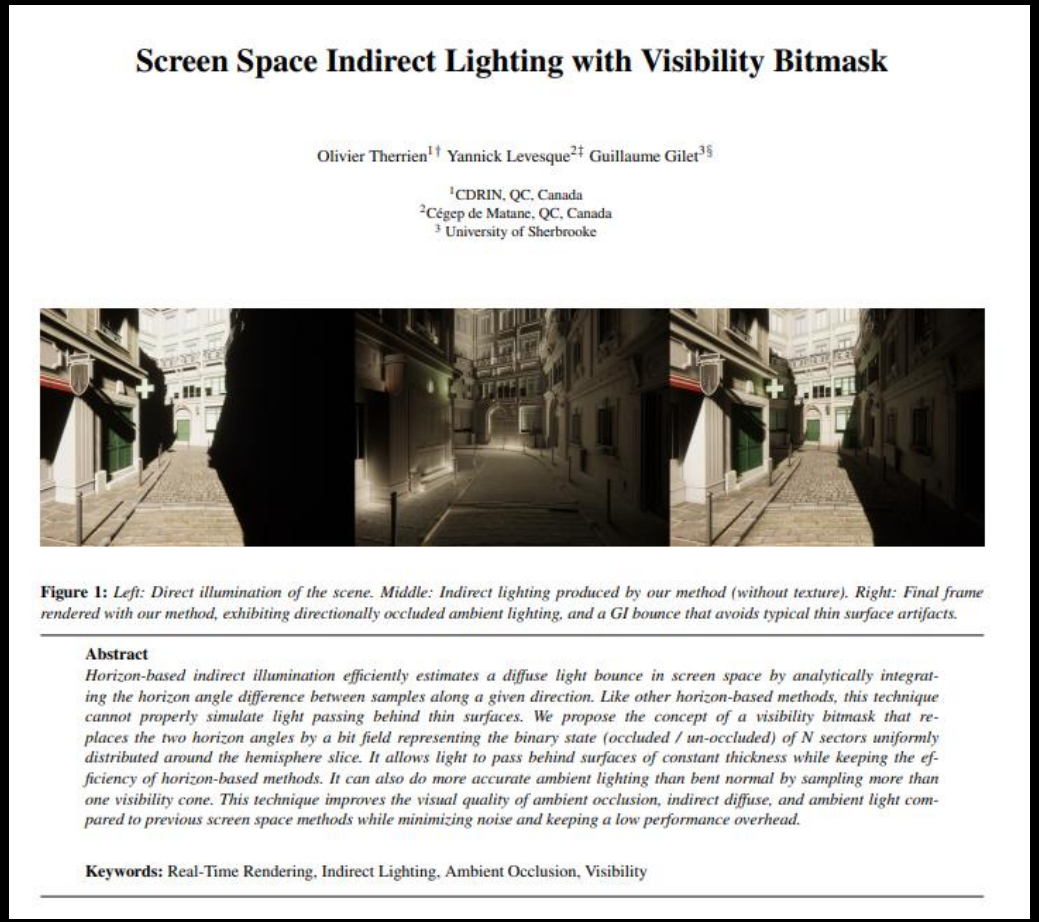
CORIN

Centre de développement
et de recherche en
intelligence numérique

Screen Space Indirect Lighting with Visibility Bitmask

Screen Space Indirect Lighting with Visibility Bitmask

- A multi-institution collaboration:
  - Olivier Therrien (CDRIN)
  - Yannick Levesque (Cégep de Matane)
  - Guillaume Gilet (University of Sherbrooke)
- Special thanks to Peter Shirley
- Publication links:
  - https://link.springer.com/article/10.1007/s00371-022-02703-y
  - https://arxiv.org/abs/2301.11376



Screen Space Indirect Lighting with Visibility Bitmask

Olivier Therrien[1†] Yannick Levesque[2‡] Guillaume Gilet[3§]

[1]CDRIN, QC, Canada
[2]Cégep de Matane, QC, Canada
[3] University of Sherbrooke

Figure 1: *Left: Direct illumination of the scene. Middle: Indirect lighting produced by our method (without texture). Right: Final frame rendered with our method, exhibiting directionally occluded ambient lighting, and a GI bounce that avoids typical thin surface artifacts.*

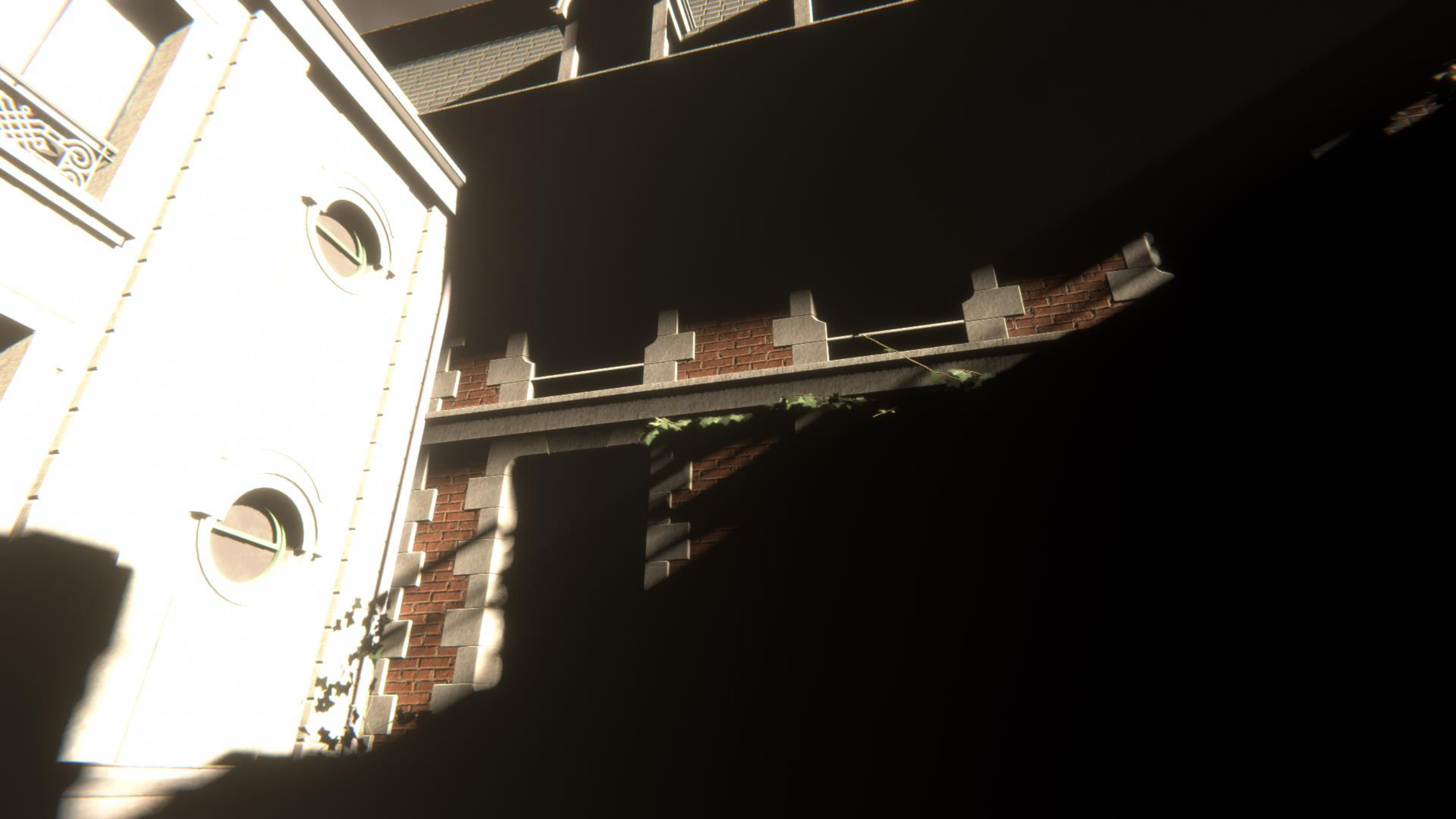
Abstract

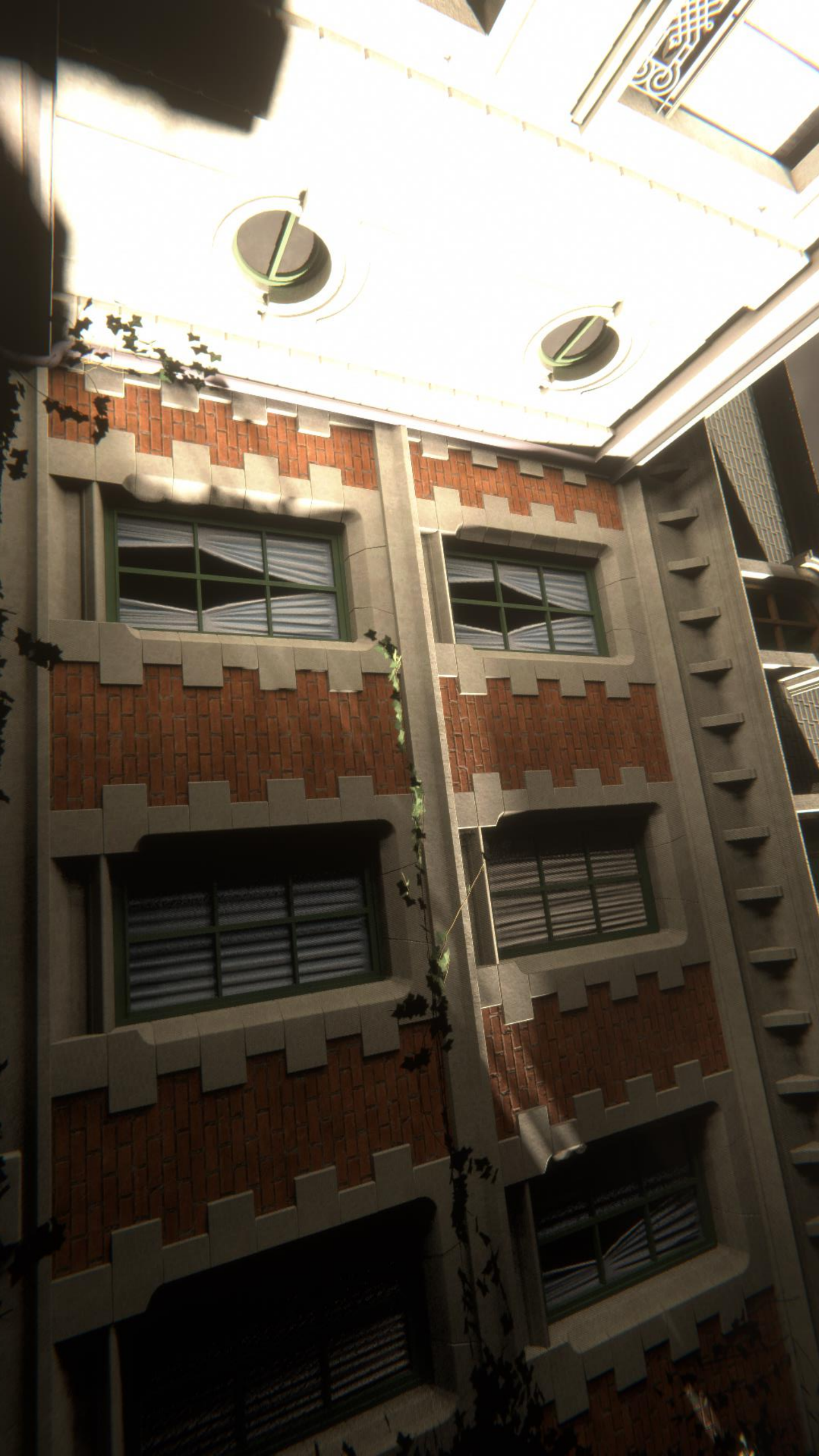
Horizon-based indirect illumination efficiently estimates a diffuse light bounce in screen space by analytically integrating the horizon angle difference between samples along a given direction. Like other horizon-based methods, this technique cannot properly simulate light passing behind thin surfaces. We propose the concept of a visibility bitmask that replaces the two horizon angles by a bit field representing the binary state (occluded / un-occluded) of N sectors uniformly distributed around the hemisphere slice. It allows light to pass behind surfaces of constant thickness while keeping the efficiency of horizon-based methods. It can also do more accurate ambient lighting than bent normal by sampling more than one visibility cone. This technique improves the visual quality of ambient occlusion, indirect diffuse, and ambient light compared to previous screen space methods while minimizing noise and keeping a low performance overhead.

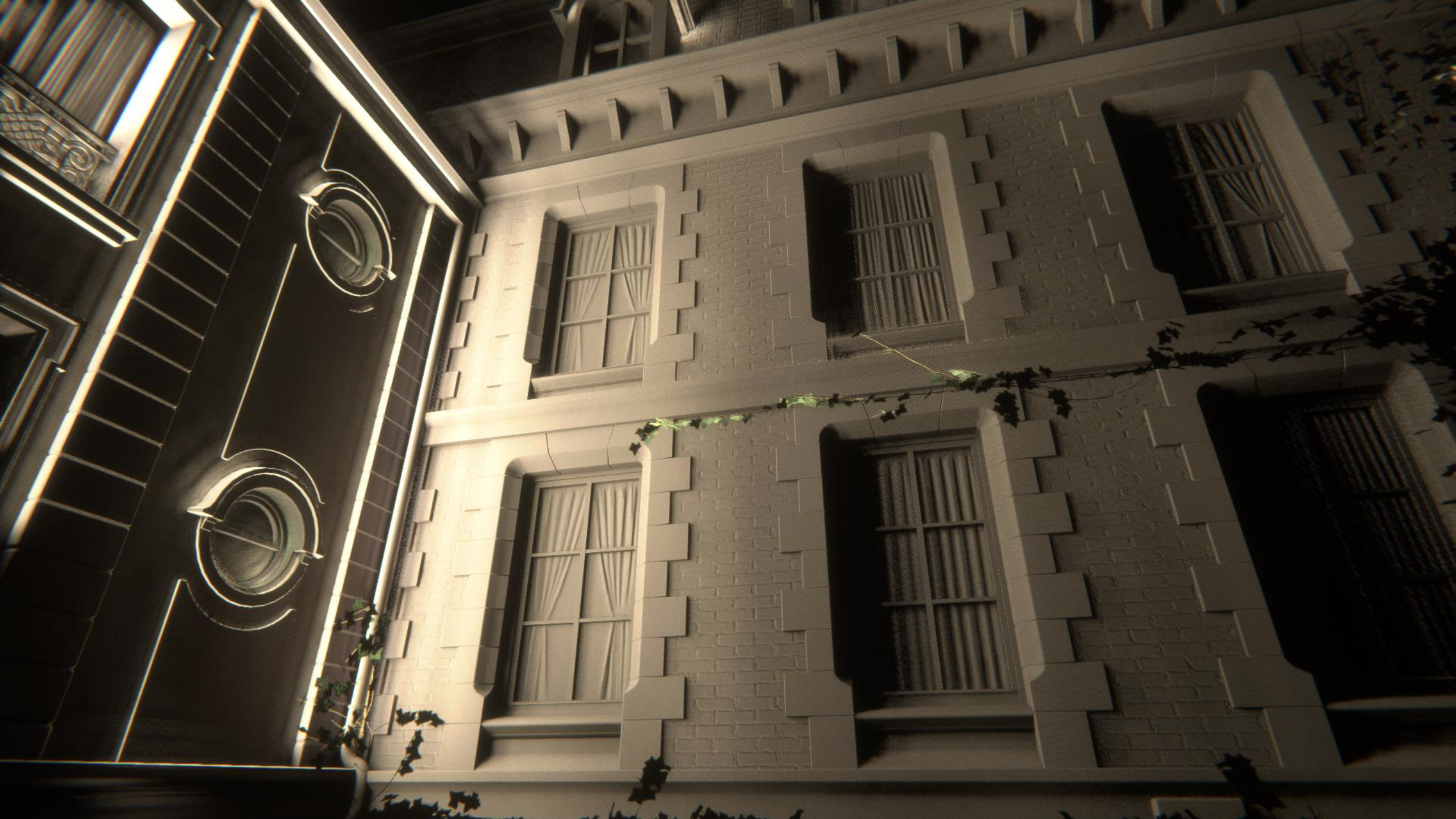
Keywords: Real-Time Rendering, Indirect Lighting, Ambient Occlusion, Visibility


12345
678910
Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu Vv Ww Xx Yy Zz


12345
678910
Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu Vv Ww Xx Yy Zz







Motivation

- Screen space?! Is this 2012?
  - Ray tracing is still expensive, and requires deep integration with the game
  - Screen space is fast, and it’s just a post-process
  - Best in class GI frameworks still use some screen space (Lumen, AMD GI-1.0)

- Instant lighting changes
- Low noise -> No ghosting, smearing or blur
- Preserves sharp details
- Mod friendly (our method is now in the RTGI Reshade shader)

Existing work: SSGI

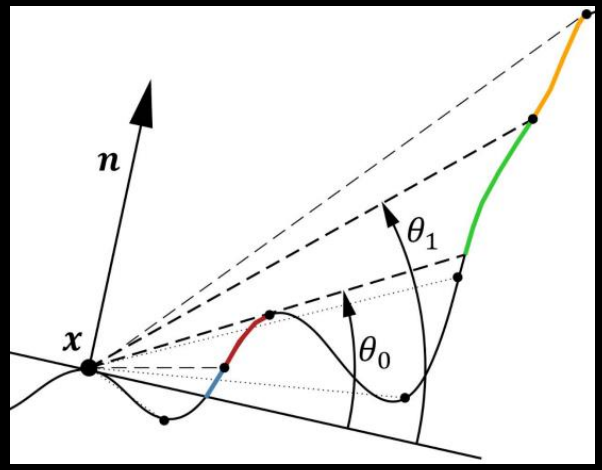
- Unity and UE4 have an SSGI implementation
- Based on screen tracing similar to SSR
  - Hi-Z Screen Space Tracing
- Very noisy: Most rays pass behind surfaces or escape outside the screen
  - Maximum 1 sample per ray (per pixel)

SSGI

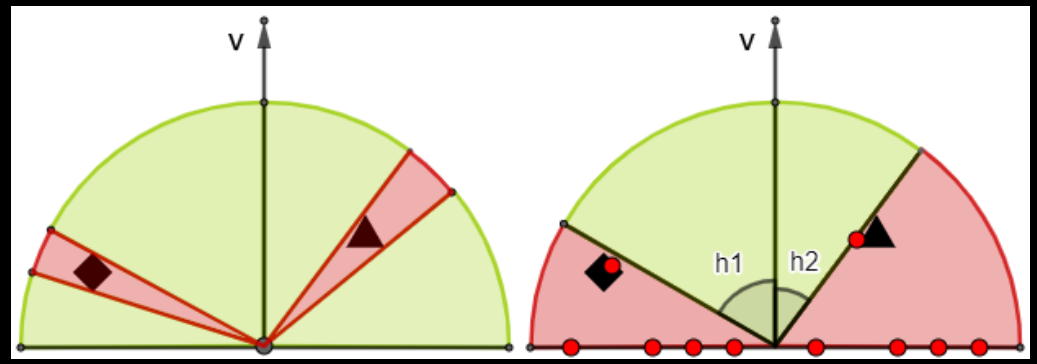
Our method

Existing work: HBIL [Mayaux 2018]

- Horizon-based technique similar to HBAO/GTAO
  - Low noise compared to Screen Space Tracing
- Works by integrating the light between two horizon angles
- Caveat: Cannot handle thin surfaces properly
  - All the space below the maximum horizon angle h1/h2 gets integrated

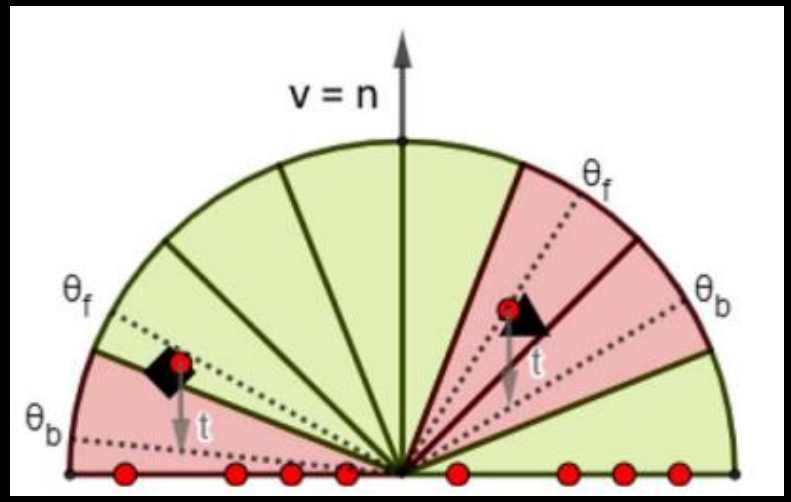


(Source: [Mayaux 2018])



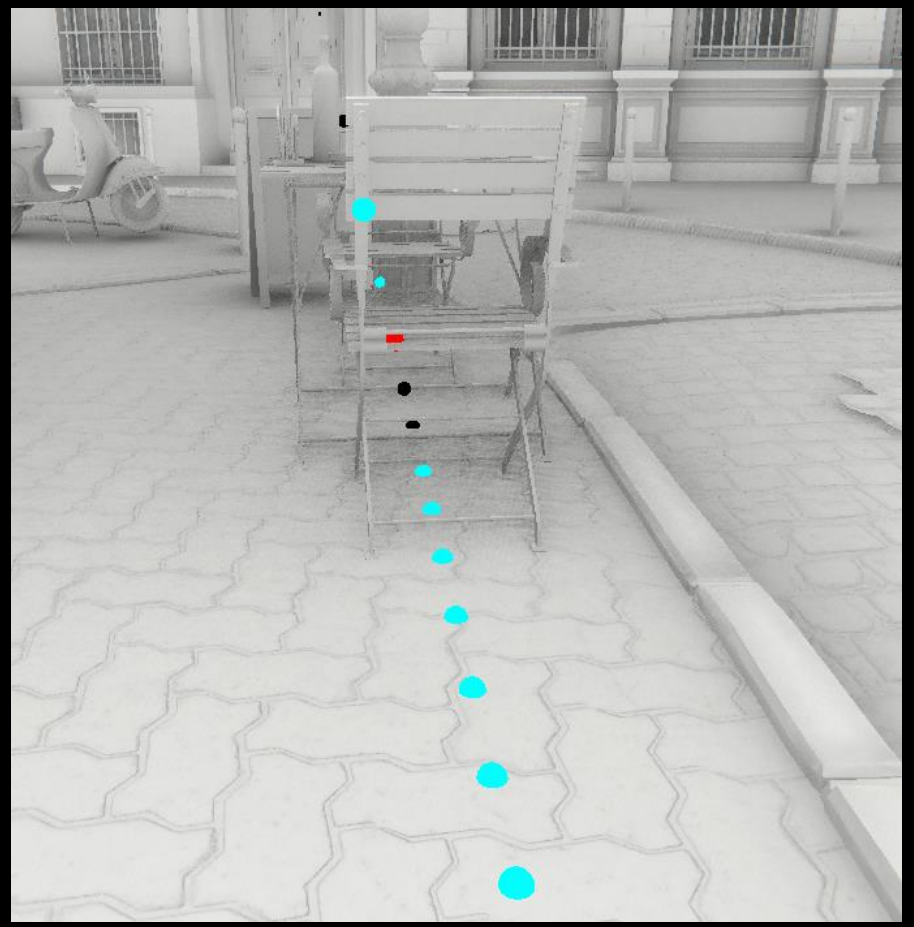
Our method: Visibility Bitmask

- We replace horizon angles by a bitmask
- Surfaces now have a thickness (constant)
  - Thickness is used to compute “back-face” angle (lower black dotted lines)
- Sectors (bits) that are in-between front-face and back-face angles get blocked (set to 1)
  - Multiple sectors can be blocked at once by a single sample via bitwise operations
- For ambient occlusion we simply count the number of set bits divided by bitmask size



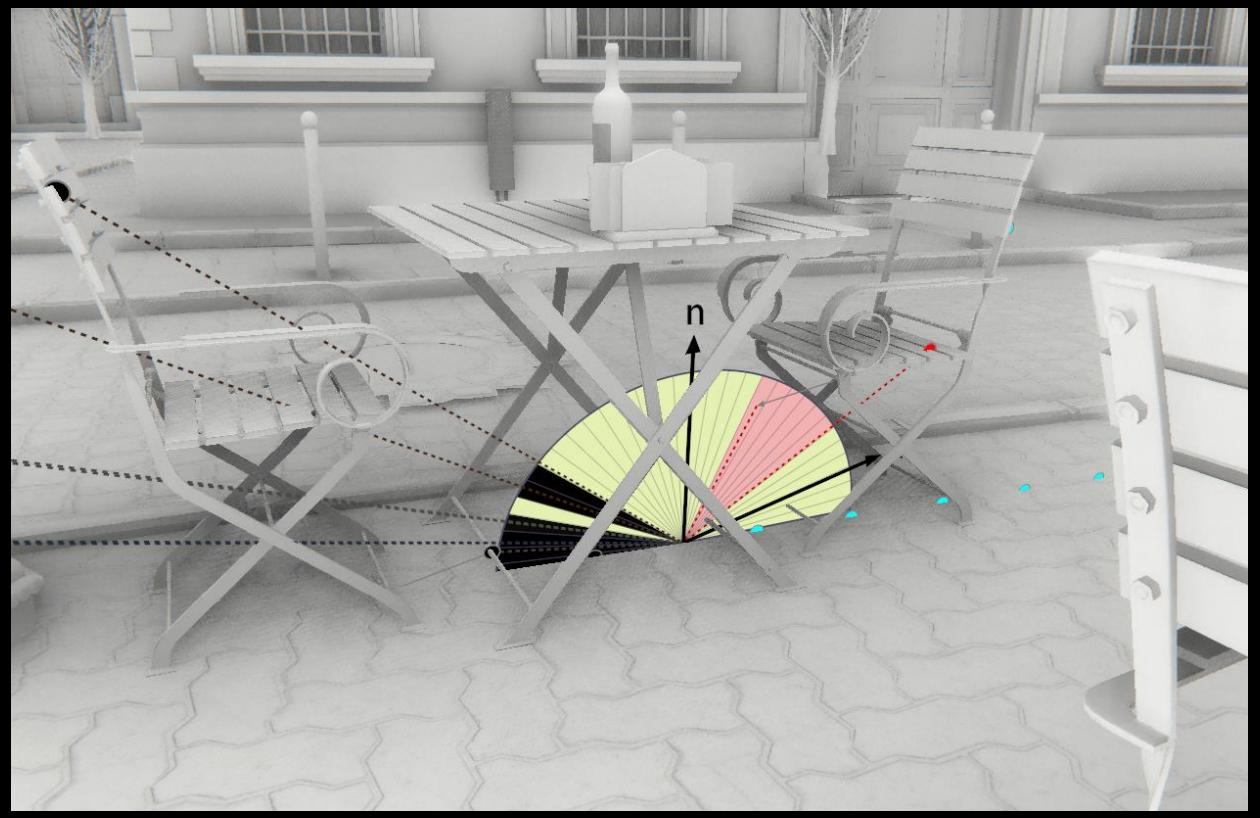
Screen space sampling

- A sampling direction is chosen randomly per pixel
- Samples are taken non-uniformly on the screen (more close to pixel, less farther away)
- Sample debug colors
  - Cyan: Rejected (sector already occluded)
  - Red: Occlusion only (not facing pixel)
  - Black or other color: Can contribute lighting
- Most samples do not contribute! (cyan)



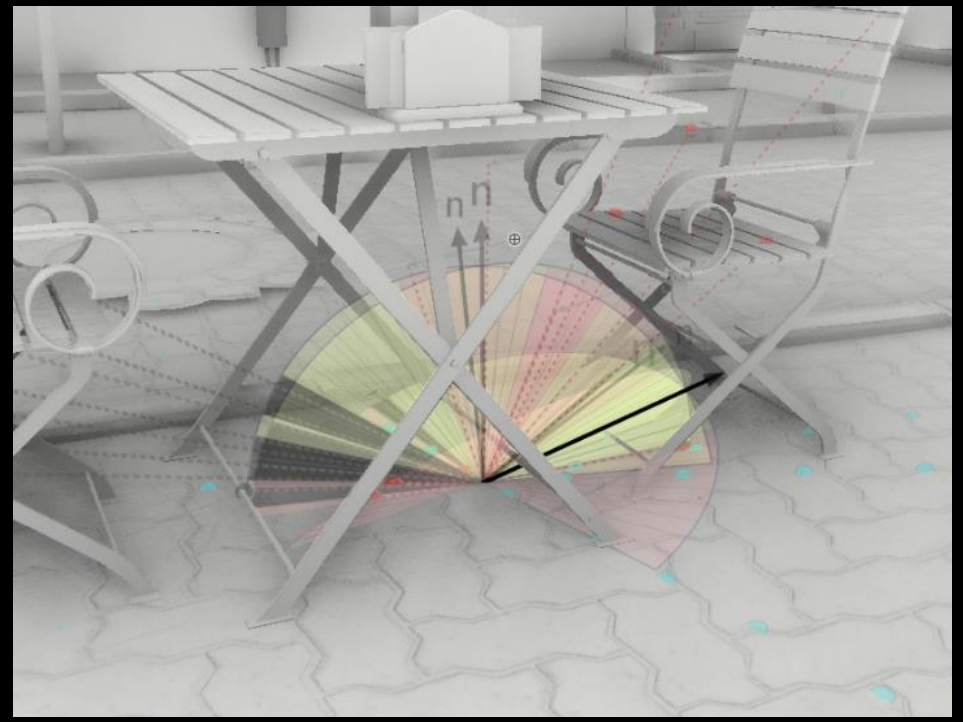
Sampling (from another point of view)

- Slice is aligned to the sampling line
- Samples have front-face and back-face projection lines that can block sectors
- Surface normal is rarely on the slice, we compute a projected normal *n*
- Samples are taken each side of view vector, not normal



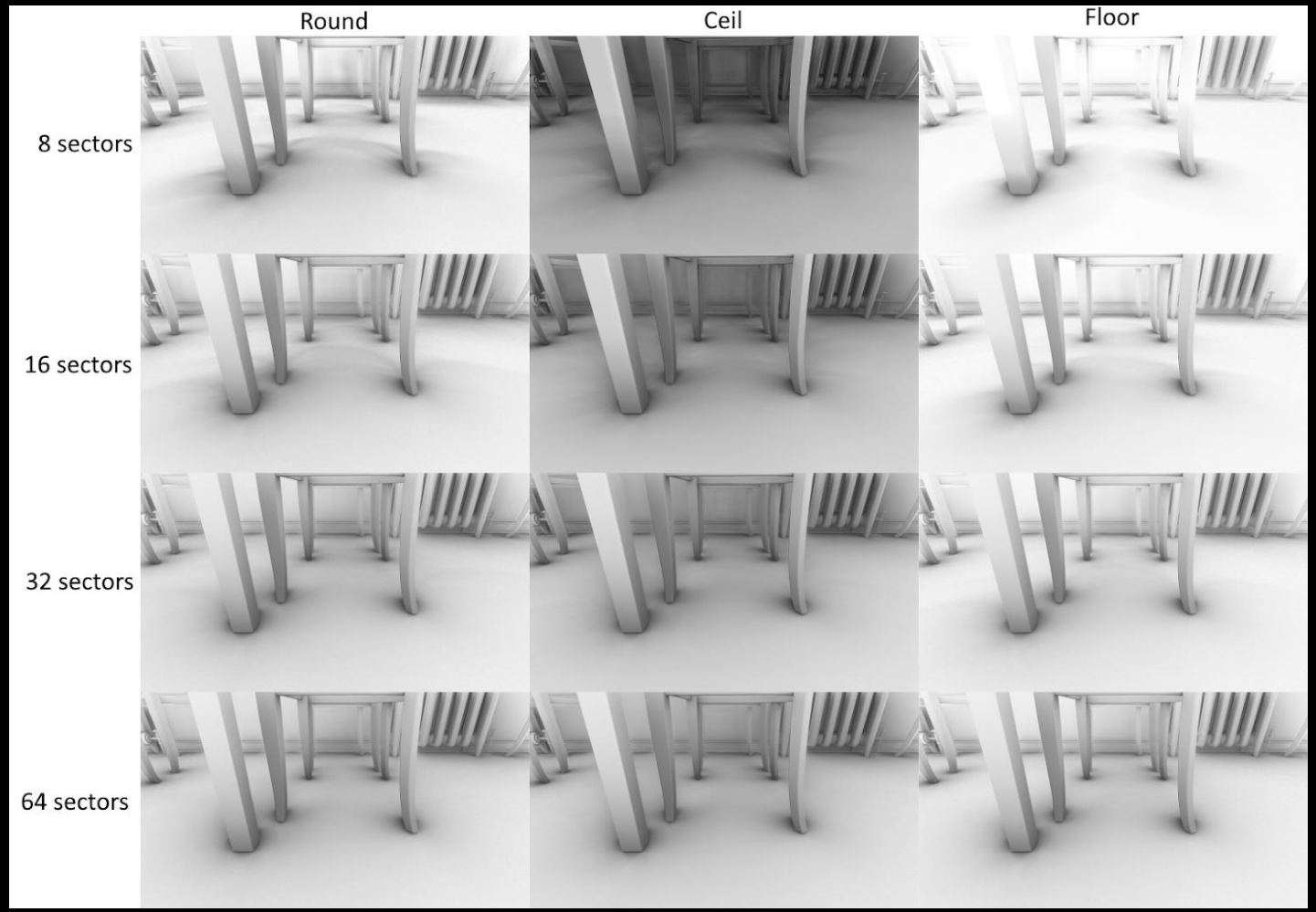
Random slice sampling direction

- Slice direction is taken at random for each pixel according to a gradient noise
- Can also have more than 1 direction per pixel (multi-sampling)
- Rotation is done in 2D on the screen
  - Slice projected normal can get very different from the surface normal



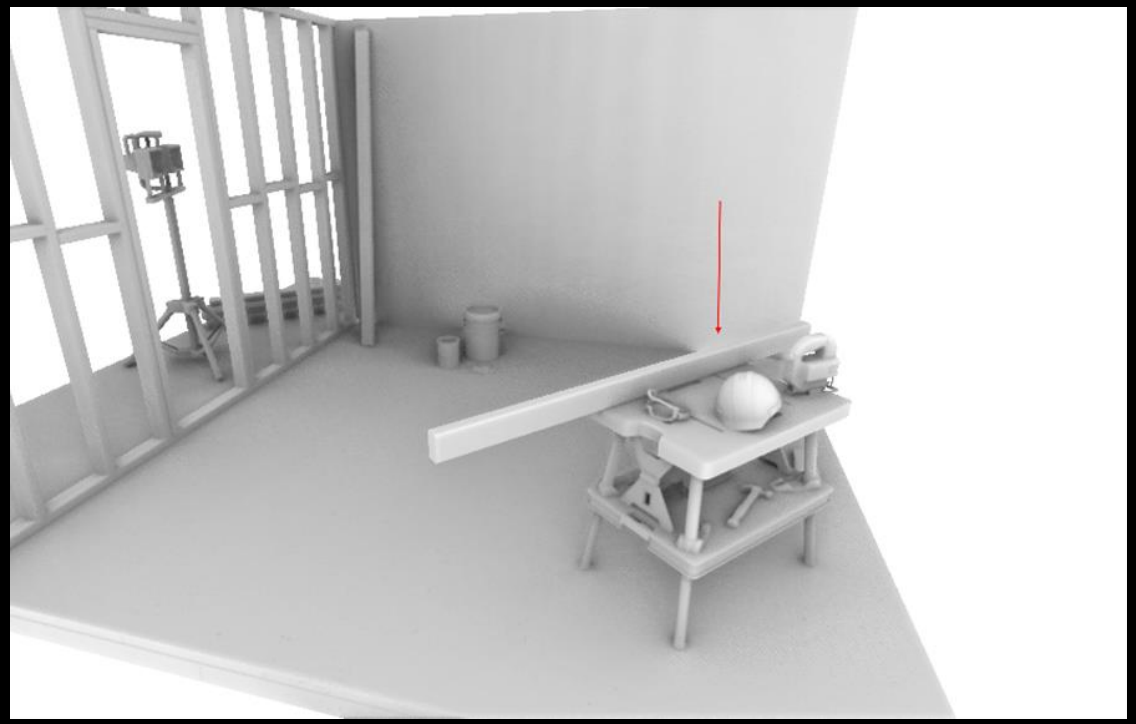
Bitmask size & rounding function

- More sectors is better for quality, but 32 is good enough for AO and indirect diffuse
- Rounding functions:
  - Round: Sample covers at least half a sector (most consistent one)
  - Ceil: Sample touches the sector
  - Floor: Sample covers the entire sector



Limitation: Light leaks from hidden surfaces

- Typical problem with single layer depth buffer
- Thickness is known (constant) but not the distance to next surface (or if there is a next surface)
  - Algo assumes there is nothing behind (causes missing occlusion)



Performance (AO, RTX 2080)

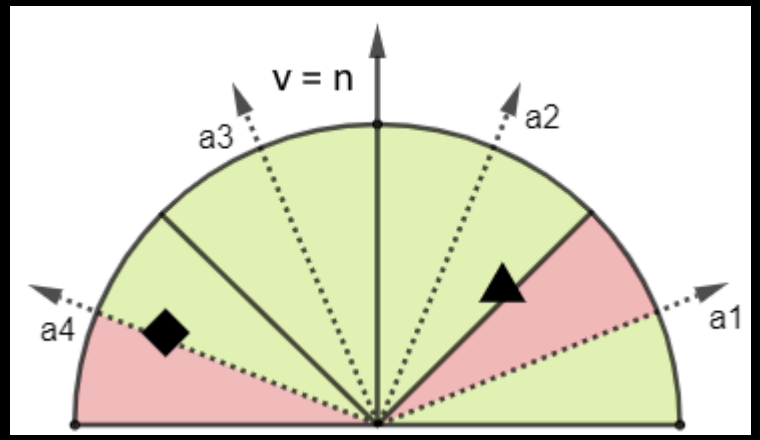
- Slight ALU overhead
  - Not very noticeable because the algorithm is bandwidth limited
  - Need to use acos() for every sample. It's a bit more expensive even with fast approximation

Radius	Sample Count	GTAO	Our Method
0.8	8	0.49 ms	0.51 ms
1	12	0.75 ms	0.77 ms
1	16	0.95 ms	0.97 ms
2	16	1.12 ms	1.13 ms
3	16	1.12 ms	1.13 ms

- Increasing the sampling radius is more costly, less samples reside in cache
  - Large radius tend to hide the ALU cost even more as bandwidth gets even more the limiting factor

Directional occlusion of ambient light

- Compatible with any ambient source
  - Spherical harmonics probes
  - Reflection probes
  - Irradiance cubemap
  - Others
- Sample the ambient source in X directions
  - a1 .. a4: We divide the hemisphere in 4 sub-regions and sample in the center direction
  - Multiply sampled light color by the ratio of un-occluded sectors per sub-region over sector count per sub-region



Directional occlusion (reflection probes)

- There is no diffuse bounce in those images (only ambient)
- Most games just multiply ambient source by pixel AO (uniform AO)
  - Fails to reconstruct directional color changes and indirect shadows
- Multisampling a detailed ambient source like reflection probes and apply occlusion directionally (per sample) gives much richer ambient light
  - Directional light changes are very noticeable
  - Can create large indirect soft shadows (column)
- Please do this in your game ☺
  - Much cheaper to compute than a GI bounce, yet it accounts for most of the “Wow” effect

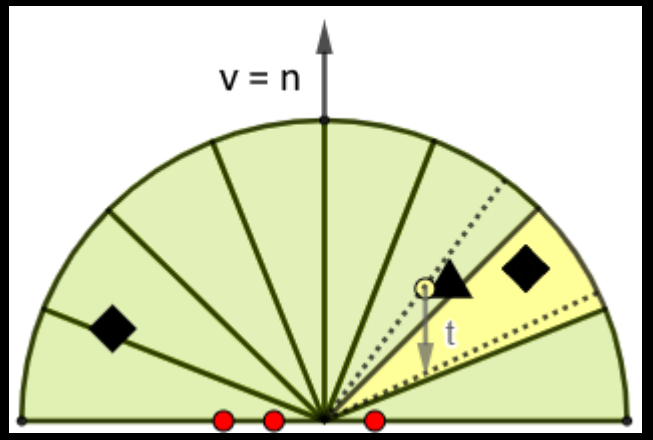
Uniform AO

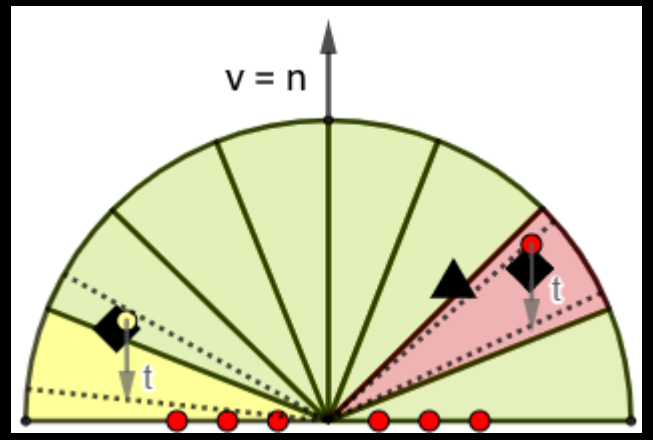
Directional AO

Directional AO + Indirect Diffuse

Indirect diffuse algorithm

- Top: Yellow sample intersects one un-occluded sector and can contribute lighting. The sector is set to an occluded state for subsequent samples
- Bottom: Sampling continues and a new object on the right is found, but it intersects an already occluded sector, so it cannot contribute lighting. The yellow sample on the left crosses an un-occluded sector and can contribute





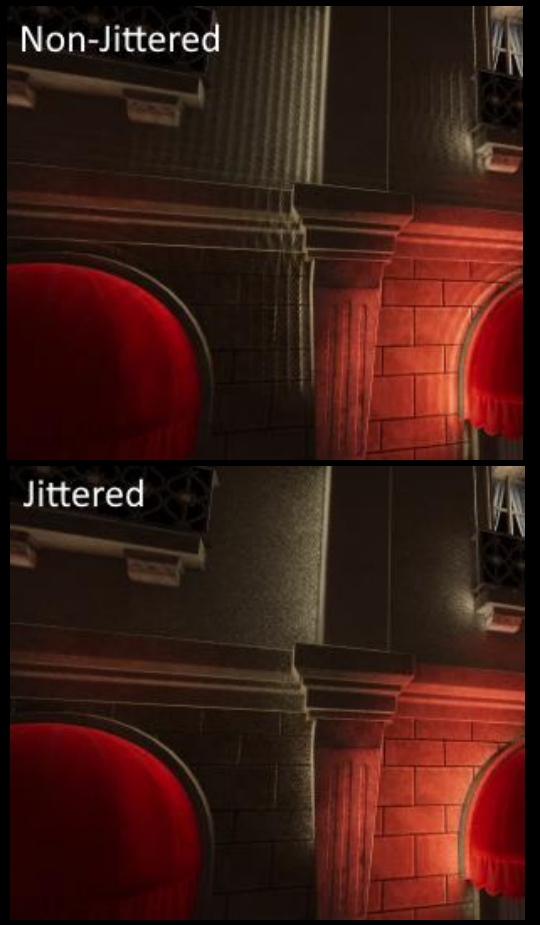
Direct lighting only

Indirect diffuse (final result)

Indirect diffuse (light only)

Banding artifact

- Banding can appear when source of light is intense and small
  - This is dependent on the step size: distance between samples becomes visible
  - Some pixels find the light source, others miss it
- The solution we found is jittering the steps for each pixel
  - Reusing samples between pixels can also help
  - More research: ideally we would want to identify important surfaces on the screen and make sure every pixel samples it



Limitation: Out of screen light

- Light needs to be on screen to contribute a bounce
  - Especially visible with long rays and intense light
- Can be mitigated by reducing GI strength, and relying more on ambient lighting
- Ultimately would need to continue sampling outside the screen
  - Realtime cubemap around camera?
  - Ray tracing?

Limitation: Occluded light

- Light behind other surfaces cannot contribute
  - Especially visible with long rays
- Can be mitigated by reducing GI strength, and relying more on ambient lighting
- No easy fix
  - Deep G-Buffers? (Would also need back-faces)
  - Stochastic Depth?
- Also: lit surfaces at grazing angles can be only 1-2 pixel wide on screen
  - Would require something like multi-view rendering

Performance (indirect diffuse, RTX 2080)

- Lots of samples + large radius can be expensive
  - Color (B10G11R11)
  - Normal (R8G8B8A8)
  - Depth (R32)
- Can use lower resolution MIP level for farther away samples to reduce bandwidth

Configuration	Sampling	Denoising	Total
a) 8 samples, radius 1, const. steps, full res.	0.9 ms	0.33 ms	1.23 ms
b) 8 samples, radius 4, const. steps, full res.	1.7 ms	0.33 ms	2.03 ms
c) 16 samples, radius 4, const. steps, full res.	2.3 ms	0.33 ms	2.63 ms
d) 16 samples, radius 4, exp. steps, full res.	2.6 ms	0.33 ms	2.93 ms
e) 32 samples, radius 4, exp. steps, full res.	4.0 ms	0.33 ms	4.33 ms
f) 16 samples, radius 4, exp. steps, half res.	0.97 ms	0.1 ms	1.07 ms

Specular light

- Visibility bitmasks can also be used to model GGX specular lobe!
- We reuse diffuse samples so the overhead is small
  - More ALU to handle GGX, but same bandwidth
  - Good enough for rough specular, but sharp reflections would require more samples

Conclusion

- We presented a screen space technique that is more accurate than previous methods for thin surfaces
  - Ambient Occlusion
  - Directional occlusion of ambient light
    - This makes a huge difference over simply applying AO on top of normal ambient
    - Comparable to bent normal but can handle thin surfaces
  - Indirect diffuse lighting
- Performance is comparable to previous methods for AO and ambient
  - Indirect lighting is more costly, but still cheap enough for a lot of games
- Rough specular is possible with little added cost over indirect light


Thank You!
CORIN
Centre de développement
et de recherche en
intelligence numérique