26.05.2020r.

Jacek Multan

indeks: 248964

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 3 – Gry

Dr Łukasz Jeleń

czwartek 9:15 – 11:00

Wstęp

Celem projektu było stworzenie gry, w której przeciwnego gracza reprezentuje komputer. W ramach zadania napisano szachy, a jako metodę sztucznej inteligencji wykorzystano algorytm alfa-beta, który jest ulepszoną wersją minimaxa.

Opis gry

Szachy są grą rozgrywaną na planszy w kształcie kwadratu o 64. polach. Każdy z graczy posiada na początku 16 bierek, z których każdy rodzaj ma inne zasady poruszania. Istnieje wiele wyjątków, które stwarzają utrudnienia podczas programowania. Jest to m. in podwójny szach, w którym tylko król może się poruszać, bicie w przelocie, podczas którego bity pionek znajduje się na innym polu, niż to na którym stawiamy poruszaną bierkę, roszada, będąca złożeniem dwóch ruchów oraz wiele więcej. W celu stworzenia wersji graficznej, wykorzystano bibliotekę SFML.

Algorytm

Jako sztuczną inteligencję, przeciwko której gra człowiek, zaimplementowano ulepszony algorytm minimax. Wybiera on scenariusz, w którym otrzyma najlepszą sytuację. W grze biorą udział dwaj zawodnicy. Jeden dąży do otrzymania jak największego wyniku, drugi do jak najmniejszego. Ich tury następują po sobie. W związku z tym algorytm symuluje wszystkie możliwe ścieżki rozgrywki i wybiera odpowiednio tą, w której otrzyma więcej lub mniej punktów, w zależności czyją kolej na danej głębokości rekurencji reprezentuje. Ma on wykładniczą złożoność obliczeniową (zakładając, że na każde x ruchów przypada x następnych), w związku z czym jest niezwykle wolny. W celu przyspieszenia go stosuje się cięcia alfa-beta. Polegają one na sprawdzaniu czy istnieje szansa, na wybranie danej ścieżki, znając część wyników. Np. gracz maksymalizujący ma do wyboru wersję, gdzie otrzyma 30 punktów lub wynik mniejszy z 20 lub jakiejś jeszcze nieobliczonej wartości. Wiedząc, że druga gałąź zwróci 20 lub mniej, nie ma sensu sprawdzać kolejnych elementów w tym poddrzewie. Pozwala to pominąć wyliczanie znacznej części scenariuszy, a tym samym przyspieszyć działanie programu. Im lepiej rokujące możliwości przyjmiemy na początek, tym większa szansa, że pominiemy więcej gałęzi. Z tej przyczyny, algorytm zaczyna przeszukiwanie od hetmana i figur, a kończy na królu poprzedzonym pionkami. Dodatkowo, żeby zwiększyć presję na przeciwniku poza zwykłymi punktami za materiał, czyli posiadane figury, algorytm dostaje premię za szacha, kontrolę centrum i ruchy do przodu. Ustawienie przewidywania na cztery kroki w przód zachowuje dobrą proporcję poziomu trudności do czasu, jaki zajmuje ruch sztucznej inteligencji, a który wynosi z reguły około sekundy.

Wnioski:

Algorytm minimax jest prostym w realizacji i skutecznym sposobem imitowania zachowań gracza. Można go wykorzystać podczas tworzenia gier turowych posiadających nieznaczną ilość dopuszczalnych ruchów. Wraz ze wzrostem głębokości czas oczekiwania na wykonanie posunięcia przez komputer bardzo szybko rośnie, więc warto implementować ulepszoną wersję algorytmu, jaką jest alfa-beta oraz stosować różnego rodzaju zabiegi, przewidujące wynik końcowy. W celu zmniejszenia losowości, należy dodać możliwie spory zakres otrzymywanych punktów. Jeśli będziemy oceniać sytuację w postaci przegrana, wygrana lub remis, w wypadku, gdy gra nie zostanie ukończona przed sprawdzeniem danej głębokości otrzymamy ruch mogący źle rokować, będący w rzeczywistości pierwszą sprawdzoną możliwością.

Źródła

- **https://stackoverflow.com – 1 V 2020 - 25 V 2020**
- **https://en.cppreference.com/ -1 V 2020 - 25 V 2020**
- **https://www.sfml-dev.org/documentation/2.5.1/ - 11 V 2020 - 25 V 2020**
- **https://www.codeproject.com/Articles/84461/MinGW-Static-and-Dynamic-Libraries - 15 V 2020**
- **https://www.youtube.com/watch?v=l-hh51ncgDI - 9 V 2020**