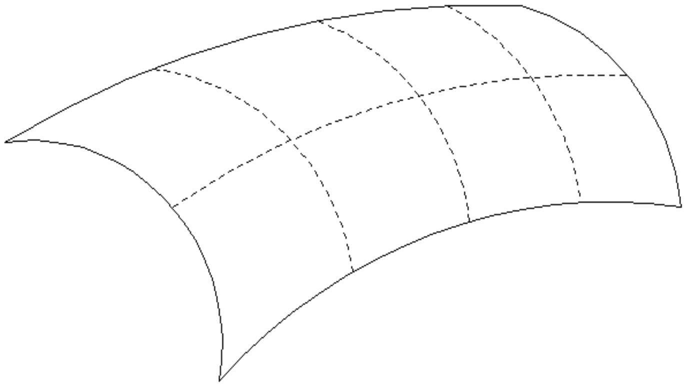
Surface Approximation with Triangle Meshes

Outline

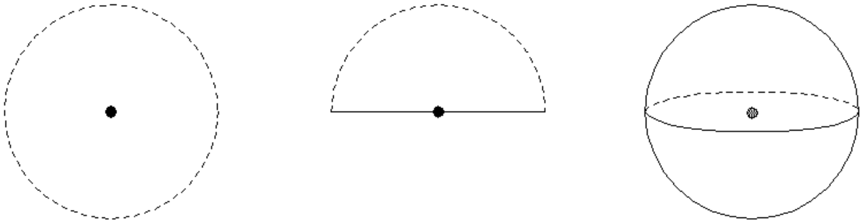
- Classification of surfaces
- Approximating surfaces with triangle meshes
- Encoding triangle meshes
- Compressed mesh representations

What is a surface?

- A surface S embedded in space is a subset of R^3 that is intrinsically two-dimensional



- For any neighborhood u_P of a point P on S
  - u_P contains at least half of an open disk (i.e., no part of S is less than two-dimensional)
  - u_P does not contain any open ball (i.e., no part of S is solid)



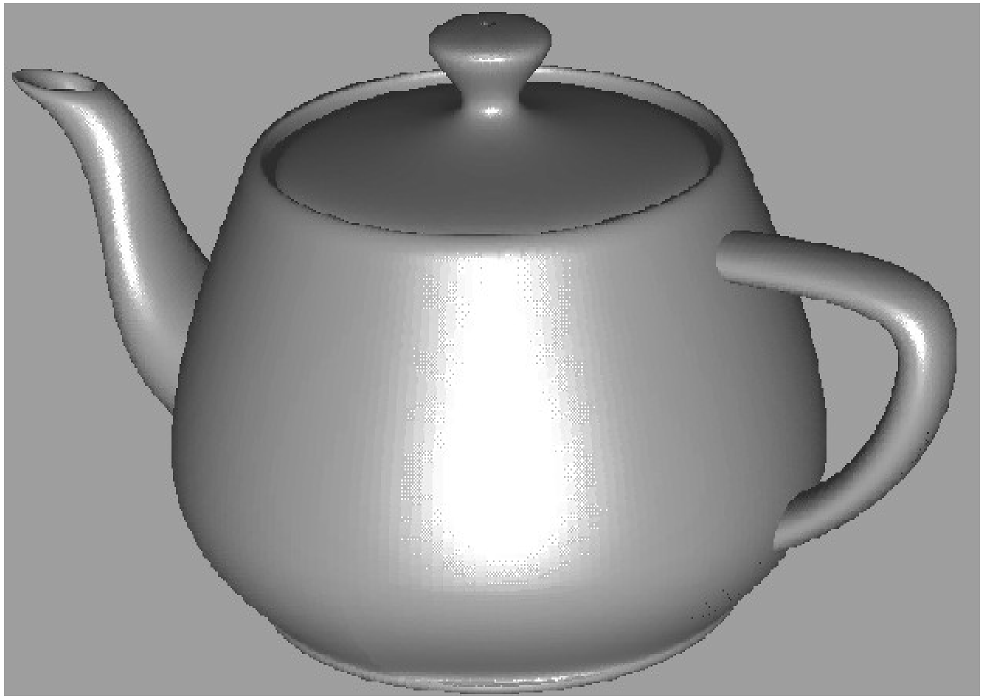
Surface Modeling

- Surfaces defined in the continuum:
  - Sources: mathematics, CAD
  - Problem: a finite (digital) representation is necessary for surface analysis and rendering
- Surfaces known at a finite set of points:
  - Source: sampling (range scanners, photogrammetry, medical data), simulation (finite element methods)
  - Problem: a surface in the continuum must be defined through a reconstruction process

Topological Characterization of Surfaces

- Manifold without boundary

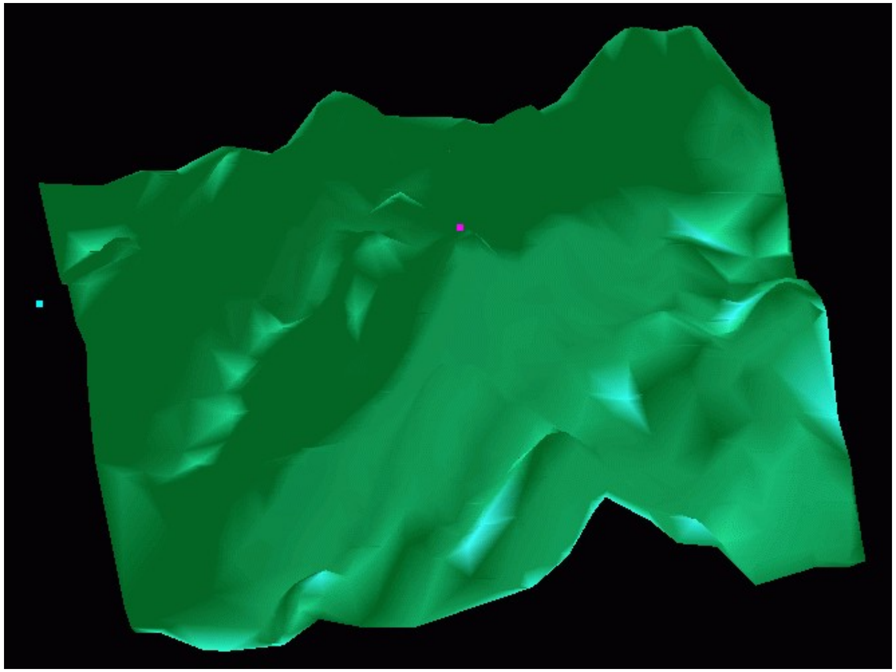
  a surface ***S*** in $\boldsymbol{R}^3$ such that any point on ***S*** has an open neighborhood homeomorphic to an open disk in $\boldsymbol{R}^2$



...Topological Characterization of Surfaces...

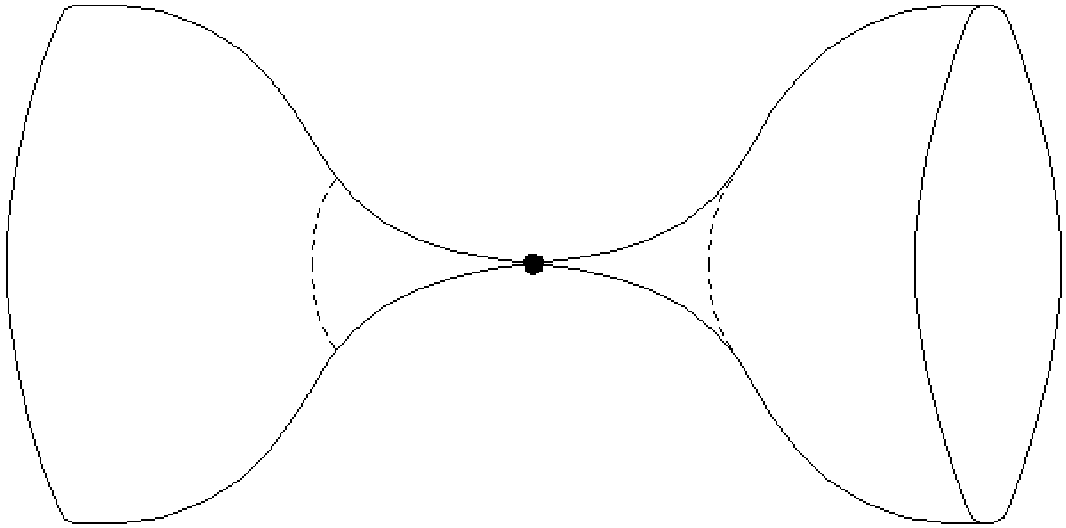
- Manifold with boundary

  a surface ***S*** in $\boldsymbol{R^3}$ such that any point on ***S*** has an open neighborhood homeomorphic to an open disk or to half an open disk in $\boldsymbol{R^2}$



...Topological Characterization of Surfaces...

- An example of a non-manifold situation



Geometric Representation of Surfaces

- Implicit form:

an *implicit surface* is the locus of solutions of an equation

$$F(x,y,z) = 0$$

where F is a mathematical expression of three variables

Problems:

- definition is too general: some expressions give objects that are not intrinsically two-dimensional
- we might not know expression $F(x,y,z)$
- even if we know F, we might not be able to solve the equation

Remark: surfaces which cannot be described in an analytic form are called *free-form* surfaces

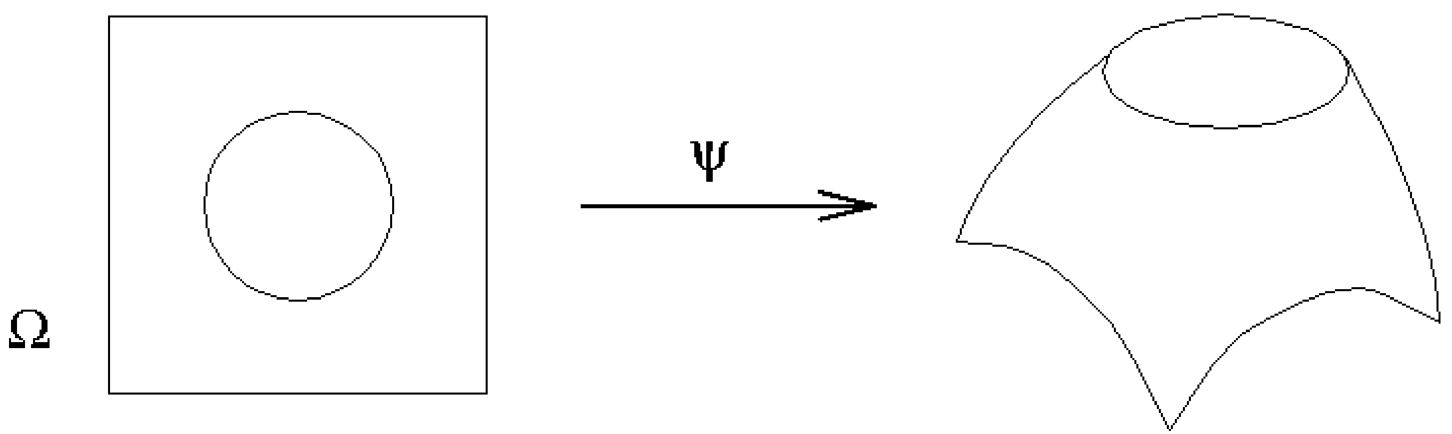
...Geometric Representation of Surfaces...

- Parametric form:

  a *parametric patch* is the image of a continuous function

  $$\psi: \Omega \subset R^2 \longrightarrow R^3$$

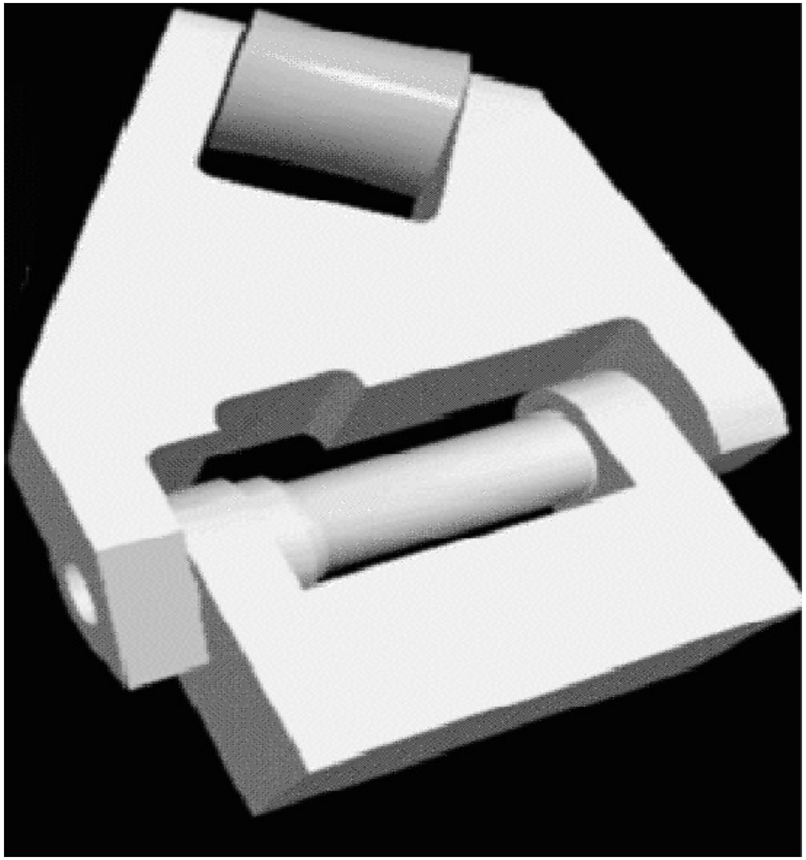
  - Ω is called *parametric space*
  - R^3 is called *physical space*
  - boundaries of Ω and of $\psi(\Omega)$ are formed by *trimming curves*



...Geometric Representation of Surfaces...

- Parametric surface:

  a collection of parametric patches properly abutting

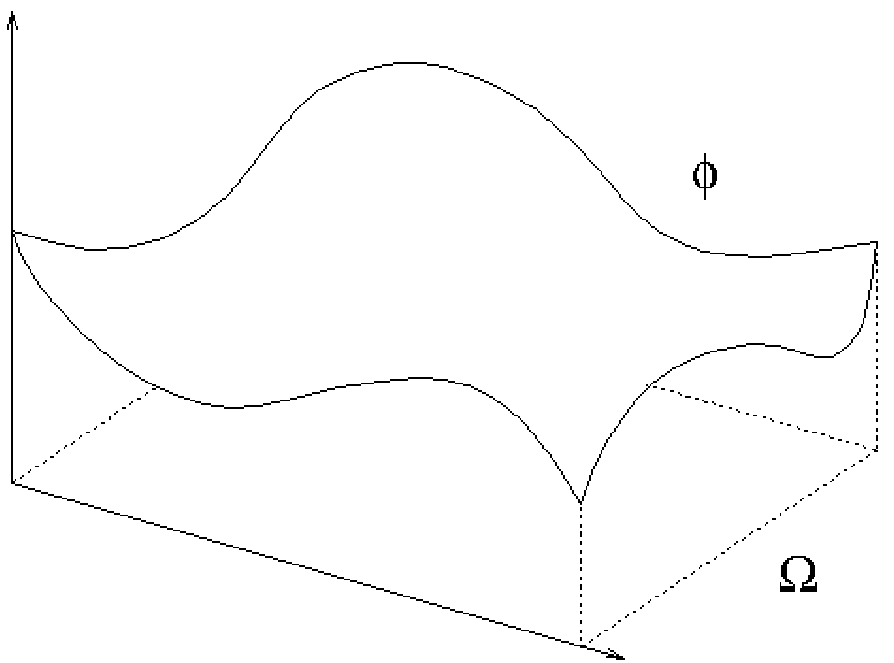


...Geometric Representation of Surfaces...

- Explicit surfaces:
  - Special case of a parametric surface
  - A surface can be represented as a bivariate function when it is the image of a scalar field

$$\phi: \Omega \subset R^2 \longrightarrow R$$

- Example: topographic surfaces



Hypersurfaces

- Generalization of explicit surfaces to higher dimensions
- Image of a scalar field

$$\phi: \Omega \rightarrow R$$

where Ω is a compact domain in R^k

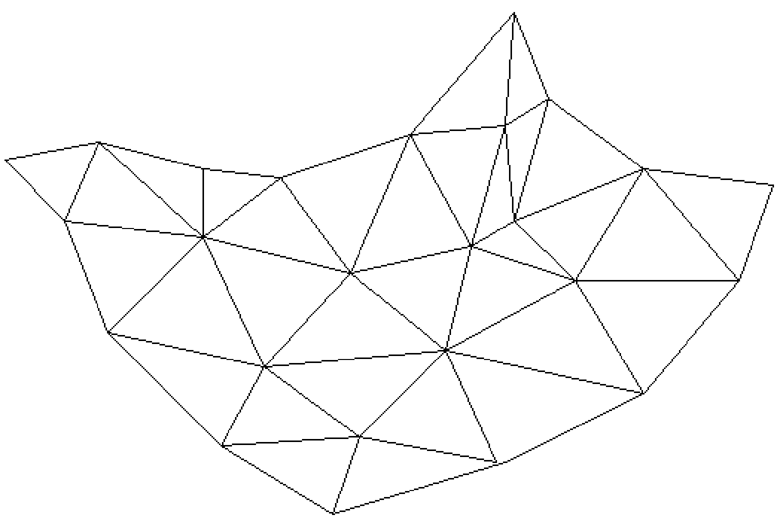
- Example: volume data (for $k=3$)

Approximating surfaces with triangle meshes

- Surface representation:

  mesh of triangles (i.e., a set of triangles such that any two of them either do not intersect or share a common edge or vertex)

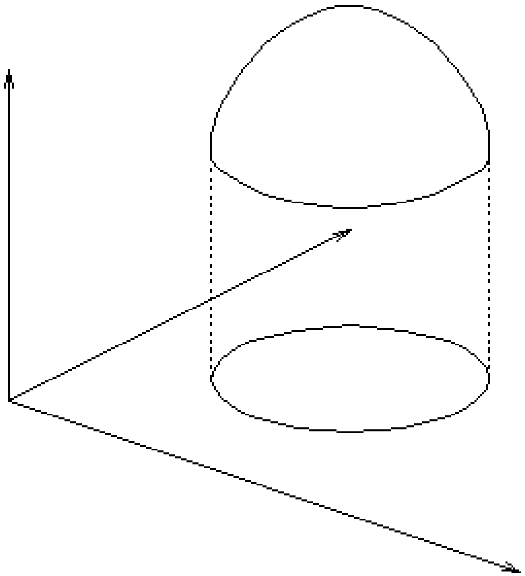
- Each triangle approximates a surface patch within a given accuracy
- Triangle meshes are easy to represent, manipulate, visualize
- Triangle meshes can be constructed from irregularly sampled data

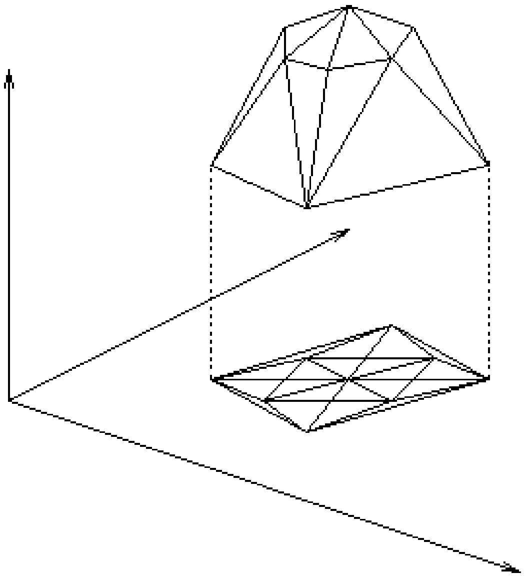


Approximating a 2-dimensional scalar field with a triangle mesh

K=2:

- A 2-dimensional scalar field is described as a function $z = \phi(x,y)$
- A triangle meshes in 3D is obtained by triangulating the domain of ϕ and lifting it to three-dimensional space





Approximating a 3-dimensional scalar field with a tetrahedral mesh

k=3:

- A 3-dimensional scalar field is defined by a function $\boldsymbol{z = \phi(x_1,x_2,x_3)}$
- An approximation is obtained by discretizing the domain of ϕ with a tetrahedral mesh
- A *tetrahedral mesh* is a set of tetrahedra such that any two of them either do not intersect or share a common face, edge or vertex

Approximating a k-dimensional scalar field with a simplicial mesh

- Discretization of the domain of a k-dimensional field $z = \phi(x_1, x_2, \ldots, x_k)$ with a simplicial mesh

$$\Downarrow$$

- Defines a linear approximation of ϕ in (k+1)-dimensional Euclidean space

How to compute the approximation?

- How well does a mesh approximate a given surface?
- We are not given surfaces but
  - *mesh of triangles* for free-form surfaces
  - *set of points* at which the field is known for scalar fields (hypersurfaces)

The Error Metric

- *Euclidean distance* between a point ***p*** and a set $Q \subseteq R^k$:

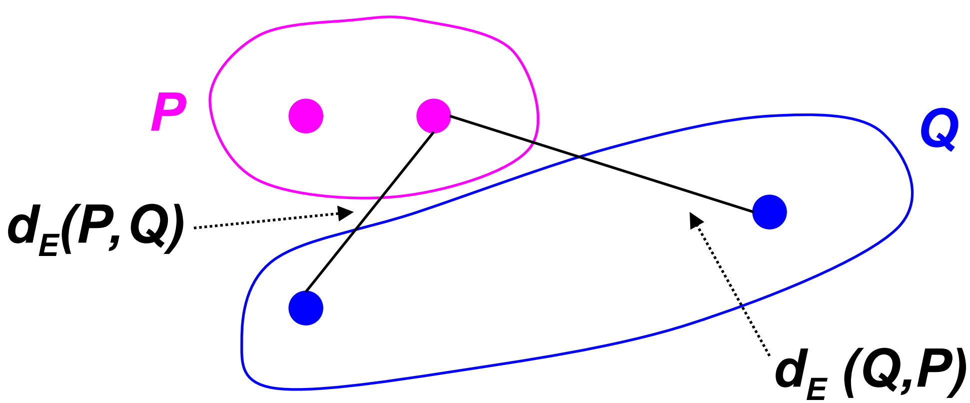
$$d(p,Q) = \inf \{ d(p,q) \mid q \text{ in } Q \}$$

where ***d(p,q)*** is the Euclidean distance between point ***p*** and ***q***

- *"Distance"* from a set ***P*** to a set ***Q***

$$d_E(P,Q) = \sup \{ d(p,Q) \mid p \text{ in } P \}$$

However $d_E(P,Q) <> d_E(Q,P)$



...The Error Metric...

- *Hausdorff distance* defined as:

$$d_H\,(P,Q) = max\ \{\ d_E\,(P,Q),\ d_E\,(Q,P)\ \}$$

It follows that $d_H\,(P,Q) = 0$ iff $P = Q$

Thus, we can express the distance between a surface ***S*** and its approximating triangle mesh ***T*** as $d_H\,(S,T)$

...The Error Metric...

Discrete case

- Free-form surfaces:
  - Surface ***S*** given as a fine mesh of triangles T_S
  - ==> we measure distance between two triangle meshes
- Hypersurfaces:
  - Scalar field ϕ is known at a finite set of points ***Q***
  - ==> we measure the distance of the points of ***Q*** from the triangle mesh ***T*** approximating the hypersurface

...The Error Metric...

The *Metro* tool [Cignoni et al. 98]

Given two triangular meshes T_1 and T_2, ***Metro*** :

- **scan converts** each triangle ***t*** of T_1 with a user-selected scan step, *or*, alternatively, chooses a set ***P*** of **points** distributed **randomly** on ***t***
- for each point ***p*** in ***P***, computes $d(p,T_2)$ (distances are computed efficiently using a bucketing data structure)

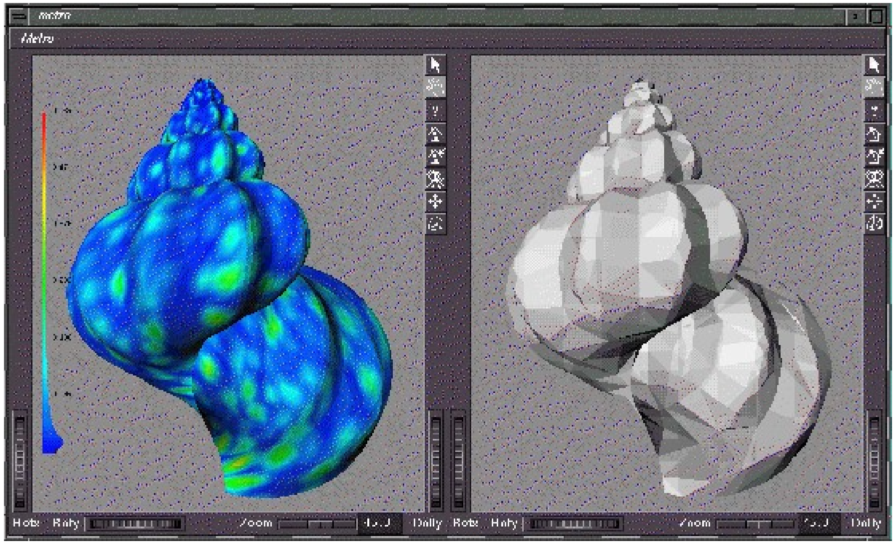
and switches meshes to be symmetric.

- precision of the evaluation depends on *sampling resolution !*
- with a sufficiently fine sampling step, almost equal results in both directions (e.g., 0.01% of mesh bounding box diagonal)

...The Error Metric...

Metro returns

- accurate **numerical** distance estimation
- a **visual** representation of the approximation error



- tool runs on SGI ws (OpenInventor)
- available in public domain

...The Error Metric...

Approximating the error on scalar fields

- Error of a point ***p*** of set ***Q*** defined as

$$e(p) = | \phi(p) - \phi_T(p) |$$

where

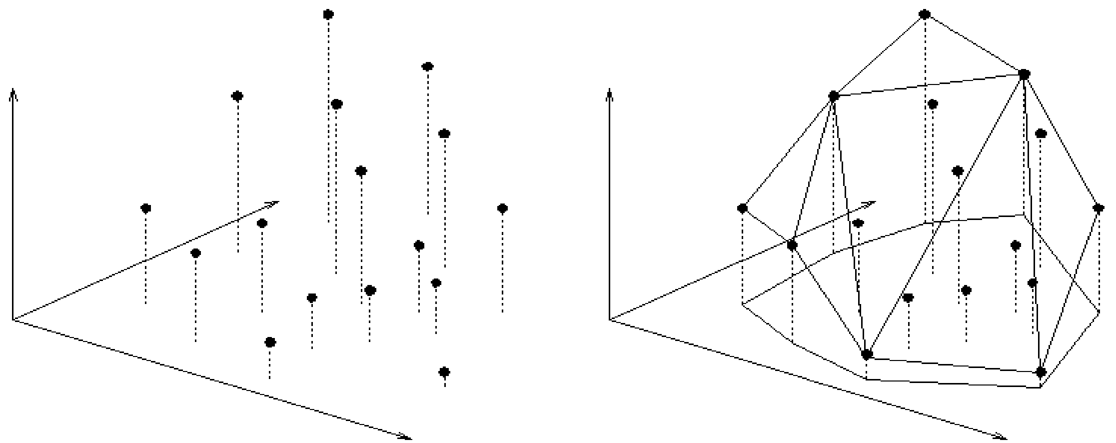
  - $\phi(p)$ is the known value of the field at ***p***
  - $\phi_T(p)$ is the approximated value of the field at ***p*** computed on the basis of simplicial mesh ***T***

...The Error Metric...

S U R F A C E S A N D M E S H E S

- Error function defined by a discrete norm:
  - *E(T,Q) = || e(p) ||*
  - *E(T,Q) = max {e(p), p $\in$ Q }*

- Example: two-dimensional scalar field (terrain)



Remarks

- More accurate representation ⇒ more triangles
- More triangles ⇒ higher storage and processing time

Tradeoff between accuracy and space / time:

- adapting the accuracy to the needs of an application can improve efficiency
- accuracy might be variable over different portions of the object

Encoding Triangle Meshes

- Two types of information encoded:
- *Geometrical*
  - position in space of the vertices
  - surface normals at the vertices
- *Topological*
  - mesh connectivity information
  - relations among triangles of the mesh

Data Structures for Triangle Meshes

List of triangles:

  - it encodes the list of triangles of the mesh
  - for each triangle, it maintains its three vertices by explicitly encoding the geometrical information associated with the vertices (position in space of the vertices, and, usually, surface normals at the vertices)
  - connectivity described through a relation between a triangle and all its vertices.

- Drawback:
    - each vertex is repeated for all triangles incident in it
- Storage cost:
    - in a triangle mesh with ***n*** vertices, there are ***~2n*** triangles
    - cost: ***18n*** floats, if geometric information associated with a vertex is just its position in space

...Data Structures for Triangle Meshes...

S U R F A C E S A N D M E S H E S

Indexed data structure:

- list of vertices + list of triangles + relation between triangles and vertices
- for each vertex: its geometrical information
- for each triangle: three references to the list of vertices

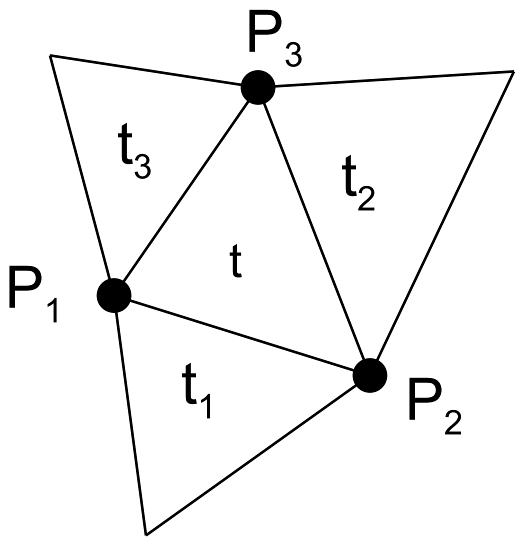
- Storage cost:

  6n log n bits
  \+
  3n floats (cost of storing geometrical information)

  since a vertex reference for a triangle requires ***log n*** bits

...Data Structures for Triangle Meshes...

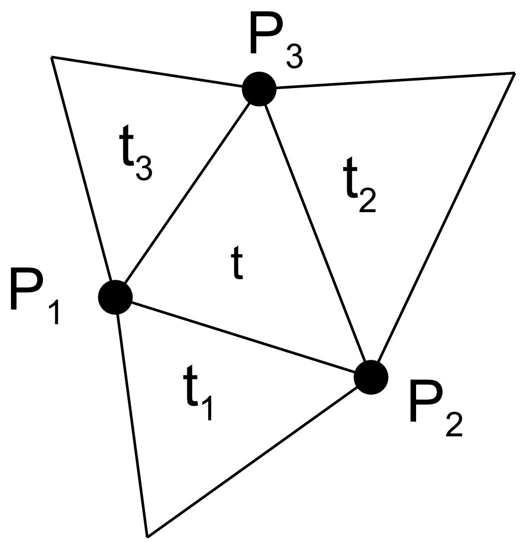
- Indexed data structure: difficult to obtain triangle adjacency information
- Triangle adjacencies are useful for algorithms which “navigate” a triangle mesh
- Idea: store, for each triangle, the indexes to its three edge-adjacent triangles as well



...Data Structures for Triangle Meshes...

Indexed data structure with adjacencies:

- list of vertices (with their geometrical information) + list of triangles
- for each triangle: references to its three vertices + references to its three adjacent triangles



$t: (P_1, P_2, P_3)\ (t_1, t_2, t_3)$

- Storage cost:
  - ***(12n log n + 6n)*** bits + ***3n*** floats (cost of storing geometrical information), since each triangle reference requires ***(log n + 1)*** bits

...Data Structures for Triangle Meshes...

Indexed data structure with adjacencies: storage costs

- ***6 n log n*** bits triangle-vertex relation
- ***6 n (log n+1)*** bits triangle-triangle relation

- Total
  - ***(12n log n + 6n)*** bits of connectivity

...Data Structures for Triangle Meshes...

S U R F A C E S   A N D   M E S H E S

Comparison of the three data structures

n = number of vertices, let one float = **32** bits

- list of triangles: $18n$ floats
  - if $n = 2^{16}$ ==> $18*2^{16}$ floats = $578*2^{16}$ bits
- indexed data structure: $6n \log n$ bits + $3n$ floats
  - if $n = 2^{16}$ ==> $96*2^{16}$ bits+ $3*2^{16}$ floats = $150*2^{16}$ bits
- indexed data structure with adjacencies: ($12n \log n + 6n$) bits+ $3n$ floats
  - if $n = 2^{16}$ ==> $198*2^{16}$ bits + $3*2^{16}$ floats = $252*2^{16}$ bits

Data Structures for Tetrahedral Meshes

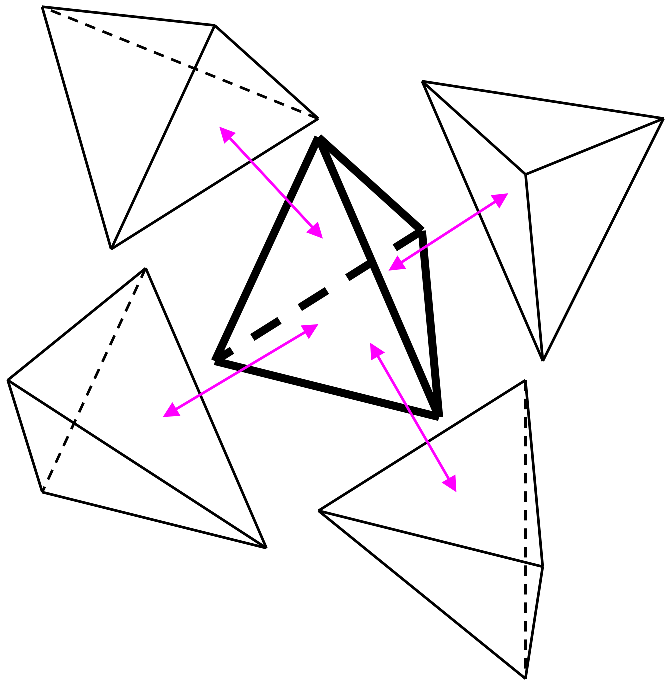
- Direct extension to three dimensions of the data structures described for triangle meshes
- Basic elements: vertices + tetrahedra
  - *list of tetrahedra*
  - *indexed structure*
  - *indexed structure with adjacencies*

...Data Structures for Tetrahedral Meshes...

- *List of tetrahedra* with their geometry inside the list

  Storage cost:

  12t, where ***t*** is the number of tetrahedra



- *Indexed structure* (each tetrahedron has references to its four vertices)

  Storage cost:

  4t log n bits **+** cost for storing geometric information

  where ***n*** is the number of vertices

...Data Structures for Tetrahedral Meshes...

- *Indexed structure with adjacencies*:

  for each tetrahedron, references to its four vertices + references to its four face-adjacent tetrahedra

Storage cost:

4t(log n + log t) bits **+** cost of storing geometric information