# Wizualizacja drzewa stanów algorytmu UCT

Plan projektu

Patryk Fijałkowski
Grzegorz Kacprowicz

4 listopada 2019

**Streszczenie**

Dokument opisuje ...

## Historia zmian

| Wersja | Data | Autor(zy) | Zmiany |
|---|---|---|---|
| 1.0 | 3.11.2019 | PF, GK | stworzenie pierwszej wersji dokumentu |

# Spis treści

# 1 Architektura aplikacji

Aplikacja będzie podzielona na 5 oddzielnych modułów: algorytm, serializacja, wizualizacja, gra 1, gra 2... Ten odpowiedzialny za to, tamten za to... Takie rozwiązanie jest dobre bo...

# 2 Moduły aplikacji

## 2.1 Wizualizacja

Kluczowy moduł aplikacji. Udostępnia funkcjonalność wizualizacji dostarczonych drzew. Dla czytelnych wizualizacji, poczyniliśmy następujące założenia:

1. Wierzchołki drzewa nie mogą się przecinać.
2. Wierzchołki będą ustawione od góry w rzędach, a przynależność do rzędów będzie zależała od odległości wierzchołków od korzenia.
3. Wierzchołki mają być narysowane możliwie najwęziej.
4. Rodzic wierzchołka ma być wycentrowany względem swoich potomków.
5. Drzewa izomorficzne mają być wizualizowane tak samo, niezależnie od głębokości.
6. Potomkowie każdego z wierzchołków mają być od siebie równo oddalone.

Aby wyznaczyć układ wierzchołków na płaszczyźnie, spełniając powyższe 6 założeń, skorzystamy z usprawnionego algorytmu Walkera, który działa w czasie liniowym względem liczby wierzchołków. Algorytm, który zaimplementujemy, pochodzi z pracy "*Improving Walker's Algorithm to Run in Linear Time*" autorstwa C. Buchheim, M. Junger, S. Leipert.

Opisywany moduł udostępni również funkcjonalność przybliżania, oddalania oraz poruszania się po wizualizacji. Opisana interaktywność ma na celu umożliwić użytkownikowi dokładne zbadanie struktury drzewa oraz poszczególnych wartości w interesujących go wierzchołkach.

## 2.2 Algorytm

Jednym z kluczowych modułów jest moduł odpowiadający za algorytm Monte Carlo Tree Search. Będzie on udostępniał funkcjonalność wyznaczenia kolejnego ruchu na podstawie dostarczonego stanu gry. Opisywany moduł będzie odpowiadał za iteracyjne tworzenie drzewa stanów i przeszukiwanie go w celu wyznaczenia najbardziej korzystnego ruchu. Użytkownik będzie miał możliwość zmiany liczby iteracji algorytmu albo ograniczenie czasowe jego działania.

Gry, które ten moduł będzie obsługiwał muszą spełniać założenia algorytmu UCT. W rozdziale trzecim opisane jest dokładniej, jakie funkcjonalności należy zapewnić, by moduł "Algorytm" mógł wyznaczać kolejne ruchy danej gry.

## 2.3 Serializacja

Serializacja jest modułem odpowiadającym za zapisywanie drzew do plików formacie binarnym lub csv. Oba schematy są rekurencyjne, bo taka jest również struktura generowanych przez aplikację drzew. To oznacza, że w celu zapisania całego drzewa, wystarczy zserializować jego korzeń.

### Serializacja binarna

W serializacji binarnej przyjmujemy opisany niżej schemat.

- **liczba całkowita** - wartość liczby zakodowanej w U2 na 4 bajtach. Bajty liczby w kolejności little endian.
- **napis**:
  - liczba bajtów w napisie *(liczba całkowita)*
  - zawartość napisu kodowana w UTF8
- **liczba zmiennoprzecinkowa** - wartość liczby zakodowanej w IEEE754 na 64 bitach w kolejności little endian.
- **wierzchołek:**
  - nazwa stanu *(napis)*
  - $m$ - liczba węzłów potomnych *(liczba całkowita)*
  - $m$ powtórzeń następującego bytu:
    - nazwa ruchu *(napis)*
    - licznik odwiedzin *(liczba całkowita)*
    - dodatkowy licznik odwiedzin *(liczba całkowita)*
    - średnia wypłata *(liczba zmiennoprzecinkowa)*
    - węzeł potomny *(wierzchołek)*

### Serializacja do plików csv

W serializacji do plików csv przyjmujemy, że każdy kolejny wiersz odpowiada kolejnemu wierzchołkowi drzewa, a kolejne wartości opisujące wierzchołek oddzielamy przecinkami. Ostatnią wartością jest liczba wierzchołków potomnych. Jeśli wierzchołek $v$ ma $k$ potomków, to $k$ wierszy w pliku pod wierszem opisującym wierzchołek $v$ opisuje potomków $v$. Każdy wierzchołek serializujemy do wiersza postaci:

**R, O, O2, W, S, D**

Oznaczenia:

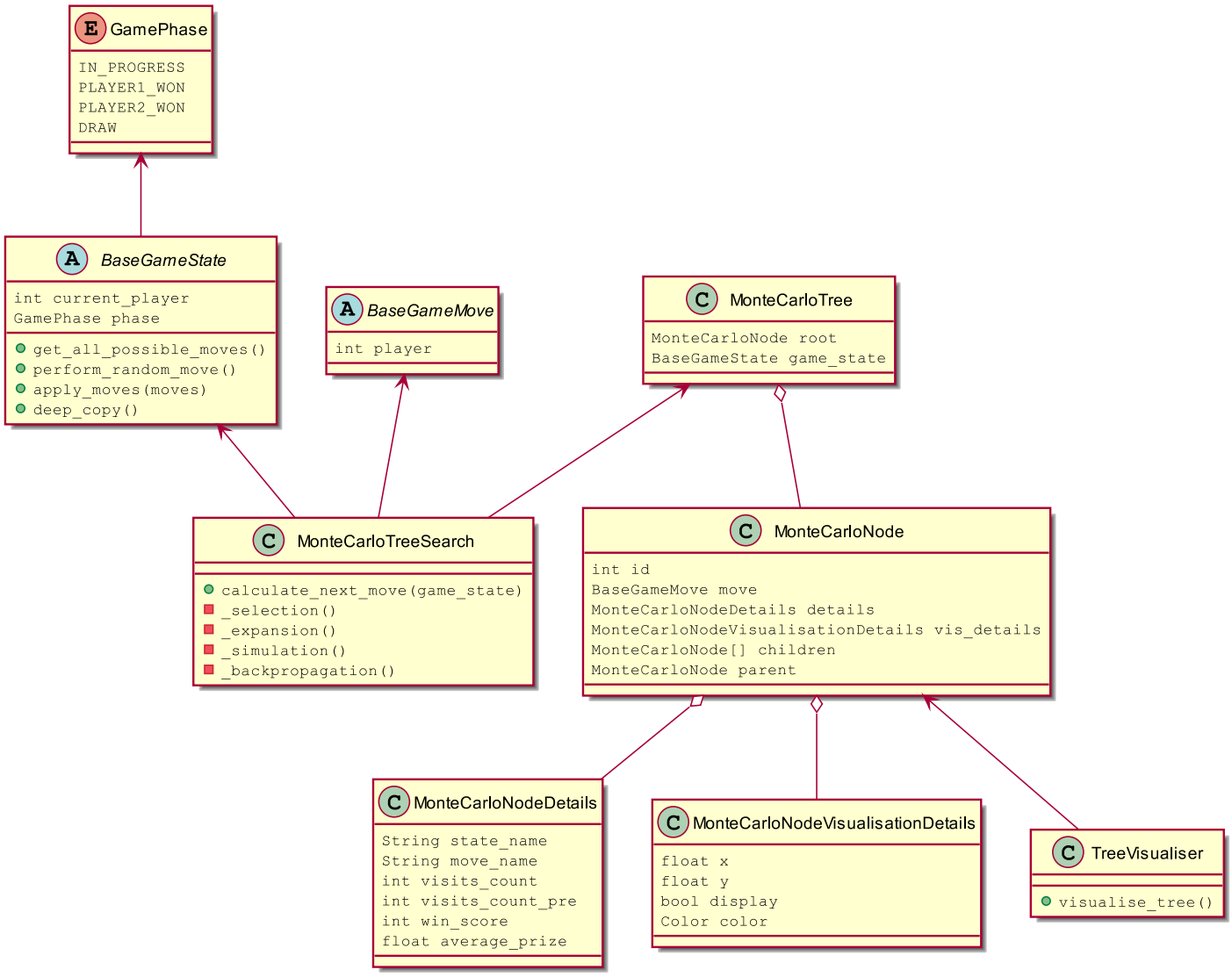
- R - nazwa ruchu
- O - licznik odwiedzin
- O2 - dodatkowy licznik odwiedzin
- W - średnia wypłata
- S - nawa stanu
- D - liczba wierzchołków potomnych

## 2.4 Gry

Aplikacja będzie udostępniała 2 gry planszowe, umożliwiające przetestowanie efektywności wizualizacji oraz algorytmu. Obie gry będą umożliwiały 3 tryby rozgrywki, opisane poniżej.

- **Człowiek kontra człowiek:** decyzje obojga graczy są podejmowane przez użytkownika aplikacji.
- **Człowiek kontra maszyna:** decyzje jednego z graczy są podejmowane przez użytkownika, natomiast drugi gracz podejmuje decyzje najoptymalniejsze z punktu widzenia algorytmu UCT.
- **Maszyna kontra maszyna:** decyzje obojga graczy są podejmowane przez algorytm.

# 3 Główne komponenty aplikacji




Rys. 1: Diagram klas głównych komponentów

Najważniejsze komponenty:

- MonteCarloTreeSearch, BaseGameMove, BaseGameState
- MonteCarloNode, MonteCarloNodeDetails, MonteCarloVisualisationDetails
- BaseSerializator, BinarySerializator, CsvSerializator

# 4 Interfejs użytkownika

Szkic głównego okna z opisem...

# 5 Wybrane technologie