What's In a Picture?

- A picture in *Radiance* is a map of RGB radiance (or irradiance) values

- The exposure of a *Radiance* picture may be adjusted without loss since it contains a dynamic range on the order of 10^77

- Individual radiance (or luminance) values may be displayed on demand by the X11 viewer, **ximage**

- The **falsecolor** program may be used to convert an image to a numerically readable value map with legend

- The **glare** program may be used to identify and analyze glare sources in a picture or scene

- Other programs (principally **rtrace**) may be used to compute values that are not easily represented as a map

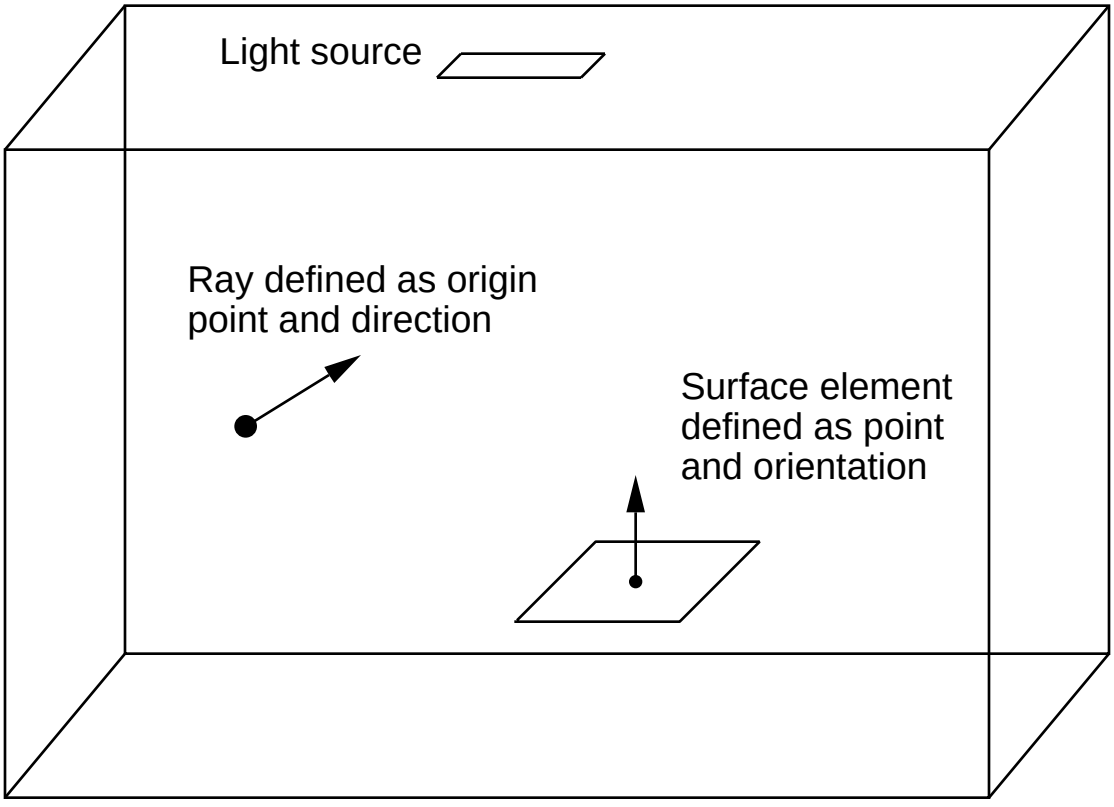
Computing Other Values

- Though most people associate *Radiance* with pictures, anything is possible
- The basic computation engine is **rtrace**
- Other programs call **rtrace** to compute what they need
  - **mkillum** computes output of "secondary sources"
  - **findglare** analyzes potential glare sources

Other Values

RTRACE

- **rtrace** is a renderer without a view
- takes one or more rays and traces them
- the result can be radiance or irradiance or...
- can also take surface element for irradiance



Other Values

RTRACE

Command Example: Query Lux Values

```
% ximage scene.pic \
      | rtrace -i -x 1 -h @render.opt scene.oct \
      | rcalc -e '$1=47*$1+117*$2+15*$3'
```

- **ximage** will produce ray origin and direction for each pick
- **rtrace -i** will trace ray to surface and compute irradiance
- **-x 1** option causes output to be flushed after each ray
- **-h** option says leave off header, other options in `render.opt`
- more convenient script called **rlux** provides same functionality
- **ximage** can get precomputed luminances by itself

Command Example: Workplane Illuminances

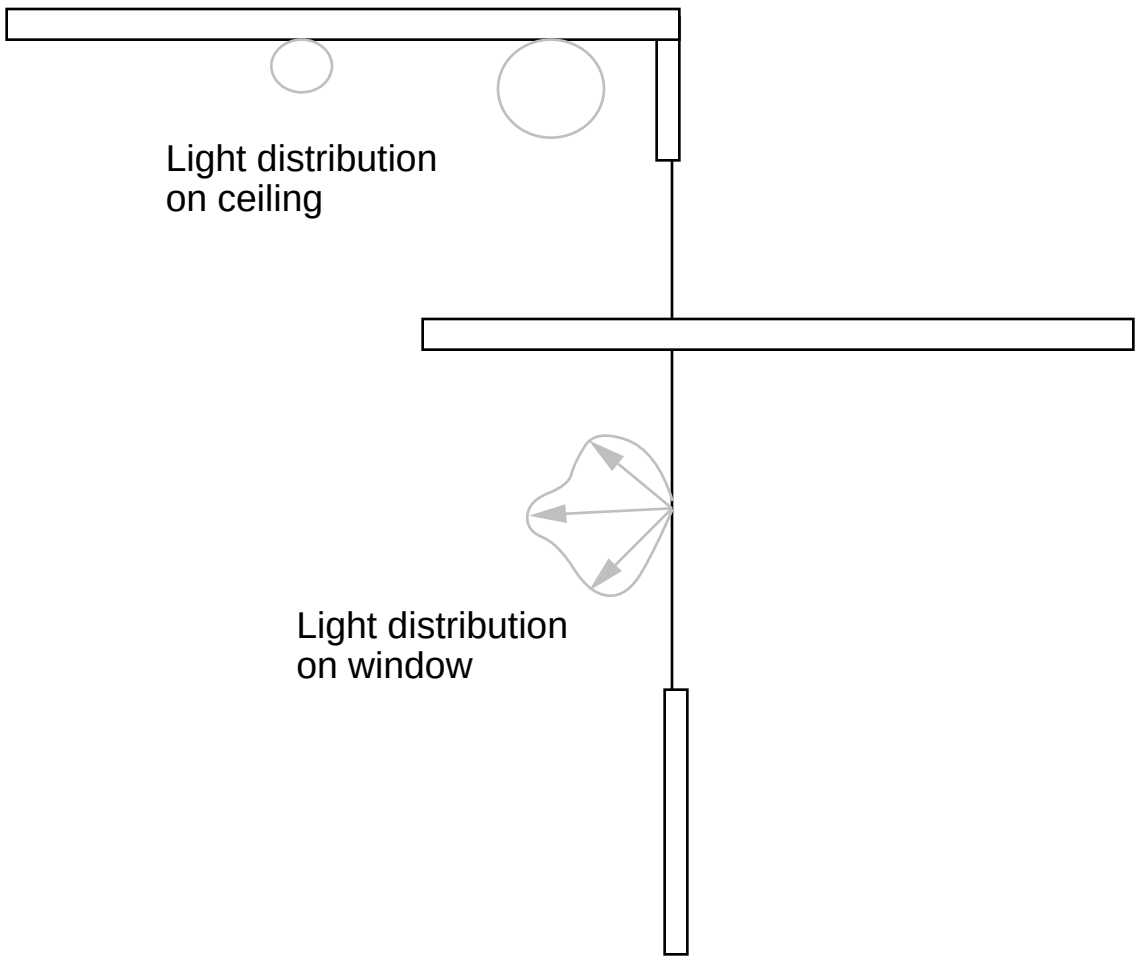
```
% cnt 10 15 \
      | rcalc -e '$1=$1/2+.25;$2=$2/2+.25;$3=1.2;$4=0;$5=0;$6=1'
      | rtrace -h -opv -I @render.opt scene.oct \
      | rcalc -e '$1=$1;$2=$2;$3=47*$4+120*$5+12*$6' \
      > values.dat
```

- **cnt** generates an array of indices
- first **rcalc** converts indices to surface elements
- **rtrace** with **-I** option computes point irradiance rather than radiance
- **-opv** says output point followed by value
- second **rcalc** command gets x, y of point and computes lux values

Other Values

MKILLUM

- Computes output of "secondary sources"
- Takes *Radiance* descriptions of surfaces
- Produces descriptions of secondary sources
- **rtrace** subprocess does acutal computation



Other Values

MKILLUM

Command Example:

```
% oconv -i scene.oct object.rad > scene0.oct
% mkillum @render.opt scene0.oct < object.rad > illum.rad
% oconv -i scene.oct illum.rad > scene1.oct
```

- first **oconv** adds an object to our initial scene's octree
- **mkillum** creates secondary source for this object
- options for **rtrace** are in the file `render.opt`
- second **oconv** puts new source into final scene

Example Object:

```
# The following special comment specifies a data file to mkillum
#@mkillum f=data/object
# Mkillum will add a suffix.  The data directory must exist

void polygon window_illum
0 0 12  5 10 15  15 10 15  15 10 20  5 10 20
```

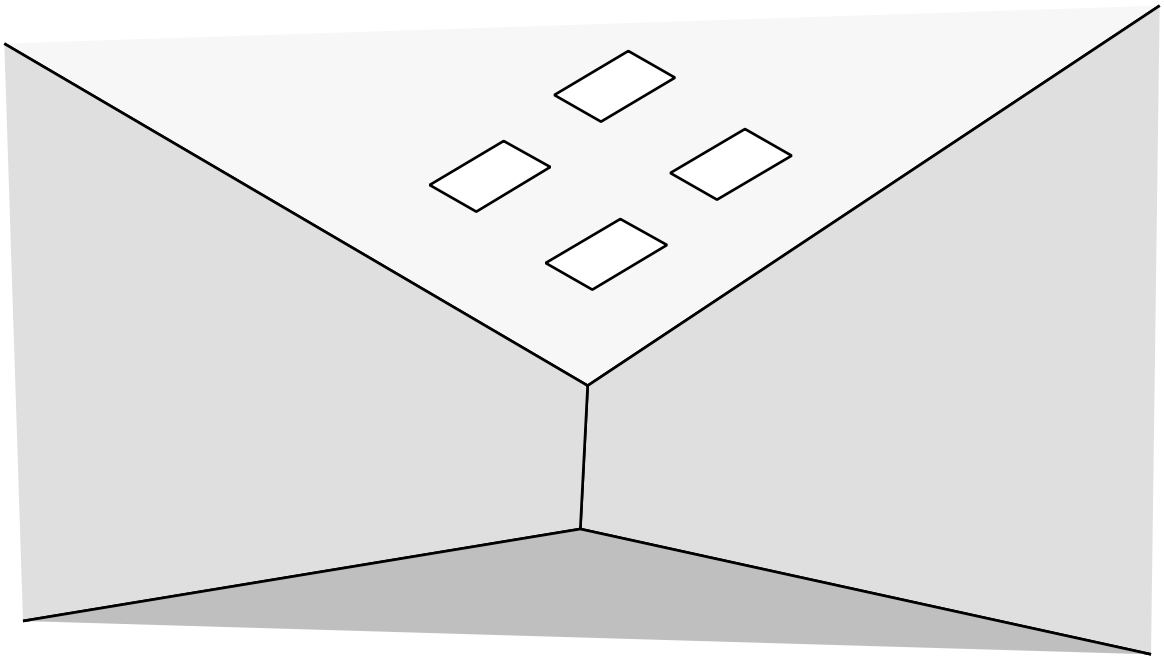
MKILLUM Output:

```
#@mkillum !
void brightdata illum_mat.dist
5 noop data/object.dat illum.cal il_alth il_azih
0
9  -1 0 0  0 0 1  0 1 0

illum_mat.dist illum illum_mat
0
0
3 10.858313 10.858313 10.858313

illum_mat polygon window_illum
0 0 12  5 10 15  15 10 15  15 10 20  5 10 20
```

Other Values

FINDGLARE

- Locates and quantifies potential glare sources
- Input is *Radiance* octree and/or picture file
- Output is list of glare sources and vertical illuminances, used by **glarendx**
- Runs **rtrace** to compute luminances not found in input picture (if any)



Other Values

FINDGLARE

- Operating principle: Search for sources of glare
  - compute luminances with rtrace (or recall them from a picture)
  - threshold is defined as 7x the average or input by user
  - if luminance is above threshold, then put this point in a glare source
  - glare source is any contiguous bright region
- Compute vertical illuminances
  - hemispherical average of luminances for each view direction
  - view direction always horizontal (i.e. perpendicular to up vector)
- Program options specify:
  - viewpoint for glare calculations
  - view direction(s)
  - threshold for glare sources (optional)
  - resolution of scene sampling
  - input picture
  - input octree
  - **rtrace** calculation options
- Send output to **glarendx** or similar program
  - computes selected glare index value from **findglare** output
- Usually accessed through interactive script, **glare**

Other Values **FINDGLARE**

Command Example:

```
% findglare -ga 15-90:15 -p scene_fish.pic \
     -av .5 .5 .5 scene.oct > scene.glr
% glarendx -t cie_cgi scene.glr > scene.cgi
```

FINDGLARE Output:

```
findglare -ga 15-90:15 -p scene_fish.pic -av .5 .5 .5 scene.oct
VIEW= -vth -vp 15 9 5 -vd 1 0 0 -vu 0 0 1 -vh 180 -vv 180
FORMAT=ascii

BEGIN glare source
        0.404864 -0.909961 -0.089756    0.059310        2253.9
        0.043190 0.981473 -0.186667     0.021310        2043.5
        0.694913 -0.706624 -0.133333    0.004492        2113.7
        0.900071 0.426059 -0.091353     0.014312        2098.4
        0.126552 0.989846 -0.064722     0.017566        2283.3
        -0.106887 0.993290 -0.044155    0.010196        2684.8
        0.886601 0.461766 0.026667      0.013529        2160.2
END glare source
BEGIN indirect illuminance
        90      85.295935
        75      83.303689
        60      81.016624
        45      78.677864
        30      76.471920
        15      73.975748
        0       70.776886
        -15     67.029089
        -30     63.067914
        -45     59.480796
        -60     57.322001
        -75     56.727562
        -90     56.894733
END indirect illuminance
```

Specialty Programs

- Designed to make existing *Radiance* functionality more accessible

- Often implemented in a C-shell script that makes calls to other programs

- Examples:
  - **glare** is a script for performing glare analysis
  - **falsecolor** is a script to make numerical value maps
  - **dayfact** is a script for computing daylight factors
  - **objview** starts **rview** rendering a single object
  - **rad** provides higher-level control over the rendering process

Specialty Programs

GLARE

- C-shell utility script, Q/A-style interaction
- Simplifies control and operation of **findglare**
- Performs glare analysis with **glarendx**
- Plots results using *Metafile* 2-d graphics routines

Specialty Programs

FALSECOLOR

- C-shell script, batch mode (i.e. no interaction)
- Calls various filters to create an informative image
- Legend relates image colors to values
- Options for:
  - legend title (normally "Nits")
  - scale (i.e. maximum value)
  - multiplier (i.e. conversion factor)
  - logarithmic scale
  - number of value divisions
  - printing minimum and maximum values
  - contour lines or bands
  - picture to use as source of values
  - picture onto which contours should be overlaid
  - colors to use for each value (for experts only)

Command Example: Log Illuminance Contours

```
% rpict -i @render.opt -vf orig.pic scene.oct \
     | falsecolor -l Lux -s 1000 -log 3 -cl -p orig.pic \
     > scene_fc.pic
```

Specialty Programs

DAYFACT

- C-shell utility script, Q/A-style interaction
- Computes daylight factor and illuminance on the specified workplane
- Calls **rtrace**, **pfilt** and **falsecolor**
- Runs jobs in the background and notifies by **mail** on completion for convenience
- Workplane must be aligned with X and Y axes