
Subject: Re: Drawing raw data to an Image / Draw object?

Posted by [blueapples](#) on Thu, 09 Oct 2008 01:03:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

This is what I've got so far. There's a pretty major problem though. The RGBA structure seems to only allow a very limited color depth, namely what can fit in 3 bytes. Magick++ (and really most image formats like JPG and PNG) allow many more colors than this makes possible. To get the conversion to work I have to use a rather crude instrument, `quantumToByte()`, which converts from Magick++'s 32 bit value to a byte. This makes high color images look just terrible... is there a way to use greater color depth with `ImageBuffer`?

```
byte quantumToByte(m::Quantum q)
{
    return q / 65535 * 255;
}
```

```
class Canvas : public Ctrl {
    Image drawImg;
```

```
public:
    typedef Canvas CLASSNAME;
    Canvas();
    void Paint(Draw& draw);
    void SetImage(m::Image& newimage, int x = 0, int y = 0, int zoom = 1);
};
```

```
/* Canvas::SetImage: Copies pixles for the current view to a new
Upp::Image object to be rendered on the control. The zoom value
is a magnification multiplier - 1 is for actual size pixels, 2 for
double sized pixels, etc. */
```

```
void Canvas::SetImage(m::Image& image, int x, int y, int zoom)
{
    Rect rect = this->GetRect();
    int width = rect.Width() / zoom;
    int height = rect.Height() / zoom;
    int zx, zy;
    RGBA p;
```

```
// Constrain the viewport to the max size of the actual image
if(width > image.columns())
    width = image.columns();
if(height > image.rows())
```

```

height = image.rows();

m::PixelPacket *pixel = image.getPixels(x, y, width, height);

ImageBuffer ib(width * zoom, height * zoom);

byte i = 0;
float v;
for(int y = 0; y < height; y++) {
    //RGBA *I = ib[y];
    i = 0;
    for(int x = 0; x < width; x++) {

        if(zoom == 1) {
            ib[y * zoom][x * zoom].a = 255;
            ib[y * zoom][x * zoom].r = quantumToByte(pixel->red);
            ib[y * zoom][x * zoom].g = quantumToByte(pixel->green);
            ib[y * zoom][x * zoom].b = quantumToByte(pixel->blue);
        } else {
            // This really should use some sort of ASM box routine
            for(int zx = x * zoom; zx < (x+1) * zoom; zx++) {
                for(int zy = y * zoom; zy < (y+1) * zoom; zy++) {
                    //if(zy + y < height && zx + x < width) {
                    //p = ib[zy + y][zx + x];
                    ib[zy][zx].a = 255;
                    ib[zy][zx].r = quantumToByte(pixel->red);
                    ib[zy][zx].g = quantumToByte(pixel->green);
                    ib[zy][zx].b = quantumToByte(pixel->blue);
                    //}
                }
            }
        }

        // Go to the next pixel
        pixel++;
    }
}

Premultiply(ib);
drawImg = ib;
}

void Canvas::Paint(Draw& draw)
{
    Rect rect = this->GetRect();
    draw.DrawRect(0, 0, rect.Width(), rect.Height(), Gray());
    draw.DrawImage(0, 0, drawImg);
}

```
