Politechnika Wrocławska

PAMSI - Projektowanie Algorytmow i Metody Sztucznej inteligencji

Projekt 3

TIC TAC TOE - Link do repozytorium

Skład grupy

Marcin Cichocki 259322

Wydział i kierunek studiów

W12N, Automatyka i Robotyka

Kod grupy zajęciowej, termin zajęć

Y03-51f, wt 13:15-15:00

Prowadzący

Dr hab. inż. Andrzej Rusiecki

Data wykonania ćwiczenia, termin oddania sprawozdania

13.06.2022, 13.06.2022

autor szablonu:

Jakub Marczak

Spis treści

1 Wprowadzenie

1.1 Algorytm MINIMAX

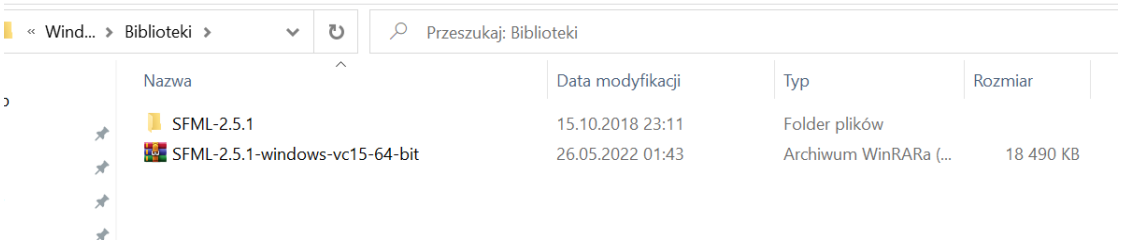
Algorytm MINIMAX to metoda minimalizowania maksymalnych możliwych strat. Alternatywnie można je traktować jako maksymalizację minimalnego zysku (maximin). Wywodzi się to z teorii gry o sumie zerowej, obejmujących oba przypadki, zarówno ten, gdzie gracze wykonują ruchy naprzemiennie, jak i ten, gdzie wykonują ruchy jednocześnie. Zostało to również rozszerzone na bardziej skomplikowane gry i ogólne podejmowanie decyzji w obecności niepewności.

1.2 Alfa Beta ciecia

Alfa beta ciecia to algorytm przeszukujący, redukujący liczbę węzłów, które muszą być rozwiązywane w drzewach przeszukujących przez algorytm min-max. Jest to przeszukiwanie wykorzystywane w grach dwuosobowych, takich jak kółko i krzyżyk, szachy, go. Warunkiem stopu jest znalezienie przynajmniej jednego rozwiązania czyniącego obecnie badaną opcję ruchu gorszą od poprzednio zbadanych opcji. Wybranie takiej opcji ruchu nie przyniosłoby korzyści graczowi ruszającemu się, dlatego też nie ma potrzeby przeszukiwać dalej gałęzi drzewa tej opcji. Ta technika pozwala zaoszczędzić czas poszukiwania bez zmiany wyniku działania algorytmu.

1.3 Biblioteka SFML

Do stworzenia GUI uzyta zostala biblioteka SFML. SFML jest darmową biblioteką, która dostarcza niskopoziomowy oraz wysokopoziomowy dostęp do karty graficznej, urządzeń wejściowych, dźwięku itp. Biblioteka SFML zyskała bardzo dużą popularność ostatnich latach i obecnie jest najczęściej polecaną oraz najchętniej wybieraną technologią przez deweloperów do wytwarzania gier 2D. By dany program dzialal na kazdym komputerze potrzebne jest sciagniecie darmowej biblioteki sfml i rozpakowanie jej na dysku C w folderze Biblioteki (nalezy stworzyc folder Biblioteki). Sciezka powinna wygladac tak:
$\backslash C : \backslash Biblioteki$.
Tutaj link do biblioteki SFML - Link do biblioteki. W moim programie wybralem wersje dla 64-bit.



Rysunek 1: Miejsce w ktorym musi znajdowac sie biblioteka SFMl

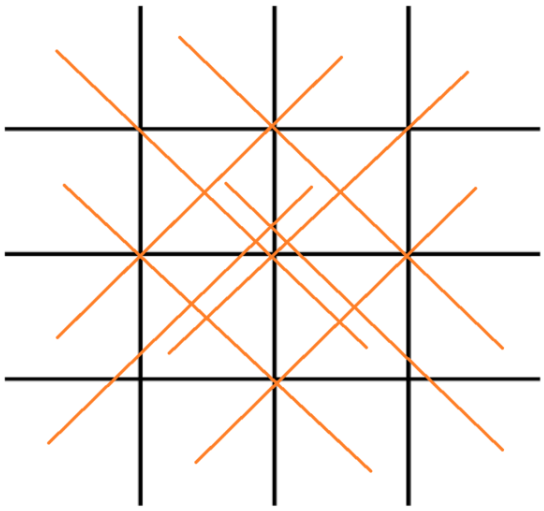
2 Dzialanie programu

2.1 GUI

Samo zaimplementowanie GUI nie bylo jakims wiekszym problem. Najwieksze wyzwanie stanowilo poznanie funkcji z biblioteki SFML i zastosowanie ich. Biblioteka SFML jest przyjazna dla uzytkownika i nie potrzeba wiekszej wiedzy zeby moc sie nia sprawnie poslugiwac.

2.2 Implementacja algorytmu MINIMAX

Algorytm MINIMAX sprawil mi najwiecej trudnosci ze wzgledu na rozbudowanie polecenie naszego zadania. Na kazdej stronie w internecie gdzie moglismy znalezc zaimplementowany algorytm MINIMAX to byl to algorytm przystowany dla planszy 3x3 a w naszym poleceneiu uzytkownik ma prawo wybrac rozmiar planszy. Gdybysmy mieli stworzyc mapy od 3x3 do 10x10 z takimi samymi zasadami do wygrania (dla mapy 3x3 potrzeba do wygranej 3 znaki w rzedzie/kolumnie/diagonalnie. Dla mapy 4x4 4 znaki w rzedzie/kolumnie/diagonalnie itd.) nie byloby to az tak trudne ale jesli uzytkownik ma te prawo do wyboru liczby znakow potrzebnych do wygranej to sytuacja zaczyna sie komplikowac poniewaz liczba mozliwosci na wygranie poprzez znaki postawione diagonalnie zwieksza sie w szybkim tempie. Dla mapy 3x3 i dla wygranej poprzez postawienie znakow diagonalnie mamy dwie opcje, diagonalnie od lewej w dol i od prawej w dol ale dla mapy 4x4 i dla trzech znakow do wygranej mozliwosc poprzez wygrane diagonalnie zwieksza sie z 2 do 8 opcji.



Rysunek 2: Pomaranczowe linie symbolizuja w jakich miejscach moga zostac postawione krzyzyki lub kolka zeby wygrac poprzez postawienie znakow diagonalnie

Tutaj zaczely sie problemy poniewaz trzeba bylo przerobic funkcje sprawdzajaca wygrana zeby sprawdzala wygrana dla kazdego mozliwego scenariusza (dla kazdej mozliwej mapy i kazdej mozliwej ilosci znakow w rzedzie/kolumnie/diagonalnie). Po przerobieniu funkcji na funkcje "uniwersalna" postanowilem ja rozbic na cztery osobne funkcje.

- Pierwsza funkcja $win\_row()$ sprawdza wygrane w rzedach.
- Druga funkcja $win\_col()$ sprawdza wygrane w kolumnach.
- Trzecia funkcja $win\_diagonalLeft()$ sprawdza wygrane diagonalnie od lewej do prawej idac w dol.
- Czwarta funkcja $win\_diagonalLeft()$ sprawdza wygrane diagonalnie od prawej do lewej idac w dol.

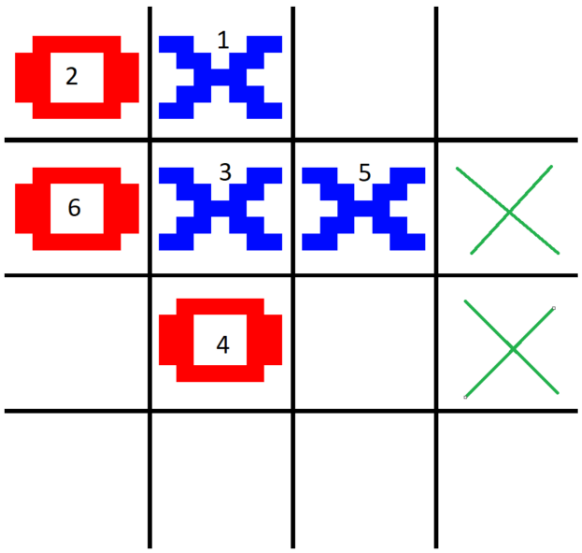
Funkcje zostaly rodzielone ze wzgledu na to ze czasem "countery" z osobych funkcji (counter=licznik znakow sasiadujacych ze soba) przechodzily do nastepnej funkcji i funkcja zwracala wygrana gdy zadna wygrana nie nastapila.

2.3 "Glupie" AI

Po zaimplementowaniu naszego algorytmu zauwazylem ze moje AI wykonuje ruchy ktore prowadza do przegranej komputera. Nie wiedzialem czy to bylo spowodane i zaczalem przegladac caly kod lecz nie dalo sie znalezc bledu. Po dwoch dniach przerwy postanowilem przejrzec wszystkie funkcje ktore wykorzystuje moj minimax do znalezenia najlepszego ruchu. Moja funkcja ktora zwraca mi dana wartosc pozycji (np czy jest remis czy przegrana) zawsze zwracala mi 0, czyli przypadek dla ktorego jest remis lub rowna pozycja. Bylo to dosc dziwne poniewaz moje funkcje sprawdzajace wygrana dzialaly bez zarzutu. Sprawdzilem wiec dokladnie funkcje *win_board_state*() (funkcja zwracaja wartosc pozycji) i rzeczywiscie przy kazdym mozliwym ruchu funkcja zwracala zero. Po naprawieniu funkcji skompilowalem program raz jeszcze ale nie oczekiwalem duzo poniewaz podczas pisania tego programu korzystalem z 5 roznych repozytoriow w ktorych zaimplementowany algorytm MINIMAX i zaden nie dzialal prawidlowo wiec myslalem ze juz jestem skazany na porazke. O dziwo program zaczal dzialac prawidlowo, blokowal kazdy moj ruch a kiedy sam zaczynal to zawsze wybieral najlepszy ruch (jak sie "podlozylem" to zawsze wybieral opcje do wygrania). Po 3 dniach implementacji tego algorytmu w koncu program zaczal dzialac prawidlowo.

2.4 Mozliwosc wygrania na komputer

Na mapie 3x3 przy ustawieniu glebokosci maksymalnej na poziomie "4" lub "5" czasem algorytm nie zauwazy najlepszego ruchu i pozwoli nam wygrac. Przy nie ustawianiu maksymalnej glebokosci i jesli damy przeliczyc komputerowi kazdy mozliwy ruch to komputer zawsze nas zablokuje. Sa jednak sytuacje w ktorych mamy duze prawdopodobienstwo ze wygramy z komputerem. Mowa tu o planszach wiekszych od 3x3 i wybraniu opcji liczby znakow potrzebych do wygranej rownej 3.



Rysunek 3: Liczbami zostaly ponumerowane kto wykonal jaki ruch. Krzyzyk to gracz a kolko to komputer. Po wykonaniu 6 ruchow mamy dwie opcje do wygrania (zielone "X") poniewaz "X" zaczyna. Jest to sytuacja w ktorej zawsze wygramy

Wiec dla kazdej mapy i dla opcji trzech znakow do wygranej wygrywa zawsze ta osoba ktora zaczyna. Algorytm po wykonaniu naszego ruchu nr 5 juz wie ze przegral i wszystkie jego drogi prowadza do przegranej wiec wykonuje ruch nr 6 ktory moze dac mu wygrana jesli gracz popelni blad.

2.5 Rzeczy do poprawy

Program dziala bez zarzutu ale mozna byloby go ulepszyc. Pierwsza rzecza do poprawy byloby ustawienie stalego pierwszego ruchu gdy zaczyna komputer. Chodzi o to ze gdy plansza jest pusta to komputer ma duzo opcji do przeliczenia ktory ruch jest najlepszy. Liczy a tak czy siak dochodzi do wniosku ze najlepszy ruch to ktorykolwiek rog planszy. Rozwiazaniem jest zeby ustawic staly pierwszy ruch dla komputera ktory jest umieszczony w ktorymkolwiek rogu poniewaz po naszym ruchu liczba mozliwych scenariuszy maleje i komputer nie bedzie az tyle liczyl ktory ruch bedzie najlepszy.
Kolejna mozliwym ulepszeniem jest ustalenie maksymalnej glebokosci dla kazdej mapy. Jesli nie ma limitu dla maksymalnej glebokosci to gra dziala "plynnie" tylko dla mapy 3x3. Przy mapie 4x4 juz liczy zbyt dlugo.

3 Wizualizacja dzialania programu - Link do filmu na YT

Powyzej odnosnik "Link do filmy na YT" ktory przekieruje na platforme youtube.com do filmiku na ktorym pokazana jest zasada dziala programu.

4 Wnioski

- Program dziala prawidlowo. Algorym MINIMAX wykonuje prawidlowe ruchy dla mapy 3x3 jesli nie ma limitu dla maksymalnej glebokosci. Wygrana z komputerem w takim przypadku jest niemozliwa.
- Jest mozliwosc 100% wygranej z komputerem jesli mapa jest wieksza od 3x3 a liczba znakow potrzeban do wygranej jest rowna 3
- Oglupienie naszego AI jest kosztem optymalizacji programu. Jesli ustawimy limit dla maksymalnej glebokosci to czasem zdarzy sie taki przypadek ze program nie doliczy sie najlepszego ruchu i da nam wygrac. Mowa tutaj o mapach wiekszych od 3x3 poniewaz na mapie 3x3 nie potrzeba nam limitu dla glebokosci.

Literatura

[1] Wikipedia MINIMAX
https://pl.wikipedia.org/wiki/Algorytm_min-max

[2] Wikipedia Alpa Beta
https://pl.wikipedia.org/wiki/Algorytm_alfa-beta

[3] MINIMAX by Geeks for Geeks
https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/

[4] MINIMAX by Never Stop Building
https://www.neverstopbuilding.com/blog/minimax

[5] MIMINAX by George Seif
https://github.com/GeorgeSeif/Tic-Tac-Toe-AI