

SPACE INVADERS

Omri Levy



ID: 318919867

Teacher: Eldad Kapitolnik

10th grade

Rishonim High School

תוכן עניינים

2.....	מבוא
6.....	נושא העבודה
7.....	אופן ההפעלה
8.....	גרסאות המערכת
9.....	תיעוד והסבר הפתרון
12.....	תרשימי זרימה
17.....	רשימת הפעולות
21.....	קוד התכנית וקובץ הריצה
22.....	דוגמאות הרצה
24.....	סיכום אישי

מבוא

קובץ ההפעלה: MAIN.EXE

קובץ הקוד הראשי: main.exe

קבצים נלווים:

הקבצים הבאים הם קבצי פרוצדורות, הם הכרחיים להידור התכנית ללא תקלות:

- drawBitmap.asm
- deleteBitmap.asm
- moveSpaceship.asm
- shootNew.asm
- updateBullet.asm
- checkForHit.asm
- drawAliens.asm
- findMax.asm
- findMin.asm
- updateAliens.asm
- updateGame.asm

סביבת העבודה

העבודה בוצעה בעזרת המדהר Turbo Assembler והמאגד (לינקר) Turbo Linker:

```
E:\ASSEMBLY\TASM\SOURCE\SPACE_~1>E:\Assembly\TASM\BIN\TASM.EXE main.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:   main.asm
Error messages:   None
Warning messages: None
Passes:           1
Remaining memory: 451k
```

המהדר Turbo Assembler מעביר הודעה הכוללת את הפרטים הבאים:

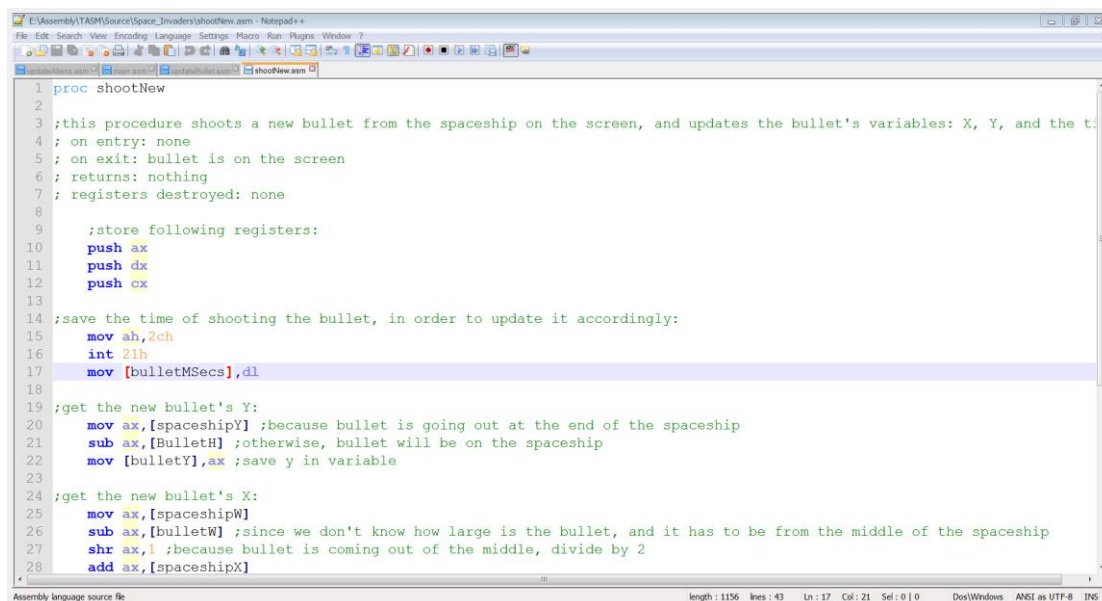
- שם הקובץ המהודר
- הודעות תקלה- במידה ויש.
- הודעות אזהרה- במידה ויש.
- כמה הודעות תקלה היו.
- כמה הודעות אזהרה היו.
- זיכרון פנוי בדיסק

```
E:\ASSEMBLY\TASM\SOURCE\SPACE_~1>E:\Assembly\TASM\BIN\TLINK.EXE main.obj
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International
```

המאגד Turbo linker מאגד את קובץ ה- *.obj, ומעביר הודעה על גירסתו הנוכחית. במידה והמאגד לא ימצא את הקובץ, תצא הודעה שונה שתודיע על הבעיה.

סביבת הפיתוח

סביבת הפיתוח בה השתמשתי לכתיבת הפרויקט הייתה Notepad++:

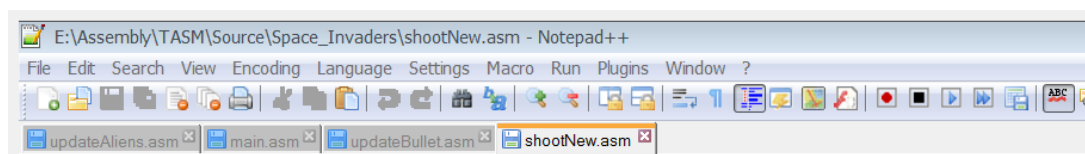


```

1 proc shootNew
2
3 ;this procedure shoots a new bullet from the spaceship on the screen, and updates the bullet's variables: X, Y, and the t:
4 ; on entry: none
5 ; on exit: bullet is on the screen
6 ; returns: nothing
7 ; registers destroyed: none
8
9 ;store following registers:
10 push ax
11 push dx
12 push cx
13
14 ;save the time of shooting the bullet, in order to update it accordingly:
15 mov ah,2ch
16 int 21h
17 mov [bulletMsecs],di
18
19 ;get the new bullet's Y:
20 mov ax,[spaceshipY] ;because bullet is going out at the end of the spaceship
21 sub ax,[BulletH] ;otherwise, bullet will be on the spaceship
22 mov [bulletY],ax ;save y in variable
23
24 ;get the new bullet's X:
25 mov ax,[spaceshipW]
26 sub ax,[bulletW] ;since we don't know how large is the bullet, and it has to be from the middle of the spaceship
27 shr ax,1 ;because bullet is coming out of the middle, divide by 2
28 add ax,[spaceshipX]

```

Notepad++ הוא עורך המיועד לתכנות, ומסוגל לפתוח מגוון רב של סוגי קבצים. בנוסף, בעת הכתיבה בשפת אסמבלי, העורך צובע דברים חשובים בצבעים שונים: שמות של אוגרים, פקודות, הערות ומספרים.



בנוסף, העורך מציע מגוון רב של כלים לעריכת הטקסט (כפי שניתן לראות בתמונה) ומאפשר לערוך כמה קבצים במקביל.

סביבת ההרצה

העבודה הודרה והופעלה בעזרת מדמה מערכת (emulator) של מערכת ההפעלה DOS-DOSBox.

The screenshot shows the DOSBox 0.74 window. The title bar reads "DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: ...". The main window has a blue background with white text. The text reads: "Welcome to DOSBox v0.74", "For a short introduction for new users type: **INTRO**", "For supported shell commands type: **HELP**", "To adjust the emulated CPU speed, use **ctrl-F11** and **ctrl-F12**.", "To activate the keymapper **ctrl-F1**.", "For more information read the **README** file in the DOSBox directory.", "HAVE FUN!", "The DOSBox Team <http://www.dosbox.com>". Below this, the command prompt shows the following commands and output: "Z:\>SET BLASTER=A220 I7 D1 H5 T6", "Z:\>mount e: e:\", "Drive E is mounted as local directory e:\", "Z:\>e:\", "E:\>cd assembly\tasm\source\space_~1", "E:\ASSEMBLY\tasm\source\space_~1>".

על החלון, כתובים הדברים הבאים (מימין לשמאל):

- הגרסה הנוכחית של DOSBox.
- CPU speed - מהירות המערכת. המשתמש יכול לקבוע את מהירות המערכת.
- Frameskip - כמות הפריימים ש-DOSBox מדלג עליהם לפני שהוא מצייר אחד כלומר, אם מכון ל-3, על כל 4 פריימים שהתכנית תשלח, DOSBox ידפיס אחד. כרגע, מכון ל-0.
- Program - שם התכנית המופעלת כרגע.

נושא העבודה



הפרויקט הוא מעין שחזור למשחק שהוצא לראשונה ביוני 1978, "פולשים מהחלל" ("Space Invaders").

לאחר הוצאתו, המשחק נהיה פופולרי תוך זמן קצר, ויצא בהמון גרסאות נוספות. היום, אני גאה להציג את גרסת האסמבלי שלי למשחק המפורסם.

מטרת המשחק

במשחק, הינך משחק בתור חללית, שמטרתה להגן על כדור הארץ מפני פולשים מהחלל החיצון. עליך להרוג את כל החייזרים לפני שיגיעו אליך ויחסלו אותך! עליך להזיז את החללית לאורך המסך, ולירות מתותח הלייזר שלה כדי לפגוע בכל החייזרים.

בגרסתו הנוכחית של המשחק, אלו הן ההוראות השלמות אליו. אולם בהמשך, יהיו רמות נוספות וקשות יותר, חייזרים מסוגים שונים, החייזרים יוכלו להשיב אש, והחללית תוכל להשתדרג, ועוד המון... השמיים הם הגבול!

הנגיעה שלי במשחק

בגרסתו הנוכחית, החייזרים הם החייזרים הלבנים שכולנו מכירים ואוהבים מהמשחק הישן. אולם, את היריה ואת החללית בחרתי לעצב בעצמי.

בעתיד, יוספו רקעים שונים, חייזרים שונים, ואפשרויות משחק שונות שלא היו קיימות במשחק הישן- השמיים הם הגבול!

אופן ההפעלה

מקשי ההפעלה:

השתמש במקשי החצים כדי להזיז את החללית לאורך המסך, ובמקש הרווח כדי לירות בחיזרים.

לחץ על מקש Esc כדי לצאת מהתכנית.

הערות:

- החללית לא יכולה לירות יותר מפעם אחת באותו הזמן- עליך לחכות לפגיעה או ליציאתה של היריה מהמסך, ורק אז לירות מחדש. כל ניסיון אחר לא יצליח, אך גם לא יפגע במערכת.
- יש להשתמש רק במקשים שצוינו לעיל. כל ניסיון להשתמש במקשים אחרים לא יעבוד, אך גם לא יפגע במערכת.

כדי לנצח במשחק, על המשתמש לפגוע בכל החיזרים שנמצאים על המסך, לפני שיגיעו לקו הגובה של החללית.

המשתמש יפסיד במשחק אם החיזרים יגיעו לקו הגובה של החללית.

התכנית תצא בעת ניצחון, הפסד או בקשה ליציאה (Esc) של המשתמש.

גרסאות המערכת

הגרסה הנוכחית (1.0.0):

- מנוע המשחק עובד, החייזרים יורדים לכיוון החללית, והחללית יכולה לירות בהם ולפגוע.
- המערכת חסינה מפני טעויות משתמש, שום מקש שילחץ ואינו בעל משמעות במשחק לא יפריע למהלך התכנית.
- התכנית מסוגלת לזהות מתי המשחק נגמר, ולצאת.

גרסה 1.1.0:

- המערכת תדע להודיע בעת ניצחון או הפסד של המשתמש
- המערכת תעשה שימוש בקול

גרסה 1.2.0:

- יהיה תפריט ראשי, אליו המשתמש יגיע בכניסה לתכנית, בעצירת המשחק, בניצחון, או בהפסד. התפריט יכלול שלוש אפשרויות: "שחק", "הוראות", או "צא".

גרסה 1.3.0:

- למשחק יהיו רמות נוספות
- למשחק יהיה סופר ניקוד
- למשחק יהיו רקעים שונים

תיעוד והסבר הפתרון

לוגיקה כללית

בעת כתיבת התכנית, היה צורך לקיים כמה תהליכים במקביל במקביל: להזיז את היריה, החייזרים, והחללית.

לשם כך, חילקתי את הפרוצדורות לפי עצמים- כל תהליך שהיה צריך להתבצע קיבל פרוצדורה אחרת. כך, ניתן היה להתנות את הקריאה לפרוצדורה, ואם אין בה צורך להמשיך לתהליכים הבאים. למשל, לא לקרוא לפרוצדורה שתזיז את החללית אם אף מקש לא נלחץ.

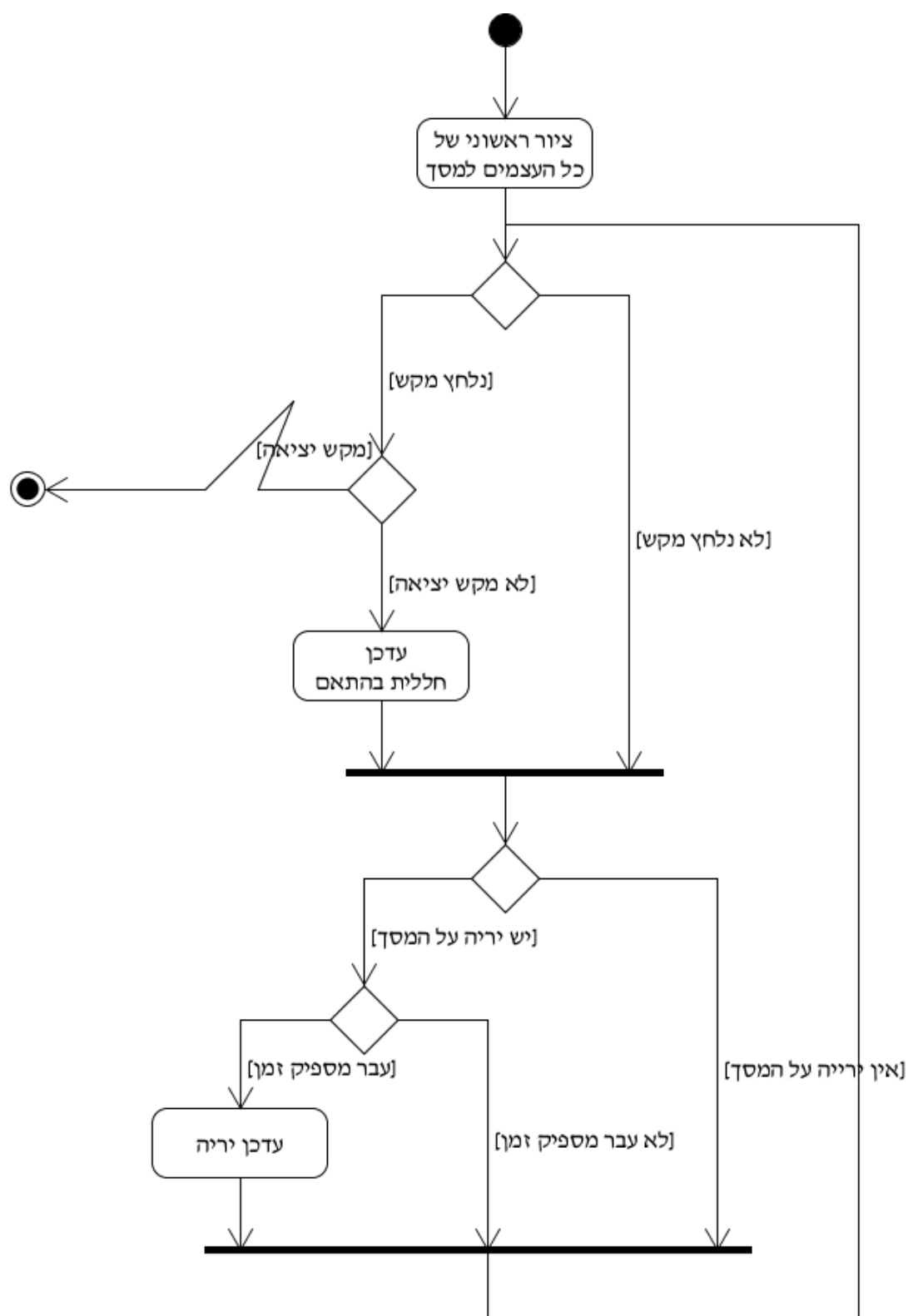
בנוסף, הייתי צריך להשהות את התזוזה של כל עצם במסך בעקבות מהירות המעבד. לשם כך, כל פרודצדורה השתמשה במשתנים קבועים של זמן, ובדקה בעצמה האם עבר מספיק זמן כדי שהיא תפעל. אם לא, הפרוצדורה לא תבצע את עבודתה.

התכנית עבדה בשיטה בה היא נמצאת בלולאה אינסופית, והיא יוצאת ממנה רק בניצחון, הפסד, או בקשה מהמשתמש.

בסופו של כל "סיבוב" שהתכנית מבצעת, כל העצמים שעודכנו מודפסים מחדש למסך.

ניתן יהיה להסביר זאת בצורה הטובה ביותר בעזרת תרשים זרימה:

(ראה עמוד הבא)



זוהי דוגמא לתכנית שמפעילה חללית שיורה, ויוצאת כאשר נלחץ מקש היציאה. ניתן לראות כי כל התהליכים מותנים, וכך, אף תהליך לא עוצר את פעילות התכנית השלמה, כלומר המעגלים שהיא מבצעת.

אחסון הזכרון של התכנית

- (א) החללית- לחללית רק נתוני ה-Bitmap הבסיסיים מאוחסנים בזיכרון: הפרטים (הציור מאוחסן בזיכרון), ה-X וה-Y (של הפינה השמאלית העליונה), האורך, והרוחב. כל נתון כזה הוא משתנה בגודל word.
- (ב) היריה- כמו לחללית, ליריה נתוני ה-Bitmap הבסיסיים שמורים בזיכרון.
- (ג) החייזרים- הפרטים, האורך והרוחב זהים עבור כל החייזרים. אולם, מכיוון שיש מספר רב של חייזרים, נתוני ה-X של כל החייזרים שמורים במערך אחד, ונתוני ה-Y במערך אחר, כל איבר במערכים בגודל word. בנוסף, שמור מונה של כמות החייזרים, בגודל byte.
- חייזר שנפגע מסומן בזיכרון על ידי כך שמתשני ה-X וה-Y שלו הם 0. כל הפרוצדורות המעדכנות יודעות לדלג על חייזרים כאלו.
- (ד) דגלים- המערכת נעזרת בדגלים רבים כדי לבצע את המשימות והתנאים:
1. דגל היריה- $1 = \text{יש ירייה על המסך}$, $0 = \text{אין ירייה על המסך}$.
 2. דגל המשחק- $0 = \text{המשחק ממשיך}$, $1 = \text{המשחק נגמר}$.
 3. כיוון החייזרים- $1 = \text{ימינה}$, $0 = \text{שמאלה}$.
- (ה) משתני השהייה- כל פרוצדורה שומרת משתנים של יחידות הזמן בהן היא משתמשת להשהייה של העצמים שהיא מעדכנת:
- א. מאיות לעדכון ה-X של החייזרים
 - ב. מאיות לעדכון ה-Y של החייזרים
 - ג. שניות לעדכון ה-Y של החייזרים
 - ד. מאיות לעדכון היריה

עבודה עם גרפיקה

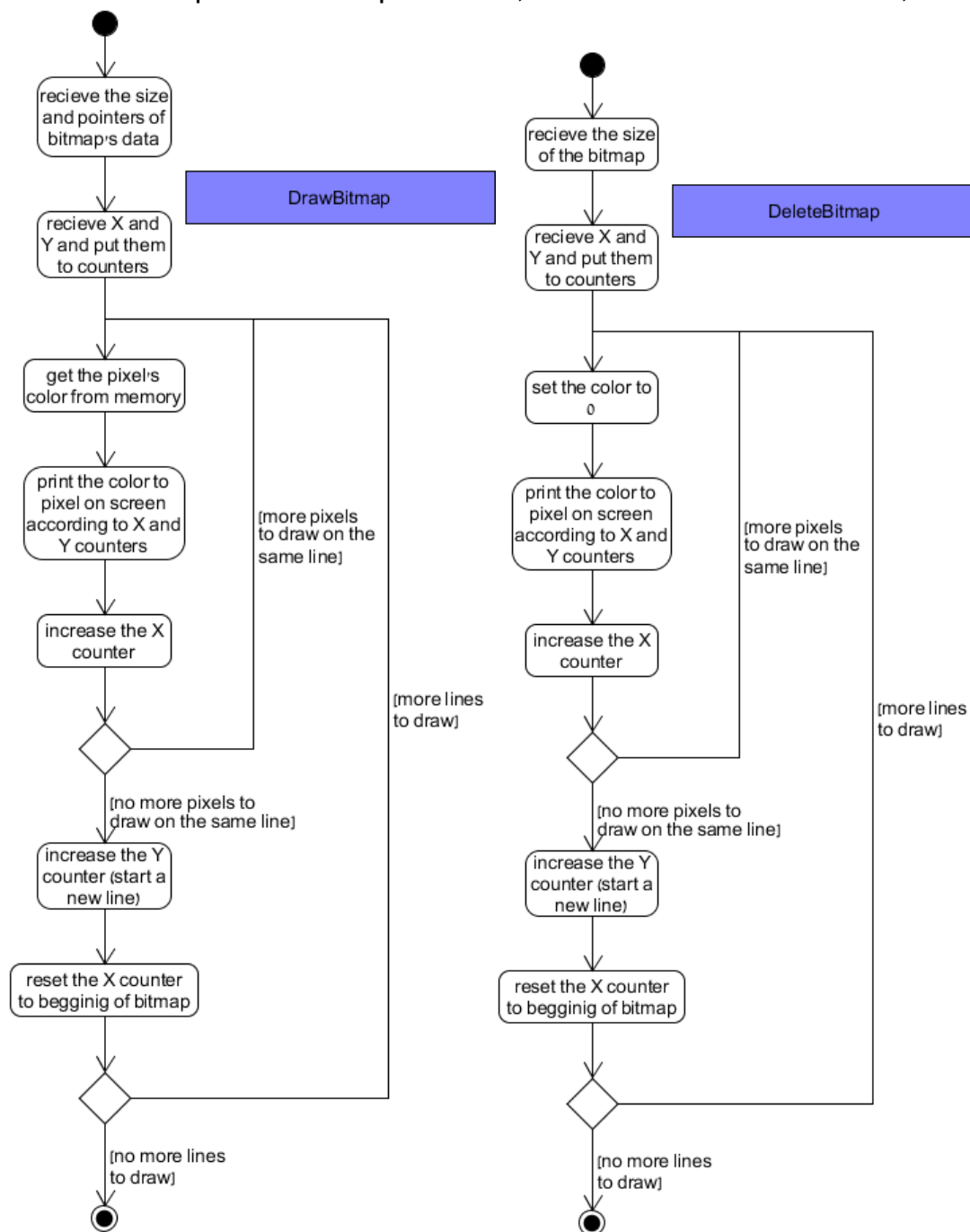
לאורך כל התכנית, כל התנאים והעדכונים נעשים אך ורק על המשתנים בזיכרון, קיים בידוד מוחלט בין לוגיקה לגרפיקה.

לכן, העבודה עם הגרפיקה היא פשוטה. ראשית, קיימת הפרוצדורה הבסיסית, שמקבלת את פרטי העצם ומציירת אותו על המסך. לאחר מכן, קיימות שתי פעולות שמתטרתן היא להקל על השימוש בפרוצדורה הבסיסית לציור העצם: פרוצדורה המקבלת את כל פרטי החייזרים ומציירת אותן, ופרוצדורה שמעדכנת את היריה ויחד עם זאת מציירת אותה במקומה החדש.

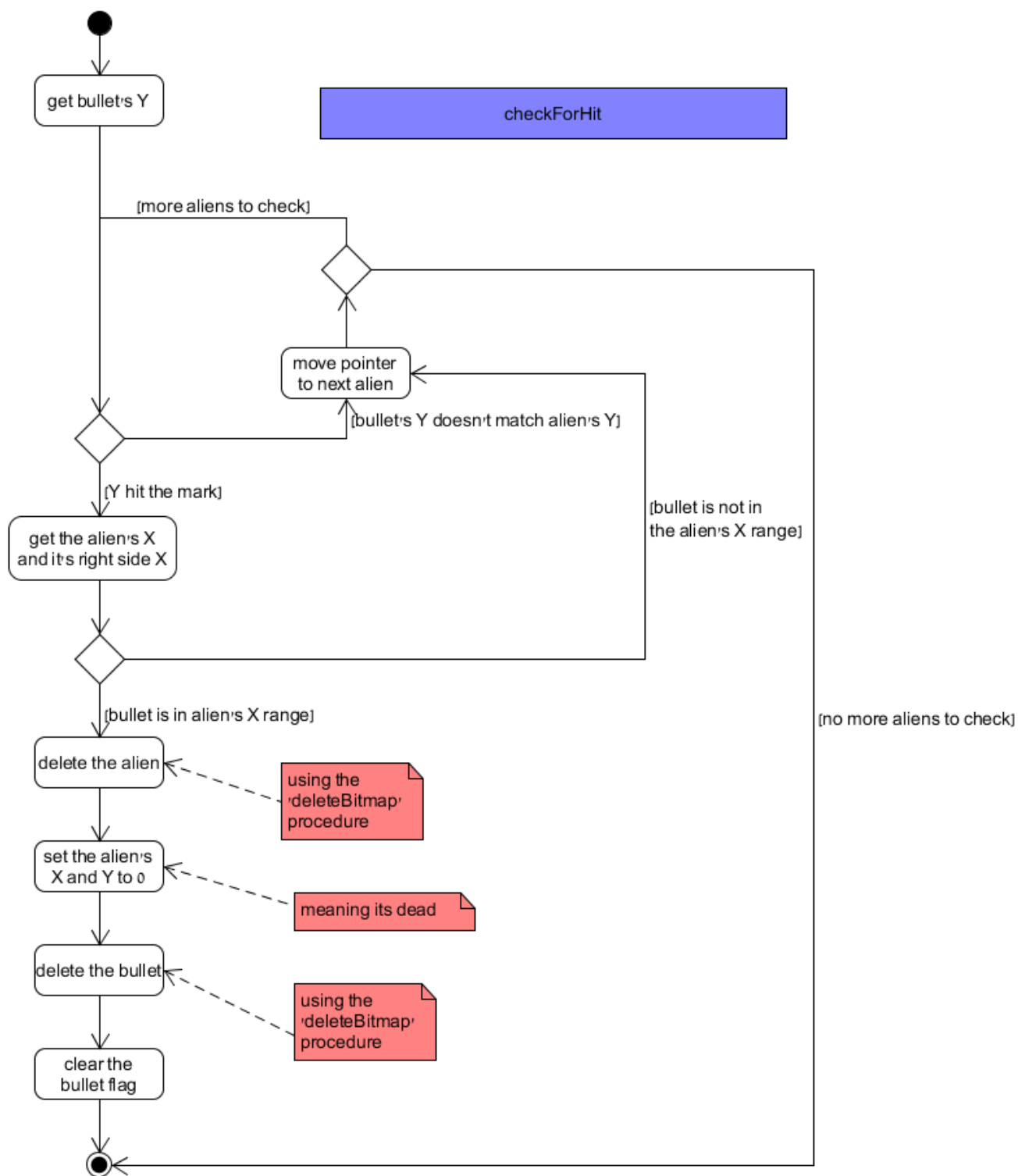
כל שאר הקטעים בתכנית לא לוקחים חלק בגרפיקה של המשחק.

תרשימי זרימה

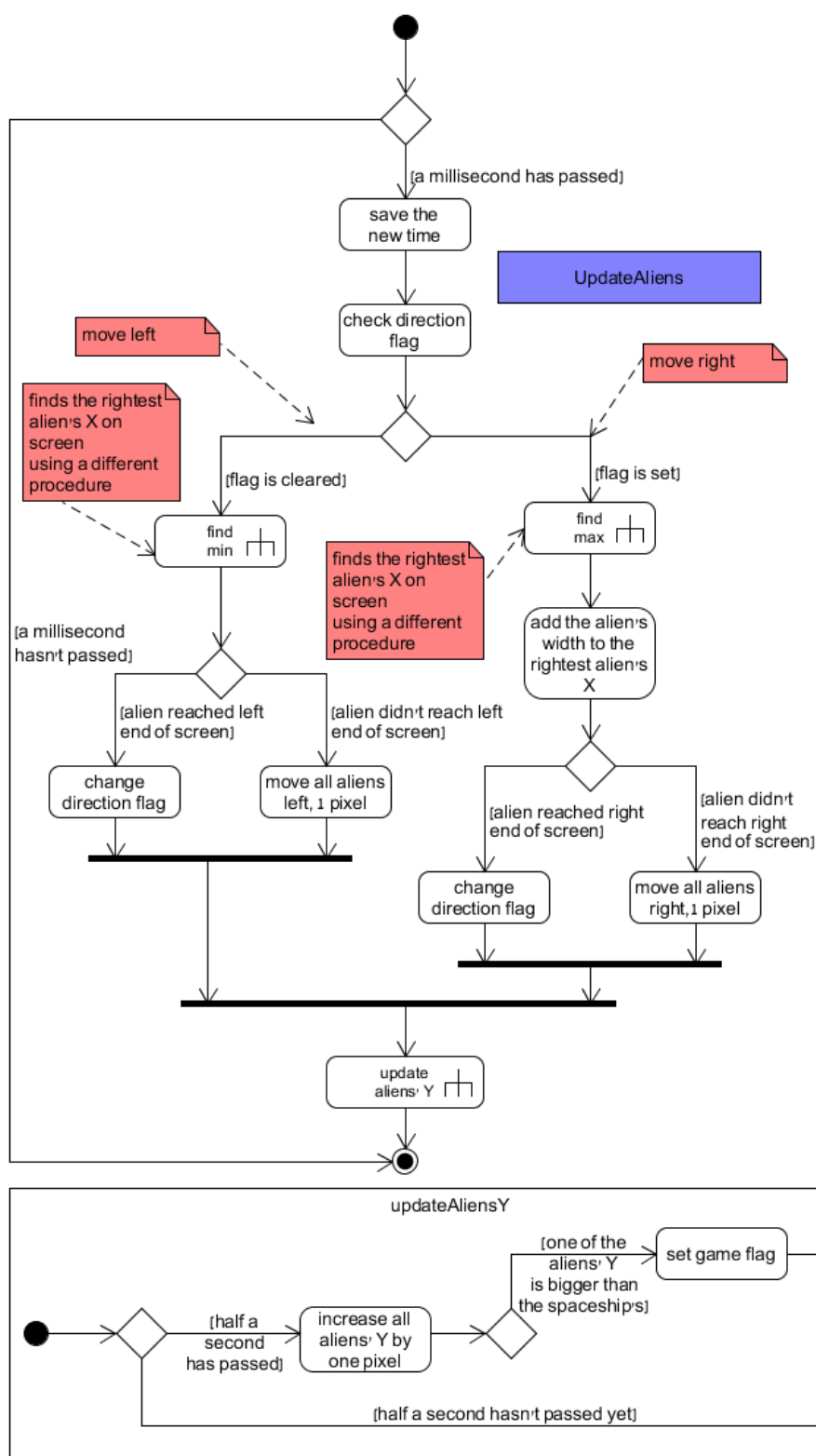
ראשית, שתי הפרוצדורות הבסיסיות ביותר, שנעשה בהן שימוש רב לאורך התכנית:



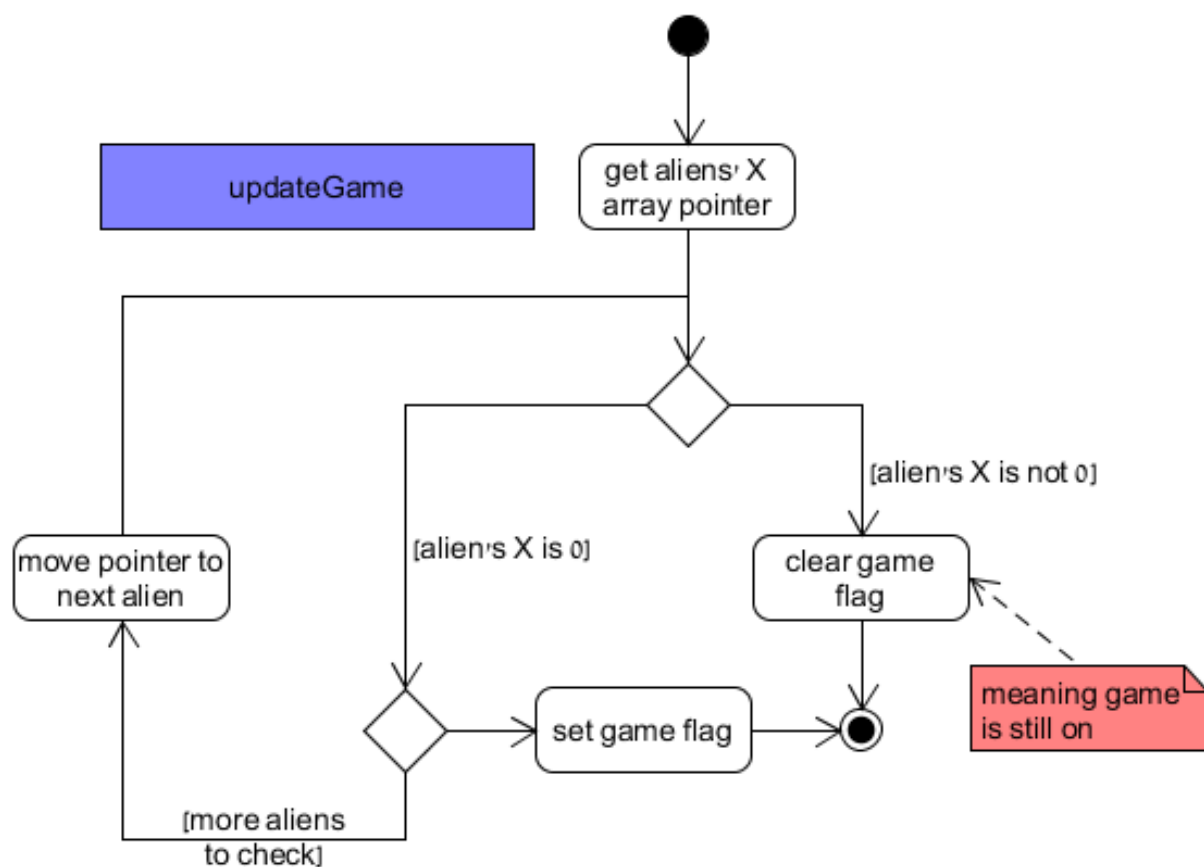
שתי הפרוצדורות כמעט זהות: תפקיד הראשונה (drawBitmap) הוא לצייר תמונה של עצם על המסך, תפקיד השנייה (deleteBitmap) הוא למחוק עצם מסוים מהמסך.



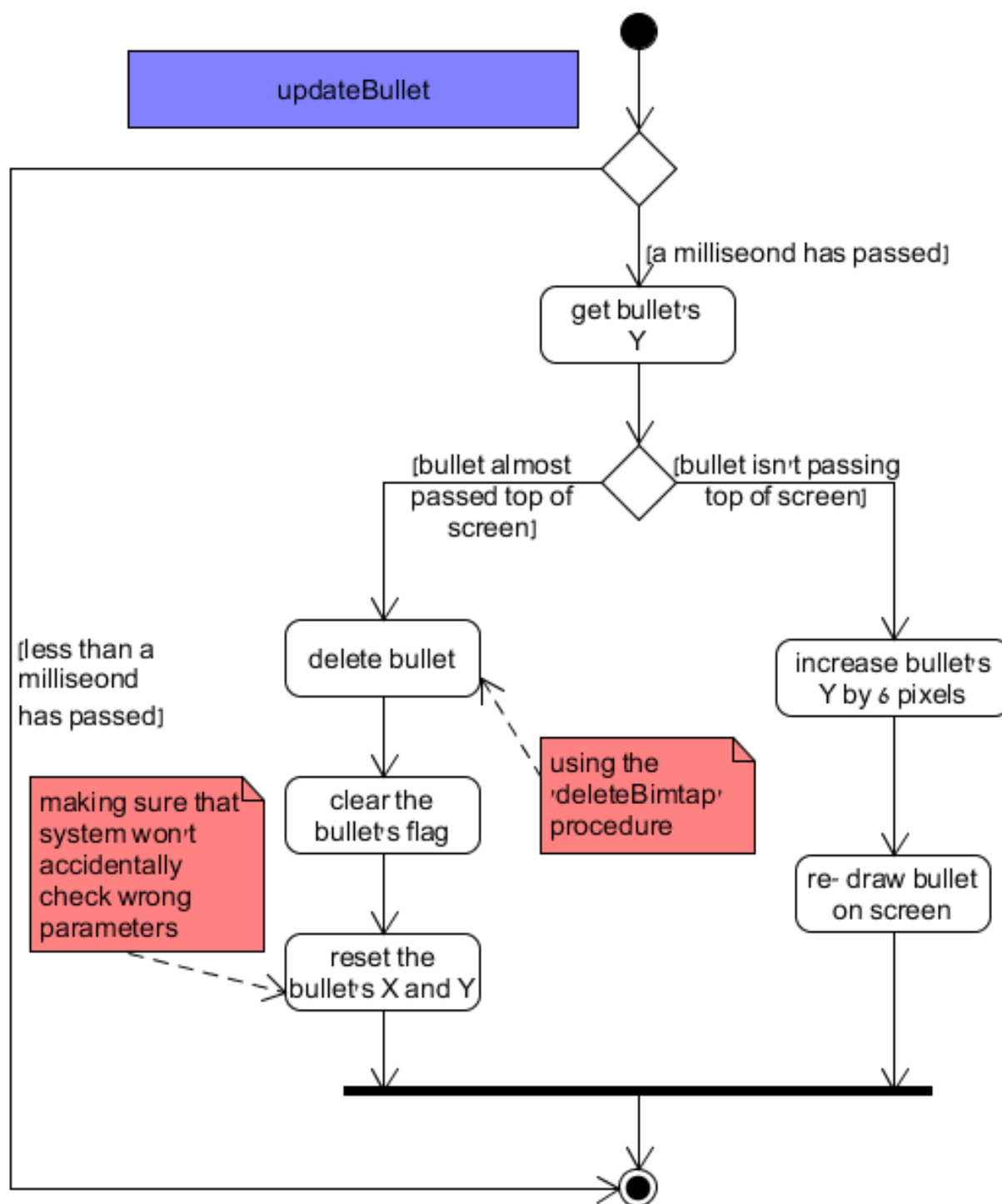
הפרוצדורה checkForHit בודקת האם הייתה פגיעה בחיזור. הפרוצדורה גם מעדכנת את המשתנים, וגם מדפיסה את העדכונים למסך.



הפרוצדורה `updateAliens` מזיזה את החייזרים ימינה או שמאלה, ולאחר מכן קוראת לפרוצדורה נוספת שמורידה את החייזרים למטה. אם אחד מהחייזרים עובר את החללית, היא מעדכנת את דגל המחשק. שימו לב! הפרוצדורה רק מעדכנת משתנים, לא מדפיסה את החייזרים על המסך!



הפרוצדורה מעדכנת את דגל המשחק- אם הוא 0, המשחק ממשיך, ואם אחד, המשחק מפסיק. אולם, הפרוצדורה רק מעדכנת את הדגל אם המשתמש ניצח- היא לא בודקת האם המשתמש הפסיד.



הפרוצדורה updateBullet מעלה את הכדור למעלה לאורך המסך, ואם הוא עובר את קצה המסך, היא מוחקת אותו ומעדכנת את הדגל בהתאם. הפרוצדורה גם מעדכנת את המשתנים וגם מדפיסה את השינויים למסך.

רשימת הפעולות

drawBitmap

לוקחת פרטי תמונה, גודל התמונה, ואת ה-X וה-Y, ומדפיסה את התמונה למסך.

- טענת כניסה: הפניה לפרטי התמונה בזכרון
- ה-X בו יש להדפיס את התמונה
- ה-Y בו יש להדפיס את התמונה
- רוחב התמונה
- אורך התמונה
- טענת יציאה: התמונה שפרטיה נדחפו למחסנית מודפסת למסך.
- מחזירה: כלום
- אוגרים שנהרסים: אף אחד

deleteBitmap

לוקחת את גודל התמונה בנוסף ל-X וה-Y, ומוחקת את התמונה מהמסך.

- טענת כניסה: ה-X בו יש למחוק את התמונה
- ה-Y בו יש למחוק את התמונה
- רוחב התמונה
- אורך התמונה
- טענת יציאה: התמונה שפרטיה נדחפו למחסנית נמחקת מהמסך.
- אוגרים שנהרסים: אף אחד

moveSpaceship

מקבלת דגל המורה על כיוון, ומזיזה את החללית בהתאם. הפרוצדורה משתמשת בפעולת `drawBitmap`. העזר `drawBitmap`.

- טענת כניסה: דגל ב-AL, 1=שמאלה, 0=ימינה.
- טענת יציאה: החללית זזה על המסך, והמשתנים שלה עודכנו.
- אוגרים שנהרסים: אף אחד

shootNew

יורה יריה חדשה מקצה החללית. הפרוצדורה משתמשת בפעולת העזר `drawBitmap`.

- טענת כניסה: כלום
- טענת יציאה: היריה מופיעה על המסך בקצה החללית, ומשתני הזמן, ה-X וה-Y מעודכנים בהתאם.
- אוגרים שנהרסים: אף אחד

updateBullet

הפרוצדורה בודקת האם עבר מספיק זמן כדי להזיז את היריה במסך- אם לא, הפרוצדורה לא תפעל. אם כן, היא מזיזה את היריה לחלקו העליון של המסך. אם היריה הגיעה לסוף המסך, הפרוצדורה תמחק אותה. הפרוצדורה משתמשת בפעולות העזר `drawBitmap` ו-`deleteBitmap`.

- טענת כניסה: כלום
- טענת יציאה: היריה זזה קדימה, ומשתני הזמן, ה-X וה-Y עודכנו בהתאם.
- אוגרים שנהרסים: אף אחד

checkForHit

הפרוצדורה בודקת האם היריה פגעה בחייזר. אם לא, היא לא עושה כלום. אם כן, היא מוחקת את החייזר והיריה, ומעדכנת את המשתנים בהתאם. הפרוצדורה משתמשת בפעולת העזר `deleteBitmap`.

- טענת כניסה: הגובה של החייזר
הרוחב של החייזר
הפניה למערך נתוני ה-X של החייזרים
הפניה למערך נתוני ה-Y של החייזרים
כמה חייזרים נמצאים במשחק (כולל חייזרים שנפגעו)
- טענת יציאה: אם הייתה פגיעה, החייזר והיריה נמחקו מהמסך, וה-X וה-Y התאפסו.
- אוגרים שנהרסים: אף אחד

drawAliens

הפרוצדורה מציירת את החייזרים שלא נפגעו על המסך. הפרוצדורה משתמשת בפעולת העזר `drawBitmap`.

- טענת כניסה: כמה חייזרים נמצאים במשחק (כולל חייזרים שנפגעו)
הפניה לפרטי החייזר בזיכרון
הפניה למערך נתוני ה-X של החייזרים
הפניה למערך נתוני ה-Y של החייזרים
הרוחב של החייזר
הגובה של החייזר
- טענת יציאה: החייזרים שלא נפגעו מודפסים למסך. אין עדכון למשתנים.
- אוגרים שנהרסים: אף אחד

findMax

הפרוצדורה מוצאת את המספר הגבוה ביותר מתוך מערך שקיבלה.

- טענת כניסה: כמות האיברים לבדיקה
- הפניה למערך האיברים
- טענת יציאה: מחזירה את המספר הגבוה ביותר באוגר AX.
- אוגרים שנהרסים: אף אחד

הערות:

- הפרוצדורה מתייחסת לכל איבר במערך בגודל word!
- הפרוצדורה מתייחסת למספרים במערך כאל לא חתומים

findMin

הפרוצדורה מוצאת את המספר הנמוך ביותר מתוך מערך שקיבלה.

- טענת כניסה: כמות האיברים לבדיקה
- הפניה למערך האיברים
- טענת יציאה: מחזירה את המספר הנמוך ביותר באוגר AX.
- אוגרים שנהרסים: אף אחד

הערות:

- הפרוצדורה מתייחסת לכל איבר במערך בגודל word!
- הפרוצדורה מתייחסת למספרים במערך כאל לא חתומים.
- הפרוצדורה לעולם לא תחזיר 0! היא תוכננה לבדוק מה הוא ה-X הנמוך ביותר של אחד מהחייזרים, אולם חייזר שה-X שלו הוא 0 הוא חייזר שנפגע. לכן, הפרוצדורה תדלג עליו ולעולם לא תחזיר 0.
- אם הפרוצדורה תחזיר ב-AX את המספר 0FFFFh, יכולים להיות רק שני מספרים במערך- 0FFFFh או 0.

updateGame

הפרוצדורה בודקת האם נשארו חייזרים חיים. אם לא, היא מעדכנת את דגל המשחק בהתאם (1= סיום משחק, 0= המשך). במילים אחרות, הפרוצדורה בודקת האם המשתמש ניצח.

- טענת כניסה: הפניה למערך נתוני ה-X של החייזרים
- כמה חייזרים נמצאים במשחק (כולל חייזרים שנפגעו)
- טענת יציאה: דגל המשחק מעודכן
- אוגרים שנהרסים: אף אחד

updateAliens

הפרוצדורה מזיזה את החייזרים על המסך, בתנאי שעבר מספיק זמן. אם החייזרים עברו מתחת לגובה החללית, הפרוצדורה מעדכנת את דגל המשחק (משום שהמשתמש הפסיד). אם החייזרים הגיעו לאחד מקצוות המסך, הפעולה משנה את כיוונם על ידי עדכון של דגל כיוון החייזרים.

הפרוצדורה משתמשת בפעולת העזר updateAliensY.

- טענת כניסה: כמה חייזרים נמצאים במשחק (כולל חייזרים שנפגעו)
 הפניה למערך נתוני ה-X של החייזרים
 הפניה למערך נתוני ה-Y של החייזרים
 הרוחב של החייזר
 הגובה של החייזר
- טענת יציאה: משתני החייזרים ודגל המשחק עודכנו, והחייזרים שעל המסך זזו.
 בנוסף, דגל כיוון החייזרים מתעדכן.
- אוגרים שנהרסים: אף אחד

updateAliensY

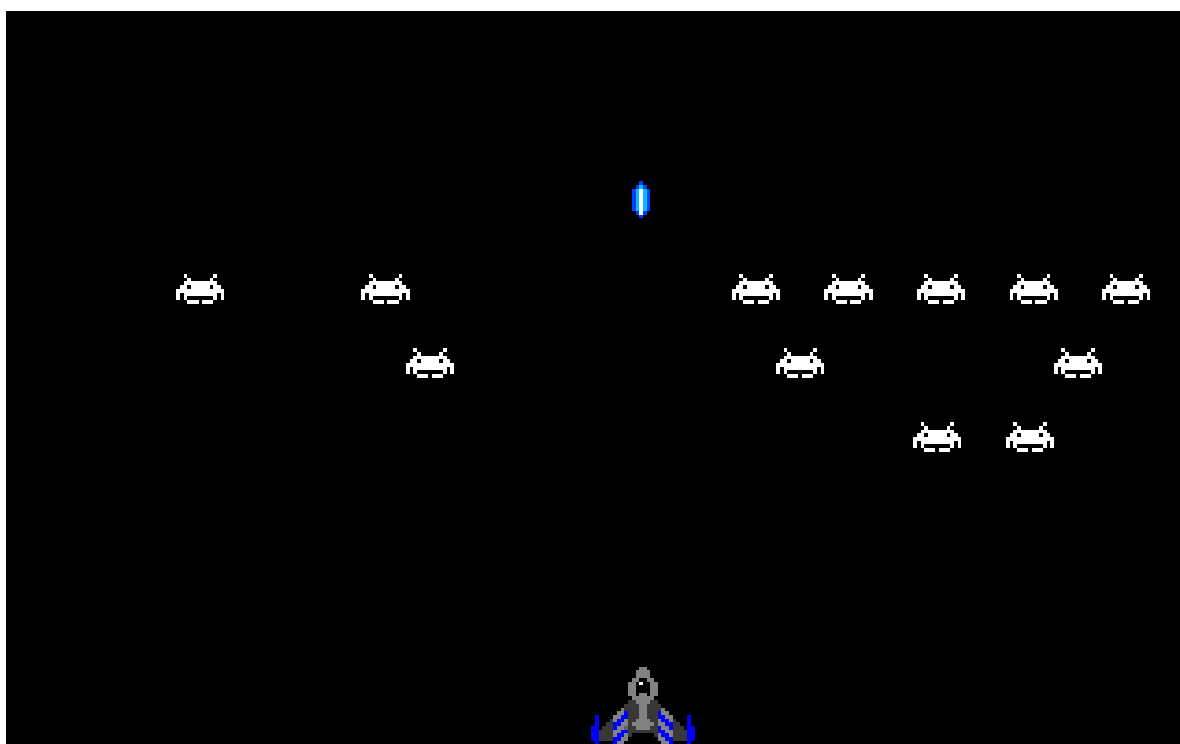
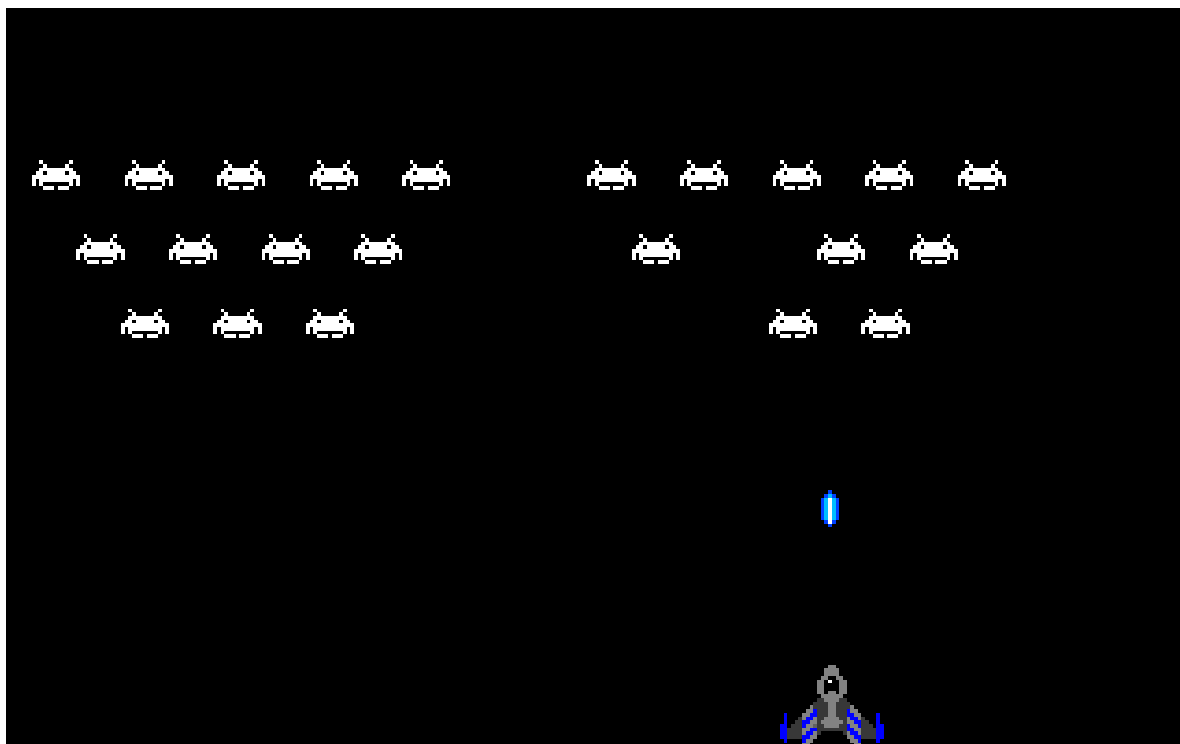
הפרוצדורה מזיזה את החייזרים בתנאי שעבר מספיק זמן. שימו לב- בניגוד לפרוצדורה הקודמת, הפרוצדורה הזו מזיזה את החייזרים למטה בלבד. היא תוכננה כפעולת עזר בשביל הפעולה updateAliens.

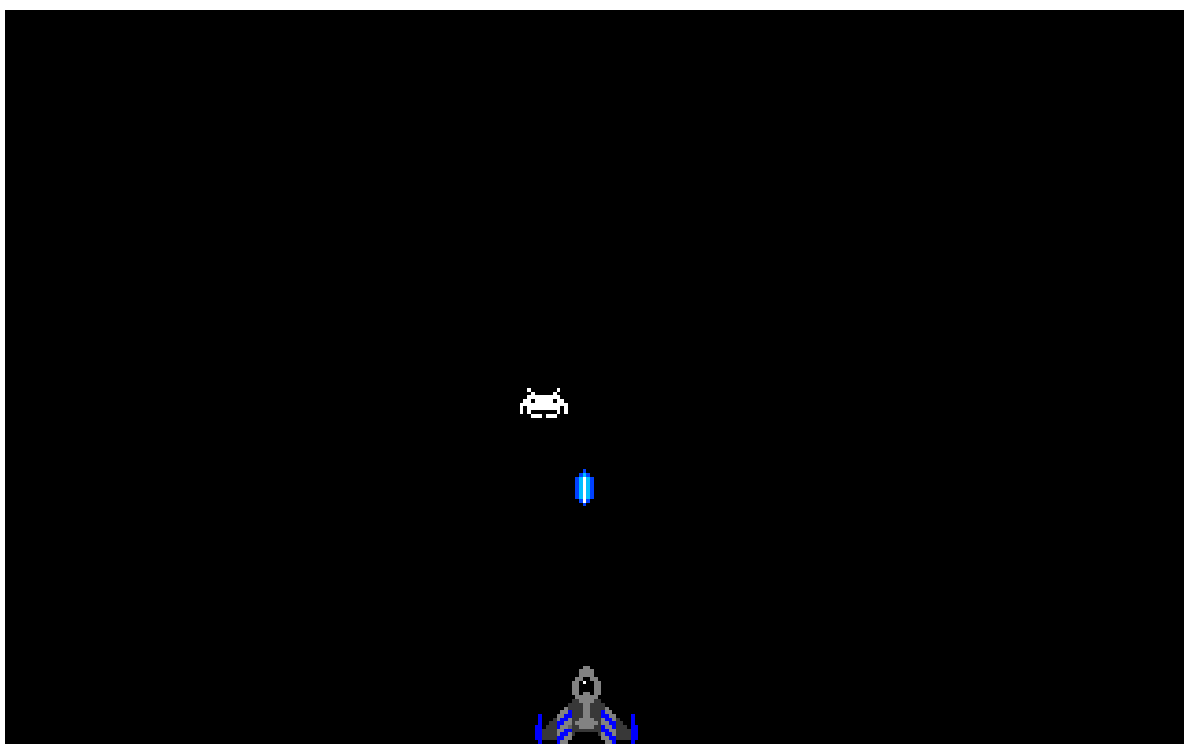
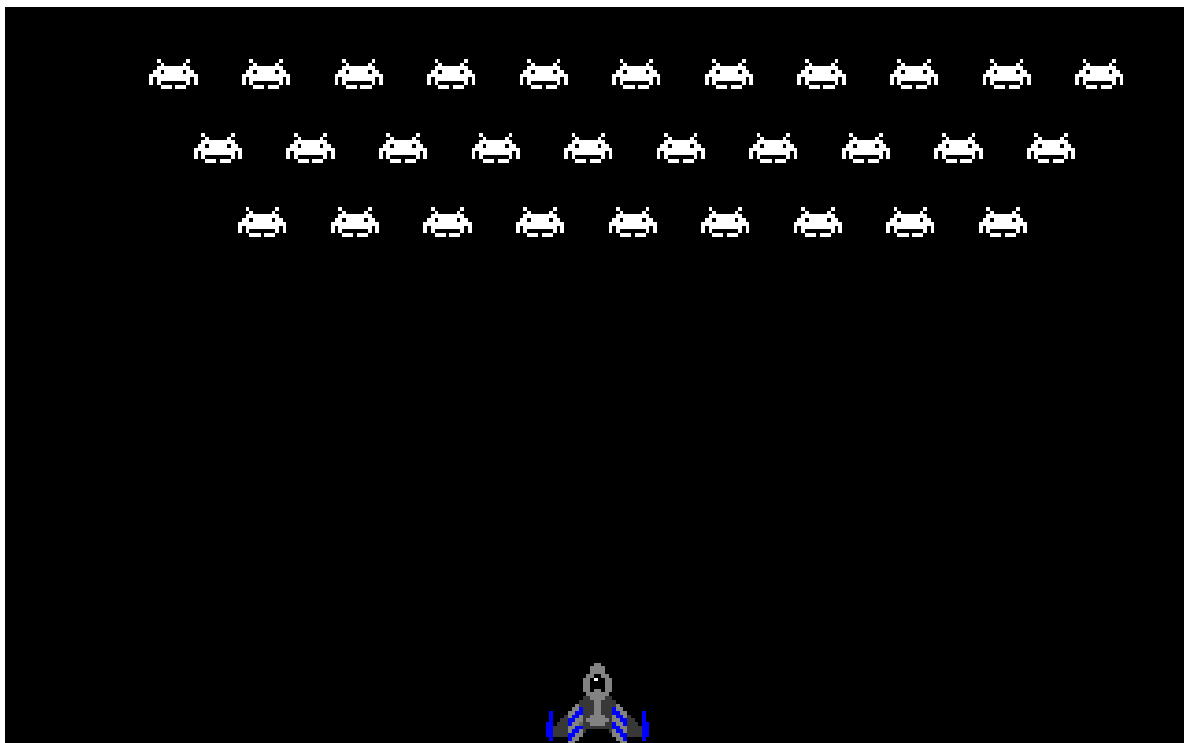
- טענת כניסה: הפניה למערך נתוני ה-Y של החייזרים
 הגובה של החייזר
- כמה חייזרים נמצאים במשחק (כולל חייזרים שנפגעו)
- טענת יציאה: אם עבר מספיק זמן, החייזרים זזו למטה והמתשנים עודכנו בהתאם.
 בנוסף, גם דגל המשחק מתעדכן.
- אוגרים שנהרסים: AX, DX, BX, CX

קוד התכנית וקובץ הריצה

דוגמאות הרצה

כמה דוגמאות הרצה של מסך המשחק:





סיכום אישי

כתיבת המשחק הייתה מהנה מאוד.

נהייתי לכתוב משחק בעצמי- לקבוע את מראה החיזורים והחללית, לשלוט במהירויות במשחק, ברמות הקושי, ובמקשי ההפעלה. בנוסף לכך, הידיעה שכעת יש לי את הכלים להוסיף למשחק כל דבר שארצה נותנת הרגשה של סיפוק.

מעבר לכתיבה, נהייתי מהתגובות של החברים והמשפחה על התוצר הסופי, שהעריכו ותמכו.

לאורך הפרויקט, למדתי איך להשתמש בגרפיקה, ואיך עובדים צגי המסך.

בנוסף, למדתי הרגלי עבודה חדשים, שלדעתי, אשתמש בהם גם בעתיד:

- למדתי שעדיף להפריד לוגיקה מגרפיקה
- למדתי שהפעולות צריכות להיות כמה שיותר כלליות- הן צריכות לקבל כמה שיותר ערכים דרך המחסנית, ולעבוד כמה שפחות על משתנים קבועים בזיכרון, כדי שניתן יהיה לעשות בהן שימושים נוספים.
- למדתי שבעת כתיבת קטעי קוד ארוכים או מסובכים, עדיף לתכנן ראשית בעזרת תרשימי זרימה ורק לאחר מכן לכתוב את הקוד.
- למדתי שבעת כתיבת פרויקטים גדולים, כדאי לכתוב סדר עבודה לפני הכתיבה- מהם החלקים הבסיסיים שיש לכתוב קודם, ומהם החלקים שתלויים באחרים כדי לפעול. למשל, לא אבדוק אם הכדור פגע בחיזורים, אם לא יצרתי את החיזורים.

אולם, מעבר לכתיבה התכנית, למדתי איך להתחיל לעבוד על פרויקט גדול ולא רק משימה שניתנת בכיתה- בסייבר ובחיים.

לסיכום, כתיבת הפרויקט הייתה מהנה מאוד, והתוצר מספק. בנוסף, אני מרגיש שלמדתי דברים רבים במהלך הכנת הפרויקט, בין עם לכתיבת תכניות בעתיד, ובין אם להתנהלות כללית בחיים.