Wydział Matematyczno-Przyrodniczy
Uniwersytet Kardynała Stefana Wyszyńskiego

Gra komputerowa
„Wilk i owce”

Michał Studziński

Warszawa 2016

1. Wstęp

Założeniem projektu była wizualizacja gry planszowej „Wilk i owce”[1] na komputerze. W tym calu należało wykorzystać wybrany przez siebie język programowania oraz zaproponować własny algorytm dla sztucznej inteligencji. Gra „Wilk i owce” jest tradycyjną grą o typie warcabowym.

Do projektu został użyty język C#[3][4]. Głównym powodem wyboru tego języka programowania były wbudowane biblioteki, które można wykorzystać w programie w celu użycia techniki obrazu 2D (podstawowa biblioteka Drawing2D), aby pokazać aktualny stanu gry. Przy użyciu „Window Form” wraz z biblioteką graficzną „Drawing2D” został uzyskany program, który wyglądem jest zbliżony do gry komputerowej. Wersja z przyciskami i odpowiednimi okienkami w znacznym stopniu ułatwiają nawigację oraz wygląd programu.

Program został napisany w programie Visual Studio 2013 64 bit[2] na platformie sytemu Windows 10 64 bit.

2. Opis modelu

2.1 Zasady Gry

Jeden z graczy dysponuje czterema pionami białymi (owcami), drugi dysponuje jednym pionem szarym (wilkiem). Wszystkie piony są obsadzone na ciemnych polach szachownicy (ciemny brąz), po których będą się poruszać. Pozycja początkowa wygląda następująco: owce obsadzają pierwszy rząd szachownicy, wilk staje na jednym dowolnym wybranym polu krańcowego rzędu szachownicy.

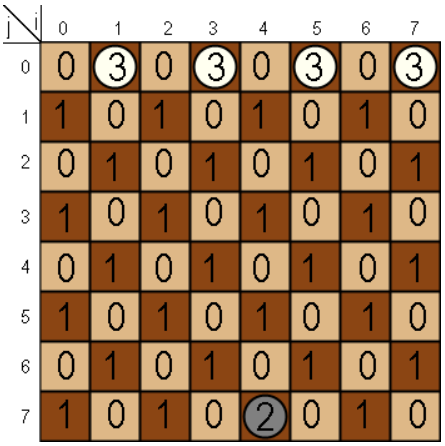
Celem gracza-wilka jest przedostanie się na pierwszy rząd pól szachownicy. Gracz dysponujący owcami wygrywa, gdy uda mu się zablokować wilka tak, żeby nie mógł on wykonać posunięcia.

Grę rozpoczyna gracz grający wilkiem. W jednym ruchu może przejść o jedno pole po przekątnej w dowolną stronę. Piony gracza grającego owcami w kolejnych posunięciach przechodzą także o jedno pole po przekątnej, lecz tylko do przodu. Wilk może cofać się, a owce nie. Ruchy wykonywane są naprzemiennie, każdy gracz musi wykonać jeden ruch gdy będzie jego kolej.

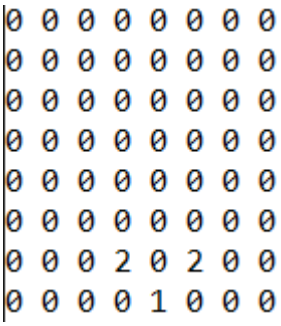
Gra wilk i owce jest znana w różnych krajach pod innymi nazwami (np. jako wilk i psy).

2.2 Algorytmy programu

2.2.1 Algorytm zarządzania ruchami wilka



tablica plansza



tablica ruch

Lista kroków:

1. Przeszukaj całą tablicę „plansza” i znajdź pozycję wilka. Szukaj położenie wilka (wartość 2 dla tablicy „plansza”) dopóki „i < 8” i „j < 8” . Zwiększaj „i = i + 1”, jeśli „i > 7” to zwiększ „j = j + 1”, a „i = 0”. Jeżeli wilk został znaleziony to przejdź do kroku 2.
2. Przeszukaj całą tablicę „plansza” i znajdź pozycję owcy (wartość 3 dla tablicy „plansza”) leżącej najdalej wilka względem zmiennej „j” (j.Owcy < j.Wilka, jeżeli jest więcej niż jedna owca na wysokości „j”, to jest wybierana ta, która ma najmniejszą zmienną „i”, w ten sposób będziemy definiować pojęcie: ostatniego pionka owcy) . Szukaj położenie ostatniej owcy dopóki „i < 8” i „j < 8” . Zwiększ „i + 1”, jeśli „i > 7” to zwiększ „j + 1”, a „i = 0”. Jeżeli owca została znaleziona to przejdź do kroku 3.
3. Wywołaj metodę „kliknięcie ” dla wartości [i,j] z kroku 1. Otrzymujemy współrzędne kolejnego ruchu dla wilka w tablicy „ruch”, przejdź do kroku 4.
4. Przeszukaj całą tablicę w poszukiwaniu dostępnego ruchu (wartość 1 dla tablicy „plansza” i wartość 2 dla tablicy „ruch”) dopóki „i < 8” i „j < 8” . Zwiększaj „i = i + 1”, jeśli „i > 7” to zwiększ „j = j + 1”, a „i = 0”. Jeżeli wolne pole zostało znalezione to przejdź do kroku 5, gdy „i > 7” i „j > 7” przejdź do kroku 6.
5. Sprawdź kolejno warianty:
   a) jeśli wygrana wilka („j” nowego ruchu wilka jest równe z „j” ostatniej owcy) to Score = 100, dodaj ruch do listy X, przejdź do kroku 4.
   b) jeśli wygrana owcy (wilk nie ma, żadnej możliwości ruchu na tablicy „ruch”) to Score = -100, dodaj ruch do listy X, przejdź do kroku 4.
   c) jeśli wartość „i” wilka przed wykonaniem ruchu dla $i \in (1;7)$ oraz wartość „j” przed wykonaniem ruchu jest większa od wartości „j” po wykonaniu ruchu to Score = 25, dodaj ruch do listy X, przejdź do kroku 4.
   d) jeśli wartość „i” wilka przed wykonaniem ruchu jest równa 0 lub 7 oraz wartość „j” przed wykonaniem ruchu jest większa od wartości „j” po wykonaniu ruchu to Score = 15, dodaj ruch do listy X, przejdź do kroku 4.
   e) jeśli wartość „i” wilka przed wykonaniem ruchu dla $i \in (1;7)$ oraz wartość „j” przed wykonaniem ruchu jest mniejsza od wartości „j” po wykonaniu ruchu to Score = 10, dodaj ruch do listy X, przejdź do kroku 4.

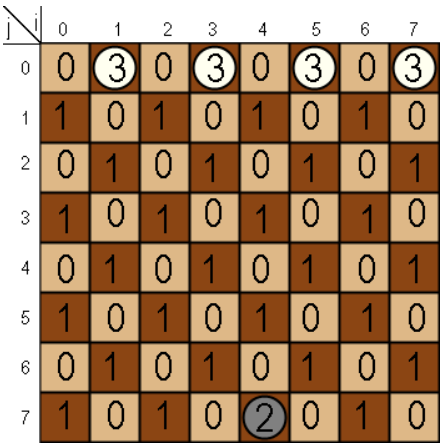
f) jeśli wartość „i” wilka przed wykonaniem ruchu jest równa 0 lub 7 oraz wartość „j” przed wykonaniem ruchu jest mniejsza od wartości „j” po wykonaniu ruchu to Score = 5, dodaj ruch do listy X, przejdź do kroku 4.
g) jeśli wartość „i” wilka przed wykonaniem ruchu jest równa 0 oraz wartość „j” przed wykonaniem ruchu jest równa 7 to Score=5, dodaj ruch do listy X, przejdź do kroku 4.

6. Przesortuj listę X od najmniejszej do największej według wartości ruchów (według wartości Score). Przejdź do kroku 7.
7. Jeżeli liczba elementów listy X > 0 to przejdź do kroku 8. W innym przypadku przejdź do kroku 13.
8. Jeżeli wartość ruchu zapisanego w liście X (elementy wybierane na podstawie wartości Score) jest większa od 15 to przypisz ten ruch z listy X do listy A, w innym przypadku przypisz do listy B. Przejdź do kroku 9.
9. Przesortuj listę A i B. Przejdź do kroku 10.
10. Jeżeli liczba elementów listy A > 0 to wylosuj jeden ruch z listy A i przypisz do zmiennej temp. Przejdź do krok 12. W innym przypadku przejdź do kroku 11
11. Wylosuj jeden ruch z listy B i przypisz do zmiennej temp. Przejdź do kroku 12.
12. Przypisz położenie temp pionka i usuń starą pozycję pionka.
13. Koniec algorytmu.

X – lista ruchów wilka (zawiera struktury „para”, które to zawierają współrzędne [i,j] przed wykonaniem ruchu, współrzędne [i,j] po wykonaniu ruchu i wartość wagi ruchu według której będą sortowane ruchy, im większa wartość tym lepszy ruch wybranego pionka)
A - lista lepszych ruchów wilka (tworzona na podstawie listy ruchów wilka)
B – lista gorszych ruchów wilka (tworzona na podstawie listy ruchów wilka)
temp – pomocnicza do której przypisujemy najlepszy ruch z listy ruchów
para – strukutra zawieraja współrzędne [i,j] przed wykonaniem ruchu, współrzędne [i,j] po wykonaniu ruchu i wartość wagi ruchu według której będą sortowane ruchy, im większa wartość tym lepszy ruch wybranego pionka
i,j – liczniki pętli (wartość początkowa to 0), i - odpowiada ze szerokość, j – odpowiada za wysokość
Score – wartość ruchu (zawiera się w strukturze „para”)

(każdy podpunkt korzysta z innych liczników [i,j] jeśli są zdefiniowane na początku podpunktu, w przypadku gdy nie ma definicji liczników [i,j] jak przykładowo w podpunkcie 5. to używane są liczniki z poprzedniego podpunktu, w tym przypadku z 4. punktu)

2.2.2 Algorytm zarządzania ruchami wszystkich owiec




```
0 1 0 0 0 0 0 0
2 0 2 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

Lista kroków:

1. Przeszukaj całą tablicę „plansza” i znajdź pozycję wilka. Szukaj położenie wilka (wartość 2 dla tablicy „plansza”) dopóki „i < 8” i „j < 8” . Zwiększaj „i = i + 1”, jeśli „i > 7” to zwiększ „j = j + 1”, a „i = 0”. Jeżeli wilk został znaleziony to przejdź do kroku 2.
2. Przeszukuj tablicę „plansza” do mementu napotkania owcy (wartość 3 dla tablicy „plansza”) dopóki „i < 8” i „j < 8” . Zwiększaj „i = i + 1”, jeśli „i > 7” to zwiększ „j = j + 1”, a „i = 0”. Jeżeli owca została znaleziona to przejdź do kroku 3, gdy „i > 7” i „j > 7” przejdź do kroku 6.
3. Wywołaj metodę „kliknięcie ” dla wartości [i,j] z kroku 2. Otrzymujemy współrzędne kolejnego ruchu dla owcy w tablicy „ruch”, przejdź do kroku 4.
4. Przeszukaj całą tablicę w poszukiwaniu dostępnego ruchu (wartość 1 dla tablicy „plansza” i wartość 2 dla tablicy „ruch”) dopóki „i < 8” i „j < 8” . Zwiększaj „i = i + 1”, jeśli „i > 7” to zwiększ „j = j + 1”, a „i = 0”. Jeżeli wolne pole zostało znalezione to przejdź do kroku 5, gdy nie ma dostępnego ruchu dla owcy „i > 7” i „j > 7” to przejdź do kroku 2.
5. Sprawdź kolejno warianty:

a) jeśli w następnym dowolnym ruchu wygrana owiec (wilk w tablicy „ruchy” nie będzie miał dostępnych ruchów) to Score = 100, dodaj ruch do listy X, kontynuuj krok 2

b) jeśli w następnym dowolnym ruchu wygrana wilka (różnica wartości „j” ostatniej owcy i wartości „j” wilka jest równa -1) to Score = -100, dodaj ruch do listy X, kontynuuj kroku 2

c) jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca i zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = 75, dodaj ruch do listy X, przejdź do kroku 2

d) jeśli różnica wartości „j” Wilka i wartości „j” jest mniejsza od 3 i wartość bezwzględna wartości „i” Wilka i wartości „i” jest mniejsza od 2 ” to Score = 15, jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

e) jeśli w rzędzie „j” znajdującym się najbliżej względem „j” wilka stoi tylko jedna owca na jednym z krańców pola (krańce pola są to kolumny o „j = 0 lub j =1”), to kolejny ruch będzie

ruchem pionka, który może przesunąć się na pozycje „j = j + 2” to Score = 50, dodaj ruch do listy X, przejdź do kroku 2

f) jeśli w rzędzie „j” znajdującym się najbliżej względem „j” wilka stoi tylko jedna owca na jednym z krańców pola (krańce pola są to kolumny o „j = 6 lub j =7”), to kolejny ruch będzie ruchem pionka, który może przesunąć się na pozycje „j = j - 2” to Score = 50, dodaj ruch do listy X, przejdź do kroku 2

g) jeśli różnica wartości „j” Wilka i wartości „j” jest mniejsza lub równa 2 i wartość bezwzględna wartości „i” Wilka i wartości „i” jest mniejsza od 2 ” to Score = 10, jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

h) jeśli różnica wartości „j” Wilka i wartości „j” jest mniejsza od 2 i wartość bezwzględna wartości „i” Wilka i wartości „i” jest mniejsza od 2 ” to Score = 8, jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

i) jeśli różnica wartości „j” Wilka i wartości „j” jest mniejsza od 3 i wartość bezwzględna wartości „i” Wilka i wartości „i” jest mniejsza od 3 ” to Score = 6, jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

j) jeśli różnica wartości „j” Wilka i wartości „j” jest mniejsza od 4 i wartość bezwzględna wartości „i” Wilka i wartości „i” jest mniejsza od 4 ” to Score = 4, jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

k) jeśli różnica wartości „j” Wilka i wartości „j” jest mniejsza od 5 i wartość bezwzględna wartości „i” Wilka i wartości „i” jest mniejsza od 5 ” to Score = 3, jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

l) jeśli różnica wartości „j” Wilka i wartości „j” jest mniejsza od 6 i wartość bezwzględna wartości „i” Wilka i wartości „i” jest mniejsza od 6 ” to Score = 2, jeśli jest sąsiednia owca po

prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

m) jeśli różnica wartości „j” Wilka i wartości „j” jest mniejsza od 7 i wartość bezwzględna wartości „i” Wilka i wartości „i” jest mniejsza od 7 ” to Score = 1, jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

n) w przeciwnym wypadku Score = 0, jeśli jest sąsiednia owca po prawej lub lewej stronie na tej samej wysokości „j” co dana owca to Score = Score + 5, jeśli jest zblokowany wilk (wilk stoi po skosie przed owcą, czyli współrzędna „i” wilka różni się od współrzędnej „i” owcy następująco „i+1 lub „i-1”) to Score = Score + 20, jeśli owca jest najdalej położoną owcą względem zmiennej „j” od wilka i w danym wierszu „j” nie ma innej owcy to Score = Score – 5, jeśli różnica wartości „j” wilka i ruchu owcy jest mniejsza od 0 to Score = Score - 70, dodaj ruch do listy X, przejdź do kroku 2

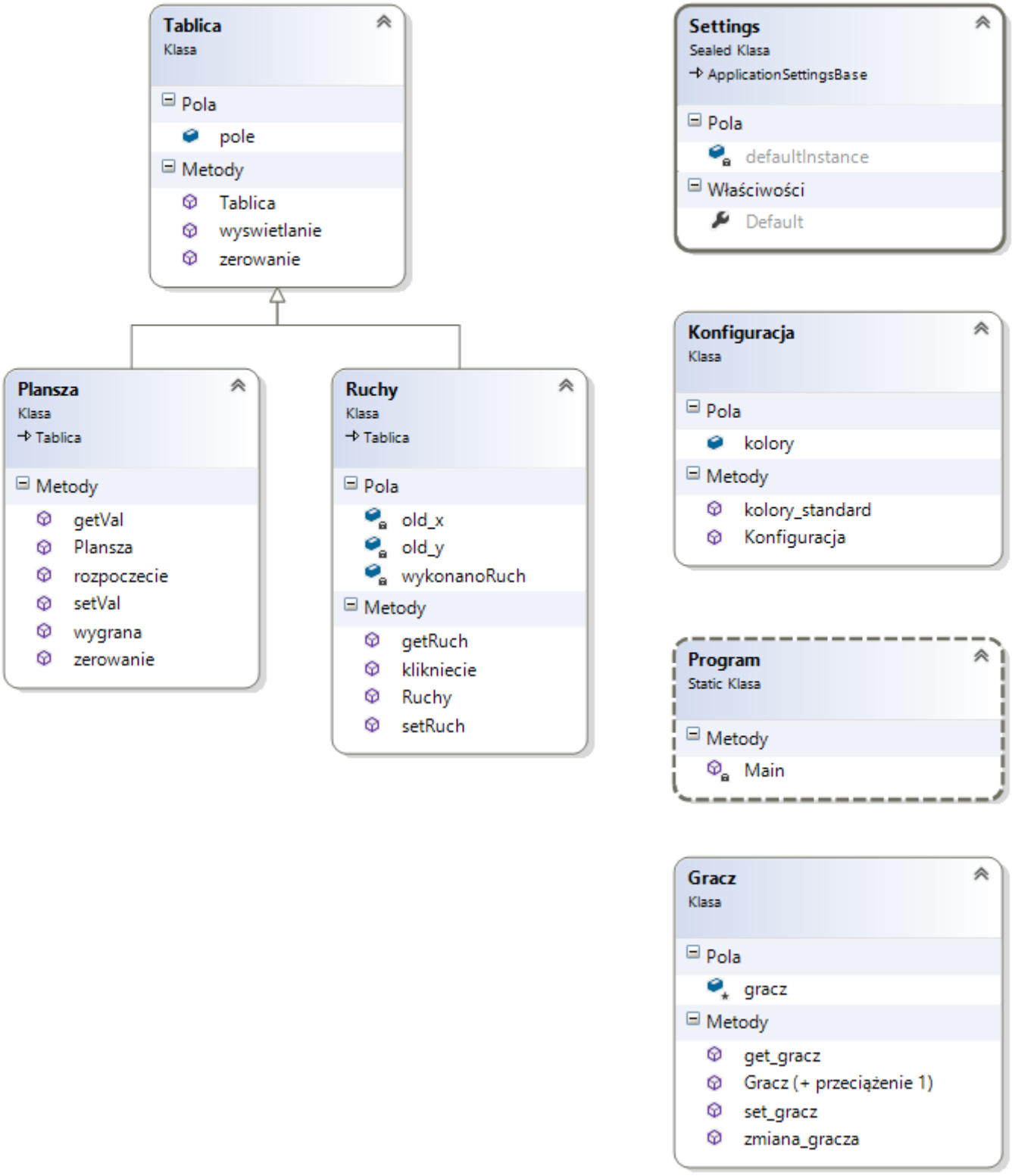
6. Przesortuj listę X od najmniejszej do największej według wartości ruchów (według wartości Score). Przejdź do kroku 7.
7. Jeżeli liczba elementów listy X > 0 to przejdź do kroku 8. W innym przypadku przejdź do kroku 13.
8. Jeżeli wartość ruchu zapisanego w liście X (elementy wybierane na podstawie wartości Score) jest większa od 15 to przypisz ten ruch z listy X do listy A, w innym przypadku przypisz do listy B. Przejdź do kroku 9.
9. Przesortuj listę A i B. Przejdź do kroku 10.
10. Jeżeli liczba elementów listy A > 0 to wylosuj jeden ruch z listy A i przypisz do zmiennej temp. Przejdź do krok 12. W innym przypadku przejdź do kroku 11
11. Wylosuj jeden ruch z listy B i przypisz do zmiennej temp. Przejdź do kroku 12.
12. Przypisz położenie temp pionka i usuń starą pozycję pionka.
13. Koniec algorytmu.

X – lista ruchów owiec (zawiera struktury „para”, które to zawierają współrzędne [i,j] przed wykonaniem ruchu, współrzędne [i,j] po wykonaniu ruchu i wartość wagi ruchu według której będą sortowane ruchy, im większa wartość tym lepszy ruch wybranego pionka)
A - lista lepszych ruchów owiec (tworzona na podstawie listy ruchów owiec)
B – lista gorszych ruchów owiec (tworzona na podstawie listy ruchów owiec)
temp – pomocnicza do której przypisujemy najlepszy ruch z listy ruchów
para – strukutra zawieraja współrzędne [i,j] przed wykonaniem ruchu, współrzędne [i,j] po wykonaniu ruchu i wartość wagi ruchu według której będą sortowane ruchy, im większa wartość tym lepszy ruch wybranego pionka
i,j – liczniki pętli (wartość początkowa to 0), i - odpowiada ze szerokość, j – odpowiada za wysokość
Score – wartość ruchu (zawiera się w strukturze „para”)

(każdy podpunkt korzysta z innych liczników [i,j] jeśli są zdefiniowane na początku podpunktu, w przypadku gdy nie ma definicji liczników [i,j] jak przykładowo w podpunkcie 5. to używane są liczniki z poprzedniego podpunktu, w tym przypadku z 4. punktu)

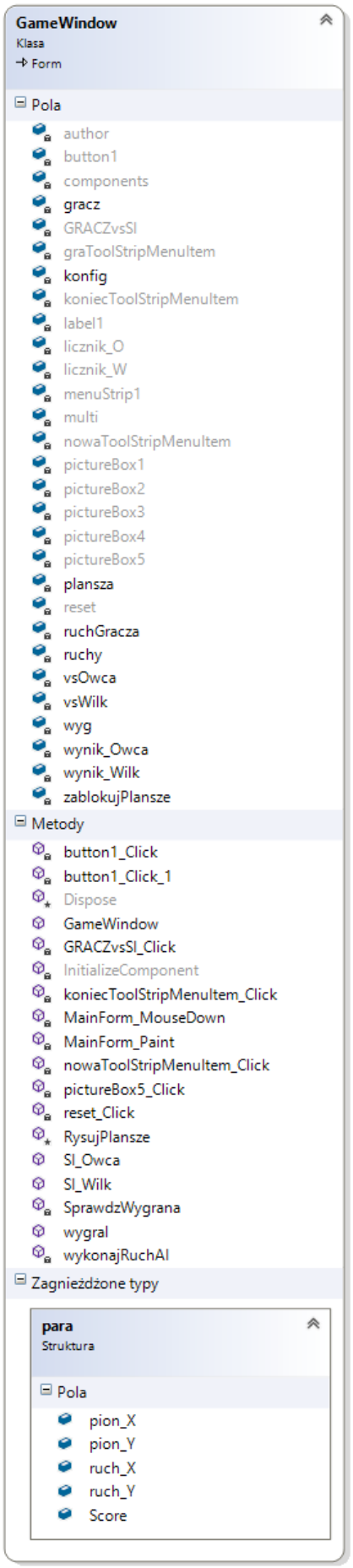
3. Opis programu

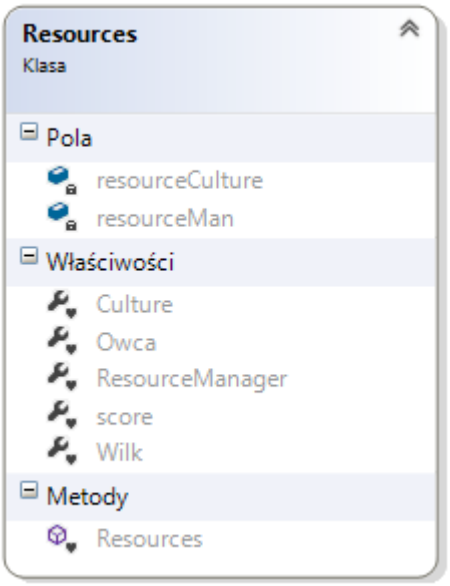
3.1 Diagramy klas



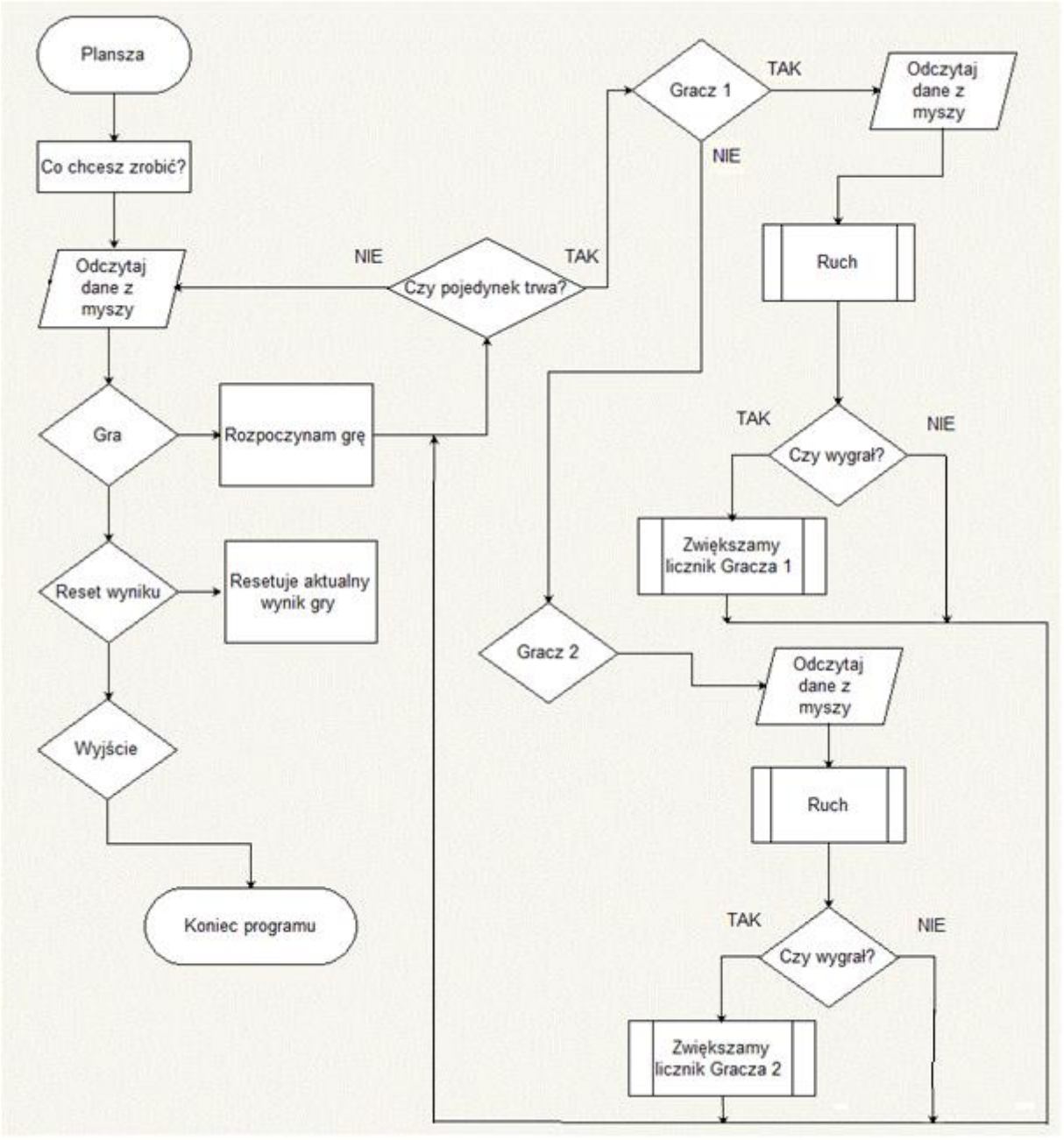
3.1 Diagramy klas

3.2 Schemat blokowy

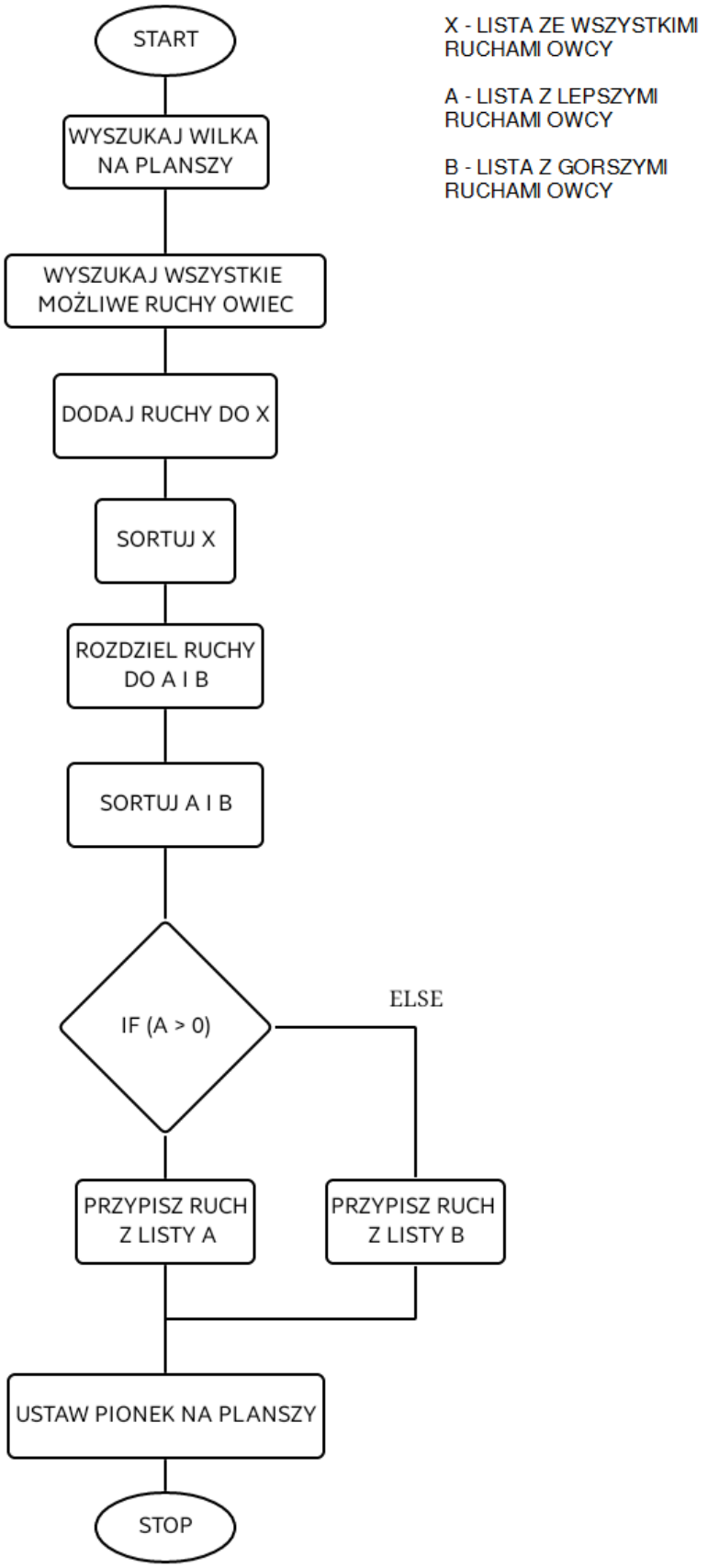




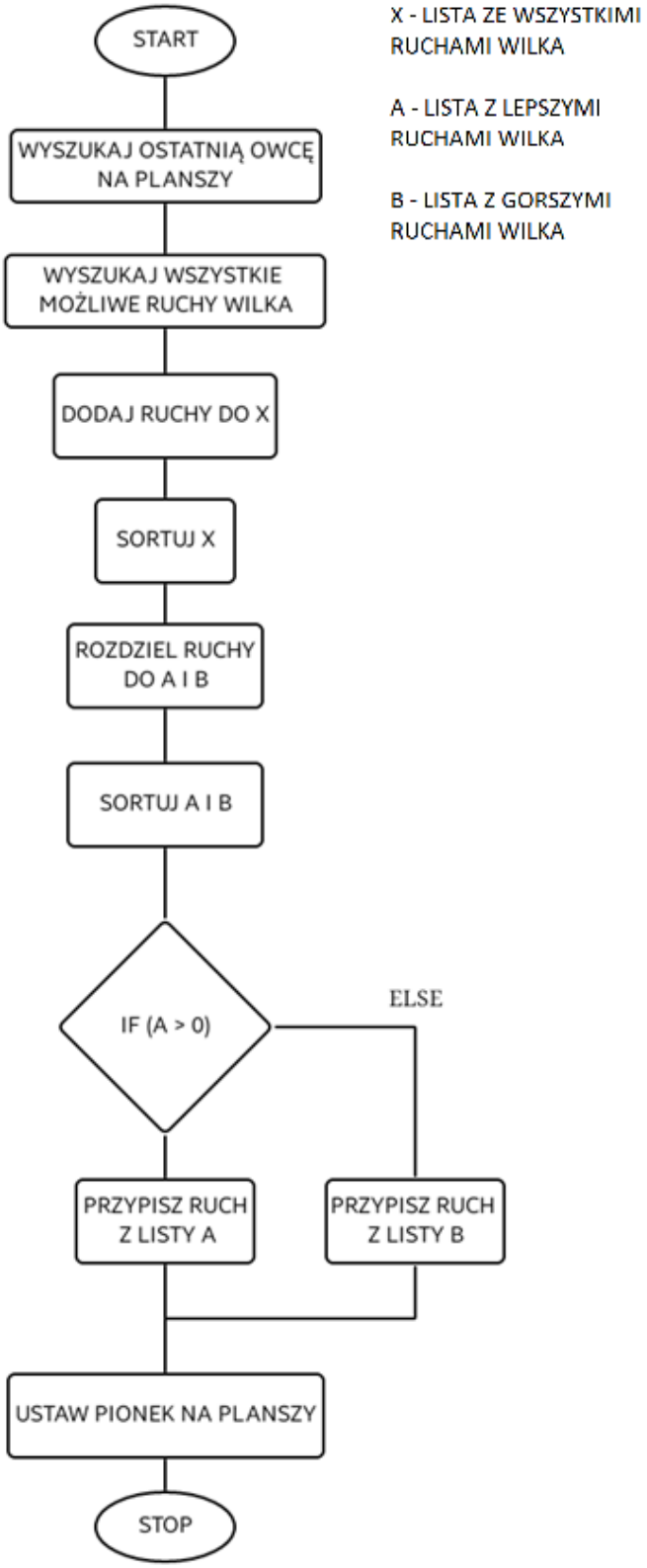
3.2.1 Główny program



3.2.2 Algorytm ruchu owiec



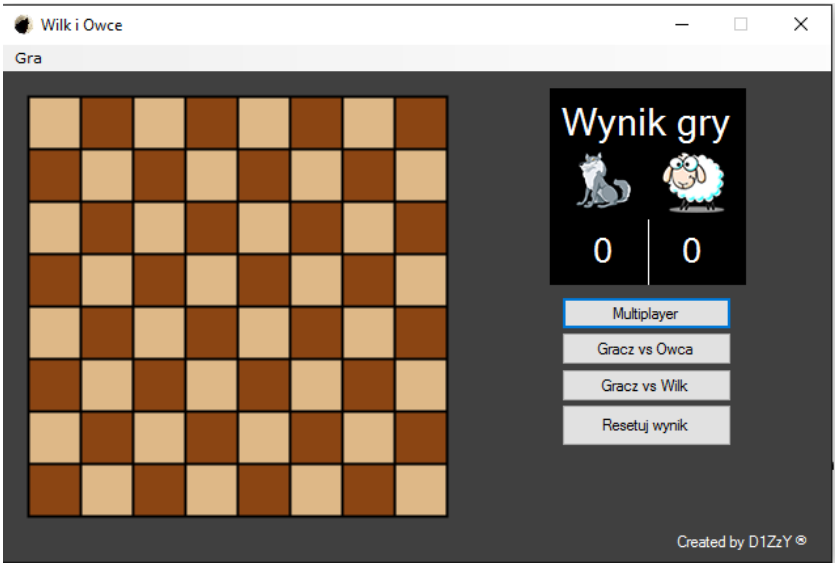
3.2.3 Algorytm ruchu wilka



3.3 Opis poszczególnych komponentów graficznych

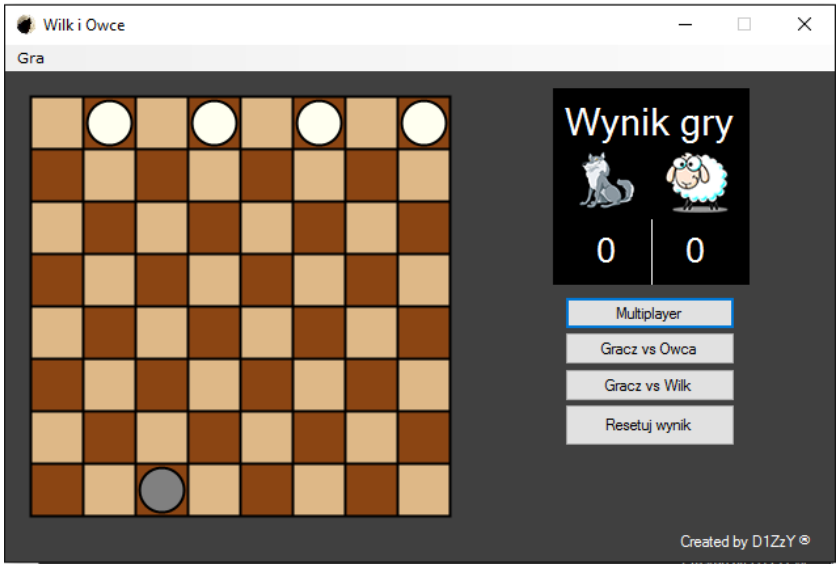
3.3.1 Ekran startowy programu

Ekran startowy ukazuje nam stronę tytułową programu. Zawiera się w niej logo uczelni oraz napis dotyczący czynności jaką należy wykonać, aby przejść do głównego menu.



Główny ekran ukazuje nam interfejs programu. Zawiera się w nim plansza do gry, aktualny wynik gry, pasek menu oraz przyciski funkcyjne.

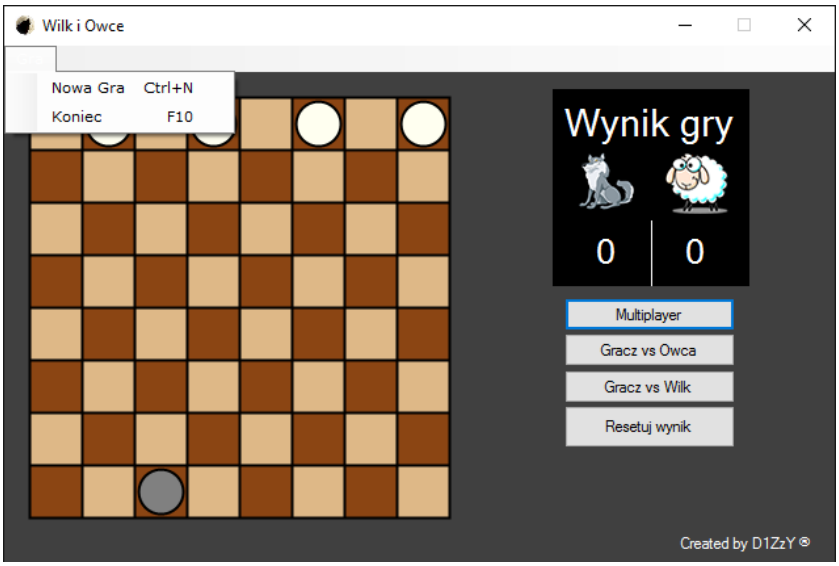
3.3.2 Ekran rozpoczęcia gry



Po naciśnięciu przycisku „Multiplayer”, który odpowiada za rozpoczęcie gry człowiek kontra człowiek, na ekranie programu ukazuje się 5 pionów. Piony białe zawsze mają taki sam układ startowy, natomiast pion szary jest umieszczany losowo na jednym z dolnych pól planszy o ciemnym kolorze.

3.3.3 Pasek menu programu

Na rzucie widnieje interfejs gry z rozwiniętym paskiem menu, w którym użytkownik może rozpocząć nową grę z drugim graczem lub zakończyć działanie programu.



3.3.4 Ekran programu po naciśnięciu na pion

Podczas naciśnięcia na dowolny pion program pokazuje miejsca, w które może poruszyć się aktualny gracz jeśli jest jego kolej.

3.3.5 Stan gry

W prawym górnym roku programu został umieszczony aktualny stan gry.

3.3.6 Powiadomienie o wygranej

Po spełnieniu warunku wygranej którego z graczy, zostaje wyświetlony komunikat o wygranej oraz zwiększony licznik stanu gry.

4. Opis klas oraz metod na podstawie fragmentów programu

4.1 Główny kod programu: GameWindow

```
// definicja odpowiednich bibliotek użytych w programie
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;

namespace PI_Game
{
    public partial class GameWindow : Form
    {

       //zmienne
        AI ai;
        Gracz gracz;
        Konfiguracja konfig;
        Plansza plansza;
        Ruchy ruchy;

        bool zablokujPlansze = false;
        bool ruchGracza = false;
        bool vsOwca = false;
        bool vsWilk = false;
        int wynik_Wilk = 0;
        int wynik_Owca = 0;
        int wyg = 0;

        public GameWindow()
        {
            InitializeComponent();

            ai = new AI();
            gracz = new Gracz();
            konfig = new Konfiguracja();
            plansza = new Plansza();
            ruchy = new Ruchy();
        }

        //komunikat o wygranej
        public void wygral(int w)
        {
            if (w == 2)
            {
                plansza.zerowanie();
                MessageBox.Show("Wygrał Wilk");
                Refresh();
                wynik_Wilk++;
                licznik_W.Text = Convert.ToString(wynik_Wilk);
                zablokujPlansze = true;
            }
            else if (w == 3)
```

```
            {
                plansza.zerowanie();
                MessageBox.Show("Wygrały Owce");
                Refresh();
                wynik_Owca++;
                licznik_O.Text = Convert.ToString(wynik_Owca);
                zablokujPlansze = true;
            }
        }

        //wywolanie metody rysujaca plansze
        private void MainForm_Paint(object sender, PaintEventArgs e)
        {
            Graphics g = e.Graphics;
            RysujPlansze(g);
        }

        //rysowanie planszy
        protected void RysujPlansze(Graphics g)
        {
            Bitmap bitmap = new Bitmap(this.ClientRectangle.Width,
this.ClientRectangle.Height);
            Graphics g2 = Graphics.FromImage(bitmap);
            g2.SmoothingMode = SmoothingMode.HighQuality;

            //tlo
            g2.FillRectangle(new SolidBrush(konfig.kolory[4]), 18, 38, 322, 322);

            for (int j = 0; j < 8; j++)
            {
                for (int i = 0; i < 8; i++)
                {
                    //pola
                    if (plansza.pole[i, j] == 0)
                        g2.FillRectangle(new SolidBrush(konfig.kolory[0]), 20 + 40 *
i, 40 + 40 * j, 38, 38);
                    else
                        g2.FillRectangle(new SolidBrush(konfig.kolory[1]), 20 + 40 *
i, 40 + 40 * j, 38, 38);

                    //zacznaczenie
                    if (ruchy.pole[i, j] == 1)
                        g2.FillRectangle(new SolidBrush(konfig.kolory[5]), 20 + 40 *
i, 40 + 40 * j, 38, 38);
                    if (ruchy.pole[i, j] == 2 || ruchy.pole[i, j] == 3)
                        g2.FillRectangle(new SolidBrush(konfig.kolory[6]), 20 + 40 *
i, 40 + 40 * j, 38, 38);

                    //pionki
                    if (plansza.pole[i, j] > 1)
                    {
                        g2.FillEllipse(new SolidBrush(konfig.kolory[4]), 21 + 40 * i,
41 + 40 * j, 36, 36);

                        if (plansza.pole[i, j] == 2)
                            g2.FillEllipse(new SolidBrush(konfig.kolory[2]), 23 + 40 *
i, 43 + 40 * j, 32, 32);

                        if (plansza.pole[i, j] == 3)
                            g2.FillEllipse(new SolidBrush(konfig.kolory[3]), 23 + 40 *
i, 43 + 40 * j, 32, 32);
                    }
```

```
            }
        }
        g.DrawImageUnscaled(bitmap, 0, 0, bitmap.Width, bitmap.Height);
    }

    //nowa gra w pasku menu
    private void nowaToolStripMenuItem_Click(object sender, EventArgs e)
    {
        plansza.rozpoczecie();
        ruchy.zerowanie();
        gracz.set_gracz(1);
        Refresh();
    }

    //koniec programu w pasku menu
    private void koniecToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Close();
    }

    //obsluga myszki na polu
    private void MainForm_MouseDown(object sender, MouseEventArgs e)
    {
        int x = e.X;
        int y = e.Y;

        if (x >= 20 && x < 340 && y >= 40 && y < 360
            && (x + 22) % 40 != 0 && (x + 21) % 40 != 0
            && (y + 2) % 40 != 0 && (y + 1) % 40 != 0)
        {
            int _x, _y;
            _x = (x - 20) / 40;
            _y = (y - 40) / 40;
            ruchy.klikniecie(_x, _y, plansza, gracz);
            Refresh();
        }
        if (ruchy.getRuch())
        {
            ruchGracza = true;
            ruchy.setRuch(false);
            wykonajRuchAI();
        }
        if (!zablokujPlansze) SprawdzWygrana(); Refresh();
    }

    //wykonanie ruchu Sztucznej Inteligencji
    private void wykonajRuchAI()
    {
        if (vsWilk && ruchGracza)
        {
            ai.SI_Wilk(plansza, gracz, ruchy);
            ruchy.zerowanie();
            Refresh();
            ruchGracza = false;
        }
        if (vsOwca && ruchGracza)
        {
            ai.SI_Owca(plansza, gracz, ruchy);
            ruchy.zerowanie();
            Refresh();
            ruchGracza = false;
        }
```

```
}

//sprawdzenie czy ktoras strona wygrala
private void SprawdzWygrana()
{
    if (vsWilk)
        if (gracz.get_gracz() == 1)
        {
            wyg = plansza.wygrana(gracz, plansza);
            wygral(wyg);
            if (wyg > 0) wyg = 0;
        }
        else
        {
            wyg = plansza.wygrana(gracz, plansza);
            wygral(wyg);
            if (wyg > 0) wyg = 0;
        }

    if (vsOwca)
        if (gracz.get_gracz() == 2)
        {
            ai.SI_Owca(plansza, gracz, ruchy);
            wyg = plansza.wygrana(gracz, plansza);
            wygral(wyg);
            if (wyg > 0) wyg = 0;
        }
        else
        {
            wyg = plansza.wygrana(gracz, plansza);
            wygral(wyg);
            if (wyg > 0) wyg = 0;
        }
    if (!vsOwca && !vsWilk)
        if (gracz.get_gracz() == 1)
        {
            wyg = plansza.wygrana(gracz, plansza);
            wygral(wyg);
            if (wyg > 0) wyg = 0;
        }
        else if (gracz.get_gracz() == 2)
        {
            wyg = plansza.wygrana(gracz, plansza);
            wygral(wyg);
            if (wyg > 0) wyg = 0;
        }
}

//przycisk - gra multiplayer
private void button1_Click(object sender, EventArgs e)
{
    zablokujPlansze = false;
    vsOwca = false;
    vsWilk = false;
    plansza.rozpoczecie();
    ruchy.zerowanie();
    gracz.set_gracz(1);
    Refresh();
}
//przycisk - resetowanie wyniku gry
private void reset_Click(object sender, EventArgs e)
{
```

```
            wynik_Wilk = 0;
            wynik_Owca = 0;
            licznik_W.Text = Convert.ToString(wynik_Wilk);
            licznik_O.Text = Convert.ToString(wynik_Owca);
        }

        //przycisk - gracz vs owca
        private void GRACZvsSI_Click(object sender, EventArgs e)
        {
            zablokujPlansze = false;
            vsOwca = true;
            vsWilk = false;
            plansza.rozpoczecie();
            ruchy.zerowanie();
            gracz.set_gracz(1);
            Refresh();
        }

        //przycisk - gracz vs wilk
        private void button1_Click_1(object sender, EventArgs e)
        {
            zablokujPlansze = false;
            vsOwca = false;
            vsWilk = true;
            plansza.rozpoczecie();
            ruchy.zerowanie();
            gracz.set_gracz(1);
            Thread.Sleep(1000);
            Refresh();
            if (vsWilk) ai.SI_Wilk(plansza, gracz, ruchy);
            ruchy.zerowanie();
            Thread.Sleep(1000);
            Refresh();
        }

        //ekran powitalny
        private void pictureBox5_Click(object sender, EventArgs e)
        {
            pictureBox5.Visible = false;
            pictureBox5.Refresh();
        }

    }
}
```

4.2 Fragment kodu programu: Plansza.cs

```
…
     //rozmieszczenie pionkow na planszy
        public void rozpoczecie()
        {
            zerowanie();
            //      gracz 2 - Owca
            for (int j = 0; j < 1; j++)
                for (int i = 0; i < 8; i++)
                    if (pole[i, j] == 1) pole[i, j] = 3;

            //      gracz 1 - Wilk
            for (int j = 7; j < 8; j++)
            {
                Random r = new Random();
                int i = r.Next(8);
```

```
            //pole[1, 2] = 2;
            if (i % 2 == 0)
            {
                if (pole[i, j] == 1) pole[i, j] = 2;
            }
            else
            {
                i = ((i * 2) - 2) / 2;
                if (pole[i, j] == 1) pole[i, j] = 2;
            }
        }
    }

    //sprawdzenie wygranej graczy
    public int wygrana(Gracz g, Plansza p)
    {
        int Wilk_X = 0;
        int Wilk_Y = 0;
        int Owca_X = 0;
        int Owca_Y = 0;

        for (int j = 0; j < 8; j++)
        {
            for (int i = 0; i < 8; i++)
            {
                if (p.pole[i, j] == 2)
                {
                    Wilk_X = i;
                    Wilk_Y = j;
                }
            }
        }

        int temp = 0;
        for (int j = 0; j < 8; j++)
        {
            for (int i = 0; i < 8; i++)
            {
                //wygrana Wilka
                if (p.pole[i, j] == 3)
                {
                    Owca_X = i;
                    Owca_Y = j;
                    temp++;
                }
                if (temp == 1) break;
            }
            if (temp == 1) break;
        }

        //wygrana Wilka
        if (Wilk_Y <= Owca_Y)
        {
            Console.WriteLine("Wilk wygral!");
            return 2;
        }

        int x = 0, y = 0;
        for (int i = 0; i < 8; i++)
        {
            for (int j = 0; j < 8; j++)
            {
```

```
                    if (p.pole[i, j] == 2)
                    {
                        x = i;
                        y = j;
                        break;
                    }
                }
                if (x != 0 || y != 0)
                    break;
            }

            if (p.pole[x, y] == 2)
            {
                if ((x > 0 && x < 7 && y > 0 && y < 7)
                && p.pole[x - 1, y - 1] != 1
                && p.pole[x + 1, y - 1] != 1
                && p.pole[x - 1, y + 1] != 1
                && p.pole[x + 1, y + 1] != 1)
                {
                    Console.WriteLine("Wygraly Owce!");
                    return 3;
                }
                if ((x == 0 && y == 7) && p.pole[x + 1, y - 1] != 1)
                {
                    Console.WriteLine("Wygraly Owce!");
                    return 3;
                }
                if ((x == 0 && y < 7) && p.pole[x + 1, y - 1] != 1 && p.pole[x + 1, y
+ 1] != 1)
                {
                    Console.WriteLine("Wygraly Owce!");
                    return 3;
                }
                if ((x == 7 && y < 7 && y > 0) && p.pole[x - 1, y - 1] != 1 &&
p.pole[x - 1, y + 1] != 1)
                {
                    Console.WriteLine("Wygraly Owce!");
                    return 3;
                }
                if ((x > 0 && y == 7) && p.pole[x - 1, y - 1] != 1 && p.pole[x + 1, y
- 1] != 1)
                {
                    Console.WriteLine("Wygraly Owce!");
                    return 3;
                }
                else return 0;
            }
            return 0;
}
```

4.3 Fragment kodu programu: Ruchy.cs

```
// Metoda odpowiada za przesuwanie pionów w obiektach, które dziedziczą tablice
//elementów po klasie Tablica

        public void klikniecie(int x, int y, Plansza p, Gracz gracz)
        {

            if (this.pole[x, y] == 0)
            {
                this.zerowanie();
                //jezeli w polu znajduje sie pionek
                if (p.pole[x, y] > 1)
                {
                    //obecnie zaznaczony pionek
                    this.pole[x, y] = 1;

                    //jezeli jest to Wilk
                    if (p.pole[x, y] == 2 && gracz.get_gracz() == 1)
                    {
                        //prawo gora
                        if (x > 0 && y > 0 && p.pole[x - 1, y - 1] == 1)
                            this.pole[x - 1, y - 1] = 2;
                        //prawo dol
                        if (x < 7 && y < 7 && p.pole[x + 1, y + 1] == 1)
                            this.pole[x + 1, y + 1] = 2;
                        //lewo gora
                        if (x < 7 && y > 0 && p.pole[x + 1, y - 1] == 1)
                            this.pole[x + 1, y - 1] = 2;
                        //lewo dol
                        if (x > 0 && y < 7 && p.pole[x - 1, y + 1] == 1)
                            this.pole[x - 1, y + 1] = 2;
                    }

                    //jezeli jest to Owca
                    else if (p.pole[x, y] == 3 && gracz.get_gracz() == 2)
                    {
                        //lewo dol
                        if (x > 0 && y < 7 && p.pole[x - 1, y + 1] == 1)
                            this.pole[x - 1, y + 1] = 2;
                        //prawo dol
                        if (x < 7 && y < 7 && p.pole[x + 1, y + 1] == 1)
                            this.pole[x + 1, y + 1] = 2;
                    }

                    this.wyswietlanie();
                }
                //przypisanie nowego polozenia pionka
                old_x = x;
                old_y = y;
            }
            else if (this.pole[x, y] == 2)
            {
                p.pole[x, y] = p.pole[old_x, old_y];
                p.pole[old_x, old_y] = 1;
                wykonanoRuch = true;
                this.zerowanie();
                gracz.zmiana_gracza();
            }

        }
```

4.4 Fragment kodu programu: AI.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PI_Game
{
    public struct para
    {
        public int pion_X;
        public int pion_Y;
        public int ruch_X;
        public int ruch_Y;
        public int Score;
        public para(int x)
        {
            pion_X = x;
            pion_Y = x;
            ruch_X = x;
            ruch_Y = x;
            Score = x;
        }
    }

    class AI
    {

        //wyszukiwanie Wilka
        public para pozycjaWilka(Plansza p)
        {
            para temp = new para(0);

            for (int j = 0; j < 8; j++)
            {
                for (int i = 0; i < 8; i++)
                {
                    if (p.pole[i, j] == 2)
                    {
                        temp.pion_X = i;
                        temp.pion_Y = j;
                    }
                }
            }
            return temp;
        }

        //wyszukiwanie ostatniej Owcy
        public para ostatniaOwca(Plansza p)
        {
            para temp = new para(0);

            for (int _j = 0; _j < 8; _j++)
            {
                for (int _i = 0; _i < 8; _i++)
                {
                    if (p.pole[_i, _j] == 3)
                    {
                        temp.pion_X = _i;
```

```
                temp.pion_Y = _j;
                temp.Score++;
            }
            if (temp.Score == 1) break;
        }
        if (temp.Score == 1) break;
    }
    return temp;
}

//wyszukiwanie Owcy najblizej strony Wilka
public para najblizaOwca(Plansza p)
{
    para temp = new para(0);

    for (int _j = 7; _j >= 0; _j--)
    {
        for (int _i = 7; _i >= 0; _i--)
        {
            if (p.pole[_i, _j] == 3)
            {
                temp.pion_X = _i;
                temp.pion_Y = _j;
                temp.Score++;
            }
            if (temp.Score == 1) break;
        }
        if (temp.Score == 1) break;
    }
    return temp;
}

//sprawdza czy dwie Owce sa obok siebie
public para dwieOwcePrzed(Plansza p)
{
    para temp = new para(0);

    for (int _j = 7; _j >= 0; _j--)
    {
        temp.Score = 0;
        for (int _i = 7; _i >= 0; _i--)
        {
            if (p.pole[_i, _j] == 3)
            {
                temp.pion_X = _i;
                temp.pion_Y = _j;
                temp.Score++;
            }
            if (temp.Score == 2) break;
        }
        if (temp.Score == 2) break;
    }
    return temp;
}

//sprawdza czy trzy Owce sa obok siebie
public para trzyOwcePrzed(Plansza p)
{
    para temp = new para(0);

    for (int _j = 7; _j >= 0; _j--)
    {
```

```
                temp.Score = 0;
                for (int _i = 7; _i >= 0; _i--)
                {
                    if (p.pole[_i, _j] == 3)
                    {
                        temp.pion_X = _i;
                        temp.pion_Y = _j;
                        temp.Score++;
                    }
                    if (temp.Score == 3) break;
                }
                if (temp.Score == 3) break;
            }
            return temp;
        }
        //sprawdza czy 4 Owce sa obok siebie
        public para czteryOwcePrzed(Plansza p)
        {
            para temp = new para(0);

            for (int _j = 7; _j >= 0; _j--)
            {
                temp.Score = 0;
                for (int _i = 7; _i >= 0; _i--)
                {
                    if (p.pole[_i, _j] == 3)
                    {
                        temp.pion_X = _i;
                        temp.pion_Y = _j;
                        temp.Score++;
                    }
                    if (temp.Score == 4) break;
                }
                if (temp.Score == 4) break;
            }
            return temp;
        }


//************************//////////////////////////////************************//
//************************//Sztuczna inteligencja Owciec//************************//
//************************//////////////////////////////************************//

        public void SI_Owca(Plansza p, Gracz g, Ruchy r)
        {
            List<para> ruchy_owiec = new List<para>();
            List<para> best = new List<para>();
            List<para> worst = new List<para>();

            para pom = new para(0);
            para Wilk = new para(0);
            para ostOwca = new para(0);
            para najbOwca = new para(0);
            para dwieOwce = new para(0);
            para trzyOwce = new para(0);
            para czteryOwce = new para(0);

            //wyszukiwanie Wilka
            Wilk = pozycjaWilka(p);

            //wyszuknie polozenia ostaniej Owcy
            ostOwca = ostatniaOwca(p);
```

```
            //wyszukiwanie Owcy najblizej strony Wilka
            najbOwca = najblizaOwca(p);

            //dwie owce
            dwieOwce = dwieOwcePrzed(p);

            //trzy owce
            trzyOwce = trzyOwcePrzed(p);

            //cztery owce
            czteryOwce = czteryOwcePrzed(p);

            for (int j = 0; j < 8; j++)
                for (int i = 0; i < 8; i++)
                    if (p.pole[i, j] == 3)
                    {
                        r.klikniecie(i, j, p, g);
                        for (int n = 0; n < 8; n++)
                        {
                            for (int m = 0; m < 8; m++)
                            {
                                if (r.pole[m, n] == 1)
                                {
                                    for (int y = 0; y < 8; y++)
                                    {
                                        for (int x = 0; x < 8; x++)
                                            if (r.pole[x, y] == 2 && p.pole[x, y] ==
1)
                                            {
                                                //wspolrzedne aktualnego polozenia
pionka
                                                pom.pion_X = m;
                                                pom.pion_Y = n;

                                                //wspolrzedne kolejnego ruchu
                                                pom.ruch_X = x;
                                                pom.ruch_Y = y;

                                                //bonusy
                                                int bonus_sasiad = 0;
                                                int bonus_za_wilka = 0;

                                                //punkty ujemne za odejscie od Wilka
                                                int kara = 0;

                                                //jesli wygrana Owiec
                                                if (Wilk.pion_Y - pom.ruch_Y == 1 ||
Wilk.pion_Y - pom.ruch_Y == 0 || Wilk.pion_Y - pom.ruch_Y == -1)
                                                {
                                                    if ((Wilk.pion_X > 0 &&
Wilk.pion_X < 7 && Wilk.pion_Y > 0 && Wilk.pion_Y < 7)
                                                    && p.pole[Wilk.pion_X - 1,
Wilk.pion_Y - 1] != 1
                                                    && p.pole[Wilk.pion_X + 1,
Wilk.pion_Y - 1] != 1
                                                    && p.pole[Wilk.pion_X - 1,
Wilk.pion_Y + 1] != 1
                                                    && p.pole[Wilk.pion_X + 1,
Wilk.pion_Y + 1] != 1)
                                                    {
                                                        pom.Score = 100;
                                                        ruchy_owiec.Add(pom);
```

```
                                                }
                                                else if ((Wilk.pion_X == 0 &&
Wilk.pion_Y == 7) && (Wilk.pion_X + 1 == pom.ruch_X && Wilk.pion_Y - 1 == pom.ruch_Y))
                                                {
                                                    pom.Score = 100;
                                                    ruchy_owiec.Add(pom);
                                                }
                                                else if((Wilk.pion_X == 0 &&
Wilk.pion_Y < 7 && Wilk.pion_Y > 0) && (p.pole[Wilk.pion_X + 1, Wilk.pion_Y - 1] == 3
                                                    && (Wilk.pion_X + 1 ==
pom.ruch_X && Wilk.pion_Y + 1 == pom.ruch_Y)))
                                                {
                                                    pom.Score = 100;
                                                    ruchy_owiec.Add(pom);
                                                }
                                                else if((Wilk.pion_X == 0 &&
Wilk.pion_Y < 7 && Wilk.pion_Y > 0) && (p.pole[Wilk.pion_X + 1, Wilk.pion_Y + 1] == 3
                                                    && (Wilk.pion_X + 1 ==
pom.ruch_X && Wilk.pion_Y - 1 == pom.ruch_Y)))
                                                {
                                                    pom.Score = 100;
                                                    ruchy_owiec.Add(pom);
                                                }
                                                else if((Wilk.pion_X == 7 &&
Wilk.pion_Y < 7 && Wilk.pion_Y > 0) && (p.pole[Wilk.pion_X - 1, Wilk.pion_Y - 1] == 3
                                                    && (Wilk.pion_X - 1 ==
pom.ruch_X && Wilk.pion_Y + 1 == pom.ruch_Y)))
                                                {
                                                    pom.Score = 100;
                                                    ruchy_owiec.Add(pom);
                                                }
                                                else if((Wilk.pion_X == 7 &&
Wilk.pion_Y < 7 && Wilk.pion_Y > 0) && (p.pole[Wilk.pion_X - 1, Wilk.pion_Y + 1] == 3
                                                    && (Wilk.pion_X - 1 ==
pom.ruch_X && Wilk.pion_Y - 1 == pom.ruch_Y)))
                                                {
                                                    pom.Score = 100;
                                                    ruchy_owiec.Add(pom);
                                                }
                                                else if((Wilk.pion_X > 0 &&
Wilk.pion_X < 7 && Wilk.pion_Y == 7) && (p.pole[Wilk.pion_X - 1, Wilk.pion_Y - 1] == 3
                                                    && (Wilk.pion_X + 1 ==
pom.ruch_X && Wilk.pion_Y - 1 == pom.ruch_Y)))
                                                {
                                                    pom.Score = 100;
                                                    ruchy_owiec.Add(pom);
                                                }
                                                else if((Wilk.pion_X > 0 &&
Wilk.pion_X < 7 && Wilk.pion_Y == 7) && (p.pole[Wilk.pion_X + 1, Wilk.pion_Y - 1] == 3
                                                    && (Wilk.pion_X - 1 ==
pom.ruch_X && Wilk.pion_Y - 1 == pom.ruch_Y)))
                                                {
                                                    pom.Score = 100;
                                                    ruchy_owiec.Add(pom);
                                                }
                                            }
                                            //wygrana Wilka
                                            if (ostOwca.pion_Y - Wilk.pion_Y == -1
&& Math.Abs(Wilk.pion_X - pom.ruch_X) == 2)
                                            {
                                                pom.Score = -100;
                                                ruchy_owiec.Add(pom);
```

```
                                            }

                                            else if (pom.pion_X > 1 && pom.pion_X
< 6 && (Wilk.pion_Y - pom.ruch_Y == 0))
                                            {
                                                pom.Score = -20;
                                                ruchy_owiec.Add(pom);
                                            }
                                            else if (pom.ruch_X >= 4 &&
Wilk.pion_X >= 4 && Wilk.pion_Y - pom.ruch_Y == 2 )
                                            {
                                                pom.Score = -80;
                                                ruchy_owiec.Add(pom);
                                            }
                                            else if (pom.ruch_X < 4 && Wilk.pion_X
< 4 && Wilk.pion_Y - pom.ruch_Y == 2)
                                            {
                                                pom.Score = -80;
                                                ruchy_owiec.Add(pom);
                                            }
                                            //dolaczenie drugiego pionka
                                            else if (Math.Abs(pom.ruch_X -
najbOwca.pion_X) == 2 && trzyOwce.pion_Y < pom.ruch_Y && pom.ruch_Y ==
najbOwca.pion_Y)
                                            {
                                                //jesli wilk przed kara przed
ruszeniem
                                                if (Math.Abs(pom.pion_X -
Wilk.pion_X) == 1 && Wilk.pion_Y - pom.pion_Y == 1)
                                                {
                                                    kara = -70;
                                                }
                                                pom.Score = 50 + kara;
                                                ruchy_owiec.Add(pom);
                                            }
                                            //dolacznie trzeciego pionka
                                            else if (dwieOwce.pion_Y == pom.ruch_Y
&& dwieOwce.Score == 2 && (dwieOwce.pion_X - pom.ruch_X == 2 || dwieOwce.pion_X -
pom.ruch_X == 4))
                                            {
                                                //jesli wilk przed kara przed
ruszeniem
                                                if (Math.Abs(pom.pion_X -
Wilk.pion_X) == 1 && Wilk.pion_Y - pom.pion_Y == 1)
                                                {
                                                    kara = -70;
                                                }
                                                pom.Score = 70 + kara;
                                                ruchy_owiec.Add(pom);
                                            }
                                            //dolacznie czwartego pionka
                                            else if (trzyOwce.pion_Y == pom.ruch_Y
&& trzyOwce.Score == 3)
                                            {
                                                //jesli wilk przed kara przed
ruszeniem
                                                if (Math.Abs(pom.pion_X -
Wilk.pion_X) == 1 && Wilk.pion_Y - pom.pion_Y == 1)
                                                {
                                                    kara = -70;
                                                }
                                                pom.Score = 80 + kara;
```

```
                                                    ruchy_owiec.Add(pom);
                                                }
                                                //jesli jest sasiad po prawej i
zblokowany wilk
                                                else if (pom.ruch_X + 2 <= 7 &&
(p.pole[pom.ruch_X + 2, pom.ruch_Y] == 3
                                                    && p.pole[pom.ruch_X + 1,
pom.pion_Y + 1] == 2))
                                                {
                                                    //jesli wilk przed kara przed
ruszeniem
                                                    if (Math.Abs(pom.pion_X -
Wilk.pion_X) == 1 && Wilk.pion_Y - pom.pion_Y == 1)
                                                    {
                                                        kara = kara + -70;
                                                    }
                                                    pom.Score = 75;
                                                    ruchy_owiec.Add(pom);
                                                }
                                                //jesli jest sasiad po lewej i
zblokowany wilk
                                                else if (pom.ruch_X - 2 >= 0 &&
(p.pole[pom.ruch_X - 2, pom.ruch_Y] == 3
                                                    && p.pole[pom.ruch_X - 1,
pom.pion_Y + 1] == 2))
                                                {
                                                    //jesli wilk przed kara przed
ruszeniem
                                                    if (Math.Abs(pom.pion_X -
Wilk.pion_X) == 1 && Wilk.pion_Y - pom.pion_Y == 1)
                                                    {
                                                        kara = kara + -70;
                                                    }
                                                    pom.Score = 75;
                                                    ruchy_owiec.Add(pom);
                                                }
                                                else
                                                {
                                                    //bonus za sasiada
                                                    if (pom.ruch_X + 2 <= 7)
                                                        if (p.pole[pom.ruch_X + 2,
pom.ruch_Y] == 3)
                                                            bonus_sasiad = 5;
                                                    if (pom.ruch_X - 2 >= 0)
                                                        if (p.pole[pom.ruch_X - 2,
pom.ruch_Y] == 3)
                                                            bonus_sasiad = 5;
                                                    //bonus za wilka po skosie przed
soba
                                                    if (pom.ruch_X + 1 <= 7 &&
pom.ruch_Y + 1 <= 7)
                                                        if (p.pole[pom.ruch_X + 1,
pom.ruch_Y + 1] == 2)
                                                            bonus_za_wilka = 20;
                                                    if (pom.ruch_X - 1 >= 0 &&
pom.ruch_Y + 1 <= 7)
                                                        if (p.pole[pom.ruch_X - 1,
pom.ruch_Y + 1] == 2)
                                                            bonus_za_wilka = 20;
                                                    //kara
                                                    if (Wilk.pion_Y - pom.ruch_Y < 0)
                                                        kara = -70;
```

```
                                                    //jesli wilk przed kara przed
ruszeniem
                                                    if (Math.Abs(pom.pion_X -
Wilk.pion_X) == 1 && Wilk.pion_Y - pom.pion_Y == 1)
                                                    {
                                                        kara = kara + -70;
                                                    }

                                                    pom.Score = 0 + bonus_sasiad +
bonus_za_wilka + kara;
                                                    ruchy_owiec.Add(pom);
                                                }
                                            }
                                        }
                                    }
                                }
                            }

            //sprawdzamy czy sa mozliwe jakies ruchy
            if (ruchy_owiec.Count > 0)
            {
                //sortowanie ruchow od najgorszych do najlepszych
                ruchy_owiec = ruchy_owiec.OrderBy(x => x.Score).ToList();

                pom = ruchy_owiec[0];

                for (int i = 0; i < ruchy_owiec.Count; i++)
                {
                    int temp_count = 1;

                    if (ruchy_owiec.Count >= 3) temp_count = 3;
                    if (ruchy_owiec.Count == 2) temp_count = 2;
                    if (ruchy_owiec.Count == 1) temp_count = 1;

                    System.Console.WriteLine("Owca => ruch nr " + i + " z [" +
ruchy_owiec[i].pion_X + ";" + ruchy_owiec[i].pion_Y + "] do "
                        + "[" + ruchy_owiec[i].ruch_X + "; " + ruchy_owiec[i].ruch_Y +
"] wynik to: " + ruchy_owiec[i].Score);
                    if (ruchy_owiec[i].Score > 0)
                    {
                        pom = ruchy_owiec[i];
                        best.Add(pom);
                    }
                    else
                    {
                        pom = ruchy_owiec[i];
                        worst.Add(pom);
                    }
                }
                //sortowanie ruchow listy z lepszymi i gorszymi
                best = best.OrderBy(x => x.Score).ToList();
                worst = worst.OrderBy(x => x.Score).ToList();

                //wybieramy ruch
                do
                {
                    if (best.Count > 0)
                    {
                        //Random rnd = new Random();
                        //pom = best[rnd.Next(0, best.Count)];
                        pom = best[best.Count - 1];
```

```
                    }
                    else
                    {
                        //Random rnd = new Random();
                        //pom = worst[rnd.Next(0, worst.Count)];
                        pom = worst[worst.Count - 1];
                    }
                } while (p.pole[pom.ruch_X, pom.ruch_Y] != 1);

                p.pole[pom.pion_X, pom.pion_Y] = 1; //pusty
                p.pole[pom.ruch_X, pom.ruch_Y] = 3; //owca
                g.zmiana_gracza();
            }
            else { }
        }

    //************************////////////////////////////////**********************//
    //************************//Sztuczna inteligencja Wilka//**********************//
    //************************////////////////////////////////**********************//

        public void SI_Wilk(Plansza p, Gracz g, Ruchy r)
        {
            List<para> ruchy_wilka = new List<para>();
            List<para> best = new List<para>();
            List<para> worst = new List<para>();

            para pom = new para(0);
            para Wilk = new para(0);
            para ostOwaca = new para(0);

            //wyszuknie polozenia Wilka
            Wilk = pozycjaWilka(p);

            //wyszuknie polozenia ostaniej Owcy
            ostOwaca = ostatniaOwca(p);

            //wybieramy pozycje w ktorej znajduje sie Wilk
            r.klikniecie(Wilk.pion_X, Wilk.pion_Y, p, g);

            for (int y = 0; y < 8; y++)
                for (int x = 0; x < 8; x++)
                    if (r.pole[x, y] == 2 && p.pole[x, y] == 1)
                    {
                        //wspolrzedne aktualnego polozenia pionka
                        pom.pion_X = Wilk.pion_X;
                        pom.pion_Y = Wilk.pion_Y;
                        //wspolrzedne kolejnego ruchu
                        pom.ruch_X = x;
                        pom.ruch_Y = y;

                        //wygrana Wilka
                        if (pom.ruch_Y == ostOwaca.pion_Y)
                        {
                            pom.Score = 100;
                            ruchy_wilka.Add(pom);
                        }
                        //jesli wygrana owcy
                        if (pom.ruch_Y - ostOwaca.pion_Y == 1 || pom.ruch_Y -
ostOwaca.pion_Y == 0 || pom.ruch_Y - ostOwaca.pion_Y == -1)
                        {
                            if ((Wilk.pion_X > 0 && Wilk.pion_X < 7 && Wilk.pion_Y > 0
&& Wilk.pion_Y < 7)
```

```
                            && p.pole[Wilk.pion_X - 1, Wilk.pion_Y - 1] != 1
                            && p.pole[Wilk.pion_X + 1, Wilk.pion_Y - 1] != 1
                            && p.pole[Wilk.pion_X - 1, Wilk.pion_Y + 1] != 1
                            && p.pole[Wilk.pion_X + 1, Wilk.pion_Y + 1] != 1)
                            {
                                pom.Score = -100;
                                ruchy_wilka.Add(pom);
                            }
                            if ((Wilk.pion_X == 0 && Wilk.pion_Y == 7) && (Wilk.pion_X
+ 1 == ostOwaca.pion_X && Wilk.pion_Y - 1 == ostOwaca.pion_Y))
                            {
                                pom.Score = -100;
                                ruchy_wilka.Add(pom);
                            }
                            if ((Wilk.pion_X == 0 && Wilk.pion_Y < 7 && Wilk.pion_Y >
0) && (p.pole[Wilk.pion_X + 1, Wilk.pion_Y - 1] == 3
                                && (Wilk.pion_X + 1 == ostOwaca.pion_X && Wilk.pion_Y
+ 1 == ostOwaca.pion_Y)))
                            {
                                pom.Score = -100;
                                ruchy_wilka.Add(pom);
                            }
                            if ((Wilk.pion_X == 0 && Wilk.pion_Y < 7 && Wilk.pion_Y >
0) && (p.pole[Wilk.pion_X + 1, Wilk.pion_Y + 1] == 3
                                && (Wilk.pion_X + 1 == ostOwaca.pion_X && Wilk.pion_Y
- 1 == ostOwaca.pion_Y)))
                            {
                                pom.Score = -100;
                                ruchy_wilka.Add(pom);
                            }
                            if ((Wilk.pion_X == 7 && Wilk.pion_Y < 7 && Wilk.pion_Y >
0) && (p.pole[Wilk.pion_X - 1, Wilk.pion_Y - 1] == 3
                                && (Wilk.pion_X - 1 == ostOwaca.pion_X && Wilk.pion_Y
+ 1 == ostOwaca.pion_Y)))
                            {
                                pom.Score = -100;
                                ruchy_wilka.Add(pom);
                            }
                            if ((Wilk.pion_X == 7 && Wilk.pion_Y < 7 && Wilk.pion_Y >
0) && (p.pole[Wilk.pion_X - 1, Wilk.pion_Y + 1] == 3
                                && (Wilk.pion_X - 1 == ostOwaca.pion_X && Wilk.pion_Y
- 1 == ostOwaca.pion_Y)))
                            {
                                pom.Score = -100;
                                ruchy_wilka.Add(pom);
                            }
                            if ((Wilk.pion_X > 0 && Wilk.pion_X < 7 && Wilk.pion_Y ==
7) && (p.pole[Wilk.pion_X - 1, Wilk.pion_Y - 1] == 3
                                && (Wilk.pion_X + 1 == ostOwaca.pion_X && Wilk.pion_Y
- 1 == ostOwaca.pion_Y)))
                            {
                                pom.Score = -100;
                                ruchy_wilka.Add(pom);
                            }
                            if ((Wilk.pion_X > 0 && Wilk.pion_X < 7 && Wilk.pion_Y ==
7) && (p.pole[Wilk.pion_X + 1, Wilk.pion_Y - 1] == 3
                                && (Wilk.pion_X - 1 == ostOwaca.pion_X && Wilk.pion_Y
- 1 == ostOwaca.pion_Y)))
                            {
                                pom.Score = -100;
                                ruchy_wilka.Add(pom);
                            }
```

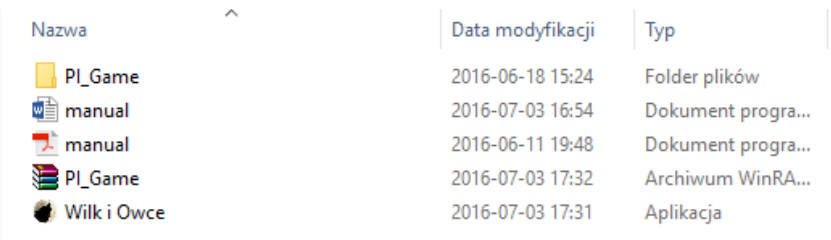
```
                        }
                        //wilk do przodu jesli jest puste miejsce <1-6>
                        if (Wilk.pion_X > 0 && Wilk.pion_X < 7 && Wilk.pion_Y >
pom.ruch_Y)
                        {
                            pom.Score = 25;
                            ruchy_wilka.Add(pom);
                        }
                        //wilk do przodu jesli jest puste miejsce <0>
                        else if ((Wilk.pion_X == 0 || Wilk.pion_Y == 7) && Wilk.pion_Y
< 7 && Wilk.pion_Y > pom.ruch_Y)
                        {
                            pom.Score = 15;
                            ruchy_wilka.Add(pom);
                        }
                        //wilk do tylu jesli jest puste miejsce <1-6>
                        else if (Wilk.pion_X > 0 && Wilk.pion_X < 7 && Wilk.pion_Y <
pom.ruch_Y)
                        {
                            pom.Score = 10;
                            ruchy_wilka.Add(pom);
                        }
                        //wilk do tylu jesli jest puste miejsce <0>
                        else if ((Wilk.pion_X == 0 || Wilk.pion_X == 7) && Wilk.pion_Y
< 7 && Wilk.pion_Y < pom.ruch_Y)
                        {
                            pom.Score = 5;
                            ruchy_wilka.Add(pom);
                        }
                        //wilk do tylu jesli po prawej stronie jest puste miejsce x =
0, y = 7
                        else if (Wilk.pion_X == 0 && Wilk.pion_Y == 7)
                        {
                            pom.Score = 5;
                            ruchy_wilka.Add(pom);
                        }
                    }


            //sortujemy dobre i zle ruchy
            if (ruchy_wilka.Count > 0)
            {
                //sortowanie listy
                ruchy_wilka = ruchy_wilka.OrderBy(x => x.Score).ToList();

                pom = ruchy_wilka[0];

                for (int i = 0; i < ruchy_wilka.Count; i++)
                {
                    int temp_count = 1;

                    if (ruchy_wilka.Count >= 3) temp_count = 3;
                    if (ruchy_wilka.Count == 2) temp_count = 2;
                    if (ruchy_wilka.Count == 1) temp_count = 1;

                    System.Console.WriteLine("Wilk => ruch nr " + i + " z [" +
ruchy_wilka[i].pion_X + ";" + ruchy_wilka[i].pion_Y + "] do "
                    + "[" + ruchy_wilka[i].ruch_X + "; " + ruchy_wilka[i].ruch_Y + "]
wynik to: " + ruchy_wilka[i].Score);
                    if (ruchy_wilka[i].Score >= ruchy_wilka[ruchy_wilka.Count -
temp_count].Score && ruchy_wilka[i].Score > 15)
                    {
                        pom = ruchy_wilka[i];
```

```
                        best.Add(pom);
                    }
                    else
                    {
                        pom = ruchy_wilka[i];
                        worst.Add(pom);
                    }
                }
                //sortowanie list
                best = best.OrderBy(x => x.Score).ToList();
                worst = worst.OrderBy(x => x.Score).ToList();

                //wybieramy ruch
                do
                {
                    if (best.Count > 0)
                    {
                        //przypisanie ruchu do pom
                        Random rnd = new Random();
                        pom = best[rnd.Next(0, best.Count)];
                    }
                    else
                    {
                        Random rnd = new Random();
                        pom = worst[rnd.Next(0, worst.Count)];
                    }
                } while (p.pole[pom.ruch_X, pom.ruch_Y] != 1);

                p.pole[pom.pion_X, pom.pion_Y] = 1; //pusty
                p.pole[pom.ruch_X, pom.ruch_Y] = 2; //wilk
                g.zmiana_gracza();
            }
            else { }
        }

    }
}
```

5. Instrukcja obsługi

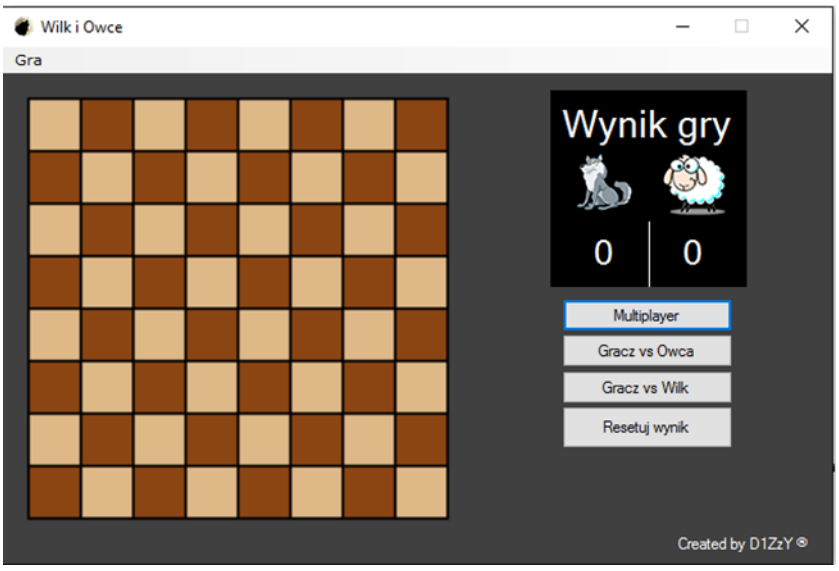
5.1 Uruchomienie aplikacji

1. Aby uruchomić grę należy umieścić dysk CD w napędzie komputera, po ukazaniu się zawartości płyty naciskamy dwa razy na „Wilk i Owce.exe”.



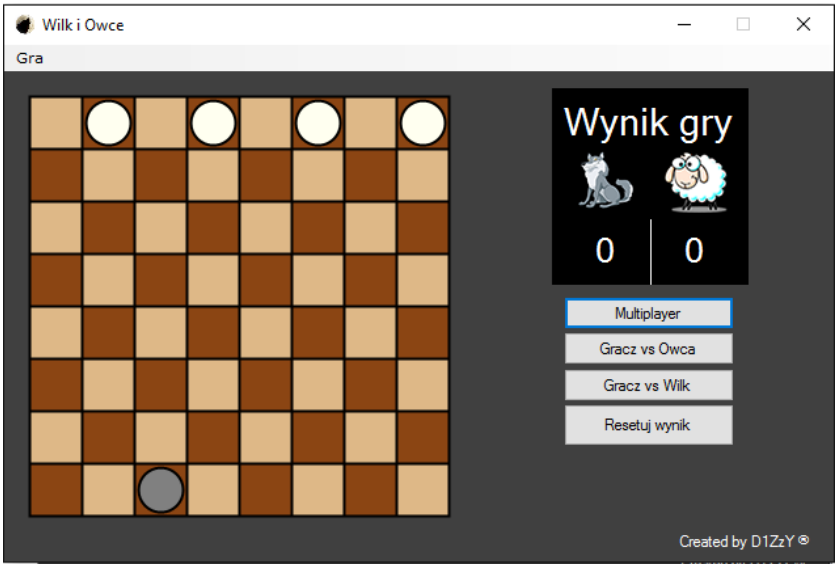
2. Po włączeniu programu zobaczymy poniższy ekran powitalny, aby przejść do menu gry musimy kliknąć na tło programu.

3. Po naciśnięciu na tło zostanie wyświetlone główne menu z szachownicą i wynikiem gry.

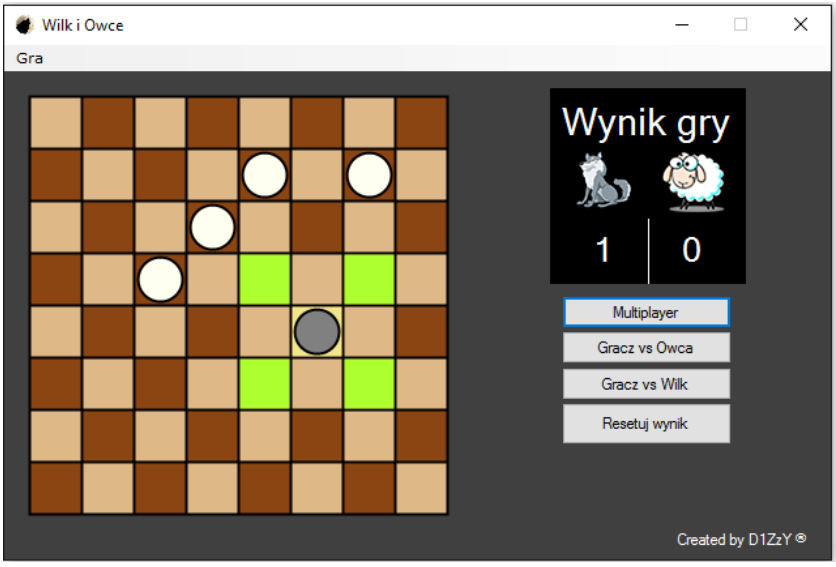


5.2 Przykładowa rozgrywka

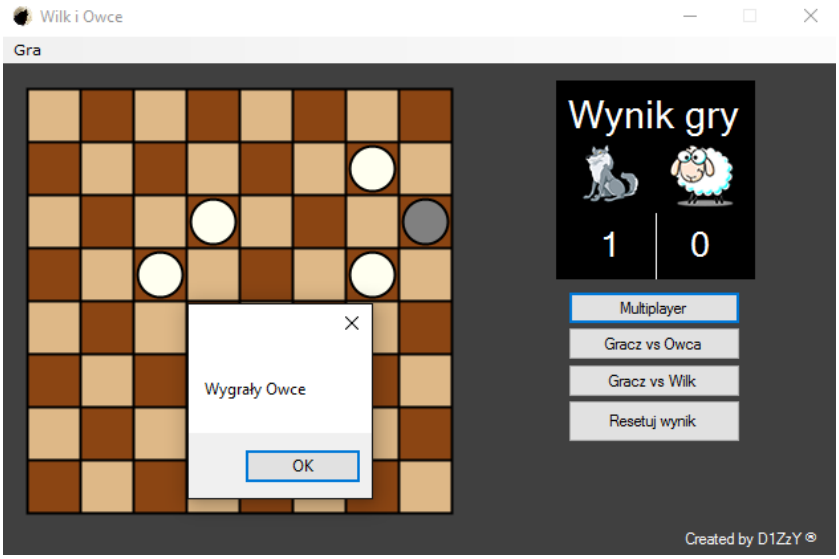
Aby rozpocząć grę wystarczy kliknąć myszką na przycisk „Multiplayer”, bądź rozwinąć pasek menu „Gra” i wybrać pozycję „Nowa gra”. Po kliknięciu zobaczymy następujące rozstawienie pionów:



Gra jest bardzo prosta w obsłudze, wystarczy kliknąć na wybrany pionek, którym chcemy poruszyć, a zostanie on podświetlony na żółto i zielony kolor pól wyznaczy nam możliwe ruchy. W grze Wilk i owce zawsze zaczyna gracz będący wilkiem.



Gracz jest informowany o wygranej jednej ze stron po spełnieniu warunków logicznych odpowiednich dla logiki gry, którą jest gra „Wilk i owce”.



Podczas gry Gracz vs Owca lub Gracz vs Wilk, gracz musi wybrać odpowiedni tryb z głównego menu,

Pozostała część interfejsu została dokładnie opisane w rozdziale „3.3. Opis poszczególnych komponentów graficznych”.

6. Zawartość dysku CD

Dysk CD zawiera cyfrową wersję dokumentacji o nazwie „manual.docx” i „manual.pdf”, program w wersji Release „Wilk i Owce.exe” oraz kod źródłowy programu w folderze „PI_Game”.

Kod źródłowy programu można otworzyć dzięki środowisku Visual Studio 2013 bądź nowszym.

Zawartość folder CD-ROM:\

Nazwa	Data modyfikacji	Typ	Rozmiar
PI_Game	2016-06-18 15:24	Folder plików	
manual	2016-07-03 16:54	Dokument progra...	820 KB
manual	2016-06-11 19:48	Dokument progra...	869 KB
PI_Game	2016-07-03 17:32	Archiwum WinRA...	2 082 KB
Wilk i Owce	2016-07-03 17:31	Aplikacja	325 KB

Zawartość folder CD-ROM:\PI_Game

Nazwa	Typ	Rozmiar
.git	Folder plików	
.vs	Folder plików	
PI_Game	Folder plików	
	Plik GITATTRIBUTES	3 KB
	Plik GITIGNORE	4 KB
PI_Game	Microsoft Visual S...	1 KB
PI_Game.v12	Visual Studio Solu...	110 KB
PI_Game.VC	Data Base File	32 320 KB

Zawartość folder CD-ROM:\PI_Game\PI_Game

Nazwa	Typ	Rozmiar
bin	Folder plików	
obj	Folder plików	
Properties	Folder plików	
AI.cs	Visual C# Source file	31 KB
App	XML Configuration File	1 KB
ClassDiagram1.cd	Class Diagram file	3 KB
GameWindow.cs	Visual C# Source file	9 KB
GameWindow.Designer.cs	Visual C# Source file	16 KB
GameWindow.resx	VisualStudio.resx.10.0	333 KB
Gracz.cs	Visual C# Source file	1 KB
ikona	Ikona	67 KB
Konfiguracja.cs	Visual C# Source file	1 KB
Owca	Plik PNG	7 KB
PI_Game	Visual C# Project file	6 KB
PI_Game.csproj.user	VisualStudio.user.10.0	1 KB
Plansza.cs	Visual C# Source file	5 KB
Program.cs	Visual C# Source file	1 KB
Ruchy.cs	Visual C# Source file	3 KB
score	Plik PNG	4 KB
Tablica.cs	Visual C# Source file	1 KB
UKSW_L	Plik PNG	154 KB
UKSW_Logo	Plik PNG	223 KB
Wilk	Plik PNG	7 KB

7. Bibliografia

Źródła internetowe:

1. Wikipedia, https://pl.wikipedia.org/wiki/Wilk_i_owce, 8.02.2015
2. Visual Studio Premium 2013 https://www.visualstudio.com/en-us/news/vs2013-community-vs.aspx
3. MSDN: Learn to Develop with Microsoft Developer Network https://msdn.microsoft.com/pl-pl/
4. Programowanie w języku C# http://4programmers.net/C_sharp 09.28.2013

OŚWIADCZENIE

Oświadczam, że złożona przeze mnie praca „Wilk i owce” jest moim samodzielnym opracowaniem. Oznacza to, że nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisałem tej rozprawy lub jej części z prac innych osób. Wyrażam zgodę na udostępnienie mojej pracy o wyżej wymienionym tytule. Oświadczam, że treść złożonej przeze mnie pracy o wyżej wymienionym tytule w wersji papierowej i elektronicznej jest tożsama.

……