\_Routine

Sets the routine the program will follow if the left button is released

input: none

output: none

Proc Mouse\_Handler far

Handles mouse clicks, checks if and what button was pressed

input: none

output: none

Proc Change\_Thickness

Checks if any of the change Thickness buttons were pressed and changes [Thickness] accordingly

input: recievs Y value at dx

output: [Thickness]

Proc Find\_What\_Shape\_Was\_Clicked

Checks what Shape button was pressed and calls the proper Proc

input: X value ac cx and Y value at dx

output: none (indirectly: shp\_Type)

SaveLoad

Proc Open\_File\_By\_CommandLine

Open File By CommandLine

Loads bmp file with name specified in CommandLine var at canvas

input: var in DATASEG - CommandLine

output: Screen Memory

Proc Command\_Line\_Invalid\_Error

print Command Line Invalid Error Message

input: Command\_Line\_Invalid\_Error\_Msg var at DATASEG

output: Screen Memory(prints the Command\_Line\_Invalid\_Error\_Msg to screen)

Proc Save\_Canvas\_To\_CtrlZ\_Temp

Save Canvas bytes To CtrlZ\_Temp

input: Screen Memory

output: To CtrlZ\_Temp

Proc CtrlZ\_Temp\_File\_Not\_Found\_Error

Prints CtrlZ\_Temp File Not Found Error

input: none

output: none

Proc Restore\_Canvas\_From\_CtrlZ\_Temp

Restores Canvas From CtrlZ\_Temp

input: CtrlZ\_Temp

output: Screen Memory

Proc Reset\_Graphic\_Cursor\_Position

Reset Graphic Cursor Position to 0,0 position

input: none

output: none

Proc Load\_Image\_Button

Load Image Button

Recives file name and position in canvas from user and Loads the selected bmp file at selected position

input: none (requires file name to be typed and a position to be selected by user)

output: Screen Memory

Special Exit Condition: Right Click

Proc Open\_File\_Button

Open File Button

Receives file name from user and Loads the selected bmp file at canvas

input: none (requires file name to be typed by user)

output: Screen Memory

Proc Save\_File

Save File Button

Recives file name from user and Saves the canvas as a bmp file with the name typed

BeAware! The Proc won't work properly if the Screen\_Width isn't divisible by 4!

input: Canvas at Screen Memory (requires file name to be typed by user)

output: File

Proc Save\_Image\_OnScreen\_To\_BMP

Save Canvas OnScreen To BMP

Saves the canvas as a bmp file with the name starting at (File\_Name + 2)

BeAware! The Proc won't work properly if the Screen\_Width isn't divisible by 4!

input: File\_Name Array at DATASEG

output: File

Proc Recive\_Input\_To\_File\_Name\_Buffer

Receive Input To File Name Buffer

input: none (requires the user to type a string and press enter)

output: File\_Name array at DATASEG (Length of inputed string as byte at File\_Name+1 and string start at File\_Name+2)

Proc Load\_UI

Loads the tools bar, recommended after every action that may paint over it

input: UI\_File\_Name\_Var Var at DATASEG

output: Screen Memory(prints the UI to screen)

Proc Load\_BMP\_File

input:

File\_Name var at DATASEG

param1 = Name Buffer Pointer

param2 = X value

param3 = Y value

output: Screen Memory(prints the bmp file to screen)

Proc Pallette\_Error

input: Pallette\_Error\_Msg var at DATASEG

output: Screen Memory(prints the Pallette\_Error\_Msg to screen)

Proc File\_Not\_Found\_Error

input: File\_Not\_Found\_Error\_Msg var at DATASEG

output: Screen Memory(prints the File\_Not\_Found\_Error\_Msg to screen)

Proc Put\_Pallette\_Var\_To\_Screen\_Memory

Will move out to screen memory the colors from Pallette var in DATASEG

video ports are 3C8h for number of first color

and 3C9h for all rest

Wrritten by Yossi Zehavi, slightly changed by me

input: var in DATASEG: Pallette

output: none (to cpu,changes the palette of the screen)

Proc Load\_Bmp\_Data\_To\_Screen

BE AWARE! After the Proc the pointer inside the file will point to the last byte

BE AWARE! bmp format saves the image so the last line is the first in the file (the order of byte inside the line is untouched) and first is the last - baisically upside down.

input: (receives through stack)

param1 = File Handle

param2 = X value, where to start loading the image

param3 = Y value, where to start loading the image

output: File

Proc Copy\_A\_Full\_Line\_From\_Screen\_To\_Screen\_Line\_Var

Copy A Full Line From the Screen To Screen Line Var

input: line's Y value at ax

output: Screen\_Line var at DATASEG

Proc ReadBmpHeader near

Read bmp Header to Header var

Read 54 bytes the Header

input: FileHandle var at DATASEG

output: Header var at DATASEG

Proc Turn\_Up\_Side\_Down

Turns Canvas Up Side Down

input: Screen Memory

output: Screen Memory

Proc Turn\_Left\_To\_Right

Turns Canvas Left To Right

input: Screen Memory

output: Screen Memory

Shapes

Proc Print\_Shape\_Button\_Rev

Lets the user select a rectangular space and draws a shape in it

Tip: Use the Set... Proc to change the shape.

input: var in DATASEG: [shp\_Type]

output: Screen Memory

Special Exit Condition: Right Button Mouse Click

Proc Set\_Shape\_To\_Circle

sets [shp\_Type] to circle value

input: none

output: var in DATASEG: shp\_Type

Proc Set\_Shape\_To\_Diagonal\_Line

sets [shp\_Type] to Diagonal Line value

input: none

output: var in DATASEG: shp\_Type

Proc Set\_Shape\_To\_Rect

sets [shp\_Type] to Rectangle value

input: none

output: var in DATASEG: shp\_Type

Proc Set\_Shape\_To\_Triangle\_90\_Deg

sets [shp\_Type] to Triangle\_90\_Deg value

input: none

output: var in DATASEG: shp\_Type

Proc Set\_Shape\_To\_Isosceles\_Triangle

sets [shp\_Type] to Isosceles Triangle value

input: none

output: var in DATASEG: shp\_Type

Proc Set\_Shape\_To\_Rhombus

sets [shp\_Type] to Rhombus value

input: none

output: var in DATASEG: shp\_Type

Proc Print\_Shape

takes the points of last release and press (of the left mouse button) and prints in the rectangular space between them the shape.

Tip: Use the Set... Proc to change the shape.

input: var in DATASEG: [shp\_Type], Last Left Press info and Last Left Release info

output: Screen Memory

Proc Print\_Shape

takes the points of last release and press (of the left mouse button) and prints in the rectangular space between them the shape.

Tip: Use the Set... Proc to change the shape.

input: var in DATASEG: [shp\_Type], Last Left Press info and Last Left Release info

output: Screen Memory

Proc Calc\_Height\_And\_Length\_Then\_Print

calculates shp\_Height and shp\_Length based on Press and release coordinates and then chooses what shape to print based on shp\_Type value

input: vars in DATASEG: shp\_Type, shp\_ReleaseX, shp\_ReleaseY, shp\_PressX, shp\_PressY

output: Screen Memory

Proc Calc\_Height\_And\_Length

calculates shp\_Height and shp\_Length based on Press and release coordinates

input: vars in DATASEG: shp\_ReleaseX, shp\_ReleaseY, shp\_PressX, shp\_PressY

output: Screen Memory

Proc Draw\_Rect

Draws rect, capable of handling negative length and height params.

input: (through stack)

param1 = height

param2 = Length

param3 = X value

param4 = Y value of the uppermost point

output: Screen Memory

Proc Print\_Rect

Draw\_Rect's Case Proc

Draws rect, capable of handling negative length and height params.

input: vars in DATASEG: shp\_Height,shp\_Length,shp\_PressX,shp\_PressY

output: Screen Memory

Proc Print\_Isosceles\_Triangle

Draw\_ Isosceles\_Triangle's Case Proc

Draws an isosceles triangle, capable of drawing to any direction (if shp\_Height is neg it will be printed downwards).

input: vars in DATASEG: shp\_Height,shp\_Length,shp\_PressX,shp\_PressY

output: Screen Memory

Proc Draw\_Isosceles\_Triangle\_Rev

DESCRIPTION: draws an isosceles triangle upwards with base length eqhivelent to double the height

input: (through stack)

Param1 = Height

Param2 = Base Length

Param3 = X value

Param4 = Y value of the lowest point

output: Screen Memory

Proc Draw\_Horizontal\_Line

Draws an Horizontal Line

input:

si = length

al = color

cx = X value

dx = Y value

output: Screen Memory

Proc Draw\_Horizontal\_Line

Draws an Horizontal Line

Tip: use Draw\_Horizontal\_Line instead, its more efficient

input:

si = length

al = color

cx = X value

dx = Y value

output: Screen Memory

Proc Draw\_Vertical\_Line

Draws a vertical Line

input:

si = length

al = color

cx = X value

dx = Y value

output: Screen Memory

Proc Print\_Rhombus

Draw\_Rhombus' Case Proc

Draws a Rhombus, capable of drawing to any direction.

input: vars in DATASEG: shp\_Height,shp\_Length,shp\_PressX,shp\_PressY

output: Screen Memory

Proc Draw\_Rhombus\_Rev

Draws a Rhombus, capable of drawing to any direction.

input: (through stack)

Param1 = Height

Param2 = Base Length

param3 = X value

param4 = Y value of the lowest point

output: Screen Memory

Proc Print\_Circle

Draw\_Circle's Case Proc

Draws a Circle, capable of receiving negative shp\_Height and shp\_Length.

BeAware! The circle's center is the center of the selected rectangular space

input: vars in DATASEG: shp\_Height,shp\_Length,shp\_PressX,shp\_PressY

output: Screen Memory

Proc Print\_Triangle\_90\_Deg

Draw\_Triangle\_90\_Deg's Case Proc

Draws a 90 Degrees Triangle, capable of receiving negative shp\_Height and shp\_Length.

BeAware! The two catheti always protrude from Press Point

input: vars in DATASEG: shp\_Height,shp\_Length,shp\_PressX,shp\_PressY

output: Screen Memory

Proc Draw\_Triangle\_90\_Deg

draws a right angled triangle

input: (through stack)

param1 = height

param2 = Length

param3 = X value

param4 = Y value

output: Screen Memory

Proc Print\_Diagonal\_Line

DrawLine2D's Case Proc

Draws a Diagonal Line

input: vars in DATASEG: shp\_PressX,shp\_PressY,shp\_ReleaseX,shp\_ReleaseY

output: Screen Memory

Proc circleV3\_Rev

Draws a circle (non recursive)

Eliminated dependency on es the original Proc had (es can be whatever before activation and the Proc would still work)

input: (through stack)

cx = radius

di = point on screen where to put the center of circle

al - color

output: Screen Memory

Bres\_Rev

Proc DrawLine2D

Draws a line from Point1 to Point2

input: vars in DATASEG: point1X, point1Y, point2X, point2Y, Color

output: Screen Memory (line on the screen)

Tools

Proc Paste\_Button

Paste Button

Lets the user select a point to paste the rect to, should only be used after Copy\_Button Proc

input: CP\_Temp\_File file and Vars in DATASEG - CP\_Height,CP\_Length,CP\_Temp\_File\_Name\_Var

output: Screen Memory

Special Exit Condition: Enter (interrupt's exit condition)

Proc Read\_Rect\_From\_CP\_Temp\_File\_To\_Screen

Read Rect From CP Temp File To\_Screen

Pastes the rect to the location provided, should only be used after Copy\_Button Proc

input: CP\_Temp\_File file, Vars in DATASEG - CP\_X,CP\_Y,CP\_Height,CP\_Length,CP\_Temp\_File\_Name\_Var

output: Screen Memory

Proc Nothing\_To\_Paste\_Error

Prints CP\_Error\_Msg

input: none

output: none

Proc Delete\_Temps\_If\_Exits

Deletes all Temporary files If they exist, should be done at exit

input: Vars in DATASEG - CP\_Temp\_File\_Name\_Var,CtrlZ\_Temp\_File\_Name\_Var

output: none

Proc Copy\_Button

Lets the user select a rectangular piece of the screen and copies it to CP\_Temp\_File

input: none

output: to CP\_Temp\_File (data is not stored as bmp or as any standard format)

Special Exit Condition: Right Click

Proc Store\_Rect\_In\_TEMP

saves the data in selected rect to file

input: vars in DATASEG - CP\_Height,CP\_Length,CP\_X,CP\_Y

output: to CP\_Temp\_File (data is not stored as bmp or as any standard format)

Proc Select\_Rect\_To\_Copy

The user draws a rect on screen using the mouse and the Proc makes perparations for Store\_Rect\_In\_TEMP Proc activation

input: none

output: vars in DATASEG: CP\_Height,CP\_Length,CP\_X,CP\_Y

Special Exit Condition: Right Click

Proc Clone\_Stamp\_Button

Clone Stamp Button, has the same functionality as the one in photoshop

Point to copy from is set first with a click of the middle mouse button

Point to set Relativity is set with the first click from the left mouse button

input: Var in DATASEG: Thickness

output: Screen Memory

Special Exit Condition: Right Click

Proc Clone\_Stamp

copies a squere around Point1 to Point2

input: recieves Point1 loacation in Screen Memory in di and Point2 loacation in Screen Memory in si

output: Screen Memory, si and di are destroyed

Proc Fill\_Button

Color all adjacent pixels with the same color to [Color]. (RECURSIVE)

input: var in DATASEG: [Color]

output: Screen Memory

Special Exit Condition: Right Click

Proc Fill

Fill Core's case Proc (RECURSIVE)

Handles input and calling to Fill\_Core

input: var in DATASEG: [Color], last left mouse click info

output: Screen Memory

Proc Fill\_Core

(TIP: Don't use, Use Fill)

BE AWERE! You shouldn't use this Proc as it requires to fill the input manually and lacks certein safeguareds, use Fill Instead

A recursive Proc that checks if the color is the color inputed in dh and if yes changes it to [Color]

Based on Yossi Zehavi's Flood in bmp program

input: cx = X, dx = Y, bl = Color to Fill,

bh = page number (in dosbox, always 0),ah = 0dh

output: Screen Memory

Proc Sampler\_Button

Sampler Button

Changes [Color] to the color of pixel the user presses with the mouse left button

input: Screen Memory

output: var in DATASEG: [Color]

Special Exit Condition: Right Click

Proc Update\_Color\_Rect\_In\_UI

Updates the Rect In UI that shows what color is chosen to the curren value of Color var

input: var in DATASEG: [Color]

output: Screen Memory

Proc Sampler

Changes [Color] to the color of pixel the user last pressed with the mouse left button

input: Screen Memory

output: var in DATASEG: [Color]

Proc Print\_Eraser\_Button

Print Eraser Button

Paint where the user presses a squere the length of [Thickness] in White

input: none

output: Screen Memory

Special Exit Condition: Right Click

Proc Pencil\_Button

Pencil Button

Paints where the user presses a squere the length of [Thickness] in [Color]

input: var in DATASEG: [Color]

output: Screen Memory

Special Exit Condition: Right Click

Proc Print\_Pencil

Paints a squere the length of [Thickness] in [Color] around the last left mouse button click location

input: var in DATASEG: [Color]

output: Screen Memory

Proc Print\_Text\_Button

allows the user to write using interrupt ah=9h of int 21h family at the location he presses the left mouse butoon

input: none

output: Screen Memory

Special Exit Condition: Right Click

Proc Print\_Text

allows the user to write using interrupt ah=9h of int 21h family at the location he last pressed the left mouse butoon

input: none

output: Screen Memory

Special Exit Condition: Enter (interrupt's exit condition)

GenProcs

Proc Calc\_Loc\_In\_Screen\_Memory

Calculates Location (address) In Screen Memory

input: Y value in dx and X Value in cx

output: address in ax (Y\*Screen\_Width+X)

Proc Get\_Last\_Left\_Press\_Coordinates

Gets Last Left Press info

input: none

output: x value in cx, y value at dx and mouse status in AX

( Mouse status guide:

bit 0 = 1 (1) - Left button pressed,

bit 1 = 1 (2) - Right button pressed,

bit 2 = 1 (4) - Middle button pressed )

Proc Get\_Last\_Left\_Release\_Coordinates

Gets Last Left Release info

input: none

output: x value in cx and y value at dx

Proc Get\_Mouse\_Position\_and\_Button\_Status

Get Mouse Position and Button Status

input: none

output: x value in cx, y value at dx and mouse status in AX

( Mouse status guide:

bit 0 = 1 (1) - Left button pressed,

bit 1 = 1 (2) - Right button pressed,

bit 2 = 1 (4) - Middle button pressed )

Proc SetGrapicMode

Sets the display moode to graphic mode

input: none

output: none

Proc SetNormalMode

Sets the display mode to normal mode

input: none

output: none

Proc Restrict\_Mouse\_to\_Canvas

Set mouse movement boundaries to within the painting ground

input: none

output: none

Proc Release\_Mouse

Release mouse movement, cancel mouse movement boundaries

input: none

output: none

Proc Hide\_Mouse\_Cursor

Hides Mouse Cursor

input: none

output: none

Proc Refresh\_Canvas

Refresh Canvas - set all canvas bytes to white value

input: none (White equ)

output: none

Proc Return\_Absolute\_Through\_Stack

Receives a value through stack and overwrites it with its absolute value

input: Parmr1 = Value (through stack)

output: Parmr1 = abs Value (through stack)

Proc Read\_Command\_Line\_From\_PSP

Read Command Line from PSP to var

Copies the command line from the buffer in the psp to CommandLine var at DATASEG in asciiz (without the 'cr' char at the end), also puts in CommandLine\_Length var the length of the command line

BeAware! Max length of Command Line is 127 and a 0 is needed at the end so CommandLine var length should be at least 128d bytes

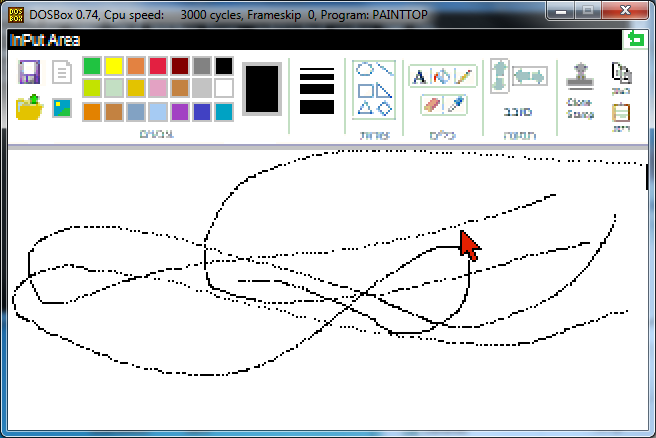
input: PSP

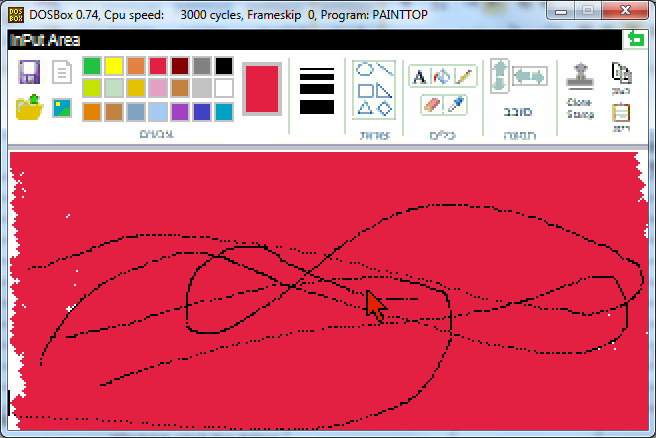
output: vars in DATASEG: CommandLine\_Length, CommandLine

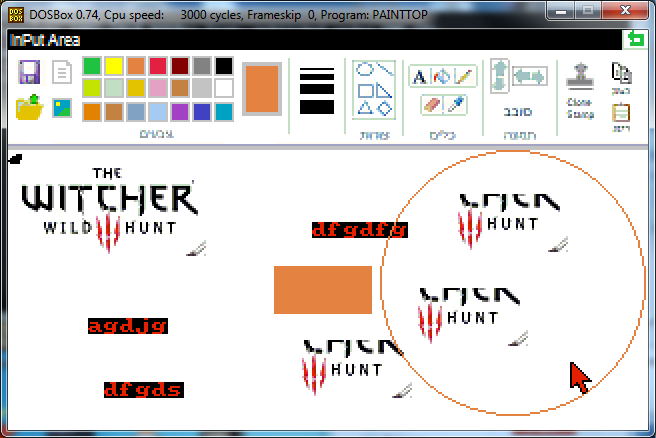
פרטים המאפשרים כניסה והפעלה:

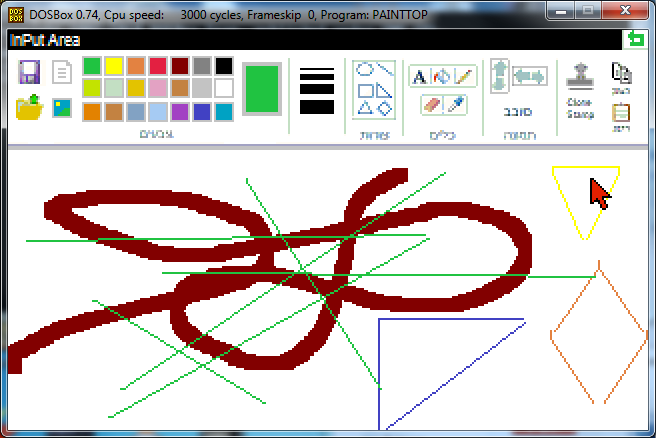
אין, תהנו מהשימוש!

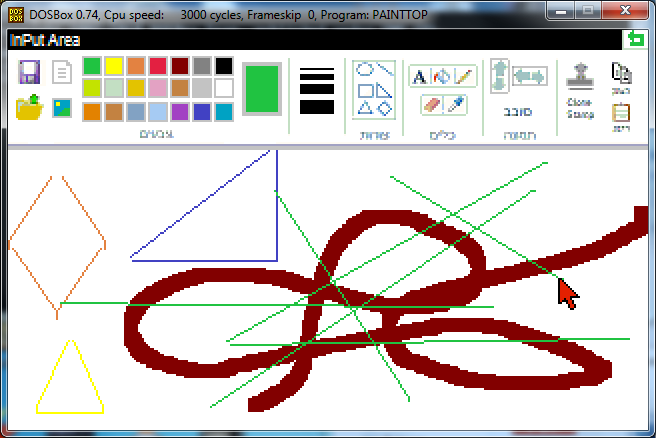
דוגמאות הרצה:











סיכום אישי:

עשיית הפרויקט הייתה מעניינת ובזמן שהוא היה מייאש לפעמים, אני חושב שהרווחתי ממנו משהו שאי אפשר להרוויח בשום דרך אחרת. אני לא חושב שיש דרך כלשהי ללמוד אסמבלי שהיא טובה כמו הפרויקט שכן ככל שהמשכתי לעבוד על הפרויקט ראיתי ממש כיצד אני משתפר מפעולה לפעולה. (לקראת סיום הפרויקט היו לי רגעים שהסתכלתי על קוד שכתבתי בתחילת הפרויקט ושאלתי את עצמי "מה חשבתי לעצמי כשכתבתי את זה?")

בסופו של דבר אני נהניתי לעשות את הפרויקט ואני חושב שהוא קידם אותי, מבלי להגזים, 'שנות אור' באסמבלי.