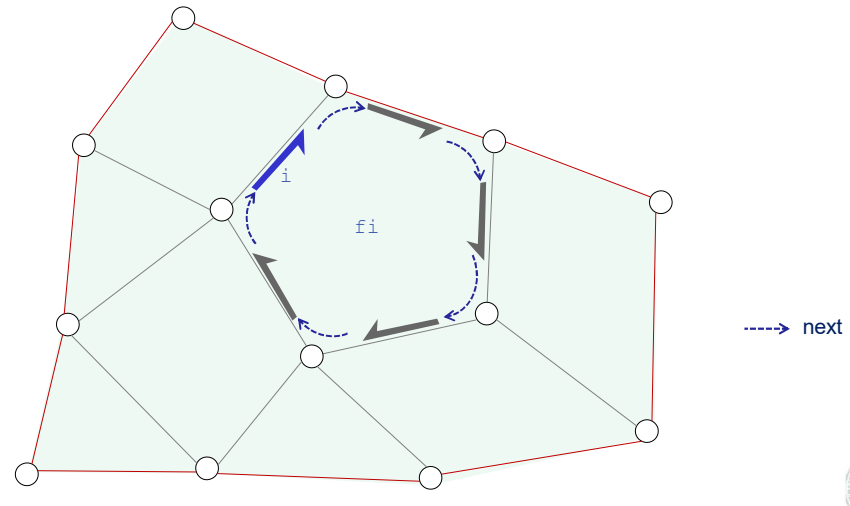



Esempio: conta quanti lati ha una faccia

✓ Dato un half-edge di indice i , corrispondente ad una data faccia, trova quanti lati ha questa faccia



-----> next



118


Esempio: conta quanti lati ha una faccia

✓ Dato un half-edge di indice i , corrispondente ad una data faccia, trova quanti lati ha questa faccia

```
int fi = he[i].opp;
if (fi == -1) ... /* non esiste la faccia */
int start = i; // half edge di partenza
int lati = 0;
do {
    lati ++;
    i = he[i].next;
} while (i!=start);
```

(era un half edge di bordo)

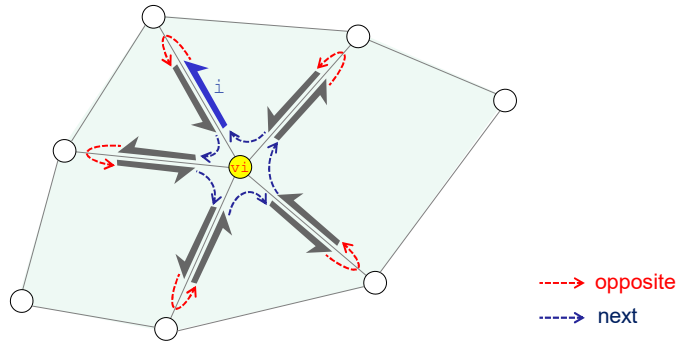
(nota: il ciclo finisce quando torno al punto di partenza)



119

Esempio: scansione di vertici attorno ad un vert

- ✓ Dato un half-edge di indice i , che emana da un vertice v_i , trova tutti i vertici v_j nella stella di v_i



120

Esempio: scansione di vertici attorno ad un vert

- ✓ Dato un half-edge di indice i , che emana da un vertice v_i , trova tutti i vertici v_j nella stella di v_i

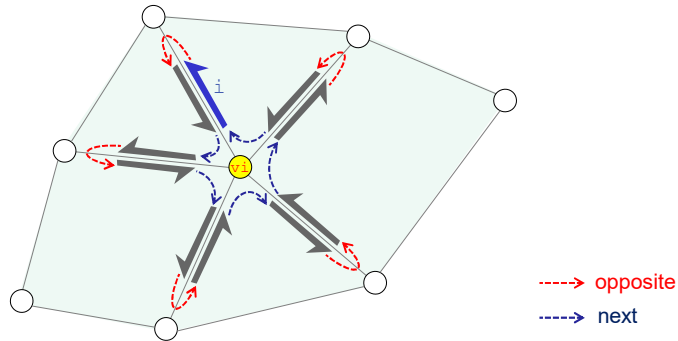
```
int vi = he[i].v;  
  
int start = i; // half edge di partenza  
  
do {  
    i = he[i].opp;  
    int vj = he[i].v;  
  
    /* qui: fai qualcosa con vertice vj */  
  
    i = he[i].next;  
} while (i!=start);
```

(nota: il ciclo finisce quando torno al punto di partenza)

121

Scansione di facce attorno ad un vert

- ✓ Dato un half-edge di indice i , che emana da un vertice v_i , trova tutte le faccie f_j nella adiacenti a v_i



(nota: il ciclo finisce quando torno al punto di partenza)



122

Esempio: scansione di facce attorno ad un vert

- ✓ Dato un half-edge di indice i , che emana da un vertice v_i , trova tutte le faccie f_j nella adiacenti a v_i

```
int vi = he[i].v;  
  
int start = i; // half edge di partenza  
  
do {  
    int fi = he[i].v;  
    /* qui: fai qualcosa con faccia fi */  
    /* se esiste: potrebbe essere -1 */  
  
    i = he[i].opp;  
    i = he[i].next;  
  
} while (i!=start);
```

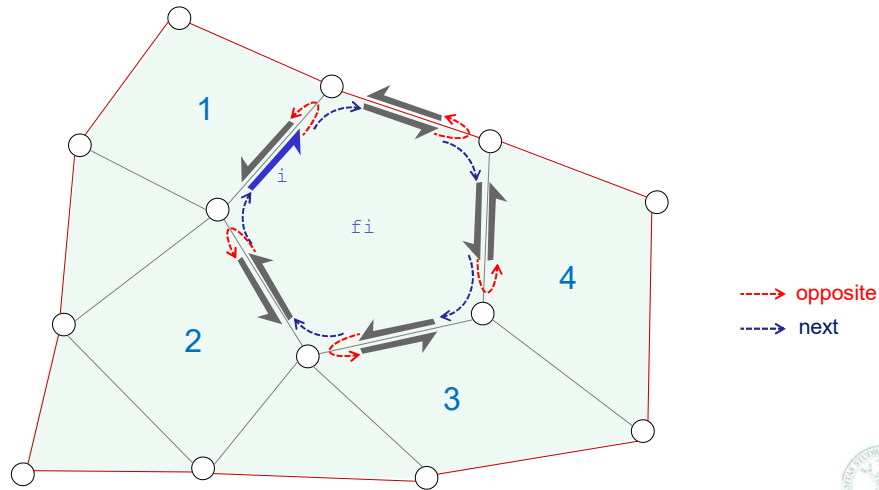
(nota: il ciclo finisce quando torno al punto di partenza)



123

Esempio: contare la faccie adiacenti da una faccia

✓ dato un indice di halfedge i , se confina con una faccia f_i , allora conta quante_facce adiacenti a f_i ci sono



124

Esempio: contare la faccie adiacenti da una faccia

✓ dato un indice di halfedge i , se confina con una faccia f_i , allora conta quante_facce adiacenti a f_i ci sono

```
int fi = he[i].f;  
if (fi == -1) ... /* non esiste la faccia */  
int start = i;  
int quante_facce = 0;  
do {  
    int j = he[i].opp;  
    if (he[j].fi != -1) quante_facce ++;  
    i = he[i].next;  
} while (i != start);
```

125

Esempio: visita di un bordo (insieme di edge)

✓ Sia dato un indice di halfedge i , confinante con una faccia f_i ;
se l'edge corrispondente è di bordo, allora scandisci tutti gli edge che fanno parte dello stesso bordo:

---> opposite
- - -> next

126

Esempio: visita di un bordo (insieme di edge)

✓ Sia dato un indice di halfedge i , confinante con una faccia f_i ;
se l'edge corrispondente è di bordo, allora scandisci tutti gli edge che fanno parte dello stesso bordo:


```
i = he[i].opp;  
int fi = he[i].f;  
if (fi == -1) {  
    int start = i;  
    do {  
        int i = he[i].next;  
        /* fa qualcosa con i... */  
    } while (i!=start);  
}
```

127

Esempio: visita di un bordo (insieme di edge)

Funziona anche su mesh con "dangling edges" (edges non adiacenti ad alcuna faccia)

campo next



128

Computo della normale di un triangolo

(Cioè del suo orientamento nello spazio)

due "edge vectors"

$$\vec{n} = (p_1 - p_0) \times (p_2 - p_0)$$


"area vector"
 un vettore ortogonale al triangolo,
 lungo quanto la doppia area del triangolo

normalizzazione

$$\hat{n} = \frac{\vec{n}}{\|\vec{n}\|}$$

"normale" del triangolo (vett unitario)

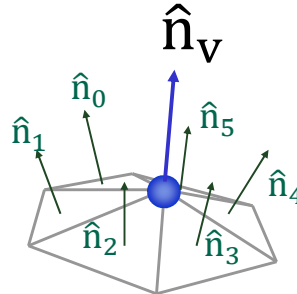
E' la doppia area del triangolo



129

Normali come attributo per vertice

Normale di un vertice
condiviso da k triangoli:
la loro media (interpolazione)



$$\hat{n}_v = \frac{\hat{n}_0 + \dots + \hat{n}_k}{\|\hat{n}_0 + \dots + \hat{n}_k\|}$$

Nota: dato che si normalizza il risultato,
non c'è bisogno di divider per k la somma delle normali per faccia per calcolare la loro media



131

Calcolo delle normali per faccia di una mesh (note)

- ✓ La normale delle facce triangolari è data dal semplice visto sopra
 - ⇒ per ciascuna faccia
- ✓ E' detta «normale» geometrica
 - ⇒ perché corrisponde effettivamente alla normale (costante) del piano su cui giace il triangolo
 - ⇒ che esiste sempre – a differenza di quads
 - ⇒ eccez: triangoli degeneri: 3 vertici allineati es, 2 vertici coincidenti
- ✓ Questa normale è orientata consistentemente solo per mesh ben orientate
 - ⇒ ad es, sempre verso l'esterno in una mesh chiusa



132

Calcolo delle normali per vertice di una mesh (note)

- ✓ La normale per vertice di una mesh non è definita in modo univoco per via geometrica
 - ⇒ quindi dobbiamo «inventarcene» una.
- ✓ La domanda che ci dobbiamo porre è:
 - ⇒ «se questa mesh (che è composta di facce *piatte*) modella una superficie invece *curva*, quale sono le normali di questa superficie?»
 - ⇒ Non esiste una risposta univoca!
 - ⇒ Dipende da quale superficie intendo rappresentare.
- ✓ Strategia spesso utilizzata in pratica:
 - ⇒ usare una interpolazione delle facce adiacenti (per es, la media)
 - ⇒ Altre strategie sono possibili:
- ✓ *le normali (per vertice) fanno parte del modello*



133

Algoritmo per computo di normale per vertice

Passo 1: computo delle normali per faccia

- ✓ Per facce non-triangulari: si possono mediare le normali costruite su ogni wedge della faccia
 - ⇒ prodotto cross dei due edge vectors adiacenti al wedge
 - ⇒ Per trovarli, navigo gli half edge attorno alla faccia

Passo 2: computo delle normali per vertice

- ✓ Per ciascun vertice: ciclo attorno a tutte le facce adiacenti e cumulando la loro normale. Alla fine del ciclo, normalizzo

Esiste un algoritmo efficiente (cioè lineare) che usa solo la struttura «lista-facce» per la connettività invece che la struttura ad «half-edge». *Sapresti identificare questo algoritmo?*



134