Bartenbach
L'chtLabor

BSDF Crash Course and the RADIANCE 3-Phase-Method

David Geisler-Moroder

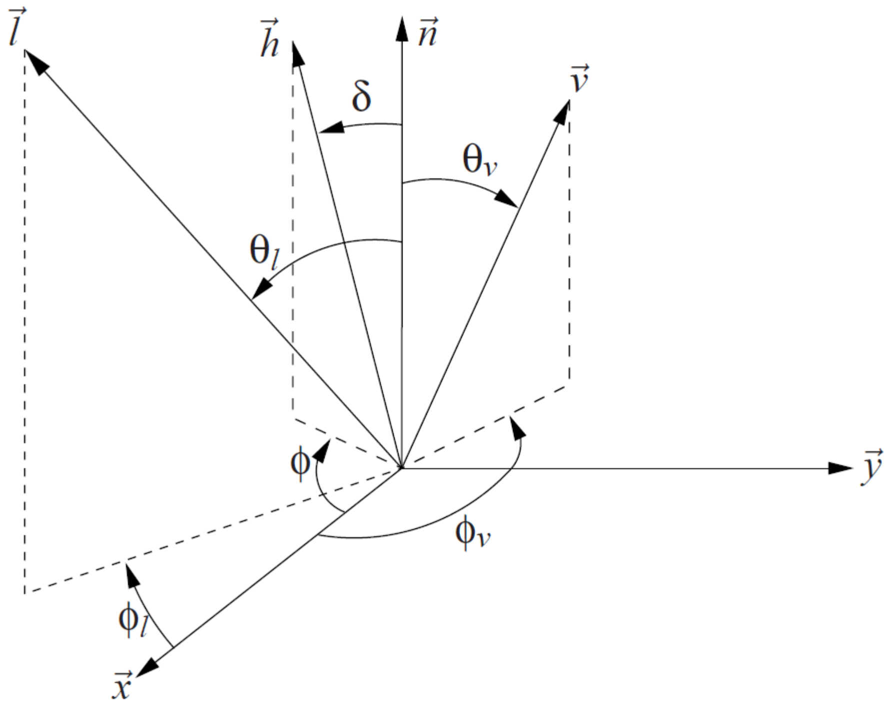
12.09.2012

OUTLINE

- basics of BSDFs
  - theory
  - discretizations
- generating BSDFs
  - measurements
  - simulations
- using BSDFs in RADIANCE
  - mkillum
  - BSDF material primitive
- using BSDFs in the RADIANCE 3-phase method
- Q & A

OUTLINE

- basics of BSDFs
  - theory
  - discretizations
- generating BSDFs
  - measurements
  - simulations
- using BSDFs in RADIANCE
  - mkillum
  - BSDF material primitive
- using BSDFs in the RADIANCE 3-phase method
- Q & A



BSDF – BASICS

BSDF, BTDF, BRDF, BSSSDF?

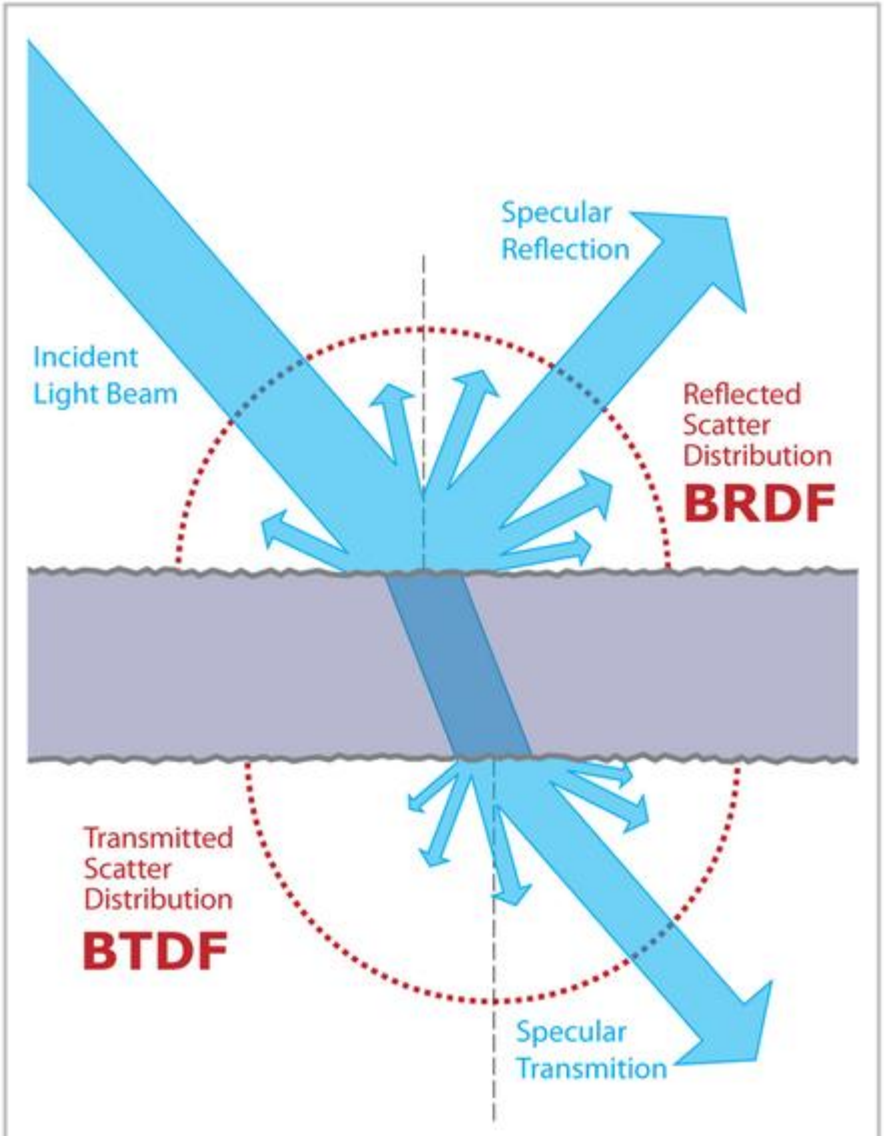
BSDF – bidirectional scattering distribution function

BRDF – bidirectional reflection distribution function

BTDF – bidirectional transmission distribution function

B(S)SSDF – bidirectional (sub)surface scattering distribution function

we are talking about data-driven BSDFs!



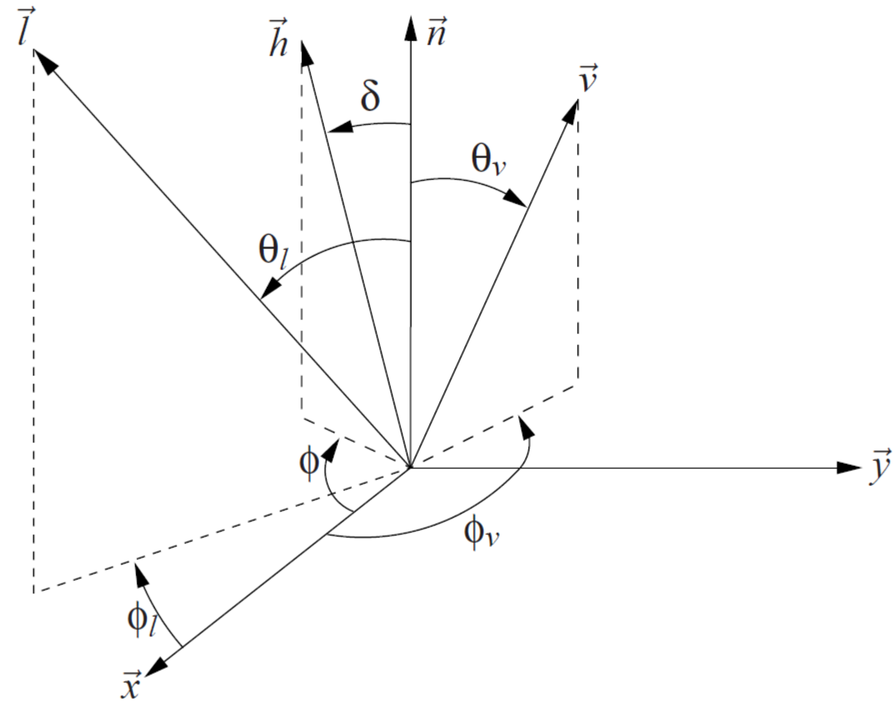
source: wikimedia.org

Bartenbach LichtLabor

rendering equation

$$L_v(\theta_v,\phi_v) = \int_0^{2\pi}\int_0^{\pi/2} L_l(\theta_l,\phi_l) f(\theta_l,\phi_l;\theta_v,\phi_v)\cos\theta_l \sin\theta_l \, d\theta_l \, d\phi_l$$

(θ_l,ϕ_l)	light source direction
(θ_v,ϕ_v)	view point direction
$f(\theta_l,\phi_l;\theta_v,\phi_v)$	BSDF
$L_l(\theta_l,\phi_l)$	radiance from light source direction
$L_v(\theta_v,\phi_v)$	radiance to view point direction

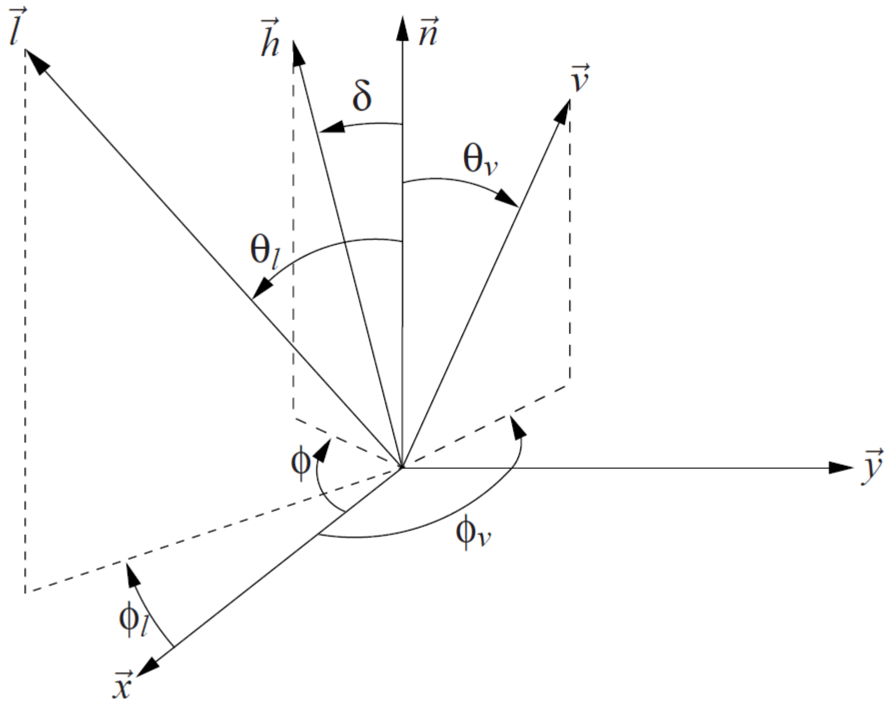


further reading:
Kajiya J. T.: The rendering equation. SIGGRAPH Comput. Graph. 20, 4 (1986), 143–150.
Nicodemus et al.: Geometrical Considerations and Nomenclature for Reflectance. NBS Monograph 160, U. S. Dept. of Commerce, 1977.

physical plausibility

1. Helmholtz reciprocity

$$f(\theta_l,\phi_l;\theta_v,\phi_v) = f(\theta_v,\phi_v;\theta_l,\phi_l)$$



2. energy balance

albedo

$$a(\theta_l,\phi_l) = \int_0^{2\pi}\int_0^{\pi/2} f(\theta_l,\phi_l;\theta_v,\phi_v)\cos\theta_v \sin\theta_v d\theta_v d\phi_v$$

bounded by 1

further reading:
Lewis R. R.: Making shaders more physically plausible, *Computer Graphics Forum (Eurographics '94 Conference Issue) 13, 3 (1994), 1–13.*

BSDF – DISCRETIZATIONS

Bartenbach LichtLabor

Klems patches

- subdivision of hemisphere into

  145 patches

- approx. **equal illuminance** from each

  patch if luminance is constant in hemisphere

- 9 **θ ranges**

  {0°-5°, 5°-15°, 15°-25°, 25°-35°, 35°-45°,

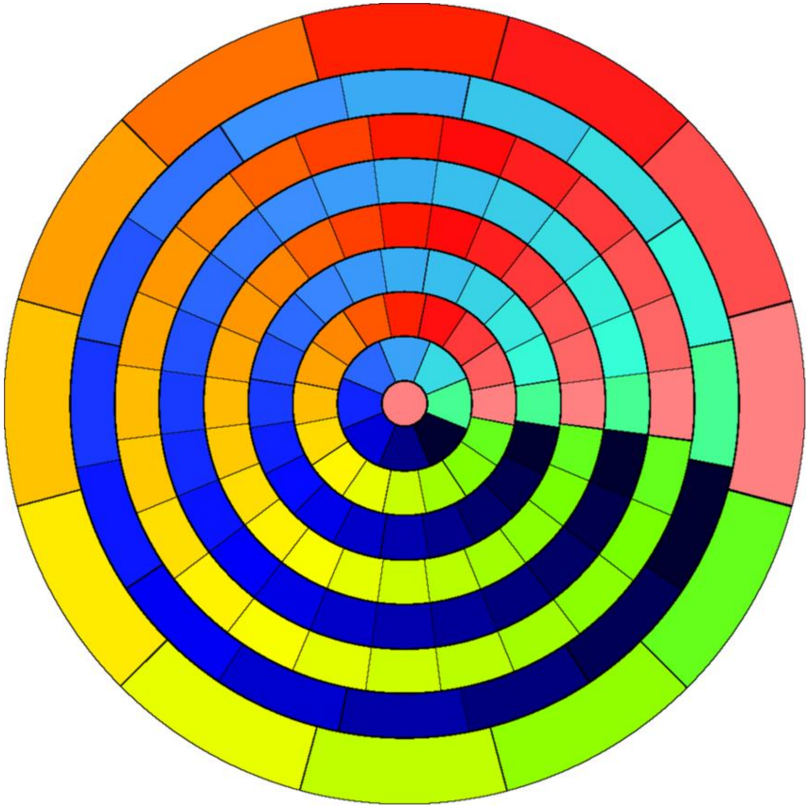
  45°-55°, 55°-65°, 65°-75°, 75°-90°}

- **ϕ subdivisions** per θ range

  {1, 8, 16, 20, 24, 24, 24, 16, 12}

- **average solid angle** $2\pi/145 = 0.0433$ sr,

  i.e. cone with 2 x 6.73° apex angle $[2\pi*(1-\cos(\alpha/2)) = 2\pi/145]$



further reading:

Klems J.H.: A new method for predicting the solar heat gain of complex fenestration systems; Overview and derivation of the matrix layer calculation. ASHRAE Transactions 100 (1), 1994

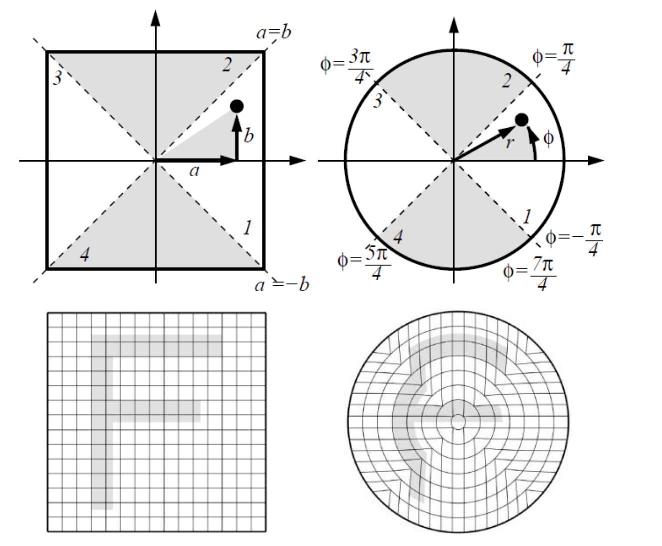
BSDF – DISCRETIZATIONS

variable resolution BSDFs

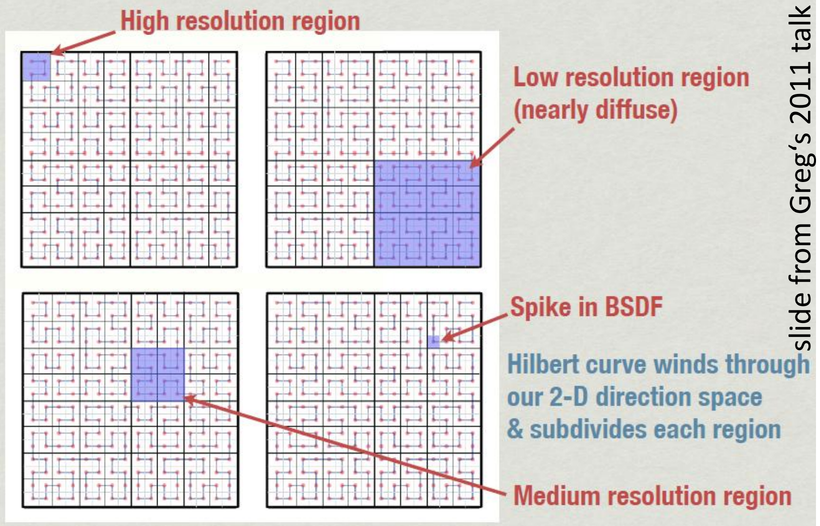
- **idea: high resolution for spikey regions**
  low resolution for smooth regions
- **based on Shirley-Chiu-mapping** (preserves fractional area, i.e. projected solid angle)
- **maximum dimensions in 4D 2^{2n} x 2^{2n}**
  (n = 4 / 5 / 6: 256^2 / 1024^2 / 4096^2)

+ efficient data structure
(ideal diffuse reflector needs 1 value {1/π})

– no matrix structure
(daylight coefficient approach)



images from Shirley & Chiu paper



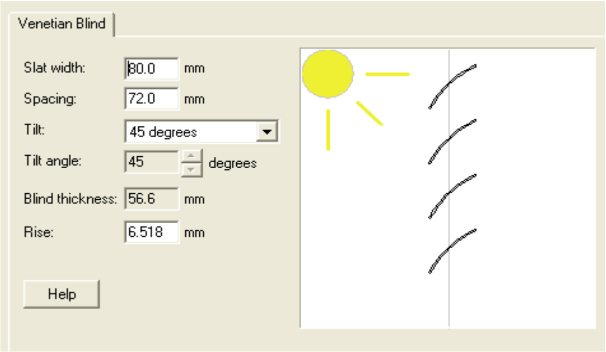
slide from Greg's 2011 talk

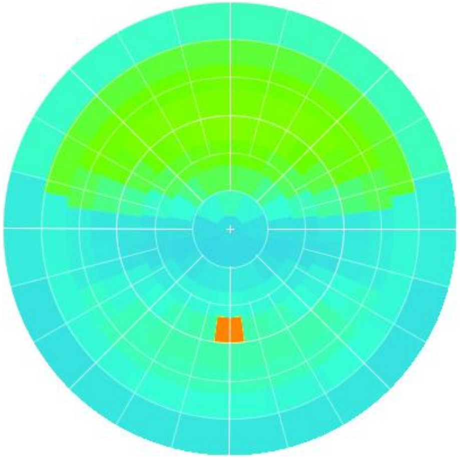
further reading:

Shirley P., Chiu K.: A Low Distortion Map between Map and Square, Journal of Graphics Tools 2(3), 1977
Ward G.: Presentations at the 10th Radiance Workshop, radiance-online.org/community/workshops/2011-berkeley-ca
Ward G. et al.: „A Practical Framework for Sharing and Rendering Real-World Bidirectional Scattering Distribution Functions", to be submitted

Outline

- basics of BSDFs
  - theory
  - discretizations
- generating BSDFs
  - measurements
  - simulations
- using BSDFs in RADIANCE
  - mkillum
  - BSDF material primitive
- using BSDFs in the RADIANCE 3-phase method
- Q & A





GENERATING BSDFs

measurement

- in-plane measurement

- classical goniometers

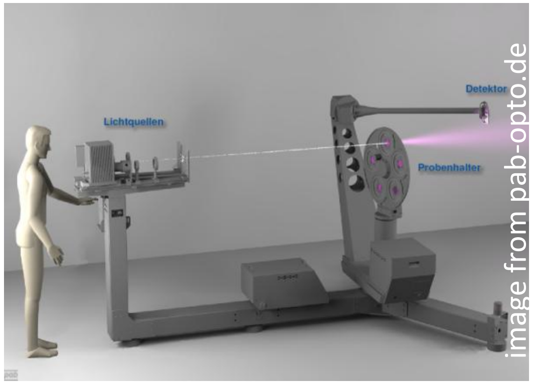


image from pab-opto.de

image from lighttec.fr

- CCD based goniometers

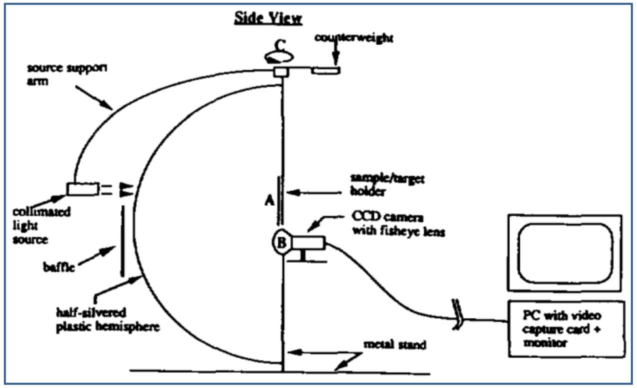


image from Greg's 1992 SG paper

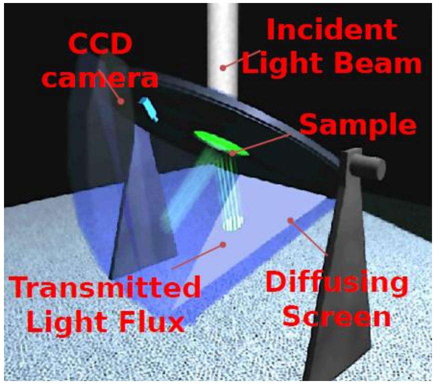


image from J.Kämpf, talk 2011

image from lighttec.fr

further reading:
pab-opto.de (Peter Apian-Bennewitz)
various talks from Radiance Workshops 2010 and 2011

GENERATING BSDFs

Bartenbach LichtLabor

simulation

- **genBSDF**

  part of the RADIANCE software package

  http://radiance-online.org/cgi-bin/viewcvs.cgi/ray/src/util/genBSDF.pl

- **WINDOW6 / WINDOW7**

  LBNL software for calculation of total window thermal performance indices

  windows.lbl.gov/software/window/window.html

- **commercial software (e.g. LucidShape, ASAP)**

  need to create own „patch – illuminantion“ and conversion from ray file to patches

further reading:
Greg’s talks from Radiance Workshops 2010 and 2011
WINDOW documetation http://windows.lbl.gov/software/window/6/w6_docs.htm

GENERATING BSDFs

genBSDF

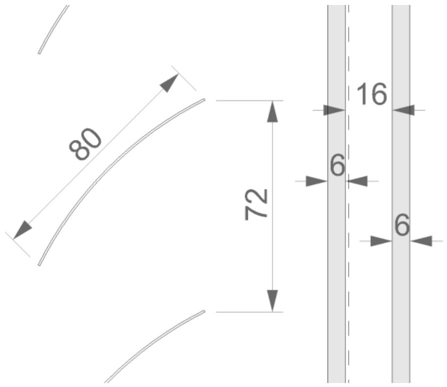
- raytracing
- Klems patches + var. resolution (3D/4D)
- geometry & material: RADIANCE scope
- light
- BSDF for subsystem or material
- parameter settings
- …

WINDOW6/7

- radiosity
- Klems patches
- limited geometry & material
- light and thermal
- BSDF for subsystem
- databases (IGDB and CGDB)
- …

example blind

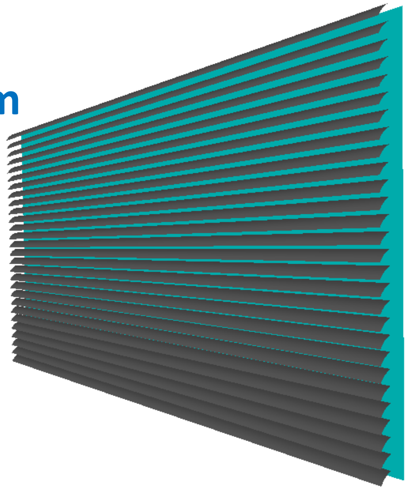
- exterior venetian blinds
- diffuse, light gray, $\rho = 48\%$
- double glazing
- tilt angle 45°



GENERATING BSDFs – EXAMPLE BLIND

Bartenbach LichtLabor

genBSDF



1. define materials and generate geometry including the glazing system
(genblinds, obj2rad, …)
x = width, y = height, z = depth
!! +z into room (no +z in model!)

2. run genBSDF

Klems:

```
genBSDF -n 8 +f +b +geom meter system.rad > system_Klems.xml
```

default: Klems, backward component, geometry into xml
add: forward component and use 8 cores

var. Res:

```
genBSDF -n 8 -t4 5 -c 10240 +f +b +geom meter system.rad > system_VarT45.xml
```

change: var. Resolution BSDF (4D) with max. resolution 1024 x 1024,
number of samples per input region

parameters good to know:

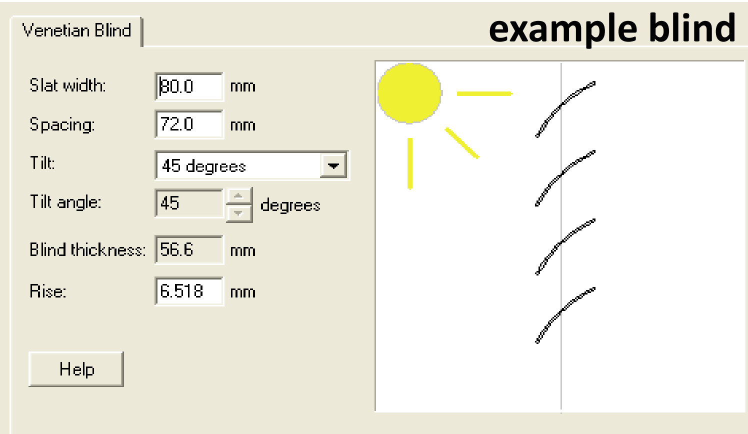
-dim x1 x2 y1 y2 z1 z2, -r "rtopts" (check the genBSDF manpage for details)

hidden parameter:

-t p[%] … percentage for rttree_reduce (size & accuracy of var. resolution BSDF),
a value < 0 skips rttree_reduce → full max. resolution

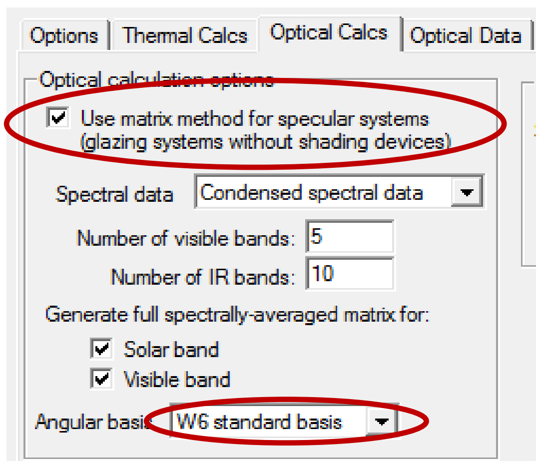
GENERATING BSDFS – EXAMPLE BLIND

WINDOW

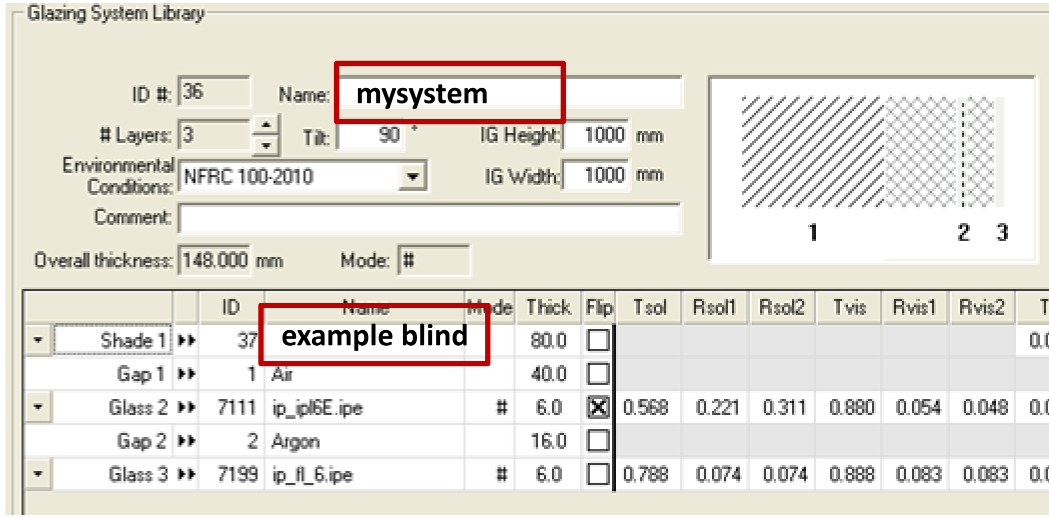


1. define shading layer

File → Preferences → Optical Calcs



2. define glazing system using the shading layer



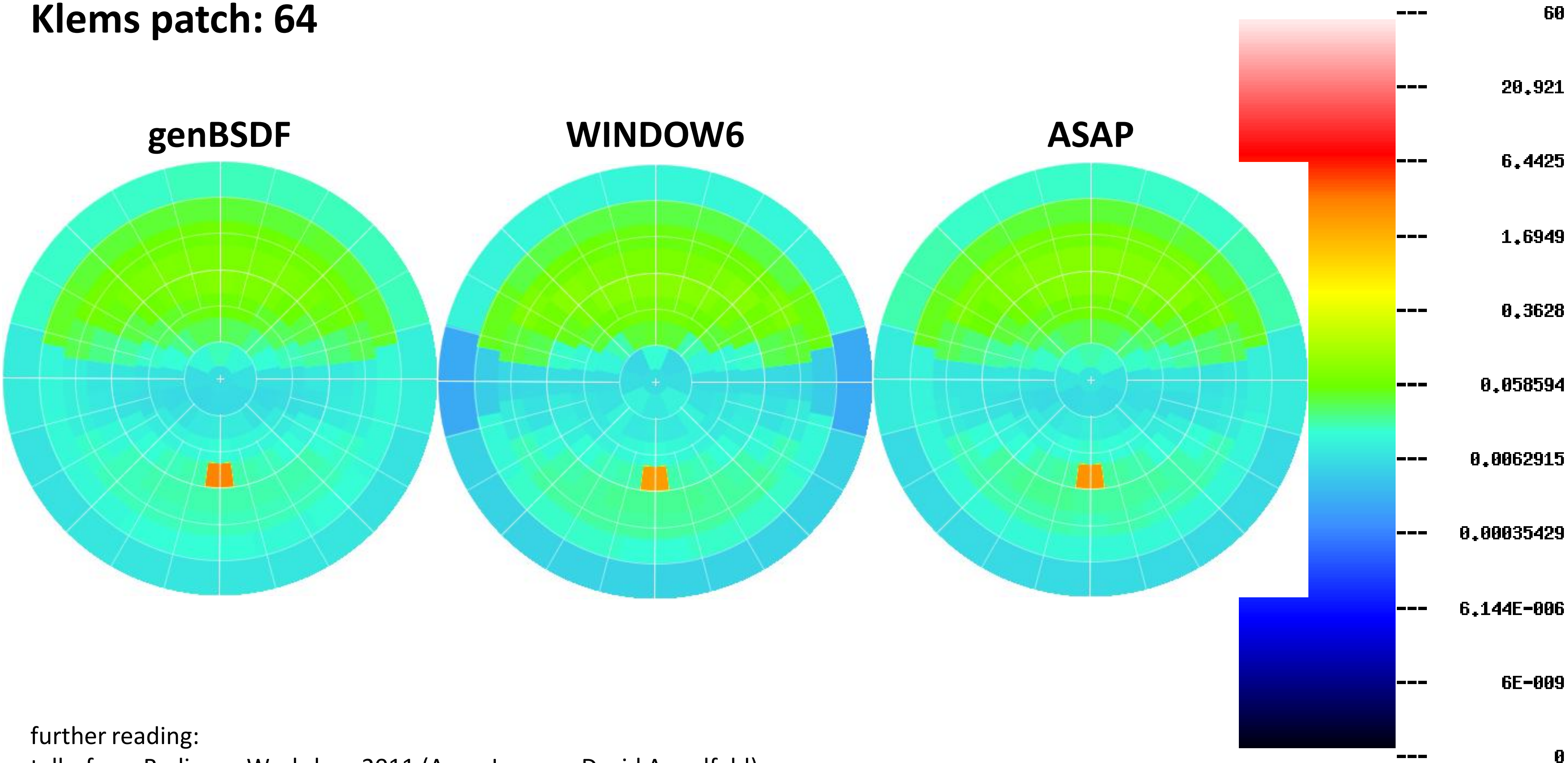
3. run calculation and pick up *mysystem.xml* at C:\Users\Public\LBNL\WINDOW6\

further reading:
talk from Radiance Workshop 2012 (Christian Kohler)

Bartenbach LichtLabor

genBSDF vs. WINDOW6 vs. ASAP

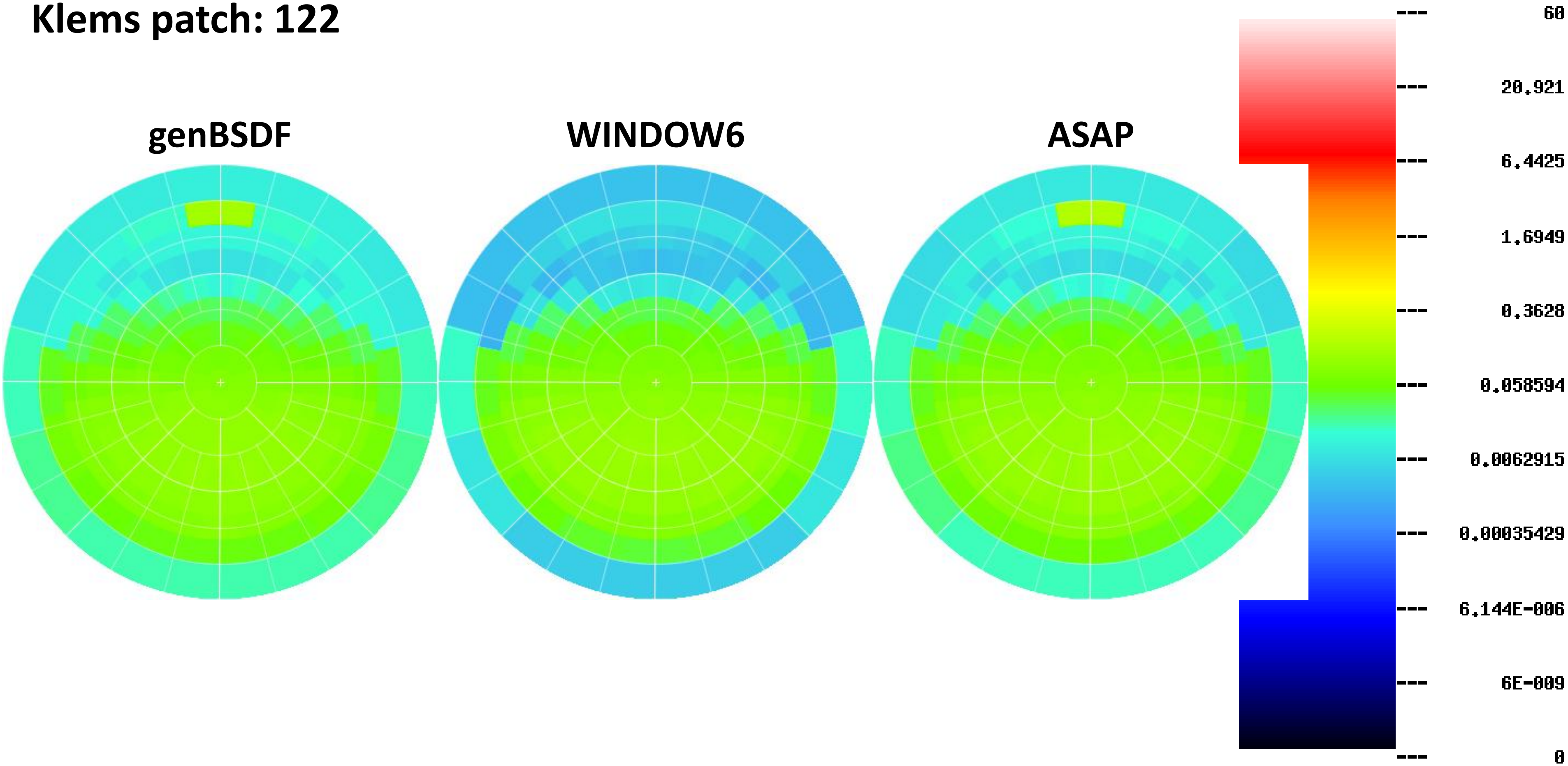
example blind
tilt angle: 0°
Klems patch: 64



further reading:
talks from Radiance Workshop 2011 (Anne Iversen, David Appelfeld)

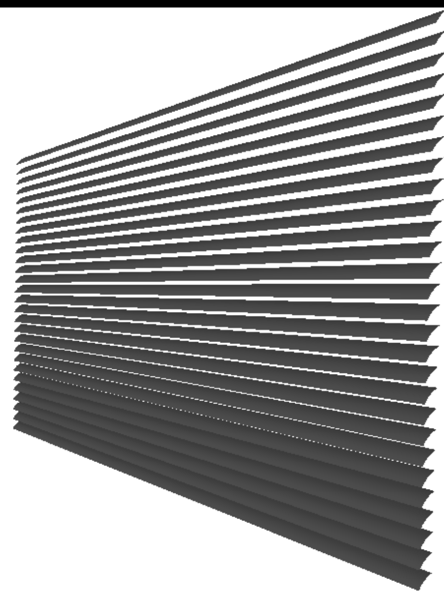
genBSDF vs. WINDOW6 vs. ASAP

example blind
tilt angle: 45°
Klems patch: 122

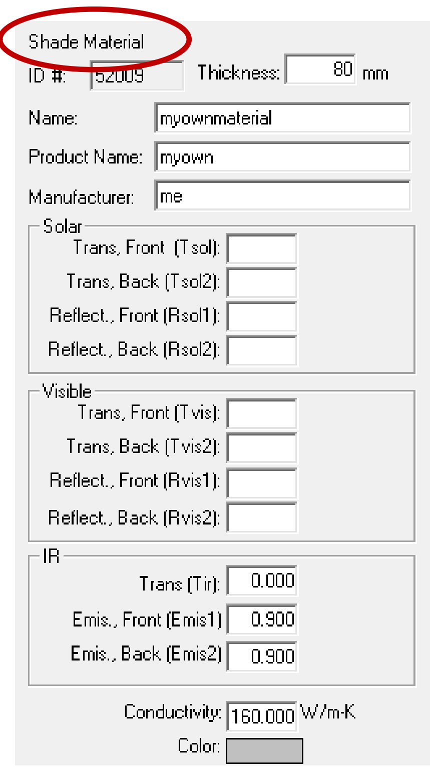


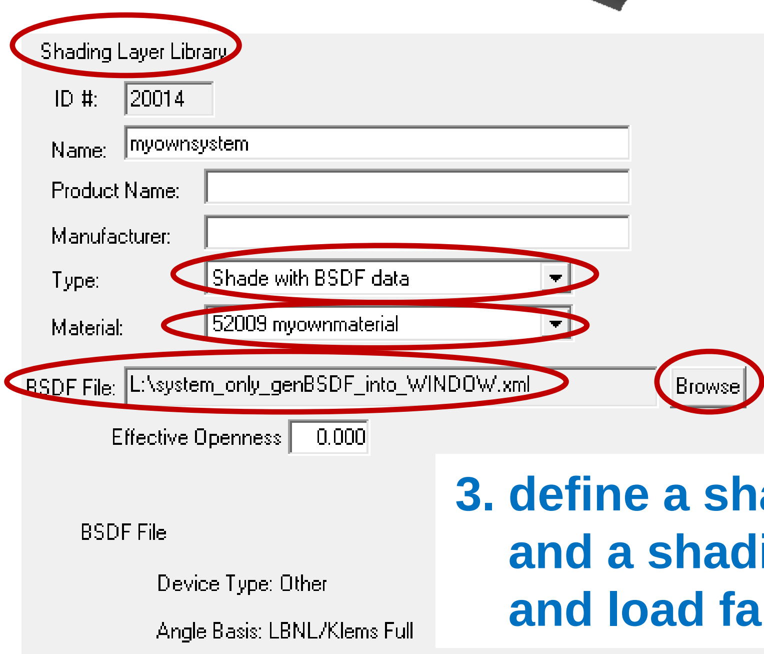
genBSDF + WINDOW

1. generate BSDF of **system only** with genBSDF +f +b



2. fake generated XML file
   - use XML file generated by WINDOW as template
   - fill "Visible" and "NIR" blocks with data from the XML file generated by genBSDF

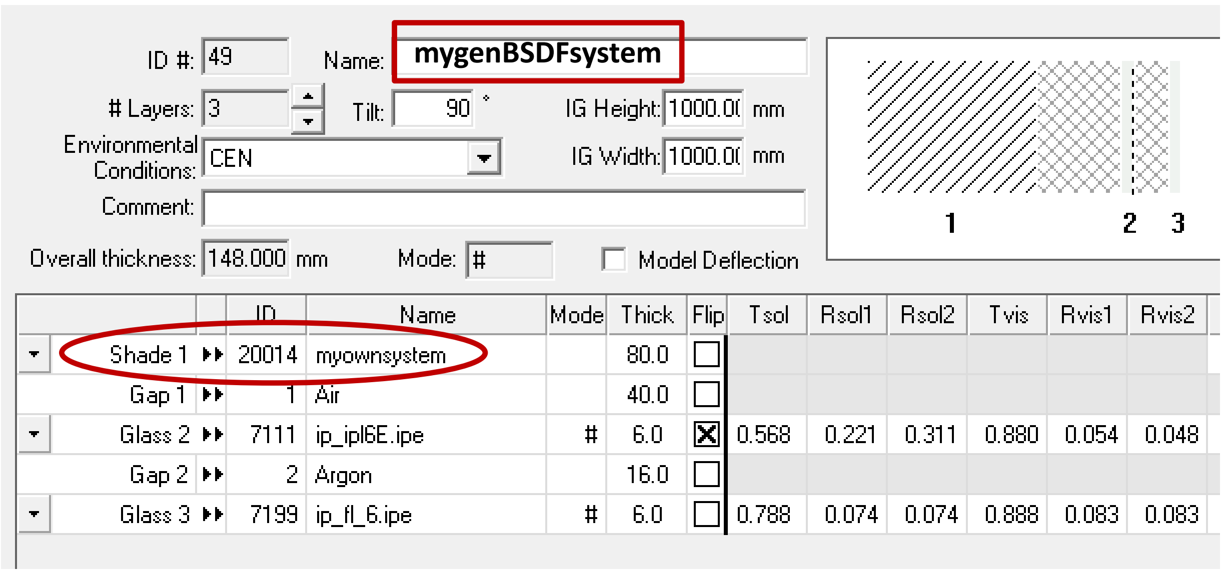




3. define a shade material (thickness) and a shading layer with BSDF data and load faked BSDF

genBSDF + WINDOW

4. define glazing system using the BSDF shading layer

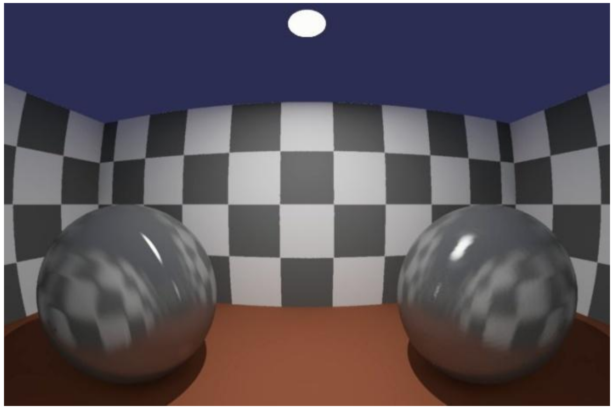


	ID	Name	Mode	Thick	Flip	Tsol	Rsol1	Rsol2	Tvis	Rvis1	Rvis2
Shade 1	20014	myownsystem		80.0	☐						
Gap 1	1	Air		40.0	☐						
Glass 2	7111	ip_ipl6E.ipe	#	6.0	☒	0.568	0.221	0.311	0.880	0.054	0.048
Gap 2	2	Argon		16.0	☐						
Glass 3	7199	ip_fl_6.ipe	#	6.0	☐	0.788	0.074	0.074	0.888	0.083	0.083

5. run calculation and pick up *mygenBSDFsystem.xml* at C:\Users\Public\LBNL\WINDOW7\

OUTLINE

- basics of BSDFs
  - theory
  - discretizations
- generating BSDFs
  - measurements
  - simulations
- **using BSDFs in RADIANCE**
  - **mkillum**
  - **BSDF material primitive**
- using BSDFs in the RADIANCE 3-phase method
- Q & A



BSDFs in mkillum

Greg announced at the workshop 2011:
("The BSDF as a First-class Citizen in Radiance")

- **mkillum** is still valuable as a means to improve rendering performance
- **mkillum** access to BSDF data will be removed in upcoming release
  - BSDF sampling is more general in rendering code
  - Incorporates reflection and variable-resolution data

thus

- just use it as usual (it is still valuable!) and
- include the BSDF via the material primitive in the scene

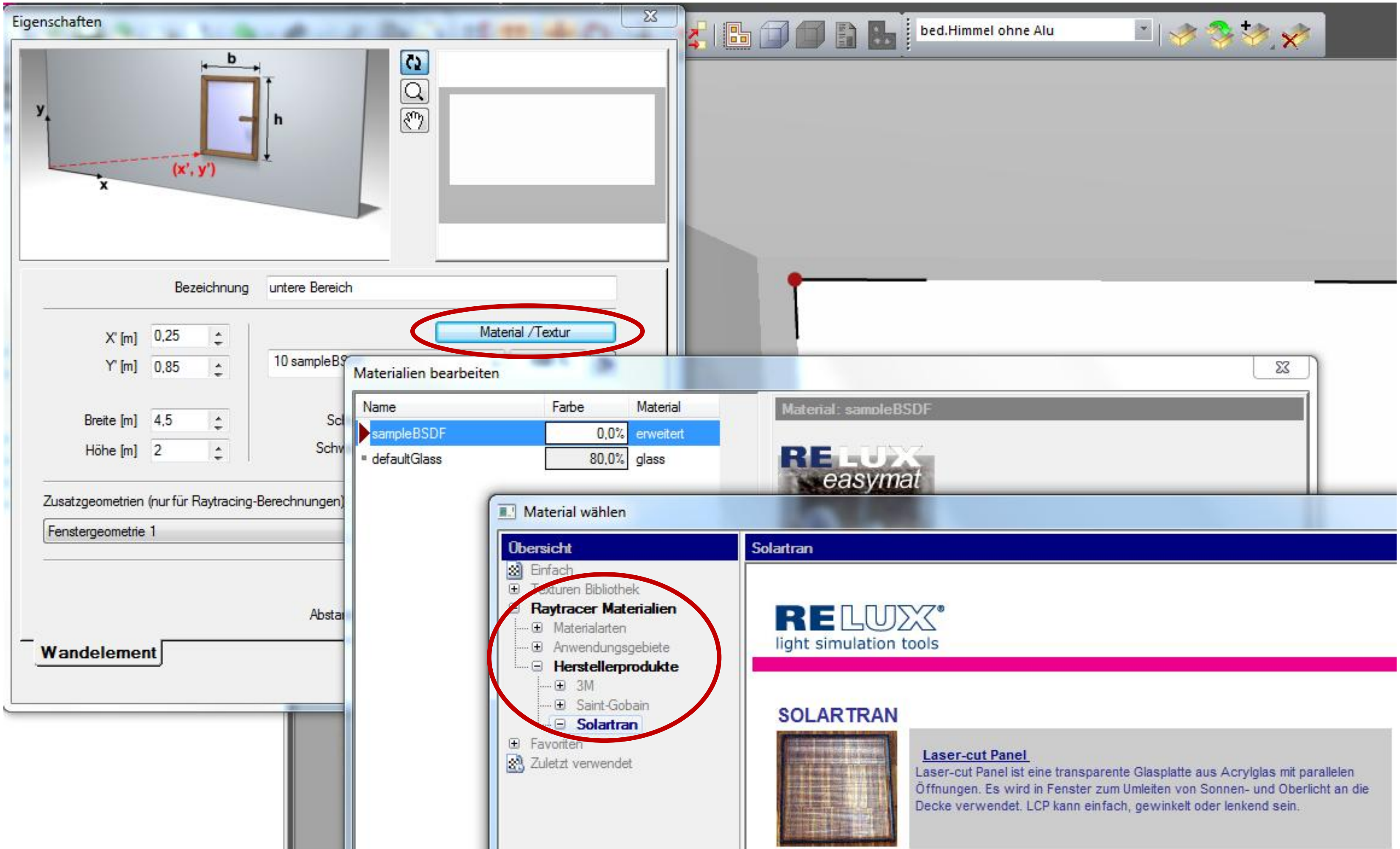
Lars Grobe will present more thoughts on the mkillum topic!

further reading:
Greg's talk "The BSDF as a First-class Citizen in Radiance" from Radiance Workshop 2011
Lars Grobe's talk from Radiance Workshop 2012

BSDFs in mkillum – used in RELUX

Carsten Bauer can tell you more…

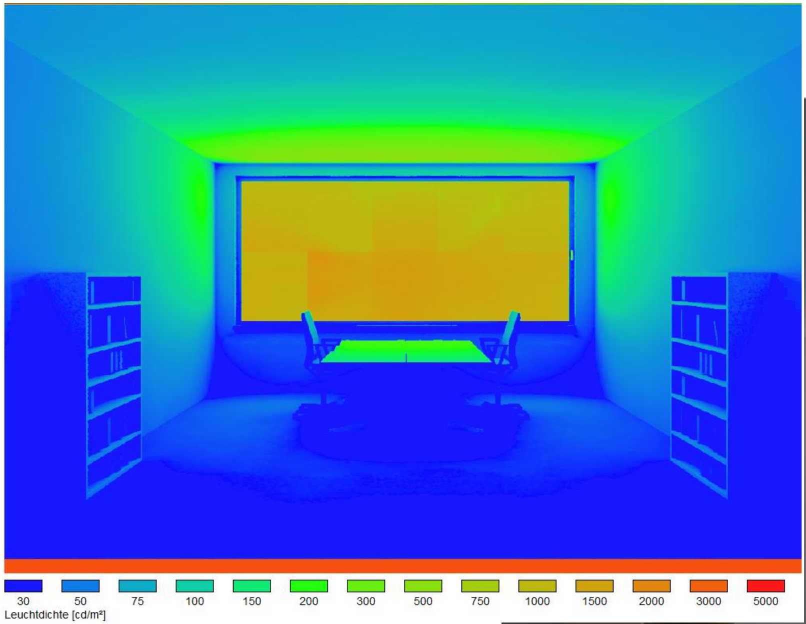
RELUX® light simulation tools

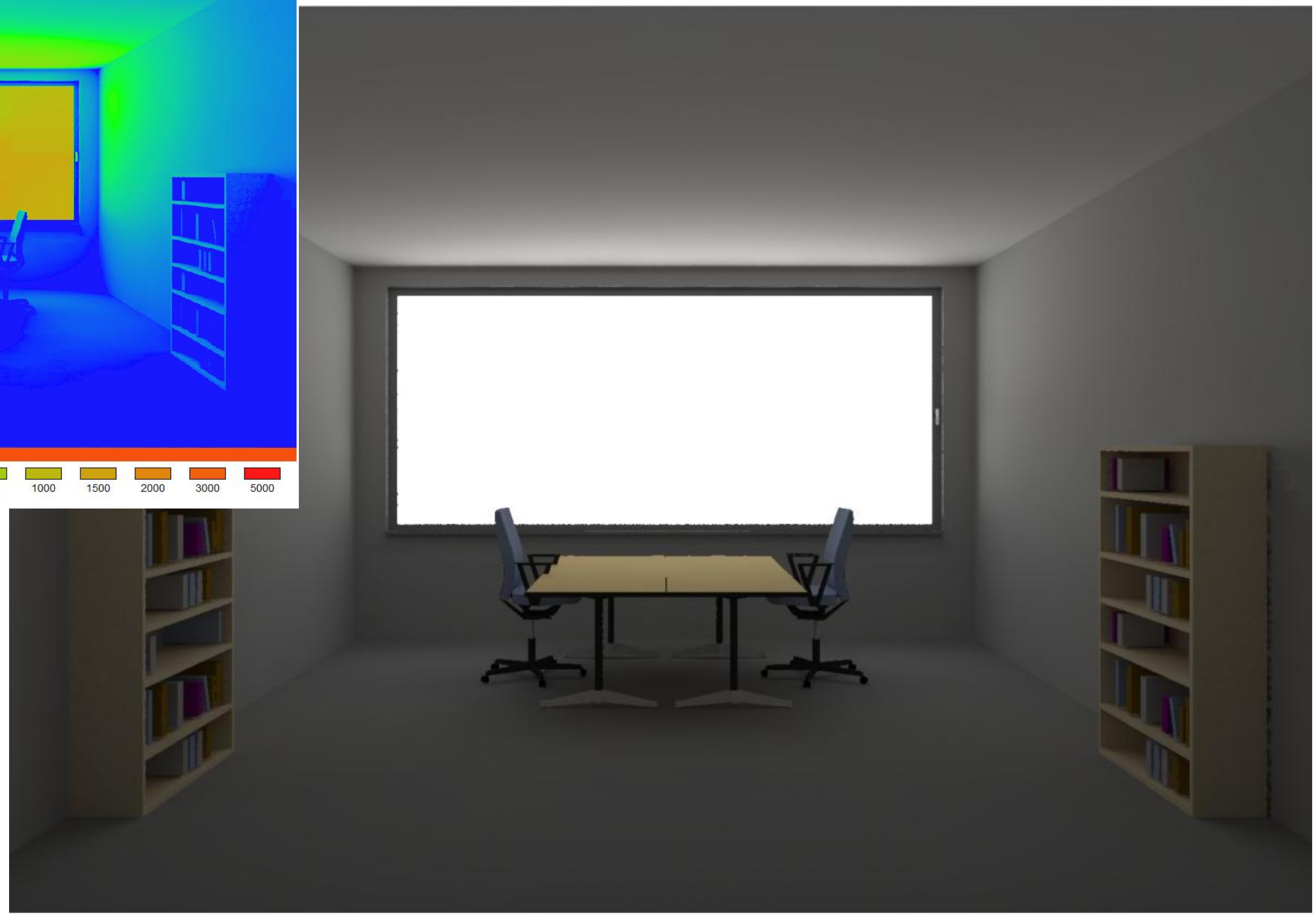


BSDFs in mkillum – used in RELUX

Carsten Bauer can tell you more…

RELUX® light simulation tools





BSDF material primitive

```
void BSDF material_name
6+ thickness system.xml up_x up_y up_z funcfile transform
0
0|3|6|9 rdf gdf bdf
        rdb gdb bdb
        rdt gdt bdt
```

thickness	0 for BSDF surface != 0 for ignoring BSDF for view/shadow rays
system.xml	BSDF XML file containing scattering data
up_x up_y up_z	up-vector for BSDF-data (+y in genBSDF)
funcfile	function file for up-vector (or . if none)
transform	transform of BSDF data (e.g. rotate with -rz α)
rdf gdf bdf	*additional* diffuse front reflection (RGB)
rdb gdb bdb	*additional* diffuse back reflection (RGB)
rdt gdt bdt	*additional* diffuse transmission (RGB)

further reading:
Greg's talk "The BSDF as a First-class Citizen in Radiance" from Radiance Workshop 2011

BSDF material primitive - example

reference material

```
void plastic2 ptest2_20_01_10
4 0 1 0 .
0
6 .1 .1 .1 .2 .01 .10
```

variable resolution BSDF

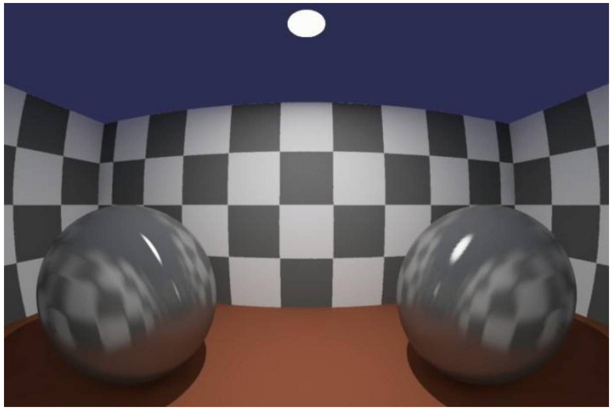
```
genBSDF -t4 6 -c 40960 +b -f -r "-ss 64" -t xx
```

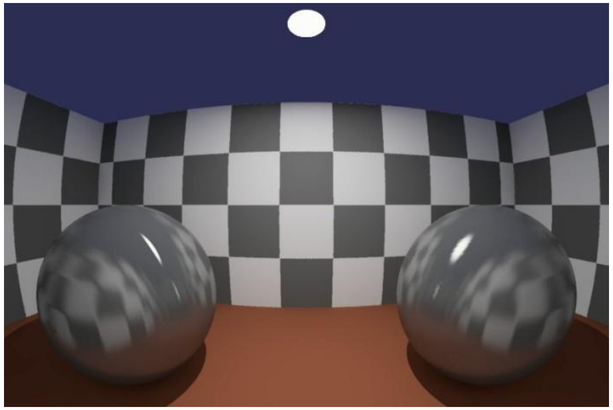
with varying degree of data reduction

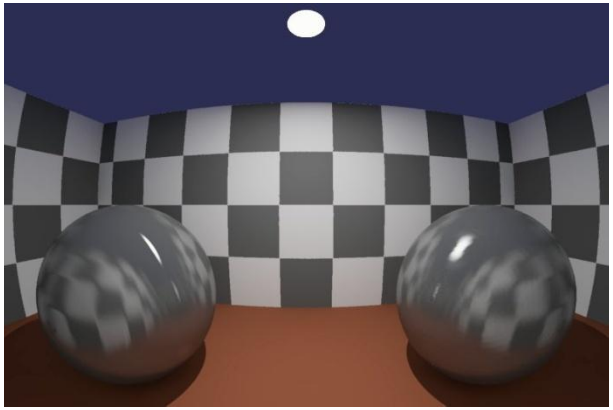
-t 0 / 95 / 99

BSDF material

```
void BSDF mat
8 0 ptest2_20_01_10.xml 0 1 0 . -rz 0
0
0
```





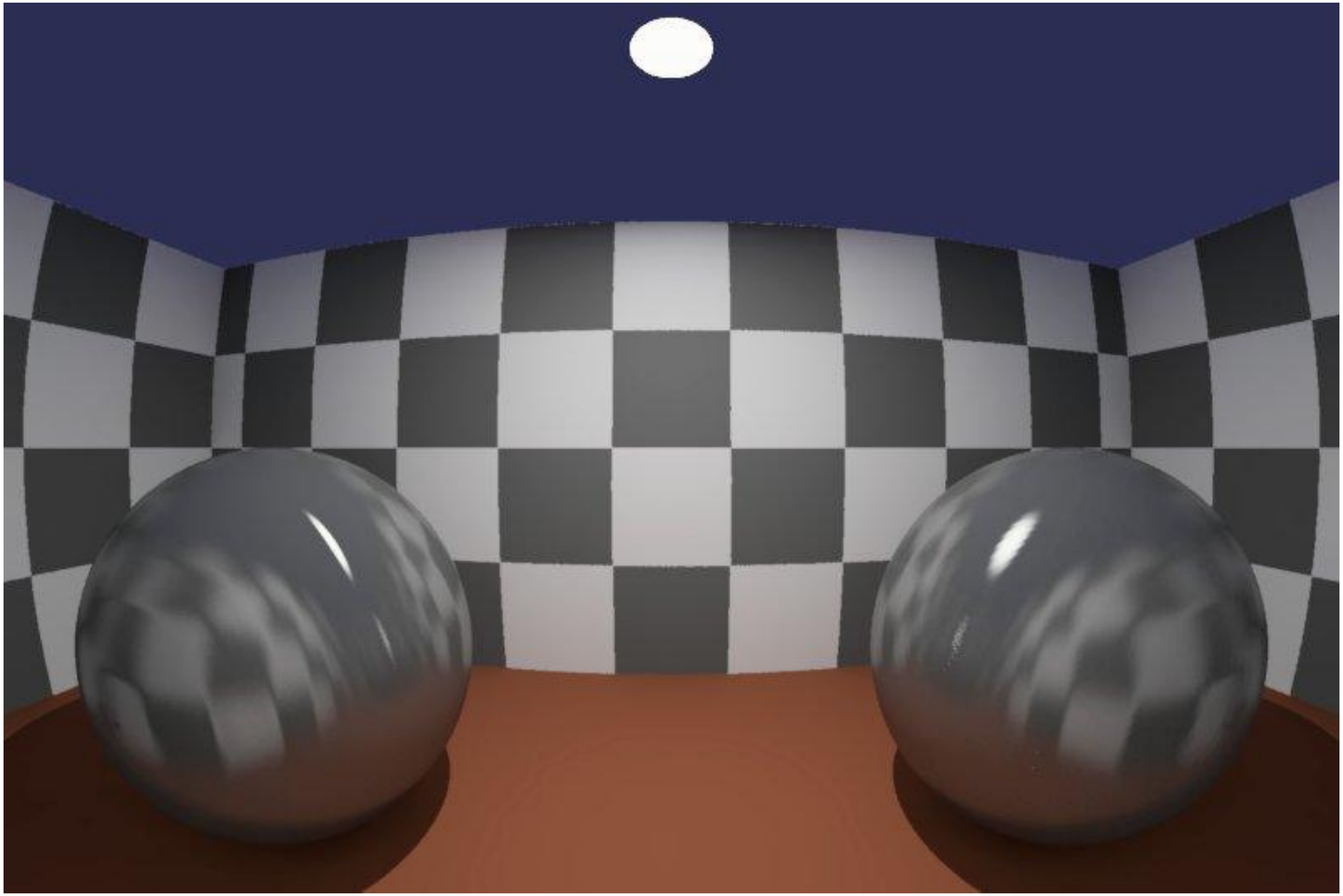


images

left: reference material

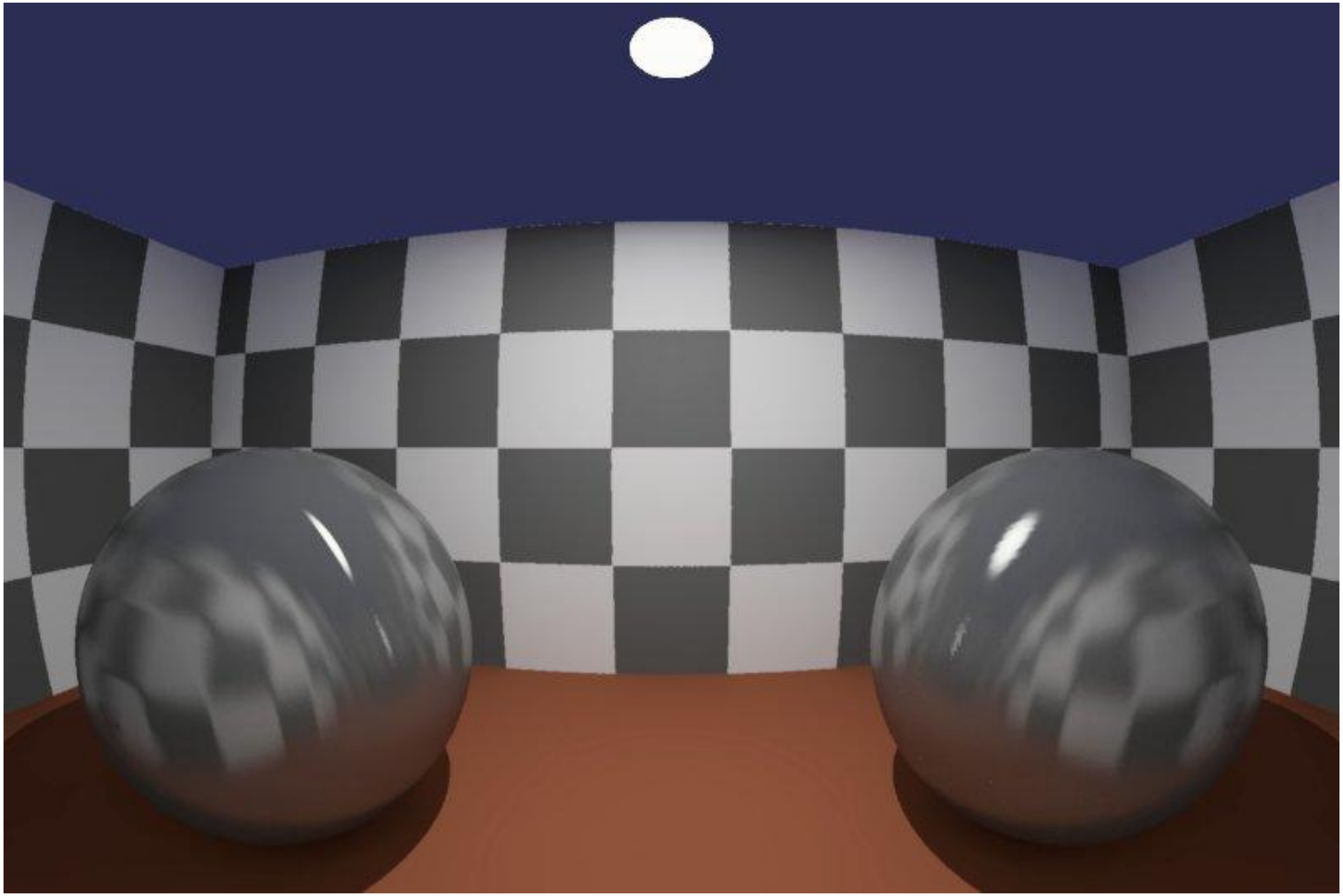
right: BSDF material

BSDF material primitive - example



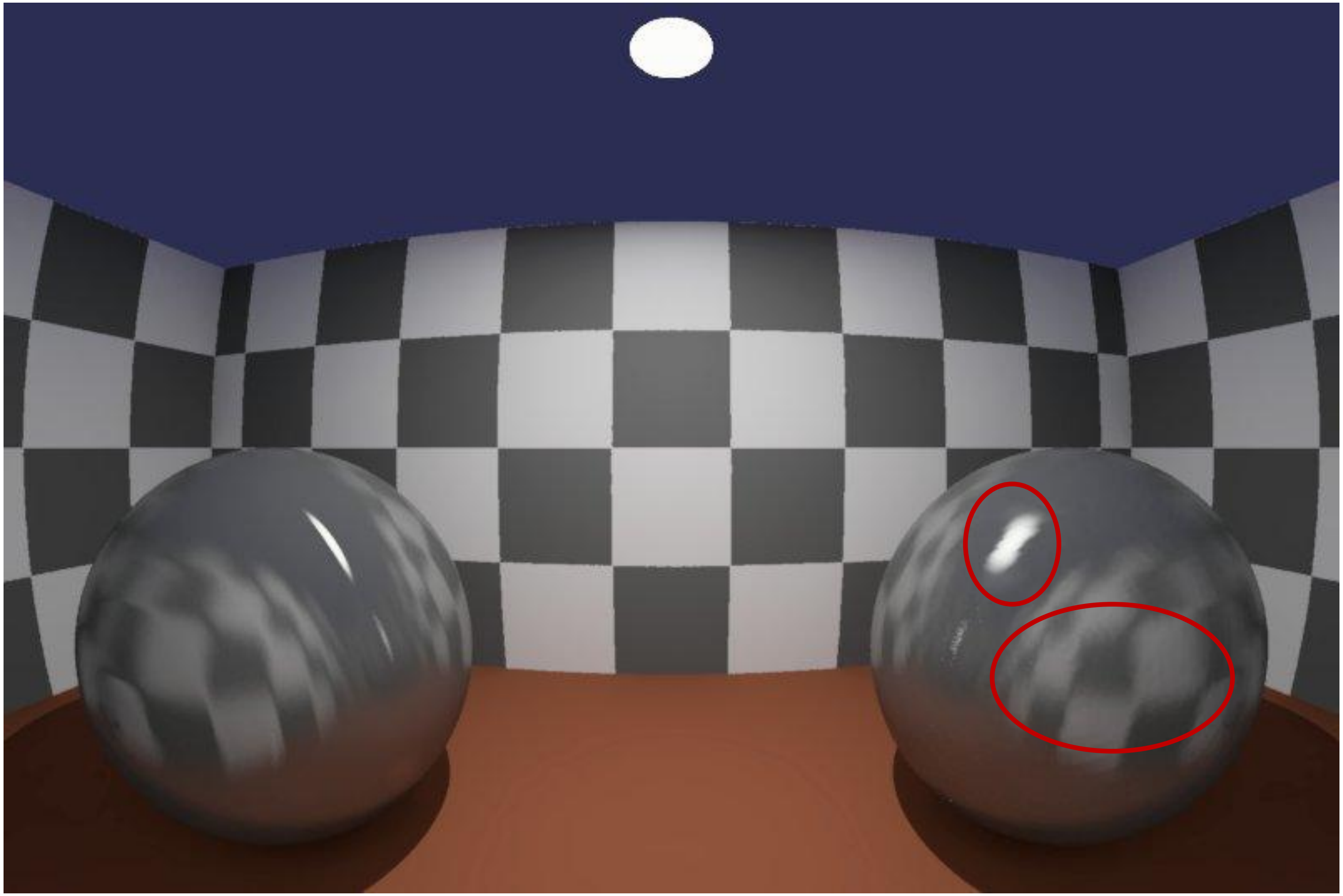
left: reference plastic2 material right: BSDF material, 0% reduction (full data, 238M)

BSDF material primitive - example



left: reference plastic2 material right: BSDF material, 95% reduction (5% data, 12M)

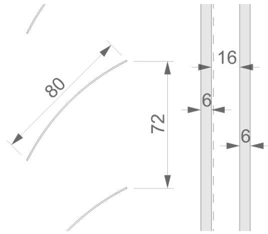
BSDF material primitive - example



left: reference plastic2 material right: BSDF material, 99% reduction (1% data, 3.9M)

test scene

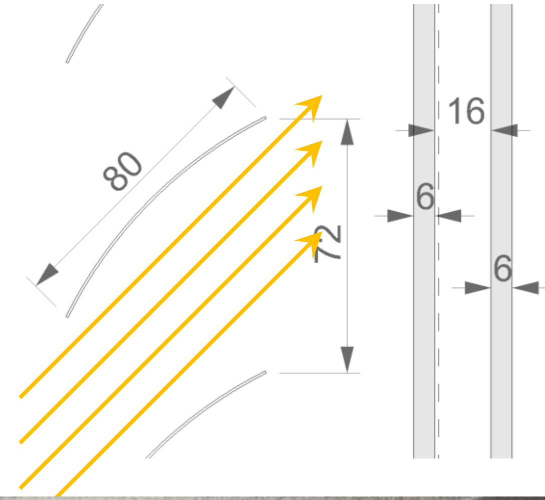
BSDF material primitive for the example blind



cie clear sky in innsbruck, september 21, 09:00 (γ = 27.9°, ϕ = -55.8°)

```
!gensky 9 21 9:00 -a 47.27 -o -11.39 -m -15 +s


void BSDF mat_fenestration
6 0 system.xml 0 0 1 .
0
0

mat_fenestration polygon window
0
0
12 -2.25 -2.7 0.85
   -2.25 -2.7 2.85
    2.25 -2.7 2.85
    2.25 -2.7 0.85
```

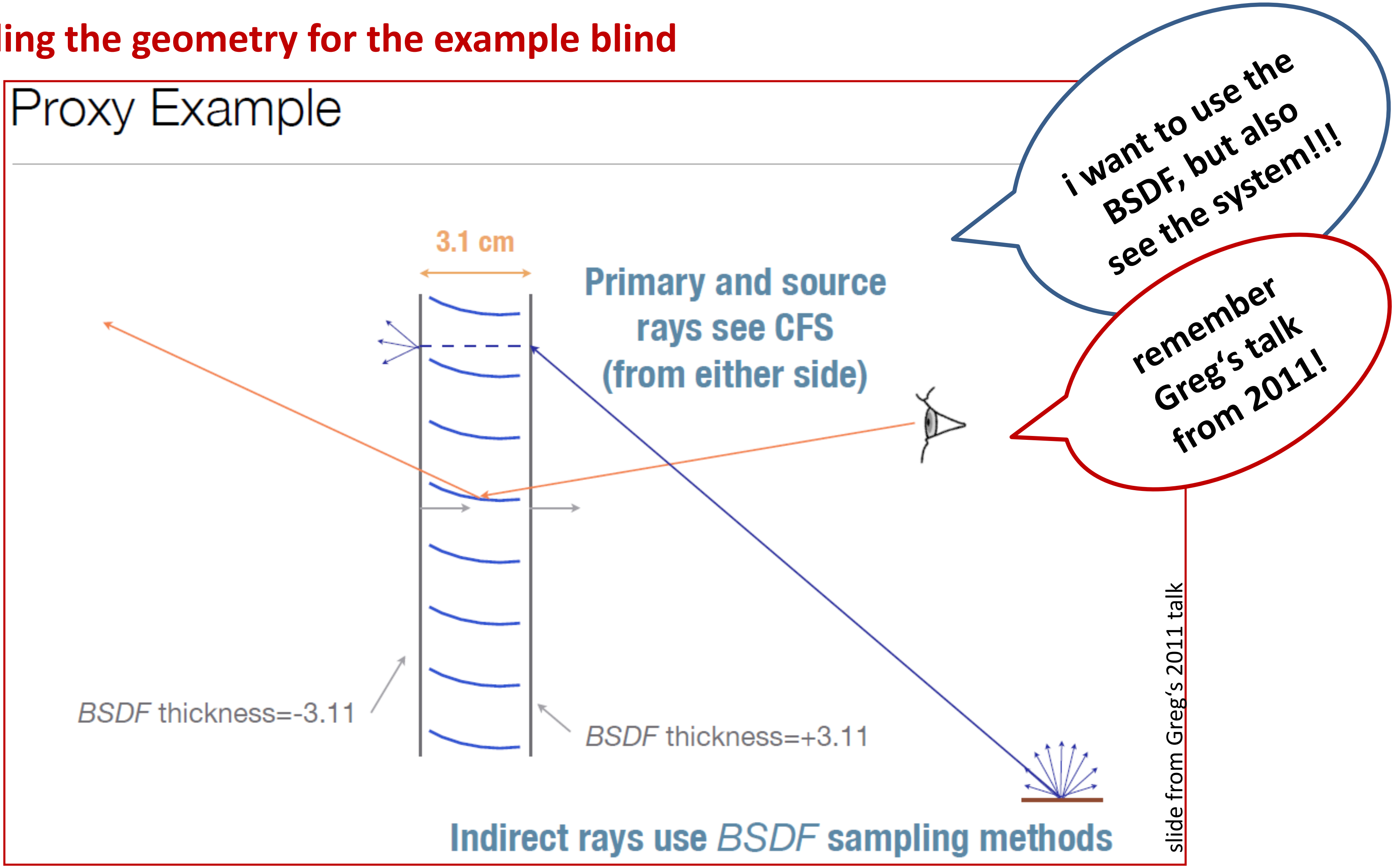
BSDF material primitive for the example blind



fake "sun" at direction (1, -1, -1), i.e. γ = -35.26°, ϕ = -45°
(sun profile angle ε = -45°)

```
void light solar 0 0 3 1e+06 1e+06 1e+06
solar source sun 0 0 4 1 -1 -1 0.5


void BSDF mat_fenestration
6 0 system.xml 0 0 1 .
0
0

mat_fenestration polygon window
0
0
12 -2.25 -2.7 0.85
   -2.25 -2.7 2.85
    2.25 -2.7 2.85
    2.25 -2.7 0.85
```

adding the geometry for the example blind



further reading:
Greg's talk "The BSDF as a First-class Citizen in Radiance" from Radiance Workshop 2011

Bartenbach LichtLabor

adding the geometry for the example blind

just specify the following in the rad-file (the `xform`-command places the window properly)

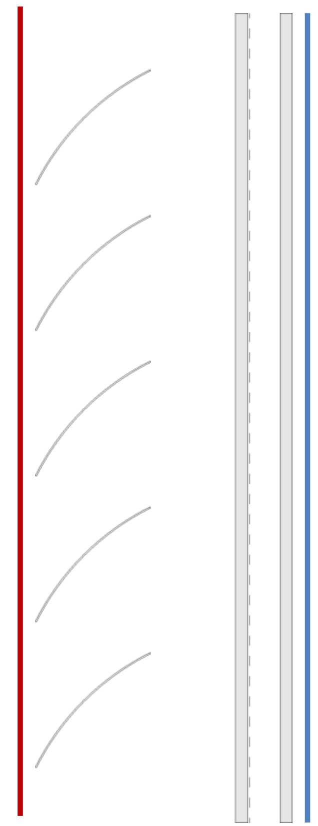
```
!pkgBSDF -s system.xml | xform -rx 90 -rz 180 -t -0.0 -2.7 1.85
```

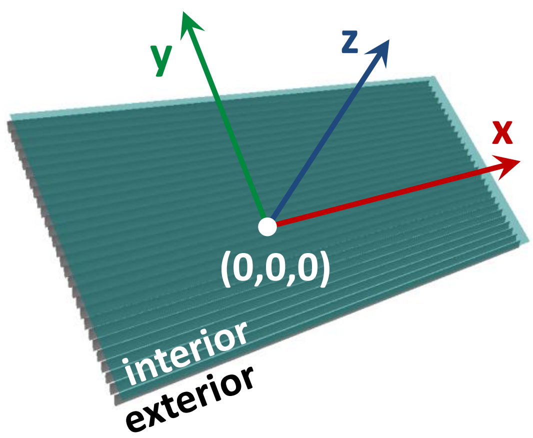
pkgBSDF provides you with

- a BSDF surface at the front that is x-y-centered at (0,0) and max(z) = 0

and – if geometry is included in system.xml (remember +geom meter) –

- a BSDF surface at the back
- detailed geometry of the whole system as used in genBSDF





adding the geometry for the example blind

```
!pkgBSDF -s system.xml | xform -rx 90 -rz 180 -t -0.0 -2.7 1.85
```

returns

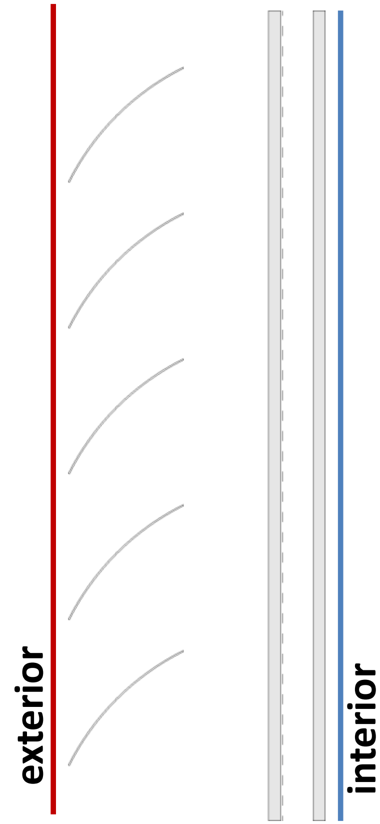
- a BSDF surface at the front

```
void BSDF m_system_f
16 0.136005 system.xml 0 1 0 . -i 1 -rx 90 -rz 180 -t -0.0 -2.7 1.85
0
0
m_system_f polygon system_f
0
0
12 ...
```

- a BSDF surface at the back

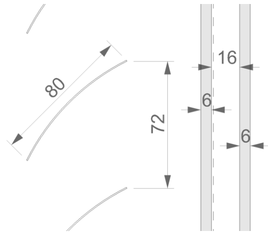
```
void BSDF m_system_b
16 -0.136005 system.xml 0 1 0 . -i 1 -rx 90 -rz 180 -t -0.0 -2.7 1.85
0
0
m_system_b polygon system_b
0
0
12 ...
```

- detailed geometry of the whole system as used in genBSDF generated by `mgf2rad` from the date in the XML-header



Bartenbach L'chtLabor

BSDF material primitive for the example blind with geometry



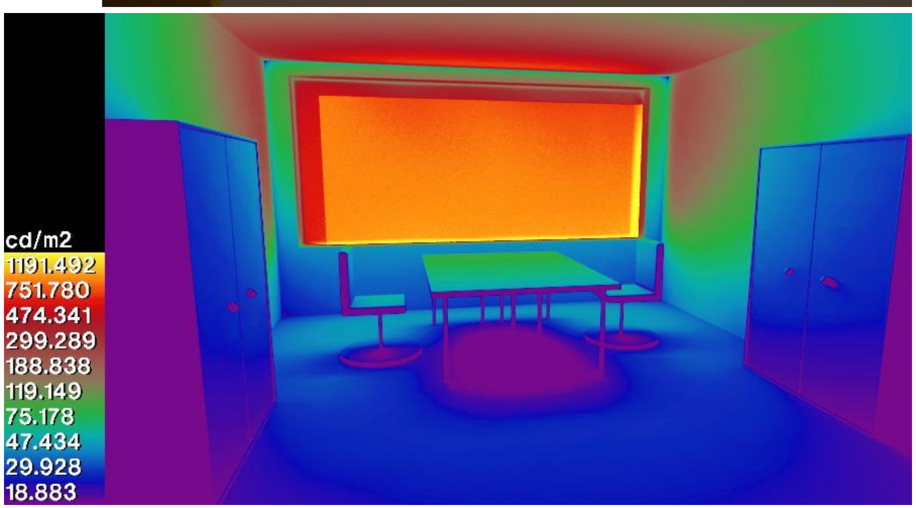
cie clear sky in innsbruck, september 21, 09:00 (γ = 27.9°, ϕ = -55.8°)

```
!gensky 9 21 9:00 -a 47.27 -o -11.39 -m -15 +s

!pkgBSDF –s system.xml | xform -rx 90 -rz 180 -t -0.0 -2.7 1.85
```

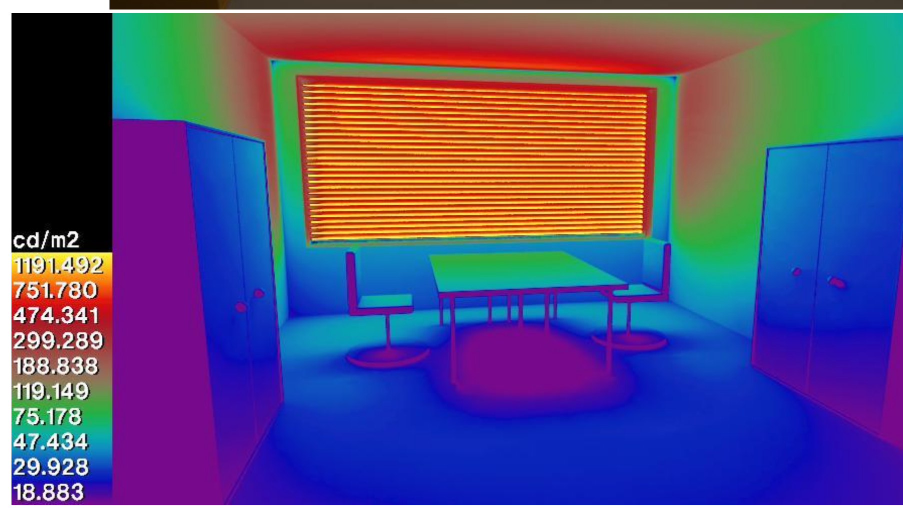
Bartenbach LichtLabor

BSDF material primitive for the example blind

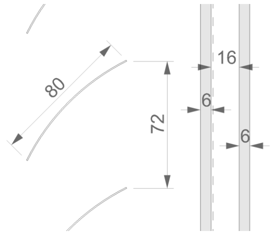
without geometry



with geometry



what about the direct sun?



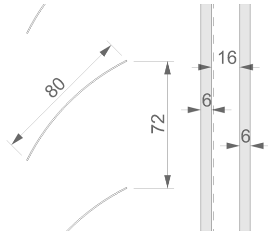
cie clear sky in innsbruck, december 21, 09:00 (γ = 7.2°, ϕ = -43.3°)

```
!gensky 12 21 9:00 -a 47.27 -o -11.39 -m -15 +s
```

```
void BSDF mat_fenestration
6 0 system.xml 0 0 1 .
0
0

mat_fenestration polygon window
0
0
12 -2.25 -2.7 0.85
   -2.25 -2.7 2.85
    2.25 -2.7 2.85
    2.25 -2.7 0.85
```

splotchy shadow edges
due to indirect calculation

what about the direct sun?

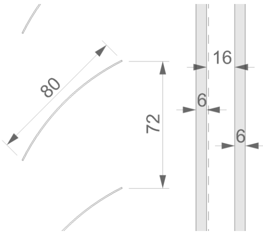


cie clear sky in innsbruck, december 21, 09:00 (γ = 7.2°, ϕ = -43.3°)

```
!gensky 12 21 9:00 -a 47.27 -o -11.39 -m -15 +s


void BSDF mat_fenestration
6 0 system.xml 0 0 1 .
0
0

mat_fenestration polygon window
0
0
12 -2.25 -2.7 0.85
   -2.25 -2.7 2.85
    2.25 -2.7 2.85
    2.25 -2.7 0.85
```

switching off indirect calculation (-aa 0) removes splotches but introduces some noise

what about the direct sun?



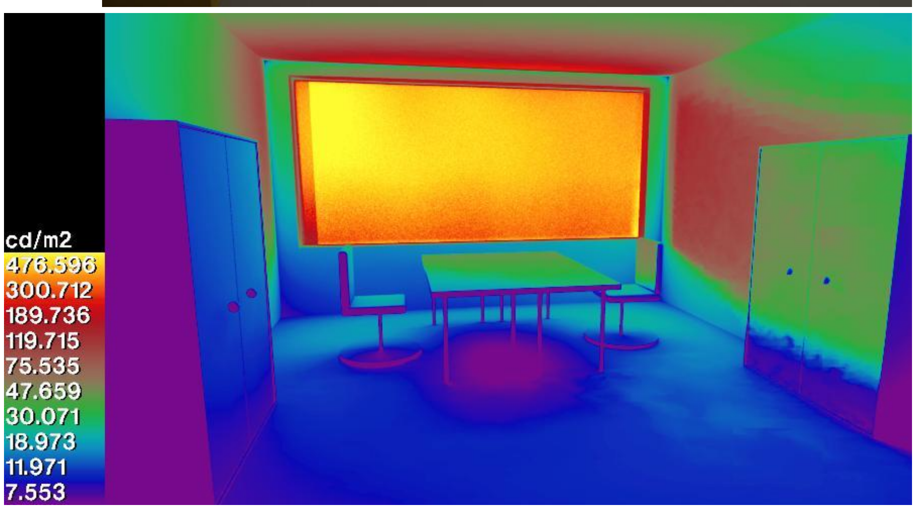
cie clear sky in innsbruck, december 21, 09:00 (γ = 7.2°, ϕ = -43.3°)

```
!gensky 12 21 9:00 -a 47.27 -o -11.39 -m -15 +s

!pkgBSDF –s system.xml | xform -rx 90 -rz 180 -t -0.0 -2.7 1.85
```

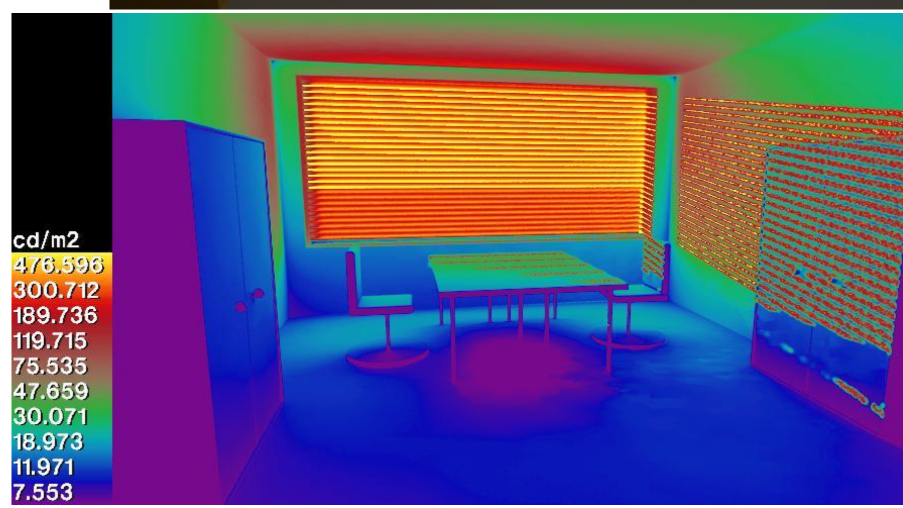
using the geometry helps, since the direct part is now treated separately

BSDF material primitive for the example

without geometry



with geometry



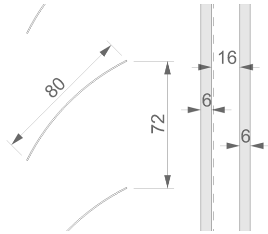
BSDF material primitive

```
void BSDF material_name
6+ thickness system.xml up_x up_y up_z funcfile transform
0
0|3|6|9 rdf gdf bdf
        rdb gdb bdb
        rdt gdt bdt
```

thickness	0 for BSDF surface >0 / <0 for ignoring BSDF for view rays
system.xml	BSDF XML file containing scattering data
up_x up_y up_z	up-vector for BSDF-data (+y in genBSDF)
funcfile	function file for up-vector (or . if none)
transform	transform of BSDF data (e.g. rotate with -rz α)
rdf gdf bdf	*additional* diffuse front reflection (RGB)
rdb gdb bdb	*additional* diffuse back reflection (RGB)
rdt gdt bdt	*additional* diffuse transmission (RGB)

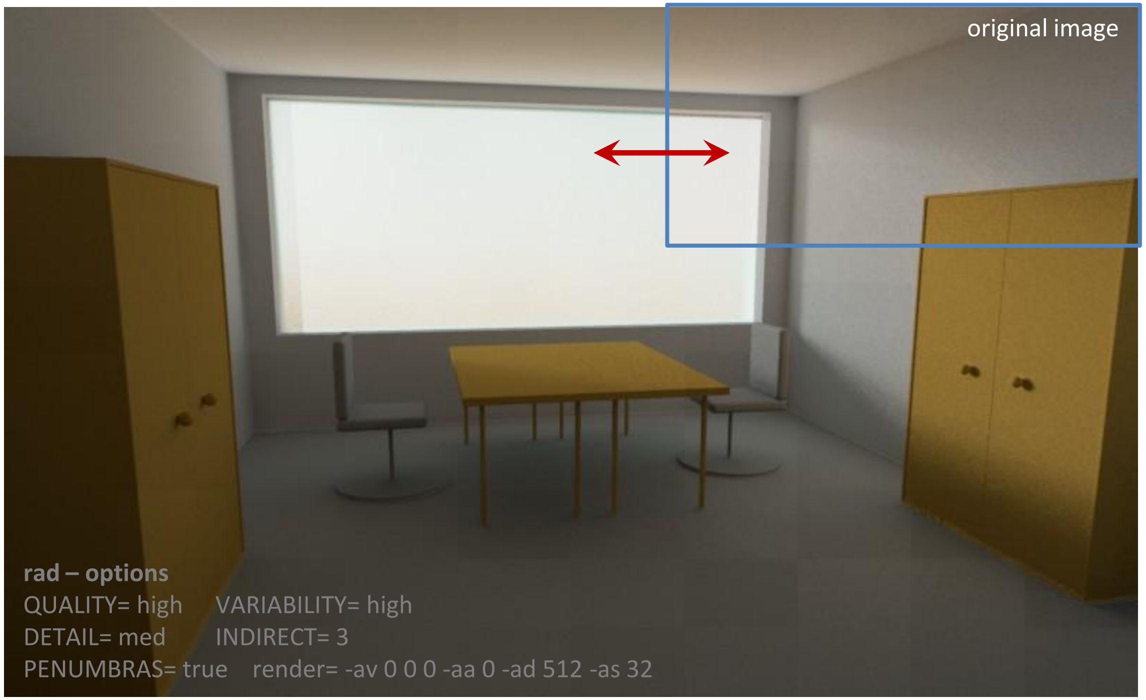
further reading:
Greg's talk "The BSDF as a First-class Citizen in Radiance" from Radiance Workshop 2011

adding some diffuse reflection



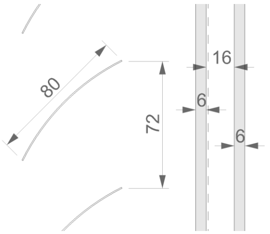
cie clear sky in innsbruck, december 21, 09:00 (γ = 7.2°, ϕ = -43.3°)

```
!gensky 12 21 9:00 -a 47.27 -o -11.39 -m -15 +s


void BSDF mat_fenestration
6 0 system.xml 0 0 1 .
0
3 0 0.5 0.5

mat_fenestration polygon window
0
0
12 -2.25 -2.7 0.85
   -2.25 -2.7 2.85
    2.25 -2.7 2.85
    2.25 -2.7 0.85
```



notice the slightly cyan-colored window

adding some diffuse transmission



cie clear sky in innsbruck, december 21, 09:00 (γ = 7.2°, ϕ = -43.3°)

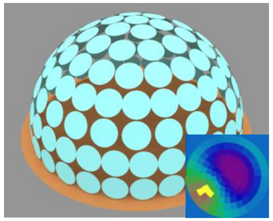
```
!gensky 12 21 9:00 -a 47.27 -o -11.39 -m -15 +s
```

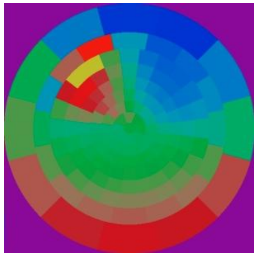
```
void BSDF mat_fenestration
6 0 system.xml 0 0 1 .
0
9 0 0 0
  0 0 0
  0.5 0.0 0.5

mat_fenestration polygon window
0
0
12 -2.25 -2.7 0.85
   -2.25 -2.7 2.85
    2.25 -2.7 2.85
    2.25 -2.7 0.85
```

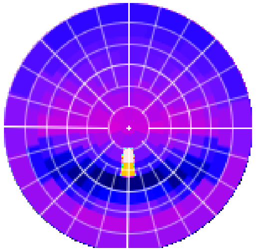
can you see any difference??? ☺

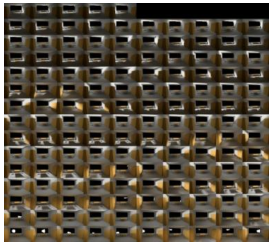
OUTLINE

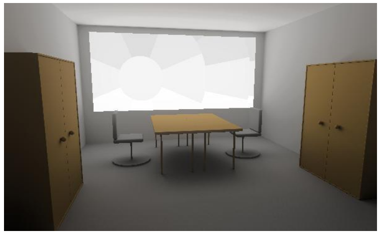
- basics of BSDFs
  - theory
  - discretizations
- generating BSDFs
  - measurements
  - simulations
- using BSDFs in RADIANCE
  - mkillum
  - BSDF material primitive
- **using BSDFs in the RADIANCE 3-phase method**
- Q & A

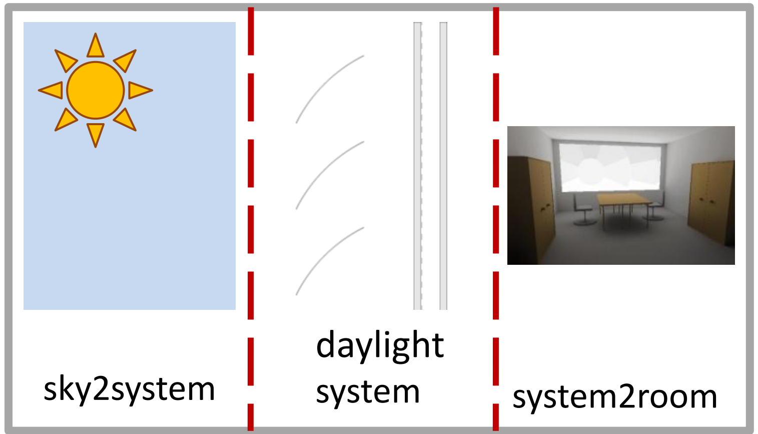












phase 1:
daylight matrix DMX

sky → exterior of daylight system

phase 2:
BSDF

$f(\theta_l,\phi_l;\theta_v,\phi_v)$

phase 3:
view matrix VMX

interior of daylight system → room

Calculation of results

L-distribution sky ➡ DMX / BSDF / VMX ➡ illuminance grid / image

further reading: Greg's talks from 2009 and 2010

A.McNeil: "The Three-Phase Method for Simulating Complex Fenestration with Radiance", online

A.McNeil, E.S.Lee: „A validation of the Radiance three-phase simulation method for modeling annual daylight performance of optically-complex fenestration systems", Journal of Building Performance Simulation

RADIANCE 3-phase daylight coefficient method

$$\begin{bmatrix}1\\ \vdots\\ n\end{bmatrix} = \begin{pmatrix} 1,1 & \cdots & 1,145 \\ \vdots & \ddots & \vdots \\ n,1 & \cdots & n,145\end{pmatrix} \begin{pmatrix} 1,1 & \cdots & 1,145 \\ \vdots & \ddots & \vdots \\ 145,1 & \cdots & 145,145\end{pmatrix} \begin{pmatrix} 1,1 & \cdots & 1,2305 \\ \vdots & \ddots & \vdots \\ 145,1 & \cdots & 145,2305\end{pmatrix} \begin{bmatrix}1\\ \vdots\\ 2305\end{bmatrix}$$

R	VMX	BSDF	DMX	S
(n x 1)	(n x 145)	(145 x 145)	(145 x 146/578/2306)	(146/578/2306 x 1)

R .... result: illuminance and luminance values

VMX ..... view matrix: contribution of every Klems‘ patch from the interior side of the daylight system (145) to every measurement point

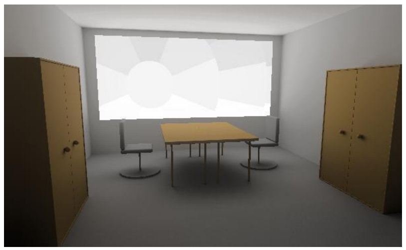
BSDF ... bidirectional scattering distribution function: function describing the properties of the daylight system (only transmission considered, no var. resolution)

DMX .... daylight matrix: contribution of every Tregenza / Reinhart sky patch (145 / 577 / 2305) and 1 ground patch to every Klems’ patch at the exterior side of the daylight system

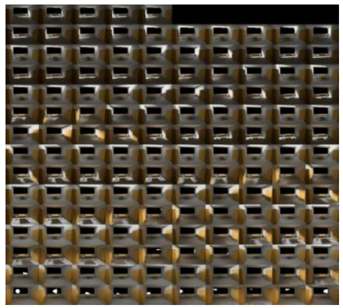
S .... sky vector: luminance of every single Tregenza / Reinhart sky patch

Simulation:	**pre-calculation:**	**VMX, BSDF, DMX**
	every time step:	**S, R (= matrix multiplication)**

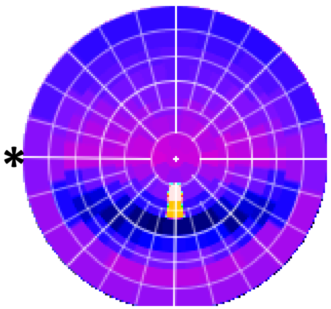
RADIANCE 3-phase daylight coefficient method



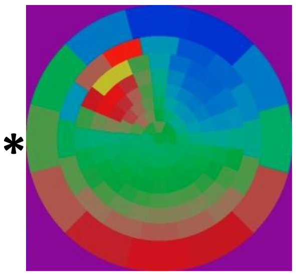
=



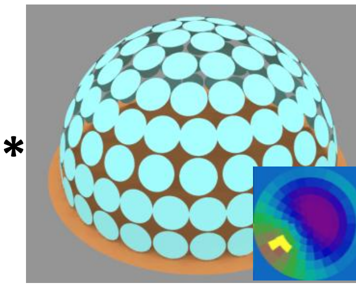
*



*



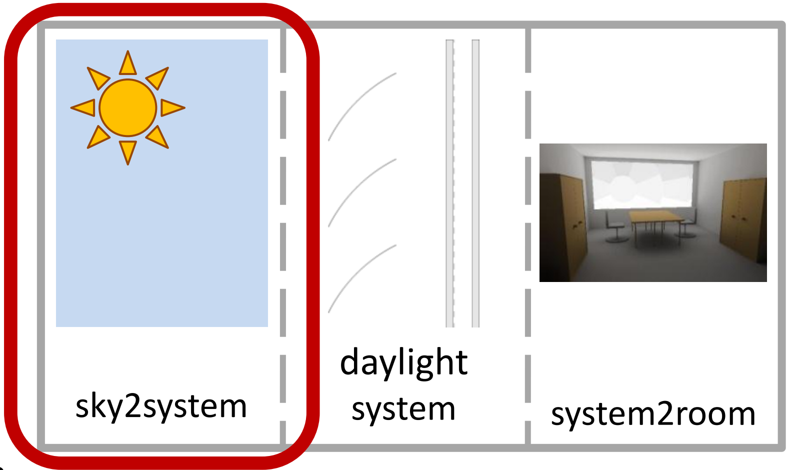
*



result — view matrix VMX — BSDF — daylight matrix DMX — sky radiance distribution

pre-calculation (time-consuming)

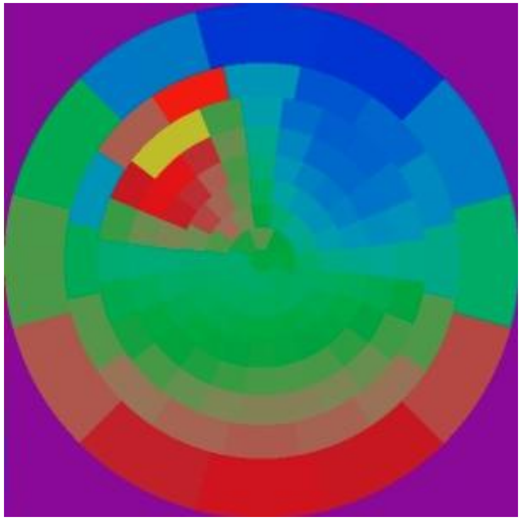
at every time step (fast matrix calculation)



phase1: daylight matrix DMX

contribution of sky part to the exterior of the daylight system

Structure
sky: Tregenza-/
Reinhart-patches
facade: Klems-patches
calculation
genklemsamp and *rtcontrib*



outgoing DMX distribution for a given sky
(i.e. incident distribution on the system)

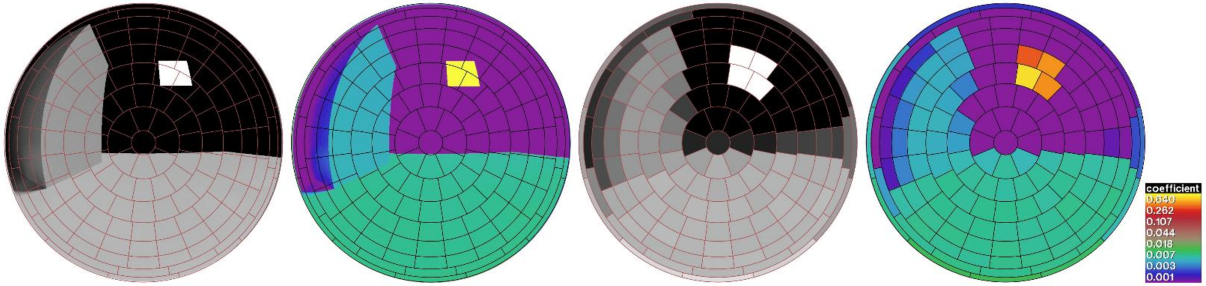


Figure 7. Renderings of contributions from Tregenza patch 74 (left two images) and visualizations of the D matrix coefficients for the same Tregenza patch (right two images). Reflections from model geometry (ground polygon and adjacent building) are included in the D matrix.

(image from Andy McNeil's tutorial)

further reading:
Andy McNeil: "The Three-Phase Method for Simulating Complex Fenestration with Radiance", sites.google.com/a/lbl.gov/andy-radiance/

phase1: daylight matrix DMX

example calculation

```
genklemsamp -c 3000 -vd 0 -1 0 -ff window_glow.rad | \
rtcontrib -n 8 -ab 3 -ad 3000 -lw 1e-9 -c 3000
          -e MF:4 -f reinhart.cal -b rbin -bn Nrbins
          -m sky_glow -fff scene_dmx.oct > south.dmx
```

window_glow.rad

```
void glow windowglow
0
0
4 1 1 1 0
windowglow polygon window
0
0
12 -2.25 -2.7 0.85
   -2.25 -2.7 2.85
    2.25 -2.7 2.85
    2.25 -2.7 0.85
```

genklemsamp program to generate Klems samples

-c: number of sample rays per Klems patch (must match)

-e: define sky subdivision
1: Tregenza (145+1)
2/4: Reinhart (577/2305+1)

-f: cal-file that calculates *rbin* and *Nrbins*

-b: current bin number

-bn: total number of bins

scene_dmx.oct
octree that contains the scene and a uniform sky with the modifier for r(t)contrib

in sky1.rad:

```
void glow sky_glow
0
0
4 1 1 1 0
sky_glow source sky1
0
0
4 0 0 1 360
```

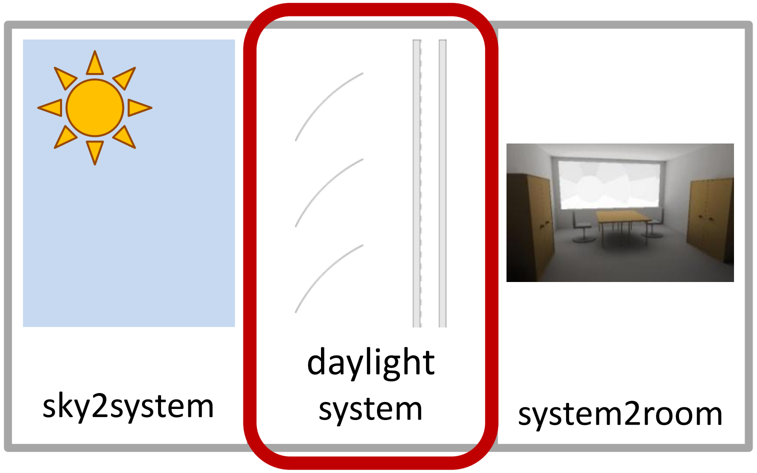
```
oconv myscene.rad sky1.rad > scene_dmx.oct
```

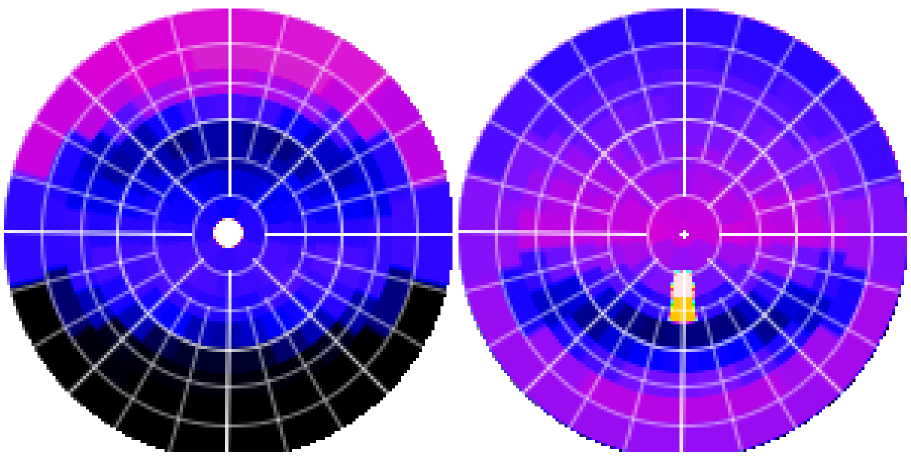
phase2: BSDF

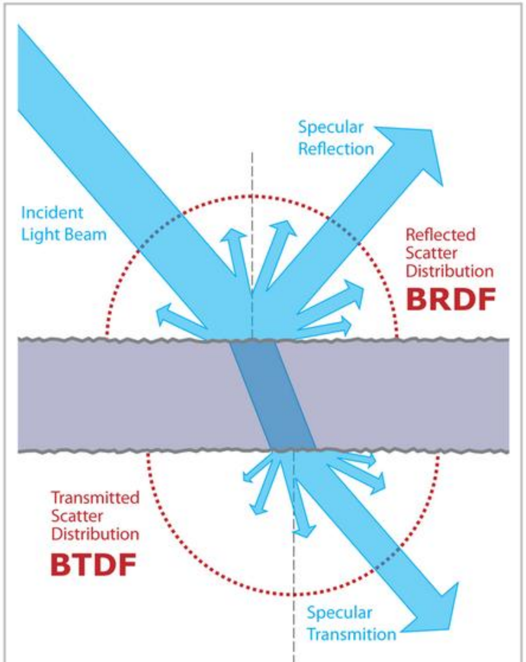
function describing the properties of the daylight system

structure
inside: Klems-patches
outside: Klems-patches

calculation
genBSDF, WINDOW6/7, forwards raytracing (see above)







source: wikimedia.org

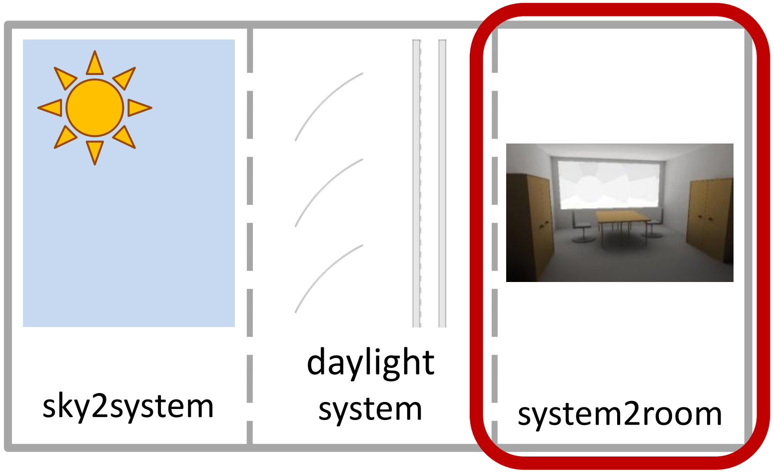
THE 3-PHASE-METHOD

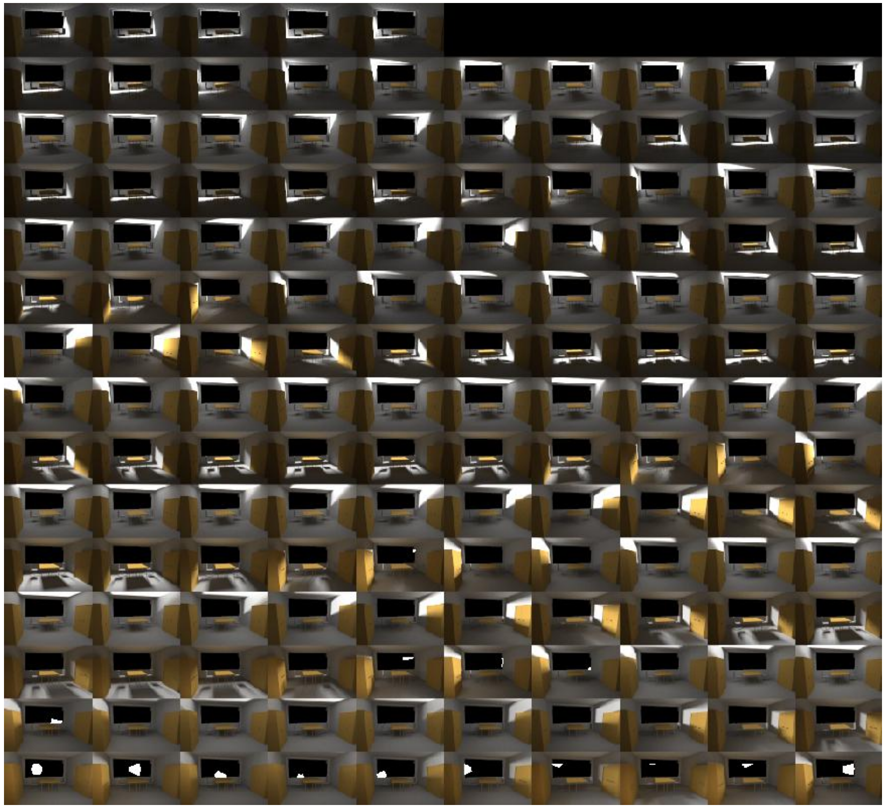
Bartenbach LichtLabor

phase3: view matrix VMX

contribution to the room illumination from every single Klems-patch

structure
facade: Klems-patches
room: points or image pixels

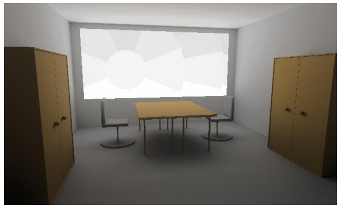




calculation
r(t)contrib, vwrays (images)

examples

- 1 irradiance grid per Klems-patch
- 1 image per Klems-patch



phase3: view matrix VMX

example calculation: irradiance grid

```
rtcontrib -n 8 -f klems_int.cal -b kbinS -bn Nkbins
          -m windowglow -I+ -ab 8 -ad 10000 -lw 1e-8
          scene_vmx.oct < E_grid.pts > E_grid.vmx
```

rtcontrib – what else?

-f: cal-file that calculates *kbinS* and *Nkbins*
-b: current bin number
-bn: total number of bins

E_grid.pts
calculation points and directions

scene_vmx.oct
octree that contains the room a uniformly glowing / lighting window with the modifier for r(t)contrib

```
void glow windowglow
0
0
4 1 1 1 0
windowglow polygon window
0
0
12 -2.25 -2.7 0.85
   -2.25 -2.7 2.85
    2.25 -2.7 2.85
    2.25 -2.7 0.85
```

} in window_glow.rad

```
oconv myscene.rad window_glow.rad > scene_vmx.oct
```

phase3: view matrix VMX

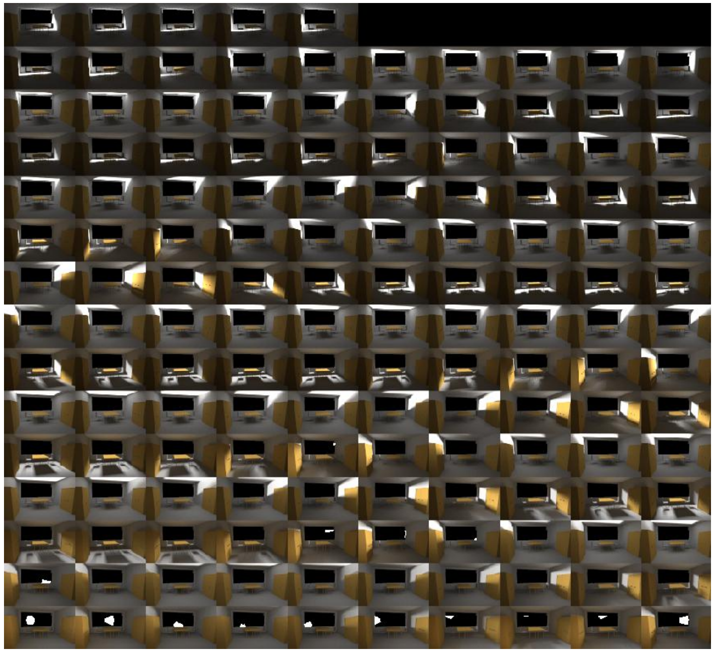
example calculation: radiance image

```
vwrays –ff -vf back_vtv.vf -x 600 -y 600 | \
rtcontrib -n 8 $(vwrays -vf back_vtv.vf -x 600 -y 600 -d)
          -ffc -fo -o img/window_%03d.hdr
          -f klems_int.cal -b kbinS -bn Nkbins
          -m windowlight -ab 8 -ad 10000 -lw 1e-6 scene_vmx.oct
```

vwrays
program to generate ray samples for specified view

$(vwrays …)
get image dimensions from vwrays

-o: specify output destination;
%03d is replaced by the respective bin number



skyvector

subdivision of sky into patches
- Tregenza [145+1]
- Reinhart [577/2305+1]

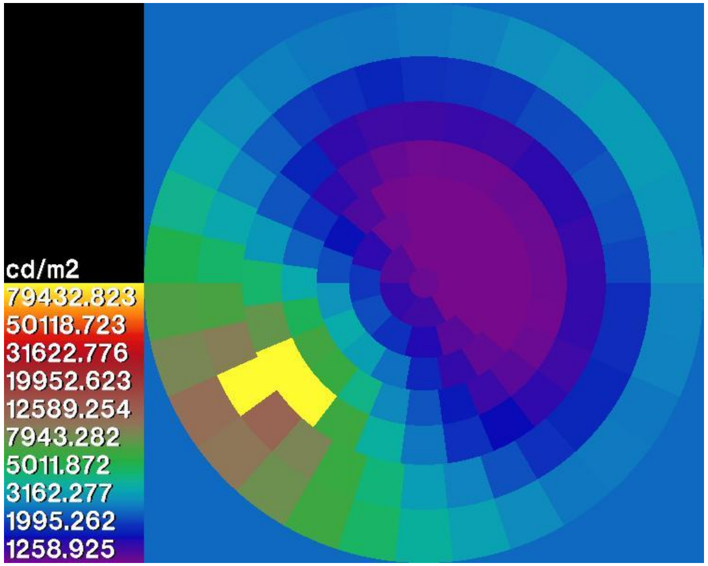
calculation
RADIANCE programs
gensky/gendaylit and genskyvec

example calculation: Reinhart patches for clear sky in Innsbruck

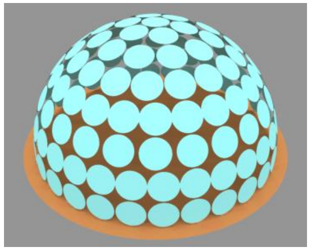
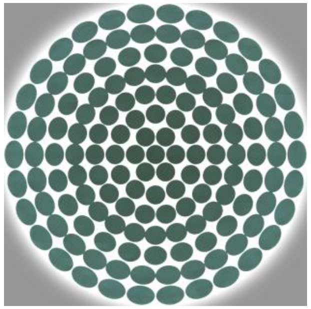
```
gensky 9 21 9:00 -a 47.27 -o -11.39 -m -15 +s | \
genskyvec -m 4 > ibk_skyvec4.skv
```

genskyvec
program to generate average radiances of sky patches

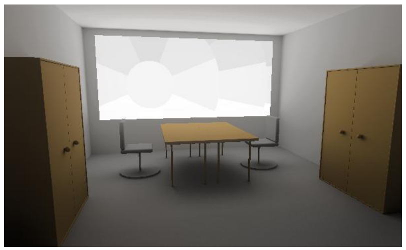
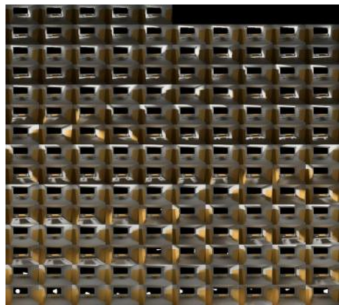
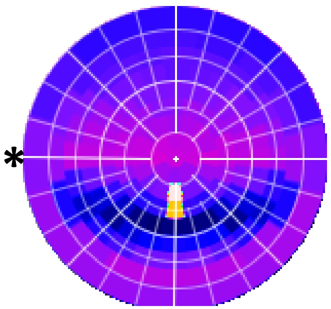
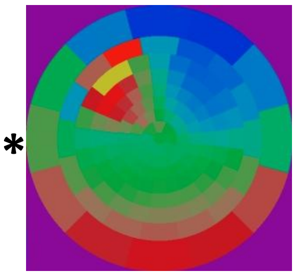
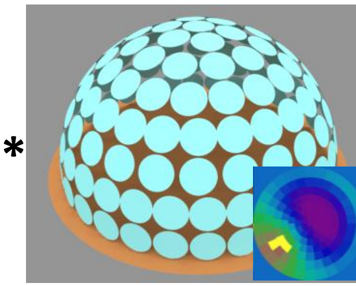
-m: define sky subdivision
1: Tregenza (145 + 1)
2/4: Reinhart (577/2305 + 1)



luminances of a clear sky in Innsbruck (-m 1):
145 Tregenza skypatches + 1 ground patch



indication of the center positions of the 145 Tregenza-skypatches

RADIANCE 3-phase daylight coefficient method


=

*

*

*


result

view matrix VMX

BSDF

daylight matrix DMX

sky radiance distribution

pre-calculation (time-consuming)

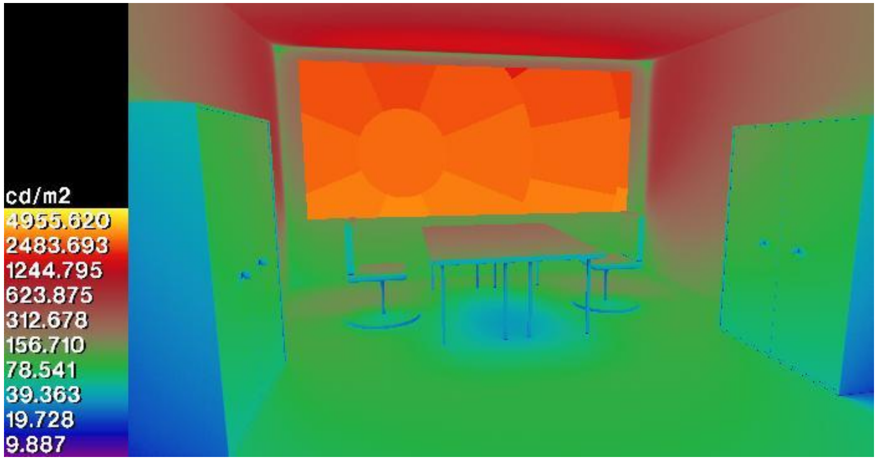
at every time step (fast matrix calculation)

combining the matrices – dctimestep

example calculation – image

```
dctimestep img/window_%03d.hdr system.xml south.dmx ibk_skyvec4.skv > result.hdr
```

view matrix	BSDF	daylight matrix	sky vector	result file
145 contribution images		contributions from sky	sky radiance distribution	radiance image



simulation of a day

climate data from S@tel-Light

data every 30 min -> generate
Perez sky with `gendaylit`

settings of the venetian blind

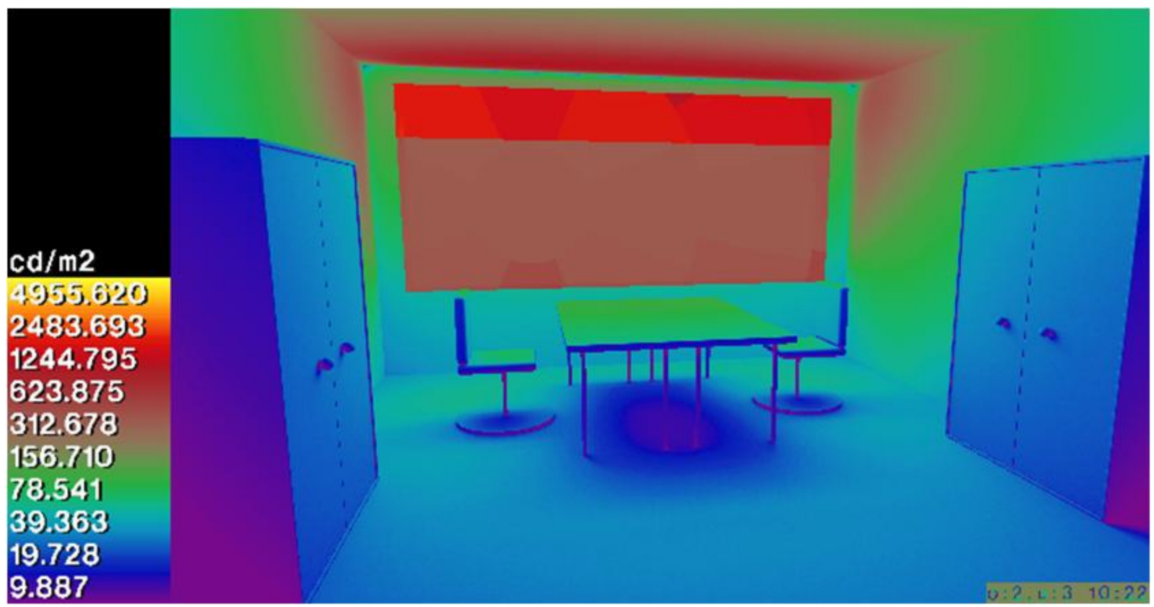
(control depending on sun angle, different settings for lower and upper part):

0 … 0° tilt angle

1 … 20° tilt angle

2 … 40° tilt angle

3 … 70° tilt angle



further information:
S@tel-Light: climate data for Europe, online

combining the matrices – dctimestep

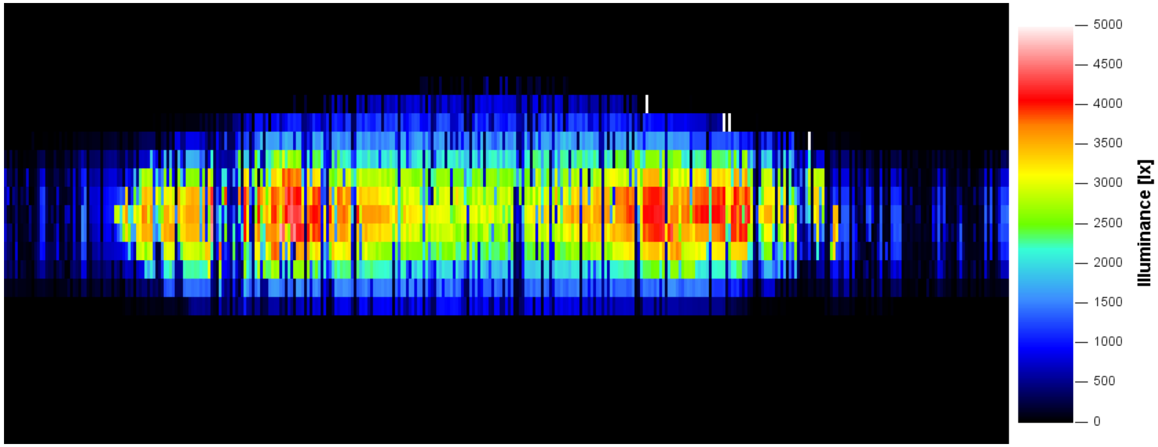
example calculation – sensor point

```
dctimestep E_grid.vmx system.xml south.dmx ibk_skyvec4.skv > result.txt
```

view matrix	BSDF	daylight matrix	sky vector	result file
with grid point contributions		contributions from sky	sky radiance distribution	irradiance values

annual calculation

```
for (hour in 1..8760) do
   generate sky_vector with gendaylit from climate_data(hour)
   calculate result(hour) with dctimestep for appropriate BSDF
done
```



further possibilities

classical daylight coefficient method („1-phase-method“)

```
dctimestep dc_matrix.dcmx skyvec.skv > result.txt/hdr
```

daylight coefficient matrix
relative contributions from sky patches to grid points / pixels
(i.e. this includes the DMX and the BSDF if any)

accerlaration of dctimestep

```
dctimestepcpu E_grid.vmx system.xml south.dmx sky.skm 8760 > result_year.txt
```

sky matrix skm
annual sky description

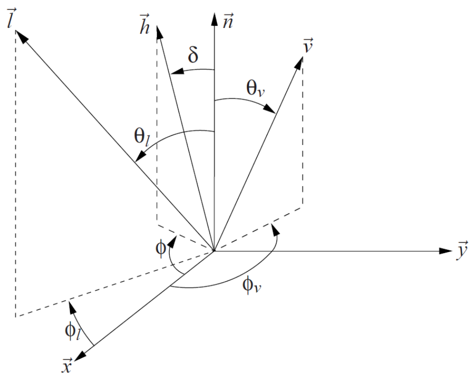
8760
hours in sky matrix

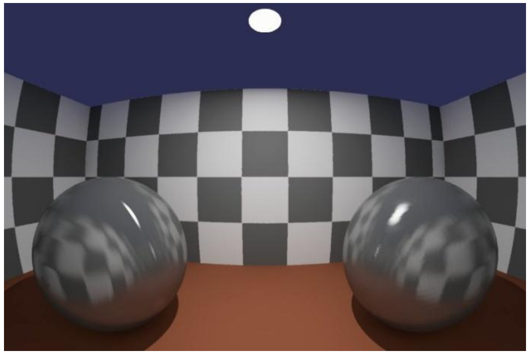
acceleration test:

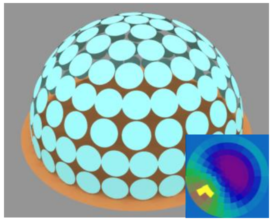
CPU i7 960 3.2 Ghz	1 time step	1 year (8760 h)	speed-up 18.8	½ year (4380 h)
dctimestepcpu	---	1m21s		---
dctimestep	0.174s	25m21s		12m40s

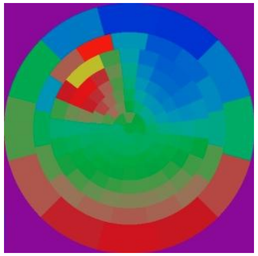
further reading:
D. Bourgeois et al.: “A Standard Daylight Coefficient Model for Dynamic Daylighting Simulations”, online stanford.edu
W. Zuo et al.: “Acceleration of Radiance for lighting simulation by using parallel computing with OpenCL”, Building Simulation 2011

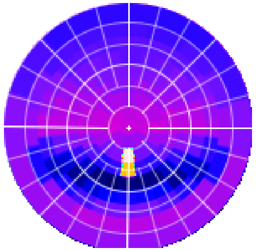
Q & A











Questions?