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:

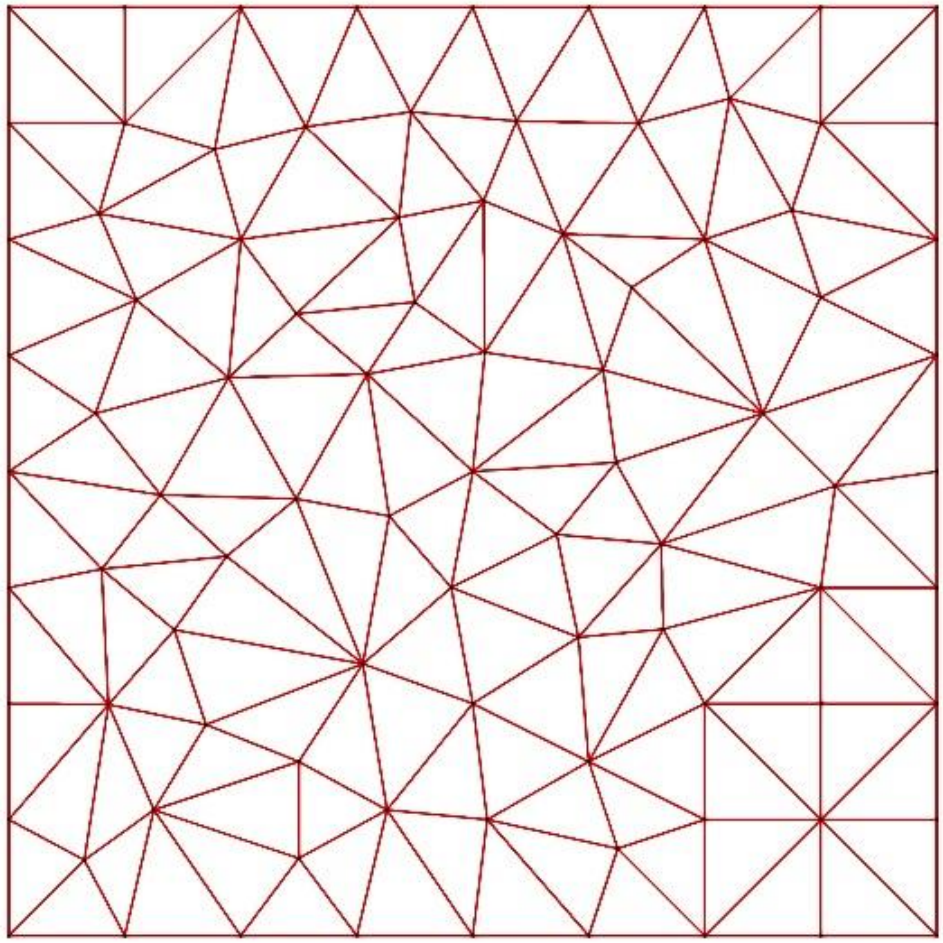
- I **vertici** dei triangoli (id, coordinata x, coordinata y);
- I **lati** dei triangoli (id, id vertice di origine, id vertice di fine);
- I **triangoli** (id, id dei vertici, id dei lati);

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);

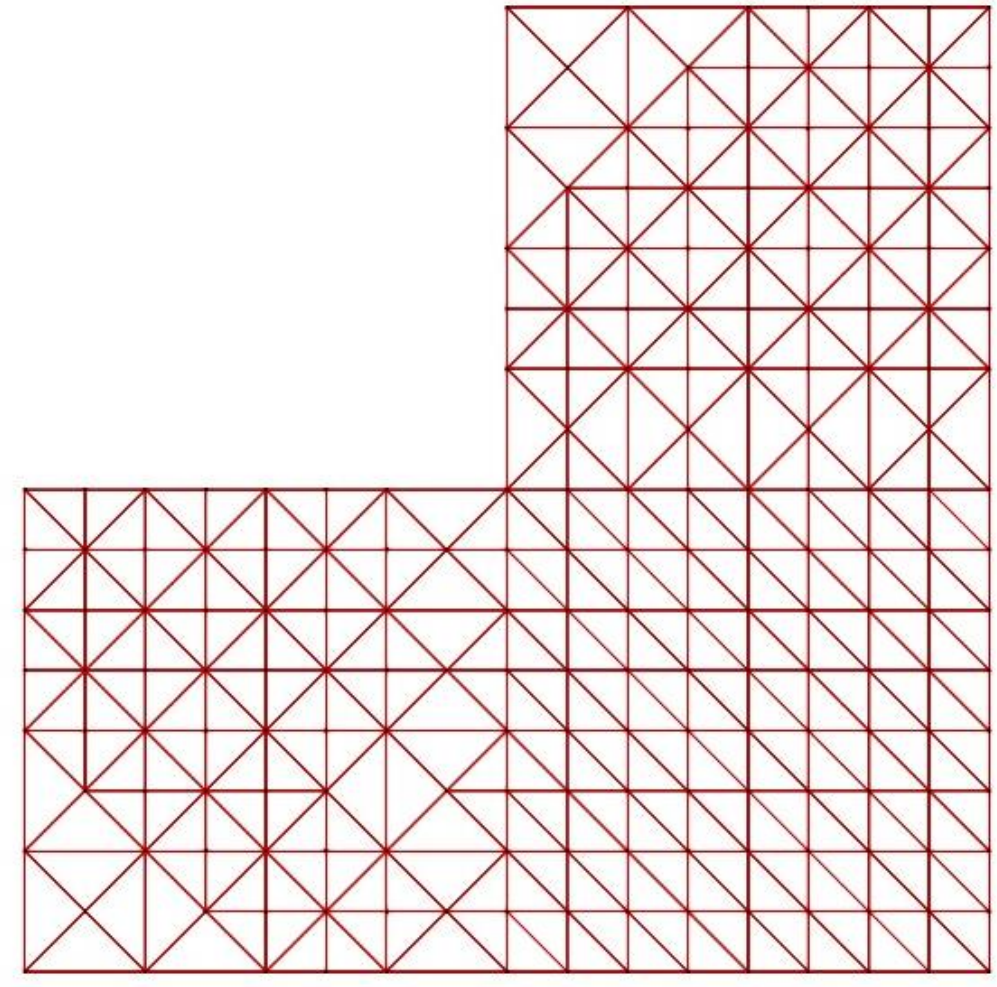
Descrizione del problema

Mesh iniziale del Test1:



Descrizione del problema

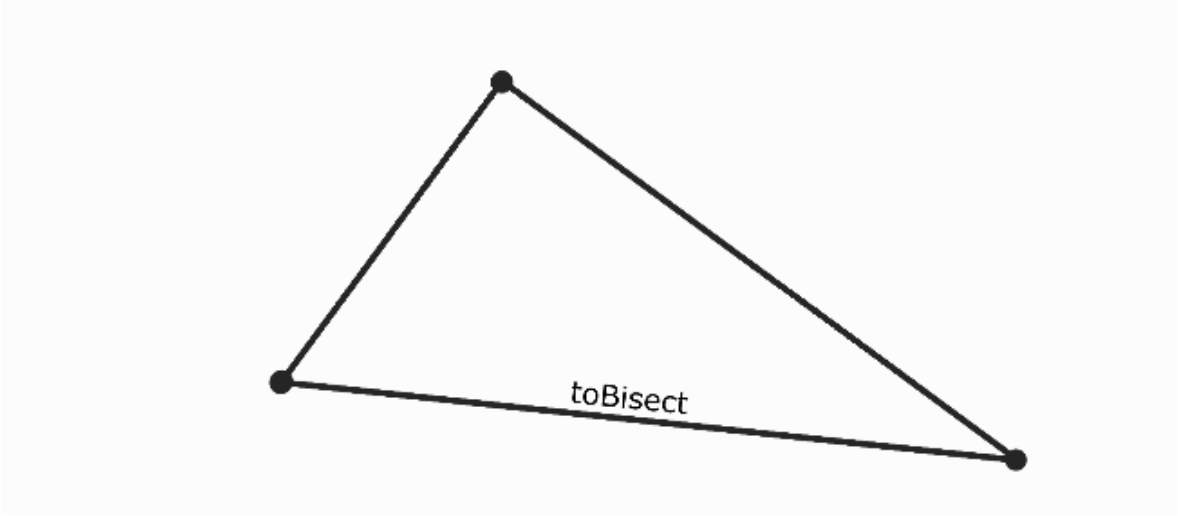
Mesh iniziale del Test2:



Soluzione

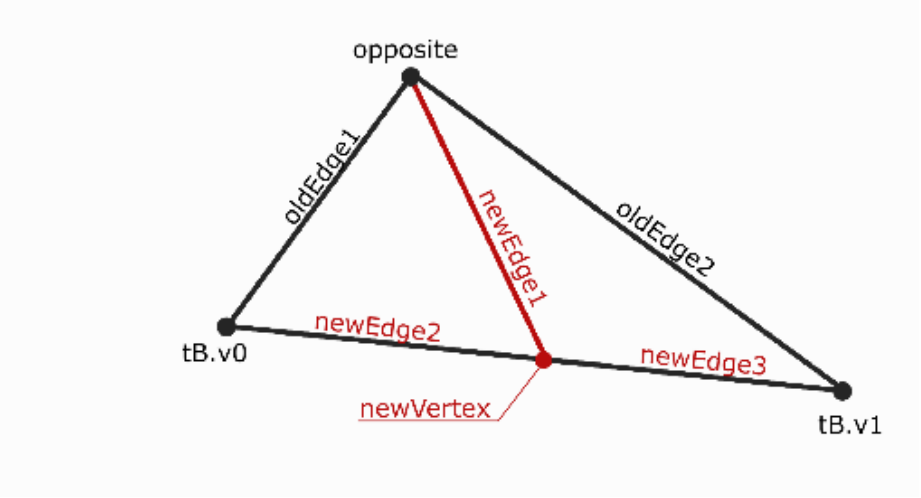
Come abbiamo ragionato:

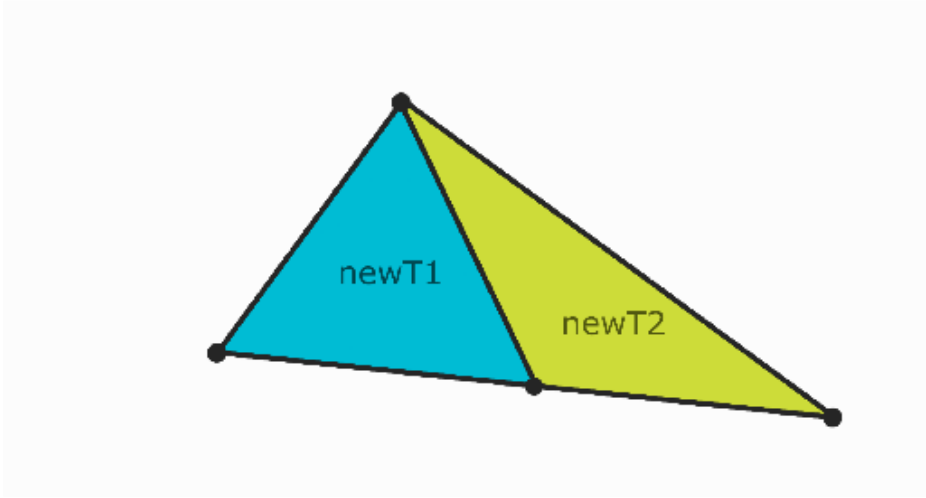
- Ordiniamo i triangoli per area decrescente in un vettore sortedTriangles;
- selezioniamo il triangolo di area maggiore, che verrà raffinato;
- andiamo a prendere il lato più lungo (toBisect) del triangolo:



Soluzione

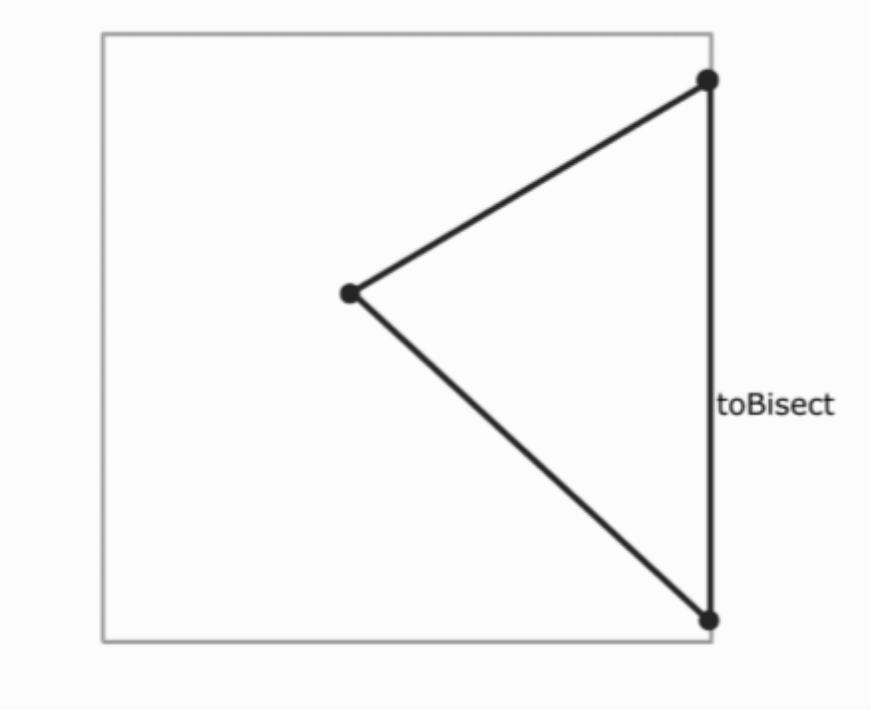
- Troviamo il punto medio di toBisect;
- bisezioniamo toBisect;
- creiamo i due nuovi triangoli newT1 e newT2, li inseriamo nelle liste di triangoli e spegniamo il triangolo appena raffinato:



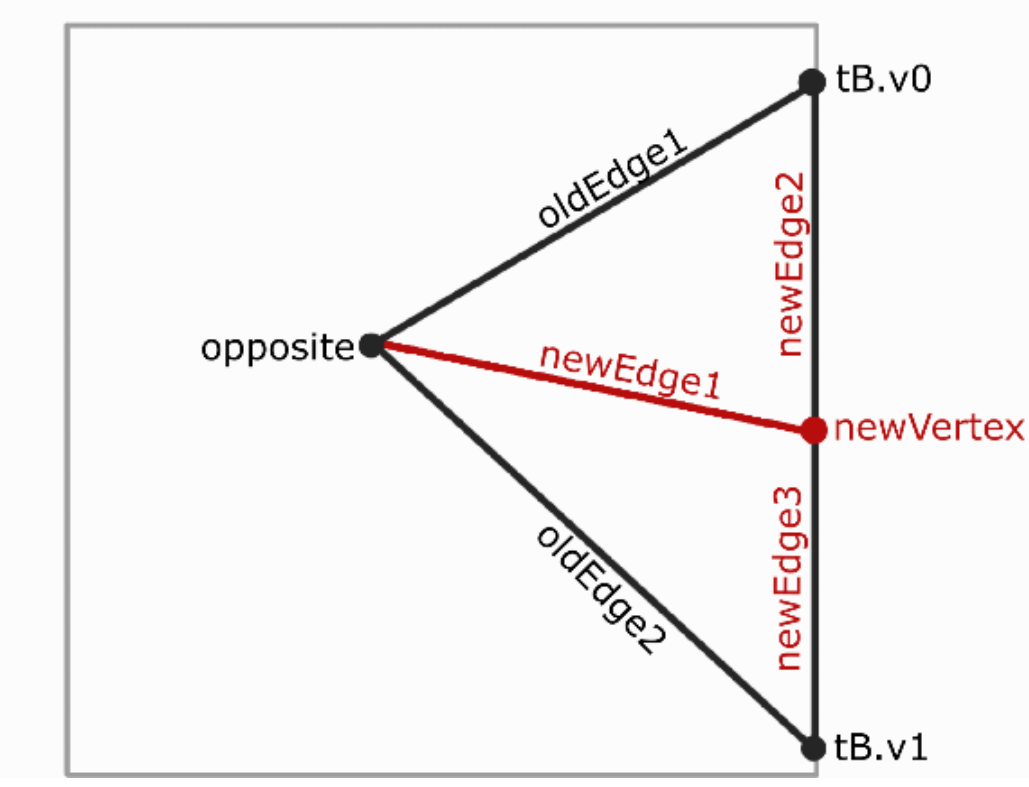


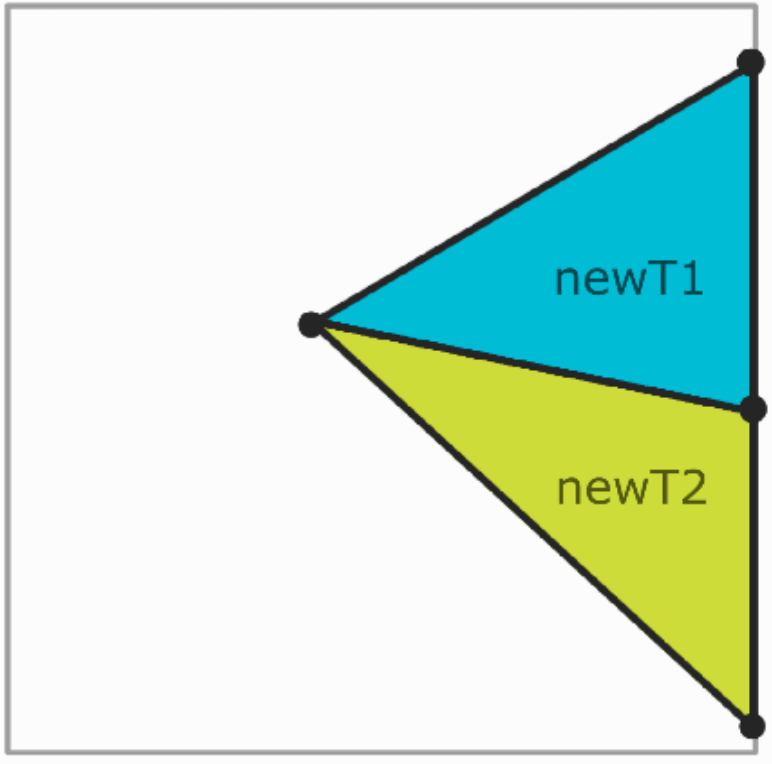
Soluzione

- 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


tB.v0
oldEdge1
newEdge2
opposite
newEdge1
newVertex
oldEdge2
newEdge3
tB.v1


newT1
newT2

Soluzione

- Altrimenti, per mantenere la mesh ammissibile, dobbiamo raffinare anche il triangolo adiacente rispetto al lato bisezionato del triangolo appena raffinato;

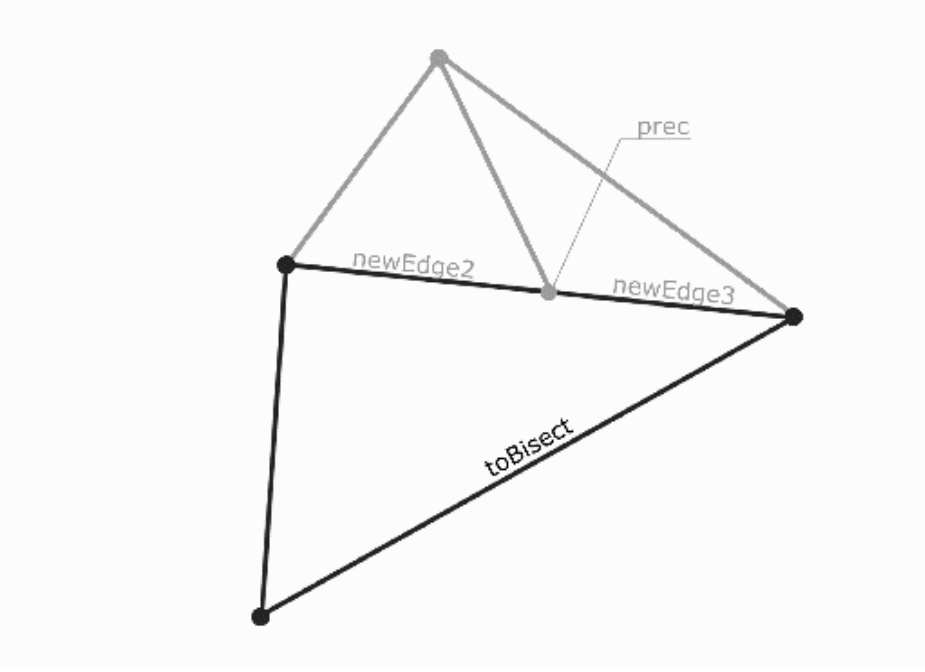
- **Ci sono due casi:**

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;

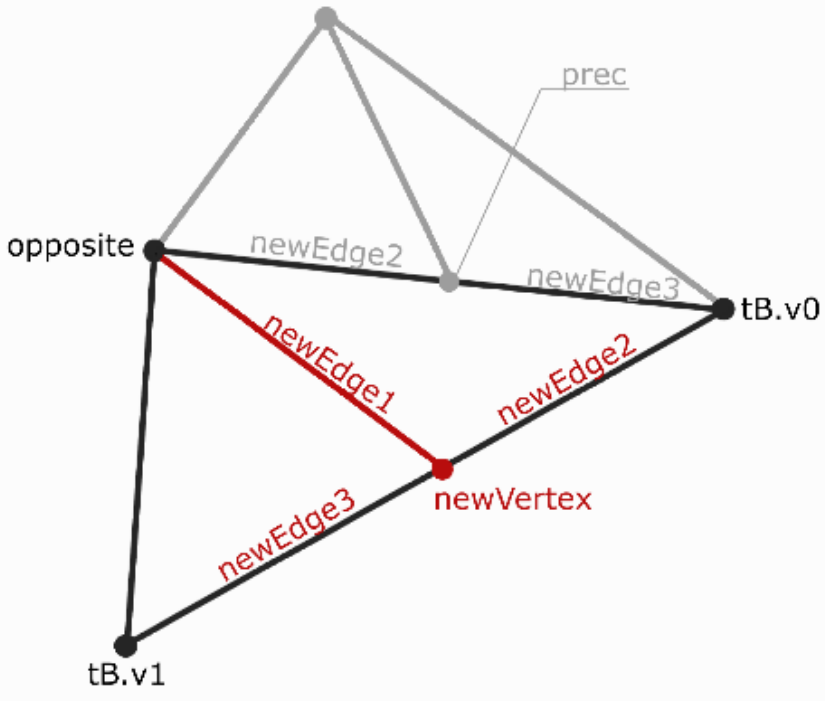
Soluzione

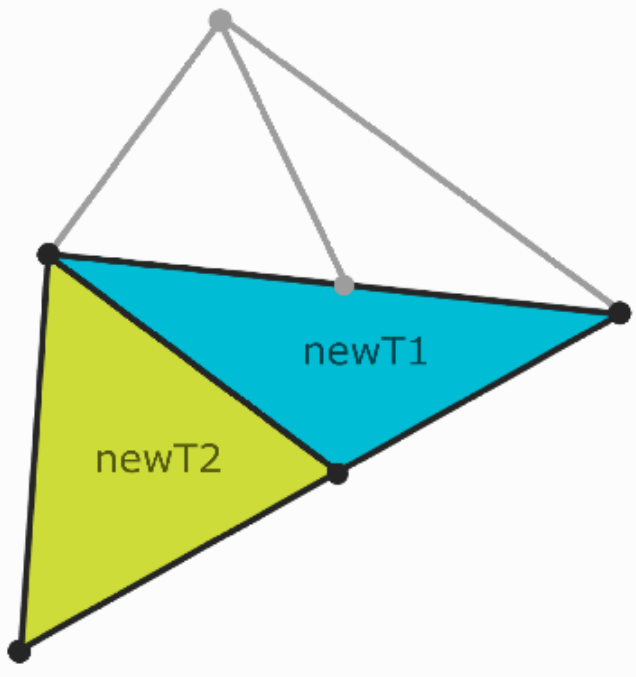
Caso 1:

- 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:

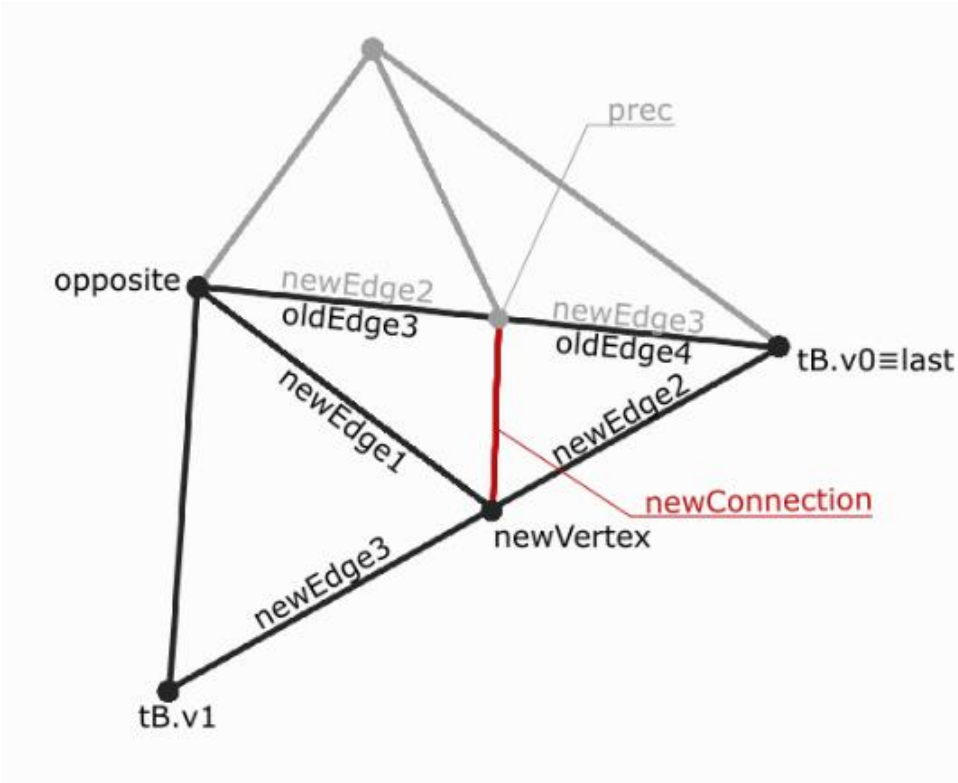


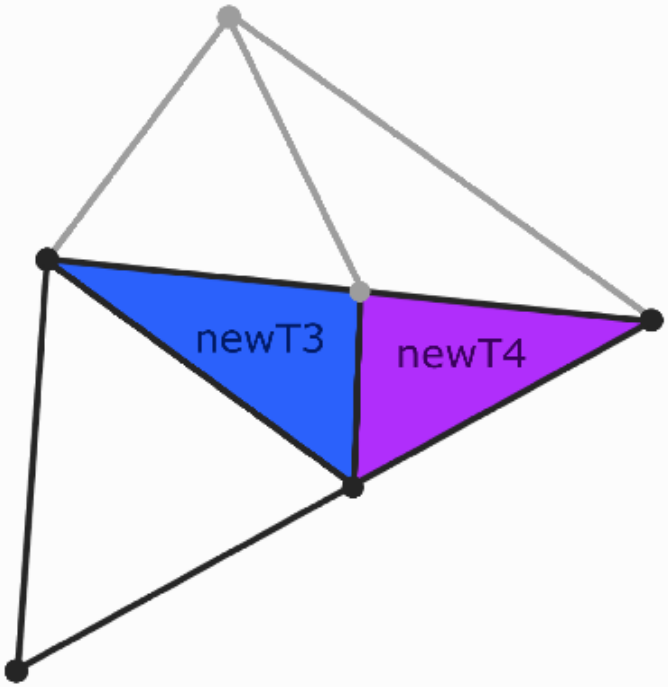
Soluzione


prec
opposite
newEdge2
newEdge3
tB.v0
newEdge1
newEdge2
newEdge3
newVertex
tB.v1


newT1
newT2

Soluzione

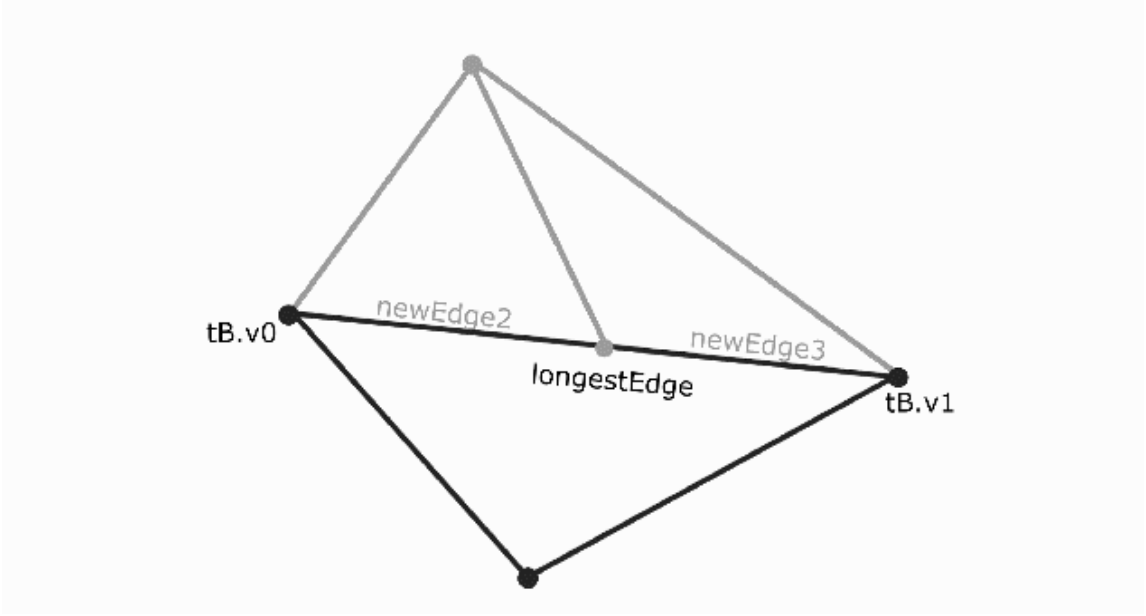

prec
opposite
newEdge2
oldEdge3
newEdge3
oldEdge4
tB.v0≡last
newEdge1
newEdge2
newConnection
newVertex
newEdge3
tB.v1


newT3
newT4

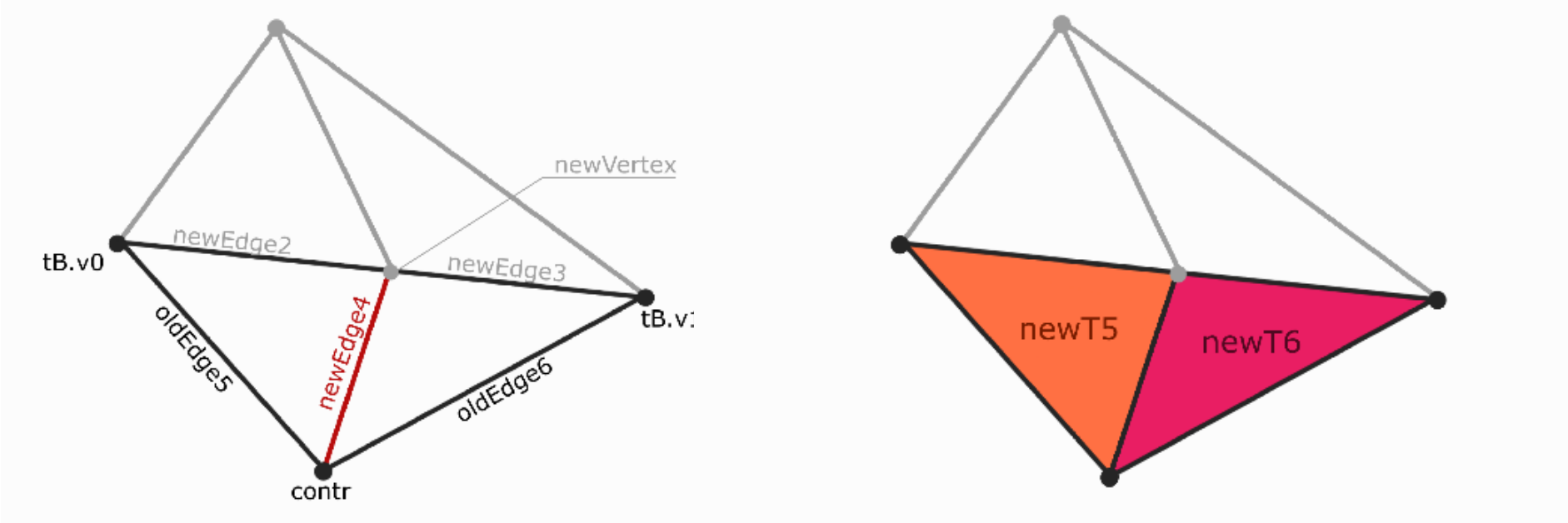
Soluzione

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


newVertex
newEdge2
newEdge3
tB.v0
tB.v1
oldEdge5
newEdge4
oldEdge6
contr
newT5
newT6

Soluzione

Condizioni di terminazione:

- Il lato da bisezionare si trova al bordo;

oppure

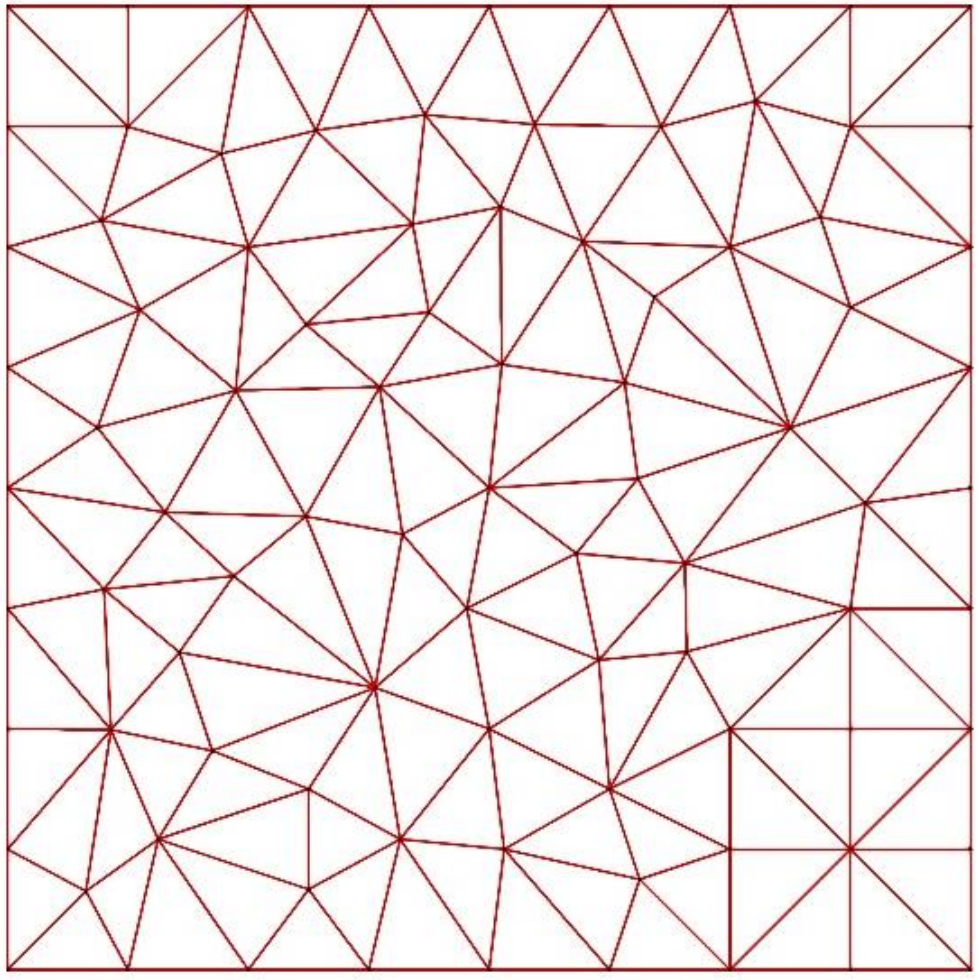
- Il triangolo adiacente ha come lato più lungo il lato precedentemente bisezionato (Caso 2);

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 : percentuale* (numero iniziale di triangoli)/100; (la percentuale è specificata dall'utente nella command line).

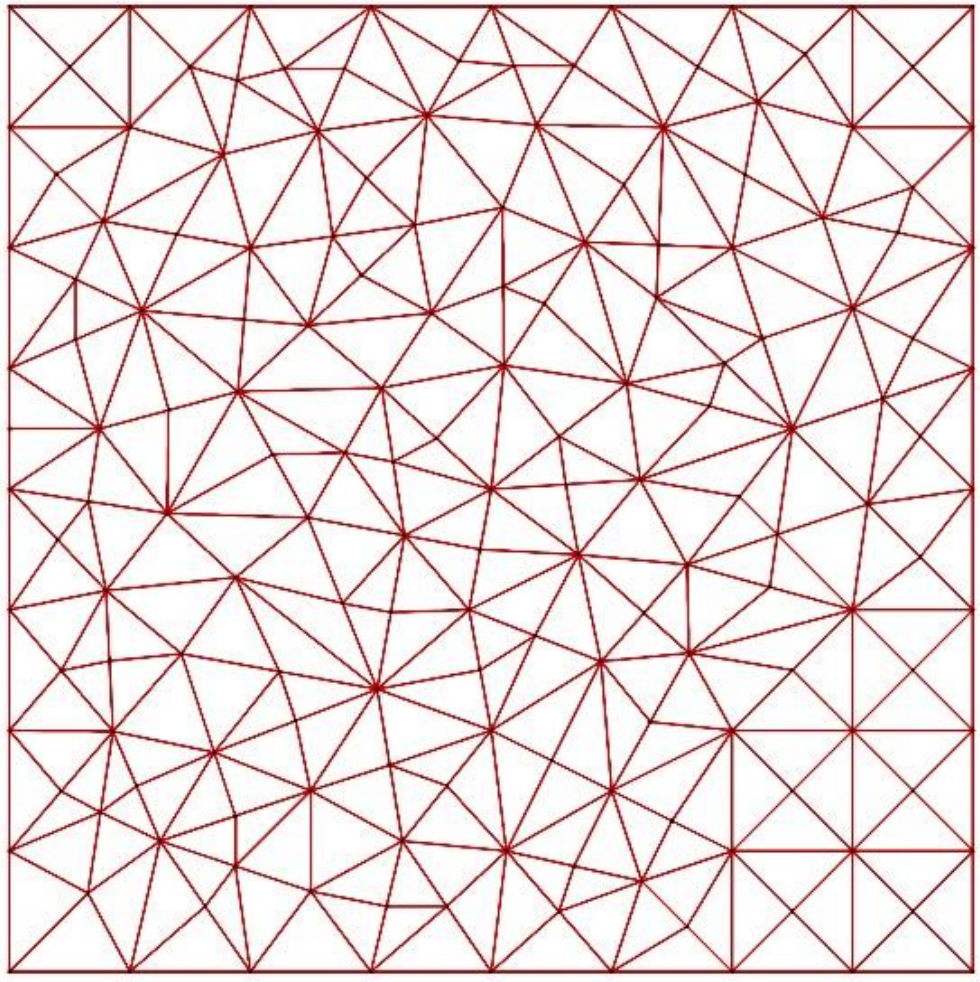
Risultati sulla mesh del Test1

Mesh iniziale:



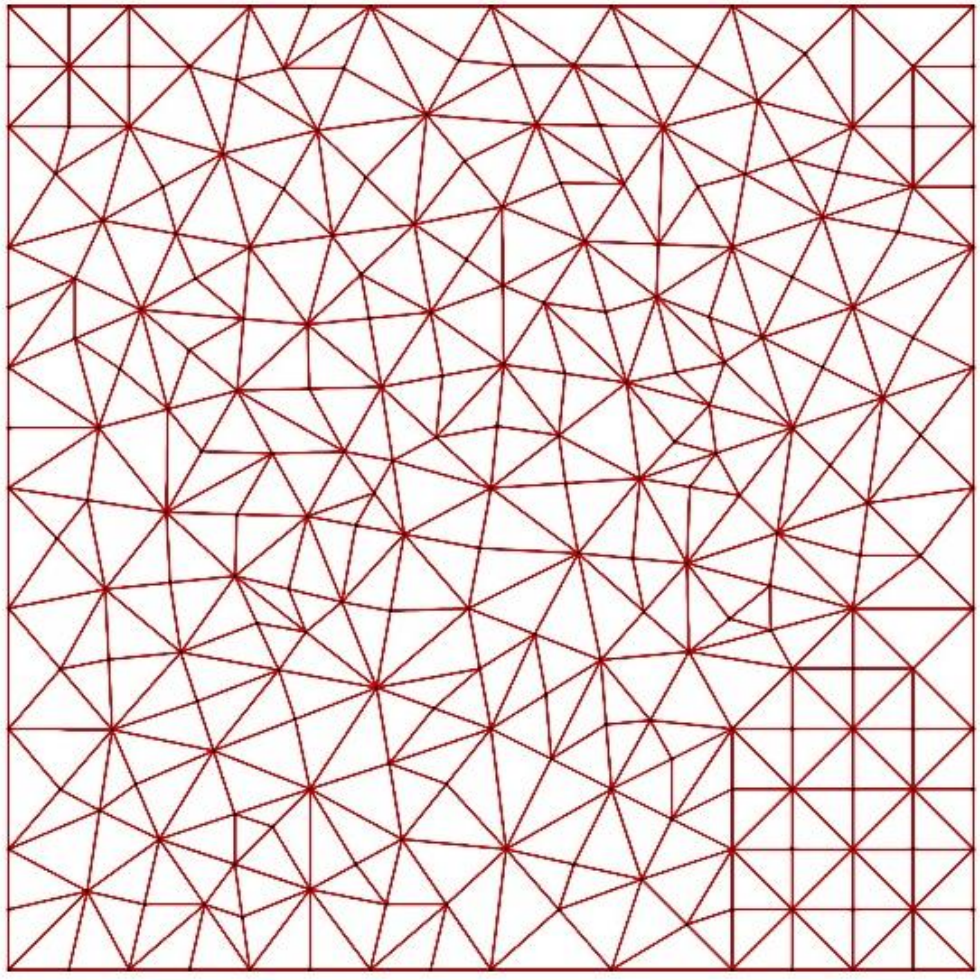
Risultati sulla mesh del Test1

Mesh raffinata con percentuale = 50%:



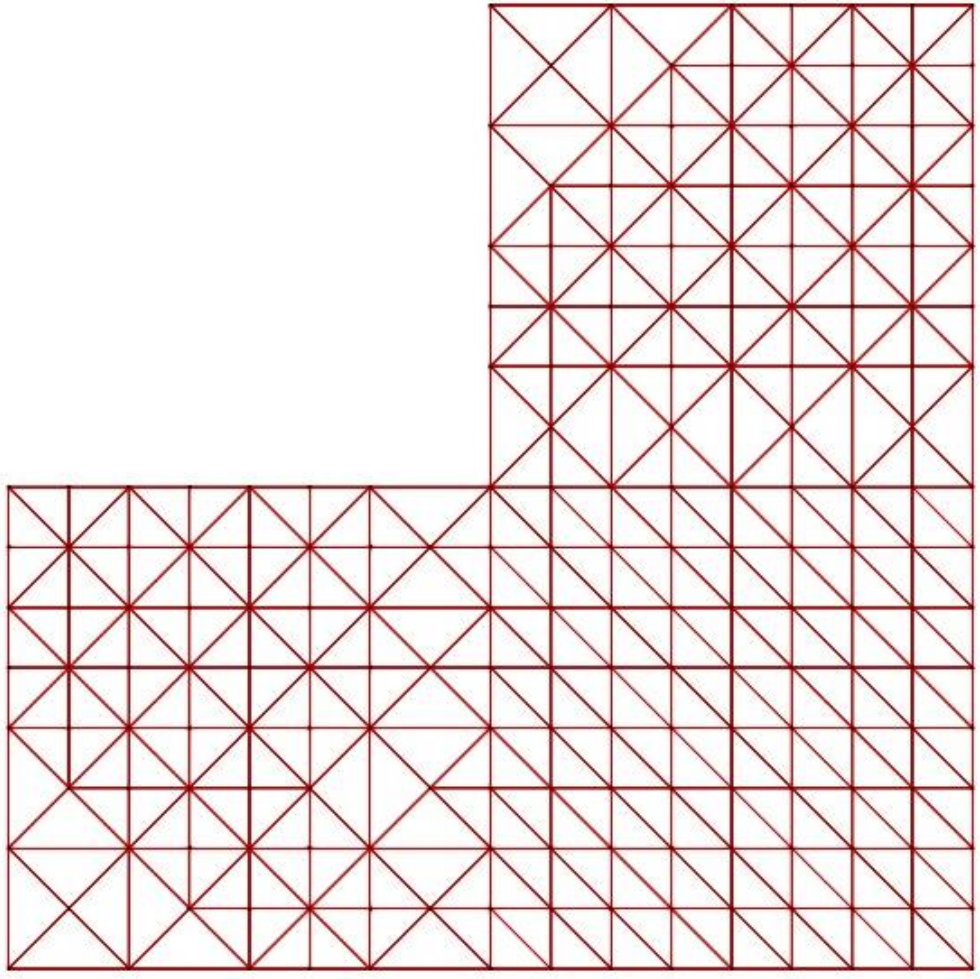
Risultati sulla mesh del Test1

Mesh raffinata con percentuale = 100%:



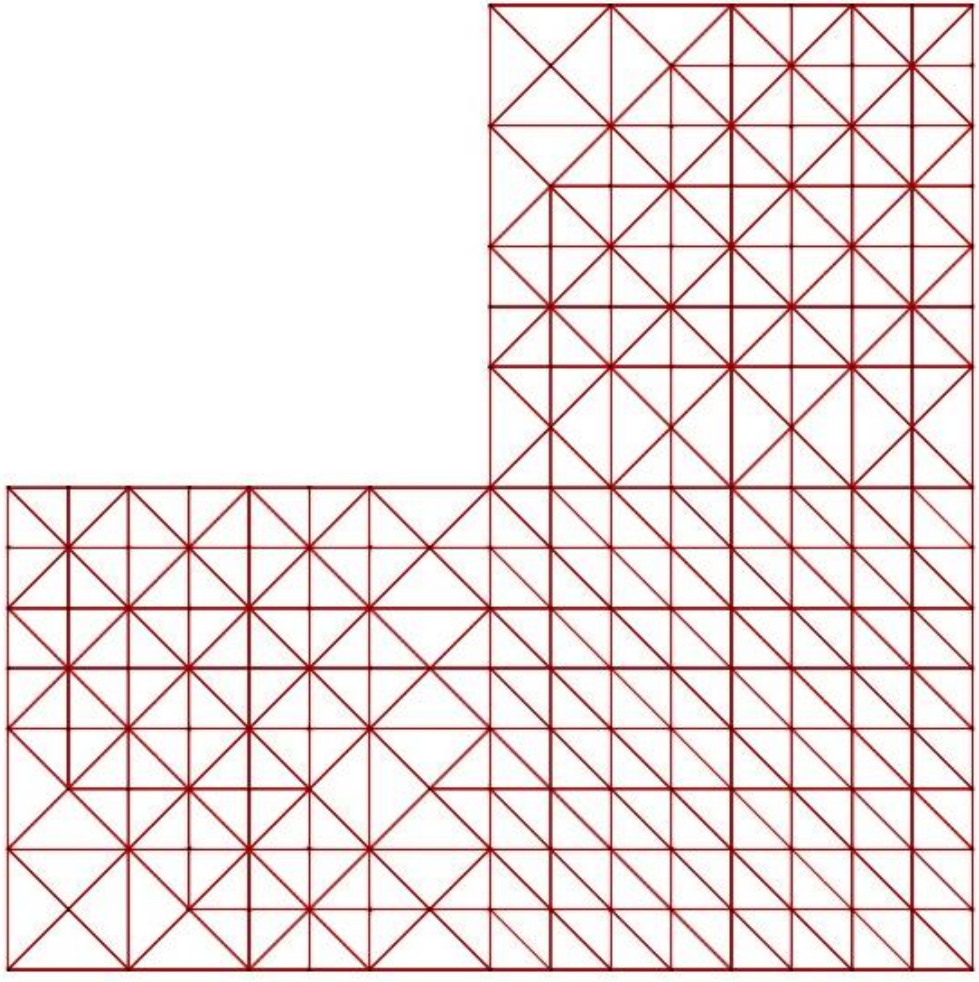
Risultati sulla mesh del Test2

Mesh iniziale:



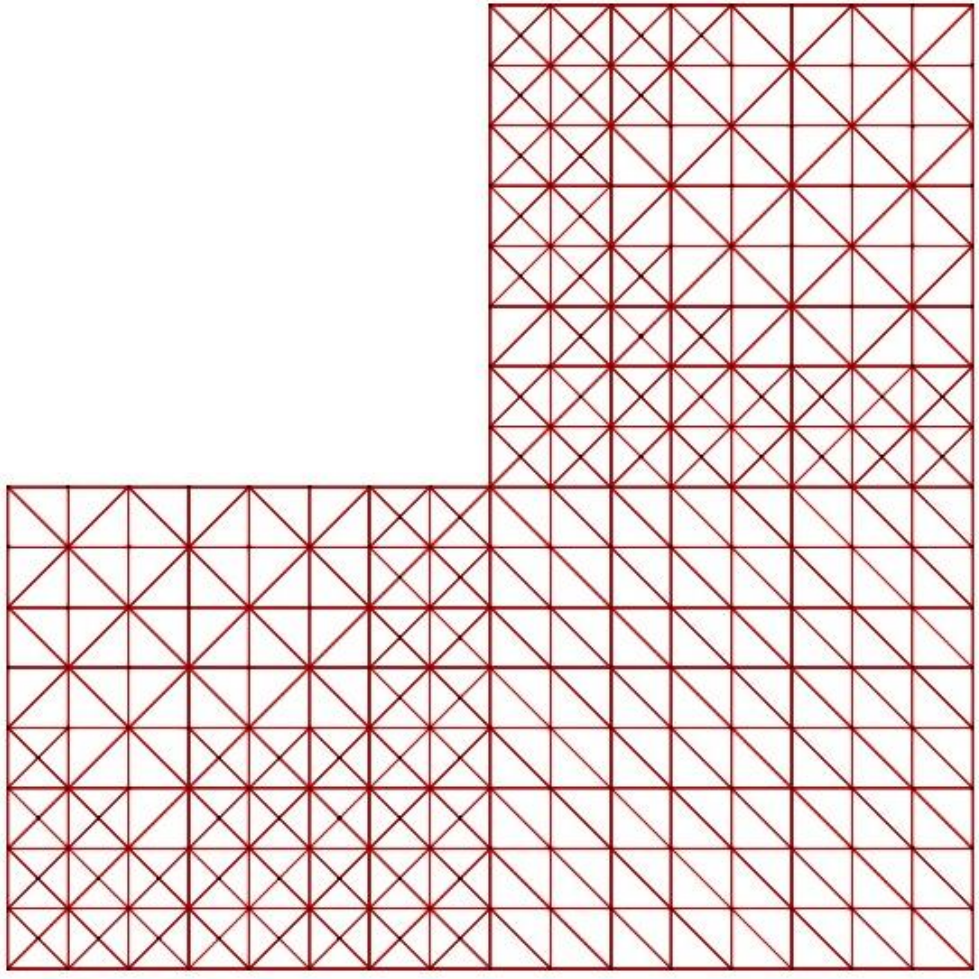
Risultati sulla mesh del Test2

Mesh raffinata con percentuale = 50%:



Risultati sulla mesh del Test2

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' algortimo di raffinamento:**
- funzione toBisect.

Builder

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

Diagramma di classe UML