GPU Ray Tracing
at the Desktop and in the Cloud

Phillip Miller, NVIDIA
Ludwig von Reiche, mental images
Ray Tracing – has always had an appeal
Ray Tracing Prediction

The future of interactive graphics is ray tracing....

And it *always* will be :) 🕒!

GPUs are making that “future” look *much* closer...
Realism versus Interaction – a Constant

- For all visual industries, realism is most often the goal.

- In Film FX – realism typically more important than time.
  - Innovation decreases time.
  - Increasing realism most often consume time gains.

- In Games and Design – time more important than realism.
  - Realism increases as real-time is maintained.
  - Design requires at least 5 to 10 FPS.
  - Games requires 30 or 60 FPS (now 120 FPS in stereo).
Realism/Time Speed/Node: Baseline

- Realism
  - 60 frames per second
  - 10 seconds
  - 5 seconds
  - 2 seconds
  - 1 second
  - 15 seconds
  - 30 seconds
  - 1 minute
  - 15 minutes
  - 30 minutes
  - 1 hour
  - 3 hours
  - 6 hours
  - 12 hours
  - 1 day
  - +1 day

- All CPU

- Film FX

- Design
- Games

NVIDIA’s permission required before redistributing © 2010
Realism/Time Speed/Node: GPU & Shaders

---

Ray Tracing “plateau”

---

Games

---

All CPU

---

Film FX

---

NVIDIA’s permission required before redistributing © 2010
Real-Time State of the Art
Real-Time State of the Art
Real-Time State of the Art
What’s behind this level realism

- A lot of talent (and time)
- using great tools
- powered by top end GPUs
- with custom shaders (CgFX, HLSL, GLSL)
- managed by a real-time scene graph
No Self Reflection
**Today**
- Limited to Raster Capabilities
- Result is tied to the scene
- High training & cost
- Intense art time

**Tomorrow**
- Physically correct
- Result works anywhere
- Far less training and cost
- Intense computations
Realism/Time Speed/Node:

Ray Tracing “plateau”

Realism

Game

Design

Film FX

GPU Shaders

NVIDIA’s permission required before redistributing

© 2010
**Interactive Ray Tracing Leadership**

- **SIGGRAPH 2008**
  - 30 FPS proof of concept, on shipping hardware
  - Later published papers on approaches
- **SIGGRAPH 2009**
  - Debuted the OptiX engine and the iray renderer
  - OptiX, iray, RealityServer 3 released 3 months later
- **Early 2010**
  - Design Garage demo in 5 weeks w/ OptiX & SceniX
- **SIGGRAPH 2010**
  - Numerous GPU rendering solutions on display
  - iray in Bunkspeed **Shot**, OptiX a v2
- **Now**
  - iray in Autodesk **3ds Max 2011**, and DS **Catia v6**
  - OptiX in **Lightworks** and numerous private applications
  - Cloud rendering with iray ready to deploy
Public Views on GPU Ray Tracing

3 years ago — A GPU *can’t* ray trace

2 years ago — NVIDIA can, but we *can’t*

1 year ago — Now everyone *can*

This year — Now many *are*

Next year — You can do it *anywhere*
NVIDIA Design Garage Demo

- Photorealistic car configurator in the hands of millions of consumers

- Uses pure GPU ray tracing
  - Est. 40-50X faster vs. a CPU core
  - 3-4X faster on GF100 than on GT200
  - Linear scaling over GPUs & CUDA Cores

- Built on SceniX with OptiX shaders – similar to other apps in development

- Rendering development speed – 5 weeks
# GPU Computing Overview

## GPU Computing Applications

| CUDA C/C++ | OpenCL | Direct Compute | Fortran | Python, Java, .NET, ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Over 90,000 developers</td>
<td>1st GPU demo</td>
<td>Microsoft API for GPU Computing</td>
<td>PGI Accelerator</td>
<td>PyCUDA</td>
</tr>
<tr>
<td>Running in Production since 2008</td>
<td>Shipped 1st OpenCL Conformant Driver</td>
<td>Supports all CUDA-Architecture GPUs (DX10 and DX11)</td>
<td>PGI CUDA Fortran</td>
<td>jCUDA</td>
</tr>
<tr>
<td>SDK + Libs + Visual Profiler and Debugger</td>
<td>Public Availability (Since April)</td>
<td>NOAA Fortran bindings</td>
<td>FLAGON</td>
<td>CUDA.NET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OpenCL.NET</td>
</tr>
</tbody>
</table>

## NVIDIA GPU

with the CUDA Parallel Computing Architecture

## Broad Adoption

- Over 250,000,000 installed CUDA-Architecture GPUs
- Over 100,000 GPU Computing Developers
- Windows, Linux and MacOS Platforms supported
- GPU Computing spans HPC to Consumer
- 250+ Universities teaching GPU Computing on the CUDA Architecture

© 2010
Many Programming Approaches in Use

- iray: CUDA C, C Runtime
- finalRender: CUDA C, C Runtime
- Furry Ball: CUDA C, C Runtime
- Arion: CUDA C, driver API
- Octane: CUDA C, driver API
- V-ray RT GPU: OpenCL
- OptiX: CUDA C, driver API with PTX stitching
- Lightworks, etc.: CUDA C, OptiX API
Solutions Vary in their GPU Exploitation

- Speed-ups vary, but a top end Fermi GPU will typically ray trace 6 to 15 times faster than on a quad-core CPU.

- A GPGPU programming challenge is to keep the GPU “busy”.
  - Gains on complex tasks often greater than for simple ones.
  - Particularly evident with multiple GPUs, where data transfers impact simple tasks more.
  - Can mean the technique needs to be rethought in how it’s scheduling work for the GPU.
  - OptiX 2.1 example – first tuned for simple, now tuned for complex, with a 30-80% speed increase.
Similarities for today’s GPU Ray Tracing

- Performance tends to scale linearly with GPU cores and core clock for a given GPU generation.
- Gains between GPU generations will vary per solution.
- Most scale well across system GPUs, with no need for SLI.
- Most solutions can “distribute” rendering, but only some support “cluster” rendering.
- Scaling efficiency will vary per solution and/or technique.
- Entire scene must fit onto the GPU’s memory* – geometry, textures, and acceleration structures.

*not a permanent situation
# GPU Computing Application Development

<table>
<thead>
<tr>
<th>Your GPU Ray Tracing Application</th>
<th>OEM Renderers (iray)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Acceleration Engines</td>
<td></td>
</tr>
<tr>
<td>e.g., OptiX ray tracing engine</td>
<td></td>
</tr>
<tr>
<td>Foundation Libraries</td>
<td></td>
</tr>
<tr>
<td>Low-level Functional Libraries</td>
<td></td>
</tr>
<tr>
<td>Development Environment</td>
<td></td>
</tr>
<tr>
<td>Languages, Device APIs, Compilers, Debuggers, Profilers, etc.</td>
<td></td>
</tr>
<tr>
<td>CUDA Architecture</td>
<td></td>
</tr>
</tbody>
</table>

© 2010
Accelerating Application Development

App Example: Auto Styling

1. Establish the Scene
   = SceniX

2. Maximize interactive quality
   + CgFX + OptiX

3. Maximize production quality
   + iray

App Example: Ray Tracing Task

1. Prepare your Scene
   = your art production path

2. Identify a ray tracing bottleneck
   + OptiX

3. Process the task and merge
   e.g., ambient occlusion
   e.g., light maps
What Ray Tracing techniques are possible?

Answer: What ever you’d like.

Unbiased rendering is currently a popular approach in commercial renderers but *by no means the only approach*

For example:
NVIDIA® OptiX™
the world’s first interactive ray tracing engine

A programmable ray tracing pipeline for accelerating interactive ray tracing applications – from complete renderers, to functions, to tasks (collision, acoustics, signal processing, radiation reflectance, etc.)

- You write the ray tracing techniques
  - OptiX makes them fast
OptiX
for faster and easier ray tracing development

Faster development
• Ray calculations are abstracted to single rays
• State of the art acceleration structures & traversers
• Programmable shaders, surfaces and cameras
• Tight coupling with OpenGL & Direct3D
• GPU issues like load balancing, scheduling, parallelism are all handled.

Flexible use
• Ray payloads can be custom
• Custom intersection goes well beyond triangles
• Not tied to a rendering language, shader model or camera model

Greatly lowers the barrier to entry
• For creating high performance ray tracing
• Developers often saving 50-75% on base effort – with much higher performance results

© 2010
Hybrid – Increasing Interactive Realism

- Glossy Reflections
- Soft Shadows
- Ambient Occlusion, etc...

• Combined as a Scene Effect with OGL or D3D

© 2010 NVIDIA's permission required before redistributing
Example: Works Zebra workflow

back plate
dx car
raytrace
Example: Works Zebra using the GPU

“on screen, I can see the difference between real-time and offline, but not between OptiX and offline”
Manager, Toyota Marketing Japan
Interactive Ray Tracing: Lightworks
Interactive Ray Tracing: Bunkspeed Shot™
3ds Max Rendering Revolution Contest
3ds Max Rendering Revolution Contest
3ds Max Rendering Revolution Contest
3ds Max Rendering Revolution Contest
*including iray
iray® from mental images

*bringing photoreal ray tracing to a product near you*

A physically correct and interactive global illumination renderer.

The perfect choice for those relating to the real-world (designers, consumers,...)

- CUDA-based w/ CPU fallback
  (massive delta – not interactive)
- Scalable across GPUs & nodes (DICE)
- Inclusion Options:
  - w/ current mental ray and RealityServer
  - Integrator Edition (for those w/o mental ray)
  - Option for SceniX (later this year)
  - Coming to numerous products in 2010
iray – in action
GPU Technology Conference (GTC 2010)

September 20-23, 2010  San Jose, CA

Now taking Submissions:

iray and OptiX

*together addressing the spectrum of rendering needs*

- With iray, you add or replace a renderer.  
  *Ideal when you want a ready-to-integrate/use photorealistic solution*

- With OptiX, you accelerate or build a renderer.  
  *OptiX is ideal when you want to accelerate hybrid & custom solutions*

**Ongoing Focus:**

- **iray** – quality, complete solution, perf
- **OptiX** – interaction, flexibility/generality, perf
- **NVIDIA** - assisting GPU ray tracing development wherever it’s desired