Optimizing code to darken a bitmap, part 1

devblogs.microsoft.com/oldnewthing/20220307-00

March 7, 2022

Raymond Chen

I needed a function to make a 32bpp ARGB image a little darker. Here's a naïve version, which we will use as our starting point:

```
union Pixel
{
    uint8_t c[4]; // four channels: red, green, blue, alpha
    uint32_t v;   // full pixel value as a 32-bit integer
};

void darken(Pixel* first, Pixel* last, int darkness)
{
  int lightness = 256 - darkness;
  for (; first < last; ++first) {
    for (int i = 0; i < 3; ++i) {
      first->c[i] = (uint8_t)(first->c[i] * lightness / 256);
    }
  }
}
```

You call this function with a range of pixels, and an integer representing how much you want to make the image darker, on a scale from 0 (no change at all) to 256 (complete blackness). The function goes through every pixel and applies the darkening factor to each of the first three channels. (The fourth channel is the alpha channel, which should stay unchanged.)

One obvious idea for improvement is to unroll the innermost loop.

```
void darken(Pixel* first, Pixel* last, int darkness)
{
  int lightness = 256 - darkness;
  for (; first < last; ++first) {
    first->c[0] = (uint8_t)(first->c[0] * lightness / 256);
    first->c[1] = (uint8_t)(first->c[1] * lightness / 256);
    first->c[2] = (uint8_t)(first->c[2] * lightness / 256);
  }
}
```

This gives a 1.8× improvement over the original plain version.

Next time, we'll try to improve on this even further by doing the operations in parallel.

Raymond Chen

Follow