Progetto Algoritmi Geometrici A.A. 2003-2004

Versione 8 gennaio 2004

1 Scopo dell'esercitazione

Implementare una triangolazione piana con le operazioni di interrogazione (estrazione delle relazioni topologiche) e di modifica, utilizzando la *struttura dati indicizzata con adiacenze* vista a lezione.

Il codice dovra' essere scritto in C o C++ e progettato come una *libreria* che possa essere utilizzata da altri programmi. Per questo sono date *precise specifiche* riguardo alla sintassi delle funzioni da rendere disponibili.

Un primo esempio di programma che utilizza la libreria e' il programma grafico che vi forniamo per poter visualizzare e controllare se l'implementazione funziona correttamente.

2 Specifiche

La struttura dati indicizzata con adiacenze e' stata spiegata a lezione.

Internamente, la vostra implementazione puo' fare uso di tipi definiti a vostro piacere (array, puntatori, strutture, classi...). Esternamente pero'deve permettere di accedere alla triangolazione riferendosi ai vertici e ai triangoli semplicemente mediante *indici* costituiti da numeri interi.

Consigli per la memorizzazione

Per avere "gratis" l'accesso ai vertici / triangoli tramite indici, consigliamo di tenere sia i vertici che i triangoli in array. Gli array per comodita' possono essere statici, dimensionati ad un numero di posizioni prefissato e ragionevolmente grande (es. dell'ordine del centinaio, e ricordate che i triangoli sono circa il doppio dei vertici).
Poiche' il numero di vertici e triangoli puo' aumentare / diminuire (con le operazioni di raffinamento / semplificazione) bisogna prevedere un meccanismo di riuso e ricompattamento delle posizioni dell'array:

- Ogni volta che si cancellano triangoli / vertici possibilmente si riusano le loro posizioni per nuovi triangoli / vertici. Le posizioni che restano libere in mezzo all'array si riempiono trasferendovi triangoli / vertici che si trovavano in fondo.

- Ogni volta che si creano triangoli / vertici possibilmente si riusano le posizioni lasciate libere da triangoli / vertici cancellati. Le posizioni che servono in piu' si occupano in fondo all'array.

- Notare che dopo un'operazione un triangolo / vertice puo' cambiare indice, e viceversa lo stesso indice puo' denotare un triangolo / vertice diverso
- Per evitare di dover cambiare tutti i riferimenti incrociati delle relazioni (TV, TT, VT parziale) si consiglia di implementare tali riferimenti incrociati con puntatori (non con gli indici interi dell'array). Gli indici dell'array serviranno solo per accedere ai triangoli / vertici dall'esterno.

Poiche' la struttura dati vista a lezione memorizza triangoli, e' necessario tenere traccia separatamente della *faccia esterna o faccia infinita* (ovvero del contorno della triangolazione) che in generale non e' un triangolo.
Per questo si puo' tenere per esempio una lista di vertici. Anche qui si consiglia usare puntatori ai vertici, non i loro indici nell'array.

Funzioni da esportare

Devono essere fornite le seguenti funzioni (con la sintassi specificata).
Cio' non impedisce di implementare *anche altre* funzioni, incluse funzioni che assolvano al medesimo compito di queste, utilizzando i tipi da voi definiti.

Funzioni generali

1. `int VertexNum()` ritorna il numero totale di vertici
2. `int TriangleNum()` ritorna il numero totale di triangoli

Nel seguito della specifica si assume che:

- gli indici interi inizianti per lettera `v` rappresentano vertici e devono essere compresi tra `0` e `VertexNum()-1`
- gli indici interi inizianti per lettera `t` rappresentano triangoli e devono essere compresi tra `0` e `TriangleNum()-1`
- uno spigolo `e` in argomento a una funzione viene specificato come la coppia di vertici v1,v2 suoi estremi; in questo caso, la coppia v1,v2 deve definire uno spigolo esistente

Alle funzioni e' richiesto comportarsi come specificato *solo* se sono soddisfatte queste condizioni. Inoltre:

- l'indice -1 e' usato per rappresentare "nessun triangolo" (per esempio nella relazione TT quando l'adiacente e' la faccia infinita)
- per brevita' scriviamo "il vertice v" intendendo il "vertice di indice `v`" nella triangolazione, ed analogamente per i triangoli, ecc.

Funzioni di interrogazione

1. `float VertexX(int v)` ritorna la coordinata x del vertice v
2. `float VertexY(int v)` ritorna la coordinata y del vertice v

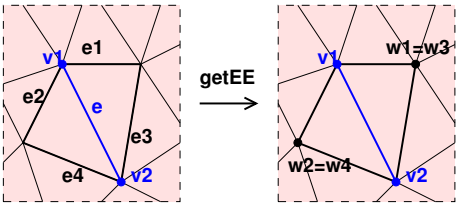

Figura 1

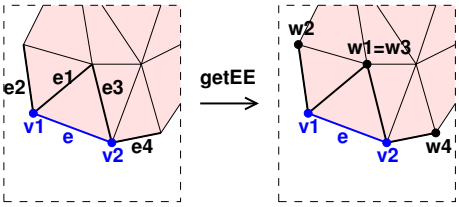

Figura 2

3. `int getPartialVT(int v)` ritorna l'indice del triangolo memorizzato in relazione VT parziale per il vertice v

4. `int getVT(int v, int * arr)` copia nell'array `arr` (che deve essere gia' allocato e di dimensione sufficiente) gli indici dei triangoli in relazione VT (totale) con il vertice v, ritorna il numero di tali triangoli

5. `int getVV(int v, int * arr)` copia nell'array `arr` (che deve essere gia' allocato e di dimensione sufficiente) gli indici dei vertici in relazione VV con il vertice v, ritorna il numero di tali vertici

6. `int getTV(int t, int * arr)` copia nell'array `arr` (che deve essere gia' allocato e di dimensione almeno 3) gli indici dei 3 vertici in relazione TV con il triangolo t, ritorna il numero di tali vertici (cioe' 3)

7. `int getTT(int t, int * arr)` copia nell'array `arr` (che deve essere gia' allocato e di dimensione almeno 3) gli indici dei 3 triangoli in relazione TT con il triangolo t, se qualcuno di essi non esiste (perche' t confina con la faccia esterna), l'indice corrispondente e' -1. Ritorna il numero di triangoli in relazione TT (cioe' 3 meno il numero di spigoli di t che confinano con la faccia esterna)

8. `int getExternalFace(int * arr)` copia nell'array `arr` (che deve essere gia' allocato e di dimensione sufficiente) gli indici dei vertici della faccia esterna infinita, ritorna il numero di tali vertici

9. `int getET(int v1, int v2, int * arr)`
   Copia nell'array `arr` (che deve essere gia' allocato e di dimensione almeno 2) gli indici dei triangoli in relazione ET con lo spigolo v1,v2, se lo spigolo e' di bordo uno dei due adiacenti e' la faccia infinita e l'indice corrispondente e' -1. Ritorna il numero di triangoli in relazione ET (cioe' 2 se spigolo interno e 1 se di bordo)

10. `int getEE(int v1, int v2, int * arr)` – *Figure 1 e 2*
    Sia `e` lo spigolo v1,v2 e sia EE(e) = e1, e2, e3, e4. Copia nell'array `arr` (che deve essere gia' allocato e di dimensione almeno 4) quattro vertici w1, w2, w3, w4 che sono nell'ordine gli estremi, diversi da v1, degli spigoli e1, e2, e gli estremi, diversi da v2, degli spigoli e3, e4. Ritorna il numero di vertici copiati (cioe' 4).

Funzioni di modifica

Ogni operazione di modifica viene eseguita solo se sono vere certe condizioni, altrimenti non viene eseguita. Ogni operazione ha un valore di ritorno intero che deve essere 1 se l'operazione e' stata eseguita con successo, 0 altrimenti.

Le operazioni sono state spiegate a lezione, qui le riprendiamo solo brevemente.

1. `int TriangleSplit(int t, float x, float y)` - *Figura 3.*
   Inserisce il punto (x,y) come nuovo vertice della triangolazione dividendo il triangolo `t` in tre triangoli. L'operazione viene eseguita solo se le coordinate x,y cadono all'interno di `t`.

2. `int AddEar(int v1, int v2, float x, float y)` - *Figura 4.*
   Implementa il caso particolare dell'operazione precedente in cui il punto cade fuori dal dominio. Inserisce il punto (x,y) come nuovo vertice della triangolazione congiungendolo con gli estremi dello spigolo v1,v2 e creando cosi' un nuovo triangolo. L'operazione viene eseguita solo se le coordinate x,y cadono fuori dal dominio, se v1,v2 sono gli estremi di uno spigolo di bordo (ovvero sono due vertici consecutivi sul perimetro della faccia esterna) e se v1 e v2 sono entrambi visibili da (x,y). Due punti si dicono visibili se il segmento che li unisce giace all'esterno del dominio (e' sufficiente controllare che non vi siano intersezioni col contorno della faccia esterna).

3. `int EdgeSplit(int v1, int v2, float x, float y)` - *Figure 5 e 6.*
   Inserisce il punto (x,y) come nuovo vertice della triangolazione dividendo in due lo spigolo di estremi v1,v2, e dividendo in due ciascun triangolo incidente su tale spigolo. L'operazione viene eseguita solo se le coordinate x,y cadono all'interno dello spigolo. Se lo spigolo e' interno alla triangolazione, vi sono due triangoli da dividere; se invece e' sul bordo, c'e' un solo triangolo da dividere.

4. `int VertexSplit(int v, int v1, int v2, float x1, float y1, float x2, float y2)` - *Figure 7 e 8.*
   Espande il vertice v in un nuovo spigolo con estremi nei punti (x1,y1) e (x2,y2) ed espande in triangoli gli spigoli v,v1 e v,v2.
   I triangoli prima incidenti in `v` e situati a sinistra della catena v1-v-v2 vanno ad essere incidenti in (x1,y1), quelli situati a destra vanno ad essere incidenti in (x2,y2).
   L'operazione viene eseguita solo se

   - il punto (x1,y1) giace a sinistra dei segmenti orientati v1-(x2,y2) e (x2,y2)-v2
   - nel caso il vertice `v` sia interno alla triangolazione: se le coordinate x1,y1 e x2,y2 cadono all'interno dell'unione dei triangoli incidenti in v
   - nel caso `v` sia sul bordo: se le coordinate x1,y1 e x2,y2 cadono o all'interno dell'unione dei triangoli incidenti in v oppure fuori dal dominio e visibili da v1 e v2
   - in ogni caso: se la sostituzione di v con v1 o v2 (a seconda del caso) dei triangoli che erano incidenti in v non inverte l'orientamento di nessuno di tali triangoli

   Nota: per orientamento si intende il verso (orario o antiorario) della svolta formata dai tre punti vertici del triangolo nell'ordine in cui sono memorizzati.

5. `int EdgeSwap(int v1, int v2)` - *Figura 9.*
   Scambia lo spigolo v1,v2 con l'altra diagonale del quadrilatero formato dai due triangoli incidenti sullo spigolo. L'operazione viene eseguita solo se lo spigolo ha due triangoli incidenti (cioe' non e' sul bordo della triangolazione) e se il quadrilatero formato da questi e' strettamente convesso.
   Nota: il quadrilatero e' strettamente convesso sse le sue due diagonali si intersecano.

6. `int VertexDelete(int v)` - *Figure 3 e 4.*
   Cancella il vertice v fondendo tra loro i triangoli che incidevano in esso. L'operazione viene eseguita solo se
   - il vertice `v` e' interno alla triangolazione ed ha esattamente tre triangoli incidenti (in questo caso e' l'operazione inversa di `TriangleSplit`)
   - oppure il vertice `v` e' sul bordo ed ha esattamente un triangolo incidente (in questo caso e' l'operazione inversa di `AddEar`)
7. `int EdgeMerge(int v1, int v2, int v3)` - *Figure 10 e 11.*
   Fonde gli spigoli v1,v2 e v2,v3 in un solo spigolo di estremi v1,v3 ed elimina tutti gli altri spigoli incidenti in v2 fondendo i triangoli che vi incidevano. L'operazione viene eseguita solo se
   - il vertice v2 e' interno alla triangolazione ed ha esattamente due vertici adiacenti oltre a v1,v3 messi in posizione alternata con v1 e v3
   - oppure v2 e' sul bordo, gli spigoli v1,v2 e v2,v3 sono sul bordo, e v2 ha un solo vertice adiacente oltre a v1,v3
8. `int EdgeCollapse(int v1, int v2, float x, float y)` - *Figure 7 e 8.*
   Collassa lo spigolo di estremi v1,v2 in un vertice di coordinate (x,y), eliminando i triangoli che incidevano su v1,v2. L'operazione viene eseguita solo se
   - (x,y) cade all'interno del quadrilatero formato dai due triangoli incidenti in v1,v2
   - e la sostituzione di v1 o v2 (a seconda del caso) con (x,y) nei triangoli che erano incidenti in v1 o v2 non inverte l'orientamento di nessuno di tali triangoli

Funzioni di input/output

Si assume che una triangolazione sia scritta su file in formato indicizzato (senza adiacenze) con la seguente sintassi:

- numero di vertici
- lista di vertici (in numero pari a quanto dichiarato sopra) come coppie di numeri reali x y
- numero di triangoli
- lista di triangoli (in numero pari a quanto dichiarato sopra) come terne di indici interi

I separatori possono essere spazi, tabulazioni, ritorni a capo anche piu' di uno e mischiati fra loro, non sono ammessi altri caratteri di interpunzione (parentesi, virgole...).
Sono dati alcuni file di esempio: `es1.txt`, `es2.txt`.

1. `int ReadTriangulation(FILE * fd)` legge da file una triangolazione e la memorizza nella struttura dati dopo aver ricostruito le informazioni non contenute nel file (relazioni VT parziale e TT, contorno della faccia esterna infinita)
2. `int WriteTriangulation(FILE * fd)` scrive su file la triangolazione

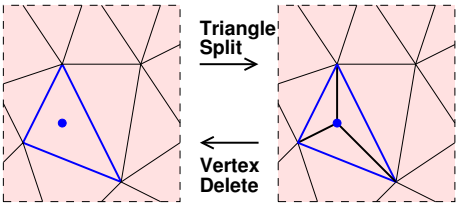

Figura 3

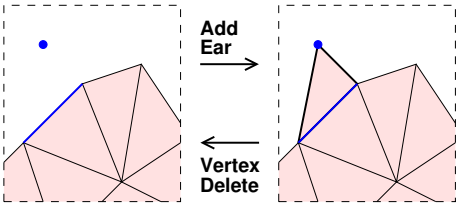

Figura 4

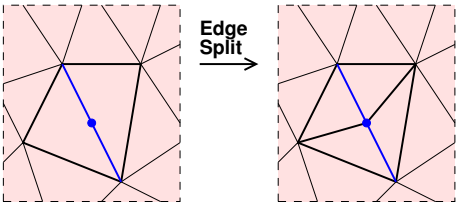

Figura 5

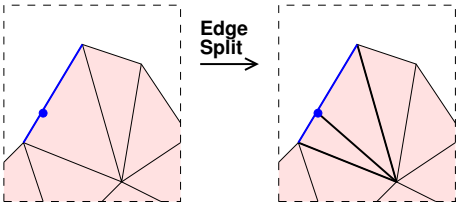

Figura 6

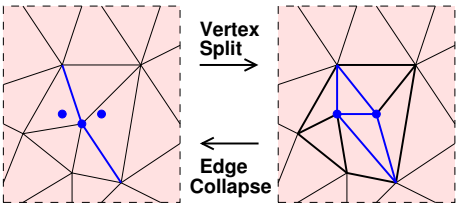

Figura 7

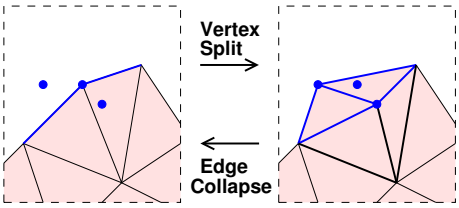

Figura 8

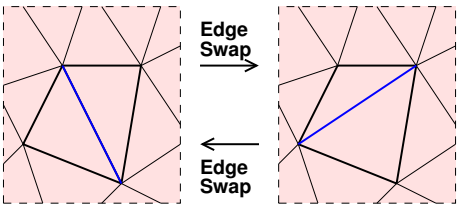

Figura 9

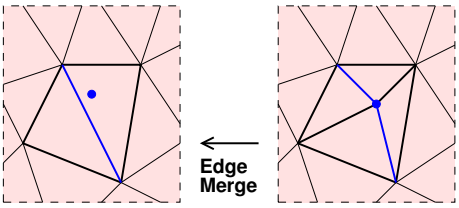

Figura 10

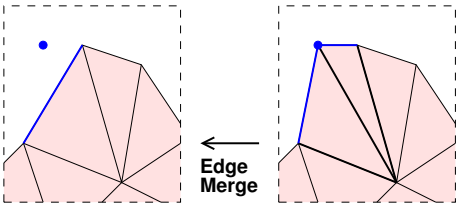

Figura 11

3 Visualizzazione

Viene fornito un programma grafico che, compilato assieme alla vostra implementazione, permettera' di visualizzare i risultati e controllare che siano corretti.

Il programma fa uso delle librerie grafiche OpenGL e Glut ed e' fornito con il supporto necessario per utilizzarlo sotto Linux nei laboratori sw1 e sw2. E' altresi' possibile eseguirlo in ambiente Windows ma per questo non viene dato supporto tecnico.

Il programma fa affidamento sul fatto che sia rispettata *esattamente la sintassi* specifica per le operazioni.

Sono dati i file:

- `trigraph.c` il programma grafico
- `tridummy.h` e `tridummy.c` programma "stupido" che codifica una triangolazione prefissata (quella del file di esempio `es1.txt`) e implementa tutte le funzioni di modifica senza fare nulla, ma stampando un messaggio su standard output
- `makefile` che per adesso compila `trigraph.c` assieme a `tridummy.c`.

Si compila con `make trigraph` e si esegue con `./trigraph`

Poi dovrete editare il makefile sostituendo a `tridummy` il nome del file contenente la vostra implementazione.
Potete usare `tridummy` come base di partenza per la vostra implementazione, completando a poco a poco le varie funzioni.

Funzionalita' del programma

Presenta un'unica finestra nella quale avviene il disegno della triangolazione e su cui e' attivo un menu' pop-up. Sono attivi anche alcuni tasti della tastiera.

Tasti attivi:

- `Q` o `q` termina l'esecuzione del programna
- `ESC` annulla l'operazione in corso, se per caso vi si bloccasse il programma provate a premere ESC, puo' darsi che sia rimasto in attesa di parametri per una qualche operazione

Il menu' contiene le voci:

- `carica` carica una triangolazione, chiede digitare il nome del file, il file viene cercato nella directory corrente
- `salva` salva la triangolazione, chiede digitare il nome del file, il file viene messo nella directory corrente
- `mostra numeri` entra nella modalita' in cui cliccare col mouse sopra un vertice, spigolo o triangolo mostra il suo numero (per uno spigolo, i numeri dei suoi vertici), per i vertici viene anche evidenziato il numero del triangolo nella relazione VT parziale – questa e' la modalita' di *default*
- `mostra V` entra nella modalita' in cui cliccare col mouse sopra un vertice, spigolo o triangolo evidenzia i vertici ad esso collegati mediante la relazione VV, VE o TV (a seconda del caso), cliccare sullo sfondo evidenzia i vertici della faccia esterna
- `mostra T` entra nella modalita' in cui cliccare col mouse sopra un vertice, spigolo o triangolo evidenzia i triangoli ad esso collegati mediante la relazione VT, ET o TT (a seconda del caso)
- `mostra E` entra nella modalita' in cui cliccare col mouse sopra uno spigolo evidenzia gli spigoli ad esso collegati mediante la relazione EE
- `mostra V e T` entra nella modalita' in cui cliccare col mouse sopra un vertice, spigolo o triangolo evidenzia sia i vertici che i triangoli ad esso collegati

- `raffina` entra nella modalita' in cui cliccare col mouse sopra un vertice o triangolo ha il seguente effetto:
  - se si clicca su un triangolo, viene invocata l'operazione `TriangleSplit` di quel triangolo nel punto dato dalla posizione del mouse
  - se si clicca su uno spigolo, viene invocata l'operazione `EdgeSplit` di quello spigolo nel punto dato dalla posizione del mouse
  - se si clicca su un vertice, viene invocata l'operazione `VertexSplit` di quel vertice, in questo caso all'utente e' richiesto di cliccare in altri due punti che saranno gli estremi dello spigolo risultante (l'utente avra' cura di posizionare i punti in modo ammissibile) e di selezionare tramite click due spigoli che dovranno essere espansi in triangoli (l'utente avra' cura di scegliere una coppia di spigoli ammissibile)
  - se si clicca sullo sfondo (faccia esterna), viene invocata l'operazione `AddEar`, in questo caso all'utente e' richiesto di selezionare tramite click uno spigolo (l'utente avra' cura di scegliere uno spigolo di bordo e ammissibile)
- `semplifica` entra nella modalita' in cui cliccare col mouse ha il seguente effetto:
  - se si clicca su un vertice interno con esattamente tre triangoli incidenti, oppure su un vertice di bordo con esattamente un triangolo incidente, viene invocata l'operazione `VertexDelete` per cancellare quel vertice
  - se si clicca su un vertice interno con esattamente quattro triangoli incidenti, oppure sul bordo con esattamente due triangoli incidenti, viene invocata l'operazione `EdgeMerge` per cancellare quel vertice, in questo caso all'utente e' richiesto di selezionare tramite click due spigoli incidenti, che dovranno essere fusi (l'utente avra' cura di selezionare una coppia di spigoli ammissibile)
  - se si clicca su uno spigolo viene invocata l'operazione `EdgeCollapse` per collassare quello spigolo, il vertice risultante viene posto alla posizione del mouse
- `scambia` entra nella modalita' in cui cliccare col mouse su uno spigolo invocata l'operazione `EdgeSwap` su quello spigolo
- `esci` termina l'esecuzione del programma

La modalita' corrente come anche altri messaggi (sull'esito con successo o errore dell'operazione, ecc.) sono mostrati in sovra-impressione alla finestra.

Convenzioni grafiche

In condizioni normali la triangolazione e' mostrata su sfondo bianco con triangoli riempiti in giallino chiaro, spigoli in rosso e vertici in nero.

Nel visualizzare le relazioni topologiche, il vertice o triangolo del quale si visualizzano le relazioni e' evidenziato in verde mentre i vertici / triangoli in relazione con esso sono evidenziati in toni di colore sfumati da rosso chiaro (il primo nella relazione) a blu chiaro (l'ultimo della relazione) passando attraverso i toni del grigio-violetto.

4 Scadenza e modalita' di consegna

Consegnare per *posta elettronica* all'indirizzo magillo@disi.unige.it un *unico file compresso* in formato .tgz oppure .zip contenente:

- *sorgenti* del programma, che deve essere scritto in ANSI C o C++.
- *makefile* modificato per compilare il vostro programma
- *brevissima documentazione* (un paio di pagine al massimo), preferibilmente in formato TXT o HTML, contenente:
  - i *nomi* dei componenti del gruppo
  - *soltanto* i dettagli implementativi non gia' descritti nel presente testo (per esempio le convenzioni usate per l'ordine in cui sono memorizzati gli elementi nelle relazioni, ecc.)

La *scadenza* per la consegna e': *20 febbraio 2004*.