中国科学技术大学
University of Science and Technology of China

GAMES102

GAMES 102在线课程

几何建模与处理基础

刘利刚

中国科学技术大学

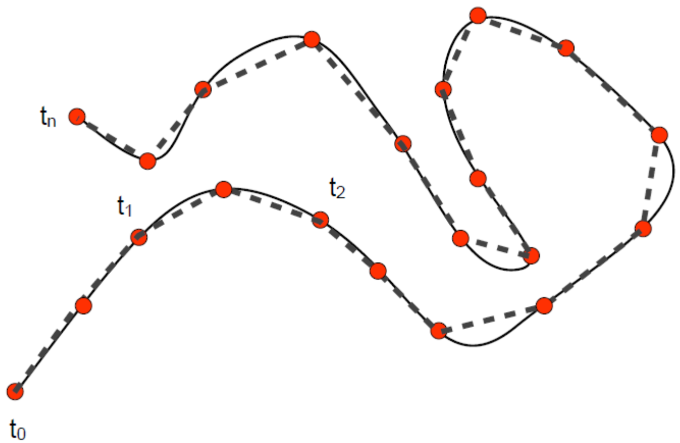
中国科学技术大学
University of Science and Technology of China

GAMES 102在线课程：几何建模与处理基础

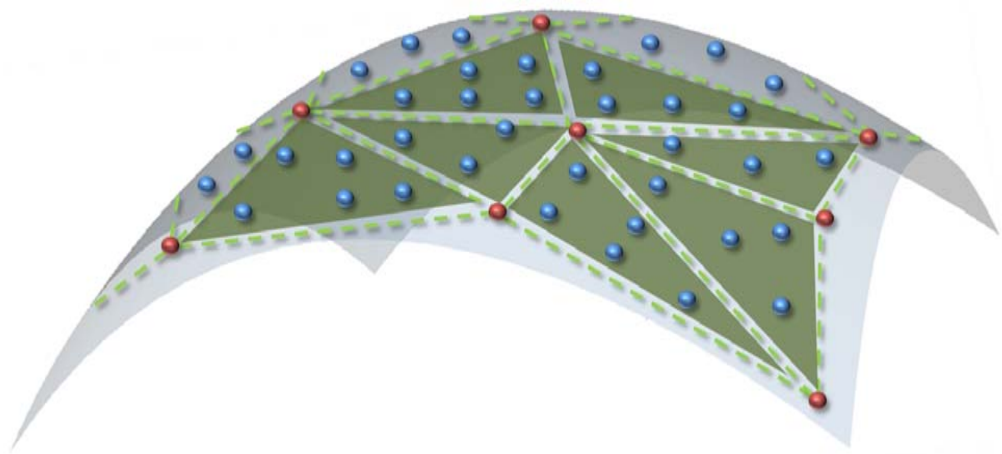
三角网格：
曲面的离散表达

绘制：离散表达

- 曲线的绘制：
  - GDI/OpenGL 绘制基本单元：点、线段
  - 曲线须离散成多边形
- 曲面的绘制：
  - OpenGL 绘制基本单元：点、线、三角形
  - 曲面须离散成三角形网格

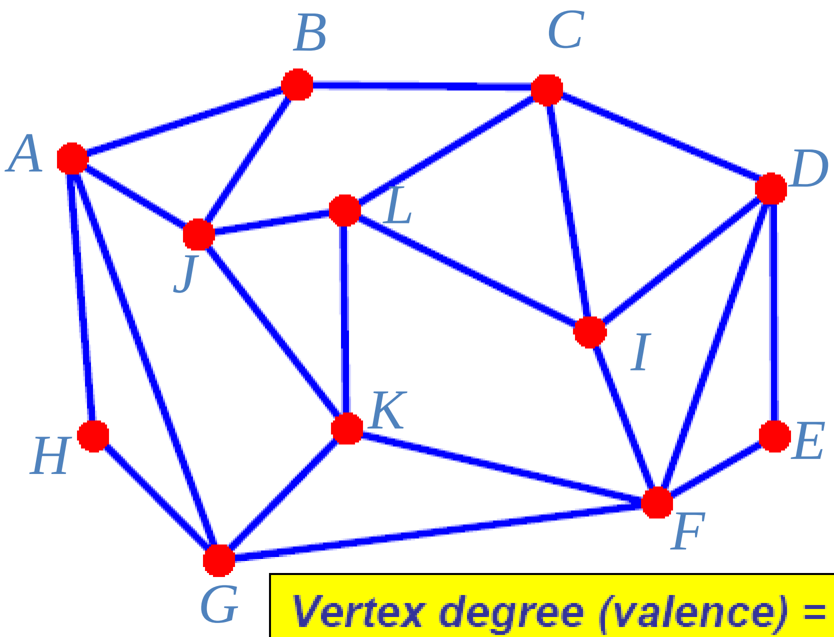


曲线的离散



曲面的离散

Standard Graph Definition



G = <**V**,**E**>
V = vertices =
{A,B,C,D,E,F,G,H,I,J,K,L}
E = edges =
{(A,B),(B,C),(C,D),(D,E),(E,F),(F,G),
(G,H),(H,A),(A,J),(A,G),(B,J),(K,F),
(C,L),(C,I),(D,I),(D,F),(F,I),(G,K),
(J,L),(J,K),(K,L),(L,I)}

Vertex degree (valence) = number of edges incident on vertex
deg(J) = 4, deg(H) = 2
k-regular graph = graph whose vertices all have degree *k*

Face: cycle of vertices/edges which cannot be shortened
F = faces =
{(A,H,G),(A,J,K,G),(B,A,J),(B,C,L,J),(C,I,J),(C,D,I),
(D,E,F),(D,I,F),(L,I,F,K),(L,J,K),(K,F,G)}

Connectivity

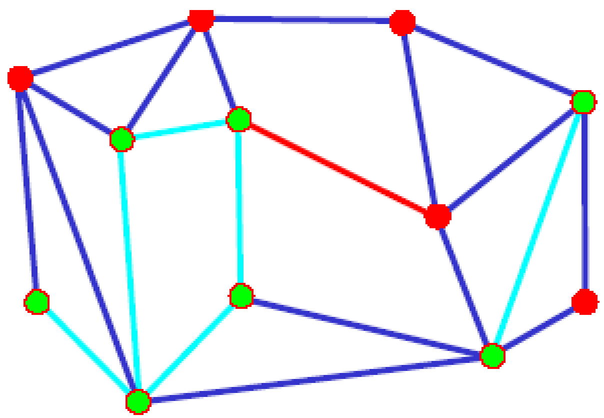
Graph is ***connected*** if there is a path of edges connecting every two vertices

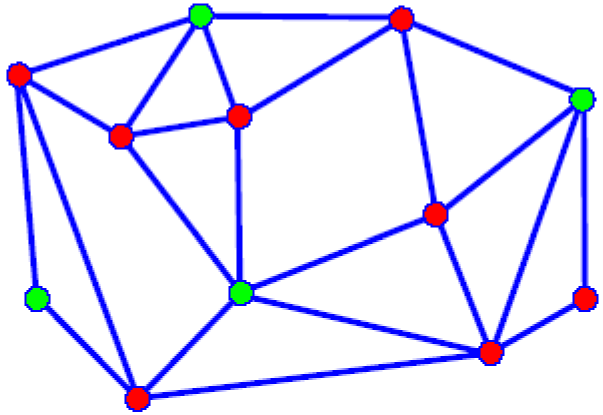
Graph is ***k-connected*** if between every two vertices there are *k* edge-disjoint paths

Graph **G’=<V’,E’>** is a ***subgraph*** of graph **G=<V,E>** if **V**’ is a subset of **V** and **E’** is the subset of **E** incident on **V’**

Connected component of a graph: maximal connected subgraph

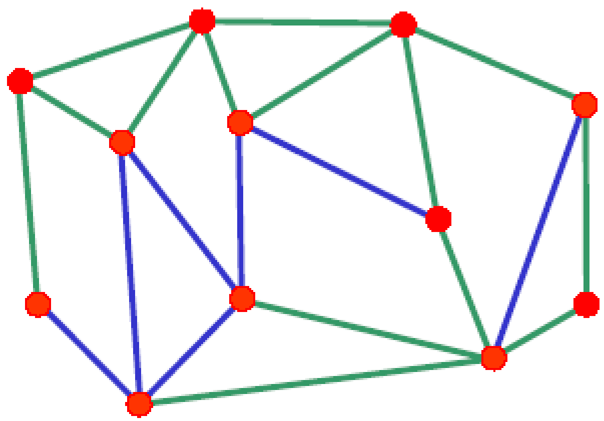
Subset **V’** of **V** is an ***independent*** set in **G** if the subgraph it induces does not contain any edges of **E**



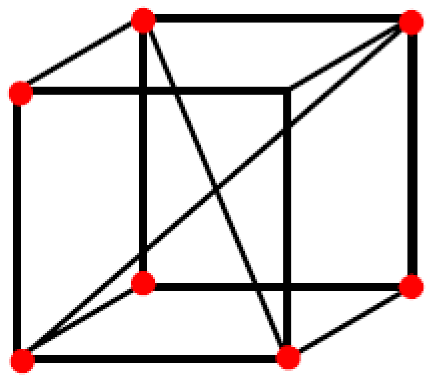


Graph Embedding

Graph is ***embedded*** in R^d if each vertex is assigned a position in R^d



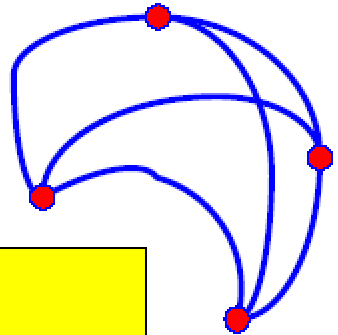
Embedding in R^2



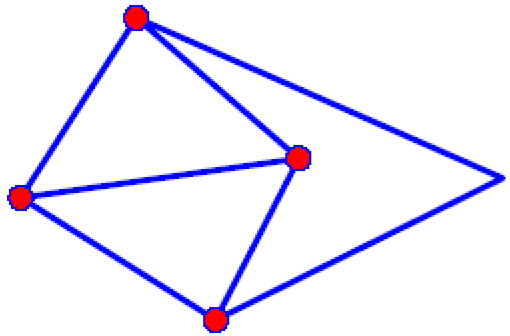
Embedding in R^3

Planar Graphs

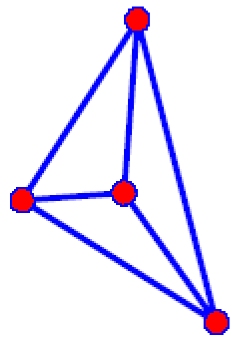
Planar Graph



Plane Graph



Straight Line Plane Graph



Planar graph: graph whose vertices and edges can be embedded in R^2 such that its edges do not intersect

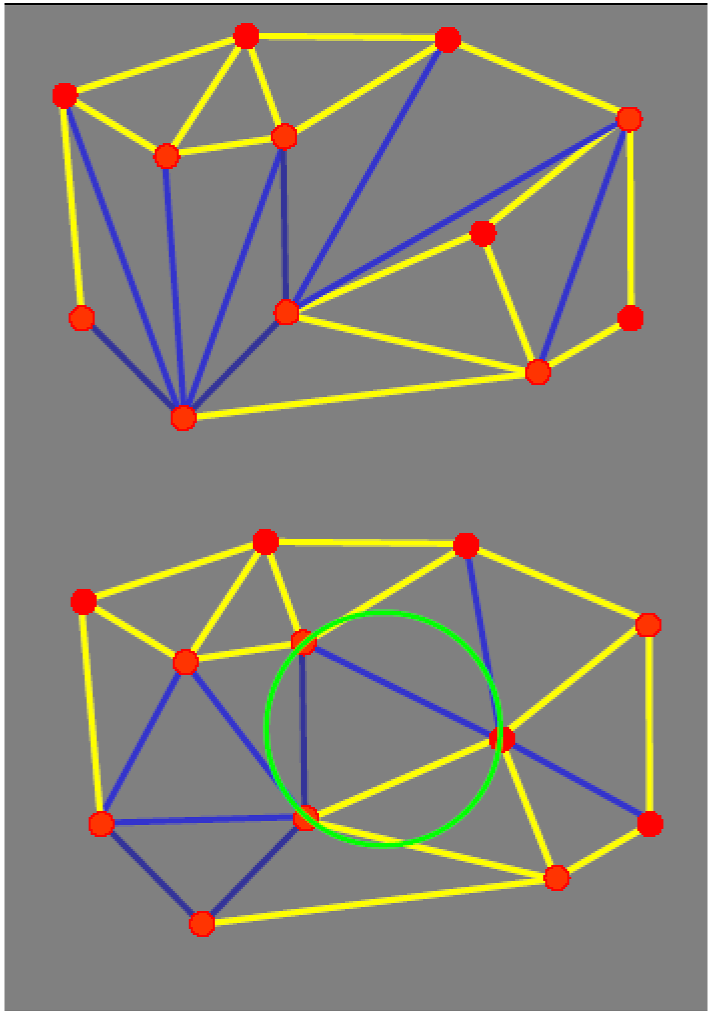
Every planar graph can be drawn as a ***straight-line plane graph***

Triangulation

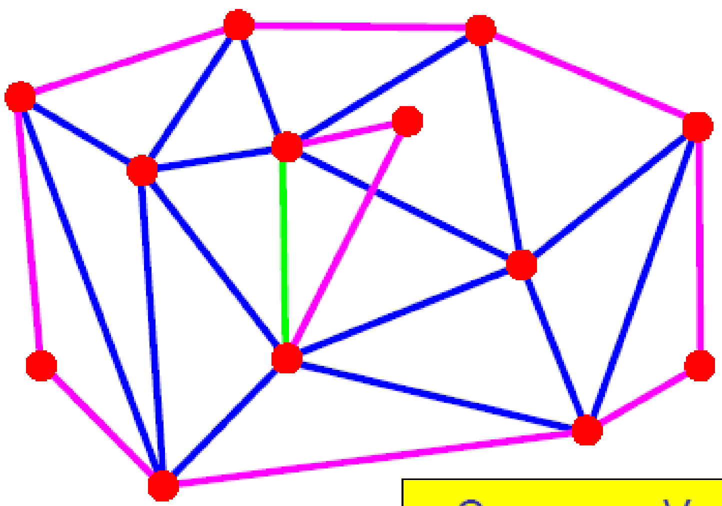
Triangulation: straight line plane graph all of whose faces are triangles

Delaunay triangulation of a set of points: unique set of triangles such that the circumcircle of any triangle does not contain any other point

Delaunay triangulation avoids long and skinny triangles



Meshes

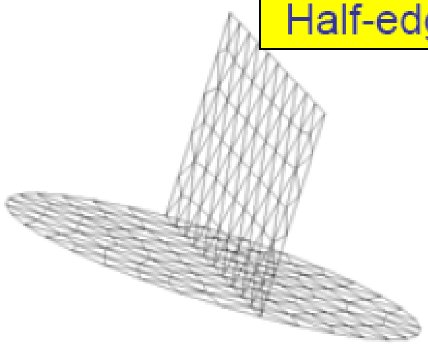


Mesh: straight-line graph embedded in R^3

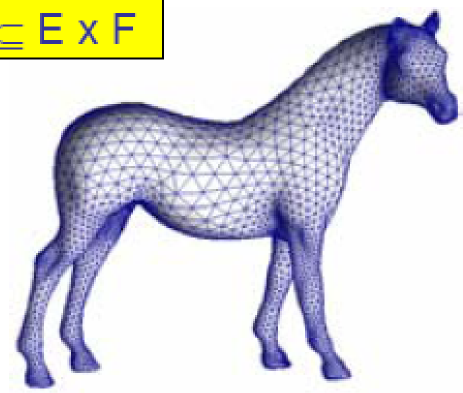
Boundary edge: adjacent to exactly *one* face
Regular edge: adjacent to exactly *two* faces
Singular edge: adjacent to more than two faces

Closed mesh: mesh with no boundary edges
Manifold mesh: mesh with no singular edges

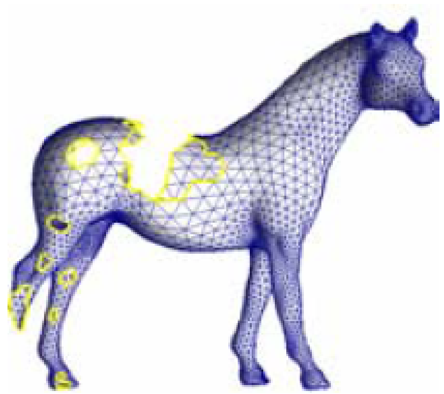
Corners $\subseteq$ V x F
Half-edges $\subseteq$ E x F



Non-Manifold

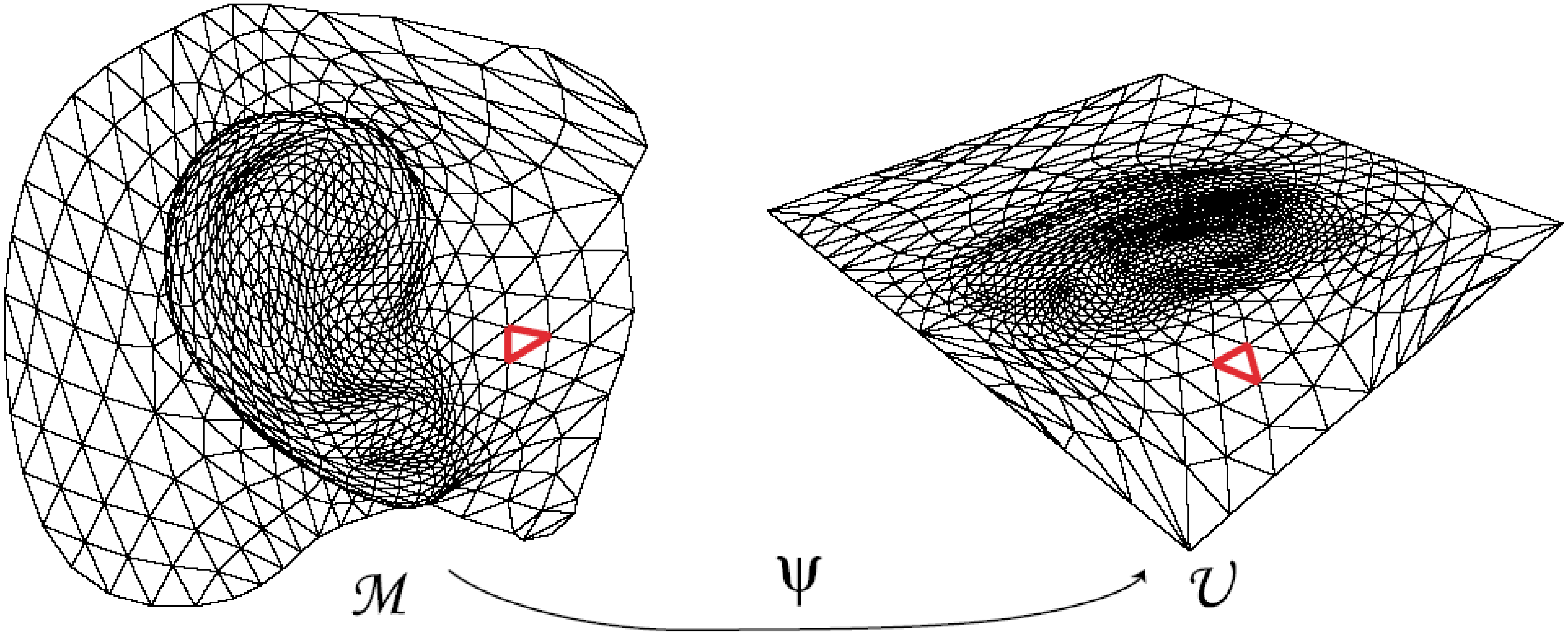


Closed Manifold

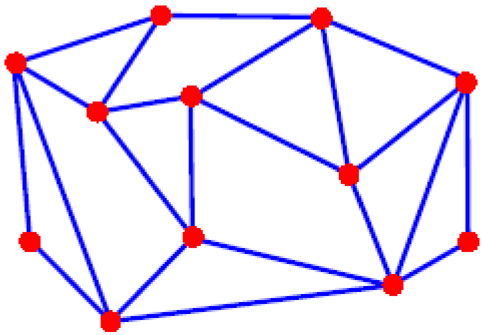


Open Manifold

Planar Graphs and Meshes



Topology



v =12
f = 14
e = 25
c = 1
g = 0
b = 1

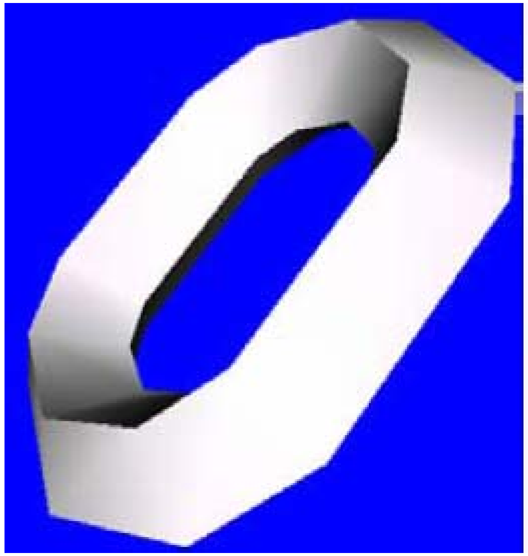
Genus of graph: *half* of *maximal* number of closed paths that do *not* disconnect the graph (number of "holes")

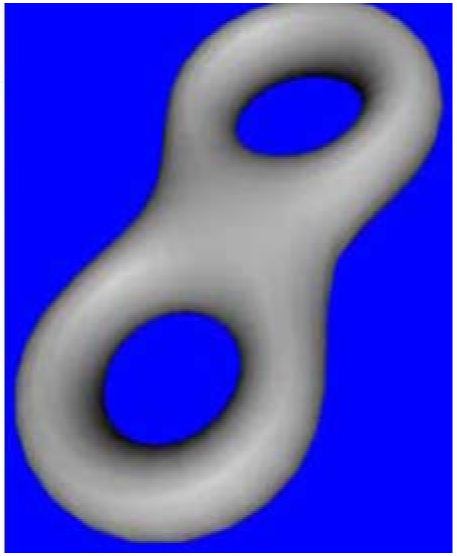
Genus(sphere)= 0
Genus(torus) = 1

Euler-Poincare Formula

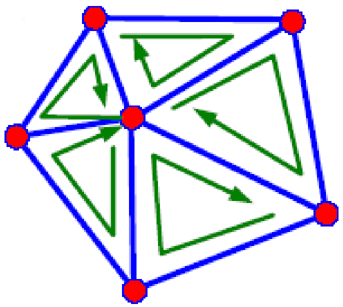
$$v+f-e = 2(c-g)-b$$

v = # vertices c = # conn. comp
f = # faces g = genus
e = # edges b = # boundaries





Orientability



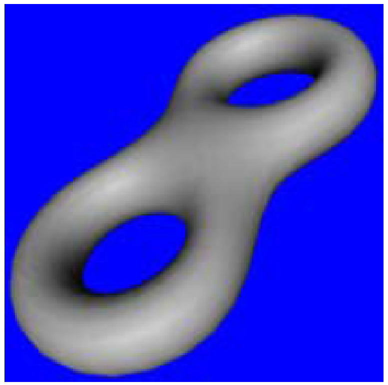
Oriented
F={(L,J,B),(B,C,L),(L,C,I), (I,K,L),(L,K,J)}
Not Oriented
F={(B,J,L),(B,C,L),(L,C,I), (L,I,K),(L,K,J)}

Orientation of a face is clockwise or anticlockwise order in which its vertices and edges are listed

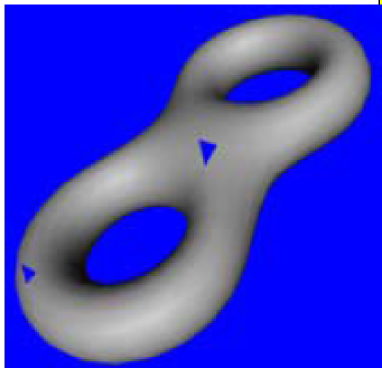
This defines the direction of face ***normal***

Straight line graph is ***orientable*** if orientations of its faces can be chosen so that each edge is oriented in *both* directions

Mobius strip or Klein bottle - not orientable

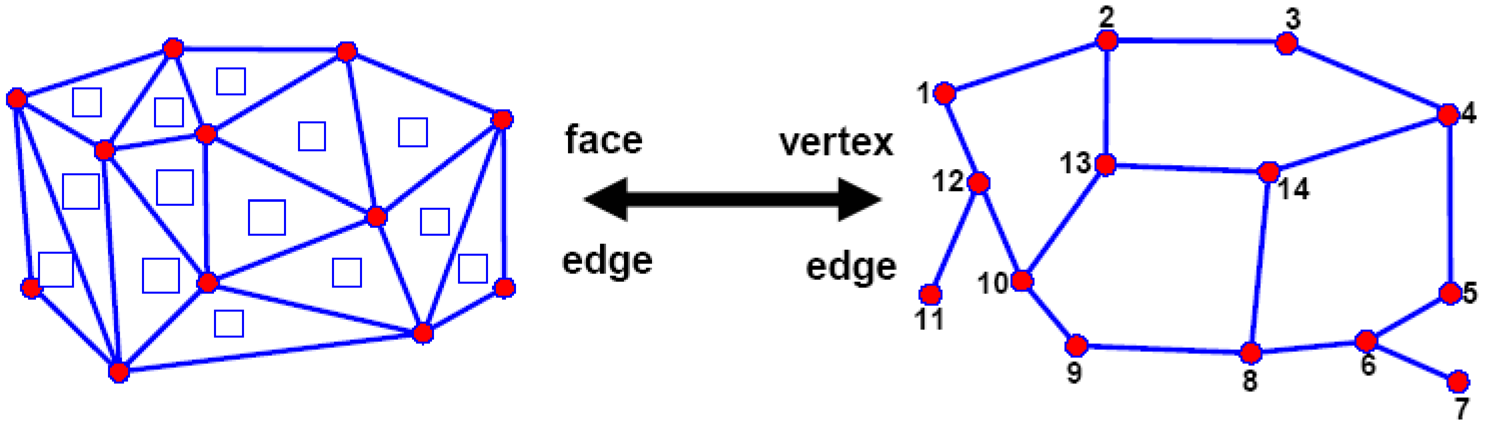


Not Backface Culled



Backface Culled

Duality



- Delaunay Triangulation vs. Voronoi Graph

网格曲面的数据结构

Uses of Mesh Data

- Rendering
  - Triangle trip
- Geometry queries
  - What are the vertices of face *#k*?
  - Are vertices *#i* and *#j* adjacent?
  - Which faces are adjacent face *#k*?
- Geometry operations
  - Remove/add a vertex/face
  - Mesh simplification
  - Vertex split, edge collapse

Storing Mesh Data (1)

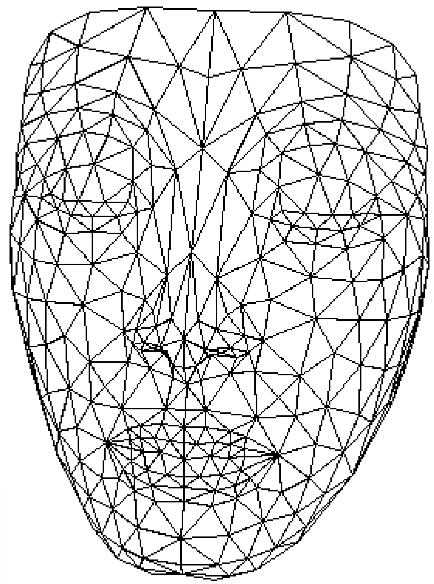
- Storage of generic meshes
  - Hard to implement efficiently
- Assume
  - Triangular
  - Orientable
  - Manifold

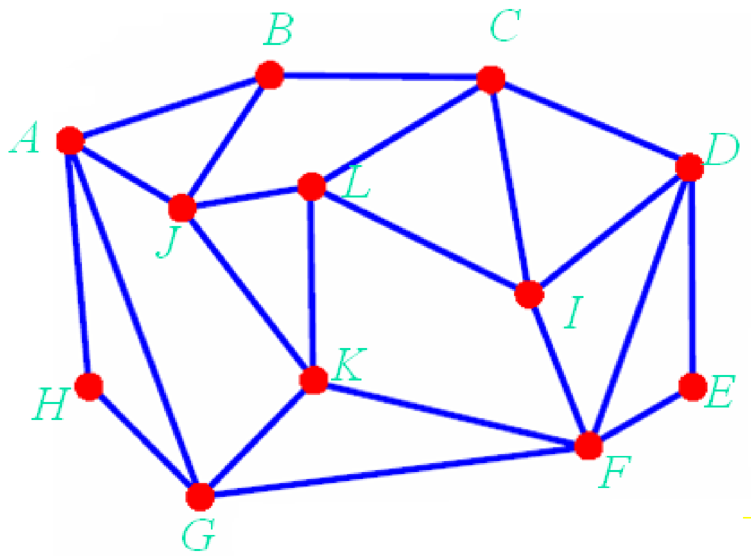
Storing Mesh Data (2)

- How "good" is a data structure?
  - Space complexity
  - Time
    - Time to construct - preprocessing
    - Time to answer a query
    - Time to perform an operation (update the data structure)
  - Trade-off between time and space
  - Redundancy

Define a Mesh (1)

- Geometry
  - Vertex coordinates
- Connectivity
  - How do vertices connected?





Define a Mesh (2)

- List of Edge
- Vertex-Edge
- Vertex-Face
- Combined

3D Mesh Surface

- Surface & material properties
  - Material color
  - Ambient, hightlight coefficients
  - Texture coordinates
  - BRDF, BTF
- Rendering properties
  - Lighting
  - Normals
  - Rendering modes

General Used Mesh Files

- General used mesh files
  - Wavefront OBJ (*.obj)
  - 3D Max (*.max, *.3ds)
  - VRML(*.vrl)
  - Inventor (*.iv)
  - PLY (*.ply, *.ply2)
  - User-defined(*.m, *.liu)
- Storage
  - Text – (Recommended)
  - Binary

Wavefront OBJ File Format

- Vertices
  - Start with char ‘v’
  - (x,y,z) coordinates
- Faces
  - Start with char ‘f’
  - Indices of its vertices in the file
- Other properties
  - Normal, texture coordinates, material, etc.

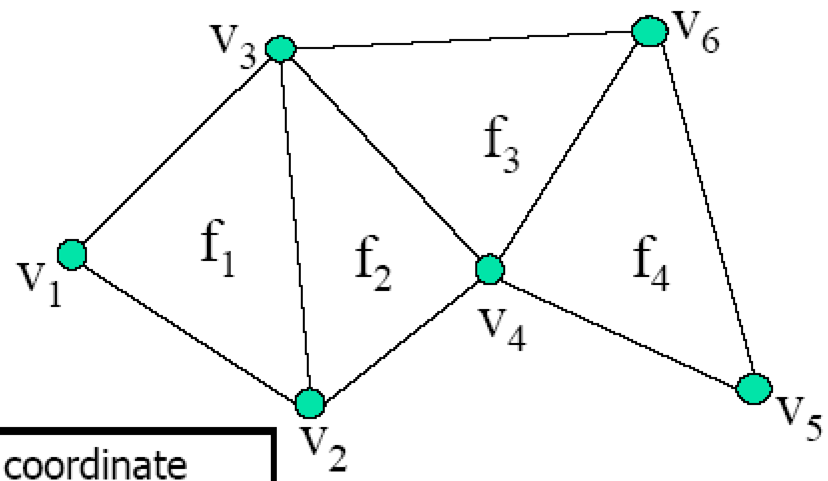
```
v 1.0 0.0 0.0
v 0.0 1.0 0.0
v 0.0 -1.0 0.0
v 0.0 0.0 1.0
f 1 2 3
f 1 4 2
f 3 2 4
f 1 3 4
```

List of Faces

List of Faces

- List of vertices
  - Position coordinates
- List of faces
  - Triplets of pointers to face vertices (c1,c2,c3)
- Queries:
  - What are the vertices of face #3?
    - Answered in O(1) - checking third triplet
  - Are vertices i and j adjacent?
    - A pass over all faces is necessary – NOT GOOD

List of Faces – Example



vertex	coordinate
v_1	(x_1,y_1,z_1)
v_2	(x_2,y_2,z_2)
v_3	(x_3,y_3,z_3)
v_4	(x_4,y_4,z_4)
v_5	(x_5,y_5,z_5)
v_6	(x_6,y_6,z_6)

face	vertices (ccw)
f_1	(v_1, v_2, v_3)
f_2	(v_2, v_4, v_3)
f_3	(v_3, v_4, v_6)
f_4	(v_4, v_5, v_6)

List of Faces – Analysis

- Pros:
  - Convenient and efficient (memory wise)
  - Can represent non-manifold meshes
- Cons:
  - Too simple - not enough information on relations between vertices & faces

Adjacency Matrix

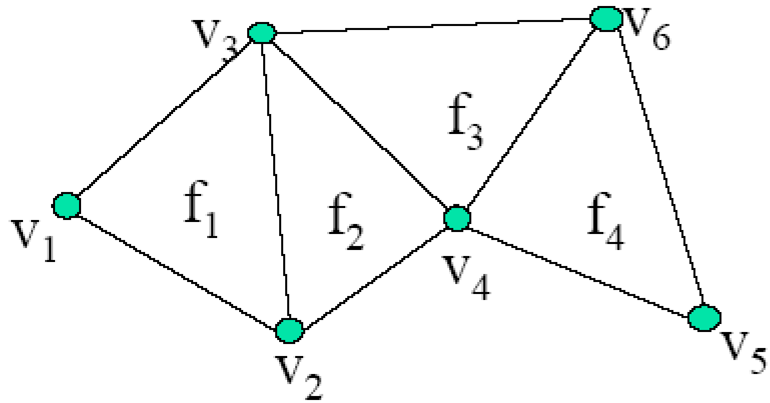
Adjacency Matrix – Definition

- View mesh as connected graph
- Given n vertices build n*n matrix of adjacency information
  - Entry (i,j) is TRUE value if vertices i and j are adjacent
- Geometric info
  - list of vertex coordinates
- Add faces
  - list of triplets of vertex indices (v1,v2,v3)

Adjacency Matrix – Example

vertex	coordinate
v_1	(x_1,y_1,z_1)
v_2	(x_2,y_2,z_2)
v_3	(x_3,y_3,z_3)
v_4	(x_4,y_4,z_4)
v_5	(x_5,y_5,z_5)
v_6	(x_6,y_6,z_6)

face	vertices (ccw)
f_1	(v_1, v_2, v_3)
f_2	(v_2, v_4, v_3)
f_3	(v_3, v_4, v_6)
f_4	(v_4, v_5, v_6)



	v_1	v_2	v_3	v_4	v_5	v_6
v_1		1	1			
v_2	1		1	1		
v_3	1	1		1		1
v_4		1	1		1	1
v_5				1		1
v_6			1	1	1	

Adjacency Matrix – Queries

- What are the vertices of face #3?
  - O(1) – checking third triplet of faces
- Are vertices i and j adjacent?
  - O(1) - checking adjacency matrix at location (i,j).
- Which faces are adjacent to vertex j?
  - Full pass on all faces is necessary

Adjacency Matrix – Analysis

- Pros:
  - Information on vertices adjacency
  - Stores non-manifold meshes
- Cons:
  - Connects faces to their vertices, BUT NO connection between vertex and its face

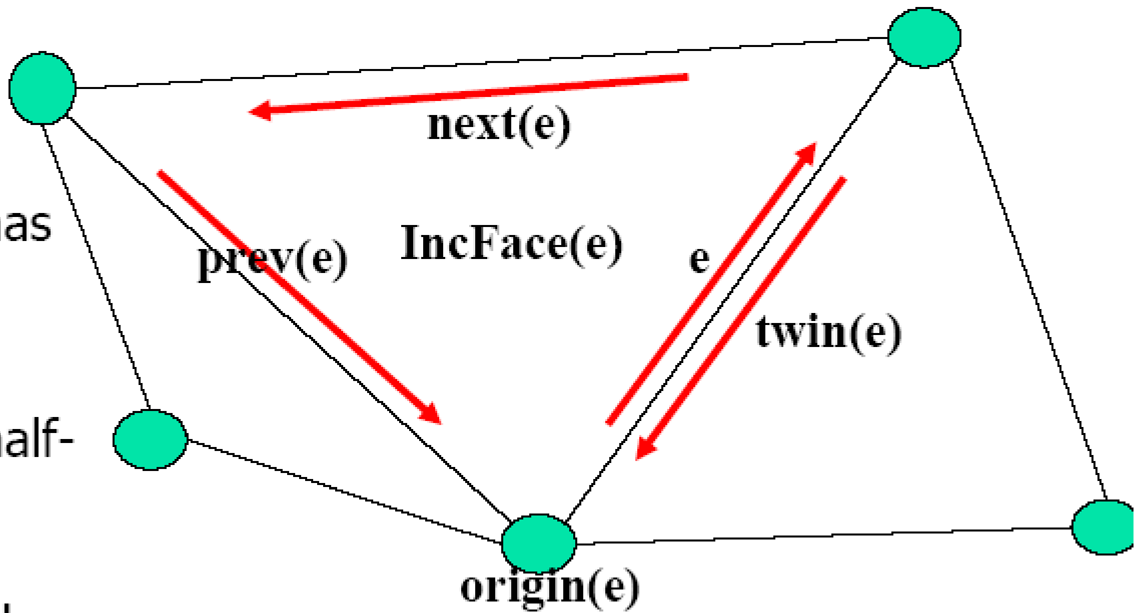
Doubly-Connected Edge List (DCEL)

DCEL

- Record for each face, edge and vertex:
  - Geometric information
  - Topological information
  - Attribute information
- Half-Edge Structure

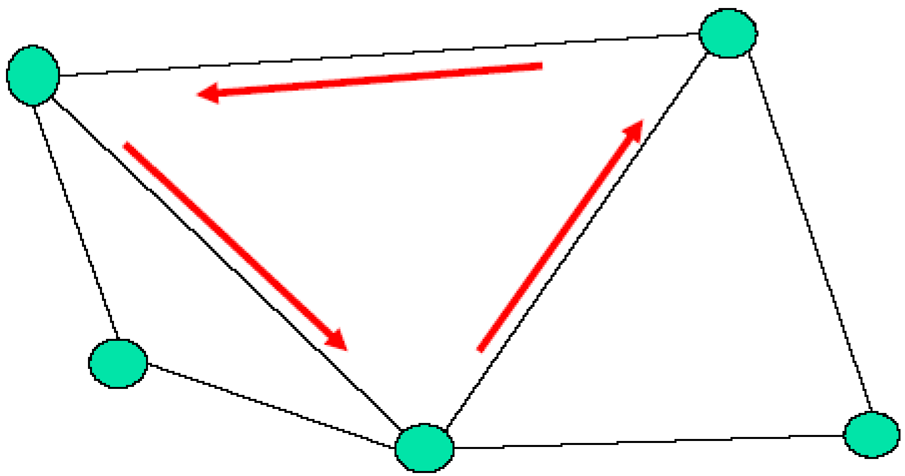
DCEL (cont.)

- Vertex record:
  - Coordinates
  - Pointer to one half-edge that has v as its origin
- Face record:
  - Pointer to one half-edge on its boundary
- Half-edge record:
  - Pointer to its origin, origin(e)
  - Pointer to its twin half-edge, twin(e)
  - Pointer to the face it bounds, IncidentFace(e) (face lies to left of e when traversed from origin to destination)
  - Next and previous edge on boundary of IncidentFace(e)

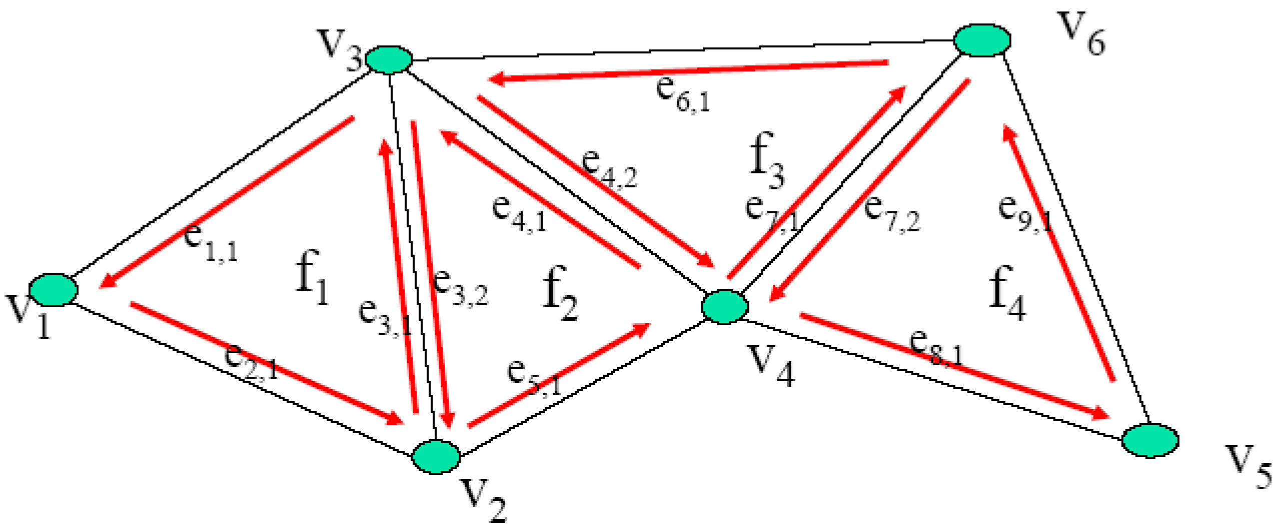


DCEL(cont.)

- Operations supported:
  - Walk around boundary of given face
  - Visit all edges incident to vertex v
- Queries:
  - Most queries are O(1)



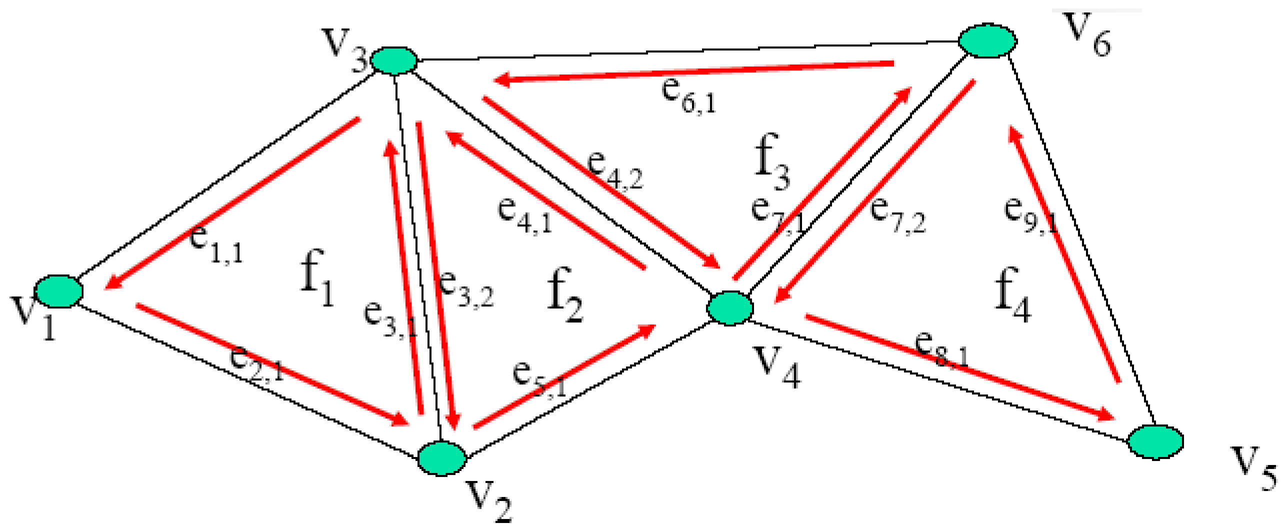
DCEL – Example



Vertex	coordinate	IncidentEdge
V_1	(x_1,y_1,z_1)	$e_{2,1}$
v_2	(x_2,y_2,z_2)	$e_{5,1}$
v_3	(x_3,y_3,z_3)	$e_{1,1}$
v_4	(x_4,y_4,z_4)	$e_{7,1}$
v_5	(x_5,y_5,z_5)	$e_{9,1}$
v_6	(x_6,y_6,z_6)	$e_{7,2}$

face	edge
f_1	$e_{1,1}$
f_2	$e_{5,1}$
f_3	$e_{4,2}$
f_4	$e_{8,1}$

DCEL – Example (cont.)



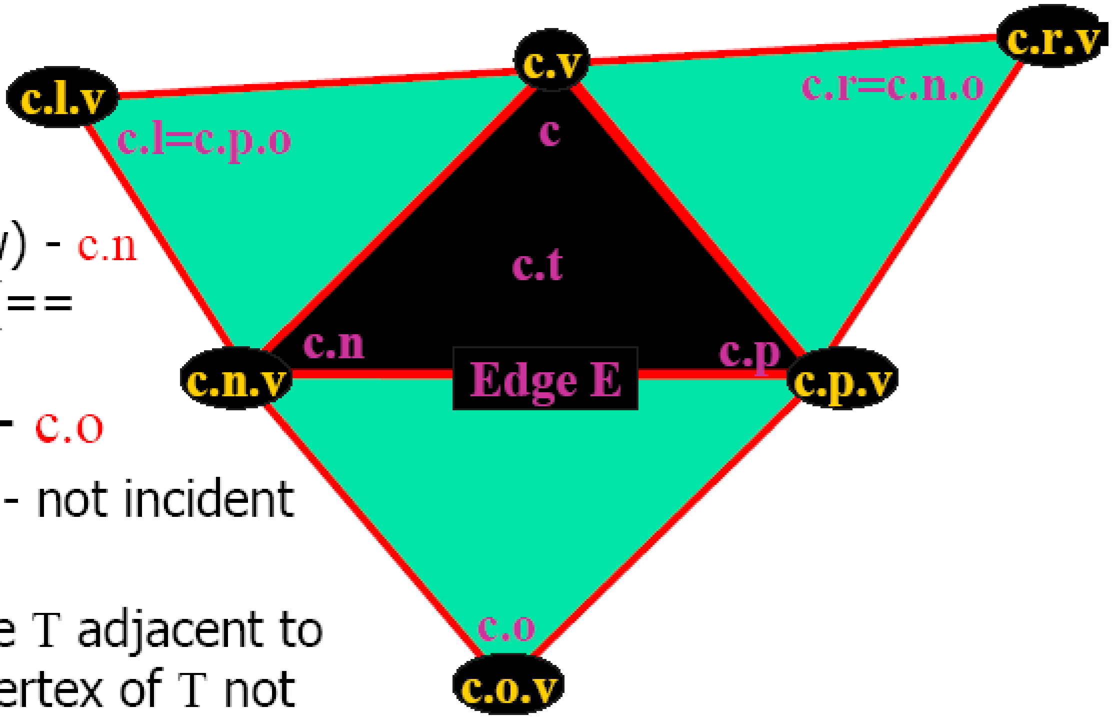
Half-edge	origin	twin	IncidentFace	next	prev
$e_{3,1}$	v_2	$e_{3,2}$	f_1	$e_{1,1}$	$e_{2,1}$
$e_{3,2}$	v_3	$e_{3,1}$	f_2	$e_{5,1}$	$e_{4,1}$
$e_{4,1}$	v_4	$e_{4,2}$	f_2	$e_{3,2}$	$e_{5,1}$
$e_{4,2}$	v_3	$e_{4,1}$	f_3	$e_{7,1}$	$e_{6,1}$

DCEL – Analysis

- Pros
  - All queries in O(1) time
  - All operations are O(1) (usually)
- Cons
  - Represents only manifold meshes

Corner Table

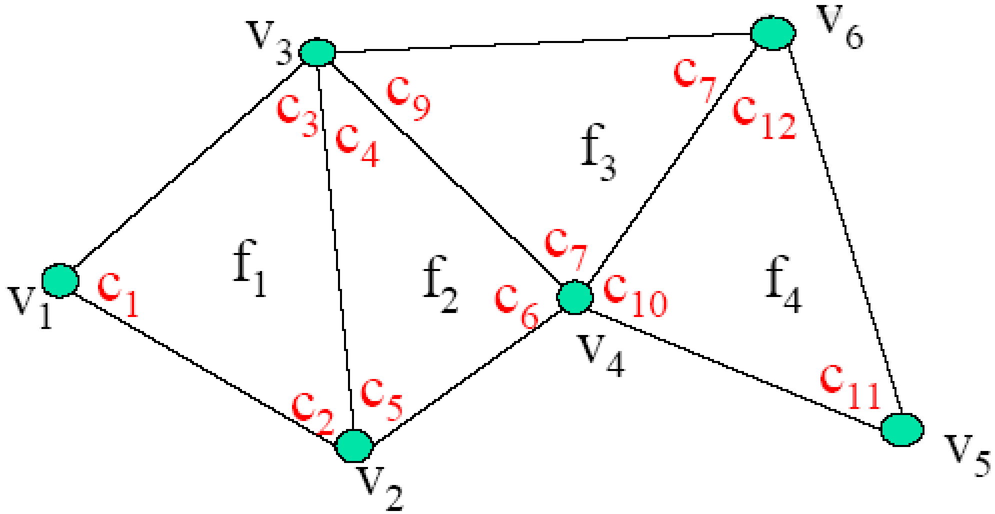
Corner : Coupling of vertex with one of its incident triangles



Corner c contains:

- Triangle – c.t
- Vertex – c.v
- Next corner in c.t (ccw) - c.n
- Previous corner - c.p (== c.n.n)
- Corner opposite c - c.o
  - E edge opposite c - not incident on c.v
  - c.o couples triangle T adjacent to c.t across E with vertex of T not incident on E
- Right corner - c.r - corner opposite c.n (== c.n.o).
- Left corner - c.l (== c.p.o == c.n.n.o)

Corner Table – Example



corner	c.v	c.t	c.n	c.p	c.o	c.r	c.l
c_1	v_1	f_1	c_2	c_3	c_6	NULL	NULL
c_2	v_2	f_1	c_3	c_1	NULL	NULL	c_6
c_3	v_3	f_1	c_1	c_2	NULL	c_6	NULL
c_4	v_3	f_2	c_5	c_6	NULL	c_7	c_1
c_5	v_2	f_2	c_6	c_4	c_7	c_1	NULL
c_6	v_4	f_2	c_4	c_5	c_1	NULL	c_7

Example Queries

- What are the vertices of face #3?
  - Check c.v of corners 9, 10, 11
- Are vertices i and j adjacent?
  - Scan all corners of vertex i, check if c.p.v or c.n.v are j
- Which faces are adjacent to vertex j?
  - Check c.t of all corners of vertex j

Corner Table – Analysis

- Pros
  - All queries in O(1) time
  - All operations are O(1) (usually)
- Cons
  - Represents only manifold meshes
  - High redundancy (but not too high …)

Practice with Utopia

- 半边(half-edge)数据结构

- 两种数据结构
  - 计算数据
  - 渲染数据

- 小练习： Mesh smoothing
  - 计算更新顶点坐标即可

中国科学技术大学
University of Science and Technology of China

谢 谢 ！