**Polygon Meshes**

- A **polygon mesh** is a collection of polygons, along with a normal vector associated to each polygon vertex:
  - An edge connects two vertices
  - A polygon is a closed sequence of edges
  - An edge can be shared by two adjacent polygons
  - A vertex is shared by at least two edges
  - A normal vector pointing "outside" is associated with each polygon vertex

**Properties:**

- **Connectedness:** A mesh is connected if there is a path of edges between any two vertices
- **Simplicity:** A mesh is simple if the mesh has no holes in it
- **Planarity:** A mesh is planar if every face of it is a planar polygon
- **Convexity:** The mesh is convex if the line connecting any two points in the mesh belongs to the mesh

**Representing Polygon Meshes**

- **Explicit (vertex list)**
  \[ P_1=(V_1,V_2,V_4) \]
  \[ P_2=(V_2,V_3,V_4) \]
- **Pointers to a vertex list**
  \[ V=(V_1,V_2,V_3,V_4) \]
  \[ P_1=(1,2,4) \]
  \[ P_2=(4,2,3) \]
- **Pointers to an edge list**
  \[ V=(V_1,V_2,V_3,V_4) \]
  \[ E_1=(1,2,P_1) \]
  \[ E_2=(2,3,P_2) \]
  \[ E_3=(3,4,P_2) \]
  \[ E_4=(4,2,P_2) \]
  \[ E_5=(4,1,P_1) \]
  \[ P_1=(E_1,E_3,E_5) \]
  \[ P_2=(E_2,E_3,E_4) \]
Plane Equations

- Given \( V_1, V_2, V_3 \), the plane normal (or the coefficients \( A, B \) and \( C \)) can be computed with the cross product:
  \[
  S = (V_3 - V_1) \times (V_2 - V_1)
  \]
  \( N = S / |S| \)
- \( D \) can be found by plugging any point of the plane in the equation \( Ax + By + Cz + D = 0 \)
- The plane equation is not unique

Volume Representation

- A collection of techniques to represent and define volumetric objects
- Desired properties:
  - Rich representation
  - Unambiguous
  - Unique representation
  - Accurate
  - Compact
  - Efficient
  - Possible to test validity

Primitive Instancing

- Define a family of parameterized objects
- The definition is procedural (a routine defines it)
- Not general, must be individually defined for each family of objects

Example: Wheels/Gears

<table>
<thead>
<tr>
<th>Diameter</th>
<th>Teeth</th>
<th>Hole</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>5</td>
</tr>
</tbody>
</table>
Sweep Volume

- **Sweep Volume**: sweeping a 2D area along a trajectory creates a new 3D object
- **Translational Sweep**: 2D area swept along a linear trajectory normal to the 2D plane
- **Tapered Sweep**: scale area while sweeping
- **Slanted Sweep**: trajectory is not normal to the 2D plane
- **Rotational Sweep**: 2D area is rotated about an axis
- **General Sweep**: object swept along any trajectory and transformed along the sweep

Spatial Occupancy Enumeration

- Space is described as a regular array of cells (usually cubes). Each cell is called a **Voxel**
- A 3D object is represented as a list of filled voxels

**Pros:**
- Easy to verify if a point (a voxel) is inside or outside an object
- Boolean operations are easy to apply

**Cons:**
- Memory costs are high
- Resolution is limited to size and shape of voxel

Octrees

- **Octree**: a 3D generalization of a Quadtree
- Each node in an Octree has eight children rather than four
- Describes a recursive partitioning of a volume into cells that are completely full or empty
**Binary Operations on Quad/Octrees**

Notation:
- **P**: Internal Node
- **E**: Empty Leaf Node
- **F**: Full Leaf Node

Union:
- \( P \cup E = E \)
- \( P \cup F = F \)
- \( E \cup F = F \)

Intersection:
- \( P \cap E = E \)
- \( P \cap F = E \)
- \( E \cap F = E \)

Complement:
- \( P^C = E \)
- \( E^C = E \)
- \( F^C = F \)

Difference:
- \( P - E = E \)
- \( P - F = F \)
- \( E - F = E \)

**Binary Space Partition Trees - BSP**

- Each internal node represents a plane in 3D space.
- Each node has 2 children pointers one for each side of the plane.
- A leaf node represents a homogeneous portion of space - either “in” or “out”.
- Easy to determine if a point is inside or outside an object (recurse down the BSP tree).

**Constructive Solid Geometry**

- Combine simple primitives using Boolean operations and represent as a binary tree.
- To generate the object the tree is processed in a depth-first pass.
- **Cons**: representation is not unique.

**Boundary Representations**

- A closed 2D surface defines a 3D object.
- At each point on the boundary there is an “in” and an “out” side.
- Boundary representations can be defined in two ways:
  - Primitive based. A collection of primitives forming the boundary (polygons, for example).
  - Freeform based (splines, parametric surfaces, implicit forms).
**Boundary Representations**

- A polyhedron is a solid bounded by a set of polygons.
- A polyhedron is constructed from:
  - Vertices $V$
  - Edges $E$
  - Faces $F$
- Each edge must connect two vertices and be shared by exactly two faces.
- At least three edges must meet at each vertex.

**Boundary Representations**

- A **simple polyhedron** is one that can be deformed into a sphere (contains no holes).
- A simple polyhedron must satisfy Euler's formula:
  $$V - E + F = 2$$

**Boundary Representations**

- Euler's formula can be generalized to a polyhedron with holes and multiple components.
  $$V - E + F - H = 2(C - G)$$
  Where:
  - $H$ is the number of holes in the faces.
  - $C$ is the number of separate components.
  - $G$ is the number of pass-through holes (genus if $C=1$).
  - $V$, $E$, and $F$ are respectively vertices, edges, and faces.

$$V - E + F - H = 2(C - G)$$
$$24 - 36 + 15 - 3 = 2(1 - 1)$$