Watch the trailer here:
https://www.youtube.com/watch?v=LXo0WdIEIJK
“PICA PICA”

Exploratory mini-game & world

- For our self-learning AI agents to play, not for humans 😊
- Uses SEED’s Halcyon R&D engine
- Goals
  - Explore hybrid raytracing with DXR
  - Clean and consistent visuals
  - Procedurally-generated worlds
  - No precomputation
Why raytracing?

- Flexible new tool in the toolbox
- Solve sparse & incoherent problems
- Unified API + performance (DXR + RTX)
- Simple high quality - easy ground truth
Self-Learning AI

Using deep reinforcement learning

- “Imitation Learning with Concurrent Actions in 3D Games” [Harmer 2018]
- 36 semantic views - 1550 fps
- Training with TensorFlow
- Future: Inference with WinML

See “Deep Learning - Beyond the Hype” tomorrow
Hybrid Rendering Pipeline

- **Deferred shading** *(raster)*
- **Direct shadows** *(raytrace or raster)*
- **Direct lighting** *(compute)*
- **Reflections** *(raytrace)*
- **Global Illumination** *(raytrace)*
- **Ambient occlusion** *(raytrace or compute)*
- **Transparency & Translucency** *(raytrace)*
- **Post processing** *(compute)*
- **Spawn a Mesh?**
  - DXR: build its bottom acceleration structure
  - Multiple geometries for multiple materials
    - Triangles, AABBs, custom
  - Mesh instances specified in top acceleration

- **Move a Mesh?**
  - Update the instance’s position/orientation in the top acceleration

- **Spawn [some] Rays?**
  - Multiple Hit and Miss shaders possible
Raytraced Reflections

- Rasterize primary visibility
- Launch rays from the G-Buffer
- Raytrace at half resolution
- Reconstruct at full resolution
  - Spatiotemporal filtering
- Works on both flat and curved surfaces
Let's launch some reflection rays:

1. Select one of the (2x2) pixels to trace
2. Reconstruct position and vectors
3. Initialize Halton & random number seq.
4. Initialize the payload
5. Prepare a new ray
6. Trace
7. Gather results from ray payload
   - Reflection Color, Direction, HitT, 1/pdf
Reflection Filtering

Inspired by Stochastic Screen-Space Reflections [Stachowiak 2015]

- For every full-res pixel, sample 16 pixels in half-res ray results
  - Blue Noise offsets, decorrelated every 2x2 pixels

- Build color bounding box of ray-hit results
  - Clamp temporal history to bounding box

- Followed by a variance-driven bilateral filter
  - Helps with rough reflections

Unfiltered (Top) and Filtered (Bottom) Results
Screen-Space Reflections  

G-Buffer Raytraced  

Path Tracing Reference
Materials

Combine multiple microfacet surface layers into a single, unified & expressive BRDF

- Inspired by Arbitrarily Layered Micro-Facet Surfaces [Weidlich 2007]
- Unified for all lighting & rendering modes
  - Raster, path-traced reference, and hybrid
  - Energy conserving & Fresnel
- Rapidly experiment with different looks
  - Bake down number of layers for production
Materials

**Standard**
- Aluminum
- Brushed Aluminum
- Coated Carbon
- Copper
- Silver Satin
- Shiny / Mat Plastic
- Dark Rubber

**Exotics**
- Glass
- Rough Glass
- Jade / Marble

*Integrated in BRDF, but built separately*
BRDF Sampling

- General idea: Launch one ray for the whole stack
  - Stochastically select a layer & sample
  - When evaluating material, estimate visibility vs other layers
    - Some energy is reflected, some is refracted:
      - Microfacet: Fresnel (refracted = 1 - Fresnel)
      - Diffuse: reflected fraction equal to albedo, remaining light absorbed; no refraction
  - Sample BSDF of selected layer
    - Attenuated by layer(s) on top

- Result is temporally filtered
  - A single value for many layers requires clever filtering

Multi-Layered Materials [Weidlich 2007]
Transparency & Translucency

Raytracing allows to unify reflections and refractions

- **Glass**
  - Order-independent (OIT)
  - Handles multiple IOR transitions

- **Translucency**
  - Inspired from Translucency in Frostbite [Barré-Brisebois 2011]
  - Inner structure scattering based on lighting traveling inside the medium

- Performance: we do it in texture-space
Translucency Breakdown

- For every valid position & normal
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - Alternatively, front + back cosine lobes
  - Perf: only do n-rays per frame
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - Alternatively, front + back cosine lobes
  - Perf: only do n-rays per frame
- Compute lighting at intersection
  - Sample previous translucency result
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - Alternatively, front + back cosine lobes
  - Perf: only do n-rays per frame
- Compute lighting at intersection
  - Sample previous translucency result
- Gather
  - Modulate with Beer-Lambert or Henyey-Greenstein phase function
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - Alternatively, front + back cosine lobes
  - Perf: only do n-rays per frame
- Compute lighting at intersection
  - Sample previous translucency result
- Gather
  - Modulate with Beer-Lambert or Henyey-Greenstein phase function
- Store new BTDF value
Translucency Filtering

Result converges over a couple frames

- Denoised or temporally-accumulated
- Temporal: build an update heuristic
  - Exponential moving average can be OK
  - Game-specific threshold to update
  - Reactive enough for moving lights & objects

- Variance-adaptive mean estimation
Translucency Shadowing

Raytracing allows for globally shadowed translucency

- Global phenomena
  - Integration via feedback
  - Objects occlude each other

- Overall more grounded and visually-convincing translucency
Transparency

Similar approach is used for glass

- Launch ray using view’s origin and direction
- Refract based on medium’s index-of-refraction (IOR)
  - Snell’s law: \( \text{refract}(\text{ray}, N, \text{iorInput} / \text{iorOutput}) \)
  - DXR: \( \text{HitKind}() \) to handle IOR transitions (Air ⇔ Glass)
- Trace a ray in the scene & sample lighting
- Tint the result by glass color + chromatic aberration
Transparency

Similar approach is used for glass:

- Launch ray using view’s origin and direction
- Refract based on medium’s index-of-refraction (IOR)
  - Snell’s law: `refract(ray, N, iorInput / iorOutput)`
  - DXR: `HitKind()` to handle IOR transitions (Air ⇔ Glass)
- Trace a ray in the scene & sample lighting
- Tint the result by glass color + chromatic aberration

- Pen: we don’t handle transparent shadows yet
Transparency

Works for clear and rough glass

- Clear
  - No filtering required
- Rough / Blurry
  - Open cone angle with Uniform Cone Sampling [PBRT]
  - Wider cone → more samples
  - Or temporal filtering

- Tint with phase function and more complex BTDF
Global Illumination

- We want a technique that:
  - Doesn’t require any precomputation
  - Doesn’t require parametrization (UVs, proxies)
  - Works for both static and dynamic scenes
  - Adaptive & refines itself on-the-fly

- Point-based / surfels for a dynamic world
  - Runtime Monte-Carlo integration
Surfel Placement

Surfel Spawning From Camera @ 1% speed
Surfel Placement

Skinned Surfels for Dynamic Objects
Sampling & Integration

- Compute surfel irradiance by PT
- When shooting rays/frame
  - Limit depth and number of paths
- Limiting depth $\neq$ fewer bounces
  - Reuse results from previous frames
  - $1 = \text{radiosity}, \infty = \text{path tracing}$
- Variance-adaptive mean estimator
- Application (Binning, culling, half-res apply + upsample)
Shadows

Accumulated and filtered in screen-space

- **Raygen**: Launch a ray towards light
  - Payload’s `miss` flag set to `true` (from Miss Shader) if it doesn’t hit geometry
  - Penumbra driven by uniform cone sampling [PBRT]

- **Temporal Reprojection**
  - Accumulates shadow and variance + TAA-style bounding box clamping

- **Filter (SVGF-like)** [Schied and NVIDIA 2017]
  - Multipass weighted spatial blur, driven by variance from temporal accumulation
Hard Raytraced Shadows
Soft Raytraced Shadows (Filtered)
What about texture level of detail?

- Mipmapping [Williams 1983] is the standard method to avoid texture aliasing:

\[
\lambda(x, y) = \log_2 [\rho(x, y)]
\]

\[
\rho(x, y) = \max \left\{ \sqrt{\left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial x} \right)^2}, \sqrt{\left( \frac{\partial u}{\partial y} \right)^2 + \left( \frac{\partial v}{\partial y} \right)^2} \right\}
\]

- Screen-space pixel maps to approximately one texel in the mipmap hierarchy
- Supported by all GPUs for rasterization via shading quad and derivatives
Texture Level-of-Detail

No shading quads for ray tracing!

- **Traditionally: Ray Differentials**
  - Estimates the footprint of a pixel by computing world-space derivatives of the ray with respect to the image plane
  - Have to differentiate (virtual offset) rays
  - Heavier payload (12 floats) for subsequent rays (can) affect performance. Optimize!

- **Alternative:** always sample mip 0 with bilinear filtering (with extra samples)
  - Leads to aliasing and additional performance cost

Ray Differentials [Igehy99]
Texture Level-of-Detail

Together with NVIDIA Research, we developed a texture LOD technique for raytracing:

- Heuristic based on triangle properties, a curvature estimate, distance, and incident angle
  - Similar quality to ray differentials with single trilinear lookup
  - Single value stored in the payload for subsequent rays

- Upcoming publication by:
  - Tomas Akenine-Möller (NV), Magnus Andersson (NV), Colin Barré-Brisebois (EA), Jim Nilsson (NV), Robert Toth (NV)
Summary

▪ Just the beginning – important new tool going forward
▪ Unified API – easy to experiment and integrate
▪ Flexible but complex tradeoffs - noise vs ghosting vs perf
▪ Can enable very high quality cinematic visuals
▪ Lots more to explore – perf, raster vs trace, sparse render, denoising, new techniques
SEED @ GDC 2018

- **DirectX: Evolving Microsoft's Graphics Platform (presented by Microsoft)**
  - Johan Andersson and Colin Barré-Brisebois
  - Content will be available online soon at www.ea.com/seed

- **Deep Learning - Beyond the Hype**
  - Magnus Nordin
  - Room 2016, West Hall, Thursday, March 22nd, 11:30am - 12:30pm

- **Creativity of Rules and Patterns: Designing Procedural Systems**
  - Anastasia Opara
  - GDC Show Floor, Thursday, March 22nd, 12:30PM-1:00PM and Friday, March 23rd @ 11:00AM-11:30AM
Thanks

- **SEED**
  - Joakim Bergdahl
  - Ken Brown
  - Dean Calver
  - Dirk de la Hunt
  - Jenna Frisk
  - Paul Greveson
  - Henrik Halen
  - Effeli Holst
  - Andrew Lauritzen
  - Magnus Nordin
  - Niklas Nummelin
  - Anastasia Opara
  - Kristoffer Sjöö
  - Tomasz Stachowiak
  - Ida Winterhaven
  - Graham Wihlidal

- **Microsoft**
  - Chas Boyd
  - Ivan Nevraev
  - Amar Patel
  - Matt Sandy

- **NVIDIA**
  - Tomas Akenine-Möller
  - Nir Benty
  - Jiho Choi
  - Peter Harrison
  - Alex Hyder
  - Jon Jansen
  - Aaron Lefohn
  - Ignacio Llamas
  - Henry Moreton
  - Martin Stich
References

- **[Harmer 2018]** Jack Harmer, Linus Gisslén, Henrik Holst, Joakim Bergdahl, Tom Olsson, Kristoffer Sjöö and Magnus Nordin “Imitation Learning with Concurrent Actions in 3D Games”. [available online](#).
- **[Igehy 1999]** Igehy, Homan. “Tracing Ray Differentials”, [available online](#).
- **[Schied 2017]** Schied, Christoph et. Al. “Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination”, NVIDIA Research, [available online](#).
- **[Stachowiak 2015]** Stachowiak, Tomasz. “Stochastic Screen-Space Reflections”, [available online](#).
- **[Williams 1983]** Williams, Lance. “Pyramidal Parametrics”, [available online](#).
Questions?
Ambient Occlusion (AO) [Langer 1994] [Miller 1994] maps and scales directly with real-time ray tracing:

- Integral of the visibility function over the hemisphere $\Omega$ for the point $\bar{p}$ on a **surface** with normal $\hat{n}$ with respect to the projected **solid angle**

- Games often approximate this in screen-space
- With RT, more grounded & improves visual fidelity!
  - Random directions $\hat{w}$
  - Can be temporally accumulated or denoised
Raytraced AO
(Same Radius as SSAO)
mGPU

Explicit Heterogenous Multi-GPU

- Parallel Fork-Join Style
- Resources copied through system memory using copy queue
- Minimize PCI-E transfers

Approach

- Run ray generation on primary GPU
- Copy results in sub-regions to other GPUs
- Run tracing phases on separate GPUs
- Copy tracing results back to primary GPU
- Run filtering on primary GPU
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)

- Tomas Akenine-Möller will lead the editorial team
  http://developer.nvidia.com/raytracinggems/