GPU TECHNOLOGY CONFERENCE

NVIDIA VXGI

DYNAMIC GLOBAL ILLUMINATION FOR GAMES

ALEXEY PANTELEEV

DEVELOPER TECHNOLOGY ENGINEER, NVIDIA

GPU TECHNOLOGY CONFERENCE

OUTLINE

- What is VXGI
- Algorithm Overview
- Engine Integration
- VXGI in UE4
- Quality and Performance
- Ambient Occlusion Mode
- Q&A

"Voxel Global Illumination (VXGI) is a ***stunning advancement****, delivering* ***incredibly realistic lighting****, shading and reflections to next-generation games and game engines."*

Geforce.com

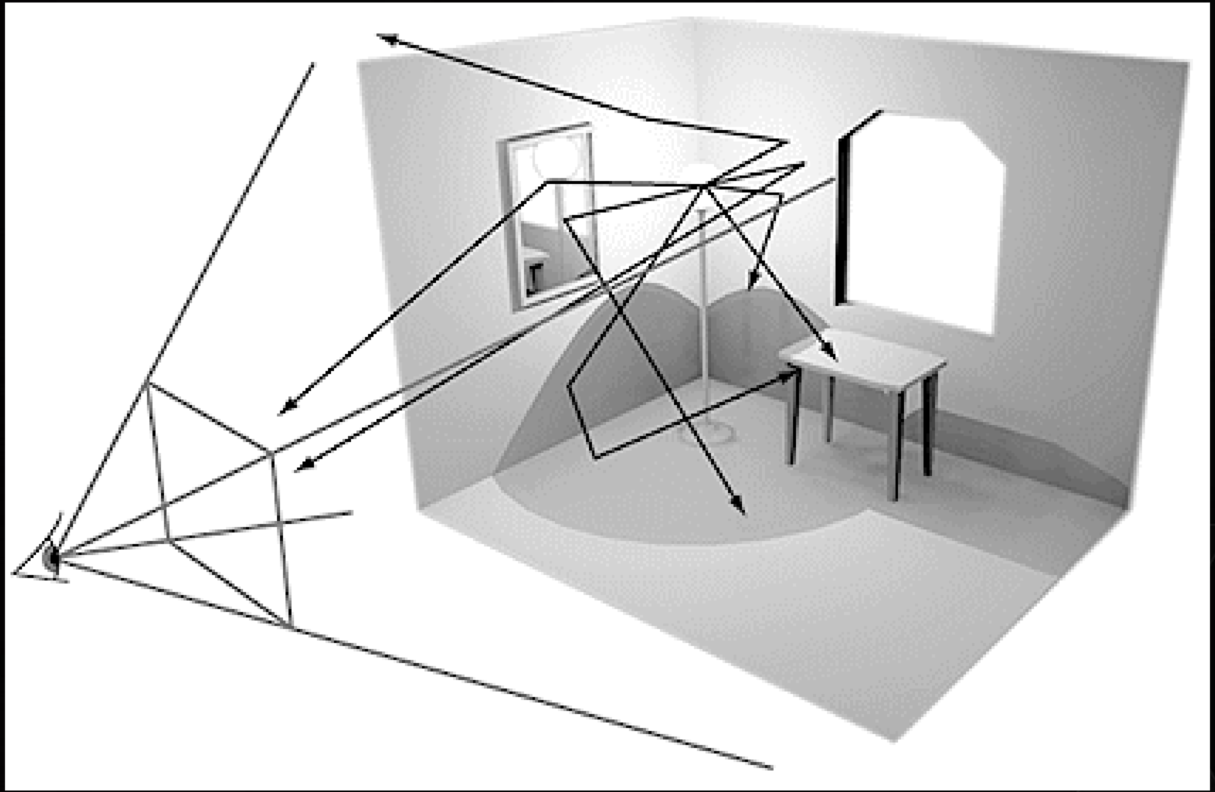
WHAT VXGI REALLY IS

- A software library that computes approximate indirect illumination
  - Works on any DX11 GPU, faster on Maxwell
  - Has to be integrated into rendering engines
  - UE4 integration available
  - One bounce of indirect illumination
- An algorithm inspired by SVOGI
  - Voxel cone tracing
  - Clip-map instead of an octree
  - Handles large and dynamic scenes well, no preprocessing

GPU TECHNOLOGY CONFERENCE

WHY COMPUTING GI IS HARD

1 of 5

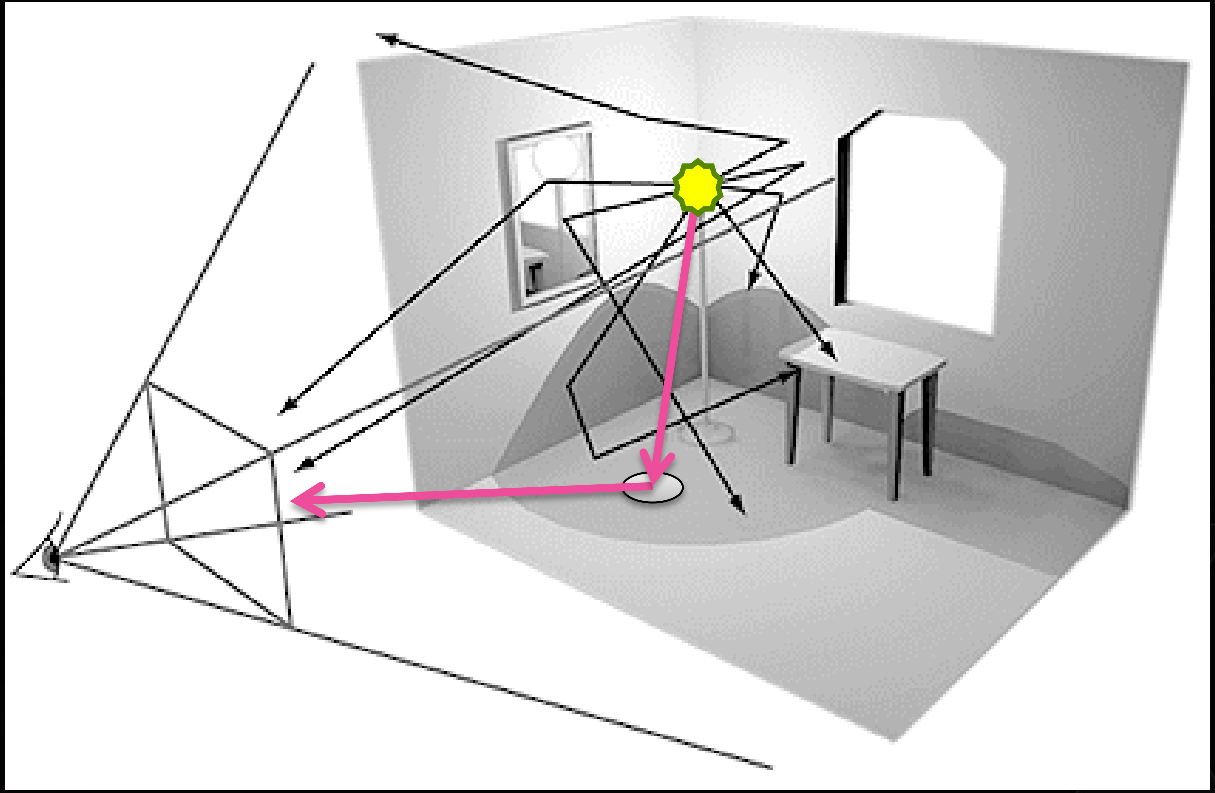


A photon can take one of many paths between the light and the observer.

GPU TECHNOLOGY CONFERENCE

WHY COMPUTING GI IS HARD

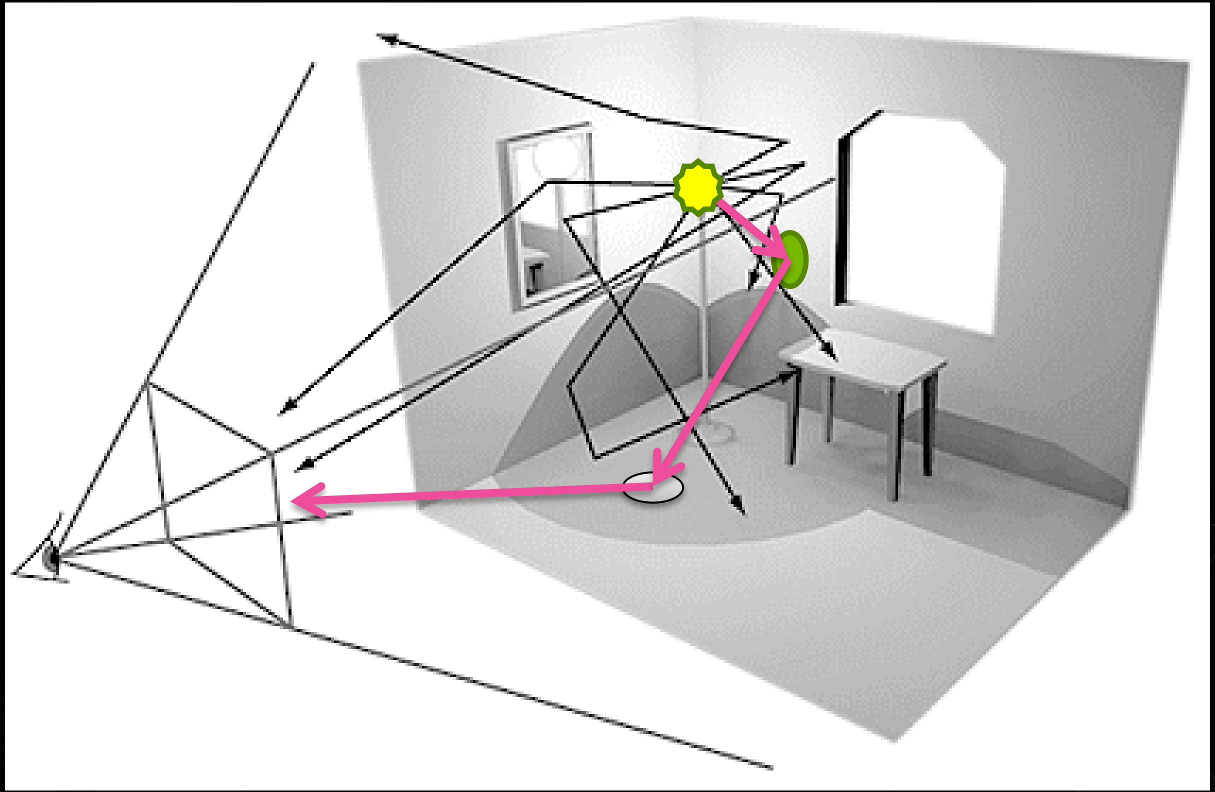
2 of 5



Direct illumination – a single possible path for every visible point on a surface.

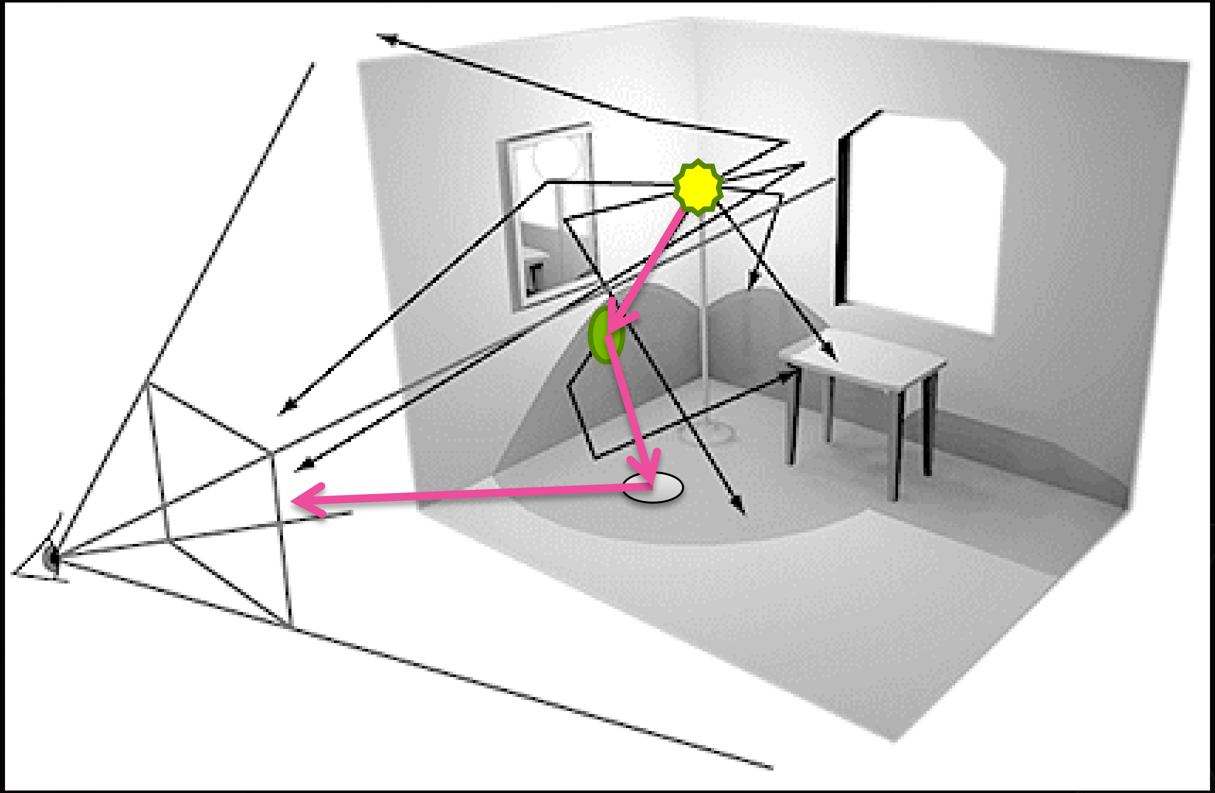
WHY COMPUTING GI IS HARD

3 of 5



One bounce indirect illumination for the same point – one of the many paths…

WHY COMPUTING GI IS HARD

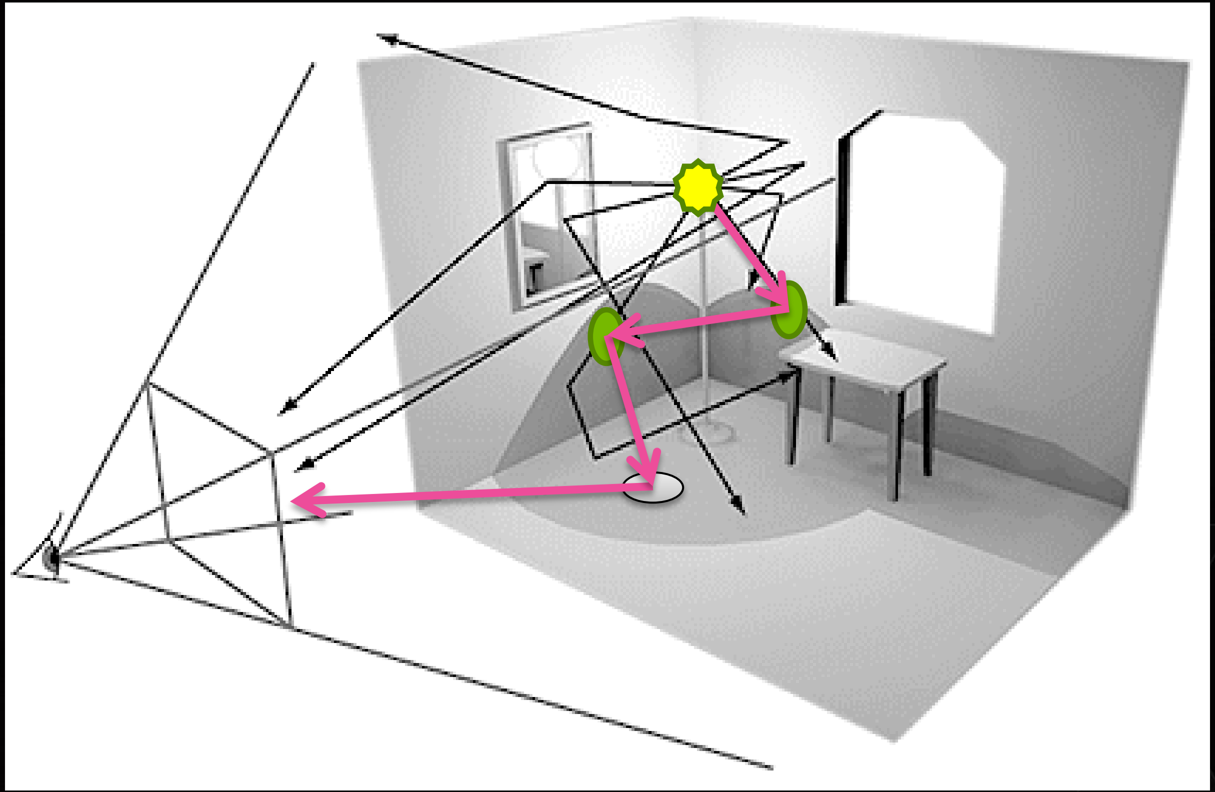


Another path for the same visible point.

GPU TECHNOLOGY CONFERENCE

WHY COMPUTING GI IS HARD

5 of 5



This path is also possible – it’s two bounce indirect illumination.

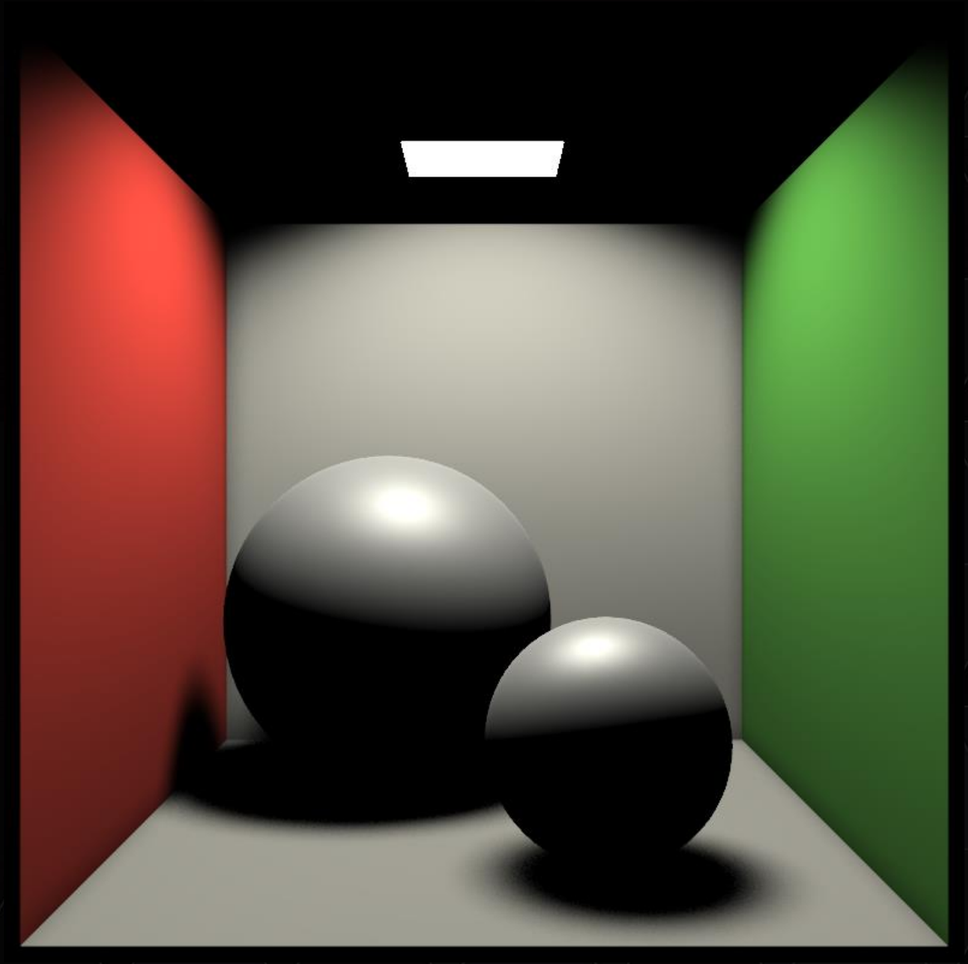
OUTLINE

- What is VXGI
- Algorithm Overview
- Engine Integration
- VXGI in UE4
- Quality and Performance
- Ambient Occlusion Mode
- Q&A

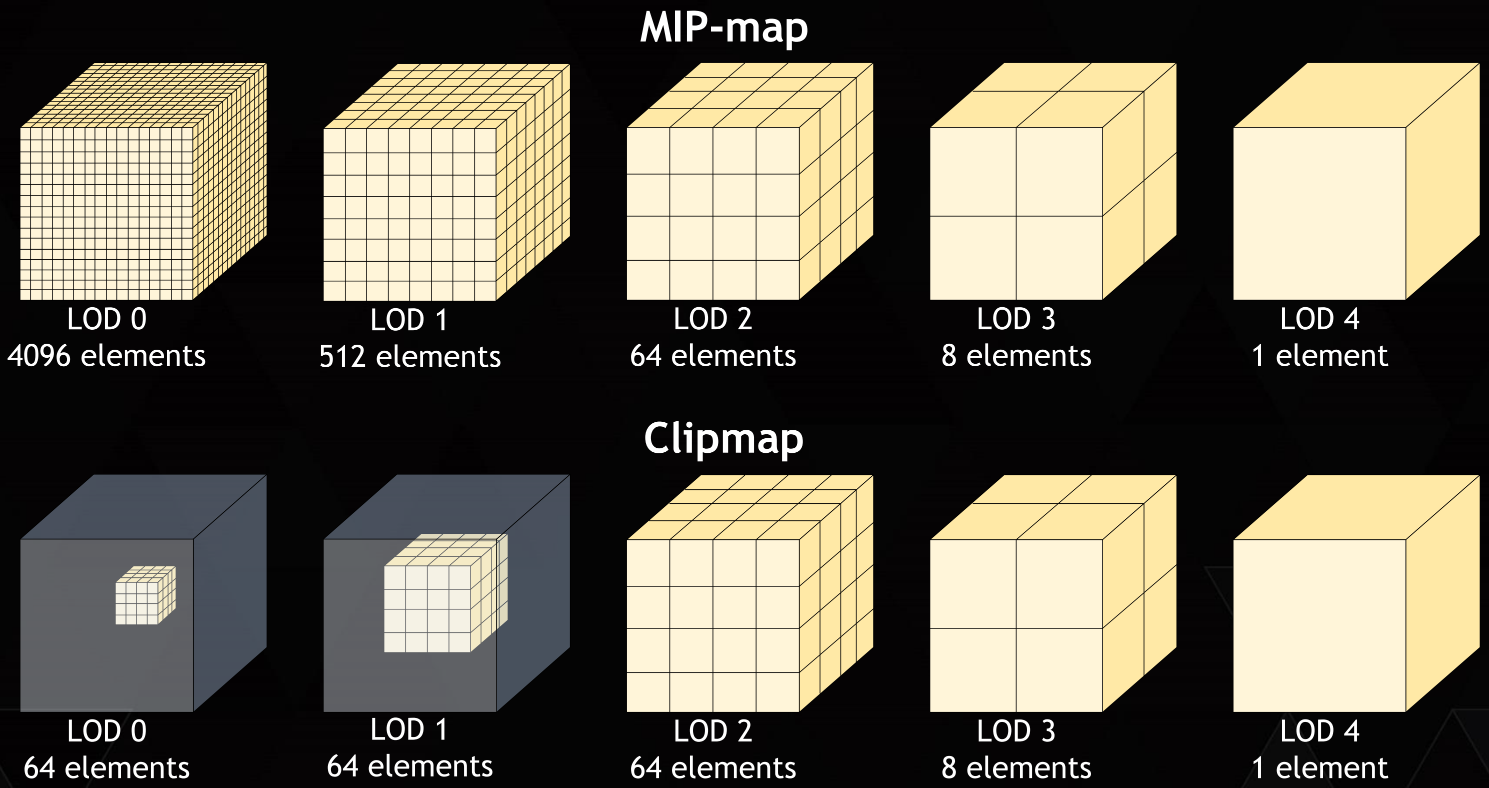
VXGI ALGORITHM OVERVIEW

- Step 1: Opacity Voxelization
- Step 2: Emittance Voxelization
- Step 3: Cone Tracing

- Use Cornell Box as an example



VOXEL STORAGE: 3D CLIP-MAP



MIP-map

LOD 0
4096 elements

LOD 1
512 elements

LOD 2
64 elements

LOD 3
8 elements

LOD 4
1 element

Clipmap

LOD 0
64 elements

LOD 1
64 elements

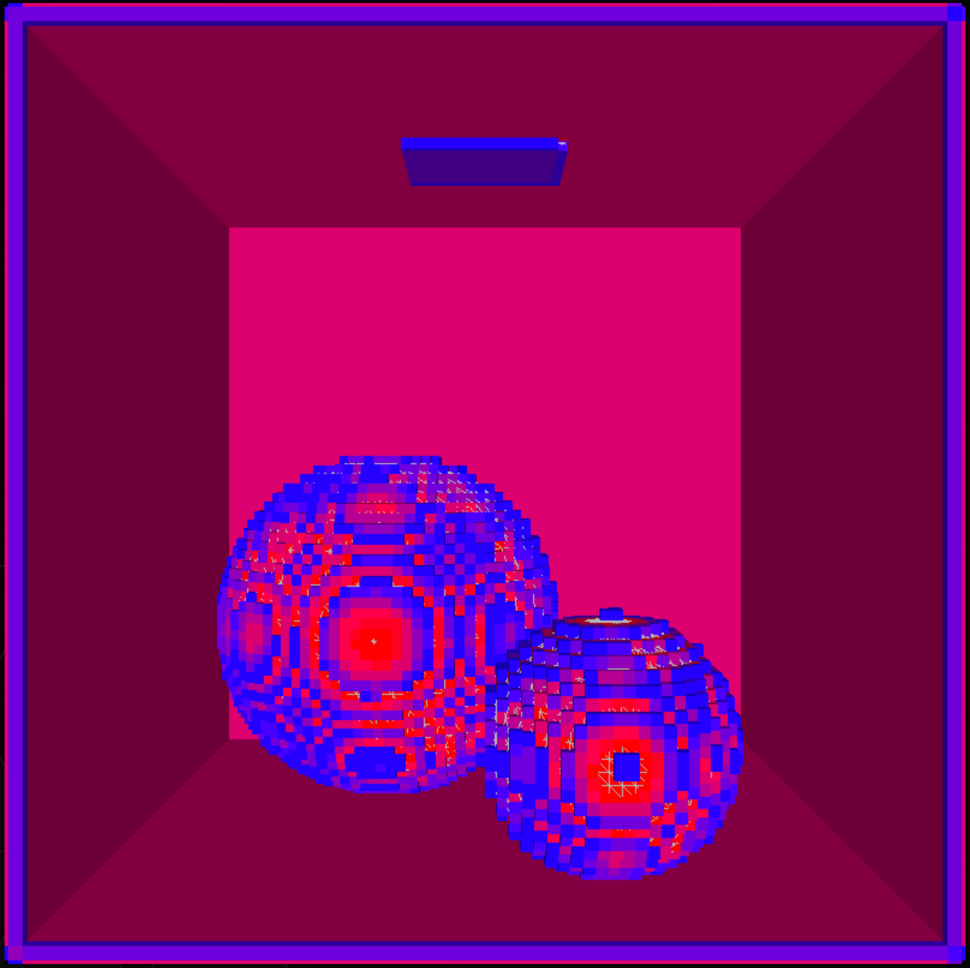
LOD 2
64 elements

LOD 3
8 elements

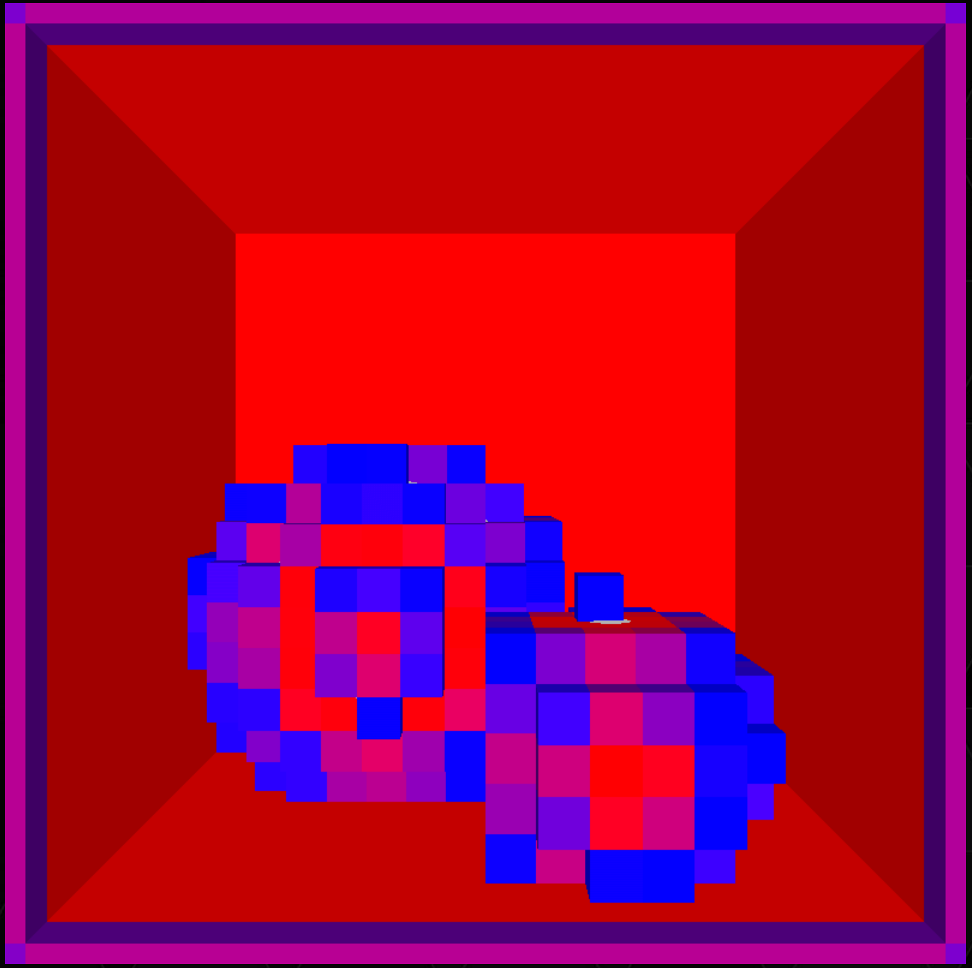
LOD 4
1 element

- Hardware addressing => much faster than SVO
- Scalable: $(32\ldots256)^3$ with 3…5 LODs, 16…56 bytes per voxel
  => 1.5 MB … 4.5 GB VRAM

OPACITY VOXELIZATION



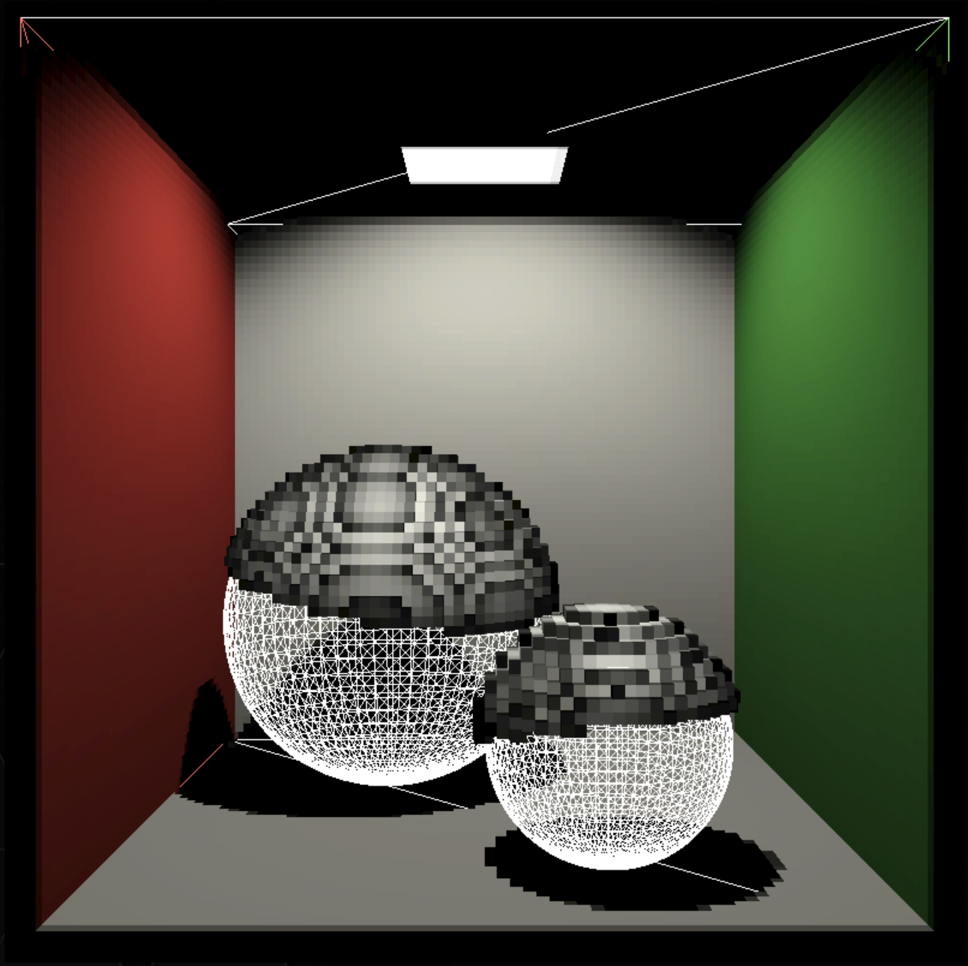
Finest level of detail (LOD 0)



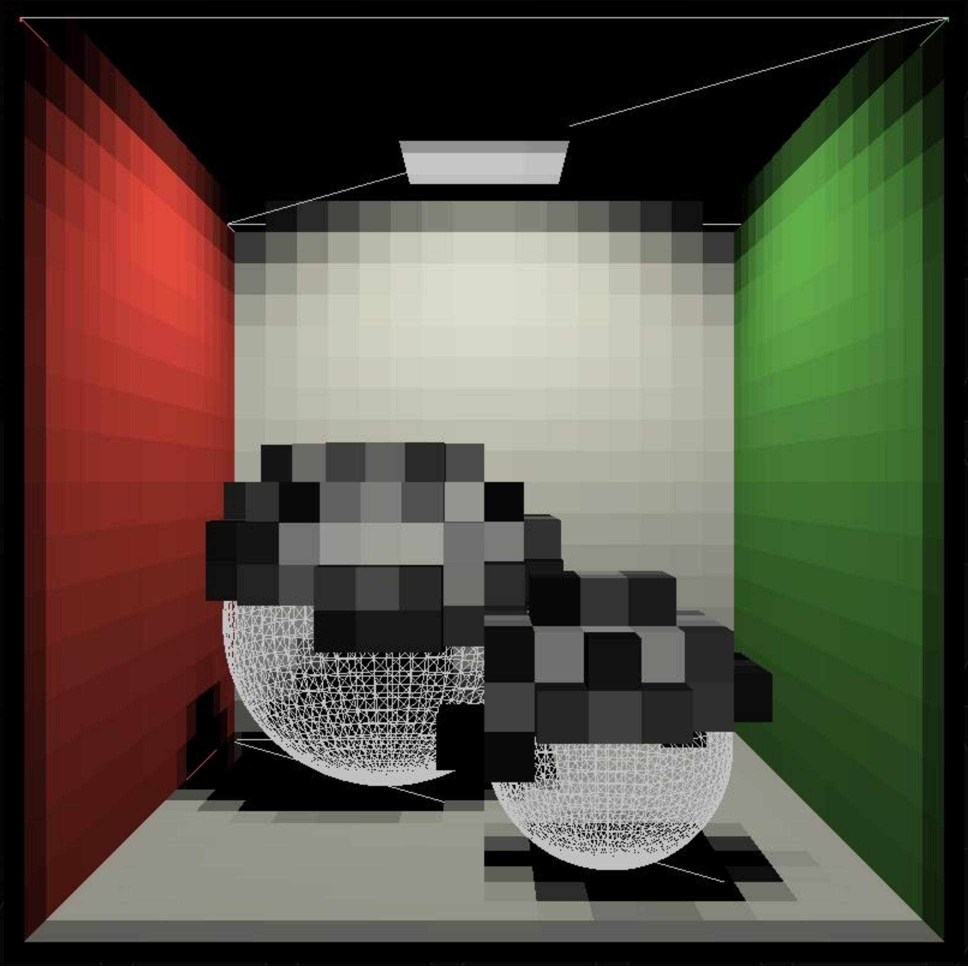
4x coarser representation (LOD 2)

GPU TECHNOLOGY CONFERENCE

EMITTANCE VOXELIZATION



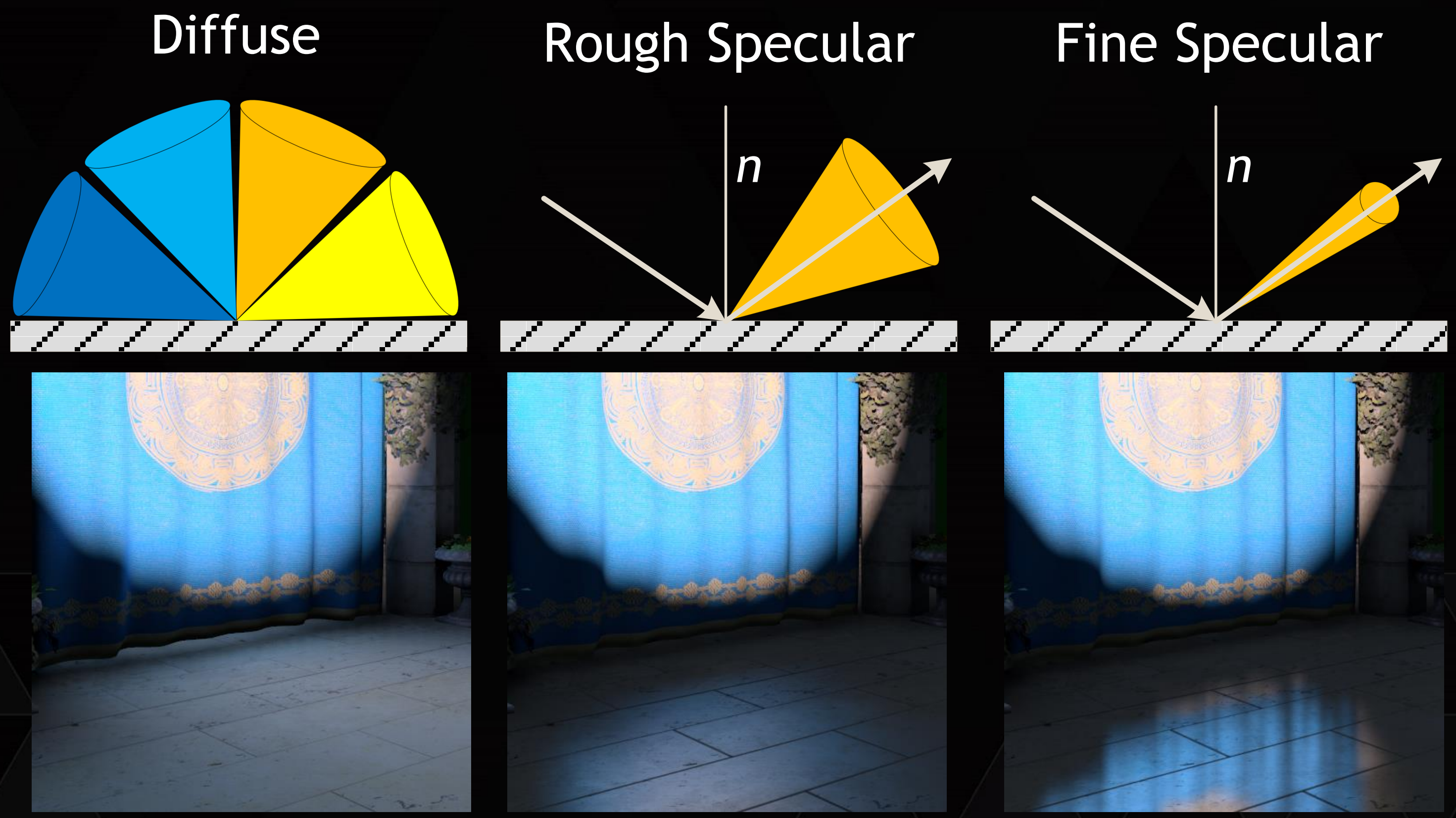
Finest level of detail (LOD 0)



4x coarser representation (LOD 2)

CONE TRACING

$$Irradiance = \sum Emittance_i \left(\frac{ConeFactor}{SampleSize}\right)^2 \prod_0^i (1 - Opacity_k)^{tStep \times OpacityCorrectionFactor}$$



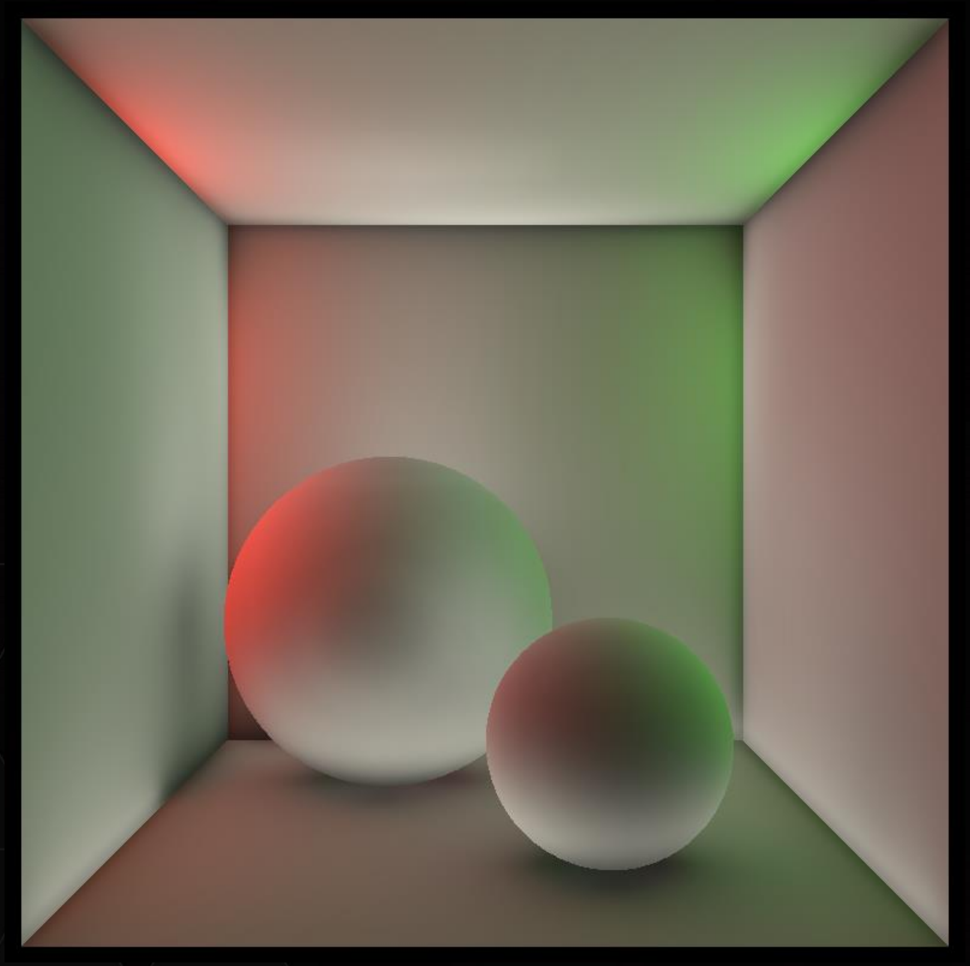
Diffuse

Rough Specular

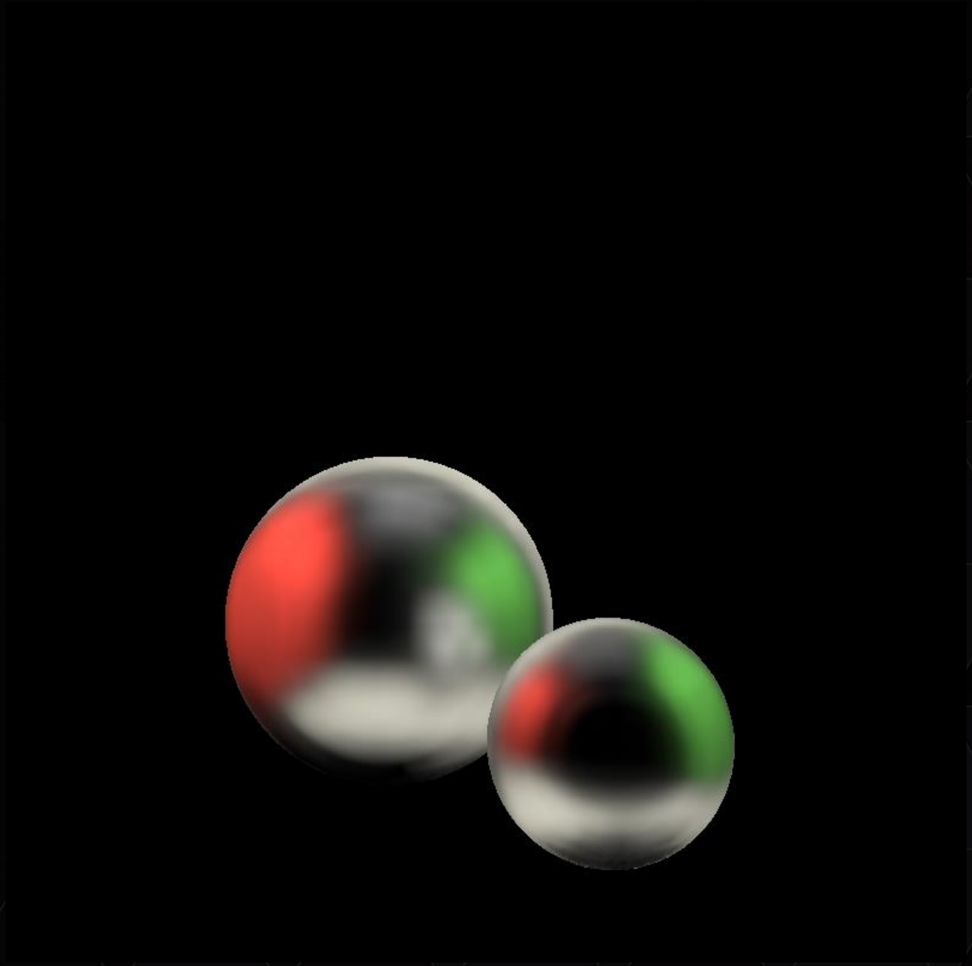
Fine Specular

GPU TECHNOLOGY CONFERENCE

RESULTS OF CONE TRACING

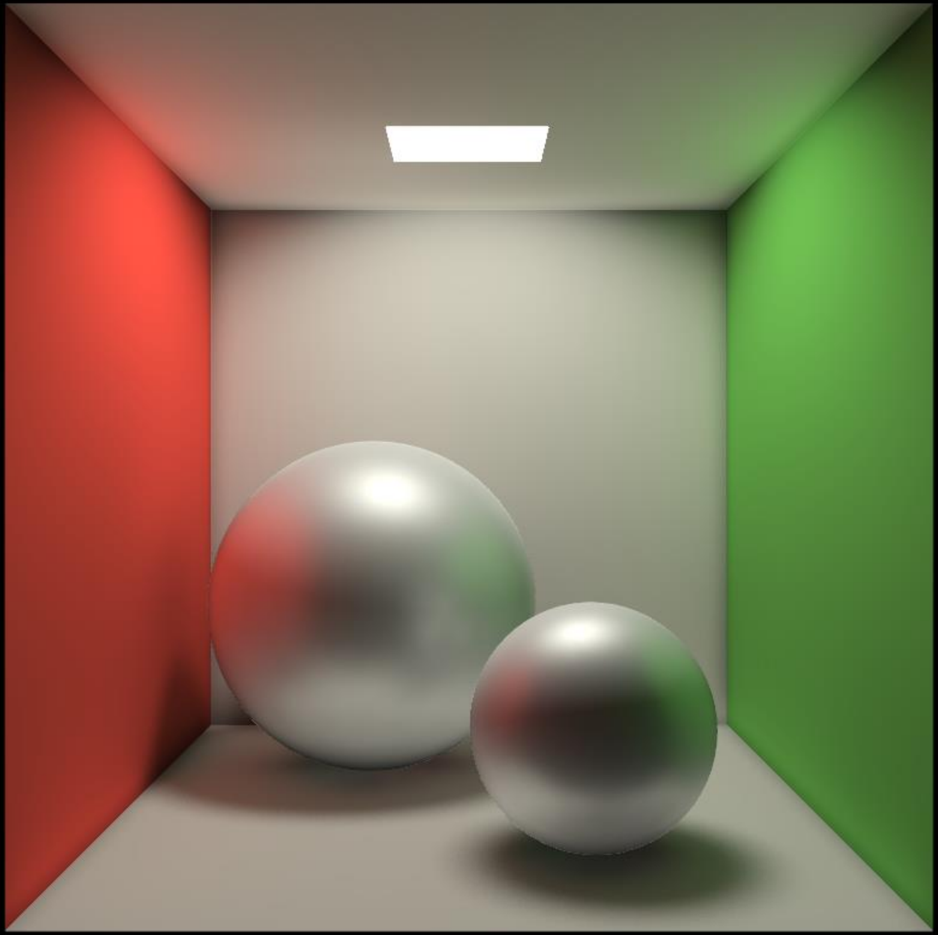


Indirect diffuse lighting

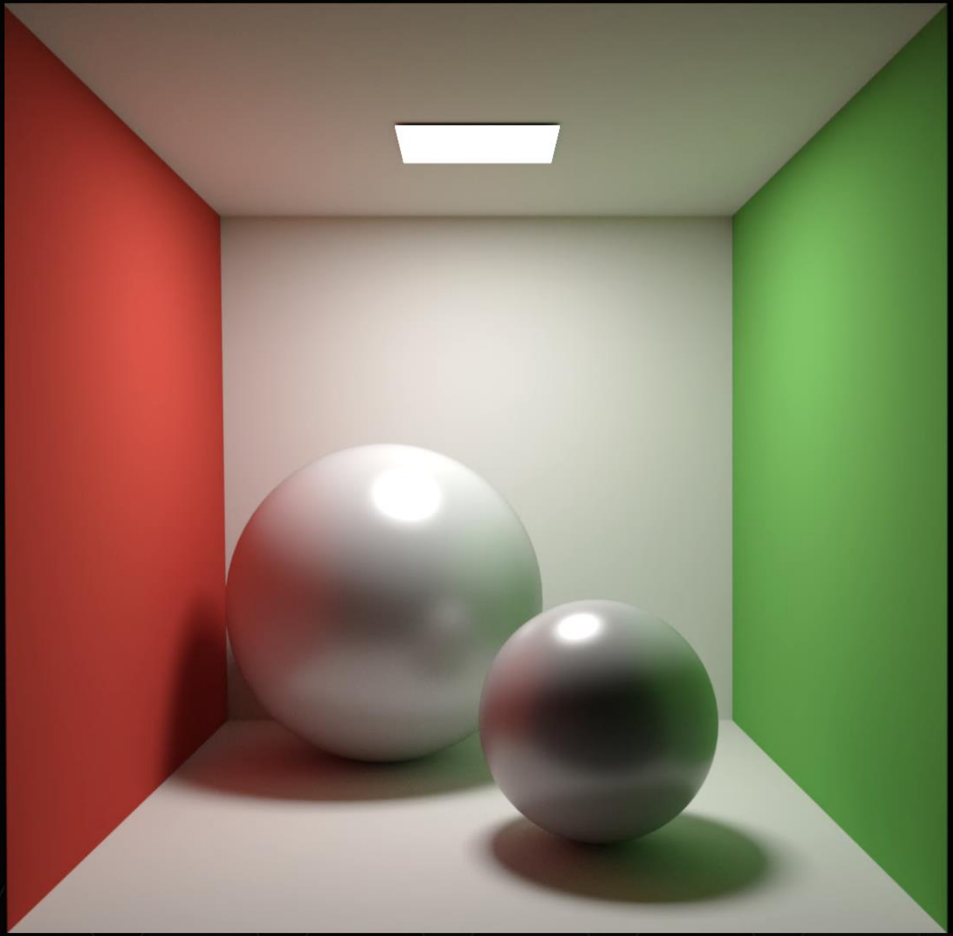


Indirect specular reflections

FINAL RESULT



Direct and VXGI Indirect combined



Reference rendering with NVIDIA Iray

DEMO: SAN MIGUEL

OUTLINE

- What is VXGI
- Algorithm Overview
- Engine Integration
- VXGI in UE4
- Quality and Performance
- Ambient Occlusion Mode
- Q&A

VXGI INTEGRATION OVERVIEW

1. Implement or copy and adapt the RHI backend
2. Initialize a GI object
3. Test voxelization and debug visualization
4. Implement app specific voxelization
5. Compute indirect illumination using a G-buffer

STEP 1. RHI BACKEND

- VXGI API is based on C++ classes
- VXGI works with the rendering APIs through RHI abstraction
  - RHI = Rendering Hardware Interface
  - Supports Direct3D 11 now
  - Will support OpenGL 4.4 soon
- The application implements the RHI backend
  - We provide a reference implementation of the DX11 backend
- The interface is stateless, consists of methods like these:
  - TextureHandle createTexture(const TextureDesc& d, const void* data);
  - void writeConstantBuffer(ConstantBufferHandle b, const void* data, size_t dataSize);
  - void dispatchCompute(const DispatchState& state, const Vector3u& groupCount);

STEP 2 & 3. INITIALIZATION

- Call VFX_VXGI_CreateGIObject(const GIParameters& params, …) supplying:
  - Voxelization parameters: clipmap geometry, quality options
  - Reference to the RHI backend
  - Custom memory allocator, error callback function, perf monitor interface

- Test voxelization using a built-in cube scene:

```
pGI->prepareForOpacityVoxelization(…)
pGI->voxelizeTestScene(position, size)
pGI->prepareForEmittanceVoxelization()
pGI->voxelizeTestScene(position, size)
pGI->finalizeVoxelization()
pGI->renderDebug(mode, viewMatrix, …)
```

STEP 4. VOXELIZATION

- Create the voxelization shaders once:
  - pGI->createVoxelizationGeometryShaderFromVS(…const void* binary…);
  - pGI->createVoxelizationPixelShader(…const char* source…);
- Voxelize scene geometry on every frame:
  - pGI->prepareForOpacityVoxelization(const UpdateVoxelizationParameters& params, …);

    ```
    VXGI::MaterialInfo info = /* your code describing the material */;
    VXGI::DrawCallState state;
    pGI->getVoxelizationState(info, state);
    pRHIBackend->applyState(state);
    pD3DContext->DrawIndexed(…);
    pD3DContext->DrawIndexed(…);
    ```
  - pGI->prepareForEmittanceVoxelization(…);

    Repeat the same sequence…
  - pGI->finalizeVoxelization();

STEP 5. TRACING

- Create a tracer once:
  - pGI->createNewTracer(&pTracer);

- Call pTracer->setInputBuffers(…) on every frame
  - gbufferDepth
  - gbufferNormal with roughness in .a
  - gbufferGeoNormal – a smoother normal channel, optional
  - environmentMap – a far-away environment map, optional

- Compute indirect illumination channels:
  - computeDiffuseChannel(const DiffuseTracingParameters& params…)
  - computeSpecularChannel(const SpecularTracingParameters& params…)

- Composite indirect lighting with your direct lighting

VOXELIZATION SHADERS

- Voxelization PS is combined from your code and our code (in HLSL)

- Your part of the shader evaluates material parameters
  - Generate any attributes in the VS and we'll get them through the GS
  - Bind any textures or other resources in the PS, just let us know where
- Your part of the shader computes emitted and reflected radiance
  - Use any lighting models, sample shadow maps, whatever
  - You can trace opacity cones when voxelizing for emittance
- Our part of the shader takes care of updating the voxel data

```
void main(MyPSInput IN)
{
    float3 color = ComputeReflectedColor(IN);
    VXGI::StoreVoxelizationData(IN.vxgiData, color);
};
```

CONE TRACING SHADERS

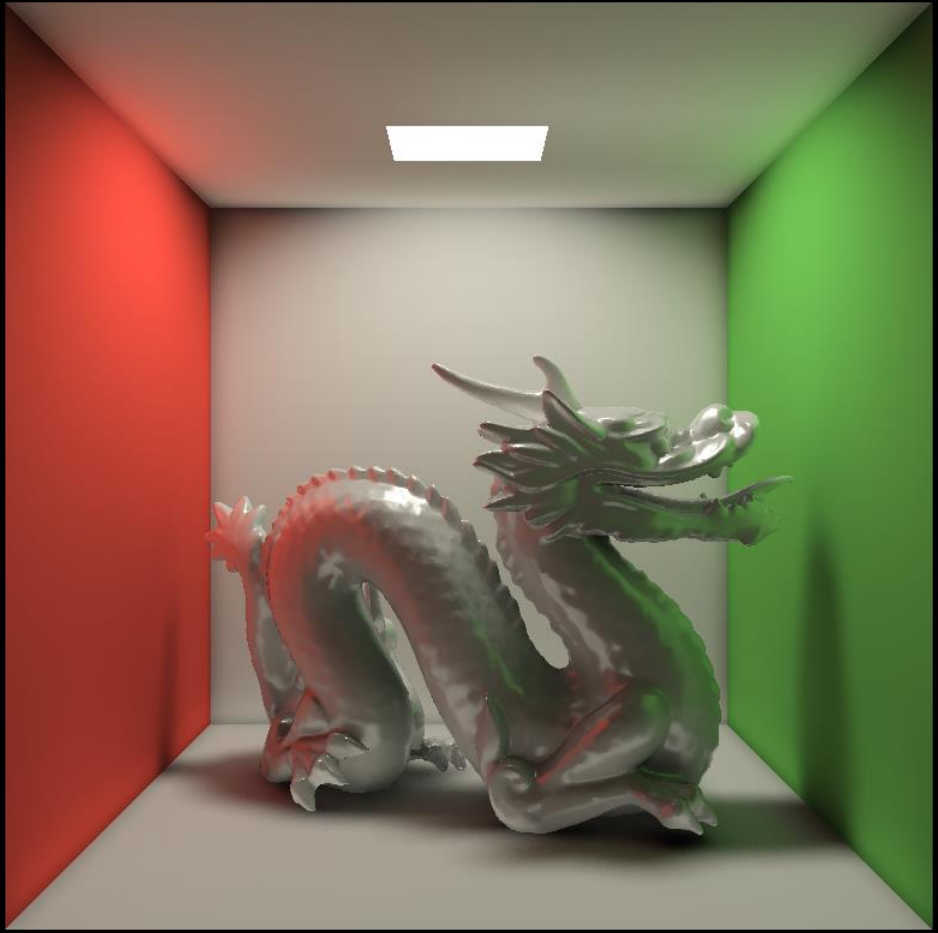
- You can create arbitrary shaders that call our cone tracing function

```
VXGI::ConeTracingArguments args = VXGI::DefaultConeTracingArguments();
    args.coneFactor = …;
    args.direction = …;
    args.firstSamplePosition = …;
VXGI::ConeTracingResults cone = VXGI::TraceCone(args);
```

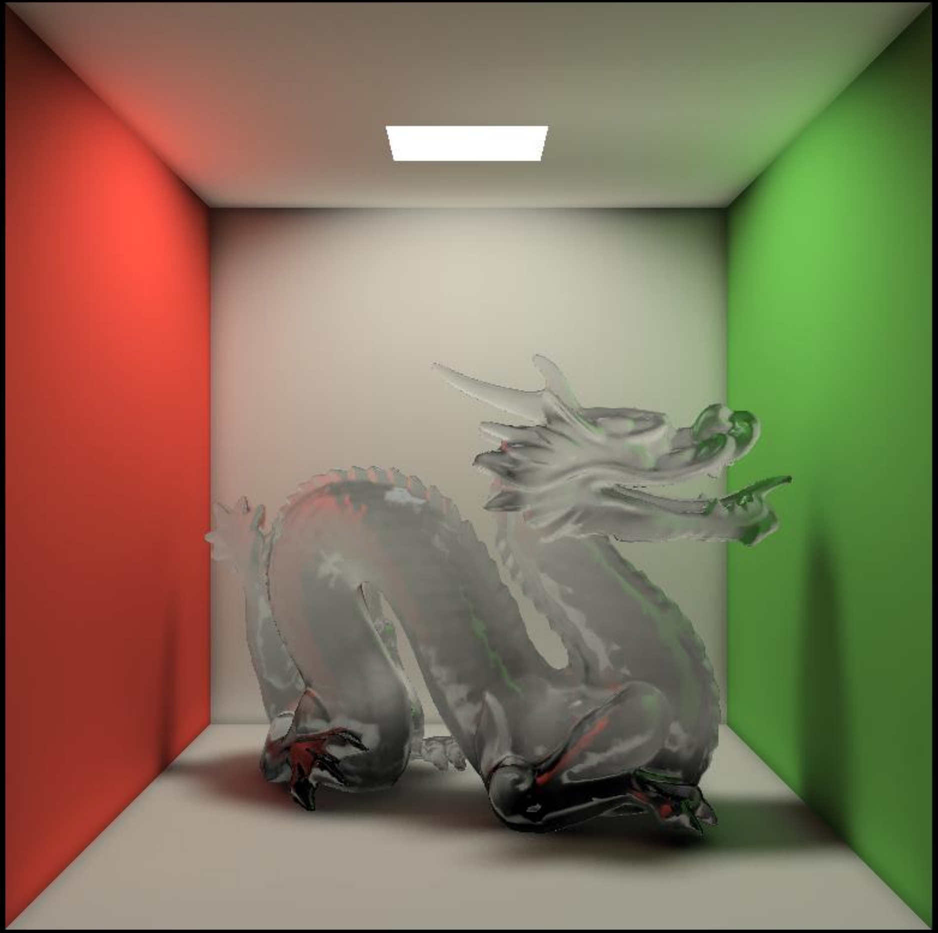
Use cone.**irradiance**, cone.ambient, cone.finalOpacity

- Use it for…
  - Advanced material effects: **refraction**, anisotropic reflection
  - Building **light maps** or reflection probes quickly
  - Implementing other diffuse illumination algorithms using our data

REFRACTIVE MATERIAL EXAMPLE



Regular VXGI diffuse + specular



Custom refraction + reflection material

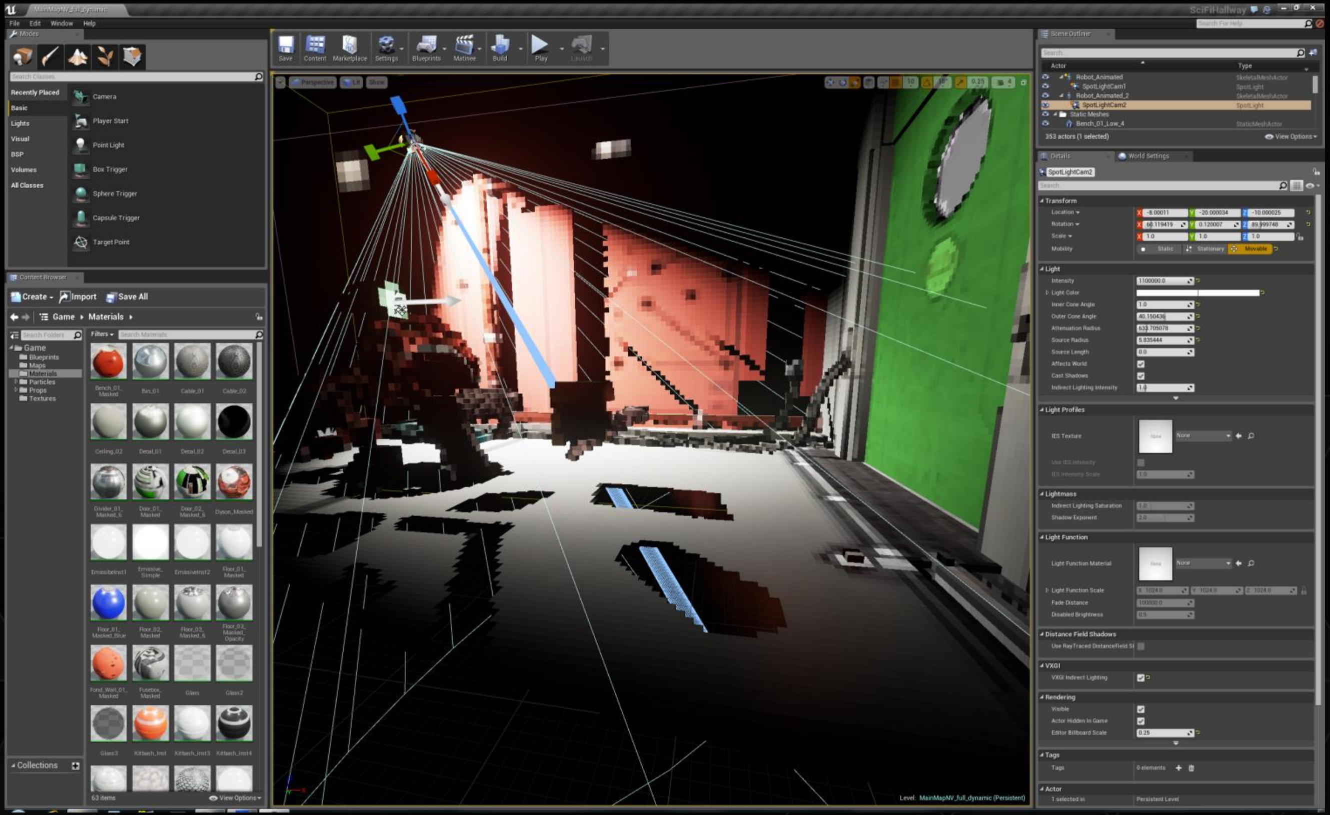
GPU TECHNOLOGY CONFERENCE

OUTLINE

- What is VXGI
- Algorithm Overview
- Engine Integration
- VXGI in UE4
- Quality and Performance
- Ambient Occlusion Mode
- Q&A

VXGI IN UE4

- https://github.com/NvPhysX/UnrealEngine
  - VXGI branch, requires a UE4 account to access



Actor	Type
Robot_Animated	SkeletalMeshActor
SpotLightCam1	SpotLight
Robot_Animated_2	SkeletalMeshActor
SpotLightCam2	SpotLight
Static Meshes	
Bench_01_Low_4	StaticMeshActor

353 actors (1 selected)

SpotLightCam2

Transform

- Location: X -8.00011, Y -20.000034, Z -10.000025
- Rotation: X 60.119419, Y 0.120007, Z 89.999748
- Scale: X 1.0, Y 1.0, Z 1.0
- Mobility: Static, Stationary, Movable

Light

- Intensity: 1100000.0
- Light Color
- Inner Cone Angle: 1.0
- Outer Cone Angle: 40.150436
- Attenuation Radius: 633.705078
- Source Radius: 5.835444
- Source Length: 0.0
- Affects World: ☑
- Cast Shadows: ☑
- Indirect Lighting Intensity: 1.0

Light Profiles

- IES Texture: None
- Use IES Intensity: □
- IES Intensity Scale: 1.0

Lightmass

- Indirect Lighting Saturation: 1.0
- Shadow Exponent: 2.0

Light Function

- Light Function Material: None
- Light Function Scale: X 1024.0, Y 1024.0, Z 1024.0
- Fade Distance: 100000.0
- Disabled Brightness: 0.5

Distance Field Shadows

- Use RayTraced DistanceField Sl: □

VXGI

- VXGI Indirect Lighting: ☑

Rendering

- Visible: ☑
- Actor Hidden In Game: ☑
- Editor Billboard Scale: 0.25

Tags

- Tags: 0 elements

Actor

- 1 selected in: Persistent Level

Level: MainMapNV_full_dynamic (Persistent)

63 items

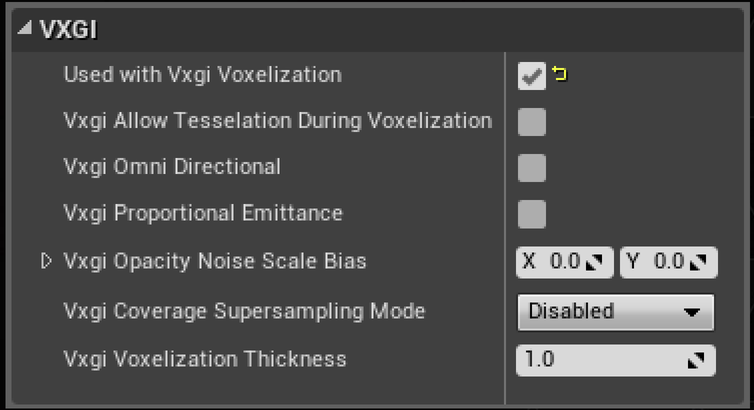
ENABLING VXGI IN A MAP

- Materials: check **Used With VXGI Voxelization**
- Lights: check **VXGI Indirect Lighting**

- **r.VXGIDebugMode 2** : opacity visualization
- **r.VXGIDebugMode 3** : emittance visualization
- **r.VXGIDebugMode 0** : regular shading

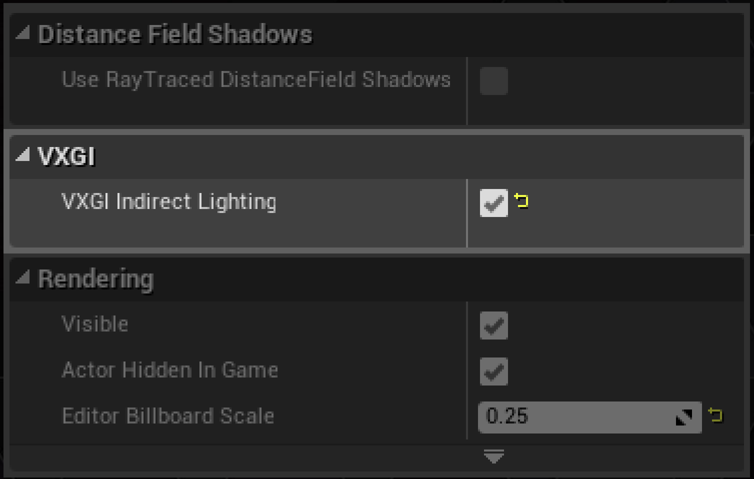
- **r.VXGI.DiffuseTracingEnable 1**
  - VXGI Diffuse is added to the UE HDR lighting

- **r.VXGI.SpecularTracingEnable 1**
  - VXGI Specular replaces UE SSR



VXGI

Used with Vxgi Voxelization	☑
Vxgi Allow Tesselation During Voxelization	□
Vxgi Omni Directional	□
Vxgi Proportional Emittance	□
Vxgi Opacity Noise Scale Bias	X 0.0 Y 0.0
Vxgi Coverage Supersampling Mode	Disabled
Vxgi Voxelization Thickness	1.0



Distance Field Shadows

Use RayTraced DistanceField Shadows	□

VXGI

VXGI Indirect Lighting	☑

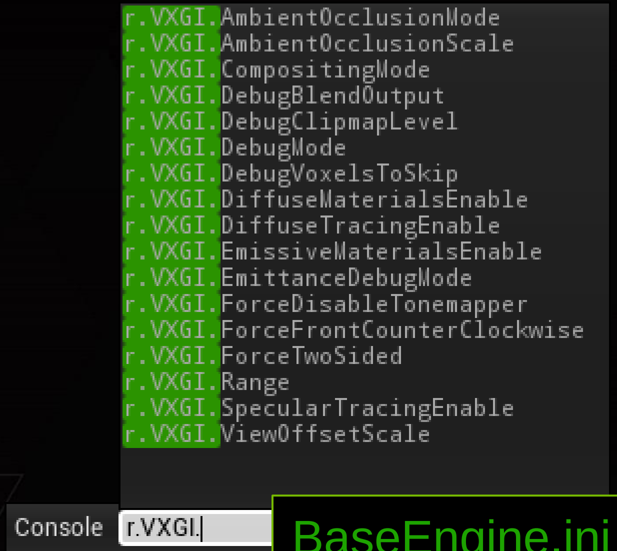
Rendering

Visible	☑
Actor Hidden In Game	☑
Editor Billboard Scale	0.25

GPU TECHNOLOGY CONFERENCE

OTHER VXGI PARAMETERS

Console Variables



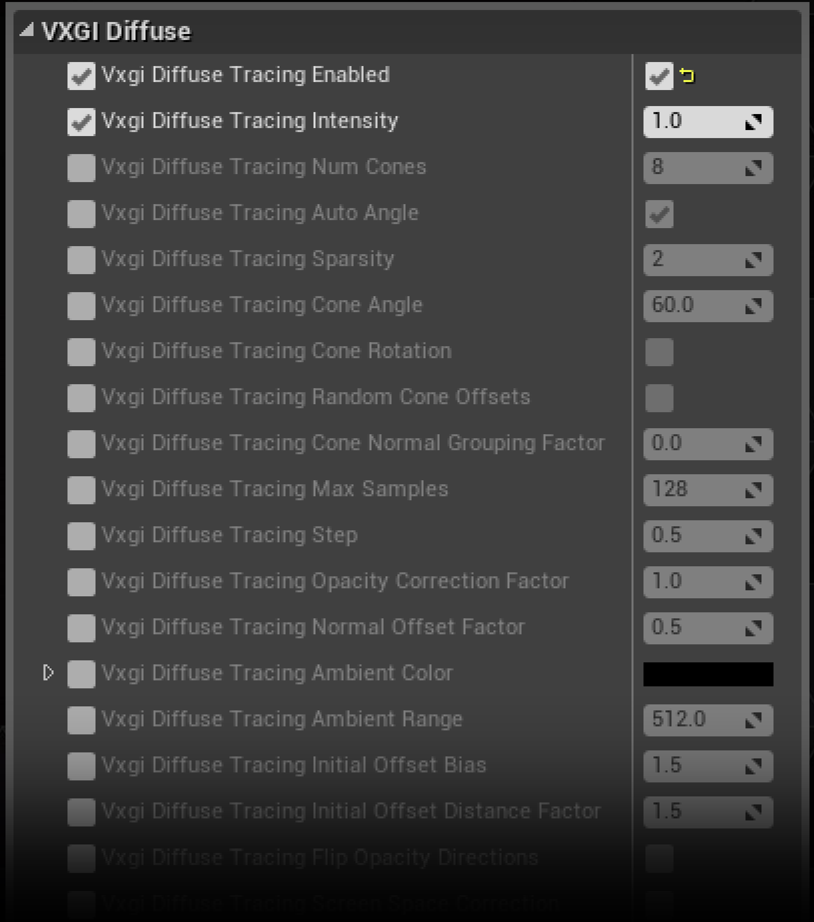
```
r.VXGI.AmbientOcclusionMode
r.VXGI.AmbientOcclusionScale
r.VXGI.CompositingMode
r.VXGI.DebugBlendOutput
r.VXGI.DebugClipmapLevel
r.VXGI.DebugMode
r.VXGI.DebugVoxelsToSkip
r.VXGI.DiffuseMaterialsEnable
r.VXGI.DiffuseTracingEnable
r.VXGI.EmissiveMaterialsEnable
r.VXGI.EmittanceDebugMode
r.VXGI.ForceDisableTonemapper
r.VXGI.ForceFrontCounterClockwise
r.VXGI.ForceTwoSided
r.VXGI.Range
r.VXGI.SpecularTracingEnable
r.VXGI.ViewOffsetScale
```

Console r.VXGI.

BaseEngine.ini

```
VxgiMapSize=128
VxgiStackLevels=5
bVxgiOpacityDirectionCount6D=true
bVxgiAmbientOcclusionMode=false
bVxgiNvidiaExtensions=true
bVxgiStoreEmittanceInFP16=false
VxgiEmittanceStorageScale=1.0
```

Cone Tracing Parameters in Post-Process Volume

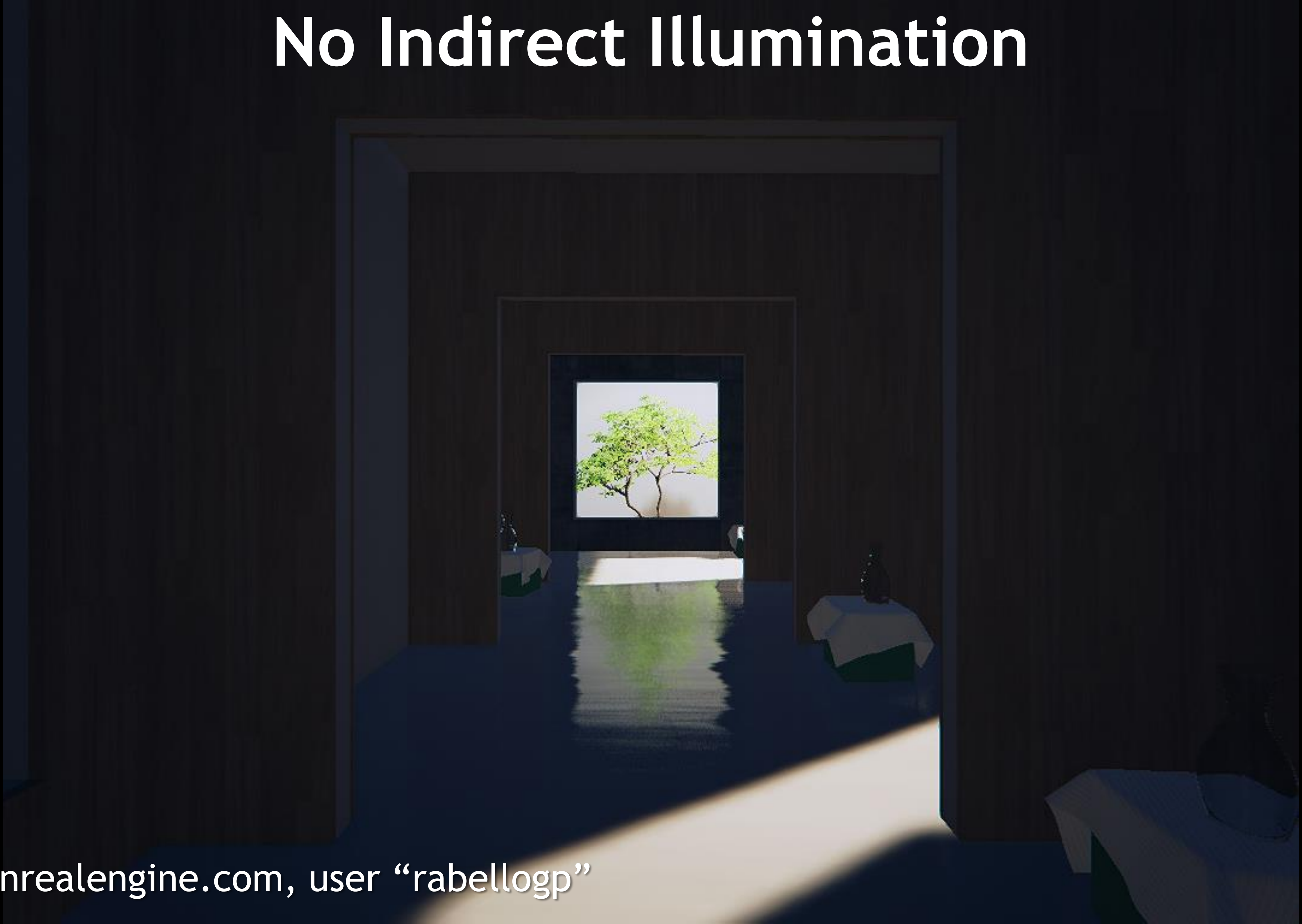


VXGI Diffuse

☑ Vxgi Diffuse Tracing Enabled	☑
☑ Vxgi Diffuse Tracing Intensity	1.0
□ Vxgi Diffuse Tracing Num Cones	8
□ Vxgi Diffuse Tracing Auto Angle	☑
□ Vxgi Diffuse Tracing Sparsity	2
□ Vxgi Diffuse Tracing Cone Angle	60.0
□ Vxgi Diffuse Tracing Cone Rotation	□
□ Vxgi Diffuse Tracing Random Cone Offsets	□
□ Vxgi Diffuse Tracing Cone Normal Grouping Factor	0.0
□ Vxgi Diffuse Tracing Max Samples	128
□ Vxgi Diffuse Tracing Step	0.5
□ Vxgi Diffuse Tracing Opacity Correction Factor	1.0
□ Vxgi Diffuse Tracing Normal Offset Factor	0.5
□ Vxgi Diffuse Tracing Ambient Color	■
□ Vxgi Diffuse Tracing Ambient Range	512.0
□ Vxgi Diffuse Tracing Initial Offset Bias	1.5
□ Vxgi Diffuse Tracing Initial Offset Distance Factor	1.5
□ Vxgi Diffuse Tracing Flip Opacity Directions	□
□ Vxgi Diffuse Tracing Screen Space Correction	□

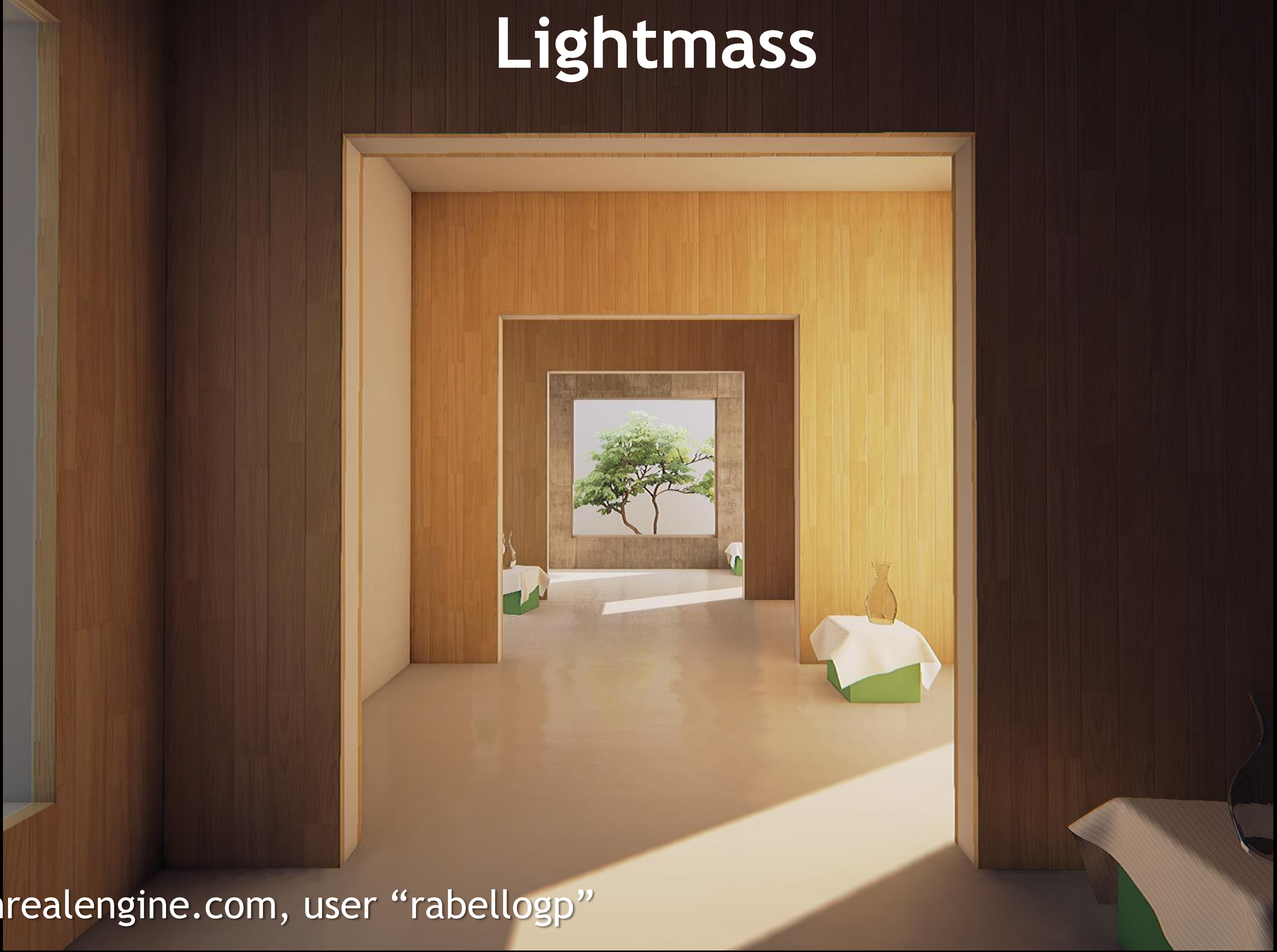
DEMO: UE4 EDITOR

No Indirect Illumination



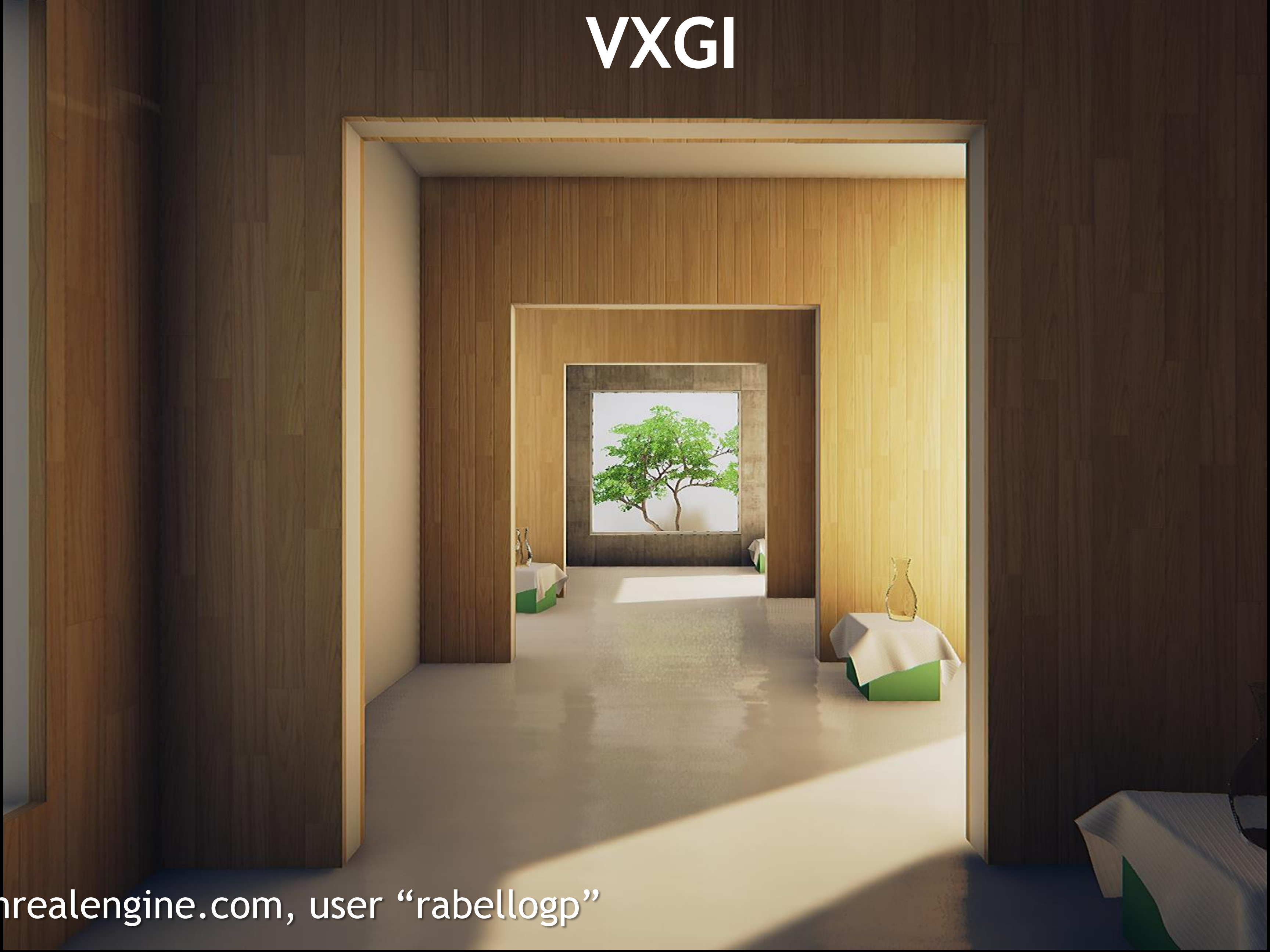
forums.unrealengine.com, user “rabellogp”

Lightmass



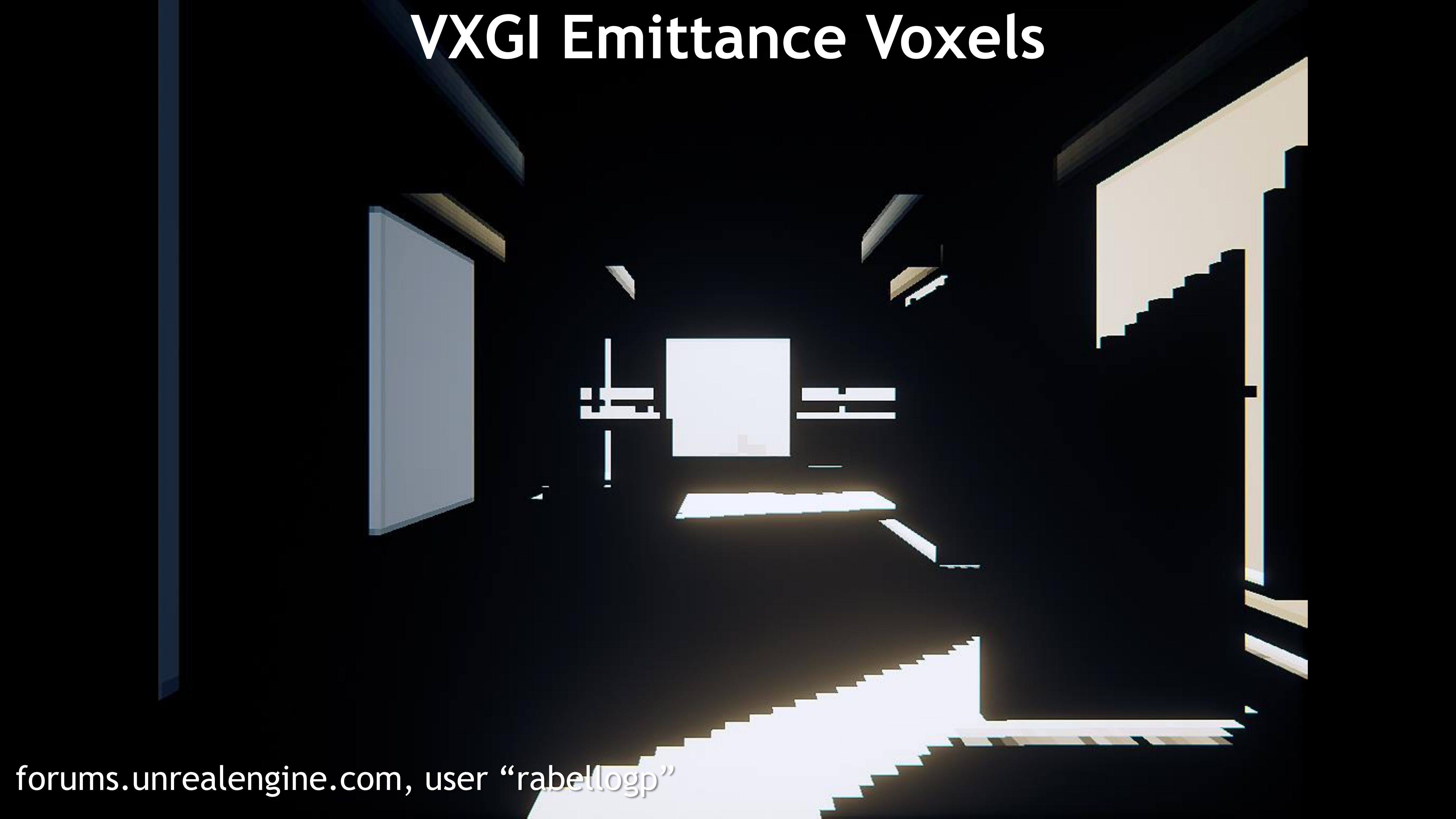
forums.unrealengine.com, user “rabellogp”

VXGI



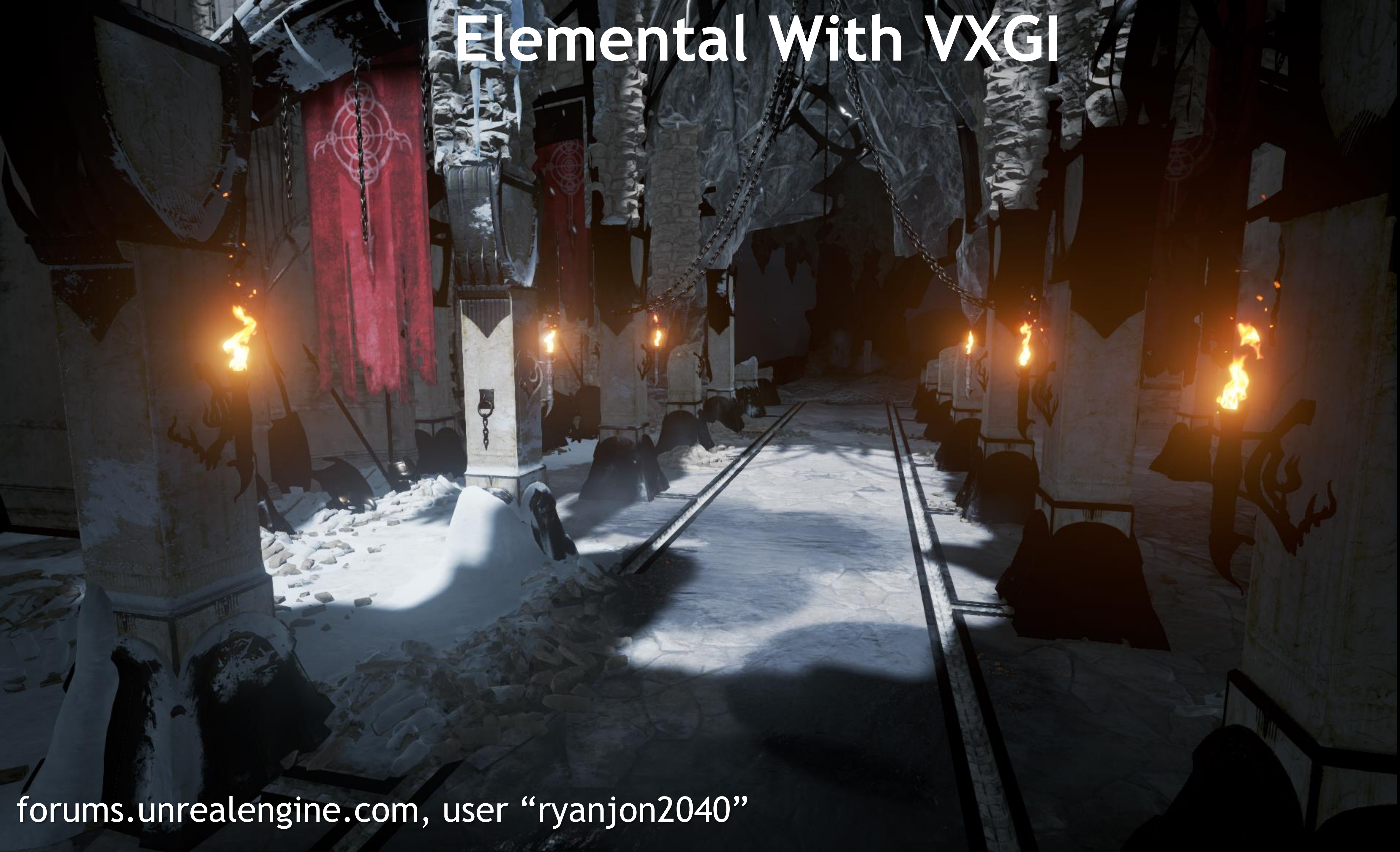
forums.unrealengine.com, user “rabellogp”

VXGI Emittance Voxels



forums.unrealengine.com, user “rabellogp”

Elemental With VXGI



forums.unrealengine.com, user “ryanjon2040”

Effects Cave

forums.unrealengine.com, user “Ad3ViLl”

Effects Cave With VXGI

forums.unrealengine.com, user “Ad3ViLl”

OUTLINE

- What is VXGI
- Algorithm Overview
- Engine Integration
- VXGI in UE4
- Quality and Performance
- Ambient Occlusion Mode
- Q&A

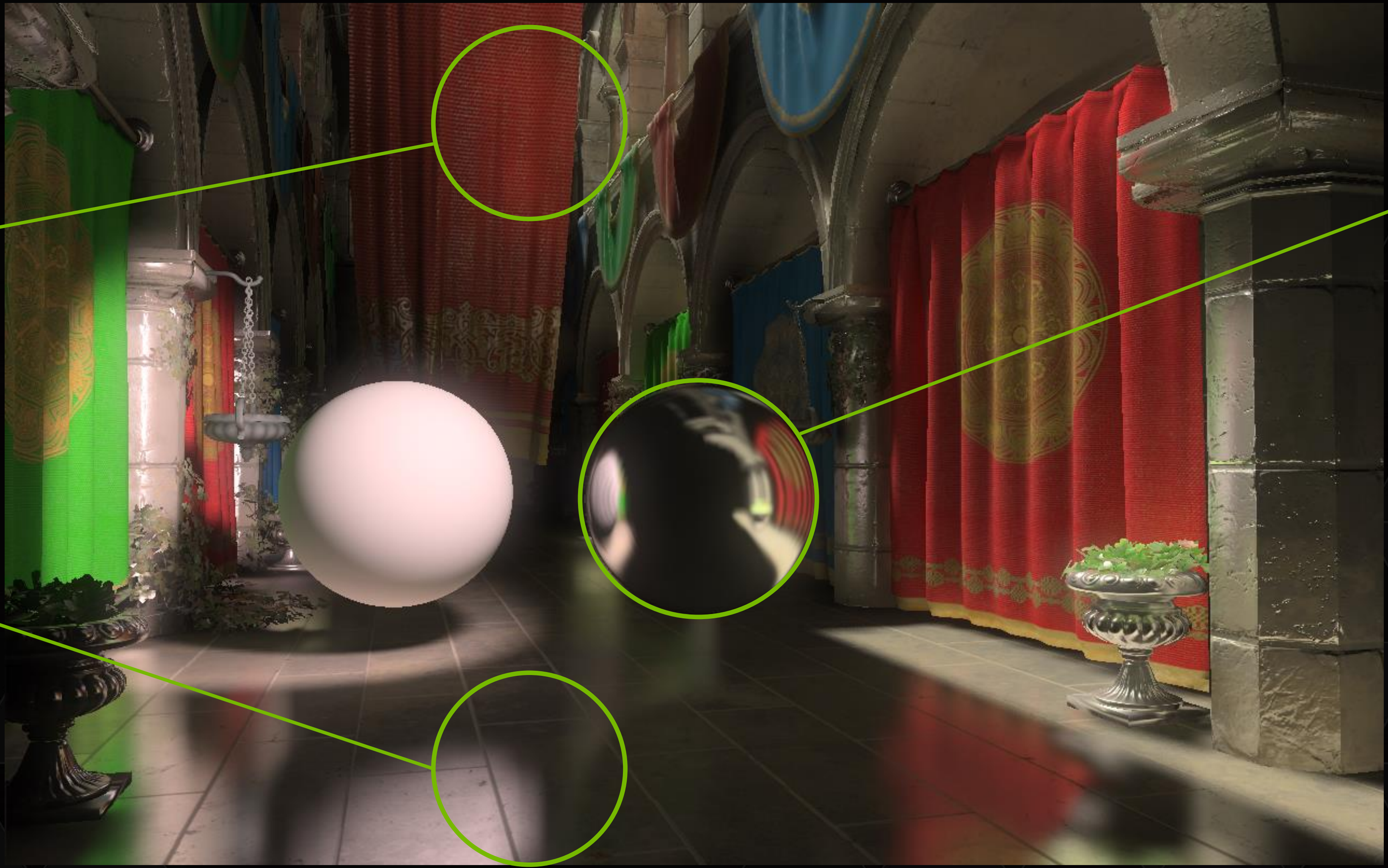
VOXELIZATION QUALITY ISSUES

- Voxelization aliasing
  - Moving lit objects sometimes flicker
  - Use supersampled emittance voxelization for small objects
  - Use temporal filtering to cancel the flicker
- Light quantization or saturation
  - RGBA8_UNORM emittance is used on non-Maxwell GPUs
  - Insufficient dynamic range to capture HDR lighting
  - Tune VoxelizationParameters::emittanceStorageScale

VOXELIZATION PERFORMANCE TIPS

- Use low-detailed meshes for voxelization
  - Disable tessellation or reduce tessellation factors
- Use a custom culling function with the voxelization GS
  - Cull triangles outside of light frustum or facing away from the light
  - Pass the function code to pGI->createVoxelizationGeometryShaderXX(...)
- Voxelize geometry for several lights at a time
- Only enable emittance supersampling for small moving objects

CONE TRACING QUALITY ISSUES

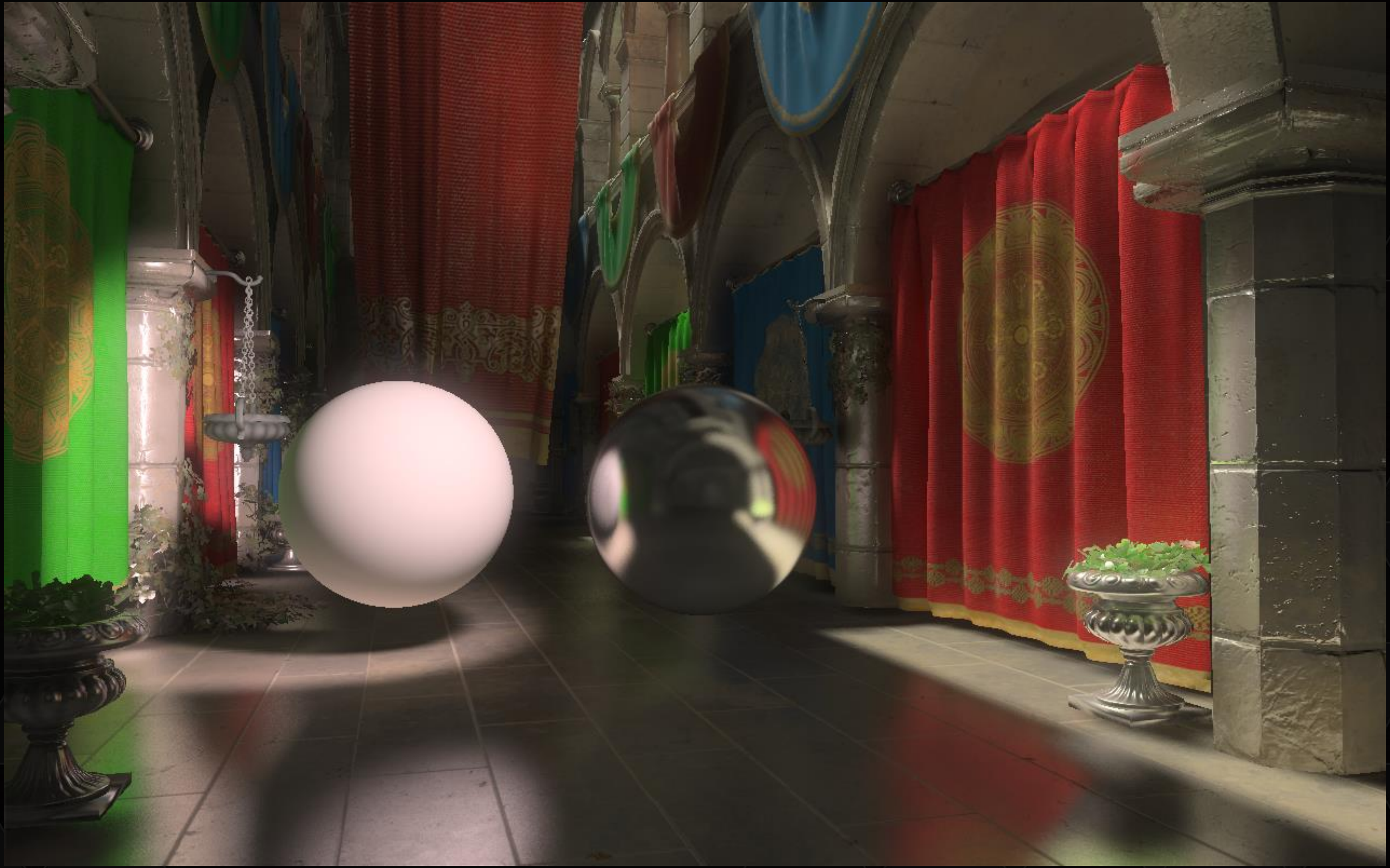


Light leaking

Voxels in reflections

Single bounce specular

IMPROVED TRACING QUALITY



SUMMARY OF TRACING ISSUES

- Light leaking
  - Light comes through walls or looks like SSS
  - Make walls thicker
  - Don’t voxelize light outside of potential visible area
- Visible voxels in specular reflections
  - Insufficient voxel resolution for mirror reflections
  - Make materials more rough or bumpy
  - Enable tangent jitter and temporal filtering
- Specular reflections are single bounce
  - Cone tracing only “sees” directly lit surfaces
  - Add constant ambient when voxelizing for emittance

  - Combine VXGI specular with other techniques

TRACING PERFORMANCE TIPS

- Use fewer diffuse cones and enable cone rotation
  - 4-8 cones is probably enough
- Use temporal filtering for diffuse and specular tracing
- Reduce the number of visible specular pixels
  - No tracing is done when gbufferNormal.a = 0.0
- Use the gbufferGeoNormal channel to speed up diffuse tracing
  - Surface detail will be preserved

PERFORMANCE EXAMPLE

- Scene: San Miguel - **3.0 M triangles**, revoxelized on every frame
- GPU: GeForce GTX 980, pre-release R349 driver

- Opacity voxelization: 6.9 ms + 1.5 ms for post-processing
- Emittance voxelization: 4.7 ms + 3.3 ms for post-processing
  - No supersampling, 1 directional light, FP16

- Diffuse tracing: 7.0 ms + 1.0 ms for interpolation
- -OR- Ambient tracing: 2.5 ms + 1.0 ms for interpolation
  - 1920x1080, 8 cones, trace every 4th pixel

- Specular tracing: 1.8 ms
  - Depends on visible geometry a lot

OUTLINE

- What is VXGI
- Algorithm Overview
- Engine Integration
- VXGI in UE4
- Quality and Performance
- Ambient Occlusion Mode
- Q&A

BONUS: VOXEL-BASED AO

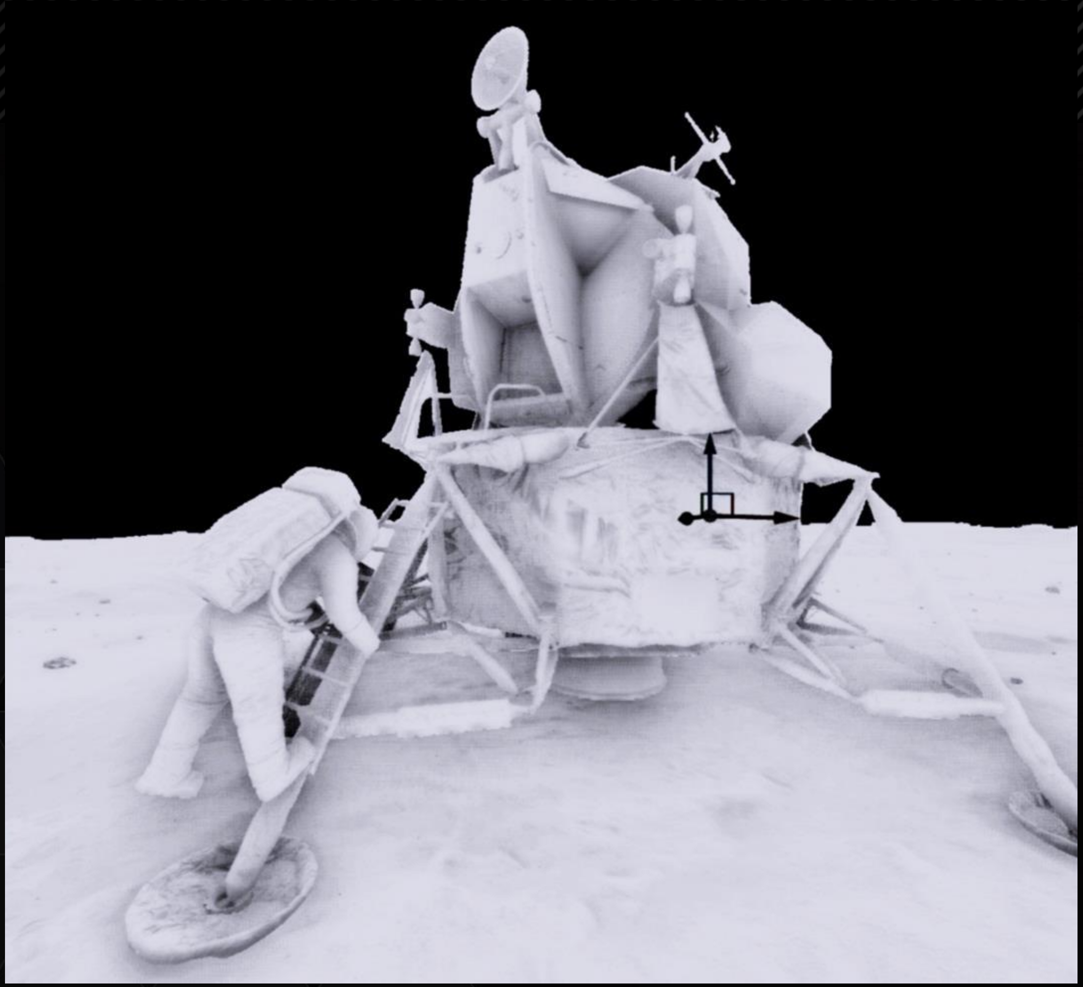
- Remove the emittance voxel textures
  - VoxelizationParameters::emittanceDirectionCount = NONE

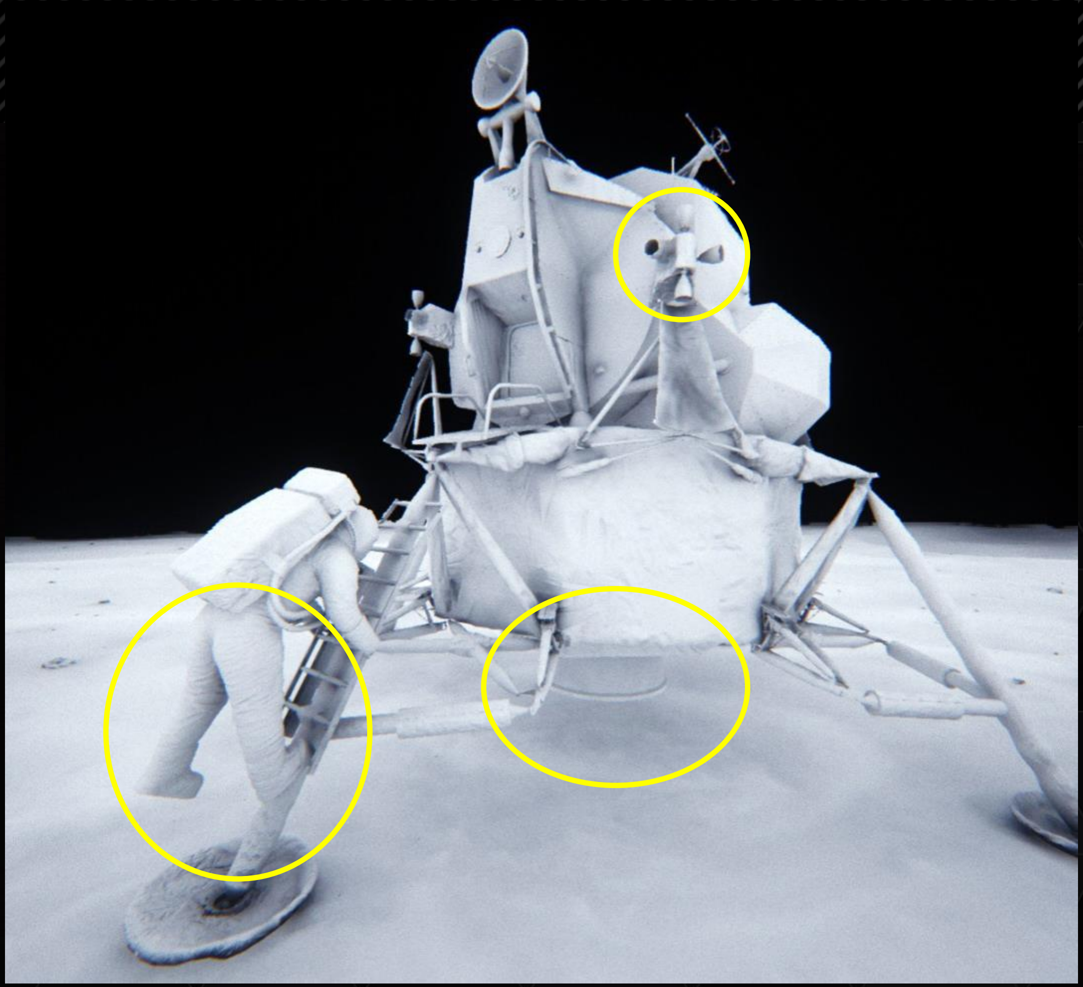
- Skip emittance voxelization and light injection
- Call pTracer->computeDiffuseChannel(…) to get the AO surface
  - DiffuseTracingParameters::ambientRange controls effect locality

- Compared to full GI…
  - Tracing is about 3x cheaper
  - Easier to integrate into apps

- Compared to SSAO…
  - World-space, stable AO effect

SSAO VS. VXGI AO





DEMO: VOXEL-BASED AO

SUMMARY

- VXGI provides an efficient real-time GI solution
  - Tuning is required to mitigate quality issues and make it work fast
- VXGI supports all DX11 GPUs
  - Maxwell produces higher quality results and works faster
- DX11 version and UE4 integration available now
- OpenGL version is being worked on

GPU TECHNOLOGY CONFERENCE

OUTLINE

- What is VXGI
- Algorithm Overview
- Engine Integration
- VXGI in UE4
- Quality and Performance
- Ambient Occlusion Mode
- Q&A

GPU TECHNOLOGY CONFERENCE

THANK YOU!

JOIN THE CONVERSATION

#GTC15