

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Temat: Gry & AI.

Nazwisko i Imię prowadzącego kurs: mgr inż. Marta Emirsajłow

Wykonawca:	
Imię i Nazwisko	Jakub Kolasa
Nr indeksu, wydział	249012, W4
Termin zajęć	Pt, 13:15-15:00
Data oddania sprawozdania	29.05.2020

1. Wprowadzenie.

Realizowaną grą będzie kółko i krzyżyk. Będzie ona posiadała możliwość wyboru rozmiaru kwadratowej planszy, długości linii wymaganej do zwycięstwa, oraz wybór między sterowaniem graczem samodzielnie, lub przy pomocy AI.

Sztuczna Inteligencja jest sterowana przy pomocy dwóch algorytmów. Pierwszym z nich jest algorytm *alpha-beta*, będący wariantem algorytmu *min-max*. Jego przewagą nad wersją podstawową jest wcześniejsze odcinanie gałęzi drzewa przeszukiwania, które już na wczesnym etapie są gorsze od pozostałych.

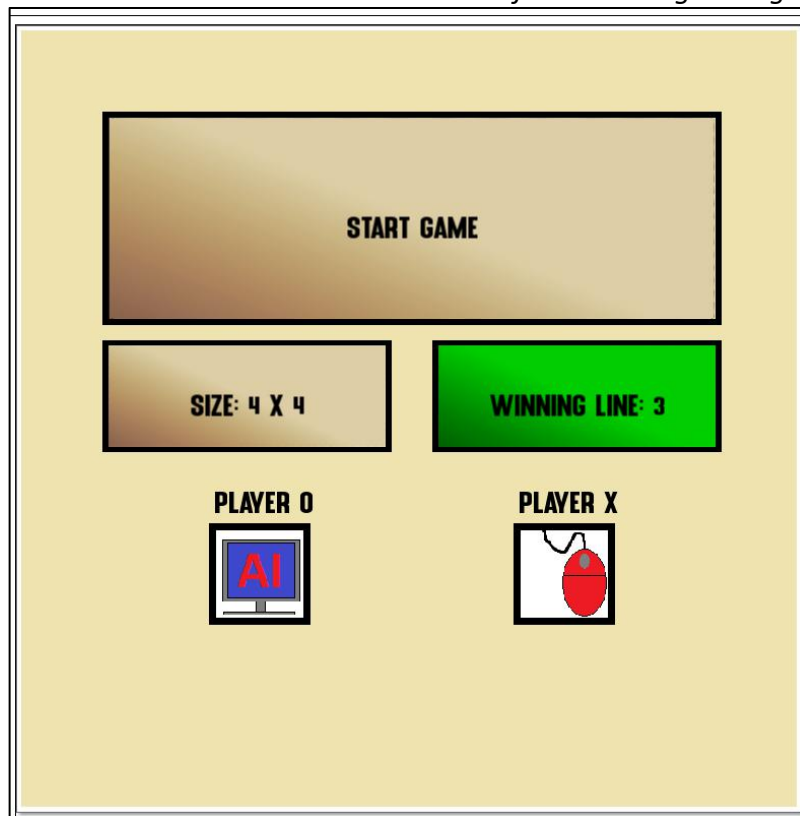
Jako, że gra umożliwia przeprowadzenie rozgrywki na planszy 15x15 stworzono drugi algorytm. Jego działanie opiera się na punktowaniu każdego pola na planszy, a następnie wybraniu takiego, który będzie najlepszą opcją dla gracza i jednocześnie zajęcie go najbardziej zaszkodzi przeciwnikowi.










Na podstawie prób i błędów zaobserwowano, że algorytm *alpha-beta* działa wydajnie dla głębokości równej 12. AI korzysta z prostszego algorytmu do momentu w którym na planszy nie znajdzie się więcej niż 12 pustych pól (*wtedy algorytm alpha-beta zyskuje największą skuteczność i rozsądny czas trwania*).

2. Opis programu.

Do napisania gry wykorzystano język C++ oraz bibliotekę SFML. Stworzono klasę *button*, a następnie przy jej pomocy stworzono interfejs oraz planszę gry. Cała gra znajduje się w ciele klasy *OandX*.

Interface menu głównego.



A 7x7 tic-tac-toe grid is shown. The grid contains several red circles (Player O) and blue crosses (Player X). A green square highlights the cell at row 4, column 6, which contains a red circle. This cell is part of a diagonal sequence of blue crosses from (row 2, col 4) to (row 6, col 8), indicating a win for Player X.

	X	O	X	O	X	
	X	O	X	O	X	
	X	O	X	O	X	
	X	O	X	O	X	
	X	O	X	O	X	

3. Algorytm *alpha-beta*.

Działanie algorytmu polega na minimalizowaniu strat oraz maksymalizacji zysków. Algorytm symuluje rozgrywkę badając jej wszystkie możliwe zakończenia. Następnie ocenia, każdy możliwy ruch na podstawie jego możliwych zakończeń. Przyjęto, że zwycięstwo, porażka oraz remis to kolejno: +1, -1, 0 punktów.

Wariant *alpha-beta* przestaje obliczać przypadki niekorzystne dla gracza znacznie przyspieszając działanie algorytmu.

4. Wnioski.

Czas pracy algorytmu *alpha-beta* wzrasta bardzo szybko wraz z rozmiarem tablicy, oraz głębokością przeszukiwania możliwych zakończeń. Zastosowanie algorytmu usprawniającego początkową rozgrywkę na planszach o większych rozmiarach znacznie poprawia, a wręcz umożliwia dalszą rozgrywkę.

Poprawa czasu działania algorytmu *alpha-beta* poprzez zmniejszanie głębokości przeszukiwania jest bezcelowa. Wykonywane przez algorytm ruchy są niedokładne, a czasem nawet losowe.

5. Źródła.

https://pl.wikipedia.org/wiki/Algorytm_min-max

https://pl.wikipedia.org/wiki/Algorytm_alfa-beta

<https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/>

<https://www.sfml-dev.org/index.php>