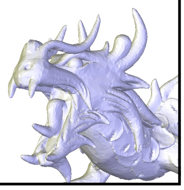
Lyon 1

Mesh and Computational Geometry

Raphaëlle Chaine
Université Claude Bernard Lyon 1

M2 ID3D
Image, Développement
et Technologie 3D
et 3A Centrale



1

Which place for Geometry in Computer Science?

- Geometric problems and applications
  - Computer Graphics
    - Collision detection
    - Illumination calculation
    - Hidden parts not taken into account
  - Robotics
    - Trajectory planning
  - Telecommunication
    - Determination of the nearest relay

2

2

Which place for Geometry in Computer Science?

- Geometric problems and applications
  - Chemistry
    - Molecular modeling
    - Pocket detection
    - Calculation of contact surfaces
  - CAD (Computer Aided Design), CAM (Computer Aided Manufacturing)
    - Integrated circuit design
    - *Reverse engineering*
  - Scientific Computation

3

3

Which place for Geometry in Computer Science?

- Geometric problems and applications
  - Museums and virtual shops
    - Digitization and availability of cultural heritage
  - Geographic Information Systems
  - Virtual creation
    - Animated films, video games
  - Digital entertainment

4

4

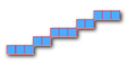
Difficulty of dealing with geometry in computer science...

5

5

Nuance between Computational Geometry and Discrete Geometry approaches

- Common goal
  - Dealing with geometric objects in a discrete world
- Discrete Geometry
  - Discretization of geometric objects on grids
    1 point = 1 pixel



6

6

Nuance between Computational Geometry and Discrete Geometry approaches

- Computational geometry
  - Handling geometric objects as abstract type instances
  - Operations consistent with the geometric properties of the objects being processed

7

7

Nuance between Computational Geometry and Discrete Geometry approaches

- Computational Geometry
  - Combinatorial objects
  - The shapes are described by assembly of geometric primitives
  - Everything can be built combinatorically on the notion of vertex

8

8

What elementary geometric objects do you think you need to model first?

9

9

What elementary geometric objects do you think you need to model first?

- Points :
  - Which points?
- Remember that we are subject to discrete arithmetic...

Floating point real representation

$(-1)^{s} * 2^{(\text{exponent}-\text{décalage})} * 1,M$ (norme IEEE)

10

10

What elementary geometric objects do you think you need to model?

- Points
- Segments

With that we can make broken lines

  - How would you model segments?
  - What is the intersection of two segments?
- Vectors, straight lines, half straight lines, ...

11

11

What elementary geometric objects do you think you need to model?

- Back to the notion of point:
  - A triplet of coordinates
  - OR A construction tree!

12

12

What elementary geometric objects do you think you need to model?

- Polygons
  - Triangle = polygon determined by a minimum number of points
- Cercles, Spheres

- Many shapes can be described combinatorically from a few points

13

13

Concept of affine combination

$$p = \sum_{i=1}^{n} \lambda_i x_i \ , \qquad \sum_{i=1}^{n} \lambda_i = 1$$

- Affine combination of n points
  - All possible combinations of these n points
  - The coefficients λ_i can be negative or positive
- Example :
  - All the points of a line AB are expressed by affine combination of A and B

14

14

Concept of convex envelope

$$p = \sum_{i=1}^{n} \lambda_i x_i \ , \qquad \sum_{i=1}^{n} \lambda_i = 1$$

- Convex envelope of n points :
  - All convex combinations of these n points: affine combinations obtained with positive coefficients λ_i
- Coefficients λ_i are denoted as barycentric coordinates
  - Unique if the number of points is lower than the dimension of the space + 1 and the points are independent

15

15

Concept of affine combination

$$p = \sum_{i=1}^{n} \lambda_i x_i \ , \qquad \sum_{i=1}^{n} \lambda_i = 1$$

- Two points :
  - Affine combination : straight line
  - Convex combinaison (positive coefficients) : segment
- Three points
  - Affine combination : plane
  - Convex combination : triangle
- Concept of barycentric coordinates

16

16

Vocabulary used in Computational Geometry

- 0-simplex : point
- 1-simplex : segment
- 2-simplex : triangle
- 3-simplex : tetrahedron
- k-simplex : Convex envelope of k+1 points being independents (affine combinations of these points = space of dimension k)

17

17

Concept of barycentric coordinates

$$p = \sum_{i=1}^{n} \lambda_i x_i \ , \qquad \sum_{i=1}^{n} \lambda_i = 1$$

- Barycentric coordinates of a point Q inside a simplex D={P_{b0}, P_{b1}, .., P_{bN}}
- Coordinate wrt P_{bj}
  - Let F_{bj} be the face in front of P_{bj} ,
  - λ_j =Volume(Q, F_{bj})/Volume(D)
  - Volume(D) = $V(D) = |\frac{1}{N!} det(P_{b1} - P_{b0}, P_{b2} - P_{b0}, .., P_{bN} - P_{b0})|$

18

18

Volume in 2D and in 3D?

- Recall
  - Cross-product of 2 vectors
    - Direction? Length?
  - Dot product of 2 vectors
- Volume in 2D
  Area(abc)=1/2 × $(b-a) \times (c-a)$
- Volume in 3D
  Volume(abcd)=1/3 × base × height

19

19

Volume in 2D and in 3D?

- Volume(abcd)=Volume(T,d)

$$\mathrm{Height}(T,d) = \langle d-a, \vec{n}_T \rangle = \left\langle d-a, \frac{(b-a)\times(c-a)}{\|(b-a)\times(c-a)\|} \right\rangle$$

$$\mathrm{Volume}(T) = \frac{1}{3} \times \mathrm{base} \times \mathrm{height} = \frac{1}{6} \times \langle d-a, (b-a)\times(c-a) \rangle$$

20

20

Vocabulary used in Algorithmic Geometry

- Face : The faces of a k-simplex defined by k+1 points are the d-simplexes that can be formed from a subset of d + 1 < k + 1 of its vertices
- Based on the previous definitions, how would you formally define what a triangulation is?

21

21

Vocabulary used in Computational Geometry

- Given a set E of points of R^k, we call triangulation of E a set of k-simplexes whose vertices are the points of E and verifying :
  - The intersection of 2 k-simplexes is either empty or a face common to the 2 k-simplexes,
  - The k-simplexes pave the convex envelope of E.

22

22

Triangulation

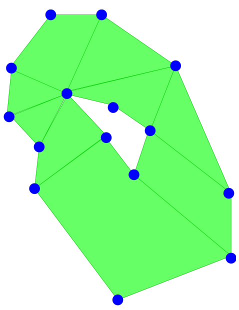
- For the sake of simplicity, we still speak of triangulation in the case where all the k-simplexes rely on vertices in a space of whose dimension is larger than k
- The coverage constraint is then released
- Example :
  Triangulation of a surface (k=2) based on 3D points

23

23

Combinatorics of connected 2D meshes

- Is there a connection between
  - the number of cells,
  - the number of edges
  - and the number of holes in this mesh ?
- To be noted: the cells are not necessarily triangles!

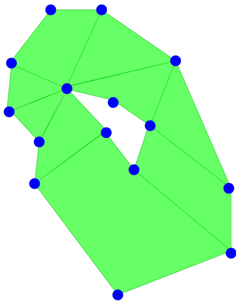


24

24

Combinatorics of connected 2D meshes

- Euler's relationship
  c-a+s=1-t
  - c : number of cells
  - a : number of edges
  - s : number of vertices (s > 1)
  - t : number of holes



25

25

Combinatorics of connected 2D meshes

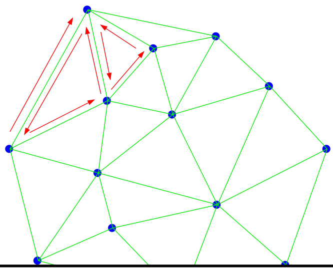
- Euler's relationship
  c-a+s=1-t
- Intuition of the evidence
  - We start from a single vertex (Euler's relationship is checked: 0-0+1 = 1-0) and we add the edges one by one by maintaining the connection.
  - Add an edge based on a new vertex: s+ = 1, a+ = 1
  - Add an edge based on existing vertices: c+ = 1 or t+ = 1
  - At each addition of an edge, (c + t) - a + s remains invariant.

26

26

Special case of triangulations

- Can we say more about the link between the number of faces, edges and vertices?

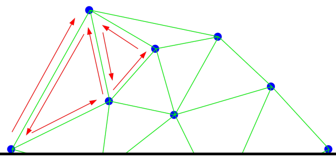


27

27

Special case of triangulations

- No hole (t = 0)
- Each cell is bordered by 3 edges
- 2a = 3c + k
  - where k is the number of edges on the convex envelope (ie. the edge)



28

28

Special case of triangulations

- Number of cells and edges of a triangulation of s points
  - Euler's relationship:
    - c-a+s=1
    - Relationship between a and c in a triangulation : 2a = 3c + k where k is the number of edges on the convex envelope
    - c = 2s - 2 - k
    - a = 3s - 3 - k

29

29

Data structures

- How to store a triangular mesh so that one can easily navigate through it?

30

30

Data structures

- Geometric information :
  - coordinates of vertex positions
- Topological information :
  - incidence and adjacency relationships between vertices, edges and faces

31

31

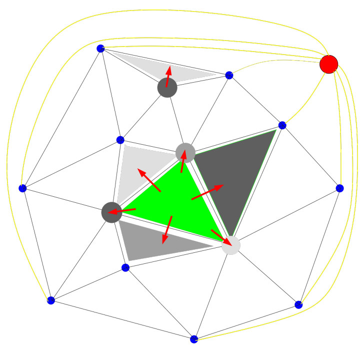
Data structures

- Triangle and vertex based representation (only for triangulated mesh)
  - <u>Triangle</u> :
    - Access to the 3 incident vertices (trigonometric order)
    - Access to the 3 adjacent triangles

    Constraint : vertex i facing adjacent triangle i
  - <u>Vertex</u> :
    - Access to 1 incident triangle
    - Access to the underlying point (geometry)

32

32



33

33

Data structures

- Case of a 2D triangulation
  - Dangling pointers / indexes on the boundary of the convex hull

  OR
  - Addition of a fictitious vertex (called infinite vertex) whose incident triangles are attached to the edges of the convex envelope
- The data structure can also be used for surface triangulations
  - Use dangling pointers/indexes for each hole boundary

    OR
  - Add fictitious vertices for each hole

34

34

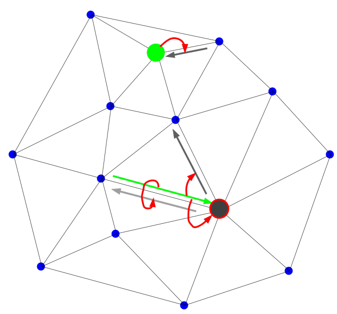
Data structures with edges

- Representation based on ½ edges and vertices
  - <u>½ edge</u> :
    - Access to the coupled ½ edge
    - Access to the next ½ edge
    - Access to the target vertex
  - <u>Vertex</u> :
    - Access to an ½ edge oriented towards the vertex
    - Access to the underlying point

35

35

Data structures



36

36