PSZT
Artur Tkaczyk
Kajetan Kaczmarek

Projekt WK.GD.8
Kółko i krzyżyk
Nieograniczona plansza, wygrywa 5 sąsiednich

I. Kluczowe decyzje projektowe
   a) Zdecydowaliśmy się podzielić projekt na moduły aby ułatwić obsługę i analizę
   b) Z uwagi na preferencje osobiste projekt został przez nas napisany w języku JAVA
   c) Z ww. powodów do obsługi warstwy graficznej użyliśmy pakietu AWT
   d) Jeżeli chodzi o samą rozgrywkę przyjęliśmy że pierwszy ruch zawsze należy do gracza tak aby ułatwić testowanie i łatwiej stawiać AI w trudnych sytuacjach
   e) Początkowo standardowy rozmiar planszy miał wynosić 20x20 a głębokość drzew po 5

II. Instrukcja uruchomienia
   a) Po uruchomieniu programu wyświetla się ekran startowy.Możemy na nim wybrać, od góry:
      1) Wysokość oraz szerokość pola gry
      2) Długość zwycięskiego ciągu
      3) Ilość poziomów drzew każdego ze sztucznych graczy
      4) Tryb gry – Gracz przeciwko AI lub AI przeciwko AI
   b) W przypadku gry z udziałem gracza ma ekranie wyświetla się pole gry.Po naciśnięciu na dowolną kratkę zostaje tam wstawiony znak gracza, a po chwili wyświetlana jest odpowiedź AI.Po spełnieniu warunku zwycięstwa gra wyświetla komunikat o przegranej lub wygranej
   c) W przypadku pojedynku dwóch AI raport z ich posunięć jest drukowany do terminala

III. Opis struktury programu
   a) Klasa View

      Klasa View implementuje elementy biblioteki AWT które odpowiadają za wyświetlanie elementów GUI gry.
   b) Klasa Controller

      Klasa Controller zawiera metody odpowiadające za obsługę przycisków w menu oraz samej gry.
   c) Klasa Start

      Wrapper do uruchomienia programu
   d) Klasa MinMax

      Klasa MinMax zawiera algorytm MiniMax.Konstruktor klasy wywołuje funkcje alfabeta, która implementuje algorytm MiniMax, przegląda stany oraz zwraca obiekt będący parą wybranego stanu gry oraz wartości funkcji przystosowania my odpowiadającej
   e) Klasa Pair

      Pozwala przekazywać jednocześnie stan gry oraz powiązaną z nim wartość funkcji przystosowania.
   f) Klasa Model

Klasa model zawiera ogólną logikę gry i pętlę w której zawiera się rozgrywka. Tworzymy w niej obiekty minmax to obsługi AI, oceniamy warunki zakończenia gry.

g) Klasa StanPlanszy

Klasa StanPlanszy służy do przechowywania stanu gry oraz do obsługi zapytań z nim związanych, tak jak czyja tura jest w danym momencie, czy gra została zakończona, oraz funkcja Przystosowania która ocenia wartość danego stanu gry z pounktu widzenia algorytmu MiniMax

IV. Raport z przeprowadzonych testów

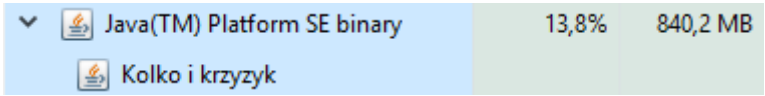
a) Przebieg gry

1) W grach z udziałem gracza AI wykazuje tendencję do maksymalizowania własnych zysków aż do zauważenia sytuacji które mogą przerodzić się w sytuacje patowe.W takim przypadku zmienia strategię z agresywnej na zachowawczą tak aby nie dopuścić do swojej porażki
2) W przypadku gry pomiędzy dwoma przeciwnikami AI ta sama sekwencja powtarza się w każdym przypadku :

- Ruch Gracza nr. 1 na pozycji 1 0
- Ruch Gracza nr. 2 na pozycji 0 2
- Ruch Gracza nr. 1 na pozycji 1 1
- Ruch Gracza nr. 2 na pozycji 0 3
- Ruch Gracza nr. 1 na pozycji 2 0
- Ruch Gracza nr. 2 na pozycji 0 4
- Ruch Gracza nr. 1 na pozycji 0 5
- Ruch Gracza nr. 2 na pozycji 1 2
- Ruch Gracza nr. 1 na pozycji 2 1
- Ruch Gracza nr. 2 na pozycji 1 3
- Ruch Gracza nr. 1 na pozycji 2 2
- Ruch Gracza nr. 2 na pozycji 1 4
- Ruch Gracza nr. 1 na pozycji 3 0
- Ruch Gracza nr. 2 na pozycji 4 0
- Ruch Gracza nr. 1 na pozycji 3 1
- Ruch Gracza nr. 2 na pozycji 2 3
- Ruch Gracza nr. 1 na pozycji 3 2
- Ruch Gracza nr. 2 na pozycji 2 4
- Ruch Gracza nr. 1 na pozycji 3 3
- Ruch Gracza nr. 2 na pozycji 3 4
- Ruch Gracza nr. 1 na pozycji 4 4

Jest to zrozumiałe jako że funkcja przystosowania oblicza te same wartości dla kolejnych pól, w związku z czym sekwencja musi się powtarzać.Jeżeli uznamy to za wadę projektu rozwiązaniem może być uwzgędnienie losowej wartości funkcji przystosowania przy genracji pierwszego stanu tak aby pierwszy ruch był zróżnicowany, co powinno kaskadowo zmienić przebieg całej rozgrywki. Podobnie, element losowy można by uwzględniać po prostu przy każdym ruchu tak aby ruchy nie zawsze były optymalne aby zróżnicować rozrywkę

b) Jeżeli chodzi o wydajność algorytmu, założona na początku wielkość planszy 20x20 oraz głębokość drzew po 5 dla obydwu AI okazały się zbyt wymagające w naszej implementacji, przynajmniej jak na warunki naszych komputerów



W związku z tym w dalszych testach zmieniliśmy standardowy rozmiar planszy na 8x8 i głębokość drzew na 3 tak aby ułatwić testowanie. Parametry te dalej mogą być dostosowane ręcznie

V.Testy z pomiarem czasu

1. Dla gier z udziałem gracza pomiar czasu ma mało sensu jako że główną składową będzie czas reakcji człowieka
2. Poniżej przedstawiamy tabelę ze zmierzonym czasem gry dla dwóch głębokości drzew dwóch komputerów oraz wynik
3. Czas wykonania w sekundach

	AI nr. 2					
AI nr. 1	Gł. drzew	1	2	3	4	5
	1	0.01	0.01	0.06	0.77	4.12
	2	0.01	0.07	0.06	0.56	4.07
	3	0.19	0.13	0.64	0.97	15.41
	4	0.42	0.42	0.48	0.137	0.445
	5	3.05	3.05	7.57	3.75	24.30

4. Wygrany

	AI nr. 2					
AI nr. 1	Gł. drzew	1	2	3	4	5
	1	2	2	2	2	2
	2	2	2	2	2	2
	3	1	1	1	1	1
	4	2	2	2	2	2
	5	1	1	2	1	1

Rezultaty pokazują że pomimo różnych głębokości drzewa AI z płytszym drzewem także jest w stanie wygrać, z powodu przewagi strategii reaktywnej.

VI. Wnioski dotyczące rezultatów

a) Dla prostej gry jak kółko i krzyżyk algorytm wydaje się pracować optymalnie.
b) Głębokość drzew drastycznie wpływa na wydajność – zmiana o 1 poziom drastycznie wydłuża czas wykonywania się algorytmu MINIMAX