SCREEN-SPACE BIAS COMPENSATION

FOR INTERACTIVE HIGH-QUALITY RENDERING WITH VIRTUAL POINT LIGHTS

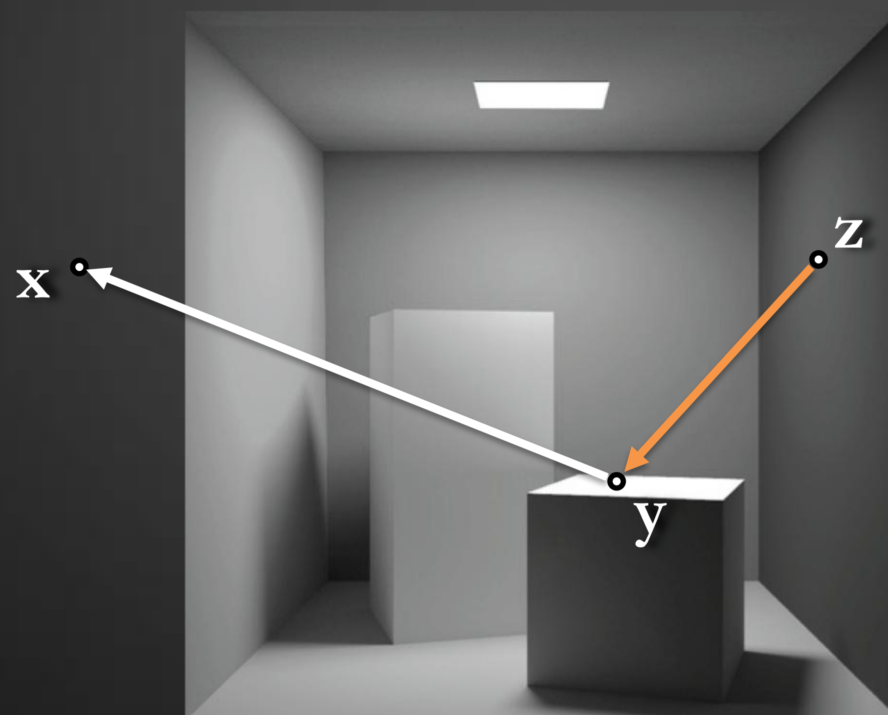
Jan Novak, Thomas Engelhardt, and Carsten Dachsbacher

Computer Graphics Group
Karlsruhe Institute of Technology

Light Transport

- Rendering equation:

$$L(\mathbf{x}\leftarrow\mathbf{y}) = \underbrace{L_e(\mathbf{x}\leftarrow\mathbf{y})}_{\text{Emitted light}} + \underbrace{\int_A f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z})\,G(\mathbf{y}\leftrightarrow\mathbf{z})\,V(\mathbf{y}\leftrightarrow\mathbf{z})\,L(\mathbf{y}\leftarrow\mathbf{z})\,dA}_{\text{Reflected light}}$$



Light Transport

- Rendering equation:

$$L(\mathbf{x}\leftarrow\mathbf{y}) = \underbrace{L_e(\mathbf{x}\leftarrow\mathbf{y})}_{\text{Emitted light}} + \underbrace{\int_A f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z}) G(\mathbf{y}\leftrightarrow\mathbf{z}) V(\mathbf{y}\leftrightarrow\mathbf{z}) L(\mathbf{y}\leftarrow\mathbf{z})\, dA}_{\text{Reflected light}}$$

- Operator notation [Arvo et al. 1994]:

$$(\mathbf{T}L)(\mathbf{x}\leftarrow\mathbf{y}) = \int_A f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z}) G(\mathbf{y}\leftrightarrow\mathbf{z}) V(\mathbf{y}\leftrightarrow\mathbf{z}) L(\mathbf{y}\leftarrow\mathbf{z})\, dA$$

$$L = L_e + \mathbf{T}L$$

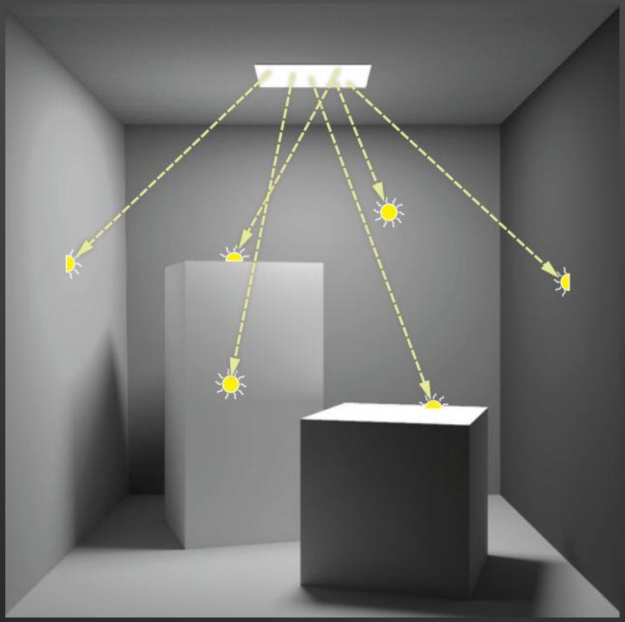
VPL Rendering

- Based on Instant Radiosity [*Keller 1997*]
- Indirect illumination approximated by **Virtual Point Lights (VPLs)** $\hat{L}$

VPL Rendering

- Based on Instant Radiosity [*Keller 1997*]
- Indirect illumination approximated by **Virtual Point Lights (VPLs)** $\hat{L}$

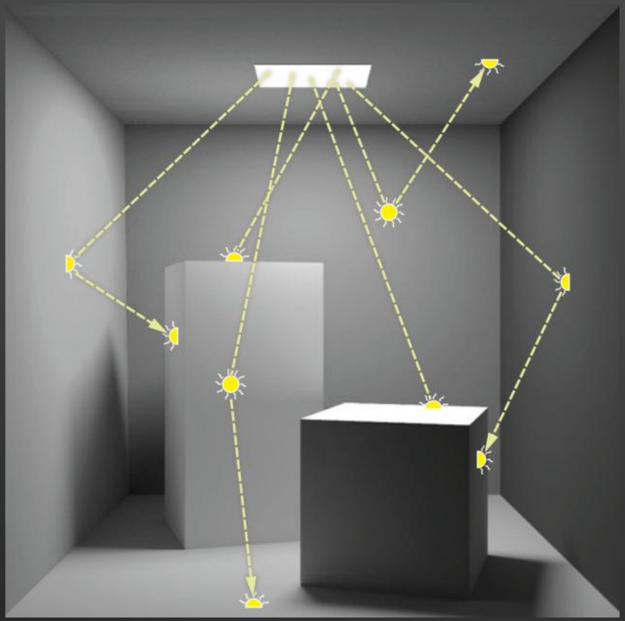
Distribution of VPLs



VPL Rendering

- Based on Instant Radiosity [*Keller 1997*]
- Indirect illumination approximated by **Virtual Point Lights (VPLs)** $\hat{L}$

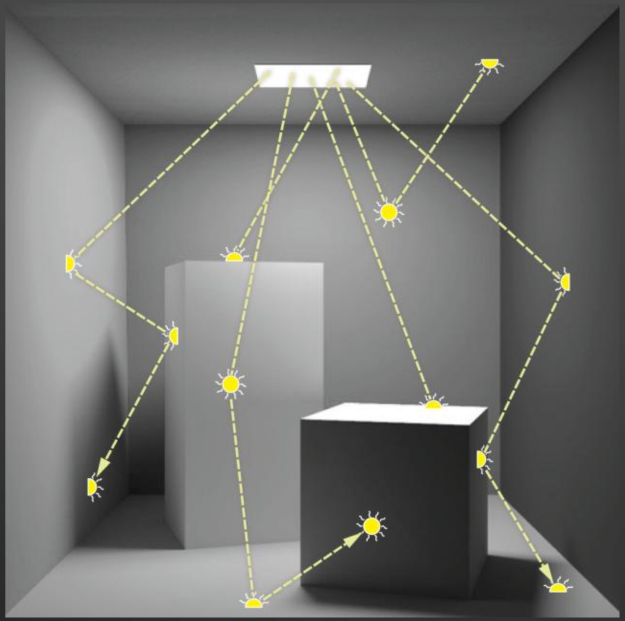
Distribution of VPLs



VPL Rendering

- Based on Instant Radiosity [*Keller 1997*]
- Indirect illumination approximated by **Virtual Point Lights (VPLs)** $\hat{L}$

Distribution of VPLs



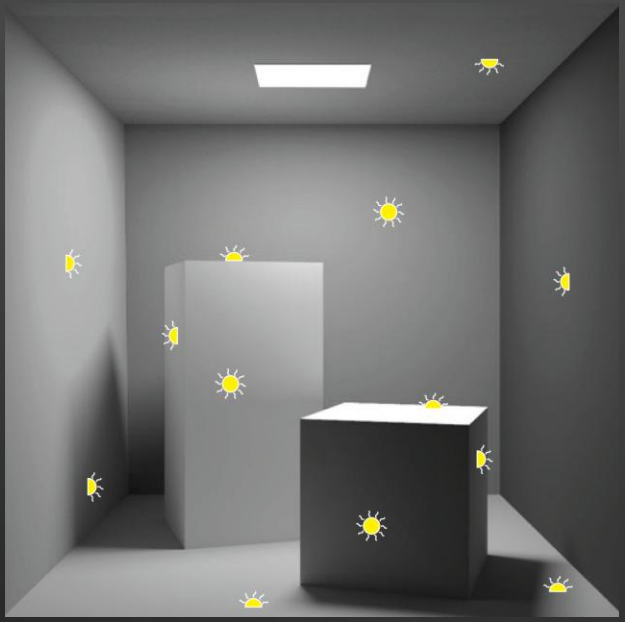
VPL Rendering

- Based on Instant Radiosity [*Keller 1997*]
- Indirect illumination approximated by **Virtual Point Lights (VPLs)** $\hat{L}$

$$L = L_e + \mathbf{T}L$$

$$L = L_e + \mathbf{T}L_e + \mathbf{T}\hat{L}$$

Distribution of VPLs

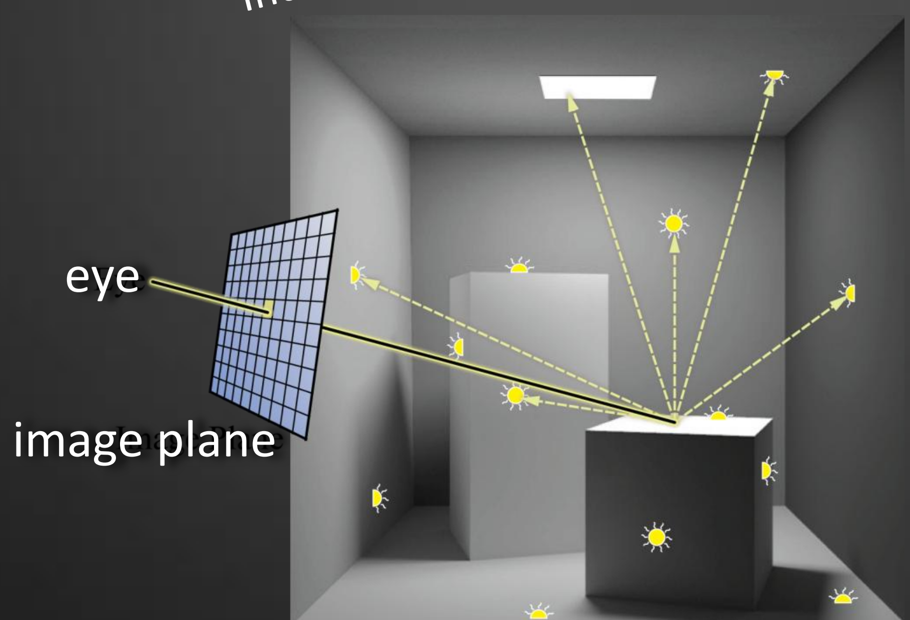


VPL Rendering

- Based on Instant Radiosity [*Keller 1997*]
- Indirect illumination approximated by **Virtual Point Lights (VPLs)** $\hat{L}$

$$L = L_e + \mathbf{T}L$$

$$L = \underbrace{L_e}_{\text{Direct emission}} + \underbrace{\mathbf{T}L_e}_{\text{Direct illumination}} + \underbrace{\mathbf{T}\hat{L}}_{\text{Indirect illumination}}$$

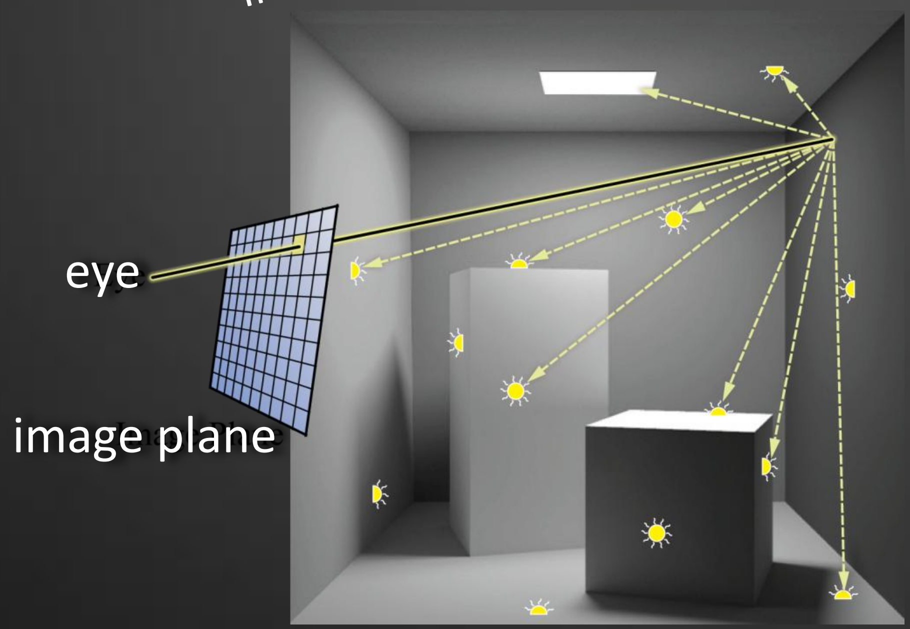


VPL Rendering

- Based on Instant Radiosity [*Keller 1997*]
- Indirect illumination approximated by **Virtual Point Lights (VPLs)** $\hat{L}$

$$L = L_e + \mathbf{T}L$$

$$L = \underbrace{L_e}_{\text{Direct emission}} + \underbrace{\mathbf{T}L_e}_{\text{Direct illumination}} + \underbrace{\mathbf{T}\hat{L}}_{\text{Indirect illumination}}$$



VPL Rendering – Singularities

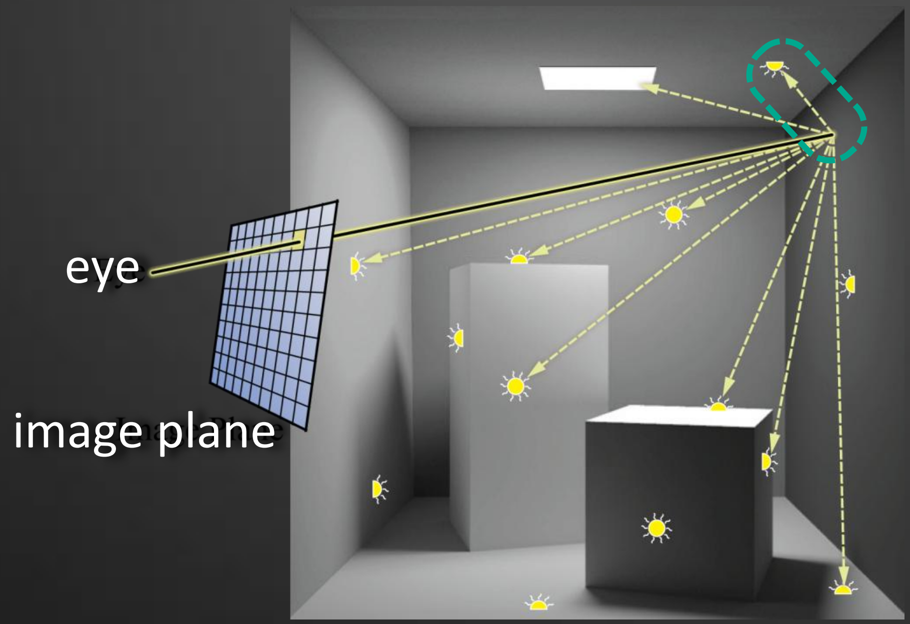
$$L = L_e + \mathbf{T}L_e + \mathbf{T}\hat{L}$$

Transport operator:

$$(\mathbf{T}\hat{L})(\mathbf{x}\leftarrow\mathbf{y}) = \sum_{i=1}^{N} f_r(\mathbf{x}\leftarrow\mathbf{y}\leftarrow\mathbf{z}_i)\, G(\mathbf{y}\leftrightarrow\mathbf{z}_i)\, V(\mathbf{y}\leftrightarrow\mathbf{z}_i)\, \hat{L}(\mathbf{y}\leftarrow\mathbf{z}_i)$$

Geometry term:

$$G(\mathbf{y}\leftrightarrow\mathbf{z}_i) = \frac{\cos^+(\theta_{\mathbf{y}})\cos^+(\theta_{\mathbf{z}_i})}{\left\|\mathbf{y}-\mathbf{z}_i\right\|^2}$$



Artifacts - Splotches

scene: Autodesk | *rendering*: Edgar Velázquez-Armendáriz

Both are mathematically correct, but…

Artifacts - Splotches

Reference (slow) rendering

Fast rendering with less VPLs

Artifacts - Splotches

Reference (slow) rendering

Fast rendering with less VPLs

Clamping VPLs' contribution

Clamp the contribution of nearby VPLs
by bounding the Geometry term.

Artifacts - Splotches

Reference (slow) rendering

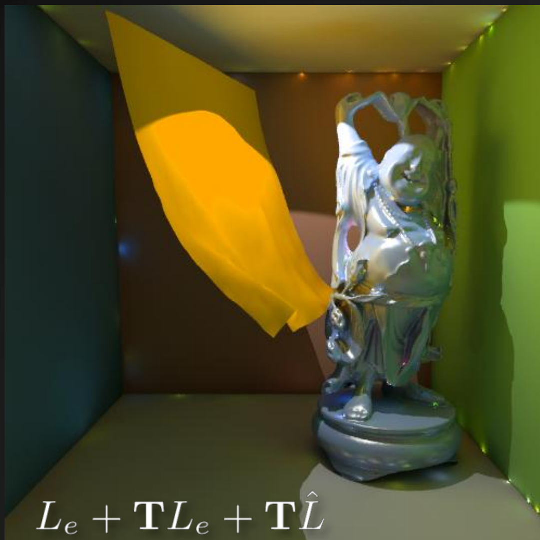
DIFFERENCE

Clamping VPLs' contribution

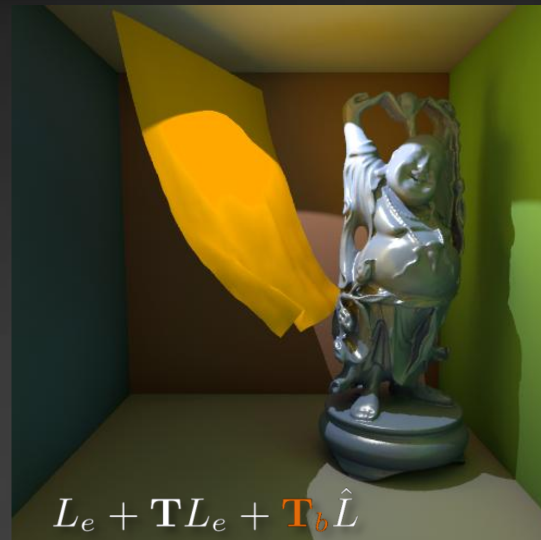
Clamping removes short distance light transport.

How do we restore the missing energy?

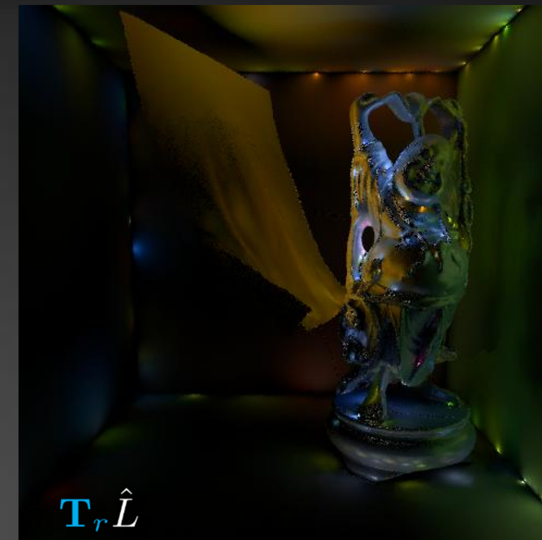
Bounded and Residual Light Transport



-



=



Complete LT: $L_e + \mathbf{T}L_e + \mathbf{T}\hat{L}$

$$\mathbf{T}\hat{L} = \sum_{i=1}^{N} f_r \, G \, V \, \hat{L}$$

Bounded indirect LT: $L_e + \mathbf{T}L_e + \mathbf{T}_b\hat{L}$

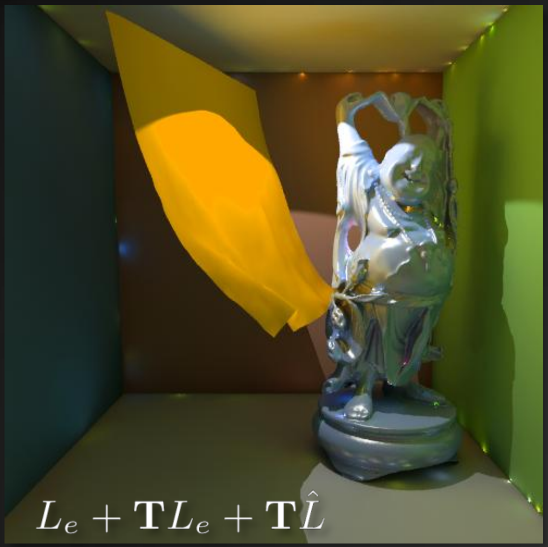
$$\mathbf{T}_b\hat{L} = \sum_{i=1}^{N} f_r \, \min(G, b) \, V \, \hat{L}$$

Residual indirect LT: $\mathbf{T}_r\hat{L}$

$$\mathbf{T}_r\hat{L} = \sum_{i=1}^{N} f_r \, \max(G - b, 0) \, V \, \hat{L}$$

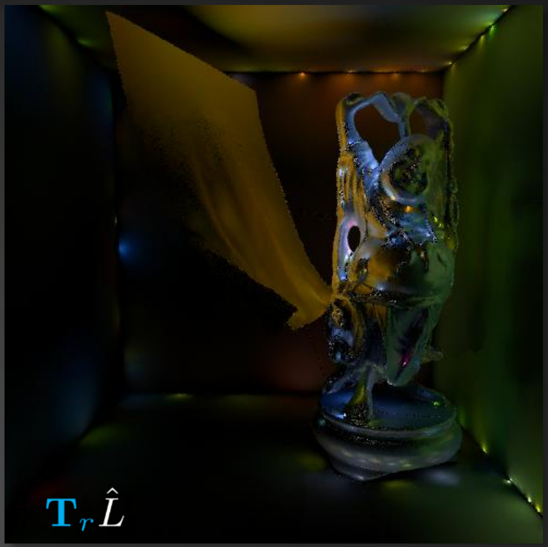
b: user-defined bound

Bounded and Residual Light Transport



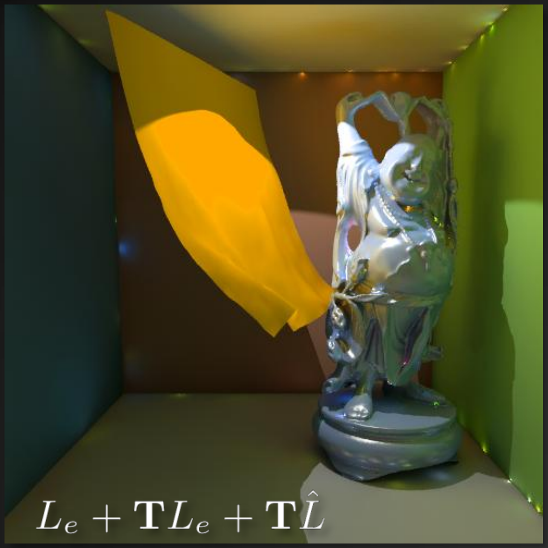
-

=



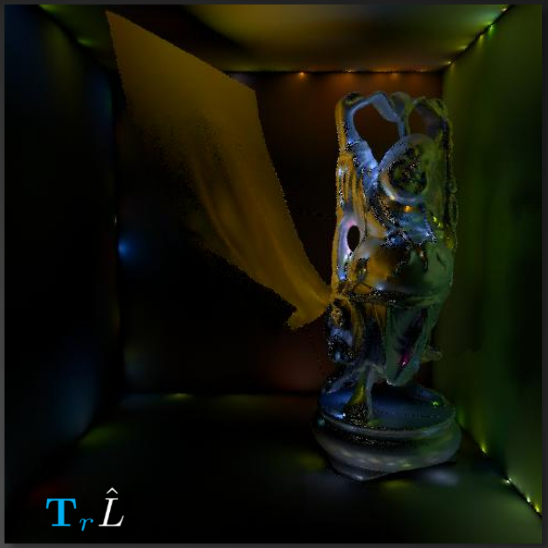
$$\mathbf{T} = \mathbf{T}_b + \mathbf{T}_r$$

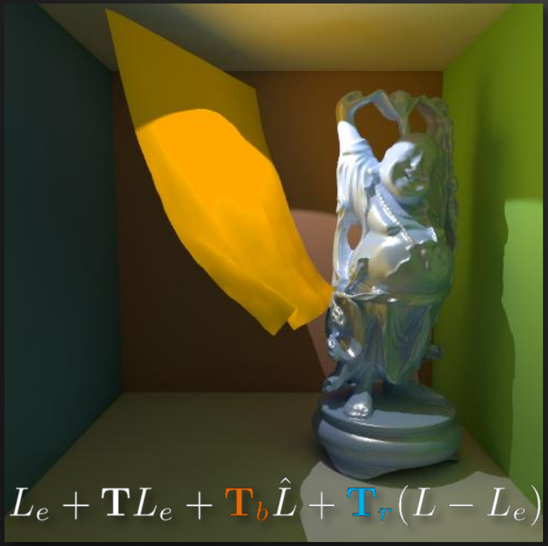
Bounded and Residual Light Transport



-

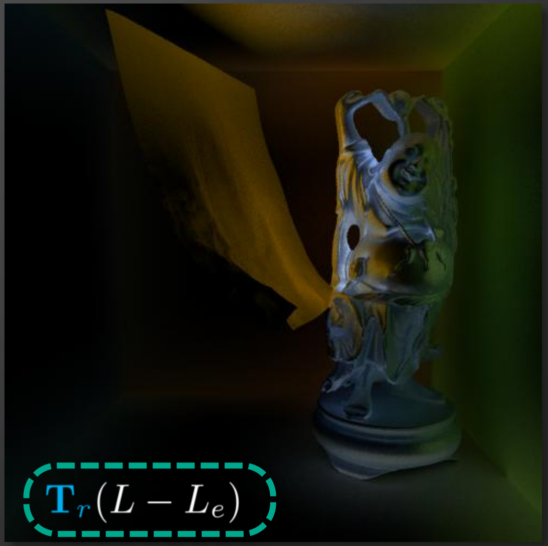
=





=

+



Residual Light Transport – Previous Work

Illumination in the Presence of Weak Singularities [*Kollig and Keller 2004*]

- Bias compensation via tracing additional rays
- Indirect illumination for residual transport computed via path tracing
- Unbiased but computational very intensive (~hours)
- Infeasible for interactive applications

$$L = L_e + \mathbf{T} L_e + \mathbf{T}_b \hat{L} + \mathbf{T}_r \hat{L}$$

$$L = L_e + \mathbf{T} L_e + \mathbf{T}_b \hat{L} + \mathbf{T}_r (L - L_e)$$

Computed using path tracing

Residual Light Transport

Our approach:

- Motivation:
  - Restore energy, remove bias
  - Interactive frame-rates
- Solution:
  - Re-use the existing (clamped) solution
  - Iteratively apply the residual transport

$$L = L_e + \mathbf{T} L_e + \mathbf{T}_b \hat{L} + \mathbf{T}_r (L - L_e)$$

$$L = L_e + \sum_{i=0}^{\infty} \underbrace{\mathbf{T}_r{}^i}_{\text{Apply iteratively}} \underbrace{(\mathbf{T} L_e + \mathbf{T}_b \hat{L})}_{\text{Compute once}}$$

Design choice: compute and apply in screen-space

Algorithm Overview

- Precomputation:
  1. Distribute VPLs (CPU)
  2. Create an Imperfect Shadow Map [*Ritschel et al. 2008*] for each VPL

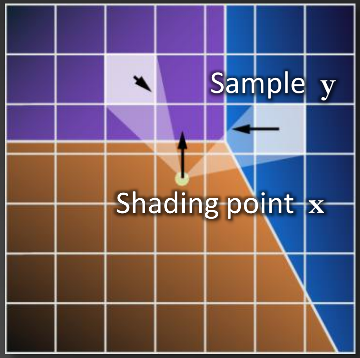
- Rendering:
  1. Render the scene to find visible surfaces
  2. Apply deferred direct and bounded VPL lighting $\mathbf{T}L_e + \mathbf{T}_b\hat{L}$
  3. N-times in screen-space:

     Compute residual transport and add it to the image

$$\sum_{i=0}^{\infty} \mathbf{T}_r{}^i(\mathbf{T}L_e + \mathbf{T}_b\hat{L})$$

Screen-Space Integration

- **FOR EACH** pixel:
  - iterate over neighboring pixels
    - **IF** $G(\mathbf{x}\leftrightarrow\mathbf{y}) > b$:
      - add contribution in “radiosity style”



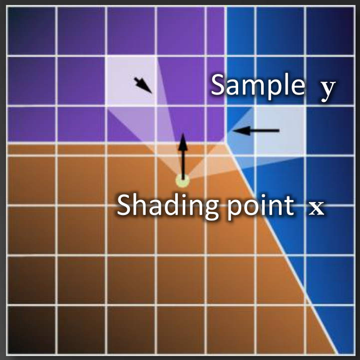
Camera view

Screen-Space Integration

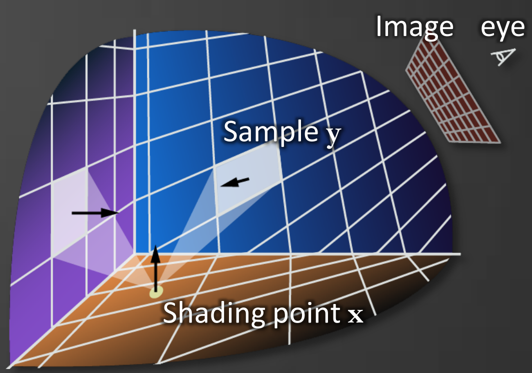
- **FOR EACH** pixel:
  - iterate over neighboring pixels
    - **IF** $G(\mathbf{x} \leftrightarrow \mathbf{y}) > b$:
      - add contribution in “radiosity style”

$$\left(\frac{\cos^{+}(\theta_{\mathbf{x}}) \cos^{+}(\theta_{\mathbf{y}})}{\|\mathbf{x} - \mathbf{y}\|^2} - b \right) A \, f_r \, \tilde{L}$$

G-buffer stores all necessary information.



Camera view

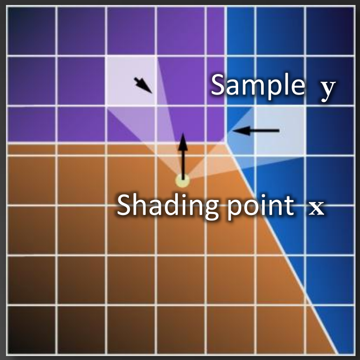


Side view

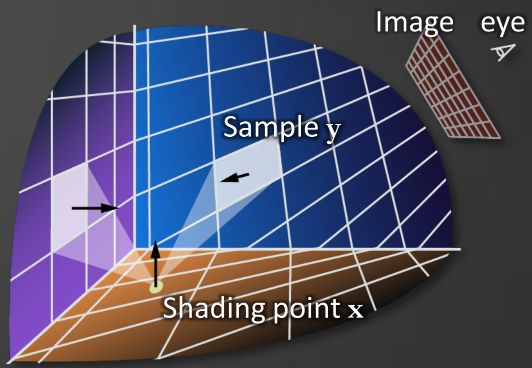
Screen-Space Integration

- **FOR EACH** pixel:
  - iterate over neighboring pixels
    - **IF** $G(\mathbf{x} \leftrightarrow \mathbf{y}) > b$:
      - add contribution in “radiosity style”

We are only interested in the clamped energy.



Camera view



Side view

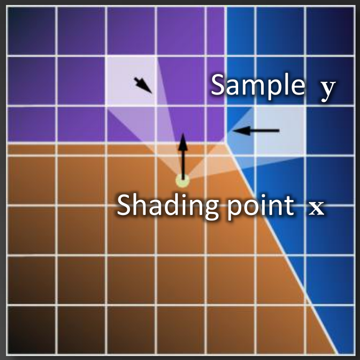
Screen-Space Integration

- **FOR EACH** pixel:
  - iterate over neighboring pixels
  - **IF** $G(\mathbf{x} \leftrightarrow \mathbf{y}) > b$:
    - add contribution in “radiosity style”

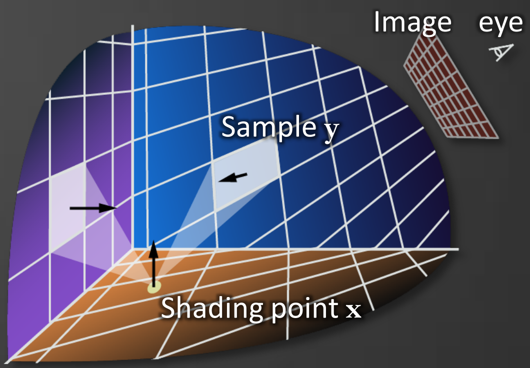
Clamping occurs in a close neighborhood only!
close in world space = close in screen-space

$$G(\mathbf{x} \leftrightarrow \mathbf{y}) = \frac{\cos^+(\theta_\mathbf{x})\cos^+(\theta_\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|^2}$$

We can conservatively estimate a bounding radius and restrict the integration to it.



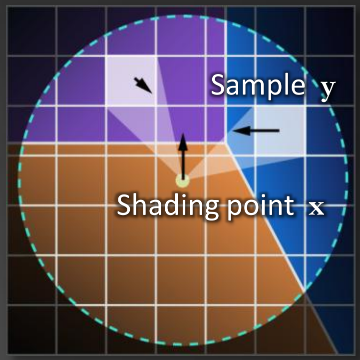
Camera view



Side view

Screen-Space Integration

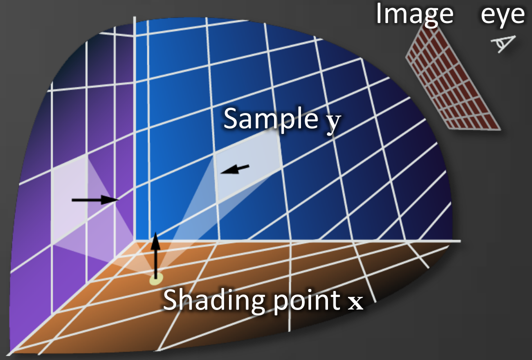
- **FOR EACH** pixel:
  - iterate over neighboring pixels
    - **IF** $G(\mathbf{x}\leftrightarrow\mathbf{y}) > b$:
      - add contribution in “radiosity style”



Camera view

Clamping occurs in a close neighborhood only!
close in world space = close in screen-space

$$G(\mathbf{x}\leftrightarrow\mathbf{y}) = \frac{\cos^+(\theta_\mathbf{x})\cos^+(\theta_\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|^2}$$

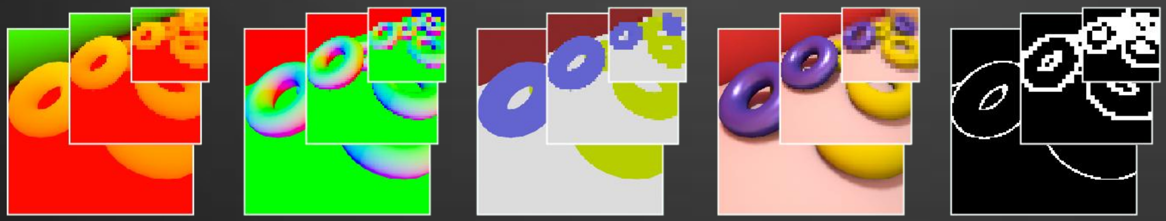


Side view

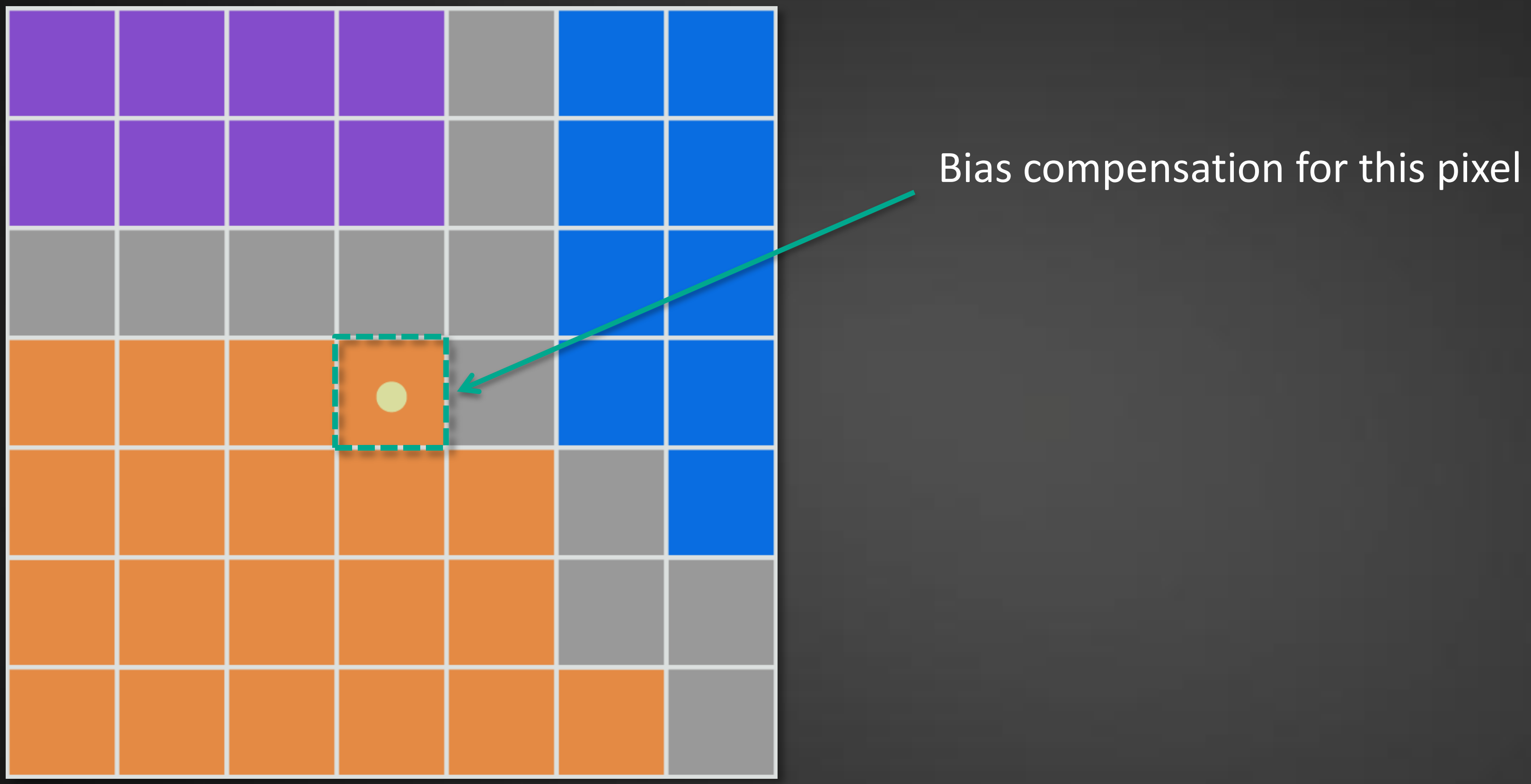
We can conservatively estimate a bounding radius
and restrict the integration to it.

Hierarchical Integration

- Still too many samples (even with the bounding radius)
- Multiresolution top-down integration
  - In the spirit of [*Nichols et al. 2009*]
  - Requires:
    - A mip-map chain of the G-Buffer and bounded illumination
    - Discontinuity buffer

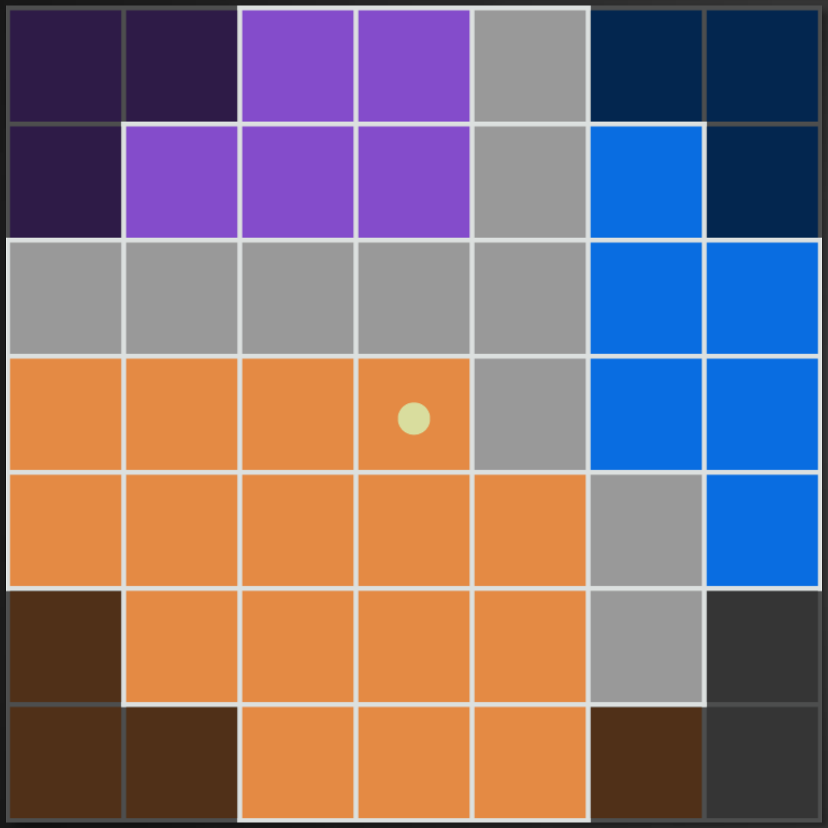


Hierarchical Integration



Coarsest level

Hierarchical Integration



Coarsest level

Limit the integration domain

Hierarchical Integration

Coarsest level

For each pixel

if *possible* add contribution

else refine

1 : non-zero contribution

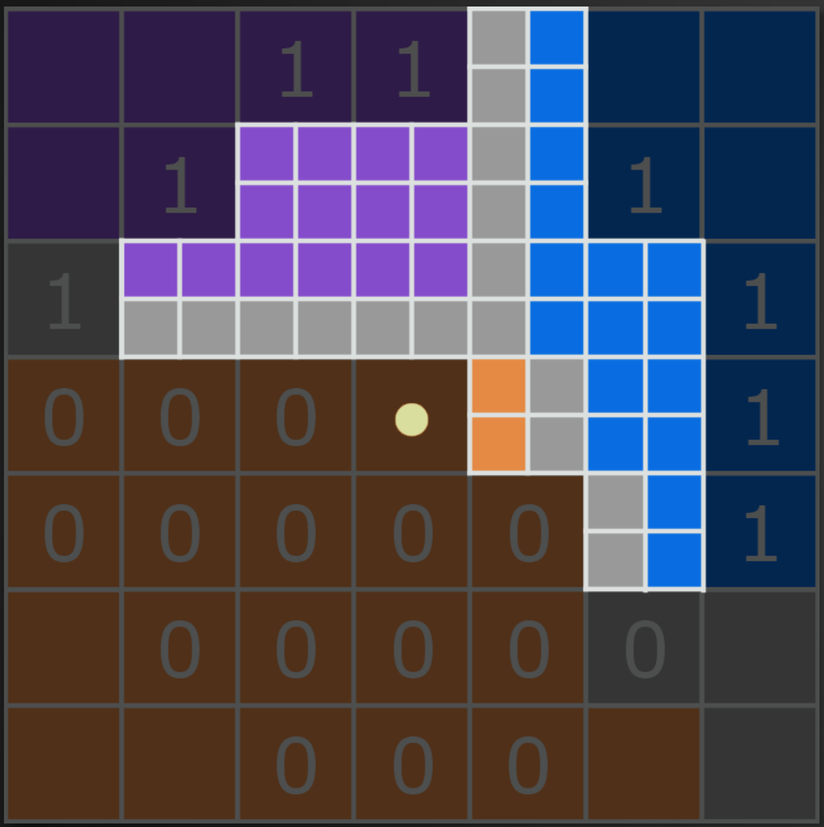
0 : zero contribution

R : must be refined

Refine when:

- G-term too high
- Discontinuity

Hierarchical Integration



Finer level

Refine

Hierarchical Integration

Finer level

For each pixel

if *possible* add contribution
else refine

1 : non-zero contribution
0 : zero contribution
R : must be refined

Refine when:

- G-term too high
- Discontinuity

Hierarchical Integration

Finest level

Refine

Hierarchical Integration

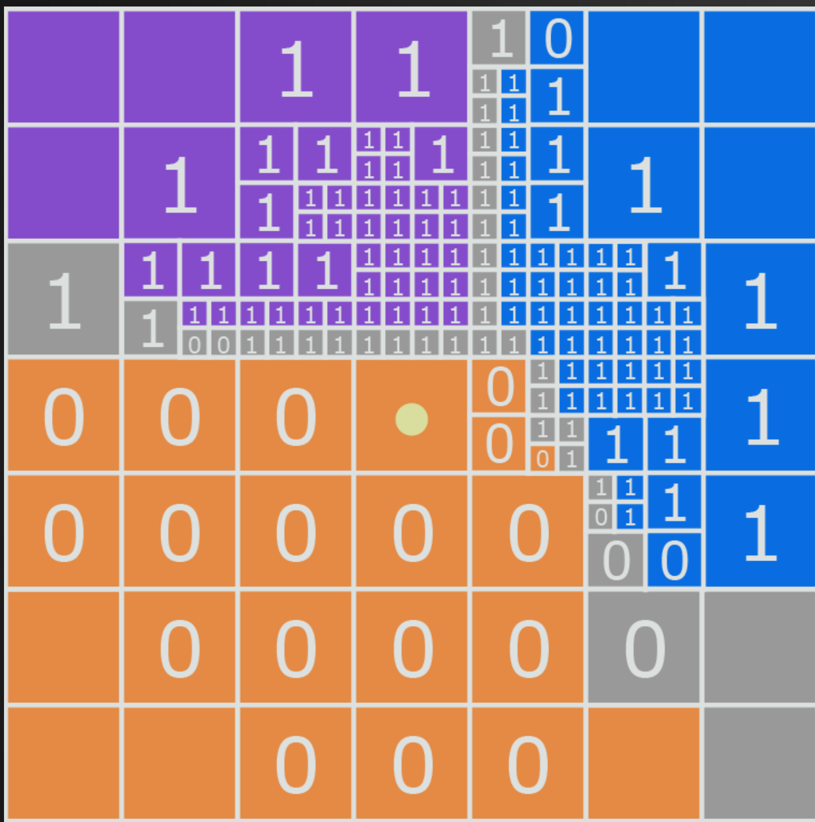
Finest level

For each pixel

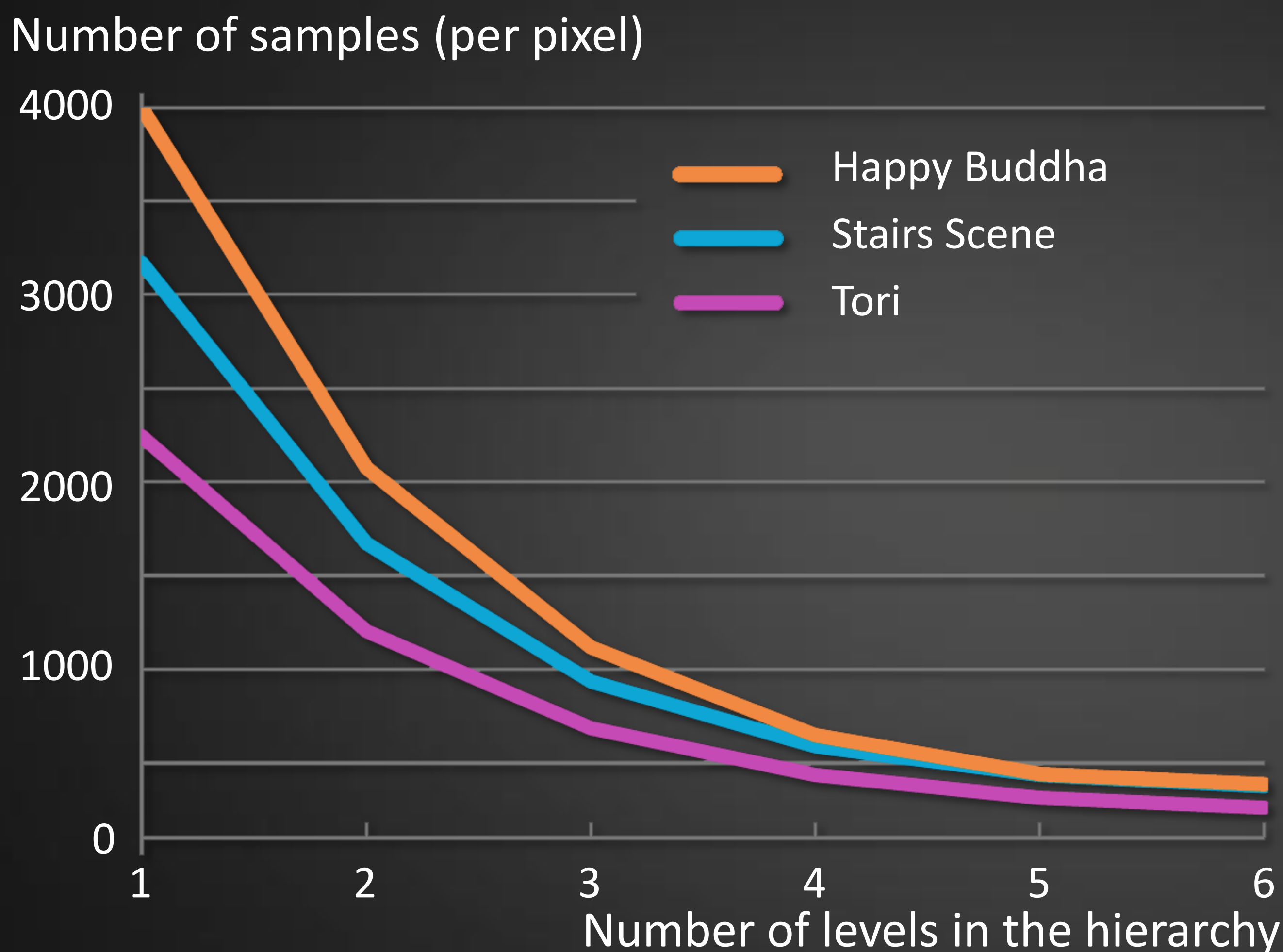
add contribution

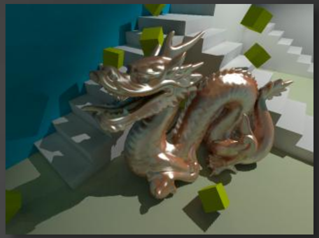
1 : non-zero contribution
0 : zero contribution

Hierarchical Integration – Overview



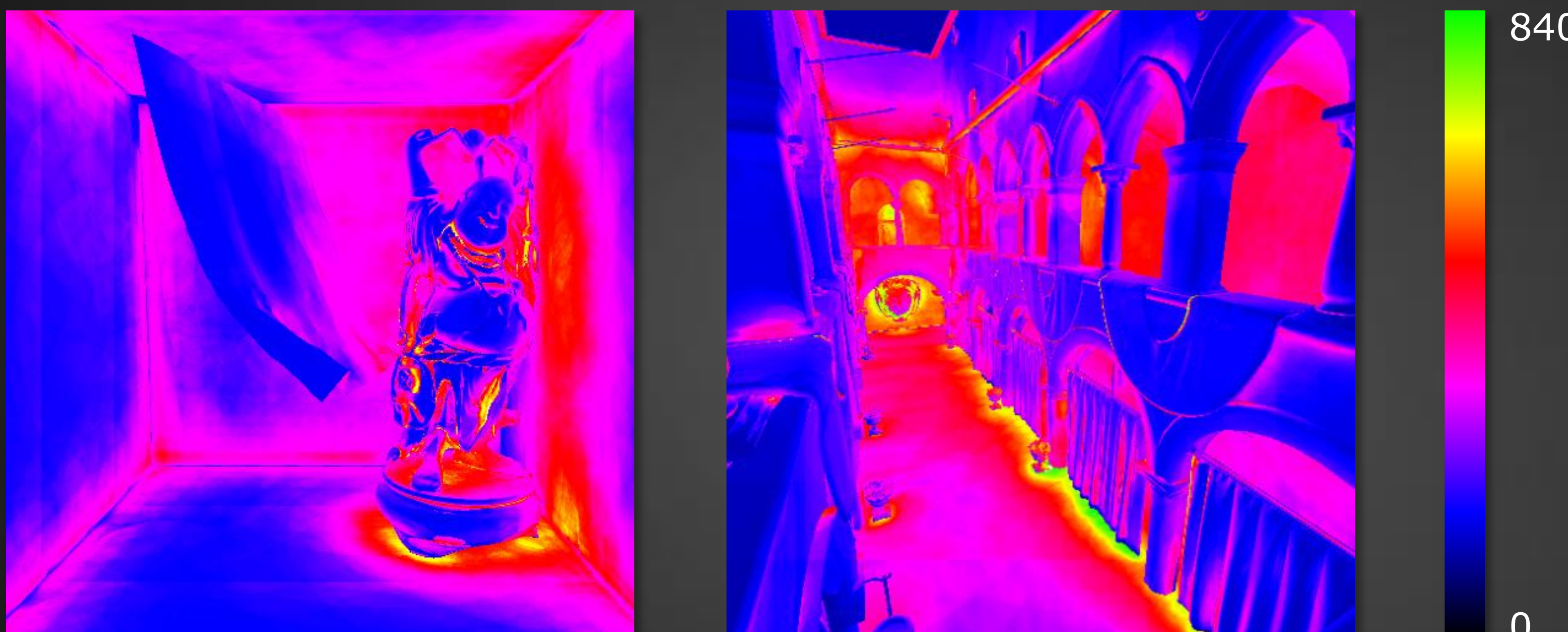
Hierarchical Integration – Evaluation





Hierarchical Integration – Evaluation

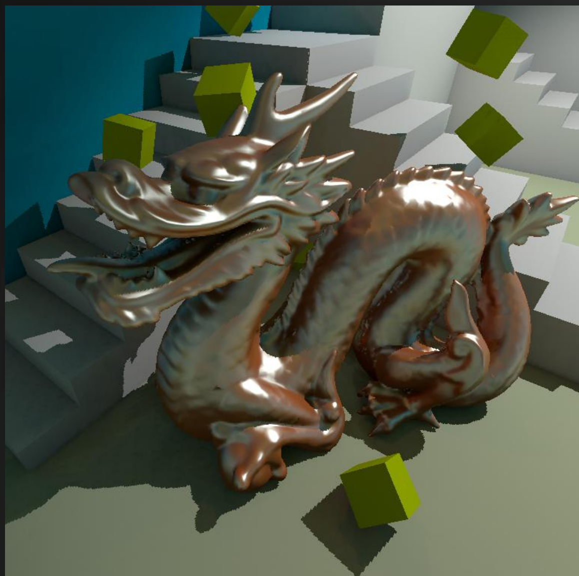
Number of samples (per pixel)



Discontinuities are costly.

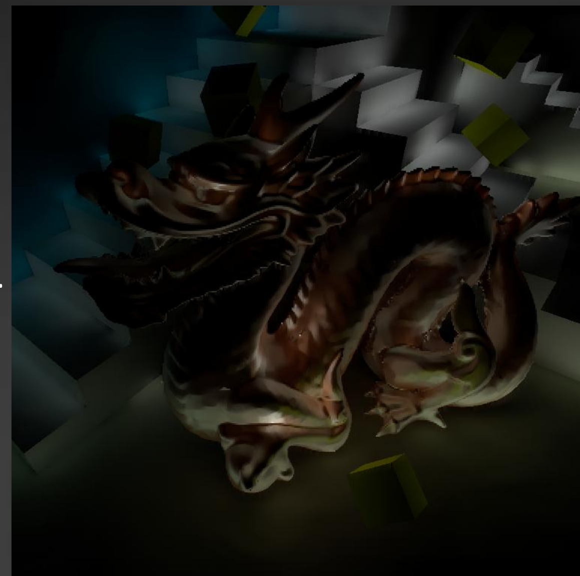
Rendering Results – Dragon

Bounded Light Transport



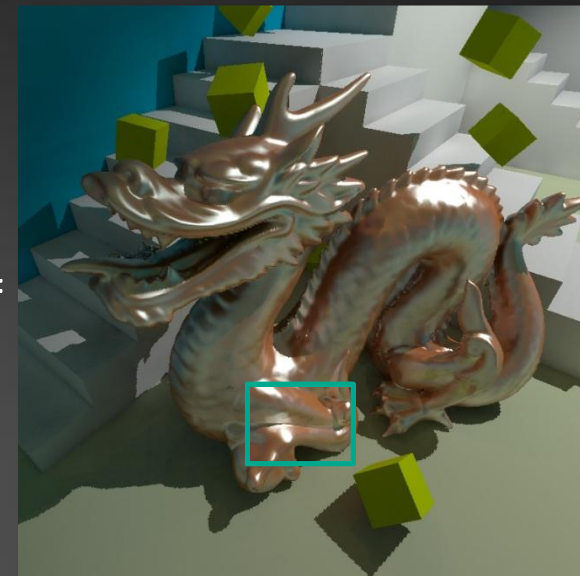
+

Residual Light Transport



=

Final Image

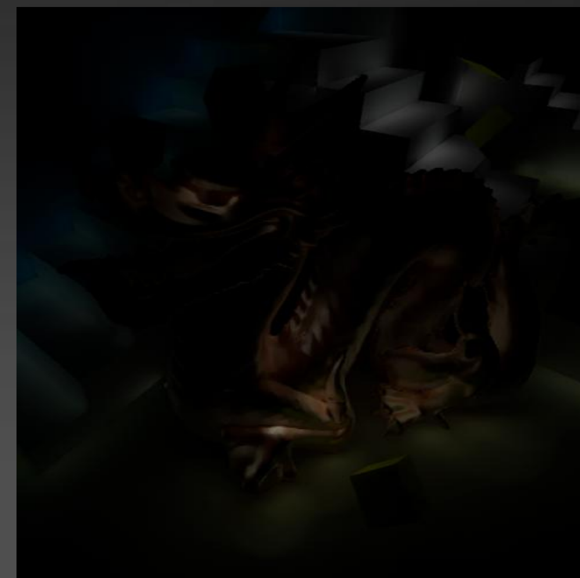


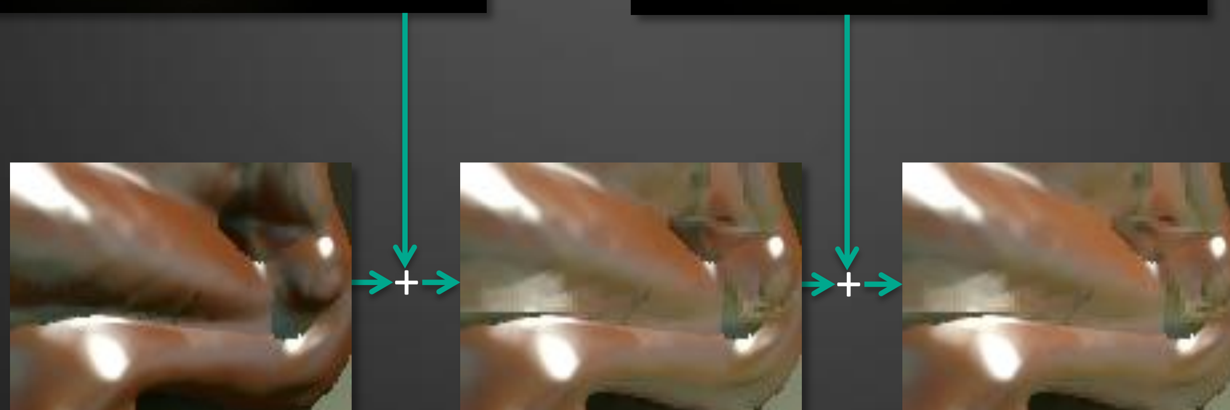
Rendered with: 1024x768 at:	No SSBC 10.3 FPS	1-step SSBC 8.2 FPS	2-step SSBC 6.4 FPS

Rendering Results – Dragon

1st step of SSBC

2nd step of SSBC

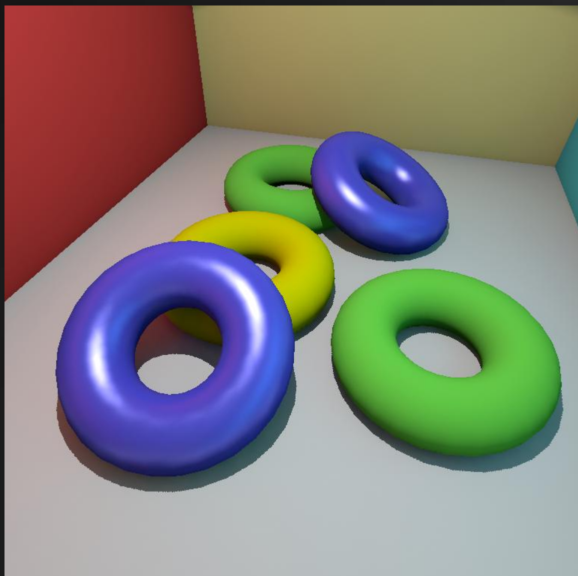




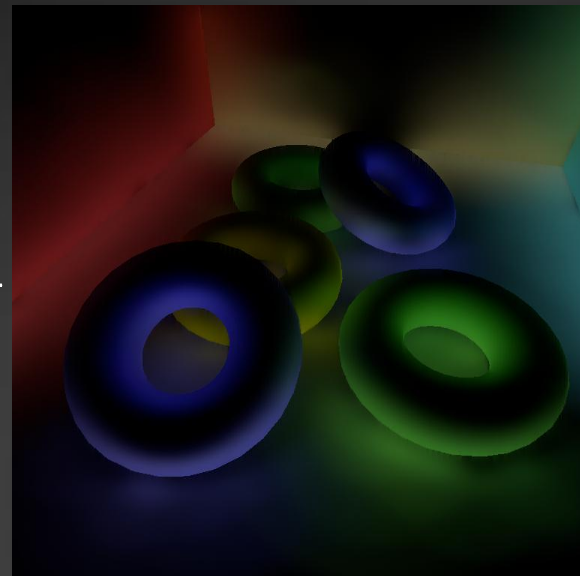
Rendered with: 1024x768 at:	No SSBC 10.3 FPS	1-step SSBC 8.2 FPS	2-step SSBC 6.4 FPS

Rendering Results – Tori

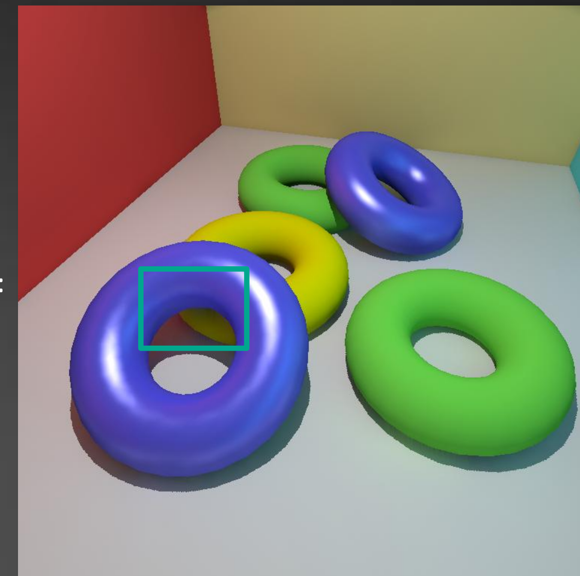
Bounded Light Transport | Residual Light Transport | Final Image



+



=



Rendered with: 1024x768 at:	No SSBC	1-step SSBC	2-step SSBC
	16.4 FPS	12.1 FPS	9.3 FPS

Rendering Results – Crytek Sponza

Bounded Light Transport

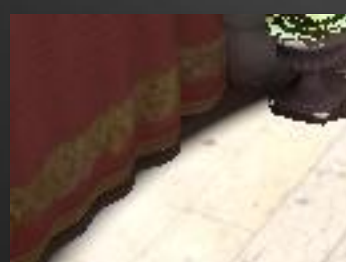
\+

Residual Light Transport

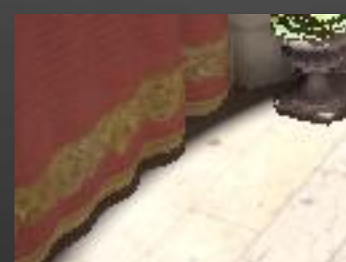
=

Final Image

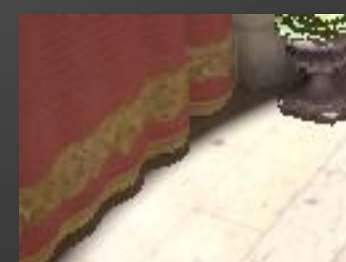
Rendered with:
1024x768 at:



No SSBC
3.8 FPS

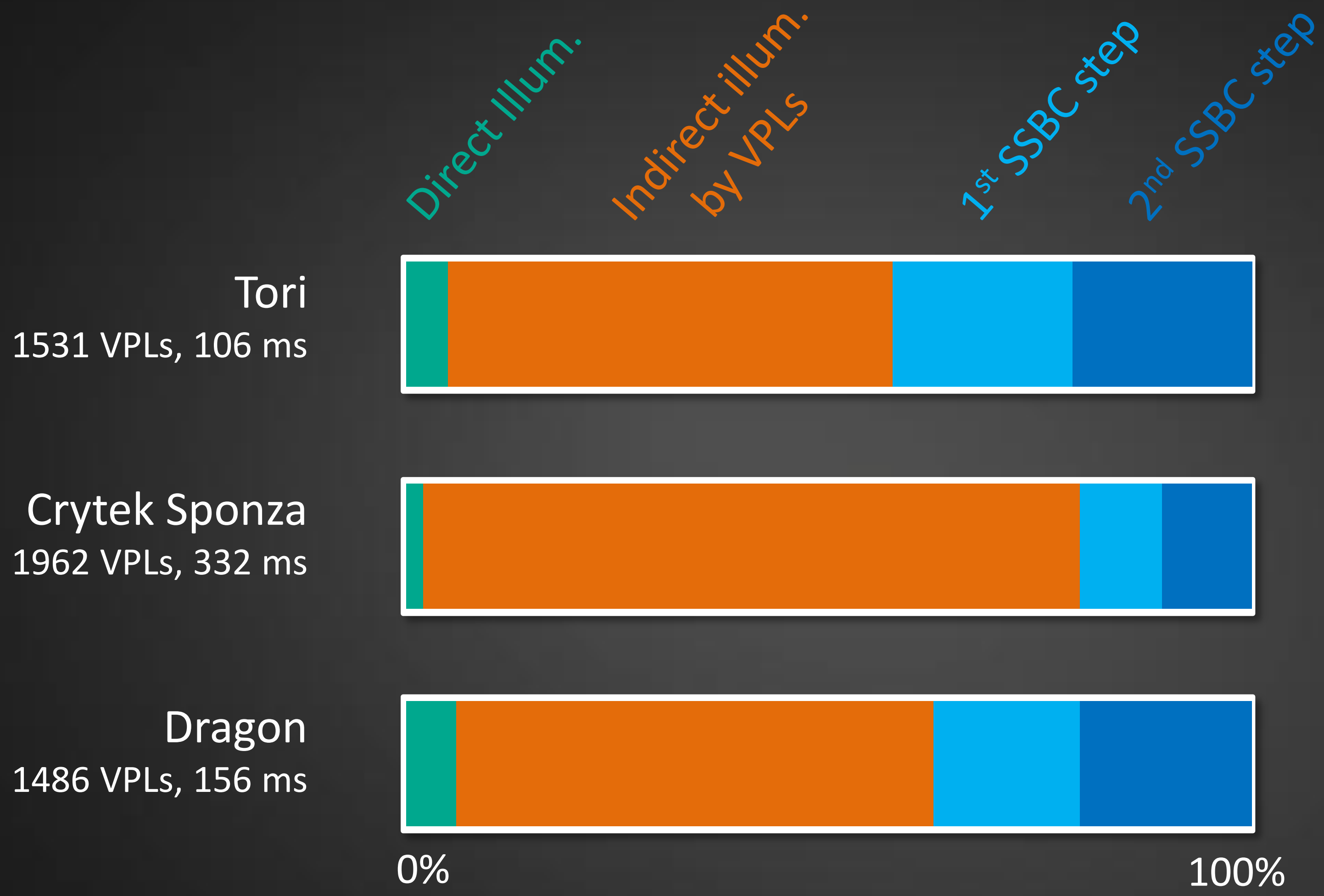


1-step SSBC
3.4 FPS



2-step SSBC
3.0 FPS

SSBC Timings

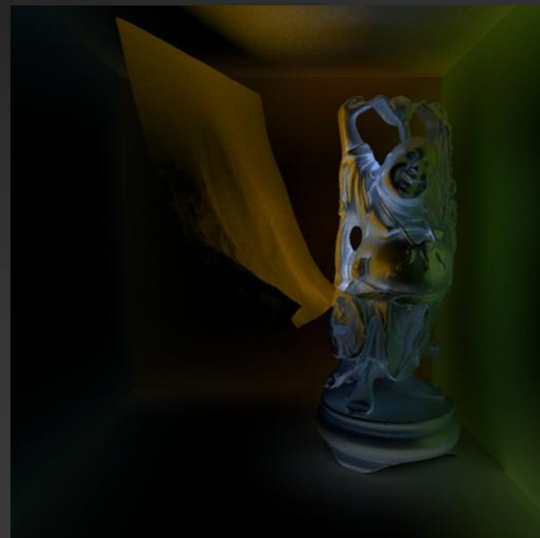


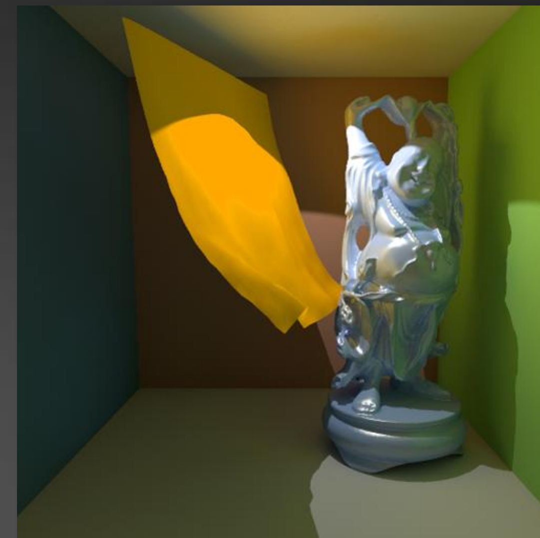
Comparison – [Kollig and Keller 2004] vs. SSBC

Compensation Only | Final Image

Bias Compensation
[Kollig and Keller 2004]

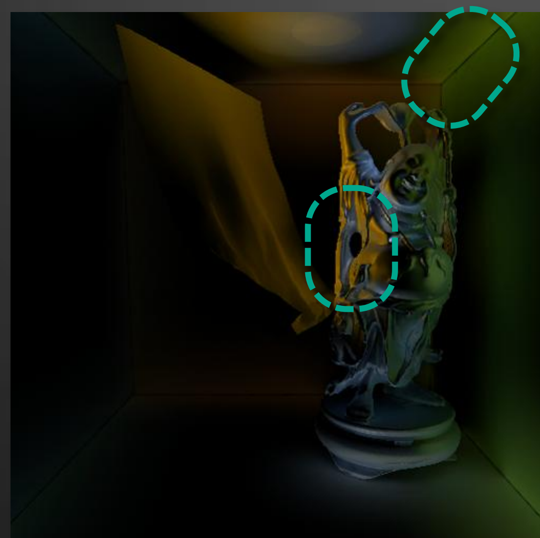
CPU ~ 10.9 hours
(8-core, 4GB RAM)

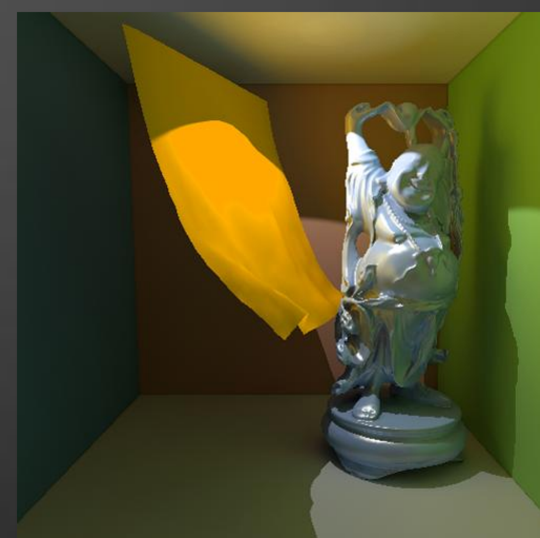




Screen-Space
Bias Compensation
(3 steps)

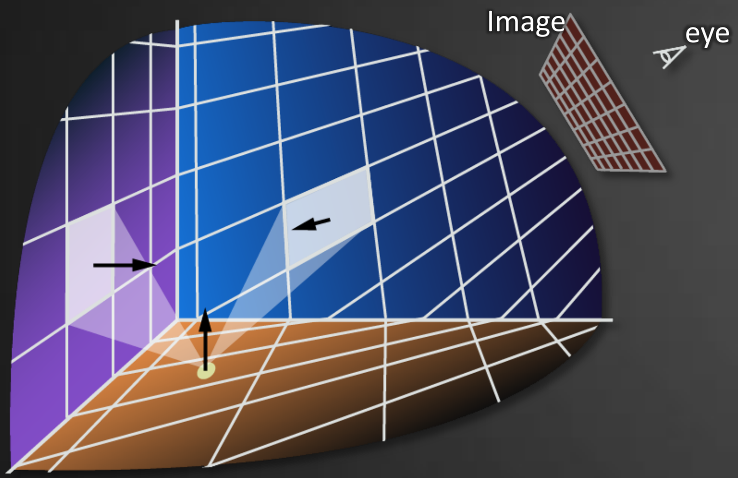
GPU ~ 550 ms
(ATI Radeon HD 5870)

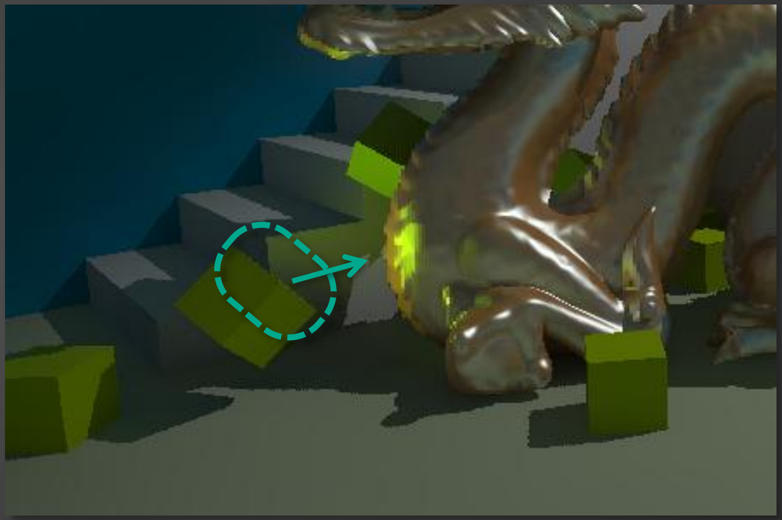




Artifacts in Screen-Space Integration

- Sources of artifacts:
  - Grazing view angles





Solution: Conservative bias compensation

Artifacts in Screen-Space Integration

- Sources of artifacts:
  - Grazing view angles
  - Hidden surfaces

Solution: Use multiple viewports (not implemented)

Conclusion

- Screen-space bias compensation (SSBC) for IR methods
  - Novel formulation of bias compensation
  - Fast approximation in screen-space
  - Feasible for integration into existing IR renderers