15.05.2020r.

Emilia Starczyk
249005

PROJEKTOWANIE ALGORYTMÓW I METODY SZTUCZNEJ INTELIGENCJI

Projekt 3

Dr inż. Łukasz Jeleń

Czwartek 9.15 - 11.00

1. Wstęp

Tematem projektu było stworzenie gry, która korzysta z technik Sztucznej Inteligencji. Do wybrania było kilka różnych gier. Spośród tych na ocenę bardzo dobrą wybrałam grę kółko i krzyżyk, gdyż jej modyfikacja była dość nietypowa i rzadko spotykana. Do zaimplementowania gry wykorzystałam algorytm cięć Alfa-Beta, który jest modyfikacją algorytmu MiniMax. Oba algorytmy do swojego działania wymagają funkcji, która będzie w jakiś sposób oceniała sytuację, dla komputera, jak i dla użytkownika i przyznawała odpowiednią ilość punktów.

2. Opis algorytmu cięć Alfa-Beta

Algorytm cięć Alfa-Beta to algorytm przeszukujący oraz redukujący liczbę węzłów drzewa, które muszą zostać rozwiązywane. Jest to algorytm wykorzystywany w grach dwuosobowych, np. szachy, go, czy kółko i krzyżyk. Algorytm cięć Alfa-Beta jest modyfikacją algorytmu MiniMax – bazuje na węzłach MIN oraz MAX i skraca jego działanie poprzez odcinanie węzłów, o których wiemy, że nie zmienią ostatecznej decyzji o ruchu komputera. Algorytm bazuje na odpowiedniej ocenie/punktacji kolejnych ruchów graczy. Zgodnie z nazwą algorytmu wykorzystuje on dwa rodzaje cięć:

a) cięcia Alfa – wybranie najlepszej wartości MAX, która przy dalszej analizie nie ulegnie zmianie, odcina dalszą analizę MIN,
b) cięcia Beta – wybranie najlepszej wartości MIN, która przy dalszej analizie nie ulegnie zmianie, odcina dalszą analizę MAX.

Odcięcie odbywa się w momencie. gdy wartość alfa w węźle jest mniejsza bądź równa wartości beta. Pseudokod wykorzystanego algorytmu:

```
/* max – true, gdy węzeł typu MAX, false, gdy węzeł typu MIN */
MiniMax(stanGry, głębokość, alfa, beta, max){
punkty=Ewaluacja(stanGry); //ocena wartości, 10 dla wygranej komputera, -10 dla gracza, 0 remis
jeśli punkty ==  10 to
        zwróć punkty - głębokość
jeśli punkty == -10 to
        zwróć punkty + głębokość
jeśli nie ma już możliwych ruchów lub głębokość == 2*rozmiar – ilość znaków w linii + 2 to
        zwróć zero
jeśli węzeł jest typu MAX to
        maxOcena = -∞
        dla każdego możliwego ruchu
                ocena = MiniMax(nowyStanGry, głębokość++, alfa, beta, false)
                maxOcena = max(ocena,maxOcena)
                alfa = max(alfa,maxOcena)
                jeśli alfa >= beta to
                        przerwij
        zwróć maxOcena
w przeciwnym razie wykonaj
        minOcena = ∞
        dla każdego możliwego ruchu
                ocena = MiniMax(nowyStanGry, głębokość++, alfa, beta, true)
                minOcena = min(ocena,minOcena)
                beta = min(beta,minOcena)
                jeśli alfa >= beta to
                        przerwij
        zwróć minOcena
}
```

3. Przebieg badań

Przygotowanie programu zaczęłam od napisania klasy, która reprezentowała by planszę do gry w kółko i krzyżyk. Napisałam metody sprawdzania planszy, najpierw dla przypadku planszy 3x3, a następnie zmodyfikowałam je dla przypadku ogólnego. Napisałam również metodę pozwalającą sprawdzić, czy zostały jeszcze jakieś możliwe ruchy do wykonania oraz metodę zwracającą informację, czyj ruch ma zostać właśnie wykonany. Dodałam również metodę wyświetlającą planszę, która w ostatecznej wersji programu nie jest już wykorzystywana. W kolejnym kroku zaimplementowałam klasę gracza. Dodałam podstawowe metody pozwalające ustawić oraz zwrócić znak. jakim gra gracz. Dodałam metody wybierania ruchu dla zwykłego użytkownika, jak i dla komputera. Zaimplementowałam algorytm MiniMax, który jest wykorzystywany w momencie, kiedy na planszy są już jakieś znaki. Jeśli grę rozpoczyna komputer, to wybiera on jedno pole spośród 4 narożników planszy. Dodałam również metodę zwracającą znak przeciwnika. Następnie zaimplementowałam klasę definiującą grę. Gra składa się z dwóch użytkowników oraz planszy. Zaimplementowałam metody ruchu gracza oraz ruchu komputera, które w fazie testów gry nie były używane. Dodatkowo zaimplementowałam metodę sprawdzającą, czy są jeszcze jakieś ruchy – metoda ta wykorzystuje metodę z planszy. Dodałam funkcje zwracające znaki graczy oraz wyświetlającą grę. W trakcie testów w konsoli potrzebna była metoda symulująca całą grę, jednak w ostatecznej wersji się nie znalazła. Po upewnieniu się, że metody działają poprawnie, przystąpiłam do pisania wersji graficznej aplikacji. W tym celu pobrałam bibliotekę *wxWidgets* z serwisu GitHub. Jest to darmowa biblioteka o otwartym kodzie źródłowym, która pozwala na tworzenie graficznych wersji aplikacji. Aby móc tworzyć graficzne elementy, trzeba było skompilować odpowiedni projekt oraz dołączyć odpowiednie biblioteki do projektu. Wymagane było też dodanie nowej zmiennej środowiskowej systemu. Po odpowiednim przygotowaniu środowiska, zaimplementowałam przyciski, listy wyboru oraz okna wpisywania w oknie głównym. Następnie zależnie od tego, co użytkownik wybierze, zaimplementowałam odpowiednia planszę oraz utworzyłam rozgrywkę. Do ostatecznej wersji dodałam tło oraz grafiki, co wymagało dodania pliku zasobów. Grę oddałam do testów w mojej rodzinie.

4. Opis aplikacji:

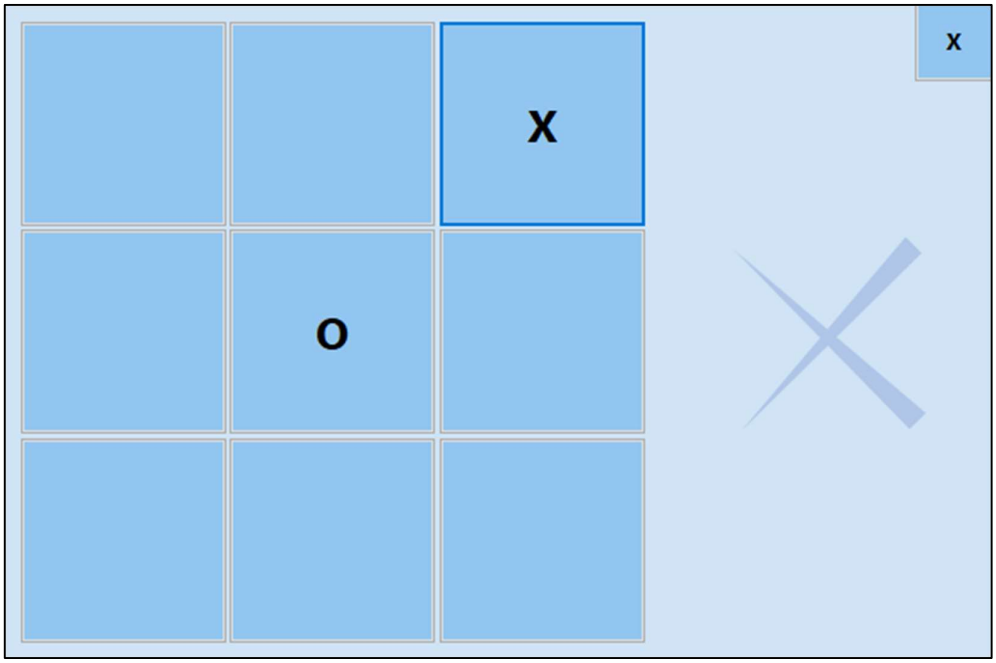
Aplikacja jest jednookienkowa, jedynie komunikaty o wyniku gry wyświetlane są jako osobne powiadomienia - pozwala to na zablokowanie użytkownikowi możliwości modyfikowania planszy i kontynuowania gry. Po przejściu do gry, ekran jest czyszczony z menu oraz tworzona jest plansza do gry.

a) Menu główne – składa się z 2 przycisków, 3 list wyboru oraz pola wpisywania.
   Od góry:
   W pierwsze okno użytkownik może wpisać swoja nazwę, która jest wykorzystywana później przy wyświetlaniu informacji o przegranej lub wygranej użytkownika.
   W polu „Kto zaczyna grę:” użytkownik może zdefiniować, czy on sam rozpoczyna grę, czy też komputer.
   W następnym polu można zdefiniować rozmiar planszy od 3x3 do 7x7. W zależności od wybranego rozmiaru modyfikowana jest lista „Liczba znaków w

linii:”. Lista ta pozwala wybrać liczbę znaków w jednej linii, która pozwala wygrać. Domyślnie liczba ta jest ustawiana na liczbę znaków równą liczbie rzędów w planszy, np. jeśli plansza jest 4x4, to liczba znaków do wygrania jest ustawiana na 4 znaki.
Przycisk „Zacznij grę” pobiera wszystkie informacje z pól, czyści ekran z menu oraz tworzy odpowiednią planszę oraz grę.
Przycisk „X” – zamyka aplikację.

Rys. 1 Menu główne

b) Okno gry – składa się z przycisków ułożonych w odpowiedni sposób, tak aby przypominał on planszę do gry w kółko i krzyżyk. Przypisane są do nich odpowiednie zdarzenia. Po przyciśnięciu przycisku uruchamiana jest funkcja wstawiająca w dane pole znak gracza, sprawdzająca, czy w danym miejscu nie nastąpiła wygrana oraz uruchamiająca analizę ruchów dla komputera. Po zdarzeniu funkcja z przycisku jest usuwana, tak, aby gracz nie mógł już modyfikować danego pola. W oknie gry znajduje się też przycisk „X”, któremu przypisane jest inne zdarzenie niż w menu głównym. Po naciśnięciu przycisku zamykana jest tylko gra. W oknie gry wyświetlana jest również grafika, która jest odpowiednio modyfikowana, w zależności od tego, czyj ruch ma zostać obecnie wykonany.



Rys. 2 Okno gry

c) Powiadomienia – okienko z informacją o remisie, wygranej lub przegranej. Po naciśnięciu przycisku „OK" plansza jest usuwana i uruchamia się menu główne gry.

Rys. 3 Informacja o wyniku gry

Wszystkie grafiki użyte w aplikacji zostały stworzone na jej potrzeby w programie INKSCAPE.

5. Podsumowanie i wnioski

Podczas testowania gry, napotkałam kilka problemów. Głównym z nich był przypadek gry, gdy plansza była danego rozmiaru, a liczba znaków do wygrania danej partii była mniejsza niż wybrany rozmiar. Przy dłuższej analizie wszystkich problemów doszłam do kilku wniosków:

- W wyżej opisanej sytuacji w głównej mierze wygrywa ten, który zaczął daną partię. Dzieje się tak, ponieważ mniejsza ilość znaków do wygrania, prowadzi do stworzenia kilku możliwych sposobów na wygranie, których komputer ani gracz nie jest w stanie w żaden sposób zablokować,
- Aby uniknąć przegrywania w takiej sytuacji, komputer mógłby zawsze zaczynać, ale jest to niezgodne z zasadami,
- Największy sens ma gra na planszy 3x3,

- Przy grach większych niż 3x3 bot myśli znacznie dłużej, ponieważ ma więcej możliwości do przeanalizowania. Nawet jeśli zastosujemy algorytm z przekazywaną głębokością, to przy dużych planszach komputer będzie znacznie dłużej myślał niż człowiek,
- Algorytm cięć Alfa-Beta to lepsza wersja MiniMax, pozwala przyspieszyć proces uzyskiwania ruchu dla komputera.

6. Bibliografia

- Never stop building, https://www.neverstopbuilding.com/blog/minimax, data dostępu do strony, ostatni raz: 14.05.2020 r.
- http://wazniak.mimuw.edu.pl/index.php?title=Sztuczna_inteligencja/SI_Modu%C5%82_8_-_Gry_dwuosobowe, data dostępu do strony, ostatni raz: 14.05.2020 r.
- J. Smart, K. Hock, S. Csomor, *Cross-Platform GUI Programming with wxWidgets,* Pearson Education, Inc.,2006.
- Piotr Wróblewski, *Algorytmy, struktury danych i techniki programowania. Wydanie V,* Helion, 2015.