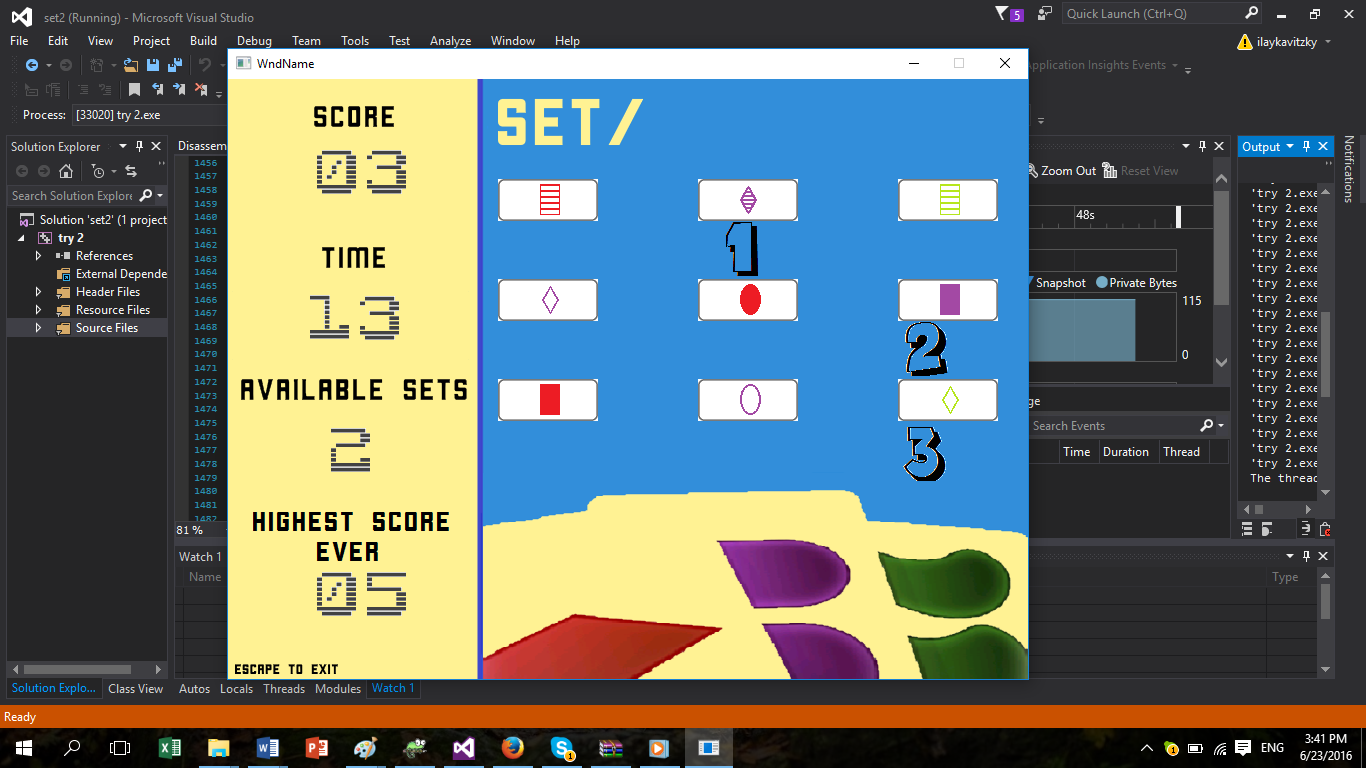
**SET**

**משחק חשיבה באסמבלי 32bit**

**עילי קויצקי**

**322520743**

תוכן עניינים:

הקדמה

תמונות

על אסמבלי

הסבר על התוכנית

הסבר על הלמידה

תהליך העבודה

קוד

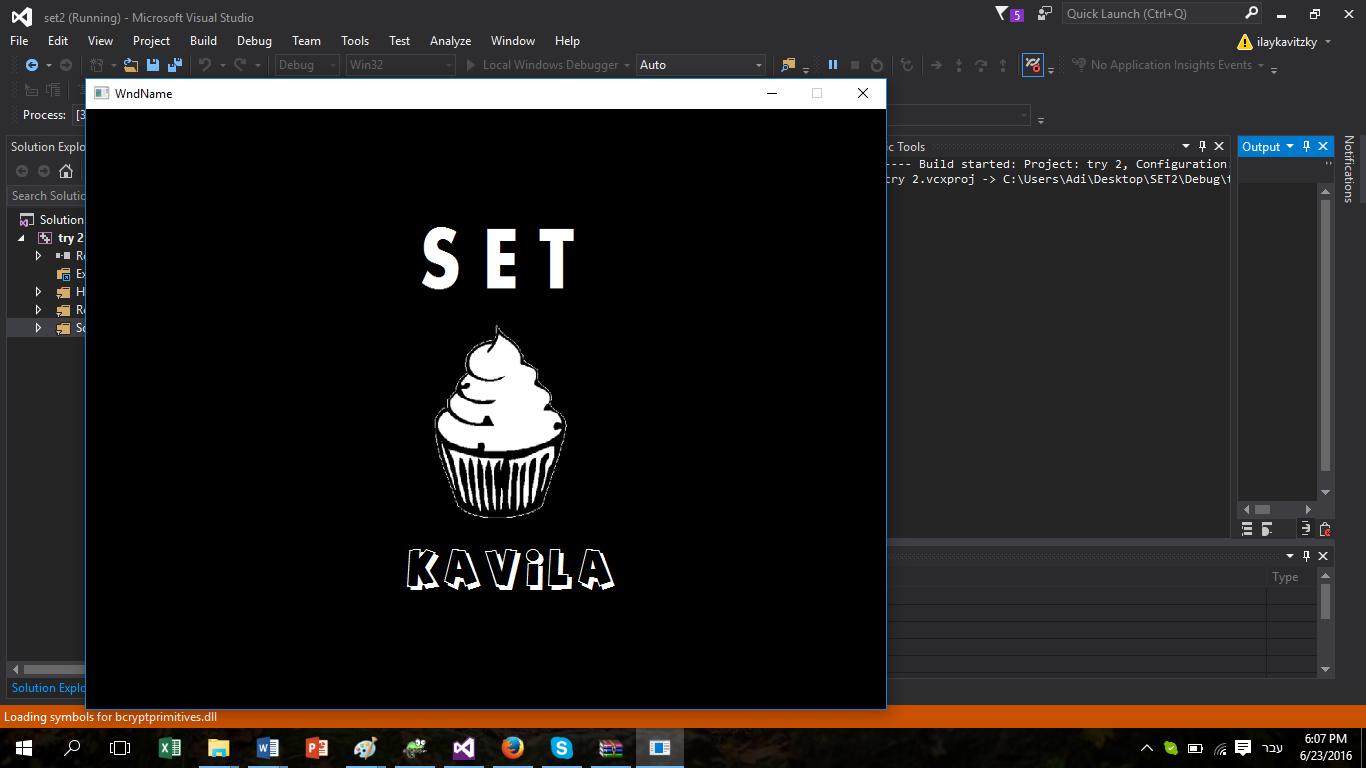
רלפקציה אישית

**הקדמה:**

ספר זה מתעד את הכנתיי של הפרוייקט שלי באסמבלי 32-ביט, פרויקט SET. ספר זה כולל את תיעוד הקוד, רפלקציה אישית שלי על הפרוייקט, הסבר על התוכנית, תהליך העבודה שלי וכן תהליך הלמידה.

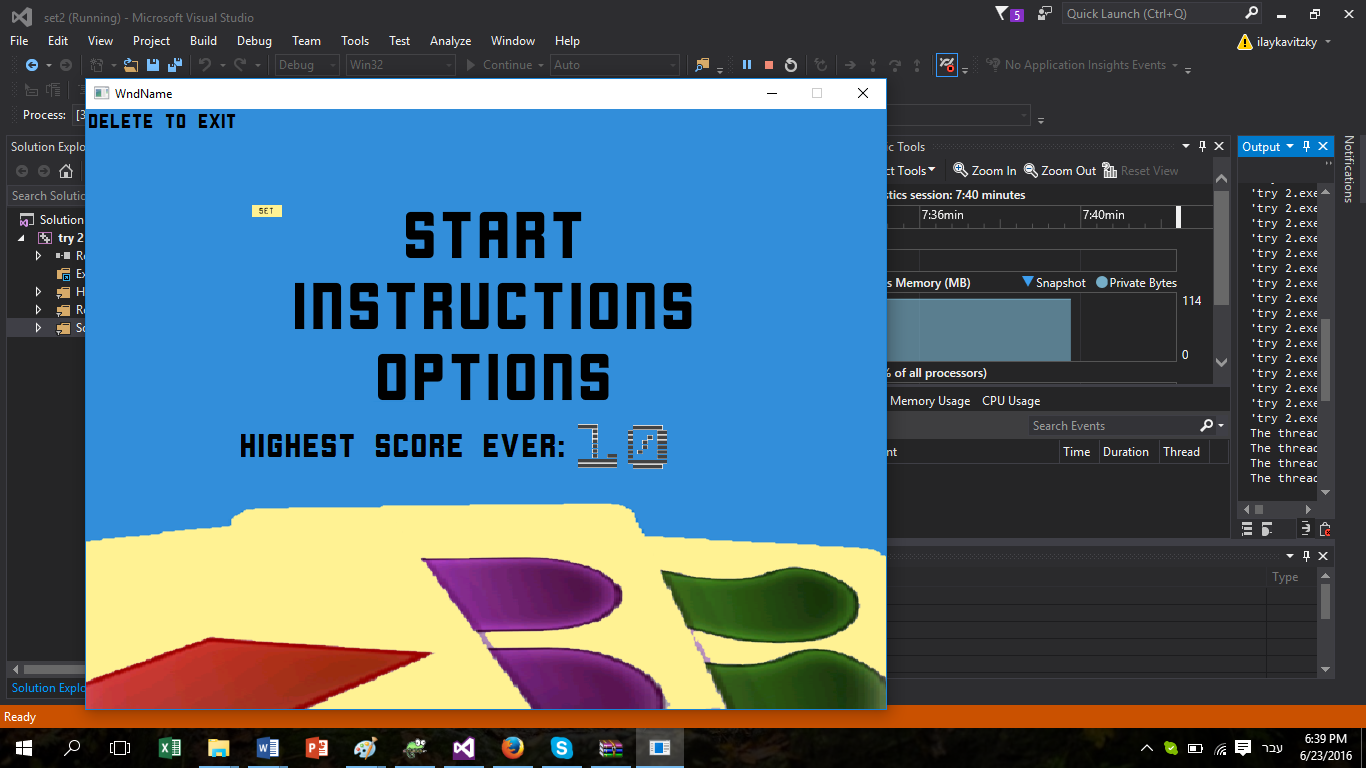
תמונות מהמשחק:

תמונת פתיחה שמכילה "קאפקייק" ש"נאכל" ועושה קול של נגיסה:

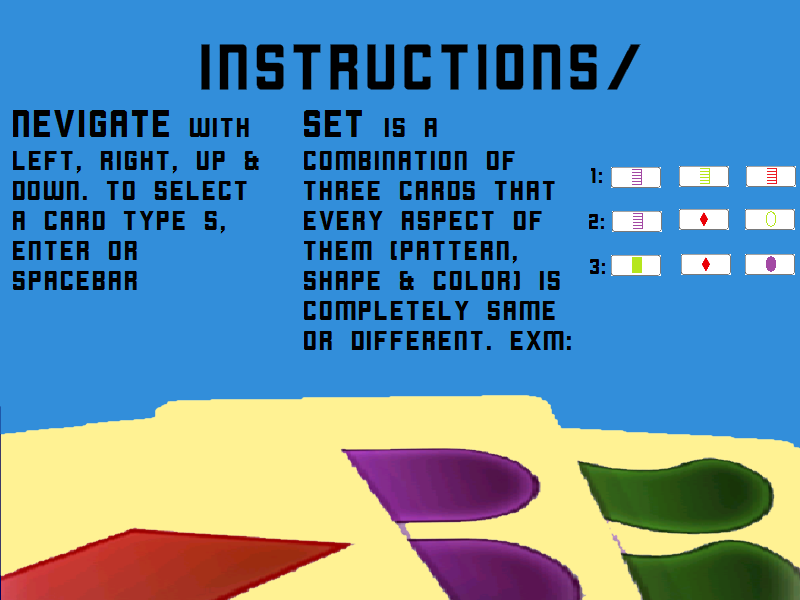




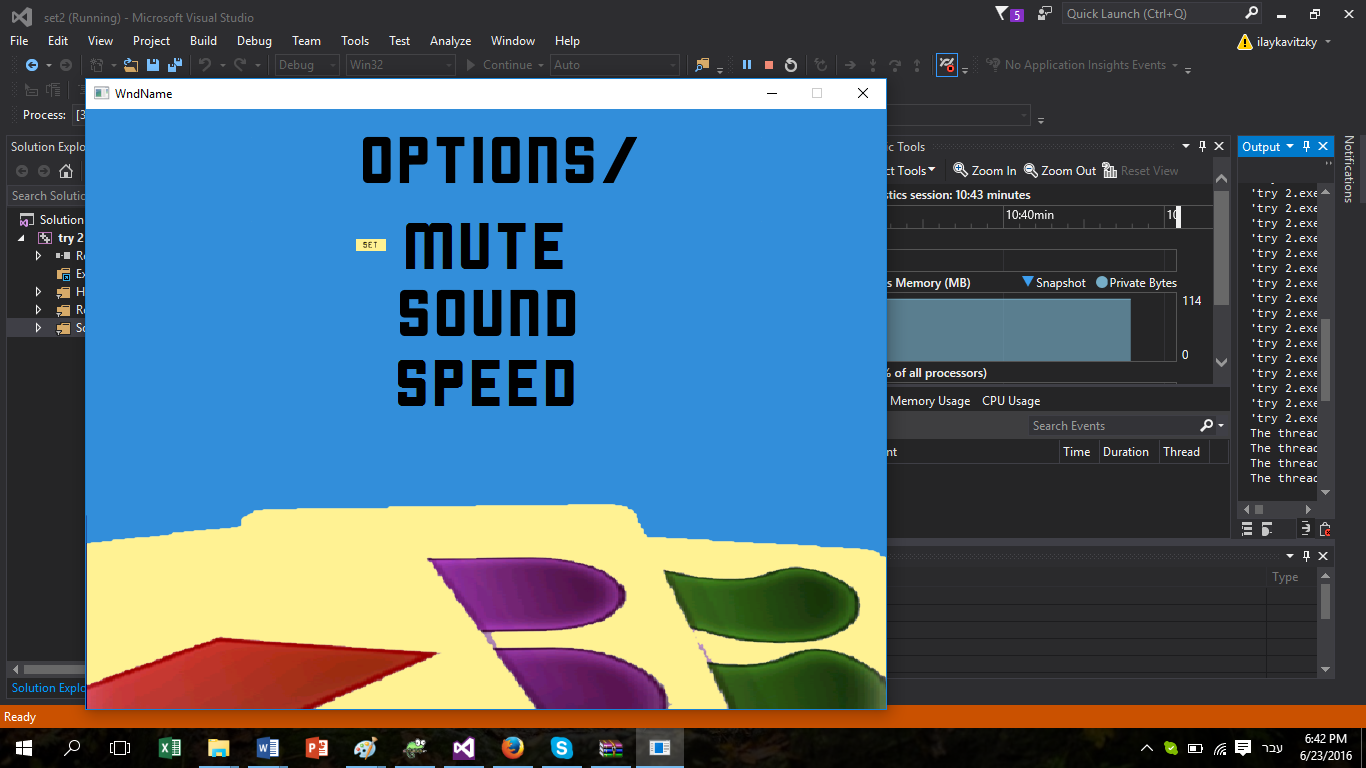
חלונית התפריט הראשי:

גם בו יש סמן, שנשלט ע"י החצים, וכאשר המשתמש לוחץ על enter, spacebar או s הוא עובר למה שהסמן מצביע עליו. למשל, למשחק עצמו, לדף הוראות, או לדף "הגדרות". כמו כן עם המשתמש יילחץ בדף זה על כפתור delete הוא ייסגור את התוכנית.

חלונית ההוראות:



חלונית ההגדרות:

בדף זה גם כן יש סמן, שכאשר המשתמש בוחר אופציה מסוימת, קוראת פעולה שונה- למשל: mute עושה שלא תהיה מוזיקה, sound גורם לכך שתהיה מוזיקה, ו speed גורם לכך שבמשחק עצמו הסמן יזוז במהירות יותר גדולה \ קטנה.

חלונית המשחק:

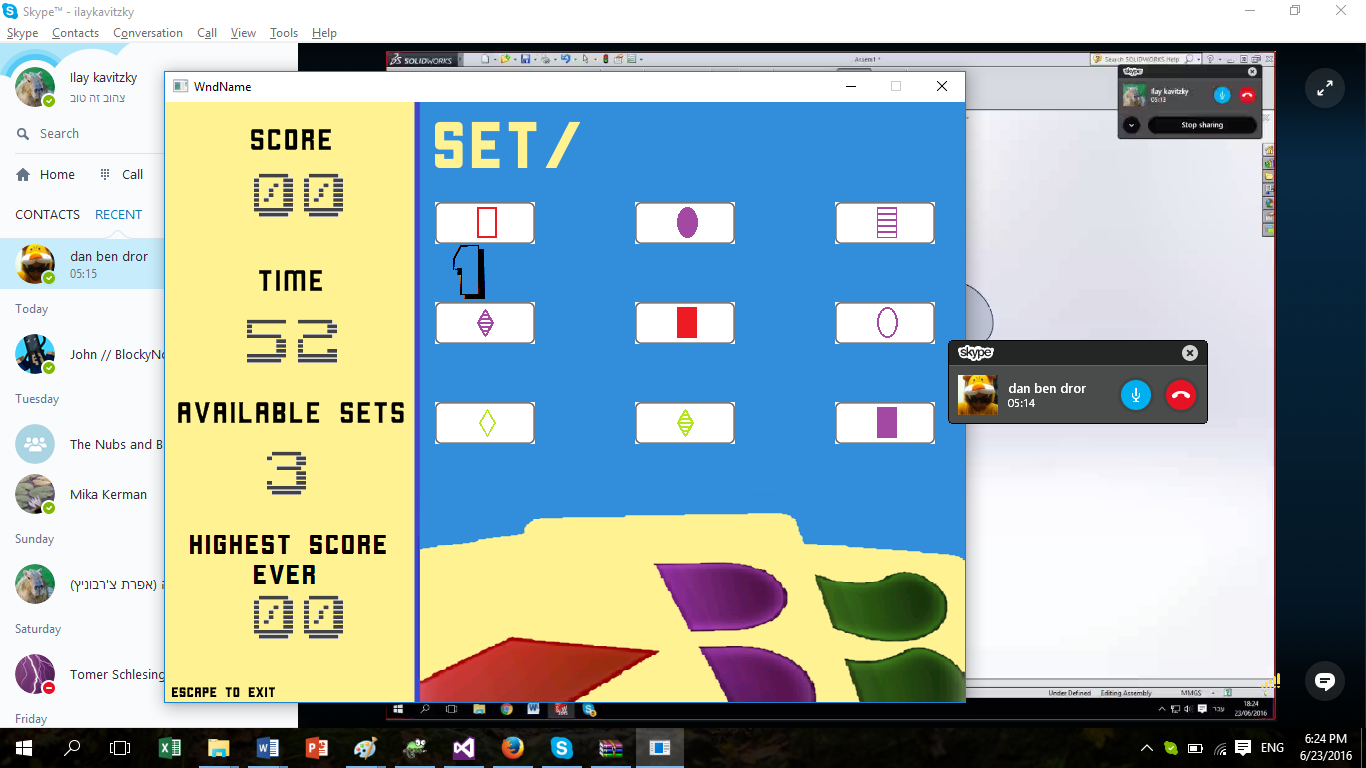
ניתן לראות את הקלפים מסודרים בצד ימין של המסך, וארבעה מדדים בצד הימני של המסך:

הניקוד שצברת עד עכשיו במשחק שלך, סט נכון מעלה 5 נקודות וטעות מורידה נקודה.

הזמן שנשאר עד סוף המשחק (מתחילים ב60 שניות).

מספר הסטים האופציונליים שניתן למצוא על המגרש.

שיא הניקוד שנצבר בכל המשחקים מפתיחת התוכנית.



על אסמבלי

אסמבלי היא שפת תכנות, הקרובה ביותר לשפת מכונה. שפה זו מורכבת מפקודות שהמעבד יכול להבין בלי צורך בתרגום – זאת אומרת כאשר אנו כותבים פקודה באסמבלי ככל הנראה שהיא פקודה אחת בשפת מכונה.

ובכן, שונה משפות תכנות יותר "גבוהות" אשר בהן כל פקודה מתרגמת ליותר מפקודה אחת בשפת מכונה.

בפרויקטים שעשינו במסגרת בית-הספר השתמשתי באמסבלי של מייקרוסופט masm.

עד לתחילת שנות ה-70 של המאה הקודמת התכנות בשפת אסמבלי היה נפוץ ביותר, אולם כעבור מספר שנים כשהתכללו המהדרים (compiler) הכתיבה בשפת האסמבלי כבר לא נהייתה כל כך מהירה באופן יחסי.

כיום המשתמשים העיקריים שמשתמשים באסמבלי לצורך תכנות הם כותבי מערכות הפעלה אשר צריכים מהירות מקסימלית של תוכניתם, וכן וכתבי וירוסים ("האקרים").

הסבר על התוכנית

התוכנית שלי מורכבת מכמה סוגים של פרוצדורות, שאחראיות למספר תפקידים בתוכנית:

putAllCards

פרוצדורה ששמה קלף רנדומלי בכל מיקום במערך של קלפי המגרש.

addCards

פרוצדורה שכאשר השחקן מצליח לעשות סט מוסיפה במקומות המתאימים במערך קלפין רנדומליים חדשים.

SetChecker

פרוצדורה שלוקחת שלושה פרמטרים (DWORD) מתרגמת אותם בעזרת FromPointerToCard ל CARD ובודקת אם הקלפים הם סט.

FromPointerToCard

פרוצדורה שלוקחת פרמטר באורך dword ומכניסה לתוך סטראקט שקוראים לו card את הערכים שאליו מצביעים הפרמטרים.

fromSelecterToLocatiosn

פרוצדורה שנועדה לקצר את הקוד, קוראים לה מתוך ההצגה שבvkhandler, נועדה לקחת מיקום של הסמן ולתרגם אותו לx ו y.

SameSameChecker

פרוצדורה שבודקת אם יש קלף כפול על המגרש.

HighScore

פרוצדורה שמעדכנת את העקך של highscore- משתנה שמכיל את התוצאה הגבוהה ביותר עד כה במשחק.

HighScoreAnzeiger

פרוצדורה שמציגה את הניקוד הכי גבוה עד כה.

OptionsProc

פרוצדורה שמריצה את מאחורי הקלעים של דף ההגדרות.

MainMenuProc

פרוצדורה שמריצה את מאחורי הקלעים של דף תפריט הראשי.

fromCardsOnBoardtoBoard

פרוצדורה שמציגה את הקלפים על המגרש.

ScoreAnzeiger

פרוצדורה שמציגה את הניקוד הנוכחי.

checkBoardForSet

פרוצדורה שבודקת את כל האופציות האפשריות אם יש סט, ומכניסה את מספר הסטים על המסך למשתנה.

VKhandler

פרוצדורה שמריצה את מאחורי הקלעים המשחק עצמו, קולטת לחיצות ומריצה פרוצדורות בהתאם.

TimerStarter

פרוצדורה שמריצה את השעון של המחשב ושמה את הערך בstartTime.

RandomCard

פונקציה שצגרילה מספר בין 1 ל27 (כמספר הקלפים האופציונליים).

UhrAnzeiger

פרוצדורה שבודקת ומעדכנת את הערך זמן שצריך להציג, וכן מציגה אותו.

הסבר על הלמידה

אני חושב שתהליך הלמידה האישי שלי היה דומה מאוד למין "קפיצה למים" לתוך הנושא. מה שעשיתי היה שבחודשיים לפני מועד הגשת העבודה התחלתי לכתוב קוד. כמו כן, מכיוון שלא היה לי ניסיון בכלל בכתיבת קוד אז בתומי כתבתי את מרבית הקוד "בפעם אחת"- וכמובן שבגלל טעות זו נתקעתי שלושה שבועות על debugging. ובכן כאשר סיימתי עם הdebug המשכתי לעשות בשיטה של צעדים קטנים, ולא קוד גדול בבת אחת. כמו כן, ניתן לראות שבמהלך התכנות שלי למדתי תוך כדי יותר ויותר על אסמבלי.

הסבר על פונקציות נבחרות

בסעיף זה אסביר על פרוצדורות שלדעתי הן הכי מורכבות ושהן בסיס הפרויקט שלי.

**פרוצדורת SetChecker:**

SetChecker PROC, pointer1:DWORD, pointer2:DWORD, pointer3:DWORD;TODO MORE ELEGANT

pusha

invoke FromPointerToCard, pointer1, 1

invoke FromPointerToCard, pointer2, 2

invoke FromPointerToCard, pointer3, 3

mov pixelCounter, 0

sameShapes:

X mov eax, [cur1.shape] \ cmp eax, [cur2.shape]

je S2

jmp differentShapes

S2:

X mov eax, [cur2.shape] \ cmp eax, [cur3.shape]

je sameColors

jmp differentShapes

differentShapes:

X mov eax, [cur1.shape] \ cmp eax, [cur2.shape]

jne dS2

jmp DIFFERENT

dS2:

X mov eax, [cur2.shape] \ cmp eax, [cur3.shape]

jne dS3

jmp DIFFERENT

dS3:

X mov eax, [cur1.shape] \ cmp eax, [cur3.shape]

jne sameColors

jmp DIFFERENT

sameColors:

X mov eax, 1 \ mov SameSame, eax

X mov eax, [cur1.color] \ cmp eax, [cur2.color]

je C2

jmp differentColors

C2:

X mov eax, [cur2.color] \ cmp eax, [cur3.color]

je samePattern

jmp differentColors

differentColors:

X mov eax, [cur1.color] \ cmp eax,[cur2.color]

jne dc2

jmp DIFFERENT

dc2:

X mov eax, [cur2.color] \ cmp eax, [cur3.color]

jne dc3

jmp DIFFERENT

dc3:

X mov eax, [cur1.color] \ cmp eax, [cur3.color]

jne samePattern

jmp DIFFERENT

samePattern:

;X mov eax, SameSame \ cmp eax, 1 \ je SameFinalLABEL

X mov eax, [cur1.pattern] \ cmp eax, [cur2.pattern]

je P2

jmp differentPattern

P2:

X mov eax, [cur2.pattern] \ cmp eax, [cur3.pattern]

je SAME

jmp differentPattern

differentPattern:

X mov eax, [cur1.pattern] \ cmp eax, [cur2.pattern]

jne dp2

jmp DIFFERENT

dp2:

X mov eax, [cur2.pattern] \ cmp eax, [cur3.pattern]

jne dp3

jmp DIFFERENT

dp3:

X mov eax, [cur1.pattern] \ cmp eax, [cur3.pattern]

jne SAME

jmp DIFFERENT

SAME:

X mov eax, 2 \ mov ThereIsSet, eax

jmp enddddd

;SameFinalLABEL:

;mov esi, 1

;mov SameFinal, esi

;jmp enddddd

DIFFERENT:

X mov eax, 1 \ mov ThereIsSet, eax

enddddd:

popa

ret

SetChecker ENDP

מה שבעצם ניתן לראות בפרוצדורה זו לקיחה מתוך struct שהגדרתי. לstruct זה קוראים card וכאן הגדרתי אותו (בdata):

CARD struct

color DWORD ?

pattern DWORD ?

shape DWORD ?

CARD ends

**פרוצדורת checkBoardForSet:**

רציתי להדגיש את פרוצדורה זו כי היא למעשה שלוש לולאות for באסמבלי, וכאשר היא מוצאת set היא מגדילה את הערך של הסטים SetsOnBoard האופציונליים על המגרש.

checkBoardForSet PROC

pusha

mov eax, 1

mov ThereIsSet, eax

; 0 1 2 3 4 5 6 7

replay:

mov ecx, 0

mov ebx, 0

mov edx, 0

mov eax, 0

MOV SetsOnBoard, EAX

first:

cmp ecx, 5

je endcbf

resetSecond:

mov ebx, ecx

second:

inc ebx

mov edx, ebx

cmp ebx, 6

jg firstAgain

third:

inc edx

cmp edx, 8

je second

loopContent:

mov eax, offset CardsOnBoard

invoke SetChecker, [eax+ecx\*4], [eax+ebx\*4], [eax+edx\*4]

cmp ThereIsSet, 2

je incSetsOnBoard

cntfrominc:

jmp third

firstAgain:

inc ecx

jmp first

incSetsOnBoard:

inc SetsOnBoard

jmp cntfrominc

endcbf:

cmp SetsOnBoard, 0

je ThereIsSetIs1

mov eax, 2

mov ThereIsSet, eax

jmp endingcbf

ThereIsSetIs1:

mov eax, 1

mov ThereIsSet, eax

endingcbf:

popa

ret

checkBoardForSet ENDP

פרוצדורה שלישית **UhrAnziger**:

רציתי להדגיש את מנגנון זה מכיוון שהוא בודק למעשה אם השנייה השתנתה, כמו כן הוא בנוי לכל שנייה שלא תהיה כשנייה ראשונה (StartTime). לפני ההצגה עצמה אני מפצל את הערך של היחידות והעשרות כדי שלא אצטרך תמונות מ0-60 אלא רק מ0-9.

UhrAnzeiger PROC

pusha

invoke GetSystemTime ,addr STime

mov cx, StartTime

mov bx, Stime.wSecond

cmp bx, cx

je putTheTimer

jne setTimeDec

setTimeDec:

dec setTime

inc StartTime

mov dx, 60

cmp StartTime, dx

je resetStartTime

mov dx, StartTimeReal

cmp setTime, dx

je endofgamebecauseua

jmp putTheTimer

resetStartTime:

mov dx, 0

mov StartTime, dx

jmp putTheTimer

endofgamebecauseua:

mov esi, 1

mov UAENDINGP, esi

jmp endUA

putTheTimer:

;for the divide:

mov ax, setTime

ove dx, 0

mov ebx, 10

div ebx

;edx – modulu

;eax – division

mov divisionAns, eax

mov moduluAns, edx

cmp setTime, 0

je endingParameter

הצגה וסוף פרוצדורה...

הקוד

.486

.model flat, stdcall

option casemap :none

putAllCards proto

addCards proto

SetChecker proto pointer1:DWORD, pointer2:DWORD, pointer3:DWORD

FromPointerToCard proto, mainpointer:DWORD, numberCurrent:DWORD

fromSelecterToLocatiosn proto, numberOfSelected:DWORD, selecterFSTL:DWORD

SameSameChecker proto

HighScore proto

HighScoreAnzeiger proto

OptionsProc proto

MainMenuProc proto

fromCardsOnBoardtoBoard proto

ScoreAnzeiger proto

checkBoardForSet proto

VKhandler proto

TimerStarter proto

RandomCard proto

UhrAnzeiger proto

include \masm32\include\windows.inc

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

include \masm32\include\msvcrt.inc

;INCLUDE Irvine32.inc

includelib msvcrt.lib

include drd.inc

includelib drd.lib

;sound

includelib \masm32\lib\winmm.lib

include \masm32\include\winmm.inc

include \masm32\include\windows.inc

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

include \masm32\include\gdi32.inc

include \masm32\include\Advapi32.inc

;include \masm32\include\masm32rt.inc

include \masm32\include\winmm.inc

include \masm32\include\comctl32.inc

;include \masm32\include\commctrl.inc

includelib \masm32\lib\winmm.lib

include \masm32\include\dialogs.inc

include \masm32\macros\macros.asm

includelib \masm32\lib\gdi32.lib

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\Comctl32.lib

includelib \masm32\lib\comdlg32.lib

includelib \masm32\lib\shell32.lib

includelib \masm32\lib\oleaut32.lib

includelib \masm32\lib\ole32.lib

includelib \masm32\lib\msvcrt.lib

include \masm32\include\msvcrt.inc

include \masm32\include\Ws2\_32.inc

includelib \masm32\lib\Ws2\_32.lib

include \masm32\include\ntoskrnl.inc

includelib \masm32\lib\ntoskrnl.lib

.data

CARD struct

color DWORD ?

pattern DWORD ?

shape DWORD ?

CARD ends

TblueFull BYTE "TblueFull.bmp",0

TblueFull\_Img Img <0,0,0,0>

CARD1 CARD <1,1,1>

TblueEmpty BYTE "TblueEmpty.bmp",0

TblueEmpty\_Img Img <0,0,0,0>

CARD2 CARD <1,2,1>

TblueStrips BYTE "TblueStrips.bmp",0

TblueStrips\_Img Img <0,0,0,0>

CARD3 CARD <1,3,1>

TredFull BYTE "TredFull.bmp",0

TredFull\_Img Img <0,0,0,0>

CARD4 CARD <2,1,1>

TredEmpty BYTE "TredEmpty.bmp",0

TredEmpty\_Img Img <0,0,0,0>

CARD5 CARD <2,2,1>

TredStrips BYTE "TredStrips.bmp",0

TredStrips\_Img Img <0,0,0,0>

CARD6 CARD <2,3,1>

TgreenFull BYTE "TgreenFull.bmp",0

TgreenFull\_Img Img <0,0,0,0>

CARD7 CARD <3,1,1>

TgreenEmpty BYTE "TgreenEmpty.bmp",0

TgreenEmpty\_Img Img <0,0,0,0>

CARD8 CARD <3,2,1>

TgreenStrips BYTE "TgreenStrips.bmp",0

TgreenStrips\_Img Img <0,0,0,0>

CARD9 CARD <3,3,1>

SblueFull BYTE "SblueFull.bmp",0

SblueFull\_Img Img <0,0,0,0>

CARD10 CARD <1,1,2>

SblueEmpty BYTE "SblueEmpty.bmp",0

SblueEmpty\_Img Img <0,0,0,0>

CARD11 CARD <1,2,2>

SblueStrips BYTE "SblueStrips.bmp",0

SblueStrips\_Img Img <0,0,0,0>

CARD12 CARD <1,3,2>

SredFull BYTE "SredFull.bmp",0

SredFull\_Img Img <0,0,0,0>

CARD13 CARD <2,1,2>

SredEmpty BYTE "SredEmpty.bmp",0

SredEmpty\_Img Img <0,0,0,0>

CARD14 CARD <2,2,2>

SredStrips BYTE "SredStrips.bmp",0

SredStrips\_Img Img <0,0,0,0>

CARD15 CARD <2,3,2>

SgreenFull BYTE "SgreenFull.bmp",0

SgreenFull\_Img Img <0,0,0,0>

CARD16 CARD <3,1,2>

SgreenEmpty BYTE "SgreenEmpty.bmp",0

SgreenEmpty\_Img Img <0,0,0,0>

CARD17 CARD <3,2,2>

SgreenStrips BYTE "SgreenStrips.bmp",0

SgreenStrips\_Img Img <0,0,0,0>

CARD18 CARD <3,3,2>

CblueFull BYTE "CblueFull.bmp",0

CblueFull\_Img Img <0,0,0,0>

CARD19 CARD <1,1,3>

CblueEmpty BYTE "CblueEmpty.bmp",0

CblueEmpty\_Img Img <0,0,0,0>

CARD20 CARD <1,2,3>

CblueStrips BYTE "CblueStrips.bmp",0

CblueStrips\_Img Img <0,0,0,0>

CARD21 CARD <1,3,3>

CredFull BYTE "CredFull.bmp",0

CredFull\_Img Img <0,0,0,0>

CARD22 CARD <2,1,3>

CredEmpty BYTE "CredEmpty.bmp",0

CredEmpty\_Img Img <0,0,0,0>

CARD23 CARD <2,2,3>

CredStrips BYTE "CredStrips.bmp",0

CredStrips\_Img Img <0,0,0,0>

CARD24 CARD <2,3,3>

CgreenFull BYTE "CgreenFull.bmp",0

CgreenFull\_Img Img <0,0,0,0>

CARD25 CARD <3,1,3>

CgreenEmpty BYTE "CgreenEmpty.bmp",0

CgreenEmpty\_Img Img <0,0,0,0>

CARD26 CARD <3,2,3>

CgreenStrips BYTE "CgreenStrips.bmp",0

CgreenStrips\_Img Img <0,0,0,0>

CARD27 CARD <3,3,3>

MMstartpage Byte "MMstartpage.bmp", 0

MMstartpageImg Img <0,0,0,0>

w\_width DWORD 800

w\_height DWORD 600

TurnsIfCondH DWORD 100

SoundSelected DWORD 0

speedSelected DWORD 0

muteSelected DWORD 0

TimerFirstXDist DWORD 80

TimerFirstYDist DWORD 215

TimerSecondXDist DWORD 130

TimerSecondYDist DWORD 215

TimerThirdXDist DWORD 450

TimerThirdYDist DWORD 2

STime SYSTEMTIME {} ; at: proc Random

TimerStop DWORD 0

timeNow dw 0

Score DWORD 0

WrongSets DWORD 0

RightSets DWORD 0

;positions:

pos1\_y DWORD 320

pos2\_y DWORD 555

pos3\_y DWORD 750

pos1\_x DWORD 80

pos2\_x DWORD 180

pos3\_x DWORD 280

firstPositionY DWORD 320

firstPositionX DWORD 80

secondPositionY DWORD 555

secondPositionX DWORD 180

thirdPositionY DWORD 750

thirdPositionX DWORD 280

obj\_limit\_x DWORD w\_width

obj\_limit\_y DWORD w\_height

;INTEGERSSSS

SPEED DWORD 100

i DWORD 0

randomcardnumber DWORD 0

pixelCounter DWORD 0

select3 DWORD 0

turn DWORD 0

HighScoreP DWORD 0

setTime dw 60

StartTime dw 0

StartTimeReal dw 0

selceter DWORD 0

ThereIsSet DWORD 0

UAENDINGP DWORD 0

SameSame DWORD 0

black DWORD 0

duplicates DWORD 0

SameFinal DWORD 0

imgsSize DWORD 0

imgsSize1 DWORD 0

imgsSize2 DWORD 0

mainMenuSelecter DWORD 0

MainMenuCounter DWORD 0

OptionsSelecter DWORD 0

OptionsCounter DWORD 0

imgsSize3 DWORD 0

ifSound DWORD 0

CondH DWORD 0

divisionAns DWORD 0

hundredParameter DWORD 0

SetsOnBoard DWORD 0

hundredCounter DWORD 0

escapeCondition DWORD 0

moduluAns DWORD 0

StartSelected DWORD 0

InstructionsSelected DWORD 0

OptionsSelected DWORD 0

;AudioFiles:

Menu BYTE "StartPage.bmp",0

MenuImg Img <0,0,0,0>

pngSplash2 BYTE "splash2.bmp",0

Splash2Img Img <0,0,0,0>

pngSplash3 BYTE "splash3.bmp",0

Splash3Img Img <0,0,0,0>

;Nummers fotos:

ZeroC BYTE "ZeroC.bmp",0

ZeroCIMG Img <0,0,0,0>

OneC BYTE "OneC.bmp",0

OneCIMG Img <0,0,0,0>

TwoC BYTE "TwoC.bmp",0

TwoCIMG Img <0,0,0,0>

ThreeC BYTE "ThreeC.bmp",0

ThreeCIMG Img <0,0,0,0>

FourC BYTE "FourC.bmp",0

FourCIMG Img <0,0,0,0>

FiveC BYTE "FiveC.bmp",0

FiveCIMG Img <0,0,0,0>

SixC BYTE "SixC.bmp",0

SixCIMG Img <0,0,0,0>

SevenC BYTE "SevenC.bmp",0

SevenCIMG Img <0,0,0,0>

EightC BYTE "EightC.bmp",0

EightCIMG Img <0,0,0,0>

NineC BYTE "NineC.bmp",0

NineCIMG Img <0,0,0,0>

;Selecters fotos:

selectedSquere1 BYTE "selectedSquere1.bmp", 0

selectedSquere1Img Img <0,0,0,0>

selectedSquere2 BYTE "selectedSquere2.bmp", 0

selectedSquere2Img Img <0,0,0,0>

selectedSquere3 BYTE "selectedSquere3.bmp", 0

selectedSquere3Img Img <0,0,0,0>

MMselecter BYTE "MMselecter.bmp", 0

MMselecterImg Img <0,0,0,0>

Instructions BYTE "Instructions.bmp", 0

InstructionsImg Img <0,0,0,0>

Options BYTE "Options.bmp", 0

OptionsImg Img <0,0,0,0>

;---AUDIO:----

Soundfile db "jumper.wav",0

BuzzerSoundfile db "Buzzer.wav",0

HaxSoundfile db "Hax.wav",0

SoundfileMunch db "Munch.wav", 0

;---AUDIO-END---

line DWORD 0

colomn DWORD 0

score DWORD 0

select1 DWORD 0

ScoreFirstXDist DWORD 87

ScoreFirstYDist DWORD 69

ScoreSecondXDist DWORD 137

ScoreSecondYDist DWORD 69

HScoreFirstXDist DWORD 544

HScoreFirstYDist DWORD 290

HScoreSecondXDist DWORD 594

HScoreSecondYDist DWORD 290

selecterProc DWORD 0

wrongSets DWORD 0

select2 DWORD 0

Ending DWORD 0

selecter DWORD 0

WHITE DWORD 77777777

SecondPositionY DWORD 0

SecondPositionX DWORD 0

rightSets DWORD 0

PositionX DWORD 0

PositionY DWORD 0

check DWORD 0

selecter1 DWORD 0

equelParameter DWORD 0

selecter2 DWORD 0

selecter3 DWORD 0

CardsOnBoard DWORD 9 dup(0)

SelectedCards DWORD 3 dup(0)

cur1 CARD <0, 0, 0>

cur2 CARD <0, 0, 0>

cur3 CARD <0, 0, 0>

border\_stop\_x DWORD 0

turn2 DWORD 0

.code

X macro args:VARARG

asm\_txt TEXTEQU <>

FORC char,<&args>

IFDIF <&char>,<!\>

asm\_txt CATSTR asm\_txt,<&char>

ELSE

asm\_txt

asm\_txt TEXTEQU <>

ENDIF

ENDM

asm\_txt

endm

FromPointerToCard Proc, mainpointer:DWORD, numberCurrent:DWORD

pusha

X mov eax, numberCurrent \ cmp eax, 1 \ je current1

X mov eax, numberCurrent \ cmp eax, 2 \ je current2

X mov eax, numberCurrent \ cmp eax, 3 \ je current3

current1:

X mov ebx, offset cur1.color

jmp cntFPTC

current2:

X mov ebx, offset cur2.color

jmp cntFPTC

current3:

X mov ebx, offset cur3.color

jmp cntFPTC

cntFPTC:

cmp mainpointer, 1

je F1

cmp mainpointer, 2

je F2

cmp mainpointer, 3

je F3

cmp mainpointer, 4

je F4

cmp mainpointer, 5

je F5

cmp mainpointer, 6

je F6

cmp mainpointer, 7

je F7

cmp mainpointer, 8

je F8

cmp mainpointer, 9

je F9

cmp mainpointer, 10

je F10

cmp mainpointer, 11

je F11

cmp mainpointer, 12

je F12

cmp mainpointer, 13

je F13

cmp mainpointer, 15

je F15

cmp mainpointer, 14

je F14

cmp mainpointer, 16

je F16

cmp mainpointer, 17

je F17

cmp mainpointer, 18

je F18

cmp mainpointer, 19

je F19

cmp mainpointer, 20

je F20

cmp mainpointer, 21

je F21

cmp mainpointer, 22

je F22

cmp mainpointer, 23

je F23

cmp mainpointer, 24

je F24

cmp mainpointer, 25

je F25

cmp mainpointer, 26

je F26

jmp finalEndFPTC

F1:

X mov edx, [CARD1.shape] \ mov [ebx+8], edx

X mov edx, [CARD1.color] \ mov [ebx], edx

X mov edx, [CARD1.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F2:

X mov edx, [CARD2.shape] \ mov [ebx+8], edx

X mov edx, [CARD2.color] \ mov [ebx], edx

X mov edx, [CARD2.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F3:

X mov edx, [CARD3.shape] \ mov [ebx+8], edx

X mov edx, [CARD3.color] \ mov [ebx], edx

X mov edx, [CARD3.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F4:

X mov edx, [CARD4.shape] \ mov [ebx+8], edx

X mov edx, [CARD4.color] \ mov [ebx], edx

X mov edx, [CARD4.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F5:

X mov edx, [CARD5.shape] \ mov [ebx+8], edx

X mov edx, [CARD5.color] \ mov [ebx], edx

X mov edx, [CARD5.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F6:

X mov edx, [CARD6.shape] \ mov [ebx+8], edx

X mov edx, [CARD6.color] \ mov [ebx], edx

X mov edx, [CARD6.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F7:

X mov edx, [CARD7.shape] \ mov [ebx+8], edx

X mov edx, [CARD7.color] \ mov [ebx], edx

X mov edx, [CARD7.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F8:

X mov edx, [CARD8.shape] \ mov [ebx+8], edx

X mov edx, [CARD8.color] \ mov [ebx], edx

X mov edx, [CARD8.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F9:

X mov edx, [CARD9.shape] \ mov [ebx+8], edx

X mov edx, [CARD9.color] \ mov [ebx], edx

X mov edx, [CARD9.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F10:

X mov edx, [CARD10.shape] \ mov [ebx+8], edx

X mov edx, [CARD10.color] \ mov [ebx], edx

X mov edx, [CARD10.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F11:

X mov edx, [CARD11.shape] \ mov [ebx+8], edx

X mov edx, [CARD11.color] \ mov [ebx], edx

X mov edx, [CARD11.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F12:

X mov edx, [CARD12.shape] \ mov [ebx+8], edx

X mov edx, [CARD12.color] \ mov [ebx], edx

X mov edx, [CARD12.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F13:

X mov edx, [CARD13.shape] \ mov [ebx+8], edx

X mov edx, [CARD13.color] \ mov [ebx], edx

X mov edx, [CARD13.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F14:

X mov edx, [CARD14.shape] \ mov [ebx+8], edx

X mov edx, [CARD14.color] \ mov [ebx], edx

X mov edx, [CARD14.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F15:

X mov edx, [CARD15.shape] \ mov [ebx+8], edx

X mov edx, [CARD15.color] \ mov [ebx], edx

X mov edx, [CARD15.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F16:

X mov edx, [CARD16.shape] \ mov [ebx+8], edx

X mov edx, [CARD16.color] \ mov [ebx], edx

X mov edx, [CARD16.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F17:

X mov edx, [CARD17.shape] \ mov [ebx+8], edx

X mov edx, [CARD17.color] \ mov [ebx], edx

X mov edx, [CARD17.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F18:

X mov edx, [CARD18.shape] \ mov [ebx+8], edx

X mov edx, [CARD18.color] \ mov [ebx], edx

X mov edx, [CARD18.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F19:

X mov edx, [CARD19.shape] \ mov [ebx+8], edx

X mov edx, [CARD19.color] \ mov [ebx], edx

X mov edx, [CARD19.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F20:

X mov edx, [CARD20.shape] \ mov [ebx+8], edx

X mov edx, [CARD20.color] \ mov [ebx], edx

X mov edx, [CARD20.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F21:

X mov edx, [CARD21.shape] \ mov [ebx+8], edx

X mov edx, [CARD21.color] \ mov [ebx], edx

X mov edx, [CARD21.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F22:

X mov edx, [CARD22.shape] \ mov [ebx+8], edx

X mov edx, [CARD22.color] \ mov [ebx], edx

X mov edx, [CARD22.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F23:

X mov edx, [CARD23.shape] \ mov [ebx+8], edx

X mov edx, [CARD23.color] \ mov [ebx], edx

X mov edx, [CARD23.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F24:

X mov edx, [CARD24.shape] \ mov [ebx+8], edx

X mov edx, [CARD24.color] \ mov [ebx], edx

X mov edx, [CARD24.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F25:

X mov edx, [CARD25.shape] \ mov [ebx+8], edx

X mov edx, [CARD25.color] \ mov [ebx], edx

X mov edx, [CARD25.pattern] \ mov [ebx+4], edx

jmp finalEndFPTC

F26:

X mov edx, [CARD26.shape] \ mov [ebx+8], edx

X mov edx, [CARD26.color] \ mov [ebx], edx

X mov edx, [CARD26.pattern] \ mov [ebx+4], edx

finalEndFPTC:

popa

ret

FromPointerToCard Endp

SameSameChecker PROC

PUSHA

mov ecx, 0

mov ebx, 0

startdl:

mov ecx, 0

jmp int2

int1:

inc ecx

cmp ecx, 8

je endSSC

resetint2:

mov ebx, ecx

int2:

inc ebx

cmp ebx, 9

je int1

mov edx, [CardsOnBoard + 4 \* ecx]

mov eax, [CardsOnBoard + 4 \* ebx]

cmp edx, eax

je ThereIsDup

jmp int2

ThereIsDup:

mov edx, 1

mov duplicates, edx

jmp endSSC

endSSC:

POPA

RET

SameSameChecker ENDP

MainMenuProc PROC

pusha

inc MainMenuCounter

X mov eax, MainMenuCounter\ cmp eax, 100

je mainMenuStarter

jmp mainMenuanzeiger

mainMenuStarter:

X mov eax, 0\ mov MainMenuCounter, eax

X invoke GetAsyncKeyState, VK\_UP \ cmp eax, 0 \ jne upMainMenu

X invoke GetAsyncKeyState, VK\_DOWN \ cmp eax, 0 \ jne downMainMenu

X invoke GetAsyncKeyState, VK\_RETURN \ cmp eax, 0 \ jne selectt

X invoke GetAsyncKeyState, VK\_SPACE \ cmp eax, 0 \ jne selectt

X invoke GetAsyncKeyState, 53h \ cmp eax, 0 \ jne selectt

jmp mainMenuanzeiger

upMainMenu:

mov eax, mainMenuSelecter

cmp eax, 0

je resetUpmm

dec eax

mov mainMenuSelecter, eax

jmp mainMenuanzeiger

resetUpmm:

mov eax, 2

mov mainMenuSelecter, eax

jmp mainMenuanzeiger

downMainMenu:

mov eax, mainMenuSelecter

cmp eax, 2

je resetdownmm

inc eax

mov mainMenuSelecter, eax

jmp mainMenuanzeiger

resetdownmm:

mov eax, 0

mov mainMenuSelecter, eax

jmp mainMenuanzeiger

selectt:

mov esi, 1

X mov eax, 0 \ cmp eax, mainMenuSelecter \ je mmStart

X mov eax, 1 \ cmp eax, mainMenuSelecter \ je mmInstructions

X mov eax, 2 \ cmp eax, mainMenuSelecter \ je mmOptions

mmStart:

mov StartSelected, esi

jmp mainMenuanzeiger

mmInstructions:

mov InstructionsSelected, esi

jmp mainMenuanzeiger

mmOptions:

mov OptionsSelected, esi

jmp mainMenuanzeiger

mainMenuanzeiger:

invoke drd\_imageDraw, offset MMstartpageImg, 0, 0

X mov eax, 0 \ cmp mainMenuSelecter, eax \ je showSelecterEins

X mov eax, 1 \ cmp mainMenuSelecter, eax \ je showSelecterZwei

X mov eax, 2 \ cmp mainMenuSelecter, eax \ je showSelecterDrei

showSelecterEins:

invoke drd\_imageDraw, offset MMselecterImg, 166, 96

jmp endingmm

showSelecterZwei:

invoke drd\_imageDraw, offset MMselecterImg, 100, 164

jmp endingmm

showSelecterDrei:

invoke drd\_imageDraw, offset MMselecterImg, 100, 240

jmp endingmm

endingmm:

popa

ret

MainMenuProc ENDP

OptionsProc PROC

pusha

inc OptionsCounter

X mov eax, OptionsCounter\ cmp eax, 100

je optoinsStarter

jmp Optionsanzeiger

optoinsStarter:

X mov eax, 0\ mov OptionsCounter, eax

X invoke GetAsyncKeyState, VK\_UP \ cmp eax, 0 \ jne upMainMenu

X invoke GetAsyncKeyState, VK\_DOWN \ cmp eax, 0 \ jne downMainMenu

X invoke GetAsyncKeyState, VK\_RETURN \ cmp eax, 0 \ jne selecttMainMenu

X invoke GetAsyncKeyState, VK\_SPACE \ cmp eax, 0 \ jne selecttMainMenu

X invoke GetAsyncKeyState, 53h \ cmp eax, 0 \ jne selecttMainMenu

jmp Optionsanzeiger

upMainMenu:

mov eax, OptionsSelecter

cmp eax, 0

je resetUpmm

dec eax

mov OptionsSelecter, eax

jmp Optionsanzeiger

resetUpmm:

mov eax, 2

mov OptionsSelecter, eax

jmp Optionsanzeiger

downMainMenu:

mov eax, OptionsSelecter

cmp eax, 2

je resetdownmm

inc eax

mov OptionsSelecter, eax

jmp Optionsanzeiger

resetdownmm:

mov eax, 0

mov OptionsSelecter, eax

jmp Optionsanzeiger

selecttMainMenu:

mov esi, 1

X mov eax, 0 \ cmp eax, OptionsSelecter \ je Osound

X mov eax, 1 \ cmp eax, OptionsSelecter \ je Omute

X mov eax, 2 \ cmp eax, OptionsSelecter \ je Ospeed

Osound:

cmp SoundSelected, esi

je mov0sound

mov SoundSelected, esi

jmp Optionsanzeiger

mov0sound:

mov esi, 0

mov SoundSelected, esi

jmp Optionsanzeiger

Omute:

cmp muteSelected, esi

je mov0mute

mov muteSelected, esi

jmp Optionsanzeiger

mov0mute:

mov esi, 0

mov muteSelected, esi

jmp Optionsanzeiger

Ospeed:

cmp speedSelected, esi

je mov0speed

mov speedSelected, esi

jmp Optionsanzeiger

mov0speed:

mov esi, 0

mov speedSelected, esi

jmp Optionsanzeiger

Optionsanzeiger:

invoke drd\_imageDraw, offset OptionsImg, 0, 0

X mov eax, 0 \ cmp OptionsSelecter, eax \ je showSelecterEins2

X mov eax, 1 \ cmp OptionsSelecter, eax \ je showSelecterZwei2

X mov eax, 2 \ cmp OptionsSelecter, eax \ je showSelecterDrei2

showSelecterEins2:

invoke drd\_imageDraw, offset MMselecterImg, 270, 130

jmp endingmm2

showSelecterZwei2:

invoke drd\_imageDraw, offset MMselecterImg, 270, 197

jmp endingmm2

showSelecterDrei2:

invoke drd\_imageDraw, offset MMselecterImg, 270, 269

jmp endingmm2

endingmm2:

popa

ret

OptionsProc ENDP

SetChecker PROC, pointer1:DWORD, pointer2:DWORD, pointer3:DWORD;TODO MORE ELEGANT

pusha

invoke FromPointerToCard, pointer1, 1

invoke FromPointerToCard, pointer2, 2

invoke FromPointerToCard, pointer3, 3

mov pixelCounter, 0

sameShapes:

X mov eax, [cur1.shape] \ cmp eax, [cur2.shape]

je S2

jmp differentShapes

S2:

X mov eax, [cur2.shape] \ cmp eax, [cur3.shape]

je sameColors

jmp differentShapes

differentShapes:

X mov eax, [cur1.shape] \ cmp eax, [cur2.shape]

jne dS2

jmp DIFFERENT

dS2:

X mov eax, [cur2.shape] \ cmp eax, [cur3.shape]

jne dS3

jmp DIFFERENT

dS3:

X mov eax, [cur1.shape] \ cmp eax, [cur3.shape]

jne sameColors

jmp DIFFERENT

sameColors:

X mov eax, 1 \ mov SameSame, eax

X mov eax, [cur1.color] \ cmp eax, [cur2.color]

je C2

jmp differentColors

C2:

X mov eax, [cur2.color] \ cmp eax, [cur3.color]

je samePattern

jmp differentColors

differentColors:

X mov eax, [cur1.color] \ cmp eax,[cur2.color]

jne dc2

jmp DIFFERENT

dc2:

X mov eax, [cur2.color] \ cmp eax, [cur3.color]

jne dc3

jmp DIFFERENT

dc3:

X mov eax, [cur1.color] \ cmp eax, [cur3.color]

jne samePattern

jmp DIFFERENT

samePattern:

;X mov eax, SameSame \ cmp eax, 1 \ je SameFinalLABEL

X mov eax, [cur1.pattern] \ cmp eax, [cur2.pattern]

je P2

jmp differentPattern

P2:

X mov eax, [cur2.pattern] \ cmp eax, [cur3.pattern]

je SAME

jmp differentPattern

differentPattern:

X mov eax, [cur1.pattern] \ cmp eax, [cur2.pattern]

jne dp2

jmp DIFFERENT

dp2:

X mov eax, [cur2.pattern] \ cmp eax, [cur3.pattern]

jne dp3

jmp DIFFERENT

dp3:

X mov eax, [cur1.pattern] \ cmp eax, [cur3.pattern]

jne SAME

jmp DIFFERENT

SAME:

X mov eax, 2 \ mov ThereIsSet, eax

jmp enddddd

;SameFinalLABEL:

;mov esi, 1

;mov SameFinal, esi

;jmp enddddd

DIFFERENT:

X mov eax, 1 \ mov ThereIsSet, eax

enddddd:

popa

ret

SetChecker ENDP

HighScore PROC

pusha

mov eax, Score

cmp eax, HighScoreP

jg incHS

jmp endHS

incHS:

mov HighScoreP, eax

endHS:

popa

ret

HighScore ENDP

putAllCards PROC;#TODO

pusha

startPAC:

mov ecx, 0

mov ebx, 0

mov edx, 0

X mov ecx, 0 \ mov duplicates, ecx

putAllCardsLabel1:

invoke RandomCard

mov edx, randomcardnumber

mov [CardsOnBoard + 4 \* ecx], edx

inc ecx

cmp ecx, 10

je NowLetsCheck

jmp putAllCardsLabel1

NowLetsCheck:

invoke SameSameChecker

X mov esi, 1 \ cmp duplicates, esi

je startPAC

invoke checkBoardForSet

mov eax, 1

cmp SameFinal, eax

je startPAC

mov eax, 1

cmp ThereIsSet, eax

je startPAC

mov eax, 2

cmp ThereIsSet, eax

je endpac

endpac:

popa

ret

putAllCards ENDP

fromCardsOnBoardtoBoard PROC ;#TODO

pusha

mov ebx, 0

jmp mainFrom

mainmain:

inc ebx

cmp ebx, 9

je endoffrom

mainFrom:

cmp ebx, 0

je onePlaces

cmp ebx, 1

je twoPlaces

cmp ebx, 2

je threePlaces

cmp ebx, 3

je fourPlaces

cmp ebx, 4

je fivePlaces

cmp ebx, 5

je sixPlaces

cmp ebx, 6

je sevenPlaces

cmp ebx, 7

je eightPlaces

cmp ebx, 8

je ninePlaces

cmp ebx, 9

je endoffrom

;#todo - real parameters

onePlaces:

X mov ecx, 270 \ mov PositionX, ecx

X mov ecx, 100 \ mov PositionY, ecx

jmp cnt

twoPlaces:

X mov ecx, 470\ mov PositionX, ecx

X mov ecx, 100 \ mov PositionY, ecx

jmp cnt

threePlaces:

X mov ecx, 670 \ mov PositionX, ecx

X mov ecx, 100 \ mov PositionY, ecx

jmp cnt

fourPlaces:

X mov ecx, 270 \ mov PositionX, ecx

X mov ecx, 200 \ mov PositionY, ecx

jmp cnt

fivePlaces:

X mov ecx, 470\ mov PositionX, ecx

X mov ecx, 200\ mov PositionY, ecx

jmp cnt

sixPlaces:

X mov ecx, 670\ mov PositionX, ecx

X mov ecx, 200 \ mov PositionY, ecx

jmp cnt

sevenPlaces:

X mov ecx, 270 \ mov PositionX, ecx

X mov ecx, 300 \ mov PositionY, ecx

jmp cnt

eightPlaces:

X mov ecx, 470 \ mov PositionX, ecx

X mov ecx, 300\ mov PositionY, ecx

jmp cnt

ninePlaces:

X mov ecx, 670 \ mov PositionX, ecx

X mov ecx, 300 \ mov PositionY, ecx

jmp cnt

cnt:

mov eax, [CardsOnBoard+ebx\*4]

cmp eax, 1

je CardOne

cmp eax, 2

je CardTwo

cmp eax, 3

je CardThree

cmp eax, 4

je CardFour

cmp eax, 5

je CardFive

cmp eax, 6

je CardSix

cmp eax, 7

je CardSeven

cmp eax, 8

je CardEight

cmp eax, 9

je CardNine

cmp eax, 10

je CardTen

cmp eax, 11

je CardEleven

cmp eax, 12

je CardTwelve

cmp eax, 13

je CardThirteen

cmp eax, 14

je CardFourteen

cmp eax, 15

je CardFiveteen

cmp eax, 16

je CardSixteen

cmp eax, 17

je CardSeventeen

cmp eax, 18

je CardEighteen

cmp eax, 19

je CardNineteen

cmp eax, 20

je CardTwenty

cmp eax, 21

je CardTwentyone

cmp eax, 22

je CardTwentytwo

cmp eax, 23

je Cardtwentythree

cmp eax, 24

je CardTwentyfour

cmp eax, 25

je CardTwentyfive

;jmp endoffrom

CardOne:

invoke drd\_imageDraw, offset TblueFull\_Img, PositionX , PositionY

jmp mainmain

CardTwo:

invoke drd\_imageDraw, offset TblueEmpty\_Img, PositionX , PositionY

jmp mainmain

CardThree:

invoke drd\_imageDraw, offset TblueStrips\_Img, PositionX , PositionY

jmp mainmain

CardFour:

invoke drd\_imageDraw, offset TredFull\_Img, PositionX , PositionY

jmp mainmain

CardFive:

invoke drd\_imageDraw, offset TredEmpty\_Img, PositionX , PositionY

jmp mainmain

CardSix:

invoke drd\_imageDraw, offset TredStrips\_Img, PositionX , PositionY

jmp mainmain

CardSeven:

invoke drd\_imageDraw, offset TgreenFull\_Img, PositionX , PositionY

jmp mainmain

CardEight:

invoke drd\_imageDraw, offset TgreenEmpty\_Img, PositionX , PositionY

jmp mainmain

CardNine:

invoke drd\_imageDraw, offset TgreenStrips\_Img, PositionX , PositionY

jmp mainmain

CardTen:

invoke drd\_imageDraw, offset SblueFull\_Img, PositionX , PositionY

jmp mainmain

CardEleven:

invoke drd\_imageDraw, offset SblueEmpty\_Img, PositionX , PositionY

jmp mainmain

CardTwelve:

invoke drd\_imageDraw, offset SblueStrips\_Img, PositionX , PositionY

jmp mainmain

CardThirteen:

invoke drd\_imageDraw, offset SredFull\_Img, PositionX , PositionY

jmp mainmain

CardFourteen:

invoke drd\_imageDraw, offset SredEmpty\_Img, PositionX , PositionY

jmp mainmain

CardFiveteen:

invoke drd\_imageDraw, offset SredStrips\_Img, PositionX , PositionY

jmp mainmain

CardSixteen:

invoke drd\_imageDraw, offset SgreenFull\_Img, PositionX , PositionY

jmp mainmain

CardSeventeen:

invoke drd\_imageDraw, offset SgreenEmpty\_Img, PositionX , PositionY

jmp mainmain

CardEighteen:

invoke drd\_imageDraw, offset SgreenStrips\_Img, PositionX , PositionY

jmp mainmain

CardNineteen:

invoke drd\_imageDraw, offset CblueFull\_Img, PositionX , PositionY

jmp mainmain

CardTwenty:

invoke drd\_imageDraw, offset CblueEmpty\_Img, PositionX , PositionY

jmp mainmain

CardTwentyone:

invoke drd\_imageDraw, offset CblueStrips\_Img, PositionX , PositionY

jmp mainmain

CardTwentytwo:

invoke drd\_imageDraw, offset CredFull\_Img, PositionX , PositionY

jmp mainmain

Cardtwentythree:

invoke drd\_imageDraw, offset CredEmpty\_Img, PositionX , PositionY

jmp mainmain

CardTwentyfour:

invoke drd\_imageDraw, offset CredStrips\_Img, PositionX , PositionY

jmp mainmain

CardTwentyfive:

invoke drd\_imageDraw, offset CgreenFull\_Img, PositionX , PositionY

jmp mainmain

CardTwentySix:

invoke drd\_imageDraw, offset CgreenEmpty\_Img, PositionX , PositionY

jmp mainmain

CardTwentySeven:

invoke drd\_imageDraw, offset CgreenStrips\_Img, PositionX , PositionY

jmp mainmain

endoffrom:

popa

ret

fromCardsOnBoardtoBoard ENDP

checkBoardForSet PROC

pusha

mov eax, 1

mov ThereIsSet, eax

; 0 1 2 3 4 5 6 7

replay:

mov ecx, 0

mov ebx, 0

mov edx, 0

mov eax, 0

MOV SetsOnBoard, EAX

first:

cmp ecx, 5

je endcbf

resetSecond:

mov ebx, ecx

second:

inc ebx

mov edx, ebx

cmp ebx, 6

jg firstAgain

third:

inc edx

cmp edx, 8

je second

loopContent:

mov eax, offset CardsOnBoard

invoke SetChecker, [eax+ecx\*4], [eax+ebx\*4], [eax+edx\*4]

cmp ThereIsSet, 2

je incSetsOnBoard

cntfrominc:

jmp third

firstAgain:

inc ecx

jmp first

incSetsOnBoard:

inc SetsOnBoard

jmp cntfrominc

endcbf:

cmp SetsOnBoard, 0

je ThereIsSetIs1

mov eax, 2

mov ThereIsSet, eax

jmp endingcbf

ThereIsSetIs1:

mov eax, 1

mov ThereIsSet, eax

endingcbf:

popa

ret

checkBoardForSet ENDP

addCards PROC

pusha

startAC:

X mov esi, 0 \ mov duplicates, esi

invoke RandomCard

mov edx, selecter1

X mov ebx, randomcardnumber \ mov [CardsOnBoard+edx\*4], ebx

invoke RandomCard

mov edx, selecter2

X mov ebx, randomcardnumber \ mov [CardsOnBoard+edx\*4], ebx

invoke RandomCard

mov edx, selecter3

X mov ebx, randomcardnumber \ mov [CardsOnBoard+edx\*4], ebx

invoke SameSameChecker

X mov esi, 1 \ cmp duplicates, esi

je startAC

invoke checkBoardForSet

mov eax, 2

cmp ThereIsSet, eax

je endd

jmp startAC

endd:

popa

ret

addCards ENDP

fromSelecterToLocatiosn PROC, numberOfSelected:DWORD, selecterFSTL:DWORD

pusha

X mov eax ,selecterFSTL \ mov selecterProc, eax

cmp selecterProc, 0

je pixel0

cmp selecterProc, 1

je pixel1

cmp selecterProc, 2

je pixel2

cmp selecterProc, 3

je pixel3

cmp selecterProc, 4

je pixel4

cmp selecterProc, 5

je pixel5

cmp selecterProc, 6

je pixel6

cmp selecterProc, 7

je pixel7

cmp selecterProc, 8

je pixel8

pixel0:

X mov eax, 280 \ mov ecx, eaX

X mov eaX, 130 \ mov ebx, eaX

jmp pixelProc

pixel1:

X mov eaX, 490 \ mov ecx, eaX

X mov eaX, 130\ mov ebx, eaX

jmp pixelProc

pixel2:

X mov eaX, 670\ mov ecx, eaX

X mov eaX, 130 \ mov ebx, eaX

jmp pixelProc

pixel3:

X mov eaX, 280 \ mov ecx, eaX

X mov eaX, 230 \ mov ebx, eaX

jmp pixelProc

pixel4:

X mov eaX, 490\ mov ecx, eaX

X mov eaX, 230 \ mov ebx, eaX

jmp pixelProc

pixel5:

X mov eaX, 670\ mov ecx, eaX

X mov eaX, 230 \ mov ebx, eaX

jmp pixelProc

pixel6:

X mov eaX, 280 \ mov ecx, eaX

X mov eaX, 330 \ mov ebx, eaX

jmp pixelProc

pixel7:

X mov eaX, 490 \ mov ecx, eaX

X mov eaX, 330 \ mov ebx, eaX

jmp pixelProc

pixel8:

X mov eaX, 670 \ mov ecx, eaX

X mov eaX, 330 \ mov ebx, eaX

jmp pixelProc

pixelProc:

cmp numberOfSelected, 0

je firstPositions

cmp numberOfSelected, 1

je secondPositions

cmp numberOfSelected, 2

je thirdPositions

firstPositions:

mov firstPositionX, ecx

mov firstPositionY, ebx

mov selecter1, edx

jmp EndSelecter

secondPositions:

mov secondPositionX, ecx

mov secondPositionY, ebx

mov selecter2, edx

jmp EndSelecter

thirdPositions:

mov thirdPositionX, ecx

mov thirdPositionY, ebx

EndSelecter:

popa

ret

fromSelecterToLocatiosn ENDP

setOnBoardAnzeiger PROC

pusha

mov esi, SetsOnBoard

X cmp esi, 0 \ je zeroSets

X cmp esi, 1 \ je oneSets

X cmp esi, 2 \ je twoSets

X cmp esi, 3 \ je threeSets

X cmp esi, 4 \ je fourSets

X cmp esi, 5 \ je fiveSets

X cmp esi, 6 \ je sixSets

X cmp esi, 7 \ je sevenSets

X cmp esi, 8 \ je eightSets

X cmp esi, 9 \ je nineSets

zeroSets:

invoke drd\_imageDraw,offset ZeroCIMG, 100, 347

jmp endsoba

oneSets:

invoke drd\_imageDraw,offset OneCIMG, 100, 347

jmp endsoba

twoSets:

invoke drd\_imageDraw,offset TwoCIMG, 100, 347

jmp endsoba

threeSets:

invoke drd\_imageDraw,offset ThreeCIMG, 100, 347

jmp endsoba

fourSets:

invoke drd\_imageDraw,offset FourCIMG, 100, 347

jmp endsoba

fiveSets:

invoke drd\_imageDraw,offset FiveCIMG, 100, 347

jmp endsoba

sixSets:

invoke drd\_imageDraw,offset SixCIMG, 100, 347

jmp endsoba

sevenSets:

invoke drd\_imageDraw,offset SevenCIMG, 100, 347

jmp endsoba

eightSets:

invoke drd\_imageDraw,offset EightCIMG, 100, 347

jmp endsoba

nineSets:

invoke drd\_imageDraw,offset NineCIMG, 100, 347

jmp endsoba

endsoba:

popa

ret

setOnBoardAnzeiger ENDP

HighScoreAnzeiger PROC

pusha

mov eax, HighScoreP

mov edx, 0

mov ebx, 10

div ebx

;edx - modulu

;eax - division

mov moduluAns, edx

mov divisionAns, eax

X mov eax, divisionAns \ cmp eax, 0

je zero

X mov eax, divisionAns \ cmp eax, 1

je one

X mov eax, divisionAns \ cmp eax, 2

je two

X mov eax, divisionAns \ cmp eax, 3

je three

X mov eax, divisionAns \ cmp eax, 4

je four

X mov eax, divisionAns \ cmp eax, 5

je five

X mov eax, divisionAns \ cmp eax, 6

je six

X mov eax, divisionAns \ cmp eax, 7

je seven

X mov eax, divisionAns \ cmp eax, 8

je eight

X mov eax, divisionAns \ cmp eax, 9

je nine

secondUA:

X mov edx, moduluAns

cmp edx, 0

je zero2

cmp edx, 1

je one2

cmp edx, 2

je two2

cmp edx, 3

je three2

cmp edx, 4

je four2

cmp edx, 5

je five2

cmp edx, 6

je six2

cmp edx, 7

je seven2

cmp edx, 8

je eight2

cmp edx, 9

je nine2

zero:

invoke drd\_imageDraw,offset ZeroCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

one:

invoke drd\_imageDraw,offset OneCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

two:

invoke drd\_imageDraw,offset TwoCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

three:

invoke drd\_imageDraw,offset ThreeCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

four:

invoke drd\_imageDraw,offset FourCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

five:

invoke drd\_imageDraw,offset FiveCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

six:

invoke drd\_imageDraw,offset SixCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

seven:

invoke drd\_imageDraw,offset SevenCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

eight:

invoke drd\_imageDraw,offset EightCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

nine:

invoke drd\_imageDraw,offset NineCIMG, HScoreFirstXDist, HScoreFirstYDist

jmp secondUA

zero2:

invoke drd\_imageDraw,offset ZeroCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

one2:

invoke drd\_imageDraw,offset OneCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

two2:

invoke drd\_imageDraw,offset TwoCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

three2:

invoke drd\_imageDraw,offset ThreeCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

four2:

invoke drd\_imageDraw,offset FourCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

five2:

invoke drd\_imageDraw,offset FiveCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

six2:

invoke drd\_imageDraw,offset SixCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

seven2:

invoke drd\_imageDraw,offset SevenCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

eight2:

invoke drd\_imageDraw,offset EightCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

nine2:

invoke drd\_imageDraw,offset NineCIMG, HScoreSecondXDist, HScoreSecondYDist

jmp endUA

endUA:

popa

ret

HighScoreAnzeiger ENDP

VKhandler PROC

pusha

inc turn2

X mov eax, turn2 \ cmp eax, SPEED

je mainui

jmp anzeiger

mainui:

X mov ebx, 0 \ mov turn2, ebx

X invoke GetAsyncKeyState, VK\_RIGHT \ cmp eax, 0 \ jne right

X invoke GetAsyncKeyState, VK\_LEFT \ cmp eax, 0 \ jne left

X invoke GetAsyncKeyState, VK\_UP \ cmp eax, 0 \ jne up

X invoke GetAsyncKeyState, VK\_DOWN \ cmp eax, 0 \ jne down

X invoke GetAsyncKeyState, VK\_RETURN \ cmp eax, 0 \ jne selectt

X invoke GetAsyncKeyState, VK\_SPACE \ cmp eax, 0 \ jne selectt

X invoke GetAsyncKeyState, 53h \ cmp eax, 0 \ jne selectt

jmp anzeiger

selectt:

inc pixelCounter

mov ebx, pixelCounter

SelecterProc:

cmp pixelCounter, 1

je firstSelection

cmp pixelCounter, 2

je secondSelection

cmp pixelCounter, 3

je thirdSelection

firstSelection:

X mov edx, selecter \ mov selecter1, edx

mov edx, selecter

mov eax, [CardsOnBoard+edx\*4]

mov [SelectedCards], eax

X mov eax ,selecter \ mov selecterProc, eax

jmp anzeiger

secondSelection:

X mov eax, selecter \ cmp eax, selecter1 \ je nothing

X mov edx, selecter \ mov selecter2, edx

mov edx, selecter

mov eax, [CardsOnBoard+edx\*4]

mov [SelectedCards+4], eax

X mov eax ,selecter \ mov selecterProc, eax

jmp anzeiger

thirdSelection:

X mov eax, selecter \ cmp eax, selecter1 \ je nothing

X mov eax, selecter \ cmp eax, selecter2 \ je nothing

X mov edx, selecter \ mov selecter3, edx

mov edx, selecter

mov eax, [CardsOnBoard+edx\*4]

mov [SelectedCards+8], eax

X mov eax ,selecter \ mov selecterProc, eax

jmp anzeiger

right:;the array of CardsOnBoard is like the cards on the board, ; 0 1 2

cmp selecter, 2 ; 3 4 5

je rightIfTwo ; 6 7 8

cmp selecter, 5

je rightIfFive

cmp selecter, 8

je rightIfEight

mov edx, selecter

inc edx

mov selecter, edx

jmp anzeiger

rightIfTwo:

mov selecter, 0

jmp anzeiger

rightIfFive:

mov selecter, 3

jmp anzeiger

rightIfEight:

mov selecter, 6

jmp anzeiger

left:

cmp selecter, 0

je leftIfZero

cmp selecter, 3

je leftIfThree

cmp selecter, 6

je leftIfSix

X mov esi, selecter \ dec esi \ mov selecter, esi

jmp anzeiger

leftIfZero:

mov selecter, 2

jmp anzeiger

leftIfThree:

mov selecter, 5

jmp anzeiger

leftIfSix:

mov selecter, 8

jmp anzeiger

up:

cmp selecter, 0

je upIfZero

cmp selecter, 1

je upIfOne

cmp selecter, 2

je upIfTwo

X mov esi, selecter \ sub esi, 3 \ mov selecter, esi

jmp anzeiger

upIfZero:

mov selecter, 6

jmp anzeiger

upIfOne:

mov selecter, 7

jmp anzeiger

upIfTwo:

mov selecter, 8

jmp anzeiger

down:

cmp selecter, 6

je downIfSix

cmp selecter, 7

je downIfSeven

cmp selecter, 8

je downIfEight

X mov esi, selecter \ add esi, 3 \ mov selecter, esi

jmp anzeiger

downIfSix:

mov selecter, 0

jmp anzeiger

downIfSeven:

mov selecter, 1

jmp anzeiger

downIfEight:

mov selecter, 2

jmp anzeiger

escapevk:

push esi

mov esi, 1234

mov escapeCondition, esi

pop esi

jmp finallll

anzeiger:

cmp pixelCounter, 0

je anzeigerSelcted0

cmp pixelCounter, 1

je anzeigerSelcted1

cmp pixelCounter, 2

je anzeigerSelcted2

cmp pixelCounter, 3

je anzeigerSelcted3

anzeigerSelcted0:

invoke fromSelecterToLocatiosn, 0, selecter

invoke drd\_imageDraw, offset selectedSquere1Img, firstPositionX , firstPositionY

jmp enddd

anzeigerSelcted1:

invoke fromSelecterToLocatiosn, 1, selecter

invoke drd\_imageDraw, offset selectedSquere1Img, firstPositionX , firstPositionY

invoke drd\_imageDraw, offset selectedSquere2Img, secondPositionX , secondPositionY

;#TODO: audio

jmp enddd

anzeigerSelcted2:

invoke fromSelecterToLocatiosn, 2, selecter

invoke drd\_imageDraw, offset selectedSquere1Img, firstPositionX , firstPositionY

invoke drd\_imageDraw, offset selectedSquere2Img, secondPositionX , secondPositionY

invoke drd\_imageDraw, offset selectedSquere3Img, thirdPositionX , thirdPositionY

jmp enddd

anzeigerSelcted3:

invoke drd\_imageDraw, offset selectedSquere1Img, firstPositionX , firstPositionY

invoke drd\_imageDraw, offset selectedSquere2Img, secondPositionX , secondPositionY

invoke drd\_imageDraw, offset selectedSquere3Img, thirdPositionX , thirdPositionY

mov ebx ,selecter1

mov ecx, selecter2

mov eax, selecter3

invoke SetChecker, [SelectedCards], [SelectedCards+4], [SelectedCards+8]

cmp ThereIsSet, 2

je Adddd

jne subScore

jmp enddd

Adddd:

add Score, 5

invoke addCards

jmp enddd

subScore:

X mov eax, Score \ cmp eax, 0 \ je MMinus

X sub eax, 1 \ mov Score, eax

MMinus:

enddd:

jmp finallll

nothing:

dec pixelCounter

finallll:

popa

ret

VKhandler ENDP

RandomCard PROC

LOCAL hprovide:HANDLE

pusha

RandomCardreset:

invoke CryptAcquireContext, addr hprovide,0,0,PROV\_RSA\_FULL,CRYPT\_VERIFYCONTEXT or CRYPT\_SILENT

invoke CryptGenRandom, hprovide, 4, offset randomcardnumber

invoke CryptReleaseContext,hprovide,0

mov eax, randomcardnumber

mov ebx, 24

xor edx, edx

div ebx

cmp edx, 0

je RandomCardreset

mov randomcardnumber, edx

popa

ret

RandomCard ENDP

ScoreAnzeiger PROC

pusha

mov eax, Score

mov edx, 0

mov ebx, 10

div ebx

;edx - modulu

;eax - division

mov moduluAns, edx

mov divisionAns, eax

X mov eax, divisionAns \ cmp eax, 0

je zero

X mov eax, divisionAns \ cmp eax, 1

je one

X mov eax, divisionAns \ cmp eax, 2

je two

X mov eax, divisionAns \ cmp eax, 3

je three

X mov eax, divisionAns \ cmp eax, 4

je four

X mov eax, divisionAns \ cmp eax, 5

je five

X mov eax, divisionAns \ cmp eax, 6

je six

X mov eax, divisionAns \ cmp eax, 7

je seven

X mov eax, divisionAns \ cmp eax, 8

je eight

X mov eax, divisionAns \ cmp eax, 9

je nine

secondUA:

X mov edx, moduluAns

cmp edx, 0

je zero2

cmp edx, 1

je one2

cmp edx, 2

je two2

cmp edx, 3

je three2

cmp edx, 4

je four2

cmp edx, 5

je five2

cmp edx, 6

je six2

cmp edx, 7

je seven2

cmp edx, 8

je eight2

cmp edx, 9

je nine2

zero:

invoke drd\_imageDraw,offset ZeroCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

one:

invoke drd\_imageDraw,offset OneCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

two:

invoke drd\_imageDraw,offset TwoCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

three:

invoke drd\_imageDraw,offset ThreeCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

four:

invoke drd\_imageDraw,offset FourCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

five:

invoke drd\_imageDraw,offset FiveCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

six:

invoke drd\_imageDraw,offset SixCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

seven:

invoke drd\_imageDraw,offset SevenCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

eight:

invoke drd\_imageDraw,offset EightCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

nine:

invoke drd\_imageDraw,offset NineCIMG, ScoreFirstXDist, ScoreFirstYDist

jmp secondUA

zero2:

invoke drd\_imageDraw,offset ZeroCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

one2:

invoke drd\_imageDraw,offset OneCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

two2:

invoke drd\_imageDraw,offset TwoCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

three2:

invoke drd\_imageDraw,offset ThreeCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

four2:

invoke drd\_imageDraw,offset FourCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

five2:

invoke drd\_imageDraw,offset FiveCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

six2:

invoke drd\_imageDraw,offset SixCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

seven2:

invoke drd\_imageDraw,offset SevenCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

eight2:

invoke drd\_imageDraw,offset EightCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

nine2:

invoke drd\_imageDraw,offset NineCIMG, ScoreSecondXDist, ScoreSecondYDist

jmp endUA

endUA:

popa

ret

ScoreAnzeiger ENDP

TimerStarter PROC

pusha

xor ecx, ecx

invoke GetSystemTime ,addr STime

mov cx, STime.wSecond

mov StartTime, cx

mov StartTimeReal, cx

popa

ret

TimerStarter ENDP

UhrAnzeiger PROC

pusha

invoke GetSystemTime ,addr STime

mov cx, StartTime

mov bx, STime.wSecond

cmp bx, cx

je putTheTimer

jne setTimeDec

setTimeDec:

dec setTime

inc StartTime

mov dx, 60

cmp StartTime, dx

je resetStartTime

mov dx, StartTimeReal

cmp setTime, dx

je endofgamebecauseua

jmp putTheTimer

resetStartTime:

mov dx, 0

mov StartTime, dx

jmp putTheTimer

endofgamebecauseua:

mov esi, 1

mov UAENDINGP, esi

jmp endUA

putTheTimer:

;for the divide:

mov ax, setTime

mov edx, 0

mov ebx, 10

div ebx

;edx - modulu

;eax - division

mov divisionAns, eax

mov moduluAns, edx

cmp setTime, 0

je endingParameter

X mov eax, divisionAns \ cmp eax, 0

je zero

X mov eax, divisionAns \ cmp eax, 1

je one

X mov eax, divisionAns \ cmp eax, 2

je two

X mov eax, divisionAns \ cmp eax, 3

je three

X mov eax, divisionAns \ cmp eax, 4

je four

X mov eax, divisionAns \ cmp eax, 5

je five

X mov eax, divisionAns \ cmp eax, 6

je six

X mov eax, divisionAns \ cmp eax, 7

je seven

X mov eax, divisionAns \ cmp eax, 8

je eight

X mov eax, divisionAns \ cmp eax, 9

je nine

secondUA:

X mov edx, moduluAns

cmp moduluAns, esi

je zero2

cmp edx, 0

je zero2

cmp edx, 1

je one2

cmp edx, 2

je two2

cmp edx, 3

je three2

cmp edx, 4

je four2

cmp edx, 5

je five2

cmp edx, 6

je six2

cmp edx, 7

je seven2

cmp edx, 8

je eight2

cmp edx, 9

je nine2

jmp endUA

zero:

invoke drd\_imageDraw,offset ZeroCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

one:

invoke drd\_imageDraw,offset OneCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

two:

invoke drd\_imageDraw,offset TwoCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

three:

invoke drd\_imageDraw,offset ThreeCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

four:

invoke drd\_imageDraw,offset FourCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

five:

invoke drd\_imageDraw,offset FiveCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

six:

invoke drd\_imageDraw,offset SixCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

seven:

invoke drd\_imageDraw,offset SevenCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

eight:

invoke drd\_imageDraw,offset EightCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

nine:

invoke drd\_imageDraw,offset NineCIMG, TimerFirstXDist, TimerFirstYDist

jmp secondUA

zero2:

invoke drd\_imageDraw,offset ZeroCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

one2:

invoke drd\_imageDraw,offset OneCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

two2:

invoke drd\_imageDraw,offset TwoCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

three2:

invoke drd\_imageDraw,offset ThreeCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

four2:

invoke drd\_imageDraw,offset FourCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

five2:

invoke drd\_imageDraw,offset FiveCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

six2:

invoke drd\_imageDraw,offset SixCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

seven2:

invoke drd\_imageDraw,offset SevenCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

eight2:

invoke drd\_imageDraw,offset EightCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

nine2:

invoke drd\_imageDraw,offset NineCIMG, TimerSecondXDist, TimerSecondYDist

jmp endUA

endUA:

jmp endendUA

endingParameter:

mov esi, 1

mov UAENDINGP, esi

endendUA:

popa

ret

UhrAnzeiger ENDP

main PROC

invoke drd\_init, w\_width, w\_height, INIT\_WINDOW

invoke PlaySound,addr SoundfileMunch,NULL,SND\_ASYNC

invoke drd\_imageLoadFile,offset Menu, offset MenuImg

invoke drd\_imageLoadFile,offset pngSplash2, offset Splash2Img

invoke drd\_imageLoadFile,offset pngSplash3, offset Splash3Img

invoke drd\_imageLoadFile,offset MMstartpage, offset MMstartpageImg

invoke drd\_imageLoadFile, offset selectedSquere1, offset selectedSquere1Img

invoke drd\_imageLoadFile, offset selectedSquere2, offset selectedSquere2Img

invoke drd\_imageLoadFile, offset selectedSquere3, offset selectedSquere3Img

invoke drd\_imageLoadFile, offset Instructions, offset InstructionsImg

invoke drd\_imageLoadFile, offset Options, offset OptionsImg

invoke drd\_imageLoadFile, offset MMselecter, offset MMselecterImg

invoke drd\_imageLoadFile, offset Menu, offset MenuImg

invoke drd\_imageLoadFile,offset ZeroC, offset ZeroCIMG

invoke drd\_imageLoadFile,offset OneC, offset OneCIMG

invoke drd\_imageLoadFile,offset TwoC, offset TwoCIMG

invoke drd\_imageLoadFile,offset ThreeC, offset ThreeCIMG

invoke drd\_imageLoadFile,offset FourC, offset FourCIMG

invoke drd\_imageLoadFile,offset FiveC, offset FiveCIMG

invoke drd\_imageLoadFile,offset SixC, offset SixCIMG

invoke drd\_imageLoadFile,offset SevenC, offset SevenCIMG

invoke drd\_imageLoadFile,offset EightC, offset EightCIMG

invoke drd\_imageLoadFile,offset NineC, offset NineCIMG

invoke drd\_imageLoadFile,offset TblueFull, offset TblueFull\_Img

invoke drd\_imageLoadFile,offset TblueStrips, offset TblueStrips\_Img

invoke drd\_imageLoadFile,offset TblueEmpty, offset TblueEmpty\_Img

invoke drd\_imageLoadFile,offset TgreenFull, offset TgreenFull\_Img

invoke drd\_imageLoadFile,offset TgreenStrips, offset TgreenStrips\_Img

invoke drd\_imageLoadFile,offset TgreenEmpty, offset TgreenEmpty\_Img

invoke drd\_imageLoadFile,offset TredFull, offset TredFull\_Img

invoke drd\_imageLoadFile,offset TredStrips, offset TredStrips\_Img

invoke drd\_imageLoadFile,offset TredEmpty, offset TredEmpty\_Img

invoke drd\_imageLoadFile,offset CblueFull, offset CblueFull\_Img

invoke drd\_imageLoadFile,offset CblueStrips, offset CblueStrips\_Img

invoke drd\_imageLoadFile,offset CblueEmpty, offset CblueEmpty\_Img

invoke drd\_imageLoadFile,offset CgreenFull, offset CgreenFull\_Img

invoke drd\_imageLoadFile,offset CgreenStrips, offset CgreenStrips\_Img

invoke drd\_imageLoadFile,offset CgreenEmpty, offset CgreenEmpty\_Img

invoke drd\_imageLoadFile,offset CredFull, offset CredFull\_Img

invoke drd\_imageLoadFile,offset CredStrips, offset CredStrips\_Img

invoke drd\_imageLoadFile,offset CredEmpty, offset CredEmpty\_Img

invoke drd\_imageLoadFile,offset SblueFull, offset SblueFull\_Img

invoke drd\_imageLoadFile,offset SblueStrips, offset SblueStrips\_Img

invoke drd\_imageLoadFile,offset SblueEmpty, offset SblueEmpty\_Img

invoke drd\_imageLoadFile,offset SgreenFull, offset SgreenFull\_Img

invoke drd\_imageLoadFile,offset SgreenStrips, offset SgreenStrips\_Img

invoke drd\_imageLoadFile,offset SgreenEmpty, offset SgreenEmpty\_Img

invoke drd\_imageLoadFile,offset SredFull, offset SredFull\_Img

invoke drd\_imageLoadFile,offset SredStrips, offset SredStrips\_Img

invoke drd\_imageLoadFile,offset SredEmpty, offset SredEmpty\_Img

invoke drd\_imageSetTransparent, offset NineCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset EightCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset SevenCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset SixCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset FiveCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset FourCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset ThreeCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset TwoCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset OneCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset ZeroCIMG, 0FFFFFFh

invoke drd\_imageSetTransparent, offset selectedSquere1Img, 0FFFFFFh

invoke drd\_imageSetTransparent, offset selectedSquere2Img, 0FFFFFFh

invoke drd\_imageSetTransparent, offset selectedSquere3Img, 0FFFFFFh

;invoke drd\_imageSetTransparent, offset objs[16], 0525252h

;invoke init

invoke drd\_imageLoadFile,offset Menu, offset MenuImg

invoke putAllCards

Lsplash1Start:

mov ebx, 0

;audio

Lsplash1:

mov ecx, 1

invoke drd\_pixelsClear, black

invoke drd\_imageDraw, offset Splash2Img, 0, 0

invoke drd\_processMessages

invoke drd\_flip

inc ebx

cmp ebx, 500

je Lsplash2Start

jmp Lsplash1

Lsplash2Start:

mov ebx, 0

Lsplash2:

mov ecx, 1

invoke drd\_pixelsClear, black

invoke drd\_imageDraw, offset Splash3Img, 0, 0

invoke drd\_processMessages

invoke drd\_flip

inc ebx

cmp ebx, 500

je mainmenumainRESET

jmp Lsplash2

mainmenuInstructions:

X invoke GetAsyncKeyState, VK\_ESCAPE\ cmp eax, 0 \ jne mainmenumainRESET

mov ecx, 1

invoke drd\_pixelsClear, black

invoke drd\_imageDraw, offset InstructionsImg, 0 , 0

invoke drd\_processMessages

invoke drd\_flip

jmp mainmenuInstructions

mainmenuOptionsrset:

mov esi, 0

mov SoundSelected, esi

mov muteSelected, esi

mov speedSelected, esi

mainmenuOptions:

mov ecx, 1

X invoke GetAsyncKeyState, VK\_ESCAPE \ cmp eax, 0 \ jne mainmenumainRESET

invoke drd\_pixelsClear, black

invoke OptionsProc

mov esi, 1

cmp SoundSelected, esi

je Mute

cmp muteSelected, esi

je Sound

cmp speedSelected, esi

je heatza

invoke drd\_processMessages

invoke drd\_flip

jmp mainmenuOptions

Mute:

mov ebx, 0

mov muteSelected, ebx

mov esi, 0

mov ifSound, esi

jmp mainmenuOptionsrset

Sound:

mov ebx, 0

mov SoundSelected, ebx

mov esi, 1

mov ifSound, esi

jmp mainmenuOptionsrset

heatza:

mov ebx, 0

mov speedSelected, ebx

mov esi, CondH

cmp esi, 0

je yesh

X MOV EAX, 100 \ MOV SPEED, EAX

mov esi, 0

mov CondH, esi

jmp mainmenuOptionsrset

yesh:

X MOV EAX, 30 \ MOV SPEED, EAX

mov esi, 1

mov CondH, esi

jmp mainmenuOptionsrset

mainmenumainRESET:

push esi

mov ax, 60

mov setTime, ax

mov esi, 0

mov MainMenuCounter, esi

mov UAENDINGP, esi

mov mainMenuSelecter, esi

mov OptionsSelected, esi

mov StartSelected, esi

mov InstructionsSelected, esi

mov esi, 490

mov HScoreFirstXDist , esi

mov esi, 314

mov HScoreFirstYDist , esi

mov esi, 540

mov HScoreSecondXDist, esi

mov esi, 314

mov HScoreSecondYDist, esi

pop esi

mov esi, 0

cmp ifSound, esi

je mainmenumain

invoke PlaySound,addr Soundfile,NULL,SND\_ASYNC

mainmenumain:

mov ecx, 1

invoke drd\_pixelsClear, black

X invoke GetAsyncKeyState, VK\_DELETE \ cmp eax, 0 \ jne ending

invoke MainMenuProc

invoke HighScoreAnzeiger

mov esi, 1

cmp StartSelected, esi

je againStarT

cmp InstructionsSelected, esi

je mainmenuInstructions

cmp OptionsSelected, esi

je mainmenuOptionsrset

invoke drd\_processMessages

invoke drd\_flip

jmp mainmenumain

againStarT:

invoke putAllCards

mov esi, 0

mov Score, esi

mov turn2, esi

mov esi, 87

mov HScoreFirstXDist , esi

mov esi, 491

mov HScoreFirstYDist , esi

mov esi, 137

mov HScoreSecondXDist, esi

mov esi, 491

mov HScoreSecondYDist, esi

invoke TimerStarter

mov esi, 0

cmp ifSound, esi

je again

invoke PlaySound,addr Soundfile,NULL,SND\_ASYNC

again:

mov ecx, 1

invoke drd\_pixelsClear, black

invoke drd\_imageDraw, offset MenuImg, 0, 0

invoke fromCardsOnBoardtoBoard

invoke VKhandler

invoke ScoreAnzeiger

invoke HighScore

invoke HighScoreAnzeiger

invoke UhrAnzeiger

mov ecx, 1

cmp UAENDINGP, ecx

je mainmenumainRESET

invoke setOnBoardAnzeiger

X invoke GetAsyncKeyState, VK\_ESCAPE \ cmp eax, 0 \ jne mainmenumainRESET

invoke drd\_processMessages

invoke drd\_flip

je mainmenumainRESET

mov esi, 4590

cmp Ending, esi

je ending

jmp again

ending:

ret 0

main ENDP

end main

רפלקציה אישית

במהלך פרויקט האסמבלי שלי למדתי רבות. למדתי בעיקר על התהליך של למידה עצמית, ושאם אני באמת רוצה ללמוד משהו לעומק עם הרבה השקעה אוכל לעשות אותו. כשהתחלתי את הפרויקט ידעתי קצת אסמבלי מהשיעורים שלנו בבית-הספר, איך בכלל לא ידעתי איך באמת כותבים פרויקט שלם ומקשרים בין כל החלקים.

כמו כן, אהבתי מאוד את תהליך העבודה הכיתתי שעשיתי, ככיתה התייעצנו ונתנו עצות, ולא חששנו לקחת רעיונות אחד מהשני ,לחשוב ביחד על פתרונות יותר טובים או לבקש עזרה מתלמידים בוגרים ממנו- הבנתי שזו לא בושה אלא מעלה, וכך גם הצלחתי לעזור לעוד הרבה אנשים אחרים מכיתתי שהתחילו את הפרויקט יותר מאוחר ממני.