Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

**Programowanie Komputerów**

**Sprawozdanie z projektu**

Autor: Michał Jagoda

Prowadzący: mgr Grzegorz Wojciech Kwiatkowski

Rok Akademicki: 2022/2023

Kierunek: Informatyka

Rodzaj Studiów: SSI

Grupa: 1

Sekcja: 2

## Opis Tematu

Tematem projektu jest wizualizacja algorytmów szukania drogi. Program możliwia użytkownikowi rysowanie przeszkód oraz porównywanieznalezionej różnymi sposobami drogi. Backend napisany został w języku c++, który skompilowany został to WASM, natomiast frontend napisany został przy użyciu HTML, CSS, JS, React.js oraz Boostrap**.**

## Specyfikacja zewnętrzna

Interfejs składa się z 4 plansz po których użytkownik przy pomocy lewego przycisku myszki może rysować przeszkody oraz zaznaczać punkty: początkowy oraz końcowy. Natomiast przy pomocy prawego przycisku myszy możliwe jest usuwanie przeszkód. Edytowanie którejkolwiek z plansz powodować będzie zmianę na pozostałych. Po naciśnięciu przycisku *start,* na każdej z plansz zostanie wyszukana droga przy użyciu jednego z algorytmów: A\*, Dijkstra, BFS, DFS, natomiast po naciśnięciu przycisku *clear* wszystkie plansze zostają wyczyszczone.

Interfejs użytkownika wykonany jest jako strona internetowa z wykorzystaniem HTML, CSS, JS, React oraz Bootstrap. Poniżej zamieszczony został rzeczywisty oraz przewidywany wygląd strony (*Rysunek 1 i 2*).

Obraz zawierający zrzut ekranu, tekst, diagram, Wykres

Opis wygenerowany automatycznie

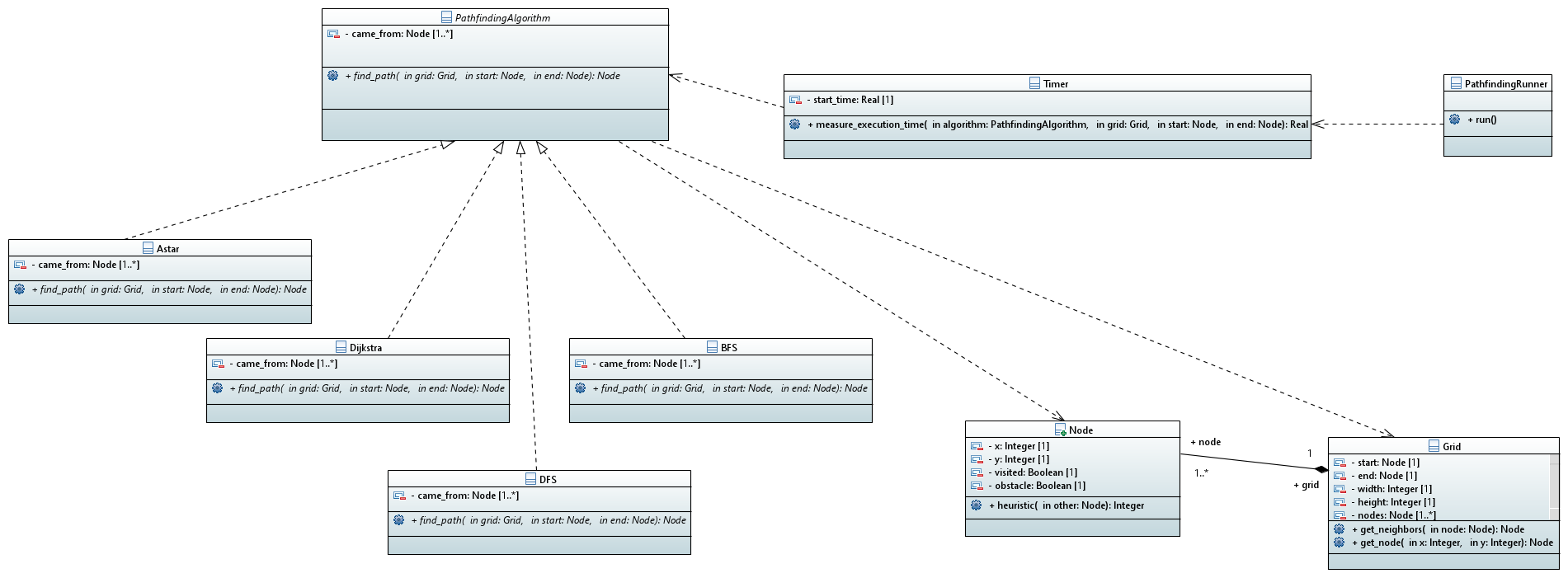
Rysunek 1

## Obraz zawierający zrzut ekranu, tekst, diagram, Prostokąt Opis wygenerowany automatycznie

Rysunek 2

## Najważniejsze klasy

Projekt napisany został w paradygmacie obiektowym. Główną częścią projektu są cztery klasy algorytmów wyszukiwania drogi. Dziedziczą one po klasie wirtualnej *PathfindingAlgorithm*. Umożliwiają one znalezienie drogi. Wymagają one dwóch klas: *Node* oraz *Grid*. Ta pierwsza reprezentuje pojedynczą komórkę grafu, natomiast druga cały graf. Mniej znaczące klasy to: *PathfindingRunner, Logger, Parser*. Pierwsza służy do wywoływania wszystkich algorytmów wielowątkowo. Druga, umożliwia logowanie wyników natomiast trzecia umożliwiawprowadzanie danych z pliku tekstowego. Diagram klas zamieszczono poniżej (Rysunek 2).



Rysunek 3Specyfikacja wewnętrzna

Główną częścią programu są cztery algorytmy wyszukiwania drogi. Ze względu na ułatwienie ich implementacji zdecydowano się na użycie pojemnika STL std::unordered\_set<Node\*> do przechowywania odwiedzonych już komórek. Zwiększa to przejrzystość kodu oraz jego szybkość.

Ze względu na specyficzny sposób działania biblioteki React.js algorytmy musiały zostać zaimplementowane niestandardowo. Aby projekt mógł poprawnie działać i aby możliwe było wizualizowanie, koniecznością było umożliwienie wykonania pojedynczego kroku algorytmu. Przez takie podejście możliwe było aktualizowanie stanu w aplikacji.

Aby napisany w c++ kod był dostępny na stronie internetowej został skompilowany do WASM. W tym celu wykorzystany został program *emscripten*, który udostępnia kompilator *emcc*. Niektóre bardziej skomplikowane typy musiały zostać zastąpione prostszymi, żeby później w aplikacji React.js możliwa była ich obsługa.

## Wykorzystywane tematy laboratoryjne

W projekcie wykorzystane zostały *modules*. Każda klasa jest osobnym modułem. Z biblioteki *ranges* wykorzystane zostały algorytmy operujące na kontenerach STL. Wielowątkowość została zaimplementowana przy użyciu biblioteki *thread*. Do walidacji danych wejściowych z pliku użyta została biblioteka *regex.*

Perspektywa rozwoju projektu

Kompilacja kodu napisanego w językach niższego poziomu (c/c++, Rust) do WASM daje możliwość do pisania szybkiego i efektywnego kodu dla przeglądarek. Powoduje to możliwość rozwoju strony o wizualizacje innych algorytmów (algorytmy sortowania, sieci neuronowe, algorytmy konwolucyjne).

Uwagi i wnioski

Użyte w projekcie tematy laboratoryjne pozwoliły na zapoznanie się z bardziej zaawansowanymi zagadnieniami, które jednak bardzo często przydają się podczas programowania. Pomimo ogromnych możliwości idących za kompilowaniem kodu do WASM, jest to zadanie trudne i czasochłonne przez co nie zawsze opłacalne. Całościowo jednak był to bardzo rozwijający projekt.