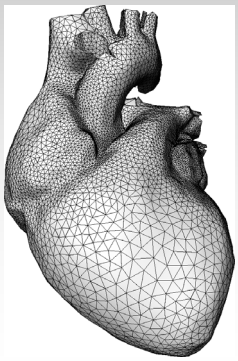
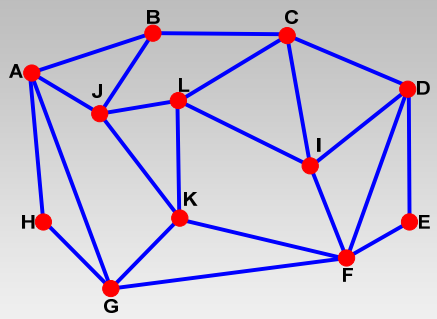
Mesh Basics: Definitions, Topology & Data Structures



Standard Graph Definitions



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,L),(C,D,I),
(D,E,F),(D,I,F),(L,I,F,K),(L,J,K),(K,F,G)}

Connectivity

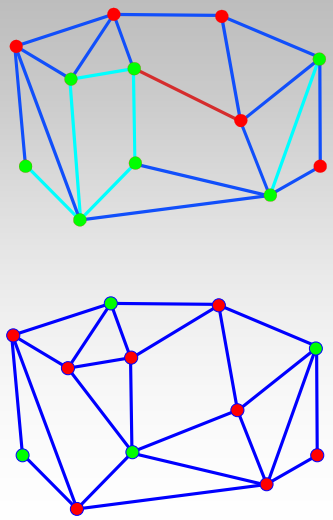
Graph is ***connected*** if there is a path of edges connecting every two vertices

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'**

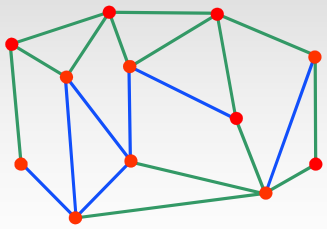
Connected component of a graph: maximal connected subgraph

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

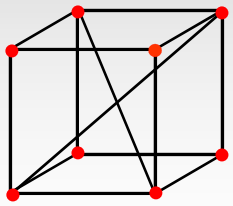


Graph Embedding

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

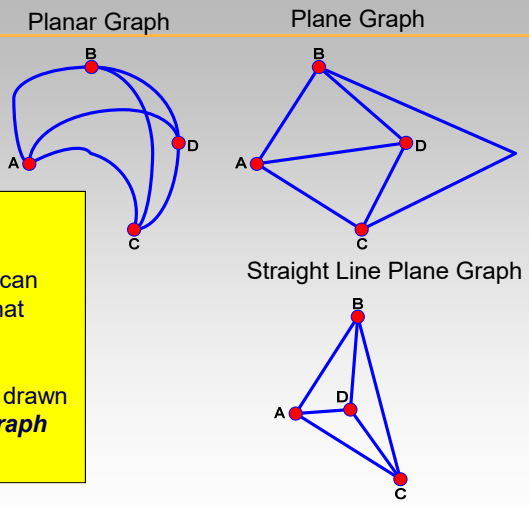


Embedding in R^2



Embedding in R^3

Planar Graphs



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

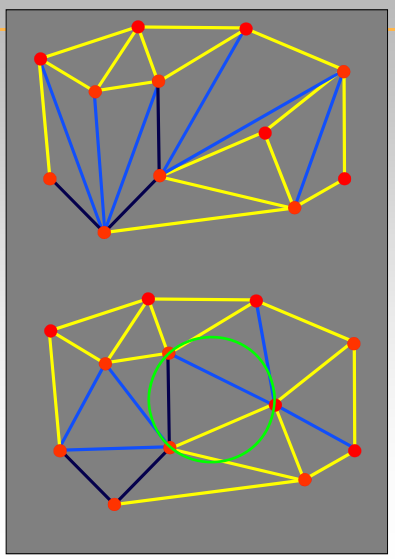
Every planar graph can be drawn as a ***straight-line plane graph***

Triangulation

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

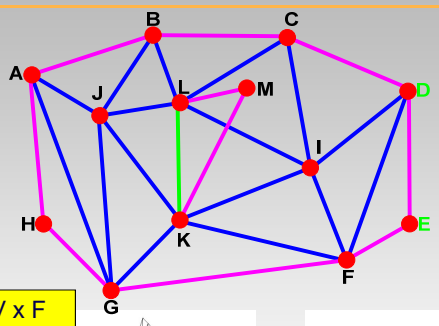
Delaunay triangulation of a set of points: unique set of triangles such that the circumcircle of any triangle does not contain any other point

Delaunay triangulation avoids long and skinny triangles



6

Meshes

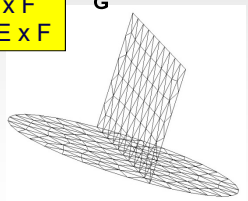


Mesh: straight-line graph embedded in R^3

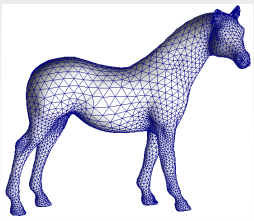
Boundary edge: adjacent to exactly *one* face
Regular edge: adjacent to exactly *two* faces
Singular edge: adjacent to more than two faces

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

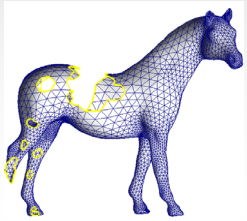
Corners $\subseteq$ V x F
Half-edges $\subseteq$ E x F



Non-Manifold



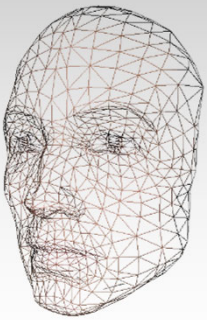
Closed Manifold

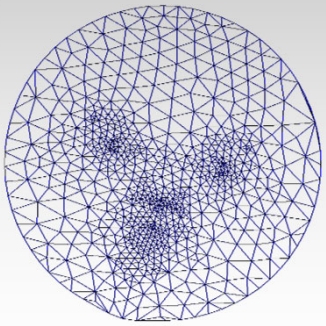


Open Manifold

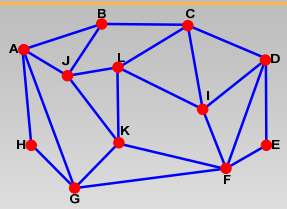
Planar Graphs and Meshes

Most **manifold meshes are planar !!**





Topology



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

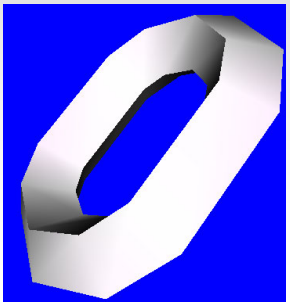
Genus of graph: *half* of *maximal* number of closed paths that do *not* disconnect the graph (number of "holes")

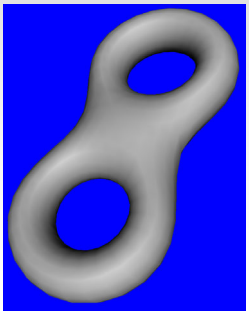
Genus(sphere)= 0
Genus(torus) = 1

Euler-Poincare Formula

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

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

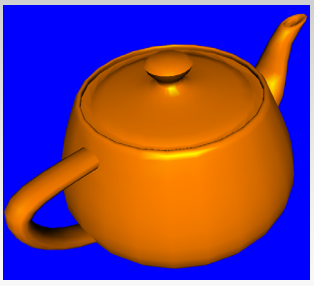


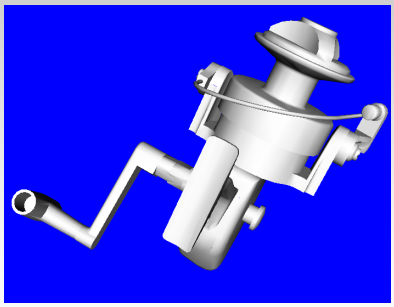


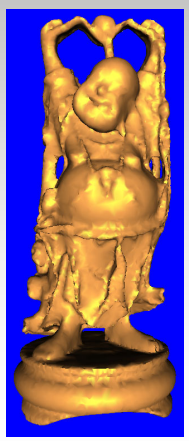
© Alla Sheffer

Topology Quiz

What can you say about the genus of these meshes ?







10

© Alla Sheffer

Exercises

Theorem: Average vertex degree in closed manifold triangle mesh is ~6

Proof: In such a mesh, f = 2e/3
By Euler's formula: v+2e/3-e = 2-2g
hence e = 3(v-2+2g) and f = 2(v-2+2g)

So Average(deg) = 2e/v = 6(v-2+2g)/v
~ 6 for large v

Corollary: Only toroidal (g=1) closed manifold triangle mesh can be regular (all vertex degrees are 6)

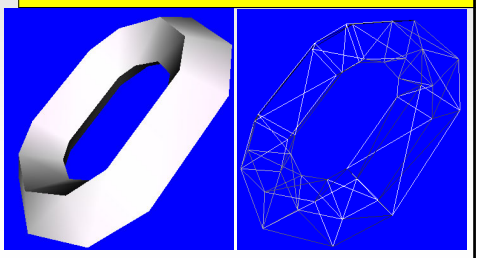
Proof: In regular mesh average degree is *exactly* 6
Can happen only if g=1

Theorem: In closed manifold mesh: $2e \geq 3f$ (equality for triangle mesh), $2e \geq 3v$

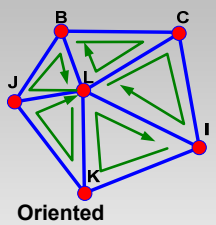
Corollary: No closed manifold trianglemesh can have 7 edges

Corollary: $2f-4 \geq v$

Does Euler's theorem imply that any planar graph has an independent set of size at least ¼ n ?



Orientability

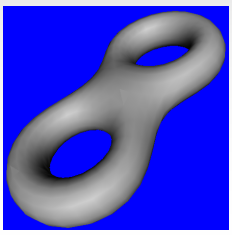


Oriented
F={(L,J,B),(B,C,L),(L,C,I), (I,K,L),(L,K,J)}
Not Oriented
F={(B,J,L),(B,C,L),(L,C,I), (L,I,K),(L,K,J)}

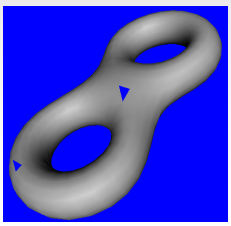
Orientation of a face is clockwise or anticlockwise order in which its vertices and edges are listed

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



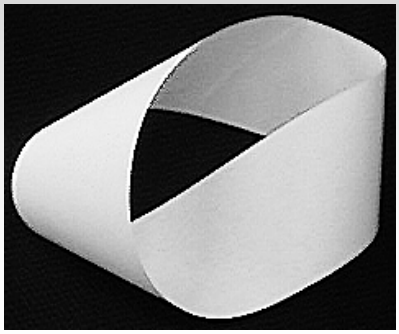
Not Backface Culled

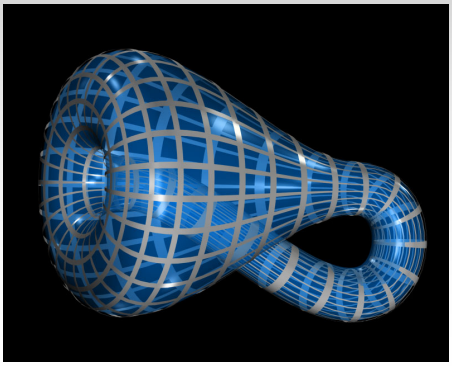


Backface Culled

Moebius & Klein

Moebius strip or
Klein bottle
- not orientable



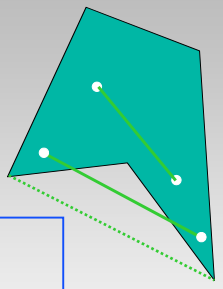


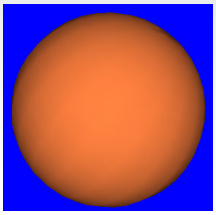
Convexity

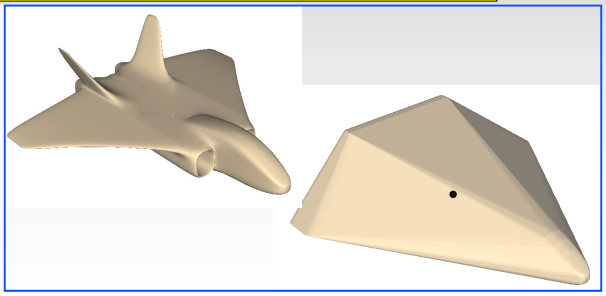
Set $C \subseteq R^d$ is ***convex*** if for any two points $p,q \in C$ and any $\alpha \in [0,1]$, $\alpha p+(1-\alpha)q \in C$

Convex hull of set $S \subseteq R^d$ is the minimal convex set C containing S

Mesh is convex if all its vertices are on its convex hull

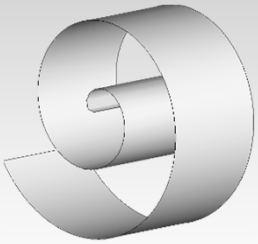






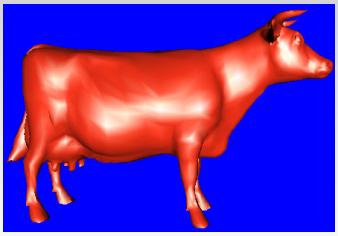
Developability

Mesh is ***developable*** if it may be embedded in R^2 without distortion



Developable

Piecewise Developable



Non-Developable

16

Mesh Data Structures

Storing mesh data

Uses of mesh data:

- Rendering
- Connectivity queries
  - *What are the vertices of face #3?*
  - *Are vertices i and j adjacent?*
  - *Which faces are adjacent to face #7?*
- Connectivity operations
  - *Remove/add a vertex/face*
- Geometry operations (change vertex positions)

Storage of generic meshes – hard to implement efficiently

Often assume: orientable , manifold & triangular

18

Storing mesh data (cont.)

How "good" is a data structure?

- Time to construct - preprocessing
- Time to answer a query
- Time to perform an operation (update the data structure)
- Space complexity
- Redundancy

19

List of faces

List of vertices (coordinates)

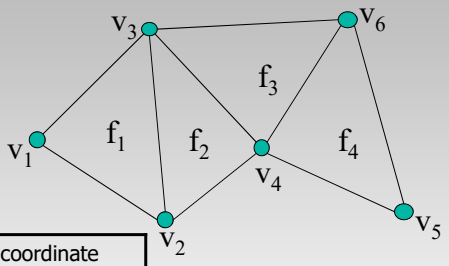
List of faces - triplets of pointers to face vertices
(c_1,c_2,c_3)

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*

20

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)

21

List of faces – pros and cons

Pros:

- Convenient and memory efficient
- Can represent non-manifold/non-orientable polygonal meshes

Cons:

- Inefficient connectivity operations - not enough information on relations between vertices & faces

22

Adjacency matrix

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

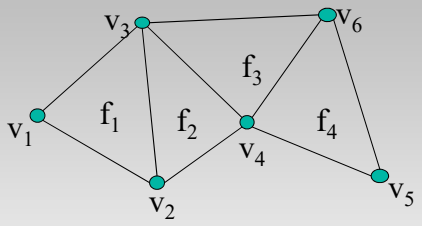
Add faces - list of triplets of vertex indices $(v1,v2,v3)$

23

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

25

Adjacency matrix – pros and cons

Pros:

- Information on vertices adjacency
- Stores non-manifold meshes

Cons:

- Connects faces to their vertices, BUT NO connection between vertex and its face
- Sufficient for some connectivity tasks but not others

26

Half-Edge Data Structure

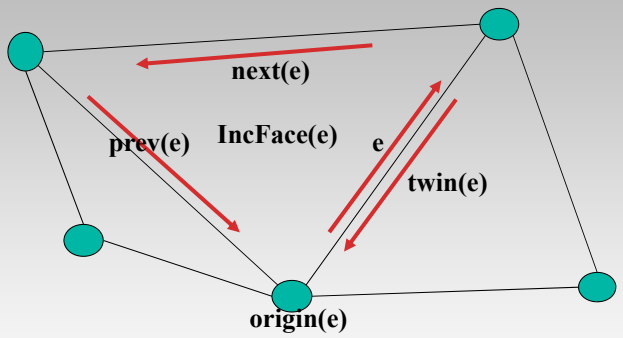
Record for each face, edge and vertex:

- Geometric information
- Topological information
- Attribute information

also called DCEL (Doubly-Connected Edge List) or Winged-Edge Structure

Half-Edge Data Structure (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)

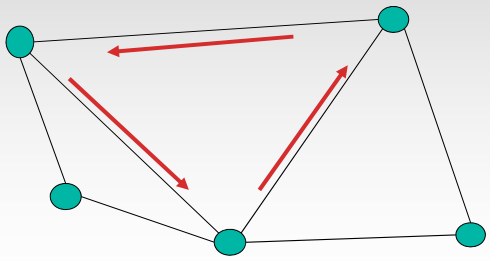
Half-Edge Data Structure (cont.)

Operations supported:

- Walk around boundary of given face
- Visit all edges incident to vertex v

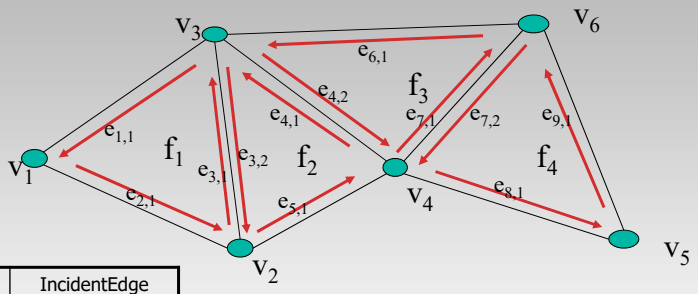
Queries:

- Most queries are O(1)



29

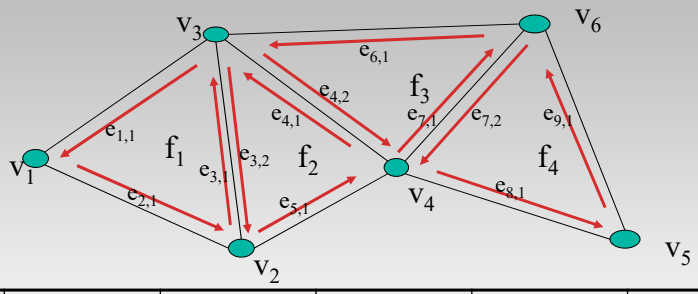
Half-Edge Data Structure - example



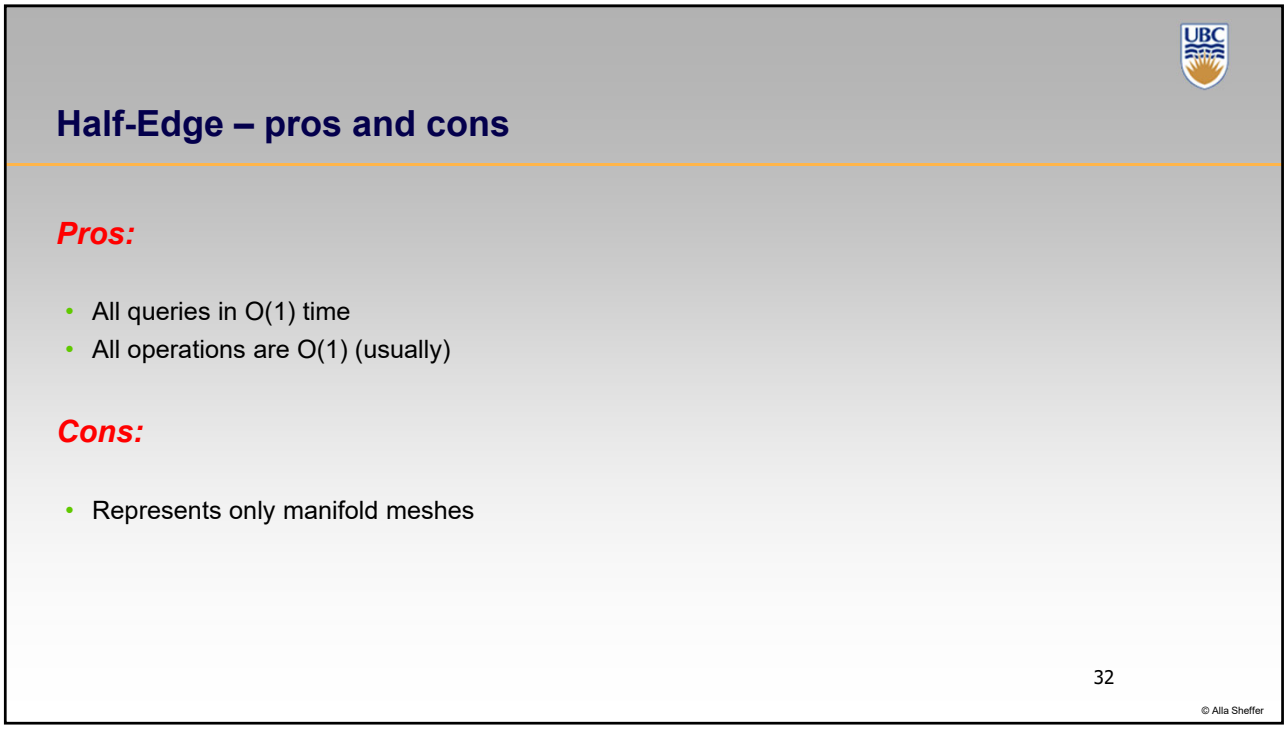
Vertex	coordinate	IncidentEdge
v_1	(x_1,y_1,z_1)	$e_{2,1}$
v_2	(x_2,y_2,z_2)	$e_{5,1}$
v_3	(x_3,y_3,z_3)	$e_{1,1}$
v_4	(x_4,y_4,z_4)	$e_{7,1}$
v_5	(x_5,y_5,z_5)	$e_{9,1}$
v_6	(x_6,y_6,z_6)	$e_{7,2}$

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

Half-Edge – 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	$e_{7,1}$	$e_{6,1}$


Half-Edge – pros and cons
Pros:
All queries in O(1) time
All operations are O(1) (usually)
Cons:
Represents only manifold meshes
32
© Alla Sheffer