Pseudo-random ramblings about programming and other geeky stuff

Sunday, 3 March 2013

Low-level Graphics on Raspberry Pi [part four]

In the part three we saw how to plot individual pixels in the framebuffer. Now let's turn the plot-pixel code into a reusable function.

First we need to move some of the variables outside of the `main` function, so we can access them in the new function - then we just move the pixel plotting code from the main into the new function and make `main` to call it. We will also move the 'draw' code into another function to separate it from the main and make it easier to read.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <linux/fb.h>
#include <sys/mman.h>

// 'global' variables to store screen info
char *fbp = 0;
struct fb_var_screeninfo vinfo;
struct fb_fix_screeninfo finfo;

// helper function to 'plot' a pixel in given color
void put_pixel(int x, int y, int c)
{
    // calculate the pixel's byte offset inside the buffer
    unsigned int pix_offset = x + y * finfo.line_length;

    // now this is about the same as 'fbp[pix_offset] = value'
    *((char*)(fbp + pix_offset)) = c;

}

// helper function for drawing - no more need to go mess with
// the main function when just want to change what to draw...
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;

            // call the helper function
            put_pixel(x, y, c);
```

```
        }
    }

}

// application entry point
int main(int argc, char* argv[])
{

    int fbfd = 0;
    struct fb_var_screeninfo orig_vinfo;
    long int screensize = 0;


    // Open the file for reading and writing
    fbfd = open("/dev/fb0", O_RDWR);
    if (!fbfd) {
      printf("Error: cannot open framebuffer device.\n");
      return(1);
    }
    printf("The framebuffer device was opened successfully.\n");

    // Get variable screen information
    if (ioctl(fbfd, FBIOGET_VSCREENINFO, &vinfo)) {
      printf("Error reading variable information.\n");
    }
    printf("Original %dx%d, %dbpp\n", vinfo.xres, vinfo.yres,
       vinfo.bits_per_pixel );

    // Store for reset (copy vinfo to vinfo_orig)
    memcpy(&orig_vinfo, &vinfo, sizeof(struct fb_var_screeninfo));

    // Change variable info
    vinfo.bits_per_pixel = 8;
    if (ioctl(fbfd, FBIOPUT_VSCREENINFO, &vinfo)) {
      printf("Error setting variable information.\n");
    }

    // Get fixed screen information
    if (ioctl(fbfd, FBIOGET_FSCREENINFO, &finfo)) {
      printf("Error reading fixed information.\n");
    }

    // map fb to user mem
    screensize = vinfo.xres * vinfo.yres;
    fbp = (char*)mmap(0,
              screensize,
              PROT_READ | PROT_WRITE,
              MAP_SHARED,
              fbfd,
              0);

    if ((int)fbp == -1) {
        printf("Failed to mmap.\n");
    }
    else {
        // draw...
        draw();
        sleep(5);
    }

    // cleanup
    munmap(fbp, screensize);
    if (ioctl(fbfd, FBIOPUT_VSCREENINFO, &orig_vinfo)) {
        printf("Error re-setting variable information.\n");
    }
    close(fbfd);

    return 0;

}
```

Now save the file as `fbtest4.c`, compile with `make  fbtest4.c` and execute `./fbtest4` - you should see the same vertical color bars at the upper half of the screen

as in part three. Making the `put_pixel` function should help to come up with new ideas for images to draw...

[Continued in part five]

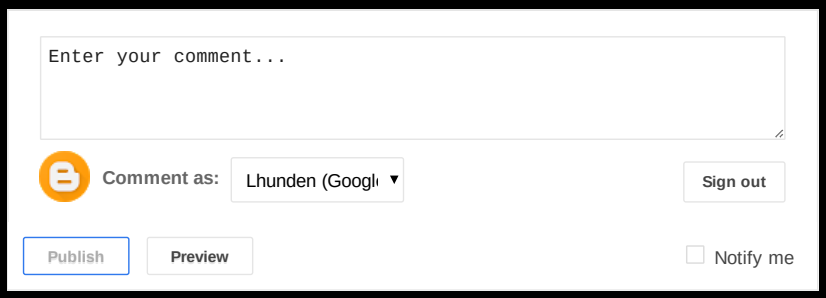
Posted by Unknown at 14:11

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)