Wizualizacja drzewa stanów algorytmu UCT

Dokumentacja wymagań projektu

Patryk Fijałkowski
Grzegorz Kacprowicz

23 października 2019

Streszczenie

Dokument opisuje ogólne założenia stworzenia projektu wizualizacji drzew stanów algorytmu z dziedziny sztucznej inteligencji - Upper Confidence Bound Applied to Trees. Krytyczne dla skuteczności tytułowej heurystyki są odpowiednia konstrukcja i trawersowanie generowanych drzew. Nasze rozwiązanie wykorzysta ten algorytm w podejmowaniu decyzji podczas grania w dwie przykładowe gry planszowe. Skupimy się na przejrzystej wizualizacji, ukazując kolejne etapy rozrastania się drzew. Finalna wersja aplikacji będzie wygodnym narzędziem do analizy działania algorytmu w czasie rzeczywistym. Przeprowadzana w dokumencie analiza wykazała, że mogą wystąpić typowe ryzyka, które nie wpłyną znacząco na terminy realizacji projektu.

Historia zmian

Wersja	Data	Autor(zy)	Zmiany
1.0	19.10.2019	PF, GK	stworzenie pierwszej wersji dokumentu
1.1	23.10.2019	PF, GK	korekty po uwagach promotora pracy
1.2	23.10.2019	PF, GK	korekty po uwagach promotora pracy
1.3	23.10.2019	PF	drobne korekty po uwagach promotora pracy

Spis treści

1 Słownictwo

- **MCTS** (Monte Carlo Tree Search) - heurystyka podejmowania decyzji w pewnych zadaniach sztucznej inteligencji, np. ruchów w grach. Najczęściej MCTS opiera się na jakimś wariancie metody UCT.
- **UCT** (Upper Confidence Bound Applied to Trees) - algorytm przeszukujący drzewo stanów rozgrywki w poszukiwaniu najbardziej opłacalnych ruchów. Algorytm stara się zachować równowagę między eksploatacją ruchów po ruchach o wysokiej średniej wygranej a eksploracją tych mało sprawdzonych.
- **Podstawowa wizualizacja** - możliwość wyświetlenia wszystkich wierzchołków drzewa. W podstawowej wizualizacji wygląd wierzchołków jest nieistotny.
- **Zaawansowana wizualizacja** - podstawowa wizualizacja wzbogacona o możliwość analizowania statystyk poszczególnych wierzchołków. Kolor wierzchołków będzie reprezentował aktualnego gracza.
- **Pełna wizualizacja** - zaawansowana wizualizacja wzbogacona o możliwość przewijania, przybliżania oraz oddalania podglądu drzewa.

2 Cel biznesowy

Algorytm UCT, będący usprawnieniem MCTS, jest powszechnie stosowanym algorytmem w sztucznej inteligencji. Jest metodą analizującą obiecujące ruchy na podstawie generowanego drzewa, która równoważy eksploatację najbardziej korzystnych z eksploracją mniej korzystnych decyzji. Każdemu wierzchołkowi drzewa odpowiada pewien stan rozgrywki, z którego algorytm rozgrywa losowe symulacje, rozszerzając potem drzewo o kolejne możliwe stany. Sposób, w jaki rozrasta się opisywane drzewo, jest kluczowy dla podejmowania przez algorytm jak najlepszych decyzji.

Celem projektu jest stworzenie aplikacji pozwalającej na wizualizację drzew algorytmu UCT. Aplikacja będzie pozwalała na wizualizowanie drzew generowanych podczas rozgrywania dwóch przykładowych gier (pozwalając przetestować rozwiązanie). Aplikacja powinna pozwalać na wizualizację drzew, ich sekwencji i róznic między kolejnymi drzewami w sekwencji. Powinna być możliwość płynnego przybliżania/oddalania i przewijania wizualizacji oraz zapisu aktualnego stanu do pliku graficznego - wszystko, aby klient mógł wygodnie korzystać z naszego programu.

Taki produkt pozwoliłby zrozumieć klientowi ideę i sposób działania algorytmu UCT.

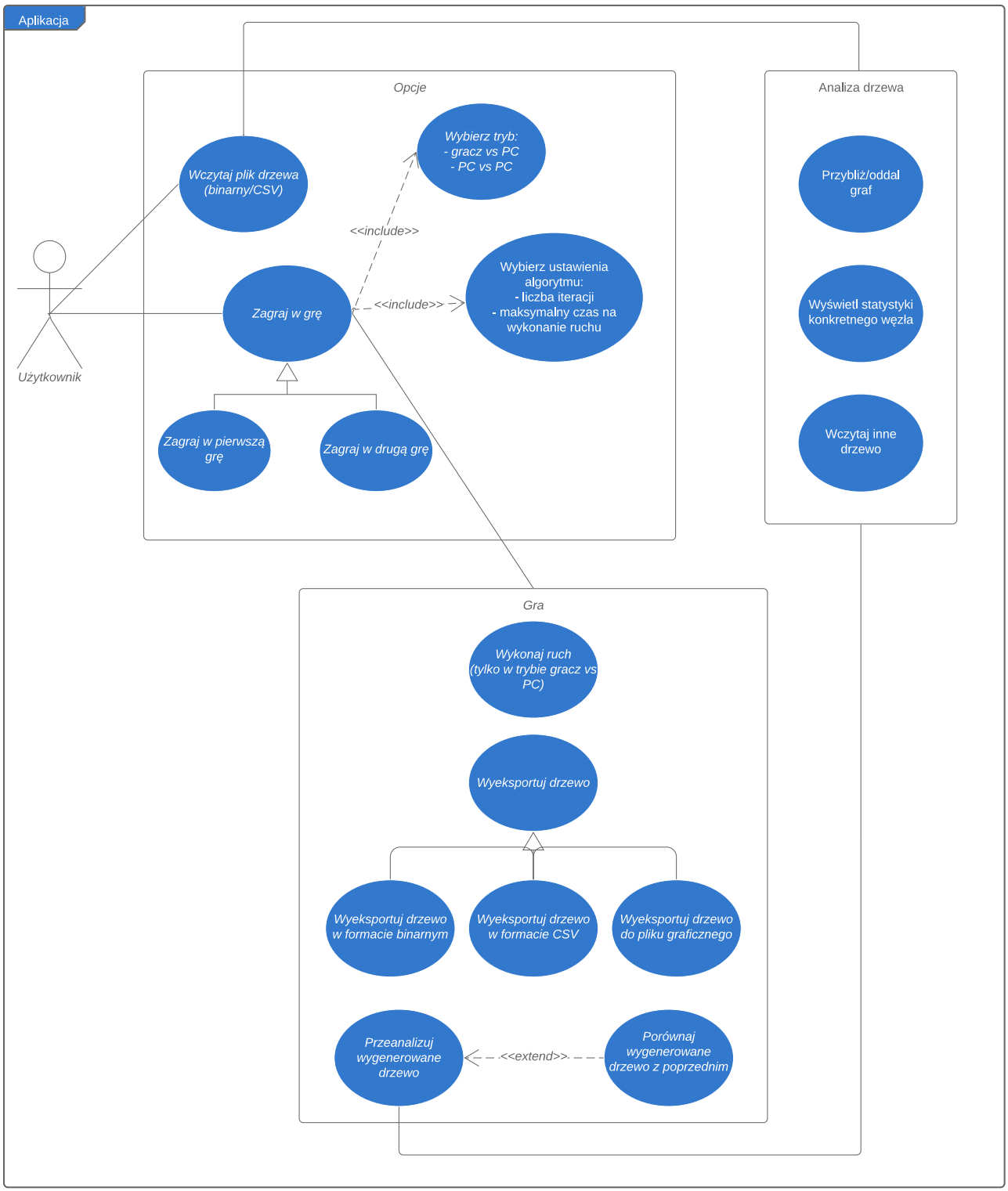
3 Wymagania funkcjonalne

Użytkownik korzystający z naszej aplikacji będzie mógł wybrać jedną z dwóch gier deterministycznych, a do wyboru będzie miał dwa tryby rozgrywki:

1. Gracz vs PC - gdzie użytkownik będzie decydował o swoich posunięciach i zmierzy się on z zaimplementowanym algorytmem,
2. PC vs PC - gdzie będzie on de facto świadkiem symulacji algorytmu, który rozgrywa partię z samym sobą.

Zanim jednak przejdzie do rozgrywki, będzie miał możliwość ustawienia parametrów algorytmu, tj. liczbę iteracji podczas tworzenia drzewa czy też maksymalny czas na ruch przeciwnika (co też wiąże się z liczbą iteracji). Druga opcja, którą będzie dysponować użytkownik, to możliwość wczytania drzewa w formacie zarówno binarnym jak i CSV. a następnie możliwość jego dogłębnej analizy. Będzie on mógł wyświetlać informacje na temat wybranego węzła, a także przybliżać i oddalać cały wygenerowany graf. Podczas samej rozgrywki gracz będzie mógł wyeksportować drzewo po wykonanym ruchu przeciwnika i analizować je w sposób opisany powyżej. Tak samo, będzie też miał możliwość zapisania go w wyżej wymienionych formatach, a także w formacie rastrowym. Użytkownik będzie mógł oglądać rozrost drzewa. Dodatkowo, będzie mógł porównać nowo wygenerowane drzewo z tym z poprzedniego kroku. Powyższe rzeczy dotyczą obu trybów gry i każdej z gier.

Diagram przypadków użycia, który ilustruje przedstawione możliwości, znajduje się na rysunku 1.



Rys. 1: Diagram przypadków użycia

4 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne opisane są w tabeli 1 przy użyciu metody FURPS. Zawiera ona między innymi opis wymagań sprzętowych aplikacji oraz jej ograniczenia wydajnościowe. Celem analizy wymagań jest jednoznaczne określenie zakresu prac i ułatwienie nam podejmowania decyzji w zakresie wybieranych rozwiązań architektonicznych.

Obszar	**Opis**
Funkcjonalność	Aplikacja będzie potrzebować komputera z odpowiednim środowiskiem uruchomieniowym dla aplikacji oraz procesor graficzny. Ponadto, wymagana jest zainstalowana biblioteka OpenGL w wersji 2.0. Dostęp do sieci nie jest potrzebny dla poprawnego działania aplikacji.
Używalność	Aplikacja będzie działać w kilku oknach wyposażonych w przejrzysty interfejs dla użytkownika. Ponadto, zostanie dostarczona obszerna dokumentacja aplikacji, w której zostanie zawarta instrukcja obsługi.
Niezawodność	Aplikacja będzie działać na komputerze lokalnym, więc będzie dostępna cały czas. Wszelkie błędy powinny być obsłużone wewnątrz aplikacji.
Wydajność	Aplikacja będzie korzystała głównie z pamięci RAM, procesora i procesora graficznego. Maksymalna dopuszczalna liczba wierzchołków analizowanych drzew to 500 000. W przypadku dużych drzew (o liczbie wierzchołków bliskiej maksymalnej) powinna odpowiadać z opóźnieniem maksymalnie 1s, a dla drzew do 100 000 wierzchołków - z opóźnieniem do 200ms.
Wsparcie	Aplikacja będzie przeznaczona dla komputerów z systemami operacyjnymi Windows, Linux oraz MacOS. Aplikacja nie będzie rozszerzalna - wszystkie obliczenia i wizualizacje będą przeprowadzone wewnątrz jednej maszyny.

Tab. 1: Analiza FURPS

5 Harmonogram

5.1 Tabela

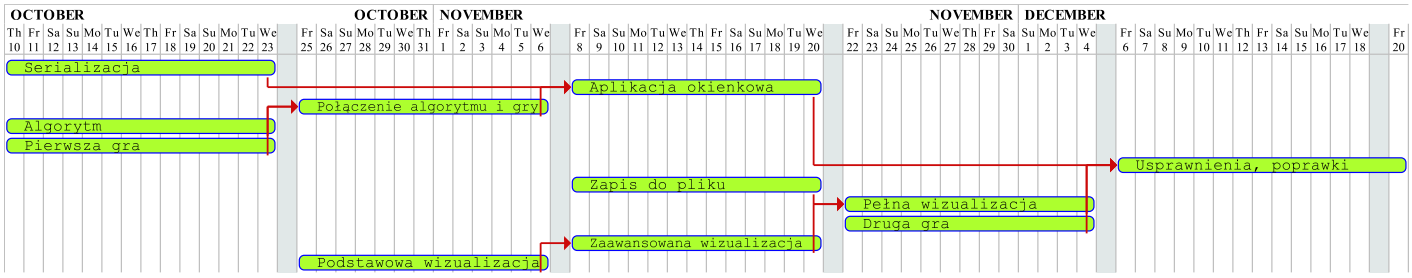
Pracę nad projektem opisuje harmonogram z tabeli 2. Tabela nie opisuje zależności między kolejnymi fazami rozwoju projektu, wyznacza jedynie planowe terminy ich zakończenia. Tworząc tabelę, staraliśmy się rozłożyć pracę regularnie na cały zaplanowany czas tworzenia projektu. Zostawiliśmy również ostatnie 2 tygodnie na doszlifowanie elementów, z którymi potencjalnie nie wyrobilibyśmy się na czas.

Deadline	**Przygotowane zadania**
24.10.2019	• Serializacja • Algorytm • Pierwsza gra
7.11.2019	• Podstawowa wizualizacja • Połączenie algorytmu i gry
21.11.2019	• Zaawansowana wizualizacja • Aplikacja okienkowa • Zapis drzew do pliku graficznego
5.12.2019	• Pełna wizualizacja • Druga gra
19.12.2019	• Usprawnienia, poprawki

Tab. 2: Harmonogram pracy

5.2 Diagram Gantta

Aby dobrze zwizualizować zależności pomiędzy kolejnymi etapami rozwoju aplikacji, sporządziliśmy diagram Gantta, przedstawiony na rysunku 2.



Rys. 2: Diagram Gantta

Jak widać na diagramie, jedynymi z najbardziej kluczowych punktów podczas rozwoju projektu są `połączenie algorytmu i gry` oraz elementy związane z `wizualizacją`. Elementy takie jak `druga gra` czy `zapis do pliku` są najmniej istotnymi punktami z punktu widzenia powodzenia projektu. Ostatnie 2 planowane tygodnie planujemy poświęcić na `usprawnienia i poprawki`, zostawiając sobie w ten sposób czas na dopracowanie komponentów, które zostały zaplanowane wcześniej i nie zostały stworzone wystarczająco dobrze.

6 Analiza ryzyka

	Czynniki pozytywne	**Czynniki negatywne**
Czynniki wewnętrzne	Atuty: • gotowość do ciężkiej pracy • motywacja • sumienność • dobrze przygotowany plan • wzajemne wsparcie i pomoc	Słabości: • ograniczona ilość czasu, który możemy poświęcić na projekt • życie studenckie • życie prywatne
Czynniki zewnętrzne	Szanse: • współpraca z promotorem	Zagrożenia: • ograniczenia płynące z wybranych technologii • wybrane technologie mogą nie działać w oczekiwany sposób • potencjalna trudność w połączeniu stworzonych komponentów aplikacji

Tab. 3: Analiza SWOT

Tabela 3 z analizą SWOT z grubsza określa różne czynniki, które będą miały kluczowy wpływ na przebieg naszego procesu tworzenia aplikacji. O ile jest dużo pozytywnych czynników, które na pewno pomogą nam w tym procesie, to tych negatywnych również nie brakuje i ważną dla nas sprawą jest znalezienie sposobu, aby się ich skutecznie wystrzec.

Wybraną przez nas technologią do napisania aplikacji (gier, algorytmu, wizualizacji) jest język programowania **Python**. Do implementacji gier będziemy posługiwać się biblioteką **PyGame** w stabilnej wersji, która jest do tego celu przeznaczona i jest popularna. Wizualizacja będzie wykorzystywać w znacznym stopniu bibliotekę **VisPy** (OpenGL), gdzie można tworzyć również kod w języku **C++** - najprawdopodobniej będzie to najbardziej problematyczna technologia. Python będzie używany w najnowszej stabilnej wersji - 3.7 - jest to zaufana technologia.

VisPy jest nową technologią, która jest wciąż rozwijana. Wybraliśmy ją ze względu na to, że
- współpracuje z GPU, co znakomicie przyspiesza rysowanie wizualizacji,
- posiada obszerną dokumentację,
- nie znaleźliśmy lepszej alternatywy (inne były najczęściej niezadowalające).

Ryzyko:
1. Nie znamy składni biblioteki OpenGL i jej nauka może zająć stosunkowo dużo czasu. Solucja: zaplanujemy i poświęcimy odpowiednią ilość czasu na testowanie i naukę potrzebnych nam funkcji.
2. VisPy może okazać się wadliwe. W takiej sytuacji szukalibyśmy sprytnych obejść, a w najgorszym wypadku znajdziemy inną bibliotekę.
3. Trudność w połączeniu komponentów - być może nie będą one działać w jednym oknie, połączenie może być bardzo czasochłonne lub sposób wymiany danych między komponentami będzie nieefektywny. Solucja: odpowiednio rozplanowany czas, który pozwoli na dobre zaplanowanie i wdrożenie optymalnej architektury.