AMD

Fusion[12]

DEVELOPER SUMMIT

AMD Fusion12 DEVELOPER SUMMIT

GLOBAL ILLUMINATION USING RAY-BUNDLE TRACING

Yusuke Tokuyoshi
Square Enix
Researcher

Takashi Sekine
Square Enix
R&D Graphics Engineer

AGENDA

- Global Illumination
- Ray-Bundle Tracing
- GI Prebaking
  - Method
  - In-House Tool
- Interactive GI
  - Instant Radiosity
  - Bidirectional Sampling with Ray-Bundles
- Summary

GLOBAL ILLUMINATION

- Improve visual realism of interactive applications
- Paths of light are ***randomly*** sampled according to a probability distribution function (PDF)
  – Path tracing [Kajiya 1986], photon mapping [Jensen 1996], instant radiosity [Keller 1997]
- Computationally expensive
  – Accuracy is dependent on the number of samples
- Efficient tracing techniques are desired

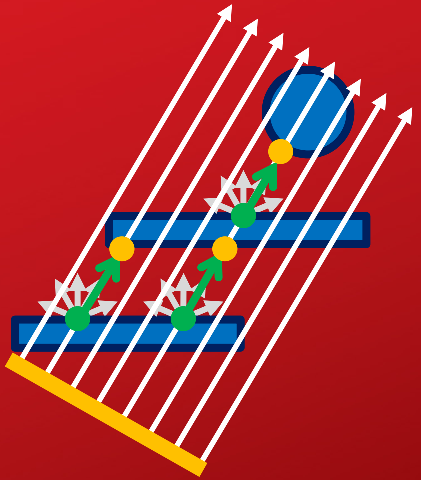

1.6 secs


6 secs


76 secs

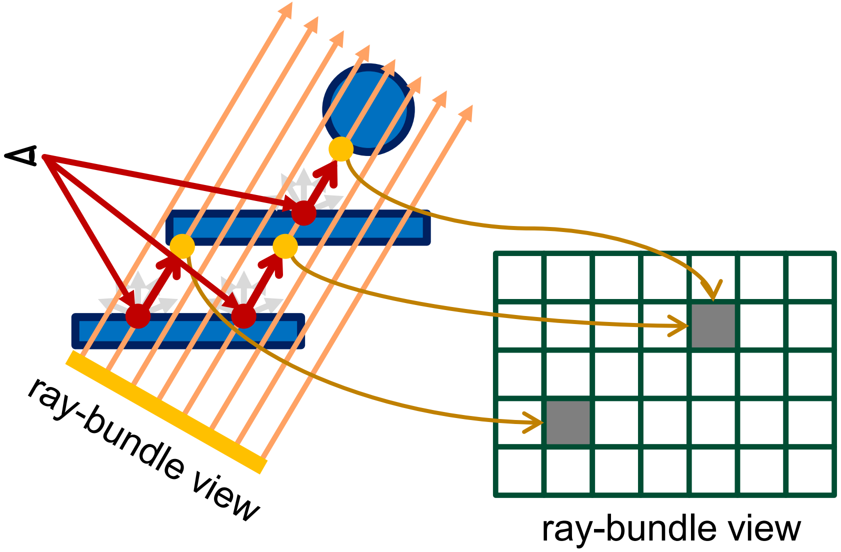
Path tracing on a CPU (480x270 pixels)

RAY-BUNDLE TRACING



RAY-BUNDLE TRACING | BASIC ALGORITHM

- Set of parallel rays
- Visibility test is accelerated with hardware rasterization
  - Focus on a single global direction
  - Rasterize for all fragments in parallel
  - Issue: **how to handle multiple fragments in a single pixel?**



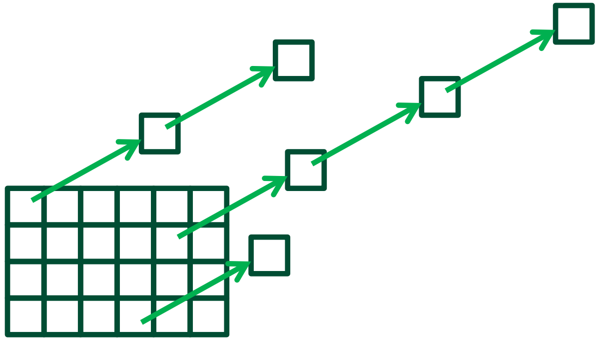
RAY-BUNDLE TRACING | RELATED WORK

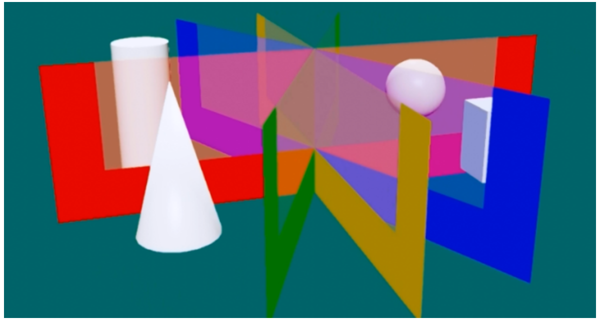
- Depth peeling [Hachisuka 2005; Niessner et al. 2010]
  - ☹ Multi-pass
- *K*-buffer [Hermes et al. 2010]
  - ☺ Single-pass
  - ☹ Limited storage per pixel
- Stochastic depth buffering [Thomsen and Nielsen 2011]
  - ☺ Single-pass
  - ☹ Approximate solution

Final gathering using depth peeling [Hachisuka 2005]

RAY-BUNDLE TRACING | OUR APPROACH

- Per pixel linked-list on DX11 GPU [Yang et al. 2010]
  - ☺ Single-pass
  - ☺ Unlimited storage per pixel
  - ☺ No need to sort for GI (opaque objects)
  - ☹ Unpredictable memory usage
    - Excessive memory has to be allocated to avoid overflow

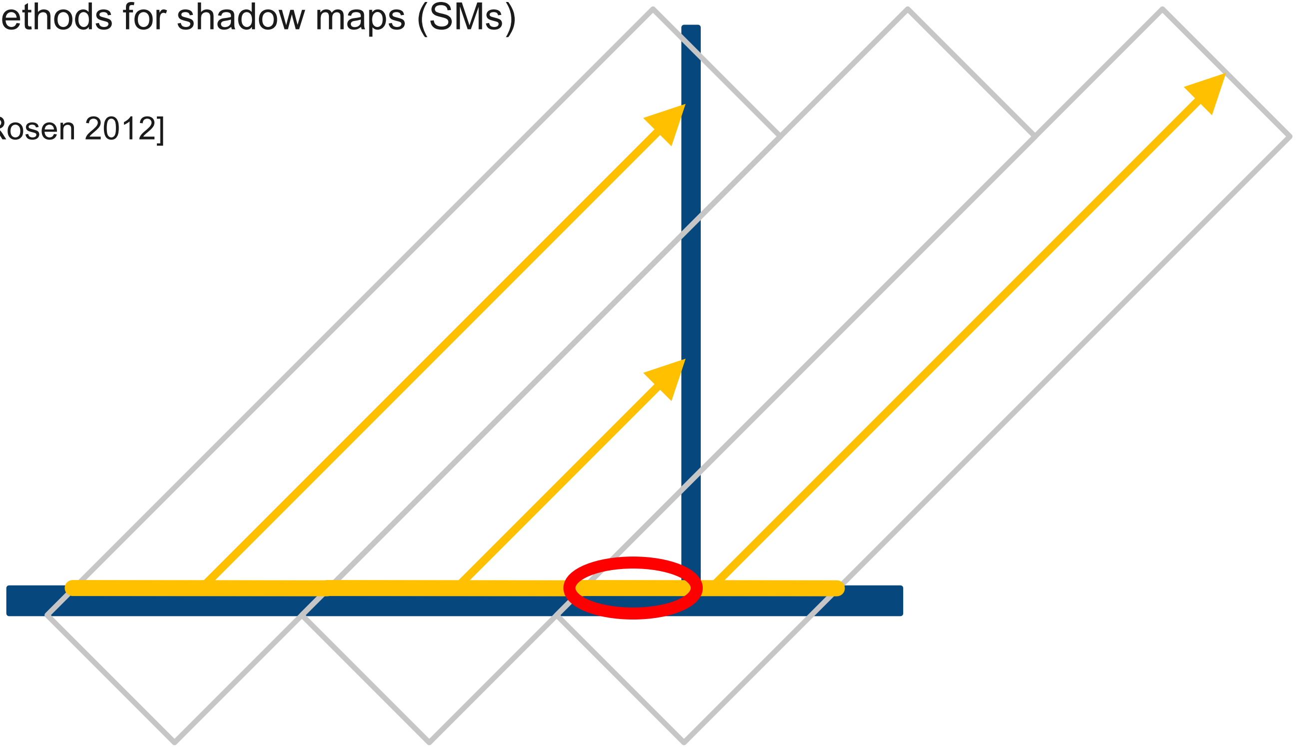




Order independent transparency (OIT)

RAY-BUNDLE TRACING | LIMITATIONS

- Light leaking
  - Reduced by anti-aliasing methods for shadow maps (SMs)
    - Cascade [Lloyd et al. 2006]
    - Rectilinear texture warping [Rosen 2012]



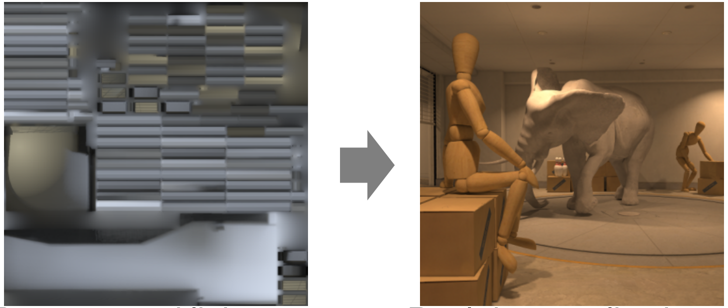
RAY-BUNDLE TRACING | APPLICATIONS

GI prebaking
for static scenes

Robust interactive GI
for dynamic scenes

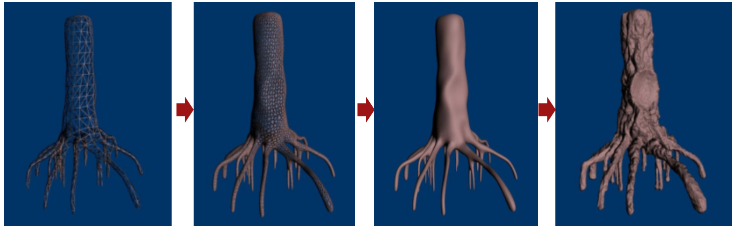
GI PREBAKING
METHOD

- Light maps
  - ☺ Easy to improve realism in real-time applications
  - ☹ Long precomputation time



Precomputed light maps　　Real-time applications

- Tessellation
  - DX11 GPUs support tessellation for real-time rendering
    - Arbitrary displacements by the domain shader
    - Easy to implement
    - Memory efficient
  - Baking system must support the same tessellation
  - This is difficult for off-line CPU rendering



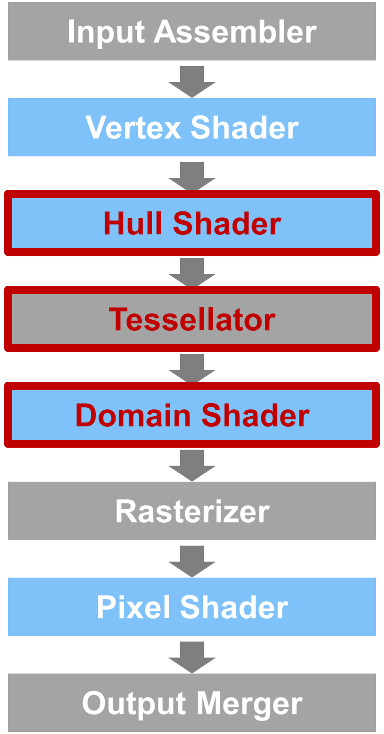
On-the-fly tessellation

GI PRE-BAKING | APPROACHES

- Pretessellation
  - ☹ Memory consuming
- Direct ray tracing with on-the-fly tessellation [Smits et al. 2000]
  - ☺ Accurate
  - ☹ Computationally expensive
  - ☹ Difficult to implement for arbitrary displacements

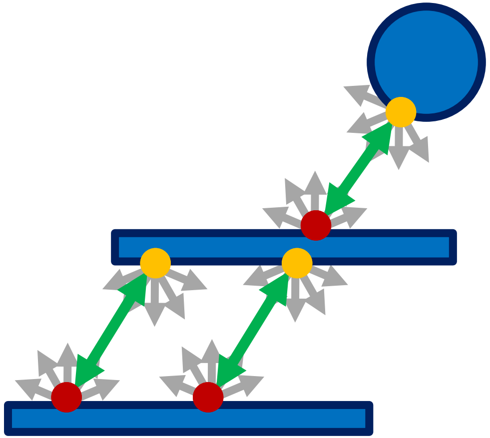
- Ray-bundle tracing on the GPU
  - Simply utilize the tessellator stage
  - The domain shader can be shared with real-time rendering

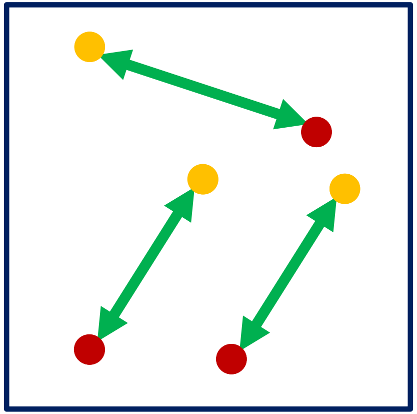
The same displacement as real-time rendering



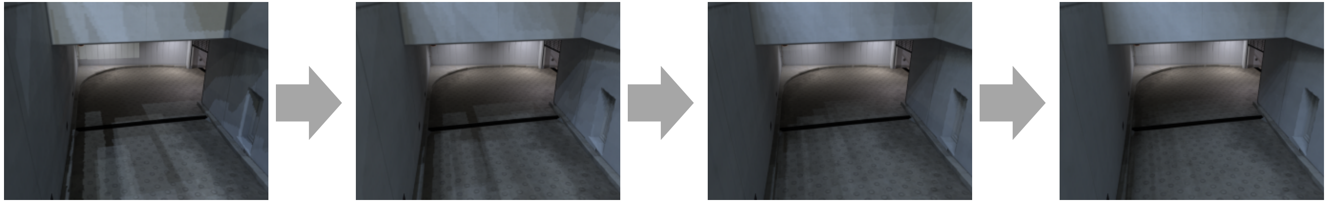
GI PRE-BAKING | RADIANCE EXCHANGE

- [Hermes et al. 2010]'s updating scheme





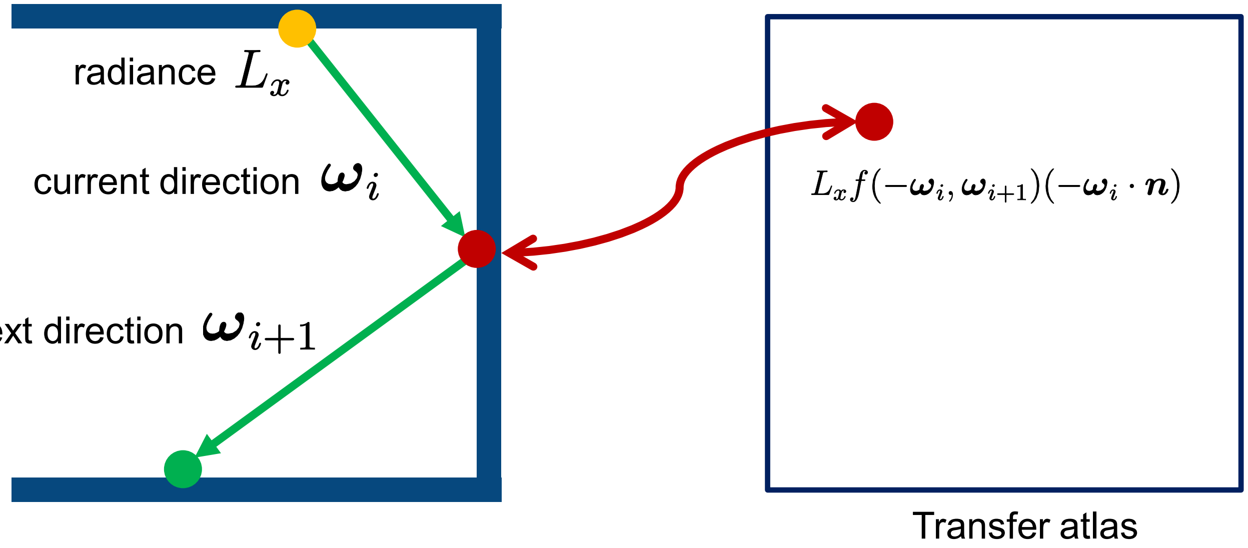
Texture atlas (light maps)



100 directions 200 directions 400 directions 800 directions

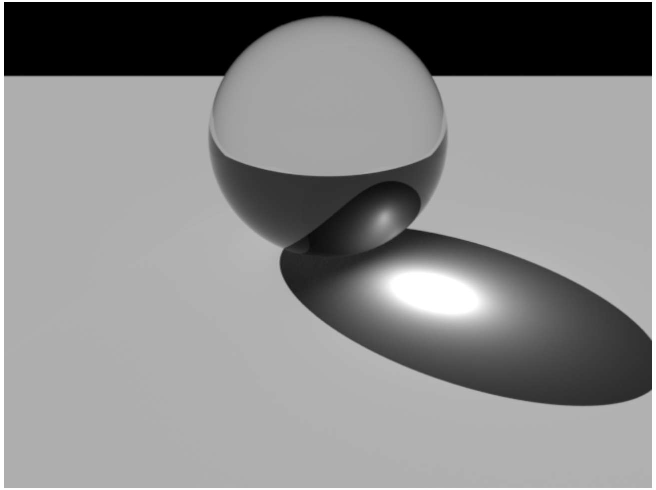
GI PRE-BAKING | RADIANCE EXCHANGE

- Arbitrary BRDFs
  - Use an additional texture atlas



GI PRE-BAKING | LIMITATIONS

- Limited by memory capacity
  - All resources must be in device memory
    - Ray-bundles, scene data, texture atlases, etc.
  - Vast scenes must be split
  - Tessellated scenes are recommended to save memory
- Weak in highly glossy surfaces
  - Cannot render caustics from perfectly specular surfaces



Caustics

GI PREBAKING

IN-HOUSE TOOL

GI PREBAKING | SHOWCASE

GI PREBAKING | SHOWCASE

GI PREBAKING | SHOWCASE

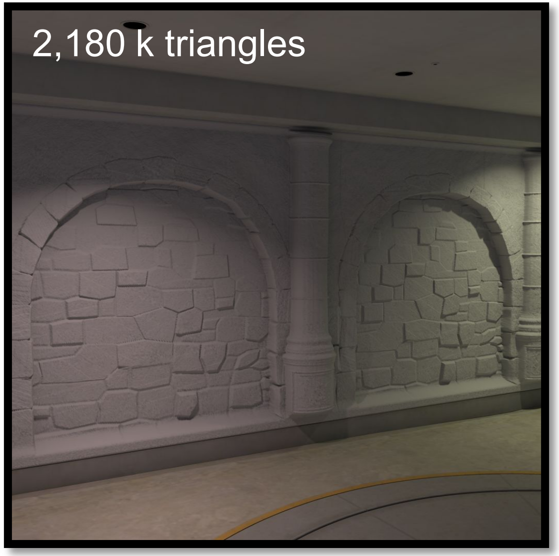
GI PREBAKING | SHOWCASE

GI PREBAKING | SHOWCASE

GI PREBAKING | SHOWCASE

GI PREBAKING | TESSELLATION

Displacement mapping



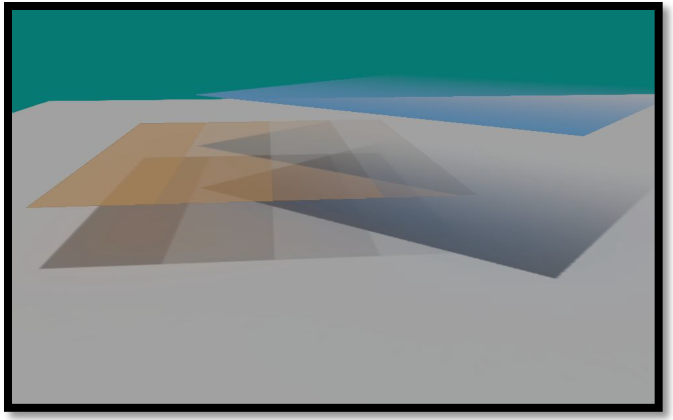
	Vertex Buffer	Height Maps
Pretessellation	178.8MB	0 MB
GPU tessellation	12.3 MB	1 MB

Light maps: 1024^2 pixels x 15
Ray-bundle: 2048^2 pixels
Directions: 3000

GI PREBAKING | ORDER INDEPENDENT TRANSPARENCY

- Sort fragments in a ray-bundle [Yang et al 2010]
  - ☺ Perfect solution
  - ☹ More complex implementation
    - Divide rasterization pass for opaque objects and transparent objects
    - Need for sorting pass
  - ☹ Increased computation time

- Stochastic approach [Enderton et al. 2010]
  - ☺ Simple implementation
  - ☺ Memory efficient
    - Node is stochastically stored according to the opacity
  - ☹ Increased variance
    - Need many samples



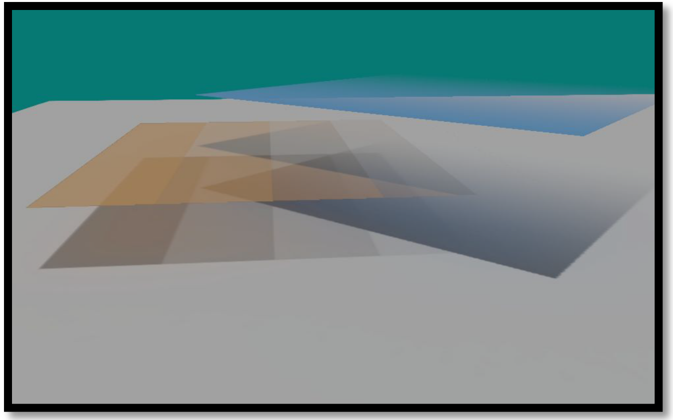
GI with transparent objects

GI PREBAKING | ORDER INDEPENDENT TRANSPARENCY

- Sort fragments in a ray-bundle [Yang et al 2010]
  - ☺ Perfect solution
  - ☹ More complex implementation
    - Divide rasterization pass for opaque objects and transparent objects
    - Need for sorting pass
  - ☹ Increased computation time

Not so computationally expensive for our background objects (almost opaque)

- Stochastic approach [Enderton et al. 2010]
  - ☺ Simple implementation
  - ☺ Memory efficient
    - Node is stochastically stored according to the opacity
  - ☹ Increased variance
    - Need many samples



GI with transparent objects

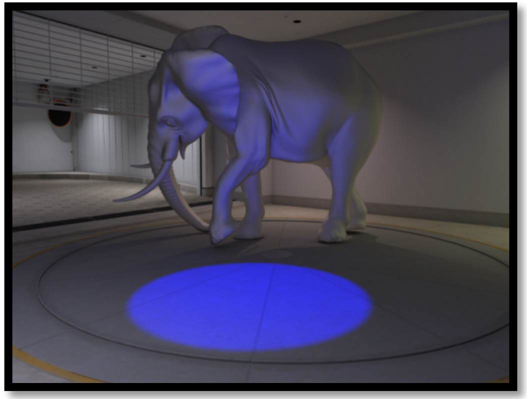
GI PREBAKING | LIGHT MAPS FOR GLOSSY MATERIALS

- Spherical harmonics
  - Used for rough glossy surfaces with 4-9 base coefficients

$$Y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}K_l^m \cos(m\varphi)P_l^m(\cos\theta), & m > 0 \\ \sqrt{2}K_l^m \sin(-m\varphi)P_l^{-m}(\cos\theta), & m < 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0 \end{cases}$$

$$K_l^m = \sqrt{\frac{(2l+1)}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}$$

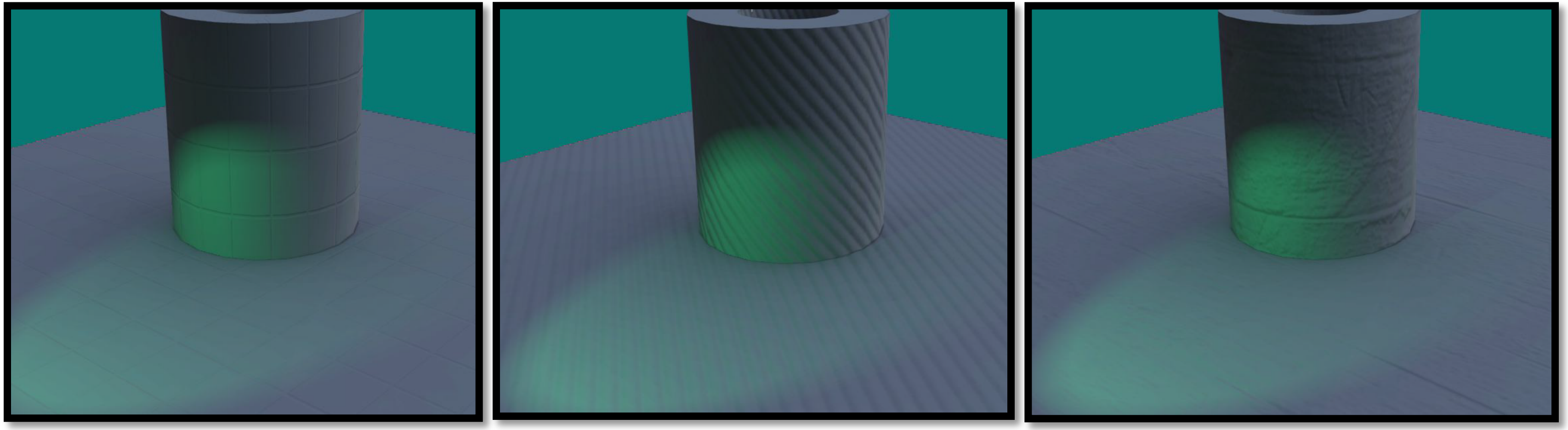
P_l^m : Associated Legendre polynomials



Light maps: 1024^2 pixels x 16
Ray-bundle: 2048^2 pixels
Directions: 3000
SH Basis: 9

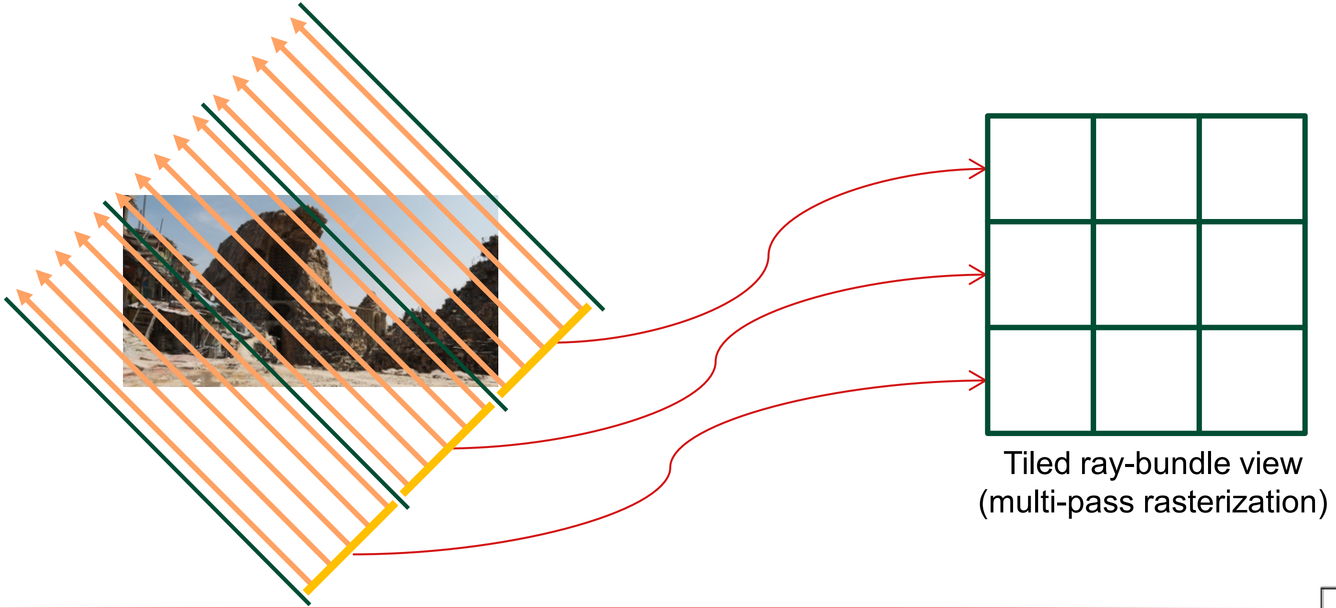
GI PREBAKING | DYNAMIC BUMP MAPPING

- Our system evaluates SH light maps using the runtime normal vector
- For changing surface details
  - E.g. dynamic bump mapping
  - Neglecting global light transport, but plausible



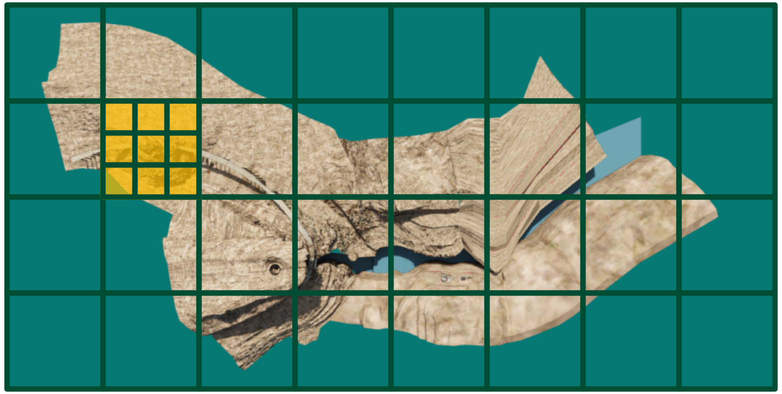
GI PREBAKING | TILING FOR VAST SCENES

- Ray-bundle resolution is limited by memory capacity
- Solved by tiling [Thibieroz 2011]
  - Multi-pass
  - Arbitrary resolution by sacrificing time



GI PREBAKING | ADAPTIVE SOLUTION FOR VAST SCENES

- Light map resolution is limited by memory capacity
- Split the scene
  - Generate light maps for each area
  - Currently focused area:
    - High-resolution light maps (output)
    - Dense ray-bundles
  - Others:
    - Low-resolution texture atlas (temporary)
    - Sparse ray-bundles

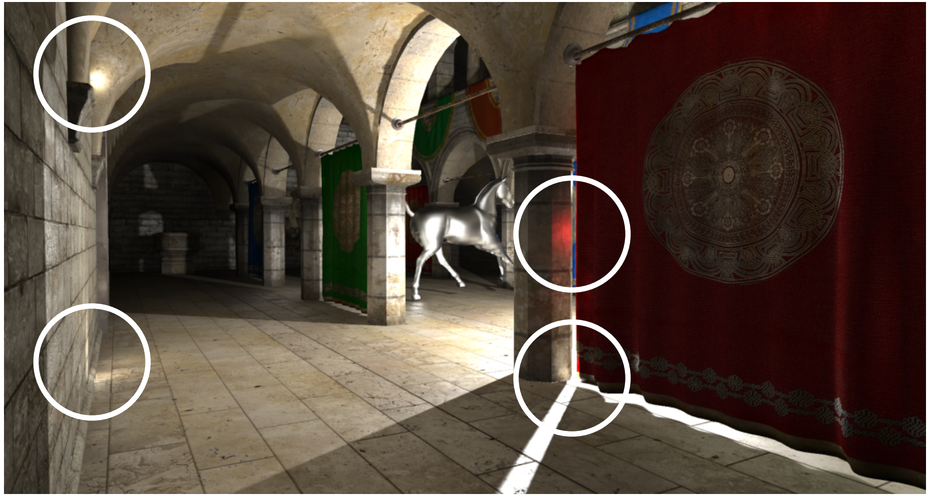


INTERACTIVE GI

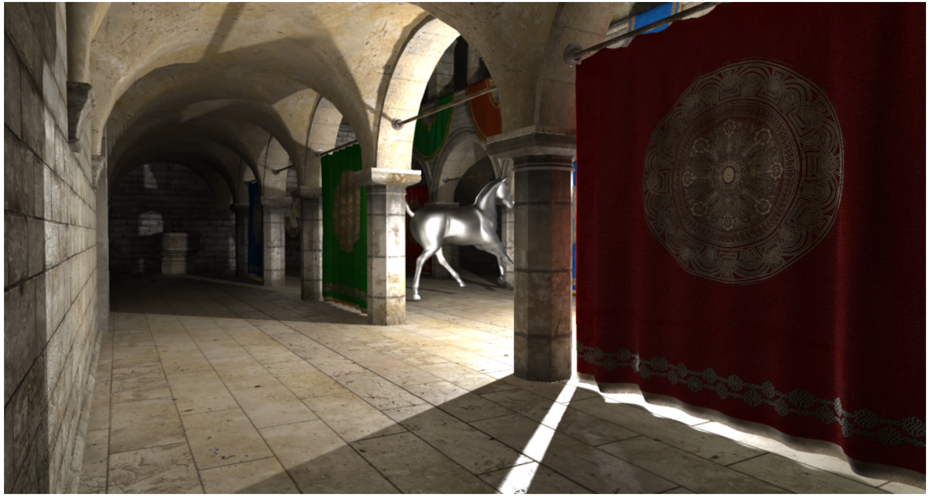
INTERACTIVE GI | OUR CONTRIBUTION

- Classic instant radiosity suffers from a large numerical error
- Ray-bundles reduce the error

GPU: Radeon HD 6990



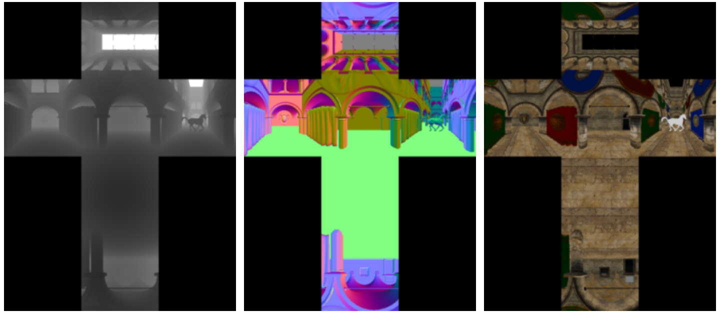
Only VPLs
(Spike artifacts and flickering)
32 ms



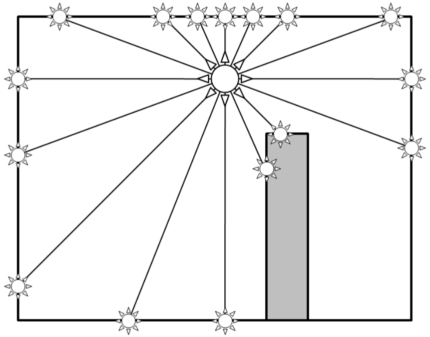
Bidirectional approach
(Artifacts are reduced)
44 ms

INTERACTIVE GI | INSTANT RADIOSITY

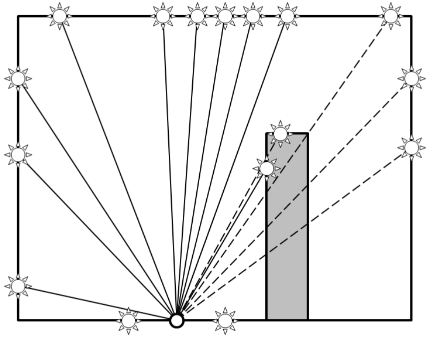
- Virtual point lights (VPLs) are emitted from light sources
- Visibility can be resolved by rasterization
  - Render reflective shadow maps (RSMs) [Dachsbacher and Stamminger 2005]
  - Sample VPLs from RSMs stochastically
  - Render SM for each VPL
  - Shading from VPLs



Cube RSM of a point light



Sample VPLs



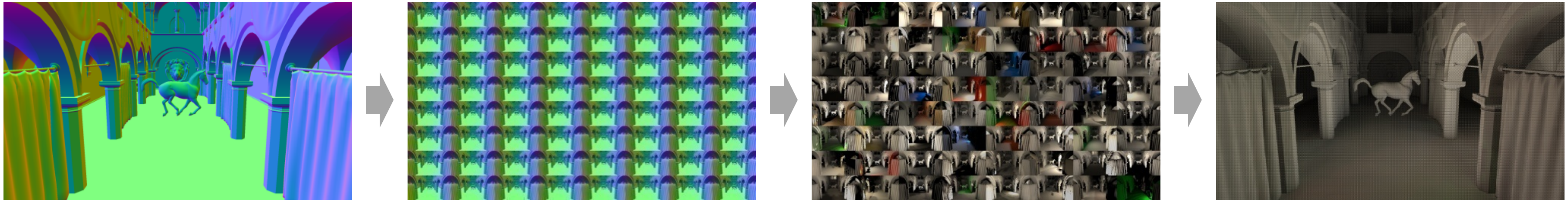
Shading

INTERACTIVE GI | INTERLEAVED SAMPLING

- Interleaved sampling reduces the shading cost
- High-frequency noise is removed by geometry-aware filtering
- Indirect illumination is low-frequency

(256 VPLs) / (8x8 interleave) = 4 VPLs / pixel

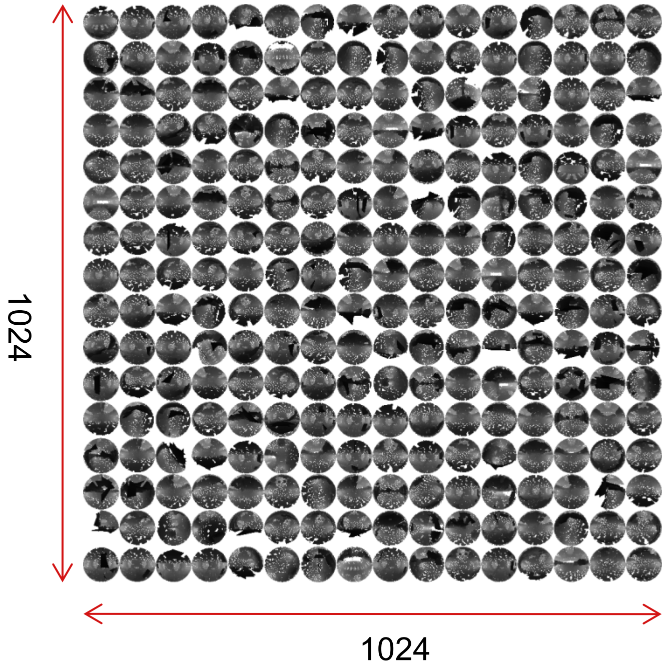
- G-buffer splitting [Segovia et al. 2006]
  - Exploit computation coherency for interleaved sampling



Split a G-buffer into small sub-buffers | Shade each small sub-buffer (coherent computation) | Gather the resulting sub-buffers

INTERACTIVE GI | IMPERFECT SHADOW MAPS [Ritschel et al. 2008; 2011]

- Low-resolution SMs for many lights (e.g., VPLs)
  - Single-pass
  - An arbitrary number of lights
  - Efficient non-linear rasterization
- Point based approximated visibility (= imperfect)
  - Holes
  - Higher bias
  - But fast!



1 render target
1 draw call
64x64 ISMs for 256 VPLs

INTERACTIVE GI | TEMPORAL REPROJECTION

- Reverse reprojection caching [Nehab et al. 2007]
  - Improve indirect illumination quality
  - Temporal anti-aliasing
  - Reduce flickering

Temporal coherence

w/o reprojection

with reprojection

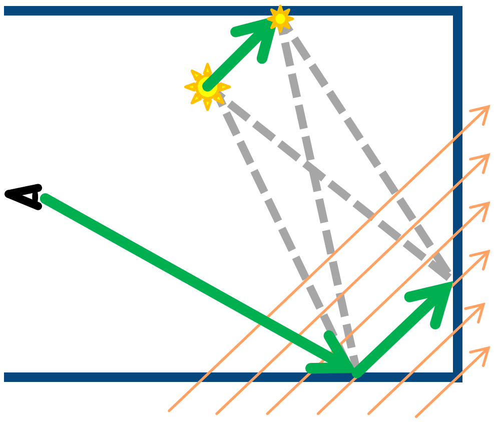
INTERACTIVE GI | FAILURE OF SAMPLING STRATEGY

- Only sample light subpaths (VPLs)
  - Singularities near VPLs
  - Temporal flickering
    - Reduced by temporal reprojection (but insufficient)

Singularities near VPLs

INTERACTIVE GI | OUR BIDIRECTIONAL APPROACH

- Bidirectional path tracing [Lafortune et al. 1993; Veach et al.1994]
  – Robust algorithm for off-line rendering
  – Generating light subpaths & eye subpaths using ray tracing
  – They are connected by shadow rays

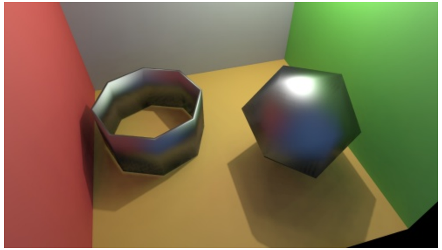


- We employ ***ray-bundles*** for eye subpaths
  – Eye subpaths: G-buffer & ray-bundles
  – Light subpaths: VPLs via RSMs
  – Connecting edges: SMs or ISMs for VPLs

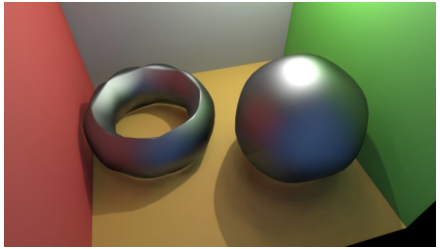
Only Rasterization!

  – Easy to implement
  – Support for GPU tessellation

GPU: Radeon HD 6990



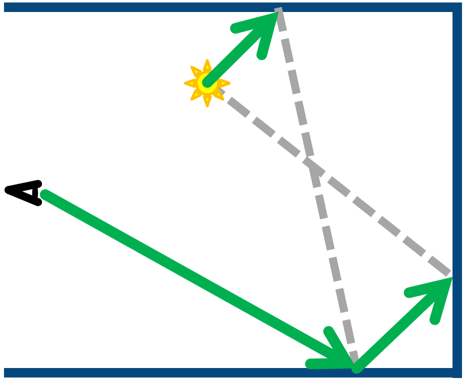
w/o tessellation
29 ms



Phong tessellation
31 ms

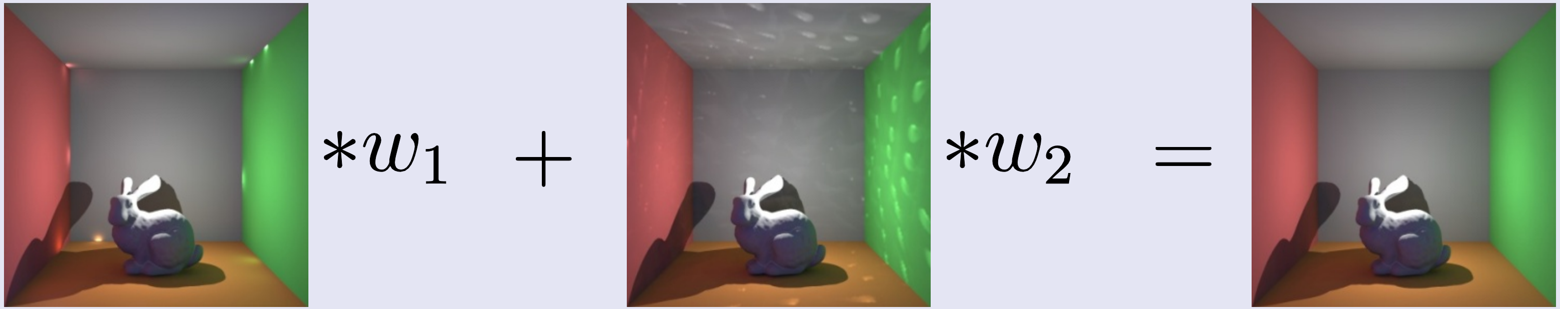
INTERACTIVE GI | MULTIPLE IMPORTANCE SAMPLING [Veach and Guibas1995]

- Combine several paths with a ***weighting function***
- Accurate results with fewer artifacts



Weighting function: $w_i(\bar{\boldsymbol{x}}) = \frac{N_i p_i(\bar{\boldsymbol{x}})}{\sum_j N_j p_j(\bar{\boldsymbol{x}})}$

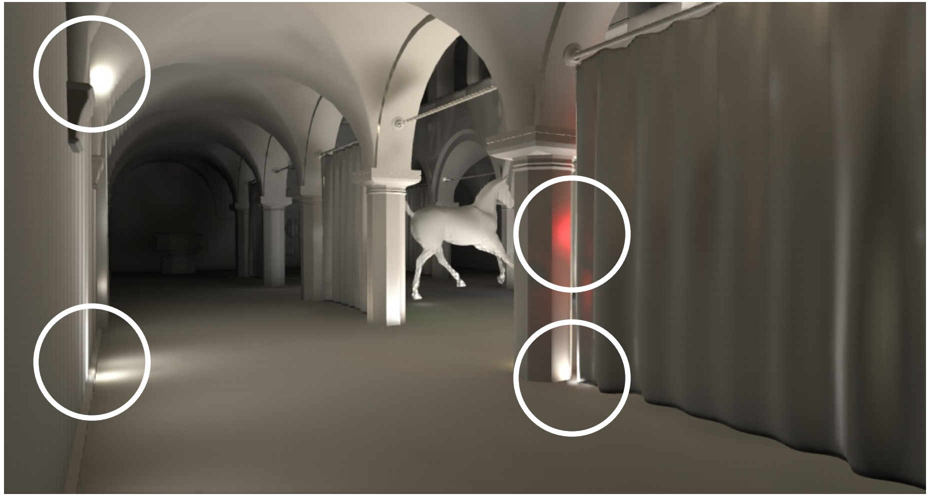
number of samples — PDF



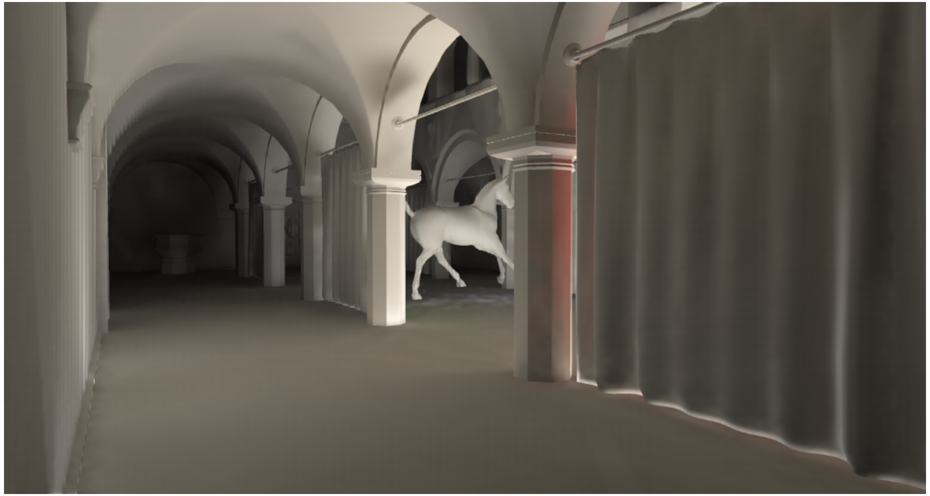
Light path via VPLs

Eye path via ray-bundles

INTERACTIVE GI | RESULTS



Only VPLs
(Spike artifacts and flickering)
32 ms

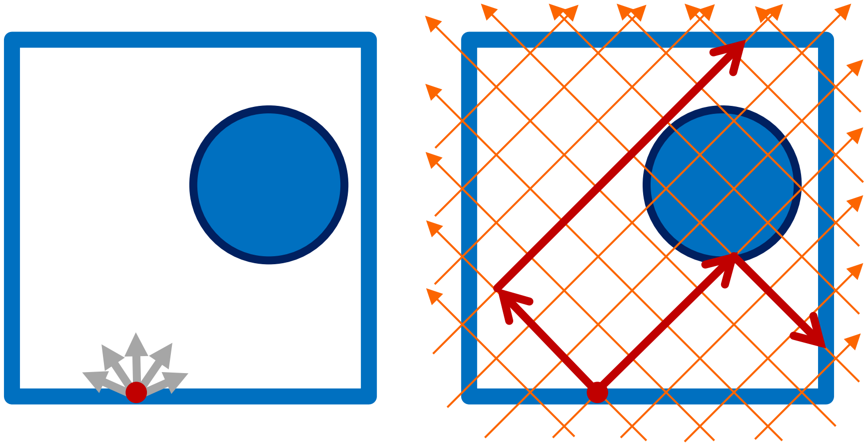


Bidirectional approach
(Artifacts are reduced)
44 ms

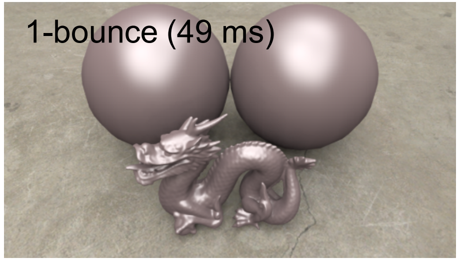
Screen: 1920x1024 resolutions, 2x2 supersampling
ISMs: 64x64 resolution, ***256 VPLs***, 16384 points
Ray-bundles: 256x256 resolution, ***16 samples***
GPU: Radeon HD 6990

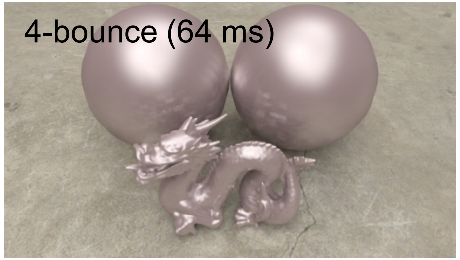
INTERACTIVE GI | MULTI-BOUNCE RENDERING

- All ray-bundles are created preliminary to solve the light transport problem
- Randomly select a single ray-bundle and reuse it for the next bounce
- ☺ No need for additional visibility tests for an arbitrary number of interreflections
- ☹ Memory consuming, only small scenes



GPU: Radeon HD 6990

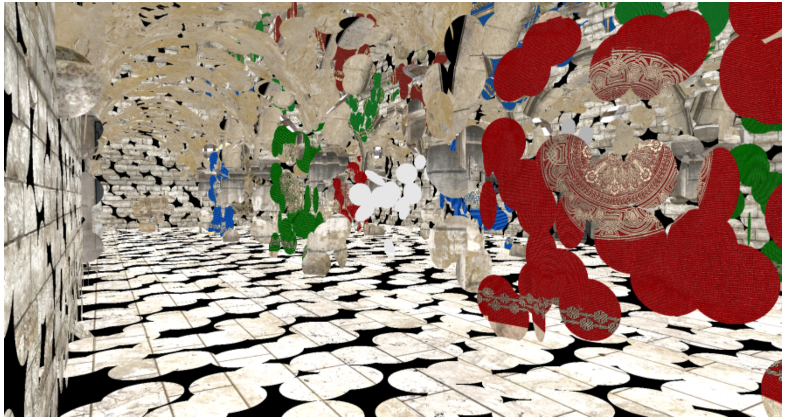




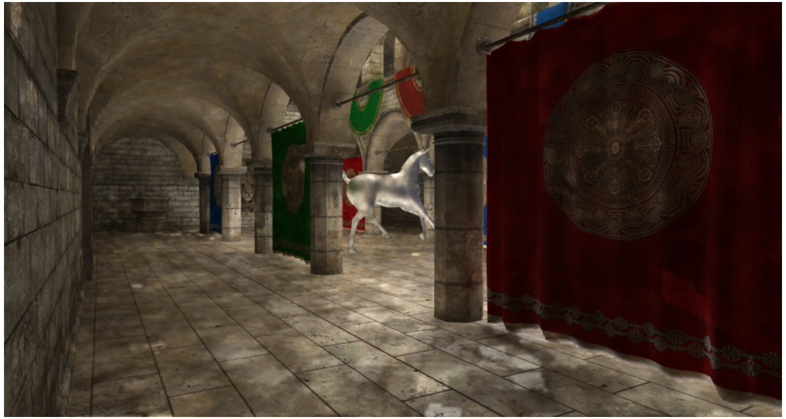
INTERACTIVE GI | IMPERFECT RAY-BUNDLE TRACING

- Point based approximation of scene geometry
- Inherit pros & cons of ISMs

GPU: Radeon HD 6990



Direct visualization of the point splats



Imperfect ray-bundle tracing
(256 directions, 3-bounce eye paths)
89 ms

SUMMARY

- Modern GPUs enable simple & fast ray-bundle tracing
- GI for tessellated scenes is easily computed with ray-bundles
- Bidirectional sampling with ray-bundles reduces the error for interactive GI

GI prebaking
for static scenes

Robust interactive GI
for dynamic scenes

THANK YOU

REFERENCES

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective shadow maps. In Proc. of I3D 2005, 203-213.

ENDERTON, E., SINTORN, E., SHIRLEY, P., LUEBKE, D. 2010. Stochastic Transparency. In Proc. of I3D 2010, 157-164.

HACHISUKA, T. 2005. High-quality global illumination rendering using rasterization. In GPU Gems 2. Addison-Wesley Professional, ch. 38, 615-634.

HERMES, J., HENRICH, N., GROSCH, T., AND MUELLER, S. 2010. Global illumination using parallel global ray-bundles. In Vision, Modeling and Visualization.

JENSEN, H. W. 1996. Global illumination using photon maps. In Proc. of the Eurographics Workshop on Rendering Techniques' 96, 21-30.

KAJIYA, J. T. 1986. The rendering equation. SIGGRAPH Comput. Graph. 20, 143-150.

KELLER, A. 1997. Instant radiosity. In Proc. of ACM SIGGRAPH' 97, 49-56.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bi-directional path tracing. In Proc. of Compugraphics' 93, 145-153.

LLOYD, B., TUFT, D., YOON, S.-E., AND MANOCHA, D. 2006. Warping and partitioning for low error shadow maps. In Proc. of EGSR 2006, 215-226.

NEHAB, D., SANDER, P. V., LAWRENCE, J., TATARCHUK, N., AND ISIDORO, J. R. 2007. Accelerating real-time shading with reverse reprojection caching. In Proc. of Graphics Hardware 2007, 25-35.

NIESSNER, M., SCHAFER, H., STAMMINGER, M. 2010. Fast indirect illumination using layered depth images. The Visual Computer, 26, 6-8, 679-686.

RITSCHEL, T., GROSCH, T., KIM, M. H., SEIDEL, H.-P., DACHSBACHER, C., AND KAUTZ, J. 2008. Imperfect shadow maps for efficient computation of indirect illumination. ACM Trans. Graph. 27, 129:1-129:8.

RITSCHEL, T., EISEMANN, E., HA, I., KIM, J. D., AND SEIDEL, H.-P. 2011. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. Comput. Graph. Forum 30, 2258-2269.

ROSEN, P,. 2012. Rectilinear texture warping for fast adaptive shadow mapping. In Proc. of I3D 2012, 151-158.

SEGOVIA., B., IEHL., J.-C., MITANCHEY., R., AND P´E ROCHE., B. 2006. Non-interleaved deferred shading of interleaved sample patterns. In Proc. of Graphics Hardware 2006, 53-60.

SMITS, B., SHIRLEY, P., AND STARK, M. M. 2000. Direct ray tracing of smoothed and displacement mapped triangles. Tech. rep.

THOMSEN, A., AND NIELSEN, K,-H,. 2011. Approximate Radiosity Using Stochastic Depth Buffering. Journal of Graphics, GPU, and Game Tools, 15, 4, 225-234.

THIBIEROZ, N. 2011. Order-independent transparency using per pixel linked lists. In GPU Pro 2. AK Peters, ch. VII, 2, 409-431.

VEACH, E., AND GUIBAS, L. J. 1994. Bidirectional estimators for light transport. In Proc. of Eurographics Rendering Workshop, 147-162.

VEACH, E., AND GUIBAS, L. J. 1995. Optimally combining sampling techniques for monte carlo rendering. In Proc. of SIGGRAPH'95, 419-428.

YANG, J. C., HENSLEY, J., GRUN, H., AND THIBIEROZ, N. 2010. Real-time concurrent linked list construction on the gpu. Comput. Graph. Forum 29, 1297-1304.

The polygon models are courtesy of Marko Dabrovic (http://hdri.cgtechniques.com/~sponza/), Frank Meinl (http://www.crytek.com/cryengine/cryengine3/downloads), Robert W. Sumner, Jovan Popovic (http://people.csail.mit.edu/sumner/research/deftransfer/data.html), and the Stanford 3D Scanning Repository (http://graphics.stanford.edu/data/3Dscanrep/).

Disclaimer & Attribution