****

**פרויקט סיום באסמבלר**

**Battleships**

**נושא העבודה:** צוללותBattleships -

**שם המגיש:** בן קירשנבאום

**ת"ז של המגיש:** 212334767

**שם המורה:** גלית קלאוס



תוכן עניינים

[נושא הפרויקט ולמה בחרתי בו 3](#_Toc483482379)

[סביבת העבודה 4](#_Toc483482380)

[הוראות המשחק 5](#_Toc483482381)

[תרשים זרימה 6](#_Toc483482382)

[פרוצדורות 7](#_Toc483482383)

[משתנים 12](#_Toc483482384)

[משתנים של שחקנים 12](#_Toc483482385)

[משתנים שמאפשרים בינה מלאכותית 12](#_Toc483482386)

[משתנים להדפסת קבצי BMP 12](#_Toc483482387)

[משתני מחרוזת 13](#_Toc483482388)

[מסכי התכנית 14](#_Toc483482389)

[קוד התכנית 16](#_Toc483482390)

[סיכום ורפלקציה 17](#_Toc483482391)

[תודות 17](#_Toc483482392)

# נושא הפרויקט ולמה בחרתי בו

בחרתי שבפרויקט שלי אכתוב משחק צוללות באסמבלי.  
בחרתי בנושא זה כי ידעתי שזה משחק שלא דורש יותר מדי גרפיקה (נושא שמאד משעמם אותי) וידרוש בעיקר סיבוכיות אלגוריתמית, אני מאד נהנה לאתגר את עצמי ולמצוא דרכים, או לייעל דרכים לעשות פעולות שונות, בקיצור, להגיע מנקודה א' לנקודה ב'.

מוקדם יותר השנה, קיבלנו משימה לכתוב בc# תוכנית בסיסית מאד של צוללות, ונהניתי מאד לכתוב אותה ולהוסיף לה כל מיני תכונות מתקדמות, כשגלית הציגה בפנינו את הפרויקט, בחרתי לכתוב שוב את משחק הצוללות, הפעם באסמבלי, כי זכרתי כמה אלגוריתמיקה וחשיבה כללה הכתיבה של המשחק בc#, ורציתי לאתגר את עצמי עוד יותר בפיתוח פרויקט רציני של המשחק בשפה אפילו עוד יותר מסובכת ומאתגרת.

# סביבת העבודה

את הפרויקט כתבתי בסביבת העבודה notepad++, השתמשתי בtasm לביצוע פעולת הקומפילציה על הקוד, tlink לביצוע link ובtd כדי לדבג, למצוא ולפתור, בעיות בפרויקט.  
השתמשתי בdosbox כאימולטור למעבד 8086, המסוגל להריץ קבצי asm.  
השתמשתי גם בתוכנה הבסיסית של windows, צייר ובעורך התמונות paint.net ליצירה ועיצוב של התמונות המוצגות למסך.  
בנוסף, השתמשתי בדיסק און קי ובgithub לשמירת גרסאות שונות וקבצי עזר של הפרויקט.

# הוראות המשחק

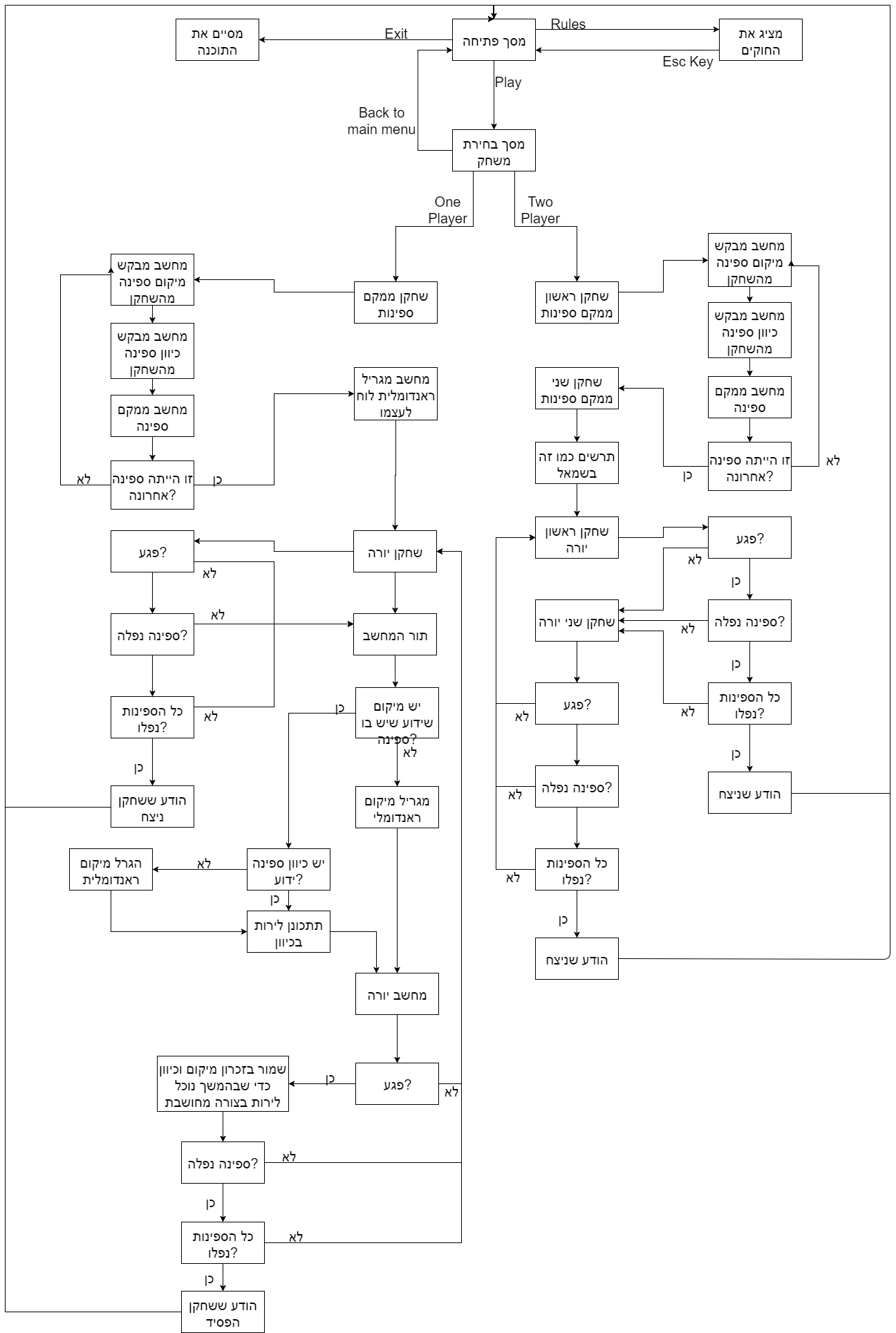
כדי להריץ את התוכנית צריך להריץ את הקובץ game.exe.  
בנוסף, בספריית ההרצה צריכים להמצא גם שאר הקבצים שבתוך battleships.rar, לשם כך יש לגרור את קובץ הrar לתיקית ההרצה, ללחוץ עליו קליק ימני ואז extract here.  
  
מטרת המשחק היא להפיל את כל הצוללות של היריב שלך.  
איך עושים את זה?  
תחילה, כל אחד מהשחקנים משתמש במקלדת כדי למקם את הצוללות שלו על המסך, חשוב מאד לקרוא את מה ההוראות שנותן לך המחשב (באנגלית), כי רק כך תוכל לדעת בוודאות שאתה עושה הכל נכון.  
לאחר מכן, השחקנים בתורות יורים זה לכיוון הצוללות של זה, מנסים לפגוע בהן, כשצוללת נפגעת, המחשב מסמן זאת ומודיע לשחקן, וכשצוללת נופלת המחשב מודיע עם הודעה מתאימה גם כן.  
כך יכול השחקן להפיל את כל הצוללות של היריב.

למשחק יש תפריט פתיחה שכולל אפשרות לשחקן אחד או שניים ואפשרות לצפות בחוקים באנגלית, שימוש בתפריט דורש עכבר, מקלדת, וכמובן ידע בשפה האנגלית כדי להבין את הכתוב.

כחלק מתפריט הפתיחה תוכל לקרוא את החוקים באנגלית שהגדרתי למשחק.  
מסמך הטקסט הבא מכיל את כל החוקים הכתובים בתפריט הפתיחה.



# תרשים זרימה



# פרוצדורות

1. Choose\_Place

* Input: active player number

Board offset

* Output: offset of the chosen place
* Description: this procedure allows the user to choose a position on the board using the keyboard.

1. Check\_legal\_to\_place

* Input: offset of place

board offset

* Output: al = 0 if legal and al = -1 if illegal
* Description: this procedure checks whether or not a position on the board is legal to place a ship part (checks its surroundings)

1. CreatePlayerBoard

* Input: active player number

Player's board offset

offset of an array keeping lengths of the ships to place

* Output: none in the stack

changes the player's board to the given settings

* Description: this procedure places ships on the board according to input from the player

1. CreateComputerBoard

* Input: computer's board offset

offset of an array keeping lengths of the ships to place

* Output: none in the stack

changes the computer's board to a randomly generated board

* Description: this procedure places ships on the board randomly

1. Clear\_board

* Input: none
* Output: none
* Description: this procedure moves to text mode effectively clearing the screen from all text

1. Present\_Board

* Input: active player number (-1 if computer)

Board's offset

* Output: none in the stack

the board presented to the screen

* Description: this procedure prints to the screen a chosen board

1. explainColors

* Input: 0 for placing stage, or 1 for shooting stage
* Output: none in the stack

the needed information presented to the screen

* Description: this procedure explains what each color on the board means

1. player\_turn

* Input: active player number

enemy's board offset

active player's guessing board offset

enemy's array of left ship offset

* Output: none in the stack

al = 1 if won or al = 0 to continue

the guessing board is changed according to the shot

the enemy's left ship array changes according to the shot

* Description: this procedure runs a single player turn, asking the player for a place to shoot and managing that shot

1. mark\_around\_fallen\_ship

* Input: guessing board offset

a number to add to the board to get one of the ship's places (offset place - offset board)

* Output: none in the stack

the guessing board is changed according to the wanted changes (surronding the fallen ship)

* Description: this procedure detects the placements of a ship and marks yellow around it so the player would know it is down

1. Wait\_for\_key\_press

* Input: none
* Output: none
* Description: This procedure waits for a key press

used to allow the players to see the board before the game continues

1. two\_player\_match

* Input: first player's: board guessing board

left ship array

second player's: board

guessing board

left ship array

* Output: none in the stack

changes all given parameters according to the game's actions

* Description: this procedure runs a two player match, creating boards and then running the game

1. reset\_variables

* Input: first player's: board guessing board

left ship array

second player's: board

guessing board

left ship array

* Output: none in the stack

the given variables reset (boards to all BLUE and left ships arrays to the Starting\_LeftShip array, used only by this procedure remembering the ships lengths)

* Description: this procedure resets all variables needed to run a match to their starting values

1. computer\_turn

* Input: player's board offset

computer's guessing board

player's left ships array

last\_hit\_part offset

known\_direction offset

* Output: none in the stack

al = 1 if won and al = 0 to continue

the guessing board is changed according to the shot

the player's left ship array changes according to the shot

* Description: this procedure runs a turn of the computer, using basic thinking to try and make some good shots

1. one\_player\_Match

* Input: first player's: board guessing board

left ship array

second player's: board

guessing board

left ship array

* Output: none in the stack

changes all given parameters according to the game's actions

* Description: this procedure runs a single player vs computer match, from creating the player's board and computer's turn to shooting to ending the game with a fitting message

1. OpenFile

* Input: offset file name
* Output: file's handle
* Description: opens a file

1. ReadHeaderPalette

* Input: file handle

offset to put header

offset to put Palette

* Output: none in the stack

changes the given places the header and palette

moves the reader pointer in the file to the start of the actual image

* Description: Reads Header and Palette

1. CopyPal

* Input: offset to read Palette from
* Output: none in the stack

the colors in the ports are changed

* Description: changes the colors from BGR (assembly color format) to RGB (BMP file color format)

1. CopyBitmap

* Input: file handle
* Output: none in the stack

copying the BMP file from the file to the data segment to the A000 segment, the graphics screen

* Description: prints to the graphic screen the BMP file (after opening file, reading palette and copying it)

1. CloseFile

* Input: file handle
* Output: none
* Description: closes an open file

1. PrintBMP

* Input: offset of file's name
* Output: none in the stack

moves to graphics mode and changes the screen to the given BMP file

* Description: this procedure prints to the screen a BMP file, doing all the things needed.

opens the file, reads header, reads palette, copies the palette, copies the BMP, and closes the file

1. MainMenu

* Input: none
* Output: none
* Description: this procedure runs the main menu, continuing from there to other menus and the game itself, or exiting using the exit button

1. PlayMenu

* Input: none
* Output: none
* Description: this procedure runs the play menu, continuing from there to games, or going back to the main menu

1. RulesMenu

* Input: none
* Output: none
* Description: this procedure runs the rules interface

# משתנים

## משתנים של שחקנים

|  |  |  |
| --- | --- | --- |
| שם משתנה | סוג / גודל | מטרה / שימוש |
| Player1Board | מערך | שומר את הערכים בלוח השחקן הראשון |
| Player2Board | מערך | שומר את הערכים בלוח השחקן השני |
| Player1Guessing | מערך | שומר את הניחושים של השחקן הראשון |
| Player2Guessing | מערך | שומר את הניחושים של השחקן השני |
| Player1LeftShip | מערך | שומר כמה חלקים מכל צוללת של השחקן הראשון עוד לא נפגעו |
| Player2LeftShip | מערך | שומר כמה חלקים מכל צוללת של השחקן השני עוד לא נפגעו |
| Starting\_LeftShip | מערך | מערך זה אינו משתנה, הוא שומר את הגדלים ההתחלתיים של הצוללות כדי שנוכל לאפס את המערכים של השחקנים בתחילת משחק. |

## משתנים שמאפשרים בינה מלאכותית

|  |  |  |
| --- | --- | --- |
| שם משתנה | סוג / גודל | מטרה / שימוש |
| Last\_Hit\_Part | מילה | זוכר את הoffset של המיקום האחרון שנורה ונמצאה בו צוללת, אם אין צוללת ידועה, ערכו 1- |
| known\_direction | בית | זוכר את הכיוון שבו הצוללת ממשיכה, ערכו 1- אם הכיוון עדיין אינו ידוע, 0 אם הכיוון מעלה, 1 אם הכיוון מטה, 2 אם הכיוון שמאלה ו3 אם הכיוון ימינה |

## משתנים להדפסת קבצי BMP

|  |  |  |
| --- | --- | --- |
| שם משתנה | סוג / גודל | מטרה / שימוש |
| MenuPic | מחרוזת | שם הקובץ של תמונת מסך הפתיחה |
| PlayPic | מחרוזת | שם הקובץ של תמונת מסך המשחק |
| rules1 | מחרוזת | שם הקובץ של תמונת החוקים הראשונה |
| rules2 | מחרוזת | שם הקובץ של תמונת החוקים השנייה |
| rules3 | מחרוזת | שם הקובץ של תמונת החוקים השלישית |
| rules4 | מחרוזת | שם הקובץ של תמונת החוקים הרביעית |
| Player1Win | מחרוזת | שם הקובץ של תמונת הניצחון שחקן ראשון |
| Player2Win | מחרוזת | שם הקובץ של תמונת הניצחון שחקן שני |
| win | מחרוזת | שם הקובץ של תמונת הניצחון נגד מחשב |
| los | מחרוזת | שם הקובץ של תמונת ההפסד נגד מחשב |
| Header | מערך | מקום לקרוא לתוכו את header התמונה |
| Palette | מערך | מקום לקרוא לתוכו את palette התמונה |
| ScrLine | מערך | מקום לקרוא לתוכו שורות מהתמונה לפני שהיא מועברת למסך |
| ErrorMsg | מחרוזת | הודעת שגיאה בפתיחת הקובץ |

## משתני מחרוזת

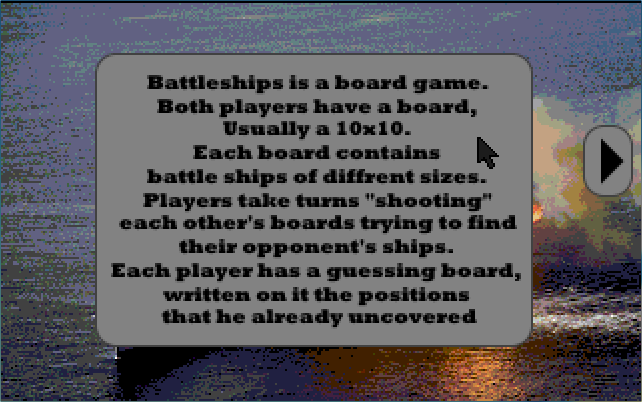
|  |  |  |
| --- | --- | --- |
| שם משתנה | סוג / גודל | מטרה / שימוש |
| blueExplain | מחרוזת | בהצגת המסך, להסביר לשחקן שמיקום כחול זה מיקום ריק |
| greenExplain | מחרוזת | בהצגת המסך, להסביר לשחקן שמיקום ירוק זה המיקום שהמצביע עליו, כלומר אם השחקן ילחץ enter המיקום הזה ייבחר |
| whiteExplain | מחרוזת | בהצגת המסך, להסביר לשחקן שמיקום לבן הוא חלק מצוללת |
| redExplain | מחרוזת | בהצגת המסך, להסביר לשחקן שמיקום אדום הוא מיקום בו הוא ירה ופגע בצוללת |
| yellowExplain | מחרוזת | בהצגת המסך, להסביר לשחקן שמיקום צהבו הוא מיקום שהתגלה כריק, כלומר אין בו צוללת בוודאות |
|  |  |  |
| active\_player\_string | מחרוזת | להדפיס את המילה player בראש המסך כשמציגים את המסך |
| active\_computer\_string | מחרוזת | להדפיס את המילה computer בראש המסך כשמציגים את המסך בתורו של המחשב |
| place\_ship1 | מחרוזת | החלק הראשון של המחרוזת שמודיעה לשחקן באיזה אורך הצוללת שהוא ממקם עכשיו |
| place\_ship2 | מחרוזת | החלק השני של אותה המחרוזת ^^^ |
| direction | מחרוזת | לשאול את השחקן לאיזה כיוון הוא רוצה שהצוללת שלו תמשיך (בשלב הצבת הצוללות כמובן) |
| direction2 | מחרוזת | להודיע לשחקן איך לאשר את כיוון הצוללת ואיך לבחור מיקום חדש |
| no\_more\_ship | מחרוזת | להודיע לשחקן שהוא סיים להציב את הצוללות שלו |
|  |  |  |
| where\_to\_shoot | מחרוזת | לשאול את השחקן באיזה מיקום על הלוח הוא רוצה לירות |
| shot\_already | מחרוזת | להודיע לשחקן שהמיקום שהוא בחר התגלה כבר ואין לו סיבה לירות שם |
| computer\_start\_turn | מחרוזת | להודיע לשחקן שמתחיל תור המחשב, ושהמחשב מוכן לירות |
| miss | מחרוזת | להודיע לשחקן שהוא פספס ביריה שלו |
| hit | מחרוזת | להודיע לשחקן שהוא פגע ביריה שלו |
| fallen\_ship | מחרוזת | להודיע לשחקן שהיריה שלו פגעה בחלק האחרון של הצוללת |
|  |  |  |
| any\_key\_to\_continue | מחרוזת | להודיע לשחקן שכדי להמשיך הוא צריך ללחוץ על מקש במקלדת |

# מסכי התכנית

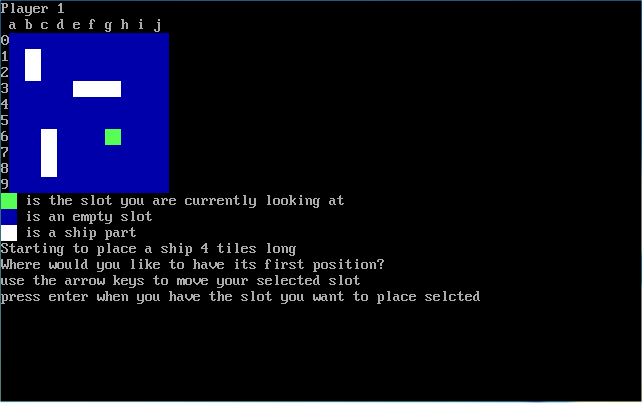
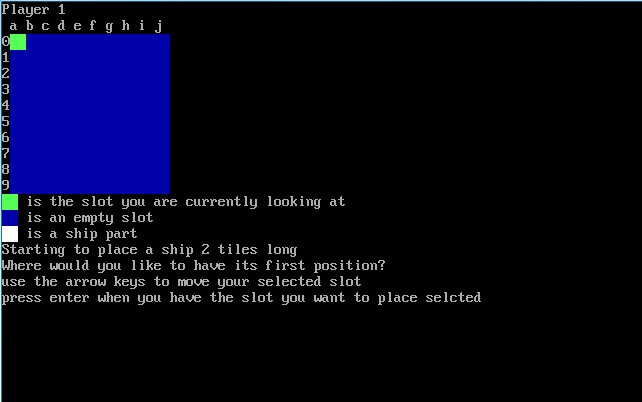
מסך הפתיחה

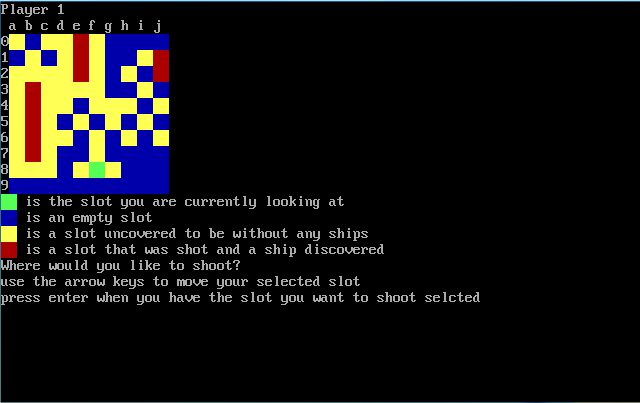
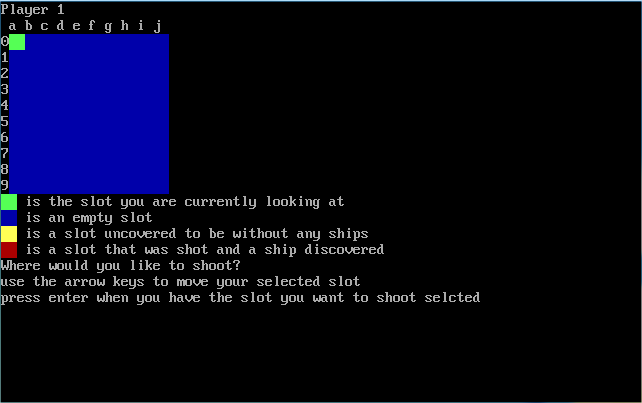


מסך המשחק, בחירת שחקן או שניים  

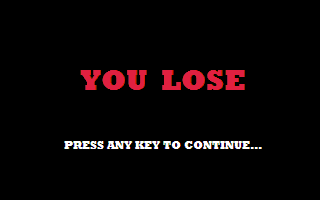
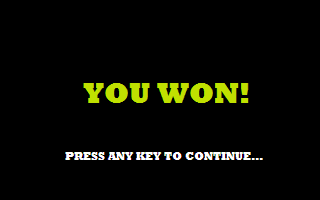
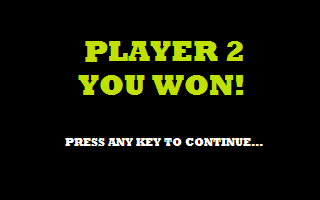

מסך החוקים, ישנם חוקים נוספים, הנמצאים במסכים דומים מאד לזה  


מהלך המשחק, בניית הלוחות



מהלך המשחק, ירייה זה בלוח של זה  


מסכי ניצחון/הפסד שונים המתאימים למצב סיום המשחק (שחקן ראשון, שחקן שני) ולסוג המשחק (שחקן יחיד, שני שחקנים)



# קוד התכנית

****

# סיכום ורפלקציה

במהלך כתיבת הפרויקט התנסיתי בסביבת עבודה שונה מאד ומסובכת יותר ממה שהכרתי לפני כן, אבל לעומת זאת, זכיתי גם לעשות דברים שאני לא יודע לעשות בשום שפה אחרת, בזכות הפשטות והבסיסיות של השפה, אסמבלי, זכיתי ללמוד הרבה מאד על אינטרפטים ואיזורי זכרון שונים ומגוונים (כמו לדוגמה (segment 0A000h.  
נהניתי להתנסות באפשרויות חדשות של פעולות, כמו להציג תמונה, טקסט, או צבעים, שהן אפשרויות שלא הכרתי בשפות אחרות לפני כן, כי למדתי אותן בצורה מאד בסיסית.

במהלך העבודה, החלטתי להוסיף אפשרות למשחק של שחקן נגד מחשב, כלומר ליצור למחשב אלגוריתם שמגריל לוח ויורה. לאחר מכן, החלטתי שמחשב שמגריל מיקום ליריה אינו מספק, זה אינו אלגוריתם חכם מספיק כדי לקרוא לזה פרויקט, אז החלטתי ליצור למחשב אלגוריתם שידע לירות בצורה טקטית, ולהפיל צוללות שלמות בצורה שיטתית, כמעט כמו ששחקן יכול.  
כתיבת אותה אינטליגנציה מלאכותית (נמוכה), דרשה ממני הרבה חשיבה ותכנון, והיה לי קשה מאד להגיע לתוצר הסופי, אבל בסופו של דבר הצלחתי.

# תודות

אני רוצה להודות לעידו גרזון וניר דפנאי, שהיו איתי בקשר לכל אורך כתיבת הפרויקט, ועזרו לי למצוא פתרונות לבעיות אלגוריתמיות, ולמדו איתי את האינטרפטים של אסמבלי, ובעזרתם הגעתי להבנה המתקדמת שלי באינטרפטים ובדרכי עבודה.  
אני רוצה להודות גם לגלית קלאוס, שפתחה את מגמת הסייבר בבית הספר שלנו, ואפשרה לי ולכל תלמידי המגמה להחשף לתוכנית "גבהים" של 8200, לחומר הלימוד ולשפה אסמבלי.