RTXGI: SCALABLE RAY TRACED GLOBAL ILLUMINATION IN REAL TIME

Adam Marrs, 3/23/2020
Real-Time Ray Tracing Applications

DirectX® Raytracing API

NVIDIA RTX GPUs
REINVENTING REAL-TIME
GLOBAL ILLUMINATION, IN REAL TIME

Light maps
[Quake97, Mitchell06]

Virtual point lights
[Keller97, Kaplanyan10, Ding14, Xu16, Luksch19]

Reflective shadow maps
[Kaplanyan10, Billeter12, Ding14, Malmros17, Xu16]

Light propagation volumes
[Kaplanyan09, Kaplanyan10, Boeckmann19]

Sparse voxel cone tracing
[Crassin11, McLaren16]

Denoised ray tracing
[Mara17, Schied17, Metro19, Archard19]

Irradiance probes/voxels
[Greger98, Ramamoorthi01, Tatarchuk05, Gilabert12, McGuire17, Majercik19]
RTX GLOBAL ILLUMINATION (RTXGI) SDK

High Level Goals

- Flexibility
  - Customizable, so you can tailor it to your specific needs

- Scalability
  - Effective solutions for a wide range of target hardware

- Convenience
  - Implement and optimize global lighting algorithms, so you don’t have to
RTX GLOBAL ILLUMINATION (RTXGI) SDK
Scalable Ray Traced Global Illumination in Real Time

- Full C++ and HLSL source code
- Sample application with full C++ and HLSL source
- Runs on *all* DXR enabled GPUs: NVIDIA Turing, NVIDIA Pascal, other vendors
- Scalable quality for GTX 1060 6GB through RTX 2080 Ti
- Available now, for free (on GitHub)

http://developer.nvidia.com/rtxgi
RTX GLOBAL ILLUMINATION (RTXGI) SDK
Dynamic Diffuse Global Illumination (DDGI)

- Based on irradiance probes
  - A common solution already used in many game engines today
- Fixes light and shadow leaking issues caused by lack of visibility information

Unreal Engine
Dunia (Far Cry engine)
Unity

LIGHT & SHADOW LEAKS

Problem: Geo Within Voxels

Light Leaking Is A Problem

Visibility Is A Problem
- Where the probe doesn’t see
- Looks like shadows

Advances in Real-Time Rendering course, SIGGRAPH 2016
Hooker 2016
**Scene Authoring**

Place volumes of probes in the 3D world.

**Ray Trace & Shade**

Trace and shade rays cast from active probes in relevant volumes.

Use previous probe data during shading for infinite bounce GI.

**Update Probes**

Blend ray traced results into probes, storing irradiance and the distance to geometry.

**Render Diffuse GI**

Compute indirect lighting and visibility from ray traced probes. No leaks.
DYNAMIC DIFFUSE GLOBAL ILLUMINATION

Algorithm

Scene Authoring
Place volumes of probes in the 3D world.

Ray Trace & Shade
Trace and shade rays cast from active probes in relevant volumes.
Use previous probe data during shading for infinite bounce GI.

Update Probes
Blend ray traced results into probes, storing irradiance and the distance to geometry.

Render Diffuse GI
Compute indirect lighting and visibility from ray traced probes. No leaks.
DYNAMIC DIFFUSE GLOBAL ILLUMINATION
Algorithm

Scene Authoring
Place volumes of probes in the 3D world.

Ray Trace & Shade
Trace and shade rays cast from active probes in relevant volumes.
Use previous probe data during shading for infinite bounce GI.

Update Probes
Blend ray traced results into probes, storing irradiance and the distance to geometry.

Render Diffuse GI
Compute indirect lighting and visibility from ray traced probes. No leaks.
DYNAMIC DIFFUSE GLOBAL ILLUMINATION

Algorithm

Scene Authoring

Place volumes of probes in the 3D world.

Ray Trace & Shade

Trace and shade rays cast from active probes in relevant volumes.

Use previous probe data during shading for infinite bounce GI.

Update Probes

Blend ray traced results into probes, storing irradiance and the distance to geometry.

Render Diffuse GI

Compute indirect lighting and visibility from ray traced probes. No leaks.
DYNAMIC DIFFUSE GLOBAL ILLUMINATION

Algorithm

**Scene Authoring**
Place volumes of probes in the 3D world.

**Ray Trace & Shade**
Trace and shade rays cast from active probes in relevant volumes.
Use previous probe data during shading for infinite bounce GI.

**Update Probes**
Blend ray traced results into probes, storing irradiance and the distance to geometry.

**Render Diffuse GI**
Compute indirect lighting and visibility from ray traced probes. No leaks.
DYNAMIC DIFFUSE GLOBAL ILLUMINATION

Algorithm

Scene Authoring
Place volumes of probes in the 3D world.

Ray Trace & Shade
Trace and shade rays cast from active probes in relevant volumes.
Use previous probe data during shading for infinite bounce GI.

Update Probes
Blend ray traced results into probes, storing irradiance and the distance to geometry.

Render Diffuse GI
Compute indirect lighting and visibility from ray traced probes. No leaks.

Independent of screen resolution and framerate
DDGI FEATURES & BENEFITS

Summary

- Infinite bounce indirect lighting, for forward or deferred renderers
- Ray traced quality, no denoising necessary
- Accelerated content creation
  - No baking, no leaks
  - No UV parameterization or probe blockers
  - In-game and in-editor lighting updates for fast iteration
RTXGI SDK: DDGI
Features and Improvements

- Flexible resource management
- Works with any material and lighting model
- Perceptual encoding to speed up convergence when large lighting changes occur
- Flexibility for programmers and artists to control performance and lighting quality
- Performance: fast probe updates using GPU shared memory
- Probe Relocation (early access)
- Probe State Classification (early access)
RTXGI SDK: DDGI
Flexible Resource Management

- Engines and renderers handle resources differently to solve different problems
- Providing a choice in how resources are managed

**SDK Managed Resources**
- SDK internally allocates/tracks/deallocates necessary GPU resources
- Can’t manipulate resources directly, but black box design is easier to use

**Application Managed Resources**
- Application allocates/tracks/deallocates GPU resources, passes pointers into SDK
- Better flexibility for advanced applications, but greater responsibility
RTXGI SDK: DDGI
Any Material or Lighting Model

- SDK does not force specific resource bindings or material properties
- Probe ray tracing step is owned and implemented by the application
- SDK provides utility functions, such as `DDGIGetProbeRayDirection(...)`, to compute unique, low discrepancy, spherically distributed directions on the unit sphere
- Write ray traced results to the SDK’s radiance texture
RTXGI SDK: DDGI

Perceptual Encoding, Hysteresis, and Artist Controls

- **Exponential weighting** when storing irradiance to improve light-to-dark convergence
  - Moves irradiance into a non-linear space that more closely matches human perception

- **Hysteresis** settings give you control over convergence time

- **Probe Change and Brightness Thresholds** give programmers and artists control over convergence speed, lighting quality, and performance tradeoffs
RTXGI SDK: DDGI
Hypothetical Frame

- New passes for DDGI are shown in green
- In practice, DDGI Lighting can be combined with an existing lighting pass
- Timeline is for illustration purposes, does not represent performance (that’s next)
## RTXGI SDK: DDGI

### Performance

*Two Rooms, GPU time in milliseconds*

**Stress Test Config** | 16,384 Probes | 144 Rays Per Probe | 1920x1080

<table>
<thead>
<tr>
<th></th>
<th>Probe RT</th>
<th>Probe Update</th>
<th>Lighting</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTX 2080 Ti</td>
<td>1.05</td>
<td>1.22</td>
<td>0.44</td>
<td>2.71</td>
</tr>
<tr>
<td>RTX 2060</td>
<td>2.34</td>
<td>2.81</td>
<td>0.93</td>
<td>6.08</td>
</tr>
<tr>
<td>GTX 1080 Ti</td>
<td>8.53</td>
<td>2.11</td>
<td>1.00</td>
<td>11.64</td>
</tr>
</tbody>
</table>
## RTXGI SDK: DDGI
### Performance

**Two Rooms**, GPU time in milliseconds

<table>
<thead>
<tr>
<th>Stress Test Config</th>
<th>16,384 Probes</th>
<th>144 Rays Per Probe</th>
<th>1920x1080</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Probe RT</th>
<th>Probe Update</th>
<th>Lighting</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTX 2080 Ti</td>
<td>1.05</td>
<td>1.22</td>
<td>0.44</td>
<td>2.71</td>
</tr>
<tr>
<td>RTX 2060</td>
<td>2.34</td>
<td>2.81</td>
<td>0.93</td>
<td>6.08</td>
</tr>
<tr>
<td>GTX 1080 Ti</td>
<td>8.53</td>
<td>2.11</td>
<td>1.00</td>
<td>11.64</td>
</tr>
</tbody>
</table>
## RTXGI SDK: DDGI

### Performance

*Two Rooms, GPU time in milliseconds*

**Stress Test Config | 16,384 Probes | 144 Rays Per Probe | 1920x1080**

<table>
<thead>
<tr>
<th></th>
<th>Probe RT</th>
<th>Probe Update</th>
<th>Lighting</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RTX 2080 Ti</strong></td>
<td>1.05</td>
<td>1.22</td>
<td>0.44</td>
<td>2.71</td>
</tr>
<tr>
<td><strong>RTX 2060</strong></td>
<td>2.34</td>
<td>2.81</td>
<td>0.93</td>
<td>6.08</td>
</tr>
<tr>
<td><strong>GTX 1080 Ti</strong></td>
<td>8.53</td>
<td>2.11</td>
<td>1.00</td>
<td>11.64</td>
</tr>
</tbody>
</table>
## RTXGI: SDK DDGI

Fast Probe Updates

**Two Rooms**, GPU time in milliseconds
Stress Test Config | 16,384 Probes | 144 Rays Per Probe

<table>
<thead>
<tr>
<th></th>
<th>RTX 2080 Ti</th>
<th>RTX 2060</th>
<th>GTX 1080 Ti</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Implementation</td>
<td>3.57</td>
<td>7.83</td>
<td>5.32</td>
</tr>
<tr>
<td>RTXGI SDK Implementation with Shared Memory Optimizations</td>
<td>1.22</td>
<td>2.81</td>
<td>2.11</td>
</tr>
<tr>
<td><strong>Speedup</strong></td>
<td><strong>2.92x</strong></td>
<td><strong>2.79x</strong></td>
<td><strong>2.52x</strong></td>
</tr>
</tbody>
</table>
## RTXGI SDK: DDGI

### Performance

<table>
<thead>
<tr>
<th>Two Rooms, GPU time in milliseconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stress Test Config</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Probe RT</th>
<th>Probe Update</th>
<th>Lighting</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTX 2080 Ti</td>
<td>1.05</td>
<td>1.22</td>
<td>0.44</td>
<td>2.71</td>
</tr>
<tr>
<td>RTX 2060</td>
<td>2.34</td>
<td>2.81</td>
<td>0.93</td>
<td>6.08</td>
</tr>
<tr>
<td>GTX 1080 Ti</td>
<td>8.53</td>
<td>2.11</td>
<td>1.00</td>
<td>11.64</td>
</tr>
</tbody>
</table>
## RTXGI SDK: DDGI

### Performance

**Two Rooms**, GPU time in milliseconds

**Stress Test Config** | 16,384 Probes | 144 Rays Per Probe | 1920x1080

<table>
<thead>
<tr>
<th></th>
<th>Probe RT</th>
<th>Probe Update</th>
<th>Lighting</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTX 2080 Ti</td>
<td>1.05</td>
<td>1.22</td>
<td>0.44</td>
<td>2.71</td>
</tr>
<tr>
<td>RTX 2060</td>
<td>2.34</td>
<td>2.81</td>
<td>0.93</td>
<td>6.08</td>
</tr>
<tr>
<td>GTX 1080 Ti</td>
<td>8.53</td>
<td>2.11</td>
<td>1.00</td>
<td>11.64</td>
</tr>
</tbody>
</table>
RTXGI SDK: DDGI
Probe Relocation (early access)

- Any regular grid of probes has a hard time correctly handling all scenarios
- **Probe Relocation**
  - Maintains world-space offsets for every probe
  - Uses the ray tracing results to determine proximity of back facing geometry
  - Attempts to move probes to more effective locations based on surrounding geometry
- Sample application includes a linear descent optimizer that relocates probes over several iterations (frames)
- **Early access feature**, expect it to be updated and improved in future releases
RTXGI SDK: DDGI
Probe Classification (early access)

- Not all probes in a scene contribute to the final lighting
  - For example, probe stuck in walls or too far outside the scene to be useful
  - These probes don’t need to spend time ray tracing or updating textures and can be disabled

- Probe Classification
  - Maintains a state value (active, inactive) for every probe
  - Marks probes as active or inactive based on the results of ray tracing

- As much as 30% to 80% of the probes can be disabled in many scenarios

- Early access feature, expect it to be updated and improved in future releases
Download the RTXGI SDK today!
http://developer.nvidia.com/rtxgi

Request Access
Go to the link above and click “Get Started”. You’ll need to fill out a short survey to request access.

Clone the GitHub Repo
After filling out the survey and receiving an approval email, you can access the full source distribution on GitHub.

Dive in with the Sample Application
A working sample application is included as an example integration that demonstrates the SDK’s functionality.
THANK YOU

Alexander Majercik
Josef Spjut
Morgan McGuire
Alan Wolfe
Ben Boudaoud

Kelsey Blanton
Alex Hyder
Ethan Einhorn
John Spitzer
QUESTIONS?

@acmarrs

Issues
REFERENCES

[Dachsbocher05] Dachsbocher and Stamminger, Reflective Shadow Maps, I3D, 2005
[Ding14] Ding, In-Game and Cinematic Lighting of The Last of Us, GDC Presentation, 2014
[Luksch19] Luksch et al., Incrementally baked Global Illumination, I3D 2019
[Malmros17] Malmros, Gears of War 4: custom high-end graphics features and performance techniques, SIGGRAPH Talk, 2017
[Metro19] Battaglia, Interview with Ben Archard on Metro: Exodus, Eurogamer, February 17, 2019
[Quake97] ID Software, Quake II, 1997
[Ramamoorthi01] Ramamoorthi and Hanrahan, An Efficient Representation for Irradiance Environment Maps, SIGGRAPH, 2001
[Tatarchuk05] Tatarchuk, Irradiance Volumes for Games, GDC 2005
[Xu16] Xu, Temporal Antialiasing In Uncharted 4, SIGGRAPH Advances in Real-Time Rendering Course, 2016