Fundamentals of Computer Graphics and Image Processing

Modeling (01)

doc. RNDr. Martin Madaras, PhD.
martin.madaras@fmph.uniba.sk

Computer Graphics

- Image processing
  - Representing and manipulation of 2D images
- **Modeling**
  - Representing and manipulation of 2D and 3D objects
- Rendering
  - Constructing images from virtual models
- Animation
  - Simulating changes over time

How the lectures should look like #1

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!
- We will go into depth as far, as there are no questions

What is Modeling?

- Representation and manipulation of objects (which ones?)
  - Acquire
  - Edit
  - Transform
  - Smooth
  - Render
  - Deform
  - Morph
  - Compress
  - Transmit
  - Analyze

What is Modeling?

- Representation and manipulation of objects
  - Acquire
  - Edit
  - Transform
  - Smooth
  - Render
  - Deform
  - Morph
  - Compress
  - Transmit
  - Analyze

Modeling

- How to represent ..
  - 2D and 3D objects in a computer?
  - Acquire computer representations of objects?
  - Manipulate representations of objects?

Quick test #1

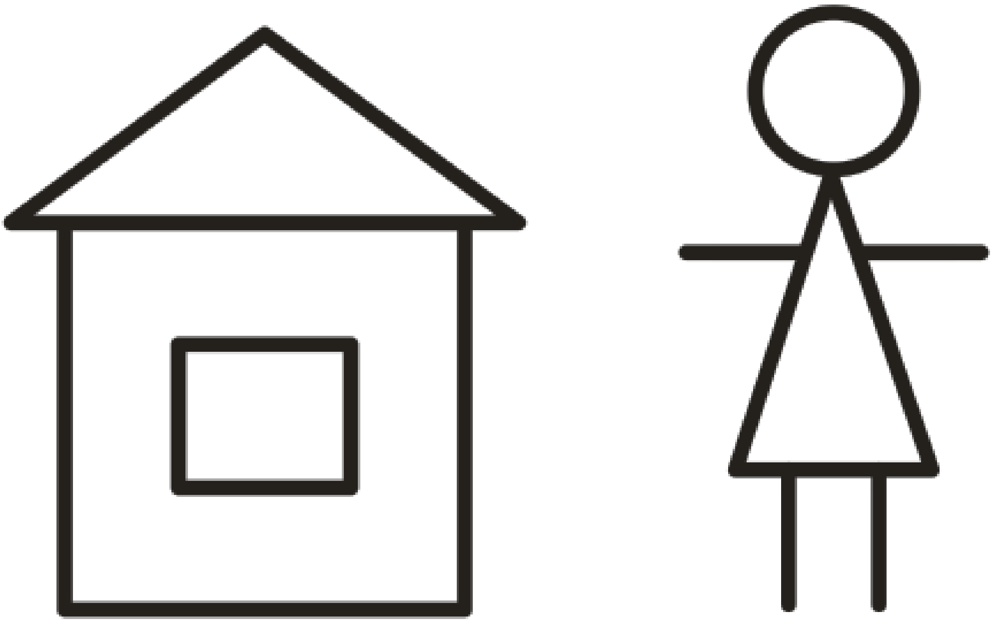
- Describe the picture

Quick test #2

- Describe the picture

Quick test #3

- Volunteers: Describe the image to others
- Others: Reproduce the image

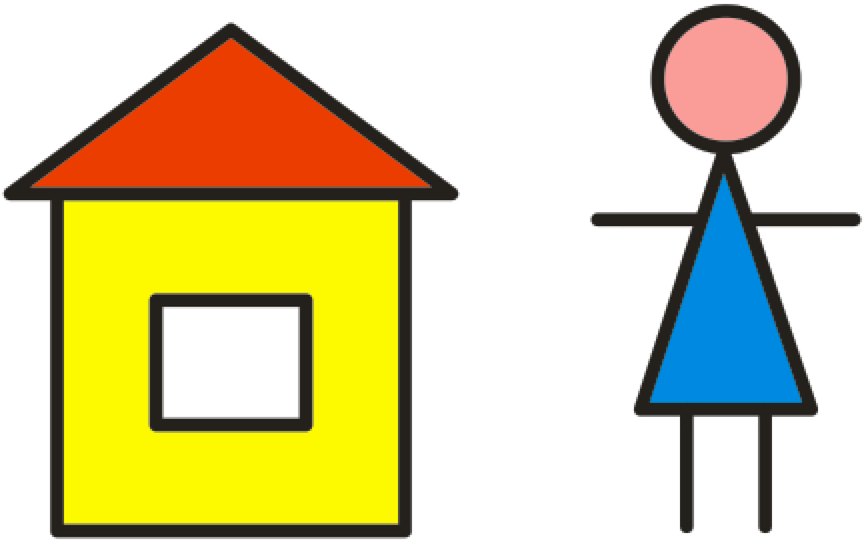


Semantic vs. numeric

- Humans – semantic representation
  - concepts, notions, meanings, emotions…
  - imprecise, ambiguous

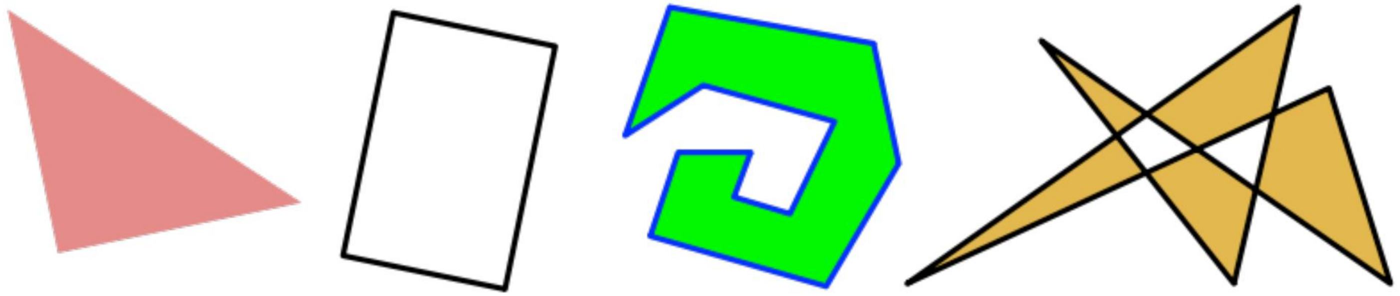
- Computers – numeric representation
  - exact, mathematical, straightforward

Detailed representation



2D Object Representations

- How to define 2D shapes?

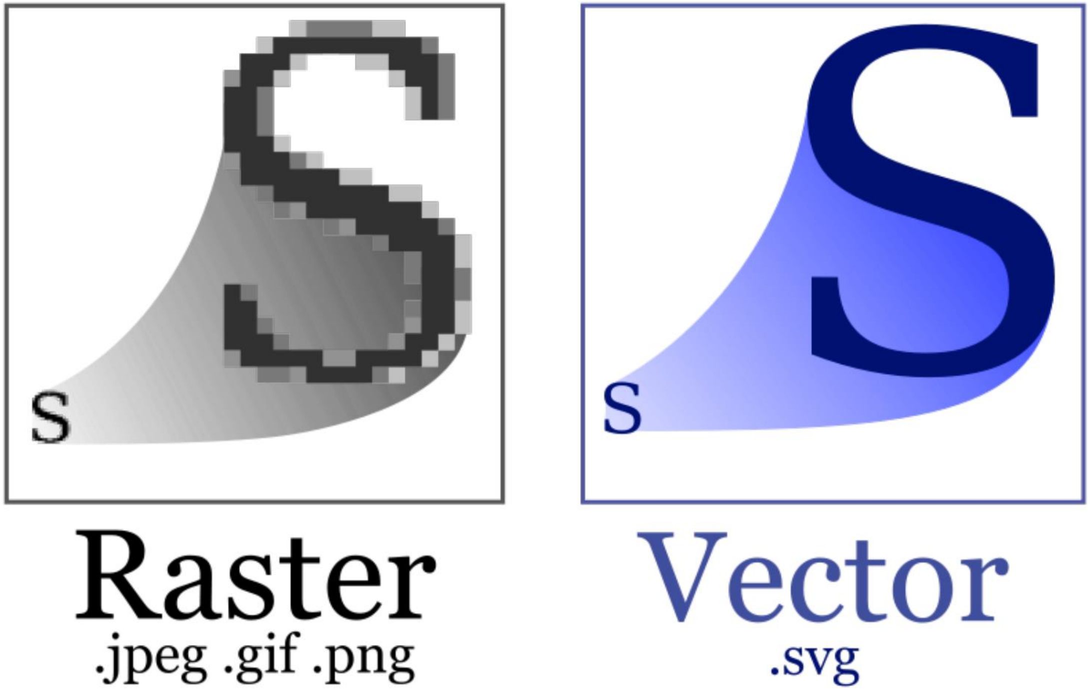


2D Object Representations

- Let's define these objects

2D Object Representations

- How to make 2D representation smooth at any scale?



Bitmaps, raster, pixels, explicit

Shapes, vectors, curves, parametric, implicit

2D Object Representations

- Lines
- Polygon (set of lines)
- Curves
- Vector image (set of curves)

2D Object Representations

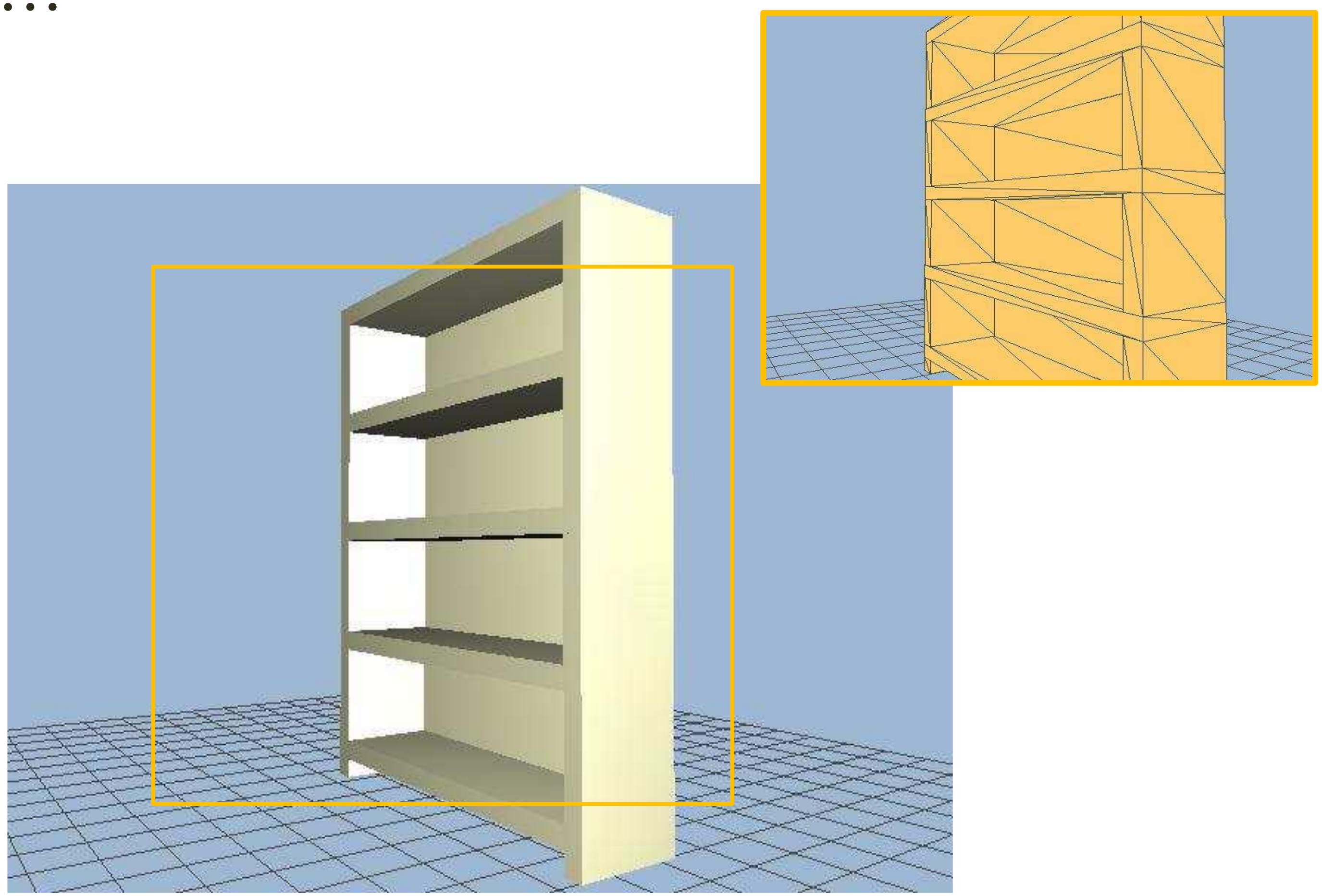
- Vector Images

3D Object Representations

- How to represent a 3D object?

3D Object Representations

- Polygons…

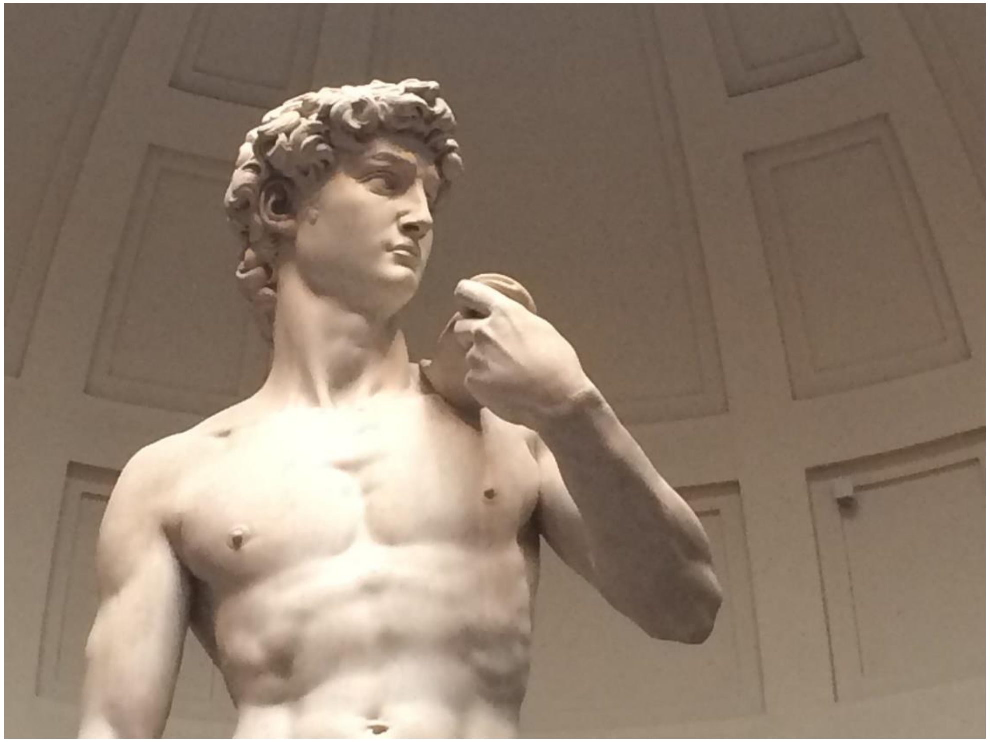


3D Object Representations

- Voxels…

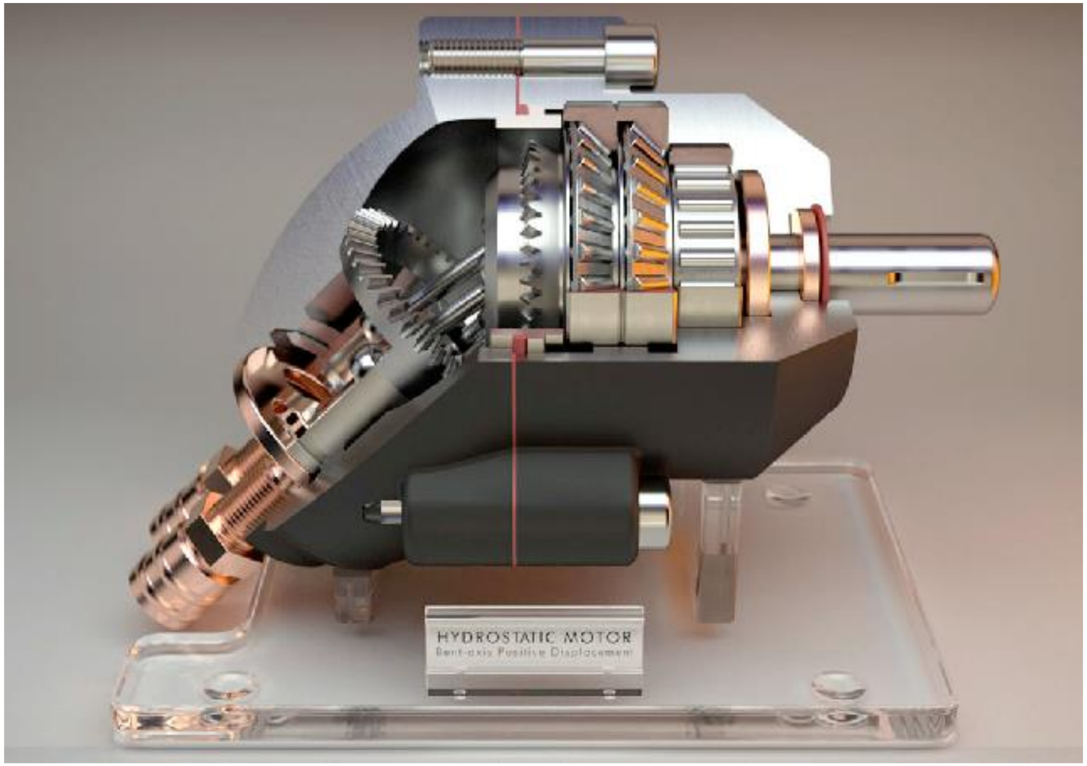
3D Object Representations

- Recreating artwork
  - How to represent?



3D Object Representations

- Mechanical objects
  - How to represent?



3D Object Representations

- Maps, Cities, Landscape
  - How to represent?

3D Object Representations

- Clouds, Smoke, Fog, Water
  - How to represent?



2D Object Representations

- Pixels
  - Images
- Lines
  - Curves
- Polygons
  - Discrete, Vector graphics

3D Object Representations

- Points
  - Range Image, Point Cloud
- Surfaces
  - Polygonal, Subdivision, Parametric, Implicit
- Solids
  - Voxels, BSP Tree, CSG, Sweep, etc.
- Hierarchical Structures
  - Scene graph, Application specific…

Why so many representation?

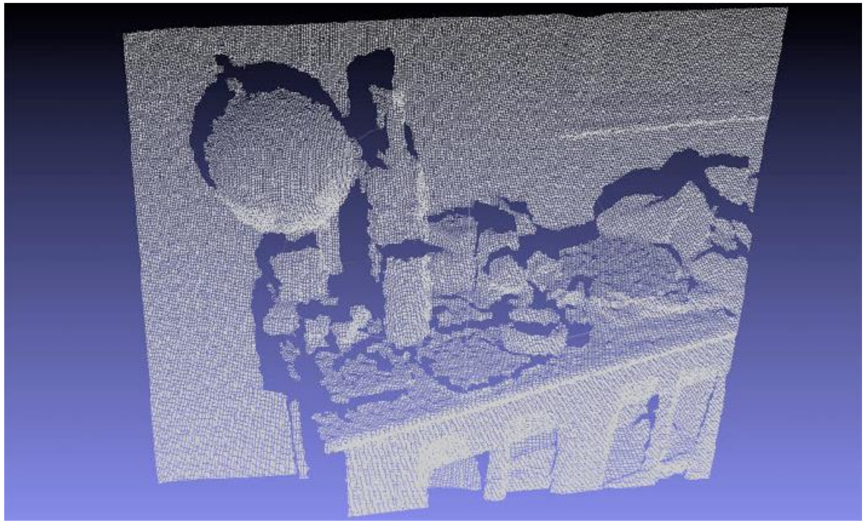
- Efficiency for different tasks
  - Rendering
  - Acquisition
  - Manipulation
  - Animation
  - Analysis
- Data structures determine algorithms

Outline

- **Points**
  - Range Image, Point Cloud
- Surfaces
  - Polygonal, Subdivision, Parametric, Implicit
- Solids
  - Voxels, BSP Tree, CSG, Sweep
- Hierarchical Structures
  - Scene graph, Application specific

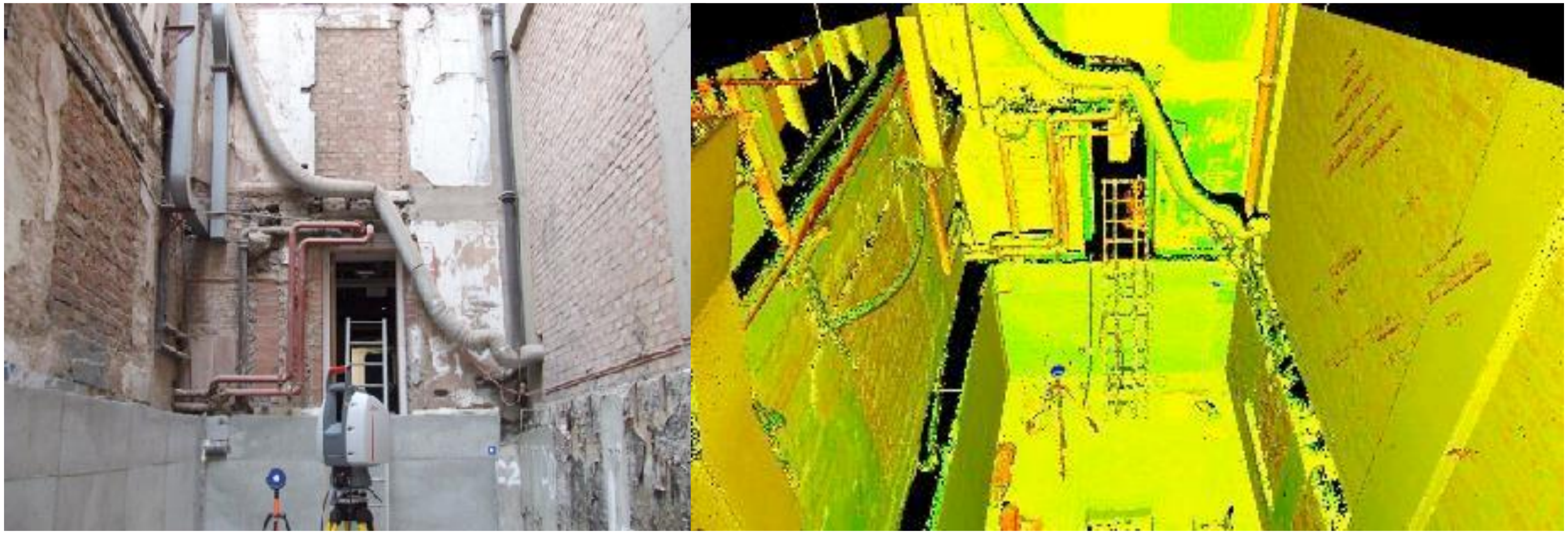
Range Image

- Set of 3D points mapping to pixels of depth image
- Structured Point Cloud
  - Acquired using a range scanner (eg. Kinect)



Point Cloud

- Unstructured set of 3D point samples
  - Acquired from multiple range scans, vision, etc.



Point Cloud Animation: Road Survey

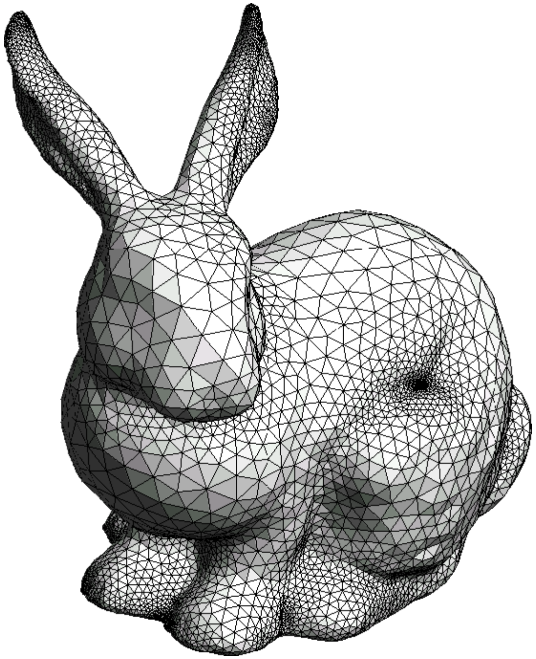
https://www.youtube.com/watch?v=f_ng212b-UM

Outline

- Points
  - Range Image, Point Cloud
- **Surfaces**
  - Polygonal, Subdivision, Parametric, Implicit
- Solids
  - Voxels, BSP Tree, CSG, Sweep
- Hierarchical Structures
  - Scene graph, Application specific

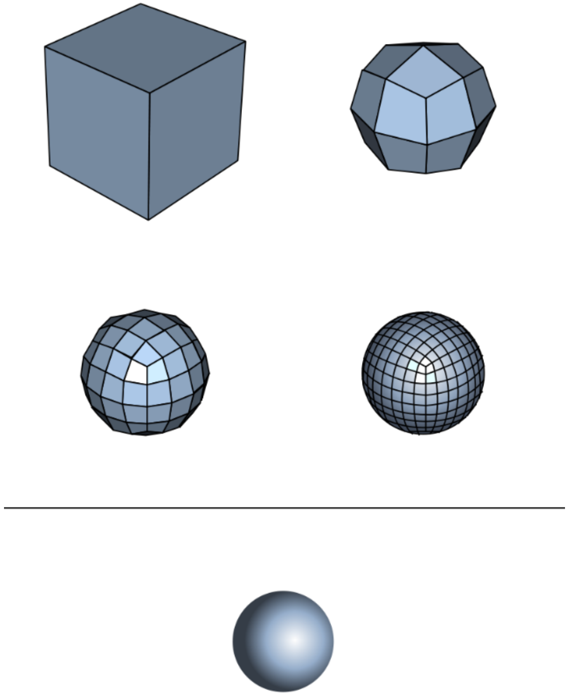
Polygonal Mesh

- Connected mesh of polygons (usually triangles)
  - Most common representation, supported in OpenGL



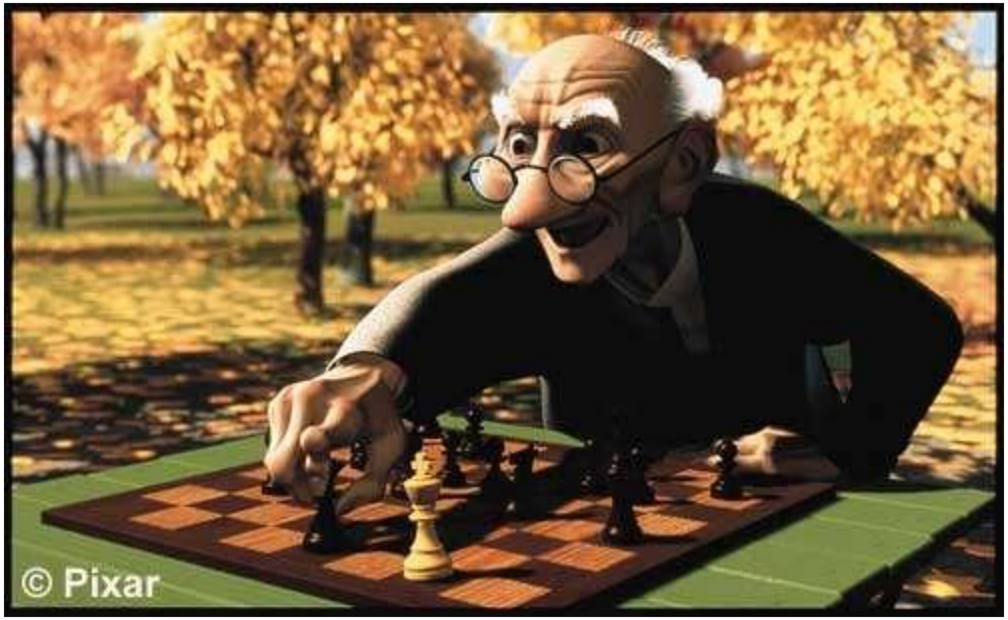
Subdivision Surface

- Coarse mesh with subdivision rule
- Smooth surfaces are defined as sequences of refinement



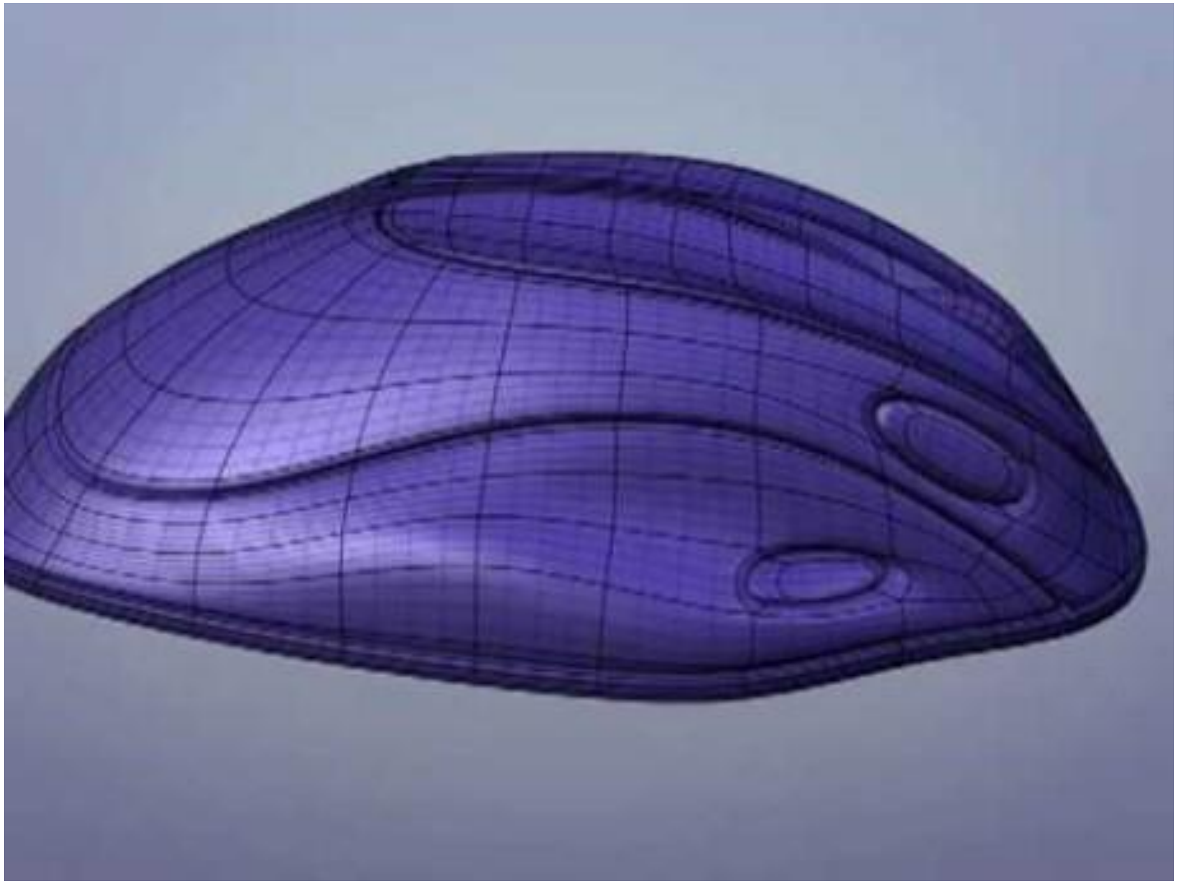
Subdivision Surface

- Properties
  - Accurate
  - Concise
  - Intuitive specification
  - Local support
  - Affine invariant
  - Arbitrary topology
  - Guaranteed continuity
  - Natural parametrization
  - Efficient display
  - Efficient intersections



Geri's game, Pixar, 1997:
https://www.youtube.com/watch?v=kweN7VLx-JE

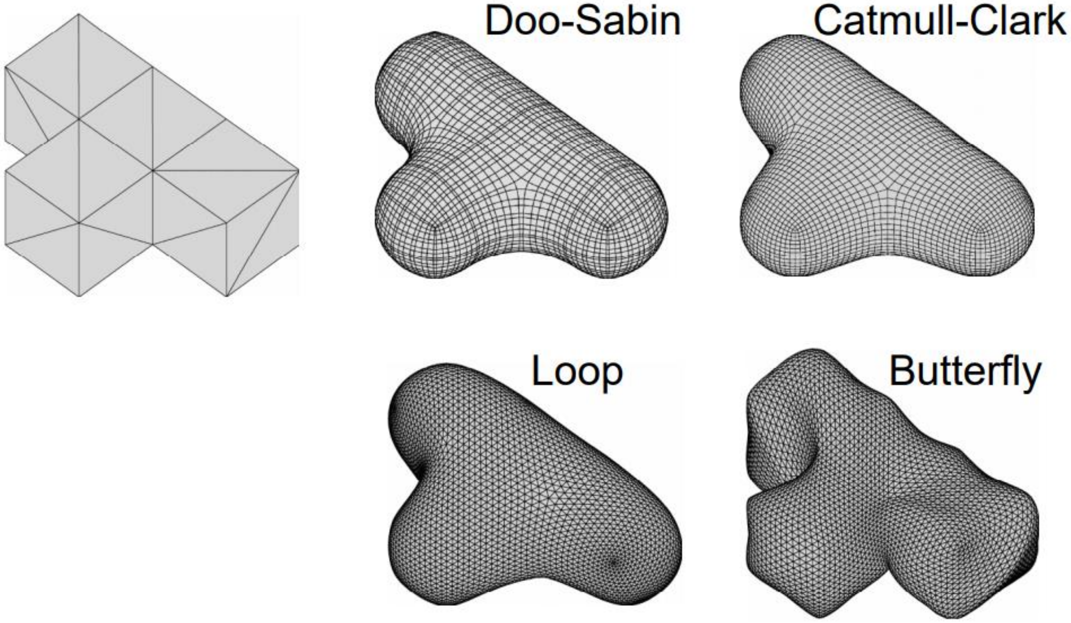
Subdivision Surfaces: Overview



https://www.youtube.com/watch?v=ckOTI2GcS-E&t=26s

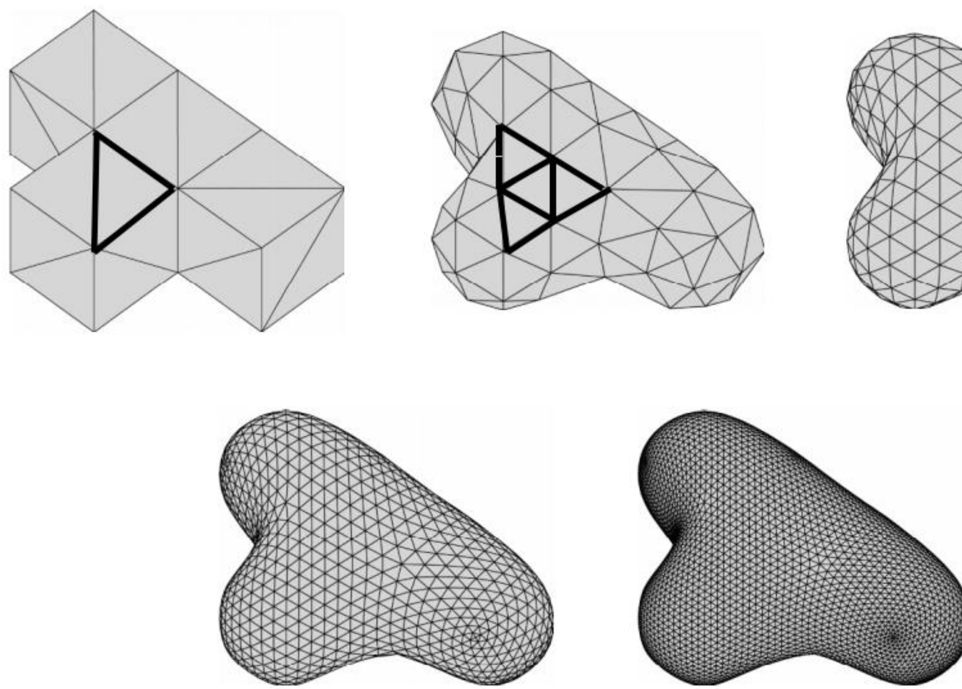
Subdivision Surfaces: Overview

- Subdivision surfaces
  - Loop subdivision
  - Catmull-Clark
  - Butterfly, Doo-Sabin, etc.



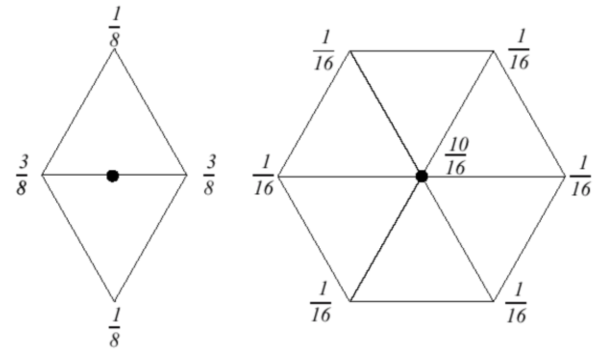
Example: Loop subdivision

- How to refine mesh ?
  - Refine each triangle into 4 triangles by splitting each edge and connecting the vertices

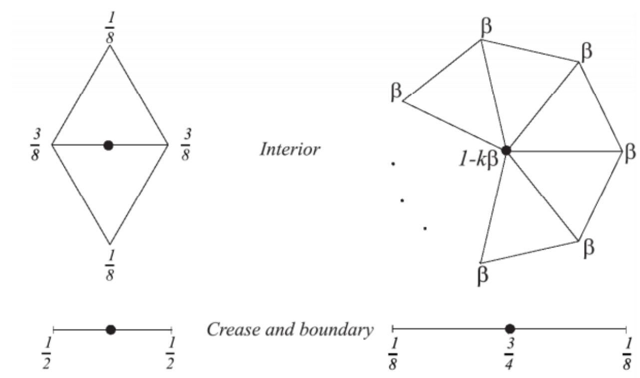


Example: Loop subdivision

- How position new vertices?
  - Choose location of the vertices as weighted average of original vertices in local neighborhood



  - Rules for *extraordinary vertices* and *boundaries*:



a. Masks for odd vertices

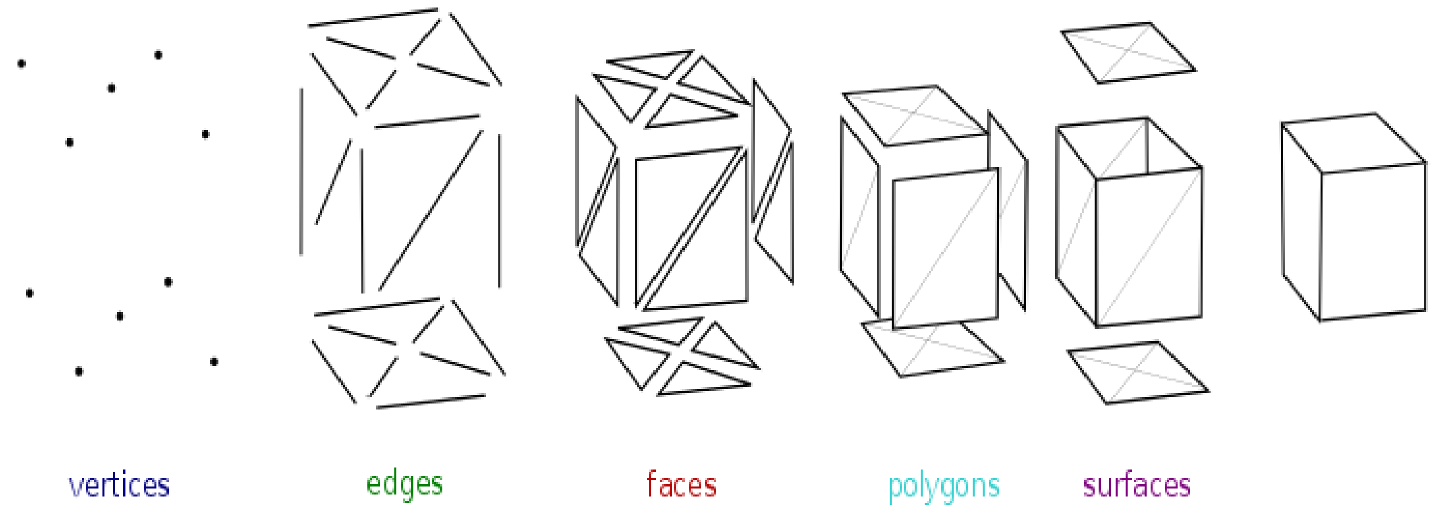
b. Masks for even vertices

Key Questions

- How to refine mesh ?
  - Aim for properties like smoothness
- **How to store mesh ?**
  - Aim for efficiency of implementing subdivision rules

Polygonal meshes

- V, E, F
- P, S

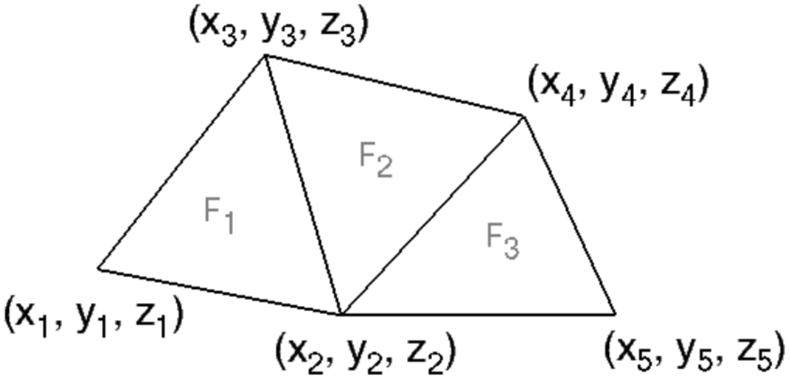


Polygonal Meshes

- Mesh Representations
  - Independent faces
  - Vertex and face tables
  - Adjacency lists
  - Winged-Edge

Independent Faces

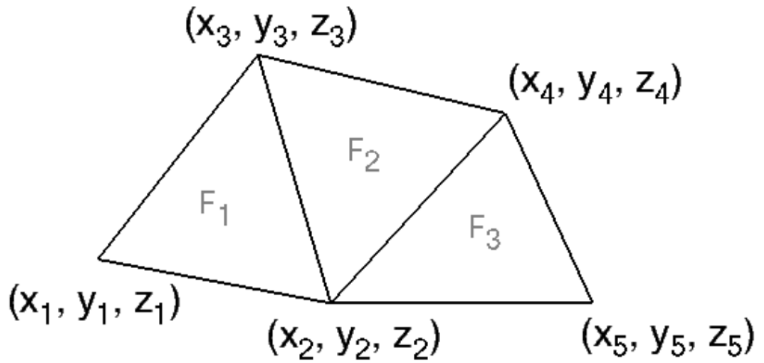
- Each face lists vertex coordinates
  - Redundant vertices
  - No **topology** information



FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

- Each face lists vertex references
  - Shared vertices
  - Still no topology information

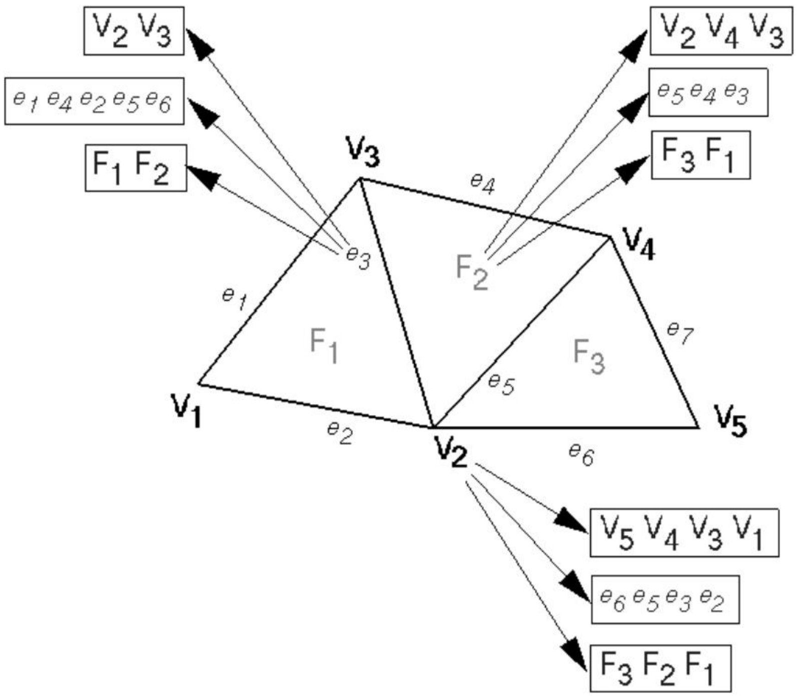


VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

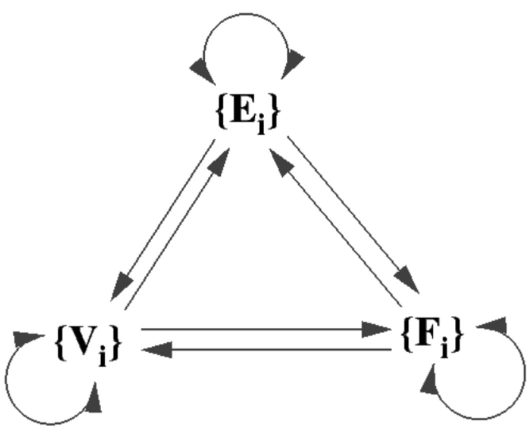
Adjacency Lists

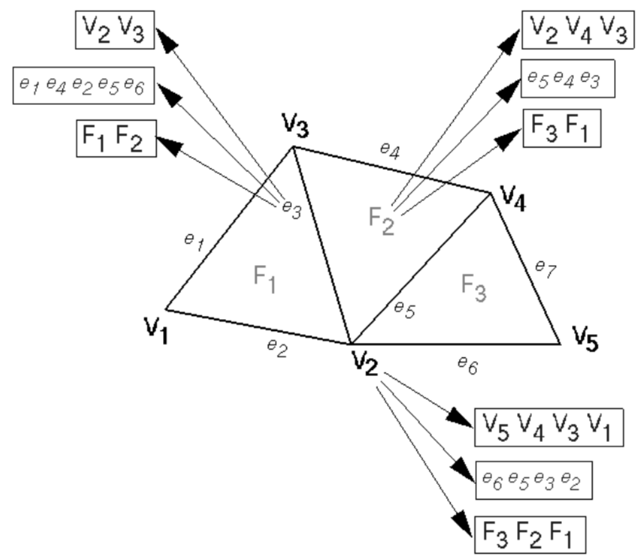
- Store all vertex, edge and face adjacency
  - Efficient topology traversal
  - Extra storage



Partial Adjacency Lists

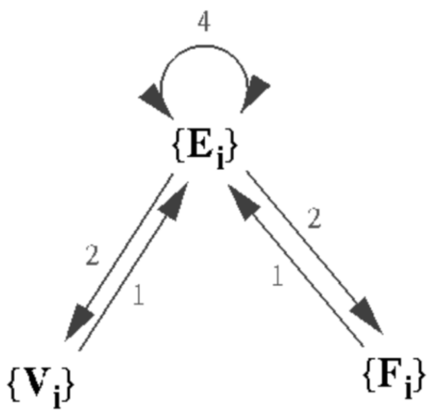
- Can we can store only some adjacency information and derive others?

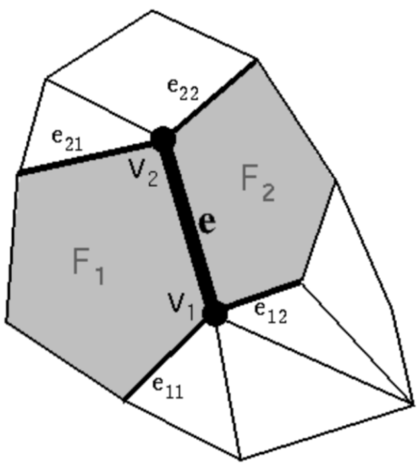




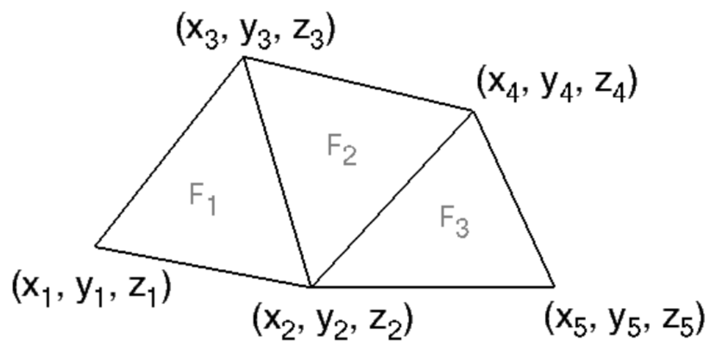
Winged Edge

- Adjacency encoded in edges
  - All adjacencies in O(1) time
  - Little extra storage
  - Arbitrary polygons





Winged Edge Example



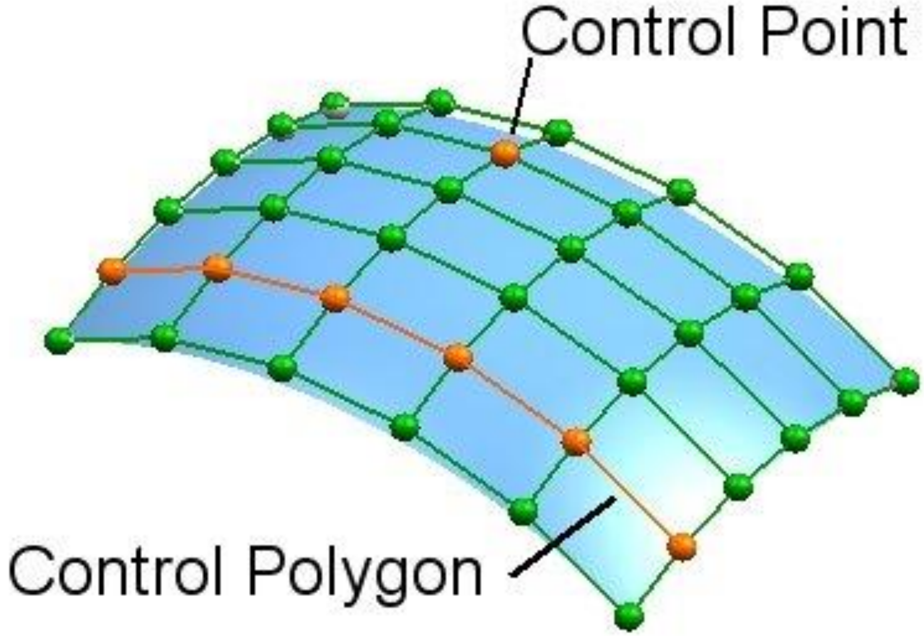
VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE					11	12	21	22
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Parametric Surface

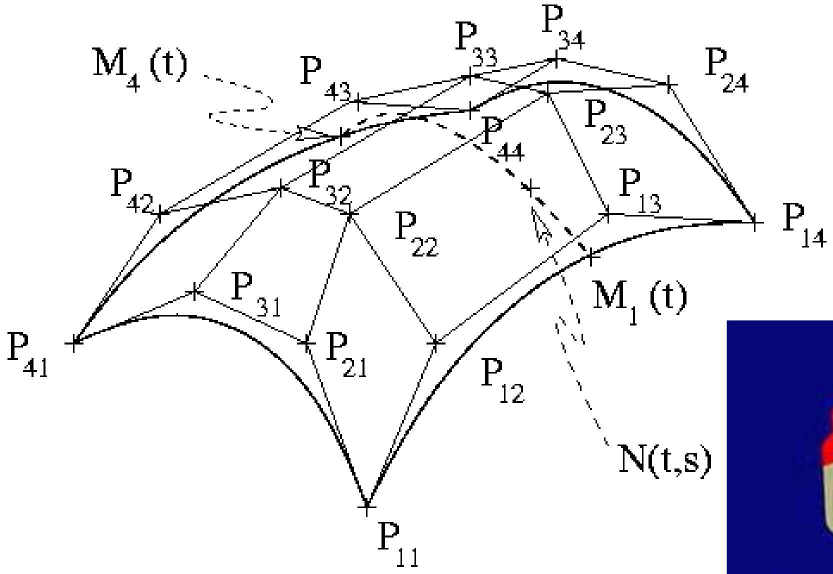
- Defined using control points
- Surfaces are defined using parametric functions
- Set m × n control points
- Parameters u,v

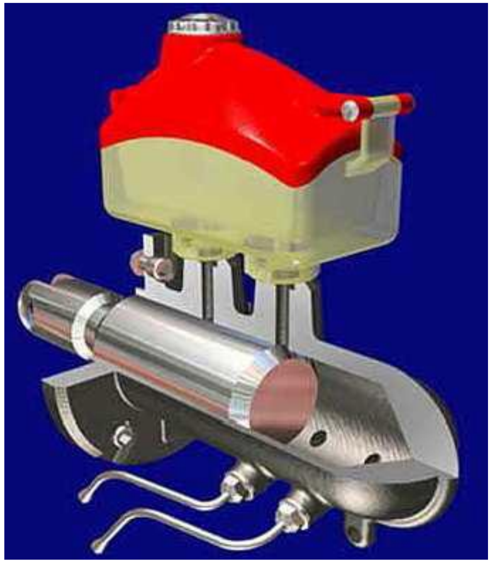


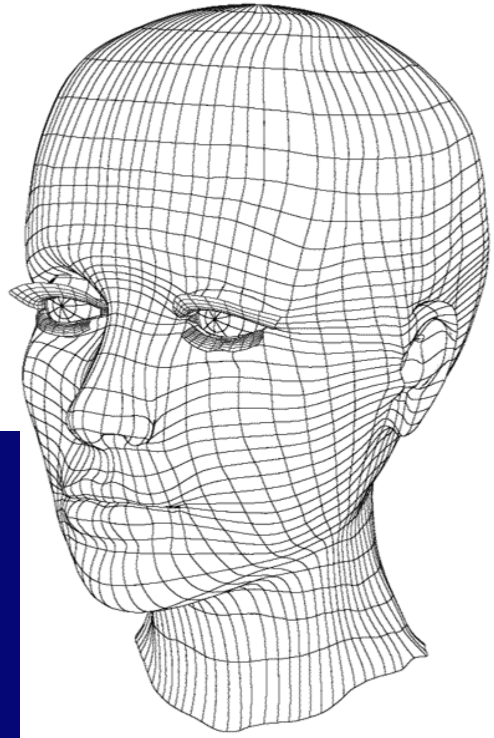
http://cadauno.sourceforge.net/

Parametric Surface

- Cubic Bezier surface, NURBS

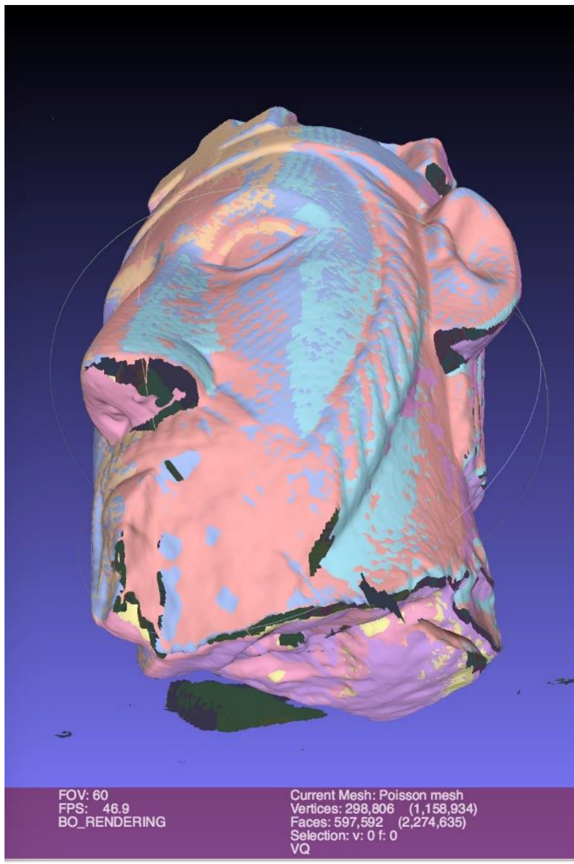




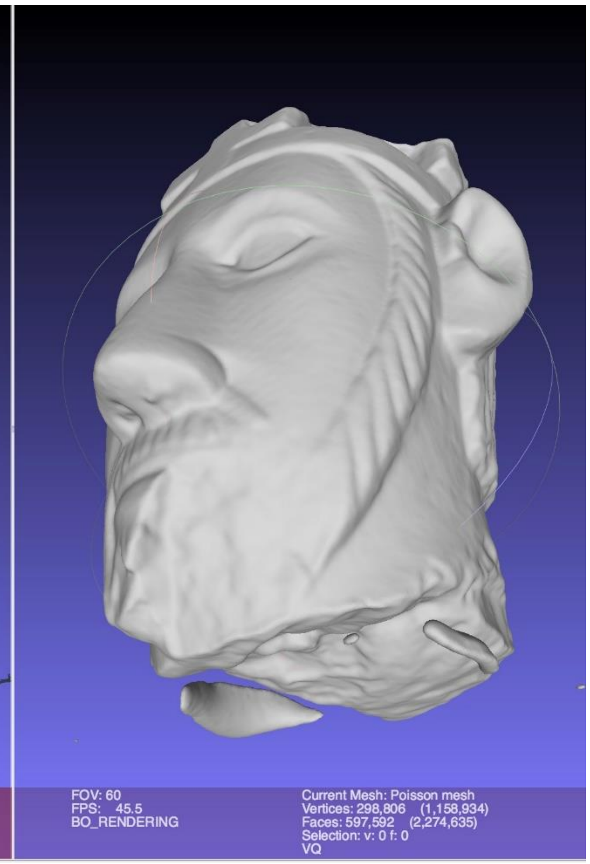


Implicit Surface

- Surface satisfying function F(x,y,z) = 0



Polygonal model

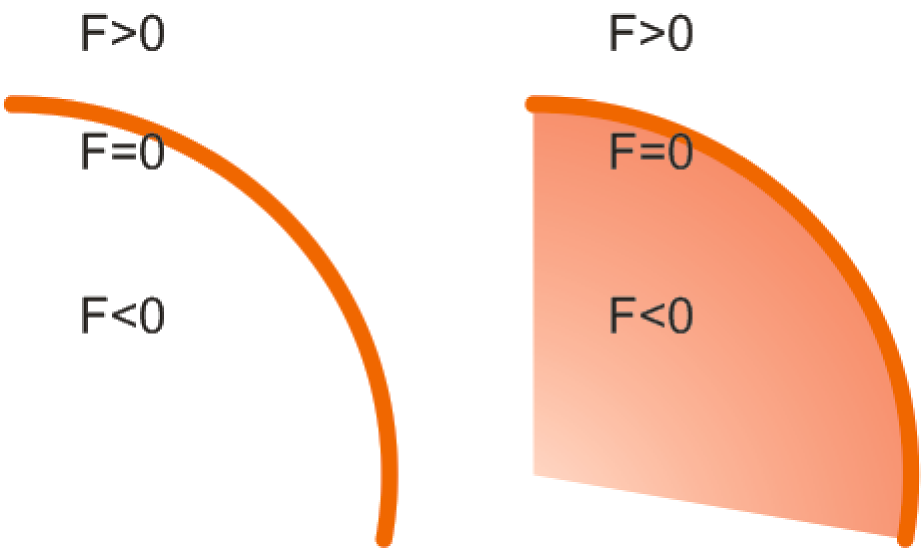


Implicit model

Question

What happens if we turn $F(x,y,z) = 0$

into $F(x,y,z) \leq 0$?



surface → volume

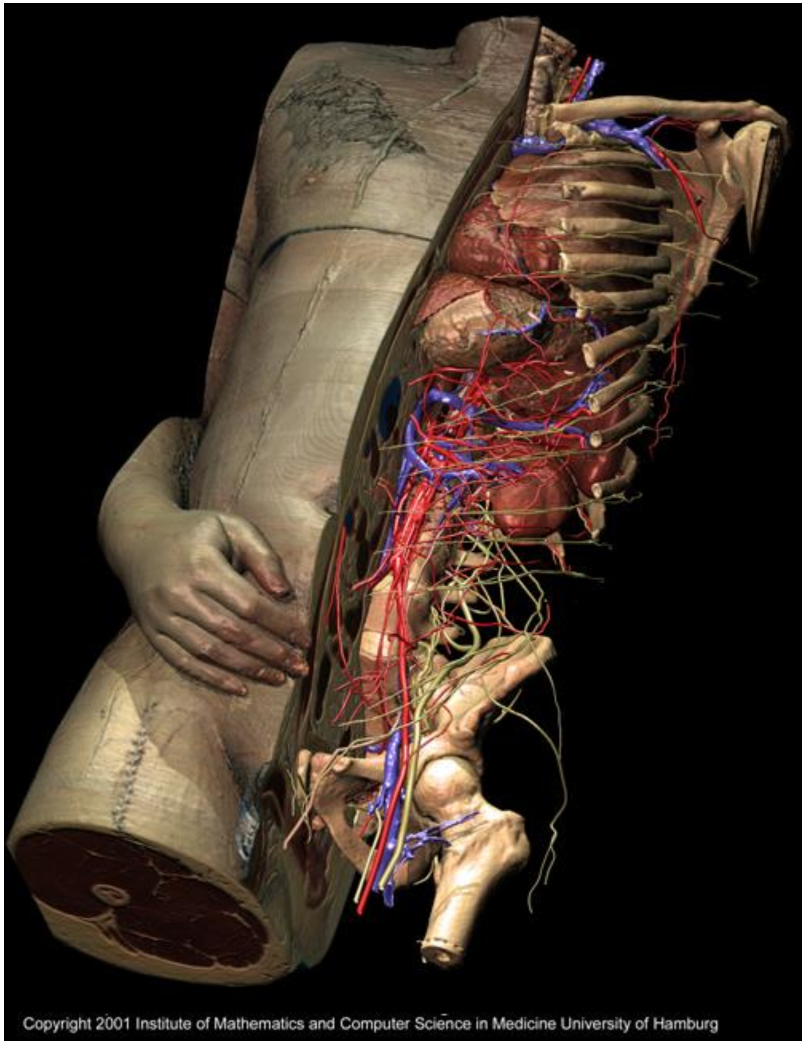
Outline

- Points
  - Range Image, Point Cloud
- Surfaces
  - Polygonal, Subdivision, Parametric, Implicit
- **Solids**
  - Voxels, BSP Tree, CSG, Sweep
- Hierarchical Structures
  - Scene graph, Application specific

Volumetric representation

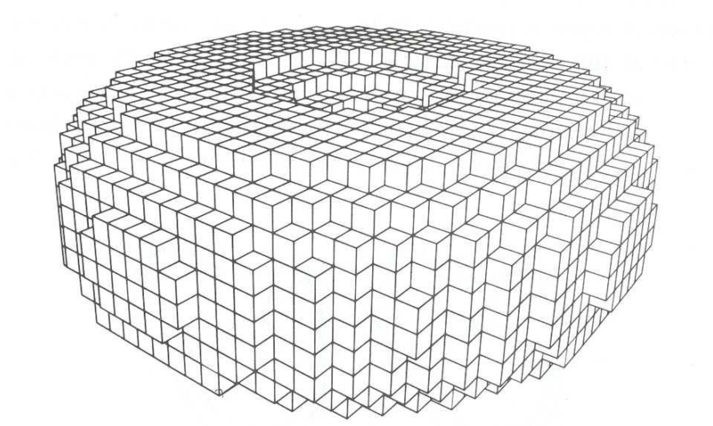
- Not only boundary but also the insides of the object

- Medicine
- Physics
- Simulations
- Animation



Voxels

- Uniform grid of volumetric samples
- Acquired using CAT, MRI scans etc.
- Volume elements, “3D pixels”
- Discrete

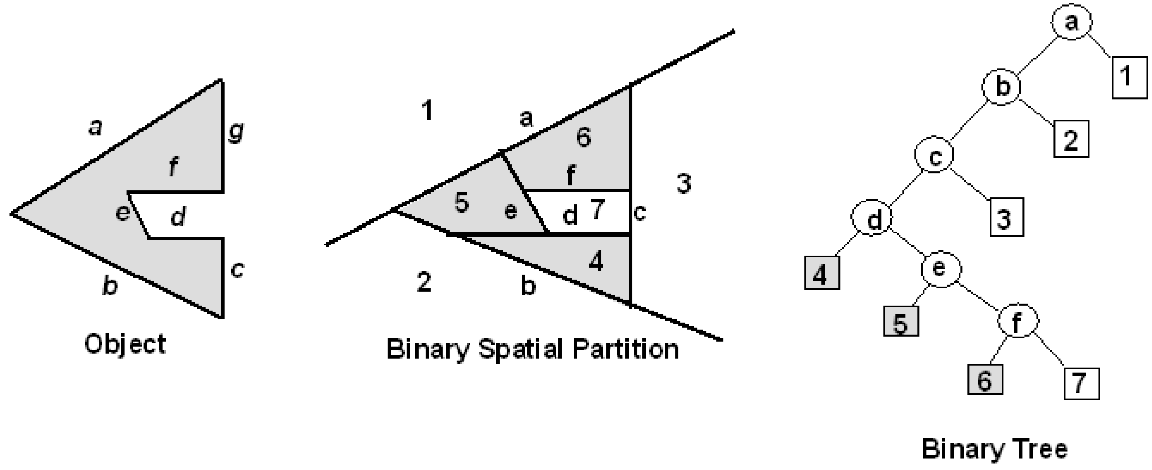


Volume Rendering Demo

https://www.youtube.com/watch?v=uSyUCLLNtMo

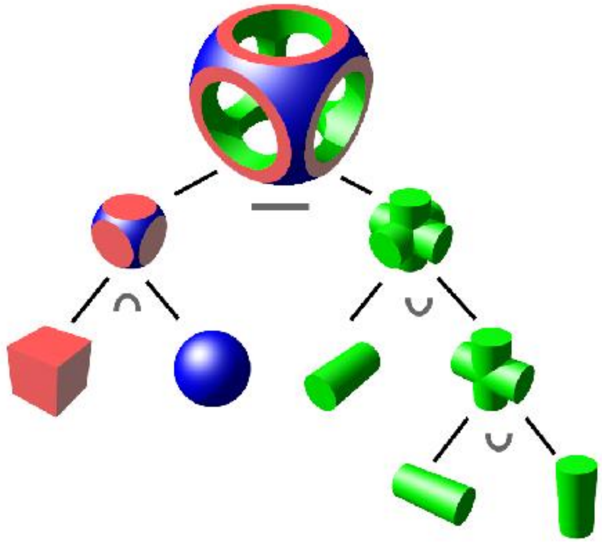
BSP Trees

- Binary space partition
  - Constructed from polygonal representations



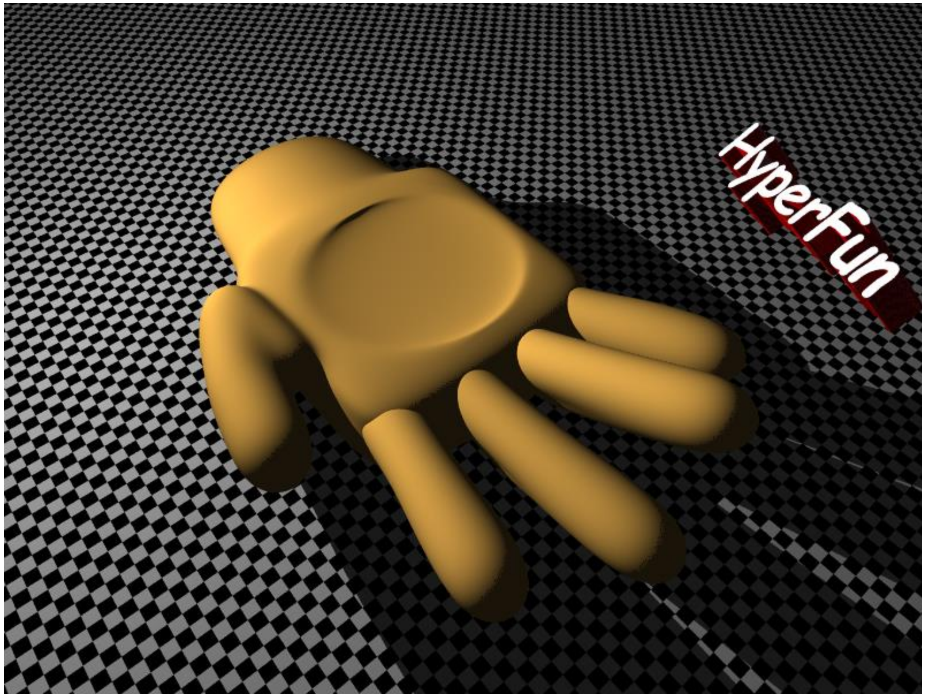
Constructive Solid Geometry

- Hierarchy of boolean operations
  - Union, Difference, Intersect applied to simple shapes



Functional Representation

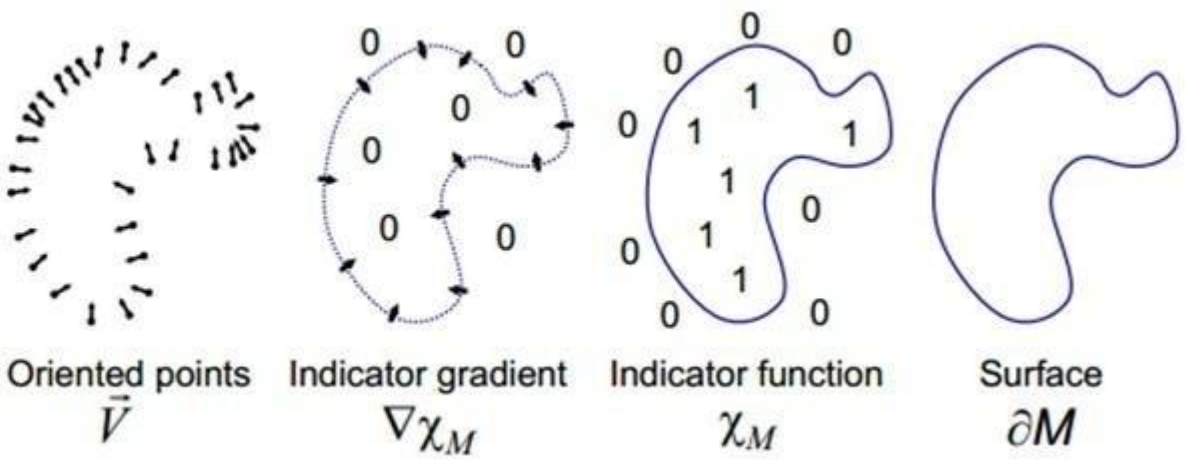
- F-rep ~ generalization of CSG
- More node functions – operators
  - e.g. object blending




```
center = [0, 0.5, 0];
se = hfSuperell(x, center, 8, 2.5, 8, 0.3, 0.3);

center = [0, -0.5, 0];
el_cly = hfEllCylZ(x, center, 4, 2);

wrist = el_cly & (8-x[3]) & (x[3]+20);

center = [0, 3.5, 0];
el1 = hfEllipsoid(x, center, 8, 1, 8);

center = [-2, 3.5, 0];
el2 = hfEllipsoid(x, center, 8, 1, 8);

center = [2, 3.5, 0];
el3 = hfEllipsoid(x, center, 8, 1, 8);

center = [-0.5, 3.5, -2];
el4 = hfEllipsoid(x, center, 8, 1, 8);

el = el1 | el2 | el3 | el4;

palm = hfBlendUni(se, wrist, 5, 2, 2) \ el;
```

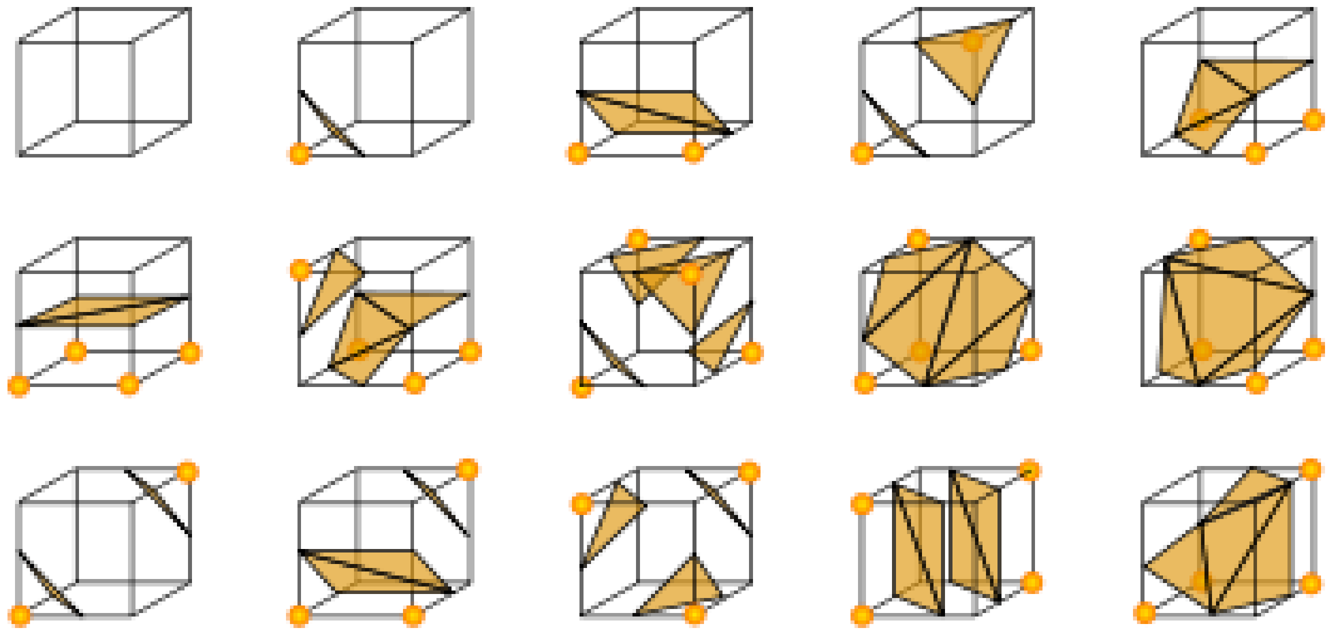
Mesh reconstruction

- Conversion into an implicit function
  - Poisson reconstruction



Mesh reconstruction

- Isosurface extraction
  - Marching cubes

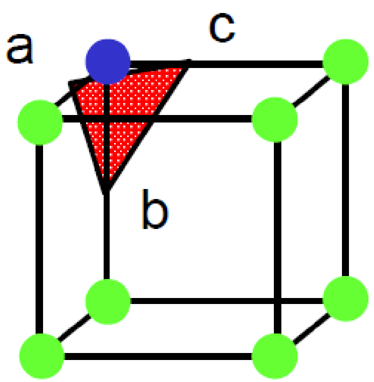


Marching cubes

- Create cubes
- Classify vertices
- Build indices
- Lookup edge list

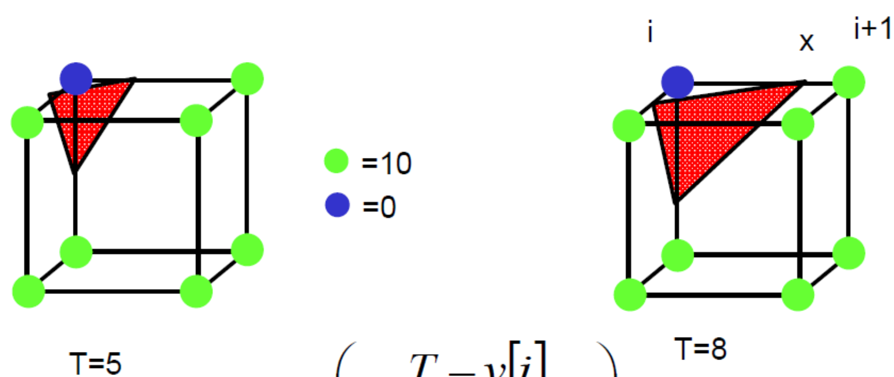
Index = 00000001

triangle 1 = a, b, c



Marching cubes

- Create cubes
- Classify vertices
- Build indices
- Interpolate Triangle Vertices



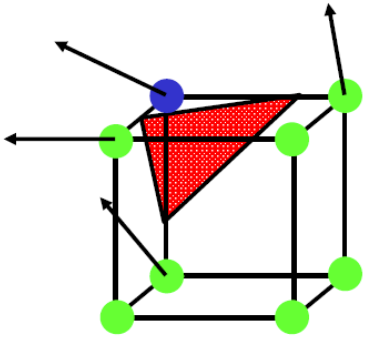
$$x = i + \left(\frac{T - v[i]}{v[i+1] - v[i]} \right)$$

Marching cubes

- Create cubes
- Classify vertices
- Build indices
- Interpolate Triangle Vertices
- Calculate normals

Calculate the normal at each cube vertex

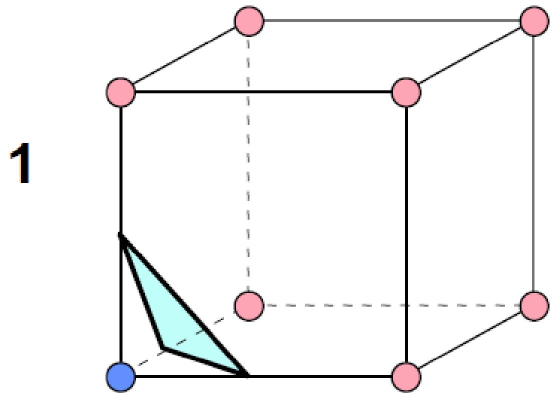
$$G_x = v_{i+1,j,k} - v_{i-1,j,k}$$
$$G_y = v_{i,j+1,k} - v_{i,j-1,k}$$
$$G_z = v_{i,j,k+1} - v_{i,j,k-1}$$

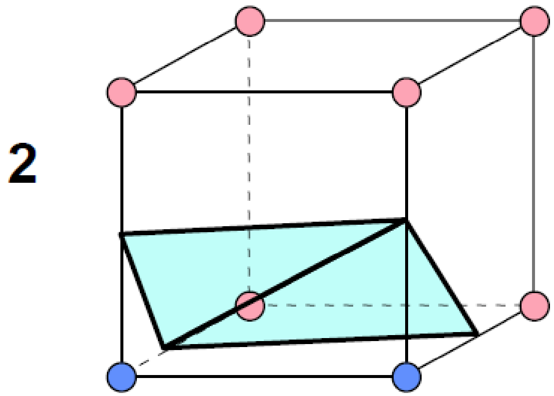


Use linear interpolation to compute the polygon vertex normal

Marching cubes

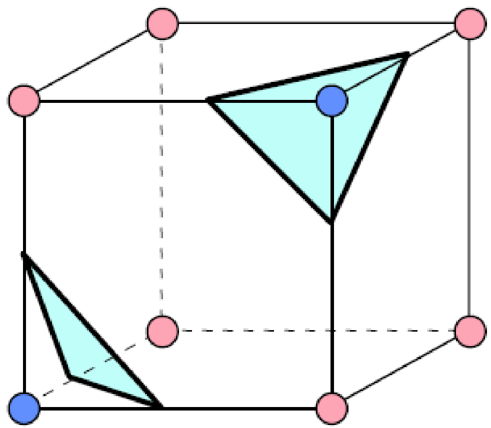
14 base cases
(other 240 derived with symmetry and rotation)



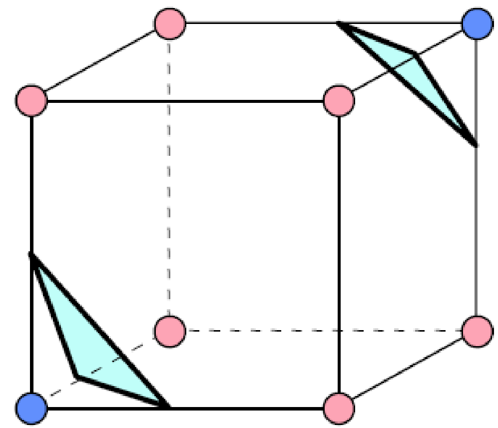


Marching cubes

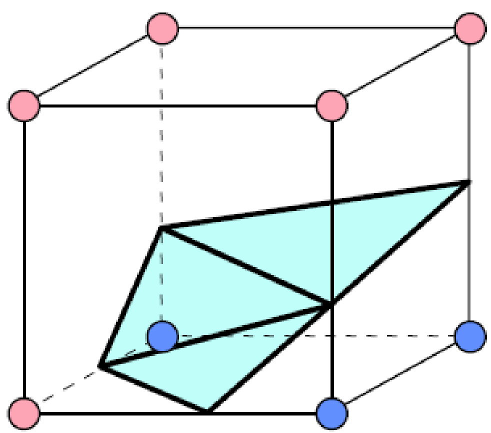
3



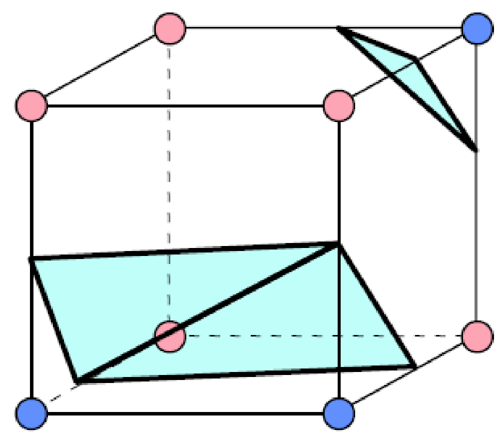
4



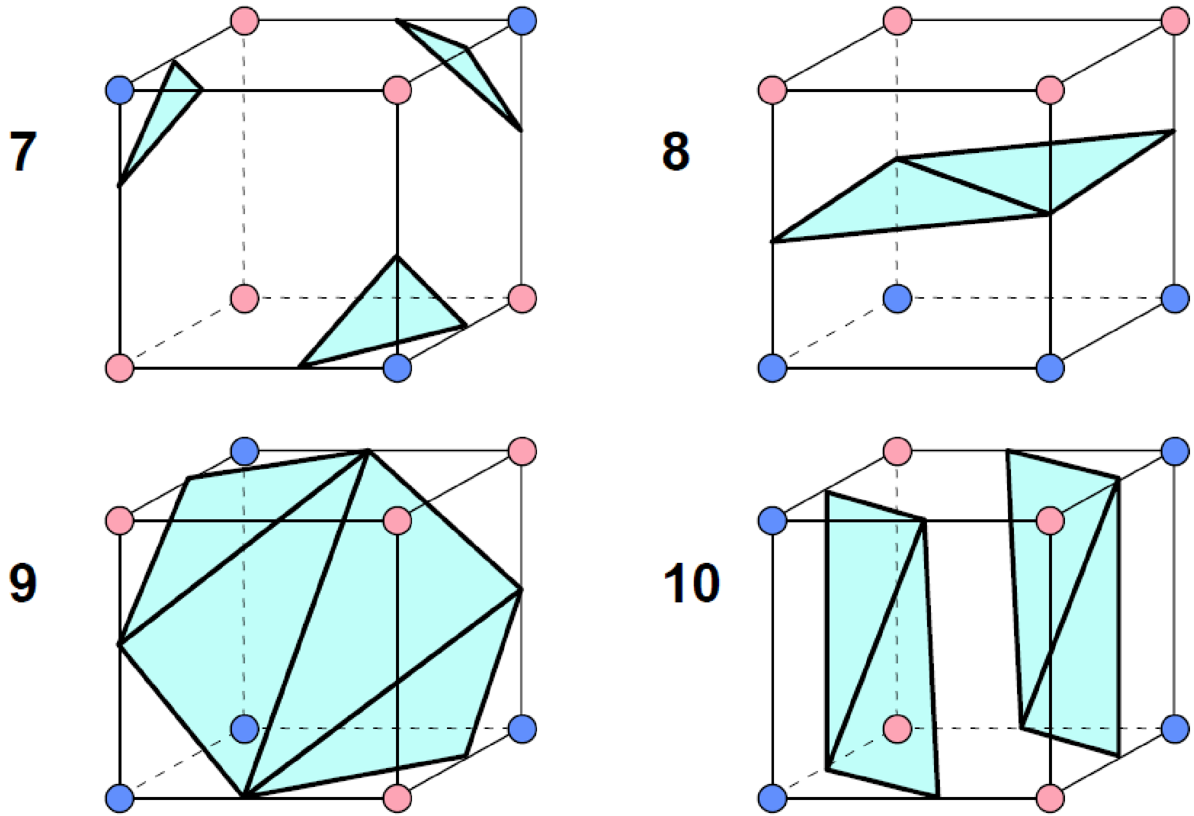
5



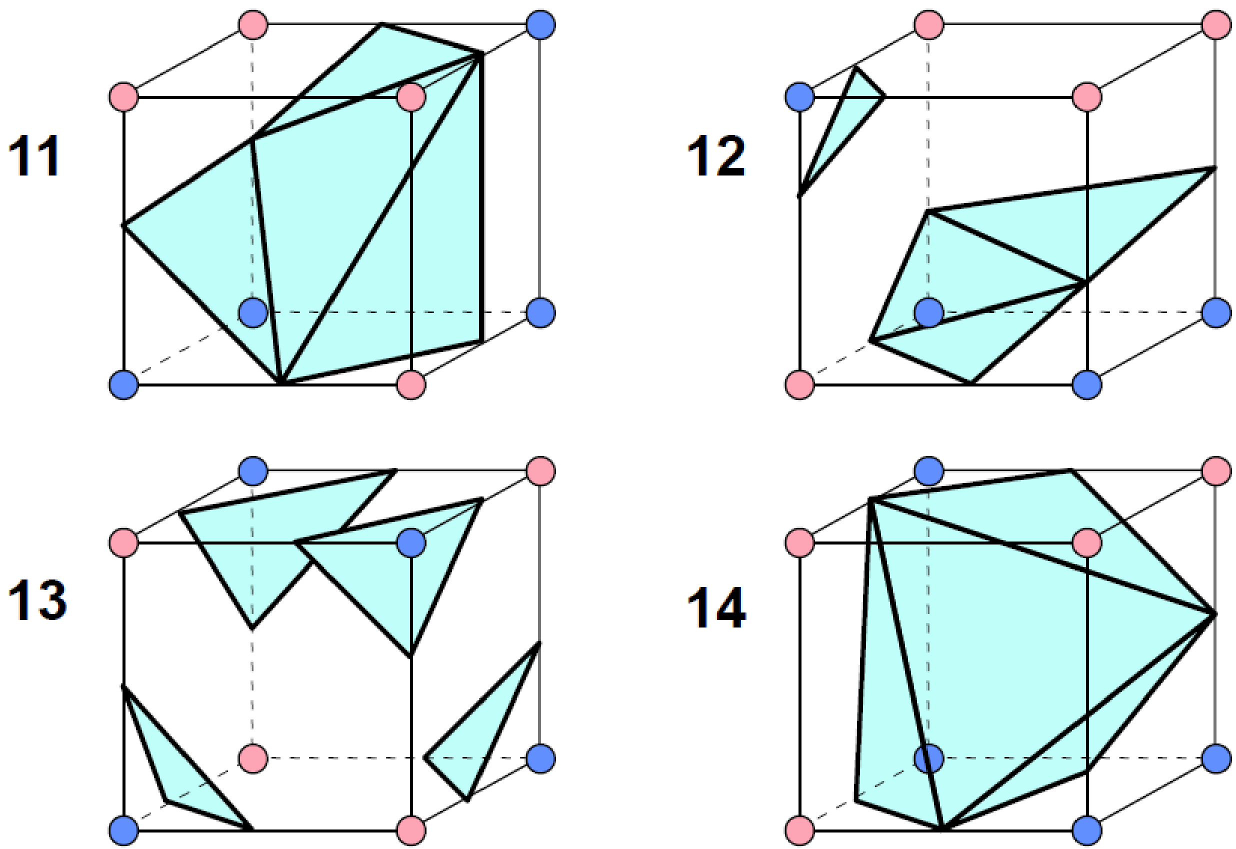
6



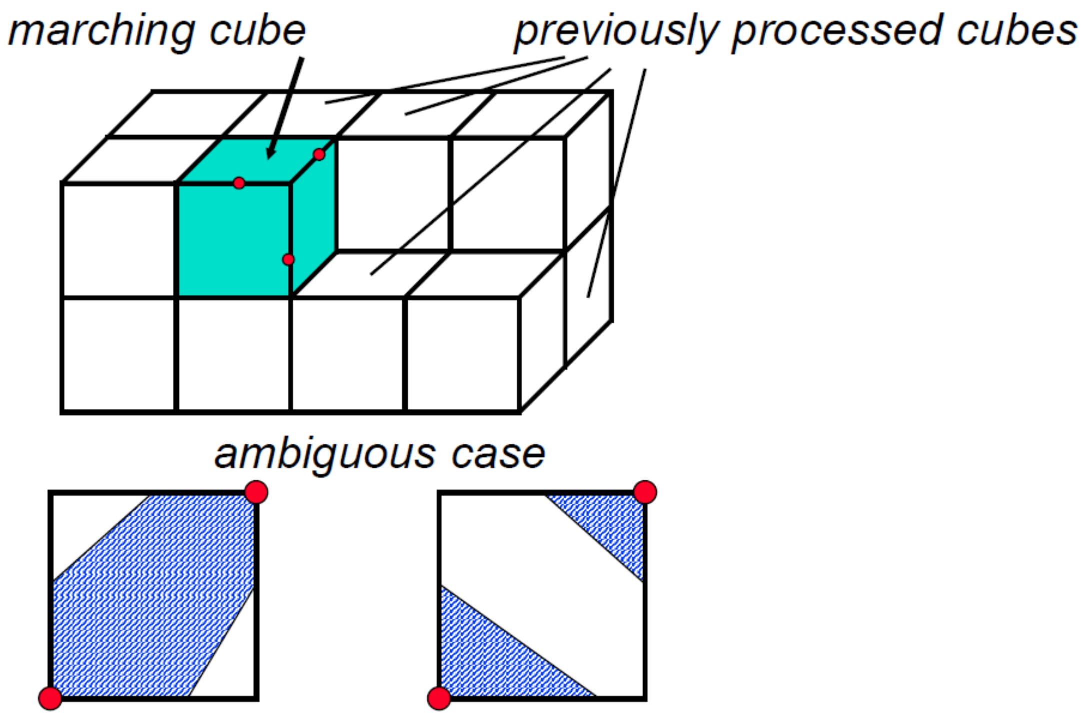
Marching cubes



Marching cubes



Marching cubes

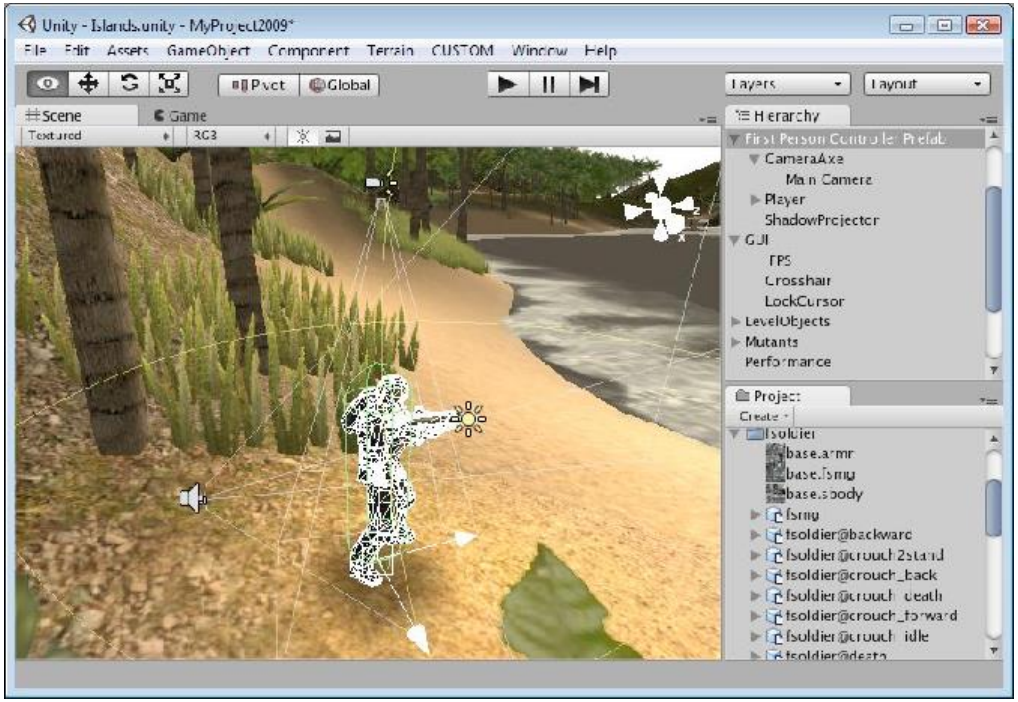


Outline

- Points
  - Range Image, Point Cloud
- Surfaces
  - Polygonal, Subdivision, Parametric, Implicit
- Solids
  - Voxels, BSP Tree, CSG, Sweep
- **Hierarchical Structures**
  - Scene graph, Application specific

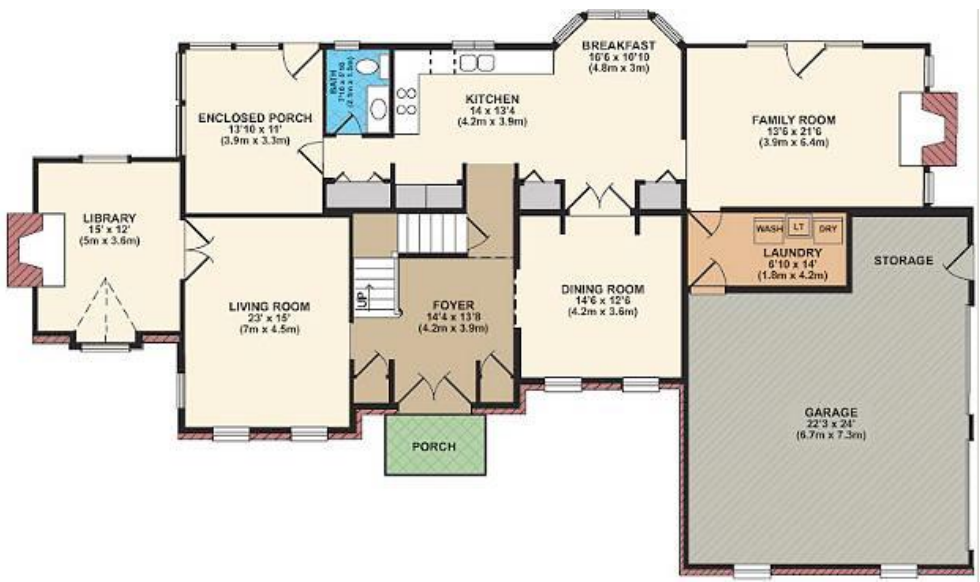
Scene Graph

- Objects organized in a hierarchical structure

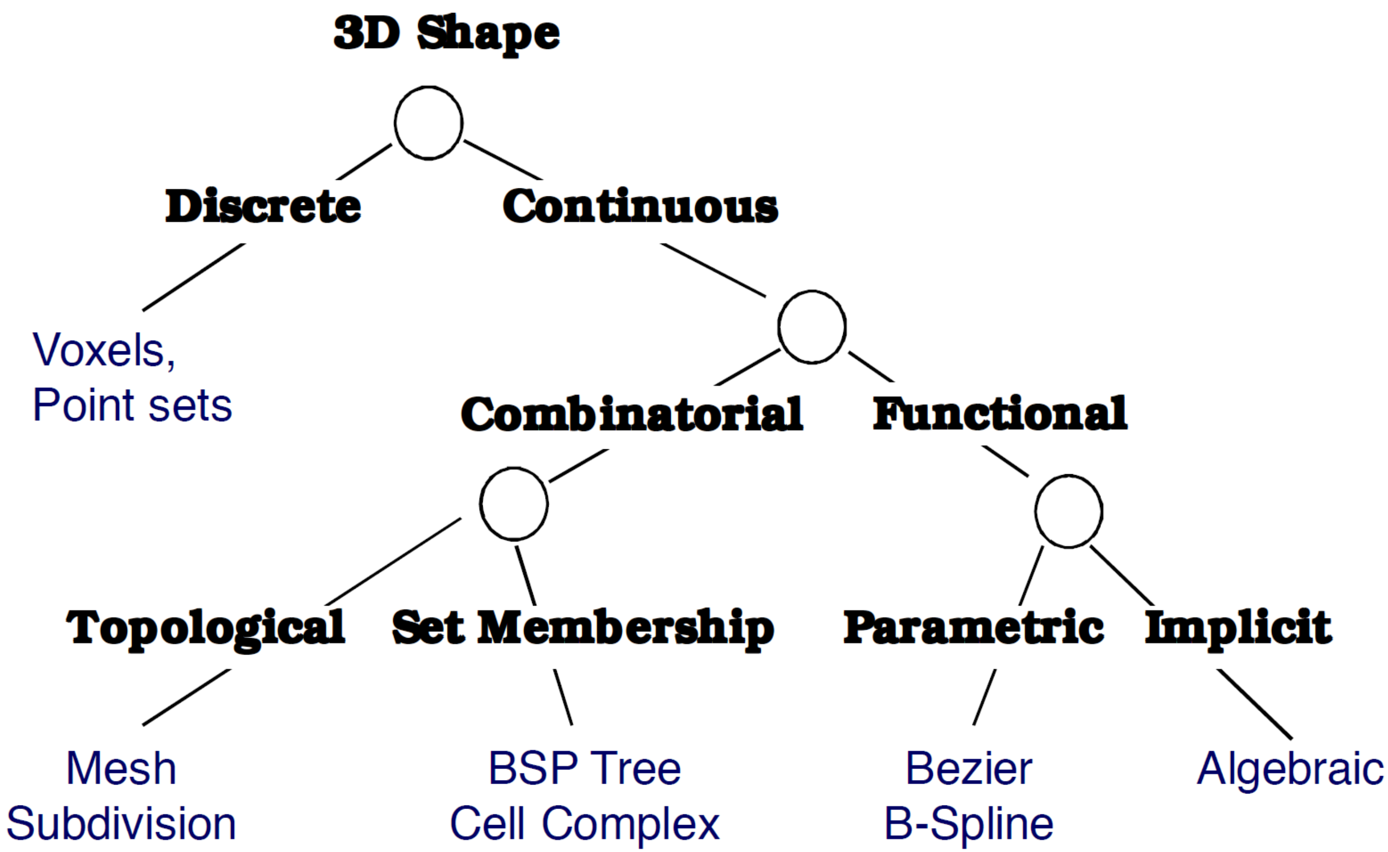


Application Specific

- Specific for given application domain



Taxonomy of 3D representations



Computational Differences

- Efficiency
  - Computational complexity (O(n log n))
  - Space/Time trade-off
  - Numerical stability/accuracy
- Simplicity
  - Hardware acceleration
  - Ease of acquisition
  - Software creation and maintenance
- Usability
  - Designer vs. computational engine

Parametric vs. polygonal

- Parametric or implicit
  - smooth, re-parametrizable
  - harder rendering
  - precise rendering

- Polygonal
  - discrete, hard to re-parametrize
  - faster rendering or rasterization
  - approximative rendering

How the lectures should look like #2

- Ask questions, please!!!
- Be communicative
- More active you are, the better for you!

Next Lecture

Transformations

Questions ?!

Skeletex
RESEARCH

www.skeletex.xyz

madaras@skeletex.xyz

martin.madaras@fmph.uniba.sk