Praca domowa 04 – convex

Michał Szczygieł

Wstęp

Zadanie składa się z 2 plików:

- Main.java
- Convex.java

CEL: Obliczyć (wyznaczyć) powierzchnię najmniejszego wielościanu wypukłego obejmującego wszystkie punkty opisane w tabeli.

Klasa Main

Klasa ta odpowiedzialna jest za wczytanie danych z bazy danych używając odpowiedniego connectora, oraz przekazanie poprawnie odebranych danych do klasy Convex.

Metody:

`public static ArrayList<Convex.Point3D> dbconnection(final String connector)` – Łączenie się z bazą danych, w celu pobrania warości.
`private static ArrayList<Convex.Point3D> retriveData(ResultSet result)`
- Odebranie danych z bazy danych I zwrócenie jej w postaci listy punktów w przestrzeni 3D.

Klasa Convex

Klasa Convex zawiera między innymi dwa klasy wewnętrzne, w celu ułatwienia dostępu do danych:

```
public class Point3D

private final class Face
```

Klasa Point3D odzwierciedla reprezentację punktu w przestrzeni 3D. Zawiera ona metody do pobierania wartości dla danych osi x,y i z w postaci „getterów”.

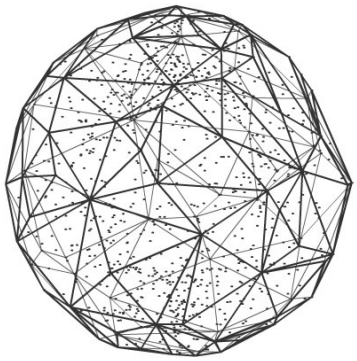
Klasa Face jest reprezentacją ściany bryły wypukłej, składającą się z 3 punktów. Klasa ta

zawiera również metody do manipulacji długościami krawędzi ściany, oraz wyliczanie powierzchni pola dla danej ściany.

Realizacja algorytmu Convex

Realizacja zadanego problemu – wyznaczania jak najmniejszej powierzchni płaszczyzny wypukłej obejmującą wszystkie punkty, odbywa się w moim przypadku metodą Convex incremental Hull. Metoda jest jedną z najłatwiejszych do implementacji, wynikającą też ze z małej złożoności problemu, jednak jest ona najwolniejszą metodą wyznaczania punktów.
W pierwszym etapie algorytm wyznacza symplex z pierwszych punktów z tabeli. W kolejnych punktach dokłada kolejne punkty dorysowując sciany i sprawdzając czy punkty są we wewnątrz Convexu, czy na zewnątrz. Złożoność obliczeniowa dla tego algorytmu:

- wstawianie elementów O(n). do listy zawierającej wszystkie punkty
- wstawianie elementu do Convexu $O(n^2)$.



Dyskusja

Dlaczego zdecydowałem się na ten algorytm, mimo że jest najwolniejszy. Gdyż wcale nie jest taki wolny, testując dla 1000 punktów, zwraca wynik w mniejszym niż 10 ms. Oczywiście przy rzędzie ilości punktów kilkuset tysięcy wydajność znacznie spada. Wtedy lepiej zastosować algorytm quickhull, którego optymistyczna wydajność wynosi O(n log h), a pesymistyczna $O(n^2+h\log n)$