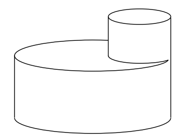
## Representing Solids

- ☆ Boundary representation
- ☆ Spatial decomposition
- ☆ Constructive solid geometry
- ☆ Boolean operations on solids

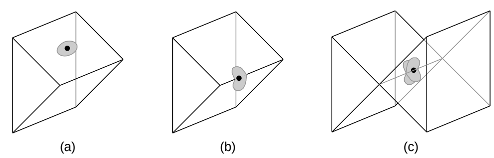
**Slide 1**

## Boundary Representation (b-rep)

- ☆ Describes an object in terms of its surface boundaries: *vertices, edges, faces*.
- ☆ Most common representation in computer graphics.
- ☆ Suitable for planar, polygonal boundaries.
- ☆ Defining faces for curved objects is tricky.



- ☆ Most b-reps support only solids whose boundaries are 2-*manifolds*.

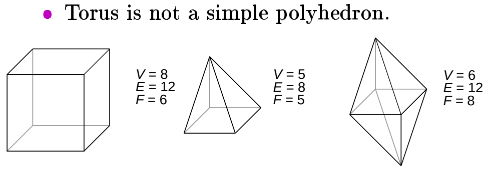



**Slide 2**



## Polyhedra

- ☆ Bounded by a set of polygons, each edge adjacent to even number of faces.
  - Adjacent to exactly two faces for 2-manifolds.
- ☆ *Simple* polyhedron: Can be deformed to a ball; no holes.
  - Examples: Cube, tetrahedron, prism, pyramid.
  - Torus is not a simple polyhedron.




- ☆ *Euler's formula* for simple polyhedra $V - E + F = 2$.
- ☆ Necessary but not sufficient condition for a simple polyhedron.

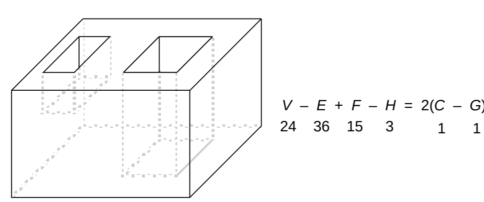
**Slide 3**

## Nonsimple Polyhedra

Euler formula generalizes to non-simple polyhedra with 2-manifold boundaries.

$$V - E + F - H = 2(C - G).$$

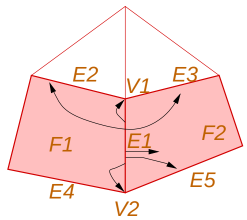
- ☆ $H$: # holes in 2D faces
- ☆ $G$: # holes passing through the polyhedra (tunnels); called *genus*
- ☆ $C$: # connected components




**Slide 4**

## Winged Edge Representation

- ☆ Used to represent simple polyhedra.
- ☆ Expedites certain operations.

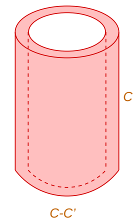



- ☆ Each edge $e$ stores
  - Two faces $f_1, f_2$ adjacent to $e$
  - Two endpoints $v_1, v_2$ of $e$
  - Two edges incident to $v_1$ immediately before and after $e$ in clockwise direction
  - Two edges incident to $v_2$ immediately before and after $e$ in clockwise direction
- ☆ Each vertex $v$ stores pointer to one of the edges incident to $v$.
- ☆ Each face $f$ stores pointer to one of the edges bounding $f$.

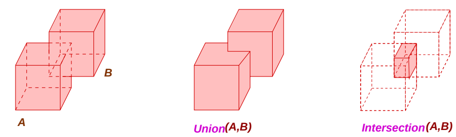
**Slide 5**

## Boolean Operations

Complex objects are defined as Boolean formula of simple objects




- ☆ Intersection
- ☆ Union
- ☆ Difference




The resulting object may have some dangling vertices, edges, and faces.

**Slide 6**



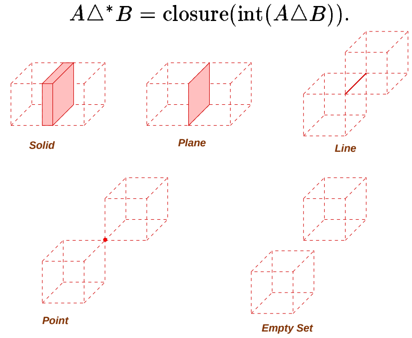
## Regularized Boolean Operations

**Interior** $(B)$**:** Points at distance $> 0$ from the complement of $B$.

**Boundary** $(B)$**:** Points at distance 0 from both $B$ and the complement of $B$.

**Closure** $(B)$**:** Interior $(B)$ + Boundary $(B)$.

$$\text{Regularize}(A) = \text{closure}(\text{int}(A))$$
$$A \triangle^* B = \text{closure}(\text{int}(A \triangle B)).$$



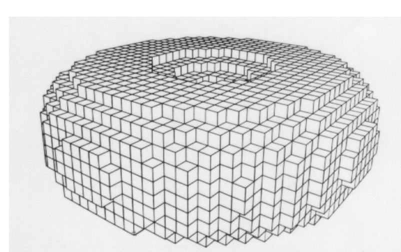

**Slide 7**

## Spatial Decomposition

- ☆ Divide the space into *primitive* cells.
- ☆ Represent all cells lying in the object.

**Spatial occupation enumeration**

- ☆ Divide the space into identical cells arranged in a fixed regular grid structures.
- ☆ 3D Analog of 2D images.
- ☆ Cells are often cubes and are called *voxels*.
- ☆ Popular representation in volume rendering and CAT.
- ☆ High storage requirement.



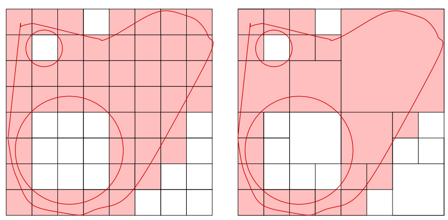
**Slide 8**

## Oct Trees

- ☆ Hierarchical representation.
- ☆ Requires much less space.
- ☆ Extension of 2D *quad tree*.

**Quad tree:**

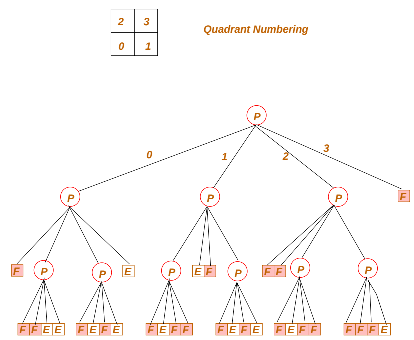
- ☆ Recursively subdivide the plane into four squares by bisecting it in both directions.
- ☆ A square is *full, empty, partially full*.
- ☆ A partially full square is further subdivided.
- ☆ Parttioning continues until a cutoff threshold is reached.



**Slide 9**

## Quad Trees

- ☆ Can be reprsented as a 4-way tree.
- ☆ Each node $v$ represents a square $Q_v$
  - If $Q_v \subseteq P$, $v$ is *black*.
  - If $Q_v \cap P = \emptyset$, $v$ is *white*.
  - Otherwise $v$ is *gray*.
  - Gray nodes are further subdivided.

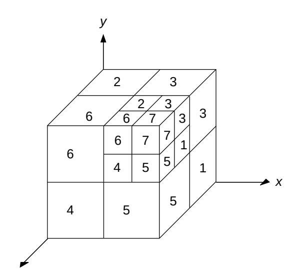



**Slide 10**



## Oct Trees

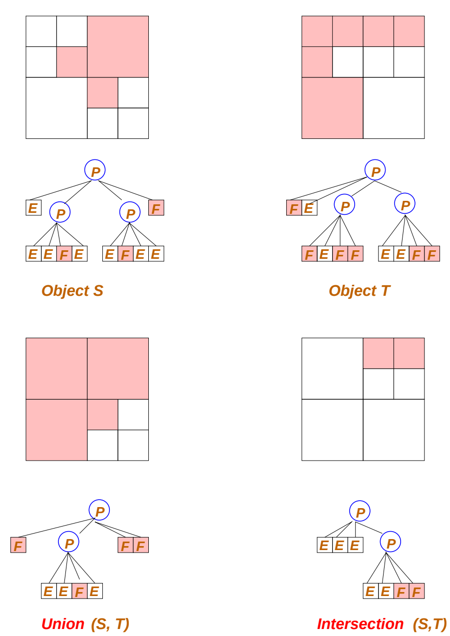
- ☆ Oct tree is a similar to quadtrees.
- ☆ Each cube is divided into eight octants.
- ☆ Useful for many operations, e.g., collision detection, ray tracing.




- ☆ Space requirement is still large. item Sensitive to the position of the object.
- ☆ Only approximate representation for nonorthogonal objects.

**Slide 11**

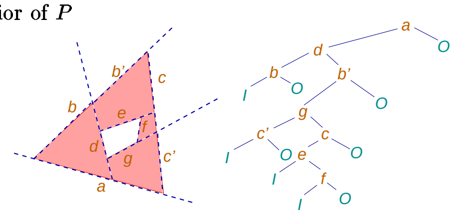
## Boolean Operations on Quad Trees




**Slide 12**

## Binary Space Partiton (BSP) Trees

$P$: Polyhedron; Normal of each face point to exterior of $P$

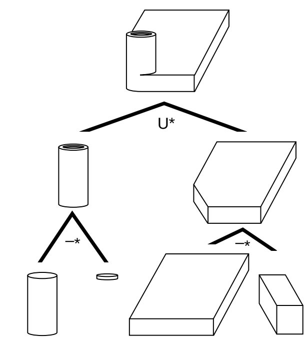



- ☆ Each interior node $v$ is associated with a plane $\pi_v$ (containing a face of $P$) and convex polytope $Q_v$.
  - $\pi_v^+$: outside halfspace bounded by $\pi_v$.
  - $\pi_v^-$: inside halfspace bounded by $\pi_v$.
- ☆ The left child $w$ of $v$ is associated with $Q_v \cap \pi^-$.
- ☆ If $Q_w$ is monochromatic, $w$ is a leaf.
- ☆ The right child $x$ of $v$ is associated with $Q_v \cap \pi^+$.
- ☆ If $Q_z$ is monochromatic, $w$ is a leaf.

**Slide 13**

## Constructive Solid Geometry (CSG)

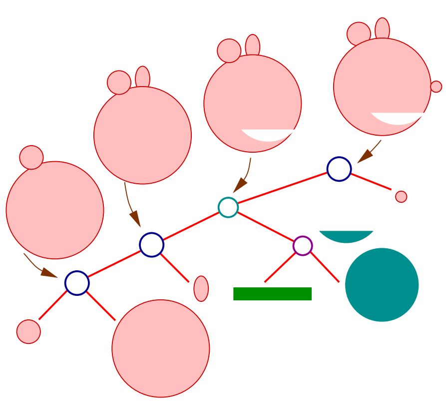
- ☆ Simple primitives are combined using regularized Boolean operations
- ☆ Object is stored as a tree with *operators* at interior nodes
- ☆ Edges of the tree are ordered
- ☆ Spatial decomposition a special case of CSG




**Slide 14**



## Constructive Solid Geometry (CSG)



**Slide 15**

## Particle Systems

- ☆ A collection of points is used to model an object
- ☆ Particles follow physical laws
- ☆ Examples
  - Smoke, fire, fog
  - Deformable objects: clothes, elastic objects, rope
  - Wave action, storm
  - Scientific visualization

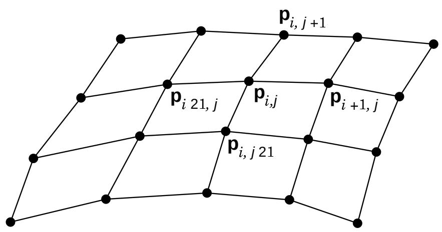
### Newtonian Particles

- ☆ Obey Newton's second law of motion
- ☆ $\mathbf{f} = m\mathbf{a}$
- ☆ $\mathbf{p}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \quad \mathbf{v}_i = \mathbf{p}'_i = \begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix}$

**Slide 16**

## Newtonian Particles

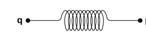
- *Independent particles.* Position of a particle does not depend on others, e.g., particles under gravity
  Each time step requires $\Theta(n)$ time.
- *Interactive particles.* Position of a particle depends on the others, e.g., stars
  Each time step requires $\Theta(n^2)$ time.
- In practice the dynamics of a particle depnds on its neighbors, e.g., clothing simulation, ropes




**Slide 17**

## Spring Forces

- Adjacent particles are connected by a spring
- $\mathbf{p}, \mathbf{q}$: Two adjacent particles; $\mathbf{d} = \mathbf{p} - \mathbf{q}$
- $\mathbf{f}$: Force on $p$ from $q$
- $s$: Stationary length of spring
- $k_s$: Spring constant




**Hook's Law:**

$\mathbf{f} = -k_s(|\mathbf{d}| - s)\dfrac{\mathbf{d}}{|\mathbf{d}|}$

- Include damping term in Hook's law
- Depends on the vecloity of $p$ and $q$
- $\mathbf{d}' = \mathbf{p}' - \mathbf{q}'$

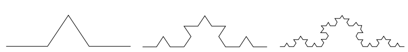
$$\mathbf{f} = -\left(k_s(|\mathbf{d}| - s) + k_d \frac{\mathbf{d}' \cdot \mathbf{d}}{|\mathbf{d}|}\right) \frac{\mathbf{d}}{|\mathbf{d}|}$$

**Slide 18**

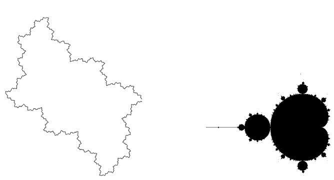


## Fractal Models

- Self similar objects.
- Repeat the same construction recursively.




- *Fractal dimension*: Scaling factor at each step.
- Examples: Julia-Fatou set and Mandelbrot sets.

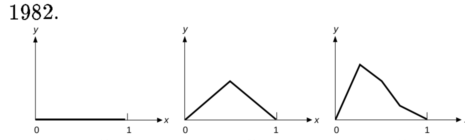


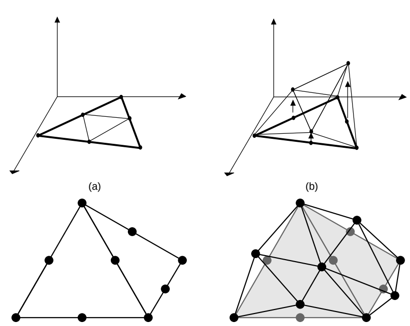
- Fractals are used to model mountains, rocks, trees, coastlines, ditches, etc.

**Slide 19**

## Fractals: Modeling Peaks

First used by Fournier, Fussell, and Carpenter in 1982.




**Slide 20**

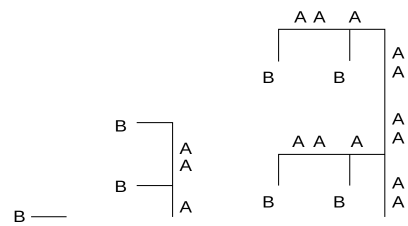
## Grammar Models

- ☆ Generalization of fractals
- ☆ A typical grammar with alphabet $\{A, B, [,], (,)\}$.
  - $A$ is a vertical segment; $(,)$: right branch; $[,]$: left branch.
  - $A \rightarrow AA$
  - $B \rightarrow A[B]AA[B]$
  - $B \rightarrow A[B]AA(B)$

**Slide 21**

## Grammar Models: Examples

Only left branch.
0. B
1. A[B]AA[B]
2. AA[A[B]AA[B]]AAAA[A[B]AA[B]]

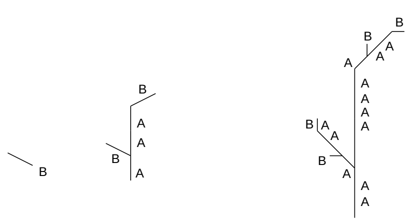


**Slide 22**



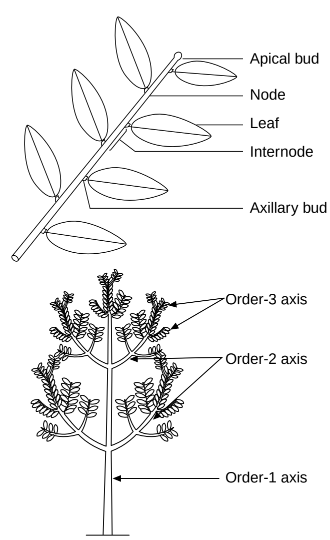
## Grammar Models: Examples

Both left and right branches
0. B,
1. A[B]AA(B),
2. AA[A[B]AA(B)]AAAA(A[B]AA(B))



**Slide 23**

## Grammar Models: Examples




**Slide 24**