Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 3

Gry & AI

Michał Maćkowiak 249464

26.05.2020

Prowadzący:

dr hab. inż. Andrzej Rusiecki

Zajęcia:

Wtorek godz. 13.15

1. Kółko i krzyżyk z AI

W moim projekcie zdecydowałem się wykonać popularną grę kółko i krzyżyk. Gra jest wyposażona w sztuczną inteligencję, przeciwko, której można grać. By bot podejmował słuszne decyzję wykorzystałem algorytm min – max. Dodatkowo w samej grze jest możliwość zdefiniowania jak duża ma być plansza oraz ile znaków w rzędzie, kolumnie lub po przekątnej ma prowadzić do wygranej.

2. Algorytm min – max

Realizacja tego algorytmu w projekcie działa w następujący sposób:

wykonanie ruchu przez AI jest wywoływane. W funkcji każdy dany ruch AI oraz gracza dostaję odpowiednią wartość. Dodatnia oznacza, że dany ruch prowadzi do zwycięstwa bota, ujemna, wygranej użytkownika, a wartość 0 oznacza remis. Funkcja rekurencyjnie wykonuje się sprawdzając, każdy następny ruch. Wykonuje ten, który ma największą wartość podczas wykonywanych rekurencji.

Rekurencja jest ograniczona przez głębie rekurencji, „depth". Oznacza ona jak długo będzie funkcja wykonywać samą siebie, czyli jak daleko ma sprawdzać ruchy. Warunkiem stopu jest wartość 0, gdyż każde następne wykonanie jest z zmniejszoną wartością o 1. Jeżeli rekurencja ma za dużą wartość dla dużych tablic algorytm będzie sprawdzać wszystkie możliwości, co sprowadza się do długiego czasu pracy. Dlatego wartość rekurencji nie może być sztywno ustalona. Rozpatruje dwa przypadki:

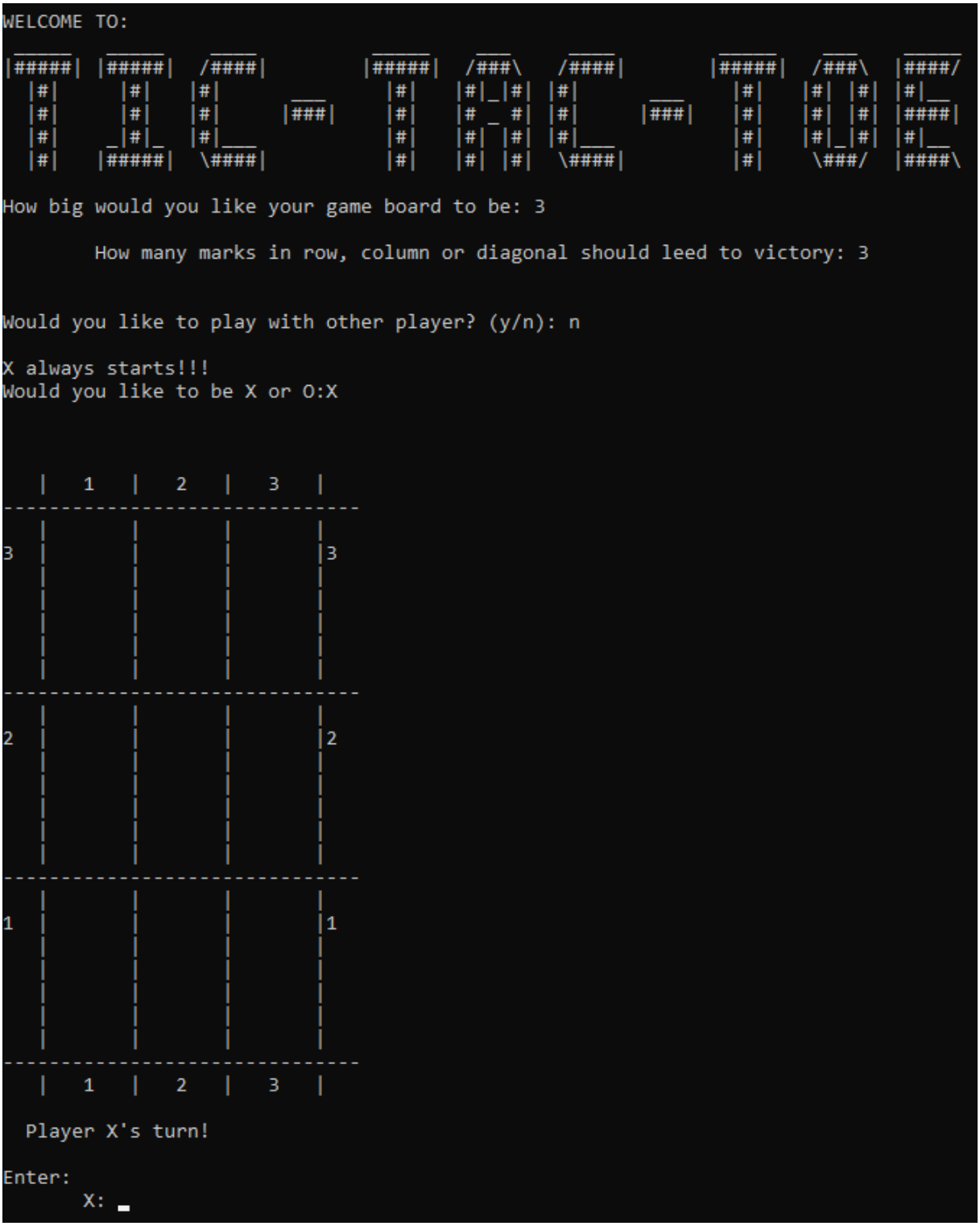
- **a.** Gra 3x3 – dla takiej rozgrywki wartość głębi może być bardzo wysoka. Jest to mała tablica więc sprawdzenie wszystkich możliwości nie jest czasochłonne. Na trybie „Release x64" w Visual Studio sprawdza w poniżej sekundę.
- **b.** Większe plansze – takie gry mogą trwać dłużej nie tylko przez zwiększoną liczbę pól. Stosując ten algorytm duża wartość głębi, mimo że podejmie najlepszą decyzję to będzie to czasochłonne. Najlepiej dopasowany czas działania oraz poprawność decyzji daje wartość głębi wyliczona ze wzoru:

$$\boldsymbol{depth} = \mathbf{1{,}4} \cdot \boldsymbol{rozmiar\ tablicy} - \boldsymbol{ilo}\text{ść } \boldsymbol{znak}\text{ów } \boldsymbol{w\ rz}\text{ędzie } \boldsymbol{do\ wygranej}$$

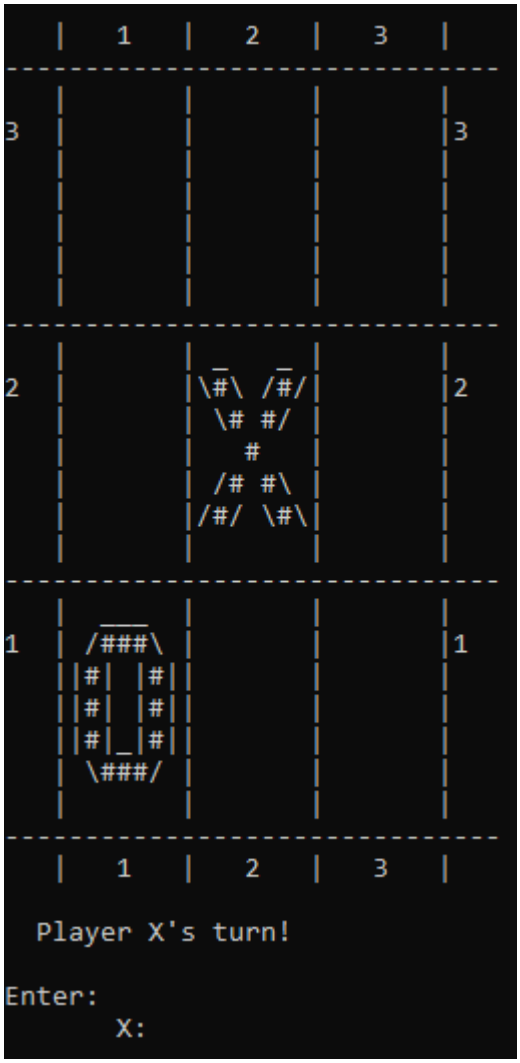
Jeżeli wartość będzie za mała to algorytm nie zdoła sprawdzić wszystkich przypadków wystarczająco daleko i będzie podejmował złe decyzje np. nie blokując gracza przed wygraną.

3. Działanie gry

Na początku program pyta o wielkość planszy, ilość znaków do wygranej oraz czy chcemy grać z AI czy z inną osobą. Wybierając AI decydujemy się na znak jakim gramy pamiętając, że „X" zawsze zaczyna. Warto zaznaczyć, że minimum to plansza 3x3 oraz 3 znaki w rzędzie. Wartości mniejsze nie mają sensu dla kółka i krzyżyk.



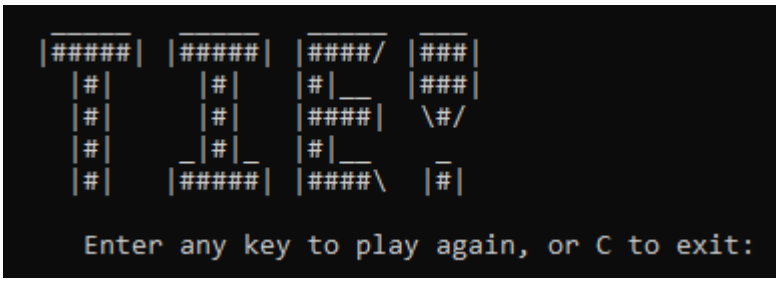
Następnie gra się zaczyna i podajemy współrzędne naszego ruchu.



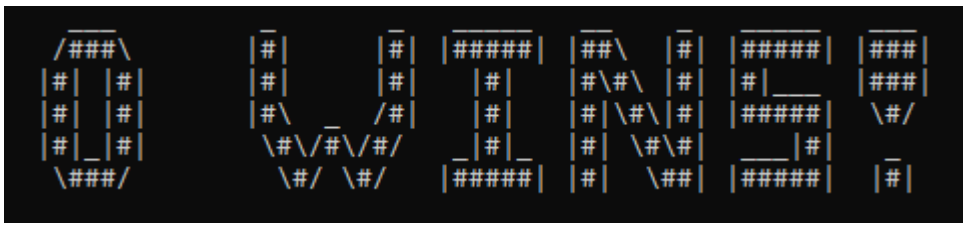
Na zdjęciu powyżej gracz wprowadził współrzędne X:2 i Y:2. „X” pojawił się na planszy oraz AI postawiło swój ruch.

Gra toczy się dalej, aż do trzesz przypadków:

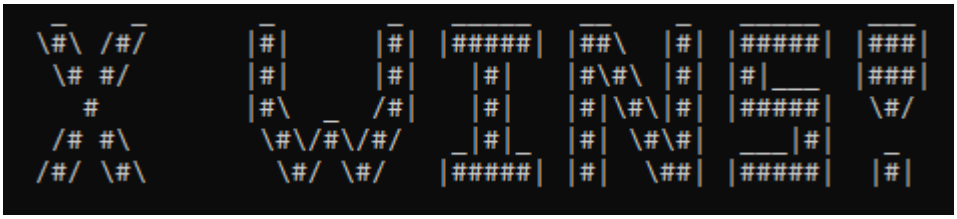
a. **remis,**



b. **wygrało O,**



c. **wygrało X.**



Po tym można rozpocząć grę na nowo lub wyjść.

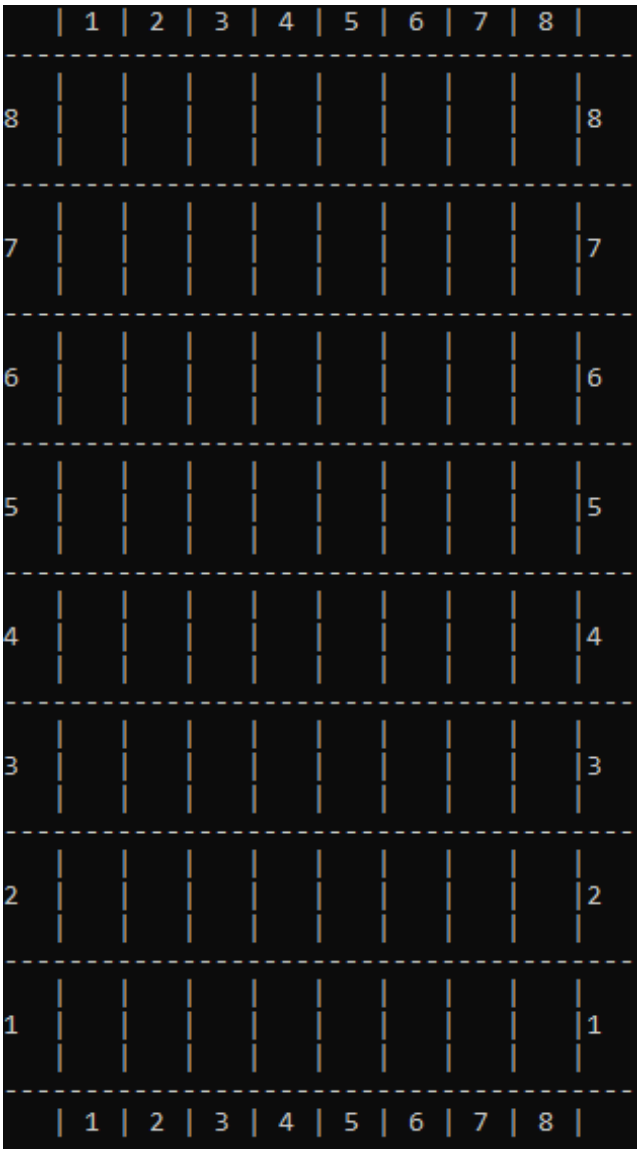
Przez graniczenia w miejscu na ekranie i w terminalu gra posiada dwa rozmiary planszy:

- mniejsze od 8,

- oraz większe lub równe 8.

Pierwsza opcja jest zaprezentowana powyżej na zdjęciu pierwszym. Znaki znajdują się tam w kwadracie 7x7 znaków.

Druga opcja posiada znaki w kwadracie 3x3.



4. Wnioski

- Dzięki zastosowaniu znacznie większej wartości głębi rekurencji dla planszy 3x3, AI nigdy nie przegrywa. Zawsze remisuje albo wygrywa. Wartość obliczona wzorem:

$$\boldsymbol{depth = 1{,}4 \cdot rozmiar\ tablicy - ilość\ znaków\ w\ rzędzie\ do\ wygranej}$$

jest za mała dla planszy 3x3.

- Współczynnik zastosowany we wzorze najlepiej pokrywał się z czasem decyzji oraz ze słusznością decyzji. Zwiększenie go powoduje dodatkowe wykonania rekurencji co wydłuża działanie algorytmu.

- Jeżeli rozmiar planszy jest stosunkowo większy niż ilość znaków do wygranej tym dłużej algorytm będzie sprawdzał dostępne ruchy.

- Gdy bot nie ma możliwości zablokowania np. w sytuacji, gdzie z więcej niż z jednej strony jest możliwa wygrana bot postanawia położyć swój marker nie próbując blokować. Wynika to z tego, że wie o przegranej, która oznacza dla niego wartość ujemną. Zremisowanie to wartość zero i wybiera opcje bardziej korzystną.

- Esencjalne było zdefiniowanie sprawdzenia czy ktoś wygrał na wszystkich płaszczyznach zwłaszcza na przekątnych.