Geometric Modeling: Digital Representation and Analysis of Shapes

Lecture 4: Mesh representations and applications

(october 16th)

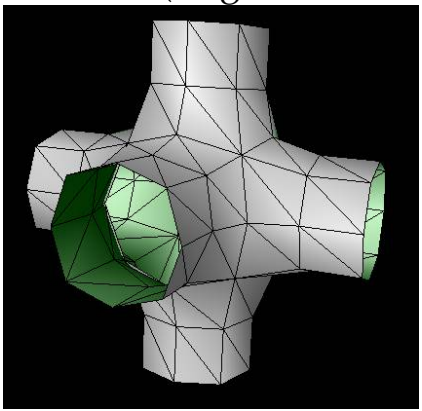
Luca Castelli Aleardi

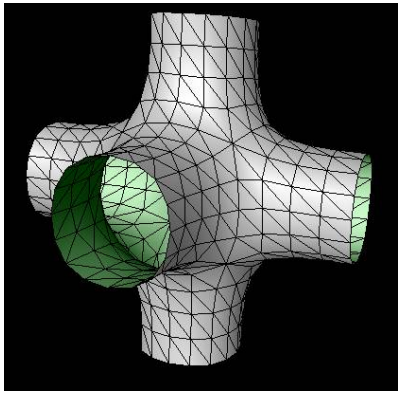
Intro
3D meshes: application domains

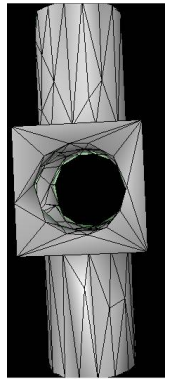
Mesh representations: applications

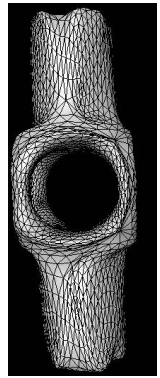
geometric modeling

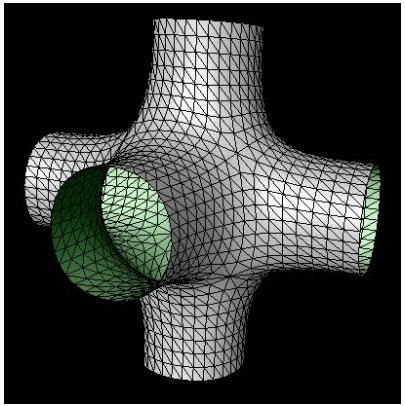
32 vertices (original model)

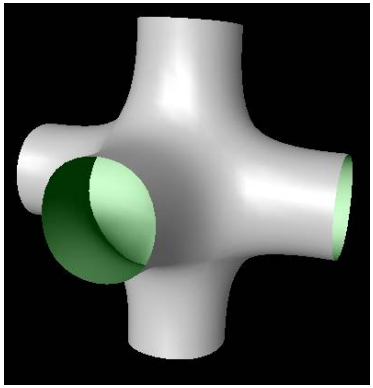






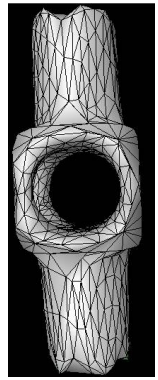


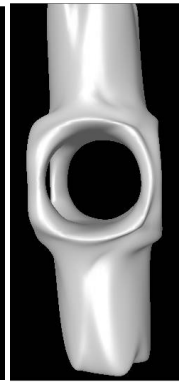




1628 vertices

3 iterations of
Loop subdivision scheme

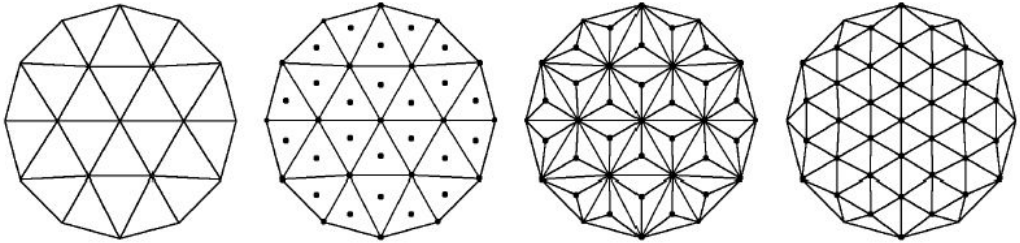


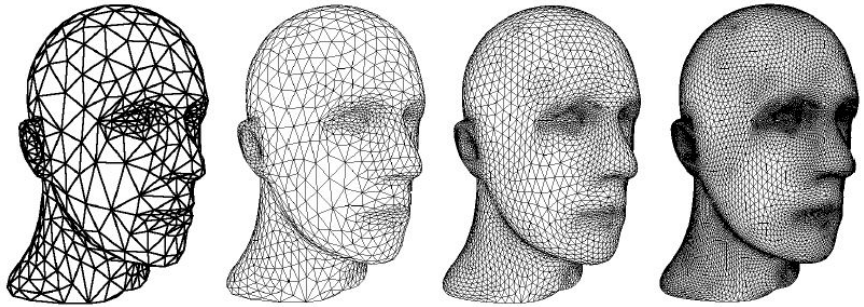


Mesh representations: applications

Subdivision surfaces

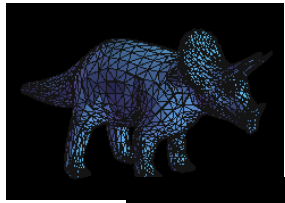
$\sqrt{3}$-subdivision (Kobbelt, SIGGRAPH 2000)

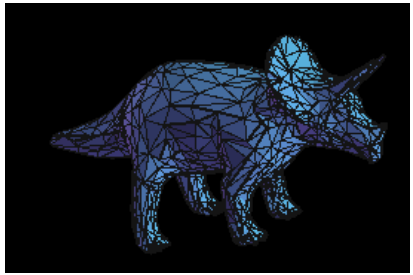


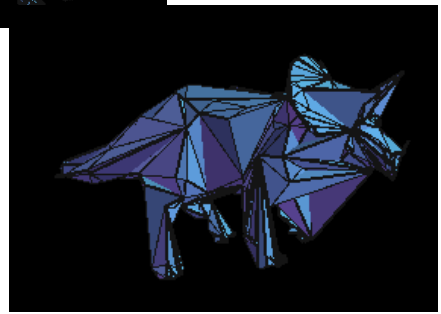


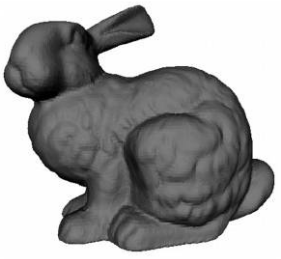
Mesh representations: simplification (3D paper sculpture)

DT Workshop







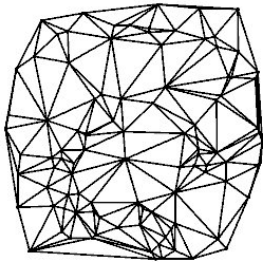


69451 faces

1000 faces

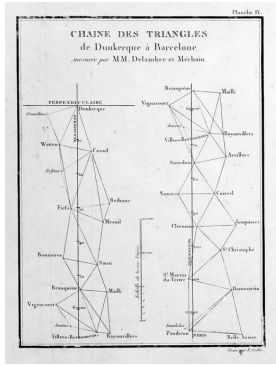
Mesh representations and computational geometry

Delaunay triangulations, Voronoi diagrams, planar meshes, ...

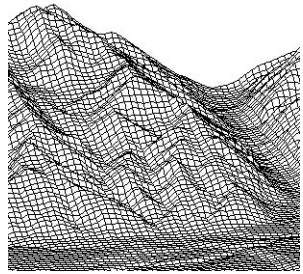


Delaunay triangulation

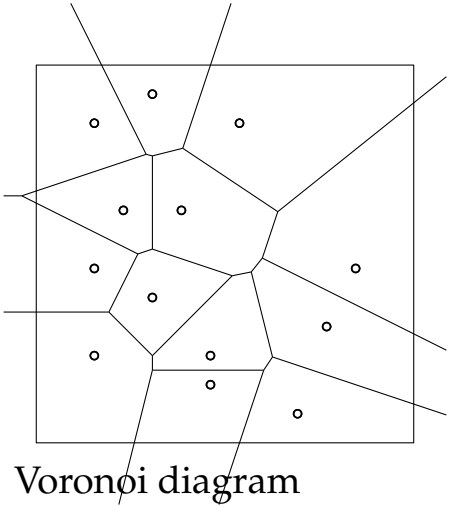
triangles meshes already used in early 19th century (Delambre et Méchain)



GIS Technology

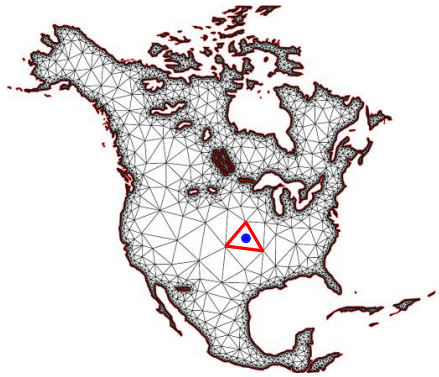


Terrain modelling



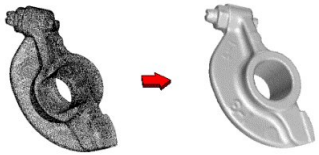
Voronoi diagram

Planar mesh by L. Rineau, M. Yvinec



Mesh representations: computational geometry

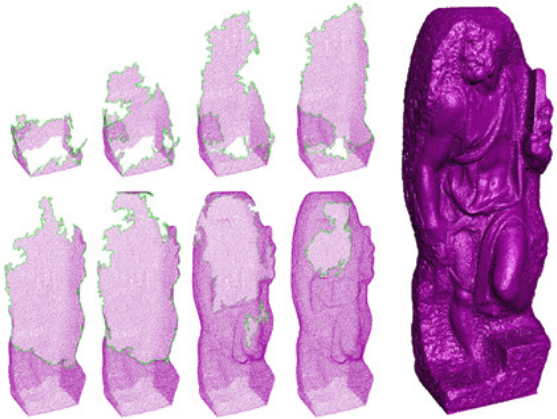
surface reconstruction from sampling



St. Matthew (Stanford's Digital Michelangelo Project, 2000)

David statue (Stanford's Digital Michelangelo Project, 2000)

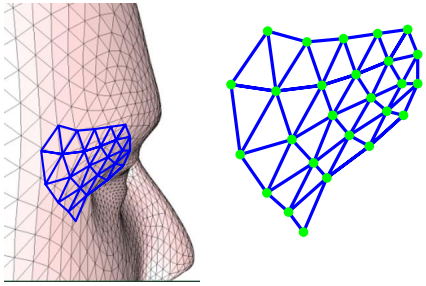
2 billions polygons
32 Giga bytes (without compression)



186 millions vertices
6 Giga bytes (for storing on disk)
several minutes for loading the model from disk

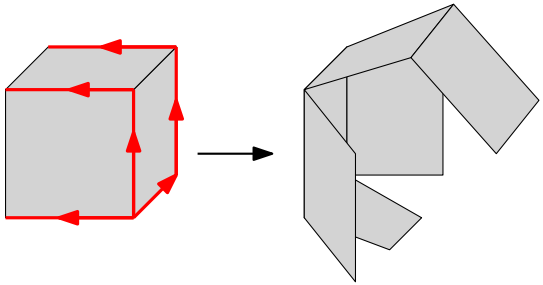
Part I
3D meshes: definition and properties

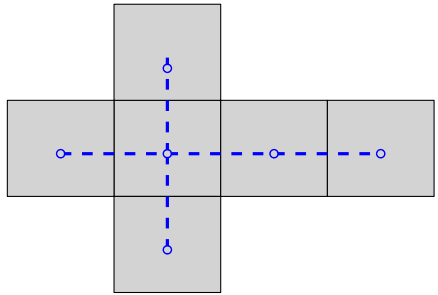
What is a (surface) mesh?



surface mesh: set of vertices, edges and faces (polygons) defining a polyhedral surface in embedded in 3D (discrete approximation of a shape)

Combinatorial structure + geometric embedding



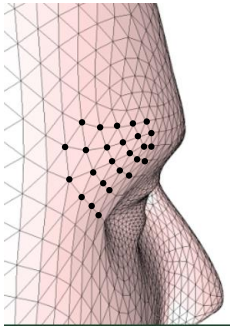


What kind of object does represent a mesh?

Geometric structure vs Combinatorial structure

Connectivity is by far the most expensive information

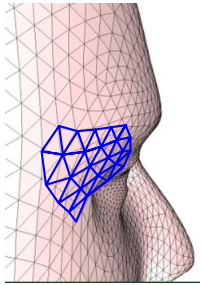
Geometry

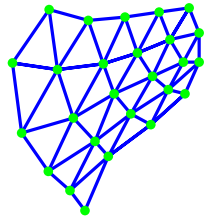


vertex coordinates

between 30 et 96 bits/vertex

”Connectivity”: the underlying triangulation





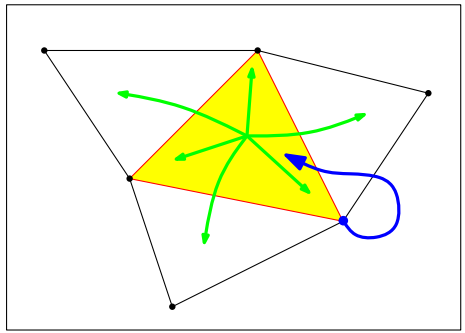
incidence relations between triangles, vertices and edges

vertex 1 reference to a triangle

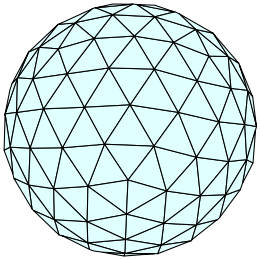
triangle 3 references to vertices
3 references to triangles

$13n \log n$ or $416n$ bits

1 reference: pointer or integer (32 bits)

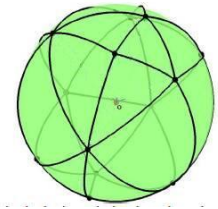


Planar and surface meshes: definition

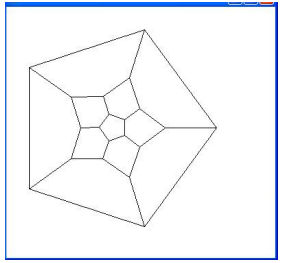


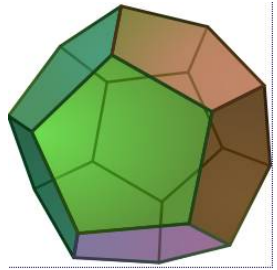
planar triangulation
embedded in R^3

triangle mesh

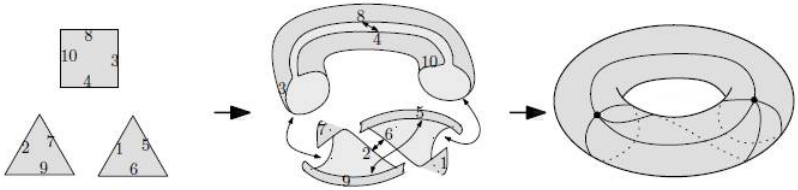


planar triangulation
spherical drawing



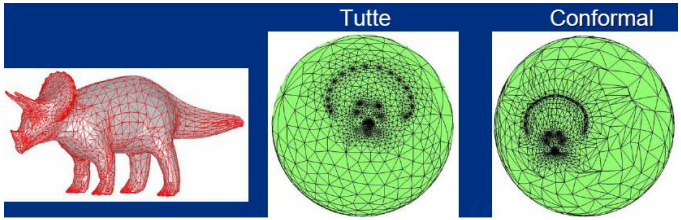


planar map
straight line drawing of a dodecahedron



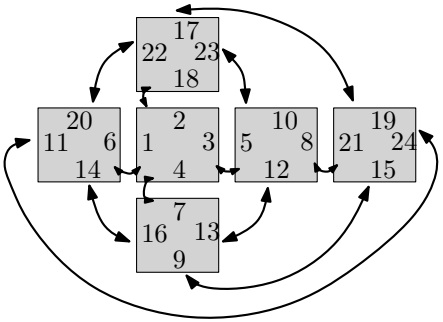
toroidal map (Eric Colin de Verdière)

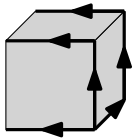
spherical parameterizations of a
triangle mesh (Gotsman, Gu Sheffer, 2003)



Surface meshes: "definition"

geomettric realization of a *simple (combinatorial) map*





3 permutations on the set H of the $2n$ half-edges

(i) α involution without fixed point;

(ii) $\alpha\sigma\phi = Id$;

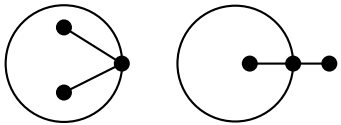
(iii) the goup generated by σ, α et ϕ transitively on H.

$\phi = (1, 2, 3, 4)(17, 23, 18, 22)(5, 10, 8, 12)(21, 19, 24, 15)\ldots$

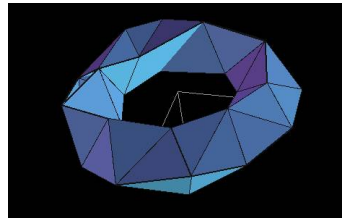
$\alpha = (2, 18)(4, 7)(12, 13)(9, 15)(14, 16)(10, 23)\ldots$

A *graph* $G = (V, E)$ is a pair of:

- a set of *vertices* $V = (v_1, \ldots, v_n)$
- a collection of $E = (e_1, \ldots, e_m)$ elements of the cartesian product $V \times V = \{(u, v) \mid u \in V, v \in V\}$ (*edges*).

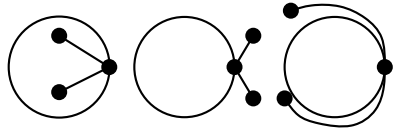


two different embeddings of the same graph



no loops, no multiple edges

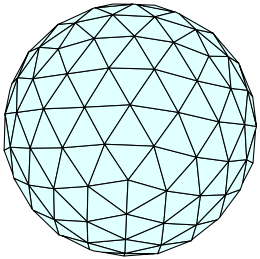
edges are line segments



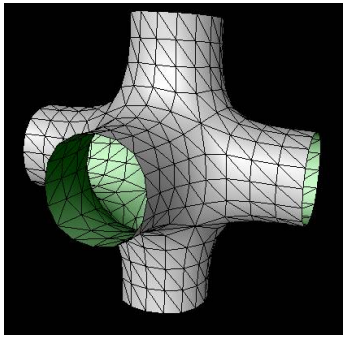
cellular embeddings of a graph defining the same (planar) map

Mesh representations: classification

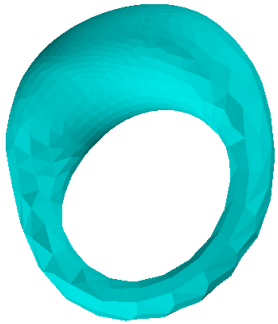
Manifold meshes



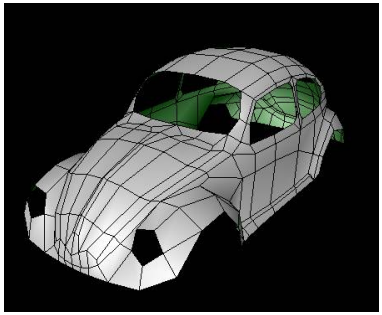
triangle meshes
no boundary



with boundaries



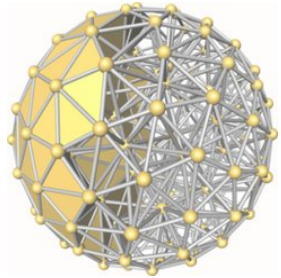
genus 1 mesh



quad meshes

polygonal meshes

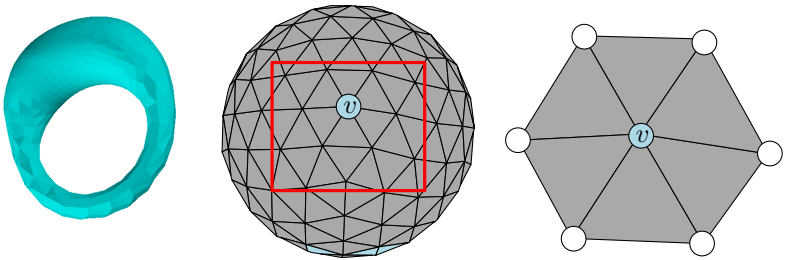
non manifold or non orientable meshes



volume (tetrahedral) meshes

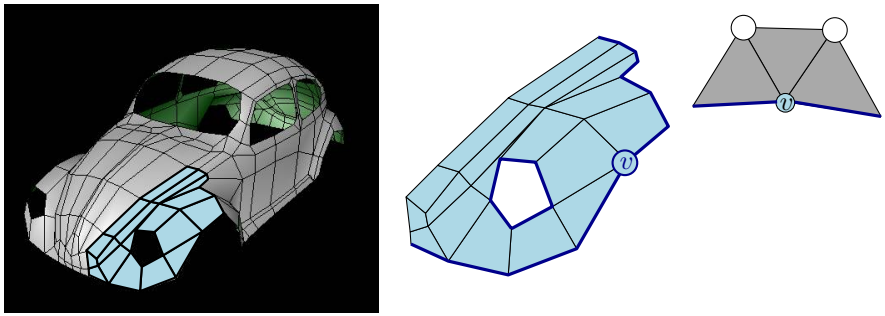
Manifold meshes

manifold mesh without boundary



Every edge is shared by exactly 2 faces

with boundaries



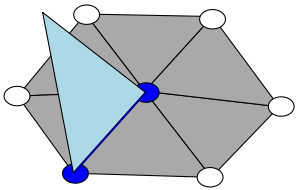
boundary edges are incident to 1 face

Manifold mesh: definition

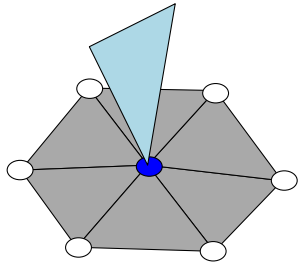
Every edge is shared by at most 2 faces

For every vertex v, the incident faces form an open or closed *fan*

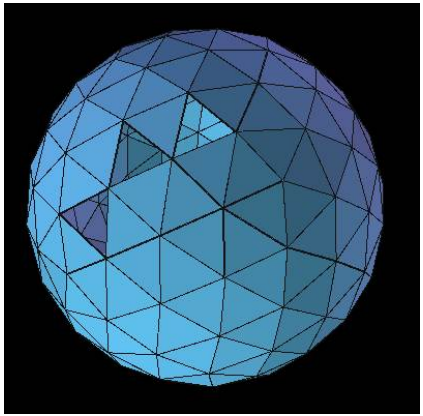
Non manifold meshes

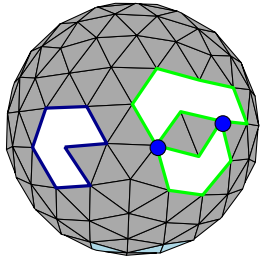


The edge is shared by more than 2 faces



vertex v has a neighbourhood that is not homeomorphic to a topological disk



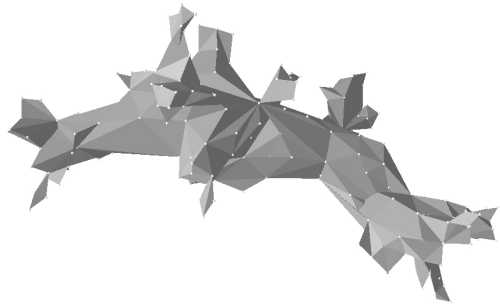


Manifold mesh: definition

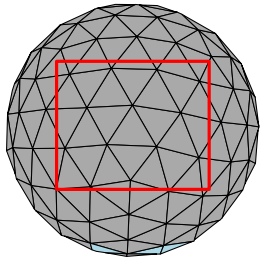
Every edge is shared by at most 2 faces

For every vertex v, the incident faces form an open or closed *fan*

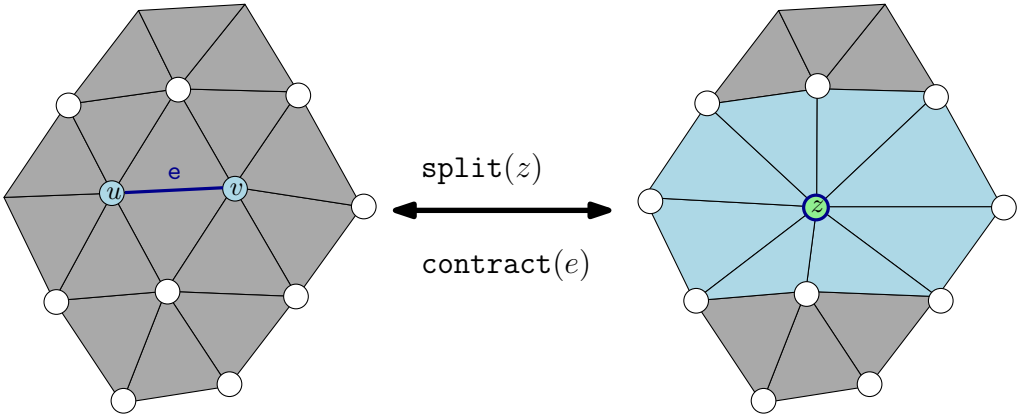
Random discrete surface (used in statistical physics)



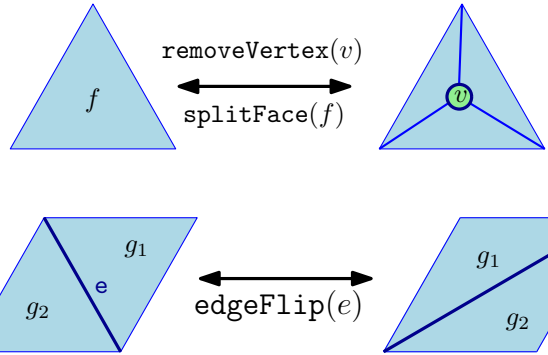
"Topological changes" (for triangle meshes)



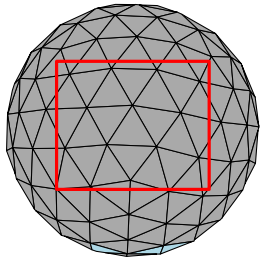
edge contraction and *vertex split*



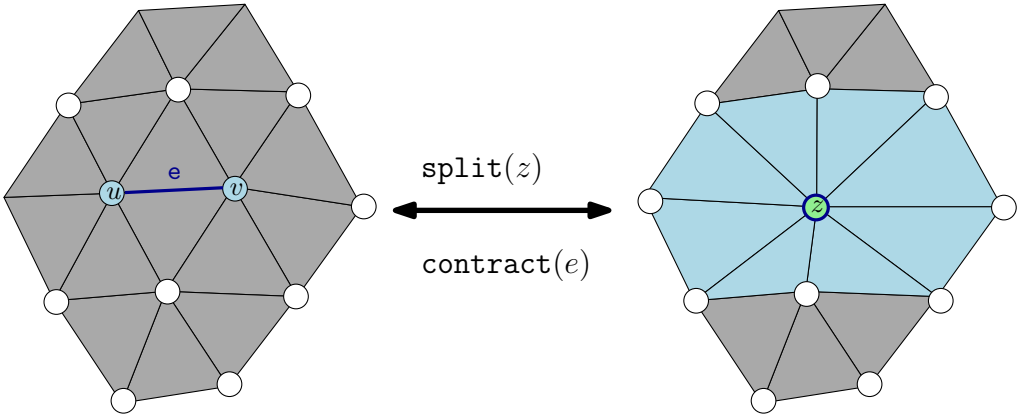
edge flip and *vertex insertion/removal*



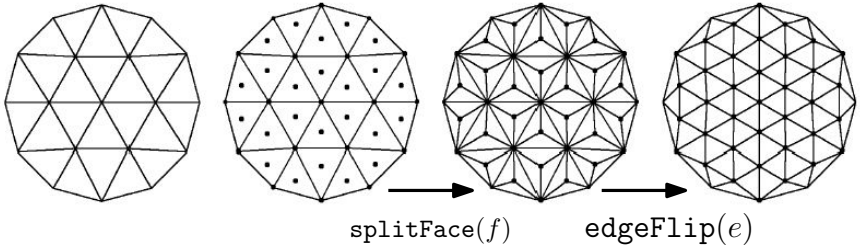
"Topological changes" (for triangle meshes)

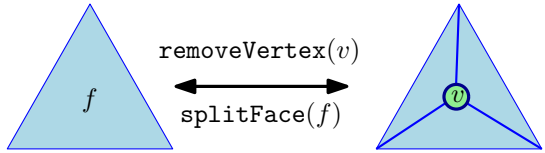


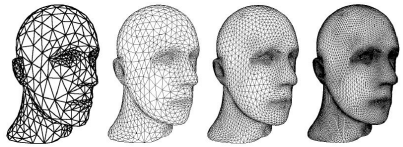
edge contraction and *vertex split*

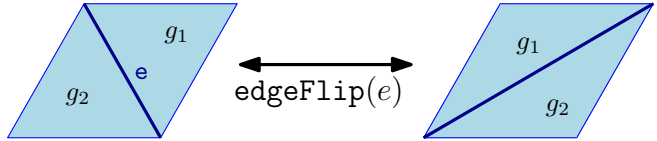


edge flip and *vertex insertion/removal*

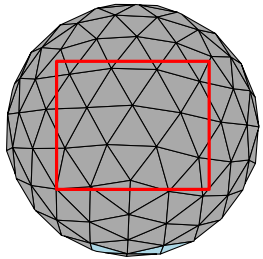


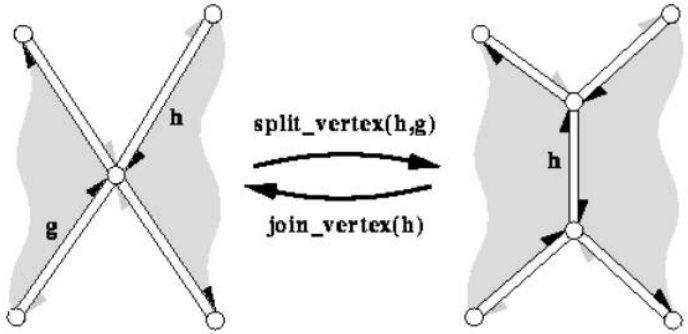


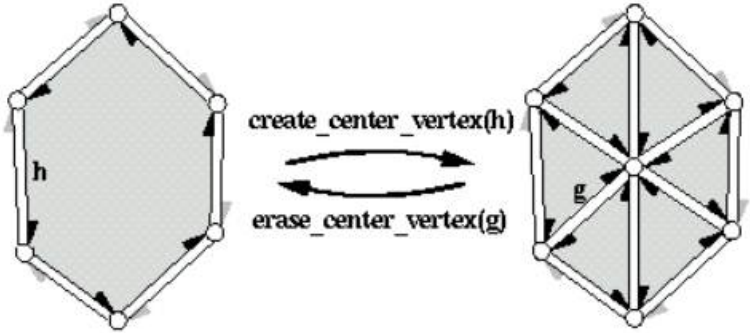


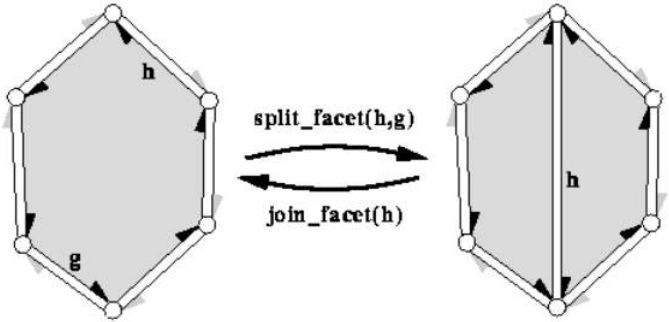


Euler operators (for general meshes)

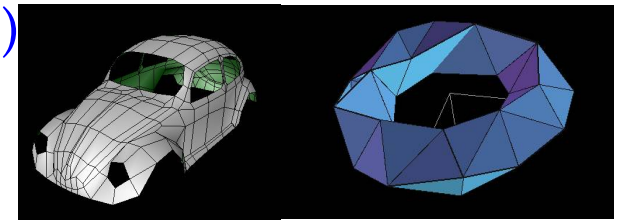


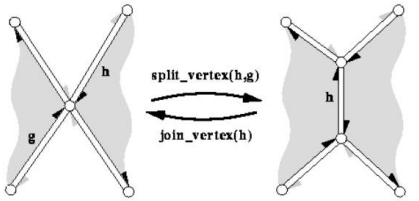


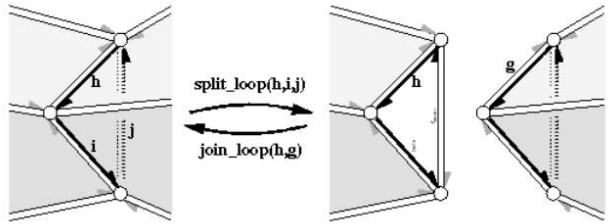


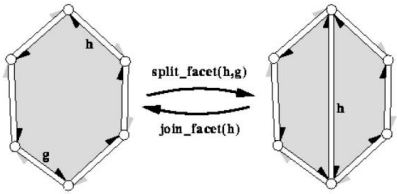


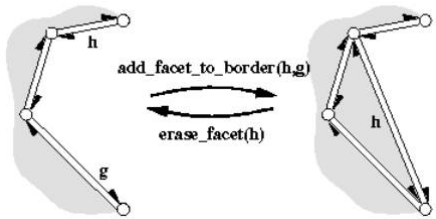
Euler operators (for general meshes)

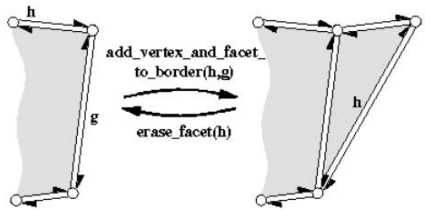


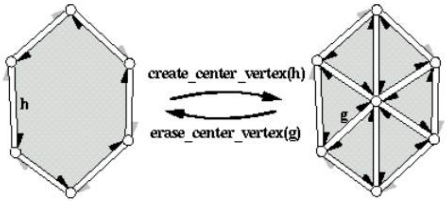






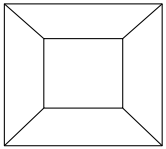






Euler-Poincaré characteristic: topological invariant

$$\chi := n - e + f$$



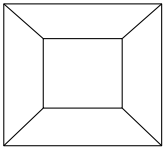
planar map

$$n - e + f = 2$$

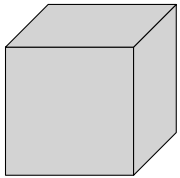
Euler's relation

Euler-Poincaré characteristic: topological invariant

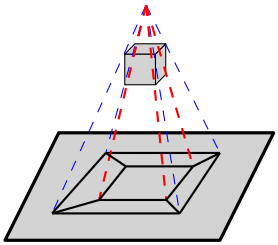
$$\chi := n - e + f$$



planar map



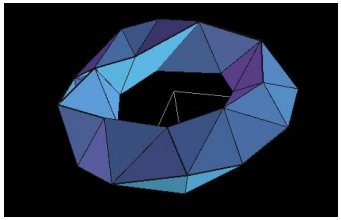
(convex) polyhedron



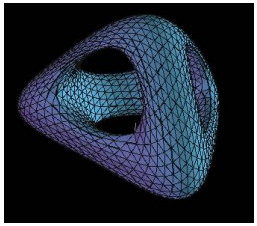
$$n - e + f = 2$$

Euler's relation

$\chi = 0$

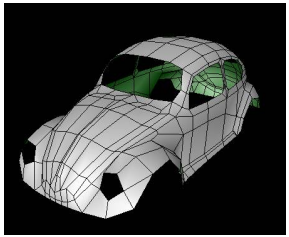


$\chi = -4$



$n = 1660$
$e = 4992$
$f = 3328$
$g = 3$

$$n - e + f = 2 - 2g$$



$n = 364$
$e = 675$
$f = 302$
$b = 11$
$g = 0$

$$n - e + f = 2 - b$$

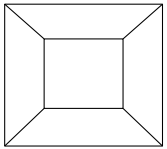
Euler's relation for polyhedral surfaces

$\chi := n - e + f$

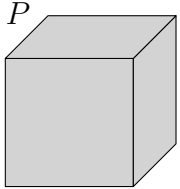
$\chi(t) = 3 - 3 + 2 = 2$

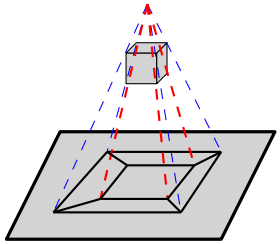
First proof: by induction

M



P



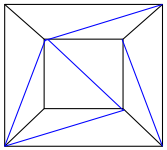


$n - e + f = 2$

Euler's relation

$\chi(M) = \chi(P) - 1$ (count exterior face)

M^t



$\chi(M) = \chi(M^t)$

invariant: the boundary (exterior) is a simple cycle

perform the removal according to a shelling order

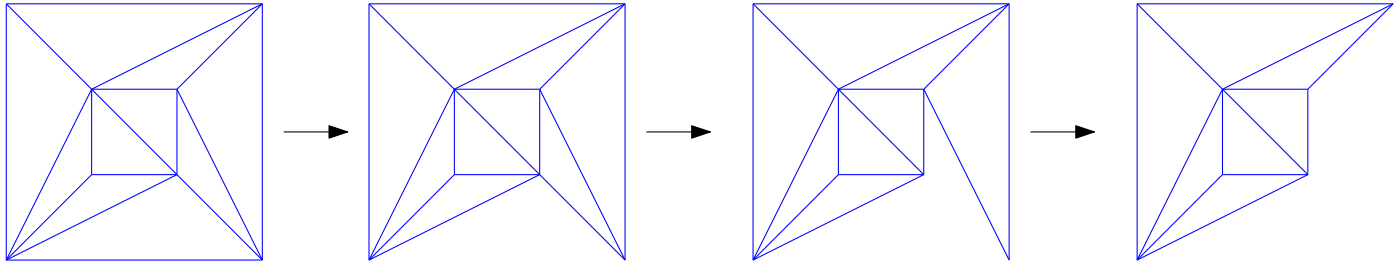
$e''' = e'' - 2$

$f''' = f'' - 1$

$n''' = n'' - 1$

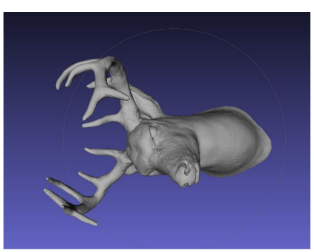
$e' = e - 1 \quad f' = f - 1 \qquad e'' = e' - 1 \quad f'' = f' - 1$

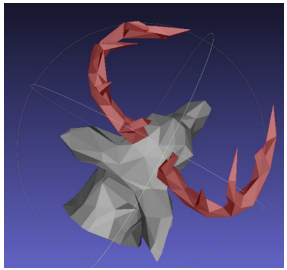
M^t

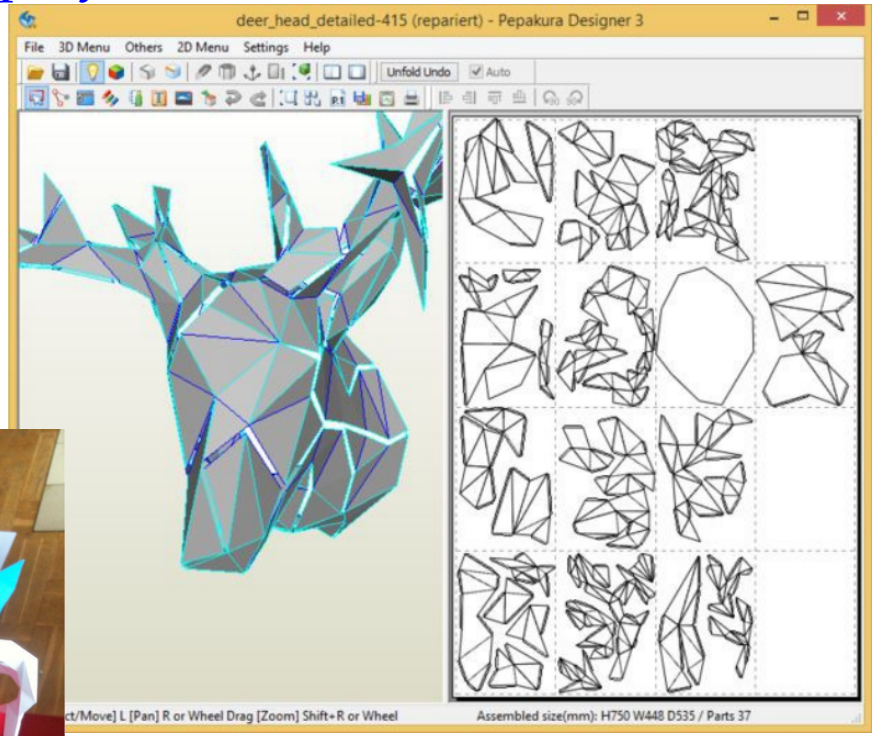


remove a boundary edge — remove a boundary edge — remove a triangle

Euler's relation: unfolding polyhedral surfaces







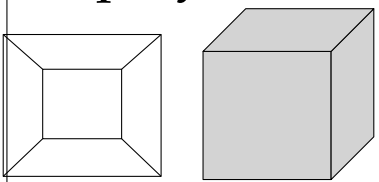
Pepakura

Euler's relation for polyhedral surfaces

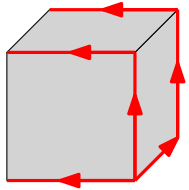
Overview of the proof

$$n - e + f = 2$$

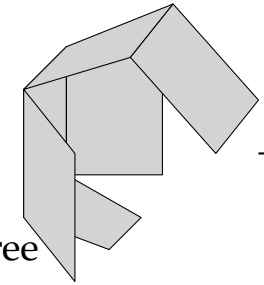
polyhedron, n vertices

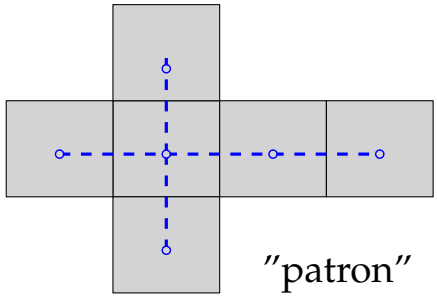


planar map

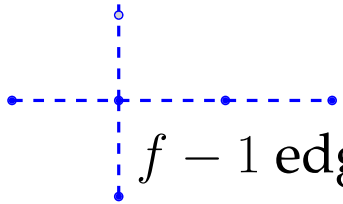


vertex spanning tree
$n - 1$ edges





"patron"



$f - 1$ edges

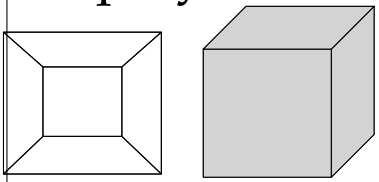
spanning tree (of the dual)

Euler's relation for polyhedral surfaces

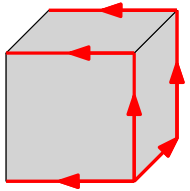
Overview of the proof

$$n - e + f = 2$$

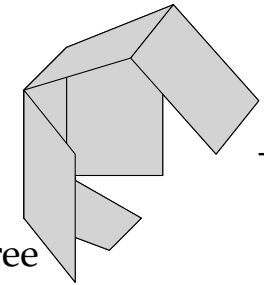
polyhedron, n vertices

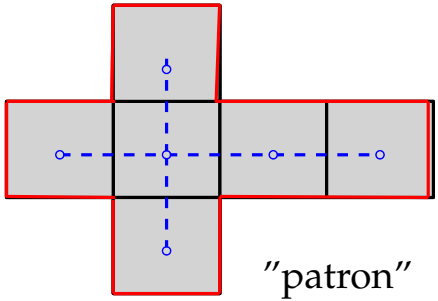


planar map



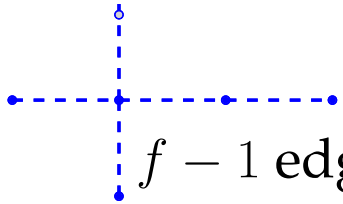
vertex spanning tree
$n - 1$ edges





"patron"

$$e = (n - 1) + (f - 1)$$



$f - 1$ edges

spanning tree (of the dual)

Euler's relation for polyhedral surfaces

Corollary: linear dependence between edges, vertices and faces

$f \leq 2n - 4$

$$e \leq 3n - 6$$

proof (double counting argument)

$$f = f_1 + f_2 + f_3 + \dots$$
$$n = n_1 + n_2 + n_3 + \dots$$

every face has degree at lest 3 ($\mathcal{G}$ is simple), we have

$$f = f_3 + f_4 + \dots$$

every edge is counted twice

$$2e = 3 \cdot f_3 + 4 \cdot f_4 + \dots$$

thus

$$2e - 3f \geq 0$$

Euler's relation for polyhedral surfaces

Corollary: linear dependence between edges, vertices and faces

$f \leq 2n - 4$

$$e \leq 3n - 6$$

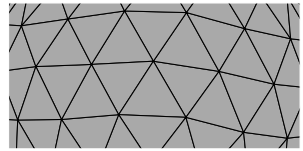
we just showed $2e - 3f \geq 0$

using Euler relation we obtain

$$3n - 6 = 3(e - f + 2) - 6 = e + (2e - 3f) \geq 0$$

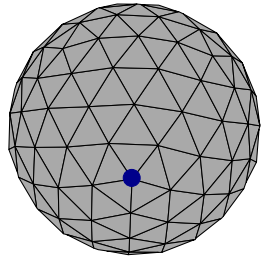
Euler's relation for polyhedral surfaces

can we construct a regular (genus 0) mesh, where every vertex has degree 6?



Euler's relation for polyhedral surfaces

we just showed $2e - 3f \geq 0$



proof (double counting argument)

Assume all the vertices have degrees ≥ 6:

the total number of vertices is: $n = n_6 + n_7 + n_8 + \ldots$

using a double counting of edges: $2e = 6 \cdot n_6 + 7 \cdot n_7 + 8 \cdot n_8 + \ldots$

$\downarrow$

$2e - 6 \cdot n \geq 0$

$$\left.\begin{array}{l} 2e - 6 \cdot n \geq 0 \\ 2e - 3f \geq 0 \end{array}\right\} \longrightarrow 6(e - n - f) = (2e - 6n) + 2(2e - 3f) \geq 0$$

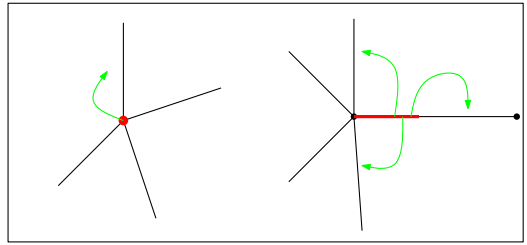
$\swarrow$

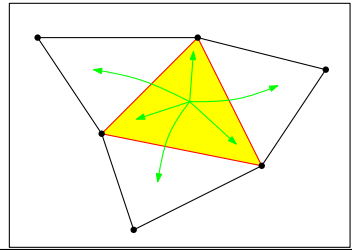
$$e - n - f \geq 0 \longrightarrow e \geq n + f$$

condtradicting Euler formula: $e = n + f - 2$

Part II
Mesh representations and data structures

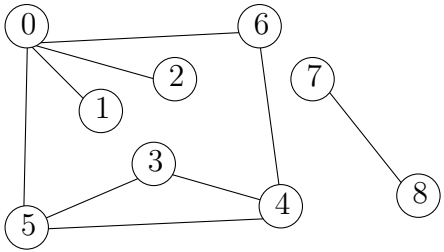
- Shared vertex representation
- Half-edge DS
- Winged edge
- Triangle based DS
- Corner Table





Graph: adjacency lists representation

easy to implement
quite compact
not efficient for traversal



for each face (of degree d), store:
- d references to adjacent vertices

for each vertex, store:
- 1 reference to its coordinates

```
class Point{
  double x;
  double y;
}
```

geometric information

```
class Vertex{
 Point p;
 List<Vertex> neigh
}
```

combinatorial information

Memory cost

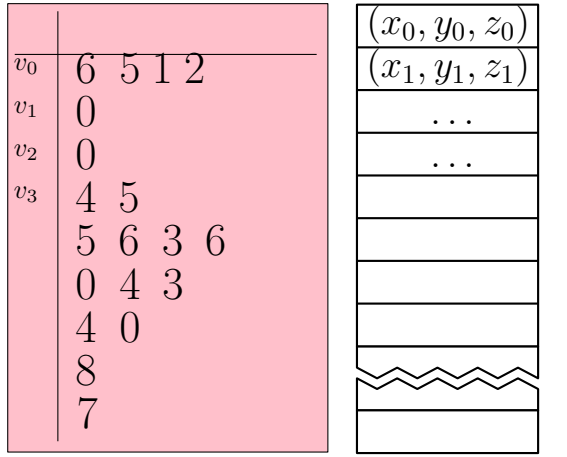
$$\sum_i deg(v_i) = 2 \times e$$

Size (number of references)

Queries/Operations

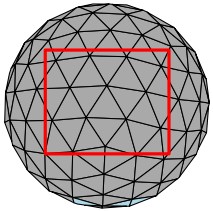
List all vertices
Test adjacency between u and v
Find the 3 neighboring faces of f
List the neighbors of vertex v

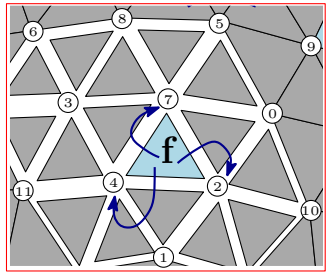
vertex locations



Shared vertex representation

easy to implement
quite compact
not efficient for traversal





for each face (of degree d), store:
- d references to incident vertices

for each vertex, store:
- 1 reference to its coordinates

```
class Point{
  double x;
  double y;
}
```

geometric information

```
class Vertex{
  Point p;
}
class Face{
Vertex[]  vertices;
}
```

combinatorial information

Memory cost

$$3 \times f = 6n$$

Size (number of references)

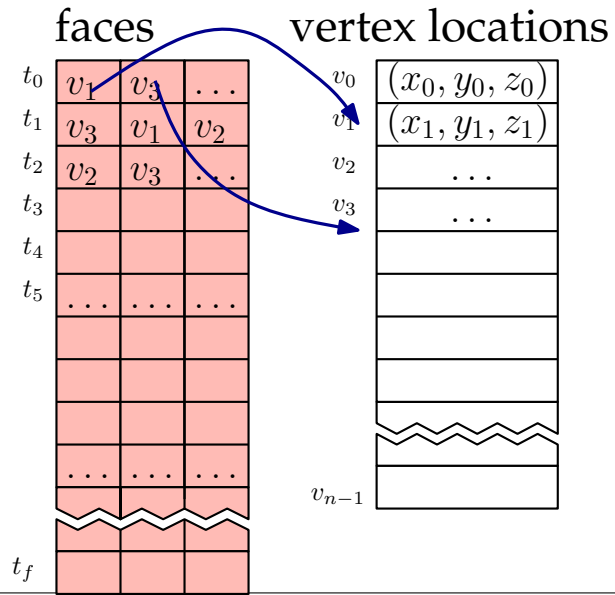
Queries/Operations

List all vertices or faces

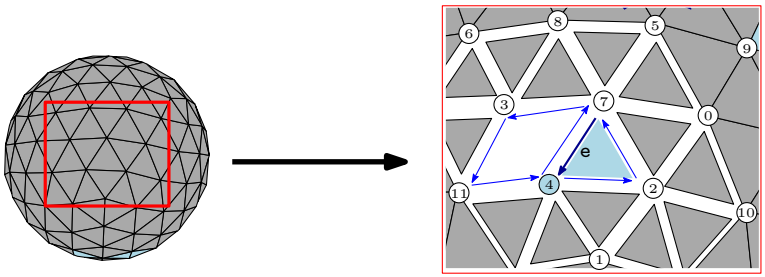
Test adjacency between u and v

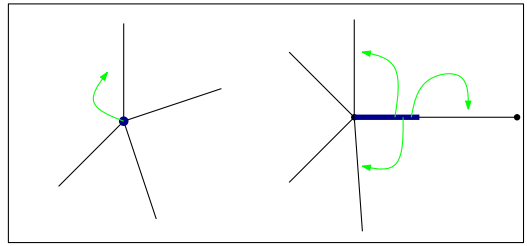
Find the 3 neighboring faces of f

List the neighbors of vertex v



Half-edge data structure: polygonal (orientable) meshes



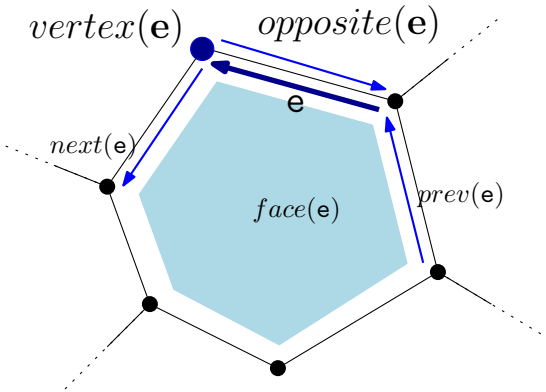


$$f + 5 \times h + n \approx 2n + 5 \times (2e) + n = 32n + n$$

Size (number of references)

```
class Point{
  double x;
  double y;
}
```

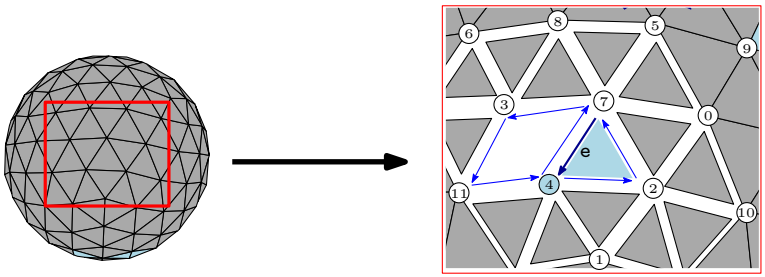
geometric information

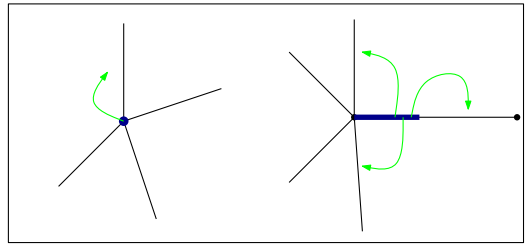



```
class Halfedge{
  Halfedge  prev, ne
  Vertex v;
  Face f;
}class Vertex{
  Halfedge  e;
  Point p;
}
 class Face{
  Halfedge  e;
}
```

combinatorial information

Half-edge data structure: polygonal (orientable) meshes



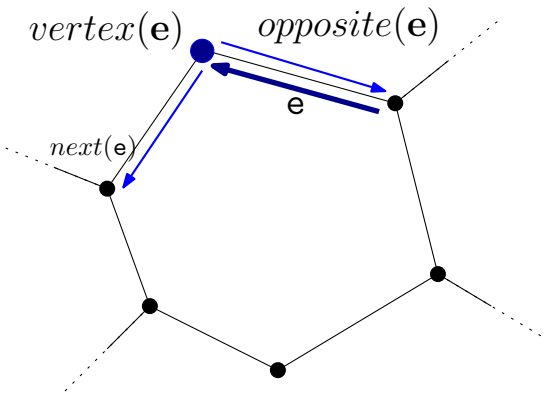


$$3 \times h + n \approx 3 \times (2e) + n = 18n + n$$

Size (number of references)

```
class Point{
  double x;
  double y;
}
```

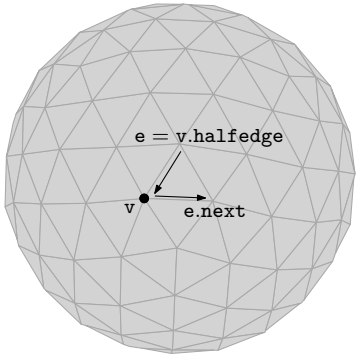
geometric information

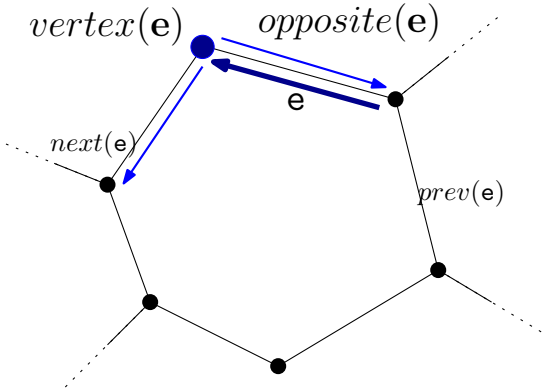



```
class Halfedge{
  Halfedge  prev, ne
  Vertex v;
  Face f;
}class Vertex{
  Halfedge  e;
  Point p;
}
 class Face{
  Halfedge  e;
}
```

combinatorial information

Half-edge data structure: efficient traversal

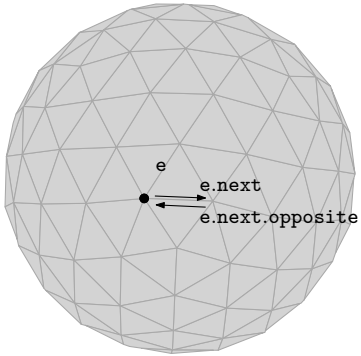


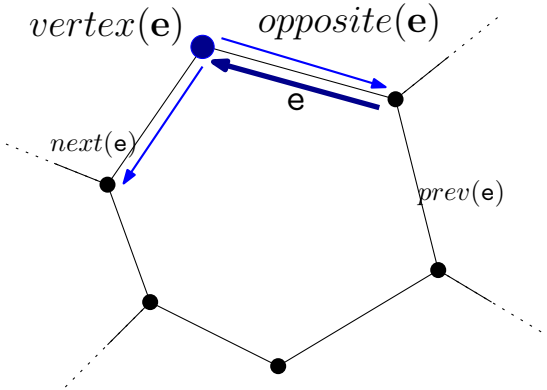



```java
public int vertexDegree(Vertex<X> v) {
    int result=0;
    Halfedge<X> e=v.getHalfedge();

    Halfedge<X> pEdge=e.getNext().getOpposite();
    while(pEdge!=e) {
        pEdge=pEdge.getNext().getOpposite();
        result++;
    }

    return result+1;
}
```

```java
public int degree() {
    Halfedge<X> e,p;
    if(this.halfedge==null) return 0;

    e=halfedge; p=halfedge.next;
    int cont=1;
    while(p!=e) {
        cont++;
        p=p.next;
    }
    return cont;
}
```

Half-edge data structure: efficient traversal

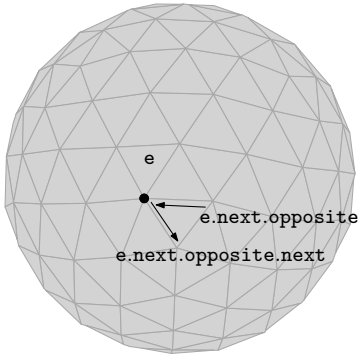


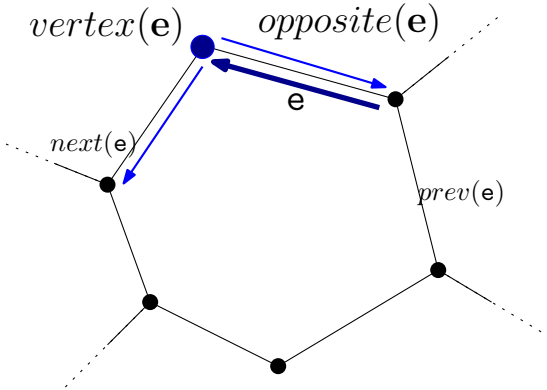



```java
public int vertexDegree(Vertex<X> v) {
    int result=0;
    Halfedge<X> e=v.getHalfedge();

    Halfedge<X> pEdge=e.getNext().getOpposite();
    while(pEdge!=e) {
        pEdge=pEdge.getNext().getOpposite();
        result++;
    }

    return result+1;
}
```

```java
public int degree() {
    Halfedge<X> e,p;
    if(this.halfedge==null) return 0;

    e=halfedge; p=halfedge.next;
    int cont=1;
    while(p!=e) {
        cont++;
        p=p.next;
    }
    return cont;
}
```

Half-edge data structure: efficient traversal

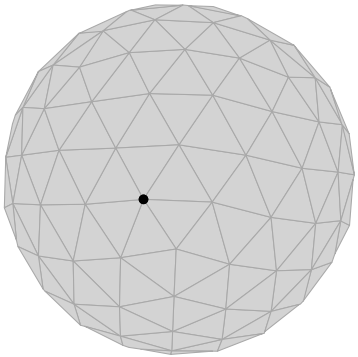


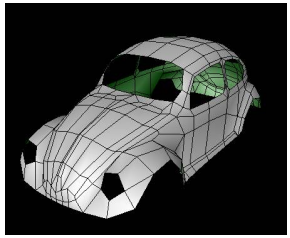


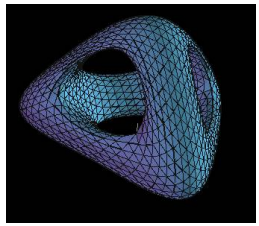

```java
public int vertexDegree(Vertex<X> v) {
    int result=0;
    Halfedge<X> e=v.getHalfedge();

    Halfedge<X> pEdge=e.getNext().getOpposite();
    while(pEdge!=e) {
        pEdge=pEdge.getNext().getOpposite();
        result++;
    }

    return result+1;
}
```

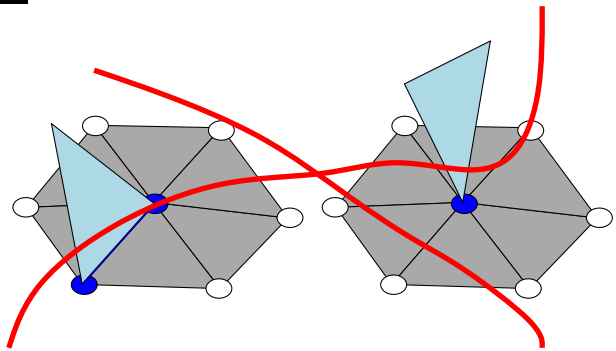
```java
public int degree() {
    Halfedge<X> e,p;
    if(this.halfedge==null) return 0;

    e=halfedge; p=halfedge.next;
    int cont=1;
    while(p!=e) {
        cont++;
        p=p.next;
    }
    return cont;
}
```

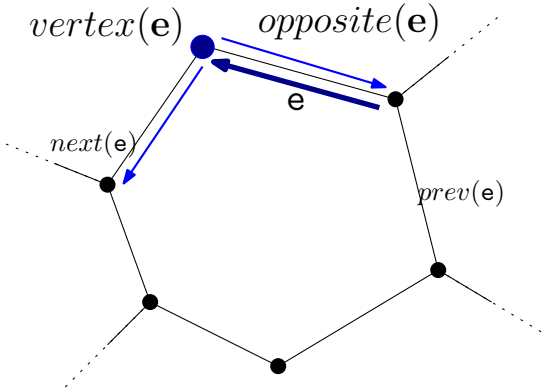
Half-edge data structure: polygonal manifold meshes





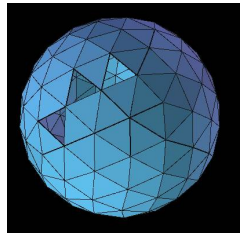


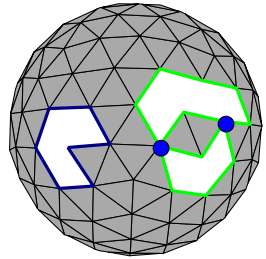




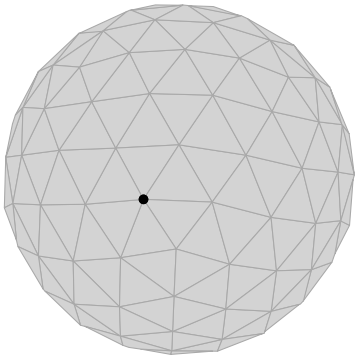
can we represent them?

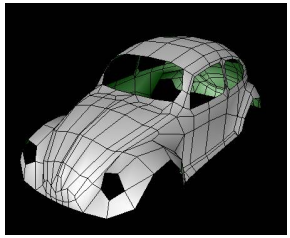
?

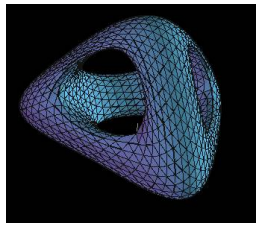


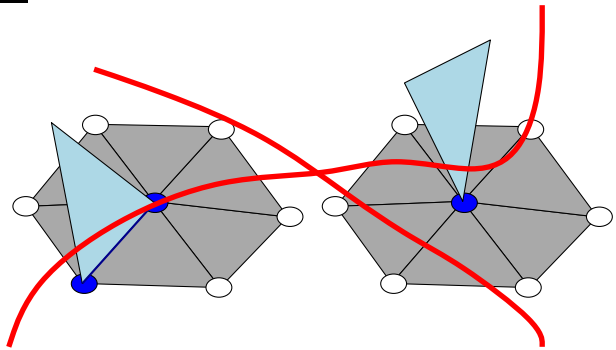


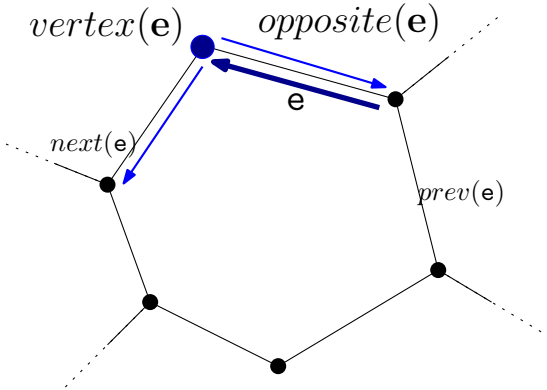
Half-edge data structure: polygonal manifold meshes





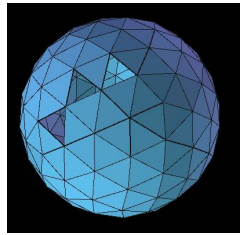


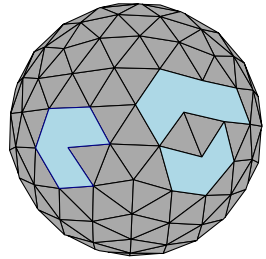




can we represent them?

yes





Triangle based DS: for triangle meshes (used in CGAL)

```
class Point{
  float x;
  float y;
  float z;
}
```

```
class Triangle{
  Triangle t1, t2
  Vertex v1, v2,
}
 class Vertex{
  Triangle  root;
  Point p;
}
```

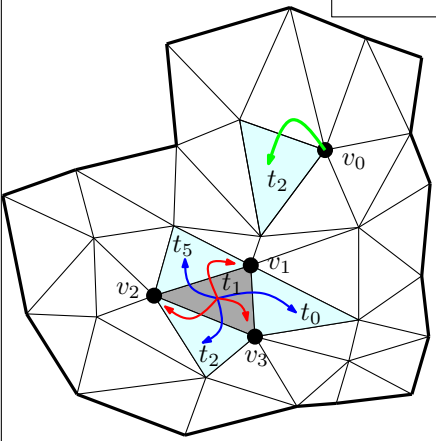
connectivity

for each triangle, store:

- 3 references to neighboring faces
- 3 references to incident vertices

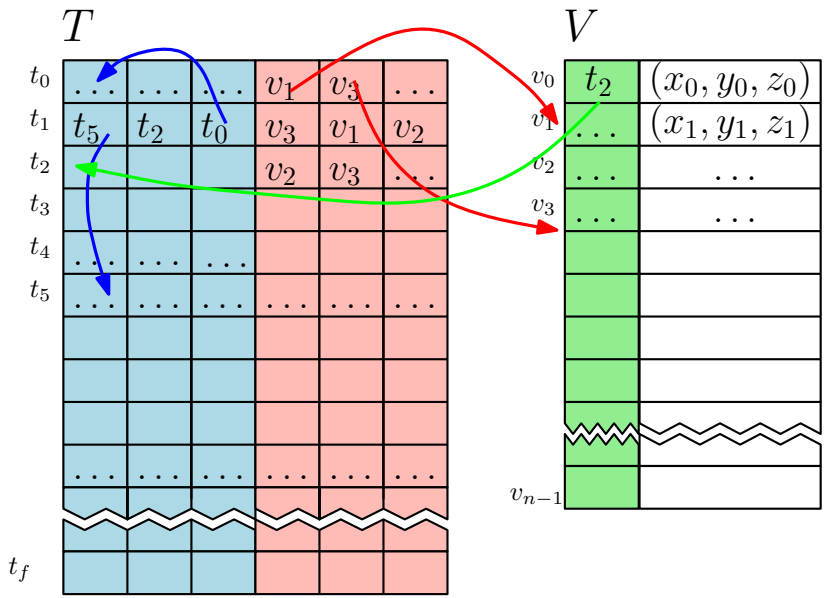
for each vertex, store:

- 1 reference to an incident face



$(3+3) \times f + n = 6 \times 2n + n = 13n$

Size (number of references)



Triangle based DS: mesh traversal operators

```
class Point{
  float x;
  float y;
  float z;
}
```

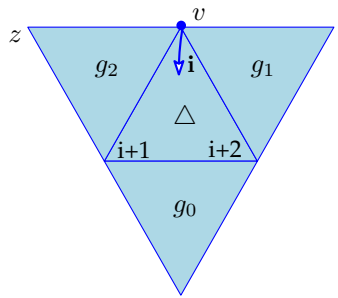
```
class Triangle{
  Triangle t1, t2
  Vertex v1, v2,
}
 class Vertex{
  Triangle  root;
  Point p;
}
```

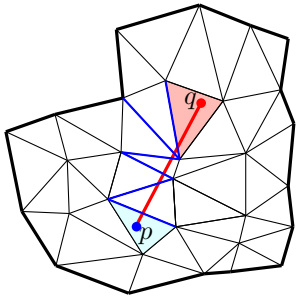
connectivity

the data structure supports the following operators

$v = \texttt{vertex}(\triangle, i)$
$\triangle = \texttt{face}(v)$
$i = \texttt{vertexIndex}(v, \triangle)$
$g_0 = \texttt{neighbor}(\triangle, i)$
$g_1 = \texttt{neighbor}(\triangle, ccw(i))$
$g_2 = \texttt{neighbor}(\triangle, cw(i))$
$z = \texttt{vertex}(g_2, \texttt{faceIndex}(g_2, \triangle))$

```
int cw(int i) {return (i + 2)%3
int ccw(int i) {return (i + 1)%
```





we can locate a point, by performing a walk in the triangulation

```
int degree(int v) {
  int d = 1;
  int f = face(v);
  int g = neighbor(f, cw(vertexIndex(v, f)));
  while (g != f) {
    int next = neighbor(g, cw(face
    int i = faceIndex(g, next);
    g = next;
    d++;
  }
  return d;
}
```

we can turn around a vertex, by combining the operators above

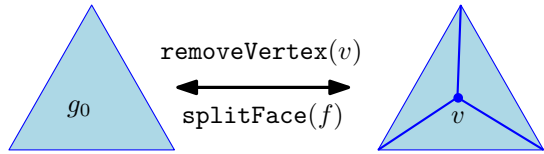
Triangle based DS: mesh update operators

```
class Point{
  float x;
  float y;
  float z;
}
```

```
class Triangle{
  Triangle t1, t2
  Vertex v1, v2,
}
 class Vertex{
  Triangle  root;
  Point p;
}
```
connectivity

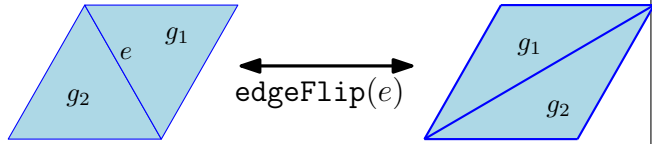
the data structure supports the following operators

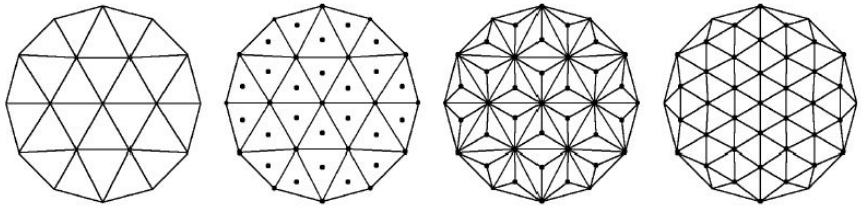
- `removeVertex`(v)
- `splitFace`(f)
- `edgeFlip`(e)

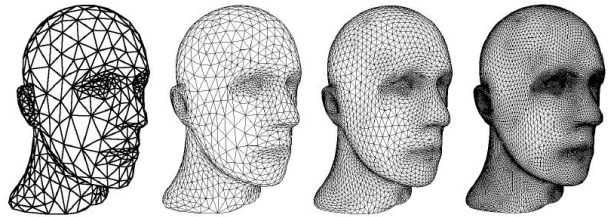


the data structure is **modifiable**

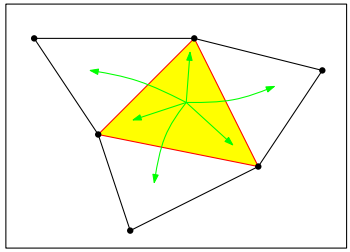
all these operators can be performed in $O(1)$ time





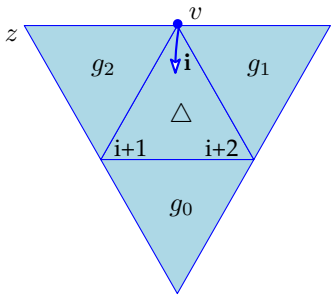


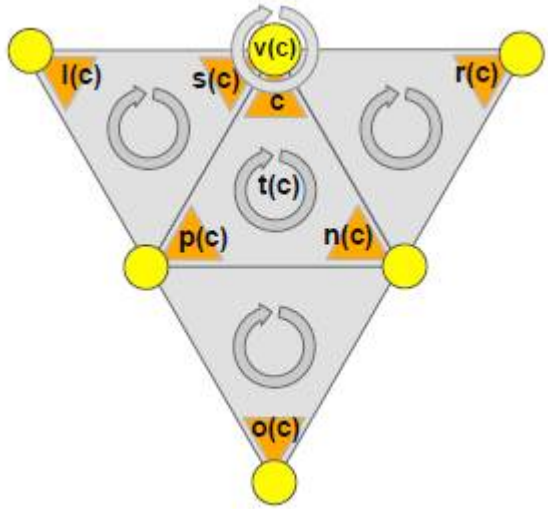
Corner Table (face based representation)



$v = \texttt{vertex}(\triangle, i)$
$\triangle = \texttt{face}(v)$
$i = \texttt{vertexIndex}(v, \triangle)$
$g_0 = \texttt{neighbor}(\triangle, i)$
$g_1 = \texttt{neighbor}(\triangle, ccw(i))$
$g_2 = \texttt{neighbor}(\triangle, cw(i))$
$z = \texttt{vertex}(g_2, \texttt{faceIndex}(g_2, \triangle))$

```
int cw(int i) {return (i + 2)%3
int ccw(int i) {return (i + 1)%
```



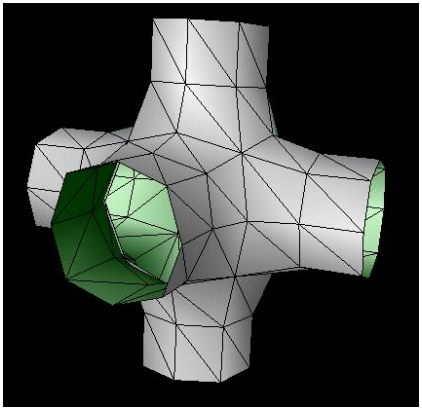


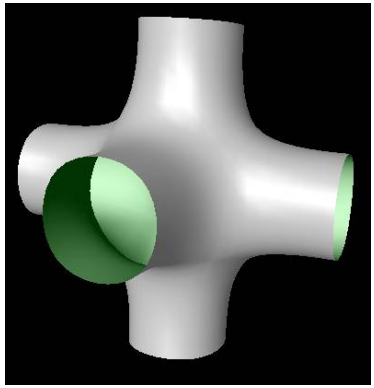

```
int t(int c) {return int(c/3);}                    // triangle of c
int v(int c) {return V[c];}                        // vertex of c
int o(int c) {return O[c];}                        // opposite
int n(int c) {if ((c%3)==2) return c−1; else return c+1;}// next in t(c)
int p(int c) {return n(n(c));}                     // previous corner
int l(int c) {return o(p(c));}                     // tip on left
int r(int c) {return o(n(c));}                     // tip on right
int s(int c) {return n(l(c));}                     // next around v(c)
```

Part III
Mesh simplification algorithms

Quadric error metric

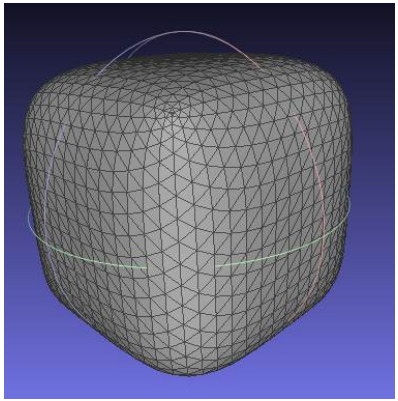
Mesh simplification: motivation and applications

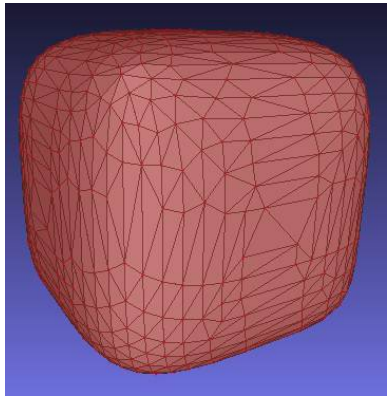


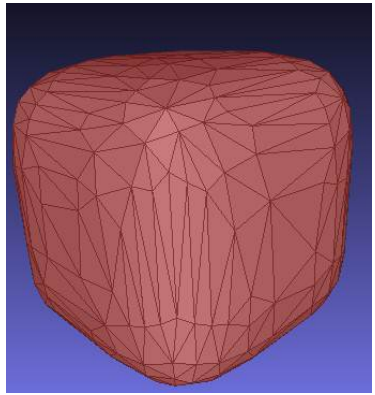


subdivision surfaces

mesh simplification

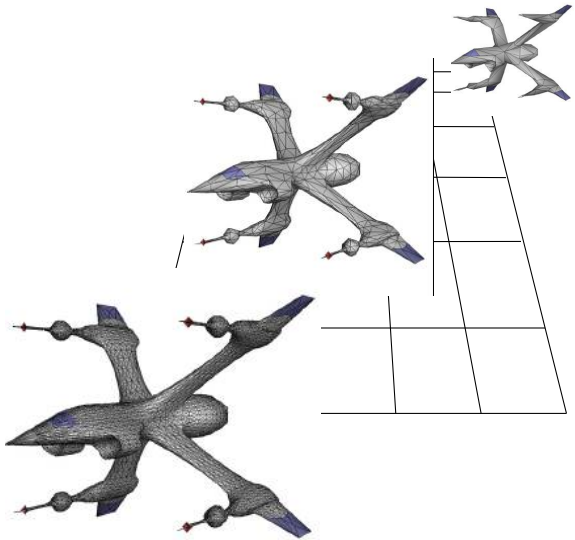






Mesh simplification: motivation and applications

Progressive meshes (H. Hoppe 1996)

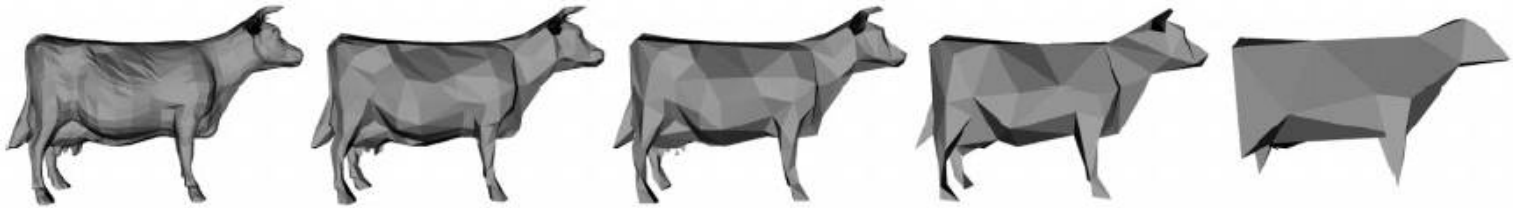


Level-of-detail (LOD) rendering, for large meshes

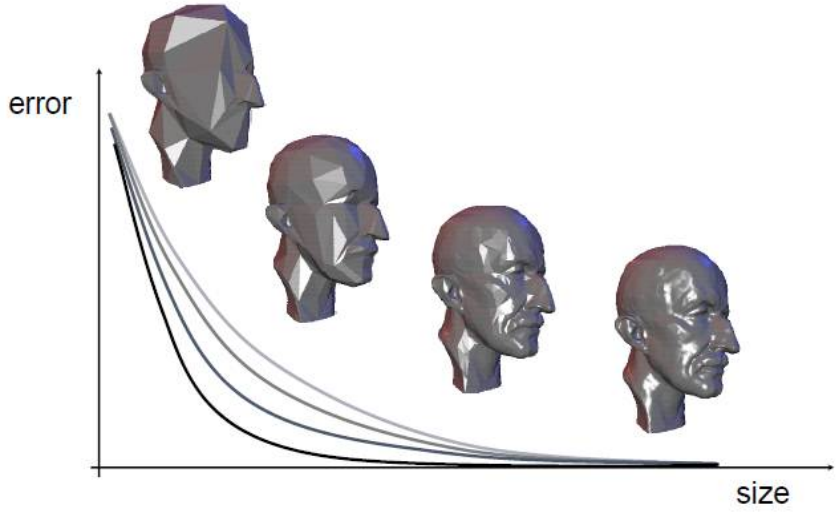
efficient geometry processing

progressive transmission and compression

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)



Mesh simplification: goals and trade-offs



Incremental decimation: general framework

Evaluate local error

Select removable elements (vertices or edges)

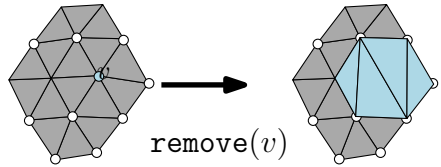
`while` $(error > \epsilon)$ { /* or $(n' > C)$*/

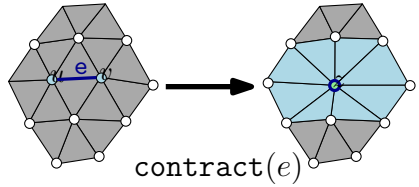
select vertex (or edge to be contracted)

remove vertex (or contract edge)

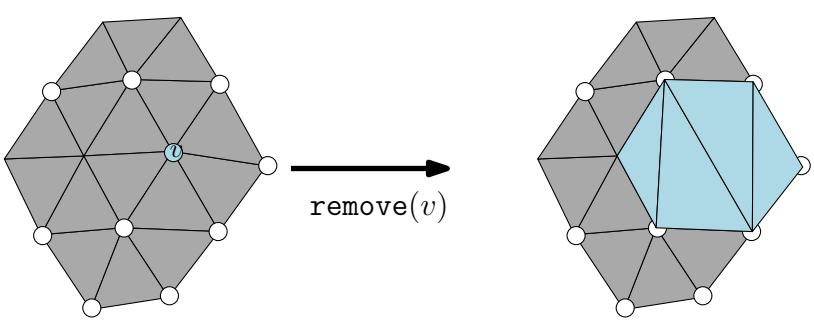
update local error for neighboring vertices (edges)

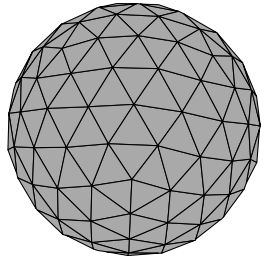
}

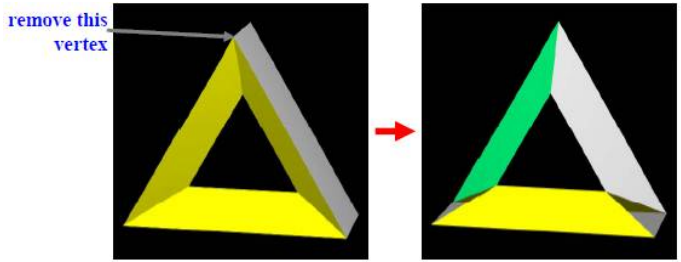




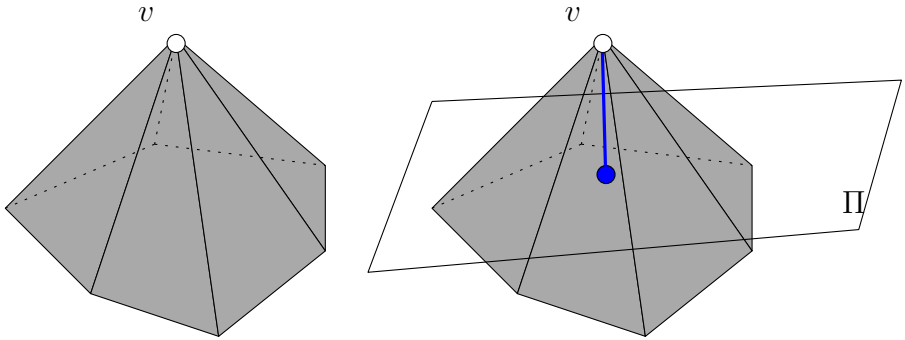
Incremental decimation: vertex removal

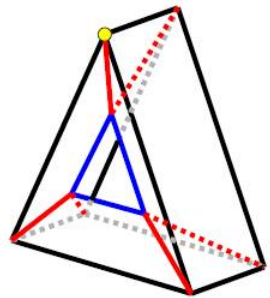






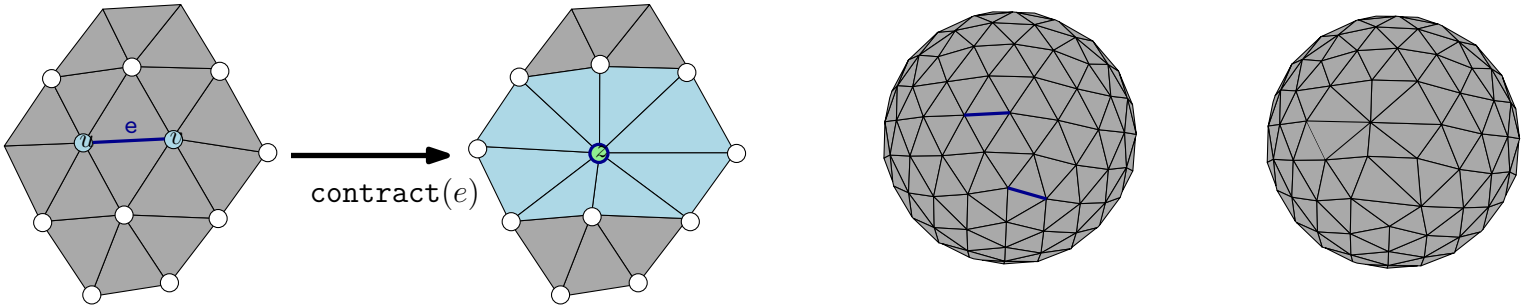
Iteratively perform vertex removals





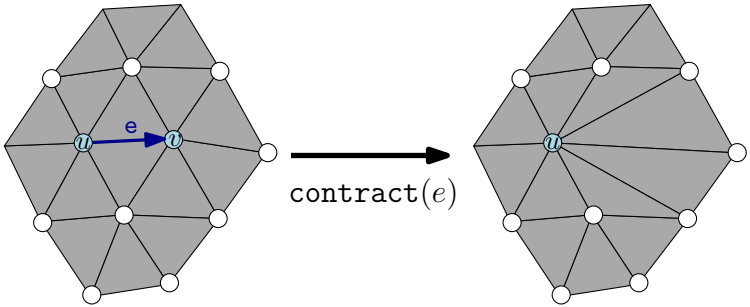
(images by C-K Shene, www.cs.mtu.edu/~shene/)

Incremental decimation: edge contraction



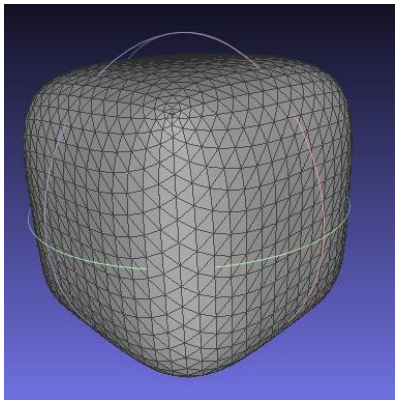
Iteratively perform edge contractions

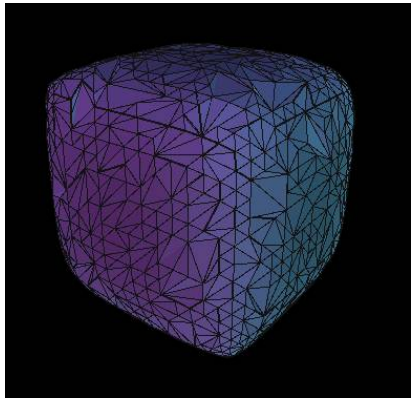
Incremental decimation: half-edge contraction

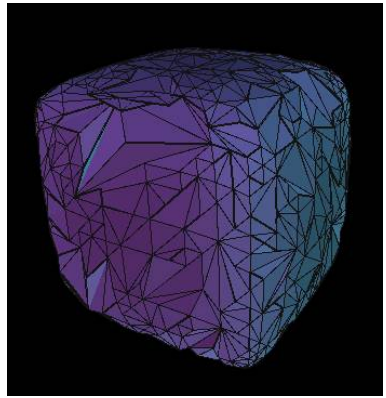


Half-edge contractions based on random selection (no geometric criteria)

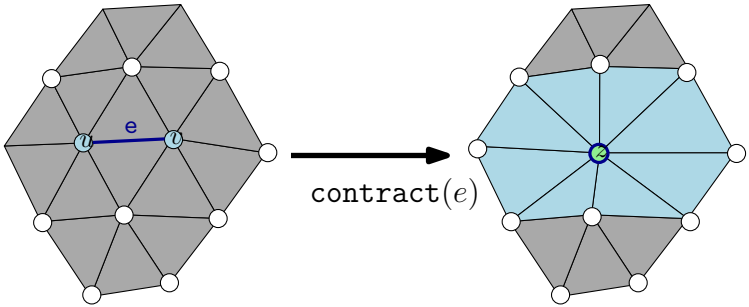
Vertex locations correspond to original coordinates of input points

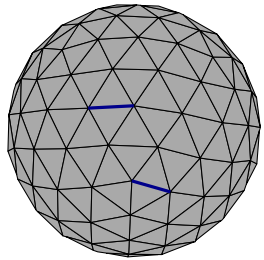


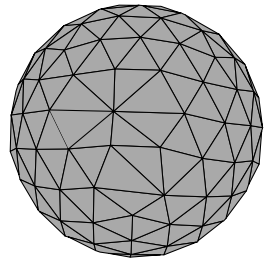




Incremental decimation: edge contraction

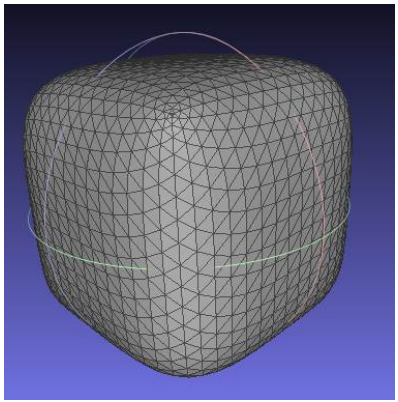


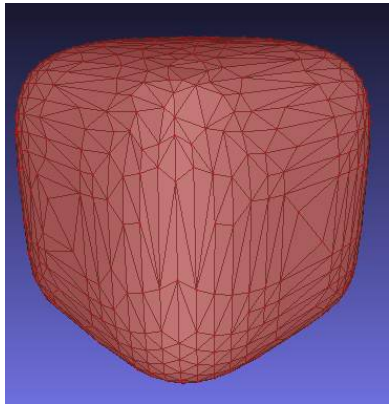




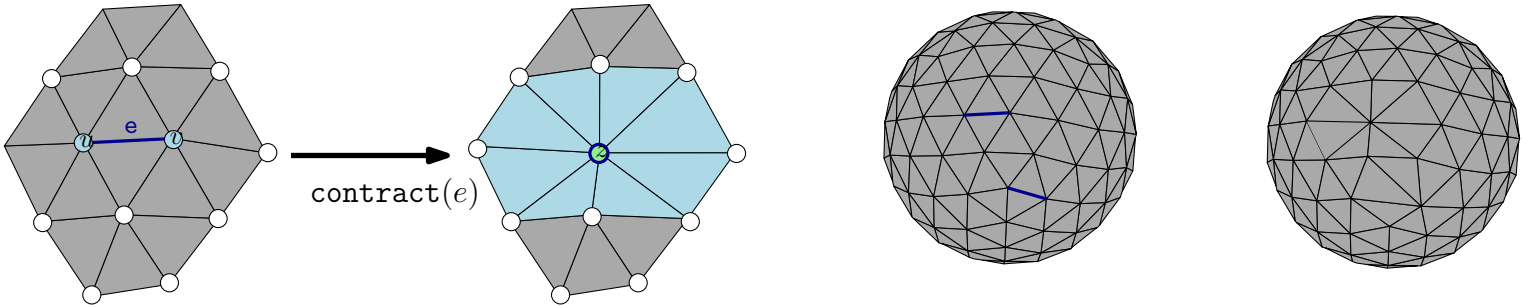
Select edge contractions based on local geometric criterion

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)





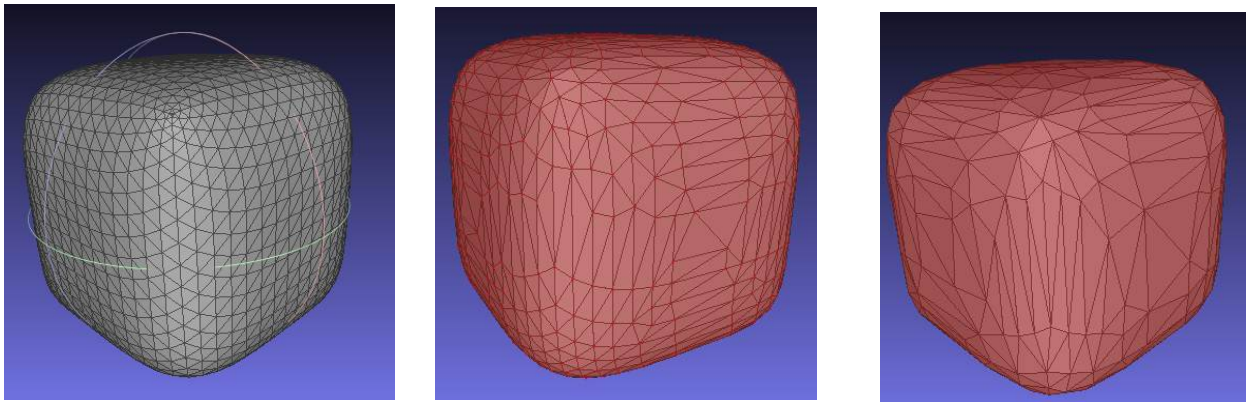
Incremental decimation: edge contraction



Select edge contractions based on local geometric criterion

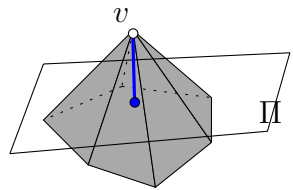
choose optimal vertex location

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)



Approximating error with quadrics

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)



Let be $\mathbf{v} = [x\ y\ z\ 1]$ a vertex (in homogeneous coordinates)

Associate to each vertex v a 4×4 matrix $\mathbf{Q}_v$

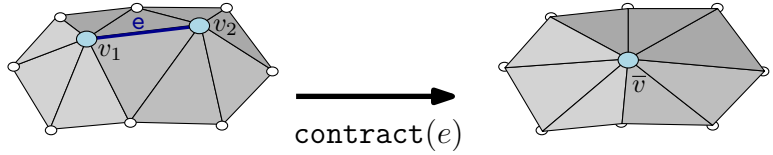
Error at vertex v: $\Delta(\mathbf{v}) := \mathbf{v}^T \mathbf{Q} \mathbf{v}$ (quadratic form)

Level set surface: $\{\mathbf{v} \mid \Delta(\mathbf{v}) = \epsilon\}$ (quadric surface)

Associate an error to new vertex location $\overline{v}$ (use additivity)

$$\overline{\mathbf{Q}} := \mathbf{Q}_1 + \mathbf{Q}_2$$

$e = (v_1, v_2) \rightarrow \overline{v}$



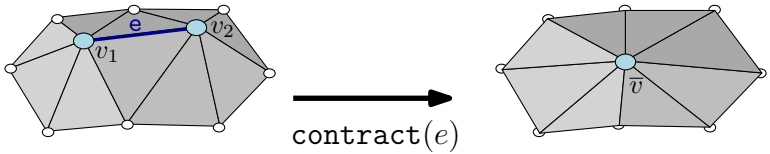
Approximating error with quadrics: optimal location

$\Delta(\mathbf{v}) := \mathbf{v}^T\mathbf{Q}\mathbf{v} \qquad \overline{\mathbf{Q}} := \mathbf{Q}_1 + \mathbf{Q}_2$

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)

$e = (v_1, v_2) \rightarrow \overline{v}$

Associate an error to new vertex location $\overline{v}$ (use additivity)



Find $\overline{\mathbf{v}}$ in order to minimize $\Delta(\overline{\mathbf{v}}) := \overline{\mathbf{v}}^T\mathbf{Q}\overline{\mathbf{v}}$

solve a linear system

$$\begin{cases} \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial x} = 0 \\ \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial y} = 0 \\ \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial z} = 0 \end{cases}$$

$$\Delta(\overline{\mathbf{v}}) := q_{11}x^2 + 2q_{12}xy + 2q_{13}xz + 2q_{14}x + q_{22}y^2 + 2q_{23}yz + 2q_{24}y + q_{33}z^2 + q_{44}$$

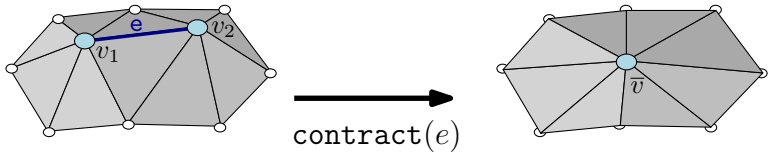
Approximating error with quadrics: optimal location

$$\Delta(\mathbf{v}) := \mathbf{v}^T\mathbf{Q}\mathbf{v} \qquad \overline{\mathbf{Q}} := \mathbf{Q}_1 + \mathbf{Q}_2$$

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)

$$e = (v_1, v_2) \rightarrow \overline{v}$$

Associate an error to new vertex location $\overline{v}$ (use additivity)



Find $\overline{\mathbf{v}}$ in order to minimize $\Delta(\overline{\mathbf{v}}) := \overline{\mathbf{v}}^T\mathbf{Q}\overline{\mathbf{v}}$

solve a linear system

$$\begin{cases} \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial x} = 0 \\ \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial y} = 0 \\ \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial z} = 0 \end{cases} \longleftrightarrow \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \overline{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\Delta(\overline{\mathbf{v}}) := q_{11}x^2 + 2q_{12}xy + 2q_{13}xz + 2q_{14}x + q_{22}y^2 + 2q_{23}yz + 2q_{24}y + q_{33}z^2 + q_{44}$$

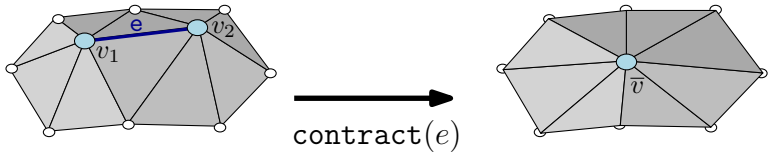
Approximating error with quadrics: optimal location

$\Delta(\mathbf{v}) := \mathbf{v}^T\mathbf{Q}\mathbf{v}$ $\qquad$ $\overline{\mathbf{Q}} := \mathbf{Q}_1 + \mathbf{Q}_2$

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)

$$e = (v_1, v_2) \rightarrow \overline{v}$$

Associate an error to new vertex location $\overline{v}$ (use additivity)



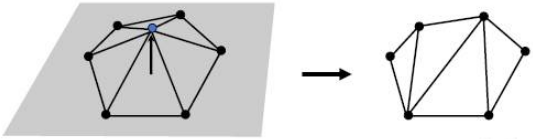
Find $\overline{\mathbf{v}}$ in order to minimize $\Delta(\overline{\mathbf{v}}) := \overline{\mathbf{v}}^T\mathbf{Q}\overline{\mathbf{v}}$

if $\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix}$ is invertible $\longrightarrow$ $\overline{\mathbf{v}} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

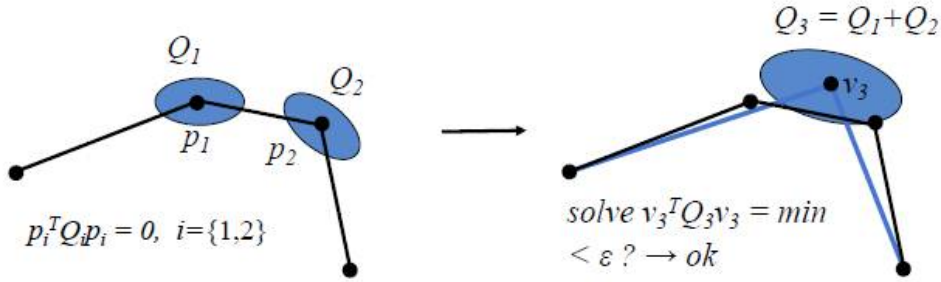
otherwise, select $\overline{\mathbf{v}} \in \{v_1, v_2, v_1+v_2\}$

Local errors

- Local distance to mesh
  - Compute average plane
  - No comparison to *original* geometry



- Error quadrics
  - Squared distance to planes at vertex
  - No bound on true error



Approximating error with quadrics: optimal location

$$\Delta(\mathbf{v}) = \Delta([v_x\ v_y\ v_z\ 1]^\mathsf{T}) = \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} (\mathbf{p}^\mathsf{T}\mathbf{v})^2$$

equation of a plane

$$ax + by + cz + d = 0$$

$$a^2 + b^2 + c^2 = 1$$

normalized vector

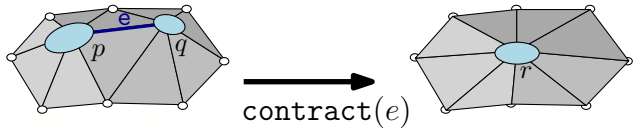
$$\mathbf{p} = [a\ b\ c\ 1]$$

$$\begin{aligned}
\Delta(\mathbf{v}) &= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} (\mathbf{v}^\mathsf{T}\mathbf{p})(\mathbf{p}^\mathsf{T}\mathbf{v}) \\
&= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{v}^\mathsf{T}(\mathbf{p}\mathbf{p}^\mathsf{T})\mathbf{v} \\
&= \mathbf{v}^\mathsf{T}\left(\sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{K}_\mathbf{p}\right)\mathbf{v}
\end{aligned}$$

$$\mathbf{K}_\mathbf{p} = \mathbf{p}\mathbf{p}^\mathsf{T} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

error metric in quadratic form

Quadric error metrics simplification: algorithm



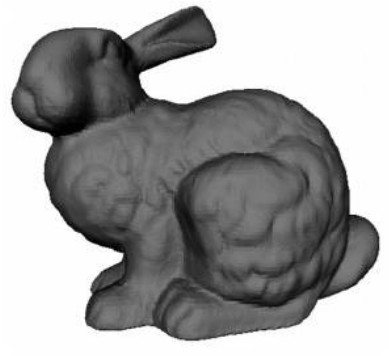
$$\mathbf{Q}_r := \mathbf{Q}_p + \mathbf{Q}_q$$

compute error and error matrix for each vertex of the mesh;
select all valid edges pq such that $|p - q| < \varepsilon$;
for each selected edge pq do
begin
minimize $\Delta(r) = [r^T(\mathbf{Q}_p + \mathbf{Q}_q)r]/2$ to find r;
let $\Delta(r) = (\Delta(p) + \Delta(q))/2$ and $\mathbf{Q}_r = \mathbf{Q}_p + \mathbf{Q}_q$;
place all selected edges in a heap using $\Delta(r)$ as a key;
end;
while there are edges on the heap do
begin
remove the top edge pq;
collapse it to the computed r;
update the mesh and the keys;
end

(pseudo-algorithm by C-K Shene, `www.cs.mtu.edu/~shene/`)

Quadric error metrics: geometric interpretation

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)



69451 faces

1000 faces

error ellipsoids for vertices
level sets of quadrics are ellipsoids

Part IV
Mesh compression algorithms

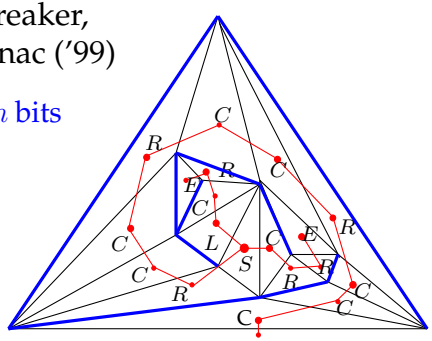
Edgebreaker

Codage de triangulations et arbres couvrants

common visual framework (Isenburg Snoeyink'05)

Edgebreaker, Rossignac ('99)

$3.67n$ bits

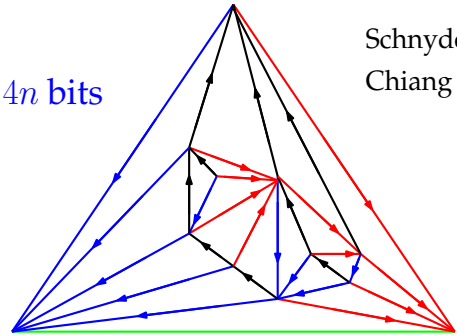


$CCCRCCRCCRECRRELCRE$

Schnyder woods

Chiang at al. ('98)

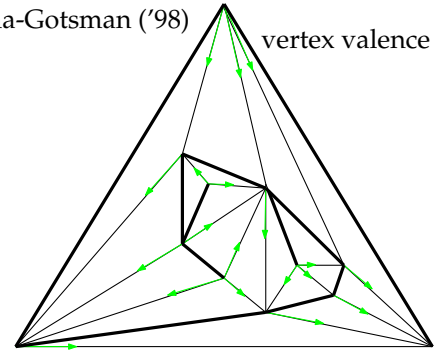
$4n$ bits



([[[ ) ( ] ( ] ( ] [[[ ) [ ) ( ]] [ ) . . .

Touma-Gotsman ('98)

vertex valence encoding

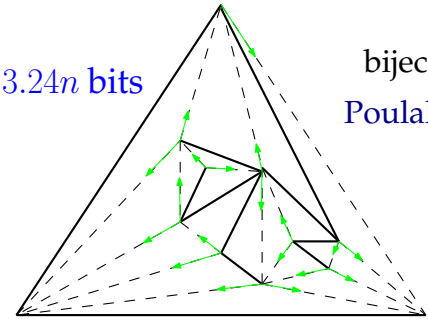


$V_5V_5V_6V_5V_4V_5V_8V_5V_5V_4S_4V_3V_4$

bijective construction

Poulalhon, Schaeffer (2003)

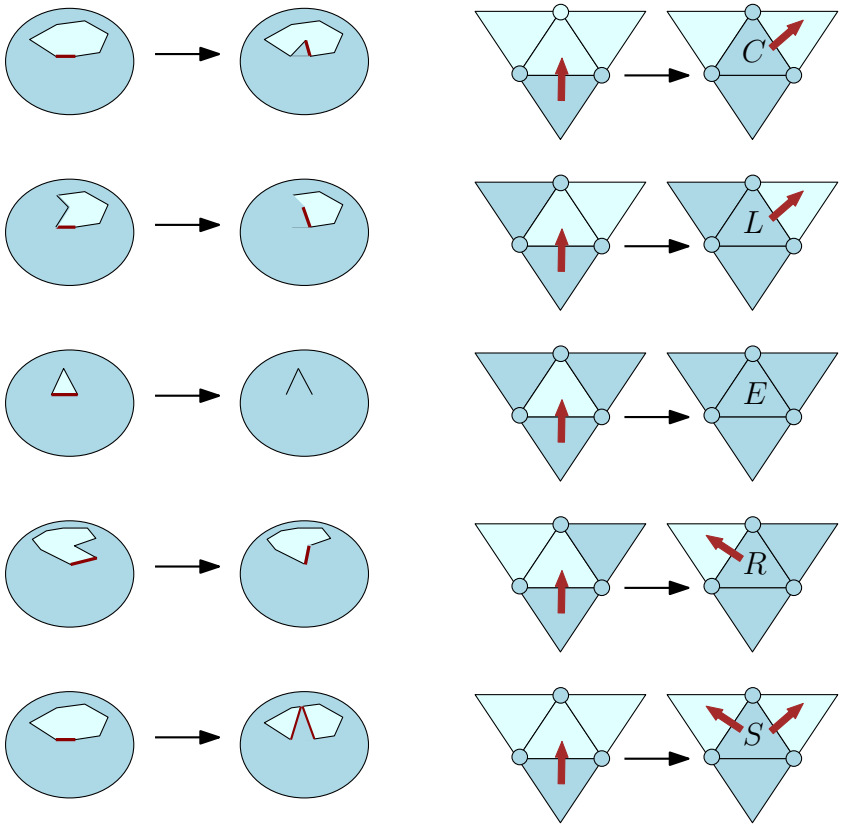
$3.24n$ bits



1101000110000010010000011001000000000

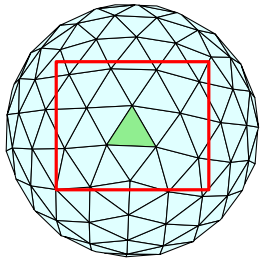
How to encode a triangulation with at most $4n$ bits

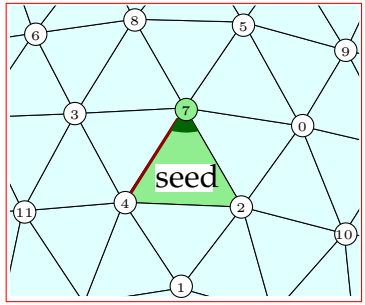
Edgebreaker (Rossignac, '99)



How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)

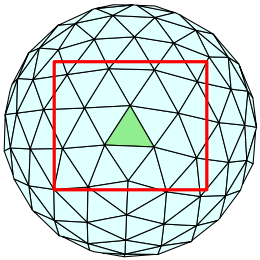


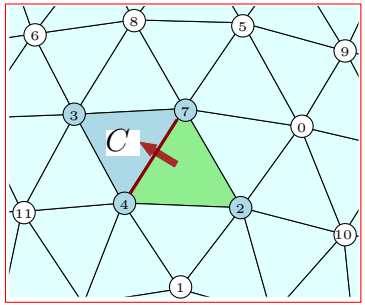


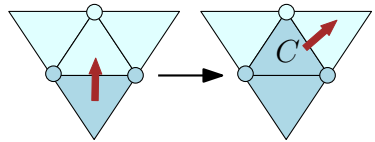
Let us start by choosing a root face and *gate* edge to start the mesh traversal

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



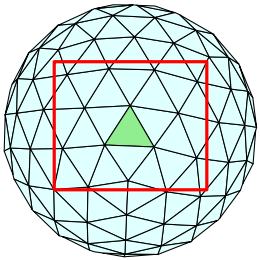


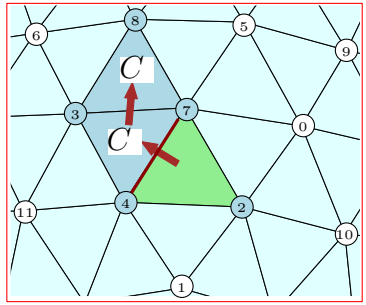


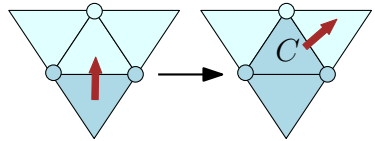
Peform a (right-most) depth traversal of the unvisited faces

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



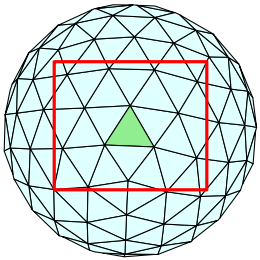


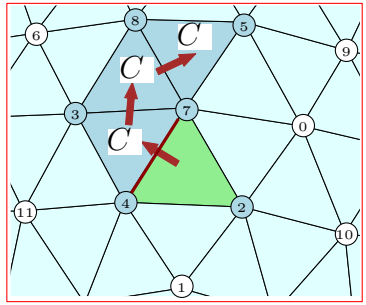


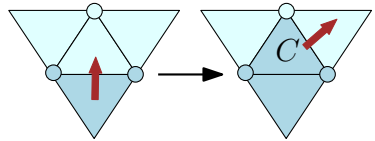
Peform a (right-most) depth traversal of the unvisited faces

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



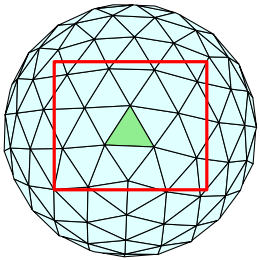


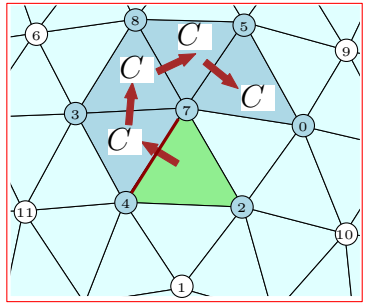


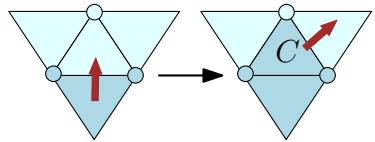
Peform a (right-most) depth traversal of the unvisited faces

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



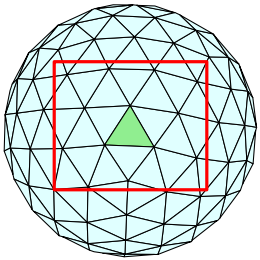


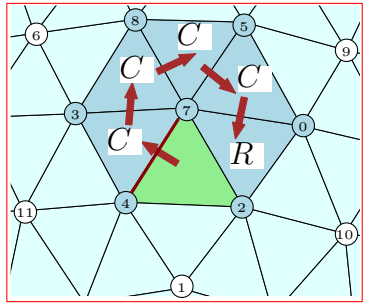


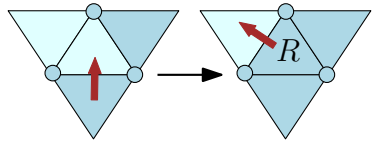
Peform a (right-most) depth traversal of the unvisited faces

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



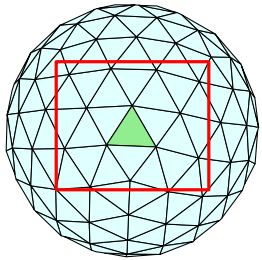


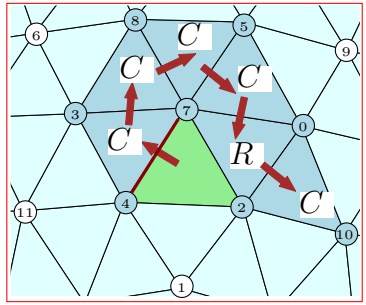


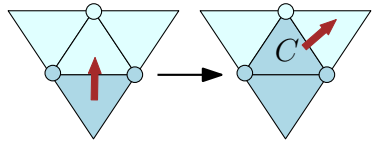
Peform a (right-most) depth traversal of the unvisited faces

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



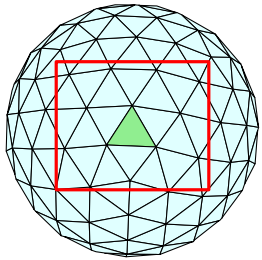


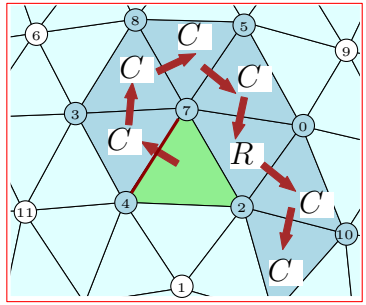


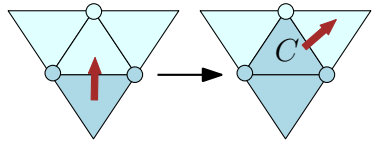
Peform a (right-most) depth traversal of the unvisited faces

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



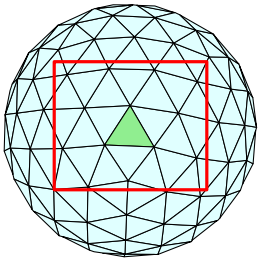


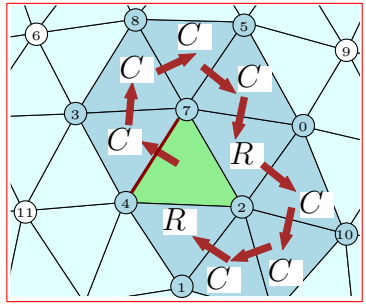


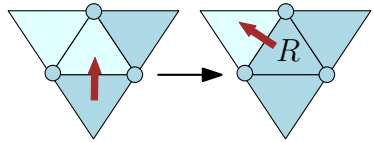
Peform a (right-most) depth traversal of the unvisited faces

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



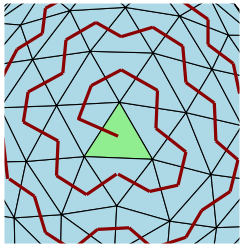


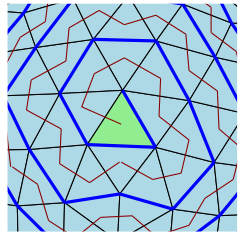


Peform a (right-most) depth traversal of the unvisited faces

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



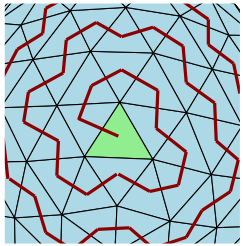


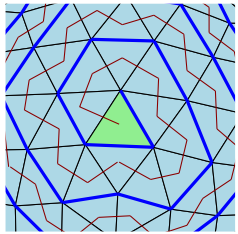
The traversal ends when all faces are visited

As result, we obtain a dual spaning tree (red), and a vertex spanning tree of the primal graph (blue edges): these two trees are not intersecting

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)



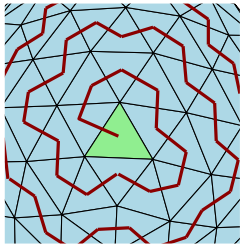


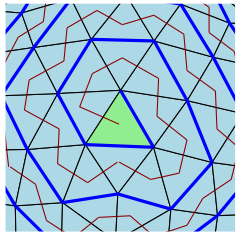
$$CCCCRCCCCRCCCR\ldots$$

The encoding is given by a word of length $f - 1 = 2n - 5$
on the alphabet $\{C, L, E, R, S\}$

How to encode a triangulation with at most $4n$ bits

Edgebreaker (Rossignac, '99)





$|C| = n - 3$
(a C symbol per vertex)

$|L| + |E| + |R| + |S| = f - |C| =$
$(2n - 4) - |C| \approx n$

$$CCCCRCCCCRCCCR\ldots$$

The word above can be encoded by a binary word of length $4n$

- use 1 bit to encode C
- use 3 bits to encode remaining symbols $\{L, E, R, S\}$

the total length is at most $1 \cdot (n - 3) + 3n \approx 4n$