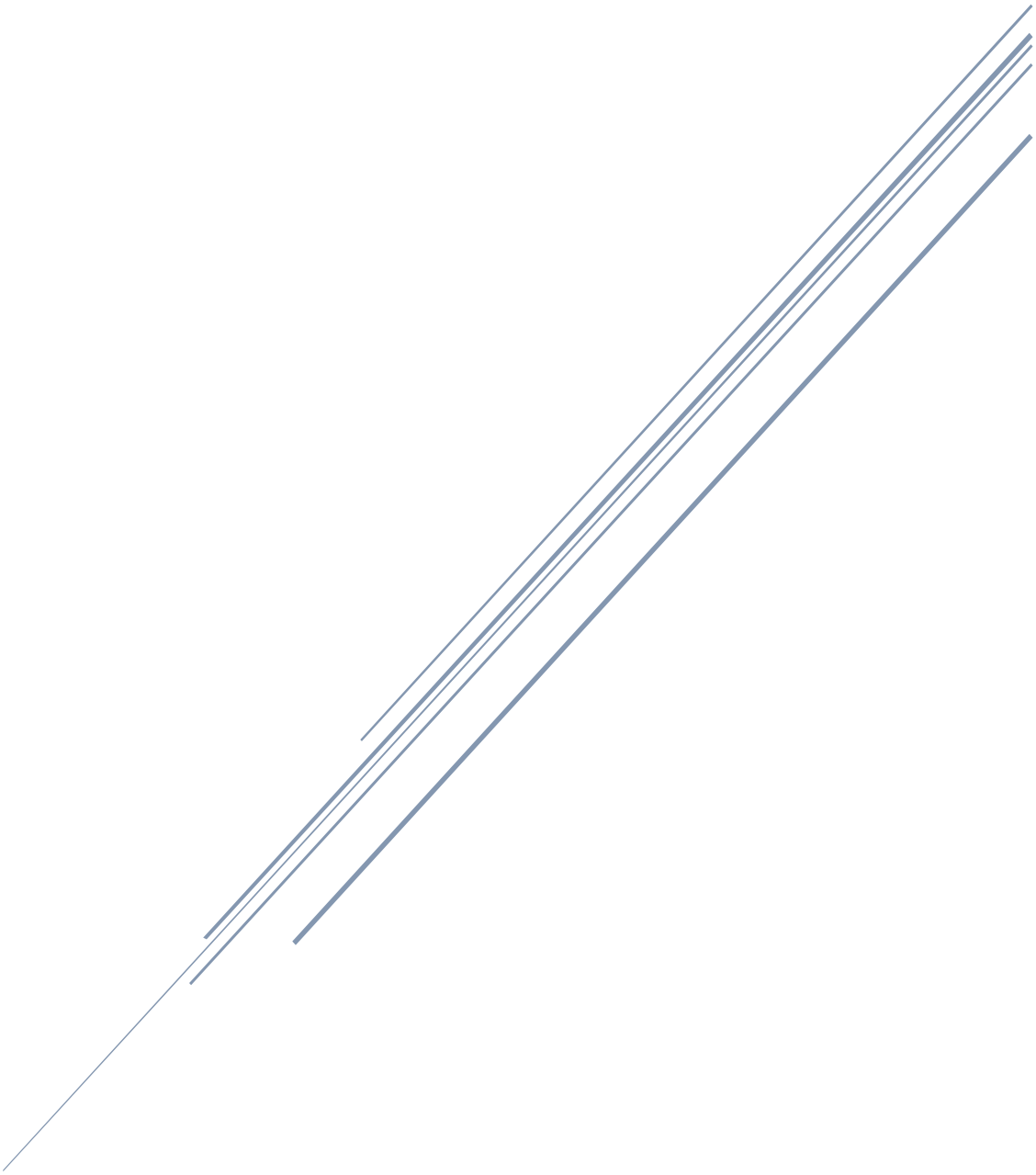
# PROJEKT 4

Kacper Kaciłowicz 248951



Mgr inż. Magda Skoczeń

Projektowanie algorytmów i metody sztucznej inteligencji

# 1. Wprowadzenie

Celem projektu było zaprogramowanie gry wykorzystującej metody sztucznej inteligencji. W tym przypadku zdecydowano się na realizację klasycznej gry „kółko i krzyżyk" rozszerzonej o możliwość wyboru wielkości planszy. Jednym z najbardziej powszechnych algorytmów stosowanych w grach jest algorytm MIniMax.

# 2. Zasady gry

- W grze bierze udział 2 graczy
- Każdy z graczy ma swój znak/tag, który zaznacza na planszy (kółko lub krzyżyk)
- Gra odbywa się na kwadratowej planszy o zadanych wymiarach
- Użytkownik przed rozpoczęciem gry ustala wielkość planszy
- Wygrywa ten gracz, który będzie miał ustawione sąsiadujące tyle znaków pod rząd ile wynosi rozmiar planszy, czyli np. na planszy 4×4 potrzeba 4 kółek z rzędu do wygranej danego gracza. Jako sąsiadujące rozumie się będące obok siebie w sposób pionowy, poziomy lub ukośny.

# 3. Opis wykorzystanych technik Sztucznej Inteligencji

MiniMax opiera się przede wszystkim na wyszukiwaniu metody minimalizującej jak największych (maksymalnych) strat. Polega on na tworzeniu drzewa rozwiązań oraz dokonywaniu przeszukiwań wewnątrz niego w celu podjęcia decyzji dzięki której uzyska najbardziej korzystny wynik. Żeby tego dokonać wydziela się dwóch graczy: Maximizer – gracz, który stara się uzyskać jak najwyższy wynik (wygrana gracza) oraz Minimizer – gracz, który stara się uzyskać jak najniższy wynik przeciwnika (porażka gracza, zatem wygrana przeciwnika). Zakłada się, że Maximizer oraz Minimizer dokonują wyboru po sobie, dzięki czemu w reprezentacji drzewa – gałęzie to są wybory graczy, a liście to są zyski/straty po wykonaniu ruchu. W efekcie SI jest w stanie zasymulować przebieg rozgrywki i znaleźć dla siebie najlepsze rozwiązanie.

Jeśli chodzi o implementację programistyczną w grach to do realizacji MiniMax stosuje się wywołania rekurencyjne, które w przypadku odnalezienia lepszej opcji zapisują jej wartość oraz np. w przypadku „kółko i krzyżyk" współrzędne ruchu jaki należy wykonać.
Najpierw wykonuje się sprawdzenie i przypisanie wyniku danej rozgrywki – czy dany ruch daje już zwycięstwo/porażkę lub czy jest możliwe wykonanie kolejnego ruchu. Następnie jeśli jest ruch Maximizera dokonuje się przeszukania wszystkich możliwych ruchów. Gdy ruch jest możliwy w zostaje on zastosowany, a następnie dokonuje się rekurencyjne wywołanie MiniMax tylko wtedy symulowany jest ruch Minimizera i dla niego kroki są takie same. W ten sposób dzieje się symulacja rozgrywki, ale funkcja zwraca informacje o jej przebiegu – czy dany ruch da wygraną, czy przegraną. Następnie zostaje zapisana wartość symulacji – danego przebiegu, a gdy zostanie znaleziona lepsza ścieżka to wartość zostaje zamieniona.

Usprawnienia wprowadzone w projekcie:

- **Algorytm Alfa-Beta –** Algorytm przeszukujący drzewa, który w trakcie swojego działania redukuje liczbę węzłów, które są konieczne do przeszukania. Jego działanie polega na wprowadzeniu dwóch zmiennych – alfa i beta, którym przypisywane są wartości danych decyzji. W efekcie w przypadku znalezienia lepszej opcji dla gracza w symulacji nie ma konieczności dokonywania dalszego przeszukiwania gałęzi i się ją „odcina", czyli nie przeszukuje jej oraz jej potomków. To usprawnienie pozwala zaoszczędzić czas przeszukiwania, gdyż nie ma konieczności przeszukiwania całego drzewa jak w przypadku klasycznego MiniMax.
- **Sprytniejsze SI –** trudno dobrać nazwę, gdyż nie jest to konkretnie nazwany i opisany algorytm. To usprawnienie SI polega na lepszym obliczaniu wyniku ruchu, ze względu na uwzględnieniu ilości ruchów. Wykorzystane są wzory $value = value - RecursionDepth$ dla Maximizera oraz $value = value + RecursionDepth$ dla Minimizera, gdzie RecursionDepth oznacza głębokość rekurencji, czyli tak naprawdę ilość ruchów do wykonania, a value to wartość ruchu, czyli efekt jaki przyniesie (wygrana/porażka). W efekcie to usprawnienie ma na celu doprowadzenie do jak najszybszej wygranej lub jak najpóźniejszej porażki.
- **Ustalenie maksymalnej głębokości rekurencji –** w przypadku większych rozmiarów plansz niż np. 3×3 zaczyna się pojawiać problem bardzo długiego podejmowania decyzji przez SI. Wynika on z bardzo wielu wywołań rekurencyjnych MiniMax – dużej głębokości rekurencji. W celu ograniczenia tego czasu zostało wprowadzone ograniczenie maksymalnej głębokości rekurencji, dzięki czemu nie zostaje wywołana rekurencja więcej razy, niż zadana. Niestety nie można nie wspomnieć o tym, że lepszy czas działania jest uzyskany kosztem sprytu SI. W przypadku mniejszych plansz jak 3×3 SI „atakuje" oraz dąży do zwycięstwa, a w przypadku większych plansz de facto jego głównym celem jest „obrona" i doprowadzenie do remisu. W przypadku tego projektu wartości głębokości rekurencji zostały ustalone na sztywno – wynoszą one 6 dla plansz o wielkości $\leq 4 \times 4$ oraz 4 dla plansz o wielkości $> 4 \times 4$.

# 4. Podsumowanie i wnioski

- Wielkość planszy ma wpływ na czas podejmowania decyzji przez SI – im większa plansza, tym dłużej funkcja się wykonuje
- SI działa poprawnie dla mniejszych plansz, dąży do zwycięstwa, natomiast dla większych początkowo wybiera dowolne wolne miejsca oraz jedynie jest w stanie doprowadzić do remisu
- Wprowadzone usprawnienia skróciły czas podejmowania decyzji przez SI

# 5. Bibliografia


1) http://wazniak.mimuw.edu.pl/index.php?title=Sztuczna_inteligencja/SI_Modu%C5%82_8_-_Gry_dwuosobowe
2) https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning
3) https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-2-evaluation-function/?ref=lbp
4) https://en.wikipedia.org/wiki/Minimax

5) https://www.youtube.com/watch?v=l-hh51ncgDI
6) https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/