NVIDIA RTX in Remedy Northlight

Juha Sjöholm
Senior Devtech Engineer
Helsinki

Remedy Entertainment

- Game studio based in Finland
  - Founded in 1995
  - 185 employees
- Best known for
  - Quantum Break
  - Alan Wake
  - Max Payne
- Upcoming
  - Control
  - CrossFire 2 (Story Mode)
- Northlight engine

RTX Work at Remedy

- RTX ray tracing experiments started in 2017
- Exploring new possibilities
- Northlight RTX demo shown at GDC 2018
  - https://www.remedygames.com/experiments-with-directx-raytracing-in-remedys-northlight-engine/
- DXR - DirectX Raytracing API
- RTX support announced for Control at Gamescom 2018
  - Details will follow
- Work continues with Turing

Agenda

- RTX integration
- Shadows
  - Sun
  - Contact
- Reflections
  - G-buffer
  - Transparent surfaces
- Indirect Diffuse Illumination

Hybrid Rendering

Combining Ray Tracing and Rasterization

- Enhance an existing rasterization based rendering pipeline.
- Resolve primary visibility through rasterization.
- Evaluate one or more effects related to lighting through ray tracing.
  - Reflections
  - Indirect Diffuse Illumination
  - Shadows
  - Ambient Occlusion

RTX Integration in Northlight

- Ray tracing acceleration structures
- Ray tracing pipeline states
- Ray tracing shader tables

Acceleration Structures

Bottom Level Structures

- Separate bottom level built for each geometry LOD.
- LOD selection for ray tracing matches LOD selection for G-buffer.

Acceleration Structures

- Separate bottom level built for each geometry LOD.
- LOD selection for ray tracing matches LOD selection for G-buffer.
- Utilize existing mesh instancing logic.
- Mesh piec▫tg

Acceleration Structures

Skinned Meshes

- On each frame, run a CS that outputs the skinned vertex data.
  - Each vertex is processed once. Indices are not touched.
- Update bottom level structure.
  - Rebuild on every Nth frame.
  - Update always may work for non-destructibles.
- Skip update if skinning matrices have not been touched.

Bottom Level Build Flags

1. PREFER_FAST_TRACE - Non-deformable geometries
2. PREFER_FAST_BUILD | ALLOW_UPDATE - Deformable objects

Bottom Level Build Flags

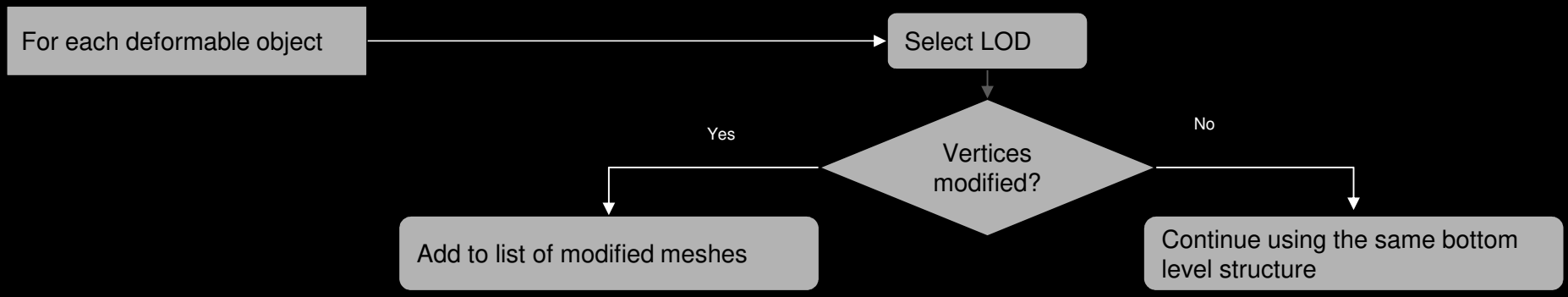
1. PREFER_FAST_TRACE - Non-deformable geometries
2. PREFER_FAST_BUILD | ALLOW_UPDATE - Deformable objects
3. PREFER_FAST_TRACE | ALLOW_UPDATE - Hero characters
4. PREFER_FAST_BUILD - F !!" #h"s$cs %ase& &e'or (a%!es) *#re&$cta%!e (o+e (e*t

Bottom Level Build Flags

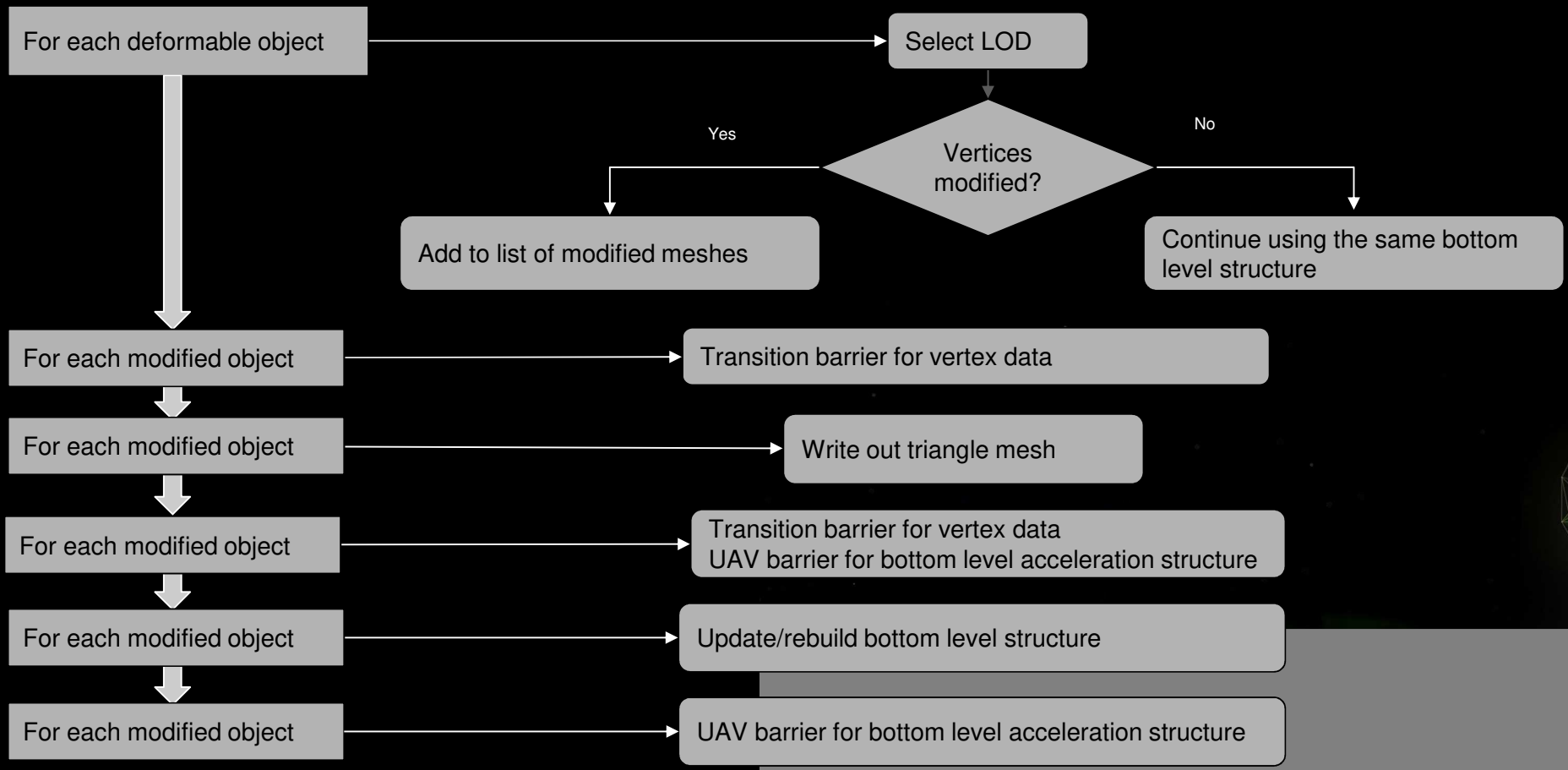
1. PREFER_FAST_TRACE - Non-deformable geometries
2. PREFER_FAST_BUILD | ALLOW_UPDATE - Deformable objects
3. PREFER_FAST_TRACE | ALLOW_UPDATE - Hero characters
4. PREFER_FAST_BUILD - F !!" #h"s$cs %ase& &e'or (a%!es) *#re&$cta%!e (o+e (e*t

- If not alpha tested, FORCE_OPAQUE flag in top level instance
- MINIMIZE_MEMORY not used

Resource Barriers



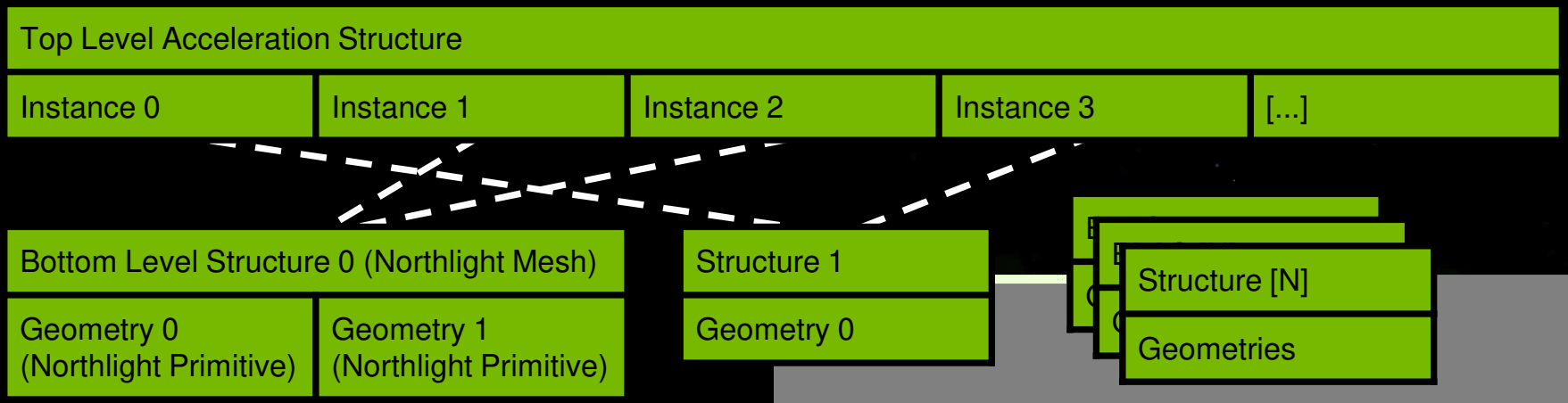
Resource Barriers



Acceleration Structures

Top Level Structure

- Simply rebuild top level on each frame.
  - Best for ray tracing performance.
- Share a bottom level structure when possible.



Pipeline States

New Concepts

- Small extension to engine effect file format.
- Separate pipeline state for each DispatchRays().
  - Optimal max values for recursion depth and payload size.
- A handful of permutations only.
  - No application side state object caching.
  - Shaders precompiled to DXIL.
  - Could still utilize collections.

Pipeline States

Shader Libraries

- Ray tracing shaders compiled as libraries. (“lib_6_3” target)
- Large existing shader codebase.
  - “static” keyword not used -> Every function is an export.

Pipeline States

Shader Libraries

- Ray tracing shaders compiled as libraries. (“lib_6_3” target)
- Large existing shader codebase.
  - “static” keyword not used -> Every function is an export.

- **/exports** to limit exports.
- **/auto-binding-space** to enable automatic register assignment.

Shader Tables

Global and Local Root Tables

- Use the global root table for almost everything.
- Compile all shaders in a pass as a single library.
- Ray generation and miss shaders use only the global root table.
- Hit groups use a couple special bindings through the local root table.

Global root table Shared for all shaders in a pass
Sampler table
CBV table SRV table UAV table

Local root table for hit groups Extends the global table
Root SRVs
Root constants

Shader Tables

Local Bindings for Hit Shaders

- Root SRVs to index and vertex buffers
- Some root constants
  - Strides for the root SRVs
  - Material id
- No references to descriptor heap

Shader Tables

Local Bindings for Hit Shaders

- Root SRVs to index and vertex buffers
- Some root constants
  - Strides for the root SRVs
  - Material id
- No references to descriptor heap

- Root SRV issues
  - No check for out-of-bounds access
  - No format conversions
  - No check for base address alignment

Shader Tables

Root SRVs and Constants for Vertex Data

- Simple, fast
- ByteAddressBuffers with dynamic attribute offsets -> no permutations

```
struct HitConstants {uint uVertexStride; uint uUVOffset; […]};
ConstantBuffer<HitConstants> g_bHitConstants : register(b0, space3);

ByteAddressBuffer g_bIndexBuffer : register(t0, space3);
ByteAddressBuffer g_bVertexBuffer : register(t1, space3);
```

Shader Tables

Bindless Access to Materials

Remedy already had “bindless” access to materials.

- Material constants in a structured buffer
- Textures in an unbounded array

```
struct HitConstants {[..], uint uMaterialID; };
ConstantBuffer<HitConstants> g_bHitConstants : register(b0, space3);

struct MaterialConstants {float fRoughnes;[…]};
StructuredBuffer<MaterialConstants> g_bMaterialConstants;

Texture2D g_tMaterialTextures[] : register(t0, space1);
```

Shader Tables

Layout

Ray generation ID	Ray type 0 Miss ID	Ray type 1 Miss ID	Geometry 0 Ray type 0 Hit ID	Geometry 0 Ray type 1 Hit ID	Geometry 1 Ray type 2 Hit ID	Geometry 1 Ray type 2 Hit ID	[...]
			Index SRV	Index SRV	Index SRV	Index SRV	[...]
			Vertex SRV	Vertex SRV	Vertex SRV	Vertex SRV	[...]
			Constants	Constants	Constants	Constants	[...]

- All other bindings come from the global root table.

Shadows

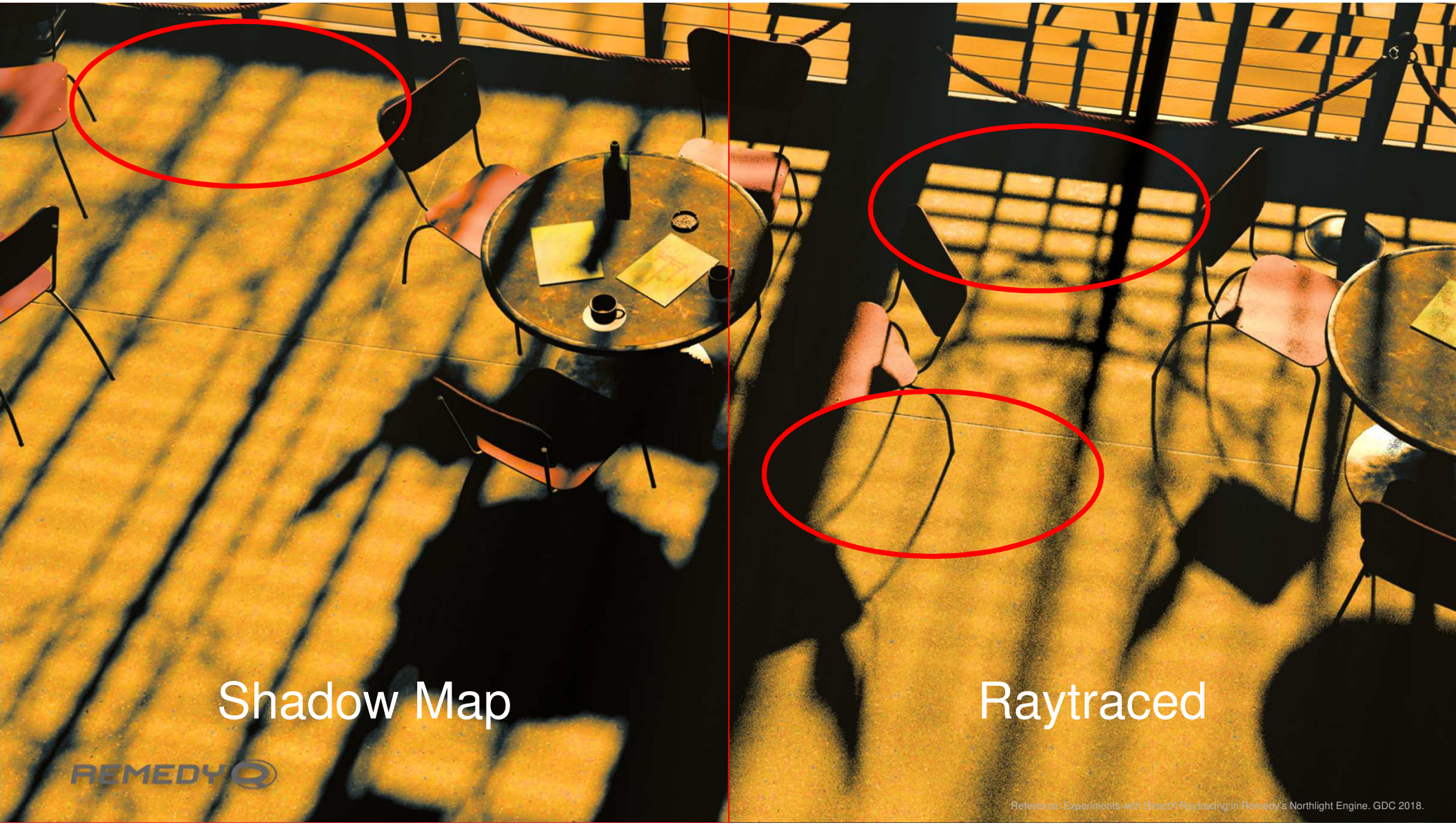
Sun

- Replacement for cascaded shadow maps.
- Very convenient when screen space shadow mask is produced anyway.

- Alpha test in Any Hit Shader
- RAY_FLAG_ACCEPT_FIRST_HIT_AND_END_SEARCH

GeForce RTX 2080 TI, 1920x1080

0.9 ms (2 rpp)



Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Shadows

Contact Shadows

- Use ray tracing to enhance shadow maps.
- Perfect details for the most influential lights.

1. Select the lights for each pixel.
2. Raytrace screen space shadow masks.
   a. You can use short (fast) rays.
3. Multiply with shadow map.

GeForce RTX 2080 TI, 1920x1080

1.4 ms (2 rpp, denoising)

Reflections

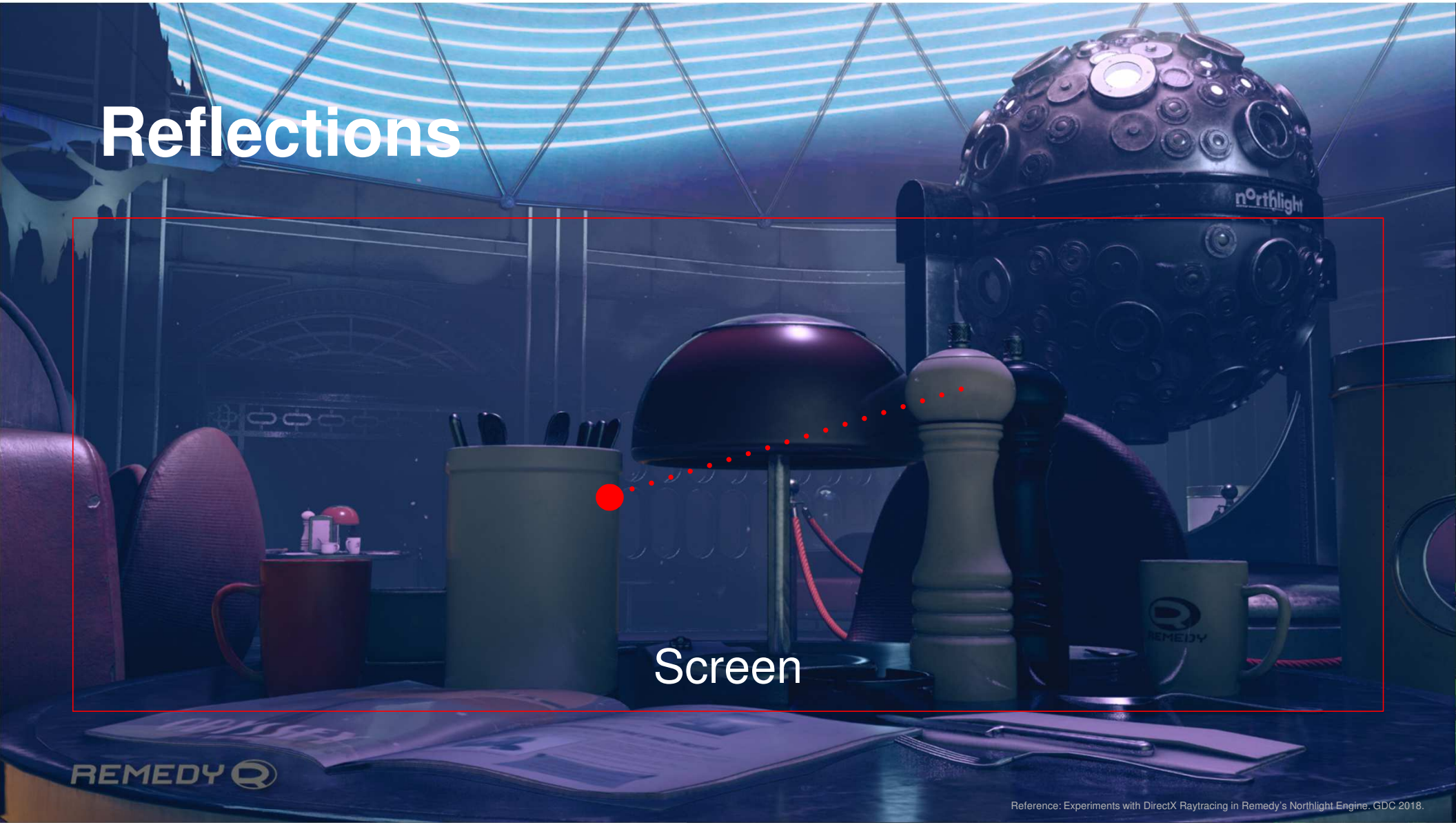
G-Buffer

- Reconstruct position from rasterized depth.
- Evaluate reflection direction based on surface normal.
- Randomize based on material properties (roughness).
- Direct replacement for screen space reflections.

Reflections

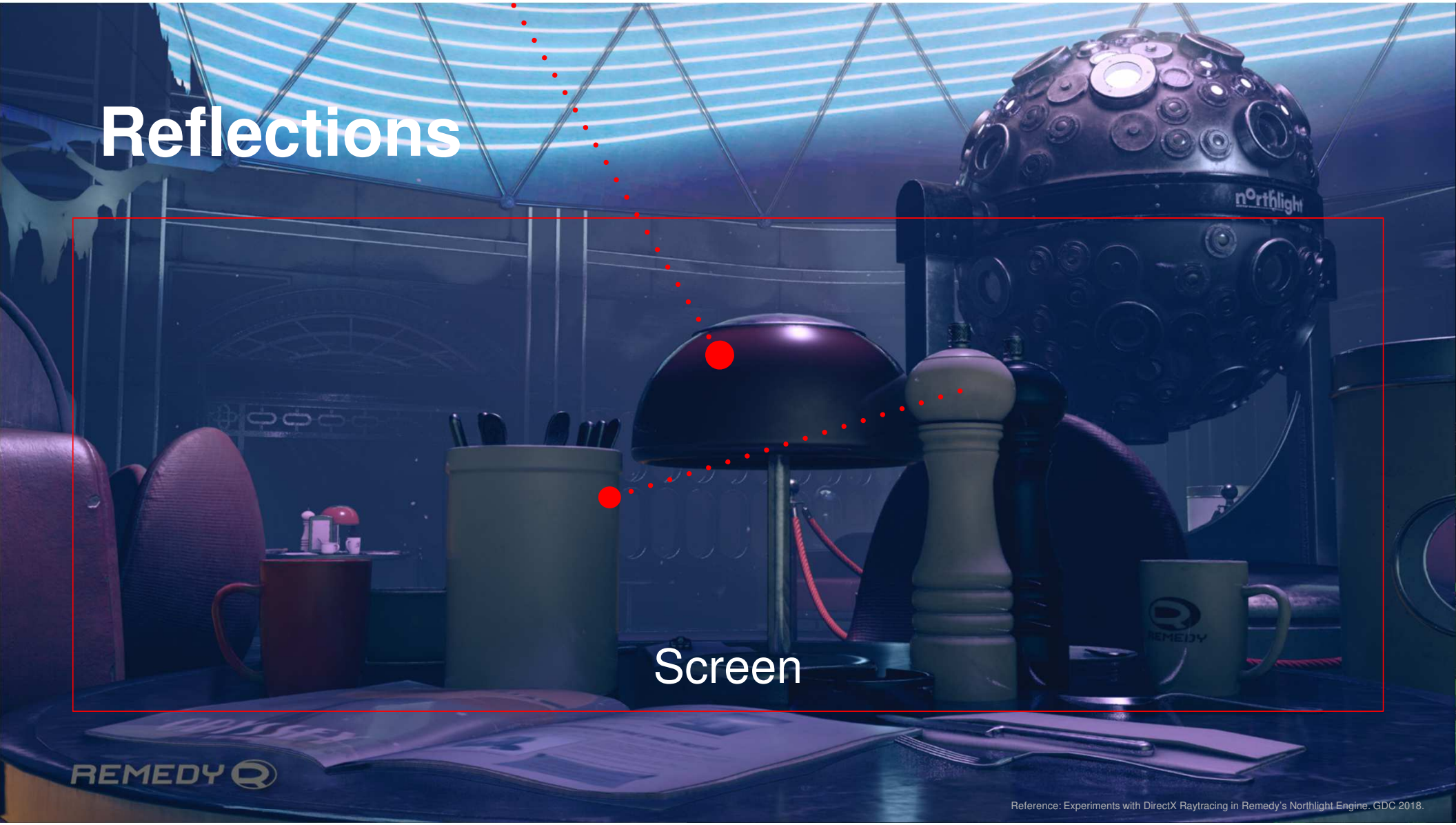
Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine, GDC 2018.

Reflections



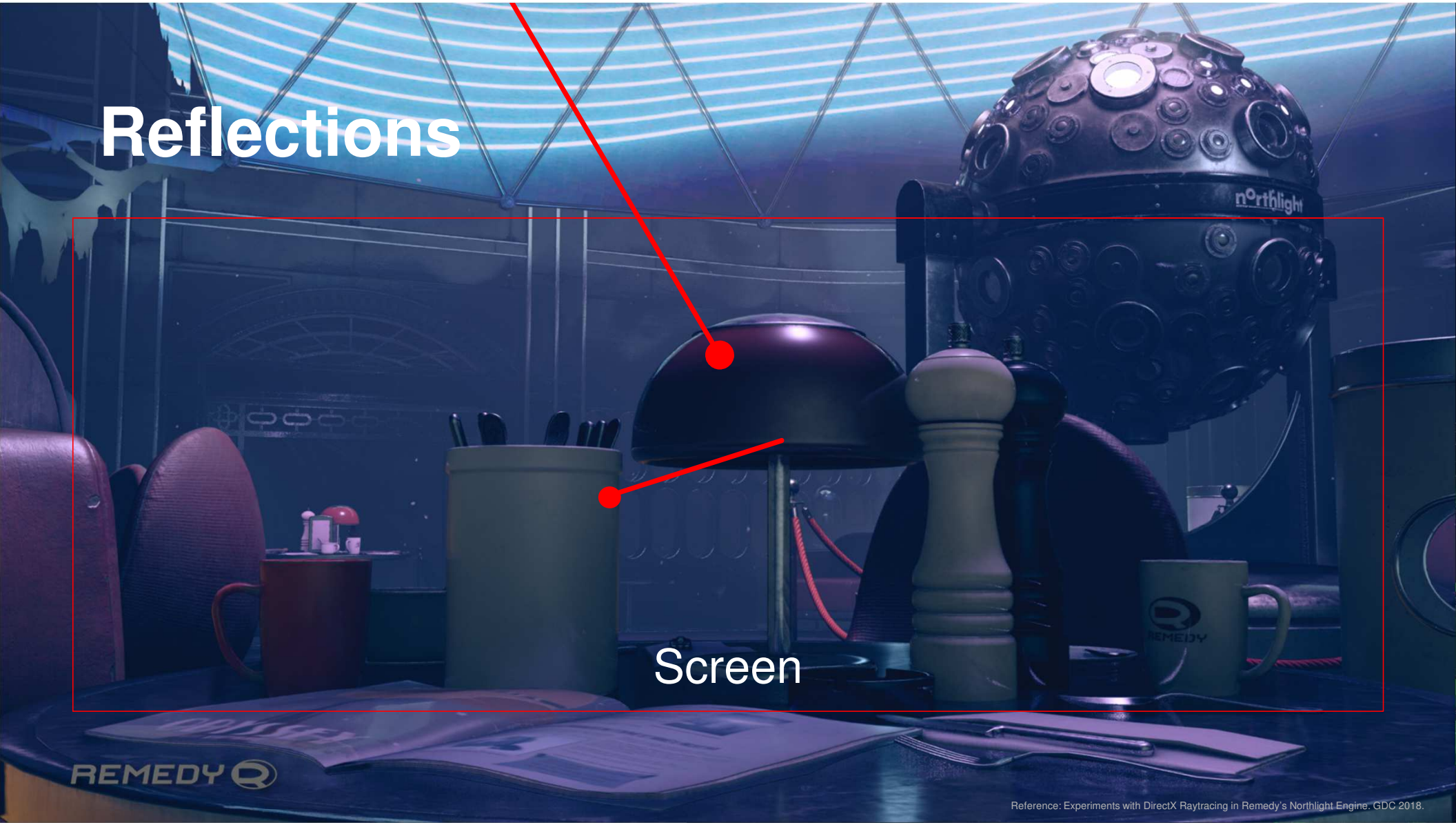
Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Reflections



Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Reflections

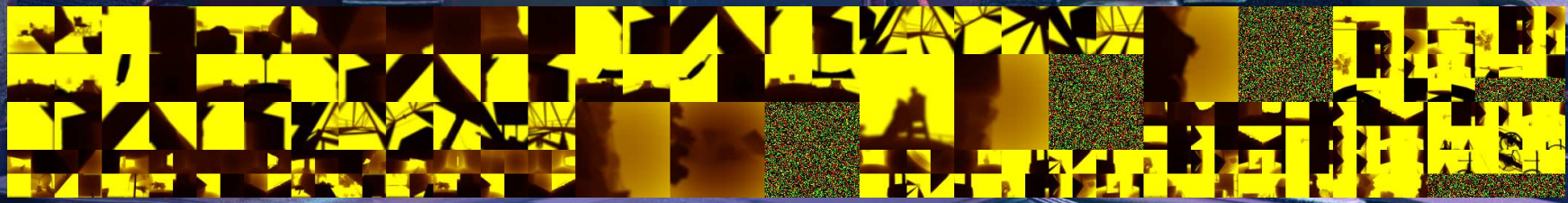


Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Reflections

Lighting Data

```
struct DeferredLightingPointData
{
    m::Vector3      vPositionInView;
    float           fShadowMapShrink;
    m::Vector3      vColor;
    float           fClipRange;
    m::Vector4      vFalloff;
    m::Vector3      vDirectionInView;
    float           fInvShadowMapRange;
    m::Matrix4      mViewToShadowClip;
    m::Vector4      vShadowAtlasOffsetScale;
    float           fRadius;
    float           fBoundsRadius;
    float           fScatterIntensityMul;
    unsigned int    uTechniqueProperties;
};
```



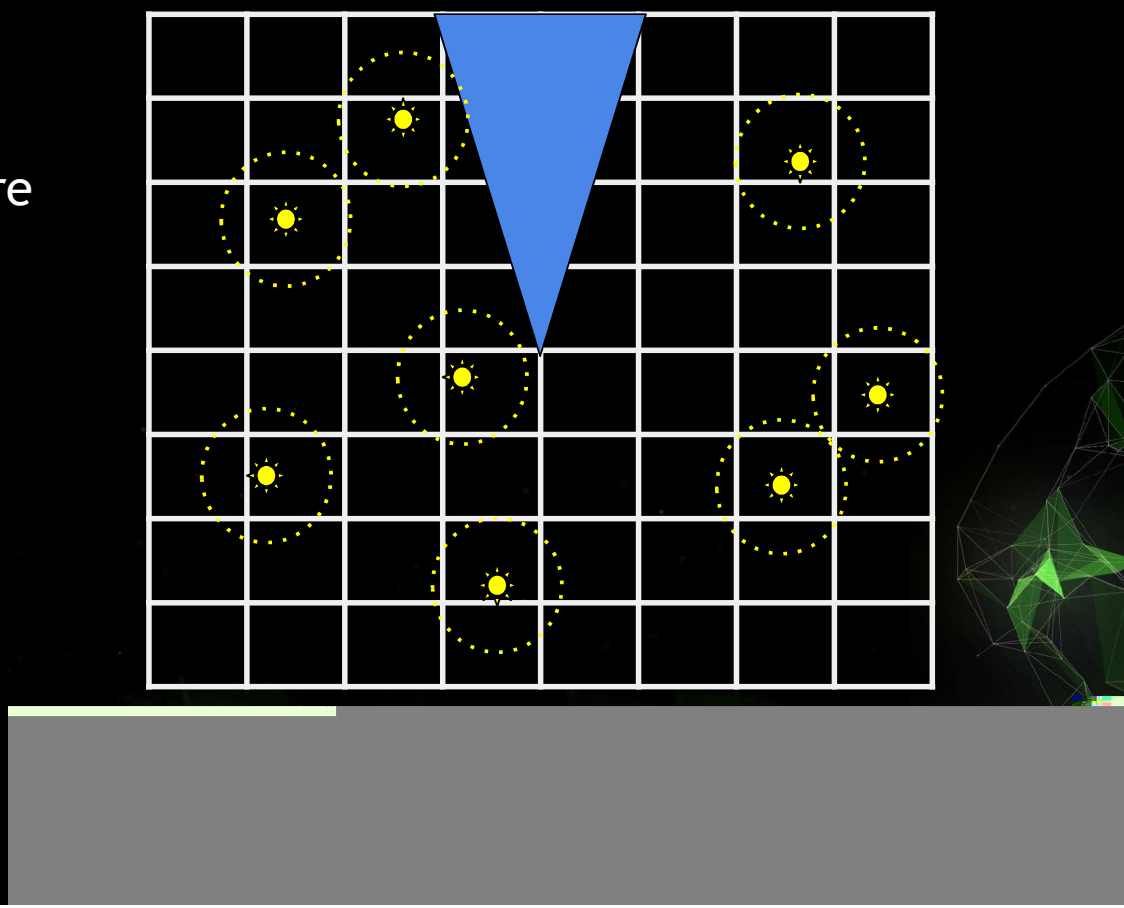
REMEDY

Reflections

Light Culling

- Lots of lights
- Reflected location can be anywhere

- View space clustering
  - Fast
  - Good speedup for reflection



Reflections

- Most shadows with shadow maps
- Sample precomputed GI on miss
- Texture LOD 0.0f
- Unified shading model

GeForce RTX 2080 TI, 1920x1080

4.4 ms (1 rpp, denoising)

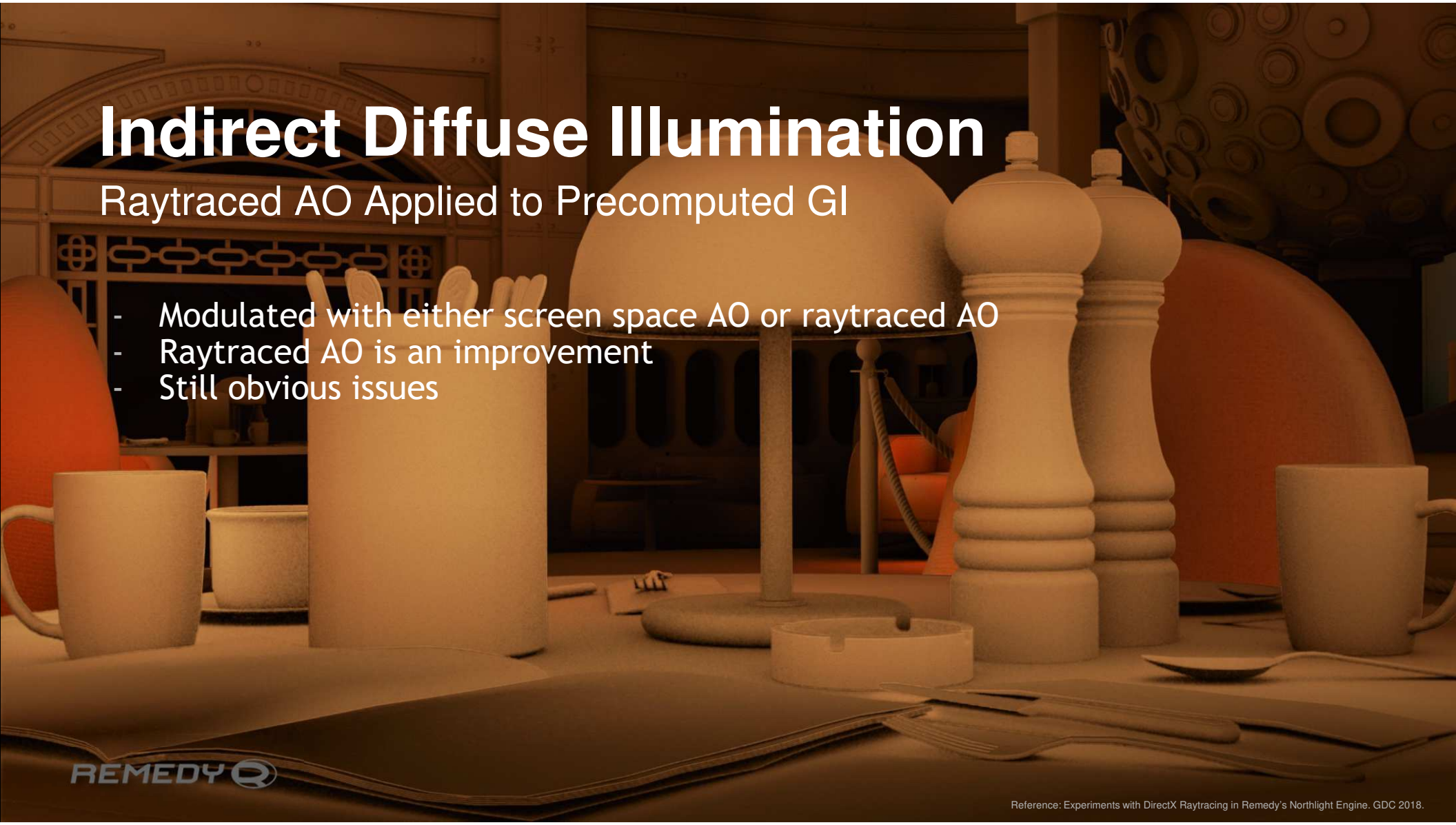
Reflections

Transparent objects

- Primary rays to G-buffer depth
  - Select transparents with cull mask
- For N closest layers
  - Reflection ray from closest-hit
  - Continue the primary ray
- After N layers
  - Process layers in any-hit
  - No more reflections

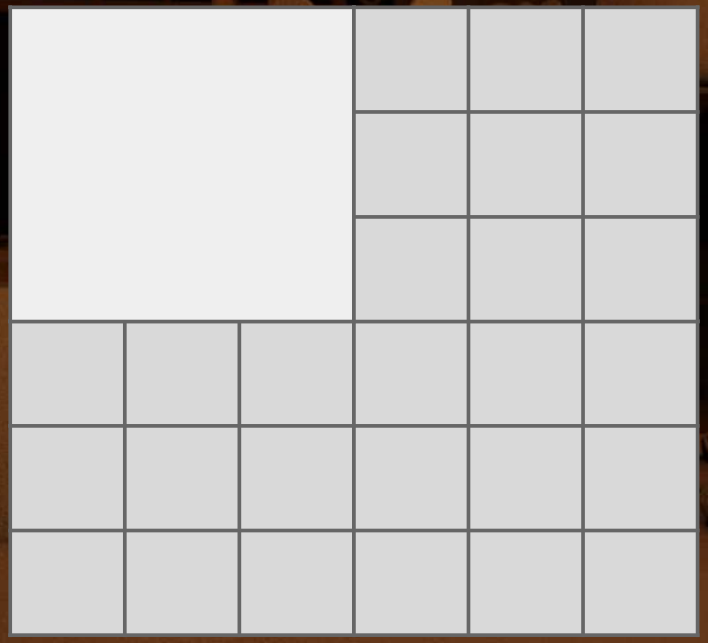
- “Weighted, Blended Order-Independent Transparency”
  - https://developer.nvidia.com/content/transparency-or-translucency-rendering


Indirect Diffuse Illumination
Starting Point - Global Illumination as in Quantum Break
- Pre-computed with path tracer
- Voxel based
- Resolution 25 cm / 10 inch
REMEDY
Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.


Indirect Diffuse Illumination
Raytraced AO Applied to Precomputed GI
- Modulated with either screen space AO or raytraced AO
- Raytraced AO is an improvement
- Still obvious issues
REMEDY
Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Indirect Diffuse Illumination

GI Data in Sparse Volume Texture



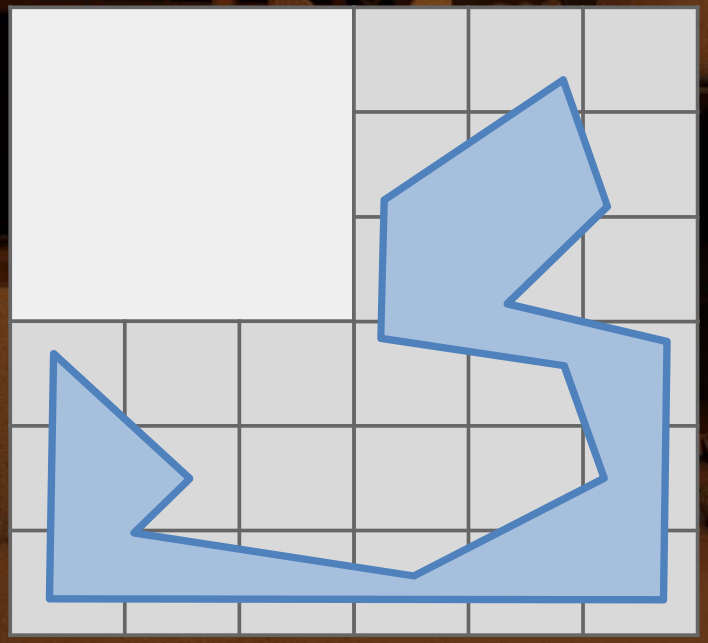
- Each cell contains lighting data that has been pre-computed with a path tracer.

REMEDY

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Indirect Diffuse Illumination

Higher Resolution Near Static Geometry



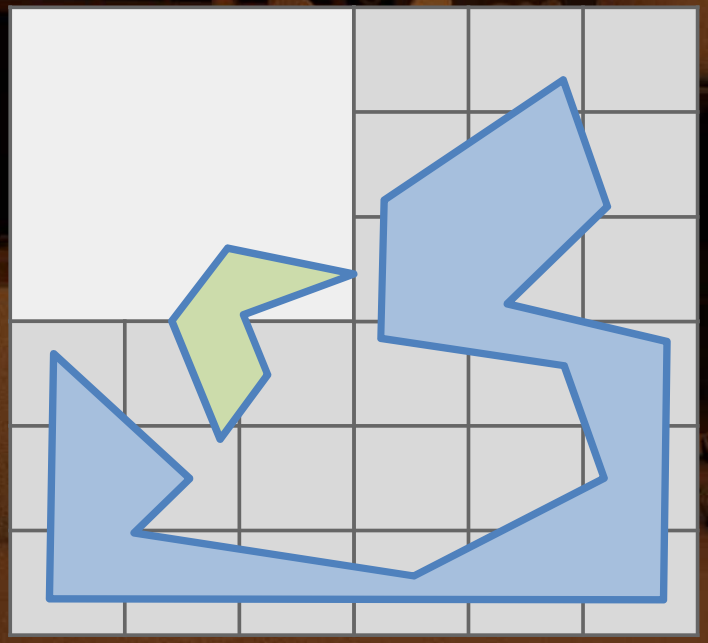
- Each cell contains lighting data that has been pre-computed with a path tracer.
- Static objects and selected lights are included in the pre-computation.

REMEDY

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Indirect Diffuse Illumination

Dynamic Objects Excluded from Pre-computing



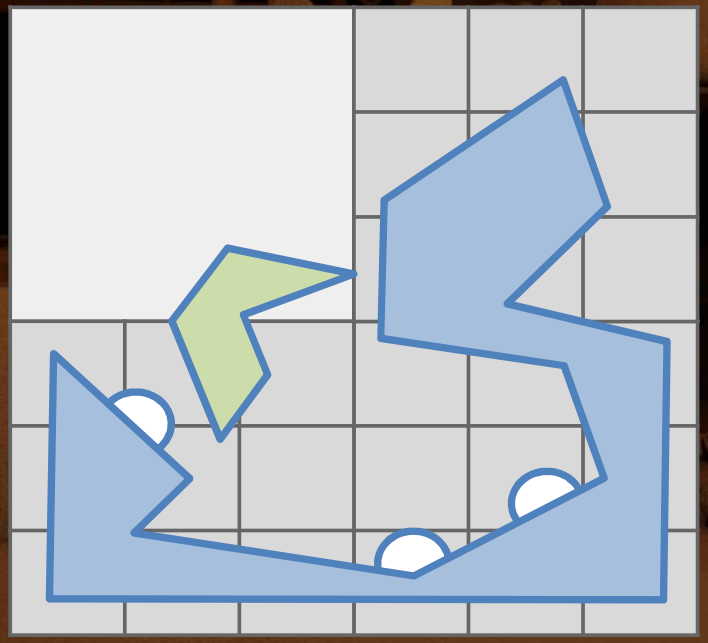
- Each cell contains lighting data that has been pre-computed with a path tracer
- Static objects and selected lights are included in the pre-computation
- Dynamic object can be in low resolution areas.

REMEDY

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Indirect Diffuse Illumination

Issues with Direct Sampling of GI Data



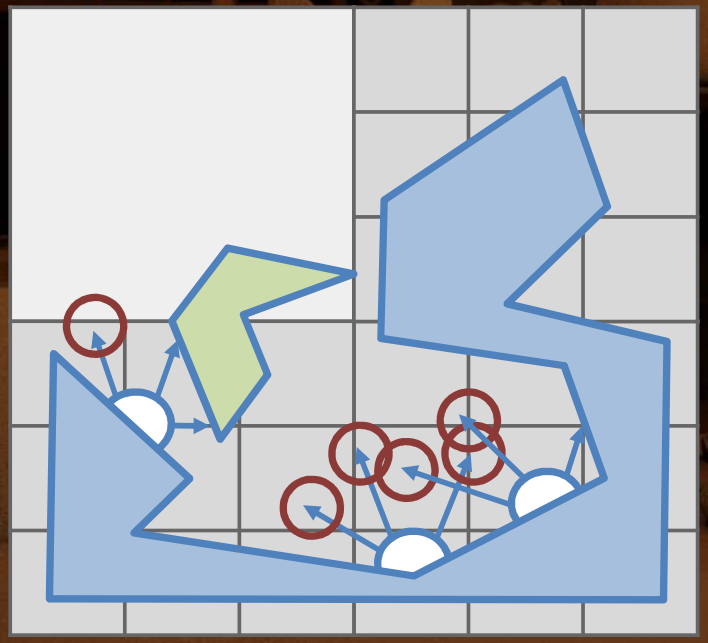
- The dynamic objects are missed.
  - AO has been the method to tie the dynamic geometry to the rest of the scene.
- Filtering of low resolution data causes banding.
- Light leaking through thin geometry.

REMEDY

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Indirect Diffuse Illumination

Sample GI in Miss Shader



- Run a raytracing pass to sample the global illumination.
- Short rays from GBuffer surface.
  - If miss, sample GI.
- Miss locations are less likely to contain leaked light.
- Cheapest option is to treat hits as black.
  - Works like ambient occlusion.

REMEDY

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

GI modulated with raytrace AO

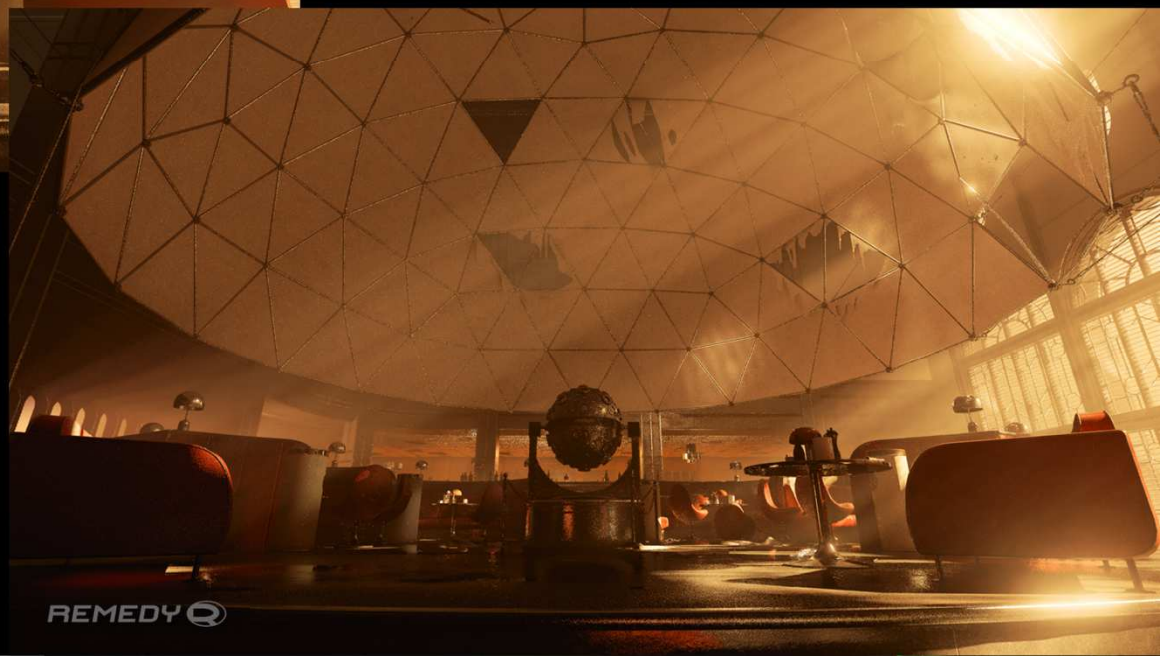
GI sampling in Miss Shader

Hit = Black

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Banding on the dome.

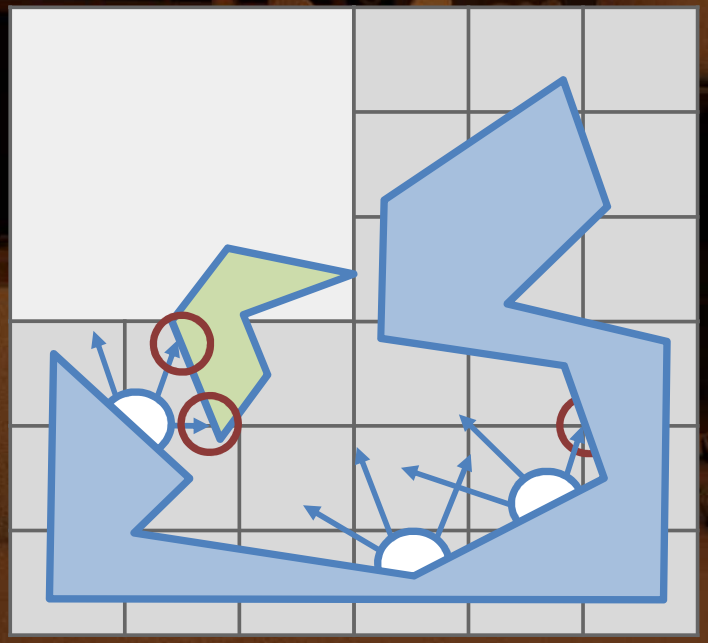
Sampling GI in Miss Shader removes the banding.



Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Indirect Diffuse Illumination

Diffuse Lighting on Hit Shader



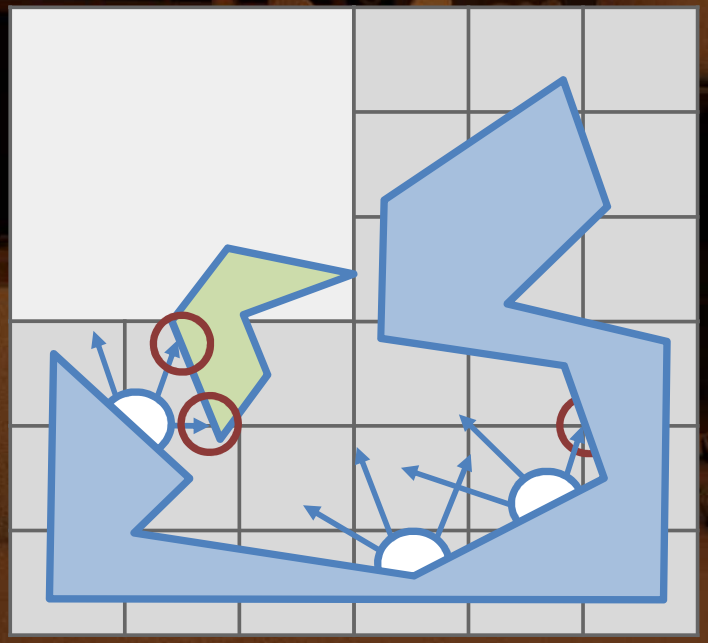
- Evaluates single bounce near field dynamic GI.
- Blend with the pre-computed GI result sampled in Miss Shader.

REMEDY

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Indirect Diffuse Illumination

Diffuse Lighting on Hit Shader



- Evaluates single bounce near field dynamic GI.
- Blend with the pre-computed GI result sampled in Miss Shader.

- Lighting and material data to hit shader as in reflections.
- View space light clustering.

GeForce RTX 2080 TI, 1920x1080

2.5 ms (1 rpp, denoising)

REMEDY

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Do nothing on hits.

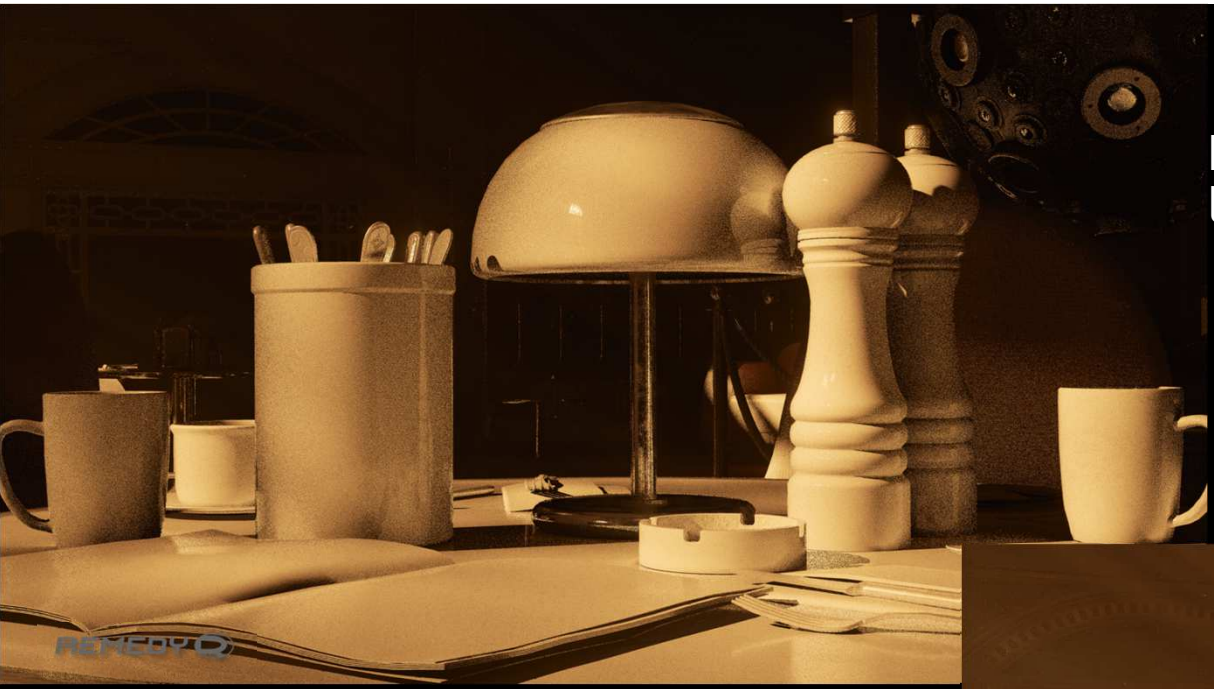
Evaluate diffuse illumination on hits.

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Direct illumination only.

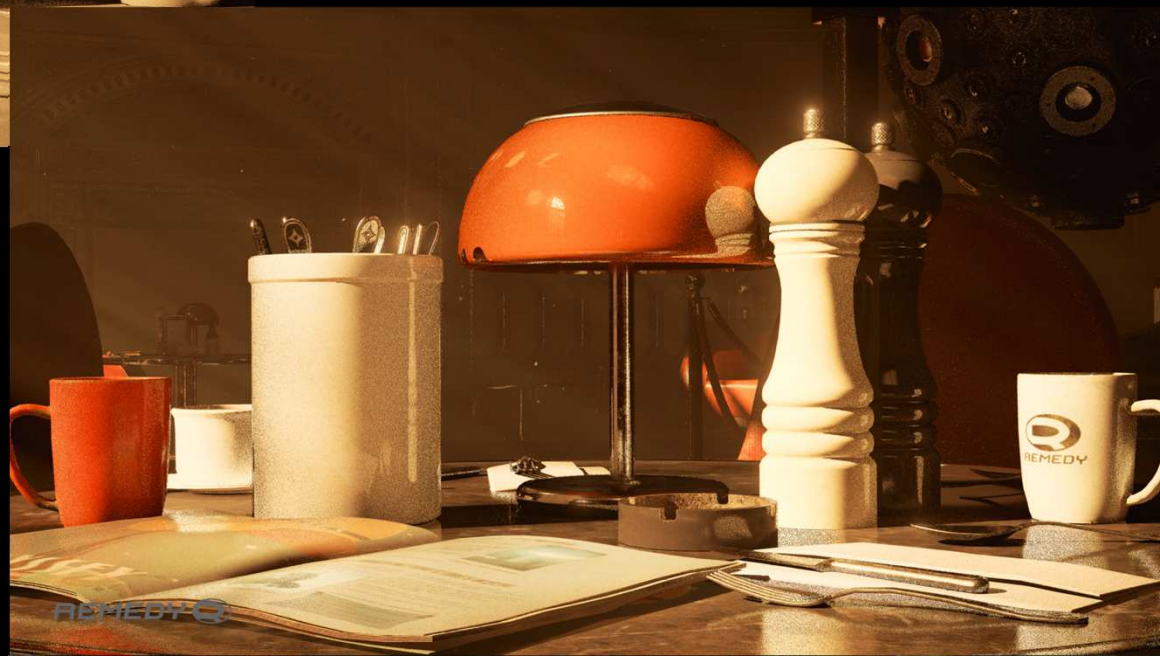
Direct and indirect diffuse illumination.

Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.



Direct lighting, indirect diffuse lighting and specular reflection.

All lighting applied to materials.



Reference: Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.

Summary

Northlight RTX Is Work in Progress

- Some integration overhead
  - It gets more productive after the initial work is done
- Some obvious straightforward effects
  - Shadows
  - Ambient occlusion
  - Reflections
- Some more creative effects
  - Contact shadows
  - Indirect Diffuse
- Useful as reference too

References

[1] Tatu Aalto. Experiments with DirectX Raytracing in Remedy's Northlight Engine. GDC 2018.
https://www.remedygames.com/experiments-with-directx-raytracing-in-remedys-northlight-engine/

The End

Questions?

Contact: jsjoholm@nvidia.com