Solid Modeling and Procedural Geometry

David Levin

University of Toronto

But First !

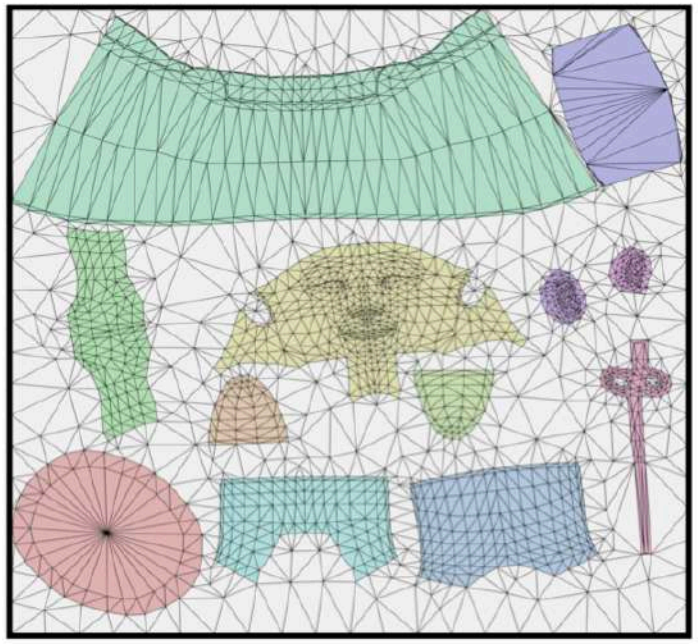
- Questions about the course ?

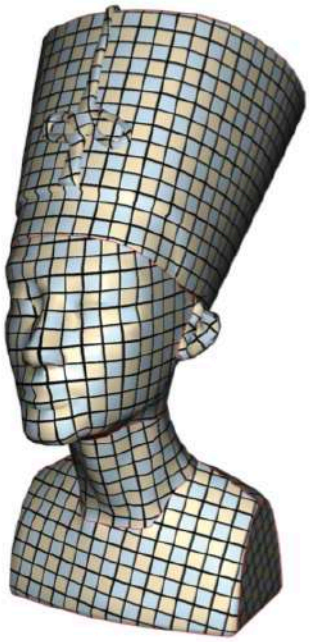
But First !

- 30 minute training session on 3D printers

Final Project Pitches !!!

Laser Cut Any 3D Model

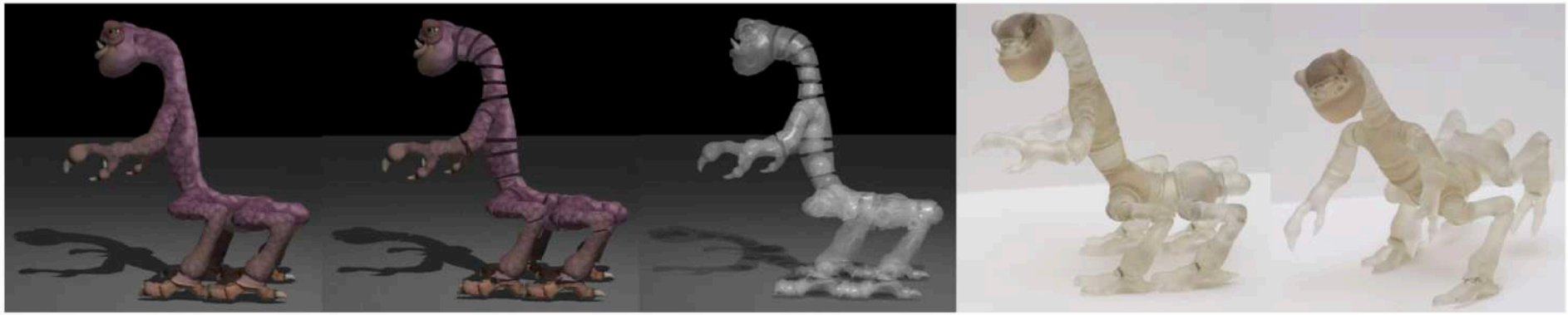




Model to Laser Cut and Laser Cutter Bed

Result

3D Printing Jointed Objects



Fabricating Articulated Characters from Skinned Meshes

Moritz Bächer
Harvard University

Bernd Bickel
TU Berlin

Doug L. James
Cornell University

Hanspeter Pfister
Harvard University

Sketch-Based VR Solid Modelling

Sketching in 3D

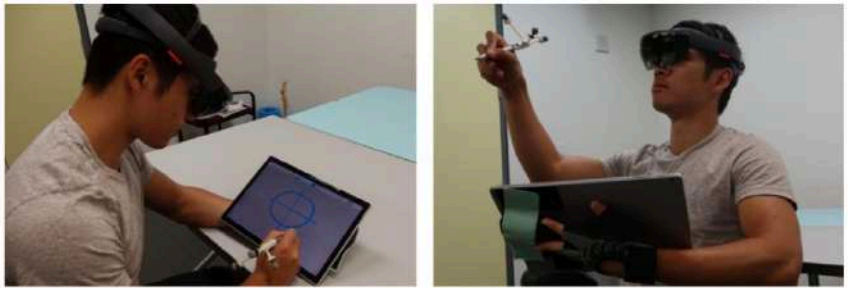
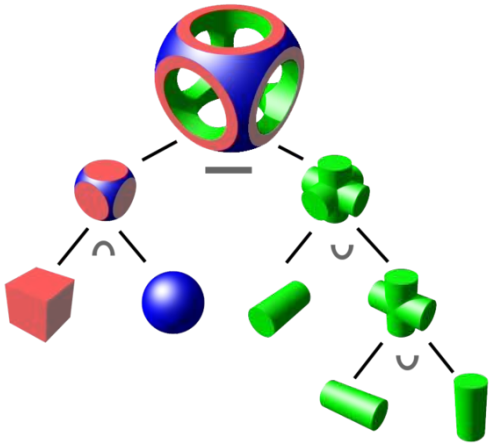


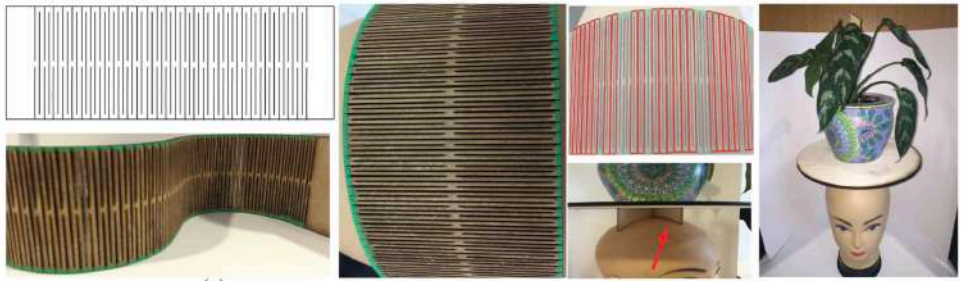
Figure 6. Setup: the user puts on the HoloLens and draws with a motion-tracked stylus, on a tablet (left), or mid-air (right) using a mouse affixed to the back of the tablet.

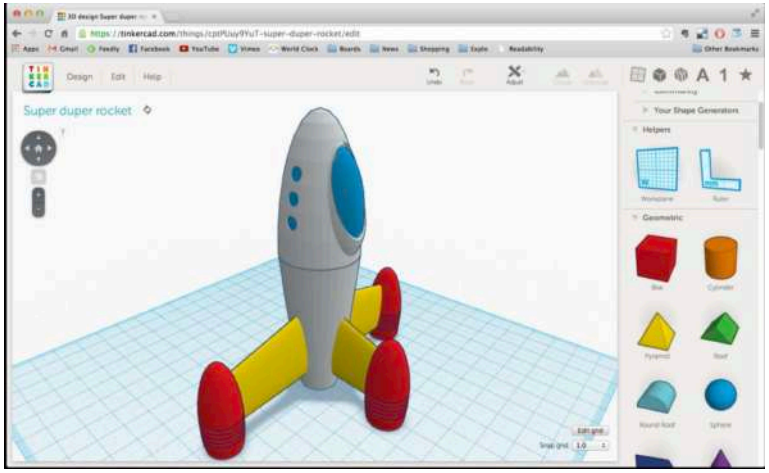


Bend-a-rule: a fabrication-based workflow for 3D planar contour acquisition

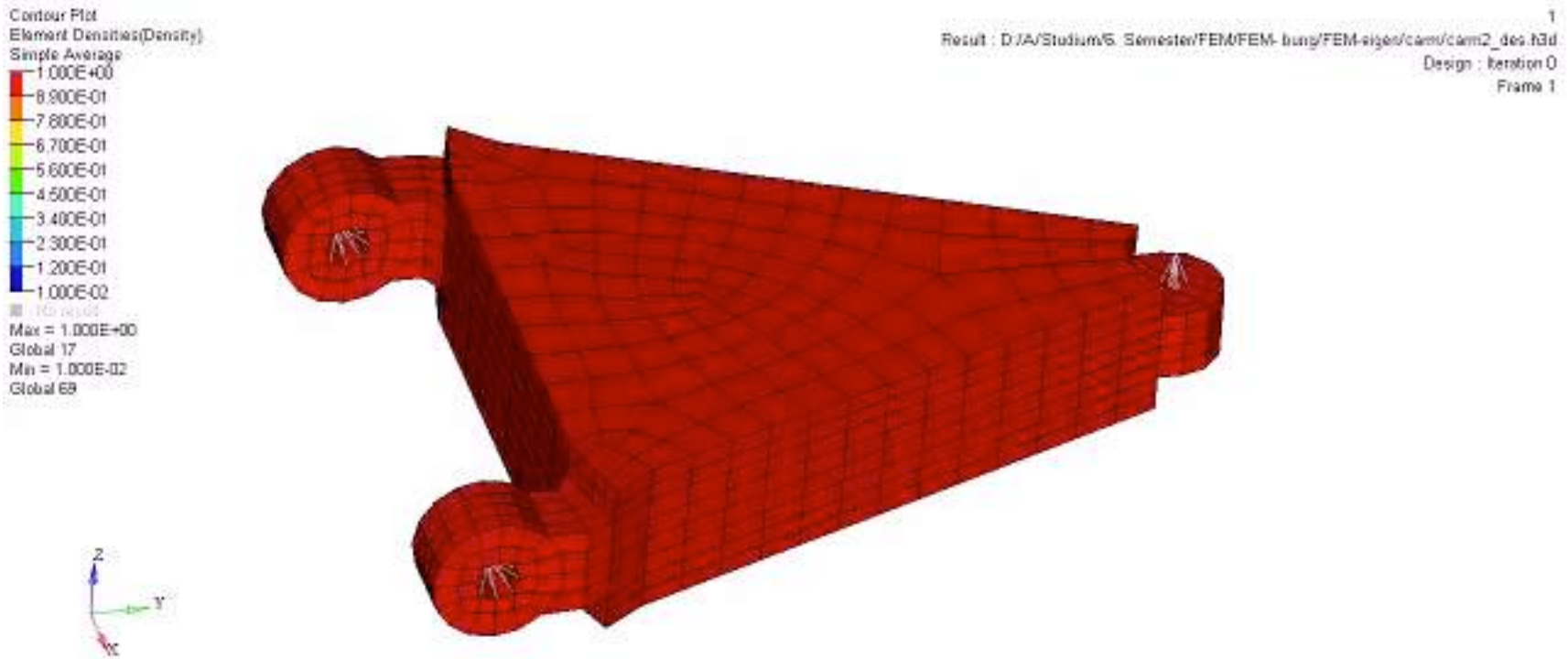
Mian Wei
Computer Science, University of Toronto
mianwei@dgp.toronto.edu

Karan Singh
Computer Science, University of Toronto
karan@dgp.toronto.edu





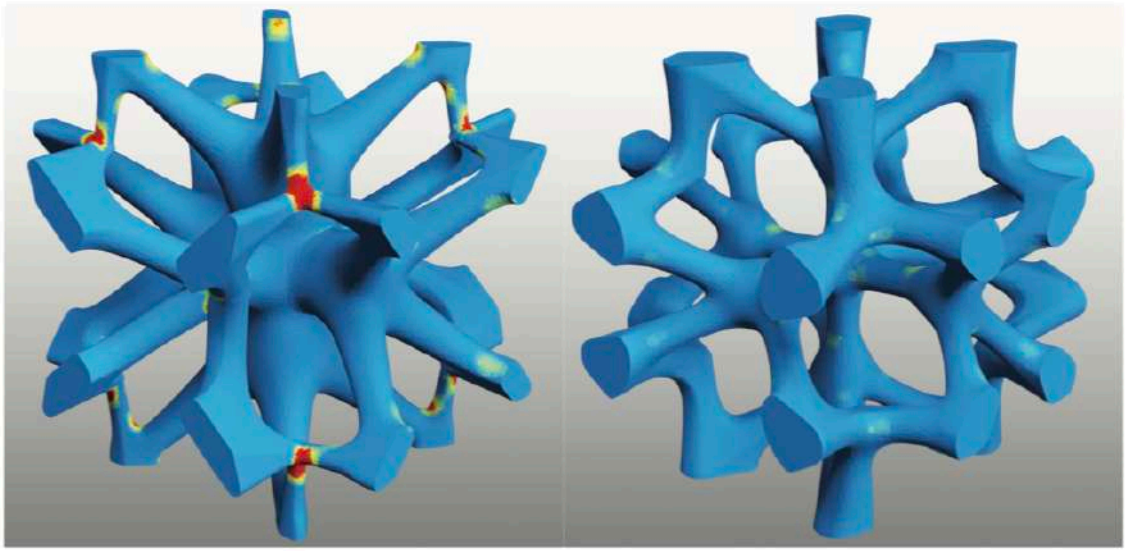
Lattice-Controlled Topology Optimization



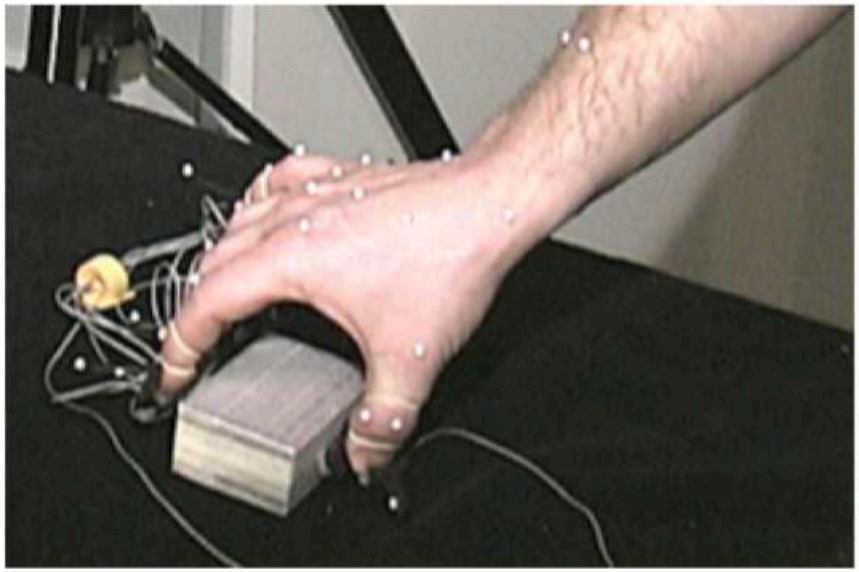
Lattice-Controlled Topology Optimization

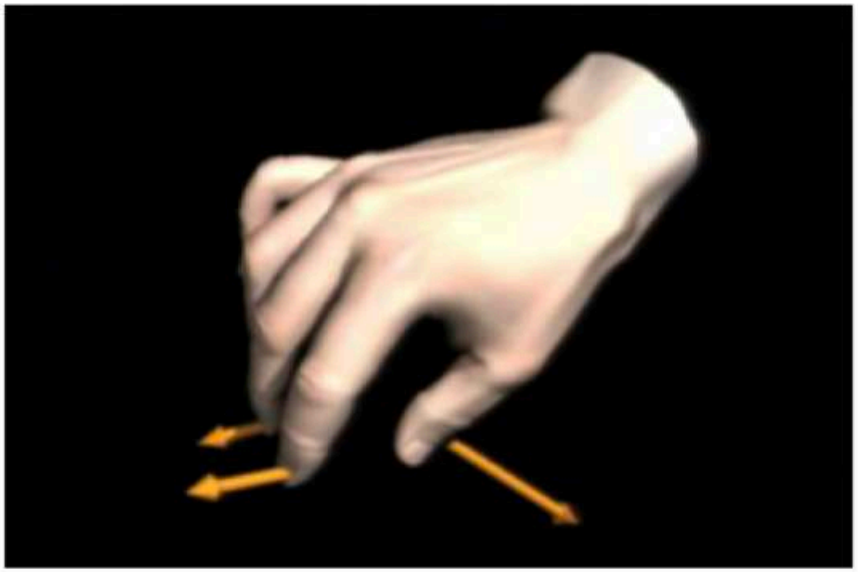
Start Here

Grow Material from Curves



A Wearable Device for Materials Measurement





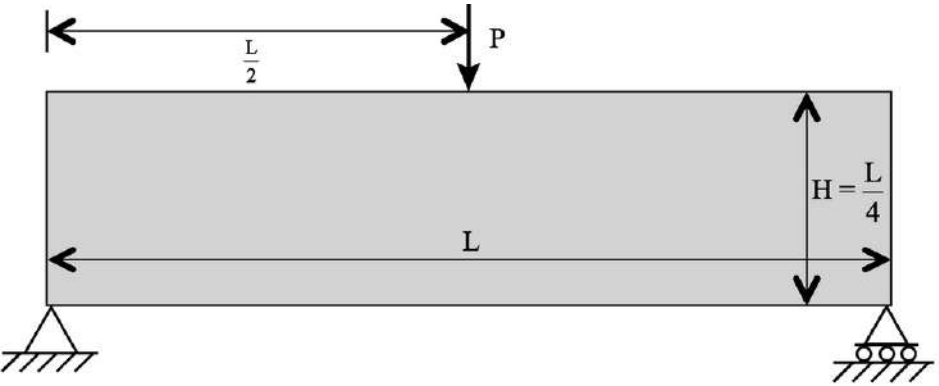
Announcements: CSC2521H F LEC0101
20189:Topics in Computer Graphics

on Capture and Synthesis

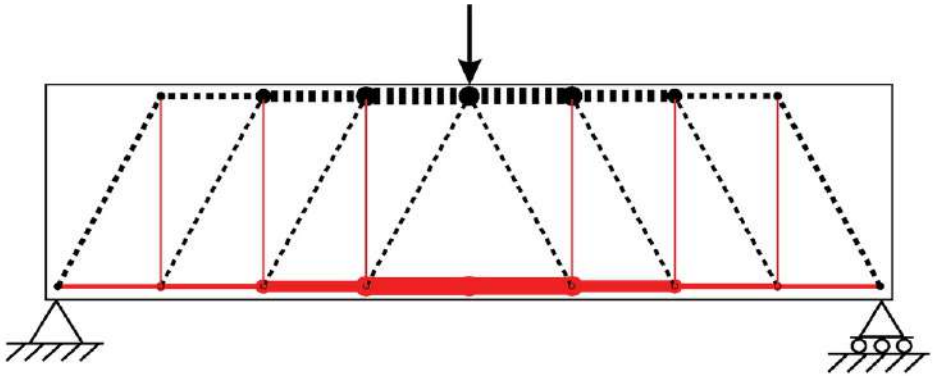
Paul G. Kry[123] Dinesh K. Pai[12]

[1]University of British Columbia [2]Rutgers University [3]EVASION / INRIA

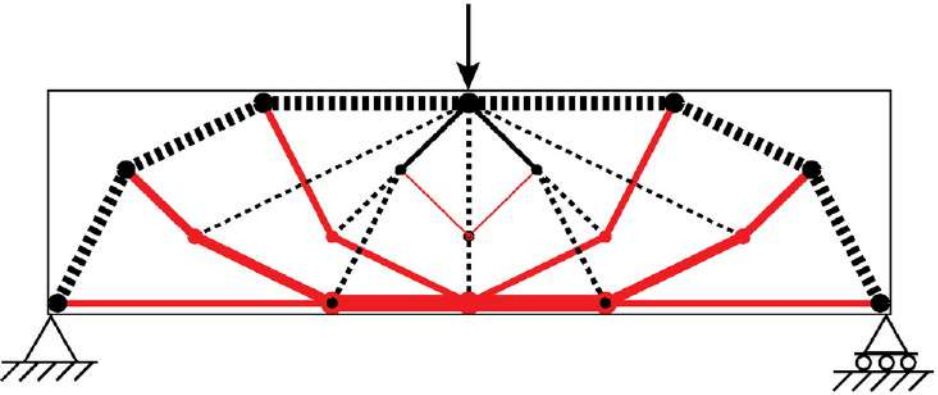
Machine Learning for Truss Optimization



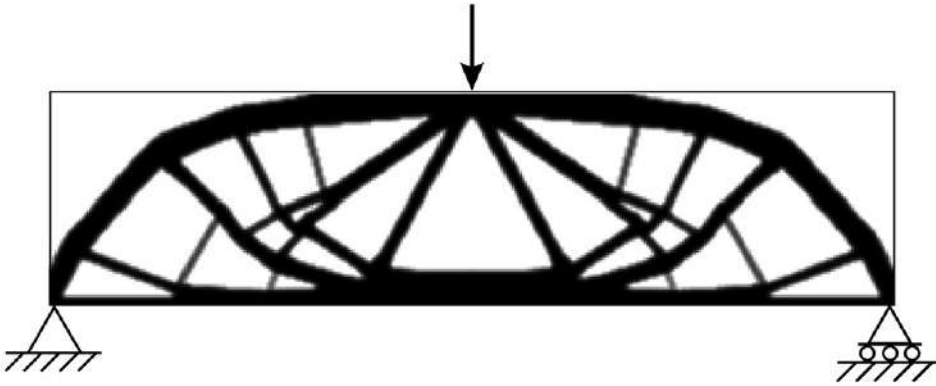
(a) Design domain



(b) Traditional strut-and-tie model



(c) Optimized truss model



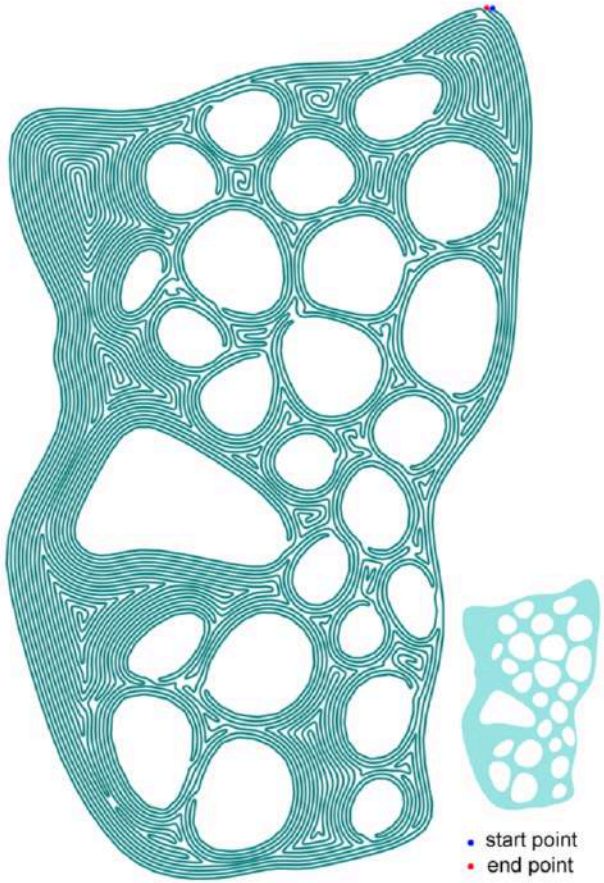
(d) Optimized continuum model

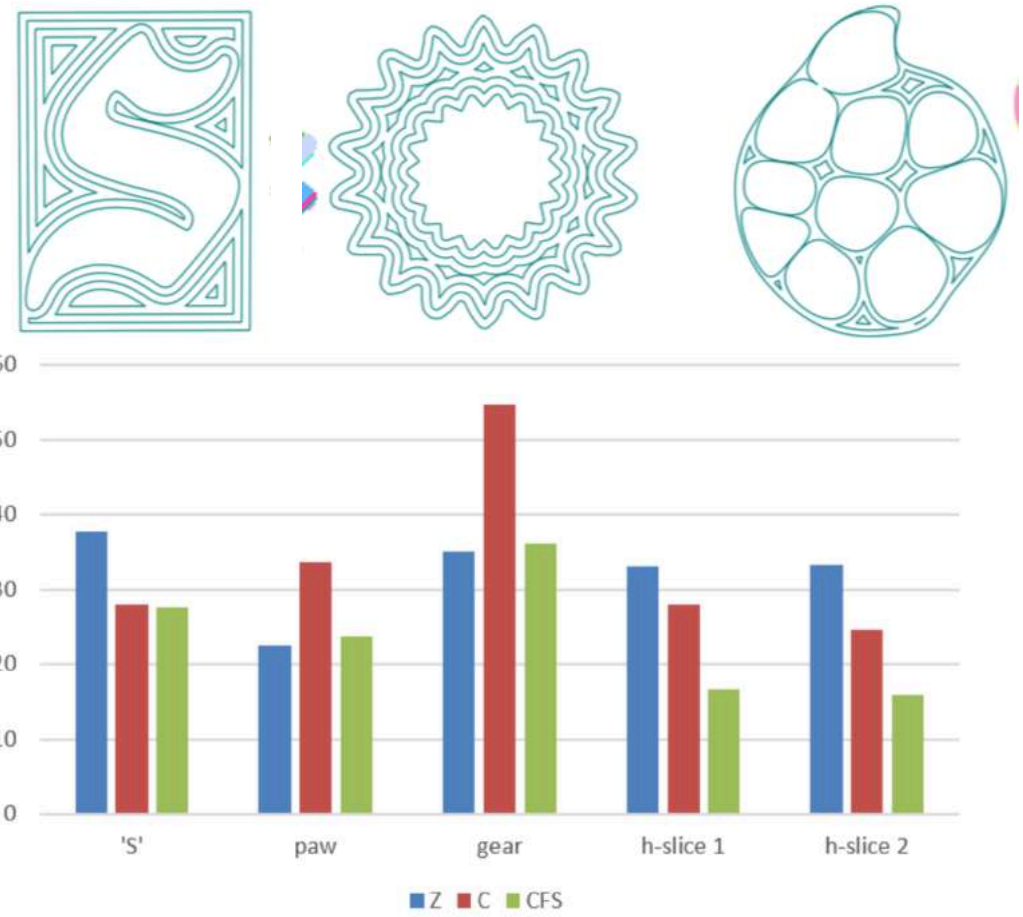
Leftovers from Last Time

- Some cool printer hardware-centric research

Better Paths for Extruders

- Better Printing Motions





Zhao et al. Connected Fermat Spirals for Layered Fabrication. SIGGRAPH 2016

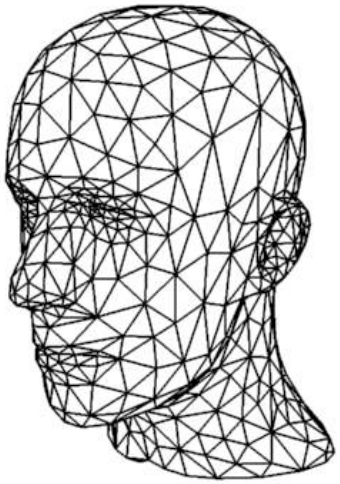
Now onto Solid Modelling!

- Unless there are any more questions

Boundary Representations (B-Reps)

- Only boundary of an object is specified
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit
- This is what we teach in Computer Graphics

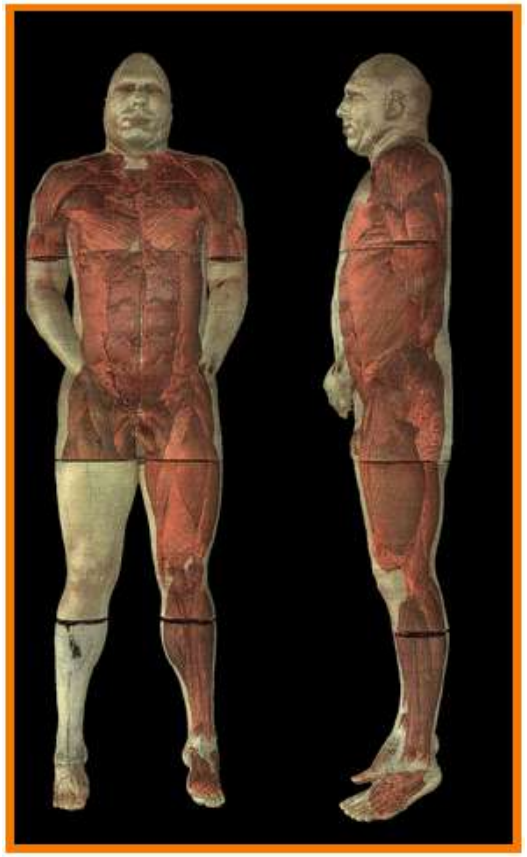
Martin Newell



Leif Kobbelt

Solid Modeling

- Represent solid interiors of objects

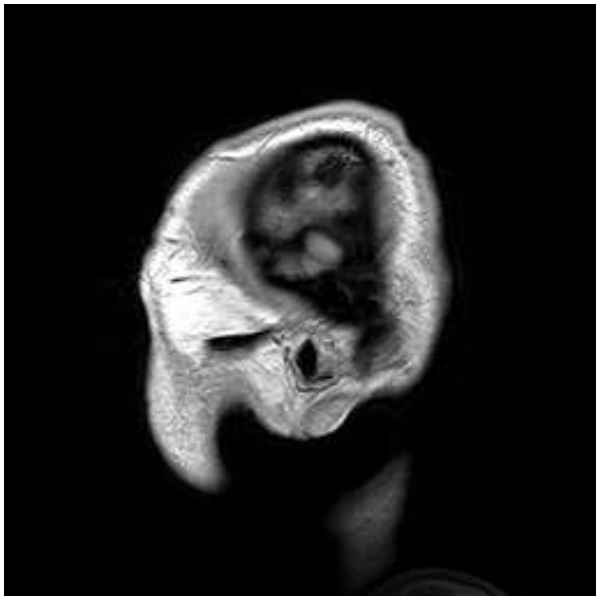


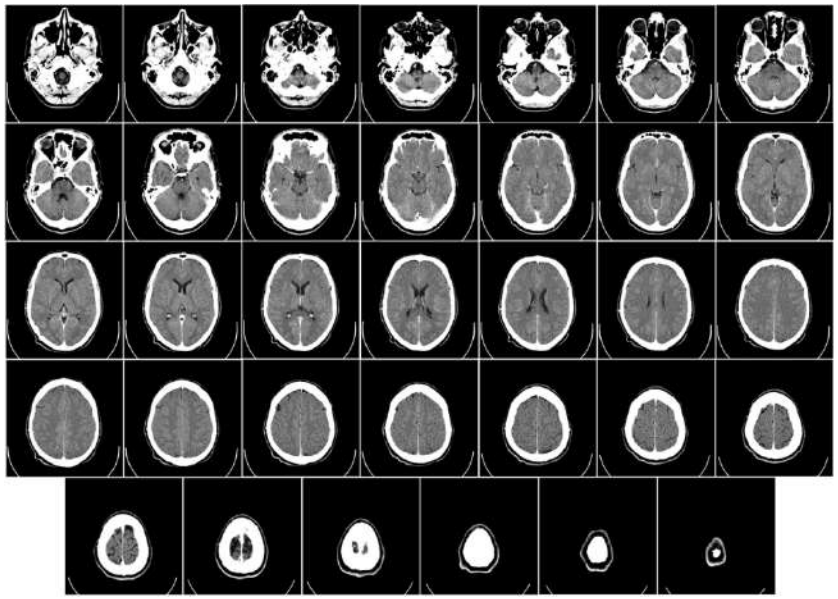
Visible Human
(National Library of Medicine)

SUNY Stony Brook

Why Volumetric Representations?

- Some acquisition methods generate solids
  - Magnetic Resonance Imaging (MRI)
  - Computed Tomography (CT/ CAT)



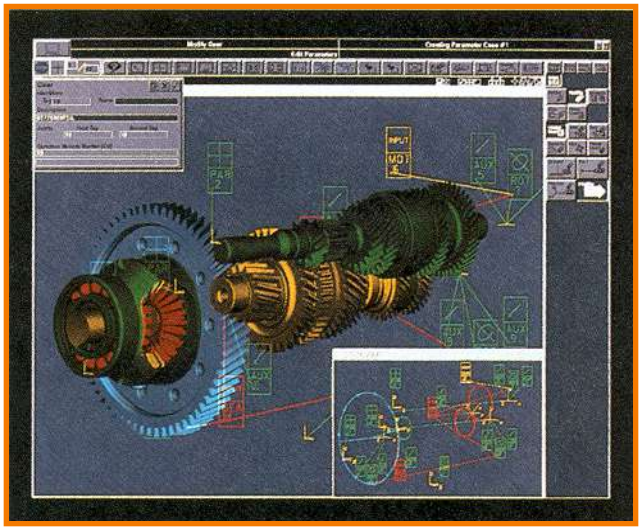


Source: Wikipedia

Why Volumetric Representations?

- Some applications require solids
  - CAD/CAM
  - material(s) need to be specified inside the object

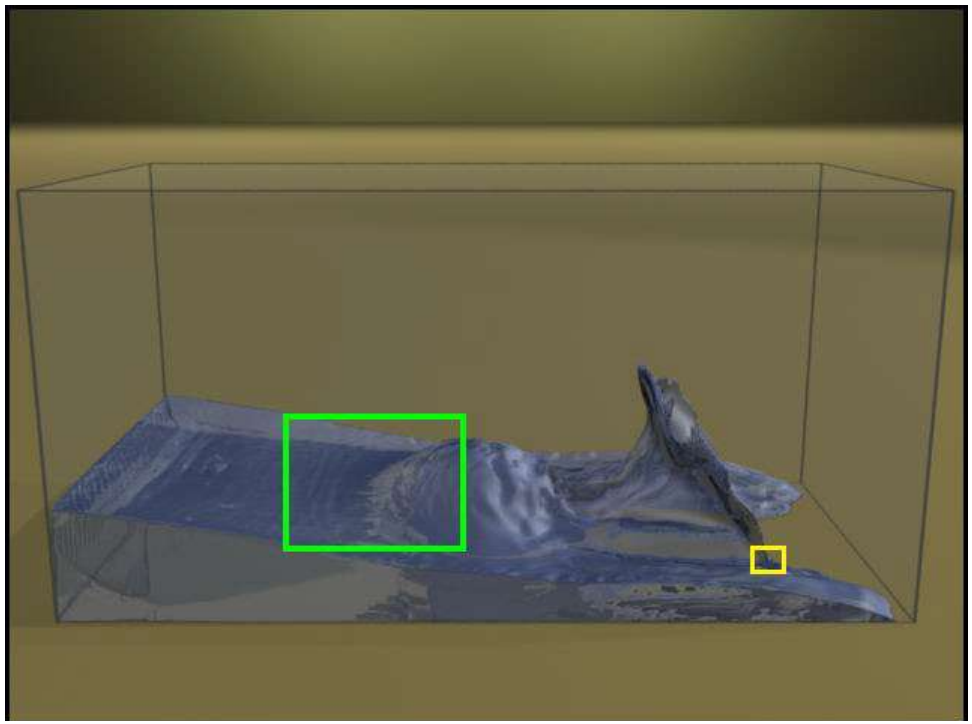
Multi-material 3D Printing

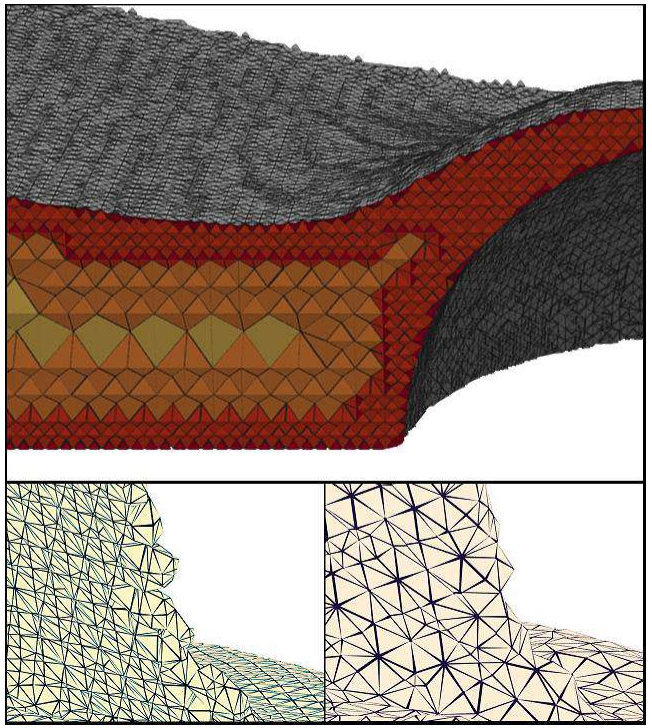


Intergraph Corporation

Why Volumetric Representations?

- Some algorithms require solids
  - Physically-based simulation

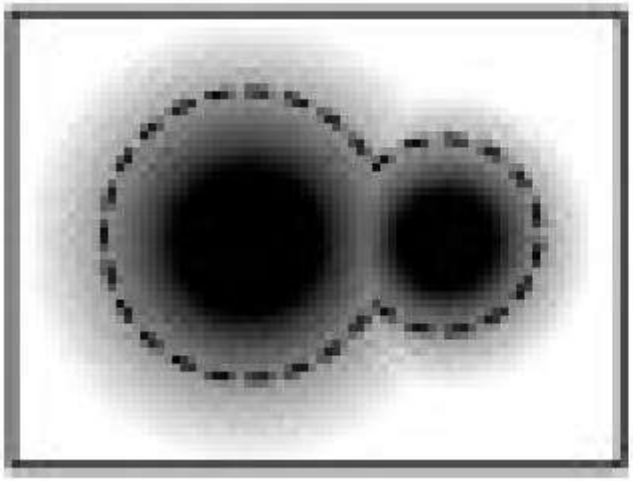




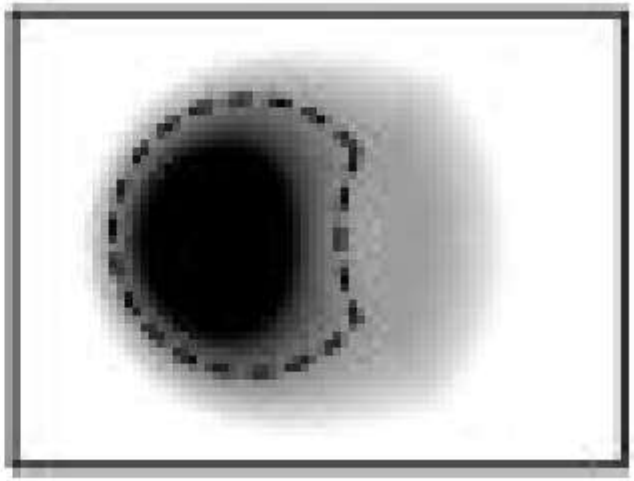
Source: Chentanez

Why Volumetric Representations?

- Some operations are easier with solids
  - Example: union, difference, intersection



Union



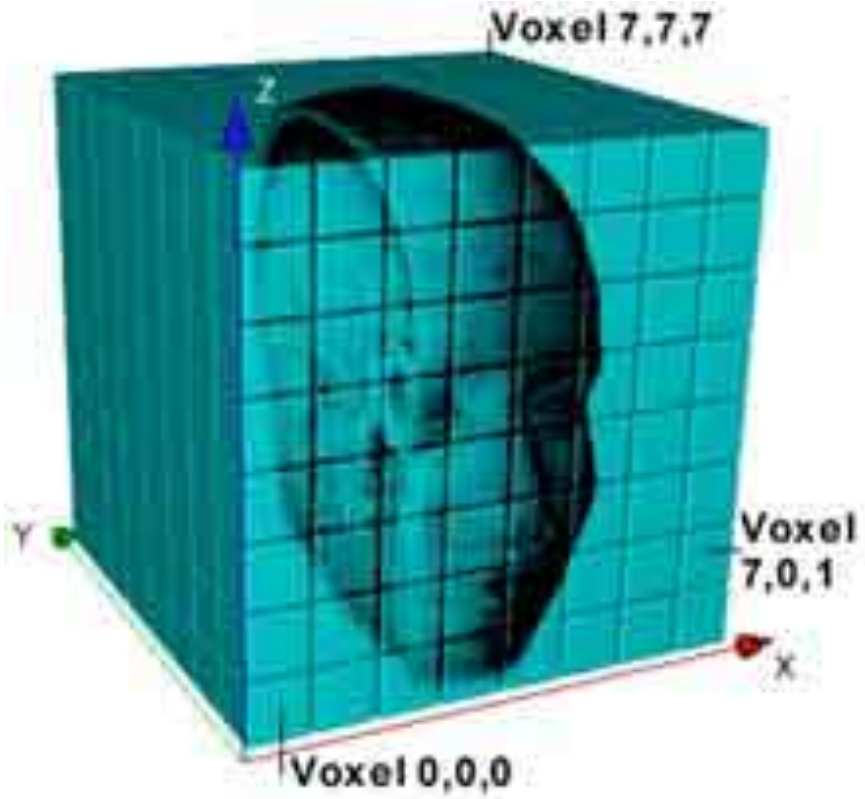
Difference

The Plan For Today

- Discrete Volume Representations
  - Voxels
    - Voxelization (from surfaces to voxels)
    - Marching Cubes (from voxels to surfaces)
  - Octrees
  - Tetrahedra
- Implicit Representations
  - Distance Fields

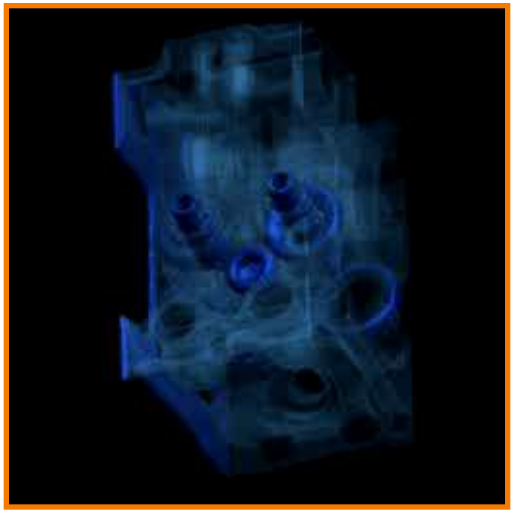
Voxels (Volume Elements)

- Partition space into a uniform grid
  - Grid cells are called voxels (like pixels)



Voxels

- Store properties of solid object with each voxel
  - Occupancy
  - Color
  - Density
  - Temperature
  - etc.

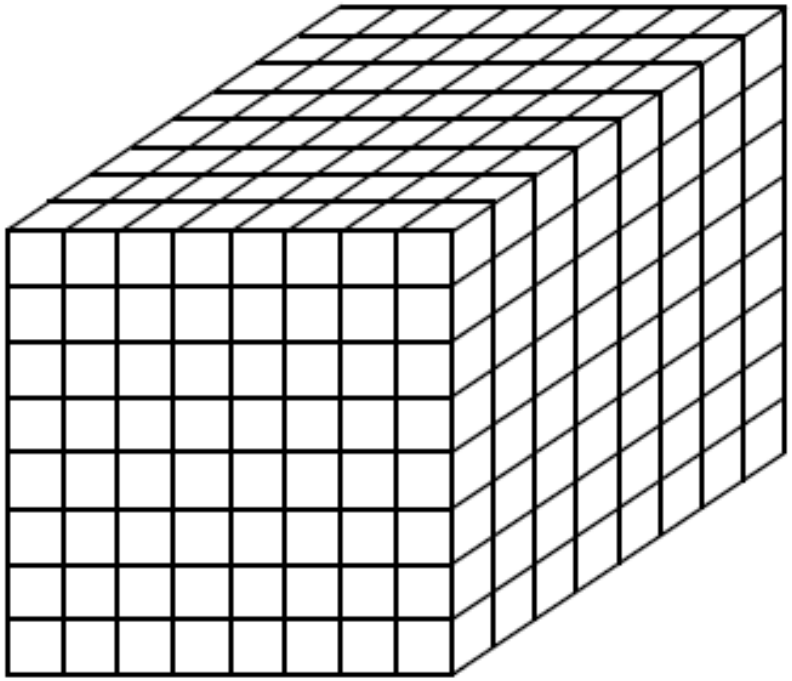


Engine Block
Stanford University

Visible Human
(National Library of Medicine)

Voxel Storage

- $O(n^3)$ storage for n x n x n grid
  - 1 billion voxels for 1000 x 1000 x 1000

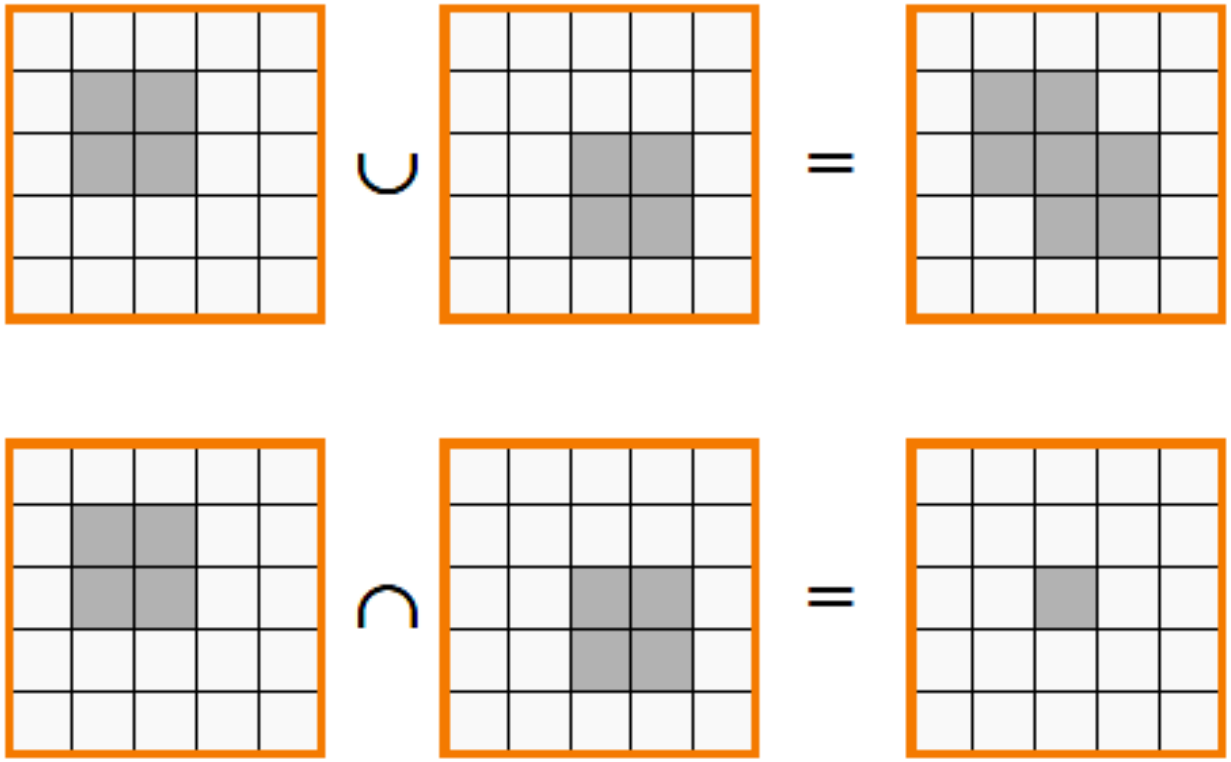


Voxel Processing

- Signal processing (just like images)
  - Reconstruction
  - Resampling
- Typical operations
  - Blur
  - Edge detection
  - Warp
  - etc.
- Often fully analogous to image processing

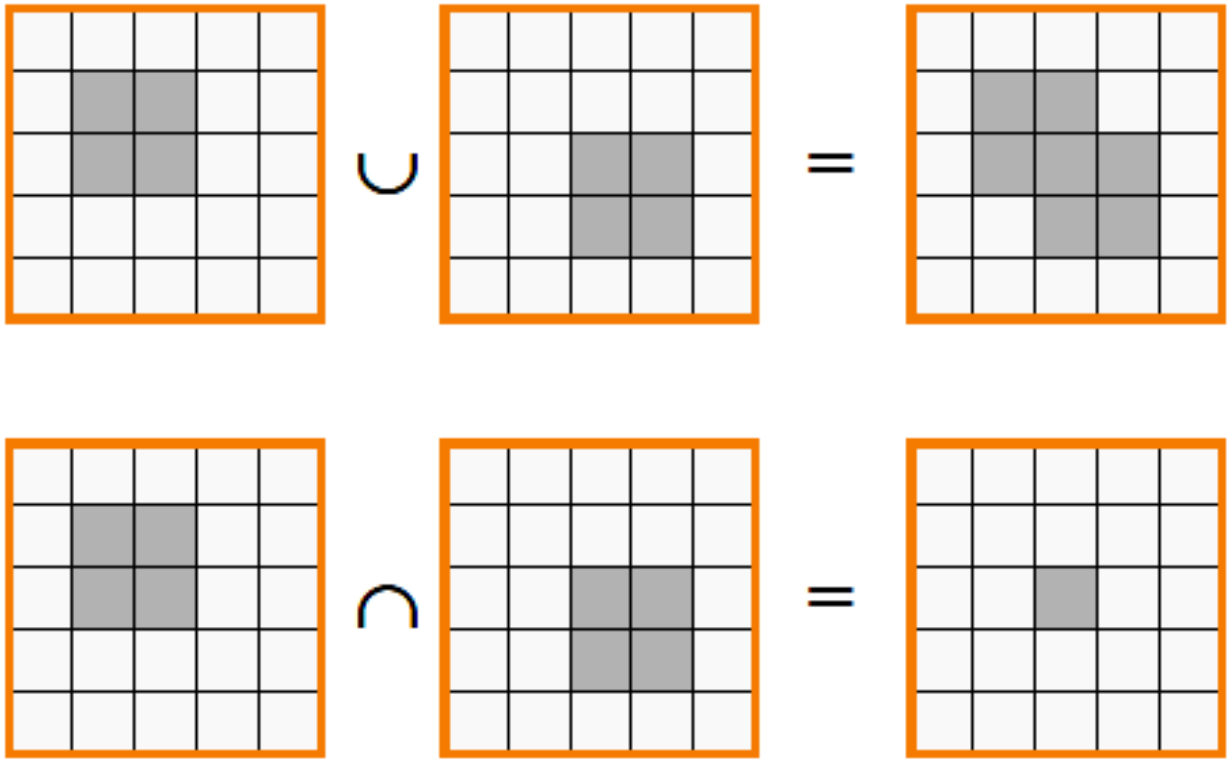
Voxel Boolean Operations

- How?



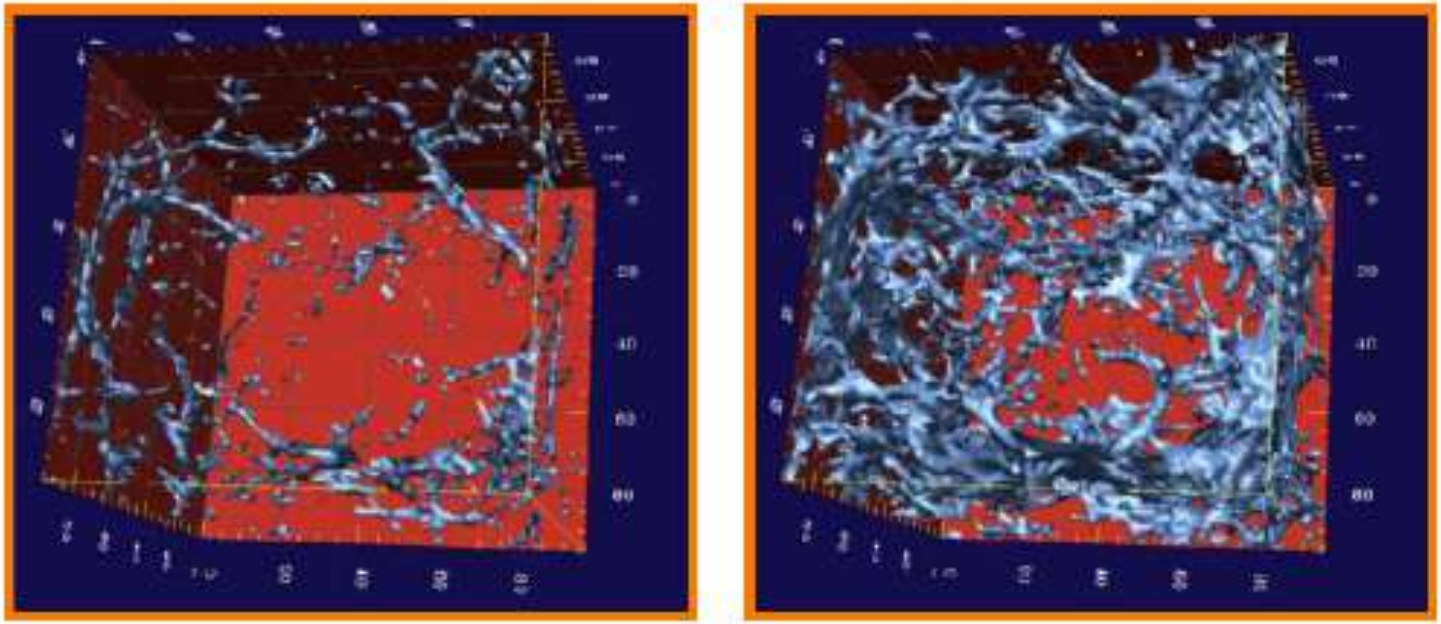
Voxel Boolean Operations

- Compare objects voxel by voxel



Voxel Display

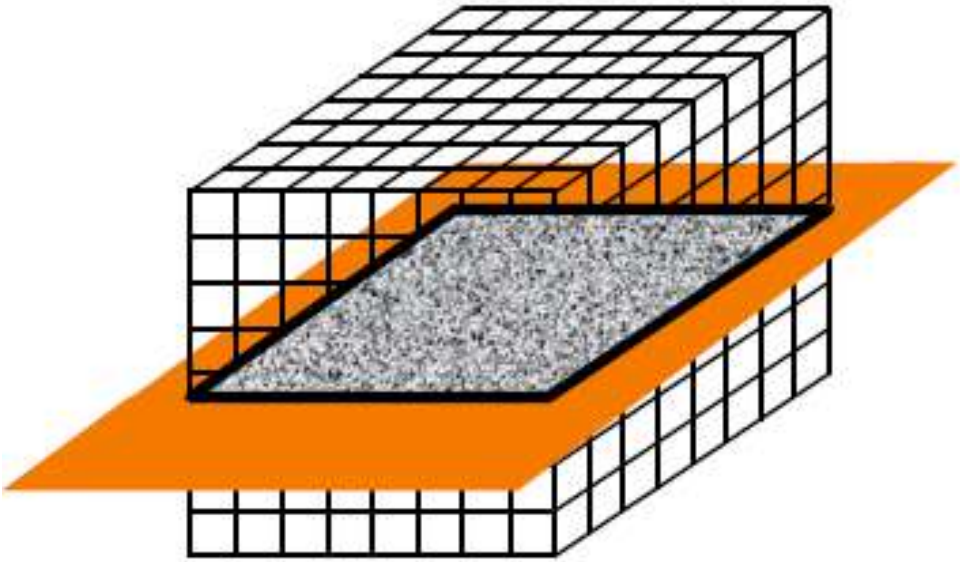
- Render surfaces bounding volumetric regions of constant value (e.g., density)

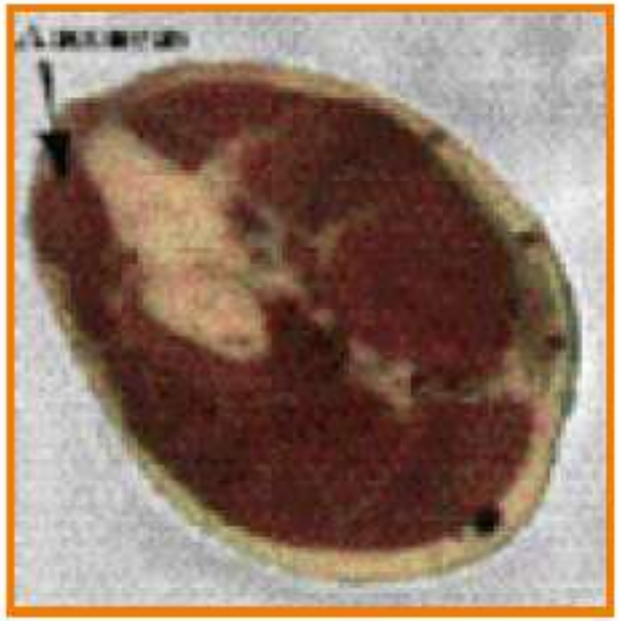


Isosurface Visualization
Princeton University

Voxel Display

- Slicing
  - Draw 2D image resulting from intersecting voxels with a plane

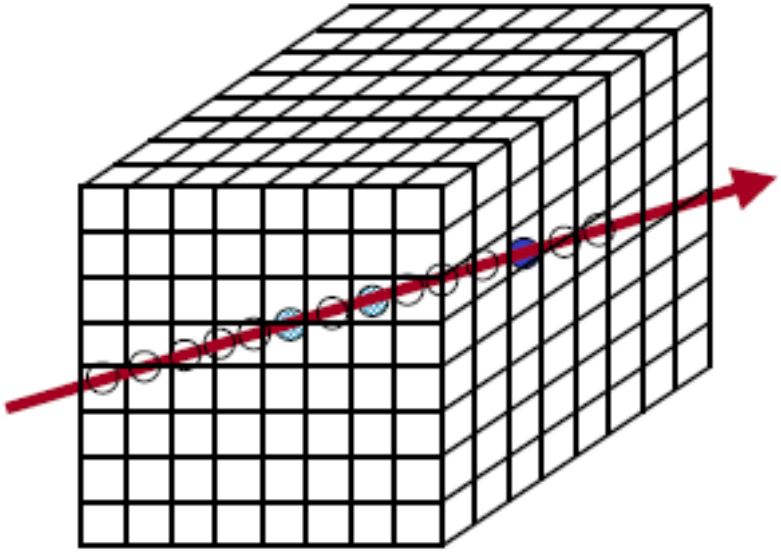


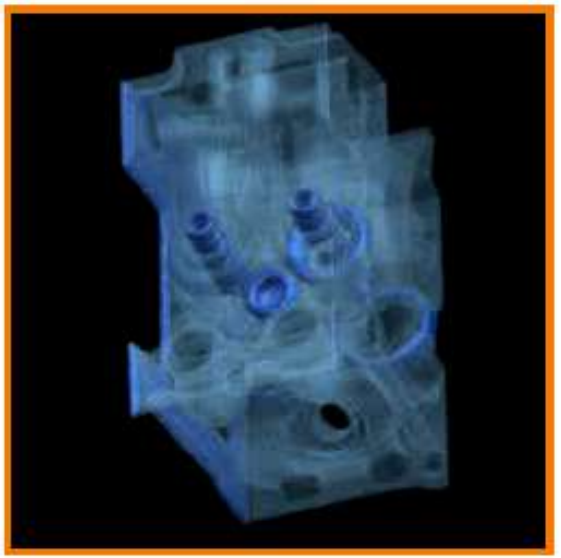


Visible Human
(National Library of Medicine)

Voxel Display

- Ray casting
  - Integrate density along rays trough voxels

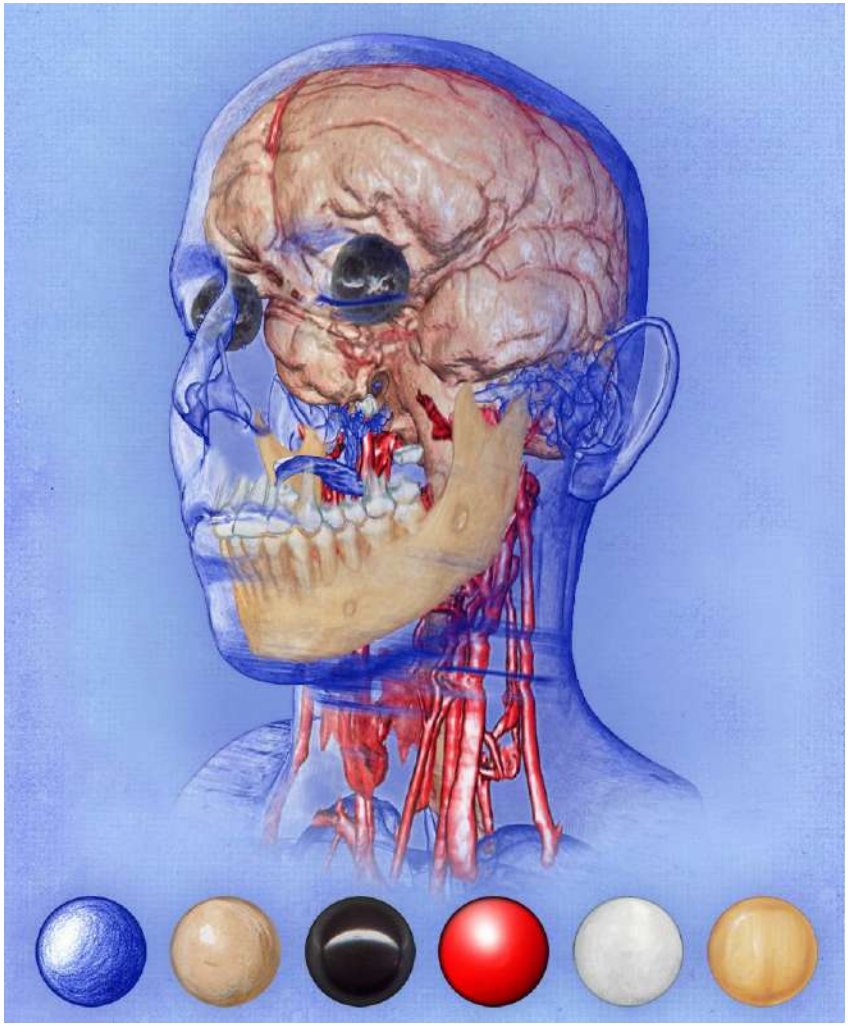




Engine Block
Stanford University

Voxel Display

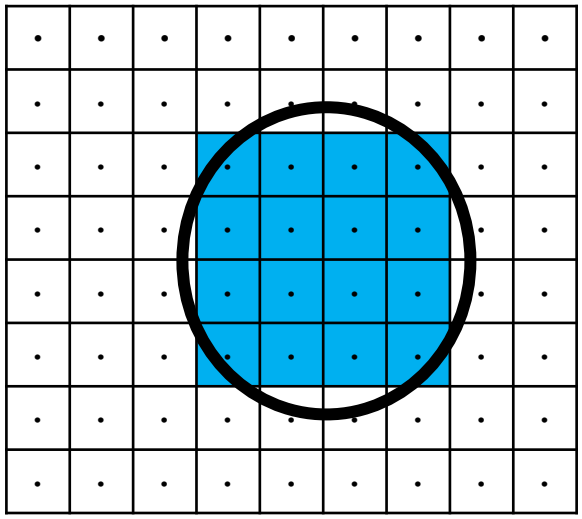
- Extended ray casting
  - Complex transfer functions
  - Map voxel densities to materials
  - Evaluate “normals” at material transitions



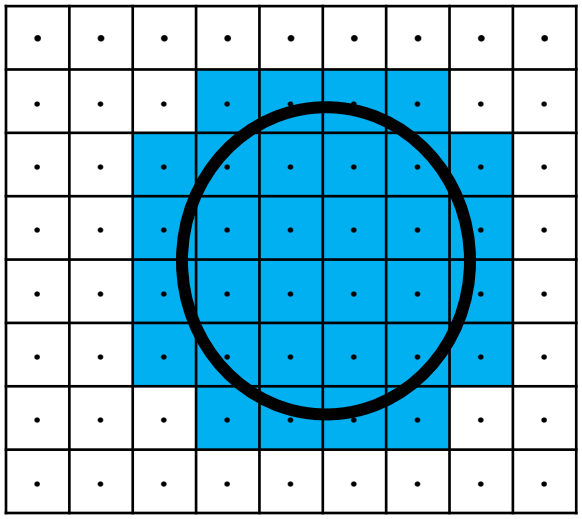
Bruckner et al. 2007

Voxelization: From Surfaces to Voxels

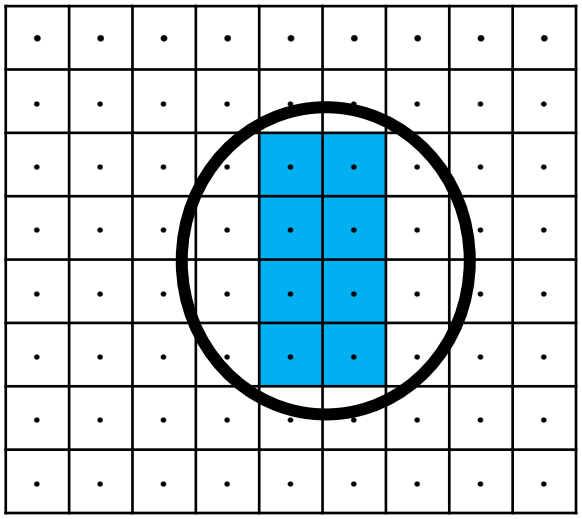
- Binary classification
  - 1: inside the volume, 0: outside



Center Inside



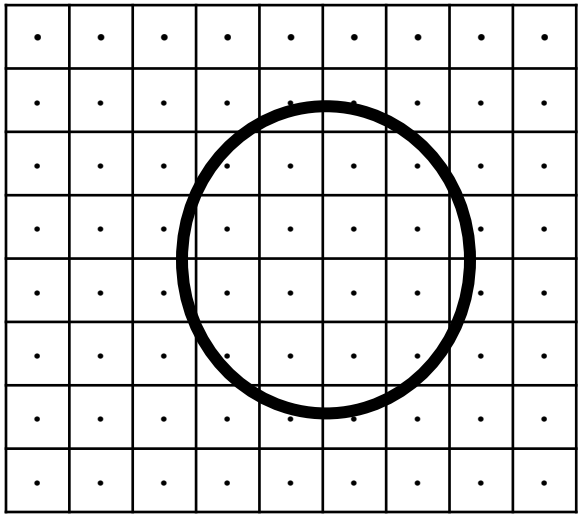
Partially Inside



Completely Inside

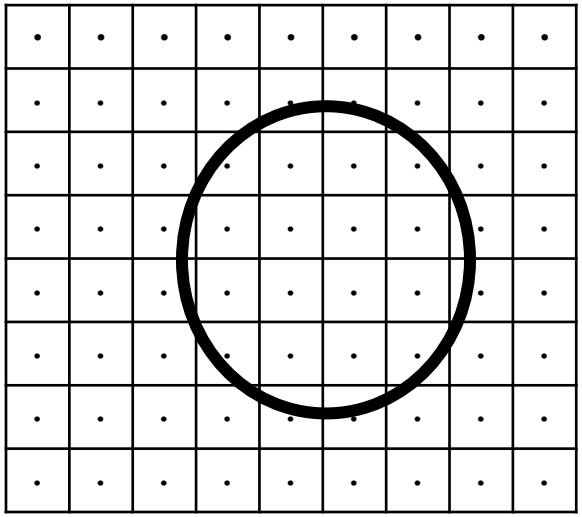
Voxelization: From Surfaces to Voxels

- How?



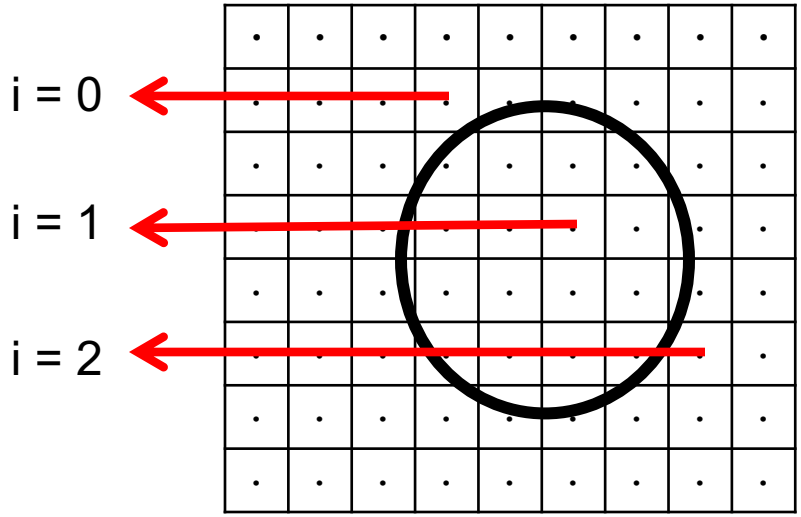
Voxelization: From Surfaces to Voxels

- Ray casting



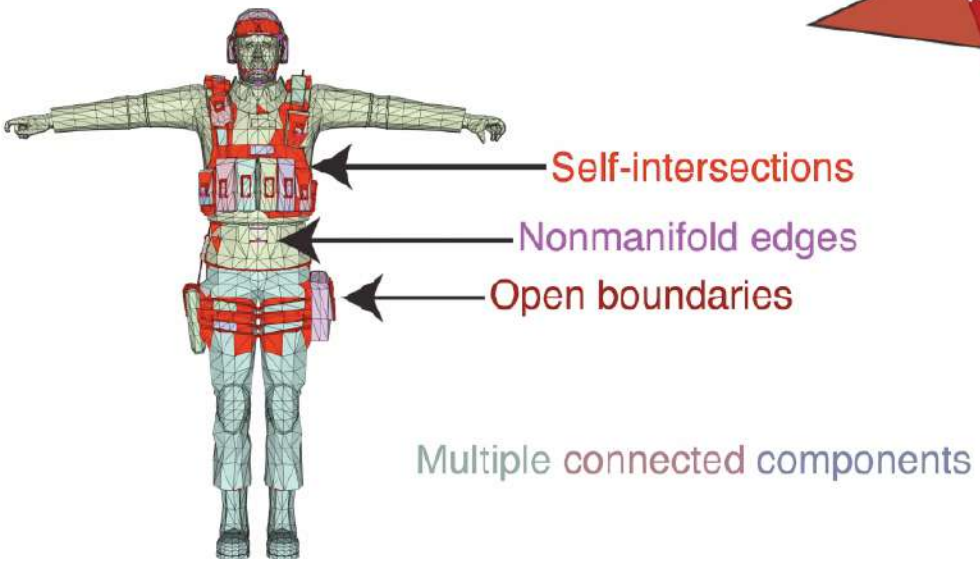
Voxelization: From Surfaces to Voxels

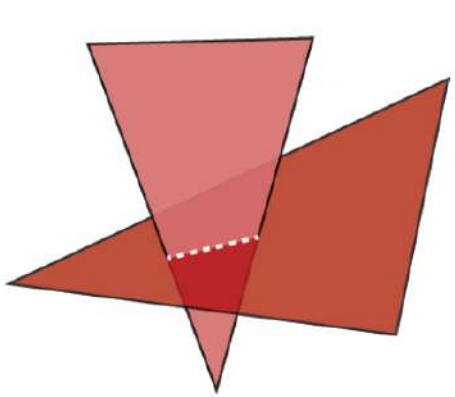
- Ray casting
  - Trace a ray from each voxel center
  - Count intersections
    - Odd: inside
    - Even: outside

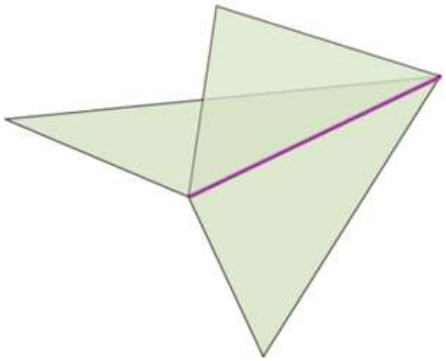


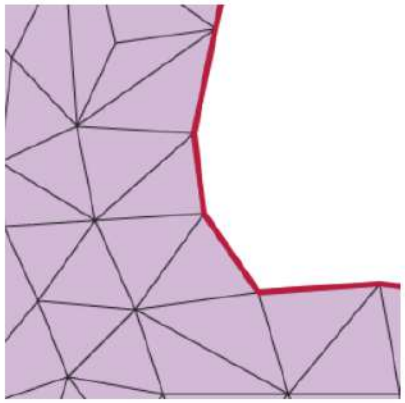
Voxelization

- Imperfect input meshes
  - Voxelization fails





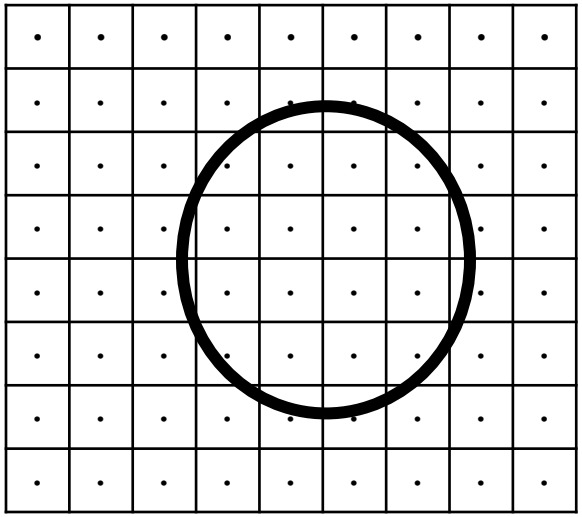




Source: Jacobson, 2013

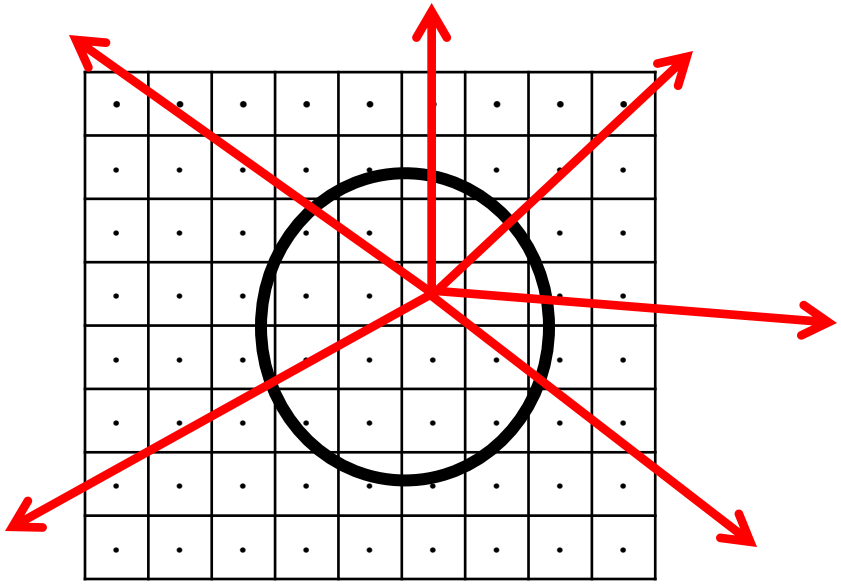
Robust Voxelization

- How?
  - Hint: use ray casting



Robust Voxelization

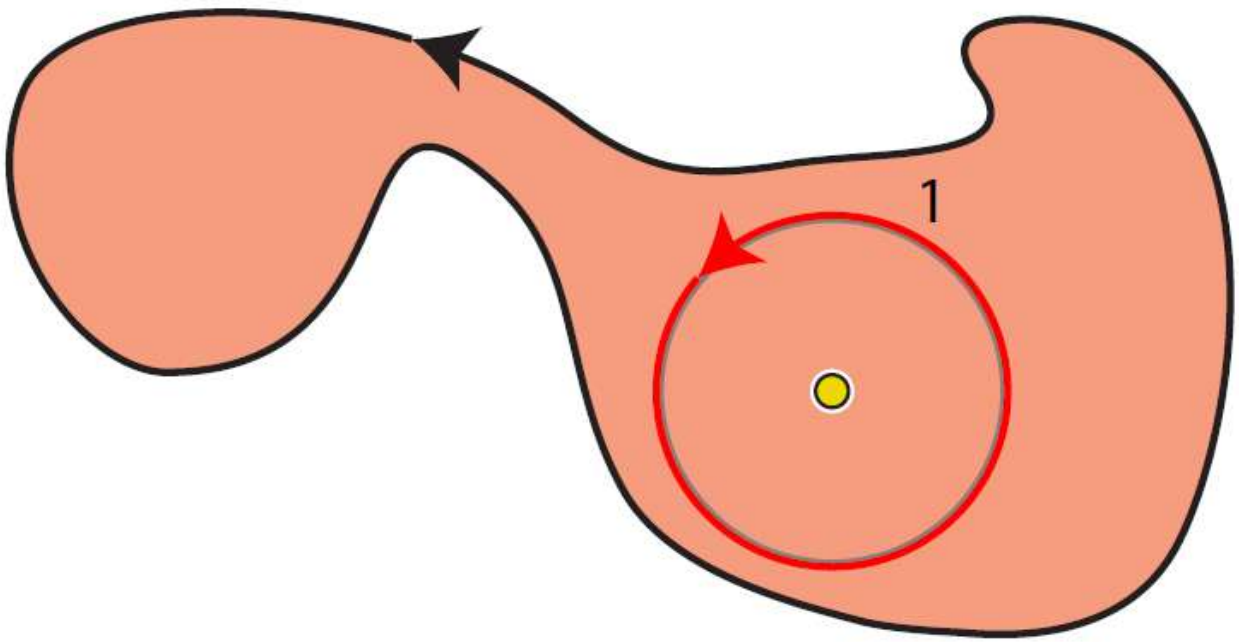
- Ray casting
  - Trace many rays in different directions
  - Combine results



Winding Numbers

- If shape is watertight, winding number is perfect measure of inside/outside

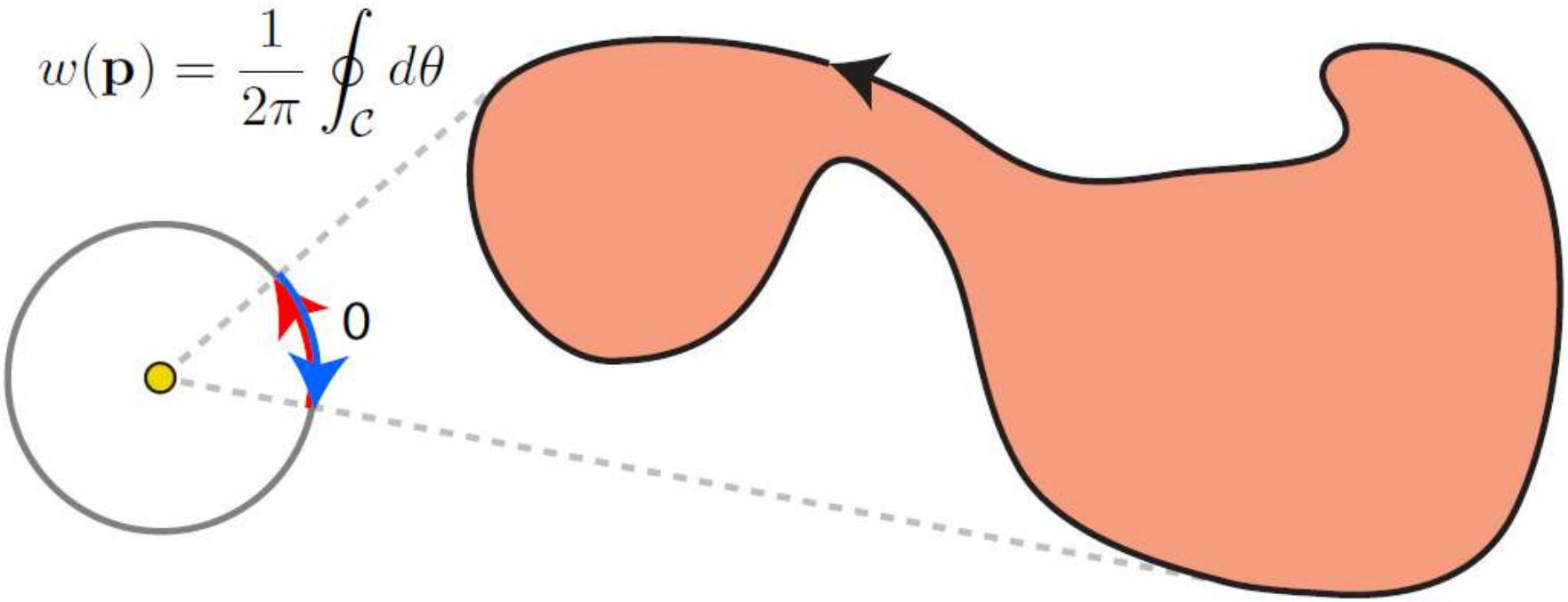
$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



Source: Jacobson, 2013

Winding Numbers

- If shape is watertight, winding number is perfect measure of inside/outside

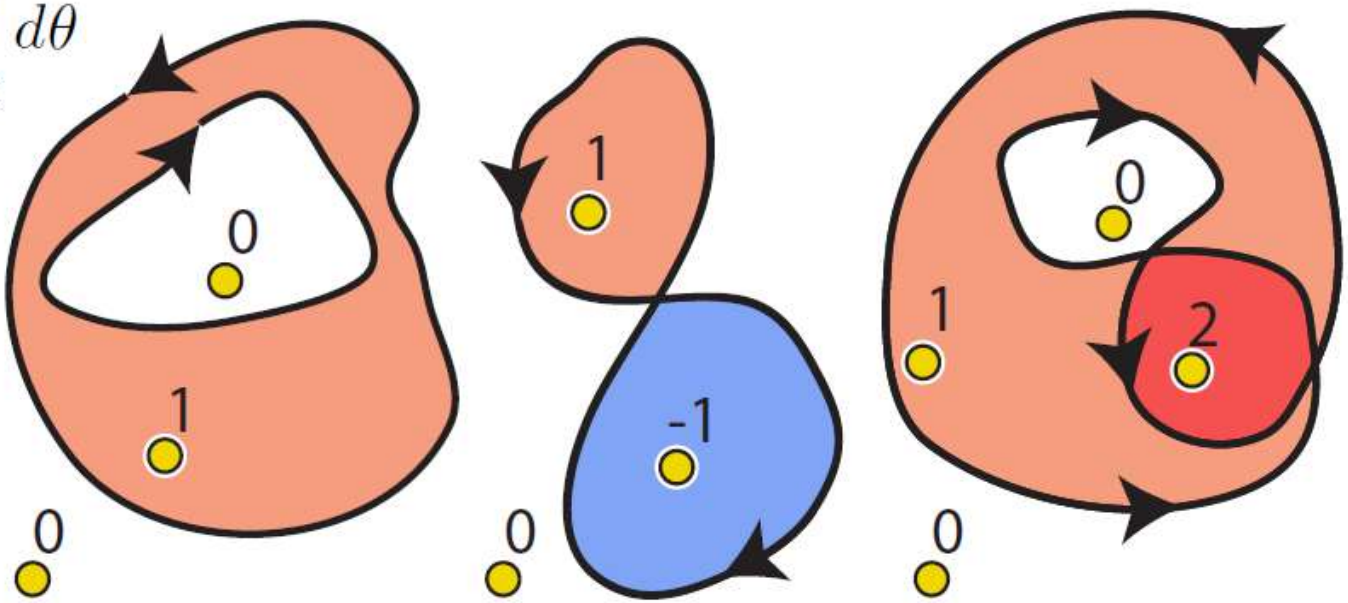


Source: Jacobson, 2013

Winding Numbers

- Winding number uses orientation to treat inside/outside as a signed integer

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



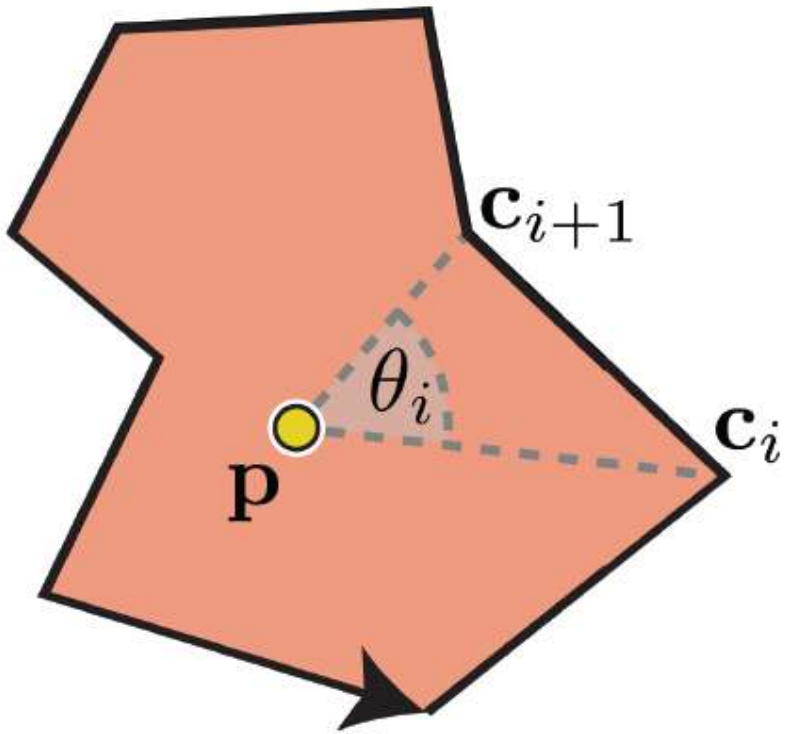
Source: Jacobson, 2013

Computing Winding Numbers

- Naïve discretization is simple and exact

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$

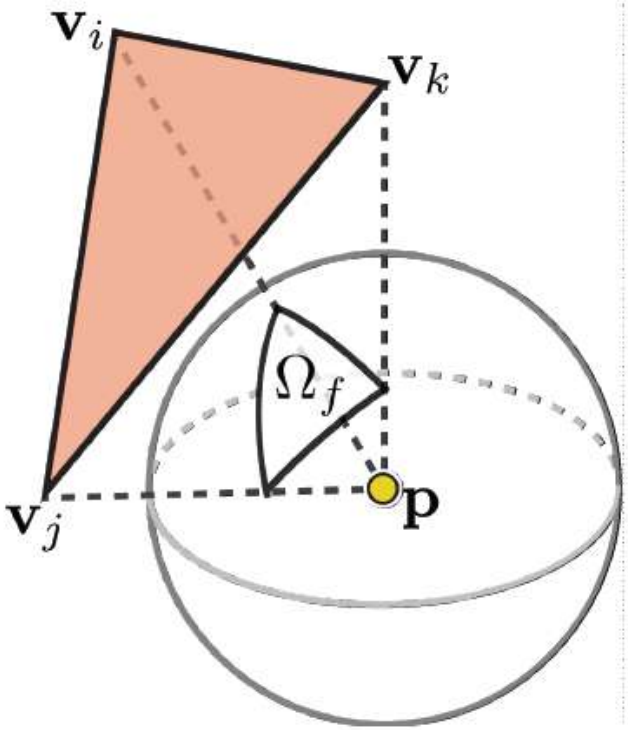
$$w(\mathbf{p}) = \frac{1}{2\pi} \sum_{i=1}^{n} \theta_i$$



Source: Jacobson, 2013

Winding Numbers in 3D

- Generalizes elegantly to 3D via solid angle



$$w(\mathbf{p}) = \frac{1}{4\pi} \iint_{\mathcal{S}} \sin(\phi) d\theta d\phi$$

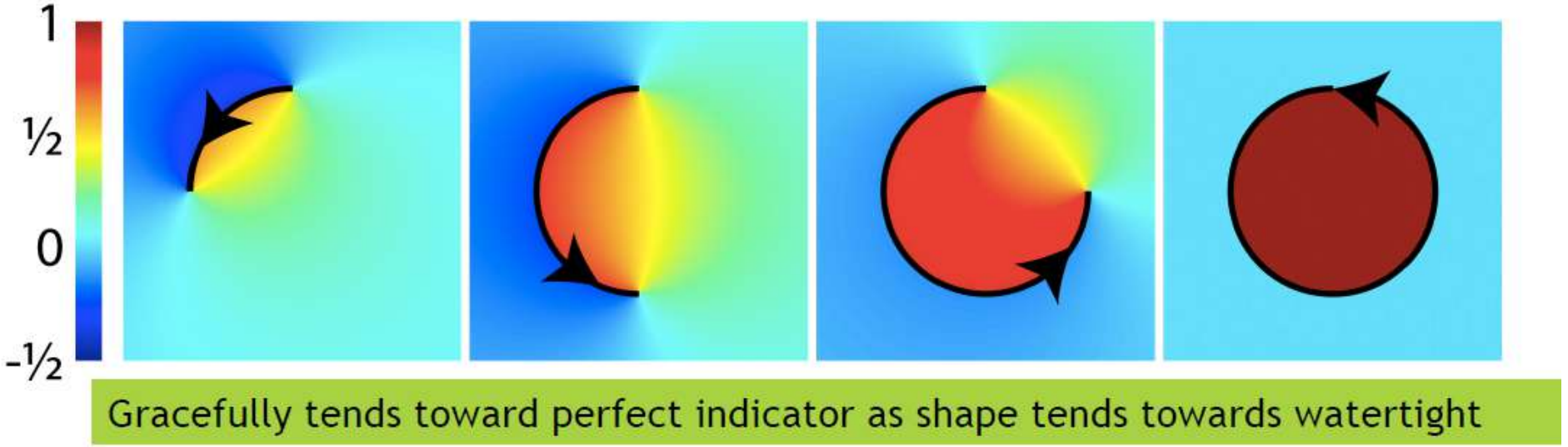
$$w(\mathbf{p}) = \frac{1}{4\pi} \sum_{f=1}^{m} \Omega_f$$

Source: Jacobson, 2013

Winding Numbers for Open Shapes

- What happens if the shape is open?

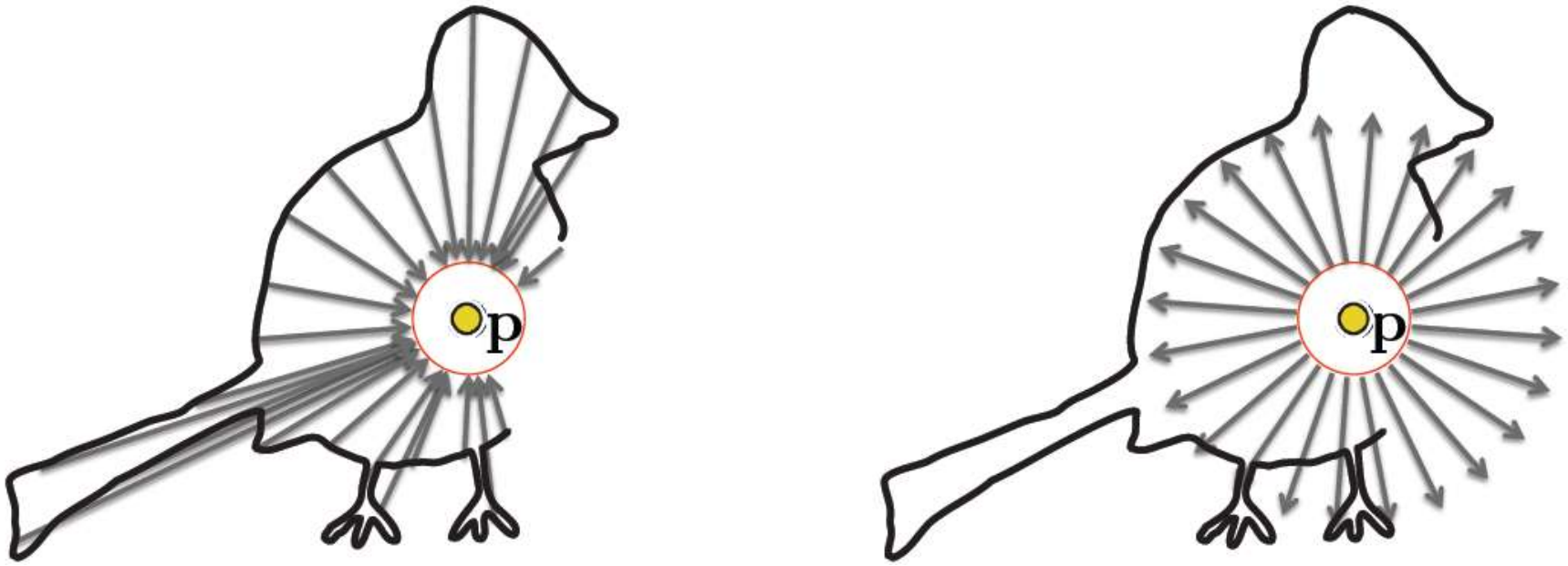
$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



Source: Jacobson, 2013

Robust Voxelization using Winding Numbers

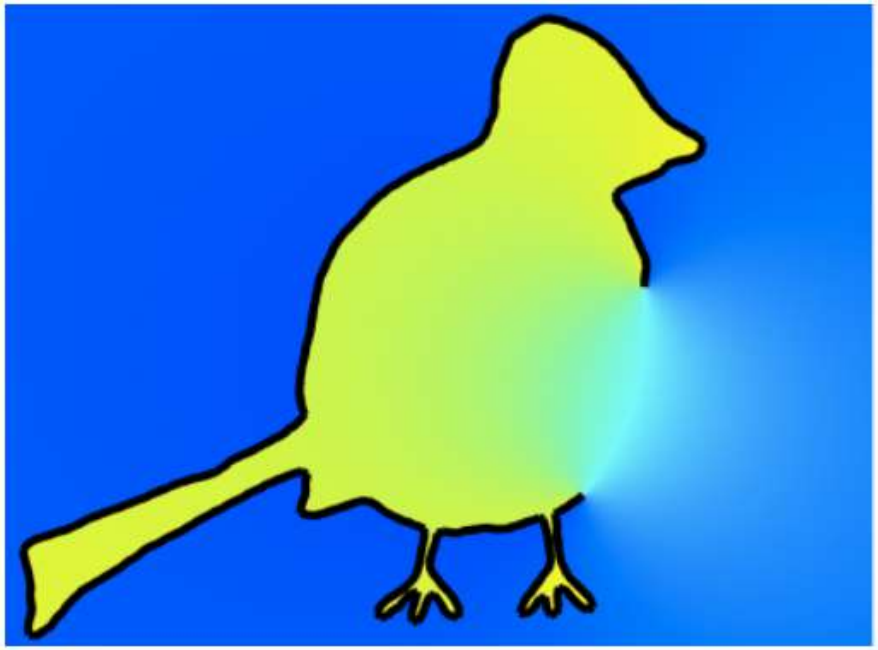
- Direct vs. sampling computation

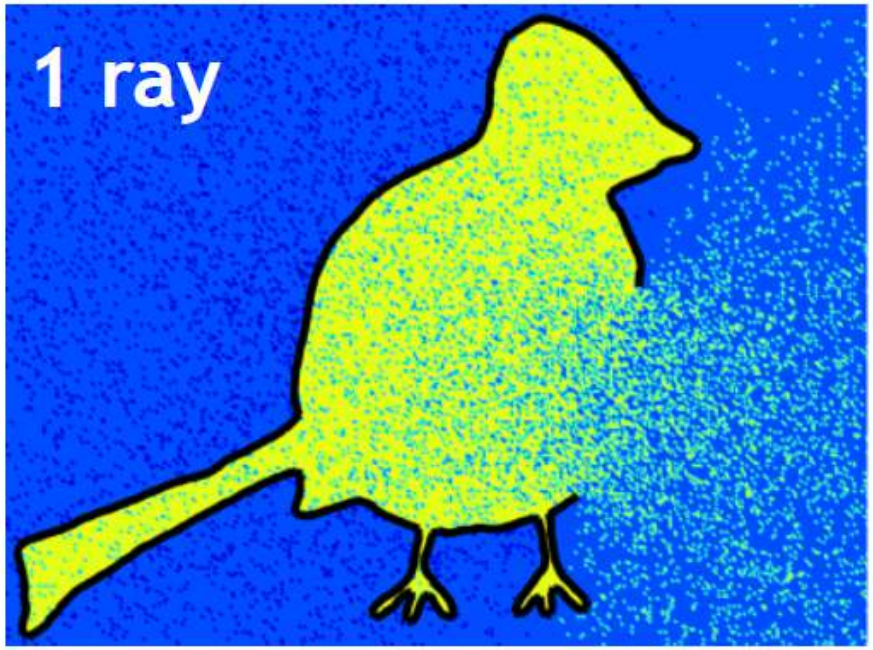


Source: Jacobson, 2013

Robust Voxelization using Winding Numbers

- Direct vs. sampling computation

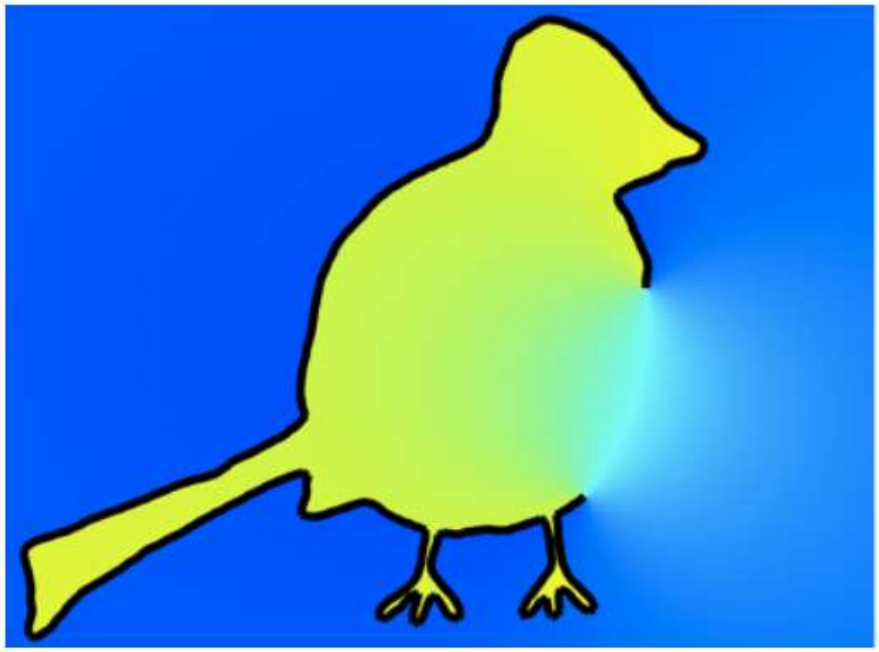


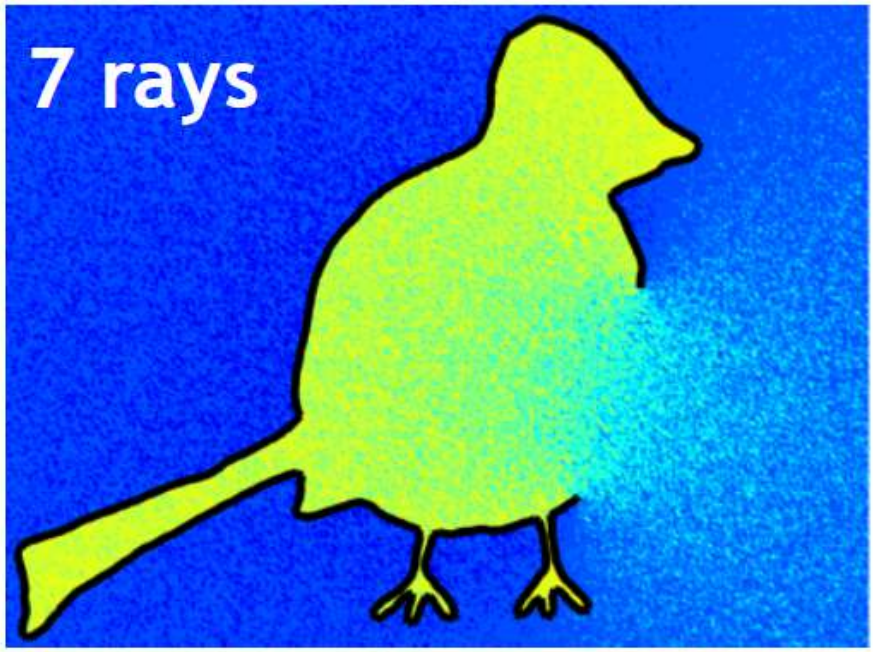


Source: Jacobson, 2013

Robust Voxelization using Winding Numbers

- Direct vs. sampling computation

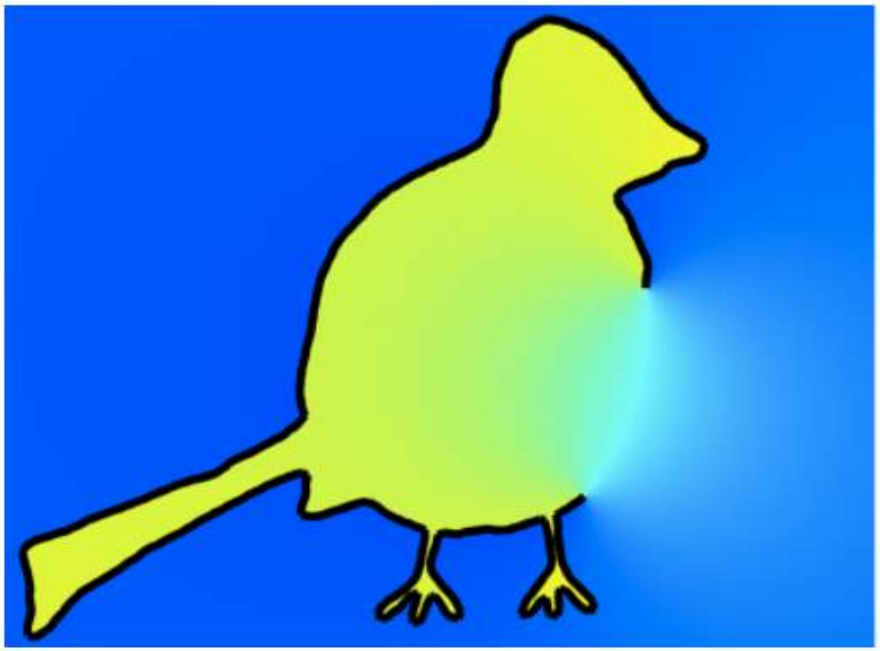


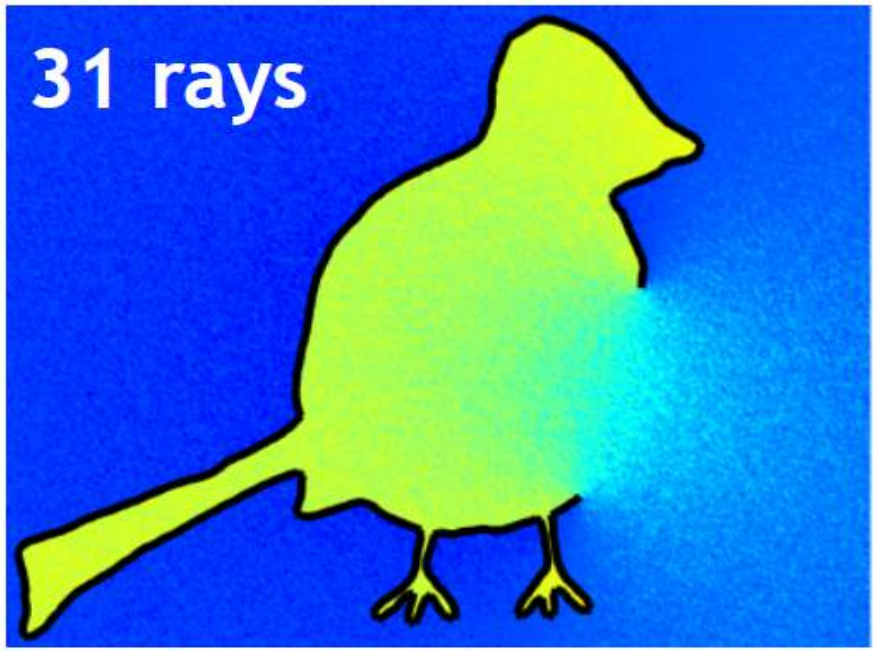


Source: Jacobson, 2013

Robust Voxelization using Winding Numbers

- Direct vs. sampling computation

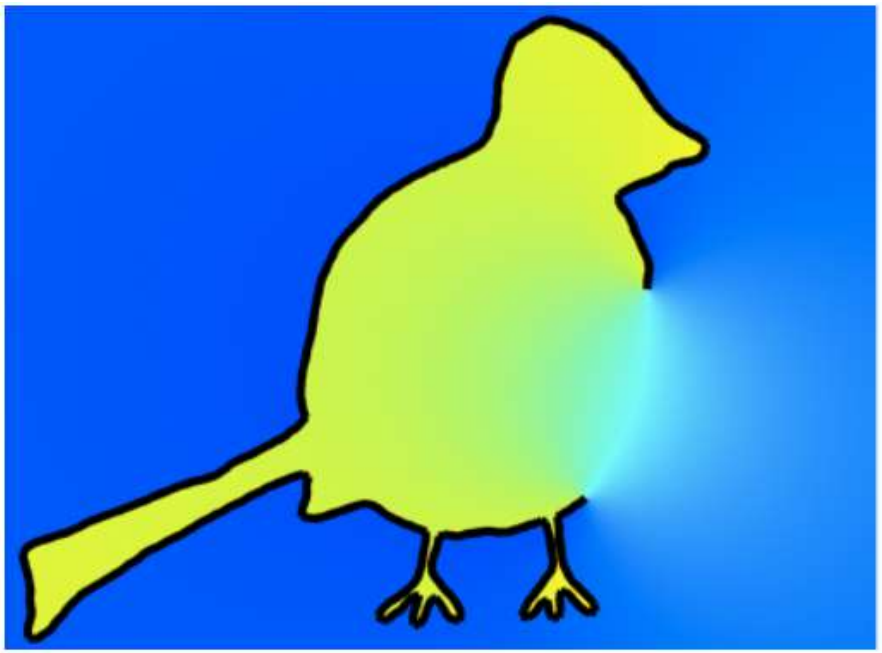


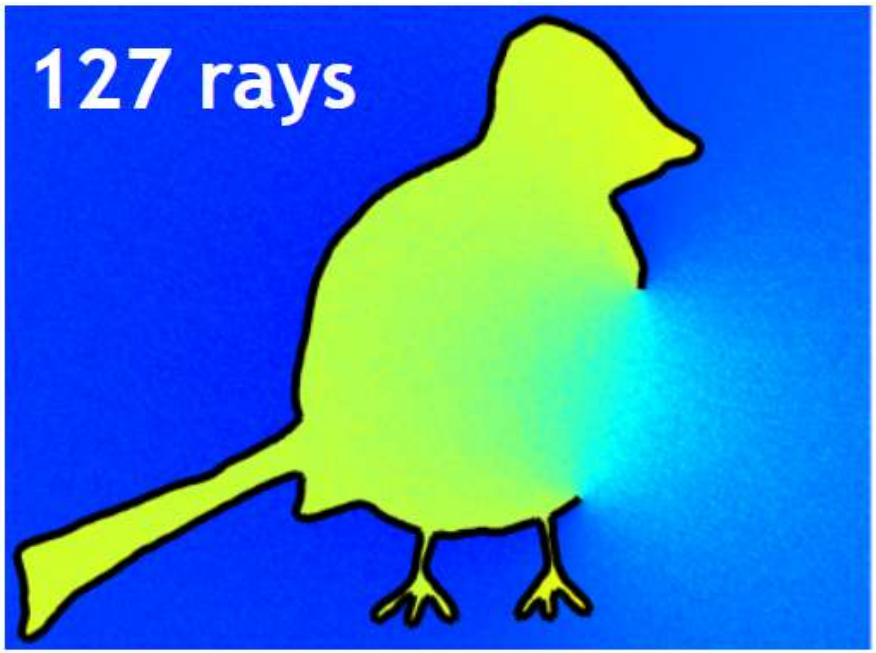


Source: Jacobson, 2013

Robust Voxelization using Winding Numbers

- Direct vs. sampling computation

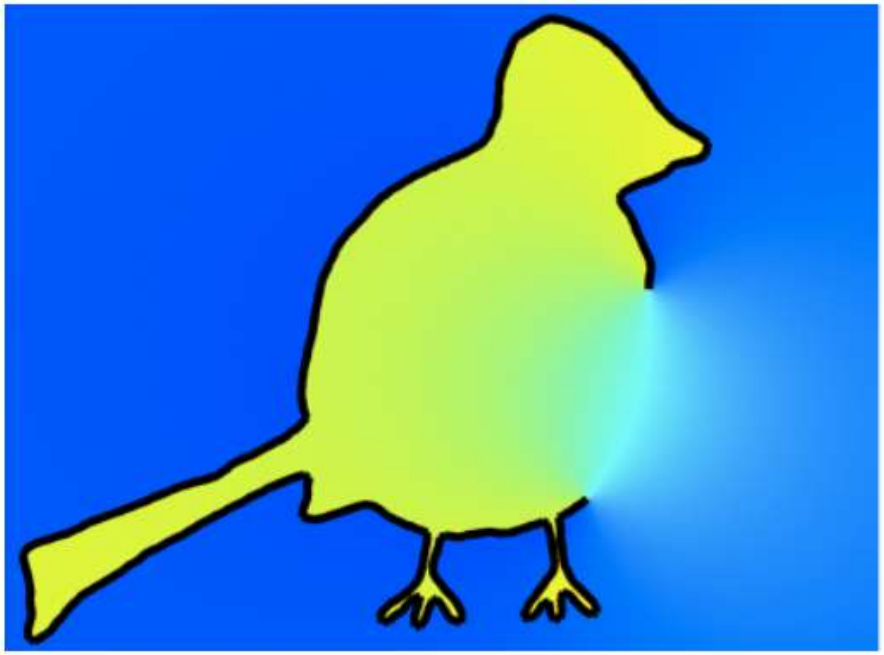


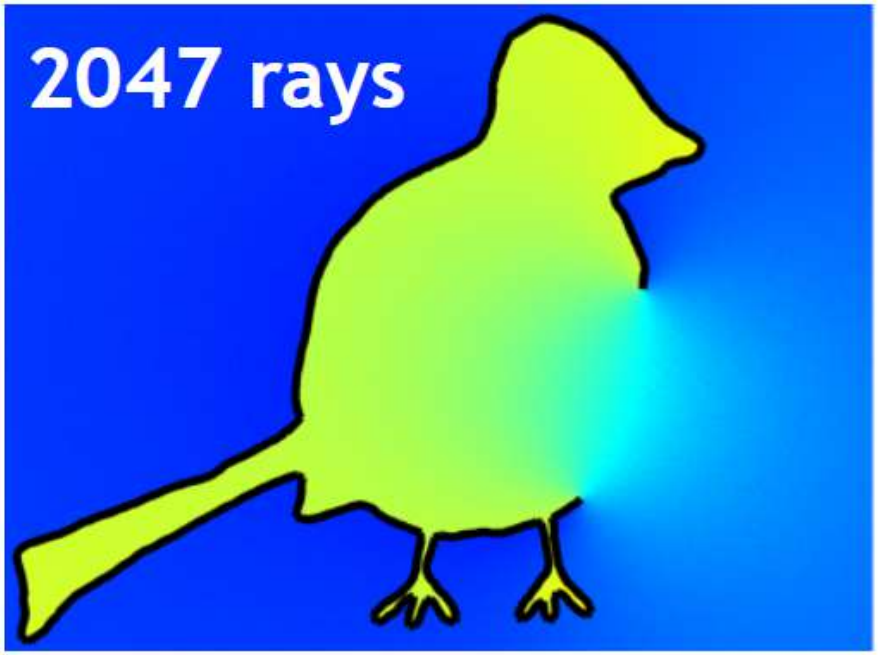


Source: Jacobson, 2013

Robust Voxelization using Winding Numbers

- Direct vs. sampling computation

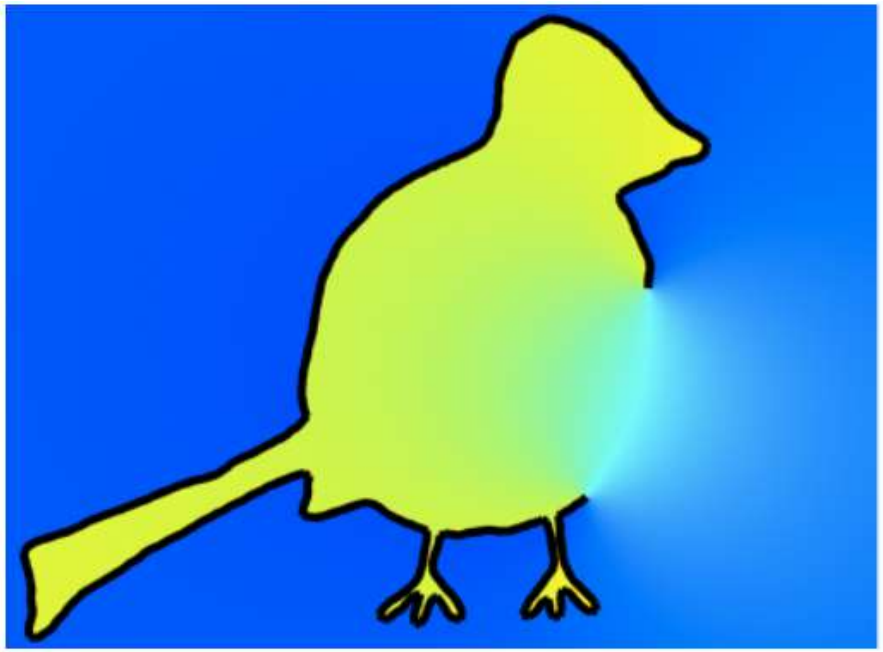




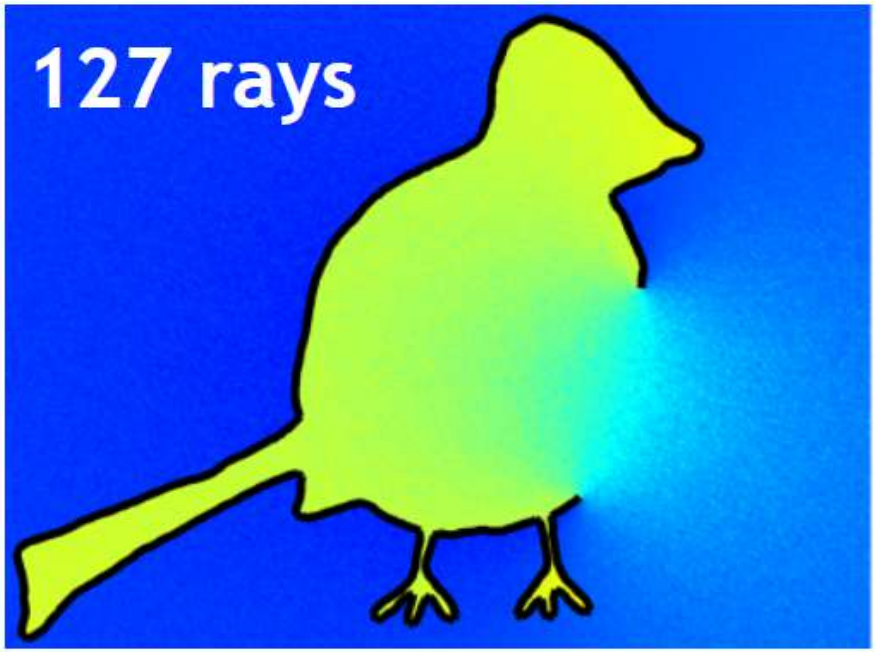
Source: Jacobson, 2013

Robust Voxelization

- Winding numbers
  - For each voxel
    - Compute its winding number by traversing over all triangles

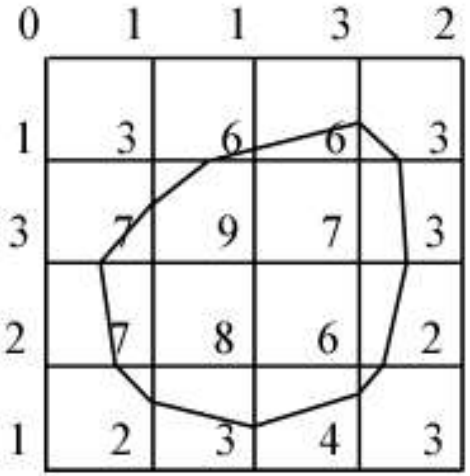


- Ray casting
  - For each voxel center
    - Trace many rays in different directions
    - Combine results



From Voxels Grids to Surfaces

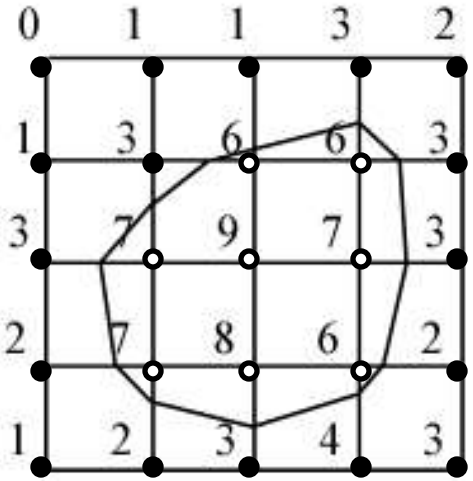
- Marching squares (2D)
  - A simple example: extract isosurface equal to 5



http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

From Voxels Grids to Surfaces

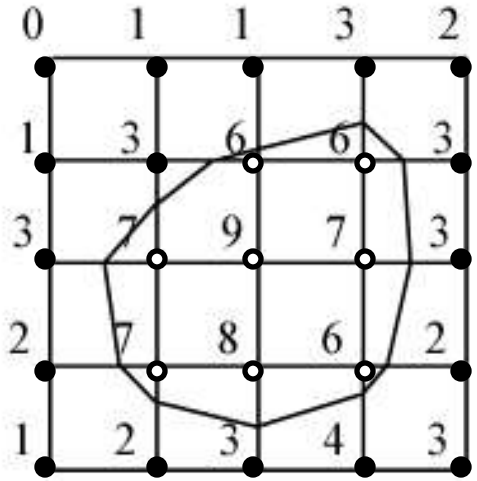
- Marching squares (2D)
  - A simple example: extract isosurface equal to 5

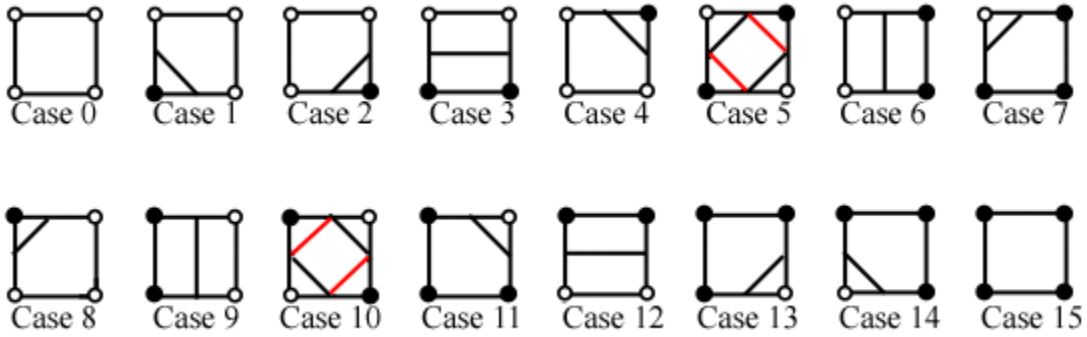


http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

From Voxels Grids to Surfaces

- Marching squares (2D)
  - A simple example: extract isosurface equal to 5

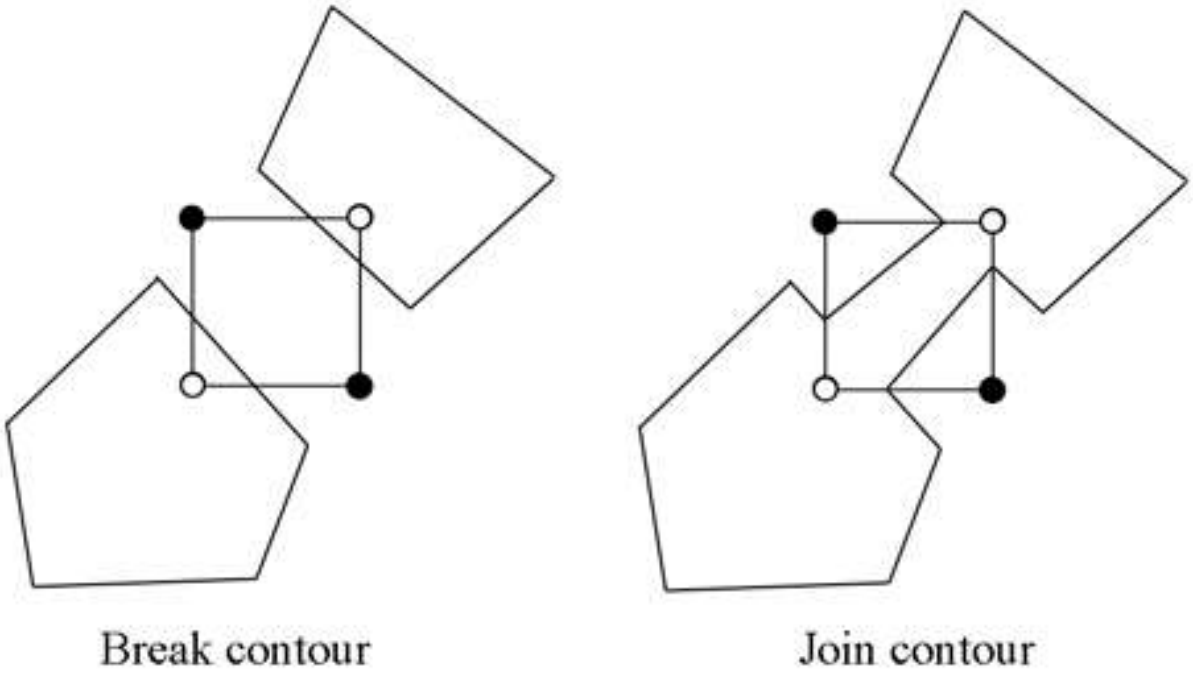




http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

From Voxels Grids to Surfaces

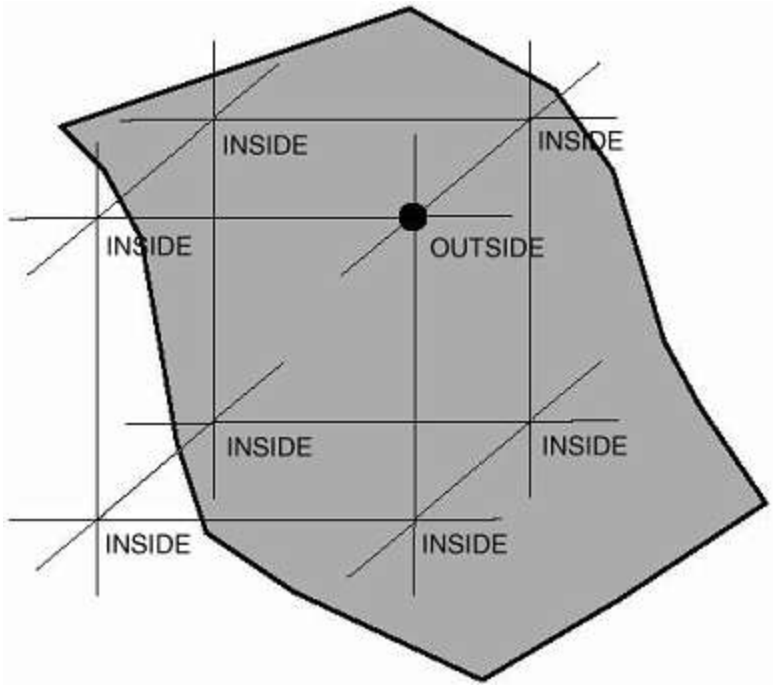
- Marching squares (2D)
  - Ambiguous cases

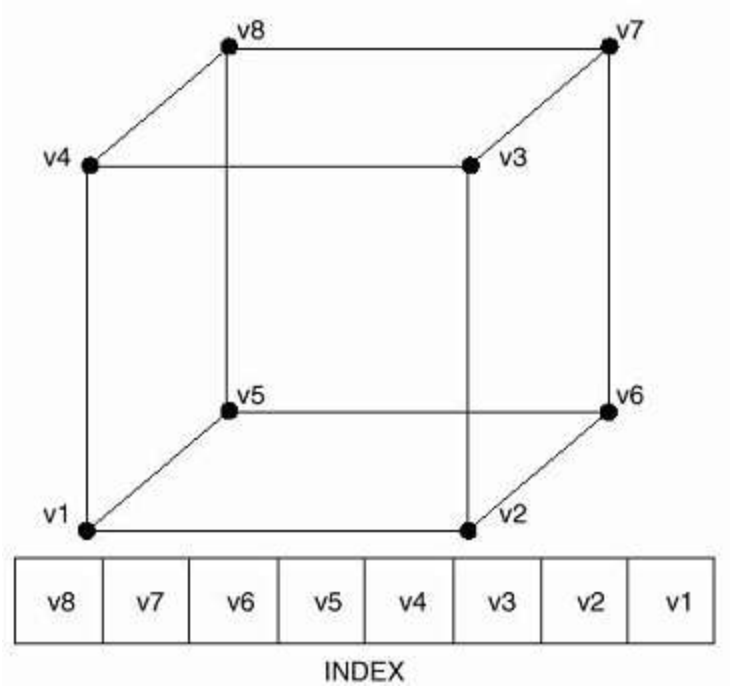


http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

From Voxels Grids to Surfaces

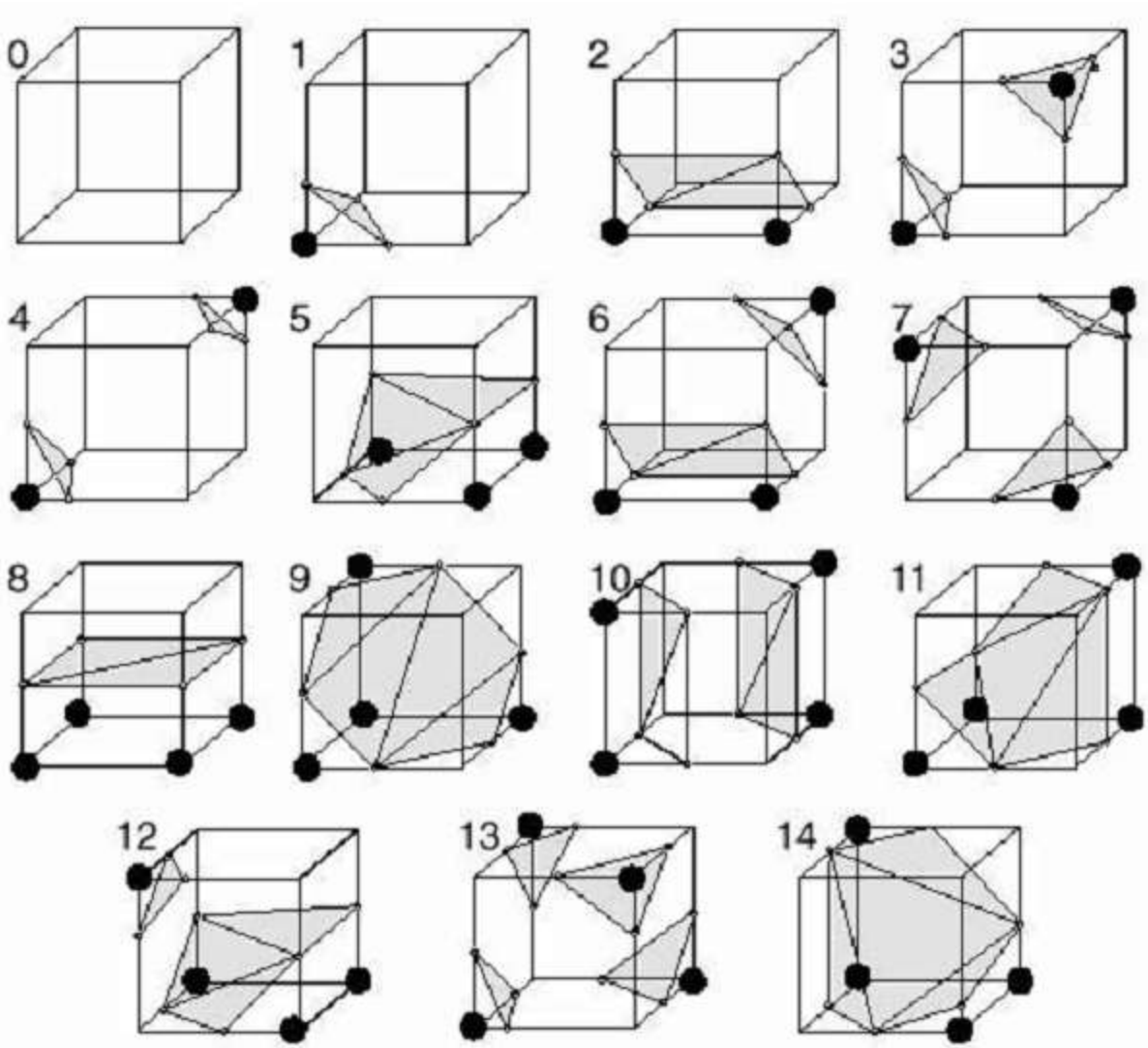
- Marching cubes (3D)
  - 256 (2^8) different situations
  - Generalized to 15 cases by rotations and symmetry





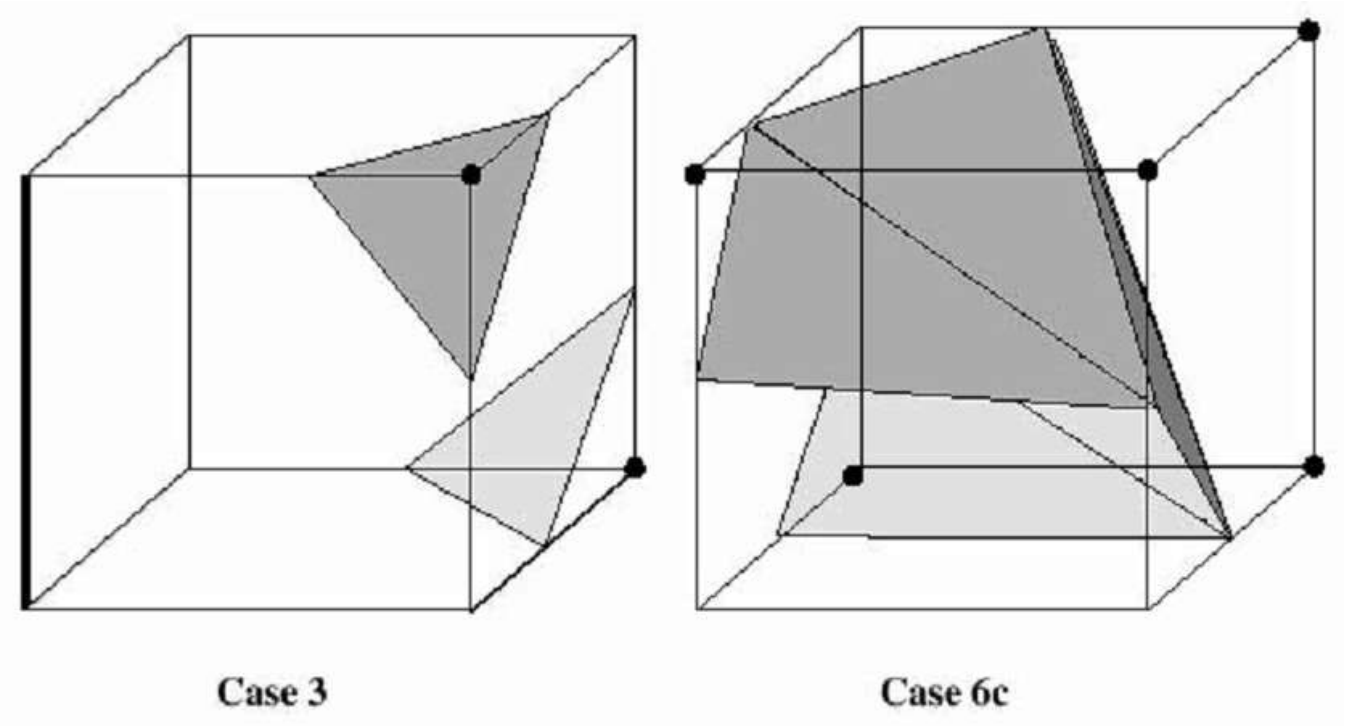
http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

Marching Cubes

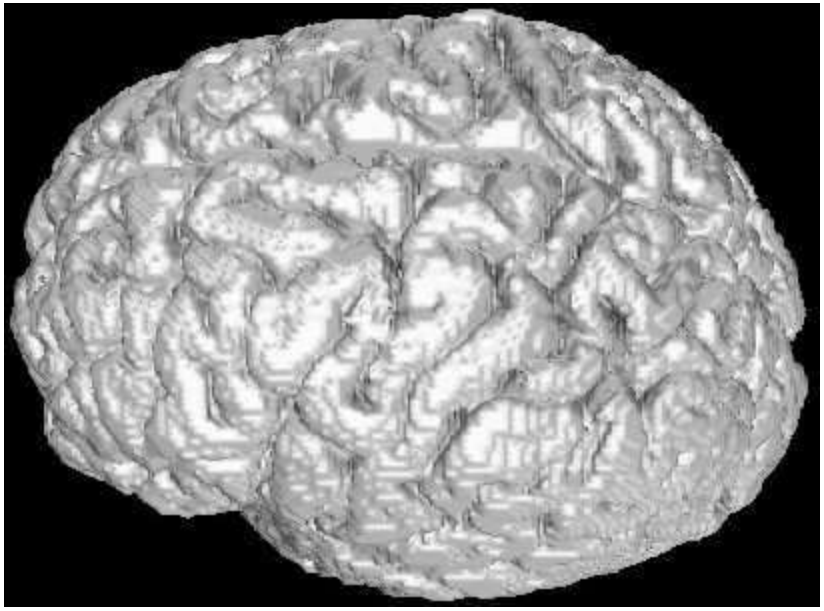


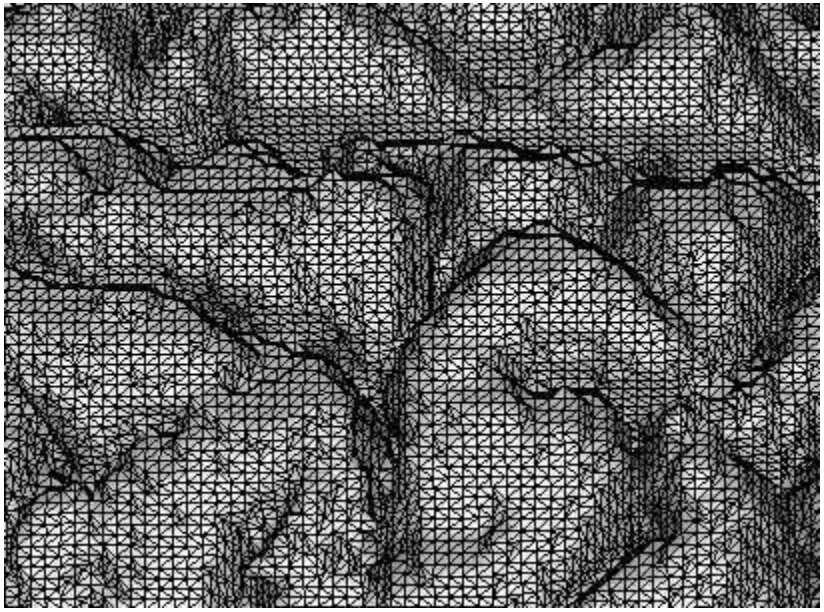
Marching Cubes

- Sometimes have to match neighbors



Marching Cubes Results





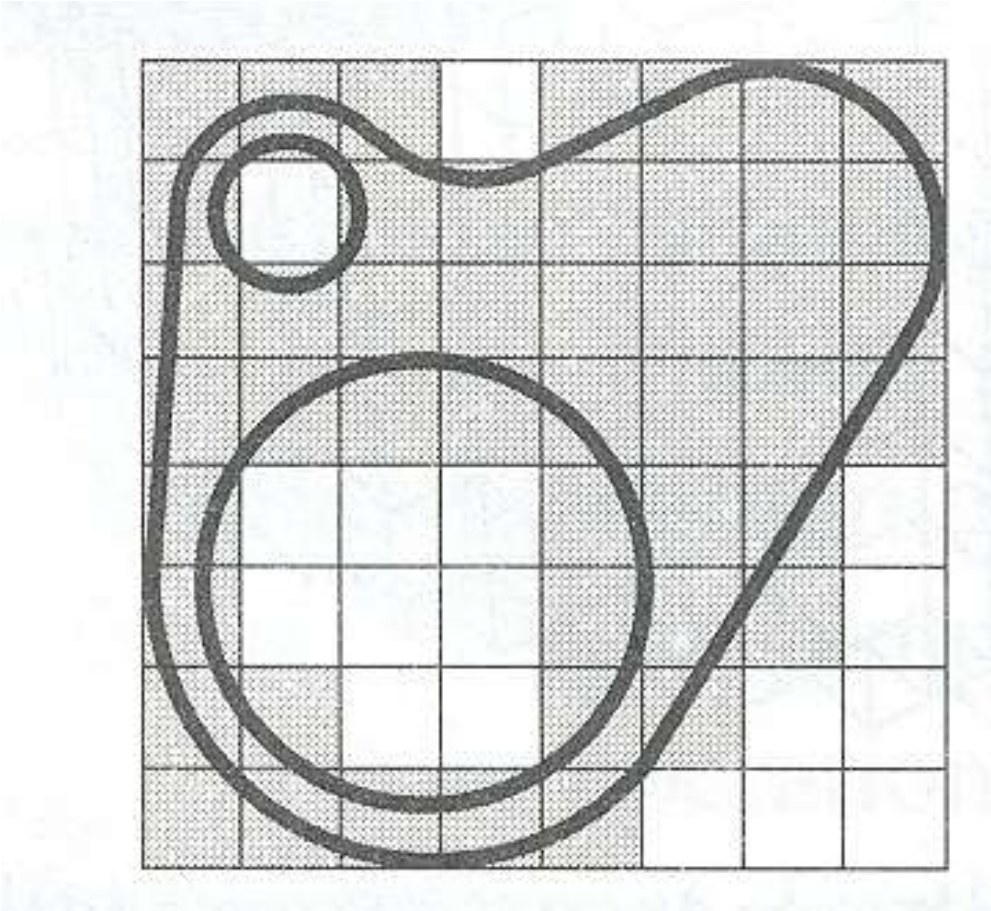
http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

Voxels

- Advantages
  - Simple, intuitive, unambiguous
  - Same complexity for all objects
  - Natural acquisition for some applications
  - Trivial Boolean operations
- Disadvantages
  - Approximate
  - Large storage requirements
  - Expensive display

Voxels

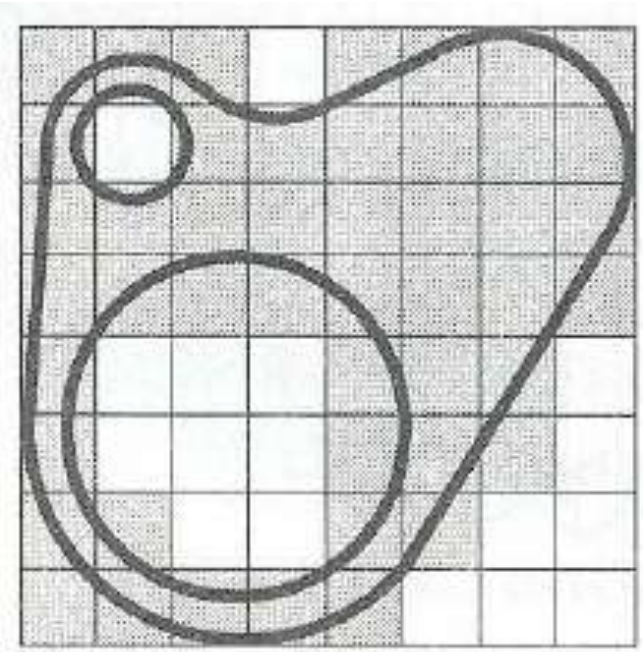
- What resolution should be used?



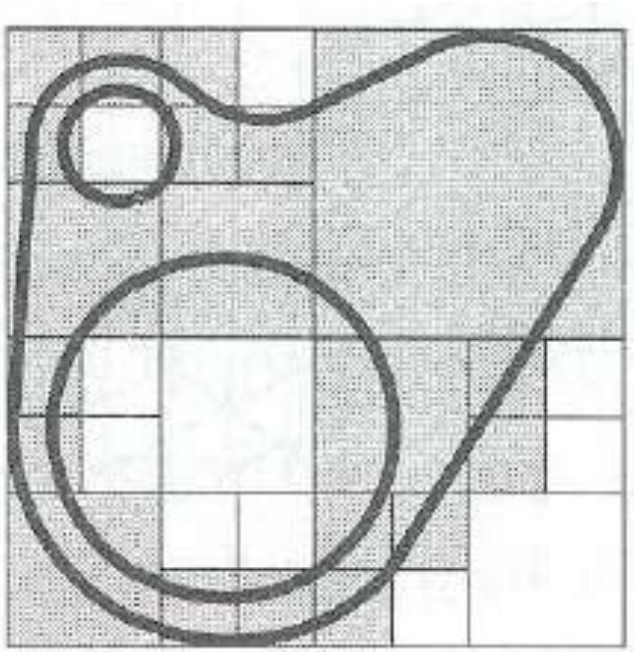
FvDFH Figure 12.21

Octrees

- Refine resolution of voxels hierarchically
  - More concise and efficient for non-uniform objects



Uniform Voxels

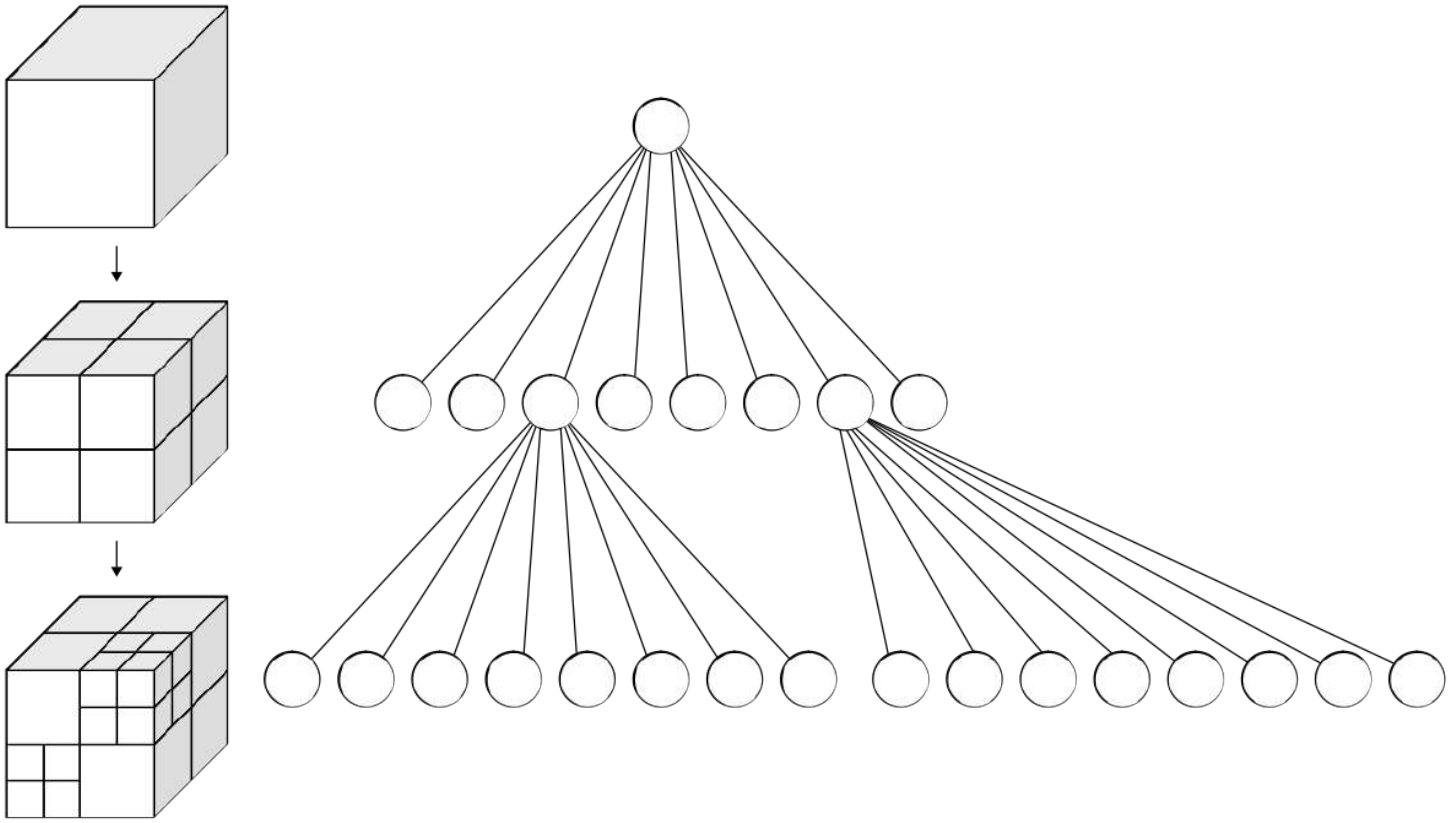


Quadtree

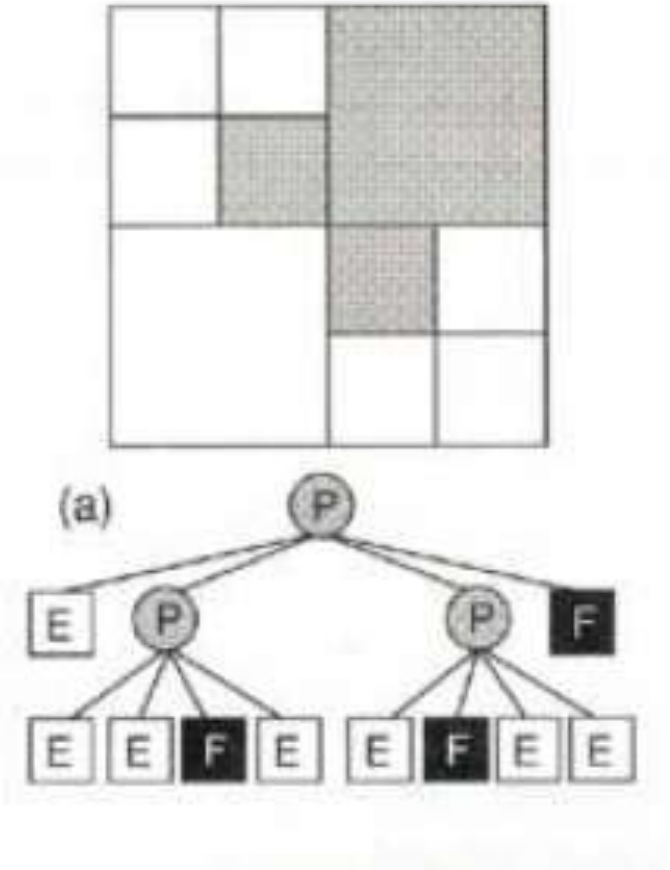
Source: FvDFH Figure 12.20

Octrees

- Encoded using a standard tree data structure
  - Empty nodes do not need to be encoded explicitly



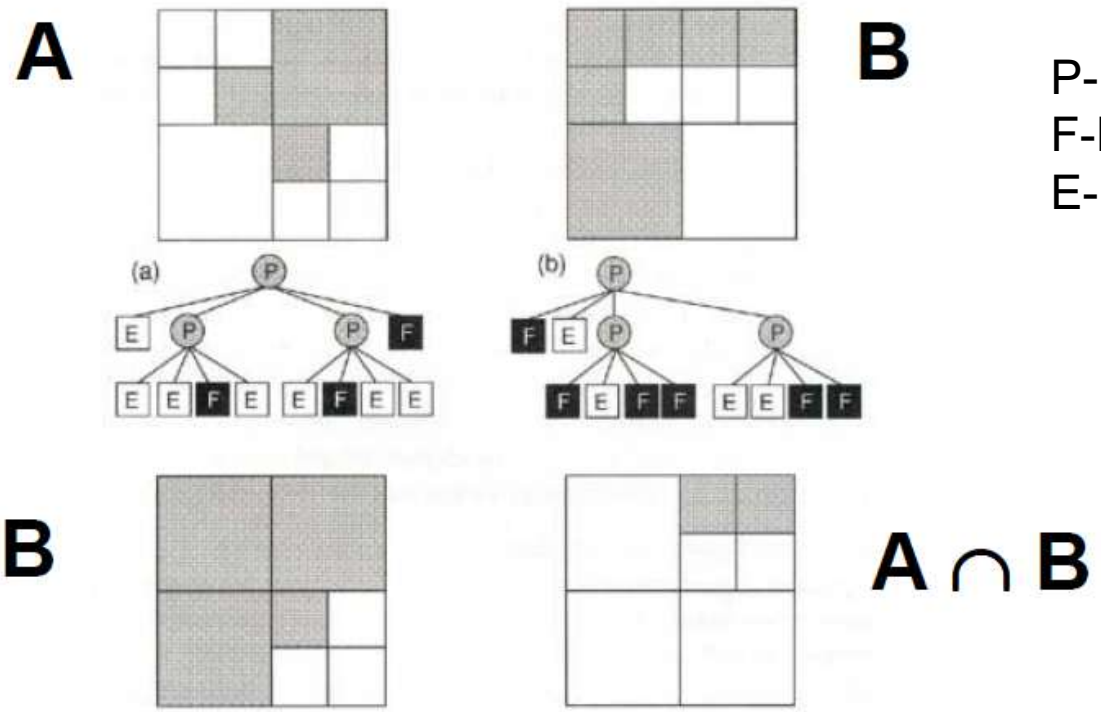
Octrees



P-Partial
F-Full
E-Empty

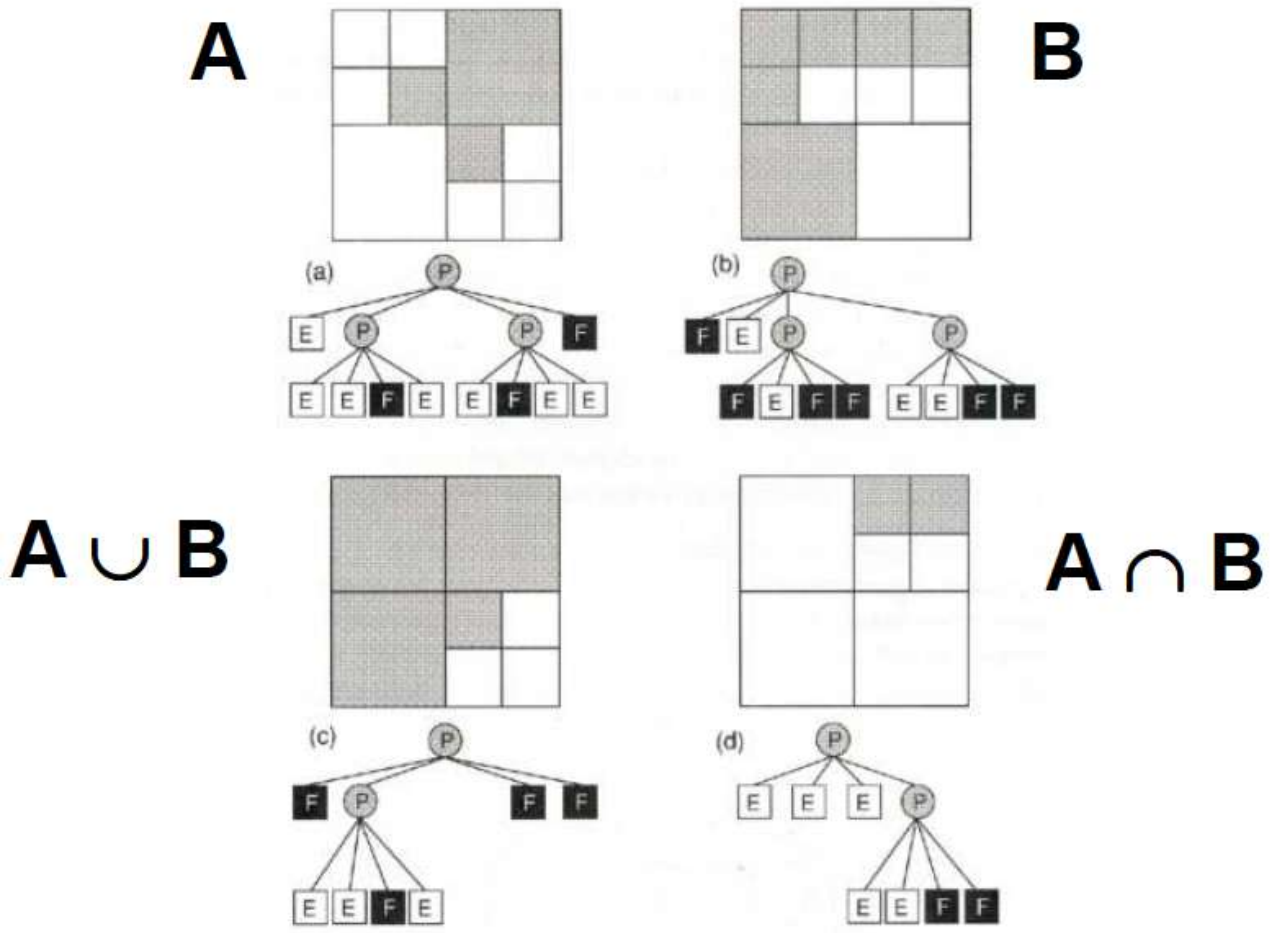
Source: FvDFH Figure 12.24

Octree Boolean Operations



Source: FvDFH Figure 12.24

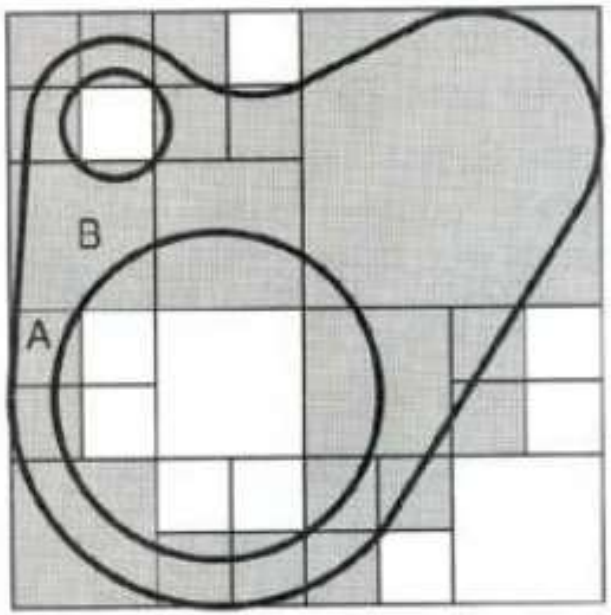
Octree Boolean Operations

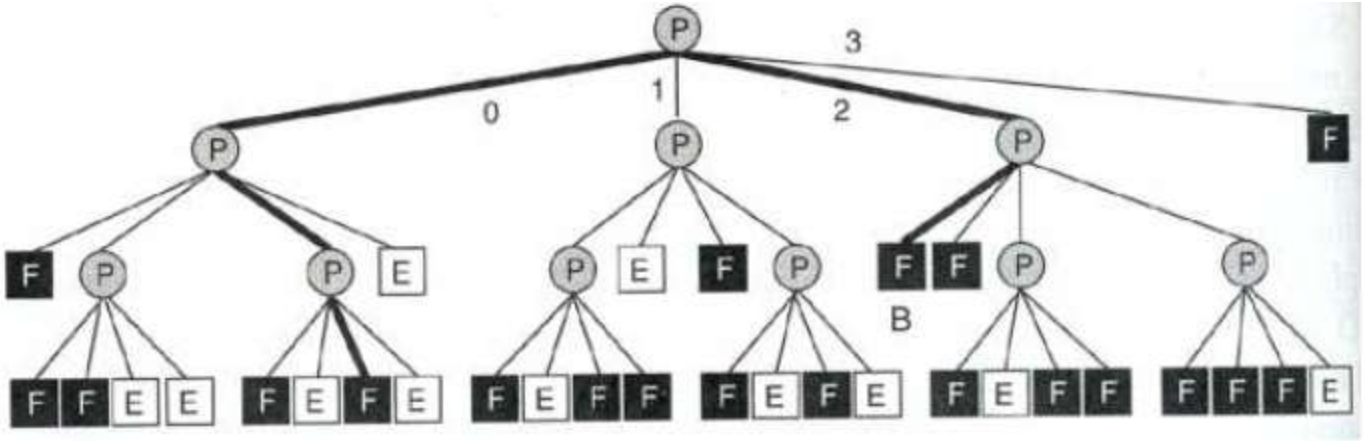


Source: FvDFH Figure 12.24

Octree Display

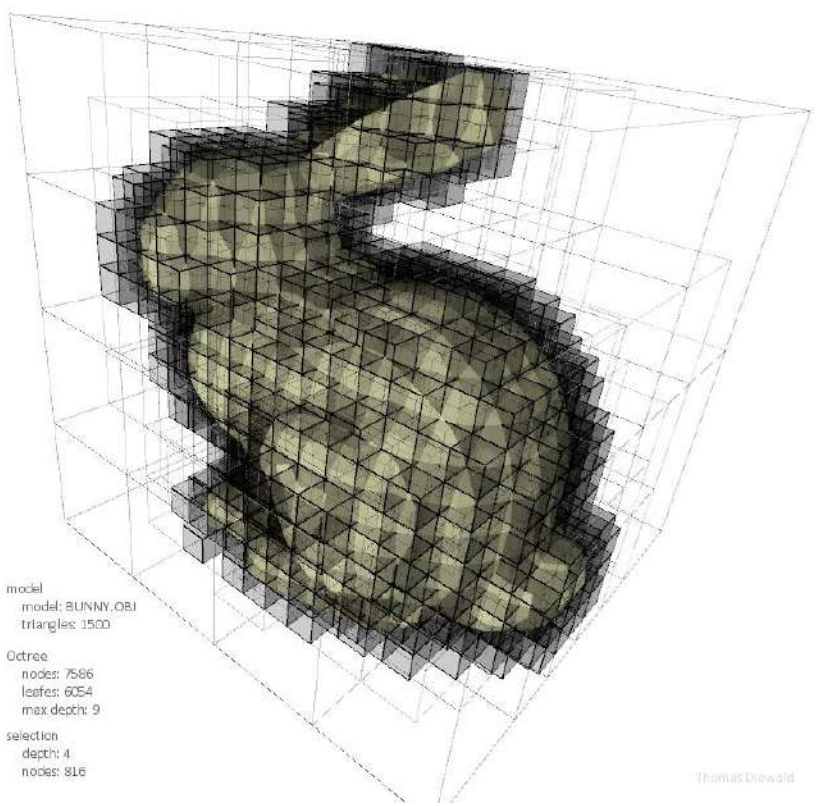
- Extend voxel methods
  - Slicing
  - Ray casting
- Finding neighbor cell requires traversal of hierarchy

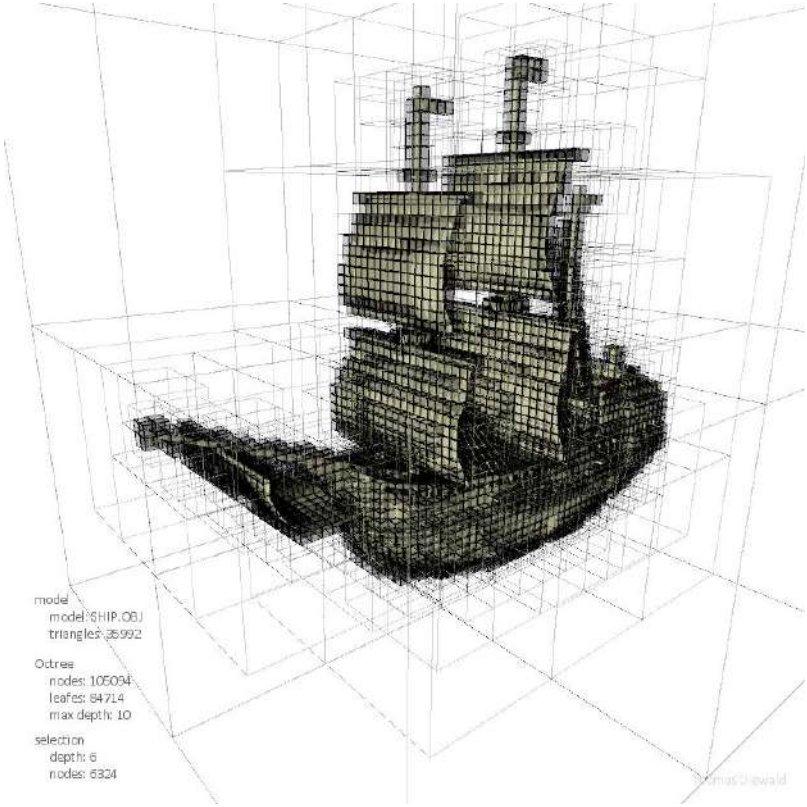




Source: FvDFH Figure 12.25

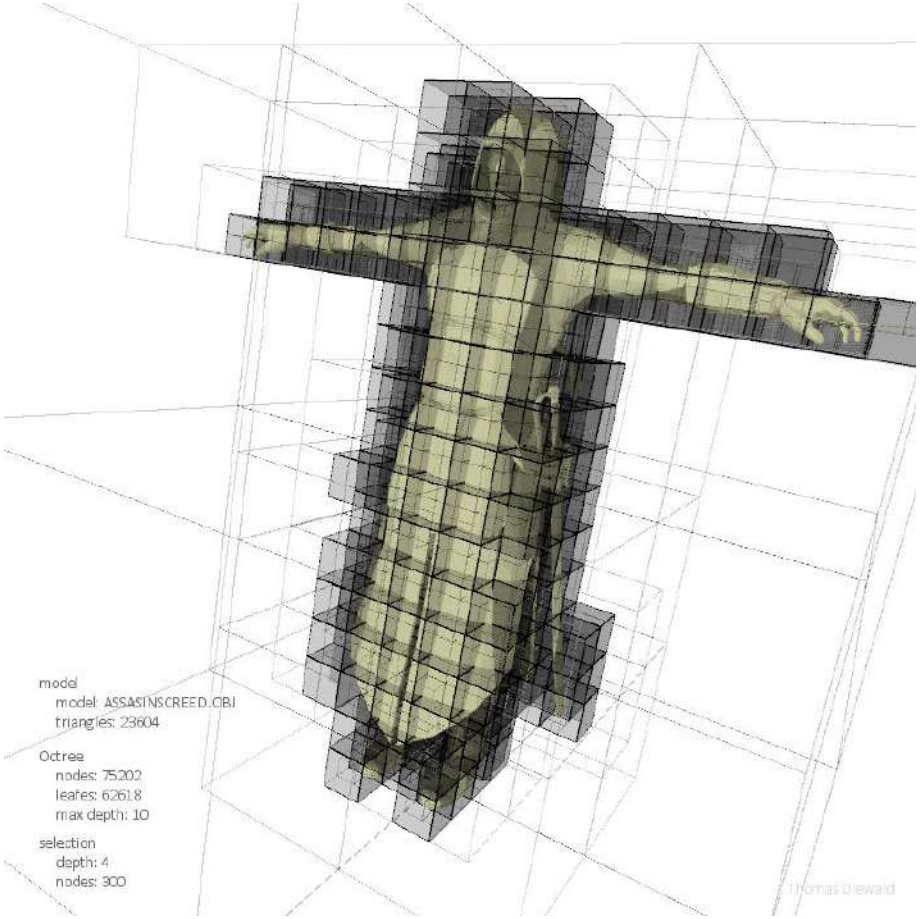
Octree Examples

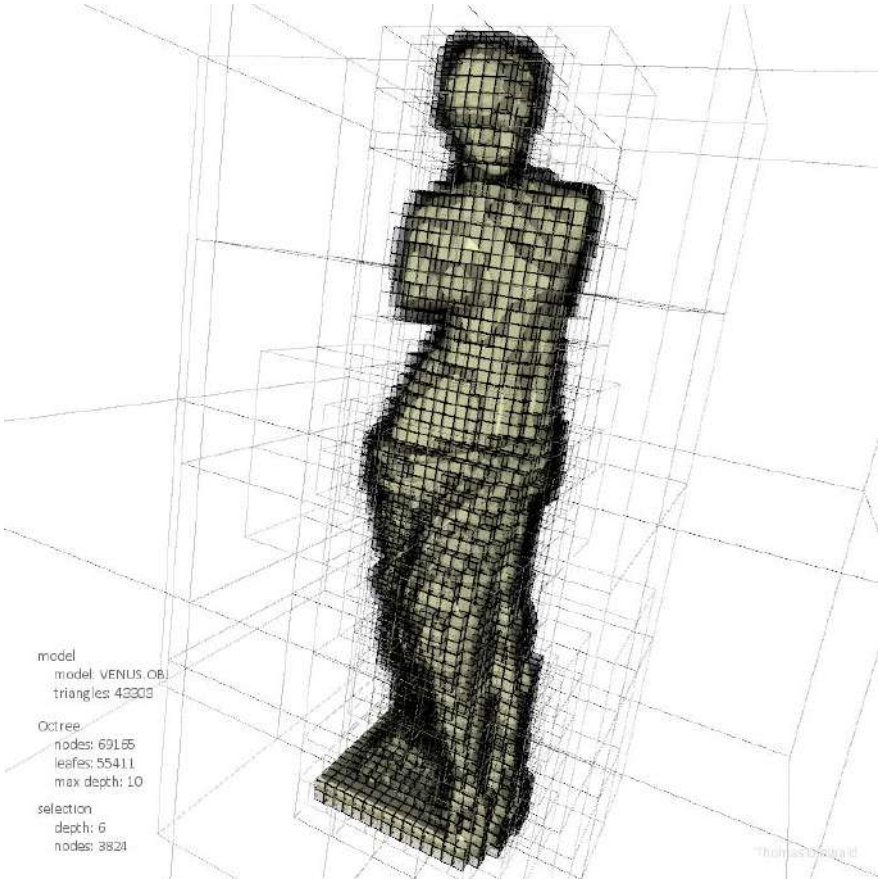




http://thomasdiewald.com

Octree Examples



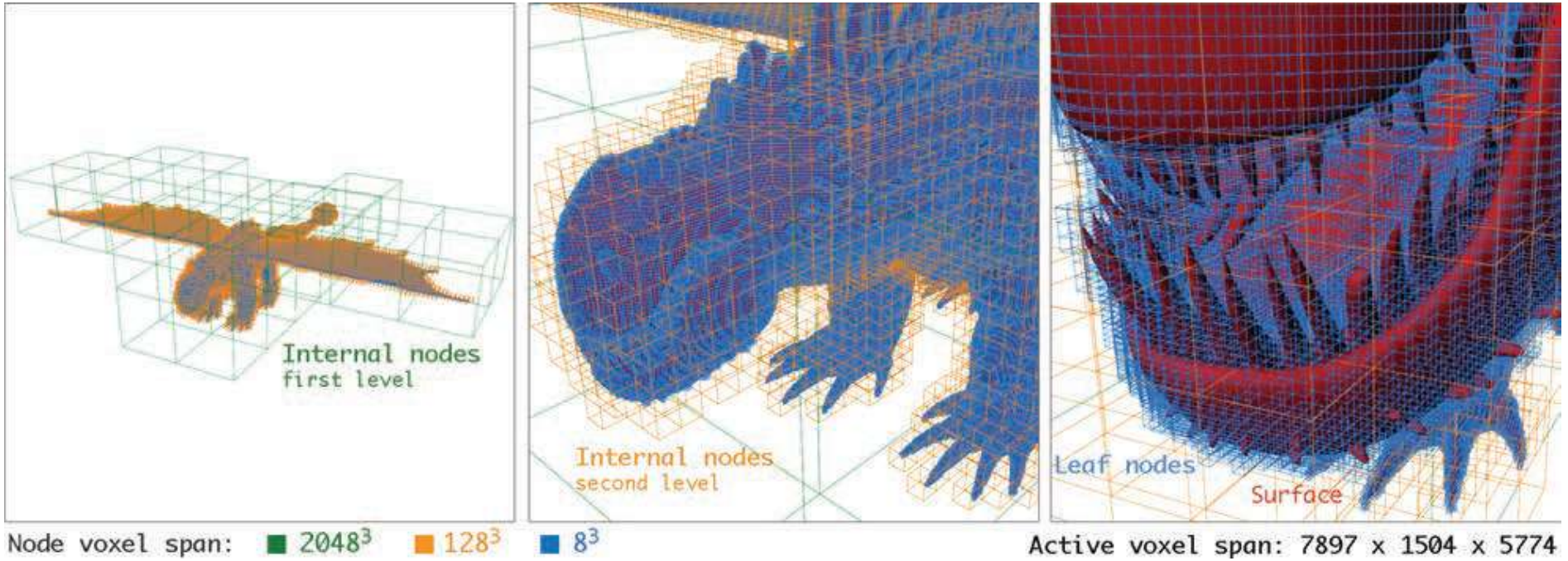


http://thomasdiewald.com

Another Voxel Representation: OpenVDB

http://www.openvdb.org

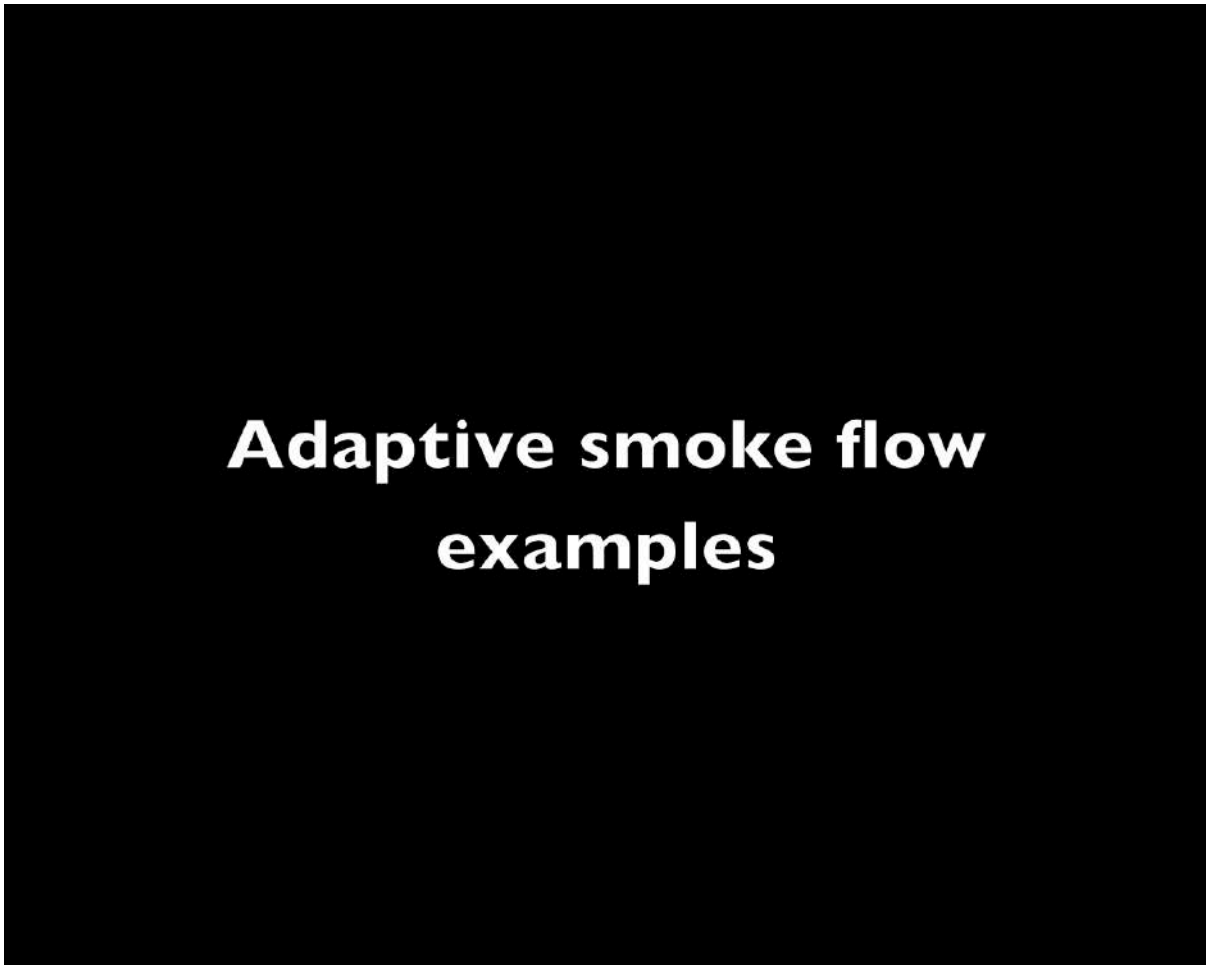
- Developed by DreamWorks Animation
  - Hierarchical, sparse representation for time-varying voxel data
  - Compact storage and fast data access



Source: Ken Museth

http://www.openvdb.org

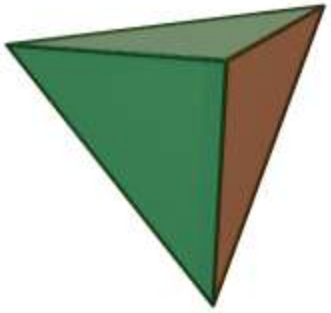
SPGrids



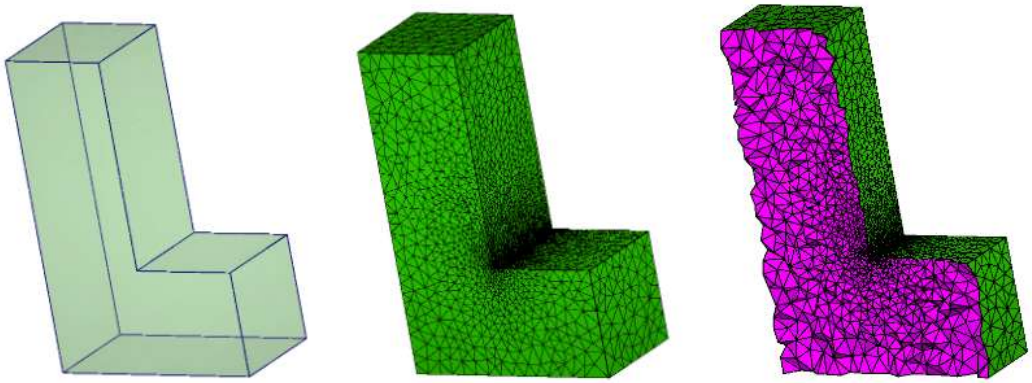
Source: **Setaluri et al.** http://pages.cs.wisc.edu/~sifakis/project_pages/SPGrid.html

Tetrahedra as Volume Representations

- Tetrahedron (Tet)
  - 4 vertices, 4 faces
- Tetrahedral mesh (Tet Mesh)
  - Similar to a standard mesh
    - A list of vertices
    - A list of tetrahedra



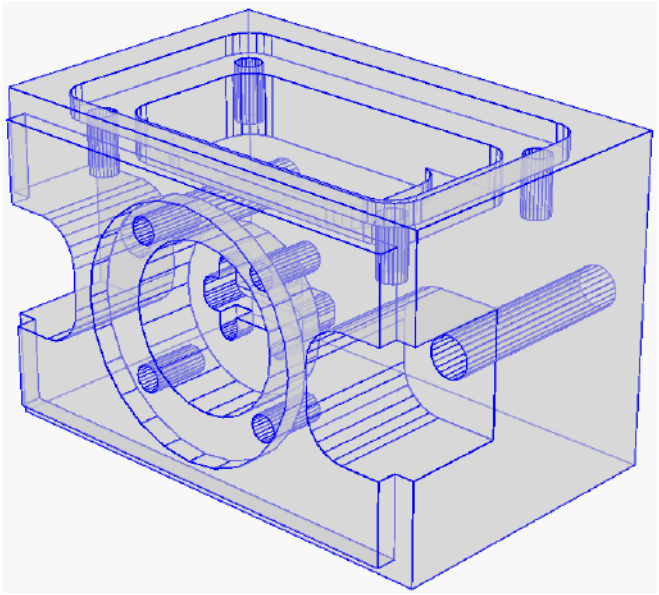
Source: Wikipedia



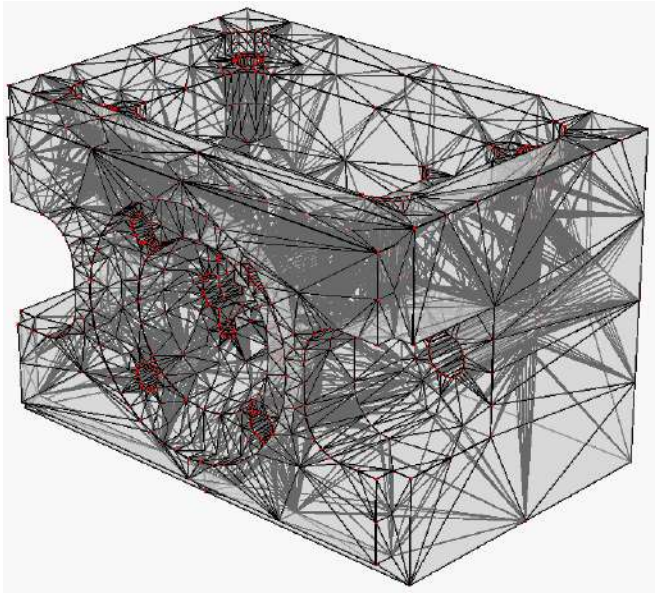
Source: Tetgen.org

Tetrahedralization

- Conversion from a surface representation to a tetrahedral mesh
  - Tetgen by Hang Si



Input Mesh

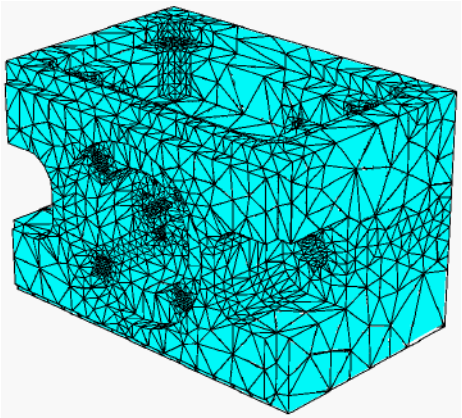


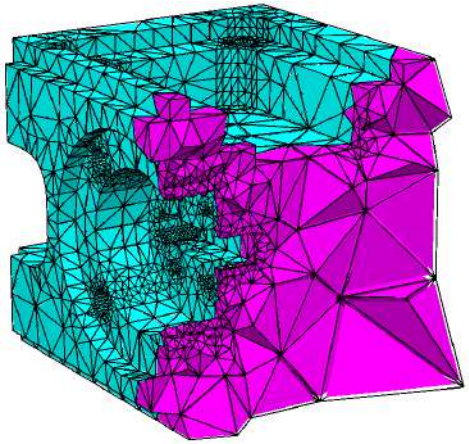
Constrained Delaunay Tetrahedralization

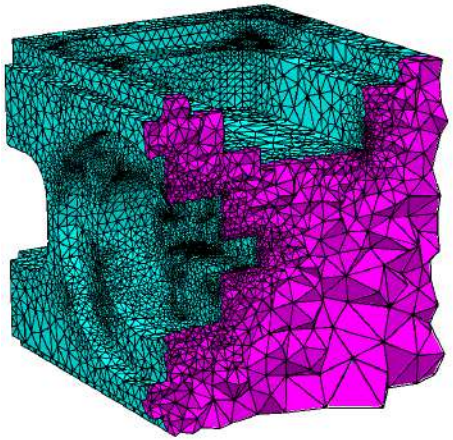
Source: Tetgen.org

Tetrahedralization

- Preserves mesh boundary
  - Boundary triangles and tet boundary coincide
- Knobs to refine subdivision
  - New vertices can be inserted
    - Guarantee tet quality (volume, angles)







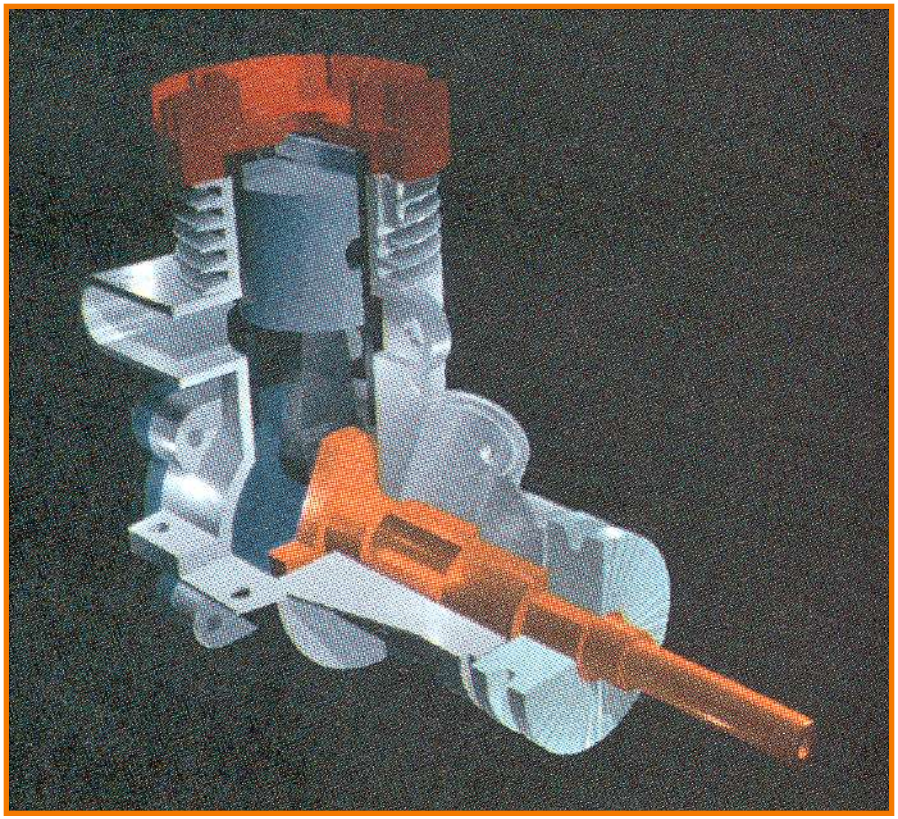
Source: Tetgen.org

Constructive Solid Geometry (CSG)

- A neat way to build complex objects from simple parts using Boolean operations
  - Very easy when ray tracing
  - Not so easy when not ray tracing

Constructive Solid Geometry (CSG) in CAD

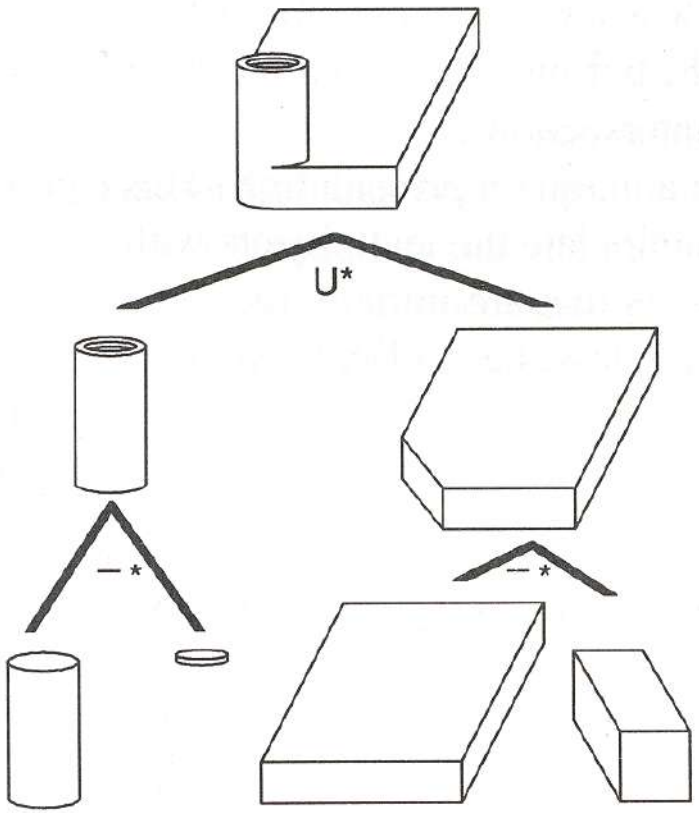
- Interactive modeling programs
  - Intuitive way to design objects



H&B Figure 9.9

Constructive Solid Geometry (CSG)

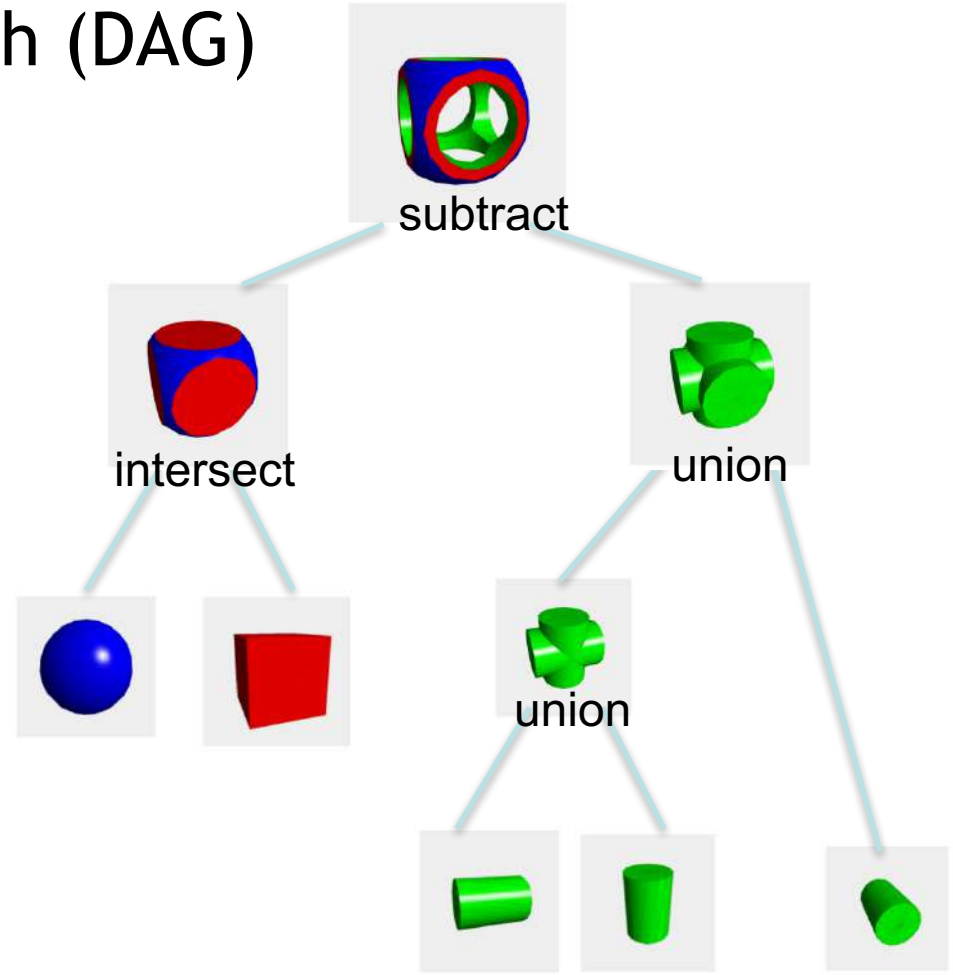
- Represent solid object as hierarchy of Boolean operations
- The Boolean operations are not evaluated, instead, objects are represented implicitly with a tree structure



FvDFH Figure 12.27

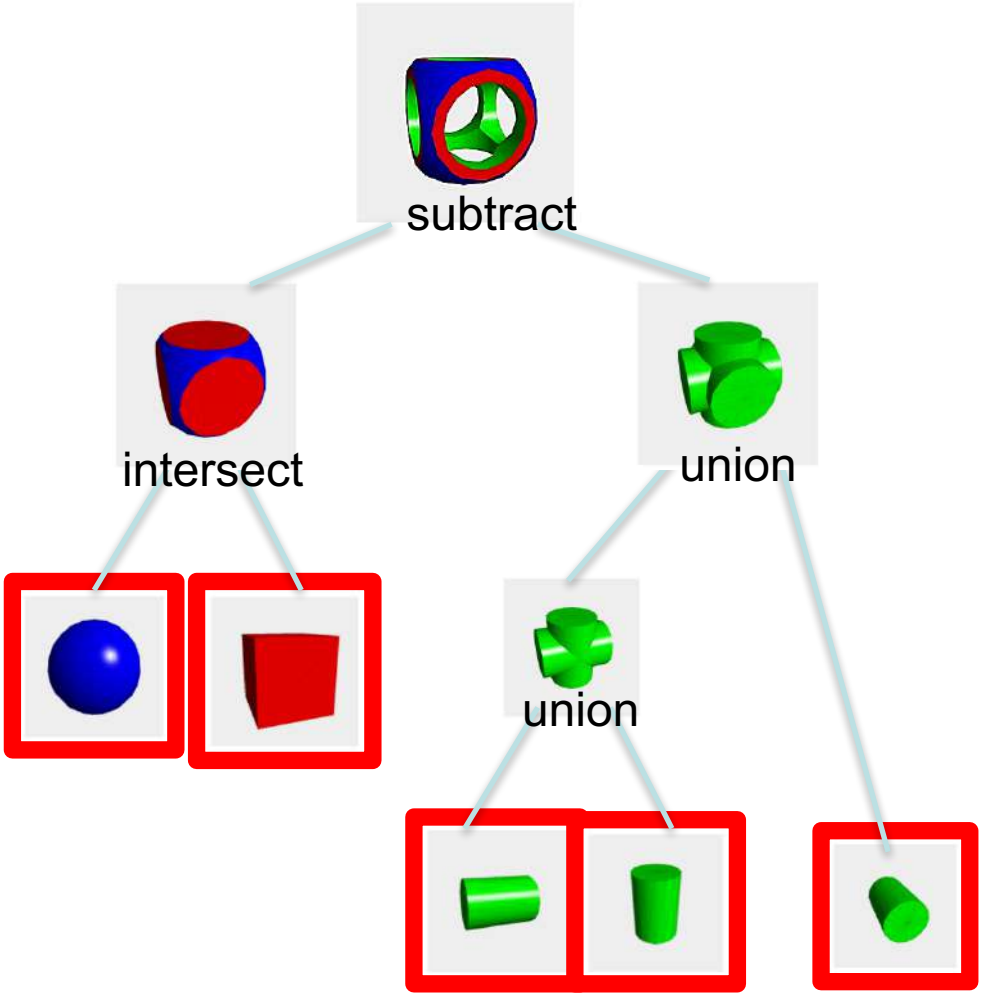
CSG Data Structure

- Binary Tree
- Directed Acyclic Graph (DAG)



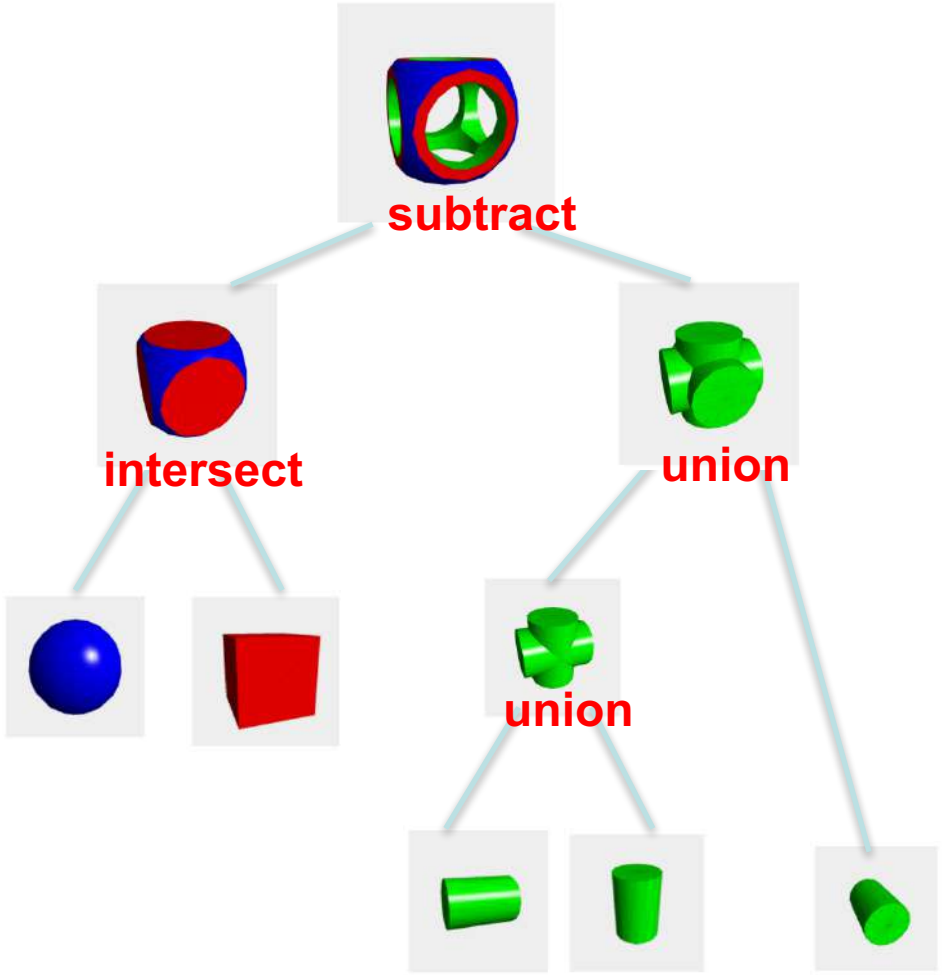
Leaves: CSG Primitives

- Simple shapes
  - Cuboids
  - Cylinders
  - Prisms
  - Pyramids
  - Spheres
  - Cones
- Extrusions
- Surfaces of revolution
- Swept surfaces

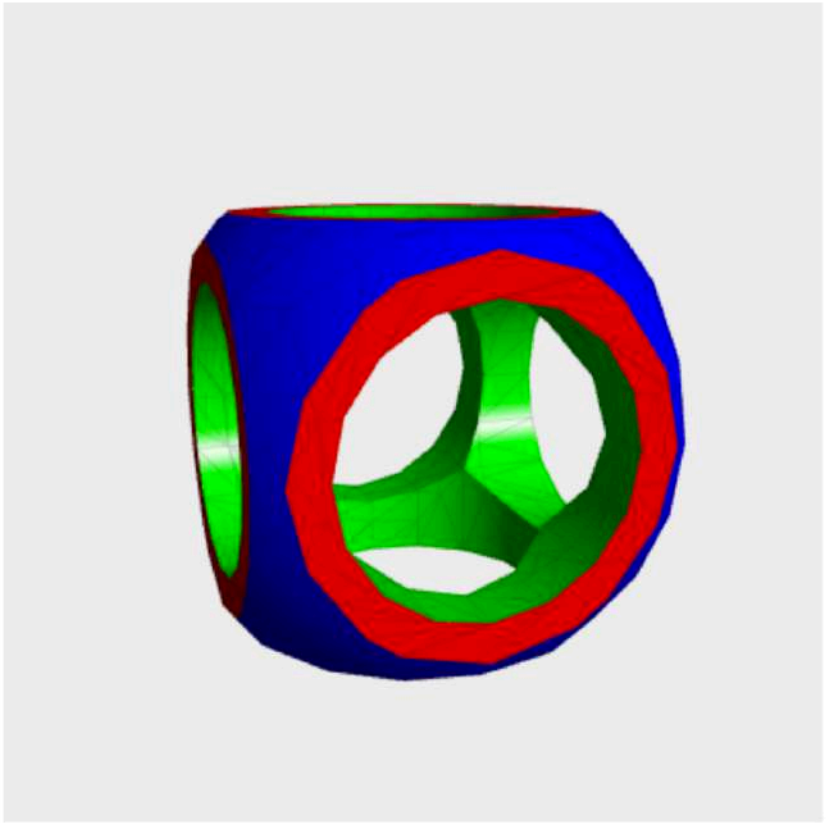


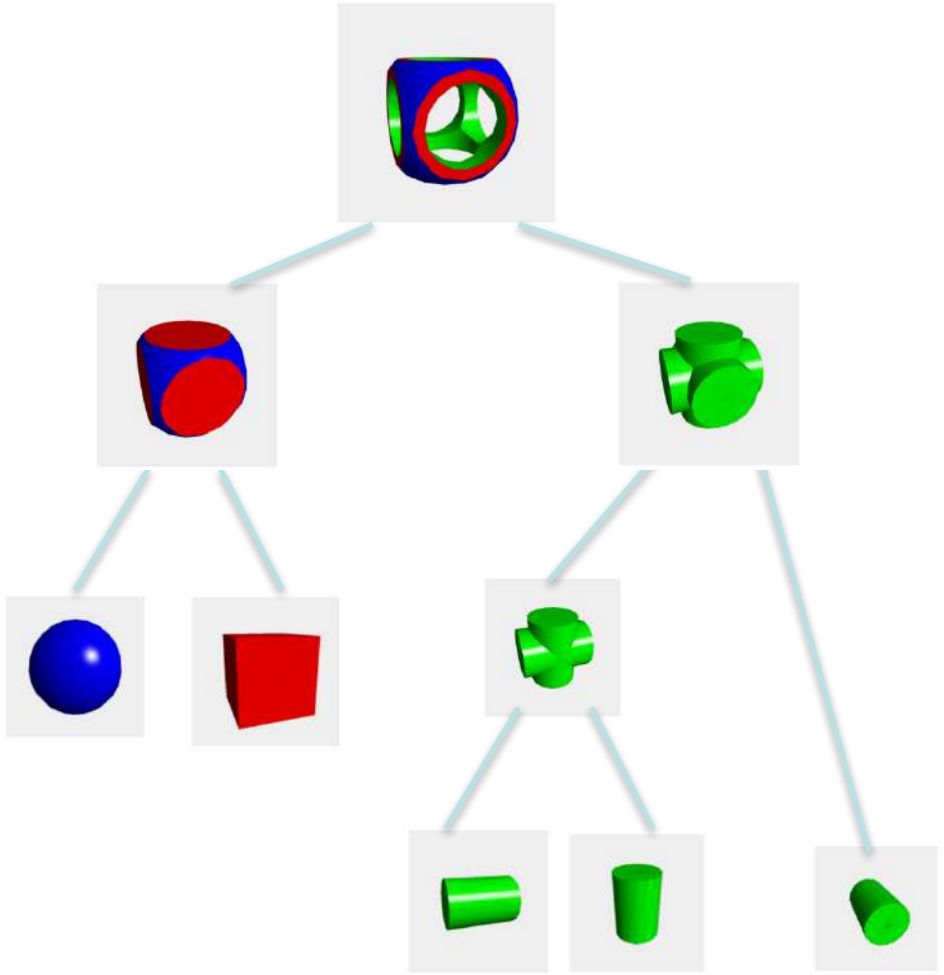
Internal Nodes

- Boolean Operations
  - Union
  - Intersection
  - Difference
- Rigid Transformations
  - Scale
  - Translation
  - Rotation
  - Shear



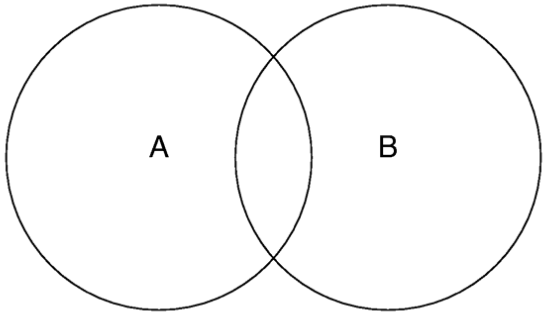
Root: The Final Object



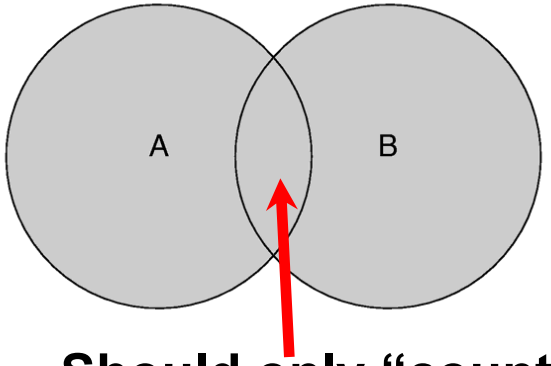


Booleans for Solids

Given overlapping shapes A and B:

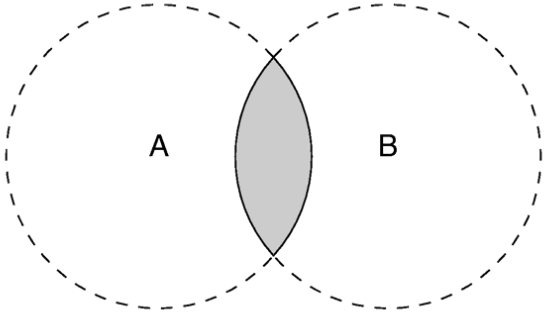


Union

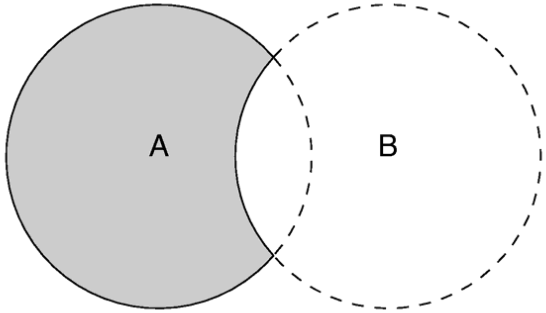


Should only "count" overlap region once!

Intersection

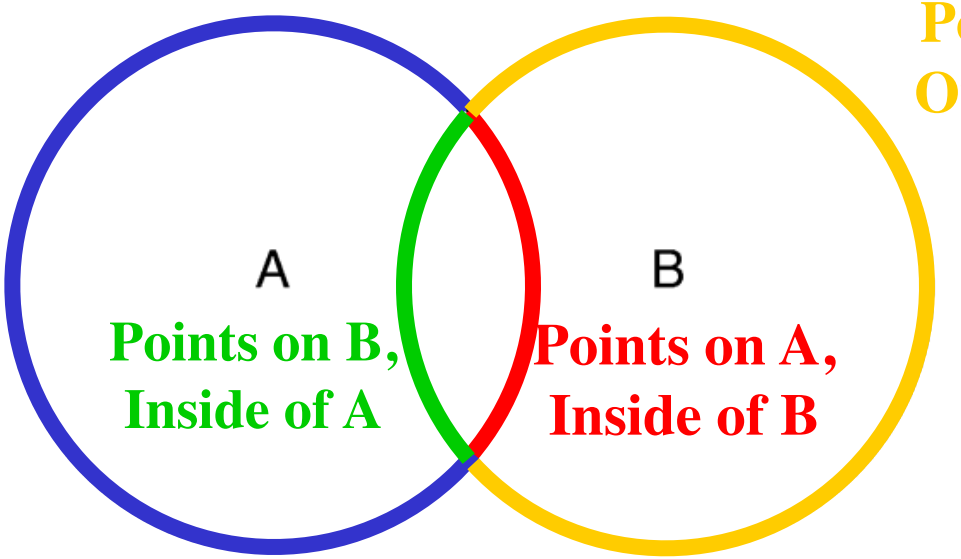


Subtraction

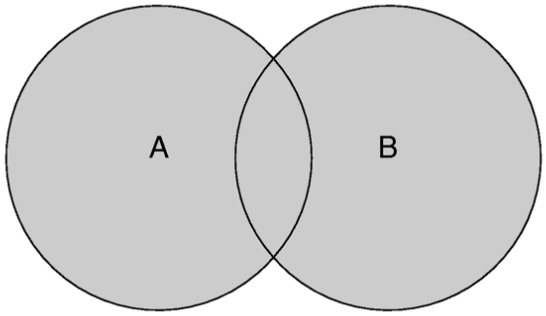


How Can We Compute CSG?

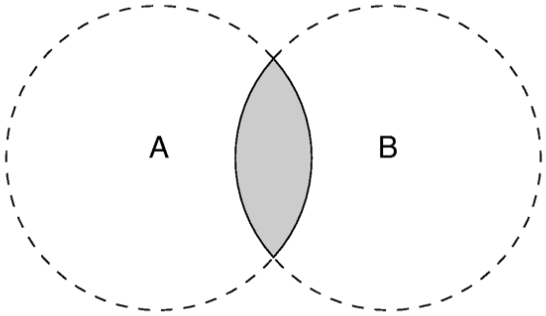
4 Cases



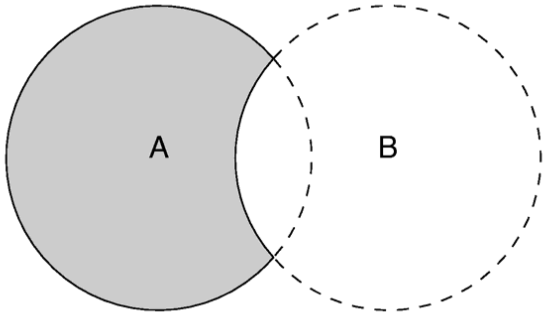
Union



Intersection

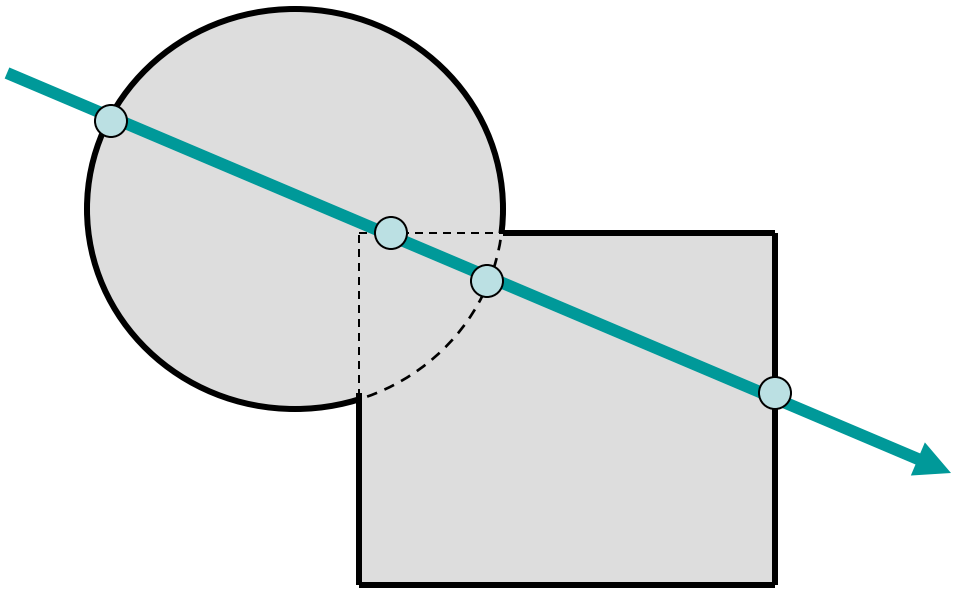


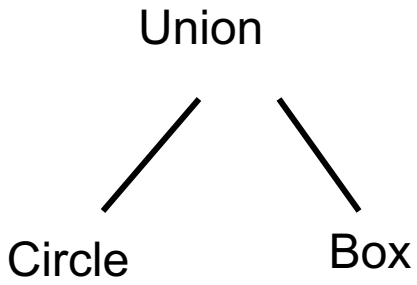
Subtraction



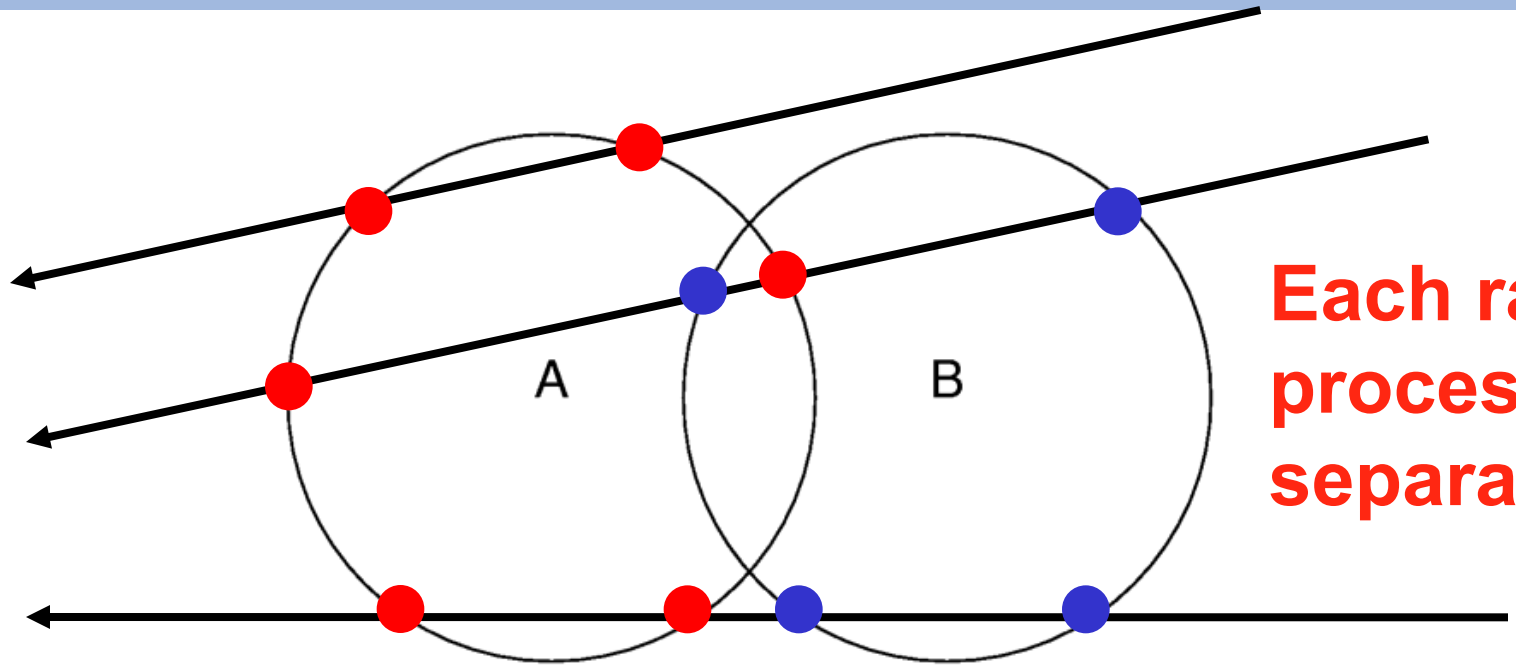
How Can We Render CSG?

- Use ray casting



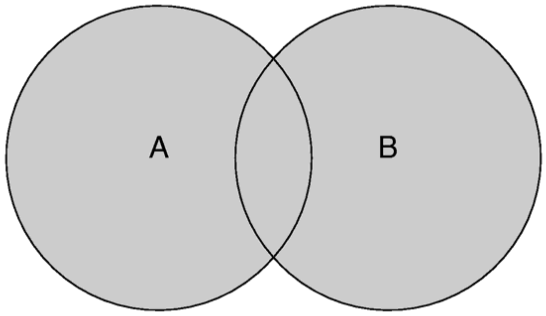


Ray Casting: Collect Intersections

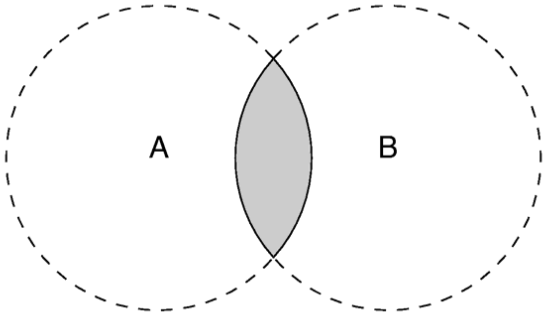


Each ray processed separately!

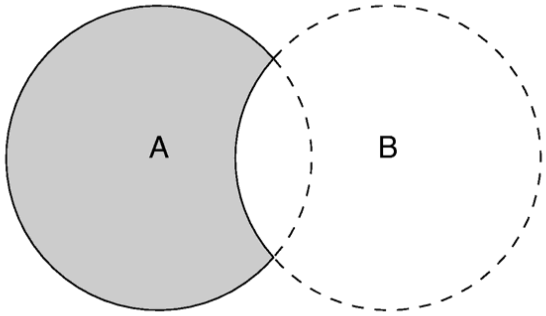
Union



Intersection

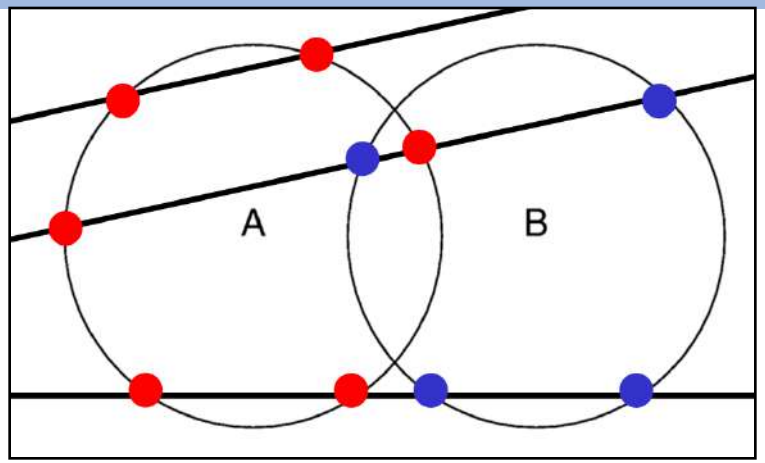


Subtraction



Ray Casting CSG

1. Test "inside" intersections:
   - Find intersections with A, test if they are inside/outside B
   - Find intersections with B, test if they are inside/outside A

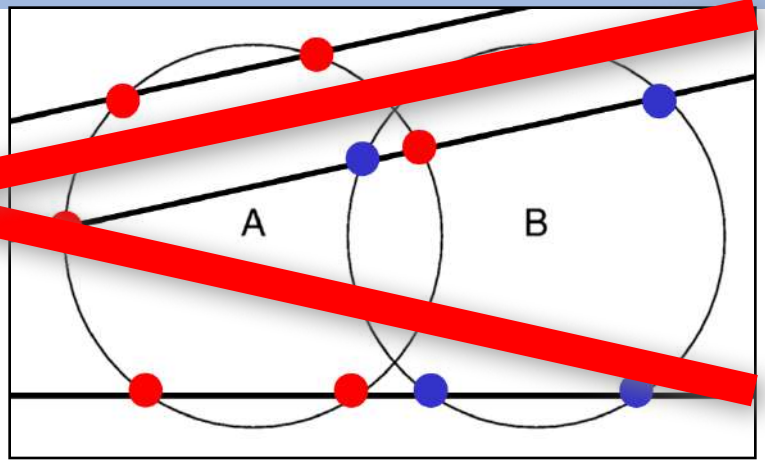


This would certainly work, but would need to determine if points are inside solids...

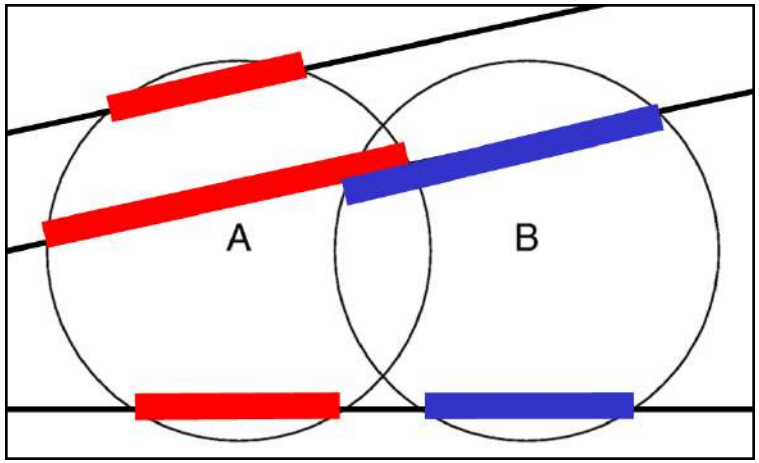
:-(

Ray Casting CSG

1. Test "inside" intersections:
   - Find intersections with A, test if they are inside/outside B
   - Find intersections with B, test if they are inside/outside A



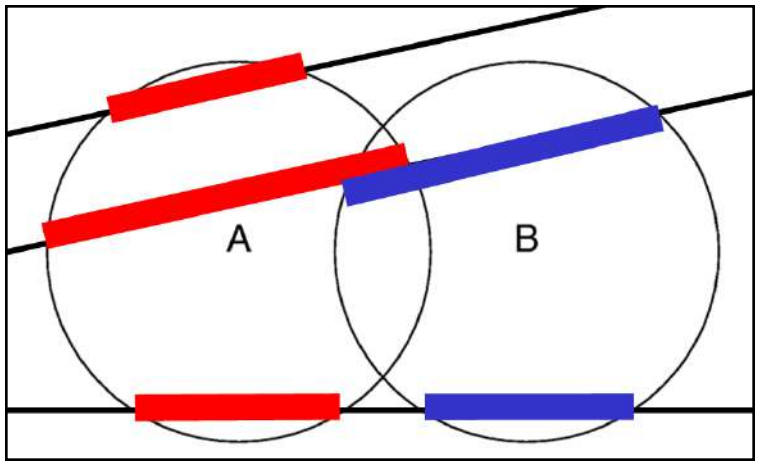
2. Overlapping intervals:
   - Find the intervals of "inside" along the ray for A and B
   - How? Just keep an “entry” / “exit” bit for each intersection
     - Easy to determine from intersection normal and ray direction
   - Compute union/ intersection/ subtraction of the intervals



Ray Casting CSG

Problem reduces to 1D for each ray

2. Overlapping intervals:
   - Find the intervals of "inside" along the ray for A and B
   - How? Just keep an “entry” / “exit” bit for each intersection
     - Easy to determine from intersection normal and ray direction
   - Compute union/ intersection/ subtraction of the intervals

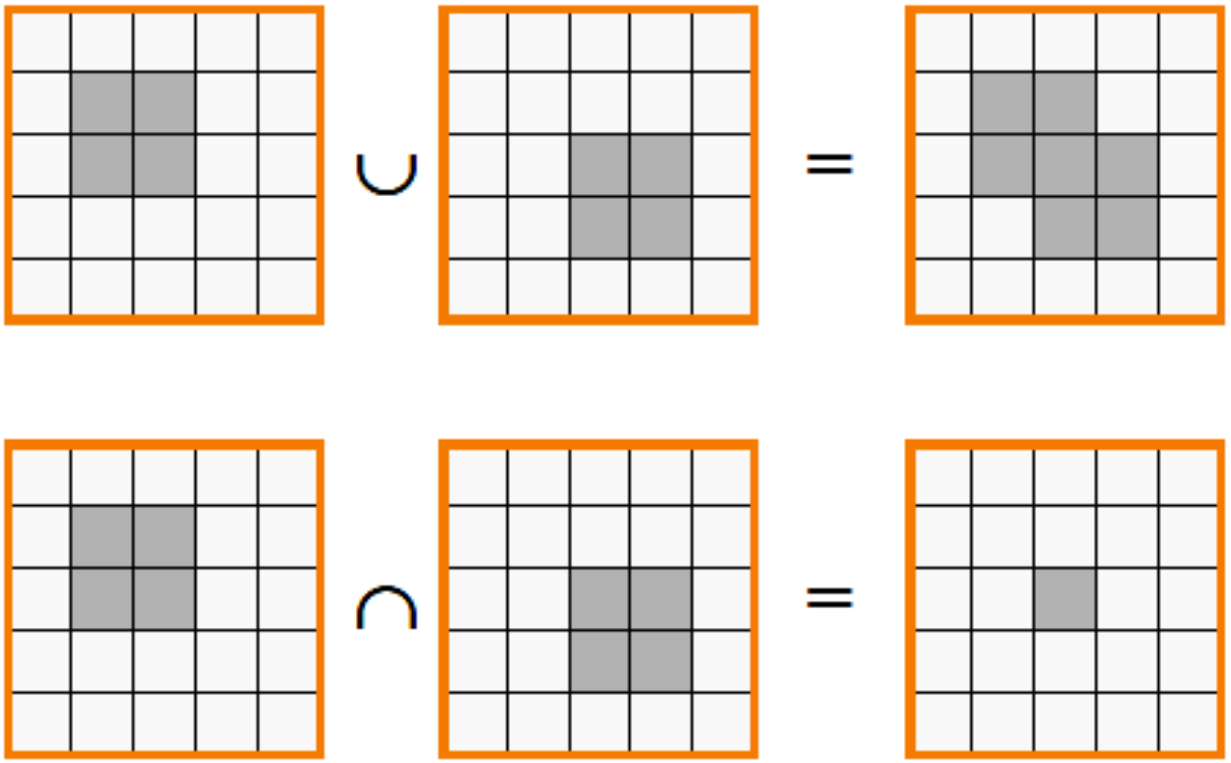


Rendering CSG is Easy with Ray Casting...

- ...but **very hard** if you actually try to compute an explicit representation of the resulting surface as a triangle mesh
- In principle very simple,
  but floating point numbers are not exact
  - E.g., points do not lie exactly on planes...
  - Computing the intersection A vs B is not necessarily the same as B vs A...
  - The line that results from intersecting two planes does not necessarily lie on either plane...
  - etc., etc.

How Can We Implement Boolean Operations?

- Use voxels/octrees/ADFs
  - We can convert from b-reps to voxels and back
  - Compare objects voxel by voxel

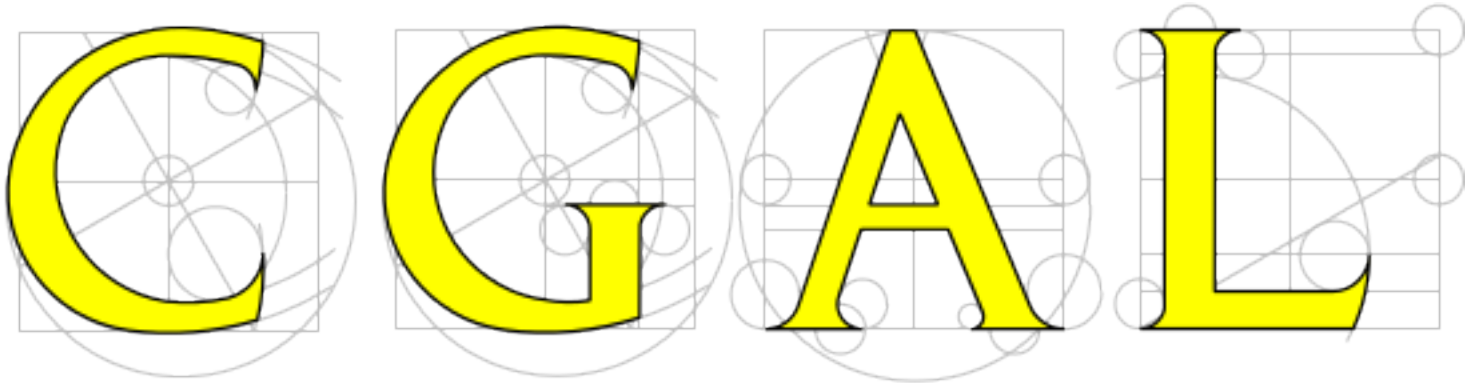


How Can We Implement Boolean Operations?

- The hard way ...
- We will not be asking you to do this
- Commercial libraries
  - e.g., Parasolid
- Open source libraries
  - e.g., CGAL

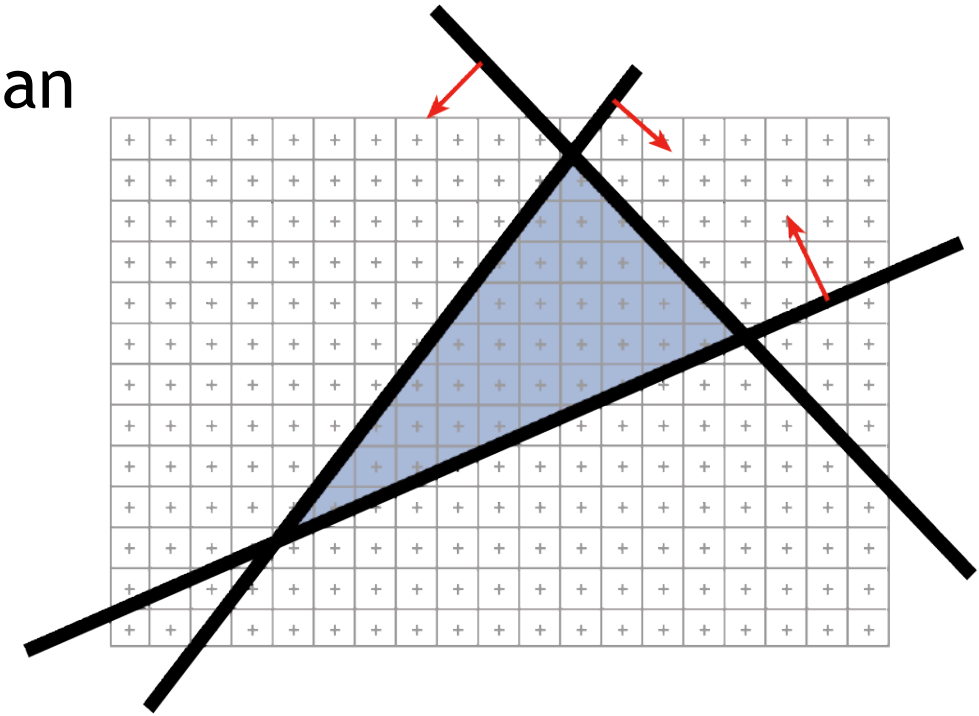
Computational Geometry Algorithms Library

- CGAL is open source
- Data structures and algorithms
  - Triangulations
  - Voronoi diagrams
  - **Boolean operations**
  - Mesh generation
  - Geometry processing
  - Search structures, …



Boolean Operations in CGAL

- Based on Nef Polyhedra
- Nef Polyhedron:
  - generated from a finite number of halfspaces by complement and intersection operation $AX+BY+CZ + D > 0$
  - union and difference can be generated from intersection and complement



Boolean Operations in CGAL

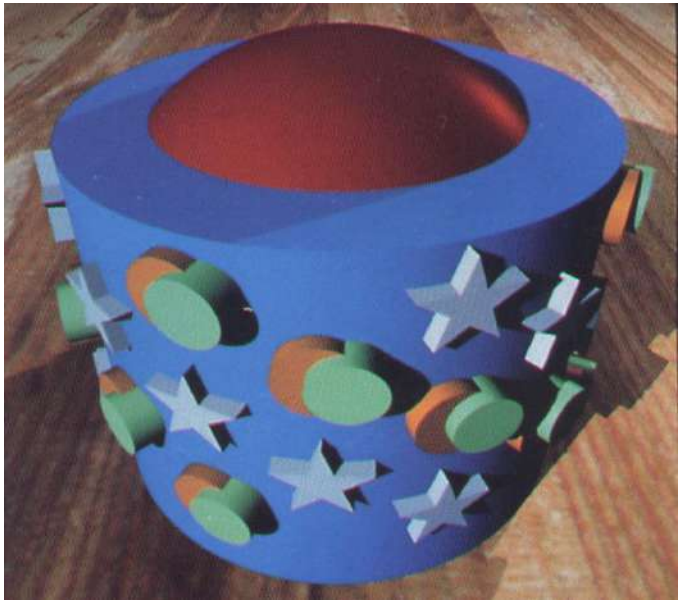
- Evaluate a CSG-tree with halfspaces as primitives and convert it into a B-rep representation

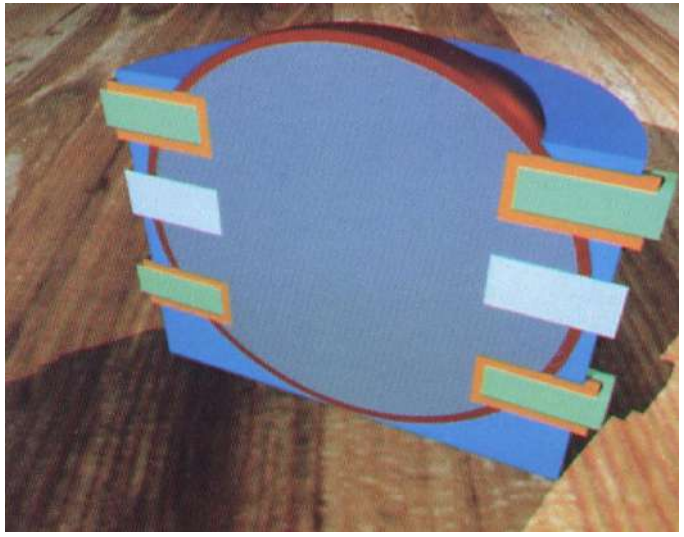
```
Nef_polyhedron N1(Plane_3( 1, 0, 0,-1));
Nef_polyhedron N2(Plane_3(-1, 0, 0,-1));
Nef_polyhedron N3(Plane_3( 0, 1, 0,-1));
Nef_polyhedron N4(Plane_3( 0,-1, 0,-1));
Nef_polyhedron N5(Plane_3( 0, 0, 1,-1));
Nef_polyhedron N6(Plane_3( 0, 0,-1,-1));

Nef_polyhedron I1(!N1 + !N2);  // open slice in yz-plane
Nef_polyhedron I2(N3 - !N4);   // closed slice in xz-plane
Nef_polyhedron I3(N5 ^ N6);    // open slice in yz-plane
Nef_polyhedron Cube1(I2 * !I1);
Cube1 *= !I3;
Nef_polyhedron Cube2 = N1 * N2 * N3 * N4 * N5 * N6;
```

http://doc.cgal.org/latest/Nef_3/index.html

CSG Examples





The Plan For Today

- Constructive Solid Geometry (CSG)
- Procedural Modeling
- OpenSCAD

Procedural Modeling

- Goal:
  - Describe 3D models algorithmically
- Best for models resulting from ...
  - Repeating processes
  - Self-similar processes
  - Random processes
- Advantages:
  - Automatic generation
  - Concise representation
  - Parameterized classes of models

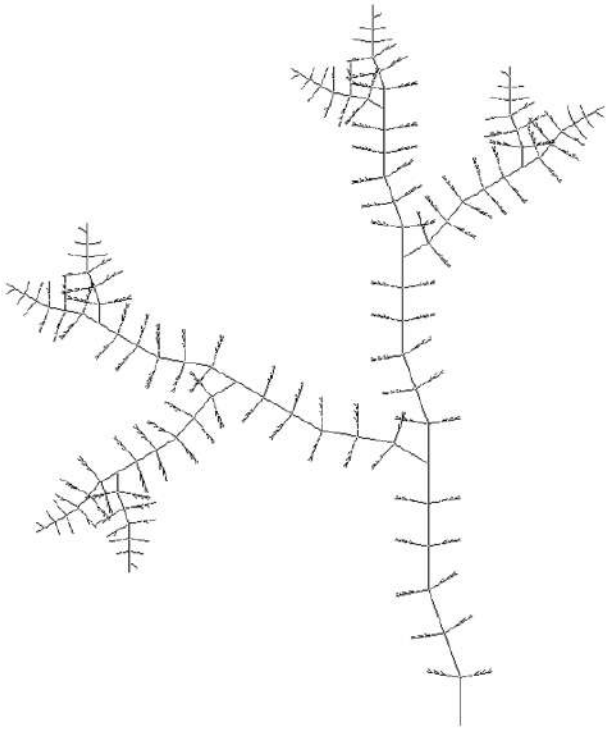
Formal Grammars and Languages

- A finite set of **nonterminal symbols: {S, A, B}**
- A finite set of **terminal symbols: {a, b}**
- A finite set of **production rules: S → AB**
- A **start symbol: S**

- Generates a set of finite-length sequences of symbols by recursively applying production rules starting with **S**

L-systems (Lindenmayer systems)

- A model of morphogenesis, based on formal grammars (set of rules and symbols)
- Introduced in 1968 by the Swedish biologist A. Lindenmayer
- Originally designed as a formal description of the development of simple multi-cellular organisms
- Later on, extended to describe higher plants and complex branching structures



L-system Example 1: Algae

- **nonterminals** : A B
- **terminals** : none
- **start** : A
- **rules** : (A → AB), (B → A)

n = 0 : A

n = 1 : AB

n = 2 : ABA

n = 3 : ABAAB

n = 4 : ABAABABA

n = 5 : ABAABABAABAAB

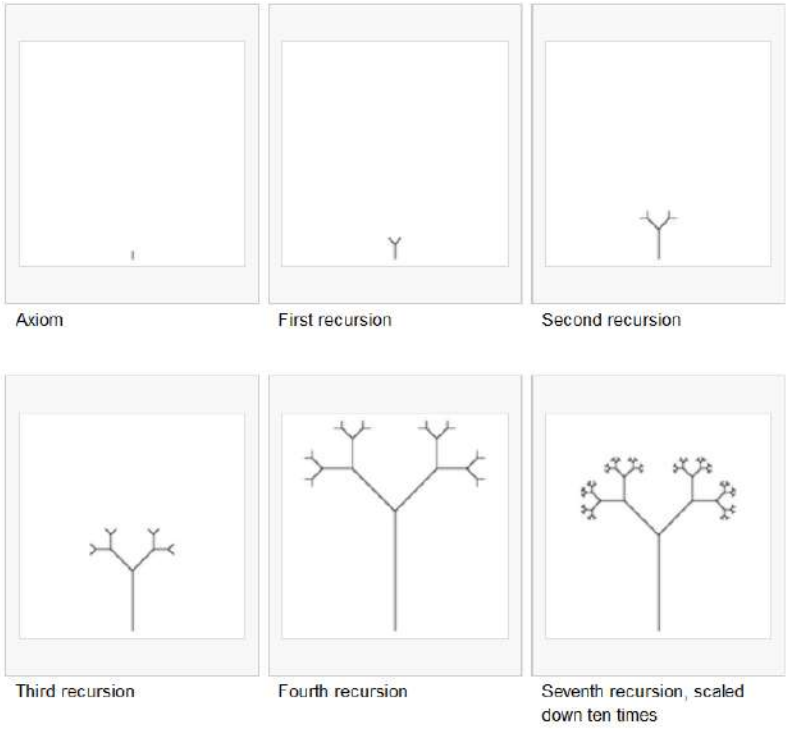
L-system Example 2

- **nonterminals** : 0, 1
- **terminals** : [,]
- **start** : 0
- **rules** : (1 → 11), (0 → 1[0]0)

start:	0
1st recursion:	1[0]0
2nd recursion:	11[1[0]0]1[0]0
3rd recursion:	1111[11[1[0]0]1[0]0]11[1[0]0]1[0]0

L-system Example 2

- **Visual representation: turtle graphics**
  - **0: draw a line segment ending in a leaf**
  - **1: draw a line segment**
  - **[: push position and angle, turn left 45 degrees**
  - **]: pop position and angle, turn right 45 degrees**



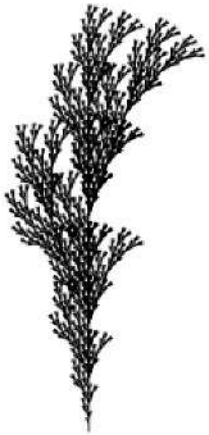
L-system Example 3: Fractal Plant

- **nonterminals** : X, F
- **terminals** : + - [ ]
- **start** : X
- **rules** : (X → F-[[X]+X]+F[+FX]-X), (F → FF)

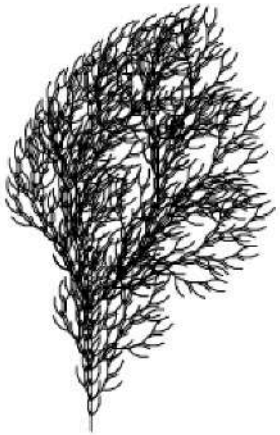
L-Systems Examples

- Tree examples

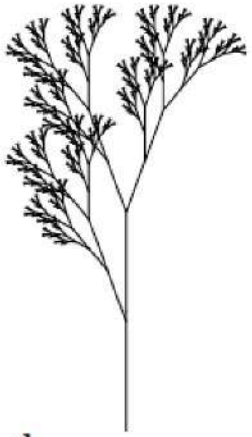
a
n=5, δ=25.7°
F
F→F[+F]F[-F]F



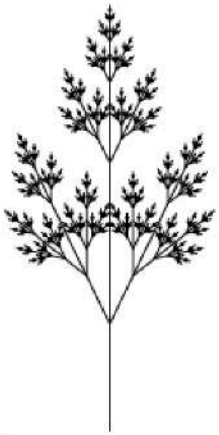
b
n=5, δ=20°
F
F →F[+F]F[-F][F]



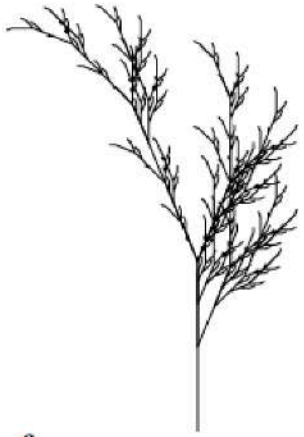
c
n=4, δ=22.5°
F
F→FF-[-F+F+F]+
[+F-F-F]



d
n=7, δ=20°
X
X →F[+X]F[-X]+X
F →FF



e
n=7, δ=25.7°
X
X→F[+X][-X]FX
F→FF



f
n=5, δ=22.5°
X
X→F-[[X]+X]+F[+FX]-X
F→FF

L-Systems Examples

Types of L-Systems

- *Deterministic*: If there is exactly one production for each symbol

  $0 \rightarrow 1[0]0$

- *Stochastic*: If there are several, and each is chosen with a certain probability during each iteration

  $0\ (0.5) \rightarrow 1[0]0$

  $0\ (0.5) \rightarrow 0$

Types of L-Systems

- *Context-free:* production rules refer only to an individual symbol
- *Context-sensitive:* the production rules apply to a particular symbol only if the symbol has certain neighbours

$$
\begin{aligned}
S &\rightarrow aSBC \\
S &\rightarrow aBC \\
CB &\rightarrow HB \\
HB &\rightarrow HC \\
HC &\rightarrow BC \\
aB &\rightarrow ab \\
bB &\rightarrow bb \\
bC &\rightarrow bc \\
cC &\rightarrow cc
\end{aligned}
$$

Types of L-Systems

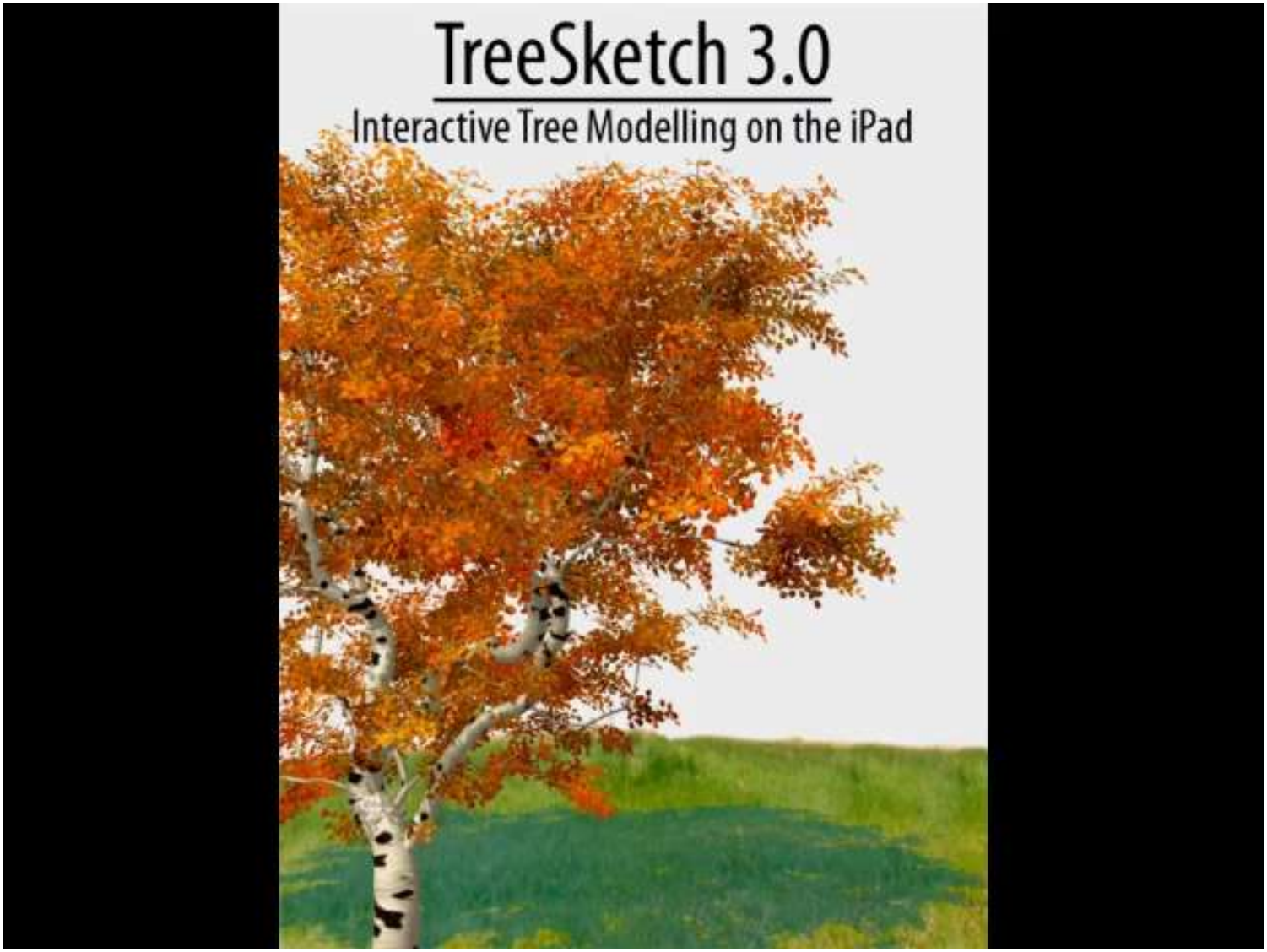
- *Nonparametric grammars:* no parameters associated with symbols
- *Parametric grammars:* symbols can have parameters
  - Parameters used in conditional rules
  - Production rules modify parameters

  - $A(x,y) : x = 0 \rightarrow A(1, y+1)B(2,3)$

Applications: Plant Modeling

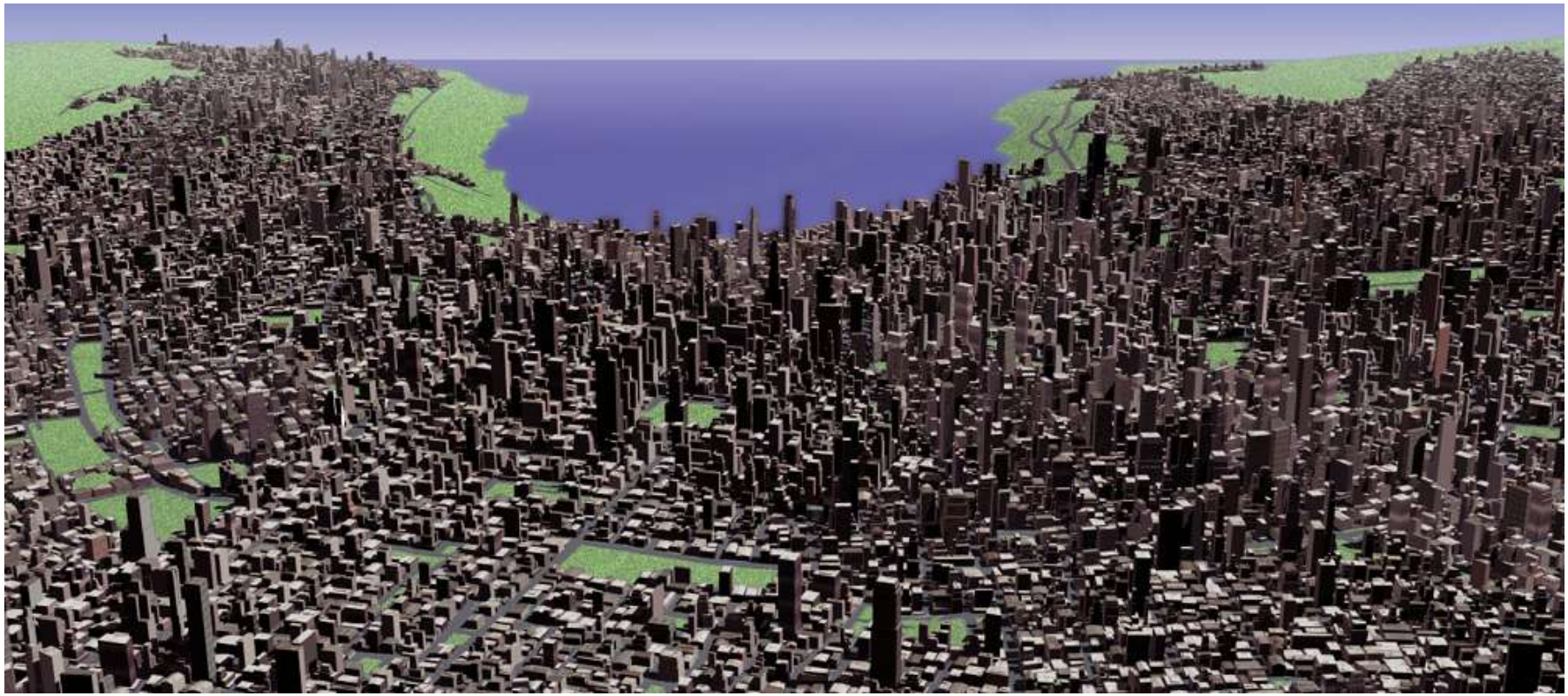
- Algorithmic Botany @ the University of Calgary
  - Covers many variants of L-Systems, formal derivations, and exhaustive coverage of different plant types.
  - http://algorithmicbotany.org/papers
  - http://algorithmicbotany.org/virtual_laboratory/

TreeSketch: Interactive Tree Modeling



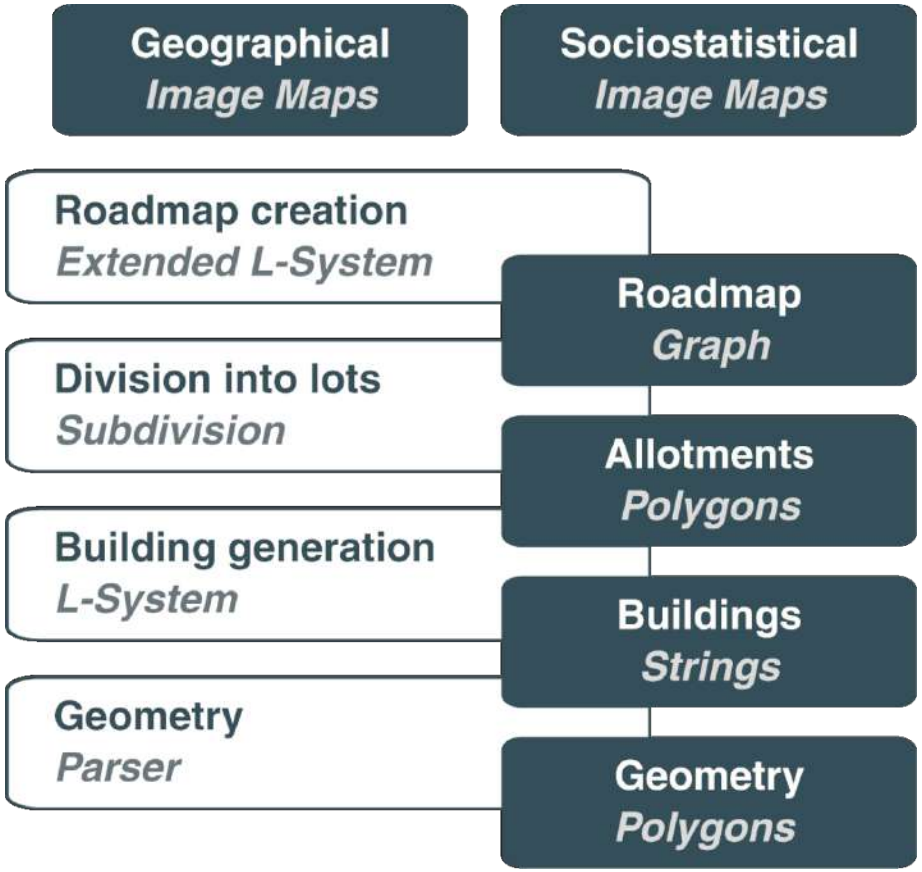
http://vimeo.com/68195050

Procedural Modeling of Cities



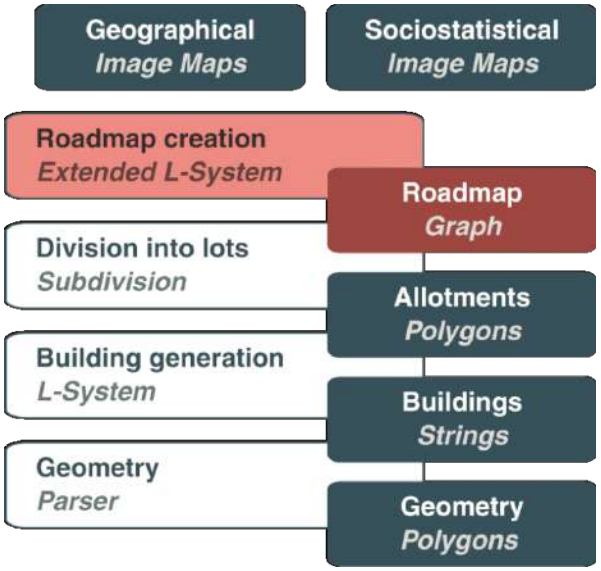
Procedural Modeling of Cities / Yoav Parish, Pascal Müller, Siggraph 2001

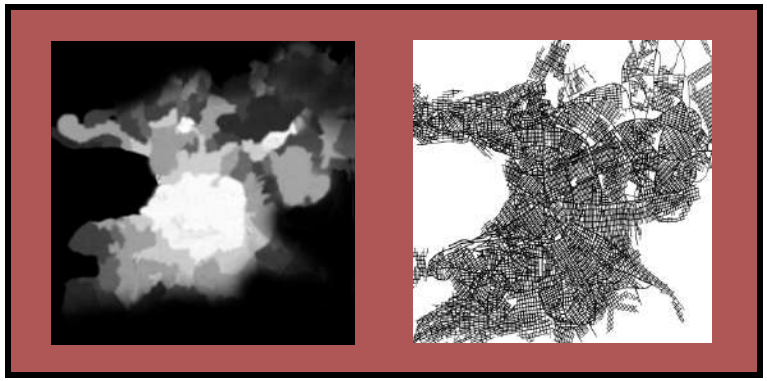
System Pipeline



Procedural Modeling of Cities / Yoav Parish, Pascal Müller, Siggraph 2001

Module 1: Streetmap Creation

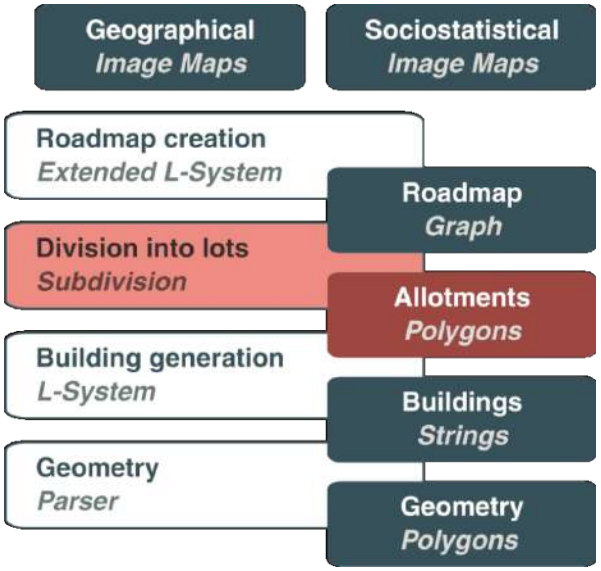


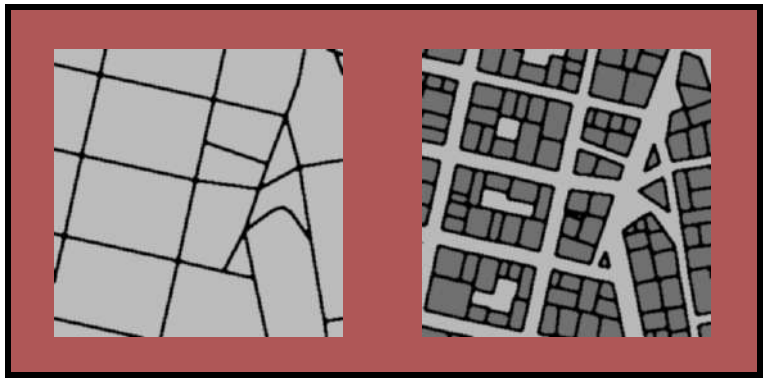


- Input:
  Image maps,
  parameters for rules
- Output:
  A street graph for
  interactive editing

Procedural Modeling of Cities / Yoav Parish, Pascal Müller, Siggraph 2001

Module 2: Division into Lots

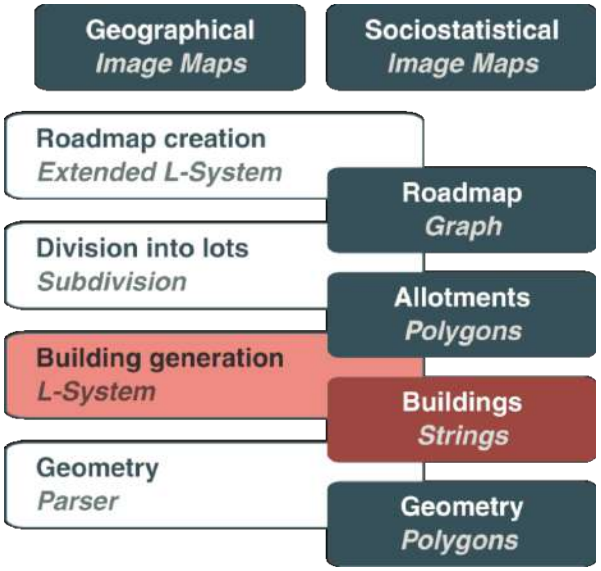


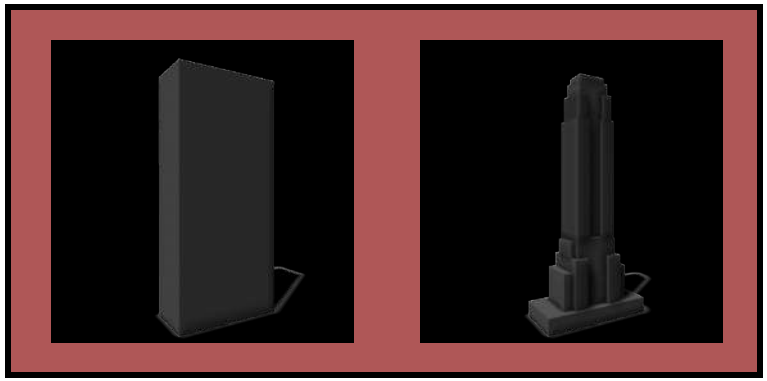


- Input:
  Street graph, area usage map
- Output:
  Polygon set of allotments for buildings

Procedural Modeling of Cities / Yoav Parish, Pascal Müller, Siggraph 2001

Module 3: Building Generation

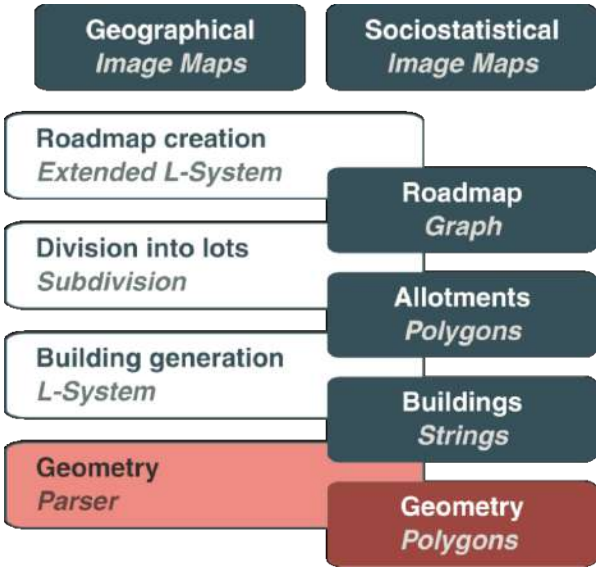


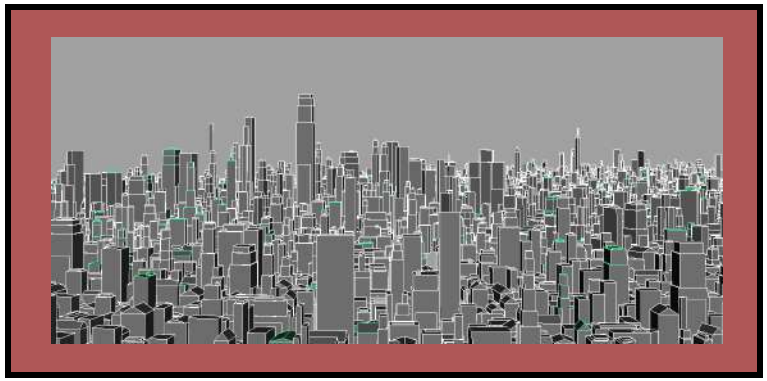


- Input:
  Lot polygons, age map and zone plan
- Output:
  Building strings with additional info

Procedural Modeling of Cities / Yoav Parish, Pascal Müller, Siggraph 2001

Module 4: Geometry and Facades





- Input:
  Strings and building type
- Output:
  City geometry and facade texture (procedural shader)

Procedural Modeling of Cities / Yoav Parish, Pascal Müller, Siggraph 2001

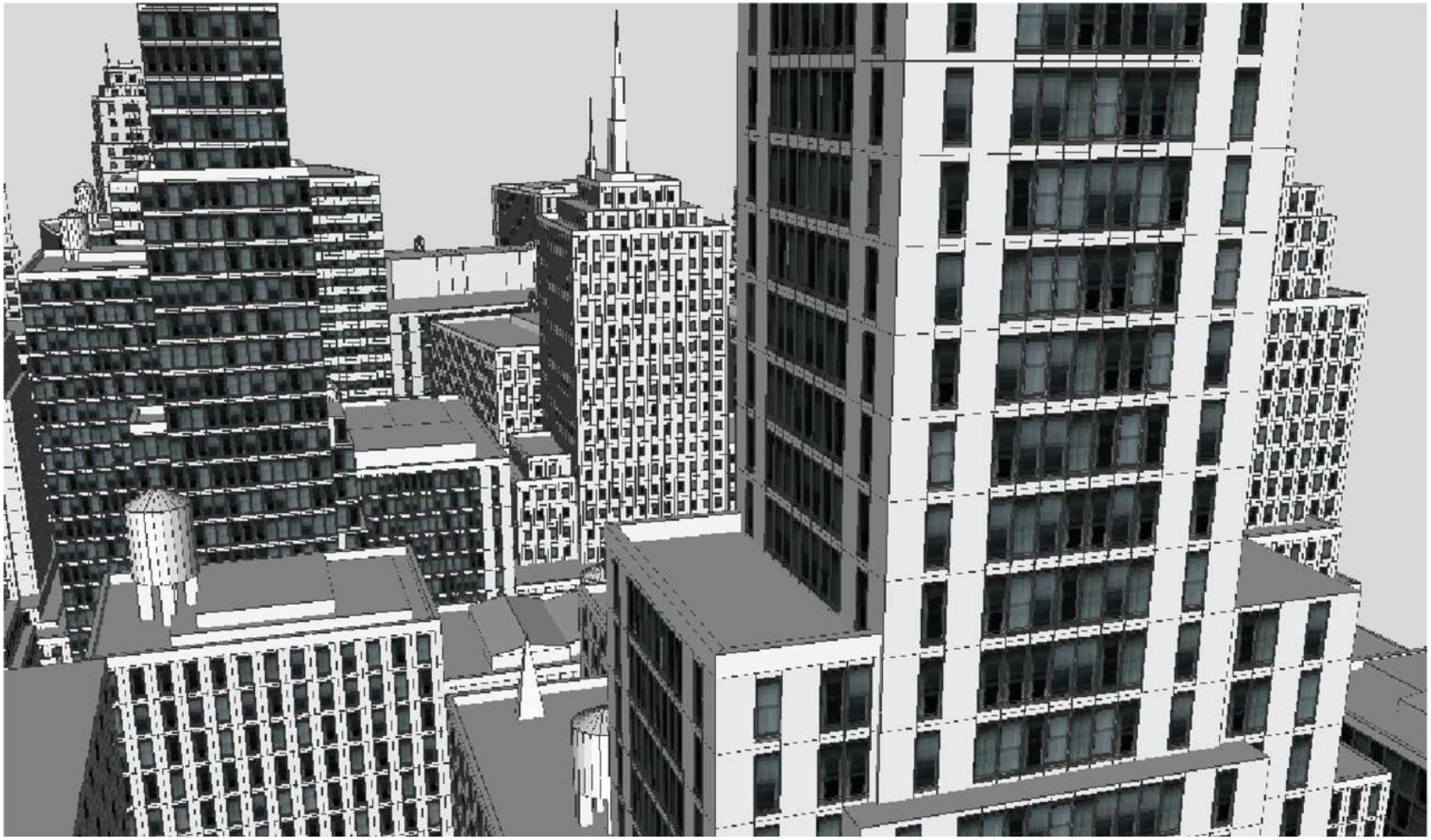
Procedural Modeling of Buildings

- Pompeii

Procedural Modeling of Buildings / Müller et al, Siggraph 2006

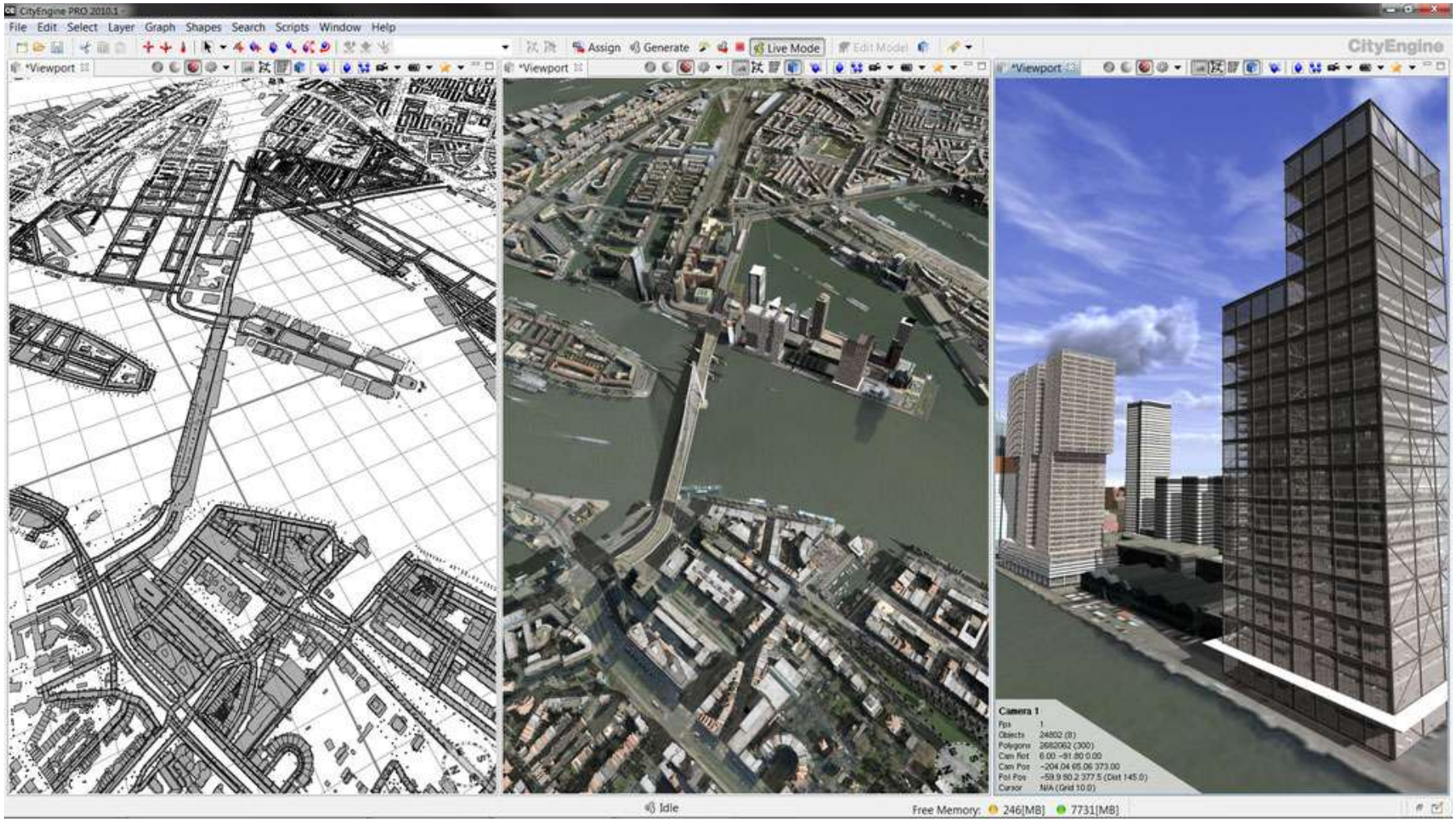
Procedural Modeling of Buildings

- Modern architecture



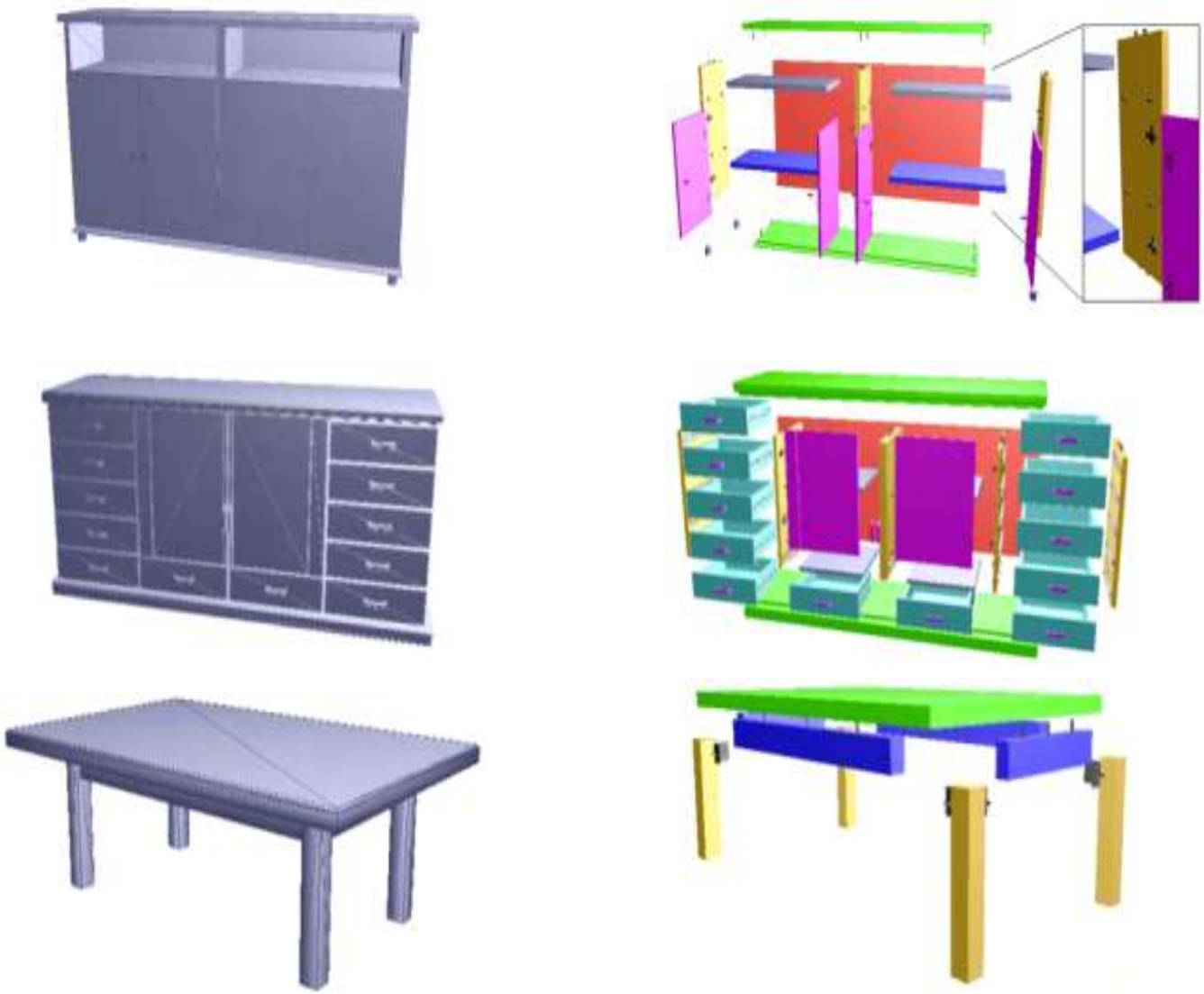
Procedural Modeling of Buildings / Müller et al, Siggraph 2006

CityEngine



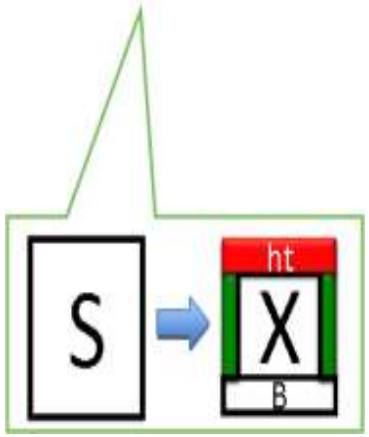
http://www.esri.com/software/cityengine/

Furniture Design using Formal Grammar

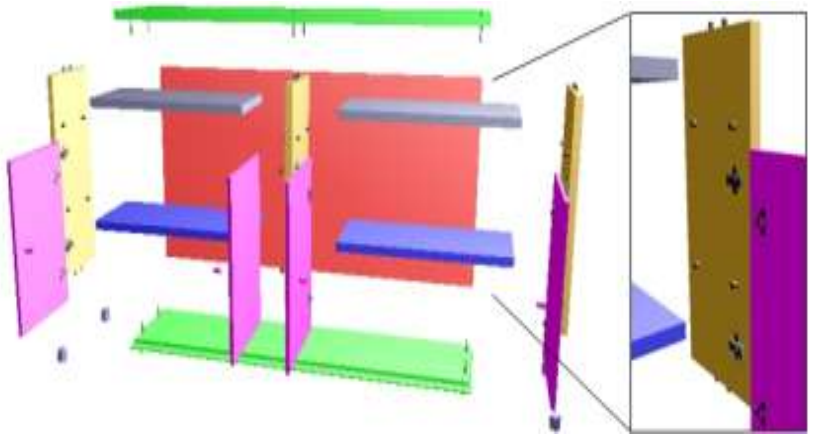


Converting 3D Furniture Models to Fabricable Parts and Connectors, Lau et al., Siggraph 2011

Furniture Design using Formal Grammar



Formal grammar



Separate parts and connectors

Formal Grammar for 2D Cabinets

$N = \{S, B, X, Y\}$

Non-terminal Symbols
- Collection of Parts

$\sum = \{hb, ht, v, ha, leg, wheel\}$

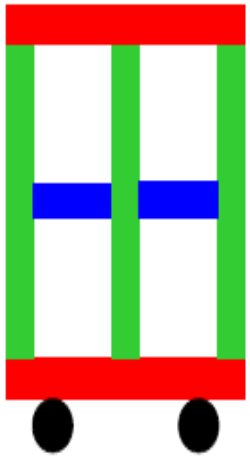
Terminal Symbols
- Separate Parts

P : Set of Production Rules

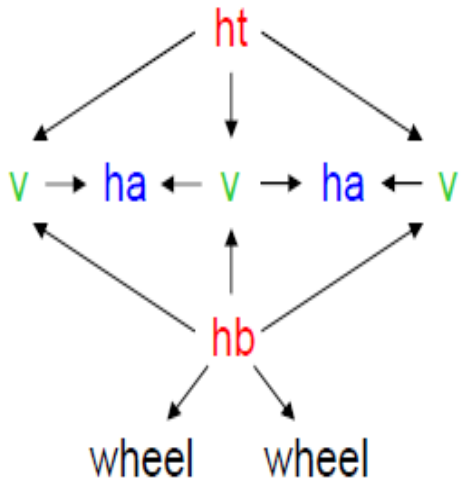
S : Start Symbol

The language specifies a directed graph,
and each graph represents parts and connectors

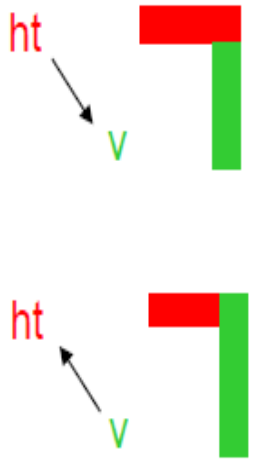
Representation of 2D Cabinets



Example 2D Cabinet



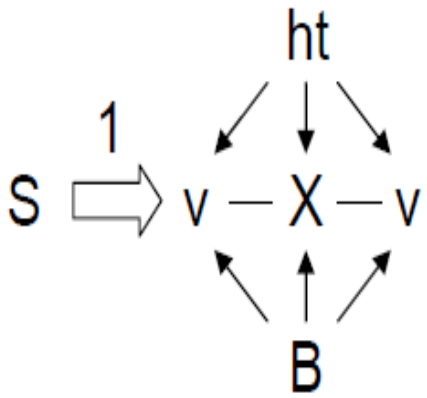
Corresponding Graph

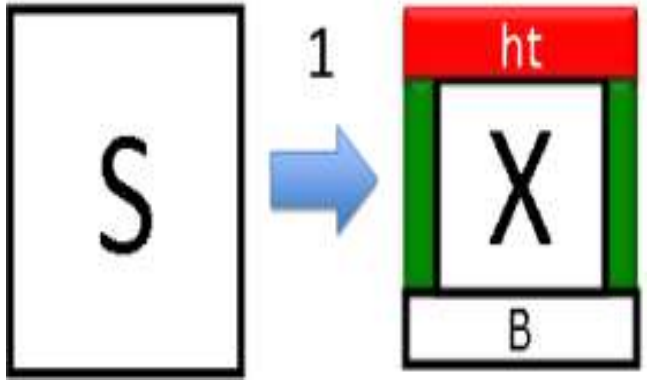


Positioning of Parts

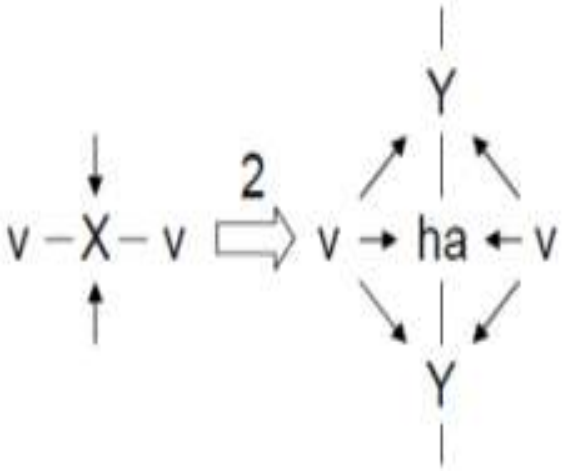
Examples of Production Rules

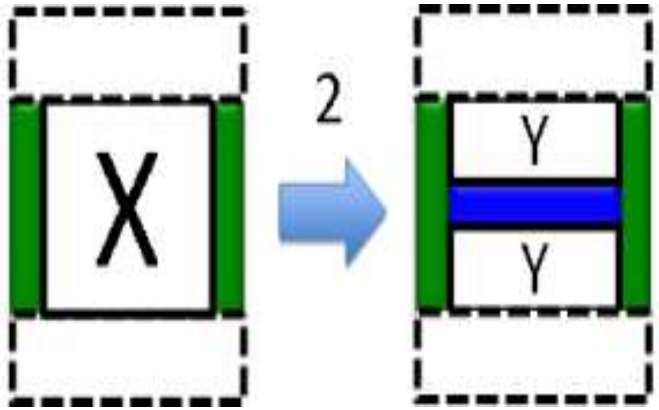
Production Rule 1





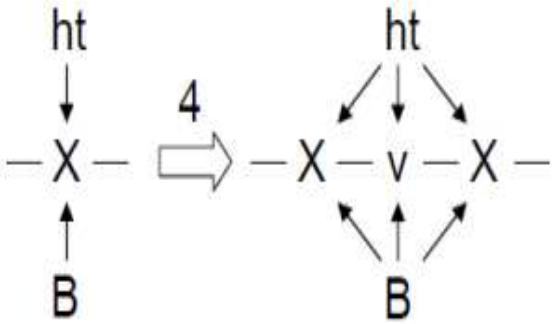
Production Rule 2

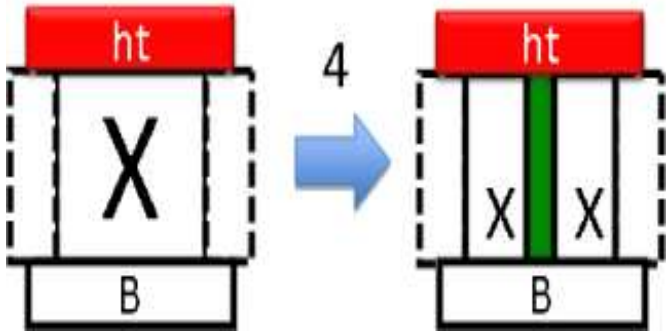




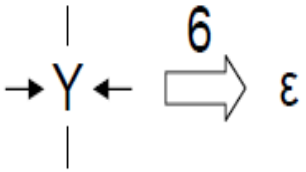
Examples of Production Rules

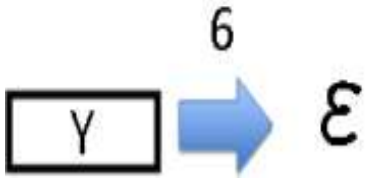
Production Rule 4



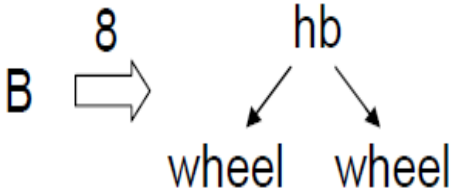


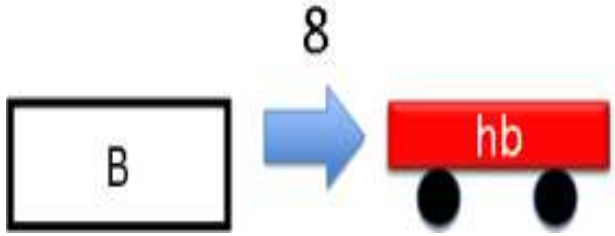
Production Rule 6



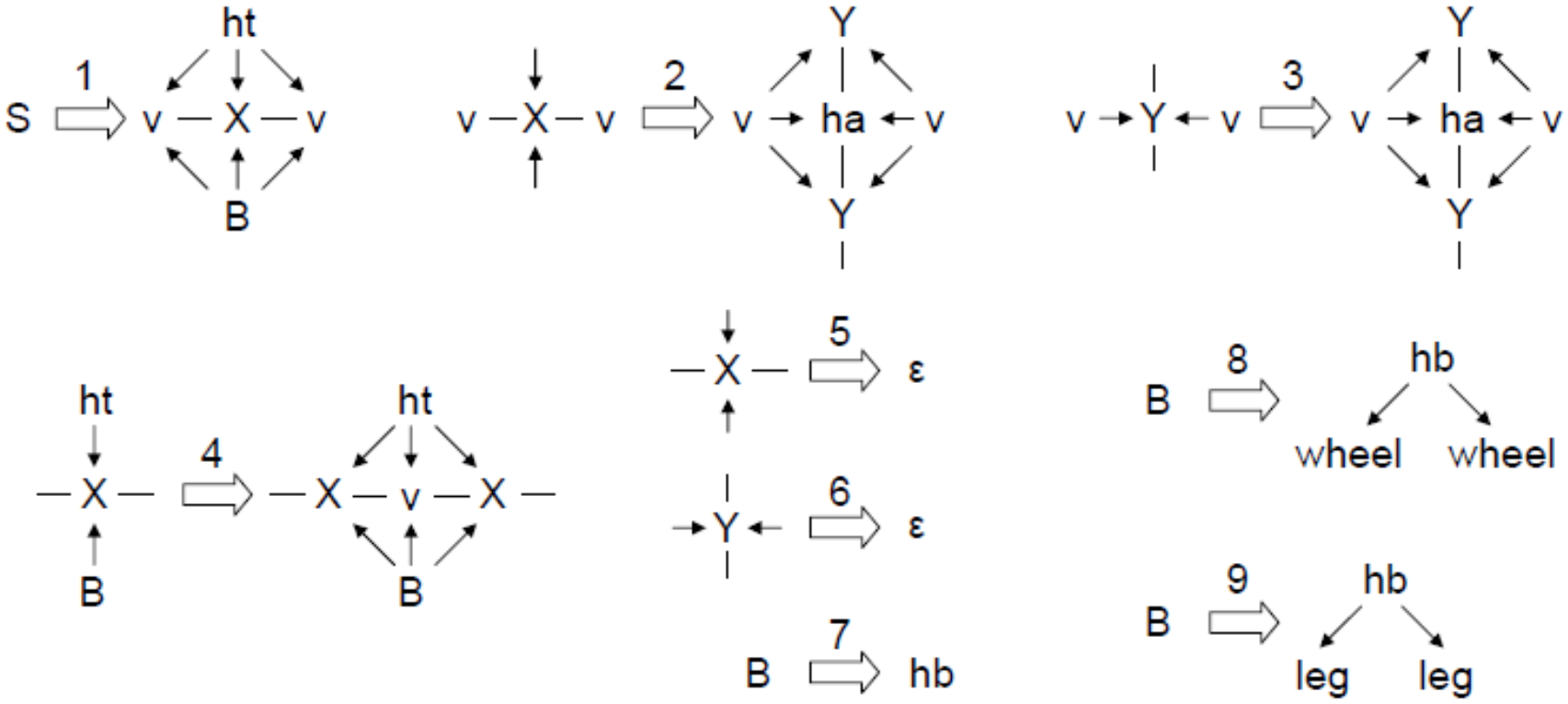


Production Rule 8

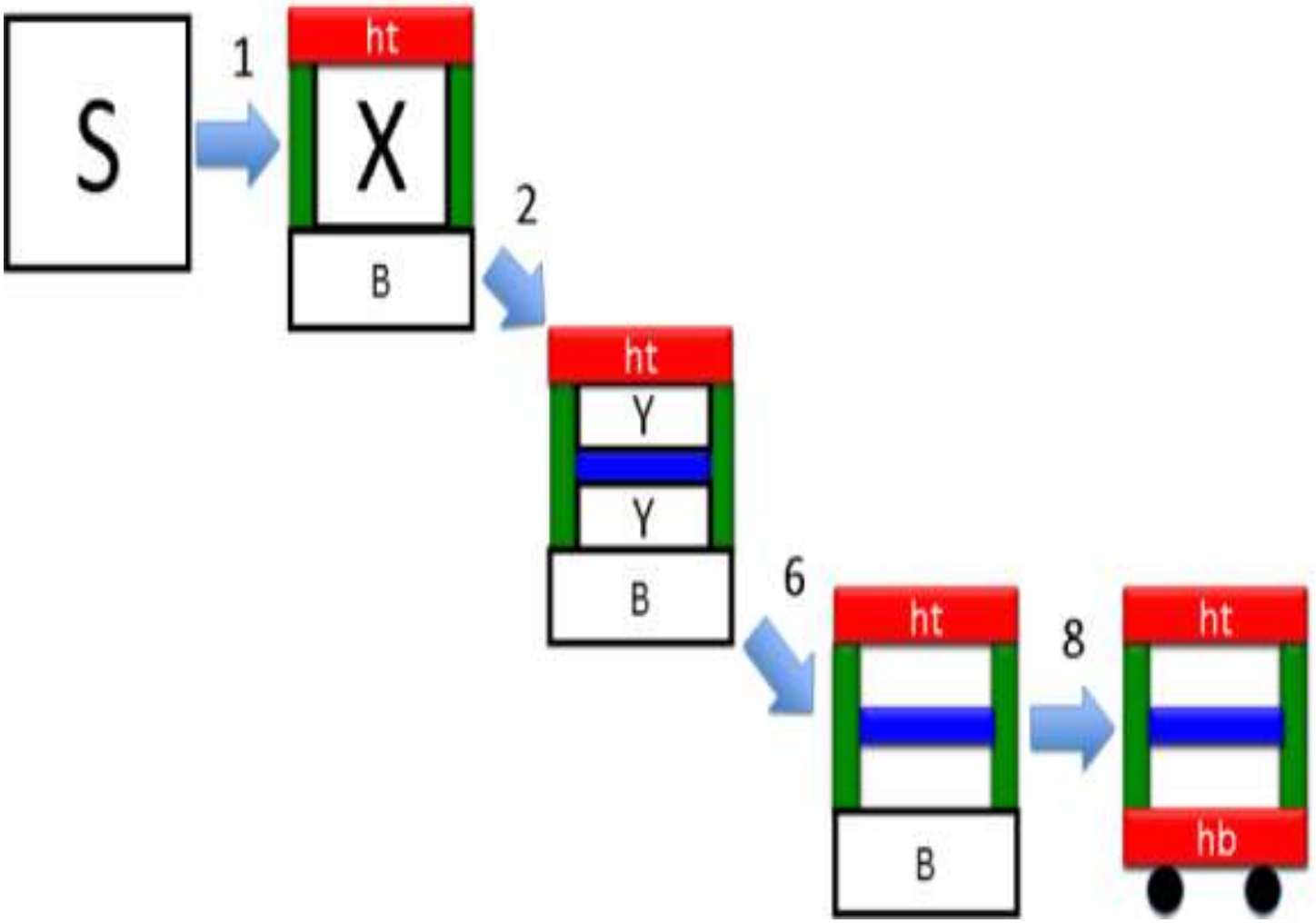




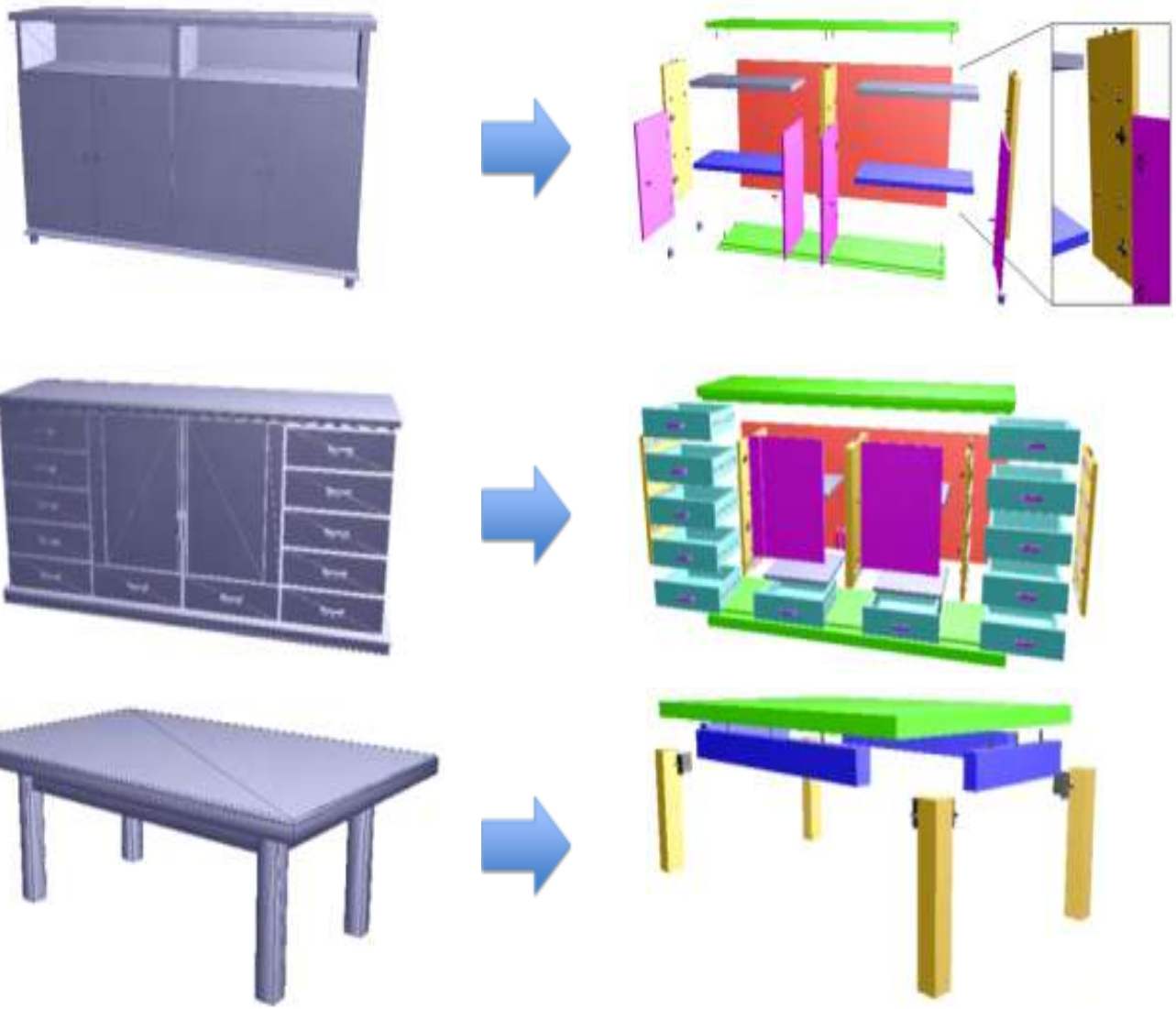
All Production Rules



Sequence of Production Rules



Inverse Procedural Modelling

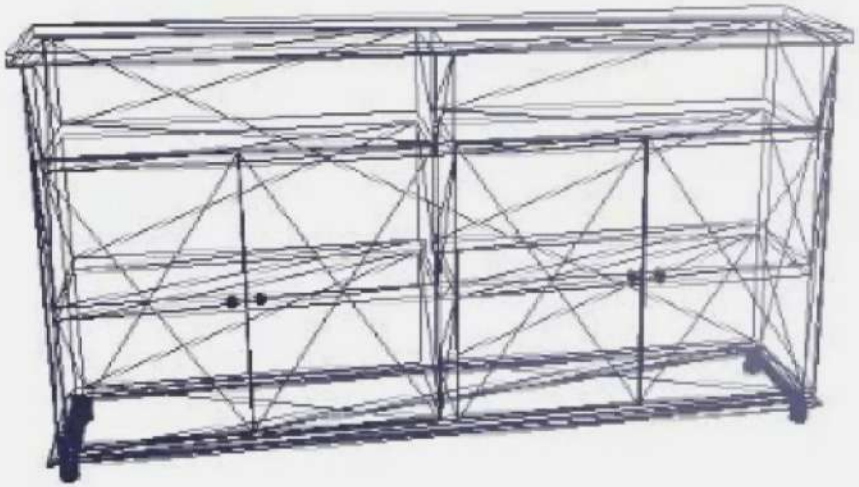


Input: 3D model

Output: Fabricatable Parts and Connectors

Converting 3D Furniture Models to Fabricable Parts and Connectors, Lau et al., Siggraph 2011

Results: IKEA ALVE Cabinet



Input: 3D model (Google Warehouse)
IKEA ALVE cabinet