Data Structures for Computer Graphics

Object Based and Image Based Representations in 2D and 3D

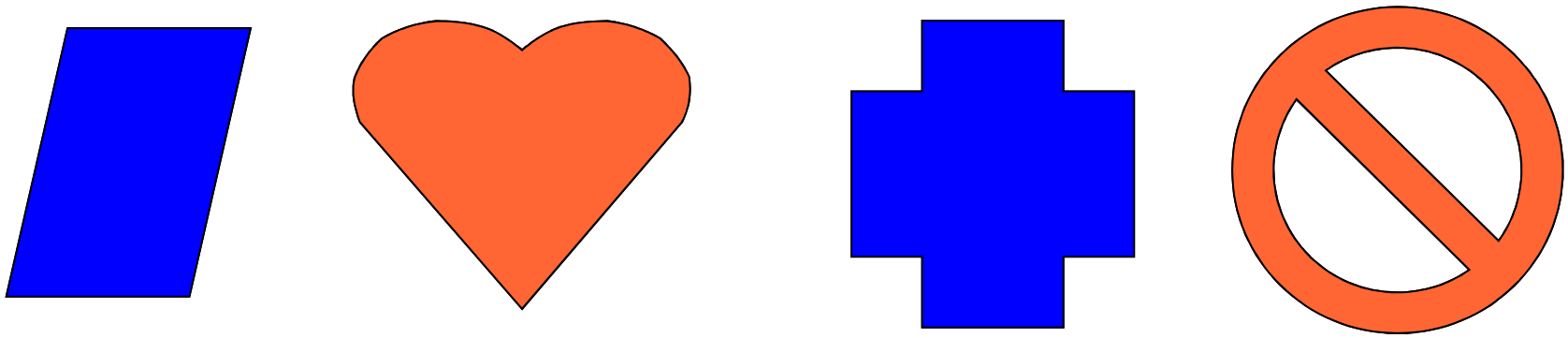
Lectured by Vlastimil Havran

Problem Definition

- What is an object in computer graphics?
- What is a shape of an object?
- How to represent an object?

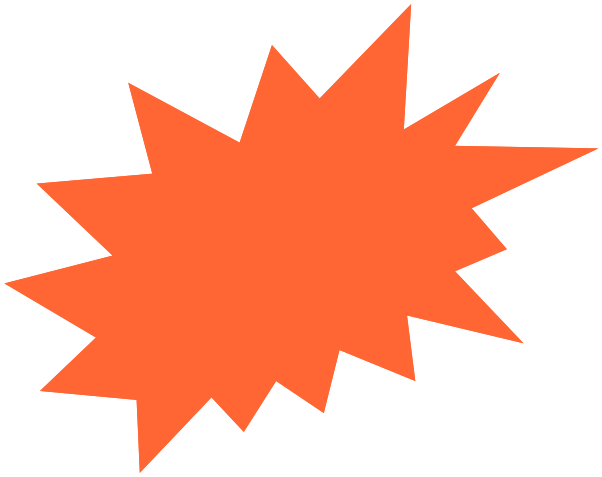
Object Definition

- Object is defined by **containment**: either a point is inside the object or not.
- Physical interpretation is easy: solid object
- Examples of 2D objects:

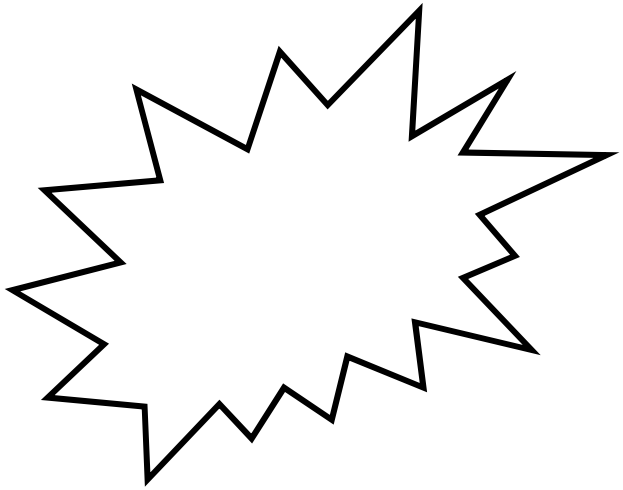


Basic Representations for Shape

- *Interior-based representation*: the interior of the object is explicitly represented



- *Boundary-based representation*: the boundary is explicitly represented



Major Differences

Interior based rep.	Boundary based rep.
• *Advantage*: Determination whether a point is inside the object or not is easy • *Disadvantage*: it is usually memory demanding	• *Advantage*: it is usually memory efficient • *Disadvantage*: determination whether a point is inside the object is less efficient (it can require to implement ray shooting)

Further Lecture Content

- Interior based representation
  - Cell based representations
  - Hierarchical representations
  - CSG
- Boundary based representation
- Other (special) representations
  - Image pyramid
  - Heightfield
  - Voxel representation

INTERIOR BASED OBJECT REPRESENTATIONS

Interior-Based Representations

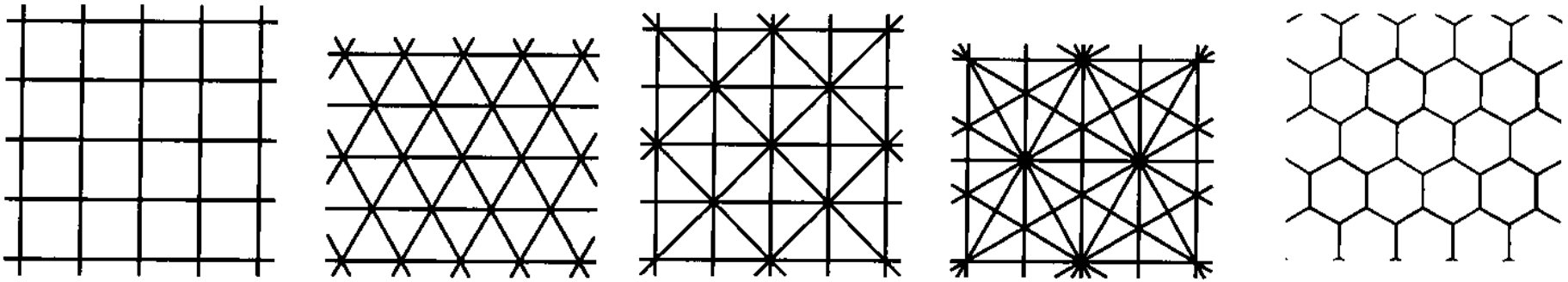
Major classification:

- A subset of unit cells (using various space-ordering of cells)
- A collection of some basic shapes
  - Only additive operation (OR)
  - Operations: **OR, XOR, MINUS**: Constructive Solid Geometry (CSG)
- Objects defined by tree leaves in n-dimensional space

A Subset of Unit Cells

- We need to specify a unit cell + ordering + containment (which cell is full/empty)
- A 2D picture is a special case of this representation
- We enumerate whether a point belongs to the shape or not
- Several basic tiling types in 2D defining cell shapes:

grid triangle isoceles triangle 30-60-90 triangle hexagon

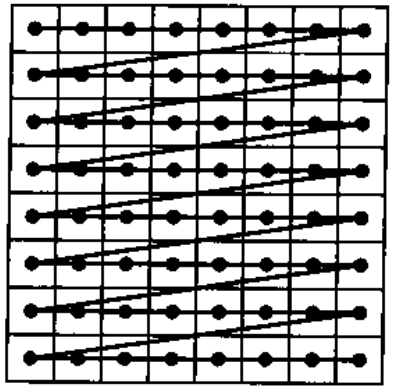


Space-Ordering of Unit Cells

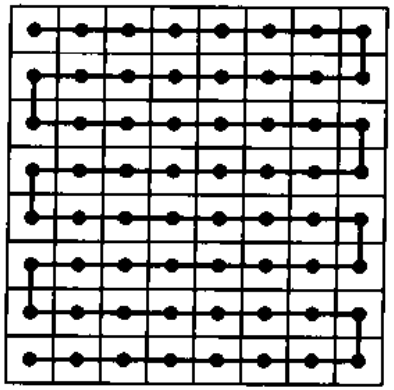
- For a plane it is embedding/mapping of 2-dimensional space into a 1-dimensional space – we need it since memory in a computer is addressed linearly
- There are several types of ordering for 2D and 3D or high-dimensional space
- Basic motivation: data locality – two neighbors in world space are likely neighbors in the memory address space

Space-Ordering Types

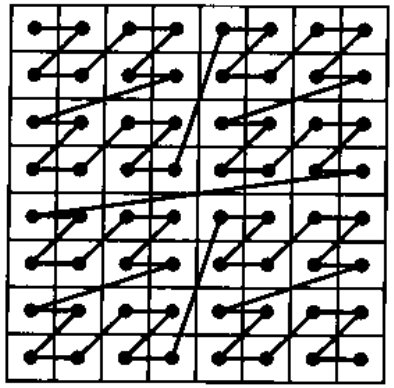
Row order



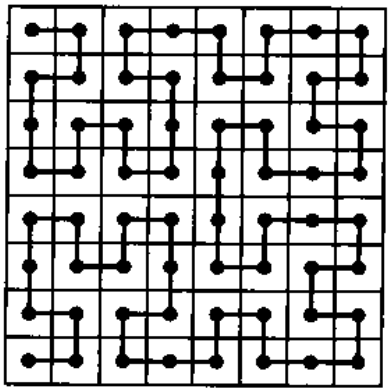
Row-prime order



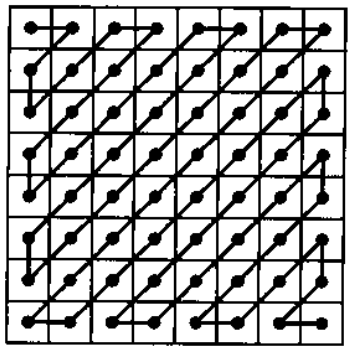
Morton order



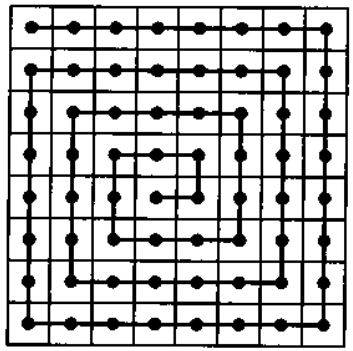
Peano-Hilbert order



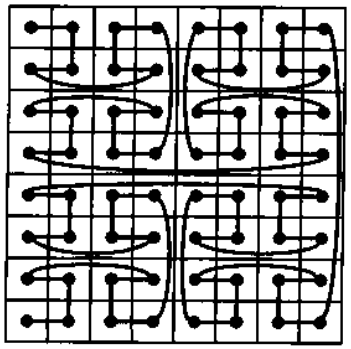
Cantor-diag. order



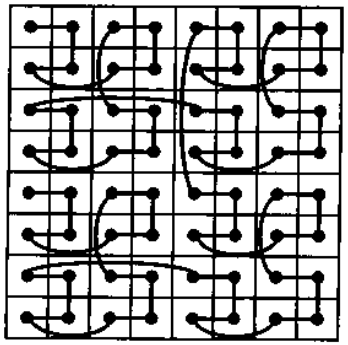
Spiral order



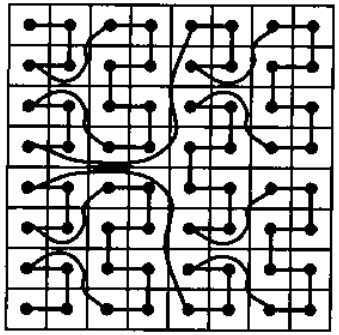
Gray order



double Gray order



U order



Space-Ordering Properties

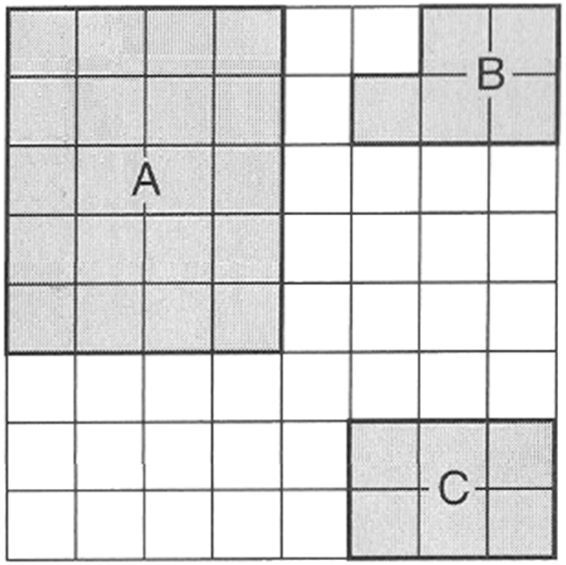
- Each location(=cell) exactly once (addressing)
- Mapping from higher (2D/3D) space to the 1D space should be *simple*, the same for inverse mapping (difficult for Peano-Hilbert order)
- The ordering should be ***stable***: the relative ordering of individual locations is preserved when the resolution is doubled (true for Morton, U, Gray, and double Gray orders).
- Two adjacent locations are also neighbors along the 1D curve and vice versa (*spatial locality*). For row-prime, Peano-Hilbert, and spiral orders the spatial locality is better.
- Algorithm for *retrieving neighbors* (=addresses of neighbors) should be simple.

Optional Homework Exercise

- Write the routines that make mappings from location (x,y) to address (A) for
  - Row order (you know already from 2D arrays)
  - Row-prime order
  - Morton order
  - Gray order
- Advanced task:
  - Peano-Hilbert order addressing
- Assume 2D array of size 2^N x 2^N

Interior-Based Object Representation using Unit Cells

- You just say which 2D/3D element of array is an object and which not.
- Example collection of three objects:



Morton order

	0	1	4	5	16	17	20	21
	2	3	6	7	18	19	22	23
A	8	9	12	13	24	25	28	29
	10	11	14	15	26	27	30	31
	32	33	36	37	48	49	52	53
	34	35	38	39	50	51	54	55
	40	41	44	45	56	57	60	61
	42	43	46	47	58	59	62	63

B

C

Problems of Unit Cells Representation

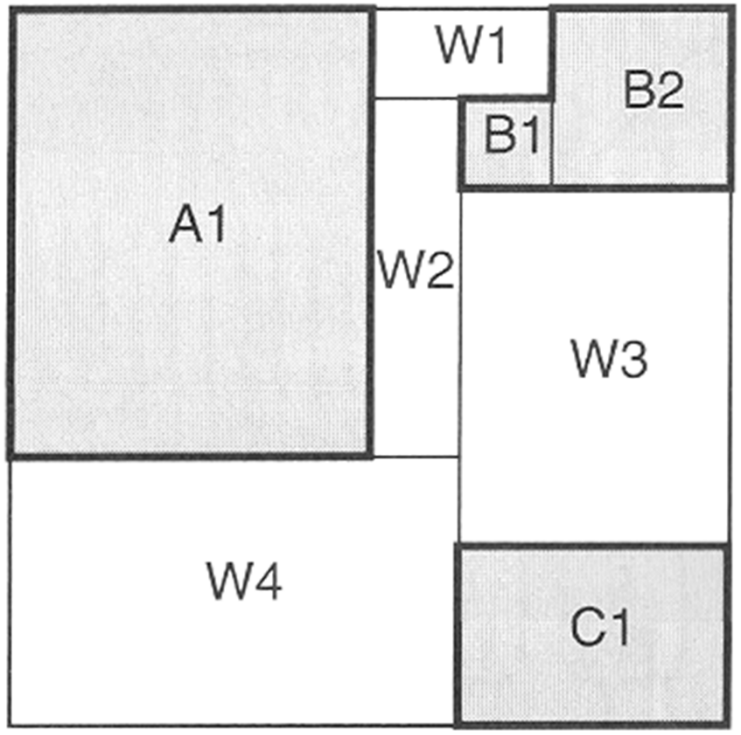
- Redundacy
  - Many empty/full cells
  - Space ordering partially helps
- How to avoid the redundancy:
  - Irregular decompositions
  - Hierarchical data structures

Irregular decompositions

- Decomposition into Arbitrary Sized Rectangular Blocks
- Irregular Grid Representation

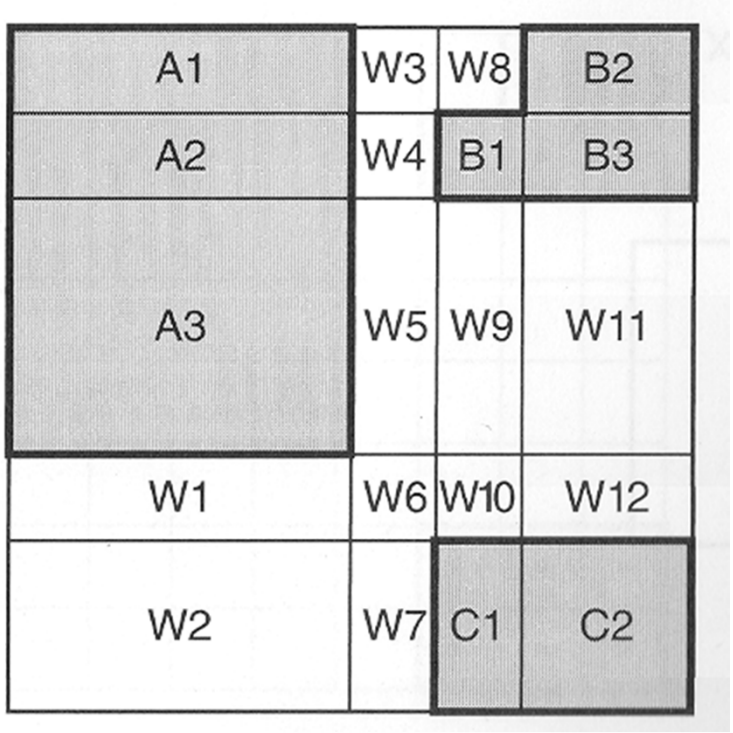
Decomposition into Arbitrary Sized Rectangular Blocks

- Find a minimum number of rectangles that cover an original shape: computationally expensive
- NP-complete problem if the region contains holes



Irregular Grid Representation

- The cells are given by an arbitrary irregular grid, which is specified separately



A1	W3	W8	B2
A2	W4	B1	B3
A3	W5	W9	W11
W1	W6	W10	W12
W2	W7	C1	C2

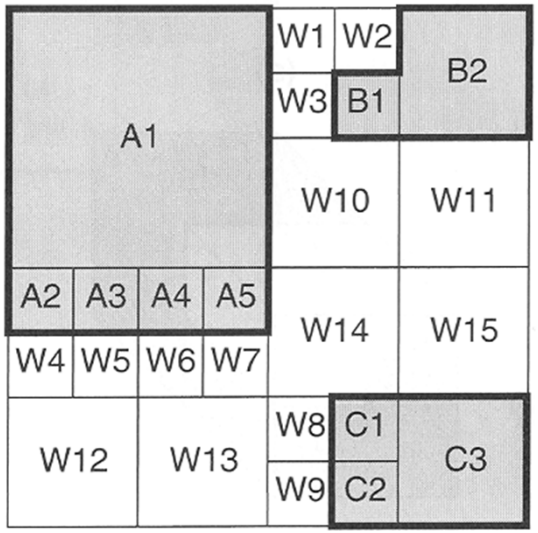
Array column	*x* range
1	[0,4)
2	[4,5)
3	[5,6)
4	[6,8)

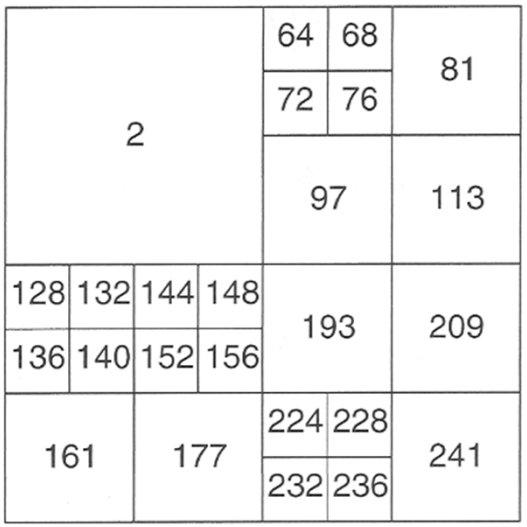
Array row	*y* range
1	[0,1)
2	[1,2)
3	[2,5)
4	[5,6)
5	[6,8)

Hierarchical representations

- Region quadtree
- Region octree
- Atree
- Region bintree
- Adaptive version of quadtree, octree, bintree
- Puzzle tree

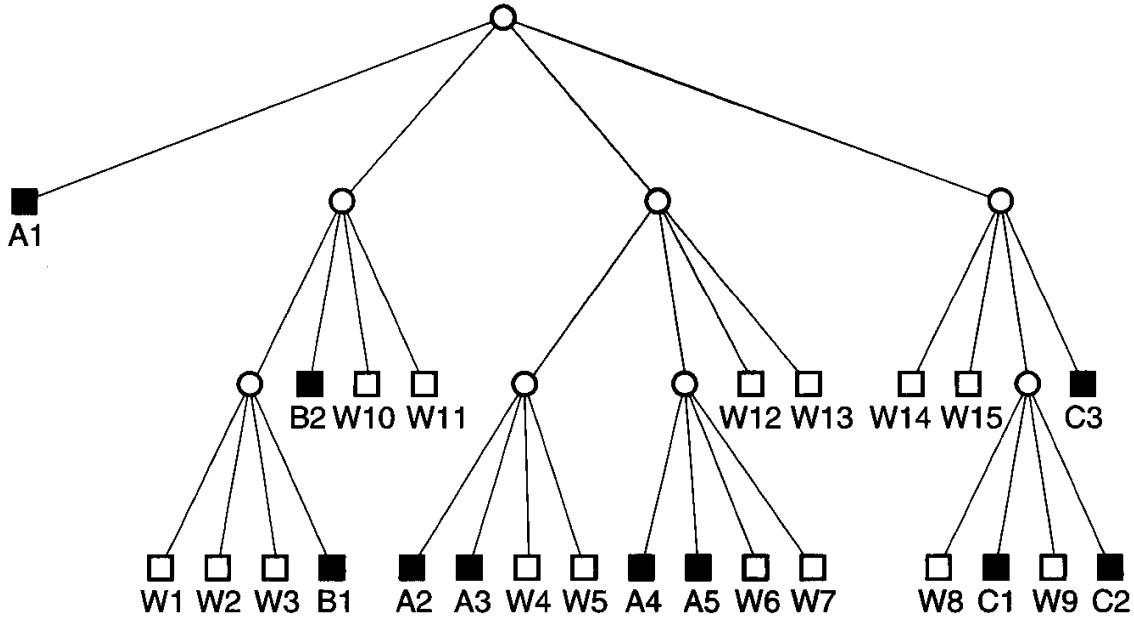
Region Quadtree





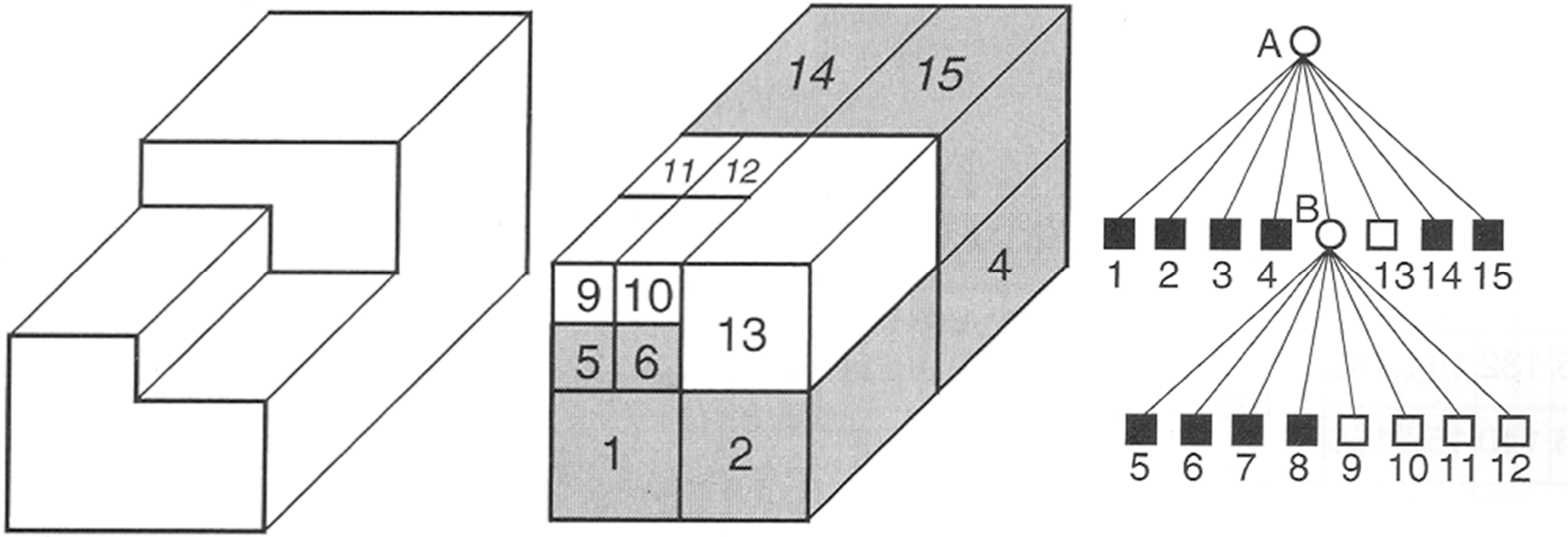
By telling interior nodes/full leaves/empty leaves we specify the shape of the object

Note that subdivision is always into four cells of the same size and shape!



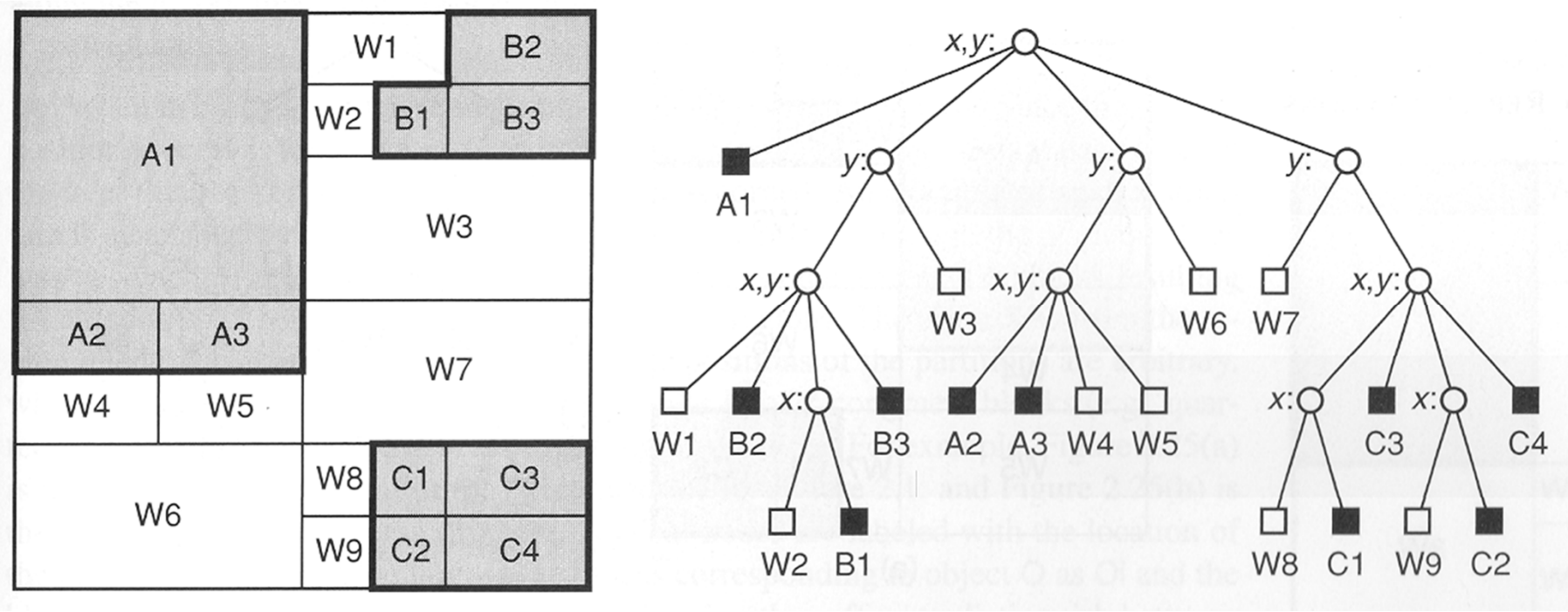
Analogy in 3D: Region Octree

- Each interior node has eight children



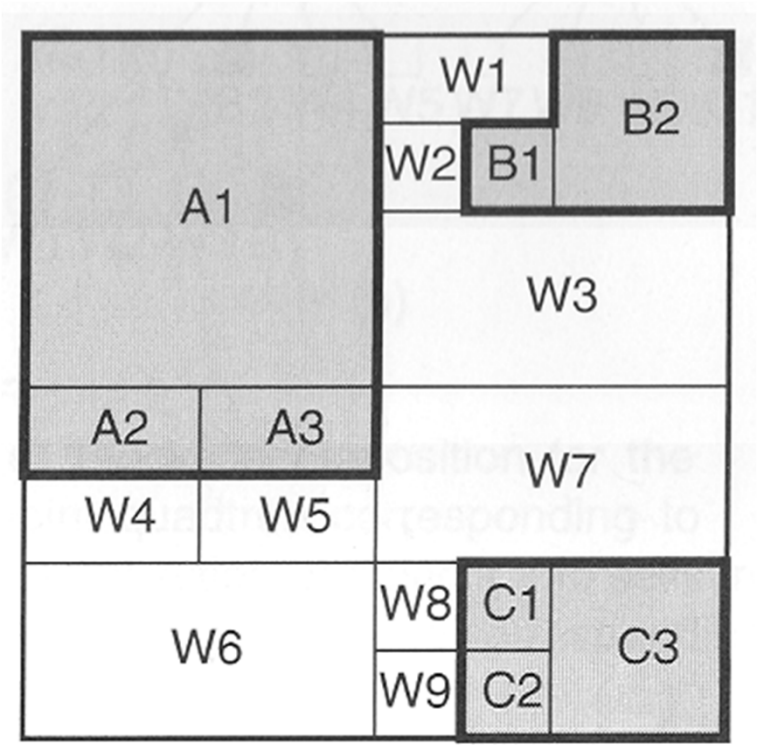
Atree

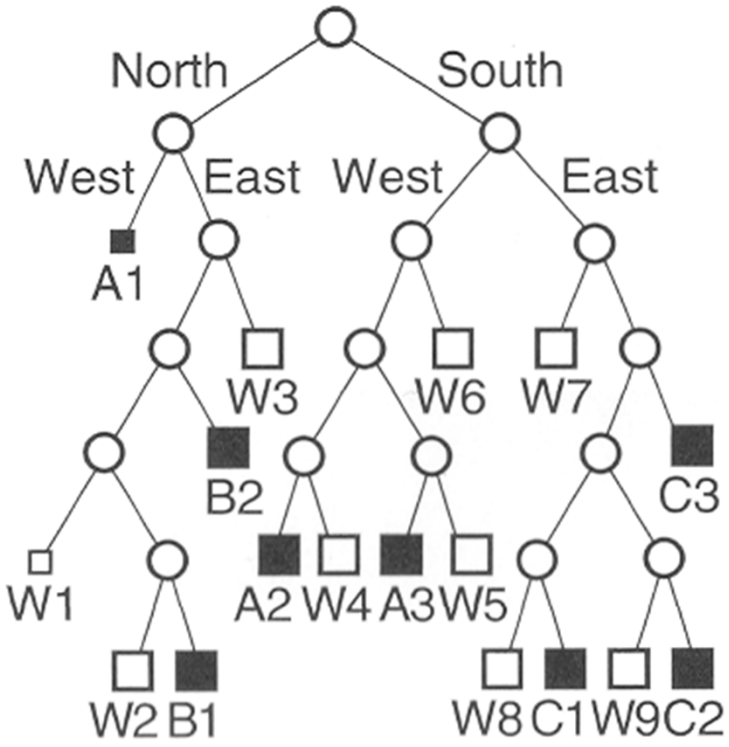
- Finer partitioning required by an application
- Arity is either two or four in 2D space (two types of interior nodes + two types of leaves)



Region Bintree (for higher dimensional data)

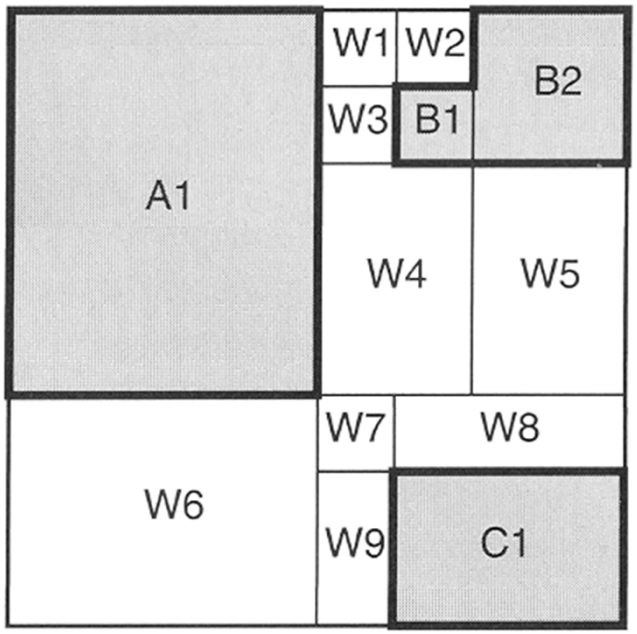
- Arity is always two, cyclic order of axes used for partitioning (x,y,x,y,x.. in 2D), (x,y,z,x,y…in 3D)
- Subdivision always into two cells of the same size and shape

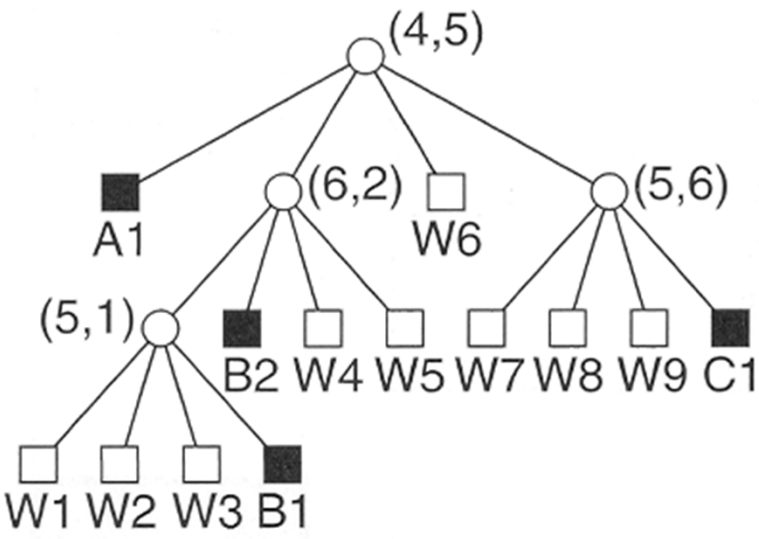




Adaptation of the Point Quadtree for Regions

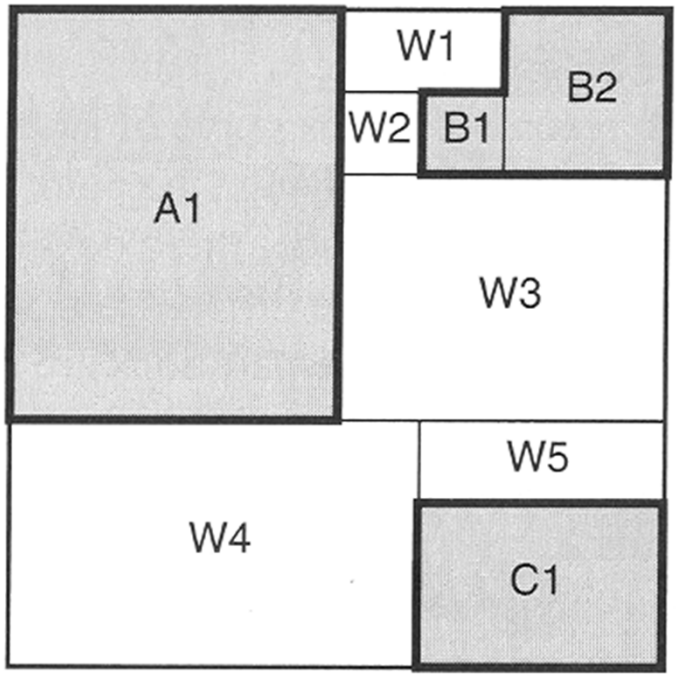
- In addition to standard region quadtree (splitting planes only in the middle) we specify the partitioning planes explicitly

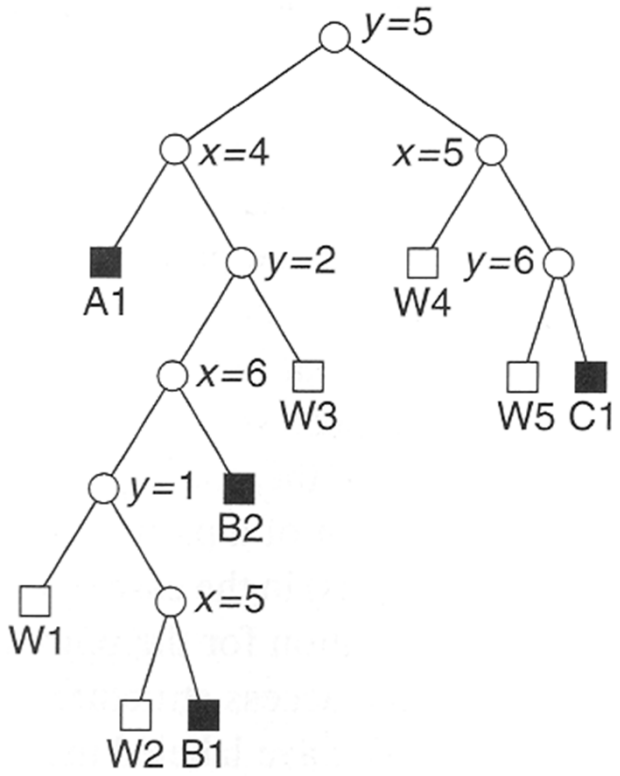




Adaptation of the Kd-tree for Regions

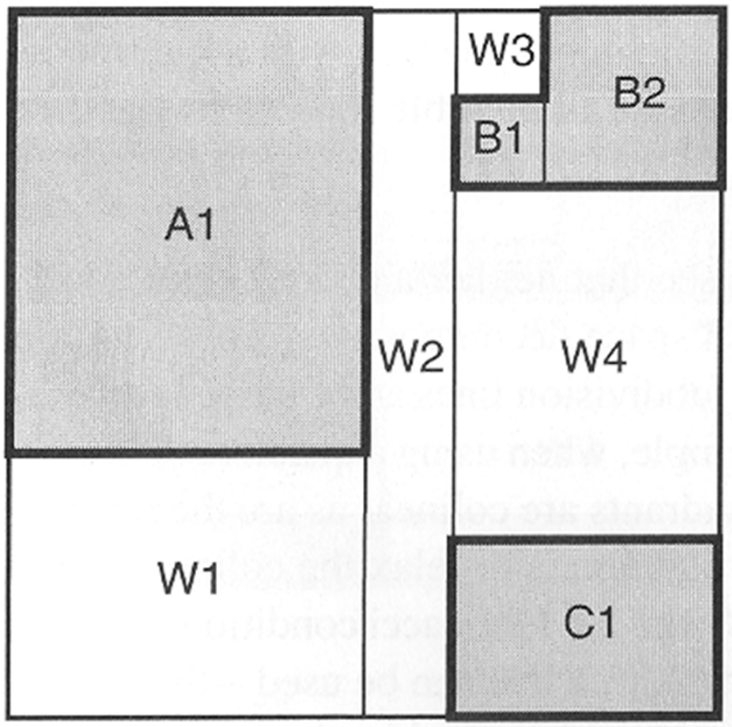
- In addition to standard region bintree (splitting planes only in the middle) we specify the partitioning planes explicitly

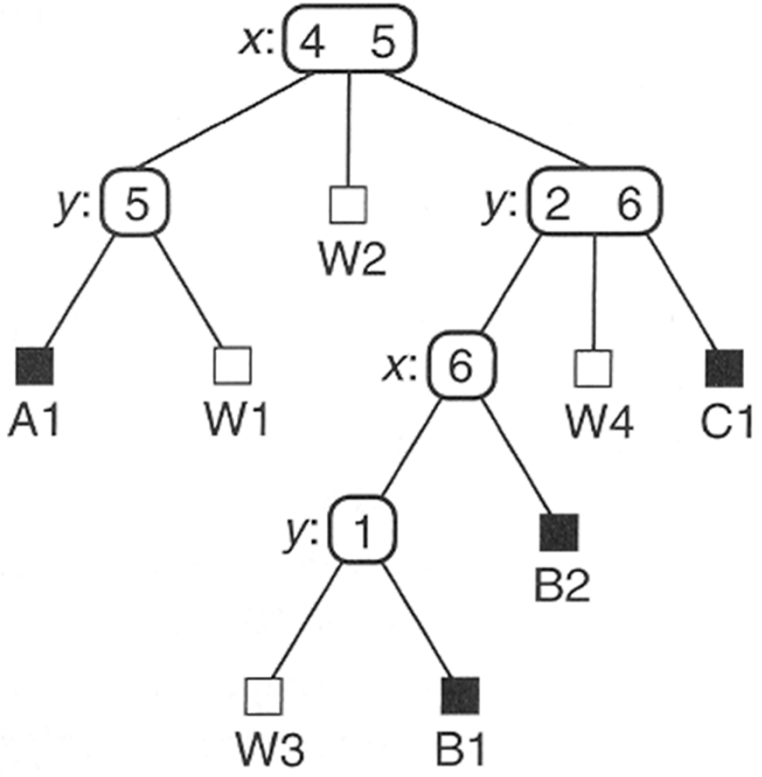




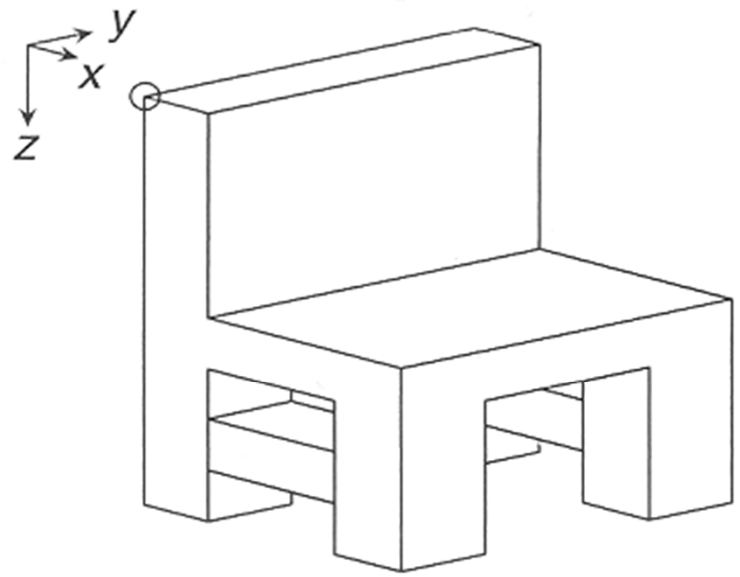
Puzzle Tree

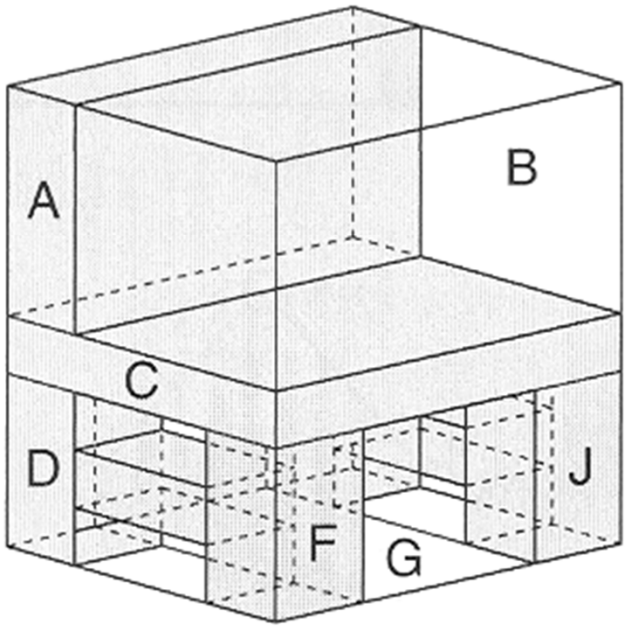
- Possibly more parallel partitioning planes at interior node (ternary or binary interior nodes)
- Position of partitioning planes stored explicitly

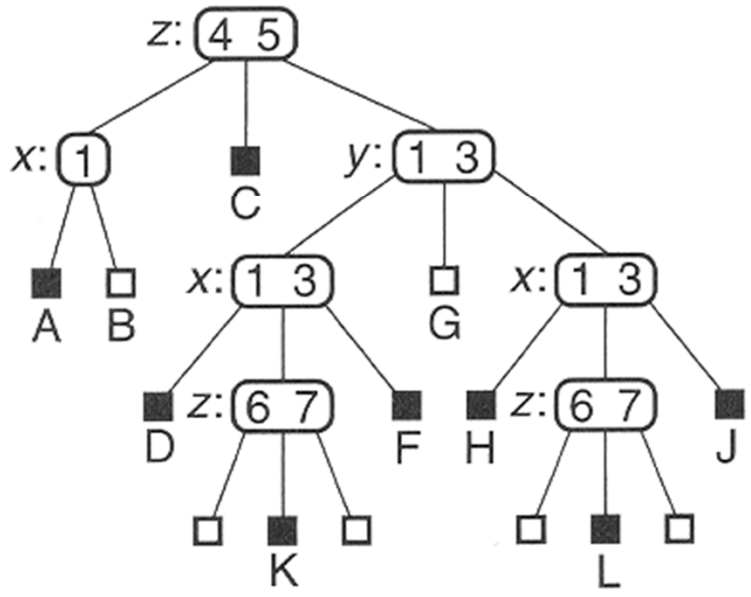




Puzzle Tree in 3D







Interior-Based Rep.: Summary

- Subdivision of some space either in the unit cells (same size) or using a tree
- Tree structures:
  - the arity of the interior nodes can be different
  - orientation of partitioning plane either implicit or explicit
  - position of partitioning planes are either implicit or explicit
- There is a tradeoff between space requirements and flexibility of data structures for various purposes

Special Representation like CSG

- Related to interior based representation, however not the same
- A collection of some basic shapes combined together with some operators

Interesting Operations over Regional Octree Data

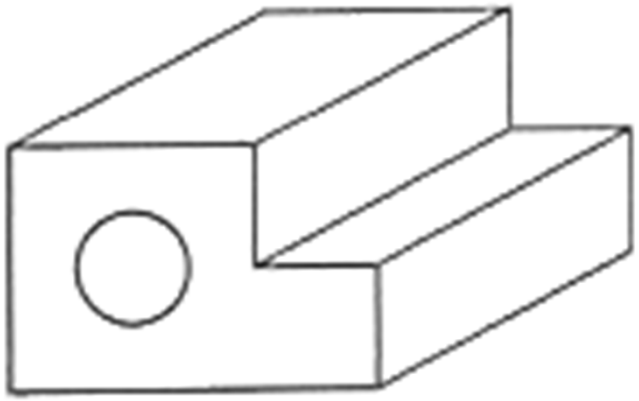
- **Input:** two objects A and B represented by regional octrees
- **Output:** object C
- **Operation:**
  - Merge two objects, operation OR
  - Subtract two objects, operation MINUS
  - Intersect two objects, operation AND
- Hint for the simple (but not efficient) algorithm:
  - traverse the tree for A until all leaves. In each leaf X_A, traverse the tree for B until you find the leaf X_B (or set of leaves) and combine the data from X_A and X_B to the leaf of output object C.
  - Time complexity is then $O(N_A \times \log N_B)$, where N_A is the number of leaves for tree of A and N_B is the number of leaves for tree of B
  - More efficient algorithm ... optimize the search in the tree of B

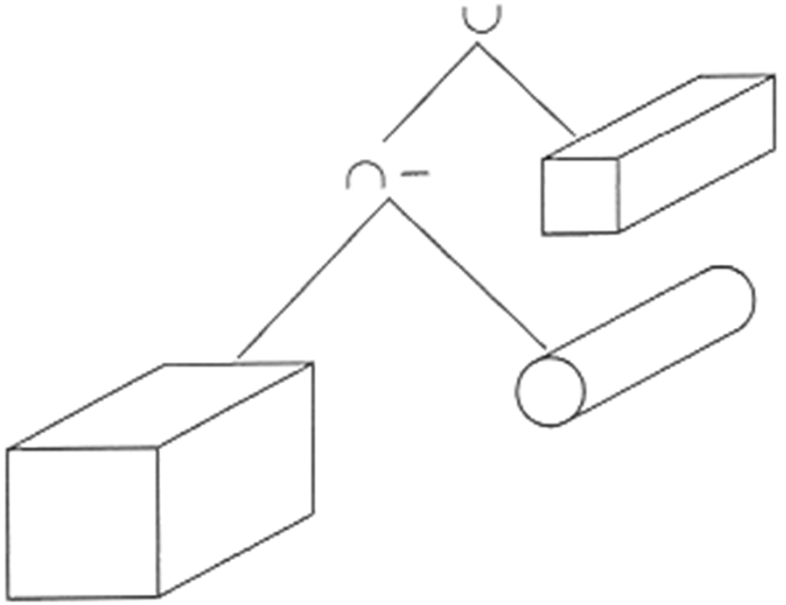
Collection of Basic Shapes

- Either only OR operation: R-trees, box-trees, R*-trees, etc. Objects are approximated by some basic shapes, typically by boxes in 2D or 3D.
- More complex combining operations
  - CSG
  - BSP trees

Constructive Solid Geometry (CSG)

- General tree with basic object shapes and combining operations OR,XOR,MINUS
- Used heavily in industry – the operations corresponds removal of material (drilling)





BOUNDARY BASED OBJECT REPRESENTATIONS

often abbreviated
to **BREP**

BREP - classification

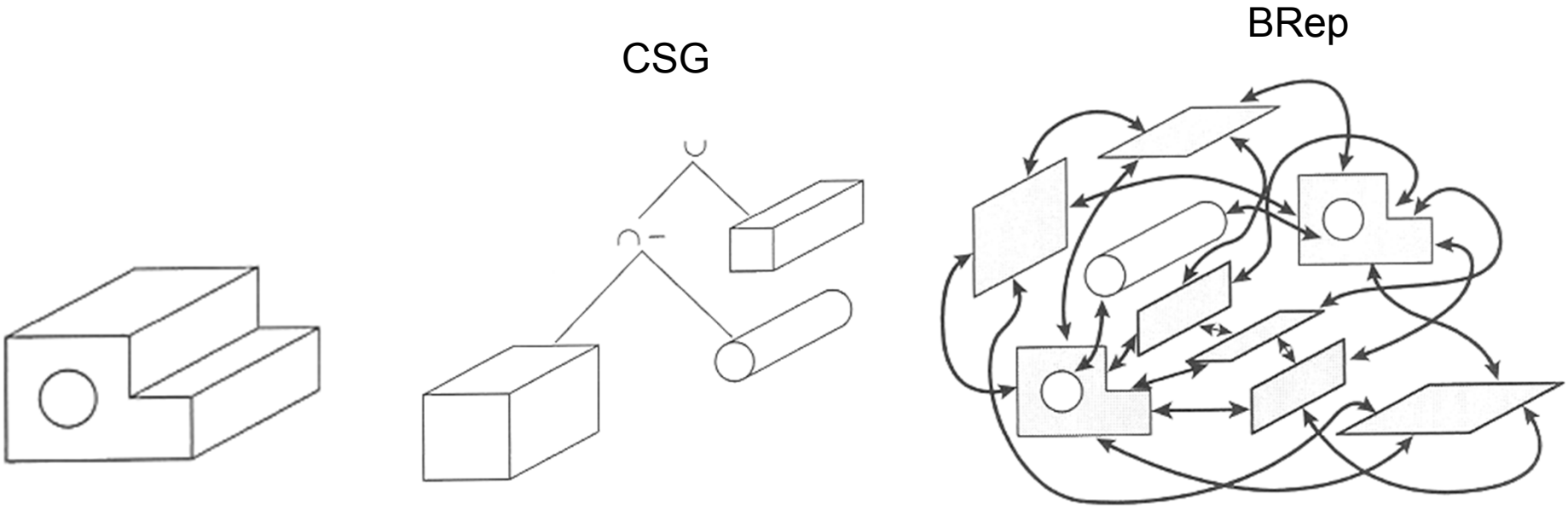
- Mesh, polygonal soup
- Winged edge data structures
- Image based shape representations

Mesh – a Set of Triangles

- An ordinary representation
- Each triangle is specified by 3 vertices
- Additionally, we can specify the normal at vertices to smooth the surface
- A compact representation:
  - vertices + normals
  - indices to 3 vertices
- Supported in many graphics file formats
- There is also need to support for texturing by texture coordinates. Example: Wavefront OBJ format.
- Sometimes called a polygonal soup – but they need not to represent a regular object.

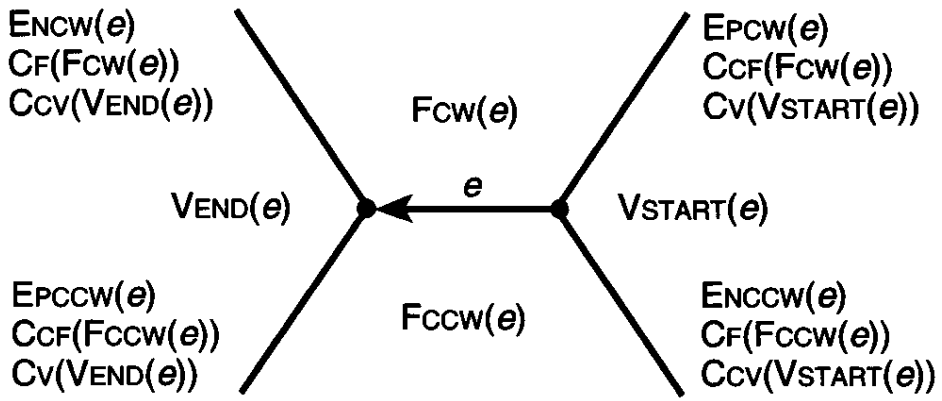
Mesh Generalization: Boundary-based Representation (BRep)

- The number of boundaries is higher than the number of basic shapes describing an object
  - A single triangle has three sides
  - A box in 3D has 6 facets
- Some facets can have complicated shape

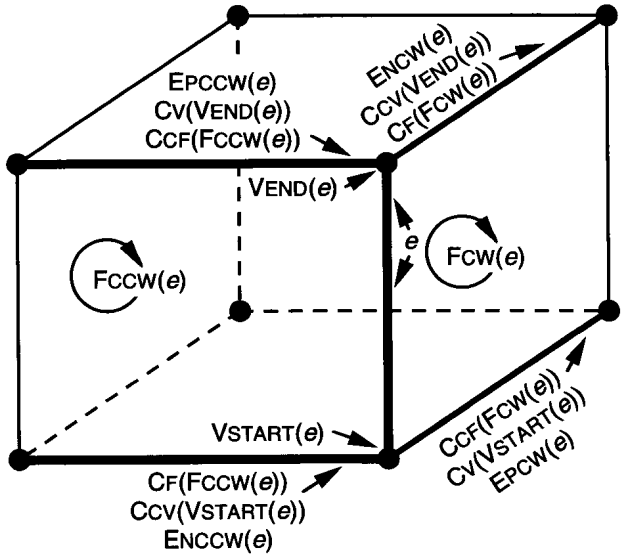


Winged-Edge Data Structure

- A collection of faces with organization for edges using oriented graph
- We need 3 tables to describe fully the shape
  - Vertex-edge table
  - Face-edge table
  - Edge-edge table
- Key element is the *winged edge*:

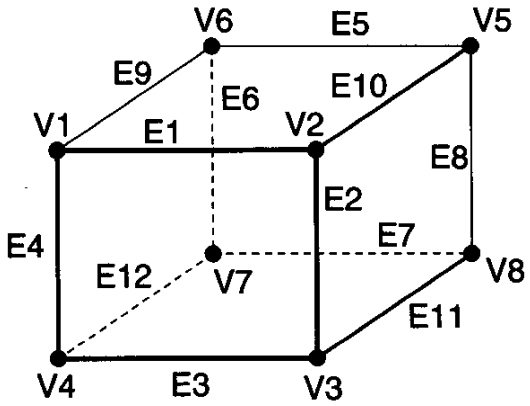


Physical Interpretation:

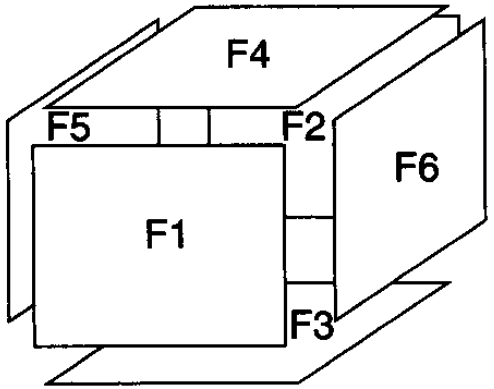


Winged-Edge Data Structure example: 3D Box

Edges



Faces



VertexEdge Table

VERTEX *v*	X	Y	Z	EDGE
V1	X1	Y1	Z1	E1
V2	X2	Y2	Z2	E2
V3	X3	Y3	Z3	E3
V4	X4	Y4	Z4	E4
V5	X5	Y5	Z5	E5
V6	X6	Y6	Z6	E6
V7	X7	Y7	Z7	E7
V8	X8	Y8	Z8	E8

FaceEdge Table

FACE *f*	EDGE
F1	E1
F2	E5
F3	E11
F4	E9
F5	E4
F6	E8

Winged-Edge Example contd.

- By three tables (VertexEdge table, FaceEdge table, Edge-Edge table) is boundary representation fully described, including connectivity
- Each row in the Edge-Edge table describes winged-edge:

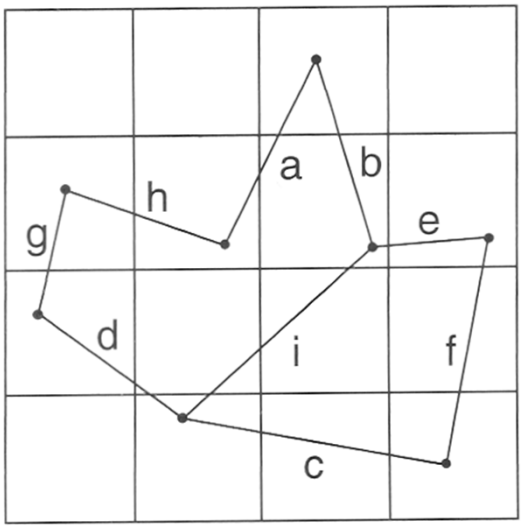
EDGE *e*	VSTART	VEND	FCW	FCCW	CCFFCW EPCW CVVSTART	CFFCW ENCW CCVVEND	CCFFCCW EPCCW CVVEND	CFFCCW ENCCW CCVVSTART
E1	V1	V2	F1	F4	**E4**	E2	**E10**	**E9**
E2	V2	V3	F1	F6	E1	E3	**E11**	**E10**
E3	V3	V4	F1	F3	E2	E4	**E12**	E11
E4	V4	V1	F1	F5	E3	E1	**E9**	E12
E5	V5	V6	F2	F4	E8	E6	**E9**	E10
E6	V6	V7	F2	F5	E5	E7	**E12**	E9
E7	V7	V8	F2	F3	E6	E8	**E11**	E12
E8	V8	V5	F2	F6	E7	E5	E10	E11
E9	V1	V6	F4	F5	E1	E5	E6	E4
E10	V5	V2	F4	F6	E5	E1	E2	E8
E11	V3	V8	F3	F6	E3	E7	E8	E2
E12	V7	V4	F3	F5	E7	E3	E4	E6

- The details of winged-edge data structures given in other courses, e.g. “Geometric Modeling” by Prof. Pavel Slavik

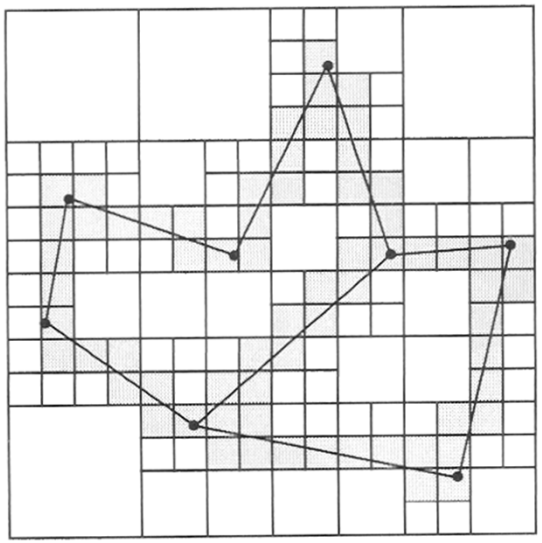
2D Representations Based on Image

- Basic idea is to represent the contour of the shape by set or sequence of line segments or arc segments (such as Bezier splines)
- The shape can be further hierarchically organized:

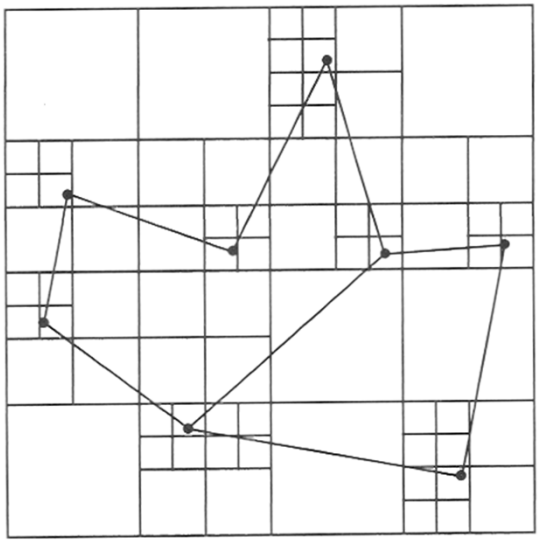
Original data



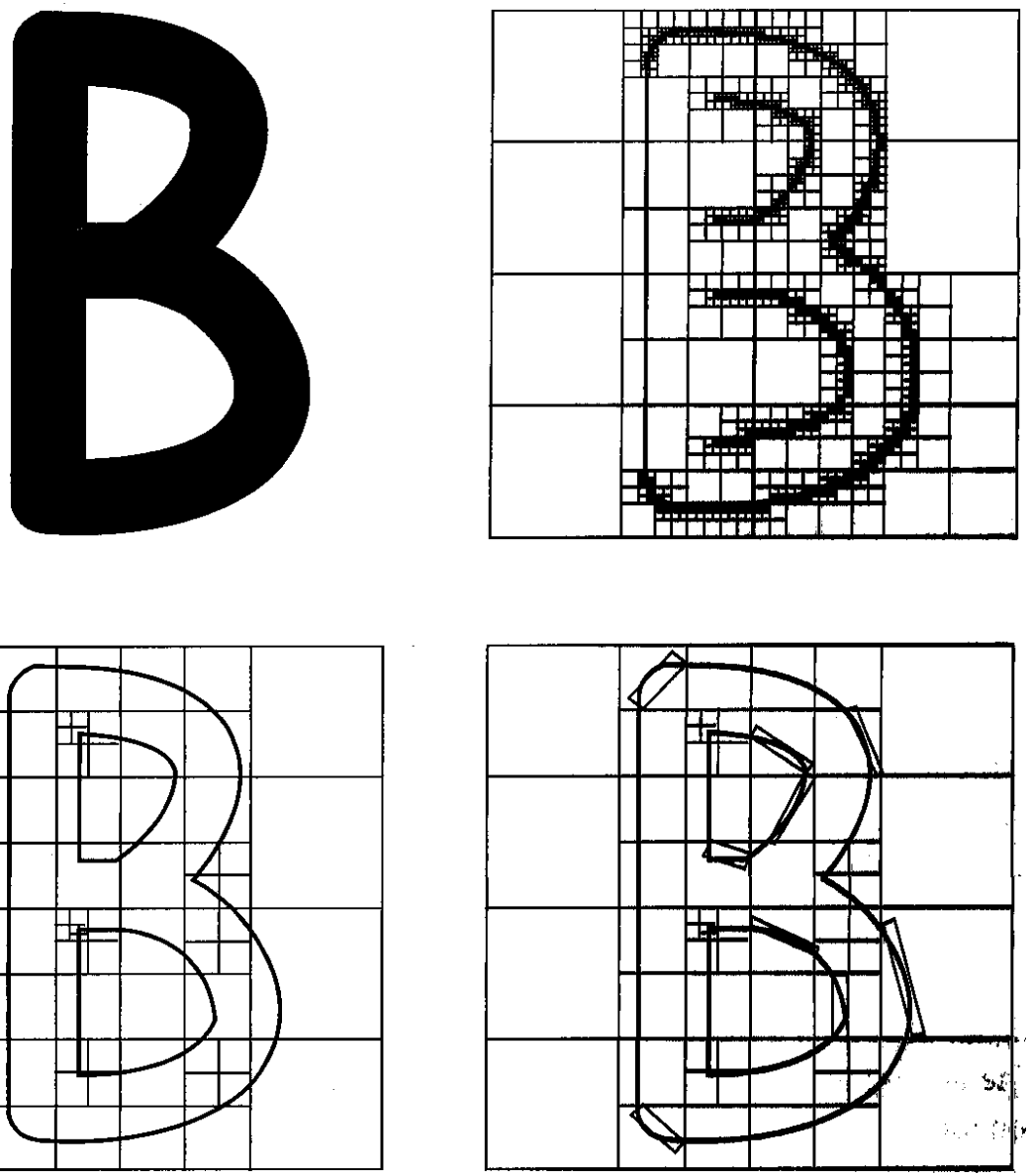
MX quadtree



Edge quadtree



Typical Usage: Fonts in Typography



Determining the Shape Interior from Boundary

- Ray casting from a point
  - Odd number of intersections - inside
  - Even number intersections – outside
- Robustness issue – edges, vertices etc.

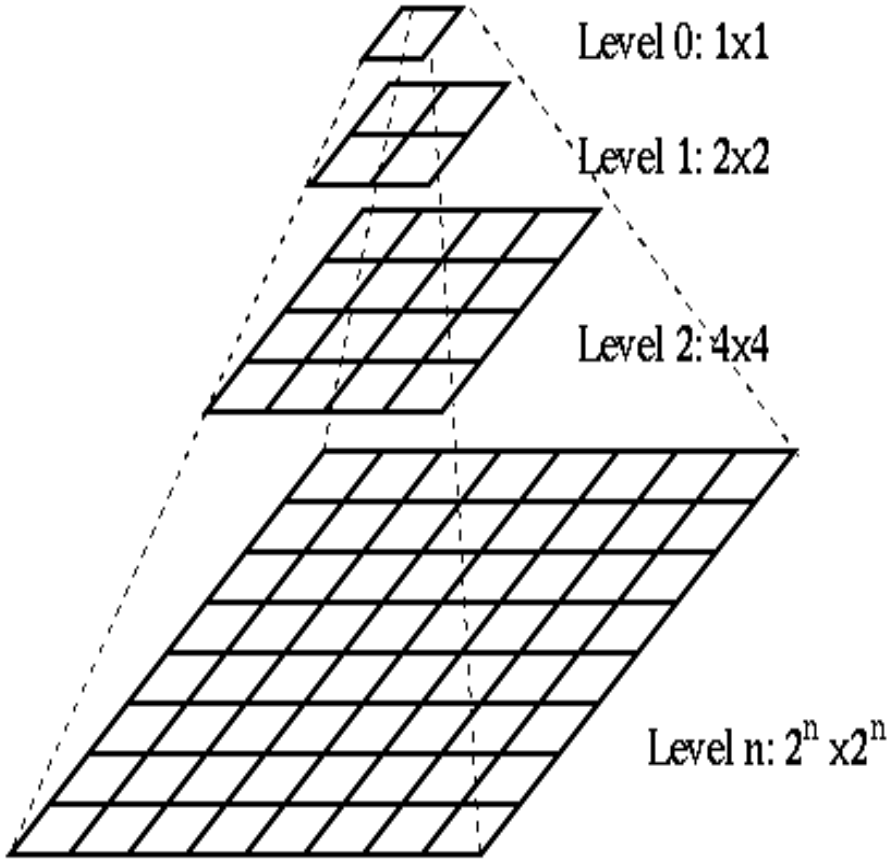
SPECIAL REPRESENTATIONS

Other (Special) Object-Based and Image-Based Representations - Classification

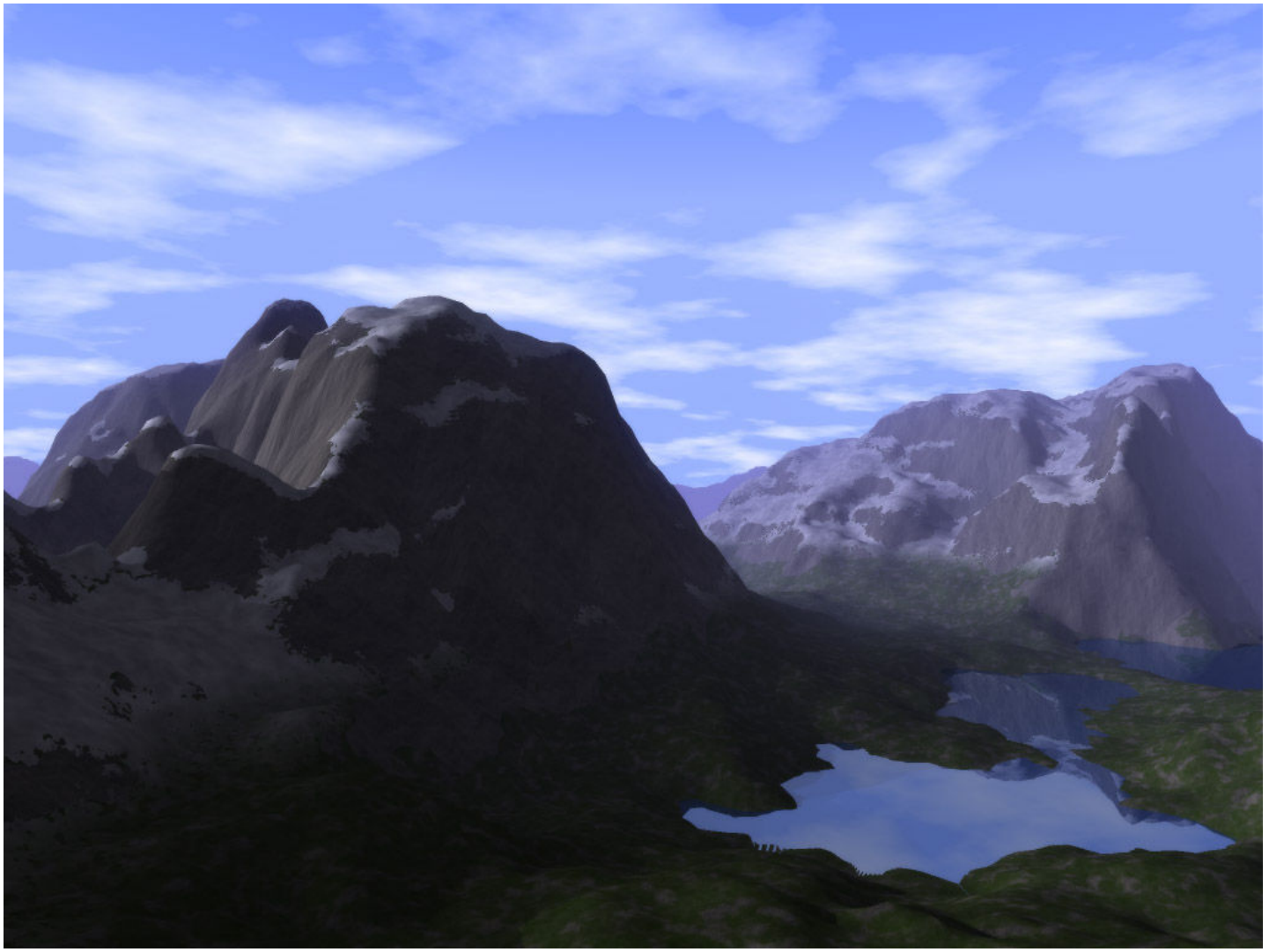
- 2.5D representations (image pyramid and heightfields)
- Point-based surface representations from scattered data measurements
- Density based representations

Image Pyramid

- Image pyramid = averaging 2^n *x* 2^n pixels at n-leves starting from single pixels
- Principle: average 4 pixels to a single pixel
- Usage in level-of-detail (LOD), textures, etc.
- Initial image resolution has to be 2^n *x* 2^n pixels

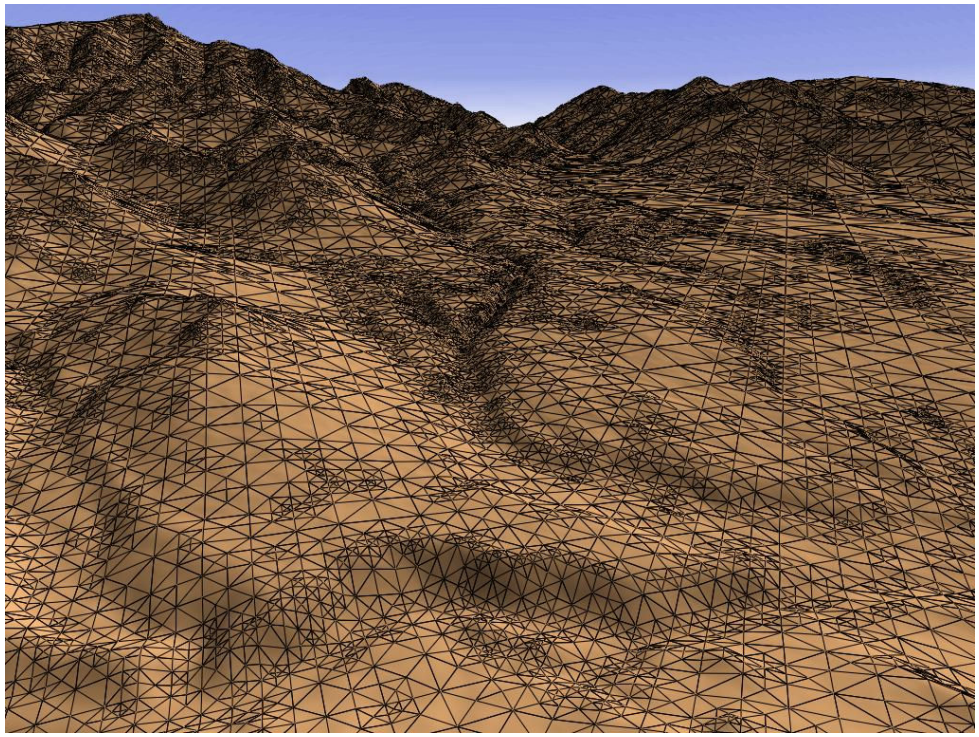


Height field example



Height-fields = uniform grid *x* height

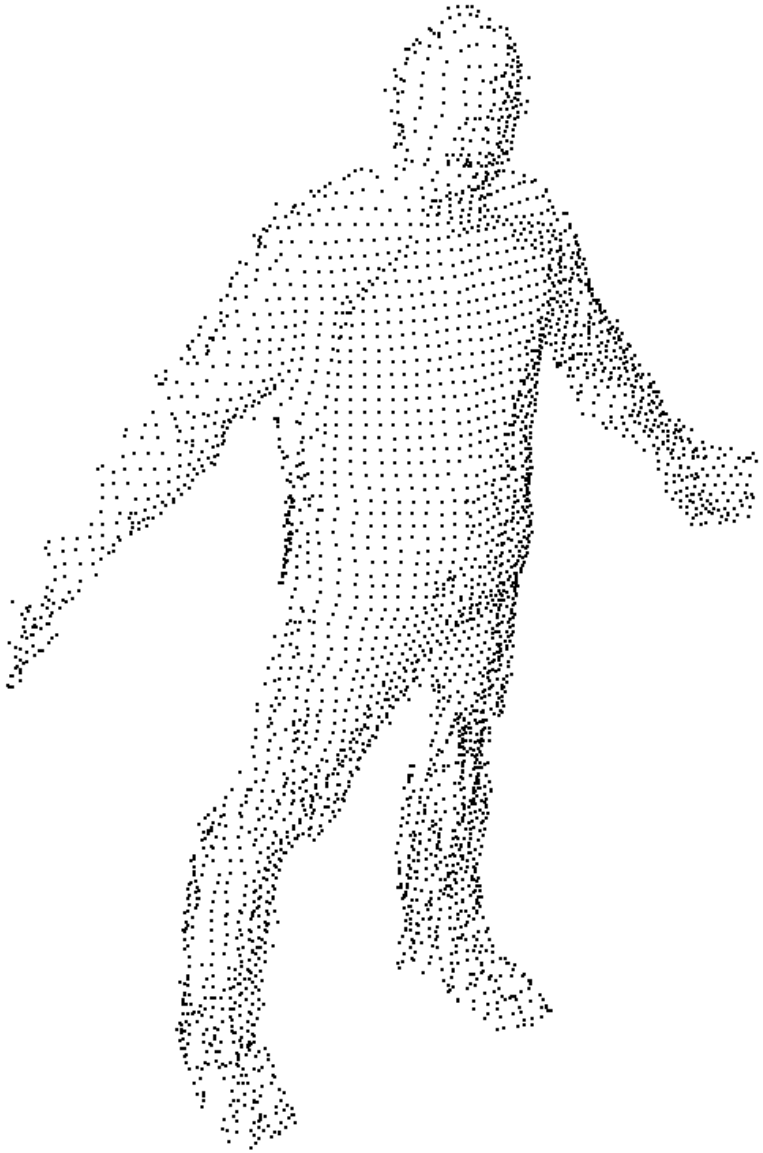
- Application: terrain modeling and rendering
- At each point of a 2D grid we specify height

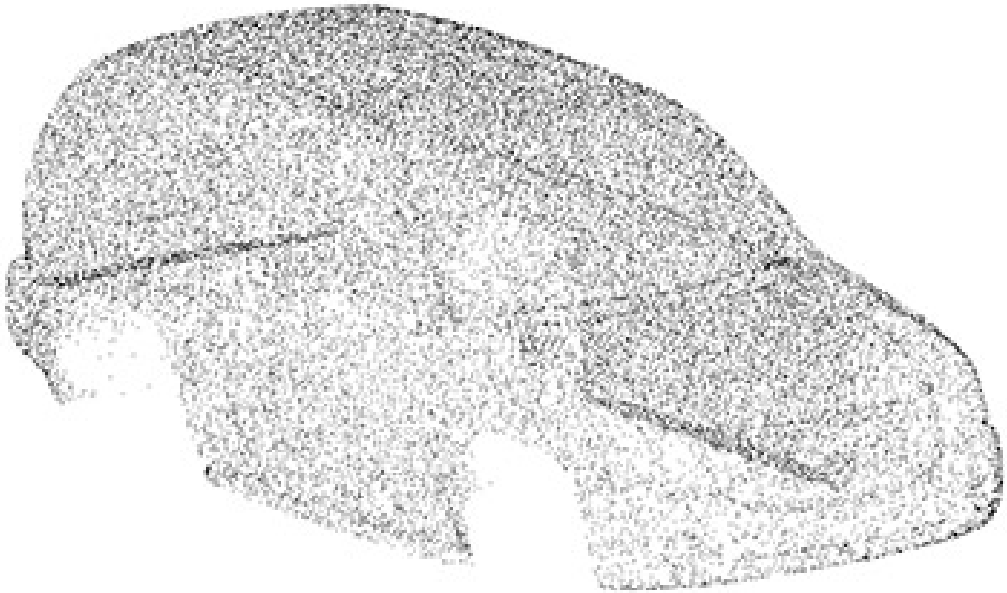


Point-based Surface Representations

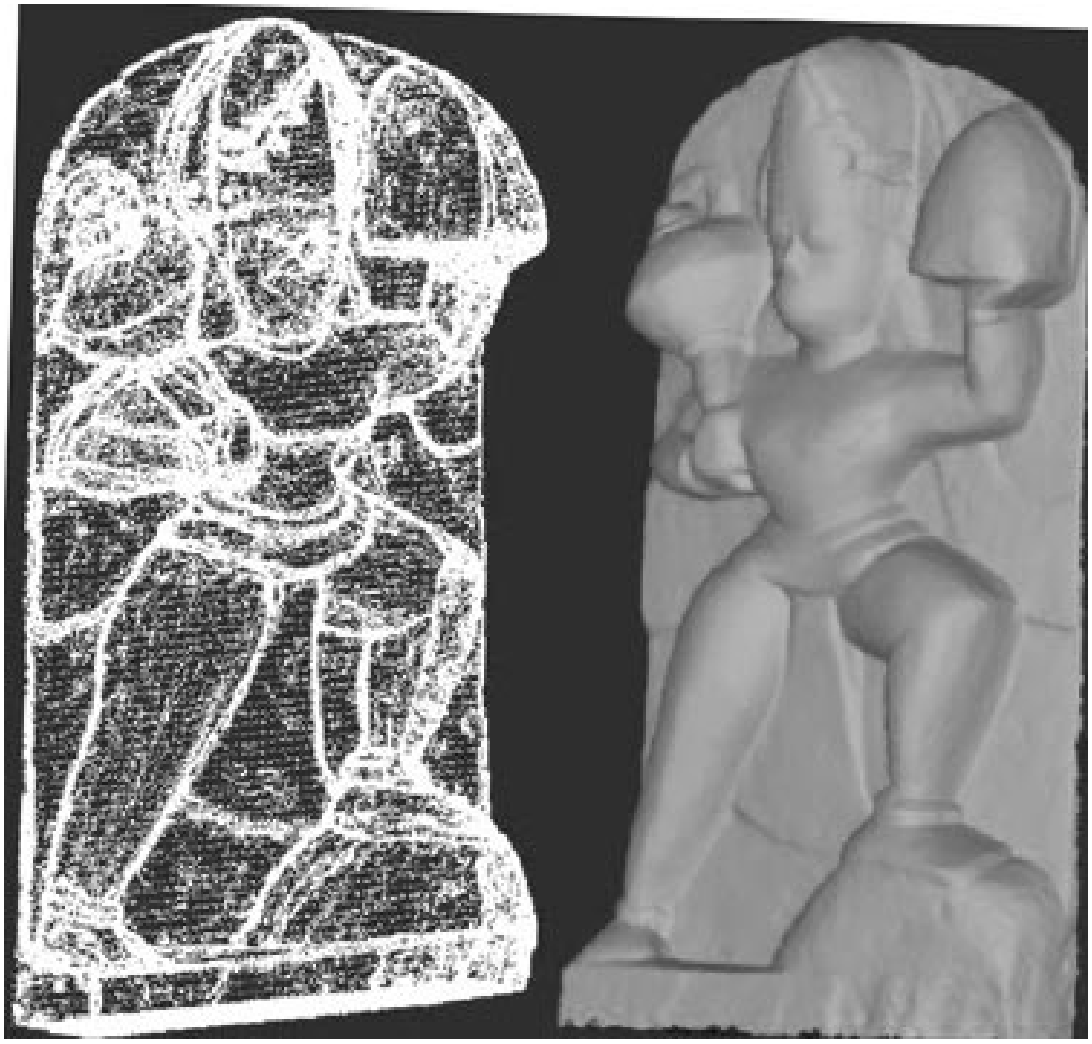
- Typically, we measure the real objects by a 3D scanner – output is a set of points called a *point cloud*
- We can convert a point cloud to another representation, many methods exist
  - Accuracy
  - Preserving features
  - Speed
- A suitable conversion method is based on nearest-neighbor search (further lectures)

Point cloud examples

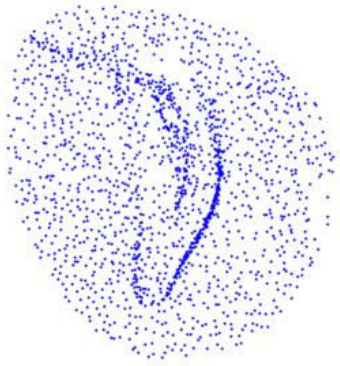




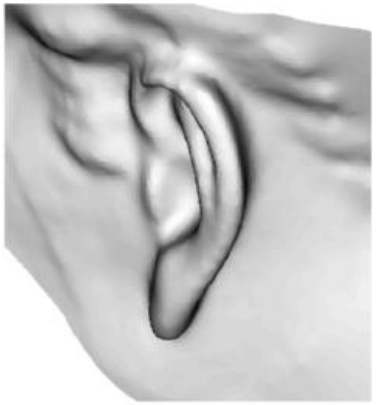
Application: Cultural Heritage Preservation



Other Point Cloud Examples



Laser scanned point cloud of an ear



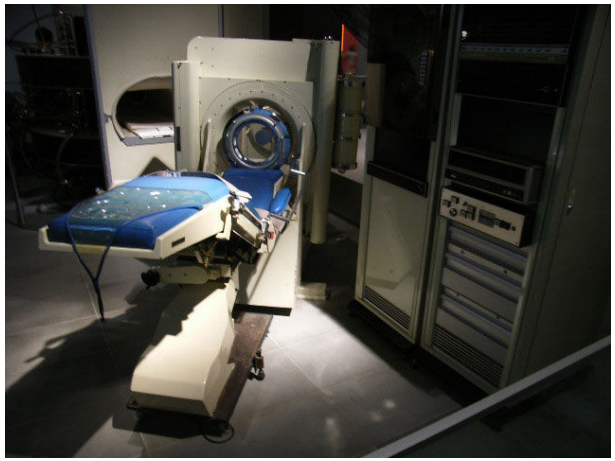
Fitted RBF surface

Laser scanned point cloud
from a hand (27,000 points)

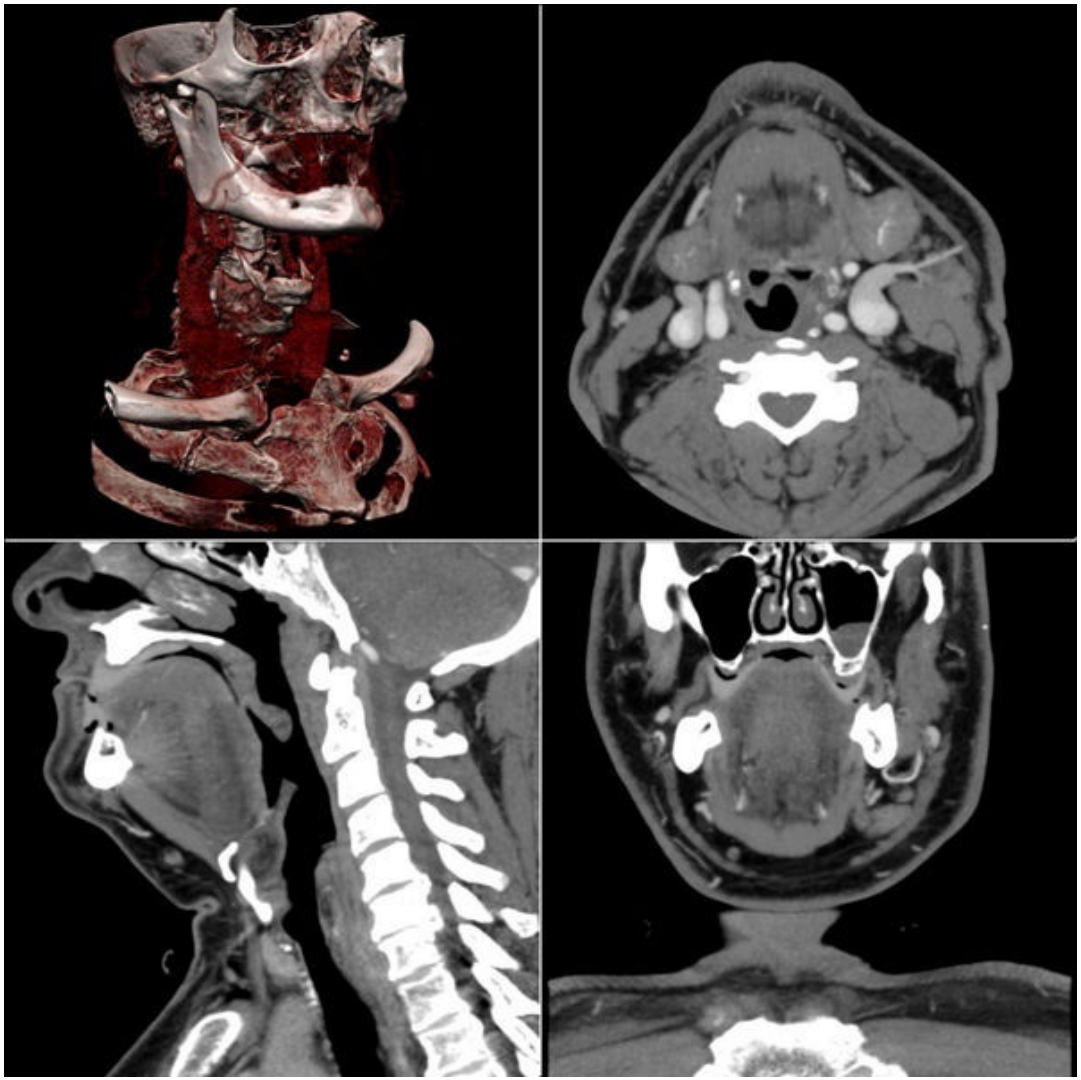
A closed RBF surface
consisting of 2,600 vertices

Density Based Representations

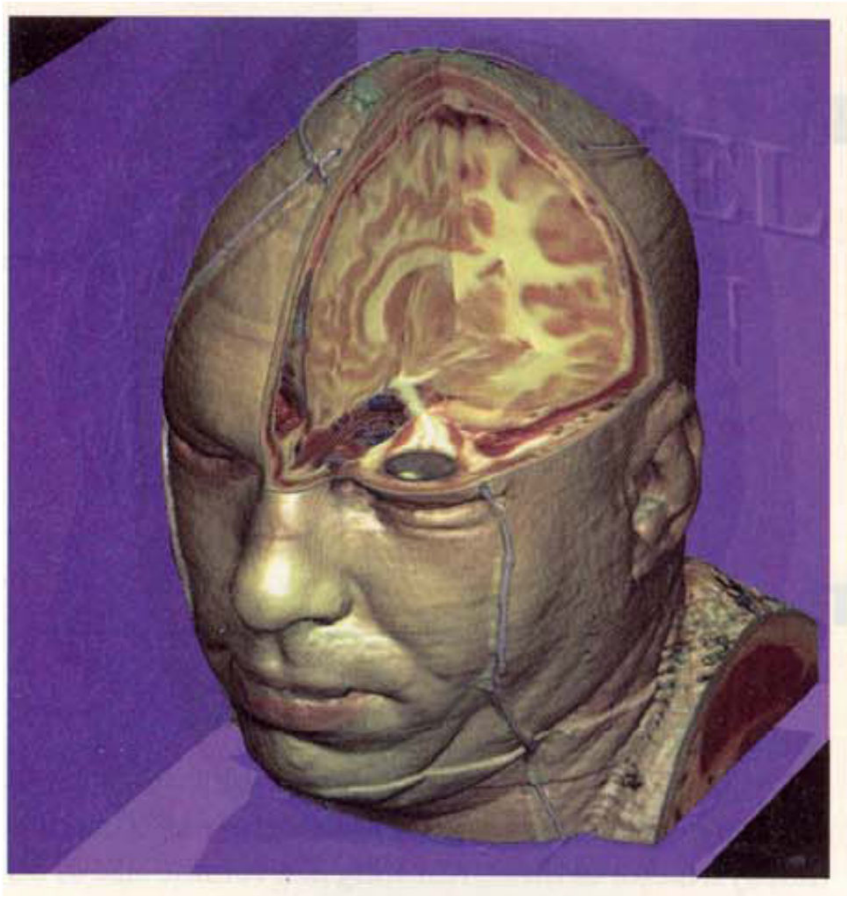
- 2D or 3D grid + specification of density
- An object is specified by a segmentation process based on density threshold
- Typically used in medical imaging and product inspection
- Source data acquired by computer tomography (CT) and magnetic resonance tomography (MRT)

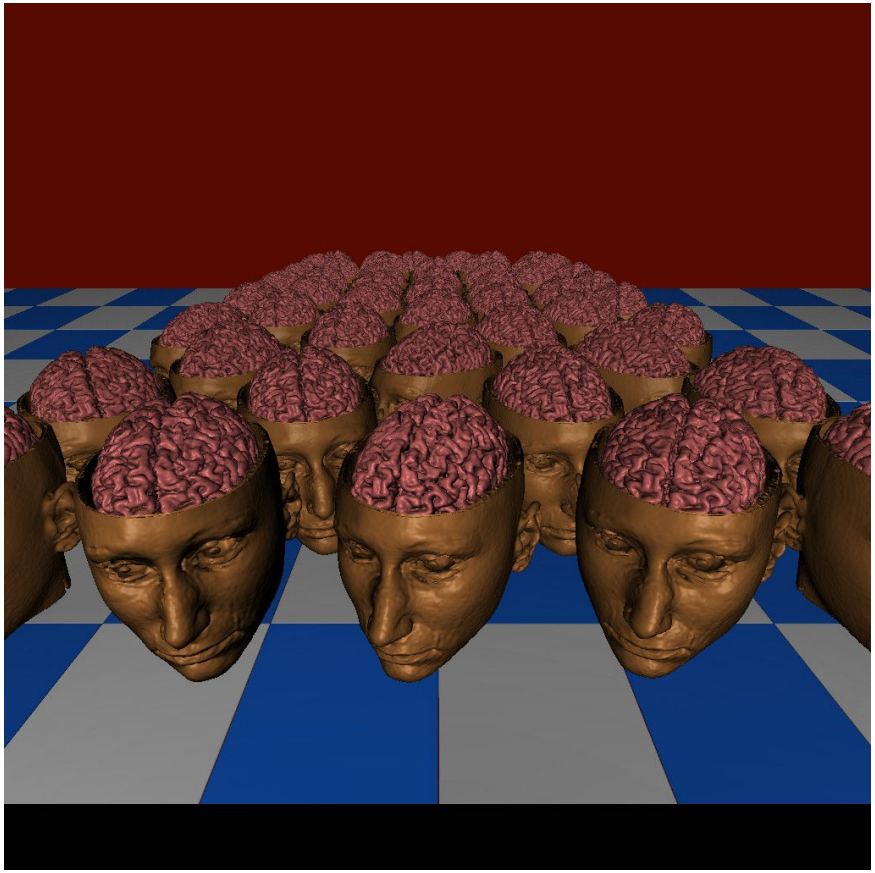


Example CT Visualizations



Some Colored Examples





Typical Problems with Object Representations

- Object representation conversion:
  - From boundary-based to interior-based representation (*rasterization*) and other way round
  - How to preserve accuracy while keeping low memory requirements for conversion back and forth
  - How to achieve the smoothness of boundary data
  - How to preserve sharp features in boundary data
- Speed
- Low memory requirements

Thank you for your attention!