Solid Modeling

- CSG (Constructive Solid Geometry) Representations: A set theoretic Boolean expression of primitive solid objects
  - Object is always valid (surface is closed, orientable and encloses a volume)

- B-Rep (Boundary Representation): Describes the oriented surface of a solid as a data structure composed of vertices, faces and edges
  - Easier to render
  - Easier for collision detection, finite element simulation

DUAL REPRESENTATION Modelers: Use both of them

Primitives

- Pre-selected from solid shapes and instantiated/transformed: blocks, polyhedra, spheres, cones, cylinder, tori (all can be represented using NURBS)

- Primitives is created by sweeping a contour along a space curves or surfaces (e.g. offsets are generated by sweeping a sphere)

- Primitives are algebraic half-spaces:

$$f(x;y;z)\ j\ _{,}\ f(x;y;z)\ \hat{o}\ 0g$$

where f(x,y,z) is an irreducible polynomial

CSG Representation

- Built from standard primitives, using regularized Boolean operations and transformations

- Each primitives is defined in a local coordinate system. The tree nodes correspond to transformations to place them in a global coordinate system

- Boolean operations or set-theoretic operations: union (U) , intersection (\ ) and difference (-)

Regularized Boolean Operations

Regularized union ([ã), regularized intersection (\ ã) and regularized subtraction (â ã)

Difference with normal set theoretic operations: the result is the closure of the operation on the interior of the two solids and used to eliminate dangling low-dimensional structures. To compute them:

- Compute A \ B in the set-theoretic sense.
- Compute the interior of A \ B (in the topological sense)
- Compute the closure the interior (i.e. all boundary points adjacent to some interior neighborhood)

The resulting solid is the *regularized intersection.* In practice, they are computed by computing A \ B and eliminate the lower-dimensional structures.

Point/Solid Classification

- Given a point (x,y,z), is it inside, outside or on the boundary of the solid
- Other queries include classifying a line, how a surface intersects a solid and a test of whether two solids intersect in a non-empty volume
- Reduce the query to classification of the primitives of the CSG tree, involving Downward propagation as well as upward propagation
- Issue: What happens when the point lies EXACTLY on the surface of the primitive

Tree Propagation

Downward Propagation

- If (x,y,z) arrives at a node specifying a Boolean operation, then it is passed unchanged to the two descendants of the node
- If (x,y,z) arrives at a node specifying a translation or rotation, the inverse translation or rotation is applied to (x,y,z) and the transformed point it sent to the node's descendant
- If (x,y,z) arrives at a leaf, then the point is classified w.r.t. that primitive and the classification (*in/on/out*) is returned to the parent of the leaf.

Upward Propagation

The messages contain point classification that must be combined at the Boolean operation nodes. No work is done at nodes representing transformations

Neighborhoods

Problem: How to classify points that lie on the surface of a primitive solid?

Solution: Use *neighborhoods*. The neighborhood of a point P w.r.t. solid **S**, is the intersection with **S** of an open ball of infinitesimal radius 2 centered at P.

P is inside **S**, iff the neighborhood is a full ball and P is outside **S,** iff the neighborhood is an empty ball. If P is on the surface of **S,** then the structure of neighborhood depends on the local topology of **S** at P.

Face: Ngbd. Is a halfspace

Edge: Ngbd. Is a wedge

Vertex: Ngbd. Is a cone

Refined Upward Propogation

Goal: Perform the respective Boolean operation on the neighborhood itself

- Account for the local geometry
- Devise suitable data structures to represent neighborhoods
- Transform the geometric data appropriately at the rigid motion nodes

In practice, involves dealing with lots of non-trivial and degenerate cases

Curve/Solid Classification

Applications: ray tracing a CSG model; boundary evaluation

- Send the line or curve description to the leaves
- Partition the curve into segments labeled inside, outside, or on the surface of the primitive
- Propagate the segments back upward, and merge them appropriately, by performing Boolean operations on them

Surface/Solid Classification

- Intersect the surface with each of the primitives from which the solid has been constructed
- Classify the resulting curves, thereby determining the bounding edges of those surface areas that are inside or outside the solid, or are on the solid's surface
- Combine the segments, appropriately oriented, constructing a boundary representation of the respective surface areas

Surface/solid classification can be used to devise a method for converting a CSG to B-rep model (based on generate-and-test paradigm)

Boundary Representations

Two parts:

- *Topological* description of the connectivity and orientation of vertices, edges and faces; in terms of incidences and adjacencies
- *Geometric* description for embedding these surface elements in space; includes vertex positions or surface equations

Manifold vs. Non-Manifold

- *Manifold* surface: Around every one of its points, there exists a neighborhood that is homeomorphic to a plane: I.e. the surface can be locally deformed into a plane without tearing it or identifying separate point with each other
- A manifold surface is orientable if we can distinguish two different sides, e.g. sphere and torus
- Non-orientable surfaces: Moebius strip, Klein bottle
- Closed orientable manifolds partition the space into the interior, the surface and the exterior
- A Boolean operation on two manifold objects may yield a non-manifold results

Common Approaches to Non-Manifold Structures

- Objects must be manifolds, so operations on solids with non-manifold results are not allowed and are considered an error
- Objects are topological manifolds, but their embedding in 3-space permits geometric coincidence of topologically separate structures (i.e. topological interpretation is given to non-manifold structures); serious robustness issues;
- Non-manifold objects are permitted, both as input and as output

Robust and Error Free Geometric Operations

- Geometric objects belong to a continuous domain, they are analyzed by algorithms doing discrete computations (e.g. floating point numbers)
- Imprecise representations; leads to contradictory information about the representation object
- Typical approach: relax the incidence tests, hard to control their implications

Floating Point Arithmetic

- Conversion errors: Can’t represent numbers exactly using binary arithmetic (e.g. 0.6)
- Roundoff errors: each arithmetic operations has roundoff error
- Catastrophic cancellation

Geometric Failures due to Floating Point Arithmetic

- Computation carried out with finite precision arithmetic
- Decision tests or questions of incidence: answered by different sequence of numerical computation
- Different sequences of computation are equivalent when exact arithmetic is used, but may result different answers using floating point arithmetic (e.g. incidence asymmetry checking whether intersections of lines correspond to the same)

Approaches for handling robustness

- Restructure the algorithm so that all interpretations of noisy numerical data and computations are logically independent
- Make interdependent logical decisions by respecting the symbolic data exactly, but can perturb the numerical data
- When making interdependent logical decisions, give priority to the numerical data

Specific Approaches

- Use exact arithmetic: simple for linear primitives; non-trivial for non-linear primitives
- Use symbolic information (hard for non-linear primitives)
- Use perturbation approaches (limited success)
- General area for non-linear primitives is an open area of research