Modeling

CMSC 435/634

Modeling?

Modeling

Creating a *model* of an object, usually out of a collection of simpler *primitives*

Primitive

A basic shape handled directly the rendering system

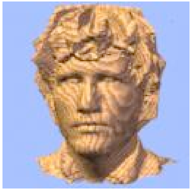
Primitives

Some common primitives

- Triangles & Polygons
  - Most common, usually the only choice for interactive
- Patches, Spheres, Cylinders, ...
  - Often converted to simpler primitives within the renderer
- Volumes
  - What's at each point in space?
  - Often with some transparent material
  - Few renderers handle both volume & surface models

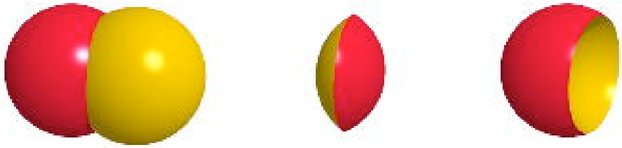
Composing primitives

- Collections of large numbers of primitives
  - Sometimes called Boundary Representation (*BRep*)



Composing primitives

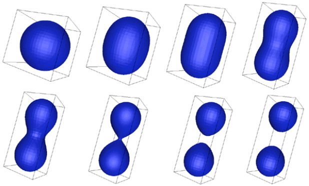
- Collections of large numbers of primitives
  - Sometimes called Boundary Representation (*BRep*)
- Constructive Solid Geometry (*CSG*)
  - Set operations (union, intersection, difference)



Images: Friedrich Lohmueller

Composing primitives

- Collections of large numbers of primitives
  - Sometimes called Boundary Representation (*BRep*)
- Constructive Solid Geometry (*CSG*)
  - Set operations (union, intersection, difference)
- Implicit Models & Blobs
  - Surface where f(x,y,z)=0
  - Sum, product, etc. of simpler functions



Images: Paul Bourke

Mesh Representations

Definitions

- Vertex: all data at a point
  - Position
  - Normal
  - Texture coordinates
  - Color
  - May count as new vertex if *any* of these differ
- Edge: Line between vertices
- Face: Area between a set of vertices and edges
  - Assume planar
  - May have fixed # vertices, may not

Mesh Representations

Application-friendly

- Polygon list
- (or whatever makes sense to the application)

Hardware-friendly

- Vertex list
- Vertex + Index lists

Mesh editing-friendly

- Face-Vertex
- Winged Edge
- Half Edge

Hybrid

Application-Friendly: Polygon List

How to make it

- Define a polygon object
- Put a bunch of them in a list

Pros

- Flexible
- Fits application needs

Cons

- Hard to figure out how polygons are connected
- Duplication of vertex data
- Inefficient to render

Hardware-friendly: Vertex Array

How to make it

- Make a list of vertices
- Every 3 form a triangle

Pros

- Relatively efficient to render

Cons

- Hard to figure out how faces are connected
- Duplication of vertex data
- Fixed number of vertices per polygon

Hardware-friendly: Vertex and Index Arrays

How to make it

- Make a list of vertices
- Make a list of which vertices connect into triangles
- Every 3 indices make a triangle

Pros

- Very efficient to render
- Share vertex data
- Finding vertices in a face easy

Cons

- Finding faces that use a vertex is hard
- Finding adjacent faces is hard
- Fixed number of vertices per polygon

Mesh editing-friendly: Face-Vertex

How to make it

- Vertex: position, list of faces
- Face: list of vertices

Pros

- Finding vertices in a face easy
- Finding faces that use a vertex is easy

Cons

- Finding adjacent faces is hard

Mesh editing-friendly: Winged-edge

How to make it

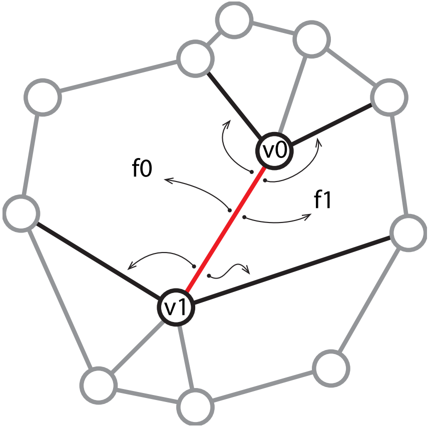
- Edge (primary structure)
  - Two vertices, two faces
  - Next and previous edges on both faces
- Vertex: position, list of edges
- Face: list of edges

Pros

- Finding vertices in a face easy
- Finding faces that use a vertex is easy
- Finding adjacent faces is easy

Cons

- Big: lots of redundant links



Half-edge

How to make it

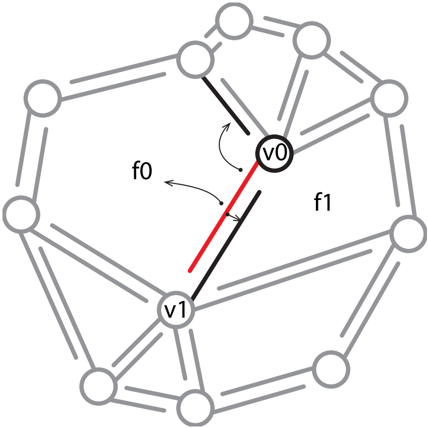
- Half-Edge (primary structure)
    - One vertex, one face
    - Pointer to pair edge
    - Next edge around face
- Face: pointer to (any) half-edge
- Vertex: pointer to (any) half-edge

Pros

- Adjacent faces
- Edges around face or vertex

Cons

- Lots of bookkeeping to update



Hybrid

Maintain multiple representations

- Separate vertex location from pointers
- Update face during edits

Delayed updates

- Do mesh updates, then rebuild index/vertex list
- Do other partial updates, then rebuild
- Traverse and build

Modeling Approaches

Manual

Procedural

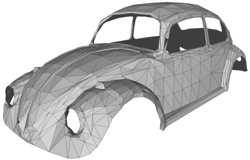
Scan

Data

Images

Manual Creation

- Text editor
  - Only very simple primitives and scenes
- High-level primitives
  - Still need to combine several somehow
- Modeling programs
  - Maya, 3D Studio, Houdini, Autocad, Blender, ...



Procedural Modeling

- Describe physical attributes through code
  - Shape
    - Output primitives
  - Density
    - Voxels
    - Couple with a conversion or rendering algorithm
  - Color, Texture
    - Enhance an existing shape

Procedural Approaches

- Fractals
- Implicit Functions
- Grammars
- Simulations

Fractals

Complex structure through self-similarity across scales

- Recursive structure
- Small features look *similar* to larger features

Iterated Equations / Mandelbrot Set

$$p' = p^2 + c$$

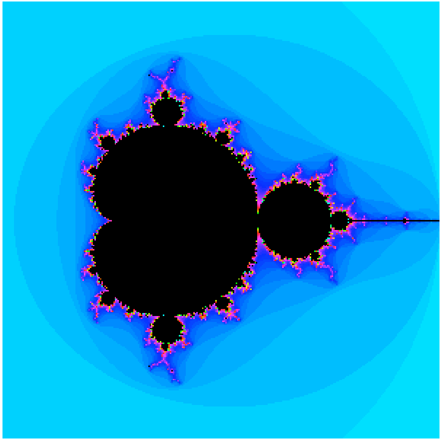
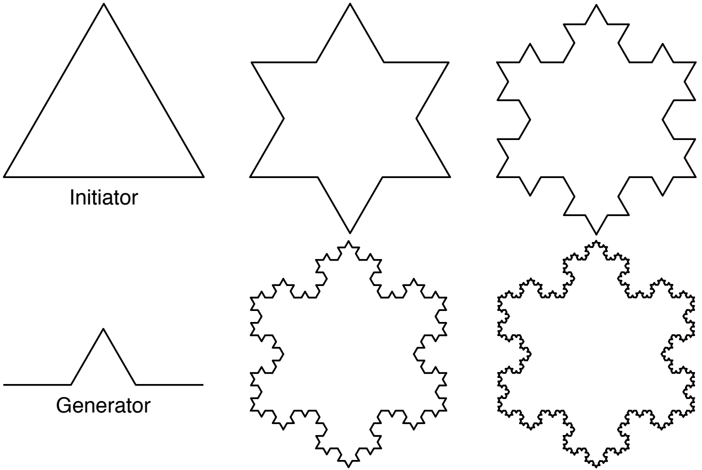


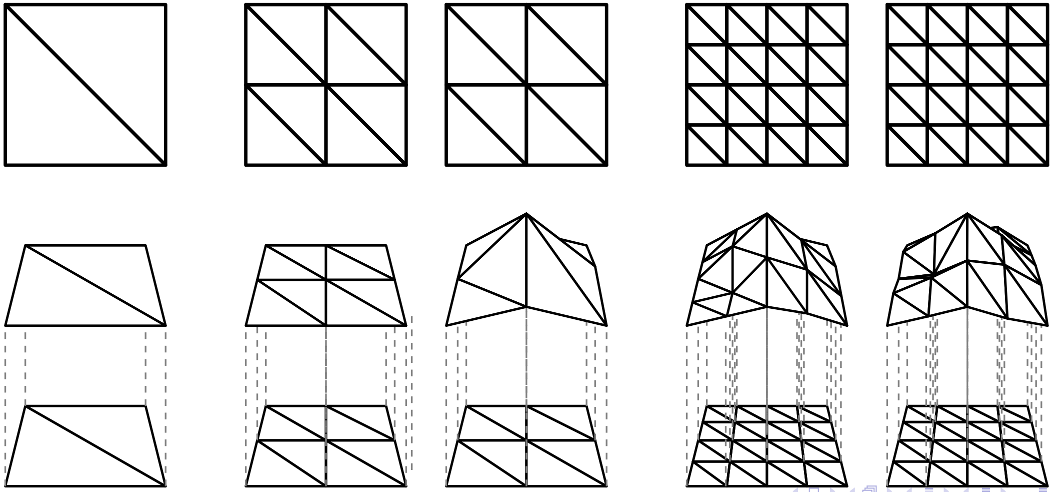
Image: David E. Joyce

Iterated Replacement / Koch Curve



Iterated Replacement / Mountains

Randomness in replacement



L-System Modeling

- Named after original developer: biologist Aristid Lindenmayer
- Use context-free grammars (CFG) to specify structural change over generations
- Often used to simulate a biological growth process
  - Plants
  - Seashells
  - ...
- Variations for other applications
  - Cities
  - Building architecture
  - Cloth weaving
  - ...

Context-Free Grammar

A CFG $G = (V, T, S, P)$ where

- V is a set of non-terminals
- T is a set of terminals
- $S \in V$ is the start symbol
- P is a set of productions (rules) of the form:
  - $A \rightarrow x$, where $A \in V, x \in (V \cup T)^*$

L-system

- L-sytem attaches geometric meaning to each symbol

L-system

- L-sytem attaches geometric meaning to each symbol
- Non-terminals
  - A, B, straight line segments
- Terminals
  - [ ], branch left 45°
  - ( ), branch right 45°

L-system

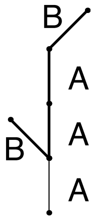
- L-sytem attaches geometric meaning to each symbol
- Non-terminals
  - A, B, straight line segments
- Terminals
  - [ ], branch left 45°
  - ( ), branch right 45°
- Rules
  - $A \rightarrow AA$
  - $B \rightarrow A[B]AA(B)$

L-system

- L-sytem attaches geometric meaning to each symbol
- Non-terminals
  - A, B, straight line segments
- Terminals
  - [ ], branch left 45°
  - ( ), branch right 45°
- Rules
  - $A \to AA$
  - $B \to A[B]AA(B)$
- Strings
  - Start: B

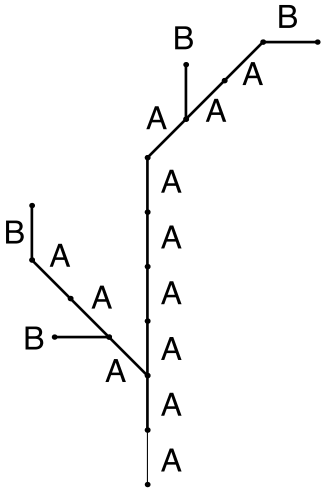
L-system

- L-sytem attaches geometric meaning to each symbol
- Non-terminals
  - A, B, straight line segments
- Terminals
  - [ ], branch left 45°
  - ( ), branch right 45°
- Rules
  - $A \to AA$
  - $B \to A[B]AA(B)$
- Strings
  - Start: B
  - $A[B]AA(B)$



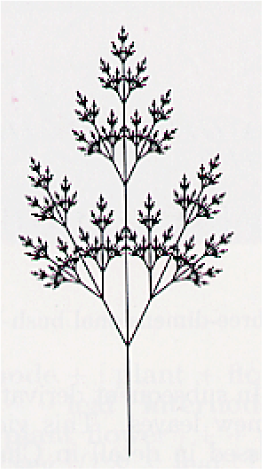
L-system

- L-sytem attaches geometric meaning to each symbol
- Non-terminals
  - A, B, straight line segments
- Terminals
  - [ ], branch left 45°
  - ( ), branch right 45°
- Rules
  - $A \rightarrow AA$
  - $B \rightarrow A[B]AA(B)$
- Strings
  - Start: B
  - $A[B]AA(B)$
  - $AA[A[B]AA(B)]AAAA(A[B]AA(B))$



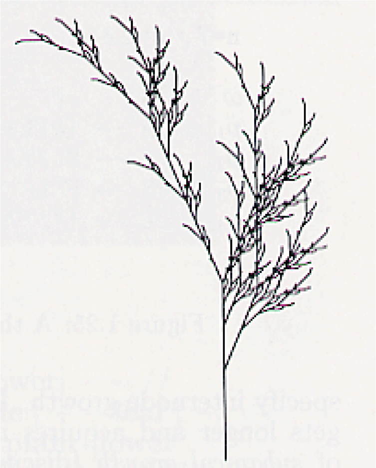
L-System Examples

- Symbols
  - [/] = push/pop
  - +/− = rotate left/right
  - $A - Z$ = straight segment
- Rules
  - 25.7°, 7 generations
  - $X \rightarrow F[+X][-X]FX$
  - $F \rightarrow FF$



L-System Examples

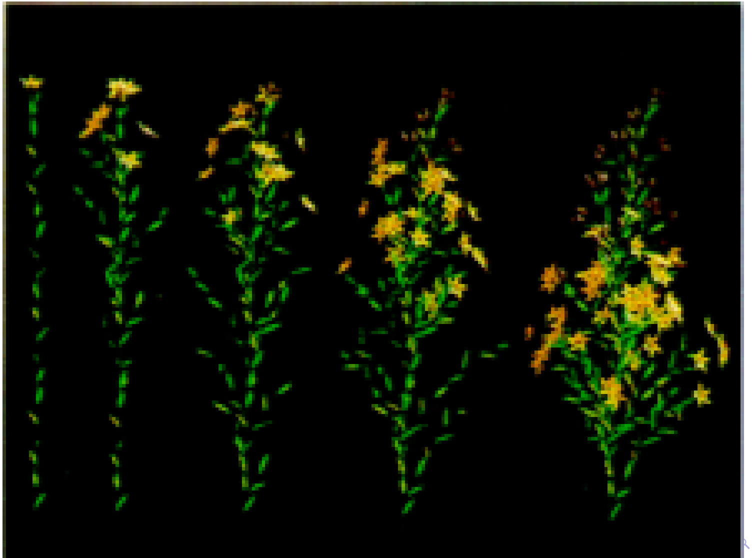
- Rules
  - 22.5°, 5 generations
  - $X \rightarrow F - [[X] + X] + F[+FX] - X$
  - $F \rightarrow FF$



L-System Examples

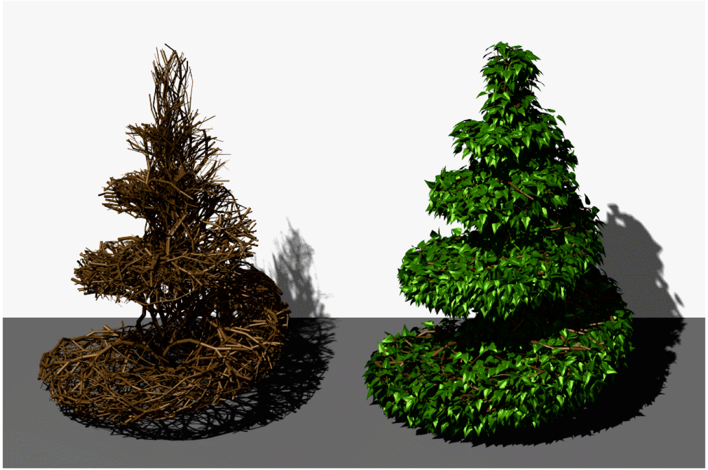
- Rules
  - 22.5°, 4 generations
  - $F \rightarrow FF-[F+F+F]+[+F-F-F]$

Additions



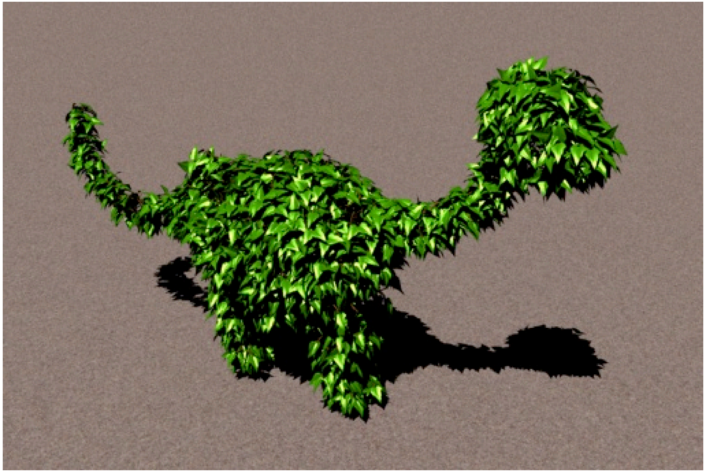
- 3D structure
- Randomness
- Leaves
- Flowers

Pruning



Prusinkiewicz, et al., SIGGRAPH 94

Pruning



Prusinkiewicz, et al., SIGGRAPH 94

Spectral Synthesis

- Alternative to explicitly defining structure
  - Define statistical properties

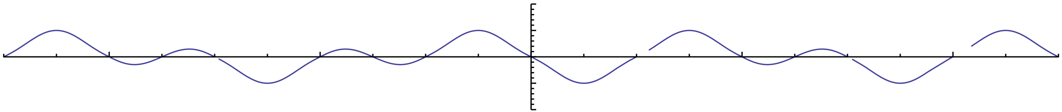
Spectral Synthesis

- Alternative to explicitly defining structure
  - Define statistical properties
- Spectral energy a function of frequency
  - Higher frequency, less energy
  - Characterizes roughness of surface
  - Natural phenomena tend to be $1/f$

Noise-Based Synthesis

Band-limited *Perlin noise* function

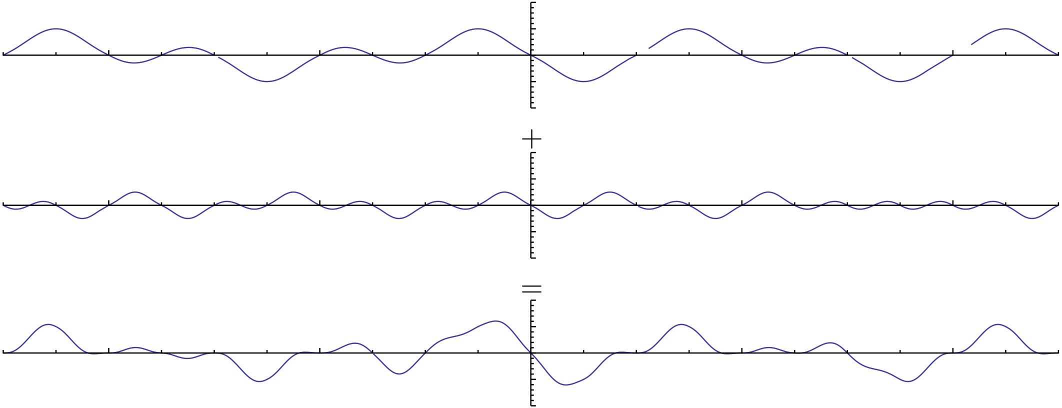
- Most energy between 1/2 and 1 cycle per unit
- Average value is 0
- Random, but repeatable
- 1D, 2D, 3D & 4D versions common



Spectral Synthesis

Sum noise *octaves*

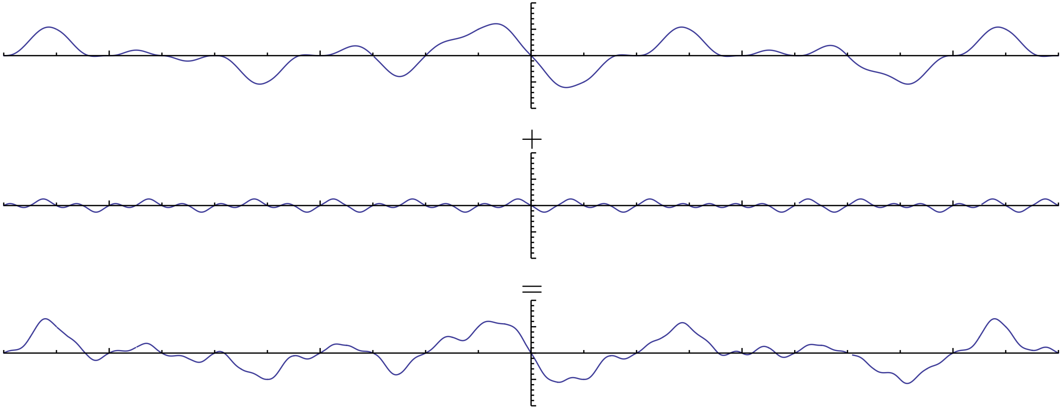
- $n(x) + \frac{1}{2}\ n(2\ x) + \frac{1}{4}\ n(4\ x) + \ldots$
- Stop adding "..." when frequency is too high to see
- Also called *fractional Brownian motion* or *fBm*



Spectral Synthesis

Sum noise *octaves*

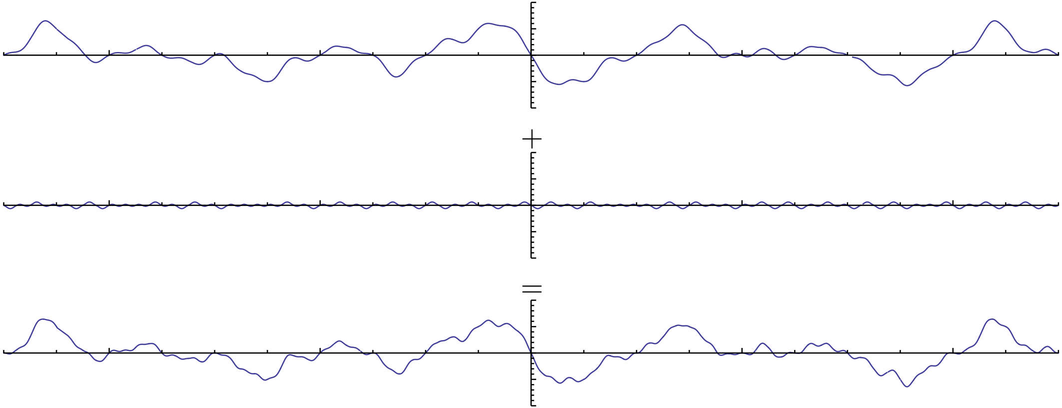
- $n(x) + \frac{1}{2}\ n(2\ x) + \frac{1}{4}\ n(4\ x) + \ldots$
- Stop adding "..." when frequency is too high to see
- Also called *fractional Brownian motion* or *fBm*



Spectral Synthesis

Sum noise *octaves*

- $n(x) + \frac{1}{2}\ n(2\ x) + \frac{1}{4}\ n(4\ x) + \ldots$
- Stop adding "…" when frequency is too high to see
- Also called *fractional Brownian motion* or *fBm*



Noise-based Landscape

Landscape height is a fBM function of x,y

- Plus whatever embellishments make it look good

Image: Ken Musgrave

Multifractal

- Change roughness across fractal
  - Scaling $(\frac{1}{2}, \frac{1}{4}, \ldots)$ becomes a function
- Here, scale is a function of altitude

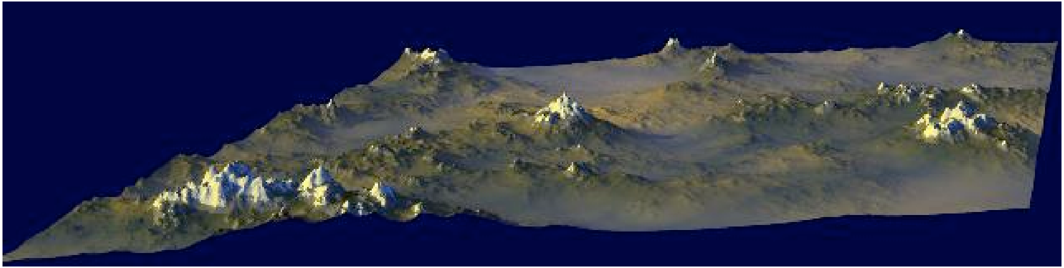
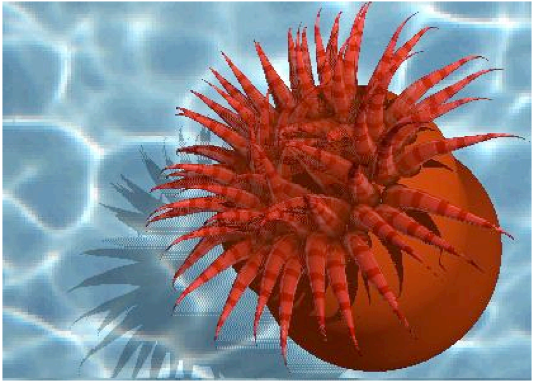


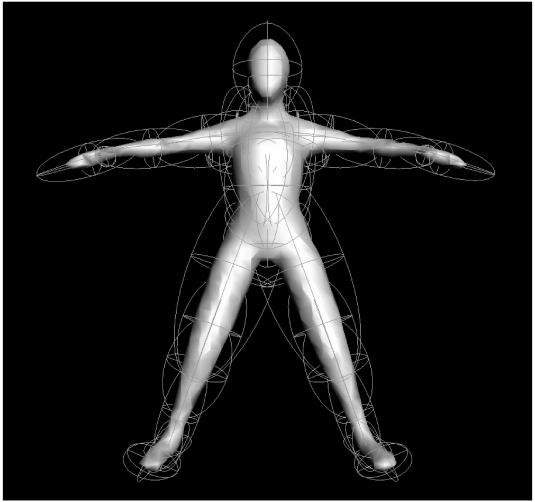
Image: Ken Musgrave

Implicit Functions or Blobby Modeling

- Model as sum of implicit functions
- Surface at threshold



Liang, et al., PG'01

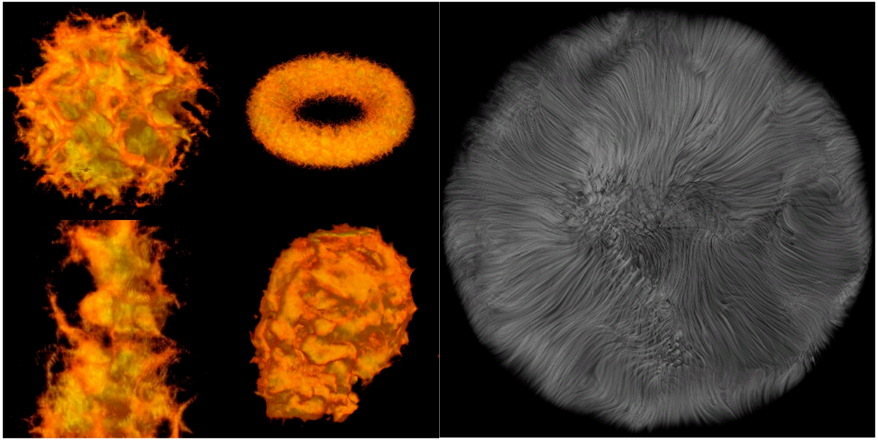


Hybrid Implicit & Polygonal

Bloomenthal, SIGGRAPH 85

Hypertexture

Add noise or turbulence to implicit functions



Perlin & Hoffert, SIGGRAPH 89

Simulations

Biological

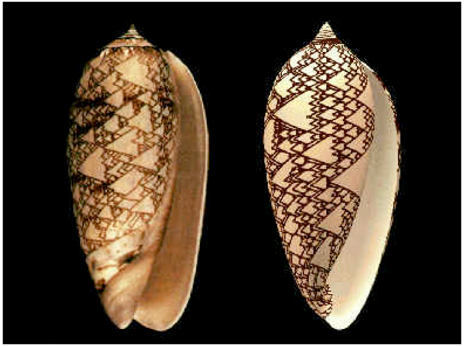
- Simulate growth, development

Physical

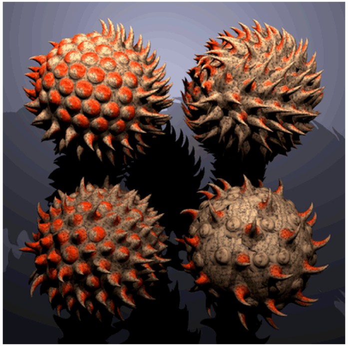
- Simulate formation or erosion

Compare to L-system or noise, where goal is just to "look right"

Biological Simulations



Fowler, et al., SIGGRAPH 92

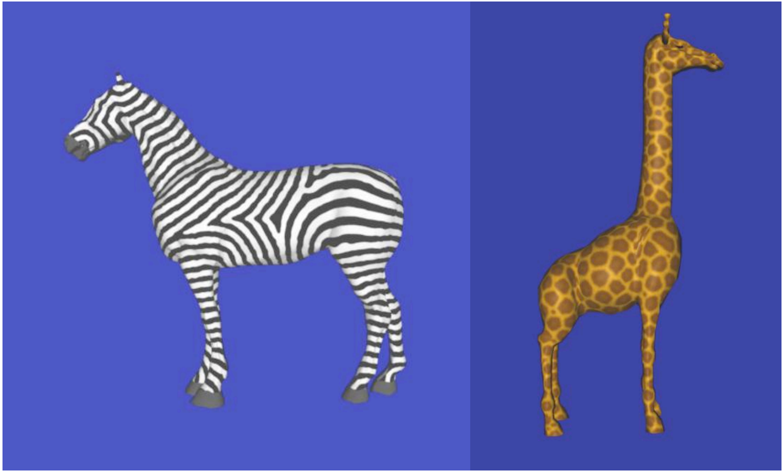


Fleischer, et al., SIGGRAPH 95

Biological Simulations

Fowler, et al., SIGGRAPH 92

Biological Simulations



Turk, SIGGRAPH 91

Physical Simulation

Erosion, Deposition

Kenji Nagashima, Visual Computer 1997

Scan from Objects

- General concept
  - Find points on surface
  - Connect into mesh
- Mechanical
- Triangulation
  - Laser
  - Structured Light
  - Multiple Cameras
- CAT scan / MRI

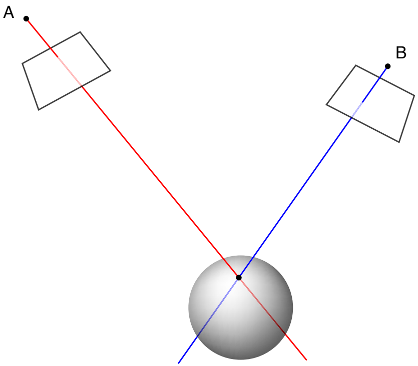
Mechanical

- Touch tip to surface
- Measure angles

Triangulation

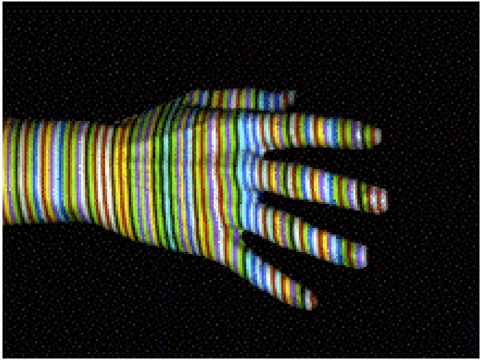
Point in space at intersection

- Ray from light A
- Ray through pixel B

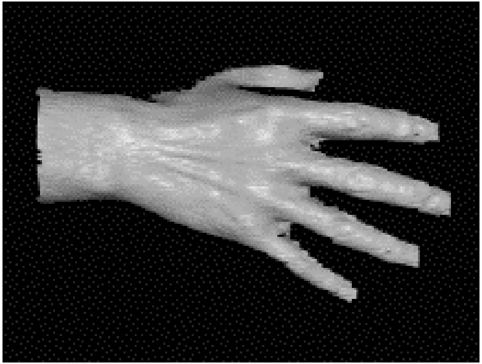


Structured Light

- Point in space at intersection of color edge from light source/projector and ray through camera pixel



projected pattern

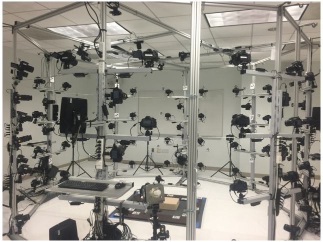


resulting model

Zhang, Curless and Seitz, 3DPVT 2002

Multiple Cameras

- Computer vision algorithm to find common features
- Triangulate to optimize cameras and points in 3D space
- Reconstruct dense mesh

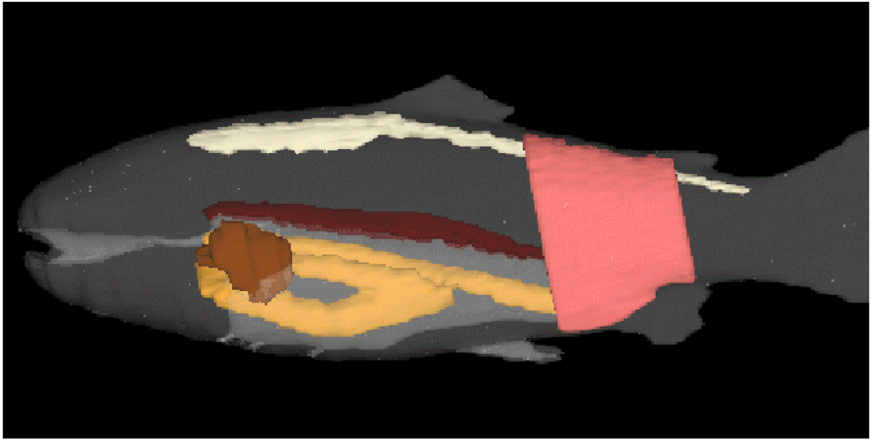


Visualization

- Data
  - measurements
  - simulation
  - information
- Present visually
  - Increase understanding
  - Recognize patterns

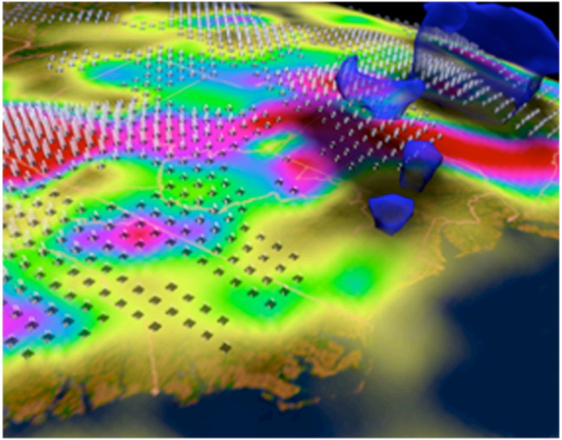
Visualization

Can be 3D Object



Visualization

Can be 3D, but showing non-visual aspects.



Visualization

Can be not traditionally geometric at all

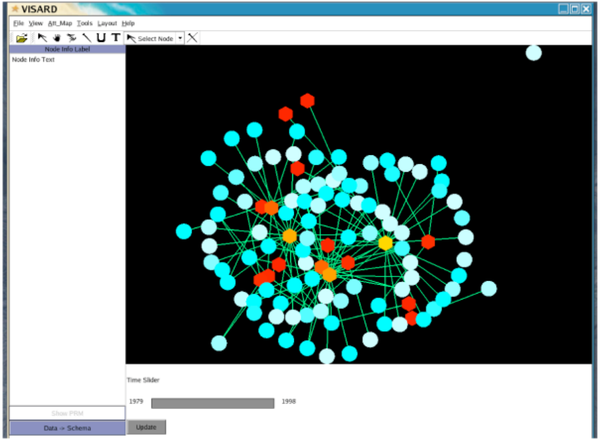


Image-based Rendering

- Construct new *novel* view using only image data
- No explicit geometric model
- Pixels in one or more cameras represent:
  - Image-Based Rendering: Color of point in space
  - Light Field Rendering: Color of light along one ray