Are We Done With Ray Tracing?
State-of-the-Art and Challenges in Game Ray Tracing

Colin Barré-Brisebois, @ZigguratVertigo
SEED – Electronic Arts
How did we get here?
GDC 2018 – DXR Unveiled

“Ray tracing is the future and ever will be”
Real-Time Ray Tracing in Software and Hardware
Real-Time Hybrid Ray Tracing in Unreal Engine 4

Left: real-world footage. Right rendered with Unity

[Tatarchuk 2019, Courtesy of Unity Technologies]
Left: real-world footage. Right rendered with Unity.
Left: real-world footage. Right rendered with Unity

[Tatarchuk 2019, Courtesy of Unity Technologies]
And many more...

- Assetto Corsa
- Atomic Heart
- Call of Duty: Modern Warfare
- Cyberpunk 2077
- Enlisted
- Justice
- JX3
- Mech Warrior V: Mercenaries
- Project DH
- Stay in the Light
- Vampire: The Masquerade – Bloodlines 2
- Watch Dogs: Legion
- Wolfenstein: Youngblood

Just the beginning of real-time ray tracing making its way into game products

We’re in for a great ride, and the work is not done! This is super exciting! 😊
Future Consoles

→ Microsoft: E3 2019 Keynote, June 9<sup>th</sup> 2019, [link](#)
→ SONY: What to Expect From SONY’s Next-Gen PlayStation, Wired Magazine, April 16<sup>th</sup> 2019, [link](#)
State of the Art
Hybrid Rendering Pipeline

Deferred shading (raster)
Direct shadows (ray trace or raster)
Lighting (compute + ray trace)
Reflections (ray trace or compute)

Global Illumination (compute and ray trace)
Ambient occlusion (ray trace or compute)
Transparency & Translucency (ray trace and compute)
Post processing (compute)
Reflections
- Launch rays from G-Buffer
- Trace at half resolution
- Supports arbitrary normal & roughness
- Extensive spatial & temporal reconstruction
Spatial Reconstruction
+Temporal Accumulation
- Secondary bilateral filter
  - Only run where variance high
    - Variance from spatial reconstruction
    - Temporally smoothed with hysteresis = 0.5
  - Much dumber than reconstruction
  - Introduces blur
- Variable kernel width, sample count
Bilateral cleanup
Temporal Anti-Aliasing
Raw Ray-traced Output
Alternatively, Hybrid RT + SSR
Hybrid RT & SSR

- Figure out which pixels can rely on screen space results
  - Otherwise, trace in world

- Performance [Deligiannis 2019]
  - Variable Rate Tracing
    - More rays for water & grazing angles
  - Ray Binning

"It Just Works": Ray-Traced Reflections in 'Battlefield V' [Deligiannis 2019]

<table>
<thead>
<tr>
<th>Process</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable Rate Tracing</td>
<td>0.37</td>
</tr>
<tr>
<td>Generate Rays</td>
<td>0.19</td>
</tr>
<tr>
<td>Ray Binning</td>
<td>0.36</td>
</tr>
<tr>
<td>Screen Space Hybrid</td>
<td>0.15</td>
</tr>
<tr>
<td>Intersect/Material Data</td>
<td>1.98</td>
</tr>
<tr>
<td>Defrag</td>
<td>0.46</td>
</tr>
<tr>
<td>Lighting</td>
<td>1.45</td>
</tr>
<tr>
<td>Spatial Filter</td>
<td>0.24</td>
</tr>
<tr>
<td>Temporal Filter</td>
<td>1.00</td>
</tr>
<tr>
<td>Image Filter</td>
<td></td>
</tr>
</tbody>
</table>
Managing Coherency

- **Coherency** is key for RTRT performance
  - Coherent $\rightarrow$ adjacent work performing similar operations & memory access
    - Camera rays, texture-space shading
  - Incoherent $\rightarrow$ thrash caches, kills performance
    - Reflection, shadows, refraction, Monte Carlo
- You’re on your own: hardware won’t take care of it for you
Managing Coherency

- **Inspiration from Offline**
  - Sort large out-of-core ray batches & ray-hits for deferred sharing
  - A few options: [Aalto2018]
    - Use shadow maps for reflection shadows
    - Split ray tracing and shading
    - Group shading per material
    - Limit tracing on roughness

- **Sorted Deferred Shading for Production Path Tracing**
  [Eisenacher 2013]
Ray Binning

Group rays that are directionally aligned to maximize coherency [Deligiannis 2019] [Benyoub 2019] [Majercik 2019]

1. Split the screen in (32x32) tiles
2. Generate (random) rays
3. Sort rays in octahedral space for ray direction binning
4. For each bin, launch rays
5. Gather hit results in G-Buffer
6. Shade in Compute Shader

"It Just Works": Ray-Traced Reflections in 'Battlefield V' [Deligiannis 2019]

A Survey of Efficient Representations for Independent Unit Vectors [Cigolle 2014]
Ambient Occlusion
\[ A_p = \frac{1}{\pi} \int_{\Omega} V_{p, \omega} (\hat{n} \cdot \hat{\omega}) d\omega \]
Remedy’s Ray-traced Ambient Occlusion in Northlight Engine [Aalto2018]
Shadows
- Launch ray towards the light
- Ray misses \(\rightarrow\) Not in shadow
- Soft Shadows?
  - Random cone directions. Width drives penumbra
Hard Raytraced Shadows

Soft Raytraced Shadows (Unfiltered)

Soft Raytraced Shadows (Filtered)
Combining Analytic Direct Illumination and Stochastic Shadows [Heitz2018]
Combining Analytic Direct Illumination and Stochastic Shadows [Heitz2018]
Transparent Shadows

- Shadows from transparency
  - Hard to get right with raster [McGuire17]

- Let’s flip it around
  - Trace towards light, like opaque
  - Accumulate absorption
    - Product of colors
    - Thin film approximation
    - Density absorption easy extension
  - Can also be soft!
Transparent Shadows (1/)

- Keep tracing until
  - Hit opaque surface or all light is absorbed or miss
Transparent Shadows (hard)
Transparent Shadows (soft)
SIGGRAPH 2019 – Are We Done With Ray Tracing? – State of the Art and Challenges in Game Ray Tracing
Transparency & Translucency
Transparency & Translucency

- Raytracing enables **accurate light scattering**

- **Transparency**
  - Order-independent (OIT)
  - Multiple index-of-refraction transitions
  - Variable roughness, refractions and absorption

- **Translucency**
  - Light scattering inside homogeneous medium

- We do this in **texture-space**
  - Handle view-dependent terms & dynamic changes to the environment

Texture-Space Glass and Translucency
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.
  - (Importance-sample phase function)
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*
- Compute lighting at intersection
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - (Importance-sample phase function)
- Compute lighting at intersection
- Gather all samples
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - (Importance-sample phase function)
- Compute lighting at intersection
- Gather all samples
- Update value in texture
Transparency

Works for clear and rough glass

- **Clear**
  - No filtering required

- **Rough**
  - *Microfacet Models for Refraction through Rough Surfaces [Walter2007]*
  - More samples + temporal filtering

- Apply phase function & BTDF
Rough Transparency
Rough Transparency
Many Lights
Many Lights

Which (most affecting) lights to choose?

Acceleration structure-based selection:
- Unity: camera-oriented acceleration structure [Benyoub 2019] [Tatarchuk 2019]
- BFV: horizontal plane light list [Deligiannis 2019]

Light Importance Sampling:
- Dynamic Many-Light Sampling for Real-Time Ray Tracing [Moreau 2019]
- Stochastic Lightcuts [Yuksel 2019]
Particles?
Particles

- Particles have to be ray-aligned
  - Not perfect since VFX are often designed assuming view-aligned particles

- Battlefield V [Deligiannis 2019]
  1. Shoot ray in Opaque TLAS
  2. Shoot again in Particle TLAS
     1. Limit length from opaque hit
     3. Blend particles with opaque hit

- Rotate odd particles 90 deg. around Y
Other Things
(yet still super important!)
Ray-traced GI

Ray-traced approaches making their way:
- Surfels [Stachowiak 2018]
- Grid [Aalto 2018]
- Probes [Majercik 2019]
Culling

- Can’t rely on Frustum Culling, since rays are in world space
- In the case one can’t have all objects in the BVH, have to find new culling heuristic
- Projected bounding sphere [Deligiannis 2019]

"It Just Works": Ray-וTraced Reflections in 'Battlefield V' [Deligiannis 2019]
Texture LOD

- **No automatic texture LOD with ray tracing**
  - Inherent of the rasterization pipeline: pixel quad derivatives

- In Ray Tracing Gems [Akenine-Möller 2019]
  - 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

- Barely scratched the surface – still work to do!
  - Still some needed improvements. Come get inspired at the talk! 😊
  - Texture Level of Detail Strategies for Real-Time Ray Tracing.
    In Ray Tracing Gems 1.1 Session Room: 501AB Wed @ 2PM
Performance Good Practices

- Minimize recursion: favor fire-and-forget / tail-recursive techniques
- Consider manual scheduling (sorting/binning) [Aalto 2018] [Deligiannis 2019]
- Minimize/pack payload and attributes
- Optimize your any-hit shaders (or don’t use them at all)
- BLAS build & update on an async queue: define update vs rebuild metric for your case, over multiple frames, overlapped with raster
- TLAS: Build instead of Update & don’t include the skybox (do in Miss Shader)
Challenges
Game Constraints

Need robust techniques for video games:
- Many animated characters
- Dynamic environments
- Moving foliage & vegetation
- Massive open worlds
- User generated content
- User created experiences

Complex & Large Scale Environments in EA BioWare’s Anthem
Transparency

- **Transparency** is far from being solved!
  - Glass
    - Clear/rough + filtered + shadowed
  - Particles & Volumetric Effects
    - Can use miss shader to update volumes / clipmaps
    - Ray marching in hit shaders?
    - Non-trivial blending & filtering

- PICA PICA: **texture-space OIT** with refractions and scattering
  - Not perfect, but one step closer

- Denoising techniques don’t work so well with transparency (and 1-2 spp)
Partial Coverage

- **Foliage**
  - Can still do alpha test (i.e.: any-hit)
  - Animated becomes a real problem

- **Defocus effects** such as motion blur and depth of field are still intractable

- Need partial coverage denoising @ 1-2 spp
Future Ray Tracing Research

- Literature has to adjust to real-time game ray tracing constraints
  - Games → budgeted amount rays/frame, rays/pixel, fixed frame times & memory budgets
  - Games → Light transport caches: surfels, voxels, lightmaps

- Moving towards decoupled shading & variable rate ray tracing
  - Texture-space Techniques
    - Caching of materials and partial solutions
    - Split the BRDF: view vs view-independent terms

- Perf R&D: Efficient sampling / integration strategies & reconstruction filtering
Global Illumination

- **Open problem even in offline rendering**
  - Variance still too high
  - Can reduce frequency
    - Prefiltering, path-space filtering
    - Denoising & reconstruction
  - Pinhole GI & lighting is not solved

- **Incoherent** shading $\rightarrow$ intractable performance
  - Have to resort to caching to amortize shading
  - PICA PICA: caching of GI via surfels
    - Issues: only spawn surfels from what you see

- Need to solve GI for **user-generated content** $\rightarrow$
Sparse BVH Tracing

- We assume ray-triangle intersection as the end-all-be-all, but what if we stopped the ray higher up the tree?
  - Treat AABBs like voxels
    - Akin to beam tracing [Heckbert 1984], or “ray bundles”
  - Explore algorithms that would benefit from broad tracing
    - Global Illumination Using Ray-Bundle Tracing [Tokuyoshi 2012]
    - Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields [Majercik 2019]
    - Cone Tracing [Crassin 2011]
    - Sound Propagation
- Acceleration is currently ray-tri & ray-AABB
  - Feels like it’s all here.
  - Expose trace “LOD” controls to developers at API level
Evolving Engines for Hybrid RT

- Bindless
- Visibility-Buffer
- Texture-space Ray Tracing
- Raster still shows some advantages for primary opaque visibility over RT
- ...

- Scene Management for RT
- Shader Compilation [Deligiannis 2019]
Summary

- Real-time ray tracing brings the game to another level
- Still have a lot of work to do to bridge offline and real-time
- DXR provides a playground where research and gamedevs can collaborate even more, and solve challenges together!
Thank You!

- SEED
- Alex Keller (NVIDIA)
- Natasha Tatarchuk (Unity Technologies)
- Marcus Wassmer (Epic Games)
- Jan Schmid and Johannes Deligiannis (DICE)
- Jon Greenberg (for BVH & beam tracing discussions)
References

- [Epic, NVIDIA and ILMxLAB 2019] "Epic Games demonstrates real-time ray tracing in Unreal Engine 4 with ILMxLAB and NVIDIA" online.
SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION

Stockholm – Los Angeles – Montréal – Remote

seed.ea.com

We're Hiring!