"Kołko i krzyżyk" jako zastosowanie algorytmu minimax

Krzysztof Górski 245079

Maj 2020

1 Wstęp

Celem projektu było zaimplementowanie "sztucznej inteligencji" w postaci algorytmu minmax. Minimax jest rodzajem algorytmu przeszukiwania, który jest wykorzystywany do podejmowania decyzji w teorii gier, w celu znalezienia optymalnego ruchu dla gracza, przy założeniu, że przeciwnik również gra optymalnie. Jest szeroko stosowany w grach turowych dla dwóch graczy, takich jak Kółko i krzyżyk, szachy, warcaby itp. Do rozwiazania i rozważania tego problemu zaimplementowano grę TicTacToe - czyli kółko i krzyżyk. Wersja tej gry jest niestandardowa i zakłada możliwość podyktowania przez gracza rozmiaru planszy na której jest toczy się rozgrywka, jak również warunku zwyciestwa, czyli liczby pól z tym samym znakiem ustawionych w jednej lini.

2 Algorytm MinMax

W Minimaxie dwaj gracze nazywają się maksymalizatorem i minimalizatorem. Maksymalizator próbuje uzyskać najwyższy możliwy wynik, podczas gdy minimalizator stara się zrobić coś przeciwnego i uzyskać najniższy możliwy wynik. Z każdym stanem planszy jest związana wartość. W danym stanie, jeśli maksymalizator ma przewagę, wynik będzie zwykle miał wartość dodatnią. Jeśli minimalizator ma przewagę w tym stanie planszy będzie miał zwykle pewną ujemną wartość. Wartości planszy są obliczane przez niektóre heurystyki, które są unikalne dla każdego rodzaju gry. W tym projekcie był to jedynie 3 warunki - zwyciestwo pierwszego lub drugiego gracza oraz remis.

2.1 Algorytm AlfaBeta

Jest to poprawiony algorytm minimax, który odrzuca niektóre niekorzystne rozwiązania pozwalając skrócić czas wykonywania algorytmu na obliczenia, które i tak nie zmieniłyby wyniku działającego algorytmu. W tej wersji pojawia się zmienna Depth-głebokość przeszukiwań drzewa, która ustala ile ruchów w przód ma przeszukać algorytm. Dla większych map najlepiej gdy ta wartość będzie stosunkowo niska, ale nie za niska, ponieważ może to doprowadzić do tego, że algorytm nie znajdzie w zbyt płytkim drzewie mozliwej wygranej i postąpi losowo, co w dluzszej perspektywie może oznaczac przegraną.

3 Konstrukcja gry

Gra pozwala graczowi zdecydiwać o rozmiaru planszy na której jest toczy się rozgrywka, jak również warunku zwyciestwa. Domyślnie rozgrywka rozpoczyna się od ruchu komputerowego "gracza".

W przypadku zmiennego pola gry nie można sprawdzać "na sztywno" wygranej. Metoda sprawdzająca wygraną działa na zasadzie sprawdzania w 4 różnych kierunkach w: pionie, poziomie, ukosie od lewej do prawej, ukosie od prawej do lewej; i sprawdza aż napotka X takich samych znaków, gdzie X to liczba zadeklarowana przez użytkownika na początku programu czyli ilość znaków w rzędzie. Jeżeli nie napotka takich samych znaków to nie ma komunikatu o wygranej któregoś z graczy. Jeżeli wszystkie pola się zapełnią, a wciąż nie ma wygranej to wyświetlony zostaje komunikat o remisie. W ostatecznej wersji algorytm sprawdza po kolei wszystkie wiersze, rzędy i skosy, jednak podczas implementowania projektu podjąłem próbę zoptymalizowania tej metody. Polegało to na poszukiwaniu wygranej na podstawie ostatniego ruchu. Sprawdzano pod kątem wygranej tylko ten wiersz, kolumnę oraz dwa skosy w których znajdował się ostatni ruch, pozowliło to przyspieszyć pracę algorytmu MinMax jednak nie był on w pełni sprawny, gdyż taka metoda nie pozwala ocenić stanu całej planszy, a jedynie warunek wygranej dla danego gracza. Spowodowało to nie umieszczenie tej metody w ostatecznej wersji programu.

4 Wnioski

W momencie wybierania głębokości wyszukiwania dla algorytmu należy uważać, aby nie wybrać za dużej - wówczas program może liczyć możliwości ruchów kilka/kilkanaście minut. Z kolei wybranie zbyt małej głębokości może doprowadzić, że postąpi losowo, co może oznaczac przegraną.
W trakcie tworazenia gry warto również wybrać jaki warunek zwycięstwa dlugośc pola rozgrywki, w przeciwnym przypadku, jeżeli zopoczynającym rozgrywkę jest komputer, wygra on zawsze i zdecydowanie szybciej.

Takie zastosowanie algorytmu minmax, a dokładnie jego wersji alfabeta, pozwala na stworzenie rozgrywki nie możliwej do wygrania dla gracza nie będącego "komputerem". Dzięki przeszukaniu drzewa, w ktorym program oszacowuje każdy ruch możemy stworzyć "sztuczną inteligencję" stwarzającą wrażenie przeciwnika nie do pokonania.