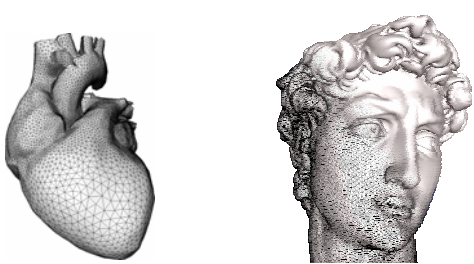


Digital Geometry Processing Introduction


Meshes: definitions & data structures



University of British Columbia

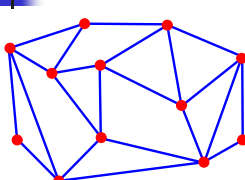
Digital Geometry=Mesh Processing (DGP)

- Processing of discrete (polygonal mesh) models
 - Typically triangular
- Most popular representation in graphics – movies, games, etc...
- Other applications
 - Engineering – CAD & simulation
 - Medical imaging, simulation, training
 - Natural sciences



University of British Columbia

Standard Graph Definitions



$G = \langle V, E \rangle$
 $V = \text{vertices} = \{A, B, C, D, E, F, G, H, I, J, K, L\}$
 $E = \text{edges} = \{(A,B), (B,C), (C,D), (D,E), (E,F), (F,G), (G,H), (H,A), (A,J), (A,G), (B,J), (K,F), (C,L), (C,I), (D,I), (D,F), (F,I), (G,K), (J,L), (J,K), (K,L), (L,I)\}$

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

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

University of British Columbia

Connectivity

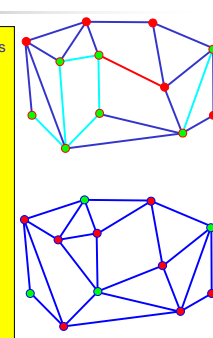
Graph is **connected** if there is a path of edges connecting every two vertices

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

Graph $G' = \langle V', E' \rangle$ is a **subgraph** of graph $G = \langle V, E \rangle$ 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

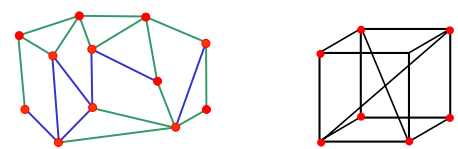
Subset V' of V is an **independent set** in G if the subgraph it induces does not contain any edges of E



University of British Columbia

Graph Embedding

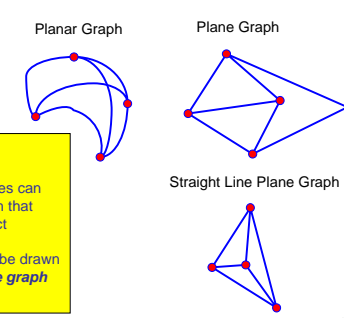
Graph is **embedded** in R^d if each vertex is assigned a position in R^d



Embedding in R^2 Embedding in R^3

University of British Columbia

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**

University of British Columbia



Digital Geometry Processing Introduction

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

University of British Columbia

Meshes

Mesh: straight-line graph embedded in \mathbb{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 $\subseteq V \times F$
Half-edges $\subseteq E \times F$

University of British Columbia

Planar Graphs and Meshes

Every manifold mesh is planar !! (almost)

University of British Columbia

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

University of British Columbia

Topology Quiz

What can you say about the genus of these meshes?

University of British Columbia

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) = $2e/v = 6(v-2+2g)/v \approx 6$ 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 triangle mesh 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 $1/4 n$?

University of British Columbia



Digital Geometry Processing Introduction

Orientability

Orientation of a face is clockwise or anticlockwise order in which its vertices and edges are listed
This defines the direction of face **normal**

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

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

University of British Columbia

Duality

face vertex
edge edge

All topological properties of a graph are preserved in its dual

University of British Columbia

Convexity

Set $C \subseteq \mathbb{R}^d$ 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 $S \subseteq \mathbb{R}^d$ is the minimal convex set C containing S

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

University of British Columbia

Developability

Mesh is **developable** if it may be embedded in \mathbb{R}^2 without distortion

University of British Columbia

Mesh Data Structures

University of British Columbia

17

Storing mesh data

Uses of mesh data:

- Rendering
- Geometry queries
 - What are the vertices of face #3?
 - Are vertices i and j adjacent?
 - Which faces are adjacent to face #7?
- Geometry operations
 - Remove/add a vertex/face
- Storage of generic meshes – hard to implement efficiently
- Assume: *orientable, manifold & triangular*


University of British Columbia



Digital Geometry Processing Introduction

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

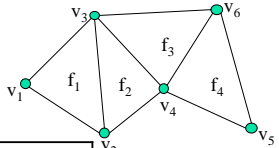
List of faces

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



22

Adjacency matrix

- View mesh as connected graph
- Given n vertices build $n \times n$ 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 (v_1, v_2, v_3)

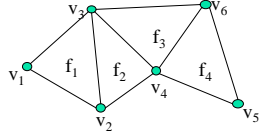


23


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	




24



Digital Geometry Processing Introduction

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



26

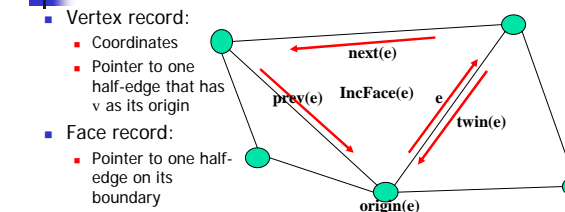
DCEL (Doubly-Connected Edge List)

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




27

DCEL (cont.)



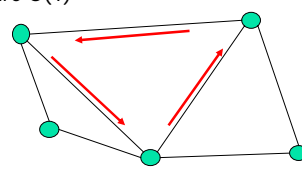

- Vertex record:
 - Coordinates
 - Pointer to one half-edge that has v as its origin
- Face record:
 - Pointer to one half-edge on its boundary
- Half-edge record:
 - Pointer to its origin, $origin(e)$
 - Pointer to its twin half-edge, $twin(e)$
 - Pointer to the face it bounds, $IncidentFace(e)$ (face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



28

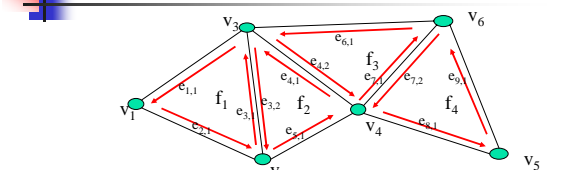
DCEL (cont.)

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


29

DCEL - example



Vertex	coordinate	IncidentEdge
v_1	(x_1, y_1, z_1)	$e_{2,1}$
v_2	(x_2, y_2, z_2)	$e_{8,1}$
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	$e_{8,1}$



30



Digital Geometry Processing Introduction

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}

University of British Columbia 31

DCEL – pros and cons

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

University of British Columbia 32

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)

University of British Columbia 33

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

University of British Columbia 34

Corner table – pros and cons

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

University of British Columbia 35

Examples of 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


University of British Columbia 36



Digital Geometry Processing Introduction

MeshMaker

- Project workspace and interface
 - rendering
 - mesh queries
 - geometry operations
- Based on
 - MFC for windows messaging
 - OpenGL for graphics drawing.
- Current version: 5.2




University of British Columbia

37

MeshMaker data structures

- Corner table
- For each vertex – list of all its corners
- Corners number $j*3, j*3+1$ and $j*3+2$ match face number j
- Corner number = EdgeID of opposite edge




University of British Columbia

38

What is it good for?

- Rendering operations invisible to user - user doesn't need to write a single rendering command
- I/O (opening/saving VRML files)
- Interacting with the model using the mouse (transformation, scaling, rotation and picking)
- Drawing lines, spheres and cylinders
- Changing colors for vertices, edges, faces, lines, spheres, cylinders and background
- Highlight vertices, edges and faces




University of British Columbia

39

What more is it good for?

- Mesh queries done usually in $O(1)$ time complexity (fast...), such as vertex neighborhood, validation, edge adjacent to both faces, etc
- Mesh geometry operations, such as vertex/face removal, vertex split, edge collapse
- Drawing text on the screen



University of British Columbia

40

MeshMaker Details

- Introduction slides MeshMaker.pdf linked from syllabus page:
 - www.cs.ubc.ca/~sheffa/dgp/syllabus.html
- Manual:
 - <http://www.cs.ubc.ca/~sheffa/dgp/software/html/index.html>
- Code:
 - <http://www.cs.ubc.ca/~sheffa/dgp/software/MeshMaker5.2.zip>




University of British Columbia

41

Graphite

- Professional mesh processing library
 - Developed by Inria, France
- Pros
 - More powerful
 - More efficient
- Cons
 - No manual & not "very" friendly
 - Linux only
- To get email Tiberiu Popa [-stpopa@cs.ubc.ca](mailto:stpopa@cs.ubc.ca)
- Will have a short demo Tuesday



University of British Columbia

42



Digital Geometry Processing Introduction

Assignment 1

- Will be published over the weekend
- Due Jan 27
- Find an independent set of vertices or faces in the mesh
 - Think of heuristics for maximal set
 - 10% bonus to best set
 - 5% bonus for second
 - Develop a user friendly interface to visualize the results and the algorithm



43

