

CS 247 – Scientific Visualization

Lecture 4: Data Representation, Pt. 2

Markus Hadwiger, KAUST



Reading Assignment #2 (until Feb 8)

Read (required):

- Data Visualization book, finish Chapter 2
- Data Visualization book, Chapter 3 until 3.5 (inclusive)
- Data Visualization book, Chapter 4 until 4.1 (inclusive)

- Continue familiarizing yourself with OpenGL if you do not know it !

Programming Assignments Schedule (tentative)



Assignment 0:	Lab sign-up: join discord, setup github account + get repo Basic OpenGL example	until	Feb 1
Assignment 1:	Volume slice viewer	until	Feb 15
Assignment 2:	Iso-contours (marching squares)	until	Mar 1
Assignment 3:	Iso-surface rendering (marching cubes)	until	Mar 15
Assignment 4:	Volume ray-casting, part 1	until	Apr 12
	Volume ray-casting, part 2	until	Apr 19
Assignment 5:	Flow vis, part 1 (hedgehog plots, streamlines, pathlines)	until	May 3
Assignment 6:	Flow vis, part 2 (LIC with color coding)	until	May 13

Programming Assignment #1: Slice Viewer



Basic tasks

- Download data into 3D volume texture
- Display three different axis-aligned slices using OpenGL texture mapping using the 3D volume texture

Minimum

- The slice position should be adjustable for each slice view.
- Make sure the aspect ratio of the shown slices is correct.
- If the window is resized, the slice is resized with the correct aspect ratio (no distortions)

Bonus

- Show all three axis aligned slices at once
- Show arbitrarily aligned slices with an interface to change the arbitrary slice

Programming Assignment #1 Example



The image shows a terminal window and a 3D visualization application side-by-side. The terminal window displays the following output:

```
#include <iostream>
G:\Development\git\Teaching\Work\CS247_Assignment1\x64\Debug\CS247_Assignment1.exe
GL_VERSION major=4 minor=3
Keyboard commands:
b - Toggle among background clear colors
w - Increase current slice
s - Decrease current slice
a - Toggle viewing axis
1 - Load lobster dataset
2 - Load head dataset
3 - Load hydrogen dataset
loading data ../../Datasets/skewed_head.dat
volume dimensions: x: 184, y: 256, z:170
downloading volume to 3D texture
increasing current slice: 86
increasing current slice: 87
increasing current slice: 88
increasing current slice: 89
increasing current slice: 90
```

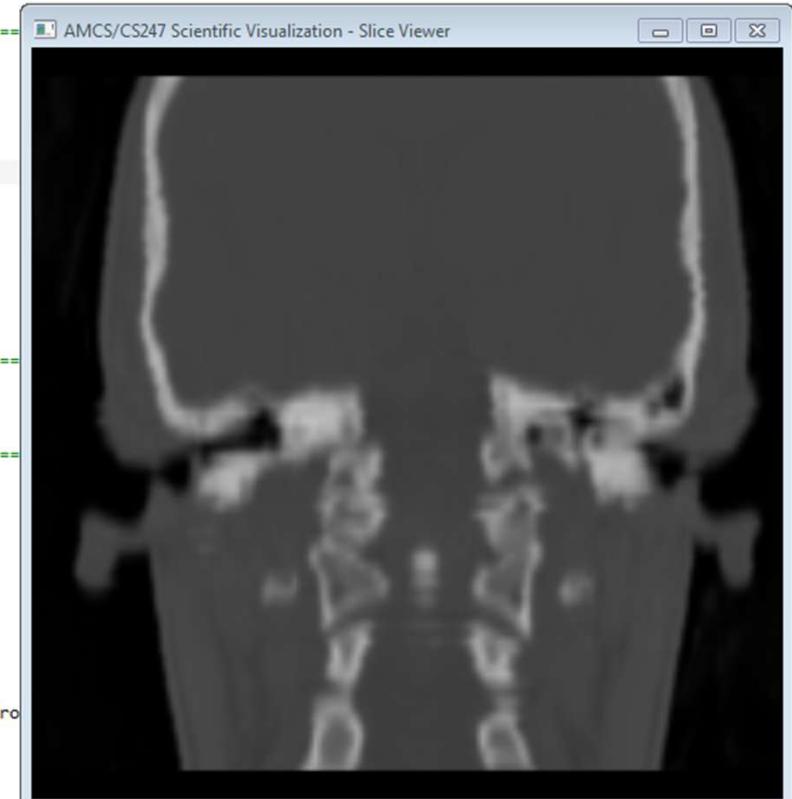
The 3D visualization application window is titled "AMCS/CS247 Scientific Visualization - Slice Viewer". It displays a grayscale 3D volume rendering of a skull, showing internal structures like the brain and eye sockets. Below the terminal window, a portion of the OpenGL error handling code is visible:

```
int printOpenGLError(char *file, int line)
{
    // Returns 1 if an OpenGL error occurred, 0 otherwise.
    GLenum glErr;
    int retCode = 0;

    glErr = glGetError();
    while (glErr != GL_NO_ERROR)
    {
        printf("glError in file %s @ line %d: %s\n", file, line, gluErrorString(glErr));
        retCode = 1;
        glErr = glGetError();
    }
    return retCode;
}

#define printOpenGLError() printOpenGLError(__FILE__ __LINE__)
```

Programming Assignment #1 Example



The screenshot shows two windows. The left window is a terminal or debugger output window titled "G:\Development\git\Teaching\Work\CS247_Assignment1\x64\Debug\CS247_Assignment1.exe". It displays the following text:

```
b - Toggle among background clear colors
w - Increase current slice
s - Decrease current slice
a - Toggle viewing axis
1 - Load lobster dataset
2 - Load head dataset
3 - Load hydrogen dataset
loading data ../../Datasets/skewed_head.dat
volume dimensions: x: 184, y: 256, z:170
downloading volume to 3D texture
increasing current slice: 86
increasing current slice: 87
increasing current slice: 88
increasing current slice: 89
increasing current slice: 90
toggling viewing axis to: 0
increasing current slice: 93
increasing current slice: 94
increasing current slice: 95
toggling viewing axis to: 1
decreasing current slice: 127
decreasing current slice: 126
decreasing current slice: 125
decreasing current slice: 124
```

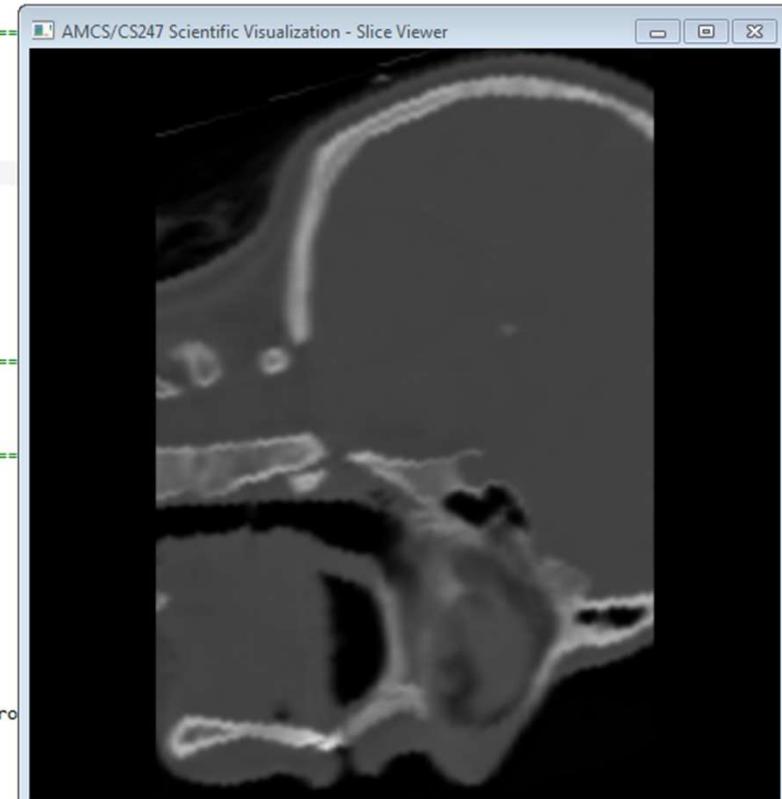
The right window is titled "AMCS/CS247 Scientific Visualization - Slice Viewer". It displays a grayscale 3D volume rendering of a head dataset, showing internal structures like the brain and skull. Below the terminal window, a portion of C++ code is visible:

```
int printOpenGLError(char *file, int line)
{
    // Returns 1 if an OpenGL error occurred, 0 otherwise.
    GLenum glErr;
    int retCode = 0;

    glErr = glGetError();
    while (glErr != GL_NO_ERROR)
    {
        printf("glError in file %s @ line %d: %s\n", file, line, gluErrorName(glErr));
        retCode = 1;
        glErr = glGetError();
    }
    return retCode;
}
```



Programming Assignment #1 Example



The screenshot shows a development environment with two windows. On the left is a terminal window titled "G:\Development\git\Teaching\Work\CS247_Assignment1\x64\Debug\CS247_Assignment1.exe". It displays the OpenGL version (GL_VERSION major=4 minor=3), keyboard commands, and a log of operations including loading datasets and increasing current slices. On the right is a window titled "AMCS/CS247 Scientific Visualization - Slice Viewer" showing a grayscale 3D volume rendering of a skull.

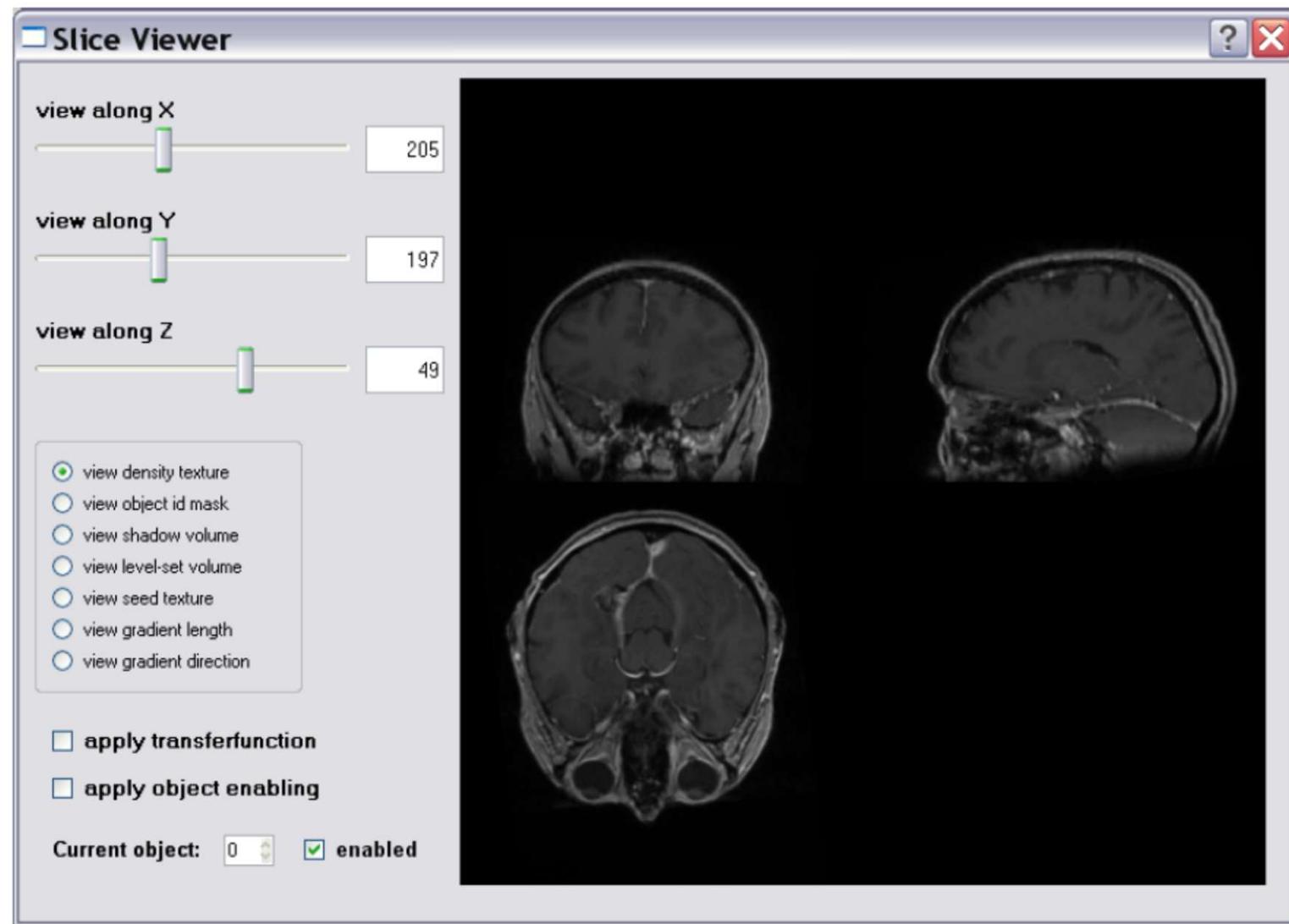
```
#include <iostream>
GL_VERSION major=4 minor=3
Keyboard commands:
b - Toggle among background clear colors
w - Increase current slice
s - Decrease current slice
a - Toggle viewing axis
1 - Load lobster dataset
2 - Load head dataset
3 - Load hydrogen dataset
loading data ../../Datasets/skewed_head.dat
volume dimensions: x: 184, y: 256, z:170
downloading volume to 3D texture
increasing current slice: 86
increasing current slice: 87
increasing current slice: 88
increasing current slice: 89
increasing current slice: 90
toggling viewing axis to: 0
increasing current slice: 93
increasing current slice: 94
increasing current slice: 95

int printOpenGLError(char *file, int line)
{
    // Returns 1 if an OpenGL error occurred, 0 otherwise.
    GLenum glErr;
    int retCode = 0;

    glErr = glGetError();
    while (glErr != GL_NO_ERROR)
    {
        printf("glError in file %s @ line %d: %s\n", file, line, gluErrorString(glErr));
        retCode = 1;
        glErr = glGetError();
    }
    return retCode;
}
```

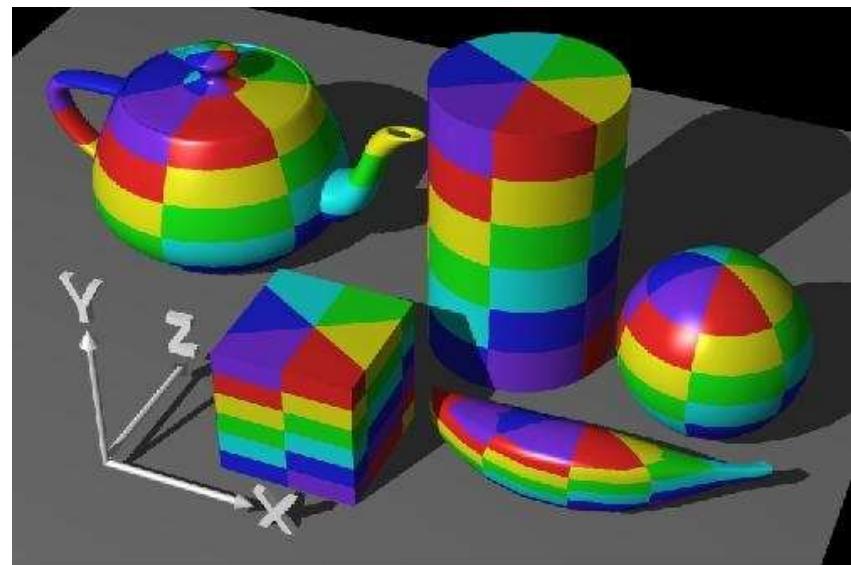
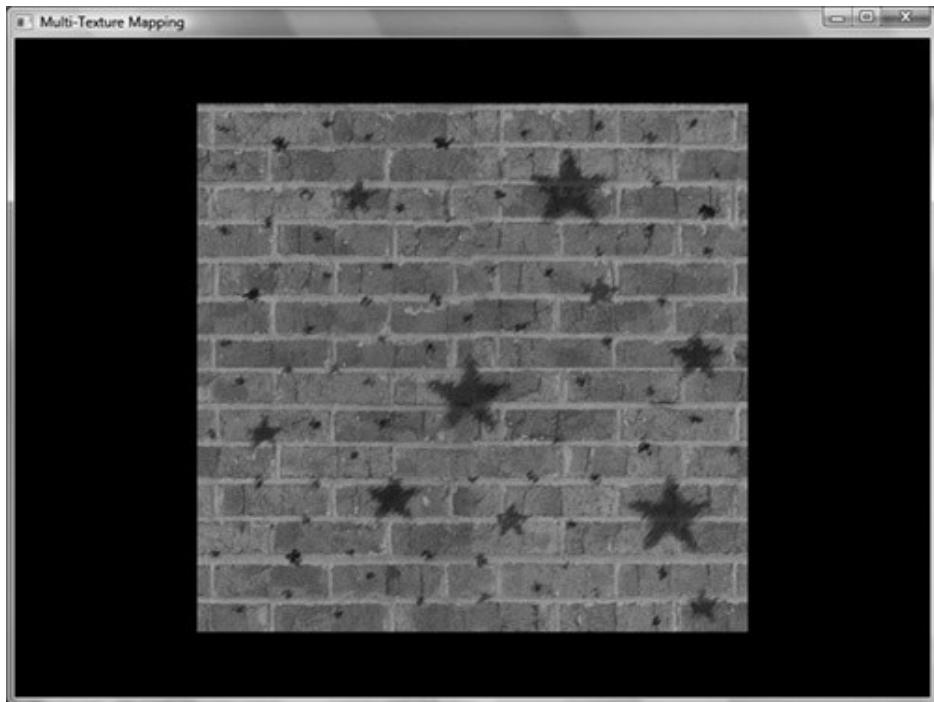


Programming Assignment #1 Example



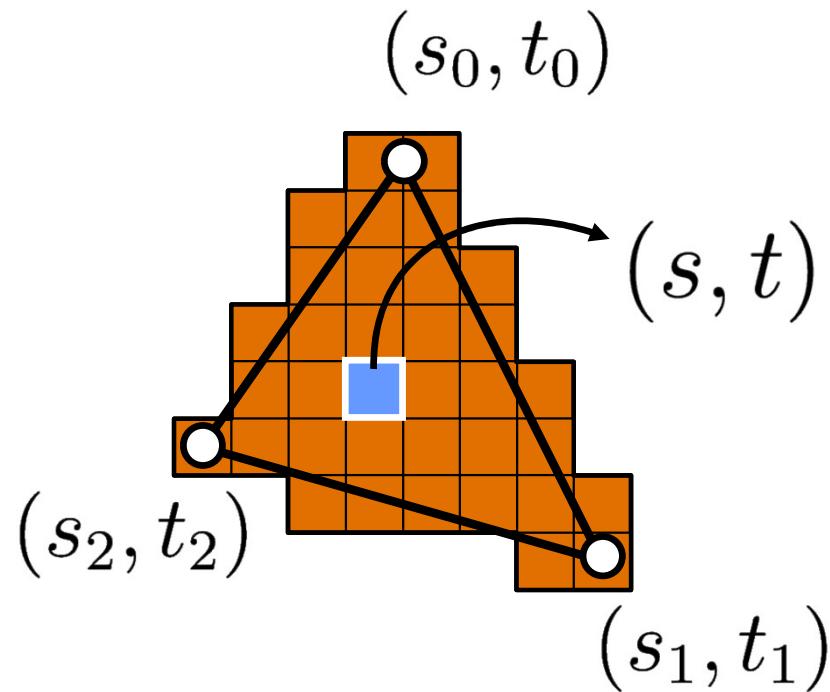


Texture Mapping



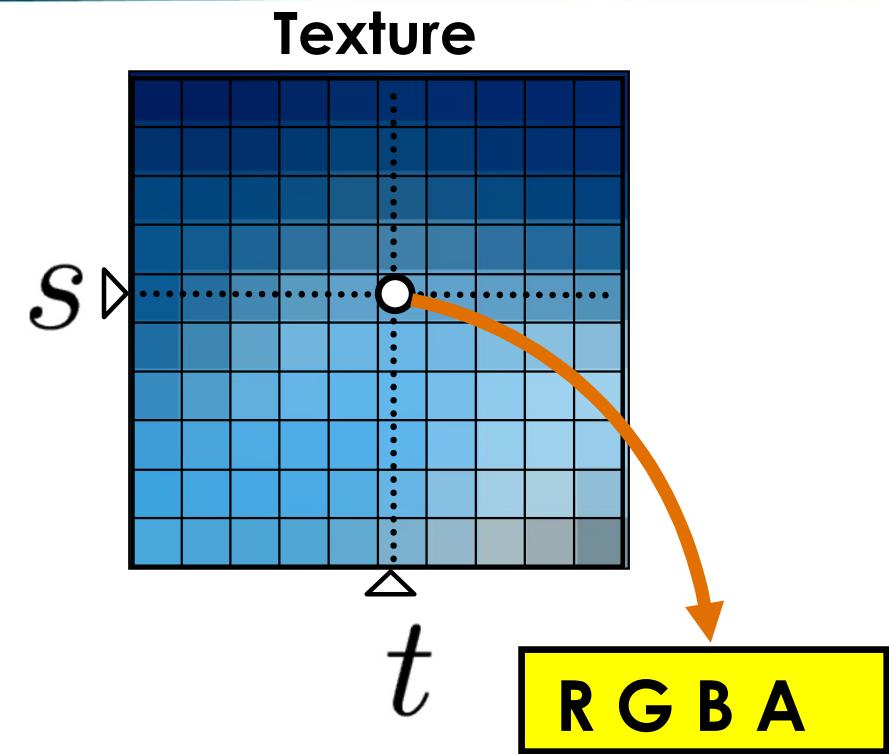


2D Texture Mapping



For each fragment:
interpolate the
texture coordinates
(barycentric)
Or:

Use arbitrary, computed coordinates

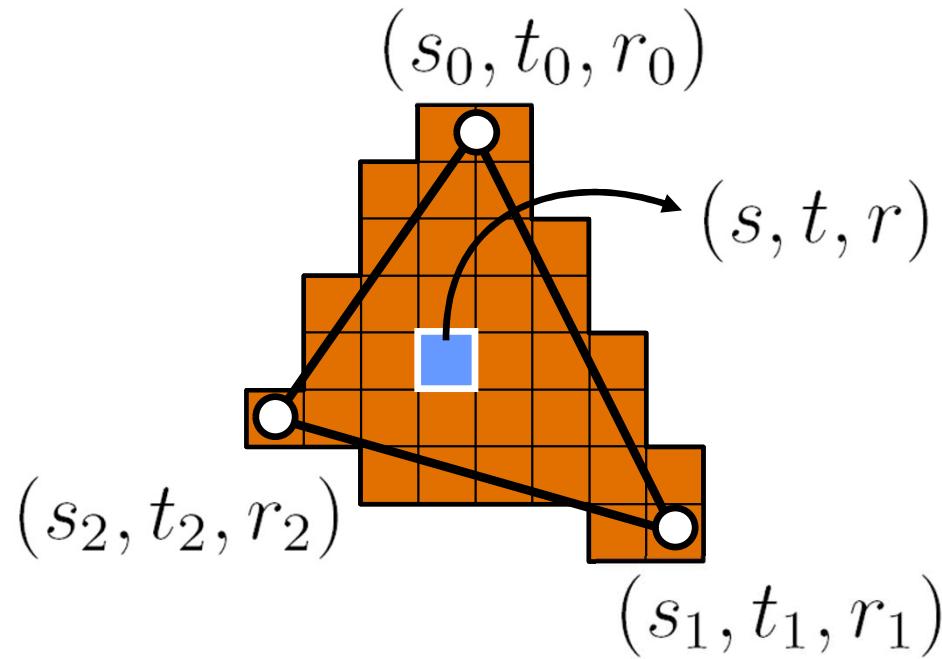


Texture-Lookup:
interpolate the
texture data
(bi-linear)
Or:

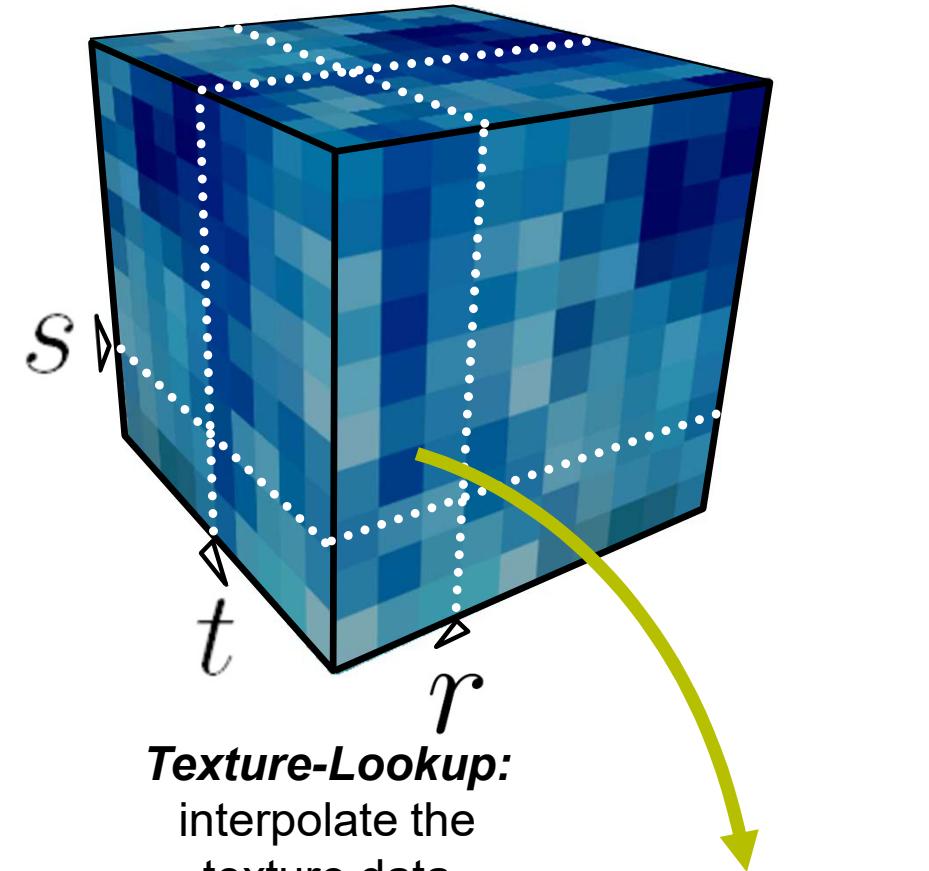
Nearest-neighbor for “array lookup”



3D Texture Mapping



For each fragment:
interpolate the
texture coordinates
(barycentric)
Or:
Use arbitrary, computed coordinates



Texture-Lookup:
interpolate the
texture data
(tri-linear)
Or:
Nearest-neighbor for “array lookup”

Data == Functions



Mathematical Functions

Associates every element of a set (e.g., X) with *exactly one* element of another set (e.g., Y)

Maps from *domain* (X) to *codomain* (Y)

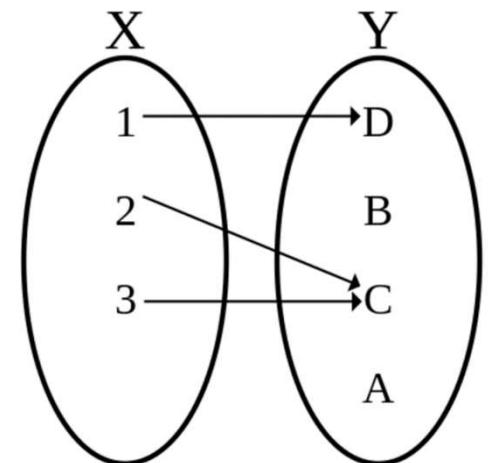
$$f: X \rightarrow Y$$

$$x \mapsto f(x)$$

Also important: *range/image*; *preimage*; continuity, differentiability, dimensionality, ...

Graph of a function (mathematical definition):

$$G(f) := \{(x, f(x)) \mid x \in X\} \subset X \times Y$$





Mathematical Functions

Associates every element of a set (e.g., X) with *exactly one* element of another set (e.g., Y)

Maps from *domain* (X) to *codomain* (Y)

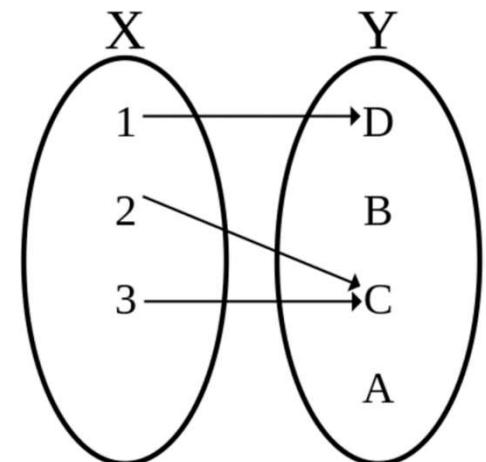
$$f: \mathbb{R}^n \rightarrow \mathbb{R}^m$$

$$x \mapsto f(x)$$

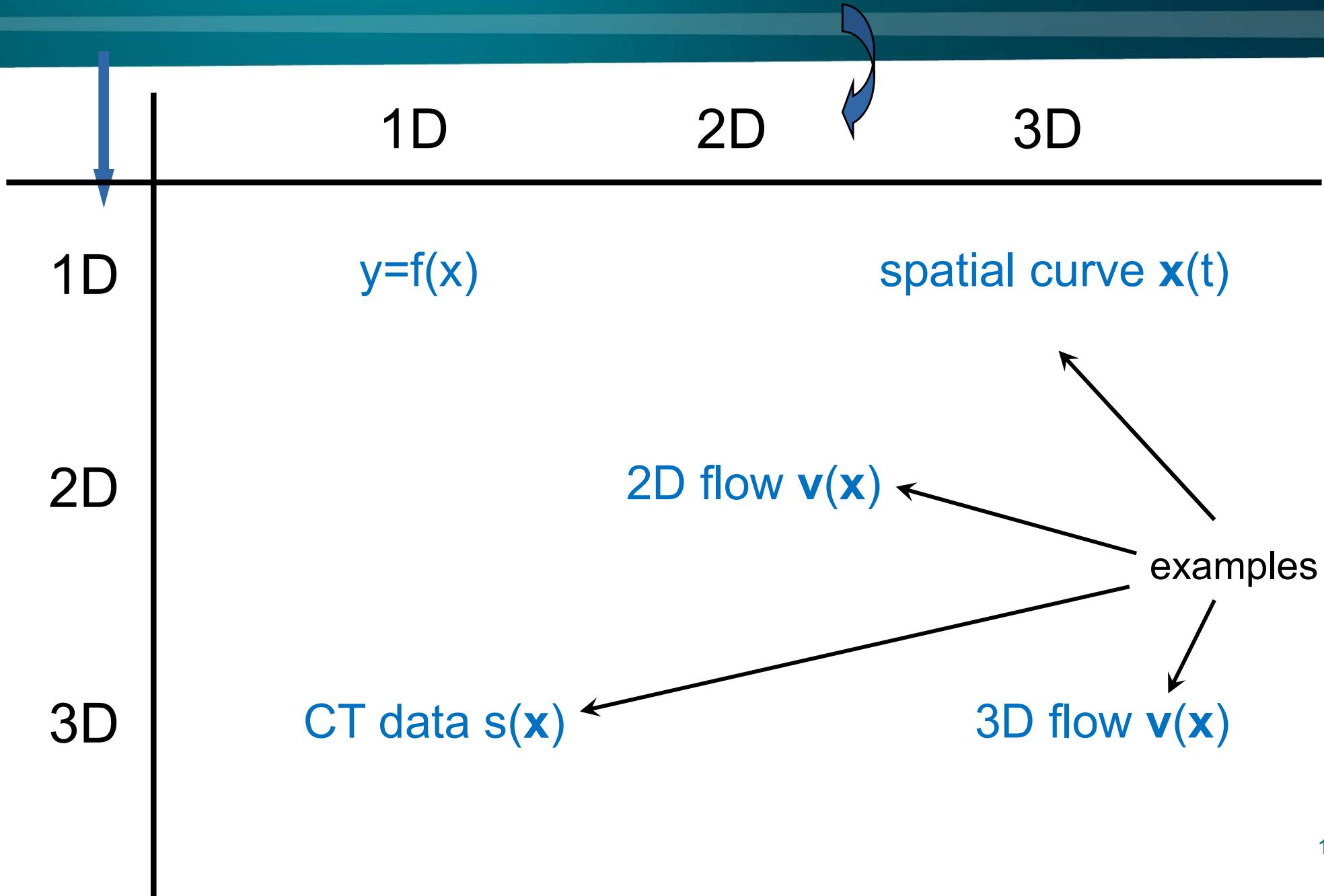
Also important: *range/image*; *preimage*;
continuity, differentiability, dimensionality, ...

Graph of a function (mathematical definition):

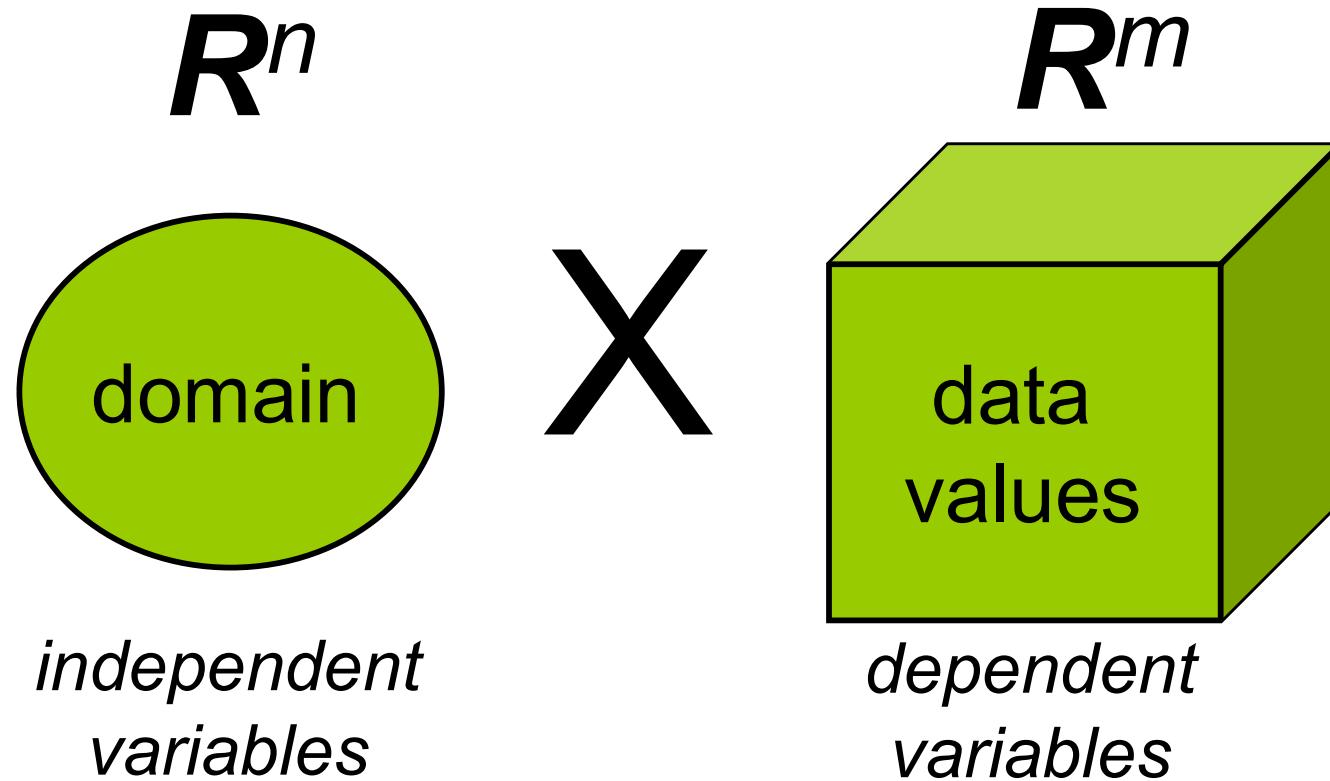
$$G(f) := \{(x, f(x)) \mid x \in \mathbb{R}^n\} \subset \mathbb{R}^n \times \mathbb{R}^m \simeq \mathbb{R}^{n+m}$$



Data Space (Domain) vs. Data Type (Codomain)



Data Representation



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



Example: Scalar Fields

2D scalar field

$$f: \mathbb{R}^2 \rightarrow \mathbb{R}$$

$$x \mapsto f(x)$$

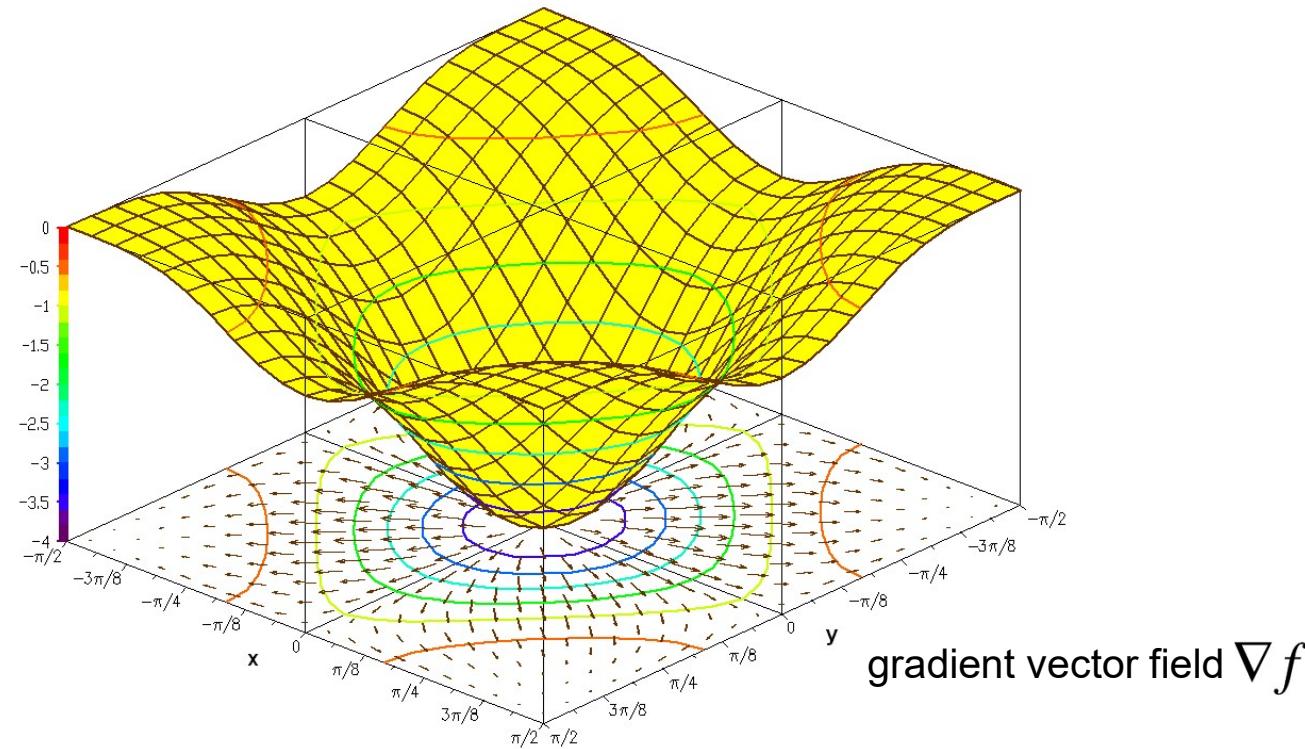
Graph: $G(f) := \{(x, f(x)) | x \in \mathbb{R}^2\} \subset \mathbb{R}^2 \times \mathbb{R} \simeq \mathbb{R}^3$

pre-image

$$S(c) := f^{-1}(c)$$

iso-contour

$$(\nabla f \neq 0)$$





Example: Scalar Fields

3D scalar field

$$f: \mathbb{R}^3 \rightarrow \mathbb{R}$$

$$x \mapsto f(x)$$

Graph: $G(f) := \{(x, f(x)) | x \in \mathbb{R}^3\} \subset \mathbb{R}^3 \times \mathbb{R} \simeq \mathbb{R}^4$

pre-image

$$S(c) := f^{-1}(c)$$

iso-surface

$$(\nabla f \neq 0)$$

?

Visualization Examples



data	description	visualization example
$N^1 \rightarrow R^1$	value series	bar chart, pie chart, etc.
$R^1 \rightarrow R^1$	scalar function over R	(line) graph
$R^2 \rightarrow R^1$	scalar function over R^2	2D-height map in 3D, contour lines in 2D, false color map
$R^2 \rightarrow R^2$	2D vector field	hedgehog plot, LIC, streamlets, etc.
$R^3 \rightarrow R^1$	scalar function over R^3 (3D densities)	iso-surfaces in 3D, volume rendering
$R^3 \rightarrow R^3$	3D vector field	streamlines/pathlines in 3D



Visualization Examples

data

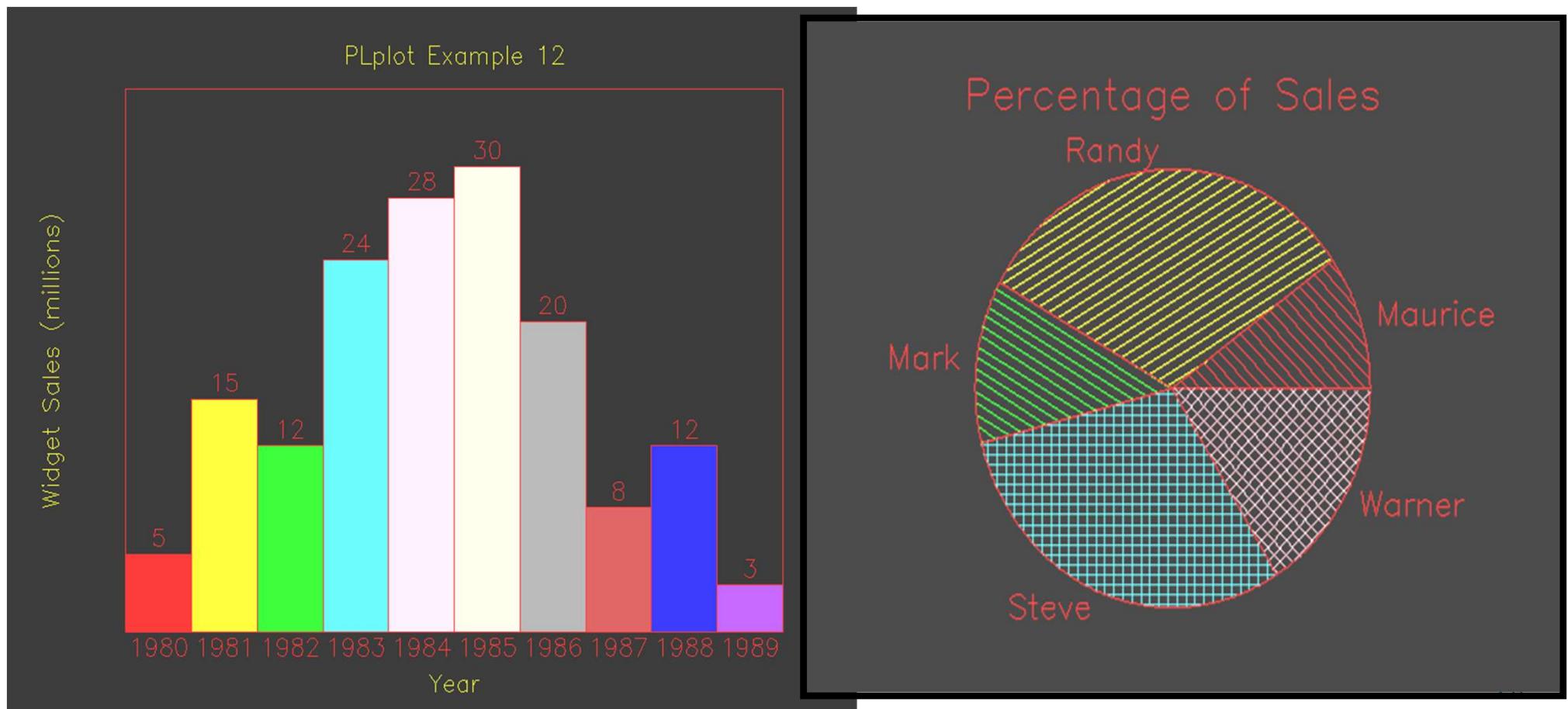
description

visualization example

$N^1 \rightarrow R^1$

value series

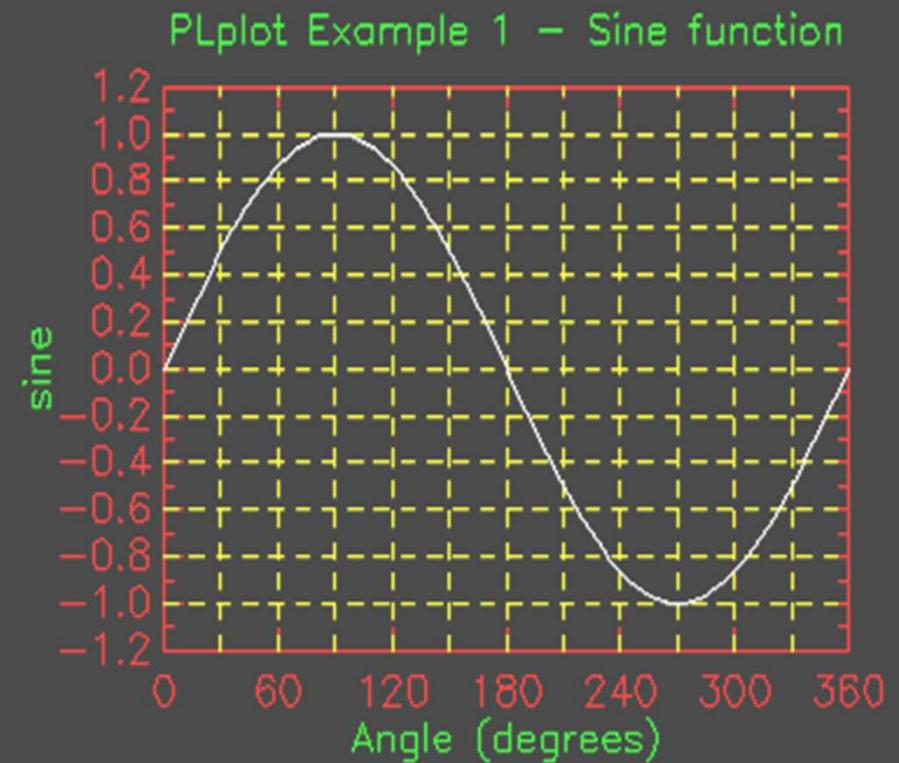
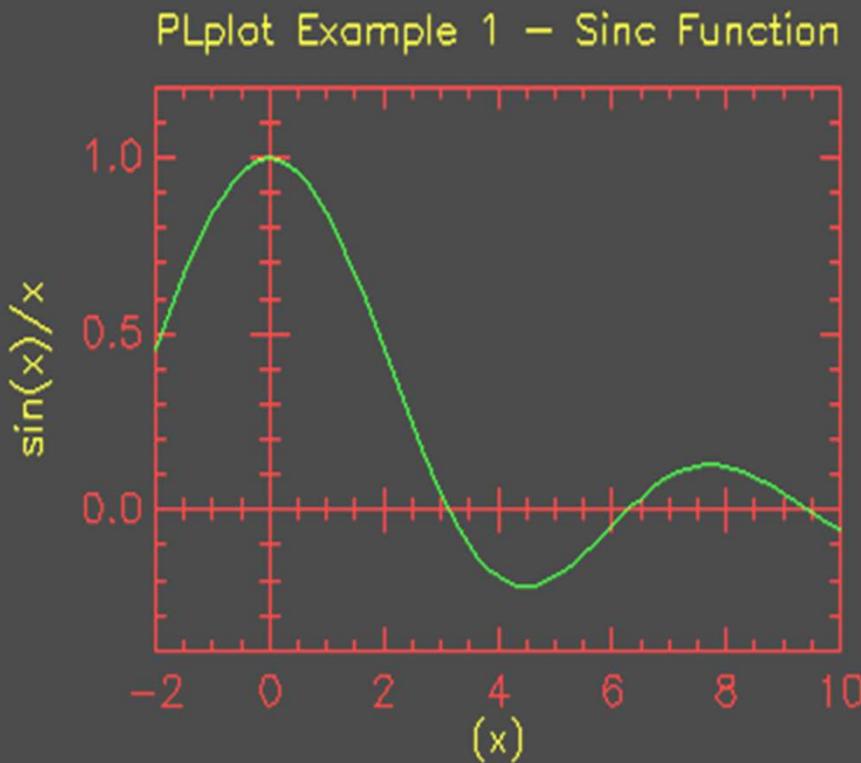
bar chart, pie chart, etc.





Visualization Examples

data	description	visualization example
$R^1 \rightarrow R^1$	function over R	(line) graph





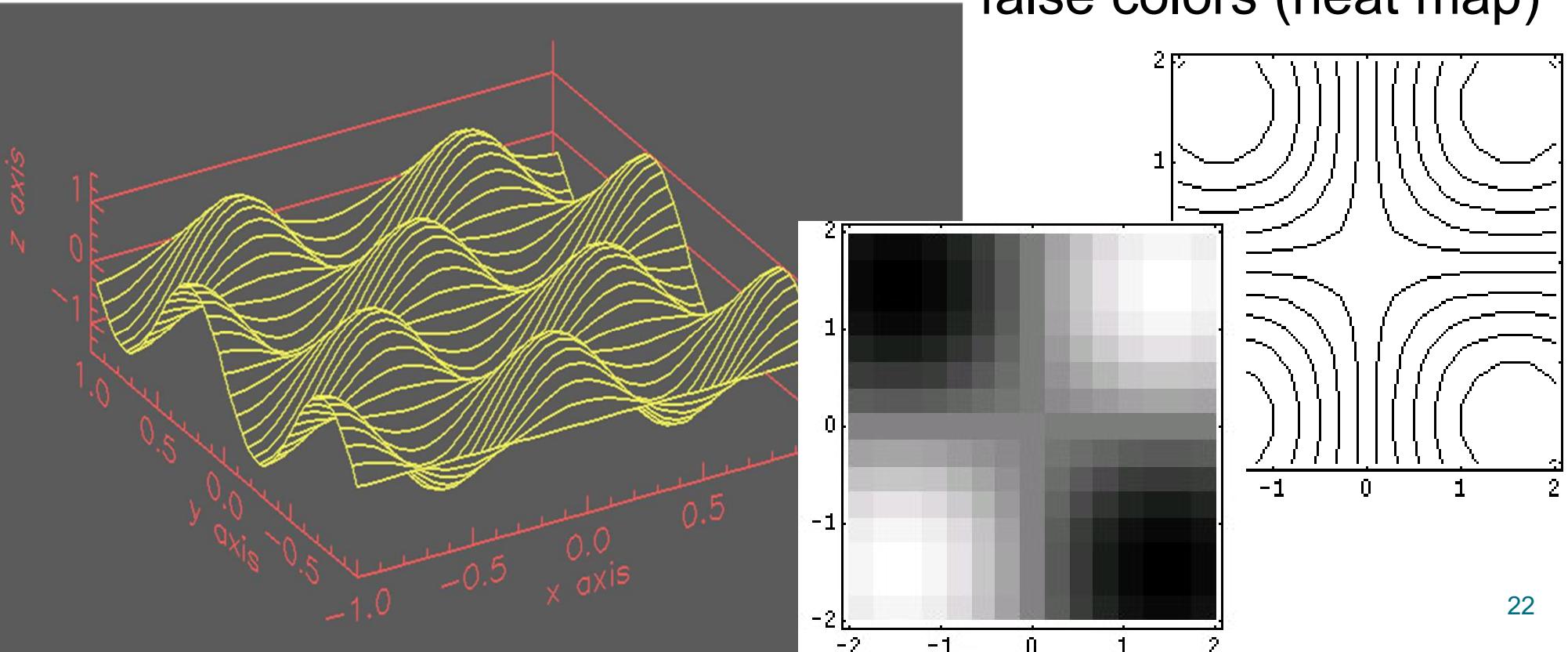
Visualization Examples

data	description	visualization example
------	-------------	-----------------------

$\mathbb{R}^2 \rightarrow \mathbb{R}^1$

function over \mathbb{R}^2

2D-height map in 3D,
contour lines in 2D,
false colors (heat map)





Visualization Examples

data

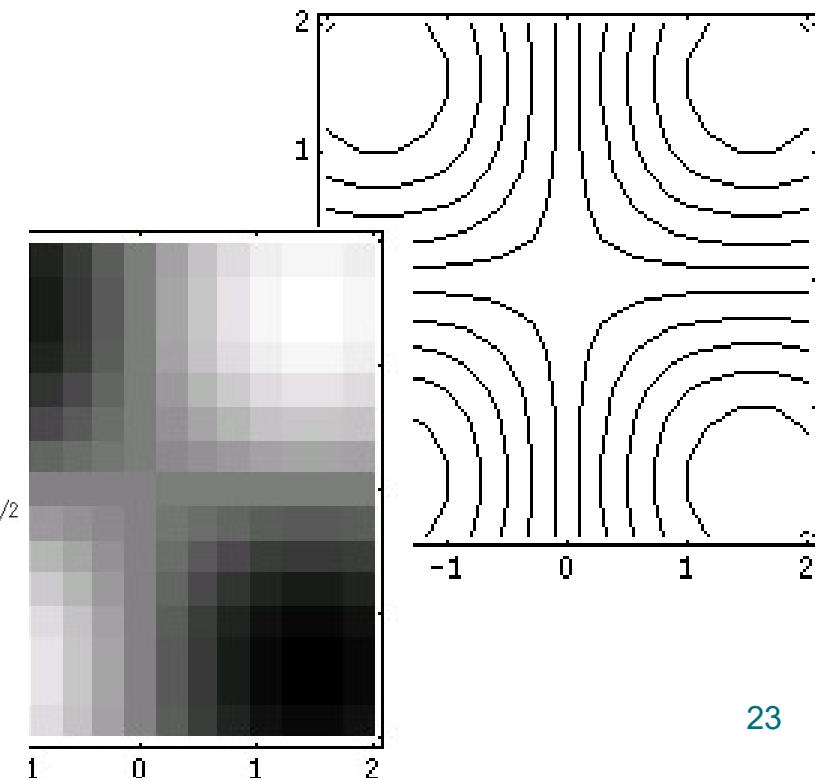
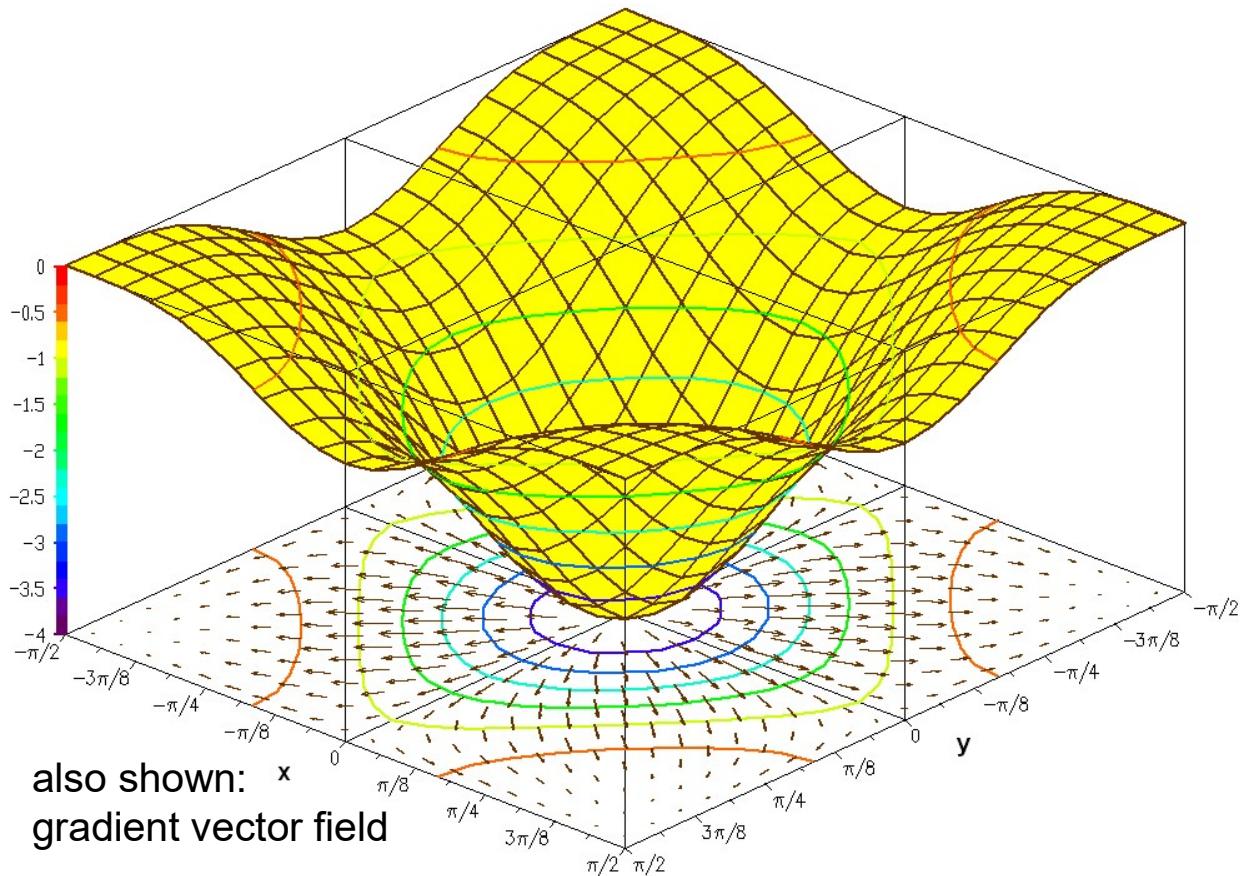
description

visualization example

$\mathbb{R}^2 \rightarrow \mathbb{R}^1$

function over \mathbb{R}^2

2D-height map in 3D,
contour lines in 2D,
false colors (heat map)





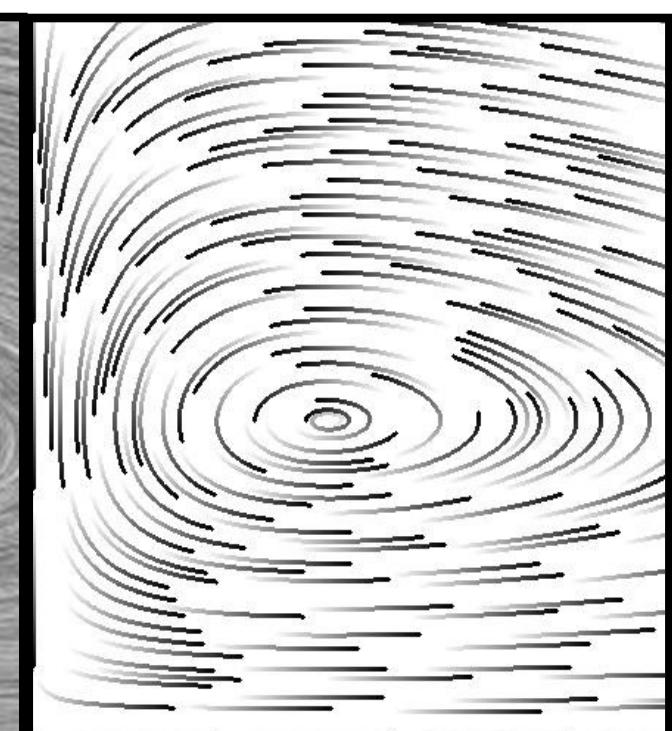
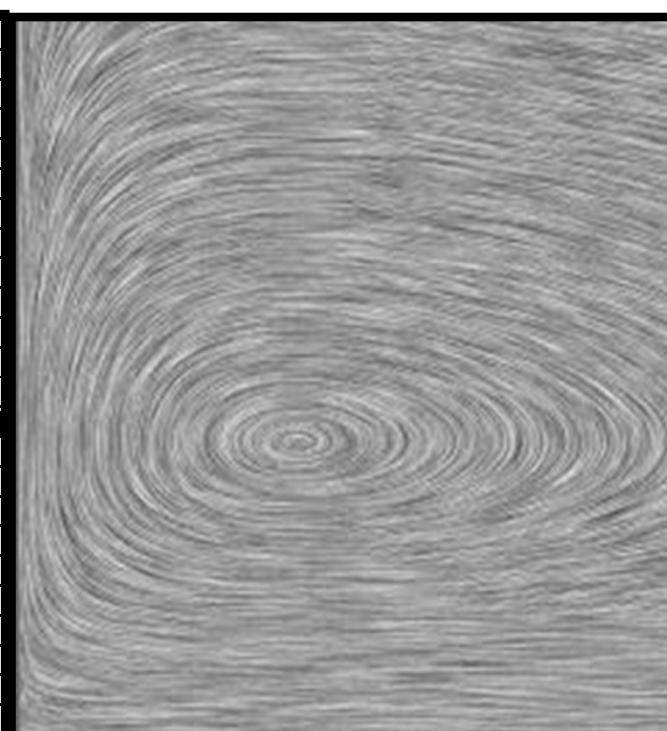
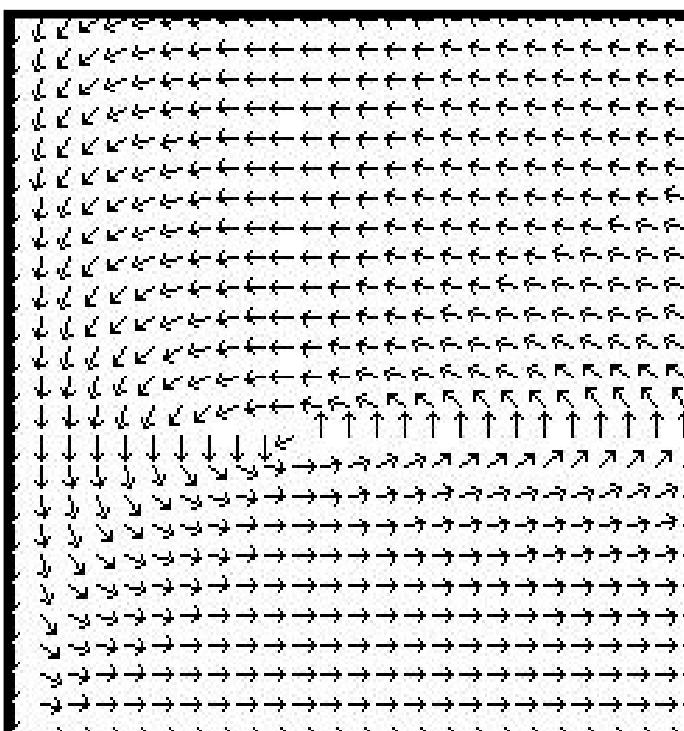
Visualization Examples

data	description	visualization example
------	-------------	-----------------------

$\mathbb{R}^2 \rightarrow \mathbb{R}^2$

2D-vector field

hedgehog plot, LIC,
streamlets, etc





Visualization Examples

data

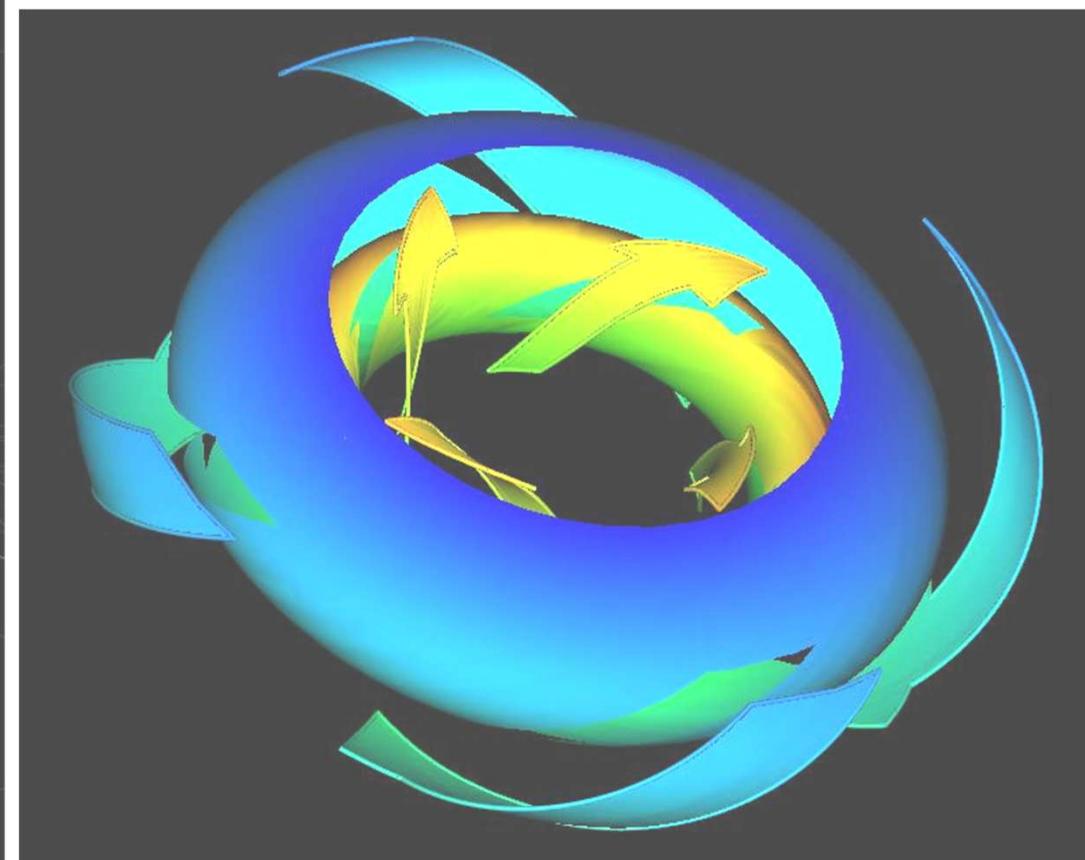
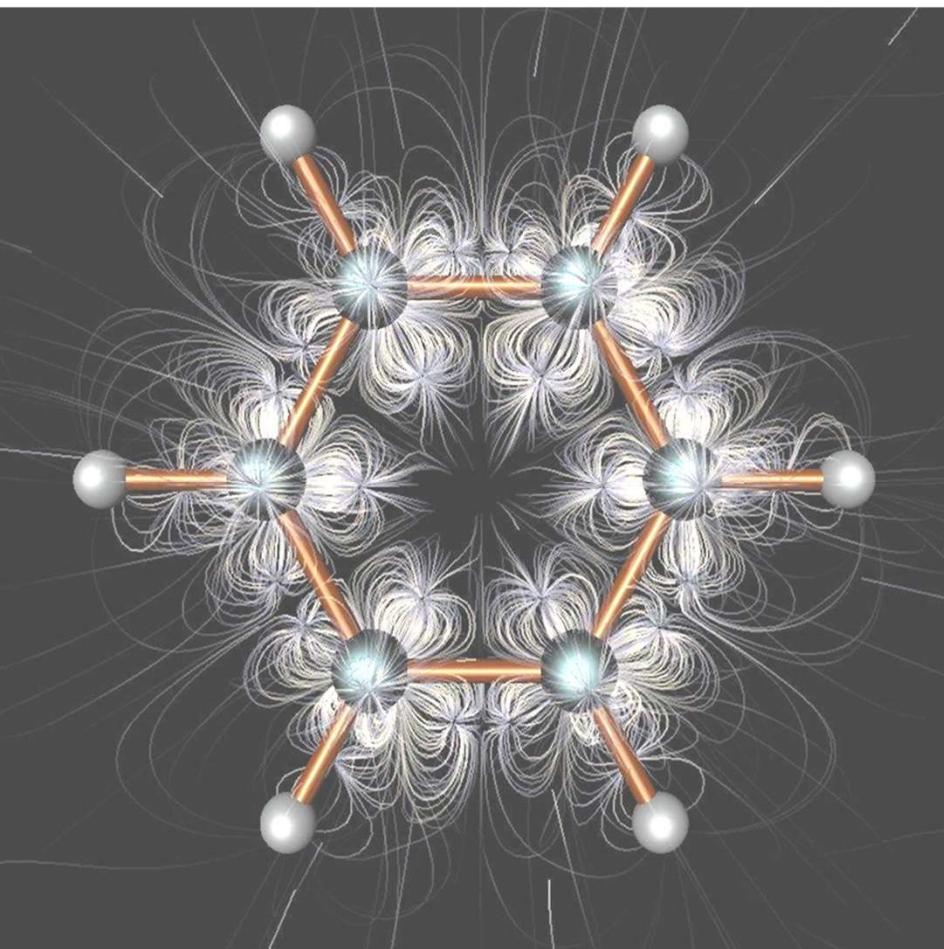
$\mathbb{R}^3 \rightarrow \mathbb{R}^3$

description

3D-flow

visualization example

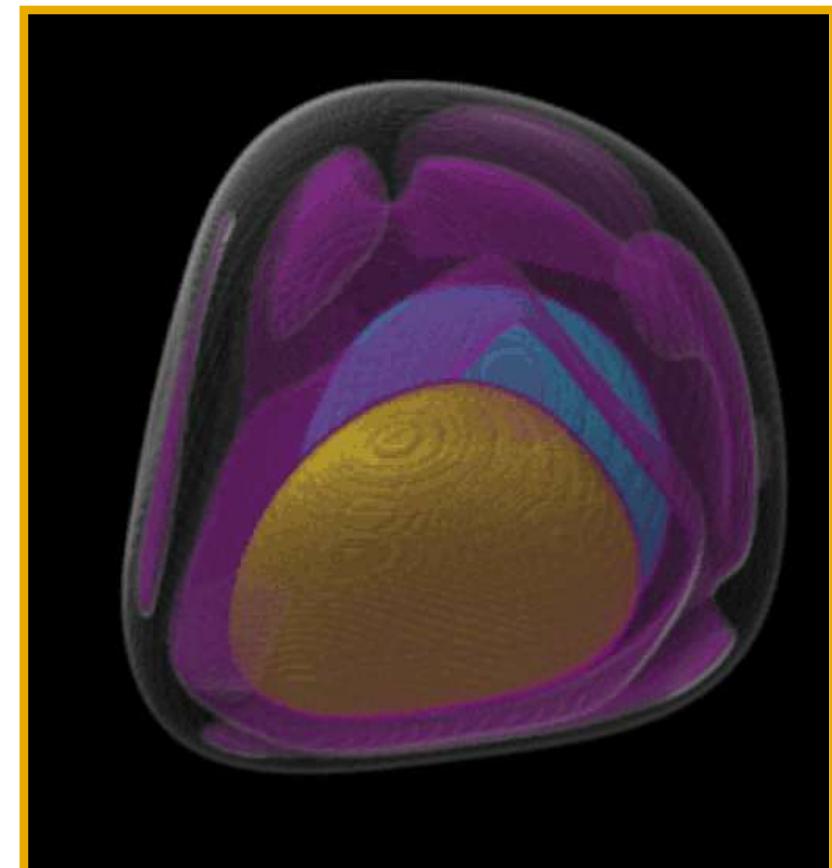
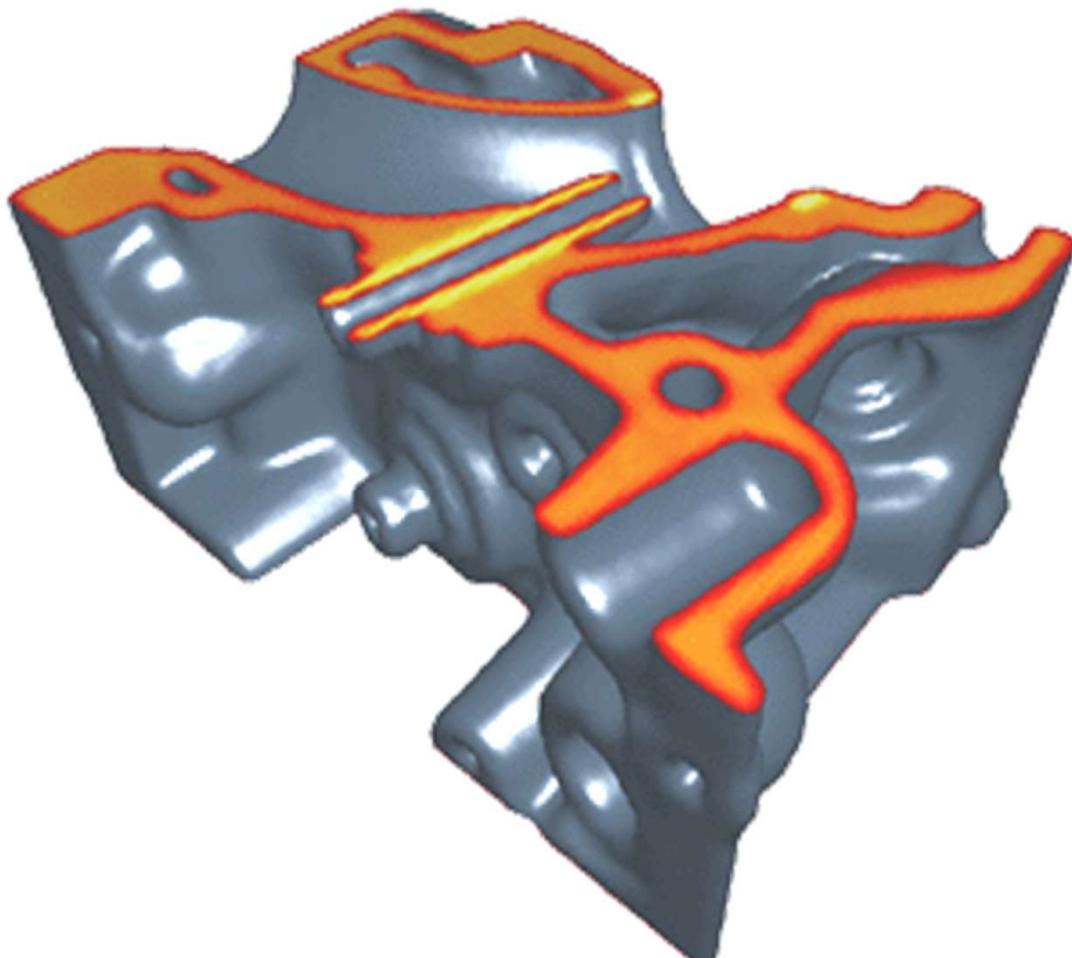
streamlines,
streamsurfaces





Visualization Examples

data	description	visualization example
$R^3 \rightarrow R^1$	3D-densities	iso-surfaces in 3D, volume rendering



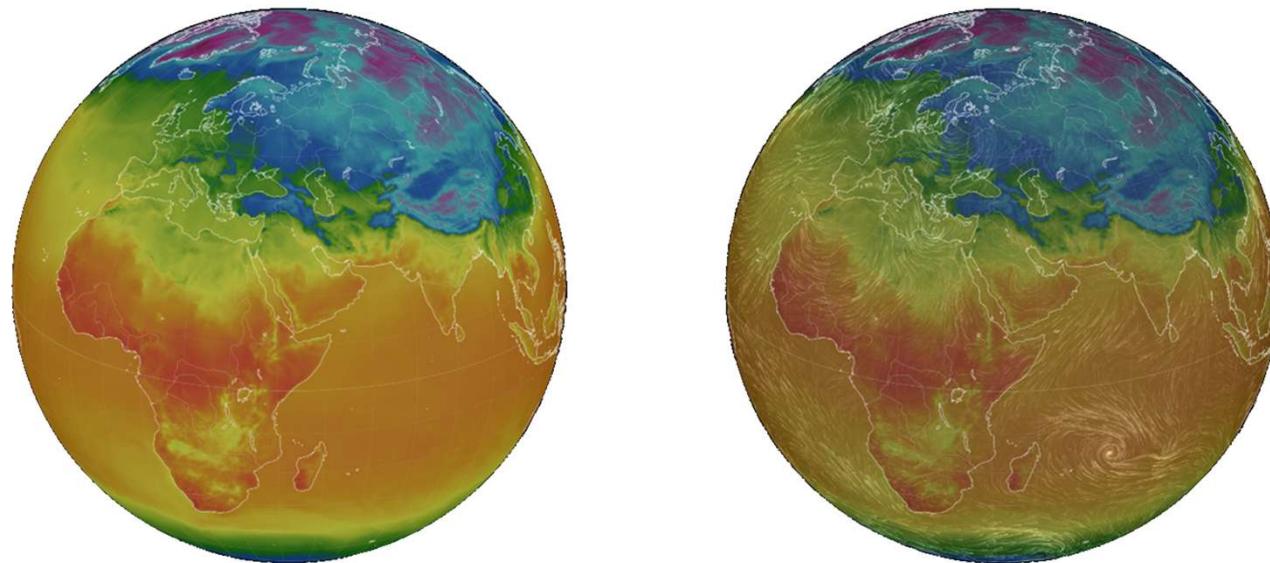


Domain is Not Always Euclidean

Manifolds



- Scalar, vector, tensor fields on manifolds

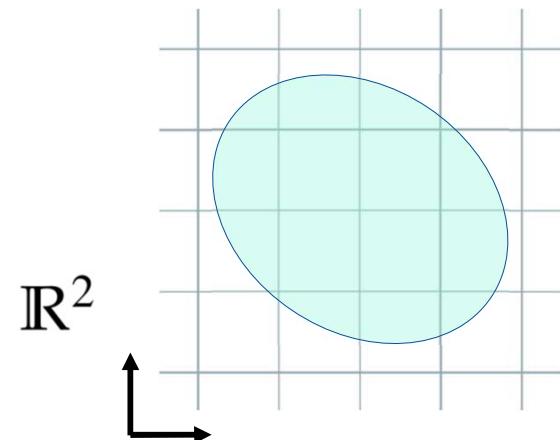
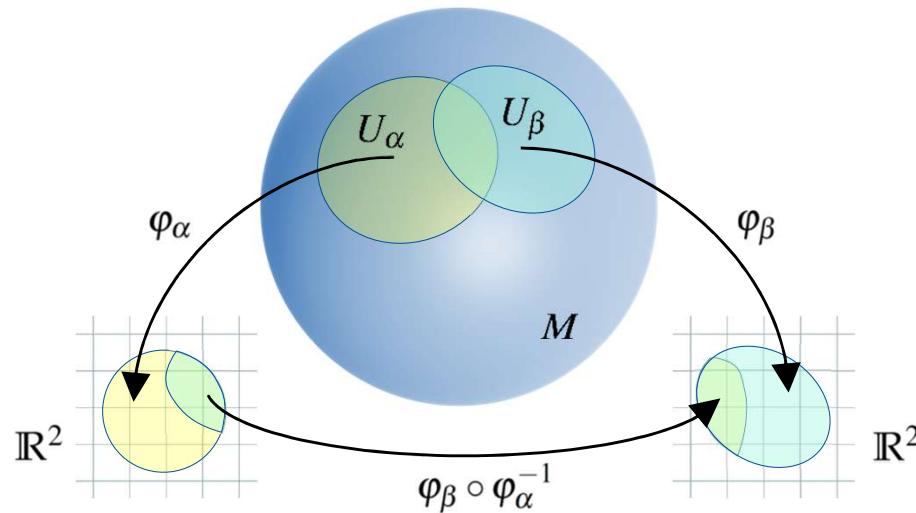




Topological Manifolds

Every point of an n -manifold is homeomorphic
(topologically equivalent) to a region of \mathbb{R}^n

Think about being able to assign coordinates to a region:
coordinate chart; (collection of charts: atlas)



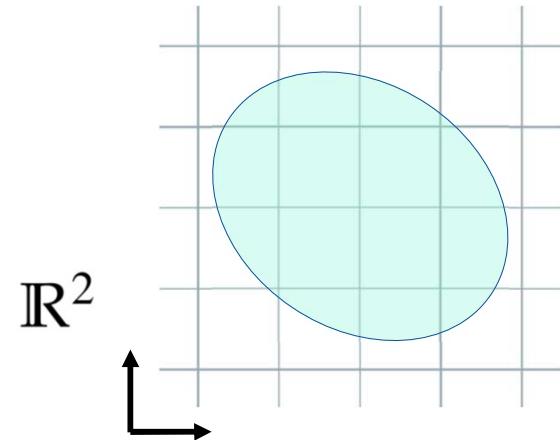
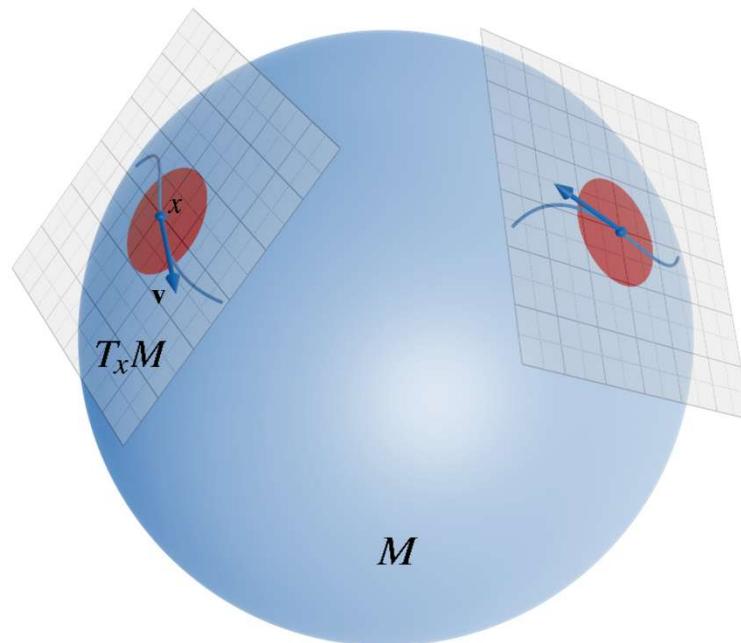


Smooth Manifolds

Well-defined tangent space at every point

- Dimensionality of each tangent space is the same as that of manifold

Enables calculus on manifolds (and vector fields, tensor fields, ...)



Sampled Functions and Data Structures

Data Representation

- Discrete (sampled) 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 different dimensions

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

Domain

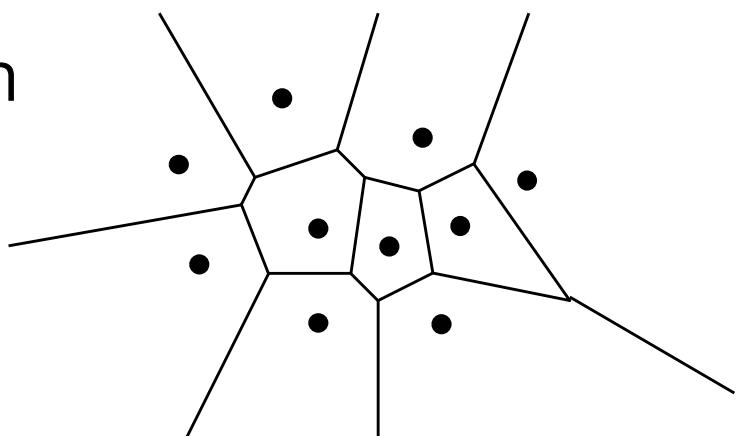
- The (geometric) shape of the domain is determined by the positions of sample points
- Domain is characterized by
 - Dimensionality: 0D, 1D, 2D, 3D, 4D, ...
 - Influence: How does a data point influence its neighborhood?
 - Structure: Are data points connected? How? (Topology)

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

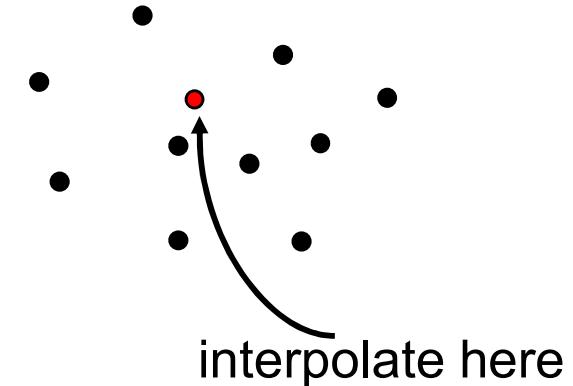
Domain

- 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
 - Nearest-neighbor interpolation



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



Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama