5 years is a long time
Particularly for one game
But finally I can talk about the renderer
Think Different

How we tried to take Conviction in a **new direction**

How this motivated two new (different) approaches...

A completely dynamic visibility system
A unique, semi-dynamic Ambient Occlusion approach
Inspiration: Bourne series, 24


In terms of rendering:

Outdoor environments
HQ ambient lighting
Lots of dynamic objects

Radical departure...
No threading

Interiors
High-contrast lighting
High cost to rendering objects
Bolted down

Plus, also carrying a little ‘extra weight’ - code bloat

The great temptation? To start from scratch...
Post Mortem: Cleanup

BSP/Portals

Old hat
Spaghetti
Holding us back

A prime example: BSP

An integral part of Unreal Engine

Showing its age
Messy implementation (we’d hacked on it)
Won’t meet our goals – dynamic and/or localised occlusion (particularly outside)

But we had a solution...

[By the way, this is me back then getting messy with the code. As I said, 5 years is a long time. 😊]
Fairly new to the scene
We had naïve expectations
Just render lots of queries – yay!
Whilst we were busy rewriting the renderer, back on the farm...

Removing the BSP:

Caused **major disruption for LDs**. They were used to blocking out maps with BSP brushes, not making meshes in 3ds Max!

**Robbed engine** of an acceleration structure

Should have **distilled** what we had!

The BSP geometry could have been converted to static mesh chunks and also used for collision, occlusion.
Avoid disruptive changes (rewrites) -> Resistance, confusion, distrust

But... you have to push boundaries

Fail Early, Fail Often. Get good at this and you reduce risk. Try 100 ways to find that one way...

An amazing personal experience
What is visibility?

Cull what you can’t see
Ideally everything
Effectively render more

Before & after
Query problems:

- Latency
- Popping
- Can’t batch
- Grouping? Complicated

Nothing worked fully

<rant>You’ll see these issues or their side effects even in commercial middleware. Either there’s popping (‘latent’ queries) or the tests must be interleaved with regular rendering, which can complicate things or restrict you.</rant>
Had plenty of ideas

One was the “Hierarchical Z Buffer”, Greene et al., Siggraph 93

But:

Ran out of time
Worried about performance
Unproved solutions (HZB in particular)
Thought we could work around every problem

Barriers lifted...
More time + inspiration (Josh & Jeremy at Siggraph ’08 – Froblins talk)
Simple, elegant

Render occluders - doing this anyway?
Build depth MIP chain
Test objects against this

[Besides read-back, all of this happens on the GPU]
These are our occluders

Typically **decoupled** from the visual meshes

Pretty abstract

You’ll see this market scene again shortly in context
This is your z-buffer on steroids

Take **max of 4 texels** when generating next level

**Skybox (white) begins to dominate**
Take the objects’ bounds (blue)
Find the screen bounds (green)
Find the right level of the HZB (orange) – the one that covers the screen bounds with <= N texels

Visibility test: Is object min_z < HZB (overlapping texels)

Advantages: Fixed cost, single batch (one big POINTLIST VB of bounds), extremely fast, in our case no frame delay

Trade off: Small objects are accurate, big objects are inaccurate. This is probabilistic. Big objects (in screen terms) are more likely to be visible anyway
We wait for the results, you could defer

CPU typically has some work to do (light re-association)
GPU can be idle as results are processed by the CPU (try to move most of this work until later in the frame)

We also test: Lights, AO fields, decals, deferred queries for main engine
We also check: Does object need to stream textures?, can it be screen-size faded/culled?, what tiles does it cover? (360)

360: No predicated tiling, no BUFFER_2_FRAMES!

Production proven!
Inspired by CC Shadow Volumes

First pass: From light

Cull hidden
Clamp visible to occluders
Visibility: Extensions

Shadows

Two steps...

Second pass: From camera

Test clamped shafts
Can reuse the HZB from the main render

Used for cascaded shadows
Rough 360 stats:

~0.35ms for all the queries (worst case – not all levels have this many)

Plus

~0.1ms for downsample (512x256 - could possibly be lower)

Also not counting occluder rendering (maybe another ~0.1ms – can be CPU bound at the moment)

versus ~0.05ms per full-screen occlusion query. Think about that for a second!

It’s so fast that we didn’t even bother with a shallow hierarchy pre-pass

Could use this to drive pre-computed visibility too

<rant>I can’t believe some developers suffer 6h+ static PVS pre-processing steps for their levels, even if they’re aiming for 60FPS.</rant>
Visibility: Demo
Not perfect
Humans get in the way

Cull the vast majority of objects
Big object != expensive object

Occluder mesh != collision mesh

Testers will find these problems
But only right at the end!

Visual & occluder need to be in sync.
Visibility: Future

Game services
Optimisations
Accuracy

L...?

We’d like to use these more in the **main engine**...

**Determinism** was an issue – P2P COOP, particularly on PC

Several passes
Rasterisation?

But what’s my main goal?...
Take artists out of the loop as much as possible. There’s no such thing as pretty visibility

Minimal production overhead. One less thing to worry about

No cheating. 100% correctness

Perhaps just have artists tag structural meshes, then we weld them all up, possibly simplify and convert to fixed-size chunks / minimise overdraw.
What is AO? This is AO

**Average shadowing at a point.** How much light can get there
Less flat ambient. Contact shadows
Constant/low-frequency ambient otherwise washes things out
Poor man’s GI. We **decouple** colour and luminance changes as an approximation

We **bake everything** but it’s still **dynamic**
**NOT SSAO...** (although it *is* accumulated in screen space 😊)
No SSAO in 2005
But this gives us control, flexibility
Competitive
Not a crease shader
The first component is static baking for self occlusion (rigid bodies) and ‘structural’ meshes (big meshes that don’t move)

Artists can control the falloff/cutoff, which is important for interiors

This is generated with a mini-renderer
Essentially accumulating shadow maps, with depth peeling to support falloff/cutoff

<rant>Put the time in to ensure that artists have fast tools. So they can iterate quickly. So nothing gets out of sync.
It’s fun to boast about having awesome server farms, but you’re doing your artists a disservice if you think a round trip time of a few hours is a good idea. Same goes for PVS.</rant>
Inspired by AO Fields paper

We use volumes in order to capture concave occlusion (e.g. under a table)

**Average + directional visibility** (like second order SH, but the scale factors cancel)

Generated offline as with self AO
Analytical
Second order zonal harmonics
1D Texture – unit sphere on Z
Attached to primary bones
Used for self occlusion too

Those blue pills are just an artistic representation. They’re really ellipsoids.
We combine (splat) fields in screen-space using **special blending**

**Composited** like deferred lighting

\[
P.rgb = B.a * A.rgb + A.a * B.rgb;
P.a = \text{dot}(A, B); // or A.a * B.a
\]
Smooth
Rendered at ¼ res.
Masking by ID for fields
Edge-aware up-sampling
Dot product in the lighting pass. Uses per-pixel normal

We’ve decoupled again: High-frequency surface normal vs. smoothly-varying AO

For our ‘look’, we scale regular diffuse/spec too and even apply contrast to the AO
Ambient Occlusion: Demo
Ambient Occlusion: Demo
This didn’t make it in...
The reverse: lighting
This is a highly active field of research
RGB occlusion/bounce/light? (see earlier)
Combine with Crytek’s GI?

We should have streamed in higher-res. volumes

I can’t imagine hardware supporting cubic interpolation of volume textures anytime soon, but a developer can dream.
I owe a debt of gratitude to all of these guys

All those artists
Josh and Jeremy for inspiration
Michael: Initial AO work
Everyone else for feedback on this talk
Questions and Answers
Rendering with Conviction

Ubisoft:

www.creatorsofemotions.com

Me: stephen.hill /at/ ubisoft.com
References

[Andersson09] Parallel Graphics in Frostbite – Current & Future, SIGGRAPH 2009 Courses
[Bavoil09] Multi-Layer Dual-Resolution Screen-Space Ambient Occlusion, SIGGRAPH 2009
[Hachisuka05] High-Quality Global Illumination Rendering Using Rasterization, GPU Gems 2, 2005
[Kaplanyan09] Light Propagation Volumes in CryEngine 3, SIGGRAPH 2009
[Kontkanen05] Ambient Occlusion Fields, I3D 2005
[Lloyd04] CC Shadow Volumes, EGSR 2004
[Shanmugam07] Hardware Accelerated Ambient Occlusion Techniques on GPUs, I3D 2007
[Shopf08] March of the Froblins: Simulation and Rendering Massive Crowds of Intelligent and Detailed Creatures on GPU, SIGGRAPH 2008 Courses