Simplification and Refinement

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

Approximating surfaces with triangle meshes

Managing the quantity of geometry vs the quality of the representation

Assumptions:

Surfaces represented by triangle meshes

Accuracy of the approximation is proportional to the number of triangles

Complexity of a surface is proportional to the number of triangles

Objective:

Always produce the simplest mesh that satisfies the accuracy required by the application

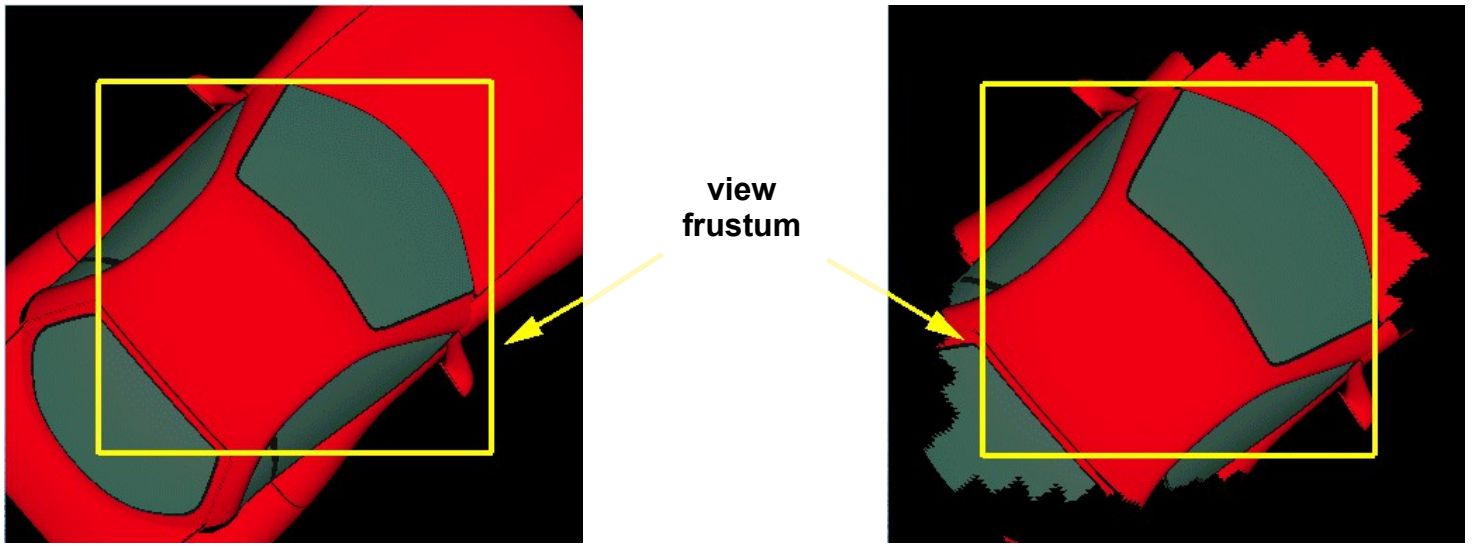
Reducing rendering cost

Simplifying is not the only way to reduce the needed number of triangles
At rendering time we can do:

View Frustum Culling

Drawing only what is in the view frustum

…



culling OFF **culling ON**

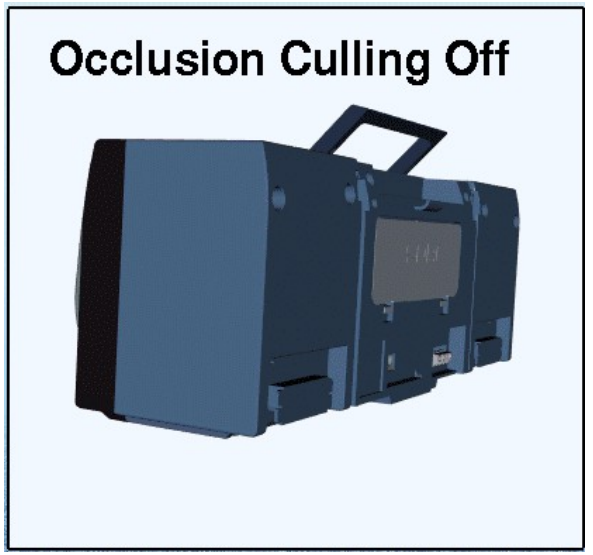
Reducing Complexity

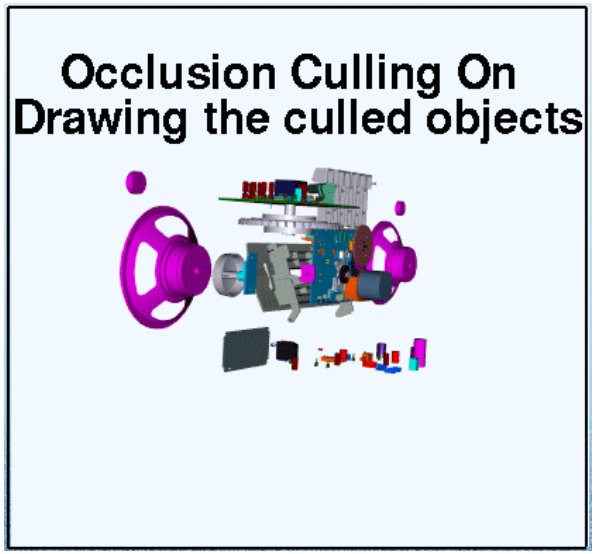
...

Occlusion/Visibility culling

Drawing only what it is probably visible

➔ ***Improve performances, but not sufficient for very large datasets***





Simplification

From a computational Point of view, fixed the size of the solution find the ***best*** representation

The optimal solution in its general form is NP Hard

Heuristics works well

(even with bounds)

Metrics

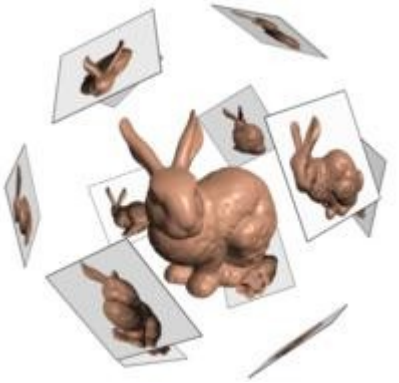
Computing the “difference” between two meshes

Geometric way

- Hausdorff

Perceptual

- Working in Image space
  - Many renderings,
    computing the difference,
    possibly in a human-like way



Appearance similarity

Difference between two images: (trivial)

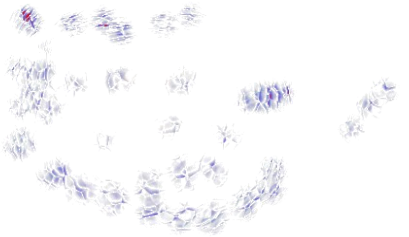
$$D(I_1, I_2) = \frac{1}{n^2} \sum_x \sum_y d(I_1(x, y), I_2(x, y))$$

Difference between two objects:

Integrate the above over all possible views

I_1

I_2



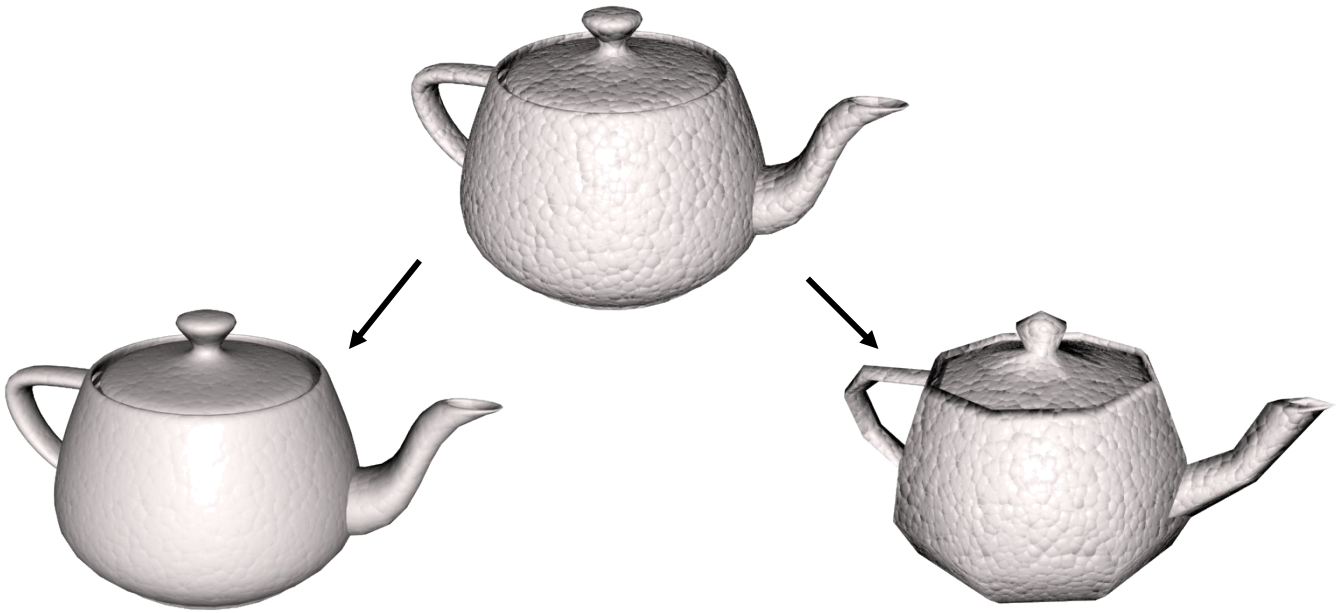
$I_1 - I_2$

Approximation error

Quantifies the notion of “similarity”

Two kinds of similarity:

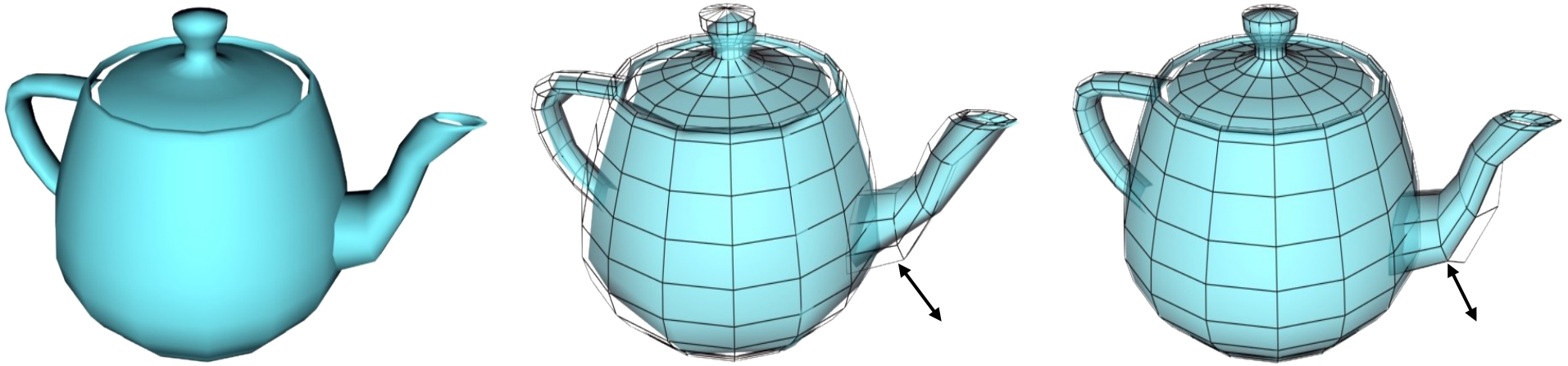
- **Geometric similarity** (surface deviation)
- **Appearance similarity** (material, normal…)



Geometric similarity

Two main components:

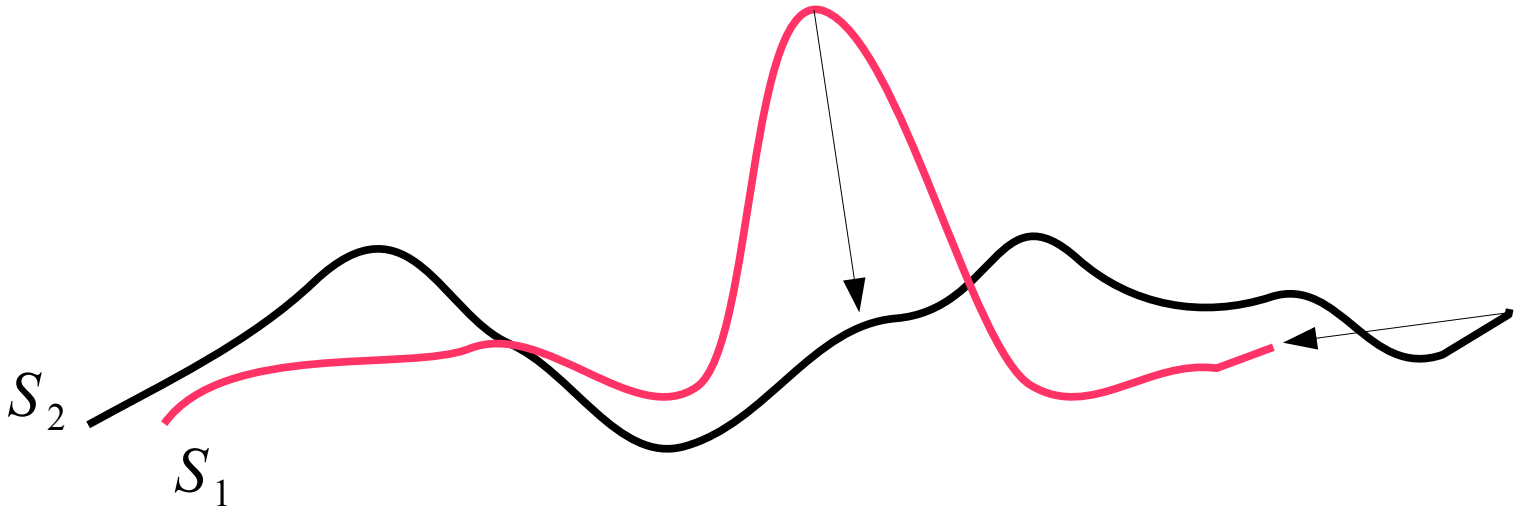
- Distance function
- Function Norm:
  - L_2: average deviation
  - L_{inf} : maximum deviation - Hausdorff distance



[Image by C. Andujar]

Hausdorff Distance

$$D_H(S_1, S_2) = \max_{x \in S_1} \left(\min_{y \in S_2} D(x, y) \right)$$



Symmetric version:

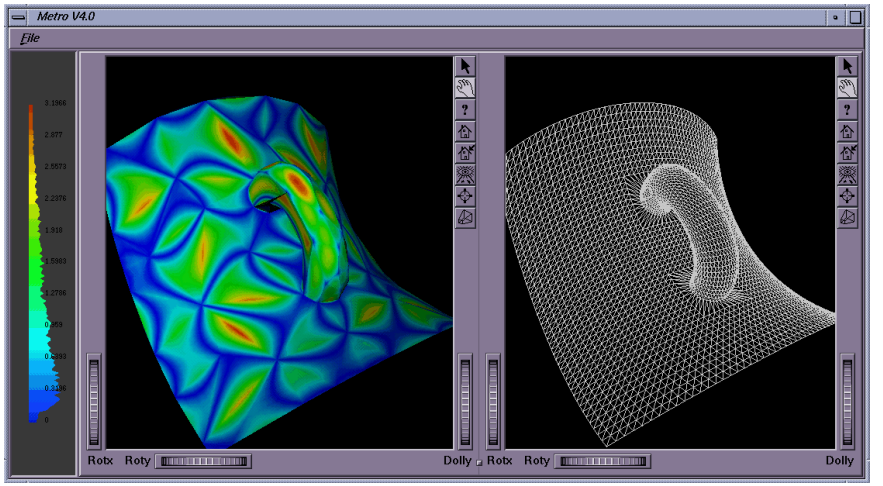
$$D(S_1, S_2) = \max \{ D_H(S_1, S_2), D_H(S_2, S_1) \}$$

Computing Hausdorf Distance

In a approximate way

Sample a surface

- Points uniformly distributed over it
- For each point x compute $\min_{y \in S_2} D(x, y)$
  - points usually very near to surface,
  - UG works well, better than trees
- Avg are useful too...



Simplification Algorithms

Simplification approaches:

incremental methods based on local updates

mesh decimation

[Schroeder et al. `92, ... + others]

energy function optimization & progressive meshes

[Hoppe et al. `93, Hoppe '96, Hoppe '97]

quadric error metrics [Garland et al. '97]

coplanar facets merging

[Hinker et al. `93, Kalvin et al. `96]

clustering

[Rossignac et al. `93, ... + others]

wavelet-based

[Eck et al. `95, + others]

volume-based

[He+ 95, He+ 96, Andujar+ 02]

Incremental methods based on ***local updates***

All of the methods such that :

simplification proceeds as a sequence of small changes of the mesh (in a greedy way)

each update reduces mesh size and [~monotonically] decreases the approximation precision

Different approaches:

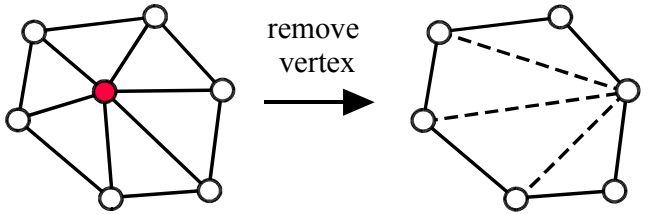
mesh decimation
energy function optimization
progressive meshes
quadric error metrics

… Incremental methods based on ***local updates ...***

Local update actions:

No. Faces

vertex removal

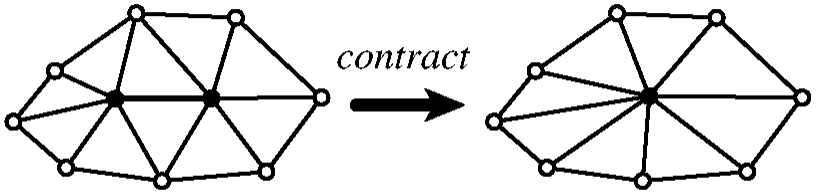


n-2

edge collapse

preserve location

new location



n-2

Before

After

Incremental Simp Method

The common greedy framework:

While *mesh size/precision is satisfactory*

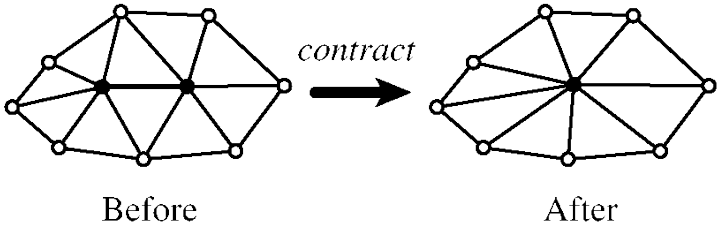
Select from a priority queue the best element to simplify

Perform simplification modifying the mesh

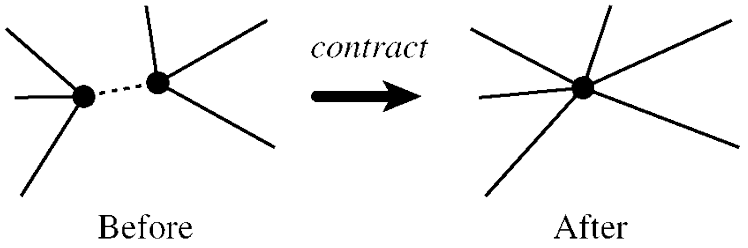
Ppdate priority queue according to the mods;

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)



Quadric Error for Surfaces

Let $\mathbf{n}^T\mathbf{v} + d = 0$ be the equation representing a plane

The squared distance of a point $\boldsymbol{x}$ from the plane is

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

This distance can be represented as a quadric

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

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

Quadric

The error is estimated by providing for each 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).

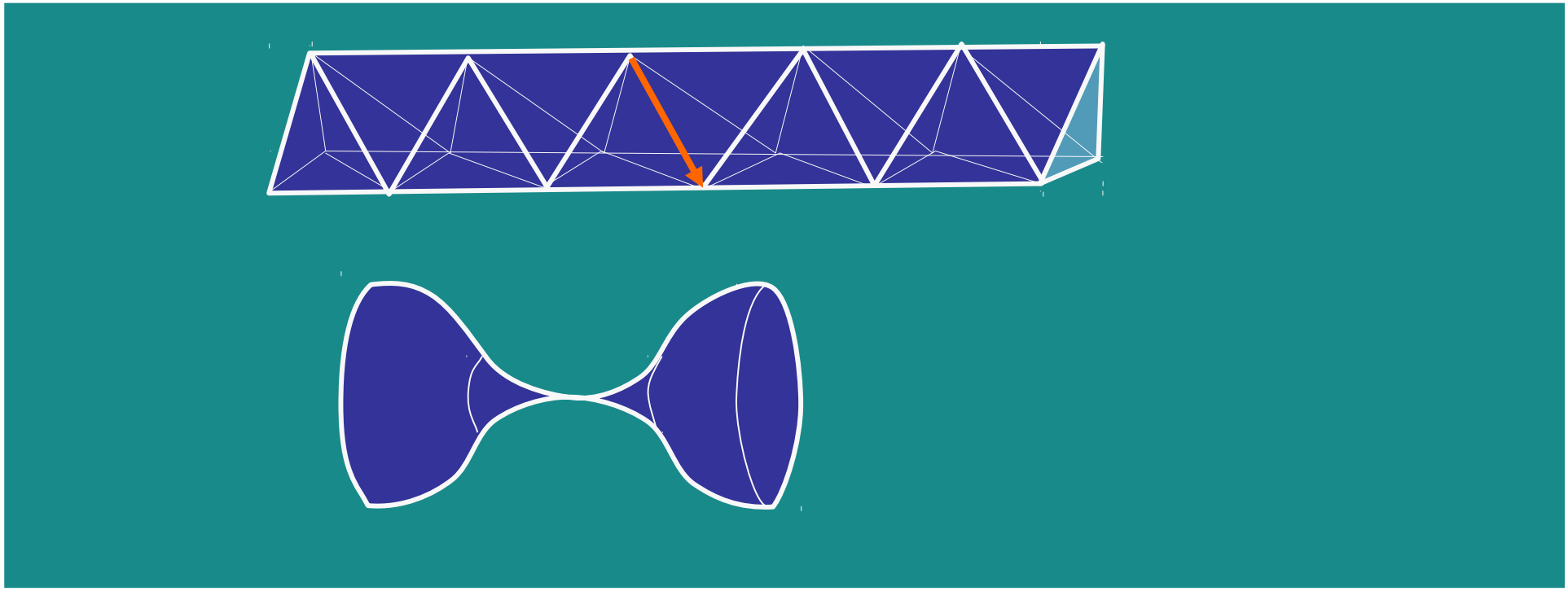
After the collapse the quadric of v is updated as follow $Q_v = Q_v + Q_w$

Topology Preservation

2-Manifold

A surface Σ in $\boldsymbol{R}^2$ such that any point on Σ has an open neighborhood homeomorphic to an open disc or to half an open disc in $\boldsymbol{R}^2$

A edge collapse can create non manifold situations

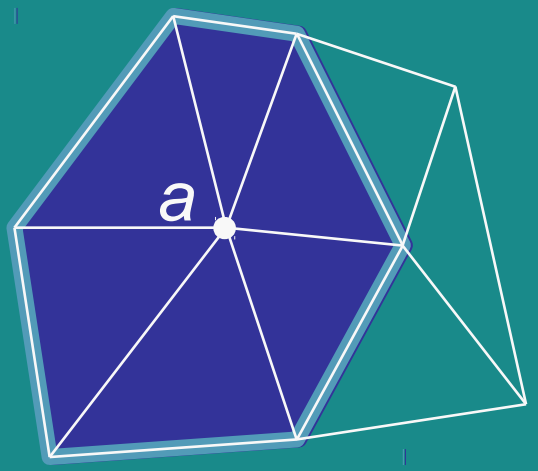


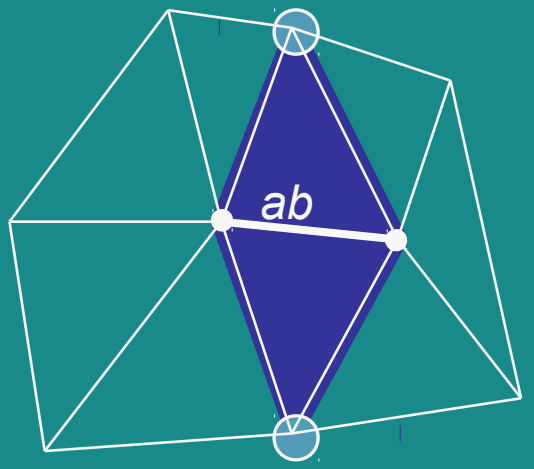
Topology Preservation

Let Σ be a 2 simplicial complex ***without boundary***

Σ’ is obtained by collapsing the edge *e* = (*ab*)

Let *Lk* (σ) be the set of all the faces of the co-faces of σ disjoint from σ





Σ and Σ’ are homeomorphic ***iff***

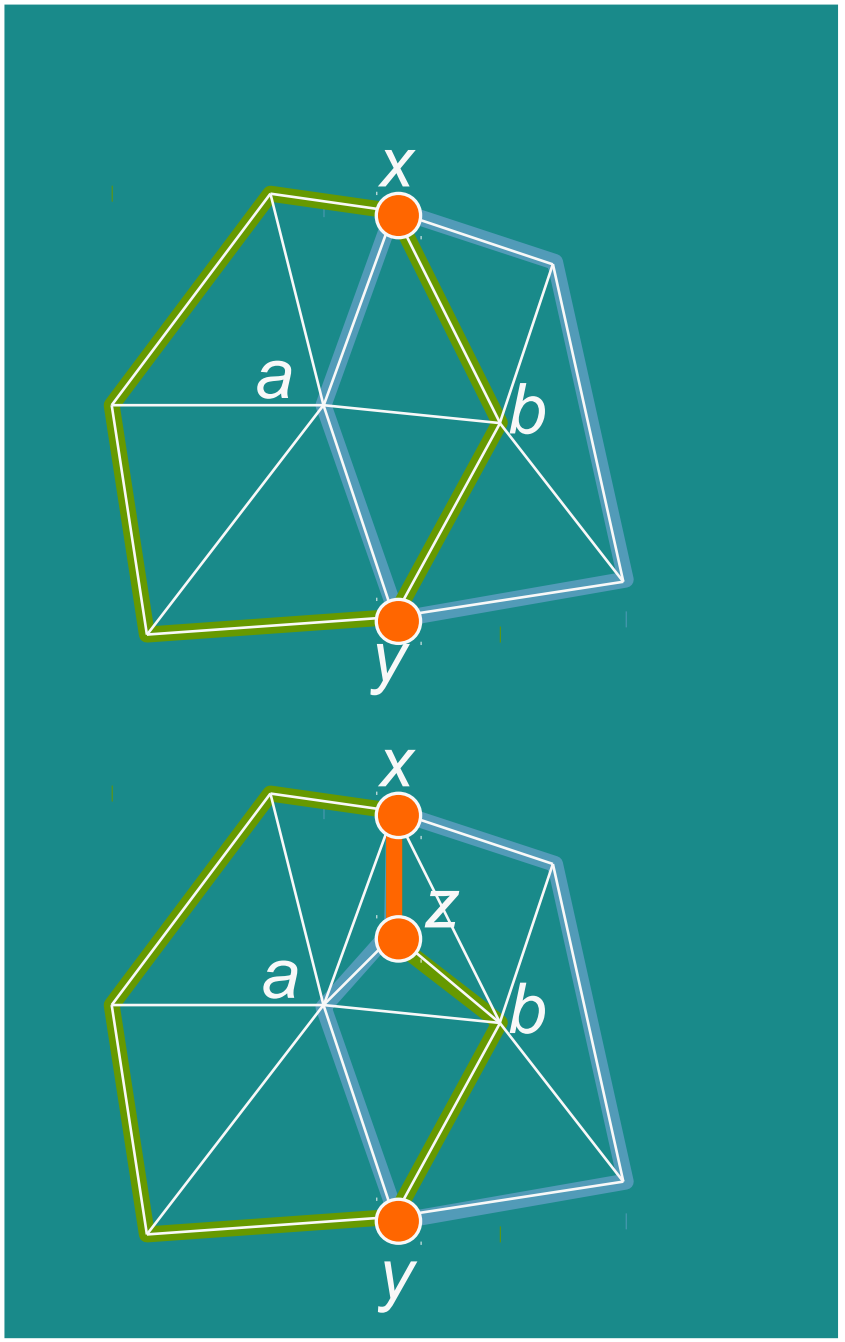
$Lk(a) \cap Lk(b) = Lk(ab)$

[Dey 99]

Topology Preservation

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

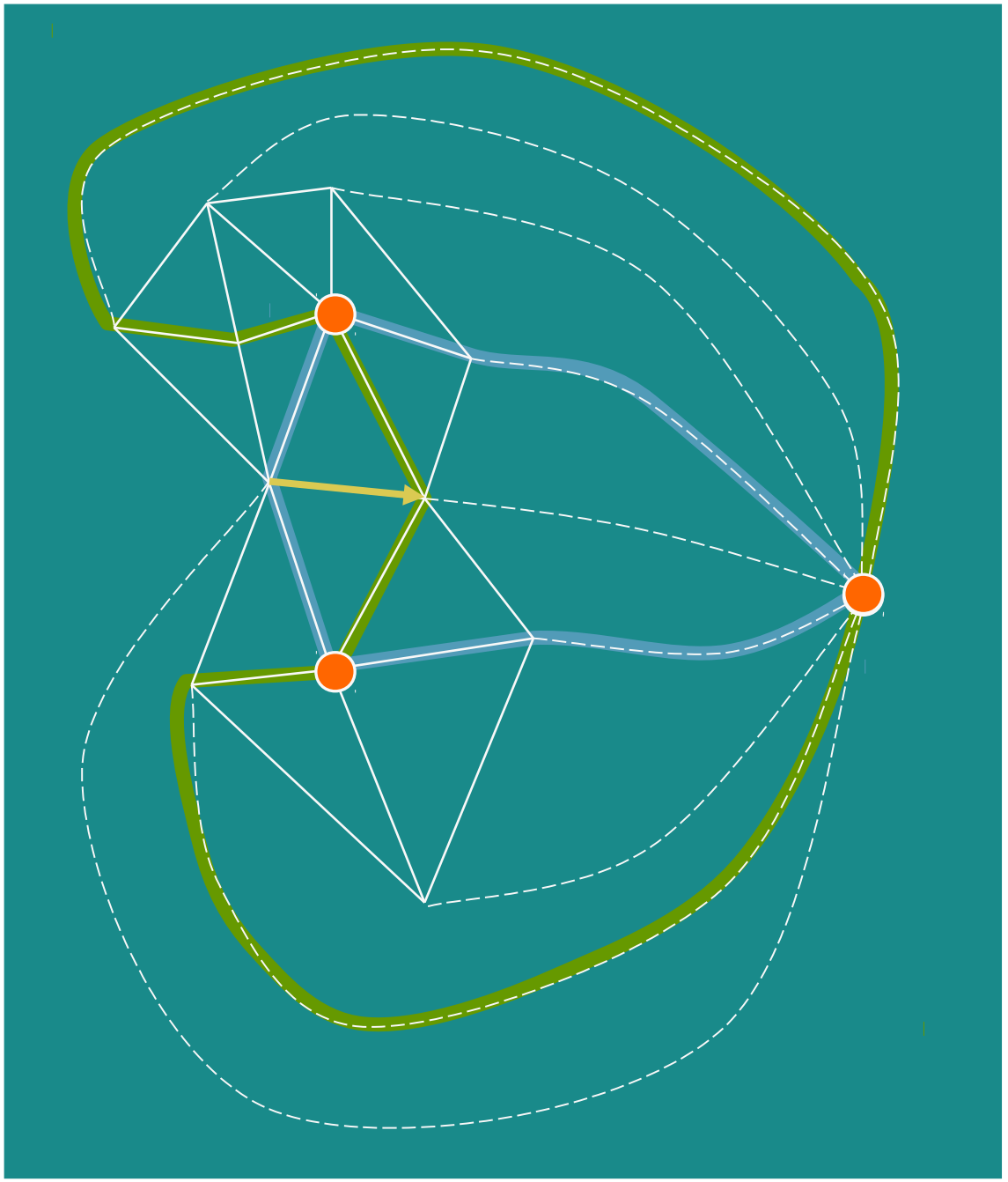
$Lk(a) \cap Lk(b) = \{x, y, z, zx\} \neq \{y, z\} = Lk(ab)$



Topology Preservation

Mesh with boundary can be managed by considering a dummy vertex v_d and, for each boundary edge *e*a dummy triangle connecting e with v_d

Think it wrapped on the surface of a sphere

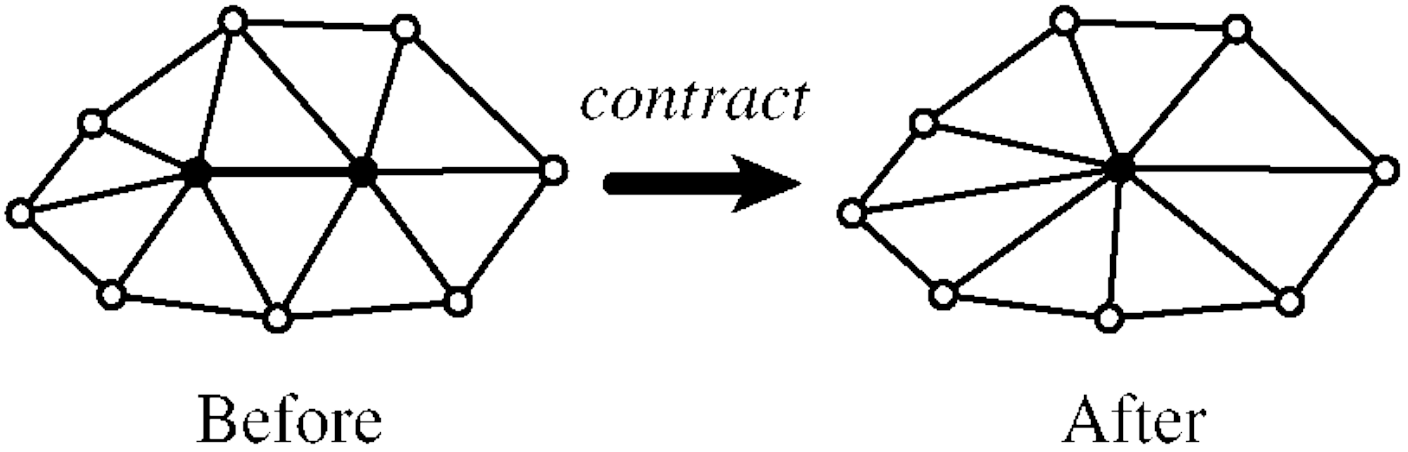


Lazy heap

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



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

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

Simplification Algorithms

Not-incremental methods:

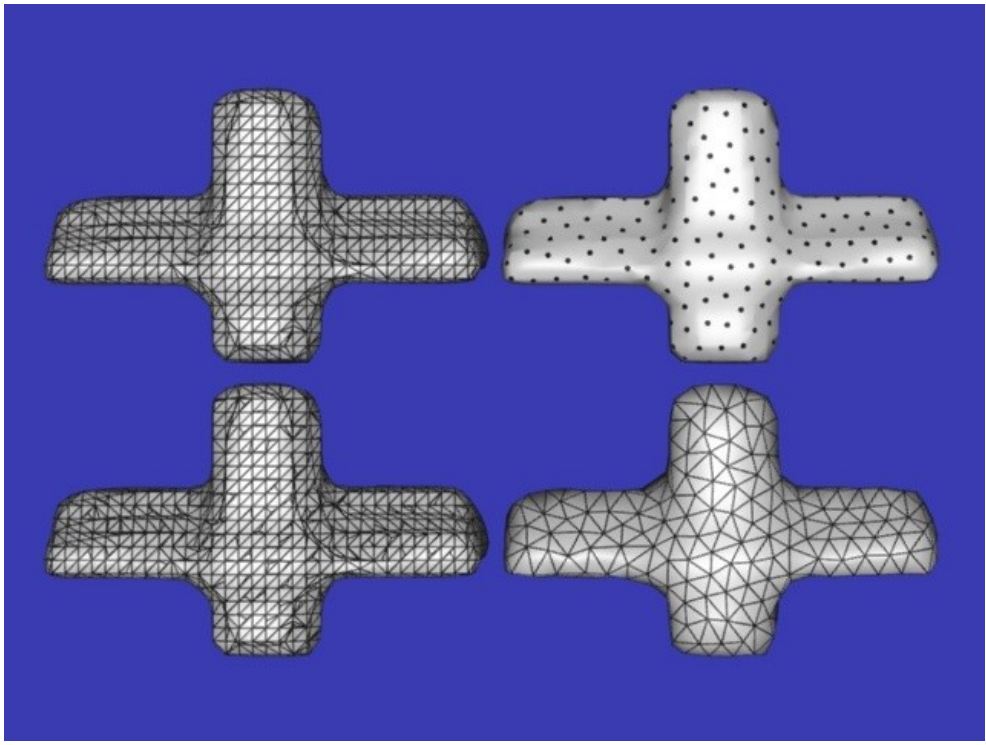
coplanar facets merging
[Hinker et al. `93, Kalvin et al. `96]

re-tiling
[Turk `92]

clustering
[Rossignac et al. `93, ... + others]

wavelet-based
[Eck et al. `95]

volume-based
[He+ 95, He+ 96, Andujar+ 02]



Clustering

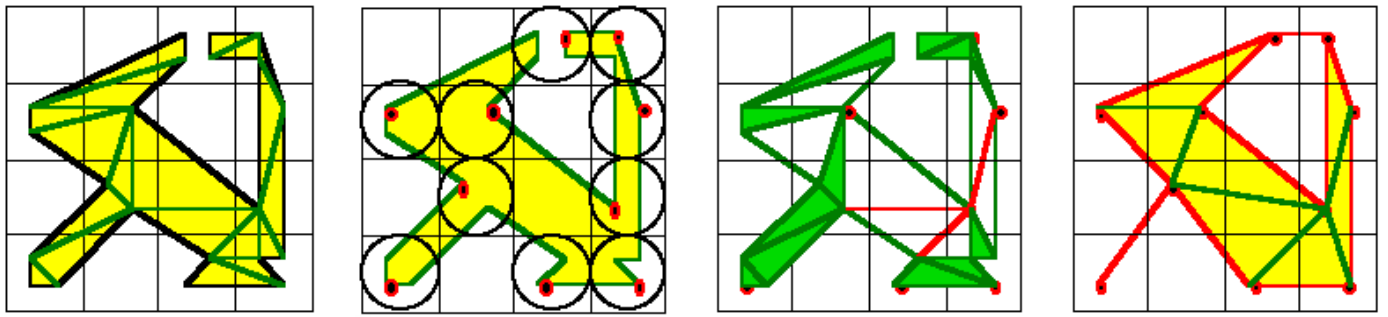
Vertex Clustering **[Rossignac, Borrel `93]**

detect and unify ***clusters*** of nearby vertices (discrete gridding and coordinates truncation)

all faces with two or three vertices in a cluster are removed

does not preserve topology (faces may degenerate to edges, genus may change)

approximation depends on grid resolution



(figure by Rossignac)

Clustering -- Examples

Simplification of a table lamp, Interaction Accelerator

IBM 3D

10,108 facets 1,383 facets 474 facets 46 facets

... Clustering...

Clustering - *Evaluation*

high efficiency

very simple implementation

low quality approximations

does not preserve topology

error is bounded by the grid cell size

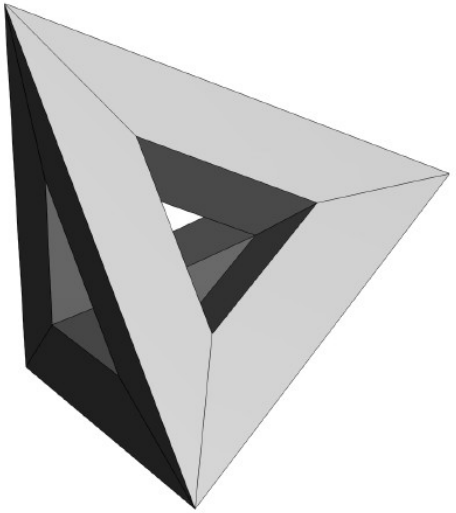
REFINEMENT
or
Subdivision Surfaces

Subdivision Surfaces

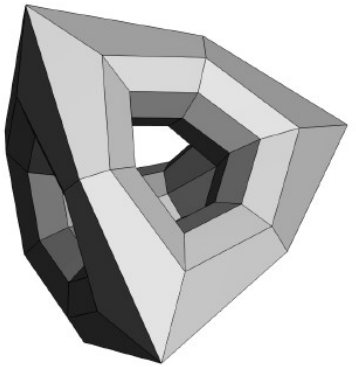
Subdivision defines a smooth curve or surface as the limit of a sequence of successive refinements.

- Si parte da una mesh poligonale
- Si suddivide i poligoni che la compongono
- Smooth della superficie muovendo i vertici

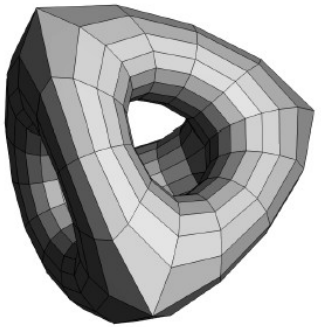
In effetti quello che si vede sono sempre approx delle vere subdiv surfaces



(a)



(b)



(c)

(d)

Esempio

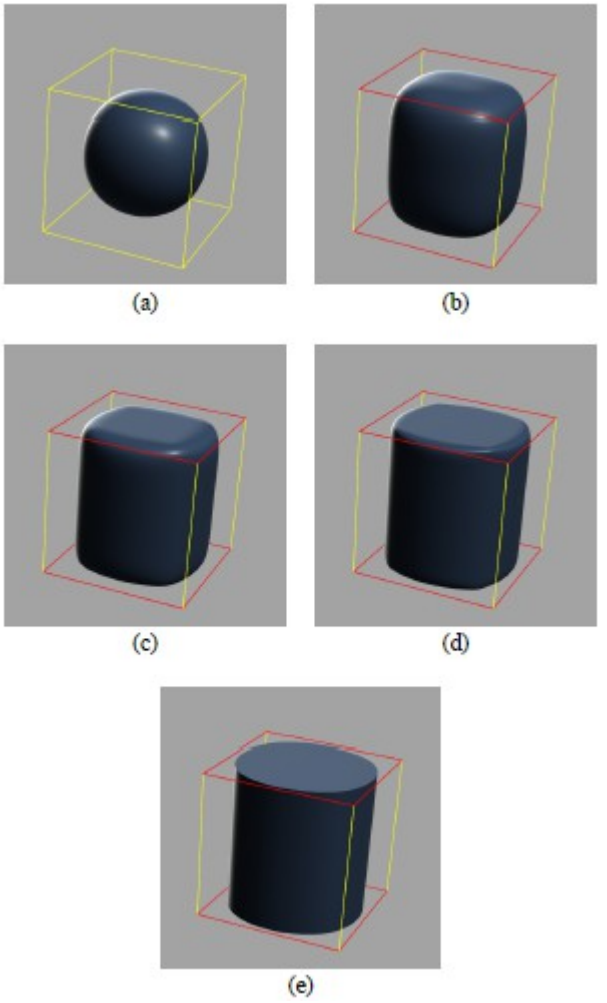
Geri's Game (1997)

Primo esempio non accademico di uso di subdivision surfaces

Smoothness

Non solo superfici smooth

Variable sharpness creases

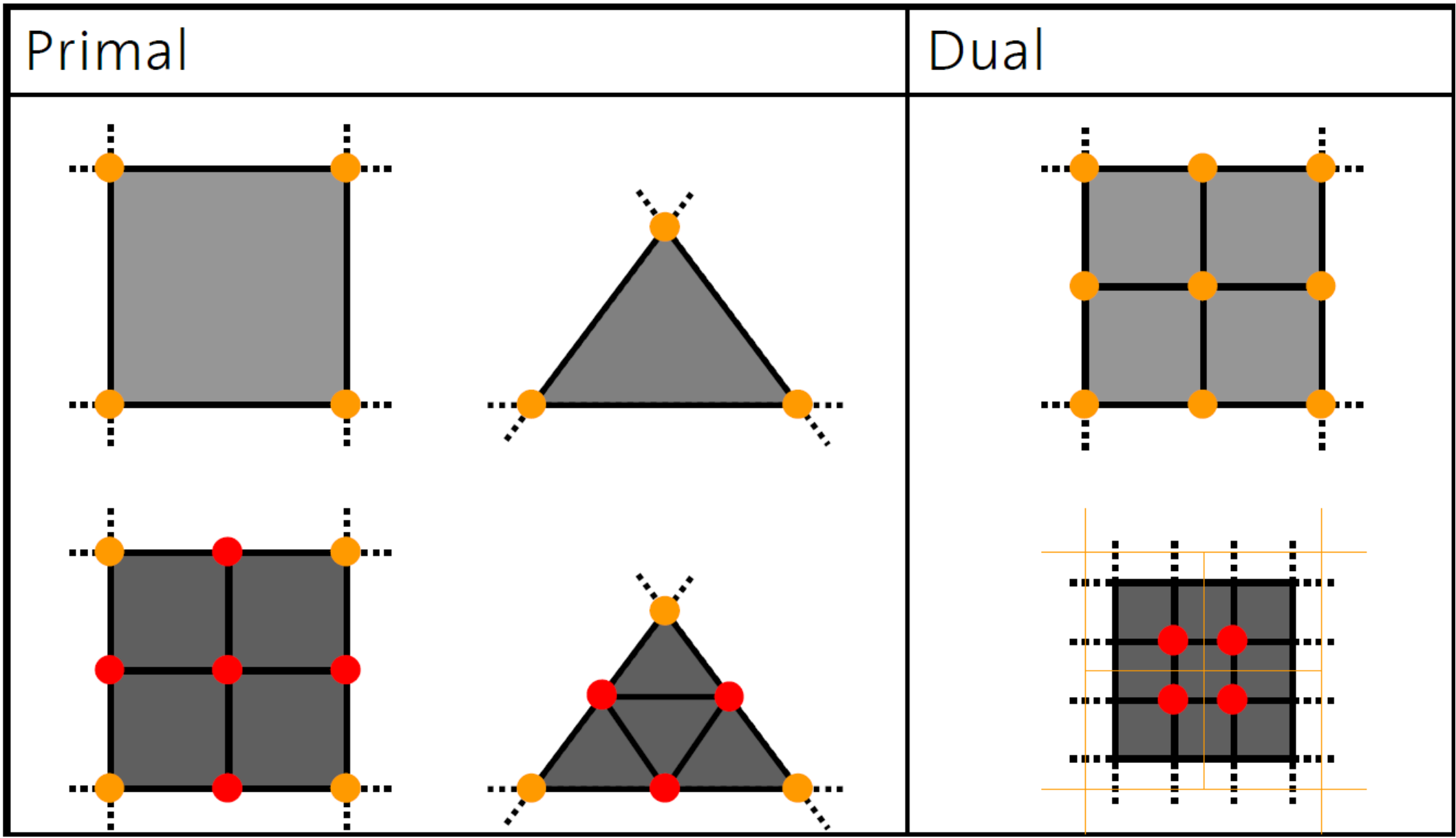


Subdivision Classification

Primal	Faces are split into sub-faces
Dual	Vertices are split into multiple vertices

Approximating	Control points not interpolated
Interpolating	Control points are interpolated

Subdivision Classification



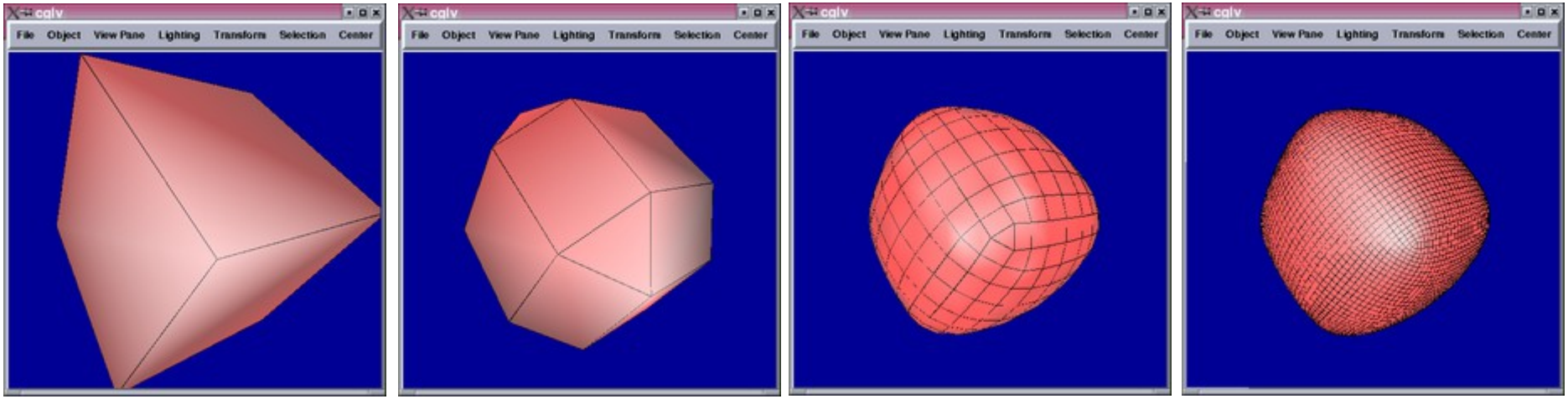
Subdivision

	Primal		*Dual*
	Triangles	Rectangles	
Approximating	Loop	Catmull-Clark	Doo-Sabin
Interpolating	Butterfly	Kobbelt	Midedge

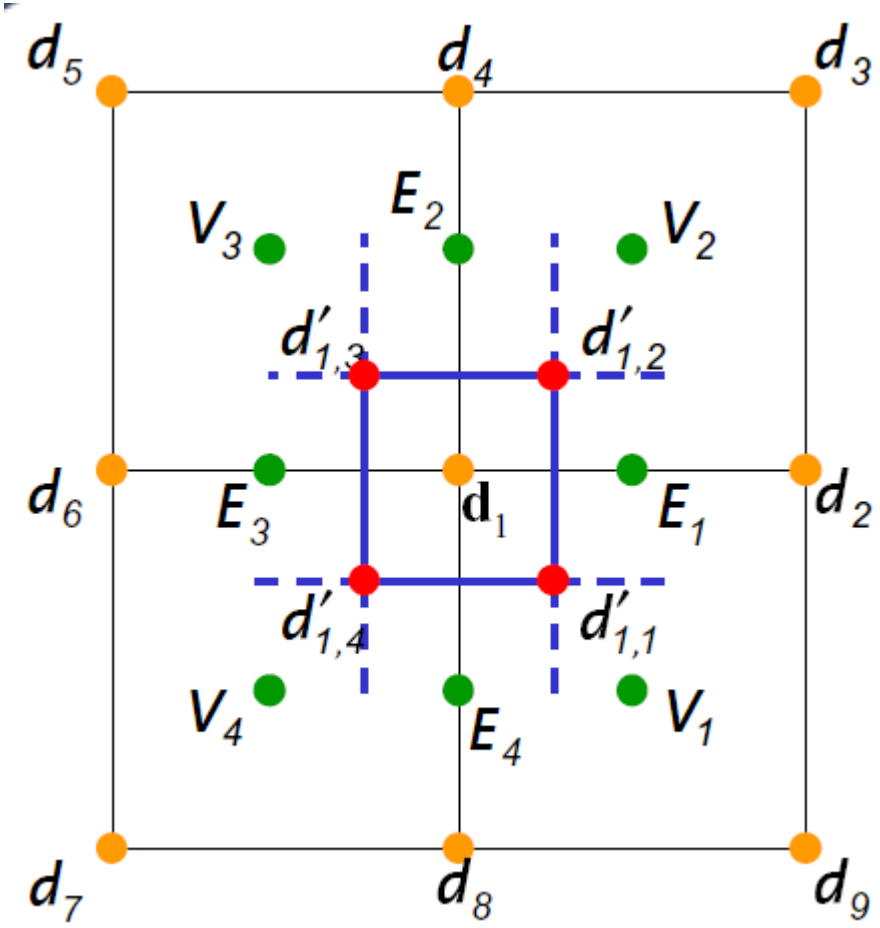
Doo Sabin

Per mesh poligonali

Duale ad ogni vertice corrisponde una nuova faccia



Doo Sabin



$$V_2 = \frac{1}{n} \cdot \sum_{j=1}^{n} d_J$$

$$E_i = \frac{1}{2}(d_1 + d_{2i})$$

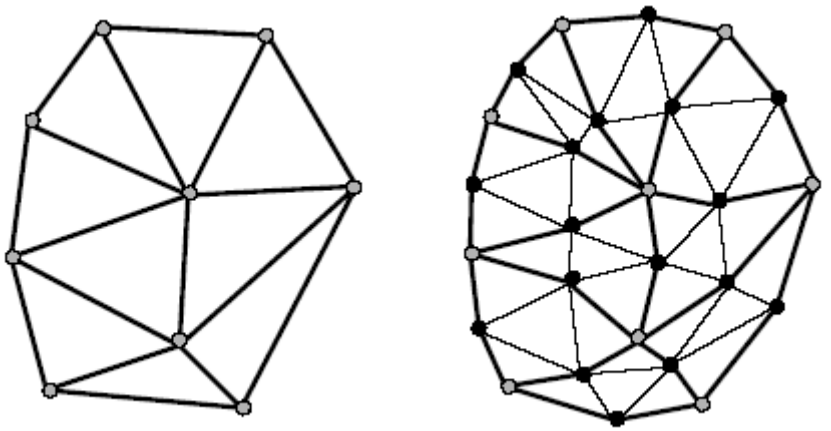
$$d'_{1,J} = \frac{1}{4}(d_1 + E_J + E_{J-1} + V_J)$$

Loop Refinement Scheme

Funziona per mesh triangolari

Vertex insertion

Ogni edge è diviso in due e i nuovi vertici sono riconnessi per formare nuovi triangoli



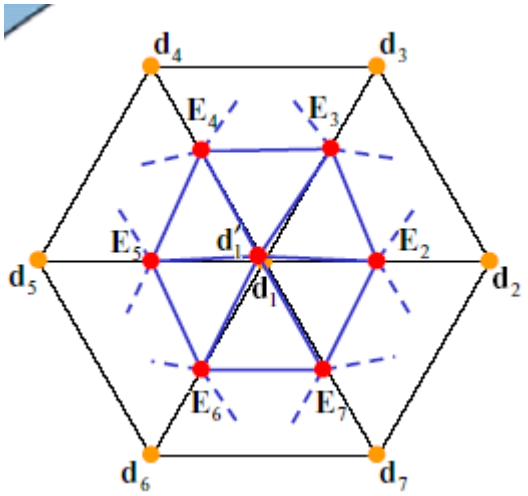
Loop Subdivison

Approssimante (non interpolante)

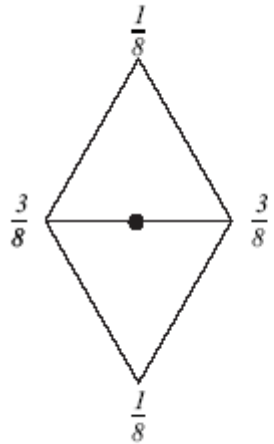
Continua

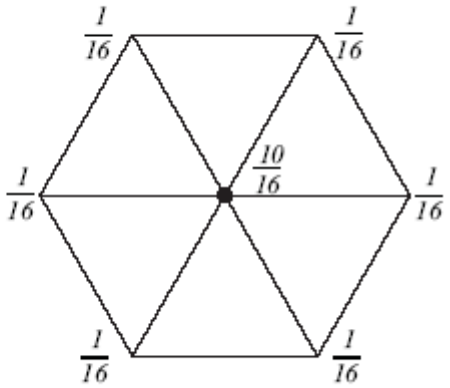
- C1 su vertici straordinari (valenza !=6)
- C2 elsewhere

Loop Subdivision



$$E_i = \frac{3}{8}(d_1 + d_i) + \frac{1}{8}(d_{i-1} + d_{i+1})$$





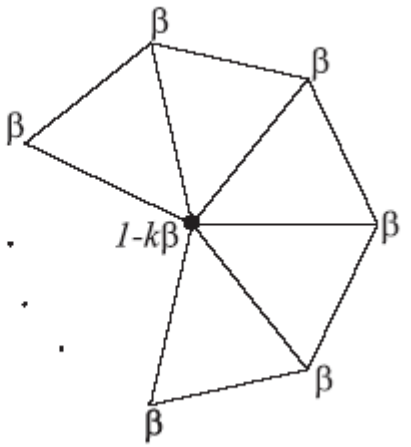
$$\mathbf{d}'_1 = \alpha_n \mathbf{d}_1 + \frac{(1-\alpha_n)}{n} \sum_{j=2}^{n+1} \mathbf{d}_j$$

$$\alpha_n = \frac{3}{8} + \left(\frac{3}{8} + \frac{1}{4} \cos \frac{2\pi}{n} \right)^2$$

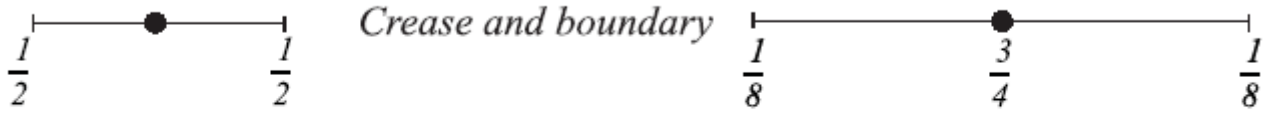
Loop Subdivision

La scelta di B non è unica

$$\beta = \frac{1}{k}\left(5/8 - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k} \right)^2 \right)$$

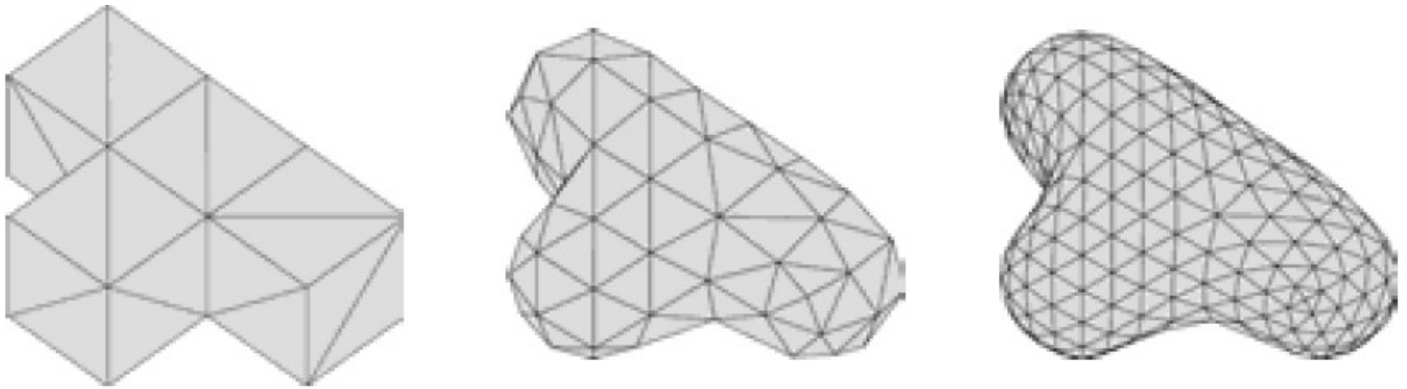


Loop Border



a. *Masks for odd vertices*

b. *Masks for even vertices*



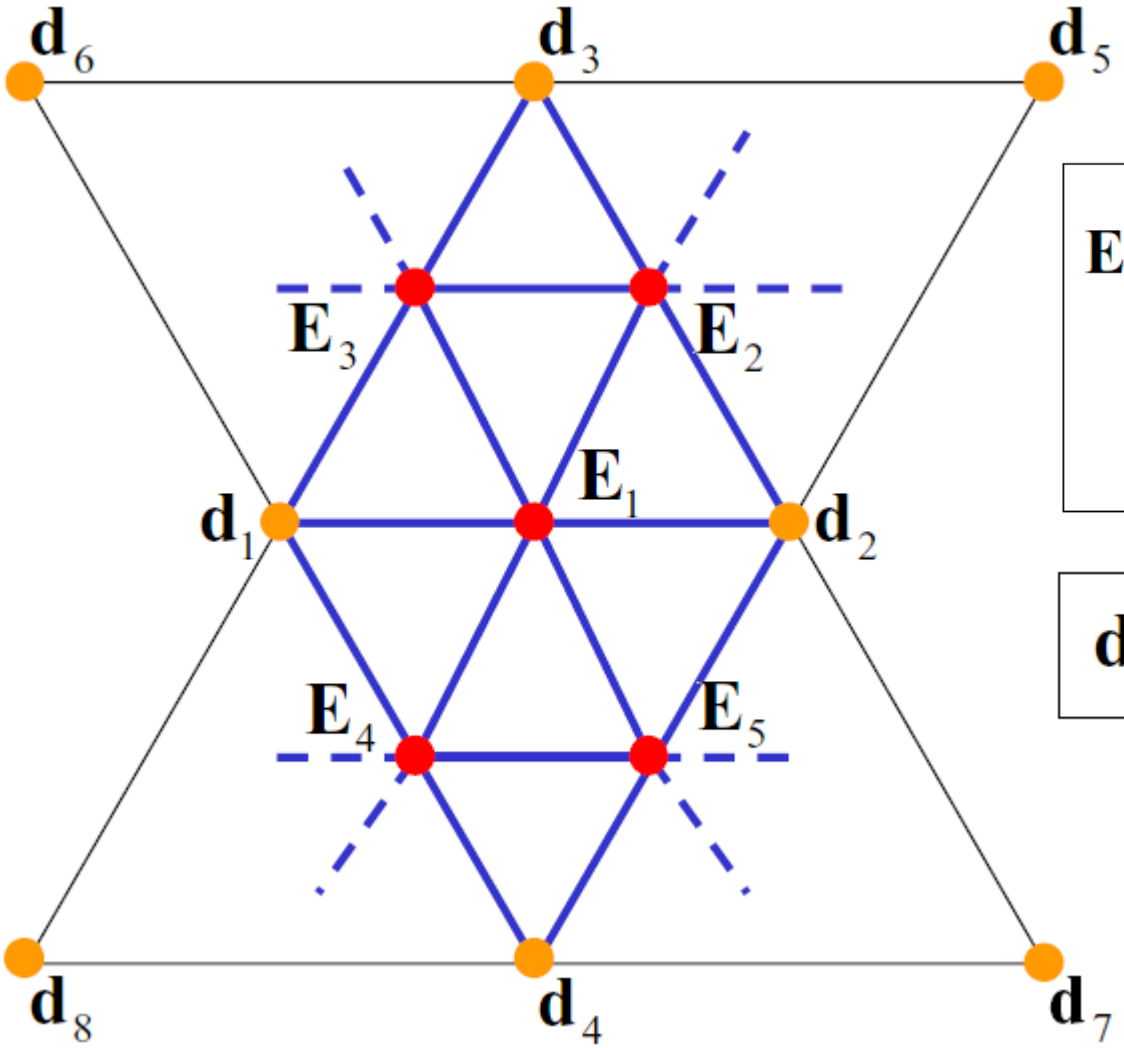
Butterfly

Interpolante (non approssimante)

Continua

- C0 su vertici straordinari (valenza <4 e >7)
- C1 elsewhere

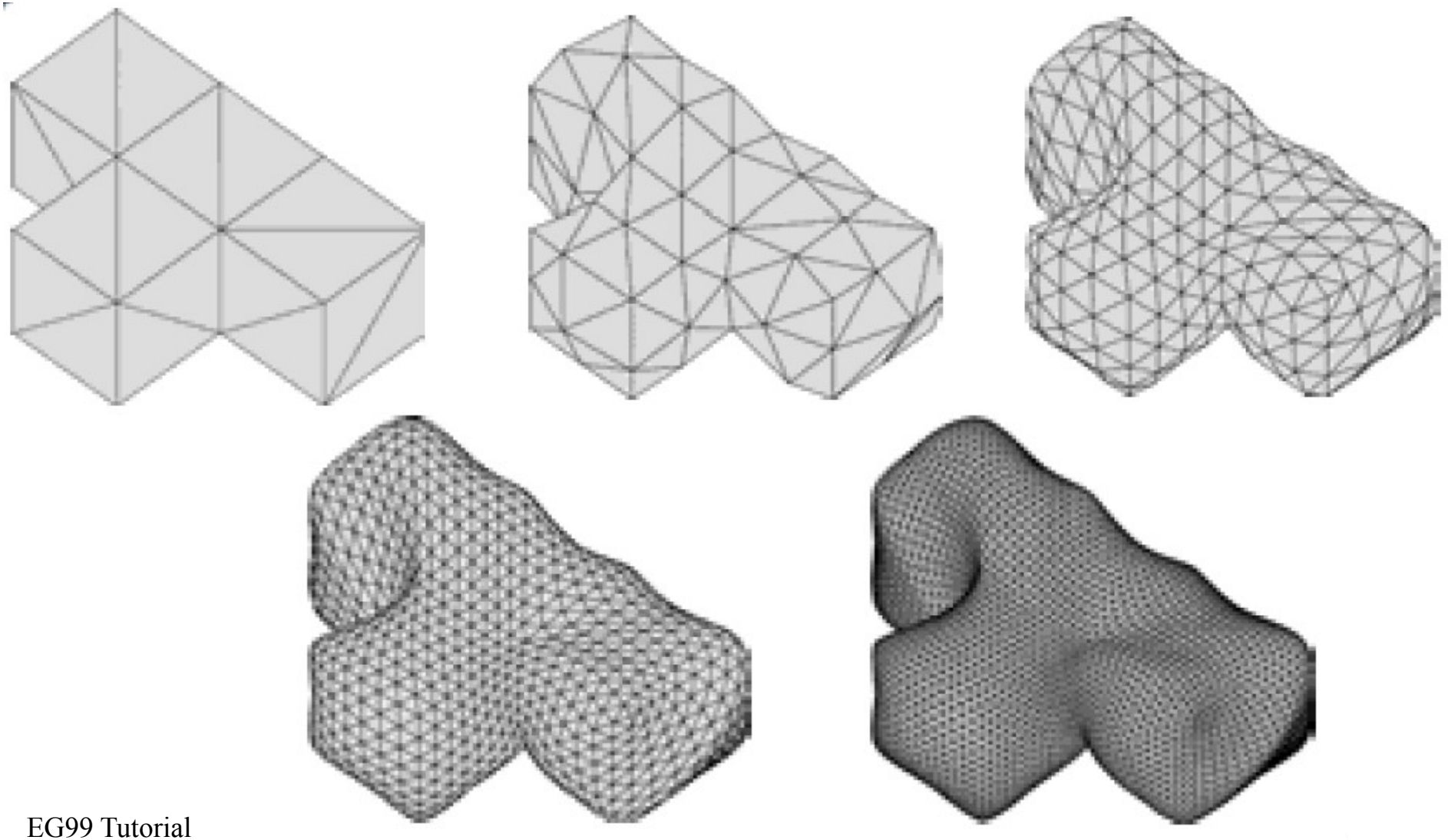
Butterfly



$$\mathbf{E}_1 = \frac{1}{2}(\mathbf{d}_1 + \mathbf{d}_2) + \omega(\mathbf{d}_3 + \mathbf{d}_4) - \frac{\omega}{2}(\mathbf{d}_5 + \mathbf{d}_6 + \mathbf{d}_7 + \mathbf{d}_8)$$

$$\mathbf{d}'_i = \mathbf{d}_i$$

Butterfly Subdivision



Simplification and Refinement

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

Approximating surfaces with triangle meshes

Managing the quantity of geometry vs the quality of the representation

Assumptions:

Surfaces represented by triangle meshes

Accuracy of the approximation is proportional to the number of triangles

Complexity of a surface is proportional to the number of triangles

Objective:

Always produce the simplest mesh that satisfies the accuracy required by the application

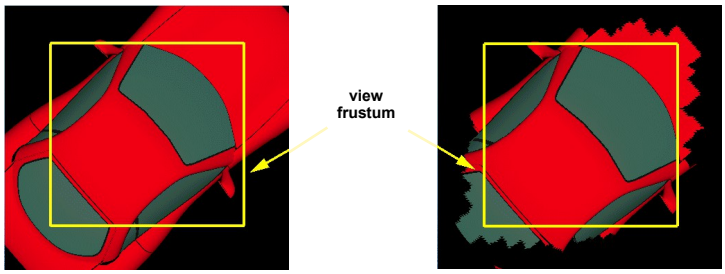
Reducing rendering cost

Simplifying is not the only way to reduce the needed number of triangles
At rendering time we can do:

View Frustum Culling

Drawing only what is in the view frustum

…



culling OFF culling ON

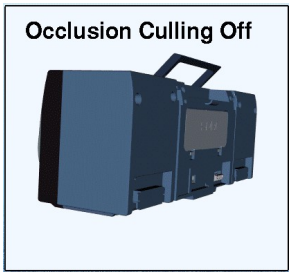
Reducing Complexity

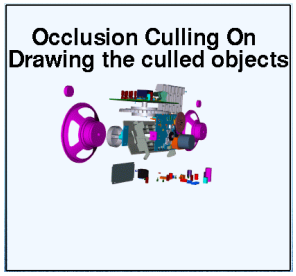
…

Occlusion/Visibility culling

Drawing only what it is probably visible

➔ ***Improve performances, but not sufficient for very large datasets***





Simplification

From a computational Point of view, fixed the size of the solution find the ***best*** representation

The optimal solution in its general form is NP Hard

Heuristics works well

(even with bounds)

Metrics

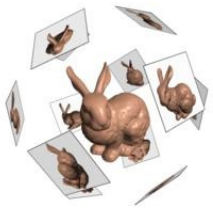
Computing the “difference” between two meshes

Geometric way

- Hausdorff

Perceptual

- Working in Image space
  - Many renderings, computing the difference, possibly in a human-like way



Appearance similarity

Difference between two images: (trivial)

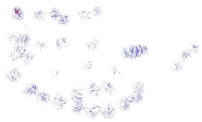
$$D(I_1, I_2) = \frac{1}{n^2} \sum_x \sum_y d(I_1(x, y), I_2(x, y))$$

Difference between two objects:

Integrate the above over all possible views

I_1

I_2



$I_1 - I_2$

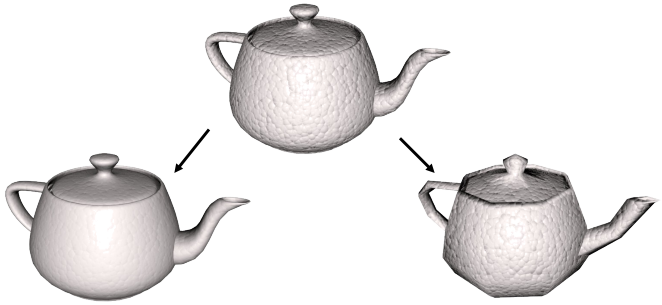
Approximation error

Quantifies the notion of “similarity”

Two kinds of similarity:

Geometric similarity (surface deviation)

Appearance similarity (material, normal…)

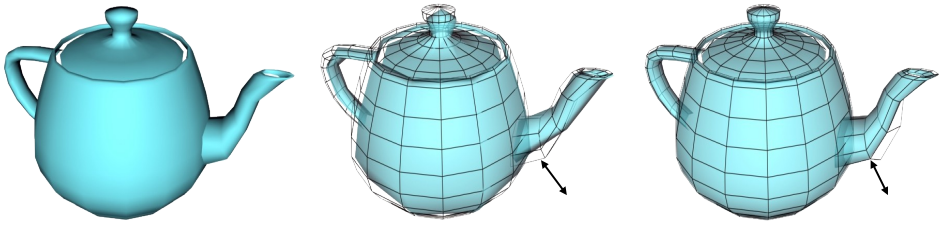


The primary goal of simplification is **to generate an approximation which resembles as much as possible** the original.

Geometric similarity

Two main components:

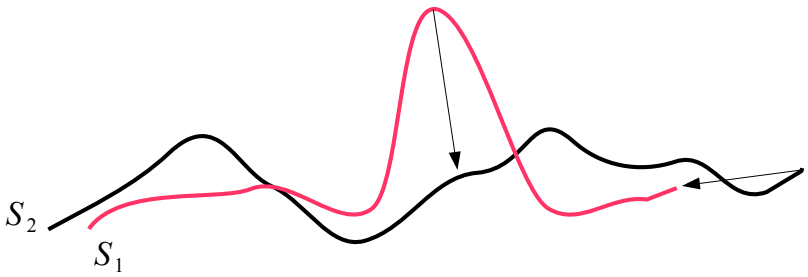
- Distance function
- Function Norm:
  - L_2: average deviation
  - L_{inf}: maximum deviation - Hausdorff distance



[Image by C. Andujar]

Hausdorff Distance

$$D_H(S_1, S_2) = \max_{x \in S_1} \left(\min_{y \in S_2} D(x, y) \right)$$



Symmetric version:

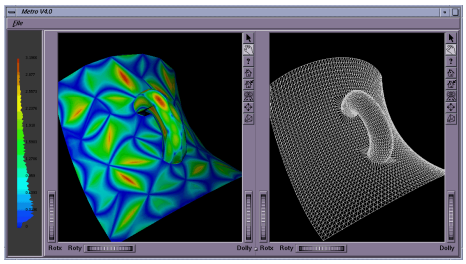
$$D(S_1, S_2) = \max \{ D_H(S_1, S_2), D_H(S_2, S_1) \}$$

Computing Hausdorf Distance

In a approximate way

Sample a surface

- Points uniformly distributed over it
- For each point x compute $\min_{y \in S_2} D(x, y)$
  - points usually very near to surface,
  - UG works well, better than trees
- Avg are useful too...



Simplification Algorithms

Simplification approaches:

incremental methods based on local updates

mesh decimation

[Schroeder et al. `92, ... + others]

energy function optimization & progressive meshes

[Hoppe et al. `93, Hoppe '96, Hoppe '97]

quadric error metrics [Garland et al. '97]

coplanar facets merging

[Hinker et al. `93, Kalvin et al. `96]

clustering

[Rossignac et al. `93, ... + others]

wavelet-based

[Eck et al. `95, + others]

volume-based

[He+ 95, He+ 96, Andujar+ 02]

Incremental methods based on ***local updates***

All of the methods such that :

simplification proceeds as a sequence of small changes of the mesh (in a greedy way)

each update reduces mesh size and [~monotonically] decreases the approximation precision

Different approaches:

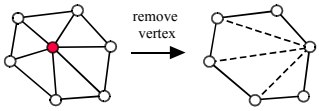
mesh decimation
energy function optimization
progressive meshes
quadric error metrics

… Incremental methods based on ***local updates ...***

Local update actions:

No. Faces

vertex removal

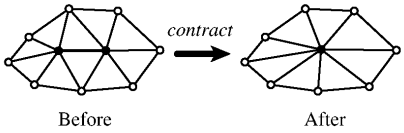


n-2

edge collapse

preserve location

new location



n-2

Before After

Incremental Simp Method

The common greedy framework:

While *mesh size/precision is satisfactory*

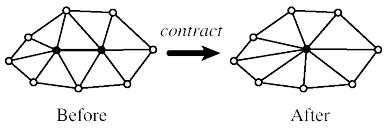
Select from a priority queue the best element to simplify

Perform simplification modifying the mesh

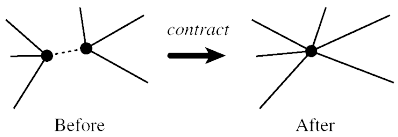
Ppdate priority queue according to the mods;

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)



Quadric Error for Surfaces

Let $\mathbf{n}^T\mathbf{v} + d = 0$ be the equation representing a plane

The squared distance of a point $\boldsymbol{x}$ from the plane is

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

This distance can be represented as a quadric

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

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

Quadric

The error is estimated by providing for each 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).

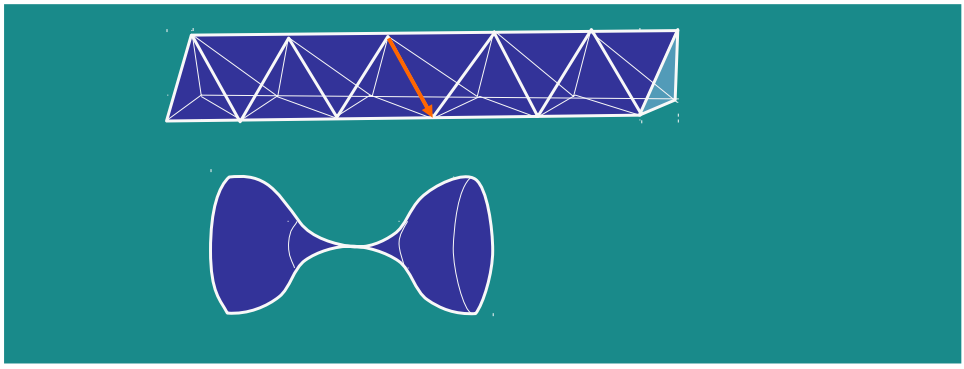
After the collapse the quadric of v is updated as follow $Q_v = Q_v + Q_w$

Topology Preservation

2-Manifold

A surface Σ in $\boldsymbol{R^2}$ such that any point on Σ has an open neighborhood homeomorphic to an open disc or to half an open disc in $\boldsymbol{R^2}$

A edge collapse can create non manifold situations

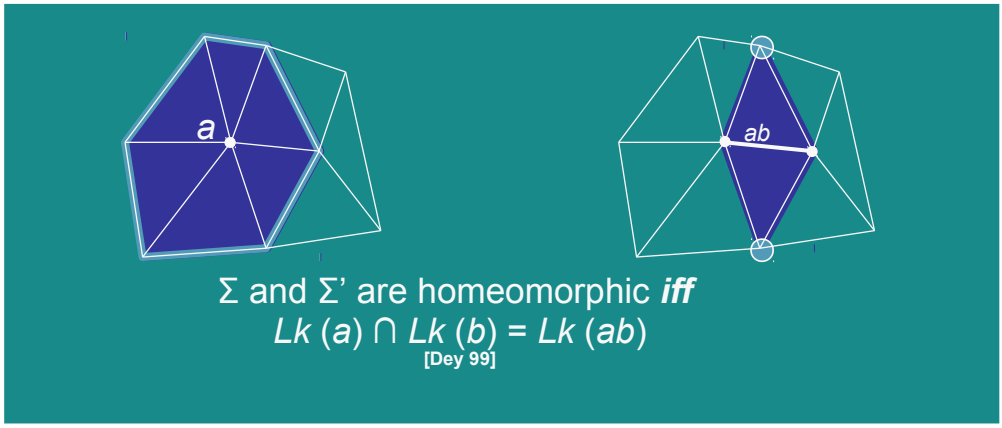


Topology Preservation

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



Σ and Σ' are homeomorphic ***iff***

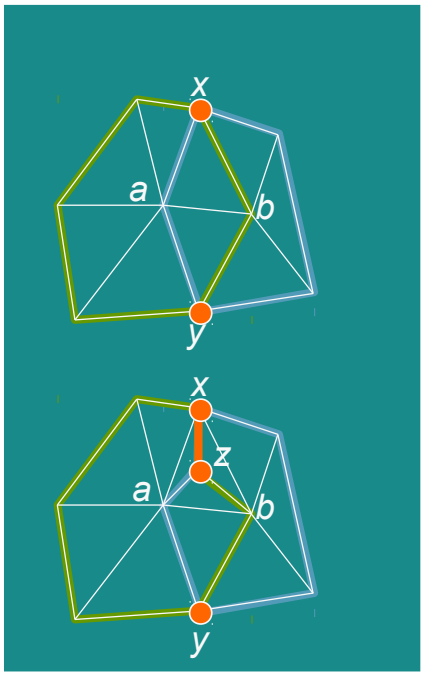
$Lk(a) \cap Lk(b) = Lk(ab)$

[Dey 99]

Topology Preservation

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

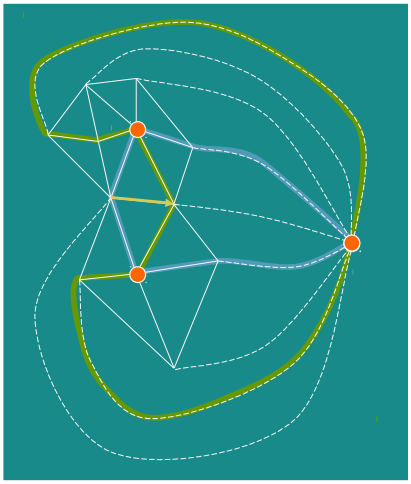
$Lk(a) \cap Lk(b) = \{x,y,z,zx\} \neq \{y,z\} = Lk(ab)$



Topology Preservation

Mesh with boundary can be managed by considering a dummy vertex v_d and, for each boundary edge *e*a dummy triangle connecting e with v_d

Think it wrapped on the surface of a sphere

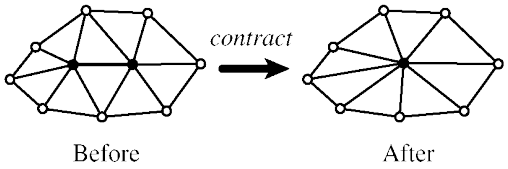


Lazy heap

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



Before After

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

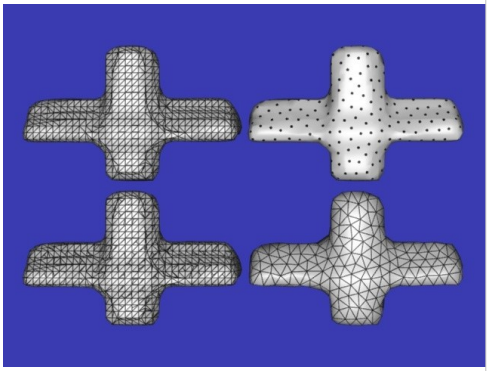
Validità collasso

- 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

Simplification Algorithms

Not-incremental methods:

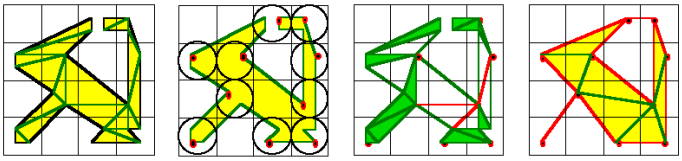
- coplanar facets merging
  [Hinker et al. `93, Kalvin et al. `96]
- re-tiling
  [Turk `92]
- clustering
  [Rossignac et al. `93, ... + others]
- wavelet-based
  [Eck et al. `95]
- volume-based
  [He+ 95, He+ 96, Andujar+ 02]



Clustering

Vertex Clustering **[Rossignac, Borrel `93]**

- detect and unify ***clusters*** of nearby vertices (discrete gridding and coordinates truncation)
- all faces with two or three vertices in a cluster are removed
- does not preserve topology (faces may degenerate to edges, genus may change)
- approximation depends on grid resolution



(figure by Rossignac)

Clustering -- Examples

Simplification of a table lamp, IBM 3D Interaction Accelerator

10,108 facets 1,383 facets 474 facets 46 facets

... Clustering...

Clustering - *Evaluation*

high efficiency

very simple implementation

low quality approximations

does not preserve topology

error is bounded by the grid cell size

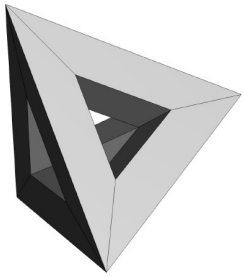
REFINEMENT
or
Subdivision Surfaces

Subdivision Surfaces

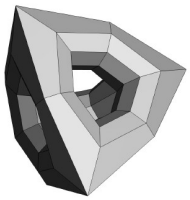
Subdivision defines a smooth curve or surface as the limit of a sequence of successive refinements.

- Si parte da una mesh poligonale
- Si suddivide i poligoni che la compongono
- Smooth della superficie muovendo i vertici

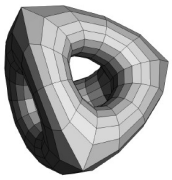
In effetti quello che si vede sono sempre approx delle vere subdiv surfaces



(a)



(b)



(c)

(d)

Esempio

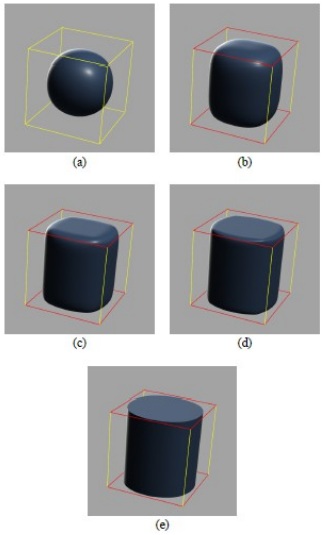
Geri's Game (1997)

Primo esempio non accademico di uso di subdivision surfaces

Smoothness

Non solo superfici smooth

Variable sharpness creases

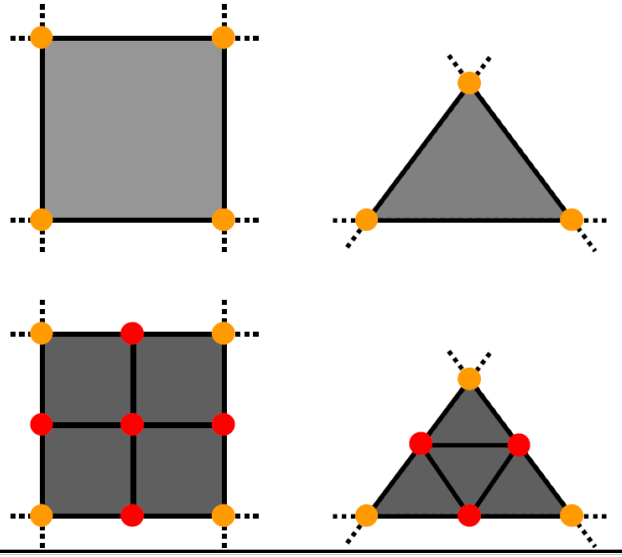
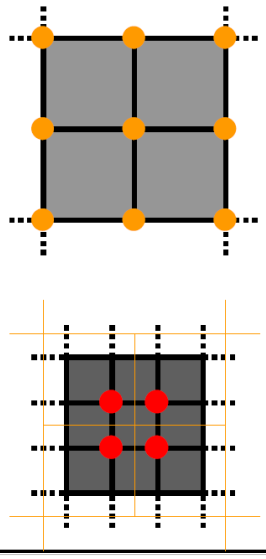


Subdivision Classification

Primal	Faces are split into sub-faces
Dual	Vertices are split into multiple vertices

Approximating	Control points not interpolated
Interpolating	Control points are interpolated

Subdivision Classification

Primal	Dual
	

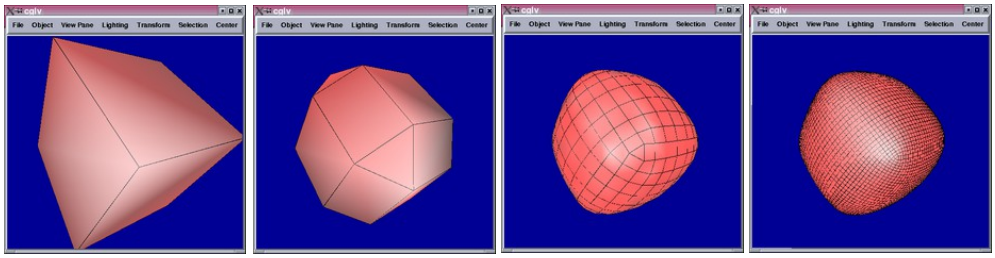
Subdivision

	Primal		*Dual*
	Triangles	Rectangles	
Approximating	Loop	Catmull-Clark	Doo-Sabin Midedge
Interpolating	Butterfly	Kobbelt	

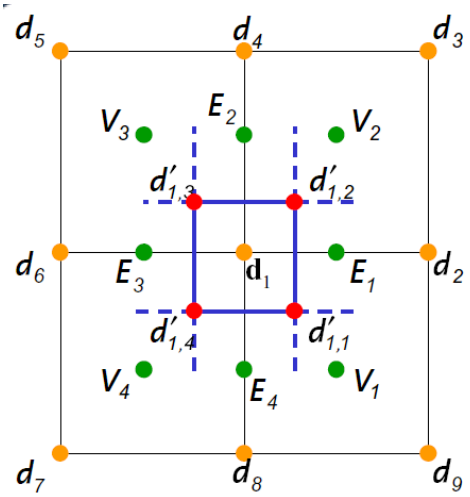
Doo Sabin

Per mesh poligonali

Duale ad ogni vertice corrisponde una nuova faccia



Doo Sabin



$$V_2 = \frac{1}{n} \cdot \sum_{j=1}^{n} d_J$$

$$E_i = \frac{1}{2}(d_1 + d_{2i})$$

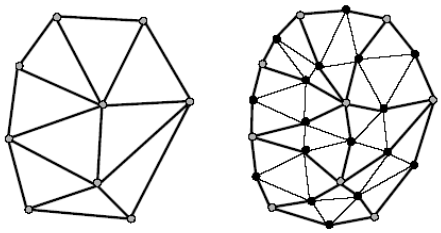
$$d'_{1,J} = \frac{1}{4}(d_1 + E_J + E_{J-1} + V_J)$$

Loop Refinement Scheme

Funziona per mesh triangolari

Vertex insertion

- Ogni edge è diviso in due e i nuovi vertici sono riconnessi per formare nuovi triangoli



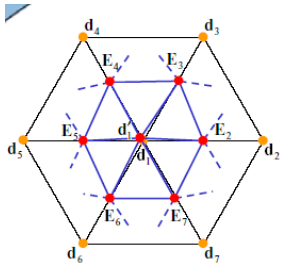
Loop Subdivison

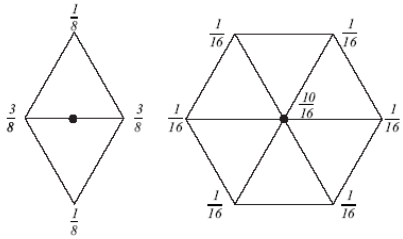
Approssimante (non interpolante)

Continua

- C1 su vertici straordinari (valenza !=6)
- C2 elsewhere

Loop Subdivision





$$E_i = \frac{3}{8}(d_1 + d_i) + \frac{1}{8}(d_{i-1} + d_{i+1})$$

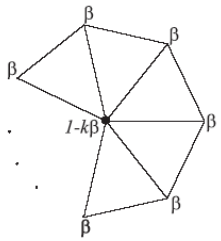
$$\mathbf{d}'_1 = \alpha_n \mathbf{d}_1 + \frac{(1-\alpha_n)}{n}\sum_{j=2}^{n+1} \mathbf{d}_j$$

$$\alpha_n = \frac{3}{8} + \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2$$

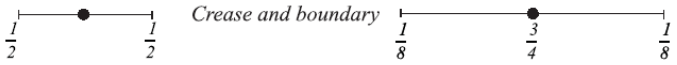
Loop Subdivision

La scelta di B non è unica

$$\beta = \frac{1}{k}\left(5/8 - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k} \right)^2 \right)$$

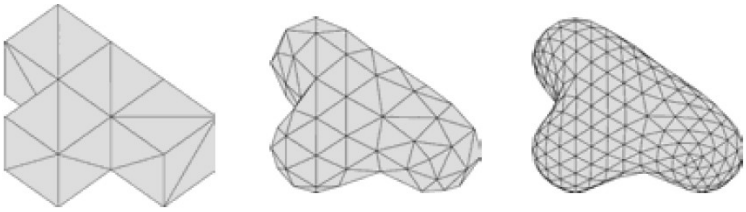


Loop Border



a. Masks for odd vertices

b. Masks for even vertices



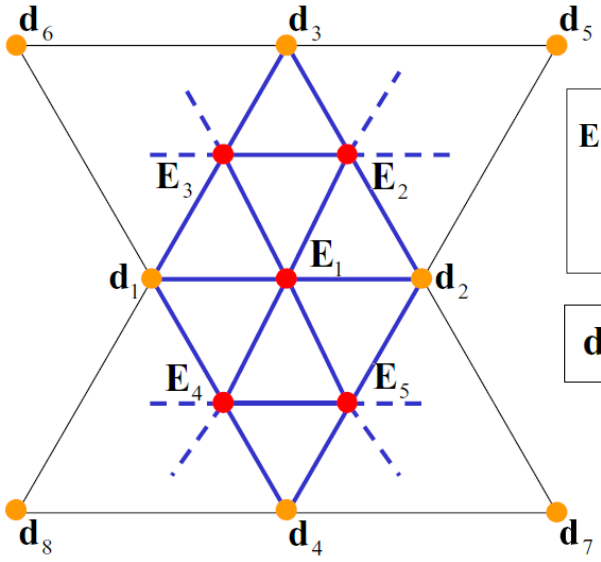
Butterfly

Interpolante (non approssimante)

Continua

- C0 su vertici straordinari (valenza <4 e >7)
- C1 elsewhere

Butterfly



$$\mathbf{E}_1 = \frac{1}{2}(\mathbf{d}_1 + \mathbf{d}_2) + \omega(\mathbf{d}_3 + \mathbf{d}_4) - \\ -\frac{\omega}{2}(\mathbf{d}_5 + \mathbf{d}_6 + \mathbf{d}_7 + \mathbf{d}_8)$$

$$\mathbf{d}_i' = \mathbf{d}_i$$

Butterfly Subdivision