

Fondamenti di Grafica Tridimensionale

Paolo Cignoni
p.cignoni@isti.cnr.it
<http://vcg.isti.cnr.it/~cignoni>

1

Incremental Simp Method

The common framework:

- **loop**
 - **select** the element to be deleted/collapsed;
 - **evaluate approximation** introduced;
 - **update** the mesh after deletion/collapse;
- until mesh **size/precision** is satisfactory;

EG99 Tutorial

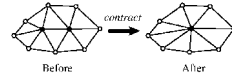
2

Quadric Error Metrics

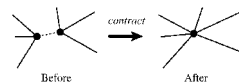
Simplification using Quadric Error

Metrics [Garland et al. Sig'97]

- Based on incremental **edge-collapsing**



- **but** can also collapse vertex couples which are **not connected** (topology is not preserved)



EG99 Tutorial

3

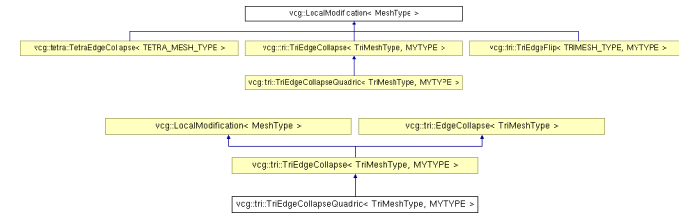
The main simplification loop

```
vcg::LocalOptimization<MyMesh> DeciSession(mesh);  
DeciSession.Init<MyTriEdgeCollapse >();  
DeciSession.SetTargetSimplices(FinalSize);  
DeciSession.SetTimeBudget(0.5f);  
  
while(DeciSession.DoOptimization() && mesh.fn>FinalSize)  
    printf("Current Mesh size %7i heap sz %9i err %9g \r",  
          mesh.fn,DeciSession.h.size(),DeciSession.currMetric);  
  
printf("mesh %d %d Error %g \n",  
       mesh.vn,mesh.fn,DeciSession.currMetric);
```

4

Classi in gioco

- LocalOptimization
 - Classe astratta per il loop di ottimizzazione
- LocalModification
 - Classe astratta per una generica operazione che modifica la mesh localmente con un certo costo
- EdgeCollapse
- TriEdgeCollapse
 - Particolare local modification
- TriEdgeCollapse Quadric



5

6

Local Modification

```

template <class MeshType> class LocalModification
{
public:
    typedef typename LocalOptimization<MeshType>::HeapType HeapType;
    typedef typename MeshType::ScalarType ScalarType;

    inline LocalModification();
    virtual ~LocalModification();

    virtual ModifierType IsOfType() = 0; // return the type of operation
    // return true if the data have not changed since it was created
    virtual bool IsUpToDate() = 0;
    // return true if no constraint disallow this operation to be performed (ex: change
    // of topology in edge collapses)
    virtual bool IsFeasible() = 0;
    // Compute the priority to be used in the heap
    virtual ScalarType ComputePriority()=0;
    // Return the priority to be used in the heap (implement static priority)
    virtual ScalarType Priority() const =0;
    // Perform the operation and return the variation in the number of simplices (>0
    // is refinement, <0 is simplification)
    virtual void Execute(MeshType &m)=0;
    // perform initialization
    static void Init(MeshType &m, HeapType&);
    virtual const char *Info(MeshType &) {return 0;}
    // Update the heap as a consequence of this operation
    virtual void UpdateHeap(HeapType&)=0;
}; //end class local modification
    
```

7

Local Modification

- Classe astratta generica
 - Potrebbe essere un edge collapse
 - Uno swap
 - Un vertex deletion ecc.
- Astrarre una generica operazione di modifica locale alla mesh
 - Adatta ad essere prioritizzata
 - Deve saper dare una prioritari'
 - Sapersi applicare alla mesh
 - Sapere se e' sempre valida

8

EdgeCollapse e TriEdgeCollapse

- EdgeCollapse
 - Classe astratta per rappresentare un collasso di un edge su una generica mesh
 - Non sa nulla di prioritá quadriche ecc
- TriEdgeCollapse
 - Generica local op basata su collasso
 - Sa aggiornare lo heap
 - Eseguirsi, sapere se e' valida ecc.
 - Da questa si deriva quella con le quadriche

9

```
template <class TRI_MESH_TYPE>
class EdgeCollapse
{
    typedef typename vcg::face::VFIterator<FaceType> VFI;
    typedef typename std::vector<vcg::face::VFIterator<FaceType> >
        VFIVec;

    static VFIVec & AV0(){static VFIVec av0; return av0;}
    static VFIVec & AV1(){static VFIVec av1; return av1;}
    static VFIVec & AV01(){static VFIVec av01; return av01;}

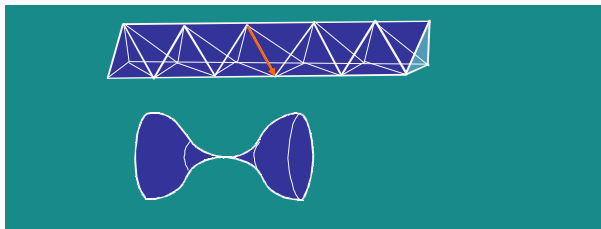
    bool LinkConditions(EdgeType pos);
    void FindSets(EdgeType &p)bool LinkConditions(EdgeType pos);
    int DoCollapse(EdgeType & c, const Point3<ScalarType> &p);
}

```

10

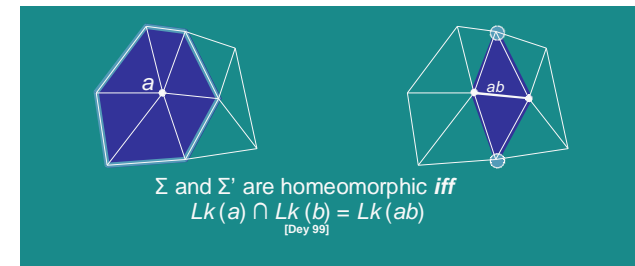
Topology Preservation

- 2-Manifold
 - A surface Σ in \mathbf{R}^2 such that any point on Σ has an open neighborhood homeomorphic to an open disc or to half an open disc in \mathbf{R}^2
- A edge collapse can create non manifold situations



Topology Preservation

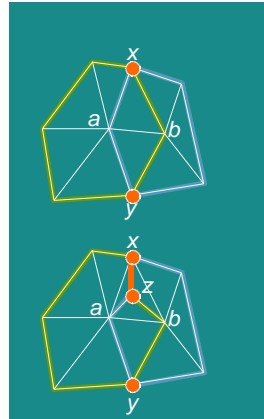
- Let Σ be a 2 simplicial complex **without boundary**
- Σ' is obtained by collapsing the edge $e = (ab)$
- Let $Lk(\sigma)$ be the set of all the faces of the co-faces of σ disjoint from σ



Topology Preservation

$$Lk(a) \cap Lk(b) = \{x, y\} = Lk(ab)$$

$$Lk(a) \cap Lk(b) = \{x, y, z, zx\} \neq \{y, z\} = Lk(ab)$$



doCollapse

```

for(i=AV0().begin();i!=AV0().end();++i)
{
  FaceType & f = *((*i).f);
  assert(f.V((*i).z) == c.V(0));
  vcg::face::VFDetach(f, ((*i).z+1)%3);
  vcg::face::VFDetach(f, ((*i).z+2)%3);
  f.SetD();
  n_face_del++;
}

//set Vertex Face topology
for(i=AV0().begin();i!=AV0().end();++i)
{
  (*i).f->V((*i).z) = c.V(1); // In tutte le facce
  incidenti in v0, si sostituisce v0 con v1
  (*i).f->VFp((*i).z) = (*i).f->V((*i).z)->VFp();
  // e appendo la lista di facce incidenti in v1 a questa faccia
  (*i).f->VF1((*i).z) = (*i).f->V((*i).z)->VF1();
  (*i).f->V((*i).z)->VFp() = (*i).f;
  (*i).f->V((*i).z)->VF1() = (*i).z;
}

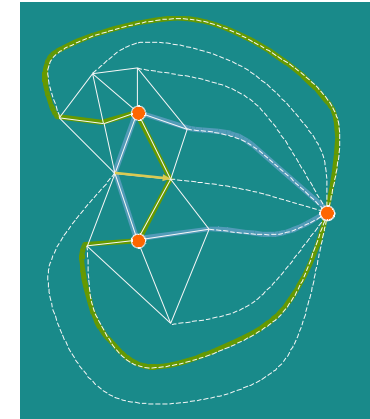
c.V(0)->SetD();
c.V(1)->P()=p;
return n_face_del;

```

15

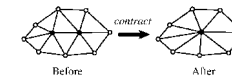
Topology Preservation

- Mesh with boundary can be managed by considering a dummy vertex v_d and, for each boundary edge e a tetrahedron connecting e with v_d
- Think it wrapped on the surface of a sphere
-



Lazy heap

- Si suppone di avere uno heap con tutte le operazioni
- Estraggo da heap e aggiorno la mesh
 - tali operazioni invalidano/modificano la mesh e quindi le priorità/validità di parte delle azioni già presenti nello Heap



16

Lazy Heap

- Due Soluzioni
 - Link espliciti elementi mesh->heap e aggiornamento dello stesso
- Lazy update
 - Si mettono nello heap tutte le nuove operazioni con la nuova priorità
 - Quando si estrae un'op dall heap si controlla che sia sempre valida
 - Di tanto in tanto garbage collection sullo heap

17

Marche incrementali

- Strumento generico per marcare oggetti in una collezione con
 - $C(\text{mark elem}) = O(1)$
 - $C(\text{unmark elem}) = O(1)$
 - $C(\text{unmark All Elem}) = O(1)$
- Memorizza per ogni elem un intero *mark* invece di un bit
- Esiste una marca globale a livello della collezione di elementi

18

Marche incrementali

- Un oggetto è marcato se
 - $\text{elem.mark} == \text{global.mark}$
- Marcatura di un elem
 - $\text{elem.mark} := \text{global.mark}$
- Smarcatura globale
 - $\text{global.mark}++$
- Spesso le marche vengono dette anche marche **temporali** per indicare che dicono quando un certo elem è valido

19

Validità collasso

- Dati
 - Ogni vertice ha una marca temporale:
 - quando e' stato modificato l'ultima volta
 - Ogni collasso (coppia di vertici) ha una marca temporale
 - quando è stato inserito nello heap
- Un collasso è valido se
 - I due vertici non sono stati cancellati
 - Il collasso e' stato messo nello heap piu recentemente della data di ultima modifica dei vertici

20

Quadric Error for Surfaces

- Let $\mathbf{n}^T \mathbf{v} + d = 0$ be the equation representing a plane
- The squared distance of a point \mathbf{x} from the plane is

$$D(\mathbf{x}) = \mathbf{x}(\mathbf{nn}^T)\mathbf{x} + 2d\mathbf{n}^T\mathbf{x} + d^2$$

- This distance can be represented as a quadric

$$Q = (A, \mathbf{b}, c) = (\mathbf{nn}^T, d\mathbf{n}, d^2)$$

$$Q(\mathbf{x}) = \mathbf{x}A\mathbf{x} + 2\mathbf{b}^T\mathbf{x} + c$$

21

Quadric

- The boundary error is estimated by providing for each boundary vertex v a quadric Q_v representing the sum of the all the squared distances from the faces incident in v
 - The error of collapsing an edge $e=(v,w)$ can be evaluated as $Q_w(v)$.
 - After the collapse the quadric of v is updated as follow $Q_v = Q_v + Q_w$

22