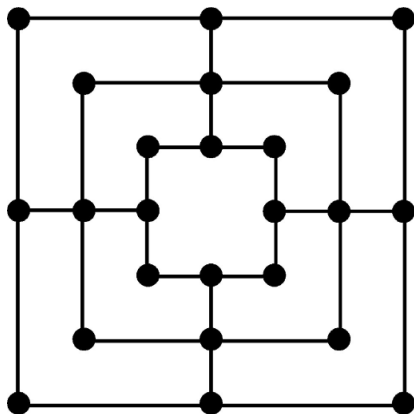




Nine Men Morris By Tom Kark

עבודת גמר בתכנון ותכנות מערכות בחלופת מערכות מומחה
שנת הלימודים תשפ"א

שם בית הספר: כיח אליאנס - חיפה
סמל בית הספר: 340190
שם העבודה: Nine Men Morris
שם התלמיד: טום קרק
שם המנחה: אריאל בר-יצחק





Nine Men Morris By Tom Kark

תוכן עניינים

3	מבוא
4	אפיון המערכת
5	• חוקי המשחק
8	ארכיטקטורה של הפרויקט
9	• Board – המחלקה
12	• BoardAB – המחלקה
14	• Const – המחלקה
15	• GameVisuals – המחלקה
17	• IndexHooks – המחלקה
18	• MorrisButton – המחלקה
19	• MorrisNode – המחלקה
21	• Move – המחלקה
22	• MoveScore – המחלקה
23	ייצוג בסיס הידע
25	תיאור אלגוריתם המחשב
29	• היוריסטיקות וחישוב הנקודות למהלך
31	מדריך למשתמש
33	רפלקציה
34	תודות
35	ביבליוגרפיה



Nine Men Morris By Tom Kark

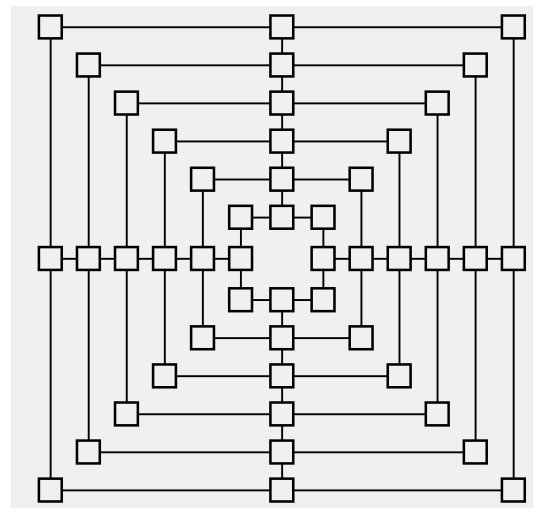
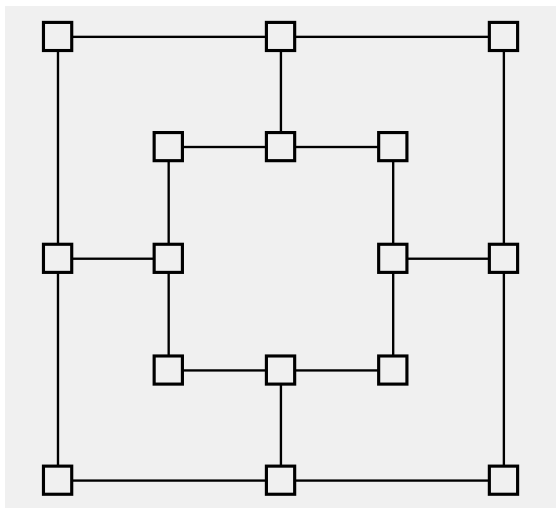
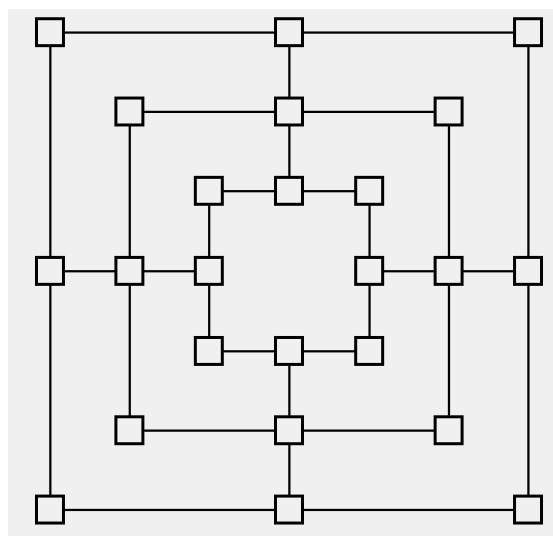
מבוא

הקדמה, רקע על המשחק / פאזל

הקדמה ורקע - הפרויקט עוסק במשחק הלוח Nine Men Morris, משחק אשר כבר קיים בעולמנו אלפי שנים, המשחק בא בכמה גרסאות כגון: Six Men Morris, Twelve Men Morris וכו', הפרויקט עצמו עוסק בNine Men Morris בלבד. בפרויקט המוצג הלוח הינו מודולארי ולכן יש מניע טבעי לחשוב שהפרויקט מייצג בעצם את שאר הגרסאות של המשחק, אבל זה לא נכון כלל וכלל, לדוגמא בTwelve Men Morris יש גם מהלכים שלא קיימים בNine Men Morris ולכן הפרויקט מייצג את Nine Men Morris בסקאלות שונות. בפרויקט ישנו גיוון רב של אפשרויות ותכונות, בין אם זה משחק של 2 אנשים אחד מול השני, או מחשב מול השחקן ואפילו מחשב מול מחשב עם עד 2 היוריסטיקות שונות. הפרויקט מעוצב מהבחינה הגרפית בצורה מושקעת ויעילה, תוך יישום של החוקים, התכונות והליך המשחק.

אפיון המערכת

לוח המשחק: לוח המשחק המקורי בנוי מ3 שכבות של "מסגרות", כל שכבה מכילה 8 תאים ולכן הלוח מסתכם ב24 תאים בסה"כ. בפרויקט זה ניתן להעלות את כמות השכבות במשחק, כלומר ניתן לשחק במעט שכבות עד כדי 2 שכבות ואפשר גם לשחק במימדים עצומים, עד 9 שכבות.





Nine Men Morris By Tom Kark

חוקי המשחק

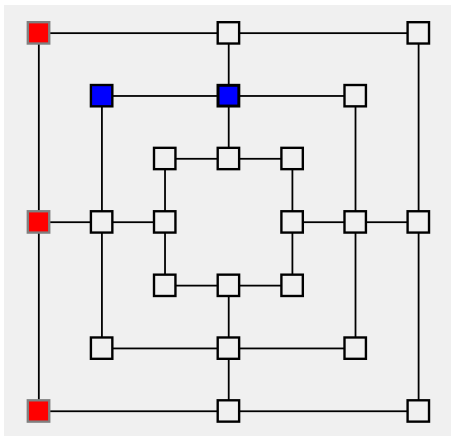
המשחק בנוי מכמה שלבים, תחילה יש לדעת כי לכל שחקן ישנן 9 חתיכות משלו (במידה ומשנים את סקלת הלוח אז מספר החתיכות משתנה בהתאם, אבל נדון במקרה ברירת המחדל במהלך אפיון חוקי המשחק), עתה נסתכל על כל שלב בנפרד על מנת להבין את החוקים ומהלך המשחק:

- שלב 0: בשלב זה כל שחקן מניח בתורו חתיכה בתא ריק במידה ועדיין יש לו חתיכות, השלב נגמר כאשר כל שחקן הניח את כל החתיכות שלו על הלוח.
 - שלב 1: שלב זה ישר מתחיל לאחר סיום שלב 0, בשלב זה כל שחקן רשאי להזיז את אחת החתיכות שלו לכל השכנים הריקים של החתיכה (תא ריק),
 - שלב 2: שלב זה אינו משותף לשני השחקנים בניגוד לשלבים הקודמים, כל שחקן מגיע לשלב זה בנפרד מהאויב (שחקן 1 יכול להיות בשלב 2 אבל שחקן 2 לא בהכרח בשלב 2), לשלב זה מגיע שחקן כאשר מספר החתיכות שלו אשר נמצאות על הלוח הוא 3, שחקן אשר מגיע לשלב זה יכול "לעוף", כלומר במקום להזיז חתיכות לשכנים הריקים שלהן, הוא יכול להזיז כל חתיכה לכל תא ריק על הלוח.
- עתה נשאלת השאלה,

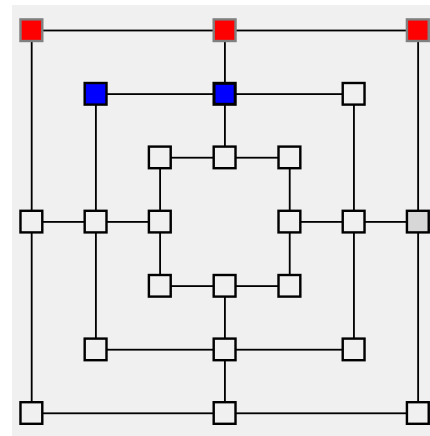
איך מצמצמים לאויב את מספר החתיכות ומתי שחקן מסוים מנצח?

מצמצמים לאויב את מספר החתיכות כאשר השחקן הנוכחי יוצר שלשה (Morris), שלשה נוצרת כאשר השחקן הנוכחי מניח חתיכה ונוצר מצב שקיימות 3 חתיכות רצופות בשורה או 3 חתיכות רצופות בטור, מצ"ב דוגמא לכל מצב:

טור

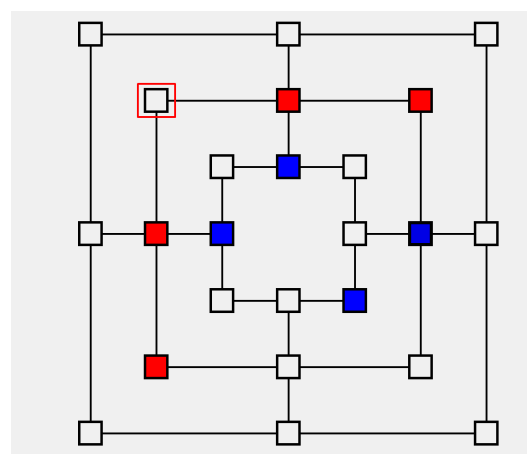


שורה



במצב זה, כאשר השחקן האדום יצר שלשה, הוא רשאי למחוק לכחול חתיכה אשר לא נמצאת בתוך שלשה כחולה, במידה וכל החתיכות של הכחול נמצאות בתוך שלשות כחולות, אז השחקן האדום רשאי למחוק חתיכה אחת מכל החתיכות.

הערה חשובה: כאשר נוצר ששחקן כלשהו, לדוגמא, האדום, מניח חתיכה ויוצר Morris 2 עדיין הוא רשאי לאכול לאויב רק חתיכה אחת ולא שתיים! כפי שניתן לראות בתמונה המצורפת מתחת, אם נשים חתיכה אדומה בתא המסומן, ניצור בעצם 2 שלשות אבל נהיה רשאים למחוק רק כחול יחיד!

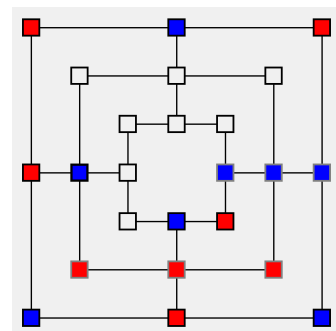




Nine Men Morris By Tom Kark
איך מגיעים לנצחון?

צריך לצמצם לאויב את מספר החתיכות ל-2, מכיוון שכאשר שחקן מגיע ל-2 חתיכות הוא לא יכול ליצור שלשות יותר ולכן הוא מפסיד.

ישנו מקרה נוסף שהוא נדיר במיוחד שגם הוא מגיע לנצחון, כאשר שחקן מצליח ללכוד את כל החתיכות של האויב, כלומר לא משנה איזו חתיכה האויב ייבחר וינסה להזיז, כל השכנים שלה לא יהיו ריקים משמע אין לה לאן לזוז, ולכן הוא מפסיד אוטומטית, דוגמא של המצב הנדיר:



כפי שניתן לראות, הכחול ניצח מכיוון שכל החתיכות האדומות חסומות, לא משנה איזה חתיכה אדומה תיבחר, היא לא תוכל לזוז ולכן הכחול ניצח.

מטרת המשחק - לנצח את האויב בכל דרך ניצחון אפשרית, בין אם זה לצמצם לאויב את כמות החתיכות ל-2 או לחסום אותו לחלוטין.

מומחיות המערכת - יכולת לשחק את המשחק כרגיל (שחקן אנושי נגד שחקן אנושי), או לשחק נגד המחשב אשר יכול לשחק ב-2 דרכים שונות בשימוש היוריסטיקות שונות ולבסוף ניתן גם למחשב לשחק מול המחשב. המחשב משחק בעזרת אלגוריתם Alpha-Beta אשר מסתכל קדימה במהלכי המשחק וקובע לעצמו מה המהלך היעיל והחכם ביותר בדרך לנצחון.

קהל היעד - כל איש אשר מעוניין לפתח את חוט המחשבה ולאתגר את עצמו במשחק מול המחשב, או כל איש אשר מחבב את המשחק.

מטרות המערכת - הפרויקט מנסה להמחיש את חווית המשחק דרך המחשב בצורה הריאליסטית ביותר, המערכת יכולה להאיר את עיני השחקן בדרכי המשחק שלה ואף לחדש לשחקן רעיונות במשחק.



Nine Men Morris By Tom Kark ארכיטקטורה של הפרויקט

(על מנת לעבור לפירוט המחלקה יש ללחוץ על השם שלה המסומן בכחול ברשימה)

ישנן 9 מחלקות שונות בפרויקט:

- [Board](#) - מחלקה שמנהלת את המשחק, בודקת ומחשבת לבד את חוקיות המהלכים וכו'. המחלקה מנהלת את הרקע של מה שרואים בגרפיקה, היא מנהלת את ההזזות, אכילות וכל הדברים העיקריים והבסיסיים של המשחק.
- [BoardAB](#) - מחלקה אשר יורשת מBoardn ובנוסף לכך מנהלת את האלפא בטא והיורסטיקות במשחק כאשר המחשב משחק.
- [Const](#) - מחלקה אשר שומרת המון תכונות של ערכים קבועים בשביל המשחק, המטרה העיקרית היא שיהיו ערכים קבועים מראש ולא יהיה שימוש רב של Magic Numbers בקוד.
- [GameVisuals](#) - המחלקה יורשת מForm מברירת מחדל, המחלקה דואגת לגרפיקה של הפרויקט ולכל הניהול של הפנים של הפרויקט, כל הלחיצות, הכפתורים והלוח שהמשתמש רואה.
- [IndexHooks](#) - המחלקה שבונה את הלוח (מבני הנתונים), יוצרת MorrisNodes ומקשרת בין כולם על מנת ליצור את הלוח, מבני הנתונים שלנו הוא בעצם גרף 4 כיווני, כלומר כל Node בעל דרגת יציאה שלא עולה על 4.
- [MorrisButton](#) - מחלקה אשר יורשת מButton ובעצם השוני העיקרי שלה היא שהיא מצביעה על חתיכה, כל התאים על הלוח הם בעצם כפתורים, אז יש צורך שכל כפתור יוכל להפנות אל MorrisNode המתאים לו כדי להקל על חלקים מסוימים באלגוריתמים.
- [MorrisNode](#) - מייצג כל תא בלוח, המחלקה בעלת תכונות רבות אשר נותנות המון מידע על כל חתיכה כמו המיקום שלה וכו'.
- [Move](#) - מחלקה אשר מייצגת מהלכים במשחק, ושומרת נתונים כגון: מי זז? לאן זז? מאיפה זז? האם אכל? מה אכל? וכו'.
- [MoveScore](#) - מחלקה שנועדה לסייע לBoardAB, המחלקה שומרת מהלך (Move) ושומרת את הציון שהיורסטיקה נתנה לו.



Nine Men Morris By Tom Kark

המחלקה – Board

משמשת כמחלקה שמייצגת את לוח המשחק והפעולות שניתן לעשות בו ובכך בעצם מנהלת את הלוח והמשחק שמתרחש.

המחלקה BoardAb יורשת ממנה.

תכונות:

- **winner** - משתנה מטיפוס שלם אשר מייצג את השחקן המנצח, במהלך המשחק ערך המשתנה 0, ו1 אם השחקן הראשון (אדום) ניצח ו2 אם השחקן השני (כחול) ניצח
- **phase** - משתנה מטיפוס שלם אשר מייצג את השלב הנוכחי של המשחק, יש שלושה שלבים, שלב 0, 1 ו2.
- **player** - משתנה מטיפוס שלם שמייצג את השחקן הנוכחי שתורו לבצע מהלך, 1 לשחקן 1 ו2 לשחקן 2
- **lastRandomized** - משתנה מטיפוס שלם אשר שומר את מספר המהלכים שעברו מאז פעם אחרונה שנאכל חייל
- **moves** - רשימה של החתיכות האפשריות שניתן להזיז אליהן חתיכות מטיפוס MorrisNode, יש בה איברים רק כאשר שחקן רשאי למחוק לאויב חתיכות, אחרת היא ריקה
- **removeMoves** - רשימה של החתיכות האפשריות שניתן למחוק לאויב מטיפוס MorrisNode, יש בה איברים רק כאשר שחקן רשאי למחוק לאויב חתיכות, אחרת היא ריקה
- **removeOn** - משתנה בוליאני שערכו הוא אמת כאשר אחד השחקנים צריך למחוק לאויב חתיכה
- **onGoing** - משתנה בוליאני שערכו הוא אמת כאשר שחקן בחר חתיכה והוא צריך לבחור לאן להזיז אותה, כלומר יש תור מתמשך כרגע, אחרת שקר.
- **gameOver** - ערך בוליאני שערכו הוא אמת כאשר המשחק נגמר ואחרת שקר
- **hooks** - משתנה מטיפוס IndexHooks ששומר את מצב הלוח
- **playerPieces** - מערך מטיפוס שלם אשר שומר את מספר החתיכות על הלוח לכל שחקן אחרי שנגמר שלב 0



Nine Men Morris By Tom Kark

- **leftPieces** - מערך מטיפוס שלם ששומר בשלב 0 את כמות החתיכות שנשארו לכל שחקן להניח על הלוח
- **morrises** - מערך מטיפוס שלם ששומר כמה שלשות יש לכל שחקן
- **moveHistory** - מחסנית מטיפוס Move אשר שומרת את היסטורית המהלכים מתחילת המשחק

פעולות:

- **Board(int layer = layers)** - פעולה בונה של המחלקה אשר מאתחלת את כל המשתנים הרלוונטים לאתחול ומגדירה לוח חדש בגודל layer, הפעולה לא מחזירה כלום
- **PlacePiece(MorrisNode src, MorrisNode dest)** - פעולה המקבלת שני משתני MorrisNode ושמה חתיכה בלוח בתא DEST, אם הsrc קיים אז הפעולה גם מוחקת את החתיכה מsrc, הפעולה מחזירה ערך מטיפוס שלם, 0 במידה שהחתיכה הונחה בהצלחה אבל לא נוצרה שלשה, 1 במידה והונחה בהצלחה וכן נוצרה שלשה, 2 חתיכה כבר קיימת שם, 3 אי אפשר לשים שם חתיכה, 4 שני הפרמטרים שגויים.
- **CountPieces(int p)** - פעולה המקבלת ערך מטיפוס שלם p וסופרת כמה חתיכות על הלוח יש לשחקן הק ומחזירה את המספר הזה, גם הוא טיפוס שלם.
- **ChoosePiece(MorrisNode node)** - פעולה המקבלת משתנה מטיפוס MorrisNode ובוחרת אותו כדי להתחיל את התור, אם אין בעיה בבחירה של החתיכה הזאת אז הפעולה מחזירה אמת אחרת היא מחזירה שקר.
- **RemovePiece(MorrisNode src)** - הפעולה מקבלת משתנה מטיפוס MorrisNode ומוחקת אותו מהלוח, הפעולה מחזירה 0 אם החתיכה נמחקה בהצלחה, 1 אם התא ריק ולכן אין חתיכה למחוק, 2 החתיכה שייכת לאותו השחקן שמנסה למחוק אותה ולכן זה לא אפשרי, 3 החתיכה לא יכולה להמחק, כנראה כי היא נמצאת בשלשה, 4 החתיכה נמחקה בהצלחה והשחקן שמחק את החתיכה ניצח.
- **CheckWin()** - פעולה אשר בודקת אם שחקן כלשהו ניצח ומחזירה ערך מטיפוס שלם, 0 אם אף אחד לא ניצח, 1 אם הראשון ניצח, 2 השני ניצח.



Nine Men Morris By Tom Kark

- **AllPossibleMoves(int p)** - פעולה שמקבלת משתנה מטיפוס שלם p ומחזירה רשימה של Move של כל המהלכים האפשריים של שחקן p.
- **PossibleMoves(MorrisNode src)** - פעולה המקבלת משתנה מטיפוס MorrisNode ומחזירה רשימה של MorrisNode של כל התאים האפשריים שניתן להזיז אליהם את src.
- **PossibleRemoves(int p)** - פעולה המקבלת משתנה מטיפוס שלם p ומחזירה רשימה של MorrisNode של כל התאים שיש בהם חתיכה של האויב של p וגם ניתן למחוק אותם.
- **CreateOrDeleteOrVerifyMill(MorrisNode node, int create = 0)** - הפעולה מקבלת משתנה מטיפוס MorrisNode ומשתנה מטיפוס שלם create אשר בבירור המחדל שווה ל 0, הפעולה עושה מספר דברים עבור create=1,2,0, עבור 0 היא מבטלת את השלשה שnode נמצא בה, עבור 1 היא יוצרת שלשה לnode במידה והוא מתאים לקריטריונים של שלשה, 2 בודקת שnode אכן בתוך שלשה ומאשרת שאין צורך לבטל זאת.
- **MoveBack()**-הפעולה מחזירה למצב הקודם של הלוח ומוציאה מהמחסנית של המהלכים את המהלך הכי אחרון, הפעולה מחזירה רשימה של MorrisNode, התאים שנמצאים ברשימה הם התאים ששוננו בשחזור של הלוח, הform מטפל ברשימה הזו, במידה והשחזור לא עבד, לדוגמא המחסנית הייתה ריקה, אז הפעולה תחזיר null.



Nine Men Morris By Tom Kark

המחלקה – BoardAB

משמשת כמחלקה שמנהלת את כל ענייני האלפא בטא וחישובי המחשב, משתמשת בפעולות מן מחלקת Board על מנת לבצע מהלכים.

המחלקה יורשת מהמחלקה Board.

תכונות: אין תכונות

פעולות:

- **BoardAB(int layer=layers)** - פעולה בונה של המחלקה, ומאתחלת את הלוח ובונה אותו.
- **Evaluate1()** - פעולה אשר מחשבת ציון בעזרת ההיוריסטיקה הפשוטה של מספר החיילים ומחזירה את הציון הזה, מטיפוס ממשי.
- **Evaluate2()** - פעולה אשר מחשבת ציון בעזרת ההיוריסטיקה המורכבת יותר ומחזירה את הציון הזה, מטיפוס ממשי.
- **CompPlay(int eval1, int eval2)** - פעולה אשר מקבלת שני טיפוסים שלמים שמייצגים איזה היוריסטיקה משומשת עבור כל שחקן, מחשבת את המהלך הכי טוב לשחקן הנוכחי, ומחזירה מערך מטיפוס MorrisNode של החתיכות והתאים שישתנו במהלך הכי טוב שנבחר.
- **Elapsed(DateTime start)** - פעולה אשר מקבלת משתנה מטיפוס DateTime ומחשבת כמה זמן עבר בסטופר.
- **BestMove(int eval1, int eval2)** - פעולה אשר מקבלת 2 משתנים מטיפוס שלם שמייצגים את ההיוריסטיקה עבור כל שחקן, הפעולה מחשבת באמצעות אלפא בטא ופעולת האלפא בטא את המהלך הכי טוב ומחזירה אובייקט מטיפוס MOVE.
- **AlphaBeta(int depth, int playerAB, double alpha, double beta, int eval1, int eval2)** - פעולה אשר מקבלת 6 משתנים מטיפוס שלם שמייצגים את ההיוריסטיקה עבור כל שחקן, אלפא ובטא, עומק האלפא בטא בעץ ומה השחקן שתורו כרגע באלפא בטא. הפעולה מחשבת את המהלך הכי טוב באמצעות אלפא בטא וממשיכה רקורסיבית עד שעוצרת בגלל עומק מוגבל או זמן מוגבל ומחזירה את הציון של המהלך הכי טוב, מספר ממשי.



Nine Men Morris By Tom Kark

- **AdvancedPlayHandler(Move m)** - פעולה אשר מקבלת אובייקט מטיפוס Move

ועושה בלוח את המהלך הזה, הפעולה מחזירה אמת אם המהלך הזה נותן ניצחון לשחקן שעושה אותו, ושקר אחרת.

- שאר הפעולות בלוח הן פעולות שקשורות להיוריסטיקת המאמר המורכבת יותר ולכן אין צורך להסבירם, מימשתי אותם עבור ההיוריסטיקה הזאת בלבד.

```
public int numOfBlockedPieces(int p) //Counts the amount of blocked pieces for player p...
4 references
public int numOf2PieceConfiguration(int p) // Counts the number of 2 piece configurations for player p...
1 reference
private int numOf2ConfigHelp(MorrisNode node)...
4 references
public int numOf3PieceConfiguration(int p) // Counts the number of 3 piece configurations for player p...
1 reference
private bool numOf3ConfigHelp(MorrisNode node)...
2 references
public int doubleMorrisCount(int p) //Checks how many double morrises are there for player p...
1 reference
private bool doubleMorrisHelp(MorrisNode node)...
```



Nine Men Morris By Tom Kark

המחלקה – Const

משמשת כמחלקה ששומרת את כל הקבועים החשובים למשחק, וכל מיני ערכים שאסור לשנותם.

תכונות:

- **location** - משתנה מטיפוס enum אשר שומר את ערך המיקומים של תאים בכל מסגרת, לדוגמה התא העליון ביותר באמצע ערכו 0, וכאשר מתקדמים עם כיוון השעון הערך של כל מיקום עולה ב1.
- **rnd** - משתנה מטיפוס Random אשר אחראי להגרלת המספרים האקראיים למיניהם
- **SIZEEX, SIZEY** - שני משתנים מטיפוס שלם אשר אחראים להגבלות גודל הלוח בתוך חלון המשחק עצמו
- **GAMMA** - משתנה מטיפוס ממשי אשר אחראי למשתנה גאמא באלפא בטא שמהדק את הציונים
- **WinVal** - משתנה מטיפוס שלם ששומר את הערך המנצח שיש לתת למהלך מנצח
- **MaxDepth** - עומק מקסימלי המגביל את האלפא בטא בבחינת העץ
- **layers** - משתנה מטיפוס שלם אשר שומר את כמות השכבות המקורית של המשחק
- **tileSize** - משתנה מטיפוס שלם אשר שומר את גודל כל תא, כל תא הוא ריבוע לכן זה גודל יחיד
- **piecesToPut** - משתנה מטיפוס שלם שקובע כמה חתיכות כל שחקן רשאי להניח בתחילת המשחק
- **TIMELIMIT_PLAY** - משתנה מטיפוס שלם ששומר את כמות הזמן שיש לתת לאלפא בטא לרוץ במידה ובחרנו בריצה על פי זמן ולא על פי עומק.
- **layerWidthX, layerWidthY** - משתנים אשר אחראים לגדלים השונים ביניהם של המסגרות בגרפיקה.



Nine Men Morris By Tom Kark

המחלקה – GameVisuals

משמשת כמחלקה שמנהלת את כל מה שקשור לגרפיקה ולניהול התוכנה.

מכיוון שבמחלקה זו יש המון תכונות והמון פעולות אפרט רק על הדברים הפרקטיים

והחשובים!

תכונות:

- **Board** - אובייקט מטיפוס BoardAB אשר מייצג את הלוח של המשחק
- **isPC** - משתנה מטיפוס בוליאני אשר משמש כמשתנה עזר לגרפיקה וניהול הלוח ומבדיל בין שחקן אמיתי לשחקן המחשב, ערך אמת כאשר תור המחשב לשחק ושקר אחרת.

פעולות:

- **GameVisuals()** - פעולה בונה של המחלקה, הפעולה מאתחלת את המחלקה ואת תכונותיה ואת לוח המשחק.
- **CreateSideTables()** - הפעולה יוצרת את המסגרות מעל ומתחת ללוח ששומרות בתוכן את החתיכות אשר כל שחקן אמור להניח.
- **DrawBoard()** - הפעולה אחראית על לצייר את הלוח ואת המסגרות וכל מה שבתוך האזור של הלוח ונעזרת בפעולות הבאות על מנת ליישם זאת, הפעולה אינה מחזירה כלום.
 - **DrawSquare(int layer, int x, int y, int width, int height, MorrisNode startMid DrawNode(MorrisNode node)**
- **changeSize(object sender, EventArgs ea)** - הפעולה מקבלת פרמטרים דיפלוטיבים של WinForms ומאתחלת את המשחק מחדש ע"פ הגודל הנתון באחת תיבות הטקסט, הפעולה לגמרי מאתחלת את המשחק מחדש, אינה מחזירה דבר.
- כל הפעולות שאצין אלו פעולות אשר מנהלות את המשחק גרפית, הן אלו שמתקשרות עם הלוח מאחורי הקלעים ומנהלות את המשחק בהתאם בצורה הגרפית.
 - **PlaceGraphical(MorrisNode src, MorrisNode dest)**
 - **GraphicalUnfocus(MorrisNode node)**



Nine Men Morris By Tom Kark

`GraphicalMoveBack(object sender, EventArgs ea)` ○

`GraphicalRemove(MorrisNode node)` ○

`GraphicalWinner()` ○

`GraphicalChoose(MorrisNode node)` ○

- `ClickHandler(object sender, System.EventArgs e)` - הפעולה מקבלת פרמטרים דיפולטיבים של WinForm ומנהלת את כל מה שקשור ללחיצות על הלוח ועל מסך התוכנה, זאת הפעולה שאחראית על ניהול אילו פונקציות אמורות להתבצע על פי לחיצות ובקשות המשתמש. הפעולה אינה מחזירה דבר.
- `CompPlayGraphic(bool isCPUPlay = false)` - פעולה אשר מקבלת פרמטר יחיד שמאותחל כברירת מחדל לfalse, הפעולה משחקת בשביל המחשב ומבצעת את השינויים הגרפיים בהתאם לבקשת המחשב. הפעולה אינה מחזירה דבר.
- `UpdateAndShowBoard()` - הפעולה מעדכנת את הלוח עם הטקסטים והאינפורמציה המתאימה בנוסף על כך מעדכנת את כל הגרפיקה למצב הנוכחי ע"פ בקשות הפונקציות המצויינות לעיל. הפעולה אינה מחזירה דבר.
- `hintClick(object sender, EventArgs e)` - הפעולה מקבלת פרמטרים דיפולטיבים ומציגה לשחקן רמזים בהיבט הגרפי, הפעולה אינה מחזירה דבר.



Nine Men Morris By Tom Kark

המחלקה – IndexHooks

משמשת כמחלקה המתחזקת את מבני הנתונים של הלוח וכהמחלקה הבונה ומקשרת בין כל התאים על הלוח.

תכונות:

- **hooks** - אובייקט מטיפוס MorrisNode אשר שומר את השורש של כל מבני הנתונים, זה התא העליון ביותר באמצע, בשכבה הפנימית

פעולות:

- **CreateLayer(int layer)** - פעולה המקבלת מספר מטיפוס שלם ומחזירה אובייקט מטיפוס MorrisNode ובעצם בונה את השכבה הנתונה בערך הפרטמטר
- **LinkLayer(MorrisNode tree, MorrisNode newLayer)** - פעולה המקבלת שני אובייקטים מטיפוס MorrisNode ומחזירה אובייקט מאותו טיפוס, הפעולה מקשרת בין שני השכבות
- **CreateTree(int layers)** - פעולה אשר מקבלת משתנה מטיפוס שלם ולא מחזירה כלום, הפעולה בונה כל שכבה בצורה איטרטיבית
- **IndexHooks(int layer = Const.layers)** - הפעולה הבונה של המחלקה, מקבלת משתנה מטיפוס שלם עם ערך ברירת מחדל שנתון במחלקה Const.



Nine Men Morris By Tom Kark
[המחלקה – MorrisButton](#)

משמשת כמחלקה אשר מייצגת את הכפתורים (תאים) על הלוח, המחלקה יורשת מן המחלקה Button.

תכונות:

- [ancestor](#) - אובייקט מטיפוס MorrisNode אשר שומר את הnode אשר מתאים לאותו כפתור
- [graphicChange](#) - משתנה מטיפוס בוליאני שמשמש כמשתנה עזר בשינויים גרפיים בלוח בין מהלכים, ערכו אמת עבור המצב בו הכפתור צריך לעבור שינוי גרפי ושקר אחרת.



Nine Men Morris By Tom Kark

המחלקה – MorrisNode

משמשת כמחלקה אשר מייצגת את התאים במשחק.

תכונות:

- **selected** - משתנה מטיפוס בוליאני אשר ערכו אמת עבור התא הנוכחי עבר שינוי ואחרת שקר
- **isInMill** - משתנה מטיפוס בוליאני אשר ערכו אמת אם התא הנוכחי נמצא בתוך שלשה אחרת שקר
- **place** - משתנה מטיפוס location אשר מייצג את המיקום של החתיכה הנוכחית
- **near1** - אובייקט מטיפוס MorrisNode אשר מייצג את השכן הבא אחריו עם כיוון השעון
- **near2** - אובייקט מטיפוס MorrisNode אשר מייצג את השכן הבא אחריו נגד כיוון השעון
- **inner** - אובייקט מטיפוס MorrisNode אשר מייצג את השכן הפנימי לו
- **outer** - אובייקט מטיפוס MorrisNode אשר מייצג את השכן החיצוני לו
- **layerNum** - משתנה מטיפוס שלם אשר מייצג את מספר השכבה אשר התא הנוכחי נמצא בו
- **btn** - אובייקט מטיפוס MorrisButton אשר מייצג את הכפתור על הלוח המקושר לתא הנוכחי
- **occupied** - משתנה מטיפוס שלם אשר מייצג את המספר של השחקן שהתא הזה שייך לו, לדוגמא אם התא הזה הוא חתיכה של השחקן הראשון אז ערך המשתנה הינו 1.



Nine Men Morris By Tom Kark

פעולות:

- **MorrisNode()** - פעולה בונה אשר אינה מקבלת שום פרמטרים ומאתחלת את המחלקה.
- **MorrisNode(location place, int layerNum)** - פעולה בונה אשר מקבלת 2 פרמטרים, הראשון הוא משתנה מטיפוס location והשני משתנה מטיפוס שלם, הפעולה מאתחלת את המחלקת ואת תכונותיה.
- **HasInner()** - פעולה אשר מחזירה משתנה מטיפוס בוליאני, הפעולה בודקת אם לתא הנוכחי יש שכן חיצוני לו.
- **HasOuter()** - פעולה אשר מחזירה משתנה מטיפוס בוליאני, הפעולה בודקת אם לתא הנוכחי יש שכן פנימי לו.



Nine Men Morris By Tom Kark

המחלקה – Move

המחלקה מנהלת ומתחזקת מהלכים בעיקר בשימוש של שחזור מהלכים, היא יודעת בכל מהלך מי זז, לאן זז, מהיכן זז ומי אכל וכו'.

תכונות:

- **src** - אובייקט מטיפוס MorrisNode שמייצג את התא אשר ממנו הזזנו במהלך
- **dest** - אובייקט מטיפוס MorrisNode אשר מייצג את התא אליו אנחנו זזים
- **ate** - אובייקט מטיפוס MorrisNode אשר מייצג את התא שבמהלך אכלנו
- **ateWasMill** - משתנה מטיפוס בוליאני אשר ערכו אמת אם במהלך אכלנו חתיכה שהייתה בתוך שלשה ואחרת שקר
- **phase** - משתנה מטיפוס שלם אשר מייצג את השלב במשחק כאשר ביצענו את המהלך

פעולות:

- **Move(MorrisNode src, MorrisNode dest, MorrisNode ate, int phase)** - הפעולה הבונה של המחלקה, מקבלת 3 אובייקטים מטיפוס MorrisNode ומשתנה אחד מטיפוס שלם ומאתחלת את המחלקה ותכונותיה.
- **SetAte(MorrisNode ate, bool wasMill 0)** - פעולה המקבלת אובייקט מטיפוס MorrisNode ומשתנה מטיפוס בוליאני, הפעולה קובעת את ערכי התכונות ate, ateWasMill ולא מחזירה דבר.



Nine Men Morris By Tom Kark

המחלקה – MoveScore

המחלקה מנהלת ומתחזקת ציוני מהלכים בשביל האלפא בטא.

תכונות:

- **move** - אובייקט מטיפוס Move שמייצג מהלך כלשהו.
- **score** - משתנה מטיפוס ממשי אשר מייצג את הציון של המהלך.

פעולות:

- **MoveScore(Move move, double score)** - פעולה בונה אשר מקבלת אובייקט מטיפוס Move ומשתנה מטיפוס ממשי אשר מאתחלת את המחלקה ואת תכונותיה.
- **CompareTo(MoveScore other)** - פעולה אשר מקבלת אובייקט מטיפוס MoveScore ומשווה אותו לאובייקט הנוכחי, מחזירה ערך מטיפוס שלם.



Nine Men Morris By Tom Kark

ייצוג בסיס הידע

בפרויקט שלי השתמשתי בהמון טכניקות ודרכים שונים לממש אותו, הדבר הראשון שלדעתי עליי להסביר זה המימוש של מבנה הלוח.

הרי השאלה הראשונה שעולה בעת תכנון הלוח זה איך בדיוק ממשים את זה? הרי בכל שכבה יש לכל תא גם שכנים מלמעלה, למטה, ימין, שמאל אבל לא לכל אחד יש את כולם. הרעיון שלי היה יעיל במיוחד ביישום שלו. הלוח עצמו מאוד מסובך ולא ניתן לממש אותו במטריצה כמו שאפשר במשחקים אחרים.

תחילה השתמשתי בenum של מיקום כפי שהסברתי באפיון המערכת, נתתי לכל נקודה על הלוח ערך מספרי אשר מתרגם לערך של מיקום, לדוגמא $TopMid=0$ כלומר תא שנמצא במיקום 0 הוא התא הנמצא הכי למעלה בשכבתו באמצע. לאחר מכן $TopRight=1$ ואז $RightMid=2$ וכן הלאה, כפי שניתן להבין יש כאן תבנית של סדרה עולה שמייצגת את מיקום התאים, כלומר ניתן להסתכל על זה כשעון, כאשר מעלים את המיקום ב1 אז אנחנו זזים אחד קדימה בכיוון השעון, ולכן ככה אנחנו מטפלים בפשטות וביעילות בבעיה של שכנים שנמצאים איתנו באותה המסגרת.

נשים לב שבכל שכבה יש לנו 4 תאים אמצעים, כל תא אמצעי מקושר לפחות לשכבה אחת, ויש גם שכבות שבהן התאים האמצעים מקושרים ל2 שכבות.

בניתי מן גרף לא מכונן בעל דרגת יציאה של 4 לכל היותר, כל תא בוודאות מכיל לפחות 2 שכנים ובמקרים מסוימים 3 ובמקרים אחרים גם 4. לכן לקחתי את מחלקת Node הידועה שלמדנו בבית הספר והרחבתי אותה, עכשיו כל צומת יכול להצביע עד 4 צמתים אחרים.

לאחר מכן בניתי את הלוח בצורה מאוד פשוטה, כתלוי במספר השכבות, הרי שדרך הבנייה שלי תומכת בלוח מודולארי, מתחילים מהשכבה הפנימית ביותר, יוצרים את הTopMid ומתקדמים לפי כיוון השעון עד שנחזור אליו, לאחר מכן מקשרים אותו לשכבה חדשה חיצונית ובונים את הלוח איטרטיבית באותה דרך, בעת כל סיום שכבה נשדך גם את שאר האמצעים שנשארו בו לשכבה החיצונית לו והשכבה הפנימית לו ובכך סיימנו את הבנייה.

כפי שניתן לשים לב, בפרויקט שלי ישנה אפשרות נוספת והיא לחזור שלב אחורה במשחק, מימנתי פעולה זו עבור השחקנים אשר עשו מהלך בטעות או אשר התחרטו במהלך שלכם מספיק מהר, וגם כי בשימוש ALPHA BETA ישנו צורך לחזור אחורה מהלכים מכיוון



Nine Men Morris By Tom Kark

שהוא מסתכל קדימה במשחק. במימוש החזרה השקעתי מחשבה רבה על מנת לחשוב איך אממש זאת בצורה הפשוטה והיעילה ביותר בהתחשב בכך שלא נידע את כמות המהלכים שיתבצעו במהלך המשחק, לבסוף הבנתי שניתן להשתמש במחסנית מכיוון שדרך העבודה שלה תואמת לדרך של המשחק, אני שומר כל מהלך במחסנית וכאשר אני רוצה לחזור שלב אחורה אני מוציא את המהלך מהמחסנית ומשחזר אותו למצב הקודם של הלוח, בשימוש המחסנית אני מצליח לא להתייחס לכמות המהלכים וגם סיבוכיות השמירה קבועה והשחזור בממוצע קבוע אך ליניארי במספר התאים במקרה הגרוע, מאוד מרשים.

בסופו של דבר הפרויקט כולל כמה מערכים פשוטים בשביל שמירת נתונים פשוטה, כמה רשימות על מנת לנהל מהלכים, גרף 4 כיווני אשר מתאר את הלוח בכללותו ומחסנית שמתארת את מהלכי המשחק הקודמים.



Nine Men Morris By Tom Kark

תיאור אלגוריתם המחשב

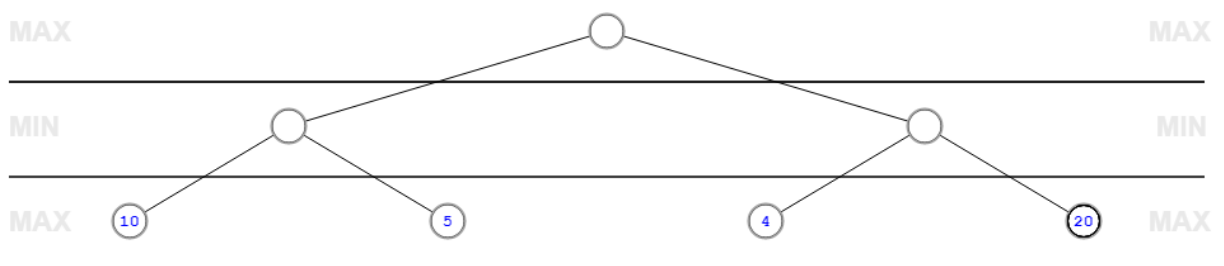
בפרויקט שלי השתמשתי באלגוריתם Alpha-Beta שמתבסס על אלגוריתם MiniMax רק שאלפא בטא משנה אותו מעט ומייעל אותו.

אלגוריתם MiniMax סורק את כל המהלכים ומתקדם במשחק וירטואלית, כלומר המשחק עצמו לא באמת מתקדם אבל האלגוריתם חושב קדימה ומתקדם, האלגוריתם בונה מן עץ מהלכים אשר ממנו הוא מחזיר מסלול אופטימלי ביותר בהנחה שכל שחקן משחק בצורה האופטימלית ביותר, בתחתית העץ (כאשר מגיעים למהלך מנצח או לעומק אשר האלגוריתם מוגבל על ידו) האלגוריתם נותן למהלך ציון היוריסטי ע"פ חישובים כלשהם שאסביר בעמודים הבאים ובעזרת ציונים אלא חוזר במעלה העץ ובוחר מהלכים אופטימליים.

כל רמה בעץ היא או מינימום או מקסימום, מכיוון שכל פעם שנרד צומת התור יתחלף לשחקן אחר ולכן יש צורך להפריד כל רמה בין מקסימום ומינימום.

אלגוריתם אלפא בטא עובד בדיוק באותה דרך רק שבמהלך הסריקה, הוא יכול להחליט במצבים מסוימים שלא צריך להמשיך לסרוק צמתים אחרים ובכך חוסך זמן, לתהליך זה קוראים גיזומים, אביא דוגמא פשוטה לעץ כזה ואסביר קודם כל איך MiniMax יעשה אותו ולאחר מכן אראה את אלפא בטא.

תחילה ניקח עץ לדוגמא:

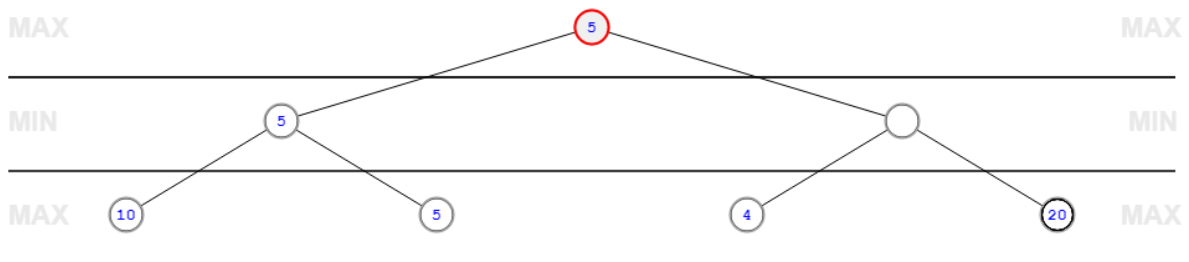




Nine Men Morris By Tom Kark

בתחילת הסריקה, המקסימום משחק, אנחנו עושים סריקת InOrder, משמע הולכים לבן השמאלי של הצומת רקורסיבית, לאחר מכן לצומת עצמו, ואז רקורסיבית לבן הימני של הצומת. נסתכל על הבן השמאלי של השורש, ונשים לב כי זה תורו של חום לשחק, ולכן עליו לבחור את הערך הקטן מבין, במקרה זה בניו הם עלים ולכן הם סוף העץ, נרד קודם לעלה השמאלי, ערכו 10, ונשמור אותו אצל אבא שלו לאחר מכן נבדוק את העלה הימני ונשווה בינו לבין הערך של אבא שלו, נבחר את הקטן מביניהם שזהו 5 ולאחר מכן נעלה את הערך הזה לשורש

כעת נלך מהשורש ימינה, ונעשה את אותו דבר עבור בנו הימני של השורש, נקבל שערך הבן הימני של השורש הוא 4, מכיוון שהשורש נמצא ברמה של מקסימום, נדרש עליו לבחור את



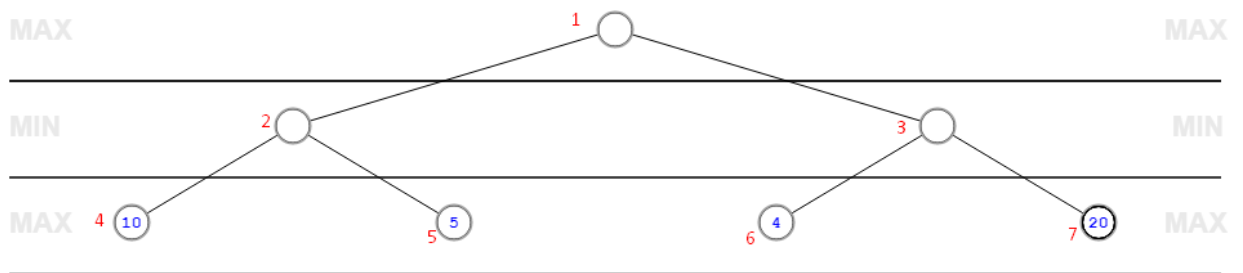
הגדול מבין, אז כעת נשווה בין ערכו של השורש (ערכו של הבן השמאלי שכבר העלנו) ובין ערכו של הבן הימני ונגלה כי ערך השורש עכשיו הוא 5 ולכן המסלול האופטימלי הוא מהשורש אל העלה שערכו 5.



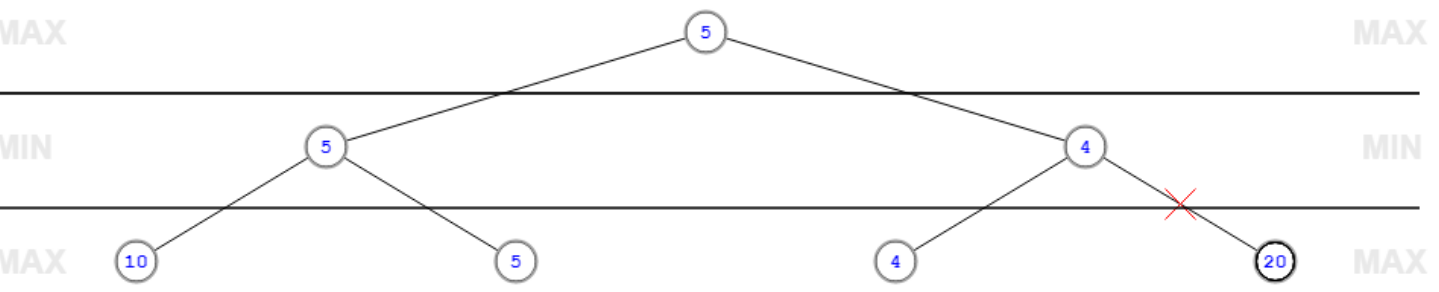
Nine Men Morris By Tom Kark

אלגוריתם אלפא בטא עובד כמעט זהה לחלוטין רק שהוא משתמש בשני משתנים, אלפא ובטא אשר איתם הוא סורק יותר חכם וגוזם צמתים מסוימים.

נגדיר ערך $\beta = \infty$, $\alpha = -\infty$, כאשר נהיה ברמת מקסימום, ונבחר ערך כלשהו מהבנים, אם הערך גדול מאז נעדכן את α להיות ערך זה, כאשר זאת רמת מינימום, אם הערך קטן מ β אז נעדכן את β להיות ערך זה, במידה ונוצרת סיטואציה שאנחנו בצומת כלשהי ו $\alpha \geq \beta$ אז אין טעם לבדוק את שאר הבנים של הצומת מכיוון שבדיקה זו לא תשפיע בכלל על התוצאה, נשים לב לאותה הדוגמא של המינימקס.



נעשה כרגיל, נרד לבן השמאלי של השורש ונשים לב כי ערכי אלפא ובטא עדיין מינוס אינסוף ואינסוף בהתאמה, עכשיו נרד לצומת בעלת הערך 10 (צומת מספר 4), מכיוון שצומת מספר 2 ברמת \min אז יש לעדכן את β שלה להיות מ 10 מכיוון ש $10 < \infty$. ולכן בצומת זו $\alpha = -\infty$, $\beta = 10$, מכיוון ש $\alpha < \beta$ נמשיך גם לבדוק את צומת 5, מכיוון ש $5 < 10$ אז נעדכן את צומת מספר 2 להיות בעלת הערכים: $\alpha = -\infty$, $\beta = 5$, כעת נחזור לשורש ונשים לב שהשורש הוא ברמת מקסימום, לכן יש לעדכן את ערכי האלפא בטא שלו להיות: $\alpha = 5$, $\beta = \infty$, מכיוון שעריך בנו השמאלי של השורש היה גדול מערך האלפא של השורש. נמשיך לבן הימני (צומת מספר 3), הוא יורש את ערכי האלפא בטא מהשורש ולכן גם אצלו





Nine Men Morris By Tom Kark

$\alpha = 5, \beta = \infty$, נרד לצומת מספר 6 ובבדוק אם ערכו קטן מערך ה β של צומת 3, נראה שזה מתקיים ולכן נעדכן את ערך הבטא של צומת 3, $\alpha = 5, \beta = 4$, מכיוון שמתקיים $\alpha \geq \beta$ נגזום את כל תת העץ של צומת מספר 3 שזה בעצם רק צומת מספר 7 ונעדכן את ערך צומת מספר 3 להיות 4:

עכשיו נבדוק אם ערך הבן הימני של השורש גדול מערך האלפא שלו, נשים לב שמתקיים $5 > 4$ ולכן לא נעדכן אותו וערך השורש הוא 5.

מדוע ביצענו את הגיזום? נשים לב שבשורש מחכה לנו הערך 5, וגם כעת הוחזר לנו הערך 4 מן צומת מספר 6 אל צומת מספר 3, מכיוון שצומת מספר 3 היא ברמת מינימום האלגוריתם ינסה לבחור את הערך הקטן ביותר מבין הבנים, אבל מכיוון שהשורש הוא רמת מקסימום הוא ינסה לבחור את הערך הגדול מבין בניו, מכיוון שערך הבן השמאלי של השורש הוא 5, לא משנה איזה ערך יותר קטן מ4 צומת מספר 3 ייבחר, 5 עדיין ינצח אותו בשורש, ולכן חסכנו בזמן וגם בעבודה.



Nine Men Morris By Tom Kark

היוריסטיקות וחישוב הנקודות למהלך

בפרויקט שלי בחרתי להשתמש ב-2 היוריסטיקות, ההיוריסטיקה הראשונה היא יחסית פשוטה אשר מתייחסת למספר החתיכות שיש לכל שחקן בלבד, כלומר האלגוריתם יעדיף מהלכים אשר או שומרים על מספר השחקנים של השחקן הנוכחי, או מהלכים אשר מורידים את מספר החתיכות של האויב, הוא יעדיף להיות בעל מספר החתיכות הגדול יותר. החישוב הוא ממש פשוט והוא נראה כך:

$\text{numOfPieces} = \text{CountPieces}(2) - \text{CountPieces}(1);$

הערה: חשוב לציין כי בפרויקט שלי maximizer הוא השחקן השני וה- minimizer הוא השחקן הראשון.

הערך שייצא פה הוא ההפרש של מספר החתיכות בין שחקן 2 לשחקן 1, נניח והעץ הגיע לעומק המקסימלי והוא רוצה לחשב את הציון של המצב הנוכחי של הלוח, נסתכל על 2 מצבים:

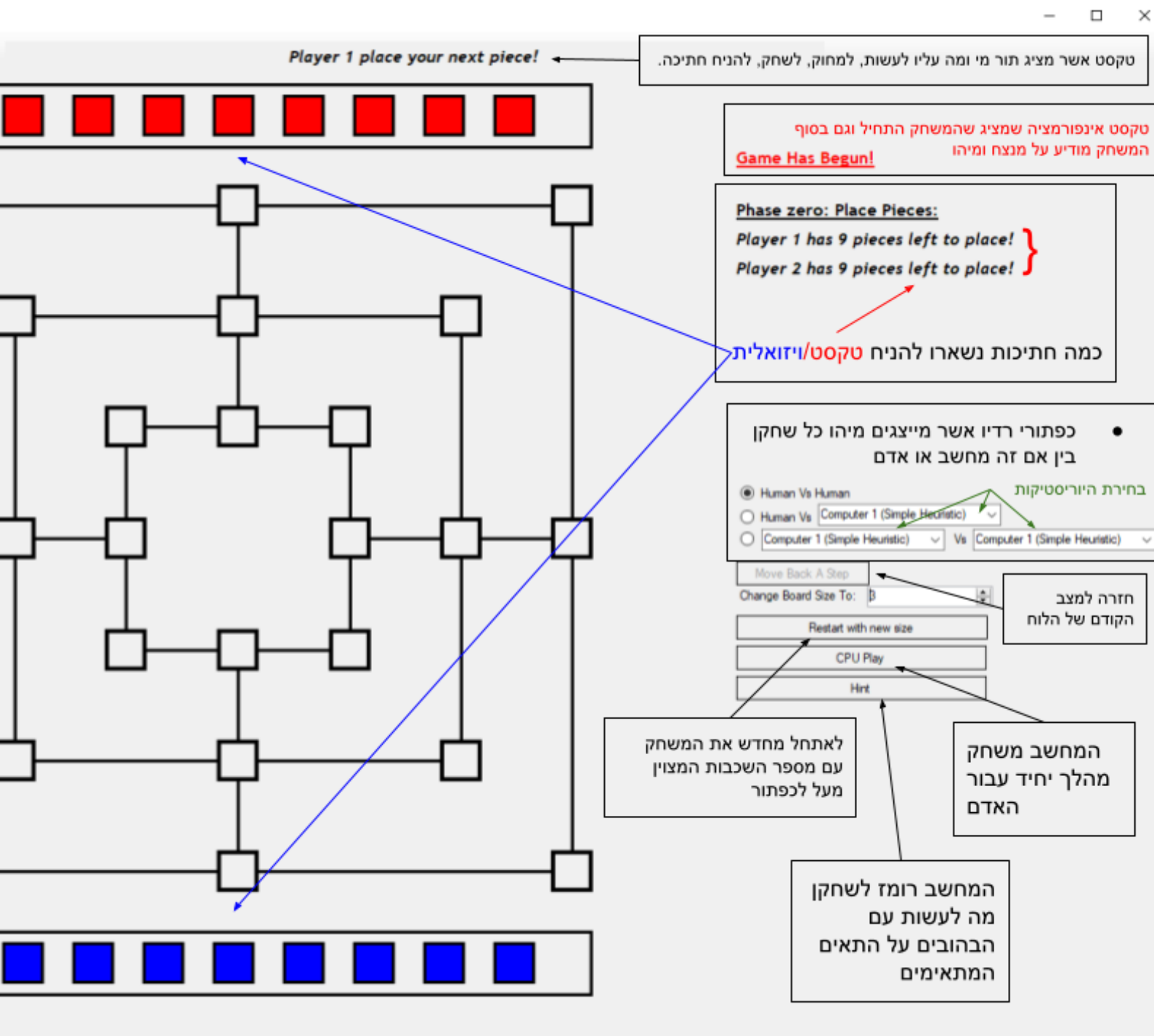
- תור שחקן 1, במידה ויש לו יותר חתיכות אז כמובן שערך ההפרש הוא שלילי וזה מצוין לשחקן מכיוון שהוא מחפש ציון כמה שיותר קטן, אם יש לו פחות חתיכות אז ערך ההפרש הוא חיובי וזה לא טוב לו.
- תור שחקן 2, במידה ויש לו יותר חתיכות אז כמובן שערך ההפרש הוא חיובי וזה מצוין לשחקן מכיוון שהוא מחפש ציון כמה שיותר גדול, אם יש לו פחות חתיכות אז ערך ההפרש הוא שלילי וזה לא טוב לו.

היוריסטיקה השנייה היא היוריסטיקה שמצאתי במאמר אקדמי אשר חקר את תופעות המשחק ובעצם דרך ניתוחים סטטיסטיים ומחקר הוא הצליח לפתח היוריסטיקה אופטימלית. אחד הדברים שרציתי לממש בפרויקט זה אופציה של מחשב נגד מחשב, ולדעתי כאשר מחשב משחק נגד עצמו עם אותה היוריסטיקה זה לא מעניין באותה מידה כמו 2 מחשבים שמשחקים בשיטות שונות, רציתי לבדוק את ההיוריסטיקה של המאמר ולכן מימשתי אותה ומספר רב של פעולות אשר היא דורשת, לא נכנסתי לעומק במשמעות ההיוריסטיקה הזו אבל היא אכן משחקת מאוד מאתגר, מספיק מאתגר.



Nine Men Morris By Tom Kark

כאשר המחשב משחק מול עצמו לפעמים המשחק מתנהל כרגיל עד שנוצרת הסיטואציה ששני השחקנים בשלב 2 (שניהם בעלי 3 חתיכות) והמשחק יכול להמשך המון זמן, לכן הוספתי מנגנון קטן בקוד אשר קולט שלאחר מספר קבוע של מהלכים מסוימים שאף אחד מהשחקנים לא אכל חתיכה, המחשב לא ייבחר את המהלך עם הניקוד הכי גבוה, אלא הוא ייבחר אחד רנדומלי מכל המהלכים שנמצאים בשלושת הציונים הכי גבוהים.



The screenshot shows the game interface for Nine Men Morris. At the top, a status bar says "Player 1 place your next piece!". Below it, a row of red squares represents Player 1's pieces, and a row of blue squares represents Player 2's pieces. The main board is a 3x3 grid of squares. A blue arrow points from the top status bar to the board, and another blue arrow points from the board to the bottom status bar.

Annotations in Hebrew:

- טקסט אשר מציג תור מי ומה עליו לעשות, למחק, לשחק, להניח חתיכה.** (Text that displays whose turn it is and what they should do, to move, to play, to place a piece.)
- טקסט אינפורמציה שמציג שהמשחק התחיל וגם בסוף המשחק מודיע על מנצח ומיהו** (Text that displays information that the game has begun and also at the end of the game informs of the winner and who.)
- Game Has Begun!**
- Phase zero: Place Pieces:**

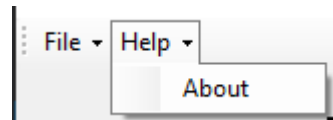
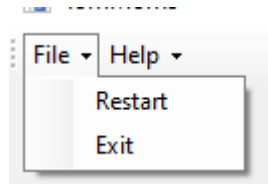
Player 1 has 9 pieces left to place!

Player 2 has 9 pieces left to place!
- כמה חתיכות נשארו להניח טקסט/ויזואלית** (How many pieces are left to place, text/visually)
- כפתורי רדיו אשר מייצגים מיהו כל שחקן בין אם זה מחשב או אדם** (Radio buttons that represent who the player is, whether it's a computer or a human)
- בחירת היוריסטיקות** (Choice of heuristics)
- חזרה למצב הקודם של הלוח** (Return to the previous board state)
- המחשב מחשב מהלך יחיד עבור האדם** (The computer calculates a single move for the human)
- המחשב רומז לשחקן מה לעשות עם הבהובים על התאים המתאימים** (The computer hints to the player what to do with the highlights on the appropriate cells)
- לאתחל מחדש את המשחק עם מספר השכבות המצוין מעל לכפתור** (Reset the game with the number of layers indicated above the button)



Nine Men Morris By Tom Kark

נוסף על כך בפינה השמאלית העליונה בתוכנה ישנו תפריט עם אפשרויות לעשות אתחול מוחלט של התוכנה, יציאה מהתוכנה, ואודות התוכנה.





Nine Men Morris By Tom Kark

רפלקציה

הפרויקט היה הפרויקט הכי מעניין שהיה לי מימי בתקופת לימודי ביסודי, חטיבה ובתיכון, הפרויקט יישם המון עקרונות שלמדנו בתכנית הלימודים הרגילה והוסיף הרבה מעבר וגם דרש בחלקים מסוימים ידע אישי, לדעתי הקשיים העיקריים בפרויקט היו לתכנת debugging מכיוון שלעשות debugging באלפא בטא היה מאוד מתיש ומעייף, בכל זאת, רקורסיה שעוברת אלפי צמתים בעץ וצריך לעשות דיבאגינג לזה, למזלי תכנתי בצורה טובה ולא היו הרבה דיבוגים בכלל באלפא בטא ולכן לא נתקעתי כל כך הרבה זמן, שמת לב לשיפור ברמת העבודה ועשייתה משום שכבר בניתי פרויקטים דומים לכך בעבר אבל לא גדולים בקנה מידה כמו זה. הידע והכלים שרכשתי במהלך העבודה הרחיבו ופיתחו לי את הידע בצורה חד משמעית ואני שמח על כך.

לסיכום, הפרויקט היה מאתגר, דרש לעבור מכשולים לא מעטים, אך התמודדתי איתם בהצלחה והראתי ביצועים מעל ומעבר, פרויקט מעניין, מוצלח ומקווה שהקהל המיועד שינסה אותו יהנה ממנו וינצל אותו למטרות למידה עצמית או שימוש פרקטי וכו'.



Nine Men Morris By Tom Kark

תודות

כל הפרויקט הזה לא היה קורה בלי המנחה שלי אריאל בר יצחק, הוא עזר לי רבות בתכנון הפרויקט והבנת בנייתו, הראה לי את הדרך לפרויקט מוצלח, מאורגן ומושקע ואני מאוד מרוצה מהתוצר הסופי.

תודה נוספת לכל מיני סרטונים אשר נעזרתי בהם במהלך בניית הפרויקט אשר לימדו אותי כל מיני דברים מרתקים אותם הוספתי לפרויקט.



Nine Men Morris By Tom Kark

ביבליוגרפיה

- ויקיפדיה, אודות המשחק וכל מיני מושגים וטכניקות למימושים שונים בקוד הפרויקט
- יוטיוב - סרטונים אשר עזרו לי להבין איך לייעל את צורות הקוד שלי ואת דרך העבודה
- StackOverflow - כאשר נתקלתי בבעיות מורכבות למינהם פניתי לאתר התמיכה הזה על מנת להבין איפה נפלתי בעת כתיבת הקוד