# Raspberry Compote

Pseudo-random ramblings about programming and other geeky stuff

Friday, 21 March 2014

## Low-level Graphics on Raspberry Pi (part X+2)

In the previous part we did some animation with page flipping ...requiring a bit ugly 'incantation' outside of the application code.

It is worth noting that not all animation or screen effects require page flipping (or v-syncing). There is no need for such for example if the change between 'frames' is small enough - like updating textual or numerical indicators - or the change is stochastic enough - like for example the legendary (at least back in the day in early-mid 1990's) fire effect.

The fire effect might in fact work as a good example ...also admittedly I love it myself and cannot wait to show it off ;) I wrote my first version of the code for a 386sx PC with a plain vanilla VGA card - tweaked into 320x240 pixels 256 color 'Mode X' somewhere around 1993 after seeing the effect on one university course mate's Amiga. Part of the demoscene ethos and big part of the fun was of course to not copy somebody else's ready-made code but to figure out the idea your self. However I am now going to be a spoiler and reveal the code to this effect (well, there are versions already floating in the internet)...

First we have to come up with a nice fiery color palette - this should give us a nice one from golden yellow, to orange, to red, to fading the red to black:

```
    // Create palette
    unsigned short r[256]; // red
    unsigned short g[256]; // green
    unsigned short b[256]; // blue
    int i;
    for (i = 0; i < 256; i++) {
        if (i < 32) {
            r[i] = i * 7 << 8;
            g[i] = b[i] = 0;
        }
        else if (i < 64) {
            r[i] = 224 << 8;
            g[i] = (i - 32) * 4 << 8;
            b[i] = 0;
        }
        else if (i < 96) {
            r[i] = 224 + (i - 64) << 8;
            g[i] = 128 + (i - 64) * 3 << 8;
            b[i] = 0;
        }
        else {
            r[i] = g[i] = 255 << 8;
            b[i] = 128 << 8;
        }
    }
    struct fb_cmap palette;
    palette.start = 0;
    palette.len = 256;
```

**Blog Archive**

- ► 2016 (6)
- ► 2015 (3)
- ▼ 2014 (9)
  - ► September (1)
  - ► April (3)
  - ▼ March (5)
    - Programming language (micro) benchmark
    - Low-level Graphics on Raspberry Pi (part X+2)
    - Low-level Graphics on Raspberry Pi (part X+1)
    - Low-level Graphics on Raspberry Pi (part X)
    - Restart
- ► 2013 (9)
- ► 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

```
    palette.red = r;
    palette.green = g;
    palette.blue = b;
    palette.transp = 0; // null == no transparency settings
    // Set palette
    if (ioctl(fbfd, FBIOPUTCMAP, &palette)) {
        printf("Error setting palette.\n");
    }
```

...and remember to restore the palette at the end of the execution! Good exercise would be to work out other nice color transitions - something like gas fire like white-yellow-blue for example.

For plotting the pixels, the algorithm has three phases: *seed*, *smooth* and *convect*. In seed phase, we 'sow the seeds for the fire' by plotting semi-randomly bright yellow pixels along the bottom line of the screen. In smooth phase, we scan through all pixels on the screen and average them with the surrounding pixels to smooth out clear color edges. And in the convect phase, we make the flames to rise up the screen slowly cooling down and fading into the background. As it is evident in the code, it is good to play around with the values for seed colors, frequency of seed pixels, smoothing factors etc:

```
        // Draw
        int maxx = var_info.xres - 1;
        int maxy = var_info.yres - 1;
        int n = 0, r, c, x, y;
        int c0, c1, c2;
        while (n++ < 200) {
            // seed
            for (x = 1; x < maxx; x++) {

                r = rand();
                c = (r % 4 == 0) ? 192 : 32;
                put_pixel(x, maxy, fbp, &var_info, c);
                if ((r % 4 == 0)) { // && (r % 3 == 0)) {
                    c = 2 * c / 3;
                    put_pixel(x - 1, maxy, fbp, &var_info, c);
                    put_pixel(x + 1, maxy, fbp, &var_info, c);
                }
            }

            // smooth
            for (y = 1; y < maxy - 1; y++) {
                for (x = 1; x < maxx; x++) {
                    c0 = get_pixel(x - 1, y, fbp, &var_info);
                    c1 = get_pixel(x, y + 1, fbp, &var_info);
                    c2 = get_pixel(x + 1, y, fbp, &var_info);
                    c = (c0 + c1 + c1 + c2) / 4;
                    put_pixel(x, y - 1, fbp, &var_info, c);
                }
            }

            // convect
            for (y = 0; y < maxy; y++) {
                for (x = 1; x < maxx; x++) {
                    c = get_pixel(x, y + 1, fbp, &var_info);
                    if (c > 0) c--;
                    put_pixel(x, y, fbp, &var_info, c);
                }
            }
        }
```

It appears that for this we need an utility function for reading the current value from a pixel on the screen:

```
// utility function to get a pixel
char get_pixel(int x, int y, void *fbp, struct fb_var_screeninfo *vinfo) {
    unsigned long offset = x + y * vinfo->xres;
    return *((char*)(fbp + offset));
}
```

And then there's the annoying blinking console cursor we need to hide...

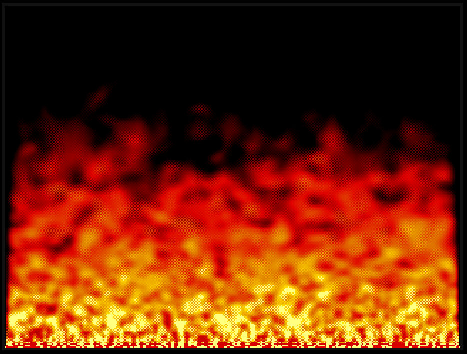
```
#include <linux/kd.h>

...
    // hide cursor
    char *kbfds = "/dev/tty";
    int kbfd = open(kbfds, O_WRONLY);
    if (kbfd >= 0) {
        ioctl(kbfd, KDSETMODE, KD_GRAPHICS);
    }
    else {
        printf("Could not open %s.\n", kbfds);
    }
```

...and remember to restore at end!

[Full source available in GitHub]

This should produce the awesome effect as seen in this screenshot (already sneak-peeked in part 2):



Starfield, shadebobs and plasma effects would not require page flipping / v-syncing and might be worth attempting... some more examples for inspiration or dwelling in the good oldskool days for example here ;)

[Continued in next part Shapes]

Posted by Unknown at 15:55

Labels: C, graphics, Linux, Raspberry Pi

## No comments:

## Post a Comment

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

Enter your comment...

Comment as: Lhunden (Googl ▾ Sign out

Publish Preview □ Notify me

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)