Wydział Elektryczny Politechniki Warszawskiej

Języki i Metody Programowania II – projekt 1

Siatkonator

Autor:
Barnaba Turek
barnabaturek@gmail.com

13 marca 2012

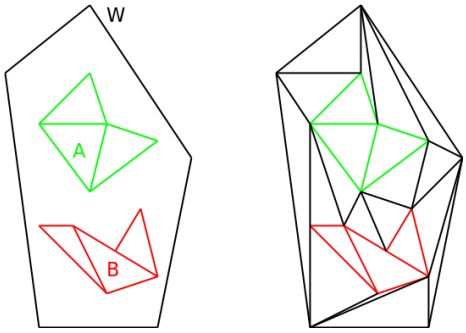
Spis treści

1 Specyfikacja funkcjonalna

1.1 Program

Siatkonator to program sklejający siatki trójkątne. Program przyjmuje jedną lub więcej siatek trójkątnych oraz jeden wielokąt.

Wynikiem działania programu jest siatka trójkątna opisująca zadany wielokąt, która zawiera podane określone siatki.



Rysunek 1: przykład sklejania zadanego wielokątu W i siatek A i B

1.2 Wywołanie

Program nie prowadzi dialogu z użytkownikiem.

Listing 1: Przykładowe wywołanie

```
$ siatkonator -e siatka1.ele -e siatka2.ele polygon.poly
```

Ostatni argument programu określa nazwę pliku (wraz ze ścieżką) w którym opisany jest wielokąt. Nie podanie tego argumentu spowoduje błąd wykonania programu.

Wcześniejsze argumenty określają opcje wykonania programu i nie są wymagane. Kolejność argumentów nie ma znaczenia.

Dostępne są następujące opcje:

`-e <name>` dodaje siatkę, która zostanie "sklejona" z wielokątem. Program zakłada, że istnieje także plik o takiej samej nazwie bazowej z rozszerzeniem `node` opisujący wierzchołki.

`-o <name>` podaje nazwę pliku wyjściowego. W przypadku braku tego parametru wynik zostanie wypisany na standardowe wejście (najpierw plik ele, potem node).

`-a <area>` określa maksymalną powierzchnię trójkąta (nie zmienia zadanych siatek!).

`-q <degrees>` określa minimalną miarę kąta w trójkącie w wygenerowanej siatce trójkątnej (nie zmienia zadanych siatek!),

W przypadku powodzenia program zwraca zero. W przeciwnym wypadku program zwraca jeden z kodów błędu:

`kod 1` Błąd otwarcia pliku (spowodowany np. brakiem uprawnień lub pliku).

`kod 2` Błąd wczytania pliku (spowodowany błędnym formatem któregoś z plików źródłowych).

`kod 3` Błędny format argumentów linii poleceń.

Ponadto program w czasie działania wypisuje informacje o swoim aktualnym stanie na wyjście `STDERR`.

1.3 Pliki

Siatkonator korzysta z plików w formatach `poly`, `ele` i `node`.

poly format pliku opisującego wielokąt

ele format pliku opisującego z których wierzchołków składają się trójkąty siatki

poly format pliku opisującego wierzchołki siatki

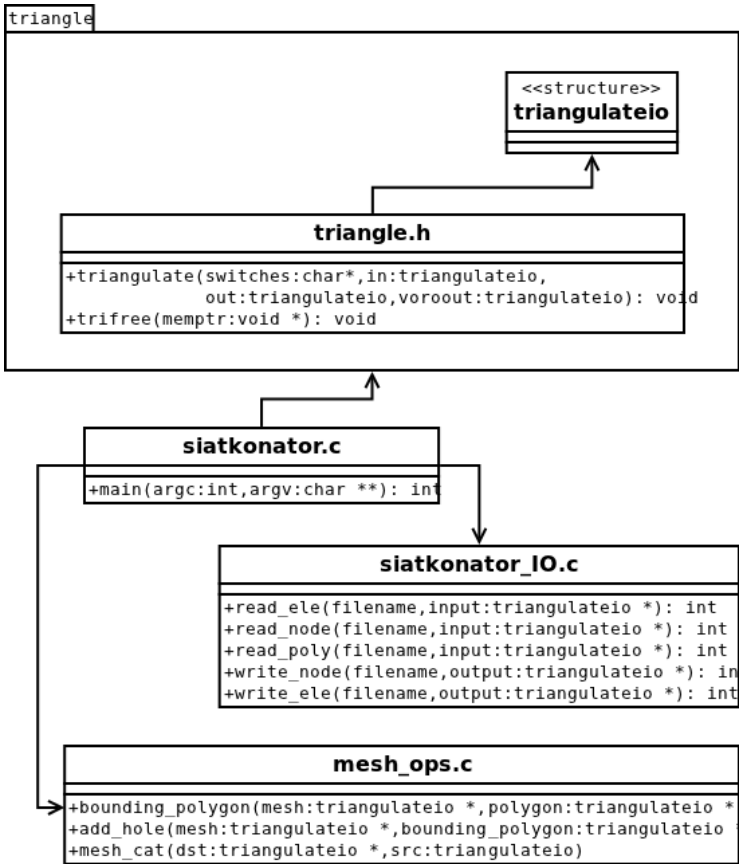
Wszystkie wykorzystywane formaty są zgodne z formatami wykorzystywanymi przez program triangle.

1.4 Dodatkowe uwagi

Odradzam użytkownikowi podawanie programowi siatek, które przecinają się ze sobą, bądź przecinają się z podanym wielokątem. Jeżeli użytkownik koniecznie chce podać bezsensowne dane, to otrzyma bezsensowne wyniki. Taka sytuacja nie jest rozumiana jako błąd programu i nie jest do niej przypisany żaden kod błędu.

2 Specyfikacja implementacyjna

2.1 Struktura programu



Rysunek 2: Diagram klas

Program podzielony jest na trzy pliki:

siatkonator.c Plik zawierający główną funkcję odpowiedzialną za wczytanie argumentów i zależnie od nich wykonanie algorytmu sklejania siatek.

siatkonator_IO.c Plik zawierający funkcje związane z czytaniem i zapisywaniem plików *node*, *ele* i *poly* do struktur `triangulateio`.

mesh_ops.c Plik zawierający funkcje obsługujące logikę sklejania siatek (znajdowanie wielokąta otaczającego siatkę, wycinanie dziur w siatkach i łączenie siatek).

common.c Plik zawierający funkcje pomocnicze (takie jak wypisywanie wiadomości).

2.1.1 Funkcje modułu `mesh_ops`

`bounding_polygon` funkcja przyjmująca wskaźnik na wczytaną gotową siatkę i znajdująca otaczający ją wielokąt (który należy wyciąć z otoczki przed przeprowadzeniem triangulacji).

`add_hole` Funkcja przyjmująca wskaźnik na wczytaną otoczkę (być może już z dziurami) i na wielokąt otaczający jedną ze sklejanych siatek. Funkcja dodaje do otoczki dziurę w miejscu, w którym zostanie w nią wklejona siatka.

`mesh_cat` Funkcja przyjmująca dwa wskaźniki na siatki i dodająca elementy drugiej siatki do elementów pierwszej siatki.

2.2 Algorytmy

Listing 2: Pseudokod algorytmu głównej funkcji programu sklejającej siatki

```
1  wczytaj dane z pliku poly do struktury p
2  dla kazdego pliku .ele:
3    wczytaj dane z plikow ele i nodes do struktury E
4    znajdz w strukturze E krawedzie, ktore wystepuja tylko raz
5    usun te krawedzie ze struktury E
6    dodaj te krawedzie do struktury P jako sekcje
7    dodaj dziure do struktury P wewnatrz kazdego trojkata ze
         struktury E
8  wykonaj triangulacje na strukturze p do siatki S
9  dla kazdego pliku .ele:
10   wstaw siatke z pliku do dziur w siatce S
11 zapisz wynik triangulacji do plikow ele i node
```

Listing 3: Pseudokod funkcji szukającej wielokąta otaczającego daną siatkę: `bounding_polygon`

```
1  H - tablica mieszajaca
2
3  Dla kazdego wierzcholka w:
4    jezeli H[w] istnieje:
5      Dodaj w do H
6    else:
7      Usun w z~H
8
9  Dla kazdego wierzcholka w:
10   jezeli H[w] istnieje:
11     dodaj w~do wielokata otaczajacego
12     usun w~z~wielokata otaczajacego
13
14 Dodaj dziure w~wielokacie otaczajacym w~miejscu srodka ciezkosci
       pierwszego elemenetu.
```

2.3 struktury danych

Struktura danych, której będę używał do komunikacji między funkcjami to struktura `triangulateio` dostarczona wraz z programem triangle. Struktura ta pozwala opisać siatkę na każdym etapie jej budowania (także siatki wyjściowe).

Jej wadą jest fakt, że nie jest specjalizowana do konkretnych zadań i wiele pól nie będzie miało zastosowania. Mimo to jest już napisana, udokumentowana i mogę być pewien, że zawiera wszystkie potrzebne pola.

Zastosowanie innej struktury, jako struktury pośredniej wymagałoby tłumaczenia jej do `triangulateio` przed wykonaniem funkcji `triangulate()`.

2.4 Rozwiązywanie konfliktów atrybutów wierzchołków

Program bierze pod uwagę tylko pierwszy atrybut każdego wierzchołka.

Wczytując pliki program zapamiętuje najniższy i najwyższy atrybut wierzchołka w danym pliku. Następnie program oblicza stałą, o którą należy przesunąć atrybuty wierzchołka w tym pliku, aby nuniknąć konfliktu.

Listing 4: Przykład

```
1  Jezeli w~pliku A.poly atrybuty wierzcholkow sa z~zakresu 1..10,
2  I~w~pliku B.node atrybuty sa z~zakresu 3..10,
3  To po wczytaniu A.poly najwiekszy znaleziony atrybut jest 10,
4  I~po wczytaniu B.node najmniejszy znaleziony atrybut bedzie 3,
5  Wiec nalezy wszystkie atrybuty wierzcholkow znalezionych w~B.node
       przesunac o~(10-3)
```

W związku z tym wynik działania programu **zależy** od kolejności podanych plików wejściowych.

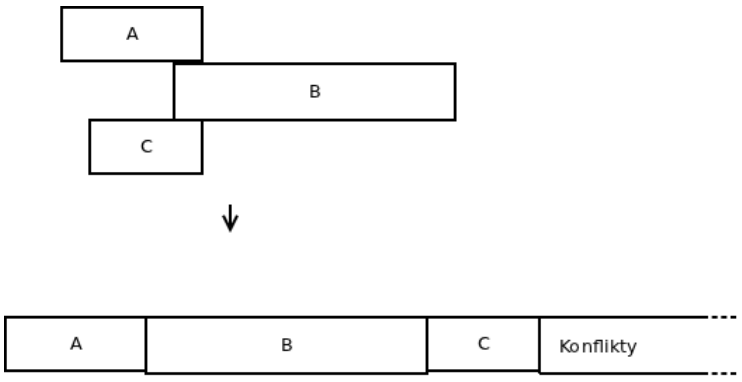
Po takim podziale pozostała przestrzeń zmiennej używana jest do rozwiązywania konfliktów wynikających z tego, że dwa wierzchołki o różnych atrybutach znalazły się w tym samym miejscu.

Jeśli oba wierzchołki mają taką samą wartość atrybutu, atrybut **nie zostaje** zachowany (wynika to z faktu rozdzielenia przestrzeni atrybutów).

Jeśli oba wierzchołki mają różną wartość atrybutów, zostaje im przydzielona nowa wartość z zakresu wartości przeznaczonego na konflikty za pomocą prostej funkcji mieszającej (znajdowana jest największa liczba pierwsza w dostępnym zakresie. Atrybut nowego wierzchołka to:

```
1 (Atrybut\ A + Atrybut\ B) % (liczba pierwsza) + offset zakresu
```

Takie rozwiązanie gwarantuje, że elementy będące połączeniem danych atrybutów zawsze będą miały ten sam atrybut w pliku wynikowym.



Rysunek 3: przykład podziału przestrzeni atrybutów