Programowanie Obiektowe i Graficzne

dokumentacja projektu Gambit

Kacper Niemczynowski, grupa 3F

13 lipca 2021

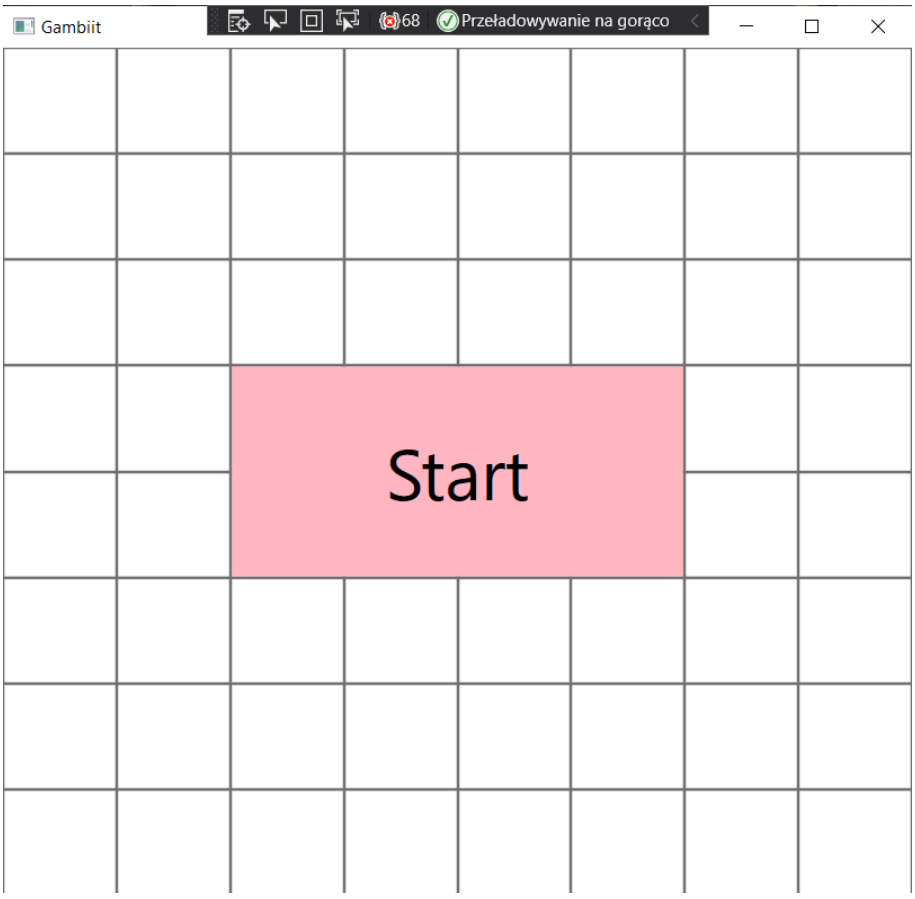
Część I

Opis programu

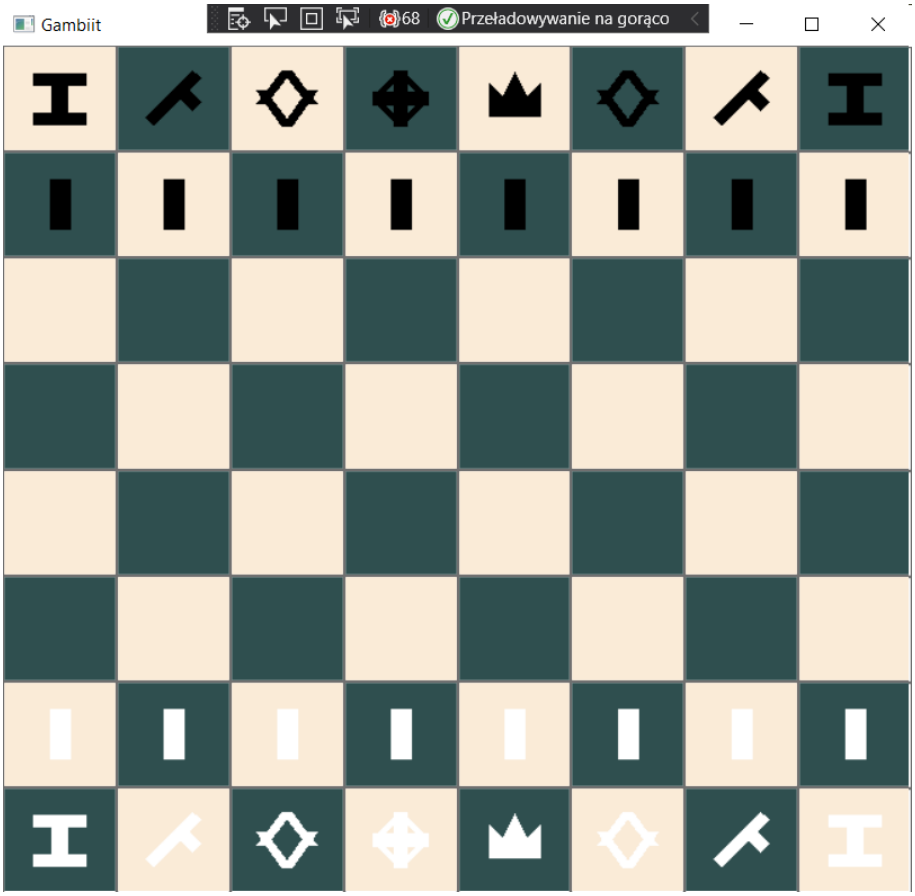
Szachy to strategiczna gra planszowa, która według źródeł pisanych znana była już w Persji w latach 70. Za jej kolebkę natomiast uznawne są Indie. Rozgrywka polega na starciu dwóch graczy - drużyny białej oraz czarnej. Wojska każdej drużyny składają się z 16 bierek - 8 pionków, których wartość szacuje się na 1 punkt, 2 wież z wartością 5 punktów każda, 2 gońców oraz 2 skoczków, których wartość równa jest 3 punktom, hetmana z wartością szacowaną na 9/10 punktów, oraz króla, któremu ciężko przypisać wartość, ponieważ jego strata kończy grę. Jednak jego siłę szacuje się na 4/5 punktów. Każda figura posiada określone możliwości ruchu/ataku, a rozgrywka opiera się na rozmieszczeniu swoich bierek tak, aby król przeciwnika jednocześnie był na polu zagrożonym atakiem (szach), oraz nie miał możliwości ruchu, bądź zasłonięcia się inną bierką (szach-mat). Szachy to gra turowa prowadzona na planszy w kształcie szachownicy 8x8, którą rozpoczyna gracz z bierkami koloru białego. Na jedną turę przypada jeden ruch bierki.

Instrukcja obsługi

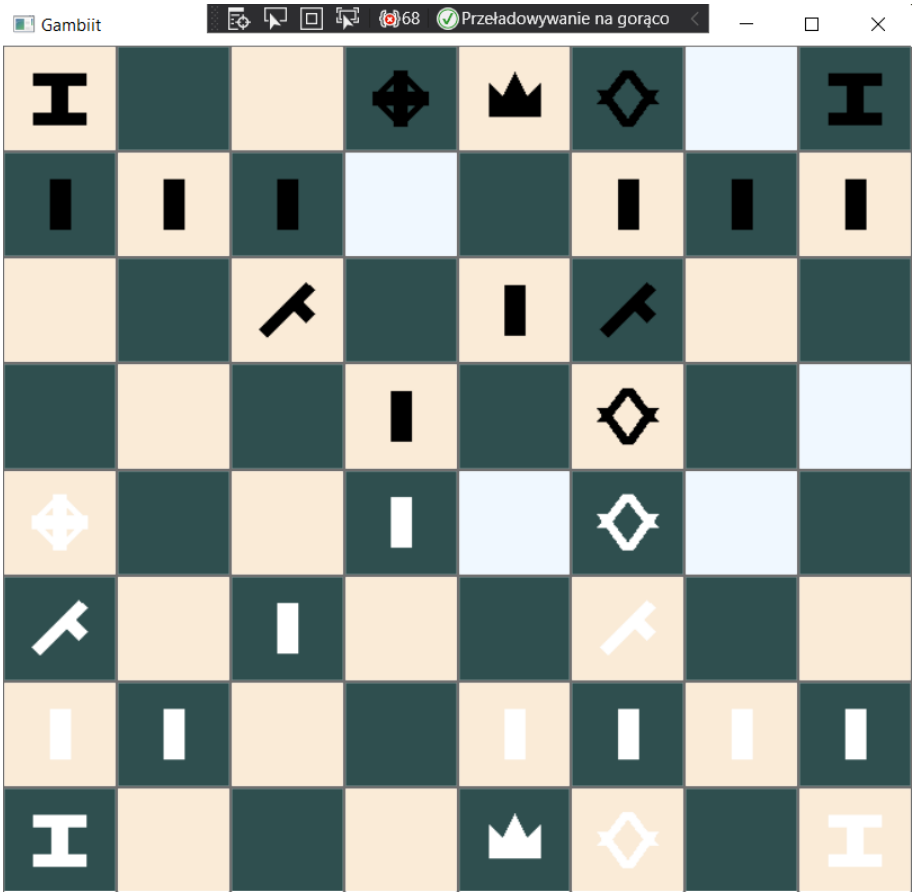
Do sterowania rozgrywką zaadaptowaną na program komputerowy potrzebna jest jedynie myszka oraz oczywiście komputer wraz z ekranem.
Okno rozgrywki prezentuje się następująco:



Gra rozpoczyna się ułożeniem figur na szachownicy:



Kliknięcie w swojej turze na bierkę ze swojego arsenału podświetli jej możliwości ruchu:



Jeśli uda nam się zamatować króla przeciwnika otrzymamy komunikat o zakończeniu rozgrywki:

Część II

Opis działania

Program stworzony jest z zachowaniem idei mvvm, co oznacza, że podzielony jest na model, view model oraz model. Główna część - model, składa się z 3 plików klas - Silnik, Armia i Pion. W klasie piona zawarte są bazowe właściwości figur - nazwa, pozycja, kolor oraz możliwośći ruchu/ataku. W klasie Armia tworzone są instancje figur, przechowywane są pozycje każdego z pionów armii oraz zaimplementowane są funkcje opowiedzialne za zebranie możliwych ruchów każdego z pionów, wykonanie owych ruchów, ściągnięcie z planszy piona zbitego, wymianę pionka, który doszedł do końca planszy (promocję), oraz obsługę roszady. Klasa Silnik natomiast, odpowiada za obsługę rozgrywki. Posiada dwuwymiarową tablicę odwzorowującą aktualną szachownicę na której na odpowienich pozycjach ustawione są figury. Naturalnie posiada również szereg funkcji nadzorujących poprawne poruszanie się bierek, ewentualne ograniczenie ruchów w przypadku wystąpienia szacha, zakończenie rozgrywki w przypadku zamatowania, przypisania odpowiednich ścieżek do plików graficznych ikon pionków oraz odpowiednich kolorów występujących na szachownicy i konwersję planszy do tablicy jednowymiarowej ułatwiającej prezentację szachownicy we view model'u.
Strona graficzna zgodnie z modelem mvvm posiada interfejs INotifyPropertyChanged odpowiadający za aktualizację zmieniających się dependency property powiązanych z kodem zawartym w części 'view' programu. Natomiast za obsługę kliknięć odpowiada interfejs ICommand.

Implementacja

Opis, zasada i działanie programu ze względu na podział na pliki, nastepnie funkcje programu wraz ze szczegółowym opisem działania (np.: formie pseudokodu, czy odniesienia do równania) Program podzielony jest na 5 plików klas - "Silnik.cs", "Armia.cs", "Pion.cs", "ViewModel.cs" i "MainWindow.xaml.cs" oraz na plik "MainWindow.xaml". Pliki graficzne zawarte są w folderze "Figury" a ich rozszerzenie to .png.

W pliku "MainWindow.xaml.cs" zawarta jest tylko jedna instrukcja tworząca i wywołująca okno aplikacji, oraz inicjalizująca jego składniki:

```
1 public partial class MainWindow : Window
2 {
3     public MainWindow()
4     {
5         InitializeComponent();
6     }
7 }
```

W pliku "MainWindow.xaml" zawarte są informacje dotyczące okna oraz tworzone i rozplanowane są przyciski, na których opiera się rozgrywka wraz z ukrytymi przyciskami odpowiadającymi za obsługę promocji pionów oraz przyciskiem startu.

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="1*"/>
        .
        .
        .
        <RowDefinition Height="1*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*"/>
        .
        .
        .
        <ColumnDefinition Width="1*"/>
    </Grid.ColumnDefinitions>

    <Button Grid.Row="0" Grid.Column="0" FontSize="50" Background="{
       Binding Kolor[0],Mode=OneWay}" Command="{Binding Clicked}"
       CommandParameter="00"><Image Source="{Binding Board[0],Mode=
       OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
       Binding Foc,Mode=OneWay}" /></Button>
    .
    .
    .
    <Button Grid.Row="3" Grid.Column="2" Content="Start" FontSize="50"
       Grid.RowSpan="2" Grid.ColumnSpan="4" Background="LightPink"
       Command="{Binding Start}" Visibility="{Binding Visibility2,Mode=
       OneWay}"/>

    <Border BorderBrush="Black" BorderThickness="5" Grid.Row="3" Grid.
       Column="2" Grid.RowSpan="1" Grid.ColumnSpan="4" Visibility="{
       Binding Visibility,Mode=OneWay}">
        <Grid Grid.Row="3" Grid.Column="2" Grid.RowSpan="1" Grid.
           ColumnSpan="4" Background="LightPink" >

            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="1*"/>
                <ColumnDefinition Width="1*"/>
                <ColumnDefinition Width="1*"/>
                <ColumnDefinition Width="1*"/>
            </Grid.ColumnDefinitions>

            <Button Grid.Column="0" Background="LightGreen" Command="{
               Binding Wybor}" CommandParameter="1">
                <Image Source="{Binding Choice[0],Mode=OneWay}" Stretch=
                   "None" Height="50" Width="50" />
            </Button>
            <Button Grid.Column="1" Background="LightGreen" Command="{
               Binding Wybor}" CommandParameter="2">
                <Image Source="{Binding Choice[1],Mode=OneWay}" Stretch=
                   "None" Height="50" Width="50" />
```

```
38              </Button>
39              <Button Grid.Column="2" Background="LightGreen" Command="{
                    Binding Wybor}" CommandParameter="3">
40                  <Image Source="{Binding Choice[2],Mode=OneWay}" Stretch=
                        "None" Height="50" Width="50" />
41              </Button>
42              <Button Grid.Column="3" Background="LightGreen" Command="{
                    Binding Wybor}" CommandParameter="4">
43                  <Image Source="{Binding Choice[3],Mode=OneWay}" Stretch=
                        "None" Height="50" Width="50" />
44              </Button>
45
46
47          </Grid>
48      </Border>
49
50  </Grid>
```

Plik "ViewModel.cs" tworzy obiekt klasy Silnik oraz tworzy dependency property:

```
1  public event PropertyChangedEventHandler PropertyChanged;
2  Model.Silnik silnik = new Model.Silnik();
3
4
5  private string[] board;
6  public string[] Board
7  {
8      get { return board; }
9      private set
10     {
11         board = value;
12
13         PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(
               nameof(Board)));
14
15     }
16 }
17
18 private string[] choice = new string[4];
19 public string[] Choice
20 {
21     get { return choice; }
22     private set
23     {
24         choice = value;
25         PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(
               nameof(Choice)));
26
27     }
28 }
29
30 private bool foc;
31 public bool Foc
32 {
33     get { return foc; }
```

```
34      private set
35      {
36
37          foc = value;
38          PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(
                nameof(Foc)));
39
40      }
41  }
42
43  private Visibility visibility = Visibility.Hidden;
44  public Visibility Visibility
45  {
46      get
47      {
48          return visibility;
49      }
50      set
51      {
52          visibility = value;
53          PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(
                nameof(Visibility)));
54      }
55  }
56
57  private Visibility visibility2;
58  public Visibility Visibility2
59  {
60      get
61      {
62          return visibility2;
63      }
64      set
65      {
66          visibility2 = value;
67          PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(
                nameof(Visibility2)));
68      }
69  }
70
71  private string[] kolor = new string[64];
72  public string[] Kolor
73  {
74      get { return kolor; }
75      private set
76      {
77          kolor = value;
78
79          PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(
                nameof(Kolor)));
80
81      }
82  }
```

Oraz funkcje obsługujące kliknięcia:

```
private ICommand start;
public ICommand Start
{
    get
    {
        return start ?? (start = new Totolotek.ViewModel.BaseClass.
            RelayCommand(Strt, null));
    }
}
public void Strt(object param)
{
    Foc = true;
    Visibility2 = Visibility.Hidden;
    Kolor = silnik.bazowykolor();
    Board = silnik.konwersja();
}

private ICommand clicked;
public ICommand Clicked
{
    get
    {
        return clicked ?? (clicked = new Totolotek.ViewModel.BaseClass.
            RelayCommand(Clck, null));
    }
}

public void Clck(object param)
{
    var tmp = (string)param;
    int x = chartonumber(Convert.ToSByte(tmp[0]));
    int y = chartonumber(Convert.ToSByte(tmp[1]));

    if (silnik.hmmm(x, y) != null)
    {
        Kolor = silnik.hmmm(x, y);
    }
    else
        Kolor = silnik.bazowykolor();

    Board = silnik.move(x, y);
    Board = silnik.attack(x, y);

    for(int i = 0; i < 8; i++)
    {
        if(Board[i] == "/ViewModel/Figury/bp.png")
        {
            Foc = false;
            Visibility = Visibility.Visible;
            Choice = silnik.rev(x,y);
        }
    }
    for (int i = 56; i < 64; i++)
    {
        if (Board[i] == "/ViewModel/Figury/cp.png")
```

```
54          {
55              Foc = false;
56              Visibility = Visibility.Visible;
57              Choice = silnik.rev(x, y);
58          }
59      }
60
61      if (silnik.checkmate())
62      {
63          //Console.WriteLine("WIN");
64          Foc = false;
65          MessageBox.Show("Szach-mat", "Partia zako[U+FFFD]czona",
                MessageBoxButton.OK, MessageBoxImage.Information);
66      }
67  }
68
69  private ICommand wybor;
70  public ICommand Wybor
71  {
72      get
73      {
74          return wybor ?? (wybor = new Totolotek.ViewModel.BaseClass.
                RelayCommand(Wyb, null));
75      }
76  }
77
78  public void Wyb(object param)
79  {
80      var tmp = (string)param;
81      int x = chartonumber(Convert.ToSByte(tmp[0]));
82      Board =  silnik.rivia(x);
83      Visibility = Visibility.Hidden;
84      Foc = true;
85  }
```

Klasa Pion zawiera 4 atrybuty, prosty konstruktor:

```
1  public int X { get; set; }
2  public int Y { get; set; }
3  public char Who { get; set; }
4  public bool Kolor { get; set; }
5
6  public Pion(int x, int y, char who, bool kolor)
7  {
8      X = x;
9      Y = y;
10     Who = who;
11     Kolor = kolor;
12 }
```

oraz metodę definiującą ruchy figur do której na wejściu przesyłana jest tablica zawarta w Klasie Silnik oraz zmienna boolowska odpowiadająca za wybór wartości zwracanej - listy 'maybe', która zawiera listę pozycji ruchów, które są zakłócone przez inne figury, lub listy 'possible', w której ruchy figury są niezakłócone:

```
public int[,] mozliwosci(char[,] plansza, bool pm)
{
    List<int> possible = new List<int>();
    List<int> maybe = new List<int>();

    switch (Who)
    {
        case 'w':
            for (int i = X + 1; i < 8; i++)
            {
                if (plansza[i, Y] == '0')
                {
                    possible.Add(i);
                    possible.Add(Y);
                }

                if (plansza[i, Y] != '0')
                {
                    maybe.Add(i);
                    maybe.Add(Y);
                    break;
                }
            }

            for (int i = X - 1; i >= 0; i--)
            {
                if (plansza[i, Y] == '0')
                {
                    possible.Add(i);
                    possible.Add(Y);
                }

                if (plansza[i, Y] != '0')
                {
                    maybe.Add(i);
                    maybe.Add(Y);
                    break;
                }
            }

            for (int i = Y + 1; i < 8; i++)
            {
                if (plansza[X, i] == '0')
                {
                    possible.Add(X);
                    possible.Add(i);
                }

                if (plansza[X, i] != '0')
                {
                    maybe.Add(X);
                    maybe.Add(i);
                    break;
                }
            }
```

```
56
57              for (int i = Y - 1; i >= 0; i--)
58              {
59                  if (plansza[X, i] == '0')
60                  {
61                      possible.Add(X);
62                      possible.Add(i);
63                  }
64
65                  if (plansza[X, i] != '0')
66                  {
67                      maybe.Add(X);
68                      maybe.Add(i);
69                      break;
70                  }
71              }
72
73              break;
74
75          case 'g':
76              int j = Y + 1;
77              for (int i = X + 1; i < 8; i++)
78              {
79                  if (j >= 8)
80                      break;
81                  if (plansza[i, j] == '0')
82                  {
83                      possible.Add(i);
84                      possible.Add(j);
85                  }
86
87                  if (plansza[i, j] != '0')
88                  {
89                      maybe.Add(i);
90                      maybe.Add(j);
91                      break;
92                  }
93
94                  j++;
95                  if (j >= 8)
96                      break;
97              }
98
99              j = Y - 1;
100             for (int i = X + 1; i < 8; i++)
101             {
102                 if (j < 0)
103                     break;
104                 if (plansza[i, j] == '0')
105                 {
106                     possible.Add(i);
107                     possible.Add(j);
108                 }
109
110                 if (plansza[i, j] != '0')
```

```
111                 {
112                     maybe.Add(i);
113                     maybe.Add(j);
114                     break;
115                 }
116                 j--;
117                 if (j < 0)
118                     break;
119             }
120 
121             j = Y - 1;
122             for (int i = X - 1; i >= 0; i--)
123             {
124                 if (j < 0)
125                     break;
126                 if (plansza[i, j] == '0')
127                 {
128                     possible.Add(i);
129                     possible.Add(j);
130                 }
131 
132                 if (plansza[i, j] != '0')
133                 {
134                     maybe.Add(i);
135                     maybe.Add(j);
136                     break;
137                 }
138                 j--;
139                 if (j < 0)
140                     break;
141             }
142 
143             j = Y + 1;
144             for (int i = X - 1; i >= 0; i--)
145             {
146                 if (j >= 8)
147                     break;
148                 if (plansza[i, j] == '0')
149                 {
150                     possible.Add(i);
151                     possible.Add(j);
152                 }
153 
154                 if (plansza[i, j] != '0')
155                 {
156                     maybe.Add(i);
157                     maybe.Add(j);
158                     break;
159                 }
160                 j++;
161                 if (j >= 8)
162                     break;
163             }
164 
165             break;
```

```
166
167         case 'h':
168             for (int i = X + 1; i < 8; i++)
169             {
170                 if (plansza[i, Y] == '0')
171                 {
172                     possible.Add(i);
173                     possible.Add(Y);
174                 }
175
176                 if (plansza[i, Y] != '0')
177                 {
178                     maybe.Add(i);
179                     maybe.Add(Y);
180                     break;
181                 }
182             }
183
184             for (int i = X - 1; i >= 0; i--)
185             {
186                 if (plansza[i, Y] == '0')
187                 {
188                     possible.Add(i);
189                     possible.Add(Y);
190                 }
191
192                 if (plansza[i, Y] != '0')
193                 {
194                     maybe.Add(i);
195                     maybe.Add(Y);
196                     break;
197                 }
198             }
199
200             for (int i = Y + 1; i < 8; i++)
201             {
202                 if (plansza[X, i] == '0')
203                 {
204                     possible.Add(X);
205                     possible.Add(i);
206                 }
207
208                 if (plansza[X, i] != '0')
209                 {
210                     maybe.Add(X);
211                     maybe.Add(i);
212                     break;
213                 }
214             }
215
216             for (int i = Y - 1; i >= 0; i--)
217             {
218                 if (plansza[X, i] == '0')
219                 {
220                     possible.Add(X);
```

```
221                     possible.Add(i);
222                 }
223 
224                 if (plansza[X, i] != '0')
225                 {
226                     maybe.Add(X);
227                     maybe.Add(i);
228                     break;
229                 }
230             }
231 
232 
233             int jj = Y + 1;
234             for (int i = X + 1; i < 8; i++)
235             {
236                 if (jj >= 8)
237                     break;
238                 if (plansza[i, jj] == '0')
239                 {
240                     possible.Add(i);
241                     possible.Add(jj);
242                 }
243 
244                 if (plansza[i, jj] != '0')
245                 {
246                     maybe.Add(i);
247                     maybe.Add(jj);
248                     break;
249                 }
250                 jj++;
251                 if (jj >= 8)
252                     break;
253             }
254 
255             jj = Y - 1;
256             for (int i = X + 1; i < 8; i++)
257             {
258                 if (jj < 0)
259                     break;
260                 if (plansza[i, jj] == '0')
261                 {
262                     possible.Add(i);
263                     possible.Add(jj);
264                 }
265 
266                 if (plansza[i, jj] != '0')
267                 {
268                     maybe.Add(i);
269                     maybe.Add(jj);
270                     break;
271                 }
272                 jj--;
273                 if (jj < 0)
274                     break;
275             }
```

```
276
277             jj = Y - 1;
278             for (int i = X - 1; i >= 0; i--)
279             {
280                 if (jj < 0)
281                     break;
282                 if (plansza[i, jj] == '0')
283                 {
284                     possible.Add(i);
285                     possible.Add(jj);
286                 }
287
288                 if (plansza[i, jj] != '0')
289                 {
290                     maybe.Add(i);
291                     maybe.Add(jj);
292                     break;
293                 }
294                 jj--;
295                 if (jj < 0)
296                     break;
297             }
298
299             jj = Y + 1;
300             for (int i = X - 1; i >= 0; i--)
301             {
302                 if (jj >= 8)
303                     break;
304                 if (plansza[i, jj] == '0')
305                 {
306                     possible.Add(i);
307                     possible.Add(jj);
308                 }
309
310                 if (plansza[i, jj] != '0')
311                 {
312                     maybe.Add(i);
313                     maybe.Add(jj);
314                     break;
315                 }
316                 jj++;
317                 if (jj >= 8)
318                     break;
319             }
320
321             break;
322
323         case 's':
324             if (X + 2 < 8 && Y + 1 < 8 && plansza[X + 2, Y + 1] == '0')
325             {
326                 possible.Add(X + 2);
327                 possible.Add(Y + 1);
328             }
329             if (X + 2 < 8 && Y + 1 < 8 && plansza[X + 2, Y + 1] != '0')
330             {
```

```
331                 maybe.Add(X + 2);
332                 maybe.Add(Y + 1);
333             }
334             if (X + 2 < 8 && Y - 1 >= 0 && plansza[X + 2, Y - 1] == '0')
335             {
336                 possible.Add(X + 2);
337                 possible.Add(Y - 1);
338             }
339             if (X + 2 < 8 && Y - 1 >= 0 && plansza[X + 2, Y - 1] != '0')
340             {
341                 maybe.Add(X + 2);
342                 maybe.Add(Y - 1);
343             }
344 
345             if (X - 2 >= 0 && Y + 1 < 8 && plansza[X - 2, Y + 1] == '0')
346             {
347                 possible.Add(X - 2);
348                 possible.Add(Y + 1);
349             }
350             if (X - 2 >= 0 && Y + 1 < 8 && plansza[X - 2, Y + 1] != '0')
351             {
352                 maybe.Add(X - 2);
353                 maybe.Add(Y + 1);
354             }
355             if (X - 2 >= 0 && Y - 1 >= 0 && plansza[X - 2, Y - 1] ==
                    '0')
356             {
357                 possible.Add(X - 2);
358                 possible.Add(Y - 1);
359             }
360             if (X - 2 >= 0 && Y - 1 >= 0 && plansza[X - 2, Y - 1] !=
                    '0')
361             {
362                 maybe.Add(X - 2);
363                 maybe.Add(Y - 1);
364             }
365 
366             if (X + 1 < 8 && Y + 2 < 8 && plansza[X + 1, Y + 2] == '0')
367             {
368                 possible.Add(X + 1);
369                 possible.Add(Y + 2);
370             }
371             if (X + 1 < 8 && Y + 2 < 8 && plansza[X + 1, Y + 2] != '0')
372             {
373                 maybe.Add(X + 1);
374                 maybe.Add(Y + 2);
375             }
376             if (X - 1 >= 0 && Y + 2 < 8 && plansza[X - 1, Y + 2] == '0')
377             {
378                 possible.Add(X - 1);
379                 possible.Add(Y + 2);
380             }
381             if (X - 1 >= 0 && Y + 2 < 8 && plansza[X - 1, Y + 2] != '0')
382             {
383                 maybe.Add(X - 1);
```

```
384                 maybe.Add(Y + 2);
385             }
386
387
388             if (X - 1 >= 0 && Y - 2 >= 0 && plansza[X - 1, Y - 2] ==
                    '0')
389             {
390                 possible.Add(X - 1);
391                 possible.Add(Y - 2);
392             }
393             if (X - 1 >= 0 && Y - 2 >= 0 && plansza[X - 1, Y - 2] !=
                    '0')
394             {
395                 maybe.Add(X - 1);
396                 maybe.Add(Y - 2);
397             }
398             if (X + 1 < 8 && Y - 2 >= 0 && plansza[X + 1, Y - 2] == '0')
399             {
400                 possible.Add(X + 1);
401                 possible.Add(Y - 2);
402             }
403             if (X + 1 < 8 && Y - 2 >= 0 && plansza[X + 1, Y - 2] != '0')
404             {
405                 maybe.Add(X + 1);
406                 maybe.Add(Y - 2);
407             }
408
409             break;
410
411         case 'k':
412             for (int i = X - 1; i < X + 2; i++)
413             {
414                 for (int l = Y - 1; l < Y + 2; l++)
415                 {
416                     if (i >= 0 && i < 8 && l >= 0 && l < 8)
417                     {
418                         if (plansza[i, l] == '0')
419                         {
420                             possible.Add(i);
421                             possible.Add(l);
422                         }
423                         if (plansza[i, l] != '0')
424                         {
425                             maybe.Add(i);
426                             maybe.Add(l);
427                         }
428                     }
429                 }
430             }
431
432             break;
433
434         case 'p':
435             if (Kolor == true)
436             {
```

```
437                 if (X + 1 < 8 && plansza[X + 1, Y] == '0')
438                 {
439                     possible.Add(X + 1);
440                     possible.Add(Y);
441 
442                     if (X == 1 && plansza[X + 2, Y] == '0')
443                     {
444                         possible.Add(X + 2);
445                         possible.Add(Y);
446                     }
447                 }
448                 if (Y + 1 < 8 && X + 1 < 8 && plansza[X + 1, Y + 1] !=
                        '0')
449                 {
450                     maybe.Add(X + 1);
451                     maybe.Add(Y + 1);
452                 }
453                 if (Y - 1 >= 0 && X + 1 < 8 && plansza[X + 1, Y - 1] !=
                        '0')
454                 {
455                     maybe.Add(X + 1);
456                     maybe.Add(Y - 1);
457                 }
458             }
459 
460             if (Kolor == false)
461             {
462                 if (X - 1 >= 0 && plansza[X - 1, Y] == '0')
463                 {
464                     possible.Add(X - 1);
465                     possible.Add(Y);
466 
467                     if (X == 6 && plansza[X - 2, Y] == '0')
468                     {
469                         possible.Add(X - 2);
470                         possible.Add(Y);
471                     }
472                 }
473                 if (Y + 1 < 8 && X - 1 >= 0 && plansza[X - 1, Y + 1] !=
                        '0')
474                 {
475                     maybe.Add(X - 1);
476                     maybe.Add(Y + 1);
477                 }
478                 if (Y - 1 >= 0 && X - 1 >= 0 && plansza[X - 1, Y - 1] !=
                         '0')
479                 {
480                     maybe.Add(X - 1);
481                     maybe.Add(Y - 1);
482                 }
483             }
484             break;
485         case '0':
486             possible.Clear();
487             maybe.Clear();
```

```
488                 break;
489         }
490 
491         int[,] wyjazd;
492         if (!pm)
493             wyjazd = listto2darray(possible);
494         else
495             wyjazd = listto2darray(maybe);
496 
497         return wyjazd;
498     }
```

Klasa Armia posiada listę obiektów klasy pionek definiowaną w konstruktorze, dwuwymiarową tablicę rozkładu swoich bierek, oraz zawiera informacje ułatwiające pracę funkcji promującej pion i obsługującej roszadę:

```
1  public char[,] rozklad;
2  List<Pion> pionki = new List<Pion>();
3  List<Pion> shadowrealm = new List<Pion>();
4  private int xdorev;
5  private int ydorev;
6  private bool[] roshada = new bool[3] { true, true, true };
7  
8  public Armia(bool x)
9  {
10     if (x)
11     {
12         rozklad = new char[8, 8] {  {'w','s','g','h','k','g','s','w' },
13                                     {'p','p','p','p','p','p','p','p' },
14                                     {'0','0','0','0','0','0','0','0' },
15                                     {'0','0','0','0','0','0','0','0' },
16                                     {'0','0','0','0','0','0','0','0' },
17                                     {'0','0','0','0','0','0','0','0' },
18                                     {'0','0','0','0','0','0','0','0' },
19                                     {'0','0','0','0','0','0','0','0' }
                                           };
20 
21         for (int i = 0; i < 8; i++)
22         {
23             for (int j = 0; j < 8; j++)
24             {
25                 if (rozklad[i, j] != '0')
26                     pionki.Add(new Pion(i, j, rozklad[i, j], true));
27             }
28         }
29     }
30     else
31     {
32         rozklad = new char[8, 8] {  {'0','0','0','0','0','0','0','0' },
33                                     {'0','0','0','0','0','0','0','0' },
34                                     {'0','0','0','0','0','0','0','0' },
35                                     {'0','0','0','0','0','0','0','0' },
36                                     {'0','0','0','0','0','0','0','0' },
37                                     {'0','0','0','0','0','0','0','0' },
38                                     {'p','p','p','p','p','p','p','p' },
```

```
                                        {'w','s','g','h','k','g','s','w' }
                                    };

        for (int i = 0; i < 8; i++)
        {
            for (int j = 0; j < 8; j++)
            {
                if (rozklad[i, j] != '0')
                    pionki.Add(new Pion(i, j, rozklad[i, j], false));
            }
        }
    }
}
```

Do funkcji TheChosenOne na wejściu przesyłana jest pozycja wybranej bierki, główna plansza z Klasy Silnik oraz zmienna boolowska działająca na podobnej zasadzie co w funkcji 'możliwości' z klasy Pion - jeśli wybierzemy true dosatniemy tablicę pozycji odpowiedzialnych za atak, jeśli natomiast wybierzemy wartość false dostaniemy tablicę ruchów niezakłóconych:

```
public int[,] TheChosenOne(int xx, int yy, char[,] plansza, bool aom)
{
    for (int i = 0; i < pionki.Count; i++)
    {
        if (pionki[i].X == xx && pionki[i].Y == yy)
        {
            if (aom)
            {
                int[,] mb = pionki[i].mozliwosci(plansza, true);
                if (mb != null)
                {
                    List<int> attck = new List<int>();
                    for (int k = 0; k < mb.Length / 2; k++)
                    {
                        if (rozklad[mb[k, 0], mb[k, 1]] == '0')
                        {
                            attck.Add(mb[k, 0]);
                            attck.Add(mb[k, 1]);
                        }
                    }
                    int[,] attack = new int[attck.Count / 2, 2];
                    attack = pionki[i].listto2darray(attck);
                    return attack;
                }
                return null;
            }
            return pionki[i].mozliwosci(plansza, false);
        }
    }
    return null;
}
```

Funkcja odpowiedzialna za ruch pionka:

```
public char armymove(int x, int y, int lastx, int lasty)
{
```

```
3       for (int i = 0; i < pionki.Count; i++)
4       {
5           if (pionki[i].X == lastx && pionki[i].Y == lasty)
6           {
7               pionki[i].move(x, y);
8               checkrosh(lastx, lasty, pionki[i].Who);
9               rozklad[lastx, lasty] = '0';
10              rozklad[x, y] = pionki[i].Who;
11              return pionki[i].Who;
12          }
13      }
14      return 'x';
15  }
```

Funkcja odpowiedzialna za informacje o ruchu wież i króli, które definiują możliwość wykonania roszady:

```
1  public void checkrosh(int x, int y, char kto)
2  {
3      if ((x == 7 && y == 7 || x == 0 && y == 7) && kto == 'w')
4      {
5          roshada[2] = false;
6      }
7      if ((x == 7 && y == 0 || x == 0 && y == 0) && kto == 'w')
8      {
9          roshada[0] = false;
10     }
11     if ((x == 7 && y == 4 || x == 0 && y == 4) && kto == 'k')
12     {
13         roshada[1] = false;
14     }
15 }
```

Funkcja odpowiedzialna za wykonanie roszady:

```
1  public int[,] roszada(bool sprawdzam, bool lewa, int x, int y)
2  {
3      if (sprawdzam)
4      {
5          if (rozklad[x, y] != 'k')
6              return null;
7          List<int> mb = new List<int>();
8          if (roshada[0] == true && roshada[1] == true)
9          {
10             if (rozklad[0, 0] == 'w' && rozklad[0, 4] == 'k' && rozklad
                   [0, 1] == '0' && rozklad[0, 2] == '0' && rozklad[0, 3] ==
                    '0')
11             {
12                 mb.Add(0);
13                 mb.Add(2);
14             }
15             if (rozklad[7, 4] == 'k' && rozklad[7, 0] == 'w' && rozklad
                   [7, 1] == '0' && rozklad[7, 2] == '0' && rozklad[7, 3] ==
                    '0')
16             {
```

```
17                  mb.Add(7);
18                  mb.Add(2);
19              }
20          }
21          if (roshada[1] == true && roshada[2] == true)
22          {
23              if (rozklad[0, 7] == 'w' && rozklad[0, 4] == 'k' && rozklad
                    [0, 6] == '0' && rozklad[0, 5] == '0')
24              {
25                  mb.Add(0);
26                  mb.Add(6);
27              }
28              if (rozklad[7, 4] == 'k' && rozklad[7, 7] == 'w' && rozklad
                    [7, 6] == '0' && rozklad[7, 5] == '0')
29              {
30                  mb.Add(7);
31                  mb.Add(6);
32              }
33          }
34          int[,] tmp = listto2darray(mb);
35          return tmp;
36      }
37      else
38      {
39          if (lewa)
40          {
41              if (rozklad[0, 0] == 'w' && rozklad[0, 4] == 'k')
42              {
43                  rozklad[0, 0] = '0';
44                  rozklad[0, 4] = '0';
45                  rozklad[0, 2] = 'k';
46                  rozklad[0, 3] = 'w';
47                  for (int i = 0; i < pionki.Count; i++)
48                  {
49                      if (pionki[i].Y == 4 && pionki[i].X == 0)
50                          pionki[i].Y = 2;
51
52                      if (pionki[i].Y == 0 && pionki[i].X == 0)
53                          pionki[i].Y = 3;
54                  }
55              }
56              if (rozklad[7, 4] == 'k' && rozklad[7, 0] == 'w')
57              {
58                  rozklad[7, 0] = '0';
59                  rozklad[7, 4] = '0';
60                  rozklad[7, 2] = 'k';
61                  rozklad[7, 3] = 'w';
62                  for (int i = 0; i < pionki.Count; i++)
63                  {
64                      if (pionki[i].Y == 4 && pionki[i].X == 7)
65                          pionki[i].Y = 2;
66
67                      if (pionki[i].Y == 0 && pionki[i].X == 7)
68                          pionki[i].Y = 3;
69                  }
```

```
70              }
71          }
72          else
73          {
74              if (rozklad[0, 7] == 'w' && rozklad[0, 4] == 'k')
75              {
76                  rozklad[0, 4] = '0';
77                  rozklad[0, 7] = '0';
78                  rozklad[0, 6] = 'k';
79                  rozklad[0, 5] = 'w';
80                  for (int i = 0; i < pionki.Count; i++)
81                  {
82                      if (pionki[i].Y == 4 && pionki[i].X == 0)
83                          pionki[i].Y = 6;
84
85                      if (pionki[i].Y == 7 && pionki[i].X == 0)
86                          pionki[i].Y = 5;
87                  }
88              }
89              if (rozklad[7, 4] == 'k' && rozklad[7, 7] == 'w')
90              {
91                  rozklad[7, 4] = '0';
92                  rozklad[7, 7] = '0';
93                  rozklad[7, 6] = 'k';
94                  rozklad[7, 5] = 'w';
95                  for (int i = 0; i < pionki.Count; i++)
96                  {
97                      if (pionki[i].Y == 4 && pionki[i].X == 7)
98                          pionki[i].Y = 6;
99
100                     if (pionki[i].Y == 7 && pionki[i].X == 7)
101                         pionki[i].Y = 5;
102                 }
103             }
104         }
105     }
106     return null;
107 }
```

Funkcja odpowiedzialna za zbicie figury:

```
1  public void death(int x, int y)
2  {
3      for (int i = 0; i < pionki.Count; i++)
4      {
5          if (pionki[i].X == x && pionki[i].Y == y)
6          {
7              pionki[i].X = 9;
8              pionki[i].Y = 9;
9              shadowrealm.Add(pionki[i]);
10             pionki.RemoveAt(i);
11             rozklad[x, y] = '0';
12         }
13     }
14 }
```

Funkcje obsługujące promocję pionów:

```
public char[] whorevive(int x, int y)
{
    death(x, y);
    xdorev = x;
    ydorev = y;
    char[] wybor = new char[4] { 'w', 's', 'g', 'h'};
    return wybor;
}

public void revive(char kto, bool kolor)
{
    pionki.Add(new Pion(xdorev, ydorev, kto, kolor));
    rozklad[xdorev, ydorev] = kto;
}
```

Funkcje wspomagające obsługę ruchów pionów podczas szachu:

```
public int[,] mozliwosciarmii(char[,] plansza, int x, int y)
{
    List<int> attck = new List<int>();
    char[,] podmianka = new char[8, 8];
    for (int i = 0; i < 8; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            podmianka[i, j] = rozklad[i, j];
        }
    }
    if(x !=8 && y != 8)
    {
        podmianka[x, y] = '0';
    }

    for (int i = 0; i < pionki.Count; i++)
    {
        int[,] mb = pionki[i].mozliwosci(plansza, true);

        if (pionki[i].X == x && pionki[i].Y == y)
            mb = null;

        if (mb != null)
        {
            for (int k = 0; k < mb.Length / 2; k++)
            {
                    if (podmianka[mb[k, 0], mb[k, 1]] == '0')
                    {
                        attck.Add(mb[k, 0]);
                        attck.Add(mb[k, 1]);
                    }
            }
        }

    }
    int[,] attack = new int[attck.Count / 2, 2];
```

```
38      attack = listto2darray(attck);
39      return attack;
40  }
41
42
43  public int[,] zasobyarmii()
44  {
45      List<int> tmp = new List<int>();
46      for (int i = 0; i < pionki.Count; i++)
47      {
48          tmp.Add(pionki[i].X);
49          tmp.Add(pionki[i].Y);
50      }
51
52      int[,] wojsko = new int[tmp.Count / 2, 2];
53      wojsko = listto2darray(tmp);
54      return wojsko;
55  }
```

Klasa Silnik tworzy obiekty klasy Armia a jej atrybuty definiują turę odpowiedniego gracza, zawierają główną planszę, oraz ułatwiają obsługę ruchów figur i promocję pionów:

```
1  Armia Biale = new Armia(false);
2  Armia Czarne = new Armia(true);
3
4  bool ruch = true;
5  bool clicked = false;
6
7  int lastmoveX = 9;
8  int lastmoveY = 9;
9
10 int xdorev;
11 int ydorev;
12
13 char[,] plansza = new char[8, 8] {  {'w','s','g','h','k','g','s','w' },
14                                     {'p','p','p','p','p','p','p','p' },
15                                     {'0','0','0','0','0','0','0','0' },
16                                     {'0','0','0','0','0','0','0','0' },
17                                     {'0','0','0','0','0','0','0','0' },
18                                     {'0','0','0','0','0','0','0','0' },
19                                     {'p','p','p','p','p','p','p','p' },
20                                     {'w','s','g','h','k','g','s','w' }
                                        };
```

Metoda 'hmmm' służy do modelowania możliwych ruchów figur, które następnie podświetlają odpowiednie miejsca na szachownicy.

```
1  public string[] hmmm(int x, int y)
2  {
3
4      int[,] tmp1;
5      int[,] tmp2;
6      int[,] tmp3;
7
8
```

```
9       if (ruch)
10      {
11          tmp1 = Biale.TheChosenOne(x, y, plansza, false);
12          tmp2 = Biale.TheChosenOne(x, y, plansza, true);
13      }
14      else
15      {
16          tmp1 = Czarne.TheChosenOne(x, y, plansza, false);
17          tmp2 = Czarne.TheChosenOne(x, y, plansza, true);
18      }
19
20      tmp1 = checker(tmp1, x, y, plansza);
21      tmp2 = checker(tmp2, x, y, plansza);
22      tmp3 = roshadachecker(plansza, x, y);
23      if (checkcheck(plansza, 8, 8))
24          tmp3 = null;
25
26
27      if (tmp1 == null && tmp2 == null && tmp3 == null)
28      {
29          Console.WriteLine("hmm");
30          return null;
31      }
32
33      string[] hym;
34      hym = bazowykolor();
35
36
37      if (tmp1 != null)
38      {
39          int[] kolor1 = new int[tmp1.Length / 2];
40
41          for (int i = 0; i < kolor1.Length; i++)
42          {
43              kolor1[i] = 8 * tmp1[i, 0] + tmp1[i, 1];
44          }
45
46          for (int i = 0; i < kolor1.Length; i++)
47          {
48              hym[kolor1[i]] = "AliceBlue";
49          }
50      }
51      if (tmp2 != null)
52      {
53          int[] kolor2 = new int[tmp2.Length / 2];
54
55          for (int i = 0; i < kolor2.Length; i++)
56          {
57              kolor2[i] = 8 * tmp2[i, 0] + tmp2[i, 1];
58          }
59
60          for (int i = 0; i < kolor2.Length; i++)
61          {
62              hym[kolor2[i]] = "Red";
63          }
```

```
64      }
65      if (tmp3 != null)
66      {
67          int[] kolor3 = new int[tmp3.Length / 2];
68
69          for (int i = 0; i < kolor3.Length; i++)
70          {
71              kolor3[i] = 8 * tmp3[i, 0] + tmp3[i, 1];
72          }
73
74          for (int i = 0; i < kolor3.Length; i++)
75          {
76              hym[kolor3[i]] = "AliceBlue";
77          }
78      }
79
80      lastmove(x, y);
81      return hym;
82  }
```

---

funkcje 'move' oraz 'attack', są bardzo podobne jednak jedna działa na tablicy ruchów bezkonfliktowych a druga na tych z możliwością ataku.

```
1  public string[] move(int x, int y)
2  {
3      if (clicked)
4      {
5          int[,] tmp;
6          int[,] tmpr;
7          if (ruch)
8          {
9              tmp = Biale.TheChosenOne(lastmoveX, lastmoveY, plansza,
                   false);
10         }
11         else
12         {
13             tmp = Czarne.TheChosenOne(lastmoveX, lastmoveY, plansza,
                   false);
14         }
15
16         tmp = checker(tmp, lastmoveX, lastmoveY, plansza);
17         tmpr = roshadachecker(plansza, lastmoveX, lastmoveY);
18         if (checkcheck(plansza, 8, 8))
19             tmpr = null;
20
21         if (tmp == null && tmpr == null)
22             return konwersja();
23
24         if(tmpr != null)
25         {
26             for (int i = 0; i < tmpr.Length / 2; i++)
27             {
28                 if (tmpr[i, 0] == x && tmpr[i, 1] == y)
29                 {
30                     rosh(tmpr, x, y, i);
```

```
31                      return konwersja();
32                  }
33              }
34          }
35          if (tmp != null)
36          {
37              for (int i = 0; i < tmp.Length / 2; i++)
38              {
39                  if (tmp[i, 0] == x && tmp[i, 1] == y)
40                  {
41                      char who;
42                      if (ruch)
43                          who = Biale.armymove(x, y, lastmoveX, lastmoveY)
                                ;
44                      else
45                          who = Czarne.armymove(x, y, lastmoveX, lastmoveY
                                );
46
47                      if (who == 'x')
48                      {
49                          return konwersja();
50                      }
51
52
53                      plansza[lastmoveX, lastmoveY] = '0';
54                      plansza[x, y] = who;
55                      //checkcheck
56
57                      ruch = !ruch;
58                      clicked = false;
59                      if (checkmate())
60                      {
61                          Console.WriteLine("WIN");
62                      }
63                  }
64              }
65          }
66      }
67      return konwersja();
68  }
69
70
71  public string[] attack(int x, int y)
72  {
73      if (clicked)
74      {
75          int[,] tmp;
76          int[,] tmpr;
77
78          if (ruch)
79          {
80              tmp = Biale.TheChosenOne(lastmoveX, lastmoveY, plansza, true
                    );
81          }
82          else
```

```
83          {
84              tmp = Czarne.TheChosenOne(lastmoveX, lastmoveY, plansza,
                    true);
85          }
86
87          //tmp = checker(tmp, x, y, plansza);
88
89          tmpr = roshadachecker(plansza, lastmoveX, lastmoveY);
90          if (checkcheck(plansza, 8, 8))
91              tmpr = null;
92
93          if (tmp == null && tmpr == null)
94              return konwersja();
95
96          if (tmpr != null)
97          {
98              for (int i = 0; i < tmpr.Length / 2; i++)
99              {
100                 if (tmpr[i, 0] == x && tmpr[i, 1] == y)
101                 {
102                     rosh(tmpr, x, y, i);
103                     return konwersja();
104                 }
105             }
106         }
107         if (tmp != null)
108         {
109             for (int i = 0; i < tmp.Length / 2; i++)
110             {
111                 if (tmp[i, 0] == x && tmp[i, 1] == y)
112                 {
113                     ///
114                     char who;
115                     if (ruch)
116                     {
117                         who = Biale.armyattack(x, y, lastmoveX,
                                lastmoveY);
118                         Czarne.death(x, y);
119                     }
120                     else
121                     {
122                         who = Czarne.armyattack(x, y, lastmoveX,
                                lastmoveY);
123                         Biale.death(x, y);
124                     }
125
126                     if (who == 'x')
127                         return konwersja();
128
129                     plansza[lastmoveX, lastmoveY] = '0';
130                     plansza[x, y] = who;
131                     //checkcheck
132
133                     ruch = !ruch;
134                     clicked = false;
```

```
135                 }
136             }
137         }
138     }
139     return konwersja();
140 }
```

Funkcja 'rosh' inicjuje roszadę na planszy:

```
1  public void rosh(int[,] tmpr, int x, int y, int i)
2  {
3      if (tmpr[i, 1] == 2)
4      {
5          if (ruch)
6          {
7              Biale.roszada(false, true, lastmoveX, lastmoveY);
8              plansza[x, y] = 'k';
9              plansza[x, y + 1] = 'w';
10             plansza[x, 0] = '0';
11             plansza[x, 4] = '0';
12         }
13         else
14         {
15             Czarne.roszada(false, true, lastmoveX, lastmoveY);
16             plansza[x, y] = 'k';
17             plansza[x, y + 1] = 'w';
18             plansza[x, 0] = '0';
19             plansza[x, 4] = '0';
20         }
21     }
22     if (tmpr[i, 1] == 6)
23     {
24         if (ruch)
25         {
26             Biale.roszada(false, false, lastmoveX, lastmoveY);
27             plansza[x, y] = 'k';
28             plansza[x, y - 1] = 'w';
29             plansza[x, 7] = '0';
30             plansza[x, 4] = '0';
31         }
32         else
33         {
34             Czarne.roszada(false, false, lastmoveX, lastmoveY);
35             plansza[x, y] = 'k';
36             plansza[x, y - 1] = 'w';
37             plansza[x, 7] = '0';
38             plansza[x, 4] = '0';
39         }
40     }
41
42     ruch = !ruch;
43     clicked = false;
44     if (checkmate())
45     {
46         Console.WriteLine("WIN");
```

```
47      }
48      Console.WriteLine("ROSZADA");
49  }
```

'rivia' oraz 'revive' są metodami używanymi do promocji pionów.

```
1  public string[] rev(int x, int y)
2  {
3      string[] tmp2 = new string[4];
4      char[] tmp1;
5      xdorev = x;
6      ydorev = y;
7
8      if (!ruch)
9      {
10             tmp1 = Biale.whorevive(x, y);
11             for (int j = 0; j < 4; j++)
12             {
13                     tmp2[j] = "b" + tmp1[j];
14             }
15     }
16     else
17     {
18             tmp1 = Czarne.whorevive(x, y);
19             for (int j = 0; j < 4; j++)
20             {
21                     tmp2[j] = "c" + tmp1[j];
22             }
23     }
24
25     for (int i = 0; i < tmp2.Length; i++)
26     {
27         switch (tmp2[i])
28         {
29             case "cw":
30                 tmp2[i] = "/ViewModel/Figury/cw.png";
31                 break;
32             case "bw":
33                 tmp2[i] = "/ViewModel/Figury/bw.png";
34                 break;
35             case "cg":
36                 tmp2[i] = "/ViewModel/Figury/cg.png";
37                 break;
38             case "bg":
39                 tmp2[i] = "/ViewModel/Figury/bg.png";
40                 break;
41             case "cs":
42                 tmp2[i] = "/ViewModel/Figury/cs.png";
43                 break;
44             case "bs":
45                 tmp2[i] = "/ViewModel/Figury/bs.png";
46                 break;
47             case "ch":
48                 tmp2[i] = "/ViewModel/Figury/ch.png";
49                 break;
```

```
50                 case "bh":
51                     tmp2[i] = "/ViewModel/Figury/bh.png";
52                     break;
53             }
54         }
55 
56         return tmp2;
57 }
58 
59 public string[] rivia(int x)
60 {
61         char wybraniec = 'h';
62 
63         switch (x)
64         {
65             case 1:
66                 wybraniec = 'w';
67                 break;
68             case 2:
69                 wybraniec = 's';
70                 break;
71             case 3:
72                 wybraniec = 'g';
73                 break;
74             case 4:
75                 wybraniec = 'h';
76                 break;
77 
78         }
79         if (!ruch)
80         {
81             Biale.revive(wybraniec, false);
82         }
83         else
84         {
85             Czarne.revive(wybraniec, true);
86         }
87         plansza[xdorev, ydorev] = wybraniec;
88 
89         return konwersja();
90 }
```

Funkcja 'bazowy kolor' czyści podświetlone pola szacownicy po kliknięciu na innego pionka, wolne pole lub po wykonaniu ruchu.

```
1 public string[] bazowykolor()
2 {
3         string[] tmp = new string[64];
4         bool przelacznik = true;
5         for (int i = 0; i < 64; i++)
6         {
7             if (przelacznik)
8                 tmp[i] = "AntiqueWhite";
9             else
10                tmp[i] = "DarkSlateGray";
```

```
11
12          przelacznik = !przelacznik;
13          if (i == 7 || (i - 7) % 8 == 0)
14              przelacznik = !przelacznik;
15      }
16      return tmp;
17  }
```

---

metoda 'konwersja' konwertuje dwuwymiarową planszę na jednowymiarową tablicę stringów oraz zapełnia ją ścieżkami do ikon figur:

```
1  public string[] konwersja()
2  {
3      string[] tmp = new string[64];
4      for (int i = 0; i < 8; i++)
5      {
6          for (int j = 0; j < 8; j++)
7          {
8              if (plansza[i, j] == '0')
9                  tmp[8 * i + j] = " ";
10             else
11             {
12                 if(Biale.rozklad[i,j] != '0')
13                     tmp[8 * i + j] = "b" + plansza[i, j];
14                 else
15                     tmp[8 * i + j] = "c" + plansza[i, j];
16             }
17
18         }
19     }
20     for (int i = 0; i < tmp.Length; i++)
21     {
22         switch (tmp[i])
23         {
24             case "cp":
25                 tmp[i] = "/ViewModel/Figury/cp.png";
26                 break;
27             case "bp":
28                 tmp[i] = "/ViewModel/Figury/bp.png";
29                 break;
30             case "cw":
31                 tmp[i] = "/ViewModel/Figury/cw.png";
32                 break;
33             case "bw":
34                 tmp[i] = "/ViewModel/Figury/bw.png";
35                 break;
36             case "cg":
37                 tmp[i] = "/ViewModel/Figury/cg.png";
38                 break;
39             case "bg":
40                 tmp[i] = "/ViewModel/Figury/bg.png";
41                 break;
42             case "cs":
43                 tmp[i] = "/ViewModel/Figury/cs.png";
44                 break;
```

```
45                 case "bs":
46                     tmp[i] = "/ViewModel/Figury/bs.png";
47                     break;
48                 case "ch":
49                     tmp[i] = "/ViewModel/Figury/ch.png";
50                     break;
51                 case "bh":
52                     tmp[i] = "/ViewModel/Figury/bh.png";
53                     break;
54                 case "ck":
55                     tmp[i] = "/ViewModel/Figury/ck.png";
56                     break;
57                 case "bk":
58                     tmp[i] = "/ViewModel/Figury/bk.png";
59                     break;
60                 case " ":
61                     tmp[i] = "/ViewModel/Figury/clear.png";
62                     break;
63             }
64         }
65         return tmp;
66 }
```

Następnie mamy serię funkcji przewidujących ruch, tak aby wykluczyć te, w których narażamy swojego króla na szach.

```
1  public bool checkcheck(char[,] fakeplansza, int x, int y)
2  {
3      int[,] tmp;
4      if (!ruch)
5      {
6          tmp = Biale.mozliwosciarmii(fakeplansza, x, y);
7      }
8      else
9      {
10         tmp = Czarne.mozliwosciarmii(fakeplansza, x, y);
11     }
12
13     if(tmp != null)
14     {
15         for (int k = 0; k < tmp.Length / 2; k++)
16         {
17             if (fakeplansza[tmp[k, 0], tmp[k, 1]] == 'k')
18             {
19                 return true;
20             }
21         }
22     }
23     return false;
24 }
25
26 public int[,] roshadachecker(char[,] planszaa, int xx, int yy)
27 {
28     int[,] tmp;
29     if (ruch)
```

```
30          tmp = Biale.roszada(true, true, xx, yy);
31      else
32          tmp = Czarne.roszada(true, true, xx, yy);
33      if (tmp == null)
34          return null;
35
36      List<int> tmpp = new List<int>();
37      char[,] podmianka = new char[8, 8];
38      int[,] after;
39      int x;
40      if (ruch)
41          x = 7;
42      else
43          x = 0;
44
45      for (int i = 0; i < 8; i++)
46      {
47          for (int j = 0; j < 8; j++)
48          {
49              podmianka[i, j] = planszaa[i, j];
50          }
51      }
52
53      for (int i = 0; i < tmp.Length / 2; i++)
54      {
55          if(tmp[i,1] == 2)
56          {
57              podmianka[x, tmp[i, 1]] = 'k';
58              podmianka[x, tmp[i, 1] + 1] = 'w';
59              podmianka[x, 0] = '0';
60              podmianka[x, 4] = '0';
61              if (!checkcheck(podmianka, 8, 8))
62              {
63                  tmpp.Add(x);
64                  tmpp.Add(tmp[i, 1]);
65              }
66              podmianka[x, tmp[i, 1]] = '0';
67              podmianka[x, tmp[i, 1] + 1] = '0';
68              podmianka[x, 0] = 'w';
69              podmianka[x, 4] = 'k';
70
71          }
72
73          if (tmp[i, 1] == 6)
74          {
75              podmianka[x, tmp[i, 1]] = 'k';
76              podmianka[x, tmp[i, 1] - 1] = 'w';
77              podmianka[x, 7] = '0';
78              podmianka[x, 4] = '0';
79              if (!checkcheck(podmianka, 8, 8))
80              {
81                  tmpp.Add(x);
82                  tmpp.Add(tmp[i, 1]);
83              }
84              podmianka[x, tmp[i, 1]] = '0';
```

```
85              podmianka[x, tmp[i, 1] - 1] = '0';
86              podmianka[x, 7] = 'w';
87              podmianka[x, 4] = 'k';
88
89          }
90      }
91      after = listto2darray(tmpp);
92      return after;
93  }
94
95  public int[,] checker(int[,] b4, int x, int y, char[,] planszaa)
96  {
97      List<int> tmp = new List<int>();
98      char[,] podmianka = new char[8,8];
99      int[,] after;
100
101     for(int i = 0; i < 8; i++)
102     {
103         for(int j = 0; j< 8; j++)
104         {
105             podmianka[i, j] = planszaa[i, j];
106         }
107     }
108
109     if (b4 != null)
110     {
111         char ktos = podmianka[x, y];
112         podmianka[x, y] = '0';
113         for (int i = 0; i< b4.Length / 2; i++)
114         {
115             if(podmianka[b4[i, 0], b4[i, 1]] != '0')
116             {
117                 podmianka[b4[i, 0], b4[i, 1]] = ktos;
118                 if (!checkcheck(podmianka, b4[i, 0], b4[i, 1]))
119                 {
120                     tmp.Add(b4[i, 0]);
121                     tmp.Add(b4[i, 1]);
122                 }
123             }
124             else
125             {
126                 podmianka[b4[i, 0], b4[i, 1]] = ktos;
127
128                 if (!checkcheck(podmianka, 8, 8))
129                 {
130                     tmp.Add(b4[i, 0]);
131                     tmp.Add(b4[i, 1]);
132                 }
133             }
134
135             podmianka[b4[i, 0], b4[i, 1]] = '0';
136         }
137         after = listto2darray(tmp);
138         return after;
139     }
```

```
140     return null;
141 }
```

Ostatnią funkcją wpływającą na rozgrywkę jest 'chceckmate'. Sprawdza możliowści armii przeciwnika jeśli zamatujemy jego króla, a w przypadku ich braku, kończy rozgrywkę.

```
1  public bool checkmate()
2  {
3      int[,] tmp;
4      int[,] tmp1;
5      int[,] tmp2;
6      List<int> zasoby = new List<int>();
7
8      if (ruch)
9      {
10         tmp = Biale.zasobyarmii();
11         for (int i = 0; i<tmp.Length/2; i++)
12         {
13             tmp1 = Biale.TheChosenOne(tmp[i,0], tmp[i,1], plansza, false
                   );
14             tmp2 = Biale.TheChosenOne(tmp[i, 0], tmp[i, 1], plansza,
                   true);
15
16             tmp1 = checker(tmp1, tmp[i, 0], tmp[i, 1], plansza);
17             tmp2 = checker(tmp2, tmp[i, 0], tmp[i, 1], plansza);
18
19             if(tmp1 != null)
20             {
21                 for (int a = 0; a < tmp1.Length / 2; a++)
22                 {
23                     zasoby.Add(tmp1[a, 0]);
24                     zasoby.Add(tmp1[a, 1]);
25                 }
26             }
27             if (tmp2 != null)
28             {
29                 for (int a = 0; a < tmp2.Length / 2; a++)
30                 {
31                     zasoby.Add(tmp2[a, 0]);
32                     zasoby.Add(tmp2[a, 1]);
33                 }
34             }
35         }
36         tmp = listto2darray(zasoby);
37     }
38     else
39     {
40         tmp = Czarne.zasobyarmii();
41         for (int i = 0; i < tmp.Length / 2; i++)
42         {
43             tmp1 = Czarne.TheChosenOne(tmp[i, 0], tmp[i, 1], plansza,
                   false);
44             tmp2 = Czarne.TheChosenOne(tmp[i, 0], tmp[i, 1], plansza,
                   true);
45
```

```
46              tmp1 = checker(tmp1, tmp[i, 0], tmp[i, 1], plansza);
47              tmp2 = checker(tmp2, tmp[i, 0], tmp[i, 1], plansza);
48
49              if (tmp1 != null)
50              {
51                  for (int a = 0; a < tmp1.Length / 2; a++)
52                  {
53                      zasoby.Add(tmp1[a, 0]);
54                      zasoby.Add(tmp1[a, 1]);
55                  }
56              }
57              if (tmp2 != null)
58              {
59                  for (int a = 0; a < tmp2.Length / 2; a++)
60                  {
61                      zasoby.Add(tmp2[a, 0]);
62                      zasoby.Add(tmp2[a, 1]);
63                  }
64              }
65          }
66          tmp = listto2darray(zasoby);
67      }
68      if (tmp.Length == 0)
69      {
70          return true;
71      }
72      else
73      {
74          for (int a = 0; a < tmp.Length / 2; a++)
75          {
76              Console.WriteLine(tmp[a, 0] + " : " + tmp[a, 1]);
77          }
78          Console.WriteLine("----------------------------");
79          return false;
80      }
81  }
```

Funkcją ułatwiającą kodowanie jest 'listto2darray'. Używanie list pomaga w optymalizacji zasobów a zmiana na dwuwymiarowe tablice ułatwia pracę twórczą przy kodzie.

```
1  public int[,] listto2darray(List<int> list)
2  {
3      int[,] array2d;
4
5      if (list != null)
6      {
7          array2d = new int[list.Count / 2, 2];
8          int ii = 0;
9          for (int i = 0; i < list.Count; i++)
10         {
11             if (i % 2 == 0)
12                 array2d[ii, 0] = list[i];
13             else
14             {
15                 array2d[ii, 1] = list[i];
```

```
16                      ii++;
17                  }
18              }
19          }
20          else
21              array2d = null;
22          return array2d;
23      }
```

Pełen kod aplikacji

MainWindow.xaml.cs

```
1  namespace Gambit
2  {
3      public partial class MainWindow : Window
4      {
5          public MainWindow()
6          {
7              InitializeComponent();
8          }
9      }
10 }
```

MainWindow.xaml

```
1  <Window x:Class="Gambit.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
               "
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility
               /2006"
6          xmlns:local="clr-namespace:Gambit"
7          xmlns:vm="clr-namespace:Gambit.ViewModel"
8          mc:Ignorable="d"
9          Title="Gambiit" Height="640" Width="660">
10
11     <Window.DataContext>
12         <vm:ViewModel x:Name="viewModel"/>
13     </Window.DataContext>
14
15
16     <Grid>
17         <Grid.RowDefinitions>
18             <RowDefinition Height="1*"/>
19             <RowDefinition Height="1*"/>
20             <RowDefinition Height="1*"/>
21             <RowDefinition Height="1*"/>
22             <RowDefinition Height="1*"/>
23             <RowDefinition Height="1*"/>
24             <RowDefinition Height="1*"/>
25             <RowDefinition Height="1*"/>
26         </Grid.RowDefinitions>
27         <Grid.ColumnDefinitions>
28             <ColumnDefinition Width="1*"/>
29             <ColumnDefinition Width="1*"/>
30             <ColumnDefinition Width="1*"/>
31             <ColumnDefinition Width="1*"/>
32             <ColumnDefinition Width="1*"/>
33             <ColumnDefinition Width="1*"/>
```

```
34              <ColumnDefinition Width="1*"/>
35              <ColumnDefinition Width="1*"/>
36          </Grid.ColumnDefinitions>
37
38          <Button Grid.Row="0" Grid.Column="0" FontSize="50" Background="{
                Binding Kolor[0],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="00"><Image Source="{Binding Board[0],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
39          <Button Grid.Row="0" Grid.Column="1" FontSize="50" Background="{
                Binding Kolor[1],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="01"><Image Source="{Binding Board[1],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
40          <Button Grid.Row="0" Grid.Column="2" FontSize="50" Background="{
                Binding Kolor[2],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="02"><Image Source="{Binding Board[2],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
41          <Button Grid.Row="0" Grid.Column="3" FontSize="50" Background="{
                Binding Kolor[3],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="03"><Image Source="{Binding Board[3],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
42          <Button Grid.Row="0" Grid.Column="4" FontSize="50" Background="{
                Binding Kolor[4],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="04"><Image Source="{Binding Board[4],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
43          <Button Grid.Row="0" Grid.Column="5" FontSize="50" Background="{
                Binding Kolor[5],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="05"><Image Source="{Binding Board[5],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
44          <Button Grid.Row="0" Grid.Column="6" FontSize="50" Background="{
                Binding Kolor[6],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="06"><Image Source="{Binding Board[6],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
45          <Button Grid.Row="0" Grid.Column="7" FontSize="50" Background="{
                Binding Kolor[7],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="07"><Image Source="{Binding Board[7],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
46          <Button Grid.Row="1" Grid.Column="0" FontSize="50" Background="{
                Binding Kolor[8],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="10"><Image Source="{Binding Board[8],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
47          <Button Grid.Row="1" Grid.Column="1" FontSize="50" Background="{
                Binding Kolor[9],Mode=OneWay}" Command="{Binding Clicked}"
                CommandParameter="11"><Image Source="{Binding Board[9],Mode=
                OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
                Binding Foc,Mode=OneWay}" /></Button>
48          <Button Grid.Row="1" Grid.Column="2" FontSize="50" Background="{
```

```
            Binding Kolor[10],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="12"><Image Source="{Binding Board[10],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
49          <Button Grid.Row="1" Grid.Column="3" FontSize="50" Background="{
            Binding Kolor[11],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="13"><Image Source="{Binding Board[11],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
50          <Button Grid.Row="1" Grid.Column="4" FontSize="50" Background="{
            Binding Kolor[12],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="14"><Image Source="{Binding Board[12],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
51          <Button Grid.Row="1" Grid.Column="5" FontSize="50" Background="{
            Binding Kolor[13],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="15"><Image Source="{Binding Board[13],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
52          <Button Grid.Row="1" Grid.Column="6" FontSize="50" Background="{
            Binding Kolor[14],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="16"><Image Source="{Binding Board[14],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
53          <Button Grid.Row="1" Grid.Column="7" FontSize="50" Background="{
            Binding Kolor[15],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="17"><Image Source="{Binding Board[15],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
54          <Button Grid.Row="2" Grid.Column="0" FontSize="50" Background="{
            Binding Kolor[16],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="20"><Image Source="{Binding Board[16],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
55          <Button Grid.Row="2" Grid.Column="1" FontSize="50" Background="{
            Binding Kolor[17],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="21"><Image Source="{Binding Board[17],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
56          <Button Grid.Row="2" Grid.Column="2" FontSize="50" Background="{
            Binding Kolor[18],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="22"><Image Source="{Binding Board[18],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
57          <Button Grid.Row="2" Grid.Column="3" FontSize="50" Background="{
            Binding Kolor[19],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="23"><Image Source="{Binding Board[19],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
58          <Button Grid.Row="2" Grid.Column="4" FontSize="50" Background="{
            Binding Kolor[20],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="24"><Image Source="{Binding Board[20],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
59          <Button Grid.Row="2" Grid.Column="5" FontSize="50" Background="{
```

```
            Binding Kolor[21],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="25"><Image Source="{Binding Board[21],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
60          <Button Grid.Row="2" Grid.Column="6" FontSize="50" Background="{
            Binding Kolor[22],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="26"><Image Source="{Binding Board[22],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
61          <Button Grid.Row="2" Grid.Column="7" FontSize="50" Background="{
            Binding Kolor[23],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="27"><Image Source="{Binding Board[23],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
62          <Button Grid.Row="3" Grid.Column="0" FontSize="50" Background="{
            Binding Kolor[24],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="30"><Image Source="{Binding Board[24],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
63          <Button Grid.Row="3" Grid.Column="1" FontSize="50" Background="{
            Binding Kolor[25],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="31"><Image Source="{Binding Board[25],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
64          <Button Grid.Row="3" Grid.Column="2" FontSize="50" Background="{
            Binding Kolor[26],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="32"><Image Source="{Binding Board[26],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
65          <Button Grid.Row="3" Grid.Column="3" FontSize="50" Background="{
            Binding Kolor[27],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="33"><Image Source="{Binding Board[27],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
66          <Button Grid.Row="3" Grid.Column="4" FontSize="50" Background="{
            Binding Kolor[28],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="34"><Image Source="{Binding Board[28],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
67          <Button Grid.Row="3" Grid.Column="5" FontSize="50" Background="{
            Binding Kolor[29],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="35"><Image Source="{Binding Board[29],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
68          <Button Grid.Row="3" Grid.Column="6" FontSize="50" Background="{
            Binding Kolor[30],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="36"><Image Source="{Binding Board[30],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
69          <Button Grid.Row="3" Grid.Column="7" FontSize="50" Background="{
            Binding Kolor[31],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="37"><Image Source="{Binding Board[31],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
70          <Button Grid.Row="4" Grid.Column="0" FontSize="50" Background="{
```

```
            Binding Kolor[32],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="40"><Image Source="{Binding Board[32],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
71          <Button Grid.Row="4" Grid.Column="1" FontSize="50" Background="{
            Binding Kolor[33],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="41"><Image Source="{Binding Board[33],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
72          <Button Grid.Row="4" Grid.Column="2" FontSize="50" Background="{
            Binding Kolor[34],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="42"><Image Source="{Binding Board[34],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
73          <Button Grid.Row="4" Grid.Column="3" FontSize="50" Background="{
            Binding Kolor[35],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="43"><Image Source="{Binding Board[35],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
74          <Button Grid.Row="4" Grid.Column="4" FontSize="50" Background="{
            Binding Kolor[36],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="44"><Image Source="{Binding Board[36],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
75          <Button Grid.Row="4" Grid.Column="5" FontSize="50" Background="{
            Binding Kolor[37],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="45"><Image Source="{Binding Board[37],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
76          <Button Grid.Row="4" Grid.Column="6" FontSize="50" Background="{
            Binding Kolor[38],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="46"><Image Source="{Binding Board[38],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
77          <Button Grid.Row="4" Grid.Column="7" FontSize="50" Background="{
            Binding Kolor[39],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="47"><Image Source="{Binding Board[39],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
78          <Button Grid.Row="5" Grid.Column="0" FontSize="50" Background="{
            Binding Kolor[40],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="50"><Image Source="{Binding Board[40],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
79          <Button Grid.Row="5" Grid.Column="1" FontSize="50" Background="{
            Binding Kolor[41],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="51"><Image Source="{Binding Board[41],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
80          <Button Grid.Row="5" Grid.Column="2" FontSize="50" Background="{
            Binding Kolor[42],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="52"><Image Source="{Binding Board[42],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
81          <Button Grid.Row="5" Grid.Column="3" FontSize="50" Background="{
```

```
            Binding Kolor[43],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="53"><Image Source="{Binding Board[43],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
82          <Button Grid.Row="5" Grid.Column="4" FontSize="50" Background="{
            Binding Kolor[44],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="54"><Image Source="{Binding Board[44],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
83          <Button Grid.Row="5" Grid.Column="5" FontSize="50" Background="{
            Binding Kolor[45],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="55"><Image Source="{Binding Board[45],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
84          <Button Grid.Row="5" Grid.Column="6" FontSize="50" Background="{
            Binding Kolor[46],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="56"><Image Source="{Binding Board[46],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
85          <Button Grid.Row="5" Grid.Column="7" FontSize="50" Background="{
            Binding Kolor[47],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="57"><Image Source="{Binding Board[47],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
86          <Button Grid.Row="6" Grid.Column="0" FontSize="50" Background="{
            Binding Kolor[48],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="60"><Image Source="{Binding Board[48],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
87          <Button Grid.Row="6" Grid.Column="1" FontSize="50" Background="{
            Binding Kolor[49],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="61"><Image Source="{Binding Board[49],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
88          <Button Grid.Row="6" Grid.Column="2" FontSize="50" Background="{
            Binding Kolor[50],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="62"><Image Source="{Binding Board[50],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
89          <Button Grid.Row="6" Grid.Column="3" FontSize="50" Background="{
            Binding Kolor[51],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="63"><Image Source="{Binding Board[51],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
90          <Button Grid.Row="6" Grid.Column="4" FontSize="50" Background="{
            Binding Kolor[52],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="64"><Image Source="{Binding Board[52],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
91          <Button Grid.Row="6" Grid.Column="5" FontSize="50" Background="{
            Binding Kolor[53],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="65"><Image Source="{Binding Board[53],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
92          <Button Grid.Row="6" Grid.Column="6" FontSize="50" Background="{
```

```
            Binding Kolor[54],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="66"><Image Source="{Binding Board[54],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
93          <Button Grid.Row="6" Grid.Column="7" FontSize="50" Background="{
            Binding Kolor[55],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="67"><Image Source="{Binding Board[55],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
94          <Button Grid.Row="7" Grid.Column="0" FontSize="50" Background="{
            Binding Kolor[56],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="70"><Image Source="{Binding Board[56],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
95          <Button Grid.Row="7" Grid.Column="1" FontSize="50" Background="{
            Binding Kolor[57],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="71"><Image Source="{Binding Board[57],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
96          <Button Grid.Row="7" Grid.Column="2" FontSize="50" Background="{
            Binding Kolor[58],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="72"><Image Source="{Binding Board[58],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
97          <Button Grid.Row="7" Grid.Column="3" FontSize="50" Background="{
            Binding Kolor[59],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="73"><Image Source="{Binding Board[59],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
98          <Button Grid.Row="7" Grid.Column="4" FontSize="50" Background="{
            Binding Kolor[60],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="74"><Image Source="{Binding Board[60],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
99          <Button Grid.Row="7" Grid.Column="5" FontSize="50" Background="{
            Binding Kolor[61],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="75"><Image Source="{Binding Board[61],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
100         <Button Grid.Row="7" Grid.Column="6" FontSize="50" Background="{
            Binding Kolor[62],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="76"><Image Source="{Binding Board[62],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
101         <Button Grid.Row="7" Grid.Column="7" FontSize="50" Background="{
            Binding Kolor[63],Mode=OneWay}" Command="{Binding Clicked}"
            CommandParameter="77"><Image Source="{Binding Board[63],Mode=
            OneWay}" Stretch="None" Height="50" Width="50" Focusable="{
            Binding Foc,Mode=OneWay}" /></Button>
102         <Button Grid.Row="3" Grid.Column="2" Content="Start" FontSize="
            50" Grid.RowSpan="2" Grid.ColumnSpan="4" Background="
            LightPink" Command="{Binding Start}" Visibility="{Binding
            Visibility2,Mode=OneWay}"/>
103
104         <Border BorderBrush="Black" BorderThickness="5" Grid.Row="3"
```

```
            Grid.Column="2" Grid.RowSpan="1" Grid.ColumnSpan="4"
            Visibility="{Binding Visibility,Mode=OneWay}">
105          <Grid Grid.Row="3" Grid.Column="2" Grid.RowSpan="1" Grid.
                 ColumnSpan="4" Background="LightPink" >
106
107              <Grid.ColumnDefinitions>
108                  <ColumnDefinition Width="1*"/>
109                  <ColumnDefinition Width="1*"/>
110                  <ColumnDefinition Width="1*"/>
111                  <ColumnDefinition Width="1*"/>
112              </Grid.ColumnDefinitions>
113
114              <Button Grid.Column="0" Background="LightGreen" Command=
                     "{Binding Wybor}" CommandParameter="1">
115                  <Image Source="{Binding Choice[0],Mode=OneWay}"
                         Stretch="None" Height="50" Width="50" />
116              </Button>
117              <Button Grid.Column="1" Background="LightGreen" Command=
                     "{Binding Wybor}" CommandParameter="2">
118                  <Image Source="{Binding Choice[1],Mode=OneWay}"
                         Stretch="None" Height="50" Width="50" />
119              </Button>
120              <Button Grid.Column="2" Background="LightGreen" Command=
                     "{Binding Wybor}" CommandParameter="3">
121                  <Image Source="{Binding Choice[2],Mode=OneWay}"
                         Stretch="None" Height="50" Width="50" />
122              </Button>
123              <Button Grid.Column="3" Background="LightGreen" Command=
                     "{Binding Wybor}" CommandParameter="4">
124                  <Image Source="{Binding Choice[3],Mode=OneWay}"
                         Stretch="None" Height="50" Width="50" />
125              </Button>
126
127
128          </Grid>
129      </Border>
130
131  </Grid>
132 </Window>
```

ViewModel.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  using System.Windows;
8  using System.Windows.Input;
9
10 namespace Gambit.ViewModel
11 {
```

```
12      class ViewModel : INotifyPropertyChanged
13      {
14          public event PropertyChangedEventHandler PropertyChanged;
15          Model.Silnik silnik = new Model.Silnik();
16
17
18          private string[] board;
19          public string[] Board
20          {
21              get { return board; }
22              private set
23              {
24                  board = value;
25                  PropertyChanged?.Invoke(this, new
                        PropertyChangedEventArgs(nameof(Board)));
26              }
27          }
28
29          private string[] choice = new string[4];
30          public string[] Choice
31          {
32              get { return choice; }
33              private set
34              {
35                  choice = value;
36                  PropertyChanged?.Invoke(this, new
                        PropertyChangedEventArgs(nameof(Choice)));
37              }
38          }
39
40
41          private bool foc;
42          public bool Foc
43          {
44              get { return foc; }
45              private set
46              {
47                  foc = value;
48                  PropertyChanged?.Invoke(this, new
                        PropertyChangedEventArgs(nameof(Foc)));
49
50              }
51          }
52
53          private Visibility visibility = Visibility.Hidden;
54          public Visibility Visibility
55          {
56              get
57              {
58                  return visibility;
59              }
60              set
61              {
62                  visibility = value;
63                  PropertyChanged?.Invoke(this, new
```

```
                PropertyChangedEventArgs(nameof(Visibility)));
64          }
65      }
66
67      private Visibility visibility2;
68      public Visibility Visibility2
69      {
70          get
71          {
72              return visibility2;
73          }
74          set
75          {
76              visibility2 = value;
77              PropertyChanged?.Invoke(this, new
                    PropertyChangedEventArgs(nameof(Visibility2)));
78          }
79      }
80
81
82      private string[] kolor = new string[64];
83      public string[] Kolor
84      {
85          get { return kolor; }
86          private set
87          {
88              kolor = value;
89              PropertyChanged?.Invoke(this, new
                    PropertyChangedEventArgs(nameof(Kolor)));
90          }
91      }
92
93
94      private ICommand start;
95      public ICommand Start
96      {
97          get
98          {
99              return start ?? (start = new Totolotek.ViewModel.
                    BaseClass.RelayCommand(Strt, null));
100         }
101     }
102     public void Strt(object param)
103     {
104         Foc = true;
105         Visibility2 = Visibility.Hidden;
106         Kolor = silnik.bazowykolor();
107         Board = silnik.konwersja();
108     }
109
110
111     private ICommand clicked;
112     public ICommand Clicked
113     {
114         get
```

```
115             {
116                 return clicked ?? (clicked = new Totolotek.ViewModel.
                        BaseClass.RelayCommand(Clck, null));
117             }
118         }
119 
120         public void Clck(object param)
121         {
122             var tmp = (string)param;
123             int x = chartonumber(Convert.ToSByte(tmp[0]));
124             int y = chartonumber(Convert.ToSByte(tmp[1]));
125 
126             if (silnik.hmmm(x, y) != null)
127             {
128                 Kolor = silnik.hmmm(x, y);
129             }
130             else
131                 Kolor = silnik.bazowykolor();
132 
133             Board = silnik.move(x, y);
134             Board = silnik.attack(x, y);
135 
136             for(int i = 0; i < 8; i++)
137             {
138                 if(Board[i] == "/ViewModel/Figury/bp.png")
139                 {
140                     Foc = false;
141                     Visibility = Visibility.Visible;
142                     Choice = silnik.rev(x,y);
143                 }
144             }
145             for (int i = 56; i < 64; i++)
146             {
147                 if (Board[i] == "/ViewModel/Figury/cp.png")
148                 {
149                     Foc = false;
150                     Visibility = Visibility.Visible;
151                     Choice = silnik.rev(x, y);
152                 }
153             }
154 
155             if (silnik.checkmate())
156             {
157                 //Console.WriteLine("WIN");
158                 Foc = false;
159                 MessageBox.Show("Szach-mat", "Partia zako[U+FFFD]czona",
                        MessageBoxButton.OK, MessageBoxImage.Information);
160             }
161         }
162 
163         private ICommand wybor;
164         public ICommand Wybor
165         {
166             get
167             {
```

```
168                 return wybor ?? (wybor = new Totolotek.ViewModel.
                        BaseClass.RelayCommand(Wyb, null));
169             }
170         }
171 
172         public void Wyb(object param)
173         {
174             var tmp = (string)param;
175             int x = chartonumber(Convert.ToSByte(tmp[0]));
176             Board =  silnik.rivia(x);
177 
178             Visibility = Visibility.Hidden;
179             Foc = true;
180         }
181 
182 
183             public int chartonumber(sbyte x)
184             {
185                 switch (x)
186                 {
187                     case 48:
188                         return 0;
189                     case 49:
190                         return 1;
191                     case 50:
192                         return 2;
193                     case 51:
194                         return 3;
195                     case 52:
196                         return 4;
197                     case 53:
198                         return 5;
199                     case 54:
200                         return 6;
201                     case 55:
202                         return 7;
203                     case 56:
204                         return 8;
205                     case 57:
206                         return 9;
207                     default:
208                         return 0;
209                 }
210             }
211     }
212 }
```

Silnik.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
```

```
5  using System.Threading.Tasks;
6
7  namespace Gambit.Model
8  {
9      class Silnik
10     {
11         Armia Biale = new Armia(false);
12         Armia Czarne = new Armia(true);
13
14         //Stack<int[]> moves = new Stack<int[]>();
15
16         bool ruch = true;
17         bool clicked = false;
18
19
20         int lastmoveX = 9;
21         int lastmoveY = 9;
22
23         int xdorev;
24         int ydorev;
25
26         char[,] plansza = new char[8, 8] {  {'w','s','g','h','k','g','s
               ','w' },
27                                            {'p','p','p','p','p','p','p
                                                  ','p' },
28                                            {'0','0','0','0','0','0','0','0'
                                                  },
29                                            {'0','0','0','0','0','0','0','0'
                                                  },
30                                            {'0','0','0','0','0','0','0','0'
                                                  },
31                                            {'0','0','0','0','0','0','0','0'
                                                  },
32                                            {'p','p','p','p','p','p','p','p
                                                  ','p' },
33                                            {'w','s','g','h','k','g','s
                                                  ','w' } };
34
35
36
37         public string[] hmmm(int x, int y)
38         {
39
40             int[,] tmp1;
41             int[,] tmp2;
42             int[,] tmp3;
43
44             if (ruch)
45             {
46                 tmp1 = Biale.TheChosenOne(x, y, plansza, false);
47                 tmp2 = Biale.TheChosenOne(x, y, plansza, true);
48             }
49             else
50             {
51                 tmp1 = Czarne.TheChosenOne(x, y, plansza, false);
```

```
52                  tmp2 = Czarne.TheChosenOne(x, y, plansza, true);
53              }
54
55              tmp1 = checker(tmp1, x, y, plansza);
56              tmp2 = checker(tmp2, x, y, plansza);
57              tmp3 = roshadachecker(plansza, x, y);
58              if (checkcheck(plansza, 8, 8))
59                  tmp3 = null;
60
61
62              if (tmp1 == null && tmp2 == null && tmp3 == null)
63              {
64                  Console.WriteLine("hmm");
65                  return null;
66              }
67
68              string[] hym;
69              hym = bazowykolor();
70
71
72              if (tmp1 != null)
73              {
74                  int[] kolor1 = new int[tmp1.Length / 2];
75
76                  for (int i = 0; i < kolor1.Length; i++)
77                  {
78                      kolor1[i] = 8 * tmp1[i, 0] + tmp1[i, 1];
79                  }
80
81                  for (int i = 0; i < kolor1.Length; i++)
82                  {
83                      hym[kolor1[i]] = "AliceBlue";
84                  }
85              }
86              if (tmp2 != null)
87              {
88                  int[] kolor2 = new int[tmp2.Length / 2];
89
90                  for (int i = 0; i < kolor2.Length; i++)
91                  {
92                      kolor2[i] = 8 * tmp2[i, 0] + tmp2[i, 1];
93                  }
94
95                  for (int i = 0; i < kolor2.Length; i++)
96                  {
97                      hym[kolor2[i]] = "Red";
98                  }
99              }
100             if (tmp3 != null)
101             {
102                 int[] kolor3 = new int[tmp3.Length / 2];
103
104                 for (int i = 0; i < kolor3.Length; i++)
105                 {
106                     kolor3[i] = 8 * tmp3[i, 0] + tmp3[i, 1];
```

```
107                 }
108
109                 for (int i = 0; i < kolor3.Length; i++)
110                 {
111                     hym[kolor3[i]] = "AliceBlue";
112                 }
113             }
114
115
116             lastmove(x, y);
117             return hym;
118         }
119
120         public string[] attack(int x, int y)
121         {
122             if (clicked)
123             {
124                 int[,] tmp;
125                 int[,] tmpr;
126
127                 if (ruch)
128                 {
129                     tmp = Biale.TheChosenOne(lastmoveX, lastmoveY,
                        plansza, true);
130                 }
131                 else
132                 {
133                     tmp = Czarne.TheChosenOne(lastmoveX, lastmoveY,
                        plansza, true);
134                 }
135
136                 //tmp = checker(tmp, x, y, plansza);
137
138                 tmpr = roshadachecker(plansza, lastmoveX, lastmoveY);
139                 if (checkcheck(plansza, 8, 8))
140                     tmpr = null;
141
142                 if (tmp == null && tmpr == null)
143                     return konwersja();
144
145                 if (tmpr != null)
146                 {
147                     for (int i = 0; i < tmpr.Length / 2; i++)
148                     {
149                         if (tmpr[i, 0] == x && tmpr[i, 1] == y)
150                         {
151                             rosh(tmpr, x, y, i);
152                             return konwersja();
153                         }
154                     }
155                 }
156                 if (tmp != null)
157                 {
158                     for (int i = 0; i < tmp.Length / 2; i++)
159                     {
```

```
160                         if (tmp[i, 0] == x && tmp[i, 1] == y)
161                         {
162                             ///
163                             char who;
164                             if (ruch)
165                             {
166                                 who = Biale.armyattack(x, y, lastmoveX,
                                        lastmoveY);
167                                 Czarne.death(x, y);
168                             }
169                             else
170                             {
171                                 who = Czarne.armyattack(x, y, lastmoveX,
                                        lastmoveY);
172                                 Biale.death(x, y);
173                             }
174
175                             if (who == 'x')
176                                 return konwersja();
177
178                             plansza[lastmoveX, lastmoveY] = '0';
179                             plansza[x, y] = who;
180                             //checkcheck
181
182                             ruch = !ruch;
183                             clicked = false;
184
185                         }
186                     }
187                 }
188
189             }
190             return konwersja();
191         }
192
193
194         public string[] move(int x, int y)
195         {
196             if (clicked)
197             {
198                 int[,] tmp;
199                 int[,] tmpr;
200                 if (ruch)
201                 {
202                     tmp = Biale.TheChosenOne(lastmoveX, lastmoveY,
                            plansza, false);
203                 }
204                 else
205                 {
206                     tmp = Czarne.TheChosenOne(lastmoveX, lastmoveY,
                            plansza, false);
207                 }
208
209                 tmp = checker(tmp, lastmoveX, lastmoveY, plansza);
210                 tmpr = roshadachecker(plansza, lastmoveX, lastmoveY);
```

```
211                 if (checkcheck(plansza, 8, 8))
212                     tmpr = null;
213
214                 if (tmp == null && tmpr == null)
215                     return konwersja();
216
217                 if(tmpr != null)
218                 {
219                     for (int i = 0; i < tmpr.Length / 2; i++)
220                     {
221                         if (tmpr[i, 0] == x && tmpr[i, 1] == y)
222                         {
223                             rosh(tmpr, x, y, i);
224                             return konwersja();
225                         }
226                     }
227                 }
228                 if (tmp != null)
229                 {
230                     for (int i = 0; i < tmp.Length / 2; i++)
231                     {
232                         if (tmp[i, 0] == x && tmp[i, 1] == y)
233                         {
234                             char who;
235                             if (ruch)
236                                 who = Biale.armymove(x, y, lastmoveX,
                                        lastmoveY);
237                             else
238                                 who = Czarne.armymove(x, y, lastmoveX,
                                        lastmoveY);
239
240                             if (who == 'x')
241                             {
242                                 Console.WriteLine("shite");
243                                 return konwersja();
244                             }
245
246
247                             plansza[lastmoveX, lastmoveY] = '0';
248                             plansza[x, y] = who;
249                             //checkcheck
250
251                             ruch = !ruch;
252                             clicked = false;
253                             if (checkmate())
254                             {
255                                 Console.WriteLine("WIN");
256                             }
257                         }
258                     }
259                 }
260
261             }
262             return konwersja();
263         }
```

```
264
265
266         public void rosh(int[,] tmpr, int x, int y, int i)
267         {
268             if (tmpr[i, 1] == 2)
269             {
270                 if (ruch)
271                 {
272                     Biale.roszada(false, true, lastmoveX, lastmoveY);
273                     plansza[x, y] = 'k';
274                     plansza[x, y + 1] = 'w';
275                     plansza[x, 0] = '0';
276                     plansza[x, 4] = '0';
277                 }
278                 else
279                 {
280                     Czarne.roszada(false, true, lastmoveX, lastmoveY);
281                     plansza[x, y] = 'k';
282                     plansza[x, y + 1] = 'w';
283                     plansza[x, 0] = '0';
284                     plansza[x, 4] = '0';
285                 }
286             }
287             if (tmpr[i, 1] == 6)
288             {
289                 if (ruch)
290                 {
291                     Biale.roszada(false, false, lastmoveX, lastmoveY);
292                     plansza[x, y] = 'k';
293                     plansza[x, y - 1] = 'w';
294                     plansza[x, 7] = '0';
295                     plansza[x, 4] = '0';
296                 }
297                 else
298                 {
299                     Czarne.roszada(false, false, lastmoveX, lastmoveY);
300                     plansza[x, y] = 'k';
301                     plansza[x, y - 1] = 'w';
302                     plansza[x, 7] = '0';
303                     plansza[x, 4] = '0';
304                 }
305             }
306             ruch = !ruch;
307             clicked = false;
308             if (checkmate())
309             {
310                 Console.WriteLine("WIN");
311             }
312             Console.WriteLine("ROSZADA");
313         }
314
315
316         public string[] rev(int x, int y)
317         {
318             string[] tmp2 = new string[4];
```

```
319             char[] tmp1;
320             xdorev = x;
321             ydorev = y;
322 
323             if (!ruch)
324             {
325                     tmp1 = Biale.whorevive(x, y);
326                     for (int j = 0; j < 4; j++)
327                     {
328                             tmp2[j] = "b" + tmp1[j];
329                     }
330             }
331             else
332             {
333                     tmp1 = Czarne.whorevive(x, y);
334                     for (int j = 0; j < 4; j++)
335                     {
336                             tmp2[j] = "c" + tmp1[j];
337                     }
338             }
339 
340             for (int i = 0; i < tmp2.Length; i++)
341             {
342                 switch (tmp2[i])
343                 {
344                     case "cw":
345                         tmp2[i] = "/ViewModel/Figury/cw.png";
346                         break;
347                     case "bw":
348                         tmp2[i] = "/ViewModel/Figury/bw.png";
349                         break;
350                     case "cg":
351                         tmp2[i] = "/ViewModel/Figury/cg.png";
352                         break;
353                     case "bg":
354                         tmp2[i] = "/ViewModel/Figury/bg.png";
355                         break;
356                     case "cs":
357                         tmp2[i] = "/ViewModel/Figury/cs.png";
358                         break;
359                     case "bs":
360                         tmp2[i] = "/ViewModel/Figury/bs.png";
361                         break;
362                     case "ch":
363                         tmp2[i] = "/ViewModel/Figury/ch.png";
364                         break;
365                     case "bh":
366                         tmp2[i] = "/ViewModel/Figury/bh.png";
367                         break;
368                 }
369             }
370 
371             return tmp2;
372         }
373 
```

```
374         public string[] rivia(int x)
375         {
376             char wybraniec = 'h';
377 
378             switch (x)
379             {
380                 case 1:
381                     wybraniec = 'w';
382                     break;
383                 case 2:
384                     wybraniec = 's';
385                     break;
386                 case 3:
387                     wybraniec = 'g';
388                     break;
389                 case 4:
390                     wybraniec = 'h';
391                     break;
392             }
393             if (!ruch)
394             {
395                 Biale.revive(wybraniec, false);
396             }
397             else
398             {
399                 Czarne.revive(wybraniec, true);
400             }
401             plansza[xdorev, ydorev] = wybraniec;
402 
403             return konwersja();
404         }
405 
406 
407         public string[] bazowykolor()
408         {
409             string[] tmp = new string[64];
410             bool przelacznik = true;
411             for (int i = 0; i < 64; i++)
412             {
413                 if (przelacznik)
414                     tmp[i] = "AntiqueWhite";
415                 else
416                     tmp[i] = "DarkSlateGray";
417 
418                 przelacznik = !przelacznik;
419                 if (i == 7 || (i - 7) % 8 == 0)
420                     przelacznik = !przelacznik;
421             }
422             return tmp;
423         }
424 
425 
426         public string[] konwersja()
427         {
428             string[] tmp = new string[64];
```

```
429             for (int i = 0; i < 8; i++)
430             {
431                 for (int j = 0; j < 8; j++)
432                 {
433                     if (plansza[i, j] == '0')
434                         tmp[8 * i + j] = " ";
435                     else
436                     {
437                         if(Biale.rozklad[i,j] != '0')
438                             tmp[8 * i + j] = "b" + plansza[i, j];
439                         else
440                             tmp[8 * i + j] = "c" + plansza[i, j];
441                     }
442                 }
443             }
444             for (int i = 0; i < tmp.Length; i++)
445             {
446                 switch (tmp[i])
447                 {
448                     case "cp":
449                         tmp[i] = "/ViewModel/Figury/cp.png";
450                         break;
451                     case "bp":
452                         tmp[i] = "/ViewModel/Figury/bp.png";
453                         break;
454                     case "cw":
455                         tmp[i] = "/ViewModel/Figury/cw.png";
456                         break;
457                     case "bw":
458                         tmp[i] = "/ViewModel/Figury/bw.png";
459                         break;
460                     case "cg":
461                         tmp[i] = "/ViewModel/Figury/cg.png";
462                         break;
463                     case "bg":
464                         tmp[i] = "/ViewModel/Figury/bg.png";
465                         break;
466                     case "cs":
467                         tmp[i] = "/ViewModel/Figury/cs.png";
468                         break;
469                     case "bs":
470                         tmp[i] = "/ViewModel/Figury/bs.png";
471                         break;
472                     case "ch":
473                         tmp[i] = "/ViewModel/Figury/ch.png";
474                         break;
475                     case "bh":
476                         tmp[i] = "/ViewModel/Figury/bh.png";
477                         break;
478                     case "ck":
479                         tmp[i] = "/ViewModel/Figury/ck.png";
480                         break;
481                     case "bk":
482                         tmp[i] = "/ViewModel/Figury/bk.png";
483                         break;
```

```
                    case " ":
                        tmp[i] = "/ViewModel/Figury/clear.png";
                        break;
                }
            }
            return tmp;
        }



        public void lastmove(int x, int y)
        {
            lastmoveX = x;
            lastmoveY = y;
            clicked = true;
        }


        public bool checkcheck(char[,] fakeplansza, int x, int y)
        {
            int[,] tmp;
            if (!ruch)
            {
                tmp = Biale.mozliwosciarmii(fakeplansza, x, y);
            }
            else
            {
                tmp = Czarne.mozliwosciarmii(fakeplansza, x, y);
            }

            if(tmp != null)
            {
                for (int k = 0; k < tmp.Length / 2; k++)
                {
                    if (fakeplansza[tmp[k, 0], tmp[k, 1]] == 'k')
                    {
                        return true;
                    }
                }
            }
            return false;
        }

        public int[,] roshadachecker(char[,] planszaa, int xx, int yy)
        {
            int[,] tmp;
            if (ruch)
                tmp = Biale.roszada(true, true, xx, yy);
            else
                tmp = Czarne.roszada(true, true, xx, yy);
            if (tmp == null)
                return null;

            List<int> tmpp = new List<int>();
            char[,] podmianka = new char[8, 8];
```

```
539             int[,] after;
540             int x;
541             if (ruch)
542                 x = 7;
543             else
544                 x = 0;
545 
546             for (int i = 0; i < 8; i++)
547             {
548                 for (int j = 0; j < 8; j++)
549                 {
550                     podmianka[i, j] = planszaa[i, j];
551                 }
552             }
553 
554             for (int i = 0; i < tmp.Length / 2; i++)
555             {
556                 if(tmp[i,1] == 2)
557                 {
558                     podmianka[x, tmp[i, 1]] = 'k';
559                     podmianka[x, tmp[i, 1] + 1] = 'w';
560                     podmianka[x, 0] = '0';
561                     podmianka[x, 4] = '0';
562                     if (!checkcheck(podmianka, 8, 8))
563                     {
564                         tmpp.Add(x);
565                         tmpp.Add(tmp[i, 1]);
566                     }
567                     podmianka[x, tmp[i, 1]] = '0';
568                     podmianka[x, tmp[i, 1] + 1] = '0';
569                     podmianka[x, 0] = 'w';
570                     podmianka[x, 4] = 'k';
571                 }
572                 if (tmp[i, 1] == 6)
573                 {
574                     podmianka[x, tmp[i, 1]] = 'k';
575                     podmianka[x, tmp[i, 1] - 1] = 'w';
576                     podmianka[x, 7] = '0';
577                     podmianka[x, 4] = '0';
578                     if (!checkcheck(podmianka, 8, 8))
579                     {
580                         tmpp.Add(x);
581                         tmpp.Add(tmp[i, 1]);
582                     }
583                     podmianka[x, tmp[i, 1]] = '0';
584                     podmianka[x, tmp[i, 1] - 1] = '0';
585                     podmianka[x, 7] = 'w';
586                     podmianka[x, 4] = 'k';
587                 }
588             }
589             after = listto2darray(tmpp);
590             return after;
591         }
592 
593         public int[,] checker(int[,] b4, int x, int y, char[,] planszaa)
```

```
        {
            List<int> tmp = new List<int>();
            char[,] podmianka = new char[8,8];
            int[,] after;

            for(int i = 0; i < 8; i++)
            {
                for(int j = 0; j< 8; j++)
                {
                    podmianka[i, j] = planszaa[i, j];
                }
            }

            if (b4 != null)
            {
                char ktos = podmianka[x, y];
                podmianka[x, y] = '0';
                for (int i = 0; i< b4.Length / 2; i++)
                {
                    if(podmianka[b4[i, 0], b4[i, 1]] != '0')
                    {
                        podmianka[b4[i, 0], b4[i, 1]] = ktos;
                        if (!checkcheck(podmianka, b4[i, 0], b4[i, 1]))
                        {
                            tmp.Add(b4[i, 0]);
                            tmp.Add(b4[i, 1]);
                        }
                    }
                    else
                    {
                        podmianka[b4[i, 0], b4[i, 1]] = ktos;

                        if (!checkcheck(podmianka, 8, 8))
                        {
                            tmp.Add(b4[i, 0]);
                            tmp.Add(b4[i, 1]);
                        }
                    }

                    podmianka[b4[i, 0], b4[i, 1]] = '0';
                }
                after = listto2darray(tmp);
                return after;
            }
            return null;
        }

        public bool checkmate()
        {
            int[,] tmp;
            int[,] tmp1;
            int[,] tmp2;
            List<int> zasoby = new List<int>();


```

```
649             if (ruch)
650             {
651                 tmp = Biale.zasobyarmii();
652                 for (int i = 0; i<tmp.Length/2; i++)
653                 {
654                     tmp1 = Biale.TheChosenOne(tmp[i,0], tmp[i,1],
                           plansza, false);
655                     tmp2 = Biale.TheChosenOne(tmp[i, 0], tmp[i, 1],
                           plansza, true);
656 
657                     tmp1 = checker(tmp1, tmp[i, 0], tmp[i, 1], plansza);
658                     tmp2 = checker(tmp2, tmp[i, 0], tmp[i, 1], plansza);
659 
660                     if(tmp1 != null)
661                     {
662                         for (int a = 0; a < tmp1.Length / 2; a++)
663                         {
664                             zasoby.Add(tmp1[a, 0]);
665                             zasoby.Add(tmp1[a, 1]);
666                         }
667                     }
668                     if (tmp2 != null)
669                     {
670                         for (int a = 0; a < tmp2.Length / 2; a++)
671                         {
672                             zasoby.Add(tmp2[a, 0]);
673                             zasoby.Add(tmp2[a, 1]);
674                         }
675                     }
676                 }
677                 tmp = listto2darray(zasoby);
678             }
679             else
680             {
681                 tmp = Czarne.zasobyarmii();
682                 for (int i = 0; i < tmp.Length / 2; i++)
683                 {
684                     tmp1 = Czarne.TheChosenOne(tmp[i, 0], tmp[i, 1],
                           plansza, false);
685                     tmp2 = Czarne.TheChosenOne(tmp[i, 0], tmp[i, 1],
                           plansza, true);
686 
687                     tmp1 = checker(tmp1, tmp[i, 0], tmp[i, 1], plansza);
688                     tmp2 = checker(tmp2, tmp[i, 0], tmp[i, 1], plansza);
689 
690                     if (tmp1 != null)
691                     {
692                         for (int a = 0; a < tmp1.Length / 2; a++)
693                         {
694                             zasoby.Add(tmp1[a, 0]);
695                             zasoby.Add(tmp1[a, 1]);
696                         }
697                     }
698                     if (tmp2 != null)
699                     {
```

```
                        for (int a = 0; a < tmp2.Length / 2; a++)
                        {
                            zasoby.Add(tmp2[a, 0]);
                            zasoby.Add(tmp2[a, 1]);
                        }
                    }
                }
                tmp = listto2darray(zasoby);
            }
            if (tmp.Length == 0)
            {
                return true;
            }
            else
            {
                for (int a = 0; a < tmp.Length / 2; a++)
                {
                    Console.WriteLine(tmp[a, 0] + " : " + tmp[a, 1]);
                }
                Console.WriteLine("----------------------------");
                return false;
            }
        }

        public int[,] listto2darray(List<int> list)
        {
            int[,] array2d;

            if (list != null)
            {
                array2d = new int[list.Count / 2, 2];
                int ii = 0;
                for (int i = 0; i < list.Count; i++)
                {
                    if (i % 2 == 0)
                        array2d[ii, 0] = list[i];
                    else
                    {
                        array2d[ii, 1] = list[i];
                        ii++;
                    }
                }
            }
            else
                array2d = null;
            return array2d;
        }
    }
}
```

Armia.cs

```
using System;
```

```
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Gambit.Model
8  {
9      class Armia
10     {
11         public char[,] rozklad;
12         List<Pion> pionki = new List<Pion>();
13         List<Pion> shadowrealm = new List<Pion>();
14         private int xdorev;
15         private int ydorev;
16         private bool[] roshada = new bool[3] { true, true, true };
17
18         public Armia(bool x)
19         {
20             if (x)
21             {
22                 rozklad = new char[8, 8] {  {'w','s','g','h','k','g','s
                       ','w' },
23                                             {'p','p','p','p','p','p','p
                                                   ','p' },
24                                             {'0','0','0','0','0','0','0','0'
                                                   },
25                                             {'0','0','0','0','0','0','0','0'
                                                   },
26                                             {'0','0','0','0','0','0','0','0'
                                                   },
27                                             {'0','0','0','0','0','0','0','0'
                                                   },
28                                             {'0','0','0','0','0','0','0','0'
                                                   },
29                                             {'0','0','0','0','0','0','0','0'
                                                   } };
30
31                 for (int i = 0; i < 8; i++)
32                 {
33                     for (int j = 0; j < 8; j++)
34                     {
35                         if (rozklad[i, j] != '0')
36                             pionki.Add(new Pion(i, j, rozklad[i, j],
                                   true));
37                     }
38                 }
39             }
40             else
41             {
42                 rozklad = new char[8, 8] {
                       {'0','0','0','0','0','0','0','0' },
43                                             {'0','0','0','0','0','0','0','0'
                                                   },
44                                             {'0','0','0','0','0','0','0','0'
                                                   },
```

```
45                                          {'0','0','0','0','0','0','0','0'
                                                },
46                                          {'0','0','0','0','0','0','0','0'
                                                },
47                                          {'0','0','0','0','0','0','0','0'
                                                },
48                                          {'p','p','p','p','p','p','p
                                                ','p' },
49                                          {'w','s','g','h','k','g','s
                                                ','w' } };
50
51                  for (int i = 0; i < 8; i++)
52                  {
53                      for (int j = 0; j < 8; j++)
54                      {
55                          if (rozklad[i, j] != '0')
56                              pionki.Add(new Pion(i, j, rozklad[i, j],
                                    false));
57                      }
58                  }
59              }
60          }
61
62
63          public int[,] TheChosenOne(int xx, int yy, char[,] plansza, bool
                aom)
64          {
65              for (int i = 0; i < pionki.Count; i++)
66              {
67                  if (pionki[i].X == xx && pionki[i].Y == yy)
68                  {
69                      if (aom)
70                      {
71                          int[,] mb = pionki[i].mozliwosci(plansza, true);
72                          if (mb != null)
73                          {
74                              List<int> attck = new List<int>();
75                              for (int k = 0; k < mb.Length / 2; k++)
76                              {
77                                  if (rozklad[mb[k, 0], mb[k, 1]] == '0')
78                                  {
79
80                                      attck.Add(mb[k, 0]);
81                                      attck.Add(mb[k, 1]);
82                                  }
83                              }
84                              int[,] attack = new int[attck.Count / 2, 2];
85                              attack = pionki[i].listto2darray(attck);
86                              return attack;
87                          }
88                          return null;
89                      }
90
91                      return pionki[i].mozliwosci(plansza, false);
92                  }
```

```
93              }
94              return null;
95          }
96
97
98
99
100
101         public char armymove(int x, int y, int lastx, int lasty)
102         {
103             for (int i = 0; i < pionki.Count; i++)
104             {
105                 if (pionki[i].X == lastx && pionki[i].Y == lasty)
106                 {
107                     pionki[i].move(x, y);
108                     checkrosh(lastx, lasty, pionki[i].Who);
109                     rozklad[lastx, lasty] = '0';
110                     rozklad[x, y] = pionki[i].Who;
111                     return pionki[i].Who;
112                 }
113             }
114             return 'x';
115         }
116
117
118         public void checkrosh(int x, int y, char kto)
119         {
120             if ((x == 7 && y == 7 || x == 0 && y == 7) && kto == 'w')
121             {
122                 roshada[2] = false;
123             }
124             if ((x == 7 && y == 0 || x == 0 && y == 0) && kto == 'w')
125             {
126                 roshada[0] = false;
127             }
128             if ((x == 7 && y == 4 || x == 0 && y == 4) && kto == 'k')
129             {
130                 roshada[1] = false;
131             }
132         }
133
134
135         public int[,] roszada(bool sprawdzam, bool lewa, int x, int y)
136         {
137             if (sprawdzam)
138             {
139                 if (rozklad[x, y] != 'k')
140                     return null;
141                 List<int> mb = new List<int>();
142                 if (roshada[0] == true && roshada[1] == true)
143                 {
144                     if (rozklad[0, 0] == 'w' && rozklad[0, 4] == 'k' &&
                            rozklad[0, 1] == '0' && rozklad[0, 2] == '0' &&
                            rozklad[0, 3] == '0')
145                     {
```

```
146                         mb.Add(0);
147                         mb.Add(2);
148                     }
149                     if (rozklad[7, 4] == 'k' && rozklad[7, 0] == 'w' &&
                            rozklad[7, 1] == '0' && rozklad[7, 2] == '0' &&
                            rozklad[7, 3] == '0')
150                     {
151                         mb.Add(7);
152                         mb.Add(2);
153                     }
154                 }
155                 if (roshada[1] == true && roshada[2] == true)
156                 {
157                     if (rozklad[0, 7] == 'w' && rozklad[0, 4] == 'k' &&
                            rozklad[0, 6] == '0' && rozklad[0, 5] == '0')
158                     {
159                         mb.Add(0);
160                         mb.Add(6);
161                     }
162                     if (rozklad[7, 4] == 'k' && rozklad[7, 7] == 'w' &&
                            rozklad[7, 6] == '0' && rozklad[7, 5] == '0')
163                     {
164                         mb.Add(7);
165                         mb.Add(6);
166                     }
167                 }
168                 int[,] tmp = listto2darray(mb);
169                 return tmp;
170             }
171             else
172             {
173                 if (lewa)
174                 {
175                     if (rozklad[0, 0] == 'w' && rozklad[0, 4] == 'k')
176                     {
177                         rozklad[0, 0] = '0';
178                         rozklad[0, 4] = '0';
179                         rozklad[0, 2] = 'k';
180                         rozklad[0, 3] = 'w';
181                         for (int i = 0; i < pionki.Count; i++)
182                         {
183                             if (pionki[i].Y == 4 && pionki[i].X == 0)
184                                 pionki[i].Y = 2;
185 
186                             if (pionki[i].Y == 0 && pionki[i].X == 0)
187                                 pionki[i].Y = 3;
188                         }
189                     }
190                     if (rozklad[7, 4] == 'k' && rozklad[7, 0] == 'w')
191                     {
192                         rozklad[7, 0] = '0';
193                         rozklad[7, 4] = '0';
194                         rozklad[7, 2] = 'k';
195                         rozklad[7, 3] = 'w';
196                         for (int i = 0; i < pionki.Count; i++)
```

```
                    {
                        if (pionki[i].Y == 4 && pionki[i].X == 7)
                            pionki[i].Y = 2;

                        if (pionki[i].Y == 0 && pionki[i].X == 7)
                            pionki[i].Y = 3;
                    }
                }
            }
            else
            {
                if (rozklad[0, 7] == 'w' && rozklad[0, 4] == 'k')
                {
                    rozklad[0, 4] = '0';
                    rozklad[0, 7] = '0';
                    rozklad[0, 6] = 'k';
                    rozklad[0, 5] = 'w';
                    for (int i = 0; i < pionki.Count; i++)
                    {
                        if (pionki[i].Y == 4 && pionki[i].X == 0)
                            pionki[i].Y = 6;

                        if (pionki[i].Y == 7 && pionki[i].X == 0)
                            pionki[i].Y = 5;
                    }
                }
                if (rozklad[7, 4] == 'k' && rozklad[7, 7] == 'w')
                {
                    rozklad[7, 4] = '0';
                    rozklad[7, 7] = '0';
                    rozklad[7, 6] = 'k';
                    rozklad[7, 5] = 'w';
                    for (int i = 0; i < pionki.Count; i++)
                    {
                        if (pionki[i].Y == 4 && pionki[i].X == 7)
                            pionki[i].Y = 6;

                        if (pionki[i].Y == 7 && pionki[i].X == 7)
                            pionki[i].Y = 5;
                    }
                }
            }
        }
        return null;
    }


    public char armyattack(int x, int y, int lastx, int lasty)
    {
        for (int i = 0; i < pionki.Count; i++)
        {
            if (pionki[i].X == lastx && pionki[i].Y == lasty)
            {
                pionki[i].move(x, y);
                checkrosh(lastx, lasty, pionki[i].Who);
```

```
252                     rozklad[lastx, lasty] = '0';
253                     rozklad[x, y] = pionki[i].Who;
254                     return pionki[i].Who;
255                 }
256             }
257             return 'x';
258         }
259 
260         public void death(int x, int y)
261         {
262             for (int i = 0; i < pionki.Count; i++)
263             {
264                 if (pionki[i].X == x && pionki[i].Y == y)
265                 {
266                     pionki[i].X = 9;
267                     pionki[i].Y = 9;
268                     shadowrealm.Add(pionki[i]);
269                     pionki.RemoveAt(i);
270                     rozklad[x, y] = '0';
271                 }
272             }
273         }
274         public char[] whorevive(int x, int y)
275         {
276             death(x, y);
277             xdorev = x;
278             ydorev = y;
279             char[] wybor = new char[4] { 'w', 's', 'g', 'h'};
280             return wybor;
281         }
282 
283         public void revive(char kto, bool kolor)
284         {
285             pionki.Add(new Pion(xdorev, ydorev, kto, kolor));
286             rozklad[xdorev, ydorev] = kto;
287         }
288 
289 
290         public int[,] mozliwosciarmii(char[,] plansza, int x, int y)
291         {
292             List<int> attck = new List<int>();
293             char[,] podmianka = new char[8, 8];
294             for (int i = 0; i < 8; i++)
295             {
296                 for (int j = 0; j < 8; j++)
297                 {
298                     podmianka[i, j] = rozklad[i, j];
299                 }
300             }
301             if(x !=8 && y != 8)
302             {
303                 podmianka[x, y] = '0';
304             }
305 
306             for (int i = 0; i < pionki.Count; i++)
```

```
307             {
308                 int[,] mb = pionki[i].mozliwosci(plansza, true);
309 
310                 if (pionki[i].X == x && pionki[i].Y == y)
311                     mb = null;
312 
313                 if (mb != null)
314                 {
315                     for (int k = 0; k < mb.Length / 2; k++)
316                     {
317 
318                             if (podmianka[mb[k, 0], mb[k, 1]] == '0')
319                             {
320                                 attck.Add(mb[k, 0]);
321                                 attck.Add(mb[k, 1]);
322                             }
323                     }
324                 }
325 
326             }
327             int[,] attack = new int[attck.Count / 2, 2];
328             attack = listto2darray(attck);
329             return attack;
330         }
331 
332 
333         public int[,] zasobyarmii()
334         {
335             List<int> tmp = new List<int>();
336             for (int i = 0; i < pionki.Count; i++)
337             {
338                 tmp.Add(pionki[i].X);
339                 tmp.Add(pionki[i].Y);
340             }
341 
342             int[,] wojsko = new int[tmp.Count / 2, 2];
343             wojsko = listto2darray(tmp);
344             return wojsko;
345         }
346 
347 
348 
349 
350         public int[,] listto2darray(List<int> list)
351         {
352             int[,] array2d;
353 
354             if (list != null)
355             {
356                 array2d = new int[list.Count / 2, 2];
357                 int ii = 0;
358                 for (int i = 0; i < list.Count; i++)
359                 {
360                     if (i % 2 == 0)
361                         array2d[ii, 0] = list[i];
```

```
362                         else
363                         {
364                             array2d[ii, 1] = list[i];
365                             ii++;
366                         }
367                     }
368                 }
369                 else
370                     array2d = null;
371                 return array2d;
372             }
373         }
374 }
```

Pion.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Gambit.Model
8  {
9      class Pion
10     {
11         public int X { get; set; }
12         public int Y { get; set; }
13         public char Who { get; set; }
14         public bool Kolor { get; set; }
15
16         public Pion(int x, int y, char who, bool kolor)
17         {
18             X = x;
19             Y = y;
20             Who = who;
21             Kolor = kolor;
22         }
23
24         public int[,] mozliwosci(char[,] plansza, bool pm)
25         {
26             List<int> possible = new List<int>();
27             List<int> maybe = new List<int>();
28
29             switch (Who)
30             {
31                 case 'w':
32                     for (int i = X + 1; i < 8; i++)
33                     {
34                         if (plansza[i, Y] == '0')
35                         {
36                             possible.Add(i);
37                             possible.Add(Y);
```

```
38                          }
39
40                          if (plansza[i, Y] != '0')
41                          {
42                              maybe.Add(i);
43                              maybe.Add(Y);
44                              break;
45                          }
46                      }
47
48                      for (int i = X - 1; i >= 0; i--)
49                      {
50                          if (plansza[i, Y] == '0')
51                          {
52                              possible.Add(i);
53                              possible.Add(Y);
54                          }
55
56                          if (plansza[i, Y] != '0')
57                          {
58                              maybe.Add(i);
59                              maybe.Add(Y);
60                              break;
61                          }
62                      }
63
64                      for (int i = Y + 1; i < 8; i++)
65                      {
66                          if (plansza[X, i] == '0')
67                          {
68                              possible.Add(X);
69                              possible.Add(i);
70                          }
71
72                          if (plansza[X, i] != '0')
73                          {
74                              maybe.Add(X);
75                              maybe.Add(i);
76                              break;
77                          }
78                      }
79
80                      for (int i = Y - 1; i >= 0; i--)
81                      {
82                          if (plansza[X, i] == '0')
83                          {
84                              possible.Add(X);
85                              possible.Add(i);
86                          }
87
88                          if (plansza[X, i] != '0')
89                          {
90                              maybe.Add(X);
91                              maybe.Add(i);
92                              break;
```

```
93                          }
94                      }
95                      break;
96
97                  case 'g':
98                      int j = Y + 1;
99                      for (int i = X + 1; i < 8; i++)
100                     {
101                         if (j >= 8)
102                             break;
103                         if (plansza[i, j] == '0')
104                         {
105                             possible.Add(i);
106                             possible.Add(j);
107                         }
108
109                         if (plansza[i, j] != '0')
110                         {
111                             maybe.Add(i);
112                             maybe.Add(j);
113                             break;
114                         }
115
116                         j++;
117                         if (j >= 8)
118                             break;
119                     }
120
121                     j = Y - 1;
122                     for (int i = X + 1; i < 8; i++)
123                     {
124                         if (j < 0)
125                             break;
126                         if (plansza[i, j] == '0')
127                         {
128                             possible.Add(i);
129                             possible.Add(j);
130                         }
131
132                         if (plansza[i, j] != '0')
133                         {
134                             maybe.Add(i);
135                             maybe.Add(j);
136                             break;
137                         }
138
139                         j--;
140                         if (j < 0)
141                             break;
142                     }
143
144                     j = Y - 1;
145                     for (int i = X - 1; i >= 0; i--)
146                     {
147                         if (j < 0)
```

```
148                         break;
149                     if (plansza[i, j] == '0')
150                     {
151                         possible.Add(i);
152                         possible.Add(j);
153                     }
154 
155                     if (plansza[i, j] != '0')
156                     {
157                         maybe.Add(i);
158                         maybe.Add(j);
159                         break;
160                     }
161 
162                     j--;
163                     if (j < 0)
164                         break;
165                 }
166 
167                 j = Y + 1;
168                 for (int i = X - 1; i >= 0; i--)
169                 {
170                     if (j >= 8)
171                         break;
172                     if (plansza[i, j] == '0')
173                     {
174                         possible.Add(i);
175                         possible.Add(j);
176                     }
177 
178                     if (plansza[i, j] != '0')
179                     {
180                         maybe.Add(i);
181                         maybe.Add(j);
182                         break;
183                     }
184                     j++;
185                     if (j >= 8)
186                         break;
187                 }
188 
189                 break;
190 
191             case 'h':
192                 for (int i = X + 1; i < 8; i++)
193                 {
194                     if (plansza[i, Y] == '0')
195                     {
196                         possible.Add(i);
197                         possible.Add(Y);
198                     }
199 
200                     if (plansza[i, Y] != '0')
201                     {
202                         maybe.Add(i);
```

```
203                         maybe.Add(Y);
204                         break;
205                     }
206                 }
207 
208                 for (int i = X - 1; i >= 0; i--)
209                 {
210                     if (plansza[i, Y] == '0')
211                     {
212                         possible.Add(i);
213                         possible.Add(Y);
214                     }
215 
216                     if (plansza[i, Y] != '0')
217                     {
218                         maybe.Add(i);
219                         maybe.Add(Y);
220                         break;
221                     }
222                 }
223 
224                 for (int i = Y + 1; i < 8; i++)
225                 {
226                     if (plansza[X, i] == '0')
227                     {
228                         possible.Add(X);
229                         possible.Add(i);
230                     }
231 
232                     if (plansza[X, i] != '0')
233                     {
234                         maybe.Add(X);
235                         maybe.Add(i);
236                         break;
237                     }
238                 }
239 
240                 for (int i = Y - 1; i >= 0; i--)
241                 {
242                     if (plansza[X, i] == '0')
243                     {
244                         possible.Add(X);
245                         possible.Add(i);
246                     }
247 
248                     if (plansza[X, i] != '0')
249                     {
250                         maybe.Add(X);
251                         maybe.Add(i);
252                         break;
253                     }
254                 }
255 
256                 int jj = Y + 1;
257                 for (int i = X + 1; i < 8; i++)
```

```
258                     {
259                         if (jj >= 8)
260                             break;
261                         if (plansza[i, jj] == '0')
262                         {
263                             possible.Add(i);
264                             possible.Add(jj);
265                         }
266 
267                         if (plansza[i, jj] != '0')
268                         {
269                             maybe.Add(i);
270                             maybe.Add(jj);
271                             break;
272                         }
273 
274                         jj++;
275                         if (jj >= 8)
276                             break;
277                     }
278 
279                     jj = Y - 1;
280                     for (int i = X + 1; i < 8; i++)
281                     {
282                         if (jj < 0)
283                             break;
284                         if (plansza[i, jj] == '0')
285                         {
286                             possible.Add(i);
287                             possible.Add(jj);
288                         }
289                         if (plansza[i, jj] != '0')
290                         {
291                             maybe.Add(i);
292                             maybe.Add(jj);
293                             break;
294                         }
295                         jj--;
296                         if (jj < 0)
297                             break;
298                     }
299 
300                     jj = Y - 1;
301                     for (int i = X - 1; i >= 0; i--)
302                     {
303                         if (jj < 0)
304                             break;
305                         if (plansza[i, jj] == '0')
306                         {
307                             possible.Add(i);
308                             possible.Add(jj);
309                         }
310 
311                         if (plansza[i, jj] != '0')
312                         {
```

```
313                         maybe.Add(i);
314                         maybe.Add(jj);
315                         break;
316                     }
317                     jj--;
318                     if (jj < 0)
319                         break;
320                 }
321 
322                 jj = Y + 1;
323                 for (int i = X - 1; i >= 0; i--)
324                 {
325                     if (jj >= 8)
326                         break;
327                     if (plansza[i, jj] == '0')
328                     {
329                         possible.Add(i);
330                         possible.Add(jj);
331                     }
332 
333                     if (plansza[i, jj] != '0')
334                     {
335                         maybe.Add(i);
336                         maybe.Add(jj);
337                         break;
338                     }
339 
340                     jj++;
341                     if (jj >= 8)
342                         break;
343                 }
344                 break;
345 
346             case 's':
347                 if (X + 2 < 8 && Y + 1 < 8 && plansza[X + 2, Y + 1]
                        == '0')
348                 {
349                     possible.Add(X + 2);
350                     possible.Add(Y + 1);
351                 }
352                 if (X + 2 < 8 && Y + 1 < 8 && plansza[X + 2, Y + 1]
                        != '0')
353                 {
354                     maybe.Add(X + 2);
355                     maybe.Add(Y + 1);
356                 }
357                 if (X + 2 < 8 && Y - 1 >= 0 && plansza[X + 2, Y - 1]
                         == '0')
358                 {
359                     possible.Add(X + 2);
360                     possible.Add(Y - 1);
361                 }
362                 if (X + 2 < 8 && Y - 1 >= 0 && plansza[X + 2, Y - 1]
                         != '0')
363                 {
```

```
364                     maybe.Add(X + 2);
365                     maybe.Add(Y - 1);
366                 }
367
368                 if (X - 2 >= 0 && Y + 1 < 8 && plansza[X - 2, Y + 1]
                        == '0')
369                 {
370                     possible.Add(X - 2);
371                     possible.Add(Y + 1);
372                 }
373                 if (X - 2 >= 0 && Y + 1 < 8 && plansza[X - 2, Y + 1]
                        != '0')
374                 {
375                     maybe.Add(X - 2);
376                     maybe.Add(Y + 1);
377                 }
378                 if (X - 2 >= 0 && Y - 1 >= 0 && plansza[X - 2, Y -
                    1] == '0')
379                 {
380                     possible.Add(X - 2);
381                     possible.Add(Y - 1);
382                 }
383                 if (X - 2 >= 0 && Y - 1 >= 0 && plansza[X - 2, Y -
                    1] != '0')
384                 {
385                     maybe.Add(X - 2);
386                     maybe.Add(Y - 1);
387                 }
388
389                 if (X + 1 < 8 && Y + 2 < 8 && plansza[X + 1, Y + 2]
                    == '0')
390                 {
391                     possible.Add(X + 1);
392                     possible.Add(Y + 2);
393                 }
394                 if (X + 1 < 8 && Y + 2 < 8 && plansza[X + 1, Y + 2]
                    != '0')
395                 {
396                     maybe.Add(X + 1);
397                     maybe.Add(Y + 2);
398                 }
399                 if (X - 1 >= 0 && Y + 2 < 8 && plansza[X - 1, Y + 2]
                        == '0')
400                 {
401                     possible.Add(X - 1);
402                     possible.Add(Y + 2);
403                 }
404                 if (X - 1 >= 0 && Y + 2 < 8 && plansza[X - 1, Y + 2]
                        != '0')
405                 {
406                     maybe.Add(X - 1);
407                     maybe.Add(Y + 2);
408                 }
409
410
```

```
411                 if (X - 1 >= 0 && Y - 2 >= 0 && plansza[X - 1, Y -
                        2] == '0')
412                 {
413                     possible.Add(X - 1);
414                     possible.Add(Y - 2);
415                 }
416                 if (X - 1 >= 0 && Y - 2 >= 0 && plansza[X - 1, Y -
                        2] != '0')
417                 {
418                     maybe.Add(X - 1);
419                     maybe.Add(Y - 2);
420                 }
421                 if (X + 1 < 8 && Y - 2 >= 0 && plansza[X + 1, Y - 2]
                        == '0')
422                 {
423                     possible.Add(X + 1);
424                     possible.Add(Y - 2);
425                 }
426                 if (X + 1 < 8 && Y - 2 >= 0 && plansza[X + 1, Y - 2]
                        != '0')
427                 {
428                     maybe.Add(X + 1);
429                     maybe.Add(Y - 2);
430                 }
431
432                 break;
433
434             case 'k':
435                 for (int i = X - 1; i < X + 2; i++)
436                 {
437                     for (int l = Y - 1; l < Y + 2; l++)
438                     {
439                         if (i >= 0 && i < 8 && l >= 0 && l < 8)
440                         {
441                             if (plansza[i, l] == '0')
442                             {
443                                 possible.Add(i);
444                                 possible.Add(l);
445                             }
446                             if (plansza[i, l] != '0')
447                             {
448                                 maybe.Add(i);
449                                 maybe.Add(l);
450                             }
451                         }
452                     }
453                 }
454
455                 break;
456
457             case 'p':
458                 if (Kolor == true)
459                 {
460                     if (X + 1 < 8 && plansza[X + 1, Y] == '0')
461                     {
```

```
462                         possible.Add(X + 1);
463                         possible.Add(Y);
464
465                         if (X == 1 && plansza[X + 2, Y] == '0')
466                         {
467                             possible.Add(X + 2);
468                             possible.Add(Y);
469                         }
470                     }
471                     if (Y + 1 < 8 && X + 1 < 8 && plansza[X + 1, Y +
                         1] != '0')
472                     {
473                         maybe.Add(X + 1);
474                         maybe.Add(Y + 1);
475                     }
476                     if (Y - 1 >= 0 && X + 1 < 8 && plansza[X + 1, Y
                        - 1] != '0')
477                     {
478                         maybe.Add(X + 1);
479                         maybe.Add(Y - 1);
480                     }
481                 }
482
483                 if (Kolor == false)
484                 {
485                     if (X - 1 >= 0 && plansza[X - 1, Y] == '0')
486                     {
487                         possible.Add(X - 1);
488                         possible.Add(Y);
489
490                         if (X == 6 && plansza[X - 2, Y] == '0')
491                         {
492                             possible.Add(X - 2);
493                             possible.Add(Y);
494                         }
495                     }
496                     if (Y + 1 < 8 && X - 1 >= 0 && plansza[X - 1, Y
                        + 1] != '0')
497                     {
498                         maybe.Add(X - 1);
499                         maybe.Add(Y + 1);
500                     }
501                     if (Y - 1 >= 0 && X - 1 >= 0 && plansza[X - 1, Y
                         - 1] != '0')
502                     {
503                         maybe.Add(X - 1);
504                         maybe.Add(Y - 1);
505                     }
506                 }
507                 break;
508
509             case '0':
510                 possible.Clear();
511                 maybe.Clear();
512                 break;
```

```
513                 }
514
515                 int[,] wyjazd;
516                 if (!pm)
517                     wyjazd = listto2darray(possible);
518                 else
519                     wyjazd = listto2darray(maybe);
520
521                 return wyjazd;
522             }
523
524
525             public void move(int x, int y)
526             {
527                 X = x;
528                 Y = y;
529             }
530
531             public int[,] listto2darray(List<int> list)
532             {
533                 int[,] array2d;
534
535                 if (list != null)
536                 {
537                     array2d = new int[list.Count / 2, 2];
538                     int ii = 0;
539                     for (int i = 0; i < list.Count; i++)
540                     {
541                         if (i % 2 == 0)
542                             array2d[ii, 0] = list[i];
543                         else
544                         {
545                             array2d[ii, 1] = list[i];
546                             ii++;
547                         }
548                     }
549                 }
550                 else
551                     array2d = null;
552                 return array2d;
553             }
554         }
555 }
```