GPU-Based Cell Projection for Large Structured Data Sets

8 – 11 March, 2007      Barcelona, Spain

André Maximo
Ricardo Marroquim
Ricardo Farias
Claudio Esperança
Introduction

• Different approaches for Volume Rendering:
  • 3D Texture
Introduction

• Different approaches for Volume Rendering:

  • 3D Texture
  • Mesh
Introduction

- Different approaches for Volume Rendering:
  - 3D Texture $\rightarrow$ texels/s
  - Mesh $\rightarrow$ triangles/s
Introduction

- Graphics card features:
  - GPU Memory
  - CPU – GPU bus bandwidth
Proposal

- Common solution → 3D Texture
- Large data sets
- Mesh → Cell Projection
- GPU
Volunit

- Split 1 hexahedron into 5 tetrahedra
  - 1 hexahedron $\rightarrow$ 8 voxels $\rightarrow$ volunit
Algorithm Overview

- Project the *volunit* → PT  [Shirley and Tuchman 1990]
- Sorting → O(1)
- Create Vertex Array Data Structure → *Volunit*
- Render the volume → GPU
Basis Hexahedron Projection

- Recent implementation of PT [Marroquim et al. 2006]
  - Basis projection class
  - Basis projected vertices coordinates
  - Basis thick vertex coordinates
  - Basis intersection parameters for computing $s_f$ and $s_b$
  - Basis rendering order
Traversals Order

- Cell Projection worst disadvantage → Sorting

- Regular data → Traversal Order
Allocate Vertex Array

- Optimizing with Vertex Array

- Setup the arrays for each volunit
GPU Rendering pipeline

- Each *volunit* is sent as 5 triangle fans
- Basis hexahedron data are stored as uniform variables
- Auxiliary textures are stored on GPU memory
Volume Interaction

- Interactive transfer function editing
- Volume clipping
Video

GPU-Based Cell Projection for Large Structured Data Sets
Results
Results

- High tetrahedra/s performance
- No GPU memory used
- More time spent on GPU than CPU

Table 1: Average frames and tetrahedra per second.

<table>
<thead>
<tr>
<th>Data set</th>
<th># Verts</th>
<th># Tet</th>
<th>fps</th>
<th>M tet/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel</td>
<td>262 K</td>
<td>1.2 M</td>
<td>70.78</td>
<td>88.5/5.99</td>
</tr>
<tr>
<td>ToothC</td>
<td>1 M</td>
<td>5 M</td>
<td>1.22</td>
<td>13.1/6.30</td>
</tr>
<tr>
<td>Tooth</td>
<td>10 M</td>
<td>52 M</td>
<td>0.24</td>
<td>12.7/6.61</td>
</tr>
<tr>
<td>Foot</td>
<td>16 M</td>
<td>83 M</td>
<td>0.81</td>
<td>67.7/6.23</td>
</tr>
<tr>
<td>Skull</td>
<td>16 M</td>
<td>83 M</td>
<td>0.61</td>
<td>51.7/6.25</td>
</tr>
<tr>
<td>Aneurism</td>
<td>16 M</td>
<td>83 M</td>
<td>2.42</td>
<td>201/5.35</td>
</tr>
</tbody>
</table>

Table 2: Setup and render times.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Setup</th>
<th>Render</th>
<th>% Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel</td>
<td>0.005 s</td>
<td>0.009 s</td>
<td>64.28 %</td>
</tr>
<tr>
<td>Tooth</td>
<td>1.377 s</td>
<td>6.484 s</td>
<td>82.47 %</td>
</tr>
<tr>
<td>Foot</td>
<td>4.556 s</td>
<td>9.074 s</td>
<td>66.57 %</td>
</tr>
<tr>
<td>Skull</td>
<td>4.606 s</td>
<td>8.593 s</td>
<td>65.10 %</td>
</tr>
<tr>
<td>Aneurism</td>
<td>0.199 s</td>
<td>0.210 s</td>
<td>51.34 %</td>
</tr>
</tbody>
</table>
Results

- High tetrahedra/s performance
- No GPU memory used
- More time spent on GPU than CPU

Table 1: Average frames and tetrahedra per second.

<table>
<thead>
<tr>
<th>Data set</th>
<th># Verts</th>
<th># Tet</th>
<th>fps</th>
<th>M tet/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel</td>
<td>262 K</td>
<td>1.2 M</td>
<td>70.78</td>
<td>88.5/5.99</td>
</tr>
<tr>
<td>ToothC</td>
<td>1 M</td>
<td>5 M</td>
<td>1.22</td>
<td>13.1/6.30</td>
</tr>
<tr>
<td>Tooth</td>
<td>10 M</td>
<td>52 M</td>
<td>0.24</td>
<td>12.7/6.61</td>
</tr>
<tr>
<td>Foot</td>
<td>16 M</td>
<td>83 M</td>
<td>0.81</td>
<td>67.7/6.23</td>
</tr>
<tr>
<td>Skull</td>
<td>16 M</td>
<td>83 M</td>
<td>0.61</td>
<td>51.7/6.25</td>
</tr>
<tr>
<td>Aneurism</td>
<td>16 M</td>
<td>83 M</td>
<td>2.42</td>
<td>201/5.35</td>
</tr>
</tbody>
</table>

Table 2: Setup and render times.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Setup</th>
<th>Render</th>
<th>% Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel</td>
<td>0.005 s</td>
<td>0.009 s</td>
<td>64.28 %</td>
</tr>
<tr>
<td>Tooth</td>
<td>1.377 s</td>
<td>6.484 s</td>
<td>82.47 %</td>
</tr>
<tr>
<td>Foot</td>
<td>4.556 s</td>
<td>9.074 s</td>
<td>66.57 %</td>
</tr>
<tr>
<td>Skull</td>
<td>4.606 s</td>
<td>8.593 s</td>
<td>65.10 %</td>
</tr>
<tr>
<td>Aneurism</td>
<td>0.199 s</td>
<td>0.210 s</td>
<td>51.34 %</td>
</tr>
</tbody>
</table>
Conclusions

• High tetrahedra/s performance
  ↓ Lower than 3D Texture

• No GPU memory used
  ↑ Larger data sets than 3D Texture

• More time spent on GPU than CPU
  ↑ closer to the theoretical limit [Roettger and Ertl 2003]
Future Works

• Rendering less primitives
  – Join the 5 tetrahedra (~20 triangles) projected shape in less triangles
  – Project the hexahedron

• Enhance the visualization with a Phong lighting model
GPU-Based Cell Projection for Large Structured Data Sets

Email: andmax@cos.ufrj.br
COPPE/UFRJ: http://www.cos.ufrj.br
LCG: http://www.lcg.ufrj.br

Thank you!