Raster Images

A **raster image** (a.k.a. picture) is a rectangular array of pixels.

The dimensions of the array are the **resolution** (number of pixels horizontally and vertically) and **depth** (bits per pixel).

Depth is controlled by the number of channels (e.g. R,G,B) and the number of bits per channel.

Resolution: two ways to specify

- sometimes measured in pixels per inch
- sometimes measured in total number of pixels

but

- given pixels-per-inch and size in inches, you could figure out #pixels,
- and given #pixels and size in inches, you could figure out pixels-per-inch.

Pixel Formats

Pixel formats are usually dictated by hardware (input devices such as scanners, and output devices such as monitors and printers).

Common pixel formats:

- **bitmap** (1 bit per pixel, black or white), no colormap
  - *O.K. for text.*
- **4 or 8 bit colormapped**
  - colormap takes 4 or 8 bits in, produces 24 bits out (8 red, 8 green, 8 blue)
  - *Good for grayscale images, OK for colorful images if colormap is chosen carefully.*
- **24 bit RGB** with optional colormap
  - pixel value is segregated into 8 bits red, 8 bits green, 8 bits blue
  - colormap (if any) is split into three lookup tables, each of which takes 8 bits in and produces 8 bits out. Colormap is typically a linear or power-law ramp.
  - *Good for any purpose; with* 2^{24}*=16,777,216 colors you can display almost anything.*
- **15 bit RGB** with optional colormap
  - similar to 24 bit RGB, but pixel value is 5 red + 5 green + 5 blue
  - *Marginal on grayscale & synthesized images, OK on scanned (noisy) images.*

Colors in X Windows

X Windows has its own (bizarre) terminology for pixel formats:

X Windows term	**Meaning**
PseudoColor	colormapped with color display (typically 4 or 8 bit)
GrayScale	colormapped with monochrome display (typically 4 or 8 bit)
TrueColor, DirectColor	RGB (typically 15 or 24 bit)

Colormaps and Window Systems

If no window system (old style) then colormap is completely under programmer control.

With X windows, your **program must cooperate with window system in its use of the colormap**, or windows, borders, and cursors could disappear. X windows reserves some colors (the number varies depending on which window manager you use).

There is one hardware colormap loaded into the frame buffer at any time.

Each window has its own software colormap, which is loaded into the hardware colormap when the input focus (usually controlled by the cursor position) changes windows. If the software colormaps differ, this can make other windows appear funny.

Other window systems do things differently.

Simple Image Quantization

Problem: display a picture with N colors using only K colors.

To make things concrete, suppose we want to display an RGB picture on an 8 bit colormapped frame buffer, so $N=2^{24}$, $K=2^8$

Simple approach: **uniform quantization**

We could allocate 3 bits for red, 3 for green, 2 for blue and do the quantization like so:

```
typedef unsigned char uchar;
uchar quantize(uchar r, uchar g, uchar b)
    {return (r&0xe0) | (g&0xe0)>>3 | (b&0xc0)>>6;}
```

bit layout: RRRGGGBB

This doesn't display smooth grayscale ramps well (you get only 8 levels each of red & green, 4 levels of blue), causing severe ***contouring*** *artifacts for some pictures. For most pictures, many colors in the colormap are little used.*

Better Image Quantization

Simple approach: **adaptive quantization**

Choose a colormap depending on the use of colors (color histogram) of each picture.

Advantage: all colors of colormap are used, less contouring.

Disadvantages: each picture has a different colormap; it takes time to find the best colormap, and to do the quantization.

Median Cut Algorithm:

1. compute color histogram of picture
2. do clustering of color histogram (in RGB space) to pick a colormap of K colors

   goal: minimize sum of squared distances between each pixel and its nearest representative, have each color used roughly an equal number of times.
3. map each pixel of picture to its nearest ‘‘representative’’ in colormap

A variant of this algorithm is used by the X Windows “xv” program.

Some Picture File Formats

JPEG: Joint Photographics Experts Group format: lossy compression.

TIFF: Tagged-Image File Format: run-length encoding with compression.

GIF: CompuServe Graphics Interchange Format: uses compression.

PPM: Portable PixMap: a very simple format, no compression.

EPS: Encapsulated PostScript: grayscale or color, usually ASCII and no compression, can be mix of raster and geometric data.

	BITS PER PIXEL	FILE SIZE	COMMENTS
JPEG	24,?	small	lossy, good for archives
TIFF	8,24,?	medium	good
GIF	1,4,8,?	medium	no good for colorful images
PPM	24	big	easy to read/write
EPS	1,2,4,8,24,?	huge	good for printing

PPM Picture File Format

The “raw” (binary) form of PPM file format has an ASCII header followed by three bytes per pixel:

```
P6 <ws> <width> <ws> <height> <ws> <maxval>
    <ws1> <pixels>
```

where

`<ws>` = whitespace: spaces, tabs, newline, or a ‘#’ and everything up to the next newline

`<ws1>` = a single space, tab, or newline

`<maxval>` = maximum channel value

`<pixels>` = RGBRGBRGB... where each R, G, B, is a single byte

Example: a 24-bit picture with width=300 and height=200 might begin:

```
P6\n# this is a comment\n300 200 255\n...
```

On Andrew, see “`man -P /usr/contributed/man ppm libppm`” or “`rtfm ppm`” for more info on the file format and an I/O library for it.