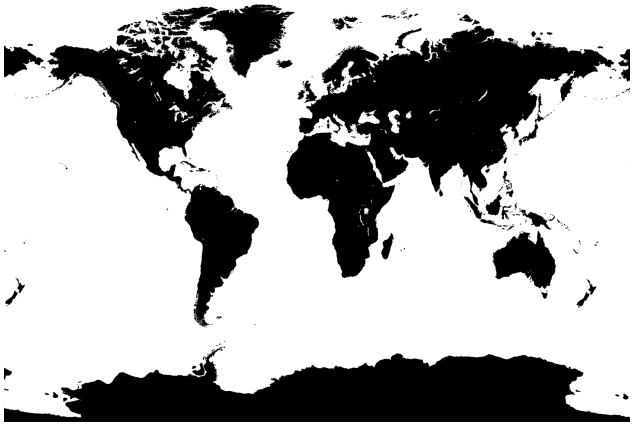
Filip Fydryszewski 160779
Karolina Tarnacka 160708
Kajetan Chrzanowski 160571

ROBOTY MOBILNE - PROJEKT
SPRAWOZDANIE

1. Wstęp

Celem projektu jest symulacja metod przeszukiwania ścieżki. W tym celu zaimplementowano metodę PRM będącą metodą planowania ścieżki ruchu robota, metodę rastrową znajdującą drogę za pomocą algorytmu A* oraz metodę RRT. Jako przestrzeń po której się poruszano wykorzystano mapę świata - rysunek nr 1. Punkt początkowy znajduję się przy Tokio, natomiast punkt końcowy znajduje się przy Londynie.

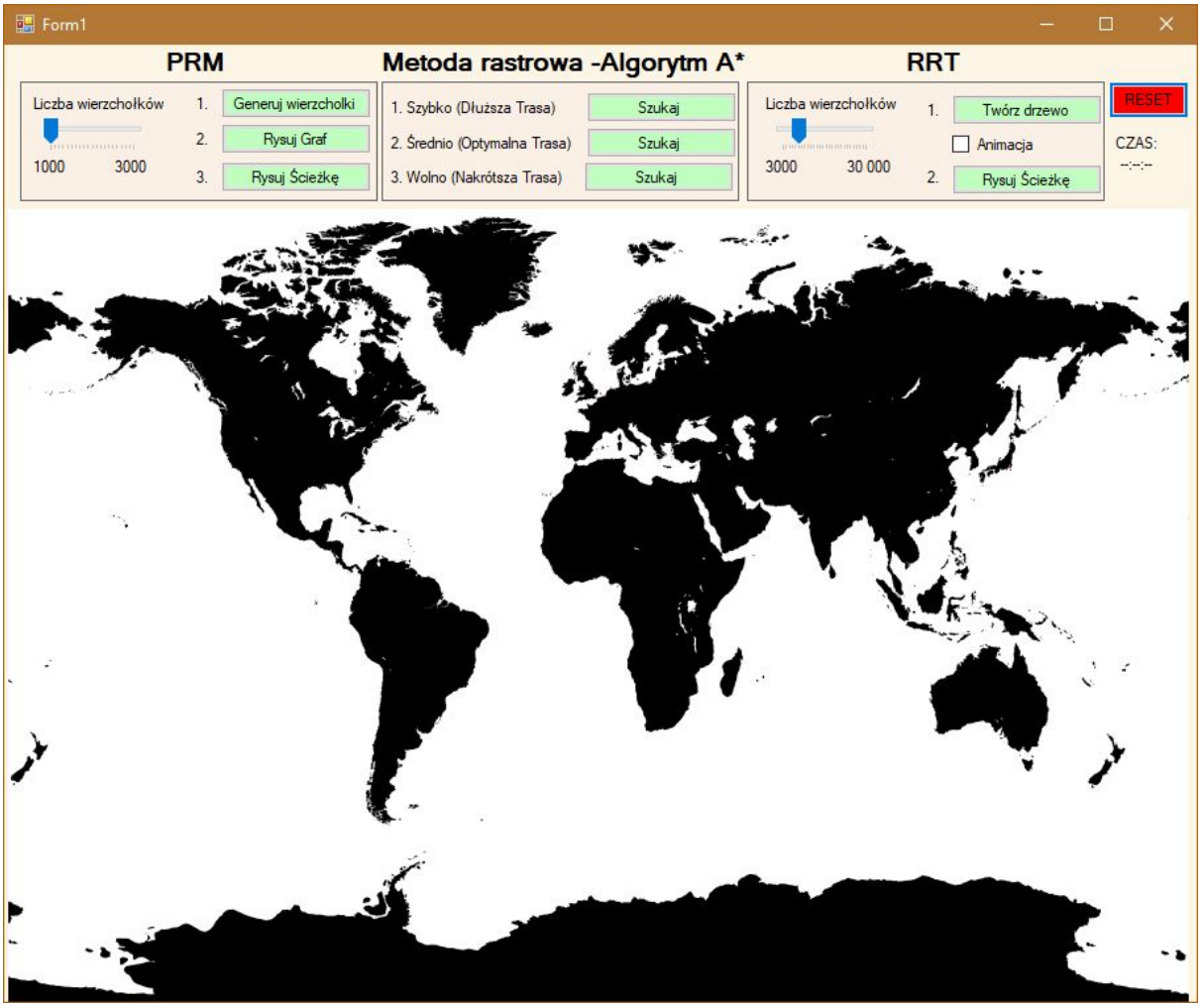


Rysunek nr 1 - mapa świata

Wolną przestrzeń konfiguracyjną stanowią zbiorniki wodne, natomiast przeszkodami były lądy.

2. Program

Program wykonano w języku C#, w środowisku Visual Studio 2017. Na rysunku nr 2 przedstawiono główny interfejs, wykorzystujący aplikację Windows Forms w ramach Microsoft .NET framework.



Rysunek nr 2 - Interfejs programu

Interfejs podzielono na 2 główne części: górną, gdzie znajdują się panele z przyciskami funkcyjnymi, oraz dolną, na której wyświetlane jest działanie programu.

W programie zaimplementowano kilka dodatkowych funkcji. Zarówno w metodzie PRM jak i RRT umożliwiono wybór liczby wierzchołków tworzących graf. Dodatkowo w metodzie RRT po zaznaczeniu pola wyboru ***animacja*** można zaobserwować proces tworzenia drzewa (przykładowa, stworzona animacja dostępna pod adresem:https://youtu.be/0WXedPqMiOM). Dla wszystkich 3 metod mierzony jest czas wyznaczania ścieżki, co wyświetlane jest pod przyciskiem RESET. Przycisk RESET umożliwia powrót do pierwotnego stanu programu. Dodatkowo funkcjonalności wprowadzono również dla metody rastrowa. Za pomocą 3 przycisków można analizować działanie tej metody pod kątem różnych przypadków, tj. szybkość algorytmu, czego efektem jest uzyskanie dłuższej trasy, optymalnej trasy oraz trasy najkrótszej, która wymaga najdłuższego działania programu.

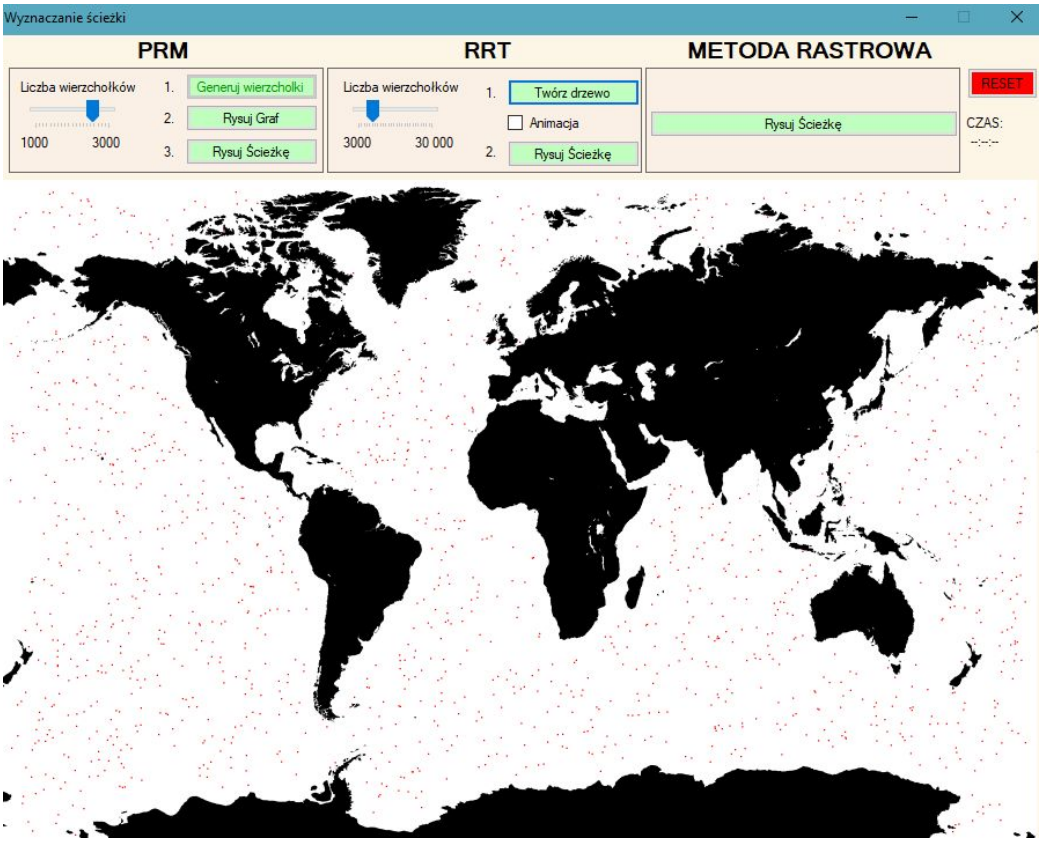
Każdy z algorytmów, przeszukiwania jak i rysowania grafu, został zaimplementowany samodzielnie.

2. Algorytm PRM.

Algorytm PRM polega na stworzeniu losowych punktów. Następnie algorytm odrzuca wierzchołki, które nie należą do przestrzeni konfiguracyjnej. Dla każdego pozostałego punktu określana jest ilość sąsiednich punktów, z którymi może się połączyć, lub maksymalna odległość do punktów najbliższych danemu wierzchołkowi. Znając pozycję sąsiadów dla każdego z wierzchołka tworzy się graf łącząc każdego sąsiada z jego wierzchołkiem.

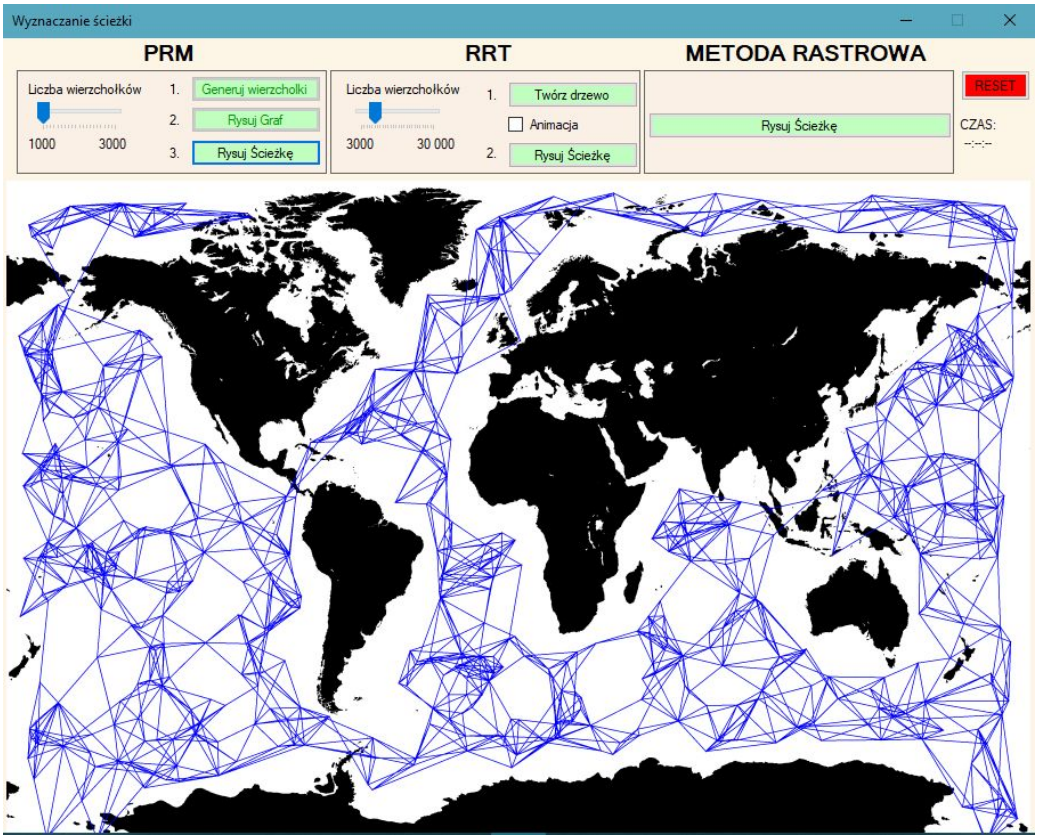
Algorytm PRM stworzono z myślą o planowaniu ścieżki dla robotów mobilnych. Użytkownik ustala liczbę losowanych wierzchołków, odległość oraz ilość sąsiadów dla danego wierzchołka. Wolna przestrzeń konfiguracyjna w łatwy sposób zapełniana jest przez algorytm, pamiętając o omijaniu przeszkód. Tak stworzony graf jest przeszukiwany przez algorytmy przeszukiwania grafu, takie jak algorytm Dijkstry oraz algorytm A*.

Losowe punkty (wierzchołki) zaznaczono kolorem czerwonym na rysunku nr 2.



Rysunek nr 2 - Wierzchołki

Na rysunku nr 3 przedstawiono wygląd otrzymanego grafu.



Rysunek nr 3 - Graf

3. Implementacja algorytmu.

Opisana w punkcie drugim metoda PRM zaimplementowana została w języku C# w środowisku Visual Studio jako aplikacja formularza Windows.

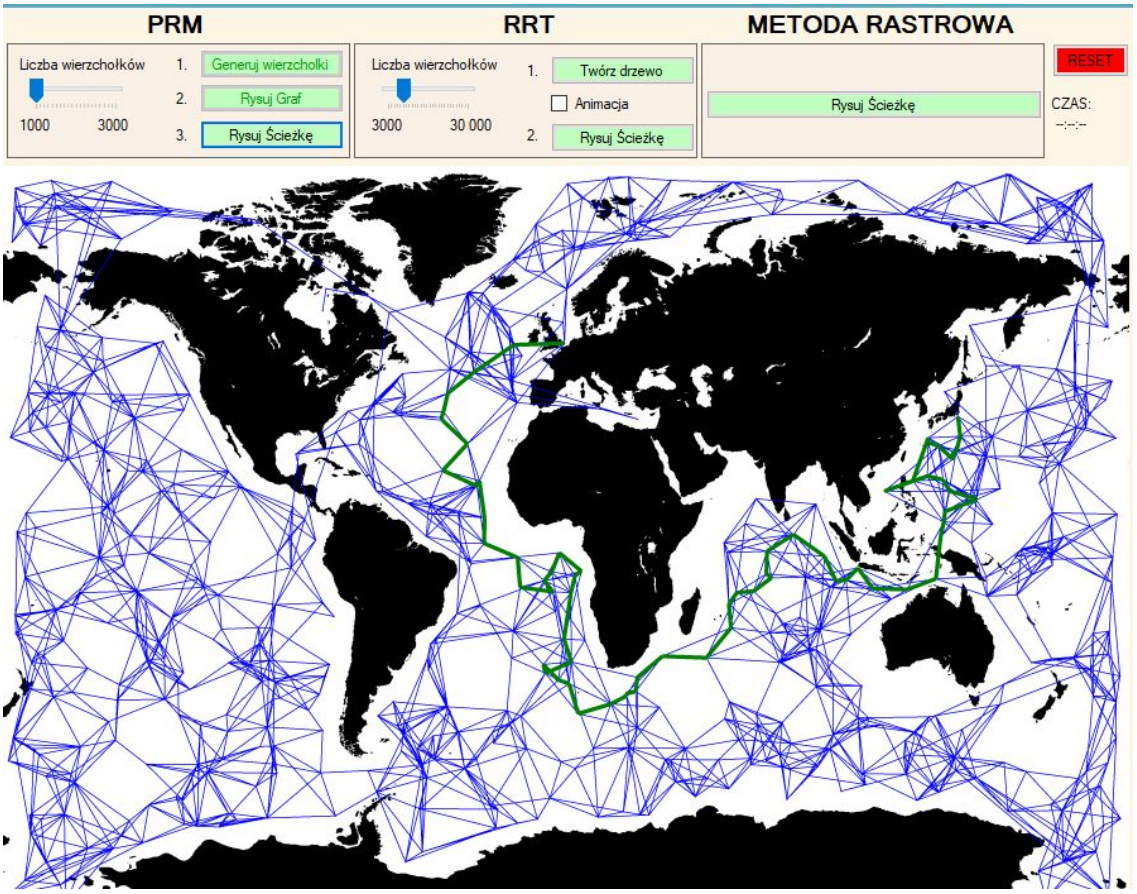
W celu przechowywania informacji o każdym z wierzchołków utworzono klasę *wierzcholek* zawierającą współrzędne losowanego wierzchołka, tablice odleglosci, pokonaną drogą, tablice sąsiadów oraz informacje czy był on odwiedzony podczas przeszukiwania grafu.

W pierwszej kolejności pseudolosowo wygenerowano 4000 wierzchołków, od razu sprawdzając czy nie są one umieszczone na kontynentach, i usuwając te które powyższego warunku nie spełniają. Wierzchołki generowane są dynamicznie oraz przechowywane w tablicy *wierzcholki.* Pierwsze dwa elementy tej tablicy stanowią punkt początkowy - okolicę Tokio, oraz punkt końcowy - okolicę Londynu.

Następnie dokonywane są między wierzchołkami połączenia. Sąsiedzi są przechowywani w tablicy sąsiadów będącej elementem klasy wierzcholek.

W dalszej części programu zakładamy minimalną odległość od sąsiada, którego możemy odwiedzić. Jeśli warunek został spełniony przechodzimy do sąsiada. Linią czarną zaznaczone są przejścia między sąsiadami. Uproszczonym warunkiem nie przecinania lini łączących wierzchołki jest to aby w połowie odległości między nimi nie było czarnych pikseli załadowanej mapy.

Linią zieloną natomiast można zaobserwować przejścia algorytmu A*.



Rysunek nr 3 - Szukanie ścieżki

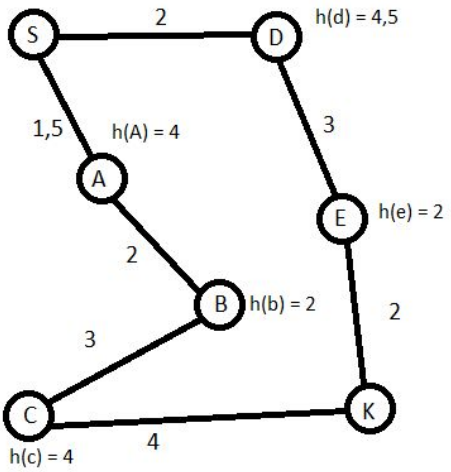
4. Metoda rastrowa - A*

Używając metody A*, która jest algorytmem heurystycznym używamy wag sąsiednich wierzchołków do poruszania się między nimi oraz odległości w linii prostej między wierzchołkiem sąsiednim, a wierzchołkiem docelowym. Dzięki czemu algorytm otrzymuje informację o oddalaniu się punktu pośredniego od punktu docelowego. Algorytm ten nie jest doskonały i nie daje gwarancji znalezienie najbardziej optymalnej drogi. Najlepiej radzi Sobie w momencie kiedy wagi dróg między sąsiadami są równe odległości między nimi, a nie kiedy drogi posiadają pewne cechy np. maksymalnej możliwej prędkości między drogami lub minimalnemu czasowi w jakim możemy się przedostać z jednego punktu do drugiego.

Droga początku w punkcie S(na podstawie Rys. nr 4):

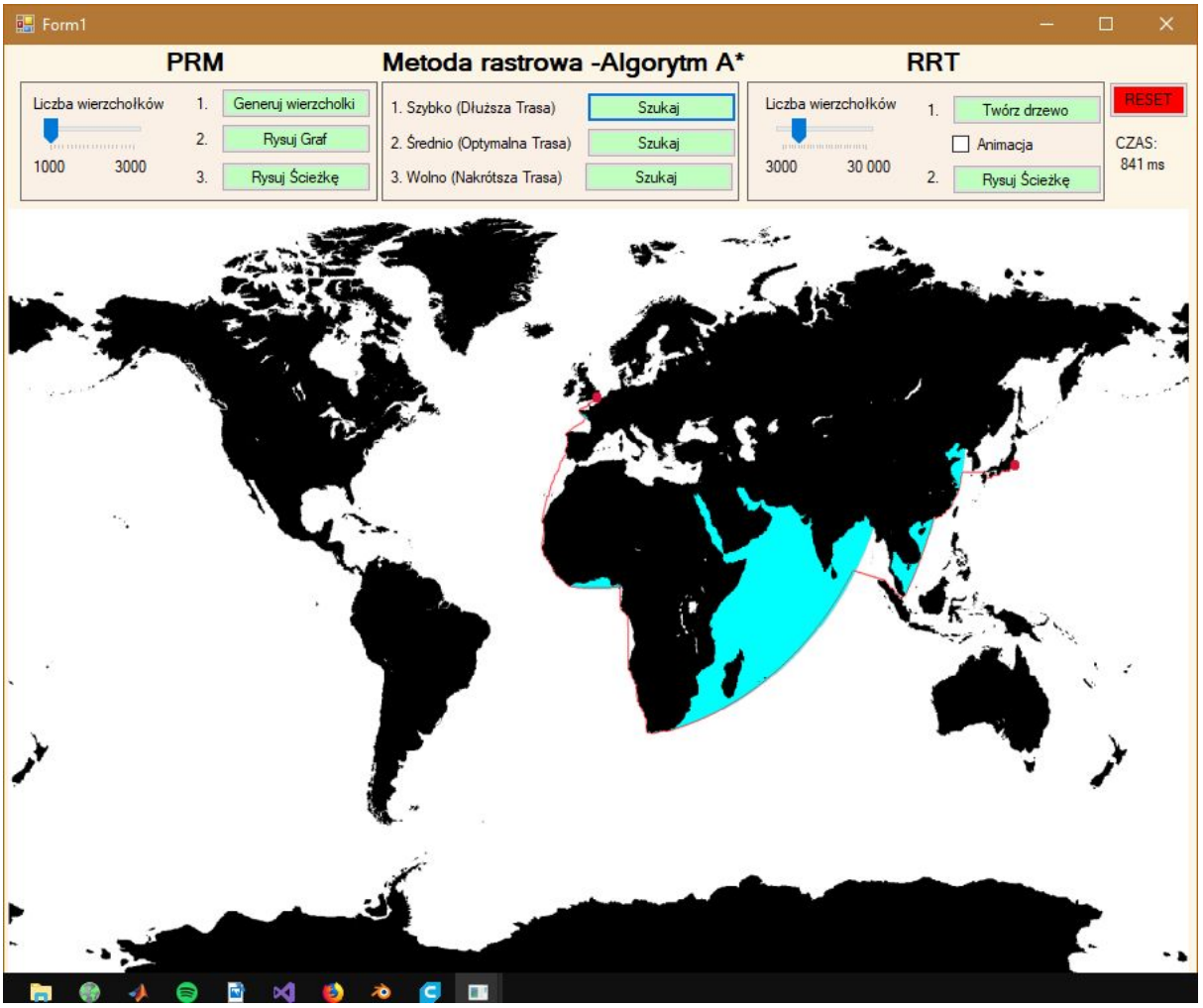
Droga:	**Otwarta:**	**Zamknięta:**
S^{0}	$A^{1,5+4}\ D^{2+4,5}$	**S**

$SA^{1,5}$	$D^{2+4,5}\ B^{3,5+2}$	SA
$SAB^{3,5}$	$D^{2+,4,5}\ C^{6,5+4}$	SAB
SD^{2}	$C^{6,5+4}\ E^{5+2}$	SABD
SDE^{5}	$C^{6,5+4}\ K^{7+2}$	SABDE
$SDEK^{7}$	-	SABDEK

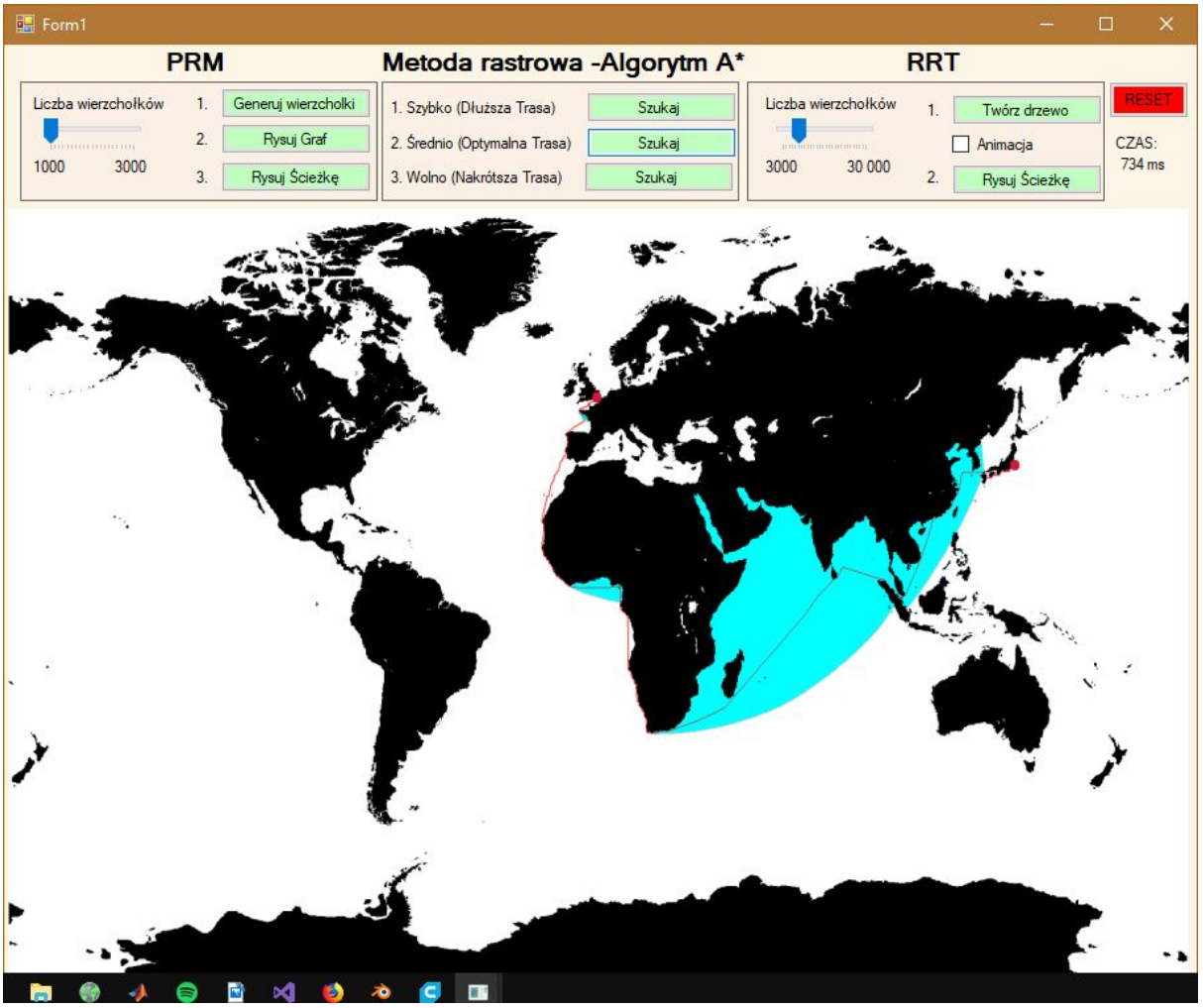


Rysunek nr 4 - Graf

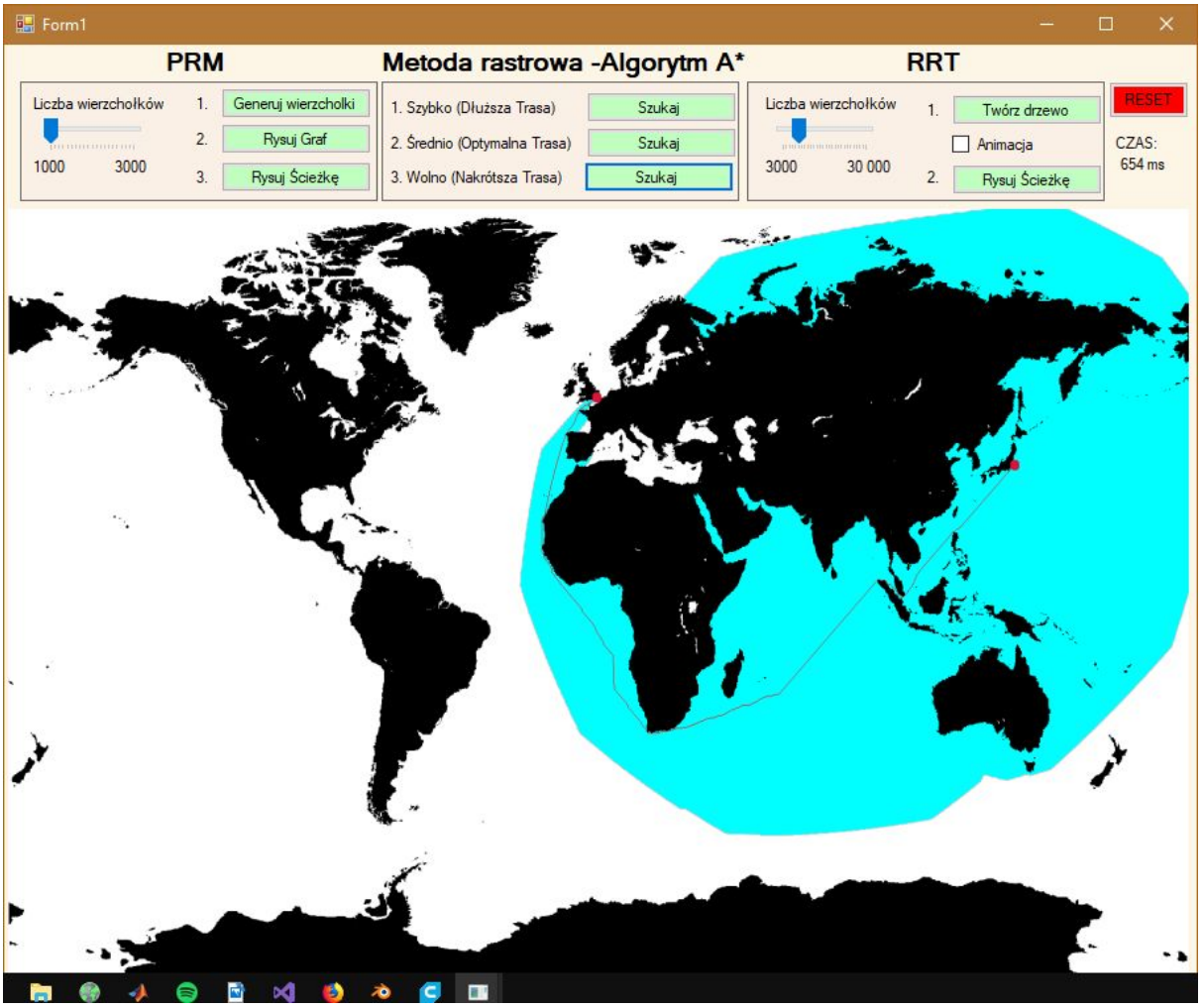
Efekty działania danej metody przedstawione zostały na rysunkach nr 5, 6 oraz 7. Trasa optymalna zaznaczona jest kolorem różowym, tak jak i punkt początkowy i końcowy. Kolorem niebieskim oznaczono potencjalne, zbadane piksele.



Rysunek nr 5 - Metoda rastrowa - najkrótszy czas



Rysunek nr 6 - Metoda rastrowa - Średnia

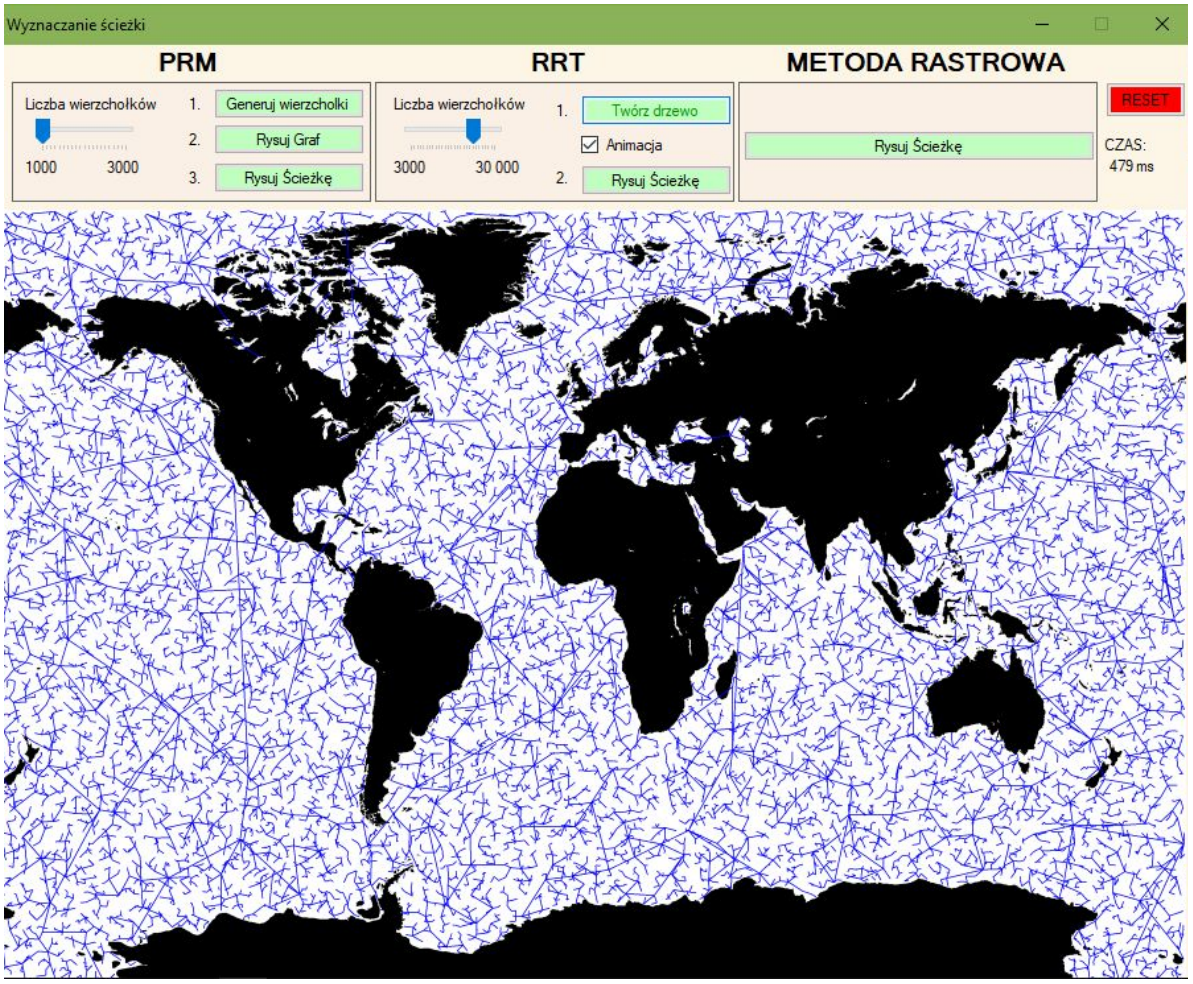


Rysunek nr 7 - Metoda rastrowa - Najkrótsza trasa

Porównując powyższe rysunki prezentujące 3 przypadki widać, jak duży obszar musi zostać przeszukany, aby znaleźć najkrótszą trasę. Niestety wiąże się to z dłuższym czasem oczekiwania.

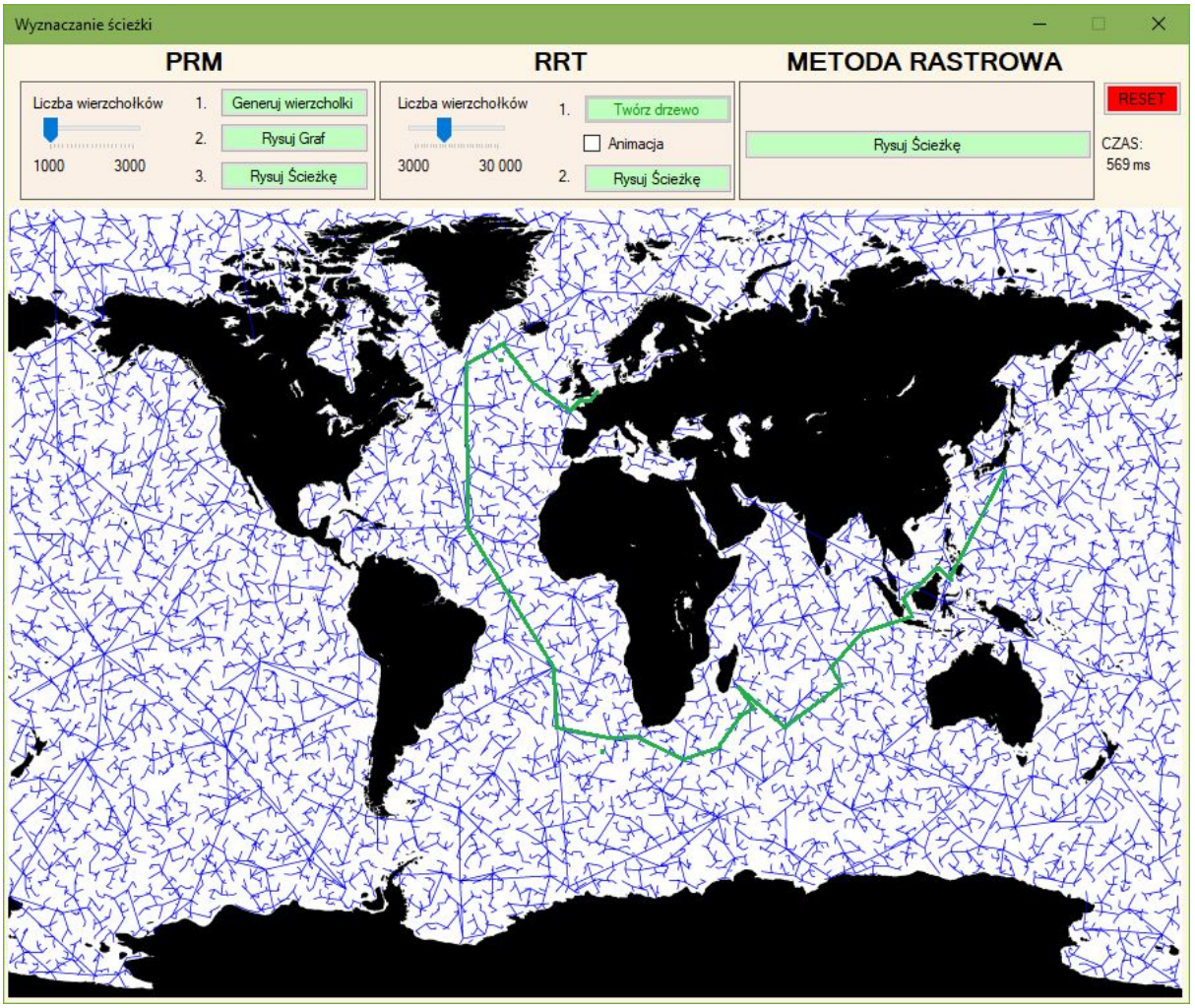
5. Algorytm RRT (Rapidly-exploring Random Tree)

Algorytm RRT jest trzecim zaimplementowanym algorytmem. Pozwala on na efektywne przeszukiwanie wielowymiarowych przestrzeni losowo tworząc drzewo wypełniające przestrzeń konfiguracyjną. Drzewo tworzone jest przyrostowo z losowo wybranych próbek. Sam algorytm polega na iteracyjnym wylosowaniu wierzchołka, znalezieniu najbliższego wierzchołka w istniejącym drzewie, a następnie połączeniu ich gdy są wystarczająco blisko siebie, lub stworzeniu wierzchołka znajdującego się na prostej między wylosowanym, a najbliższym, w określonej odległości. W międzyczasie sprawdzane jest czy stworzony wierzchołek nie zostanie umieszczony na lądzie, bądź go nie przecina. Działanie zaimplementowanej metody znajduje się na rysunku nr 8.



Rysunek nr 8 - Drzewo RRT

Tworzone drzewo przechowywane jest w tablicy punktów ***drzewo***. Rozmiar drzewa (ilość wierzchołków) można wybrać za pomocą slidera o zakresie 3 000 - 30 000. Na rysunku nr 9 znajduje się wyszukana ścieżka.



Rysunek nr 9 - Ścieżka RRT

6. Podsumowanie

Omawiany projekt pozwolił nam na wykorzystanie wiedzy teoretycznej w praktyce. Przedstawiane metody wyznaczania drogi do celu okazały się być różnorodne w implementacji, jak i późniejszym wykorzystaniu przy przeszukiwaniu grafu. Każda z metod posiada różne modyfikacje usprawniające szybkość działania, dokładność, bądź tworząca “bezpieczniejszą” ścieżkę, są one jednak znacznie trudniejsze w implementacji.

Kierując się teorią, za pomocą metody rastrowej powinniśmy otrzymać najbardziej optymalne wyniki. Jest to spowodowane tym, że graf składa się z komórek o identycznym kształcie, które są pikselami używanej bitmapy. Wynikiem końcowym przeszukiwania tak zbudowanego grafu są spójne fragmenty ścieżki wiodącej tuż przy linii brzegowej, co niemożliwe jest przy metodach RRT oraz PRM ze względu na losowość wierzchołków. Na podstawie wyników otrzymanych za pomocą tej metody wyraźnie widać, że dzięki niej otrzymujemy najkrótszą możliwą trasę z Tokio do Londynu.

Dodatkowo implementując metodę rastrową wprowadzono dodatkowe usprawnienia działania programu, które pokazują, że w danym projekcie metoda ta jest najoptymalniejsza.

Wyniki otrzymane za pomocą przeszukiwania grafu PRM oraz grafu RRT są podobne. Główną niedogodnością była losowość wierzchołków, jak i ich sąsiadów, co przy przeszukiwaniu grafu okazało się być problematyczne. Dodatkowo metody te wymagają tworzenia nowego grafu według określonych wytycznych, co zwiększa nakład pracy.