arch2rad (1)	- convert Architrion text file to *Radiance* description
bgraph (1)	- do a set of batch graphs to a metafile
calc (1)	- calculator
cnt (1)	- index counter
cv (1)	- convert between metafile formats
dayfact (1)	- compute illuminance and daylight factor on workplane
dgraph (1)	- do a set of graphs to a dumb terminal
ev (1)	- evaluate expressions
falsecolor (1)	- make a false color *Radiance* picture
findglare (1)	- locate glare sources in a *Radiance* scene
gcomp (1)	- do computations on a graph file.
genblinds (1)	- generate a *Radiance* description of venetian blinds
genbox (1)	- generate a *Radiance* description of a box
genclock (1)	- generate a *Radiance* description of a clock
genprism (1)	- generate a *Radiance* description of a prism
genrev (1)	- generate a *Radiance* description of surface of revolution
gensky (1)	- generate a *Radiance* description of the sky
gensurf (1)	- generate a *Radiance* description of a curved surface
genworm (1)	- generate a *Radiance* description of a functional worm
getbbox (1)	- compute bounding box for *Radiance* scene
getinfo (1)	- get header information from a *Radiance* file
glare (1)	- perform glare and visual comfort calculations
glarendx (1)	- calculate glare index
histo (1)	- compute 1-dimensional histogram of N data columns
ies2rad (1)	- convert IES luminaire data to *Radiance* description
igraph (1)	- interactive graphing program
imagew (1)	- output metafile to Apple Imagewriter
impress (1)	- convert metafile to imPress language for imagen
lam (1)	- laminate lines of multiple files
lampcolor (1)	- compute spectral radiance for diffuse emitter
libmeta.a (3)	- simplified interface to metafile(5)
lookamb (1)	- examine ambient file values
macbethcal (1)	- compute color compensation based on measured Macbeth chart
meta2tga (1)	- convert metafile to Targa image format
metafile (5)	- graphics command interface, similar to plot(5)
mgf2meta (1)	- convert Materials and Geometry Format file to Metafile graphics
mgf2rad (1)	- convert Materials and Geometry Format file to *Radiance* description
mkillum (1)	- compute illum sources for a *Radiance* scene
mx80 (1)	- output metafile to Epson mx-80
neat (1)	- neaten up output columns
normpat (1)	- normalize *Radiance* pictures for use as patterns.
obj2rad (1)	- convert Wavefront .obj file to *Radiance* description
objline (1)	- create metafile line drawings of *Radiance* object(s)
objview (1)	- view *Radiance* object(s)

oconv (1)	- create an octree from a *Radiance* scene description
pcomb (1)	- combine *Radiance* pictures.
pcompos (1)	- composite *Radiance* pictures.
pcond (1)	- condition a *Radiance* picture for output
pdfblur (1)	- generate views for depth-of-field blurring
pexpand (1)	- expand requested commands in metafile
pextrem (1)	- find minimum and maximum values in *Radiance* picture
pfilt (1)	- filter a *Radiance* picture
pflip (1)	- flip a *Radiance* picture.
pinterp (1)	- interpolate/extrapolate view from pictures
plotin (1)	- convert plot(5) to metafile(5) primitives
pmblur (1)	- generate views for camera motion blurring
protate (1)	- rotate a *Radiance* picture.
psign (1)	- produce a *Radiance* picture from text.
psmeta (1)	- convert metafile to PostScript
psort (1)	- sort primitives in metafile as requested
pvalue (1)	- convert *Radiance* picture to/from alternate formats
ra_bn (1)	- convert *Radiance* picture to/from Barneyscan image
ra_gif (1)	- convert *Radiance* picture to Compuserve GIF
ra_pict (1)	- convert *Radiance* pictures to Macintosh PICT files
ra_ppm (1)	- convert *Radiance* picture to/from a Poskanzer Portable Pixmap
ra_pr (1)	- convert *Radiance* picture to/from pixrect rasterfile
ra_pr24 (1)	- convert *Radiance* picture to/from 24-bit rasterfile
ra_ps (1)	- convert *Radiance* picture to a PostScript file
ra_rgbe (1)	- change run-length encoding of a *Radiance* picture
ra_t16 (1)	- convert *Radiance* picture to/from Targa 16 or 24-bit image file
ra_t8 (1)	- convert *Radiance* picture to/from Targa 8-bit image file
ra_tiff (1)	- convert *Radiance* picture to/from a TIFF color or greyscale image
ra_xyze (1)	- convert between *Radiance* RGBE and XYZE formats
rad (1)	- render a *Radiance* scene
rad2mgf (1)	- convert *Radiance* scene description to Materials and Geometry Format
raddepend (1)	- find *Radiance* scene dependencies
ranimate (1)	- compute a *Radiance* animation
rcalc (1)	- record calculator
replmarks (1)	- replace triangular markers in a *Radiance* scene description
rpict (1)	- generate a *Radiance* picture
rpiece (1)	- render pieces of a *Radiance* picture
rtrace (1)	- trace rays in *Radiance* scene
rview (1)	- generate *Radiance* images interactively
t4014 (1)	- output metafile to Tektronix t4014 graphics terminal
tabfunc (1)	- convert table to functions for rcalc, etc.
thf2rad (1)	- convert GDS things file to *Radiance* description
tmesh2rad (1)	- convert a triangular mesh to a *Radiance* scene description

total (1)	- sum up columns
trad (1)	- graphical user interface to *Radiance* rad(1) program
ttyimage (1)	- *Radiance* driver for dumb ASCII terminal
vgaimage (1)	- *Radiance* picture display program for VGA
vwright (1)	- normalize a *Radiance* view, shift it to the right
x11meta (1)	- output metafile graphics to *X11*
xform (1)	- transform a *Radiance* scene description
xglaresrc (1)	- dislpay glare sources under *X11*
ximage (1)	- *Radiance* driver for *X* window system
xshowtrace (1)	- interactively show rays traced on *Radiance* image under *X11*

NAME

arch2rad - convert Architrion text file to *Radiance* description

SYNOPSIS

arch2rad [-n][-m mapfile] [input]

DESCRIPTION

Arch2rad converts an Architrion text file to a *Radiance* scene description. The material names for the surfaces will assigned based on the default mapping or the mapping rules file given in the -*m* option. A mapping file contains a list of materials followed by the conditions a surface must satisfy in order to have that material.

For example, if we wanted all surfaces for blocks with RefId "thingy" and Color 152 to use the material "wood", and all other surfaces to use the material "default", we would create the following mapping file:

```
     default ;
     wood (RefId "thingy") (Color 152) ;
```

All surfaces would satisfy the first set of conditions (which is empty), but only the surfaces in blocks with RefId "thingy" and Color 152 would satisfy the second set of conditions.

Each rule can have up to one condition per qualifier, and different translators use different qualifiers. In *arch2rad*, the valid qualifiers are *Layer, Color*, *Face* and *RefId*. A condition is either a single value for a specific attribute, or an integer range of values. (Integer ranges are specified in brackets and separated by a colon, eg. [-15:27], and are always inclusive.) A semicolon is used to indicate the end of a rule, which can extend over several lines if necessary.

The semantics of the rule are such that "and" is the implied conjunction between conditions. Thus, it makes no sense to have more than one condition in a rule for a given qualifier. If the user wants the same material to be used for surfaces that satisfy different conditions, they simply add more rules. For example, if the user also wanted surfaces in blocks with RefId "yohey" with Colors between 50 and 100 to use "wood", they would add the following rule to the end of the example above:

```
     wood (Color [50:100]) (RefId "yohey") ;
```

Note that the order of conditions in a rule is irrelevant. However, the order of rules is very important, since the last rule satisfied determines which material a surface is assigned.

By convention, the identifier "void" is used to delete unwanted surfaces. A surfaces is also deleted if it fails to match any rule. Void is used in a rule as any other material, but it has the effect of excluding all matching surfaces from the translator output. For example, the following mapping would delete all surfaces in the Layer 2 except those with the color "beige", to which it would assign the material "beige_cloth", and all other surfaces would be "tacky":

```
     tacky ;
     void (Layer 2) ;
     beige_cloth (Layer 2) (Color "beige") ;
```

If neither the -m nor the -n options are not used, arch2rad uses the default mapping file "/usr/local/lib/ray/lib/arch.map". This file simply assigns materials based on color, using the identifiers "c0" through "c255". Appropriate materials for these identifiers are contained in "/usr/local/lib/ray/lib/arch.mat".

The -n option may be used to produce a list of qualifiers from which to construct a mapping for the given Architrion file. If the *-m* option is used also, only those blocks matched in the mapping file will be added to the qualifier list.

DETAILS

Architrion blocks are divided into about 6 polygons. The reference, opposite and end faces must all be quadrilaterals (ie. four-sided polygons), though one or more faces may disappear in certain degenerate cases. The bottom face will usually be a quadrilateral, though it may be written out as two triangles if the face is non-planar or one triangle if there is a degenerate side. The top face is treated the same as the bottom face.

Openings are currently handled using the antimatter material type. An antimatter material called "opening" is defined that "clips" all faces for the current block, and patches the edges of the hole with the material defined for the face "sill". If no rule is given specifically for the sill face, then the most specific material (ie. the material in the latest rule) for this block is used. An antimatter opening will not function properly if there is another surface intersecting it, or rendering is attempted from within the opening. Overlapping openings or openings with shared boundaries will also fail. There is currently no support of Architrion "frame" libraries.

Naming of the output faces is based on layer number, reference id and output block number (sequential from 1 to the total number of output blocks). If there is no reference id name, the layer name is used (if available) instead of the layer number. If there is a reference id number but no name, that is added as well. Names are truncated to the first 12 characters, so the ends of long names may be lost. Also, spaces in names are replaced by underscores ('_'). Finally, the face id is added to the end of the block name for each output polygon. An example identifier for a polygon is:

```
l3.window_overh.3155.ref
```

This would be the reference face of output block number 3155, reference id name "window overhangs" in layer number 3.

EXAMPLE

To create a qualifier list for building.txt:

```
arch2rad -n building.txt > building.qual
```

To translate building.txt into a *Radiance* file using the mapping building.map:

```
arch2rad -m building.map building.txt >
building.rad
```

To create an octree directly from an Architrion file using the default mapping and materials:

```
oconv source.rad /usr/local/lib/ray/lib/arch.mat
'!arch2rad building.txt' > building.oct
```

FILES

```
/usr/local/lib/ray/lib/arch.map
/usr/local/lib/ray/lib/arch.mat
```

AUTHOR

Greg Ward

SEE ALSO

`ies2rad(1)`, `oconv(1)`, `thf2rad(1)`, `xform(1)`

NAME

bgraph - do a set of batch graphs to a metafile

SYNOPSIS

bgraph [-type ..][+variable value ..][file ..]

DESCRIPTION

Bgraph reads each graph *file* in sequence and converts it to a plot suitable for use by a metafile driver program. If no files are given, the standard input is read.

The graph type can be defined with a -type option. Types are simply include files which set default values for certain variables. The actual include file name is the type concatanated with ".plt". Typical types are "scatter", "line", and "curve". A scatter graph shows only points. A line graph shows only lines connecting points. A curve graph shows both points and connecting lines.

Variables can be set explicitly with +variable value options. The following standard graph variables are supported:

fthick The frame thickness, valued from 0 to 4. A value of 0 turns the frame off.

grid The grid: 1 is on, 0 is off.

include The include file name. Graph input is taken from the file. If the file is not found in the current directory, it is searched for in a set of standard locations.

legend The legend title.

othick The origin axis thickness, valued from 0 to 4. A value of 0 turns the origin off.

period The period for a polar plot. For a plot in degrees, use 360. For radians, use 6.283. A value of 0 (the default) indicates a Cartesian plot.

subtitle The graph subtitle.

symfile The point symbol metafile.

tstyle The frame tick mark style. The default value is 1, which is outward tick marks. A value of 2 is inward ticks, 3 is cross ticks. A value of 0 disables frame tick marks.

title The graph title.

xlabel The x axis label.

xmap The x axis mapping function. An x axis mapping xmap(x)=log(x) produces a log x axis.

xmax The x axis maximum.

xmin The x axis minimum.

xstep The x axis step.

ylabel The y axis label.

ymap The y axis mapping function. An y axis mapping ymap(y)=log(y) produces a log y axis.

ymax The y axis maximum.

ymin The y axis minimum.

ystep The y axis step.

In addition to the standard graph variables, each curve has a set of variables. The variables for curve 'A' all begin with the letter 'A'; the variables for curve 'B' all begin with the letter 'B', and so on. Up to 8 curves are supported on a single graph, 'A' through 'H'. The variables for curve 'A' are:

A The function for curve 'A'. If Adata is undefined, *xmin*, *xmax* and *Anpoints* are used to determine which x values to plot. If *Adata* is defined and *A* is a function of a single variable (ie. A(x)), data values are interpreted as x values to be plotted. If *Adata* is defined and *A* is a function of two variables (ie. A(x,y)), data values are interpreted as (x,y) pairs and *A* becomes a mapping function for the data.

Acolor The color for curve A. The values 1-4 map to black, red, green, and blue respectively. A value of 0 turns curve A off.

Adata The point data for curve 'A'. If *Adata* is set to the name of a file, data is read and interpreted from that file. If *Adata* is set to a command (beginning with an exclamation, '!'), the output of the command is read as data. Otherwise, data is read from the current file. Data values are separated by white space and/or commas. A semicolon or end of file indicates the end of data.

Alabel The label for curve 'A'. The curve label is printed in the legend when a curve is defined.

Alintype The line type for curve 'A', valued from 0 to 4. A value of 0 turns line drawing off. A value of 1 is solid, 2 is dashed, 3 is dotted, and 4 is dotdashed.

Anpoints The number of symbol points for curve 'A'. If *Adata* is defined, all points will be connected with the selected curve line, but only *Anpoints* points will be indicated with a symbol. This prevents messy graphs when large number of points are defined. If *A* is defined and *Adata* is not, *Anpoints* is used along with *xmin* and *xmax* to determine which x values to plot.

Asymsize The symbol radius for curve 'A'. The default size is 100. A value of 0 turns symbols off.

Asymtype This is the name of a segment in symfile which defines the graphics symbol for curve 'A'.

Athick The line thickness for curve 'A', valued from 0 to 4. A thickness of 0 turns line drawing off.

GRAPH FILE FORMAT

A graph file contains definitions for graph and curve variables. These definitions fall one per line in the following formats:

```
vreal     = expression        # real variable
vfunction(x)   = expression(x)        # function
vstring   = "string"         # string variable
vdata     = filename         # data file
vdata     = "!command"       # data generator
variable  = continued \
line                         # newline escaped
vdata     =
# data v1 v2 v3 v4 ... ;
```

Comments are preceded by a '#', and continue to the end of the line. Except for comments, the newline can be escaped with a backslash. Note that in the special case where data is contained in the graph file, a definition will continue over more than one line. Data values can be separated by commas or white space, and reading continues until a semicolon is reached. No comments are allowed in the data section of a file.

An expression is an algebraic formula containing numbers, variables, functions, and the standard operators {+,,*,/,^,(,)} (see `calc(1)`). Besides the variables described in the last section, definitions of intermediate real variables and functions are allowed for convenience. They may be used in expressions of graph and curve variables.

EXAMPLE

A file to graph the sine function is:

```
title = "Sine Function from 0 to Pi"
PI = 3.141592653589793
A(x) = sin(x)
xmin = 0
xmax = PI
Anpoints = 100
```

Or, to graph selected points:

```
title = "Sine Function at 0, .2, .6, and .8" A(x)
= sin(x)
Adata =
0 , .2
6 , .8
;
```

The commands to plot these files might be:

```
bgraph -line sine1.plt | impress | ipr

bgraph -curve +ymin -1 +ymax 1 sine2.plt | t4014
```

FILES

```
/usr/local/lib/meta/*.mta
/usr/local/lib/meta/*.plt
*.plt
```

AUTHOR

Greg Ward

BUGS

There is no mechanism provided for undefining a variable. An axis mapping function which is not invertible (monotonically increasing or decreasing) confuses the program terribly.

SEE ALSO

calc(1), dgraph(1), gcomp(1), igraph(1), impress(1), metafile(5), mx80(1), mt160l(1), t4014(1), xmeta(1)

NAME

calc - calculator

SYNOPSIS

calc [file]

DESCRIPTION

Calc is a algebraic calculator designed primarily for interactive use. Each formula definition *file* is read and compiled. The standard input is then read, expressions are evaluated and results are sent to the standard output.

An expression contains real numbers, variable names, function calls, and the following operators:

```
     + - * / ^
```

Operators are evaluated left to right, except '^', which is right associative. Exponentiation has the highest precedence; multiplication and division are evaluated before addition and subtraction. Expressions can be grouped with parentheses. Each result is assigned a number, which can be used in future expressions. For example, the expression ($3*10) is the result of the third calculation multiplied by ten. A dollar sign by itself may be used for the previous result. All values are double precision real.

In addition, variables and functions can be defined by the user. A variable definition has the form:

```
     var = expression ;
```

Any instance of the variable in an expression will be replaced with its definition. A function definition has the form:

```
     func(a1, a2, ..) = expression ;
```

The expression can contain instances of the function arguments as well as other variables and functions. Function names can be passed as arguments. Recursive functions can be defined using calls to the defined function or other functions calling the defined function.

To define a constant expression, simply replace the equals sign ('=') with a colon (':') in a definition. Constant expressions are evaluated only once, the first time they are used. This avoids repeated evaluation of expressions whose values never change. Ideally, a constant expression contains only numbers and references to previously defined constant expressions and functions. Constant function definitions are are replaced by their value in any expression that uses them with constant arguments. All predefined functions and variables have the constant attribute. Thus, "sin(PI/4)" in an expression would be immediately replaced by ".707108" unless sin() or PI were redefined by the user. (Note that redefining constant expressions is not a recommended practice!)

A variable or function's definition can be displayed with the '?' command:

```
? name
```

If no name is given, all definitions are printed. The '>' command writes definitions to a file:

```
> file
```

Similarly, the '<' command loads definitions.

The following library of predefined functions and variables is provided:

PI the ratio of a circle's circumference to its diameter.

if(cond, then, else)
: if cond is greater than zero, then is evaluated, otherwise else is evaluated. This function is necessary for recursive definitions.

select(N, a1, a2, ..)
: return aN (N is rounded to the nearest integer). This function provides array capabilities. If *N* is zero, the number of available arguments is returned.

rand(x)
: compute a random number between 0 and 1 based on x.

floor(x)
: return largest integer not greater than x.

ceil(x)
: return smallest integer not less than x.

sqrt(x)
: return square root of x.

exp(x)
: compute e to the power of x (e approx = 2.718281828).

log(x)
: compute the logarithm of x to the base e.

log10(x)
: compute the logarithm of x to the base 10.

sin(x), cos(x), tan(x)
: trigonometric functions.

asin(x), acos(x), atan(x)
inverse trigonometric functions.

atan2(y, x)
inverse tangent of y/x (range -pi to pi).

AUTHOR

Greg Ward

SEE ALSO

`ev(1)`, `rcalc(1)`, `tabfunc(1)`

NAME

cnt - index counter

SYNOPSIS

cnt N ..

DESCRIPTION

Cnt counts from 0 to N-1, producing N lines of output. If multiple arguments are given, *cnt* produces a nested array of values where the final counter rotates fastest through its range. *Cnt* is most useful in conjunction with *rcalc*(*1*) to produce array values.

EXAMPLE

To create a 3 by 5 array:

```
cnt 3 5
```

AUTHOR

Greg Ward

SEE ALSO

`lam(1)`, `neat(1)`, `rcalc(1)`, `total(1)`

NAME

cv - convert between metafile formats

SYNOPSIS

cv [-r][file ..]

DESCRIPTION

Cv reads each human readable metafile *file* in sequence and converts it to a binary form. If the option *-r* is specified, the reverse conversion is performed.

If no input files are specified, the standard input is read.

EXAMPLE

To convert the binary file meta.bin to its human-readable equivalent, and put the result in meta.human

```
cv -r meta.bin > meta.human
```

AUTHOR

Greg Ward

SEE ALSO

`meta(3)`, `metafile(5)`, `pexpand(1)`, `psort(1)`

NAME

dayfact - compute illuminance and daylight factor on workplane

SYNOPSIS

dayfact [falsecolor options]

DESCRIPTION

Dayfact is an interactive script for computing workplane illuminance, and daylight factors and potential daylight savings using *rtrace*(*1*). The script *falsecolor*(*1*) is then used to draw contour lines on the resulting *Radiance* picture.

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

`falsecolor(1)`, `glare(1)`, `rtrace(1)`, `ximage(1)`

NAME

dgraph - do a set of graphs to a dumb terminal

SYNOPSIS

dgraph [-w width][-l length][+variable value ..][file ..]

DESCRIPTION

Dgraph reads each graph *file* in sequence and converts it to a character plot displayable on any ascii device. If no files are given, the standard input is read.

Across the top of the plot, the extrema are printed. This is the only indication of the axis size. Curves are represented with their respective letter ('A' for curve A, etc.) at each point. Where two or more curves cross, a number is shown instead.

The size of the output array can be specified as a certain *width* and *length*. The default size is 79 by 22.

Variables can be set explicitly with +variable value options. See `bgraph(1)` for details.

EXAMPLE

To get a quick glimpse of the sine function from 0 to 4.

```
dgraph
 A(x)=sin(x)
 Anpoints=100
 xmin=0
 xmax=4
 ^D
```

AUTHOR

Greg Ward

BUGS

There is no mechanism provided for undefining a variable.

SEE ALSO

`bgraph(1)`, `calc(1)`, `gcomp(1)`, `igraph(1)`

NAME

ev - evaluate expressions

SYNOPSIS

ev 'expr' ..

DESCRIPTION

Ev evaluates expressions given on the command line, and sends the results to the standard output, one per line. An expression contains real numbers, function calls, and the following operators:

```
+ - * / ^
```

Operators are evaluated left to right, except '^', which is right associative. Powers have the highest precedence; multiplication and division are evaluated before addition and subtraction. Expressions can be grouped with parentheses. All values are double precision real.

The following library of functions is available:

if(cond, then, else)
: if cond is greater than zero, then is evaluated, otherwise else is evaluated. This function is necessary for recursive definitions.

select(N, a1, a2, ..)
: return aN (N is rounded to the nearest integer). This function provides array capabilities. If *N* is zero, the number of available arguments is returned.

rand(x)
: compute a random number between 0 and 1 based on x.

floor(x)
: return largest integer not greater than x.

ceil(x)
: return smallest integer not less than x.

sqrt(x)
: return square root of x.

exp(x)
: compute e to the power of x (e approx = 2.718281828).

log(x)
: compute the logarithm of x to the base e.

log10(x)
: compute the logarithm of x to the base 10.

sin(x), cos(x), tan(x)
trigonometric functions.

asin(x), acos(x), atan(x)
inverse trigonometric functions.

atan2(y, x)
inverse tangent of y/x (range -pi to pi).

EXAMPLE

To pass the square root of two and the sine of .5 to a program:

```
program `ev 'sqrt(2)' 'sin(.5)'`
```

AUTHOR

Greg Ward

SEE ALSO

calc(1), rcalc(1)

NAME

falsecolor - make a false color *Radiance* picture

SYNOPSIS

falsecolor [-i input][-p picture][-cb | -cl][-e][-s scale][-l label][-n ndivs][-log decades][-m mult][-r redv][-g grnv][-b bluv]

DESCRIPTION

Falsecolor produces a false color picture for lighting analysis. Input is a rendered *Radiance* picture.

By default, luminance is displayed on a linear scale from 0 to 1000 nits, where dark areas are blue and brighter areas move through the spectrum to red. A different scale can be given with the -*s* option. The default multiplier is 179, which converts from radiance or irradiance to luminance or illuminance, respectively. A different multiplier can be given with -*m* to get daylight factors or whatever. For a logarithmic rather than a linear mapping, the -*log* option can be used, where *decades* is the number of decades below the maximum scale desired.

A legend is produced for the new image with a label given by the -*l* option. The default label is "Nits", which is appropriate for standard *Radiance* pictures. If the -*i* option of *rpict*(*1*) was used to produce the image, then the appropriate label would be "Lux".

If contour lines are desired rather than just false color, the -*cl* option can be used. These lines can be placed over another *Radiance* picture using the -*p* option. If the input picture is given with -*ip* instead of -*i*, then it will be used both as the source of values and as the picture to overlay with contours. The -*cb* option produces contour bands instead of lines, where the thickness of the bands is related to the rate of change in the image. The -*n* option can be used to change the number of contours (and corresponding legend entries) from the default value of 8.

The -e option causes extrema points to be printed on the brightest and darkest pixels of the input picture.

The remaining options, -r, -g, and -b are for changing the mapping of values to colors. These are expressions of the variable *v*, where *v* varies from 0 to 1. These options are not recommended for the casual user.

If no -i or -ip option is used, input is taken from the standard input. The output image is always written to standard output, which should be redirected.

EXAMPLES

To create a false color image directly from *rpict*(*1*):

```
rpict -vf default.vp scene.oct | falsecolor >
scene.pic
```

To create a logarithmic contour plot of illuminance values on a *Radiance* picture:

```
rpict -i -vf default.vp scene.oct > irrad.pic
rpict -vf default.vp scene.oct > rad.pic
falsecolor -i irrad.pic -p rad.pic -cl -log 2 -l
Lux > lux.pic
```

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

`getinfo(1)`, `pcomb(1)`, `pcompos(1)`, `pextrem(1)`, `pfilt(1)`, `pflip(1)`, `protate(1)`, `psign(1)`, `rpict(1)`, `ximage(1)`

NAME

findglare - locate glare sources in a *Radiance* scene

SYNOPSIS

findglare [-v][-ga angles][-t threshold][-r resolution][-c][-p picture][view options] [[rtrace options] octree]

DESCRIPTION

Findglare locates sources of glare in a specific set of horizontal directions by computing luminance samples from a *Radiance* picture and/or octree. *Findglare* is intended primarily as a preprocessor for glare calculation programs such as *glarendx*(*1*), and is usually accessed through the executive script *glare*(*1*).

If only an octree is given, findglare calls rtrace to compute the samples it needs. If both an octree and a picture are specified, *findglare* calls rtrace only for samples that are outside the frame of the picture. If *findglare* does not have an octree and the picture does not completely cover the area of interest, a warning will be issued and everything outside the picture will be treated as if it were black. It is preferable to use a picture with a fisheye view and a horizontal and vertical size of at least 180 degrees (more horizontally if the *-ga* option is used -- see below). Note that the picture file must contain correct view specifications, as maintained by *rpict*(*1*), *rview*(*1*), *pfilt*(*1*) and *pinterp*(*1*). Specifically, *findglare* will not work on pictures processed by *pcompos*(*1*) or *pcomb*(*1*). It is also essential to give the proper rtrace options when an octree is used so that the calculated luminance values are correct.

The output of findglare is a list of glare source directions, solid angles and average luminances, plus a list of indirect vertical illuminance values as a function of angle. Angles are measured in degrees from the view center, with positive angles to the left and negative angles to the right.

By default, findglare only computes glare sources and indirect vertical illuminance for the given view (taken from the picture if none is specified). If the view direction is not horizontal to begin with (ie. perpendicular to the view up vector), *findglare* will substitute the closest horizontal direction as its view center. The *-ga* option can be used to specify a set of directions to consider about the center of view. This specification is given by a starting angle, ending angle, and step angle like so:

start-end:step

All angles must be whole degrees within the range 1 to 180. Multiple angle ranges may be separated by commas, and individual angles may be given without the ending and step angles. Note that *findglare* will complain if the same angle is given twice either explicitly or implicitly by two ranges.

Findglare normally identifies glare sources as directions that are brighter than 7 times the average luminance level. It is possible to override this determination by giving an explicit luminance threshold with the *-t* option. It usually works best to use the 'l' command within *ximage*(*1*) to decide what this value should be. Alternatively, one can use the 't' command within *rview*(*1*). The idea is to pick a threshold that is well above the average level but smaller than the source areas.

If the sources in the scene are small, it may be necessary to increase the default sample resolution of *findglare*(*1*) using the *-r* option. The default resolution is 150 vertical samples and a proportional number of horizontal samples. If besides being small, the sources are not much brighter than the threshold, the *-c* flag should be used to override *findglare*'*s* default action of absorbing small sources it deems to be insignificant.

The -v flag switches on verbose mode, where findglare reports its progress during the calculation.

EXAMPLE

To calculate the glare sources in the image "scene.pic":

```
findglare -p scene.pic > scene.glr
```

To compute the Guth visual comfort probability from this result:

```
glarendx -t guth_vcp scene.glr
```

To compute the glare for a set of angles around the view "good.vp" from the octree "scene.oct" using an ambient level of .1:

```
findglare -vf good.vp -ga 10-60:10 -av .1 .1 .1
scene.oct > scene.glr
```

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

getinfo(1), glare(1), glarendx(1), pfilt(1), rpict(1), rtrace(1), rview(1), xglaresrc(1), ximage(1)

NAME

gcomp - do computations on a graph file.

SYNOPSIS

gcomp [-amilh][+variable value ..][file ..]

DESCRIPTION

Gcomp reads each graph *file* in sequence and computes the specified calculations. The type options are as follows:

-n	Print the name of each curve.
-a	Print average and standard deviation of each curve.
-m	Print minimum and maximum for each curve.
-i	Print Romberg's approximation to the integral of each curve.
-l	Print the slope, intercept, and correlation coefficient using the least squares method of linear regression.
-h	Do not print a header in the output.

The calculations will be displayed as columns in the order they are specified on the command line. If no files are given, the standard input is read.

Variables can be set explicitly with +variable value options. The only truely useful variables for this program are xmin and xmax. They determine boundaries for the calculations.

EXAMPLE

```
 To compute the approximate integral of
sin(x)/log(x) from 2 to 4:
 gcomp -i
 A(x)=sin(x)/log(x);
 Anpoints=100;
 xmin=2;
 xmax=4;
 ^D
```

AUTHOR

Greg Ward

BUGS

Only the y values can be used for computation.

SEE ALSO

`bgraph(1)`, `calc(1)`, `dgraph(1)`, `igraph(1)`

NAME

genblinds - generate a *Radiance* description of venetian blinds

SYNOPSIS

genblinds mat name depth width height nslats angle [-r|+r rcurv]

DESCRIPTION

Genblinds produces a *Radiance* scene description of a set of venetian blinds. The *depth* of the blinds (X dimension) is given first, followed by the *width* (Y dimension), followed by the *height* (Z dimension). The number of slats to place evenly within this height is given as *nslats*. The *angle* of the blind, where zero is perfectly horizontal and a positive angle tilts the positive X edge upwards, is given in degrees. The blinds are initially situated so that the corner of the bottom blind is *height*/*nslats*/*2* above the XY plane, and all coordinates are positive. Each new slat is placed *height*/*nslats* above the previous one, until the top slat is at *height* - *height*/*nslats*/*2* . The blinds may of course be moved from this starting point with the *xform*(*1*) command.

If curved blinds are desired, a radius of curvature may be given with the +/-*r* option. If given as -*r*, The curvature is downward (which is the usual configuration). If the option is given as +*r*, then the curvature is upward. The radius indicates how far from each slat its effective cylindrical center resides. Each slat will be broken into as many polygons as is necessary to keep the delta changes in angle less than 10 degrees. (Note that this may result in a rather large number of polygons.)

EXAMPLE

To produce a curved set of blinds with 15 slats:

```
genblinds white blind 1 46 88 118 15 -r 1 >
blinds.rad
```

AUTHOR

Jean-Louis Scartezzini and Greg Ward

SEE ALSO

`genbox(1)`, `genrev(1)`, `gensurf(1)`, `genworm(1)`, `rpict(1)`, `rview(1)`, `xform(1)`

NAME

genbox - generate a *Radiance* description of a box

SYNOPSIS

genbox mat name xsiz ysiz zsiz [-i][-r rad | -b bev]

DESCRIPTION

Genbox produces a *Radiance* scene description of a parallelepiped with one corner at the origin and the opposite corner at (*xsiz*, *ysiz*, *zsiz*). The sides of the box will be parallel to the three coordinate planes. The surfaces that make up the box will be modified by *mat* and their identifiers will begin with *name*. The *-i* option can be used to produce a box with inward directed surface normals. The *-r* option can be used to specify the radius for rounded edges. The *-b* option can be used to specify the indentation for beveled edges.

EXAMPLE

To produce a rectangular box made of wood with beveled edges:

```
genbox wood box1 5 8 3 -b .5 > box1
```

AUTHOR

Greg Ward

BUGS

Because spheres and cylinders are used to construct boxes with rounded edges, a transparent box of this type appears quite messy.

SEE ALSO

`genrev(1)`, `gensurf(1)`, `genworm(1)`, `rpict(1)`, `rview(1)`, `xform(1)`

NAME

genclock - generate a *Radiance* description of a clock

SYNOPSIS

genclock [-f face_mat][-c case_mat][-n name] { HH:MM hours }

DESCRIPTION

Genclock produces a *Radiance* scene description of an analog clock showing the given hour. The hour may either be given as *HH*:*MM* or decimal *hours*.

The face of the clock will have a radius of 1.0 units, with the surrounding case 1.1 (2.2 diameter). The origin is at the center of the back, and the face looks in the positive X-direction. The 12 o'clock direction corresponds to the positive Z-axis. (The Y-axis direction is 3 o'clock.) The *xform*(*1*) command may be used to resize and relocate the clock as desired.

Normally, genclock produces all of the materials necessary for its own description, but options are provided to specify alternate materials for the face and case. The numbers on the face are in dark lettering, so the face material must be relatively light for them to show up well. By default, the clock is given the name ``clock,'' but this may be changed with the -*n* option.

EXAMPLE

To produce a 12 inch diameter clock showing 10:35 and hang it at 60 on a wall facing the Y-direction at Y=10:

genclock 10:35 | xform -s 6 -rz 90 -t 20 10 60

AUTHOR

Greg Ward

SEE ALSO

`genbox(1)`, `genrev(1)`, `gensurf(1)`, `genworm(1)`, `rpict(1)`, `rview(1)`, `xform(1)`

NAME

genprism - generate a *Radiance* description of a prism

SYNOPSIS

genprism mat name { - | vfile | N v1 v2 .. vN } [-l lvect][-r radius][-c][-e]

DESCRIPTION

Genprism produces a *Radiance* scene description of a prism, or extruded polygon. The polygon to extrude lies in the z==0 plane, and is given as a list of (x,y) pairs on the standard input (-), or from the file *vfile*, or on the command line preceded by the number of vertices, *N*. The order of the vertices and the extrusion vector *lvect* (default (0,0,1)) determine the surface orientations. The surfaces that make up the prism will be modified by *mat* and their identifiers will begin with *name*. The -*r* option may be used to round the corners of the object using spheres and cylinders. The -*c* option inhibits generation of a face connecting the last vertex to the first. The -*e* option inhibits generation of the end polygons.

EXAMPLE

To produce a equilateral triangular prism:

```
genprism clear_plastic prism 3 0 0 .5 .866 1 0
```

AUTHOR

Greg Ward

BUGS

The rounding option only works for opaque prisms with outward facing normals. If the normals face inward, the appearance will be bizarre.

SEE ALSO

`genbox(1)`, `genrev(1)`, `gensurf(1)`, `genworm(1)`, `rpict(1)`, `rview(1)`, `xform(1)`

NAME

genrev - generate a *Radiance* description of surface of revolution

SYNOPSIS

genrev mat name 'z(t)' 'r(t)' nseg [-e expr][-f file][-s]

DESCRIPTION

Genrev produces a *Radiance* scene description of a surface of revolution. The object will be composed of *nseg* cones, cups, cylinders, tubes or rings following the parametric curve defined by *z*(*t*) (height) and *r*(*t*) (radius). When *z* is increasing with t, the surface normal points outward. When *z* is decreasing, the normal points inward. The variable *t* used in the function expressions varies from 0 to 1 in even steps of *1*/*nseg*. The expressions are of the same type used in *Radiance* function files. Auxiliary expressions and/or files may be specified in any number of -*e* and -*f* options. The -*s* option smooths the surfaces using Phong normal interpolation.

EXAMPLE

To generate a torus with an inner radius of 1 and an outer radius of 3:

```
genrev steel torus 'sin(2*PI*t)' '1+cos(2*PI*t)'
32
```

AUTHOR

Greg Ward

BUGS

The -s option doesn't modify the surface normal correctly for the opposite side.

SEE ALSO

`calc(1)`, `genbox(1)`, `gensurf(1)`, `genworm(1)`, `rpict(1)`, `rview(1)`, `xform(1)`

NAME

gensky - generate a *Radiance* description of the sky

SYNOPSIS

gensky month day time [options]

gensky -ang altitude azimuth [options] gensky -defaults

DESCRIPTION

Gensky produces a *Radiance* scene description for the CIE standard sky distribution at the given month, day and time. By default, the time is interpreted as local standard time on a 24-hour clock. The time value may be given either as decimal hours, or using a colon to separate hours and minutes. If the time is immediately followed (no white space) by a North American or European time zone designation, then this determines the standard meridian, which may be specified alternatively with the *-m* option. The following time zones are understood, with their corresponding hour differences from Greenwich Mean Time:

Standard time:

YST	PST	MST	CST	EST	AST	NST	GMT	WET	MET	MEZ
9	8	7	6	5	4	3.5	0	-1	-2	-2

Daylight savings time:

YDT	PDT	MDT	CDT	EDT	ADT	NDT	BST	WETDST	METDST	MESZ
8	7	6	5	4	3	2.5	-1	-2	-3	-3

If the time is preceded by a plus sign ('+'), then it is interpreted as local solar time instead. It is very important to specify the correct latitude and longitude using the *-a* and *-o* options to get the correct solar angles.

The second form gives the solar angles explicitly. The altitude is measured in degrees above the horizon, and the azimuth is measured in degrees west of South.

The third form prints the default option values.

The output sky distribution is given as a brightness function, *skyfunc*. Its value is in watts/steradian/meter2. The x axis points east, the y axis points north, and the z axis corresponds to the zenith. The actual material and surface(s) used for the sky is left up to the user. For a hemispherical blue sky, the description might be:

```
!gensky 4 1 14

skyfunc glow skyglow
0
0
4 .9 .9 1 0

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

Often, skyfunc will actually be used to characterize the light coming in from a window.

In addition to the specification of a sky distribution function, *gensky* suggests an ambient value in a comment at the beginning of the description to use with the *-av* option of the *Radiance* rendering programs. (See `rview(1)` and `rpict(1)`.) This value is the cosine-weighted radiance of the sky in watts/steradian/meter2.

Gensky supports the following options.

-s Sunny sky without sun. The sky distribution will correspond to a standard CIE clear day.

+s Sunny sky with sun. In addition to the sky distribution function, a source description of the sun is generated.

-c Cloudy sky. The sky distribution will correspond to a standard CIE overcast day.

-i Intermediate sky without sun. The sky will correspond to a standard CIE intermediate day.

+i Intermediate sky with sun. In addition to the sky distribution, a (somewhat subdued) sun is generated.

-u Uniform cloudy sky. The sky distribution will be completely uniform.

-g *rfl*

Average ground reflectance is *rfl*. This value is used to compute *skyfunc* when Dz is negative. Ground plane brightness is the same for *-s* as for *+s*. (Likewise for *-i* and *+i*, but see the *-r* option below.)

-b *brt*

The zenith brightness is *brt*. Zenith radiance (in watts/steradian/meter2) is normally computed from the sun angle and sky turbidity (for sunny sky). It can be given directly instead, using this option.

-B *irrad*

Same as *-b*, except zenith brightness is computed from the horizontal diffuse irradiance (in watts/meter2).

-r *rad*

The solar radiance is *rad*. Solar radiance (in watts/steradian/meter2) is normally computed from the solar altitude. This option may be used to override the default calculation. If a value of zero is given, no sun description is produced, and the contribution of direct solar to ground brightness is neglected.

-R *irrad*

Same as *-r*, except solar radiance is computed from the horizontal direct irradiance (in watts/meter2).

-t *trb*

The turbidity factor is *trb*. Greater turbidity factors correspond to greater atmospheric scattering. A turbidity factor of 1.0 indicates an ideal clear atmosphere (i.e. a completely dark sky). Values less than 1.0 are physically impossible.

The following options do not apply when the solar altitude and azimuth are given explicitly.

-a lat

The site latitude is *lat* degrees north. (Use negative angle for south latitude.) This is used in the calculation of sun angle.

-o *lon*

The site longitude is *lon* degrees west. (Use negative angle for east longitude.) This is used in the calculation of solar time and sun angle. Be sure to give the corresponding standard meridian also! If solar time is given directly, then this option has no effect.

-m *mer*

The site standard meridian is *mer* degrees west of Greenwich. (Use negative angle for east.) This is used in the calculation of solar time. Be sure to give the correct longitude also! If solar time is given directly, then this option has no effect.

EXAMPLE

To produce a sunny sky for July 4th at 2:30pm Eastern daylight time at a site latitude of 42 degrees, 89 degrees west longitude:

```
gensky 7 4 14:30EDT +s -a 42 -o 89
```

To produce a sunny sky distribution for a specific sun position but without the sun description:

```
gensky -ang 23 -40 -s
```

FILES

```
/usr/local/lib/ray/skybright.cal
```

AUTHOR

Greg Ward

SEE ALSO

`rpict(1)`, `rview(1)`, `xform(1)`

NAME

gensurf - generate a *Radiance* description of a curved surface

SYNOPSIS

gensurf mat name 'x(s,t)' 'y(s,t)' 'z(s,t)' m n [-e expr][-f file][-s]
gensurf mat name 'x(s,t)' 'y(s,t)' dfile m n [-e expr][-f file][-s]
gensurf mat name dfile dfile dfile m n [-s]

DESCRIPTION

Gensurf produces a *Radiance* scene description of a functional surface defined by the parametric equations *x*(*s*,*t*), *y*(*s*,*t*), and *z*(*s*,*t*). The surface normal is defined by the right hand rule as applied to (*s*,*t*). *S* will vary from 0 to 1 in steps of *1*/*m*, and *t* will vary from 0 to 1 in steps of *1*/*n*. The surface will be composed of 2*m*n or fewer triangles and quadrilaterals. The expressions are of the same type used in *Radiance* function files. Auxiliary expressions and/or files may be specified in any number of -*e* and -*f* options. The -*s* option adds smoothing (surface normal interpolation) to the surface.

The second invocation form reads z data values from the file *dfile*. This file must give either m*n or (m+1)*(n+1) floating point z values. If m*n values are given, then the values correspond to the centroid of each quadrilateral region. If (m+1)*(n+1) values are given, then the values correspond to the vertices of each quadrilateral region. The ordering of the data in the file is such that the s values are changing faster than the t values. If a minus ('-') is given for *dfile*, then the values are read from the standard input.

The third invocation form is used to read coordinate triplets from a file or the standard input. The three *dfile* arguments must all be the same, and the corresponding file must contain three floating point values for each point location. The ordering and other details are the same as those described for z value files above.

EXAMPLE

To generate a tesselated sphere:

```
gensurf crystal ball 'sin(PI*s)*cos(2*PI*t)'
'cos(PI*s)' 'sin(PI*s)*sin(2*PI*t)' 7 10
```

To generate a 10x20 smoothed height field from 12 recorded vertex z values:

```
gensurf dirt ground '10*s' '20*t' height.dat 2 3
-s
```

AUTHOR

Greg Ward

BUGS

The smoothing operation requires that functions be defined beyond the [0,1] boundaries of s and t.

SEE ALSO

`calc(1)`, `genbox(1)`, `genrev(1)`, `genworm(1)`, `rpict(1)`, `rview(1)`, `xform(1)`

NAME

genworm - generate a *Radiance* description of a functional worm

SYNOPSIS

genworm mat name 'x(t)' 'y(t)' 'z(t)' 'r(t)' nseg [-e expr][-f file]

DESCRIPTION

Genworm produces a *Radiance* scene description of a worm defined by the parametric equations *x*(*t*), *y*(*t*), *z*(*t*), and *r*(*t*) (the radius). *T* will vary from 0 to 1 in steps of *1*/*nseg*. The surface will be composed of *nseg* cones or cylinders and *nseg+1* spheres. The expressions are of the same type used in *Radiance* function files. Auxiliary expressions and/or files may be specified in any number of *-e* and *-f* options.

EXAMPLE

To generate a banana:

```
genworm yellow banana '0' '5*sin(t)' '5*cos(t)'
'.4-(.5t)*(.5-t)' 20
```

AUTHOR

Greg Ward

BUGS

Since the worm is constructed of intersecting surfaces, only opaque materials should be used with this object. Also, a worm cannot double back inside itself without making a mess.

SEE ALSO

`calc(1)`, `genbox(1)`, `genrev(1)`, `gensurf(1)`, `rpict(1)`, `rview(1)`, `xform(1)`

NAME

getbbox - compute bounding box for *Radiance* scene

SYNOPSIS

getbbox [-w][-h] [input ..]

DESCRIPTION

Getbbox reads each scene description *input* and computes the minimum axis-aligned parallelopiped that will enclose all of the objects. Each *input* can be either a file name, or a command (enclosed in quotes and preceded by a '!'). If no arguments are given, the standard input is read. A hyphen ('-') can also be used to indicate the standard input.

The -w option suppresses warnings. The -h option suppresses the header line "xmin xmax ymin ymax zmin zmax".

EXAMPLE

To compute the bounding box for the object "thingy":

```
getbbox thingy
```

To preview "scene":

```
preview -v FOUR -b `getbbox -h scene' scene`
```

NOTES

Since expanding a scene can require considerable overhead, it is better to use the bounding cube produced by *oconv*(*1*) and read by *getinfo*(*1*) if an octree exists for the scene. However, there are certain circumstances, such as foreign object placement, that require knowing the bounding box rather than just the bounding cube.

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

getinfo(1), mgf2meta(1), oconv(1), xform(1)

NAME

getinfo - get header information from a *Radiance* file

SYNOPSIS

getinfo [-d][file ..]

getinfo -

DESCRIPTION

Getinfo reads the header of each *Radiance file* and writes it to the standard output. Octree and picture files are in a binary format, which makes it difficult to determine their content. Therefore, a few lines of text are placed at the beginning of each file by the *Radiance* program that creates it. The end of the header information and the start of the data is indicated by an empty line. The *-d* option can be used to print the dimensions of an octree or picture file instead. For an octree, *getinfo -d* prints the bounding cube (xmin ymin zmin size). For a picture, *getinfo -d* prints the y and x resolution (-Y yres +X xres). If no *file* is given, the standard input is read.

The second form of getinfo with a hyphen simply removes the header and copies the body of the file from the standard input to the standard output.

EXAMPLE

To print the header information from scene1.oct and scene2.pic:

```
getinfo scene1.oct scene2.pic
```

AUTHOR

Greg Ward

SEE ALSO

`oconv(1)`, `pfilt(1)`, `rpict(1)`, `rview(1)`

NAME

glare - perform glare and visual comfort calculations

SYNOPSIS

glare [glarefile [picture [octree]]]

DESCRIPTION

Glare is an interactive script for executing programs used to locate glare sources and compute glare indices and visual comfort probability. If no *glarefile* is given, the program prompts the user for a one. If the file does not exist, *glare* asks the user some questions about the scene in question then runs *findglare*(*1*) to compute values to store in the file. *Glare* then presents the user a menu of available glare index calculations. After choosing a calculation, *glare* offers to store the result (usually not useful) or plot the information (but only for multiple glare angles).

If you are creating a new glarefile, it usually works best to start with a displayed image for reference during the interrogation.

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

`dayfact(1)`, `findglare(1)`, `glarendx(1)`, `igraph(1)`, `rpict(1)`, `xglaresrc(1)`, `ximage(1)`

NAME

glarendx - calculate glare index

SYNOPSIS

glarendx -t type [-h] [glarefile]

DESCRIPTION

Glarendx computes the selected glare index *type* from the given *glarefile* produced by *findglare*(*1*). *Glarendx* computes one value for each (indirect illuminance) angle in the input file. If no *glarefile* is given, *glarendx* reads from the standard input. The *-h* option can be used to remove the information header from the output.

Glarendx understands the following arguments for type:

guth_vcp
: Guth Visual Comfort Probability

cie_cgi
: CIE Glare Index (Einhorn)

ugr
: Unified Glare Rating

brs_gi
: Glare Index

dgi
: Daylight Glare Index

guth_dgr
: Guth Disability Glare Rating

vert_dir
: Direct Vertical Illuminance

vert_ill
: Total Vertical Illuminance

vert_ind
: Indirect Vertical Illuminance

Alternatively, a symbolic or hard link to the program with any of the above names may be used in place of the *type* argument.

AUTHORS

Greg Ward, Raphael Compagnon

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

`findglare(1)`, `glare(1)`, `igraph(1)`, `rpict(1)`, `xglaresrc(1)`, `ximage(1)`

NAME

histo - compute 1-dimensional histogram of N data columns

SYNOPSIS

histo xmin xmax nbins

histo imin imax

DESCRIPTION

Histo bins columnular data on the standard input between the given minimum and maximum values. If three command line arguments are given, the third is taken as the number of data bins between the first two real numbers. If only two arguments are given, they are both assumed to be integers, and the number of data bins will be equal to their difference plus one. The bins are always of equal size.

The output is N+1 columns of data (for N columns input), where the first column is the centroid of each division, and each row corresponds to the frequencies for each column around that value.

All input data is interpreted as real values, and columns must be white-space separated. If any value is less than the minimum or greater than the maximum, it will be ignored on the input. (I.e., it will not contribute to any frequency count.)

EXAMPLE

To count data values between -1 and 1 in 50 bins:

```
histo -1 1 50 < input.dat
```

To count frequencies of integers between 0 and 255:

```
histo 0 255 < input.dat
```

AUTHOR

Greg Ward

SEE ALSO

`cnt(1)`, `lam(1)`, `neat(1)`, `rcalc(1)`, `tabfunc(1)`, `total(1)`

NAME

ies2rad - convert IES luminaire data to *Radiance* description

SYNOPSIS

ies2rad [options] [input ..]

DESCRIPTION

Ies2rad converts one or more IES luminaire data files to the equivalent *Radiance* scene description. The light source geometry will always be centered at the origin aimed in the negative z direction, with the 0 degree plane along the x axis. Usually, two output files will be created for every input file, one scene file (with a ".rad" suffix) and one data file (with a ".dat" suffix). If the IES input file includes tilt data, then another data file will be created (with a "+.dat" suffix). If the *-s* option is used, the scene data will be sent to the standard output instead of being written to a file. Since the data file does not change with other options to *ies2rad*, this is a convenient way to specify different lamp colors and multipliers inline in a scene description. If the *-g* option is used, then an octree file will be created (with the ".oct" suffix). The root portion of the output file names will be the same as the corresponding input file, unless the *-o* option is used. The output files will be created in the current directory (no matter which directory the input files came from) unless the *-l* or *-p* options are used.

Ies2rad assigns light source colors based on information in a lamp lookup table. Since most lamps are distinctly colored, it is often desirable to override this lookup procedure and use a neutral value that will produced colorbalanced renderings. In general, it is important to consider lamp color when an odd assortment of fixture types is being used to illuminate the same scene, and the rendering can always be balanced by `pfilt(1)` to a specific white value later.

-l libdir Set the library directory path to libdir. This is where all relative pathnames will begin for output file names. For light sources that will be used by many people, this should be set to some central location included in the RAYPATH environment variable. The default is the current working directory.

-p *prefdir*

Set the library subdirectory path to *prefdir*. This is the subdirectory from the library where all output files will be placed. It is often most convenient to use a subdirectory for the storage of light sources, since there tend to be many files and placing them all in one directory is very messy. The default value is the empty string.

-o *outname*

Set the output file name root to *outname*. This overrides the default output file name root which is the same as the input file. This option may be used for only one input file, and is required when reading data from the standard input.

-s

Send the scene information to the standard output rather than a separate file. This is appropriate when calling *ies2rad* from within a scene description via an inline command. The data file(s) will still be written based on the output file name root, but since this information is unaffected by command line options, it is safe to have multiple invocations of *ies2rad* using the same input file and different output options. The *-s* option may be used for only one input file.

-d*units*

Output dimensions are in *units*, which is one of the letters 'm', 'c', 'f', or 'i' for meters, centimeters, feet or inches, respectively. The letter specification may be followed by a slash ('/') and an optional divisor. For example, *-dm/1000* would be millimeters. The default output is in meters, regardless of the original units in the IES input file. Note that there is no space in this option.

-i *rad*

Ignore the crude geometry given by the IES input file and use instead an illum sphere with radius *rad*. This option may be useful when the user wishes to add a more accurate geometric description to the light source model, though this need is obviated by the recent LM-63-1995 specification, which uses MGF detail geometry. (See *-g* option below.)

-g

If the IES file contains MGF detail geometry, compile this geometry into a separate octree and create a single instance referencing it instead of including the converted geometry directly in the *Radiance* output file. This can result in a considerable memory savings for luminaires which are later duplicated many times in a scene, though the appearance may suffer for certain luminaires since the enclosed glow sources will not light the local geometry as they would otherwise.

-f *lampdat*

Use *lampdat* instead of the default lamp lookup table (lamp.tab) to map lamp names to xy chromaticity and lumen depreciation data. It is often helpful to have customized lookup tables for specific manufacturers and applications.

-t *lamp*

Use the given lamp type for all input files. Normally, *ies2rad* looks at the header lines of the IES file to try and determine what lamp is being used in the fixture. If any of the lines is matched by a pattern in the lamp lookup table (see the -f option above), that color and depreciation factor will be used instead of the default (see the -c and -u options). The *lamp* specification is also looked up in the lamp table unless it is set to "default", in which case the default color is used instead.

-c *red grn blu*

Use the given color if the type of the lamp is unknown or the -t option is set to "default". If unspecified, the default color will be white.

-u *lamp*

Set the default lamp color according to the entry for *lamp* in the lookup table (see the -f option). This is the color that will be used if the input specification does not match any lamp type patterns. This option is used instead of the -c option.

-m *factor*

Multiply all output quantities by *factor*. This is the best way to scale fixture brightness for different lamps, but care should be taken when this option is applied to multiple files.

EXAMPLE

To convert a single IES data file in inches with color balanced output and 15% lumen depreciation, creating the files "fluor01.rad" and "fluor01.dat" in the current directory:

```
ies2rad -di -t default -m .85 fluor01.ies
```

To convert three IES files of various types to tenths of a foot and put them in the library "/usr/local/lib/ray" subdirectory "source/ies":

```
ies2rad -df/10 -l /usr/local/lib/ray -p source/ies
ies01 ies02 ies03
```

To convert a single file and give the output a different name:

```
     ies2rad -o fluorescent ies03
```

ENVIRONMENT

RAYPATH	directories to search for lamp lookup table

AUTHOR

Greg Ward

BUGS

In pre-1991 standard IES files, all header lines will be examined for a lamp table string match. In post-1991 standard files, only those lamps with the [LAMP] or [LAMPCAT] keywords will be searched. The first match found in the file is always the one used. This method of assigning colors to fixtures is less than perfect, and the IES would do well to include explicit spectral information somehow in their specification.

The IESNA LM-63 specification prior to 1995 provided three basic source shapes, rectangular, round, and elliptical. The details of these shapes is vague at best. Rectangular sources will always be rectangular, but ies2rad will approximate round sources as spherical if the height is close to or greater than the width and length, and as a ring otherwise. Elliptical sources are treated the same as round sources. The 1995 standard rectifies this problem by including detailed luminaire geometry as MGF data, though nothing in the standard requires manufacturers to provide this information.

SEE ALSO

`mgf2rad(1)`, `oconv(1)`, `pfilt(1)`, `rad2mgf(1)`, `rpict(1)`, `xform(1)`

NAME

igraph - interactive graphing program

DESCRIPTION

Igraph is a crude interactive program for creating line and scatter plots. Provisions for reading and writing graph files as well as changing variables and getting output are provided from a menu.

Graph files and variables are as described in bgraph(1).

AUTHOR

Greg Ward

BUGS

There is no mechanism provided for undefining a variable.

SEE ALSO

`bgraph(1)`, `calc(1)`, `impress(1)`, `metafile(5)`, `mx80(1)`, `mt160l(1)`, `t4014(1)`

NAME

imagew - output metafile to Apple Imagewriter

SYNOPSIS

imagew [-c | -r] file ..

DESCRIPTION

Imagew reads each metafile *file* in sequence and converts it to output suitable for the Apple Imagewriter dot-matrix printer. If the option *c* is specified, the input files are only conditioned for output, ie. expanded and sorted (see pexpand and psort). This is useful if many copies of the same output is desired. If the option *r* is instead specified, the input is assumed already to be conditioned. If no input files are specified, the standard input is read.

-c	Condition the input only.
-r	Input is already conditioned, output only.

EXAMPLE

To print the plot file example.plt:

bgraph example.plt | output imagew

FILES

see `pexpand(1)` and `psort(1)`

AUTHOR

Greg Ward

SEE ALSO

`bgraph(1)`, `cv(1)`, `igraph(1)`, `impress(1)`, `mx80(1)`, `output(1)`, `pexpand(1)`, `psort(1)`

NAME

impress - convert metafile to imPress language for imagen

SYNOPSIS

impress [-c | -r] file ..

DESCRIPTION

Impress reads each metafile *file* in sequence and converts it to output suitable for the imagen line of printers. If the option *c* is specified, the input files are only conditioned for output, ie. expanded (see pexpand). This is useful if many copies of the same output is desired. If the option *r* is instead specified, the input is assumed already to be conditioned. If no input files are specified, the standard input is read.

-c Condition the input only.

-r Input is already conditioned, output only.

EXAMPLE

To print the plot file example.plt to the Impress printer ip4:

```
bgraph example.plt | impress | lpr -P ip4
```

FILES

see `pexpand(1)`

AUTHORS

William LeFebvre and Greg Ward

SEE ALSO

`bgraph(1)`, `igraph(1)`, `imagew(1)`, `mx80(1)`, `t4014(1)`

NAME

lam - laminate lines of multiple files

SYNOPSIS

lam [-tC] file1 file2 ..

DESCRIPTION

Lam simply joins lines from multiple files, separating them with the given tab character (TAB by default). If the files do not have the same number of lines, then shorter files will stop contributing to the output as they run out.

A hyphen ('-') by itself can be used to indicate the standard input.

EXAMPLE

To join files output1 and output2, separated by a comma:

```
lam -t, output1 output2
```

AUTHOR

Greg Ward

SEE ALSO

`cnt(1)`, `neat(1)`, `rcalc(1)`, `tabfunc(1)`, `total(1)`

NAME

lampcolor - compute spectral radiance for diffuse emitter

SYNOPSIS

lampcolor [lamptable]

DESCRIPTION

Lampcolor is an interactive program for computing the appropriate color for a diffuse emitter. From the total lumen output and a simple description of the fixture geometry, the radiance for a diffuse fixture is calculated. Since most light fixtures are not really diffuse, the preferred method is to use luminaire distribution data (see *ies2rad*(*1*)).

A table of lamp colors and depreciation factors is used to get the color for a specific lamp type. If "white" is specified, the lamp will be uncolored. This is probably preferable for scenes with only a single variety of lamp in order to produce color-balanced images.

ENVIRONMENT

`RAYPATH`	Directories to check for lamp table

FILES

`lamp.tab`	Default lamp lookup table

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

`ies2rad(1)`

NAME

lookamb - examine ambient file values

SYNOPSIS

lookamb [-d][-r][-h] [ambfile]

DESCRIPTION

Lookamb examines an ambient file produced by *rpict*(*1*), *rtrace*(*1*), or *rview*(*1*). The default mode prints the position, direction, level, weight, radius, value, position gradient and direction gradient for each stored ambient value preceded by a label appropriate for human interpretation. The *-d* option prints the same quantities in 18 columns without any expanatory text, and is more appropriate as input to another program.

The -r option performs the reverse conversion, taking the output of lookamb and reproducing the original ambient file on the standard output.

The -h option causes lookamb not to print out the header information on the output, or not to expect it on the input with the *-r* option.

If no file is given, lookamb reads from the standard input.

NOTES

Before release 2.2 of *Radiance*, ambient files were in a machine-dependent format. Since that is no longer the case, ambient files can now be moved freely between machines without any conversions. Thus, the only reason to use *lookamb* now is to examine the contents of an ambient file.

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `oconv(1)`, `pfilt(1)`, `pinterp(1)`, `rpict(1)`, `rtrace(1)`, `rview(1)`

NAME

macbethcal - compute color compensation based on measured Macbeth chart

SYNOPSIS

macbethcal [-d debug.pic][-p xul yul xur yur xll yll xlr ylr] scannedin.pic [calibout.cal]
macbethcal -c [-d debug.pic] [measured.xyY [calibout.cal]]

DESCRIPTION

Macbethcal takes a scanned image or measurement set of a Macbeth ColorCheckerMcolor rendition chart and computes a color mapping function suitable as input to *pcomb*(*1*).

In the first form, macbethcal takes a scanned image of a Macbeth chart that has been converted into a *Radiance* picture using a fixed procedure. When used properly as input to *pcomb*, the computed calibration file will adjust the brightness and color of any similarly scanned and converted image so as to best match the original. If the lighting conditions are carefully controlled (as in the case of a flatbed scanner), it is even possible to get reliable reflectance values this way, at least within 10% or so. The input picture is named on the command line. The output calibration file will be written to the standard output if no file name is given on the command line.

In the second form, the input is from a file containing measured values for each Macbeth color. This file must contain entries of the form:

```
N   x   y   Y
```

Where N is the number of the corresponding Macbeth color. (See back of ColorChecker chart for color names and indexing, but it basically starts from the upper left with 1 and proceeds in English text order to the lower right, which is 24.) The values *x*, *y* and *Y* are the 1931 CIE (x,y) chromaticity coordinates followed by the luminance for that color, which can be in any units. If a white value is known (i.e. maximum output level), then it may be given as entry number 0. The entries may be in any order, and comments may be included delimited by a pound sign ('#') and continuing to the end of line. It is recommended that measurements be done for all 24 colors, but the only required entries are the 6 neutral values on the bottom row of the chart.

Computing a mapping from measured colors is usually more convenient when calibrating a particular output device. This is accomplished by printing the picture *macbeth_spec.pic* (which may be found in the standard

Radiance library directory in the lib subdirectory) and measuring the output with a chroma meter or spectrophotometer.

For a scanned image, the locations of the 24 Macbeth patches in the input picture must be known. If the chart borders are not at the edges of the input picture, or the chart has been reversed or rotated or is uncentered or at an oblique angle, then it is necessary to specify the pixel locations of the corners of the chart with the *-p* option. The corner postions (x,y pixel addresses as given by the *ximage*(*1*) ``p'' command) are ordered on the command line: upper-left, upper-right, lower-left, lower-right (i.e. English text ordering). These coordinates should be the outside corner positions of the following patches:

upper-left	= 1. dark skin
upper-right	= 6. bluish green
lower-left	= 19. white
lower-right	= 24. black

If the chart has been flipped or rotated, simply give the pixel positions of the appropriate patch corners, wherever they are in the image. (Note: if the *Radiance* picture has been flipped or rotated with *pflip*(*1*) or *protate*(*1*), *ximage* will report the original pixel positions if the *-c* option was not used by the reorienting program(s). This will be wrong, so be sure to use the *-c* option.) *Macbethcal* can handle a chart with any orientation or perspective warping if the corner coordinates are given correctly. The debug picture output is the best way to check for consistency. (See the *-d* option, below.)

The -d option may be used to specify an additional output file, which will be a picture comparing the scanned image processed according to the computed mapping against the standard Macbeth colors. It is a good idea to use the debug option to check that the color patches are being located correctly, and to see how well *macbethcal* does at matching colors. The center of each patch will show the target color; the left side of each patch will show the original color, and the right side will show the corrected value. If the match works well, the debug picture should have a sort of ``notch on the left'' look in each patch. Macbeth colors that could not be matched because they were out of gamut on this device are indicated with diagonal lines drawn through the associated target colors.

METHOD

Macbethcal computes the color mapping in two stages. The first stage uses the six neutral color patches at the bottom of the Macbeth chart to compute a piecewise linear approximation to the brightness mapping of each RGB primary. The second stage looks at all the colors that are within the device's gamut to compute a least-sqaures fit for a linear color transformation from the measured space into the standard *Radiance* RGB space (as defined by the three primaries in src/common/color.h).

Thanks to the nature of inverse mappings, this method should work either for converting scanned data to match the original, or for preconditioning pictures to be sent to specific output devices. In other words, the same calibration file works either for correcting scanned images OR precorrecting images before printing.

A warning is printed if some unsaturated colors are determined to be out of gamut, as this may indicate a poor rendition or improper picture alignment. The debug picture will show which colors were excluded by drawing diagonal lines through their entries.

NOTE

It is very important that the same settings be applied when scanning or printing other images to be calibrated with the computed file. In particular, all exposure adjustments should be fixed manually, and no tweaking of the settings should be done along the way. The final result will be best if the original scanned image is not too far off from what it should be. In the case of slide and negative scanners, it is best to apply the recommended calibration file for the type of film used, so long as this calibration is fixed and not adjusted on a per-image basis.

CHART AVAILABILITY

The Macbeth chart is available at most photographic supply stores, or may be ordered directly from Macbeth:

```
Macbeth
 Munsell Color
 405 Little Britain Rd.
 New Windsor, NY 12553-6148
 tel. 1-800-622-2384 (USA)
 fax. 1-914-561-0267
```

The chart sells for under $50 US at the time of this writing.

EXAMPLES

To compute a calibration for a FunkyThing scanner and check the results:

```
ra_tiff -r mbscan.tif mbscan.pic
macbethcal -d debug.pic mbscan.pic FunkyThing.cal
ximage debug.pic
```

To apply this computed calibration to another scanned image:

```
ra_tiff -r another.tif | pcomb -f FunkyThing.cal -
> another_calib.pic
```

To compute a calibration file for the BigWhiz film recorder, after taking measurements of a slide made from macbeth_spec.pic:

```
macbethcal -c macbeth_spec.xyY BigWhiz.cal
```

To prepare a picture prior to output on the same film recorder:

```
pcomb -f BigWhiz.cal standard.pic > toprint.pic
```

To use pcond(1) to also adjust the image for human response:

```
pcond -f BigWhiz.cal -h standard.pic > toprint.pic
```

AUTHOR

Greg Ward
Paul Heckbert supplied code for perspective projective mapping

SEE ALSO

`calc(1)`, `pcomb(1)`, `pcond(1)`, `pfilt(1)`, `ximage(1)`

NAME

libmeta.a - simplified interface to metafile(5)

SYNOPSIS

```
extern FILE *pout;
mline(x, y, type, thick, color)
mpoly(x, y, border, pat, color)
mdraw(x, y)
mtext(x, y, s, cpi, color)
        char *s;
mvstr(xmin, ymin, xmax, ymax, s, d, thick, color) char *s;
mrectangle(xmin, ymin, xmax, ymax, pat, color)
mtriangle(xmin, ymin, xmax, ymax, d, pat, color)
msegment(xmin, ymin, xmax, ymax, sname, d, thick, color) char
        *sname;
msetpat(pat, pattern)
        char *pattern;
mopenseg(sname)
        char *sname;
mcloseseg()
minclude(fname)
        char *fname;
mendpage()
mdone()
```

DESCRIPTION

The routines in *libmeta* provide a simple interface to the `metafile(5)` 2D graphics stream. Output from these routines is sent to *pout*. *Pout* defaults to the standard output, and should be piped to the appropriate device driver.

All coordinates range from 0 to 16383 and map to a square area on the output device. *D* values are one of 'r', 'u', 'l' and 'd' corresponding to right, up, left, and down respectively. *Color* values range from 0 to 3 and normally correspond to black, cyan, green and blue. *Pattern* values range from 0 to 3 and default to solid, thick lines, thin lines, and candystripe. Pattern value mapping may be changed via *setpat*. All strings are null-terminated, and do not contain newlines.

Mline starts a line at the given coordinates. The line type is a number from 0 to 3 corresponding to solid, dashed, dotted, and dot-dashed. The line thickness, *thick*, is a number from 0 to 3. Connected lines are drawn with successive calls to *mdraw*.

Mpoly starts a polygon at the given coordinates. The boolean *border* specifies whether or not a border is desired around the polygon. *Mdraw* is used to add vertices to the polygon. The polygon will be closed automatically after the last call.

Mtext prints a string of hardware characters starting at the given coordinates. The characters per inch are *cpi*. Text is always oriented to the right.

Mvstr places a vector character string within the given boundaries. The string is oriented according to *d*. The character line thickness is given by *thick.*

Mrectangle fills the given box with pat. Mtriangle fills the half-box with orientation *d* in the given boundaries. Right corresponds to a triangle in the lower right half of the box. Up corresponds to a triangle in the upper right, left is upper left, and down is lower left.

Msegment places an instance of the segment sname within the given boundaries. The segment is oriented according to *d*, where 'r' is null rotation. If either *thick* or *color* is nonzero, its value will replace corresponding values in the segment primitives. (For area filling, thick changes the fill pattern.)

Msetpat maps pat to pattern. Pattern is a string of the form "Pn" where n is a number between 0 and 11.

Mopenseg opens the segment named sname. All graphics calls up to a matching call to *mcloseseg* are stored under *sname*. An instance of the segment is obtained with a call to *mseg*_m*ent*. Segments can be nested to any level, and redefining segments is allowed. Beware of calls to *mtext* within a segment, since text will not rotate or scale.

Minclude includes the graphics metafile fname in the output stream. *Mendpage* advances to the next screen or page. On a terminal, the bell rings and a line is read to prevent premature erasure. *Mdone* completes metafile output, and is the only required call.

DIAGNOSTICS

None.

SEE ALSO

`t4014(1)`, `mx80(1)`, `impress(1)`, `primout(3)`, `metafile(5)`

NAME

meta2tga - convert metafile to Targa image format

SYNOPSIS

meta2tga [-c | -r][-x width][-y height][-m minrad][-o outname] file ..

DESCRIPTION

Meta2tga reads each metafile *file* in sequence and converts it to a compressed, color-mapped Targa file. The result is sent to the standard output (which must be redirected) unless the *-o* option is used. The argument to the *-o* option specifies the base file name, to which a page number and ".tga" is added as a suffix. Note that this option must be present in order to produce more than a single page of output.

The default output resolution is 400 by 400, but a different resolution can be given with the *-x* and *-y* options.

The -m option can be used to set a minimum value for the line radius in pixels. This may be helpful for improving the readability of high resolution output. The default value is 0, which allows lines of one pixel thickness.

If the option -c is specified, the input files are only conditioned for output, ie. expanded (see pexpand). This is useful if many copies of the same output is desired. If the option *-r* is instead specified, the input is assumed already to be conditioned. If no input files are specified, the standard input is read.

EXAMPLE

To convert the plots examp1.plt and examp2.plt to 1024x1024 Targa files:

```
bgraph examp1.plt examp2.plt | meta2tga -o examp
-x 1024 -y 1024
```

FILES

see `pexpand(1)` and `psort(1)`

AUTHOR

Greg Ward

SEE ALSO

bgraph(1), igraph(1), imagew(1), mx80(1), pexpand(1), psort(1), ra_t8(1), t4014(1), tbar(1)

NAME

metafile - graphics command interface, similar to plot(5)

DESCRIPTION

The *metafile* graphics format was designed with the primary goal of serving as a temporary file for routines which output to dot-matrix and other line-at-a-time devices. As a result, all of the "primitives" are completely selfcontained to facilitate sorting.

A primitive is a command which can itself be plotted. Into this catagory fall line segments, rectangle and triangle fills, matrix and vector strings. Every primitive has a zeroeth argument which contains bundled attribute information, and an extent. The extent gives the x and y minimum and maximum values which enclose the primitive. The extent is used in sorting, and typically also in describing the primitive. For example, a line segment will be described completely by its enclosing rectangle and attributes including specification of which diagonal the segment falls on. Other primitives will have additional arguments, such as vector string, which must specify the string to be output within its extent.

"Global" commands separate the primitives and allow functions which affect all commands. These are commands such as end of page, pause, open and close segment, set, unset and reset, and a special global, end of file. The end of file command is included to facilitate finding the end of file on systems which do not keep track exactly. Global commands sometimes have arguments. The open command, for instance, specifies the name of the segment. Global commands never have extents.

The metafile commands are as follows:

F end of file: no arguments.
When end of file is reached, all processing stops.

E end of page: no arguments.
This causes the device to advance to the next screen or page. If the output device is a terminal, it will beep and wait for the user to hit return before clearing the screen.

P pause: arguments specify the message to be printed. This causes output to be flushed and the controlling terminal to be opened. The user is then prompted with the specified string followed by the message "- (hit return to continue)". If no string is specified, the bell is sounded without a message. After the user hits return, output continues. This command is useful when the user is required for some part of the output, such as changing paper or pens.

D draw global: no arguments.

This global forces flushing of output and updating of device.

I include file: arg0 TRUE if standard file. The include global causes the contents of the named file to be substituted in the include command's location. If arg0 is 1 (TRUE), a standard location is searched if the file is not found in the working directory. If arg0 is 0 (FALSE), the file must be in the working directory. Include files can be nested to the number of allowed open files.

S set: arg0 specifies what to set (from meta.h): SALL: place context mark on current settings. SPAT0: set pattern 0 to the specified value. SPAT1: set pattern 1 to the specified value. SPAT2: set pattern 2 to the specified value. SPAT3: set pattern 3 to the specified value. The set command is used to globally affect certain attributes. The zeroeth argument specifies the variable to set, and the arguments following specify the value. Pattern values can have two forms. The first form begins with the letter 'P', immediately followed by an integer between 0 and 11. This selects one from the following patterns: solid, thick \ thin \ mixed \ thick ///, thin ///, mixed ///, crisscross, web. The default pattern settings are: 0=P0, 1=P1, 2=P2, 3=P3. The second form gives the explicit values for a pattern. The set all command makes a context mark with the current settings. All settings which follow can be undone with the unset all command.

U unset: arg0 specifies what to unset (from meta.h): SALL: return to previous context.

SPAT0: set pattern 0 to the previous value. SPAT1: set pattern 1 to the previous value. SPAT2: set pattern 2 to the previous value. SPAT3: set pattern 3 to the previous value. The unset command returns a variable to its previous value. The unset all command returns the settings to the values they had in the previous context. If no context has been marked by set all, variables are returned to their default values.

R reset: arg0 specifies what to reset (from meta.h): SALL: reset all variables.

SPAT0: set pattern 0 to the default value. SPAT1: set pattern 1 to the default value.

SPAT2: set pattern 2 to the default value. SPAT3: set pattern 3 to the default value. The reset command returns a variable to its default setting. The reset all command returns all variables to their initial state.

O open segment: arguments specify segment name. The commands following up to a C (close segment) are not to be output, but are to be stored in the named segment. Segment names can contain any ascii character (except newline) in any sequence of reasonable length. Segment definitions are local to the enclosing segment. Side effects should be avoided in segments by balancing calls to set and unset. A segment cannot reference itself.

C close segment: no arguments.
The current segment is closed, which completes its usable definition.

l line segment: fields of arg0 are:
100: orientation: positive slope, negative slope. 060: type: solid, dashed, dotted, dotted-dashed. 014: width: 0, 12, 24, 48, 96 units. 003: color: black, red, green, blue.

r rectangle fill: fields of arg0 are: 100: toggle: OR fill, XOR fill.
014: pattern: choice of 4 (see set). 003: color: black, red, green, blue. Fills the given extent with the specified pattern. Toggle (XOR) fill allows the reversal of previous fills to an area.

t triangle fill: fields of arg0 are:
100: toggle: OR fill, XOR fill.
060: orientation: right (& down), up, left, down. 014: pattern: choice of 4 (see set). 003: color: black, red, green, blue. Fills the given half-rectangle with the specified pattern. A triangle is oriented to the right if the the area between the positive-sloped diagonal and the lower right corner of the extent is filled. Rotating this triangle ccw successively yields up, left and down triangles. Toggle (XOR) fill allows the reversal of previous fills to an area.

p polygon fill: fields of arg0 are:
100: border: no border, line border. 060: orientation: right (& down), up, left, down. 014: pattern: choice of 4 (see set). 003: color: black, red, green, blue. The argument string gives a blank separated list of the polygon vertices in the form: "x0 y0 x1 y1 x2 y2 ... ". The coordinates must be integers ranging between 0 and 16383. The last vertex will be connected to the first, and the polygon will be filled in with the specified pattern. If a border is requested, one will be drawn of solid black zero width lines. All polygon fills will toggle, therefore other polygon and toggled triangle and rectangle fills will affect the final appearance of the image. For example, a polygon drawn inside another polygon of the same pattern will make a hole.

m matrix string: fields of arg0 are:

100: strike: single, double.

060: density: 10 cpi, 12 cpi, 17 cpi, 20 cpi. 014: size: normal, double width, double height, double both. 003: color: black, red, green, blue. The upper left corner of the extent is used to place the beginning of the string specified after the command. More sophisticated drivers will use the extent for clipping, but the size of the characters will not be altered.

v vector string: fields of arg0 are:

060: orientation: right, up, left, down. 014: thickness: 0, 12, 24, 48, 96 units. 003: color: black, red, green, blue. The string specified following the command will be made to fit within the given extent.

s print segment: fields of arg0 are:

060: orientation: right, up, left, down. 014: thickness: 0, 12, 24, 48, 96 units. 003: color: black, red, green, blue. The segment whose name is specified in the arguments will be oriented according to arg0 and made to fit in the given extent. The thickness and color of the lines in the segment will be changed also according to arg0. In the case of area fill, it is the pattern rather than the width which will change. The segment must have been previously defined using the open segment global. Note that matrix strings will not transfer well since they cannot be oriented or scaled.

The metafile has two basic formats. The first format is meant to be user readable, and has the form:

c arg0 xmin ymin xmax ymax 'args

Where c is the single letter command, arg0 is the octal value for arg0, xmin ymin xmax ymax are the extent (ranging from 0 to 16283), and the optional args following the backquote are additional arguments, terminated by a newline. If the command is a global, the extent is not present. If the global has no arg0, 0200 is appropriate. Any global which has a following string must have a value for arg0 (< 0200). Comments are permitted on lines beginning with a pound sign ('#').

The second format is roughly equivalent, but packs the extrema into two bytes each. It takes between one quarter and one third as much space, and much less processing to use this type of file, hence it is the default format for all of the programs. Conversion between formats is accomplished with `cv(1)`.

FILES

The standard location for metafiles used by the programs is /usr/lib/meta/, but can be changed by setting the environment variable MDIR. This is useful for systems where the owner does not have access to the /usr/lib/ directory. It also allows the user to create his own metafiles for vector characters and other symbols.

BUGS

The command for line segment ('l') is awkward at best.

AUTHOR

Greg Ward

SEE ALSO

`cv(1)`, `meta(3)`, `pexpand(1)`, `primout(3)`, `psort(1)`

NAME

mgf2meta - convert Materials and Geometry Format file to Metafile graphics

SYNOPSIS

mgf2meta [-t threshold] {x|y|z} xmin xmax ymin ymax zmin zmax [input ..]

DESCRIPTION

Mgf2meta converts one or more Materials and Geometry Format (MGF) files to a 2-D orthographic projection along the selected axis in the *metafile*(*1*) graphics format. All geometry is clipped to the specified bounding box, and the resulting orientation is as follows:

Projection	Orientation
x	Y-axis right, Z-axis up
y	Z-axis right, X-axis up
z	X-axis right, Y-axis up

If multiple input files are given, the first file prints in black, the second prints in red, the third in green and the fourth in blue. If more than four input files are given, they cycle through the colors again in three other line types: dashed, dotted and dot-dashed.

The -t option may be used to randomly throw out line segments that are shorter than the given *threshold* (given as a fraction of the plot width). Segments are included with a probability equal to the square of the line length over the square of the threshold. This can greatly reduce the number of lines in the drawing (and therefore improve the drawing speed) with only a modest loss in quality. A typical value for this parameter is 0.005.

All MGF material information is ignored on the input.

EXAMPLE

To project two MGF files along the Z-axis and display them under X11:

```
mgf2meta z 0 10 0 15 0 9 building1.mgf
building2.mgf x11meta -r
```

To convert a *Radiance* scene to a line drawing in *Radiance* picture format:

```
     rad2mgf scene.rad | mgf2meta x `getbbox -h
     scene.rad` meta2tga | ra_t8 -r > scene.pic
```

AUTHOR

Greg Ward

SEE ALSO

getbbox(1), meta2tga(1), metafile(5), mgf2rad(1), pflip(1), protate(1), psmeta(1), ra_t8(1), rad2mgf(1), t4014(1), x11meta(1)

MGF web site "http://radsite.lbl.gov/mgf/HOME.html"

NAME

mgf2rad - convert Materials and Geometry Format file to *Radiance* description

SYNOPSIS

mgf2rad [-m matfile][-e mult][-g dist] [input ..]

DESCRIPTION

Mgf2rad converts one or more Materials and Geometry Format (MGF) files to a *Radiance* scene description. By definition, all output dimensions are in meters. The material names and properties for the surfaces will be those assigned in MGF. Any materials not defined in MGF will result in an error during translation. Light sources are described inline as IES luminaire files, and *mgf2rad* calls the program *ies2rad*(*1*) to translate these files. If an IES file in turn contains an MGF description of the local fixture geometry, this may result in a recursive call to *mgf2rad*, which is normal and should be transparent. The only side-effect of this additional translation is the appearance of other *Radiance* scene and data files produced automatically by *ies2rad*.

The -m option may be used to put all the translated materials into a separate *Radiance* file. This is not always advisable, as any given material name may be reused at different points in the MGF description, and writing them to a separate file loses the contextual association between materials and surfaces. As long as unique material names are used throughout the MGF description and material properties are not redefined, there will be no problem. Note that this is the only way to get all the translated materials into a single file, since no output is produced for unreferenced materials; i.e. translating just the MGF materials does not work.

The -e option may be used to multiply all the emission values by the given *mult* factor. The *-g* option may be used to establish a glow distance (in meters) for all emitting surfaces. These two options are employed principally by *ies2rad*, and are not generally useful to most users.

EXAMPLE

To translate two MGF files into one *Radiance* materials file and one geometry file:

```
mgf2rad -m materials.rad building1.mgf
building2.mgf > building1+2.rad
```

To create an octree directly from two MGF files and one *Radiance* file:

```
oconv '!mgf2rad materials.mgf scene.mgf'
source.rad > scene.oct
```

FILES

`tmesh.cal`	Used to smooth polygonal geometry
`*.rad`	*Radiance* source descriptions created by ies2rad
`*.dat`	*Radiance* source data created by ies2rad
`source.cal`	Used for IES source coordinates

AUTHOR

Greg Ward

SEE ALSO

`ies2rad(1)`, `mgf2meta(1)`, `obj2rad(1)`, `oconv(1)`, `rad2mgf(1)`, `xform(1)`

MGF web site "http://radsite.lbl.gov/mgf/HOME.html"

NAME

mkillum - compute illum sources for a *Radiance* scene

SYNOPSIS

mkillum [rtrace options] octree [< file ..] mkillum [rtrace options] -defaults

DESCRIPTION

Mkillum takes a prepared *Radiance* scene description and an octree and computes light source distributions for each surface, replacing them with secondary sources whose contributions can be computed more efficiently by *rpict*(*1*) and *rview*(*1*). This type of optimization is most useful for windows and skylights which represent concentrated sources of indirect illumination. *Mkillum* is not appropriate for very large sources or sources with highly directional distributions. These are best handled respectively by the ambient calculation and the secondary source types in *Radiance*.

The arguments to mkillum are passed directly to rtrace(1), which is used to compute the light distributions for the input surfaces. These surfaces can be any combination of polygons, spheres and rings. Other surfaces may be included, but *mkillum* cannot compute their distributions.

By default, mkillum reads from its standard input and writes to its standard output. It is possible to specify multiple input files in a somewhat unconventional fashion by placing a lesser-than symbol ('<') before the file names. (Note that this character must be escaped from most shells.) This is necessary so *mkillum* can tell where the arguments to *rtrace*(*1*) end and its own input files begin.

VARIABLES

Mkillum has a number of parameters that can be changed by comments in the input file of the form:

```
     #@mkillum variable=value option switch{+|-} ..
```

String or integer variables are separated from their values by the equals sign ('='). Options appear by themselves. Switches are followed either by a plus sign to turn them on or a minus sign to turn them off.

Parameters are usually changed many times within the same input file to tailor the calculation, specify different labels and so on. The parameters and their meanings are described below.

o=string

Set the output file to string. All subsequent scene data will be sent to this file. If this appears in the first comment in the input, nothing will be sent to the standard output. Note that this is not recommended when running *mkillum* from *rad*(*1*), which expects the output to be on the standard output.

m=string

Set the material identifier to string. This name will be used not only as the new surface modifier, but it will also be used to name the distribution pattern and the data files. The distribution name will be *string* plus the suffix ".dist". The data file will be named *string* plus possibly an integer plus a ".dat" suffix. The integer is used to avoid accidently writing over an existing file. If overwriting the file is desired, use the *f* variable below.

f=string

Set the data file name to string. The next data file will be given this name plus a ".dat" suffix. Subsequent files will be named *string* plus an integer plus the ".dat" suffix. An existing file with the same name will be clobbered. This variable may be unset by leaving off the value. (See also the *m* variable above.)

a Produce secondary sources for all of the surfaces in the input. This is the default.

e=*string*

Produce secondary sources for all surfaces except those modified by *string*. Surfaces modified by *string* will be passed to the output unchanged.

i=string

Only produce secondary sources for surfaces modified by *string*.

n Do not produce any secondary sources. All input will be passed to the output unaffected.

b=*real*

Do not produce a secondary source for a surface if its average brightness (radiance) is less than the value *real*.

c={*d*|*a*|*n*}

> Use color information according to the given character. If the character is *d*, then color information will be used in three separate data files and the distribution will be fully characterized in terms of color. If the character is *a*, then only the average color is computed and the distribution will not contain color information. If the character is *n*, even the average distribution color will be thrown away, producing secondary sources that are completely uncolored. This may be desirable from a color-balancing point of view.

d=integer

> Set the number of direction samples per projected steradian to *integer*. The number of directions stored in the associated data file will be approximately this number multiplied by pi for polygons and rings, and by 4pi for spheres. If *integer* is zero, then a diffuse source is assumed and no distribution is created.

s=integer

> Set the number of ray samples per direction to *integer*. This variable affects the accuracy of the distribution value for each direction as well as the computation time for *mkillum*.

l{+|-}

> Switch between light sources and illum sources. If this switch is enabled (*l*+), *mkillum* will use the material type "light" to represent surfaces. If disabled (*l*-), *mkillum* will use the material type "illum" with the input surface modifier as its alternate material. The default is *l*-.

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

`oconv(1)`, `rad(1)`, `rpict(1)`, `rtrace(1)`, `rview(1)`

NAME

mx80 - output metafile to Epson mx-80

SYNOPSIS

mx80 [-c | -r] file ..

DESCRIPTION

Mx80 reads each metafile *file* in sequence and converts it to output suitable for the Epson line of printers, specifically the mx-80 and fx-80. If the option *c* is specified, the input files are only conditioned for output, ie. expanded and sorted (see pexpand and psort). This is useful if many copies of the same output is desired. If the option *r* is instead specified, the input is assumed already to be conditioned. If no input files are specified, the standard input is read.

-c	Condition the input only.
-r	Input is already conditioned, output only.

EXAMPLE

To print the plot file test.plt:

```
bgraph test.plt | output mx80
```

FILES

see `pexpand(1)` and `psort(1)`

AUTHOR

Greg Ward

BUGS

Currently, different character widths and densities are not supported.

SEE ALSO

`bgraph(1)`, `cv(1)`, `igraph(1)`, `impress(1)`, `output(1)`, `pexpand(1)`, `psort(1)`

NAME

neat - neaten up output columns

SYNOPSIS

neat [format]

DESCRIPTION

Neat reads from its standard input and neatens up columns separated by white space using the specified format. The format is a string consisting of a positive integer followed by an alignment character and another integer. The alignment character is usually a decimal point ('.'), but it can be any non-digit.

The alignment character is used as the central point of each column. The total column field width will be the number to the left of the alignment character plus one for the alignment character itself plus the number to the right of the alignment character.

If a field does not contain the alignment character, it will be printed to the left of where the alignment character would have appeared. If a field is too long to print within the specified format, the entire field will be printed and that row will not be aligned with the rest.

The default format is "8.8".

EXAMPLE

To examine a file with columns of numbers:

```
neat 10.8 < input | more
```

BUGS

Columns wider than the total width of the format specification will be printed without any separating white space.

The program does not do anything special with tabs on the input.

AUTHOR

Greg Ward

SEE ALSO

`cnt(1)`, `lam(1)`, `rcalc(1)`, `total(1)`

NAME

normpat - normalize *Radiance* pictures for use as patterns.

SYNOPSIS

normpat [-v][-b][-f][-r maxres] picture ..

DESCRIPTION

Normpat normalizes one or more *Radiance* pictures to an average brightness of 1.0 and optionally removes fundamental frequencies and blends the edges of the image. The original images are overwritten during this process, and it is recommended that the program work on copies of the pictures for this reason.

The -r option can be used to set the maximum horizontal or vertical resolution of the final result, which should not be greater than 256 for most patterns (due to the associated memory burden during rendering). The *-f* option uses a Fourier transform to remove the lowest frequencies from the image, reducing the noticeability of pattern repitition. The *-b* option can be used to blend the edges of the image so that when it is tiled, the seams are less apparent. The *-v* option turns on the verbose flag, which prints on the standard output progress messages as the script runs.

Normpat is a shell script that makes calls to other *Radiance* programs that do the actual work.

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `pcomb(1)`, `pcompos(1)`, `pfilt(1)`, `pflip(1)`, `protate(1)`, `psign(1)`, `ra_bn(1)`, `ra_pr(1)`, `ra_t8(1)`, `ra_t16(1)`, `rpict(1)`

NAME

obj2rad - convert Wavefront .obj file to *Radiance* description

SYNOPSIS

obj2rad [-n][-f][-m mapfile][-o objname] [input]

DESCRIPTION

Obj2rad converts a Wavefront .obj file to a *Radiance* scene description. The material names for the surfaces will assigned based on the mapping rules file given in the *-m* option. If no mapping file is given, the identifiers given by the "usemtl" statements will be used as the material names. If no "usemtl" statements are found, the group names (given by the "g" statement) will be used instead. Failing this, the default material "white" will be used.

A mapping file contains a list of materials followed by the conditions a surface must satisfy in order to have that material. For example, if we wanted all faces in the Group "thingy" with texture Map "pine" to use the material "wood", and all other surfaces to use the material "default", we would create the following mapping file:

```
    default ;
    wood (Group "thingy") (Map "pine") ;
```

All faces would satisfy the first set of conditions (which is empty), but only the faces in the Group "thingy" with texture Map "pine" would satisfy the second set of conditions.

Each rule can have up to one condition per qualifier, and different translators use different qualifiers. In *obj2rad*, the valid qualifiers are *Material*, *Map*, *Group*, *Object* and *Face*. A condition is either a single value for a specific attribute, or an integer range of values. (Integer ranges are specified in brackets and separated by a colon, eg. [-15:27], and are always inclusive.) A semicolon is used to indicate the end of a rule, which can extend over several lines if necessary.

The semantics of the rule are such that "and" is the implied conjunction between conditions. Thus, it makes no sense to have more than one condition in a rule for a given qualifier. If the user wants the same material to be used for surfaces that satisfy different conditions, they simply add more rules. For example, if the user also wanted faces between 50 and 175 in the Group "yohey" to use "wood", they would add the following rule to the end of the example above:

```
    wood (Face [50:175]) (Group "yohey") ;
```

Note that the order of conditions in a rule is irrelevant. However, the order of rules is very important, since the last rule satisfied determines which material a surface is assigned.

By convention, the identifier "void" is used to delete unwanted surfaces. A surface is also deleted if it fails to match any rule. Void is used in a rule as any other material, but it has the effect of excluding all matching surfaces from the translator output. For example, the following mapping would delete all surfaces in the Object "junk" except those with the Group name "beige", to which it would assign the material "beige_cloth", and all other surfaces would be "tacky":

```
tacky ;
void (Object "junk") ;
beige_cloth (Object "junk") (Group "beige") ;
```

The -n option may be used to produce a list of qualifiers from which to construct a mapping for the given .obj file. This is also useful for determining which materials must be defined when no mapping is used.

The -f option is used to flatten all faces, effectively ignoring vertex normal information. This is sometimes desirable when a smaller model or more robust rendering is desired, since interpolating vertex normals takes time and is not always reliable.

The -o option may be used to specify the name of this object, though it will be overriden by any "o" statements in the input file. If this option is absent, and there are no "o" statements, *obj2rad* will attempt to name surfaces based on their group associations.

If no input files are given, the standard input is read.

DETAILS

The following Wavefront statements are understood and translated by *obj2rad*.

# A comment. This statement is passed to the output verbatim. It has no effect.

f A polygonal face. If the vertices have associated surface normals, the face will be broken into quadrilaterals and triangles with the appropriate *Radiance* textures to interpolate them. Likewise, if the face is non-planar, it will be broken into triangles. Each face in the input file is assigned a number, starting with 1, and this number may be used in the material mapping rules.

g Group association. The following faces are associated with the named group(s). These may be used in the mapping rules, where a rule is matched if there is an association with the named Group. (I.e. since there may be multiple group associations, any match is considered valid.) If a mapping file is not used and no "usemtl" statement has been encountered, the main group is used for the surface material identifier.

o Object name. This is used to name the following faces, and may be used in the mapping rules.

usemap

A texture map (i.e. *Radiance* pattern) name. The name may be used in the material mapping rules, but the indexing of *Radiance* patterns is not yet supported.

usemtl

A material name. The name may be used in mapping rules, or will be used as the *Radiance* material identifier if no mapping is given.

v A vertex, given by its x, y and z coordinates.

vn A vertex normal, given by its x, y and z direction components. This vector will be normalized by *obj2rad*, and an error will result if it has length zero.

vt A vertex texture coordinate. Not currently used, but will be if we ever get around to supporting Wavefront textures.

All other statement types will be ignored on the input. A final comment at the end of the *Radiance* output file will give some indication of how successful the translation was, since it will mention the number of statements *obj2rad* did not recognize.

EXAMPLE

To create a qualifier list for triceratops.obj:

```
obj2rad -n triceratops.obj > triceratops.qual
```

To translate triceratops.obj into a *Radiance* file using the mapping triceratops.map:

```
obj2rad -m triceratops.map triceratops.obj >
triceratops.rad
```

NOTES

Many good and useful Wavefront object files are available by anonymous ftp from "avalon.chinalake.navy.mil" in the /pub/objects/obj directory.

FILES

`tmesh.cal` - used for triangle normal interpolation
`surf.cal` - used for quadrilateral normal interpolation

AUTHOR

Greg Ward

SEE ALSO

`arch2rad(1)`, `ies2rad(1)`, `oconv(1)`, `thf2rad(1)`, `xform(1)`

NAME

objline - create metafile line drawings of *Radiance* object(s)

SYNOPSIS

objline [input ..]

DESCRIPTION

Objline takes one or more *Radiance* scene files and produces four parallel line projections using calls to *rad2mgf*(*1*) and *mgf2meta*(*1*). The output must be redirected to a suitable destination for *metafile*(*5*) 2-d graphics, such as *x11meta*(*1*) or *psmeta*(*1*).

The four projections presented are along the X-axis (displayed in the upper left quadrant), along the Y-axis (upper right), along the Z-axis (lower left) and an oblique view (lower right). If multiple *Radiance* input files are given, they are shown in different colors and line styles. (Materials are ignored, so materials files are best left out.) If no input files are given on the command line, the standard input is read.

EXAMPLES

To create a line drawing of the *Radiance* file "myfile.rad" and display under X11:

```
objline myfile.rad | x11meta -r &
```

To create a line drawing of three objects in different colors and send to the printer:

```
objline obj1.rad obj2.rad obj3.rad | psmeta | lpr
```

To create a line drawing of a room and convert into a 1024x1024 *Radiance* picture:

```
objline room.rad | meta2tga -x 1024 -y 1024 |
ra_t8 -r > drawing.pic
```

AUTHOR

Greg Ward

SEE ALSO

meta2tga(1), metafile(5), mgf2meta(1), psmeta(1), ra_t8(1), rad2mgf(1), x11meta(1)

NAME

objview - view *Radiance* object(s)

SYNOPSIS

objview input ..

DESCRIPTION

Objview renders a *Radiance* object interactively using *rview*(*1*). This program is merely a shell script that adds some light sources to the scene then calls *oconv*(*1*) to make an octree, followed by *rview*(*1*) with a sensible viewpoint.

Commands beginning with an exclamation point ('!') may be given as well as any number of scene files (see *oconv*(*1*) manual page for explanation), but the standard input may not be used ('-' option of *oconv*(*1*)).

AUTHOR

Greg Ward

SEE ALSO

`oconv(1)`, `rpict(1)`, `rview(1)`

NAME

oconv - create an octree from a *Radiance* scene description

SYNOPSIS

oconv [-i octree | -b xmin ymin zmin size][-n objlim][-r maxres][-f][-w][-] [input ..]

DESCRIPTION

Oconv adds each scene description *input* to *octree* and sends the result to the standard output. Each *input* can be either a file name, or a command (enclosed in quotes and preceded by a '!'). If any of the surfaces will not fit in *octree*, an error message is printed and the program aborts. If no *octree* is given, a new one is created large enough for all of the surfaces.

The -b option allows the user to give a bounding cube for the scene, starting at *xmin ymin zmin* and having a side length *size*. If the cube does not contain all of the surfaces, an error results. The *-b* and *-i* options are mutually exclusive.

The -n option specifies the maximum surface set size for each voxel. Larger numbers result in quicker octree generation, but potentially slower rendering. Smaller values may or may not produce faster renderings, since the default number (5) is close to optimal for most scenes.

The -r option specifies the maximum octree resolution. This should be greater than or equal to the ratio of the largest and smallest dimensions in the scene (ie. surface size or distance between surfaces). The default is 1024.

The -f option produces a frozen octree containing all the scene information. Normally, only a reference to the scene files is stored in the octree, and changes to those files may invalidate the result. The freeze option is useful when the octree file's integrity and loading speed is more important than its size, or when the octree is to be relocated to another directory, and is especially useful for creating library objects for the "instance" primitive type. If the input octree is frozen, the output will be also.

The -w option suppresses warnings.

A hyphen by itself ('-') tells oconv to read scene data from its standard input. This also implies the *-f* option.

The only scene file changes that do not require octree regeneration are modifications to non-surface parameters. If the coordinates of a surface are changed, or any primitives are added or deleted, oconv must be run again.

Programs will abort with a "stale octree" message if they detect any dangerous inconsistencies between the octree and the input files.

Although the octree file format is binary, it is meant to be portable between machines. The only limitation is that machines with radically different integer sizes will not work together. For the best results, the -f option should be used if an octree is to be used in different environments.

DIAGNOSTICS

There are four basic error types reported by oconv:

warning
: a non-fatal input-related error

fatal
: an unrecoverable input-related error

system
: a system-related error

internal
: a fatal error related to program limitations

consistency
: a program-caused error

Most errors are self-explanatory. However, the following internal errors should be mentioned:

Too many scene files
: Reduce the number of scene files by combining them or using calls to `xform(1)` within files to create a hierarchy.

Set overflow in addobject (id)
: This error occurs when too many surfaces are close together in a scene. Sometimes a dense scene can be accomodated by increasing the maximum resolution (by powers of two) using the *-r* option, but usually this error indicates something is wrong. Either too many surfaces are lying right on top of each other, or the bounding cube is inflated from an oversized object or an improper *-b* specification. Chances are, the surface "id" is near one of those causing the problem.

Hash table overflow in fullnode
: This error is caused by too many surfaces. If it is possible to create an octree for the scene at all, it will have to be done in stages using the *-i* option.

EXAMPLE

To add book1, book2 and a transformed book3 to the octree "scene.oct":

```
oconv -i scene.oct book1 book2 '!xform -rz 30
book3' > newscene.oct
```

AUTHOR

Greg Ward

NOTES

In the octree, the names of the scene files are stored rather than the scene information. This means that a new octree must be generated whenever the scene files are changed or moved. Also, an octree that has been moved to a new directory will not be able to find scene files with relative pathnames. The freeze option avoids these problems. *make*(*1*) or *rad*(*1*) can be used to automate octree creation and maintenance.

SEE ALSO

`getbbox(1)`, `getinfo(1)`, `make(1)`, `rad(1)`, `rpict(1)`, `rview(1)`, `rtrace(1)`, `xform(1)`

NAME

pcomb - combine *Radiance* pictures.

SYNOPSIS

pcomb [-w][-x xres][-y yres][-f file][-e expr] [[-o][-s factor][-c r g b] input ..]

DESCRIPTION

Pcomb combines equal-sized *Radiance* pictures and sends the result to the standard output. By default, the result is just a linear combination of the input pictures multiplied by *-s* and *-c* coefficients, but an arbitrary mapping can be assigned with the *-e* and *-f* options. Negative coefficients and functions are allowed, and *pcomb* will produce color values of zero where they would be negative.

The variables ro, go and bo specify the red, green and blue output values, respectively. Alternatively, the single variable *lo* can be used to specify a brightness value for black and white output. The predefined functions *ri*(*n*), *gi*(*n*) and *bi*(*n*) give the red, green and blue input values for picture *n*. To access a pixel that is nearby the current one, these functions also accept optional x and y offsets. For example, *ri*(*3,-2,1*) would return the red component of the pixel from picture 3 that is left 2 and up 1 from the current position. Although x offsets may be as large as width of the picture, y offsets are limited to a small window (+/- 8 pixels) due to efficiency considerations. However, it is not usually necessary to worry about this problem -- if the requested offset is not available, the next best pixel is returned instead.

For additional convenience, the function li(n) is defined as the input brightness for picture *n*. This function also accepts x and y offsets.

The constant nfiles gives the number of input files present, and *WE* gives the white efficacy (lumens/brightness) for pixel values. The variables *x* and *y* give the current output pixel location for use in spatially dependent functions, the constants *xmax* and *ymax* give the input resolution, and the constants *xres* and *yres* give the output resolution (usually the same, but see below). The constant functions *re*(*n*), *ge*(*n*), *be*(*n*), and *le*(*n*) give the exposure values for picture *n*, and *pa*(*n*) gives the corresponding pixel aspect ratio. Finally, for pictures with stored view parameters, the functions *Ox*(*n*), *Oy*(*n*) and *Oz*(*n*) return the ray origin in world coordinates for the current pixel in picture *n*, and *Dx*(*n*), *Dy*(*n*) and *Dz*(*n*) return the normalized ray direction. In addition, the function *T*(*n*) returns the distance from the origin to the aft clipping plane (or zero if there is no aft plane), and the function *S*(*n*) returns the solid angle of the current pixel in steradians (always zero for parallel views). If the current pixel is outside the view region, *T*(*n*) will return a negative value, and *S*(*n*) will return zero.

The -w option can be used to suppress warning messages about invalid calculations. The *-o* option indicates that original pixel values are to be used for the next picture, undoing any previous exposure changes or color correction.

The -x and -y options can be used to specify the desired output resolution, *xres* and *yres*, and can be expressions involving other constants such as *xmax* and *ymax*. The constants *xres* and *yres* may also be specified in a file or expression. The default output resolution is the same as the input resolution.

The -x and -y options must be present if there are no input files, when the definitions of *ro*, *go* and *bo* will be used to compute each output pixel. This is useful for producing simple test pictures for various purposes. (Theoretically, one could write a complete renderer using just the functional language...)

The standard input can be specified with a hyphen ('-'). A command that produces a *Radiance* picture can be given in place of a file by preceeding it with an exclamation point ('!').

EXAMPLES

To produce a picture showing the difference between pic1 and pic2:

```
pcomb -e 'ro=ri(1)-ri(2);go=gi(1)-gi(2);bo=bi(1)-
bi(2)' pic1 pic2 > diff
```

Or, more efficiently:

```
pcomb pic1 -s -1 pic2 > diff
```

To precompute the gamma correction for a picture:

```
pcomb -e 'ro=ri(1)^.4;go=gi(1)^.4;bo=bi(1)^.4' pic
> pic.gam
```

To perform some special filtering:

```
pcomb -f myfilt.cal -x xmax/2 -y ymax/2 input.pic
> filtered.pic
```

To make a picture of a dot:

```
pcomb -x 100 -y 100 -e 'ro=b;go=b;bo=b;b=if((x-
50)^2+(y50)^2-25^2,0,1)' > dot
```

AUTHOR

Greg Ward

SEE ALSO

`calc(1)`, `getinfo(1)`, `pcompos(1)`, `pfilt(1)`, `rpict(1)`

NAME

pcompos - composite *Radiance* pictures.

SYNOPSIS

pcompos [-x xres][-y yres][-b r g b][-lh h][-la] [-t min1][+t max1][-l lab][=SS] pic1 x1 y1 .. or
pcompos [-a ncols][-s spacing][-o x0 y0][options] pic1 pic2 ..

DESCRIPTION

Pcompos arranges and composites *Radiance* pictures and sends the result to the standard output. Each input picture must be accompanied by an anchor point (unless the *-a* option is used, see below). This anchor point is the usually position of the picture's left lower corner in the final output, but can be changed for individual pictures with an *=SS* option, where *S* is one of '-', '+' or '0', indicating the minimum, maximum or center of the image, respectively. (For example, =+- would indicate the anchor is relative to the right lower corner, and *=-0* would indicate the anchor is relative to the center of the left edge.) Negative anchor coordinates result in the input being cropped at the origin. By default, the size of the output picture will be just large enough to encompass all the input files. By specifying a smaller dimension using the *-x* and *-y* options, input files can be cropped at the upper boundary. Specifying a larger dimension produces a border. The *-b* option specifies a background color to appear wherever input files do not cover. The default value is black (0 0 0).

If input files overlap, later pictures will overwrite earlier ones. By default, input files are copied unconditionally within the output boundaries. The *-t* option specifies a lower threshold intensity under which input pixels will not be copied to the output. The *+t* option specifies an upper threshold. These options are useful for cutting around irregular boundaries in the input.

The -l option can be used to specify a label for a specific picture, which will be given a height determined by the *-lh* option (default 24 pixels) and placed in the upper left corner of the picture. This label is generated by the program *psign*(*1*). The *-la* option instructs *pcompos* to label each picture automatically by its name. This is particularly useful in conjunction with the *-a* option for producing a catalog of images (see example below). The *-l* option may still be used to override the default label for a picture.

The -a option can be used to automatically compute anchor points that place successive pictures next to each other in *ncols* columns. The ordering will place the first picture in the lower left corner, the next just to the right of it, and so on for *ncols* pictures. Then, the next row up repeats the pattern until all the input pictures have been added to the output. If the pictures are of different size, *pcompos* will end up leaving some background areas in the output picture. There will also be an unfinished row at the top if the number of pictures is not evenly divided by *ncols*. The *-s N* option will cause each image to be separated by at least N pixels. The *-o x0 y0* option specifies a nonzero anchor point for the bottom left image.

The standard input can be specified with a hyphen ('-'). A command that produces a *Radiance* picture can be given in place of a file by preceeding it with an exclamation point ('!').

EXAMPLE

To put a copyright label at the bottom of a picture:

```
psign Copyright 1987 | pcompos pic.inp 0 0 +t .5 -
384 64 > pic.out
```

To make a catalog of images separated by white 10-pixel borders:

```
pcompos -la -a 4 -s 10 -b 1 1 1 dog*.pic >
alldogs.pic
```

NOTES

Since there is a limit to the number of open files and processes, large collections of images must be created in stages. Even if the system limit on open files is large, *pcompos* places an artificial limit of 64 on the number of open files and/or processes.

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `pfilt(1)`, `psign(1)`, `rpict(1)`

NAME

pcond - condition a *Radiance* picture for output

SYNOPSIS

pcond [options] input [output]

DESCRIPTION

Pcond conditions a *Radiance* picture for output to a display or hard copy device. If the dynamic range of the scene exceeds that of the display (as is usually the case), *pcond* will compress the dynamic range of the picture such that both dark and bright regions are visible. In addition, certain limitations in human vision may be mimicked in order to provide an appearance similar to the experience one might have in the actual scene.

Command line switches turn flags off and on, changing program behavior. A switch given by itself toggles the flag from off to on or on to off depending on its previous state. A switch followed by a `+' turns the option on explicitly. A switch followed by a `-' turns the option off. The default is all switches off. Other options specify output device parameters in order to get more accurate color and contrast.

-h[+-]

> Mimic human visual response in the output. The goal of this process is to produce output that correlates strongly with a person's subjective impression of a scene. This switch is a bundle of the *-a*, *-v*, *-s* and *-c* options.

-a[+-]

> Defocus darker regions of the image to simulate human visual acuity loss. This option will not affect well-lit scenes.

-v[+-]

> Add veiling glare due to very bright regions in the image. This simulates internal scattering in the human eye, which results in a loss of visible contrast near bright sources.

-s[+-]

> Use the human contrast sensitivity function in determining the exposure for the image. A darker scene will have relatively lower exposure with lower contrast than a well-lit scene.

-c[+-]

If parts of the image are in the mesopic or scotopic range where the cone photoreceptors lose their efficiency, this switch will cause a corresponding loss of color visibility in the output and a shift to a scotopic (blue-dominant) response function.

-w[+-]

Use a center-weighted average for the exposure rather than the default uniform average. This may improve the exposure for scenes with high or low peripheral brightness.

-i *fixfrac*

Set the relative importance of fixation points to *fixfrac*, which is a value between 0 and 1. If *fixfrac* is zero (the default), then no fixation points are used in determining the local or global adaptation. If *fixfrac* is greater than zero, then a list of fixation points is read from the standard input. These points are given as tabseparated (x,y) picture coordinates, such as those produced by the *-op* option of *ximage*(*1*). The foveal samples about these fixation points will then be weighted together with the global averaging scheme such that the fixations receive *fixfrac* of the total weight. If *fixfrac* is one, then only the fixation points are considered for adaptation.

-l[+-]

Use a linear response function rather than the standard dynamic range compression algorithm. This will prevent the loss of usable physical values in the output picture, although some parts of the resulting image may be too dark or too bright to see.

-e *expval*

Set the exposure adjustment for the picture to *expval*. This may either be a real multiplier, or a (fractional) number of f-stops preceeded by a \`+' or \`-'. This option implies a linear response (see the *-l* option above).

-u *Ldmax*

Specifies the top of the luminance range for the target output device. That is, the luminance (in candelas/m^2) for an output pixel value of (R,G,B)=(1,1,1). The default value is 100 cd/m^2.

-d *Lddyn*

Specifies the dynamic range for the target output device, which is the ratio of the maximum and minimum usable display luminances. The default value is 32.

-p *xr yr xg yg xb yb xw yw*

Specifies the RGB primaries for the target output device. These are the 1931 CIE (x,y) chromaticity values for red, green, blue and white, respectively. Usually, the white value is set to (.333,.333) to avoid color balance problems in the display.

-f *macbeth.cal*

Use the given output file from *macbethcal*(*1*) to precorrect the color and contrast for the target output device. This does a more thorough job than a simple primary correction using the *-p* option. Only one of *-f* or *-p* may be given.

-x *mapfile*

Put out the final mapping from world luminance to display luminance to *mapfile*. This file will contain values from the minimum usable world luminance to the maximum (in candelas/m^2) in one column, and their corresponding display luminance values (also in candelas/m^2) in the second column. This file may be used for debugging purposes, or to plot the mapping function created by *pcond*.

EXAMPLES

To display an image as a person might perceive it in the actual scene:

```
pcond -h final.pic > display.pic
ximage display.pic ; rm display.pic &
```

To do the same on a 24-bit display with known primary values:

```
setenv DISPLAY_PRIMARIES ``.580 .340 .281 .570
.153 .079 .333 .333"
pcond -h -p $DISPLAY_PRIMARIES final.pic | ximage &
```

To prepare a picture to be sent to a film recorder destined eventually for a slide projector with a minimum and maximum screen luminance of 0.2 and 125 candelas/m^2, respectively:

```
pcond -b 0.2 -t 125 final.pic > film.pic
```

To do the same if the output colors of the standard image ``ray/lib/lib/macbeth_spec.pic" have been measured:

```
macbethcal -c mbfilm.xyY > film.cal
pcond -b 0.2 -t 125 -f film.cal final.pic >
film.pic
```

To further tweak the exposure to bring out certain areas indicated by dragging the right mouse button over them in *ximage*:

```
     ximage -op -t 75 final.pic | pcond -i .5 -b 0.2 -t
     125 -f film.cal final.pic > film.pic
```

AUTHOR

Greg Ward

SEE ALSO

getinfo(1), macbethcal(1), pcompos(1), pflip(1), pinterp(1), pvalue(1), protate(1), ra_xyze(1), rad(1), rpict(1), ximage(1)

NAME

pdfblur - generate views for depth-of-field blurring

SYNOPSIS

pdfblur aperture distance nsamp viewfile

DESCRIPTION

Pdfblur takes the given *viewfile* and computes *nsamp* views based on a focus distance of *distance* and an aperture diameter of *aperture* (both in world coordinate units). When rendered and averaged together, these views will result in a picture with the specified depth of field. Either *pin_terp*(*1*) or *rpict*(*1*) may be called to do the actual work. (The given *viewfile* must also be passed on the command line to the chosen renderer, since *pdfblur* provides supplemental view specifications only.)

For pinterp, feed the output of pdfblur to the standard input of *pinterp* and apply the *-B* option to blur views together. In most cases, a single picture with z-buffer is all that is required to get a satisfactory result, though the perfectionist may wish to apply three pictures arranged in a triangle about the aperature, or alternatively apply the *-ff* option together with the *-fr* option of *pinterp*. (The latter may actually work out to be faster, since rendering three views takes three times as long as a single view, and the *-fr* option will end up recomputing relatively few pixels by comparison.)

To use pdfblur with rpict, apply the -S option to indicate a rendering sequence, and set the *-o* option with a formatted file name to save multiple output pictures. When all the renderings are finished, combine them with the *pcomb*(*1*) program, using appropriate scalefactors to achieve an average. Note that using *rpict* is MUCH more expensive than using *pin_terp*, and it is only recommended if the scene and application absolutely demand it (e.g. there is prominent refraction that must be modeled accurately).

For both pinterp and rpict, the computation time will be proportional to the number of views from *pdfblur*. We have found a *nsamp* setting somewhere between 5 and 10 to be adequate for most images. Relatively larger values are appropriate for larger aperatures.

EXAMPLES

To use *pinterp* to simulate an aperture of 0.5 inches on a lens focused at a distance of 57 inches:

```
rpict -vf myview -x 640 -y 480 -z orig.zbf
scene.oct > orig.pic
pdfblur 0.5 57 8 orig.pic | pinterp -B -vf
orig.pic -x 640 -y 480 orig.pic orig.zbf >
blurry.pic
```

To use rpict exclusively to do the same:

```
pdfblur .5 57 5 myview | rpict -S 1 -vf myview -x
640 -y 480 -o view%d.pic scene.oct
pcomb -s .2 view1.pic -s .2 view2.pic -s .2
view3.pic -s .2 view4.pic -s .2 view5.pic >
blurry.pic
```

AUTHOR

Greg Ward

BUGS

This program really only works with perspective views.

SEE ALSO

`pcomb(1)`, `pinterp(1)`, `pmblur(1)`, `rcalc(1)`, `rpict(1)`, `vwright(1)`

NAME

pexpand - expand requested commands in metafile

SYNOPSIS

pexpand [+/-EPDOCSURIlrtmvsp] file ..

DESCRIPTION

Pexpand reads each metafile *file* in sequence and expands any commands specified in a '+' option, and deletes any commands specified in a '-' option. This is necessary because most drivers will not support many metafile commands such as 'v' (vector string) and 'O', 'C', and 's' (segments). *Pexpand* will expand '+' instances into the corresponding primitives which are supported, and delete all '-' instances.

Certain commands are currently considered basic, and must be supported by all drivers. The commands 'D', 'E', 'S', 'U', 'R', 'l', 'r', and 't' are basic (see `metafile(5)`).

If no input files are specified, the standard input is read.

+EPDOCSURIlrtmvsp
: Expand the requested command(s).

-EPDOCSURIlrtmvsp
: Delete the requested command(s).

EXAMPLE

To expand vector strings and segements, and delete pauses from "meta":

```
pexpand +vOCs -P meta
```

FILES

`/usr/lib/meta/vchars.mta` (see metafile(5))

AUTHOR

Greg Ward

SEE ALSO

`metafile(5)`, `psort(1)`

NAME

pextrem - find minimum and maximum values in *Radiance* picture

SYNOPSIS

pextrem [-o] [picture]

DESCRIPTION

Pextrem locates the minimum and maximum values for the input *picture*, and prints their pixel locations and color values. The first line printed contains the x and y pixel location (x measured from the left margin, y measured from the bottom), followed by the red, green and blue values. The second line printed contains the same information for the maximum value.

The -o option prints the original (radiance) values, undoing any exposure or color correction done on the picture.

If no input picture is given, the standard input is read.

AUTHOR

Greg Ward

BUGS

The luminance value is used for comparison of pixels, although in certain anomolous cases (ie. highly saturated colors) it is possible that *pextrem* will not pick the absolute minimum or maximum luminance value. This is because a fast integer-space comparison is used. A more reliable floating-point comparison would be slower by an order of magnitude.

SEE ALSO

`falsecolor(1)`, `getinfo(1)`, `pcomb(1)`, `pcompos(1)`, `pextrem(1)`, `pfilt(1)`, `pflip(1)`, `protate(1)`, `psign(1)`, `rpict(1)`, `ximage(1)`

NAME

pfilt - filter a *Radiance* picture

SYNOPSIS

pfilt [options] [file]

DESCRIPTION

Pfilt performs anti-aliasing and scaling on a *Radiance* picture. The program makes two passes on the picture file in order to set the exposure to the correct average value. If no *file* is given, the standard input is read.

-x res

Set the output x resolution to res. This must be less than or equal to the x dimension of the target device. If *res* is given as a slash followed by a real number, the input resolution is divided by this number to get the output resolution. By default, the output resolution is the same as the input.

-y *res*

Set the output y resolution to *res*, similar to the specification of the x resolution above.

-p *rat*

Set the pixel aspect ratio to *rat*. Either the x or the y resolution will be reduced so that the pixels have this ratio for the specified picture. If *rat* is zero, then the x and y resolutions will adhere to the given maxima. Zero is the default.

-c

Pixel aspect ratio is being corrected, so do not write PIXASPECT variable to output file.

-e *exp*

Adjust the exposure. If *exp* is preceded by a '+' or '-', the exposure is interpreted in f-stops (ie. the power of two). Otherwise, *exp* is interpreted as a straight multiplier. The individual primaries can be changed using *-er, -eg* and *-eb*. Multiple exposure options have a cumulative effect.

-t lamp

Color-balance the image as if it were illuminated by fixtures of the given type. The specification must match a pattern listed in the lamp lookup table (see the -f option below).

-f lampdat

Use the specified lamp lookup table rather than the default (lamp.tab).

-1

Use only one pass on the file. This allows the exposure to be controlled absolutely, without any averaging. Note that a single pass is much quicker and should be used whenever the desired exposure is known and star patterns are not required.

-2

Use two passes on the input. This is the default.

-b

Use box filtering (default). Box filtering averages the input pixels corresponding to each separate output pixel.

-r *rad*

Use Gaussian filtering with a radius of *rad* relative to the output pixel size. This option with a radius around 1 and a reduction in image width and height of 2 or 3 produces the highest quality pictures. A radius greater than 1 results in a defocused picture.

-m *frac*

Limit the influence of any given input pixel to *frac* of any given output pixel. This option may be used to mitigate the problems associated with inadequate image sampling, at the expense of a slightly blurred image. The fraction given should not exceed the output picture dimensions over the input picture dimensions (x_o*y_o/x_i/y_i), or blurring will occur over the entire image. This option implies the *-r* option for Gaussian filtering, which defaults to a radius of 1.

-h *lvl*

Set intensity considered "hot" to *lvl*. This is the level above which areas of the image will begin to exhibit star diffraction patterns (see below). The default is 100 watts/sr/m2.

-n *N*

Set the number of points on star patterns to *N*. A value of zero turns star patterns off. The default is 0. (Note that two passes are required for star patterns.)

-s *val*

Set the spread for star patterns to *val*. This is the value a star pattern will have at the edge of the image. The default is .0001.

-a

Average hot spots as well. By default, the areas of the picture above the hot level are not used in setting the exposure.

ENVIRONMENT

RAYPATH directories to search for lamp lookup table

FILES

```
/usr/tmp/rt??????
```

AUTHOR

Greg Ward

SEE ALSO

getinfo(1), ies2rad(1), pcompos(1), pflip(1), pinterp(1), pvalue(1), protate(1), rad(1), rpict(1), ximage(1)

NAME

pflip - flip a *Radiance* picture.

SYNOPSIS

pflip [-h][-v][-c] input [output]

DESCRIPTION

Pflip flips a *Radiance* picture horizontally and/or vertically. The *-h* option results in a horizontal exchange, and the *-v* option results in a vertical exchange. Both options may be applied.

The -c option indicates that the action is to correct an improper original image orientation, thus the recorded scanline ordering should not be changed.

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `pcompos(1)`, `pfilt(1)`, `protate(1)`, `psign(1)`, `rpict(1)`

NAME

pinterp - interpolate/extrapolate view from pictures

SYNOPSIS

pinterp [view options][-t threshold][-z zout][-f *type*][-B][-a|q][-e exposure][-n] pictfile zspec ..

DESCRIPTION

Pinterp interpolates or extrapolates a new view from one or more *Radiance* pictures and sends the result to the standard output. The input picture files must contain correct view specifications, as maintained by *rpict*(*1*), *rview*(*1*), *pfilt*(*1*) and *pinterp*. Specifically, *pinterp* will not work on pictures processed by *pcompos*(*1*) or *pcomb*(*1*). Each input file must be accompanied by a z specification, which gives the distance to each pixel in the image. If *zspec* is an existing file, it is assumed to contain a short floating point number for each pixel, written in scanline order. This file is usually generated by the *-z* option of *rpict*(*1*). If *zspec* is a positive number rather than a file, it will be used as a constant value for the corresponding image. This may be useful for certain transformations on "flat" images or when the viewpoint remains constant.

The -n option specifies that input and output z distances are along the view direction, rather than absolute distances to intersection points. This option is usually appropriate with a constant z specification, and should not be used with *rpict*(*1*) z files.

The -z option writes out interpolated z values to the specified file. Normally, this information is thrown away.

Pinterp rearranges the pixels from the input pictures to produce a reasonable estimate of the desired view. Pixels that map within the *-t* threshold of each other (.02 times the z distance by default) are considered coincident. With the *-a* option, image points that coincide will be averaged together, giving a smooth result. The *-q* option turns averaging off, which means that the first mapped pixel for a given point will be used. This makes the program run faster and take less memory, but at the expense of image quality. By default, two or more pictures are averaged together, and a single picture is treated with the faster algorithm. This may be undesirable when a quick result is desired from multiple input pictures in the first case, or a single picture is being reduced in size (anti-aliased) in the second case.

Portions which were hidden or missing in the input pictures must be "filled in" somehow, and a number of methods are provided by the *-f* option. The default value for this option is *-fa*, which results in both foreground and background filling. The foreground fill algorithm spreads each input pixel to cover all output pixels within a parallelogram corresponding to that pixel's projection in the new view. Without it, each input pixel contributes to at most one output pixel. The background algorithm fills in those areas in the final picture that have not been filled with foreground pixels. It does this by looking at the boundary surrounding each blank area and picking the farthest pixels to each side, assuming that this will make a suitable background. The *-ff* option tells the program to use only the foreground fill, the *-fb* option says use only background fill, and the *-f0* option says not to use either fill algorithm.

Even when both fill algorithms are used, there may still be some unfilled pixels. By default, these pixels are painted black and assigned a z distance of zero. The *-fc* option can be used to change the color used for unfilled pixels, and the *-fz* option can be used to set the z distance (always along the view direction). Alternatively, the *-fr* option can be used to compute these pixels using *rtrace*(*1*). The argument to this option is a quoted string containing arguments for *rtrace*. It must contain the octree used to generate the input pictures, along with any other options necessary to match the calculation used for the input pictures. The *-fs* option can be used to place a limit on the distance (in pixels) over which the background fill algorithm is used. The default value for this option is 0, which is interpreted as no limit. A value of 1 is equivalent to turning background fill off. When combined with the *-fr* option, this is roughly equivalent to the *-ps* option of *rpict*(*1*).

In order of increasing quality and cost, one can use the -fa option alone, or the *-fr* option paired with *-fs* or *-ff* or *-f0*. The last combination will result in the recalculation of all pixels not adequately accounted for in the input pictures, with an associated computational expense. It is rare that the *-fs* option results in appreciable image degradation, so it is usually the second combination that is used when the background fill algorithm results in objectionable artifacts.

The -B option may be used to average multiple views read from the standard input into a single, blurred output picture. This is similar to running *pinterp* multiple times and averaging the output together with a program like *pcomb*(*1*). This option is useful for simulating motion blur and depth of field. (See also *pdfblur*(*1*).) The input views are reported in the information header of the output file, along with the averaged view. The picture dimensions computed from the first view will be the ones used, regardless whether or not the subsequent views agree. (The reported pixel aspect ratio in the output is determined from these original dimensions and the averaged view.) Note that the expense of the *-fr* option is proportional to the number of views computed, and the *-z* output file will be the z-buffer of the last view interpolated rather than an averaged distance map.

In general, pinterp performs well when the output view is flanked by two nearby input views, such as might occur in a walk-through animation sequence. The algorithms start to break down when there is a large difference between the view desired and the view(s) provided. Specifically, obscured objects may appear to have holes in them and large areas at the image borders may not be filled by the foreground or background algorithms. Also, specular reflections and highlights will not be interpolated very well, since their view-dependent appearance will be incompletely compensated for by the program. (The *-a* option offers some benefit in this area.)

The -e option may be used to adjust the output image exposure, with the same specification given as for *pfilt*. The actual adjustment will be rounded to the nearest integer fstop if the *-q* option is in effect (or there is only a single input picture).

EXAMPLE

To interpolate two frames of a walk-through animation, anti-alias to 512x400 and increase the exposure by 2.5 fstops:

```
pinterp -vf 27.vf -a -x 512 -y 400 -e +2.5 30.pic
30.z 20.pic 20.z > 27.pic
```

To extrapolate a second eyepoint for a stereo pair and recalculate background regions:

```
pinterp -vf right.vf -ff -fr "-av .1 .1 .1
scene.oct" left.pic left.z > right.pic
```

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `pdfblur(1)`, `pfilt(1)`, `pmblur(1)`, `rpict(1)`, `ranimate(1)`, `rtrace(1)`, `rview(1)`

NAME

plotin - convert plot(5) to metafile(5) primitives

SYNOPSIS

plotin file ..

DESCRIPTION

Plotin reads each `plot(5)` *file* in sequence and converts it to output suitable for use by the metafile filters. If no input files are specified, the standard input is read.

EXAMPLE

To plot the graph example.grf to the imagen:

```
     graph < example.grf | plotin | impress | ipr
```

FILES

see `pexpand(1)`

AUTHOR

Greg Ward

SEE ALSO

`graph(1G)`, `metafile(5)`, `plot(1)`, `plot(5)`, `plotout(1)`

NAME

pmblur - generate views for camera motion blurring

SYNOPSIS

pmblur speed nsamp v0file v1file

DESCRIPTION

Pmblur takes two viewfiles and generates *nsamp* views starting from *v0file* and moving towards *v1file*. When rendered and averaged together, these views will result in a picture with motion blur due to a camera changing from v0 to v1 in a relative time unit of 1, whose shutter is open starting at v0 for speed of these time units. Either *pinterp*(*1*) or *rpict*(*1*) may be called to do the actual work. (The given *v0file* must also be passed on the command line to the chosen renderer, since *pmblur* provides supplemental view specifications only.)

For pinterp, feed the output of pmblur to the standard input of *pinterp* and apply the *-B* option to blur views together. In most cases, two pictures with z-buffers at v0 and v1 will get a satisfactory result, though the perfectionist may wish to apply the *-ff* option together with the *-fr* option of *pin_terp*.

To use pmblur with rpict, apply the -S option to indicate a rendering sequence, and set the *-o* option with a formatted file name to save multiple output pictures. When all the renderings are finished, combine them with the *pcomb*(*1*) program, using appropriate scalefactors to achieve an average. Note that using *rpict* is MUCH more expensive than using *pin_terp*, and it is only recommended if the scene and application absolutely demand it (e.g. there is prominent refraction that must be modeled accurately).

For both pinterp and rpict, the computation time will be proportional to the number of views from *pmblur*. We have found a *nsamp* setting somewhere between 5 and 10 to be adequate for most images. Relatively larger values are appropriate for faster camera motion.

EXAMPLES

To use *pinterp* to simulate motion blur between two frames of a walkthrough animation, where the camera shutter is open for 1/4 of the interframe distance:

```
pmblur .25 8 fr1023.pic fr1024.pic | pinterp -B
-vf fr1023.pic -x 640 -y 480 fr1023.pic fr1023.zbf
fr1024.pic fr1024.zbf > fr1023b.pic
```

AUTHOR

Greg Ward

BUGS

Changes in the view shift and lift vectors or the fore and aft clipping planes are not blurred.

SEE ALSO

`pcomb(1)`, `pdfblur(1)`, `pinterp(1)`, `rcalc(1)`, `rpict(1)`, `vwright(1)`

NAME

protate - rotate a *Radiance* picture.

SYNOPSIS

protate [-c] input [output]

DESCRIPTION

Protate rotates a *Radiance* picture 90 degrees clockwise. This is useful for output on hardcopy devices with aspect ratios opposite to the input picture.

The -c option indicates that the action is to correct an improper original image orientation, thus the recorded scanline ordering should not be changed.

EXAMPLES

To rotate an image 90 degrees counterclockwise, use `pflip(1)` first to rotate the image 180 degrees:

```
     pflip -h -v input temp
     protate temp output
     rm temp
```

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `pcompos(1)`, `pfilt(1)`, `pflip(1)`, `psign(1)`, `rpict(1)`

NAME

psign - produce a *Radiance* picture from text.

SYNOPSIS

psign [options] [text]

DESCRIPTION

Psign produces a *Radiance* picture of the given *text*. The output dimensions are determined by the character height, aspect ratio, number of lines and line length. (Also the character size if text squeezing is used.) If no *text* is given, the standard input is read.

-cb r g b
: Set the background color to r g b The default is white (1 1 1).

-cf *r g b*
: Set the foreground color to *r g b* The default is black (0 0 0).

-dr Text reads to the right (default).

-du Text reads upwards.

-dl Text reads to the left (upside down).

-dd Text reads downwards.

-h *cheight*
: Set the character height to *cheight*. The default is 32 pixels.

-a *aspect*
: Set the character aspect ratio (height/width) to *aspect*. The default value is 1.67.

-x *xsize*
: Set the horizontal image size to *xsize*. Use with -*y* option (below) in place of the -*h* specification to control output image size directly. If the character aspect ratio (-*a* option, above) is nonzero, then one of the specified x or y output dimensions may be reduced to maintain this ratio. If direction is right (-*dr*) or left (-*dl*), then it is not necessary to give the -*y* option, since it can be computed from the character height (-*h*).

-y *ysize*
: Set the vertical image size to *ysize*. Use with the -*x* option (described above). If direction is up (-*du*) or down (-*dd*), then it is not necessary to give the -*x* option, since it can be computed from the character height (-*h*).

-s *spacing*

Set the intercharacter spacing to *spacing*. The magnitude of this value is multiplied by the character height over the aspect ratio (ie. the character width) to compute the desired distance between characters in the output. The sign of the value, positive or negative, determines how this ideal spacing is used in the actual placement of characters. If *spacing* is positive, then the overall width of the line will not be affected, nor will indentation of textual elements. Thus, the text format will be mostly unaffected. However, spacing between characters will reflect their relative size for a more natural appearance. If *spacing* is negative, characters will be squeezed together to meet the spacing critereon, regardless of how it might affect the format of the output. The default value for *spacing* is zero, which is interpreted as uniformly spaced characters.

-f *fontfile*

Load the font from *fontfile*. The default font is helvet.fnt

EXAMPLE

To put a big "Hi!" on the terminal:

```
psign -h 22 -a 1 -cb 0 0 0 -cf 1 1 1 Hi | ttyimage
```

ENVIRONMENT

`RAYPATH` path to search for font files

AUTHOR

Greg Ward

BUGS

The entire bitmap is stored in memory, which can be a problem for large and/or high-resolution signs.

SEE ALSO

`aed(1)`, `getinfo(1)`, `pcompos(1)`, `pfilt(1)`, `ttyimage(1)`

NAME

psmeta - convert metafile to PostScript

SYNOPSIS

psmeta file ..

DESCRIPTION

Psmeta reads each metafile *file* in sequence and converts it to PostScript output suitable for a standard letter-size page. The file produced may also be read into programs that can handle Encapsulated PostScript. If no input files are specified, the standard input is read.

EXAMPLE

To print the plot file example.plt to the printer:

```
bgraph example.plt | psmeta | lpr
```

AUTHOR

Greg Ward

SEE ALSO

`bgraph(1)`, `igraph(1)`, `imagew(1)`, `lpr(1)`, `mx80(1)`, `t4014(1)`

NAME

psort - sort primitives in metafile as requested

SYNOPSIS

psort [+/-x][+/-y][+/-X][+/-Y] file ..

DESCRIPTION

Psort reads each metafile *file* in sequence and sorts primitives between globals according to the option specification. Lower case options mean the corresponding minimum, upper case indicates the maximum value. A '+' before the option means sort in order of increasing values, '-' means decreasing. The order the options appear on the command line is the order in which the extrema are examined. For example, the options *-Y +x* would mean "sort on decreasing ymax, then increasing xmin values".

If no input files are specified, the standard input is read.

EXAMPLE

To sort the file "meta" in order of increasing xmax, then decreasing ymin:

psort +X -y meta

FILES

```
     /usr/tmp/psXXXXa
     /usr/tmp/psXXXXb
```

BUGS

Aborting the program will sometimes leave files in /usr/tmp.

AUTHOR

Greg Ward

SEE ALSO

`metafile(5)`, `pexpand(1)`

NAME

pvalue - convert *Radiance* picture to/from alternate formats

SYNOPSIS

pvalue [options] [file]
pvalue -r [options] file1 file2 file3

DESCRIPTION

Pvalue converts the pixels of a *Radiance* picture to or from another format. In the default mode, pixels are sent to the standard output, one per line, in the following ascii format:

```
xpos ypos red green     blue
```

If no file is given, the standard input is read.

The reverse conversion option (-r) may be used with a single input file or when reading from the standard input, but if the second form is used with three separate input files, the three primaries are presumed to be separated in these files.

-u
: Print only unique values in the output, skipping runs of equal pixels. Specifying *+u* turns this option off, which is the default.

-o
: Print original values, before exposure compensation. Specifying *+o* uses final values, which is the default.

-h
: Do not print header. Specifying *+h* causes the header to be printed, which is the default.

-H
: Do not print the resolution string. (See also the *-r* option below.) Specifying an input resolution for reverse conversion also turns this option off. Specifying *+H* causes the resolution string to be printed, which is the default.

-s *nbytes*
: Skip the specified number of bytes on the input header. This option is useful for skipping unintelligible headers in foreign file formats. (Does not work when reading from standard input.)

-e *exposure*

Adjust the exposure by the amount specified. If the exposure is being given as a conversion factor, use +*e* instead, so an EXPOSURE line will not be recorded in the header (if any).

-g *gamma*

Set gamma correction for conversion. When converting from a *Radiance* picture to another format, the inverse gamma is applied to correct for monitor response. When converting to a *Radiance* picture (-*r* option), the gamma is applied directly to recover the linear values. By default, *gamma* is set to 1.0, meaning no gamma correction is performed.

-d

Data only, do not print x and y pixel position.

-da

Same as -*d*.

-di

Print ascii integer values from 0 to 255+. If +*di* is given, the integer values will be preceded by the x and y pixel locations.

-db

Print binary byte values from 0 to 255.

-df

Print binary float values.

-dd

Print binary double values.

-R

Reverse ordering of colors so that the output is blue then green then red. The default ordering (specified with +*R*) is red then green then blue.

-n

The RGB values are non-interleaved, meaning that all the red, green and blue data are stored together in separate chunks. Interleaving may be turned on with the +*n* option, which is the default.

-b

Print brightness values rather than RGB. Specifying +*b* turns this option off, which is the default.

-pP

Put out only the primary *P*, where *P* is one of upper or lower case `R', `G' or `B' for red, green or blue, respectively. This option may be used to separate the *Radiance* primaries into three files with three separate runs of *pvalue*, or only one file when only one primary is needed. Note that there is no space between this option and its argument.

-r

Perform reverse conversion. Input is in the format given by the other options. The x and y resolution must be specified on the command line, unless the image file contains a *Radiance* resolution string at the beginning (see -*H* option above and -*y* option below). Specifying +*r* converts from a *Radiance* picture to other values, which is the default.

-y *res*

Set the output y resolution to *res*. If +*y* is specified, then the scanlines are assumed to be in increasing order (ie. bottom to top). The default value for this option is 0, which means that the picture size and scanline order must appear as the first line after the header (if any) in the input file. Either an upper or lower case `Y' may be used for this option. Since *Radiance* files always contain such a line, this option has no effect for forward conversions.

+x *res*

Set the output x resolution to *res*. If -*x* is specified, then the scanlines are assumed to be in decreasing order (ie. right to left). The ordering of the -*y* and +*x* options determines whether the scanlines are sorted along x or along y. Most *Radiance* pictures are sorted top to bottom, then left to right. This corresponds to a specification of the form ``-y yres +x xres''. Either an upper or lower case `X' may be used for this option. Like the -*y* option, -*x* options have no effect for forward conversions.

EXAMPLE

To look at the original, unique pixels in picture:

```
pvalue -o -u picture | more
```

To convert from a 512x400 8-bit greyscale image in bottom to top, left to right scanline ordering:

```
pvalue -r -db -b -h +y 400 +x 512 input.im >
flipped.pic pflip -v flipped.pic > final.pic
```

AUTHOR

Greg Ward

BUGS

The *-r* option does not work with the *-u* option. Also, input pixel locations are ignored during a reverse conversion, so this information is not used in determining the scanline ordering or picture size.

SEE ALSO

`getinfo(1)`, `pcompos(1)`, `pfilt(1)`, `pflip(1)`, `protate(1)`, `rpict(1)`, `rtrace(1)`, `rview(1)`

NAME

rad - render a *Radiance* scene

SYNOPSIS

rad [-s][-n][-t][-e][-V][-w][-v view][-o device] rfile [VAR=value ..]

DESCRIPTION

Rad is an executive program that reads the given *rfile* and makes appropriate calls to *oconv*(*1*), *mkillum*(*1*), *rpict*(*1*), *pfilt*(*1*), and/or *rview*(*1*) to render a specific scene. Variables in *rfile* give input files and qualitative information about the rendering(s) desired that together enable *rad* to intelligently set parameter values and control the simulation.

Normally, commands are echoed to the standard output as they are executed. The *-s* option tells *rad* to do its work silently. The *-n* option tells *rad* not to take any action (ie. not to actually execute any commands). The *-t* option tells *rad* to bring rendering files up to date relative to the input (scene description) files, without performing any actual calculations. If no octree exists, it is still necessary to run *oconv*(*1*) to create one, since the *-t* option will not create invalid (i.e. empty) files, and a valid octree is necessary for the correct operation of *rad*. The *-e* option tells *rad* to explicate all variables used for the simulation, including default values not specified in the input file, and print them on the standard output.

Normally, rad will produce one picture for each view given in *rfile*. The *-v* option may be used to specify a single desired view. The *view* argument may either be a complete view specification (enclosed in quotes and beginning with an optional identifier) or a number or single-word identifier to match a view defined in *rfile*. If the argument is one of the standard view identifiers, it may or may not be further elaborated in *rfile*. (See "view" variable description, below.) If the argument does not match any views in *rfile* and is not one of the standard views, no rendering will take place. This may be convenient when the only action desired of *rad* is the rebuilding of the octree. In particular, the argument "0" will never match a view.

If the -V option is given, each view will be printed on the standard output before being applied, in a form suitable for use in a view file or *rpict* rendering sequence. This is helpful as feedback or for accessing the *rad* view assignments without necessarily starting a rendering.

By default, rad will run rpict and pfilt to produce a picture for each view. The *-o* option specifies an output device for *rview* (usually "x11") and runs this interactive program instead, using the first view in *rfile* or the view given with the -*v* option as the starting point.

Additional variable settings may be added or overridden on the command line following *rfile*. Upper case variables specified more than once will result in a warning message (unless the -*w* option is present), and the last value given will be the one used.

The -w option turns off warnings about multiply and misassigned variables.

Rendering variable assignments appear one per line in rfile. The name of the variable is followed by an equals sign ('=') and its value(s). The end of line may be escaped with a backslash ('\') though it is not usually necessary since additional variable values may be given in multiple assignments. Variables that should have only one value are given in upper case. Variables that may have multiple values are given in lower case. Variables may be abbreviated by their first three letters. Comments in *rfile* start with a pound sign ('#') and proceed to the end of line.

The rendering variables, their interpretations and default values are given below.

OCTREE

> The name of the octree file. The default name is the same as *rfile* but with any suffix replaced by ".oct".

ZONE

> This variable specifies the volume of interest for this simulation. The first word is either "Interior" or "Exterior", depending on whether the zone is to be observed from the inside or the outside, respectively. (A single letter may be given, and case does not matter.) The following six numbers are the minimum and maximum X coordinates, minimum and maximum Y, and minimum and maximum Z for the zone perimeter. It is important to give the zone as it is used to determine many of the rendering parameters. The default exterior zone is the bounding cube for the scene as computed by *oconv*.

EXPOSURE

This variable tells *rad* how to adjust the exposure for display. It is important to set this variable properly as it is used to determine the ambient value. An appropriate setting may be discovered by running *rview* and noting the exposure given by the "exposure =" command. As in *rview* and *pfilt*, the exposure setting may be given either as a multiplier or as a number of f-stop adjustments (eg. +2 or -1.5). There is no default value for this variable. If it is not given, an average level will be computed by *pfilt* and the ambient value will be set to 10 for exterior zones and 0.01 for interior zones.

scene

This variable is used to specify one or more scene input files. These files will be given together with the materials file(s) and any options specified by the "oconv" variable to *oconv* to produce the octree given by the "OCTREE" variable. If the "scene" variable is not present, then the octree must already exist in order for *rad* to work. Even if this variable is given, *oconv* will not be run unless the octree is out of date with respect to the input files. Note that the order of files in this variable is important for *oconv* to work properly, and files given in later variable assignments will appear after previous ones on the *oconv* command line.

materials

This variable is used to specify files that, although they must appear on the *oconv* command line, do not affect the actual octree itself. Keeping the materials in separate files allows them to be modified without requiring the octree to be rebuilt (a sometimes costly procedure). These files should not contain any geometry, and the *-f* option must not be given in the "oconv" variable for this to work.

illum

This variable is used to specify files with surfaces to be converted into illum sources by *mkil_lum*(*1*). When this variable is given, additional octree files will be created to contain the scene before and after illum source conversion. These files will be named according to the (default) value of the *OCTREEE* variable, with either a '0' or a '1' appearing just before the file type suffix (usually ".oct").

objects

This variable is used for files that, although they do not appear on the *oconv* command line, contain geometric information that is referenced indirectly by the scene files. If any of these files is changed, the octree will be rebuilt. (The *raddepend*(*1*) command may be used to find these dependencies automatically.)

view

This variable is used to specify a desired view for this zone. Any number of "view" lines may be given, and each will result in a rendered picture (unless the *-v* or *-o* option is specified). The value for this variable is an optional identifier followed by any number of view options (see *rpict*(*1*) for a complete listing). The identifier is used in file naming and associating a desired view with the *-v* command line option. Also, there are several standard view identifiers defined by *rad*. These standard views are specified by strings of the form "[Xx]?[Yy]?[Zz]?[vlcah]?". (That is, an optional upper or lower case X followed by an optional upper or lower case Y followed by an optional upper or lower case Z followed by an optional lower case V, L, C, A or H.) The letters indicate the desired view position, where upper case X means maximum X, lower case means minimum and so on. The final letter is the view type, where 'v' is perspective (the default), 'l' is parallel, 'c' is a cylindrical panorama, A perspective view from maximum X, minimum Y would be "Xy" or "Xyv". A parallel view from maximum Z would be "Zl". If "ZONE" is an interior zone, the standard views will be inside the perimeter. If it is an exterior zone, the standard views will be outside. Note that the standard views are best used as starting points, and additional arguments may be given after the identifier to modify a standard view to suit a particular model. The default view is "X" if no views are specified. A single specified view of "0" means no views will be automatically generated.

UP

The vertical axis for this scene. A negative axis may be specified with a minus sign (eg. "-Y"). There is no default value for this variable, although the standard views assume Z is up if no other axis is specified.

RESOLUTION

This variable specifies the desired final picture resolution. If only a single number is given, this value will be used for both the horizontal and vertical picture dimensions. If two numbers are given, the first is the horizontal resolution and the second is the vertical resolution. If three numbers are given, the third is taken as the pixel aspect ratio for the final picture (a real value). If the pixel aspect ratio is zero, the exact dimensions given will be those produced. Otherwise, they will be used as a frame in which the final image must fit. The default value for this variable is 512.

QUALITY

This variable sets the overall rendering quality desired. It can have one of three values, "Low", "Medium" or "High". These may be abbreviated by their first letter, and may be in upper or lower case. Most of the rendering options will be affected by this setting. The default value is "L".

PENUMBRAS

This is a boolean variable indicating whether or not penumbras are desired. A value of "TRUE" will result in penumbras (soft shadows), and a value of "FALSE" will result in no penumbras (sharp shadows). True and false may be written in upper or lower case, and may be abbreviated by a single letter. Renderings generally proceed much faster without penumbras. The default value is "F".

INDIRECT

This variable indicates how many diffuse reflections are important in the general lighting of this zone. A direct lighting system (eg. fluorescent troffers recessed in the ceiling) corresponds to an indirect level of 0. An indirect lighting system (eg. hanging fluorescents directed at a reflective ceiling) corresponds to an indirect level of 1. A diffuse light shelf reflecting sunlight onto the ceiling would correspond to an indirect level of 2. The setting of this variable partially determines how many interreflections will be calculated. The default value is 0.

PICTURE

This is the root name of the output picture file(s). This name will have appended the view identifier (or a number if no id was used) and a ".pic" suffix. If a picture corresponding to a specific view exists and is not out of date with respect to the given octree, it will not be rerendered. The default value for this variable is the root portion of *rfile*.

RAWFILE

This is the root name of the finished, raw rpict output file. If specified, *rad* will rename the original *rpict* output file once it is finished and filtered rather than removing it, which is the default action. The given root name will be expanded in the same way as the "PICTURE" variable, and if the "RAWFILE" and "PICTURE" variables are identical, then no filtering will take place.

ZFILE

This is the root name of the raw distance file produced by the *-z* option of *rpict*. To this root name, an underscore plus the view name plus a ".zbf" suffix will be added. If no "ZFILE" is specified, none will be produced.

AMBFILE

This is the name of the file where "ambient" or diffuse interreflection values will be stored by *rpict* or *rview*. Although it is not required, an ambient file should be given whenever an interreflection calculation is expected. This will optimize successive runs and minimize artifacts. An interreflection calculation will take place when the "QUALITY" variable is set to High, or when the "QUALITY" variable is set to Medium and "INDIRECT" is positive. There is no default value for this variable.

DETAIL

This variable specifies the level of visual detail in this zone, and is used to determine image sampling rate, among other things. If there are few surfaces and simple shading, then this should be set to Low. For a zone with some furniture it might be set to Medium. If the space is very cluttered or contains a lot of geometric detail and textures, then it should be set to High. The default value is "M".

VARIABILITY

This variable tells *rad* how much light varies over the surfaces of this zone, and is used to determine what level of sampling is necessary in the indirect calculation. For an electric lighting system with uniform coverage, the value should be set to Low. For a space with spot lighting or a window with sky illumination only, it might be set to Medium. For a space with penetrating sunlight casting bright patches in a few places, it should be set to High. The default value is "L".

OPTFILE

This is the name of a file in which *rad* will place the appropriate rendering options. This file can later be accessed by *rpict* or *rview* in subsequent manual runs using the at-sign ('@') file insert option. (Using an "OPTFILE" also reduces the length of the rendering command, which improves appearance and may even be necessary on some systems.) There is no default value for this variable.

REPORT

This variable may be used to specify a reporting interval for batch rendering. Given in minutes, this value is multiplied by 60 and passed to *rpict* with the -*t* option. If a filename is given after the interval, it will be used as the error file for reports and error messages instead of the standard error. (See the -*e* option of *rpict*(*1*).) There is no default value for this variable.

oconv

This variable may be used to specify special options to *oconv*. See the *oconv*(*1*) manual page for a list of valid options.

mkillum

This variable may be used to specify additional options to *mkillum*. See the *rtrace*(*1*) manual page for a list of valid options.

render

This variable may be used to specify additional options to *rpict* or *rview*. These options will appear after the options set automatically by *rad*, and thus will override the default values.

pfilt

This variable may be used to specify additional options to *pfilt*. See the *pfilt*(*1*) manual page for details.

EXAMPLES

A minimal input file for *rad* might look like this:

```
# ::::::::::
# sample.rif
# ::::::::::
# The octree we want to use:
OCTREE= tutor.oct
# w/o this line, name would be "sample.oct"
# Our scene input files:
scene= sky.rad outside.rad room.rad srcwindow.rad
# The interior zone cavity:
ZONE= I 0 3 0 2 0 1.75  # default would be scene
bounding cube
# The z-axis is up:
UP= Z    # no default - would use view spec.
# Our exposure needs one f-stop boost:
EXPOSURE= +1  # default is computed ex post facto
```

Note that we have not specified any views in the file above. The standard default view "X" would be used if we were to run *rad* on this file. If we only want to see what default values *rad* would use without actually executing anything, we can invoke it thus:

```
rad -n -e sample.rif
```

This will print the variables we have given as well as default values *rad* has assigned for us. Also, we will see the list of commands that *rad* would have executed had the *-n* option not been present. (Note if the octree, "tutor.oct", is not present, an error will result as it is needed to determine some of the opiton settings.)

Different option combinations have specific uses, ie:

```
rad -v 0 sample.rif OPT=samp.opt
# build octree, put options in "sample.opt"
rad -n -e -s sample.rif > full.rif
# make a complete rad file
rad -n sample.rif > script.sh
# make a script of commands
rad -V -v Zl -n -s sample.rif > plan.vf
# make a plan view file
rad -t sample.rif
# update files after minor change to input
rad -s sample.rif &
# execute silently in the background
```

If we decide that the default values *rad* has chosen for our variables are not all appropriate, we can add some more assignments to the file:

```
QUAL= MED # default was Low
DET= Low  # default was Medium - our space is
almost empty
PEN= True # we want to see soft shadows from our
window
VAR= hi   # daylight can result in fairly harsh
lighting
view= XYa -vv 120   # let's try a fisheye view
PICT= tutor    # our picture name will be
"tutor_XYa.pic"
```

Note the use of abbreviations, and the modification of a standard view. Now we can invoke *rad* to take a look at our scene interactively with *rview*:

```
rad -o x11 sample.rif
```

Rad will run oconv first to create the octree (assuming it doesn't already exist), then *rview* with a long list of options. Let's say that from within *rview*, we wrote out the view files "view1.vp" and "view2.vp". We could add these to "sample.rif" like so:

```
view= vw1 -vf view1.vp   # Our first view
view= vw2 -vf view2.vp   # Our second view
RESOLUTION= 1024    # Let's go for a higher
resolution result
```

To start *rview* again using vw2 instead of the default, we use:

```
rad -o x11 -v vw2 sample.rif
```

Once we are happy with the variable settings in our file, we can run *rad* in the background to produce one image for each view:

```
rad sample.rif REP=5 >& errfile &
```

This will report progress every five minutes to "errfile".

FILES

`$(PICTURE)_$(view).unf`	Unfinished output of *rpict*

AUTHOR

Greg Ward

BUGS

Incremental building of octrees is not supported as it would add considerable complexity to *rad*. Complicated scene builds should still be left to *make*(*1*), which has a robust mechanism for handling hierarchical dependencies. If *make* is used in this fashion, then only the "OCTREE" variable of *rad* is needed.

The use of some pfilt options is awkward, since the "EXPOSURE" variable results in a single pass invocation (the *-1* option of *pfilt*) and two passes are necessary for certain effects, such as star patterns. The way around this problem is to specify a "RAWFILE" that is the same as the "PICTURE" variable so that no filtering takes place, then call *pfilt* manually. This is preferable to leaving out the "EXPOSURE" variable, since the exposure level is needed to accurately determine the ambient value for *rpict*.

The use of upper and lower case naming for the standard views may be problematic on systems that don't distinguish case in filenames.

SEE ALSO

`make(1)`, `mkillum(1)`, `oconv(1)`, `pfilt(1)`, `raddpend(1)`, `ranimate(1)`, `rpict(1)`, `rtrace(1)`, `rview(1)`, `touch(1)`, `vgaimage(1)`, `ximage(1)`

NAME

rad2mgf - convert *Radiance* scene description to Materials and Geometry Format

SYNOPSIS

rad2mgf [-dU] [input ..]

DESCRIPTION

Rad2mgf converts one or more *Radiance* scene files to the Materials and Geometry Format (MGF). Input units are specified with the *-mU* option, where *U* is one of 'm' (meters), 'c' (centimeters), 'f' (feet) or 'i' (inches). The assumed unit is meters, which is the required output unit for MGF (thus the need to know). If the input dimensions are in none of these units, then the user should apply *xform*(*1*) with the *-s* option to bring the units into line prior to translation.

The MGF material names and properties for the surfaces will be those assigned in *Radiance*. If a referenced material has not been defined, then its name will be invoked in the MGF output without definition, and the description will be incomplete.

LIMITATIONS

Although MGF supports all of the geometric types and the most common material types used in *Radiance*, there is currently no support for advanced BRDF materials, patterns, textures or mixtures. Also, the special types "source" and "antimatter" are not supported, and all light source materials are converted to simple diffuse emitters (except "illum" materials, which are converted to their alternates). These primitives are reproduced as comments in the output and must be replaced manually if necessary.

The *Radiance* "instance" type is treated specially. Rad2mgf converts each instance to an MGF include statement, using the corresponding transformation and a file name derived from the octree name. (The original octree suffix is replaced by ".mgf".) For this to work, the user must separately create the referenced MGF files from the original *Radiance* descriptions. The description file names can usually be determined using the *getinfo*(*1*) command run on the octrees in question.

EXAMPLE

To convert three *Radiance* files (in feet) to one MGF file:

```
rad2mgf -df file1.rad file2.rad file3.rad >
scene.mgf
```

To translate a *Radiance* materials file to MGF:

```
rad2mgf materials.rad > materials.mgf
```

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `ies2rad(1)`, `mgf2meta(1)`, `mgf2rad(1)`, `obj2rad(1)`, `oconv(1)`, `xform(1)`

MGF web site "http://radsite.lbl.gov/mgf/HOME.html"

NAME

raddepend - find *Radiance* scene dependencies

SYNOPSIS

raddepend file ..

DESCRIPTION

Raddepend uses *getbbox*(*1*) to expand scene file arguments and find file dependencies for *make*(*1*) or *rad*(*1*). *Raddepend* looks only in the current directory, so dependencies hidden elsewhere in the filesystem will not be found or named.

The output is the name of files, one per line, that were accessed during the expansion of the input file arguments. The file arguments are excluded from the list. If no input files are given, the standard input is read.

AUTHOR

Greg Ward

BUGS

On some older NFS systems, the file access dates are not updated promptly. As a result, *raddepend* may not be 100% reliable on these systems. If the output seems to be missing essential files, this is no doubt why. The only fix is to put in a longer sleep time between the *getbbox* call and the final *ls*(*1*).

SEE ALSO

`make(1)`, `oconv(1)`, `rad(1)`, `xform(1)`

NAME

ranimate - compute a *Radiance* animation

SYNOPSIS

ranimate [-s][-n][-e][-w] ranfile

DESCRIPTION

Ranimate is an executive program that reads the given *ran_file* and makes appropriate calls to *rad*(*1*), *rpict*(*1*), *pin_terp*(*1*), and/or *pfilt*(*1*) to render an animation. Variables in *ranfile* indicate input files, process servers (execution hosts), output directories and file names, and various other controls and options.

Normally, commands are echoed to the standard output as they are executed. The *-s* option tells *ranimate* to do its work silently. The *-n* option tells *ranimate* not to take any action (ie. not to actually execute any commands). The *-e* option tells *ranimate* to explicate all variables used for the animation, including default values not specified in the input file, and print them on the standard output.

The -w option turns off warnings about multiply and misassigned variables.

Normally, ranimate will produce one animation frame for each view given in the specified view file. If an animation has ended or been killed in an incomplete state, however, *rani_mate* will attempt to pick up where the earlier process left off. If the process is still running, or was started on another machine, *ranimate* will report this information and exit.

Animation variable assignments appear one per line in ran_file. The name of the variable is followed by an equals sign ('=') and its value(s). The end of line may be escaped with a backslash ('\') though it is not usually necessary since additional variable values may be given in multiple assignments. Variables that should have only one value are given in upper case. Variables that may have multiple values are given in lower case. Variables may be abbreviated by their first three letters, except for "host", which must have all four. Comments in *ranfile* start with a pound sign ('#') and proceed to the end of line.

The animation variables, their interpretations and default values are given below.

DIRECTORY

The name of the animation directory. All temporary files generated during the animation will be placed in this directory, which will be created by *ranimate* if it does not exist. A file named "STATUS" will also be created there, and will contain current information about the animation process. This variable has no default value, and its setting is required.

OCTREE

The name of the octree file for a static scene walk-through animation. There is no default value for this variable, and any setting will be ignored if the *ANIMATE* variable is also set (see below).

ANIMATE

The scene generation command for a dynamic animation. This command, if given, will be executed with the frame number as the final argument, and on its standard output it must produce the complete octree for that frame. Care must be taken that this command does not create any temporary files that might collide with same-named files created by other animation commands running in parallel. Also, the command should produce no output to the standard error, unless there is a fatal condition. (I.e., switch all warnings off; see the BUGS section, below.) There is no default animation command, and either this variable or the *OCTREE* variable must be set.

VIEWFILE

This variable names a file from which ranimate may extract the view for each frame in the animation. This file should contain one valid view per frame, starting with frame 1 on line 1, regardless of the setting of the *START* variable. An exception is made for a view file with only a single view, which is used for every frame of a dynamic scene animation. This variable is required, and there is no default value.

START

The initial frame number in this animation sequence. The minimum value is 1, and if a later starting frame is given, *ranimate* assumes that the earlier frames are included in some other *ranfile*, which has been previously executed. (See the *NEXTANIM* variable, below.) The default value is 1.

END

The final frame number in this sequence. The minimum value is equal to the *START* frame, and the default value is computed from the number of views in the given *VIEWFILE*.

EXPOSURE

This variable tells *ranimate* how to adjust the exposure for each frame. As in *pfilt*, the exposure setting may be given either as a multiplier or as a number of f-stop adjustments (eg. +2 or -1.5). Alternatively, a file name may be given, which *ranimate* will interpret as having one exposure value per line per frame, beginning with frame 1 at line 1. (See also the *VIEWFILE* variable, above.) There is no default value for this variable. If it is not given, an average level will be computed by *pfilt* for each frame.

BASENAME

The base output file name for the final frames. This string will be passed to the *-o* and *-z* options of rpict, along with appropriate suffixes, and thus should contain a *printf*(*3*) style integer field to distinguish one frame number from another. The final frames will use this name with a ".pic" suffix. The default value is the assigned *DIRECTORY* followed by "/frame%03d".

host

A host to use for command execution. This variable may be assigned a host name, followed by an optional number of parallel processes, followed by an optional directory (relative to the user's home directory on that machine), followed by an alternate user name. Multiple *host* assignments may appear. It is not advisable to specify more than one process on a single-CPU host, as this just tends to slow things down. The default value is "localhost", which starts a single process in the current directory of the local machine.

RIF

This variable specifies a *rad* input file to use as a source of rendering options and other variable settings. If given, *ranimate* will execute *rad* and create an options file to later pass to *rpict* or *rtrace*. Besides prepending the *render* variable, *ranimate* will also extract default settings for the common variables: *OCTREE*, *RESOLUTION*, *EXPOSURE* and *pfilt*. Following the file name, overriding variable settings may be given, which will be passed to *rad* on the command line. Settings with spaces in them should be enclosed in quotes. The execution of *rad* will also update the contents of the octree, if necessary. There is no default value for this variable.

DISKSPACE

Specify the amount of disk space (in megabytes) available on the destination file system for temporary file storage. *Ranimate* will coordinate its batch operations based on this amount of storage, assuming that there is either enough additional space for all the final frames, or that the given *TRANSFER* command will move the finished frames to some other location (see below). The default value is 100 megabytes.

ARCHIVE

After each batch rendering is finished and checked for completeness, *ranimate* will execute the given command, passing the names of all the original pictures and z-buffer files generated by *rpict.* Normally, the archive command copies the original files to a tape device or somewhere that they can be retrieved in the event of failure in the frame interpolation stages. After the archive command has successfully completed, the original renderings are removed. There is no default value for this variable, meaning that the original unfiltered frames will be simply be removed. Note that the last one or two rendered frames may not be copied, archived or removed in case there is a another sequence picking up where this one left off.

TRANSFER

The command to transfer the completed animation frames. The names of all the finished frames is appended to this command before it is executed. Normally, the transfer command does something such as convert the frames to another format and/or copy them to tape or some other destination device before removing them. If this variable is not given, the final frames are left where they are. (See *BASENAME,* above.)

RSH

The command to use instead of rsh(1) to execute commands remotely on another machine. The arguments and behavior of this program must be identical to the UNIX *rsh* command, except that the *-l* option will always be used to specify an alternate user name rather than the *user@host* convention. Th *-l* option may or may not appear, but the *-n* option will always be used, and the expected starting directory will be that of the remote user, just as with *rsh.*

NEXTANIM

This variable specifies the next *ranfile* to use after this sequence is completed. This offers a convenient means to continue an animation that requires different control options in different segments. It is important in this case to correctly set the *START* and *END* variables in each *ranfile* so that the segments do not overlap frames.

OVERSAMPLE

This variable sets the multiplier of the original image size relative to the final size given by the *RESOLUTION* variable. This determines the quality of anti-aliasing in the final frames. A value of 1 means no anti-aliasing, and a value of 3 produces very good anti-aliasing. The default value is 2. (A fractional value may be used for previews, causing low resolution frames with large, blocky pixels to be produced.)

INTERPOLATE

This variable sets the number of frames to interpolate between each rendered frame in a static scene walk-through. Z-buffers for each rendered frame will be generated by *rpict,* and *pinterp* will be called to perform the actual "tweening." This results in a potentially large savings in rendering time, but should be used with caution since certain information may be lost or inaccurate, such as specular highlights and reflections, and objects may even break apart if too few renderings are used to interpolate too much motion. The default value for this variable is 0, meaning no interpolation. Interpolation is also switched off if the *ANIMATE* variable is specified.

MBLUR

This variable specifies the number of additional samples to be taken at each final frame for motion blurring. An optional shutter speed may be given as a second argument, which indicates the fraction of the between-frame time during which the camera shutter is open. (The default value is 1, meaning the shutter is open the whole time.) The *pmblur*(*1*) command is used to generate the given number of additional views for *pinterp* to average together. The default value is 0, meaning no motion blurring. This option does not currently work with the *ANIMATE* variable, since pinterp only works for static environments.

RTRACE

This boolean variable tells *ranimate* whether or not to employ *rtrace* during frame interpolation using the *-fr* option to *pinterp*. If set to True, then the same rendering options and static octree are passed to *rtrace* as are normally used by *rpict*. The default value is False. Note that this variable only applies to static environment walkthroughs (i.e., no *ANIMATE* command).

RESOLUTION

This variable specifies the desired final picture resolution. If only a single number is given, this value will be used for both the horizontal and vertical picture dimensions. If two numbers are given, the first is the horizontal resolution and the second is the vertical resolution. If three numbers are given, the third is taken as the pixel aspect ratio for the final picture (a real value). If the pixel aspect ratio is zero, the exact dimensions given will be those produced. Otherwise, they will be used as a frame in which the final image must fit. The default value for this variable is 640.

render

This variable may be used to specify additional options to *rpict* or *rtrace*. These options will appear after the options set automatically by *rad*, and thus will override the default values.

pinterp

This variable may be used to specify additional options to *pinterp*, which is used to interpolate frames for a static scene walk-through. (See the *pinterp* man page, and the *INTERPOLATE* variable.) Do not use this variable to set the *pinterp -fr* option, but use the *RTRACE* setting instead.

pfilt

This variable may be used to specify additional options to *pfilt*. If this variable is given in the *ranfile*, then *pfilt* will always be used. (Normally, *pfilt* is called only if *pinterp* is not needed or automatic exposure is required.) See the *pfilt* manual page for details.

EXAMPLES

A minimal input file for *ranimate* might look like this:

```
# ::::::::::
# sample.ran
# ::::::::::
# The rad input file for our static scene:
RIF= tutor.rif
# The spool directory:
DIRECTORY= anim1
# The view file containing one view per frame:
VIEWFILE= anim1.vf
# The amount of temporary disk space available:
DISKSPACE= 50 # megabytes
```

Note that most of the variables are not set in this file. If we only want to see what default values *ranimate* would use without actually executing anything, we can invoke it thus:

```
ranimate -n -e sample.ran
```

This will print the variables we have given as well as default values *ranimate* has assigned for us. Also, we will see the list of commands that *ranimate* would have executed had the *-n* option not been present.

Usually, we execute ranimate in the background, redirecting the standard output and standard error to a file:

```
ranimate sample.ran >& sample.err &
```

If we decide that the default values ranimate has chosen for our variables are not all appropriate, we can add some more assignments to the file:

```
host= rays 3 ~greg/obj/tutor ray   # execute as
ray on multi-host "rays"
host= thishost # execute one copy on this host
also
INTERP= 3 # render every fourth frame
RES= 1024 # shoot for 1024x resolution
MBLUR= 5 .25   # apply camera motion blur
EXP= anim1.exp # adjust exposure according to file
pfilt= -r .9   # use Gaussian filtering
ARCHIVE= tar cf /dev/nrtape   # save original
renderings to tape
```

Note the use of abbreviation for variable names.

FILES

`$(DIRECTORY)/STATUS`	animation status file
`$(DIRECTORY)/*`	other temporary files
`$(BASENAME).pic`	final animation frames

AUTHOR

Greg Ward

BUGS

Due to the difficulty of controlling processes on multiple execution hosts, the *-n* option of *ranimate* is not useful in the same way as *rad* for generating a script of executable commands to render the sequence. It may give an idea of the sequence of events, but certain temporary files and so forth will not be in the correct state if the user attempts to create a separate batch script.

If multiple processors are available on a given host and the *RTRACE* variable is set to True, then the *-PP* option of *rtrace* should be employed, but it is not. There is no easy way around this problem, but it has only minor consequences in most cases. (The *-PP* option is used for *rpict*, however.)

The current implementation of the remote shell does not return the exit status of the remote process, which makes it difficult to determine for sure if there has been a serious error or not. Because of this, *ranimate* normally turns off warnings on all rendering processes, and takes any output to standard error from a remote command as a sign that a fatal error has occurred. (This also precludes the use of the *-t* option to report rendering progress.) If the error was caused by a process server going down, the server is removed from the active list and frame recovery takes place. Otherwise, *ranimate* quits at that point in the animation.

The current execution environment, in particular the RAYPATH variable, will not be passed during remote command execution, so it is necessary to set whatever variables are important in the remote startup script (e.g., ".cshrc" for the C-shell). This requirement may be circumvented by substituting the *on*(*1*) command for *rsh*(*1*) using the *RSH* control variable, or by writing a custom remote execution script.

If a different remote user name is used, ranimate first attempts to change to the original user's directory with a command of the form *cd ~uname* . This works under *csh*(*1*), but may fail under other shells such as *sh*(*1*).

If multiple hosts with different floating point formats are used, *pinterp* will fail because the Z-buffer files will be inconsistent. (Recall that *pinterp* is called if INTERPOLATE > 0 and/or MBLUR > 0.) Since most modern machines use IEEE floating point, this is not usually a problem, but it is something to keep in mind.

SEE ALSO

pfilt(1), pinterp(1), pmblur(1), rad(1), rpict(1), rsh(1), rtrace(1)

NAME

ra_bn - convert *Radiance* picture to/from Barneyscan image

SYNOPSIS

ra_bn [-g gamma][-e +/-stops] { input|- } [output] ra_bn -r [-g gamma][-e +/-stops] input [output]

DESCRIPTION

Ra_bn converts between *Radiance* and Barneyscan native RGB image files. Since Barneyscan images are stored in three files, one for each color component, only the root file name is given and the program appends the suffixes "red", "grn" and "blu". The *-g* option specifies the exponent used in gamma correction; the default value is 2.0. An exponent of 1.0 turns gamma correction off. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The *-r* option invokes a reverse conversion, from a Barneyscan image to a *Radiance* picture.

AUTHORS

Greg Ward

SEE ALSO

`pfilt(1)`, `ra_ppm(1)`, `ra_pr(1)`, `ra_pr24(1)`, `ra_t8(1)`, `ra_t16(1)`, `ra_tiff(1)`, `ximage(1)`

NAME

ra_gif - convert *Radiance* picture to Compuserve GIF

SYNOPSIS

ra_gif [-b][-d][-c ncolors][-g gamma][-e +/-stops][-n sampfac] input [output]

DESCRIPTION

Ra_gif converts from *Radiance* to Compuserve GIF colormapped, compressed image files. In the default mode, a *Radiance* picture is converted to a color-mapped GIF file of the same horizontal and vertical dimensions with 8-bits per pixel. The *-b* option converts the image to black and white. The *-d* option turns off dithering. The *-c* option allows fewer than 256 colors (and fewer than 8 bits per pixel). The *-g* option specifies the exponent used in gamma correction; the default value is 2.2. An exponent of 1.0 turns gamma correction off. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The *-n* option specifies a sampling factor for neural network color quantization. This value should be between 1 and 80, where 1 takes the longest and produces the best results in small areas of the image. If no value is given, a faster median cut algorithm is used. If the output file is missing, the standard output is used.

AUTHORS

Greg Ward
Paul Haeberli
David Rowley
Anthony Dekker provided the code for neural network color quantization

SEE ALSO

`pfilt(1)`, `ra_bn(1)`, `ra_ppm(1)`, `ra_pr(1)`, `ra_pr24(1)`, `ra_t8(1)`, `ra_t16(1)`, `ra_tiff(1)`, `ximage(1)`

NAME

ra_pict - convert *Radiance* pictures to Macintosh PICT files

SYNOPSIS

ra_pict [-e +/- *stops*] [-v] [-g *gamma*] [*infile* [*out_file*]]

DESCRIPTION

Ra_pict converts a *Radiance* picture, as produced by rpict (1) to a Macintosh PICT file. The picture will be a 24 bit PICT 2 picture, using a single bit map (DirectRect).

OPTIONS

-g *gamma*
: This sets an explict gamma correction for the image. If it is not specified, the default value of 2.0 is used.

-v
: Invokes verbose mode, and gives line on standard error giving the size of the picture and the gamma correction used.

-e +/- *stops*
: Adjusts the exposure by stops.

infile
: Specifies the file to read the picture from. If none is specified, it takes it from standard input. If standard input is used, then the picture is sent to standard output.

outfile
: Specifies the file to send the PICT file to. If none is specified, it is sent to standard output.

EXAMPLES

```
ra_pict mypict.pic mypict.pict
```

Will convert the *Radiance* picture mypict.pic, giving the Macintosh PICT mypict.pict.

```
ra_pict -g 2.2 mypict.pic mypict.pict
```

Will convert the file using a gamma of 2.2.

SEE ALSO

BUGS

Does not yet do Macintosh PICT to *Radiance* PIC, as this PICT files are a lot more complex than just one bit map. This is left as an exercise for the reader :-)

AUTHOR

Russell Street

NAME

ra_ppm - convert *Radiance* picture to/from a Poskanzer Portable Pixmap

SYNOPSIS

ra_ppm [-r][-a][-g gamma][-e +/-stops] [input [output]]

DESCRIPTION

Ra_ppm converts between *Radiance* and Poskanzer Portable Pixmap formats. The *-g* option specifies the exponent used in gamma correction; the default value is 2.2. An exponent of 1 turns gamma correction off. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The *-r* option invokes a reverse conversion, from a Pixmap to a *Radiance* picture. If the output file is missing, the standard output is used. If the input file is missing as well, the standard input is used.

The -a option produces a standard ASCII Pixmap representation instead of the default binary file. The file is much larger and the conversion is much slower, which is why this format is not normally used.

With the -r option, the type of the Pixmap input file is determined automatically. *Ra_ppm* will read either greyscale or color Pixmaps, but will only produce color Pixmaps on output.

NOTES

The Poskanzer Portable Bitmap Plus package contains translators between the Pixmap format and many of the dozen or so image file "standards" that exist. At the time of this writing, the software is free and available by anonymous ftp from export.lcs.mit.edu (18.30.0.238) in the file "contrib/pbmplus.tar.Z".

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

pfilt(1), ra_bn(1), ra_pr(1), ra_pr24(1), ra_t8(1), ra_t16(1), ra_tiff(1), ximage(1)

NAME

ra_pr - convert *Radiance* picture to/from pixrect rasterfile

SYNOPSIS

ra_pr [-d][-b][-c ncolors][-g gamma][-e +/-stops] input [output]
ra_pr -r [-g gamma][-e +/-stops] [input [output]]

DESCRIPTION

Ra_pr converts between *Radiance* and pixrect rasterfile formats. In the default mode, a *Radiance* picture is converted to a pixrect rasterfile of the same horizontal and vertical dimensions with 8-bits per pixel. The *-d* option turns off dithering. The *-b* option converts the image to black and white, for improved quality on greyscale monitors. Only with this option can the input be taken from stdin. The *-c* option allows fewer than 256 colors. The *-g* option specifies the exponent used in gamma correction; the default value is 2.2. An exponent of 1.0 turns gamma correction off. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The *-r* option invokes a reverse conversion, from a pixrect rasterfile to a *Radiance* picture. If the output file is missing, the standard output is used.

AUTHORS

Greg Ward
Paul Heckbert provided the code for color quantization

BUGS

Only standard 8-bit color rasterfiles are read or written.

SEE ALSO

`pfilt(1)`, `ra_bn(1)`, `ra_ppm(1)`, `ra_pr24(1)`, `ra_t8(1)`, `ra_t16(1)`, `ra_tiff(1)`, `ximage(1)`

NAME

ra_pr24 - convert *Radiance* picture to/from 24-bit rasterfile

SYNOPSIS

ra_pr24 [-r | -rgb][-g gamma][-e +/-stops] [input [output]]

DESCRIPTION

Ra_pr24 converts between *Radiance* and 24-bit pixrect rasterfile formats. The *-g* option specifies the exponent used in gamma correction; the default value is 2.2. An exponent of 1 turns gamma correction off. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The *-r* option invokes a reverse conversion, from a 24-bit rasterfile to a *Radiance* picture. If the *-rgb* option is used, the output rasterfile will be in RGB byte-ordering rather than the more standard (for Sun) BGR ordering. Byte ordering is determined automatically for the reverse conversion. If the output file is missing, the standard output is used. If the input file is missing as well, the standard input is used.

AUTHOR

Greg Ward

BUGS

Only standard 24-bit color rasterfiles are read or written.

SEE ALSO

`pfilt(1)`, `ra_bn(1)`, `ra_ppm(1)`, `ra_pr(1)`, `ra_t8(1)`, `ra_t16(1)`, `ra_tiff(1)`, `ximage(1)`

NAME

ra_ps - convert *Radiance* picture to a PostScript file

SYNOPSIS

ra_ps [-b|c][-A|B|C][-e +/-stops][-n ncopies] [input [output]]

DESCRIPTION

Ra_ps translates a *Radiance* picture to a greyscale or color Adobe PostScript file for printing on a laser printer or importing to a page layout program. The -*c* option tells *ra_ps* to produce color output. (The default is greyscale, which may be specified explicitly with the -*b* option.)

The -A option specifies that the output should be in uncompressed ASCII hexstring format (the default). The -*B* option specifies that the output should be in uncompressed binary string format. The file size will be roughly half that of the ASCII equivalent, but some printers and especially some printer connections do not support binary transfer, so this option should be used with caution. The -*C* option specifies that the output should be in run-length compressed ASCII format. The file size will be one half to one tenth as large as the hexstring equivalent and can be sent over any network or by e-mail. The only disadvantage is that it will actually take longer to print on some printers, since the "readhexstring" procedure is generally faster than a custom replacement.

The -e option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The -*n* option specifies the number of copies to print of this image. It is often preferable to use this option instead of the -*#N* option of *lpr*(*1*), since the latter often results in duplication of the input file with a large associated cost.

The output from ra_ps is designed to be compatible with the Encapsulated PostScript standard, which means that the resulting file may be incorporated into documents by page layout programs that can read in EPS files. Unfortunately, there is currently no option for generating a preview bitmap, so the image will show up on the screen as a rectangular area only.

WARNINGS

Ra_ps assumes an output page that is 8.5 inches (21.5 cm) wide by 11 inches (28 cm) tall. The image will be centered on such a page, with 0.5 inch minimum margins on each side. If the actual output page is larger, the image will be towards the lower left corner. If the output page is smaller, part of the image may be cut in the upper right corner. If the image fits better on its side (ie. landscape mode), it will be rotated automatically.

PostScript printers are notoriously slow at printing large images, and the halftoning methods used are often inadequate for producing nice grey levels.

AUTHOR

Greg Ward

SEE ALSO

`pfilt(1)`, `ra_bn(1)`, `ra_pr(1)`, `ra_pr24(1)`, `ra_t8(1)`, `ra_t16(1)`, `ra_ppm(1)`, `ra_tiff(1)`, `ximage(1)`

NAME

ra_rgbe - change run-length encoding of a *Radiance* picture

SYNOPSIS

ra_rgbe [-r][-e +/-stops] [input [output]]

DESCRIPTION

Ra_rgbe converts between *Radiance* run-length encoded and flat formats. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. By default, *ra_rgbe* produces a flat *Radiance* picture file from any type of *Radiance* input picture. The *-r* option causes *ra_rgbe* to produce a runlength encoded file instead. If the output file is missing, the standard output is used. If the input file is missing as well, the standard input is used.

NOTES

The file format for *Radiance* pictures was changed between release 1.4 and release 2.0. The older format can still be read by all the programs, but only the newer format is produced. This newer format cannot be read by *Radiance* software prior to release 2.0.

Ra_rgbe provides some downward compatibility by producing files that can be read by older *Radiance* software. The resultant files are also easier to manipulate with programs designed to read raw raster data.

The other use for ra_rgbe is as a quicker way to adjust the exposure of a *Radiance* picture than *pfilt*(*1*), since *ra_rgbe* only allows integer f-stop changes. In this mode, *ra_rgbe* should be used with the *-r* option.

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

`pfilt(1)`, `ra_xyze(1)`

NAME

ra_t16 - convert *Radiance* picture to/from Targa 16 or 24-bit image file

SYNOPSIS

ra_t16 [-2][-3][-g gamma][-e +/-stops] [input [output]]
ra_t16 -r [-g gamma][-e +/-stops] [input [output]]

DESCRIPTION

Ra_t16 converts between *Radiance* and Targa 16-bit or 24-bit RGB image files (type 2 in Targa's documentation). In the default mode, a *Radiance* picture is converted to an RGB file of the same horizontal and vertical dimensions with 16-bits per pixel. The *-3* option tells the program to produce a 24-bit image file instead. The *-g* option specifies the exponent used in gamma correction; the default value is 2.2. An exponent of 1.0 turns gamma correction off. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The *-r* option invokes a reverse conversion, from a 16-bit or 24-bit Targa file to a *Radiance* picture. The determination of depth is made automatically on reverse translation, so the *-2* and *-3* options are not necessary. If the output file is missing, the standard output is used.

AUTHORS

Greg Ward

BUGS

Run-length encoded files can be read but not written with this program.

SEE ALSO

`pfilt(1)`, `ra_bn(1)`, `ra_ppm(1)`, `ra_pr(1)`, `ra_pr24(1)`, `ra_t8(1)`, `ra_tiff(1)`, `ximage(1)`

NAME

ra_t8 - convert *Radiance* picture to/from Targa 8-bit image file

SYNOPSIS

ra_t8 [-d][-b][-c ncolors][-g gamma][-e +/-stops][-n sampfac] input [output]

ra_t8 -r [-g gamma][-e +/-stops] [input [output]]

DESCRIPTION

Ra_t8 converts between *Radiance* and Targa 8-bit color-mapped image files (type 1 in Targa's documentation). In the default mode, a *Radiance* picture is converted to a colormapped Targa file of the same horizontal and vertical dimensions with 8-bits per pixel. The *-d* option turns off dithering. The *-b* option converts the image to black and white. Only with this option can the input be taken from stdin. The *-c* option allows fewer than 256 colors. The *-g* option specifies the exponent used in gamma correction; the default value is 2.2. An exponent of 1.0 turns gamma correction off. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The *-n* option specifies a sampling factor for neural network color quantization. This value should be between 1 and 80, where 1 takes the longest and produces the best results in small areas of the image. If no value is given, a faster median cut algorithm is used. The *-r* option invokes a reverse conversion, from an 8-bit Targa file to a *Radiance* picture. If the output file is missing, the standard output is used.

AUTHORS

Greg Ward
Anthony Dekker provided the code for neural network color quantization

BUGS

Run-length encoded files can be read but not written with this program.

SEE ALSO

`pfilt(1)`, `ra_bn(1)`, `ra_ppm(1)`, `ra_pr(1)`, `ra_pr24(1)`, `ra_t16(1)`, `ra_tiff(1)`, `ximage(1)`

NAME

ra_tiff - convert *Radiance* picture to/from a TIFF color or greyscale image

SYNOPSIS

ra_tiff [-r][-b][-z][-g gamma][-e +/-stops] input output

DESCRIPTION

Ra_tiff converts between *Radiance* and TIFF image formats. The *-g* option specifies the exponent used in gamma correction; the default value is 2.2, which is the recommended value for TIFF images. The *-b* option can be used to specify an 8-bit greyscale TIFF output file. The type of input file is determined automatically. The *-z* option will result in LZW compression of the TIFF output file. Decompression is automatically determined for TIFF input. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. The *-r* option invokes a reverse conversion, from a TIFF image to a *Radiance* picture. The *Radiance* picture file can be taken from the standard input or sent to the standard output by using a hyphen ('-') in place of the file name, but the TIFF image must be to or from a file.

AUTHOR

Greg Ward
Sam Leffler really wrote this program because he wrote the TIFF image manipulation library that was used.

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

BUGS

Only TIFF 8-bit greyscale and 24-bit RGB color are supported.

SEE ALSO

`pfilt(1)`, `ra_bn(1)`, `ra_ppm(1)`, `ra_pr(1)`, `ra_pr24(1)`, `ra_t8(1)`, `ra_t16(1)`, `ximage(1)`

NAME

ra_xyze - convert between *Radiance* RGBE and XYZE formats

SYNOPSIS

ra_xyze [-r][-e exposure][-c | -u][-p xr yr xg yg xb yb xw yw] [input [output]]

DESCRIPTION

Ra_xyze converts between *Radiance* RGBE (red,green,blue,exponent) and XYZE (CIE X,Y,Z,exponent) formats. The *-e* option specifies an exposure compensation, which may be given as a decimal multiplier or in f-stops (powers of two). By default, *ra_xyze* produces a flat XYZE *Radiance* picture file from any type of *Radiance* input picture. To override these defaults, the *-c* option may be used to specify run-length encoded output, or the *-u* option may be used to specify a flat output.

The -r option causes ra_rgbe to produce a run-length encoded RGBE file instead. The *-p* option may be used to override the standard *Radiance* RGB primary colors to tailor the image for a particular output device or representation. The eight floating-point arguments to this option are the 1931 CIE (x,y) chromaticity coordinates of the three RGB primaries plus the white point, in that order. The new primaries will be recorded in the header of the output file, so that the original information may be fully recovered later. It is not necessary that the input file by in XYZE format. Th *-r* option may therefore be used to convert from one RGB primary representation to another using the *-p* option.

If the output file is missing, the standard output is used. If the input file is missing as well, the standard input is used.

NOTES

The CIE standard used is the 1931 2-degree observer, and the correct output representation relies on the original *Radiance* input description being defined properly in terms of the standard *Radiance* RGB primaries, whose CIE (x,y) chromaticity values are defined in the header file in src/common/color.h. In this same file is a standard for the luminous efficacy of white light (WHTEFFICACY), which is used as a conversion between lumens and watts throughout *Radiance*. This same factor is applied by *ra_xyze* when converting between the radiometric units of the RGBE format and the photometric units of the XYZE format. The purpose of this factor is to ensure that the Y component of the CIE representation is luminance in units of candelas/meter^2.

Although the white point calibration of monitors is often off-white, we recommend using equal-energy white as the primary value because a person viewing the monitor will automatically adjust his or her perception such that the display appears white. Using a measured white point, one may get the absolute CIE color up on the screen, but it will appear to be shifted due to the viewer's transient adaptation.

Most of the *Radiance* picture filters should work uniformly on either RGBE or XYZE files, so it is not necessary to convert back to RGBE format except for conversion or display, in which case the correct primaries for the chosen output device should be specified with the *-p* option if they are known.

EXAMPLES

To convert RGBE output from *rpict*(*1*) into run-length encoded XYZE format:

```
rpict [options] scene.oct | ra_xyze -c >
scene_xyz.pic
```

To prepare a *Radiance* picture for display on a calibrated NTSC monitor:

```
ra_xyze -r -p .670 .330 .210 .710 .140 .080 .333
.333 stand.pic ntsc.pic
```

AUTHOR

Greg Ward

BUGS

Any color correction applied to the original image is not removed or translated by *ra_xyze*, and it may result in color shifts in the output. If color preservation is important and the correction is unwanted, it is best to remove it with *pfilt*(*1*) using the *-er*, *-eg* and *-eb* options first. (Simply look at the header and apply the reciprocal primaries of all COLORCORR= lines multiplied together.) Better still, get the picture before color correction is applied.

SEE ALSO

`pfilt(1)`, `ra_rgbe(1)`, `rpict(1)`

NAME

rcalc - record calculator

SYNOPSIS

rcalc [-b][-l][-n][-w][-u][-tS][-i format][-o format][-f source][-e expr][-s svar=sval] file ..

DESCRIPTION

Rcalc transforms "records" from each *file* according to the given set of literal and relational information. By default, records are separated by newlines, and contain numeric fields separated by tabs. The *-tS* option is used to specify an alternate tab character. A *-i format* option specifies a template for an alternate input record format. *Format* is interpreted as a specification string if it contains a dollar sign '$'. Otherwise, it is interpreted as the name of the file containing the format specification. In either case, if the format does not end with a newline, one will be added automatically. A *-o format* option specifies an alternate output record format. It is interpreted the same as an input specification. The variable and function definitions in each *-f source* file are read and compiled. The *-e expr* option can be used to define variables on the command line. Since many of the characters in an expression have special meaning to the shell, it should usually be enclosed in single quotes. The *-s svar*=*sval* option can be used to assign a string variable a string value. The *-b* option instructs the program to accept only exact matches. By default, tabs and spaces are ignored except as field separators. The *-l* option instructs the program to ignore newlines in the input, basically treating them the same as tabs and spaces. Normally, the beginning of the input format matches the beginning of a line, and the end of the format matches the end of a line. With the *-l* option, the input format can match anywhere on a line. The *-w* option causes non-fatal error messages (such as division by zero) to be supressed. The *-u* option causes output to be flushed after each record. The *-n* option tells the program not to get any input, but to produce a single output record. Otherwise, if no files are given, the standard input is read.

Format files associate names with string and numeric fields separated by literal information in a record. A numeric field is given in a format file as a dollar sign, followed by curly braces enclosing a variable name:

```
This is a numeric field: ${vname}
```

A string variable is enclosed in parentheses:

```
This is a string field: $(sname)
```

The program attempts to match literal information in the input format to its input and assign string and numeric fields accordingly. If a string or numeric field variable appears more than once in the input format, input values for the corresponding fields must match (ie. have the same value) for the whole record to match. Numeric values are allowed some deviation, on the order of 0.1%, but string variables must match exactly. Thus, dummy variables for "don't care" fields should be given unique names so that they are not all required to take on the same value.

For each valid input record, an output record is produced in its corresponding format. Output field widths are given implicitly by the space occupied in the format file, including the dollar sign and braces. This makes it impossible to produce fields with fewer than four characters. If the *-b* option is specified, input records must exactly match the template. By default, the character following each input field is used as a delimiter. This implies that string fields that are followed by white space cannot contain strings with white space. Also, numeric fields followed but not preceded by white space will not accept numbers preceded by white space. Adjacent input fields are advisable only with the *-b* option. Numeric output fields may contain expressions as well as variables. A dollar sign may appear in a literal as two dollar signs ($$).

The definitions specified in -e and -f options relate numeric output fields to numeric input fields. For the default record format, a field is a variable of the form $N, where N is the column number, beginning with 1. Output columns appear on the left-hand side of assignments, input columns appear on the right-hand side.

A variable definition has the form:

```
     var = expression ;
```

Any instance of the variable in an expression will be replaced with its definition.

An expression contains real numbers, variable names, function calls, and the following operators:

```
     + - * / ^
```

Operators are evaluated left to right. Powers have the highest precedence; multiplication and division are evaluated before addition and subtraction. Expressions can be grouped with parentheses. All values are double precision real.

A function definition has the form:

```
     func(a1, a2, ..) = expression ;
```

The expression can contain instances of the function arguments as well as other variables and functions. Function names can be passed as arguments. Recursive functions can be defined using calls to the defined function or other functions calling the defined function.

The variable cond, if defined, will determine whether the current input record produces an output record. If *cond* is positive, output is produced. If *cond* is less than or equal to zero, the record is skipped and no other expressions are evaluated. This provides a convenient method for avoiding inappropriate calculations. The following library of predefined functions and variables is provided:

PI the ratio of a circle's circumference to its diameter.

if(cond, then, else)
: if cond is greater than zero, then is evaluated, otherwise else is evaluated. This function is necessary for recursive definitions.

select(N, a1, a2, ..)
: return aN (N is rounded to the nearest integer). This function provides array capabilities. If *N* is zero, the number of available arguments is returned.

rand(x)
: compute a random number between 0 and 1 based on x.

floor(x)
: return largest integer not greater than x.

ceil(x)
: return smallest integer not less than x.

sqrt(x)
: return square root of x.

exp(x)
: compute e to the power of x (e approx = 2.718281828).

log(x)
: compute the logarithm of x to the base e.

log10(x)
: compute the logarithm of x to the base 10.

sin(x), cos(x), tan(x)
: trigonometric functions.

asin(x), acos(x), atan(x)
: inverse trigonometric functions.

atan2(y, x)
: inverse tangent of y/x (range -pi to pi).

EXAMPLE

To print the square root of column two in column one, and column one times column three in column two:

```
rcalc -e '$1=sqrt($2);$2=$1*$3' inputfile >
outputfile
```

AUTHOR

Greg Ward

BUGS

String variables can only be used in input and output formats and -*s* options, not in definitions.

Tabs count as single spaces inside fields.

SEE ALSO

`calc(1)`, `cnt(1)`, `ev(1)`, `lam(1)`, `tabfunc(1)`, `total(1)`

NAME

replmarks - replace triangular markers in a *Radiance* scene description

SYNOPSIS

replmarks [-e][-m newmod][-s scale] { -x objfile | -i octree } modname .. [file ..]

DESCRIPTION

Replmarks replaces triangular markers identified by the modifier *modname* in each scene description *file* and writes the result to the standard output. The *-x* option indicates that each marker should be replaced by an appropriate *xform*(*1*) command on *objfile*. The *-i* option indicates that each marker should be replaced by an instance of *octree*. One of these two options must appear on the command line, along with *modname*, the modifier used by markers in the file.

Multiple modifiers may be given, as long as each one is preceded by its own *-x* or *-i* option.

The transformation for each marker is determined by its location and orientation. A marker should be a right triangle pointing like a half-arrow in the direction of the transformed x-axis, x'. The longest side is the hypoteneuse, the second longest side is the x'-axis, and the third longest side indicates the direction of the y'-axis. Any additional sides will be ignored (ie. a quadrilateral may be used instead of a triangle if the extra side is small). The z'-axis is determined by the cross product of the x' and y' axes, and the origin is the common vertex between x' and y'.

The size of the marker is ignored unless the -s option is used, where *scale* is a multiplier for the x'-axis length to indicate the total scale factor. For example, a *scale* value of 5 with a marker length of .5 would result in a total scale factor of 2.5 to be used in the transformation.

The -e option causes commands in the file to be expanded, and is required to replace markers from commands in the input file. Even with this option, *replmarks* will not examine objects for markers. Specifically, an object included by *replmarks* as a result of a *-x* expansion will be transferred verbatim, without regard to any surfaces therein that might have been considered as marks if they were on the main input.

The -m option causes all replaced objects to be given the modifier *newmod*. Otherwise, the new object surfaces will use their originally defined modifiers. A different replacement modifier may be given for each marker type. The marker modifier name itself is only used to identify markers, and will not appear in the output in any form.

If no input file is given, the standard input is read.

EXAMPLE

To replace all polygons with the modifier "knobs" in the file input with a transformed "knob.rad" and write the result to output:

```
replmarks -x knob.rad knobs input > output
```

To use instances of "tree.oct" with scaling set to three times the tree marker length:

```
replmarks -s 3 -i tree.oct tree input > output
```

AUTHOR

Greg Ward

SEE ALSO

`arch2rad(1)`, `ies2rad(1)`, `xform(1)`

NAME

rpict - generate a *Radiance* picture

SYNOPSIS

rpict [options] [$EVAR] [@file] [octree] rpict [options] -defaults

DESCRIPTION

Rpict generates a picture from the *Radiance* scene given in *octree* and sends it to the standard output. If no *octree* is given, the standard input is read. Options specify the viewing parameters as well as giving some control over the calculation. Options may be given on the command line and/or read from the environment and/or read from a file. A command argument beginning with a dollar sign ('$') is immediately replaced by the contents of the given environment variable. A command argument beginning with an at sign ('@') is immediately replaced by the contents of the given file.

In the second form shown above, the default values for the options (modified by those options present) are printed with a brief explanation.

Most options are followed by one or more arguments, which must be separated from the option and each other by white space. The exceptions to this rule are the *-vt* option and the boolean options. Normally, the appearance of a boolean option causes a feature to be "toggled", that is switched from off to on or on to off depending on its previous state. Boolean options may also be set explicitly by following them immediately with a '+' or '-', meaning on or off, respectively. Synonyms for '+' are any of the characters "yYtT1", and synonyms for '-' are any of the characters "nNfF0". All other characters will generate an error.

-vtt Set view type to t. If t is 'v', a perspective view is selected. If t is 'l', a parallel view is used. A cylindrical panorma may be selected by setting t to the letter 'c'. This view is like a standard perspective vertically, but projected on a cylinder horizontally (like a soupcan's-eye view). Two fisheye views are provided as well; 'h' yields a hemispherical fisheye view and 'a' results in angular fisheye distortion. A hemispherical fisheye is a projection of the hemisphere onto a circle. The maximum view angle for this type is 180 degrees. An angular fisheye view is defined such that distance from the center of the image is proportional to the angle from the central view direction. An angular fisheye can display a full 360 degrees. Note that there is no space between the view type option and its single letter argument.

-vp *x y z*

Set the view point to *x y z* . This is the focal point of a perspective view or the center of a parallel projection.

-vd *xd yd zd*

Set the view direction vector to *xd yd zd* .

-vu *xd yd zd*

Set the view up vector (vertical direction) to *xd yd zd* .

-vh *val*

Set the view horizontal size to *val*. For a perspective projection (including fisheye views), *val* is the horizontal field of view (in degrees). For a parallel projection, *val* is the view width in world coordinates.

-vv *val*

Set the view vertical size to *val*.

-vo *val*

Set the view fore clipping plane at a distance of *val* from the view point. The plane will be perpendicular to the view direction for perspective and parallel view types. For fisheye view types, the clipping plane is actually a clipping sphere, centered on the view point with radius *val*. Objects in front of this imaginary surface will not be visible. This may be useful for seeing through walls (to get a longer perspective from an exterior view point) or for incremental rendering. A value of zero implies no foreground clipping.

-va *val*

Set the view aft clipping plane at a distance of *val* from the view point. Like the view fore plane, it will be perpendicular to the view direction for perspective and parallel view types. For fisheye view types, the clipping plane is actually a clipping sphere, centered on the view point with radius *val*. Objects behind this imaginary surface will not be visible. A value of zero means no aft clipping, and is the only way to see infinitely distant objects such as the sky.

-vs *val*

Set the view shift to *val*. This is the amount the actual image will be shifted to the right of the specified view. This is option is useful for generating skewed perspectives or rendering an image a piece at a time. A value of 1 means that the rendered image starts just to the right of the normal view. A value of -1 would be to the left. Larger or fractional values are permitted as well.

-vl val

Set the view lift to val. This is the amount the actual image will be lifted up from the specified view, similar to the *-vs* option.

-vf *file*

Get view parameters from *file*, which may be a picture or a file created by rview (with the "view" command).

-x *res*

Set the maximum x resolution to *res*.

-y *res*

Set the maximum y resolution to *res*.

-pa *rat*

Set the pixel aspect ratio (height over width) to *rat*. Either the x or the y resolution will be reduced so that the pixels have this ratio for the specified view. If *rat* is zero, then the x and y resolutions will adhere to the given maxima.

-ps *size*

Set the pixel sample spacing to the integer *size*. This specifies the sample spacing (in pixels) for adaptive subdivision on the image plane.

-pt *frac*

Set the pixel sample tolerance to *frac*. If two samples differ by more than this amount, a third sample is taken between them.

-pj *frac*

Set the pixel sample jitter to *frac*. Distributed ray-tracing performs anti-aliasing by randomly sampling over pixels. A value of one will randomly distribute samples over full pixels. A value of zero samples pixel centers only. A value between zero and one is usually best for lowresolution images.

-dj *frac*

Set the direct jittering to *frac*. A value of zero samples each source at specific sample points (see the *-ds* option below), giving a smoother but somewhat less accurate rendering. A positive value causes rays to be distributed over each source sample according to its size, resulting in more accurate penumbras. This option should never be greater than 1, and may even cause problems (such as speckle) when the value is smaller. A warning about aiming failure will issued if *frac* is too large. It is usually wise to turn off image sampling when using direct jitter by setting -ps to 1.

-ds *frac*

Set the direct sampling ratio to *frac*. A light source will be subdivided until the width of each sample area divided by the distance to the illuminated point is below this ratio. This assures accuracy in regions close to large area sources at a slight computational expense. A value of zero turns source subdivision off, sending at most one shadow ray to each light source.

-dt *frac*

Set the direct threshold to *frac*. Shadow testing will stop when the potential contribution of at least the next and at most all remaining light source samples is less than this fraction of the accumulated value. (See the *-dc* option below.) The remaining light source contributions are approximated statistically. A value of zero means that all light source samples will be tested for shadow.

-dc *frac*

Set the direct certainty to *frac*. A value of one guarantees that the absolute accuracy of the direct calculation will be equal to or better than that given in the *-dt* specification. A value of zero only insures that all shadow lines resulting in a contrast change greater than the *-dt* specification will be calculated.

-dr *N*

Set the number of relays for secondary sources to *N*. A value of 0 means that secondary sources will be ignored. A value of 1 means that sources will be made into first generation secondary sources; a value of 2 means that first generation secondary sources will also be made into second generation secondary sources, and so on.

-dp *D*

Set the secondary source presampling density to D. This is the number of samples per steradian that will be used to determine ahead of time whether or not it is worth following shadow rays through all the reflections and/or transmissions associated with a secondary source path. A value of 0 means that the full secondary source path will always be tested for shadows if it is tested at all.

-dv Boolean switch for light source visibility. With this switch off, sources will be black when viewed directly although they will still participate in the direct calculation. This option may be desirable in conjunction with the *-i* option so that light sources do not appear in the output.

-sj *frac*

Set the specular sampling jitter to *frac*. This is the degree to which the highlights are sampled for rough specular materials. A value of one means that all highlights will be fully sampled using distributed ray tracing. A value of zero means that no jittering will take place, and all reflections will appear sharp even when they should be diffuse. This may be desirable when used in combination with image sampling (see *-ps* option above) to obtain faster renderings.

-st *frac*

Set the specular sampling threshold to *frac*. This is the minimum fraction of reflection or transmission, under which no specular sampling is performed. A value of zero means that highlights will always be sampled by tracing reflected or transmitted rays. A value of one means that specular sampling is never used. Highlights from light sources will always be correct, but reflections from other surfaces will be approximated using an ambient value. A sampling threshold between zero and one offers a compromise between image accuracy and rendering time.

-bv Boolean switch for back face visibility. With this switch off, back faces of opaque objects will be invisible to all rays. This is dangerous unless the model was constructed such that all surface normals on opaque objects face outward. Although turning off back face visibility does not save much computation time under most circumstances, it may be useful as a tool for scene debugging, or for seeing through one-sided walls from the outside. This option has no effect on transparent or translucent materials.

-av *red grn blu*

Set the ambient value to a radiance of *red grn blu* . This is the final value used in place of an indirect light calculation. If the number of ambient bounces is one or greater and the ambient value weight is non-zero (see *-aw* and *-ab* below), this value may be modified by the computed indirect values to improve overall accuracy.

-aw *N*

Set the relative weight of the ambient value given with the *-av* option to *N*. As new indirect irradiances are computed, they will modify the default ambient value in a moving average, with the specified weight assigned to the initial value given on the command and all other weights set to 1. If a value of 0 is given with this option, then the initial ambient value is never modified. This is the safest value for scenes with large differences in indirect contributions, such as when both indoor and outdoor (daylight) areas are visible.

-ab *N*

Set the number of ambient bounces to *N*. This is the maximum number of diffuse bounces computed by the indirect calculation. A value of zero implies no indirect calculation.

-ar *res*

Set the ambient resolution to *res*. This number will determine the maximum density of ambient values used in interpolation. Error will start to increase on surfaces spaced closer than the scene size divided by the ambient resolution. The maximum ambient value density is the scene size times the ambient accuracy (see the *-aa* option below) divided by the ambient resolution. The scene size can be determined using *getinfo*(*1*) with the *-d* option on the input octree. A value of zero is interpreted as unlimited resolution.

-aa *acc*

Set the ambient accuracy to *acc*. This value will approximately equal the error from indirect illuminance interpolation. A value of zero implies no interpolation.

-ad *N*

Set the number of ambient divisions to *N*. The error in the Monte Carlo calculation of indirect illuminance will be inversely proportional to the square root of this number. A value of zero implies no indirect calculation.

-as *N* Set the number of ambient super-samples to *N*. Super-samples are applied only to the ambient divisions which show a significant change.

-af *fname*

Set the ambient file to *fname*. This is where indirect illuminance will be stored and retrieved. Normally, indirect illuminance values are kept in memory and lost when the program finishes or dies. By using a file, different invocations can share illuminance values, saving time in the computation. Also, by creating an ambient file during a low resolution rendering, better results can be obtained in a second high resolution pass. The ambient file is in a machine-independent binary format which may be examined with *lookamb*(*1*). The ambient file may also be used as a means of communication and data sharing between simultaneously executing processes. The same file may be used by multiple processes, possibly running on different machines and accessing the file via the network (ie. *nfs*(*4*)). The network lock manager *lockd*(*8*) is used to insure that this information is used consistently. If any calculation parameters are changed or the scene is modified, the old ambient file should be removed so that the calculation can start over from scratch. For convenience, the original ambient parameters are listed in the header of the ambient file. *Getinfo*(*1*) may be used to print out this information.

-ae *mat*

Append *mat* to the ambient exclude list, so that it will not be considered during the indirect calculation. This is a hack for speeding the indirect computation by ignoring certain objects. Any object having *mat* as its modifier will get the default ambient level rather than a calculated value. Any number of excluded materials may be given, but each must appear in a separate option.

-ai *mat*

Add *mat* to the ambient include list, so that it will be considered during the indirect calculation. The program can use either an include list or an exclude list, but not both.

-aE *file*

Same as *-ae*, except read materials to be excluded from *file*. The RAYPATH environment variable determines which directories are searched for this file. The material names are separated by white space in the file.

-aI *file*

Same as *-ai*, except read materials to be included from *file*.

-me *rext gext bext*

Set the global medium extinction coefficient to the indicated color, in units of 1/distance (distance in world coordinates). Light will be scattered or absorbed over distance according to this value. The ratio of scattering to total scattering plus absorption is set by the albedo parameter, described below.

-ma *ralb galb balb*

Set the global medium albedo to the given value between 0 0 0 and 1 1 1. A zero value means that all light not transmitted by the medium is absorbed. A unitary value means that all light not transmitted by the medium is scattered in some new direction. The isotropy of scattering is determined by the Heyney-Greenstein parameter, described below.

-mg *gecc*

Set the medium Heyney-Greenstein eccentricity parameter to *gecc*. This parameter determines how strongly scattering favors the forward direction. A value of 0 indicates perfectly isotropic scattering. As this parameter approaches 1, scattering tends to prefer the forward direction.

-ms *sampdist*

Set the medium sampling distance to *sampdist*, in world coordinate units. During source scattering, this will be the average distance between adjacent samples. A value of 0 means that only one sample will be taken per light source within a given scattering volume.

-i Boolean switch to compute irradiance rather than radiance values. This only affects the final result, substituting a Lambertian surface and multiplying the radiance by pi. Glass and other transparent surfaces are ignored during this stage. Light sources still appear with their original radiance values, though the *-dv* option (above) may be used to override this.

-lr *N*

Limit reflections to a maximum of *N*.

-lw *frac*

Limit the weight of each ray to a minimum of *frac*. During ray-tracing, a record is kept of the final contribution a ray would have to the image. If it is less then the specified minimum, the ray is not traced.

-S *seqstart*

Instead of generating a single picture based only on the view parameters given on the command line, this option causes *rpict* to read view options from the standard input and for each line containing a valid view specification, generate a corresponding picture. This option is most useful for generating animated sequences, though it may also be used to control rpict from a remote process for network-distributed rendering. *Seqstart* is a positive integer that will be associated with the first output frame, and incremented for successive output frames. By default, each frame is concatenated to the output stream, but it is possible to change this action using the *-o* option (described below). Note that the octree may not be read from the standard input when using this option.

-o *fspec*

Send the picture(s) to the file(s) given by *fspec* instead of the standard output. If this option is used in combination with *-S* and *fspec* contains an integer field for *printf*(*3*) (eg. "%03d") then the actual output file name will include the current frame number. *Rpict* will not allow a picture file to be clobbered (overwritten) with this option. If an image in a sequence already exists (*-S* option), *rpict* will skip until it reaches an image that doesn't, or the end of the sequence. This is useful for running rpict on multiple machines or processors to render the same sequence, as each process will skip to the next frame that needs rendering.

-r *fn*

Recover pixel information from the file *fn*. If the program gets killed during picture generation, the information may be recovered using this option. The view parameters and picture dimensions are also recovered from *fn* if possible. The other options should be identical to those which created *fn*, or an inconsistent picture may result. If *fn* is identical to the file specification given with the *-o* option, *rpict* will rename the file prior to copying its contents. This insures that the old file is not overwritten accidentally. (See also the *-ro* option, below.) If *fn* is an integer and the recover option is used in combination with the *-S* option, then *rpict* skips a number of view specifications on its input equal to the difference between *fn* and *seqstart*. *Rpict* then performs a recovery operation on the file constructed from the frame number *fn* and the output file specification given with the *-o* option. This provides a convenient mechanism for recovering in the middle of an aborted picture sequence. The recovered file will be removed if the operation is successful. If the recover operation fails (due to lack of disk space) and the output file and recover file specifications are the same, then the original information may be left in a renamed temporary file. (See FILES section, below.)

-ro *fspec*

This option causes pixel information to be recovered from and subsequently returned to the picture file *fspec*. The effect is the same as specifying identical recover and output file names with the *-r* and *-o* options.

-z *fspec*

Write pixel distances out to the file *fspec*. The values are written as short floats, one per pixel in scanline order, as required by *pinterp*(*1*). Similar to the *-o* option, the actual file name will be constructed using *printf*(*3*) and the frame number from the *-S* option. If used with the *-r* option, *-z* also recovers information from an aborted rendering.

-P *pfile*

Execute in a persistent mode, using *pfile* as the control file. This option must be used together with *-S*, and is incompatible with the recover option (*-r*). Persistent execution means that after reaching end-of-file on its input, *rpict* will fork a child process that will wait for another *rpict* command with the same *-P* option to attach to it. (Note that since the rest of the command line options will be those of the original invocation, it is not necessary to give any arguments besides *-P* for subsequent calls.) Killing the process is achieved with the *kill*(*1*) command. (The process ID in the first line of *pfile* may be used to identify the waiting *rpict* process.) This option may be less useful than the *-PP* variation, explained below.

-PP *pfile*

Execute in continuous-forking persistent mode, using *pfile* as the control file. The difference between this option and the *-P* option described above is the creation of multiple duplicate processes to handle any number of attaches. This provides a simple and reliable mechanism of memory sharing on most multiprocessing platforms, since the *fork*(*2*) system call will share memory on a copy-on-write basis. This option may be used with *rpiece*(*1*) to efficiently render a single image using multiple processors on the same host.

-t *sec*

Set the time between progress reports to *sec*. A progress report writes the number of rays traced, the percentage completed, and the CPU usage to the standard error. Reports are given either automatically after the specified interval, or when the process receives a continue (-CONT) signal (see *kill*(*1*)). A value of zero turns automatic reporting off.

-e *efile*

Send error messages and progress reports to *efile* instead of the standard error.

-w Boolean switch for warning messages. The default is to print warnings, so the first appearance of this option turns them off.

EXAMPLES

```
rpict -vp 10 5 3 -vd 1 -.5 0 scene.oct > scene.pic
```

```
rpict -S 1 -o frame%02d.pic scene.oct <
keyframes.vf
```

ENVIRONMENT

`RAYPATH` the directories to check for auxiliary files

FILES

`/usr/tmp/rtXXXXXX` common header information for picture sequence
`rfXXXXXX` temporary name for recover file

DIAGNOSTICS

If the program terminates from an input related error, the exit status will be 1. A system related error results in an exit status of 2. If the program receives a signal that is caught, it will exit with a status of 3. In each case, an error message will be printed to the standard error, or to the file designated by the *-e* option.

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `lookamb(1)`, `oconv(1)`, `pfilt(1)`, `pinterp(1)`, `rad(1)`, `rtrace(1)`, `rview(1)`

NAME

rpiece - render pieces of a *Radiance* picture

SYNOPSIS

rpiece [-v][-x xres][-y yres][-X xdiv][-Y ydiv][-F|R syncfile][-T timelim] [$EVAR] [@file] [rpict options] -o picture octree

DESCRIPTION

Rpiece renders a *Radiance* picture a piece at a time, calling *rpict*(*1*) to do the actual work. This is useful for running multiple *rpict* processes on cooperating machines to render a single picture, which is a shared file specified with the -*o* option. The overall picture dimensions will be *xres* by *yres* (or smaller, depending on the -*pa* option and other view options), and the picture will be rendered in *xdiv* by *ydiv* pieces.

There are two basic methods for telling rpiece which piece(s) of a picture to render. The explicit method is to write on the standard input the *X* and *Y* position of the desired piece(s), where *X* runs from zero to *xdiv-1* and *Y* runs from zero to *ydiv-1*. (The lower left piece of a picture corresponds to (0,0) in this system.) Alternatively, the implicit specification method uses a synchronization file to determine which piece is to be rendered next. Specified with the -*F* option, *syncfile* initially contains the values for *xdiv* and *ydiv*, so the -*X* and -*Y* options are unnecessary. (However, they are used if *syncfile* does not exist.) The first *rpiece* process puts a lock on *syncfile* and modifies its contents before starting work on the first piece of the image. It writes the *X* and *Y* position of the piece it will work on, so the next *rpiece* process to modify *syncfile* will start on the next piece. (When it finishes with its piece, it appends the index to the end of *syncfile*.) This procedure continues until all the pieces are done, at which point all of the *rpiece* processes will terminate.

The -R option may be used instead of -F if some of the pieces were not properly finished by previous (killed) runs of *rpiece*. This option should be used by at most one *rpiece* process, since multiple recover processes would likely rerender the same pieces.

The -v flag switches on verbose mode, where rpiece reports to the standard output after each piece begins and after each piece is finished.

Options may be given on the command line and/or read from the environment and/or read from a file. A command argument beginning with a dollar sign ('$') is immediately replaced by the contents of the given environment variable. A command argument beginning with an at sign ('@') is immediately replaced by the contents of the given file.

EXAMPLE

First *rpiece* process is started on the machine "goober":

```
     goober% echo 1 8 > syncfile
     goober% echo -F syncfile -x 1024 -y 1024 -vf view
     -o picture octree > args
     goober% rpiece @args &
```

Second rpiece processes is started on the machine "sucker":

```
     sucker% rpiece @args &
```

NOTES

Due to NFS file buffering, the network lock manager is employed to guarantee consistency in the output file even though non-overlapping writes are used. This would tend to slow the process down if *rpiece* were to wait for this I/O to complete before starting on the next piece, so *rpiece* forks separate processes to hang around waiting for I/O completion. The number of processes thus designated is set by the MAXFORK macro in the program (compiled in the src/util directory). If the fork call is slow on a system, it may actually be better to set MAXFORK to zero. In other cases, the network lock manager may be so slow that this value should be increased to get the best utilization.

The output picture is not run-length encoded, and can be quite large. The approximate size (in kilobytes) can be computed by the simple formula:

```
     filesize = xres*yres/256
```

Make sure that there is enough space on the filesystem to hold the entire picture before beginning. Once the picture is finished, the *ra_rgbe*(*1*) program with the -r option may be used to convert to a run-length encoded picture for more efficient storage, although *pfilt*(*1*) or any of the other *Radiance* picture filters will do the same thing.

The ALRM signal may be used to gracefully terminate an *rpiece* process after it finishes the current piece. This permits other currently running or subsequently started *rpiece* process(es) to continue rendering the picture without loss. The -*T* option will send the ALRM signal to *rpiece* after the specified number of (decimal) hours. This is the best way to force a time limit on the computation, since information will not be lost, though the process may continue for some time afterwards to finish its current piece.

BUGS

This program may not work on some systems whose NFS lock manager is unreliable. In particular, some System V derivative UNIX systems often have problems with the network lock manager. If the output is scrambled or rpict aborts with some ambient file related problem, you should just remove the ambient file and go back to normal rendering.

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `pfilt(1)`, `ra_rgbe(1)`, `rpict(1)`, `ximage(1)`

NAME

rtrace - trace rays in *Radiance* scene

SYNOPSIS

rtrace [options] [$EVAR] [@file] octree rtrace [options] -defaults

DESCRIPTION

Rtrace traces rays from the standard input through the *Radiance* scene given by *octree* and sends the results to the standard output. Input for each ray is:

xorg yorg zorg xdir ydir zdir

If the direction vector is (0,0,0), then the output is flushed. This may be useful for programs that run *rtrace* as a separate process. In the second form, the default values for the options (modified by those options present) are printed with a brief explanation.

Options may be given on the command line and/or read from the environment and/or read from a file. A command argument beginning with a dollar sign ('$') is immediately replaced by the contents of the given environment variable. A command argument beginning with an at sign ('@') is immediately replaced by the contents of the given file. Most options are followed by one or more arguments, which must be separated from the option and each other by white space. The exceptions to this rule are the boolean options. Normally, the appearance of a boolean option causes a feature to be "toggled", that is switched from off to on or on to off depending on its previous state. Boolean options may also be set explicitly by following them immediately with a '+' or '-', meaning on or off, respectively. Synonyms for '+' are any of the characters "yYtT1", and synonyms for '-' are any of the characters "nNfF0". All other characters will generate an error.

-f*io* Format input according to the character *i* and output according to the character *o*. *Rtrace* understands the following input and output formats: 'a' for ascii, 'f' for single-precision floating point, and 'd' for double-precision floating point. In addition to these three choices, the character 'c' may be used to denote 4-byte floating point (*Radiance*) color format for the output of values only (-*ov* option, below). If the output character is missing, the input format is used. Note that there is no space between this option and its argument.

-o*spec* Produce output fields according to *spec*. Characters are interpreted as follows:

o origin (input)

d	direction (normalized)
v	value (radiance)
w	weight
l	effective length of ray
L	first intersection distance
p	point of intersection
n	normal at intersection (perturbed)
N	normal at intersection (unperturbed)
s	surface name
m	modifier name

If the letter 't' appears in *spec*, then the fields following will be printed for every ray traced, not just the final result. Spawned rays are indented one tab for each level. Note that there is no space between this option and its argument.

-te mat

Append mat to the trace exclude list, so that it will not be reported by the trace option (-o*t*). Any ray striking an object having *mat* as its modifier will not be reported to the standard output with the rest of the rays being traced. This option has no effect unless the 't' option has been given as part of the output specifier. Any number of excluded materials may be given, but each must appear in a separate option.

-ti *mat*

Add *mat* to the trace include list, so that it will be considered during the indirect calculation. The program can use either an include list or an exclude list, but not both.

-tE *file*

Same as *-te*, except read materials to be excluded from *file*. The RAYPATH environment variable determines which directories are searched for this file. The material names are separated by white space in the file.

-tI *file*

Same as *-ti*, except read materials to be included from *file*.

-i Boolean switch to compute irradiance rather than radiance values. This only affects the final result, substituting a Lambertian surface and multiplying the radiance by pi. Glass and other transparent surfaces are ignored during this stage. Light sources still appear with their original radiance values, though the *-dv* option (below) may be used to override this. This option is especially useful in conjunction with `ximage(1)` for computing illuminance at scene points.

-I Boolean switch to compute irradiance rather than radiance, with the input origin and direction interpreted instead as measurement point and orientation.

-h Boolean switch for information header on output.

-x *res*

Set the x resolution to *res*. The output will be flushed after every *res* input rays. A value of zero means that no output flushing will take place.

-y *res*

Set the y resolution to *res*. The program will exit after *res* scanlines have been processed, where a scanline is the number of rays given by the *-x* option, or 1 if *-x* is zero. A value of zero means the program will not halt until the end of file is reached. If both *-x* and *-y* options are given, a resolution string is printed at the beginning of the output. This is mostly useful for recovering image dimensions with *pvalue*(*1*), and for creating valid *Radiance* picture files using the color output format. (See the *-f* option, above.)

-dj frac

Set the direct jittering to frac. A value of zero samples each source at specific sample points (see the *-ds* option below), giving a smoother but somewhat less accurate rendering. A positive value causes rays to be distributed over each source sample according to its size, resulting in more accurate penumbras. This option should never be greater than 1, and may even cause problems (such as speckle) when the value is smaller. A warning about aiming failure will issued if *frac* is too large.

-ds *frac*

Set the direct sampling ratio to *frac*. A light source will be subdivided until the width of each sample area divided by the distance to the illuminated point is below this ratio. This assures accuracy in regions close to large area sources at a slight computational expense. A value of zero turns source subdivision off, sending at most one shadow ray to each light source.

-dt *frac*

Set the direct threshold to *frac*. Shadow testing will stop when the potential contribution of at least the next and at most all remaining light sources is less than this fraction of the accumulated value. (See the *-dc* option below.) The remaining light source contributions are approximated statistically. A value of zero means that all light sources will be tested for shadow.

-dc *frac*

Set the direct certainty to *frac*. A value of one guarantees that the absolute accuracy of the direct calculation will be equal to or better than that given in the *-dt* specification. A value of zero only insures that all shadow lines resulting in a contrast change greater than the *-dt* specification will be calculated.

-dr *N*

Set the number of relays for secondary sources to *N*. A value of 0 means that secondary sources will be ignored. A value of 1 means that sources will be made into first generation secondary sources; a value of 2 means that first generation secondary sources will also be made into second generation secondary sources, and so on.

-dp *D*

Set the secondary source presampling density to D. This is the number of samples per steradian that will be used to determine ahead of time whether or not it is worth following shadow rays through all the reflections and/or transmissions associated with a secondary source path. A value of 0 means that the full secondary source path will always be tested for shadows if it is tested at all.

-dv

Boolean switch for light source visibility. With this switch off, sources will be black when viewed directly although they will still participate in the direct calculation. This option is mostly for the program *mkillum*(*1*) to avoid inappropriate counting of light sources, but it may also be desirable in conjunction with the *-i* option.

-sj *frac*

Set the specular sampling jitter to *frac*. This is the degree to which the highlights are sampled for rough specular materials. A value of one means that all highlights will be fully sampled using distributed ray tracing. A value of zero means that no jittering will take place, and all reflections will appear sharp even when they should be diffuse.

-st *frac*

Set the specular sampling threshold to *frac*. This is the minimum fraction of reflection or transmission, under which no specular sampling is performed. A value of zero means that highlights will always be sampled by tracing reflected or transmitted rays. A value of one means that specular sampling is never used. Highlights from light sources will always be correct, but reflections from other surfaces will be approximated using an ambient value. A sampling threshold between zero and one offers a compromise between image accuracy and rendering time.

-bv Boolean switch for back face visibility. With this switch off, back faces of opaque objects will be invisible to all rays. This is dangerous unless the model was constructed such that all surface normals on opaque objects face outward. Although turning off back face visibility does not save much computation time under most circumstances, it may be useful as a tool for scene debugging, or for seeing through one-sided walls from the outside. This option has no effect on transparent or translucent materials.

-av *red grn blu*

Set the ambient value to a radiance of *red grn blu* . This is the final value used in place of an indirect light calculation. If the number of ambient bounces is one or greater and the ambient value weight is non-zero (see -*aw* and -*ab* below), this value may be modified by the computed indirect values to improve overall accuracy.

-aw *N*

Set the relative weight of the ambient value given with the *-av* option to *N*. As new indirect irradiances are computed, they will modify the default ambient value in a moving average, with the specified weight assigned to the initial value given on the command and all other weights set to 1. If a value of 0 is given with this option, then the initial ambient value is never modified. This is the safest value for scenes with large differences in indirect contributions, such as when both indoor and outdoor (daylight) areas are visible.

-ab *N*

Set the number of ambient bounces to *N*. This is the maximum number of diffuse bounces computed by the indirect calculation. A value of zero implies no indirect calculation.

-ar *res*

Set the ambient resolution to *res*. This number will determine the maximum density of ambient values used in interpolation. Error will start to increase on surfaces spaced closer than the scene size divided by the ambient resolution. The maximum ambient value density is the scene size times the ambient accuracy (see the *-aa* option below) divided by the ambient resolution. The scene size can be determined using *getinfo*(*1*) with the *-d* option on the input octree.

-aa *acc*

Set the ambient accuracy to *acc*. This value will approximately equal the error from indirect illuminance interpolation. A value of zero implies no interpolation.

-ad *N*

Set the number of ambient divisions to *N*. The error in the Monte Carlo calculation of indirect illuminance will be inversely proportional to the square root of this number. A value of zero implies no indirect calculation.

-as *N*

Set the number of ambient super-samples to *N*. Super-samples are applied only to the ambient divisions which show a significant change.

-af *fname*

Set the ambient   file to *fname*. This is where indirect illuminance will be stored and retrieved. Normally, indirect illuminance values are kept in memory and lost when the program finishes or dies. By using a file, different invocations can share illuminance values, saving time in the computation. The ambient file is in a machineindependent binary format which can be examined with *lookamb*(*1*). The ambient file may also be used as a means of communication and data sharing between simultaneously executing processes. The same file may be used by multiple processes, possibly running on different machines and accessing the file via the network (ie. *nfs*(*4*)). The network lock manager *lockd*(*8*) is used to insure that this information is used consistently. If any calculation parameters are changed or the scene is modified, the old ambient file should be removed so that the calculation can start over from scratch. For convenience, the original ambient parameters are listed in the header of the ambient file. *Getinfo*(*1*) may be used to print out this information.

-ae mat

Append mat to the ambient exclude list, so that it will not be considered during the indirect calculation. This is a hack for speeding the indirect computation by ignoring certain objects. Any object having *mat* as its modifier will get the default ambient level rather than a calculated value. Any number of excluded materials may be given, but each must appear in a separate option.

-ai *mat*

Add *mat* to the ambient include list, so that it will be considered during the indirect calculation. The program can use either an include list or an exclude list, but not both.

-aE *file*

Same as *-ae*, except read materials to be excluded from *file*. The RAYPATH environment variable determines which directories are searched for this file. The material names are separated by white space in the file.

-aI *file*

Same as *-ai*, except read materials to be included from *file*.

-me *rext gext bext*

Set the global medium extinction coefficient to the indicated color, in units of 1/distance (distance in world coordinates). Light will be scattered or absorbed over distance according to this value. The ratio of scattering to total scattering plus absorption is set by the albedo parameter, described below.

-ma *ralb galb balb*

Set the global medium albedo to the given value between 0 0 0 and 1 1 1. A zero value means that all light not transmitted by the medium is absorbed. A unitary value means that all light not transmitted by the medium is scattered in some new direction. The isotropy of scattering is determined by the Heyney-Greenstein parameter, described below.

-mg *gecc*

Set the medium Heyney-Greenstein eccentricity parameter to *gecc*. This parameter determines how strongly scattering favors the forward direction. A value of 0 indicates perfectly isotropic scattering. As this parameter approaches 1, scattering tends to prefer the forward direction.

-ms *sampdist*

Set the medium sampling distance to *sampdist*, in world coordinate units. During source scattering, this will be the average distance between adjacent samples. A value of 0 means that only one sample will be taken per light source within a given scattering volume.

-lr *N*

Limit reflections to a maximum of *N*.

-lw *frac*

Limit the weight of each ray to a minimum of *frac*. During ray-tracing, a record is kept of the final contribution a ray would have to the image. If it is less then the specified minimum, the ray is not traced.

-e *efile*

Send error messages and progress reports to *efile* instead of the standard error.

-w Suppress warning messages.

-P *pfile*

Execute in a persistent mode, using *pfile* as the control file. Persistent execution means that after reaching end-of-file on its input, *rtrace* will fork a child process that will wait for another *rtrace* command with the same *-P* option to attach to it. (Note that since the rest of the command line options will be those of the original invocation, it is not necessary to give any arguments besides *-P* for subsequent calls.) Killing the process is achieved with the *kill*(*1*) command. (The process ID in the first line of *pfile* may be used to identify the waiting *rtrace* process.) This option may be used with the *-fr* option of *pin_terp*(*1*) to avoid the cost of starting up *rtrace* many times.

-PP *pfile*

Execute in continuous-forking persistent mode, using *pfile* as the control file. The difference between this option and the *-P* option described above is the creation of multiple duplicate processes to handle any number of attaches. This provides a simple and reliable mechanism of memory sharing on most multiprocessing platforms, since the *fork*(*2*) system call will share memory on a copy-on-write basis.

EXAMPLES

To compute radiance values for the rays listed in samples.inp:

```
rtrace -ov scene.oct < samples.inp > radiance.out
```

To compute illuminance values at locations selected with the 't' command of *ximage*(*1*):

```
ximage scene.pic | rtrace -h -x 1 -i scene.oct |
rcalc -e '$1=47.4*$1+120*$2+11.6*$3'
```

To compute an image with an unusual view mapping:

```
cnt 640 480 | rcalc -e 'xr:640;yr:480' -f
unusual_view.cal | rtrace -x 640 -y 480 -fac
scene.oct > unusual.pic
```

ENVIRONMENT

`RAYPATH` the directories to check for auxiliary files.

FILES

`/usr/tmp/rtXXXXXX` common header information for picture sequence

DIAGNOSTICS

If the program terminates from an input related error, the exit status will be 1. A system related error results in an exit status of 2. If the program receives a signal that is caught, it will exit with a status of 3. In each case, an error message will be printed to the standard error, or to the file designated by the -*e* option.

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `lookamb(1)`, `oconv(1)`, `pfilt(1)`, `pinterp(1)`, `pvalue(1)`, `rpict(1)`, `rview(1)`, `ximage(1)`

NAME

rview - generate *Radiance* images interactively

SYNOPSIS

rview [rpict options][-o dev][-b][-pe exposure] [$EVAR] [@file] octree

rview [options] -defaults

rview -devices

DESCRIPTION

Rview generates *Radiance* images using *octree*. Options specify the viewing parameters as well as giving some control over the calculation. Options may be given on the command line and/or read from the environment and/or read from a file. A command argument beginning with a dollar sign ('$') is immediately replaced by the contents of the given environment variable. A command argument beginning with an at sign ('@') is immediately replaced by the contents of the given file. The options are the same as for `rpict(1)`, with a few notable exceptions. The *-r, -z, -S, -P, -PP* and *-t* options are not supported, and *-o* specifies which output device is being used instead of the output file. The *-x, -y* and *-pa* options are unnecessary, since *rview* scales the display image to the specified output device. Additionally, the *-b* option improves the display on greyscale monitors, and *-pe* may be used to set an initial exposure value.

In the second form, the default values for the options are printed with a brief explanation. In the third form, the list of supported output devices is displayed.

Rview starts rendering the image from the selected viewpoint and gradually improves the resolution of the display until interrupted by keyboard input. *Rview* then issues a prompt (usually ':') and accepts a command line from the user. *Rview* may also stop its calculation and wait for command input if the resolution of the display has reached the resolution of the graphics device. At this point, it will give the 'done:' prompt and await further instructions. If *rview* runs out of memory due to lack of resources to store its computed image, it will give the 'out of memory:' prompt. At this prompt, the user can save the image, quit, or even restart a new image, although this is not generally recommended on virtual memory machines for efficiency reasons.

Rview is not meant to be a rendering program, and we strongly recommend that *rpict*(*1*) be used instead for that purpose. Since *rpict*(*1*) does not store its image in memory or update any display of its output, it is much faster and less wasteful of its resources than *rview*. *Rview* is intended as a quick interactive program for deciding viewpoints and debugging scene descriptions and is not suited for producing polished images.

COMMANDS

Once the program starts, a number of commands can be used to control it. A command is given by its name, which can be abbreviated, followed by its arguments.

aim [*mag* [*x y z*]]

Zoom in by *mag* on point *x y z* . The view point is held constant; only the view direction and size are changed. If *x y z* is missing, the cursor is used to select the view center. A negative magnification factor means zoom out. The default factor is one.

^C Interrupt. Go to the command line.

exposure [*spec*]

Adjust exposure. The number *spec* is a multiplier used to compensate the average exposure. A value of 1 renormalizes the image to the computed average, which is usually done immediately after startup. If *spec* begins with a '+' or '-', the compensation is interpreted in f-stops (ie. the power of two). If *spec* begins with an '=', an absolute setting is performed. An '=' by itself permits interactive display and setting of the exposure. If *spec* begins with an '@', the exposure is adjusted to present similar visibility to what would be experienced in the real environment. If *spec* is absent, or an '@' is followed by nothing, then the cursor is used to pick a specific image location for normalization.

frame [*xmin ymin xmax ymax*]

Set frame for refinement. If coordinates are absent, the cursor is used to pick frame boundaries. If "all" is specified, the frame is reset to the entire image.

free Free cached object structures and associated data. This command may be useful when memory is low and a completely different view is being generated from the one previous.

last [*file*]

Restore the previous view. If a view or picture *file* is specified, the parameters are taken from the last view entry in the file.

L [*vw* [*rfile*]]

Load parameters for view *vw* from the *rad*(*1*) input file, *rfile*. Both *vw* and *rfile* must be given the first call, but subsequent calls will use the last *rfile* as a default, and "1" as the default view (ie. the first view appearing in *rfile*). If *rview* was started by *rad*, then the *rfile* parameter will initially default to the rad input file used.

move [*mag* [*x y z*]]

Move camera *mag* times closer to point *x y z* . For a perspective projection (or fisheye view), only the view point is changed; the view direction and size remain constant. The view size must be modified in a parallel projection since it determines magnification. If *x y z* is missing, the cursor is used to select the view center. A negative magnification factor decreases the object size. The default factor is one. Care must be taken to avoid moving behind or inside other objects.

new Restart the image. Usually used after the "set" command.

pivot *angle* [*elev* [*mag* [*x y z*]]]

Similar to the "move" command, but pivots the view about a selected point. The *angle* is measured in degrees around the view up vector using the right hand rule. The optional *elev* is the elevation in degrees from the pivot point; positive raises the view point to look downward and negative lowers the view point to look upward.

quit Quit the program.

^R Redraw the image. Use when the display gets corrupted. On some displays, occassionally forcing a redraw can improve appearance, as more color information is available and the driver can make a better color table selection.

rotate *angle* [*elev* [*mag*]]

Rotate the camera horizontally by *angle* degrees. If an elevation is specified, the camera looks upward *elev* degrees. (Negative means look downward.)

set [*var* [*val*]]
: Check/change program variable. If *var* is absent, the list of available variables is displayed. If *val* is absent, the current value of the variable is displayed and changed interactively. Otherwise, the variable *var* assumes the value *val*. Variables include: ambient value (av), ambient value weight (aw), ambient bounces (ab), ambient accuracy (aa), ambient divisions (ad), ambient radius (ar), ambient samples (as), black&white (b), direct jitter (dj), direct sampling (ds), direct threshold (dt), direct visibility (dv), irradiance (i), limit weight (lw), limit recursion (lr), medium extinction (me), medium albedo (ma), medium eccentricity (mg), medium sampling (ms), pixel sample (ps), pixel threshold (pt), back face visibility (bv), specular jitter (sj), and specular threshold (st). Once a variable has been changed, the "new" command can be used to recompute the image with the new parameters. If a program variable is not available here, it may show up under some other command or it may be impossible to change once the program is running.

trace [*xbeg ybeg zbeg xdir ydir zdir*]
: Trace a ray. If the ray origin and direction are absent, the cursor is used to pick a location in the image to trace. The object intersected and its material, location and value are displayed.

view [*file* [*comments*]]
: Check/change view parameters. If *file* is present, the view parameters are appended to a file, followed by *comments* if any. Alternatively, view options may be given directly on the command line instead of an output view file. Otherwise, view parameters are displayed and changed interactively.

V [*vw* [*rfile*]]
: Append the current view as view *vw* in the rad file *rfile*. Compliment to *L* command. Note that the view is simply appended to the file, and previous views with the same name should be removed before using the file with *rad*.

write [*file*]
: Write picture to *file*. If argument is missing, the current file name is used.

^Z
: Stop the program. The screen will be redrawn when the program resumes.

ENVIRONMENT

`RAYPATH`	the directories to check for auxiliary files
`DISPLAY_GAMMA`	the value to use for monitor gamma correction

AUTHOR

Greg Ward

SEE ALSO

`getinfo(1)`, `lookamb(1)`, `oconv(1)`, `pfilt(1)`, `rad(1)`, `rpict(1)`, `rtrace(1)`

NAME

t4014 - output metafile to Tektronix t4014 graphics terminal

SYNOPSIS

t4014 [-c | -r] file ..

DESCRIPTION

t4014 reads each metafile *file* in sequence and converts it to output suitable for the Tektronix t4014 and 4016 graphics terminals. If the option *c* is specified, the input files are only conditioned for output, ie. expanded and sorted (see pexpand and psort). This is useful if many copies of the same output is desired. If the option *r* is instead specified, the input is assumed already to be conditioned. If no input files are specified, the standard input is read.

-c	Condition the input only.
-r	Input is already conditioned, output only.

EXAMPLE

To display the plot example.plt:

```
bgraph example.plt | t4014
```

FILES

see `pexpand(1)`

AUTHOR

Greg Ward

BUGS

Area fill is noticably lacking. Line textures fail on most 4014 emulators.

SEE ALSO

`bgraph(1)`, `cv(1)`, `igraph(1)`, `pexpand(1)`

NAME

tabfunc - convert table to functions for rcalc, etc.

SYNOPSIS

tabfunc [-i] func1 [func2 ..]

DESCRIPTION

Tabfunc reads a table of numbers from the standard input and converts it to an expression suitable for *calc*(*1*), *rcalc*(*1*) and their cousins. The input must consist of a M x N matrix of real numbers, with exactly one row per line. The number of columns must always be the same in each line, separated by whitespace and/or commas, with no missing values. The first column is always the independent variable, whose value indexes all of the other elements. This value does not need to be evenly spaced, but it must be either monotonically increasing or monotonically decreasing. (I.e. it cannot go up and then down, or down and then up.) Maximum input line width is 4096 characters and the maximum number of data rows is 1024. Input lines not beginning with a numerical value will be silently ignored.

The command-line arguments given to tabfunc are the names to be assigned to each column. *Tabfunc* then produces a single function for each column given. If there are some columns which should be skipped, the dummy name "0" may be given instead of a valid identifier. (It is not necessary to specify a dummy name for extra columns at the end of the matrix.)

The -i option causes tabfunc to produce a description that will interpolate values in between those given for the independent variable on the input.

EXAMPLE

To convert a small data table and feed it to rcalc for some calculation:

```
rcalc -e `tabfunc f1 f2 < table.dat` -f com.cal
```

AUTHOR

Greg Ward

SEE ALSO

`cnt(1)`, `lam(1)`, `neat(1)`, `rcalc(1)`, `total(1)`

NAME

thf2rad - convert GDS things file to *Radiance* description

SYNOPSIS

thf2rad [-n name][-r rad] [input ..]

DESCRIPTION

Thf2rad converts one or more GDS things files to a *Radiance* scene description. The material names for the surfaces will be those assigned in GDS. The *-n* option may be used to give a name prefix to all the surfaces. The *-r* option may be used to specify a radius for line segments. By default, this value is zero, which means that lines will be ignored. By setting it to some positive value, cylinders of the given radius will represent lines.

EXAMPLE

To translate two things files into one *Radiance* file with the prefix "gds":

```
thf2rad -n gds building1.thf building2.thf >
building1+2.rad
```

To create an octree directly from a things file, giving lines a radius of 0.1:

```
oconv source.rad materials.rad '!thf2rad -r .1
building1.thf' > building1.oct
```

AUTHOR

Greg Ward and Charles Ehrlich

SEE ALSO

`arch2rad(1)`, `ies2rad(1)`, `oconv(1)`, `xform(1)`

NAME

tmesh2rad - convert a triangular mesh to a *Radiance* scene description

SYNOPSIS

tmesh2rad [-o obj][-m mat][-p pat] [input ..]

DESCRIPTION

Tmesh2rad converts one or more triangle-mesh files to a *Radiance* scene description. The *-o* option may be used to assign a default object name. The single letter "T" is used if no name is given on the command line or in the file. The *-m* option may be used to assign a default material name. The non-material "void" is used as a default if none is given on the command line or in the file. The *-p* option may be used to assign a default picture for a surface pattern. If none is given on the command line or in the file, the surface will not have an associated pattern.

FILE FORMAT

A triangle-mesh is a free-format ASCII file composed of the following eight primitive types. Each primitive is begun with a single, white-space-delimited letter:

# Comment

> Whatever follows up until the end of line is passed as a comment to the output. Note that there must be at least one space or tab following the pound-sign.

o name

> The white-space-delimited string name is used as a prefix for all following output triangles.

m *material*

> The white-space-delimited string *material* is used as the modifier name for all following output triangles.

p picture

> The white-space-delimited string picture is used as the name of the *Radiance* picture file to be used as a pattern for all following output triangles with properly defined vertices. (See *i* primitive below.)

v id x y z

Defines the vertex *id* with 3-dimensional coordinates *x*, *y* and *z*. The identifier, *id* must be some small, non-negative integer value. If the same integer is used for a later vertex definition, this definition will be lost, though any triangles using the vertex prior to its redefinition will be unaffected.

n nx ny nz

Defines a surface normal vector with the 3dimensional components *nx*, *ny* and *nz*. This vector will be associated with the most recently defined vertex, and is often placed on the same line as the vertex definition for clarity. The vector need not be normalized.

i u v

Defines a picture index for the most recently defined vertex. The *u* value will be used to lookup the horizontal pixel coordinate in the currently defined picture. The *v* value will be used to lookup the vertical pixel coordinate. (See the *Radiance* reference manual for details on picture coordinate values.) As with associated surface normals, picture indices are interpolated using barycentric coordinates based on the triangle vertices. If these coordinates are calculated correctly, this should result in a smooth mapping of a pattern onto the surface mesh.

t *id1 id2 id3*

Create a triangle connecting the three vertices identified by *id1*, *id2* and *id3*. The right-hand rule is used to determine the default surface normal orientation, and this should not be too far from the associated vertex normals (if any). All three vertices must have an associated normal if the triangle is to be smoothed. If a picture file is defined and all three vertices have pattern indices associated with them, then this picture will be used as a pattern to modify the triangle's color.

We realize there are many similar T-mesh file formats in existence, and that it would have been just as easy to support one of these formats directly. The disadvantage to supporting an existing format is that conversion from other formats might prove difficult. It was our hope to provide a "greatest common multiple" format that would support all similar T-mesh formats, rather than supporting WaveFront's .obj format (for example) and being unable to associate a pattern with an object. Converting from other formats should be relatively straightforward. In many cases, an *awk*(*1*), *rcalc*(*1*) or even a *sed*(*1*) script should be sufficient.

EXAMPLE

Here is an example T-mesh file:

```
# Our object name:
o test_object # Our material:
m puce
# Our vertices:
v 1 10 15 5
v 2 10 -15 5
v 3 0    -15 0
v 4 -10 15 -5
# Two triangles joined together:
t 1 2 3
t 2 3 4
```

Which generates the following output:

```
## T-mesh read from: <stdin>
# Our material:
# Our vertices:
# Two triangles joined together:
puce polygon test_object.1
0
0
9
     10   15   5
     10   -15  5
     0    -15  0

puce polygon test_object.2
0
0
9
     10   -15  5
     0    -15  0
     -10  15   -5
```

Here is another example:

```
# A partial cylinder:
m BluePlastic
v 1 -14.673 -3.119 50 n -0.95677 -0.203374
1.17936e-10
v 2 -12.136 -8.817 -50 n -0.791363 -0.574922
4.84915e-10
v 3 -12.136 -8.817 50 n -0.791363 -0.574922
4.84915e-10 t 1 2 3
m OrangePlastic
v 1 -7.501 -12.991 50 n -0.549094 -0.812427
-1.45812e-09
v 2 -12.136 -8.817 50 n -0.791363 -0.574922
4.84915e-10
v 3 -12.136 -8.817 -50 n -0.791363 -0.574922
4.84915e-10 t 1 2 3
m BluePlastic
v 1 -1.568 -14.918 50 n -0.171094 -0.965568
-5.69788e-09
v 2 -7.501 -12.991 50 n -0.549094 -0.812427
-1.45812e-09
v 3 -7.501 -12.991 -50 n -0.429001 -0.881759
-3.6502e-09 t 1 2 3
```

Note that the same three vertices were used repeatedly, and intermingled with the triangle definitions.

AUTHOR

Greg Ward

BUGS

Triangle smoothing doesn't work very well for glass or trans material types in *Radiance* , since textures cause distorted transmission through these materials. It is best to use the dielectric material type if smooth transmission is desired.

SEE ALSO

`arch2rad(1)`, `awk(1)`, `ies2rad(1)`, `thf2rad(1)`, `oconv(1)`, `rcalc(1)`, `sed(1)`, `xform(1)`

NAME

total - sum up columns

SYNOPSIS

total [-m][-sE | -p | -u | -l][-tC][-N [-r]] [file ..]

DESCRIPTION

Total sums up columns of real numbers from one or more files and prints out the result on its standard output.

By default, total computes the straigt sum of each input column, but multiplication can be specified instead with the *-p* option. Likewise, the *-u* option means find the upper limit (maximum), and *-l* means find the lower limit (minimum).

Sums of powers can be computed by giving an exponent with the *-s* option. (Note that there is no space between the *-s* and the exponent.) This exponent can be any real number, positive or negative. The absolute value of the input is always taken before the power is computed in order to avoid complex results. Thus, *-s1* will produce a sum of absolute values. The default power (zero) is interpreted as a straight sum without taking absolute values.

The -m option can be used to compute the mean rather than the total. For sums, the arithmetic mean is computed. For products, the geometric mean is computed. (A logarithmic sum of absolute values is used to avoid overflow, and zero values are silently ignored.)

A count can be given as the number of lines to read before computing a result. By default, *total* reads each file to its end before producing its result, but the *-N* option (where N is a decimal integer) tells *total* to produce a result and reset the calculation after every N input lines. In addition, the *-r* option can be specified to override reinitialization and thus give a running total every N lines. If the end of file is reached, the current total is printed and the calculation is reset before the next file (with or without the *-r* option).

The -tC option can be used to specify the input and output tab character. The default tab character is TAB.

If no files are given, the standard input is read.

EXAMPLE

To compute the RMS value of colon-separated columns in a file:

```
total -t: -m -s2 input
```

To produce a running product of values from a file:

```
total -p -1 -r input
```

BUGS

If the input files have varying numbers of columns, mean values will certainly be off. *Total* will ignore missing column entries if the tab separator is a non-white character, but cannot tell where a missing column should have been if the tab character is white.

AUTHOR

Greg Ward

SEE ALSO

`cnt(1)`, `lam(1)`, `neat(1)`, `rcalc(1)`, `tabfunc(1)`

NAME

trad - graphical user interface to *Radiance* rad(1) program

SYNOPSIS

trad [rfile]

DESCRIPTION

Trad is a graphical user interface to *rad*(*1*), which controls the operation of the basic *Radiance* scene compiling, rendering and picture filtering programs. Trad also includes links to a few utilities for displaying and converting results, but most of what it does can be done by editing a small text file, called the "rad input file". Scene creation still requires the use of a text or graphical editor, or translation from some external CAD format.

Trad is based on the Tcl/Tk wish(1) "windowing shell" written by John Ousterhout. (See below for instructions on installing this package if you do not have it already.)

The trad interface divides the rendering problem into seven screens: File, Scene, Zone, Views, Options, Action and Results. The File screen is used to load and save rad input files (a.k.a. project files). The Scene screen is used to name the *Radiance* input files associated with a particular project. The Zone screen is used to assign *rad* variables specific to the section of the model being rendered. The Views screen is used to define specific views to be rendered and set the picture file names and dimensions. The Options screen is used to adjust rendering quality and other parameters. The Action screen is used to initiate interactive and batch renderings. The Results screen is used to display, convert and print the rendered *Radiance* pictures.

If trad is called with no rad input file name on the command line, it will start with the File screen and you must enter a valid project file before you will be allowed to continue. If *rfile* is given, then *trad* attempts to open this file. If no such file exists, *trad* assumes you are creating a new file by this name and goes to the Scene screen so you may identify the appropriate *Radiance* input files. If the file exists but not all renderings have been finished, *trad* goes first to the Action screen, assuming you will want to do something. If the file exists and all renderings have completed and are up-to-date, *trad* goes to the Results screen so that you may examine the final pictures.

Trad includes an extensive help facility, which may be accessed either by pressing the "HELP" button and searching through the category and topic menus, or by holding the Control key and pressing the left mouse button on the mysterious widget (i.e. the button, entry window, or list box you are curious about).

INSTALLING TCL/TK

The Tcl/Tk package is available by anonymous ftp from ftp.smli.com (204.153.12.45) in the /pub/tcl directory. (Tcl stands for "Tool Command Language," and is pronounced "tickle".) *Trad* is based on Tk release 4.0, which in turn is based on Tcl release 7.4. Although *trad* uses only the *wish* program from this package, *wish* itself depends on additional Tcl and Tk libraries, and the two toolkits must be compiled one after the other.

To compile the Tcl/Tk package, download the files "tcl7.4.tar.Z" and "tk4.0.tar.Z" and uncompress and untar them. Then, follow the instructions in the README file in the tcl7.4/ directory to configure the Makefile for your system and install the software. (Usually, it is best to do this as root or have it done by your system administrator if you do not have root privileges.) Then, do the same in the tk4.0/ directory.

Trad should run without modification once this is done correctly.

AUTHOR

Greg Ward
wish and Tcl/Tk language by John Ousterhout

SEE ALSO

`oconv(1)`, `pfilt(1)`, `rad(1)`, `rpict(1)`, `rview(1)`, `wish(1)`, `ximage(1)`

NAME

ttyimage - *Radiance* driver for dumb ASCII terminal

SYNOPSIS

ttyimage [-c resolu][-r] [pixfile]

DESCRIPTION

Ttyimage takes the *Radiance* picture file *pixfile* and displays it on a dumb terminal. If no *pixfile* is given, the standard input is read.

AUTHOR

Greg Ward

SEE ALSO

`aed(1)`, `pfilt(1)`, `rpict(1)`, `ximage(1)`

NAME

vgaimage - *Radiance* picture display program for VGA

SYNOPSIS

vgaimage [-c ncolors][-d][-b][-m][-g gamma][-e +/-stops] pixfile

DESCRIPTION

Vgaimage takes the *Radiance* picture file *pixfile* and displays it on a VGA or Super VGA card. The *-c* option specifies the number of colors to use (default fills color table). The *-d* option turns off color dithering. The *-b* option displays the image in black and white (greyscale). The *-m* option forces monochrome output. The *-g* option specifies the exponent used in gamma correction; the default value is given by the environment variable GAMMA, or 2.2 if GAMMA is undefined. The *-e* option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency.

If no pixfile is given, input is read from stdin provided either the *-b* or *-m* option is in effect.

COMMANDS

Once a picture is displayed, the user may perform a number of operations. Some of the operations make use of an area of interest, defined by pressing a mouse button and dragging the cursor over a section of the image. Pressing a button and immediately releasing it defines a single point as the area of interest. A command is a single character.

q	Quit program.
<return>	Display the radiance averaged over the area of interest.
l	Display the luminance value in the area of interest.
c	Display the color in the area of interest.

ENVIRONMENT

```
GAMMA     the default gamma correction value
```

AUTHORS

Greg Ward

SEE ALSO

`aedimage(1)`, `pfilt(1)`, `rpict(1)`, `rtrace(1)`, `ximage(1)`

NAME

vwright - normalize a *Radiance* view, shift it to the right

SYNOPSIS

vwright distance

vwright name

DESCRIPTION

Vwright shifts a *Radiance* view from a picture or view file given on the standard input the specified distance to the right, putting out a complete set of view parameters in a single line on the standard output. This utility is most often used to compute a right-eyed view from a left-eye view for stereo imaging.

The distance given is in world coordinate units. A negative value indicates a shift to the left rather than the right.

The second form substitutes a name prefix in place of the shift distance, and produces constant assignments on the standard output suitable for passing directly to *rcalc*(*1*). For a given prefix *N*, the constant names are as follows:

Nt:	view type ('v'==1,'l'==2,'a'==3,'h'==4,'c'==5)
Npx:	view point x value
Npy:	view point y value
Npz:	view point z value
Ndx:	view direction x value (normalized)
Ndy:	view direction y value (normalized)
Ndz:	view direction z value (normalized)
Nux:	view up vector x value (normalized)
Nuy:	view up vector y value (normalized)
Nuz:	view up vector z value (normalized)
Nh:	view horizontal size
Nv:	view vertical size
Ns:	view shift
N1:	view lift
No:	view fore clipping distance
Na:	view aft clipping distance
Nhx:	derived horizontal image vector x value (normalized)

Nhy: derived horizontal image vector y value (normalized)
Nhz: derived horizontal image vector z value (normalized)
Nhn: derived horizontal image vector multiplier
Nvx: derived vertical image vector x value (normalized)
Nvy: derived vertical image vector y value (normalized)
Nvz: derived vertical image vector z value (normalized)
Nvn: derived vertical image vector multiplier

EXAMPLES

To start *rpict*(*1*) on a view .06 meters left of the view in the file "right.vf":

```
rpict `vwright -.06 < right.vf` scene.oct >
right.pic &
```

To move the rad(1) view named "left" 2.5 inches to the right and render from there:

```
rad -v "vright `rad -n -s -V -v left examp.rif |
vwright 2.5`" examp.rif &
```

To pass a view to rcalc for conversion to some other view:

```
rcalc -n -e `vwright orig < orig.vf` -f
viewmod.cal -o view.fmt > new.vf
```

AUTHOR

Greg Ward

SEE ALSO

`pdfblur(1)`, `rad(1)`, `rcalc(1)`, `rpict(1)`, `rview(1)`

NAME

x11meta - output metafile graphics to X11

SYNOPSIS

x11meta [-c | -r] file ..

DESCRIPTION

X11meta reads each metafile *file* in sequence and sends it to the default X window system. If the option *c* is specified, the input files are only conditioned for output, ie. expanded and sorted (see pexpand and psort). If the option *r* is instead specified, the input is assumed already to be conditioned. If no input files are specified, the standard input is read.

-c	Condition the input only.
-r	Input is already conditioned, output only.

EXAMPLE

To plot the chart example.bar:

```
bgraph example.plt | x11meta
```

FILES

see `pexpand(1)` and `psort(1)`

AUTHOR

Greg Ward

SEE ALSO

`cv(1)`, `fx80(1)`, `metafile(5)`, `mt160l(1)`, `pexpand(1)`, `psort(1)`, `tcurve(1)`, `tscat(1)`, `tplot(1)`

NAME

xform - transform a *Radiance* scene description

SYNOPSIS

xform [-c][-I][-n name][-m newmod][xf0][-a n1 xf1 ..][-i 1 xff] file ..

DESCRIPTION

Xform transforms each scene description *file* according to the options given. The *-c* option causes commands in the file not to be expanded. The default is to execute all in line commands. (See note below about file names.) The *-n* option causes all identifiers to be prefixed with *name*. The *-m* option causes all surfaces to be given the modifier *newmod*. The *-I* option causes all surfaces to be inverted, reversing their surface normal orientations. These options are followed by the transformation options, which are described below.

If one or more scene files are given on the command line, *xform* will search the *Radiance* library directories for each file. (No search takes place if a file name begins with a '.', '/' or '~' character.) Unless the *-c* option is present, *xform* will also change to that file's directory before loading it. Thus, any commands executed within that file will happen in that file's directory, which simplifies object hierarchy construction. If no *file* is given, the standard input is read.

The transformation consists of a sequence of operations which are executed in the order they appear.

OPTIONS

-t *x y z*
: Translate the scene along the vector *x y z* .

-rx degrees
: Rotate the scene *degrees* about the x axis. A positive rotation corresponds to counter-clockwise when looking down the axis.

-ry *degrees*
: Rotate the scene *degrees* about the y axis.

-rz *degrees*
: Rotate the scene *degrees* about the z axis.

-s *factor*
: Scale the scene by *factor*.

-mx Mirror the scene about the yz plane.

-my Mirror the scene about the xz plane.

-mz Mirror the scene about the xy plane.

-i *count*

Iterate (repeat) the following transformation (up to the next -*i* option) *count* times. This option is primarily to support the -*a* option, which is described below.

Arrays

An array is a repeated transformation that results in a repeated object. It is specified using the -*a* option, which takes the number to repeat as its argument. The objects will step by the transformation given between this -*a* option and the next -*a* or -*i* option. The first object will have zero applications of the transform. A two dimensional array is given by two different transformations each followed by an array count.

EXAMPLE

To rotate "book" 30 degrees about the x axis then move 20 in y, prepending the name book1:

```
xform -n book1 -rx 30 -t 0 20 0 book > book1
```

To expand all commands and see what information is actually used by *Radiance*:

```
xform scene | more
```

To create a two dimensional array of 20 lights, after an initial rotation and followed by a global translation (no command expansion):

```
xform -c -rz 90 -a 5 -t 2 0 0 -a 4 -t 0 1.5 0 -i 1
-t 0 0 10 light
```

ENVIRONMENT

`RAYPATH` path to search for scene files

AUTHOR

Greg Ward

BUGS

Only regular (distortion-free) transformations are allowed.

SEE ALSO

`genbox(1)`, `gensurf(1)`, `oconv(1)`, `replmarks(1)`, `rpict(1)`, `rview(1)`

NAME

xglaresrc - dislpay glare sources under X11

SYNOPSIS

xglaresrc [-n windowname][-c r g b] pictfile [glarefile]

DESCRIPTION

Xglaresrc displays the sources located by *findglare*(*1*) in the picture *pictfile* by displaying their average luminance and drawing circles around them in the image. If *pictfile* is already being displayed by *ximage*(*1*), *xglaresrc* will raise (or deiconify) this window. If *pictfile* is not being displayed, *xglaresrc* will start *ximage*(*1*) in the background automatically. (To quit from *ximage*(*1*), you can type 'q' in its display window.) Usually, *pictfile* will be the image that was used by *findglare*(*1*) to locate glare sources.

If the desired image is being displayed under a different name, perhaps even by a different display program than *xim_*a*ge*(*1*), the -*n* option can be used to give an alternative *windowname* for locating the correct display window, and the -*c* option can be used to specify a different line and text color (default 1 0 0). If the inpute *glarefile* is not given, *xglaresrc* reads from its standard input.

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

SEE ALSO

`findglare(1)`, `glare(1)`, `glarendx(1)`, `ximage(1)`

NAME

ximage - *Radiance* driver for X window system

SYNOPSIS

ximage [=geometry][-di display][-c ncolors][-d][-b][-m][-g gamma][-f][-e spec][-ospec][-t intvl][-s] picture ..

DESCRIPTION

Ximage takes one or more *Radiance* picture files and displays them on an X server. The *-c* option specifies the number of colors to use (default fills color table). The *-d* option turns off color dithering. The *-b* option displays the image in black and white (greyscale). The *-m* option forces monochrome output. The *-g* option specifies the exponent used in gamma correction; the default value is 2.2. The *-f* option stores a Pixmap on the server side for faster refresh. This may not work with large images on some servers. The *-o* option specifies a sequence of information to print to the standard output for the `t' command (see below). The *-t* option specifies a minimum interval (in milliseconds) between successive ray outputs in mouse tracking mode (right button pressed).

The -e option specifies an exposure compensation in f-stops (powers of two). Only integer stops are allowed, for efficiency. If the special word, *auto* is given instead of a number of stops, then *ximage* performs an automatic exposure adjustment similar to *pcond*(*1*), compressing the dynamic range of the image to fit within the dynamic range of the display. If the special word, *human* is given instead, then *ximage* performs an exposure adjustment similar to *pcond* with the *-s* and *-c* options, which compensate for human contrast and color sensitivity at the corresponding scene luminance levels. This option yeilds and appearance of the scene on the display that closely matches what would be experienced in the real world.

The -s option tells ximage to display multiple pictures sequentially, rather than all at once. If no *picture* is given, input is read from stdin provided either the *-b* or *-m* option is in effect, or the X server is capable of 24-bit color. However, many of the commands given below will not work.

COMMANDS

Once a picture is displayed, the user may perform a number of operations. Some of the operations make use of an area of interest, defined by pressing the left mouse button and dragging the cursor over a section of the image.

Pressing the button and immediately releasing it defines a single point as the area of interest. A command is a single character.

q

Quit picture. (Also Q or ^D.)

<space>

Redraw the area of interest.

^R

Redraw the entire image.

<return>

Display the radiance averaged over the area of interest.

l

Display the luminance value in the area of interest. This assumes that the image was correctly computed in terms of luminance.

c

Display the color in the area of interest.

p

Display the x and y location of the cursor.

i

Identify identical pixels by assigning a random color at the cursor position. This is useful for displaying contours, especially when combined with the -b option.

t

Print information about the pixel under the cursor according to the string following the -*o* command line option. The valid characters for this option correspond roughly to the other *ximage* commands:

o	ray origin
d	ray direction
v	radiance value
l	luminance value
p	pixel position

The default output is ``-ood", which prints the ray origin and direction. This can be used as input to `rtrace(1)` to get additional information about the image (ie. pipe the output of ximage into rtrace). Pressing the middle mouse button is equivalent to typing the `t' key. Pressing and holding the right mouse button is equivalent to continuously pressing the `t' key.

=
: Adjust the exposure to the area of interest. A crude adjustment is made immediately, and the number of stops is printed while the colors are resampled. After a few seconds to a minute, the final image is redisplayed. If the area of interest is already within 1/2 stop of the ideal, no adjustment is made.

@
: Same as `=' command, only the exposure is adjusted to provide roughly the same visibility for the selected region on screen as a viewer would experience in the actual space. Like the `l' command, this adjustment assumes that the image has been correctly computed in terms of luminance. (See also the `h' command, below.)

a
: Perform automatic exposure compensation, as if *ximage* were started with the *-e auto* option.

h
: Perform human expsoure compensation, as if *ximage* were started with the *-e human* option.

0
: Reset the origin to the upper left corner of the image. This command is used to restore the original image position after using the shift or control key with the mouse to relocate the image within the frame (see below).

f
: Switch on the fast redraw option (*-f*), loading the image pixmap over to the server side. This command is useful when network delays are causing slow image refresh, and the user didn't notice it until after *ximage* was started.

F
: Switch off the fast redraw option. This frees up some memory on the server, as well as the color table for other windows.

In addition to the commands listed above, the control or shift key may be held while the cursor is dragged to reposition the image within the window.

X RESOURCES

`radiance.gamma`	the default gamma correction value

ENVIRONMENT

DISPLAY_GAMMA	the default gamma correction value

AUTHORS

Greg Ward
Anat Grynberg (Paris)
Philip Thompson (MIT)

SEE ALSO

`aedimage(1)`, `pcond(1)`, `pfilt(1)`, `rpict(1)`, `rtrace(1)`, `rview(1)`, `xglaresrc(1)`, `xshowtrace(1)`

NAME

xshowtrace - interactively show rays traced on *Radiance* image under X11

SYNOPSIS

xshowtrace [-s][rtrace options] octree picture

DESCRIPTION

Xshowtrace takes a *Radiance* octree and a picture file and displays it on an X11 window server using *ximage*(*1*). The picture should have been created from a previous *rpict*(*1*) or *rview*(*1*) calculation using the given octree. Once the image is displayed, the user can use the 't' command of *ximage* to select points on the image to display the ray tree. *Rtrace* then produces a ray tree, which *xshowtrace* will display (in red on a color screen). The *-s* option slows the display of each ray traced to make it easier to follow the process.

AUTHOR

Greg Ward

BUGS

If the pointer is moved between the time 't' is pressed and *xshowtrace* starts drawing rays, the rays will be displaced.

SEE ALSO

`oconv(1)`, `rpict(1)`, `rtrace(1)`, `rview(1)`, `ximage(1)`