ALGORYTMY GEOMETRYCZNE

Sprawozdanie z ćwiczenia 3.

Bartosz Kucharz

Grupa nr 1

1. Specyfikacja techniczna:

Komputer na którym wykonywano obliczenia:

System operacyjny: Windows 10 x64
Procesor: Intel® Core™ i5-5300U CPU @ 2.30Ghz
Pamięć RAM: 8GB

Obliczenia zostały wykonywane w języku Python 3 z wykorzystaniem bibliotek: matplotlib oraz zaproponowanego narzędzia graficznego.

2. Opis ćwiczenia

Ćwiczenie polegało na dostosowaniu aplikacji graficznej , aby można było zadawać wielokąty przy użyciu myszki. Należało zaimplementować algorytmy: sprawdzania y-monotoniczności, klasyfikacji wierzchołków oraz triangulacji oraz przetestować ich działanie dla różnych zadanych wielokątów.

3. Zestawy danych

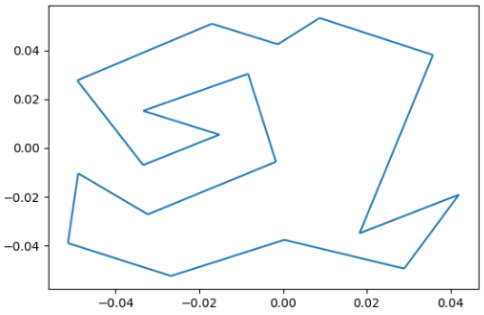
Program korzysta z dostarczonego narzędzia graficznego. Wielokąty można zadawać myszką, wprowadzając kolejne punkty wielokąta w kierunku przeciwnym do ruchu wskazówek zegara. Następnie figura jest zapisywana do pliku json w postaci listy krawędzi. Na potrzeby algorytmów została zaimplementowana funkcja konwertująca listę krawędzi na listę punktów.

Do testów wybrano siedem figur:

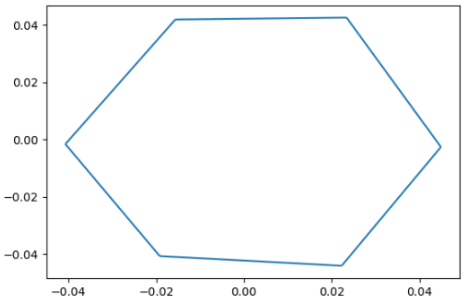
1) niemonotoniczny wielokąt przedstawiony na wykładzie
2) sześciokąt
3) okrąg
4) lotka skierowana w dół
5) lotka skierowana w górę
6) odwrócona lotka
7) proporzec

Wybranie niemonotonicznej figury miało na celu sprawdzenie poprawności funkcji sprawdzającej y-monotoniczność oraz funkcji klasyfikującej punkty wielokąta.

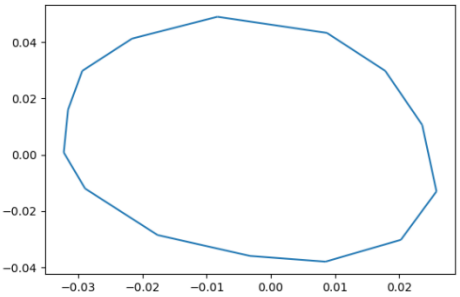
Następne dwie figury testują podstawowe przypadki triangulacji figur wypukłych. Pozostałe cztery to figury wklęsłe. Lotki testują przypadek w którym trójkąty „odcinane” są z tej samej gałęzi figury lub z przeciwnej. Sprawdzana jest również niezależność działania algorytmu od kierunku w którym figura jest zwrócona. Proporzec testuje „odcinania” trójkątów z obu gałęzi.



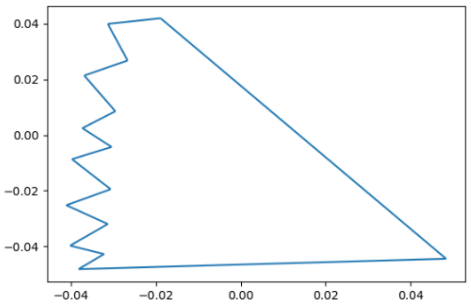
Rys. 1 Figura 1. Figura niemonotoniczna



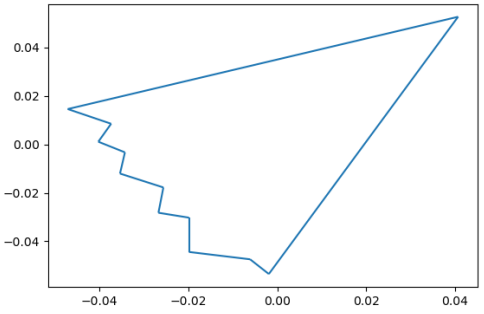
Rys. 2 Figura 2. Sześciokąt



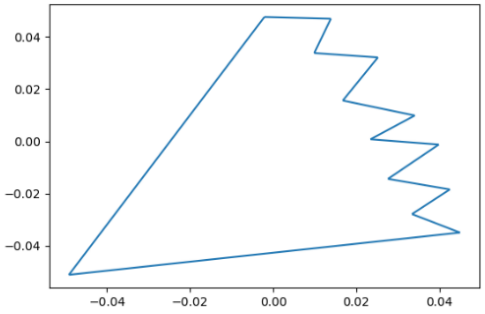
Rys. 3 Figura 3. Okrąg



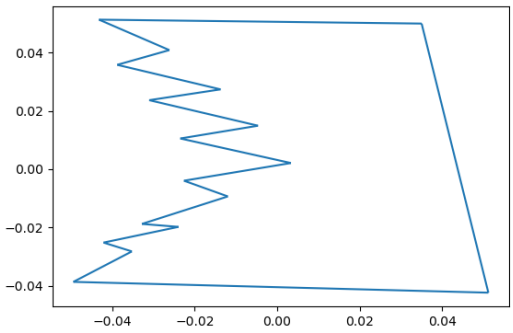
Rys. 4 Figura 4. Lotka skierowana w dół



Rys. 5 Figura 5. Lotka skierowana w górę



Rys. 6 Figura 6. Odwrócona lotka



Rys. 7 Figura 7. Proporzec

4. Y-monotoniczność

Implementacja algorytmu sprawdzającego y-monotoniczność zadanego wielokąta polegała na wyznaczeniu punktów o najmniejszej oraz największej współrzędnej y. Następnie przechodząc od najwyższego do najniższego punktu po kolejnych wierzchołkach lewej gałęzi sprawdzano czy każdy następny punkt jest położony nie wyżej niż poprzedni. Czynność powtórzono dla prawej gałęzi. Jeśli znaleziono parę punktów, które nie spełniały tej zależności zwracano informację że dany wielokąt jest niemonotoniczny. W przeciwnym wypadku wielokąt był y-monotoniczny.

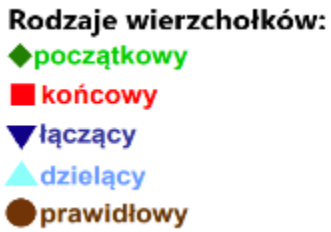
5. Klasyfikacja wierzchołków

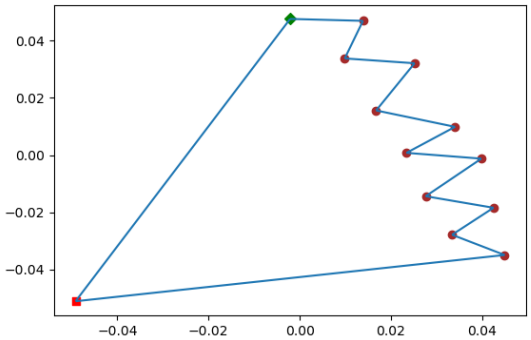
Klasyfikacja wierzchołków wielokąta polegała na jednorazowym przejściu po liście punktów, porównania położenia każdego punktu z jego sąsiadami i zapisania go do odpowiedniej listy. Położenia punktów były określane za pomocą wyznacznika 3x3 własnej implementacji.

Rodzaje punktów i kryteria podziału:

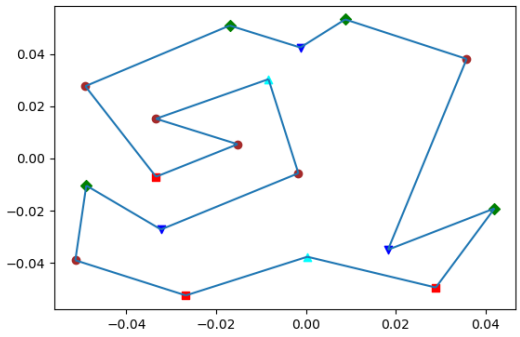
- początkowy, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny < π,
- końcowy, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny < π,
- łączący, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny > π,
- dzielący, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny > π,
- prawidłowy, w pozostałych przypadkach (ma jednego sąsiada powyżej, drugiego – poniżej).

W zadanych wielokątach klasyfikacja punktów była poprawna. Figura 1. Zawierała wszystkie rodzaje punktów. Natomiast w pozostałych wielokątach, które były y-monotoniczne wszystkie punktów oprócz najwyższego i najniższego względem osi y były prawidłowe.





Rys. 8 Klasyfikacja wierzchołków dla figury 6.



Rys. 9 Klasyfikacja wierzchołków dla figury 1.

6. Triangulacja

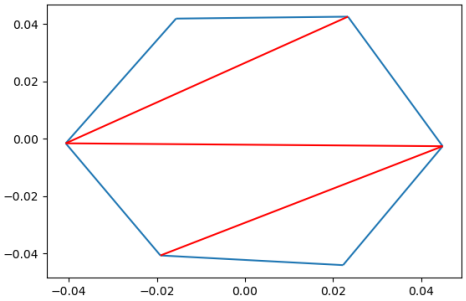
Algorytm jako dane wejściowe przyjmuję wielokąt w postaci listy punktów zadanych w kolejności przeciwnej do ruchu wskazówek zegara. Taka struktura ułatwia przypisanie punktom gałęzi na której się znajdują oraz ich sortowanie. Łączenie wierzchołków polega na zapisie pary punktów do listy krawędzi. Wynik zapisany w liście krawędzi może być użyty do wizualizacji bez potrzeby dodatkowej konwersji.

Algorytm na początku dzieli punkty na te znajdujące się w lewej i prawej gałęzi. Aby to zrobić znajduje najwyższy oraz najniższy punkt w liście wierzchołków wielokąta, Następnie korzysta z uporządkowanego ułożenia punktów w liście przechodząc po następnych wierzchołkach aż do napotkania punktu najniższego, zapisując do listy wynikowej kolejne wierzchołki na odpowiadających indeksach wraz z informacją znajdowania się na lewej gałęzi. Czynność powtórzono znowu wychodząc od najwyższego punktu tym razem dla malejących indeksów dla gałęzi prawej.
Następnie otrzymana lista krotek (punkt, gałąź) została posortowana względem współrzędnej y.
Inicjowany jest stos na którym początkowo znajdują się dwa najwyższe punkty. Wykonywana jest główna pętla algorytmu rozważająca każdy wierzchołek z osobna w posortowanej kolejności.

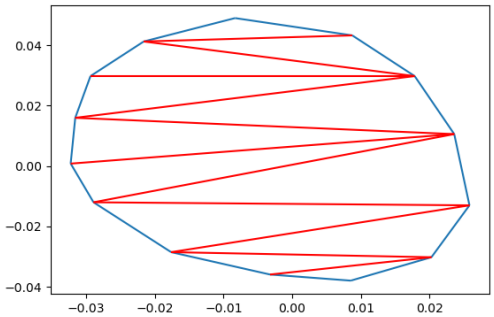
Jeśli dany wierzchołek znajduję się na przeciwnej gałęzi niż ostatni ze stosu to łączymy go ze wszystkimi wierzchołkami ze stosu oprócz ostatniego.
Następnie na stos dodawane są dwa ostatnio rozważane wierzchołki.

W przypadku gdy rozważany wierzchołek znajduję się na tej samej gałęzi co ostatni ze stosu zdejmujemy po kolei wierzchołki ze stosu i łączymy w przypadku gdy powstała przekątna znajduję się wewnątrz wielokąta. Sprawdzamy to za pomocą wyznacznika. Po wyczerpaniu stosu dodajemy do niego ostatnio zdjęty wierzchołek oraz rozważany wierzchołek.

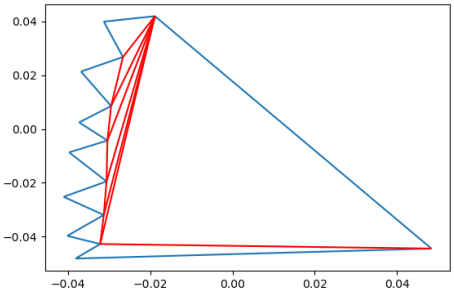
Po zakończeniu wykonywania się głównej pętli dodatkowo dodajemy krawędzie z ostatniego wierzchołka do wszystkich ze stosu oprócz pierwszego i ostatniego. Ostatecznie zwracana jest wynikowa lista przekątnych będąca wynikiem triangulacji.



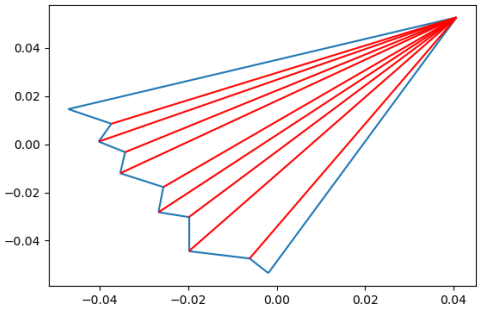
Rys. 10 Triangulacja figury 2.



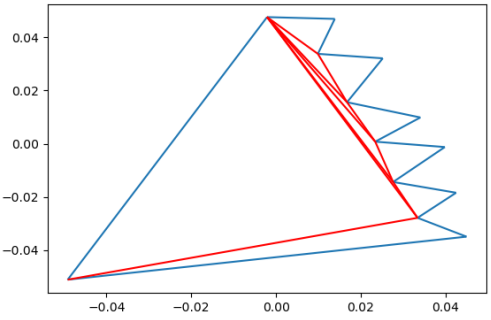
Rys. 11 Triangulacja figury 3.



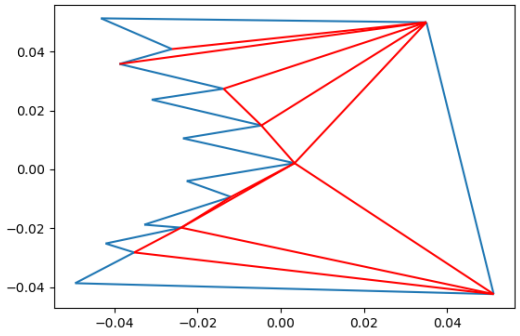
Rys. 12 Triangulacja figury 4.



Rys. 13 Triangulacja figury 5.



Rys. 14 Triangulacja figury 7.



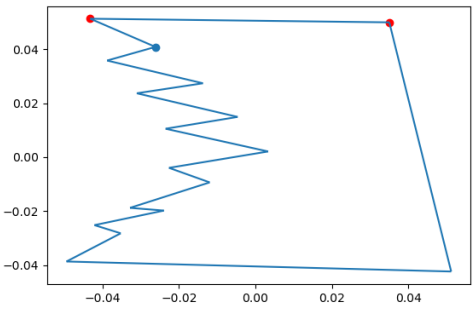
Rys. 15 Triangulacja figury 7.

7. Wizualizacja

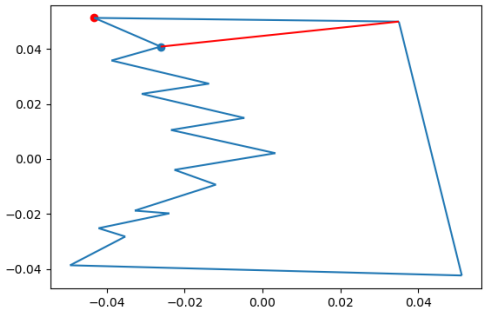
Wizualizacja została zrealizowana zapisując sceny zawierające obecnie rozważany punkt, przekątne triangulacji oraz punkty znajdujące się na stosie w danym momencie trwania algorytmu. Sceny tworzone były podczas dodawania nowych krawędzi oraz przy rozpoczęciu każdej iteracji pętli głównej.

Poniżej przedstawione wizualizacja triangulacji figury 7. z pominięciem kilku etapów ze względu na ich dużą liczbę.

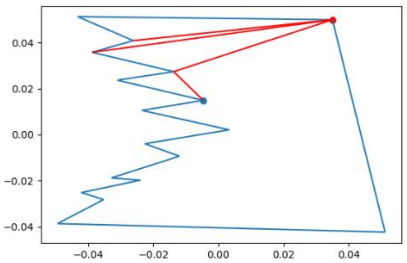
Na niebiesko zaznaczony jest obecnie rozważany punkt, a na czerwono punkty znajdujące się w danym momencie na stosie. Czerwone linie są przekątnymi triangulacji.



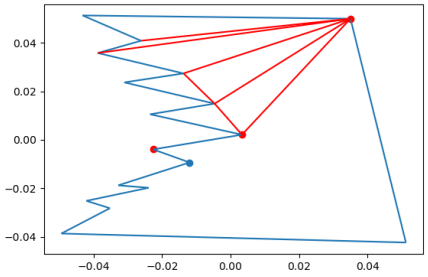
Rys. 16 Początkowy stan przed rozpoczęciem głównej pętli



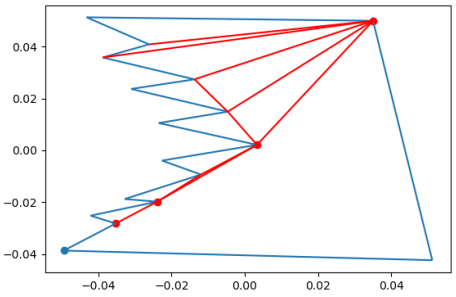
Rys. 17 Odcięcie pierwszego trójkąta



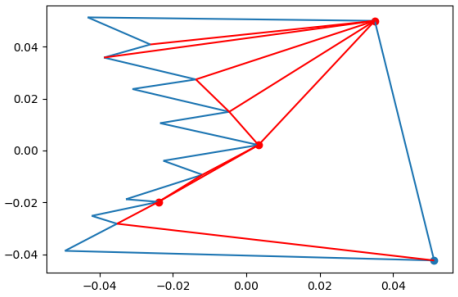
Rys. 18 Połączenie z punktem na tej samej gałęzi



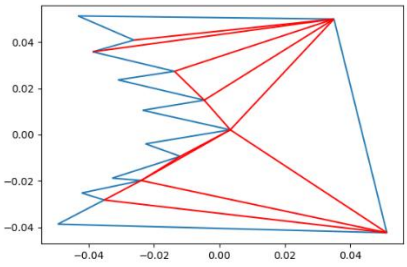
Rys. 19 Pośredni etap trwania algorytmu



Rys. 20 Zakończenie działania głównej pętli



Rys. 21 Rozpoczęcie łączenia punktów od ostatniego wierzchołka



Rys. 22 Ostateczny wynik triangulacji

8. Podsumowanie

Na podstawie przeprowadzonych obliczeń oraz wizualizacji można stwierdzić że przedstawiony algorytm triangulacji działa poprawnie. Zestawy danych zostały dobrane w taki sposób żeby sprawdzić wszystkie możliwe konfiguracje działania algorytmu. Klasyfikacja wierzchołków również zadziałała bezbłędnie. Implementacja sprawdzania y-monotoniczności działa poprawnie, jednak można by ją weryfikacją na podstawie klasyfikacji punktów. Dobrane struktury przechowywania wielokąta oraz triangulacji ułatwiły pracę ze zbiorami oraz wizualizację danych.