Computação Visual e Multimédia

10504: Mestrado em Engenharia Informática

Chap. 9 — Object Data Structures

Object Data Structures

Outline

….:

- Motivation.
- Geometric structures versus topological structures.
- Topological data structures: introduction.
- Incidence and adjacency relationships.
- Spaghetti data structure.
- DCEL data structure.
- Symmetric data structure.
- Topological inference and reasoning on incidence and adjacency.
- Euler operators (still incomplete!)

Geometric object data structures

Purpose:

- data structures for representing and manipulating geometric objects in space.

Requirement:

- the stored information must allow for an *unambiguous* representation of the subdivision.

Evaluation:

- *space complexity*: amount of space (memory) needed for storing all information (entities and relations) that is explicitly represented
- *time complexity* of the algorithms for calculating relations that are not explicitly represented

Topological Data Structures

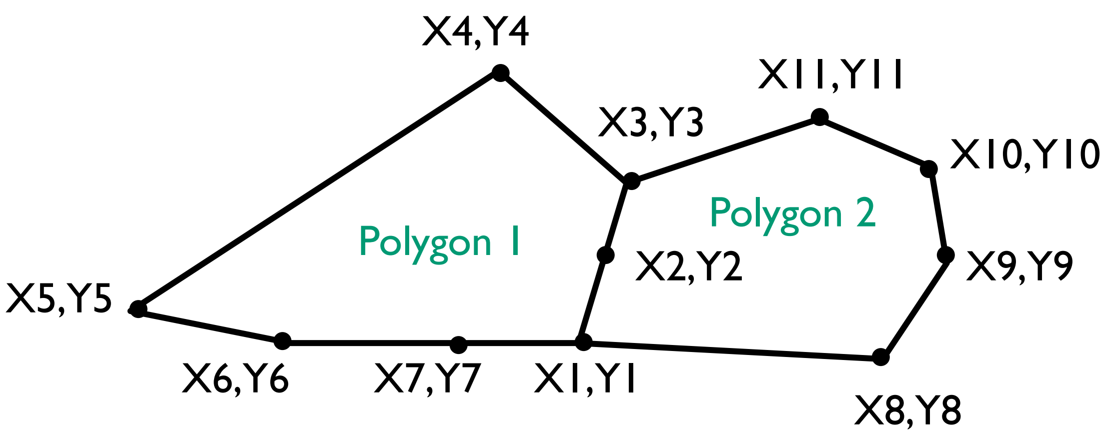
Topology / connectivity

- Generic sets of entities: *vertices*, *edges* and *faces*
- Overlayed sets of entities: only *meet* and *disjoin*
- ***Meet***: topological relation that defines connectivity between entities. Entities of different dimension are “connected” in different ways: relations (vertex-, edge-, face-based)
- ***Disjoin***: topological relation that defines the entities of lower dimension are in the booundary of of higher dimension entities.

Spaghetti data structure

- Spaghetti data structure: represents sets of points, lines and polygons
- Can be used for both generic sets of entities and overlayed sets (plane subdivisions)
- The geometry of any spatial entity is described independently of other entities
- No topology/connectivity information is recorded

- Points, lines and polygons are stored separately.
- For each polygon, we store a (ordered) list of coordinates of points on its boundary.



Polygon 1	Polygon 2
X1,Y1	X8,Y8
X2,Y2	X9,Y9
X3,Y3	X10,Y10
X4,Y4	X11,Y11
X5,Y5	X3,Y3
X6,Y6	X2,Y2
X7,Y7	X1,Y1
X1,Y1	X8,Y8

Spaghetti data structure: pros & cons

Advantages:

- simplicity
- easy insertions of new entities (all entities are independent)

Disadvantages:

- inefficient for topological queries

  No easy way of solving queries such as: "do Polygon 1 and 2 share a common bounding line?"
  Need to analyse all coordinates of points of Polygon 1 and compare with those of Polygon 2 and see if two consecutive pairs are the same: inefficient!!

- redundancies (and consequently, possible inconsistencies!)

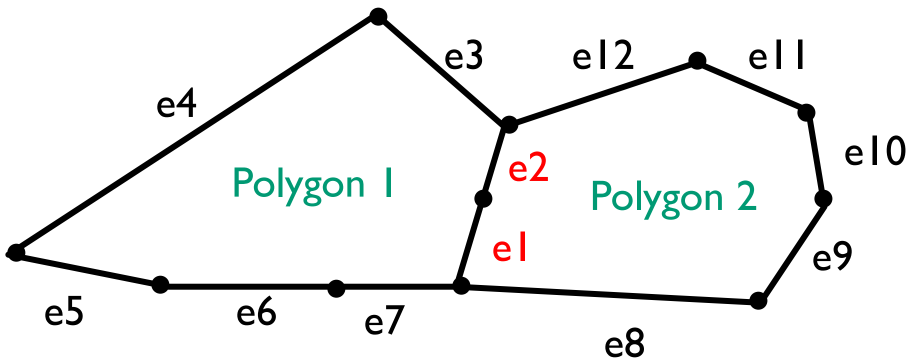
  Coordinates of points along common boundary are recorded twice!
  Redundancy: if we update coordinates of a point, we need to update them everywhere!

Topological data structures: motivation

- Storing connectivity information explicitly allows for more efficient spatial *queries*.
- Topology/connectivity: important criterion to establish the *correctness* (integrity, consistency) of geometric objects, with applications in CAD, geographical databases, etc.

Example:

If we store relation FE explicitely (i.e., for each polygon we store a list of IDs of edges bounding it), the query: *"do Polygon 1 and 2 share a common bounding line?"* only requires checking whether the two lists contain any common IDs



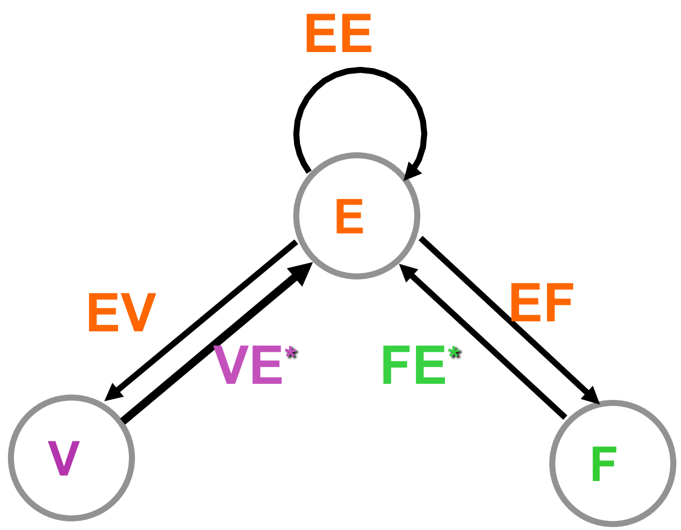
Polygon 1	Polygon 2
e1	e8
e2	e9
e3	e10
e4	e11
e5	e12
e6	e2
e7	e1

Doubly-connected edge list (DCEL)

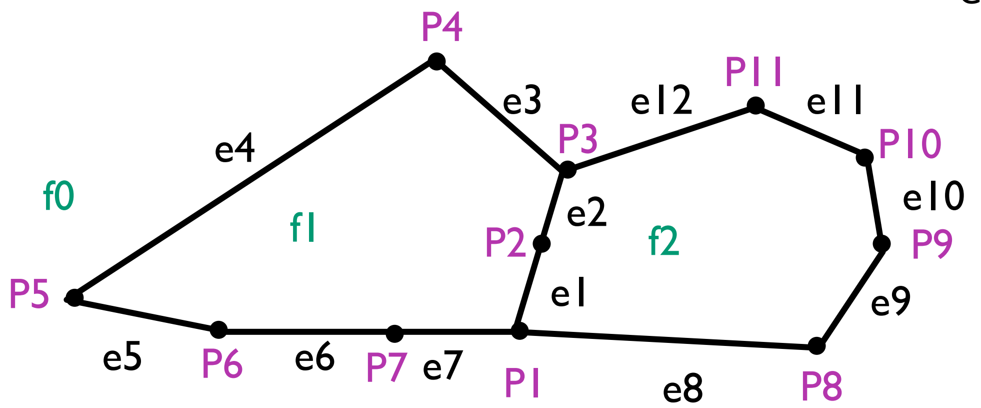
– Preparata and Shamos (1985)

DCEL structure stores:

- three sets of entities V, E, F
- three edge-based relations EV, EE, EF
- two partial relations: FE* and VE*
  - FE*: associates a face *f* with one of the edges bounding *f*
  - VE*: associates a vertex *v* with one of the edges incident in *v*



Example



Entities	
V	P1,P2,….., P11
E	e1, e2,….., e12
F	f0,f1,f2

Edge - based relations			
	EV	**EF**	**EE**
e1	P1,P2	f1,f2	e7,e2
e2	P2,P3	f1,f2	e1,e12
e3	P3,P4	f1,f0	e2,e4
e4	P4,P5	f1,f0	e3,e5
e5	P5,P6	f1,f0	e4,e6
e6	P6,P7	f1,f0	e5,e7
e7	P7,P1	f1,f0	e6,e8
e8	P1,P8	f2,f0	e1,e9
e9	P8,P9	f2,f0	e8,e10
e10	P9,P10	f2,f0	e9,e11
e11	P10,P11	f2,f0	e10,e12
e12	P11,P3	f2,f0	e11,e3

Partial relations			
	VE*		**FE***
P1	e1	f0	e3
P2	e2	f1	e3
P3	e3	f2	e1
P4	e4		
P5	e5		
P6	e6		
P7	e7		
P8	e8		
P9	e9		
P10	e10		
P11	e11		

DCEL: space complexity

For every ***edge****:*

- 3 constant relations are stored (involving 2 entities): 6e

For every ***face****:*

- 1 relation involving one entity: f

For every ***vertex****:*

- 1 relation involving one entity: n

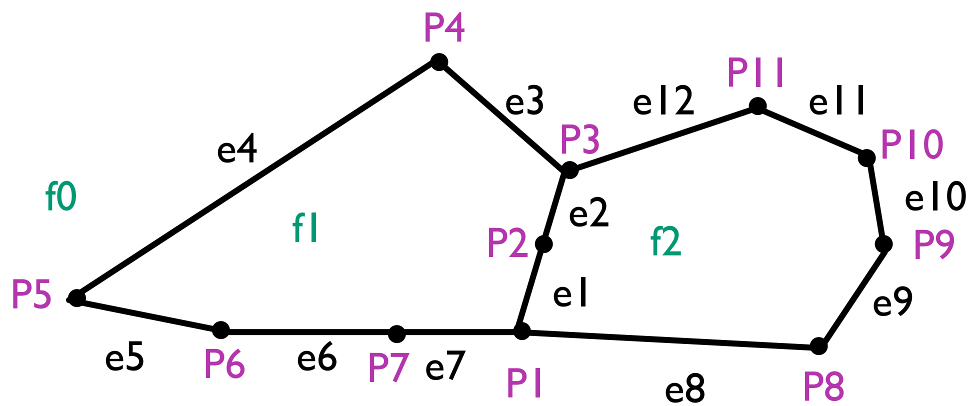
TOTAL space required to represent relations: $6e + f + n$

For each vertex we also store the two geometric coordinates: $2n$

DCEL: time complexity for FE

Calculating complete relation FE: ***Obtained by combining FE* and EE***

- For example, given a face f1, we find the first bounding edge e3 using FE*. Then using EE we find the successor e4 of e3 (in counter-closkwise order) on the boundary of f1: if e3 is oriented in such a way that f1 is on its left hand side, then e4 is the second of the two edges associated with e3 through EE
- We apply the same method (2nd element of EE) to obtain all other edges on the boundary of f1, until we reach e3 again.



DCEL: FV and FF

FV relation:

- FV can be obtained by combining FE and EV: for each bounding edge e of a given face *f* (obtained with FE), we consider its endpoints using EV

FF relation:

- FF can be obtained by combining FE and EF: for each bounding edge e of a given face *f* (obtained with FE), we consider the other face *f'* obtained by using EF

VE relation:

- Homework...

VV relation:

- Homework...

VF relation:

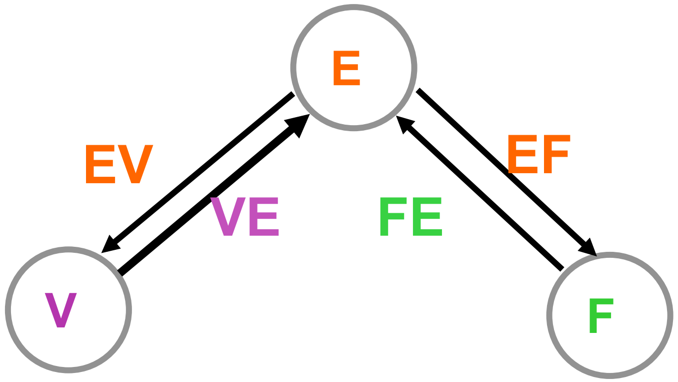
- Homework...

Symmetric data structure

– Woo (1985)

Symmetric structure stores:

- three sets of entities: V, E, F
- relation EV and its inverse VE
- relation FE and its inverse EF

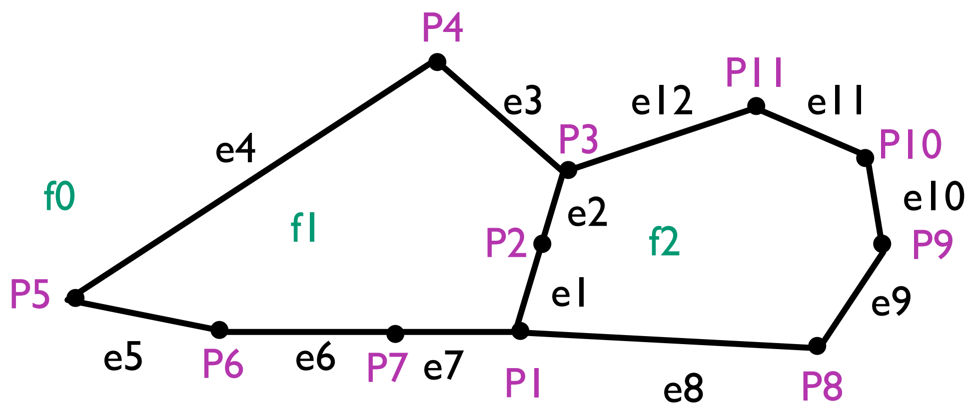


Example

	EV	EF
e1	P1,P2	f1,f2
e2	P2,P3	f1,f2
e3	P3,P4	f1,f0
e4	P4,P5	f1,f0
e5	P5,P6	f1,f0
e6	P6,P7	f1,f0
e7	P7,P1	f1,f0
e8	P1,P8	f2,f0
e9	P8,P9	f2,f0
e10	P9,P10	f2,f0
e11	P10,P11	f2,f0
e12	P11,P3	f2,f0

	VE
P1	e1,e7,e8
P2	e2,e1
P3	e3,e2,e12
P4	e4,e3
P5	e5,e4
P6	e6,e5
P7	e7,e6
P8	e8,e9
P9	e9,e10
P10	e10,e11
P11	e11,e12

	FE
f0	e3, e4,e5,e6,e7,e8,e9,e10,e11,e12
f1	e3,e4,e5,e6,e7,e1,e2
f2	e1,e8,e9,e10,e11,e12,e2



Symmetric structure: space complexity

For every ***edge****:*

- 2 constant relations are stored (involving 2 entities): 4e

For every ***face****:*

- 1 variable relation (FE). Every edge is common to two faces, so each edge is stored twice: 2e

For every ***vertex****:*

- 1 variable relation (VE). Every edge has two endpoints, so each edge is stored twice: 2e

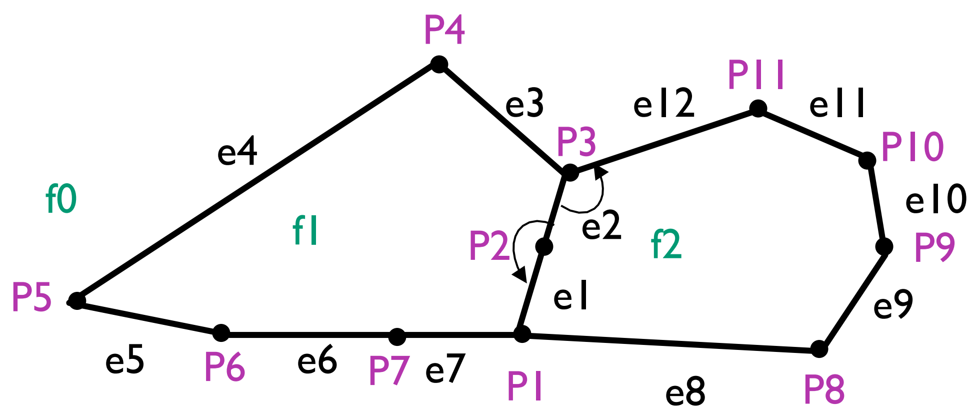
TOTAL space required to represent relations: 8e

For each vertex we also store the two geometric coordinates: 2*n*

Symmetric structure: EE

Calculating relation EE: ***Obtained by combining EV and VE (or EF and FE)***

– For example, if we want to calculate EE(e2)=(e1,e12), we retrieve the endpoints P2 and P3 of e2 using EV. To retrieve e1 we consider the successor of e2 in the list associated with P2 through VE (for e12 the successor of e2 in the list associated with P3). To do this in constant time, for each edge we need to store the position of the edge in the lists associated to its endpoints through VE.



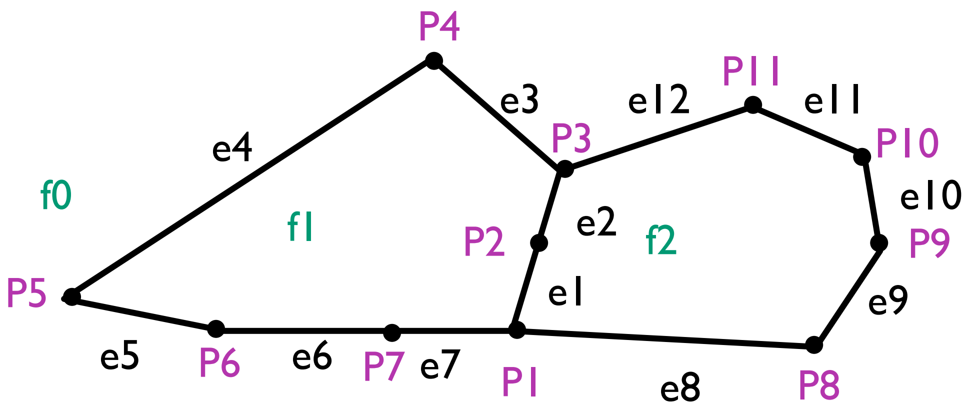
Symmetric structure: FF , FV, VV, VF

As in DCEL:
- FF: FE+EF
- FV: FE+EV
- VV:VE+EV
- VF:VE+EF

Example: FF

FF(f1)=(f0,f2) obtained combining:

- FE(f1)=(e3,e4,e5,e6,e7,e1,e2)
- EF(e3)=(f1,f0)
- EF(e4)=(f1,f0)
- EF(e5)=(f1,f0)
- EF(e6)=(f1,f0)
- EF(e7)=(f1,f0)
- EF(e1)=(f1,f2)
- EF(e2)=(f1,f2)



Euler operators

Motivation for studying Euler operators:

- Allow the incremental construction of complex objects from basic building blocks such as vertices, edges and faces.
- Applications: geometric CAD kernels, computational animation systems, etc.

To be continued....

Summary:

...:

- Motivation.
- Geometric structures versus topological structures.
- Topological data structures: introduction.
- Incidence and adjacency relationships.
- Spaghetti data structure.
- DCEL data structure.
- Symmetric data structure.
- Topological inference and reasoning on incidence and adjacency.
- Euler operators (still incomplete!)