Shape Representation

Basic problem
- We make pictures of things
- How do we describe those things?
    - Many of those things are shapes
    - Other things include motion, behavior…
    - Graphics is a form of simulation and modeling

Two general types of representations
- Surface representations
    - Concerned only with the surface of the shape, not the interior
    - Generally more efficient at representing surface
    - Polygons are one example
- Volume representations
    - Also concerned with interiors, surfaces can be extracted

Surface reps (Boundary reps or "B-reps")
- Parametric surfaces
    - Basic idea
        - Define curves and surfaces using parametric formula
        - Allow modelers to manipulate shape with "control points"
    - Structure
        - Use parametric representations
        - Q(t) = T M G, where
            - T = [t^3 t^2 t 1]
                - Segment runs from 0 to 1 in t
            - M = basis matrix (4x4)
            - G = geometry matrix (4x4)
        - Geometry matrix
            - Contains four geometry elements (pts, vectors)
            - These are the controls the user manipulates in modeling
                - curves interpolate or bend towards them
            - Every differently shaped curve will have a different geometry matrix
        - Basis matrix
            - Describes relative weighting of geometry as t varies
                - Described by the *blending function*
            - Different curve types (see below) have different bases
                - Basis is constant for a given curve type
    - Curve issues
        - Locality
            - If I change a geometry element, how much of the curve changes?
        - Joining
            - For complex curves/shapes, must join two or more curves together
            - Would like join to be continuous ("smooth")
                - G0, G1, C1, C2 continuity
        - Effect of transformations
            - Invariance:
                - Transform(Q(t)) = T M transform(G)?
            - Affine invariance
                - Under scales, translations, rotations
            - Perspective invariance
                - 2D curves from projected 3D control points
        - Convex hull property
            - Curve lies inside convex hull of geometry
                - Convex hull: smallest convex polygon (hedron) that contains control points
            - Useful for trivial rejects in clipping, intersection tests

- Useful for added intuition in modeling

- Types of curves
  - Bezier
    - Geometry
      - [P1 P2 P3 P4]
      - Four control points
        - Curve interpolates endpoints
        - Slope at ends equals slope of [P2-P1], [P4-P3]
    - Joining
      - G0: overlay endpoints
      - G1: give [P4-P3], [P5-P4] same slope
      - C1: same magnitude
      - C2: hard
    - Locality: none
    - Invariance: affine, not perspective
    - Convex hull property
  - B-Splines
    - Problem:
      - Joining with continuity is difficult
      - There is no C2 continuity
    - Solution:
      - Curves (sequences of segs) are defined by a sequence of n control points
      - Each sequential set of 4 points defines a segment
      - Each new segment shares 3 points with previous segment
      - *Knots* are the boundaries between each curve segment in t
    - Examples
      - Given a B-spline with n control points
        - There will be n-3 curve segments
        - There will be n-2 knots
      - Given 4 control points
        - one segment
        - two knots (begin and end)
      - Given 10 segments
        - 13 control points
        - 11 knots
    - Uniform, nonrational B-splines
      - Uniform: knots are spaced at equal intervals of parameter t
      - Nonrational: curve does not use ratios of two polynomial equations
      - Joining: C2 continuity is guaranteed
      - Locality: only the 4 segments containing a ctrl pt are affected
      - Invariance: affine, not perspective
      - Convex hull property for each segment
      - Other issues:
        - No control points are interpolated
        - Overlapping control points allow interpolation
          - But at the cost of continuity
    - Nonuniform, nonrational B-splines
      - Nonuniform: curve segs can be defined over non-equal intervals of t
      - As before but:
        - Can overlap knots

                - Allows interp points with better control of continuity
            - Can insert knots
                - Allows arbitrary control of locality
        - Nonuniform, rational B-splines (NURBS)
            - Rational: $x(t) = X(t)/W(t)$, $y(t) = Y(t)/W(t)$….
            - Like before but:
                - Invariance: affine and perspective
                - Can define conics (circles, parabolas…)
- Patches
    - A generalization from 1D to 2D
    - Most of previous discussion applies
    - General approach
        - Intuitively
            - Imagine sweeping a parametric curve Q(s) along a dimension t
            - The geometry vector, and thus the shape of Q(s), changes as a function of t
            - You defined an arbitrarily shaped patch!
        - In equations
            - $P(s,t) = S M Q(t) = S M G M^t T^t$
- Critique
    - Advantages
        - Accuracy: polys/lines are always only approximations of curves
        - Succinctness: need lots of polys to describe one smooth surface
        - Modeling: controlling shape
            - twiddling vertices is annoying, need higher level control
    - Problems
        - Rendering speed
            - Complex models may have as many patches as polygonal models
            - Rendering patches is slower
        - Portability
            - Patches are not the lowest common denominator
- Subdivision surfaces
    - New research
    - Give much better control of locality
    - A patch can be subdivided into smaller patches
    - In this way, a hierarchy of locality is formed

Volume representations
- Constructive Solid Geometry (CSG)
    - Domain: engineering and machining
    - Basic idea:
        - Shapes are described with set operations on primitive volumes
            - Union, difference, intersection
        - Such a description describes "how" to construct a shape
    - Structure
        - A binary tree
        - Leaves are primitives
        - Internal nodes perform set ops on children
    - Critique
        - Advantages
            - A continuous description
            - Describes the modeling process
            - Some powerful modeling functionality
        - Disadvantages

            - Joins are all discontinuous
            - Poor control of locality
            - Hard to render
    - Discrete volume reps (voxels)
        - Domain: largely medicine and science
        - Basic idea:
            - We surround the shape (or region) with a box
            - We sample the entire box with a regular grid
            - Each sample is called a "voxel" (volume pixel)
            - So there are many "shapes" in the volume
        - Structure
            - a 3D grid of points, with at least one value at each point
            - value represents "density" in most settings
        - Critique
            - Advantages
                - Captures interiors well
            - Disadvantages
                - Takes up a lot of space!
                    - 256^3 * 4 = 67 megs basic input
                - Fairly hard to render
                    - Ray tracing
                    - Surface extraction using Marching Cubes Alg
                    - This is changing with Mitsubishi card
    - Implicit functions
        - Domain: amorphous, merging shapes; bounding volumes
        - Basic idea:
            - We have a function in 3D space
            - The surface is all points w/ same value in that func: *isosurface*
                - f(x,y,z) = k
            - Example: sphere
                - x^2 + y^2 + z^2 = 1
            - We add complexity with collections of these
        - Structure
            - A collection of generator shapes
            - For each of these generators g
                - A distance function d which returns distance from g
                - A potential function f assigns a value to each distance d
            - A blending function B merges potentials – often simple addition
        - Critique
            - Strengths
                - Good for collision detection
                - Can represent conics like spheres, cylinders
                - Used in blobby modeling for molecules, etc
                - Can merge shapes by moving the generators
            - Problems:
                - Constraints needed. Semicircle?
                - Joining is a problem
                    - unwanted merges/seps
                - Also hard to render: voxelize, ray trace
        - New work addresses some control problems
- Other and newer approaches
    - BSP trees
        - These can also be used as modeling reps
        - Advantage in hierarchical formation
        - Enables set ops and fast collision detection
        - Main problem:

- avoiding polygonal explosion makes implementation very hard

Particle systems
- Domain: for moving clouds of things, e.g. clouds, fireworks, flocks
- Structure:
  - A set of points or particles
  - Each point has a largely random behavior and a lifespan
  - Particles are rendered as blurs onscreen or sprites

Fractal models
- Domain: clouds, mountains, sea
- Fractals are used to generate shape procedurally
- Usually these are converted into another representation

L-systems
- Domain: plant description
- Grammars used procedurally to describe plant development
- Eventually converted into another representation

Distance fields
- Can be viewed as adaptively sampled implicit function
- Because it is a sampled rep, can deviate from functional limitations

Point representations
- Domain: very large (1 billion vertex) models
- Like polygons, but no edges, just vertices
  - Vertices have associated color, orientation
- Basic problem: how to fill the gaps?
  - “Splatting” from volume rendering
  - Colors are “blurred” across local screen region
- Until recently, main problem was aliasing