

# דמקה - Checkers

סטודנט: ריאד מחאג'נה

מנחה : תמר צמח



# תוכן עניינים

1.תקציר.....	3
2. חוקי המשחק.....	4-5
תרשים השתלשלות תור של שחקן.....	6
3. UI ממשק משתמש.....	7
דף שחקן נגד שחקן(PVP).....	8
דף שחקן נגד מחשב(PVC).....	9
דף משחק ברשת.....	10-12
כמה תמונות מהמשחק.....	13-14
4. מחלקות.....	15-20
מחלקת Ntree.....	21
מחלקת MINIMAX.....	22-25
5.רעיונות לשיפורים.....	26

# תקציר

דמקה הוא משחק לוח שמשחקים בדרך כלל על לוח בן 64 (8 על 8) משבצות או על לוח בן 100 (10 על 10) משבצות ("דמקה בינלאומית"). המשחק מיועד לשני שחקנים שלכל אחד מהם יש 12 אבני משחק בגרסת 64 המשבצות, או 20 אבני משחק בגרסת 100 המשבצות. לכל שחקן צבע משלו.

המשחק נפוץ מאוד והוא דורש תחכום, שימוש רב בטקטיקה וידע תאורטי. הדמקה עוזרת לפתח מחשבה לוגית מופשטת – הודות לפשטות הכללים ילדים יכולים להגיע לבניית שרשראות לוגיות ארוכות למדי. המשחק מהווה כלי חינוכי יעיל ותורם ליכולות הניתוח והחשיבה של השחקנים.

מטרת המשחק היא להוריד מהלוח ("לאכול") את כל האבנים של היריב, או לחסום אותן, כלומר לא לאפשר ליריב לקיים מסע (צעד/מהלך). שחקן שנשאר ללא אבנים או ללא מסע אפשרי מוכרז כמפסיד.

הפרויקט הכתב בשפת C# בשימוש ב-Windows Presentation Foundation (WPF)

WPF מאפשרת יצירת תוכנות בעלות ממשק משתמש מרהיב, תוך שימוש בהאצת חומרה, זכות שהייתה שהיה שמורה עד כה למשחקים. בין השימושים השגרתיים באפליקציות WPF ניתן למצוא: חלונות שמתאימים את תוכנם לגודל השטח המוקצה להם, אנימציות חלקות, ושימוש נרחב בוידאו ואודיו.

WPF הושפעה רבות מטכנולוגיות UI כגון HTML ו Flash וכוללת שפה חדשה בשם XAML ליצירת ממשק משתמש בצורה הצהרתית (דקלרטיבית).

# תקציר

באפליקציה שבנית ייש שני סוגים של משחק:

1. שחקן נגד שחקן (multiplayer):

יש שתי אופציות לשחק נגד שחקן, אופציה אחת היא לשחק ללא חיבור לרשת (offline), המשחק מאפשר לשחקנים לשחק גם באבנים שחורות וגם באבנים לבנות באותו חלון.

האופציה השנייה היא לשחק נגד שחקן דרך חיבור לרשת, השחקן הראשון יהיה הhost והשני יהיה הclient, החיבור בין שני השחקנים הוא מסוג TCP דרך localhost.

הhost יהיה השחקן עם האבנים הלבנות והclient יהיה השחקן עם האבנים השחורות, אם אחד השחקנים סוגר את החלון של האפליקציה זה אומר שהוא נכנע והיריב יהיה המנצח.

2. שחקן נגד מחשב:

המחשב יחליט איזה תנועה לבצע לפי אלגוריתם מינימקס, למחשב יש 3 רמות קושה easy, medium, וhard כך שזה נקבע על ידי העומק של אלגוריתם מינימקס.

המחשב משחק עד סוף המשחק, הוא לא נכנע ולא יסכים לסיים את המשחק בתיקו.

# חוקי המשחק

## תנועה :

- אבן יכולה לנוע צעד אחד קדימה בצורת אלכסון בתנאי שהמשבצת פנויה(ריקה).
- מלך יכול לנוע לכל הכיוונים באלכסון (כלומר גם אחורה), ללא הגבלה על מספר המשבצות בדרך.
- כשאבן משחק מגיעה לשורה האחרונה, היא הופכת להיות מלך.

## אכילה:

- אבן יכולה לבצע אכילה כאשר אבן משחק מונחת במשבצת סמוכה לאבן היריב, ומעבר לאבן היריב יש מקום פנוי. חובה לבצע אותו , אם בתום הדילוג אפשרי דילוג נוסף חובה לבצע גם אותו.
- בדומה לאבנים הרגילות, גם המלך מחויב לבצע דילוגים כאשר יש לו אפשרות לעשות זו, המלך יכול לבצע דילוג גם כאשר אבנו של היריב אינה צמודה אליו באותו האלכסון. לאחר הדילוג המלך יעצור במרובע הראשון, המלך יכול לבצע דילוגים בכל הכיוונים באלכסון.

# חוקי המשחק

## ■ סיום המשחק

■ ניצחון:

- שחקן היריב לא נותרו כלל כלים על הלוח
- השחקן היריב נכנע
- לשחקן היריב אין אפשרות לבצע מהלך מאחר שכליו חסומים

■ תיקו:

- אם העמדה חוזרת על עצמה במשך שלוש פעמים.
- כאשר לכל שחקן יש רק אבן אחת
- בכל שלב במשחק רשאי כל אחד מהצדדים להציע תיקו, והצד השני רשאי להסכים או לסרב

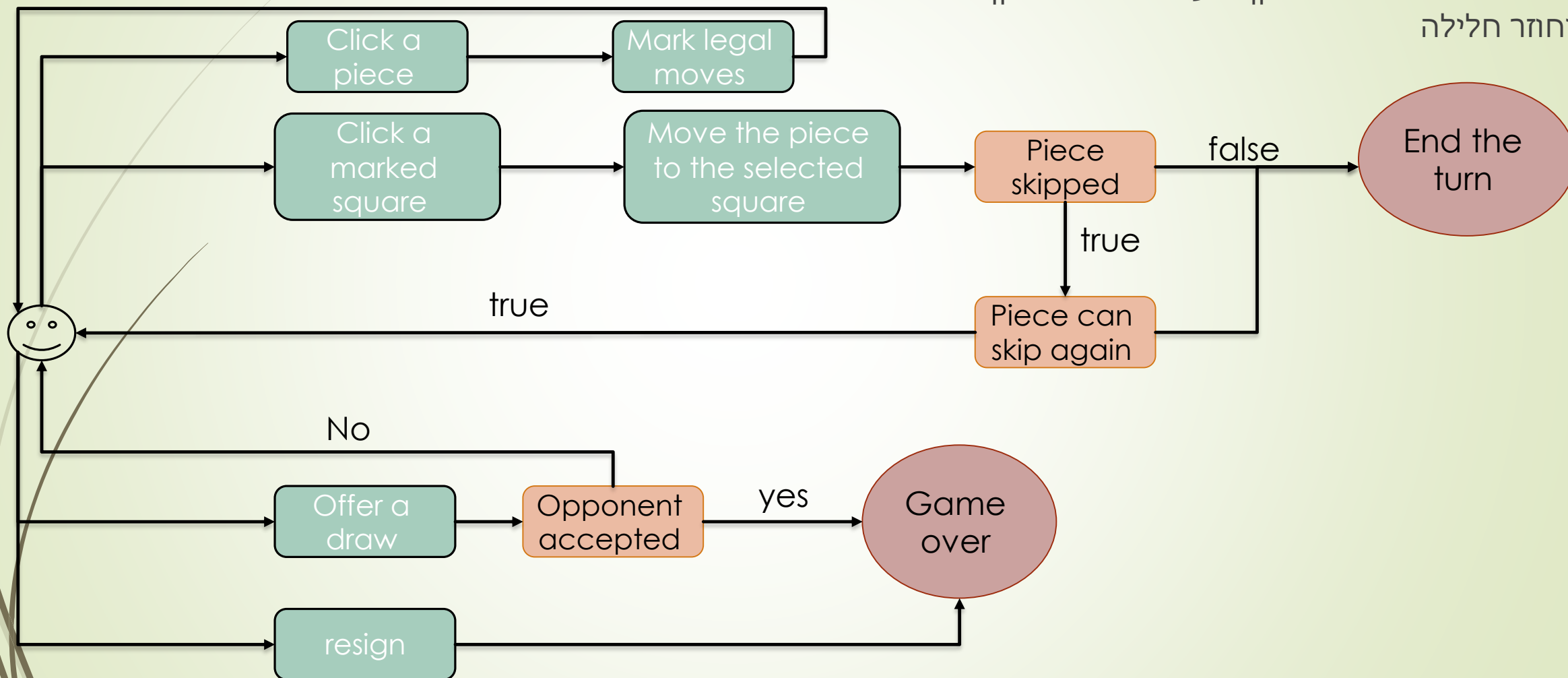
■ לוח המשחק 8X8

■ כל שחקן מתחיל עם 12 אבנים

■ השחקן עם האבנים הלבנים מתחיל ראשון

# תרשים השתלשלות השחקן

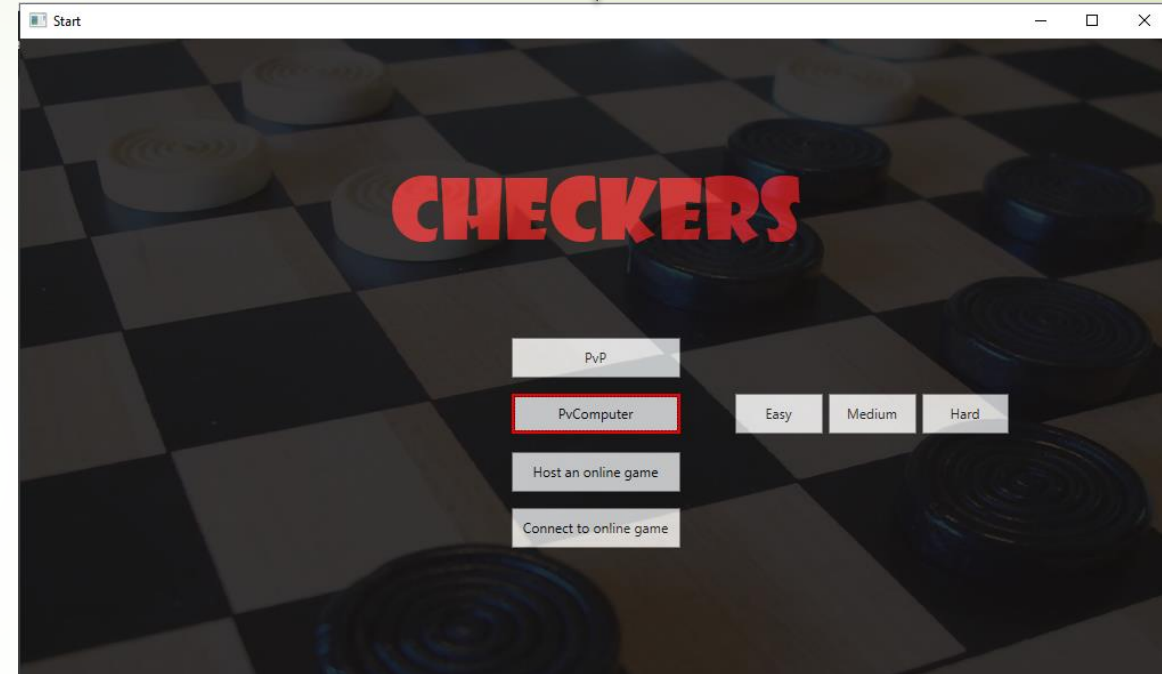
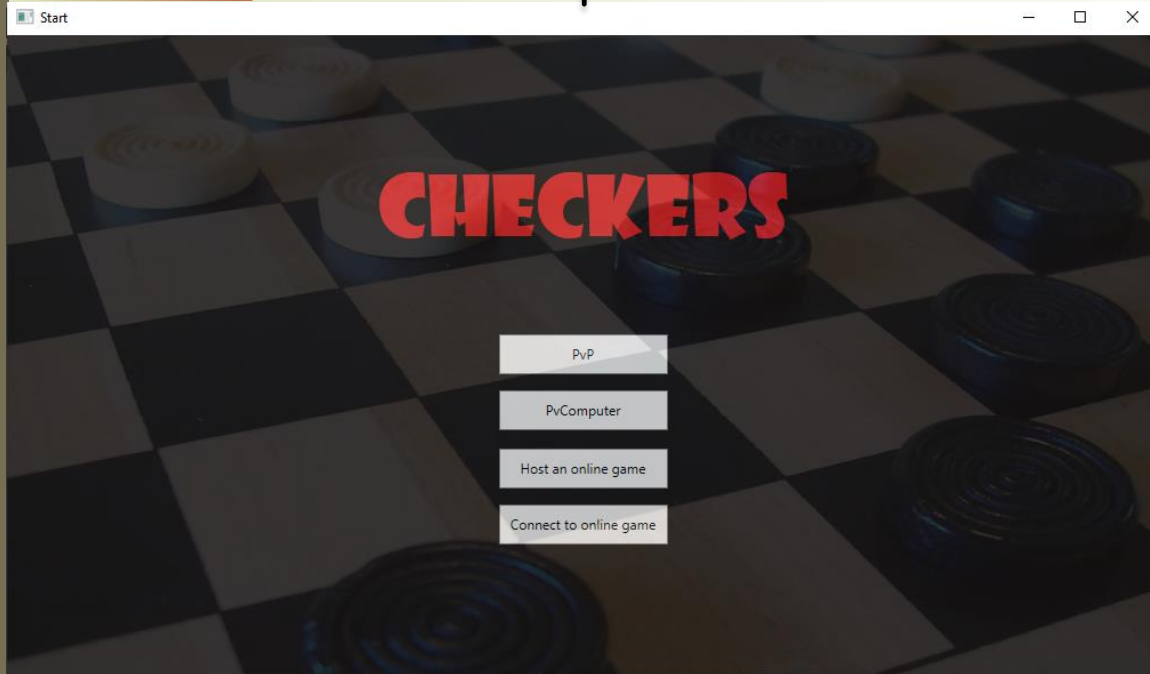
תרשים 1 - השתלשלות תור של שחקן במשחק: התור מתחיל בצד שמאל של התרשים כאשר השחקן בוחר מה תהיה הפעולה. מתוך החלטה ראשונית זו בעצם מתקדם כל התור לפי הנראה בתרשים הנ"ל עד להשלמת תורו של השחקן ומעבר התור לשחקן הבא וחוזר חלילה





# UI ממשק משתמש

אחרי לחיצה על כפתור PvComputer



Pvp: שחקו נגד שחקן כש שני השחקנים משחקים על אותו מחשב

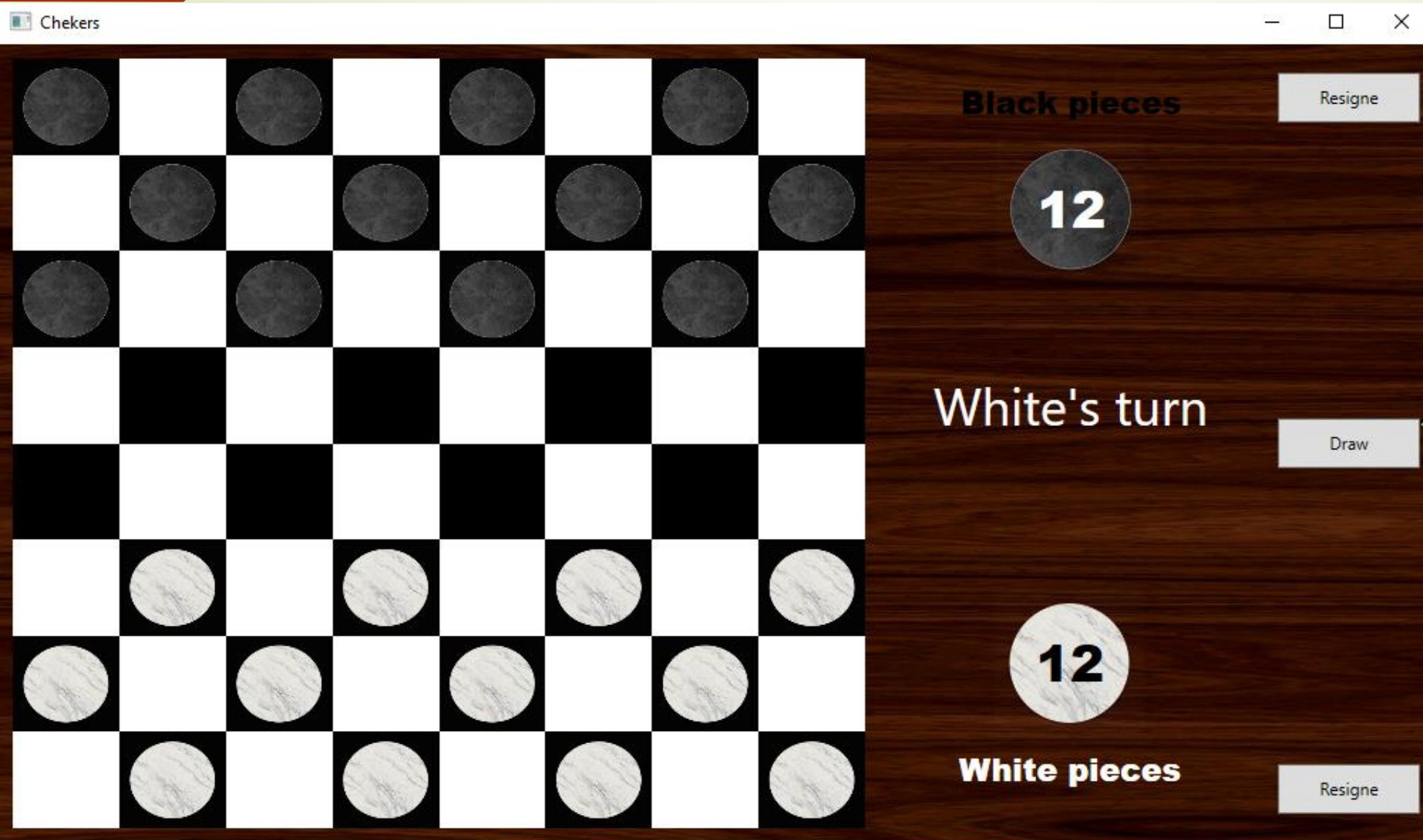
PvComputer: שחקן נגד מחשב המשחק לפי אלגוריתם Minimax כך שהקושי של המשחק נקבע על ידי עומק החישוב של האלגוריתם:

מצב קל מפעיל את האלגוריתם על עומק 2, מצב בינוני על עומק 3 ומצב קשה על עומק 4.

Host an/connect to online game: לאריח/להתחבר ל- משחק של שחקן נגד שחקן על ידי חיבור לרשת כך ששני השחקנים מתקשרים אחד עם השני דרך ה localhost



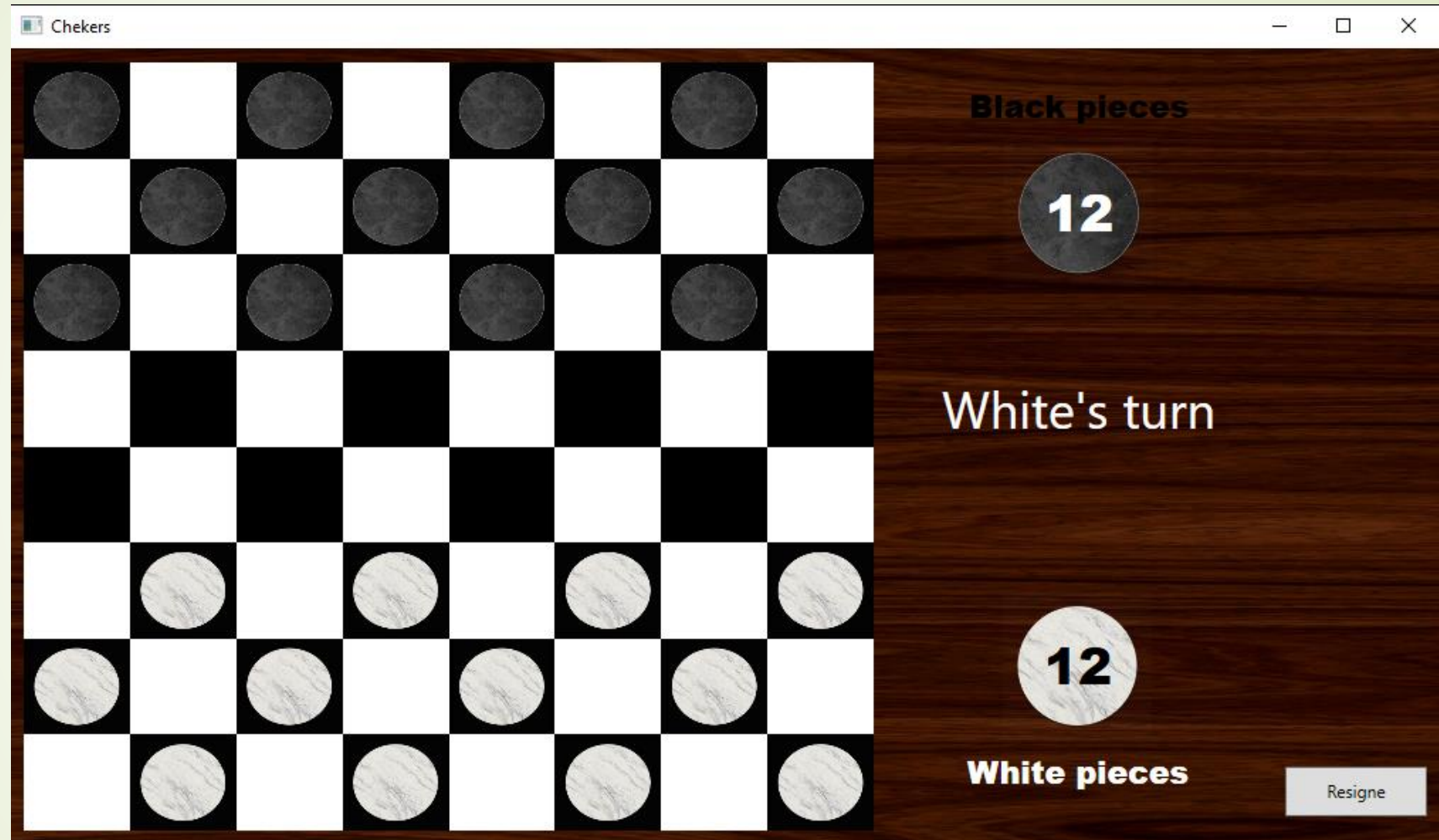
# Pvp page



this button  
appears  
only in pvp  
mode

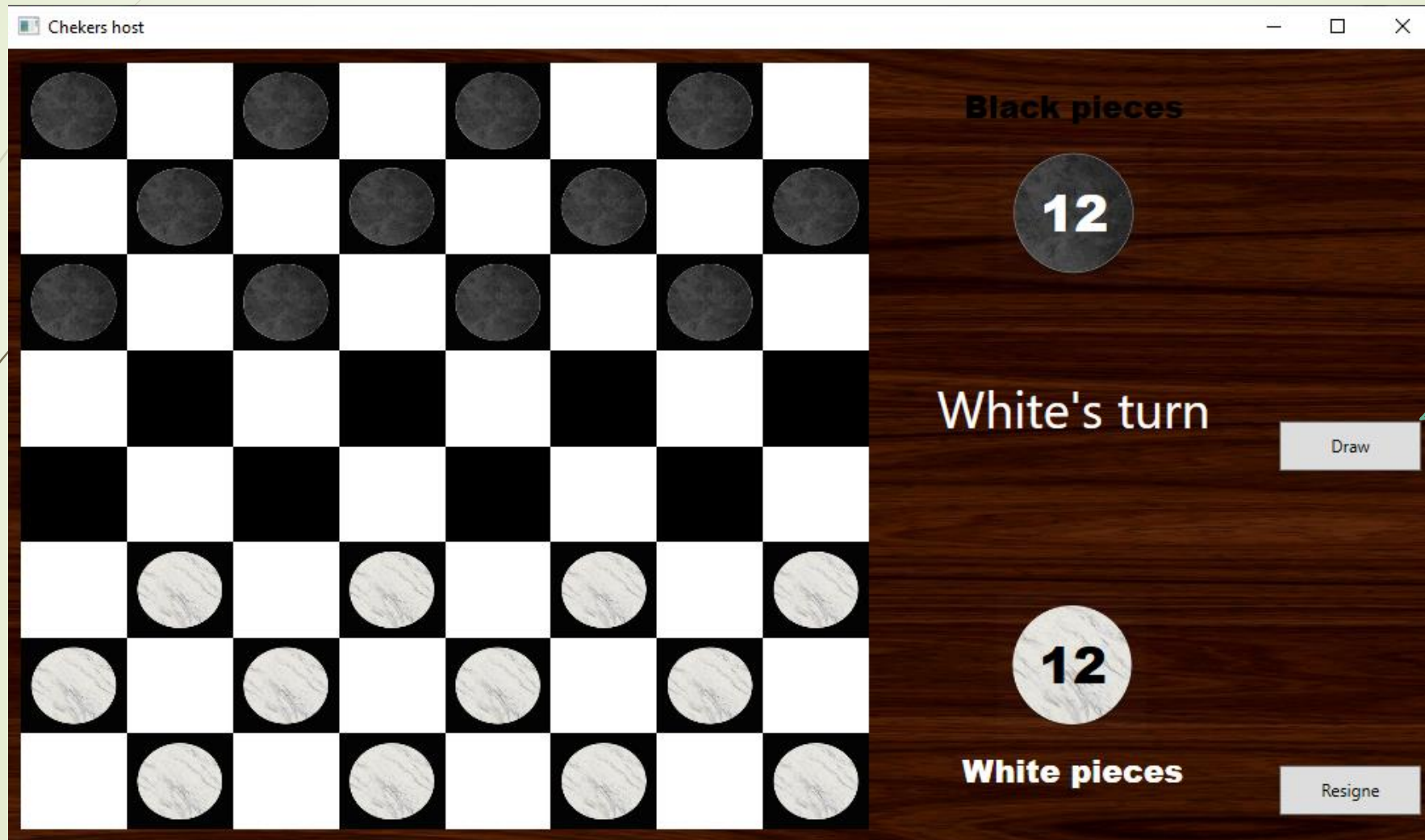
In pvp mode  
draw button  
make a draw  
on the spot

# PvComputer page



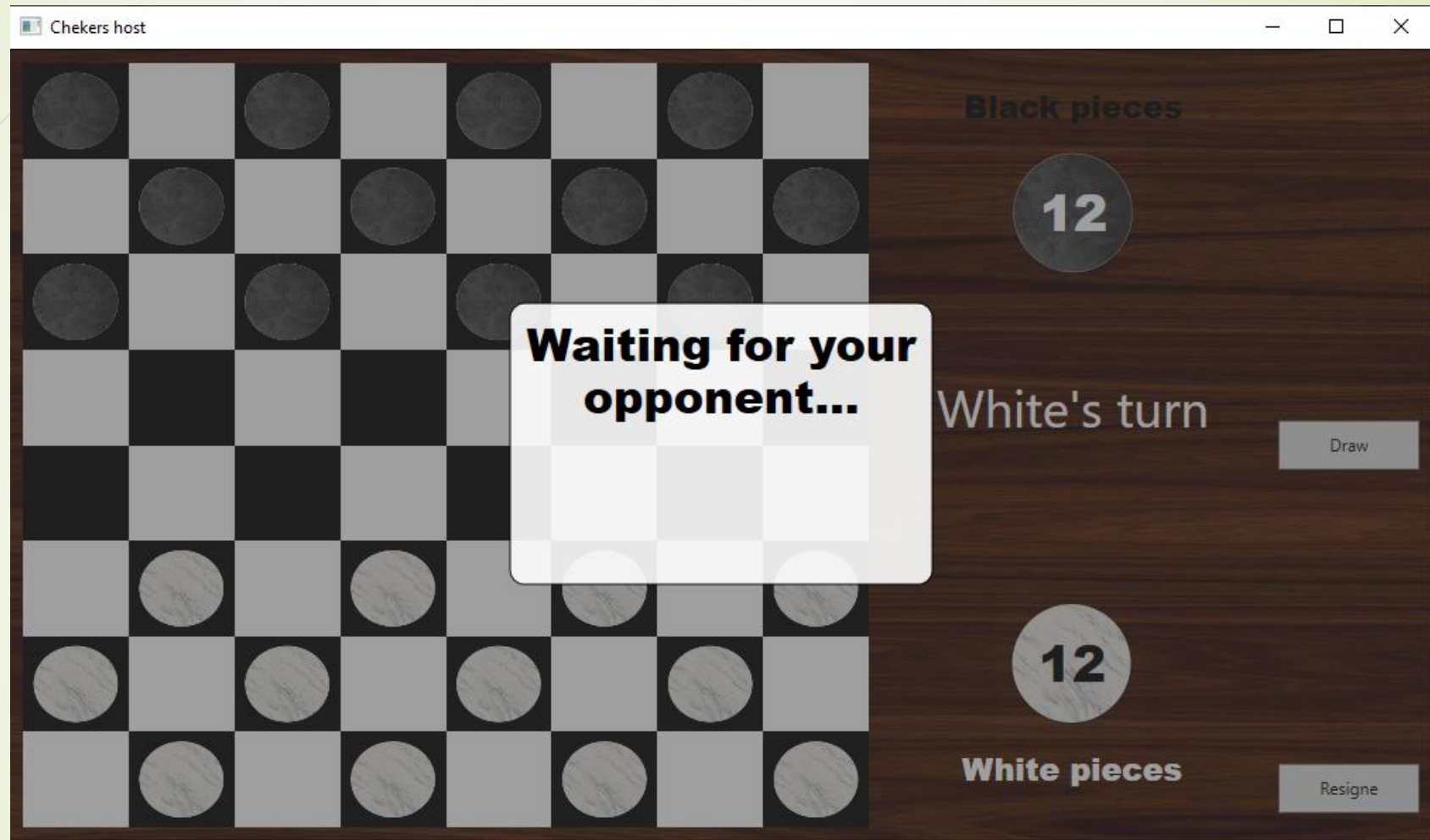
בPvComputer mode אין כפתור לתיקו כי המחשב תמיד דוחה תיקו, המחשב ממשיך לשחק עד הסוף.

# Network game page



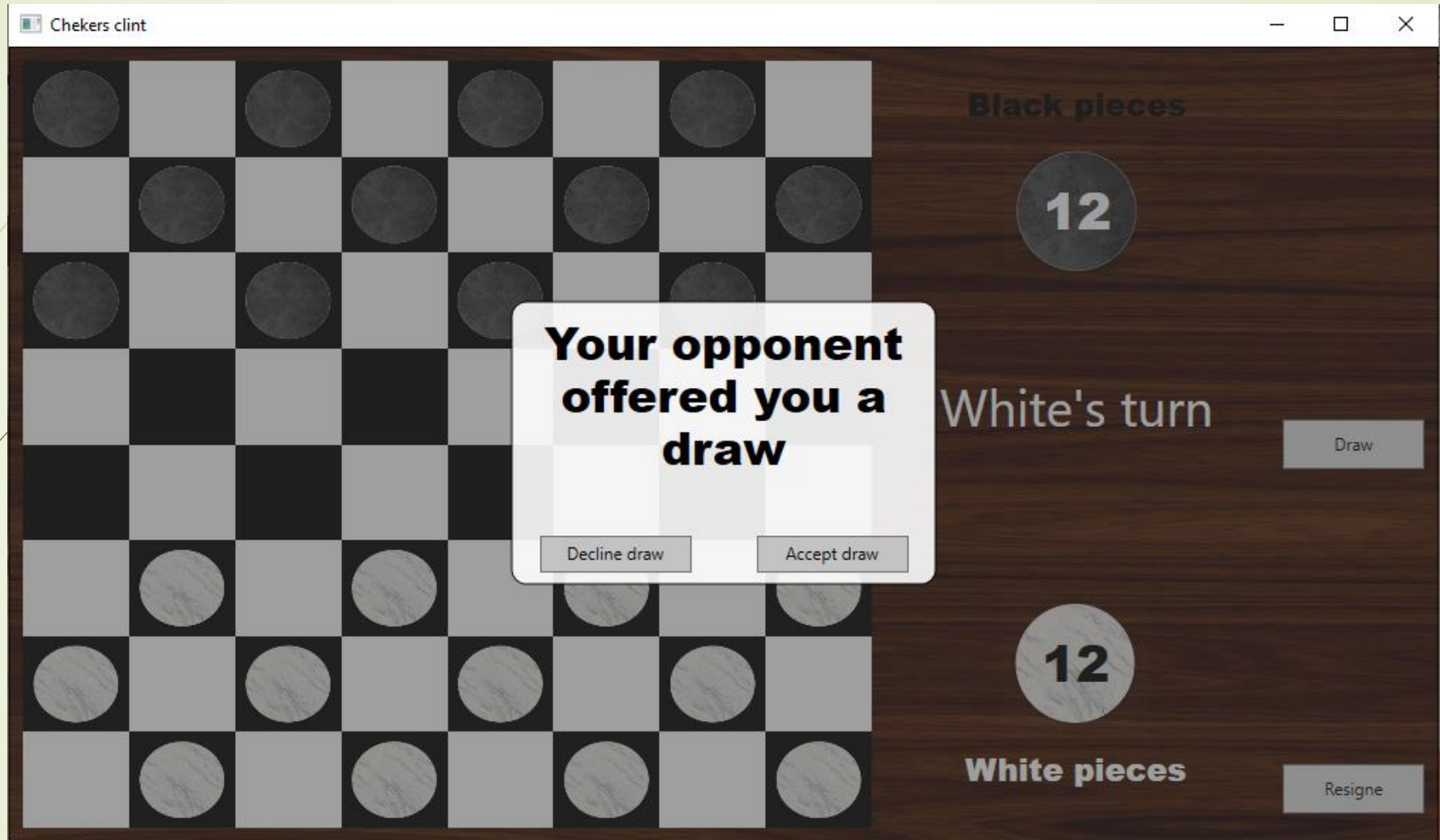


# Network game page

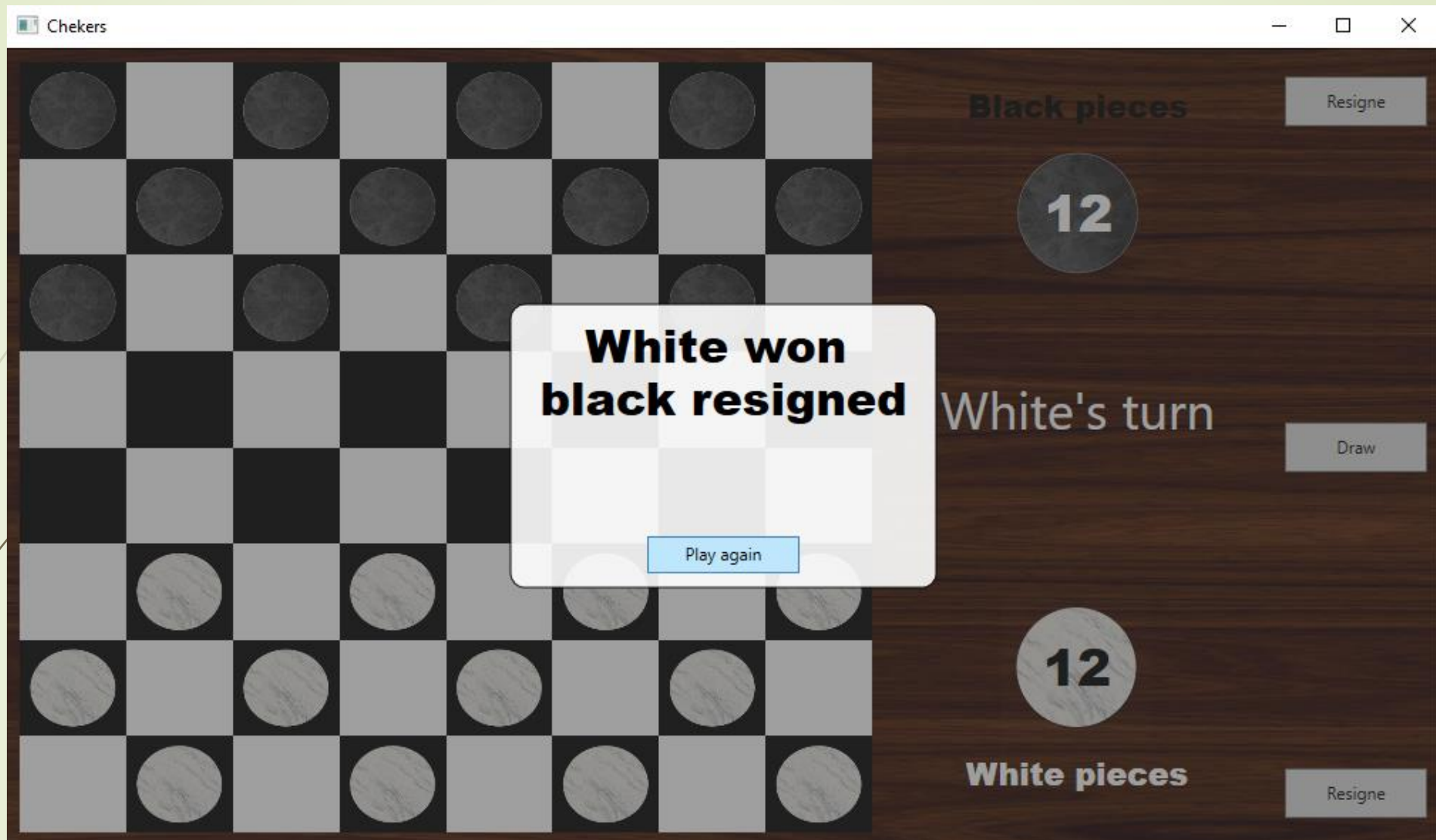


מסך של שחקן אחרי שהוא מציע תיקו

# Network game page

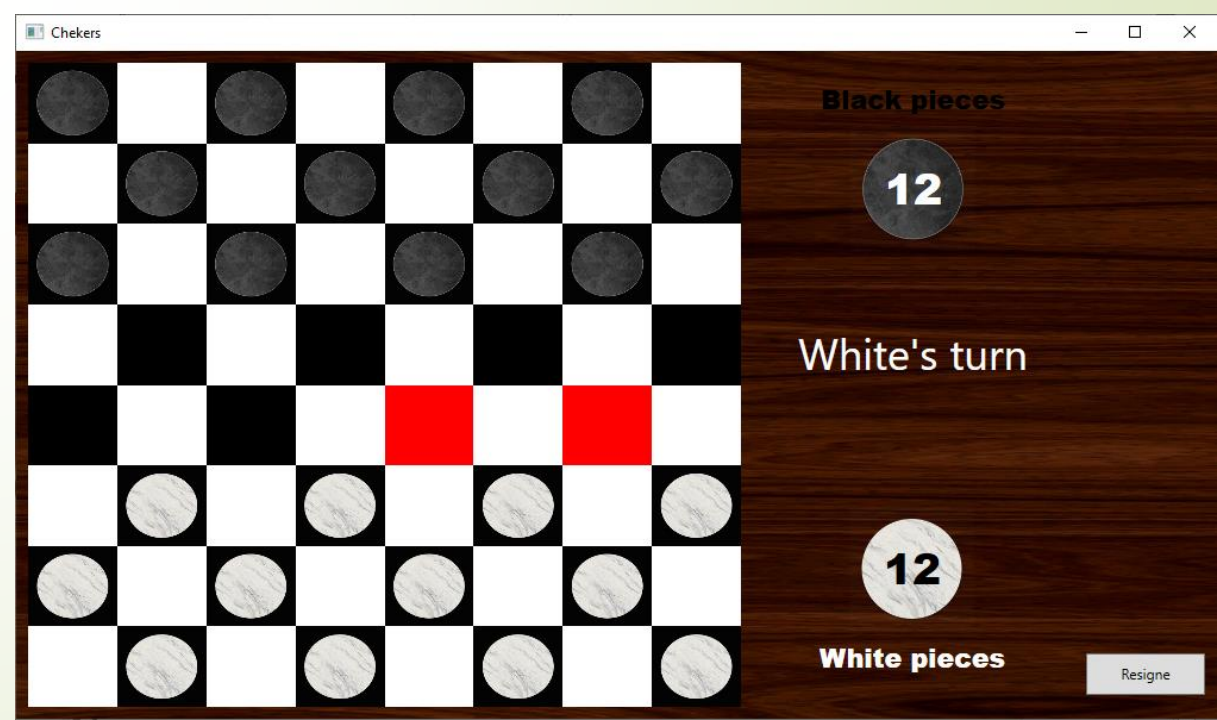
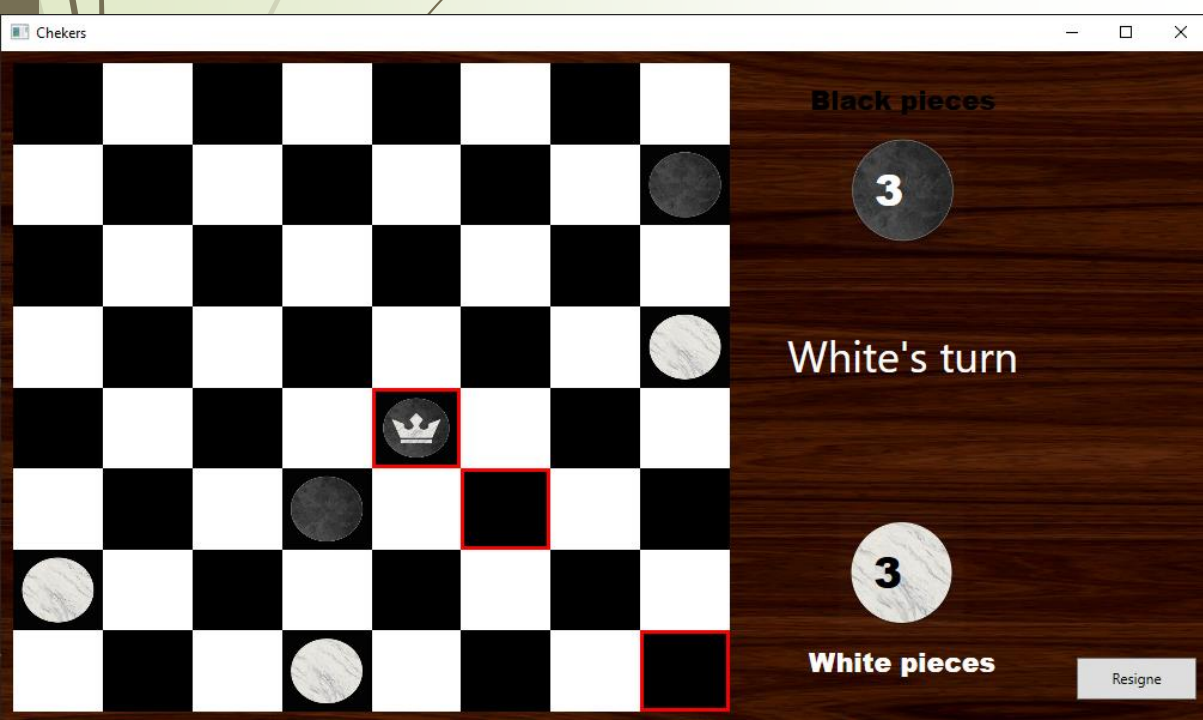
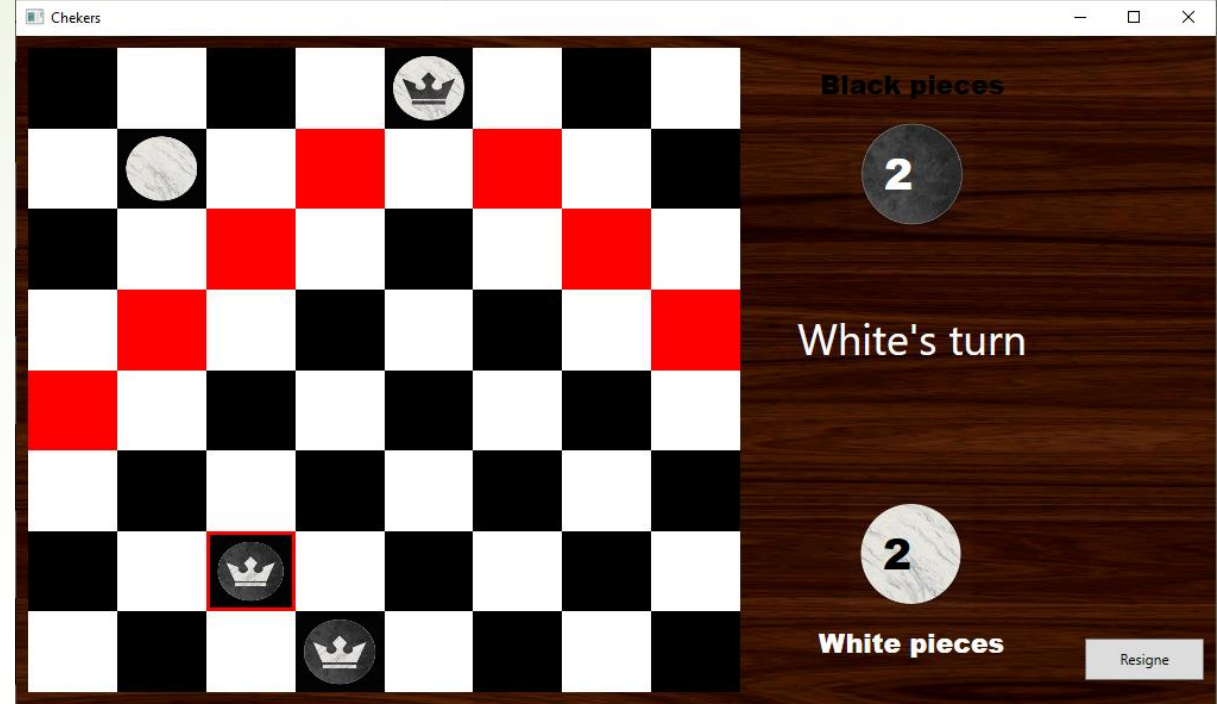
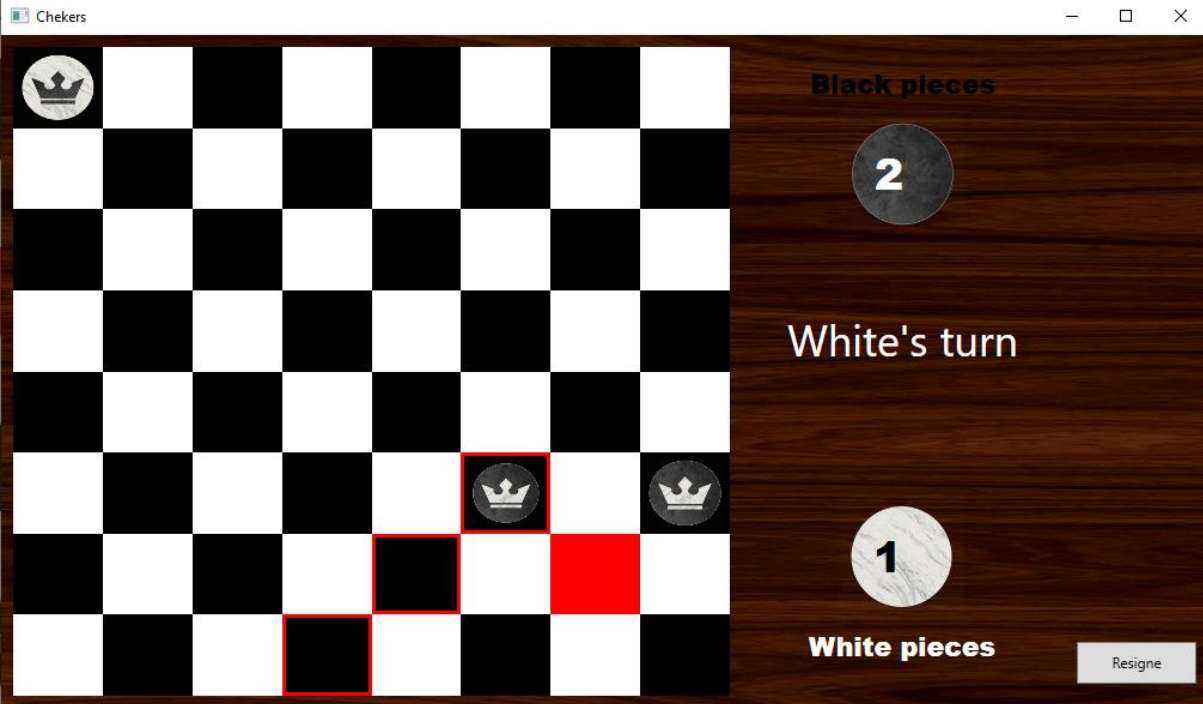


מסך של שחקן אחרי שהיריב הציע תיקו



כך נראה מסך אחרי שאחד השחקנים נכנע





# מחלקות

## שדות כללים

variables	Variable purpose
Board	מיצג את הלוח של המשחק כך שכל איבר במטריצה מיצג משבצת בלוח.
White_turn	משתנה בוליאני אם הערך שלו הוא TRUE אז זה התור של השחקן עם האבנים הלבנים אחרת אז זה התור של השחקן עם האבנים השחורים.
P_lock	משתנה בוליאני אם הערך שלו הוא TRUE אז השחקן דלג ויכול לדלג עוד פעם.
Ate	משתנה בוליאני אם הערך שלו הוא TRUE אז השחקן דלג .
Bpice	מיצג מספר אבנים שחורים במשחק
Wpice	מיצג מספר אבנים לבנים במשחק
prevb	מיצג האבן האחרון שהשחקן לחץ עלו
playedMoves	מערך של מחרוזות המייצגות את ההיסטוריה של המשחק

## שדות מיוחדות בPvComputer

variables	Variable purpose
MiniMax m	משתנה שמיצג את Computer

## שדות מיוחדות במשחק דרך הרשת

variables	Variable purpose
Socket s	השקע ששני השחקנים מחוברים/מדברים דרכו
BackgroundWorker mr	משתנה שמקבל הודעה מהשחקן היריב ומבצע פעולות בהתאם לתוכן ההודעה
TcpListener server	מיצג חיבור מסוג TCP מצד הHost
TcpClient clint	מיצג חיבור מסוג TCP מצד הClient
ishost	משתנה בוליאני שמספק מידע אם השחקן הוא ה Client או ה Host

## פונקציות ואלגוריתמיקה

FUNCTION	DESCRIPTION
BoardToString	מחזירה מחרוזת המתארת את לוח המשחק
PlayerCanSkip : bool	תחזיר TRUE אם השחקן שצורו עכשיו יכול לדלג
piceCanSkip : bool	תחזיר TRUE אם אבן מסוים יכול לדלג
CanMove : bool	תחזיר TRUE אם אבן מסוים יכול לנוע
markIgl	מצביעה כל המשבצות שהאבן יכול לנוע אליהן
unmarkall	הופכת כל משבצת אדומה לשחורה
PlayerCanMove : bool	תחזיר TRUE אם יש אבו לשחקן שתורו עכשיו יכול לנוע
Repetition: bool	בודקת אם יש חזרה המשחק חזר לאותו מצב 3 פעמים, אם כן אז המשחק יסתיים בתיקו
BuildBoard	מעדכן את המשחק לפי המחרוזת שהוא מקבל


## פונקציות מיוחדות במשחק ברשת

FUNCTION	DESCRIPTION
SocketConnected : bool	בודקת אם החיבור עדיין קיים
Web_Closing	סוגרת את החיבור בין שהשחקנים ביציאה מהאפליקציה
StrToByte : byte[]	הופכת מחרוזת למערך של ביטים
ByteToStr : string	הופכת מערך של ביטים למחרוזת
Mr_DoWork	פונקציה המגדירה לBackgroundWorker מה לעשות כשהוא מופעל
Acc_Click	הסכם בין שני השחקנים שהמשחק יסתיים בתיקו
Dec_Click	דחיית הצעה לתיקו

# תרגום משחק למחרוזת

```
public static string BoardToString(ref Button[,] p)
{
    string s = "";
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (p[i, j].Background == white_k)
            {
                s += "W";
            }
            else if (p[i, j].Background == white_p)
            {
                s += "w";
            }
            else if (p[i, j].Background == black_p)
            {
                s += "b";
            }
            else if (p[i, j].Background == black_k)
            {
                s += "B";
            }
            else if (p[i, j].Background == Brushes.White)
            {
                s += "e";
            }
            else if (p[i, j].Background == Brushes.Black || p[i, j].Background == Brushes.Red)
            {
                s += "E";
            }
        }
    }
    return s;
}
```

w  White piece



W  White king

b  Black piece

B  Black king

e  Empty white square

E  Empty black square

r  Empty red square (minimax)  
 Empty black square






# מחלקת Funcs

מחלקה שמכילה את המתודות הבאות:

, unmarkall, marklgl, CanMove, piceCanSkip, PlayerCanSkip, BoardToString  
PlayerCanMove ו-repetition.

הפונקציות האלה מאוד שמושיות ומשתמשים בהם בכל המחלקות והמטרה מהמחלקה הזו הוא להימנע משכפול קוד.





# מחלקת NTree

המחלקה הזו היא מימוש למבנה נתונים של עץ, כך לכל קדקוד יש שדה Data הוא מחרוזת המייצגת מצב כלשהוא במשחק, ושדה Value שהוא הערכה למצב השמור ב Data .  
לכל קדקוד יש רשימה מ'ושרת המייצגת הבנים של הקדקוד.

לקדקודים בעץ אין הגבלה למספר הבנים.

המטרה מהמחלקה הזו היא לעזור לי לממש אלגוריתם Minimax כך שכל מצב הוא מצב אפשרי למשחק וכל הבנים שלו הם המצבים שאפשר להגיע להם מהאב .

# אלגוריתם MiniMax

- משחקים רבים בנויים על העיקרון של סט חוקים קבוע ומוגדר היטב ומשחק לסירוגין. לדוגמה: שחמט ודמקה. בתורת המשחקים, עץ מינימקס הוא עץ הפורס את האפשרויות למשחק של שחקן א', את התגובות של שחקן ב' לכל פעולה של שחקן א', את תגובותיו של א' לתגובותיו של ב' וכך הלאה. מעשית מוגבל עומקו של העץ על ידי הזמן וזיכרון המחשב העומדים לרשותנו.
- העלים בעץ שנוצר הם מצבים סטטיים שנגיע אליהם לאחר רצף של מהלכים. ניתן ציון לכל מצב סטטי שכזה, שישקף כמה המצב טוב מבחינתנו.
- סרוק את העץ החל בעלים, דרך הקדקודים שמעליהם עד השורש הציון שניתן לכל קדקוד הוא הערך הגבוה ביותר של העלים/הקדקודים שתחתיו, אם אותו קדקוד מסמל מהלך שלי, והערך הנמוך ביותר של העלים/קדקודים שתחתיו אם אותו קדקוד מסמל מהלך של היריב (כי ברור שהוא יבחר באפשרות הטובה ביותר בשבילו - הכי גרועה בשבילי).
- כשנגיע לשורש, נבחר בקדקוד שמתחת לשורש עם הציון הטוב ביותר.

# מחלקת MiniMax

המחלקה הזו היא מימוש לאלגוריתם MiniMax שדרכו המחשב מחליט איזה תנועה לבצע.

אופן הפעולה של האלגוריתם הוא כך:

האלגוריתם מקבל כקלט מצב של משחק בצורת מחרוזת ועובר על כל אבני המשחק ובודק אם יש לאבן תנועה חוקית אם כן אז אני מבצע את התנועה ומוסיף את המצב החדש כבן למצב הנוכחי ומפעיל את האלגוריתם על המצב החדש שקבלתי בצורה רקורסיבית.

אחרי שהמעבר על כל האבנים מסתיים אני עובר על הבנים של הקדקוד הנוכחי ומחזיר את הבן בעל הערך המקסימלי או המינימלי בהתאם לערך של המשתנה  $\max$ .

בסוף האלגוריתם יחזיר את data של הבן בעל ערך מקסימלי בשורש העץ, אם יש יותר מאחד אז האלגוריתם יבחר בצורה אקראית.

תנאי העצירה הוא אם האלגוריתם הגיע לעומק המקסימלי או שאין תנועות חוקיות במצב, ואז נחשב אץ ההערכה של המצב.

ההערכה של המצב היא מספר הנקודות של אבנים השחורים מינוס מספר הנקודות של האבנים הלבנים כך שכל אבן רגיל הוא נקודה אחת וכל מלך זה 4 נקודות.

```

public string GetNextMove(string b)
{
    white_turn = false;
    t = new NTree(b);
    BuildBoard(b);
    TreeBulid(t, b, p, 0, true);
    int ii = 1;
    int maxval = -100;
    string maxstr = b;
    NTree maxtree = t.GetChild(ii);
    var cil = t.GetChild(ii);
    while (cil != null)
    {
        if (cil.value > maxval)
        {
            maxval = cil.value;
            maxstr = cil.data;
            maxtree = cil;
        }

        cil = t.GetChild(++ii);
    }

    List<string> possipleMoves = new List<string>();
    ii = 1;
    cil = t.GetChild(ii);
    while (cil != null)
    {
        if (cil.value == maxval)
        {
            possipleMoves.Add(cil.data);
        }

        cil = t.GetChild(++ii);
    }
    Random r = new Random();
    int rInt = r.Next(0, possipleMoves.Count);

    return possipleMoves[rInt];
}

```

בונים עץ מינימקס

בודקים הערך  
המקסימלי במצבים  
האפשריים

בוחרים בצורה  
אקראית אחד  
המצבים שיש להם  
ערך מקסימלי

## מקטע קוד ממתודת TreeBulid תנאי עצירה של הפונקציה

```
if(!b.Contains("b") && !b.Contains("B"))
{
    root.value = -12;
    return;
}

if (!b.Contains("w") && !b.Contains("W"))
{
    root.value = 12;
    return;
}

if (cuurd == maxDepth)
{
    int wp = 0, bp = 0;
    foreach (char c in root.data)
    {
        switch (c)
        {
            case 'w':
                wp++;
                break;
            case 'W':
                wp += 4;
                break;
            case 'b':
                bp++;
                break;
            case 'B':
                bp += 4;
                break;
        }
    }

    root.value = bp - wp;
    return;
}
```

המשחק  
הסתיים

האלגוריתם  
הגיע לעומק  
המקסימלי



# רעיונות לשיפורים

## שיפור ל AI של המחשב:

- להדגיר למחשב כמה פתיחות במקום להתחיל לחשב תנועות במינימקס מהתחלת המשחק.
- אלגוריתם מינימקס רץ בזמן הגיוני רק אם העומק שלו הוא 4 או פחות, בעזרת Alpha-beta pruning
- לאלגוריתם תהיה אפשרות לרוץ בזמן הוגן בעומק יותר גדול, ואז יהיה יותר קשה לנצח משחק נגד מחשב.

## שיפור למשחק ברשת:

- במקום שהתוכנית תחבר את שני השחקנים לLOCALHOST אפשר לעשות שרת שיהיה מסוגל לחיבורים בין השחקנים, מה שמאפשר ליותר ממשחק אחד להתקיים באותו זמן.