Pseudo-random ramblings about programming and other geeky stuff

Wednesday, 10 February 2016

Low-level Graphics on Raspberry Pi [more palette]

In a previous post we briefly looked at palette animation. Now (hopefully) a slightly more appetising example of what could be done with this technique.

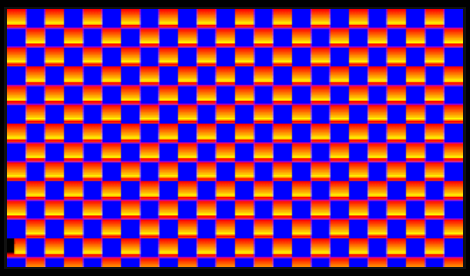
Let's draw some 'rainbow striped' blocks and customise the palette to include 16 colors sliding from red to yellow:

```
...
void draw() {
    ...
    // colored blocks
    for (y = 0; y < vinfo.yres; y += t) {
        int xoffset = y / t % 2;
        for (x = t * xoffset; x < vinfo.xres; x += t * 2) {
            int x2, y2;
            for (y2 = 0; y2 < t; y2++) {
                for (x2 = 0; x2 < t; x2++) {

                    // color based on y2 value
                    // using the custom colors (16+)
                    int c = 16 + (y2 % 16);

                    // draw pixel
                    put_pixel(x + x2, y + y2, c);

                }
            }
        }
    }
}
...
int main(int argc, char* argv[])
{
    ...

    // Set palette
    ...
    for(i = 0; i < 16; i++) {
        // red-yellow gradient
        // note that Linux provides more precision (0-65535),
        // so we multiply ours (0-255) by 256
        r[i] = 255 << 8;
        g[i] = ((15 - i) * 16) << 8;
        b[i] = 0;
    }

...
```

And then animate the palette by rotating the custom color entries:

```
...
        int j;
        int fps = 30; // frames per second
        int d = 5; // duration in seconds
        // repeat for given time
        for(j = 0; j < fps * d; j++) {
            // store color 0 in temp variables
            int rt = r[0];
            int gt = g[0];
            int bt = b[0];
            // replace colors by copying the next
            for(i = 0; i < 15; i++) {
                r[i] = r[i+1];
                g[i] = g[i+1];
                b[i] = b[i+1];
            }
            // restore last one from temp
            r[15] = rt;
            g[15] = gt;
            b[15] = bt;
            // Note that we set up the 'pal' structure earlier
            // and it still points to the r, g, b arrays,
            // so we can just reuse 'pal' here
            if (ioctl(fbfd, FBIOPUTCMAP, &pal) != 0) {
                printf("Error setting palette.\n");
            }
            usleep(1000000 / fps);
        }
...
```

Compile with `'gcc -o fbtest5y fbtest5y.c'` and run with `'./fbtest5y'`. The blue checker board with holes looks like floating on top of flowing lava waves ;)



Full code available in GitHub.

[Continued in next part Even more palette]

Posted by Unknown at 14:33

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)