Volume Rendering, pt 2

Hank Childs, University of Oregon
Announcements

- Project 8 statement
- Need a lecture on moviemaking?
- Grad students:
  - still need to do short presentations
    - I will do one soon
    - Come to OH and let’s chat
Review
X-rays

Emitter

Film/image
Volume rendering

- Important visualization technique for 3D data
- Use combination of color and transparency to see entire 3D data set at one time.

There are multiple ways to do volume rendering. I will describe one way today (raycasting). That will help explain the technique. I will describe alternate ways in future lectures.
Volume rendering overview

Camera

Pixels on the screen

3D data
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 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
Outline

• Find Ray For That Pixel
• Intersect Volume With Ray
• Calculate Color From Intersection
• Assign Color To Pixel
How do we specify a camera?

The “viewing pyramid” or “view frustum”.

Frustum: In geometry, a frustum (plural: frusta or frustums) is the portion of a solid (normally a cone or pyramid) that lies between two parallel planes cutting it.

class Camera
{
public:
    double near, far;
    double angle;
    double position[3];
    double focus[3];
    double up[3];
};
From Pixels to Rays

From: Zhang, CSUSM, Introduction to Ray Tracing

This answers the “find ray for this pixel” question

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

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

What is the result of the ray-volume intersection?
Ray-Volume Intersection

How would you calculate this intersection?
Ray-Volume Intersection: How do you do it?

• Find first cell intersected
  – Intersection is at a face
• Find where ray exits that cell
  – That face is shared with another cell
• Keep going until you exit the volume, one cell at a time.

What would this technique give you?
Intersect Volume With Ray

What data would we be working with?
Face Intersections

Can we calculate the intersections at the faces? Can we calculate the data at the faces?

Is this a good idea? Why or why not?

(it is not a good idea if we interpolate between the face values)

How do we fix this?
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?
New Material
Outline

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

Pseudocolor
Var: hardyglobal
Units: Joules

1.096  2.294  3.493  4.691  5.890
Transfer function

- Assigns a color and an opacity to each scalar value.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Abstraction for assigning color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pseudocoloring</td>
<td>Color Map</td>
</tr>
<tr>
<td>Volume Rendering</td>
<td>Transfer Function</td>
</tr>
</tbody>
</table>
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
Applying a transfer function

**Sample** | **Scalar Value** |
---|---|
0 | 5.8 |
1 | 4.7 |
2 | 5.8 |
3 | 3.5 |
4 | 1.1 |

<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>255</td>
<td>0.75</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>0</td>
<td>255</td>
<td>255</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
</tbody>
</table>

Quiz: calculate (approximate) results from transfer function for each sample

---

1.096 | 2.294 | 3.493 | 4.691 | 5.890

---

UNIVERSITY OF OREGON
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)
Transparency

• If you have an opaque red square in front of a blue square, what color would you see?  
  – Red

• If you have a 50% transparent red square in front of a blue square, what color would you see?  
  – Purple

• If you have a 100% transparent red square in front of a blue square, what color would you see?  
  – Blue
Formula For Transparency

- Front = (Fr,Fg,Fb,Fa)
  - a = alpha, transparency factor
    - Sometimes percent
    - Typically 0-255, with 255 = 100%, 0 = 0%
- Back = (Br,Bg,Bb,Ba)
- Equation = (Or = Fa*Fr+(1-Fa)*Ba*Br,
  Og = Fa*Fg+(1-Fa)*Ba*Bg,
  Ob = Fa*Fb+(1-Fa)*Ba*Bb,
  Oa = Fa+(1-Fa)*Ba)

Alpha component is important! Any observations?
Representing Colors

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

• Slides use straight RGBA

Wikipedia page is great reference:
http://en.wikipedia.org/wiki/Alpha_compositing
Representing colors: example

- 50% transparent red square
- Could use
  - (128, 0, 0, 0.5)  # pre-multiplied
  - --or--
  - (255, 0, 0, 0.5)  # straight RGBA
- The issue is whether the 0.5 is multiplied into the color components first.
- And changes equation
Formula For Transparency

- **Front** = (Fr, Fg, Fb, Fa)
  
  - \(a = \alpha\), transparency factor
  
  - Sometimes percent
  
  - Typically 0-255, with 255 = 100%, 0 = 0%

- **Back** = (Br, Bg, Bb, Ba)

- **Equation** =
  
  - \(O_r = Fa \times Fr + (1-Fa) \times Ba \times Br\),
  
  - \(O_g = Fa \times Fg + (1-Fa) \times Ba \times Bg\),
  
  - \(O_b = Fa \times Fb + (1-Fa) \times Ba \times Bb\),
  
  - \(O_a = Fa + (1-Fa) \times Ba\)

Alpha component is important! Any observations?
Formula For Transparency

• Front = (Fr, Fg, Fb, Fa)
  – a = alpha, transparency factor
    • Sometimes percent
    • Typically 0-255, with 255 = 100%, 0 = 0%
• Back = (Br, Bg, Bb, Ba)

• Equation = (Or = Fr+(1-Fa)*Ba*Br,
  Og = Fg+(1-Fa)*Ba*Bg,
  Ob = Fb+(1-Fa)*Ba*Bb,
  Oa = Fa+(1-Fa)*Ba)

Alpha component is important! Any observations?
Transparency

• If you have an 25% transparent red square (255,0,0) in front of a blue square (0,0,255), what color would you see (in RGB)?
  – (192,0,64)

• If you have an 25% transparent blue square (0,0,255) in front of a red square (255,0,0), what color would you see (in RGB)?
  – (64,0,192)
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 = \((Fa*Fr+(1-Fa)*Ba*Br, Fa*Fg+(1-Fa)*Ba*Bg, Fa*Fb+(1-Fa)*Ba*Bb, Fa+(1-Fa)*Ba))\)

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

• Initialize pixel color / alpha with TransferFunction(first sample)

• Iterate over remaining samples
  – Front-to-back order
  – pixel color / alpha updated each time using composite function
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?
Ray-Volume Intersection: sampling

What happens when we change the sampling rate?

Imagine if we had half or twice as many transparent squares...

Should the picture change if we change the sampling rate?
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?
Opacity Correction

The assigned opacity also depends on the sampling rate. For example, when using fewer slices, the opacity has to be scaled up, so that the overall intensity of the image remains the same. Equation 3 is used for correcting the transfer function opacity whenever the user changes the sampling rate $s$ from the reference sampling rate $s_0$:

**Equation 3 Formula for Opacity Correction**

$$ A = 1 - (1 - A_0)^{s_0/s} $$

Next Time

• A lot more about volume rendering
  – More techniques
  – Multi-variate volume rendering
  – Lighting calculations
  – Optimizations
  – Combinations with surface rendering
New Material: Volume Rendering
Another idea for rendering (triangles)

World space:
Triangles in native Cartesian coordinates
Camera located anywhere

Camera space:
Camera located at origin, looking down -Z
Triangle coordinates relative to camera frame

Image space:
All viewable objects within
-1 <= x,y,z <= +1

Screen space:
All viewable objects within
-1 <= x, y <= +1

Device space:
All viewable objects within
0 <= x <= width, 0 <= y <= height

We don’t need to do it this way for volume rendering (although we could)
Different Types of Volume Rendering

- Image Order: iterate over pixels
- Object Order: iterate over data
Object Order: 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.
Object Order: 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: A Framework For Rendering

• Compositing: use combination of color and transparency to enable visualization of entire 3D data

• Alternate ideas:
  – maximum value along ray
  – average value along ray
  – distance to key value along ray
Ray functions: compositing
Ray functions: maximum
Ray functions: average value
Ray functions: distance to value
Volume rendering

• More on volume rendering
  – Shading
  – Multi-variate volume rendering
  – Optimizations
  – Combinations with 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

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>32°F</td>
<td>1.096</td>
</tr>
<tr>
<td>77°F</td>
<td>2.294</td>
</tr>
<tr>
<td>122°F</td>
<td>3.493</td>
</tr>
<tr>
<td>167°F</td>
<td>4.691</td>
</tr>
<tr>
<td>212°F</td>
<td>5.890</td>
</tr>
</tbody>
</table>

Max: 5.890
Min: 1.096
Multi-variate transfer functions
Multi-variate transfer functions
Volume rendering

• More on volume rendering
  – Shading
  – Multi-variate 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.
    • You do this for pre-defined project
Volume rendering

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

How was this picture made?