NAME

aedimage - RADIANCE driver for AED 512 color graphics terminal

SYNOPSIS

aedimage [**–d**][**–b**] **pixfile**

DESCRIPTION

Aedimage takes the RADIANCE picture file *pixfile* and displays it on an AED 512 color graphics terminal. The picture's size must be equal to or smaller than the device resolution, which is 512 by 512. The *–d* option produces an image with random color dithering. This gives a more realistic display of color at the expense of some spatial resolution. It also requires longer to put up the display, since run length encoding is not effective with dithered colors. The *–b* option displays the image in black and white, improving the speed and reducing quantization effects.

COMMANDS

Once a picture is displayed, a blinking cursor appears, and the user may perform a number of operations. A command is a single character, preceded by an optional count.

q Quit program.

<return> Display the radiance at the cursor position. If a count N is given, the square area above and to the right of cursor containing N by N pixels is averaged for the reported value.

l Display the luminance value at the cursor position. Averaging is done in the same manner as for radiance.

c Display the color at the cursor position. Averaging is done in the same manner as for radiance.

p Display the x and y location of the cursor.

w Shift all values in the color table down by the count N (default 1).

W Shift all values in the color table up by the count N (default 1).

r Shift red values down.

R Shift red value up.

g Shift green values down.

G Shift green values up.

b Shift blue values down.

B Shift blue values up.

AUTHORS

Greg Ward
Paul Heckbert provided the code for color quantization

SEE ALSO

pfilt(1), rpict(1), ximage(1)

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 dot-dashed.

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)

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
ˆD
```

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. 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.

select(N, a1, a2, ..)
: return aN (N is rounded to the nearest integer). 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 images. 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 image:

```
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

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

genprism - generate a RADIANCE description of a prism

SYNOPSIS

genprism mat name { **- | vfile | N v1 v2 .. vN** } [**–l lvect**][**–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 *–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

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 hour [**options**]
gensky -ang altitude azimuth [**options**]
gensky -defaults

DESCRIPTION

Gensky produces a RADIANCE scene description for the CIE standard sky distribution at the given date and local standard time. If the hour is preceded by a plus sign ('+'), then it is interpreted as local solar time instead of standard time. 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.

+c 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.* (See *–r* option below.)

-b *brt* The zenith brightness is *brt.* Zenith brightness (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.

-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.

-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.

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 standard time at a site latitude of 42 degrees, 108 degrees west longitude, and a 110 degree standard meridian:

```
gensky 7 4 14.5 +s -a 42 -o 108 -m 110
```

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-(.5-t)*(.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), 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**]

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)
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.

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), igraph(1), rpict(1), xglaresrc(1), ximage(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 a third file will be created (with a "+.dat" 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 color-balanced 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 can be used for only one input file, and is required when reading data from the standard input.

-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 geometry given by the IES input file and use instead an illum sphere with radius *rad.* This option is useful when the user wishes to add a more accurate geometric description to the light source model.

-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

The IES specification provides 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 IES standard should really pay more attention to source geometry.

SEE ALSO

oconv(1), pfilt(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 bar chart example.bar:

tbar example.bar **|** output imagew

FILES

see pexpand(1) and psort(1)

AUTHOR

Greg Ward

SEE ALSO

cv(1), impress(1), mx80(1), mt160(1), mt160l(1), output(1), pexpand(1), psort(1), tbar(1), tcurve(1), tscat(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 scatter plot example.sct to the ip4 printer:

tscat example.sct **|** impress **|** ipr -P ip4

FILES

see pexpand(1)

AUTHORS

William LeFebvre and Greg Ward

SEE ALSO

ipr(1), imagew(1), mx80(1), t4014(1), tbar(1), tcurve(1), tscat(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), 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

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

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.

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), 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 bar chart example.bar:

tbar example.bar **|** output mx80

FILES

see pexpand(1) and psort(1)

AUTHOR

Greg Ward

BUGS

Currently, different character widths and densities are not supported.

SEE ALSO

cv(1), image(1), impress(1), mt160(1), mt160l(1), output(1), pexpand(1), psort(1), tbar(1), tcurve(1), tscat(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 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

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 *side.* 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. The default number is 5.

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) can be used to automate octree creation and maintenance.

SEE ALSO

getbbox(1), getinfo(1), make(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 (+/- 4 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 variable *nfiles* gives the number of input files present. Additionally, the variables *x* and *y* give the current output pixel location for use in spatially dependent functions, *xmax* and *ymax* give the input resolution, and *xres* and *yres* give the output resolution (usually the same, but see below). The functions *re(n), ge(n), be(n),* and *le(n)* give the exposure values for picture *n.*

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+(y-50)^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**] **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 position of the picture's lower left corner in the final output. 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

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 .6 and a reduction in image width and height of 2 or 3 produces the highest quality pictures. A radius greater than 0.7 results in a somewhat defocused picture.

-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 1000.

-n *N* Set the number of points on star patterns to *N.* A value of zero turns star patterns off. The default is 0.

-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/pf??????

AUTHOR

Greg Ward

SEE ALSO

getinfo(1), ies2rad(1), pcompos(1), pflip(1), pinterp(1), pvalue(1), protate(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*][**−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 try and produce a reasonable estimate of the desired view. The first input picture specified takes priority in the final output, but if subsequent pictures contain pixels that map closer to the viewer by the *−t* threshold (.02 by default), then those pixels will be used instead. 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.

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 will appear to have holes in them and large areas at the image borders will not be filled by the foreground or background algorithms. Also, specular reflections and highlights will not be interpolated correctly, since their view-dependent appearance will not be accounted for by the program.

EXAMPLE

To interpolate two frames of a walk-through animation:

```
pinterp -vf 27.vf -x 1024 -y 800 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), pfilt(1), rpict(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

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

EXAMPLE

To print the scatter plot example.sct to the ip4 printer:

tscat example.sct **|** psmeta **|** lpr -P ap5

AUTHOR

Greg Ward

SEE ALSO

lpr(1), imagew(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**]

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.

−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.

−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.

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

−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. 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". 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

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_pict - convert Radiance pictures to Macintosh PICT files

SYNOPSIS

ra_pict [**−e +/-** *stops*] [**−v**] [**−g** *gamma*] [*infile* [*outfile*]]

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

rpict(1)

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_pixar - convert RADIANCE picture to/from PIXAR picture

SYNOPSIS

ra_pixar input.pic output.pix
ra_pixar -r input.pix output.pic

DESCRIPTION

Ra_pixar converts between RADIANCE and PIXAR file formats. In the default mode, a RADIANCE picture is converted to a PIXAR picture of the same horizontal and vertical dimensions with 12 bits per color per pixel. The *−r* option invokes a reverse conversion, from a PIXAR picture to a RADIANCE picture.

AUTHOR

David G. Jones (djones@awesome.berkeley.edu)

BUGS

Only standard PIXAR 12-bit,RGB files are written.
Input PIXAR files are less restricted, though not well tested.

SEE ALSO

ra_pr(1), ra_t16(1), pfilt(1), ximage(1)

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 ASCII file

SYNOPSIS

ra_ps [**−e +/-stops**][**−n ncopies**] [**input** [**output**]]

DESCRIPTION

Ra_ps translates a RADIANCE to a greyscale Adobe PostScript file for printing on a laser printer. 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.

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. 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 run-length 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)

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**] **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 color-mapped 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 *−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
Paul Heckbert provided the code for 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

rcalc - record calculator

SYNOPSIS

rcalc [**–b**][**–l**][**–n**][**–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 an input format matches the beginning of a line. With the *–l* option, an input format can match anywhere on a line. 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. 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 pre-defined functions and variables is provided:

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.

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

recno the number of records recognized thus far.

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), ev(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. The *–m* option causes all surfaces to be given the modifier *newmod.* If no *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**] [**ˆfile**] [**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 a caret ('ˆ') 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.

-vt*t* Set view type to *t.* If *t* is 'v', a perspective view is selected. If *t* is 'l', a parallel view is used. 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.*

-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 can 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 *size.* This specifies the sample spacing 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 low-resolution 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.

-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.

-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 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 can be given, but each must appear in a separate option.

-ai *mat* Add *mat* 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.*

–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 *–di* option (below) 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.

-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 file options *–o, –r,* and *–z.* 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 is 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.

EXAMPLE

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

/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), rtrace(1), rview(1)

NAME

rpiece - render pieces of a RADIANCE picture

SYNOPSIS

rpiece [**−v**][**−x xpsiz**][**−y ypsiz**][**−X xmult**][**−Y ymult**][**−F syncfile**] [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. Each piece has a size *xpsiz* by *ypsiz* (or smaller, depending on the *−pa* option and other view options), and the total picture dimensions are these parameters multiplied by *xmult* and *ymult.* Thus, the picture is broken into *xmult* by *ymult* smaller pieces, and each piece is rendered separately.

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 *xmult−1* and *Y* runs from zero to *ymult−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 *xmult* and *ymult,* 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. This will continue until all the pieces are done, at which point all of the *rpiece* processes will terminate.

The *−v* flag switches on verbose mode, where *rpiece* reports to the standard output after each piece is finished.

EXAMPLE

First *rpiece* process is started on the machine "goober":

```
goober% echo 1 8 > syncfile
goober% rpiece -F syncfile -x 1024 -y 128 -vf view -o picture octree &
```

Second *rpiece* processes is started on the machine "sucker":

```
sucker% rpiece -F syncfile -x 1024 -y 128 -vf view -o picture octree &
```

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. In any case, setting *xmult* to one and dividing the picture in the vertical direction will result in fewer system-related delays.

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 = xpsiz∗xmult∗ypsiz∗ymult/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.

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**] [**ˆfile**] **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 a caret ('ˆ') 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' can be used to denote 4-byte floating point (Radiance) color fromat 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.

–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. This option overrides the *–o* option.

–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.

-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.

-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 machine-independent 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 can be given, but each must appear in a separate option.

-ai *mat* Add *mat* 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.*

-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 *pinterp(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=54∗$1+106∗$2+20∗$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

/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**] [**$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 a caret ('^') 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. 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* is absent, 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.

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.

ˆR 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 look 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 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), pixel sample (ps), pixel threshold (pt), 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.

write *[file]*
Write picture to *file.* If argument is missing, the current file name is used.

ˆZ Stop the program. The screen will be redrawn when the program resumes.

ENVIRONMENT

RAYPATH the directories to check for auxiliary files. GAMMA the value to use for monitor gamma correction.

AUTHOR

Greg Ward

SEE ALSO

getinfo(1), lookamb(1), oconv(1), pfilt(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 scatter plot example.sct:

tscat example.sct **|** t4014

FILES

see pexpand(1)

AUTHOR

Greg Ward

BUGS

Area fill is noticably lacking. Line textures fail on most 4014 emulators.

SEE ALSO

cv(1), fx80(1), pexpand(1), tbar(1), tcurve(1), tscat(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 .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

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)

NAME

ttyimage - RADIANCE driver for X window system

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

xform - transform a RADIANCE scene description

SYNOPSIS

xform [**–e**][**–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 *–e* option causes commands in the file to be expanded, and is recommended when xform is used as a command in a scene file for efficiency. 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 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 -e scene | more

To create a two dimensional array of 20 lights, after an initial rotation and followed by a global translation:

xform -rz 90 -a 5 -t 2 0 0 -a 4 -t 0 1.5 0 -i 1 -t 0 0 10 light

AUTHOR

Greg Ward

BUGS

Only regular (distortion-free) transformations are allowed.

The array option does not work for scenes read from the standard input.

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 *ximage(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**][**−c ncolors**][**−d**][**−b**][**−m**][**−g gamma**][**−f**][**−e +/-stops**] **pixfile**

DESCRIPTION

Ximage takes the RADIANCE picture file *pixfile* and displays it on an X window 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 *−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. 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 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.

<space> Redraw the area of interest.

ˆR Redraw the entire image.

<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.

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 the ray origin and direction for the given cursor location. This can be used as input to rtrace(1) to get additional information about the image (ie. pipe the output of ximage into rtrace).

= 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 stop of the ideal, the bell is sounded and no adjustment is made.

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).

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.

ENVIRONMENT

GAMMA the default gamma correction value

AUTHORS

Greg Ward
Paul Heckbert provided the code for color quantization

SEE ALSO

aedimage(1), pfilt(1), rpict(1), rtrace(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)