Dokumentacja

Zaawansowane programowanie w C++

Projekt: Gra w Monopoly z komputerem

Zespół: Kamil Kośnik, Kacper Radzikowski

Repozytorium projektu: https://github.com/FRSH-0109/Monopoly-vs-AI-Game

Liczba linii kodu w projekcie: 21 888

Liczba testów: 726 asercje

Procentowe pokrycie kodu testami: 80%

Liczbę godzin poświęconą na projekt: 282h

Opis oryginalny:
Gra w Monopoly z komputerem. Proszę skupić się na rozwoju AI w grze. Jedno z możliwych podejść jest ogólnie opisane tu, a tu jest implementacja. Można zajrzeć (i użyć jasno wyodrębnionych fragmentów) jednej z wielu bibliotek, np. tinyai, open-neat. Przed rozpoczęciem realizacji projektu proszę zapoznać się z zawartością strony.

Źródła:
https://www.franksworld.com/2022/01/03/ai-learns-insane-monopoly-strategies/
https://github.com/b2developer/MonopolyNEAT
https://staff.elka.pw.edu.pl/~rbiedrzy/ZPR/index.html

Opis projektu przez zespół:
Projekt realizuje rozgrywkę popularnej gry planszowej Monopoly, w której, w tym przypadku, przeciwko sobie udział będzie brał gracz/gracze fizyczni oraz gracz/gracze kontrolowani poprzez algorytm sztucznej inteligencji.

Zasady oryginalnej gry Monopoly na której wzorze będzie opierać się realizacja:
https://www.hasbro.com/common/instruct/00009.pdf
https://www.ultraboardgames.com/monopoly/game-rules.php

Opis funkcjonalności oraz porównanie z wstępnymi założeniami:
Zrealizowane:

- Komunikacja z użytkownikiem

    - Graficzny Interfejs użytkownika, który umożliwia graczom interakcję z grą
        - Elementy GUI, takie jak przyciski, menu, grafiki itp.
        - Powiadomienia dla informowania gracza o zachodzących zdarzeniach
- Menu początkowe służące do wyboru warunków rozgrywki
    - ilości graczy
    - ich rodzaju (człowiek lub AI)
    - ich poziomu w przypadku gracza AI
- Plansza:
    - Opis planszy do gry, włącznie z polami, ich nazwami i kolorami.
    - Wizualna reprezentacja planszy poprzez proste grafiki
    - Gracze i ich pionki reprezentowani przez kolory
- Talia kart Szansa
- Silnik gry
    - Ustalenie i kontrolowanie kolejności graczy w turze
    - Losowe generowanie talii kart Szansa
    - Realizowanie ruchów graczy
    - Realizowanie podjętych przez graczy decyzji
    - Zarządzanie majątkiem graczy (m.in. gotówka, posiadane nieruchomości)
    - Zarządzanie taliami kart Szansa
    - Realizowanie efektów kart Szansa
    - Zarządzanie wymianami między graczami
    - Kontrola przebywania graczy w więzieniu
    - Określanie wartości czynszu
    - Stawianie budynków na nieruchomości
    - Zarządzanie aukcjami
    - Zarządzanie zastawianiem budynków
    - Sprawdzanie warunków eliminacji/zwycięstwa

Zrealizowane częściowo:

- Mechanizm sztucznej inteligencji przeciwników
    - Różne poziomy trudności gracza SI uzyskane dzięki algorytmowi uczącemu NEAT
    - Samodzielność prowadzenia rozgrywki przez gracza SI

Nierealizowane:

- Stos kart kasa społeczna i ich wpływ na rozgrywkę, w zamian pola są bez akcji
- Karty “Wyjścia z więzienia” nie zostały zaimplementowane pod kątem ich wykorzystania

Napotkane problemy:

- **Realizacja algorytmu NEAT –** największym problemem jaki nas spotkał były komplikacje związane z wyborem i implementacją biblioteki realizującej algorytm NEAT. W trakcie prac okazało się, że dostępne otwarcie biblioteki realizujące te zadanie nie były jakościowe. Wszystkim brakowało klarownej dokumentacji, część zwyczajnie nie działała. Sprawiło to, że implementacja algorytmu została zrealizowana znacznie później co w połączeniu z długim wymaganym czasem treningu algorytmu dało nieidealny wynik działania algorytmu.

Popełnione błędy:

- **Skala projektu –** Na przestrzeni prac nad projektem wyraźnym problemem, który się pojawiał była rosnąca jego skala. Błąd polegał na uświadomieniu sobie, że próba bardzo

dokładnego zrealizowania zestawu reguł gry skutkował dużym, nieosiągalnym do wykonania w terminie nakładem prac, co widać w różnicy między założonym czasem pracy, a jego faktyczną wartością.

- **Lepszy podział architektury -** niektóre mechanizmy, głównie klasa Graczy mogła zostać lepiej zaplanowana, lepiej realizując podział na część odpowiedzialną za wizualizację wyników działania logiki oraz rzeczywistą logikę.
- **Powtarzanie kodu** - niektóre fragmenty kodu (głównie te związane z wektorem pól na planszy realizowanych za pomocą std::variant) powinny były zostać sprowadzone do oddzielnych funkcji. Jest to błąd powiązany z niedocenieniem skali projektu, ponieważ na początku uznając, że nie będą to powszechnie stosowane mechanizmy nie rozważyliśmy takiego kroku.

Lista Zadań

Zadanie	**Czas zakładany**	**Czas rzeczywisty**
Stworzenie szablonu projektu umożliwiającego pracę każdemu z członków oraz płynną synchronizację postępów	5h	5h
Implementacja biblioteki graficznej SFML	5h	5h
Stworzenie programu jako niezależnego okna z podstawowym menu gry	5h	5h
Mechanizm wyboru rodzaju rozgrywki i jej rozpoczęcie z poziomu menu	10h	20h
Stworzenie/wygenerowanie prostych grafik dla elementów rozgrywki	5h	5h
Zaplanowanie struktury i implementacja obiektów reprezentujących graczy	10h	10h
Zaplanowanie struktury i implementacja obiektów reprezentujących planszę do gry oraz jej pola	15h	15h
Zaplanowanie struktury i implementacja podstawowego silnika gry	10h	30h
Mechanizm ustalenia kolejności graczy w turze	3h	3h
Mechanizm kontrolowania kolejności graczy w turze	3h	3h
Metoda tasująca talię do gry	3h	3h
Metoda odpowiadająca za ruch	4h	5h
Metoda podejmująca działania po wylądowaniu gracza na polu	10h	3h
Interfejs do realizacji decyzji graczy	10h	10h
Metoda odpowiedzialna za transfer pieniędzy między graczami	4h	5h
Mechanizm wymian między graczami	4h	10h
Stworzenie kolekcji kart Szansa i Kasa Społeczna w formie pliku typu JSON	3h	5h
Metoda dobierania karty Szansa/Kasa Społeczna	4h	5h
Mechanizm kontrolowania stanu gracza w więzieniu	4h	5h
Metoda do stawiania budynków na nieruchomości	4h	5h
Metoda zarządzająca aukcjami	4h	20h
Metoda do zastawiania nieruchomości	5h	10h
Metoda do pozbywania się zastawu nieruchomości	5h	5h

Metoda do kontroli warunków zwycięstwa w grze	5h	5h
Implementacja biblioteki dla sztucznej inteligencji	5h	20h
Zaplanowanie struktury i implementacja sztucznej inteligencji jako gracza	10h	10h
Realizacja procesu uczenia sztucznej inteligencji algorytmem NEAT	20h	30h
Implementacja uzyskanych algorytmów graczy SI do rozgrywki z podziałem na poziomy	5h	---
Testowanie całkowitej rozgrywki ze wszystkimi funkcjonalnościami	5h	20h
Zakończenie projektu oraz zdanie go w finalnej formie	2h	5h
SUMA	**177h**	**282h**