Secrets of Parthenon Renderer

Toshiya Hachisuka

The University of Tokyo
Disclaimer

Not all of my observations are fully validated by scientific experiments, though they are based on my experience.

Take them with a grain of salt!

Many images are removed from the original slides due to copyrights.
15 years ago...
GPUs in 2002 ⇒ 2017

- More complex operations (64 inst. ⇒ 64K inst.)
- Faster computation (30G FLOPS ⇒ 3T FLOPS)
What is “Parthenon Renderer”?

- CPU/GPU combined offline rendering system
- Released in 2002 (= the rise of the GPGPU era)
- Publicly and commercially available back then
What is “Parthenon Renderer”? 

Pentium4 2.7 GHz & Radeon 9700 Pro
What is “Parthenon Renderer”? 

[Image of various 3D rendered objects and scenes]
Why now?

- Examples of how techniques become (non) obsolete
- High-ends in 2002 are low-ends in 2017
- Hopefully useful to predict the future
System Overview
How Parthenon Works

- Photon mapping + Final Gathering
- Mapping computation to rasterization units
- Asynchronous computation with CPU and GPU
How Parthenon Works

- Photon mapping + Final Gathering
- Mapping computation to rasterization units
- Asynchronous computation with CPU and GPU
Photon Mapping
Photon Mapping
Photon Mapping
Photon Mapping
Photon Mapping + Final Gathering

• “Clean” the rough solution
Photon Mapping + Final Gathering
Photon Mapping + Final Gathering
Photon Mapping + Final Gathering
Observations

- Algorithmic complexity
- Computation cost
Observations - Complexity

• Final gathering is a **simple** process
  • Sample rays over the hemisphere

• Photon mapping is a **complex** process
  • Sampling light sources and BRDFs, kNN search
Observations - Cost

- Photon mapping is cheap, but final gathering is not

<table>
<thead>
<tr>
<th>Scene</th>
<th>Number of Rays</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>32000000</td>
</tr>
<tr>
<td>3</td>
<td>64000000</td>
</tr>
<tr>
<td></td>
<td>96000000</td>
</tr>
<tr>
<td></td>
<td>128000000</td>
</tr>
<tr>
<td></td>
<td>160000000</td>
</tr>
</tbody>
</table>

Legend:
- Yellow: Photon Mapping
- Blue: Ray Tracing
- Purple: Final Gathering

Number of Rays
Main Idea

- CPU (in 2002)
  - Good at complex tasks but slow

- GPU (in 2002)
  - Good at simple tasks but fast
Main Idea

• CPU (in 2002)
  • Photon mapping and ray tracing

• GPU (in 2002)
  • Final gathering
Does this make sense today?

- GPU ray tracing is practical today
- Should be able to do everything on GPU today
- Only if you have a good GPU
Solution for Low-end GPUs

- Not everyone has high-end GPUs
- GeForce GTX 580 ≈ 1.5T FLOPS
- GeForce GT 520 ≈ 150G FLOPS
Solution for Low-end GPUs

- Not everyone has high-end GPUs
- Radeon 9800 XT ≈ 50G FLOPS (in 2002)
- GeForce GT 520 ≈ 150G FLOPS
Solution for Low-end GPUs

• Some rough estimates

• 100M rays/sec on GPU

  10M rays/sec on a single CPU core

• 1.5T FLOPS (10x faster than a single CPU core)

  150G FLOPS (as fast as a single CPU core)
Solution for Low-end GPUs

Low-end GPUs in 2017
≈
High-end GPUs in 2002
≈
Single core of CPUs in 2017
How Parthenon Works

- Photon mapping + Final Gathering
- Mapping computation to rasterization units
- Asynchronous computation with CPU and GPU
Precomputation

- Store the result of photon mapping into a mesh
- Similar to light maps computation
- Directional info encoded by SH coefficients
Grouping by Position
Grouping by Position
Grouping by Position
Grouping by Position
Grouping by Position
Grouping by Position

- Map the FG process on rendering cube maps
Grouping by Position

- Map the FG process on rendering cube maps
Grouping by Position

- Too many rasterizations of the scene
- Number of final gathering points
  = Number of pixels
  = Number of rasterization passes
  = $O(1M)$
- Recent research use this with many approximations
Grouping by Direction
Grouping by Direction
Grouping by Direction

• Ray bundle [Szirmay-Kalos and Purgathofer 1998]
• Can be mapped into a parallel projection
Parallel Projection
Parallel Projection
Parallel Projection
Grouping by Direction

- Significantly fewer rasterizations of the scene
- Number of final gathering directions 
  = Number of final gathering samples
  = Number of rasterization passes
  = $O(100) \ll O(1M)$

- More details in GPU Gems 2
Performance in 2002

Grouping by Position

Grouping by Direction

CPU ray tracing

GPU ray tracing

Number of Samples per second [K samples / sec]
Performance in 2017

Grouping by Position

Grouping by Direction

CPU ray tracing

GPU ray tracing

Number of Samples per second [M samples / sec]
Performance in 2017

- Grouping by Position
  - With Approximations
- Grouping by Direction
- CPU ray tracing
  - Optimized
- GPU ray tracing
  - Static Scenes

Number of Samples per second [M samples/sec]
Does this make sense today?

- Grouping by Position
- Grouping by Direction
- CPU ray tracing
- GPU ray tracing

Number of Samples per second [M samples / sec]
How Parthenon Works

- Photon mapping + Final Gathering
- Mapping computation to rasterization units
- Asynchronous computation with CPU and GPU
Main Idea

• CPU
  • Photon mapping and ray tracing

• GPU
  • Final gathering
Main Idea

- CPU
  - Photon mapping and ray tracing
    - Do both at the same time
- GPU
  - Final gathering
Asynchronous Computation

CPU

GPU
Asynchronous Computation

CPU

Photon Mapping

GPU
Asynchronous Computation

CPU

Photon Mapping  Ray tracing

GPU
Asynchronous Computation

Photon Mapping | Ray tracing

CPU

GPU

FG
Asynchronous Computation

CPU

Photon Mapping  Ray tracing
FG  FG

GPU
Asynchronous Computation
Asynchronous Computation

CPU

Photon Mapping
Ray tracing
Sync
FG FG FG FG FG FG

GPU
Asynchronous Computation

CPU

Photon Mapping

Ray tracing

Sync

Ray tracing

GPU
Asynchronous Computation

CPU

Photon Mapping  Ray tracing  Sync  Ray tracing  Sync
FG  FG  FG  FG  FG  FG  FG  FG  FG  FG  FG

GPU
Asynchronous Computation

CPU

Photon Mapping  Ray tracing  Sync  Ray tracing  Sync  Ray tracing
FG  FG  FG  FG  FG  FG  FG  FG  FG  FG  FG  FG
Does this make sense today?

- Final gathering typically needs far more samples
- 40 FG samples ≈ 1 RT sample (in 2002)
Does this make sense today?

- Final gathering typically needs far more samples
- 40 FG samples $\approx 1$ RT sample (in 2002)
- 400 FG samples $\approx 1$ RT sample (in 2017)
Some Other Details

- Utilizes shadow mapping units
- Direct illumination and 1st photon trace
- Fake caustics
- Über shader (i.e., single shader handles all materials)
- No choice in 2002
- Still compromised choice today for some systems
Closing Remarks
In retrospect ...

- Testing for many GPUs was painful
- Parthenon runs on both Radeon and GeForce
- Only solution for testing is to actually run
- Checking specs do not help in the end, you know

- Still true today
- Worse in my opinion since GPUs are everywhere
In retrospect ...

- Heterogeneous computation was painful
- Power balance of CPU and GPU has changed a lot
- Managing duplicated codes for CPU and GPU
- Maybe still true today
- OpenCL can be a solution if it works as designed
In retrospect ...

- Going to the right direction of GPU rendering
  - but too early - users were not ready
  - and too immature - technology was not there

- Still somewhat true today, but much better
  - People recognized well what GPUs can do
  - Virtually anything on CPUs can be done on GPUs
Summary

- Parthenon Renderer
- One of the first GPU rendering systems
- Many choices are out-of-date, but not all of them
- Some remarks
  - Heterogeneous computing might not be a good idea
  - Supporting different GPUs can still be painful
  - Old techniques for high-ends can be useful and practical for low-ends now