Object Representations

- Types of objects: geometrical shapes, trees, terrains, clouds, rocks, glass, hair, furniture, human body, etc.

- Not possible to have a single representation for all:
  - Polygon surfaces
  - Spline surfaces
  - Procedural methods
  - Physical models
  - Solid object models
  - .....
Polygon Surfaces

- Set of adjacent polygons representing the object exteriors.
- All operations linear, so fast.
- Non-polyhedron shapes can be approximated by polygon meshes.
- Smoothness is provided either by increasing the number of polygons or interpolated shading methods.

Levels of detail | Interpolated shading
Data Structures

- Data structures for representing polygon surfaces:
  - Efficiency
    - Intersection calculations
    - Normal calculations
    - Access to adjacent polygons
  - Flexibility
    - Interactive systems
    - Adding, changing, removing vertices, polygons
  - Integrity
### Polygon Tables

1. **Vertices**
   - $V_1: (x_1, y_1, z_1)
   - $V_2: (x_2, y_2, z_2)
   - $V_3: (x_3, y_3, z_3)
   - $V_4: (x_4, y_4, z_4)
   - $V_5: (x_5, y_5, z_5)
   - $V_6: (x_6, y_6, z_6)
   - $V_7: (x_7, y_7, z_7)
   - $V_8: (x_8, y_8, z_8)

2. **Edges**
   - $E_1: V_1, V_2$
   - $E_2: V_2, V_3$
   - $E_3: V_2, V_5$
   - $E_4: V_4, V_5$
   - $E_5: V_3, V_4$
   - $E_6: V_4, V_7$
   - $E_7: V_7, V_8$
   - $E_8: V_6, V_8$
   - $E_9: V_1, V_6$
   - $E_{10}: V_5, V_6$
   - $E_{11}: V_5, V_7$

3. **Polygons**
   - $S_1: E_1, E_3, E_{10}, E_9$
   - $S_2: E_2, E_5, E_4, E_3$
   - $S_3: E_{10}, E_{11}, E_7, E_8$
   - $S_4: E_4, E_6, E_{11}$

4. **Forward pointers:** i.e. to access adjacent surfaces edges

   - $V_1: E_1, E_9$
   - $V_2: E_1, E_2, E_3$
   - $V_3: E_2, E_5$
   - $V_4: E_4, E_5, E_6$
   - $V_5: E_3, E_4, E_{10}, E_{11}$
   - $V_6: E_8, E_9, E_{10}$
   - $V_7: E_6, E_7, E_{11}$
   - $V_8: E_7, E_8$
   - $E_1: S_1$
   - $E_2: S_2$
   - $E_3: S_1, S_2$
   - $E_4: S_2, S_4$
   - $E_5: S_2$
   - $E_6: S_4$
   - $E_7: S_3$
   - $E_8: S_3$
   - $E_9: S_1$
   - $E_{10}: S_1, S_3$
   - $E_{11}: S_3, S_4$
• Additional geometric properties:
  - Slope of edges
  - Normals
  - Extends (bounding box)

• Integrity checks
  - $\forall V, \exists E_a, E_b$ such that $V \in E_a, V \in E_b$
  - $\forall E, \exists S$ such that $E \in S$
  - $\forall S, S$ is closed
  - $\forall S_1, \exists S_2$ such that $S_1 \cap S_2 \neq \emptyset$
  - $S_k$ is listed in $E_m \iff E_m$ is listed in $S_k$
Polygon Meshes

- Triangle strips: 
  123, 234, 345, ..., 10 11 12
  1 2 3 4 5 6 7 8 9 10 11 12

- Quadrilateral meshes: 
  \( n \times m \) array of vertices
Plane Equations

- **Equation of a polygon surface:**
  \[ Ax + By + Cz + D = 0 \]
  Linear set of equations:
  \[(A/D)x_k + (B/D)y_k + (C/D)z_k = -1, \quad k = 1, 2, 3\]

\[
A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2) \\
B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2) \\
C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \\
D = -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)
\]

- **Surface Normal:**
  \[ N = (A, B, C) \]
  extracting normal from vertices:
  \[ N = (V_2 - V_1) \times (V_3 - V_1) \]
• Find plane equation from normal

\[
(A, B, C) = N \\
N \cdot (x, y, z) + D = 0
\]

\(P\) is a point in the surface (i.e. a vertex)

\(D = -N \cdot P\)

• Inside outside tests of the surface:

\[Ax + By + Cz + D < 0, \quad \text{point is inside the surface}\]

\[Ax + By + Cz + D > 0, \quad \text{point is outside the surface}\]
Spline Representations

- **Spline curve**: Curve consisting of continuous curve segments approximated or interpolated on polygon control points.

- **Spline surface**: A set of two spline curves matched on a smooth surface.

- **Interpolated**: Curve passes through control points.

- **Approximated**: Guided by control points but not necessarily passes through them.
• Convex hull of a spline curve: smallest polygon including all control points.

• Characteristic polygon, control path: vertices along the control points in the same order.
• Parametric equations:
  \[ x = x(u), \quad y = y(u), \quad z = z(u), \quad u_1 \leq u \leq u_2 \]

• Parametric continuity: Continuity properties of curve segments.
  
  – Zero order: Curves intersects at one end-point: \( C^0 \)
  
  – First order: \( C^0 \) and curves has same tangent at intersection: \( C^1 \)
  
  – Second order: \( C^0, C^1 \) and curves has same second order derivative: \( C^2 \)
• Geometric continuity: Similar to parametric continuity but only the direction of derivatives are significant. For example derivative (1,2) and (3,6) are considered equal.

• $G^0, G^1, G^2$: zero order, first order, and second order geometric continuity.
Spline Equations

- Cubic curve equations:

\[
x(u) = a_{x} u^3 + b_{x} u^2 + c_{x} u + d_{x} \\
y(u) = a_{y} u^3 + b_{y} u^2 + c_{y} u + d_{y} \quad 0 \leq u \leq 1 \\
z(u) = a_{z} u^3 + b_{z} u^2 + c_{z} u + d_{z}
\]

\[
x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} = U \cdot C
\]

- General form: \( x(u) = U \cdot M_s \cdot M_g \)

\( M_g \): geometric constraints (control points)

\( M_s \): spline transformation (blending functions)
Natural Cubic Splines

• Interpolation of $n+1$ control points. $n$ curve segments. $4n$ coefficients to determine

• Second order continuity. 4 equation for each of $n-1$ common points:

$$x_k(1) = p_k, \quad x_{k+1}(0) = p_k, \quad x_k'(1) = x_{k+1}'(0) \quad x_k''(1) = x_{k+1}''(0)$$

4$n$ equations required, $4n-4$ so far.

• Starting point condition, end point condition.

$$x_1(0) = p_0, \quad x_n(1) = p_n$$

• Assume second derivative 0 at end-points or add phantom control points $p_{-1}$, $p_{n+1}$.

$$x_1''(0) = 0, \quad x_n''(1) = 0$$
• Write $4n$ equations for $4n$ unknown coefficients and solve.

• Changes are not local. A control point effects all equations.

• Expensive. Solve $4n$ system of equations for changes.
Hermite Interpolation

- End point constraints for each segment is given as:

\[ P(0) = p_k, \quad P(1) = p_{k+1}, \quad P'(0) = Dp_k, \quad P'(1) = Dp_{k+1} \]

- Control point positions and first derivatives are given as constraints for each end-point.

\[
P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a & b & c & d \end{bmatrix} \quad P'(u) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} \begin{bmatrix} a & b & c & d \end{bmatrix}
\]

\[
\begin{bmatrix} p_k & p_{k+1} & Dp_k & Dp_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}
\]
Segments are local. First order continuity

Slopes at control points are required.

Cardinal splines and Kochanek-Bartel splines approximate slopes from neighbor control points.

\[
\begin{bmatrix}
  a \\
  b \\
  c \\
  d \\
\end{bmatrix} =
\begin{bmatrix}
  0 & 0 & 0 & 1 \\
  1 & 1 & 1 & 1 \\
  0 & 0 & 1 & 0 \\
  3 & 2 & 1 & 0 \\
\end{bmatrix}^{-1}
\begin{bmatrix}
  p_k \\
  Dp_k \\
  \vdots \\
  \vdots \\
\end{bmatrix} =
\begin{bmatrix}
  2 & -2 & 1 & 1 \\
  -3 & 3 & -2 & -1 \\
  0 & 0 & 1 & 0 \\
  1 & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
  p_k \\
  p_{k+1} \\
  Dp_k \\
  Dp_{k+1} \\
\end{bmatrix} = M_H
\begin{bmatrix}
  p_k \\
  p_{k+1} \\
  Dp_k \\
  Dp_{k+1} \\
\end{bmatrix}
\]

\[P(u) = p_k(2u^3 - 3u^2 + 1) + p_{k+1}(-2u^3 + 3u^2) + Dp_k(u^3 - 2u^2 + u) + Dp_{k+1}(u^3 - u^2)\]
Bézier Curves

- A Bézier curve approximates any number of control points for a curve section (degree of the Bézier curve)
Polynomial degree of a Bézier curve is one less than the number of control points.

3 points : parabola
4 points : cubic curve
5 points : fourth order curve

\[
P(u) = \sum_{k=0}^{n} p_k \text{BEZ}_{k,n}(u), \quad 0 \leq u \leq 1
\]

\[
\text{BEZ}_{k,n}(u) = \binom{n}{k} u^k (1-u)^{n-k}
\]

• Polynomial degree of a Bézier curve is one less than the number of control points.
  3 points : parabola
  4 points : cubic curve
  5 points : fourth order curve
• Properties of Bézier curves:
  - Passes through start and end points
    \[ P(0) = p_0, \quad P(1) = p_n \]
  - First derivatives at start and end are:
    \[ P'(0) = -n \, p_0 + n \, p_1, \quad P'(1) = -n \, p_{n-1} + n \, p_n \]
  - Lies in the convex hull
• Joining Bézier curves:
  - Start and end points are same ($C^0$)
  - Choose adjacent points to start and end in the same line ($C^1$)
    \[ p_{a,n} = p_{b,0}, \quad p_{b,1} = p_{a,n} + (p_{a,n} - p_{a,n-1}) \]
  - For second order ($C^2$) choose the next point in terms of the previous 2 of the other segment.
    \[ p_{b,2} = p_{a,n-2} + 4(p_{a,n} - p_{a,n-1}) \]
Cubic Bézier Curves

- Most graphics packages provide Cubic Béziers.

\[
\begin{align*}
\text{BEZ}_{0,3}(u) &= (1-u)^3 \\
\text{BEZ}_{1,3}(u) &= 3u(1-u)^2 \\
\text{BEZ}_{2,3}(u) &= 3u^2(1-u) \\
\text{BEZ}_{3,3}(u) &= u^3 \\
\end{align*}
\]

\[
P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot M_{\text{Bez}} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}
\]

\[
M_{\text{Bez}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}
\]
Bézier Surfaces

- Cartesian product of Bézier blending functions:

\[ P(u, v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} \text{BEZ}_{j,m}(v) \text{BEZ}_{k,n}(u) \quad 0 \leq u, v \leq 1 \]
Bézier Patches

- A common form of approximating larger surfaces by tiling with cubic Bézier patches. $m=n=3$
- 4 by 4 = 16 control points.
• Matrix form

\[ P(u, v) = U \cdot M_{Bez} \cdot P \cdot M_{Bez}^T \cdot T^T = \]

\[
\begin{bmatrix}
    u^3 & u^2 & u & 1
\end{bmatrix}
\begin{bmatrix}
    -1 & 3 & -3 & 1 \\
    3 & -6 & 3 & 0 \\
    -3 & 3 & 0 & 0 \\
    1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
    p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} \\
    p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} \\
    p_{2,0} & p_{2,1} & p_{2,2} & p_{2,3} \\
    p_{3,0} & p_{3,1} & p_{3,2} & p_{3,3}
\end{bmatrix}
\begin{bmatrix}
    -1 & 3 & -3 & 1 \\
    3 & -6 & 3 & 0 \\
    -3 & 3 & 0 & 0 \\
    1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
    v^3 \\
    v^2 \\
    v \\
    1
\end{bmatrix}
\]

• Joining patches:
similar to curves. \( C^0, C^1 \) and \( C^2 \) can be established by choosing control points accordingly.
Displaying Curves and Surfaces

- Horner's rule: less number of operations for calculating polynomials.

\[ x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \]
\[ x(u) = ((a_x u + b_x) u + c_x) u + d_x \]

- Forward differences calculations: Incremental calculation of the next value.
  - Linear case:
    \[ u_{k+1} = u_k + \delta, \quad k = 0, 1, 2 \ldots \quad u_0 = 0 \]
    \[ x_k = a_x u_k + b_x \]
    \[ x_{k+1} = a_x (u_k + \delta) + b_x \]
    \[ x_{k+1} = x_k + \Delta x \]
    \[ \Delta x = a_x \delta \]
• Cubic equations

\[ x_k = a_x u_k^3 + b_x u_k^2 + c_x u_k + d_x \]
\[ x_{k+1} = a_x (u_k + \delta)^3 + b_x (u_k + \delta)^2 + c_x (u_k + \delta) + d_x \]

\[ \Delta x_k = 3 a_x \delta u_k^2 + (3 a_x \delta^2 + 2 b_x \delta) u_k + (a_x \delta^3 + b_x \delta^2 + c_x \delta) \]

\[ \Delta x_{k+1} = \Delta x_k + \Delta^2 x_k \]
\[ \Delta^2 x_{k+1} = \Delta^2 x_k + \Delta^3 x_k \]

\[ \Delta^2 x_k = 6 a_x \delta^2 u_k + 6 a_x \delta^3 + 2 b_x \delta^2 \]
\[ \Delta^3 x_k = 6 a_x \delta^3 \]

\[ x_0 = d_x \]
\[ \Delta x_0 = a_x \delta^3 + b_x \delta^2 + c_x \delta \]
\[ \Delta^2 x_0 = 6 a_x \delta^3 + 2 b_x \delta^2 \]
\[ \Delta^3 x_k = 6 a_x \delta^3 \]
\[ x_0 = d_x \]
\[ \Delta x_0 = a_x \delta^3 + b_x \delta^2 + c_x \delta \]
\[ \Delta^2 x_0 = 6 a_x \delta^3 + 2 b_x \delta^2 \]

- **Example:**
  \((a_x, b_x, c_x, d_x) = (1, 2, 3, 4), \quad \delta = 0.1\)
  \[ \Delta^3 x_k = 6 \delta^3 = 0.006 \]

\[
\begin{array}{ccc}
\hline
x & \Delta x & \Delta^2 x \\
\hline
4.000 & 0.321 & 0.046 \\
4.321 & 0.367 & 0.052 \\
4.688 & 0.419 & 0.058 \\
5.107 & 0.477 & 0.064 \\
5.584 & 0.541 & 0.070 \\
6.125 & 0.611 & 0.076 \\
6.736 & 0.687 & 0.082 \\
7.423 & 0.769 & 0.088 \\
8.192 & 0.857 & 0.094 \\
9.049 & 0.951 & 0.100 \\
\hline
\end{array}
\]
Sweep Representations

• Use reflections, translations and rotations to construct new shapes.
Hierarchical Models

- Combine smaller/simpler shapes to construct complex objects and scenes.
- Stored in trees or similar data structures
- Operations are based on traversal of the tree
- Keeping information like bounding boxes in tree nodes accelerate the operations.
Scene Graphs

- DAG's (Directed Acyclic Graphs) to represent scenes and complex objects.

- Nodes: Grouping nodes, Transform nodes, Level Of Detail nodes, Light Source nodes, Attribute nodes, State nodes.
  Leaves: Object geometric descriptions.

- Why not tree but DAG?

- Available libraries: i.e. www.openscenegraph.org

- Efficient display of objects, picking objects, state change and animations.
Constructive Solid Geometry

- Combine multiple shapes with set operations (intersection, union, deletion) to construct new shapes.

\[ A \cup B \quad A \cap B \quad A - B \quad B - A \]
• Set operations and transformations combined:

\[
\text{union}(\text{transA(box)}, \text{diff}(\text{transB(box)}, \text{transC(cylinder)}))
\]
Ray casting methods are used for rendering and finding properties of volumes constructed with this method.

Simply +1 for outside inside
-1 for inside outside transition. Positives are solid.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E U B</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E N B</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E - B</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>B - E</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Similarly find unit cubes interior to calculate mass, center of mass etc.
Octrees

- Divide a volume in equal binary partitions in all dimensions recursively to represent solid object volumes. Combining leaf cubes give the volume.

- 2D: quadtree
• 2D: quadtree; 3D: octree

• Volume data: Medical data like Magnetic Resonance. Geographical info (minerals etc.)

• 2D: Pixel; 3D: voxel.

• Volumes consisting of large continuous subvolumes with properties. Volumes with many wholes, spaces. Surface information is not sufficient or tracktable.

• Keeping all volume in terms of voxels, too expensive: space and processor.
• 8 elements at each node.
• If volume completely resides in a cube, it is not further divided: leaf node
• Otherwise nodes recursively subdivided.
• Extends of a tree node is the extend of the cube it defines.
• Surfaces can be extracted by traversing the leaves with geometrical adjacency.
Fractal Geometry Methods

- Synthetic objects: regular, known dimension
- Natural objects: recursive (self repeating), the higher the precision, the higher the details you get.
- Example: tree branches, terrains, textures.
- Classification:
  - Self-similar: scaled-down shape is similar to original
  - Self-affine: self similar with different scaling parameters and transformations. Statistical when random parameters are involved.
  - Invariant: non-linear transformations, i.e. Complex space.
• Fractal dimension:
  - Detail variation of a self similar object. Denoted as $D$.
  - Fragmentation of the object.

\[ ns^D = 1 \]
\[ D = \frac{\ln n}{\ln (1/s)} \]

$n$: number of pieces  \hspace{1cm} s$: scaling factor

\[ n = 4 \quad s = 1/3 \quad D = \frac{\ln 4}{\ln 1/(1/3)} = 1.2619 \]
Random Mid-point Variation

- Find the midpoint of an edge A-B. Add a random factor and divide the edge in two as: A-M, M-A at each step.

- Usefull for height maps, clouds, plants.

2D:  
\[
\begin{align*}
    x_m &= \frac{(x_A + x_B)}{2} \\
    y_m &= \frac{(y_A + y_B)}{2} + r, \quad r: \text{a random value in } 0-c \\
    c &= c \times f, \quad f: \text{a fraction in } 0-1
\end{align*}
\]

3D: For corners of a square: A, B, C, D

\[
\begin{align*}
    z_{AB} &= \frac{(z_A + z_B)}{2} + r, \quad z_{BC} = \frac{(z_B + z_C)}{2} + r, \\
    z_{CD} &= \frac{(z_C + z_D)}{2} + r, \quad z_{DA} = \frac{(z_D + z_A)}{2} + r \\
    z_M &= \frac{(z_{AB} + z_{BC} + z_{CD} + z_{DA})}{4} + r
\end{align*}
\]