Pseudo-random ramblings about programming and other geeky stuff

Thursday, 7 March 2013

Low-level Graphics on Raspberry Pi [part five]

So far we have used some fixed palette values for the colors (see part four) - namely 0-15 to draw the 16 bars. These happen to be the default Linux framebuffer colors (slight variations between distributions to be expected). Since we are using palette indexes, we don't really know what are the actual colors output. The Linux framebuffer interface provides a function `FBIOGETCMAP` to read the color palette:

```
// Get palette information
unsigned short r[256];
unsigned short g[256];
unsigned short b[256];
unsigned short a[256];
memset(r, 0, 256 * sizeof(unsigned short));
memset(g, 0, 256 * sizeof(unsigned short));
memset(b, 0, 256 * sizeof(unsigned short));
memset(a, 0, 256 * sizeof(unsigned short));
struct fb_cmap pal;
pal.start = 0;
pal.len = 0;
pal.red = r;
pal.green = g;
pal.blue = b;
pal.transp = a;
if (ioctl(fbfd, FBIOGETCMAP, &pal)) {
  printf("Error reading palette.\n");
}
```

...would give us the red, green and blue components of the colors in the palette. So most likely `r[0]=0, g[0]=0, b[0]=0` for the color index 0 i.e. black (#000000 in HTML) - `r[1]=0, g[1]=0, b[1]=255` for color index 1 i.e. blue (#0000FF). Colors from index 16 onwards are most likely all black.

Unfortunately reading this information is not supported by the RPi framebuffer driver ...this surely is not how I wanted these posts to be Raspberry Pi specific :(

After a bit of trial and error (and 'stealing' the initial values from another Linux system using the above code), I managed to get pretty close to the RPi default colors. In this example we initialise some hard-coded values (note that I use the possibly more familiar range 0-255, matching many systems like the HTML color scheme) for the colors (the enum is not used here but might prove handy later on - think of `put_pixel(x, y, RED)`), set them to the system palette starting at index 16 (i.e. just after the default colors) and then draw the familiar color bars at the top half of the screen - and bars with same colors but using different palette indexes at the lower half for comparison:

```
...
// after includes
```


Blog Archive

- ► 2016 (6)
- ► 2015 (3)
- ► 2014 (9)
- ▼ 2013 (9)
  - ► April (2)
  - ▼ March (4)
    - Low-level Graphics on Raspberry Pi (part six)
    - Low-level Graphics on Raspberry Pi (part five)
    - Coding Gold Dust: How to break out from an infinit...
    - Low-level Graphics on Raspberry Pi (part four)
  - ► February (1)
  - ► January (2)
- ► 2012 (2)

Code Repository

- Low-level Graphics on RPi

Discussion

- Low-level Graphics on RPi
- Python Programming on RPi
- Java Programming on RPi

Links

- Raspberry Pi
- Python

```
// default framebuffer palette
typedef enum {
  BLACK        =  0, /*   0,   0,   0 */
  BLUE         =  1, /*   0,   0, 172 */
  GREEN        =  2, /*   0, 172,   0 */
  CYAN         =  3, /*   0, 172, 172 */
  RED          =  4, /* 172,   0,   0 */
  PURPLE       =  5, /* 172,   0, 172 */
  ORANGE       =  6, /* 172,  84,   0 */
  LTGREY       =  7, /* 172, 172, 172 */
  GREY         =  8, /*  84,  84,  84 */
  LIGHT_BLUE   =  9, /*  84,  84, 255 */
  LIGHT_GREEN  = 10, /*  84, 255,  84 */
  LIGHT_CYAN   = 11, /*  84, 255, 255 */
  LIGHT_RED    = 12, /* 255,  84,  84 */
  LIGHT_PURPLE = 13, /* 255,  84, 255 */
  YELLOW       = 14, /* 255, 255,  84 */
  WHITE        = 15  /* 255, 255, 255 */
} COLOR_INDEX_T;

static unsigned short def_r[] =
    { 0,   0,   0,   0, 172, 172, 172, 172,
     84,  84,  84,  84, 255, 255, 255, 255};
static unsigned short def_g[] =
    { 0,   0, 172, 172,   0,   0,  84, 172,
     84,  84, 255, 255,  84,  84, 255, 255};
static unsigned short def_b[] =
    { 0, 172,   0, 172,   0, 172,   0, 172,
     84, 255,  84, 255,  84, 255,  84, 255};

...

void draw() {

    int x, y;

    for (y = 0; y < (vinfo.yres / 2); y++) {
        for (x = 0; x < vinfo.xres; x++) {

            // color based on the 16th of the screen width
            int c = 16 * x / vinfo.xres;

            // default colors at upper half
            put_pixel(x, y, c);
            // our own colors at lower half
            put_pixel(x, y + (vinfo.yres / 2), c + 16);

        }
    }

}

...
// in main after opening the fb device
    // Set palette
    unsigned short r[256];
    unsigned short g[256];
    unsigned short b[256];
    memset(&r, 0, 256); // initialise with zeros
    memset(&g, 0, 256);
    memset(&b, 0, 256);
    int i;
    for(i = 0; i < 16; i++) {
        // copy the hard-coded values
        // note that Linux provides more precision (0-65535),
        // so we multiply ours (0-255) by 256
        r[i] = def_r[i] << 8;
        g[i] = def_g[i] << 8;
        b[i] = def_b[i] << 8;
    }
    struct fb_cmap pal;
    pal.start = 16; // start our colors after the default 16
    pal.len = 256; // kludge to force bcm fb drv to commit palette...
    pal.red = r;
    pal.green = g;
    pal.blue = b;
```

```
    pal.transp = 0; // we want all colors non-transparent == null
    if (ioctl(fbfd, FBIOPUTCMAP, &pal)) {
        printf("Error setting palette.\n");
    }

...
```

...I only hope my monitor color tolerance is about ok and your eyes not much sharper than mine ;)

Now you could go and fill the palette with any colors you can come up with and draw something interesting. If you use the first 16 color 'slots' in the palette for your own colors, remember to set the default colors at the end of your program:

```
  // before closing the fbfd
  // reset palette
  palette.start = 0;
  palette.len = 256;
  palette.red = def_red;
  palette.green = def_green;
  palette.blue = def_blue;
  palette.transp = 0;
  if (ioctl(fbfd, FBIOPUTCMAP, &palette)) {
    printf("Error setting palette.\n");
  }
```

...otherwise when returning to the console, you might not be able to read any of the text ;) In case this happens (or some other issue with the graphics garbles the screen), one way to fix thing is to type (possibly 'blindly') the commands:

```
fbset -depth 32
fbset -depth 16
```

...this (changes the color depth and resets back to the RPi default and) should clear refresh the screen.

[Continued in part six]

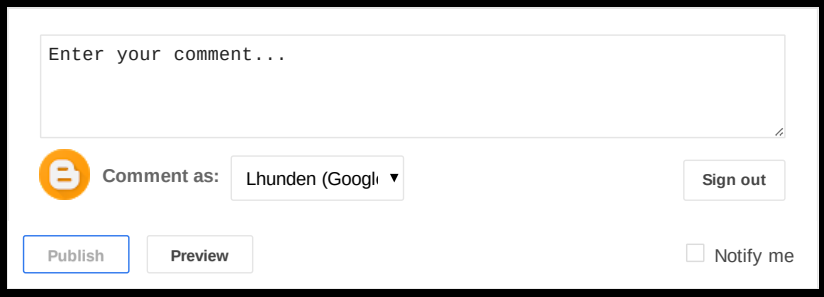
Posted by Unknown at 15:08

Labels: C, graphics, Linux, Raspberry Pi

No comments:

Post a Comment

Note: only a member of this blog may post a comment.



Newer Post Home Older Post

Subscribe to: Post Comments (Atom)