Subject: Inverse palette conversion algorithm...
Posted by mirek on Wed, 05 Apr 2006 08:11:49 GMT
View Forum Message <> Reply to Message

I have spend a 3 days optimizing this little snippet (for new Image) and I think it is quite interesting implementation, so I am posting it here so that fruits of my efforts are a little bit more public:

```
enum {
 RASTER_MAP_R = 32,
 RASTER_SHIFT_R = 3,
 RASTER_MAP_G = 64,
 RASTER_SHIFT_G = 2,
 RASTER_MAP_B = 16,
 RASTER_SHIFT_B = 4,

 RASTER_MAP_MAX = 64
};

struct PaletteCv {
 Buffer<byte> cv;

 byte *At(int r, int b)          { return cv + (r << 10) + (b << 6); }
 byte  Get(const RGBA& b) const   { return cv[(int(b.r >> RASTER_SHIFT_R) << 10) +
                                     (int(b.g >> RASTER_SHIFT_G)) +
                                     (int(b.b >> RASTER_SHIFT_B) << 6)]; }
 PaletteCv()                     { cv.Alloc(RASTER_MAP_R * RASTER_MAP_G * RASTER_MAP_B); }
};

struct sPalCv {
 PaletteCv&  cv_pal;
 const RGBA *palette;
 int         ncolors;
 bool        done[RASTER_MAP_G];
 struct Ginfo {
  int   dist;
  byte  g;
  byte  ii;
 };

 enum { BINS = 16, BINSHIFT = 11 };

 Ginfo       line[BINS][256];
 Ginfo      *eline[BINS];
 byte       *gline;

 static int Sq(int a, int b) { return (a - b) * (a - b); }
```

```
 void SetLine(int r, int b);
 int  Get(int g);

 sPalCv(const RGBA *palette, int ncolors, PaletteCv& cv_pal);
};

void sPalCv::SetLine(int r, int b)
{
 gline = cv_pal.At(r, b);
 r = 255 * r / (RASTER_MAP_R - 1);
 b = 255 * b / (RASTER_MAP_B - 1);
 for(int i = 0; i < BINS; i++)
  eline[i] = line[i];
 for(int i = 0; i < ncolors; i++) {
  int dist = Sq(palette[i].r, r) + Sq(palette[i].b, b);
  int bini = dist >> BINSHIFT;
  Ginfo *t = eline[bini >= BINS ? BINS - 1 : bini]++;
  t->dist = dist;
  t->g = palette[i].g;
  t->ii = i;
 }
 ZeroArray(done);
}

int sPalCv::Get(int g)
{
 if(done[g])
  return gline[g];
 int gg = 255 * g / (RASTER_MAP_G - 1);
 int ii = 0;
 int dist = INT_MAX;
 for(int th = 0; th < BINS; th++) {
  Ginfo *s = line[th];
  Ginfo *e = eline[th];
  while(s < e) {
   int sdist = Sq(s->g, gg) + s->dist;
   if(sdist < dist) {
    ii = s->ii;
    dist = sdist;
   }
   s++;
  }
  if(th < BINS - 1 && dist < ((th + 1) << BINSHIFT))
   break;
 }
 done[g] = true;
 gline[g] = ii;
 return ii;
```

```
}

sPalCv::sPalCv(const RGBA *palette, int ncolors, PaletteCv& cv_pal)
: cv_pal(cv_pal), ncolors(ncolors), palette(palette)
{
 byte ender[256];
 for(int b = 0; b < RASTER_MAP_B; b++) {
  ZeroArray(ender);
  for(int r = 0; r < RASTER_MAP_R; r++) {
   SetLine(r, b);
   int g = 0;
   while(g < RASTER_MAP_G) {
    int ii = Get(g);
    int eg = max<int>(g, ender[ii]);
    if(Get(eg) == ii)
     while(eg < RASTER_MAP_G - 1 && Get(eg + 1) == ii)
      eg++;
    else
     while(Get(eg) != ii)
      eg--;
    ender[ii] = eg;
    g++;
    while(g <= eg - 1) {
     gline[g] = ii;
     done[g] = true;
     g++;
    }
   }
  }
 }
}
```

(BTW, level of optimization: This is almost 20 times faster than brute-force algorithm...)

Mirek
