Projektowanie algorytmów i metody sztucznej inteligencji

Projekt 3

Termin zajęć: Wykonał:

Czwartek 9:15 Eryk Matecki 249484

Prowadzący: dr inż. Łukasz Jeleń

1. Wstęp

Projekt polegał na stworzeniu gry wykorzystującej swego rodzaju algorytm sztucznej inteligencji. W tym projekcie gra została wyposażona w prostą wersję graficzną. W aplikacji został wykorzystany algorytm Minimax. Ten algorytm jest stosowany w prostych grach logicznych do wyznaczania optymalnych ruchów. Jest algorytmem rekurencyjnym, który wywołuje sam siebie do analizy kolejnych ruchów w grze. Algorytm Minimax jest to metoda minimalizowania maksymalnych możliwych strat. Alternatywnie można je traktować jako maksymalizację minimalnego zysku. Wywodzi się to z teorii gry o sumie zerowej, obejmujących oba przypadki, zarówno ten, gdzie gracze wykonują ruchy naprzemiennie, jak i ten, gdzie wykonują ruchy jednocześnie. Algorytm minimax jest drzewowym algorytmem rekurencyjnym polegającym na minimalizowaniu/maksymalizowaniu strat. W jego postaci „kółko-krzyżykowej" rozważany jest aktualny stan planszy i symulacyjnie przewidywane są ruchy graczy w każdym możliwym przebiegu zdarzeń. Na podstawie założenia o potrzebnej ilości figur do uzyskania zwycięstwa wyciąga się odpowiednie wnioski czy dany ruch jest opłacalny. W projekcie wykorzystano algorytm Alfa-Beta. Jest to algorytm przeszukujący, redukujący liczbę węzłów, które muszą być rozwiązywane w drzewach przeszukujących przez algorytm min-max. Warunkiem stopu jest znalezienie przynajmniej jednego rozwiązania czyniącego obecnie badaną opcję ruchu gorszą od poprzednich opcji. Wybranie takiej opcji nie przyniosłoby korzyści graczowi ruszającemu się, dlatego nie ma potrzeby przeszukiwać dalej gałęzi drzewa tej opcji. Ta technika pozwala zaoszczędzić czas przeszukiwania bez zmiany wyniku działania algorytmu.

Zapis w pseudokodzie:

```
function minimax(position, depth, alpha, beta, maximizingPlayer)
    if depth == 0 or game over in position
        return static evaluation of position

    if maximizingPlayer
        maxEval = -infinity
        for each child of position
            eval = minimax(child, depth - 1, alpha, beta false)
            maxEval = max(maxEval, eval)
            alpha = max(alpha, eval)
            if beta <= alpha
                break
        return maxEval

    else
        minEval = +infinity
        for each child of position
            eval = minimax(child, depth - 1, alpha, beta true)
            minEval = min(minEval, eval)
            beta = min(beta, eval)
            if beta <= alpha
                break
        return minEval
```

2. Opis gry

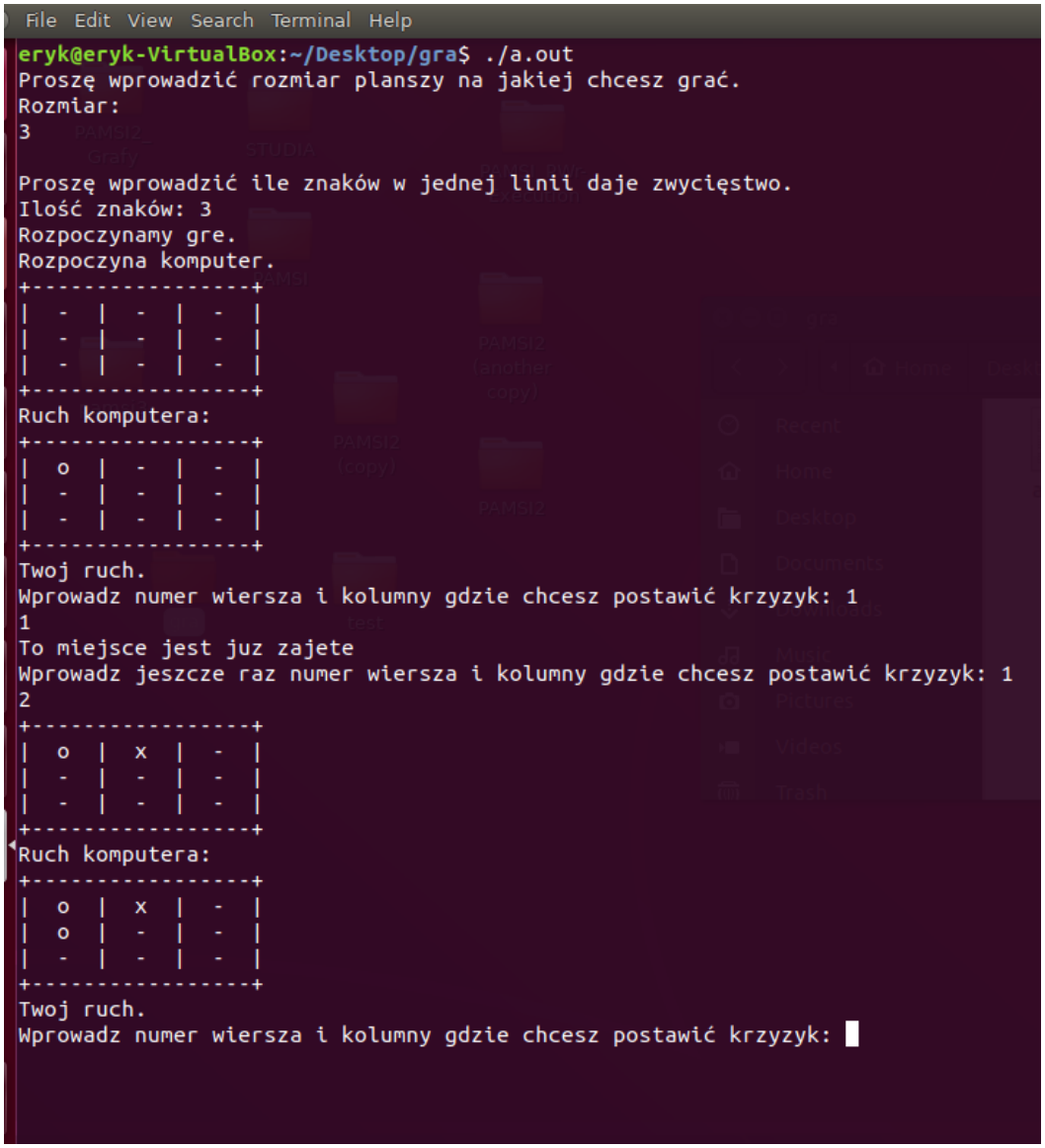
Gra została zaimplementowana przy użyciu języka C++. Utworzony został prosty interfejs graficzny użytkownika pojawiający się w terminalu. Na początku program prosi o podanie jaką wielkość planszy chcemy mieć, np. rozmiar 3 oznacza planszę o wymiarach 3x3, rozmiar 4 planszę 4x4 itd.. Następnie należy podać ile znaków w jednej linii oznaczać będzie wygranie. Zostało stworzone zabezpieczenie uniemożliwiające wybranie ilości znaków w linii potrzebnych do wygrania przekraczającej rozmiary planszy oraz ww. ilość musi być większa lub równa 2, ponieważ gdyby wygrywał pojedynczy znak, gra nie miałaby żadnego sensu. Po wybraniu rozmiarów planszy i ilości wygrywających znaków, rozpoczyna się gra. Na początku zostaje wylosowany uczestnik rozpoczynający, osoba fizyczna lub komputer. Na samym początku plansza wypełniona jest „-„ co oznacza, że w żadnym polu nie znajduję się jeszcze żaden znak, plansza jest pusta. Następnie podajemy współrzędne, w których ma znaleźć się nasz ruch. Najpierw podajemy liczbę odpowiadającą wierszom, a następnie kolumnom. Ruchom komputera przypisane są „o”, a ruchom użytkownika „x”. W zależności od wyniku potyczki, wyświetlane są komunikaty o zwycięstwie, porażce lub remisie użytkownika. Przykładowa gra na planszy o wymiarach 4x4 zakończona remisem prezentuje się na „screenach” poniżej.

```
File  Edit  View  Search  Terminal  Help
eryk@eryk-VirtualBox:~/Desktop/gra$ ./a.out
Proszę wprowadzić rozmiar planszy na jakiej chcesz grać.
Rozmiar: 4

Proszę wprowadzić ile znaków w jednej linii daje zwycięstwo.
Ilość znaków: 4
Rozpoczynamy gre.
Rozpoczyna gracz.
+-----------------------+
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 1
1
+-----------------------+
|  x  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
+-----------------------+
Ruch komputera:
+-----------------------+
|  x  |  o  |  -  |  -  |
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 2
2
+-----------------------+
|  x  |  o  |  -  |  -  |
|  -  |  x  |  -  |  -  |
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
+-----------------------+
Ruch komputera:
+-----------------------+
|  x  |  o  |  o  |  -  |
|  -  |  x  |  -  |  -  |
|  -  |  -  |  -  |  -  |
|  -  |  -  |  -  |  -  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 3
3
+-----------------------+
|  x  |  o  |  o  |  -  |
|  -  |  x  |  -  |  -  |
|  -  |  -  |  x  |  -  |
|  -  |  -  |  -  |  -  |
+-----------------------+
Ruch komputera:
+-----------------------+
|  x  |  o  |  o  |  -  |
|  -  |  x  |  -  |  -  |
|  -  |  -  |  x  |  -  |
|  -  |  -  |  -  |  o  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 2
1
+-----------------------+
|  x  |  o  |  o  |  -  |
|  x  |  x  |  -  |  -  |
|  -  |  -  |  x  |  -  |
|  -  |  -  |  -  |  o  |
+-----------------------+
Ruch komputera:
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  -  |  -  |
|  -  |  -  |  x  |  -  |
|  -  |  -  |  -  |  o  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 3
1
+-----------------------+
```

```
File  Edit  View  Search  Terminal  Help
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 3
1
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  -  |  -  |
|  x  |  -  |  x  |  -  |
|  -  |  -  |  -  |  o  |
+-----------------------+
Ruch komputera:
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  -  |  -  |
|  x  |  -  |  x  |  -  |
|  o  |  -  |  -  |  o  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 2
3
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  x  |  -  |
|  x  |  -  |  x  |  -  |
|  o  |  -  |  -  |  o  |
+-----------------------+
Ruch komputera:
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  x  |  o  |
|  x  |  -  |  x  |  -  |
|  o  |  -  |  -  |  o  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 3
4
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  x  |  o  |
|  x  |  -  |  x  |  x  |
|  o  |  -  |  -  |  o  |
+-----------------------+
Ruch komputera:
```

```
File  Edit  View  Search  Terminal  Help
Ruch komputera:
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  x  |  o  |
|  x  |  -  |  x  |  -  |
|  o  |  -  |  -  |  o  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 3
4
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  x  |  o  |
|  x  |  -  |  x  |  x  |
|  o  |  -  |  -  |  o  |
+-----------------------+
Ruch komputera:
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  x  |  o  |
|  x  |  o  |  x  |  x  |
|  o  |  -  |  -  |  o  |
+-----------------------+
Twoj ruch.
Wprowadz numer wiersza i kolumny gdzie chcesz postawić krzyzyk: 4
2
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  x  |  o  |
|  x  |  o  |  x  |  x  |
|  o  |  x  |  -  |  o  |
+-----------------------+
Ruch komputera:
+-----------------------+
|  x  |  o  |  o  |  o  |
|  x  |  x  |  x  |  o  |
|  x  |  o  |  x  |  x  |
|  o  |  x  |  o  |  o  |
+-----------------------+

KONIEC GRY. Padł remis!!
eryk@eryk-VirtualBox:~/Desktop/gra$
```

Teraz zaprezentuję przykładową grę na planszy o rozmiarach 3x3, w której zaczynał komputer i użytkownik wykonał niedozwolony ruch.



Widać, że użytkownik chciał postawić krzyżyk w miejscu już zajętym. Program mu na to nie pozwolił i poprosił o ponowne wybranie miejsca, w którym ma umieścić znak.

3. Wnioski

- Pozytywny wpływ na działanie algorytmu wybierającego najlepszy ruch miało dodanie algorytmu alfa-beta.
- Gdy rozmiar planszy jest większy od 3 to algorytm działa nieco gorzej, ponieważ bo zostało użyte ograniczenie głębokości w celu lepszego działania dla standardowego rozmiaru gry kółko i kryżyk.

- Zastosowany algorytm alfa-beta to wersja mini-maxa, która odcina niektóre przeszukiwane gałęzie drzewa. Co za tym idzie czas przeszukiwania zostaje ograniczony do najbardziej optymalnych poddrzew. Dla większych rozmiarów plansz algorytm potrzebuje sporo czas, aby podjąć decyzję, który ruch będzie najlepszy. Rozwiązaniem tego problemu mogłoby być podłączenie bazy danych zawierającej możliwe ustawienia wraz z oceną, co eliminowałoby potrzebę przeszukiwania drzew.
- Program nagrywający przysporzył mi trochę problemów przez to, że ucinał mi nagrywanie w losowych momentach. Z tego powodu zamieszczam dwa krótkie filmiki. Na jednym z nich widać kompilacje programu, wybranie rozmiaru planszy i ilości wygrywających znaków. Następnie rozpoczyna komputer i widać uniemożliwienie użytkownikowi wybrania zajętego już miejsca na planszy. Na drugim filmie widać przykładową rozgrywkę zakończoną porażką użytkownika, co zostało potwierdzone odpowiednim komunikatem na ekranie.

Bibliografia:

- https://pastebin.com/rZg1Mz9G?fbclid=IwAR0ERE_XWY8xe2U3KAQf1S1MFVI4SxsPQIaG6AaOLdDtfxLZfwlVnDfThvw
- https://pl.wikipedia.org/wiki/Algorytm_alfa-beta
- https://www.youtube.com/watch?v=l-hh51ncgDI