Ray Tracing in Games with NVIDIA RTX

Ignacio Llamas, Sr. Manager of Real-time Rendering SW
Edward Liu, Sr. Real-time Rendering Engineer
NVIDIA

Booth #223 - South Hall
www.nvidia.com/GDC

Agenda

Introduction to Ray Tracing for Games

RTX, DXR and GameWorks

Real-time Ray Tracing and Denoising

Path Tracing and Light Baking

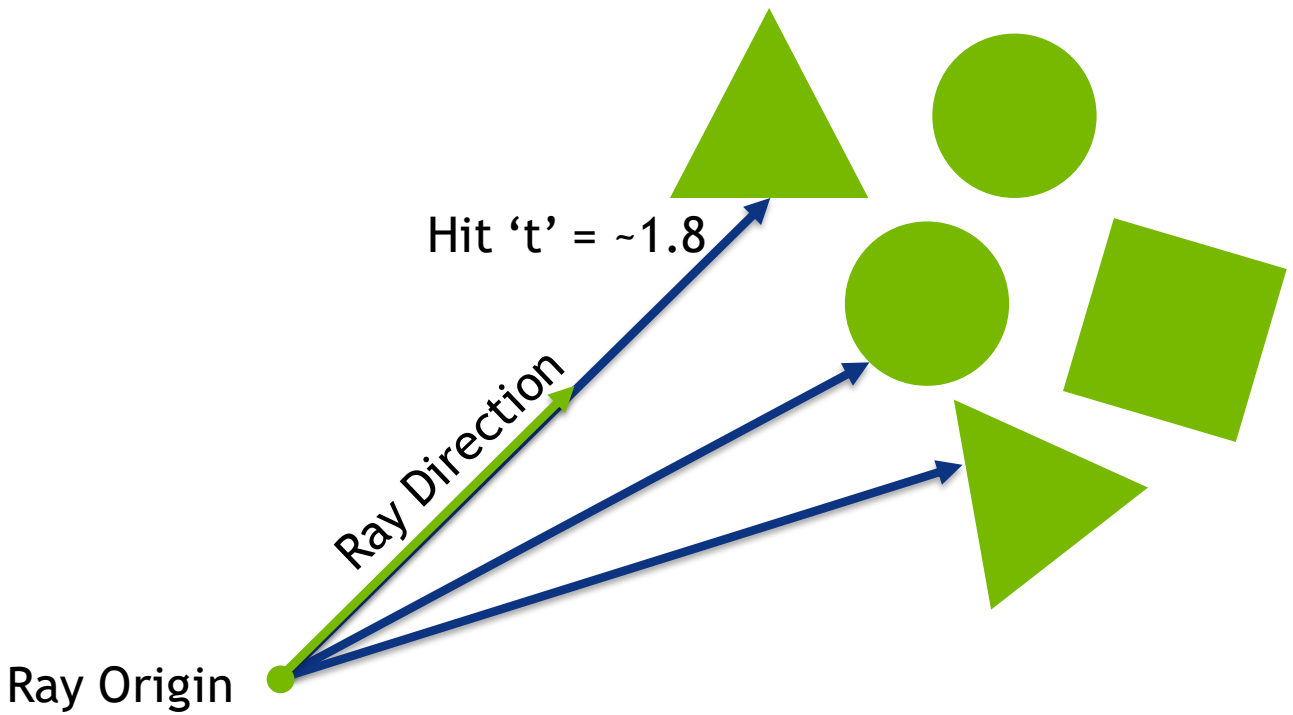
Integrating Ray Tracing into your engine

Introduction to Ray Tracing for Games

Ray Tracing 101

What is Ray Tracing

- Ray Tracing: a primitive to query the intersections of rays against some geometry



Ray Tracing 101

Applications in Games

- Higher quality in-game rendering
  - Reflections
  - Ambient Occlusion
  - Shadows
- Content Creation Workflows
  - Light baking
  - Cinematic rendering
  - Path traced reference
- Non-rendering applications
  - Audio simulation in VR (VRWorks Audio)
  - Physics / Collision detection
  - AI

Ray Tracing 101

Applications in Games

- Higher quality in-game rendering
  - Reflections
  - Ambient Occlusion
  - Shadows
- Content Creation Workflows
  - Light baking
  - Cinematic rendering
  - Path traced reference
- Non-rendering applications
  - Audio simulation in VR (VRWorks Audio)
  - Physics / Collision detection
  - AI

Ray Tracing 101

Rendering

- Commonly used to solve the **rendering equation** with Monte Carlo sampling

$$L(\omega_o) = L_e(\omega_o) + \int_{\delta} L(\omega_i) f(\omega_o, \omega_i) |\omega_i \cdot n| d\omega_i$$

- Natural solution for Shadows, AO, Reflections, Light Baking, Film Rendering

Ray Tracing 101

Ray Tracing in Rendering Today

- Similar primitives used today in games in real-time in various forms:
  - Screen space ray tracing, distance field ray tracing, voxel cone tracing
- In this talk: **ray tracing against full scene geometry** (mostly triangles)
- Rendering with Ray Tracing usually requires many samples for high quality results
  - From a few hundred to a few thousand
  - Depending on the complexity of the integrand (scene)

Ray Traced Shadows

Ray Traced Ambient Occlusion

Ray Traced Reflections

Ray Tracing 101

- 100s of samples per pixel are not practical in real time
- Small number of rays per pixel possible on current HW

- **Is that useful enough?**

Ray Tracing 101

- 100s of samples per pixel are not practical in real time
- Small number of rays per pixel possible on current HW

- Is that useful enough?

- Previous images generated with 1-2 samples per pixel + real-time denoising
- **Yes: high quality results possible with few samples per pixel + denoising**

RTX, DXR and GameWorks

NVIDIA RTX & DirectX Raytracing (DXR)

Context

- NVIDIA® RTX™: ray-tracing technology for Volta GPUs
- The result of a decade of GPU ray tracing R&D at NVIDIA
- Exposed via Microsoft's new DirectX Raytracing API (DXR) for DirectX 12
- New pipeline: the **Ray Tracing Pipeline**

RTX & DXR

Benefits

- Powerful and flexible programming model, similar to NVIDIA OptiX
- Easy to write efficient ray tracing algorithms
- DXR makes it easy to integrate into engines that already use DirectX 12
- DXR provides IHV-agnostic abstraction
- RTX provides an efficient implementation on NVIDIA Volta GPUs
- RTX on Volta delivers performance needed for real-time ray tracing
- *For more DXR details see Matt Sandy's talk at 3:30 pm, http://aka.ms/DXR and NVIDIA blog*

GameWorks Ray Tracing

More context

- GameWorks: tools, simulation and rendering technology for developers
- Announcing **GameWorks Ray Tracing Denoising** modules
- Ray Traced Shadows, Ambient Occlusion and Reflections Denoising
- Available soon

Related Work at GDC

Be sure to check these out too

- Advanced Graphics Day (Monday): Microsoft, Futuremark and Remedy
- Epic-NVIDIA-ILMxLAB Reflections demo earlier today
- Additional talks later today in this room: NV Research, EA SEED, EA Frostbite
- Talk by Microsoft + EA about future of DirectX (DXR)

This Talk

What to expect

- A showcase of possibilities enabled by real-time ray tracing with RTX
- Details about NVIDIA's new GameWorks Ray Tracing Denoising modules
- Our goal: inspire you to start experimenting with real-time ray tracing

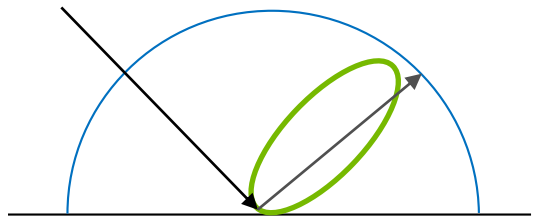
Real-Time Ray Tracing and Denoising

Source of Noise in Ray Tracing

- The rendering equation is solved with Monte Carlo sampling

$$L(\omega_o) = \int_{\delta} L(\omega_i) f(\omega_o, \omega_i) |\omega_i \cdot n| d\omega_i = \sum_{i=0}^{n} L(\omega_i) f(\omega_o, \omega_i) |\omega_i \cdot n| / p(\omega_i)$$

- Every term in the estimator is a complicated function over the hemisphere
  - Incoming radiance, visibility, BRDF, and sampling Pdf
- Insufficient sampling leads to high variance in the estimator



Previous Denoising Work

- Very active area of research
  - Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination [Schied, et al. 17]
  - Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder [CHAITANYA, el al. 17]
  - Multiple Axis-Aligned Filters for Rendering of Combined Distribution Effects [Wu, et al. 17]
  - Kernel-predicting Convolutional Networks for Denoising Monte Carlo Renderings [Bako, et al. 17]
  - … And many others
- Also products like NVIDIA OptiX 5.0 AI Denoiser

Previous Denoising Work

- Input often contains 10s or hundreds of samples
  - Or rely on temporal reprojection (ghosting issue without correct motion vectors)
- Denoising cost from ~100ms to minutes
- Noisy primary visibility samples (DoF, Motion blur)
- Results often really close to ground truth, measured in rMSE

Real-Time Denoising Requirement

- Input image with extremely low sample count
  - We target 1 sample per pixel
- Ghosting free, temporally stable
- Denoising budget of ~1 ms for 1080p target resolution on gaming class GPU
- Perceptually close to ground truth, able to preserve:
  - physically correct contact hardening
  - Elongation and anisotropy
- Smooth G-Buffer (i.e.: no DOF, or motion blur, as common in film rendering)

Denoiser Design Choices

Many options

- Image space vs. Lightmap space vs. Path space vs. Light field space
- Temporal / history data vs. single-frame
- Image data vs. Scene data: ray hit distance, surface roughness, normal, light position, size etc.
- Denoising of individual terms separately vs. denoising all the lighting at once
- AI based vs. hand-crafted

State of the Art Real-Time Denoising

“One sample per pixel can achieve a lot!”

- Exciting results with glossy reflections, area light soft shadows, and AO
- Effect-specific denoising, using scene information to estimate optimal filter foot print
- State of the art quality at extremely low sample count
  - Works well with even 1spp
  - No temporal reprojection involved ➔ ghosting/lagging free

Ray Traced Shadows, 1spp Denoised

Ray Traced Ambient Occlusion, 2spp Denoised

Ray Traced Reflections, 1spp Denoised

RTX Demo Video for slides 31-70:
https://www.youtube.com/watch?v=tjf-1BxpR9c

Ray Traced Area Light Shadows and Denoising

Ray Traced Area Light Soft Shadow

Why Ray Traced Shadows?

- More physically correct soft shadow visuals
  - Higher quality contact hardening even for really large light source
    - Not possible with shadow mapping based algorithms
  - More accurate geometry than character capsule shadows
  - Supports non-rigid motion (skinning, etc) unlike Distance Field Shadows
  - Can be combined with analytical area lighting
- Eliminate common shadow mapping artifacts:
  - Sampling rate mismatch, shadow acne, peter panning, CSM seams, etc

Soft Shadows with 1spp

Soft Shadows 1spp Denoised

Soft Shadows Ground Truth

Shadow Mapping

Area Light Soft Shadow Denoising

Denoiser Overview

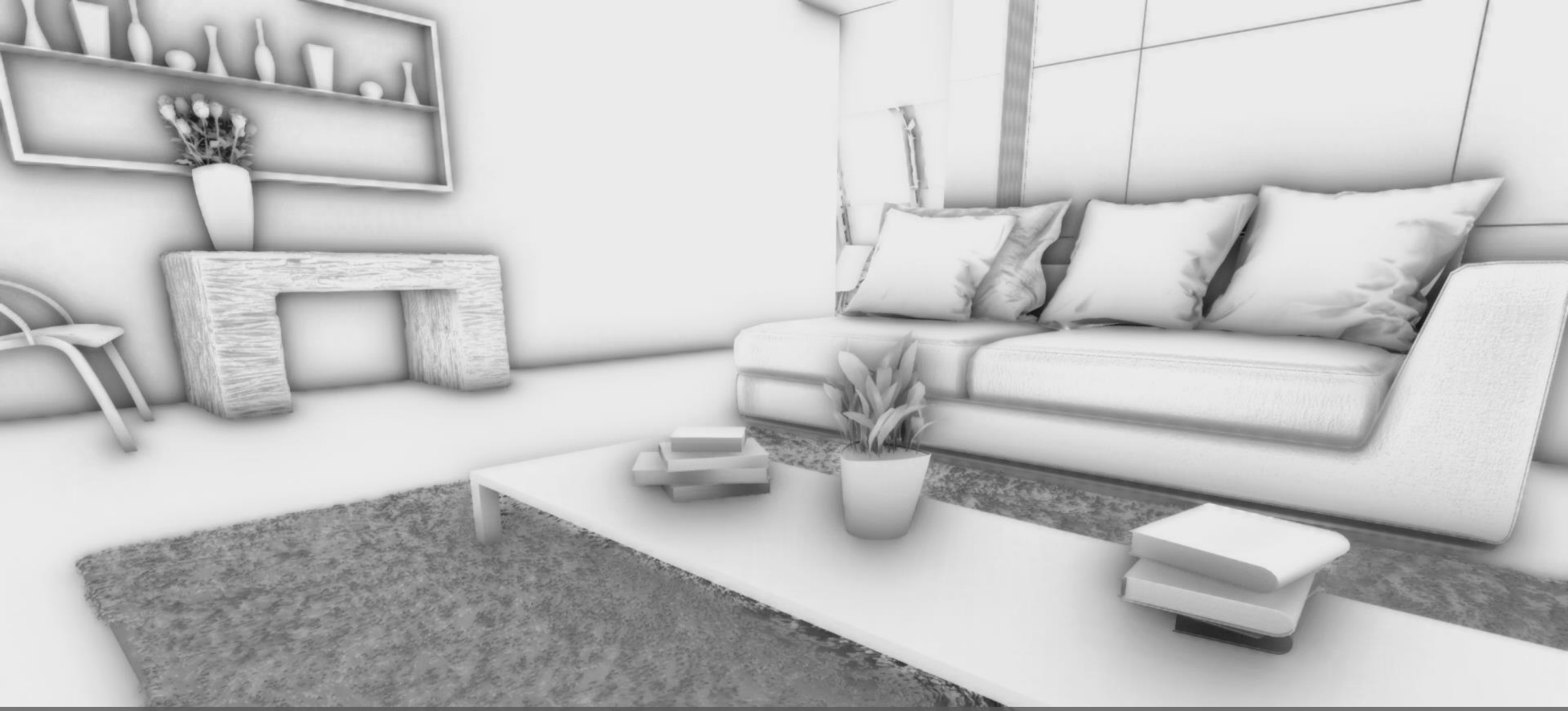
- Denoising only applied to visibility term (split sum approximation)
  - $L(\omega_o) = \int_\delta f(\omega_o,\omega_i)L_d(\omega_i)\mathrm{V}(\omega_i)|\omega_i \cdot n|d\omega_i \approx \underbrace{\int_\delta \mathrm{V}(\omega_i)d\omega_i}_{\text{Denoised!}} \underbrace{\int_\delta f(\omega_o,\omega_i)L_d(\omega_i)|\omega_i \cdot n|d\omega_i}_{\text{Analytical Approx.}}$
- Per light source denoising
- Inputs: Hit distance, Scene depth, Normal, Light size and direction
- Separated cross bilateral filter with variable filter radius and weights
- Different denoisers for radial lights, directional lights and rectangular lights
- Output: filtered visibility term

Ray Traced Ambient Occlusion and Denoising

Ray Traced Ambient Occlusion

Why Ray Tracing?

- Higher quality results compared with existing techniques
  - Physically correct ambient occlusion
  - SSAO leaves ‘dark halo’ around depth discontinuities
  - Screen Space techniques miss occlusion from geometry that is not visible:
    - Off screen boundaries
    - Behind the camera
    - Surfaces close to parallel to the view vector (e.g.: tables, chairs, very common)



Screen Space AO

Ambient Occlusion 2spp Denoised

Ambient Occlusion Ground Truth

Ambient Occlusion with 2spp

Denoising Ray Traced Ambient Occlusion

Denoiser Overview

- Separated cross bilateral filter
  - Estimate filter kernel size in world space based on hit T
- Using ideas from ‘Axis-Aligned Filtering for Interactive Physically-Based Diffuse Indirect Lighting’ [Mehta, el al. 13]
- Larger kernel size in open region, smaller in contact region
  - Visibility changes slower in open region, thus can share more spatially
  - Preserve contact darkening
- At 1spp, achieved smooth far field occlusion
- With 2-4 spp, can recover detailed occlusion in contact region

Ray Traced Reflections and Denoising

Ray Traced Reflections + Denoising

vs. Screen Space Reflections

- Screen Space Ray Traced Reflections
  - Missing data due to off screen ray hits
  - Incorrect specular on reflections
    - Specular of primary shading point being reused was computed with view direction instead of reflected ray direction


Screen Space Reflections


Ray Traced Reflections

Ray Traced Reflections + Denoising

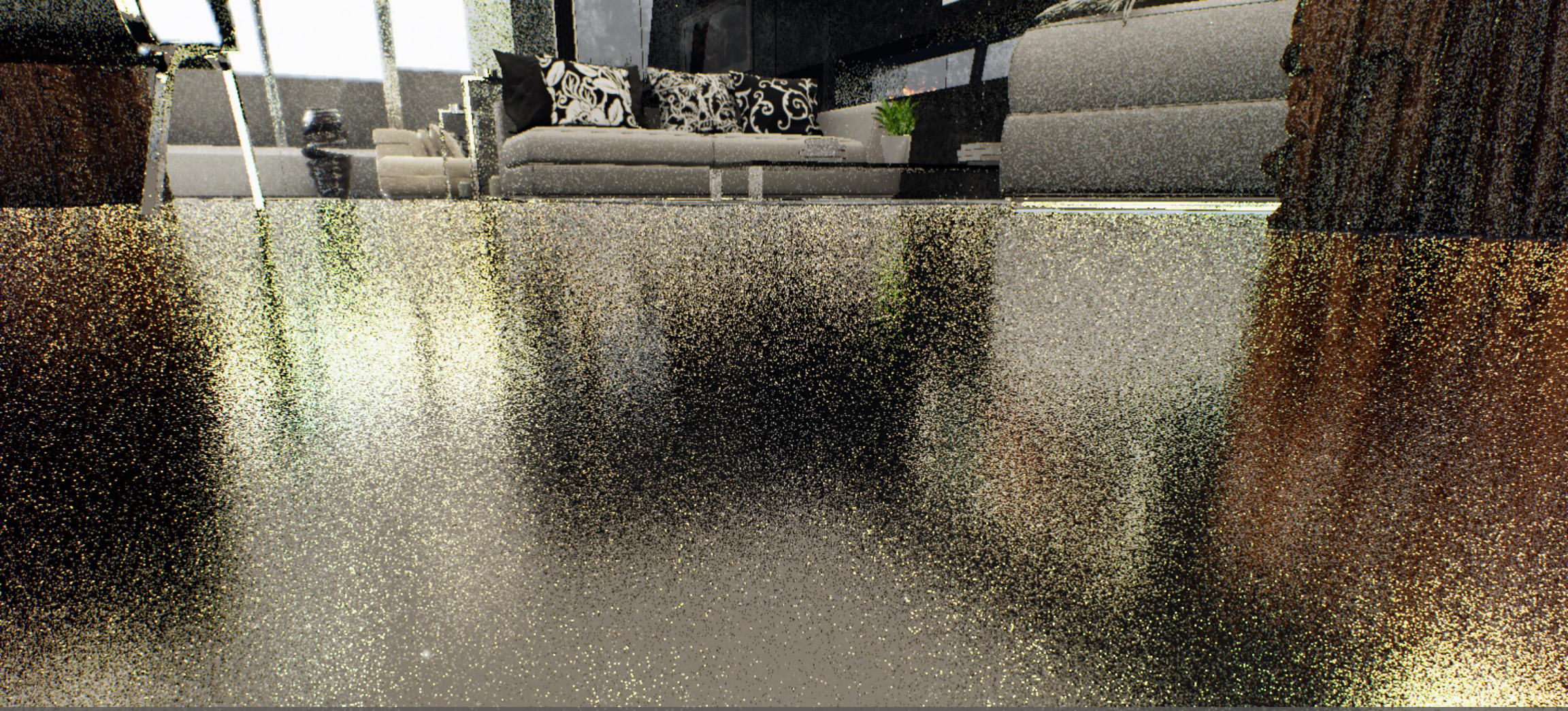
vs. pre-integrated light probes

- Pre-integrated light probes / environment maps
  - Usually static. Dynamic is rarely practical
  - Can be very wrong if not placed correctly
  - Roughness range discretized into fixed number of levels
  - Pre-integration with view vector perpendicular to receiver plane misses NDF anisotropy

Ray Traced Reflections + Denoising

vs. Planar Reflections

- Planar reflections
  - Only works for planar surfaces
  - Does not scale to many of these
  - No correct glossy reflections



Glossy Reflections with 1spp. Roughness = 0.18

Glossy Reflections 1spp Denoised

Glossy Reflections Ground Truth

Stochastic Screen Space Reflections + Probe

Glossy Reflections Denoising

Denoiser overview

- Denoising applied only to the incoming radiance
  - $L(\omega_o) = \int_\delta L(\omega_i) f(\omega_o, \omega_i) |\omega_i \cdot n| d\omega_i \approx \underbrace{\int_\delta L(\omega_i) d\omega_i}_{\text{Denoised!}} \underbrace{\int_\delta f(\omega_o, \omega_i) |\omega_i \cdot n| d\omega_i}_{\text{Pre-Integrated}}$
- Specular albedo included in pre-integration term so won't be overblurred
- Denoising only the radiance term, no noise from BRDF sampling
- Cross bilateral filter with aniso kernel footprint estimated based on projecting the BRDF lobe foot print back to screen space

Limitations

of current denoising technology

- **Shadows** denoiser results can be of lower quality for overlapping penumbra from two occluders with very different distances from the receiver
- **Reflections** denoiser is further from ground truth as roughness increases (e.g.: squared roughness > 0.4)
- **AO** denoiser may need at 2 or more rays to capture finer details

- Expect to continue improving over time to match more closely ground truth with better performance

Future Work

- Deep learning based denoisers
  - Using same input as these hand-crafted denoisers
  - Expect this could be the best solution
- Denoising all lighting component together to save time and memory
- Hybrid approaches combining screen space with world space tracing

Reference In-Engine Path Tracer

Path Traced Reference Image

Next Gen Rendering Work Flow

- Powerful tool to improve content creation pipeline
  - Render path traced image as target for tuning real-time rendering techniques
  - Material tuning, light probe placement, lighting setup, overall brightness…
- Extremely useful for rendering programmers too
  - Tuning and validating the look of real-time rendering algorithms
  - Adjusting shading of area lights
  - Tuning denoisers
  - **Know what you are missing in your hacks**

In-Engine Path Tracing

Easier today than it has ever been

- Powerful programming model in DXR abstracts all the complexity, making it easy to write light transport algorithms.
- RTX provides optimized implementations of Acceleration Structure build/update, as well as traversal and scheduling of ray tracing and shading
- Ability to use HLSL means you can reuse much of your existing shading code
- Physically based shading commonly used in game engines today makes building path tracer in game engine easier than it used to be

In-Engine Path Tracing

What we did in UE4

- Experimental Path Tracer written by us as soon as DXR was integrated and working with ray traced reflections.
- Not production quality at this point
- Separate ray generation shader runs path tracing loop
- Simplified Pixel Shader that only evaluates the material graph, but no lighting
  - Output returned to path tracing ray gen shader via payload
  - Engine-generated material shaders can be used directly (except where using ddx/ddy)

In-Engine Path Tracing

What we did in UE4

- BRDF importance sampling on top of UE4's shading model
- Light source sampling for all light types to do NEE
- Analytical ray-light intersection to support MIS initially
- Importance-sampled SkyLight cubemap
- Light culling when sampling from many light sources

DL Based Path Tracing Denoising

NVIDIA OptiX 5.0 AI Denoiser

- Integrated NVIDIA OptiX AI Denoiser, recently released in 5.0
- Improves quality of the rendered image from 0.8 ssim to 0.99 ssim (with respect to target reference image)
- Fast on Volta GPUs thanks to Tensor Cores
- Blend smoothly between noisy image and denoised image starting at some set number of iterations (depending on scene)

In-Engine Path Tracing + AI Denoising

Light Baking

In-Engine Light Baking

- Leverages a lot of the same code written for a reference path tracer
- Different types of light baking
  - 2D Lightmaps
  - volumetric lightmaps
  - environment capture cubemaps)
- Different modes of baking
  - Preview mode
  - Batch mode

Lightmap Baking Preview Mode

- Focus on updating only the lightmap texels contributing to the current image.
- Enables quick iteration for artists & lighters.
  - Adjust lights, move objects and see results instantly.
- Our experience:
  - Tried approach that launches path tracing work in lightmap space
  - In the end found it easier to get good results by path tracing in screen space while accumulating output into lightmap texels using NvInterlockedAddFp32 (via NVAPI)
  - Progressive lightmap computation, combined with smart temporal reuse and lightmap space denoising allows near real-time updates

Lightmap Baking Batch Mode

- Utilize one or more GPUs to bake all the lightmaps in a level until converged
- Can dispatch a ray generation shader 2D grid per lightmap texture, or per lightmap atlas
- Two ways to get to object space data at a lightmap texel:
  - Traditional: conservative rasterization of interpolated vertex attributes in lightmap space
  - New: ray tracing an Acceleration Structure built over the 2D lightmap UVs
- Each has pros and cons. New approach is convenient but loses benefits of conservative raster

Instant Lightmap Baking Preview

Game Engine Ray Tracing Integration

Game Engine Integration

High Level Plan

- Extend Graphics API abstraction with RT shader types and RT commands
- Build and Update Acceleration Structures
- Create shaders and create the Ray Tracing Pipeline State Object
- Update Shader Parameters
- Start experimenting with ray tracing techniques

Game Engine Integration

Build and Update Acceleration Structures

- Build Acceleration Structure for static geometry once
- Rebuild Top Level Acceleration Structure every frame
- Update Bottom Level Acceleration Structure for deforming geometry every frame
- Use Compute shader to process deformable geometry into a temporary buffer to use as input to Acceleration Structure Update and shaders

Game Engine Integration

Shaders and Shader Parameters

- Use dxcompiler utility function to convert VS+PS pair to Hit shaders
  - Supports generation of both Closest Hit and Any Hit shader (alpha-test only)
- Register shaders and create the Ray Tracing Pipeline State Object
- Update shader parameters:
  - Naïve approach: update shader parameters every frame for every object
  - Optimal: update only the shader parameters that changed

Game Engine Integration

Start Experimenting, then Optimize

- Replace existing deferred passes for shadows, AO and reflections with ray generation shaders + denoising
- Add new techniques such as path tracing and lightmap baking

- Some optimization tips:
  - Allow artists to create simplified materials for reflection shading
  - Build a single bottom level Acceleration Structure for an object with multiple sub-objects with different materials overlapping spatially
  - Use RAY_FLAG_ACCEPT_FIRST_HIT_AND_END_SEARCH for shadow and AO rays even if you need the hit distance for denoising purposes

Game Engine Integration

Challenges and solutions

- Decals: early prototype outside engine showed promise
- Tessellation: start by disabling it if possible, otherwise limit update rate, output tessellated geometry to buffers and provide as input to AS builder
- Texture LOD: tried approaches such as replicating part of the Pixel Shader code to compute UVs at two auxiliary points to be able to compute derivatives
  - Does not seem worth it in most scenes, unless tracing primary rays or flat mirror rays and texture has high frequencies
  - TAA can eliminate a lot of the aliasing

Wrapping Up

Takeways

RTX and DXR brings real-time ray tracing to developers

- Biggest change in Graphics APIs since programmable shaders
- A few rays per pixel practical today for lighter weight effects
- More expensive ones (path tracing) can provide useful workflow improvements
- We showed a few examples of what's possible
- Expect you to come up with bright new ideas
- Hope this was inspiring and exciting

Thanks & Acknowledgements

- **NVIDIA:** Alex Bogomjakov, Evan Hart, Matthias Hollaender, Matthew Johnson, Martin-Karl Lefrancois, Adam Marrs, Laurent Ruhlmann, Jacopo Pantaleoni, Fredrik Liljegren, Cem Cebenoyan, Martin Stich, Steve Parker, Tony Tamasi, Jon Story, Louis Bavoil, Iain Cantlay, Nuno Subtil, Aaron Lefohn, Jon Hasselgren, Jacob Munkberg, Chris Wyman, Marco Salvi, Alex Keller, Carsten Wachter, Magnus Andersson, Robert Toth, Pascal Gautron, Shivan Taher, the rest of the SW team who made RTX possible
- **Epic Games:** Marcus Wassmer, Brian Karis, Patrick Kelly, Juan Canada, Arne Schober, Uriel Doyon, Yuriy O'Donnell, Jerome Platteaux, Kim Libreri and the rest of the team who worked on the Reflections demo

Thank you!

Ray Tracing Gems

Call for Papers

- A new book series with focus on real-time and interactive ray tracing for game development using the DXR API.
- We invite articles on the following topics:
  Basic ray tracing algorithms, effects (shadows, reflections, etc), non-graphics applications, reconstruction, denoising, & filtering, efficiency and best practices, baking & preprocessing, ray tracing API & design, rasterization and ray tracing, global Illumination, BRDFs, VR, deep learning, etc.
- Important dates
  - 15th of October 2018: submission deadline for full papers
  - GDC 2019: publication of Ray Tracing Gems (paper version + e-book)
- Eric Haines and Tomas Akenine-Möller will lead the editorial team
  http://developer.nvidia.com/raytracinggems/

Questions?

Ignacio Llamas, illamas AT nvidia.com

Edward Liu, edliu AT nvidia.com

Booth #223 - South Hall

www.nvidia.com/GDC