Kamil Buczko
Michał Fijałkowski
Marcin Jancurevic

Dokumentacja. PSZT
Lines Of Action

Opis zadania

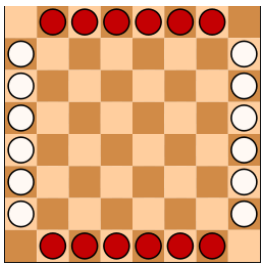
Napisać aplikację grającą w lines of action. Aplikacja powinna umożliwiać grę przeciwko sztucznym graczom. Głębokość drzewa dla sztucznych graczy powinna być parametrem aplikacji. Należy zapewnić prosty interfejs graficzny.

Cel projektu

Celem projektu było zaprojektowanie programu, dobrze grającego w grę Lines Of Action. Program został realizowany w Javie. Do testów użyliśmy narzędzia automatyzującego – Maven.

Zasady gry

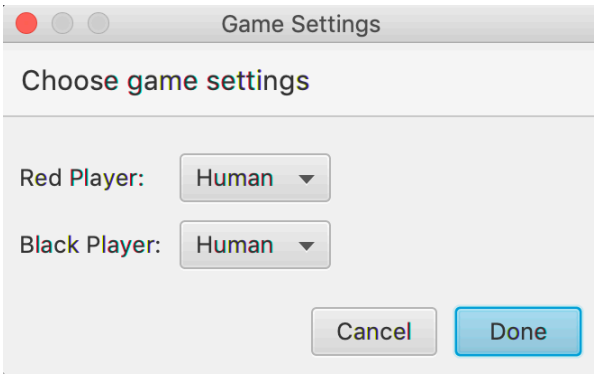
- Celem gry jest połączenie wszystkich swoich pionków w jedną grupę (pionek jest połączony z innym, gdy ma chociażby jednego sąsiada w jednym z 8 pól wokół siebie).
- Pionek się przesuwa w jednym z 8 kierunków, o ilość pól równą liczbie pionków obu kolorów stojących wzdłuż linii ruchu pionka.
- Pionek może przeskoczyć inny pionek tego samego koloru.
- Pionek nie może przeskakiwać pionka o innym kolorze, ale może „zjeść" go poprzez wykonanie ruchu na pionek o innym kolorze.
- Połączenie pionków w jeden łańcuch jest celem gry i powoduje zwycięstwo.
- W wypadku, gdy gracze podczas jednego ruchu połączą pionki w grupy, wygrywa gracz, który to zrobił pierwszy.
- W wypadku, gdy gracz nie ma ruchów, przegrywa grę.
- Pozycja startowa pionków:



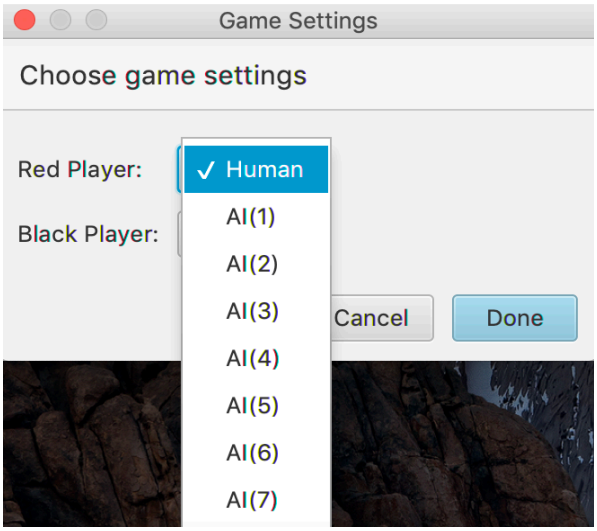
Plan działania

- Otrzymanie zadania.
- Spotkanie wstępne – zostały omówione wstępne założenia, podzielona praca i wybrana technologia, w której realizujemy projekt.
- Napisanie programu ze zwykłą planszą od szachów i dodanie do niej pionków.
- Stworzenie logiki gry – poruszanie się pionków, uwzględnienie założeń, które powstają z zasad gry.
- Stworzenie pełnej gry 2-osobowej.
- Dodanie funkcji heurystycznej – napisanie bota, „wytrenowanie" bota poprzez zmianę parametrów i śledzenie wyników gry dwóch botów ze sobą.
- Dodanie testów i stworzenie dokumentacji.
- Oddanie projektu.

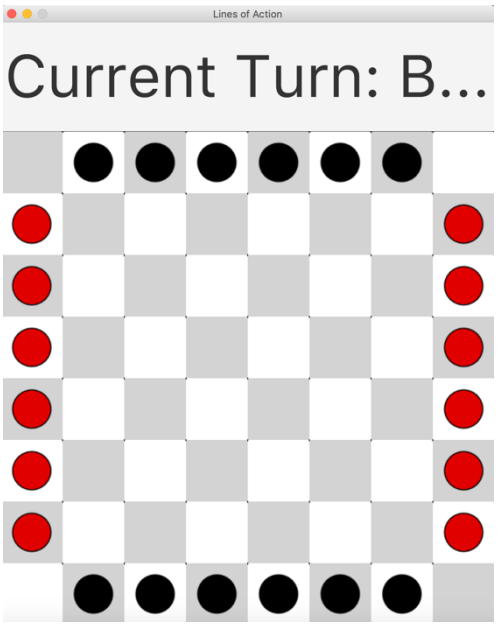
Elementy gry



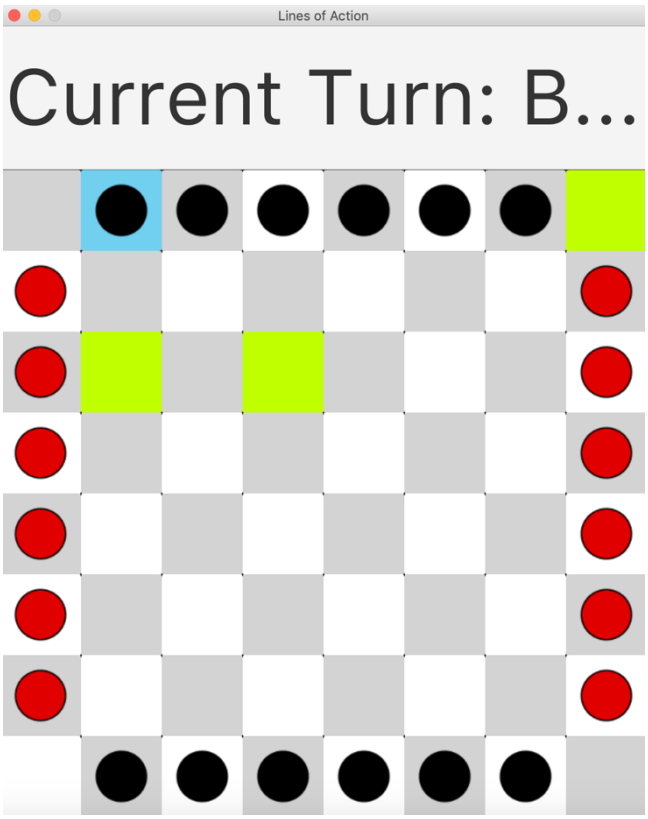
Picture 1 - Startowe MENU gry



Picture 2 - możliwe wybory startowego MENU



Picture 3 – Pozycja pionków przed rozpoczęciem gry



Picture 4 - Możliwość wyboru ruchu (w tym wypadku dla pionków czarnych)

Picture 5 - Zwycięstwo (w tym wypadku pionków czerwonych)

Funkcja Heurystyczna

Ważnym elementem funkcji jest to, by była obliczana w miarę szybko. Próbowaliśmy to osiągnąć za pomocą parametrów:

Centralizacja – to bardzo ważny aspekt, bo nie możemy przeskoczyć pionka przeciwnika. Z tego wynika, że pionki będące bliżej środka są trudniejsze do zablokowania przy zagrożeniu (sytuacja, w której gracz jest zmuszony rozbić formację swoich pionków, bo oponent może połączyć swoje pionki po jednym ruchu). Także centralizacja jest ważna, gdyż mając swoje pionki na środku planszy, mamy większą możliwość zablokowania pionka przeciwnika, innymi słowy – pionki, które dominują środek są ważniejsze. Oczywiste jest to, bo gra się rozpoczyna z pionkami, które stoją przy „końcach" planszy, dlatego też, jeśli chcemy zwyciężyć, musimy chociażby częścią pionków przejść przez środek. Pionki bliżej środku planszy są oceniani większą ilością punktów, niż te przy rogach, bo są trudniejsze do zablokowania. Ocenianie jest realizowane poprzez tablicę, do której dla każdego pola planszy zostały przypisane wartości „punktów". Centralizacja zmusza pionki dążyć na środek.

Każdemu polu planszy przypisujemy wartość w pliku konfiguracyjnym:

```
row0 = -2, -1, -1, -1, -1, -1, -1, -2
row1 = -1, 0 , 0 , 0 , 0 , 0 , 0 , -1
row2 = -1, 0 , 1 , 1 , 1 , 1 , 0 , -1
row3 = -1, 0 , 1 , 2 , 2 , 1 , 0 , -1
row4 = -1, 0 , 1 , 2 , 2 , 1 , 0 , -1
row5 = -1, 0 , 1 , 1 , 1 , 1 , 0 , -1
row6 = -1, 0 , 0 , 0 , 0 , 0 , 0 , -1
row7 = -2, -1, -1, -1, -1, -1, -1, -2
```

Na podstawie tych wartości mamy także zdefiniowane maksymalną sumę wartości pionków dla każdej ilości:

```
maxPosValue = 0, 2, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16
```

1. *Obliczenie sumy wartości teraźniejszych pozycji pionków*
2. *Odczytanie maksymalnej sumy pozycji pionków dla ilości pionków będących teraz na planszy*
3. *Dzielenie sumy z punktu 1 przez sumę z punktu 2*

waga_centralizacji = suma_wartości_pozycji_pionków / max_suma_pozycji_pionków;

Jednolitość – obliczenie jest bardzo proste, lecz przez ten parametr bardzo dobre możemy opisać sytuację na planszy. Jednolitość obliczamy jako pole najmniejszego prostokąta, który zawiera wszystkie pionki jednego koloru. Im to pole jest mniejsze, tym lepiej oceniamy tą sytuację, gdyż z tego wynika, że pionki są blisko siebie. Używamy ją po to, by unikać sytuacji, że większość pionków stoją w jednej grupie, a kilka pionków są na drugim końcu planszy.

1. *Obliczenie pola prostokąta zawierającego wszystkie pionki*
2. *Obliczenie minimalnego pola prostokąta zawierającego wszystkie pionki, co tak naprawdę jest równe ilości naszych pionków na planszy*
3. *Podzielenie pola z punktu 2 przez pole z punktu 1*

waga_jednolitości = minimalne_pole_prostokąta / teraźniejsze_pole_prostokąta

*Każdy prostokąt zawiera wszystkie nasze pionki

Środek masy – obliczany jest środek masy każdego pionka, a następnie, w zależności od pozycji pionka na planszy jest on oceniany. Im bliżej środka planszy, tym ocena jest lepsza. Parametr ten pomaga programowi zrozumieć, że lepsze są pozycje bliskie do pionków w tym samym kolorze. Nie znaczy to jednak, że jeśli środek masy jest na środku planszy, to pionki będą tam. Równie dobrze mogą być rozłożone na brzegach planszy.

W pliku konfiguracyjnym mamy zdefiniowaną minimalną sumę odległości dla każdej ilości pionków:

```
minDistanceSum = 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14
```

1. *Obliczanie sumy odległości pionków od środka dla teraźniejszego układy pionków*
2. *Odczytanie minimalnej możliwej sumy odległości pionków dla ilości pionków będących teraz na planszy*
3. *Waga środka masy jest obliczona przez odwrotność różnicy sumy z punktu 1 i sumy z punktu 2, odwracamy, bo im mniejsza jest wartość wagi środka masy, tym pionki są bliżej środka masy*

waga_środka_masy = 1 / (suma_odległości_pionków – min_suma_odległości_pionków);

Wniosek

W takiej grze jak LOA zbudowanie funkcji heurystycznej nie jest prostym zadaniem. Zwykłe porównanie ilości grup spójnych na planszy nie jest najlepszym rozwiązaniem, gdyż takie programy przegrywają z naszym. W porównaniu do jednego z najlepszych programów grających w LOA – MIA nasz funkcja heurystyczna działa wolniej, chociaż MIA ma tą funkcję rozbudowaną o wiele więcej. Kluczowym elementem w takim programie jest dobranie poprawnych parametrów do gry, w czym mogłoby pomóc uczenie maszynowe.