



LIGHTING THE DARK CORNERS OF *RADIANCE*

Greg Ward, Anywhere Software



WHAT WE'LL LEARN

- ❖ Little-known facts about *Radiance* scene descriptions
- ❖ Useful file types, tips, and tools
- ❖ *Radiance* programs many don't know but might appreciate



TOPICS

- I. The Information Header
- II. Materials & Modifiers in Scene Files
- III. Wavefront .OBJ Files and Their Uses
- IV. Octrees and Triangle Meshes
- V. The **rsensor** Program
- VI. Utilities in *Radiance* Not Tied to Simulation

I. THE INFORMATION HEADER

- ❖ N non-empty lines of ASCII text describing how a file was made
- ❖ Usually begins with “#?RADIANCE” line to identify it
- ❖ Ends with a single empty line - essentially two EOLs, “\n\n”
- ❖ Metadata often included in lines of the form:
VARIABLE= Value, which may be anything up to next EOL
- ❖ Last line often “FORMAT=*content_data_identifier*”

TYPICAL INFO HEADER

```
#?RADIANCE
SOFTWARE= RADIANCE 5.1a by LBNL
oconv room4.rad
rtrace -ffc -x 4000 -y 4000 -n 4
EXPOSURE=2.519842e+00
CAPDATE= 2017:02:13 12:03:02
GMT= 2017:02:13 20:03:02
PRIMARIES= 0.6400 0.3300 0.2900 0.6000 0.1500 0.0600 0.3333 0.3333
FORMAT=32-bit_rle_rgbe
```

* commands used to create file

THE GETINFO TOOL

- ❖ Prints info header, prints common file dimensions (**getinfo -d**)
 - Recognizes octree and image types and prints dimensions
- ❖ **getinfo -a** adds each command word as line in new header
 - Reads original on stdin, sends modified to stdout
- ❖ **getinfo -r** replaces metadata variable with new setting (or deletes)
- ❖ **getinfo -** removes header from stdin (**getinfo -d** - also removes size)
- ❖ **getinfo -c** runs given command on raw data and enjoins output

GETINFO EXAMPLES (1)

- ❖ Print headers for all HDR pictures in current directory:

```
getinfo *.hdr
```

- ❖ Print all image dimensions:

```
getinfo -d *.hdr
```

- ❖ Get octree dimensions, but don't echo file name:

```
getinfo -d < scene.oct
```

GETINFO EXAMPLES (2)

- ❖ Add owner metadata to a picture:

```
getinfo -a "OWNER= Greg Ward" < orig.hdr > my.hdr
```

- ❖ Insist picture has a different exposure than the one recorded:

```
getinfo -r "EXPOSURE=1.75" < orig.hdr > corr.hdr
```

- ❖ Add offset of [1.3 9.1 -2.7] to every triple in “float” format matrix:

```
getinfo -c rcalc -if3 -of -e \  
`$1=$1+1.3;$2=$2+9.1;$3=$3-2.7' < in.mtx > adj.mtx
```

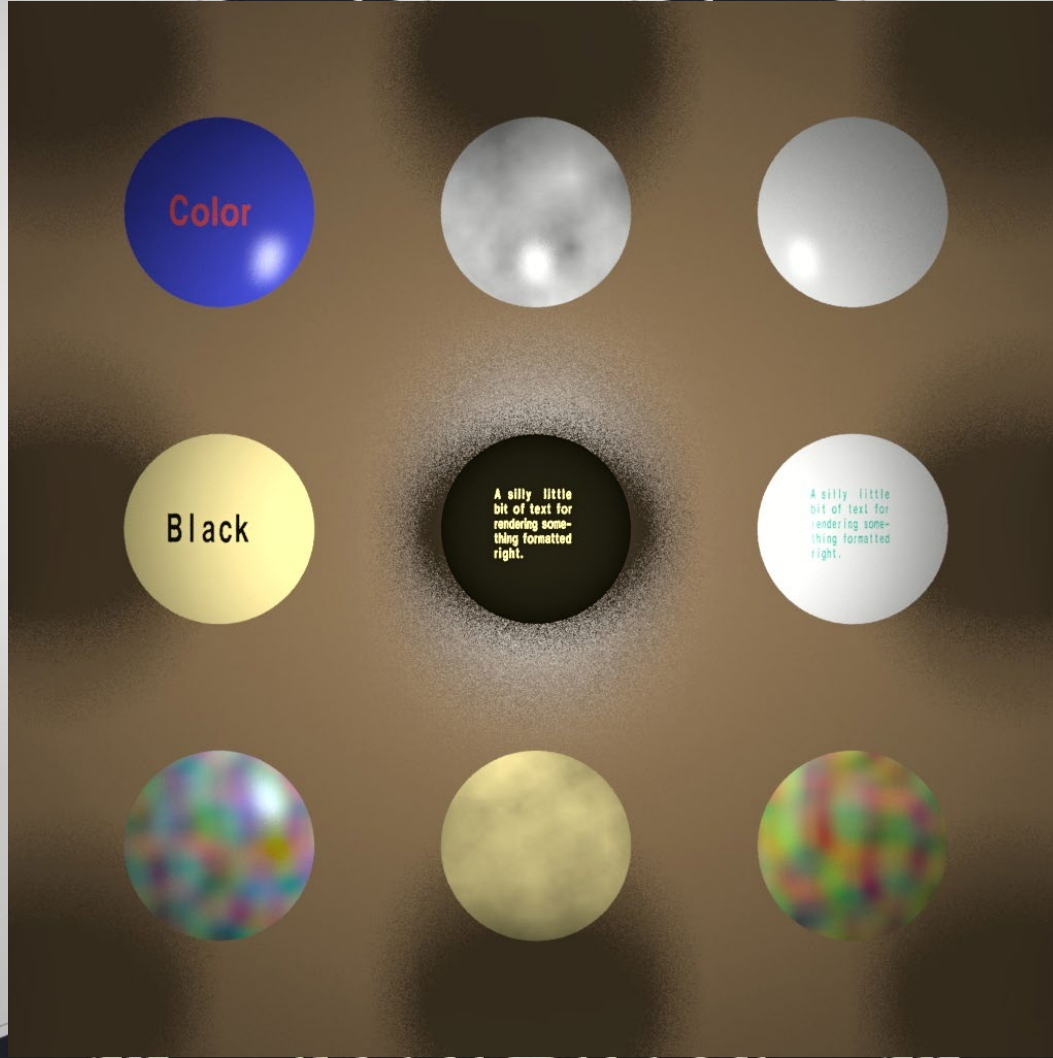
QUESTIONS ANSWERED BY QUESTIONS

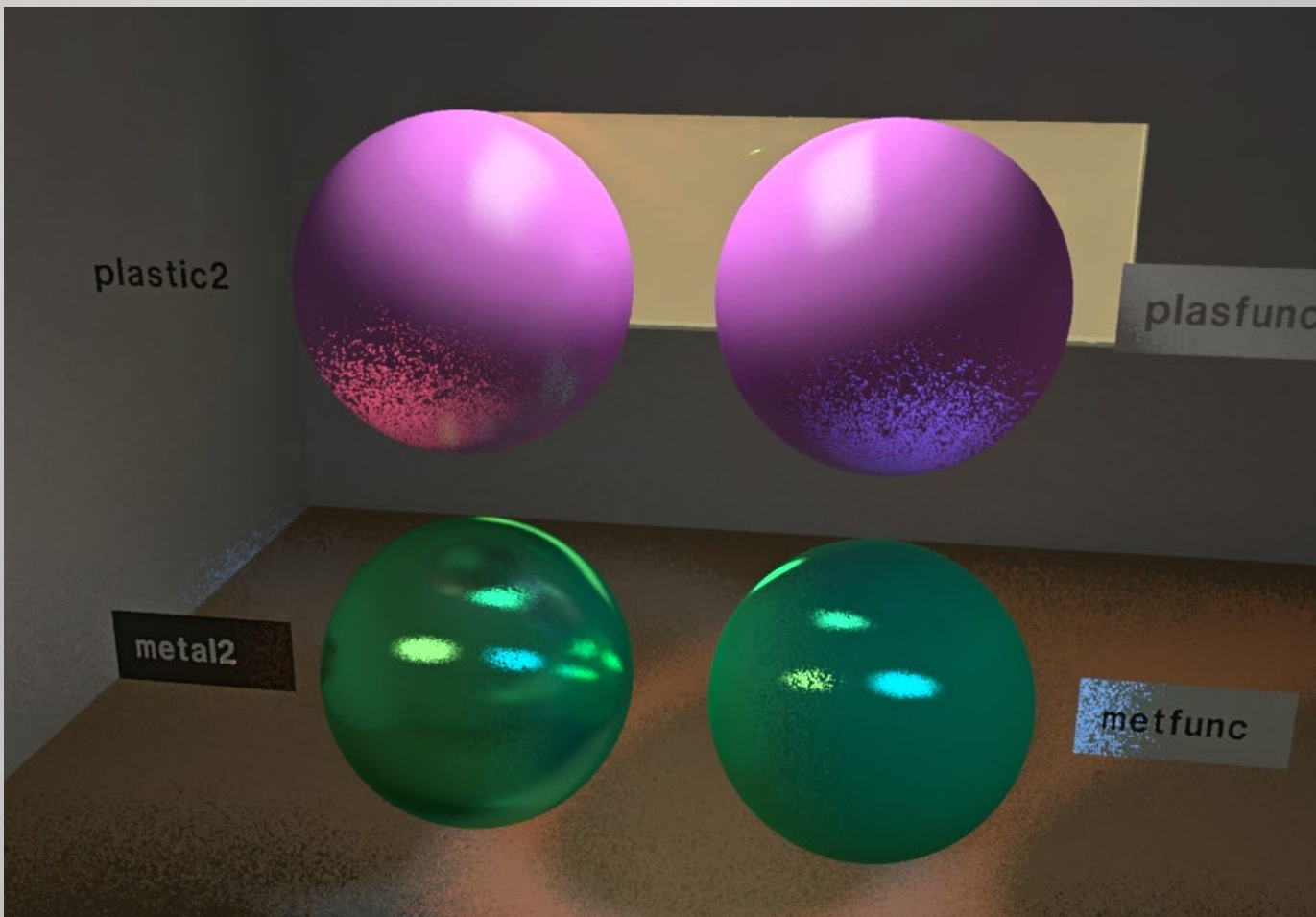
- ❖ Why have an information header at all?
 - Ever wondered what's in your “scene1_v2.oct” file two months on?
- ❖ Why not have a more sophisticated metadata standard?
 - Have you ever had to write a TIFF parser?
- ❖ What's with the “#?RADIANCE” line at the top?
 - Look into the Unix **file** command, which relies on a magic number or sequence at the start of each binary file
 - better than relying on standard suffixes, which can be ambiguous

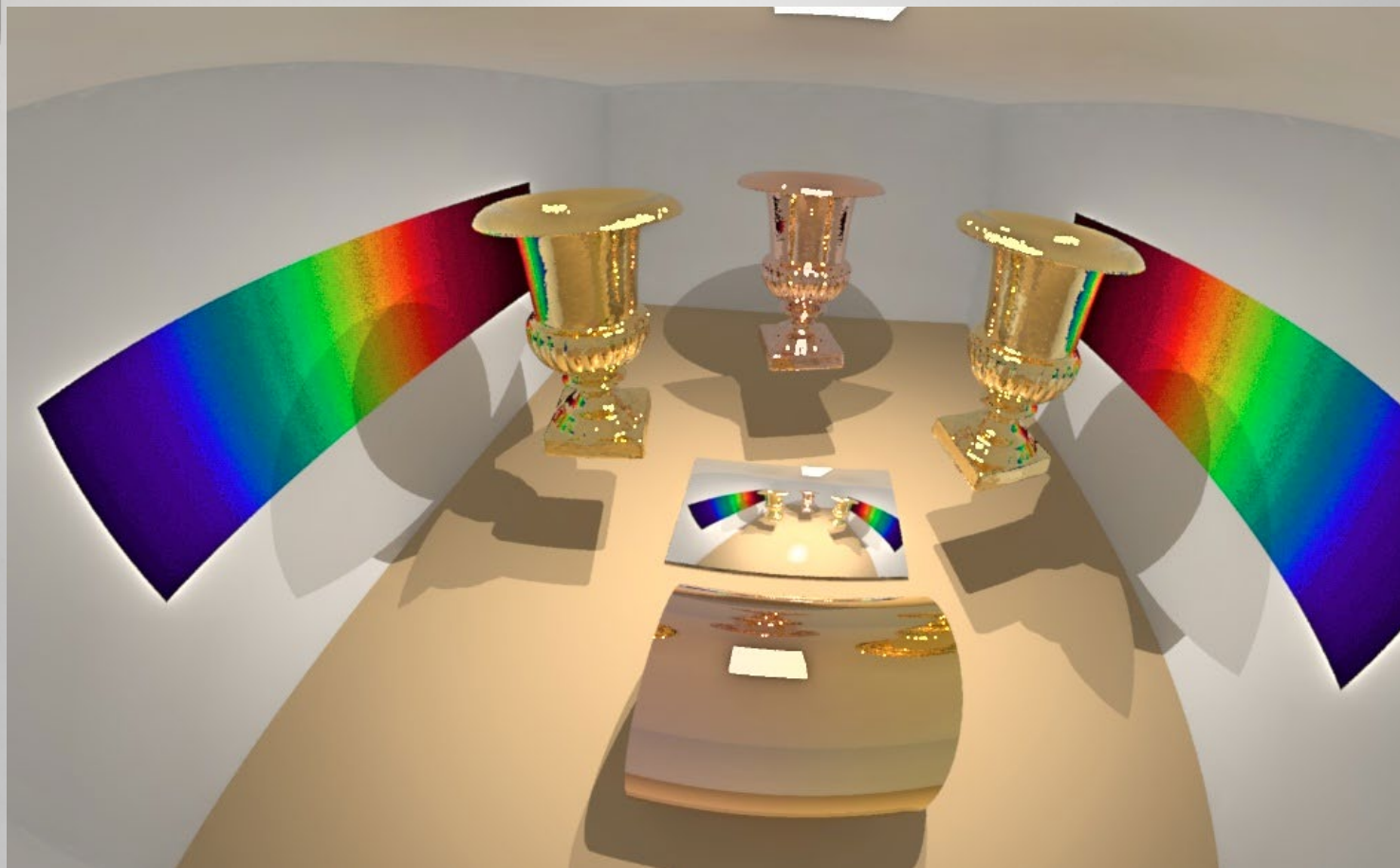
II. MATERIALS & MODIFIERS IN SCENE FILES



Examples from ray/test/renderers







RADIANCE INPUT TOPICS

- ❖ Scene file basics for beginners
- ❖ Scene file basics for experts and haters (I'm both)
- ❖ What is a modifier, and what can't be a modifier?
- ❖ What is a material?
- ❖ What are aliases and how do I use them?
- ❖ What is a volume? A secondary source? A virtual source material?
- ❖ Auxiliary files and the fiddly bits around variables & functions

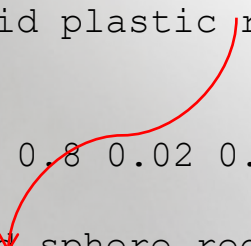
BEGINNER SCENE BASICS

- ❖ A *Radiance* scene is made of “primitives,” which are independent bits of text defining surfaces and associated metadata
- ❖ Surfaces are typically unordered, but most recent “modifier” definition is applied, so sequencing matters for those
- ❖ A line beginning with ‘!’ character is executed as command whose output should be more scene data inserted at this point in input
 - Great for generators and hierarchies

BEGINNER SCENE BASICS

```
# A comment is on its own line, starting with # sign
# Every primitive follows form:
#   modifier type identifier
#   Ns [Ns string words, which may be "quoted"]
#   0
#   Nr [Nr real values]
```

```
void plastic red
0
0
5 0.8 0.02 0.015 0.04 0.09
                                # free-format
red sphere red_ball
0 0 4 10 15 4 2
```

 **modifier chain**

BEGINNER SCENE BASICS

```
# Remember what we said about commands?
```

```
!xform -n ch1 -s .5 -rz 27 -t 10 -5 19 chair.rad
```

```
void alias another_name_for_red  
    red
```

```
another_name_for_red ring red_disc  
0 0 8  
    19 5 3  
    0 0 1  
    0 3
```

EXPERT HATER BASICS

- ❖ Why are primitives so primitive?
 - Designed around `scanf()` call -- I had written too many parsers
 - Didn't think people would have to do much hand-editing (wrong!)
- ❖ Fundamentally good things:
 - Inline commands, well-defined physical ranges, so many materials
- ❖ Fundamentally bad things:
 - Argument typing, RGB color assumed, so many materials!
- ❖ Antimatter and the ultimate fate of the universe

TO MODIFY OR NOT TO MODIFY?

- ❖ Anything that is not a surface or a distant *source* can be a modifier
- ❖ The “void” keyword stands in for no modifier, and is the starting point in any modifier chain
 - A modifier may also be redefined or added to by reusing its name
- ❖ Some modifiers refer to other modifiers via their string arguments
 - E.g., the *illum* and *mirror* materials may specify an alternate modifier
 - The *mix** types may combine two modifiers, and one may be “void”
 - The *antimatter* type can hide selected surface modifiers in a volume
- ❖ An *alias* can also reconnect modifier chains

WHAT IS A MATERIAL?

❖ Here is the current set of *Radiance* material primitives:

- *plastic, metal, glass, trans, dielectric, plastic2, metal2, trans2, interface, plasfunc, metfunc, glow, light, illum, spot, mist, mirror, transfunc, BRTDfunc, BSDF, aBSDF, WGMDfunc, plasdata, metdata, transdata, antimatter, prism1, prism2, ashik2*

❖ You can only have one material in a modifier chain

- The *mix** modifier types are the only exception this rule
- Non-material modifiers may come before and/or after the material
- Any surface missing a material in its modifier chain is invisible
- Spectral modifiers were introduced to enable rendering beyond RGB

WHAT ARE ALIASES FOR?

❖ In its simplest form, an alias associates a new id with a previously-defined modifier:

`inherit alias new_name original_name`

- the “inherit” keyword above says use `original_name`’s modifier

`new_modifier alias new_name original_name`

- would apply “new_modifier” instead to `original_name`
- we could have used “void” to end modifier chain at `original_name`

❖ Note that *aliases* apply to any modifier, not just final or material

WHAT ABOUT VOLUMES?

- ❖ A volume is simply an enclosed set of surfaces in *Radiance*
 - Surface normal orientations often matter for volumes
- ❖ Opaque materials don't care if surfaces form a volume or represent infinitely thin films; they look the same from the outside
- ❖ Some non-opaque *Radiance* materials assume thin film surfaces:
 - *glass*, *trans**, *BRTDfunc*, *BSDF*, *prism1*, *prism2*, and *WGMDfunc*
- ❖ Other *Radiance* materials assume interaction with a volume:
 - *dielectric*, *interface*, *mist*, *anitmatter*, and sometimes *BSDF*



antimatter letter volumes subtract
chunks from a dielectric cylinder

GREG WARD

SECONDARY SOURCES?

- ❖ Early optimization tells *Radiance* a window or surface is a particularly important source of indirect illumination
 - Classic is *illum* material for surface at window (**mkillum**)
 - Also useful for skylights, diffuse reflectors, etc.
- ❖ Care is taken to avoid double-counting in the diffuse indirect part
- ❖ Photon-mapping is an alternate approach to this problem

Secondary source on
window handles light
reflected by sky and
blinds

Allows direct from
sun to pass between
blind slats



VIRTUAL SOURCES?

❖ Virtual sources are created during a prepass when **-dr** > 0 and virtual source material is present in scene

- Currently *mirror*, *prism1*, *prism2*

❖ Number of virtual sources added to render is around:

(how many sources see a virtual surface)^N where N is **-dr** setting

- Mitigated by virtual-view calculation
- People who live in mirror houses shouldn't throw rays!

❖ Troublesome to combine with *mist*, but it works

*mirror creates
virtual suns*

*illum's for sky
through window
and light reflected
off diffuse ceiling*



AUXILIARY FILES

- ❖ Files referenced by *Radiance* scene primitives
 - Includes *.cal files, fonts, data files, pictures, and BSDFs (XML)
- ❖ Typically refers to scene directory or library path in \$RAYPATH
- ❖ With exception of text types that just use fonts, most primitives referencing auxiliary files employ .cal-style expressions for coordinates
 - Thus important to understand .cal variables and functions



```
# Secondary source material  
# from light shelf office
```

```
void brightdata wind_illum.dist  
5 noneg ./dist/window6.dat  
    illum.cal il_alth il_azih
```

```
0
```

```
9
```

1.000000	0.000000	0.000000
0.000000	0.000000	-1.000000
0.000000	1.000000	0.000000

Function defined
in "rayinit.cal"

```
# Secondary source material  
# from light shelf office
```

```
void brightdata wind_illum.dist
```

```
5 noneg ./dist/window6.dat
```

Data file

Library .cal file → illum.cal il_alth il_azih

```
0
```

```
9
```

Defined variables

1.000000	0.000000	0.000000
0.000000	0.000000	-1.000000
0.000000	1.000000	0.000000

Real arguments used by
il_alth and il_azih definitions

CONTENTS OF ILLUM.CAL

```
norm_rad(r) : if( r, r, r+2*PI );
```

```
il_alt = Dz;
```

```
il_azi = norm_rad(Atan2(Dy, Dx));
```

```
il_alth = sq(-Dx*A7-Dy*A8-Dz*A9);
```

```
il_azih = norm_rad( Atan2(-Dx*A4-Dy*A5-Dz*A6,  
                          -Dx*A1-Dy*A2-Dz*A3) );
```

Variables we are defining in this .cal file

```
norm_rad(r) : if( r, r, r+2*PI );
```

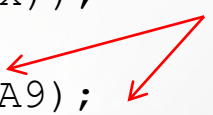
```
il_alt = Dz;
```

```
il_azi = norm_rad(Atan2(Dy, Dx));
```

```
il_alth = sq(-Dx*A7-Dy*A8-Dz*A9);
```

```
il_azih = norm_rad( Atan2(-Dx*A4-Dy*A5-Dz*A6,  
                          -Dx*A1-Dy*A2-Dz*A3) );
```

These are the two
we are accessing



Predefined constants and constant functions

```
norm_rad(r) : if( r, r, r+2*PI );
```

```
il_alt = Dz;
```

```
il_azi = norm_rad(Atan2(Dy, Dx));
```

```
il_alth = sq(-Dx*A7-Dy*A8-Dz*A9);
```

```
il_azih = norm_rad( Atan2(-Dx*A4-Dy*A5-Dz*A6,  
                          -Dx*A1-Dy*A2-Dz*A3) );
```

Variables defined by ray under evaluation

```
norm_rad(r) : if( r, r, r+2*PI );
```

```
il_alt = Dz;
```

```
il_azi = norm_rad(Atan2(Dy, Dx));
```

```
il_alth = sq(-Dx*A7-Dy*A8-Dz*A9);
```

```
il_azih = norm_rad( Atan2(-Dx*A4-Dy*A5-Dz*A6,  
                          -Dx*A1-Dy*A2-Dz*A3) );
```

Variables corresponding to modifier's real arguments

```
norm_rad(r) : if( r, r, r+2*PI );
```

```
il_alt = Dz;
```

```
il_azi = norm_rad(Atan2(Dy, Dx));
```

```
il_alth = sq(-Dx*A7-Dy*A8-Dz*A9);
```

```
il_azih = norm_rad( Atan2(-Dx*A4-Dy*A5-Dz*A6,  
                          -Dx*A1-Dy*A2-Dz*A3) );
```

CONTENTS OF WINDOW6.DAT

```
2                                #dimensions
1.071429 0.071429 8           outer dimension (il_alth) limits & count
0.000000 6.283185 23         inner dimension (il_azih) limits & count
                                followed by ordered grid values...

-6.823423e-01 -4.923063e-01 -8.828832e-01 5.304009e-01 4.327287e-01
9.542055e-01 1.380869e+00 2.228299e+00 2.279667e+00 2.344502e+00
2.270633e+00 2.632300e+00 2.667573e+00 2.756300e+00 2.221162e+00
2.477349e+00 1.903239e+00 1.512074e+00 1.405038e+00 1.232479e+00
1.422599e+00 9.834322e-01 -6.823423e-01 3.553007e+00 3.362971e+00
3.753548e+00 2.340264e+00 2.437936e+00 1.916460e+00 1.489796e+00
6.423660e-01 5.909978e-01 5.261627e-01 6.000318e-01 2.383655e-01
2.030917e-01 1.143653e-01 6.495036e-01 3.933157e-01 9.674263e-01
1.358591e+00 1.465627e+00 1.638187e+00 1.448066e+00 1.887233e+00
3.553007e+00 3.573945e+00 2.571717e+00 1.799098e+00 8.189627e-01
7.727998e-01 7.764430e-01 6.961560e-01 7.216548e-01 6.951168e-01
...
```

BACK TO ILLUM.CAL

```
norm_rad(r) : if( r, r, r+2*PI );
```

```
il_alt = Dz;
```

```
il_azi = norm_rad(Atan2(Dy, Dx));
```

```
il_alth = sq(-Dx*A7-Dy*A8-Dz*A9);
```

```
il_azih = norm_rad( Atan2(-Dx*A4-Dy*A5-Dz*A6,  
                          -Dx*A1-Dy*A2-Dz*A3) );
```

il_alth range is 0-1

il_azih range is $0-2\pi$

AUXILIARY OPERATIONS

- ❖ Evaluation context for a given modifier may include:
 - Global variables and functions from “rayinit.cal” or command-line
 - Some variables access information from the ray under evaluation
 - Local variables and functions defined in named .cal file if any
 - The **A_n** or **arg(n)** modifier arguments
- ❖ Functions must be defined in a .cal file, but variable names may be replaced with an inline expression if preferred
 - E.g., instead of putting “var = Dx/2” in a .cal file, use string “Dx/2”
 - If you want spaces in your modifier’s expression, enclose in quotes



OTHER THINGS TO REMEMBER

- ❖ A .cal file defines functions and variables, not a sequence of steps
- ❖ Order isn't critical, but there are advantages to defining things before referring to them in a given .cal file
- ❖ Defining recursive or mutually recursive functions is fun!
 - Calls are more efficient if you pass arguments that are complicated expressions you would otherwise need to repeat in an evaluation
- ❖ No one “gets” .cal files the first time
 - Comment files generously for the sake of others and your later self



Materials and Modifiers

QUERIES?

III. WAVEFRONT .OBJ FILES

- ❖ One of earliest, most widely used exchange formats in CAD world
- ❖ Simple, shared-vertex, polygonal b-rep description
 - Compact and efficient for meshes & most architectural geometry
- ❖ Group and material names enable simple shading associations
- ❖ Original **obj2rad** and newer **robjutil** import .OBJ with useful controls
- ❖ The **gensurf** and **genbox** tools optionally produce .OBJ output
- ❖ The **obj2mesh** program converts .OBJ to memory-efficient .rtm file
 - Offers only means to import mapped images (local coordinates)

OBJ2RAD & ROBJUTIL

- ❖ Original **obj2rad** converter takes user-input rules to assign material names to surfaces (**-n** and **-m** options)
- ❖ New **robjutil** can output modified .OBJ as well as *Radiance* scene
 - Identifies and removes degenerate and redundant faces
 - Option to merge vertices within a given epsilon distance
 - Option to turn all polygons into triangles
 - Options to keep or remove specific materials or groups
 - Option to apply **xform**-style transform
 - Option to output *Radiance* scene (rather than modified .OBJ)

EXAMPLE ROBJUTIL USE

```
# File processed by: robjectil +r -t -n -c 1e-6 +v -g Standard.3.5/8
CoconutPoint_Model_v2.obj
# 12403045 statements read from "CoconutPoint_Model_v2.obj"
# 2713096 faces
#           9648628 vertices, 0 texture coordinates, 0 surface normals
# Deleting the following groups:
#           Standard.3.5/8
#           (2495674 faces removed)
# Coalesced 546530 vertices with epsilon of 1e-06
# Removed 3 degenerate faces
# Removed 14245 duplicate faces
```

(Removed problematic group of
unknown origin)

```
DEFAULT_MATERIAL polygon AAPL_Ceiling_Exposed.1
```

```
...
```

GENSURF & GENBOX

❖ Since **gensurf** generates parametric surfaces, (s,t) can be texture coordinates which are preserved with the **-o** option to output .OBJ

```
gensurf mat nam " $x(s,t)$ " " $y(s,t)$ " " $z(s,t)$ " m n -s -o | obj2mesh > rtm
```

❖ In contrast, **genbox** offers .OBJ output because it calls **robjutil**'s library so no additional programming was required

- Also may come in handy for compact mesh representations

❖ Tool-generated .OBJ output may be concatenated to create a scene

THE OBJ2MESH COMPILER

- ❖ Similar to **oconv**, **obj2mesh** compiles scene geometry into an efficient form for rendering that includes an octree structure
- ❖ By sharing vertex info and optimizing coordinates, a *Radiance* triangle mesh uses approx. $1/9^{\text{th}}$ the memory of equiv. octree instance
- ❖ As mentioned, an RTM also enables local surface coordinates for texture atlases and the like
 - A mesh with *vt* coordinates defines **Lu** and **Lv** ray modifier variables

A 13MB VASE (RTM)



Would be twice as big as frozen
octree without texture coordinates

A 634MB CAROUSEL (RTM)



An ornate, symmetrical decorative border in a light gray color, featuring intricate scrollwork and floral motifs, framing the central text area against a dark background.

Wavefront .OBJ file

QUERIES?

THE **RSENSOR** PROGRAM

- ❖ Development sponsored by AEC (Zack Rogers) to calculate accurate light sensor signal for SPOT program
- ❖ Calls renderer as library to evaluate uniformly weighted random distribution of rays computed by discrete Monte Carlo inversion
- ❖ Produces either averaged value or sampling rays for annual sims
 - Averages send extra rays to sample sources, controlled by **-dn**
- ❖ Command-line view origin, direction, and up places each sensor

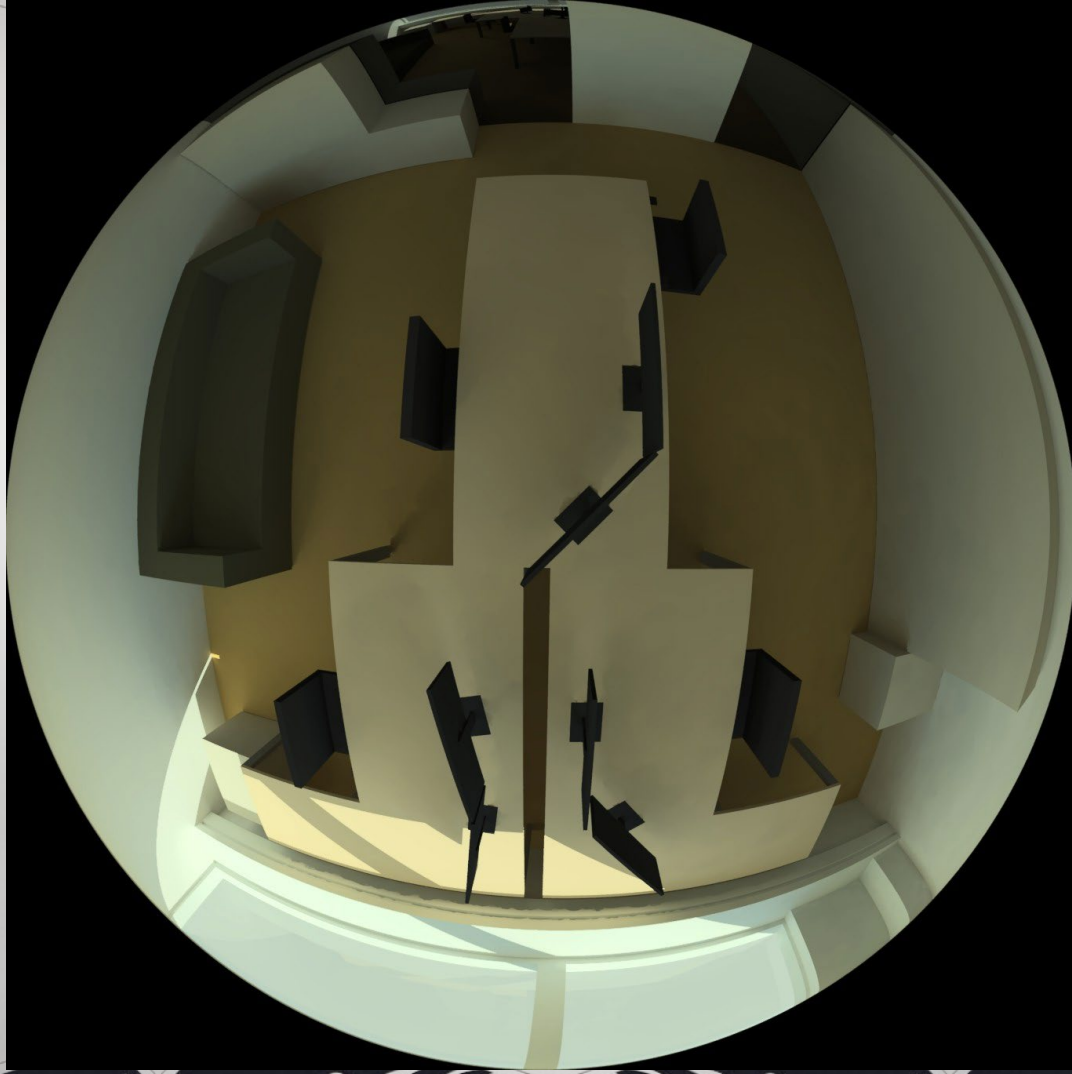
SPOT SENSOR FORMAT

		Azimuth Angle			
Polar Angle	Elevation angle, f	Azimuth angle, q, degrees -->			
	degrees	0	90	180	270
	0	.02	.04	.02	.04
	45	.01	.02	.01	.02
	90	.001	.002	.001	.002

SENSOR SUBTLETIES

- ❖ All azimuth angles must be specified, 90° resolution minimum
 - Duplicate column values for a radially symmetric distribution
- ❖ Polar angles (rows) go from 0 to maximum field of view
 - Polar angles above last row are assigned zero sensitivity
- ❖ Ray values are effectively weighted by sensitivity function
 - Tracing and averaging ray values equiv. to coefficient integral = 1.0
- ❖ Spectral simulation is supported, but sensitivity table is Y only

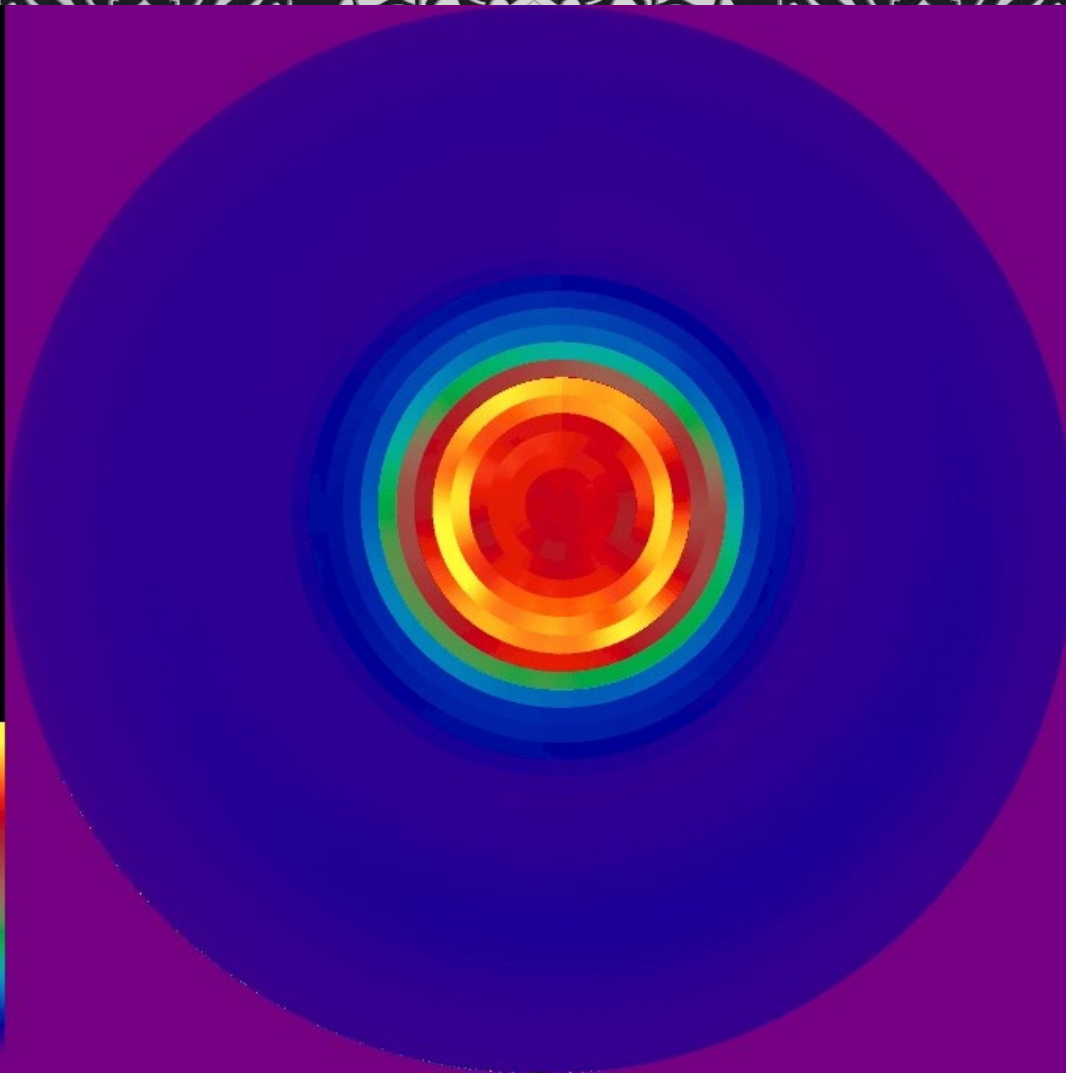
Example Ceiling View



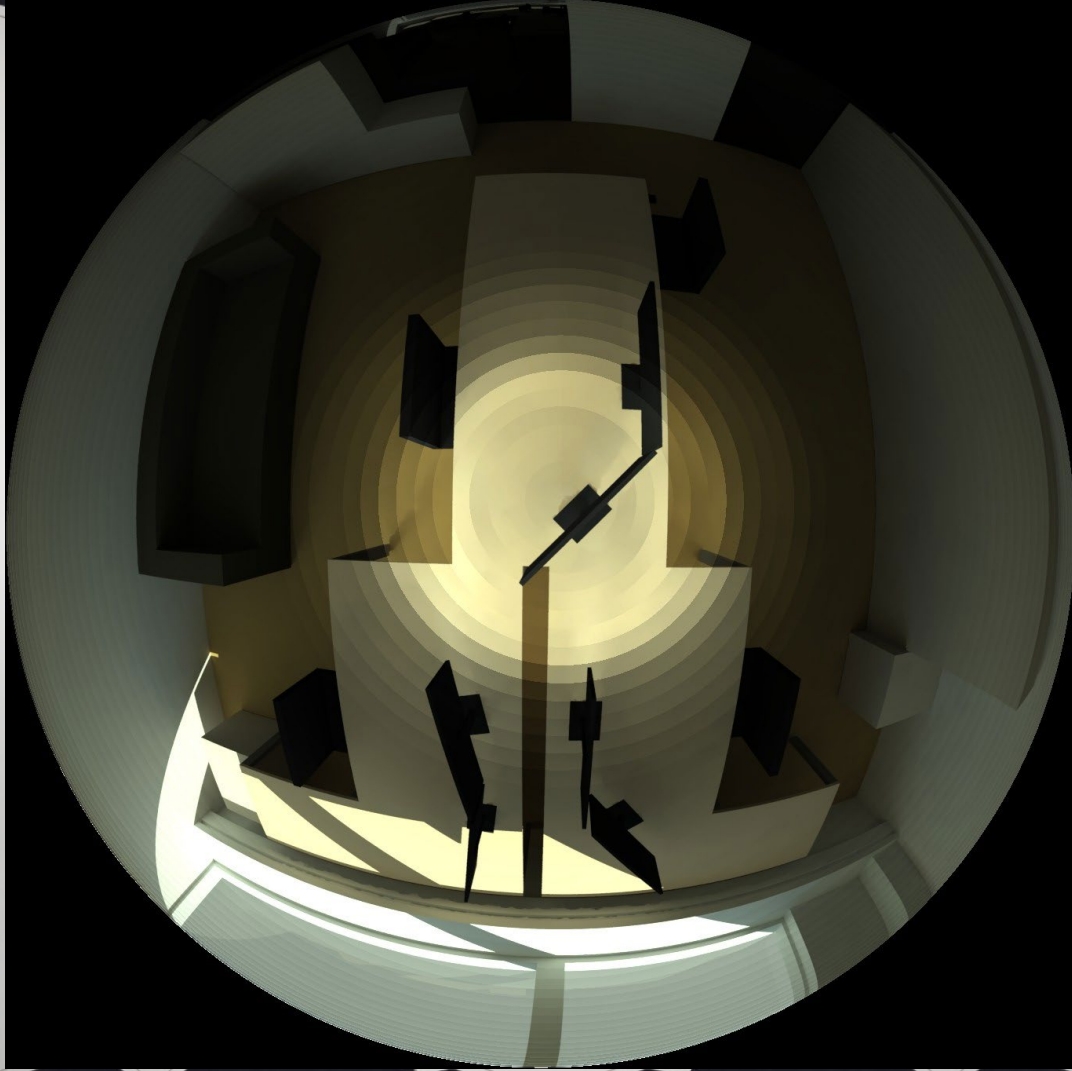
Example Sensitivity

Weight

1
0.875
0.75
0.625
0.5
0.375
0.25
0.125
0



Example Sensor Product



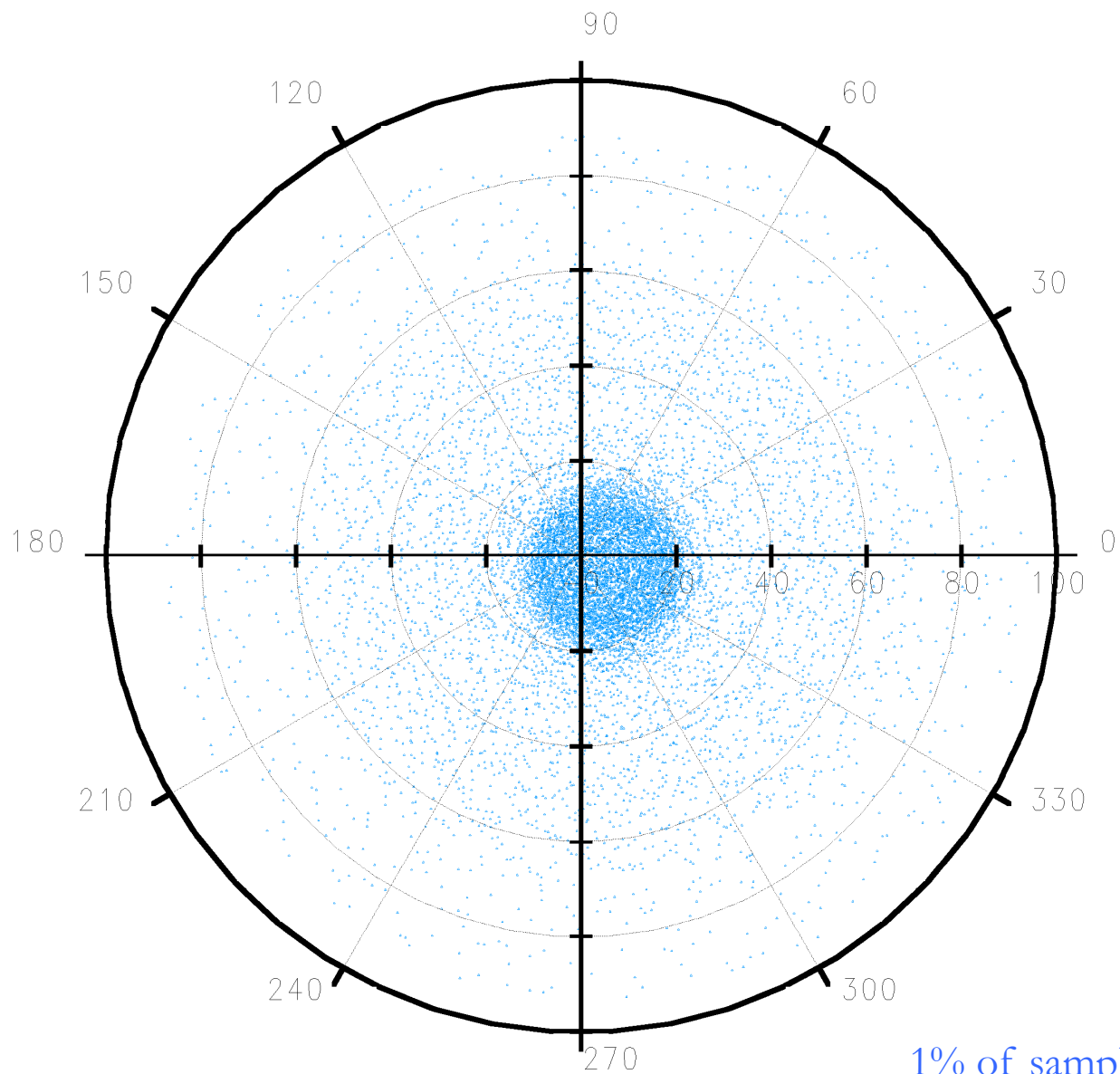
RSENSOR COMMANDS

```
rsensor -h -vf Views/ceil_sens2.vf -rd 1000000 -n 4 -dn 0 @day_render.opt \
Novitas_01PCI_Spatial.dat day_model.oct
1.4492 1.7517 1.0337
```

```
rsensor -h -vf Views/ceil_sens2.vf -rd 1000000 Novitas_01PCI_Spatial.dat . \
| rtrace -h -n 4 @day_render.opt day_model.oct \
| total -m
3.57722706 4.32354358 2.5513827
```

Multiplied by the spherical integral of the weighting function (0.40514) we get:

1.4493 1.7517 1.0337



1% of samples



rsensor Program

QUERIES?



NON-SIMULATION UTILITIES

- ❖ General data manipulation
 - **cnt, rcalc, total, rlam, rsplit**
- ❖ Special-purpose data analysis
 - **histo, tabfunc**
- ❖ N-tuple matrix operations
 - **rcollate, rcrop, rmtxop, rcomb**



GENERAL DATA MANIPULATION

- ❖ **cnt** generates (optionally random) index list to drive other tools
- ❖ **rcalc** flexibly operates on real input and produces real output
- ❖ **total** sums or averages columnar input, computes RMS, etc.
- ❖ **rlam** combines columns from different streams (laminates)
- ❖ **rsplit** separates columns into multiple output streams

CNT

❖ Argument list is grid dimensions, produces indices starting at 0

- **cnt 3 2**

0	0
0	1
1	0
1	1
2	0
2	1

❖ **cnt -s** option shuffles output rows randomly, faster than **sort -R**

❖ First 20 factorials in random order:

```
cnt -s 20 | rcalc -e 'fact(n):if(n-1.5,n*fact(n-1),1); $1=fact($1+1)'
```

RCALC

- ❖ Takes real input in N columns, produces M output columns
 - Alternatively, input or output format may be given with **-i** or **-o**
- ❖ Uses .cal expressions and files
 - Searches \$RAYPATH libraries
- ❖ E.g., compute luminance from *Radiance* RGB:
`rcalc -e '$1=179*(.265*$1+.670*$2+.065*$3)' rgb.txt > lum.txt`
- ❖ E.g., stretch Z dimension by factor of 2 for triangle file:
`rcalc -p -i trin.fmt -o troz2.fmt orig.rad > zstretch2.rad`

RCALC

trin.fmt:

\$(mod) polygon \$(nam)

0

0

9

$\{v1x\} \{v1y\} \{v1z\}$

$\{v2x\} \{v2y\} \{v2z\}$

$\{v3x\} \{v3y\} \{v3z\}$

troz2.fmt:

\$(mod) polygon \$(nam)

0

0

9

$\{v1x\} \{v1y\} \{2*v1z\}$

$\{v2x\} \{v2y\} \{2*v2z\}$

$\{v3x\} \{v3y\} \{2*v3z\}$

rcalc -p option says pass everything to output that doesn't match input format

TOTAL

- ❖ Default action is to sum separately each numeric input column
- ❖ Options provided to total every N rows, report running totals, and perform operations before summation
- ❖ The **-sE** option raises input values to the power of E before summing, extracting E th root if **-m** also (e.g., **-s2 -m** is RMS)
- ❖ The **-p** option computes the products instead of sums
- ❖ The **-l** and **-u** options report minimum and maximum, instead

RLAM

- ❖ Simple but useful: laminate records from multiple input streams
 - Called a “record” rather than a “column” for generality
- ❖ Arguments list files, stdin (-), !commands, in desired order
- ❖ Options for binary i/o, alternate column separators
- ❖ If an input runs out of rows, output continues with other streams
 - **-in** *M* option says cut off input after *M* rows

RSPLIT

- ❖ Complements **rlam**, splits single input stream to multiple outputs
- ❖ Options for mixed ASCII and binary input in any combination
 - Default separator is newline, unlike **rlam**
 - Space as separator interpreted as any length of white space
- ❖ Provides options for header management, since it is most useful with **rtrace** program and used by **rt pict** script
- ❖ Record outputs may be stdout, named files, and commands
 - The **-a** option says to append an output file (can't be a stream)

SPECIAL PURPOSE DATA ANALYSIS

❖ **histo** takes a single column of data and produces a (cumulative) histogram

❖ **tabfunc** takes a set of dependent data columns that follow a monotonic independent data column, and produces a set of functions in .cal file format

HISTO

- ❖ Single input value column expected, has two command forms:
 1. **histo** *xmin xmax nbins*
 2. **histo** *imin imax*
- ❖ First form gives minimum and maximum real limits and number of desired bins
 - Input values outside the given range are silently ignored
- ❖ Second form takes integer values from *imin* to *imax* inclusive and produces a count of each on output (i.e., *imax-imin+1* bins)

HISTO

- ❖ **histo -c** option reports a cumulative distribution
- ❖ **histo -p** reports percentage of total in place of integer bin counts
- ❖ Out-of-range input values still count in total percentage
 - **histo -c -p** may produce <100 at final bin if inputs outside of range

TABFUNC

❖ Little-known utility takes a table of dependent real values and converts into one function per column in .cal format

```
cnt 7 | rcalc -e 't=PI/2/7*$1' -e '$1=t; $2=sin(t)' \  
      | tabfunc isin
```

```
isin(x):select(x/0.224399+1,0,0.2225209,0.4338837,0.6234898,  
0.7818315,0.9009689,0.9749279);
```

- Adding **tabfunc -i** option:

```
interp_arr2`(i,x,f):(i+1-x)*f(i)+(x-i)*f(i+1);  
interp_arr`(x,f):if(x-1,if(f(0)-x,interp_arr2`(floor(x),x,f),f(f(0))),f(0));  
isin`p5957(i):select(i,0,0.2225209,0.4338837,0.6234898,  
0.7818315,0.9009689,0.9749279);  
isin(x):interp_arr`(x/0.224399+1,isin`p5957);
```

TABFUNC

- ❖ Most efficient way to convert 1-D data tables to .cal functions
- ❖ Linear interpolation even works with unevenly spaced samples
 - Finds closest independent value prior to interpolation
- ❖ E.g., get animation path from list of views and intervals between
 - See “spline.cal” file in src/cal/cal directory
- ❖ Some **select()** calls end up being quite large, but current library handles these efficiently

N-TUPLE MATRIX OPS

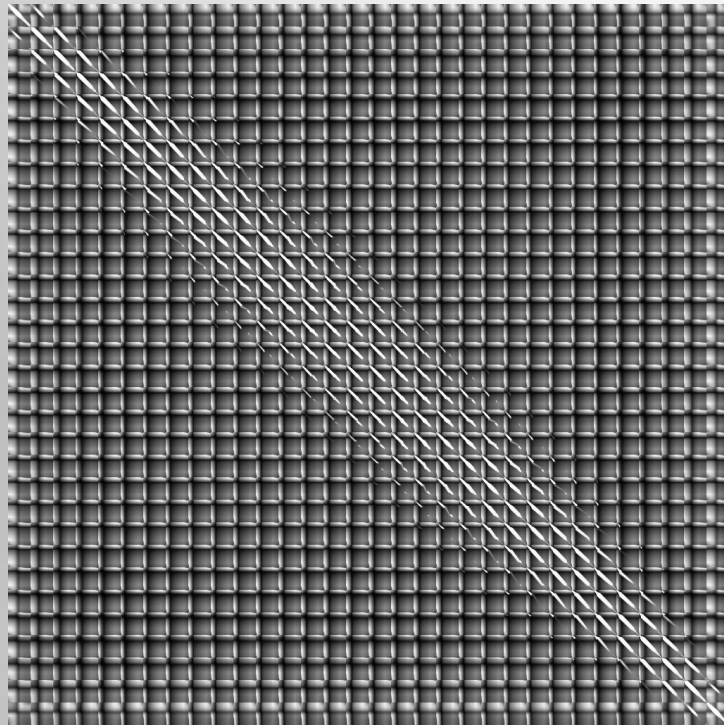
- ❖ **rcollate** transposes and reshapes matrix without interpreting elements
- ❖ **rcrop** efficiently truncates matrix or image rows/columns
- ❖ **rmtxop** performs concatenation, addition, subtraction, multiplication, division, color transforms, and matrix transpose in memory
 - Accepts BSDF (XML) matrices, similar to **dctimestep**
- ❖ **rcomb** optimized for larger matrices, generating one output row at a time in smaller memory footprint with more general math operations, like **pcomb**

RCOLLATE

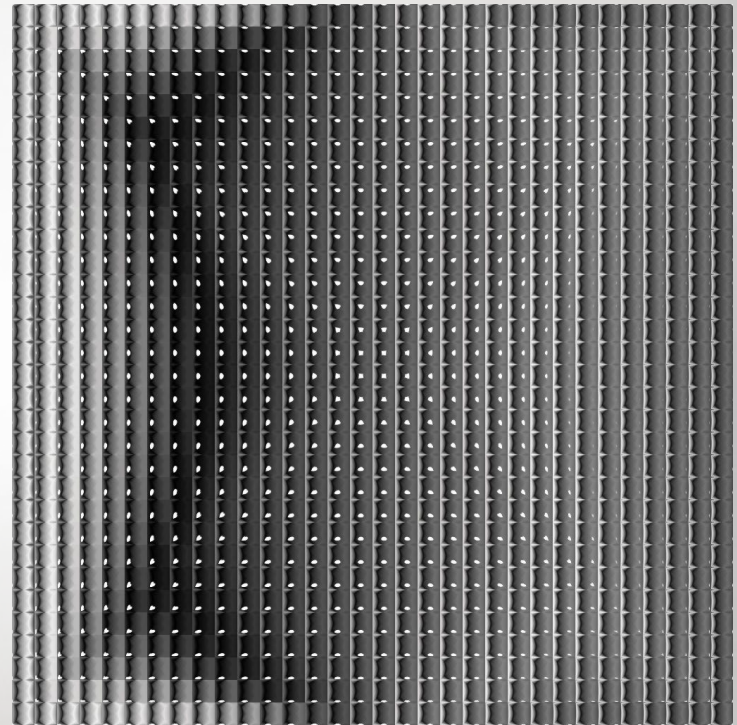
- ❖ Quick tool for adjusting headers and reshaping tabulated data
- ❖ Often used to add or adjust row/column counts
- ❖ Transpose operation does not interpret elements
 - Faster not to parse ASCII to real, but implies no type conversion
- ❖ Options for tensor to matrix of matrices reordering
 - Mostly useful for visualizing 4-D BSDFs, etc.

Venetian Blinds BSDF Example

Shirley-Chiu BSDF as 2-D



rcollate -o 32x32x32x32



RCROP

❖ Universal cropping tool for matrices and picture files

- No type conversion -- output format matches input

❖ Unusual but useful syntax:

rcrop *r_first c_first r_count c_count* [*input* [*output*]]

- if *r_count* is 0, include all rows $\geq r_first$
- if *c_count* is 0, include all columns $\geq c_first$
- negative *r_count* or *c_count* means subtract from dimension max
- Columns are X coordinates and rows (Y) count from top=0

❖ Adjusts VIEW metadata if present to match cropping

RMTXOP

- ❖ The original Swiss army knife of N-tuple matrix operations
 - Performs concatenation, per-element add, subtract, multiply and divide
 - Associative operations (i.e., concatenation) are ordered for efficiency
 - Non-associative operations may require multiple invocations
 - Matrix prep operations include transpose, element transforms, and value scaling
 - Convenient shorthand provided for standard color space transforms
- ❖ Accepts XML files on input, extracting either BTDF or BRDF
- ❖ Each binary operation is completed before loading the next operand, so there are at most 3 matrices in memory at one time

RMTXOP EXAMPLES

Multiply skylight BTDF by glazing layer's spectral transmittance and concatenate with **D** (daylight) and **S** (sky) matrices to obtain Klems angular output at each hour of the year

```
rmtxop -c `cat rooflight_spec.txt` T70_Reference_Hall_Skylight.xml \  
daylight_r4.mtx Stuttgart_daytime_sky4.mtx > kf_day.mtx
```

RMTXOP EXAMPLES

Multiply skylight BTDF by glazing layer's spectral transmittance and concatenate with **D** (daylight) and **S** (sky) matrices to obtain Klems angular output at each hour of the year

Final workplane illuminance concatenates spectral sampling, converts to Y channel (with 179 multiplier) at very end

```
rmtxop -c `cat rooflight_spec.txt` T70_Reference_Hall_Skylight.xml \  
daylight_r4.mtx Stuttgart_daytime_sky4.mtx > kf_day.mtx
```

```
rmtxop wp_sensor_kf.mtx kf_day.mtx -c Y > day_wpill.mtx
```

RCOMB

- ❖ Lives halfway between **rmtxop** and **pcomb**, falling short on some features but surpassing both in others
 - Applies arbitrary expressions one output row at a time
 - Avoids need to load entire input matrix, which can be prohibitive
 - Allows for a single (final) matrix concatenation
- ❖ Default operation is element-wise addition (like **pcomb**)
- ❖ .cal files and expressions allow arbitrary per-element operations

RCOMB

❖ Variables are predefined for each pixel operation:

- **r** is output row number, from 0 to **nrows**-1
- **c** is output column number, from 0 to **ncols**-1
- **ci**(i,p) is the p^{th} component from the i^{th} input
- **nfiles** constant is the number of inputs
- **ncomp** is the color component count
- **wl**(j) is the center wavelength (in nm) for component j

❖ **co**(p) is the user-assigned function defining p^{th} output component

- If component operations are identical, a **co** variable may be assigned instead calling **ci**(i) function, which loops through components

RComb EXAMPLES

Compute ratio between melanopic and photometric
luminance at each pixel of hyperspectral picture

```
rcomb -e 'co=ci(1)/(ci(2)+1e-7)' -fc -c M render.hsr -c Y render.hsr > ratio.hdr
```

RComb EXAMPLES

Compute ratio between melanopic and photometric luminance at each pixel of hyperspectral picture

Reverse spectral values Red ↔ Blue

```
rcomb -e 'co=ci(1)/(ci(2)+1e-7)' -fc -c M render.hsr -c Y render.hsr > ratio.hdr
```

```
rcomb -e 'co(p)=ci(1,ncomp+1-p)' -fc orig.hsr > reversed.hsr
```



Non-simulation Utilities

QUERIES?



THANK YOU FOR YOUR
ATTENTION