Marek Adamiec 160222
Klaudia Klębowska 160820
Marcin Lewandowski 160205
Michał Sulewski 160557

07.01.2019r.

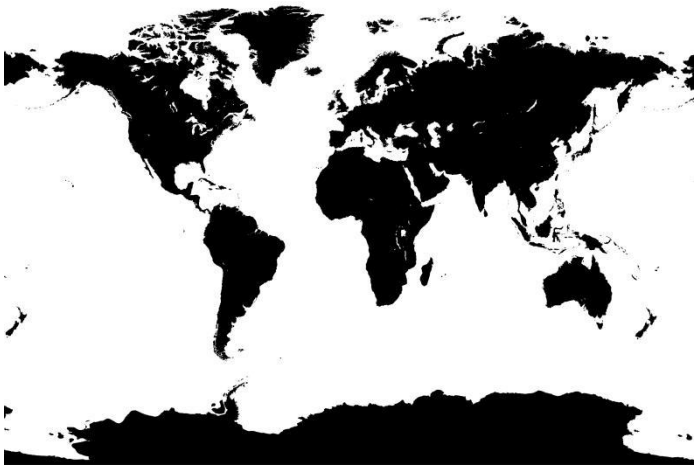
ROBOTY MOBILNE

Projekt grupowy - Raport

1.Temat projektu.

Numer	Tytuł	Opis	Wymagania
2	Grafowe metody planowania ścieżki	Porównanie **trzech** wybranych grafowych metod planowania ścieżki. Początek trasy u wybrzeży Japonii na wysokości Tokio, koniec u wschodnim wybrzeżu Anglii na wysokości Londynu. Wolną przestrzeń konfiguracyjną stanowią wszystkie zbiorniki wodne, kontynenty i wyspy to przeszkody.	1. Trzeba zaimplementować metodę PRM. 2. Porównać PRM z dwoma dowolnie wybranymi metodami: metodą dekompozycji przestrzeni, rastrową metodą grafu widoczności itd. (Wykład 6) 3. Program musi wizualizować zarówno końcowe rozwiązanie, jak i niewykorzystane wierzchołki grafu. 4. Metoda wyszukiwania w grafie powinna być optymalna.

2.Mapa świata.



Rys.1 Mapa świata użyta w projekcie

3.Opis metody PRM.

PRM czyli z angielskiego: *Probabilistic Roadmap* to prosta metoda probabilstyczna planująca ścieżkę z punktu startowego do punktu końcowego.

Wykonywana jest ona w następujących krokach:

1.Wybranie losowych punktów na obszarze.

2. Sprawdzenie czy punkty nie są położone na obszarach kolizyjnych (w tym przypadku na kontynentach).

3.Połączenie wybranych punktów z ich sąsiednimi punktami

4.Wyeliminowanie połączeń kolizyjnych (w tym przypadku przechodzących przez kontynenty).

5.Wybranie punktu startowego i końcowego i połączenie wybranych punktów ze sobą.

6.Znalezienie optymalnej ścieżki pomiędzy dwoma punktami.

4.Środowisko i język programowania.

Pierwszym etapem w projekcie było wybranie środowiska oraz języka programowania. Wybrany został program Visual Studio 2015, ze względu na dobrą znajomość programu przez wykonawców projektu i obecność graficznych bibliotek oraz język C#.

5.Opis eksperymentów i ich wyników.

Działanie programu opiera się na zaimplementowanym przez nas algorytmie PRM, tj. wylosowanie punktów na bitmapie, odrzuceniu punktów niespełniających kryteriów (znajdujących się na lądzie), utworzeniu krawędzi łączących wierzchołki sąsiadujące ze sobą, w taki sposób, aby nie przechodziły one przez ląd - udało się to uzyskać dzięki zastosowaniu metody Bresenhama, która w tym przypadku polegała na sprawdzeniu czy piksele, przez które przebiega linia łącząca dwa punkty, nie są w kolorze czarnym (nie są częścią lądu). W celu przeszukiwania grafu oraz znalezienia najkrótszej ścieżki z punktu startowego do punktu końcowego użyliśmy algorytmu Dijkstry. Podjęliśmy także próbę zaimplementowania algorytmu A* - kod znajduje się w komentarzu pod programem. Największą przeszkodą okazała się być instrukcja *foreach*, w której niemożliwe było dodawania elementów do listy, co było kluczowe do zadziałania tego algorytmu.

Etapy pracy i działanie programu:

1.Zapoznanie się z działaniem algorytmów PRM, A* oraz Djikstry.

2.Próba implementacji algorytmu A* oraz Djikstry na podstawie gotowych rozwiązań.

3. Przygotowanie prostego interfejsu do wyświetlania efektów algorytmu na mapie (zawierającego elementy: *PictureBox, Button i Label*).

Implementacja PRM:

5.Utworzenie losowych punktów, których współrzędne są przechowywane w tablicy dwuwymiarowej bufor.

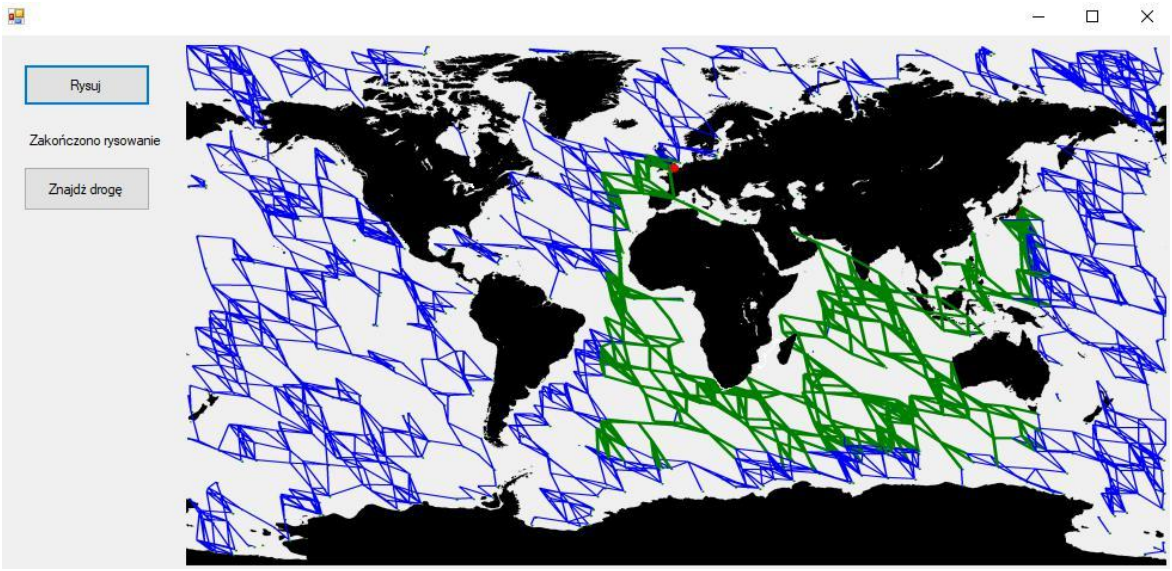
6.Narysowanie na mapie wylosowanych punktów (z wykluczeniem tych, które nie znajdują się na „wodzie") oraz punktów startu i końca (Tokio oraz Londyn).

7.Utworzenie funkcji liczącej odległość między dwoma punktami korzystającej z twierdzenia Pitagorasa.

8.Narysowanie zadanych ścieżek (z minimalną i maksymalną odległością) między sąsiadującymi ze sobą punktami.

9.Znalezienie drogi z punktu startowego (Tokio) do końcowego (Londyn).

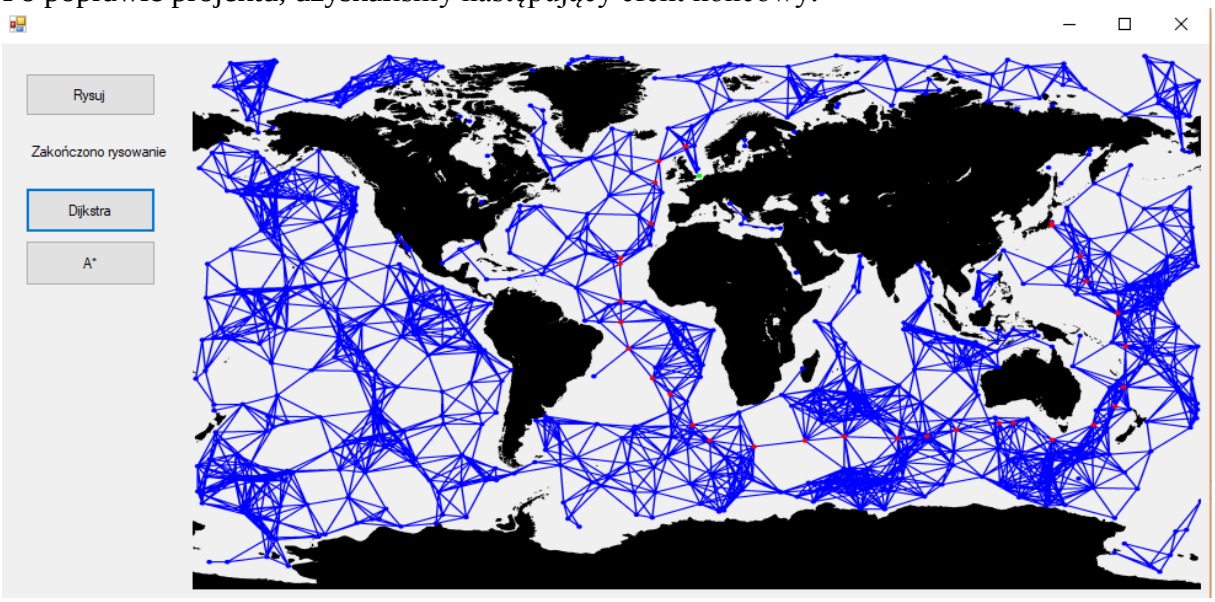
Początkowe działanie programu przedstawia się następująco:



Rys.2 Wygląd końcowego efektu

Aby uzyskać lepsze efekty, zostały stworzone struktury oraz listy, co znacznie poprawiło działanie programu. Punkt startowy oraz końcowy oznaczono za pomocą zielonych punktów, natomiast znalezioną ścieżkę oznaczono czerwonymi punktami, które pokrywają się z wylosowanymi wierzchołkami.

Po poprawie projektu, uzyskaliśmy następujący efekt końcowy:



Rys.3 Wygląd końcowego efektu (po poprawie)

Ważna uwaga: W przypadku, gdy nie ma połączenia między punktem startu i końca, program wypisuje w konsoli wiadomość związaną z brakiem dostępu do punktu końcowego (z brakiem drogi). W takim wypadku należy uruchomić program ponownie.

Wnioski:

Jak widać zadziałanie algorytmu znajduje satysfakcjonujące rozwiązanie tj. najkrótszą drogę pomiędzy zadanymi punktami. Po poprawie, połączenia wierzchołków nie przechodzą przez kontynenty.

Zwiększenie liczby losowanych punktów lub możliwej odległości pomiędzy wierzchołkami spowoduje mniejsze ryzyko wystąpienia błędu w programie, jednak wiąże się to z większą liczbą obliczeń oraz mniejszą przejrzystością.