NAME

rgb - Silicon Graphics rgb image file format

SYNOPSIS

#include <image.h>

DESCRIPTION

IRIS image files are used to store 1,2 and 3 dimensional arrays of pixel values that contain either 1 or 2 bytes per pixel. Pixel values are signed integers that cover the range 0..255 or -32768..32767 (i.e. 1 or 2 bytes). Image files are currently used to store rgb screen dumps, black and white images, color index images, as well as colormaps. The image library provides tools to manipulate these files. To include the image library place the token -limage on the compile line for your program. Also, be sure to include image.h from /usr/include/gl in any source files that use these routines. The following routines provide a procedural interface to image files:

Opening and Closing an Image File

IMAGE *iopen (file, mode [, type, dim, xsize, ysize, zsize]) char *file; register char *mode; unsigned int type, dim, xsize, ysize, zsize;

Opens an image file for reading or writing and returns a pointer to IMAGE in the same style as the UNIX standard i/o library. A return value of 0 means the function failed to successfully open the file that was named. To open an image file for reading, iopen should be called with 2 arguments, the name of the image file to open and a mode of "r". The dimensions of the image may be determined by referencing `image->xsize`, `image->ysize`, and `image->zsize`, where image is the value returned by iopen. xsize and ysize are defined in terms of pixels while zsize describes the number of channels (i.e. layers) the image contains. The value of `image->dim` indicates whether the image is just a single row (one dimensional) or is an array of rows (two dimensional) or is an array of 2 dimensional images (three dimensional). An rgb image can be thought of as a set of three 2 dimensional images. Sometimes this is referred to as a 3 channel image. An rgb color image consists of 3 channels (one channel each for the red green and blue components of the image) and is represented as a three dimensional image that is xsize by ysize by 3. A black and white image has one channel and is represented as a two dimensional image that is xsize by ysize. Other information may be found in `image->name` (holds the string that is usually the same as the actual image filename), `image->colormap` (defines whether the image is a series of intensity values, or color lookup table indices, or an actual colormap), `image->max` (the maximum intensity stored in the image), and `image->min` (the minimum intensity stored in the image). To open an image file for writing, iopen should be called with 7 arguments, the name of the image file to open, and a mode of "w", followed by the type, the number of dimensions and the xsize, ysize and zsize of the image. The type indicates how many bytes are stored per pixel value, and whether the image file should be run-length encoded. Type may be given as RLE(1), RLE(2), VERBATIM(1), or VERBATIM(2). Run- length encoded (RLE) image files are more efficiently stored than verbatim files where no compression algorithm is used. 1 or 2 in the above specifies how many bytes are used for each pixel in a colormap image, or for each channel in an rgb image. RLE(2) or VERBATIM(2) is used to store color index images that contain 12 bits per pixel. RLE(1) is the recommended default for rgb and black and white images.

iclose (image) register IMAGE *image;

Closes an image file that was open for reading or writing. All output is flushed to the output file, and the output file is closed.

Reading and Writing an Image File

The following functions allow pixels to be transferred to and from an image file. These functions provide an interface to an image file that is independent of whether the image file happens to be run length encoded, and independent of whether it maintains 1 or 2 bytes per pixel.

putrow (image, buffer, y, z) register IMAGE *image; unsigned short *buffer; unsigned y, z;

Writes a row of pixels to the specified image file. The buffer should be an array of shorts that contain the pixel values of a colormap image or one of the 3 channels of an rgb image. If the image file maintains only one byte per pixel, then the values passed in the buffer should be in the range 0..255. The row of the image to be written is given by y, while z selects which channel of the image to write to. The first channel of the image is channel 0. A black and white image will have only 1 channel while rgb images have 3 channels. In an rgb image, channel 0 is used to store red while channel 1 stores green, and channel 2 stores blue pixel data. The y argument should be greater than or equal to zero and less than the ysize of the image. The rows of the image file may be written in any order.

getrow (image, buffer, y, z) register IMAGE *image; unsigned short *buffer; register unsigned int y, z;

Reads a row of pixels from the specified image file. The buffer should be an array of shorts to receive pixel values of a colormap image or one of the 3 channels of an rgb image. The row of the image to be read is given by y, while z selects which channel of the image to read from. The first channel of a image is channel 0. A black and white image will have only 1 channel, while an rgb image will have 3. The y argument should be greater than or equal to zero and less than the ysize of the image. The rows of the image file may be read in any order.

Miscellaneous Functions

isetname (image, name) IMAGE *image; char *name;

Copies the character string name into the name field of the image file. NOTE: handling names when processing two files together is not well supported and is not encouraged.

isetcolormap (image, colormap) IMAGE *image; int colormap;

Tells ipaste and some printing utilities whether the pixel values should be interpreted as color-index pixels or intensities. A gray scale image consists of one channel of intensities, while an rgb image has three independent channels of pixel intensities, one channel for each red, green and blue intensities. The argument colormap may be one of following three values: CM_NORMAL is the default indicating that the pixels are intensity values. 0 is black and a value of 255 in the image is white. Black and white images and rgb images are stored with CM_NORMAL. CM_SCREEN indicates that the pixels were copied from the screen and must be transformed by a color map to be meaningful. Colormaps can also be stored in image files. CM_COLORMAP means that the pixels in the image file represent a color map

An Example

The following example shows how to open an image file and read its contents. More examples may be found in /usr/people/4Dgifts/iristools/imgtools.

```
/*
     *      readimage - Read an image file and print its pixel values.
     *
     *      To compile:  cc readimage.c -o readimage -limage
     *
     *                              Paul Haeberli - 1991
     */
    #include <gl/image.h>

    main(argc,argv)
    int argc;
    char **argv;
    {
        IMAGE *image;
        int x, y, z;
        short *rbuf, *gbuf, *bbuf;

    /* print usage message */
        if( argc<2 ) {
            fprintf(stderr,"usage: readimage infile0);
            exit(1);
        }
    /* open the image file */
        if( (image=iopen(argv[1],"r")) == NULL ) {
            fprintf(stderr,"readimage: can’t open input file %s0,argv[1]);
            exit(1);
        }

    /* print a little info about the image */
        printf("Image x and y size in pixels: %d
    %d0,image->xsize,image->ysize);
        printf("Image zsize in channels: %d0,image->zsize);
        printf("Image pixel min and max: %d %d0,image->min,image-max);

    /* allocate buffers for image data */
        rbuf = (short *)malloc(image->xsize*sizeof(short));
        gbuf = (short *)malloc(image->xsize*sizeof(short));
        bbuf = (short *)malloc(image->xsize*sizeof(short));

    /* check to see if the image is B/W or RGB */
        if(image->zsize == 1) {
            printf("This is a black and write image0);
            for(y=0; y<image->ysize; y++) {
                getrow(image,rbuf,y,0);
                printf("row %d: ",y);
                for(x=0; x<image->xsize; x++)
                    printf("%d |",rbuf[x]);
                printf("0);
            }
        } else if(image->zsize >= 3) {  /* if the image has alpha zsize is 4
            printf("This is a rgb image0);
            for(y=0; y<image->ysize; y++) {
                getrow(image,rbuf,y,0);
                getrow(image,gbuf,y,1);
                getrow(image,bbuf,y,2);
                printf("row %d: ",y);
                for(x=0; x<image->xsize; x++)
                    printf("%d %d %d |",rbuf[x],gbuf[x],bbuf[x]);
                printf("0);
            }
        }
    }
```

BUGS

There are too many video image file formats.

NAME

intro - introduction to games and demos

DESCRIPTION

The manual pages with the section 6 suffix are for recreational and educational programs normally found in the directory **/usr/games**; however, Silicon Graphics currently does not ship any such programs, so this directory does not exist. The manual pages with the suffix 6D are the demonstration programs found in the directory **/usr/demos/General_Demos, /usr/demos/O2** or **/usr/demos/OCTANE** Beginning with the 6.3 release of IRIX, these demos can be accessed from several mechanisms: - from a web-based interface, which can be started using the toolchest Find; Demos menu option - from the system iconcatalog, which can be started using the toolchest Find; Iconcatalog; Demos - from buttonfly (/usr/demos/buttonfly) - and directly from the directories /usr/demos/General_Demos, /usr/demos/O2 and /usr/demos/OCTANE Both buttonfly and the directory noted above can be accessed using the Find; File QuickFind menu option or Selected; File QuickFind menu option available from the system toolchest The availability of these programs and the data they use will vary from system to system and from release to release.

BUGS

The following error message (or something similiar) may appear when exiting out of the demos improperly: XIO: fatal IO error 131 (Connection reset by peer) on X server "localhost:0.0" after 44 requests (44 known processed) with 0 events remaining. Ignore this error message. The system has not been affected in any way. The proper way to exit out of demos is by : 1) typing the ESC key or 2) clicking on the right mouse button and selecting Exit from the menu For more information, please read the man pages for a particular demo.

SEE ALSO

buttonfly(6D).

NAME

abs - get the absolute value of an image

SYNOPSIS

abs inimage outimage

DESCRIPTION

abs generates outimage which contains the absolute value of the input image. This treats input values as signed 8-bit quantities. Input values of 0 map to white. 255 also maps to white. 128 maps into black.

NAME

add - add two images together

SYNOPSIS

add inimage1 inimage2 outimage

DESCRIPTION

add takes inimage1 and inimage2's intensities and adds them together and stores the result in outimage. If the sum is greater than 255 the result is set to 255.

NAME

addborder - surround an input image with a border image

SYNOPSIS

addborder inimage borderimage outimage

DESCRIPTION

addborder Surrounds the input image with the border image. The four quadrants of the border image are placed at the four corners of the input image, and the edges are extended to connect the corners surrounding the border image. Three existing borber image files can be found in ˜4Dgifts/iristools/images/{cutline.bw, regmark.bw, shadow.bw}.

SEE ALSO

addframe(6D)

NAME

addframe - add a border to an image

SYNOPSIS

addframe inimage outimage [width [r g b]]

DESCRIPTION

addframe adds a constant width border to a monochrome or color image. The default width is 1 pixel, and the default color is black.

NAME

addnoise - add noise to an image

SYNOPSIS

addnoise inmage outimage noiseimage mag

DESCRIPTION

addnoise adds noise from the noise image to the input image. The noise image tiles the input image. The magnitude of the noise is scaled by a floating point magnitude mag. It's range is [0.0 ... 1.0]. 0.0 adds no noise into outimage, while 1.0 add alot of noise. Typicaly *randimg* is used to create a noise pattern (*histeq* can also be used).

SEE ALSO

randimg(6D), histeq(6D)

NAME

assemble - assemble an array of smaller images

SYNOPSIS

assemble nx ny outimage imgfiles...

DESCRIPTION

assemble assembles an nx by ny array of smaller images. The catch here is that all images being assembled have to have the same x/y dimensions. The nx by ny array works like this: nx is the number of images that are going to be sitting side by side in the horizontal direction, and ny is the number of images side by side in the vertical direction. Order imgfiles so that the first image will be the one sitting in the bottom left-hand corner of outimage, with the second sitting either above it or to its right. If fewer than nx times ny images are given on the command line, the last image is used repeatedly.

NAME

bgpaste - paste an image onto the root window

SYNOPSIS

bgpaste [-t r g b] [-o xorig yorig] inimage bgpaste [-t r g b] -n numimgs xorg yorg img [xorg yorg img . . .]

DESCRIPTION

bgpaste works in one of two distinct ways: either feed it one image, which by default is automatically centered (or explicitly specify the image's origin), or feed it a list of one or more images preceeded by their respective x/y origins to be painted as a composite in the root window. In the first case, *bgpaste* centers and pastes inimage onto the root window regardless of inimage's size. You can override the "centering" default and explicitly position your own image manually by using the *-o* option to specify the image's absolute- screenspace origin as measured from the bottom-left corner of the graphics display screen. The *xorig*,*yorig* pair is defined in terms of the image's bottom-left corner. Negative values are legal to specify and will "plant" the image's origin offscreen to the left and/or the bottom of the screen origin. In the second case, the *-n* flag tells *bgpaste* you are including a list of *n* images, where each image is preceeded by its respective x/y origin pair (again, negative values are legal). The intersection of the composite set of images with that of the root window size is calculated and a buffer of that size is allocated which will contain the composite. The images will be painted in the order enumerated: the first image listed will be painted first, the last will be painted last. Any "background" within the composite, as well as any remaining area of the root window not included in the intersection, will be painted with the default gray color or one defined using the *-t* flag. In either of the above descriptions, if either the X or the Y size (or both) of inimage/composite is smaller than the screen size, a gray background is painted where the image/composite doesn't appear. This gray default color can be redefined to be a specific RGB triplet using the *-t* option followed by the *r g b* integer triplet. Use izoom(6D) if you wish to blowup an inmage that is smaller than the screen's X and/or Y size.

NOTE

When employing *bgpaste -n* performance will degrade because of the creation of the buffer which stores the composite root window image.

SEE ALSO

izoom(6D)

NAME

blend - linearly interporlate two images

SYNOPSIS

blend inimage1 inimage2 outimage param

DESCRIPTION

blend linearly interpolates two images. param is a floating point number that controls the weighting of the two input images. A value of 0.0 will use only imimage1, while a value of 1.0 will use only inimage2. Values outside the range [0.0...1.0] will extrapolate instead of interpolate between the two images.

NAME

blur - low pass filter an image

SYNOPSIS

blur inimage outimage [pix] or [xpix ypix]

DESCRIPTION

blur low pass filters the input image. This is done by izooming the image down and then izooming it back up with filtering. The amount of blurring is controlled by pix or xpix and ypix. A pix value of 10 will cause the blur to zoom the image down until it is about 10 pixels across, and then zoom it back up to the original size. Large pix values end up making the result less blurry.

SEE ALSO

convolve(6D)

NAME

btree - display an image using a binary tree ordering.

SYNOPSIS

btree inimage [depth]

DESCRIPTION

btree displays an image using a progressive technique. First it is drawn as a 2 by 2 array of rectangles, then rectangles are added to make it a 4 by 4 array. The resolution is increased in this way to the default level which is depth 8. To manually set the level of resolution to be other than 8, specify depth as some integer value.

NAME

cglue - create an rgb image out of 3 black and white images

SYNOPSIS

cglue red.bw grn.bw blu.bw outimage.rgb

DESCRIPTION

cglue constructs an rgb image by combining three black and white (bw) images. These black and white images are treated as intensity maps for the red, green, and blue channels of the rgb generated image.

NOTE

This program requires that all the source images on the command line have the same x/y size.

NAME

conimg - create a constant image

SYNOPSIS

conimg outimage xsize ysize [bwval] [r g b]

DESCRIPTION

conimg creates an image with a constant color. If only one value is specified, a black and white (one channel) image is created. If three values are specified, a color image is created. If no color value is provided, a black, one channel image is created. The image values are given in the range 0..255.

NAME

convolve - convolve an input image with a kernel

SYNOPSIS

convolve inimage outimage kernelimage [-m -d]

DESCRIPTION

convolve convolves the input image with a kernel image. Kernel images can be created with the program greyscale. To do a convolution, the kernel image is stepped across the surface of the input image. At each position, all the image values under the kernel are multiplied by the corresponding kernel values. The sum of all these products is divided by the sum of all the values in the kernel. The result is put into the output image. If the -m option is given, the maximum value of all the multiplies is put into the output image instead of the normalized sum of the products. If the -d option is given, a delta value is calculated wherever the kernel is greater than 128.

SEE ALSO

blur(6D)

NAME

cscale - scale the rgb colors of an image

SYNOPSIS

cscale inimage outimage.rgb r g b [-d]

DESCRIPTION

cscale can be used to color balance an image by scaling the r, g, and b channels by different factors. Each scale factor is an integer in the range 0 to 255. A scale factor of 0 is interpreted to mean scale by 0.0, while a scale factor of 255 means scale by 1.0. Each scale factor multiplies a color channel of the source image. If the -d option is given, the input colors are divided by the given factors.

NAME

dotgen - make an image of two crossed sinusoidal wave patterns

SYNOPSIS

dotgen outimage size scalex scaley [-a]

DESCRIPTION

dotgen generates an image of two crossed sinusoidal wave patterns. The output image can be used as input to thresh to create halftoned images. The output of *dotgen* is a square image size pixels on a side. scalex and scaley are floating point values that can be used to change the density and rotation of the dot pattern. If they are both integers, the output image will match up with its self when it tiles the plane. The **-a** option causes *dotgen* to take the absolute value of the sum of the two sinusoids.

SEE ALSO

thresh(6D)

NAME

duotone - make a color duotone image from a black and white image

SYNOPSIS

duotone inimage.bw outimage.rgb r g b

DESCRIPTION

duotone creates a duotoned color image from a single channel image. The r g b arguments specify the color that a 50 percent gray value in the source image should become in the duotone image. Try experimenting with “doutone in.bw out.rgb 200 120 80”.

SEE ALSO

tobw(6D)

NAME

fieldmerge - merge two field images into one frame

SYNOPSIS

fieldmerge inimage0 inimage1 outimage

DESCRIPTION

Standard NTSC Video displays 30 frames per second. Each frame is composed of two fields that are interlaced. When recording video animations it is sometimes good to record 60 fields per second, but to do this two fields must be merged together into a single frame by interleaving scanlines from two images. *fieldmerge* interleaves two images in this way.

NOTE

This program requires that all the source images on the command line have the same x/y size.

NAME

fitimg - force an image to be a specific size.

SYNOPSIS

fitimg inimage outimage xsize ysize

DESCRIPTION

fitimg uniformly scales a picture to a specific size. This is accomplished by scaling the image to be smaller than the specified size, and surrounding it by a white border as needed to make the final image exactly xsize by ysize pixels. The aspect ratio of the source image is preserved. This is useful for making an array of images using “assemble”.

SEE ALSO

assemble(6D)

NAME

fromalias - converts an Alias image to an Iris image

SYNOPSIS

fromalias aliasimage outimage.rgb

DESCRIPTION

fromalias converts an Alias image to an IRIS image file image.

SEE ALSO

toalias(6D)

NAME

frombin - create an RGB Iris image file from a binary dump of image data

SYNOPSIS

frombin image.bin outimage.rgb xsize ysize [zsize]

DESCRIPTION

frombin reads a binary dump of some image data, and creates an IRIS image file. If only xsize and ysize are given, then a single channel black and white image is created. The first byte of the input file becomes the lower left pixel in the resulting image. If a zsize of 3 is given, a color image is created, by first reading all the red band, followed by the green and blue bands.

SEE ALSO

tobin(6D)

NAME

fromcmap - convert a color map into an image with one scanline

SYNOPSIS

fromcmap colors.map outimage.rgb

DESCRIPTION

fromcmap converts a color map into an RGB image with scanline. The width of the image will be the number of colors in the color map. Each pixel will contain the RGB color of the corresponding color map entry.

NAME

fromcube - convert a Cubicomp/Vertigo image file to IRIS format

SYNOPSIS

fromcube cubicomp.pic outimage.rgb

DESCRIPTION

fromcube converts a Cubicomp/Vertigo image file into an Iris image file format.

NAME

fromdi - convert an old .di dithered image into an RGB image

SYNOPSIS

fromdi colors.map outimage.rgb

DESCRIPTION

fromdi converts an old .di dithered (8-bits) image into an RGB image. *ipaste* works differently on 4D machines than it originally did on 2000/3000 IRIS machines. While it is true that on the 2000/3000 machines image files made by the program dither (making an 8-bit image) would work with *ipaste* , on the 4D's *ipaste* was written to operate on RGB (24-bit) images or SCREEN (16-bit starting at zero) images but not with DITHERED (8-bits starting at some offset) images. Refer to the Note in the ipaste(1G) man page stating that if one REALLY wants to view their DITHERED images on a 4D, they will first need to use *fromdi* to convert them from DITHERED into RGB format, and then can use *ipaste* to display them.

SEE ALSO

ipaste(1G)

NAME

fromface - convert a UNIX faceserver image into IRIS format

SYNOPSIS

fromface in.face out.bw

DESCRIPTION

fromface converts a UNIX faceserver image file into IRIS image file format.

NAME

fromgif - convert a GIF image into an IRIS image

SYNOPSIS

fromgif inimage.gif outimage.rgb

DESCRIPTION

fromgif converts a Compuserve GIF image file into an IRIS image file.

SEE ALSO

togif(6D)

NAME

frommac - convert a MacPaint image into an IRIS image

SYNOPSIS

frommac inimage.mac outimage.bw

DESCRIPTION

frommac converts a MacPaint image into a black and white image IRIS file.

SEE ALSO

tomac(6D)

NAME

frompic - convert a MOVIE BYU .PIC image to an IRIS image

SYNOPSIS

frompic inimage.PIC outimage.rgb

DESCRIPTION

frompic converts a MOVIE BYU .PIC image file to IRIS image file format.

NAME

fromppm - convert an image in Jef Poskanzer’s format into an IRIS image

SYNOPSIS

fromppm inimage.ppm outimage.rgb

DESCRIPTION

fromppm converts a PPM image file to an IRIS image file. PBMPLUS, is by Jef Poskanzer. It is a comprehensive format conversion and image manipulation package. The latest version is always available via anonymous FTP as expo.lcs.mit.edu:contrib/pbmplus.tar.Z and ftp.ee.lbl.gov:pbmplus.tar.Z

SEE ALSO

toppm(6D)

NAME

fromrla - converts a Wavefront image to an IRIS image

SYNOPSIS

fromrla inimage.rla outimage.rgb

DESCRIPTION

fromrla converts a Wavefront .rla image file into an rgb IRIS image file.

NAME

fromsun - convert a sun image into an IRIS image

SYNOPSIS

fromsun inimage.ras outimage.rgb

DESCRIPTION

fromsun converts any type of SUN rasterfile image into an IRIS image file.

SEE ALSO

tosun(6D)

NAME

fromtarga - convert a targa image into an IRIS image

SYNOPSIS

fromtarga inimage.tga outimage.rgb

DESCRIPTION

fromtarga converts a type 2 RGB TARGA image into an IRIS image. Most targa images are displayed directly on monitors with no gamma correction. The typical gamma is about 2.2, so you need to gammawarp the output image by 2.2 to get it into a linear intensity space.

SEE ALSO

gammawarp(6D), totarga(6D)

NAME

fromxbm - convert an X Bitmap image into an IRIS image

SYNOPSIS

fromxbm xbitmap outimage.bw

DESCRIPTION

fromxbm converts an X Bitmap file into an IRIS image file.

NAME

fromxwd - convert an xwd file into an IRIS image

SYNOPSIS

fromxwd inimage.xwd outimage.rgb

DESCRIPTION

fromxwd converts an xwd file to IRIS image file format.

NAME

fromyuv - convert an Abekas yuv image into an IRIS image

SYNOPSIS

fromyuv inimage.yuv outimage.rgb [-s]

DESCRIPTION

fromyuv converts an digital video image in Abekas yuv format into IRIS image file format. This will normally create an image that is 720 by 486 pixels. This image will have non-square pixels. If square pixels are desired, use the “-s” option. This will create an IRIS image that is 640 by 486 pixels.

SEE ALSO

toyuv(6D)

NAME

gammawarp - lighten or darken an image

SYNOPSIS

gammawarp inimage outimage gamma

DESCRIPTION

gammawarp lightens or darkens an image by changing the gamma. A gamma value that is less than 1.0 will lighten grey tones, while a gamma value that is greater that 1.0 will darken grey tones.

SEE ALSO

gamcal(6D), gamma(6D)

NAME

gendit - perform general image dithering

SYNOPSIS

gendit inimage.rgb outimage.rgb nr ng nb [-s]

DESCRIPTION

gendit uses an ordered dither to create an image with fewer colors. The argument nr selects how many levels or red to use, while ng, and nb select the number of green and blue levels respectively. The minimum value for nr, ng, or nb is 2. The total number of colors used will be nr*ng*nb. The “-s” option is used to offset the dither patterns used for red, green and blue by 2 pixels.

NAME

greyscale - make different patterns

SYNOPSIS

greyscale outimage.bw xsize ysize patternno

DESCRIPTION

greyscale generates a variety of different patterns. A monochrome image is created that is xsize by ysize pixels in size. patternno selects what kind of image is created. Patternno Description ____ 0 ramp [0...255] in the X direction 1 ramp [0...255] in the y direction 2 circular ramp pattern 3 all white image of 255 4 horizontail stripes white and black 5 All black except 1% of pixels are in the range [128...255] 6 black and white checker board pattern 7 grid pattern 8 gamma test pattern 9 gaussian in the x direction 10 gaussian in the y direction 11 contrast test image 12 all black image 13 all white image 14 random noise image 15 circle 16 rolf 17 sinusoidal line test image 18 Bayer threshold image for dithering

SEE ALSO

gamcal(6D), gamma(6D)

NAME

halftone - half-tone an image

SYNOPSIS

halftone inimage outimage freq angle [patternno]

DESCRIPTION

halftone converts the input image into a halftoned image, using Hollaway's technique. The density of the halftone screen is controlled by the floating point value freq. Smaller values of "freq" make a larger dot pattern. The angle of the halftone patterns is controlled by the argument "angle". The integer argument "patternno" selects the halftone pattern. 0 is the default and creates a dot pattern. A pattern value of 1 will create a line screen, while 2, 3, 4, and 5 create other halftone patterns.

NAME

hipass3 - high pass filter an image

SYNOPSIS

hipass3 inimage outimage mag

DESCRIPTION

hipass3 performs a high pass filter on an image using a 3 by 3 filter kernel. mag specifies how much to increase the high frequencies. A value of 0.0 will not change the image. 1.0 will significantly increase the high frequencies in the image.

NAME

hist - compute and display the histogram of an image file.

SYNOPSIS

/usr/sbin/hist inimage

DESCRIPTION

hist reads an image file specified by the user, then computes and displays the histogram of the image file. The red hash-marks indicate the boundaries of the range of intensities for a given image. The black line starting in the bottom left corner and going up to the top right is the indicator of the distribution function of the histogram. Pressing LEFTMOUSE will print the pixel value currently under the cursor.

NAME

histeq - histogram equalize an image file

SYNOPSIS

histeq inimage outimage [-r]

DESCRIPTION

histeq performs histogram equalization on an image file. Thie -r option causes noise to be added to reduce quatization artifacts.

SEE ALSO

hist(6D), imgexp(6D)

NAME

iavg - average a set of images

SYNOPSIS

iavg outimage imgfiles . . .

DESCRIPTION

iavg averages a set of images. All the input images must be the same xsize and ysize in pixels.

NAME

iblend - blend two images using a mat

SYNOPSIS

iblend inimage1 inimage2 outimage matimg.bw

DESCRIPTION

iblend blends between inimage1 and inimage2 to create outimage. Pixel values from the image matimg.bw are used to select how much of inimage1 and inimage2 to use for each pixel. The output image will be the minimun of the input image sizes.

SEE ALSO

blend(6D)

NAME

iflip - flip an image

SYNOPSIS

iflip inimage outimage [x y xy yx 90 180 or 270]

DESCRIPTION

iflip flips an image in the following ways: either in the **x** or **y** direction, with **xy** the upper-left and lower-right corners are flipped, with **yx** the lower-left and upper-right corners are flipped, or rotates an image 90, 180, or 270 degrees in a clockwise direction.

NAME

imean - find the average pixel value of an image

SYNOPSIS

imean inimage

DESCRIPTION

imean finds the average pixel value of inimage. imean prints the average RGB color of all the pixels in the image to stdout.

SEE ALSO

hist(6D)

NAME

imgexp - expand the range of pixel values in an image.

SYNOPSIS

/usr/sbin/imgexp inimage outimage [min max]

DESCRIPTION

imgexp expands the range of pixel values in an image. Pixel values less than or equal to min are mapped to 0, while pixel values greater than or equal to max are mapped to 255. If min and max are not provided on the command line, then the minimum and maximum pixel values in the image are used. This can be used to manipulate the contrast of images.

SEE ALSO

hist(6D)

NAME

imgsize - print the size of an image

SYNOPSIS

imgsize inimage [-2]

DESCRIPTION

imgsize prints the xsize, ysize, and zsize of the named image. If the -2 option is given, only the xsize and ysize are printed. This is useful in shell scripts that perform a sequence of image processing operations.

SEE ALSO

istat(6D)

NAME

imgwrap - shift pixels left one bit.

SYNOPSIS

imgwrap inimage outimage

DESCRIPTION

imgwrap shifts image pixel values left one bit.

NAME

intro - introduction to games and demos

DESCRIPTION

The manual pages with the section 6 suffix are for recreational and educational programs normally found in the directory **/usr/games**; however, Silicon Graphics currently does not ship any such programs, so this directory does not exist. The manual pages with the suffix 6D are the demonstration programs found in the directory **/usr/demos/General_Demos, /usr/demos/O2** or **/usr/demos/OCTANE** Beginning with the 6.3 release of IRIX, these demos can be accessed from several mechanisms: - from a web-based interface, which can be started using the toolchest Find; Demos menu option - from the system iconcatalog, which can be started using the toolchest Find; Iconcatalog; Demos - from buttonfly (/usr/demos/buttonfly) - and directly from the directories /usr/demos/General_Demos, /usr/demos/O2 and /usr/demos/OCTANE Both buttonfly and the directory noted above can be accessed using the Find; File QuickFind menu option or Selected; File QuickFind menu option available from the system toolchest The availability of these programs and the data they use will vary from system to system and from release to release.

BUGS

The following error message (or something similiar) may appear when exiting out of the demos improperly: XIO: fatal IO error 131 (Connection reset by peer) on X server "localhost:0.0" after 44 requests (44 known processed) with 0 events remaining. Ignore this error message. The system has not been affected in any way. The proper way to exit out of demos is by : 1) typing the ESC key or 2) clicking on the right mouse button and selecting Exit from the menu For more information, please read the man pages for a particular demo.

SEE ALSO

buttonfly(6D).

NAME

invert - invert an image

SYNOPSIS

invert inimage outimage

DESCRIPTION

invert inverts an image. pixel values of 0 become 255 and pixel values of 255 map to 0. The result is to invert the tonal scale of an image.

NAME

iroll - roll an image in x and y directions

SYNOPSIS

iroll inimage outimage xroll yroll

DESCRIPTION

iroll rotationaly rolls an image. The integers xroll and yroll specify how many pixels to roll the image in the x direction or the y directon.

NAME

iset - set the type of an image.

SYNOPSIS

/usr/sbin/iset newtype imgfiles

DESCRIPTION

iset sets the type of an image. This determines which part of the color map ipaste uses to display the image. The four types of viewable images are NORMAL, DITHERED, SCREEN, and COLORMAP. These are the four values newtype can have (each must be spelled in all capital letters as above). A NORMAL image is an RGB or monochrome image. A DITHERED image is a color image using only 8 bits to represent the original 24-bit true RGB image. This image type is obsolete on 4D machines. A SCREEN image contains color indexes. A COLORMAP image is used to store colormaps.

SEE ALSO

istat(6D), rle(6D), verbatim(6D)

NAME

istat - print the header information of a list of image files.

SYNOPSIS

/usr/sbin/istat imagefiles . . .

DESCRIPTION

istat prints the header information of a list of image files. x/ysize give the image's screen size in pixels; zsize is the number of channels in the image. An RGB image will typically have three channels, while a Monochrome image will use one channel. Min and max are the range of pixel intensity values in the image. Bpp describes how many bytes are stored in each channel of the image; either 1 byte or 2 bytes. Type of image can be NORMAL, DITHERED, SCREEN, or COLORMAP. Storage refers to the way the data is compressed: rle is a run-length encoded image, verb is a verbatim image which means the data is not compressed in any way.

SEE ALSO

iset(6D), rle(6D), verbatim(6D)

NAME

izoom - magnify or shrink an image

SYNOPSIS

/usr/sbin/izoom inimage outimage xscale yscale [-i -b -t -q -m or -g] [-w blurfactor]

DESCRIPTION

izoom magnifies or shrinks an image with or without filtering. xscale and yscale are floating point scale factors. The filtering method is one pass, uses 2-d convolution, and is optimized by integer arithmetic and precomputation of filter coefficients. Normally *izoom* uses a triangle filter kernel in both x and y directions. The **-i** (impulse) option causes izoom to do no filtering as the image is resized. The **-b** (box) option causes izoom to use a box as the filter kernel. The **-t** (triangle) option is the default. The **-q** (quadratic) option indicates that a quadratic function should be used as the filter kernel. The **-m** option uses a Mitchell kernel and the **-g** option uses a Gaussian kernel. The **-w blurfactor** option specifies the width of the reconstruction filter. This will effect how blurry the resulting image is. If you want more blur use a larger number. The default value is 1.0. NOTE: *izoom* does not work on dithered images which are nothing more than color look-up table indices. To perform any such image processing one must first use something like *fromdi* (in the moregltools subsystem, and also can be found in */usr/people/4Dgifts/iristools/imgtools/fromdi.c*), which converts the dithered image into an RGB image.

NAME

loadmap - loads the colormap from a file

SYNOPSIS

loadmap file.map

DESCRIPTION

loadmap loads the current contents of the color map from a file. The color indices and color map entries must be in a file written by *savemap*(1G).

SEE ALSO

makemap(1g), savemap(1g)

NAME

mapimg - translates a screen image into an RGB image

SYNOPSIS

mapimg image.sc image.rgb temp.map

DESCRIPTION

mapimg translates a screen image into a full RGB image using the given color map. The color map is usually generated by *savemap*.

NAME

max - get the maximum of two images

SYNOPSIS

max inimage1 inimage2 outimage

DESCRIPTION

max calculates the maximum of two images and puts the result in outimage.

NAME

min - calculate the minimum of two images

SYNOPSIS

min inimage1 inimage2 outimage

DESCRIPTION

min calculates the minimum of two images and puts the results in outimage.

SEE ALSO

max(6D)

NAME

movie - show a series of images in a sequence

SYNOPSIS

movie images . . . [-sx]

DESCRIPTION

movie shows a series of images as a movie. The -sx option causes the movie to be surounded by a white frame. This program requires all the source images on the command line have the same x/y size.

NAME

mult - multiply two images

SYNOPSIS

mult inimage1 inimage2 outimage

DESCRIPTION

mult multiplies two images. The value 255 is used to represent 1.0 while 0 represents 0.0.

NAME

noblack - remove all the black from an image

SYNOPSIS

noblack inimage.rgb outimage.rgb

DESCRIPTION

noblack scales each pixel in an RGB image so at least one of its components is 255. This has the effect of removing all the black from an image.

NAME

oneband - get a single band of an image

SYNOPSIS

oneband inimage.rgb outimage.bw band

DESCRIPTION

oneband extracts a single band of a color image. Normally band should be a number in the range [0...3]. Band 0 is the red part of and RGB image, while band 1 and 2 are the green and blue bands. Band 3 potentially represents alpha.

SEE ALSO

cglue(6D)

NAME

over - put one image on top of another

SYNOPSIS

over underimage overimage outimage xpos ypos

DESCRIPTION

over places one image over (on top of) another image. xpos and ypos specify the position on underimage that the overimage’s origin will be placed at.

SEE ALSO

assemble(6D), subimg(6D)

NAME

perhist - print percent histogram values for an image

SYNOPSIS

perhist inimage [minpercent maxpercent] [-3]

DESCRIPTION

perhist analyzes the histogram of an image and prints two pixel values. 1 percent of the pixels in the image will less than the first pixel value printed, while 99 percent of the pixels will be less than the second value printed. These two pixel values can be used as arguments to “imgexp” to increase the contrast of an image with out loosing too much detail in the shadows or the highlights. The default min percent and max percent values are 1 perecnt and 99 percent by default, but these values can be changed by specifying them on the command line. The “-3” option causes the percent histogram values to be printed independently for each channel of a color image.

SEE ALSO

imgexp(6D), hist(6D)

NAME

postcard - make an image look like a postcard

SYNOPSIS

postcard inimage outimage

DESCRIPTION

postcard creates a postcard image by adding a black frame to the input image and placing this on a white rectangle that has the exact same aspect ratio as a standard postcard.

NAME

quant - quantify an image

SYNOPSIS

quant inimage outimage nlevels

DESCRIPTION

quant quantifies an image to have n levels. The output image will only have nlevels different pixel values.

SEE ALSO

thresh(6D)

NAME

randimg - generate a noise image

SYNOPSIS

randimg outimage xsize ysize [seed]

DESCRIPTION

randimg makes an image of random noise. Seed initializes the random number generator. A seed value of 0 makes *randimg* use its process id as the seed.

SEE ALSO

addnoise(6D)

NAME

rectimg - display a color or BW image on the iris

SYNOPSIS

rectimg inimage

DESCRIPTION

rectimg displays a color or black and white image on the iris. This program is a simple version intended primarily for demo use. This will only work on machines that support RGB mode.

NAME

repcolor - replace specified colors within an image

SYNOPSIS

repcolor inimage.rgb outimage.rgb ir ig ib or og ob dist

DESCRIPTION

repcolor replaces colors in the input image that are within dist of ir ig ib within the color or og ob. The arguments ir, ig, ib, or, og, and ob, and dist should be values in the range of 0 to 255.

NAME

rle - force an image to be stored using run length encoding

SYNOPSIS

/usr/sbin/rle inimage outimage

DESCRIPTION

Sometimes images are stored with no compression in verbatim format. *rle* converts an image to be stored using run length encoding.

SEE ALSO

iset(6D), istat(6D), verbatim(6D)

NAME

saturate - change an image’s saturation

SYNOPSIS

saturate inimage.rgb outimage.rgb satval

DESCRIPTION

saturate changes the saturation of an image. A satval of 0.0 will make the image black and white (no color). A satval of 1.0 will leave the image unchanged. A satval of 2.0 will double the amount of color in an image.

SEE ALSO

tobw(6D)

NAME

scope - explore an image of any size

SYNOPSIS

scope inimage

DESCRIPTION

scope allows viewing an image of any size. This tool can be very useful particularly with images larger than 1280x1024 pixels. The image appears on the right hand side of the window, and a blowup of it is on the left initially focused at the image’s center. “xpos” and “ypos” indicate the pixel location of the white square box, “color” gives the red, green, and blue values at that location, and “zoom” indicates the zoom factor (its’ default is 10). By moving the mouse over the image on the right and pressing LEFTMOUSE, you can change the window of focus on the right. The up and down arrow keys zoom out and in respectively.

NAME

scrsave - save a part of the screen in an image file

SYNOPSIS

/usr/sbin/scrsave outimage.rgb [x1 x2 y1 y2] -b

DESCRIPTION

scrsave saves a portion of the screen into either an RGB or a Black and White image file. *scrsave* with no other arguments than the output filename will save the contents of the entire screen to an IRIS image file. By giving additional arguments a smaller region of the screen can be saved. *scrsave dump.rgb 0 99 0 99* saves a 100 pixel by 100 pixel square in the lower left hand corner of the screen. When invoked, the **-b** option will save the screen data as a Black and White (1 byte per pixel deep) image rather than as an RGB (3 bytes per pixel deep) image.

SEE ALSO

snapshot(6D)

NAME

setlum - modifies the luminance on an image

SYNOPSIS

setlum inimage.rgb inimage.bw outimage.rgb

DESCRIPTION

setlum modifies the luminance (brightness) of each pixel in the first input image (inimage.rgb) to be the same as the pixel value in inimage.bw.

NAME

shear - shear an image diagonally

SYNOPSIS

shear inimage outimage slope

DESCRIPTION

shear diagonally shears an image. If slope is 1.0, the shear will make vertical lines into 45 degree diagonal lines. If slope is 0.5 the left edge of the image gets transformed into a line that goes from the lower left-hand corner into the middle of the top edge.

NAME

slide - zoom an image up for full screen display

SYNOPSIS

slide inimage [backno]

DESCRIPTION

slide zoom an image up to the full screen size for display. This is useful when creating slides by photographing the screen directly. The option backno is used to select the color that the screen should be cleared to before drawing the image. backno can be one of three values, 0, 1, or 2. A value of 0 will draw a black background, a value of 1 draws a white background, and a value of 2 makes the background the same color as the lower lefthand pixel in the image.

SEE ALSO

izoom(6D), grid(6D)

NAME

snapshot - save a portion of the screen in an image file

SYNOPSIS

/usr/sbin/snapshot [-b]

DESCRIPTION

snapshot reads an area of the screen specified by the user, and saves it in an image file. To use *snapshot*, place the *snapshot* button window someplace other than where you wish to grab. Then, with the input focus attached (i.e. the mouse is inside the *snapshot* window), hold down a modifier key (shift, ctrl) on the keyboard to maintain the input focus, and move the mouse to one of the four corners of the section of the screen you wish to save. Now press **left** mouse and continue holding it down while you stretch out a red rubberband to the opposite corner of the area of interest. To tell *snapshot* to make the image file, go back to the *snapshot* window, press the **right** mouse and choose one of the two “Save” menu items. You can repeat this sequence in various ways until such time as you wish to exit. At this point, you can choose one of the two exit menu items with the **right** mouse. To move the *snapshot* window itself, use your favorite window manager accelerator functions, such as ALT+F7. **Leftmouse functionality** The **left** mouse button stretches, reshapes, moves or starts an entirely new rubberband for you. The cursor is the constant visual indicator of what will happen if you press **left** mouse. As long as your input focus is directed to *snapshot* you will see one of 4 different cursor types depending on the location of the mouse: **camera cursor -** will appear when you are on top of any area of the console screen other than on the sides or inside of the rubberband area of interest. **corner cursor -** will appear when you are in the immediate vicinity of one of the 4 corners of the currently placed rubberband. **horizontal/vertical cursor -** will appear when you are in the immediate vicinity of one of the 4 sides of the currently placed rubberband. **move cursor -** will appear when you are fully inside the rubberband area. When your cursor is anywhere other than on top of the *snapshot* window, whichever of the four cursors you see will tell you what will happen at that point if you press the **left** mouse button: if you see the **camera** cursor this means that by pressing the **left** mouse, you will start creating a new rubberband that you can stretch out in any direction which will stop when you let go of the mouse button; when you see either the **horizontal**, **vertical**, or **corner** cursors this means that pressing **left** mouse at this time will enable you to stretch the corner or side of interest and continue doing so until you release the mouse button; when the **move** cursor is visible (while inside of the rubberband), pressing **left** mouse at this point enables you to move the entire rubberband in its current shape and size until you let go of the mouse. When you see the **move** cursor, you may also press **middle** mouse to move the rubberband. To pop the *snapshot* button window, press down the **left** mouse button while your cursor is on top of the window, and release it without moving more than one pixel in any direction. **Pop-up Menu options** Snapshot uses the gl command **fullscrn()** which has some “humorous” side effects. One of them is that unless the cursor is on top of the actual window for the graphics program (in this case, the *snapshot* button window), pressing **right** mouse will NOT bring up that program’s menu. Thus, to access the pop-up menu options, you must always bring the cursor back on top of the *snapshot* button window before pressing **right** mouse to access *snapshot*’s pop-menu. The pop-up menu currently has five items defined:

- The first item reads **Save scrn as snap.rgb** if you have just started up *snapshot* and have not yet swept out a rubberband. This will create an image file of the entire console screen (notice that at this point there is a red rubberband that encloses the entire console screen). Or else it will read **Save as snap.rgb** indicating that a rubberband area of interest currently exists.
- The second item--**New file name**--will throw up a squat rudimentary textport prompting you to input a new output image file name. If, after having called up the textport, you decide you don’t want to change the output image file name, simply pressing carriage with an empty string will exit the textport and not change the filename.
- The third item--**Ipaste snap.rgb**--allows you to paste up the image you have most recently made. Notice that after you have swept out some sub-section of the screen with the red rubberband, but before you have yet selected **Save as snap.rgb**, the **Ipaste** entry shows up as a grey color instead of the solid black of the other menu items. This is because you have not yet created the actual image file-- hence there is nothing for ipaste to lock on to out in the IRIS universe. Once you have chosen **Save as snap.rgb**, then when you pop-up the menu again, you will see that the **Ipaste** menu item is now solid black indicating that ipaste now has a fix on the currently saved image file

you have created. The same thing will happen after you have selected **New file name** but before you save an image into it. Notice that ipaste(1G) now recognizes the **Esc** key as a short-cut to closing the ipaste image window. This is especially useful when ipaste is called with the **-n** option--as *snapshot* uses it--since there is no border to specify a call to exit from.

- The fourth item--**Redraw Rubberband**--will redraw the rubberband. This is usefull for when something else erases the rubberband.
- The fifth item--**Save and Exit**--will save whatever you currently have selected, and then exit the program.
- The sixth item--**Exit**--will simply exit the program without saving anything that may be currently defined to be *snapshot*ed.

NOTES

There is a window constraint that affects *ipaste*(*1G*) which users of *snapshot* will run into: under 4Dwm, the minimum *ipaste* window width is now constrained to be 88 pixels. See the *ipaste*(*1G*) man page for more details about why this is so. On machines having less than 24 bits available for RGB display, the displayed image may appeared to have lost some quality. This occurs when the image that was saved was in colormap mode. The pixel color index is expanded into the full RGB information. When redisplayed, the RGB information is dithered on these machines to approximate the original image. When you have selected the **Save ... as ...** pop-up menu item and *snapshot* is busy reading pixels, the cursor will change to an hourglass until this proces s is finished. Another visual cue (in case you move the cursor elsewhere and let go of the input focus) is that the word “Snapshot” that is written on top of the *snapshot* button window--which is normally WHITE--turns to RED for the duration of the pixel reading/image file building sequence. It reverts to WHITE when the image file is completed. The **-b** option includes a bell-ringing audio cue which will then ringbell with a short duration upon completion of every **Save** operation. The text string “snapshot” which appears in the *snapshot* button window will always turn RED when an image file is being created, and return to WHITE when finished, but the **-b** ring-the-bell option was included for those wishing to be more forcefully appraised that *snapshot* is ready for more input action. Regarding what is actually saved into your image file, the pixels that are underneath the red rubberband are NOT grabbed by *snapshot*. This means that where specific pixel boundaries are critical, you must be sure that what you want to make into an image file is exactly inside the red rubberband--but not underneath these red border lines. The one exception to this is when the program is first invoked. As mentioned above, *snapshot* starts up with the default red rubberband set to the full console screen. In this case, if you select **Save scrn as snap.rgb**, the red rubberband will first disappear, then an image file of size XMAXSCREEN by YMAXSCREEN will be created, and finally the red rubberband will reappear.

BUGS

It is possible under extreme circumstances to get fragments of the red outline to remain on the screen. If this happens, place the red rubberband over the fragments and then move the rubberband again. *snapshot* makes use of the fullscrn() GL command which, as the Reference Manual warns, must be used “with caution or a sense of humor.” In this case, caution is advised: when wishing to access the pop-up menu, not only must your cursor be moved back on top of the *snapshot* button window, but to work as intended, you must release whichever key on the keyboard you have been holding down to maintain the input focus while the cursor has been outside of this button window. Not releasing said keyboard button will produce “humor[ous]” results when playing with the pop-up menu. Another side effect of using fullscrn() while drawing the rubberband in the overlay or popup planes is collision with other utilities also using the overlay or popup planes. To restore the snaphot rubberband, select “Redraw Rubberband” from the popup menu. *snapshot* is not yet smart enough to make sure there is enough free space on the disk partition from where *snapshot* itself was originally executed, before it blindly goes off and attempts to allocate enough memory to build an image file of the area you specify. Hence, if you find that an image that you paste up on the screen looks “funny”, run DF(1) to first confirm that the disk partition that *snapshot* is running on has not had all of its “avail” space used up.

NAME

sub - subtract two images

SYNOPSIS

sub inimage1 inimage2 outimage

DESCRIPTION

sub subtracts two images. This caculates 128+(inimage2-inimage1).

SEE ALSO

add(6D)

NAME

subimg - extract a sub-region from an image

SYNOPSIS

subimg inimage outimage x1 x2 y1 y2

DESCRIPTION

subimg extracts a region from an image. The region to be extracted is specified by x1 x2, y1 and y2. These coordinates are relative to the bottom left corner of the image. Negative values may be used to give coordinates from the upper right corner. To extract an image inset 10 pixels from a source image use coordinates 10 -10 10 -10.

NAME

thresh - threshold one image with another

SYNOPSIS

thresh inimage outimage threshimage

DESCRIPTION

thresh thresholds one image with another. The threshimage is repeated to tile the entire surface of the input image. Whenever the input image is greater than the threshimage, the output pixel is made 255 otherwise the output pixel is made 0. This can be used to halftone or dither images.

SEE ALSO

dotgen(6D) imgexp(6D)

NAME

tile - repeats an image in two dimensions

SYNOPSIS

tile inimage outimage xsize ysize

DESCRIPTION

tile repeats an image in two dimensions. An output image that is xsize by ysize pixels is created by repeating the input image.

SEE ALSO

assemble(6D)

NAME

toalias - Convert an IRIS image to an Alias image

SYNOPSIS

toalias inimage.rgb aliasimage

DESCRIPTION

toalias converts an IRIS image file to an Alias image format.

SEE ALSO

fromalias(6D)

NAME

toascii - use text characters to represent an image

SYNOPSIS

toascii inimage.bw

DESCRIPTION

toascii prints textual characters that represent the black and white image used as input. Output is sent to stdout. In order to better view the results, it is recommended that the size of the black and white input image be somewhere in the range of 50 to 200 in x, and to use a wsh that is of a point size around 4 or 5 and an width of at least 120.

NAME

tobin - Convert an Iris image to binary dump of pixel data

SYNOPSIS

tobin inimage.rgb out.bin

DESCRIPTION

tobin converts an Iris image file to a binary dump of pixel data. If the input image is a single channel image, the first byte written to the output file will be the lower left pixel. For multi-channel images, each channel is written in succession.

SEE ALSO

frombin(6D)

NAME

tobw - convert a color image to black and white

SYNOPSIS

tobw *colorimage bwimage*

DESCRIPTION

tobw converts a stored color image to black and white and saves it. The color file must already exist as *colorimage*; the black and white file will be named *bwimage*.

NAME
togif - convert an IRIS image to a Compuserve GIF image

SYNOPSIS
togif inimage.rgb outimage.gif

DESCRIPTION
togif converts an IRIS image file to a Compuserve GIF image file format.

SEE ALSO
fromgif(6D)

NAME

tomac - convert an IRIS image to MacPaint format

SYNOPSIS

tomac inimage.bw outimage.mac

DESCRIPTION

tomac converts an IRIS image file to a MacPaint image file format.

SEE ALSO

frommac(6D)

NAME

tonews - convert an IRIS image into NeWS format

SYNOPSIS

tonews irisimage outfile.im8

DESCRIPTION

tonews converts an IRIS image file to NeWS format.

NAME

topict - convert an IRIS image to Macintosh PICT format

SYNOPSIS

topict inimage out.pict

DESCRIPTION

topict converts an IRIS image file into Macintosh PICT format.

NAME

toppm - convert an IRIS image file into Jef Poskanzer’s ppm image format

SYNOPSIS

toppm inimage.rgb outimage.ppm

DESCRIPTION

toppm converts an IRIS image file into Jef Poskanzer’s ppm image file format. PBMPLUS, is by Jef Poskanzer. It is a comprehensive format conversion and image manipulation package. The latest version is always available via anonymous FTP as expo.lcs.mit.edu:contrib/pbmplus.tar.Z and ftp.ee.lbl.gov:pbmplus.tar.Z

SEE ALSO

frompppm(6D)

NAME

tops - Convert an iris image to PostScript

SYNOPSIS

tops inimage [-l screendensity] [-a screenangle] [-p pixelsperinch] [-x xpixelsperinch] [-y ypixelsperinch] [-b bitsperpixel] [-B] [-t scaletrim] [-m maxxinches maxyinches] [-h] [-o xorg yorg] [-rgb] [-RGB] [-cmyk] [-CMYK] [-eps] [-I]

DESCRIPTION

tops converts an IRIS image file into Postscript. This program can generate black and white or color PostScript. It can create binary as well as ASCII PostScript. It also can generate encapsulated PostScript with a preview image. The **-l** specifies the halftone screen density to use in the output image. The default is a 40 line per inch screen. The **-a** option specifies the screen angle to use in the printed image. The default is a 45 degree screen. The **-p** option specifies how many pixels per inch the image printed at. For instance, the IRIS screen pixel density is 98 pixels per inch. If you want to print a part of the screen at actual size use an option **-p 98.0**. The **-x** and **-y** options let you specify the x and y pixel densities independently. The **-b** option specifies how many bits to use when describing images. The valid options are 1, 2, 4, and 8. The default is 8 bits per pixel giving 256 shades in a black and white image. If the **-B** option is given, binary PostScript for the image is generated. This makes the output file about half as big, but you should only use this option if you know that the printer you are using can handle binary PostScript data. The **-t** option allows you to give a value that is used to scale the x and y pixel densities given above. Values less than 1.0 will print a smaller image. Normally, tops scales the image to fit a 8.5 by 11.0 inch page. The **-m** option option lets you describe how large the page is. Normally, tops uses a standard dot screen. The **-h** option generates PostScript that uses a line screen instead. The **-o** option allows you to specify an origin for the image. Normally, tops generates black and white PostScript. If the **-rgb** or the **-RGB** option is given, then rgb color PostScript will be generated. The only difference between these two options is whether 1 function is used to read the image data, or 3 functions are used. If the **-cmyk** or the **-CMYK** option is given, then cmyk color PostScript will be generated. The only difference between these two options is whether 1 function is used to read the image data, or 4 functions are used. The **-eps** option will cause Encapsulated PostScript to be generated, with a preview bitmap. This option should be used if you want to include the PostScript in a document. The **-I** option will generate an Encapsulated PostScript image with a preview bitmap, but only a low resolution version of the input image is saved. This file can then be included into a document for position only. Later the low resolution image can be replaced with high resolution image data.

NAME

toscitex - Convert IRIS images into Scitex CT2T images

SYNOPSIS

toscitex imagefiles

DESCRIPTION

toscitex converts a bunch of IRIS image files into Scitex CT2T images on the cartridge tape.

NAME

tosun - convert an IRIS image to a sun raster file

SYNOPSIS

tosun

DESCRIPTION

tosun converts an IRIS image file into a sun raster file format. This creates a 24-bit SUN rasterfile.

SEE ALSO

fromsun(6D)

NAME

totarga - Convert from an IRIS image to a type 2 (RGB) targa image

SYNOPSIS

totarga inimage.rgb outimage.tga

DESCRIPTION

totarga converts from an IRIS image to a type 2 (RGB) targa image. Most targa images are displayed directly on monitors with no gamma correction. The typical gamma is about 2.2, so you need to gammawarp the input image by 0.454545 to get it out of the linear intensity space, and into monitor space before using totarga.

SEE ALSO

gammawarp(6D), fromtarga(6D)

NAME

toyuv - convert an IRIS image to yuv format

SYNOPSIS

toyuv inimage.rgb outimage.yuv

DESCRIPTION

toyuv converts an IRIS image to yuv format so it can be sent to an Abekas digital video frame store. This program normally expects an IRIS image that is 720 by 486 pixels. This image should have non-square pixels. If you want to use square pixels, provide toyuv with an IRIS image that is 640 by 486 pixels and use the “-s” option.

SEE ALSO

fromyuv(6D)

NAME

verbatim - force an image to be stored without run length encoding

SYNOPSIS

/usr/sbin/verbatim inimage outimage

DESCRIPTION

verbatim converts an image to be stored with no compression. Usually images are stored using run length encoding in rle format.

SEE ALSO

iset(6D), istat(6D), rle(6D)

NAME

vhist - display a 3-D volume histogram of a color image

SYNOPSIS

vhist inimage.rgb

DESCRIPTION

vhist Displays the histogram of a color image using a 3 dimensional representation. The rgb colors of the pixels in the image are mapped into x, y and z directions inside a unit cube. This cubical representation can be rotated interactively.

SEE ALSO

hist(6D), histeq(6D), imgexp(6D)

NAME

xzoom - magnify or minify an image in the x direction

SYNOPSIS

xzoom inimage outimage xscale yscale [-i -b -t -m or -q] [-w blurfactor]

DESCRIPTION

xzoom magnifies or minifies an image in the x direction with or without filtering (xscale and yscale are floating point values). The filtering method is one pass, uses 1-d convolution, and is optimized by integer arithmetic and precomputation of filter coefficients. Normally xzoom uses a triangle filter kernel in the x direction. The *-i* (impulse) option causes xzoom to do no filtering as the image is resized. The *-b* (box) option causes xzoom to use a box as the filter kernel. The *-t* (triangle) option is the default. The *-m* (mitchell) option uses a cubic filter kernel. The *-q* (quadratic) indicates that a quadradic function should be used as the filter kernel.

SEE ALSO

izoom(6D)