PHILIPS
sense and simplicity

3D Visualization @ Philips Healthcare, interventional X-Ray

Danny Ruijters
26 Juni 2012
About me

• Danny Ruijters
• Engineering degree: RWTH Aachen University of Technology
• MSc: ParisTech
• PhD: KU Leuven, TU/eindhoven
• With Philips since 2001
• Currently Principal Scientist @ interventional X-Ray innovation
Introduction
Philips Healthcare
PHILIPS

Produces image data
What is Coronary Artery Disease?

• Coronary Artery Disease (CAD) is a condition in which plaque builds up inside the coronary arteries. These arteries supply the heart muscle with oxygen-rich blood.

• Plaque is made up of fat, cholesterol, calcium, and other substances found in the blood. When plaque builds up in the arteries, the condition is called atherosclerosis.
Buildup of plaque
Stenosis of coronary arteries
Visualization

- Volume Rendering
- Fast Volume Rendering
- Fused Visualization
Volume Rendering @ Philips

- Own implementation, no VTK etc
- C++
- OpenGL
- Platform independent
Volume rendering: infinitelly small element

\[ c_i', \alpha_i \]

\[ c_i' = c_i \alpha_i + c_{i+1}' (1 - \alpha_i) \]
Row of infinitely small elements

\[
c'_0 = c_0 \alpha_0 + c'_1 (1-\alpha_0) \\
c'_1 = c_1 \alpha_1 + c'_2 (1-\alpha_1) \\
c'_0 = c_0 \alpha_0 + c'_1 (1-\alpha_0) \\
= c_0 \alpha_0 + (c_1 \alpha_1 + c'_2 (1-\alpha_1))(1-\alpha_0) \\
= c_0 \alpha_0 + c_1 \alpha_1 (1-\alpha_0) + c'_2 (1-\alpha_0)(1-\alpha_1) \\
= \Sigma (c_n \alpha_n \prod(1-\alpha_m))
\]
\[ \sum_0^{\infty} \left( c_n \alpha_n \prod_0^{n-1} (1-\alpha_m) \right) \]

Color contribution per element: \( \text{color}_n = c_n \alpha_n \)

Transparency per element: \( \alpha_n = 1 - e^{-\alpha'} \)

\( \prod (1-\alpha_m) = \prod e^{-\alpha'} = e^{-\sum \alpha'} \)

\[ \sum_0^{\infty} \left( c_n \alpha_n \prod_0^{n-1} (1-\alpha_m) \right) = \sum_0^{\infty} \left( \text{color}_n \cdot e^{-\sum \alpha'} \right) \]

\[ \int_0^{D} \text{color}(\lambda) \cdot e^{-\int_0^{\lambda} \alpha(\lambda') d\lambda'} d\lambda \]
\[
\int_0^D \text{color}(\bar{x}(\lambda)) \cdot e^{-\int_0^\lambda \alpha(\bar{x}(\lambda')) d\lambda'} d\lambda
\]
GPU Volume Rendering

Engel and Ertl: “Interactive high-quality volume rendering with flexible consumer graphics hardware,”
Eurographics 2002

\[ i = \sum_{n=0}^{N} (\alpha_n c_n \cdot \prod_{n'=0}^{n} (1 - \alpha_{n'})) \]
Visualization

- Volume Rendering
- Fast Volume Rendering
- Fused Visualization
GPU Bottlenecks

- **Video Memory**
  - Geometry
  - Commands
  - Textures
  - Frame Buffer
  - Texture Cache Size Limited

- **On-Chip Cache Memory**
  - Pre-TnL Cache
  - Post-TnL Cache
  - Triangle Setup
  - Rasterization
  - Fragment Shading and Raster Operations
  - Triangle Throughput Limited
  - Rasterization Limited
  - Fragment Shader Limited

- **Bus Limited**

---

**GPU Processes**
- **Geometry**
- **Commands**
- **Textures**
- **Frame Buffer**
- **Texture Cache**
- **Vertex Shading (T&L)**
- **Triangle Setup**
- **Rasterization**
- **Fragment Shading and Raster Operations**

---

**CPU**
Bricking

- Raw data
- One texture per brick
- Skip “empty” bricks
- Overlap: 0/1/2 voxels
- Memory overhead
Octree

- One octree per brick
- Transfer function
- Traverse hierarchy
- Smallest octree level
Early ray termination

\[ C_{i+1} = (1 - A_i) \cdot \alpha_i \cdot c_i + C_i \]
\[ A_{i+1} = (1 - A_i) \cdot \alpha_i + A_i \]

- Front-to-back rendering: under operator
- Front faces of each brick
- Early z-test
- Polygons are still rasterized…
- But prevents texture lookups and fragment shader execution
- Not useful for (very) sparse data sets
Phong Shading

\[ I = I_a k_a + I_i (k_d (L \cdot N) + k_s (R \cdot V)^n) \]
To phong or not to phong?

\[ I = I_a k_a + I_i (k_d(L \cdot N) + k_s(R \cdot V)^n) \]
## Results


<table>
<thead>
<tr>
<th>Graphics card</th>
<th>(a) Optimized</th>
<th>(b) Non-optimized</th>
<th>(a) / (b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nVidia QuadroFX 3400</td>
<td>73.5 fps</td>
<td>9.6 fps</td>
<td>7.66</td>
</tr>
<tr>
<td>ATi FireGL X1, xy aligned</td>
<td>83.3 fps</td>
<td>0.23 fps</td>
<td>362</td>
</tr>
<tr>
<td>ATi FireGL X1, non xy aligned</td>
<td>27.4 fps</td>
<td>0.23 fps</td>
<td>119</td>
</tr>
<tr>
<td>3Dlabs Wildcat 7110</td>
<td>21.3 fps</td>
<td>0.38 fps</td>
<td>56.1</td>
</tr>
</tbody>
</table>

![Graph of results](image_url)
Visualization

- Volume Rendering
- Fast Volume Rendering
- Fused Visualization
Fused visualization
Back to Front Volume Rendering

Over-operator:

\[ I_i = c_i \alpha_i + I_{i-1} (1-\alpha_i) \]

Z-buffer determines whether a sample origins from the mesh, or the voxel data.
First render mesh
Then draw a slab of the morphological dataset

Ruijters, Babic, ter Haar Romeny, Suetens: “Silhouette Fusion of Vascular and Anatomical Data,” ISBI'06, pp. 121-124
Example
Stencil buffer
Fused Visualization

38 fps
nVidia QuadroFX 3400

Ruijters, Babic, Homan, Mielekamp, ter Haar Romeny, Suetens:
"Real-time integration of 3-D multimodality data in interventional neuroangiography",
J. Electronic Imaging 18(3), 2009
Clinical Applications

• Needle guidance
Needle planning
Intra-operative registration
Needle Navigation

Conclusions
Conclusions

• Fast volume rendering is important during interventions
• Volume rendering can be accelerated by smart tailoring towards the graphics hardware
• Fused visualization can be used to present multiple datasets in one combined image
• This enables new clinical procedures
Thank you!!!