Review: Continuity
- continuity definitions
  - \( C^0 \): share join point
  - \( C^1 \): share continuous derivatives
  - \( C^2 \): share continuous second derivatives
- piecewise Bézier: no continuity guarantees

Review: Geometric Continuity
- derivative continuity is important for animation
  - if object moves along curve with constant parametric speed, should be no sudden jump at knots
- for other applications, tangent continuity suffices
  - requires that the tangents point in the same direction
  - referred to as \( G^1 \) geometric continuity
- curves could be made \( C^2 \) with a re-parameterization
  - geometric version of \( C^2 \) is \( G^2 \), based on curves having the same radius of curvature across the knot

Achieving Continuity
- Hermite curves
  - user specifies derivatives, so \( C^1 \) by sharing points and derivatives across knot
- Bézier curves
  - they interpolate endpoints, so \( C^0 \) by sharing control points
  - introduce additional constraints to get \( C^1 \)
    - parametric derivative is a constant multiple of vector joining first/last 2 control points
    - \( C^1 \) achieved by making \( P_{i+1} \) colinear, with \( J = \frac{P_{i+1} - P_i}{t} \)
    - \( C^2 \) comes from further constraints on \( P_{i+1} \) and \( P_{i+2} \)
  - leads to...

B-Spline Curve
- start with a sequence of control points
- select four from middle of sequence
- B-Spline doesn’t interpolate (touch) any of them but approximates the going through \( P_3 \) and \( P_4 \)
- Bezier and Hermite go between \( P_3 \) and \( P_4 \)
**B-Spline**
- by far the most popular spline used
- $C_0$, $C_1$, and $C_2$ continuous

![B-Spline Diagram](image1.png)

---

**Reading**
- FCG Chapter 28 Spatial Field Visualization
  - Chap 23 (2nd ed)

**Surface Graphics**
- objects explicitly defined by surface or boundary representation
- mesh of polygons

![Surface Graphics Diagram](image2.png)

---

**Volume Graphics**
- pros
  - formidable technique for data exploration
  - hardware acceleration cheap
- cons
  - rendering algorithm has high complexity!
  - special purpose hardware costly (~$3K-$10K)

![Volume Graphics Example](image3.png)

---

**Isosurfaces**
- 2D scalar fields: isolines
- contour plots, level sets
- topographic maps
- 3D scalar fields: isosurfaces

![Isosurfaces Example](image4.png)

---

**Spatial/Scientific Visualization**
- for some data, difficult to create polygonal mesh
- voxels: discrete representation of 3D object
- volume rendering: create 2D image from 3D object
- translate raw densities into colors and transparencies
- different aspects of the dataset can be emphasized via changes in transfer functions

![Spatial/Scientific Visualization Example](image5.png)

---

**MC 1: Create a Cube**
- consider a cube defined by eight data values

![MC 1 Diagram](image6.png)

---

**MC 2: Classify Each Voxel**
- classify each voxel according to whether lies outside the surface (value > iso-surface value)
- inside the surface (value <= iso-surface value)

![MC 2 Diagram](image7.png)

---

**MC 3: Build An Index**
- binary labeling of each voxel to create index

![MC 3 Diagram](image8.png)

---

**MC 4: Lookup Edge List**
- use index to access array storing list of edges
- all 256 cases can be derived from 15 base cases

![MC 4 Diagram](image9.png)
MC 4: Example

- index = 00000001
- triangle 1 = a, b, c

MC 5: Interpolate Triangle Vertex

- for each triangle edge
- find vertex location along edge using linear interpolation of voxel values

MC 6: Compute Normals

- calculate the normal at each cube vertex
- use linear interpolation to compute the polygon vertex normal

MC 7: Render!

Direct Volume Rendering
- do not compute surface

Rendering Pipeline
- data set has application-specific values
  - temperature, velocity, proton density, etc.
  - assign these to color/opacity values to make sense of data
  - achieved through transfer functions

Transfer Functions
- map data value to color and opacity

Volume Rendering Algorithms
- ray casting
  - image order, forward viewing
- splatting
  - object order, backward viewing
- texture mapping
  - object order
  - back-to-front compositing
Ray Traversal Schemes

Ray Traversal - First
• first: extracts iso-surfaces (again!)

Ray Traversal - Average
• average: looks like X-ray

Ray Traversal - MIP
• max: Maximum Intensity Projection
  • used for Magnetic Resonance Angiogram

Ray Traversal - Accumulate
• accumulate: make transparent layers visible

Splatting
• each voxel represented as fuzzy ball
  • 3D gaussian function
  • RGBA value depends on transfer function
  • fuzzy balls projected on screen, leaving footprint called splat
  • composite front to back, in object order

Texture Mapping
• 2D: axis aligned 2D textures
  • back to front compositing
  • commodity hardware support
  • must calculate texture coordinates, warp to image plane
• 3D: image aligned 3D texture
  • simple to generate texture coordinates