3D Modeling

COS 426, Spring 2014

Princeton University

Syllabus

I. Image processing

II. Modeling

III. Rendering

IV. Animation

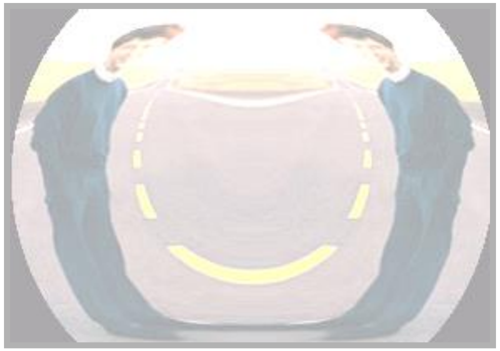
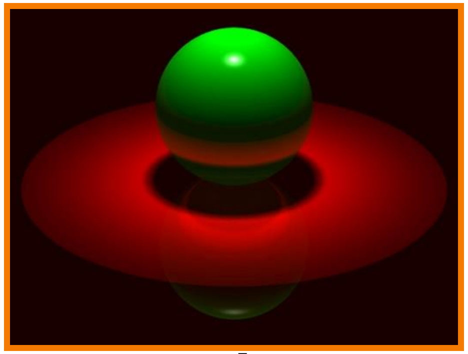
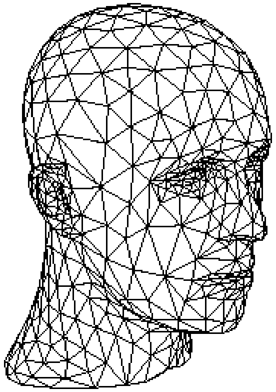


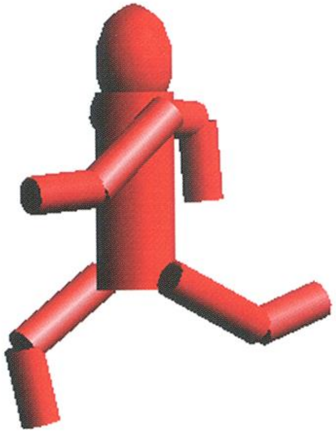
Image Processing
(Rusty Coleman, CS426, Fall99)



Rendering
(Michael Bostock, CS426, Fall99)



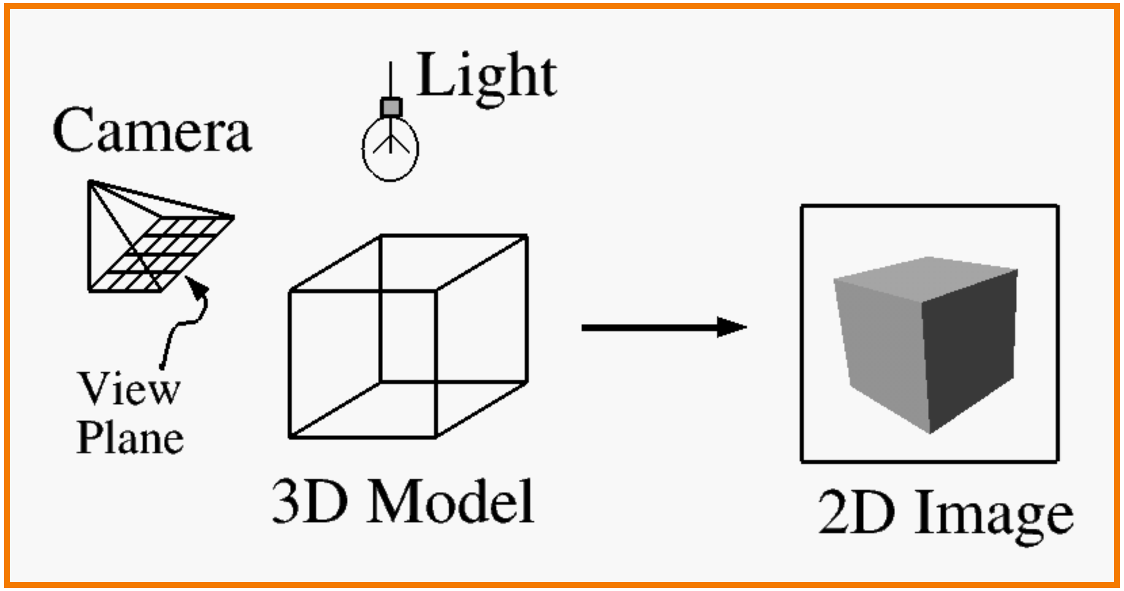
Modeling
(Dennis Zorin, CalTech)



Animation
(Angel, Plate 1)

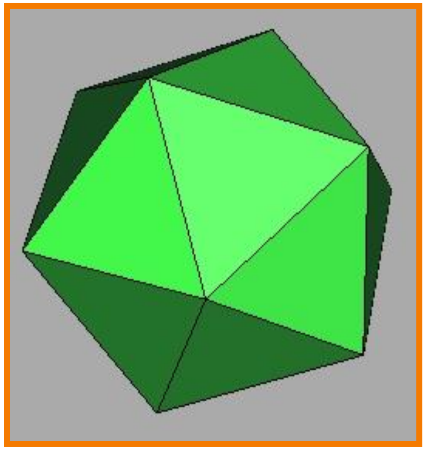
What is 3D Modeling?

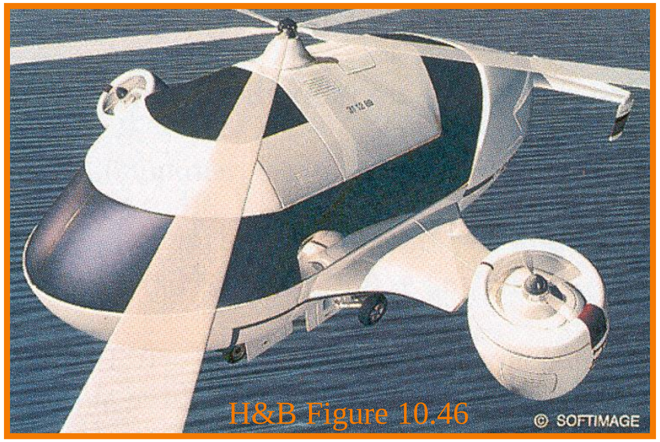
- Topics in computer graphics
  - Imaging = *representing 2D images*
  - Rendering = *constructing 2D images from 3D models*
  - Modeling = *representing 3D objects*
  - Animation = *simulating changes over time*

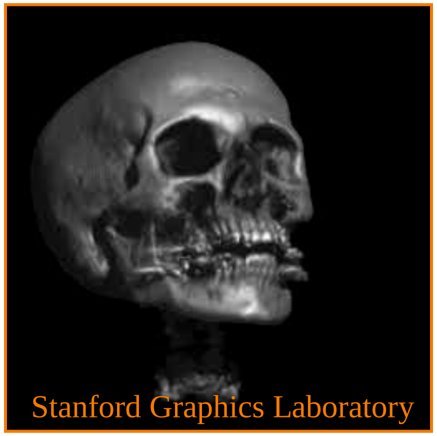


Modeling

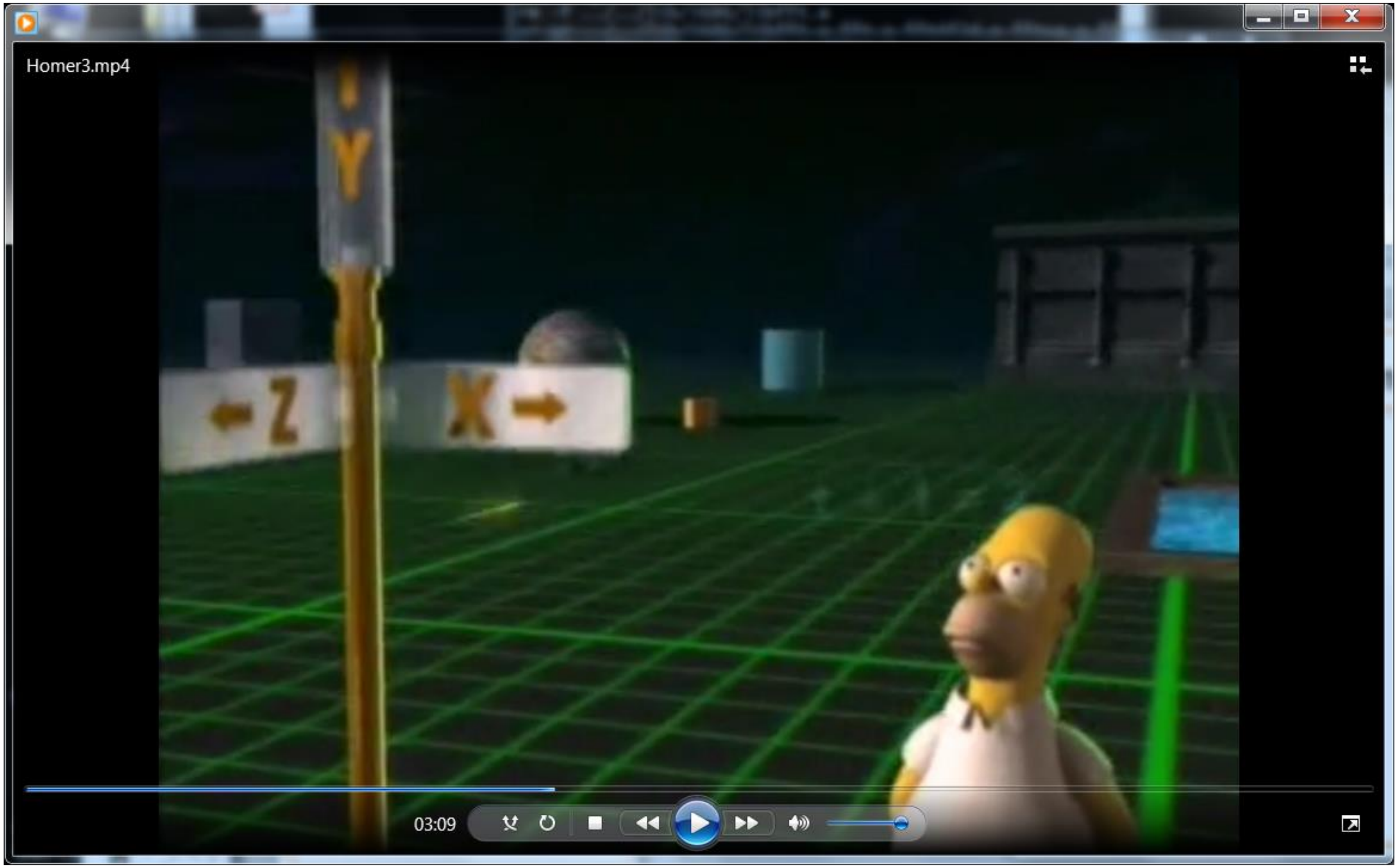
- How do we ...
  - Represent 3D objects in a computer?
  - Acquire computer representations of 3D objects?
  - Manipulate computer representations of 3D objects?







Modeling Background



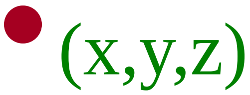
Modeling Background

- Scene is usually approximated by 3D primitives
  - Point
  - Vector
  - Line segment
  - Ray
  - Line
  - Plane
  - Polygon

3D Point

- Specifies a location
  - Represented by three coordinates
  - Infinitely small

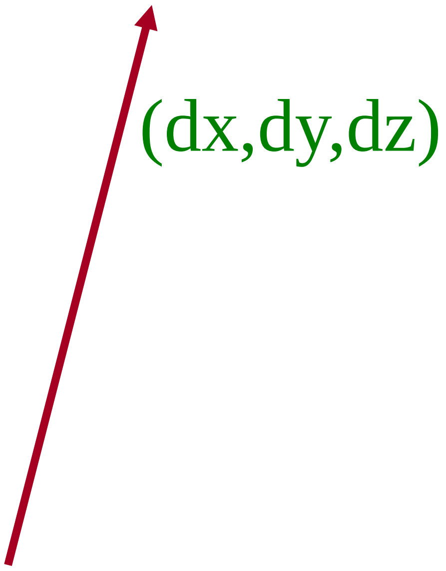
```
typedef struct {
    Coordinate x;
    Coordinate y;
    Coordinate z;
} Point;
```



3D Vector

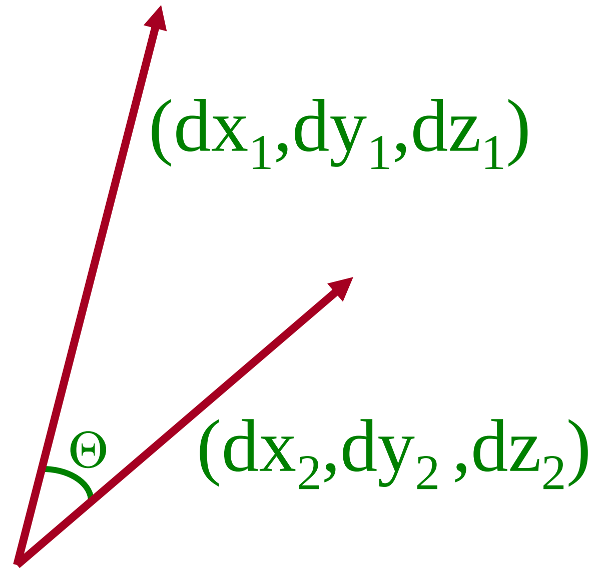
- Specifies a direction and a magnitude
  - Represented by three coordinates
  - Magnitude ||V|| = sqrt(dx dx + dy dy + dz dz)
  - Has no location

```
typedef struct {
    Coordinate dx;
    Coordinate dy;
    Coordinate dz;
} Vector;
```



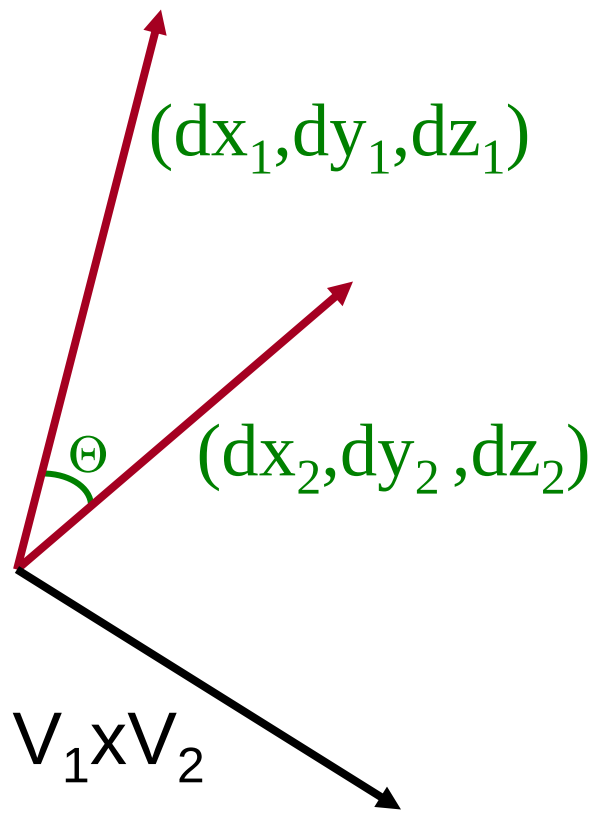
3D Vector

- Dot product of two 3D vectors
  - $V_1 \cdot V_2 = ||V_1|| \; || V_2 || \cos(\Theta)$



3D Vector

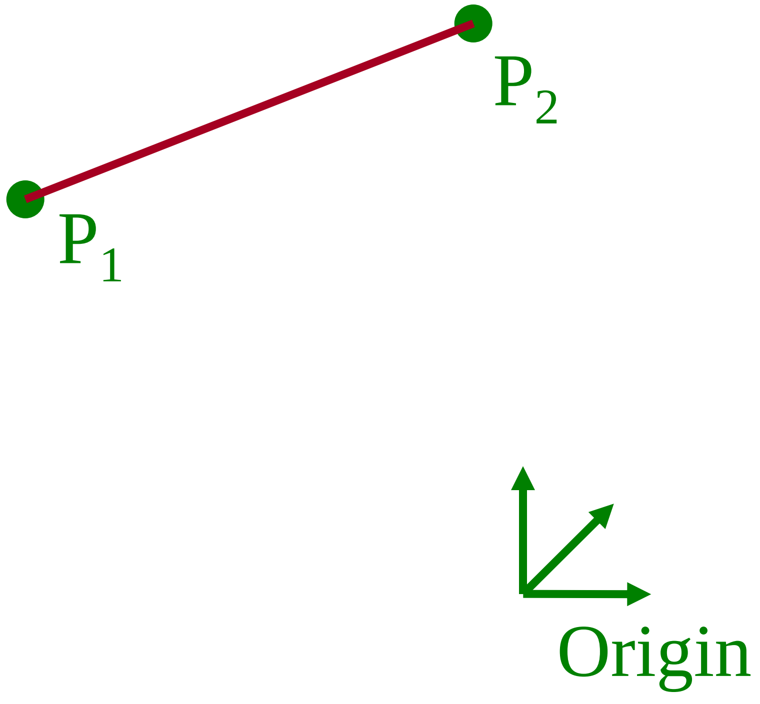
- Cross product of two 3D vectors
  - $V_1 \cdot V_2 = (dy_1dx_2 - dz_1dy_2, dz_1dx_2 - dx_1dz_2, dx_1dy_2 - dy_1dx_2)$
  - $V_1 x V_2$ = vector perpendicular to both V_1 and V_2
  - $||V_1 x V_{2}|| = ||V_1|| \, || V_2|| \sin(\Theta)$



3D Line Segment

- Linear path between two points
  - Parametric representation:
    - » $P = P_1 + t\,(P_2 - P_1), \quad (0 \leq t \leq 1)$

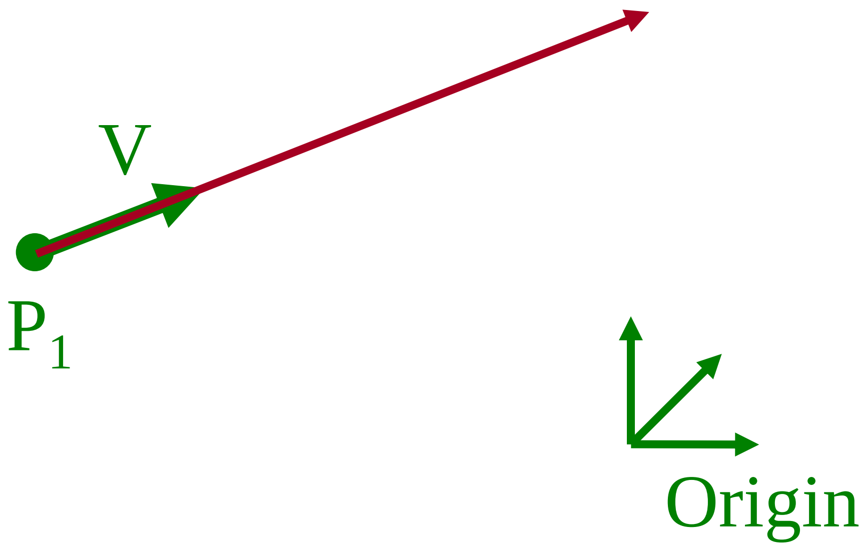
```
typedef struct {
    Point P1;
    Point P2;
} Segment;
```



3D Ray

- Line segment with one endpoint at infinity
  - Parametric representation:
    - » $P = P_1 + t\,V, \quad (0 <= t < \infty)$

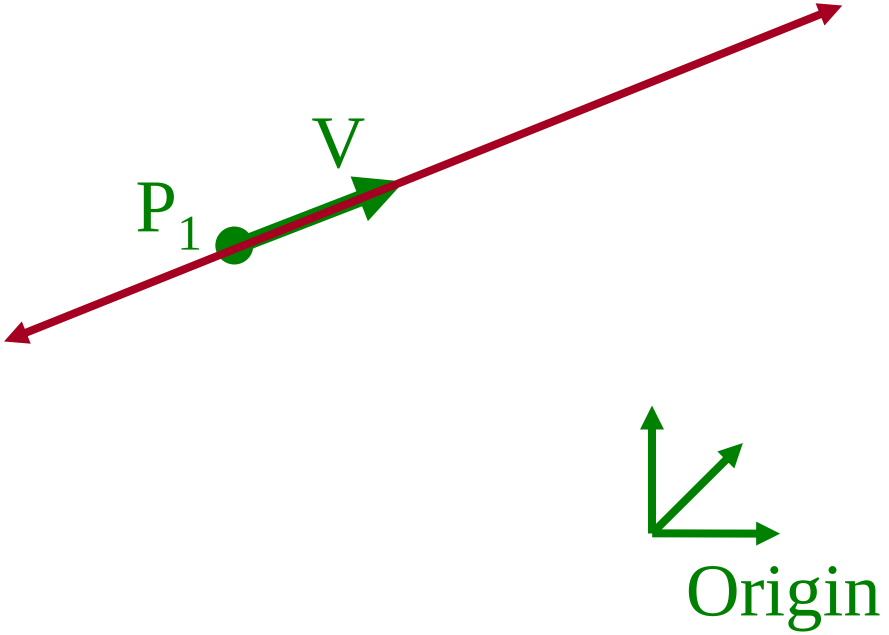
```
typedef struct {
    Point P1;
    Vector V;
} Ray;
```



3D Line

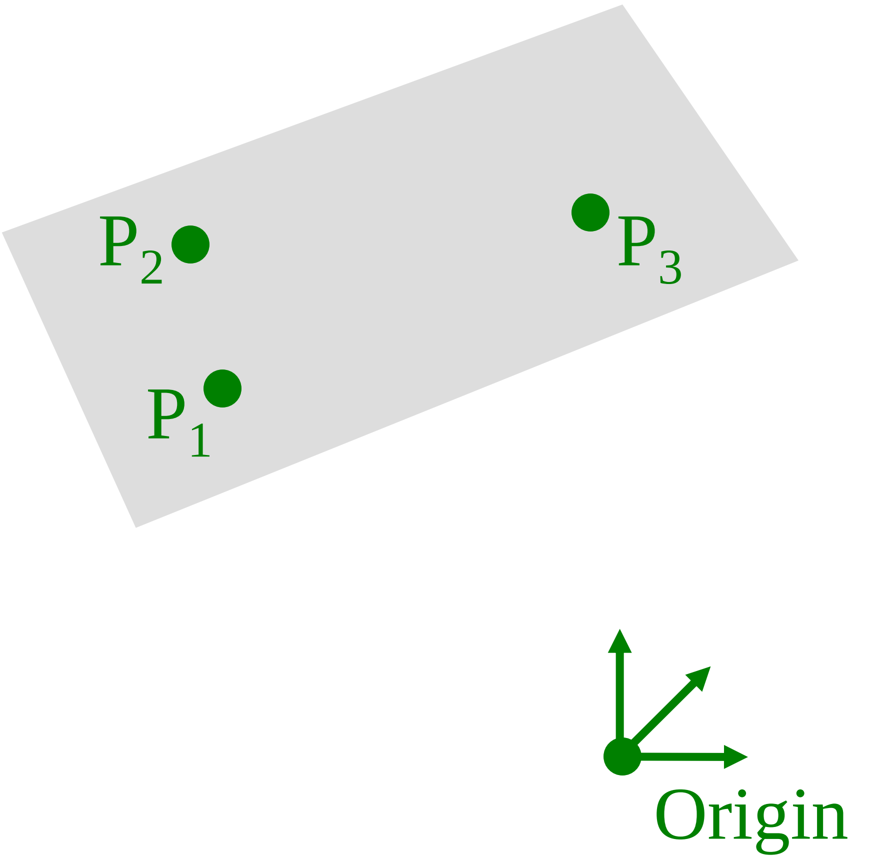
- Line segment with both endpoints at infinity
  - Parametric representation:
    - » $P = P_1 + t\,V, \quad (-\infty < t < \infty)$

```
typedef struct {
    Point P1;
    Vector V;
} Line;
```



3D Plane

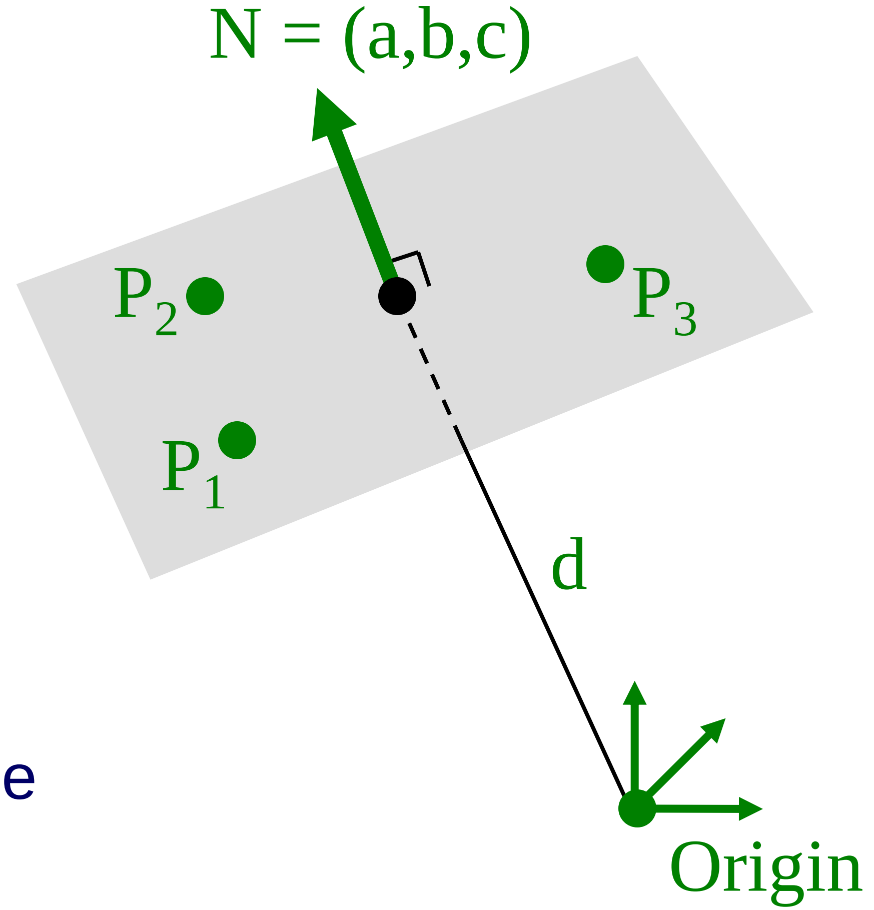
- A linear combination of three points



3D Plane

- A linear combination of three points
  - Implicit representation:
    - » $P \cdot N + d = 0$, or
    - » $ax + by + cz + d = 0$

```
typedef struct {
    Vector N;
    Distance d;
} Plane;
```

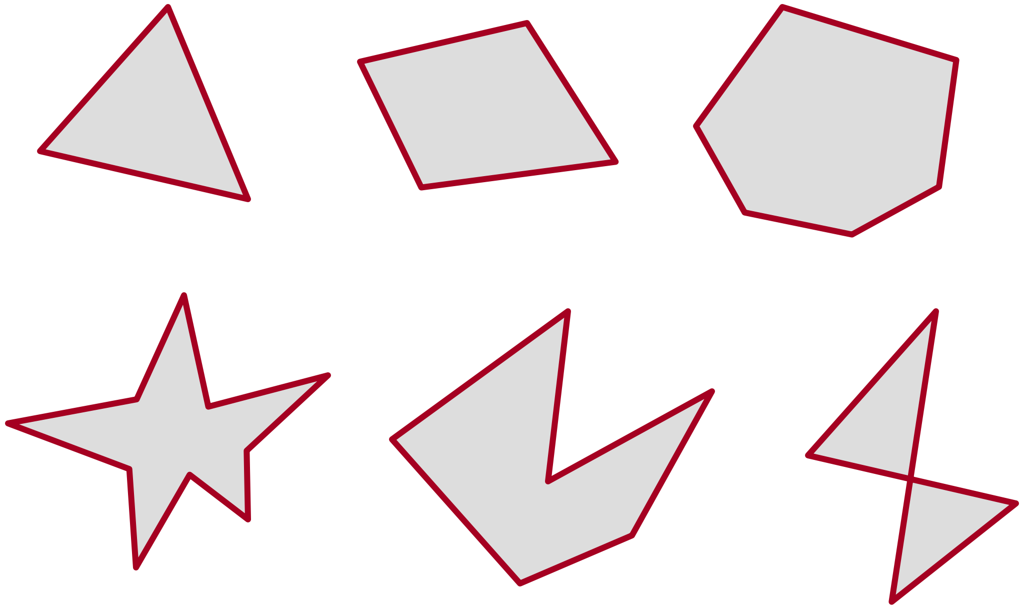


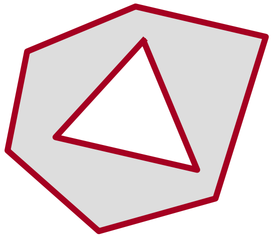
  - N is the plane “normal”
    - » Unit-length vector
    - » Perpendicular to plane

3D Polygon

- Set of points "inside" a sequence of coplanar points

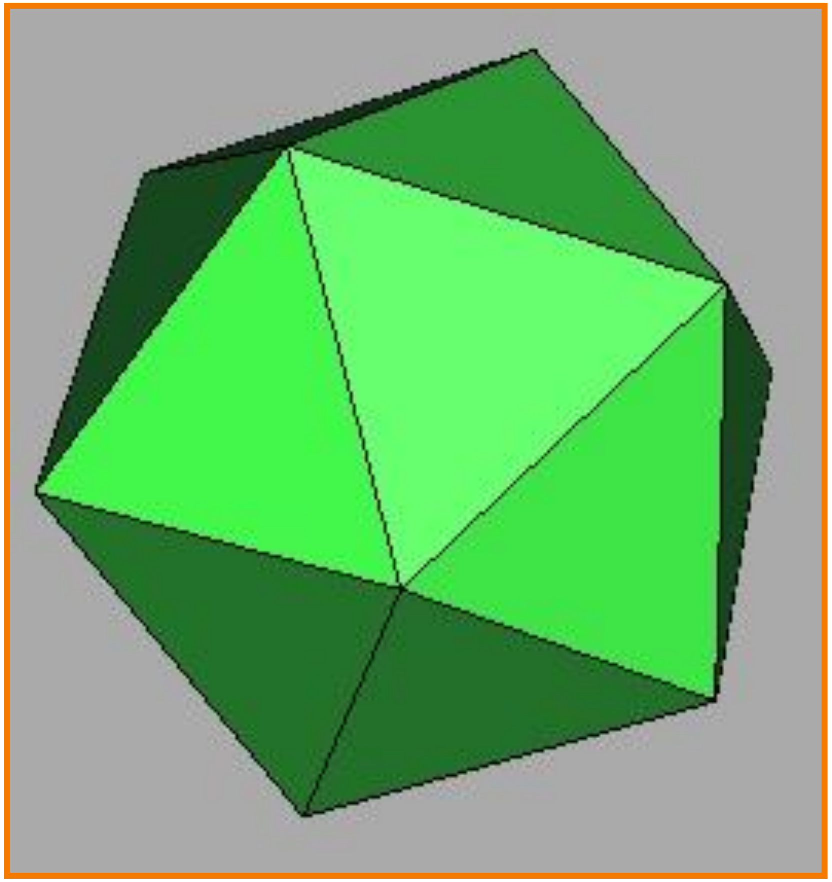
```
typedef struct {
    Point *points;
    int npoints;
} Polygon;
```





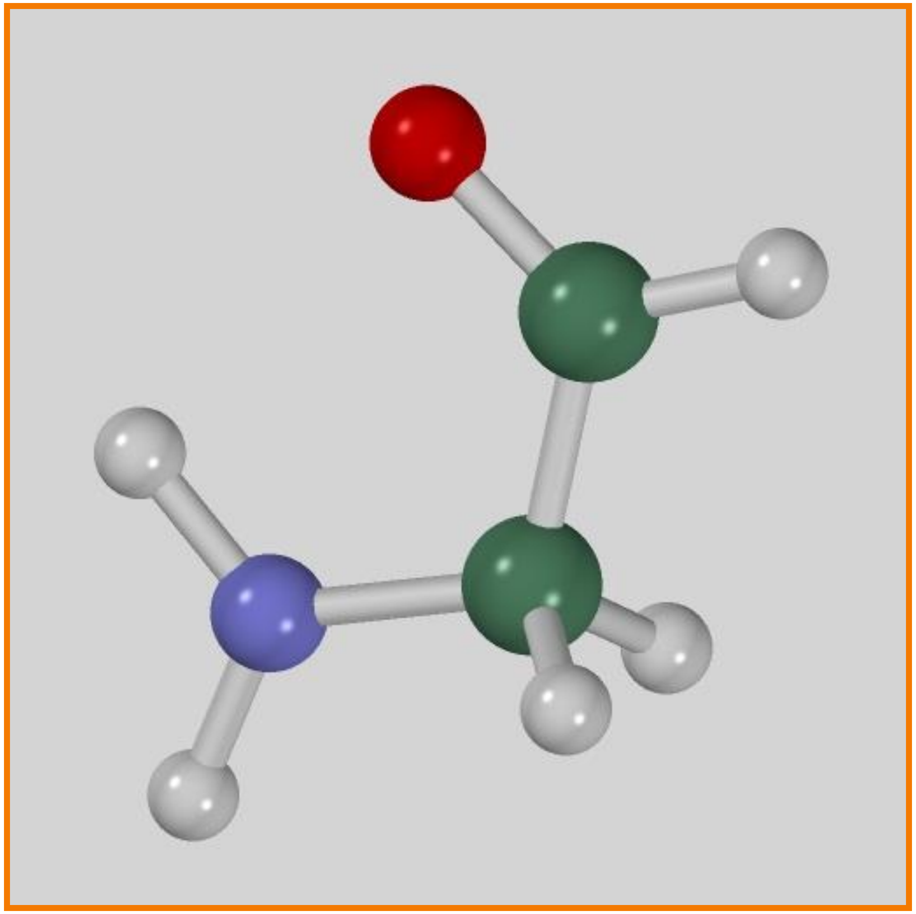
Points are in counter-clockwise order

3D Object Representations



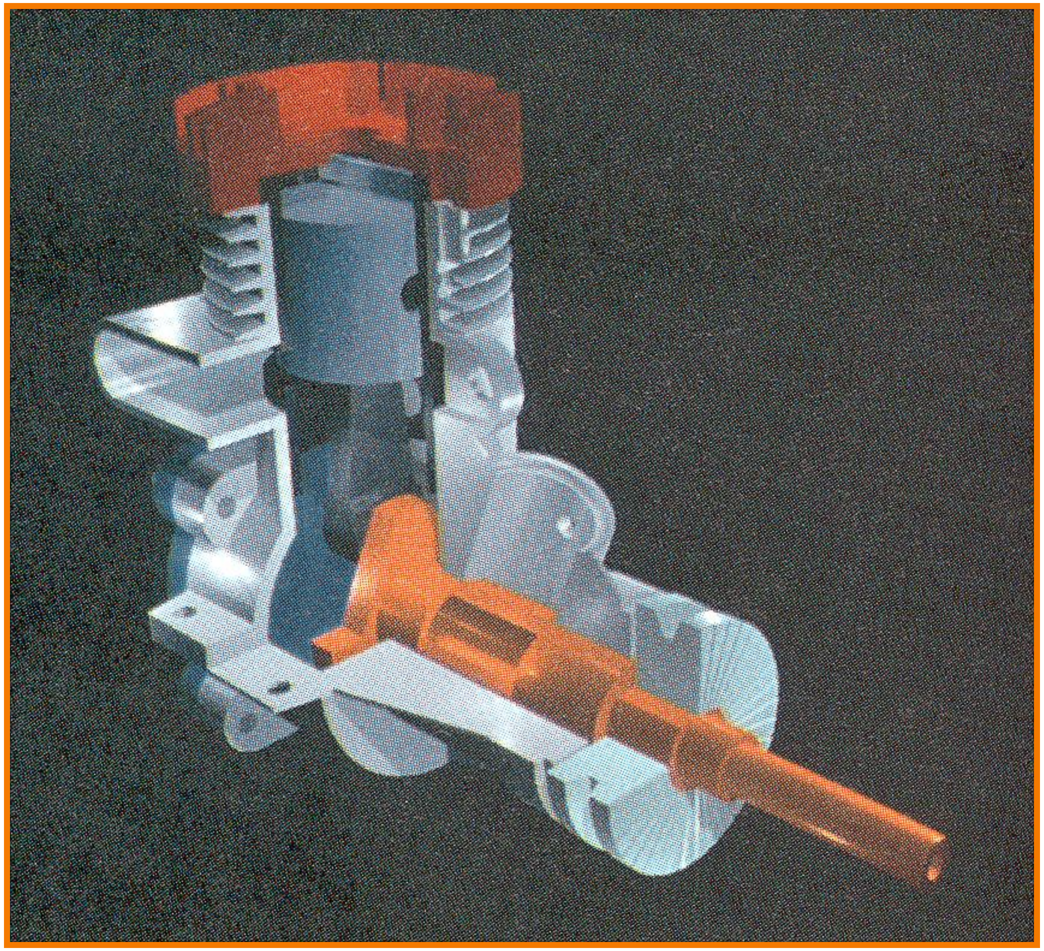
How can this object be represented in a computer?

3D Object Representations



How about this one?

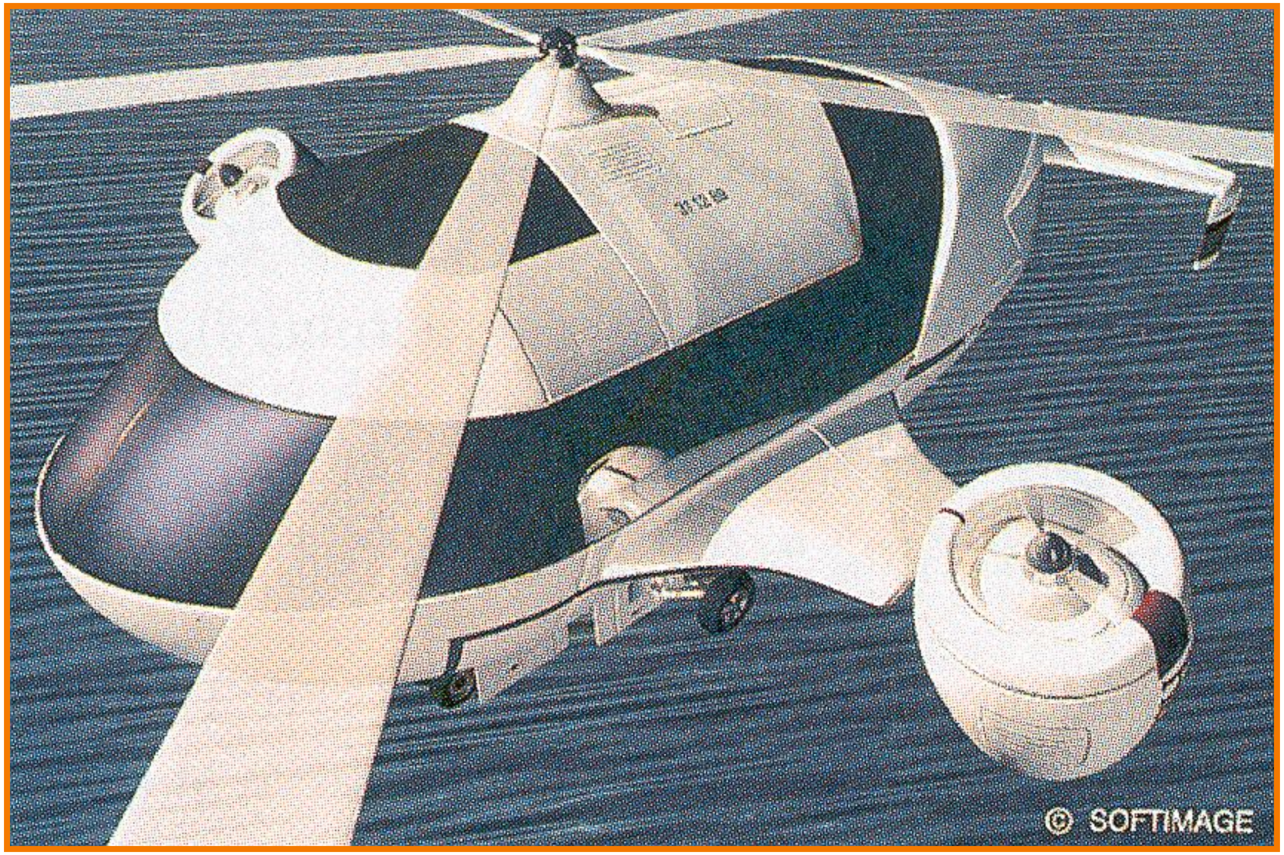
3D Object Representations



H&B Figure 9.9

This one?

3D Object Representations



H&B Figure 10.46

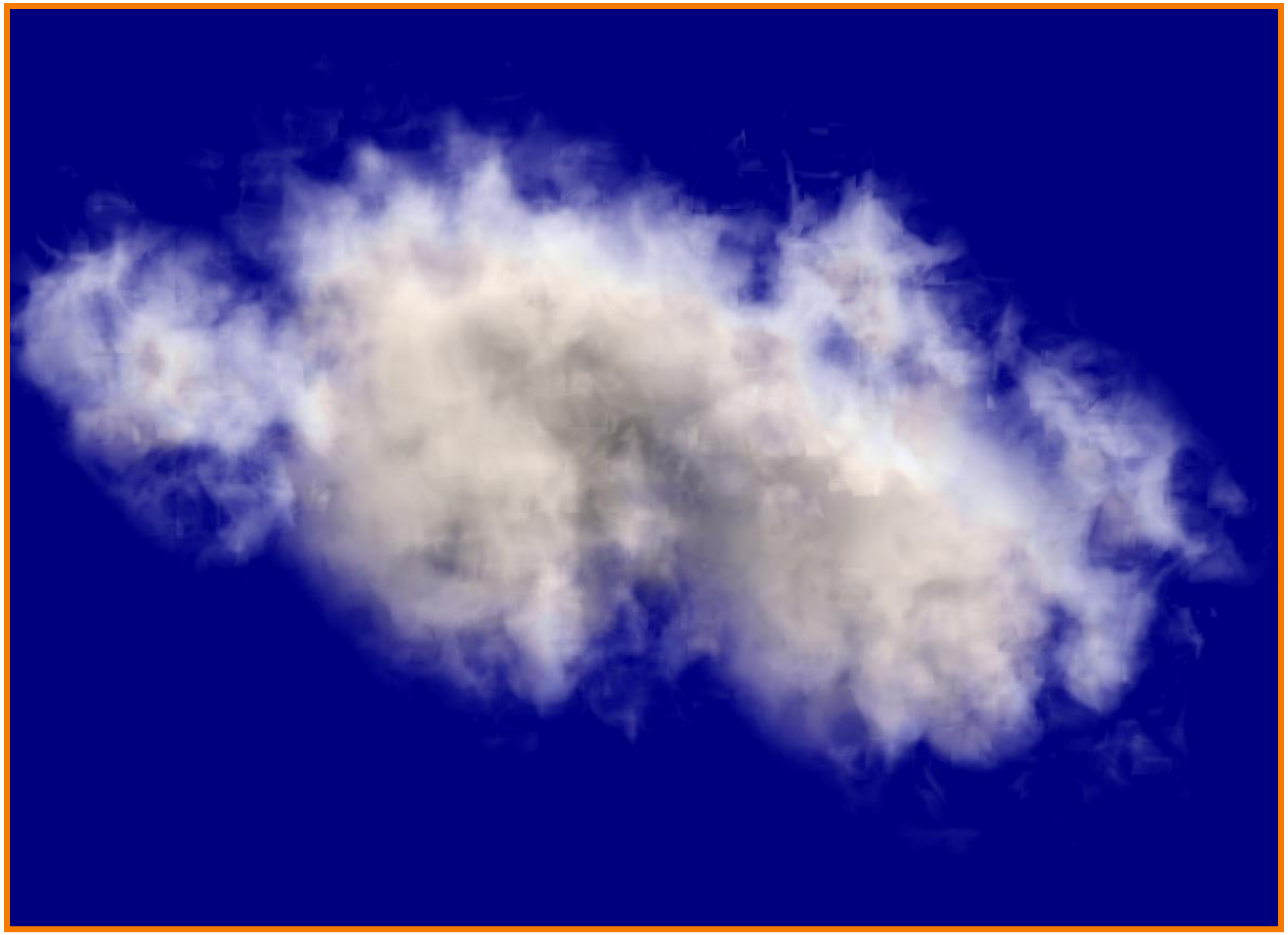
This one?

3D Object Representations

This one?

Stanford Graphics Laboratory

3D Object Representations



This one?

3D Object Representations

- Points
  - Range image
  - Point cloud
- Surfaces
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

Equivalence of Representations

- Thesis:
  - Each representation has enough expressive power to model the shape of any geometric object
  - It is possible to perform all geometric operations with any fundamental representation
- Analogous to Turing-equivalence
  - Computers and programming languages are Turing-equivalent, but each has its benefits…

Why Different Representations?

Efficiency for different tasks

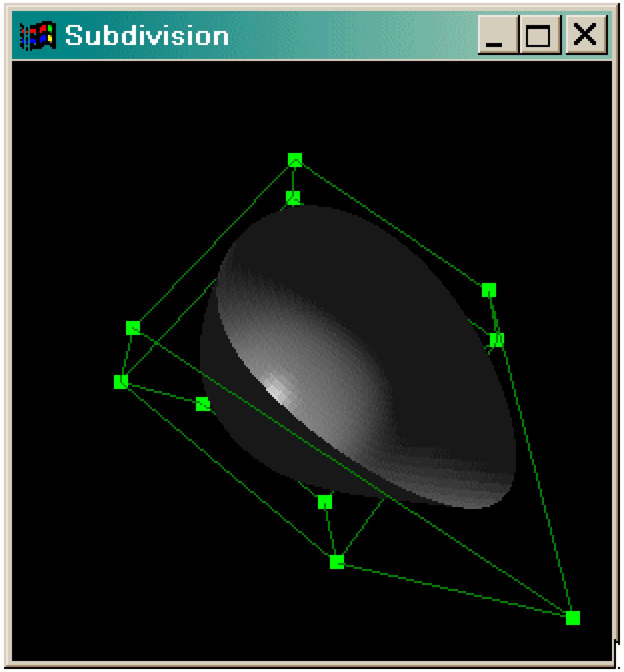
- Acquisition
- Rendering
- Manipulation
- Animation
- Analysis

Data structures determine algorithms

Why Different Representations?

Desirable properties depend on intended use

- Easy to acquire
- Accurate
- Concise
- Intuitive editing
- Efficient editing
- Efficient display
- Efficient intersections
- Guaranteed validity
- Guaranteed smoothness
- etc.



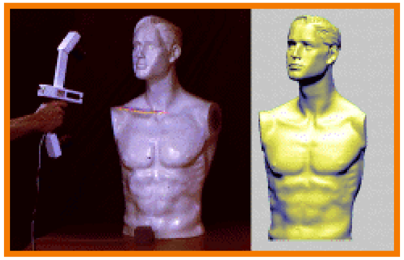
3D Object Representations

- Points
  - Range image
  - Point cloud
- Surfaces
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

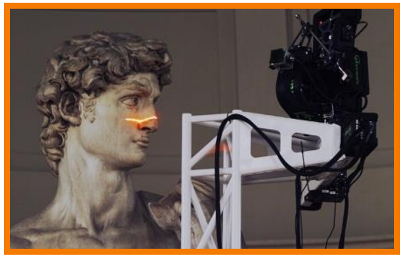
Range Image

Set of 3D points mapping to pixels of depth image

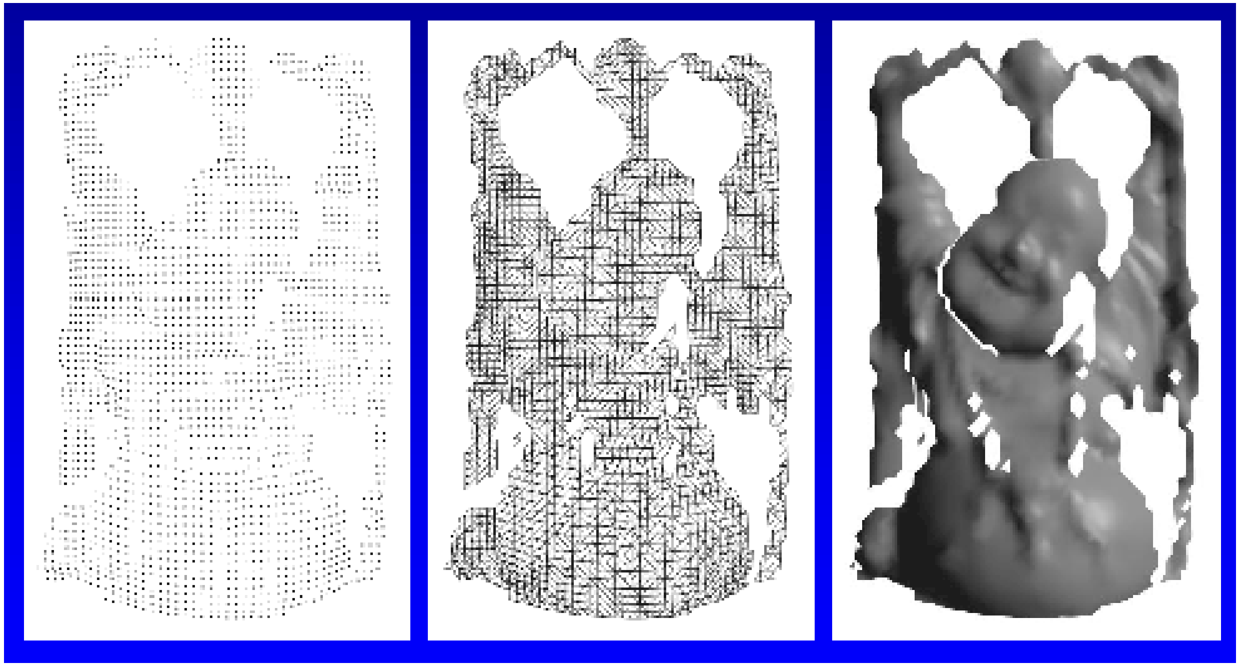
- Can be acquired from range scanner



Cyberware



Stanford



Range Image   Tesselation   Range Surface

Brian Curless
SIGGRAPH 99
Course #4 Notes

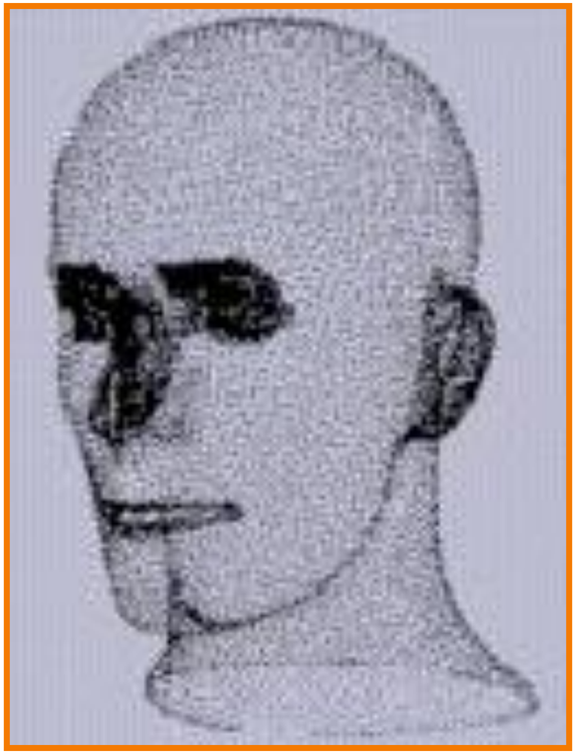
Point Cloud

Unstructured set of 3D point samples

- Acquired from range finder, computer vision, etc

Polhemus

Microscribe-3D



Hoppe

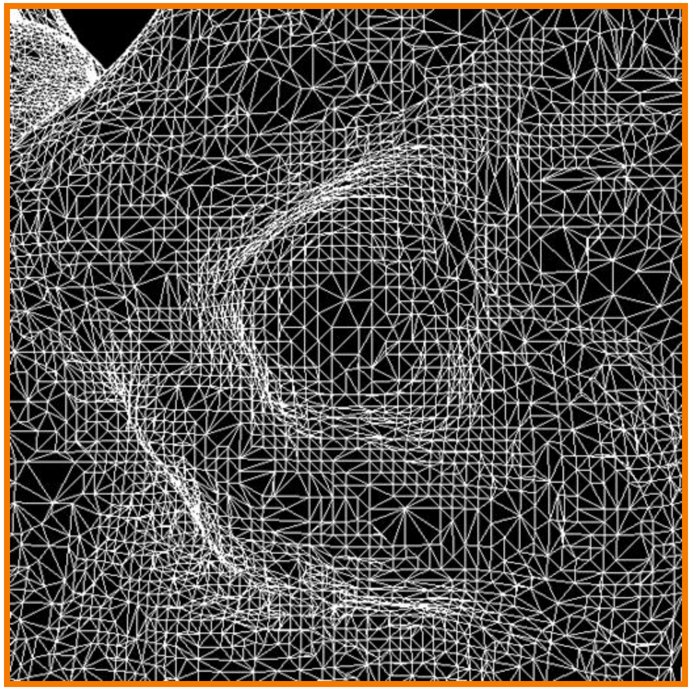
Hoppe

3D Object Representations

- Points
  - Range image
  - Point cloud
- Surfaces
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

Polygonal Mesh

Connected set of polygons (usually triangles)

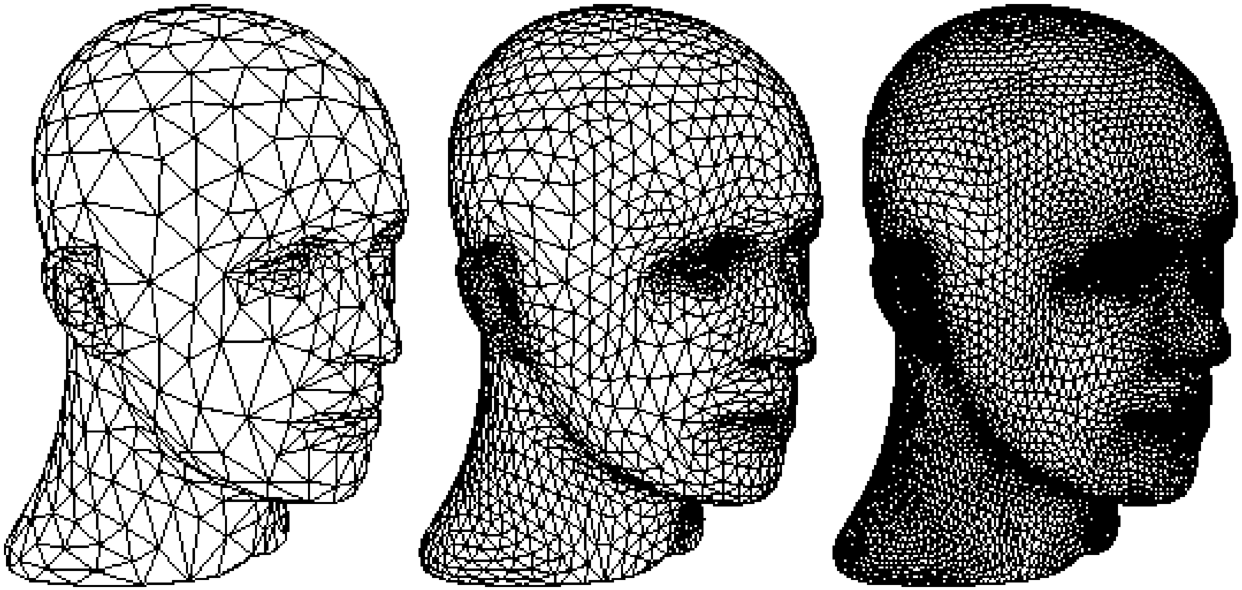


Stanford Graphics Laboratory

Subdivision Surface

Coarse mesh & subdivision rule

- Smooth surface is limit of sequence of refinements

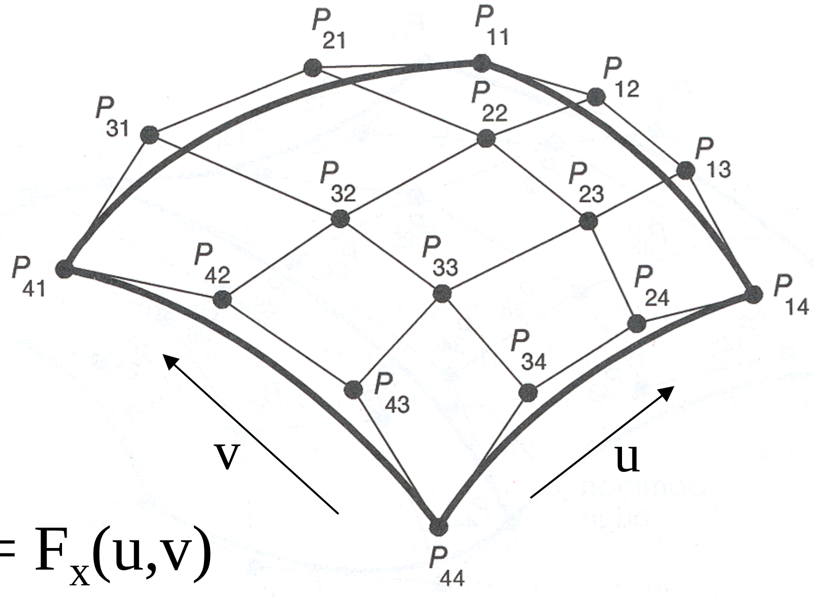


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Parametric Surface

Tensor-product spline patches

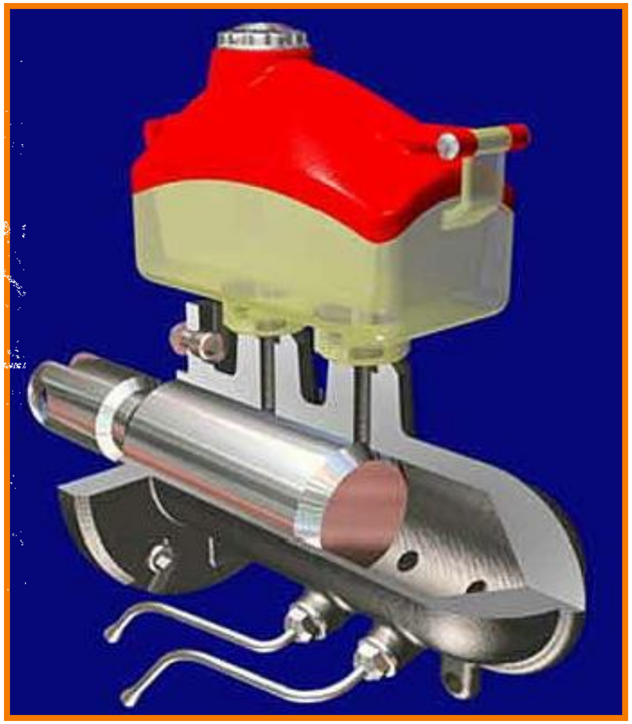
- Each patch is parametric function
- Careful constraints to maintain continuity



$x = F_x(u,v)$

$y = F_y(u,v)$

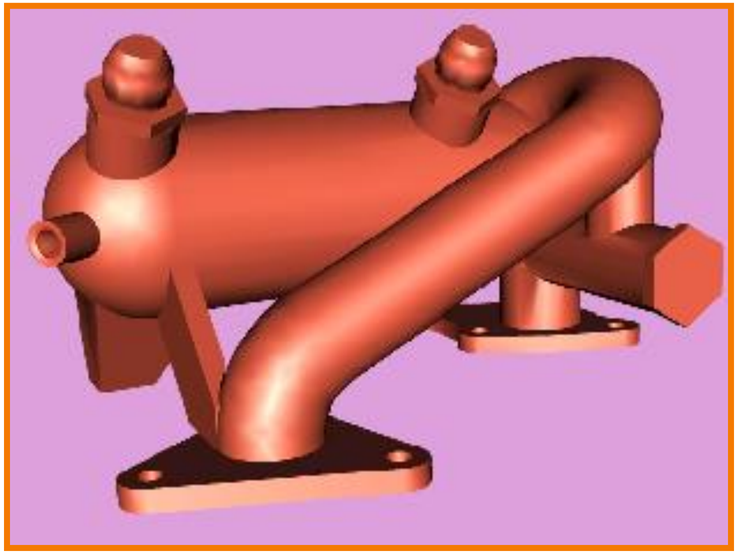
$z = F_z(u,v)$



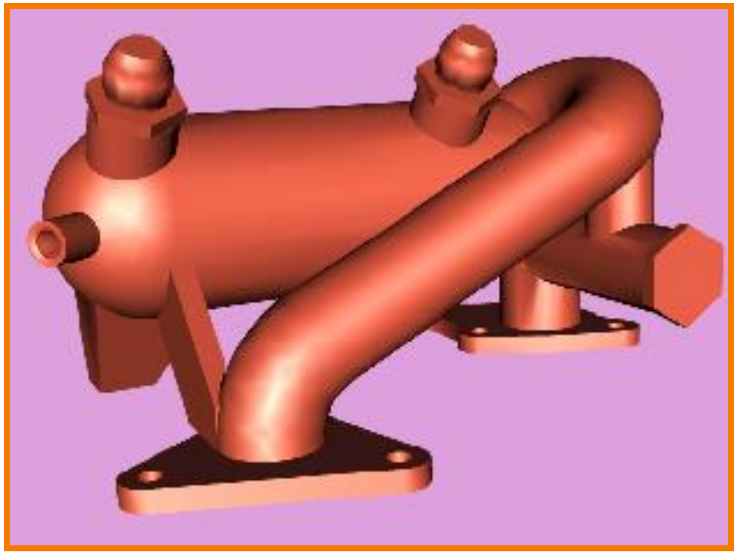
FvDFH Figure 11.44

Implicit Surface

Set of all points satisfying: $F(x,y,z) = 0$



Polygonal Model



Implicit Model

Bill Lorensen
SIGGRAPH 99
Course #4 Notes

3D Object Representations

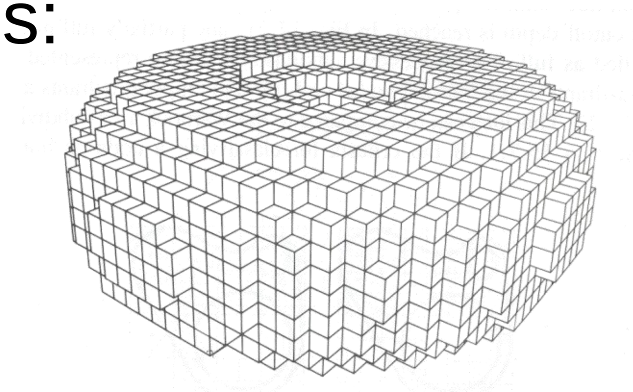
- Points
  - Range image
  - Point cloud
- Surfaces
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

Voxel grid

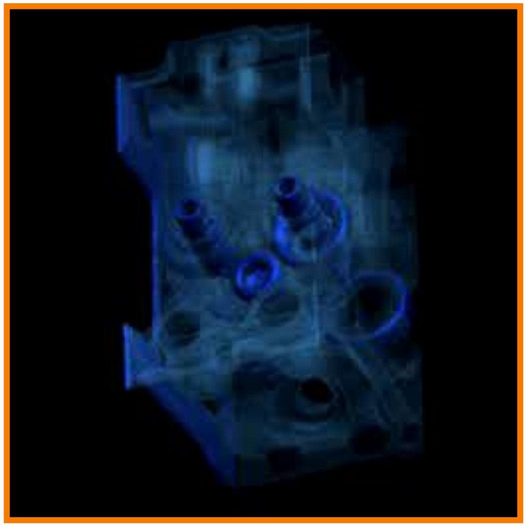
Uniform volumetric grid of samples:

- Occupancy
  (object vs. empty space)
- Density
- Color
- Other function
  (speed, temperature, etc.)

- Often acquired via
  simulation or from
  CAT, MRI, etc.



FvDFH Figure 12.20

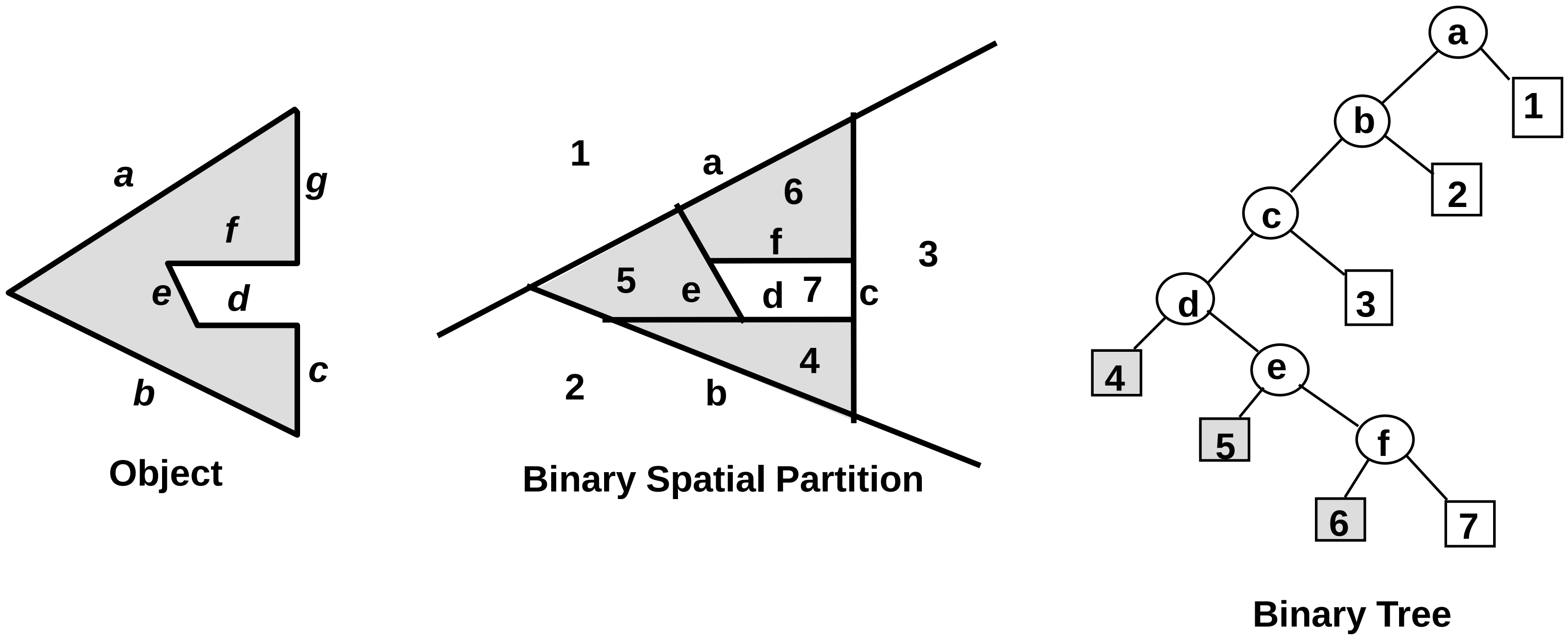


Stanford Graphics Laboratory

BSP Tree

Hierarchical **B**inary **S**pace **P**artition with solid/empty cells labeled

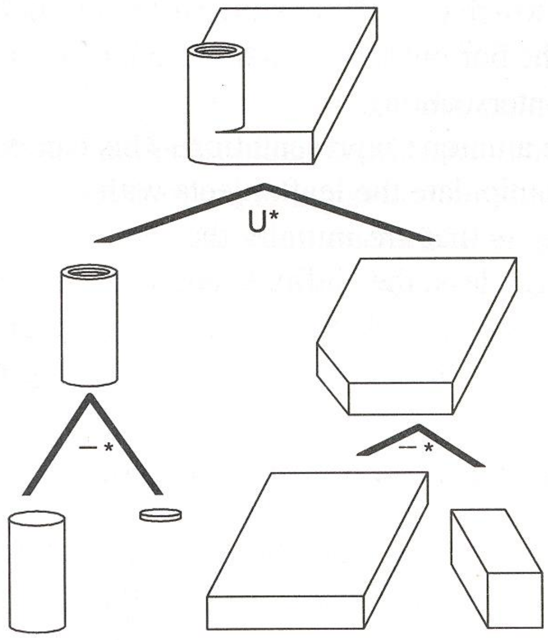
- Constructed from polygonal representations



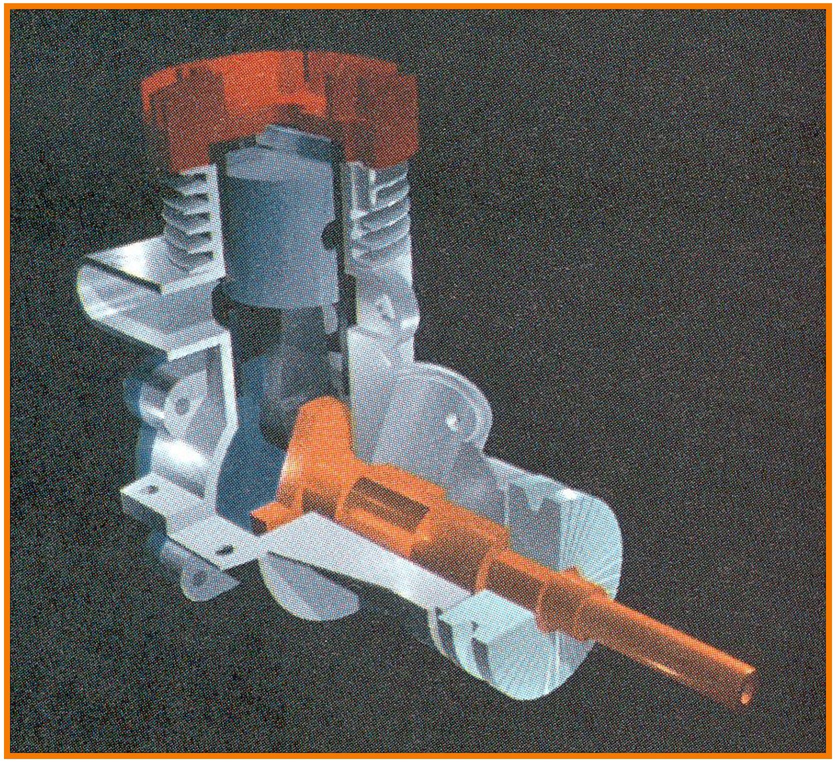
Naylor

CSG

Constructive **S**olid **G**eometry: set operations (union, difference, intersection) applied to simple shapes



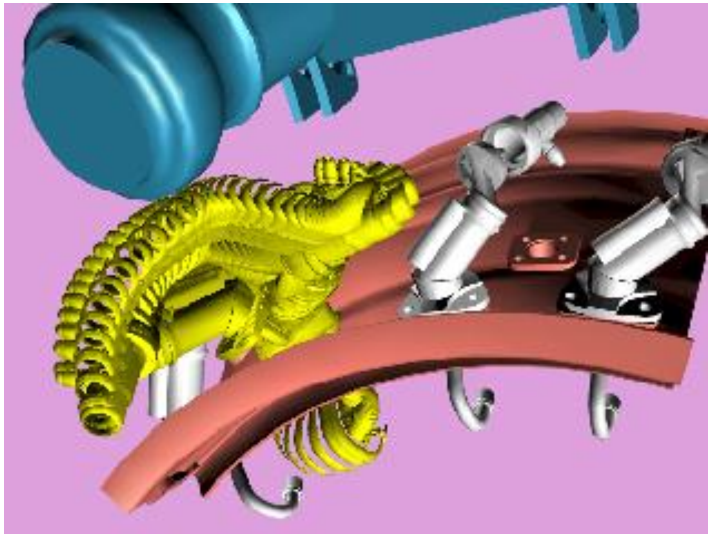
FvDFH Figure 12.27



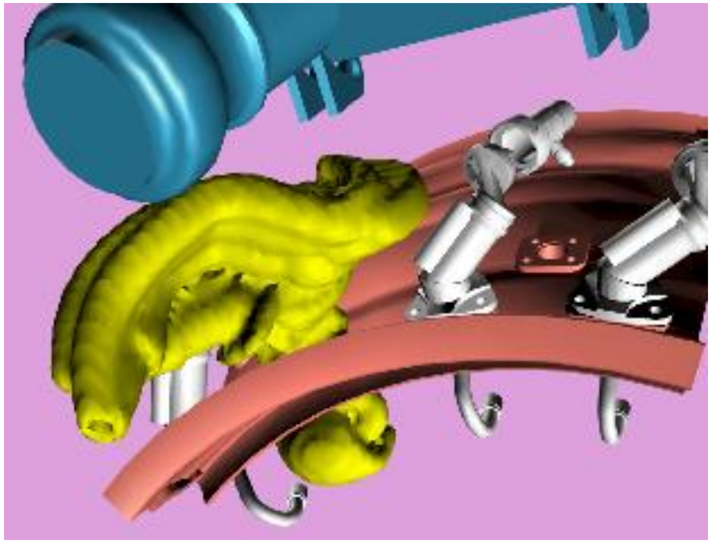
H&B Figure 9.9

Sweep

Solid swept by curve along trajectory



Removal Path



Sweep Model

Bill Lorensen
SIGGRAPH 99
Course #4 Notes

3D Object Representations

- Points
  - Range image
  - Point cloud
- Surfaces
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

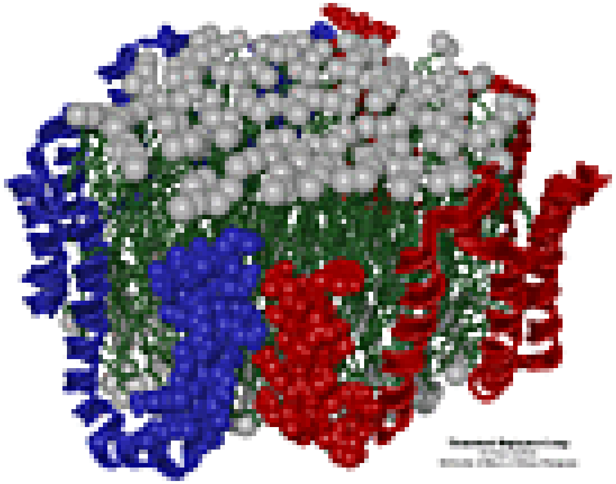
Scene Graph

Union of objects at leaf nodes

Bell Laboratories

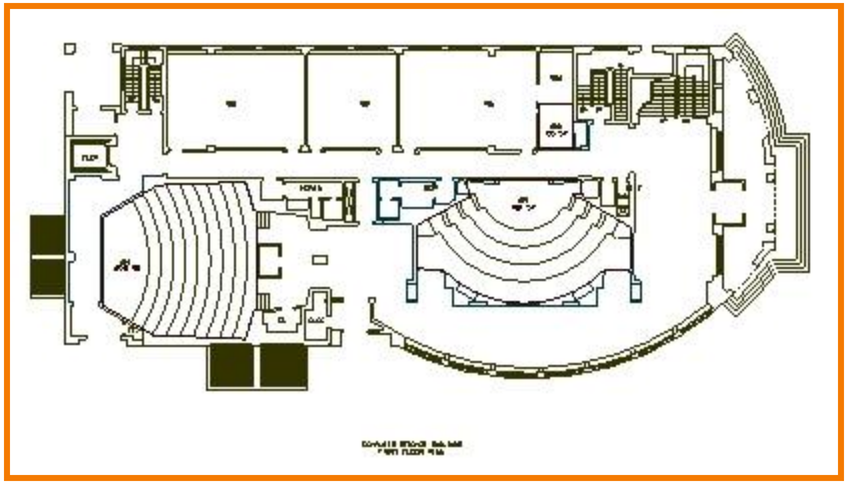
avalon.viewpoint.com

Application Specific



Apo A-1

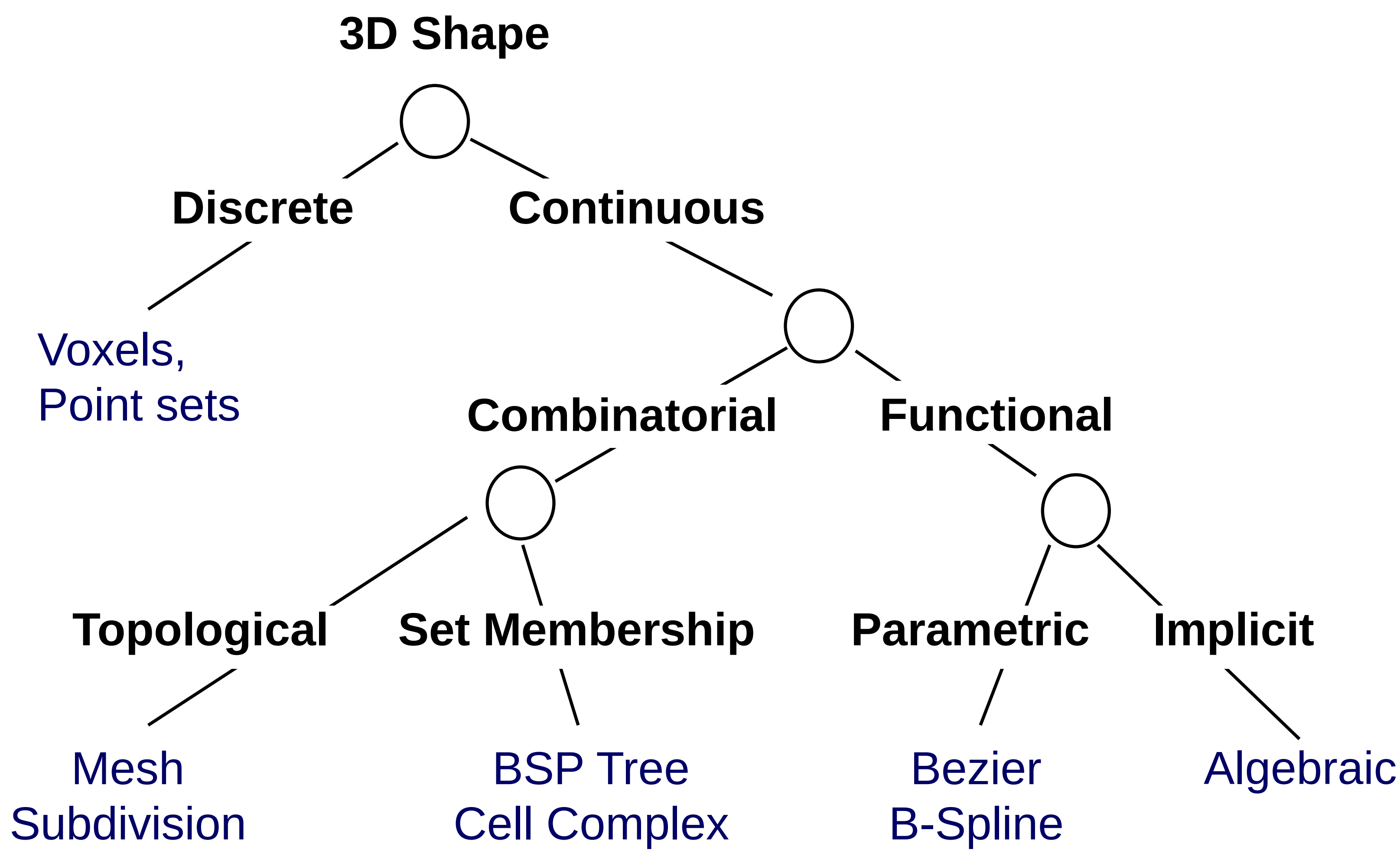
(Theoretical Biophysics Group,
University of Illinois at Urbana-Champaign)



Architectural Floorplan

(CS Building, Princeton University)

Taxonomy of 3D Representations



Naylor

Equivalence of Representations

- Thesis:
  - Each representation has enough expressive power to model the shape of any geometric object
  - It is possible to perform all geometric operations with any fundamental representation
- Analogous to Turing-equivalence
  - Computers and programming languages are Turing-equivalent, but each has its benefits…

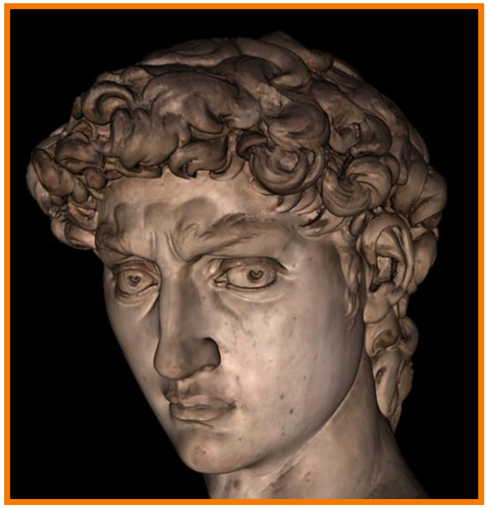
Computational Differences

- Efficiency
  - Representational complexity (e.g. volume vs. surface)
  - Computational complexity (e.g. $O(n^2)$ vs $O(n^3)$)
  - Space/time trade-offs (e.g. z-buffer)
  - Numerical accuracy/stability (e.g. degree of polynomial)
- Simplicity
  - Ease of acquisition
  - Hardware acceleration
  - Software creation and maintenance
- Usability
  - Designer interface vs. computational engine

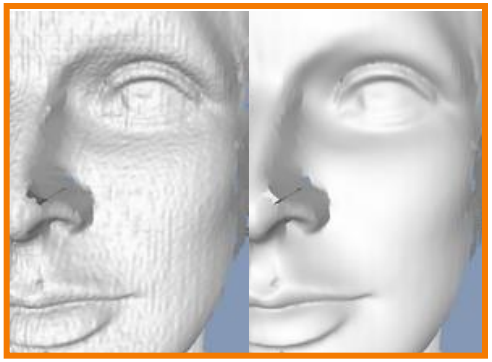
Modeling Operations

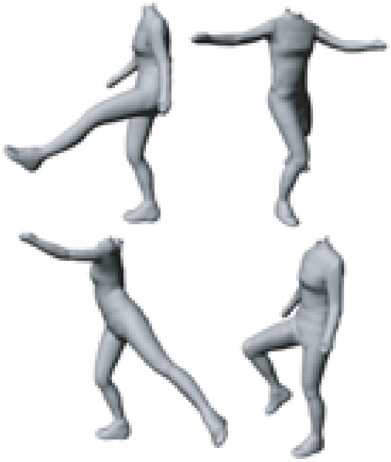
What can we do with a 3D object representation?

- Edit
- Transform
- Smooth
- Render
- Animate
- Morph
- Compress
- Transmit
- Analyze
- etc.


Digital Michelangelo


Pirates of the Caribbean


Thouis "Ray" Jones


Sand et al.

Upcoming Lectures

- Points
  - Range image
  - Point cloud
- Surfaces
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific