Petascale Molecular Ray Tracing: Accelerating VMD/Tachyon with OptiX

John E. Stone
Theoretical and Computational Biophysics Group
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign

http://www.ks.uiuc.edu/
S4400, GPU Technology Conference
10:00-10:25, Room LL21C, San Jose Convention Center,
San Jose, CA, Thursday March 27, 2014
VMD – “Visual Molecular Dynamics”

- Visualization and analysis of:
  - molecular dynamics simulations
  - particle systems and whole cells
  - cryoEM densities, volumetric data
  - quantum chemistry calculations
  - sequence information
- User extensible w/ scripting and plugins
- http://www.ks.uiuc.edu/Research/vmd/
Goal: A Computational Microscope
Study the molecular machines in living cells

Ribosome: target for antibiotics
Poliovirus
Lighting Comparison

Two lights, no shadows

Two lights, hard shadows, 1 shadow ray per light

Ambient occlusion + two lights, 144 AO rays/hit
“My Lights are Always in the Wrong Place…”

Two lights, harsh shadows, 1 shadow ray per light per hit

Ambient occlusion (~80%) + two lights (~20%), 144 AO rays/hit
Computational Biology’s Insatiable Demand for Processing Power

- Lysozyme
- ApoA1
- ATP Synthase
- STMV
- Ribosome
- HIV capsid

Number of atoms:
- 1986
- 1990
- 1994
- 1998
- 2002
- 2006
- 2010
- 2014

- $10^4$
- $10^5$
- $10^6$
- $10^7$
- $10^8$
Visualization Goals, Challenges

• Increased GPU acceleration for visualization of petascale molecular dynamics trajectories
• Overcome GPU memory capacity limits, enable high quality visualization of >100M atom systems
• Use GPU to accelerate not only interactive-rate visualizations, but also photorealistic ray tracing with artifact-free ambient occlusion lighting, etc.
• Maintain ease-of-use, intimate link to VMD analytical features, atom selection language, etc.
Why Built-In VMD Ray Tracing Engines?

- **No disk I/O** or communication to outboard renderers
- **Eliminate unnecessary data replication** and host-GPU memory transfers
- Directly operate on VMD internal molecular scene, quantized/compressed data formats
- Implement all **curved surface primitives**, volume rendering, texturing, shading features required by VMD
- **Same scripting, analysis, atom selection**, and rendering features are available on all platforms, **graceful CPU fallback**
VMD GPU-Accelerated Ray Tracing Engine: “TachyonL-OptiX”

• Complementary to VMD OpenGL GLSL renderer that uses fast, interactivity-oriented rendering techniques
• Key ray tracing benefits: ambient occlusion lighting, shadows, high quality transparent surfaces, …
  – Subset of Tachyon parallel ray tracing engine in VMD
  – GPU acceleration w/ CUDA+OptiX ameliorates long rendering times associated with advanced lighting and shading algorithms
    • Ambient occlusion generates large shadow test workload
    • Transparent surfaces and transmission rays can increase secondary ray counts by another order of magnitude
  – Adaptation of Tachyon to the GPU required careful avoidance of GPU branch divergence, use of GPU memory layouts, etc.
VMD w/ OpenGL GLSL vs. GPU Ray Tracing

• GPU Ray Tracing:
  – Entire scene resident in GPU on-board memory for speed
  – RT performance is **heavily dependent on BVH** acceleration, particularly for scenes with large secondary ray workloads – shadow rays, ambient occlusion shadow feelers, transmission rays
  – RT **BVH structure regenerated / updated each trajectory timestep**, for some petascale visualizations BVH gen. can take up to ~25 sec!

• OpenGL GLSL:
  – No significant per-frame preprocessing required
  – Minimal persistent GPU memory footprint
  – Implements point sprites, ray cast spheres, pixel-rate lighting, …
TachyonL-Optix GPU Ray Tracing w/ OptiX+CUDA

- OptiX/CUDA kernels can only run for about 2 seconds uninterrupted.
- GPU RT therefore cannot go wild with uninterrupted recursion, internal looping within shading code, or **GPU timeout will occur and kernel will be terminated** by OS/driver.
- Complex ray tracing algorithms broken out into **multi-pass algorithms**:
  - Many GPU kernel launches (up to hundreds in some cases).
  - Intermediate rendering state written to GPU memory at end of each pass.
  - Intermediate rendering state is reloaded at the start of the next pass.
  - **Examples**: state of multiple random number generators, color accumulation buffers, are stored and reloaded in our current implementation.
VMD “QuickSurf” Representation

All-atom HIV capsid simulations w/ up to 64M atoms
chromatophore
from purple bacteria
200 proteins, 3700 cofactors
10 million atoms
GPU Ray Tracing of HIV-1 on Blue Waters

- 64M atom simulation, 1079 movie frames
- **Ambient occlusion lighting**, shadows, transparency, antialiasing, depth cueing, **144 rays/pixel minimum**
- GPU memory capacity hurdles:
  - Surface calc. and ray tracing each use *over 75% of K20X 6GB on-board GPU memory* even with quantized/compressed colors, surface normals, …
  - Evict non-RT GPU data to host prior to ray tracing
  - Eviction was *still required* on a test machine with a **12GB Quadro K6000** GPU – the multi-pass surface algorithm grows the per-pass chunk size to reduce the number of passes
HIV-1 Parallel HD Movie Rendering on Blue Waters Cray XE6/XK7

New “TachyonL-OptiX” on XK7 vs. Tachyon on XE6: K20X GPUs yield **up to eight times** geom+ray tracing speedup

<table>
<thead>
<tr>
<th>Node Type and Count</th>
<th>Script Load Time</th>
<th>State Load Time</th>
<th>Geometry + Ray Tracing</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 XE6 CPUs</td>
<td>7 s</td>
<td>160 s</td>
<td>1,374 s</td>
<td>1,541 s</td>
</tr>
<tr>
<td>512 XE6 CPUs</td>
<td>13 s</td>
<td>211 s</td>
<td>808 s</td>
<td>1,032 s</td>
</tr>
<tr>
<td>64 XK7 Tesla K20X GPUs</td>
<td>2 s</td>
<td>38 s</td>
<td>655 s</td>
<td>695 s</td>
</tr>
<tr>
<td>128 XK7 Tesla K20X GPUs</td>
<td>4 s</td>
<td>74 s</td>
<td>331 s</td>
<td>410 s</td>
</tr>
<tr>
<td>256 XK7 Tesla K20X GPUs</td>
<td>7 s</td>
<td>110 s</td>
<td>171 s</td>
<td>288 s</td>
</tr>
</tbody>
</table>
Future Work

• Improve multi-pass ray casting implementation
• Improve GPU BVH regen speed for time-varying geometry, MD trajectories
• Performance improvements for ambient occlusion sampling strategy
• Interactive RT in VMD
• Continue tuning of GPU-specific RT intersection routines, memory layout
• Add GPU-accelerated movie encoder back-end
Acknowledgements

• Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign

• NCSA Blue Waters Team

• NVIDIA CUDA Center of Excellence, University of Illinois at Urbana-Champaign

• NVIDIA OptiX team – especially James Bigler

• NVIDIA CUDA team

• Funding:
  – NSF OCI 07-25070
  – NSF PRAC “The Computational Microscope”
  – NIH support: 9P41GM104601, 5R01GM098243-02
NIH BTRC for Macromolecular Modeling and Bioinformatics

Beckman Institute
University of Illinois at Urbana-Champaign

1990-2017
GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/


GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/


GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/


GPU Computing Publications
http://www.ks.uiuc.edu/Research/gpu/

- Adapting a message-driven parallel application to GPU-accelerated clusters.

- GPU acceleration of cutoff pair potentials for molecular modeling applications.

