Spatial/Scientific Visualization

Week 12, Fri Apr 9

http://www.ugrad.cs.ubc.ca/~cs314/Vjan2010
News

• Reminders
  • H4 due Mon 4/11 5pm
  • P4 due Wed 4/13 5pm
• Extra TA office hours in lab 005 for P4/H4
  • Fri 4/9 11-12, 2-4 (Garrett)
  • Mon 4/12 11-1, 3-5 (Garrett)
  • Tue 4/13 3:30-5 (Kai)
  • Wed 4/14 2-4, 5-7 (Shailen)
  • Thu 4/15 3-5 (Kai)
  • Fri 4/16 11-4 (Garrett)
Cool Pixar Graphics Talk Today!!

• The Funnest Job on Earth: A Presentation of Techniques and Technologies Used to Create Pixar's Animated Films (version 2.0)
• Wayne Wooten, Pixar
• Fri 4/9, 4:00 to 5:30 pm, Dempster 110
  • great preview of CPSC 426, Animation :-)

• overlaps my usual office hours :-(
• poll: who was planning to come today?
Project 4

• I've now sent proposal feedback on proposals to everyone where I have specific concerns/responses
  • no news is good news
• global reminders/warnings
  • you do need framerate counter in your HUD!
  • be careful with dark/moody lighting
    • can make gameplay impossible
    • backup plan: keystroke to brighten by turning more/ambient light
• reminder on timestamps
  • if you demo on your machine, I will check timestamps of files to ensure they match code you submitted through handin
  • they must match! do *not* change anything in the directory
  • clone code into new directory to keep developing or fix tiny bugs
    • so that I can quickly check that you've not changed anything else
Review: GPGPU Programming

• General Purpose GPU
  • use graphics card as SIMD parallel processor
  • textures as arrays
  • computation: render large quadrilateral
  • multiple rendering passes
Review: Splines

• *spline* is parametric curve defined by *control points*
  • *knots*: control points that lie on curve
  • engineering drawing: spline was flexible wood, control points were physical weights

A Duck (weight)

Ducks trace out curve
Review: Hermite Spline

- user provides
  - endpoints
  - derivatives at endpoints
Review: Bézier Curves

- four control points, two of which are knots
  - more intuitive definition than derivatives
- curve will always remain within convex hull (bounding region) defined by control points
Review: Basis Functions

• point on curve obtained by multiplying each control point by some basis function and summing
Review: Comparing Hermite and Bézier

Hermite

Bézier
Review: Sub-Dividing Bézier Curves

- find the midpoint of the line joining $M_{012}$, $M_{123}$, call it $M_{0123}$
Review: de Casteljau’s Algorithm

- can find the point on Bézier curve for any parameter value \( t \) with similar algorithm
  - for \( t=0.25 \), instead of taking midpoints take points 0.25 of the way

demo: [www.saltire.com/applets/advanced_geometry/spline/spline.htm](http://www.saltire.com/applets/advanced_geometry/spline/spline.htm)
Review: Continuity

- piecewise Bézier: no continuity guarantees

- continuity definitions
  - $C^0$: share join point
  - $C^1$: share continuous derivatives
  - $C^2$: share continuous second derivatives

![Diagram showing different levels of continuity](image)
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 $C^1$ 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

• Bezier curves
  • they interpolate endpoints, so $C^0$ by sharing control pts
  • introduce additional constraints to get $C^1$
    • parametric derivative is a constant multiple of vector joining first/last 2 control points
    • so $C^1$ achieved by setting $P_{0,3}=P_{1,0}=J$, and making $P_{0,2}$ and $J$ and $P_{1,1}$ collinear, with $J-P_{0,2}=P_{1,1}-J$
    • $C^2$ comes from further constraints on $P_{0,1}$ and $P_{1,2}$

• leads to...
B-Spline Curve

• start with a sequence of control points
• select four from middle of sequence
  \((p_{i-2}, p_{i-1}, p_i, p_{i+1})\)
  • Bezier and Hermite goes between \(p_{i-2}\) and \(p_{i+1}\)
  • B-Spline doesn’t interpolate (touch) any of them but approximates the going through \(p_{i-1}\) and \(p_i\)
B-Spline

• by far the most popular spline used
• \( C_0, \ C_1, \) and \( C_2 \) continuous

demo: www.siggraph.org/education/materials/HyperGraph/modeling/splines/demoprog/curve.html
B-Spline

- locality of points

*Figure 10-41*
Local modification of a B-spline curve. Changing one of the control points in (a) produces curve (b), which is modified only in the neighborhood of the altered control point.
Geometric Modelling

• much, much more in CPSC 424!
  • offered next year
Spatial/Scientific Visualization
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

• pros
  • fast rendering algorithms available
  • hardware acceleration cheap
  • OpenGL API for programming
  • use texture mapping for added realism

• cons
  • discards interior of object, maintaining only the shell
  • operations such cutting, slicing & dissection not possible
  • no artificial viewing modes such as semi-transparencies, X-ray
  • surface-less phenomena such as clouds, fog & gas are hard to model and represent
Volume Graphics

- 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
Volume Graphics

• pros
  • formidable technique for data exploration

• cons
  • rendering algorithm has high complexity!
  • special purpose hardware costly (≈$3K-$10K)

volumetric human head (CT scan)
Isosurfaces

- 2D scalar fields: isolines
  - contour plots, level sets
  - topographic maps
- 3D scalar fields: isosurfaces
Volume Graphics: Examples

- anatomical atlas from visible human (CT & MRI) datasets
- industrial CT - structural failure, security applications
- flow around airplane wing
- shockwave visualization: simulation with Navier-Stokes PDEs
Isosurface Extraction

• array of discrete point samples at grid points
  • 3D array: voxels
• find contours
  • closed, continuous
  • determined by iso-value
• several methods
  • marching cubes is most common

Iso-value = 5
MC 1: Create a Cube

- consider a cube defined by eight data values
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 3: Build An Index

• binary labeling of each voxel to create index

<table>
<thead>
<tr>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
<th>v6</th>
<th>v7</th>
<th>v8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Index: v1, v2, v3, v4, v5, v6, v7, v8
MC 4: Lookup Edge List

- use index to access array storing list of edges
  - all 256 cases can be derived from 15 base cases

The 15 Cube Combinations
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

\[ x = i + \left( \frac{T - v[i]}{v[i+1] - v[i]} \right) \]

- T=5
- T=8

\( v[i] = 10 \)
\( v[i+1] = 0 \)
MC 6: Compute Normals

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

\[
G_x = v_{i+1,j,k} - v_{i-1,j,k}
\]
\[
G_y = v_{i,j+1,k} - v_{i,j-1,k}
\]
\[
G_z = v_{i,j,k+1} - v_{i,j,k-1}
\]
MC 7: Render!
Direct Volume Rendering

- do not compute surface
Rendering Pipeline

Classify
Classification

- 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
Transfer Functions

Human Tooth CT

Gordon Kindlmann
Setting Transfer Functions

- can be difficult, unintuitive, and slow
Rendering Pipeline
Light Effects

- usually only consider reflected part

\[ \text{Light} = \text{refl.} + \text{absorbed} + \text{trans.} \]

\[ \text{Light} = \text{ambient} + \text{diffuse} + \text{specular} \]

\[ I = k_a I_a + k_d I_d + k_s I_s \]
Rendering Pipeline

Classify → Shade → Interpolate
Interpolation

2D
- given:
- needed:
- nearest neighbor

1D
- given:
- needed:
- linear
Rendering Pipeline

Classify → Shade → Interpolate → Composite
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

Intensity

Max

Average

Accumulate

First

Depth
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