

RAFFINAMENTO

Aurona Gashi, Francesco Matteazzi, Matteo Tonti

Giovedì 22 Giugno 2023

Descrizione del problema

- **Obiettivo:** a partire da una mesh triangolare, raffinare un sottoinsieme di triangoli per ottenere una mesh più fine.

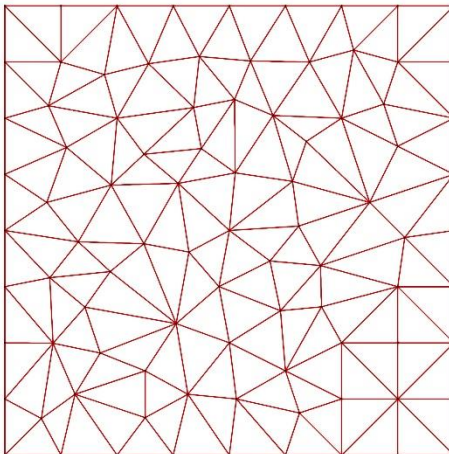
Il programma prende in input da tre file CSV la mesh iniziale, ovvero:

- I **vertici** dei triangoli (id, coordinata x, coordinata y);
- I **lati** dei triangoli (id, id vertice di origine, id vertice di fine);
- I **triangoli** (id, id dei vertici, id dei lati);

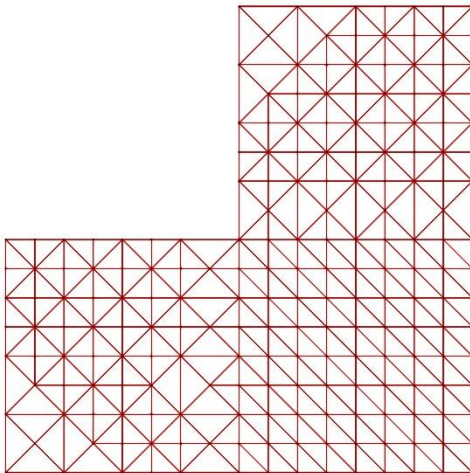
Restituisce in output, in tre diversi file CSV, la mesh raffinata, ovvero:

- I **vertici** dei nuovi triangoli (id, coordinata x, coordinata y);
- I **lati** dei nuovi triangoli (id, id vertice di origine, id vertice di fine);
- I **nuovi triangoli** (id, id dei vertici);

Mesh iniziale del Test1:

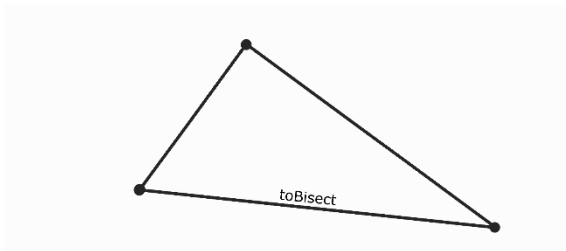


Mesh iniziale del Test2:



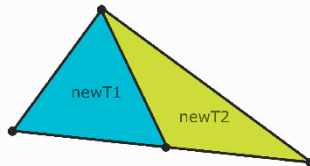
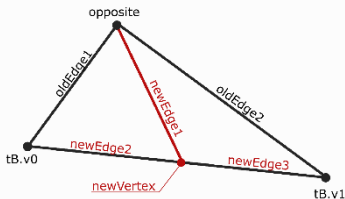
Come abbiamo ragionato:

- Ordiniamo i triangoli per area decrescente in un vettore sortedTriangles;
- selezioniamo il triangolo di area maggiore, che verrà raffinato;
- andiamo a prendere il lato più lungo (toBisect) del triangolo:



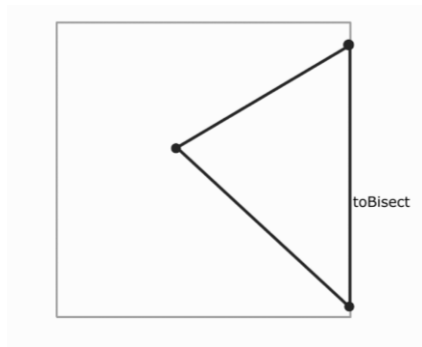
Soluzione

- Troviamo il punto medio di toBisect;
- bisezioniamo toBisect;
- creiamo i due nuovi triangoli newT1 e newT2, li inseriamo nelle liste di triangoli e spegniamo il triangolo appena raffinato:

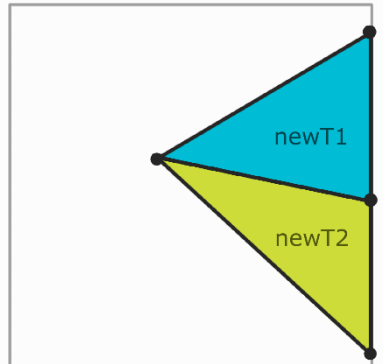
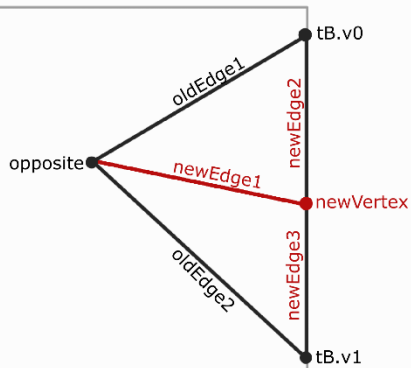


Soluzione

- Se il lato più lungo (toBisect) del triangolo da bisezionare si trova al bordo, non ci sono triangoli adiacenti rispetto a toBisect e quindi la mesh risulta ammissibile;
- passiamo quindi a raffinare il prossimo triangolo con area maggiore:



Soluzione

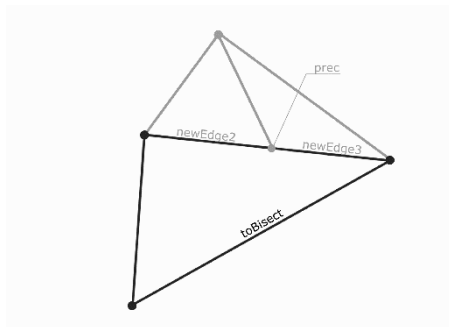


- Altrimenti, per mantenere la mesh ammissibile, dobbiamo raffinare anche il triangolo adiacente rispetto al lato bisezionato del triangolo appena raffinato;
- **Ci sono due casi:**
 1. Il triangolo adiacente ha come lato più lungo un lato diverso da quello precedentemente bisezionato;
 2. Il triangolo adiacente ha come lato più lungo il lato precedentemente bisezionato;

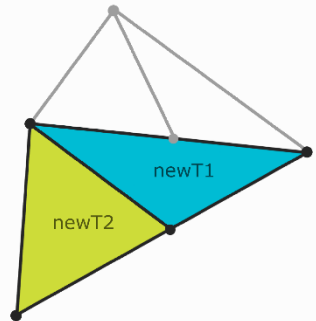
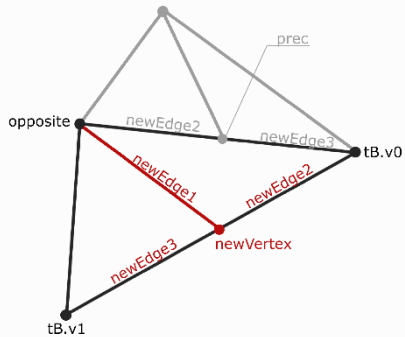
Soluzione

Caso 1:

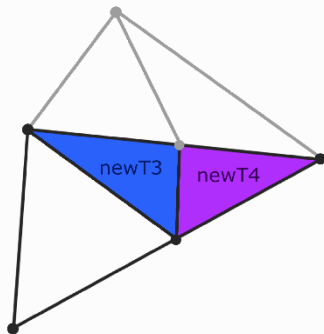
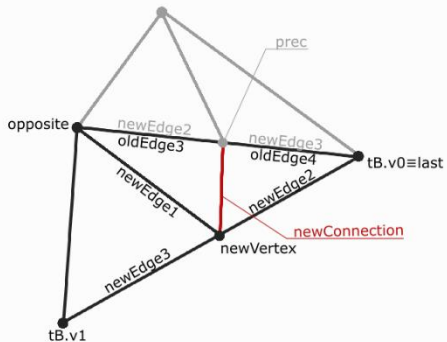
- Bisezioniamo il lato più lungo del lato adiacente;
- creiamo i due nuovi triangoli newT1 e new T2 e li inseriamo nelle liste di triangoli;
- uniamo il punto medio del lato bisezionato precedentemente al vertice opposto (di newT1 o newT2, che verrà "spento") ;
- creiamo i nuovi triangoli newT3 e newT4 e li inseriamo nelle liste di triangoli:



Soluzione

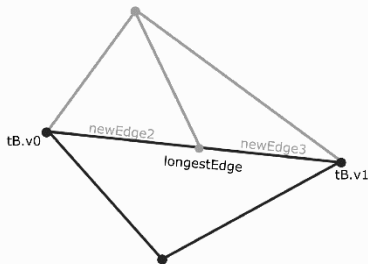


Soluzione

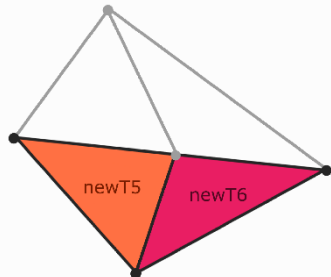
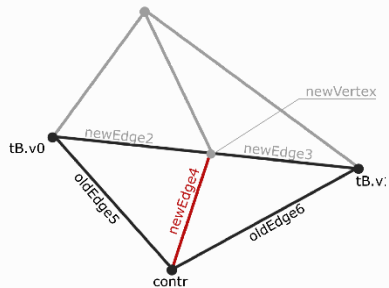


Caso 2:

- Uniamo con un nuovo lato il punto medio del lato precedentemente bisezionato con quello del vertice opposto del triangolo adiacente;
- creiamo i due nuovi triangoli newT5 e newT6, li inseriamo nelle liste di triangoli e spegniamo il triangolo raffinato:



Soluzione



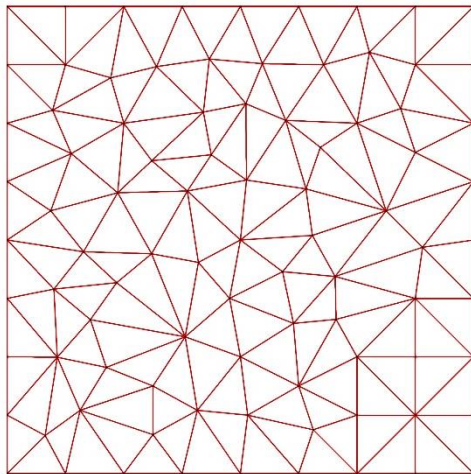
Condizioni di terminazione:

- Il lato da bisezionare si trova al bordo;
oppure
- Il triangolo adiacente ha come lato più lungo il lato precedentemente bisezionato (Caso 2);

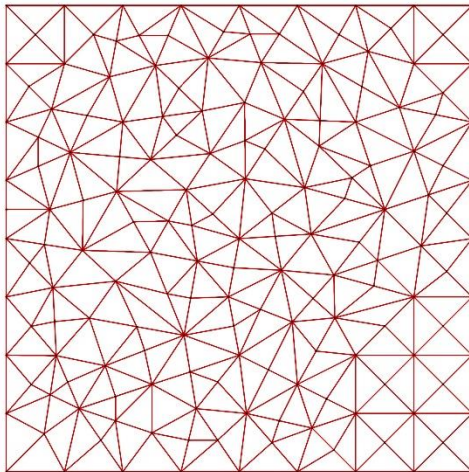
In entrambi i casi, la mesh risulterà ammissibile e quindi procederemo a raffinare il triangolo con area maggiore tra i rimanenti ("attivi") , prendendolo dalla lista riordinata di triangoli.

- Il numero di iterazioni totale è dato da : $\text{percentuale}^* (\text{numero iniziale di triangoli})/100$; (la percentuale è specificata dall'utente nella command line).

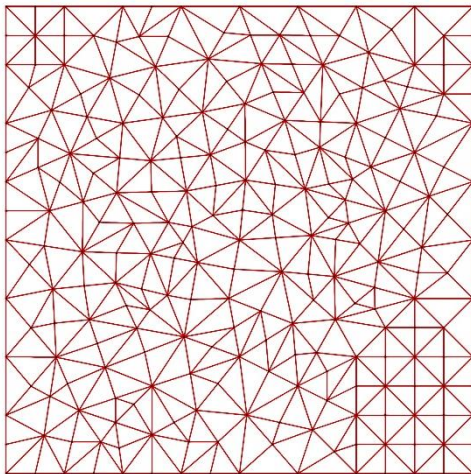
Mesh iniziale:



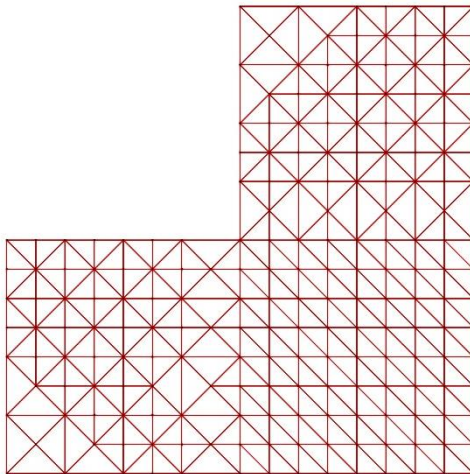
Mesh raffinata con percentuale = 50%:



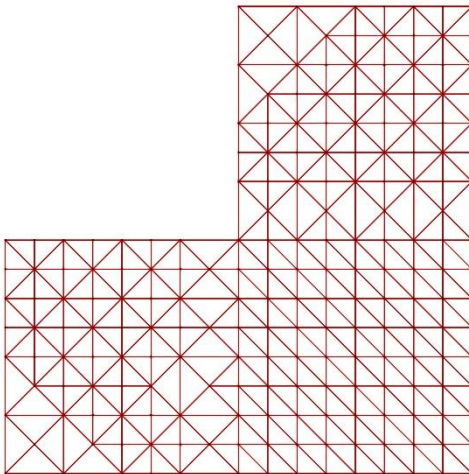
Mesh raffinata con percentuale = 100%:



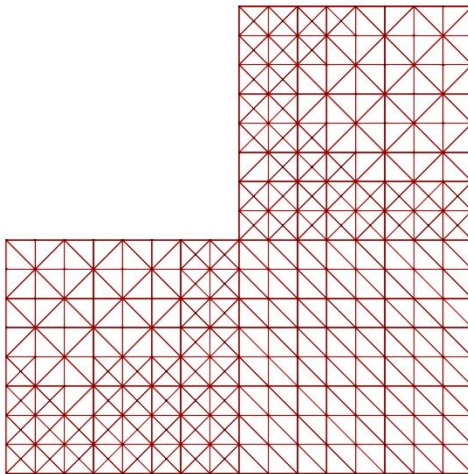
Mesh iniziale:



Mesh raffinata con percentuale = 50%:



Mesh raffinata con percentuale = 100%:



Unit tests realizzati

1. Test sui costruttori delle classi:

- classe Vertex (vertice);
- classe Edge (lato);
- classe Triangle (triangolo).

2. Test sugli algoritmi di sorting:

- Heapsort;
- Insertion sort;

3. Test sull' import:

- vertici;
- lati;
- triangoli;

4. Test sull' algoritmo di raffinamento:

- funzione Bisect.

Builder

- Varie implementazioni nello stesso oggetto;
- separazione della logica;
- facile da implementare.

Diagramma di classe UML

