Basics

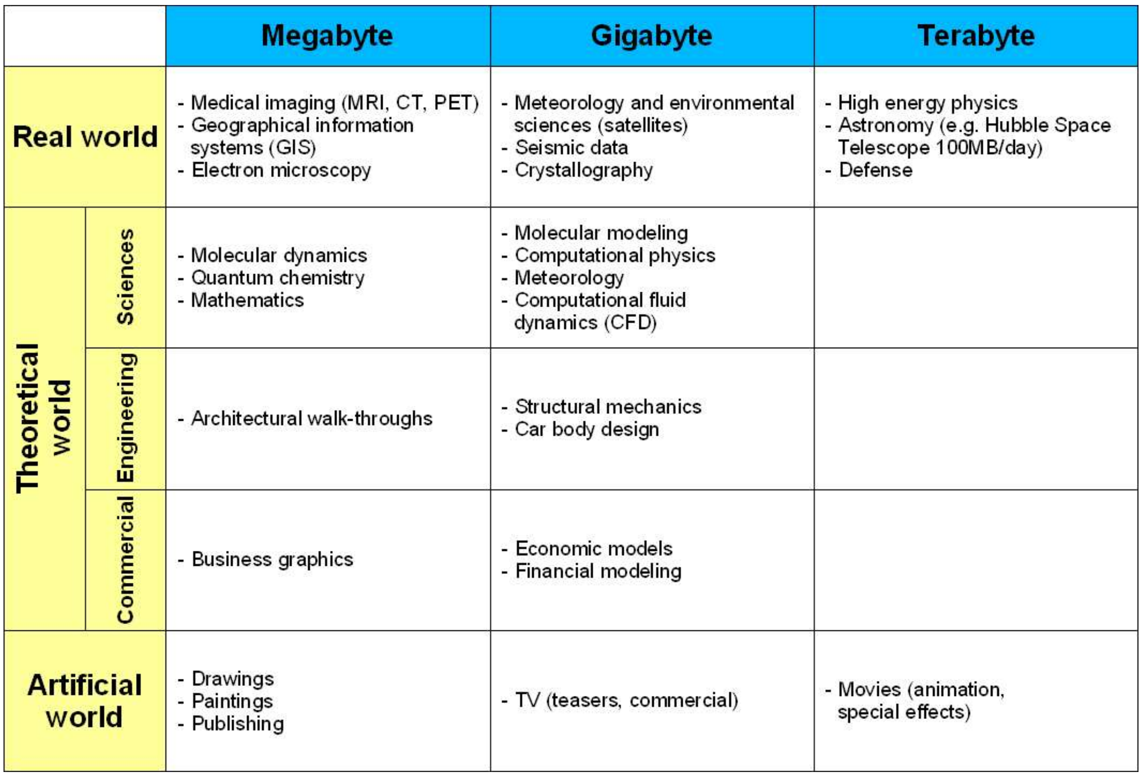
1.Data Sources

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. Therefore, techniques to efficiently visualize large-scale data sets and new data types need to be developed.

Data may come from different sources:

- Real world (measurement and observation)
- Theoretical world (data is calculated with mathematical and technical models)
- Artificial world (data is designed)

The size of the amount of data depends on the number of parameters of interest, the number of values per parameter, and the number of points which are considered.
The following table shows the possible data sources and the amount of data created.



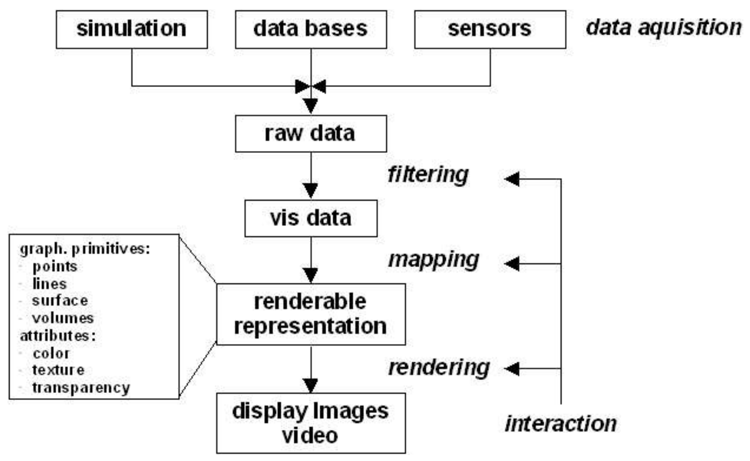
		Megabyte	Gigabyte	Terabyte
Real world		- Medical imaging (MRI, CT, PET) - Geographical information systems (GIS) - Electron microscopy	- Meteorology and environmental sciences (satellites) - Seismic data - Crystallography	- High energy physics - Astronomy (e.g. Hubble Space Telescope 100MB/day) - Defense
Theoretical world	Sciences	- Molecular dynamics - Quantum chemistry - Mathematics	- Molecular modeling - Computational physics - Meteorology - Computational fluid dynamics (CFD)	
	Engineering	- Architectural walk-throughs	- Structural mechanics - Car body design	
	Commercial	- Business graphics	- Economic models - Financial modeling	
Artificial world		- Drawings - Paintings - Publishing	- TV (teasers, commercial)	- Movies (animation, special effects)

Data Sources

2.The Visualization process

2.1.The visualization pipeline

The aim of the visualization process is to represent non-geometric data by images. This process consists of several steps which can be arranged in the so called visualization pipeline.

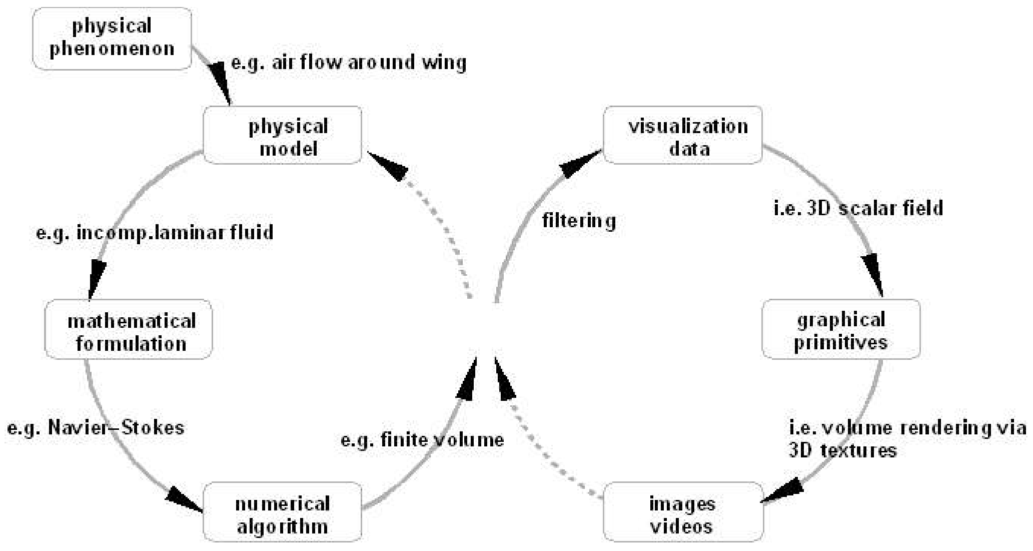


The visualization pipeline

The filtering step maps data to data. The raw data is prepared for the following steps of the visualization pipeline. One example for filtering is a data format conversion into a specific format. If only parts of the data are to be visualized a clipping or slicing can be applied. Sometimes denoising or resampling can be necessary. In case of too few measured values interpolation makes it possible to generate values between the sample points. Furthermore, data can be classified or segmented in this step.
In the mapping step, which is the main part of the visualization process, data is mapped to graphical primitives and their attributes (e.g. color). Examples for mapping techniques are:

scalar field	→	isosurface
2D field	→	height field
vector field	→	vectors
tensor field	→	glyphs
3D field	→	volume

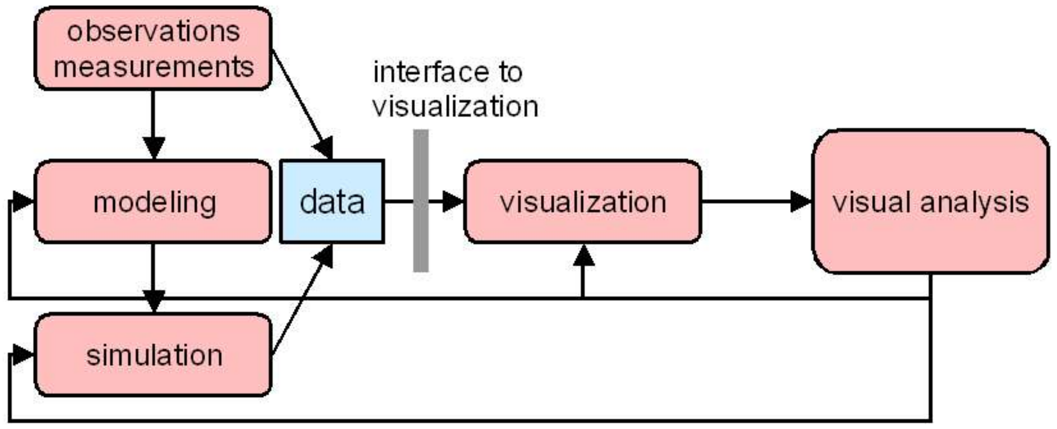
These mapping techniques will be discussed in detail within this course.
In the rendering step geometry data is mapped to image data. Shadows, lighting, and shading are used to produce images with more realism.
The following example shows the simulation of the flow within a fluid around a wing. The simulation is tightly coupled with the visualization cycle.



Simulation of the flow within a fluid around a wing

2.2. Visualization scenarios

The above illustration shows the cycle of visual analysis. Starting with observations and measurements, hypotheses and mathematical models can be established. Based on this models simulations can be performed which generate a large amount of data.
Both the calculated and the measured data can now be processed and provided for visual analysis. Since the aim of the visual analysis is to understand the data including its underlying structures, feedback (coupling) to all of the three processes (visualization, simulation, modeling) is necessary.

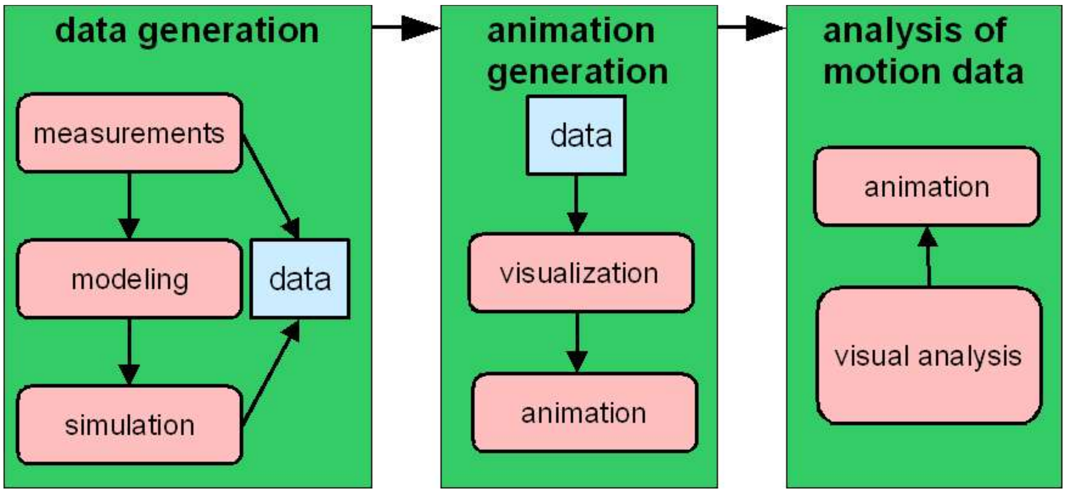


The cycle of visualization analysis

Especially interaction in simulation and modeling causes a strong demand for fast processing. Therefore four different visualization scenarios were established:

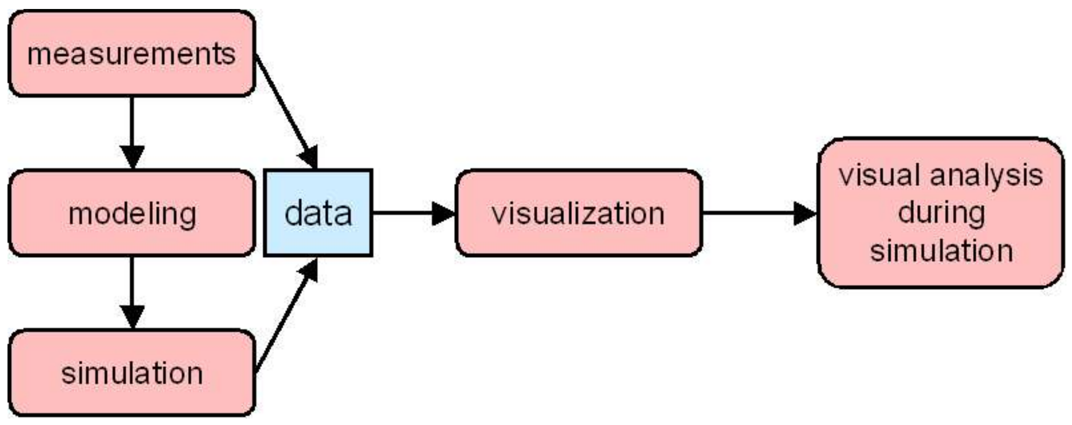
- motion mode
- tracking
- interactive post processing
- interactive steering

The **motion mode** is based on three separated steps. The first step consists of the whole process of data generation including modeling and simulation. The visualization process is performed in the second step. The result of the visualization is not a single image but a animation which is viewed by the user in the visual analysis.



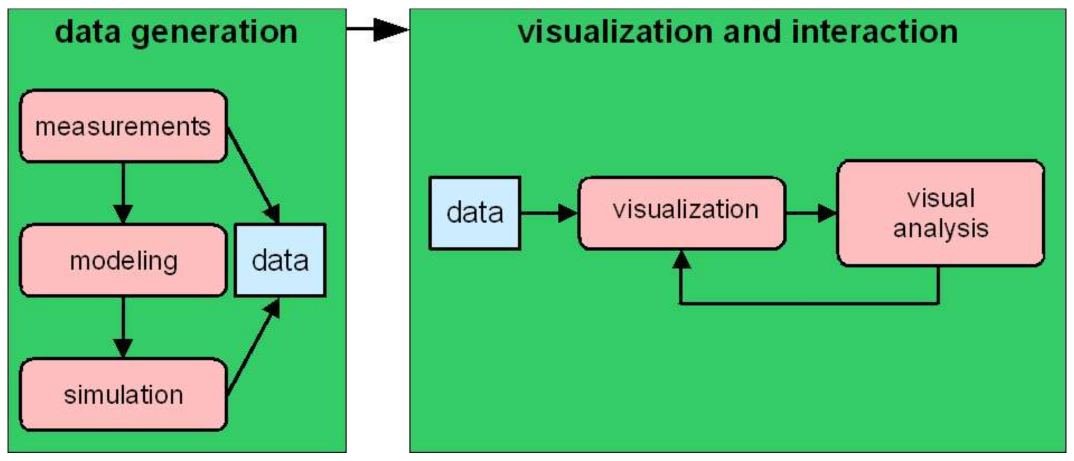
Motion mode

In the visualization scenario **tracking** the analysis process is directly connected to modeling and simulation. As soon as data is measured or calculated it is processed and offered to the analysis.



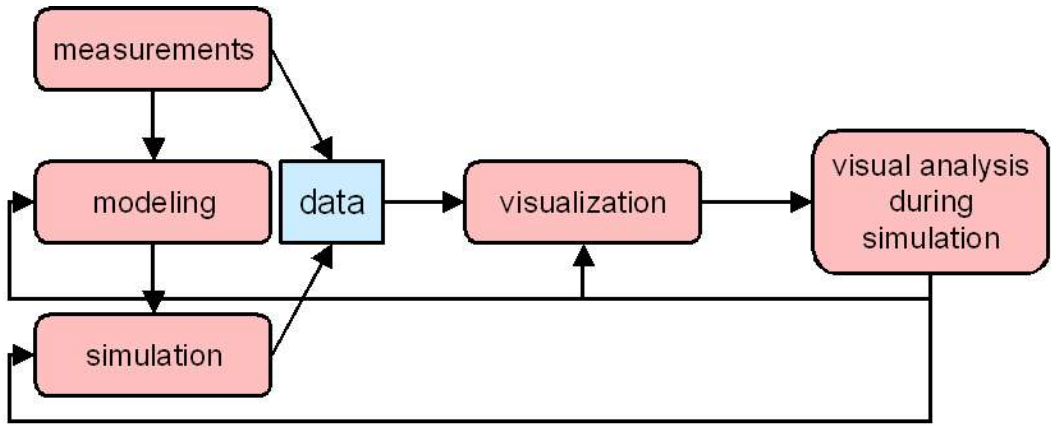
Tracking

In **interactive post processing** the visualization process is separated from data generation, modeling, and simulation.



Interactive post processing

Only with **interactive steering** the user can interact with the three processes modeling, simulation, and visualization.



Interactive steering

3.Sources of Error

To minimize the possibilities for errors in the different steps of the visualization process the following questions and hints are useful.
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 ?
- Selecting the “right” variable: Does this variable allow an analysis of the relevant features ?
- Functional model for resampling: What kind of information do we introduce by interpolation and approximation ?

Mapping:

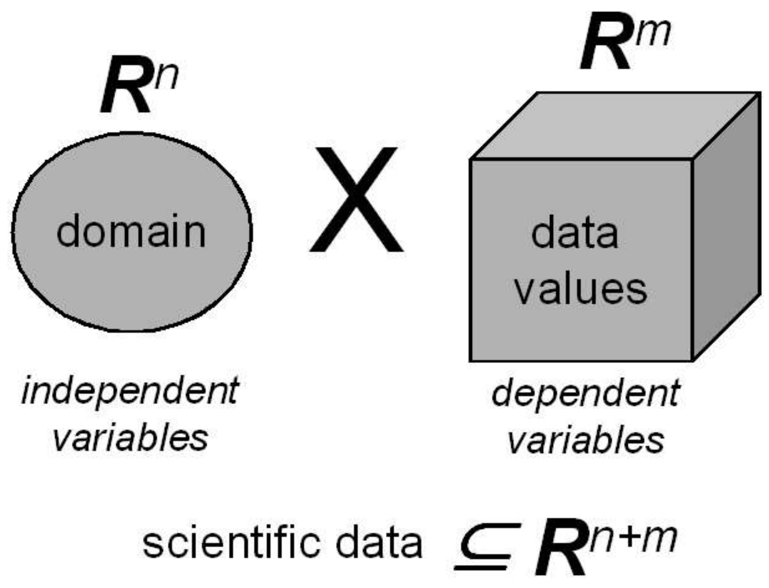
- Are we choosing the graphical primitives appropriately in order to depict the kind of information we want to get out of the data ?
- Think of some real world analogue.

Rendering:

- The need for interactive rendering often determines the chosen abstraction level.
- Consider limitations of the underlying display technology: ergo, color quantization.
- Carefully add “realism”: (the most realistic image is not necessarily the most informative one).

4.Data Representation

Data can be considered as a subset of the space $\mathbb{R}^{n+m}$, assuming that there are n independent and m dependent variables.



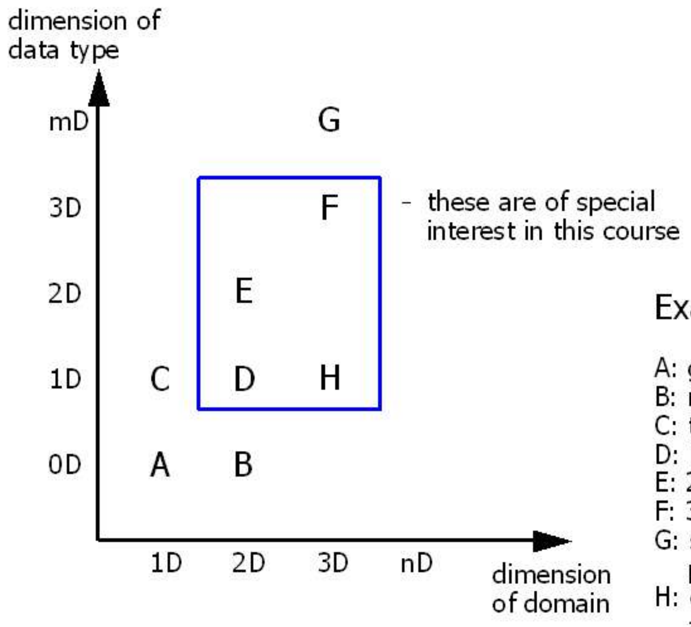
Data representation

$\mathbb{R}^{n+m}$

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. The following table shows graphical primitives which form meshes in spaces of different dimension.

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

The visualization techniques are classified according to both the dimension of the domain of the problem (independent parameters) and the type and dimension of the data to be visualized (dependent parameters).
Examples for visualization problems:



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

Visualization problems

≈

≈ ≈

5.The Domain

The space in which the data is considered is called domain. The points for which data is available are called sample points. For the visualization the following characteristics are of interest:

- Dimension of the domain
- Influence
- Structure (connection between sample points)

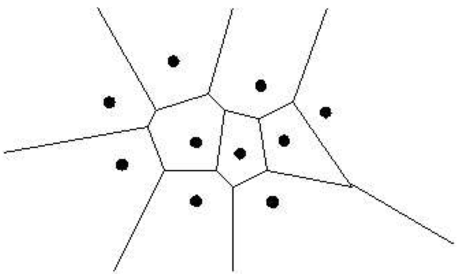
The (geometric) shape of the domain is determined by the positions of sample points.
Dimension of the domain:
The independent variables can be discrete or continuous. If there are more than two dimensions a projection is necessary. This may cause occlusion and ambiguity.
Influence:
The values at the 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.
There are different types of influence:

- Point influence (the data value influences only the point itself)
- Local influence (the data value influences a certain region)
- Global influence (each sample might influence any other point within the domain)

To obtain a data value at each point of the domain for a set of arbitrarily distributed sample points a *cell-wise interpolation* or the *Voronoi-diagram* can be used.
The **Voronoi-diagram** is a decomposition of the domain. It is constructed by creating a region around each sample point, which can be considered as the area of influence of that sample point. Each region contains only a single sample point.
A certain region contains all points that are closer to that sample than to every other sample. This is a separation of the n-dimensional space $\Re^n$ into regions R_i where:

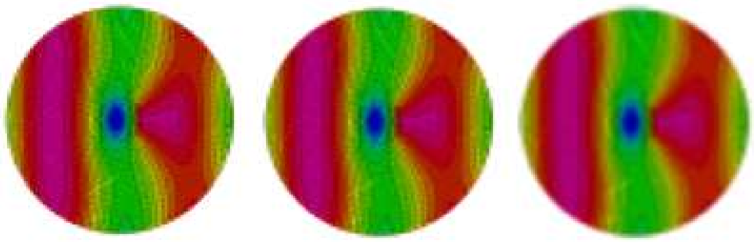
$$R_i = \{X \in \Re^n : \|X - P_j\| < \|X - P_k\| \forall k \neq j \wedge k \leq m\}$$

and $P_j (j=1,2,\ldots,m)$ is the sample point of the region R_i
Each point in a certain region is assigned the value of the corresponding sample point.
An example for a Voronoi-diagram:



A Voronoi-diagram

The **scattered data interpolation** is a solution for local or global influence problems. Here, a function is constructed which gives a value at each point and especially for the sample points returns exactly the data value. At each point the weighted average of all sample points in the domain is computed. The support of each sample point to the value is determined by weighting functions. For this purpose radial basis functions, which will be discussed later in the course, may be used to simulate decreasing influence with increasing distance from samples. In the case of local influence the value of the weighting function is zero for a certain sample point, if the distance between the interpolation point and that sample point is greater than the radius of the circle of influence.


Radial basis functions with increasing support.

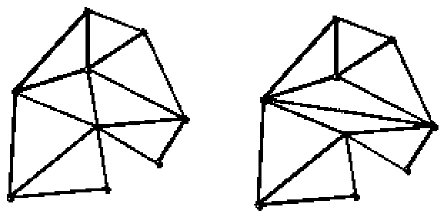
6.Data Structures

There are some important requirements for data structures. Of course a convenient access should be possible but the structure should also be space efficient. When choosing a data structure portability should also be taken into account. For example. a binary file is less portable but more time efficient than a text file. The advantage of the text file is that it is human readable.

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. The topology specifies the structure (connectivity) of the data.

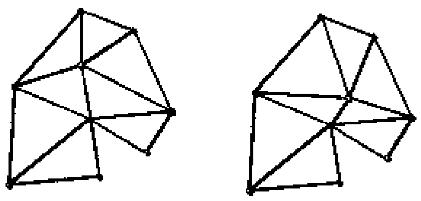
The terms topology and geometry have to be distinguished. In topology qualitative questions about geometrical structures are the main concern. An example is a underground map which does not tell you how far one station is from the other, but rather how the lines are connected. Topology means all properties of a geometric shape that remain unchanged even when the image is smoothly distorted.

The following examples show the difference between geometrical and topological equivalence.


Geometrical equivalence

These two shapes have the same geometry (vertex positions) but different topology.

Topological equivalent shapes can be transformed into each other by stretching and squeezing, without tearing or sticking together bits which were previously separated.


Topological equivalence

These two shapes are topologically equivalent.

6.1.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.

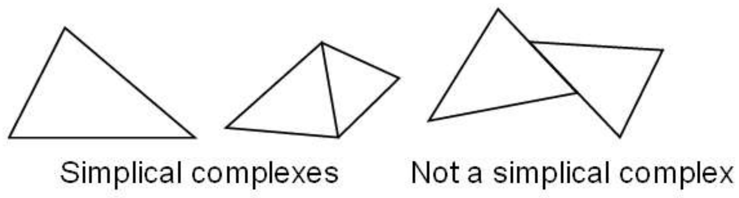
A simplex in $\Re^n$ is the convex hull of n+1 affinely independent points. Partitions via simplices are called triangulations.

A simplicial complex is a collection C 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

A simplical complex is a space with a triangulation.

Example:



Simplical complexes

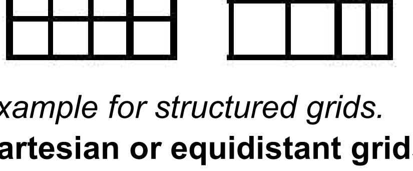
Grids can be classified as being structured or unstructured. These two types are distinguished by the way the elements or cells meet. Structured grids have a regular topology and regular or irregular geometry. Unstructured grids have both irregular topology and geometry.

A structured grid is often composed of sets of connected parallelograms (hexahedra), with cells being equal or distorted with respect to (non-linear) transformations. They may require more elements or badly shaped elements in order to precisely cover the underlying domain. Its topology is represented implicitly by an n-vector of dimensions. Its geometry is represented explicitly by an array of points. Every interior point has the same number of neighbors.

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. For unstructured grids grid points and connectivity must be stored. Dedicated data structures are needed to allow for efficient traversal and thus data retrieval. An unstructured grid is often composed of triangles or tetrahedra. Fewer elements are needed to cover the domain.

In the following sections we consider a 3-dimensional domain.

6.1.1.Structured grids



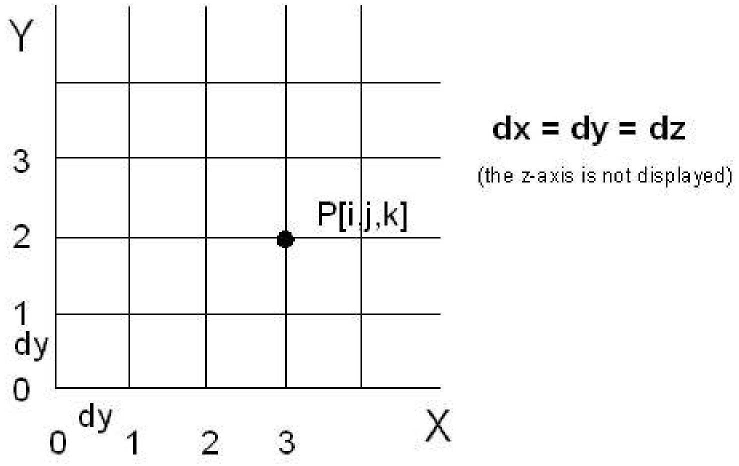
Example for structured grids.

Cartesian or equidistant grids:

The cells and points of this type of grid are numbered sequentially with respect to increasing X, then Y, then Z, or vice versa. The number of sample points N_p and the number of cells N_c are given by:

$$N_p = N_x \cdot N_y \cdot N_z$$
$$N_c = (N_x - 1) \cdot (N_y - 1) \cdot (N_z - 1)$$

where N_x, N_y, N_z are the number of points in each dimension.



Cartesian grid

The vertex positions are given implicitly from [i,j,k]:

$$P[i,j,k].x = origin.x + i \cdot dx$$
$$P[i,j,k].y = origin.y + j \cdot dy$$
$$P[i,j,k].z = origin.z + k \cdot dz$$

The global vertex index can be computed as follows:

$I[i,j,k] = k \cdot N_y \cdot N_x + j \cdot N_x + i$, where

$$k = I \operatorname{div} (N_y \cdot N_x)$$
$$j = (I \bmod (N_y \cdot N_x)) \operatorname{div} N_x$$
$$i = (I \bmod (N_y \cdot N_x)) \bmod N_x$$

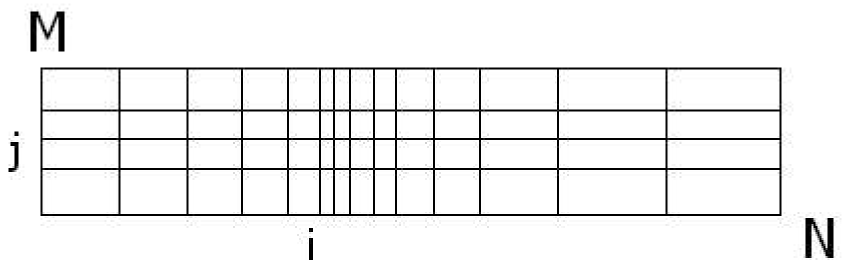
Uniform grids:

Uniform grids are similar to Cartesian grids. They consist of equal cells but with different resolution in at least one dimension $(dx \neq dy (\neq dz))$. The spacing between the grid points is constant in each dimension. Therefore, the same indexing scheme as for Cartesian grids can be used. This grid type most likely occurs in applications where the data is generated by a 3D imaging device providing different sampling rates in each dimension.

Rectilinear Grids

The topology of rectilinear grids is still regular but there is an irregular spacing between grid points. The scaling of positions along each axis is non-linear. Spacing, x_coord[L], y_coord[M] and z_coord[N] must be stored explicitly. The axes are still perpendicular to each other. The topology is still implicit.

An example for a rectilinear grid:



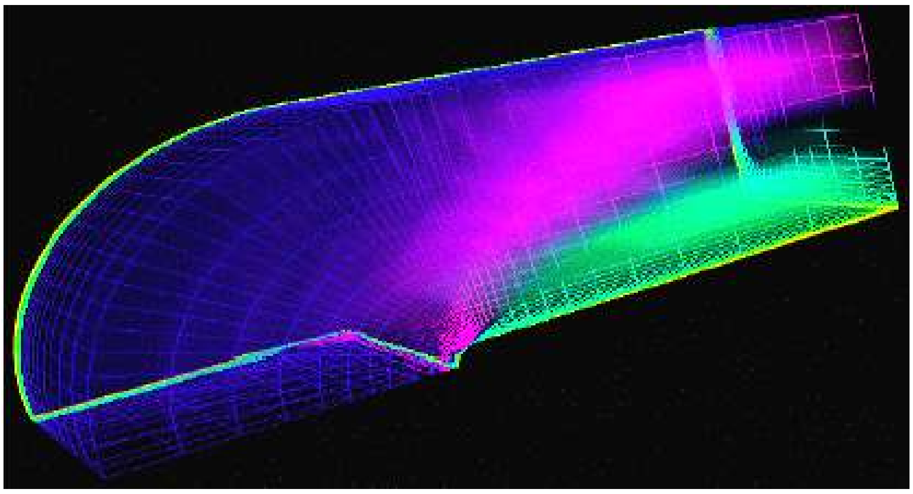
Rectilinear grid

Curvilinear grids

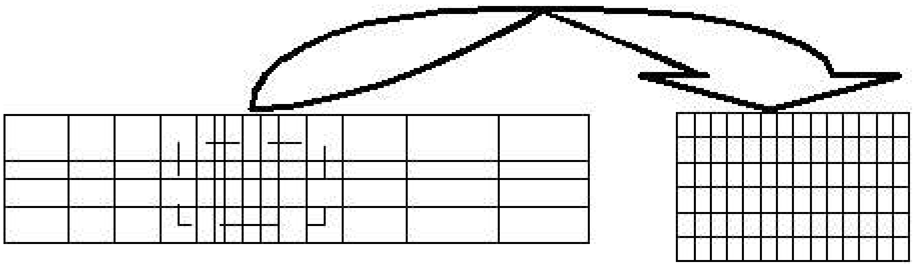
Also in curvilinear grids the topology is regular but the location of grid points is arbitrary. The topology is also implicit:

- x_coord[L,M,N]
- y_coord[L,M,N]
- z_coord[L,M,N]

The geometric structure might result in concave grids.


Curvilinear grid.

Multigrids:
Multigrids are used to avoid wasted detail if the focus is only in some areas. In these regions of interest the grid is just finer than in the rest of it.
A Multigrid:

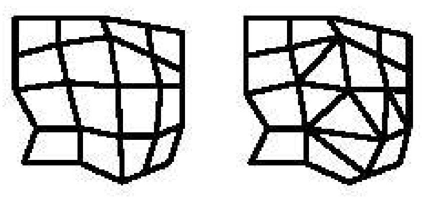

Multigrid

Structured grids can be stored in a 3D array.
Therefore arbitrary samples can be directly accessed by indexing a particular entry in the array. The topological information is implicitly coded. Adjacent elements can be directly accessed.
Cartesian, uniform, and rectilinear grids are necessarily convex.
The visibility ordering of elements with respect to any viewing direction is given implicitly.
The rigid lay out, in general, 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.

Typical implementation of structured grids (for the example of a triangle mesh):

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

6.1.2.Unstructured grids



Example for unstructured grids.

Unstructured grids are composed of arbitrarily positioned and connected elements. They can be composed of one unique element type or they can be hybrid. Triangle meshes in 2D and tetrahedral grids in 3D are most common.

Typical implementations of unstructured grids:

1. Direct form:

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

Example:

- Face 1

```
x1, y1, z1
x2, y2, z2
x3, y3, z3
```

- Face 2

```
x2, y2, z2
x3, y3, z3
x4, y4, z4
```

The data values have to be stored additionally.

This representation contains much redundancy which may cause a lack of storage.

2. Indirect form:

In the indexed face set all vertices are stored only once. The faces are stored in a face list. The elements of the face list do not contain the vertices but just indices these vertices.

Example:

- vertex list

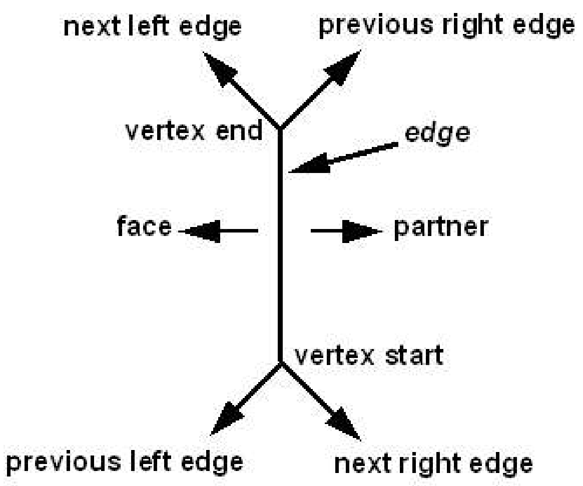
```
x1, y1, z1
x2, y2, z2
x3, y3, z3
x4, y4, z4
```

- face list

```
1, 2, 3
1, 2, 4
3, 2, 4
```

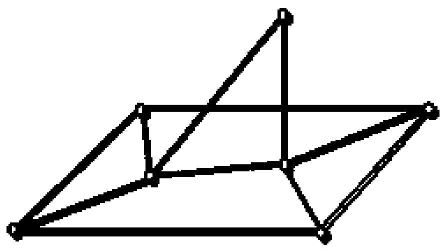
This structure is more efficient than the direct approach in terms of memory requirements, but global search is still necessary to find local information (i.e. which faces share an edge).

This problem is solved with the winged edge data structure [Baumgart 75]. This edge-based structure stores for each edge references to adjacent edges and faces.



Winged edge data structure

With this data structure it is easy to determine the faces sharing an edge/vertex. It is also possible to "walk around" the edges of a face.
For every vertex a pointer to an arbitrary edge that is incident to it has to be stored. For every face a pointer to an edge on its boundary is needed. This structure assumes that every edge has at most two faces which meet at an edge. That means only a manifold triangle mesh is possible. A 2-manifold is a 2D-surface where at every point on the surface a surrounding area can be found that looks like a disk (up to deformations). That means everything could be flattened out to a plane.



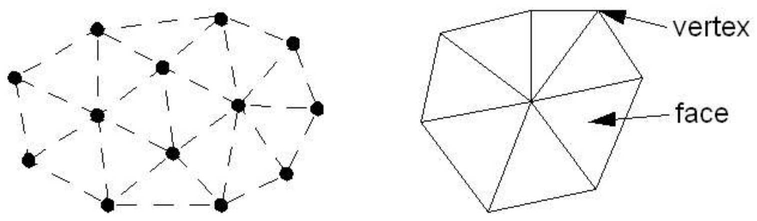
Not a 2-manifold

6.1.3.Hybrid grids

Combinations of different grid types are called hybrid grids.

6.2. Scattered data

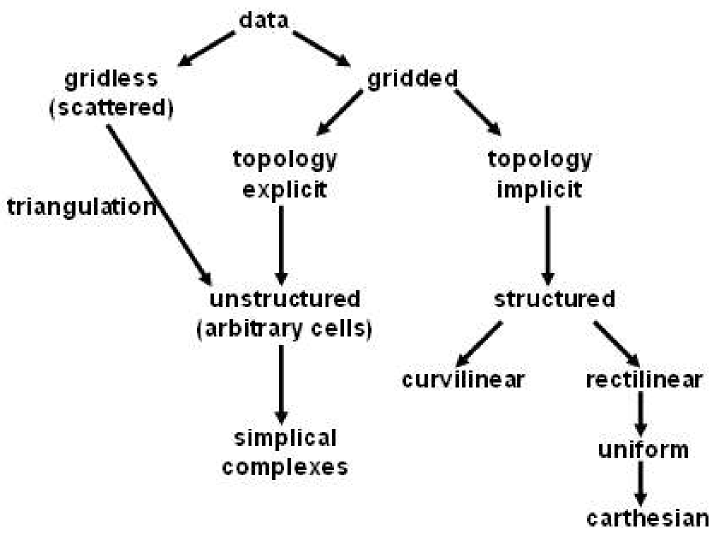
Scattered data consists of irregularly distributed positions without connectivity information. To get a connectivity it is necessary to find a "good" triangulation, i.e. a triangular/tetrahedral mesh with scattered points as vertices.



Scattered data and triangulation

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. Long and thin triangles should be avoided. This leads to the so called Delaunay triangulation which is discussed later in this course.

Summary:



Overview of all grid types.

7.Data values

Data values are characterized by the following aspects:

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

Data Type:

- Scalar data
  is given by a function $f(x_1,...,x_n): \mathbb{R}^n \rightarrow \mathbb{R}$ with n independent variables x_i
- Vector data
  represents direction and magnitude and is given by an n-tupel $(f_1,...,f_n)$ with $f_k = f_k(x_1,...,x_n), (n \geq 2) \wedge (1 \leq k \leq n)$
- Tensor data
  Tensors represent a set of directions and magnitudes. A tensor of level k is given by $t_{i1,i2,...,ik}(x_1,...,x_n)$

Range of values:
Values can be classified by being qualitative or quantitative.
Qualitative values use non-metric scales. If there is a order along the scale the values are ordinal otherwise nominal. That means for nominal values it is only possible to determine whether two values are equal or not.
Quantitative values use metric scales. Therefore also the distance between two values can be determined. The range of quantitative values can be either discrete or continuous.

Structure of the data:
The following structures can be distinguished:

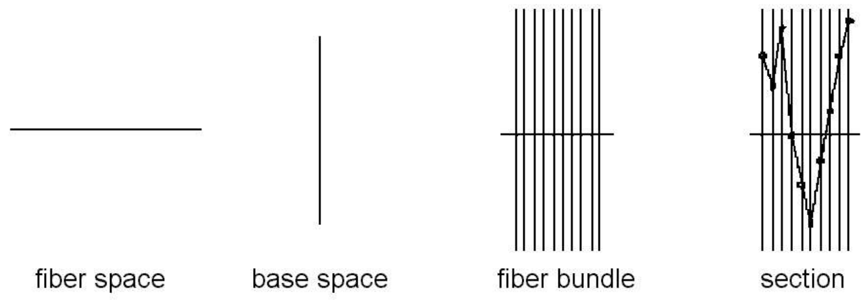
- sequential
  The values are stored in a list.
- relational
  The values are stored in a table.
- hierarchical
  The values are stored in a tree structure.
- network structure
  The values are stored in a network structure.

8.Data Classification

The aim of data classification is to provide additional information about the data to the visualization process.
One approach can be found in [Bergeron 89]. Bergeron and Grinstein consider data as m-dimensional data elements on a n-dimensional grid with the following notation:
L_m^n
Examples for m-dimensional data on
- separate points: L_m^0
- a line: L_m^1
- a surface: L_m^2
- a surface: L_m^2
- a (uniform) n-dimensional grid: L_m^n

This notation is easy to handle but important aspects of data and grid types are missing.
Brodlie defines the so called "Underlying Field" on which data is defined and from which the "Visualizing Entity" E is extracted. E is a function defined by the domain and the range of the data. The independent variables are described by their dimension and their influence, the dependent variables by their dimension and their data type.
Examples:
E_3^{V3} means a vectorial quantity with three components which is given in a 3-dimensional domain.
$E_{[2]}^{5S}$ represents five scalar quantities which are associated with a separated region of a 2-dimensional domain.
The brackets in the expression represents the influence of the data values:
- Point influence: no brackets
- Local influence: [ ]
- Global influence: { }

Another notation is the one by Butler. He defines the so called "fiber bundles". A fiber bundle is build of two spaces, the "base space", which contains the independent variables, and the "fiber space", which contains the dependent variables. Each point of the base space is associated a copy of the fiber space. Data can be specified by:
- the description of the base space
  as a multi-dimensional space with given topology
- the description of the fiber space
  e.g. as a multi-dimensional vector space
- the choice of points in the fiber space
  from each fiber space a point is chosen and by this a so called section, which corresponds to a concrete data set, is built



Fiber bundles

The notation in [Wong97] only differs between the dimension of data (number of dependent variables v) and the dimension of the domain (number of independent variables d). Data with n independent variables and m dependent variables is represented by the expression $n\,\boldsymbol{d}\,m\,\boldsymbol{v}$
The following examples compare these different notations.

Example 1:

Data: unordered set of points with scalar values

Bergeron and Grinstein	L_1^0
Brodlie	$E_{\{0\}}^S$
Butler	$base=set\,,\, fiber=float:[-\infty\,,\infty]$
Wong	$0\,\boldsymbol{d}\,1\,\boldsymbol{v}$

Example 2:

Data: ordered set of points with scalar values

Bergeron and Grinstein	L_1^0
Brodlie	$E_{[0]}^S$
Butler	$base=ordered\,set\,,\, fiber=float:[-\infty\,,\infty]$
Wong	$0\,\boldsymbol{d}\,1\,\boldsymbol{v}$

Example 3:

Data: scalar volume data set on a uniform grid

Bergeron and Grinstein	L_1^3
Brodlie	E_3^S
Butler	$base=3\,D-reg-grid\,,\, fiber=char:[0{,}255$
Wong	$3\,\boldsymbol{d}\,1\,\boldsymbol{v}$

Example 4:

Data: flow data on a curvilinear grid

Bergeron and Grinstein	L_3^3
Brodlie	E_3^{V3}

Butler	$base = 3D - curvilin - grid\ ,\ fiber = float^3 : [-\infty , \infty]^3$
Wong	$3\,\boldsymbol{d}\,3\,\boldsymbol{v}$

Example 5:

Data: 3D volume with 3 scalar and 2 vector data values

Bergeron and Grinstein	L_9^3
Brodlie	E_3^{3S2V3}
Butler	$base = 3D - reg - grid\ ,\ fiber = float \times float \times float \times float^3 \times float^3$
Wong	$3\,\boldsymbol{d}\,9\,\boldsymbol{v}$