2. Basics

- Data sources
- Visualization pipeline
- Data representation
  - Domain
  - Data structures
  - Data values
  - Data classification

2.1. Data Sources

- The capability of traditional presentation techniques is not sufficient for the increasing amount of data to be interpreted
  - Data might come from any source with almost arbitrary size
  - Techniques to efficiently visualize large-scale data sets and new data types need to be developed

- Real world
  - Measurements and observation
- Theoretical world
  - Mathematical and technical models
- Artificial world
  - Data that is designed

2.1. Data Sources

- Real-world measurements
  - Medical Imaging (MRI, CT, PET)
  - Geographical information systems (GIS)
  - Electron microscopy

MB

  - Meteorology and environmental sciences (satellites)
  - Seismic data
  - Crystallography

GB

  - High energy physics
  - Astronomy (e.g. Hubble Space Telescope 100MB/day)
  - Defense

TB

2.1. Data Sources

- Theoretical world
- Computer simulations
  - Sciences
    - Molecular dynamics
    - Quantum chemistry
    - Mathematics

MB

    - Molecular modeling
    - Computational physics
    - Meteorology
    - Computational fluid mechanics (CFD)

GB

  - Engineering
    - Architectural walk-throughs

MB

    - Structural mechanics
    - Car body design

GB

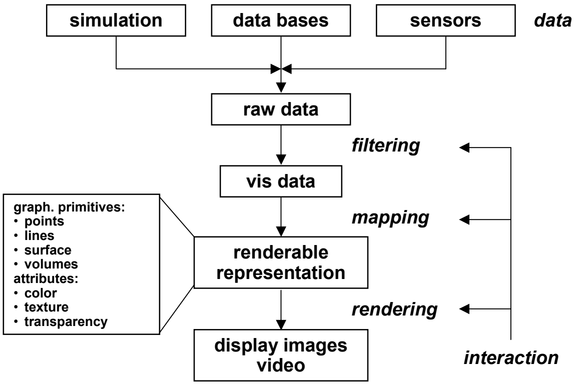
2.1. Data Sources

- Theoretical world
- Computer simulations
  - Commercial
    - Business graphics **MB**
    - Economic models
    - Financial modeling **GB**
- Information systems
  - Stock market (300 Mio. transactions per day in NY) **TB**
  - Market and sales analysis
  - World Wide Web !!!

2.1. Data Sources

- Artificial world
  - Drawings **MB**
  - Painting
  - Publishing
  - TV (teasers, commercials) **GB**
  - Movies (animations, special effects) **TB**

2.2. Visualization Pipeline

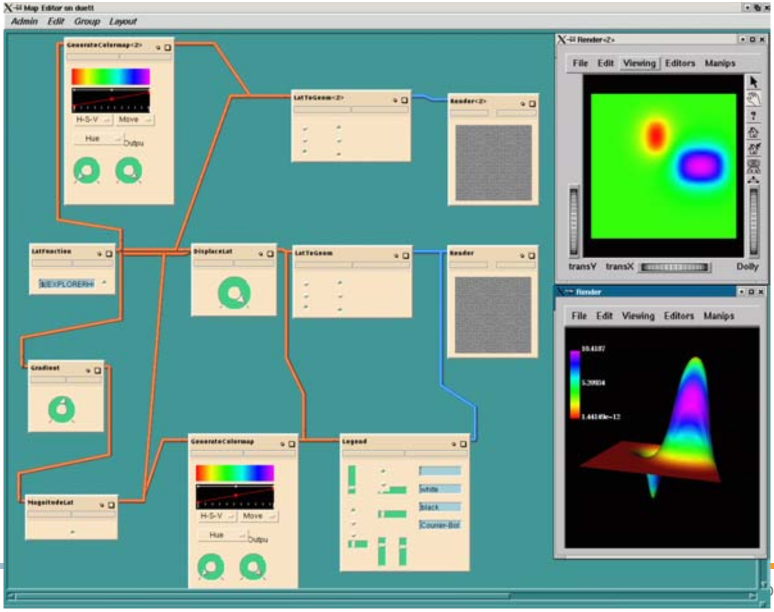


filtering
data -> data
- data format conversion
- clipping/cropping/denoising
- slicing
- resampling
- interpolation/approximation
- classification/segmentation

mapping
data ->graphical primitives
- scalar field ->isosurface
- 2D field ->height field
- vector field ->vectors
- tensor field ->glyphs
- 3D field -> volume

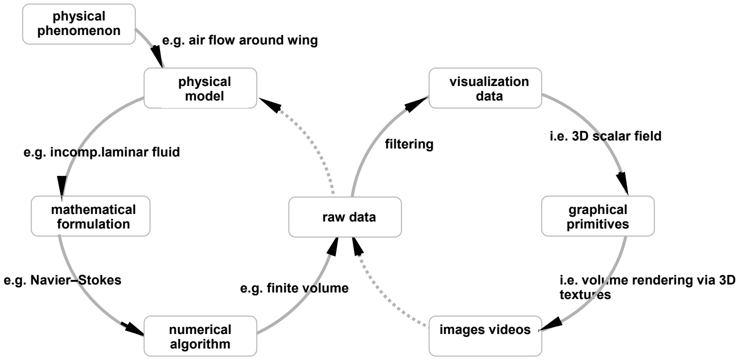
rendering:
- geometry/images/volumes
- “realism“ – e.g. : shadows, lighting, shading

2.2. Visualization Pipeline



2.2. Visualization Pipeline

- Example: simulation of the flow within a fluid around a wing



physical phenomenon

e.g. air flow around wing

physical model

e.g. incomp.laminar fluid

mathematical formulation

e.g. Navier–Stokes

numerical algorithm

e.g. finite volume

raw data

filtering

visualization data

i.e. 3D scalar field

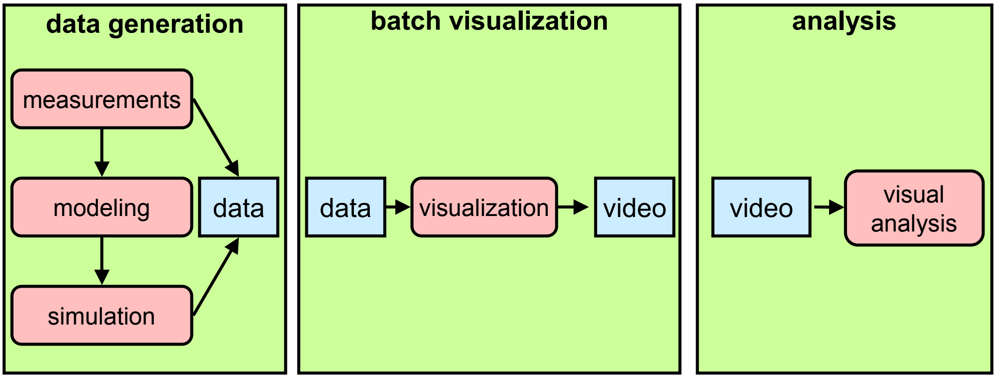
graphical primitives

i.e. volume rendering via 3D textures

images videos

2.2. Visualization Pipeline

- Scenario: video/movie mode – offline, no interaction



data generation

measurements

modeling

simulation

data

batch visualization

data

visualization

video

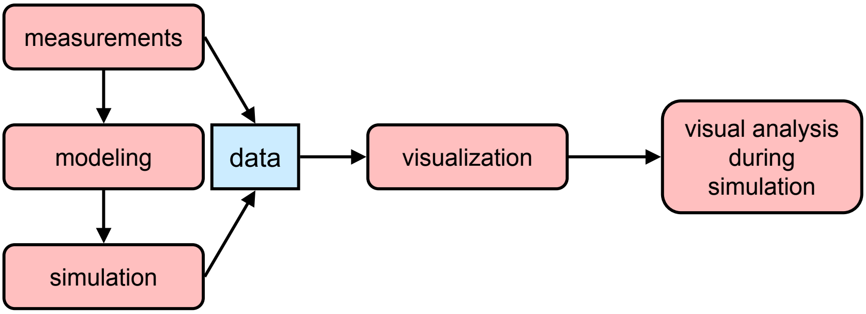
analysis

video

visual analysis

2.2. Visualization Pipeline

- Scenario: tracking – online, no interaction



measurements

modeling

data

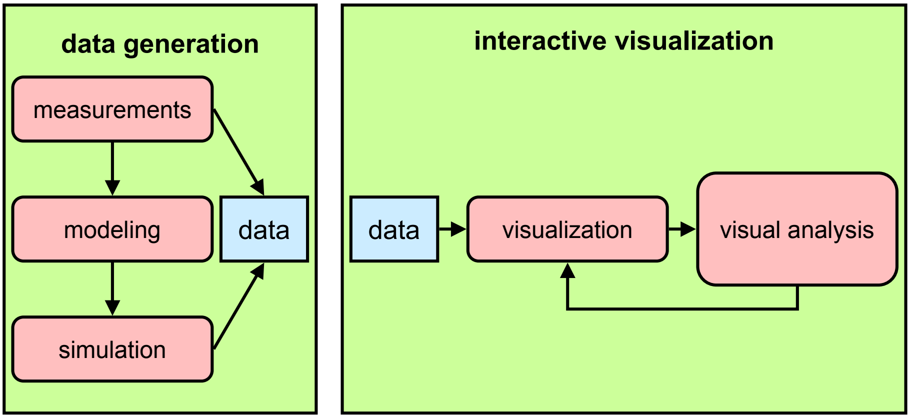
visualization

visual analysis during simulation

simulation

2.2. Visualization Pipeline

- Scenario: interactive post processing / visualization - offline



data generation

measurements

modeling

data

simulation

interactive visualization

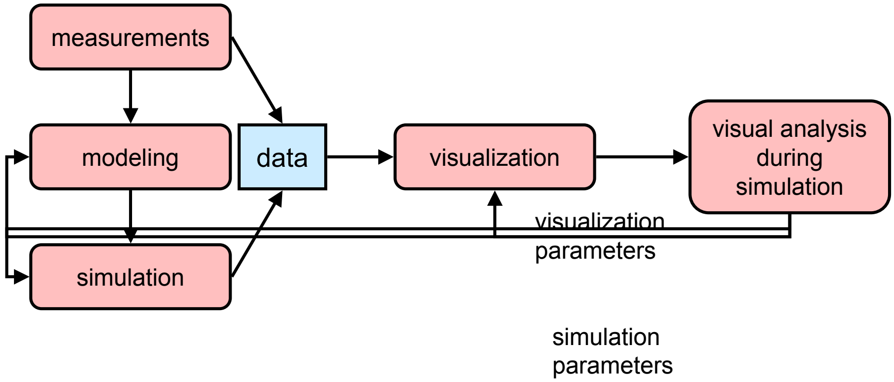
data

visualization

visual analysis

2.2. Visualization Pipeline

- Scenario: interactive steering / computational steering



2.3. Sources of Error

- Data acquisition
  - Sampling: are we (spatially) sampling data with enough precision to get what we need out of it?
  - Quantization: are we converting “real” data to a representation with enough precision to discriminate the relevant features?
- Filtering
  - Are we retaining/removing the “important/non-relevant” structures of the data ?
  - Frequency/spatial domain filtering
    - Noise, clipping and cropping
- Selecting the “right” variable
  - Does this variable reflect the interesting features?
  - Does this variable allow for a “critical point” analysis ?

2.3. Sources of Error

- Functional model for resampling
  - What kind of information do we introduce by interpolation and approximation?
- Mapping
  - Are we choosing the graphical primitives appropriately in order to depict the kind of information we want to get out of the data?
  - Think of some real world analogue (metapher)
- Rendering
  - Need for interactive rendering often determines the chosen abstraction level
  - Consider limitations of the underlying display technology
    - Data color quantization
  - Carefully add “realism”
    - The most realistic image is not necessarily the most informative one

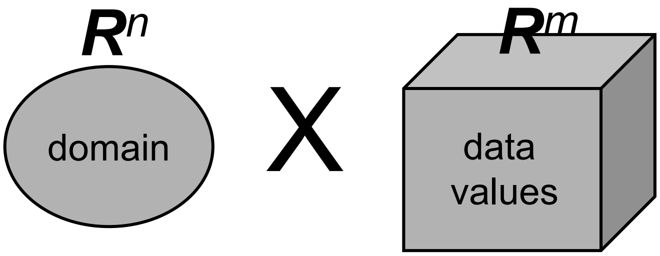
---

2.4. Data Representation

Overview of data attributes:

- Data domain
  - 0D, 1D, 2D, 3D, ...
- Data type
  - Scalar, vector, tensor, multivariate
- Range of values
  - Qualitative (non-metric scale)
    - Ordinal (order relation exists)
    - Nominal (no order relation exists: pairs are equal or not equal)
  - Quantitative
- Data structure

2.4. Data Representation



$\boldsymbol{R}^n$ domain X $\boldsymbol{R}^m$ data values

independent variables

dependent variables

scientific data $\subseteq \boldsymbol{R}^{n+m}$

2.4. Data Representation

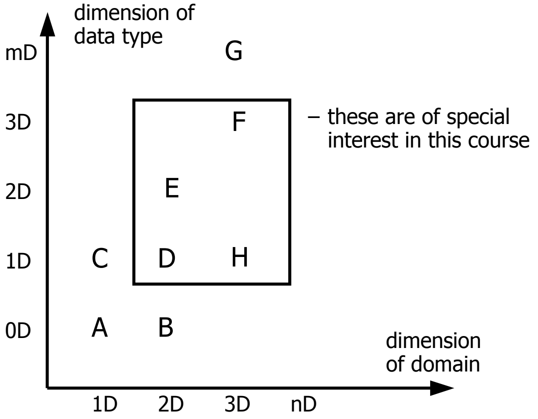
- Discrete representations
  - The objects we want to visualize are often ‘continuous’
  - But in most cases, the visualization data is given only at discrete locations in space and/or time
  - Discrete structures consist of samples, from which grids/meshes consisting of cells are generated

- Primitives in multi dimensions

dimension	cell	mesh
0D	points	
1D	lines (edges)	polyline(–gon)
2D	triangles, quadrilaterals (rectangles)	2D mesh
3D	tetrahedra, prisms, hexahedra	3D mesh

2.4. Data Representation

- Classification of visualization techniques according to
  - Dimension of the domain of the problem (independent params)
  - Type and dimension of the data to be visualized (dependent params)



– these are of special interest in this course

Examples:

A: gas station along a road
B: map of cholera in London
C: temperature along a rod
D: height field of a continent
E: 2D air flow
F: 3D air flow in the atmosphere
G: stress tensor in a mechanical part
H: ozon concentration in the atmosphere

2.5. Domain

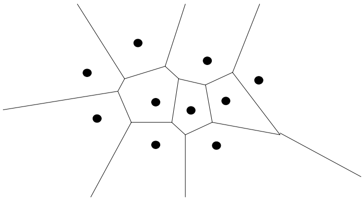
- The (geometric) shape of the domain is determined by the positions of sample points
- Domain is characterized by
  - Dimension
  - Influence
  - Structure

2.5. Domain

- Influence of data points
  - Values at sample points influence the data distribution in a certain region around these samples
  - To reconstruct the data at arbitrary points within the domain, the distribution of all samples has to be calculated
- Point influence
  - Only influence on point itself
- Local influence
  - Only within a certain region
    - Voronoi-diagram
    - Cell-wise interpolation (see later in course)
- Global influence
  - Each sample might influence any other point within the domain
    - Material properties for whole object
    - Scattered data interpolation

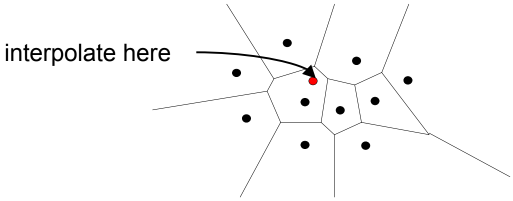
2.5. Domain

- Voronoi-diagram
  - Construct a region around each sample point that covers all points that are closer to that sample than to every other sample
  - Each point within a certain region gets assigned the value of the sample point



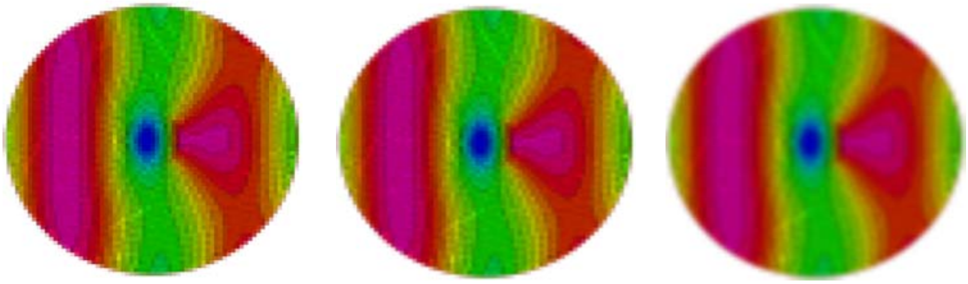
2.5. Domain

- Scattered data interpolation
  - At each point the weighted average of all sample points in the domain is computed
  - Weighting functions determine the support of each sample point
    - Radial basis functions simulate decreasing influence with increasing distance from samples
  - Schemes might be non-interpolating and expensive in terms of numerical operations



2.5. Domain

- Example
  - Radial basis functions with increasing support



2.6. Data Structures

- Requirements:
  - Convenience of access
  - Space efficiency
  - Lossless vs. lossy
  - Portability
    - binary – less portable, more space/time efficient
    - text – human readable, portable, less space/time efficient

- Definition
  - If points are arbitrarily distributed and no connectivity exists between them, the data is called scattered
  - Otherwise, the data is composed of cells bounded by grid lines
  - **Topology** specifies the structure (**connectivity**) of the data
  - **Geometry** specifies the **position** of the data

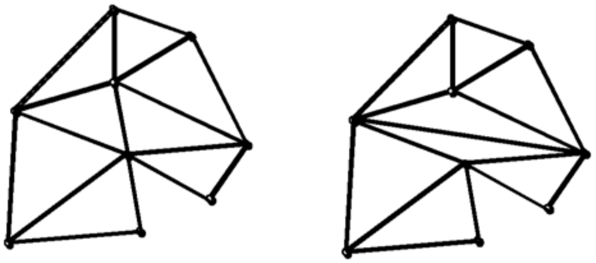
2.6. Data Structures

- Some definitions concerning topology and geometry
  - In topology qualitative questions about geometrical structures are the main concern.
    - Does it have any holes in it ?
    - Is it all connected together
    - Can it be separated into parts ?

- Underground map does not tell you how far one station is from the other, but rather how the lines are connected (topological map)

2.6. Data Structures

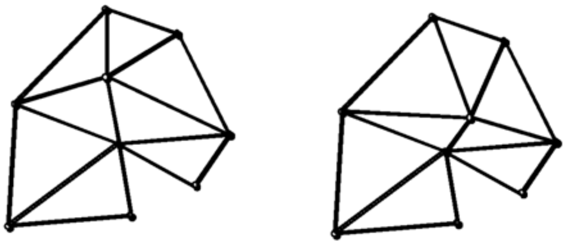
- Topology
  - Properties of geometric shapes that remain unchanged even when under distortion



Same geometry (vertex positions), different topology (connectivity)

2.6. Data Structures

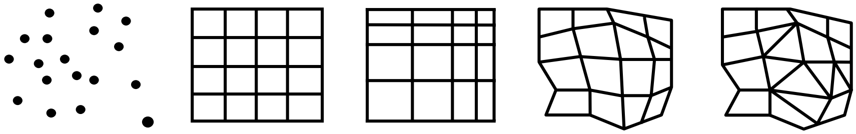
- Topologically equivalent
  - Things that can be transformed into each other by stretching and squeezing, without tearing or sticking together bits which were previously separated



topologically equivalent

2.6. Data Structures

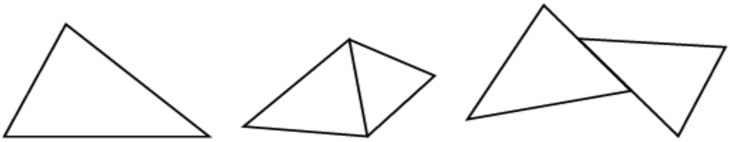
- Grid types
  - Grids differ substantially in the simplicial elements (or cells) they are constructed from and in the way the inherent topological information is given



scattered uniform rectilinear structured unstructured

2.6. Data Structures

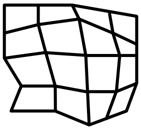
- A simplex in R^n
  - the convex hull of n+1 affinely independent points
  - 0: points, 1: lines, 2: triangles, 3: tetrahedra
- Partitions via simplices are called triangulations
- Simplical complex is a collection of simplices with:
  - Every face of an element of *C* is also in *C*
  - The intersection of two elements of *C* is empty or it is a face of both elements
- Simplical complex is a space with a triangulation



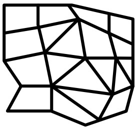
Simplical complexes Not a simplical complex

2.6. Data Structures

- Structured and unstructured grids can be distinguished by the way the elements or cells meet
- Structured grids
  - Have a regular topology and regular / irregular geometry
- Unstructured grids
  - Have irregular topology and geometry



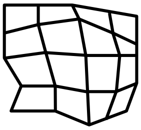
structured



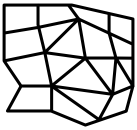
unstructured

2.6. Data Structures

- Characteristics of structured grids
  - Easier to compute with
  - Often composed of sets of connected parallelograms (hexahedra), with cells being equal or distorted with respect to (non-linear) transformations
  - May require more elements or badly shaped elements in order to precisely cover the underlying domain
  - Topology is represented implicitly by *n*-vector of dimensions
  - Geometry is represented explicitly by an array of points
  - Every interior point has the same number of neighbors



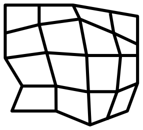
structured

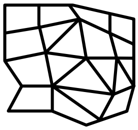


unstructured

2.6. Data Structures

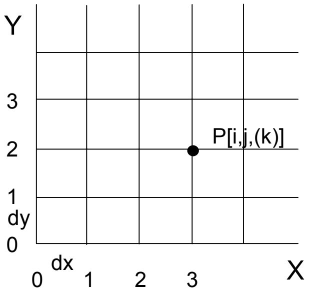
- If no implicit topological (connectivity) information is given the grids are termed unstructured grids
  - Unstructured grids are often computed using quadtrees (recursive domain partitioning for data clustering), or by triangulation of points sets
  - The task is often to create a grid from scattered points
- Characteristics of unstructured grids
  - Grid point geometry **and** connectivity must be stored
  - Dedicated data structures needed to allow for efficient traversal and thus data retrieval
  - Often composed of triangles or tetrahedra
  - Less elements are needed to cover the domain


structured


unstructured

2.6. Data Structures

- Cartesian or equidistant grids
  - Structured grid
  - Cells and points are numbered sequentially with respect to increasing X, then Y, then Z, or vice versa
  - Number of points = Nx•Ny•Nz
  - Number of cells = (Nx-1)•(Ny-1)•(Nz-1)



dx = dy = dz

2.6. Data Structures

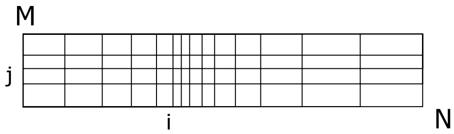
- Cartesian grids
  - Vertex positions are given implicitly from [i,j,k]:
    - P[i,j,k].x = origin + i • dx
    - P[i,j,k].y = origin + j • dy
    - P[i,j,k].z = origin + k • dz
  - Global vertex index I[i,j,k] = k•Ny•Nx + j•Nx + i
    - k = I / (Ny•Nx)
    - j = (I % (Ny•Nx)) / Nx
    - i = (I % (Ny•Nx) % Nx)
  - Global index allows for linear storage scheme
    - Wrong access pattern might destroy cache coherence

2.6. Data Structures

- Uniform grids
  - Similar to Cartesian grids
  - Consist of equal cells but with different resolution in at least one dimension (dx ≠ dy (≠ dz))
  - Spacing between grid points is constant in each dimension -> same indexing scheme as for Cartesian grids
  - Most likely to occur in applications where the data is generated by a 3D imaging device providing different sampling rates in each dimension
  - Typical example: medical volume data consisting of slice images
    - Slice images with square pixels (dx = dy)
    - Larger slice distance (dz > dx = dy)

2.6. Data Structures

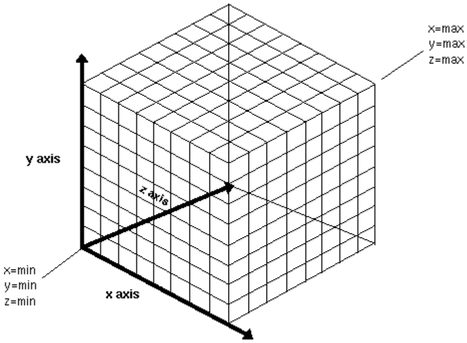
- Rectilinear grids
  - Topology is still regular but irregular spacing between grid points
    - Non-linear scaling of positions along either axis
    - Spacing, x_coord[L], y_coord[M], z_coord[N], must be stored explicitly
  - Topology is still implicit



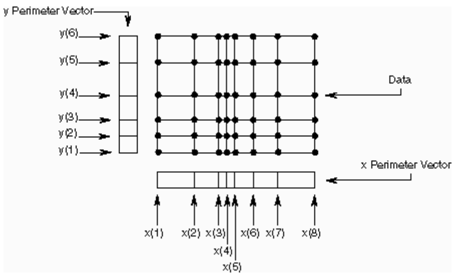
2.6. Data Structures

- Iris Explorer data structures

3D uniform lattice

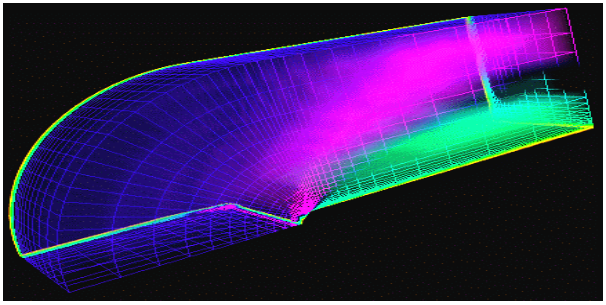


2D perimeter lattice



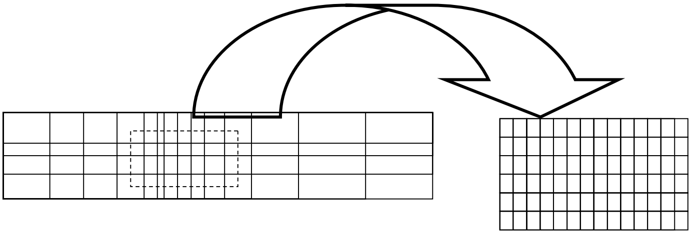
2.6. Data Structures

- Curvilinear grids
  - Topology is still regular but irregular spacing between grid points
    - Positions are non-linearly transformed
  - Topology is still implicit, but vertex positions are explicitly stored
    - x_coord[L,M,N]
    - y_coord[L,M,N]
    - z_coord[L,M,N]
  - Geometric structure might result in concave grids



2.6. Data Structures

- Multigrids
  - Focus in arbitrary areas to avoid wasted detail
  - “blow up” regions of interest, i.e. finer grid
  - Difficulties in the boundary region (i.e. interpolation)



2.6. Data Structures

- Characteristics of structured grids
  - Structured grids can be stored in a 2D / 3D array
  - Arbitrary samples can be directly accessed by indexing a particular entry in the array
  - Topological information is implicitly coded
    - Direct access to adjacent elements at random
  - Cartesian, uniform, and rectilinear grids are necessarily convex
  - Their visibility ordering of elements with respect to any viewing direction is given implicitly
  - Their rigid layout prohibits the geometric structure to adapt to local features
  - Curvilinear grids reveal a much more flexible alternative to model arbitrarily shaped objects
  - However, this flexibility in the design of the geometric shape makes the sorting of grid elements a more complex procedure

---

2.6. Data Structures

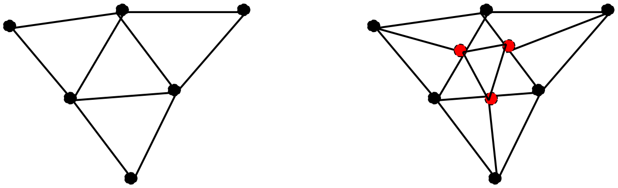
- Typical implementation of structured grids

```
DataType *data = new DataType[Nx•Ny•Nz];
val = data[i•(Ny•Nz) + j•Nz + k];
```

… code for geometry …

2.6. Data Structures

- Unstructured grids
  - Composed of arbitrarily positioned and connected elements
  - Can be composed of one unique element type or they can be hybrid (tetras, hexas, prisms)
  - Triangle meshes in 2D and tetrahedral grids in 3D are most common
  - Can adapt to local features (small vs. large cells)
  - Can be refined adaptively
  - Simple linear interpolation in simplices



2.6. Data Structures

- Typical implementations of unstructured grids
  - Direct form

Coords for vertex 1

```
x1,y1,(z1)
x2,y2,(z2)   face 1
x3,y3,(z3)
x2,y2,(z2)
x3,y3,(z3)   face 2
x4,y4,(z4)

...
```

2D

```
struct face
  float verts[3][2]
  DataType val;
```

3D

```
struct face
  float verts[3][3]
  DataType val;
```

- Additionally store the data values
- Problems: storage space, redundancy

2.6. Data Structures

- Typical implementations of unstructured grids
  - Indirect form

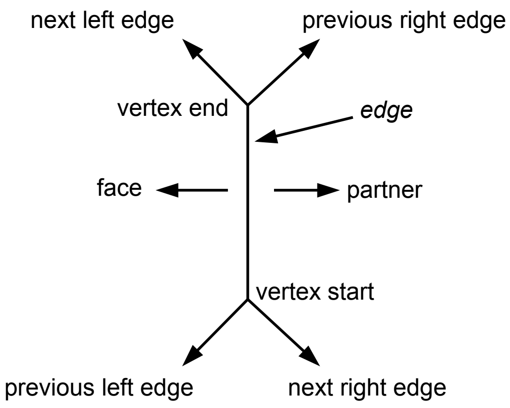
Coords for vertex 1

vertex list	face list
`x1,y1,(z1)`	`1,2,3`
`x2,y2,(z2)`	`1,2,4`
`x3,y3,(z3)`	`3,2,4`
`x4,y4,(z4)`	
...	...

  - Indexed face set
  - More efficient than direct approach in terms of memory requirements
  - But still have to do global search to find local information (i.e. what faces share an edge)

2.6. Data Structures

- Typical implementations of unstructured grids
  - Winged-edge data structure [Baumgart 1975]



next left edge

previous right edge

vertex end

edge

face

partner

vertex start

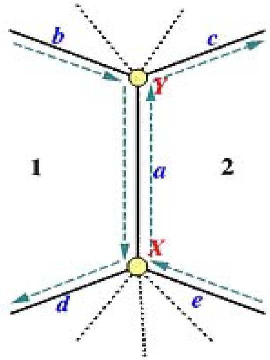
previous left edge

next right edge

counter-
clockwise
orientation

2.6. Data Structures

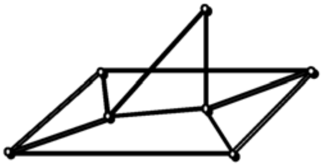
- Winged-edge data structure
  - Edge-based data structure, allows to answer queries
    - Faces sharing an edge
    - Faces sharing a vertex
    - Walk around edges of face
  - Stores for every vertex a pointer to an arbitrary edge that is incident to it
  - Stores for every face a pointer to an edge on its boundary
  - Implicit assumption:
    - Every edge has at most two faces which meet at edge ⇔ two-manifold topology



2.6. Data Structures

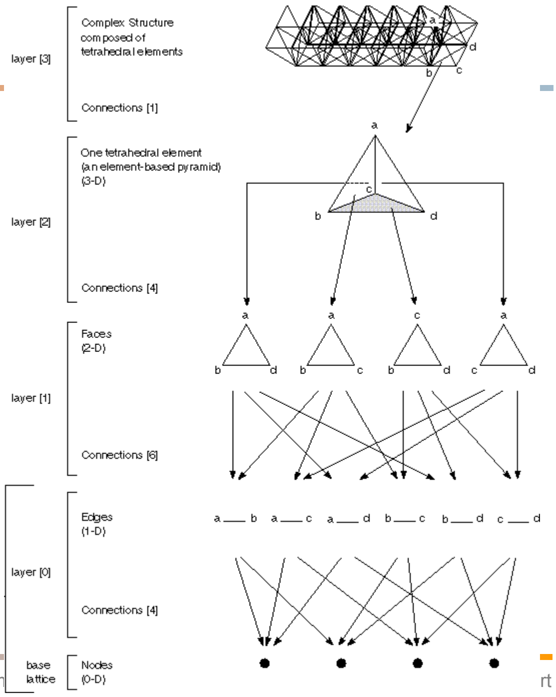
- Manifold meshes
  - 2-manifold is a surface where at every point on the surface a surrounding area can be found that looks like a disk
  - Everything can be flattened out to a plane
  - Sharp creases and edges are possible needs more than one normal per vertex

  - Example for an non-manifold:



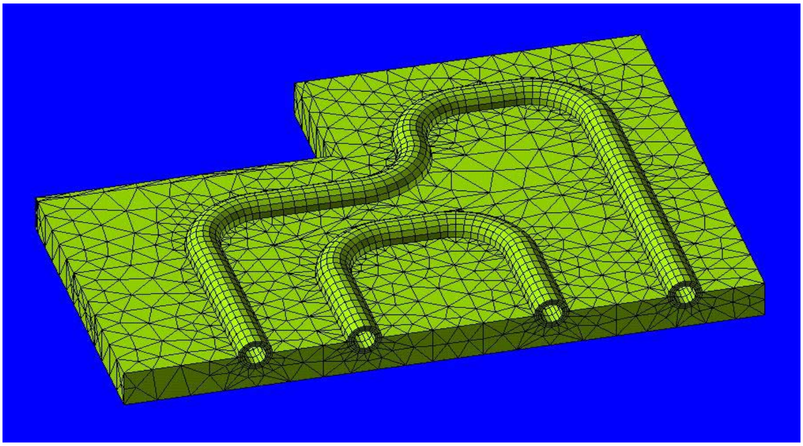
2.6. Data Structures

- Iris Explorer tetrahedral grid

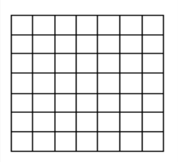


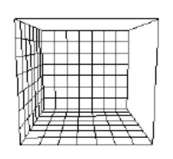
2.6. Data Structures

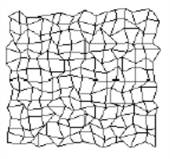
- Hybrid grids
  - Combination of different grid types

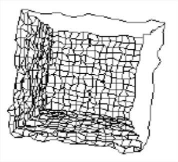


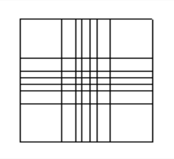
2.6. Data Structures

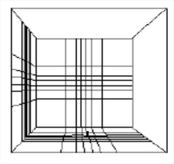

2D-Regular Grid


3D-Regular Grid

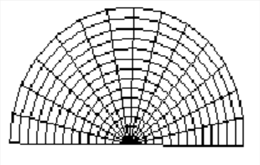

2D-Irregular Grid

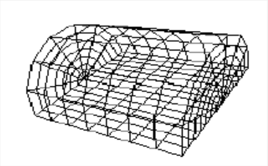

3D-Irregular Grid

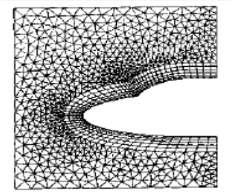

2D-Block-Structured Grid


3D-Block-Structured Grid

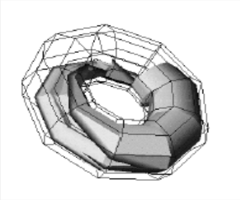
2.6. Data Structures


2D-Structured Grid

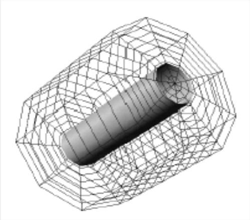

3D-Structured Grid


2D-Hybrid Grid

2.6. Data Structures



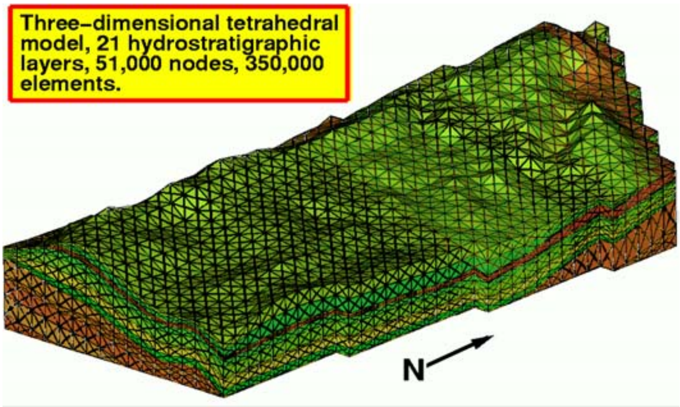
3D-Structured Grid " Torus"



3D-structured Grid "full cylinder"

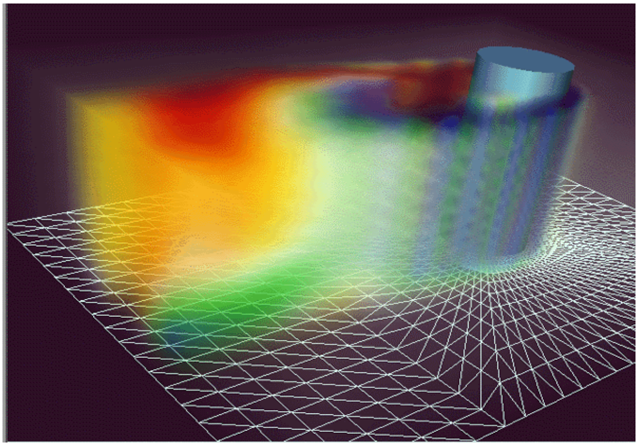
2.6. Data Structures

- Example



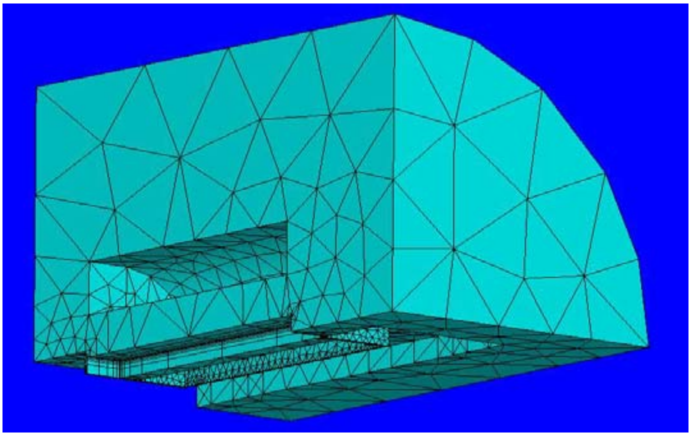
2.6. Data Structures

- Example



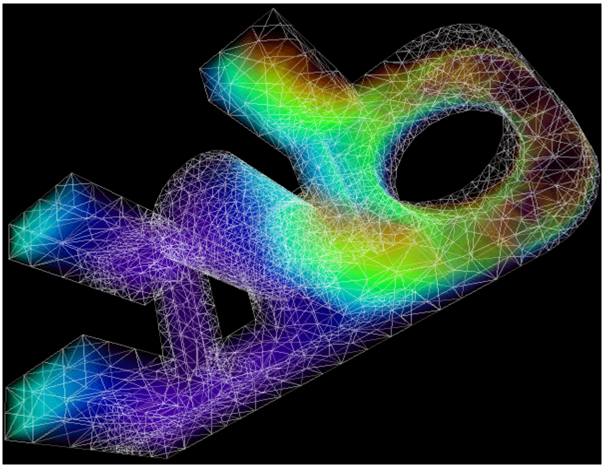
2.6. Data Structures

- Example



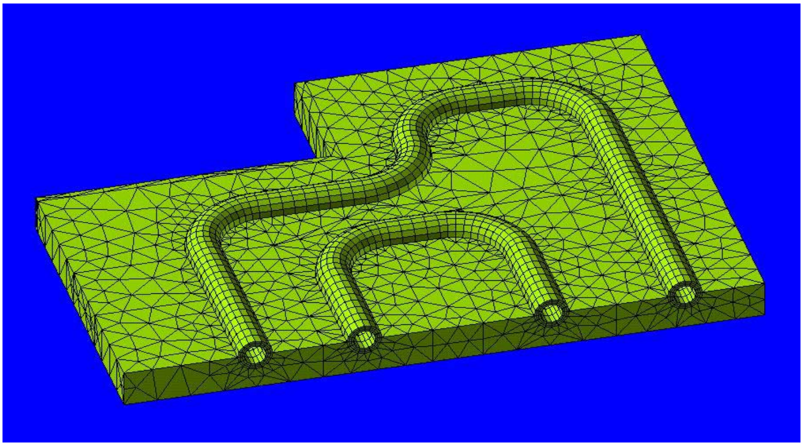
2.6. Data Structures

- Example



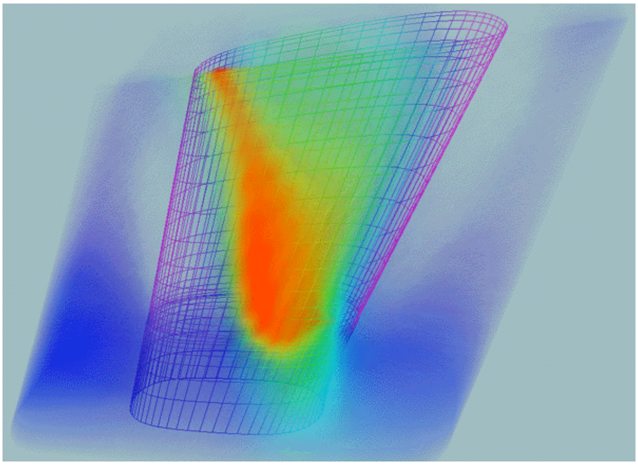
2.6. Data Structures

- Example



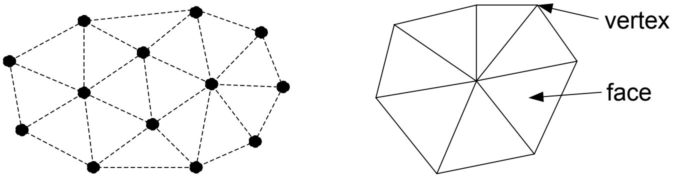
2.6. Data Structures

- Example



2.6. Data Structures

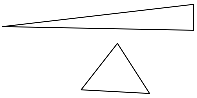
- Scattered data
  - Irregularly distributed positions without connectivity information
  - To get connectivity find a “good” triangulation (triangular/tetrahedral mesh with scattered points as vertices)

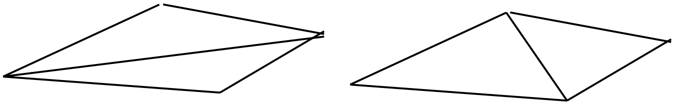


2.6. Data Structures

- For a set of points there are many possible triangulations
  - A measure for the quality of a triangulation is the *aspect ratio* of the so-defined triangles
  - Avoid long, thin ones
  - Delaunay triangulation (later in the course)

$\dfrac{\textit{radius of incircle}}{\textit{radius of circumcircle}}$ *or* *maximal/minimal angle in triangle*





2.7. Data Values

- Characteristics of data values
  - Range of values
  - Data type (scalar, vector, tensor data; kind of discretization)
  - Dimension (number of components)
  - Error (variance)
  - Structure of the data

2.7. Data Values

- Range of values
    - Qualitative
        - Non-metric
        - Ordinal (order along a scale)
        - Nominal (no order)
    - Quantitative
        - Metric scale
        - Discrete
        - Continuous

2.7. Data Values

- Data types
    - Scalar data
      is given by a function $f(x_1,\ldots,x_n):\mathbf{R}^n \to \mathbf{R}$ with n independent variables x_i
    - Vector data
      represent direction and magnitude and
      is given by a n-tupel $(f_1,\ldots,f_n)$ with $f_k=f_k(x_1,\ldots,x_n)$, $n \geq 2$ and $1 \leq k \leq n$
    - Tensor data
      for a tensor of level k is given by $t_{i1,i2,\ldots,ik}(x_1,\ldots,x_n)$
      a tensor of level 1 is a vector, a tensor of level 2 is a matrix, …

- Structure of the data
    - Sequential (in the form of a list)
    - Relational (as table)
    - Hierarchical (tree structure)
    - Network structure

2.7. Data Classification

- Classification according to Bergeron & Grienstein,1989:

- L^n_m *m*-dimensional data on an *n*-dimensional grid
- Examples for *m*-dimensional data
  - On arbitrary positions (L^0_m)
  - On a line (L^1_m)
  - On a surface (L^2_m)
  - On a (uniform) 3D grid (L^3_m)
  - On a (uniform) *n*-dimensional grid (L^n_m)

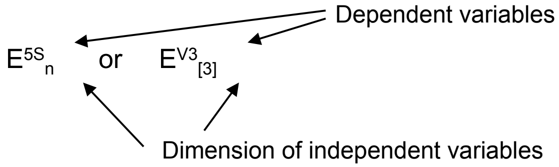
- Important aspects of data and grid types are missing

2.7. Data Classification

- Classification according to Brodlie 1992:

- Underlying Field: domain of the data
- Visualizing entity (E)
- E is a function defined by domain and range of data
- Independent variables: dimension and influence
  [ ]: data defined on region, { }: data enumerated
- Dependent variables: dimension and data type

- Examples



E^{5S}_{n} or $E^{V3}_{[3]}$

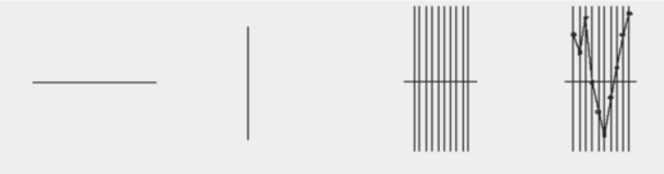
Dependent variables

Dimension of independent variables

2.7. Data Classification

- Classification via *fiber bundles* according to Butler 1989:

- Fiber bundle:
  - base space: independent variables
  - fiber space: dependent variables
- Definition of sections in fiber space
- Connection to differential geometry



fiber space base space fiber bundle section

2.7. Data Classification

- Specification according to Wong 1997

- Dimension of the data values: dependent variables *v*
- Dimension of domain: independent variables *d*

- Data with *n* independent variables and *m* dependent variables:

*n***d***m***v**

2.7. Data Classification

- Example:
  Unordered set of points with scalar values
- Bergeron & Grienstein L^{0}_{1}
- Brodlie $E^{S}_{\{0\}}$
- Butler base = set, fiber = float:[-∞, ∞]
- Wong $\mathit{0}\mathbf{d}\mathit{1}\mathbf{v}$

2.7. Data Classification

- Example:
  Ordered set of points with scalar values
- Bergeron & Grienstein L^{0}_{1}
- Brodlie $E^{S}_{[0]}$
- Butler base = ordered set, fiber = float:[-∞, ∞]
- Wong $\mathit{0}\mathbf{d}\mathit{1}\mathbf{v}$

2.7. Data Classification

- Example:
  Scalar volume data set on a uniform grid
- Bergeron & Grienstein L^3_1
- Brodlie E^S_3
- Butler base = 3D-reg-grid, fiber = char:[0,255]
- Wong **3d***1***v**

2.7. Data Classification

- Example:
  Flow data on a curvilinear grid
- Bergeron & Grienstein L^3_3
- Brodlie E^{V3}_3
- Butler base = 3D-curvilin-grid, fiber = float^3:[-∞, ∞]^3
- Wong **3d***3***v**

2.7. Data Classification

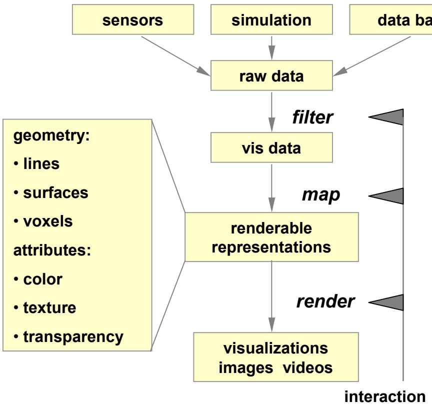
- Example:
  3D volume with 3 scalar and 2 vector data values
- Bergeron & Grienstein L^{3}_{9}
- Brodlie E^{3S2V3}_{3}
- Butler base = 3D-reg-grid, fiber = float x float x float x float^3 x float^3
- Wong $3\mathbf{d}9\mathbf{v}$

2.7 Time dependency

- Discretization in time with constant or variable time steps
- Time dependence of
  - Data only (grid remains constant)
    e.g. time series of CT data, CFD of an airplane
  - Data and grid geometry (topology remains constant)
    e.g. crashworthiness of cars
  - Data, grid geometry and topology
    e.g. engine simulation with moving piston

2.7 Visualization Pipeline Revisited

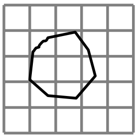
visualization pipeline



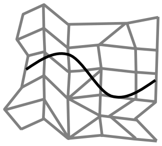
mapping – classification

	scalar	vector	tensor/MV
3D	volume rend. isosurfaces	stream ribbons topology	glyphs icons
2D	height fields color coding	arrows LIC	attributes ymbols
1D			

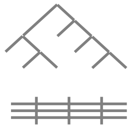
different grid types → different algorithms



3D scalar fields
cartesian medical
datasets



3D vector fields
un/structured
CFD



trees, graphs, tables,
data bases
InfoVis