CS 247 – Scientific Visualization
Lecture 7: Data Representation, Pt. 5; Scalar Field Visualization, Pt. 1

Markus Hadwiger, KAUST

Reading Assignment #4 (until Feb 22)

Read (required):

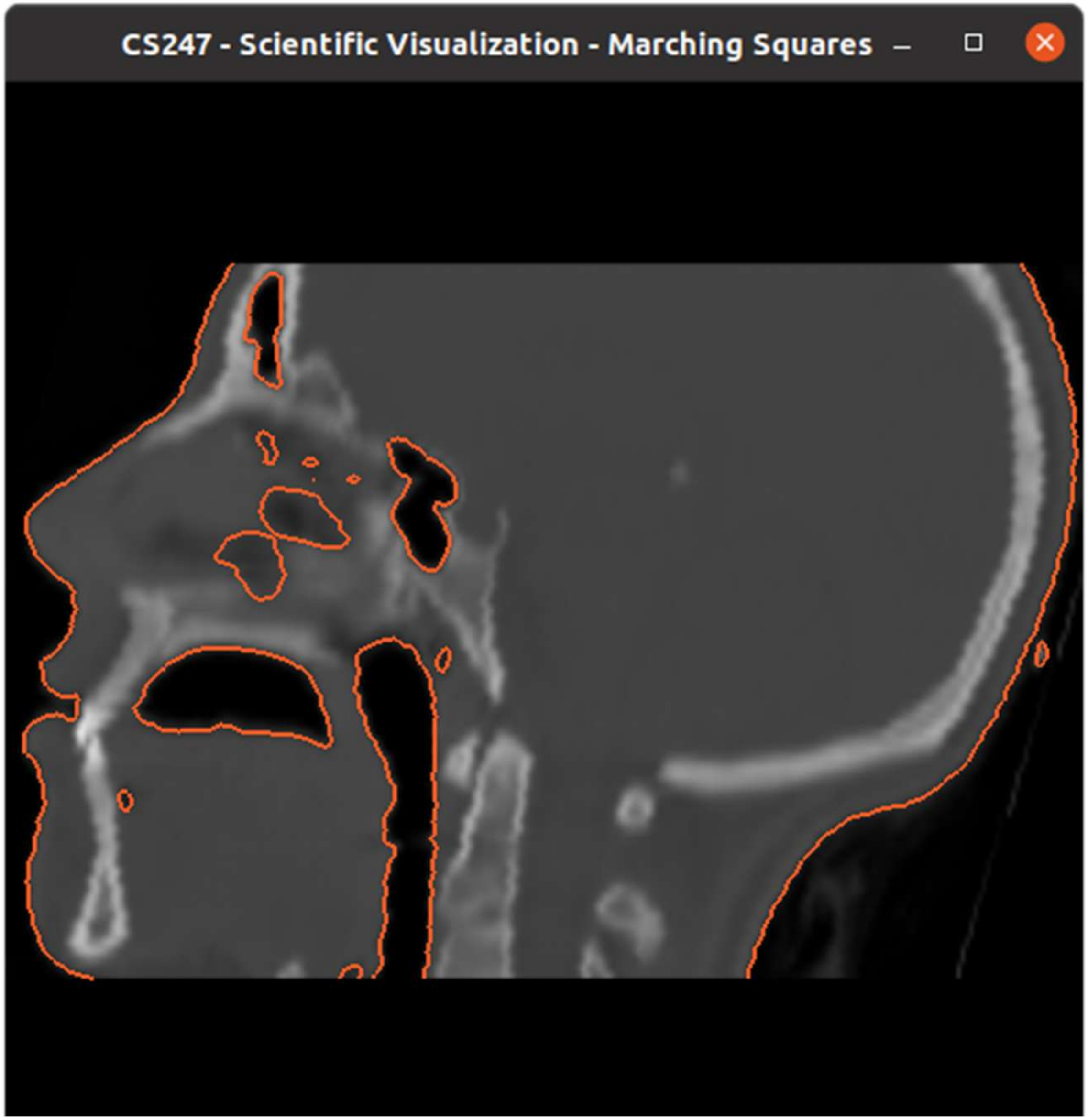
- Real-Time Volume Graphics book, Chapter 5 until 5.4 inclusive (*Terminology, Types of Light Sources, Gradient-Based Illumination, Local Illumination Models*)
- Paper:
  Marching Cubes: A high resolution 3D surface construction algorithm,
  Bill Lorensen and Harvey Cline, ACM SIGGRAPH 1987
  [> 22,100 citations and counting...]

  `https://dl.acm.org/doi/10.1145/37402.37422`

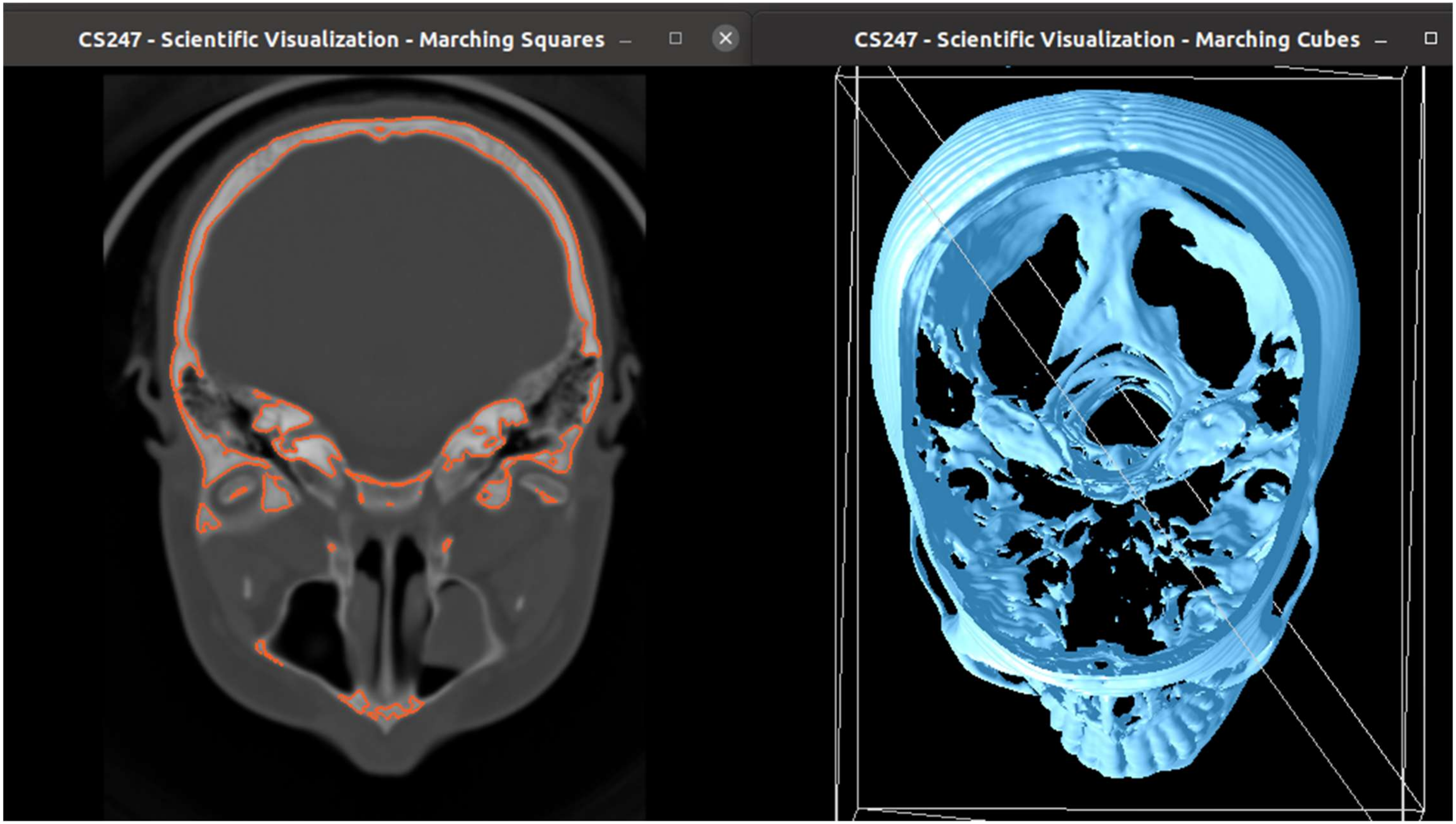
Programming Assignments Schedule (tentative)

Assignment 0:	Lab sign-up: join discord, setup github account + get repo	until	**Feb 1**
	Basic OpenGL example		
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**

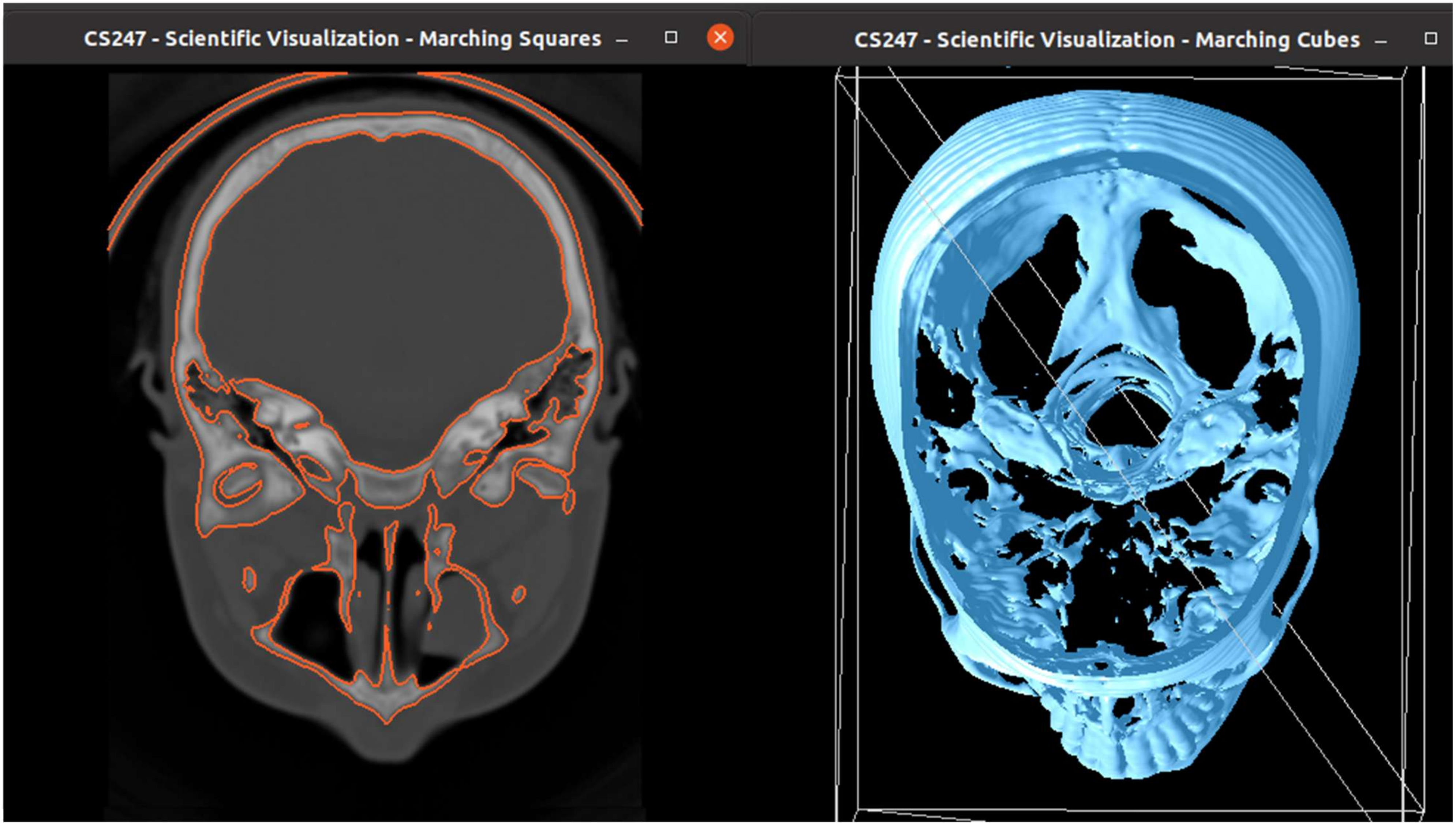
Programming Assignment 2



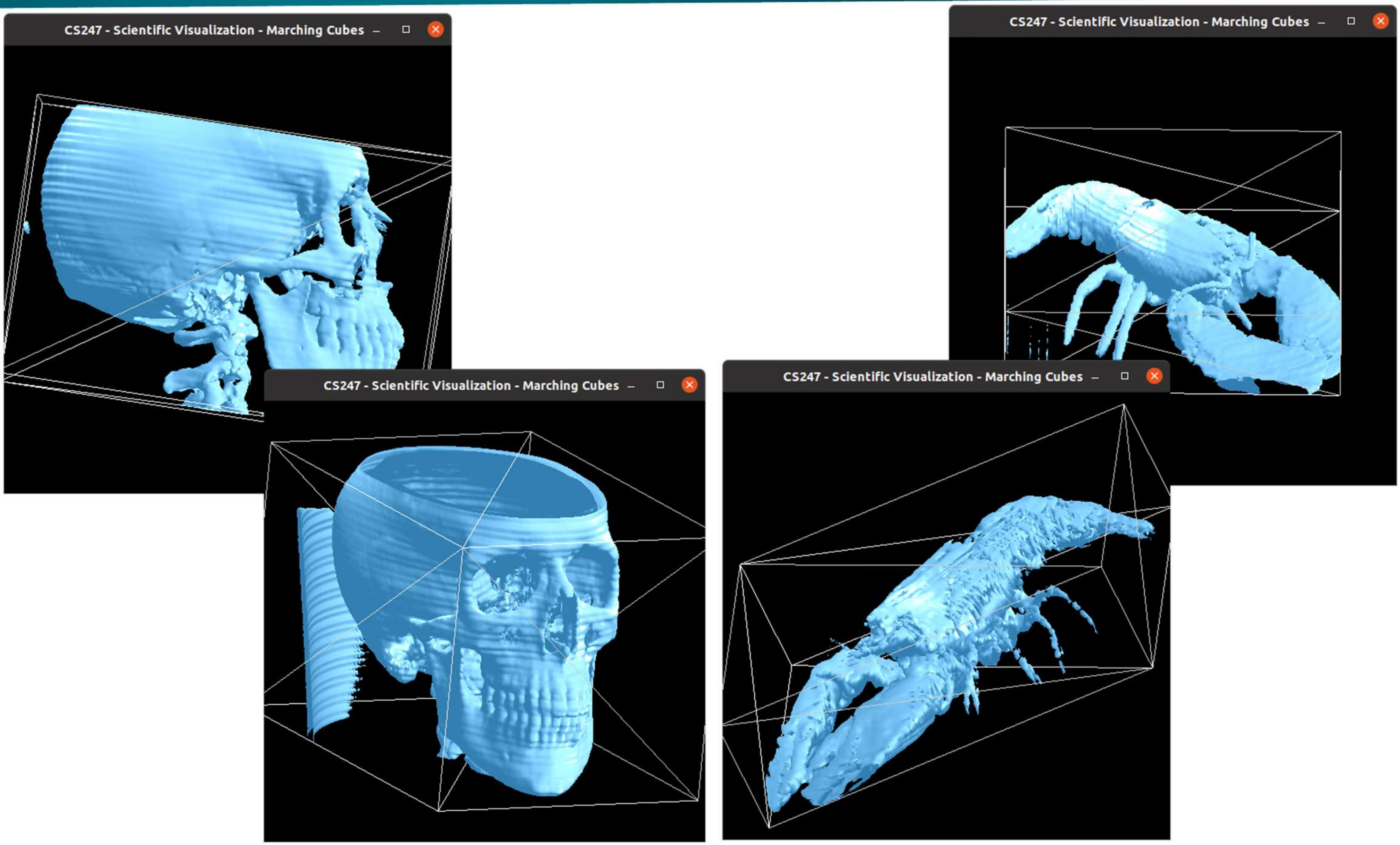
Programming Assignment 2 + 3



Programming Assignment 2 + 3

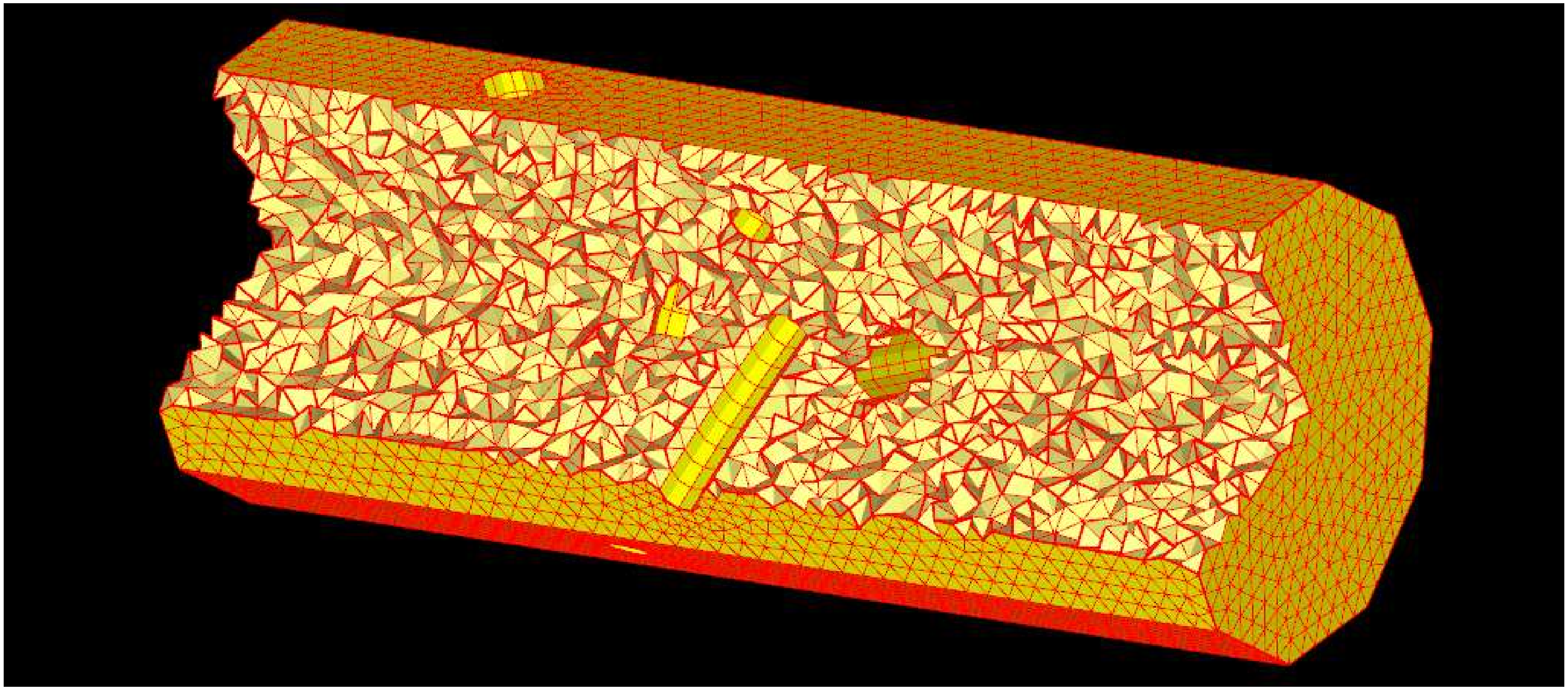


Programming Assignment 3



Common Unstructured Grid Types (1)

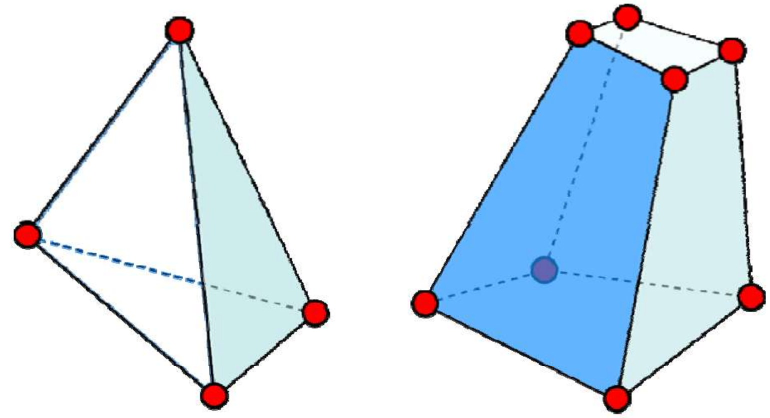
- Simplest: purely tetrahedral



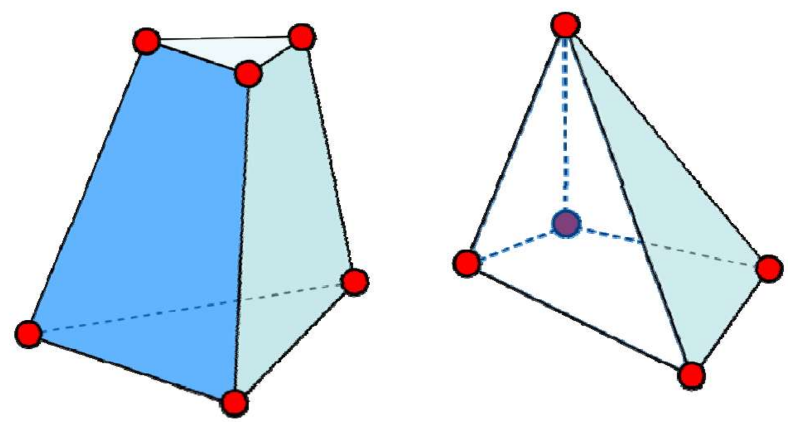
Unstructured grids

3D unstructured grids:

- cells are tetrahedra or hexahedra



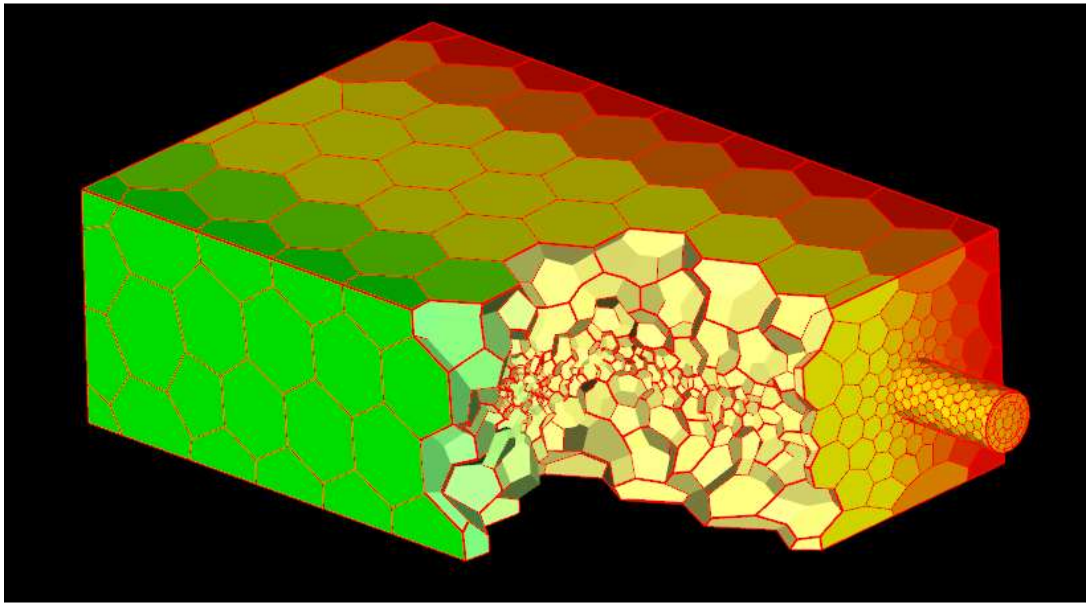
- mixed grids (“zoo meshes”) require additional types:
  wedge (3-sided prism), and pyramid (4-sided)

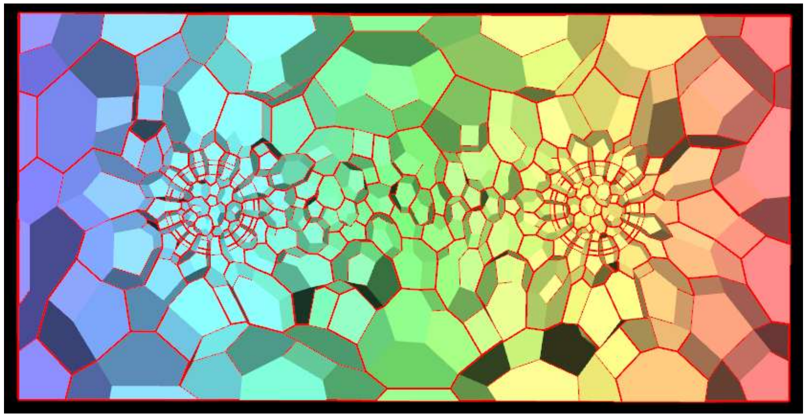


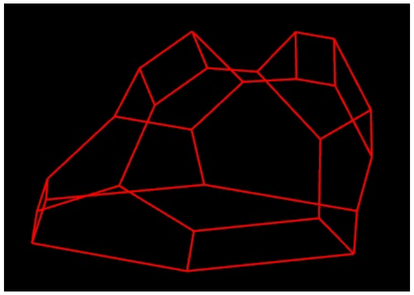
Common Unstructured Grid Types (3)

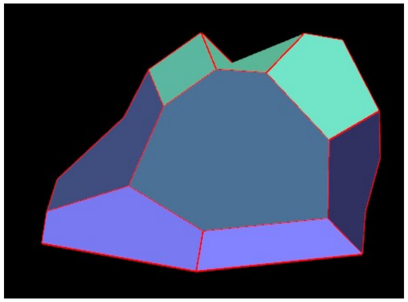
(Nearly) arbitrary polyhedra

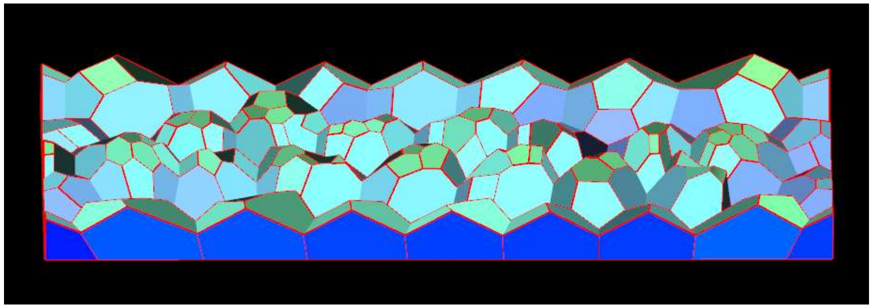
- Possibly non-planar faces







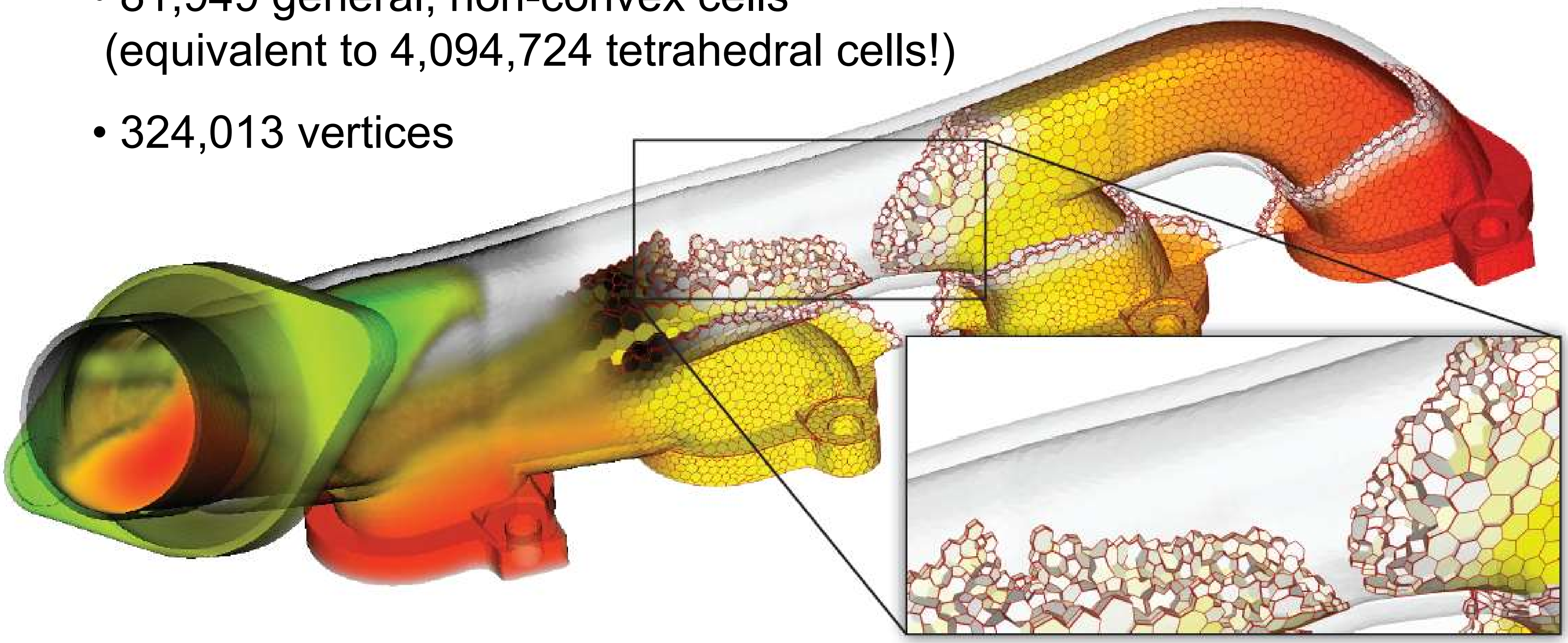




Example: General Polyhedral Cells

Exhaust manifold

- 81,949 general, non-convex cells
  (equivalent to 4,094,724 tetrahedral cells!)
- 324,013 vertices



- Color coding: temperature distribution

Unstructured Grid (Mesh) Data Structures

Unstructured 2D Grid: Direct Storage

Store list of vertices; vertices shared by triangles are replicated

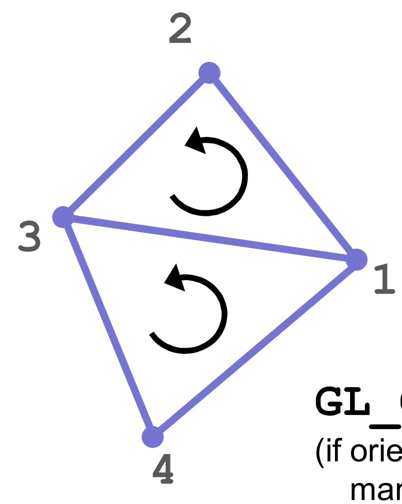
Render, e.g., with OpenGL immediate mode, …

coords for vertex 1 →

```
x1,y1,z1  ┐
x2,y2,z2  │ face 1
x3,y3,z3  ┘
x1,y1,z1  ┐
x3,y3,z3  │ face 2
x4,y4,z4  ┘

...
```

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



GL_CCW (if orientable manifold)

Redundant, large storage size, cannot modify shared vertices easily

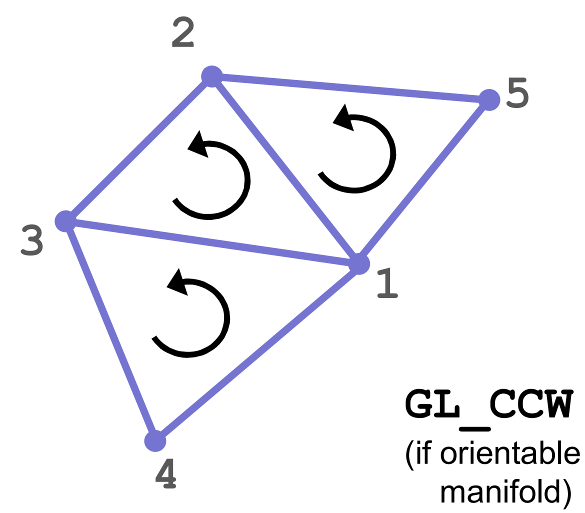
Store data values per face, or separately

Unstructured 2D Grid: Indirect Storage

Indexed face set: store list of vertices; store triangles as indexes

Render using separate vertex and index arrays / buffers

coords for vertex 1	vertex list	face list
	`x1,y1,(z1)`	`1,2,3`
	`x2,y2,(z2)`	`1,3,4`
	`x3,y3,(z3)`	`2,1,5`
	`x4,y4,(z4)`	`...`
	`...`	



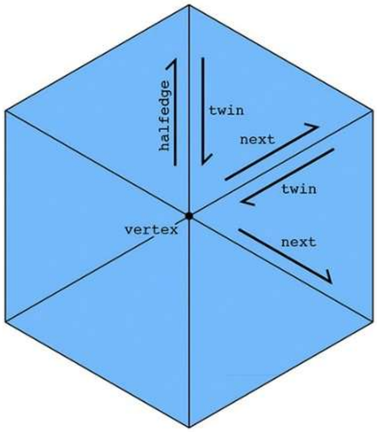
Less redundancy, more efficient in terms of memory

Easy to change vertex positions; still have to do (global) search for shared edges (local information)

Unstructured 2D Grids: Connectivity/Incidence

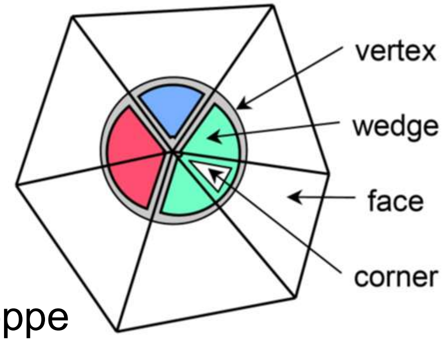
Half-edge (doubly-connected edge list) data structure

- Pointer to half-edge (twin) in neighboring face (mesh needs to be orientable 2-manifold)
- Pointer to next half-edge in same face
- Half-edge associated with one vertex, edge, face



Modifications: attributes, mesh simplification, …

- Vertices, corners, wedges, faces
- Express attribute continuity vs. discontinuity



Hugues Hoppe

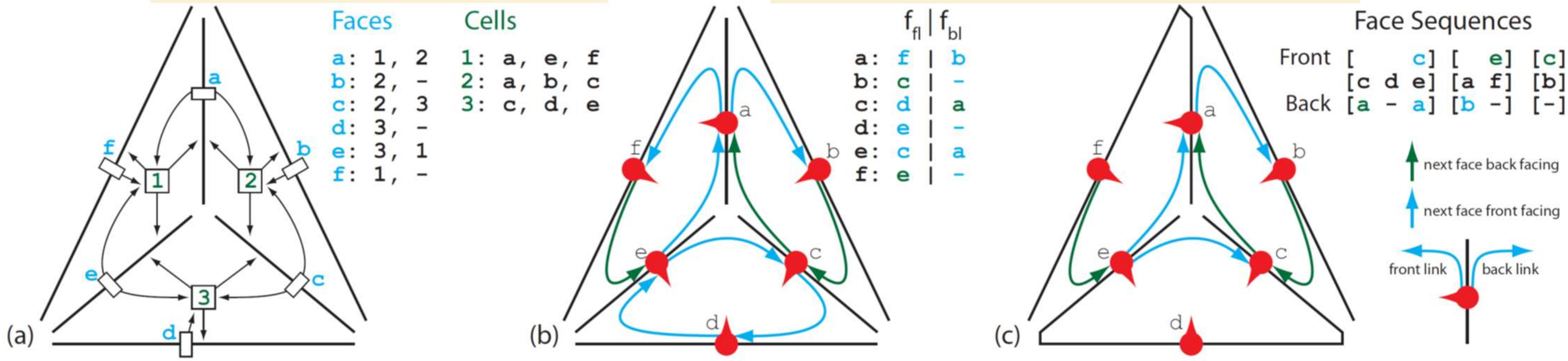
Visualization often needs volumetric version of these ideas (tet meshes, polyhedral meshes, …)

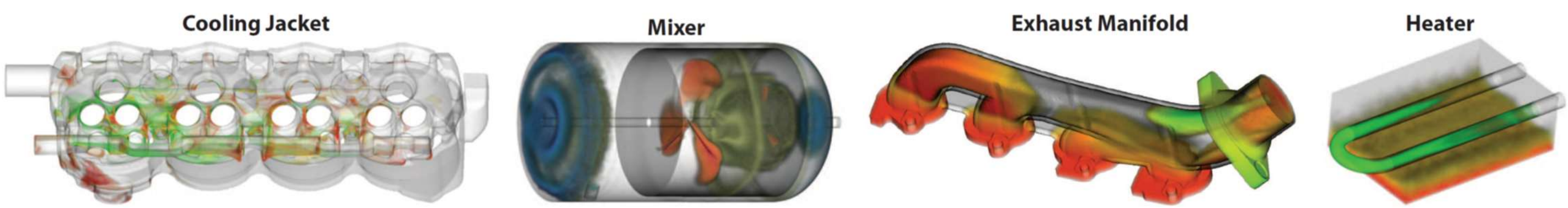
3D Grids: Two-Sided Face Sequence Lists

General polyhedral grids (arbitrary polyhedral cells); example: TSFSL (Muigg et al., 2011)

standard face/cell incidence

two-sided face (sequence) lists





	Cooling Jacket	Mixer	Exhaust Manifold	Heater
Cells/Vertices/Faces:	1,538K / 1,631K / 4,707K	1,362K / 7,432K / 8,869K	82K / 324K / 441K	17K / 68K / 91K
Tetrahedra:	17,044K (~8.5 byte/tet)	89,417K (~7.5 byte/tet)	4,095K (~7.0 byte/tet)	851K (~7.0 byte/tet)
Celltypes:	tets/pyramids/wedges/hexas	general (non-convex) polyhedra	general (non-convex) polyhedra	general (non-conv.) polyh.
Bricks/Cell Overhead:	4/1.7%	10/8.6%	1/0%	1/0%
TSFSL Creation Time:	4.0s	9.0s	1.7s	1.0s

Scalar Fields

Scalar Fields are Functions

- 1D scalar field: $\Omega \subseteq R \rightarrow R$
- 2D scalar field: $\Omega \subseteq R^2 \rightarrow R$
- 3D scalar field: $\Omega \subseteq R^3 \rightarrow R$

  $\rightarrow$ **volume visualization!**

more generally: $\Omega \subseteq$ n-manifold

Basic Visualization Strategies

Mapping to geometry

- Function plots
- Height fields
- Isocontours/isolines, isosurfaces

Color mapping

Specific techniques for 3D data

- Indirect volume visualization
- Direct volume visualization
- Slicing

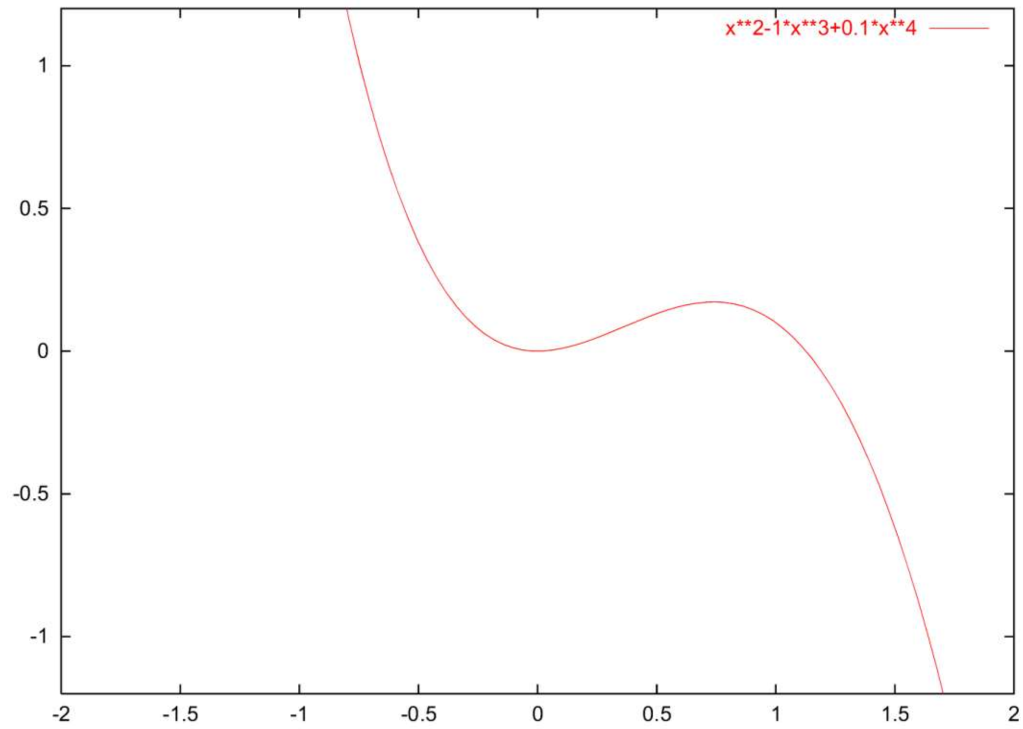
Visualization methods depend heavily on dimensionality of domain

Function Plots and Height Fields (1)

Function plot for a 1D scalar field

$$\{(x, f(x)) | x \in \mathbb{R}\}$$

- Points
- 1D manifold: line

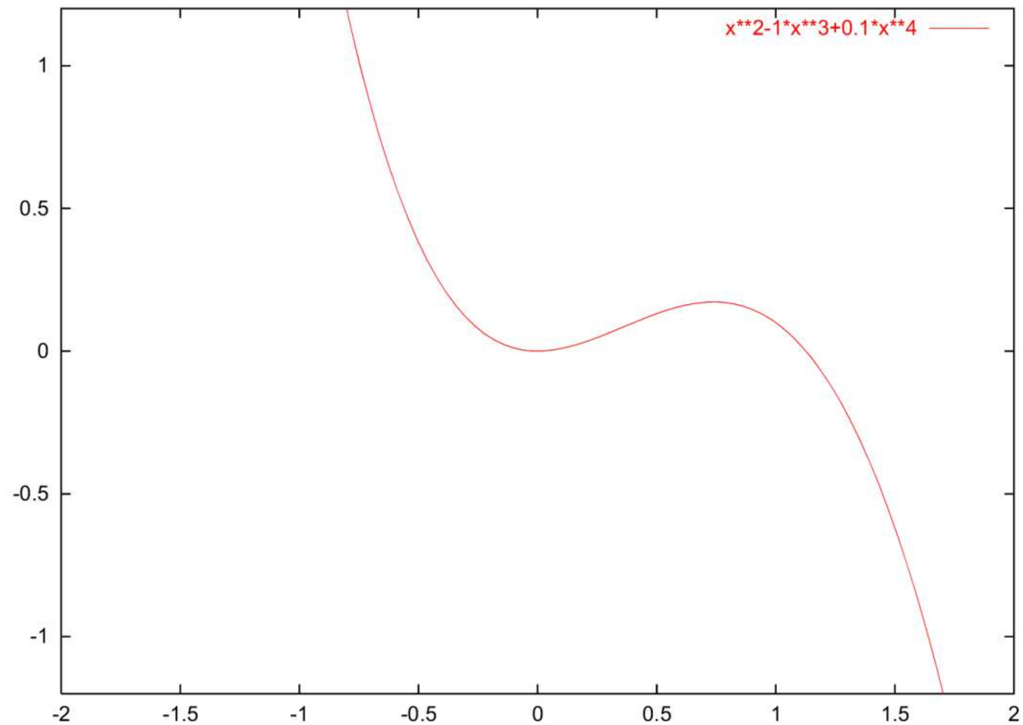


Function Plots and Height Fields (1)

Function plot for a 1D scalar field

$$\{(s, f(s)) \,|\, s \in \mathbb{R}\}$$

- Points
- 1D manifold: line



Function Plots and Height Fields (2)

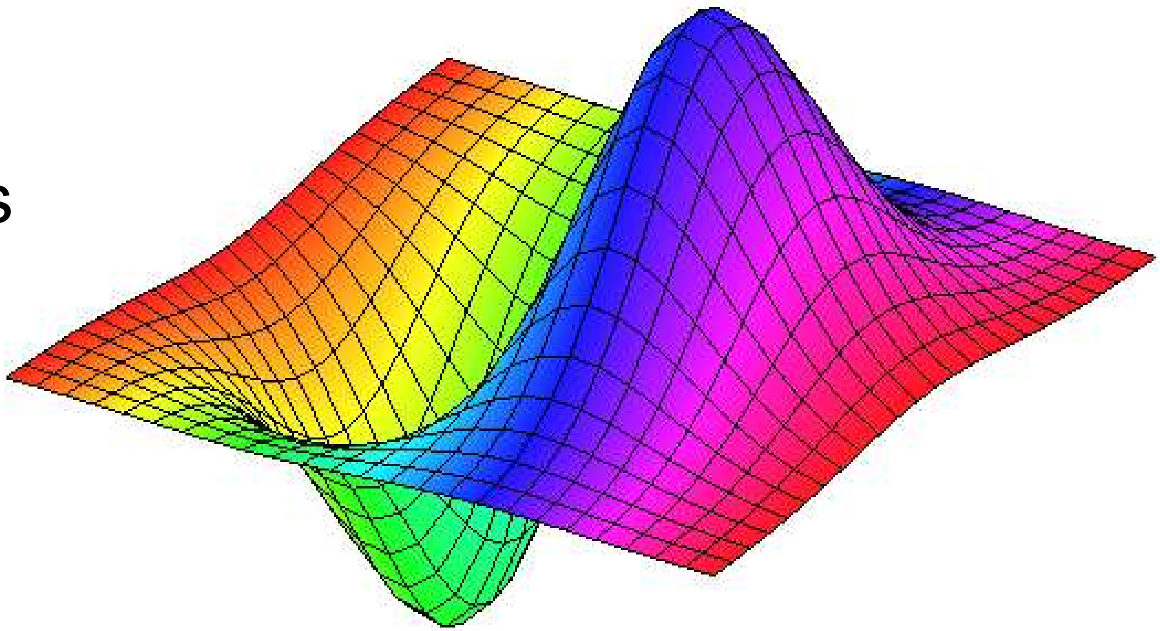
Function plot for a 2D scalar field

$$\{(x, f(x)) | x \in \mathbb{R}^2\}$$

- Points
- 2D manifold: surface

Surface representations

- Wireframe
- Hidden lines
- Shaded surface



Function Plots and Height Fields (2)

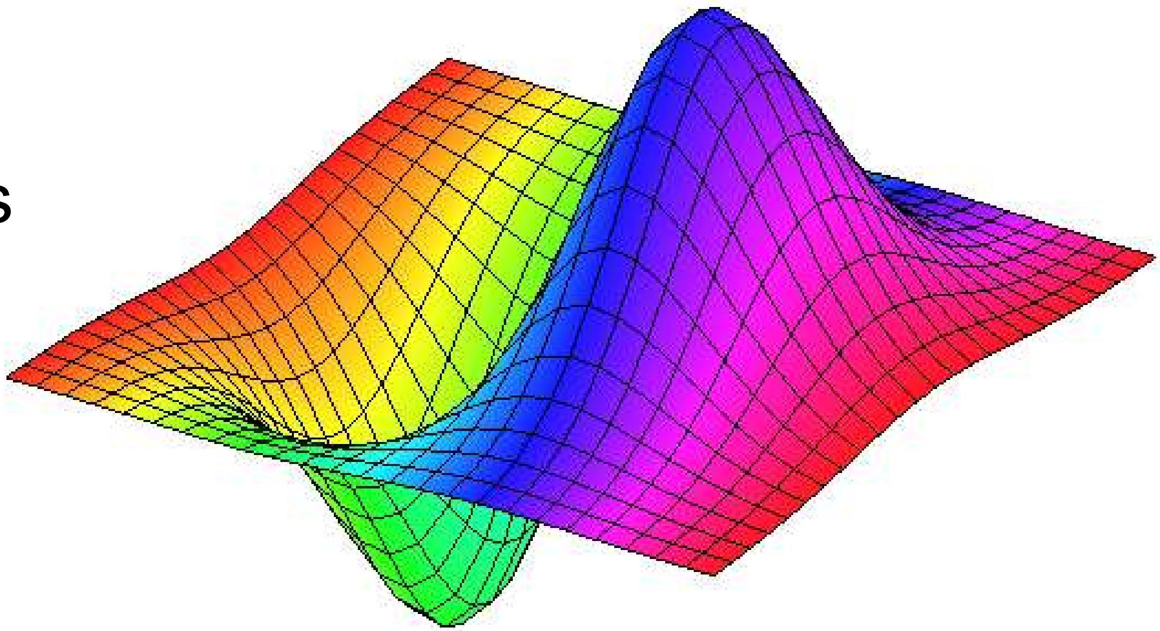
Function plot for a 2D scalar field

$$\{(s,t,f(s,t))\,|\,(s,t)\in\mathbb{R}^2\}$$

- Points
- 2D manifold: surface

Surface representations

- Wireframe
- Hidden lines
- Shaded surface

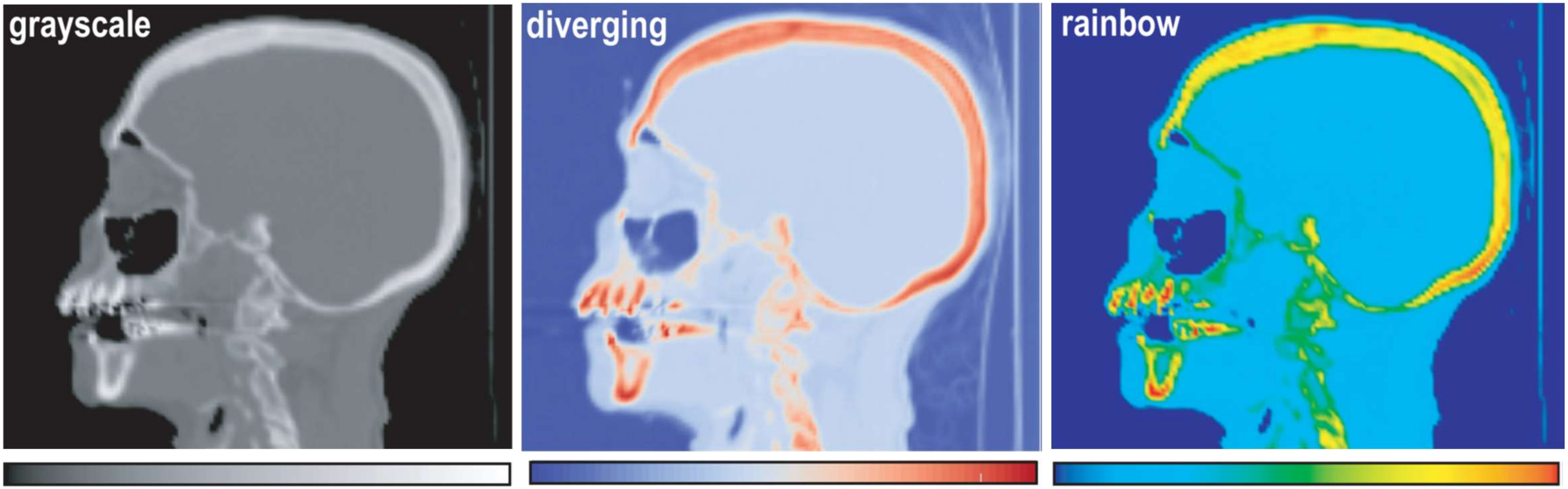


Color Mapping / Color Coding

Map scalar value to color

- Color table (e.g., array with RGB entries)
- Procedural computation; manual specification

With opacity (alpha value “A”): 1D *transfer function* (RGBA table, …)



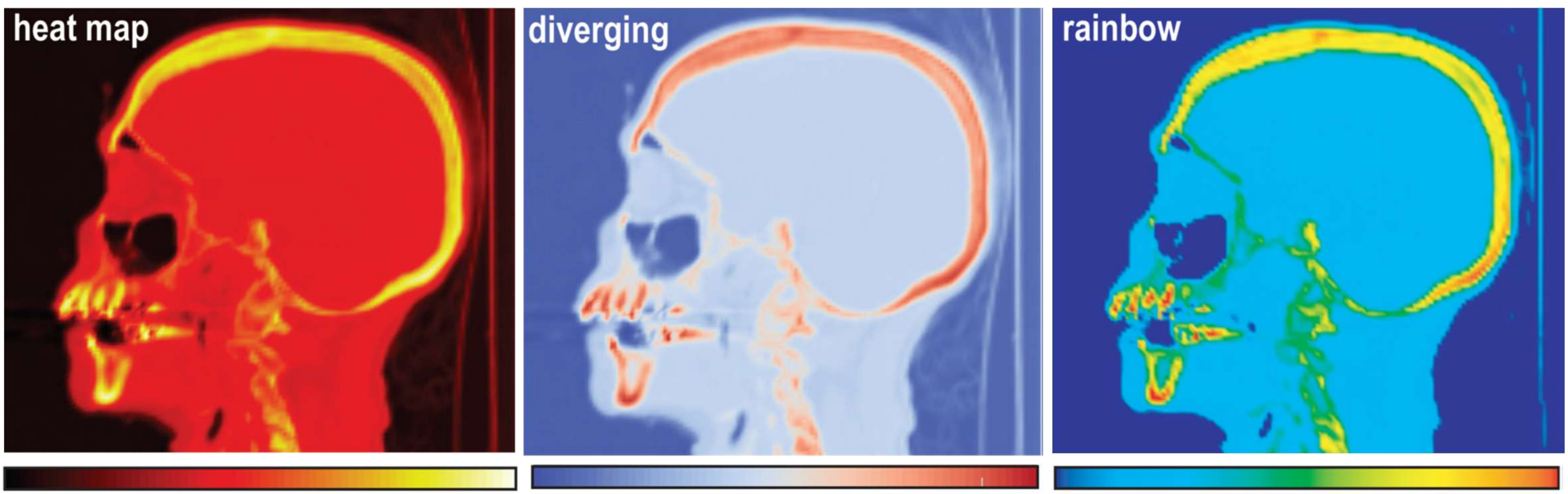
not recommended!

Color Mapping / Color Coding

Map scalar value to color

- Color table (e.g., array with RGB entries)
- Procedural computation; manual specification

With opacity (alpha value “A”): 1D *transfer function* (RGBA table, …)

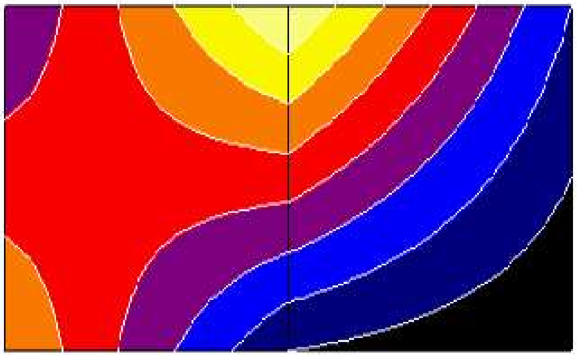


not recommended!

Contours

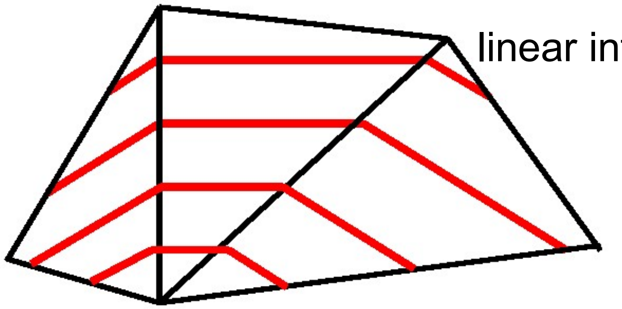
Set of points where the scalar field *s* has a given value *c*:

$$S(c) := f^{-1}(c) \qquad S(c) := \{x \in \mathbb{R}^n : f(x) = c\}$$



Common contouring algorithms

- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra



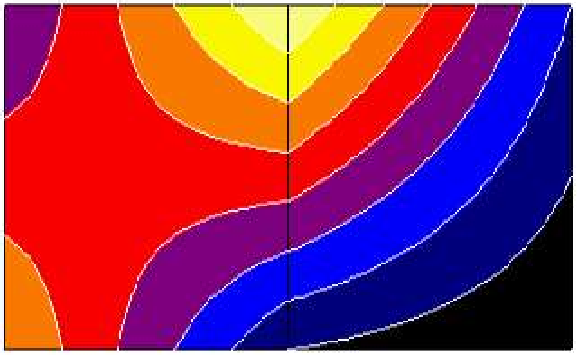
Implicit methods

- Point-on-contour test
- Isosurface ray-casting

Contours

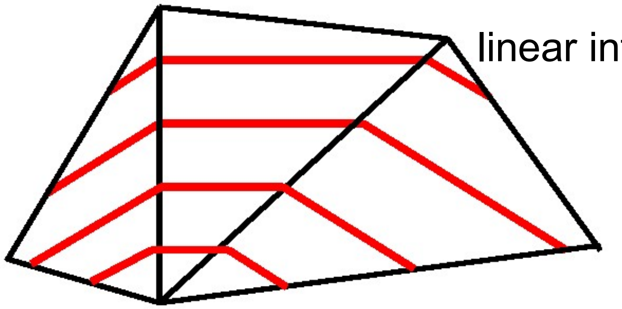
Set of points where the scalar field *s* has a given value *c*:

$$S(c) := f^{-1}(c) \qquad S(c) := \{x \in \mathbb{R}^2 : f(x) = c\}$$



Common contouring algorithms

- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra



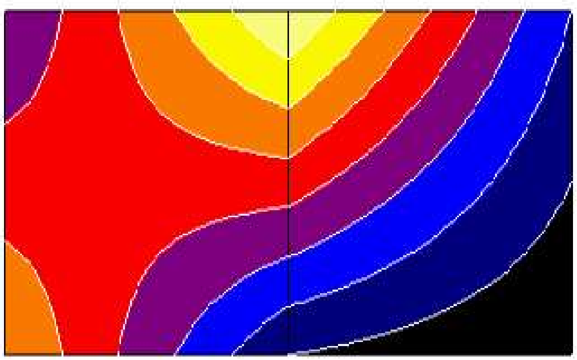
Implicit methods

- Point-on-contour test
- Isosurface ray-casting

Contours

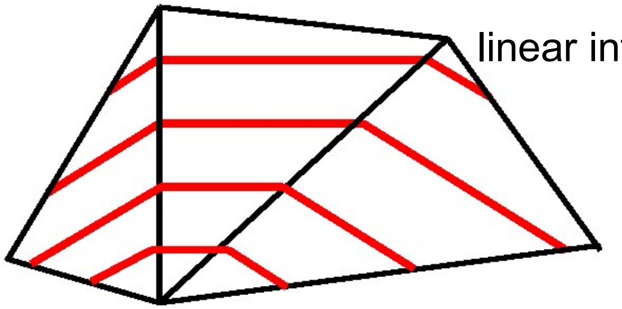
Set of points where the scalar field *s* has a given value *c*:

$$S(c) := f^{-1}(c) \qquad S(c) := \{x \in \mathbb{R}^3 : f(x) = c\}$$



Common contouring algorithms

- 2D: marching squares, marching triangles
- 3D: marching cubes, marching tetrahedra



Implicit methods

- Point-on-contour test
- Isosurface ray-casting

What are contours?

Set of points where the scalar field *s* has a given value *c*:

$$S(c) := \{x \in \mathbb{R}^n : f(x) = c\}$$

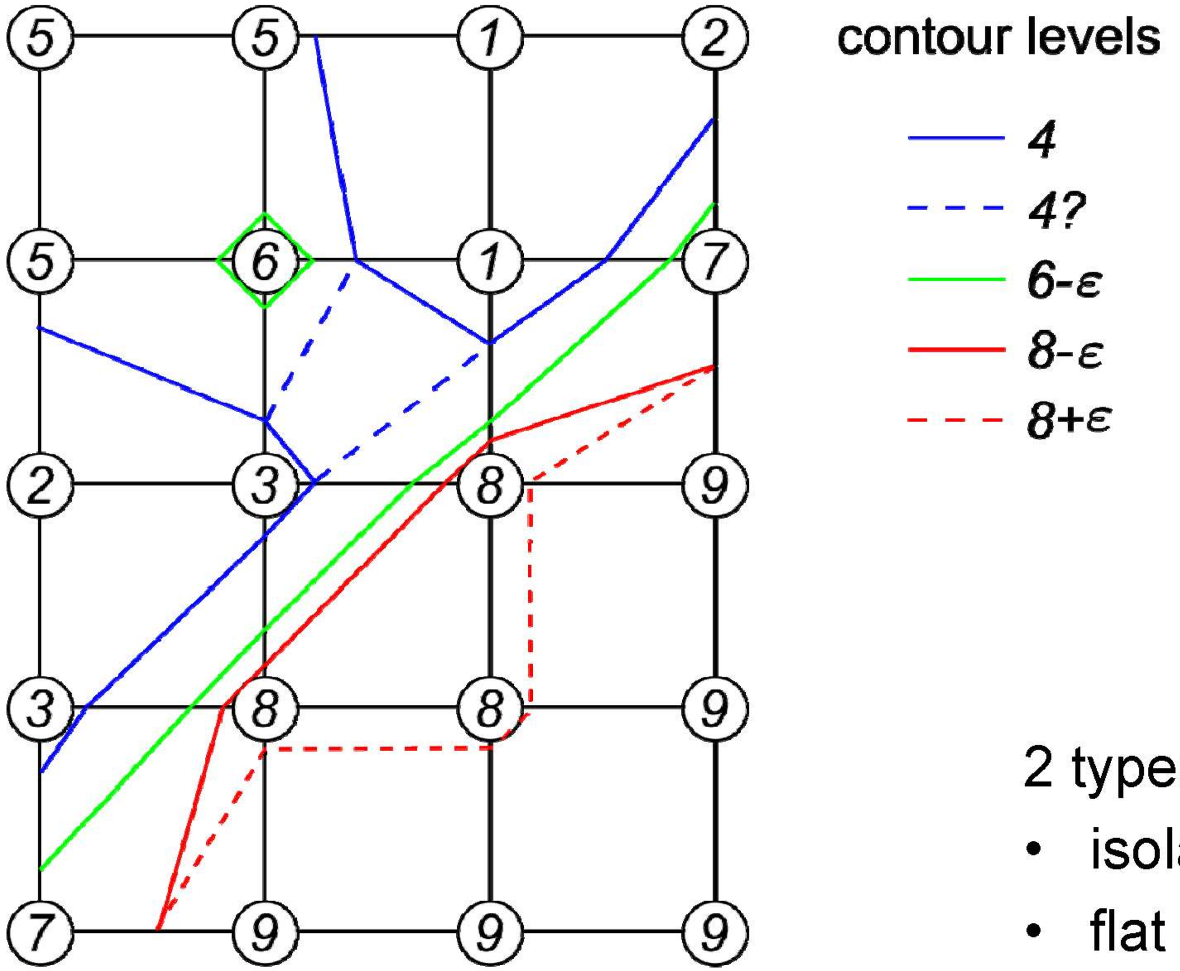
Examples in 2D:

- height contours on maps
- isobars on weather maps

Contouring algorithm:

- find intersection with grid edges
- connect points in each cell

Example



2 types of degeneracies:

- isolated points (*c*=6)
- flat regions (*c*=8)

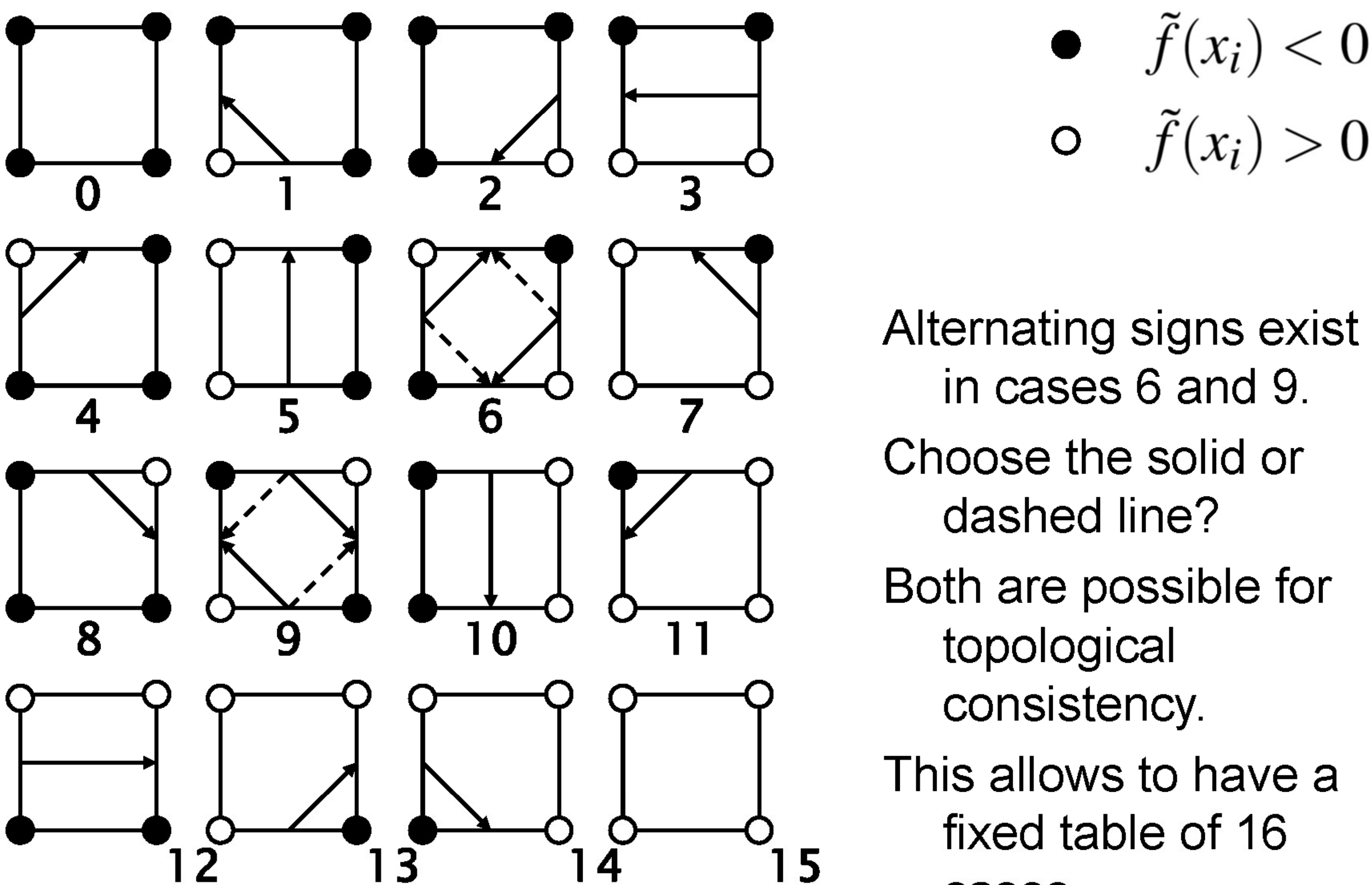
Basic contouring algorithms:

- cell-by-cell algorithms: simple structure, but generate disconnected segments, require post-processing
- contour propagation methods: more complicated, but generate connected contours

"Marching squares" algorithm (systematic cell-by-cell):

- process nodes in ccw order, denoted here as x_0, x_1, x_2, x_3
- compute at each node $\mathbf{x}_i$ the reduced field $\tilde{f}(x_i) = f(x_i) - (c - \varepsilon)$ (which is forced to be nonzero)
- take its sign as the i^{th} bit of a 4-bit integer
- use this as an index for lookup table containing the connectivity information:

Contours in a quadrangle cell



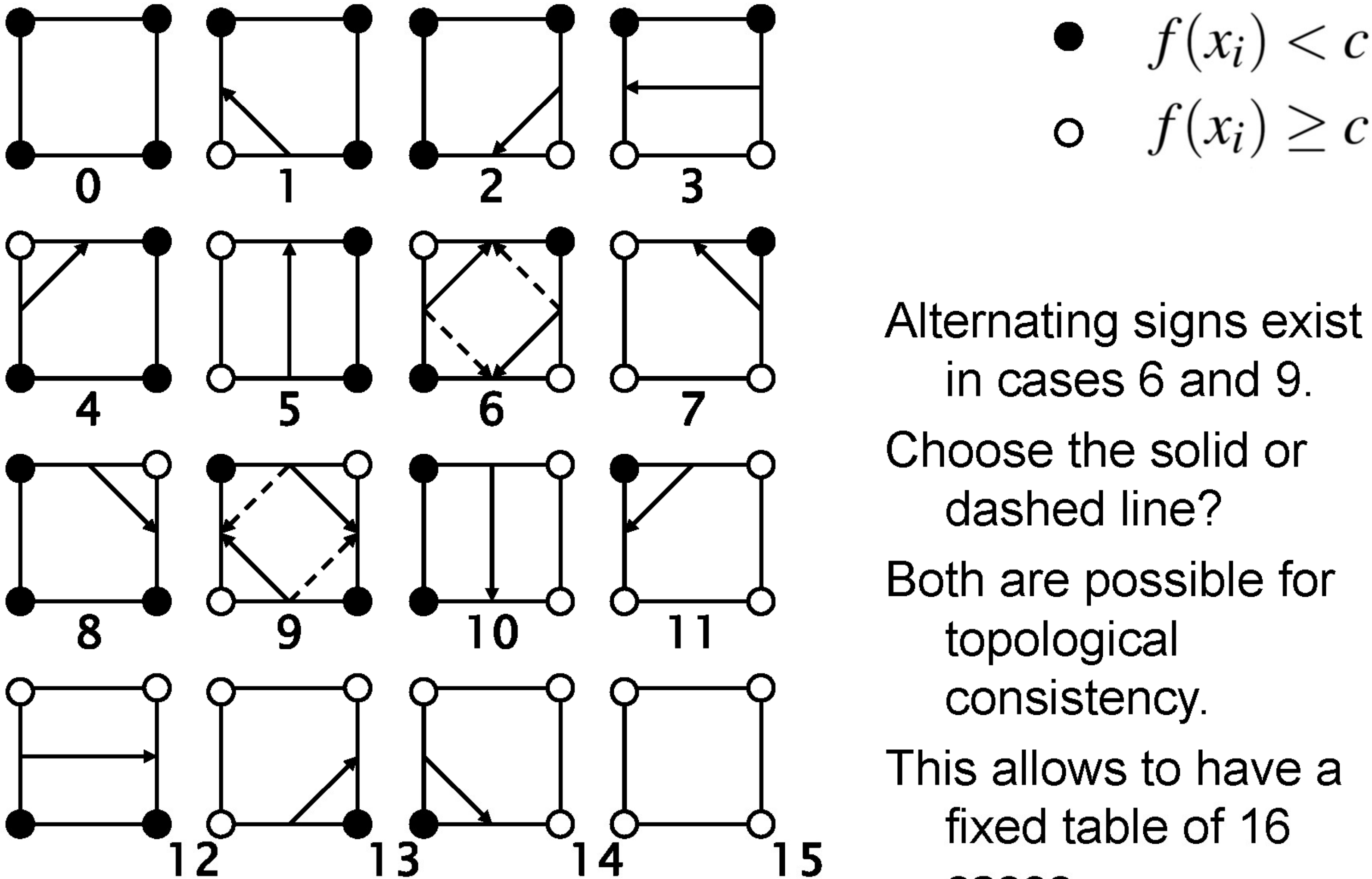
Alternating signs exist in cases 6 and 9.

Choose the solid or dashed line?

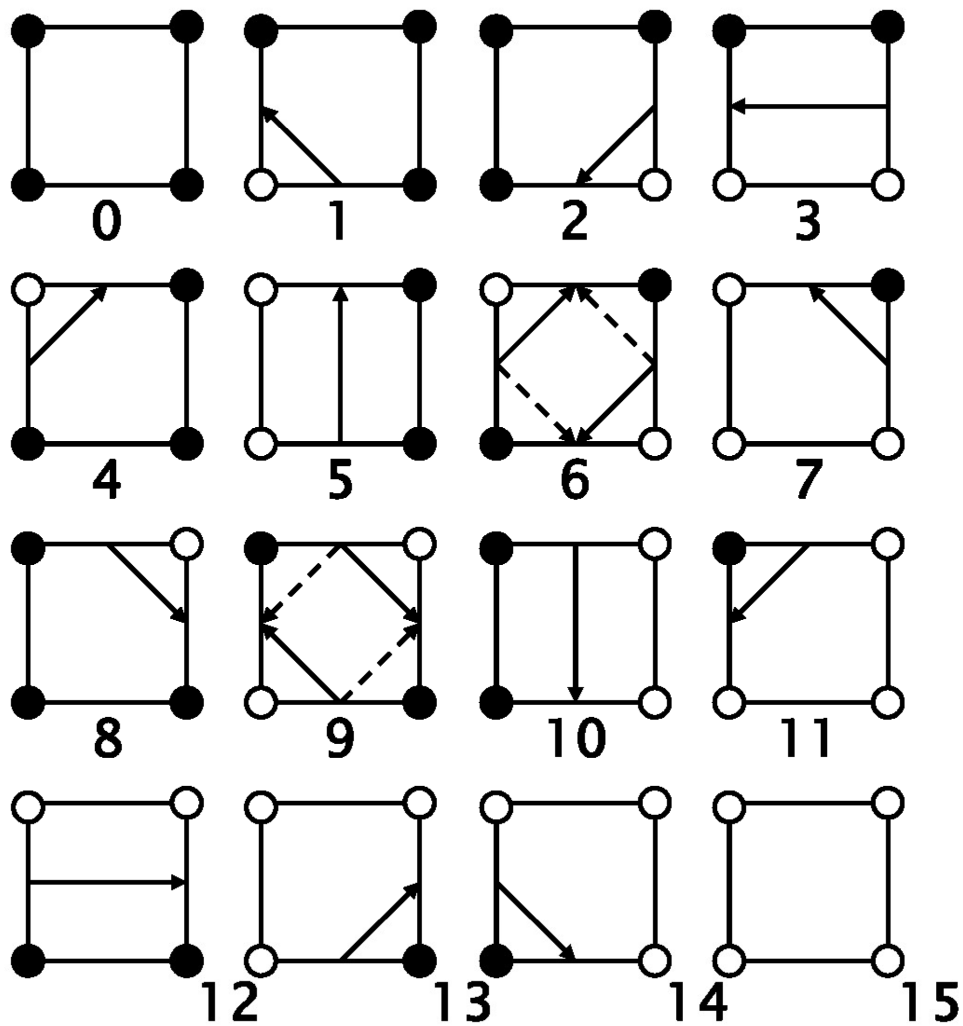
Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Contours in a quadrangle cell



- Alternating signs exist in cases 6 and 9.
- Choose the solid or dashed line?
- Both are possible for topological consistency.
- This allows to have a fixed table of 16 cases.



● $f(x_i) \le c$

○ $f(x_i) > c$

Alternating signs exist in cases 6 and 9.

Choose the solid or dashed line?

Both are possible for topological consistency.

This allows to have a fixed table of 16 cases.

Orientability (1-manifold embedded in 2D)

Orientability of 1-manifold:

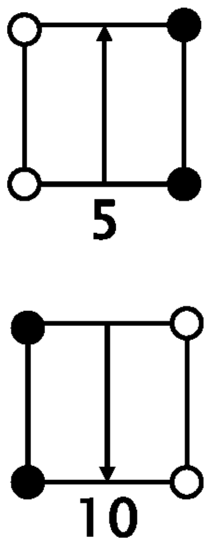
Possible to assign consistent left/right orientation

not orientable

Moebius strip
(only one side!)

Iso-contours

- Consistent side for scalar values…
  - greater than iso-value (e.g, *left* side)
  - less than iso-value (e.g., *right* side)
- Use consistent ordering of vertices
  (e.g., larger vertex index is "tip" of arrow;
  if (0,1) points "up", "left" is left, …)



● $\tilde{f}(x_i) < 0$

○ $\tilde{f}(x_i) > 0$

Orientability (2-manifold embedded in 3D)

Orientability of 2-manifold:

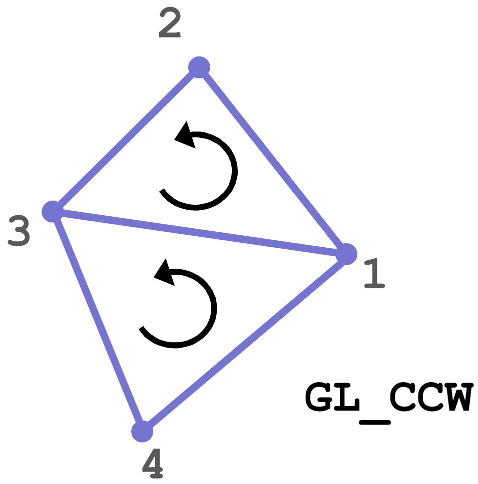
Possible to assign consistent normal vector orientation

not orientable

Moebius strip
(only one side!)

Triangle meshes

- Edges
  - Consistent ordering of vertices: CCW (counter-clockwise) or CW (clockwise) (e.g., (3,1,2) on one side of edge, (1,3,4) on the other side)
- Triangles
  - Consistent front side vs. back side
  - Normal vector; or ordering of vertices (CCW/CW)
  - See also: "right-hand rule"



Thank you.

Thanks for material

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