Szymon Nowak-Trzos 411246
Informatyka rok II, gr.2 środa nieparzysta 13:00-14:30

Algorytmy Geometryczne – Ćwiczenie 3 Sprawozdanie

1. Wstęp

1.1. Cel Ćwiczenia

Celem ćwiczenia była implementacja algorytmów do stwierdzania czy wielokąt jest y-monotoniczny oraz do triangulacji takiego wielokąta.

1.2. Użyte Algorytmy

W ramach ćwiczenia zaimplementowane zostały algorytmy:

- podział wierzchołków na początkowe, końcowe, łączące, dzielące i prawidłowe,
- sprawdzanie czy wielokąt jest y-monotoniczny,
- triangulacja wielokąta monotonicznego.

1.3. Użyte Narzędzia

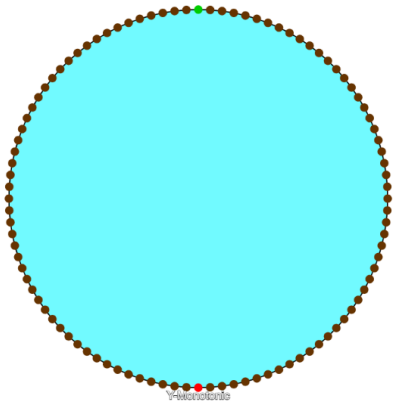
Na potrzeby ćwiczenia zostało stworzone własne narzędzie graficzne obsługujące animacje, dodawanie punktów, dodawanie wielokątów oraz zapisywanie i odczytywanie wielokątów z pliku. Narzędzie to napisane zostało w języku JavaScript przy pomocy biblioteki p5.js Narzędzie oraz cały kod ćwiczenia znajduje się w edytorze online pod adresem https://editor.p5js.org/Szyntos/sketches/wwrTFdY97. Algorytm został uruchamiany na procesorze Intel Core i5-1135G7 2.40GHz.

2. Szczegóły wykonywania ćwiczenia

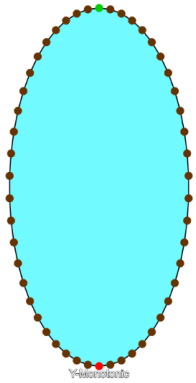
2.1. Zbiory Danych

Algorytm został testowany na następujących zbiorach:

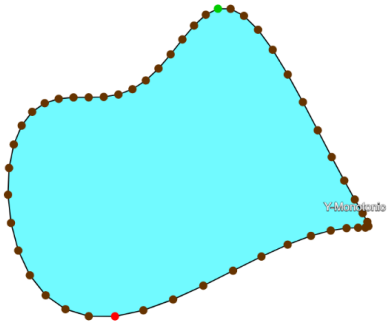
- 100 punktów leżących na okręgu;
- 50 punktów leżących na elipsie;
- 2 zbiory o liczności 50 punków utworzone przez równanie parametryczne:
  $x = R/k_1 * (\frac{\cos(n_1*\alpha)}{a_1} + \frac{\cos(m_1*\alpha)}{b_1} + \frac{\sin(l_1*\alpha)}{c_1})$, $y = R/k_2 * (\frac{\cos(n_2*\alpha)}{a_2} + \frac{\cos(m_2*\alpha)}{b_2} + \frac{\sin(l_2*\alpha)}{c_2})$
  O różnych wartościach k, a, b, c, m, n, l;
- Kwadrat o liczności 50 punków zadany równaniem parametrycznym:
  $x = R * (|\cos(\alpha)| * \cos(\alpha) + |\sin(\alpha)| * \sin(\alpha)), y = R * (|\cos(\alpha)| * \cos(\alpha) - |\sin(\alpha)| * \sin(\alpha))$;
- Połowy wybranych zbiorów.



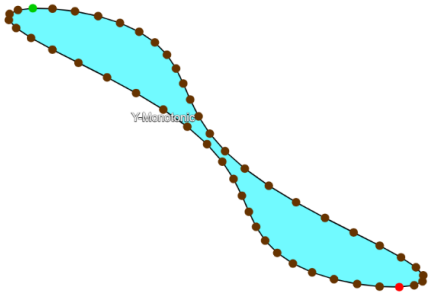
Rysunek 1.



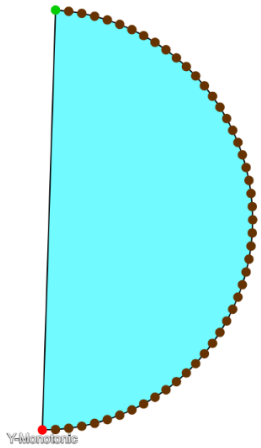
Rysunek 2.



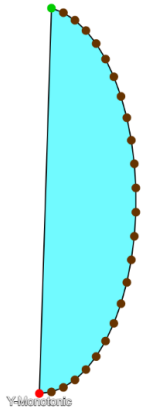
Rysunek 3.



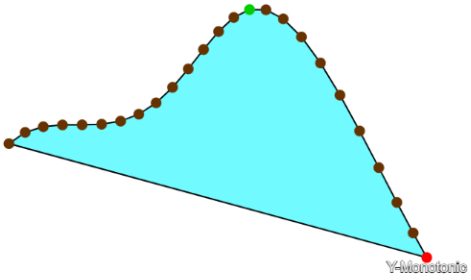
Rysunek 4.



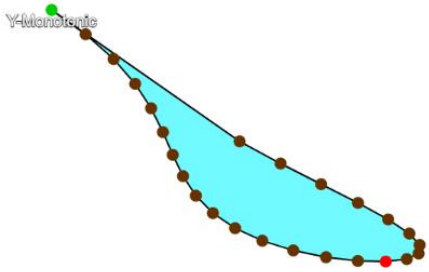
Rysunek 5.



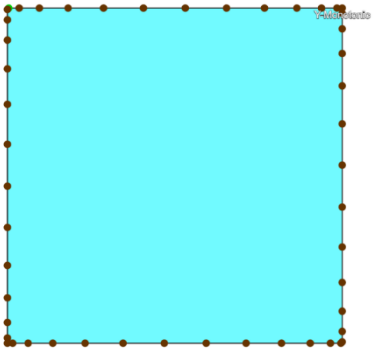
Rysunek 6.



Rysunek 7.



Rysunek 8.



Rysunek 9.

Zbiory zostały wybrane tak, aby przetestować działanie algorytmu w przypadku:

- Dużej gęstości punktów
- Punktów leżących prawie poziomo
- Punktów współliniowych
- Równo oddalonych od siebie punktów
- Gładkich oraz ostrych kształtów

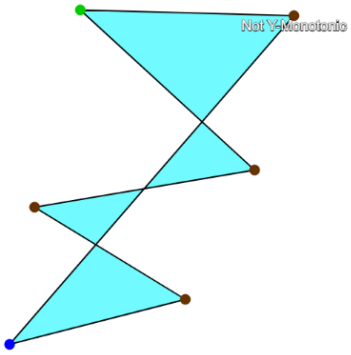
Algorytm poprawnie wyznacza triangulację powyższych zbiorów.

2.2. Struktury Danych

W narzędziu zaimplementowane zostały następujące klasy:

- Point – posiada współrzędne punktu oraz jego typ
- Line – posiada 2 punkty, początkowy oraz końcowy
- PointsCollection – posiada listę punktów oraz typ
- LinesCollection – posiada listę linii
- Scene – posiada 2 listy obiektów PointsCollection oraz 2 listy obiektów LinesCollection

Każdy wielokąt jest przechowywany jako obiekt PointsCollection. Dany wielokąt składa się z listy punktów uporządkowanych przeciwnie do ruchu wskazówek zegara. Struktura ta wynika ze sposobu rysowania wielokąta w bibliotece p5.js oraz z natury zadania. Wadą tej struktury jest możliwość zadania nieprawidłowego wielokąta (rys. 10).



Rysunek 10.

Triangulacja zaś jest przechowywana całkowicie w obiekcie LinesCollection jako lista krawędzi składających się na triangulację.

2.3. Algorytmy

2.3.1. Algorytm Podziału Wierzchołków

Algorytm działa następująco:

- Przechodzimy po tablicy punktów,
- Wybieramy 3 punkty – **a**, **b**, **c** – badamy klasyfikacje punktu środkowego,
- Zmieniamy typ danego punktu biorąc pod uwagę wielkość kąta **abc** oraz względne położenie punktów,
- Kąt jest determinowany poprzez własną funkcję obliczającą wyznacznik macierzy 3x3.

Ze względu na strukturę danych, tablica punktów jest już posortowana przeciwnie do ruchu wskazówek zegara, więc punkty **a**, **b** oraz **c** są następujące po sobie.

Kolorowanie punktu jest uzyskane przez metodę **draw()** klasy Point, która bierze pod uwagę typ punktu.

Algorytm wykonuje się od razu po wprowadzeniu nowego wielokąta.

2.3.2. Algorytm Sprawdzający Monotoniczność Wielokąta

Algorytm zakłada poprzedni podział punktów w wielokącie.
Jeśli jakikolwiek punkt jest łączący lub dzielący to wielokąt nie jest y-monotoniczny, a w przeciwnym wypadku jest, co jest wpisywane jako typ obiektu PointsCollection.

Algorytm wykonuje się od razu po wprowadzeniu nowego wielokąta.

2.3.3. Algorytm Triangulacji Wielokąta Monotonicznego

Algorytm został zaimplementowany wg instrukcji przedstawionych na wykładzie. Minimalny oraz maksymalny punkt został przydzielony zarówno do prawej i do lewej części wielokąta, przez co algorytm triangulacji musiał być odpowiednio zmieniony. Wynikiem algorytmu jest obiekt LinesCollection przechowujący wszystkie krawędzie triangulacji danego wielokąta

2.4. Animacje

Animacje zostały zaimplementowane poprzez zatrzymywanie działania algorytmu w kluczowych momentach oraz rysowaniu niepełnej triangulacji.

3. Podsumowanie

Poprawne wykonanie ćwiczenia wymagało zaimplementowania nowych struktur danych przechowujących wielokąty oraz triangulację. Należało również dokładnie przyjrzeć się algorytmowi triangulacji oraz sprawdzania monotoniczności tak, aby poprawnie je zaimplementować na nowo dodanych strukturach danych. Algorytm działa poprawnie na dobrze zadanych wielokątach, aczkolwiek potrafi się mylić podczas procesu zadawania wielokąta ze względu na wadę opisaną w punkcie 2.2.