Dokumentacja gr-hor.ipynb

Program realizuje algorytm Greinera-Hormanna przy użyciu pakietu Jupyter Notebook, napisany został w języku Python 3.7.

W ramach programu zaimplementowany został szereg struktur danych i funkcji:

**Struktury danych:**

* Wielokąt reprezentowany jest przez klasę Polygon zawierającą:
  + Odwołanie do jednego z wierzchołków wielokąta, wierzchołki te tworzą cykliczną listę dwukierunkową: first
* Wierzchołek reprezentowany jest przez klasę Vertex zawierającą:
  + Współrzędne: x, y
  + Odwołania do wierzchołków sąsiednich: prev, next
  + Informację czy wierzchołek jest przecięciem: intersect
  + Informację czy wierzchołek jest zdegenerowanym wierzchołkiem początkowym (leży na boku wielokąta: degen
  + Dodatkowo dla przecięć potrzebujemy:
    - Informacja o rodzaju przecięcia: entry\_exit
    - Odwołanie do bliźniaczego wierzchołka w drugim wielokącie: neighbour
    - Względna odległość od poprzedniego wierzchołka w wielokącie początkowym: alpha

**Metody:**

* **Polygon:**
  + **to\_list():** zwraca wielokąt w formie listy punktów
  + **copy():** zwraca kopię wielokąta (obiekt klasy Polygon)
  + **rotate\_list():** odwraca kolejność w jakiej reprezentowany jest wielokąt
  + **add\_vertex(x, y):** dodaje wierzchołek Vertex(x,y) przed wierzchołek *first*
  + **insert\_vertex(u, v, to\_insert):** umieszcza wierzchołek to\_insert pomiędzy wierzchołki *u* i *v,* dodatkowe wierzchołki pomiędzy *u* i *v* sortuje według parametru *alpha*
  + **test\_location(vertex):** zwraca
    - OUT jeżeli *vertex* leży poza wielokątem
    - IN jeżeli *vertex* leży we wielokącie
    - ON jeżeli *vertex* leży na boku wielokąta
* **Vertex:**
  + **next\_vertex():** zwraca następny wierzchołek niebędący dodanym w trakcie trwania algorytmu Greinera-Hormanna wierzchołkiem reprezentującym przecięcie

**Funkcje:**

* ***greiner\_hormann(polygon1, polygon2, mode):***

Funkcja realizująca algorytm Greinera-Hormanna. Argumentami funkcji są dwa wielokąty reprezentowane w kolejności odwrotnej do ruchu wskazówek zegara oraz tryb w jakim chcemy uruchomić algorytm (UNION lub INTERSECTION). Zwraca krotkę (polygons\_list, phases), gdzie polygons\_list to lista wielokątów wynikowych, a phases to lista specjalnych obiektów, potrzebnych do wizualizacji algorytmu.

* ***union(polygon1, polygon2)***:

greiner\_hormann(polygon1, polygon2, UNION)

* ***intersection(polygon1, polygon2)***:

greiner\_hormann(polygon1, polygon2, INTERSECTION)

* ***show\_operation(polygon1, polygon2, operation):***

Funkcja pobiera jako argumenty dwa wielokąty oraz nazwę funkcji (union lub intersection) której wynik ma pokazać na wykresie. Zwraca obiekt klasy Plot, który pokazuje wynik działania funkcji *operation*.

* ***show\_operations(polygon1, polygon2):***

Funkcja przyjmuje jako argumenty dwa wielokąty. Zwraca obiekt klasy Plot, który reprezentuje wynik działania obu operacji na tych wielokątach

* ***animated\_operation(polygon1, polygon2, operation):***

Funkcja pobiera jako argumenty dwa wielokąty oraz nazwę funkcji dla której ma wykonać animację (union lub intersection). Zwraca obiekt klasy Plot, który pokazuje krok po kroku realizację danego algorytmu.

* ***load\_polygons(plot):***

Funkcja pobiera jako argument wykres na którym wprowadzone zostały ręcznie wielokąty. Zwraca listę tych wielokątów.

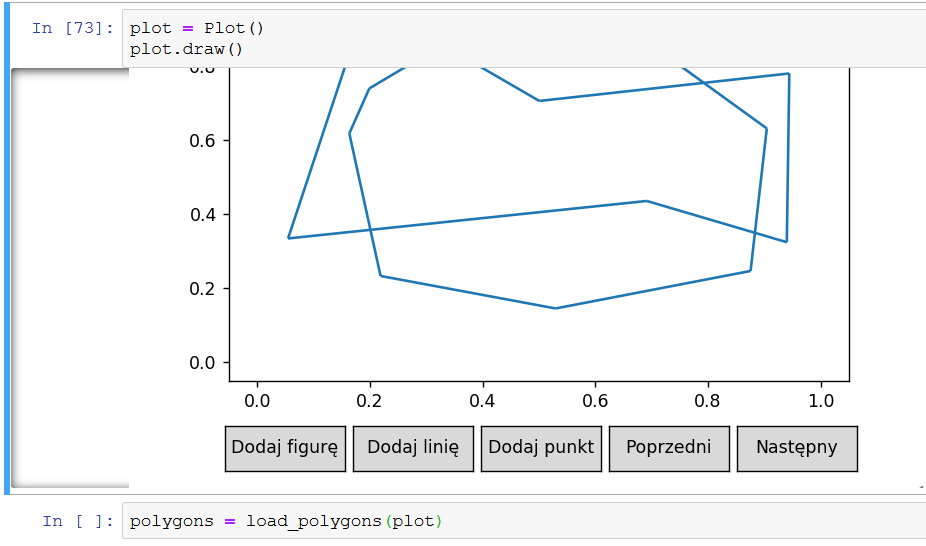
* ***polygon\_to\_json(polygons, filename):***

Funkcja pobiera jako argumenty listę wielokątów (obiektów klasy Polygon) oraz nazwę pliku json do którego zapisane zostaną wielokąty *polygons.*

* ***polygons\_from\_json(filename):***

Funkcja pobiera jako argument nazwę pliku typu json. Zwraca listę obiektów klasy Polygon, które były zapisane w pliku o nazwie *filename*.

**Przykład dodawania wielokątów:**

1. Tworzymy obiekt klasy Plot
2. Aby wyświetlić dowolny obiekt klasy Plot wywołujemy na nim metodę draw()
3. Po pojawieniu się pustego wykresu naciskamy przycisk „Dodaj figurę”
4. Dodajemy punkty nowego wielokąta w kolejności odwrotnej do ruchu wskazówek zegara, aż do połączenia ostatniego boku z wierzchołkiem początkowym
5. Aby dodać kolejny wielokąt naciskamy dwukrotnie przycisk „Dodaj figurę” i powtarzamy punkt 4.
6. W kolejnej komórce zapisujemy do zmiennej listę wielokątów przy pomocy funkcji load\_polygons(nazwa\_zmiennej)

**Funkcje union, intersection, show\_operation, show\_operations, animated\_operation zwracają obiekt klasy Plot, aby wyświetlić wynik operacji wywołujemy metodę draw() np.:**

plot = union(polygon1, polygon2)  
plot.draw()