**שחמט**

שם התלמיד: יונתן צ'פאל

שם המנחה: ניר דויק

**מדריך למשתמש**

ברוכים הבאים למדריך למשתמש למשחק השחמט. מדריך זה ידריך אותך בתהליך ההפעלה ויסביר את המסכים והאפשרויות השונות הזמינות במשחק.

הפעלה:

כדי להפעיל את משחק השחמט יש לפתוח את Dosbox ולכתוב בתוך תיקיית המשחק: **Chess**

**המשחק משתמש בעכבר ובמקלדת על מנת לפעול**

מסך הפתיחה:

**מסך הפתיחה** הוא המסך הראשוני שמופיע עם הרצת המשחק.

כאן תמצאו את שם המשחק.

כדי להמשיך ל**מסך הראשי** יש ללחוץ על מקש הרווח.

כדי להמשיך ל**מסך הקרדיטים** יש ללחוץ על המקש 'C'.

כל מקש אחר ינגן צליל כישלון.

מסך הקרדיטים:

**מסך הקרדיטים** הוא מסך המראה את האנשים שעבדו ועזרו למשחק הזה (בעיקר בוסט לאגו).

כדי לחזור ל**מסך הפתיחה** יש ללחוץ על כל מקש במקלדת.

מסך ראשי:

ה**מסך הראשי** הוא המקום בו משחקים את המשחק עצמו.

במסך תראה לוח שחמט עם כל הכלים, המהלך האחרון שנעשה והיתרון המספרי של הלוח

האפשרויות הזמינות במסך הראשי:

בצע תנועה: כדי לשחק את המהלך שלך, לחץ או הקש על היצירה הרצויה ולאחר מכן לחץ או הקש על ריבוע היעד.

פרישה: לחץ או הקש על אפשרות זו כדי לצאת מהמשחק.

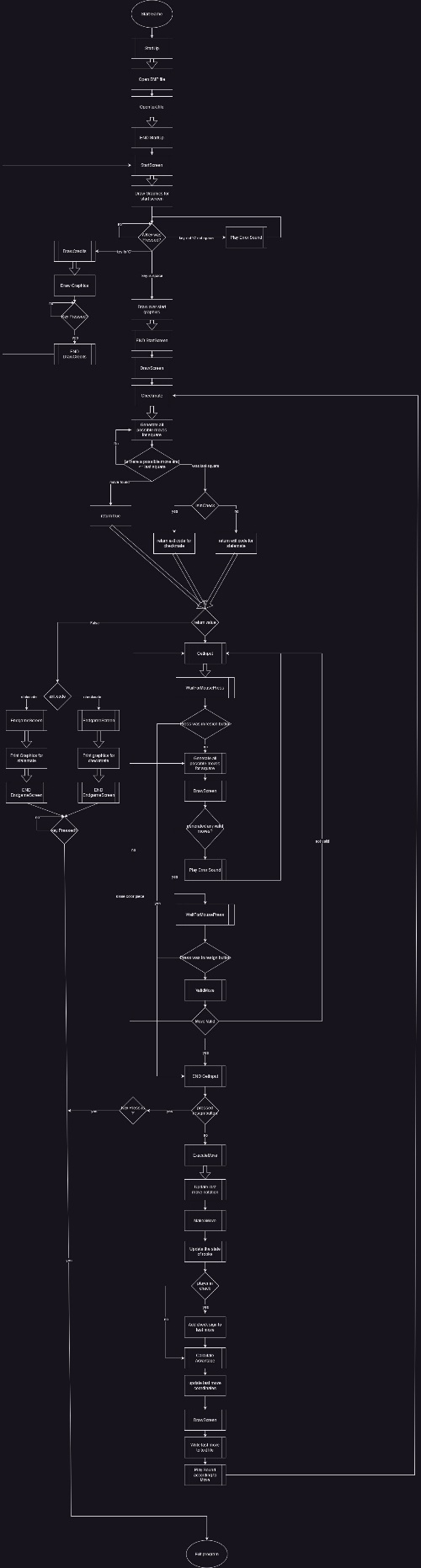
מסך סיום:

**מסך הסיום** מופיע כשהמשחק מסתיים, או על ידי שח-מט, או פרישה.

הוא מציג את תוצאת המשחק ("לבן מנצח", "פט", "שחור מנצח")

כדי לסגור את המשחק יש ללחוץ על כל כפתור במקלדת

תרשים זרימה כללי למשחק:



רשימת פרוצדורות:

--------------------------------------------------------------------------------------------------

Min

Args: two numbers n1, n2 (signed)

Action: checking the min number (n1 or n2)

Return: ax = the min number

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

IsInCheck

Args: None

Action: checks if the king is in check

Return: ax = is the king in check (-2 for king is check, exists for any other value)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

WaitForKey

Args: None

Action: clears the keyboard buffer and wait for key input

Return: al = ascii code for key

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

PlaySound

Args: offset of sound array

Action: plays the hrz values stored in the array, each note is 1 tick longer

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

OpenFile

Args: reference to filename on dx

Action: opens the bmp graphics file for read-only

Return: Returns file handle to ax or -2 on error

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

OpenTextFile

Args: reference to filename on dx

Action: opens the txt game record file for read-only and clears the previous game from file

Return: Returns file handle to [recordHandle] or -2 on error

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

ReadBMPHeader

Args: BX = file handle, DX = bmp header buffer

Action: Read BMP file header, 54 bytes, into [dx]

Return: the header stored on [tmpHeader]

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

ReadBMPPalette

Args: BX = file handle, DX = palette buffer

Action: Read BMP file color palette, 256 colors \* 4 bytes (400h)

Return: the palette stored on [Palette]

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

CopyBMPPalette

Args: si = palette buffer

Action: Copy the colors palette to the video memory registers

Return: the palette stored on [Palette]

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

CopyBMPToMemory

Args: None

Action: Copy bitmap to memory

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

CloseFile

Args: Bx = file handle

Action: Close the BMP file

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

CloseTextFile

Args: None

Action: closes the txt file

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

DrawFromMemoryAdvanced

Args: width, height, starting screen X, start screen Y, offset in BMP X, offset in BMP Y

Action: draw a segment of a bmp photo to any location on the screen

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

WriteToText

Args: None

Action: writes the last move into the txt file

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

startUp

Args: None

Action: enters graphics mode, open BMP file and txt record file

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

printBoard

Args: None

Action: prints the chess board to screen

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

PrintPieces

Args: None

Action: prints the chess pieces to the screen

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

HighlightLastMove

Args: the last move's starting and ending offsets (X, Y)

Action: prints the last move to the screen

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

DrawScreen

Args: None

Action: print the board, pieces, the last move, resign button, and game stats to the screen

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

EndgameScreen

Args: who won the game (-2 for a mate, -3 for a stalemate)

Action: print the win state the the screen, close the txt file

Return: None

--------------------------------------------------------------------------------------------------

;--------------------------------------------------------------------------------------------------;

;PrintCredits

;Args: None

;Action: prints the [credit] string array

;Return: None

;--------------------------------------------------------------------------------------------------;

;--------------------------------------------------------------------------------------------------;

;StartScreen

;Args: None

;Action: print the starting game graphics until key press, if enter draw over the graphics and return, if 'C' print credits and wait until key press to return to start screen

;Return: None

;--------------------------------------------------------------------------------------------------;

--------------------------------------------------------------------------------------------------

ReverseMove

Args: (X, Y) for start and location, piece that was in start and loc, offset of white and black king

Action: recives the coordinates for start and loc offsets of a move, and displays all possible moves of the selected piece

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

CheckMove

Args: starting offset (X, Y), location offset (X, Y)

Action: checks if move leaves the player in check and then restores the board state

Return: ax = does the move leave the player in check (-2 for leaves the player in check, doesn't leave the player in check for any other value)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

ValidMoveLoop

Args: cx = number of times to run, di starting location offset, si increment number

Action: add all the possible moves according to the args to [Highlight] array

Return: updated [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

RookValid2

Args: (X, Y) of start offset of the rook

Action: add all the possible moves of rook to the [Highlight] array

Return: updated [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

BishopValid2

Args: (X, Y) of start offset of the bishop

Action: add all the possible moves of bishop to the [Highlight] array

Return: updated [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

QueenValid2

Args: (X, Y) of start offset of the queen

Action: add all the possible moves of queen to the [Highlight] array

Return: updated [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

KnightValidChecker2

Args: (X, Y) diff from loc offset to start offset, ax = row (y), cx = col (x)

Action: add the rook move to the [Highlight] array if valid

Return: updated [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

KnightValid2

Args: (X, Y) of start offset of the knight

Action: add all the possible moves of knight to the [Highlight] array

Return: updated [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

CastlingThreatCheckKingValid2

Args: cx = number of itirations, di = the first block offset to check, si = diff between blocks

Action: check if there is a threat in the way of castling

Return: ax = can you castle (-2 you can't, any other value you can)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

KingValid2

Args: (X, Y) of start offset of the king

Action: add all the possible moves of king to the [Highlight] array

Return: updated [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

SimulateEnPassant2

Args: di = source offset, si = loc offset, ax = is en passant valid

Action: a unique version of checkMove for simulating EnPassant move

Return: ax = is en pasaant possible (-2 not possible, any other values possible)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

PawnValid2

Args: (X, Y) of start offset of the pawn

Action: add all the possible moves of pawn to the [Highlight] array

Return: updated [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

ResetHighlight

Args: None

Action: Resets the [Highlight] array to 0

Return: updated empty [Highlight] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

PrintHighlight

Args: None

Action: print the [Highlight] array to the screen

Return: None

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

ShowHighlight

Args: starting location in [startX] and [startY]

Action: generates all possible moves for the piece in the start offset

Return: ax = if piece was valid (-2 for not valid, any other value for valid), updated [Highlight] array (1 for possible move, 0 for not possible move)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

AllThreatsLoop

Args: cx = number of times to run, di = starting location offset, si = increment number

Action: updates the [ThreatMap] cells to active according to the args

Return: updated [ThreaMap] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

RookAllThreats

Args: the starting location of the rook (X, Y)

Action: add all possible attacking moves of the rook to the [ThreatMap] array

Return: updated [ThreaMap] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

BishopAllThreats

Args: the starting location of the bishop (X, Y)

Action: add all possible attacking moves of the bishop to the [ThreatMap] array

Return: updated [ThreaMap] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

QueenAllThreats

Args: the starting location of the queen (X, Y)

Action: add all possible attacking moves of the queen to the [ThreatMap] array

Return: updated [ThreaMap] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

KnightAllThreatsChecker

Args: ax = Y origin of rook, cx = X origin of rook, bx = [board] offset, diff between start to target cell (X, Y)

Action: check if the atack move is possible, if so add it to the [ThreatMap] array

Return: updated [ThreaMap] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

KnightAllThreats

Args: the starting location of the knight (X, Y)

Action: add all possible attacking moves of the knight to the [ThreatMap] array

Return: updated [ThreaMap] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

KingAllThreats

Args: the starting location of the king (X, Y)

Action: add all possible attacking moves of the king to the [ThreatMap] array

Return: updated [ThreaMap] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

PawnAllThreats

Args: the starting location of the pawn (X, Y)

Action: add all possible attacking moves of the pawn to the [ThreatMap] array

Return: updated [ThreaMap] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

CreateThreatMap

Args: None

Action: add all the threats to the current player

Return: updated [ThreaMap] array (1 for active square, 0 for inactive square)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

Checkmate

Args: None

Action: check all the possible moves for a player for one that doesn't leave the king in check

Return: ax = is there a move that the player is not in check (-2 for check mate, -3 for stalemate, valid for any other value)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

PossibleMove

Args: None

Action: check for a valid move in the [Highlight array]

Return: ax = was such move found (-2 = not such move found, a move was found)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

IsExit

Args: cx = X value, dx = Y value

Action: check if the offset of (X, Y) on screen is in the range of the resign button

Return: ax = if (X, Y) value in range of resign button (-2 for yes, any other value for no)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

ValidMove3

Args: start and loc offset (X, Y) in vars

Action: checks if the action selected was a valid action and returns accordinly (updates the castle array if necessary

Return: ax = move status (-2 = not valid, -3 = piece of the same color selected, any other value = valid)

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

ProccesInput

Args: offset of (X, Y) of a press

Action: waits for a valid key press in range

Return: ax = wants to exit (-2 = wants to exit, any other values = valid), press (X, Y) in [offset\_x\_ProccesInput]-[offset\_y\_ProccesInput]

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

GetInput2

Args: None

Action: recives the coordinates for start move, displays all possible moves for the piece, recives loc move and checks if the move is valid

Return: ax = player wants to exit (-2 for yes, any other values for no), (X, Y) values for start and loc of a move

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

UpdateRook

Args: None

Action: checks if a rook that hasn't moved had moved

Return: the updated [rooks] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

MakeAMove

Args: starting offset (X, Y), Location offset (X, Y)

Action: executes the move needed (including castling, promotion, en passant)

Return: updated [enPassan] and [kings] array

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

UpdateLastMove

Args: (X, Y) for start and location

Action: changes the [lastMove] var to the newest move executed according to chess notations

Return: updated [lastMove] string

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

CalcAdv

Args: None

Action: sums the material of each player and stores the result

Return: the sum of material as a string in [advantage]

--------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------

ExecuteMove

Args: (X, Y) for start and location

Action: update the [lastMove] var, the [rooks] array, and [adv], make the move, play the appropriate sound, draw the screen after the move, and record the move to the txt file

Return: None

--------------------------------------------------------------------------------------------------

;--------------------------------------------------------------------------------------------------;

;DrawBlackScreen

;Args: None

;Action: changes all pixels to be black

;Return: None

;--------------------------------------------------------------------------------------------------;

רפלקציה

נתקלתי בהרבה קשיים במהלך הפרויקט. פחות או יותר כל פעם שניסיתי להוסיף משהו, זה לא עבד על הפעם הראשונה. החלק הראשון שניסיתי להוסיף אלה היו הגרפיקות. בהתחלה ניסיתי להדפיס פיקסל-פיקסל אך מהר מאוד הבנתי שזה לא יעבוד ואני צריך דרך אחרת. בספר השתמשו בקובץ bmp אך לא הבנתי איך בדיוק ליישם את זה בפרויקט שלי והתקשיתי עם זה. יועד שלח לי קישור שהיה בספר (אך משום מה לי הוא לא עבד) על דף פורום שמסביר איך לעבוד יותר טוב עם קבצי bmp אחרי זה כל מה שנשאר לי זה היה להבין איפה ליצור את זה. שיצרתי את זה בפיינט או פוטושופ, הצבעים לא יצאו בדיוק כמו רציתי, ואז יועד **שוב** גילה לי את התוכנה asperite שאיתה יכולתי ליצור את הגרפיקות בקלות יחסית ולהוסיף את כל הגרפיקות לפרויקט. לאחר מכן ניסיתי ליצור פונקציה המקבלת מהלך ובודקת אם הוא תקין. זה היה הפרוצדורות הראשונות שיצרתי לכן היו לי בעיות עם פושים, פופים. ניהול זיכרון בכללי ויעילות, אז לקח לי כמה ימים עד שהצלחתי לגרום לכל הפונקציות לעבוד. בהמשך ניסיתי להוסיף את הפונקציות שמייצרות את כל המהלכים ה"מאיימים" של החיילים (כלומר כל האכילות בשביל לבדוק אם המלך בשח או האם המהלך שהוא יבצע יכניס אותו לשח), אז בינתיים לא דאגתי לדברים כמו הצרחה, ואן פסה לא עשיתי עדיין. זה היה קשה במיוחד להוסיף. אין שום אינדיקציה ויזואלית לגבי מה המחשב מבצע ולא יכולתי לדעת אם הוא חישב נכון את כל האיומים. ישבתי שבוע, אפילו יותר רק יושב מול הדיבאגר ובודק מקרה קצה אחרי מקרה קצה לבדוק שהוא חישב את האיומים כמו שצריך. לאחר מכן יצרתי את השחמט. השחמט בבסיס דומה מאוד לפונקציות האיומים אך ישנם מהלכים נוספים שהיו צריכים לבדוק (במיוחד אלו של חייל), והיו מקרים מסוימים שמה שהתאים לכל האיומים לא התאים בתור מהלך תקין (למשל אכילה של חייל משלך). היה לי מאוד קשה להוסיף את החייל כי האכילות שונות לפי צבע החייל (החייל היחיד עם החוק הזה) וגם איך שהוא זז קדימה משתנה על פי צבע החייל ומיקומו (תזוזה ראשונה אפשר לזוז פעם אחת או פעמיים). שוב הדרך לפתור היה לשבת ולעבור מקרה מקרה, משחק משחק עד שחשבתי שמצאתי את על מקרי הקצה. בכללי, המכשול העיקרי בפרויקט שלי הוא הכמות העצומה של דברים לבדוק, מקרי הקצה הרבים, ולבדוק שאכן הכל עובד. לא פעם אחת ולא פעמיים חשבתי שהכל עובד ופתאום רצף מהלכים מסוים, או רצף לחיצות מסוימים הקריסו לי את כל המשחק בלי שום סיבה נראית לעין, או שאפילו במקרים מעצבנים במיוחד. המשחק עשה את מה שהוא היה אמור לעשות אם היית מגיע לפוזיציה בדרך אחת. אך אם היית מגיע אליה בדרך אחרת זה לא היה עובד. דברים כאלו מאוד הקשו עלי לסיים את המשחק ולקחו לי הרבה זמן לפתור. אחרי זה, פתרתי באגים קטנים (המלך יכול לאכול את עצמו, הפונקציה שלי לקבלת האינפוט מהשחקן הייתה משנה לי את הזיכרון לפעמים ומקריסה לי את התוכנית, מקרי קיצון מסוימים לא עובדים). לאחר מכן עבדתי על להוסיף את המהלכים המיוחדים כמו קידום, הצרחה ואן פסה. ההצרחה במיוחד גרמה לי לבעיות, לבדוק אם המלך זז, האם הצריחים זזו, האם יש איום בהצרחה. בגלל הצורה שהפונקציות שלי עבדו זה היה שכתוב יחסית גדול כדי להוסיף את ההצרחה והאן פסה. במיוחד הבדיקה של תזוזת הצריחים (איפה לקרוא לה בקוד) השתנה לפי מקרי קצה שונים שראיתי ובדקתי. לבסוף הוספתי פיצ'רים של אורחות חיים, מסך פתיחה וסיום, המהלך האחרון והיתרון של הלוח. בסוף אמרתי שאני אוסיף משהו שיראה לך את כל המהלכים האפשריים של החייל שלחצת עליו. חשבתי שזה שעה-שעתיים... כמה טעיתי! זה לקח שכתוב של עשרות פונקציות, הוספה של הרבה פונקציות גם כן, והמון שעות של ניסוי וטעיה. אך לבסוף זה עבד. אחרי שיצרתי אותם הבנתי שניתן לגמרי למחוק את כל פונקציית השחמט שלי, ופונקציית הבדיקה של האם מהלך הוא מהלך תקין, והחליפן בפונקציות הרבה יותר קצרות ויעילות ובכך לחסוך ~650 שורות קוד. גם המיקום קריאה של הפונקציה שבודקת את התקינות של מהלך נכנה לתוך פונקציית קבלת הקלט והוספתי פיטצ'רים נוספים של אורחות חיים כגון קול (גם זה יצר בעיות בהתחלה מכיוון שהלא עבדתי עם השעון לפני כן והייתי צריך לייעל את הקוד עוד יותר), מסך קרדיטים, ושינוי הדרך שאני מקבל לחיצות מעכבר. לבסוף גם הוספתי תיעוד של המשחק האחרון לקובץ טקסט.

בחיים שלי לא עבדתי על פרויקט כזה גדול בשפת תכנות נמוכה כזו, הפרויקט הזה באמת גרם לי לחשוב על תכנון הקוד שלי, על ההבנה שלי איך המחשב עובד וככה גם איך לייעל את הקוד שלי. הוא עזר ליכולות התכנון והתיעוד שלי (שצריכים עזרה), ועזר לי ללמוד איך לדאבג יותר טוב (כי עם כל השעות שהשקעתי חסר לי אם לא למדתי כלום). אני מאוד נהנתי מהפרויקט, יותר ממה שציפיתי ואני באמת ממש מאוד גאה בו, ואני מקווה שגם מי שישחק בו ייהנה לשחק בו כמו שאני נהנתי ליצור אותו.

מקורות חיצוניים

<https://www.daniweb.com/programming/software-development/threads/77762/how-to-clear-keyboard-buffer>

<https://stackoverflow.com/questions/68045082/how-to-print-a-string-centered-vertical-and-horizontal-in-video-mode>

<https://www.ic.unicamp.br/~pannain/mc404/aulas/pdfs/Art%20Of%20Intel%20x86%20Assembly.pdf>

<https://stanislavs.org/helppc/idx_interrupt.html>

<https://www.rapidtables.com/code/text/ascii-table.html>

<https://pdos.csail.mit.edu/6.828/2018/readings/hardware/SoundBlaster.pdf>

ספר אסמבלי

מצגות בקלאסרום אסמבלי