SIGGRAPH2010

The People Behind the Pixels

Real-time Diffuse Global Illumination in CryENGINE 3

Anton Kaplanyan
antonk@crytek.de

This chapter of the course is devoted to the real-time solution to a diffuse indirect lighting that is used in the large-scale cross-platform commercial game engine CryENGINE 3™.
My name is Anton Kaplanyan and I'm a researcher in Crytek GmbH.

Crytek GmbH

- 10 years in game development
- ~650 employees in 5 offices across Europe
- Multicultural company with 30+ languages
- Shipped:
  - FarCry on CryENGINE 1 in 2001 (PC only)
  - Crysis and Crysis Warhead on CryENGINE 2 in 2007-8 (PC only)
- Multi-platform consoles-ready CryENGINE 3
- Currently working hard on Crysis 2…
  - Q4 2010

Crytek is an interactive entertainment development company founded in 1997 - formally turned into a company in 1999. The main office is located in Frankfurt am Main.

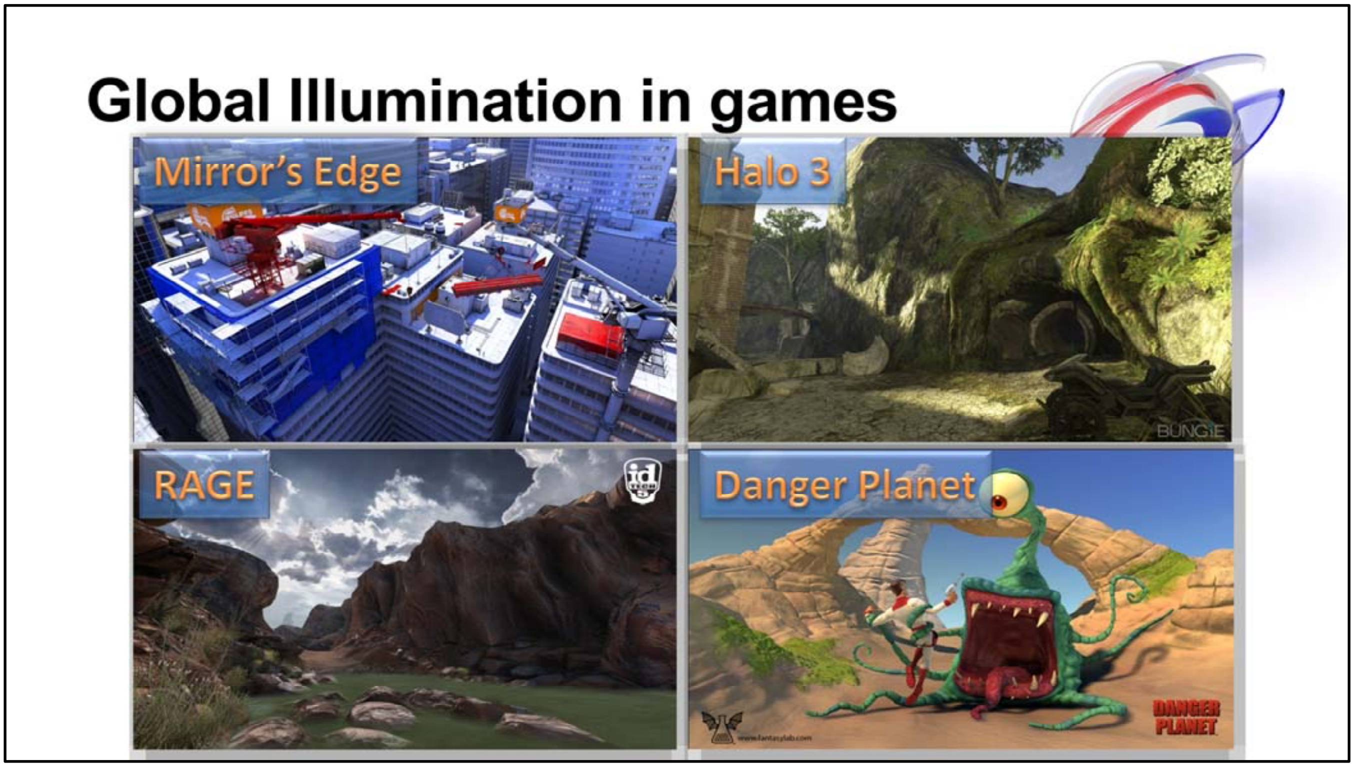
Crytek is dedicated to the creation of high-quality video games for PC and next generation consoles and real-time 3D-Technologies such as the CryENGINE®, one of the world’s most advanced and award winning 3d engines.

In addition to the in-house development of video-games, Crytek also licenses the CryEngine technology to other developers to create games, movies and serious games.

Crytek's development teams and studios are comprised of game professionals from 36+ different nations. Crytek's world-wide official language is English.

We are best known for developing the game Far Cry and the CryEngine that the game uses, and more recently Crysis and CryEngine 2.

We are currently working on the upcoming blockbuster Crysis 2 based on the CryEngine 3.



Global Illumination is an integral part of the most of the state-of-the-art games and engines.
It obviously enriches the picture and adds to the image quality and its perception.
Inherently, it is very important to provide a plausible picture to a gamer to deliver a better gaming experience.

Why <u>dynamic</u> Global Illumination?

- Most games use precomputed indirect lighting (Lightmaps, PRT etc.)
  - Means static scene/lighting
- CryENGINE 3® includes following features:
  - Dynamic deferred lighting
  - Objects' breakability as a part of game-play
- That cancels out all precomputed GI methods
  - We've tried out most of it (including Lightmaps, PRT, RAM etc)
- But we came up with a solution….

The majority of current games and game engines use approaches with precomputed global illumination. That imposes some inherent limitations, like static lighting conditions and/or static objects. Due to this fact the game development process becomes more complicated and consequently lowers the gaming experience.

We always position our engine as an precomputations-free game engine. We provide physically based phenomenon like dynamic time of day changes and objects' breakability to enrich gaming experience and simplify game production process.

That forced us to drop many precomputation-based global illumination approaches, such as Lightmaps and Precomputed Radiance Transfer.

However we came up with a brand-new technique that suits our purposes and has very good characteristics for real-time gaming graphics.

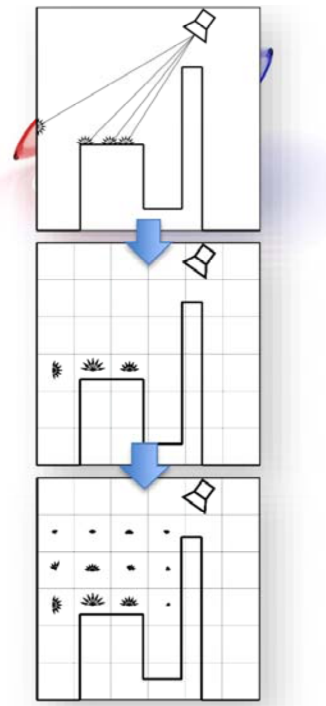
Here is a screenshots from the upcoming blockbuster Crysis 2. Notice the indirect illumination on columns and flowerbeds.

The left bottom part demonstrates the same scene with **constant ambient term** to emphasize the difference.

CASCADED LIGHT PROPAGATION VOLUMES

Core Idea

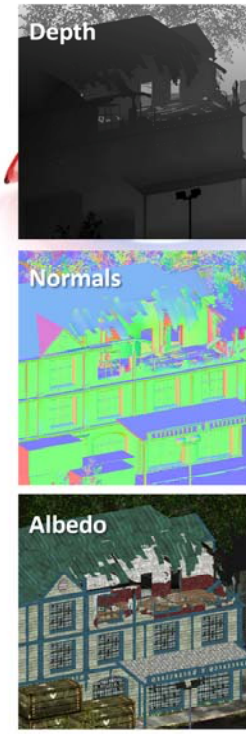
1. Sample lit surfaces
   – Treat them as secondary light sources
2. Cluster samples into a uniform coarse 3D grid
   – Sum up and average radiance in each cell
3. Iteratively propagate radiance to adjacent cells, works only for diffuse
4. Lit the scene with the resulting grid



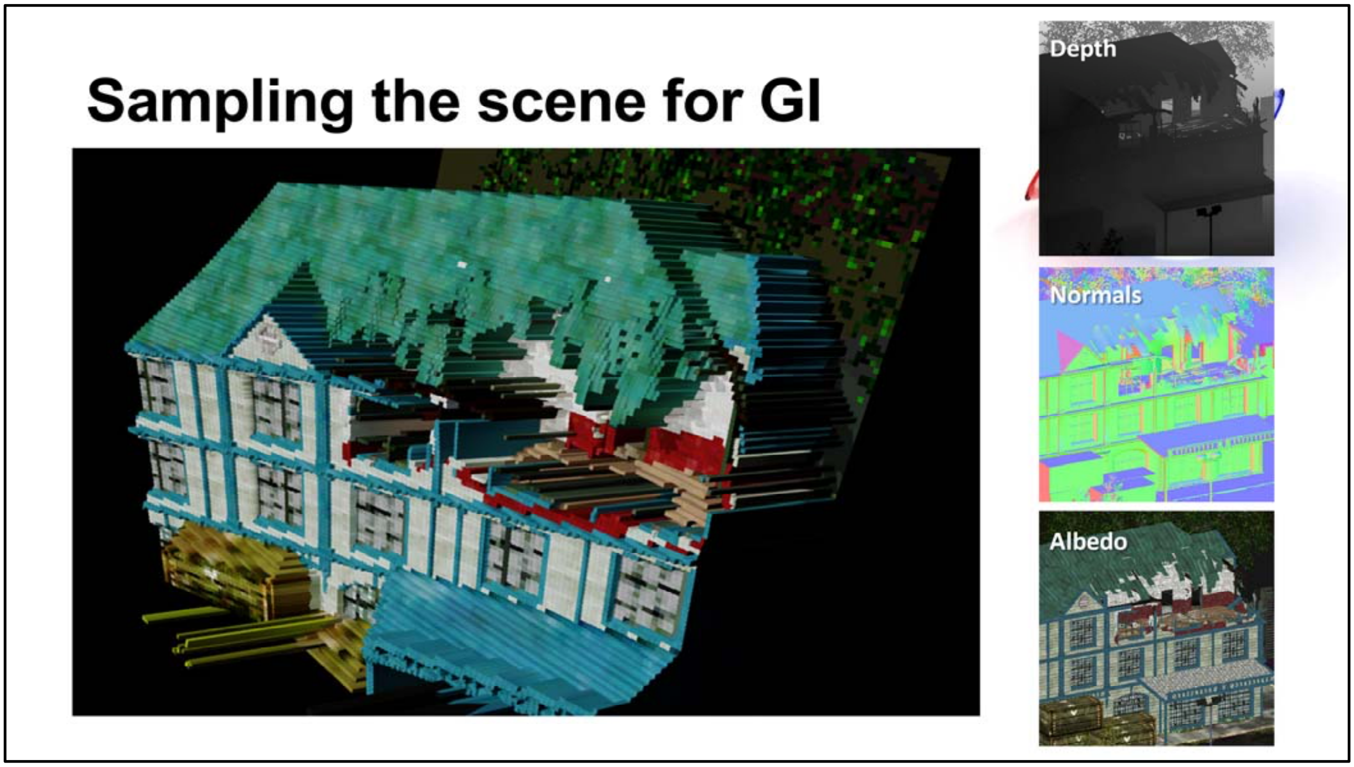
Consider the primary light source emanating **light rays**.
Assuming the whole scene consists of **only diffuse surfaces**.
Each ray excites a secondary emission of **bounced radiance** along a visible hemisphere of surface element.
We introduce a **regular 3D grid**.
Approximate the bounced radiance by this grid.
Accumulate all the bounced results inside of each cell into that cell.
Thus we have an **initial** accumulated indirect radiance distribution.
After we've got the initial reflected light distribution in the grid, we **propagate** the radiance iteratively around the 3D grid until the **light passes the entire grid**.
Several highlights of the idea:
-We use a **many-lights** approach for approximation of reflected light
- Also we use a regular **3D grid** to approximate lighting in 3D world space and use sampled lit surfaces to initialize the 3D grid with the initial lighting distribution.
- We use a few bands of **SH basis** to approximate lighting in angular space
- The i**terative light propagation approach** lowers down the rendering complexity of many secondary lights.

Sampling the scene for GI

- We use *surfels* (aka “points”, “disks”)
  - Surfel == surface element
- All lit surfels can be flattened into 2D map in light’s space
- Reflective Shadow Maps [DS05]
  - Fastest way to sample lit surfels on GPU
  - Even excessively



Point cloud produced by direct illumination of a single light source could be flattened into 2D map in light’s space. We use Reflective Shadow Maps to sample lit surfaces.
Reflective Shadow Maps are an extension of shadow maps and store not only **depth**, but also **normal and reflected flux** of the surface seen from the light source.
This is essentially a very **fast** method to sample **lit surfaces** of the scene **on GPU**.
All pixels of such a shadow map can bee seen as **indirect light sources** that generate the one-bounce indirect illumination in a scene, similarly to Instant Radiosity approach.

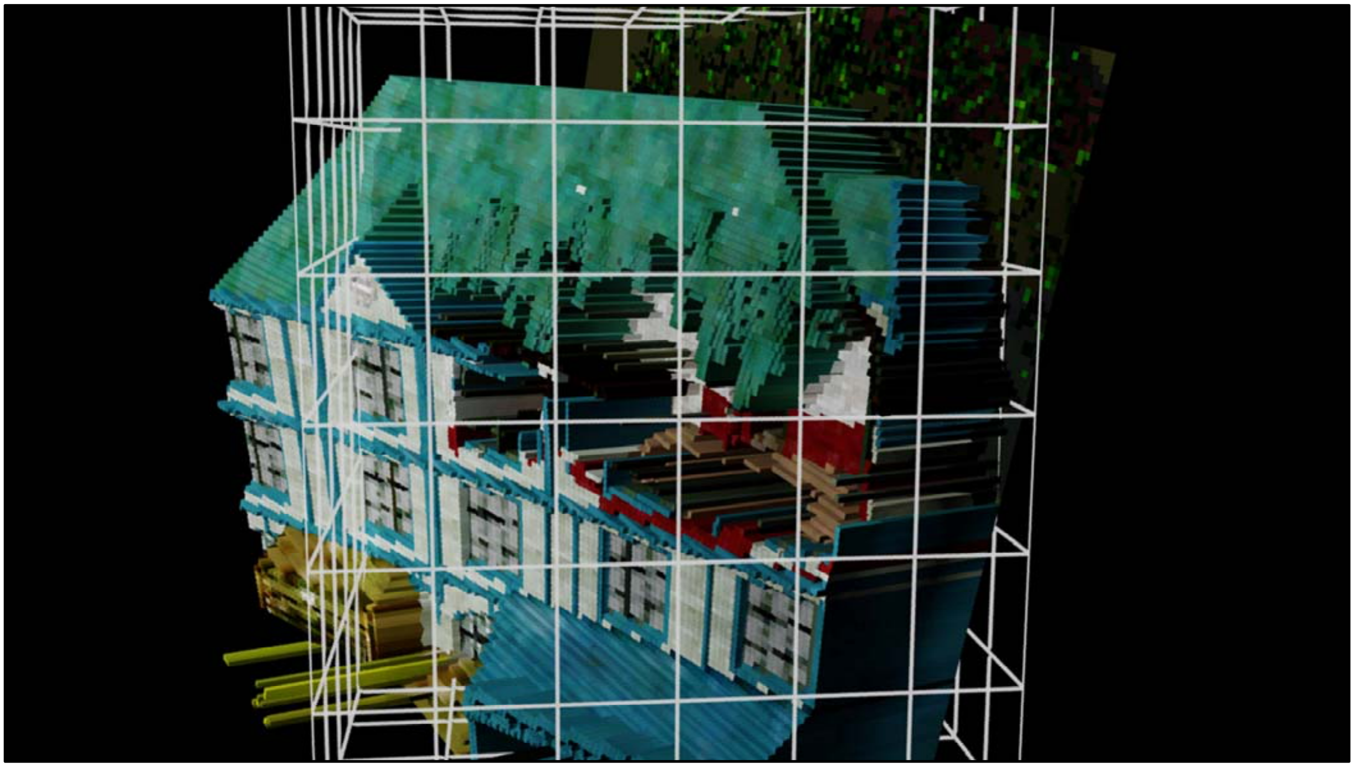


The example of scene sampled with Reflective Shadow Map. Note that all the lit surfels are efficiently represented by a layered 2D texture.

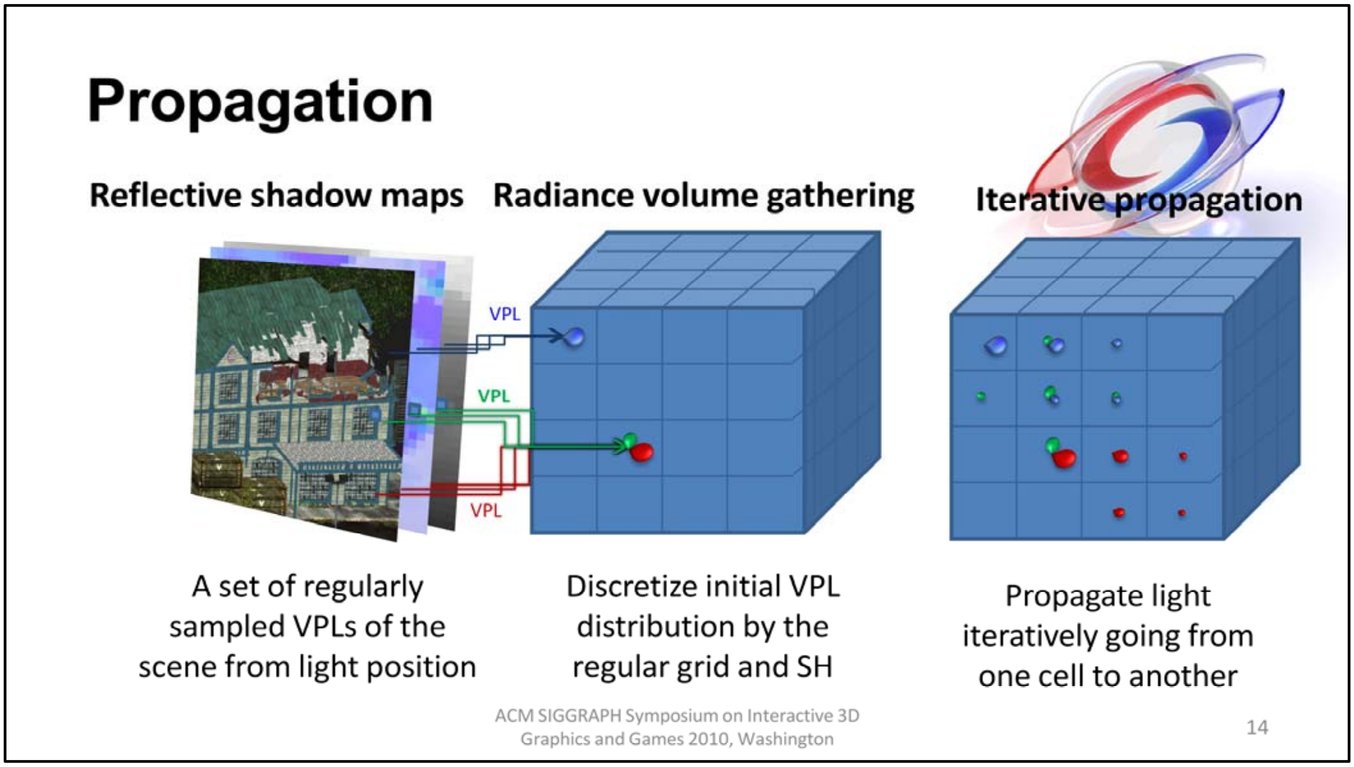
Clustering Surfels

- Lit surfels represented as *Virtual Point Lights*
  - Comes from Instant Radiosity approach [Keller97]
- Distribute each surfel into the closest grid cell
  - Similar to PBGI, light-cuts and radiosity clustering
- Convert all VPLs into outgoing radiance distribution
  - Represent in Spherical Harmonics with lower bands
  - Sum it up in the center of owner grid cell
  - Done completely on GPU using rasterization

The **Reflective Shadow Map** is an input data for this stage. We treat it as a set of **virtual point lights** (VPLs).
We use **point rendering** to distribute each VPL of the RSM into the 3D grid efficiently on **GPU**.
We create a **radiant intensity distribution** out of **orientation** and **colored intensity** of each texel of RSM.
We use **additive blending** to efficiently accumulate contribution of all VLPs from RSM in parallel.
Thus we've got a 3D grid **initialized by the initial distribution of reflected light** in the end of this process.



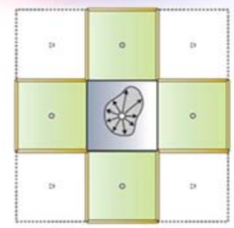
Light Propagation Volume is created as a **bounding volume** for the sampled scene’s surfels.



After we‘ve got an **approximation of outgoing indirect lighting** in each cell of our 3D grid, we can apply an **iterative process of energy propagation** across the grid.

Propagation, cont'd

- Local cell-to-cell propagation across the 3D grid
  - Similar to SH Discrete Ordinate Method for participating media illumination [GRWS04]
- 6 axial directions with contour faces as a propagation wave front
- Accumulate the resulting SH coefficients into the destination cell for next iteration



propagation along axial directions

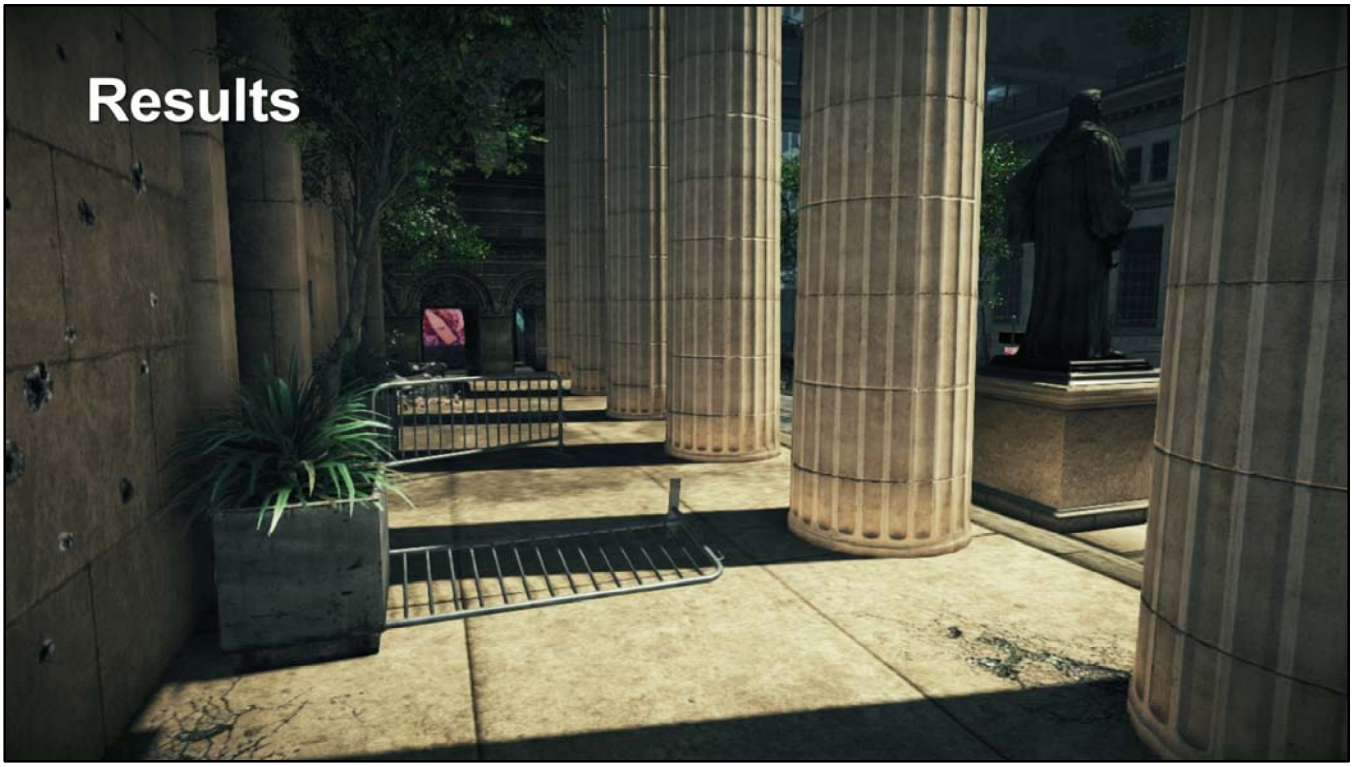
We do several iterations (a precise number is determined by the brightest intensity of initial distribution, see paper for more details) to bring the system to the **energy equilibrium**.
Many variations of this method are widely used in computations of the **scattering process** in participating media.
For each iteration of the propagation we consider **6 axial directions** (depicted with 4 on the top right figure in 2D).
We use contour faces as a **propagation wave front** (the contour depicted in yellow on the top right figure).

Final scene rendering with LPV

- Look-up resulting grid 3D texture at certain position with h/w trilinear interpolation
- Convolve the irradiance with cosine lobe of surface’s normal being illuminated
- Apply dampening factor to avoid self-bleeding
  - Compute directional derivative towards normal
  - Dampen based on gradient deviation from the intensity distribution direction

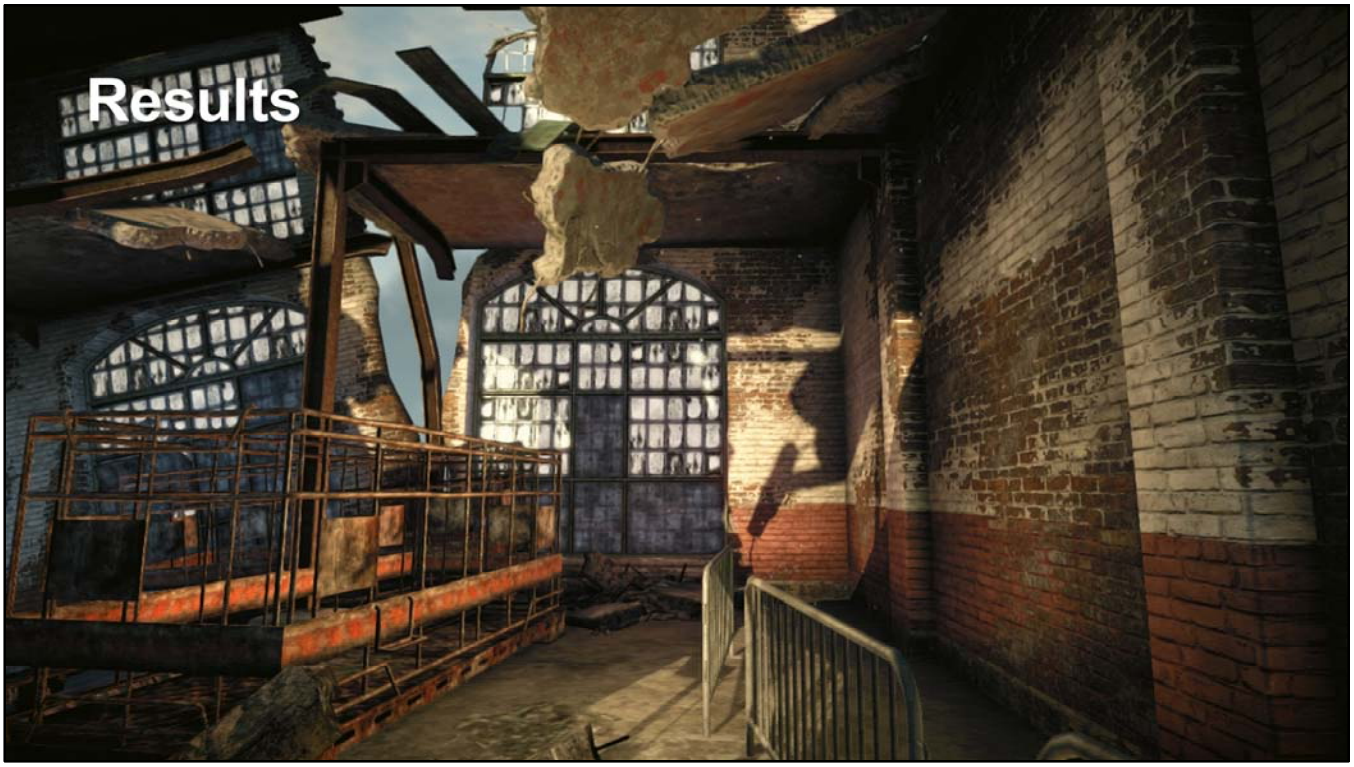
During a usual scene rendering (either forward or deferred) we look up the radiance distribution from this grid using world space position. Then we integrate this distribution with weighting by a clamped cosine lobe induced by surface’s normal.
To avoid improper trilinear interpolation we check if the radiance gradient direction along surface’s normal matches the radiance flow itself. See [KD10] for more details.

This is a screenshot of indoor scene using Light Propagation Volumes

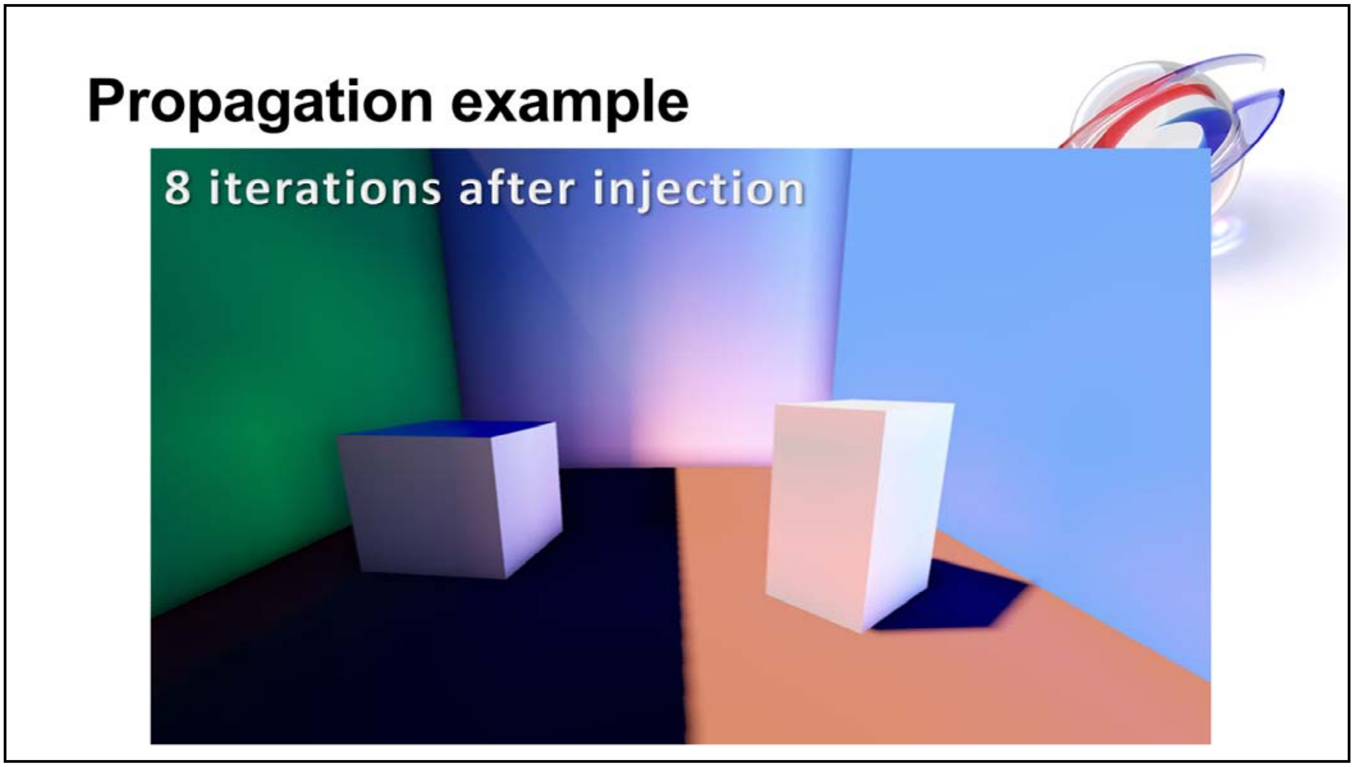


This is a screenshot of indoor scene using Light Propagation Volumes

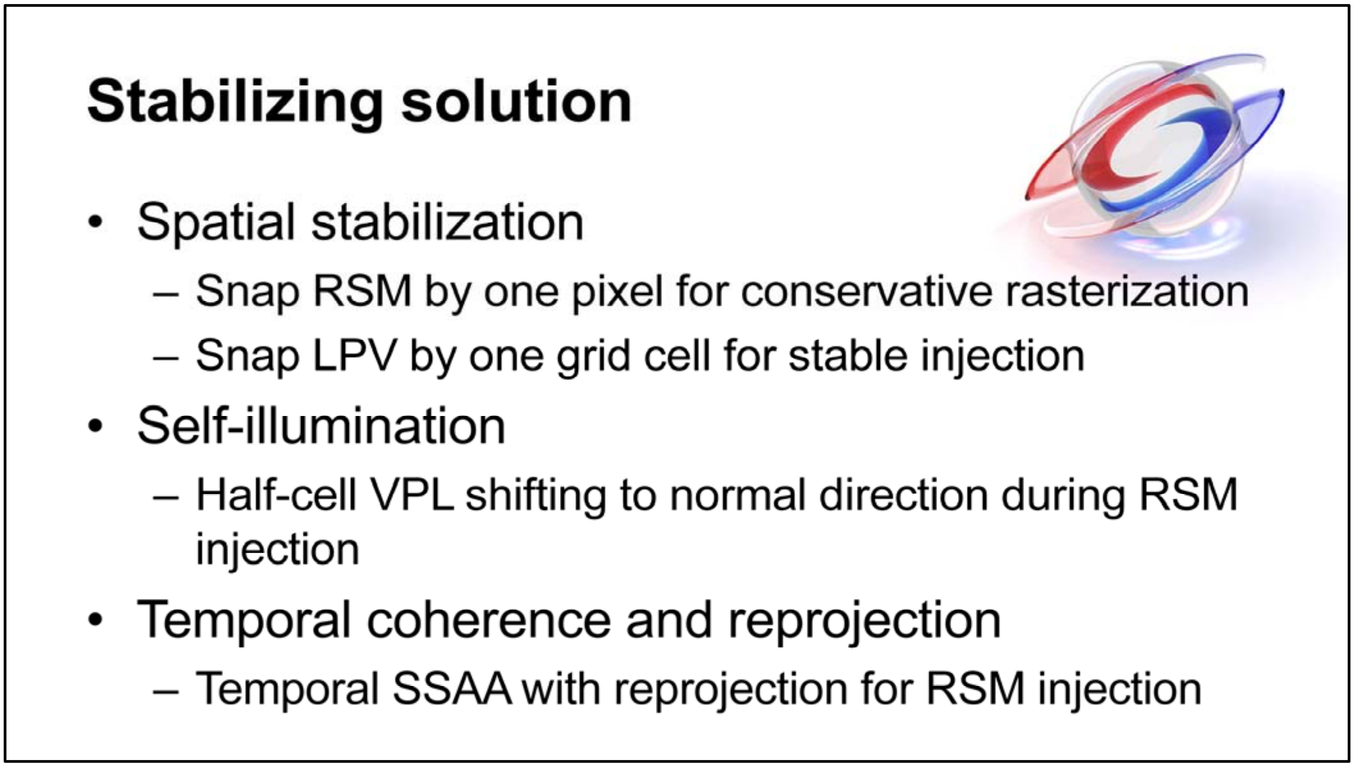
This is a screenshot of indoor scene using Light Propagation Volumes



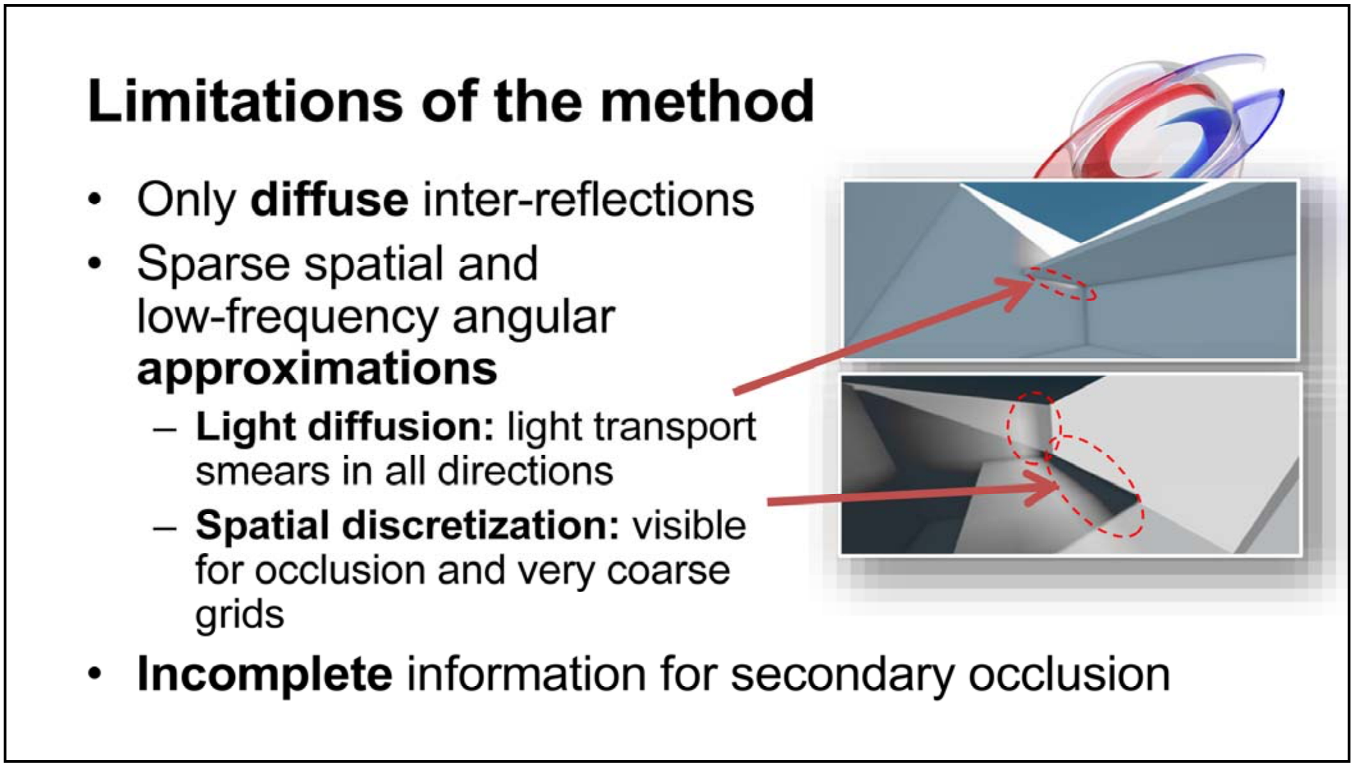
This is a screenshot of indoor scene using Light Propagation Volumes



Here is a sequence demonstrating the propagation results for **different number of iterations** for light propagation stage.
It’s a Cornell-box-like room with the blue right wall, red floor, green left wall (which is in shadow) and grey back wall.
The grid is very **coarse** to emphasize the contribution of each subsequent iteration.



Stability is an important issue of this technique.
Spatio-temporal stability of both LPV and RSM during movements is required.
Temporal antialiasing (jittering) can be employed to achieve better scene sampling with RSM across several frames.
All the issues mentioned on the previous slide could be avoided within these steps (details in the I3D paper).



This method has several obvious limitations.
1. It takes only **diffuse** inter-reflections into account.
2. Also the spatial and angular approximations are the sources of error:
- The **light smearing** happen during the light propagation process because of **low-frequency** angular approximation in each cell of the LPV.
- Another error comes from **spatial discretization** and mostly noticeable with **sparse grids** as shown on the right bottom picture.
3. And the information about the indirect occluders is **incomplete** in our approach as we use only **two views** to sample all the potential occluders, which might be not sufficient for some cases.

Multi-resolution approach

- Render several nested RSMs at different resolutions
  - Inspired by cascaded shadow maps technique
  - Simulates uneven multi-resolution rendering on GPU
  - Distribute objects into **different** RSMs based on their size
- Inject RSMs into corresponding LPVs
  - Create nested LPV grids that bound RSM frustums
  - Do propagation and rendering independently
  - Propagate from inner LPV to outer one

Using a single LPV to compute the light propagation in an entire scene (with acceptable resolution) would require a **very large grid**.
Instead we use a **set of nested grids** moving with the viewer similar to cascaded shadow maps but in 3D.
The nested grid approach allows us to use **grids of smaller size** (typically **32^3** cells) and thus **reduce** the number of required propagation iterations.
The important property of the cascaded approach we use is the **orthogonality**. That allows us to have different scales of indirect radiance distribution at different cascades. Also that means that we can simply add up the contribution of all cascades together without caring about overlapping indirect illumination.

Examples of cascaded approach in a huge outdoor scene.

Extensions

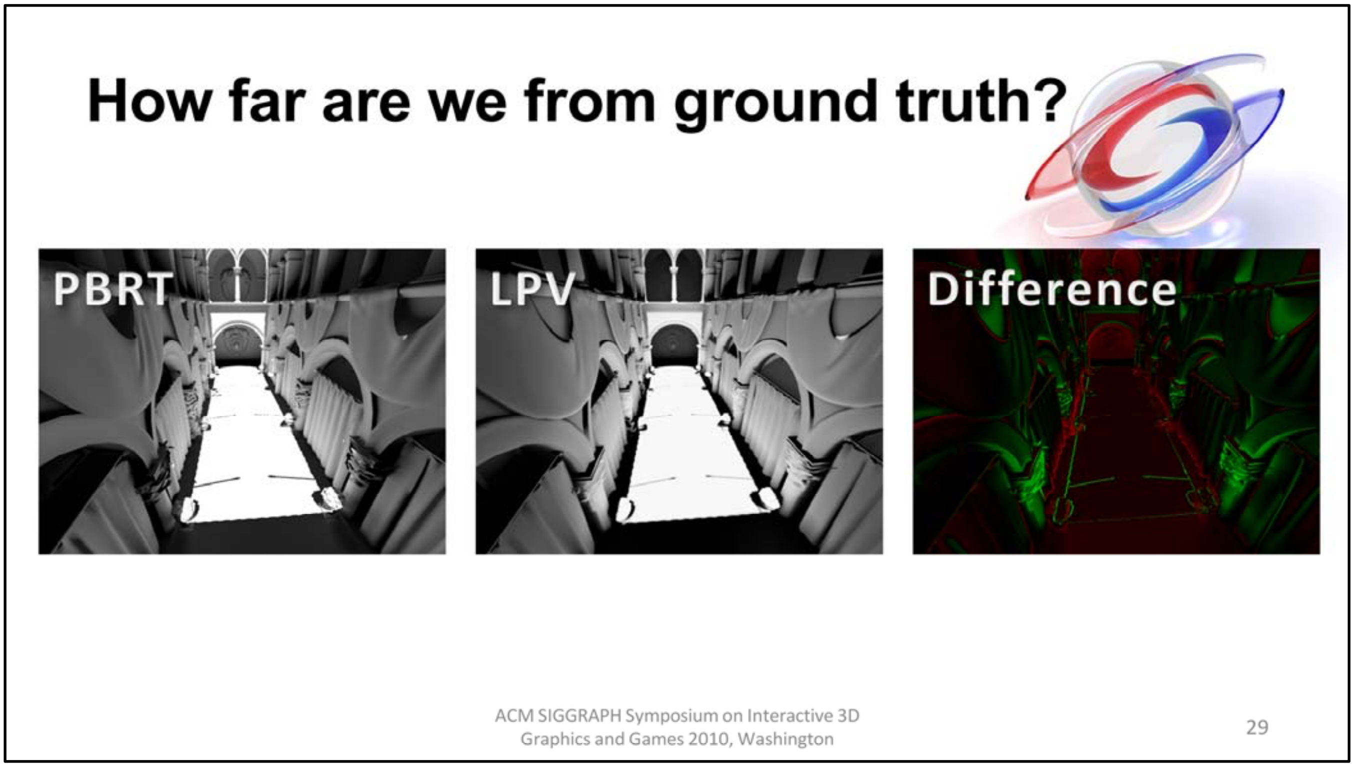
- Transparent objects
- Lighting caching for massive lighting approximation
  - Inject analytical radiance into grid cells covered by light
- Secondary occlusion with additional occlusion grid
  - Multiple bounces possible using the same trick
- Glossy reflections by partial matching in LPV
- Participating media illumination
  - Comes inherently from the propagation process' nature

Light Propagation Volumes has a lot of **natural extensions** as it is essentially a coarse **volumetric representation** of radiance distribution.
Some of these extensions are mentioned here. For the full list of extensions and examples please refer to I3D 2010 paper "Cascaded Light Propagation Volumes" [KD10]

Here is an example of global illumination applied to **particle effects**.

Why does it work so good?

- Human perception of Indirect Lighting
  - Very sensitive for contact lighting (corners, edges etc.)
  - Indirect lighting is mostly in low frequency
    - Even for indirect shadows
    - Smooth gradients instead of flat ambient in shadow
  - Approximated as diffusion process in participating media
- Cascades: importance-based clustering
  - Emitters are distributed across cascades based on its size



Here is the **comparison** to the “ground truth” off-line rendering.
Reference solution is **on the left** and rendered with PBRT.
The result of our method with 3 cascades is **in the middle**.
You can see the difference on the right colored as a red/green heatmap.
The most of the error comes from a lack of fine-grained indirect occlusion on this image.
However that found to be not an issue in the production.

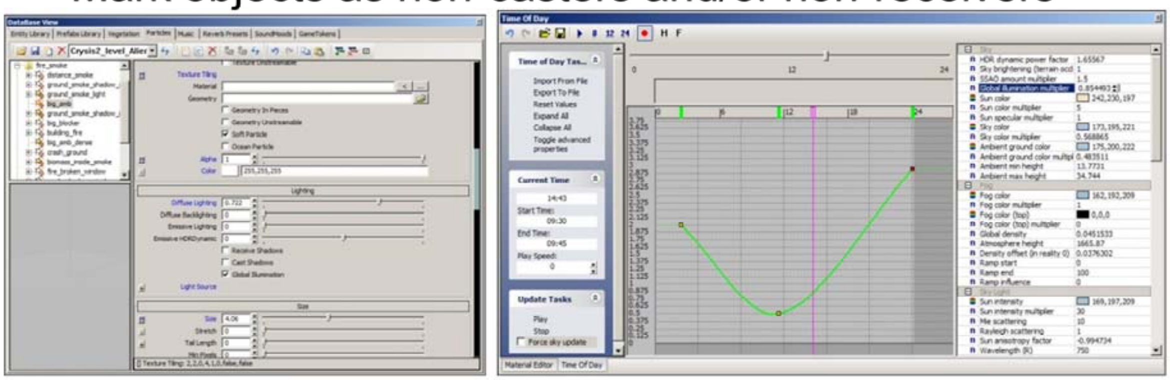
Comparison

	Lightmaps	Precomputed Radiance Transfer	Light Propagation Volumes
Image quality	Very good	Good	Good
Memory budget	Medium	Medium	Low and fixed
Dynamic lighting	-	+	+
Dynamic objects	-	-	+
Secondary Occlusion	+	+	+ (via extensions)
Multiple bounces	+	+	- (unstable solution)
Area covered	Whole scene	Whole scene	Limited (w/o cascades)
Auxiliary data	Moderate	Huge	None
Affecting production	Strong	Strong	Low

Here is a comparison table provided for our method versus the most commonly used real-time methods in games.
The area covered by the technique becomes an important property for our technique. The area covered by one cascade is small, however the area covered by three cascades becomes sufficient for majority of game scenes.

Tools for game production

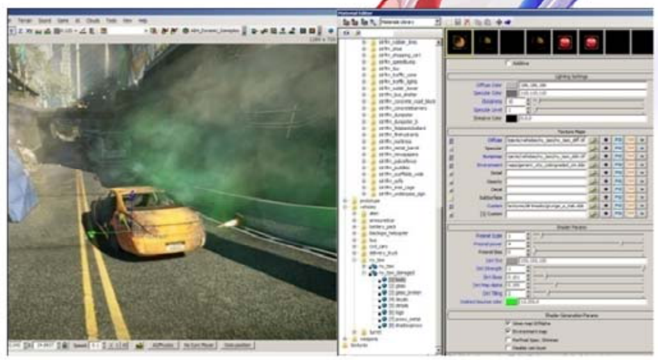
- GI editing tools for artists:
  - GI intensity for each direct light contributing into GI
  - Mark objects as non-casters and/or non-receivers



We provide a rich **set of tools** to control Global Illumination.
We have global and time-varying intensity multiplier for indirect lighting.
Also we have some additional tools.
We provide per-object artistic control over global illumination **receivers** for both solid and transparent objects including particle effects.

Tools for game production

- GI tools for artists:
  - Per material indirect color and intensity
  - Optionally apply on any transparent objects and particles



  - Clip areas: provides control over indoors
  - Transition areas: provides smooth GI changes across level areas / game events

We provide a **per-object** control over the bleeding **intensity** and **color**. That means that an artist can change the **indirect color** of an object without touching any other material parameters **in real-time**. You can see the example in the middle figure. The bounced color of the yellow taxi is set to green. You can see the Global Illumination effect from the taxi on the surrounding particles.

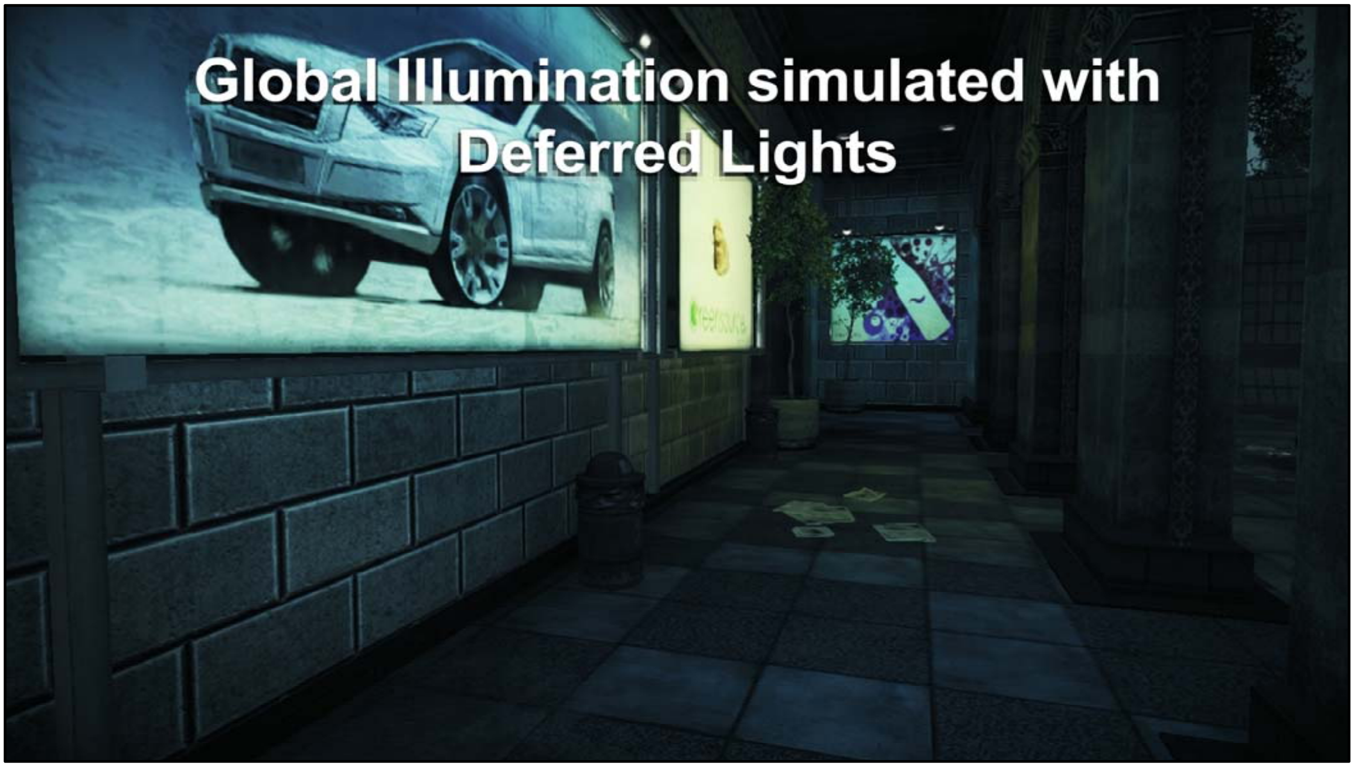
….

This set of tool is provided to be sufficient and very convenient for real-time tweaking of Global Illumination effect by artists.

Combination with other techniques

- Multiply with SSAO to add micro-occlusion details
- Deferred environment probes
  - Combined to augment for distant GI
- Fill lights and deferred lights
  - Simulating GI with fill lights at some places
  - Important for artists for GI stylization

We use Screen-Space Ambient Occlusion as a **multiplicative factor** to add more high frequency occlusion details.
Also we provide our artists with **fill lights**, **negative lights** and simple **deferred point lights** to give them freedom in achieving the effects they want to. This becomes especially important in places or cut-scenes where an **dramatic view** is much more important rather than physically-correct Global Illumination effect.
In addition, we augment our technique with preconvolved diffuse environment probes to approximate **distant indirect lighting** in outdoors. This term usually comes from huge global contributors like sky, terrain, ocean etc.



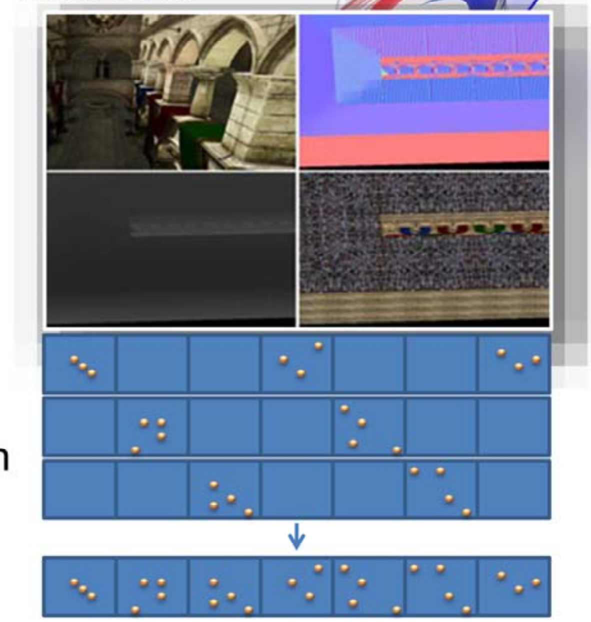
This is an example of the simulated Global Illumination manually created by artists. In this particular case the sampling of the scene by several area light sources would take a lot of time. Moreover as light sources are very coarse spatially, it is possible to easily emulate the Illumination with several deferred point light sources with almost no performance impact.

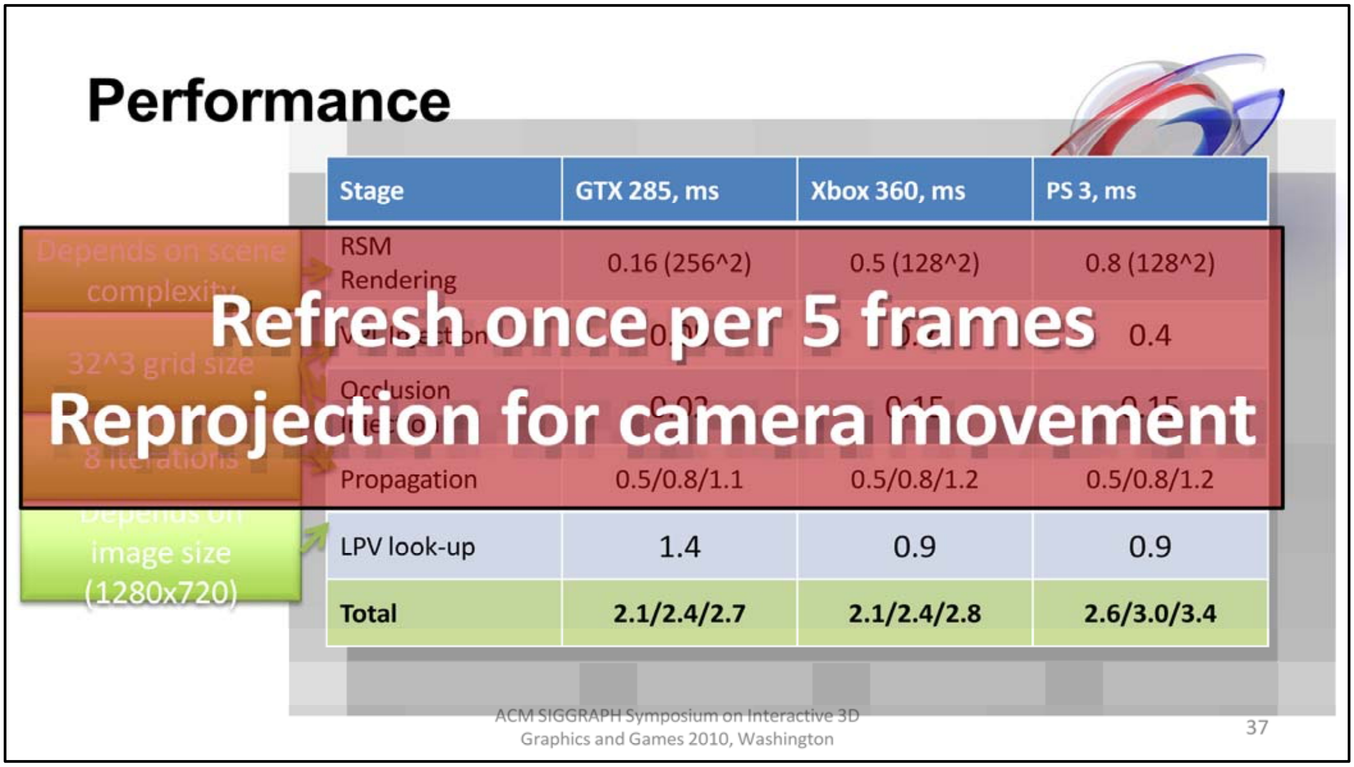
Console optimizations

- For both consoles
  - Store everything in signed **QUVW8 format**, [-1;1] with **scaling factor**
  - Use **h/w 3D textures** and trilinear filtering
- Xbox 360
  - Unwrap RT **vertically** to avoid **bank conflicts** during injection *(next slide)*
  - Use API bug **work-around** to resolve into a 3D slice
- PlayStation 3
  - Use memory aliasing for **render into 3D texture**
  - Use **2x MSAA aliasing** to reduce pixel work twice

Console optimizations , cont'd

- Render Reflective Shadow Map
  - Usually 128 x 128 is ok
- Inject each pixel into unwrapped LPV with a swarm of points
  - 16384 points in one DIP
  - Use vertex texture fetch on X360
  - Use R2VB on PlayStation 3
- Multi-layered unwrapping to avoid bank conflicts during RSM injection
- Combine LPV rendering pass with SSAO to amortize the cost





Performance of our method is provided in this table. Note that different stages of the algorithm depend on different data, like scene complexity and screen resolution.
This is a very important advantage for real-time technique, because it decomposes the complexity of the algorithm, providing more predictable performance.
Notice that we can reuse the results of propagation across **multiple frames.**

Performance, cont'd

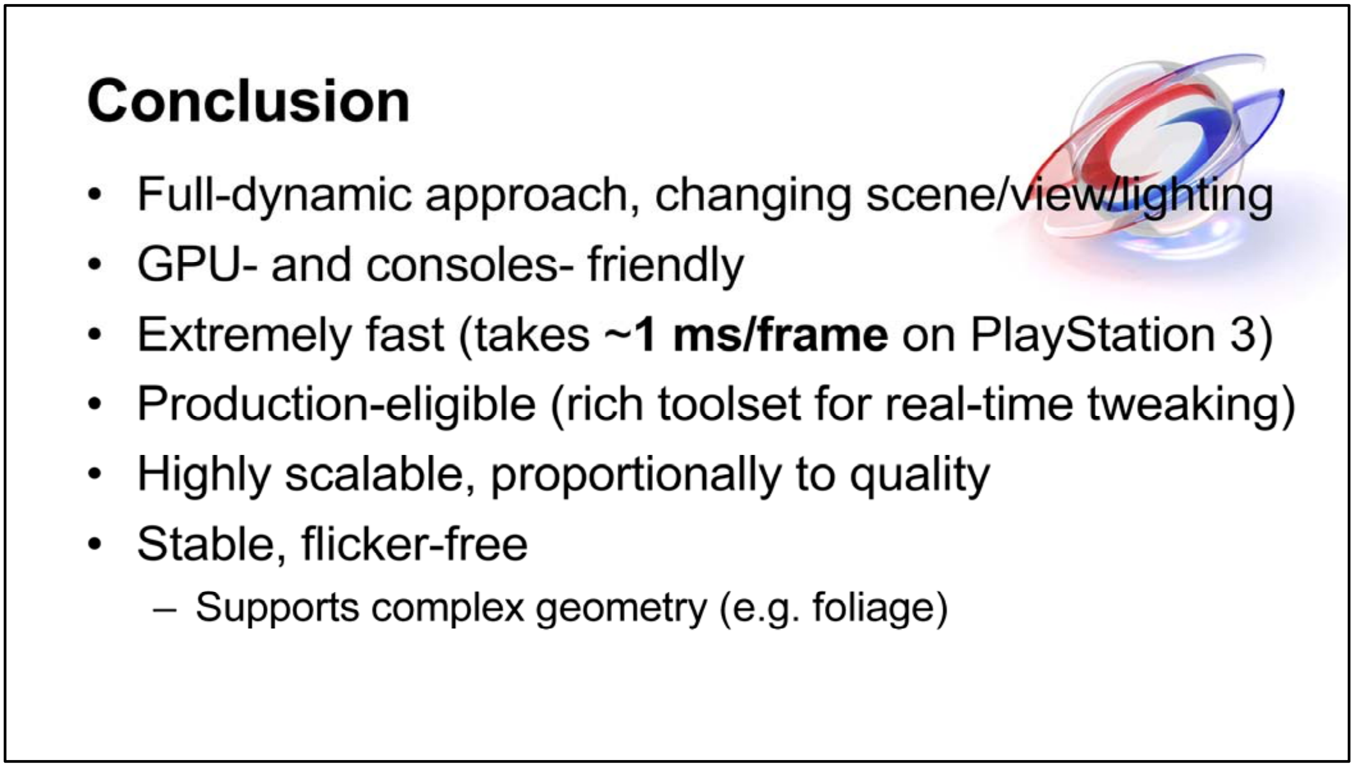
Stage	GTX 285, ms	Xbox 360, ms	PS 3, ms
RSM Rendering	0.16 (256^2)	0.5 (128^2)	0.8 (128^2)
VPL Injection	0.05	0.2	0.4
Occlusion Injection	0.02	0.15	0.15
Propagation	0.5/0.8/1.1	0.5/0.8/1.2	0.5/0.8/1.2
LPV look-up	1.4	0.9	0.9
Total (per frame)	**1.5/1.6/1.7**	**1.1/1.1/1.3**	**1.2/1.3/1.4**

Once per 5 frames: RSM Rendering, VPL Injection, Occlusion Injection, Propagation

Once per frame: LPV look-up

ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2010, Washington 38

We regenerate GI data **once per 5 frames** (which is proved to be enough) and fade in new results smoothly.
Also we do a reprojection from the old LPV to the new LPV in case of intensive camera movements.



We presented an efficient method for the rendering of plausible indirect lighting in fully dynamic, complex scenes in real-time that uses volumetric representations of the light and geometry in a scene.
We demonstrated our method in various real game scenes in combination with wide-spread real-time rendering techniques.

I'd like to acknowledge the whole Crytek team for their support and especially Tiago Sousa and Pierre-Yves Donzallaz for their help.

References

- [Bunnel05] Bunnel, M. 2005 “Dynamic ambient occlusion and indirect lighting”, GPU Gems 2
- [Christensen07] Christensen, P. 2007. “Point-based approximated color bleeding,” Tech Memo, Pixar.
- [DS05] Dachsbacher, C., and Stamminger, M. 2005. Reflective shadow maps. In Proc. of the Symposium on Interactive 3D Graphics and Games
- [GRWS04] Geist, R., Rasche, K., Westall, J., and Schalkoff, R. J. 2004. Lattice-boltzmann lighting. In Rendering Techniques 2004 (Proc. of the Eurographics Symposium on Rendering
- [KD10] Kaplanyan A., Dachsbacher C. 2010. Cascaded Light Propagation Volumes, In Proc. of the ACM SIGGraph Symposium on Interactive 3D Graphics and Games
- [KELLER97] Keller, A. 1997. Instant radiosity. In SIGGRAPH ’97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques