NAME

pcomb - combine RADIANCE pictures

SYNOPSIS

pcomb [**-h**][**-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. The variable and function definitions in each *−f source* file are read and compiled from the RADIANCE library where it is found. 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 (+/- 32 pixels) due to efficiency considerations. However, it is not usually necessary to worry about this problem -- if the requested offset is not available, the next best pixel is returned instead.

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

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

The *−h* option may be used to reduce the information header size, which can grow disproportionately after multiple runs of *pcomb* and/or *pcompos(1).* 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' inp.hdr > gam.hdr
```

To perform some special filtering:

```
pcomb −f myfilt.cal −x xmax/2 −y ymax/2 input.hdr > filtered.hdr
```

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
```

ENVIRONMENT

RAYPATH	the directories to check for auxiliary files.

AUTHOR

Greg Ward

SEE ALSO

getinfo(1), icalc(1), pcompos(1), pfilt(1), rpict(1)