Sprawozdanie

Bartłomiej Pluta

Wprowadzenie i opis programu

Program omawiany w niniejszym sprawozdaniu umożliwia rozegranie partii w warcaby z komputerem. Program został napisany obiektowo w języku C++ z graficznym interfejsem przy wykorzystaniu biblioteki SFML[1].

Gra jest w pewnym stopniu elastyczna, to znaczy, możliwe jest definiowanie m.in. rozmiaru planszy (pionki ustawią się same), koloru gracza, koloru rozpoczynającego, czy poziomu AI (czyli wysokości drzewa MiniMax). Gra jest uruchamiana poleceniem:

```
checkers COLOR AI
```

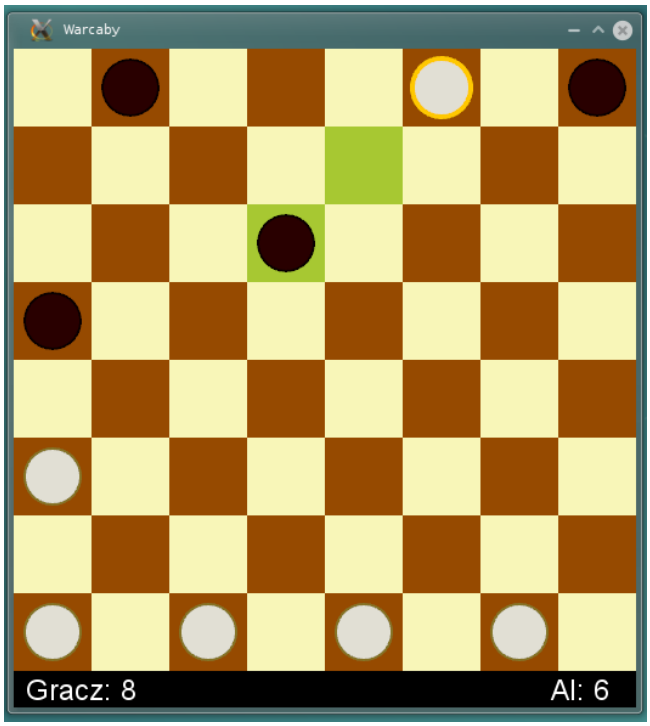
gdzie:
`COLOR` – kolor gracza (dozwolone wartości: `white` i `black`)
`AI` – poziom AI (z zakresu 2 – 8)

Jest również możliwe uruchomienie programu za pomocą samego polecenia `checkers`. Wtedy zostaną przyjęte domyślne parametry.
Parametry, z którymi uruchamiana jest gra, wyświetlane są na standardowym wyjściu.

[1] http://www.sfml-dev.org/

Interfejs

Interfejs gry został zaprojektowany jak najbardziej minimalistycznie. W dolnym lewym rogu znajduje się liczba zbitych przez gracza pionków należących do AI, a w dolnym prawym rogu liczba zbitych przez AI pionków gracza.



Rysunek 1: Interfejs gry.

Pola oznaczone na zielono reprezentują ostatni ruch AI. Pola delikatnie pojaśnione (nie przedstawione na rys. 1) to dozwolone ruchy gracza.

Pionki z wyraźnym złotym pierścieniem to damki.

Grafiki użyte w grze są w stu procentach generowane programowo – są to tzw. prymitywy, czyli proste obiekty geometryczne takie jak kwadrat, czy koło. Czcionka użyta do wyświetlania punktacji to arial (TrueType Font) i jest ona umieszczona w folderze `res/`.

Pliki źródłowe projektu

Plik	Opis zawartości pliku
def.hh	Definicje podstawowych stałych i typów używanych podczas pisania gry.
misc.hh	Implementacje podstawowych pomocniczych typów, takich jak Vector, Color czy Movement.
object.hh	Implementacja abstrakcyjnej klasy Object reprezentującej obiekt na planszy[2].
pawn.hh pawn.cpp	Klasa pionka, który dziedziczy po klasie Object
board.hh board.cpp	Klasa planszy, zawierająca między innymi tablice pionków oraz podstawowe metody obsługi dozwolonych ruchów i bić.
minimax.hh minimax.cpp	Klasa zawierająca metodę heurystyczną oraz metodę implementującą algorytm MiniMax z cięciami alfa-beta.
game.hh game.cpp	Klasa spajająca grę, która zawiera podstawowe metody obsługi rozgrywki.

Tablica 1: Pliki źródłowe projektu.

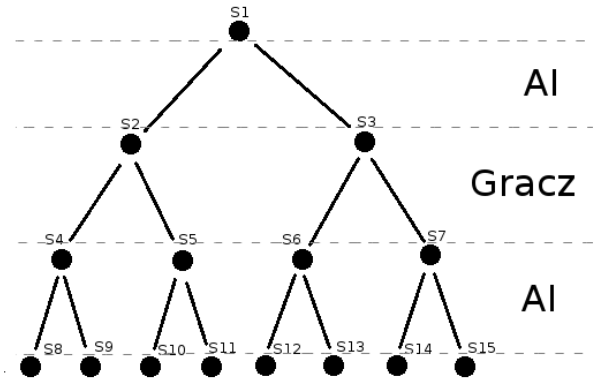
Przyjęte zasady gry

W trakcie pisania, przyjęte zostały następujące zasady gry:

- Pierwszy ruch należy do zawodnika białego.
- Gracze wykonują posunięcia po jednym ruchu na przemian, zawsze swoimi własnymi figurami.
- Pionki poruszają się ukośnie do przodu (o jedno pole), o ile miejsce jest wolne.
- Pionek staje się damką po osiągnięciu linii promocji (przeciwległa krawędź dla krawędzi startowej).
- Damka może wykonać ruch dopiero gdy wykona ruch przeciwnik.
- Damka porusza się podobnie jak pionek, z tym że dozwolone są ruchy do tyłu.
- Biciem nazywa się przeskoczenie po ukosie figury przeciwnika, jeżeli tuż za nim jest wolne miejsce.
- Bicie figury przeciwnika może być wykonywane zarówno do przodu, jak i do tyłu. Liczy się jako jeden ruch.
- Bicie jest obowiązkowe.
- Bicia wielokrotne są niedozwolone.

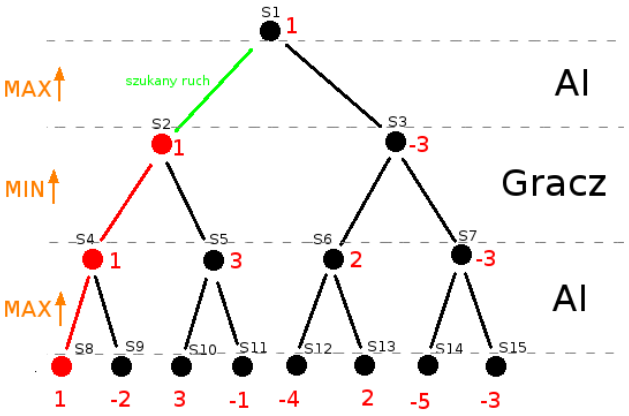
[2]Klasa ta jest pozostałością po pierwotnym koncepcie, gdzie pionki gracza i AI osobno dziedziczą bezpośrednio po Object. Pomysł ten został porzucony na rzecz aktualnego, w którym pionki dziedziczą po klasie Object, lecz dopiero na poziomie klasy Game jest ustalane, do kogo one należą.

Algorytm MiniMax z cięciem Alfa-Beta



Rysunek 2: Przykładowe drzewo gry.

Grę w warcaby możemy rozpisać za pomocą tzw. drzewa gry (rys. 2) które przedstawia wszystkie możliwe scenariusze przez n kolejnych ruchów (gdzie n – głębokość drzewa). W grze został zaimplementowany algorytm MiniMax z cięciem Alfa-Beta (jego kod znajduje się w pliku `src/minimax.cpp`). Jest to algorytm maksymalizujący zysk gracza przy założeniu, że jego przeciwnik wykonuje ruchy minimalizujące zysk gracza.



Rysunek 3: Przykładowe działanie algorytmu MiniMax.

Pseudokod algorytmu MiniMax:

```
Algorytm minimax(S, d)
 Wejście: Stan gry S, aktualna głębokość drzewa d

 Jeżeli jest to poziom liści
   zwróć 1 jeśli wygrana, 0 jeśli remis, -1 jeśli przegrana

 Jeśli jest teraz ruch przeciwnika // (d%2)
   best_value := -INF
   Dla każdego możliwego ruchu m
     S' = S(m) // utwórz tymczasowy stan S i wykonaj ruch m na nim
     v := minimax(S', d+1)
     best_value := min(best_value, v)
   zwróć best_value

 w przeciwnym wypadku
   best_value := INF
   Dla każdego możliwego ruchu m
     S' = S(m) // utwórz tymczasowy stan S i wykonaj ruch m na nim
     v:= minimax(S', d+1)
     best_value := max(best_value, v)
   zwróć best_value
```

Łatwo można zauważyć, że tak napisany algorytm niepotrzebnie przeszukuje gałęzie, które nie mają wpływu na ostateczny wynik. Aby temu zapobiec, a zaoszczędzoną moc obliczeniową wykorzystać np. do zwiększenia poziomu przeszukiwania, zastosowany został algorytm cięć Alfa-Beta, który jest modyfikacją algorytmu MiniMax, polegającą na odcinaniu gałęzi czyniącej obecnie badaną opcję gorszą od poprzednio zbadanych.

Funkcja heurystyczna

W pełnej wersji, algorytm MiniMax przeszukuje całe drzewo gry, od bieżącego ruchu do samego rozstrzygnięcia partii, przez co gracz dysponujący takim algorytmem może w najgorszym przypadku zremisować. Z racji, że w przypadku nietrywialnych gier (jakimi niewątpliwie są warcaby), drzewo gry jest monumentalne, praktycznie nie jest możliwe przeszukiwać je całe na domowej klasy sprzęcie. W tym celu stosuje się tzw. funkcje heurystyczne – czyli inaczej – funkcje oceniające.

Funkcja heurystyczna jest funkcją odwzorowującą stan gry w liczby rzeczywiste.

$$h : \mathbb{S} \to \mathbb{R}$$

Gdzie:

$$\mathbb{S} = \{S_1, S_2, \ldots, S_n\} - \text{przestrzeń stanów}$$

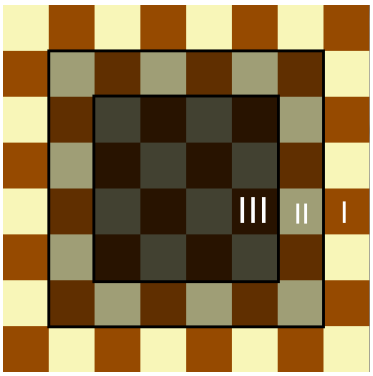
Należy pamiętać, że funkcja heurystyczna jest tylko pewną aproksymacją rzeczywistej i obiektywnej wyceny stanu gry, a nad jej realizacją i odpowiednią kalibracją(bilansowaniem) czuwają eksperci. Dąży się do tego, aby funkcje heurystyczne były kombinacją liniową innych funkcji heurystycznych cząstkowych,

tj.

$$h(s) = \alpha_1 h_1(s) + \alpha_2 h_2(s) + \ldots + \alpha_k h_k(s),\ \alpha_1 \ldots \alpha_k \in \mathbb{R}$$

Zasada trzech obszarów

Zasada trzech obszarów [3] jest cząstkową funkcją heurystyczną oceniającą pionki według ich położenia. Premiuje pionki przy krawędzi(obszar I) a karze pionki „pchające się“ grupą na sam środek warcabnicy (obszar III).



Rysunek 4: Obszary warcabnicy.

Niech $\mathbb{P}$ będzie zbiorem pionków aktualnego stanu gry.

$$f_1(p) = \begin{cases} a_1 & \text{jeśli p znajduje się w pierwszym obszarze} \\ a_2 & \text{jeśli p znajduje się w drugim obszarze} \\ a_3 & \text{jeśli p znajduje się w trzecim obszarze} \end{cases}$$

Gdzie $p \in \mathbb{P}$ jest pionkiem, a $a_1 > a_2 > a_3$ to punktacje za pionek znajdujący się w odpowiednim obszarze. Cząstkowa funkcja heurystyczna w tym wypadku jest sumą punktacji dla wszystkich pionków:

$$h_1(s) = \sum_p f_1(p)$$

Pionki i damki

Drugą funkcją jest wycena pionków i damek.
Niech $\mathbb{P}$ będzie zbiorem pionków aktualnego stanu gry.

$$f_2(p) = \begin{cases} b_1 & \text{jeśli p jest zwykłym pionkiem} \\ b_2 & \text{jeśli p jest damką} \end{cases}$$

Gdzie $p \in \mathbb{P}$ jest pionkiem, a $b_1 < b_2$ to punktacje za odpowiednie pionki. Cząstkowa funkcja heurystyczna w tym wypadku jest również sumą punktacji dla wszystkich pionków:

$$h_2(s) = \sum_p f_2(p)$$

Ostatecznia funkcja heurystyczna

Ostateczna funkcja heurystyczna ma postać:

$$h(s) = \alpha_1 h_1(s) + \alpha_2 h_2(s)$$

A odpowiednie parametry są przedstawione w tabelce: 2.

a_1	=	1
a_2	=	1
a_3	=	1
b_1	=	1
b_2	=	1
α_1	=	1
α_2	=	1

Tablica 2: Współczynniki funkcji heurystycznej.

Wnioski

Z racji braku obecności eksperta przy implementacji heurystyki, algorytm mimo stosunkowo wysokiego drzewa minimaksowego, nie jest trudny do pokonania (m.in. podchodzi prosto pod bicia, mając do wykonania lepsze ruchy).

Literatura

[1] `https://pl.wikipedia.org/wiki/Algorytm_min-max`

[2] `https://pl.wikipedia.org/wiki/Algorytm_alfa-beta`

[3] `http://sequoia.ict.pwr.wroc.pl/~witold/aiarr/2009_projekty/warcaby/`

[4] `http://www.warcaby.pl/index.php/kodeks-warcabowy/134-rozdzia-i-oficjalne-reguy-gry-w-warcaby`