AMD'S RAY TRACING RESEARCH

TAKAHIRO HARADA

INTRODUCTION

- High level overview of our ray tracing related R&D

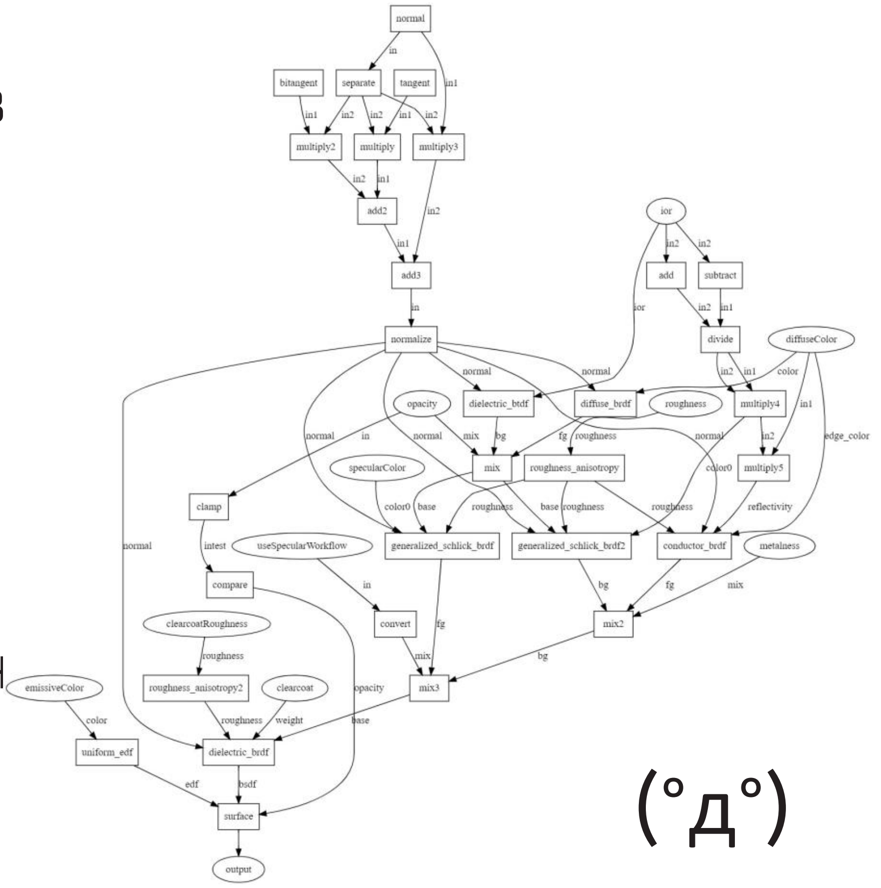
- Topics
  - Geometric representation
  - Improvements in importance resampling techniques
  - Volume sampling
  - Orochi
  - HIPRT
  - Radeon ™ ProRender

MULTI-RESOLUTION GEOMETRIC REPRESENTATION

DO WE ALWAYS NEED TO RAY CAST AGAINST FINE GEOMETRIES?

- Problem description
  - Is it possible to reduce the algorithmic complexity of ray cast using B
  - Explicit LOD is the solution for rasterization
    - Requires both pre-processing and additional memory for storing these
    - Can we do better?
  - What do we do for materials at another geometric representation?
    - Simply filtering parameters do not work for complex material graph

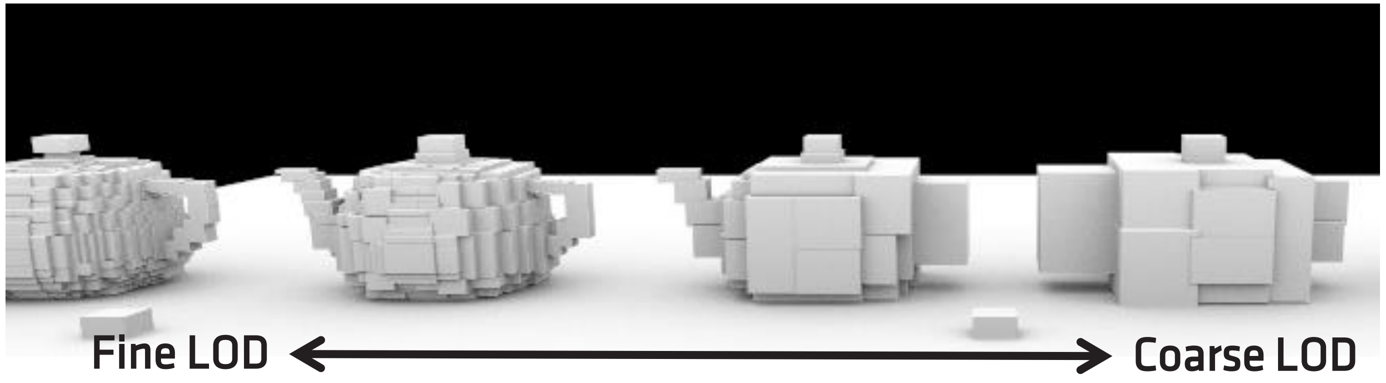
- Our geometric representation
  - Re-using the existing data structure for ray tracing, which is the BVH
    - No need for additional data
  - Stochastic material sampling
    - No assumption about materials on the geometry we simplify
    - No restriction on the material

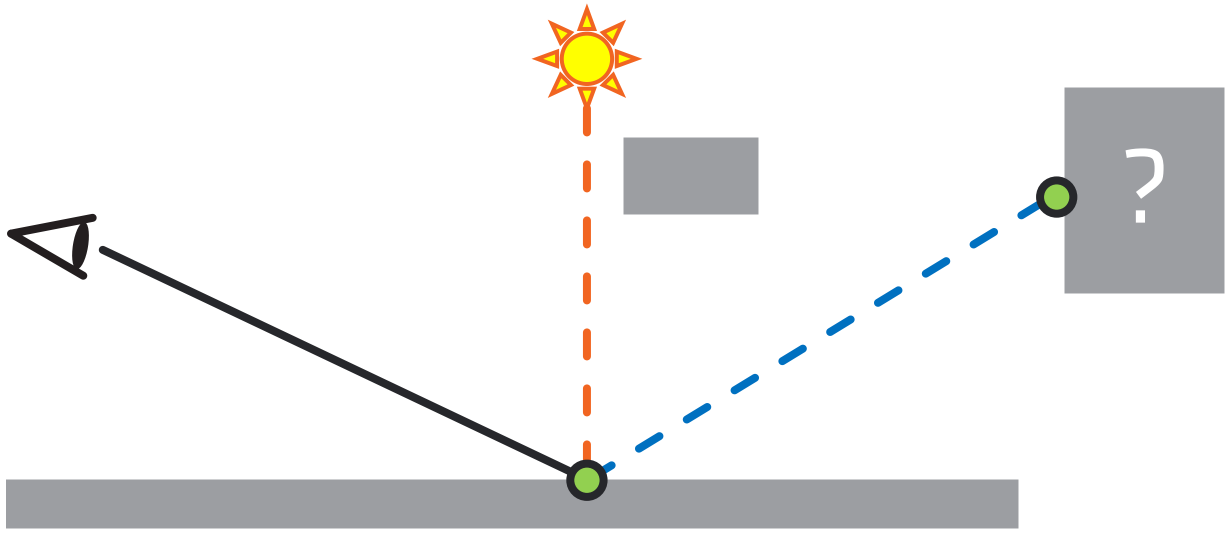


(°д°)

METHOD

- Geometric approximation
  - Use AABBs as a coarse approximation
  - Multi-resolution representation of geometry without any additional precomputation
  - Use ray cone for LOD computation

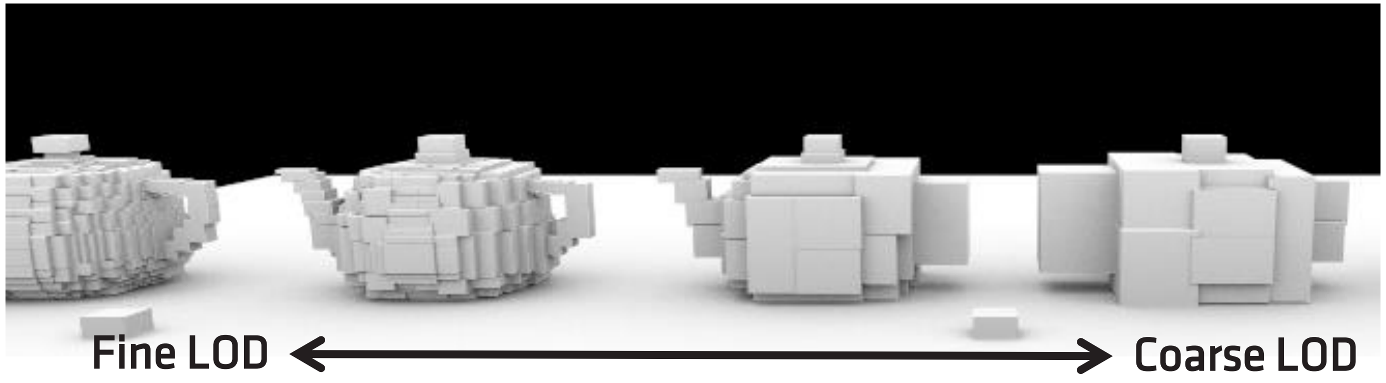


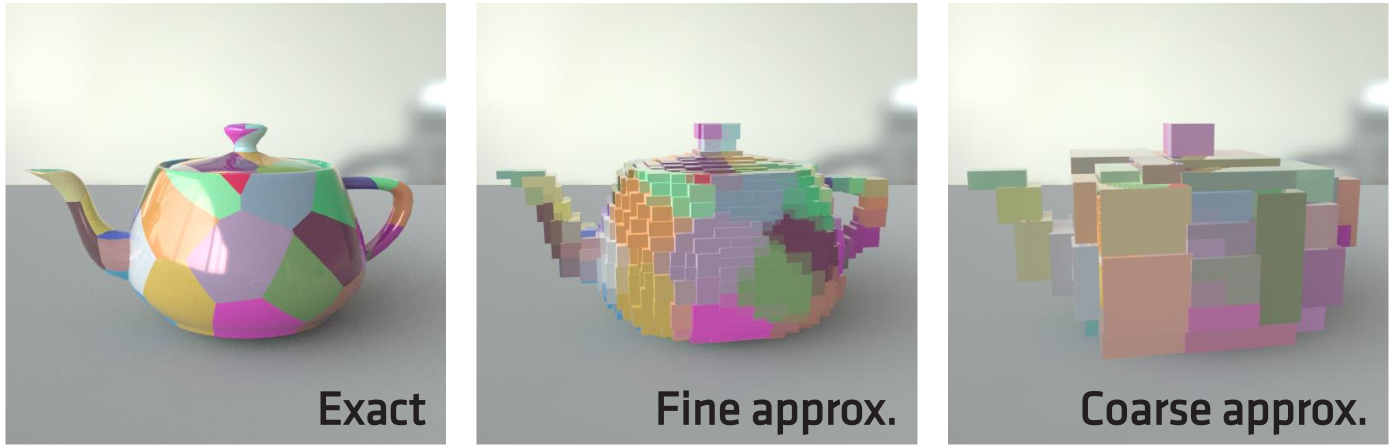


METHOD

- Geometric approximation
  - Use AABBs as a coarse approximation
  - Multi-resolution representation of geometry without any additional precomputation
  - Use ray cone for LOD computation

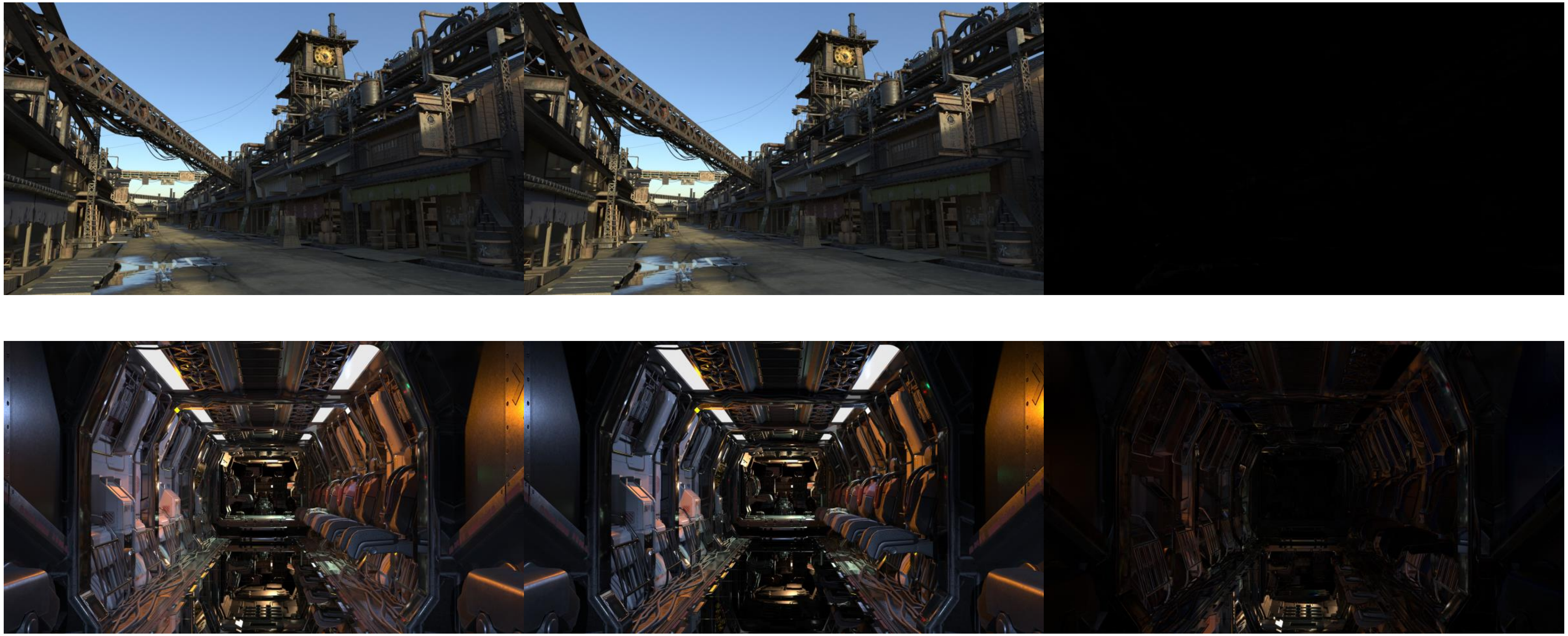
- Stochastic material sampling
  - Sample a triangle of node's descendants and use its material for shading
  - Only two integers are stored in each node
  - Support complex shading network





Procedural texture can be filtered

RESULTS



Path traced Approximation Diff

Multi-Resolution Geometric Representation using BoundingVolume Hierarchy for Ray Tracing, S. Ikeda et al., GPUOpen Tech Report, No. 22-02-f322

IMPORTANCE RESAMPLING
+ RESERVOIR SAMPLING RELATED

IMPORTANCE RESAMPLING + RESERVOIR SAMPLING

- Importance resampling
  - Importance Resampling for Global Illumination, Talbot et al., 2005

- Reservoir sampling
  - Stochastic Light Culling, Tokuyoshi and Harada, 2016

- Importance resampling + Weighted reservoir sampling
  - Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting, Bitterli et al., 2020

WORLD-SPACE SPATIOTEMPORAL RESERVOIR REUSE

RESERVOIR-BASED SPATIOTEMPORAL IMPORTANCE RESAMPLING (RESTIR)

- Reservoir-based resampling is a great solution for light sampling
  - Allows to efficiently sample from thousands of light sources
  - GPU-friendly streaming approach makes for good performance
  - No pre-processing == Compatible with dynamic scenes

- Unfortunately, the original approach does not scale to sampling lights at secondary path vertices
  - Screen-space reservoir reuse
  - Limited to path vertices directly visible from camera

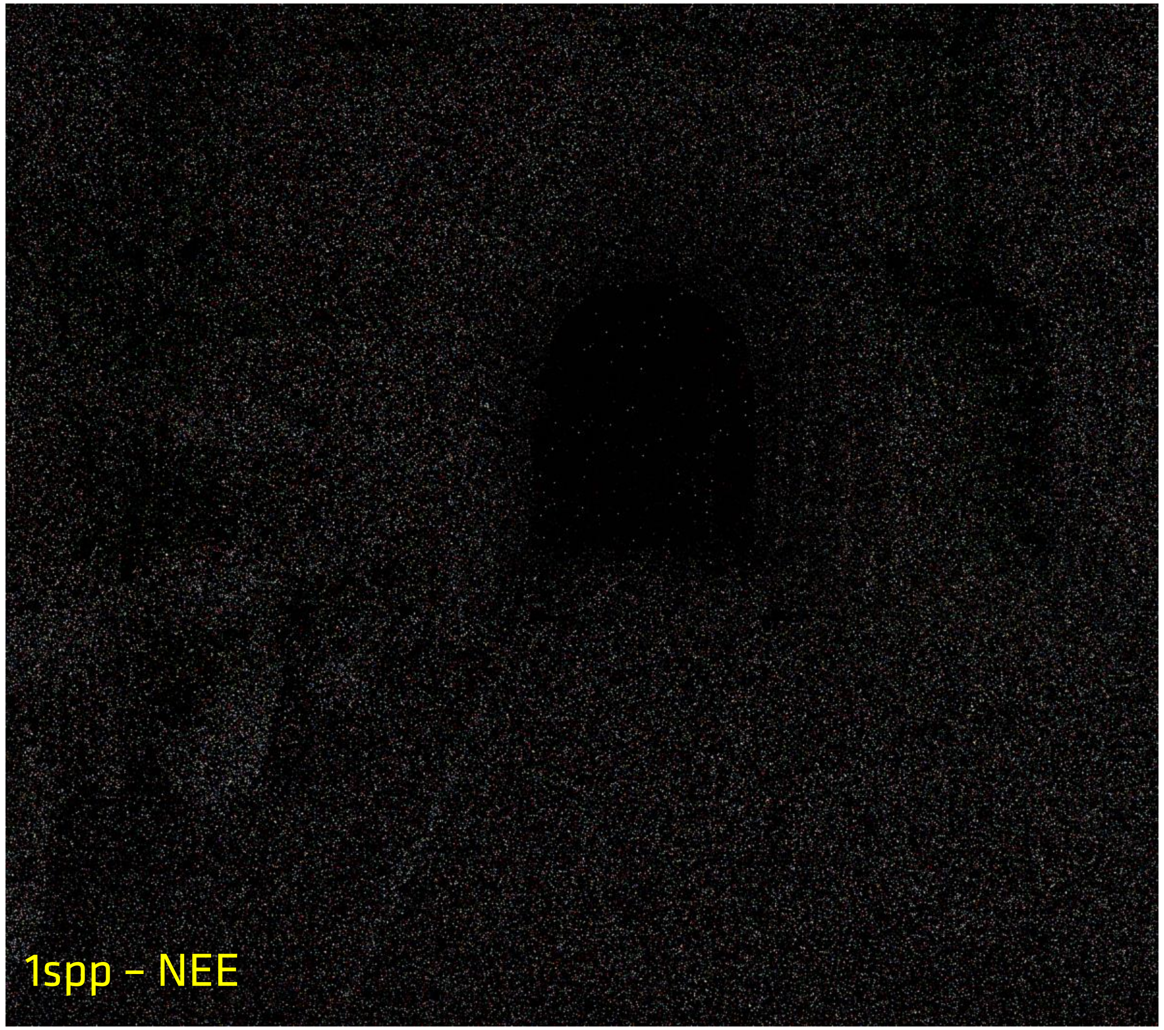
- We extended the reservoir reuse using a world-space structure

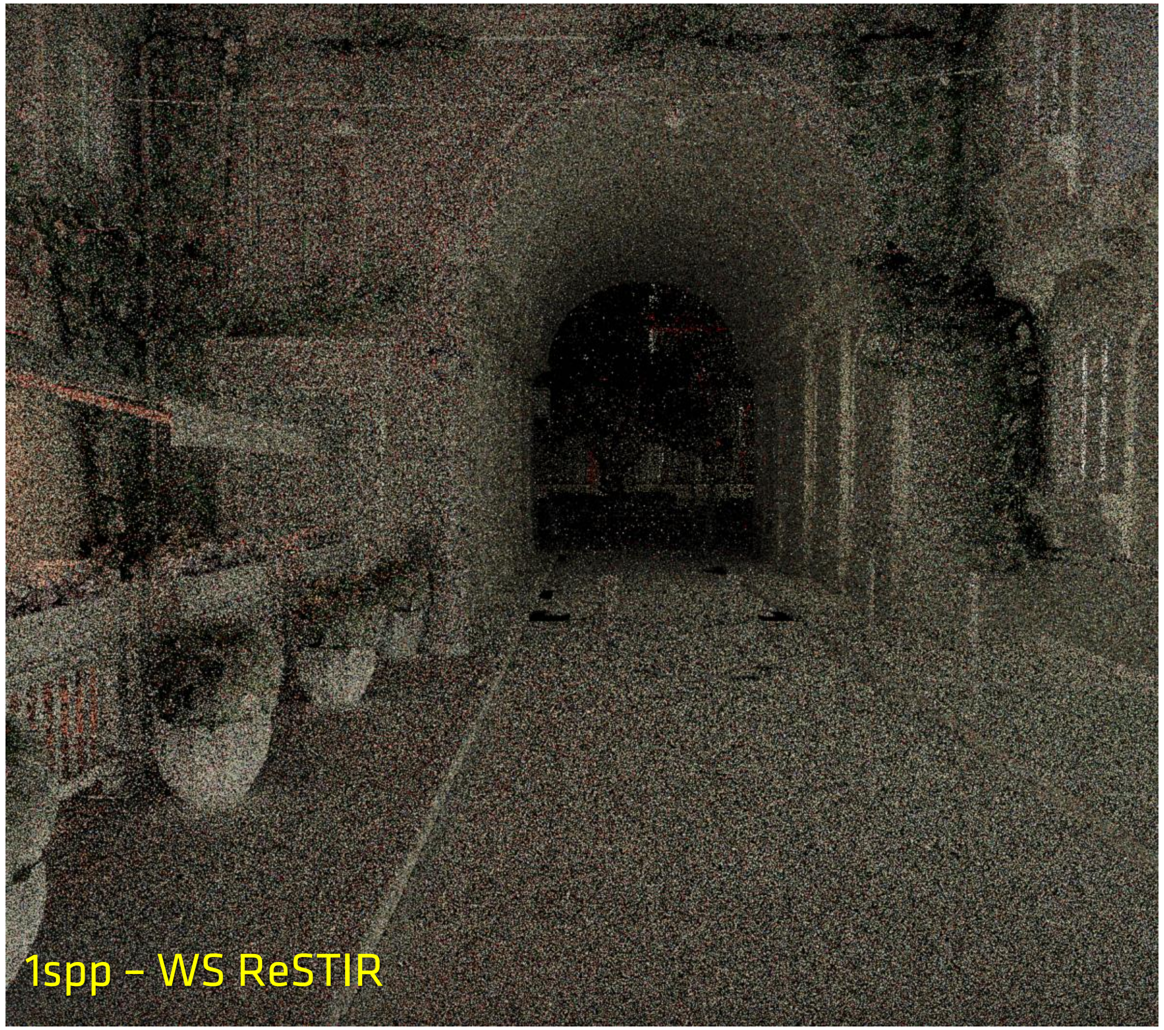
WORLD-SPACE SPATIOTEMPORAL RESERVOIR REUSE

- Adaptive world-space reservoir cache using spatial hashing
  - Quantize the world-space position based on an adaptive cell size and hash it
  - Conflicts can be resolved using linear probing
  - Create a list of reservoirs for every hash cell

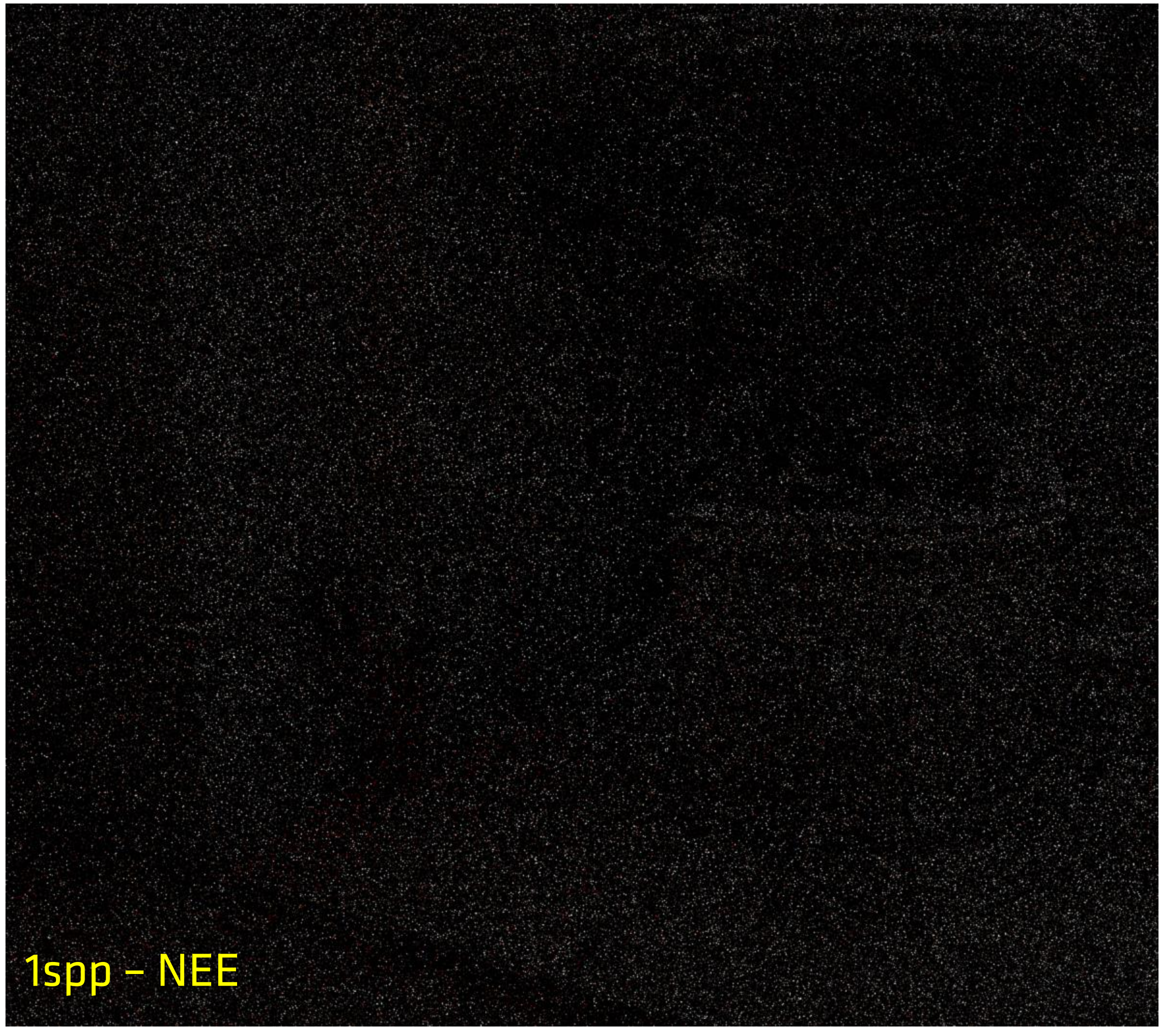
- This allows to efficiently resample from many neighbor reservoirs
  - We find nearby reservoirs in a single lookup through jittering
  - Reservoir resampling available anywhere in world space
  - Neighbor path vertices can share the effort in finding the best light sample!

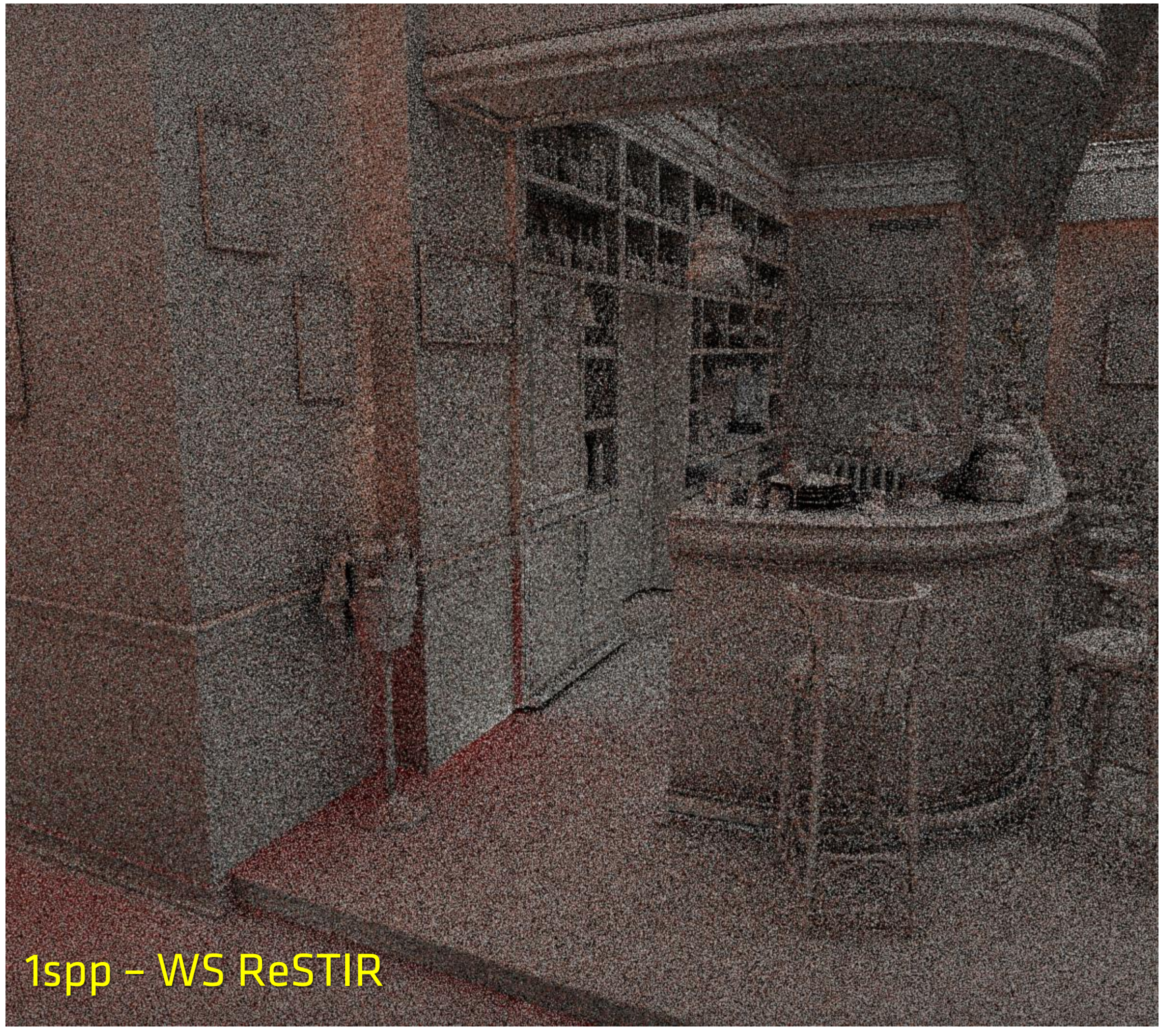
RESULTS – 30,672 AREA LIGHTS





RESULTS – 37,436 AREA LIGHTS



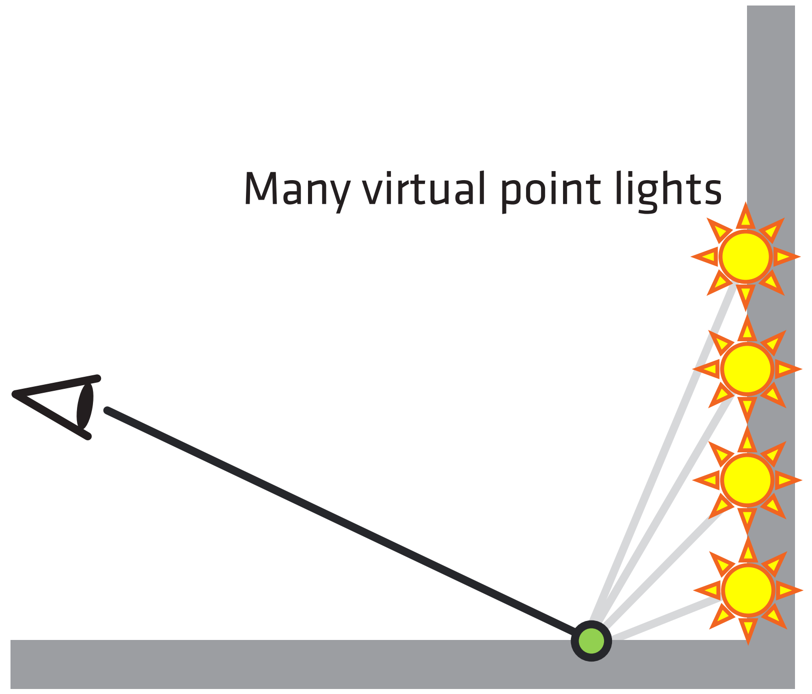


World-Space Spatiotemporal Reservoir Reuse for Ray-Traced Global Illumination, Guillaume Boissé, SIGGRAPH Asia 2021, 2021

TILED RESERVOIR SAMPLING FOR MANY-LIGHT RENDERING

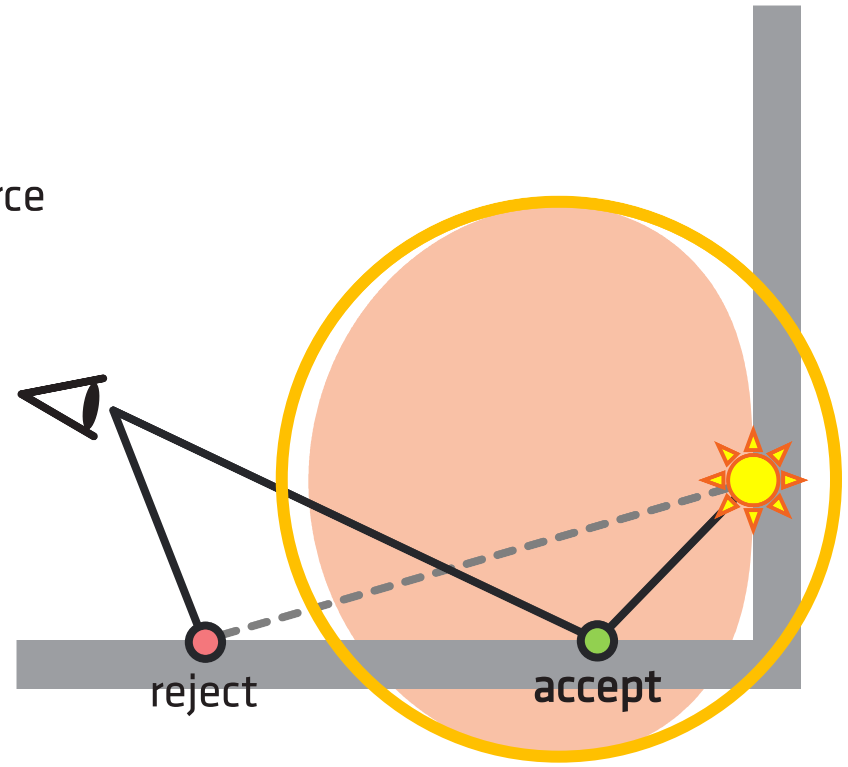
CANDIDATE SAMPLES FOR RESERVOIR SAMPLING

- Quality of reservoir sampling depends on initial candidate samples
- Uniform sampling of tens of candidates is insufficient ☹
  - Especially for lights close to surfaces (e.g., virtual point lights)
- Need an **efficient sampling technique for candidates**



STOCHASTIC LIGHT CULLING FOR MANY CANDIDATES

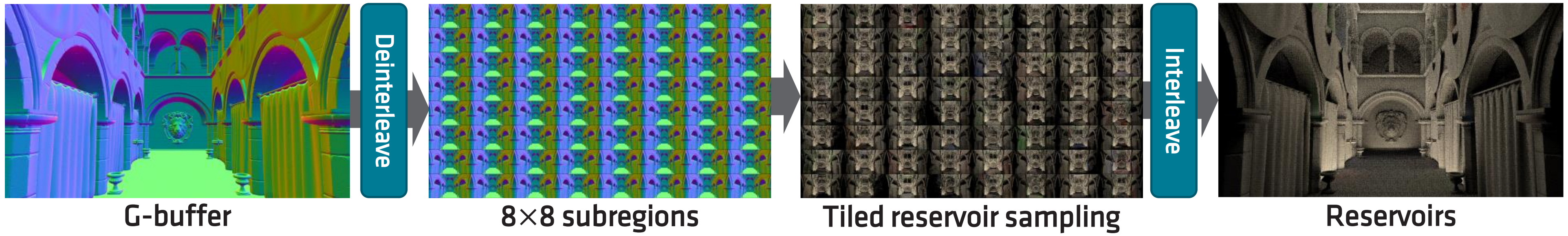
1. Generate 1024 candidates/tile
2. Apply **stochastic light culling** [Tokuyoshi and Harada 2017] for each tile
   - Tiled culling using random light ranges
   - Acts as Russian roulette according to the fall-off of each light source
3. Perform reservoir sampling for survived candidates



Y. Tokuyoshi and T. Harada. 2017. Stochastic Light Culling for VPLs on GGX Microsurfaces. Comput. Graph. Forum 36, 4, 55–63
Y. Tokuyoshi. 2021. Tiled Reservoir Sampling for Many-Light Rendering. AMD Tech. Rep, No. 21-11-ecdc

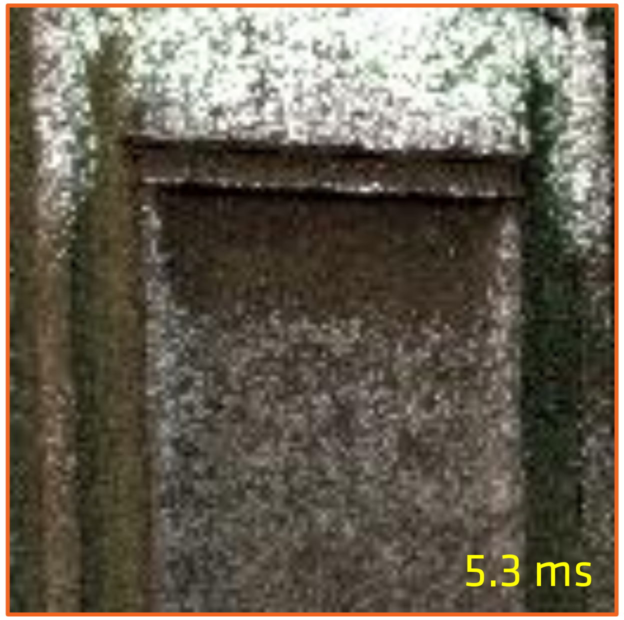
DECORRELATION OF VARIANCE

- Stochastic light culling can produce a positive correlation of variance between adjacent pixels
- Decorrelate the variance by **deinterleaving pixels** [Segovia et al. 2006]
  - Split 65536 candidates into 8x8 deinterleaved screen subregions (1024 candidates/subregion)
  - **Coherent memory access** in each tile ☺
  - Randomization of deinterleaving pixels further decorrelates the variance

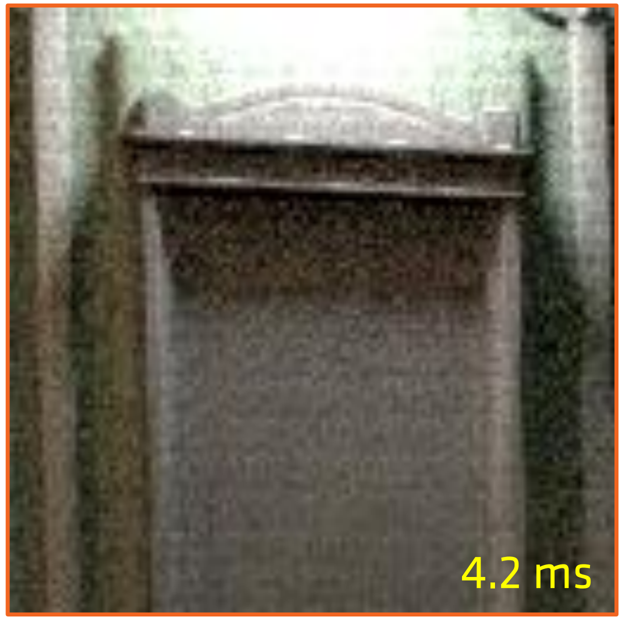


B. Segovia, J-C. Iehl, R. Mitanchey, and B. Péroche. 2006. Non-Interleaved Deferred Shading of Interleaved Sample Patterns. In GH' 06. 53–60

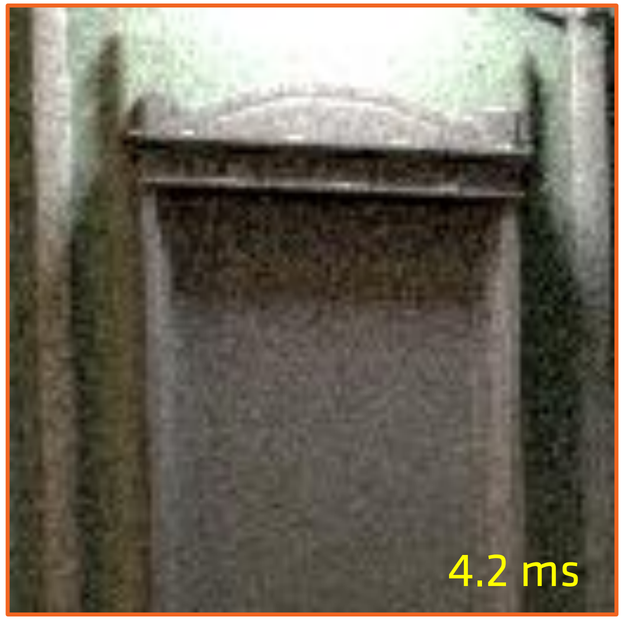
RESULTS



Previous



Ours



Ours
(+ random deinterleaving)

Y. Tokuyoshi. 2021. Tiled Reservoir Sampling for Many-Light Rendering. AMD Tech. Rep, No. 21-11-ecdc

STOCHASTIC LIGHT CULLING FOR VOLUME RENDERING

INTRO

- Volume rendering is essential to add realism to a rendered image
  - Light shafts, Fog effect
  - Volume scattering on primary ray is more visible compared to scattering on deeper paths
- A tree-based light sampling is often used in rendering with many lights
  - Need to maintain a tree for volume scattering

- Is there a lightweight light sampling technique?
  - Specialized for volume scattering on primary rays

LIGHT CULLING

- Light culling
  - Popular in real time
  - We can cull lights to reduce the computational cost ☺
  - Introduces darkening bias, not usable ☹

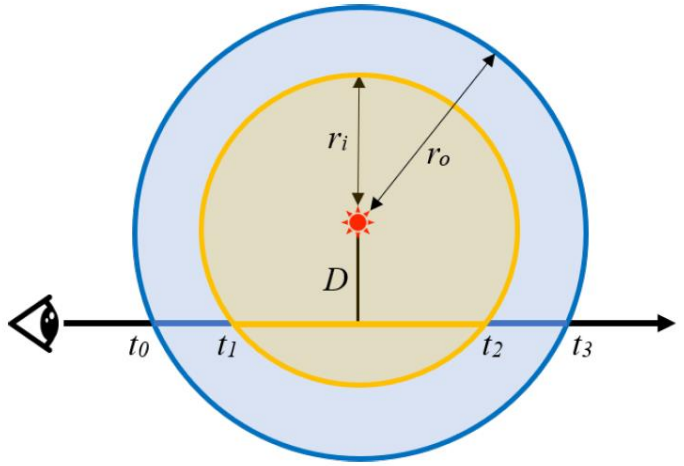
- Stochastic light culling
  - Give lights a finite range sampled stochastically
  - We can cull lights to reduce the computational cost ☺
  - Does not make it dark, converges with enough samples ☺
  - Proposed for surface lighting
  - Applicable for volume scattering?

STOCHASTIC LIGHT CULLING FOR VOLUMES

- Not a sphere-point intersection
- Need ray-sphere intersection

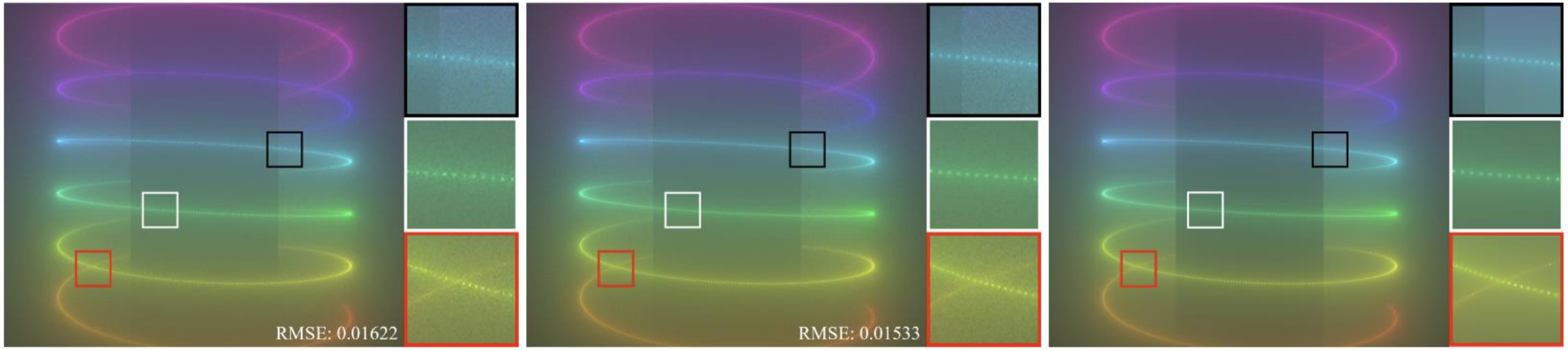
- Define 2 bounding spheres in the range of fall-off function
  - Inner sphere for the 100% acceptable range
  - Outer sphere for a stochastic range
- Formulate the importance calculation for 3 cases
  - No intersection
  - Intersect to the outer sphere but not to the inner sphere
  - Intersect to both spheres

- We can use screen-space tiled light culling
  - No additional data structure ☺



RESULTS

- Equal-time comparison in 1 second
- Improved convergence compared to baseline
  - Baseline: reservoir sampling to choose one light with equiangular sampling



(a) *Reservoir sampling [Cha82]* **(b)** *Our method + reservoir sampling* **(c)** *Reference*

Stochastic Light Culling for Single Scattering in Participating Media, S. Fujieda et al., Eurographics 2022 short paper

OROCHI

OROCHI - SUPPORT BOTH HIP AND CUDA WITH EASE

- Problem description
  - Need to maintain 2 backends, HIP and CUDA
    - Requires compilation of each backend separately
  - Need to explicitly link against HIP or CUDA (load-time linking)
  - Unnecessary complication developers need to think about

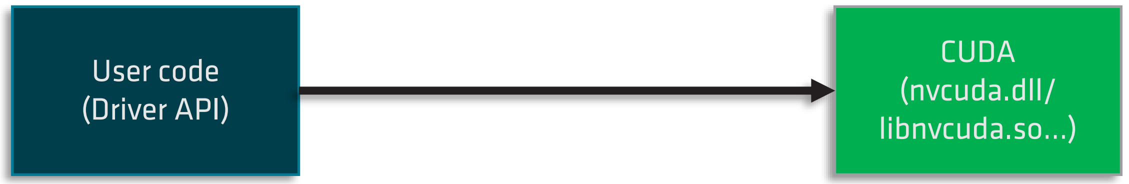
- Orochi is a solution for this
  - No linking required at compile-time, instead load APIs dynamically (runtime linking)
  - Query for devices
    - Returns # of devices you have (Can query multiple vendors (both AMD and NVIDIA GPUs))
  - Create a context on a device you like to use
    - No need to worry about which one is AMD, NVIDIA
  - Can compile your app without SDKs installed
  - Supports Windows and Linux

- Open source
  - https://github.com/GPUOpen-LibrariesAndSDKs/Orochi
    - Contributions are welcome!

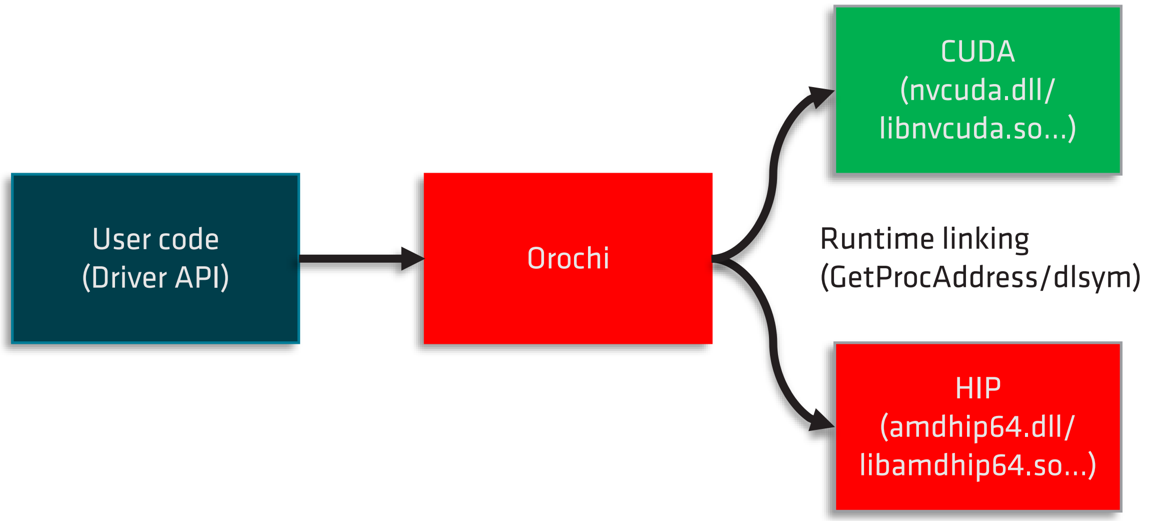
Yamata no Orochi (ヤマタノオロチ, 八岐大蛇) is a legendary eight-headed and eight-tailed Japanese dragon. Public domain image courtesy Wikipedia (https://en.wikipedia.org/wiki/Yamata_no_Orochi#/media/File:YamataNoOrochi.jpg)

FOR CUDA USER

- Now

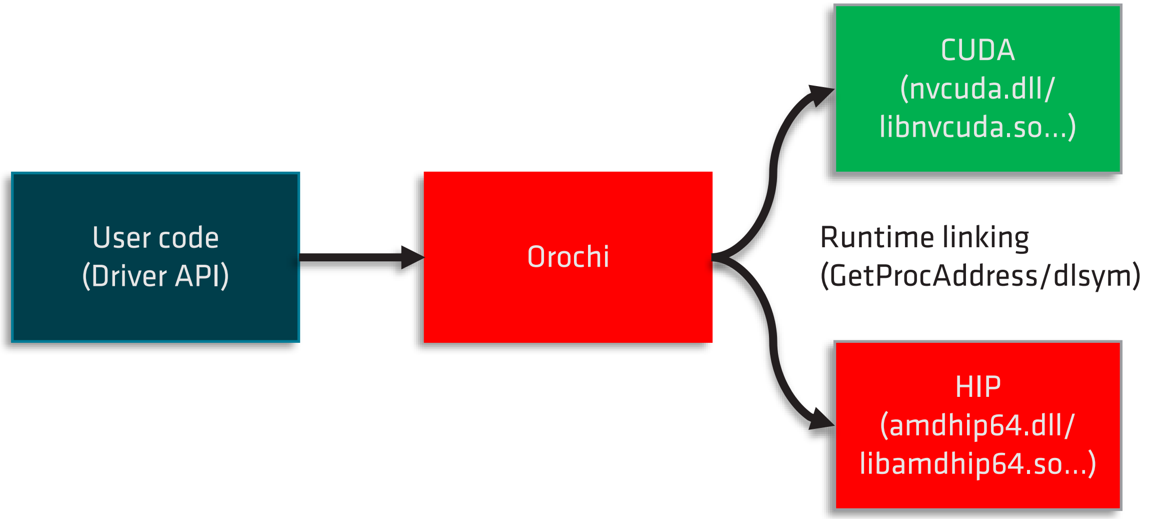


- With Orochi


User code (Driver API)
Orochi
CUDA (nvcuda.dll/ libnvcuda.so...)
Runtime linking (GetProcAddress/dlsym)
HIP (amdhip64.dll/ libamdhip64.so...)

ARCHITECTURE

- Architecture



- API example

```
#include <Orochi/Orochi.h>

{
  oroInitialize( ( oroApi )( ORO_API_CUDA | ORO_API_HIP ), 0 );

  e = oroInit( 0 );
  int nDevicesTotal;
  e = oroGetDeviceCount( &nDevicesTotal );
  int nAMDDevices;
  e = oroGetDeviceCount( &nAMDDevices, ORO_API_HIP );
  int nNVIDIADevices;
  e = oroGetDeviceCount( &nNVIDIADevices, ORO_API_CUDA );

  printf( "# of devices: %d\n", nDevicesTotal );
  printf( "# of AMD devices: %d\n", nAMDDevices );
  printf( "# of NV devices: %d\n\n", nNVIDIADevices );

  for( int i = 0; i < nDevicesTotal; i++ )
  {
    oroDevice device;
    e = oroDeviceGet( &device, i );
    char name[128];
    e = oroDeviceGetName( name, 128, device );

    oroCtx ctx;
    e = oroCtxCreate( &ctx, 0, device );
    ...
  }
}
```

Initialize Orochi
(load libraries for
both CUDA and HIP)

Replace hip... or cu...
with oro...

HIPRT

Ray tracing API in HIP

HIPRT

- RDNA™ 2 GPUs have HW-accelerated ray tracing
  - DXR and Vulkan APIs
- HIPRT
  - HW-accelerated ray tracing in HIP
- The first release in March 2022

HIPRT: RAY TRACING API IN HIP

- Ray Tracing Library
  - Ray tracing with bounding volume hierarchies (BVH)
  - Scalability (construction speed vs. ray tracing sped)
  - HW ray tracing acceleration
- General purpose
  - Offline and online rendering
  - Triangles, custom primitives, motion blur, dynamic geometry
- Cross-platform
  - AMD and Nvidia GPUs
  - Windows and Linux OSes
- Lightweight and Expressiveness
  - Written on top of drivers
  - Minimalistic setup
  - Yet providing flexibility and lower-level control
  - Designed for high performance
  - Thin driver layer
  - General purpose language based on modern C++ standards

API EXAMPLE

Host code:

```
// Triangle mesh
hiprtTriangleMeshPrimitive mesh;
mesh.triangleIndices = d_triIdxBuf;
mesh.vertices = d_vtxBuf;

// Create and build geometry
hiprtGeometry geom;
hiprtCreateGeometry( ..., &geom );
hiprtBuildGeometry( ..., geom );


// Build trace kernel
hiprtBuildTraceKernel( ... );
```

- Device Code

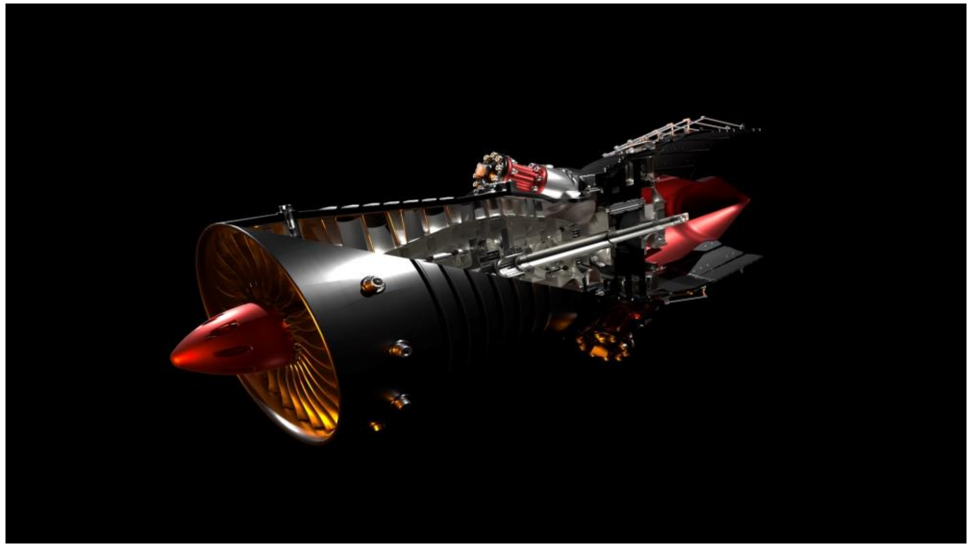
```
__global__ void RayTraceKernel(
    hiprtScene     scene,
    ... )
{
    ...


    hiprtRay ray;
    generateRay( ..., &ray.origin, &ray.direction );
    ray.maxT = worldExtent;


    hiprtSceneTraversalClosest tr( scene, ray, 0xffffffff );
    hiprtHit hit = tr.getNextHit();
```

RADEON™ PRORENDER

RECENT FEATURE ADDITION

- Toon shader, contour render
  - Mix non-photo realistic in photo-real rendering
- Colored shadow
  - Control shadow color
- Cutting plane
  - See inside without modifying geometry
- Atmospheric volume
  - Faster way to realize light shafts
- Primitive variables
  - Complex material with simple set up
- Enhanced Cryptomatte AOVs
  - Captures secondary visibilities too
- Flexible ramp node
  - Material network can be used as an input
- HIP support
  - New backend in HIP

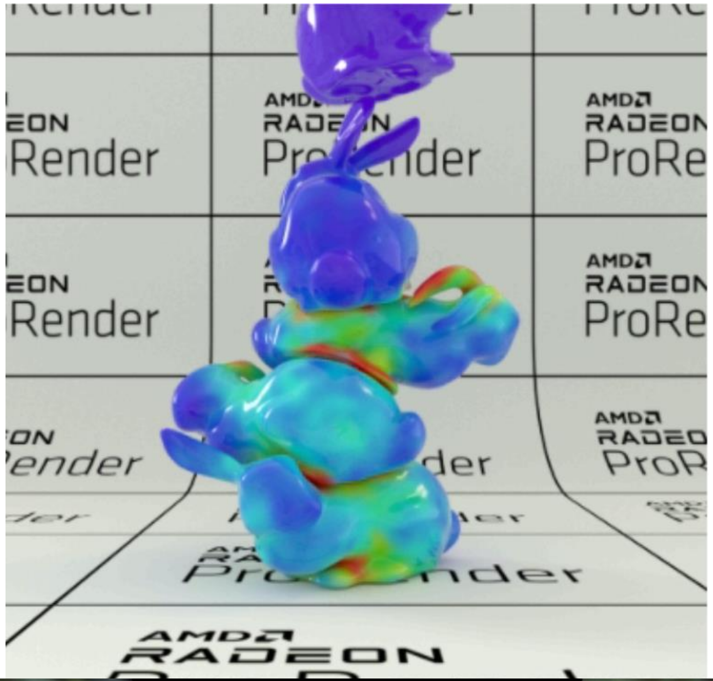
Yohsuke Nakano @ Morgenrot inc.

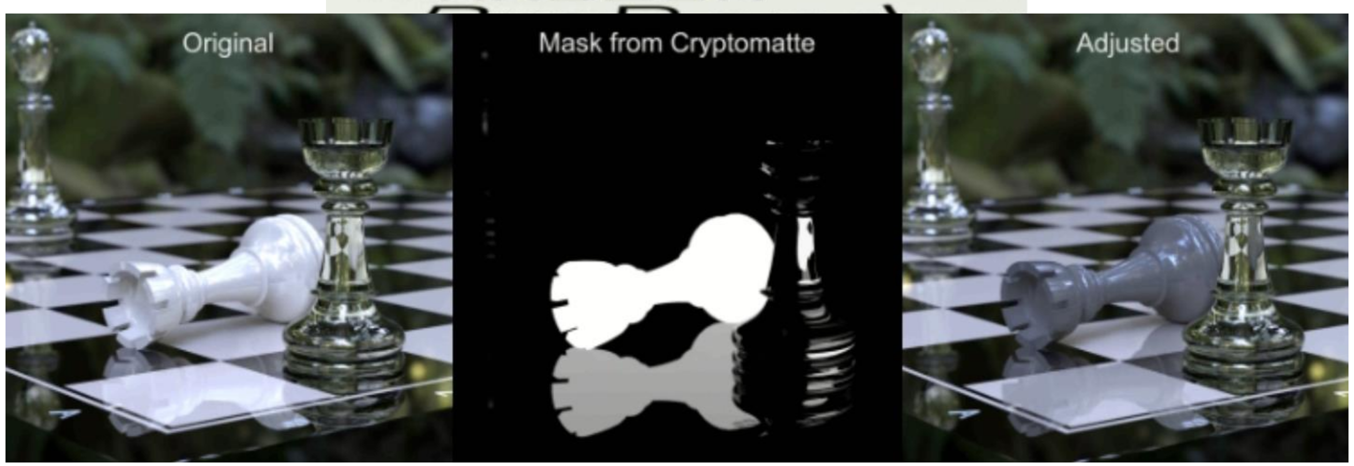


RECENT FEATURE ADDITION

- Toon shader, contour render
  - Mix non-photo realistic in photo-real rendering
- Colored shadow
  - Control shadow color
- Cutting plane
  - See inside without modifying geometry
- Atmospheric volume
  - Faster way to realize light shafts
- Primitive variables
  - Complex material with simple set up
- Enhanced Cryptomatte AOVs
  - Captures secondary visibilities too
- Flexible ramp node
  - Material network can be used as an input
- HIP support
  - New backend in HIP

https://gpuopen.com/learn/radeon-prorender-2-02-10/
https://gpuopen.com/learn/introducing-prorender-sdk-2-02-11/



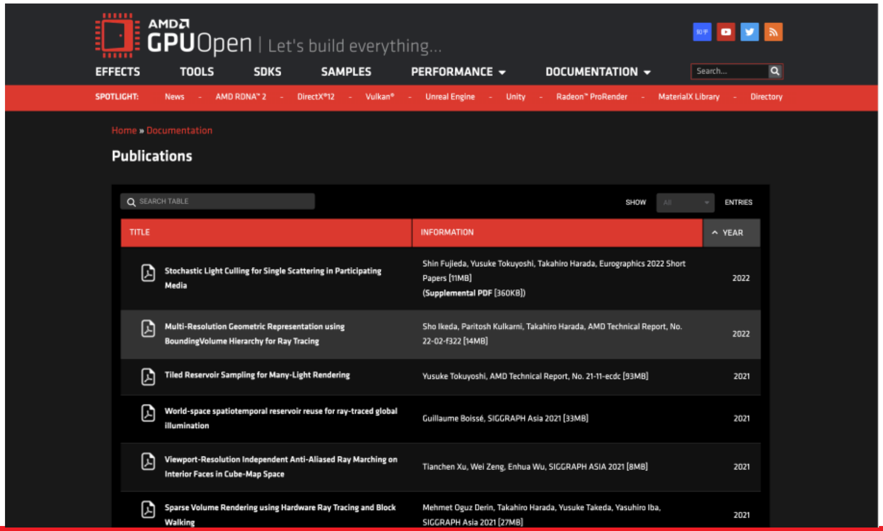


CONCLUSION

- Presented some ray tracing related R&Ds done in AMD
  - Geometric representation
  - Improvements in importance resampling techniques
  - Volume sampling
  - Orochi
  - HIPRT
  - Radeon ™ ProRender

REFERENCES

- Multi-Resolution Geometric Representation using Bounding Volume Hierarchy for Ray Tracing, Sho Ikeda, Paritosh Kulkarni, Takahiro Harada, AMD Technical Report, No. 22-02-f322, 2022
- Tiled Reservoir Sampling for Many-Light Rendering, Yusuke Tokuyoshi, AMD Technical Report, No. 21-11-ecdc, 2021
- World-Space Spatiotemporal Reservoir Reuse for Ray-Traced Global Illumination, Guillaume Boissé, SIGGRAPH Asia 2021, 2021
- Stochastic Light Culling for Single Scattering in Participating Media, Shin Fujieda, Yusuke Tokuyoshi, Takahiro Harada, Eurographics 2022 Short Papers, 2022
- https://gpuopen.com/learn/publications/



AMD GPUOpen | Let's build everything...
EFFECTS
TOOLS
SDKS
SAMPLES
PERFORMANCE
DOCUMENTATION
Search...
SPOTLIGHT: News - AMD RDNA™ 2 - DirectX®12 - Vulkan® - Unreal Engine - Unity - Radeon™ ProRender - MaterialX Library - Directory
Home » Documentation
SEARCH TABLE
SHOW All ENTRIES

Publications

TITLE	INFORMATION	YEAR
Stochastic Light Culling for Single Scattering in Participating Media	Shin Fujieda, Yusuke Tokuyoshi, Takahiro Harada, Eurographics 2022 Short Papers [11MB] (Supplemental PDF [360KB])	2022
Multi-Resolution Geometric Representation using BoundingVolume Hierarchy for Ray Tracing	Sho Ikeda, Paritosh Kulkarni, Takahiro Harada, AMD Technical Report, No. 22-02-f322 [14MB]	2022
Tiled Reservoir Sampling for Many-Light Rendering	Yusuke Tokuyoshi, AMD Technical Report, No. 21-11-ecdc [93MB]	2021
World-space spatiotemporal reservoir reuse for ray-traced global illumination	Guillaume Boissé, SIGGRAPH Asia 2021 [33MB]	2021
Viewport-Resolution Independent Anti-Aliased Ray Marching on Interior Faces in Cube-Map Space	Tianchen Xu, Wei Zeng, Enhua Wu, SIGGRAPH ASIA 2021 [8MB]	2021
Sparse Volume Rendering using Hardware Ray Tracing and Block Walking	Mehmet Oguz Derin, Takahiro Harada, Yusuke Takeda, Yasuhiro Iba, SIGGRAPH Asia 2021 [27MB]	2021

DISCLAIMER & ATTRIBUTION

AMD GPUOpen

EPYC RADEON RYZEN AMD