**שמות המגישים:**

עידו אולמר 203428453

איציק ישר 302732029 [itzikyashar@gmail.com](mailto:itzikyashar@gmail.com)

**החלטות שונות:**

* עבור הסטטיסטיקה של מספר הפעמים ששחקן הגיע למספר דסקיות של שתיים – החישוב מתבצע עבור מספר התורות (תור כפי שהוגדר בתרגיל) שבו השחקן התחיל ממצב של שתי דסקיות.
* עבור הסטטיסטיקה של הזמן הממוצע הנדרש לשחקן לבצע מהלך – בכל תור הזמן המחושב הוא ממוצע הזמנים של השחקן עבור אותו משחק, כולל זמן הביצוע בתור הנוכחי.
* עבור הדרישה של רספונסיביות – כפי שנאמר בכיתה, דאגנו לרספונסיביות עד גודל לוח של 10X10.
* בחרנו להציג גרפית את האפשרות/איסור על מהלך כלשהו על ידי הצגת דסקית מעל משבצת "חוקית" ואי הצגה של דסקית במשבצת לא חוקית. מהלכים אלו קוראים בכניסה ויציאה של סמן העכבר אל המשבצת הרצויה.
* את הסטטיסטיקות של המשחק ניתן לראות בכל עת וגם לאחר ניצחון של שחקן כלשהו, הסטטיסטיקות נשארות עד אשר יבחר משחק חדש.

**הסבר על פונקציות מרכזיות:**

המשחק מורכב ממספר מחלקות לוגיות:

class Player

class Stats

class GameMaster

class AI

ושאר הפונקציות נמצאות בסקופ הגלובלי אחראיות על הUI

מחלקת הPLAYER ומחלקת הSTATS אחראיות על שמירת מידע רלוונטי על השחקן ועל סטטיסטיקות בהתאמה.

מחלקת הGameMaster אחראית על הניהול הלוגי של המשחק, המחלקה מחזיקה שתי שחקנים, סטטיסטיקות, לוח ואת גודלו, לוח שמייצג אפשרויות למהלך כלשהו, ומשתנה שמייצג מי אמור לשחק.  
פונקציות מרכזיות:

isValidMove(i, j)

מקבלת קורדינאטה בלוח ובודקת האם השחקן שעכשיו תורו רשאי לבצע מהלך במשבצת זו.

updateMatrixOfPossibilities(i, j)

מקבלת קורדינאטה למשבצת, ומכינה לוח חדש שמייצג מה היה קורה אילו השחקן הניח דסקית על הוקורדינאטה שהתקבלה. הרעיון מאחורי זה הוא השימוש בTrainer

makeAMove()

הפונקציה שמקבעת את המהלך האפשרי שנשלח לפונקציה הקודמת (updateMatrixOf…), למעשה הסדר שבא לפני ביצוע מהלך הוא סדר הצגת הפונקציות; בדוק אם המהלך חוקי, עדכן את לוקח האפשרויות, בצע את המהלך.

class AI

constructor(i\_Game) // GameMaster

מחלקה שנועדה לנהל את מהלכי הAI, פונקציה מרכזית:

makeAIMove()

הפונקציה מבצעת מהלך של הAI במשחק שאליו הוא שייך (מה שקיבל בCOTR )הפונקציה עוברת מחליטה מה לעשות על ידי קריאה לפונקציה

miniMax(i, j, depth, wantMax, alpha, beta)

הפונקציה מממשת עץ החלטות מותאם למשחק.

בUI:

function onMouseClick\_Cell(i, j)

משוייך לכל לחיצה בעבר על משבצת בלוח – הפונקציה בודקת האם מותר לשחקן לבצע את המלך, ואם מותר אז היא מבצעת אותו.

function \_nextTurn()

אחרי שמתבצע מהלך, הפונקציה הנ"ל אחראית לעדכון הלוח, סטטיסטיקות והתור.

function onMouseEnter\_Cell(i, j)

כאשר שחקן נכנס עם העבר למשבצת כלשהי, הפונקציה מציגה לו האם מותר לו להניח דסקית במיקום הנ"ל, ואם מצב התלמדות עובד, הפונקציה גם תדאג להראות מה תהיה התוצאה של מהלך כזה.

function Run()

עושה התאמות ראשונות כמו הוספת ליסנרים לכפתורים,

function \_createGameTable()

יוצר את לוח המשחק הפיזי בצורה דינאמית

בנוסף יש המון מחלקות בUI שאחריות בעיקר על עדכון נתונים ופעולות בסיסיות כל כפתורים וכו'..

**מימוש בונוסים:**

* בונוס 1 – מימוש התחלת משחק חדש + סטטיסטיקה נוספת לפי הדרישה. התחלת משחק חדש מתבצעת על ידי כפתור התחלת משחק (הרגיל). חישוב הסטטיסטיקה הנ"ל מחושבת בתור הממוצע של הזמן לביצוע המהלך בתור הנוכחי + ממוצעי הזמן לביצוע מהלך של כל המשחקים הקודמים.
* בונוס 2 – מצב מתלמד. את פוטנציאל הרווח הצגנו על הלוח עצמו בדמות שינוי צבעם של הדסקיות הפוטנציאליות לרווח לפי צבע השחקן המרוויח.
* בונוס 3 – בחירת גודל לוח. מומש על ידי כפתור של רשימה נגללת.
* בונוס 4 – אלגוריתם AI: ממוש של עץ החלטות minimax עם שימוש α,β לצורך יעילות  
  שיטת הדירוג של ההחלטות מתבסס על ניקוד למשבצות קריטיות בלוח עם התבוננות על ניקוד כללי ומצב סופי של המשחק, שיטת הדירוג די בסיסית ובשביל להעלות את רמת ההצלחה של הAI אפשר להוסיף ניקוד למשבצות שלא ניתן להפוך אותם ולהוריד למשבצות שניתן להפוך. (את שיטת הדירוג המורכבת יותר מימשנו ללוח 10 ולוח 8, לאחרים הדירוג די נאיבי)