PPM Images

Representations of image data

- Images (e.g. digital photos) consist of a rectangular array of discrete picture elements called *pixels.*
- An image consisting of 200 rows of 300 pixels per row contains 200 x 300 = 60,000 individual pixels.
- The Portable PixMap *(.ppm)* format is a particularly simple way used to encode a rectangular image (picture) as uncompressed data file.
- The *.ppm* file can viewed with a number of tools including *xv* and *gimp.*
- Other well known formats include JPEG (.jpg), TIFF (.tif), GIF (.gif), and PNG (.png)

We will use *.ppm* to represent two types of images

Grayscale images –

- correspond to black / white photographs
- each pixel consists of 1 byte which should be represented as *unsigned char*
- a value of 0 is solid black
- a value of 255 is bright white
- intermediate are “gray” values of increasing brightness.

Color images -

- correspond to color photographs
- each pixel consists of 3 bytes with *each byte* represented by an *unsigned char*
- this format is call RGB three bytes represent the
    - red component
    - green component
    - blue component
- when *red* == *green* == *blue* a grayscale “color” is produced
- (255, 255, 255) is bright white

Colors are additive

- (255, 255, 0) = red + green = bright yellow
- (255, 0, 255) = red + blue = magenta (purple)
- (0, 255, 255) = blue + green = cyan (turquoise)
- (255, 255, 255) = red + green + blue = white

PPM file structure

ppm header
packed image data

The *.ppm* header

```
P6
# This is a comment
600 400 255
```

- The P6 indicates this is a color image (P5 means grayscale)
- The width of the image is 600 pixels
- The height of the image is 400 pixels
- The 255 is the maximum value of a pixel.
- Following the 255 is a \n (0x0a) character.

The image data

- The red component of the upper left pixel must *immediatly follow* the new line.
- There must be a total of 3 x 600 x 400 = 720,000 bytes of data.

Building a .ppm file

```
/* p12.c */

/* Construct a solid color image of the specified color */

/* Input arguments                                      */
/*     width  in pixels                                 */
/*     height in pixels                                 */
/*     red value                                        */
/*     green value                                      */
/*     blue value                                       */

#include <stdio.h>

int main(
{
   int width;
   int height;
   int count = 0;

   unsigned int red;
   unsigned int green;
   unsigned int blue;

   fscanf(stdin, "%d", &width);
   fscanf(stdin, "%d", &height);

   fscanf(stdin, "%d", &red);
   fscanf(stdin, "%d", &green);
   fscanf(stdin, "%d", &blue);

   fprintf(stdout, "P6\n");
   fprintf(stdout, "%d %d 255\n", width, height);

   while (count < (width * height))
   {
      fprintf(stdout, "%c%c%c", red, green, blue);
      count = count + 1;
   }
   return(0);
}
```

%c format prevents data conversion
%c format generates 1 byte of output
no blank spaces permitted in format spring

Sample input file

```
class/101/examples ==> cat pic1.in
200 150
200  64 224
```

Running the program with input and output redirection

```
 p12 < pic1.in > pic1.ppm
```

Viewing the data with a hexadecimal / ASCII dump utility

```
class/101/examples ==> hexdump < pic1.ppm | more

        0 - 50 36 0a 32 30 30 20 31 35 30 20 32 35 35 0a c8
            P  6     2  0  0     1  5  0     2  5  5
       10 - 40 e0 c8 40 e0 c8 40 e0 c8 40 e0 c8 40 e0 c8 40
            @        @        @        @        @        @
       20 - e0 c8 40 e0 c8 40 e0 c8 40 e0 c8 40 e0 c8 40 e0
                  @        @        @        @        @
       30 - c8 40 e0 c8 40 e0 c8 40 e0 c8 40 e0 c8 40 e0 c8
               @        @        @        @        @
```

Note that

200 base 10 is c8 base 16

64 base 10 is 40 base 16 (and is the ASCII code for the @ character)

224 base 10 is E0 base 16

The *ls -l* command command can be used to show the size of the entire file

class/101/examples ==> ls -l pic1.ppm
-rw------- 1 westall iiimd 90015 Sep 13 13:03 pic1.ppm

15 (header length)
3x 200 x 150 = 90000 (image data)
90015

This is the image: