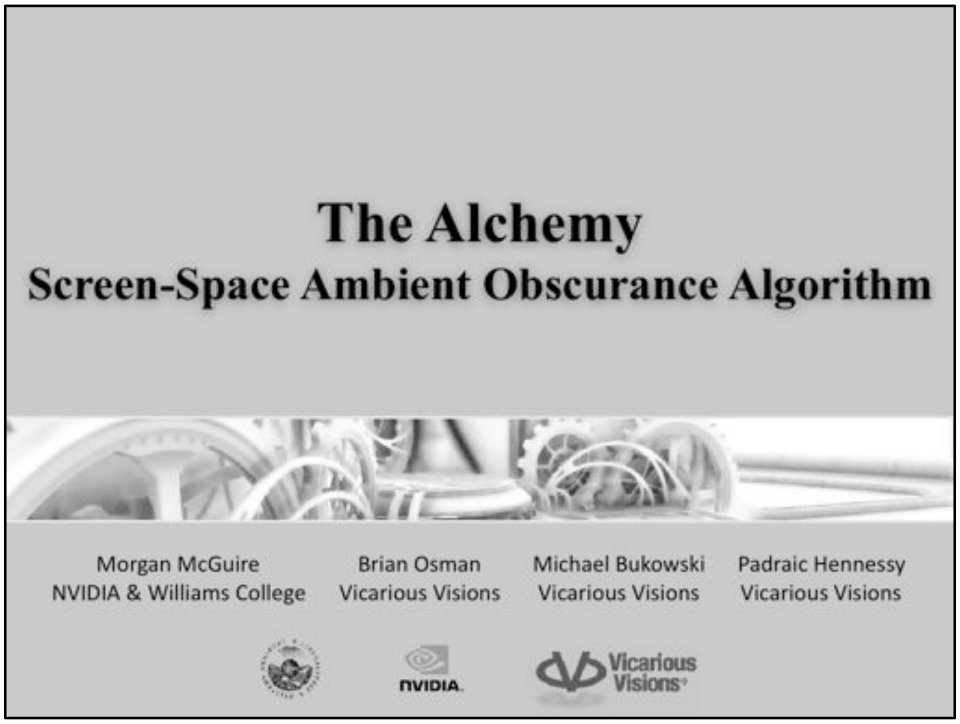


The Alchemy
Screen-Space Ambient Obscurance Algorithm

Morgan McGuire, NVIDIA & Williams College
Brian Osman, Vicarious Visions
Michael Bukowski, Vicarious Visions
Padraic Hennessy, Vicarious Visions

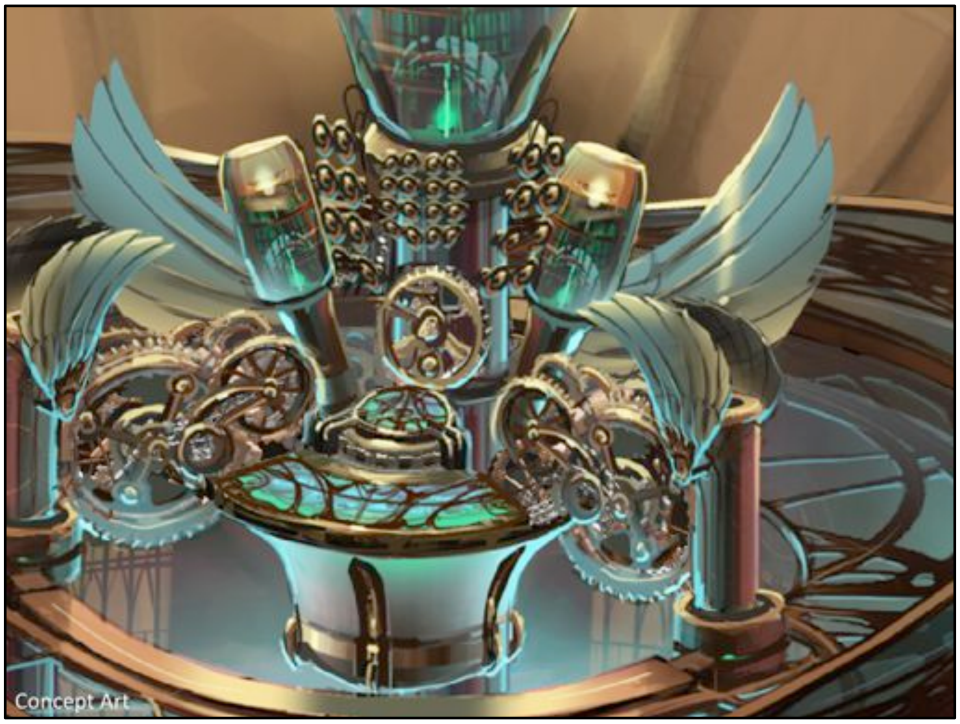
This talk describes some work that we performed at the Vicarious Visions game studio to increase the quality of the real-time lighting in the Alchemy game engine, which we use in most of the games produced by the studio.

When creating the visuals for a game, the concept artists first hand-paint images showing the environments and characters. The modeling team then creates early models and the rendering team designs shading algorithms to bring those models close to the original concept art.

We were at this stage for one game when the concept artists delivered these paintings of a steam-punk ampitheater:

Concept Art

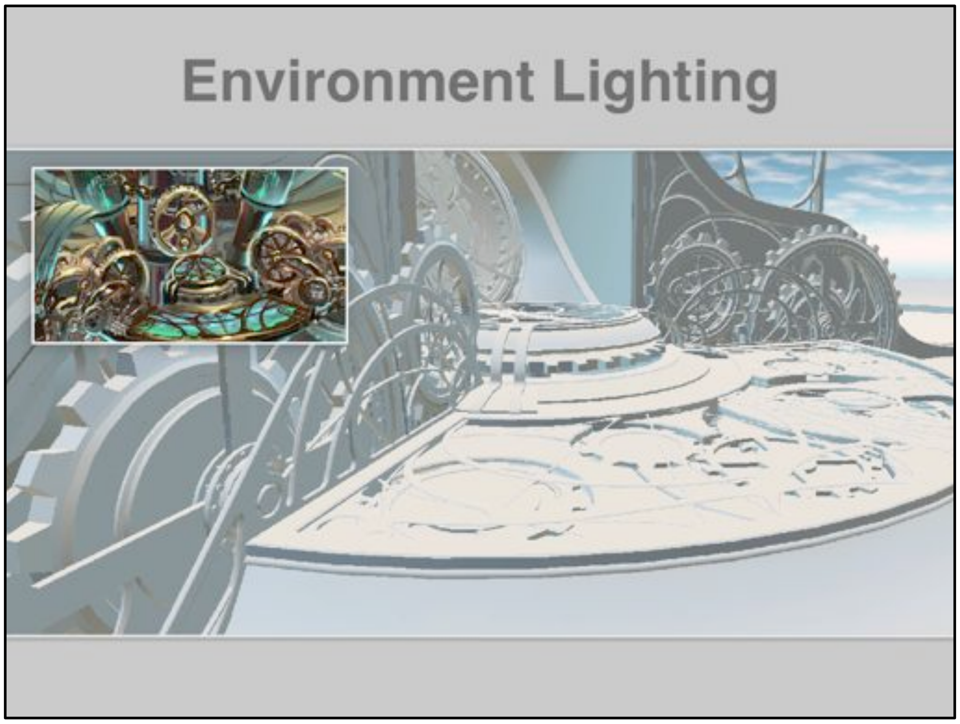
This whole thing is the size of a football field. Let me zoom in so that you can see the “stage” in the center of the ampitheatre



It is constructed from a dense collection of gears and thin parts.   The whole assembly rises up during the game, and is then elevated over a city. Here's the view from behind the performers, standing on the stage

Concept Art

So, given this concept art, the modeling artists built the stage for us, and we rendered it with the full lighting model but no textures…



Environment Lighting

The geometry is beautiful…it is a good match for the concept art. [CLICK]

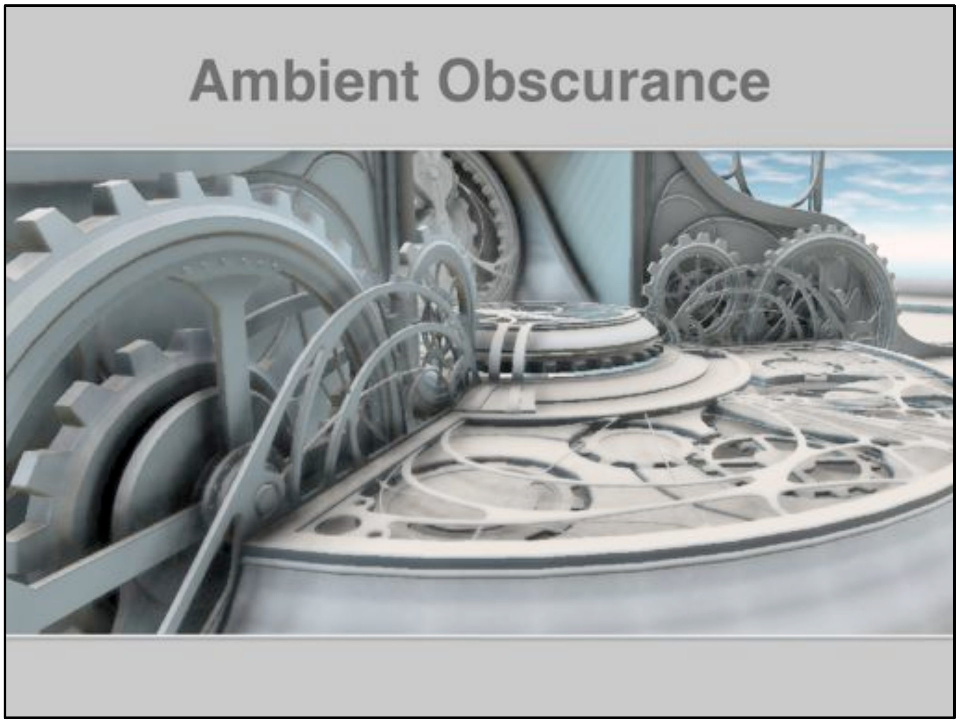
But the visuals lack contrast that brings out all of the fine detail in the painting.

The problem is that our lighting model doesn't capture all of the local shadowing effects from the gears.

It isn't possible to compute true global illumination in real-time on a platform like Xbox 360 or even a GeForce 590 at the scale of those little gear teeth.

We could precompute light maps for a static scene, but for a scene with dynamic elements (like all of those gears!) we need a real-time illumination solution.

Our solution was to compute the local shadowing due to ambient obscurance in real-time, in screen space. This is a popular approach in current games.
Here's this scene with ambient obscurance and environment lighting…

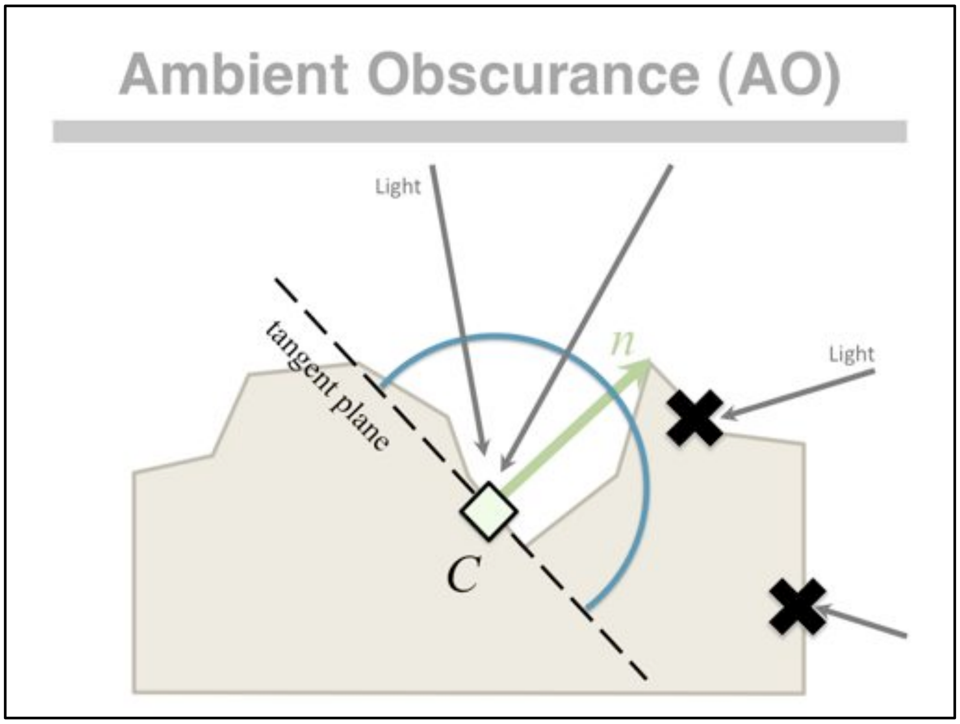


Ambient Obscurance

The fine details now pop, and we can see how the gears are overlapped. Note that there's occlusion at many scales in this scene because the sizes of and distances between gears span a range from a few centimeters to many meters.

Our algorithm can compute the ambient obscurance term at each pixel at HD resolution in about 5 ms across a variety of hardware, from today's GeForce 580 down to the 5-year old Xbox 360 and PS3. We scale by varying quality parameters that I'll describe with the algorithm. You could of course fix quality and scale performance.

I'll now describe the ambient obscurance problem, a line of related screen space techniques, and our new algorithm.



Ambient Obscurance (AO)

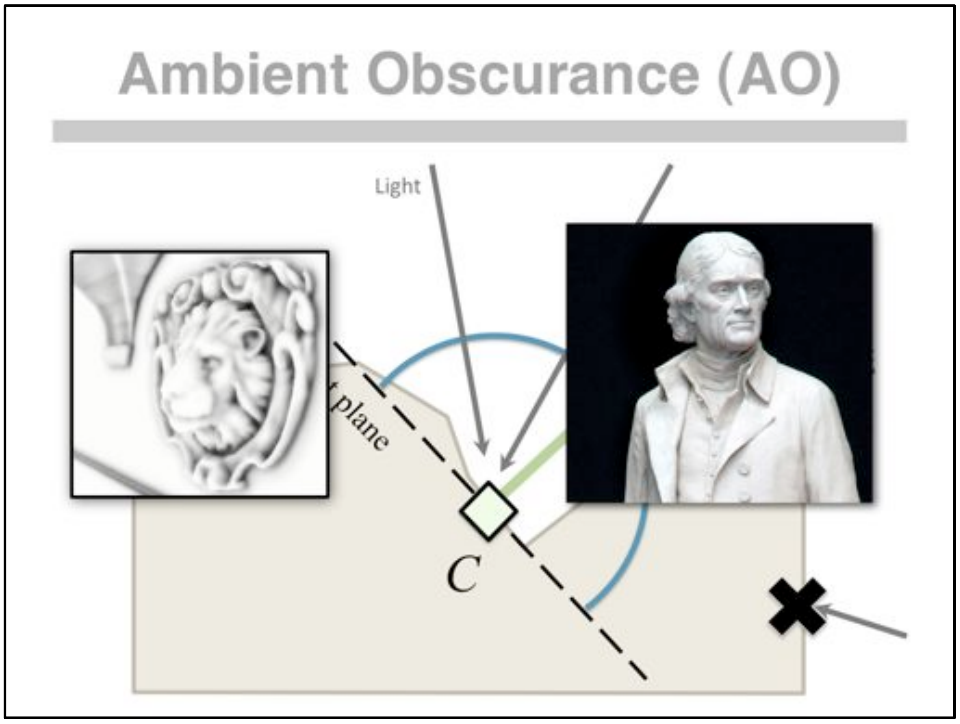
Here's a 2D schematic of an object with rough geometry. Say that I'm estimating the contribution of distant indirect light at point C inside this local valley.

C has a normal and tangent plane.

All of the light that C can reflect to the viewer must arrive in the hemisphere above this tangent plane.

Some light headed towards C actually arrives…
And some is SHADOWED or OBSCURED by the nearby surface.

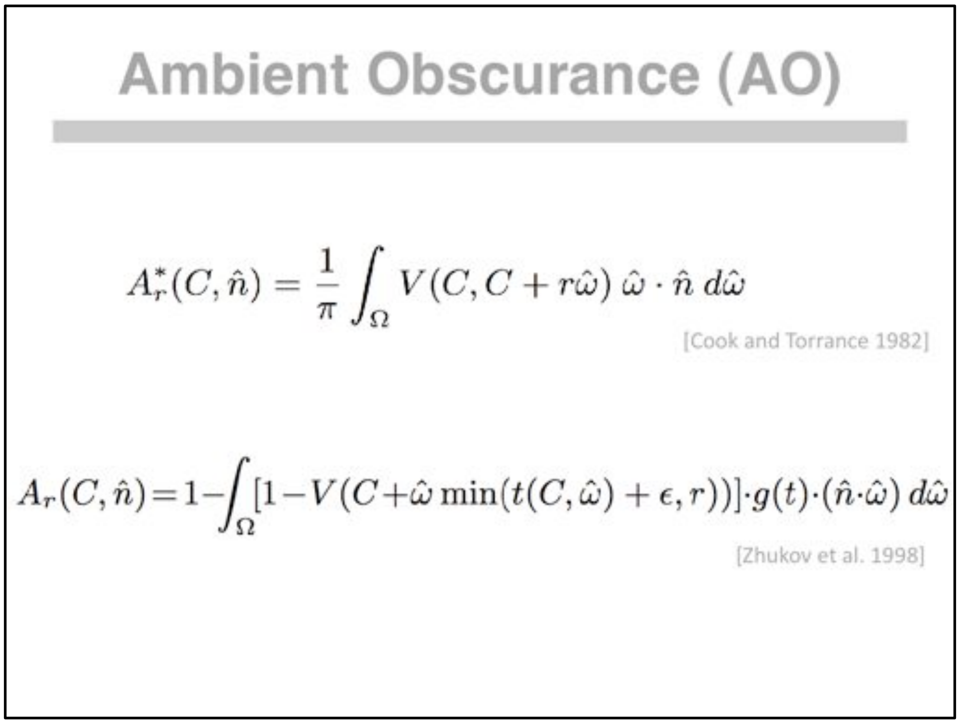
We want to know how much indirect light is obscured…specifically, we want to know the projected-area weighted solid angle that is obscured.



Ambient Obscurance (AO)

The resulting factor looks like this…

Which is essentially what you'd see if this was an all-white surface on a cloudy day. This image was rendered by our algorithm. To show you that this is physically plausible term, here's a photograph of an actual white statue on a cloudy day.



Ambient Obscurance (AO)

$$A_r^*(C,\hat{n}) = \frac{1}{\pi}\int_{\Omega} V(C, C + r\hat{\omega})\,\hat{\omega}\cdot\hat{n}\,d\hat{\omega}$$

[Cook and Torrance 1982]

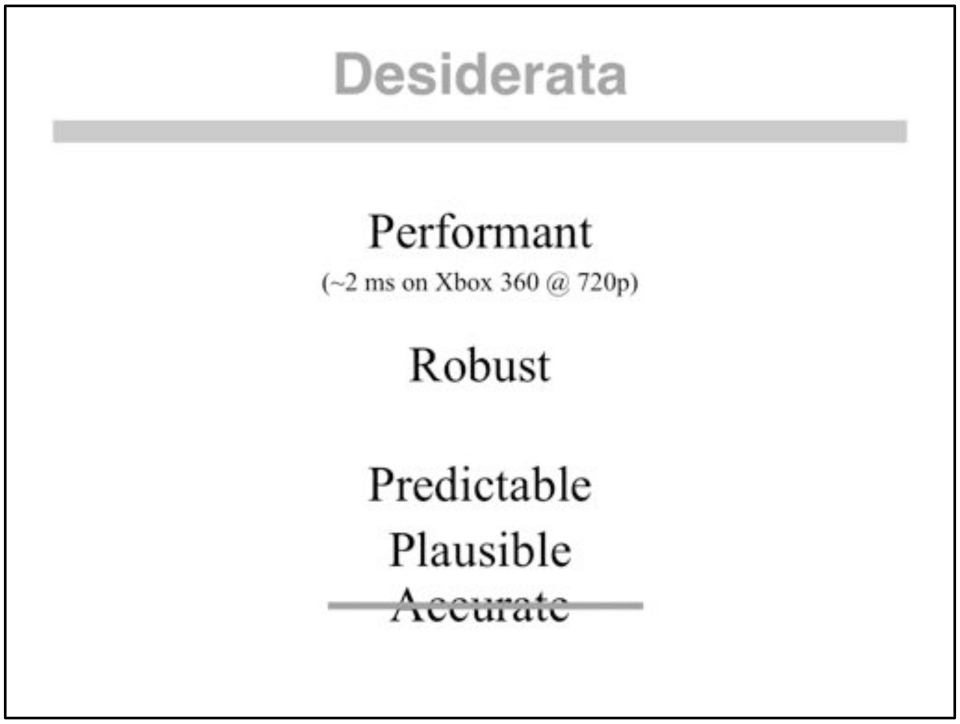
$$A_r(C,\hat{n}) = 1 - \int_{\Omega} [1 - V(C + \hat{\omega}\min(t(C,\hat{\omega}) + \epsilon, r))] \cdot g(t) \cdot (\hat{n}\cdot\hat{\omega})\,d\hat{\omega}$$

[Zhukov et al. 1998]

Ambient OCCLUSION is the integral of local occlusion over the hemisphere, weighted by projected visibility. One typically computes this "A" function, sometimes called accessibility, which is the value between 0 and 1 that you should then scale indirect illumination by; i.e., it is a kind of "shadow value" for diffuse global illumination.

Zhukov et al extended this to its modern form of AMBIENT OBSCURANCE, where the impact of the term falls off with distance so as to smoothly blend high-frequency occlusion discovered by the ambient obscurance algorithm with the occlusion discovered by a lower-frequency global illumination algorithm.

You can see McGuire 2010 for a derivation of these terms and the conditions under which they can be employed accurately within a complete solution to rendering equation.



Desiderata

Performant

(~2 ms on Xbox 360 @ 720p)

Robust

Predictable

Plausible

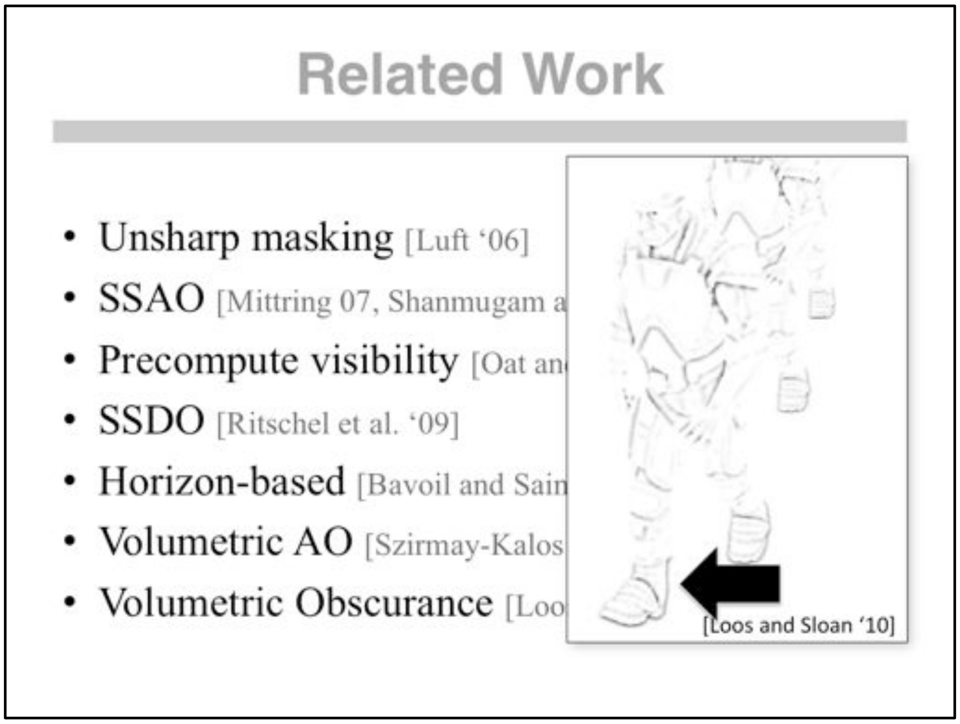
~~Accurate~~

These are the properaties that you might generally desire in a lighting algorithm.

Graphics algorithm design is about finding a workable solution to an overconstrained problem. We can't have all of these for the level of performance and target architectures on which our games ship, so we have to decide what to drop.

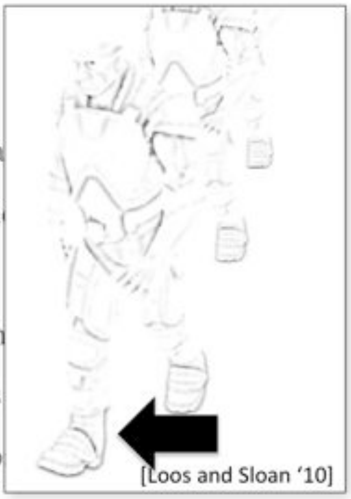
For our designers, the advantage of physically-motivated rendering is that it is robust and predictable, like the real world. Those properties allow them to employ effects with confidence and to adjust lighting intuitively.
The physical accuracy is less important. [click] They are trying to match a vision set forth by the concept art, and they'll tune whatever parameters we can provide to match that vision—which might not be strictly physical rendering anyway.

So we remove the accuracy constrating. We're looking for the high-level phenomena of real lighting, but aren't concerned with producing photorealism.



Related Work

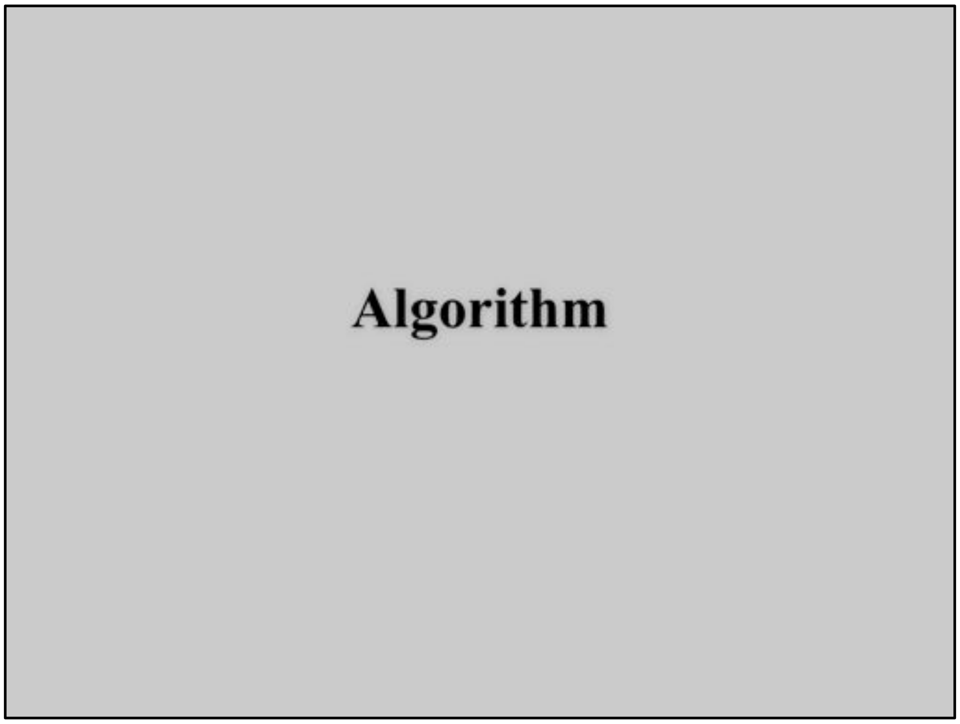
- Unsharp masking [Luft '06]
- SSAO [Mittring 07, Shanmugam a
- Precompute visibility [Oat an
- SSDO [Ritschel et al. '09]
- Horizon-based [Bavoil and Sain
- Volumetric AO [Szirmay-Kalos
- Volumetric Obscurance [Loo



Each of these extends the previous one with some new element to improve performance or image quality, and carries the other ideas forward.
Alchemy picks up after HBAO and Volumetric AO.

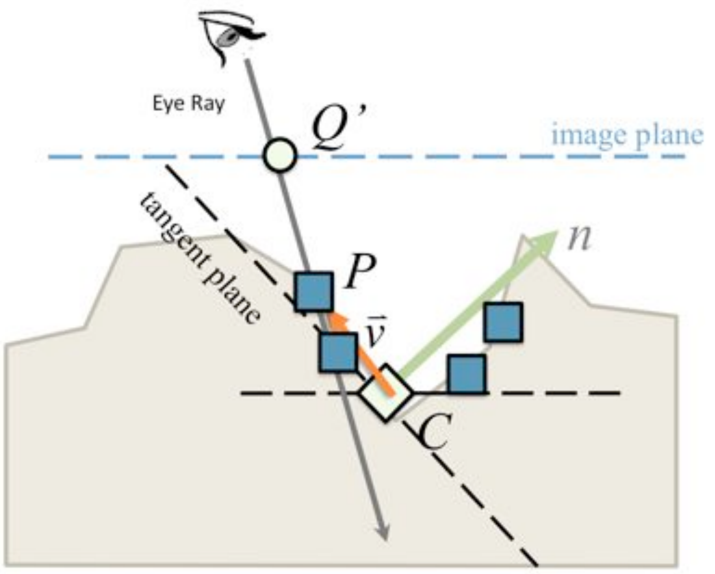
We initially implemented low-resolution Volumetric Obscurance with upsampling in our engine.
We wanted slightly better performance so that we could compute AO at full resolution on PC rather than interpolating.

More significantly, we wanted to extend the algorithm to produce more plausible contact shadows. Here's a result by Loos and Sloan showing the AO around a soldier walking on flat ground. The contact shadow near the boot should be a pool on the ground. Instead, it appears as an outline.



Algorithm

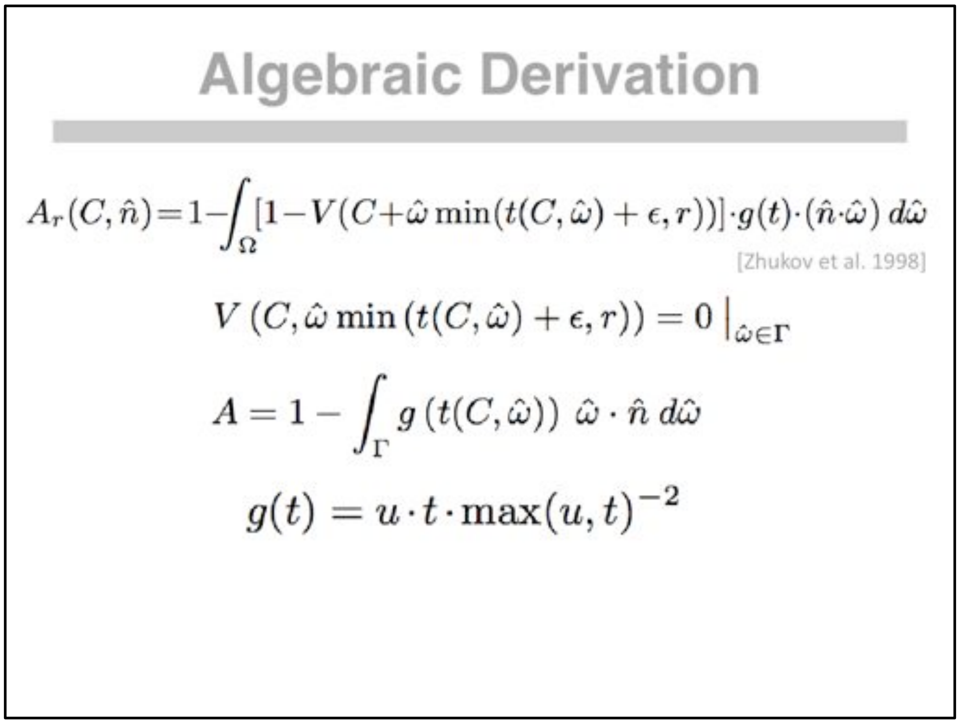
Geometric Derivation



Choose a point Q in image space near the projection of C.

Read the depth buffer there, which gives the location of some point in the world, P, that may obscure C.

The vector v from C to P tells us the projected area factor and distance attenuation. We sample many points in this manner and accumulte an estimate of the obscurance of the hemisphere.



Algebraic Derivation

$$A_r(C,\hat{n}) = 1 - \int_{\Omega} [1 - V(C + \hat{\omega}\min(t(C,\hat{\omega}) + \epsilon, r))] \cdot g(t) \cdot (\hat{n} \cdot \hat{\omega})\, d\hat{\omega}$$

[Zhukov et al. 1998]

$$V\left(C, \hat{\omega}\min\left(t(C,\hat{\omega}) + \epsilon, r\right)\right) = 0\,\Big|_{\hat{\omega}\in\Gamma}$$

$$A = 1 - \int_{\Gamma} g\left(t(C,\hat{\omega})\right)\ \hat{\omega} \cdot \hat{n}\, d\hat{\omega}$$

$$g(t) = u \cdot t \cdot \max(u,t)^{-2}$$

(Wave hands and move through this fast; the details are in the paper)

Start with Zhukov’s expression.

Let Gamma be the solid angle subtended by occluders; that’s the area where local visibility is zero.

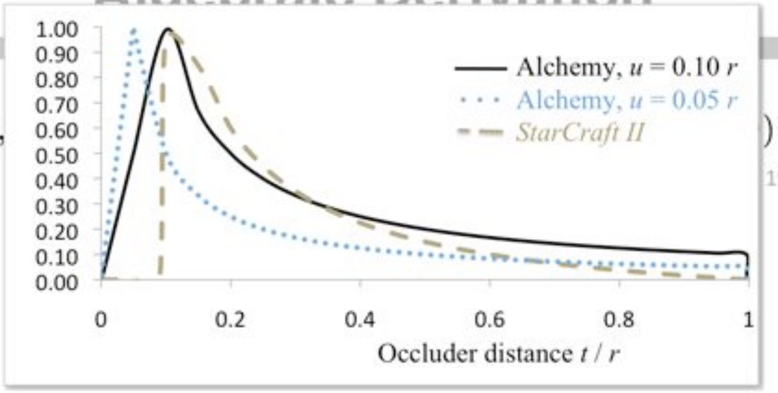
Restricting the domain of integration to Gamma simplifies this expression.

Zhukov’s “g” function is the artistically-tuned falloff. You can chose anything that produces the level of smoothness you desire.
We choose this specific function, which matches the properties of the fallof that StarCraft II’s AO function uses, but has a mathematical form that is particularly efficient to evaluate.

Algebraic Derivation

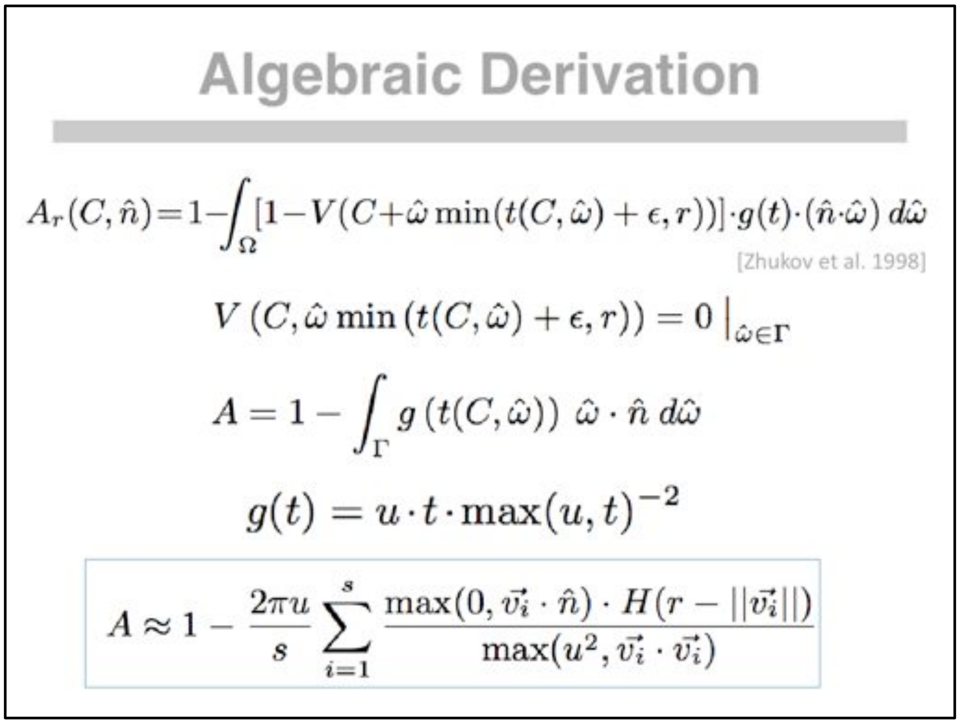
$A_r(C,$ … $)\,d\hat{\omega}$

1998]



$$g(t) = u \cdot t \cdot \max(u, t)^{-2}$$

The dashed line is StartCraft II, the solid black and dotted blue are our function for two values of the parameter u. You can see that they have the same overall shape: discounting occlusion very near a surface (i.e., eliminating self-occlusion), rapidly rising to full weight, and then falling off with distance.



Algebraic Derivation

$$A_r(C,\hat{n}) = 1 - \int_{\Omega} [1 - V(C + \hat{\omega}\min(t(C,\hat{\omega}) + \epsilon, r))] \cdot g(t) \cdot (\hat{n} \cdot \hat{\omega})\, d\hat{\omega}$$

[Zhukov et al. 1998]

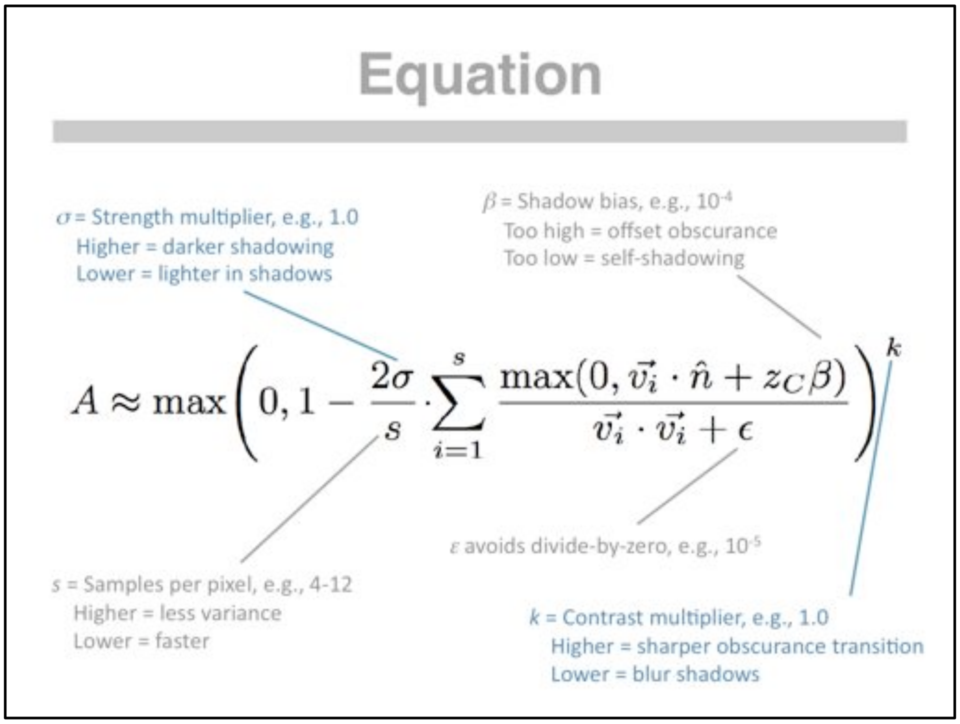
$$V\left(C, \hat{\omega}\min\left(t(C,\hat{\omega}) + \epsilon, r\right)\right) = 0\, \Big|_{\hat{\omega}\in\Gamma}$$

$$A = 1 - \int_{\Gamma} g\left(t(C,\hat{\omega})\right)\ \hat{\omega}\cdot\hat{n}\, d\hat{\omega}$$

$$g(t) = u \cdot t \cdot \max(u,t)^{-2}$$

$$\boxed{A \approx 1 - \frac{2\pi u}{s}\sum_{i=1}^{s}\frac{\max(0, \vec{v_i}\cdot\hat{n})\cdot H(r - ||\vec{v_i}||)}{\max(u^2, \vec{v_i}\cdot\vec{v_i})}}$$

Substituting g into the previous equation and numerically estimating the integral gives this expression. The exact form that we evaluate changes slightly when we discover occluders by sampling in screen space instead of world space…



Equation

$$A \approx \max\left(0, 1 - \frac{2\sigma}{s} \cdot \sum_{i=1}^{s} \frac{\max(0, \vec{v_i} \cdot \hat{n} + z_C \beta)}{\vec{v_i} \cdot \vec{v_i} + \epsilon}\right)^k$$

σ = Strength multiplier, e.g., 1.0
Higher = darker shadowing
Lower = lighter in shadows

β = Shadow bias, e.g., 10^{-4}
Too high = offset obscurance
Too low = self-shadowing

ε avoids divide-by-zero, e.g., 10^{-5}

s = Samples per pixel, e.g., 4-12
Higher = less variance
Lower = faster

k = Contrast multiplier, e.g., 1.0
Higher = sharper obscurance transition
Lower = blur shadows

The important properties of this expression are that:
- it is computationally efficient to evaluate in screen space
- it is intuitive to tune
- produces predictable, robust AO phenomena

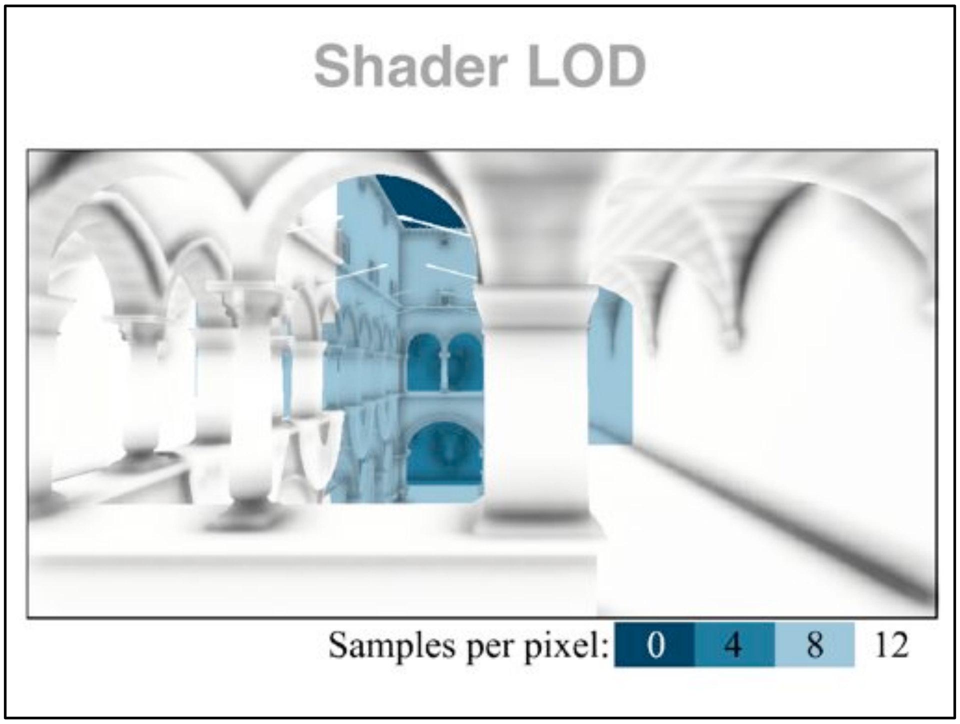
Z_C is the camera-space z value for the point being shaded.
N is the camera-space normal at the point being shaded.
V_i is the camera-space vector from the current point to another point discovered by sampling nearby in the depth buffer.

You just sample some number of points, say 8, in screen space and evaluate this expression over them, then multiple indirect illumination by that value.

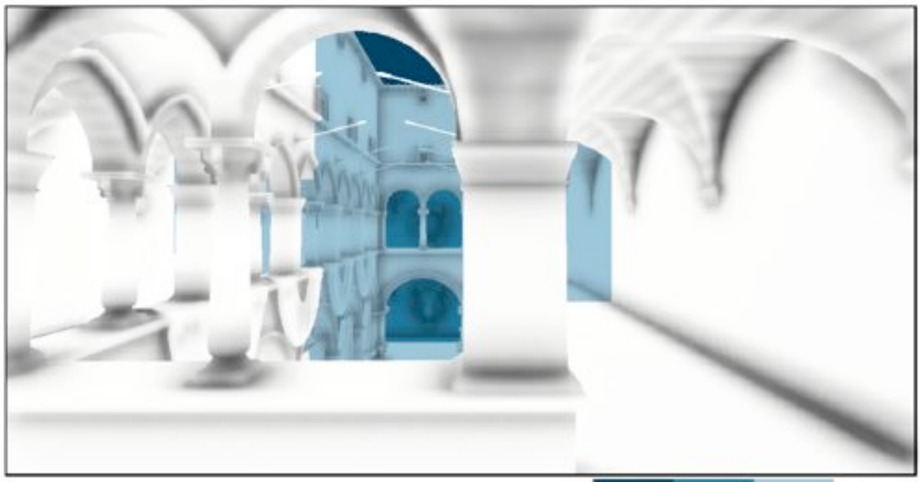
A programmer tunes the parameters in gray for the engine’s numerical precision and target platform.
Artists tune the blue parameters for aesthetics.

V will be incorrect near the edge of the screen, so we render a bigger image than the actual frame and then crop down, just like with bloom.

GET HERE AFTER ABOUT 7 minutes



Shader LOD

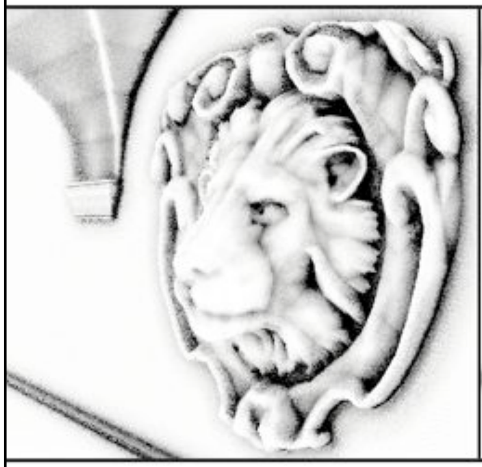


Samples per pixel: 0 4 8 12

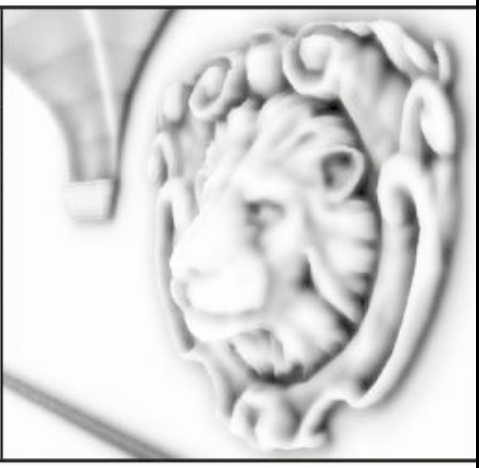
Talk about banding artifacts. Self-shadowing because interpolated normals don't match geometry.

You can address this by using geometric normals instead of interpolated normals.

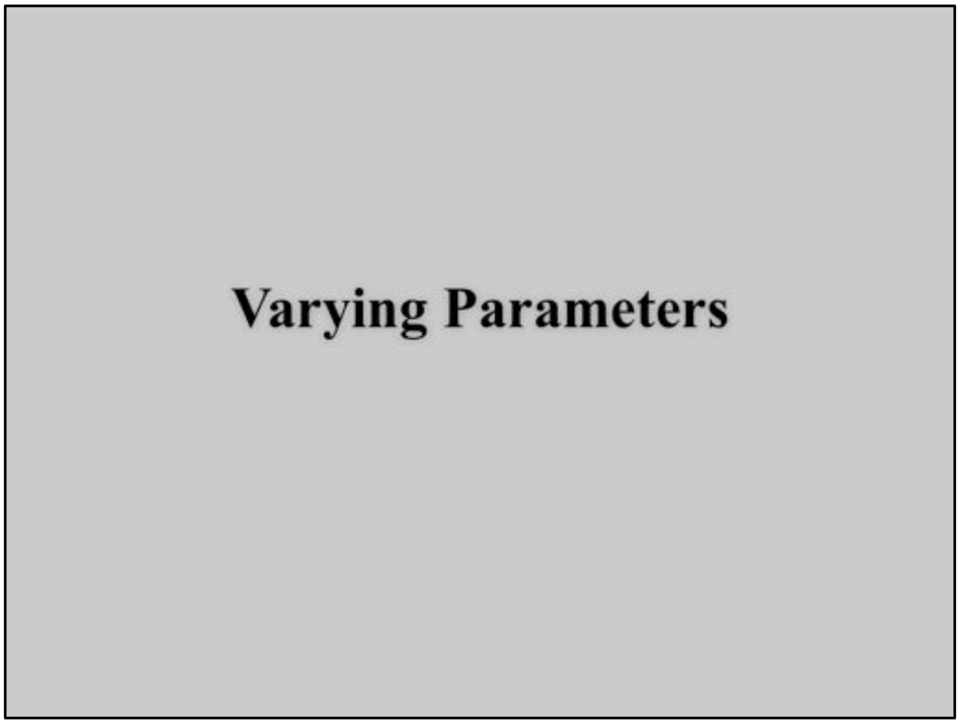
Cross-Bilateral Reconstruction



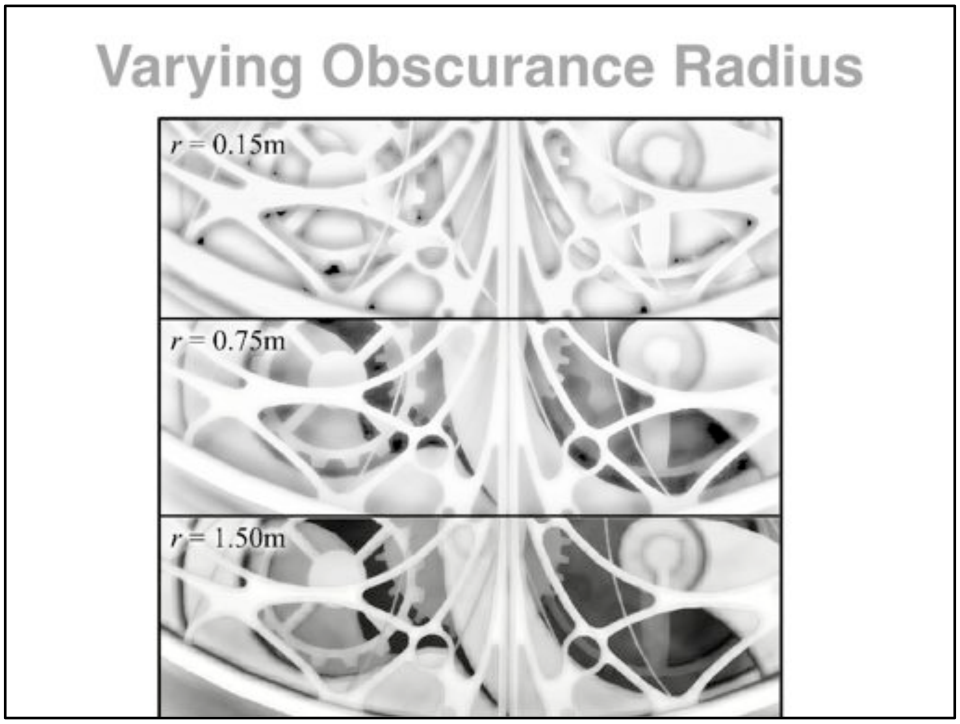
Raw obscurance estimate,
tuned to over-darken for visualization



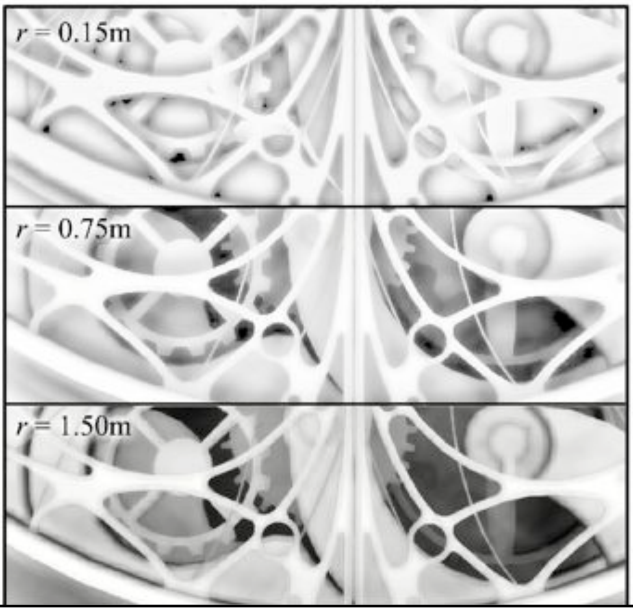
Cross-bilateral reconstruction with
“separated” depth edge-preserving
Gaussian kernel



Varying Parameters

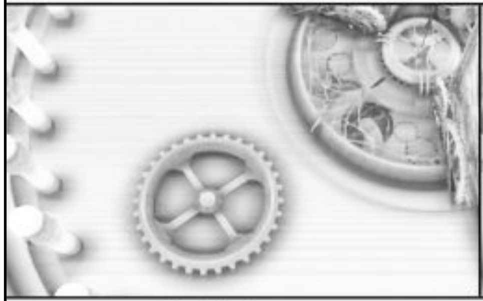


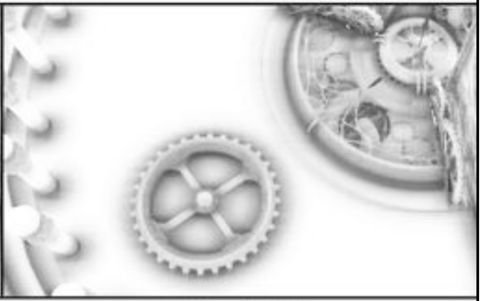
Varying Obscurance Radius



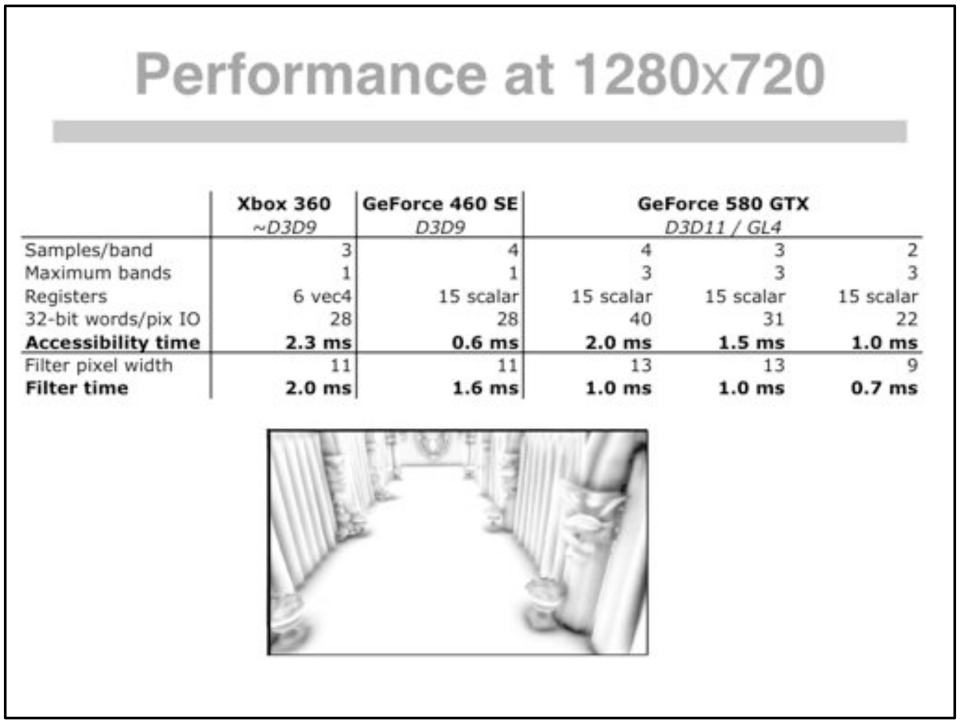
Increasing radius caputres details at more scales but impacts performance

Bias


Bias $\beta = 0$ m


Bias $\beta = 10^{-4}$ m

Artiacts due to roundoff errors at
16-bit depth precision



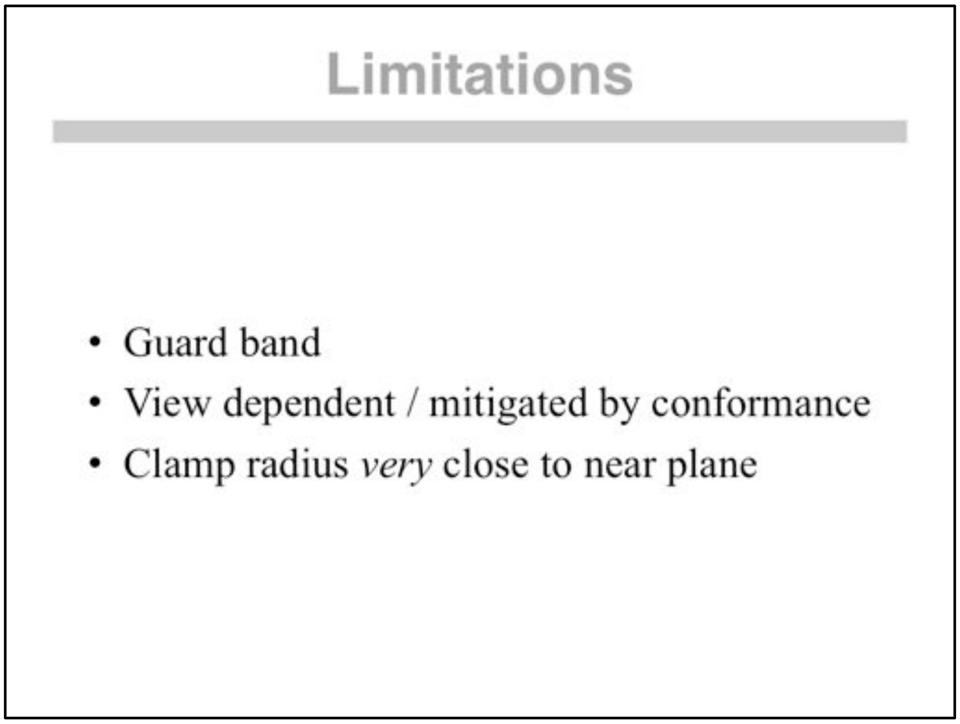
Performance at 1280x720

	Xbox 360	**GeForce 460 SE**	**GeForce 580 GTX**		
	~D3D9	*D3D9*	*D3D11 / GL4*		
Samples/band	3	4	4	3	2
Maximum bands	1	1	3	3	3
Registers	6 vec4	15 scalar	15 scalar	15 scalar	15 scalar
32-bit words/pix IO	28	28	40	31	22
Accessibility time	**2.3 ms**	**0.6 ms**	**2.0 ms**	**1.5 ms**	**1.0 ms**
Filter pixel width	11	11	13	13	9
Filter time	**2.0 ms**	**1.6 ms**	**1.0 ms**	**1.0 ms**	**0.7 ms**

Performance for Alchemy AO in the the Crytek Sponza. We vary parameters across platforms to maintain performance within a factor of two while raising quality.

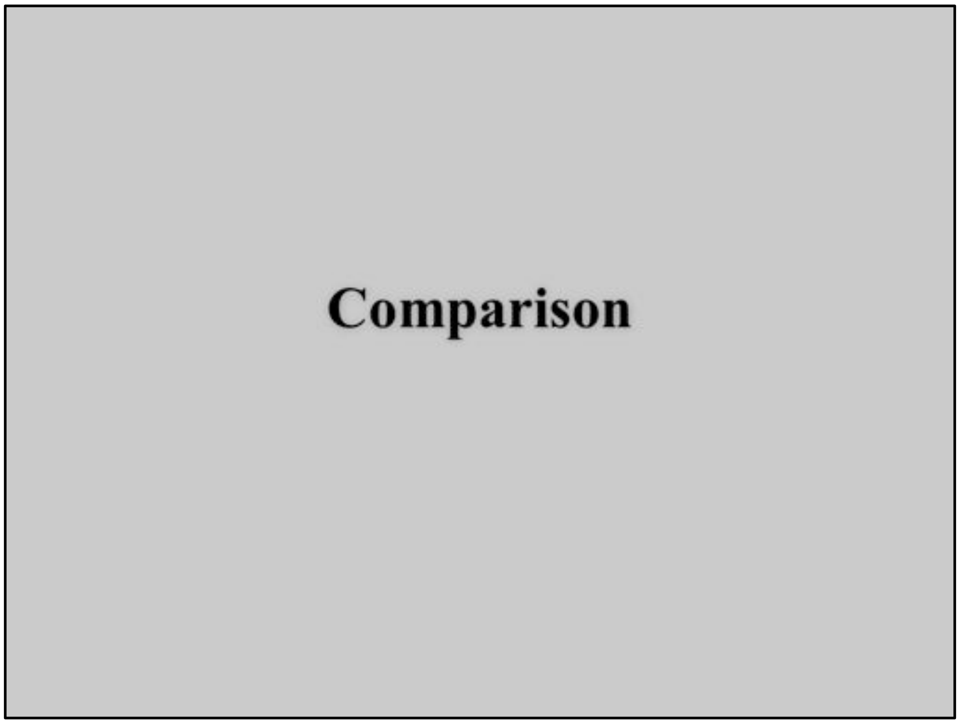
The filter time can be amortized over other effects, like screen space-shadows and glossy reflection, so it has zero additional overhead in some rendering engines.

(D3D9 doesn't branch inside the shader, so that version doesn't have LOD)

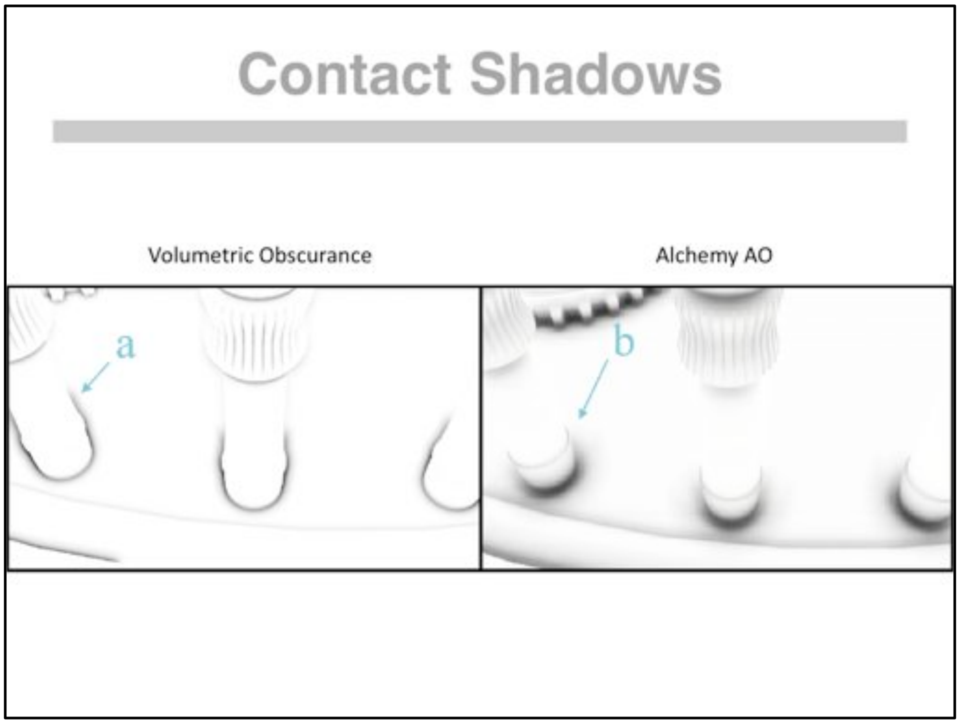


Limitations

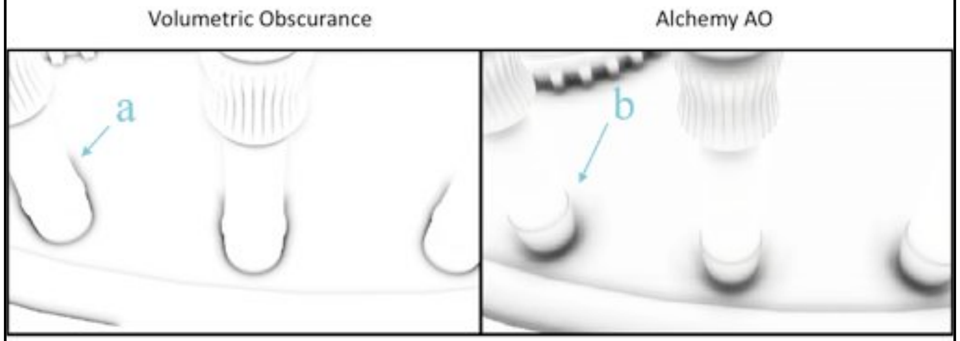
- Guard band
- View dependent / mitigated by conformance
- Clamp radius *very* close to near plane



Comparison



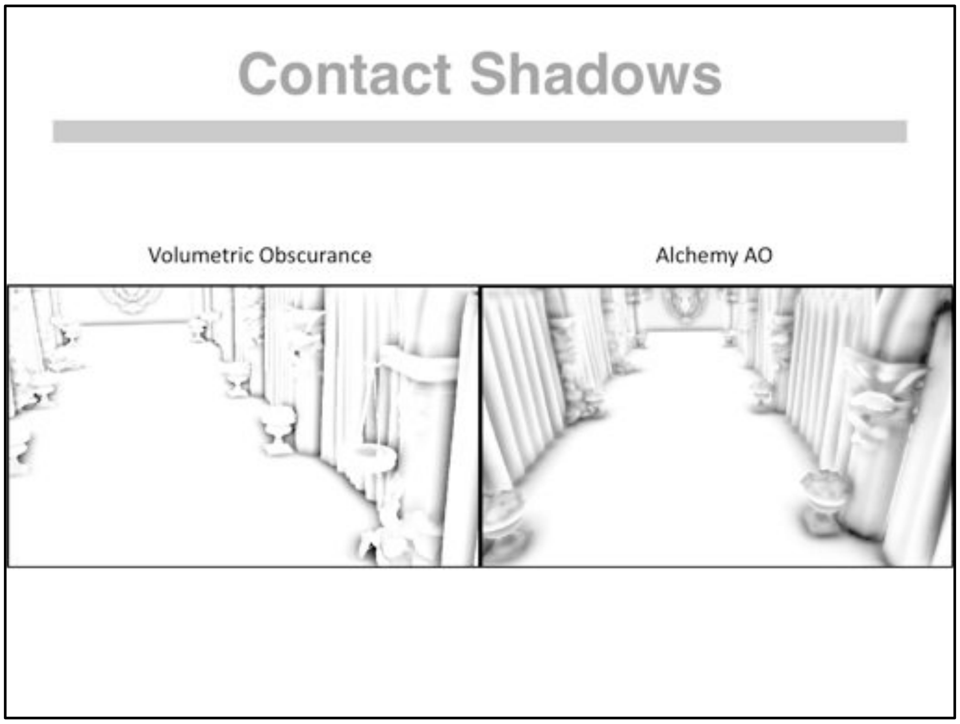
Contact Shadows



Comparison to volumetric obscurance. These render in about the same time.

Note “floating shadows” on the left.

Recall ROBUST and PREDICTABLE goal.

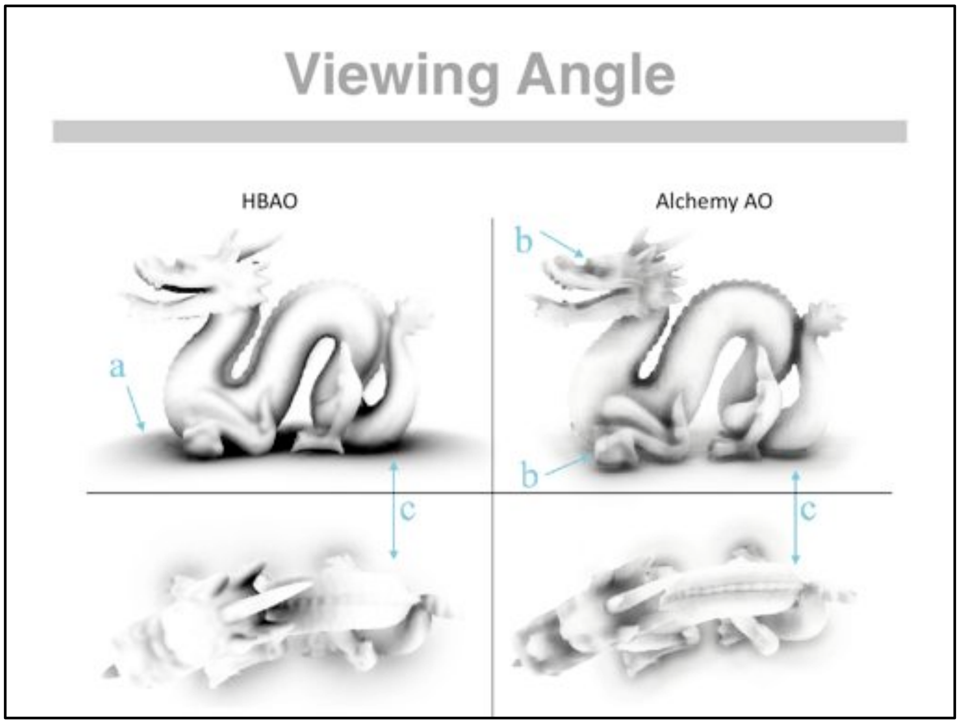


Contact Shadows

Volumetric Obscurance

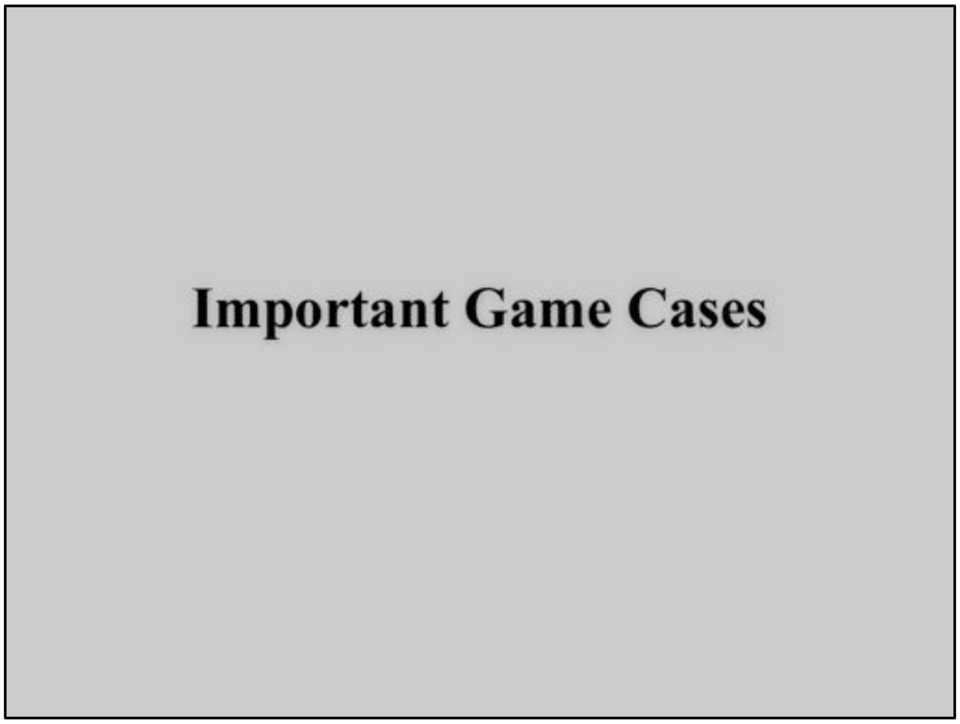
Alchemy AO

Comparison to volumetric obscurance

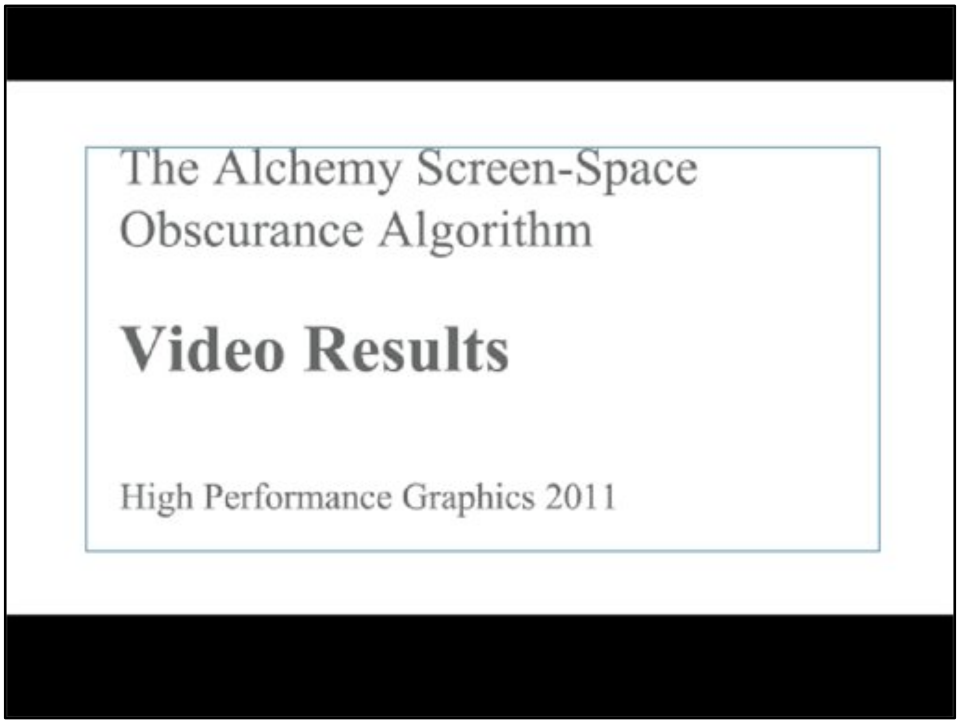


Viewing Angle

B: more fine details from processing every pixel
C: more consistent intensity with angle



Important Game Cases



The Alchemy Screen-Space Obscurance Algorithm

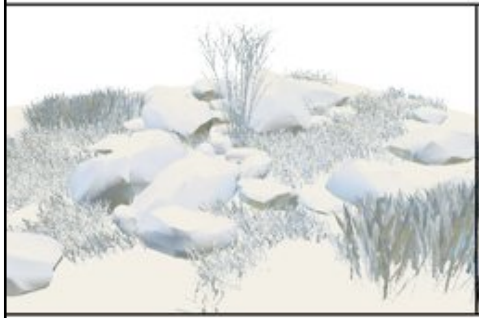
Video Results

High Performance Graphics 2011

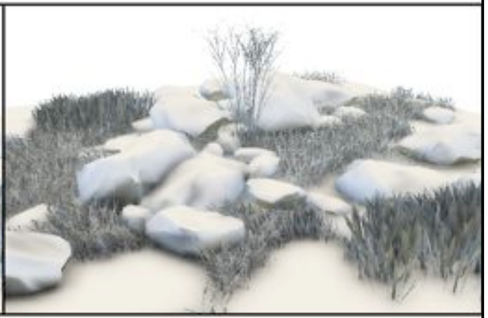
I’m showing you the entire frame, including the guard band, in these images, so that you can see how the AO disappears there. You’d normally crop about 20 pixels off of each side as shown by the blue box.

Fine Foliage Details

Environment Only



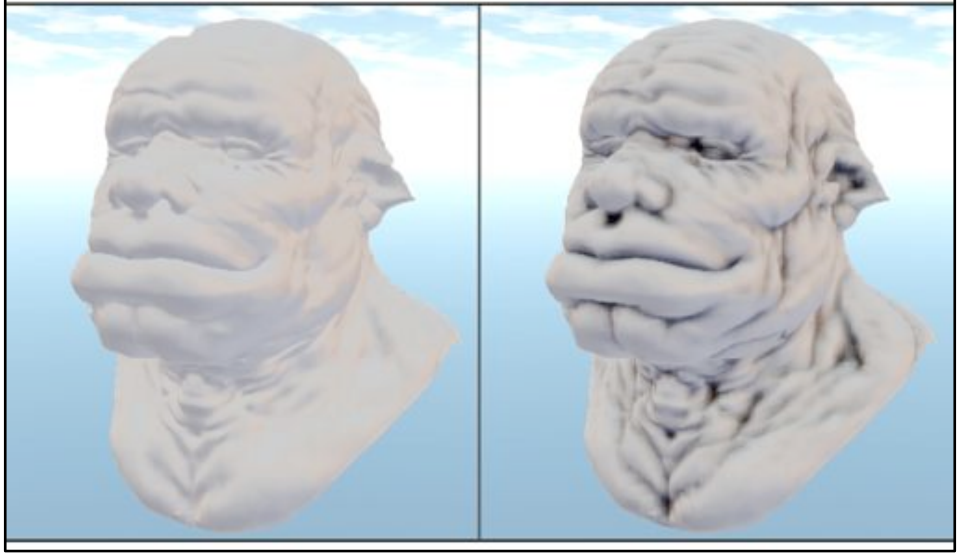
Modulated by Alchemy AO

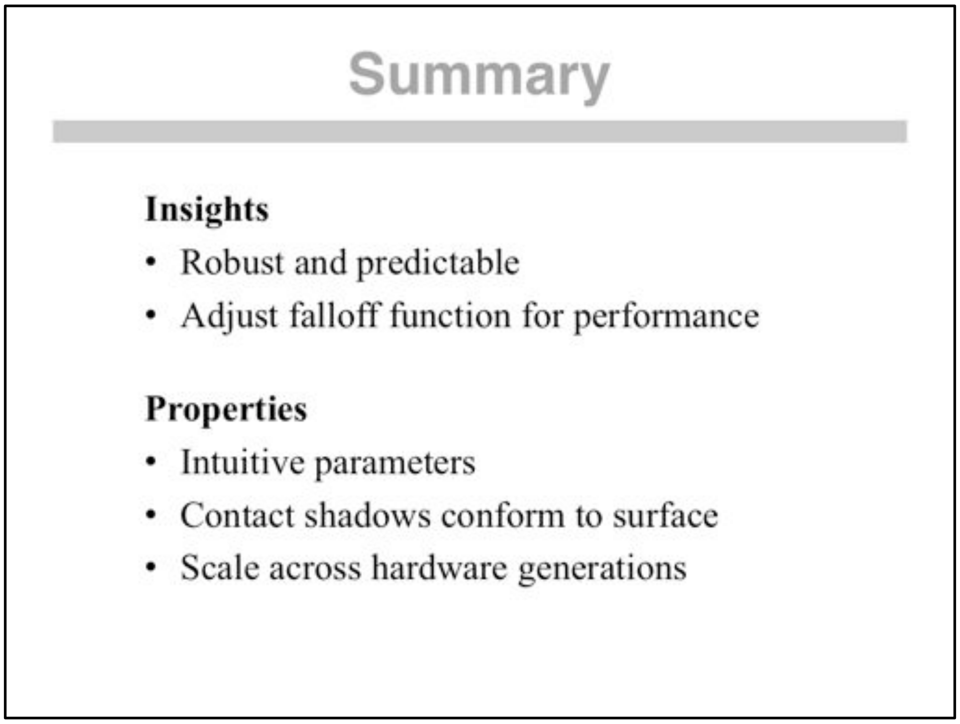


Character

Environment Only

Modulated by Alchemy AO





Summary

Insights

- Robust and predictable
- Adjust falloff function for performance

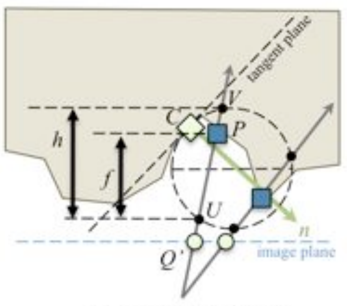
Properties

- Intuitive parameters
- Contact shadows conform to surface
- Scale across hardware generations

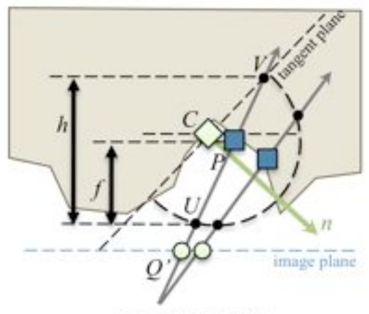
Vicarious Visions

e-mail morgan@cs.williams.edu for GLSL source

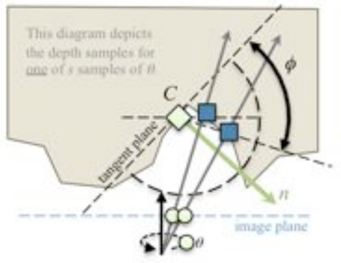
ADDITIONAL MATERIAL



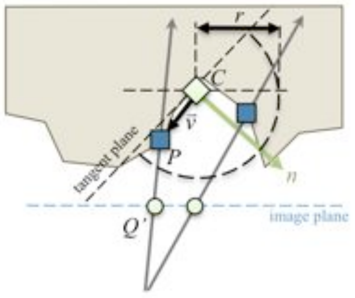
Szirmay-Kalos et al. [09, 10]
(Volumetric AO)



Loos and Sloan [10]
Volumetric Obscurance


This diagram depicts the depth samples for one of s samples of θ.
C
φ
tangent plane
n
image plane
θ

Bavoil and Sainz [08,09]
(Horizon-Based AO)



The New Alchemy Algorithm