Advances in Real-Time Voxel-Based GI

Alexey Panteleev, Senior Developer Technology Engineer
Rahul Sathe, Senior Developer Technology Engineer
March 21, 2018

Booth #223 - South Hall
www.nvidia.com/GDC
Recap on VXGI

- Voxel Global Illumination
  - Inspired by Sparse Voxel Octree Global Illumination (SVOGI)
  - Clip-map used instead of octree

- Fully dynamic scene support
  - Voxelizing a game-like scene from scratch takes only a few ms
  - Supports multi-bounce GI through a temporal feedback loop on irradiance
Cornell Box Scene
VXGI Algorithm: Voxelization

Opacity

Emittance / Light
VXGI Algorithm: Tracing

Diffuse

Rough Specular

Fine Specular

![Image of VXGI Algorithm: Tracing](image)
Results of Cone Tracing

Indirect Diffuse

Indirect Specular
Final Result

Direct Lighting Only

Direct and Indirect
Voxel Ambient Occlusion

- VXAO

- Easier to compute than full global illumination
  - No light processing, only opacity

- Easier to integrate into engines
  - No materials or lights during voxelization

- Looks better than screen-space techniques
  - World-space, not screen-space
  - Best if combined with small-scale SSAO
Area Lights with VXGI
Better Area Lights with VXGI 2.0
Voxel Area Lighting
Area Lighting

- A hot topic of interest in the industry and academia
Area Lighting

- A hot topic of interest in the industry and academia

- Linearly Transformed Cosines
  - A new technique invented in 2016
Area Lighting

- A hot topic of interest in the industry and academia

- Linearly Transformed Cosines
  - A new technique invented in 2016
  - Impressive lighting for area lights
  - Complexity is $O(n)$
    - $n$ is # of edges

https://eheitzresearch.wordpress.com/415-2/
Area Lighting

- A hot topic of interest in the industry and academia

- Linearly Transformed Cosines
  - A new technique invented in 2016
  - Impressive lighting for area lights
  - Complexity is $O(n)$
    - $n$ is # of edges

- But the occlusion is missing

https://eheitzresearch.wordpress.com/415-2/
Voxel Area Lighting

- VXAL
Voxel Area Lighting

- VXAL
- Distribute cones over the area light
Voxel Area Lighting

- VXAL
- Distribute cones over the area light
- Occlusion
  - Large Scale: Cone tracing similar to VXAO
  - Small Scale: Screen space shadows
Voxel Area Lighting

- VXAL
- Distribute cones over the area light
- Occlusion
  - Large Scale: Cone tracing similar to VXAO
  - Small Scale: Screen space shadows
- Irradiance
  - Diffuse and Specular
  - Linearly Transformed Cosines
Voxel Area Lighting

- VXAL
- Distribute cones over the area light
- Occlusion
  - Large Scale: Cone tracing similar to VXAO
  - Small Scale: Screen space shadows
- Irradiance
  - Diffuse and Specular
  - Linearly Transformed Cosines
- Modulate irradiance with occlusion
Voxel Area Lighting

- VXAL
- Distribute cones over the area light
- Occlusion
  - Large Scale: Cone tracing similar to VXAO
  - Small Scale: Screen space shadows
- Irradiance
  - Diffuse and Specular
  - Linearly Transformed Cosines
- Modulate irradiance with occlusion
- Apply material parameters like albedo, composite into the final view
Area Lights

- Multiple area lights supported
  - Rectangular in shape
  - Textured or Solid color
  - Each light has some rendering cost
  - Dynamic textures are not free

- Wide range of quality settings
  - Tracing resolution: half-res to quarter-res
  - Occlusion quality: use more or fewer cones per unit angle
    - Actual number of cones is adaptive and varies per pixel
Linearly Transformed Cosines

- BRDF
  - How much light transfers from incoming directions to outgoing directions

https://eheitzresearch.wordpress.com/415-2/
Linearly Transformed Cosines

- **BRDF**
  - How much light transfers from incoming directions to outgoing directions

- **Shading:**
  - Integrate BRDF over the light’s spherical projection

https://eheitzresearch.wordpress.com/415-2/
Linearily Transformed Cosines

- BRDF
  - How much light transfers from incoming directions to outgoing directions

- Shading:
  - Integrate BRDF over the light’s spherical projection

- Analytic solutions exist, but only for simple BRDFs
  - e.g. Phong, but very expensive

https://eheitzresearch.wordpress.com/415-2/
Linearly Transformed Cosines

- Integrals invariant under linear transformations
  - Transform to the distribution
  - Transform to the polygon
  - Results are same

\[
\int_{P_1} D_1(\omega) \, d\omega = E[I] \\
\int_{P_2} D_2(\omega) \, d\omega = E[I]
\]

\[
\int_{P_1} D_1(\omega) \, d\omega = \int_{P_2} D_2(\omega) \, d\omega
\]

https://eheitzresearch.wordpress.com/415-2/
Linearly Transformed Cosines

- Integrals invariant under linear transformations
  - Transform to the distribution
  - Transform to the polygon
  - Results are same

- Parameterized linear transforms
  - View Angle & Roughness
  - Pre-computed and stored in textures

\[
\int_{P_1} D_1(\omega) \, d\omega = E[l] \quad \int_{P_2} D_2(\omega) \, d\omega = E[l]
\]

\[
\int_{P_1} D_1(\omega) \, d\omega = \int_{P_2} D_2(\omega) \, d\omega
\]

https://eheitzresearch.wordpress.com/415-2/
Linearily Transformed Cosines

- Integrals invariant under linear transformations
  - Transform to the distribution
  - Transform to the polygon
  - Results are same

- Parameterized linear transforms
  - View Angle & Roughness
  - Pre-computed and stored in textures

- Prefiltered textures for textured lights

\[
\int_{P_1} D_1(\omega) \, d\omega = E[l] \quad \int_{P_2} D_2(\omega) \, d\omega = E[l]
\]

\[
\int_{P_1} D_1(\omega) \, d\omega = \int_{P_2} D_2(\omega) \, d\omega
\]

https://eheitzresearch.wordpress.com/415-2/
VXAL API

- Set the area lights
  - Position, orientation, size, color, texture, etc.
### VXAL API

- **Set the area lights**
  - Position, orientation, size, color, texture, etc.

- **Set the area lights tracing parameters**
  - Occlusion quality, screen-space shadows, temporal filtering, etc.
VXAL API

- Set the area lights
  - Position, orientation, size, color, texture, etc.

- Set the area lights tracing parameters
  - Occlusion quality, screen-space shadows, temporal filtering, etc.

- Set the view information
  - Projection matrices, viewports, etc.
  - Provide G-buffer channels
VXAL API

- Set the area lights
  - Position, orientation, size, color, texture, etc.
- Set the area lights tracing parameters
  - Occlusion quality, screen-space shadows, temporal filtering, etc.
- Set the view information
  - Projection matrices, viewports, etc.
  - Provide G-buffer channels
- Returns
  - Diffuse Irradiance channel
  - Specular Irradiance channel
Future Work

- Support other types of area lights
  - Maybe disk or line lights
- Improve image quality
  - Near-field occlusion
  - Flickering in low-res modes
References

VXAL DEMO
VXGI 2.0 New Features
(besides VXAL)
One-Pass Voxelization

- VXGI 1.0:
  - Separate voxelization passes for opacity and emittance
  - Twice the CPU cost, almost twice the GPU cost - on top of other rendering passes

- VXGI 2.0:
  - Can do everything in one pass
  - Or multiple, up to the application
  - Each pass adds some opacity and emittance to the voxel volume
Custom G-Buffer Layouts

- VXGI 1.0 requires a specific data layout and projection
  - Hardware depth, linear normals, roughness in normal.w
  - Planar projection only
- VXGI 2.0 takes HLSL code to load geometry info for a pixel
  - Anything that resolves to a position and normal will do
  - VRWorks MRS and LMS projections, or anything else
  - Many tracing settings can vary per-pixel
View Reprojection

- VXGI 1.0 supports reusing lighting information from the previous frame
  - Temporal reprojection or temporal filtering
- VXGI 2.0 adds reuse between views in the same frame
  - Compute lighting for the left eye
  - Reproject matching surfaces to the right eye
  - Fill the holes
  - No limits on the number of views
Simultaneous VXGI, VXAO, and VXAL

- VXGI 1.0 had two modes
  - AO mode: ambient occlusion channel is produced
  - GI mode: diffuse channel is produced with ambient lighting added

- VXGI 2.0 changes how the GI mode behaves
  - Diffuse channel has AO in the alpha component
  - Can compose as necessary on the application side

- VXAL is independent
  - Separate API
  - Same behavior in GI and AO modes
Other Improvements
Simpler Voxel Formats

- 3D or 6D opacity replaced by scalar
  - Same quality, better performance
  - Can do fractional opacity materials now

- Multiple emittance formats replaced by single FLOAT16
  - With a functional detour for GPUs which do not support FP16 atomics
  - Occlusion-only mode with no emittance can still be enabled
Simpler and More Flexible Materials

- Fewer controls from the CPU side
  - Most of MaterialInfo members removed
  - Only Adaptive Material Sampling Rate is still there

- More powerful on the shader side
  - Fractional opacity, variable per-voxel
  - Two-sided materials with different reflected colors
Simpler Tracing Controls

- VXGI 1.0:
  - numCones, coneAngle, normalGroupingFactor, …, …

- VXGI 2.0:
  - quality, softness, directionalSamplingRate, …
    - Adjust Quality and Softness to get an acceptable look
    - Then adjust the sampling rate and temporal filtering to get a usable noise level
There’s More...

- Separate SSAO pass
- Support for pre-view translation
- Improved upscaling and temporal filters
- Non-cubic voxel volumes
- Reduced light leaking
- Fine control over D3D extensions
- Improved NVRHI
Summary
Summary

▪ New version: VXGI 2.0

▪ VXAL: High-quality area lighting with shadows

▪ Lots of smaller new features

▪ Better performance than VXGI 1.0

▪ Available soon
  o Mid-April 2018
  o SDK and Unreal Engine 4 integration
Thank you!

- Questions?

- Contact us:
  - alpanteleev@nvidia.com
  - rsathe@nvidia.com