**חוברת למתכנת – איקס עיגול "משודרג"**

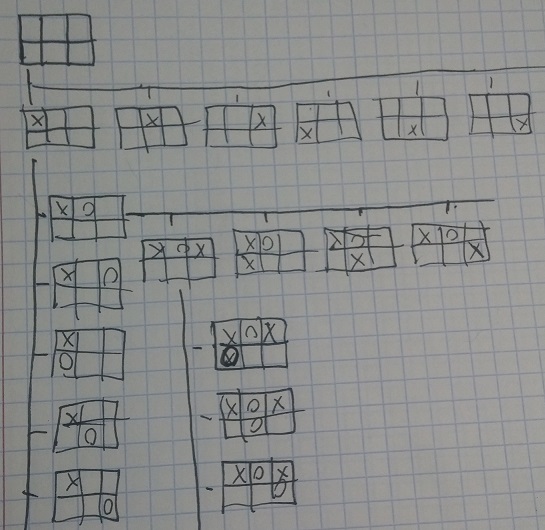
5 עקרונות מרכזיים בקוד:

1. חשיבה צעדים קדימה והחלטה על איזה מהלך לבצע.

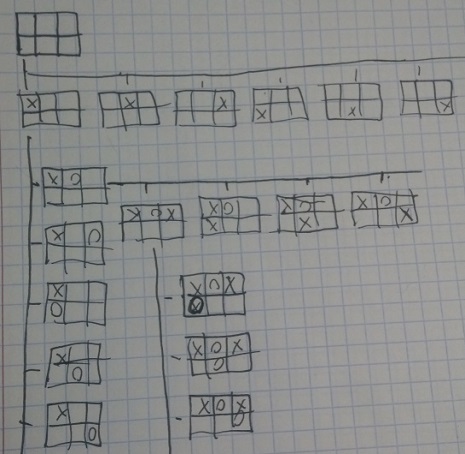
החשיבה של הצעדים קדימה מתבצעת בשתי הפונקציות: FirstComBestMove וComBestMove. הפונקציה FirstComBestMove אחראית על ביצוע הצעד שהמחשב עושה, על ידי בדיקה של מהם המהלכים האפשריים שהמחשב יכול לשחק, ובדיקה כמה "טוב" כל אחד מהם. כדי לעשות זאת המחשב מדמה את המהלך ומשחק כל אחד מהם בכאילו, כלומר מבלי באמת לבצע אותם. לאחר שביצע בכאילו אחד מהמהלכים הוא מוסיף את הניקוד שמהלך זה הביא למערכי הניקוד(אסביר לפי מה הוא נותן את הניקוד בעקרון מאוחר יותר).

לאחר מכן הוא קורא לפונקציה ComBestMove אשר מחזירה לו את הניקוד של מהלך זה באמצעות זה שהיא עושה דבר דומה למה שהוא עשה. היא בעצם משחקת "בכאילו" כל מהלך אפשרי שהשחקן יכול לבצע כתגובה למהלך הראשון ש FirstComBestMove עשה כבר "בכאילו". לאחר שהיא מבצעת אחד מהמהלכים הללו "בכאילו" היא מוסיפה את הניקוד שמהלך זה מוסיף למערכי הניקוד באותה **שיטה כמו** FirstComBestMove**, וקוראת לעצמה שוב, הפעם שתדמה את המהלכים שהמחשב יכול לעשות כתגובה למהלך זה של השחקן. תהליך זה קורה חוזר על עצמו שוב ושוב עד שהפונקציה מגיעה ל"מכסה" של כמות הצעדים קידמה. כלומר אם השחקן הכניס כקלט לתוכנית שהוא רוצה שרמת הקושי(כלומר כמות הצעדים קדימה שהמחשב יחשוב) היא 6, אז לאחר שהתבצעו 6 תורות(התור הראשון יתבצע ב** FirstComBestMov**והחמישה האחרים יתבצעו ב**ComBestMove**) "בכאילו", אז הפונקציה תקרא לפונקציה שאחראית על החישוב הכולל של הלוח על פי הניקוד שמאוחסן במערכי הניקוד ועל פי כך תקבע את הניקוד של המהלכים האפשריים האחרונים בשרשרת. לאחר שנקבע את הניקוד של המהלכים האפשריים שבסוף השרשרת, נבחר איזה אחד מהם לשלוח חזרה לפונקציה שקראה לפונקציה זו(ליתר דיוק זה** ComBestMove **שקוראת לעצמה אבל הכוונה היא לשלוח חזרה את הערך למי שקרא לה) על פי מי שהיא דימתה במהלכים אלו. אם מהלכים אלו מדמים את השחקן אז הוא יבחר במהלך עם הניקוד הנמוך ביותר, מכיוון שהניקוד מתאר את הניקוד של המחשב וככל שלמחשב יש פחות נקודות זה אומר שלשחקן יש יותר, וישלח את הניקוד חזרה לפונקציה שקראה לו(שוב, שולח את זה חזרה לעצמו מכיוון שפונקציה זו קראה לעצמה). במידה ומהלכים אלו מדמים את המחשב הוא יחזיר לפונקציה שקראה לו את הניקוד של המהלך עם הניקוד הגבוה יותר. הנוסף, בלי קשר לאיזו דמות מהלכים אלו מייצגים, תמיד לאחר שהפונקציה**  ComBestMove **קיבלה ניקוד סופי של מהלך מסוים, היא תוריד ממערכי הניקוד את הניקוד שהוסיפה להם מקודם. כמו התהליך בו פונקציה זו קוראת לעצמה, גם תהליך זה, של בחירה מבין כמה מהלכים ושליחה חזרה של הניקוד של המהלך שנבחר, חוזר על עצמו כמה פעמים עד שלבסוף הערכים מוחזרים חזרה** FirstComBestMov **שבוחר במהלך עם הערך הגבוה ביותר ומבצע אותו. נקודה חשובה נוספת היא שהפונקציה לא תקרא לעצמה כדי שמבדוק את המהלכים האפשריים כתגובה למהלך שהיא עושה כרגע, במידה ומהלך זה מביא לניצחון. בנוסף הניקוד של מהלך זה יהיה הניקוד הגובה ביותר שהמשתנה של הניקוד יכול להכיל(הוא מסוג int אז המקסימום של Int).  
בוא נראה הדגמה לכל התהליך, על לוח ריק של 2\*3 ו4 צעדים קדימה, כאשר המחשב מתחיל:**

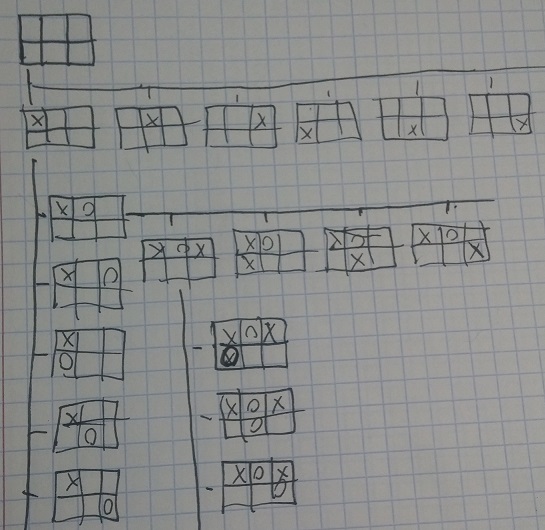
* **קוראים לפונקציה** FirstComBestMov
* **היא בודקת את המהלכים האפשריים שלה ורואה שיש שישה מהם:**

****

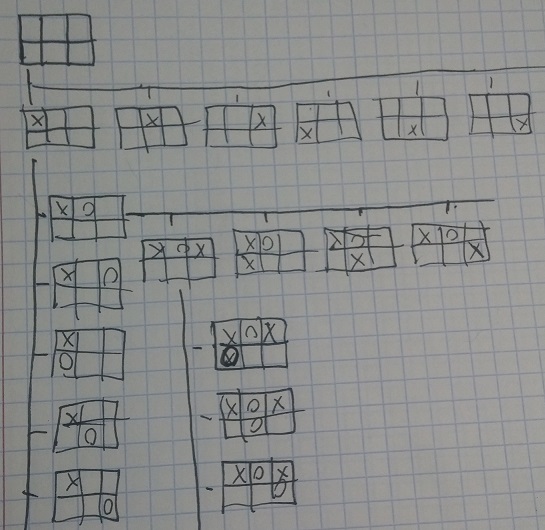
* **היא קוראת לפונקציה** FirstComBestMov **כדי שתחזיר לה מה הניקוד של המהלך האפשרי הראשון**
* **הפונקצייה** FirstComBestMov **בודקת מה הניקוד של מהלך זה על ידי בדיקה של איזה מהלכים אפשריים יש לו ורואה שיש 5:**

****

* **הפונקצייה** FirstComBestMov **קוראת לעצמה כדי לבדוק מה הניקוד של המהלך האפשרי הראשון.**
* **תהליך זה חוזר על עצמו עד שנגיע לזה שכבר סימנו 4 צעדים על הלוח, כלומר כאן:**

****

* **הפונקצייה** FirstComBestMov **תקרא לפונקציה** ScoreSystem **שתחזיר לה את הניקוד של הלוח בכל אחד משלושת המהלכים.**
* **בואו נניח לדוגמא שעיגול לא ניצח ושהפונקציה החזירה את הניקוד הבא:**

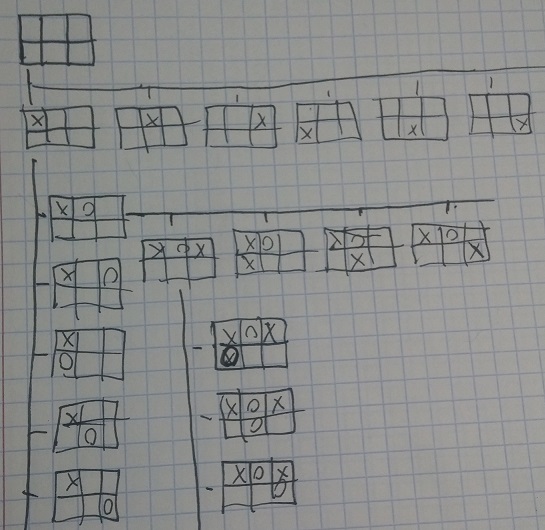
****

80

5

8

* **כעת בגלל שמהלכים אלו דימו את השחקן הערך שיוחזר לפונקציה שקראה לה הוא הערך של הניקוד הנמוך ביותר, 5 במקרה הזה**
* **כעת הניקוד של המהלך שגרם לשלושה מהלכים האלו יהיה שווה לניקוד שהוא קיבל חזרה מהפונקציה וזה 5**

****

5

* **גם הניקוד של המהלכים האפשריים שסמוכים לו יבדקו, עד שלבסוף מהלך זה ישלח חזרה את הניקוד של המהלך עם הניקוד הגבוה ביותר למהלך שקרא לו**
* **לבסוף יוחזר ל**FirstComBestMov **שישה ערכים, והמהלך עם הערך של ניקוד הגדול ביותר יבחר ויוצג על לוח המשחק.**

1. "אלפא בטא"

עקרון זה הוא בעצם מנגנון שנועד לייעל את תהליך החשיבה קדימה של ComBestMove. אם אנו יודעים שבהדמיה שפונקציה זו עושה בעצם המחשב תמיד יבחר בניקוד הגבוה ביותר והשחקן בניקוד הנמוך ביותר, ניתן לחסוך הרבה בדיקות מיותרות של ניקוד על ידי לפסול אותן מראש.

בוא נניח מצב בו למחשב יש 5 מהלכים אפשריים, והוא כבר חישב אחד מהם ויצא לו שהניקוד הוא 6 לדוגמא, וכעת הוא יקרא שוב ל ComBestMove שיבדוק את הניקוד של המהלך האפשרי הבא, בעזרת בדיקת ה4 מהלכים האפשריים שהשחקן יכול לעשות כתגובה. אם הניקוד של אחד מהמהלכים האפשריים של השחקן, יהיה קטן מהניקוד של המהלך האפשרי הקודם של המחשב אשר הוא כבר בדק את הניקוד שלו, כמו 4 לדוגמא אז לא יהיה טעם להמשיך לבדוק את המהלכים האפשריים של השחקן כתגובה למהלך המדומה האחרון של המחשב. זאת מכיוון שידוע לנו שהשחקן תמיד יבחר במהלך עם הניקוד האפשרי הקטן ביותר, ולכן לא משנה איזה ניקוד עוד יצא לו, הוא לעולם לא יהיה יבחר במספר גדול מ4, כלומר זה הערך המקסימלי שיכול השחקן להחזיר. ולכן, התור של המחשב שקרא לשחקן שכבר יצא לו מהלך עם ניקוד 6, לעולם לא יבחר במספר שקטן או שווה ל4, מכיוון שהמחשב תמיד בוחר את המספר הגדול. ולכן בעצם מספיק שיצא בתור השחקן מהלך עם ניקוד קטן מ6 וכבר אין טעם להמשיך לבדוק את המהלכים האפשריים האחרים של השחקן באותו תור מדומה. אותו עקרון תופס גם על התור המדומה של המחשב, אם השחקן המדומה כבר חישב מהלך עם ניקוד 2, וכעת הוא קרא שוב לComBestMove שיבדוק בשבילו מה הניקוד של מהלך אחר באמצעות בדיקת הניקוד של המהלכים האפשריים של המחשב כתגובה. אז אם אחד המהלכים של המחשב יהיה עם ניקוד גדול מ2, לא יהיה טעם להמשיך לבדוק את שאר המהלכים האפשריים של המחשב בתור מדומה זה כי השחקן כבר לעולם לא יבחר באפשרות זו.

1. **ניקוד על פי רצפים, אחרי ביצוע מהלך**

**כדי שנבין איך עובד ניקוד על פי רצפים חשוב להבין תחילה את העקרונות הבסיסיים של ניקוד כאשר עובדים על המשחק הקלאסי של איקס עיגול, לוח בגודל 3\*3 ורצף של 3 כדי לנצח, אשר תוכנית זו מתבססת עליו ועל עקרונותיו.**

* **ניקוד נפרד לכל סוג ניצחון אפשרי, כלומר התוכנית נותנת ניקוד בנפרד לניצחון בשורה - לכל שורה בנפרד, לניצחון בטור – כל טור בנפרד, ניקוד לאלכסון שבא משמאל למעלה לימין למטה, ולאלכסון שבא מימין למעלה לשמאל למטה.**
* **הניקוד של כל x הוא 1 ושל כל עיגול הוא 1000. אין זה אומר שהעיגול פי 1000 יותר חשוב לניקוד מהאיקס, אלא זו פשוט שיטה שמשומשת בקוד כדי שזה יהיה אפשרי לאחסן באותו משתנה מסוג int גם כמה עיגולים וגם כמה איקסים יש באותה שורה\טור\אלכסון.**
* **כמו שניתן להבין מהשורה הקודמת המשמעות של ניקוד של כל שורה\טור\אלכסון הוא בעצם כמה איקסים ועיגולים יש בו.**
* **אחרי כל מהלך – כלומר סימון כפתור עם איקס או עיגול יתווסף ניקוד רק בשורה, טור ולפעמים אלכסון שהכפתור הזה נמצא בתוכם**

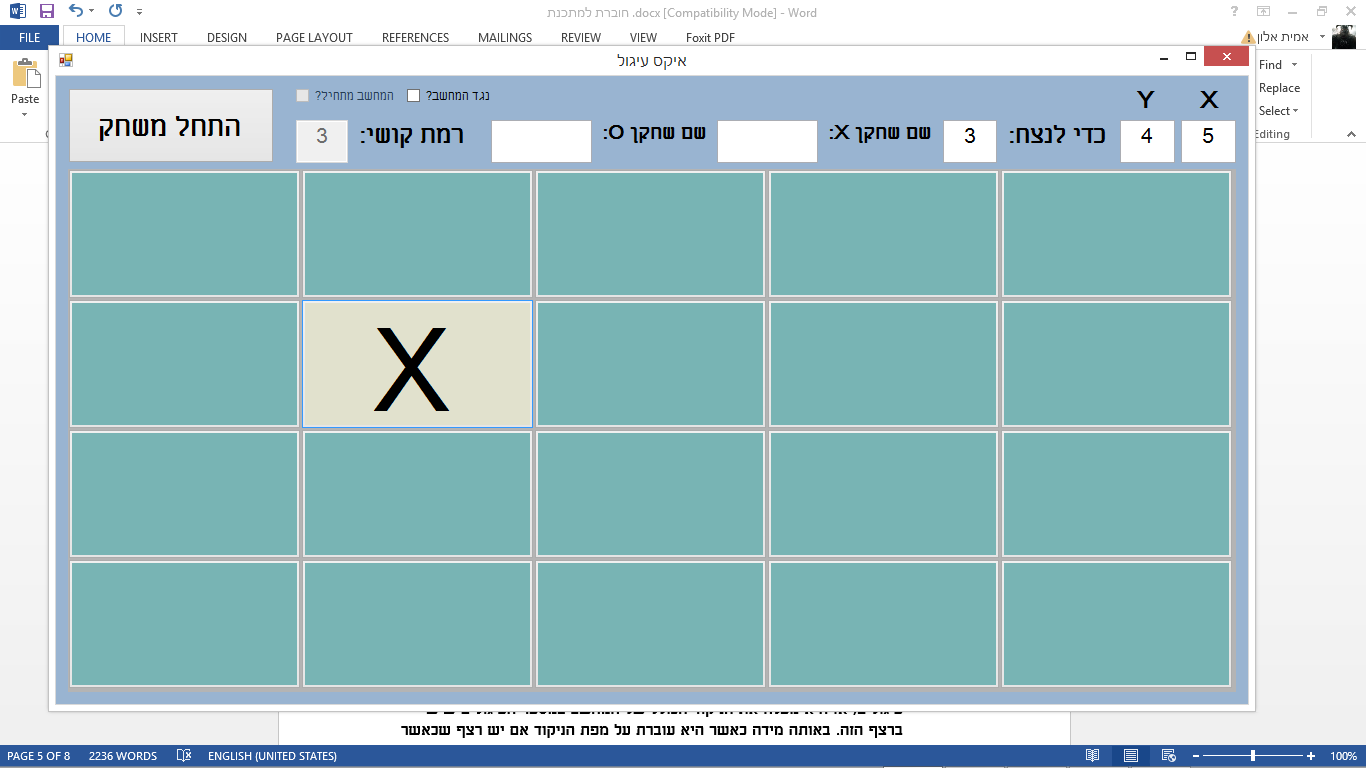
**ניקוד על פי רצפים הכרחי כאשר אנו רוצים לאפשר למחשב לדעת לשחק כאשר הרצף ניצחון קטן מגודל הלוח, וכאשר הלוח מלבני, ולא ריבועי(כלומר יכול להיות 2\*8 או 4\*5 או כל זוג מספרים שלמים חיוביים שלא שווים אחד לשני).**

**כפי שאמרתי בתחילת ההסבר, הניקוד לפי רצף מתבסס על שיטת הניקוד הרגילה, אך במקום ניקוד בנפרד לכל שורה\טור\אלכסון הפעם יש ניקוד נפרד לכל רצף ניצחון אפשרי. למרות שיש לכל רצף ניקוד עדיין יש חלוקה לשורות\עמודות\ואלכסונים במערכים שאחראיים על הניקוד. מערך אחד מתאר את כל הרצפים האפשריים לניצחון בשורה, האחר לכל הרצפים האפשריים בטור, עוד אחד לכל הרצפים האפשריים שנמצאים באלכסון שמתחיל למעלה מימין ונגמר למטה בשמאל, ואחרון לכל הרצפים האפשריים שנמצאים על אלכסון שמתחיל למעלה משמאל נגמר למטה בימין.**

**גם כאן קורה הכל כמו בשיטת הניקוד הבסיסית איך עם שינוי אחד משמעותי.**

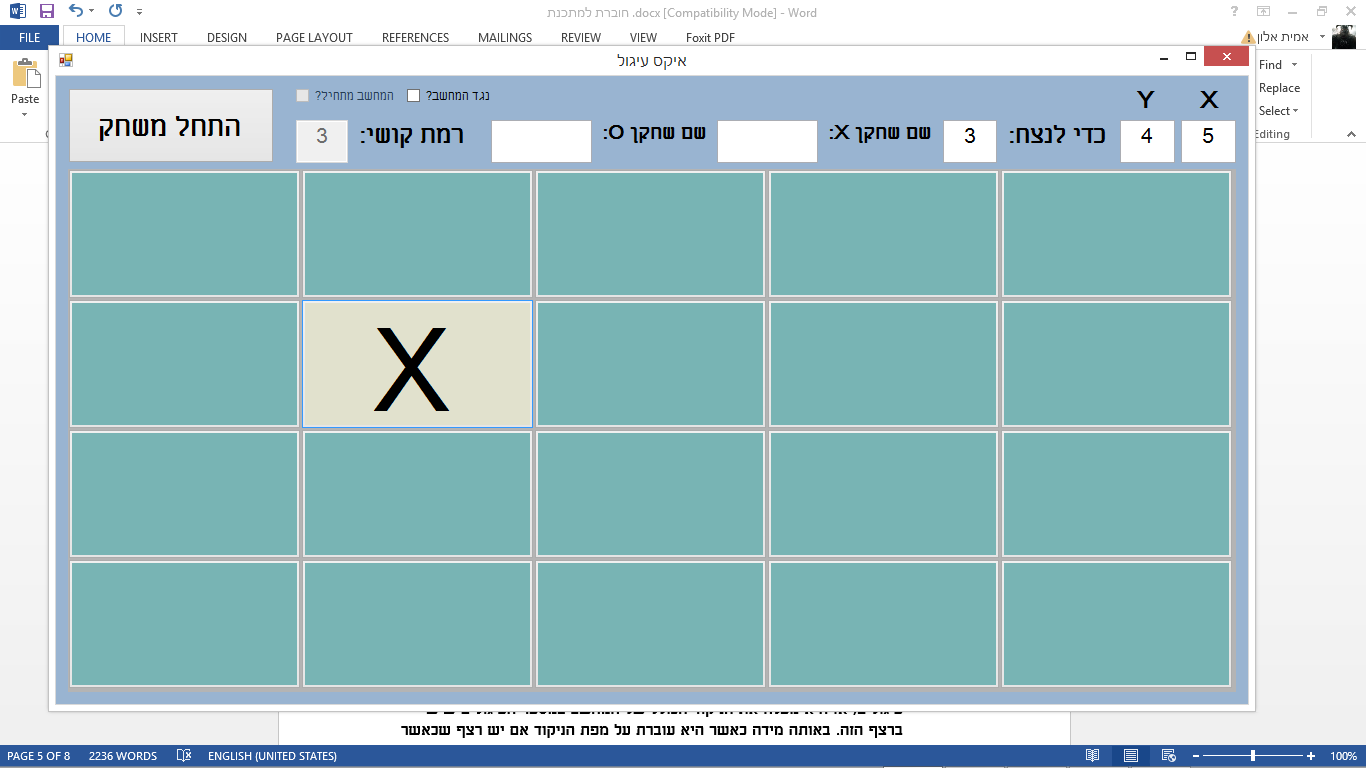
**במקום לתת אוטומטית פלוס 1 או פלוס 1000 כאשר נסמן איקס או עיגול בטור ובשורה ואולי אלכסון שהכפתור מצא עליו, הפעם יש צורך להוסיף 1 או 1000 בכל הרצפים שכפתור זה קשור אליהם(כלומר הרצפים שמכילים אותו) .**

**דוגמא להשפעה של מהלך על מערכי הניקוד:**

**בואו נראה כיצד יראו מערכי הניקוד לאחר ביצוע המהלך הבא בלוח של 5\*4 וכאשר צריך רצף של 3 כדי לנצח:**

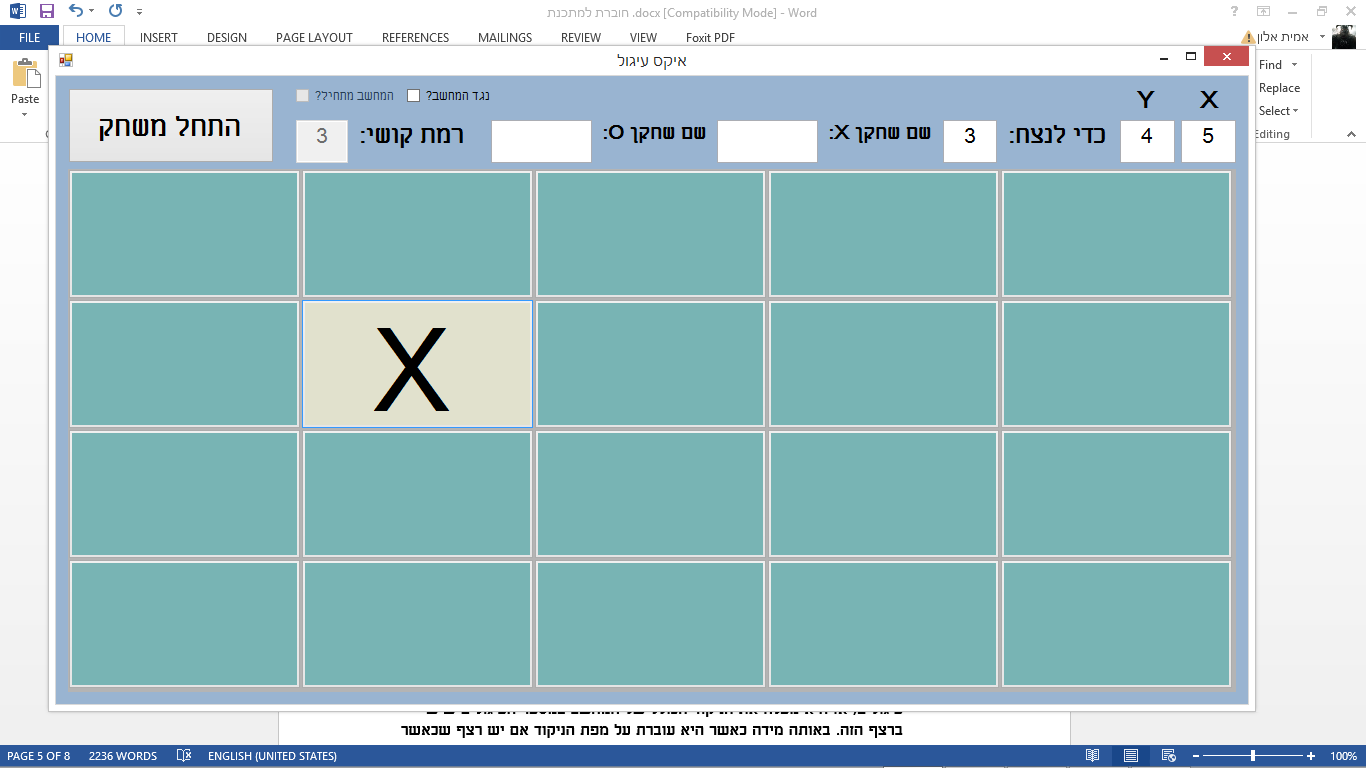
**מערך הניקוד של הרצפים האפשריים בשורה יהיה בגודל 3\*4, מכיוון שיש 4 שורות שבן אפשר לנצח, ובכל שורה יש 3 רצפים אפשריים. המערך יראה כך:**

|  |  |  |
| --- | --- | --- |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **0** | **0** | **0** |
| **0** | **0** | **0** |

**הסיבה שהוספנו 1 בשתי מקומות היא שהמהלך ששיחקנו יכול להשלים 2 רצפים של ניצחון בשורה שונים, והם:**

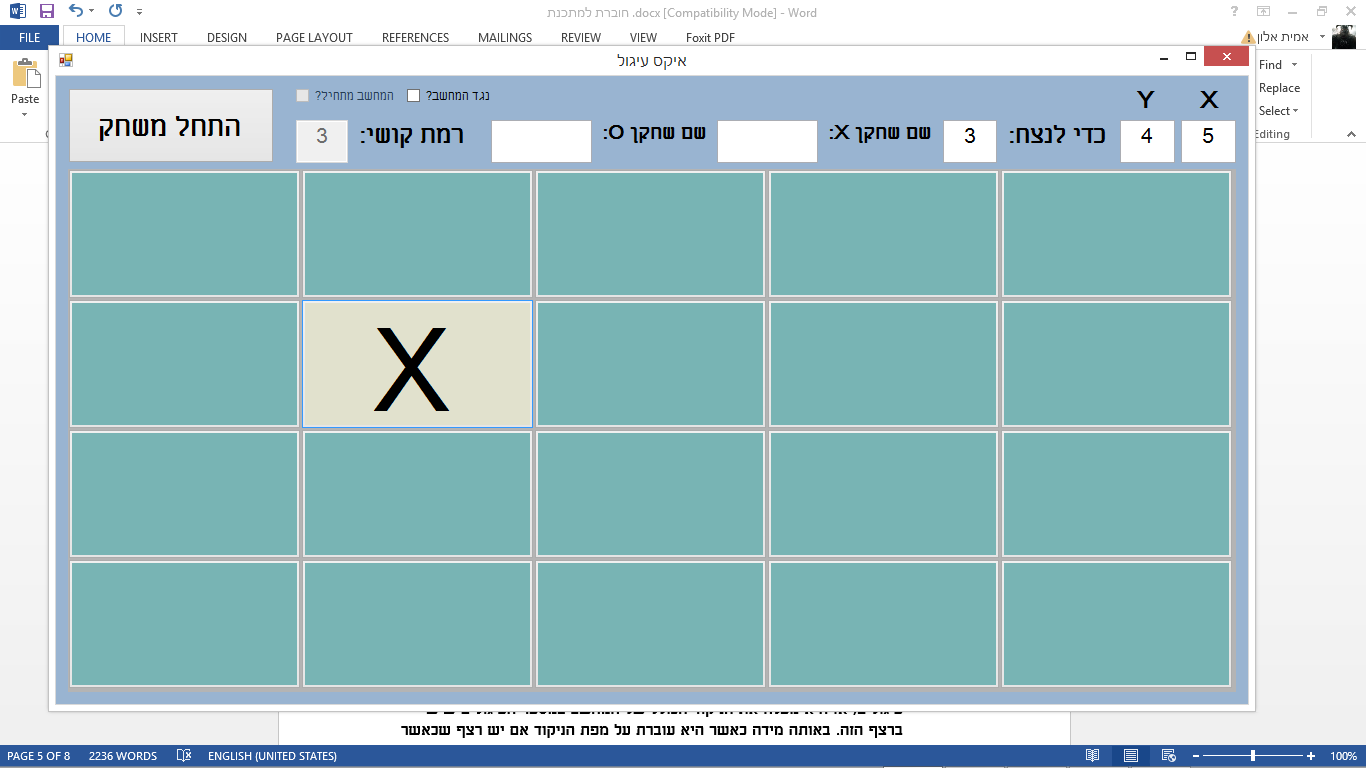
**מערך הנקודות של רצפים אפשריים בטור יהיה בגודל 5\*2 ויראה כך לאחר המהלך:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **1** | **0** |
| **0** | **0** | **0** | **1** | **0** |

**הסיבה לכך שהוספנו ניקוד אחד בשתי מקומות הוא כמו מקודם, בגלל שהמהלך ששיחקנו יכול ליצור שני רצפים אפשריים בטור שונים, והם:**

**מערך הניקוד של הרצפים האפשריים של אלכסונים ראשיים(אלכסונים שמגיעים משמאל למעלה לימין למטה) יהיה בגודל 3\*2 ויראה כך:**

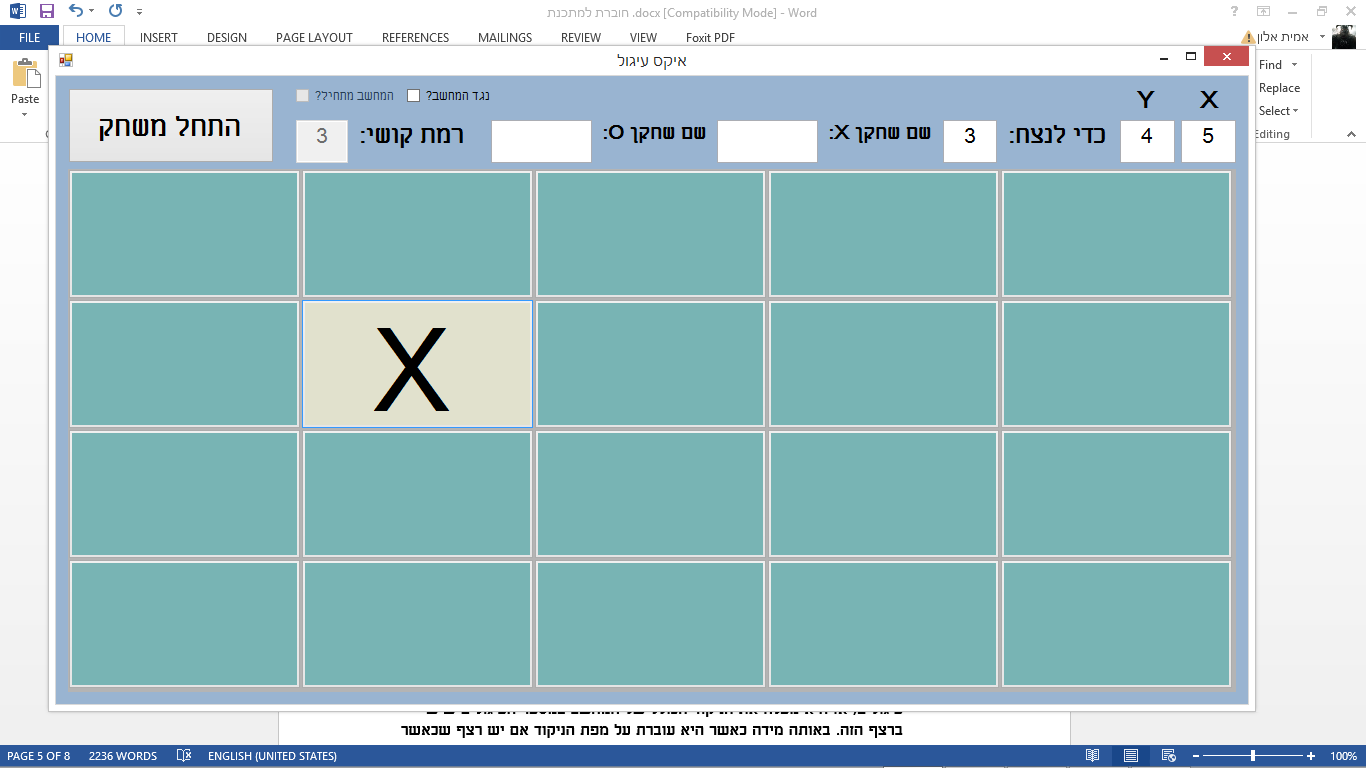
|  |  |  |
| --- | --- | --- |
| **0** | **0** | **1** |
| **0** | **1** | **0** |

**גם כאן הסיבה לכך שהוספנו 1 בשתי מקומות שונים היא בגלל שהמהלך ששיחקנו יכול ליצור שני רצפים אפשריים שונים של אלכסון ראשי, והם:**

ולבסוף מערך הניקוד של האלכוסנים המשניים יהיה גם בגודל 3\*2 ויראה כך:

|  |  |  |
| --- | --- | --- |
| **0** | **0** | **1** |
| **0** | **0** | **0** |

הסיבה לכך שהפעם הוספנו ניקוד אחד רק במקום אחד היא שמהלך זה יכול לעזור לבנות רק רצף של אלכסון משני, והוא:



1. חישוב ניקוד כולל של הלוח

בעזרת זה שיצרנו מפה של הניקוד של הלוח על פי רצפים, כעת ניתן לחשב מספר בודד אחד שייצג לנו גם את הניקוד של האיקסים וגם העיגולים – כלומר מספר אחד שהוא כל הניקוד ולא הרבה מערכים נפרדים.

בעצם הפונקציה ScoreSystem אשר אחראית על לתת את אותו ניקוד כללי, מחשבת את הניקוד הכולל של איקס(השחקן) ואת הניקוד הכולל של עיגול(המחשב), והמספר שהיא תחזיר יהיה הניקוד של העיגול פחות הניקוד של איקס(כלומר אם הניקוד חיובי סימן שמלמחשב יש ניקוד גבוה יותר כרגע, ואם הניקוד שלילי אז סימן שלשחקן יש יותר ניקוד. הניקוד הכולל של כל שחקן נקבע כך: הפונקציה עוברת על כל המפה של הניקוד של הרצפים(כלומר על כל המקומות בכל המערכים של הניקוד), כל פעם שהיא נתקלת ברצף שמתחלק ב1000 ללא שארית והתוצאה גדולה מ0, כלומר נתקלת ברצף שבו יש רק עיגולים, אז היא מעלה את הניקוד הכולל של המחשב במספר העיגולים שיש ברצף הזה. באותה מידה כאשר היא עוברת על מפת הניקוד אם יש רצף שכאשר מחלקים אותו ב1000 יש שארית והתוצאה היא 0, כלומר רצף שמכיל רק איקסים אז היא מעלה הניקוד הכולל של השחקן במספר האיקסים שיש ברצף זה.

1. בדיקת ניצחון

בזכות זה שיש לנו מפת ניקוד מוכנה כבר, הבדיקה של האם הכפתור שסמנו עכשיו הביא לניצחון או לא הופכת לקלה הרבה יותר. במקום שנצטרך לסרוק את מערך הכפתורים שיש לנו, כלומר את הלוח על מנת לזהות רצף של עיגולים או איקסים באורך המתאים, מספיק לנו להסתכל על מפת הניקוד, ולראות האם יש רצף כלשהו שמכיל בתוכו את כפתור זה, ולראות אם ערכו שווה לאורך שדרוש לניצחון במידה ואנו בודקים השחקן ניצח, או לראות אם ערכו חלקי 1000 שווה לאורך שדרוש לניצחון. לדוגמא אם הרצף לניצחון הוא 3, אז השחקן ניצח אם הערך של אחד מהרצפים הוא 3, ומחשב ניצח אם הערך של אחד הרצפים שווה ל3000.

תיאור כללי של כל הפונקציות הקיימות בקוד:

* private int ScoreSystem(int[,] MainAlachson, int[,] SecondAlachson, int[,] Shura, int[,] Amuda)

הפונקציה מקבלת את מפת הניקוד על פי רצפים הנוכחית של הלוח ומחזירה את הניקוד הכולל של המחשב פחות הניקוד של השחקן

קלט:

4 מערכים דו מימדיים, אחד מתאר את הניקוד של הרצפים שקשורים לניצחון בשורה, אחר מתאר את הניקוד של הרצפים שקשורים בטור, אחד את הרצפים שקשורים לאלכוסונים שמגיעים מימין למעלה לשמאל למטה(אלכסונים משניים) והאחרון את הרצפים שקשורים לאלכסונים שמגיעים משמאל למעלה לימין למטה(אלכסונים ראשיים).

פלט:

מספר שלם שמייצג את הניקוד הכולל של המחשב על פי המערכים שהפונקציה קיבלה כקלט, פחות הניקוד הכולל של השחקן.

* private int ComBestMove(Button[,] ButtonArr2, int StepsLeft2, bool ComTurn, int BiggestScore, int LowestScore)

פונקציה רקורסיבית, אחראית על דימוי המהלכים של המחשב ושל השחקן לסירוגין על ידי קריאה לעצמה עד שמגיעה למצב בו StepsLeft2 שווה ל0, כלומר שהיא לא אמורה לבדוק עוד צעדים קדימה, ואז היא קוראת ל ScoreSystem שתחזיר את הניקוד של הלוח במצב זה. כל פעם היא בודקת אם מדובר בתור מדומה של המחשב או של השחקן. במידה ומדובר בתור מדומה של השחקן היא מחזירה לפונקציה שקראה לה(לרוב לעצמה אך גם בסוף לFirstComBestMove) את הניקוד של המהלך עם הניקוד הקטן ביותר. במידה ומדובר בתור מדומה של המחשב היא תחזיר את הניקוד של המהלך עם הניקוד הכי גבוה.

קלט:

מערך כפתורים שבעצם מייצג את הלוח כולל כל נתוניו הישירים(טקסט וכו')

מספר שמתאר כמה צעדים נשאר לפונקציה לעשות, כל פעם יורד ב1

משתנה בוליאני שקובע אם תור המחשב או השחקן

מספר שמתאר את הניקוד של המהלך עם הניקוד הכי גבוה בינתיים בתור הקודם(כלומר בתור שקרא לתור מדומה זה)

מספר כמו הקודם רק שמתאר מה היה הניקוד של המהלך עם הניקוד הכי נמוך

פלט:

מספר שמתאר את הניקוד הכי נמוך ואו הכי גבוה שהיה בתור האחרון שנבדק על ידי פונקציה זו.

* private void FirstComBestMove(Button[,] ButtonArr, int StepsLeft)

פונקציה זו אחראית על התור המדומה הראשון שקורה לפני ComBestMove. פונקציה זו בעצם קוראת לפונקציה ComBestMove ובוחרת לשחק את המהלך שקיבל את הניקוד הגבוה ביותר. לאחר שהיא בוחרת במהלך עם הניקוד הגבוה ביותר היא משחקת אותו באמת על הלוח שנמצא בחלון התוכנית, ובודקת האם היה ניצחון.

קלט:

מערך כפתורים שמסמל את המצב הנוכחי של הלוח

מספר שמתאר כמה צעדים קדימה צריכה הפונקציה לחשוב

פלט:

אין

* private void ButtonArrayClickComp (object sender, EventArgs e)

הפונקציה שנקראת כאשר השחקן לוחץ על אחד מהכפתורים שעל הלוח כאשר המשחק הוא בין השחקן למחשב. אחראית על לסמן את המקום שהשחקן בחר במידה והוא בחר מקום פנוי ועל לבדוק האם מהלך זה הביא לניצחון ובמידה ולא לקרוא לפונקציה FirstComBestMove שתבצע את המהלך הבא של המחשב.

קלט:

EventArgs e – משתנה מסוג EventArgs שמכיל את הפרטים של האירוע. Object Sender – פרמטר שמכיל הפנייה לאובייקט שקרא לפונקציה זו.

פלט:

אין

* private void ButtonArrayClickPlayers (object sender, EventArgs e)

פונקציה זו נקראת כאשר השחקן לוחץ על אחד מהכפתורים שעל הלוח כאשר המשחק הוא בין שני שחקנים. היא אחראית על לסמן את הכפתור שעליו לחצנו, ולהחליט האם לסמן שמה איקס או עיגול על פי תור מי זה. בנוסף היא אחראית על להודיע לנו כאשר יש תיקו או ניצחון.

קלט:

EventArgs e – משתנה מסוג EventArgs שמכיל את הפרטים של האירוע. Object Sender – פרמטר שמכיל הפנייה לאובייקט שקרא לפונקציה זו.

פלט:

אין

* private string Win(int Shura, int Amuda, bool Player)

פונקציה זו אחראית על בדיקה האם יש ניצחון כרגע לדמות שנבחרה כקלט(משתמש\שחקן), כתוצאה ממהלך מסוים שדמות זו ביצעה. ומחזירה משתנה האם מהלך זה הביא לניצחון, תיקו או לכלום.

קלט:

מספר שמעיד באיזה שורה התרחש המהלך

מספר שמעיד באיזה טור התרחש המהלך

משתנה בוליאני שקובע באם המהלך בוצע על ידי השחקן או המחשב

פלט:

String שמתאר את התוצאות של מהלך זה:

"Win" - מהלך זה הביא לניצחון להדמות שעשתה אותו

"Draw" - מהלך זה הביא לתיקו

"Lose" – לא באמת הפסד, מהלך זה לא גרם לכלום

* private void btn\_StartGame\_Click(object sender, EventArgs e)

הפונקציה נקראת כאשר השחקן לחץ על הכפתור להתחלת המשחק. היא אחראית על נתינת ערכים למשתנים החשובים בתוכנית אשר השחקן מזין, על בדיקה שערכים אלו תקניים ועל יצירת לוח המשחק.

קלט:

EventArgs e – משתנה מסוג EventArgs שמכיל את הפרטים של האירוע. Object Sender – פרמטר שמכיל הפנייה לאובייקט שקרא לפונקציה זו.

פלט:

אין

* private void disableBoard()

פונקציה זו נקראת על ידי פונקציות אחרות במצב של ניצחון או תיקו, והיא אחראית על מחיקת טקסט זמני מהכפתורים במידה והיה עליהם טקסט זמני, ועל נטרול הלוח כדי שאי יהיה אפשר להמשיך ללחוץ על הכפתורים שהוא מכיל עד שיתחיל משחק חדש.  
קלט:  
 אין  
פלט:

אין

* private void XO\_Load(object sender, EventArgs e)

הפונקציה הזו נקראת כאשר עולה התוכנית. היא אחראית שעם עליית התוכנית סימון ברירת המחדל יהיה שהמשחק הוא נגד המחשב.

קלט:

EventArgs e – משתנה מסוג EventArgs שמכיל את הפרטים של האירוע. Object Sender – פרמטר שמכיל הפנייה לאובייקט שקרא לפונקציה זו.

פלט:

אין

* private void chkbox\_vsComp\_CheckedChanged(object sender, EventArgs e)

פונקציה זו אחראית על לאפשר או לנטרל אפשרות לכתוב בתיבה שרשום לידה "שם שחקן O:", על לנטרל או לאפשר לסמן את התיבה של האם המחשב מתחיל, ולנטרל או לאפשר לבחור כמה צעדים קדימה המחשב יחשוב. היא בוחרת לעשות זאת לפי האם מדובר במשחק של שני שחקנים או נגד המחשב - כלומר על פי הסימון של התיבה שקשורה לנגד מי המשחק.

קלט:

EventArgs e – משתנה מסוג EventArgs שמכיל את הפרטים של האירוע. Object Sender – פרמטר שמכיל הפנייה לאובייקט שקרא לפונקציה זו.

פלט:

אין