Radiance 4.2 Changes for 2014

Greg Ward

for Lawrence Berkeley National Lab

Highlights

- Official 4.2 release (finally!)
- Improved Perlin noise function
- Improved sampling appearance in **rpict**
- New “origin” command in **rvu**
- Hessian-based error control for irradiance caching
- New **getinfo -c** option
- **rcollate** and **rmtxop** utilities manipulate matrices
- **rfluxmtx** program to simplify **rcontrib** operation

Radiance 4.2 Release

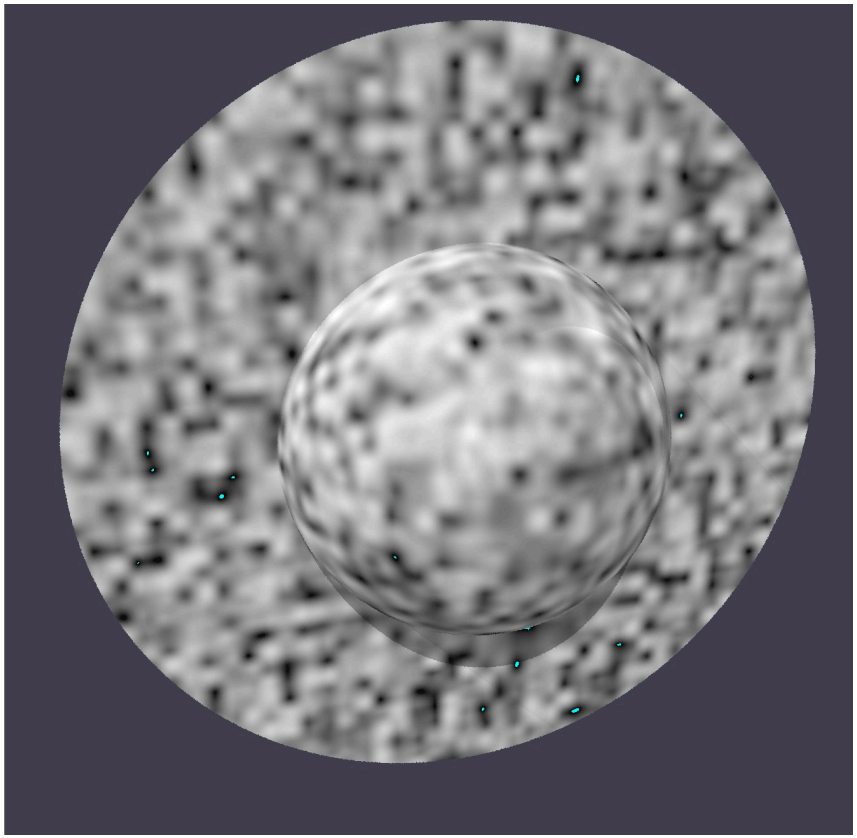
- Last official release (4.1) was November 2011
- Most people continue to use the CVS HEAD, which is also compiled and available from NREL
- Even so, it's good to make it official when a stable point is reached
- It seemed like it was time

Improved Perlin Noise Function

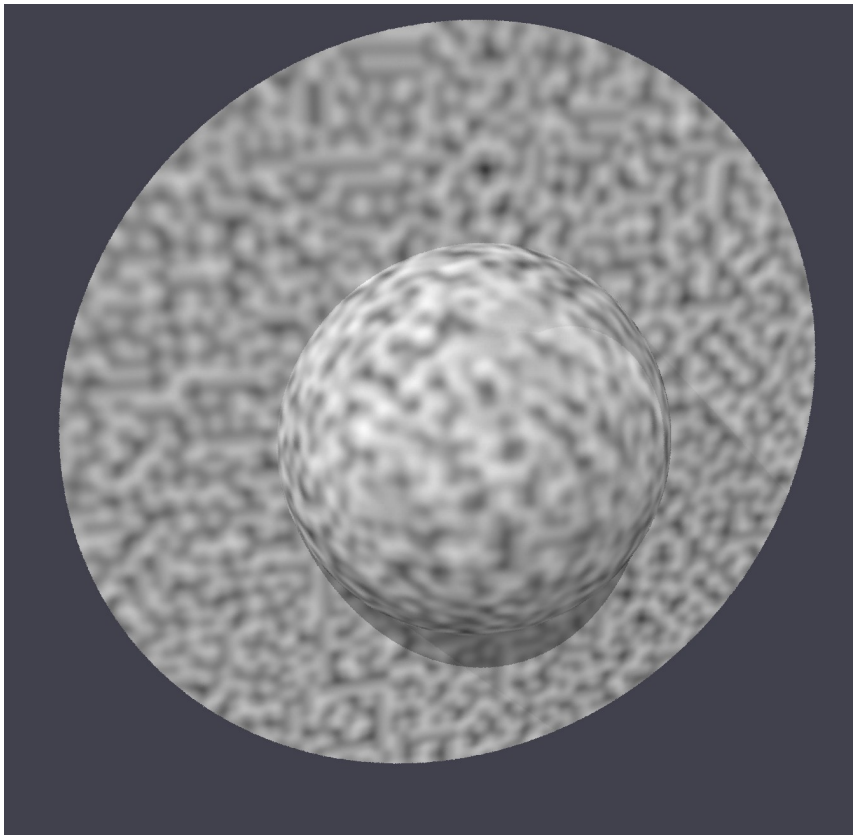
- Original implementation did not follow Perlin's version very closely
  - not as band limited as it should be
  - had issues with over-range values
- New implementation based on Perlin's improved version from 2002 SIGGRAPH
  - translation from java by Rahul Narain
  - much nicer output

Improved Perlin Noise Function

Original implementation



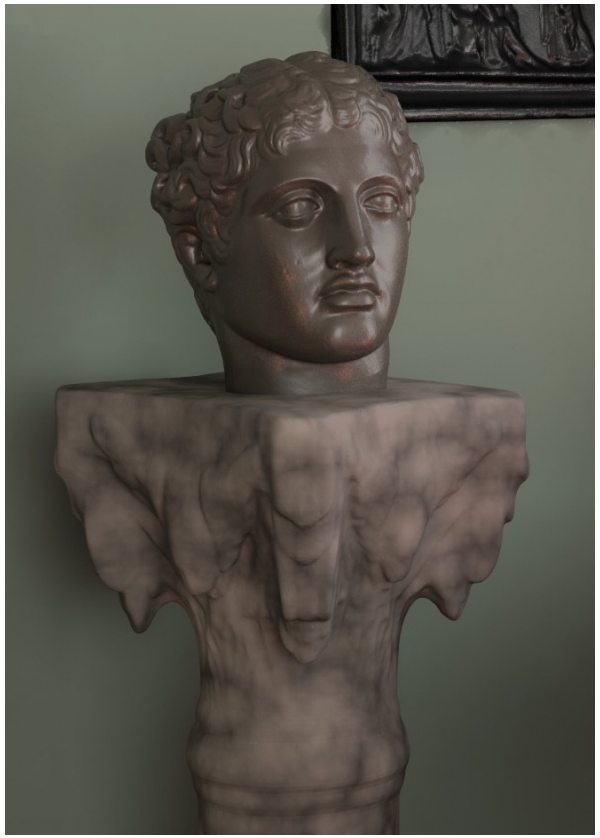
New & improved version



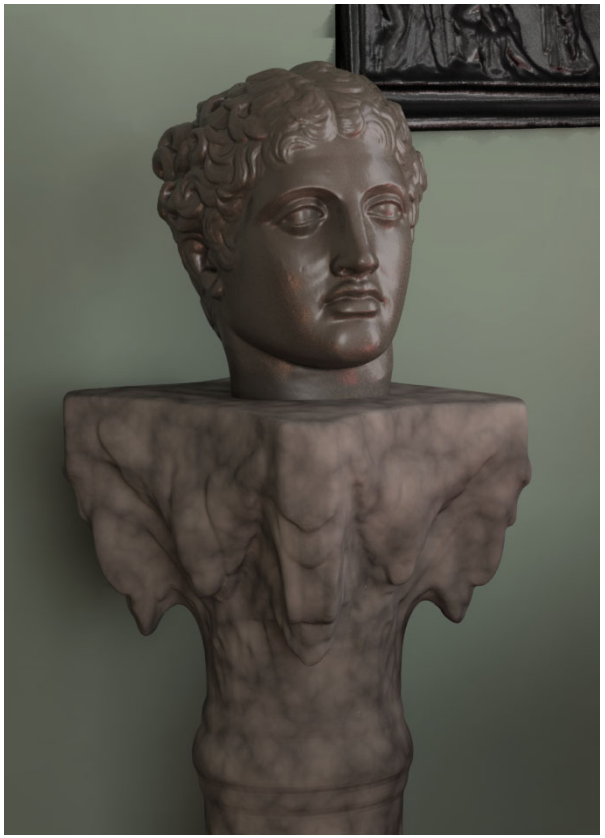
Blue regions are outside (-1,1) range

Improved Perlin Noise Function

Marble using original noise



Marble using improved noise

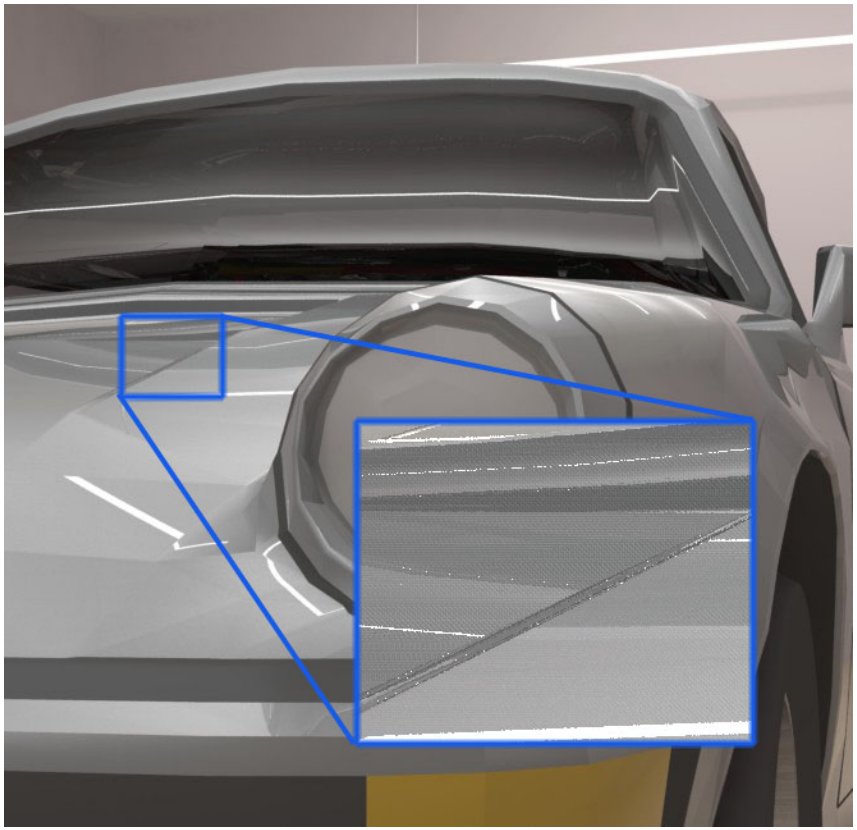


Improved RPICT Sampling

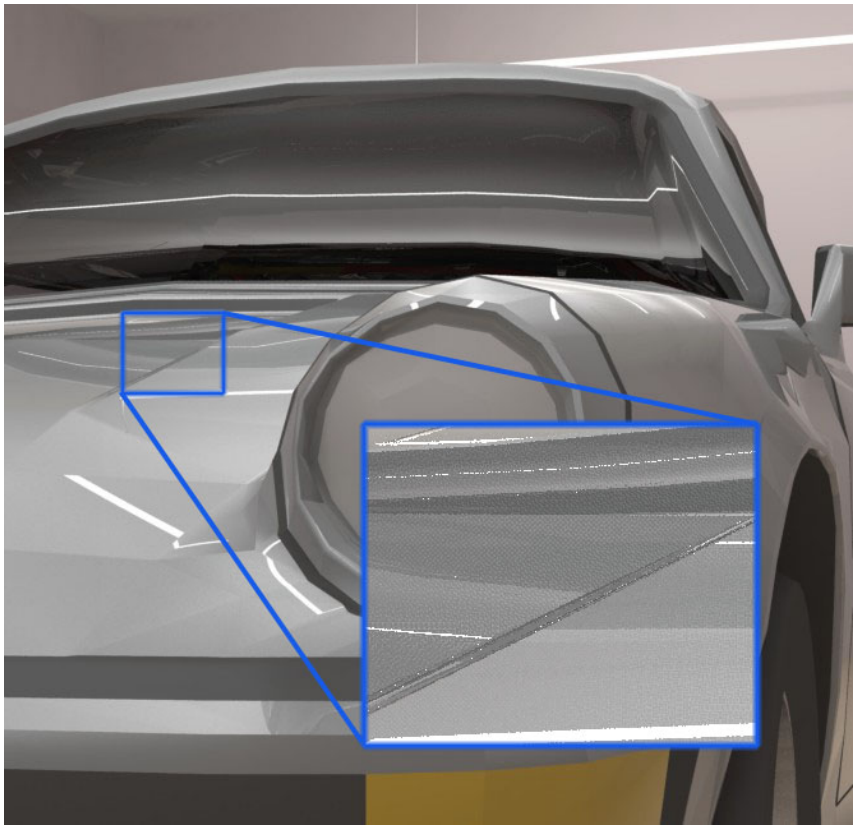
- With **-u0** (default) option, **rpict** uses anti-correlated sampling
- Original implementation gave pixels a "brushed" appearance
- New implementation uses a 2-D Hilbert space-filling curve to improve appearance

Improved RPICT Sampling

Old “brushed” look



New “screen door” look



Improved RPICT Sampling

Old “brushed” look

New “screen door” look

Easier to see section from windscreen

New RVU “origin” Command

- Suggested by John Mardaljevic
- Provides a convenient means to get “fly on the wall” view
- Helpful for debugging light leaks, etc.
- Usage:

origin [xo yo zo [xd yd zd]]

Without arguments, use cursor to choose surface to look away from…

RVIEW “origin” Command

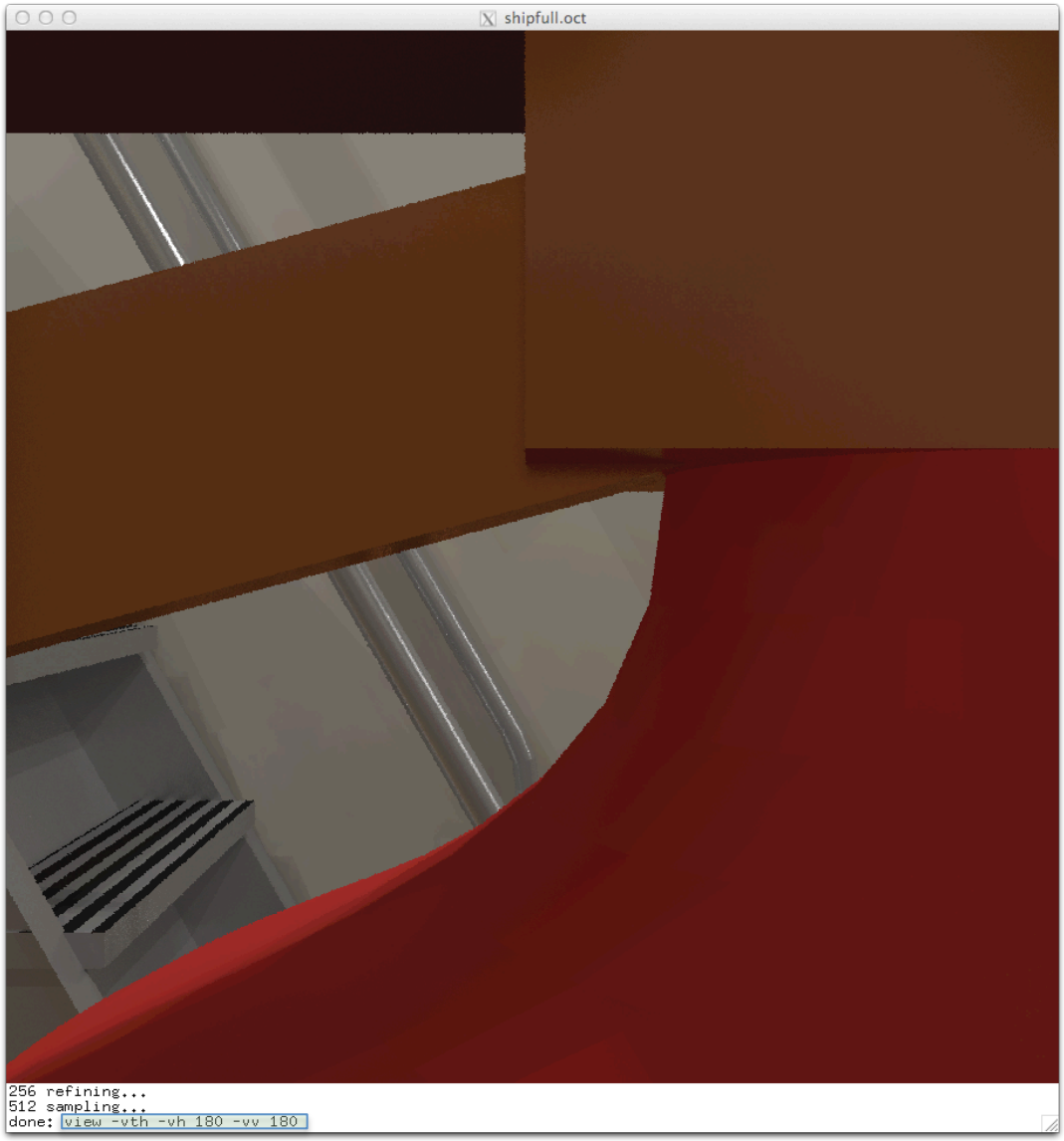
Follow “origin” command by selecting the surface position where you want your new view point to be placed, looking in the direction of the surface normal.

The “up” vector will be changed automatically if the normal vector coincides with the current one, unlike other view commands.

RVIEW “origin” Command

The new view will have the same type and size as the previous one, to be consistent with other **rvu** commands.

This can be changed with a subsequent “view” command.



RVIEW “origin” Command

Oftentimes, the user will want to change the view to a hemispherical fisheye or similar to study the light arriving at the surface point.

view -vth -vh 180 -vv 180

Hessian-based Error Control in Irradiance Cache

- The Hessian matrix holds 2nd order derivatives or "curl" of a multi-dimensional scalar function
- In *Radiance*, we can use it to bound errors in the indirect lighting calculation
- The Hessian tells us what errors to expect as we extrapolate a cached irradiance value
- This in turn tells us how closely to space our calculations to maximize efficiency

Schwarzhaupt et al., 2012

From the ACM SIGGRAPH Asia 2012 conference proceedings.

Practical Hessian-Based Error Control for Irradiance Caching

Jorge Schwarzhaupt* UC San Diego
Henrik Wann Jensen† UC San Diego
Wojciech Jarosz‡ Disney Research Zürich

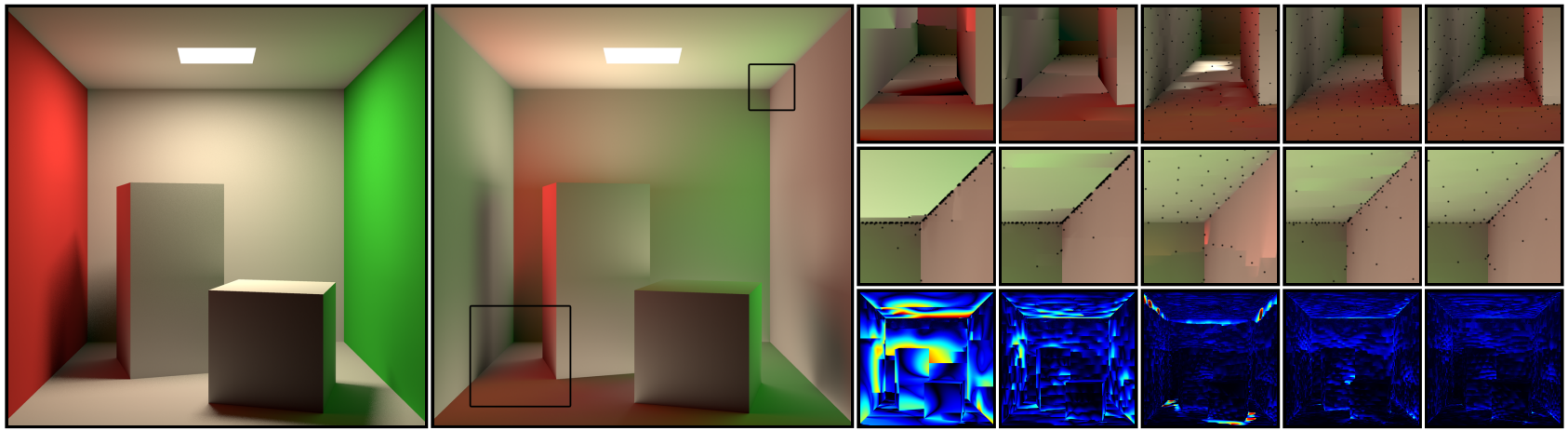
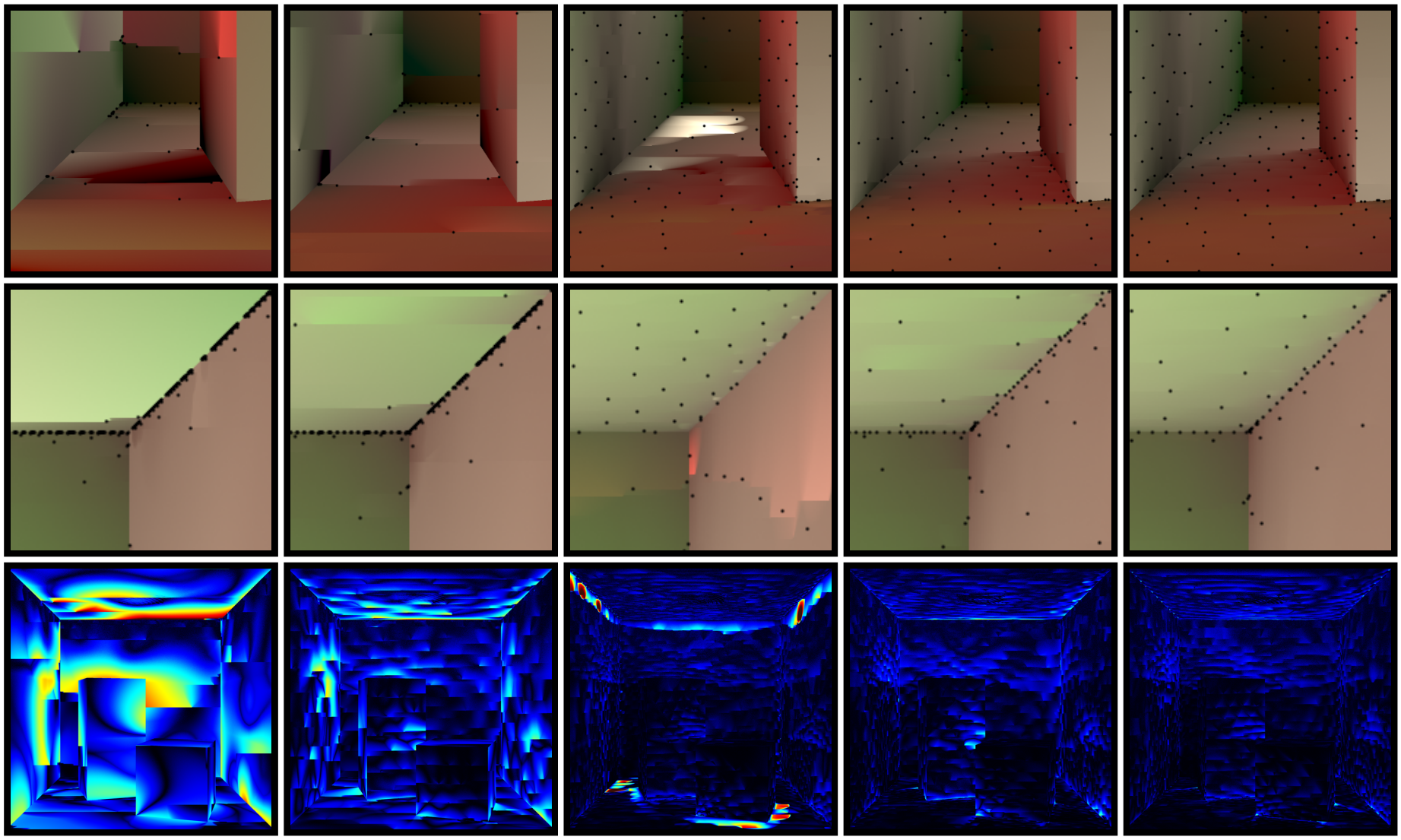
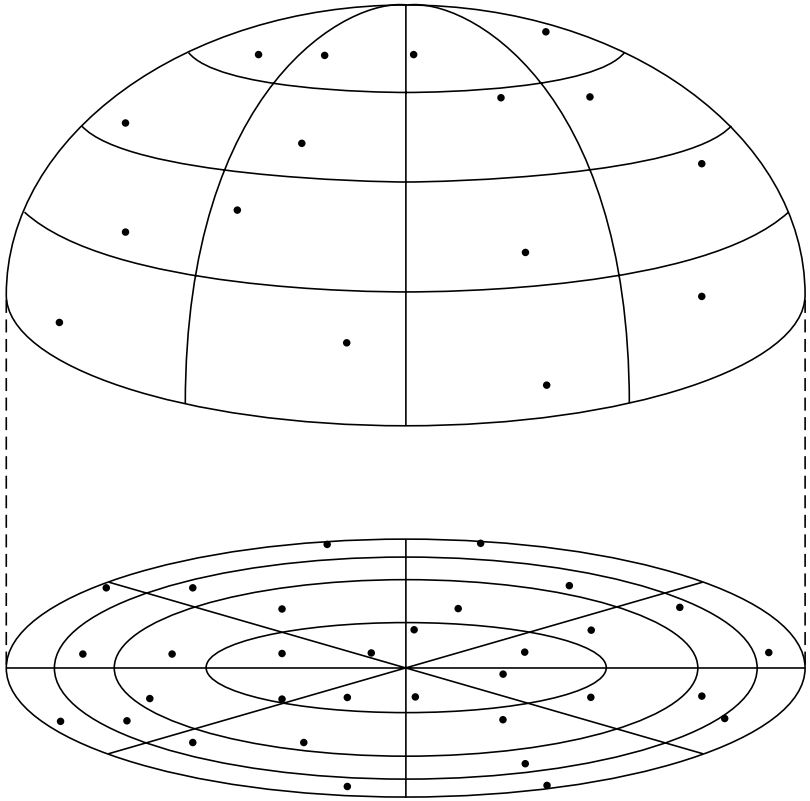


Figure 1: *Our new Occlusion Hessian significantly outperforms both the Pure and the Bounded Split-Sphere (clamped to the gradient and 150px max spacing) for irradiance caching. It also performs significantly better than the recently published occlusion-unaware Hessian error metrics by Jarosz et al. [2012].*

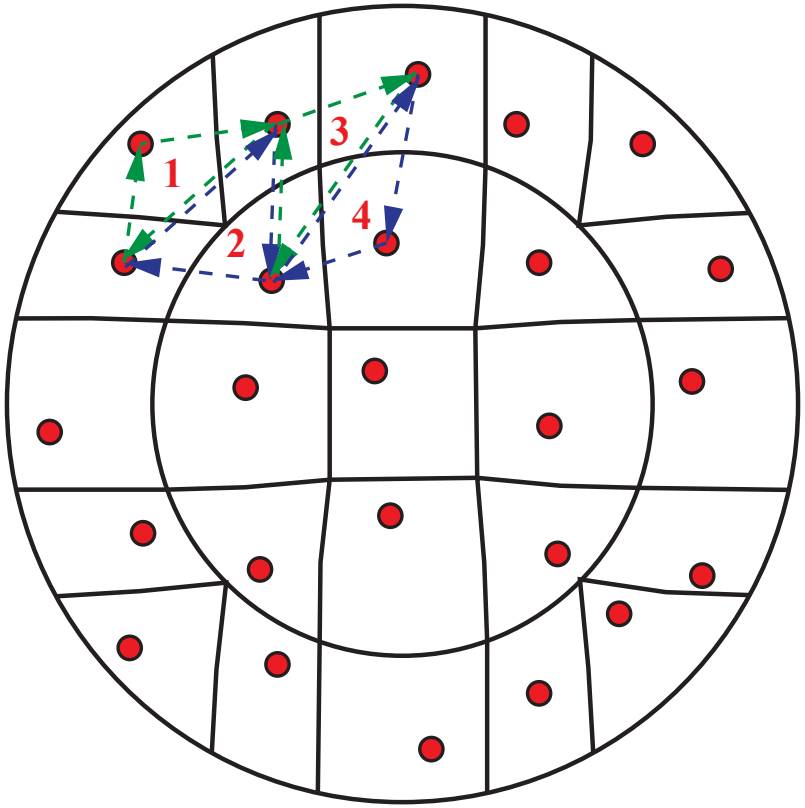

Original Split Sphere from Ward et al. 1988
Current Method Used in *Radiance*
Radiometric Hessian of Jarosz et al. 2012
Geometric Hessian of Jarosz et al. 2012
Occlusion Hessian of Schwarzhaupt et al. 2012

Improved Sampling of Hemisphere

Theta/Phi Sampling Pattern



Shirley-Chiu Sampling

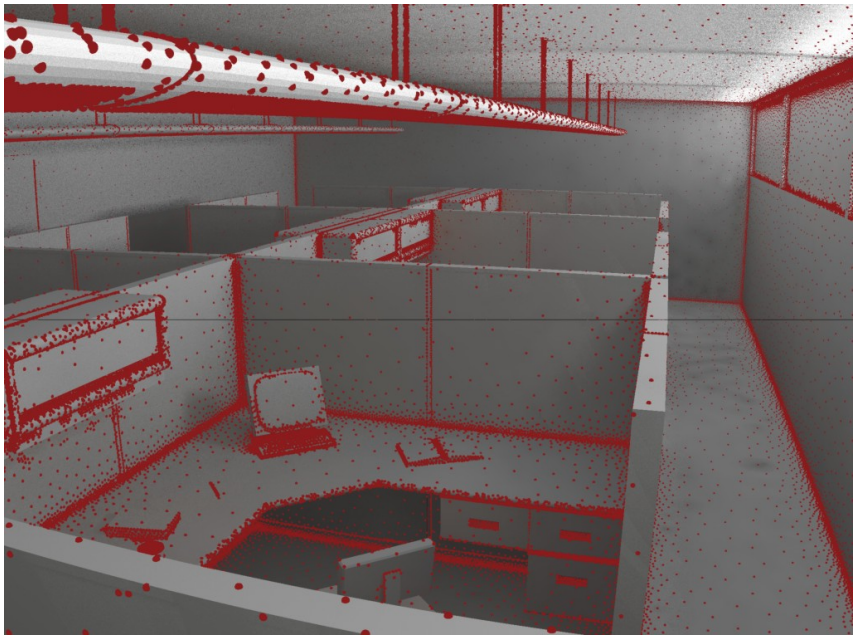


Example Scene

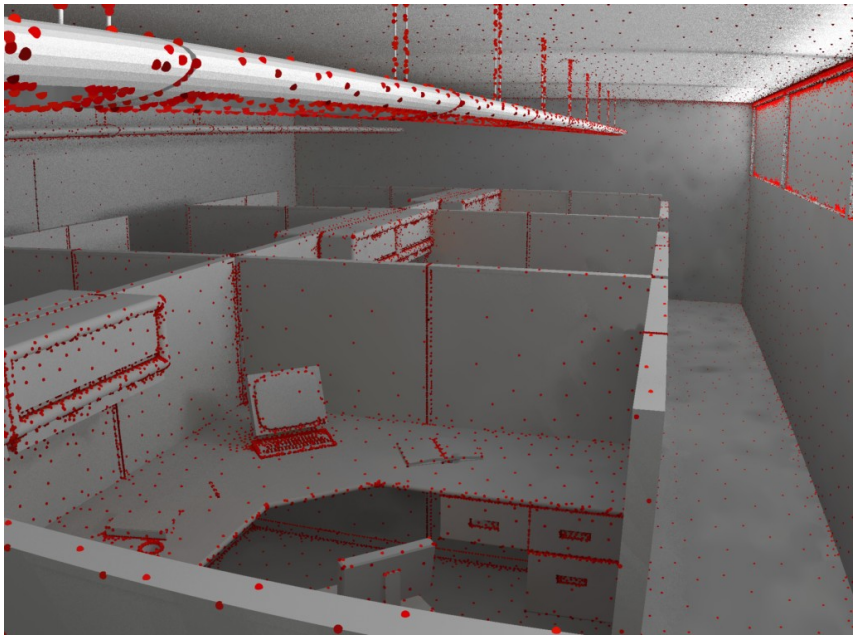
NRC cubicle office with 3M daylight redirecting film

Irradiance Cache Comparison

Old cache value placement



New cache value placement



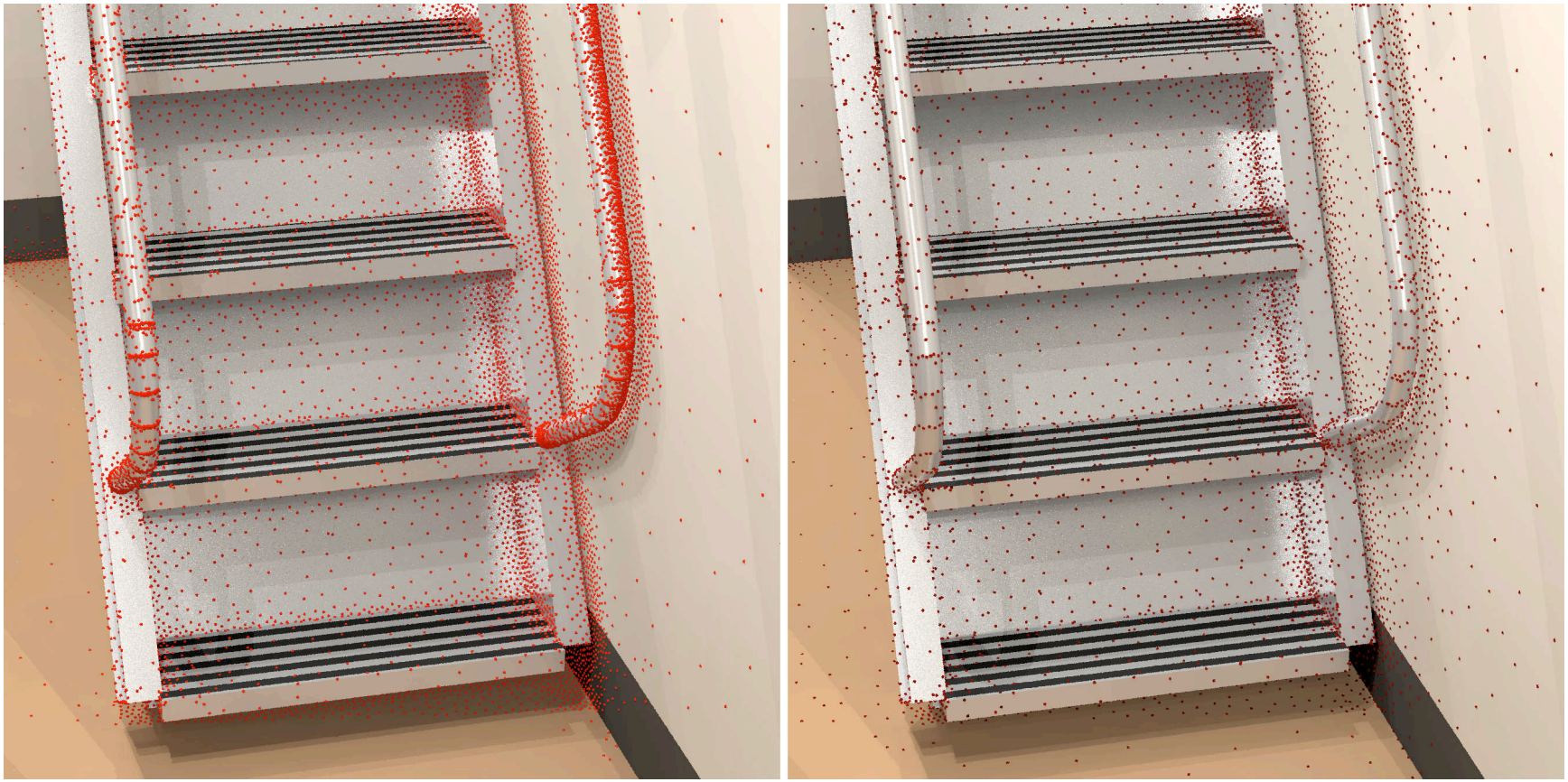
Approx. 1/3 as many values for similar accuracy

Cruiser Model (1)

Original

Hessian

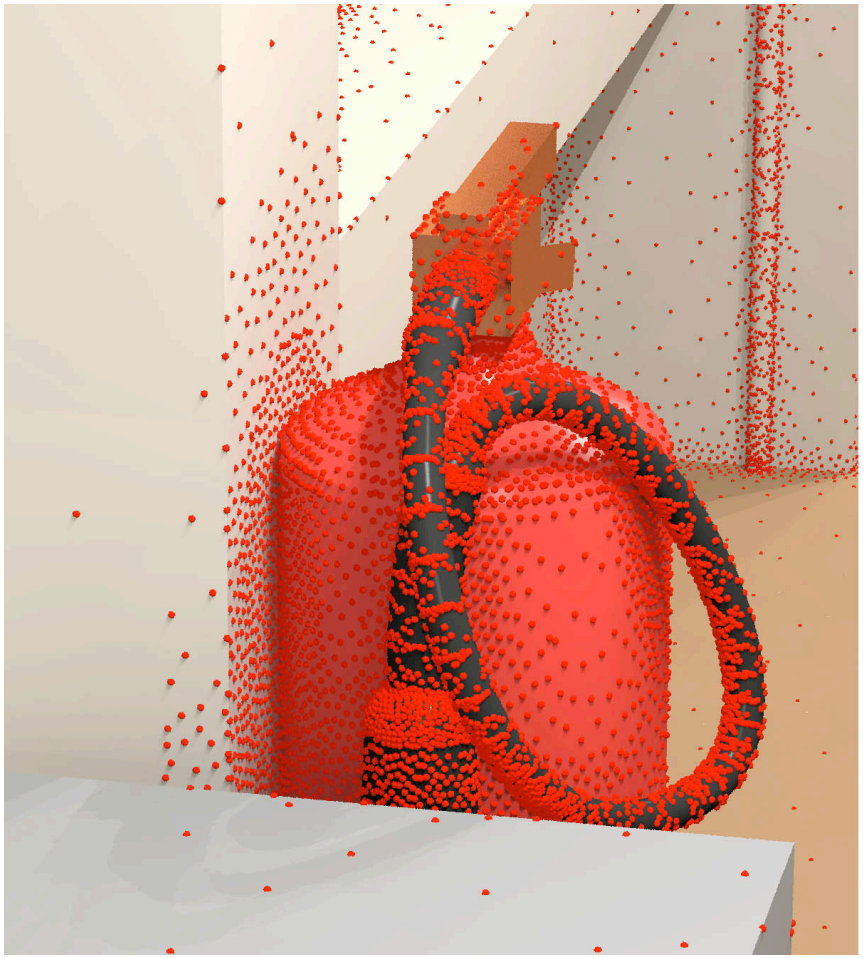
Cruiser Model (2)



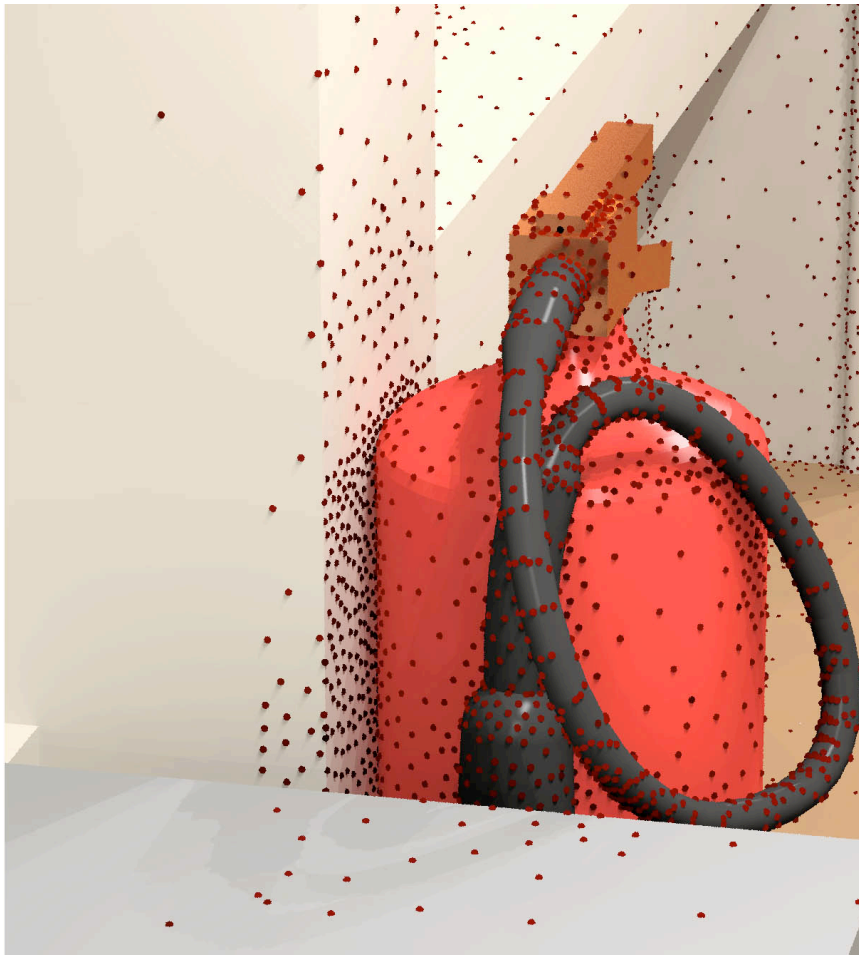
Original

Hessian

Cruiser Model (3)

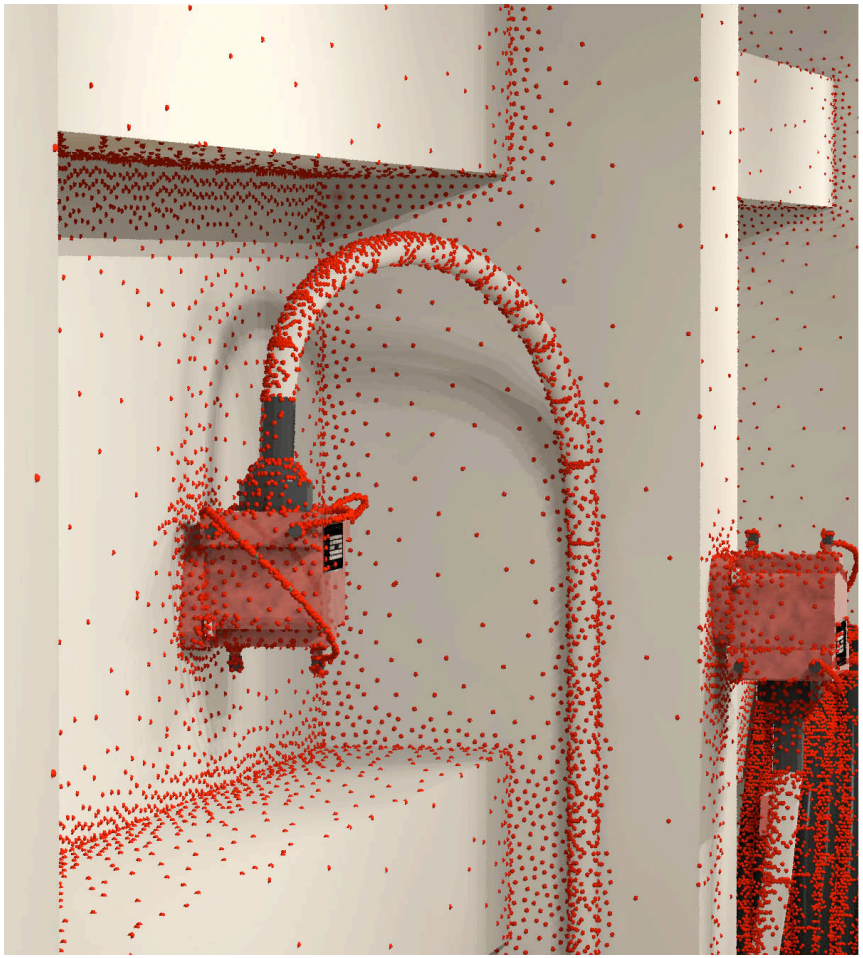


Original



Hessian

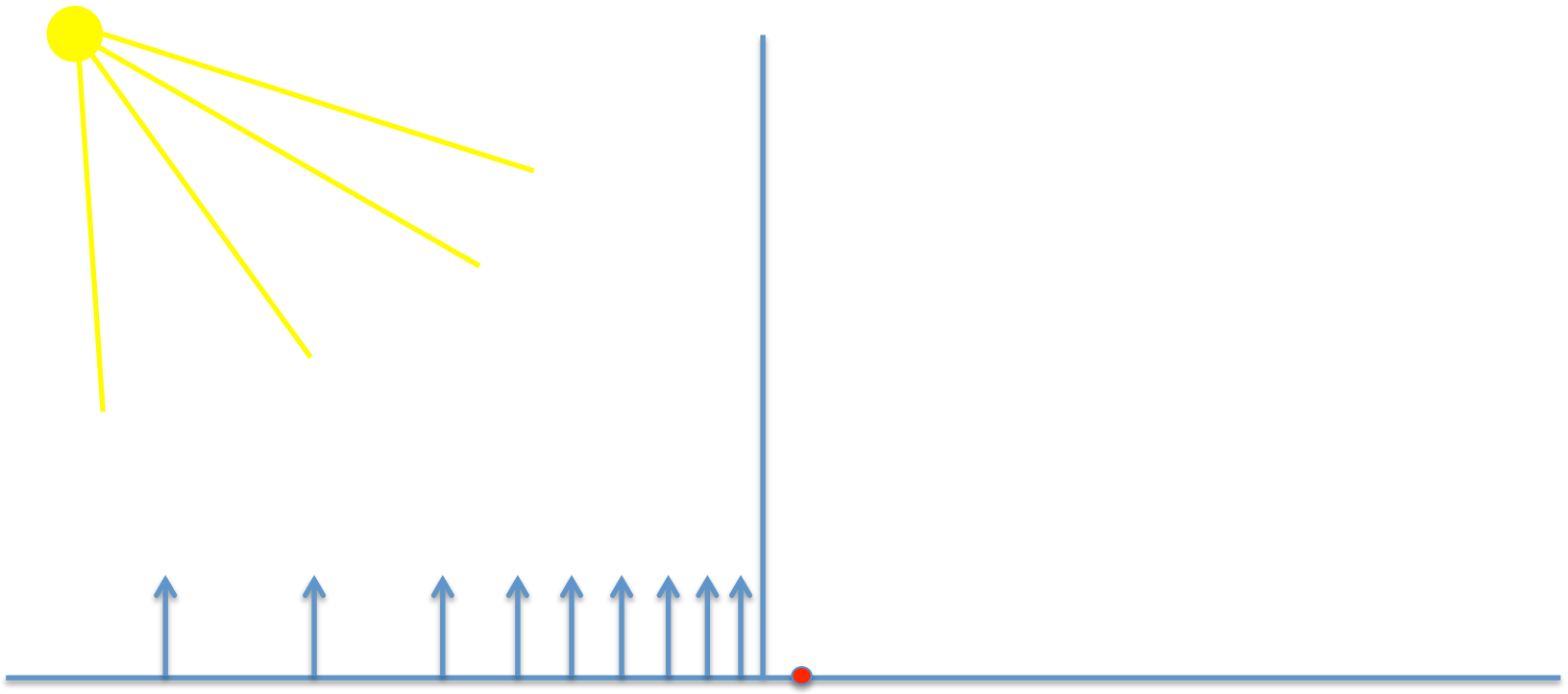
Cruiser Model (4)



Original

Hessian

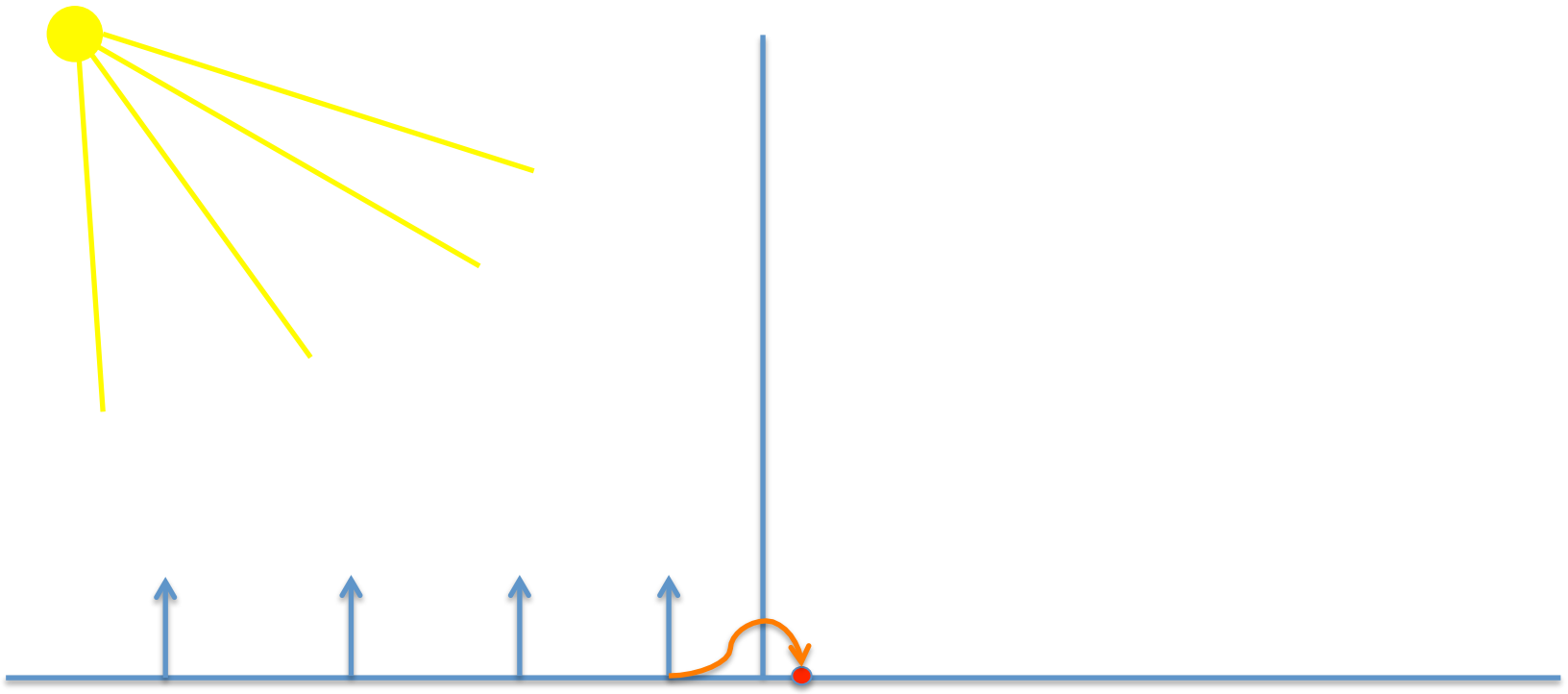
Issue with T-junctions



Old value placement

Evaluation point safe from other side

Issue with T-junctions



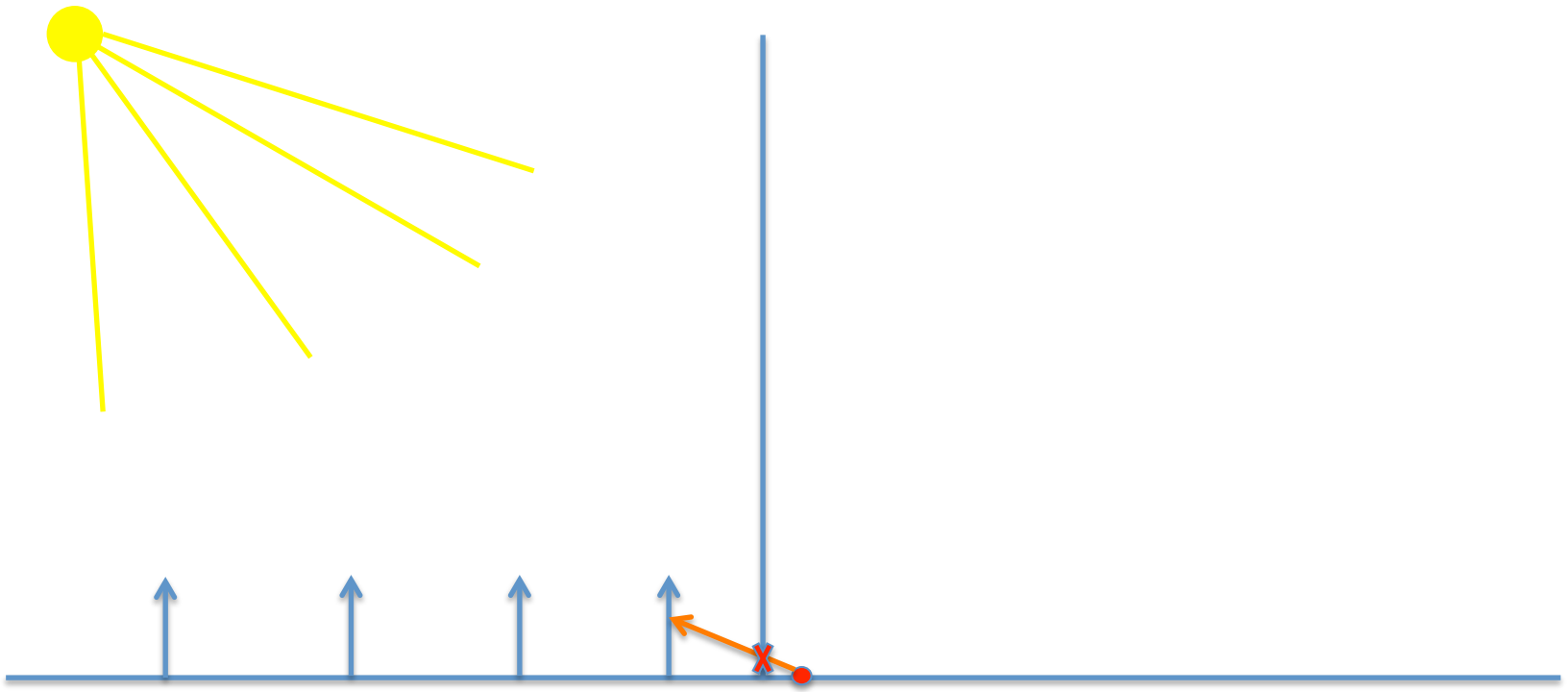
New value placement Evaluation point sees other side

Result: indirect illumination where there should be none

Issue with T-junctions

- Nearly fatal flaw, unnoticed by authors
  - spoke with Wojciech & Henrik; they said it did not show up in any of their test scenes (no T-junctions)
  - most fixes reintroduce issue with packed corners
- After 3 or 4 failed attempts, I found a fix
  - add visibility check on certain cache values
  - store “corral” flags identifying possible occluders
  - solves issue without crowding values in corners

Issue with T-junctions

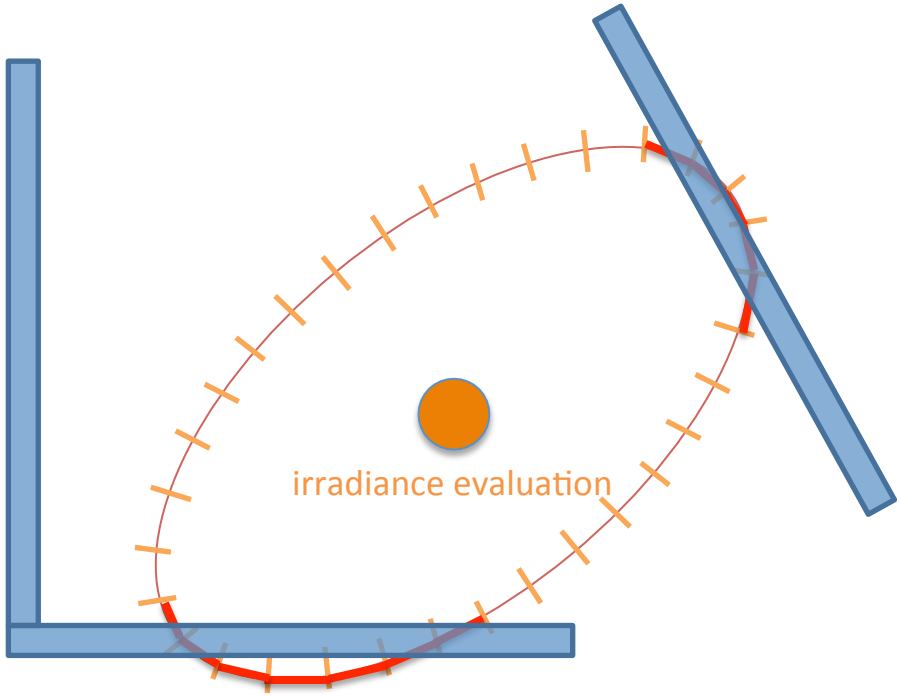


New value placement

Short ray test finds obstruction

Only test when corral says there's a hazard

Assigning Corral Flags



Set of 32 corral flags indicate potential occluders

These directions will be checked with short rays before extrapolation

Results Comparison

Schwarzhaupt et al.

With corral check

Changes from User Perspective

- Old ambient files will not be recognized
  - new structure adds elliptical radii & corral flag
  - more compact representation
- Adjusted **-aa** to match old behavior, rather than target actual accuracy
  - too sensitive and difficult to control otherwise
- Other options are essentially the same
- Compile with **-DOLDAMB** to get old behavior

New **getinfo -c** Option

- Hacker's option to manipulate data segment
- Runs the given command after header, as if:

```
( getinfo < file ; \
echo $command ; \
getinfo - < file | $command )
```

- Example:

```
getinfo -c rcalc -if -of -e '$1=10*$1' < input.mtx > output.mtx
```

New **rcollate** and **rmtxop** Comands

- General multi-component matrix operations
- **rcollate** "reshapes" matrix data, changing $N_1 \times M_1$ matrix into $N_2 \times M_2$ matrix (same number of elements)
  - can "transposes" matrix elements N×M ⇒ M×N
  - adds/removes/manipulates header
- **rmtxop** loads matrixes into double arrays, concatenates, scales, sums, transforms

Matrix File Format

#?RADIANCE
gendaymtx -of nyc01.sp1.wea
LATLONG= 42.75000000 -73.80000000
NROWS=146
NCOLS=8760
NCOMP=3
FORMAT=float

followed by binary float data,
row is outer sort, then columns, then components
…

The file above is 146*8760*3*4 bytes = 14.6 Mbytes

rcollate Details

- Never interprets data elements
  - fast: ASCII components are copied as text blocks
  - can transpose very large matrix by mapping file
  - useful for feeding **rcalc** & converting its output
  - however: cannot convert data between formats
- Flexible regarding information header
  - can work without header (**-h**)
  - can add missing header (**-hi**)
  - can remove unwanted header (**-ho**)

rcollate Examples

Feed **rcalc** one RGB record at a time & remove header:

```
rcollate -oc 1 -oh input.mtx | rcalc -e '$1=$1+$2+$3'
```

Transpose a 100×25 2-comp. float matrix w/o headers:

```
rcollate -h -ir 100 -ic 25 -ff2 -t input.mtx > output.mtx
```

Add a header to binary data as required by **rmtxop**:

```
rcollate -ih -ir 30 -ic 19 -fd3 input.mtx | rmtxop …
```

rmtxop Details

- Reads matrix data, operates in memory
  - minimal header information needed:
    - NCOMP, NROWS, NCOLS, format
  - also accepts *Radiance* pictures and BSDF (XML) files
  - writes ASCII, float, double, and RGBE formats
  - has as many as 3 matrices in RAM at a time
    - uses double type (8-bytes/component) for all input
- More flexible than **dctimestep**
  - but somewhat less memory- & time-efficient
    - **dctimestep** uses float data, optimizes matrix multiplication

rmtxop Examples

Convert matrix BSDF to picture:

```
rmtxop -fc bsdf.xml > bsdf.hdr
```

Concatenate matrix from *stdin* and write as ASCII:

```
… | rmtxop -fa left.mtx - > output.mtx
```

Convert RGB matrix to grayscale and add another:

```
rmtxop -c .3 .6 .1 rgb1.mtx + gry2.mtx > gryout.mtx
```

Concatenate three matrices, transposing second:

```
rmtxop inp1.mtx -t inp2.mtx inp3.mtx > out.mtx
```

New **rfluxmtx** Program

- Long-promised front-end to **rcontrib**
- Simplifies common operations
  - generates hemispherical surface samples
  - sets **rcontrib** bin variables and *.cal files
- Generalizes sampling for light pipes, etc.
- Replacement for **genklemsamp**
  - will also simplify **genBSDF** implementation

Basic **rfluxmtx** Operation

rfluxmtx [-v][rcontrib options] sender.rad receiver.rad [-i system.oct][system.rad ..]

- Most options are simply passed to **rcontrib**
  - the **-v** option reports on execution
- Sender file contains single sender object
  - special comments identify sampling basis
- Receiver file contains one or more objects
  - similar comments indicate sampling bins
- System files given to **oconv** before receiver.rad

Comparison to **genklemsamp**

```
oconv -w -f material_detailed.rad simple.rad \
        dummysky.rad > dumbsky.oct
genklemsamp -vd -0.415671599 0.909514773 0 -c 20000 \
        material.rad bg4wind.rad \
        | rcontrib -n 2 -c 20000 -faf -e MF:4 -f reinhart.cal \
                -b rbin -bn Nrbins -m skyglow \
                @rtc_dmx.opt dumbsky.oct \
        > bg4.dmx
rm dumbsky.oct


rfluxmtx -w -n 2 -c 20000 -ff @rtc_dmx.opt bg4wind.rad \
        dummysky.rad -w material.rad simple.rad \
        > bg4.dmx
rfluxmtx: opening pipe to: rcontrib -fo+ -n 2 -w -ab 2 -ad 300 -fdf -c 20000 \
        -f reinhartb.cal -p MF=4,rNx=0,rNy=0,rNz=-1,Ux=0,Uy=1,Uz=0 \
        -bn Nrbins -b rbin -m skyglow -b 0 -m groundglow -y 145 \
        '!oconv -f -w material.rad simple.rad dummysky.rad'
rfluxmtx: sampling 145 directions
```

Sender File

```
rfluxmtx -v -n 2 -c 20000 -ff @rtc_dmx.opt
bg4wind.rad \
      dummysky.rad -w material_detailed.rad simple.rad \
      > bg4.dmx
```

```
#@rfluxmtx h=kf u=+Z

Translucent_20 polygon zone02.rad00014b
0
0
12
   -0.733460650921     11.5416867963          0.762
   -0.733460650921     11.5416867963          2.7178
    0.652638345832     12.1751696194          2.7178
    0.652638345832     12.1751696194          0.762
```

No need to define material “Translucent_20”

Receiver File

```
rfluxmtx -v -n 2 -c 20000 -ff @rtc_dmx.opt
bg4wind.rad \
        dummysky.rad -w material_detailed.rad simple.rad \
        > bg4.dmx
```

BEFORE

```
void glow skyglow
0
0
4 1 1 1 0

skyglow source sky
0
0
4 0 0 1 360
```

```
#@rfluxmtx h=u

void glow groundglow
0
0
4 1 1 1 0

groundglow source ground
0
0
4 0 0 -1 180
```

```
#@rfluxmtx h=r4 u=+Y

void glow skyglow
0
0
4 1 1 1 0

skyglow source sky
0
0
4 0 0 1 180
```

Separate (uniform) ground source

Advantages of **rfluxmtx**

- Simpler operation
  - manages **rcontrib** parameters/order
  - generates source sample rays
- Handles non-planar sources & receivers
- Unifies hemispherical sampling methods
  - consistent application of Tregenza & Reinhart sky, Klems hemispherical bases, Shirley-Chiu disk
- Sender & receiver need not be parallel
- Receiver may be reused as subsequent sender

Pass-through Mode

- Specify '-' in place of sender file, e.g.:

  sample_generator | rfluxmtx [options] - receiver.rad

- **rfluxmtx** executes **rcontrib**, but does not generate sample rays
  - standard input is sent to **rcontrib** directly
- Same behavior as executing command reported by **-v** option
  - provided primarily as a convenience

Example Pass-through Mode

```
vwrays -ff -vf back.vf -x 600 -y 600 \
      | rfluxmtx -v `vwrays -vf back.vf -x 600 -y 600 -d` -n 4 \
            -ffc -ab 12 -ad 50000 -lw 2e-5 - window.rad testroom.mat testr
rfluxmtx: running: rcontrib -fo+ -n 4 -ab 12 -ad 50000 -lw 2e-5 -x 600 -y
      -ld- -ffc -c 1 -o vmx/window_%03d.hdr -f klems_full.cal \
      -bn Nkbins -b 'kbin(0,1,0,0,0,1)' -m windowglow \
      '!oconv -f testroom.mat testroom.rad window.rad'
```

```
#@rfluxmtx h=kf u=Z o=vmx/window_%03d.hdr

void glow windowglow
0
0
4    1    1    1     0

windowglow polygon window
0
0
…
```

Conclusions

- Radiance 4.2 release is official
- Includes new Hessian-based “ambient” calc
- Many(!) other improvements since 4.1
- New **rfluxmtx** program present, but still undergoing bug fixing & improvements
  - Keep up-to-date with CVS HEAD if interested
- Photon-mapping integration building on 4.2