פרויקט יחידה אסמבלי32 2018 - Chrome dino

כיתה: י'3

מנחה: יהודה אור

מגיש: אופק כפיר – 323821991

תאריך הגשה – 20.06.2018



תוכן עניינים:

3 – הסבר על המשחק ורפלקציה

4-5 – תמונות מהמשחק

6-9 – הסבר על אסמבלי והמחשב

10– פונקציות DRD

11-12– האוגרים

13- תרשים UML

14-19 – הסבר על הפונקציות

20-33 – הקוד

34- קישורים

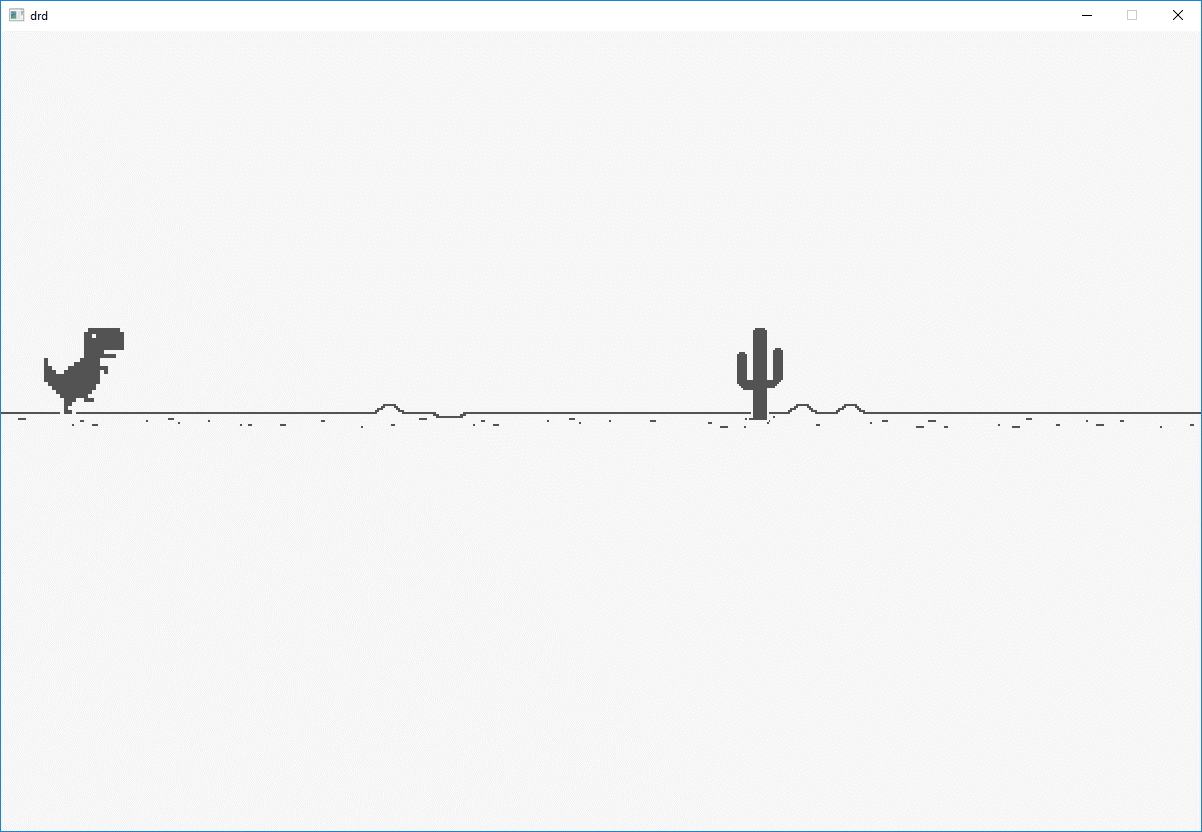
הסבר על המשחק:

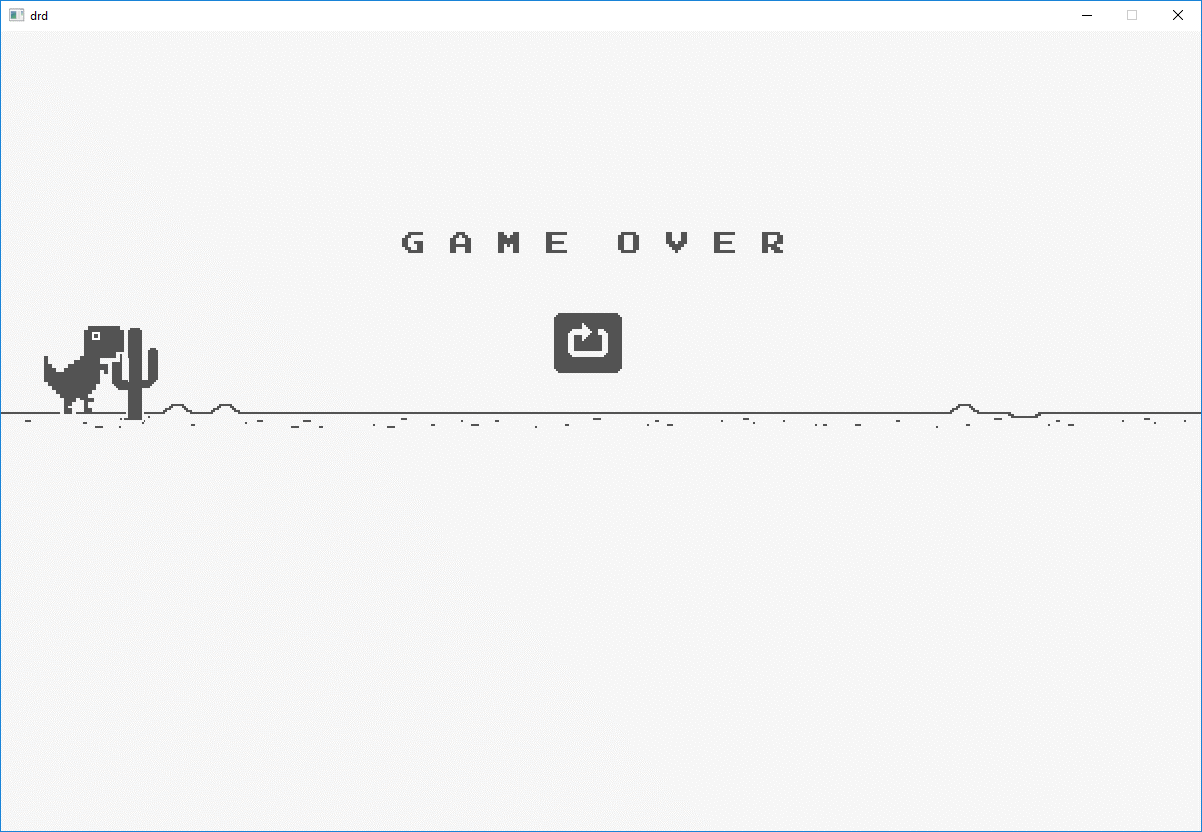
המשחק מבוסס על הEaster egg שנמצא כאשר אתה בלי אינטרנט בדפדפני Chromium, כאשר אתה לוחץ על רווח, הEaster egg מפעיל משחק שכולל דינוזאור שמטרתו לברוח, ככל שאתה משיג יותר ניקוד ככה אתה יותר טוב. הדינוזאור הפך לפופלרי במיוחד, ונעשו עליו מאות חיקויים. אני החלטתי לעשות את הeaster egg באסמבלי32 עם אמצעים גרפיים.

רפלקציה:

במהלך העבודה על המשחק, נתקלתי בבעיות רבות שלא ידעתי את פתרונם והייתי צריך להשתמש באינטרנט פעמים רבות על מנת למצוא פתרונות לבעיות אשר אתגרו את המחשבה שלי. (למשל: לא ידעתי שedx מאחסן את השארית של המספר ולכן צריך לאפס אותו לפני פעולות מתמטיות). למדתי איך לחפש ממוקד באינטרנט פתרונות לבעיות שלי, ולשאול בפורומים. במהלך העבודה השתמשתי במדריכים רבים ברשת אשר עזרו לי בהבנת השפה ואיך הדברים עובדים במחשב ובראשם **ארגון המחשב ושפת סף מאת ברק גונן** במהלך העבודה גם למדתי איך להשתמש בסביבת עבודה של Masm32 וVisual studio ולמדתי גם איך להשתמש בכלי השימושי שנקרא Git על מנת לשמור את העבודה שלי ולתעד את השינויים עם הזמן. (הקישור לrepo יצורף בסוף הספר תחת "קישורים".)

תמונות מהמשחק:





קצת על אסמבלי והמחשב

המחשב כיום מורכב מרכיבים רבים: מעבד, דיסק קשיח, ראם, לוח אם, מערכת הפעלה, המעבד הגרפי, המסך ועוד. בין כל הרכיבים קיימת הרמוניה ולכל רכיב תפקיד שונה במחשב הייחודי לו, אני ארחיב על כל אחד מהרכיבים אשר אנו מתעסקים בהם כיום.

המעבד - CPU (Central processing unit )

המעבד הינו היחידת עיבוד המרכזית של המחשב, בו מתבצעים כל החישובים של המחשב והפקודות ההכרחיות לכל יחידה דיגיטלית. בכל מעבד ישנם כמה מרכיבים הכרחיים:

* אוגרים – מיועדים לאחסון ערכים ופעולות מתמטיות (הרחבה בהמשך)
* יחידה לוגית – אחראית על הפעולות המתמטיות ועל השערים הלוגים (XOR,AND,NOR,OR... (
* יחידת בקרה – אחראית על קריאת ופיענוח הפקודות.

לכל מעבד ארכיטקטורה בלעדית משלו על פי היצרן של המעבד, בפרויקט זה השתמשנו בארכיטקטורת 8086 מאת אינטל אשר יצאה ב1976 (עובדת בונוס: לכבוד ה40 שנה יצא השנה מעבד i7-8086K אשר נמצא עכשיו בשוק במהדורה מוגבלת). ל8086 של אינטל יצאו המשכים רבים עד היום והם נחשבים כחלק ממשפחת X86 שהיא הארכיטקטורה הנפוצה ביותר כיום במחשבים.

זיכרון (ראם)

הראם הינו הזיכרון המהיר במחשב, והוא נועד על מנת לגשת במהירות אל התכנית ולאחסון הנתונים של התכנית במקום לגשת כל פעם אל הדיסק הקשיח שנחשב זיכרון איטי. הזיכרון המהיר במחשב מחולק ל3 חלקים:

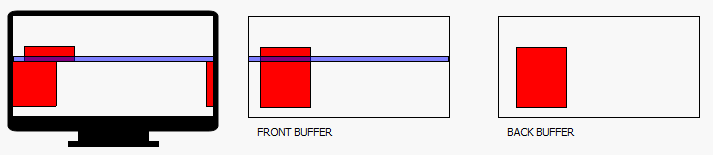
* הראם- נועד על מנת לרשום את השפת מכונה ואת הקוד שלנו.
* קלט/פלט – נועד על מנת לקלוט את המכשירים השונים של המחשב, כמו המקלדת.
* ROM (Read only memory )- חלק זה בזיכרון אינו ניתן לכתיבה ומאוחסנים בו הוראות של היצרן על מנת שהמחשב יפעל כמו שצריך, זיכרון זה ניתן לקריאה אך לא לכתיבה.

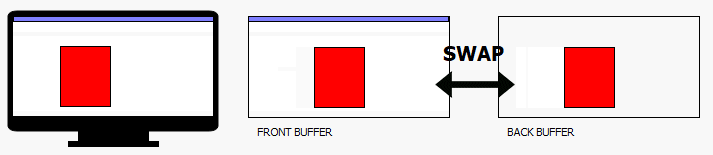
תוכן הראם נמחק ברגע שמנתקים אותו מהחשמל, ולכן הוא זיכרון זמני.

המעבד הגרפי

המעבד הגרפי הינה יחידת עיבוד שנמצאת במעבד או ביחידה נפרדת מהמעבד והיא אחראית על התמונה במחשב והצגתו למסך, המעבד הגרפי מציג את התמונה כמטריצה גדולה של פיקסלים למסך והוא יכול לבצע חישובים מסובכים שהCPU לא יכול.

המעבד הגרפי עובד ככה שהוא מצייר לבאפר האחורי את התמונה הבאה שהמחשב רוצה להראות, ומעביר אותה לבאפר הקדמי שמשם המסך לוקח את התמונה. בזמן שהמסך מצייר את התמונה מהבאפר הקדמי, הבאפר האחורי כבר עסוק בלצייר את התמונה הבאה וכך הלאה.





הדיסק הקשיח

הדיסק הקשיח הינו המקום בו מאחסנים את הקבצים הדרושים להפעלת התוכנה, כמו התמונות והסאונד. הדיסק הקשיח הוא אחסון תמידי ונועד לטווח ארוך והוא נשמר גם כאשר הוא מנותק מהחשמל.

אז מה זה אסמבלי?:

אסמבלי היא שפת סף אשר נחשבת הכי קרובה למחשב אחרי שפת מחשב (בינארית). אסמבלי לעומת שפות עיליות כמו Java, C#, C++ נחשבת שפה שקשה לקרוא, אם תסתכל במבט ראשון על קוד באסמבלי ככל הנראה שלא תבין כלום רק אחרי קריאה והבנה מעמיקה, לעומת Java למשל שכל פקודה שם ברורה למוח האנושי גם אם אין רקע נרחב בתכנות. באסמבלי אתה נדרש להגיד למערכת מה לעשות בדיוק ובלי ידע נרחב באיך עובד המחשב והרכיבים בו לא תצליח לכתוב קוד. באסמבלי יש שימוש נרחב באוגרים ובזיכרון ואם אתה עובר על משהו שמנוגד לשפת המחשב המחשב יודיע לך על זה מיד.

אז כמו שאתם שואלים, למה בכלל צריך לתכנת באסמבלי אם אפשר להשתמש בשפות עיליות אשר יותר שימושיות וקלות? ישנן כמה סיבות מרכזיות:

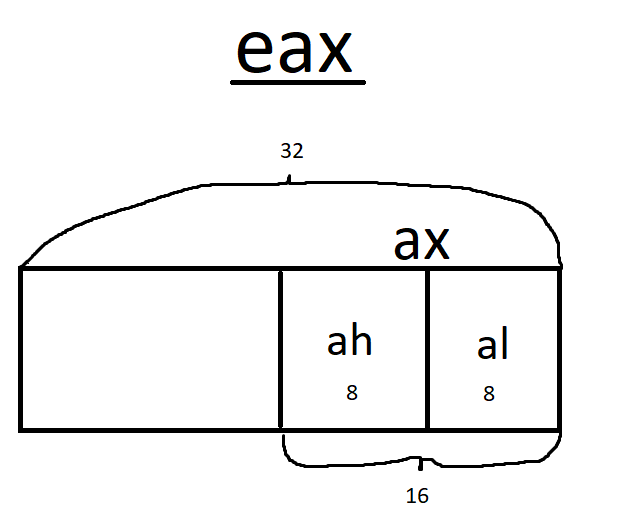
* אסמבלי עוזרת להבנה מעמיקה של המחשב וחלקיו, דבר שיעזור בעתיד בהבנה של בעיות תכנות לא שגרתיות.
* בגלל שאסמבלי כל כך קרובה למחשב, השפה הינה יעילה מאוד מבחינת משאבים לעומת שפות עיליות.
* אם צריך לרשום משהו שקרוב למחשב כמו דרייבר, דברים יכולים להשתבש מאוד מהר ברגע שרושמים אותם בשפות עיליות, ולכן כדאי לדעת אסמבלי על מנת לתקן בעיות אלו.

במהלך העבודה השתמשתי בסביבת העבודה של מיקרוסופט Visual studio ביחד עם ספריית masm32 שהיא בעצם הספרייה של מיקרוספוט משנות ה90 ועם הספרייה הגרפית drd מאת שי שלום. Visual studio הינה סביבת עבודה שימושית ביותר הכוללת Git, דיבאג יעיל שכולל צפייה בערכים, סדר של הקבצים והמשאבים.

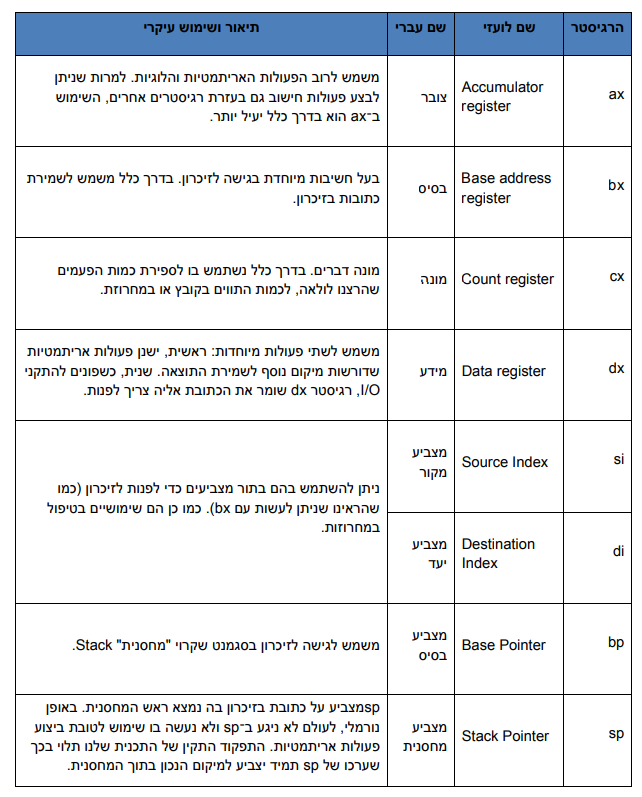
פונקציות של DRD אשר השתמשתי בהם בפרויקט והסבר:

|  |  |
| --- | --- |
| drd\_init | פונקצייה אשר פותחת חלון על פי x ו y |
| drd\_imageLoadFile | פונקצייה אשר טוענת את התמונה מהספריית קבצים בתוך התוכנה |
| drd\_imageSetTransparent | פונקצייה אשר מגדירה את הצבע שיש להסיר מהתמונה על מנת שהחלק המרכזי בתמונה יהיה לבד |
| drd\_pixelsClear | פונקצייה שמנקה כל דבר שיש בחלון |
| drd\_imageDrawCrop | פונקצייה שחותכת חלק מהתמונה ומציירת אותו על המסך |
| drd\_imageDraw | פונקצייה שמציירת תמונה על המסך |
| drd\_flip | פונקצייה אשר מציגה את מה שיש בפרונט באפר על המסך |
| drd\_processMessages | פונקצייה שמאפשרת לחלון לקבל פקודות מהמשתמש |

האוגרים:

במשפחת המעבדים x86 משתמשים באוגרים אשר הם 32 ביט. אוגר של 32 ביט מורכב משני אוגרים של 16 ביט ואוגר של 16 ביט מורכב מאוגרים של 8 ביט למשל מצורף סרטוט של האוגר eax (32 ביט) ואת האוגרים אשר כוללים אותו

מצורף רשימת האוגרים ב16 ביט לקוחה מהספר של ברק גונן – ארגון המחשב ושפת סף

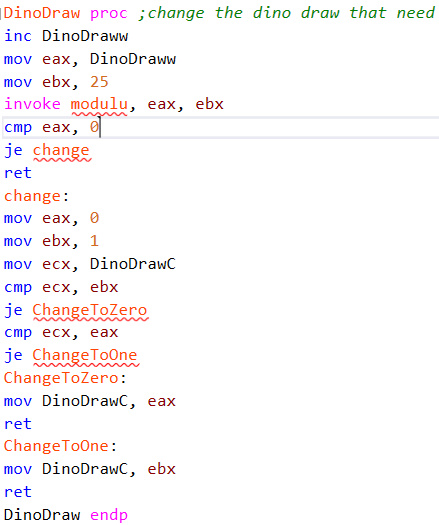


תרשים UML :

הסבר של הפונקציות:

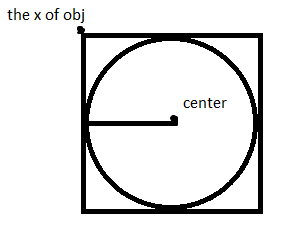
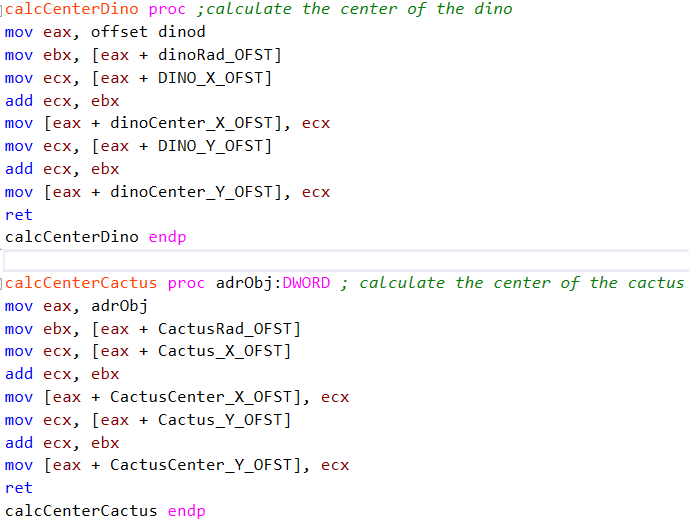
Dinodraw:

פונקצייה זו עושה את האנימציה של הרגליים של הדינוזאור על פי מונה שברגע שהשארית שלו עם 25 שווה ל0, הוא משנה את הרגל.



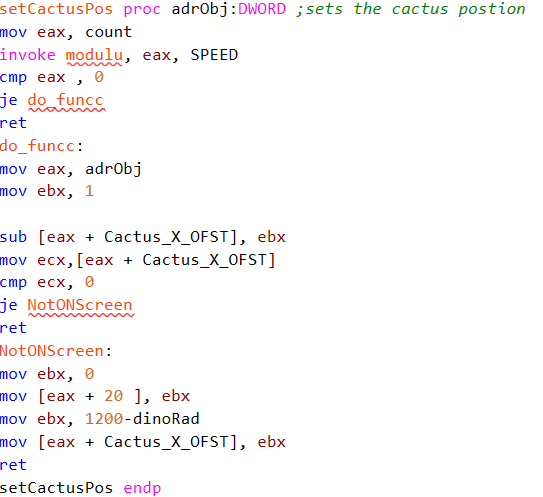
CalcCenterdino/cactus

שתי הפונקציות האלו פועלות על אותו עיקרון, אנחנו יודעים מה הרדיוס של הדינוזאור/קקטוס, ולכן אפשר לדעת על פי חיסור הרדיוס מהX של האובייקט מה האמצע של האובייקט.



SetCactusPos

הפונקצייה מתאמת בין המהירות של הרקע למהירות שבה הקקטוס זז ומורידה מהX יחידה אחת כל פעם, במקרה והקקטוס לא על המסך הוא חוזר להתחלה.

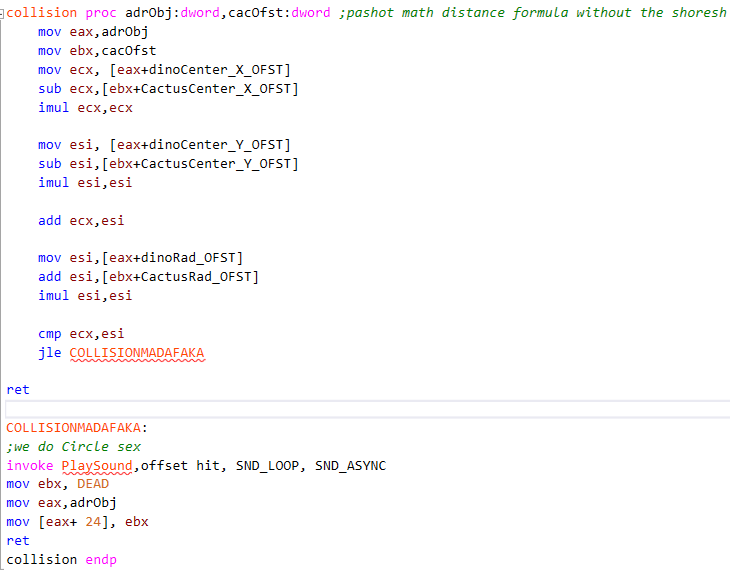


Collision

הפונקציה מחשבת את המרחק בין האובייקט למכשול על פי נוסחת המרחק במתמטיקה אנליטית (בלי השורש)

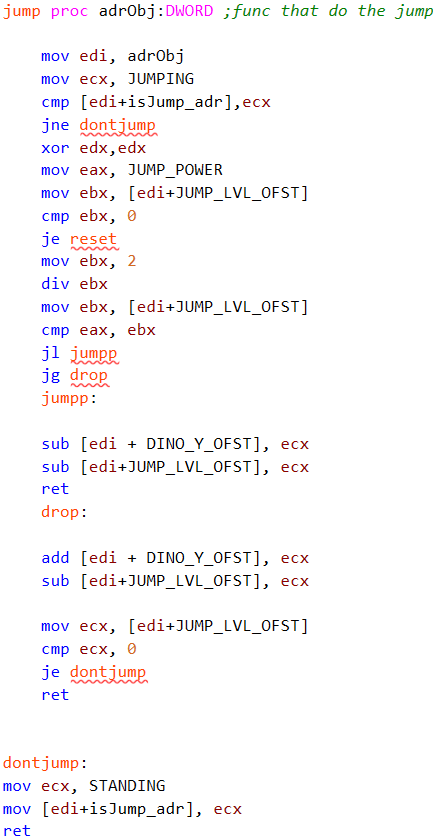
√((x1-x2)2+(y1-y2)2)

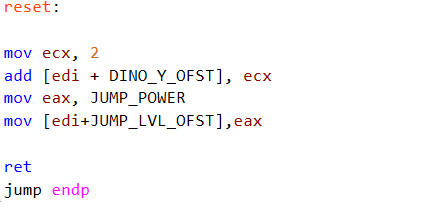
במקרה והמרחק קטן מסכום הרדיוסים אז הדמויות בהכרח נפגשות והפונקציה מפעילה את הפקודות הנדרשות על מנת שהמשחק יעצור



Jump

הפונקציה גורמת לדמות לקפוץ על פי JUMP\_LVL, הפונקציה מחלקת את הכוח ל2, ככה שהוא עולה הוא בחצי מהכוח שלו, וכשהוא יורד הוא ב0 כוחות. הפונקציה מורידה מערך הy 1 בכל פעם בעלייה, ומוסיפה לערך ה y אחד בכל פעם בירידה, עד שהוא מגיע לנקודה ההתחלתית.





הקוד המלא:

Main.asm

include \masm32\include\masm32rt.inc

include drd.inc

includelib drd.lib

include data.inc

include main\_funcs.inc

;sound

include \masm32\include\winmm.inc

includelib \masm32\lib\winmm.lib

.code

main proc

invoke init ;openning the window and loading the images

loopa:

invoke restart, offset dinod ;check if need to restart

invoke update ;update the loop and the funcs

invoke draw ;drawing the game

jmp loopa

ret

main endp

end main

funcs.inc

*;sound*

include \masm32\include\winmm.inc

includelib \masm32\lib\winmm.lib

.code

Key\_pressed\_space proc ;func that check if the space is pressed and active jump if it is

mov ebx, edi

add ebx, SPACE\_OFST

invoke GetAsyncKeyState,[ebx] ;throw to eax 1 if key pressed

cmp eax, 0

jne jumpfunc

ret

jumpfunc:

mov ecx,JUMPING

mov [edi+isJump\_adr], ecx

ret

Key\_pressed\_space endp

jump proc adrObj:DWORD ;func that do the jump

mov edi, adrObj

mov ecx, JUMPING

cmp [edi+isJump\_adr],ecx

jne dontjump

xor edx,edx

mov eax, JUMP\_POWER

mov ebx, [edi+JUMP\_LVL\_OFST]

cmp ebx, 0

je reset

mov ebx, 2

div ebx

mov ebx, [edi+JUMP\_LVL\_OFST]

cmp eax, ebx

jl jumpp

jg drop

jumpp:

sub [edi + DINO\_Y\_OFST], ecx

sub [edi+JUMP\_LVL\_OFST], ecx

ret

drop:

add [edi + DINO\_Y\_OFST], ecx

sub [edi+JUMP\_LVL\_OFST], ecx

mov ecx, [edi+JUMP\_LVL\_OFST]

cmp ecx, 0

je dontjump

ret

dontjump:

mov ecx, STANDING

mov [edi+isJump\_adr], ecx

ret

reset:

mov ecx, 2

add [edi + DINO\_Y\_OFST], ecx

mov eax, JUMP\_POWER

mov [edi+JUMP\_LVL\_OFST],eax

ret

jump endp

little funcs.inc

.686

.code

modulu proc first:DWORD, sec:DWORD ; returns x%y

xor edx, edx

mov eax,first

div sec

mov eax,edx ; eax contains the value of x%y

ret

modulu endp

DinoDraw proc ;change the dino draw that need

inc DinoDraww

mov eax, DinoDraww

mov ebx, 25

invoke modulu, eax, ebx

cmp eax, 0

je change

ret

change:

mov eax, 0

mov ebx, 1

mov ecx, DinoDrawC

cmp ecx, ebx

je ChangeToZero

cmp ecx, eax

je ChangeToOne

ChangeToZero:

mov DinoDrawC, eax

ret

ChangeToOne:

mov DinoDrawC, ebx

ret

DinoDraw endp

calcCenterDino proc ;calculate the center of the dino

mov eax, offset dinod

mov ebx, [eax + dinoRad\_OFST]

mov ecx, [eax + DINO\_X\_OFST]

add ecx, ebx

mov [eax + dinoCenter\_X\_OFST], ecx

mov ecx, [eax + DINO\_Y\_OFST]

add ecx, ebx

mov [eax + dinoCenter\_Y\_OFST], ecx

ret

calcCenterDino endp

calcCenterCactus proc adrObj:DWORD ; calculate the center of the cactus

mov eax, adrObj

mov ebx, [eax + CactusRad\_OFST]

mov ecx, [eax + Cactus\_X\_OFST]

add ecx, ebx

mov [eax + CactusCenter\_X\_OFST], ecx

mov ecx, [eax + Cactus\_Y\_OFST]

add ecx, ebx

mov [eax + CactusCenter\_Y\_OFST], ecx

ret

calcCenterCactus endp

Random proc max:dword;return a random number in eax between 0 and max

Rdrand eax

Xor edx,edx

div max

mov eax,edx

ret

Random endp

setCactusPos proc adrObj:DWORD ;sets the cactus postion

mov eax, count

invoke modulu, eax, SPEED

cmp eax , 0

je do\_funcc

ret

do\_funcc:

mov eax, adrObj

mov ebx, 1

sub [eax + Cactus\_X\_OFST], ebx

mov ecx,[eax + Cactus\_X\_OFST]

cmp ecx, 0

je NotONScreen

ret

NotONScreen:

mov ebx, 0

mov [eax + 20 ], ebx

mov ebx, 1200-dinoRad

mov [eax + Cactus\_X\_OFST], ebx

ret

setCactusPos endp

IsDrawCactus proc ;tells if there cactus on screen

invoke Random, 500

invoke modulu, eax, 43

cmp eax, 0

je tell\_to\_active

ret

tell\_to\_active:

mov eax, 1

mov CactusDraw, eax

ret

IsDrawCactus endp

collision proc adrObj:dword,cacOfst:dword ;pashot math distance formula without the shoresh

mov eax,adrObj

mov ebx,cacOfst

mov ecx, [eax+dinoCenter\_X\_OFST]

sub ecx,[ebx+CactusCenter\_X\_OFST]

imul ecx,ecx

mov esi, [eax+dinoCenter\_Y\_OFST]

sub esi,[ebx+CactusCenter\_Y\_OFST]

imul esi,esi

add ecx,esi

mov esi,[eax+dinoRad\_OFST]

add esi,[ebx+CactusRad\_OFST]

imul esi,esi

cmp ecx,esi

jle COLLISIONMADAFAKA

ret

COLLISIONMADAFAKA:

;we do Circle sex

invoke PlaySound,offset hit, SND\_LOOP, SND\_ASYNC

mov ebx, DEAD

mov eax,adrObj

mov [eax+ 24], ebx

ret

collision endp

CalculateDistance proc

xor edx,edx

inc DistanceCount

invoke modulu, DistanceCount, 10

cmp eax,0

je incCount

ret

incCount:

inc Distance

ret

CalculateDistance2 endp

data.inc

.const

DINO\_Y\_ONSCREEN = 294

DINO\_X\_ONSCREEN = 40

SPEED = 2 ;3 on laptop, 12 desktop

Jump\_speed = 3 ;; 4 on laptop, 16 desktop

DINO\_X\_OFST = 0

DINO\_Y\_OFST = 4

dinoRad\_OFST = 44

dinoRad = 35

dinoCenter\_X\_OFST = 36

dinoCenter\_Y\_OFST = 40

Cactus\_X\_OFST = 0

Cactus\_Y\_OFST = 4

CactusCenter\_X\_OFST = 8

CactusCenter\_Y\_OFST = 12

CactusRad\_OFST = 16

STANDING = 0

JUMPING = 1

isJump\_adr = 28

SPACE\_OFST = 16

DOWN\_OFST = 20

JUMP\_LVL\_OFST = 32

JUMP\_POWER = 430

DRCY = 12

DEAD = 0

ALIVE = 1

.data

h dword 800

w dword 1200

x dword 0

count dword ?

JumpCount DWORD ?

Distance DWORD 0

DistanceCount DWORD 0

DinoDraww DWORD 0

DinoDrawC DWORD 0 ;0 for false, 1 for true

CactusDraw DWORD 0 ; 0 for false, 1 for true

KEYS struct

jump DWORD ?

crouch DWORD ?

KEYS ends

Cactus struct

pos POINT<>; 0 x / 4 y

center POINT<> ;8 x/ 12 y

RAD DWORD dinoRad ; 16

IsScreen DWORD 0;20

Cactus ends

Character struct

pos POINT<> ;0 x /4 y

drc POINT<> ; 8- 1/ 12 - 2

keyboard KEYS<> ; jump 16 / crouch 20

isAlive DWORD ALIVE ;24

isJump DWORD STANDING ;28

JUMP\_LVL DWORD JUMP\_POWER ;32

center POINT<>; 36 x / 40 y

RAD DWORD dinoRad; 44

Character ends

Images struct

dino Img<>

dinoUP Img<>

dinoDOWN Img<>

background Img<>

cactus Img<>

dinof Img<>

gameover Img<>

restarttt Img<>

Images ends

res Images<>

score db "score-reached.wav", 0

hit db "hit.wav",0

jump\_sound db "button-press.wav",0

dino db "pics\dino.png",0

dinoUP db "pics\leg\_up.png",0

dinoDOWN db "pics\leg\_down.png",0

background db "pics\background.png",0

cactus1 db "pics\cactus1.png",0

dinof db "pics\dinof.png", 0

gameover db "pics\gameover.png",0

restarttt db "pics\restart.png", 0

distance db "0", 0

dinod Character<<DINO\_X\_ONSCREEN,DINO\_Y\_ONSCREEN>,<1,0>,<VK\_SPACE,VK\_DOWN>, <>>

michsol Cactus <<1200 - dinoRad,DINO\_Y\_ONSCREEN >, <>>

michsol2 Cactus <<1200, DINO\_Y\_ONSCREEN>, <>>

קישורים

ארגון המחשב ושפת סף – ברק גונן, גבהים - <http://data.cyber.org.il/assembly/gvahim_assembly_book.pdf>

קישור לGithub שבו עשיתי את הפרויקט - <https://github.com/computerboy0555/asmproject>

התמונות המקוריות של המשחק מתוך הספריית קוד פתוח של Chromium:

<https://cs.chromium.org/chromium/src/components/neterror/resources/?dr=C&g=0>