Volume Rendering, Part 2

November 15th, 2013  Hank Childs, University of Oregon
Announcements

• Final project
  – Proposals due today
    • Worth 0% of your grade
      – ½ credit if late
  – No proposals for pre-defined project ... just available

  – Two prompts:
    • Volume rendering
    • Make a movie
      – VTK
      – VisIt
Upcoming Schedule

- 11/15: Volume rendering part 2
- 11/20: VisIt
- 11/22: uncertainty visualization (KP)
- 11/27: alternate lecture (force directed layouts)
  - 11/21 & 12/2
- 12/4: medical visualization (Eric Anderson)
- 12/6: unstructured grids
- Faculty fireside (parallel visualization):
  - Week of 12/2 or week of 12/9
- Final: Thurs 12/12, 3:15PM, Rm 220?
Volume rendering

- Technique for rendering entire volume at one time
- Image order: iterate over pixels (via rays)
- Object order: iterate over data
- Typical method of determining color:
  - Compositing (use transfer function)
    - But there are alternatives to compositing
Ray casting game plan:
For every pixel on the screen,
Find ray for that pixel
Intersect volume with ray
Calculate color from intersection
Assign color to pixel

Which of these 4 steps will be easy? Which will be hard?
Outline

• Find Ray For That Pixel
• Intersect Volume With Ray
• Calculate Color From Intersection
• Assign Color To Pixel
From Pixels to Rays

\[
\begin{align*}
\mathbf{r} & = \frac{\text{look} \times \mathbf{u}}{|\text{look} \times \mathbf{u}|} \\
\mathbf{v} & = \frac{\text{look} \times \mathbf{r}}{|\text{look} \times \mathbf{u}|} \\
\Delta x & = \frac{2 \tan(\text{fov}_x / 2)}{W} \frac{\mathbf{r}}{u} \\
\Delta y & = \frac{2 \tan(\text{fov}_y / 2)}{H} \frac{\mathbf{r}}{v} \\
d(i, j) & = \frac{\text{look}}{|\text{look}|} + \frac{(2i + 1 - W)}{2} \Delta x + \frac{(2j + 1 - H)}{2} \Delta y
\end{align*}
\]

This answers the “find ray for this pixel” question
Outline

• Find Ray For That Pixel
• Intersect Volume With Ray
• Calculate Color From Intersection
• Assign Color To Pixel
Ray-Volume Intersection: sampling

Sampling is the most common method for “ray-casting” volume rendering

Do we know how to do this sampling?
How to sample quickly

• Multiple strategies.
• For now, similar as before:
  – Find first cell intersected
    • Intersection is at a face
  – Find where ray exits that cell
    • Are there samples within the cell? Then sample them!!
  – Go to next cell (which shares a face) and repeat
  – Keep going until you exit the volume, one cell at a time, and see what samples it covers

Approximately how many samples will we calculate?
Outline

• Find Ray For That Pixel
• Intersect Volume With Ray
• Calculate Color From Intersection
• Assign Color To Pixel
Transfer Function

Volume
Var: hardyglobal
Units: Joules

Max: 5.890
Min: 1.096
“Alpha Channel”

- Represents opacity
  - 1.0 or 255: fully opaque
  - 0: fully transparent

- Stored alongside RGB
  - Referred to as RGBA

- Floating point (1.0) vs byte (255):
  - Precision vs uniformity with RGB & performance
Transparency

Quiz: If you have a red square that is 50% opaque in front of a black background, what color would you see?

– Represent your answer in terms of (R, G, B)

Answer: (128, 0, 0)
Representing Colors

• Two ways of representing colors
  – Pre-multiplication
  – Straight RGBA (non-premultiplied)

• Slides use straight RGBA.
  – Will describe pre-multiplication afterwards

Wikipedia page is great reference:
http://en.wikipedia.org/wiki/Alpha_compositing
Formula For Transparency

• Front = (Fr,Fg,Fb,Fa)
• Back = (Br,Bg,Bb,Ba)
• Equation = (Fa*Fr+(1-Fa)*Br,
              Fa*Fg+(1-Fa)*Bg,
              Fa*Fb+(1-Fa)*Bb,
              Fa+(1-Fa)*Ba)

Alpha component is important! Any observations?
Transparency

- If you have an 50% transparent red square (255,0,0,128) in front of an opaque blue square (0,0,255,255), what color would you see (in RGB)?
  - (128,0,64)

- If you have an 50% transparent blue square (0,0,255,128) in front of an opaque red square (255,0,0,255), what color would you see (in RGB)?
  - (64,0,128)
Calculating Color

<table>
<thead>
<tr>
<th>Sample</th>
<th>Scalar Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.8</td>
</tr>
<tr>
<td>1</td>
<td>4.7</td>
</tr>
<tr>
<td>2</td>
<td>5.8</td>
</tr>
<tr>
<td>3</td>
<td>3.5</td>
</tr>
<tr>
<td>4</td>
<td>1.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>0</td>
<td>0.75</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
</tbody>
</table>

- Equation = \((Fa \times Fr + (1-Fa) \times Br, Fa \times Fg + (1-Fa) \times Bg, Fa \times Fb + (1-Fa) \times Bb, Fa + (1-Fa) \times Ba)\)

Apply this equation over and over until you run out of samples (then use background color)
Opacity Adjustment

• Consider ray with this profile

What is the opacity of these three samples together?
Opacity Adjustment

• Consider ray with this profile

New interpretation: opacity is for a range, not a point. If you double the sampling, then the range is reduced. Typically think of opacity for a given sampling rate and then adjust as sampling rate changes.

What is the opacity of these five samples together?
Outline

• Find Ray For That Pixel
• Intersect Volume With Ray
• Calculate Color From Intersection
• Assign Color To Pixel
Assign Color To Pixel

• Allocate a buffer for storing RGB values
  – Buffer should have one RGB for every pixel on the screen.

• As you calculate color for a ray, assign that color to its corresponding buffer entry

• When you have all of the colors, put that image up on the screen, as if you had rendered it using graphics cards.
Volume rendering overview

Ray casting game plan:
For every pixel on the screen,
Find ray for that pixel
Intersect volume with ray
Calculate color from intersection
Assign color to pixel

Which of these 4 steps were easy? Which were hard?
Volume rendering

- Technique for rendering entire volume at one time
- Image order: iterate over pixels (via rays)
- Object order: iterate over data
- Typical method of determining color:
  - Compositing (use transfer function)
    - But there are alternatives to compositing
Splatting

• Turn every point into a disk (aligned to the camera)
  – Color and transparency vary
• Render the splats from back to front using graphics hardware.
Transparent Planes

- **Strategy:**
  - Slice the volume by many planes (200-1000)
  - Apply transfer function to each vertex on the plane
  - Result: plane with variation in color and transparency
  - Render the planes from back to front

These can be quickly rendered using “2D textures” or “3D textures.”

*Image from VTK book*
Volume rendering

- Technique for rendering entire volume at one time
- Image order: iterate over pixels (via rays)
- Object order: iterate over data
- Typical method of determining color:
  - Compositing (use transfer function)
    - But there are alternatives to compositing
Ray functions: compositing
Ray functions: maximum

Image from VTK book
Ray functions: average value
Ray functions: distance to value
Volume rendering

- More on volume rendering
  - Shading
  - Multi-variate volume rendering
  - Optimizations
  - Combinations with surfaces
Shading surfaces

This is done by calculating surface normal and then calculating light reflection (or lack of light reflection from light source)
Shading volumes

Want to do all the same lighting equations, but we need a surface normal ... for a volume. What to do?

Answer: use gradient of field for “surface” normal
Volume rendering

- More on volume rendering
  - Shading
  - Multi-variate volume rendering
  - Optimizations
  - Combinations with surfaces
Multi-variate volume rendering

- Simplest form
Multi-variate transfer functions
Multi-variate transfer functions
Volume rendering

- More on volume rendering
  - Shading
  - Multi-variante volume rendering
  - Optimizations
  - Combinations with surfaces
Optimizing Volume Rendering

• Big topic:
  – How to find samples quickly?
  – How to use advanced HW (GPUs) efficiently?

• Early ray termination
  – Just stop going when opacity gets greater than some threshold.
Volume rendering

• More on volume rendering
  – Shading
  – Multi-variate volume rendering
  – Optimizations
  – Combinations with surfaces
Surface Rendering + Volume Rendering

How was this picture made?
And now...

- You know everything I think you should know after having taken a sci-vis class.
- I am proud of this class.
- Next few weeks should be fun lectures.
- Projects will hopefully be fun too.